# Design with Microprocessors

## Lecture 9

## Motors

**Year 3 CS**
**Academic year 2023/2024**
**1st Semester**

**Lecturer: Radu Dănescu**

# Direct Current (DC) Motors

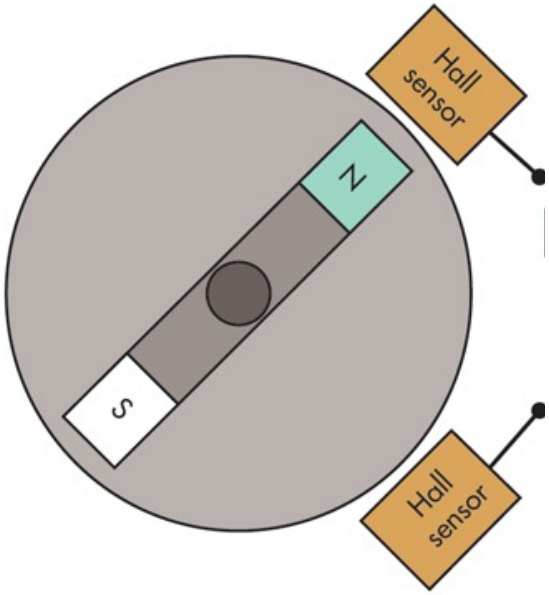
**DC Motor with reduction (gearbox)**

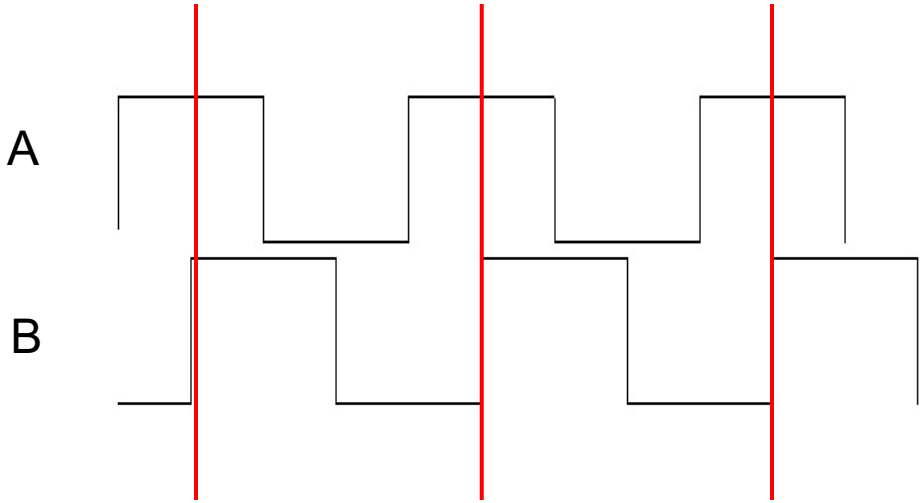
- Classic DC motor, the voltage controls the speed, and polarity controls the direction
- Continuous rotation as long as powered
- Gear box with different ratios (1:19, 1:53, 1:48, etc)
    - Increase of torque by reducing the rotation speed


# Direct Current (DC) Motors

**Measuring the rotation speed**

•Quadrature Hall sensor (magnetic)

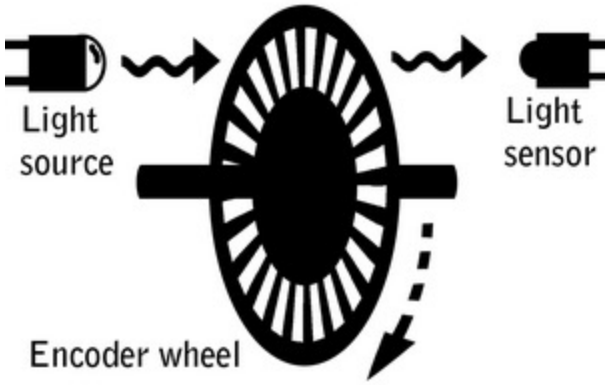



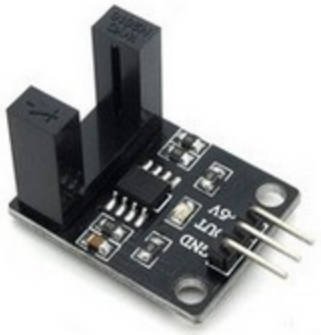

- Speed: frequency of pulses
- Sense: monitor the rising or falling edges of one signal (use it as clock)
    - The state of the other signal at the chosen front gives the sense of rotation

# Direct Current (DC) Motors

**Measuring the rotation speed**

•Perforated wheel (code wheel) + IR light sensor



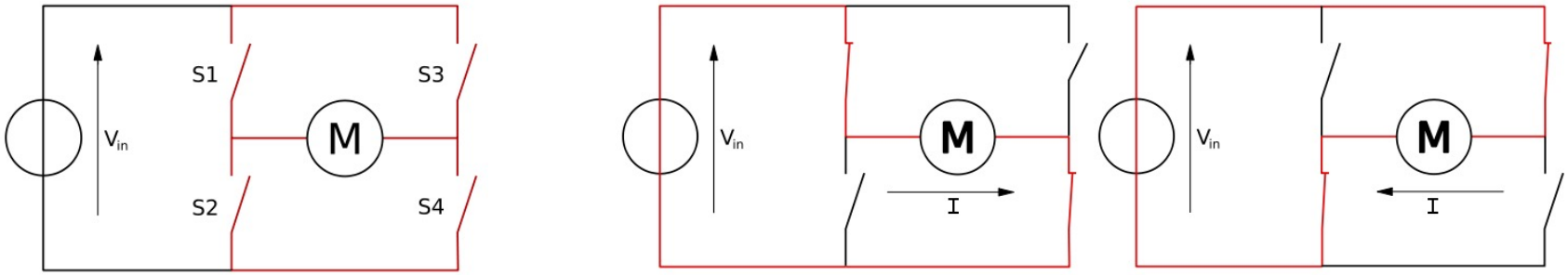



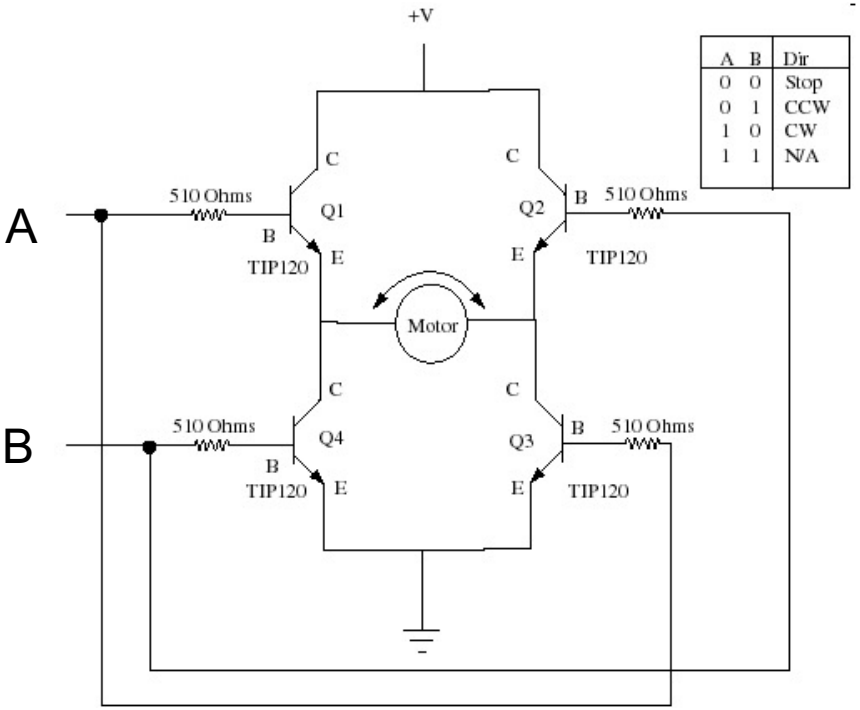
- Passing or blocking the IR beam causes a train of pulses for measuring speed.
- How can we measure the sense ?

# Direct Current (DC) Motors

## The H Bridge

•Controlling the starting/stopping and sense of rotation of a motor




| A | B | Dir |
|---|---|---|
| 0 | 0 | Stop |
| 0 | 1 | CCW |
| 1 | 0 | CW |
| 1 | 1 | N/A |

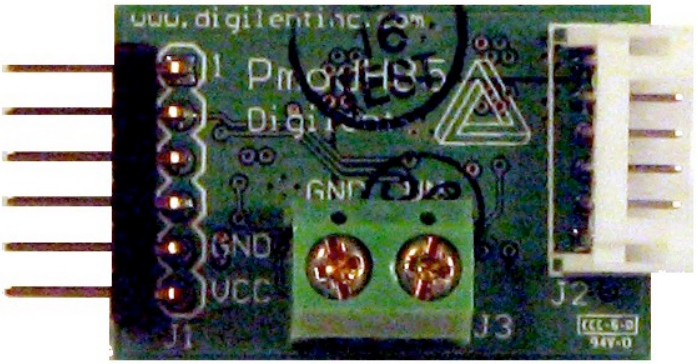
| A | B | Dir |
|---|---|---|
| 0 | 0 | Stop |
| 0 | 1 | CCW |
| 1 | 0 | CW |
| 1 | 1 | N/A |

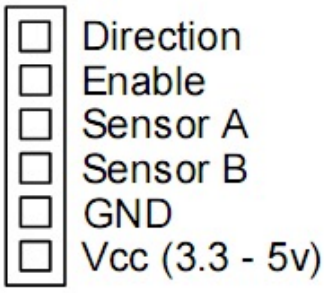
- Short-circuit!

# Direct Current (DC) Motors

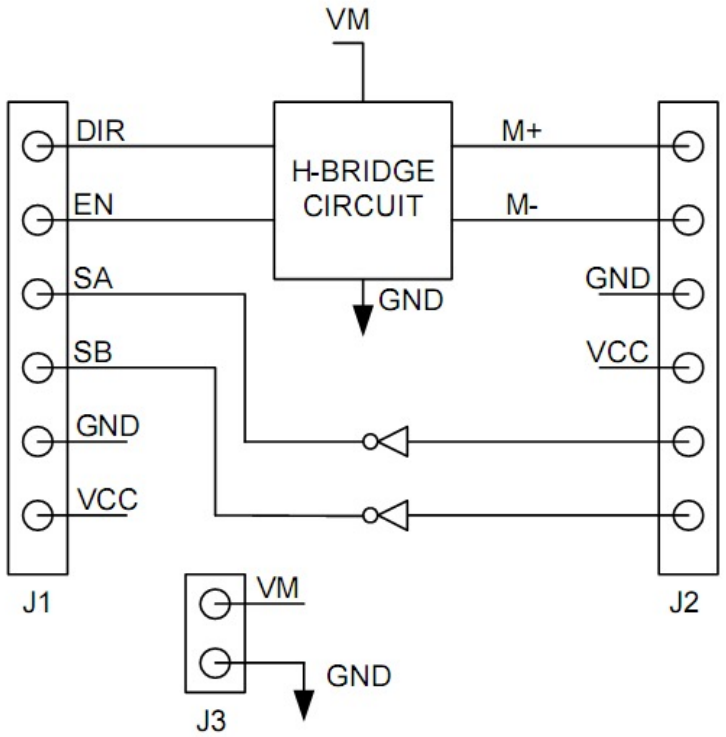
**The H Bridge**

- Digilent PMOD HB5
- DIR – controls the direction (sense)
- EN – if '1', the motor is on – PWM can be attached for variable speed
- A = EN and DIR, B = EN and (not DIR) **– Prevents short-circuits.**
- SA, SB – signals from the motor's built in Hall sensors




**HB5 6-Pin Header, J1**



# Direct Current (DC) Motors

**The motor driver L298N Dual H-Bridge**

•https://ardushop.ro/en/electronics/84-dual-h-bridge-for-dc-and-stepper-motors.html

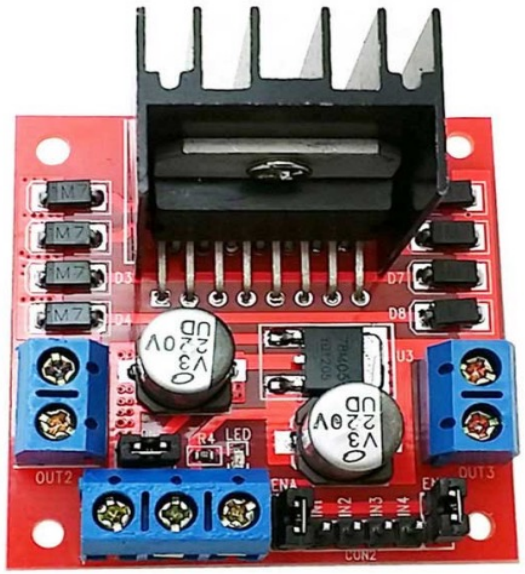
•The driver connects to 4 digital pins of Arduino, connected to the driver's In1, In2, In3 and In4 pins.

•Motor powering voltage : 5… 35 V

•Logic circuit voltage: 5 V (may be used to power the Arduino board)

•Can power motors that require a maximum of 2 Amperes (2000 mA).

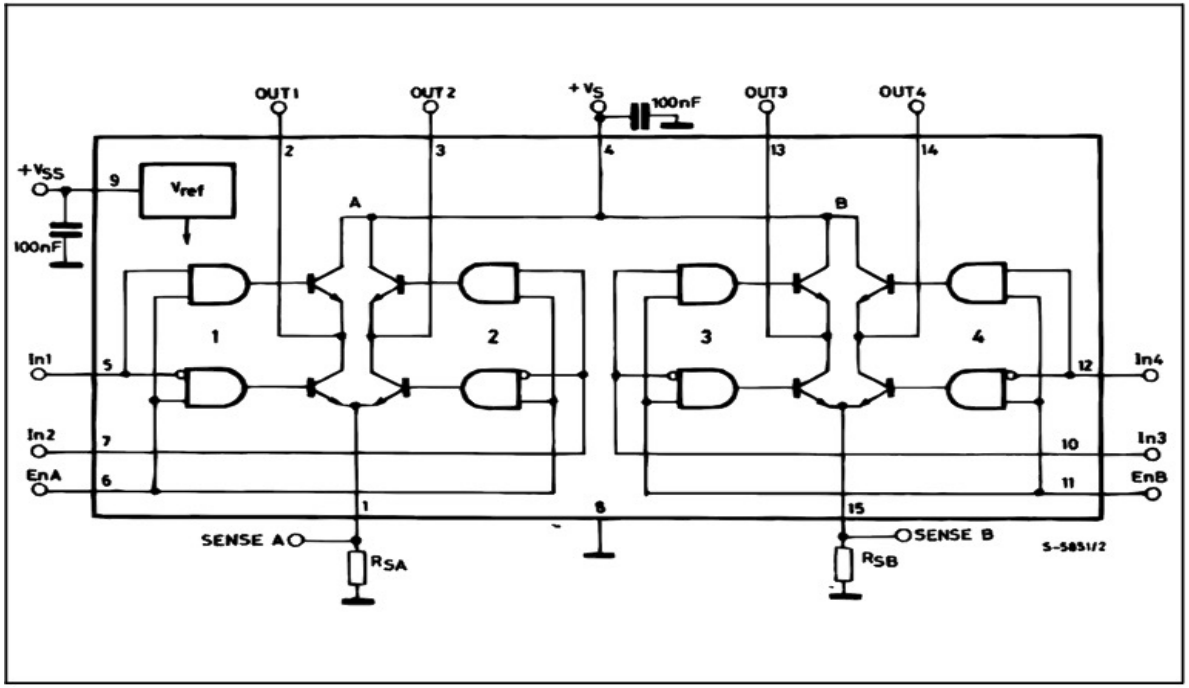
•Can control 2 DC motors, or one stepper motor



# Direct Current (DC) Motors

**The motor driver L298N Dual H-Bridge**

•Schematic of the L298N integrated circuit




| In1 | In2 | Effect |
|---|---|---|
| 0 | 0 | Motor 1 stopped (brake) |
| 0 | 1 | Motor 1 on – forward |
| 1 | 0 | Motor 1 on – reverse |
| 1 | 1 | Motor 1 stopped (brake) |

| In3 | In4 | Effect |
|---|---|---|
| 0 | 0 | Motor 2 stopped (brake) |
| 0 | 1 | Motor 2 on – forward |
| 1 | 0 | Motor 2 on – reverse |
| 1 | 1 | Motor 2 stopped (brake) |

# Direct Current (DC) Motors

## Example: Turning two motors, both directions

```
int MOTOR2_PIN1 = 3; // Each motor has 2 pins. If they have different digital values
int MOTOR2_PIN2 = 5; // the motor turns one way or another
int MOTOR1_PIN1 = 6;
int MOTOR1_PIN2 = 9;

void setup() {
  // The motor pins configured as output
  pinMode(MOTOR1_PIN1, OUTPUT);
  pinMode(MOTOR1_PIN2, OUTPUT);
  pinMode(MOTOR2_PIN1, OUTPUT);
  pinMode(MOTOR2_PIN2, OUTPUT);
}

void loop() {
    // 2 motors, 2 directions, 4 combinations, 1 sec each
    go(255,-255);
    delay(1000);
    go(-255,-255);
    delay(1000);
    go(-255,255);
    delay(1000);
    go(255,255);
    delay(1000);
}
```

```
// control function, speed for M1 and M2
void go(int speedLeft, int speedRight) {

if (speedLeft > 0) { // positive speed for pin 1
    analogWrite(MOTOR1_PIN1, speedLeft);
    analogWrite(MOTOR1_PIN2, 0);
  }
  else
 {
    analogWrite(MOTOR1_PIN1, 0);
    analogWrite(MOTOR1_PIN2, -speedLeft); // negative
speed,                          // absolute value of speed on pin2
  }

 if (speedRight > 0) {
    analogWrite(MOTOR2_PIN1, speedRight);
    analogWrite(MOTOR2_PIN2, 0);
  }
else
{
    analogWrite(MOTOR2_PIN1, 0);
    analogWrite(MOTOR2_PIN2, -speedRight);
  }
}
```

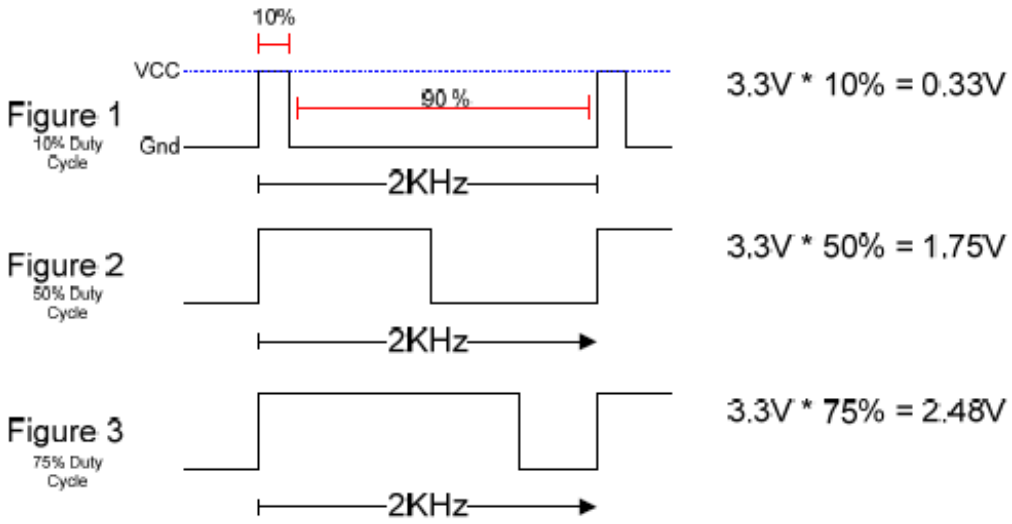
# Direct Current (DC) Motors

## Controlling the speed using PWM

An analog circuit controls the speed using a variable voltage. For a digital circuit we have two options:

•Using a variable resistance circuit to control the voltage applied to the motor (complicated solution, wastes energy as heat)

•Intermittent application of the full voltage, as PWM.



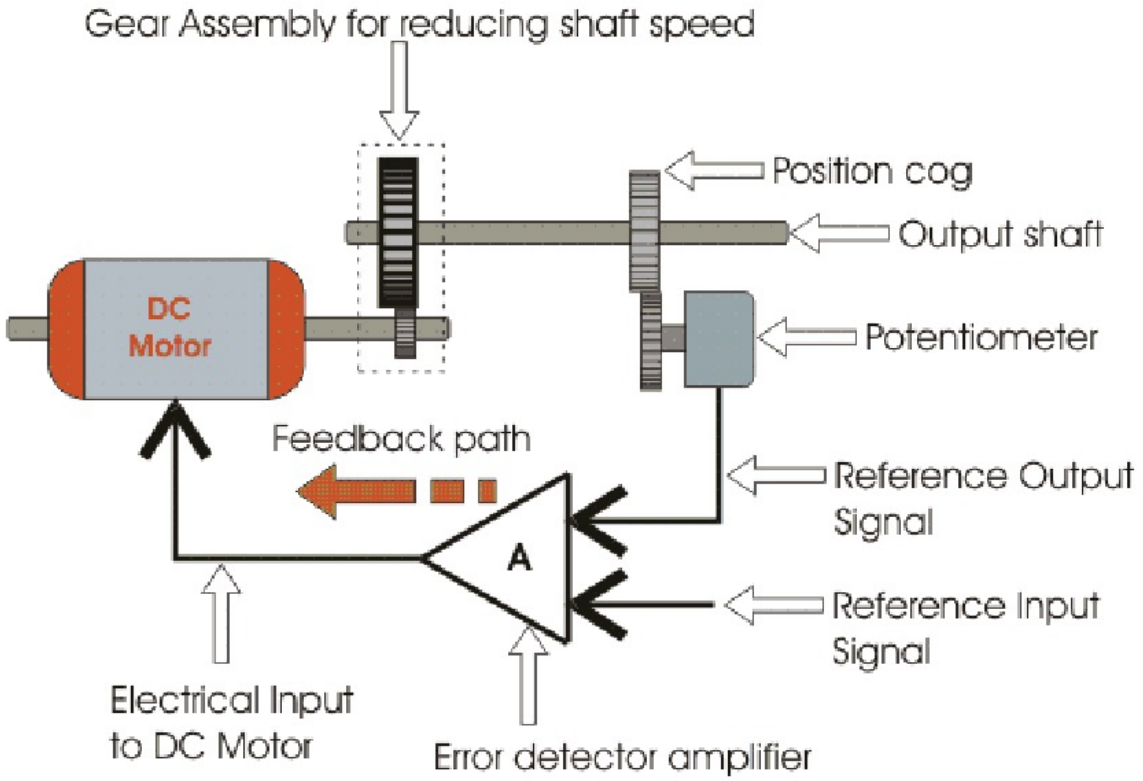

•When voltage is applied, the motor is driven by the electromagnetic force.
•When voltage is off, inertia causes the motor to keep moving on for a while.
•If the frequency of pulses is high enough, the starting + inertia moving allows the motor to perform a uniform motion.

# Servo motors

**The servo motor**

•Use of a feedback mechanism to maintain a given position, set by a control signal (analog or digital)

•Contains a DC motor, gears, and a control circuit.



# Servo motors

**The servo motor (ex: GWS Servo Kit)**

•The width of a pulse controls the amplitude of rotation.

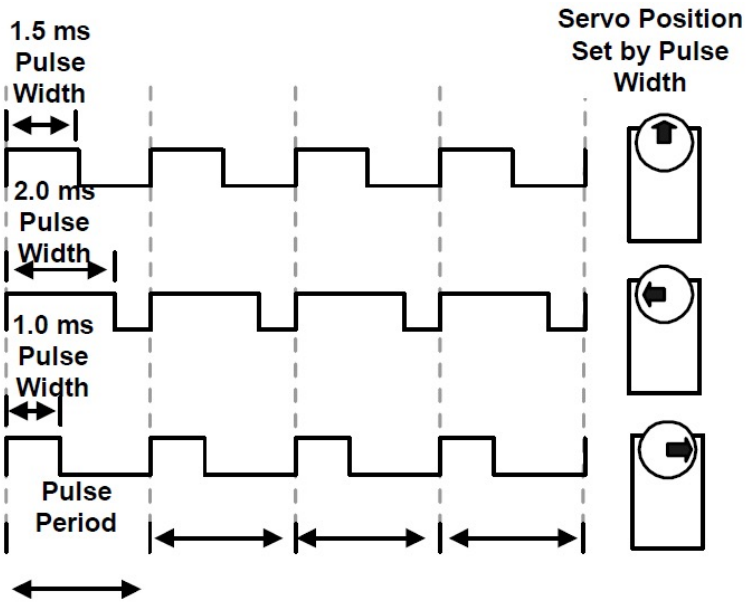
Typical control pulse widths:

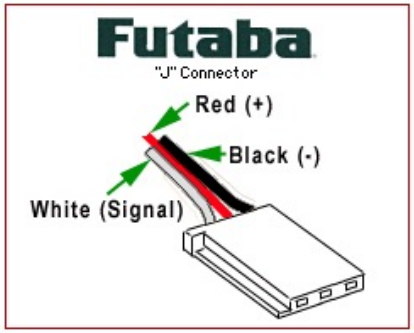
•1.5 ms – neutral (middle)

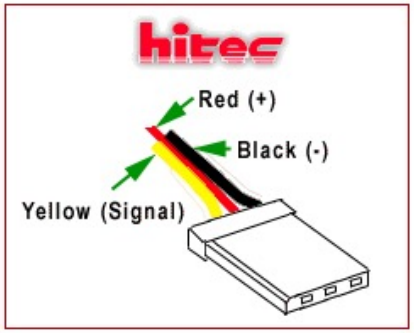
•1 ms – maximum left (or right)

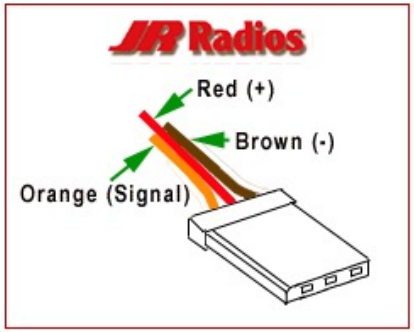
•2 ms – maximum right (or left)

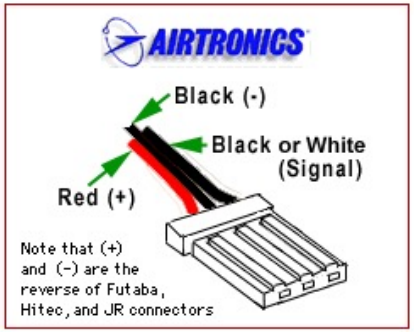
•PWM coding, carry frequency between 30 and 60 Hz



# Servo Motor Control- Arduino

The **Servo** library:

- Can control up to 12 servo motors on most Arduino boards
- 48 motors on Arduino Mega.
- Use of this library will disable analogWrite() (PWM) on pins 9 and 10, even if no servo motor is connected to these pins (except on Arduino Mega).
- On Arduino Mega, 12 servos can be used without affecting PWM; using more servos will disable PWM on pins 11 and 12.

**Connecting a servo to Arduino** (3 wires): Vcc, Gnd, signal.

- Vcc → to the 5V pin of the board.
- Gnd (black or brown) → to Arduino's GND.
- Signal pin (yellow, orange or white) connected to a digital pin.

**Note**: the motors require significant power! To use more than 2 servos, use an external power source.

# Servo Motor Control- Arduino

**Methods of the Servo class**:

***servo*.attach(pin) / *servo*.attach(pin, min, max)** – attaches the Servo object to pins
•servo: an instance of class Servo
•pin: number of the digital pin where the servo control signal will be attached
•min (optional): pulse width, in microseconds, corresponding to the minimum angle (0 degrees) of the servo motor (implicitly 544)
•max (optional): pulse width, in microseconds, corresponding to the maximum angle (180 degrees) of the servo motor (implicitly 2400)

***servo*.detach()** – detaches the Servo object from the pin.

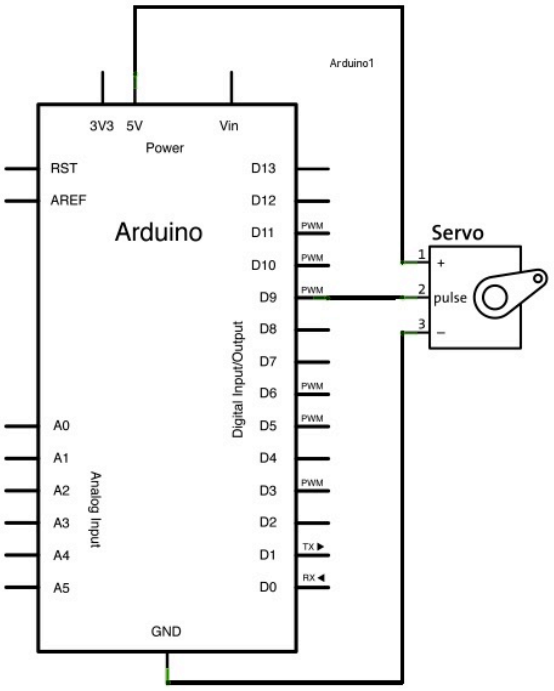
**boolean val *servo*.attached()** – checks whether the Servo object is attached to a pin.

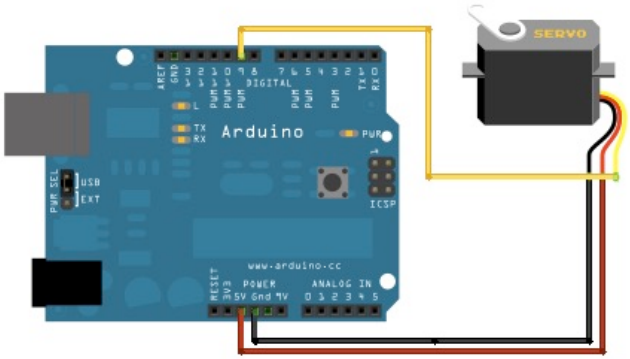
***servo*.write (angle)** – writes an angle value (0 .. 180) to the servo, controlling its rotation:
•*Standard Servo* ⇒ sets the angle of the motor axle (degrees) causing the motor to move towards the specified position.
•*Continuous rotating Servo* ⇒ configures the rotation speed (0: maximum speed in one direction; 180: maximum speed in the opposite direction; ≈ 90: stopped)

**int val = *servo*.read()** – reads the current servo angle, configured by the last call of write().

# Servo Motor Control- Arduino

**Example:** Rotates the axle of a servo by sweeping to and fro between 0 and 180 degrees (http://arduino.cc/en/Tutorial/Sweep)

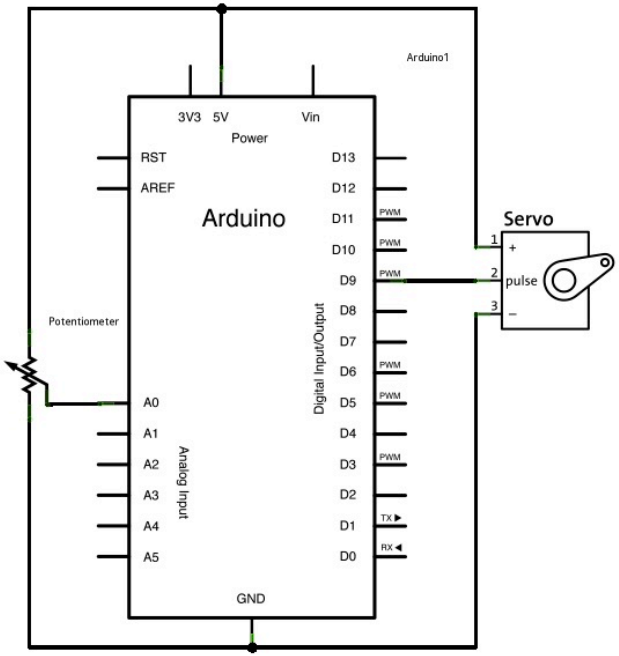


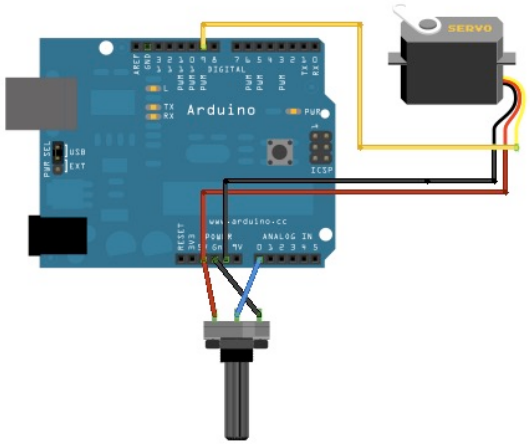




```
#include <Servo.h>

Servo myservo; // object for controlling the Servo
int pos = 0;        // variable for the current position of the axle

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

 void loop()
{
   for(pos = 0; pos < 180; pos += 1)  // from 0 to 180 degrees
   {
      myservo.write(pos); // sets the desired position
      delay(15);                 // wait 15 ms to allow the motor to position
   }

   for(pos = 180; pos>=1; pos-=1)     // sweep back
   {
       myservo.write(pos);
       delay(15);
    }
}
```

# Servo Motor Control- Arduino

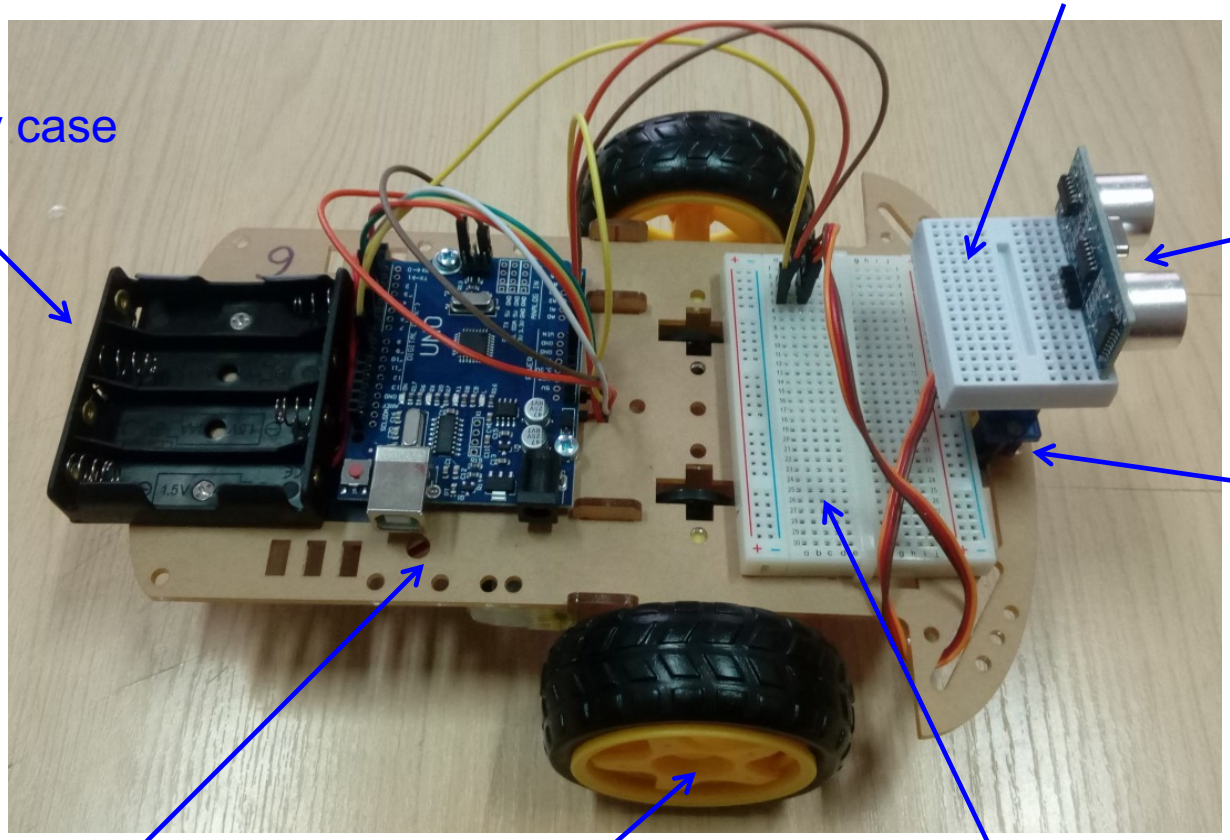
**Example:** Controlling the angle of a servo using Arduino and a potentiometer (http://arduino.cc/en/Tutorial/Knob)



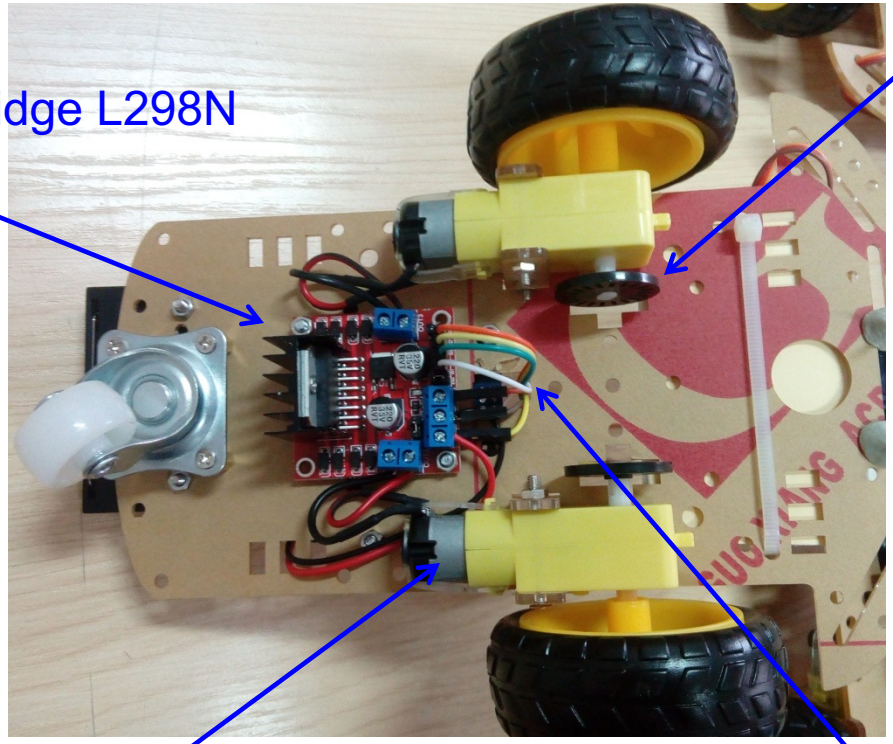




```
#include <Servo.h>

Servo myservo;  // object for controlling the Servo

int potpin = 0;  // analog pin for reading the potentiometer
int val;         // variable for the analog value
int angle;       // the servo angle


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

void loop()
{
  val = analogRead(potpin); // reads the value of the potentiometer
  angle = map(val, 0, 1023, 0, 179); // scale the read value (0…1023)
                                     // to the range 0… 179
  myservo.write(angle); // write the new position for the servo
  delay(15);            // wait for the servo to move
}
```

# An experimental robot



# An experimental robot



Code wheel for speed measurement
No IR sensor !

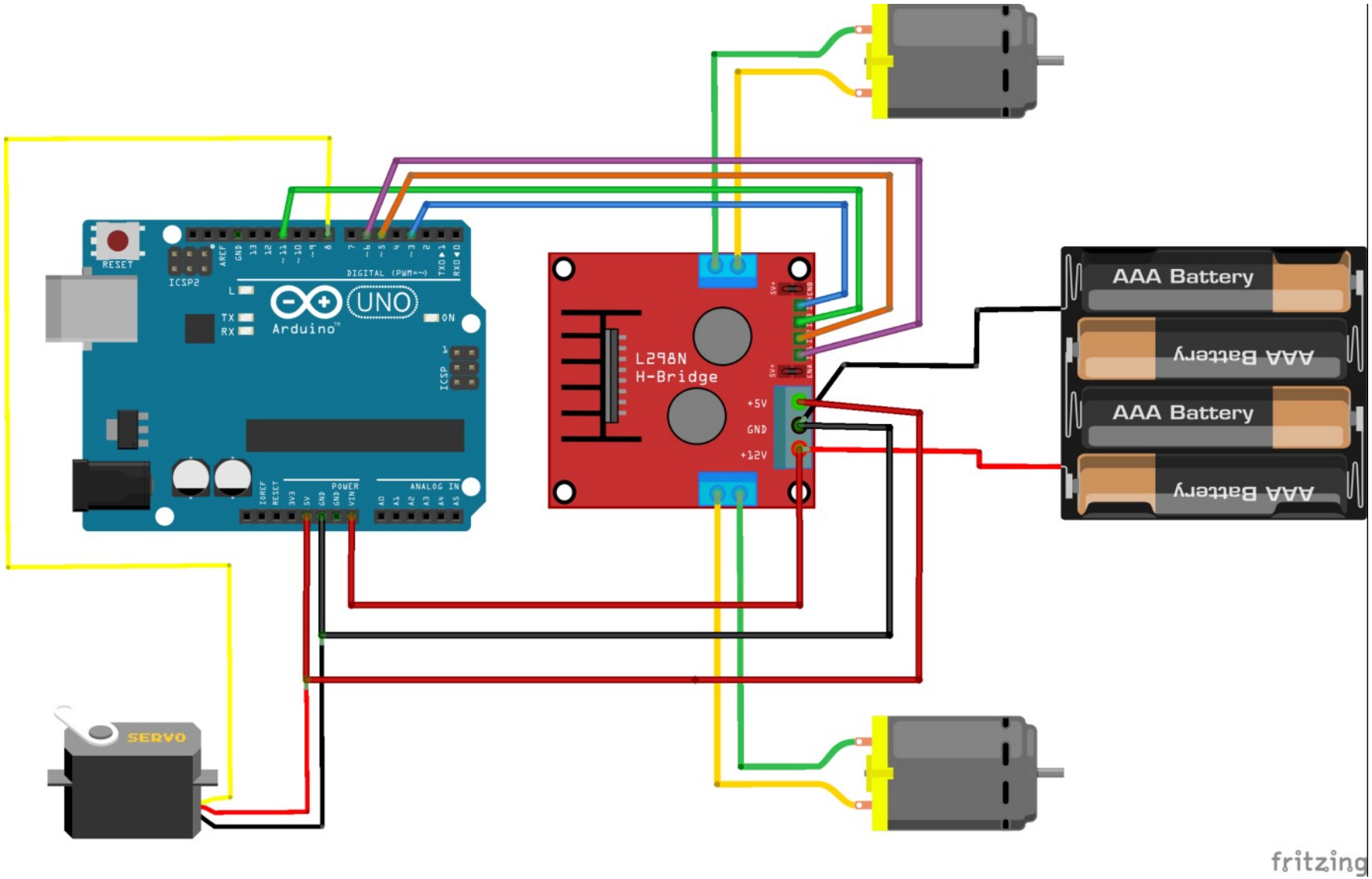
Dual H-bridge L298N

DC Motor

Wheels

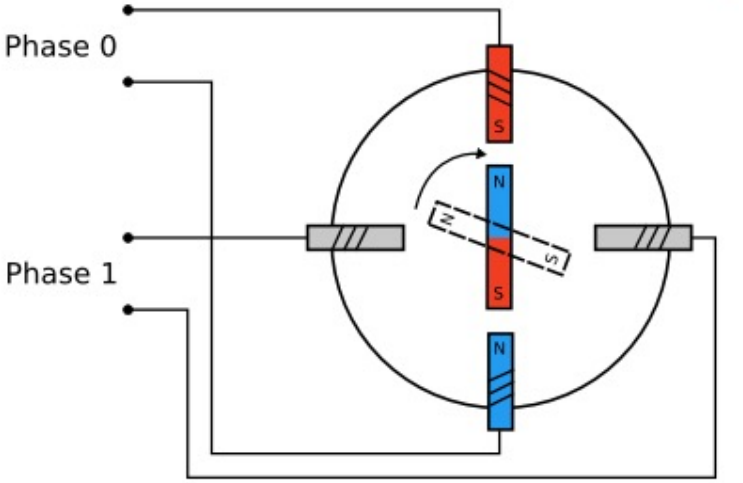
Control wires (In1, … In4)

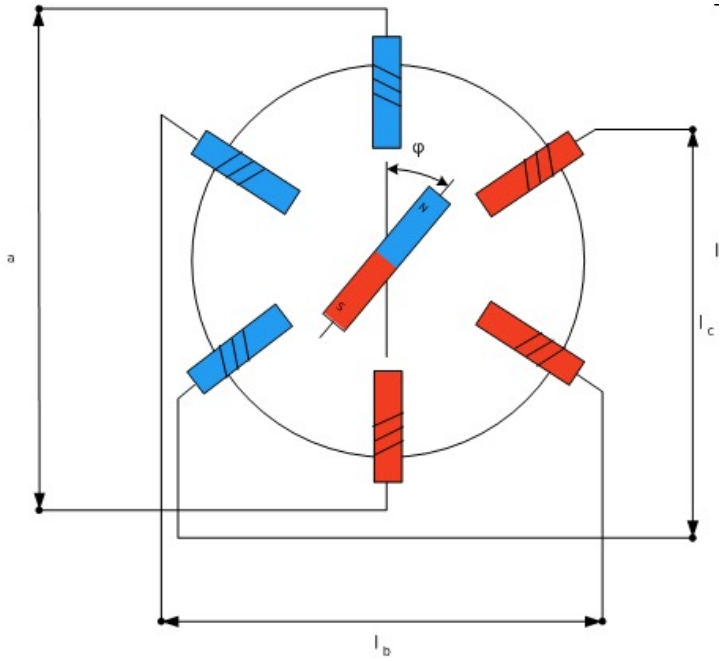
# An experimental robot



# Stepper motors

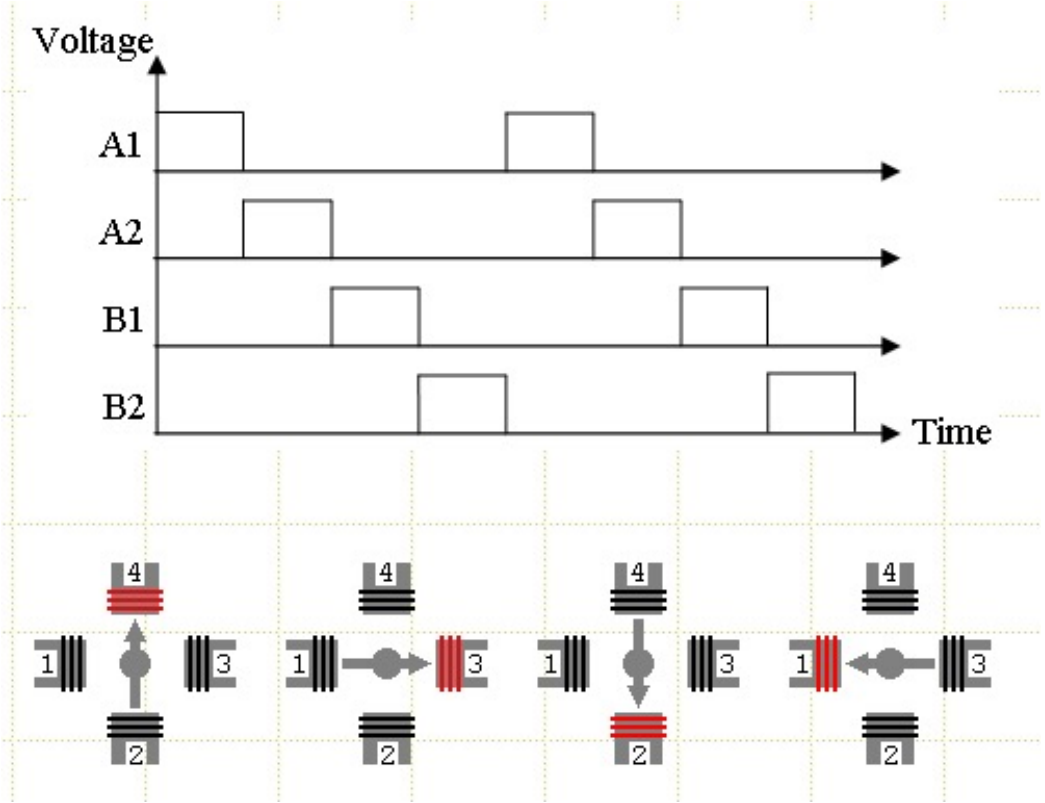
•The rotation is achieved step by step, by selective activation of the coils
•The currents in the coils are changed by electronic control, unlike the classical DC motors which use mechanical control (brushes)



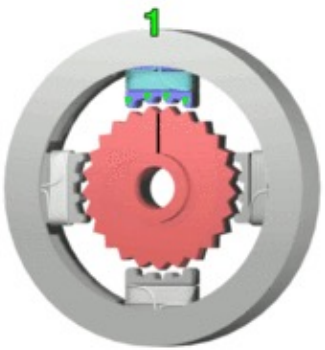
# Stepper motors

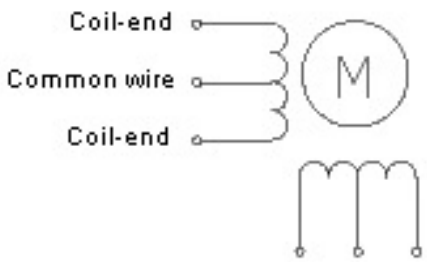
•A stepper motor controller must generate the right sequence for activating the coils
•One can achieve complete rotations, or partial rotations, depending on the number od pulses - precise control of the rotation amount!

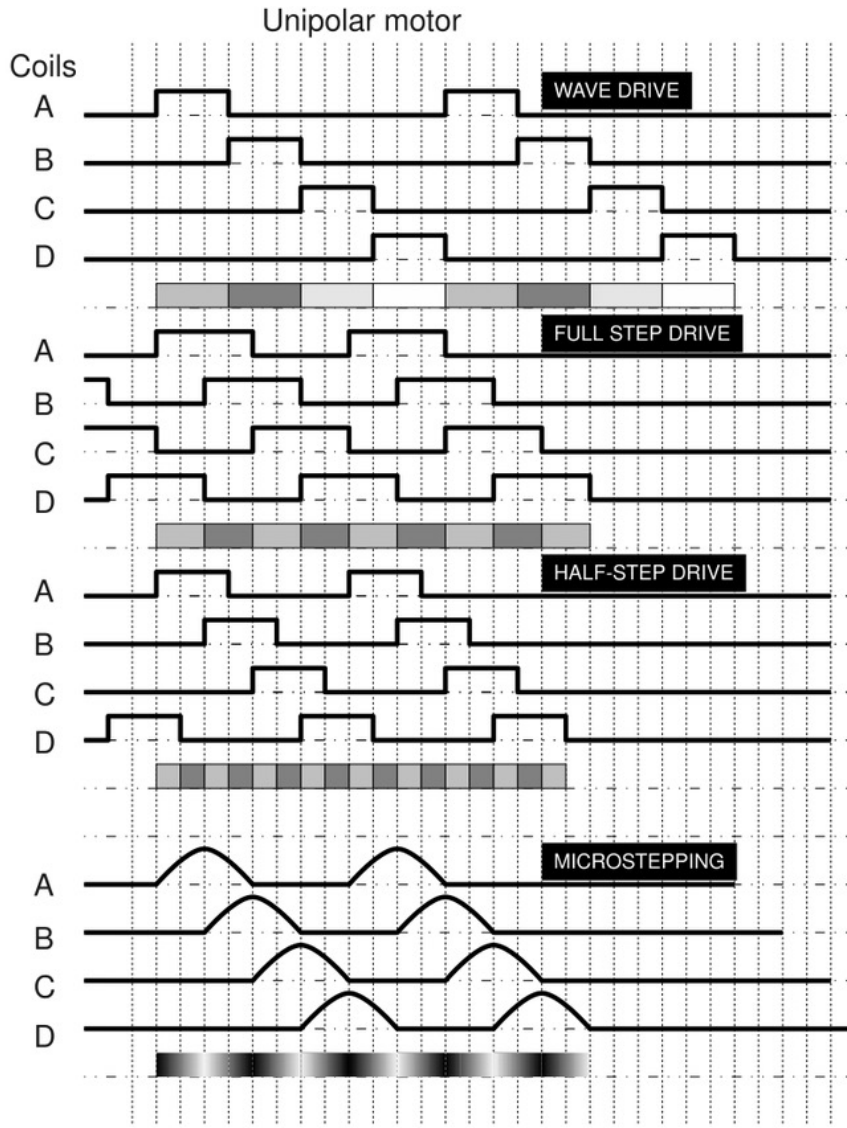


# Stepper motors

## Unipolar stepper motor




**Wave drive, or full step one phase**

- Low torque, rarely used. 25 teeth / 4steps for a complete rotation of a tooth $\Rightarrow$ 25*4 = 100 steps for a complete rotation $\Rightarrow$ each step will have 360/100 = 3.6 °

**Full step two phase drive**

- Maximum torque, most used

**Half stepping**

- Low torque (70%) / resolution x2 (8 steps for moving a tooth $\Rightarrow$ 25*8 = 200 steps for a complete rotation $\Rightarrow$ 1 step = 1.8 °

**Micro-stepping**

- Smoother operation

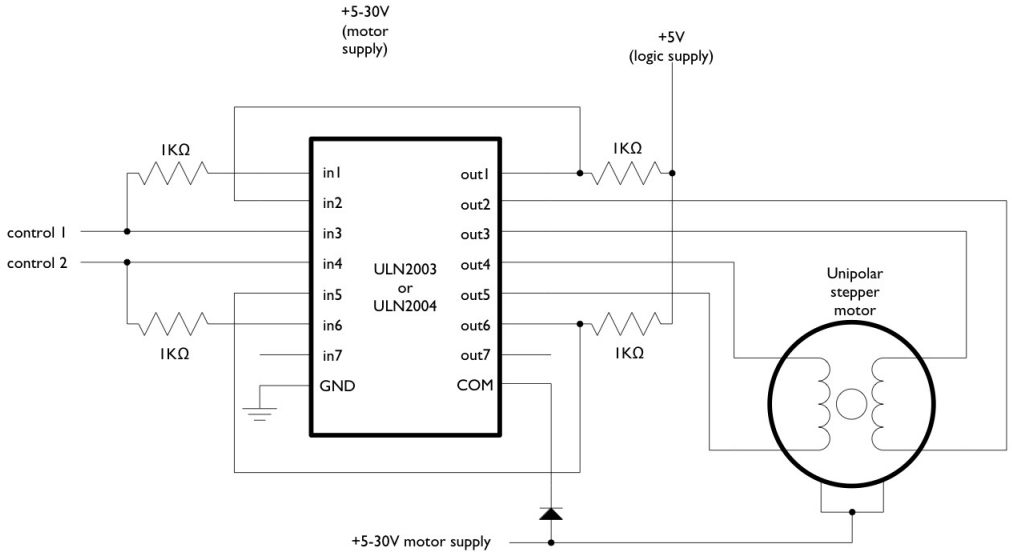
# Stepper motors with Arduino

**The Arduino Stepper library** (http://arduino.cc/en/reference/stepper)

- Allows the control of unipolar or bipolar stepper motors. For using this library, you need a stepper motor and a hardware interface for it.

To create a Stepper class object, call the constructor:

**Stepper(***steps, pin1, pin2***)** - ex: Stepper myStepper = Stepper(100, 5, 6);
**Stepper(***steps, pin1, pin2, pin3, pin4***)**
int *steps*: number of steps for a complete rotation(ex: 360 / 3.6 = 100 steps)
int *pin1*, *pin2*: two pins for the hardware interface (2 pin configuration)
int *pin3, pin4*: *optional* additional pins for the hardware interface, 4 pin configuration




2 pin/wire setup

**Control sequence (2 pin):**

| Step | pin 1 | pin 2 |
|---|---|---|
| 1 | low | high |
| 2 | high | high |
| 3 | high | low |
| 4 | low | low |

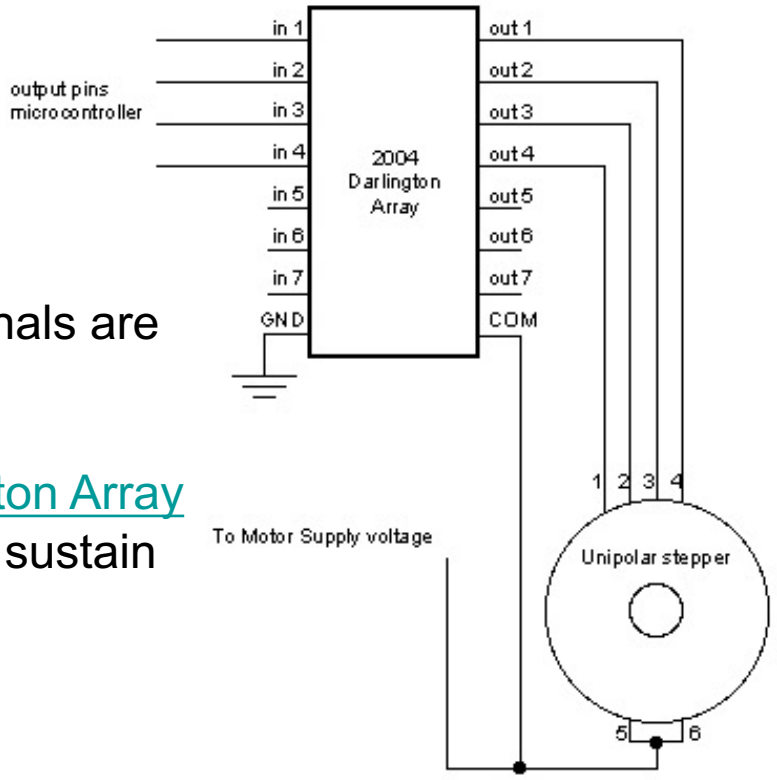
# Stepper motors with Arduino

**Control sequence (4 pins):**

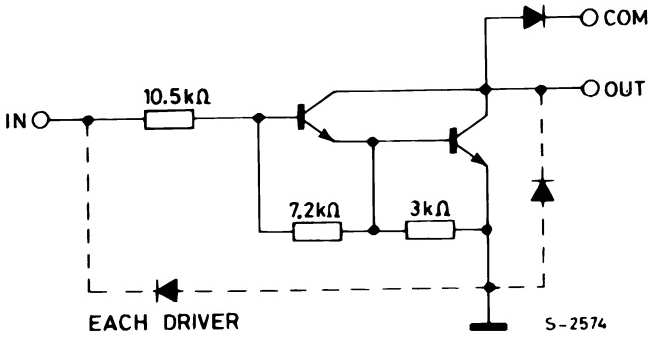
| Pas | pin 1 | pin 2 | pin 3 | pin 4 |
|---|---|---|---|---|
| 1 | high | low | high | low |
| 2 | low | high | high | low |
| 3 | low | high | low | high |
| 4 | high | low | low | high |

When using the Stepper library, the control signals are generated by the library!

Example of hardware interface: U2004 Darlington Array
- High voltage and intensity. Each channel can sustain 500 mA, with peaks up to 600 mA.




4 pin / wire setup



# Stepper motors with Arduino

**Functions of the Stepper library** (http://arduino.cc/en/reference/stepper)

**setSpeed**(long *rpms*) – sets the motor rotation speed, in rotations per minute (RPMs). This function does not start the motor, but only sets the speed that will be used when the step() function is called.

**step**(int *steps*) – Rotates the motor for a specified number of steps, with the configured speed.

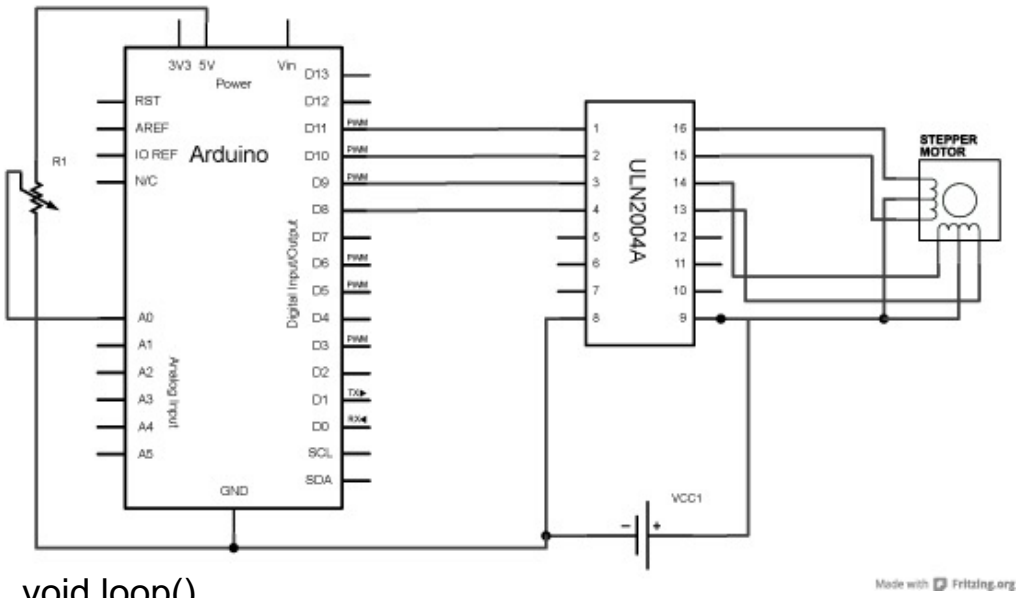
•int *steps*: number of steps for the motor to execute: pozitive (+) rotate in one direction, negative (-) rotate in the oposite direction

•Function is blocking: it will wait until the motor finishes the rotation to exit. (Ex: for 1 RPM, with the steps parameter set at 100, the function will block the program 1 minute for a complete rotation of the motor).

•For better control, call this function with a smaller number of steps, and higher speed.

# Stepper motors with Arduino

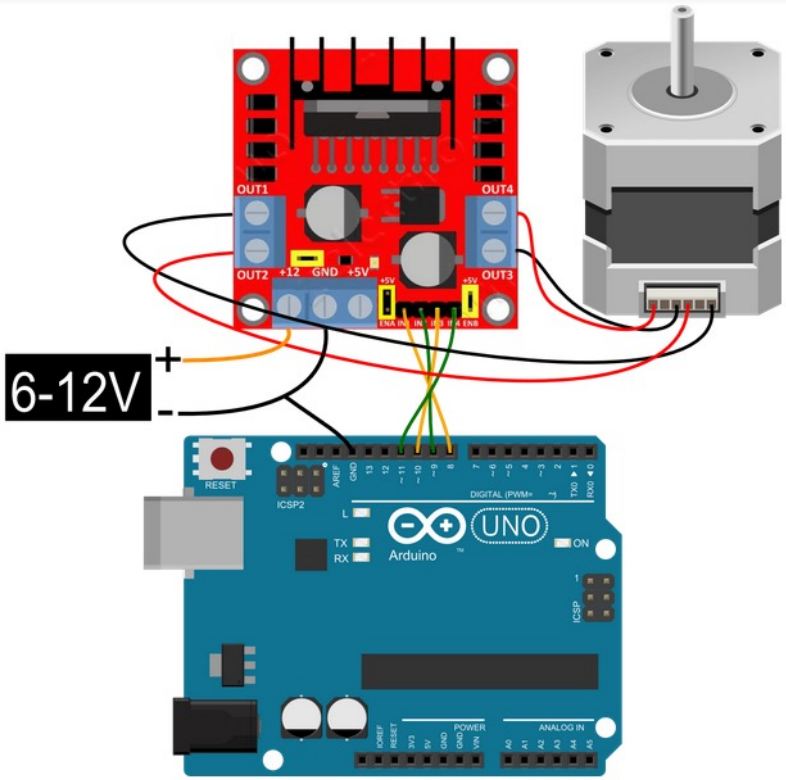
**Example: Stepper motor controlled by a knob potentiometer**
([http://arduino.cc/en/Tutorial/MotorKnob](http://arduino.cc/en/Tutorial/MotorKnob))




```
#include <Stepper.h>
#define STEPS 100

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

// the previous read from the potentiometer
int previous = 0;

void setup()
{
  // motor speed, 30 RPM
  stepper.setSpeed(30);
}

void loop()
{
  // read the potentiometer state
  int val = analogRead(0);
  // move the motor with the difference between reads
  stepper.step(val - previous);
  // the current value becomes the previous
  previous = val;
}
```

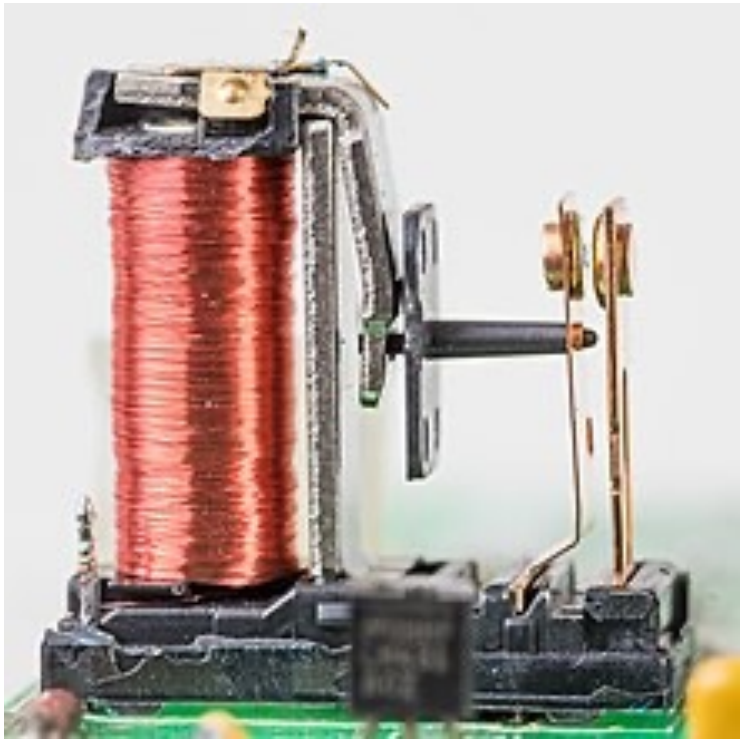
# Stepper motors and L298N

**The stepper can also be controlled using the dual H-bridge**
Source: https://coeleveld.com/arduino-stepper-l298n/

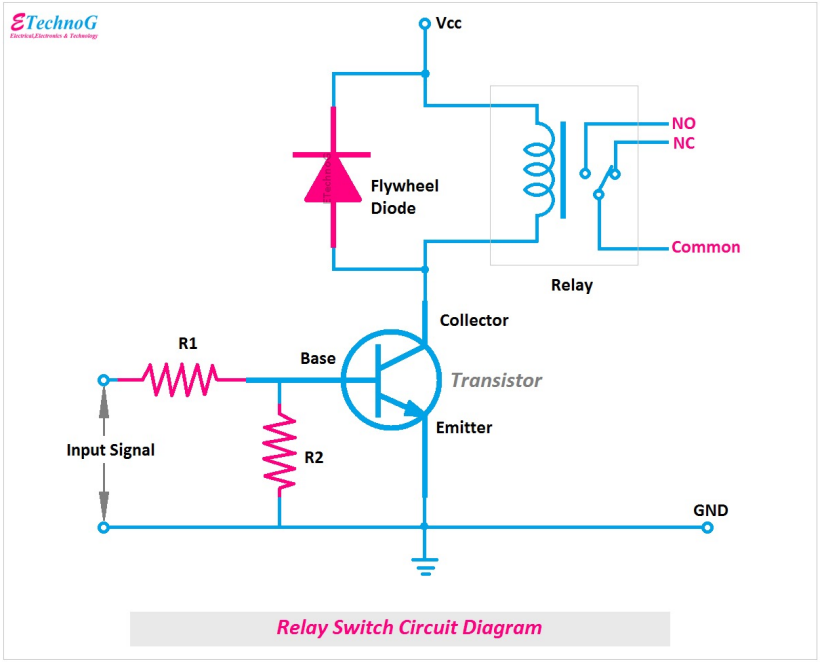



```
#include <Stepper.h>

const int stepsPerRevolution = 200;  // change with the
// specifications of your own motor

// init the library with pins 8 …11:
Stepper myStepper(stepsPerRevolution, 8, 9, 10, 11);

void setup() {
  // set speed to  60 rpm:
  myStepper.setSpeed(60);
  // init the serial interface
  Serial.begin(9600);
}

void loop() {
  // complete rotation, clockwise
  Serial.println("clockwise");
  myStepper.step(stepsPerRevolution);
  delay(500);

  // complete rotation, counterclockwise
  Serial.println("counterclockwise");
  myStepper.step(-stepsPerRevolution);
  delay(500);
}
```

# Relays

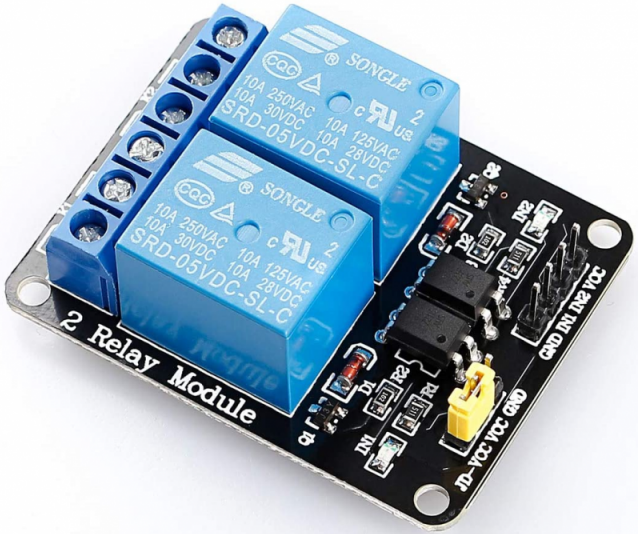


Source: https://en.wikipedia.org/wiki/Relay

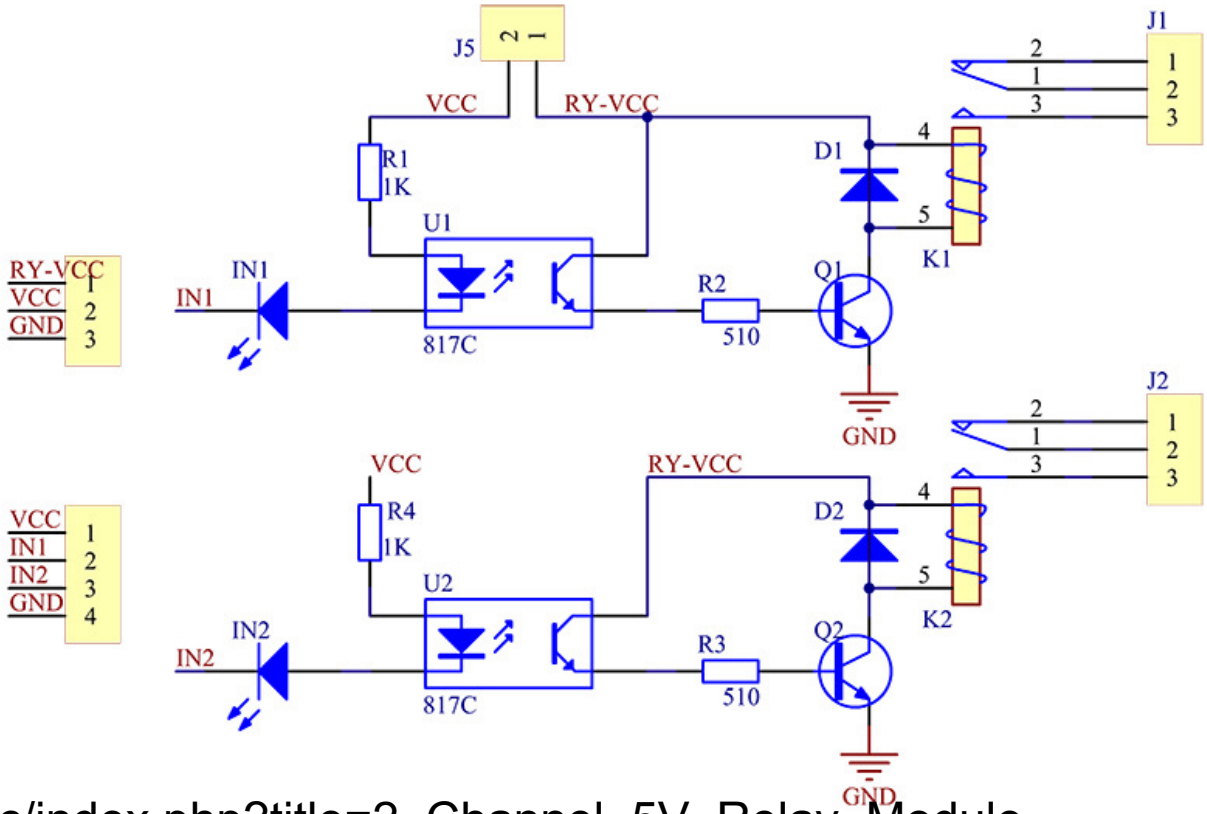



https://www.etechnog.com/2021/12/what-is-relay-switch-circuit.html

# Relay modules for Arduino

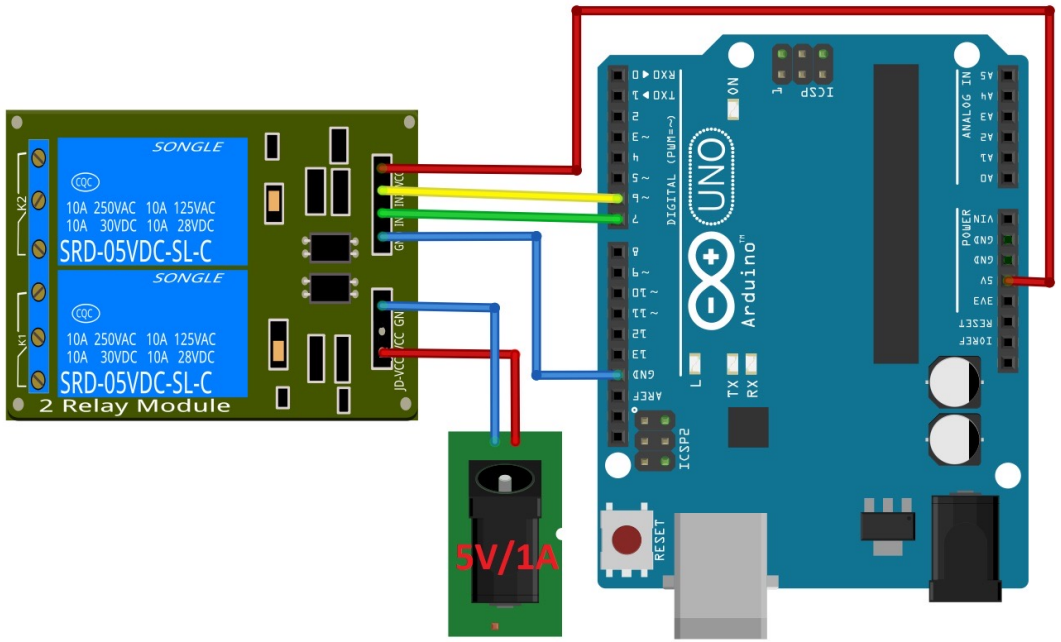



Max load: AC 250V/10A, DC 30V/10A
Trigger current: 5mA
Trigger voltage: 5V




Source: http://wiki.sunfounder.cc/index.php?title=2_Channel_5V_Relay_Module

# Relay modules for Arduino




```
const int Releu1 = 7;
const int Releu2 = 6;

void setup() {
  pinMode(Releu1, OUTPUT);
  pinMode(Releu2, OUTPUT);
}

void loop() {
  digitalWrite(Releu1, HIGH);
  digitalWrite(Releu2, LOW);
  delay(1000);
  digitalWrite(Releu2, HIGH);
  digitalWrite(Releu1, LOW);
  delay(1000);
}
```

Source: https://ardushop.ro/ro/home/50-modul-releu-2-canale.html