Classes of “Arduino uno”

Servo motor

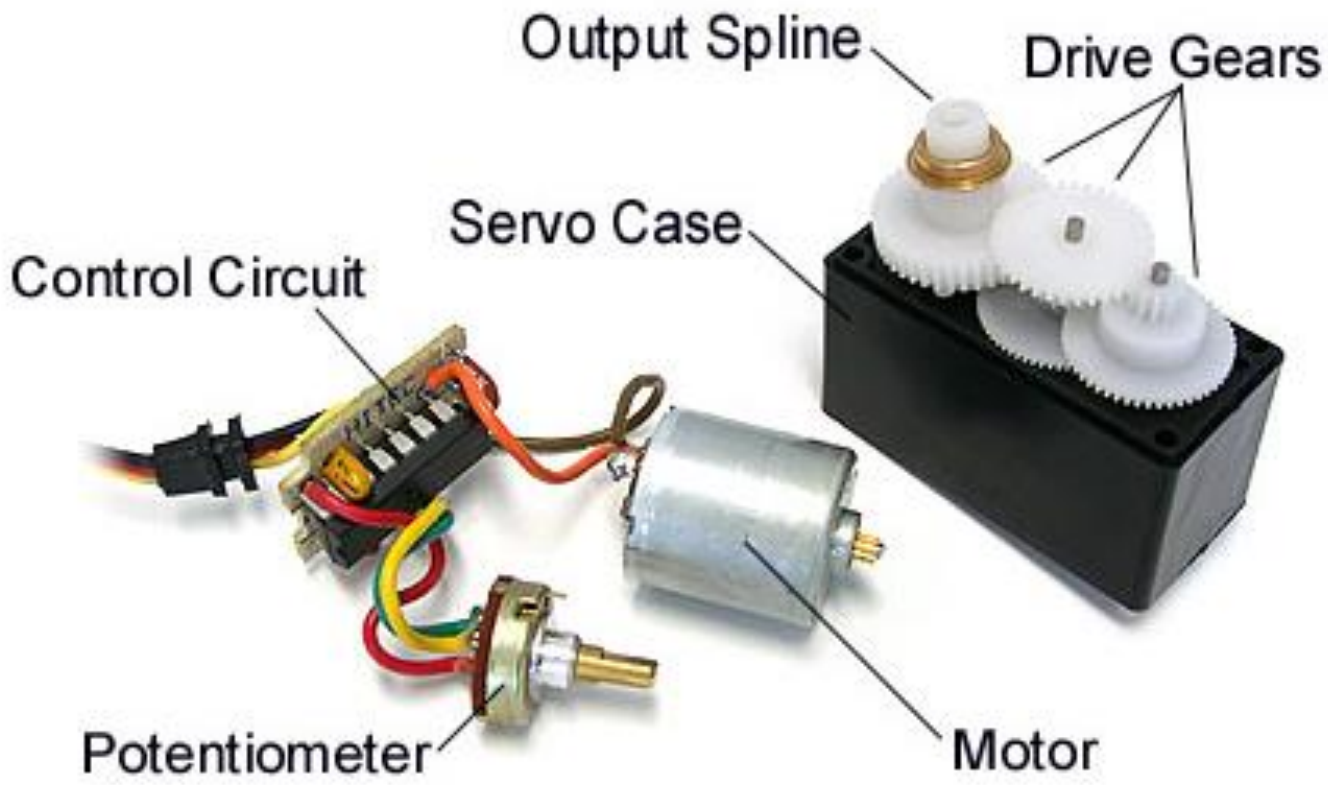
http://arduino.cc/en/Tutorial/Knob

Beginning Arduino

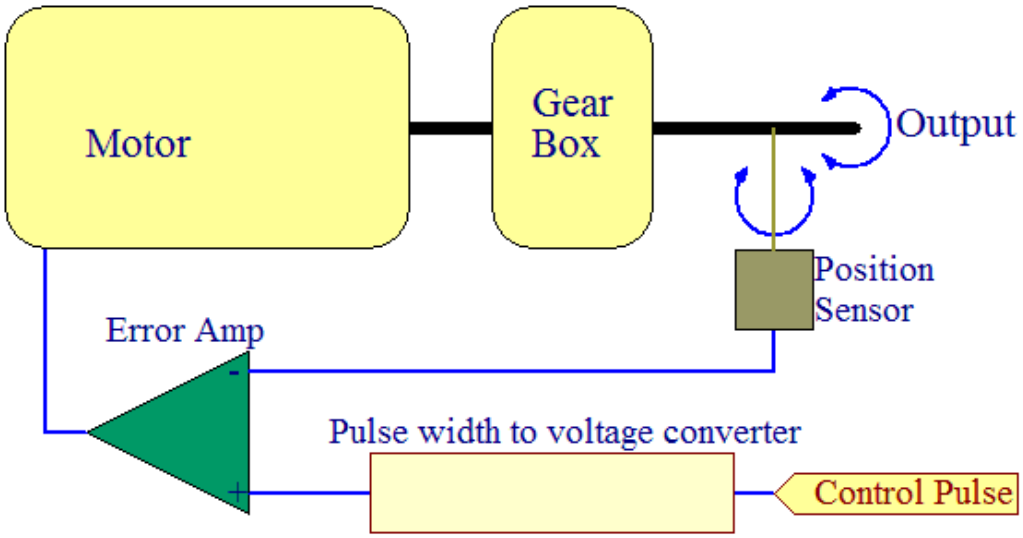
Servo motors

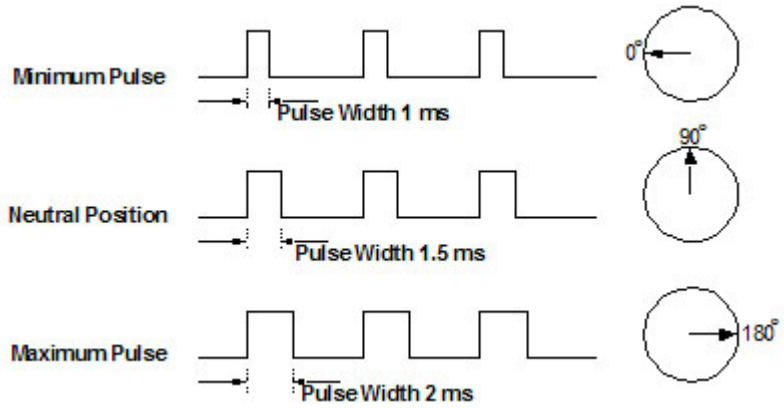
They are characterized by their momentum and their velocity.

Servo motor- Component parts.



Servo motor- Control pulses and control diagram.





http://www.digitalnemesis.com/info/docs/rcservo/

Servos have integrated gears and a shaft that can be precisely controlled. Standard servos allow the shaft to be positioned at various angles, usually between 0 and 180 degrees. Continuous rotation servos allow the rotation of the shaft to be set to various speeds.

Servos draw considerable power, so if you need to drive more than one or two, you'll probably need to power them from a separate supply (i.e. not the +5V pin on your Arduino).

(http://arduino.cc/en/Reference/Servo)

Servo motor

Servo motors have three wires:
power, ground, and signal.

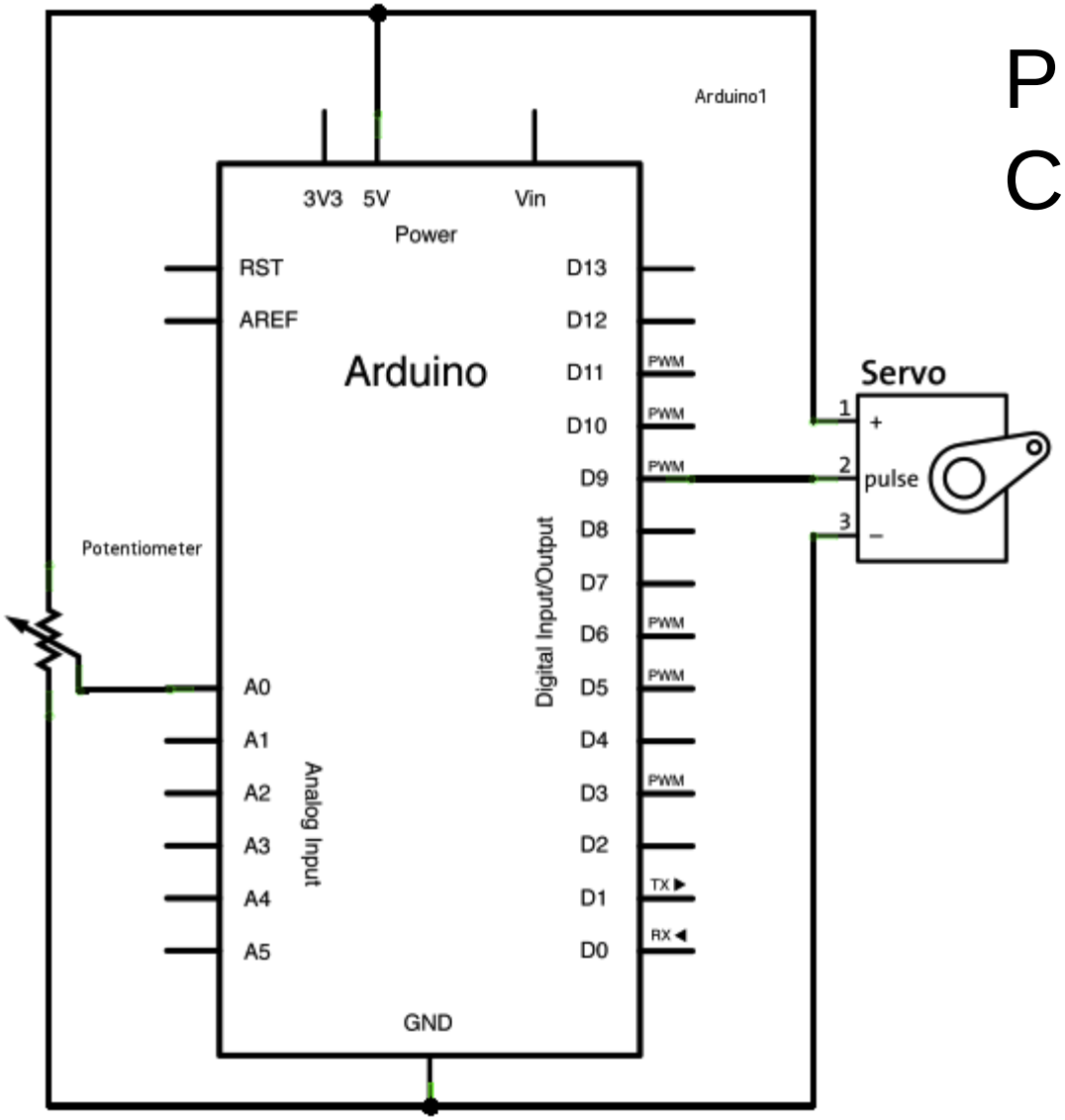
The power wire (red) should be connected to +5V power supply.

The ground wire (black, brown) should be connected to the power supply ground and to the ground pin on the Arduino board.

The signal pin (yellow or orange) should be connected to a PWM pin output on the Arduino board.

Project Servo.
Controls the position of a RC (hobby) servo motor with one Arduino and a potentiometer.

Servo motor

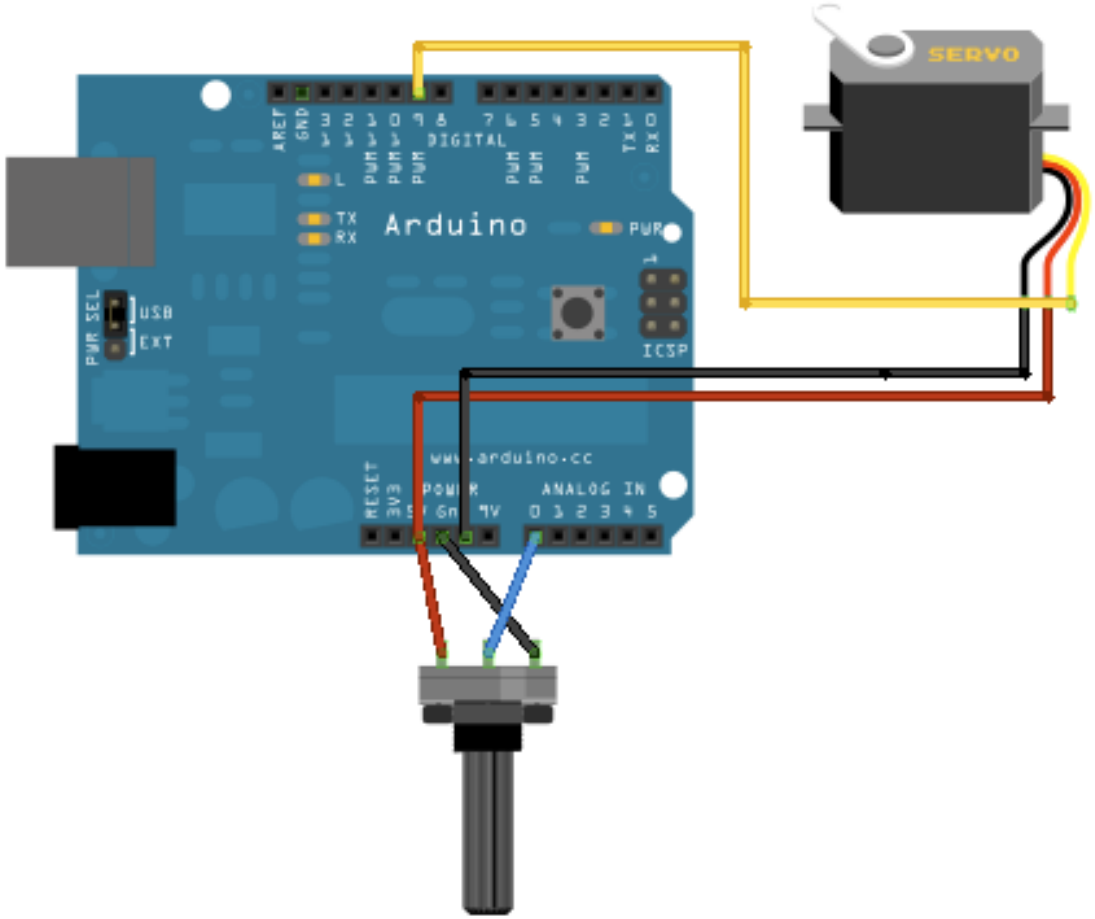


Project Servo Circuit

http://fritzing.org/

Servo motor

PB view



Servo motor- code

```
// Controlling a servo position using a potentiometer (variable resistor)
// by Michal Rinott <http://people.interaction-ivrea.it/m.rinott>
#include <Servo.h>
Servo myservo; // create servo object to control a servo
int potpin = 0; // analog pin used to connect the potentiometer
int val; // variable to read the value from the analog pin
void setup()
        { myservo.attach(9); // attaches the servo on pin 9 to the servo object
         }
void loop()
        { val = analogRead(potpin);
// reads the value of the potentiometer (value between 0 and 1023)
val = map(val, 0, 1023, 0, 179);
// scale it to use it with the servo (value between 0 and 180)
myservo.write(val); // sets the servo position according to the scaled value
delay(15); // waits for the servo to get there
}
```

Code analysis:

First, the Servo.h library is included:

#include <Servo.h>

Then a Servo object called **myservo is declared:**

Servo myservo; // Create a servo object

In the setup loop, you attach the servo you have just created to Pin 9

myservo.attach(9); // Attaches the servo on Pin 9 to the servo object

The attach command attaches a created servo object to a designated pin. The attach command can take three parameters. The first parameter is the pin, the second is the minimum (0 degree) angle in pulse width in microseconds (defaults to 544), and the third parameter is the maximum degree angle (180 degrees) in pulse width in microseconds (defaults to 2400). For most purposes we can simply set the pin and ignore the optional second and third parameters.

We can connect up to 12 servos to an Arduino

Note that using this library disables the analogWrite (PWM) function on Pins 9 and 10.

In the main loop, read the analog value from the potentiometer

int angle = analogRead(0); // Read the pot value

Then that value is mapped so the range is now between 0 and 180,

angle=map(angle, 0, 1023, 0, 180); // Map the values from 0 to 180 degrees

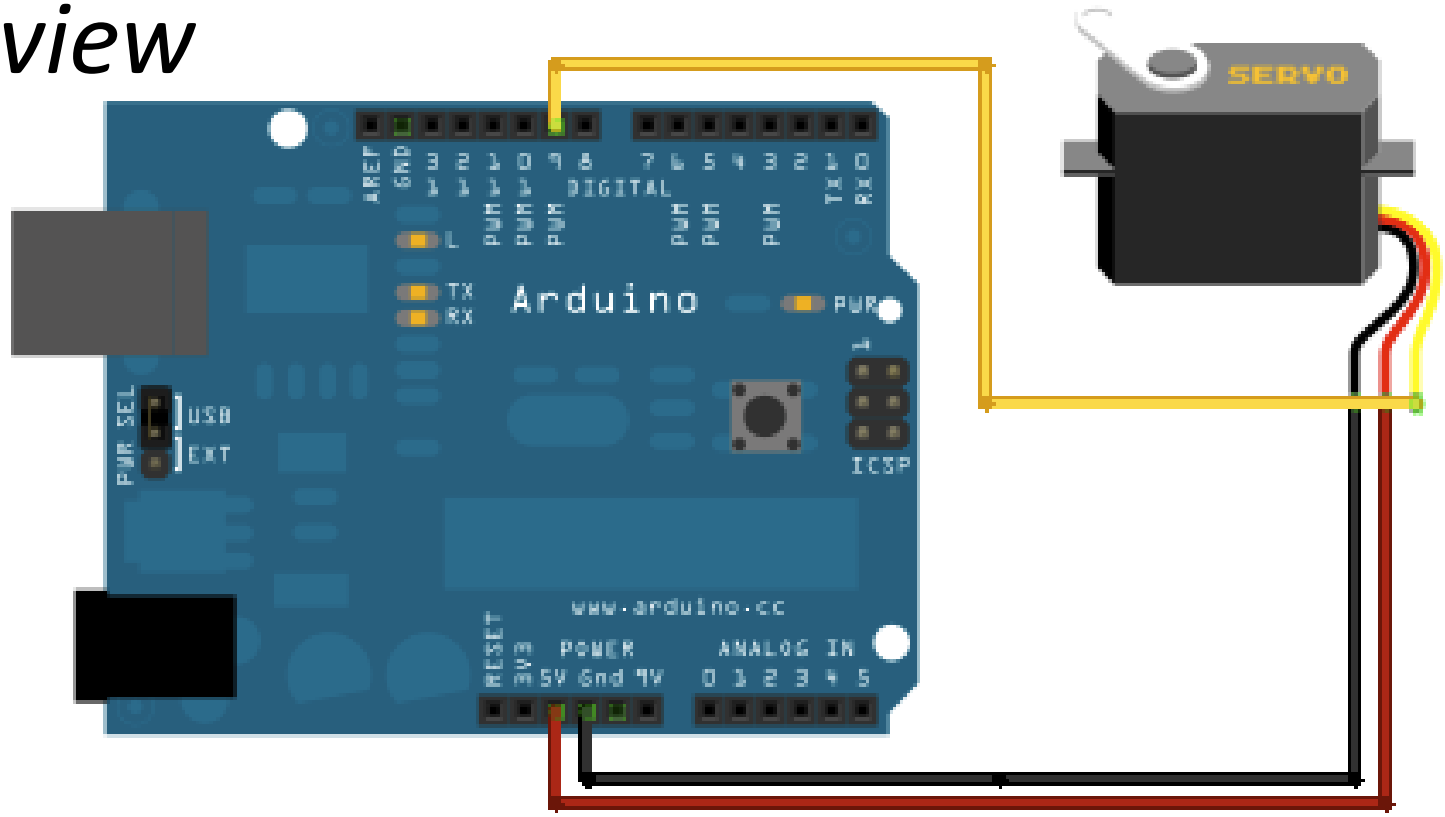
Then you take your servo object and write the appropriate angle, in degrees, to it (the angle must be between 0 and 180 degrees):

Project: Sweep

Sweeps the shaft of a RC servo motor back and forth across 180 degrees.

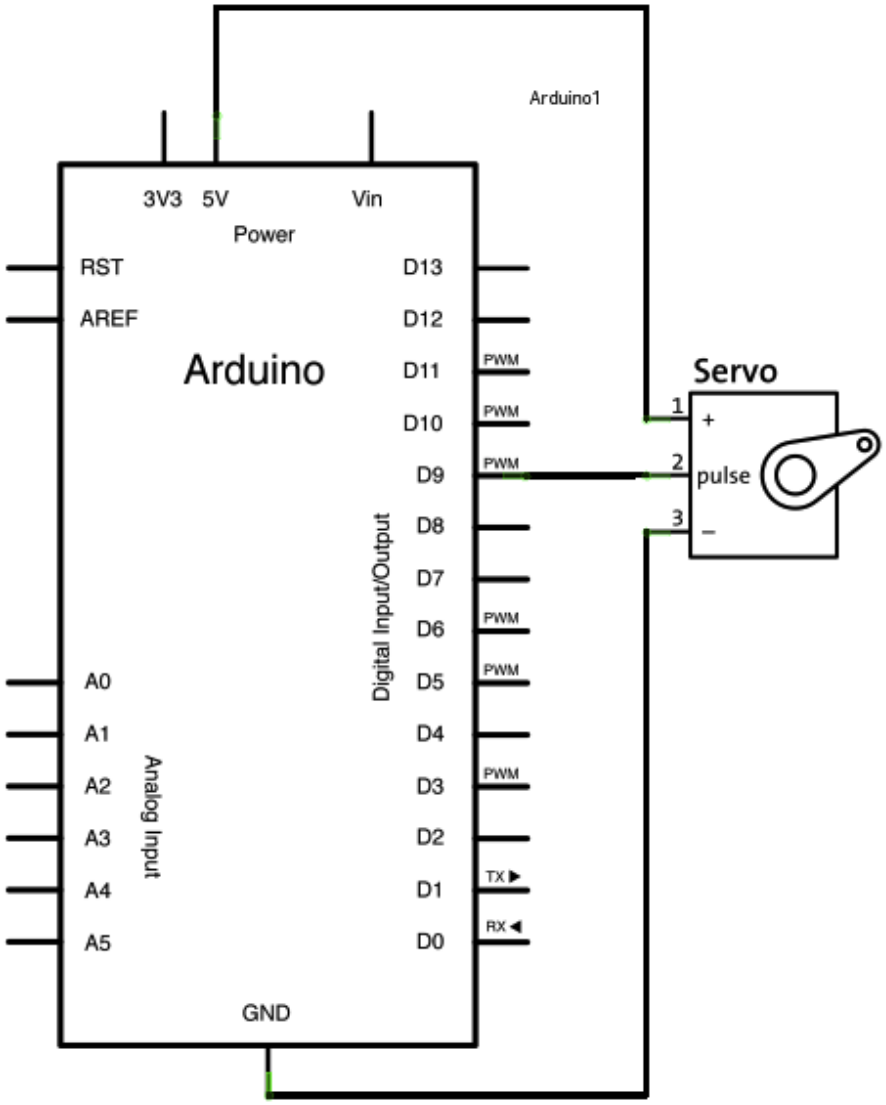
Sweep Project-

PB view



Sweep Project

Circuit view



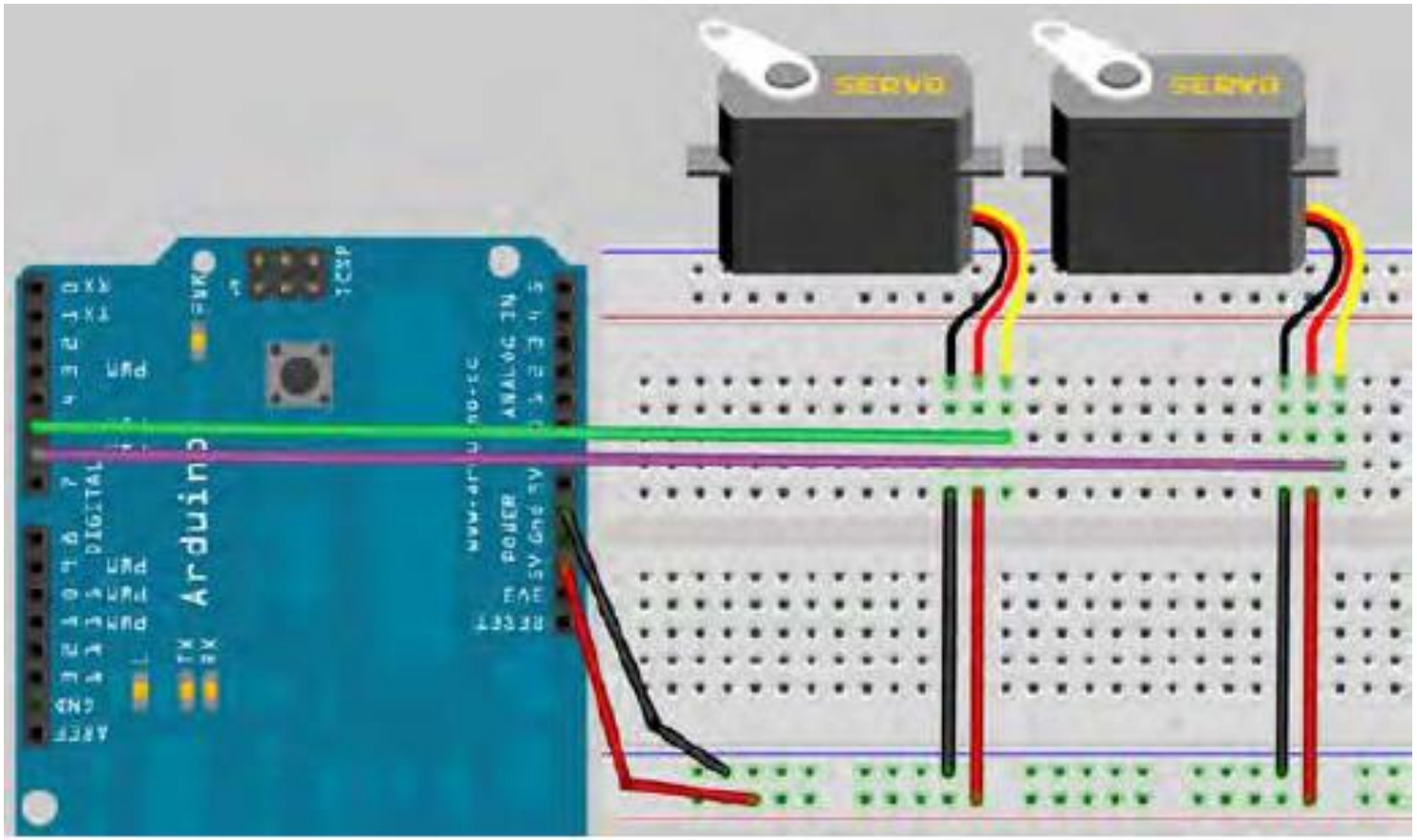
```
// Sweep
// by BARRAGAN <http:
//barraganstudio.com>
// This example code is in the public domain.
#include <Servo.h>
Servo myservo; // create servo object to control a servo
int pos = 0; // variable to store the servo position
 void setup()
              {
myservo.attach(9); // attaches the servo on pin 9 to the servo object
              }
void loop()
 {
 for(pos = 0; pos < 180; pos += 1)          // goes from 0 degrees to 180 degrees
{
                                                  // in steps of 1 degree
 myservo.write(pos);              // tell servo to go to position in variable 'pos'
delay(15); // waits 15ms for the servo to reach the position }
for(pos = 180; pos>=1; pos-=1) // goes from 180 degrees to 0 degrees
{ myservo.write(pos); // tell servo to go to position in variable 'pos'
delay(15); // waits 15ms for the servo to reach the position } }
```

Dual Servo Control

In this project you will control two servos using commands from
the serial monitor.
You can cannibalize the code from Project RGB lamp to make this one!!

Dual Servo

PB view



```
// Project Dual servo
#include <Servo.h>
char buffer[10];
Servo servo1; // Create a servo object
Servo servo2; // Create a second servo object
void setup()
{
servo1.attach(5); // Attaches the servo on pin 5 to the servo1 object
servo2.attach(6); // Attaches the servo on pin 6 to the servo2 object
Serial.begin(9600);
Serial.flush();
servo1.write(90); // Put servo1 at home position
servo2.write(90); // Put servo2 at home postion
Serial.println("STARTING...");
}
void loop()
{
if (Serial.available() > 0) { // Check if data has been entered
int index=0;
delay(100); // Let the buffer fill up
int numChar = Serial.available(); // Find the string length
```

```
if (numChar>10) {
numChar=10;
}
while (numChar--) {
// Fill the buffer with the string
buffer[index++] = Serial.read();
}
splitString(buffer); // Run splitString function
}
}
void splitString(char* data) {
Serial.print("Data entered: ");
Serial.println(data);
char* parameter;
parameter = strtok (data, " ,"); //String to token
while (parameter != NULL) { // If we haven't reached the end of the string...
setServo(parameter); // ...run the setServo function
parameter = strtok (NULL, " ,");
}
// Clear the text and serial buffers
for (int x=0; x<9; x++) {
buffer[x]='\0';
}
```

```
Serial.flush();
}
void setServo(char* data) {
if ((data[0] == 'L') || (data[0] == 'l')) {
int firstVal = strtol(data+1, NULL, 10); // String to long integer
firstVal = constrain(firstVal,0,180); // Constrain values
servo1.write(firstVal);
Serial.print("Servo1 is set to: ");
Serial.println(firstVal);
}
if ((data[0] == 'R') || (data[0] == 'r')) {
int secondVal = strtol(data+1, NULL, 10); // String to long integer
secondVal = constrain(secondVal,0,255); // Constrain the values
servo2.write(secondVal);
Serial.print("Servo2 is set to: ");
Serial.println(secondVal);
}
}
```

Stepper motor

Operation principle of stepper motor.

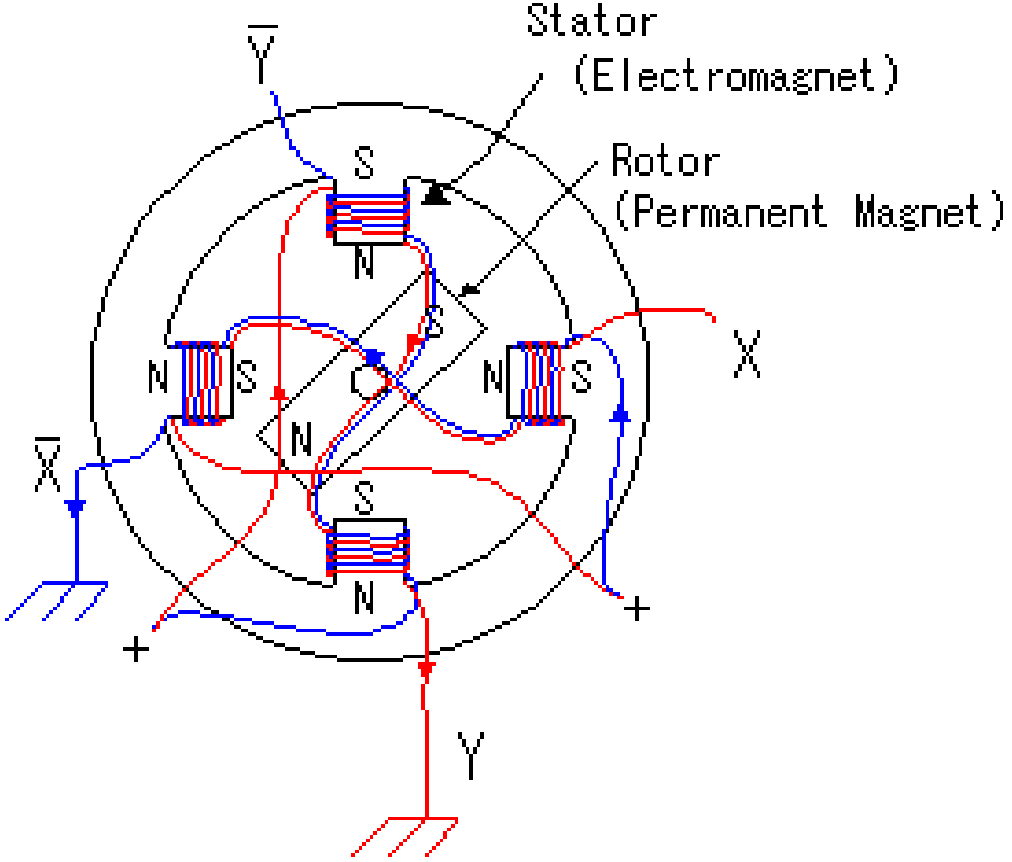
There are many kind of stepper motors.
Unipolar type, Bipolar type, Single-phase type, Multi-phase type...
Single-phase stepper motor is often used for quartz watch.

You will learn about the operation principle of the **2-phase unipolar PM type stepper motor.**

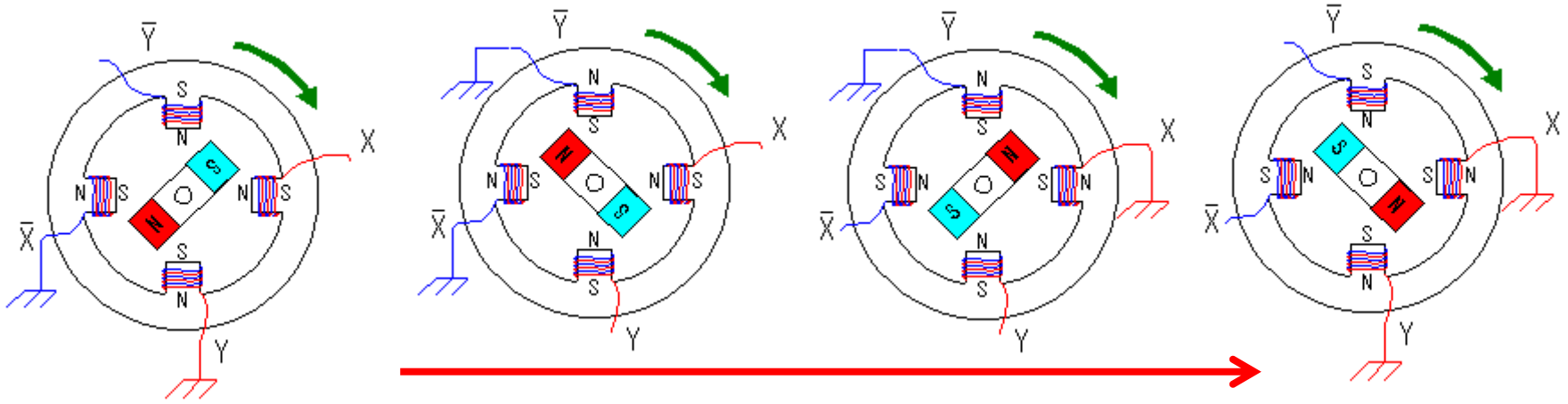
What is a stepper motor?

1- A stepper motor is a brushless, synchronous electric motor that converts digital pulses into mechanical shaft rotation.
2- Every rotation is divided into a discrete number of steps, in many cases 200 steps, and the motor must be sent a separate pulse for each step.
3- The stepper motor can only take one step at a time and each step is the same size, typically 1.8°. The motor's position can be controlled without any feedback mechanism.
4- As the digital pulses increase in frequency, the step movement changes into continuous rotation, with the rotation speed directly proportional to the frequency of the pulses.

In the PM type stepper motor, a permanent magnet is used for rotor and coils are put on stator. The stepper motor model which has 4-poles is shown in the figure. In case of this motor, step angle of the rotor is 90 degrees.



Steps and operation principle of a stepper motor.



X	$\overline{X}$	Y	$\overline{Y}$
0	1	0	1
1	0	0	1
1	0	1	0
0	1	1	0

Clockwise control

X, $\overline{X}$, Y and $\overline{Y}$ are controlled in the following order.

X	$\overline{X}$	Y	$\overline{Y}$	Step angle
0	1	0	1	0°
1	0	0	1	90°
1	0	1	0	180°
0	1	1	0	270°

X	$\overline{X}$	Y	$\overline{Y}$
0	1	0	1
0	1	1	0
1	0	1	0
1	0	0	1

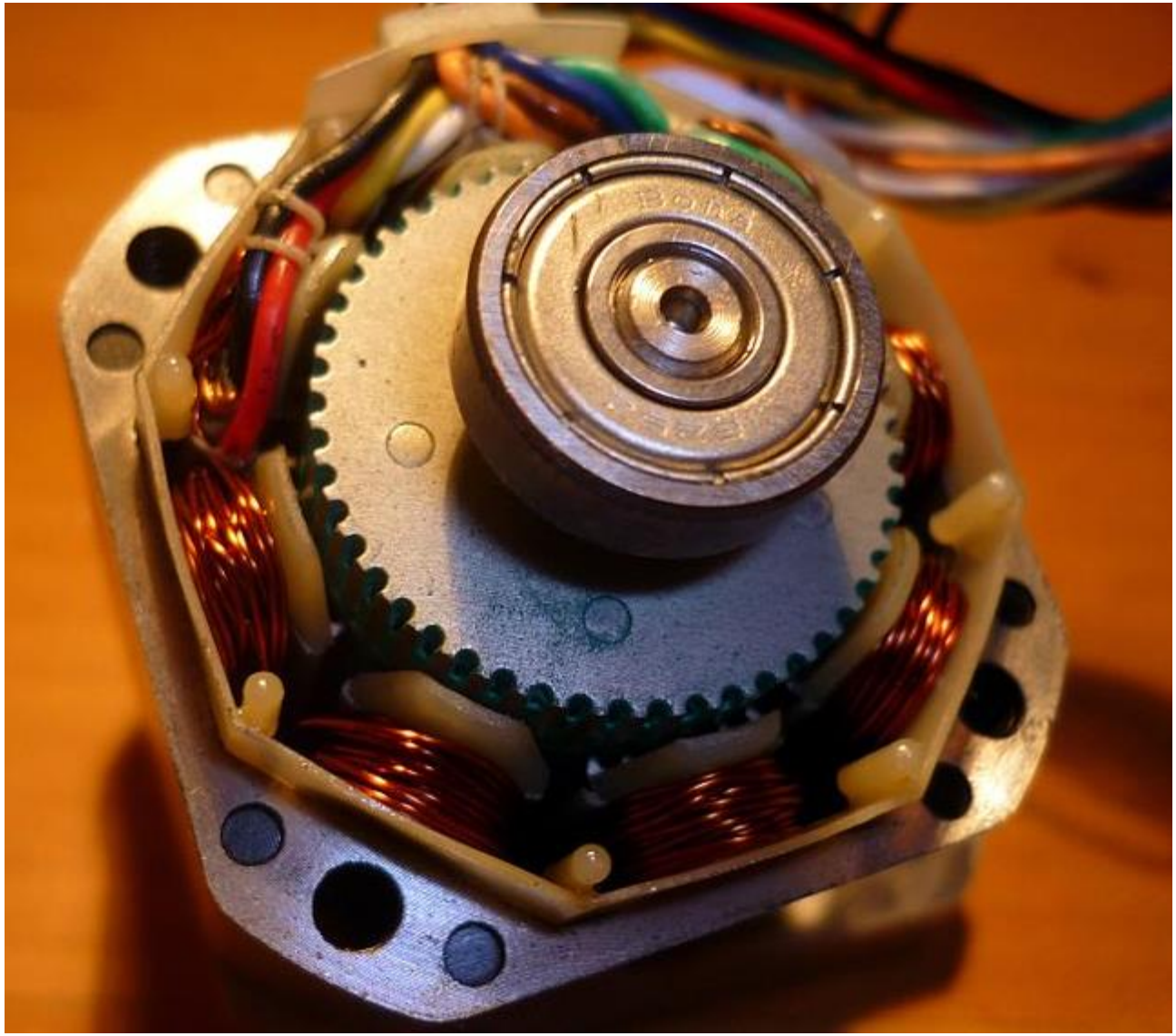
Counterclockwise control

X, $\overline{X}$, Y and $\overline{Y}$ are controlled in the following order.

X	$\overline{X}$	Y	$\overline{Y}$	Step angle
0	1	0	1	0°
0	1	1	0	-90°
1	0	1	0	-180°
1	0	0	1	-270°

"0" means grounding.

http://www.piclist.com/images/www/hobby_elec/e_step1.htm



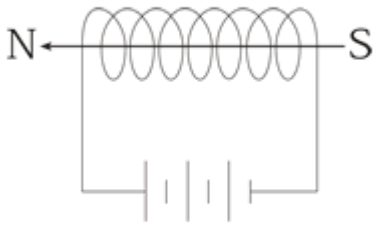


Figure 1.
Magnetic field created by
energizing a coil winding

Figure 2. "One phase on" stepping sequence for two phase motor.

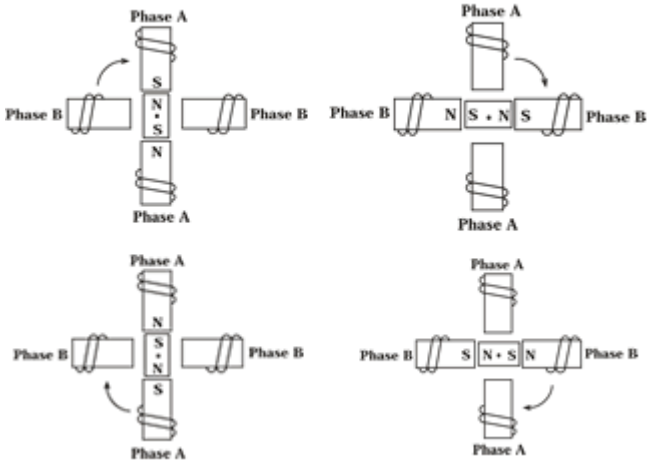
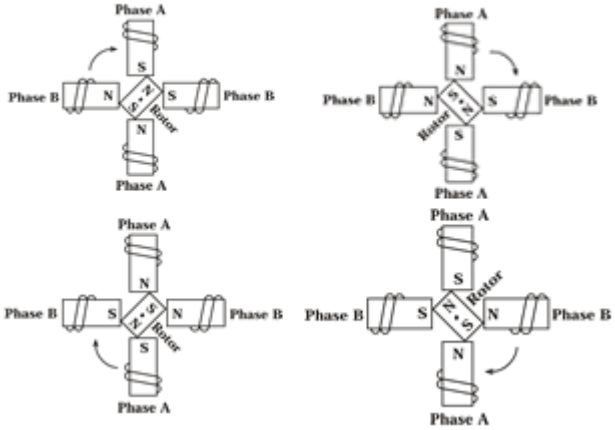


Figure 3. "Two phase on" stepping sequence for two phase motor.



Half stepping

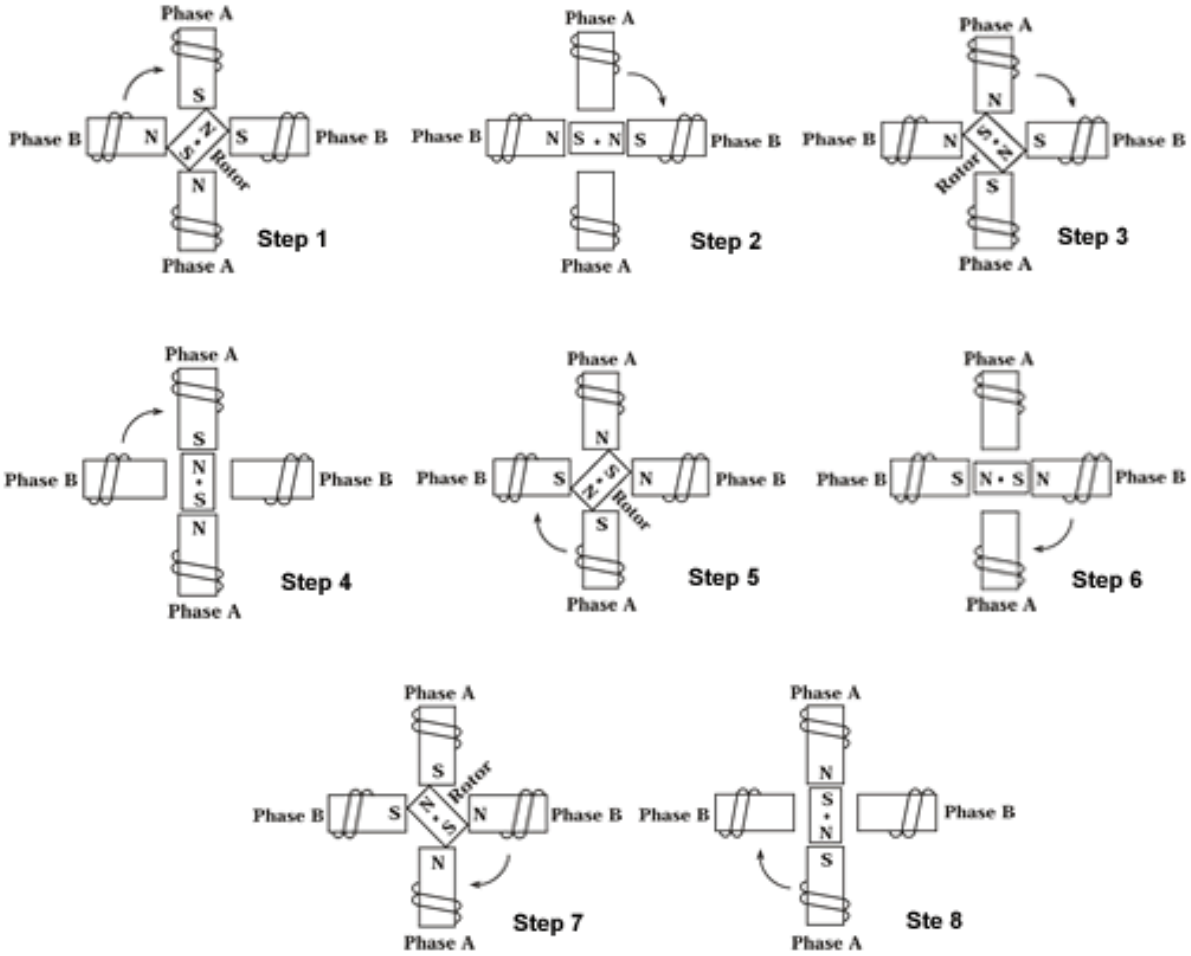
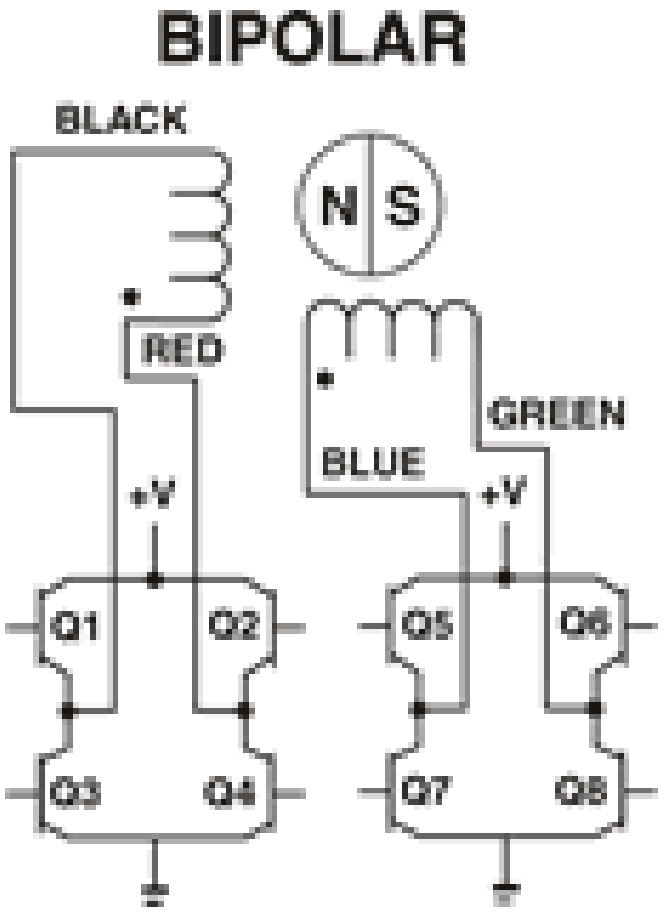


Figure 4. Half-stepping – 90° step angle is reduced to 45° with half-stepping.

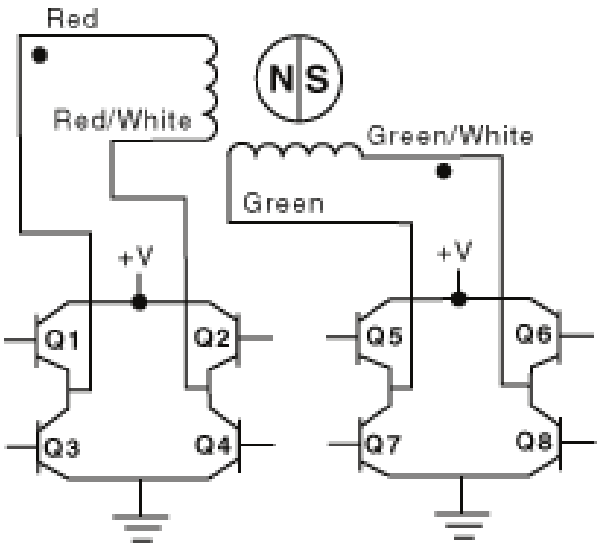
Bipolar winding

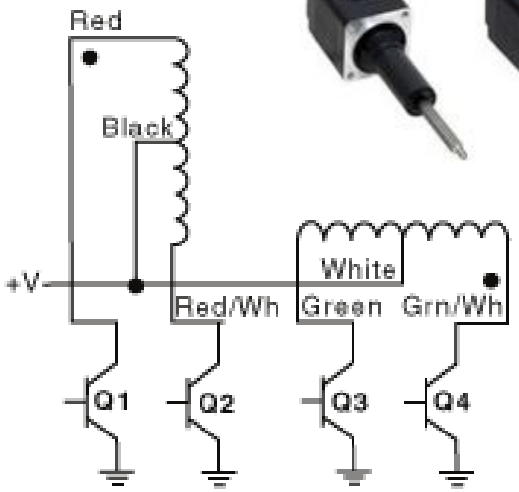
The two phase stepping sequence described utilizes a bipolar coil winding. Each phase consists of a simple winding. Current inversion through transistors keying, causes magnetic polarity inversion.



CW Rotation →	BipolarStep	Q2-Q3	Q1-Q4	Q6-Q7	Q5-Q8	CCW Rotation →
	1	ON	OFF	ON	OFF	
	2	OFF	ON	ON	OFF	
	3	OFF	ON	OFF	ON	
	4	ON	OFF	OFF	ON	
	1	ON	OFF	ON	OFF	

Hybrid Actuator Wiring Diagram





Linear Actuator Types: Captive, External, and Non-Captive

Extend

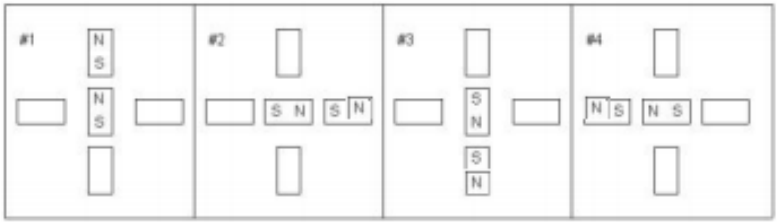
Bipolar	Q2-Q3	Q1-Q4	Q6-Q7	Q5-Q8
Unipolar	Q1	Q2	Q3	Q4
Step	.	.	.	.
1	ON	OFF	ON	OFF
2	OFF	ON	ON	OFF
3	OFF	ON	OFF	ON
4	ON	OFF	OFF	ON
1	ON	OFF	ON	OFF

Retract

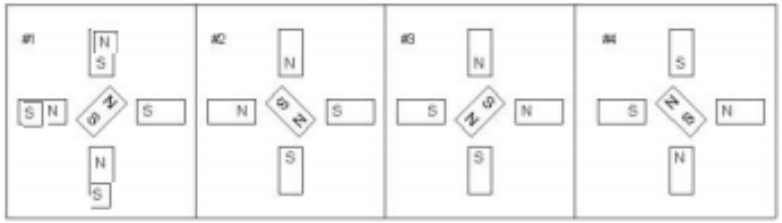
http://www.haydonkerk.com/LinearActuatorProducts/StepperMotorLinearActuators/

```
/*
Controle de motor de passos de duas
bobinas
Rotina simples para rotacionar em uma
direção
 */

void setup() {
  for(int a=2;a<=5;a++)
    pinMode(a, OUTPUT);
}

void loop() {

  int val=10;

  //roda uma vez o motor anti-horario
  digitalWrite(5, HIGH);
  digitalWrite(4, LOW);
  digitalWrite(3, HIGH);
  digitalWrite(2, LOW);
  delay(val);

  digitalWrite(5, LOW);
  digitalWrite(4, HIGH);
  digitalWrite(3, HIGH);
  digitalWrite(2, LOW);
  delay(val);

  digitalWrite(5, LOW);
  digitalWrite(4, HIGH);
  digitalWrite(3, LOW);
  digitalWrite(2, HIGH);
  delay(val);

  digitalWrite(5, HIGH);
  digitalWrite(4, LOW);
  digitalWrite(3, LOW);
  digitalWrite(2, HIGH);
  delay(val);
}
```

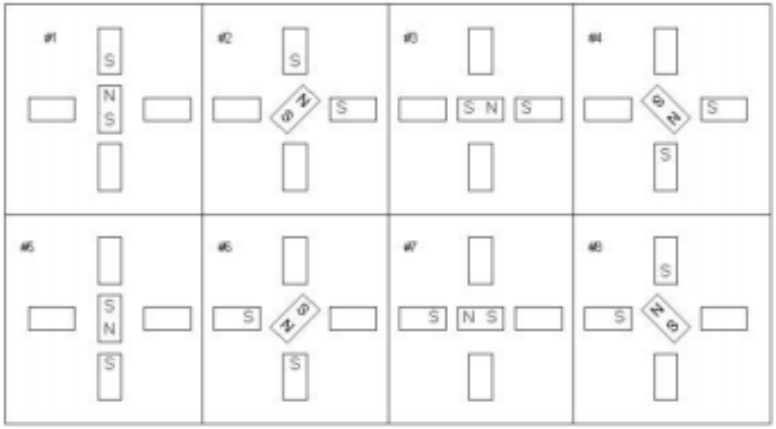
```
//passo 5 fios
int val;
void setup() {
  pinMode(12, OUTPUT);
  pinMode(11, OUTPUT);
  pinMode(10, OUTPUT);
  pinMode(9, OUTPUT);
}

void loop() {

  digitalWrite(12, HIGH);
  digitalWrite(11, LOW);
  digitalWrite(10, LOW);
  digitalWrite(9, LOW);
  delay(val);

  digitalWrite(12, LOW);
  digitalWrite(11, HIGH);
  digitalWrite(10, LOW);
  digitalWrite(9, LOW);
  delay(val);

  digitalWrite(12, LOW);
  digitalWrite(11, LOW);
  digitalWrite(10, HIGH);
  digitalWrite(9, LOW);
  delay(val);

  digitalWrite(12, LOW);
  digitalWrite(11, LOW);
  digitalWrite(10, LOW);
  digitalWrite(9, HIGH);
  delay(val);
}
```



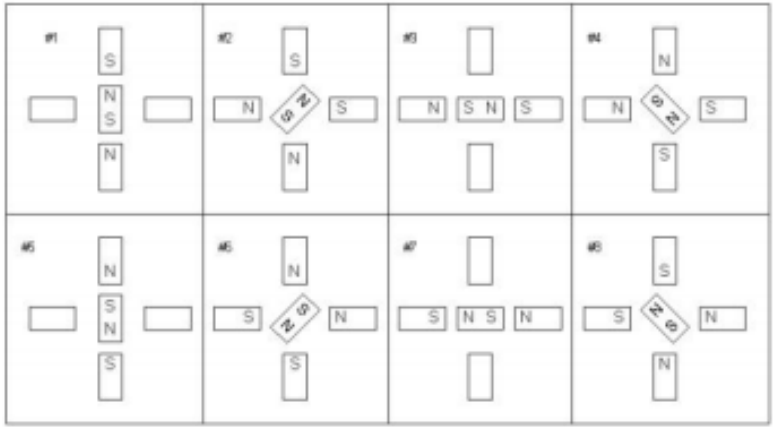
(a) Motor Unipolar de passo inteiro



(b) Motor Bipolar de passo inteiro



(c) Motor unipolar de meio passo



(d) Motor Bipolar de meio passo