OPERATION OF SERVO AND STEPPER MOTORS AND CODING

SET-2

Figure 1. A Servo Motor

By

AAYUSH KUMAR (Electrical)

Servo Motors

There are many types of servo motors and their main feature is the ability to precisely control the position of their shaft. A servo motor is a closed-loop system that uses position feedback to control its motion and final position.

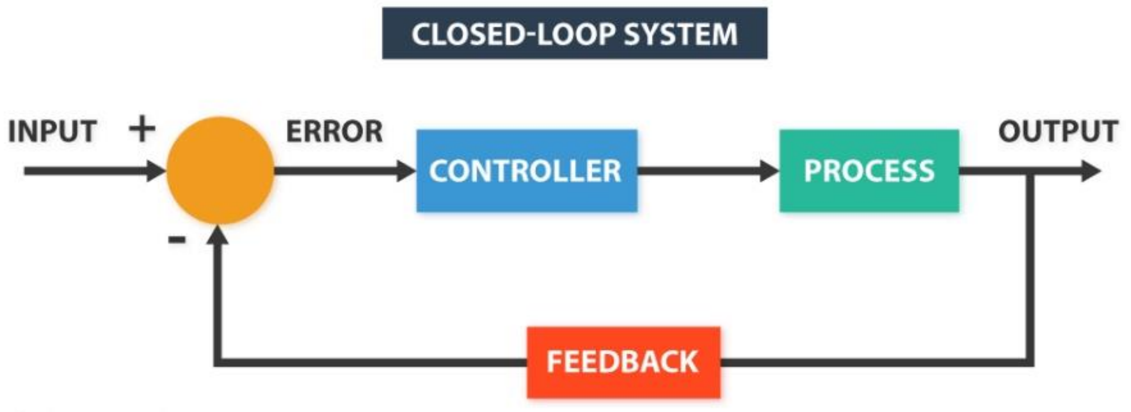


Figure 2. Chart for working of Servo Motor

In industrial type servo motors the position feedback sensor is usually a high precision encoder, while in the smaller RC or hobby servos the position sensor is usually a simple potentiometer. The actual position captured by these devices is fed back to the error detector where it is compared to the target position. Then according to the error, the controller corrects the actual position of the motor to match with the target position.

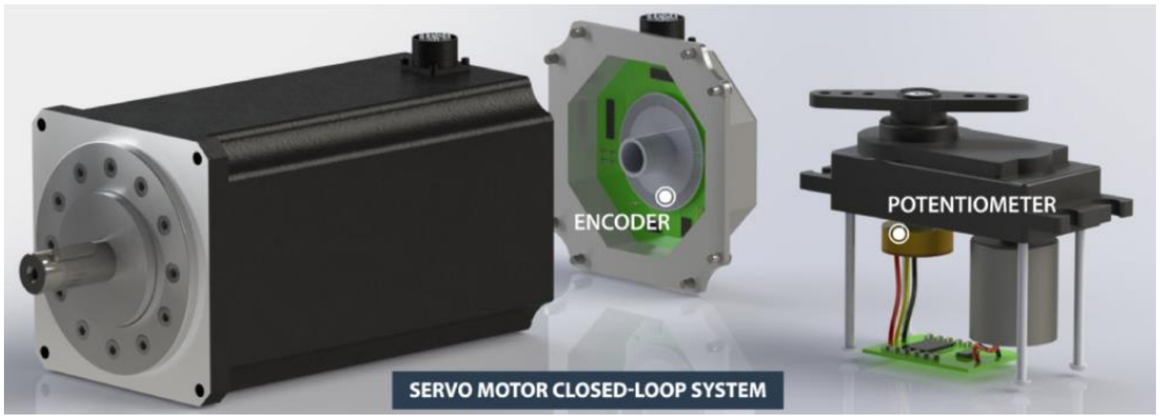


Figure 3. Servo Motor showing Encoder and Potentiometer

Hobby servos are small in size actuators used for controlling RC toys cars, boats, airplanes etc. They are also used by engineering students for prototyping in robotics, creating robotic arms, biologically inspired robots, humanoid robots and so on.

Working of Servo Motors

Inside a hobby servo there are four main components, a DC motor, a gearbox, a potentiometer and a control circuit. The DC motor is high speed and low torque, but the gearbox reduces the speed to around 60 RPM and at the same time increases the torque.

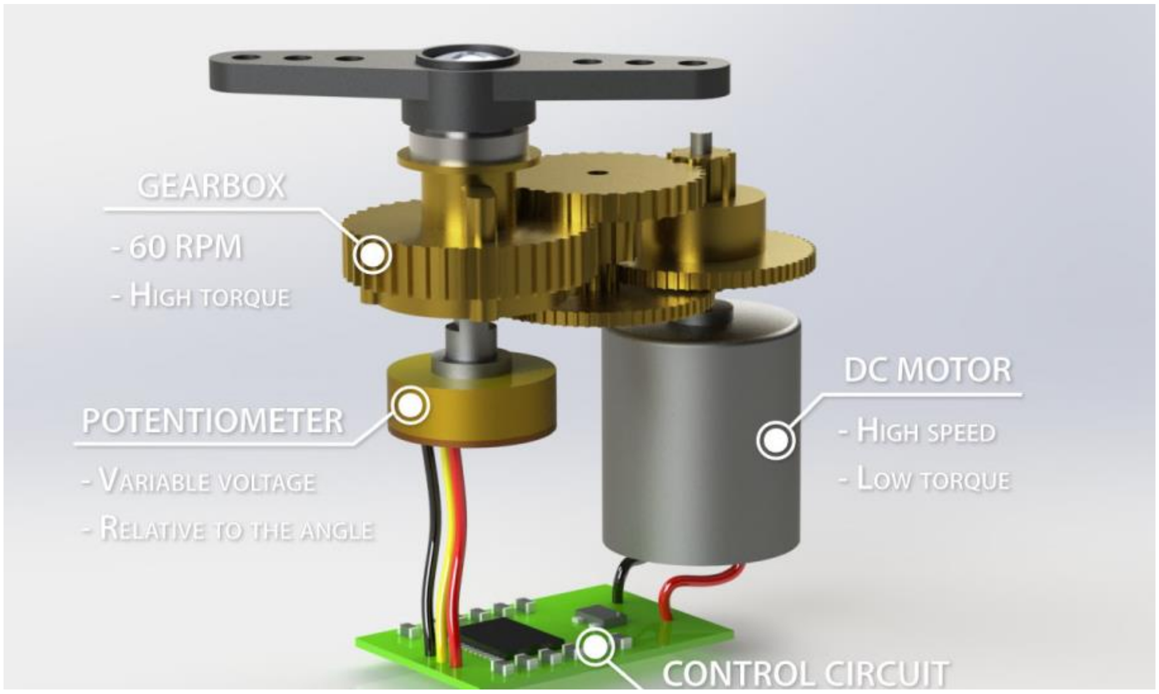


Figure 4. Parts of a Servo Motor

The potentiometer is attached on the final gear or the output shaft, so as the motor rotates the potentiometer rotates as well, thus producing a voltage that is related to the absolute angle of the output shaft. In the control circuit, this potentiometer voltage is compared to the voltage coming from the signal line. If needed, the controller activates an integrated H-Bridge which enables the motor to rotate in either direction until the two signals reach a difference of zero.

A servo motor is controlled by sending a series of pulses through the signal line. The frequency of the control signal should be 50Hz or a pulse should occur every 20ms. The width of pulse determines angular position of the servo and these types of servos can usually rotate 180 degrees (they have a physical limits of travel).

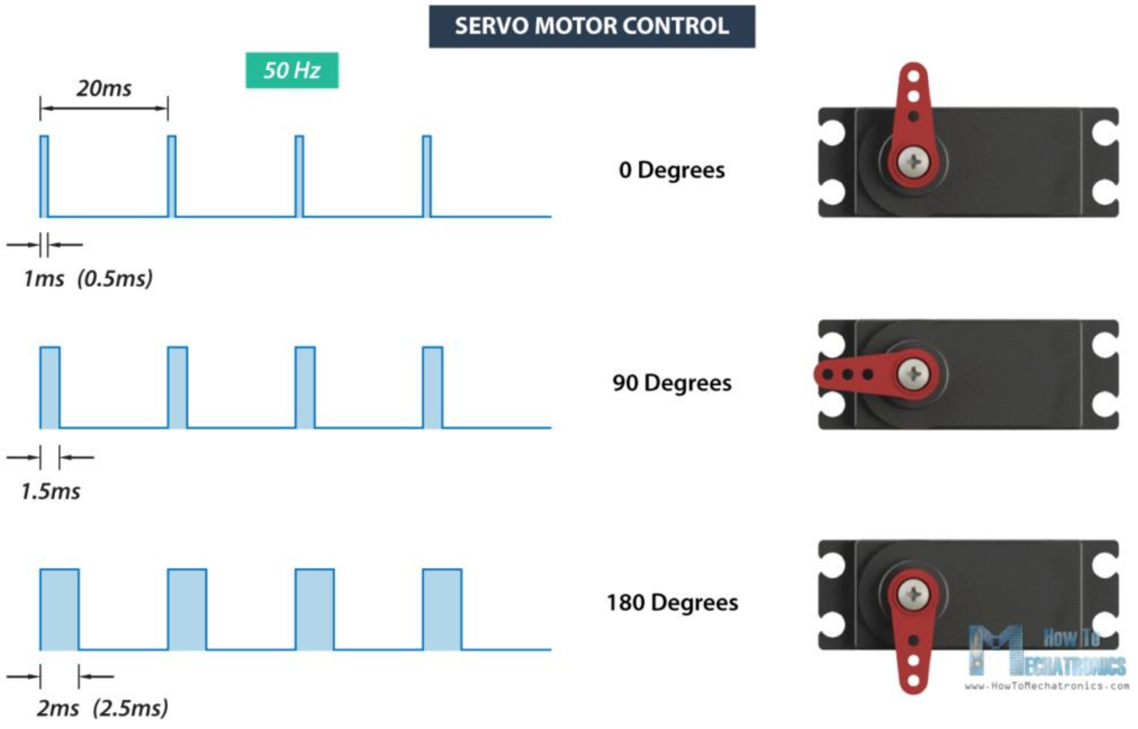


Figure 5. Servo Motor Control

Generally, pulses with 1ms duration correspond to 0 degrees position, 1.5ms duration to 90 degrees and 2ms to 180 degrees. Though the minimum and maximum duration of the pulses can sometimes vary with different brands and they can be 0.5ms for 0 degrees and 2.5ms for 180 degrees position.

Arduino Servo Motors Control

Let's put the above said to test and make a practical example of controlling a hobby servo using Arduino. I will use the MG996R which is a high-torque servo featuring metal gearing with stall torque of 10 kg-cm. The high torque comes at a price and that's the stall current of the servo which is 2.5A. The running current is from 500mA to 900mA and the operating voltage is from 4.8 to 7.2V.

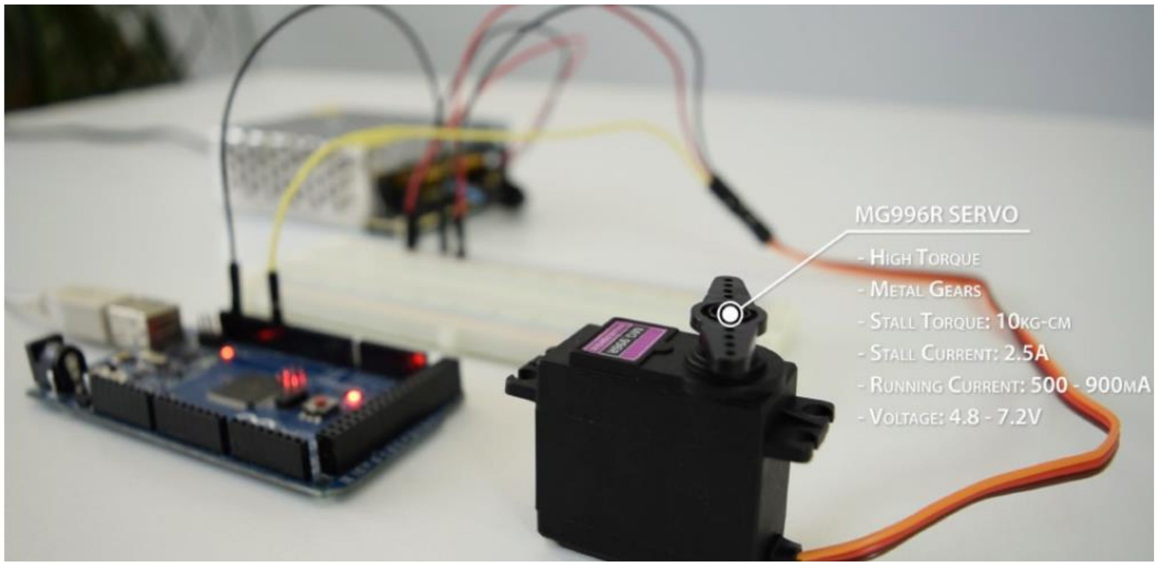


Figure 6. Specifications of the MG996R Servo

The current ratings indicate that we cannot directly connect this servo to the Arduino, but we must use a separate power supply for it. Here's the circuit schematic for this:

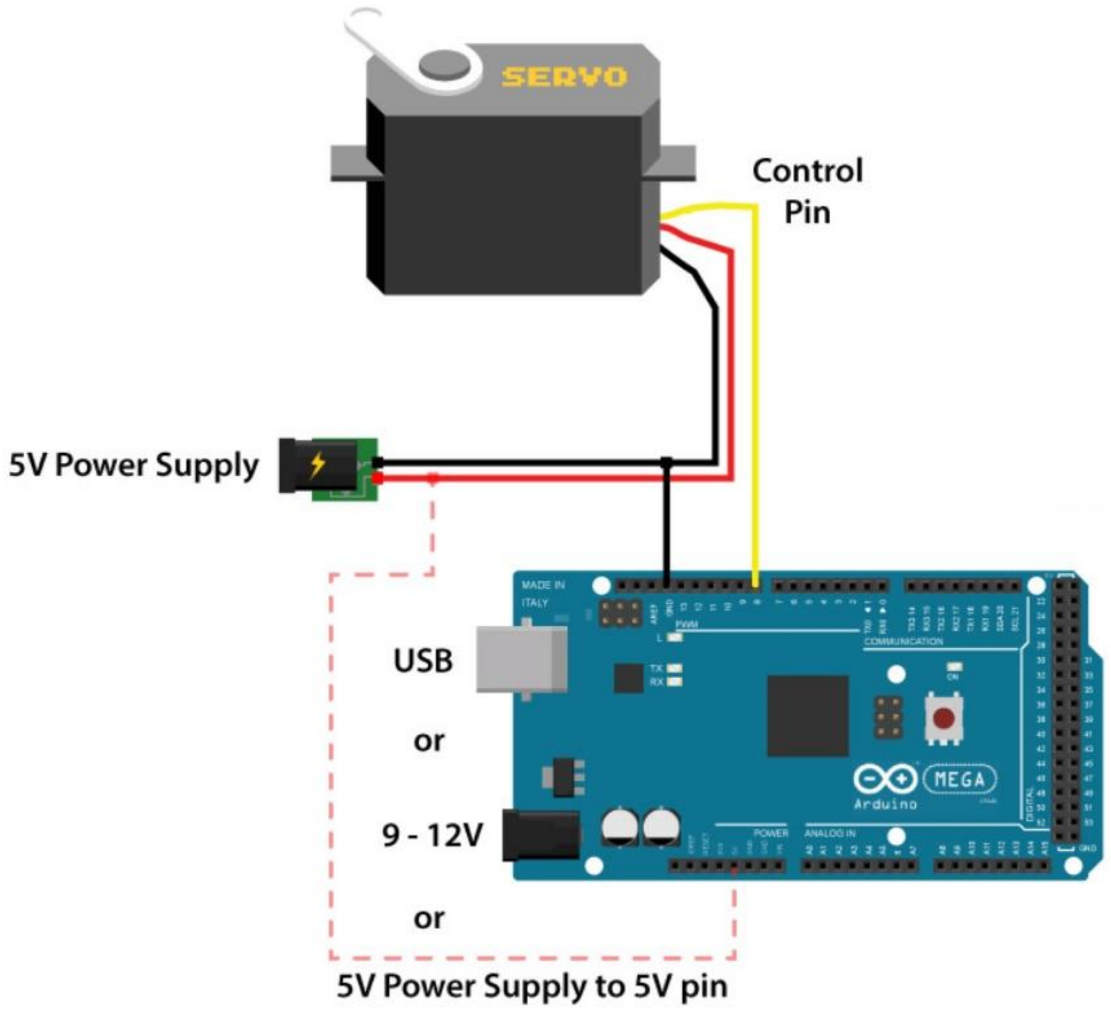


Figure 7. Schematic

We simply need to connect the control pin of the servo to any digital pin of the Arduino board, connect the Ground and the positive wires to the external 5V power supply, and connect the Arduino ground to the servo ground.

Arduino Servo Motor Control Code

```
#include <Servo.h>

Servo myservo; // create servo object to control a servo

void setup() {
myservo.attach(9,600,2300); // (pin, min, max)
}

```

```
void loop() {
myservo.write(0); // tell servo to go to a particular angle
delay(1000);

myservo.write(90);
delay(500);

myservo.write(135);
delay(500);

myservo.write(180);
delay(1500);
}
```

Stepper Motors

The A4988 is a micro stepping driver for controlling bipolar stepper motors which has built-in translator for easy operation. This means that we can control the stepper motor with just 2 pins from our controller, or one for controlling the rotation direction and the other for controlling the steps.

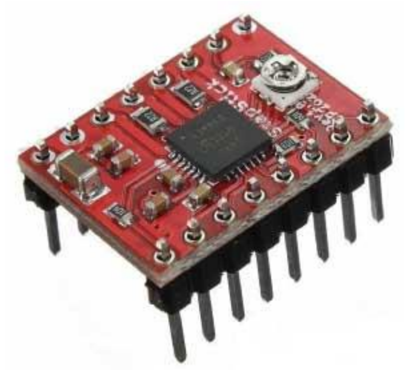
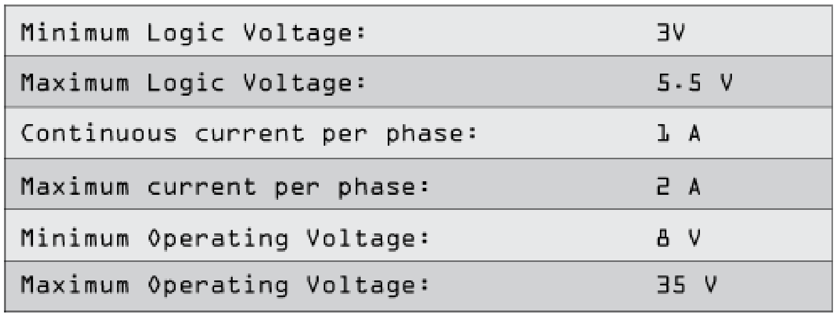


Figure 8. A4988 Driver

The Driver provides five different step resolutions: full-step, haft-step, quarter-step, eight-step and sixteenth-step. Also, it has a potentiometer for adjusting the current output, over-temperature thermal shutdown and crossover-current protection.

Its logic voltage is from 3 to 5.5 V and the maximum current per phase is 2A if good addition cooling is provided or 1A continuous current per phase without heat sink or cooling.



Minimum Logic Voltage:	3V
Maximum Logic Voltage:	5.5 V
Continuous current per phase:	1 A
Maximum current per phase:	2 A
Minimum Operating Voltage:	8 V
Maximum Operating Voltage:	35 V

Figure 9. Specifications of A4988

A4988 Stepper Driver Pinout

Now let’s close look at the pinout of the driver and hook it up with the stepper motor and the controller. We will start with the 2 pins on the button right side for powering the driver, the VDD and Ground pins that we need to connect them to a power supply of 3 to 5.5 V and in our case that will be our controller, the Arduino Board which will provide 5 V. The following 4 pins are for connecting the motor. The 1A and 1B pins will be connected to one coil of the motor and the 2A and 2B pins to the other coil of the motor. For powering the motor, we use the next 2 pins, Ground and VMOT that we need to connect them to Power Supply from 8 to 35 V and, we need to use decoupling capacitor with at least 47 µF for protecting the driver board from voltage spikes.

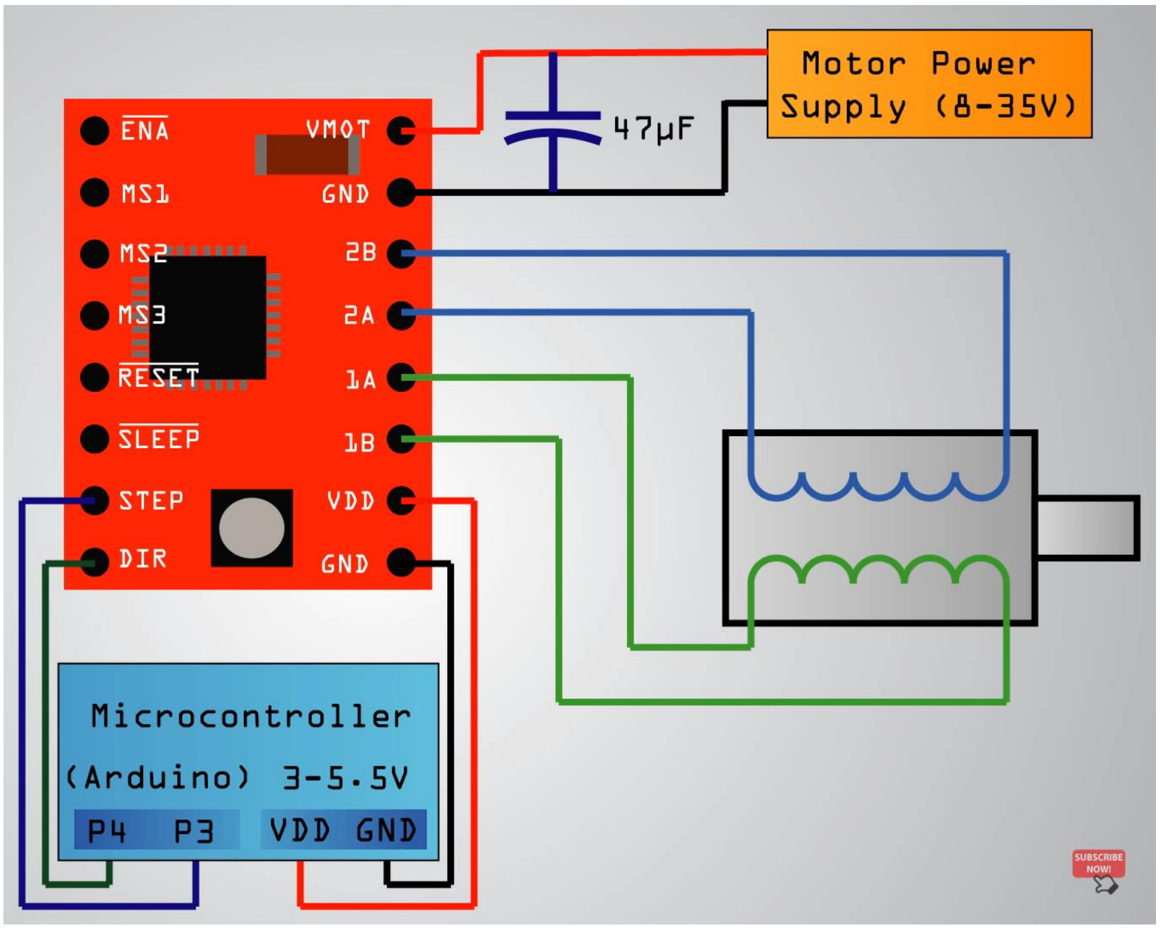
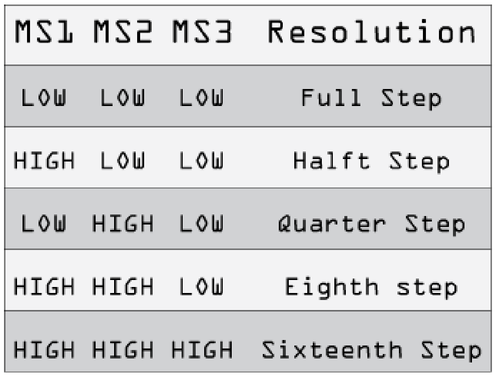


Figure 10. Connections

The next two 2 pins, Step and Direction are the pins that we use for controlling the motor movements. The Direction pin controls the rotation direction of the motor and we need to connect it to one of the digital pins on our microcontroller, or in our case I will connect it to the pin number 4 of my Arduino Board.

With the Step pin we control the mirosteps of the motor and with each pulse sent to this pin the motor moves one step. So that means that we don't need any complex programming, phase sequence tables, frequency control lines and so on, because the built-in translator of the A4988 Driver takes care of everything. Here we also need to mention that these 2 pins are not pulled to any voltage internally, so we should not leave them floating in our Next is the SLEEP Pin and a logic low puts the board in sleep mode for minimizing power consumption when the motor is not in use.

Next, the RESET pin sets the translator to a predefined Home state. This Home state or Home Microstep Position can be seen from these Figures from the A4988 Datasheet. So, these are the initial positions from where the motor starts and they are different depending on the microstep resolution. If the input state to this pin is a logic low all the STEP inputs will be ignored. The Reset pin is a floating pin so if we don't have intention of controlling it with in our program we need to connect it to the SLEEP pin to bring it high and enable the board.



MS1	MS2	MS3	Resolution
LOW	LOW	LOW	Full Step
HIGH	LOW	LOW	Halft Step
LOW	HIGH	LOW	Quarter Step
HIGH	HIGH	LOW	Eighth step
HIGH	HIGH	HIGH	Sixteenth Step

Figure 11. Truth Table

The next 3 pins (MS1, MS2 and MS3) are for selecting one of the five step resolutions according to the above truth table. These pins have internal pull-down resistors so if we leave them disconnected, the board will operate in full step mode.

The last one, the ENABLE pin is used for turning on or off the FET outputs. So, a logic high will keep the outputs disabled.

Circuit Schematics

Here's the complete circuit schematics. I will use the drive in Full Step Mode so I will leave the 3 MS pins disconnected and just connect the Direction and the Step pins of the drive to the pins number 3 and 4 on the Arduino Board and as well the Ground and the 5 V pins for powering the board. Also I will use a 100µF capacitor for decoupling and 12V, 1.5A adapter for powering the motor. I will use a NEMA 17 bipolar Stepper Motor and its wires A and C will be connected to the pins 1A and 1B and the B and D wires to the 2A and 2B pins.

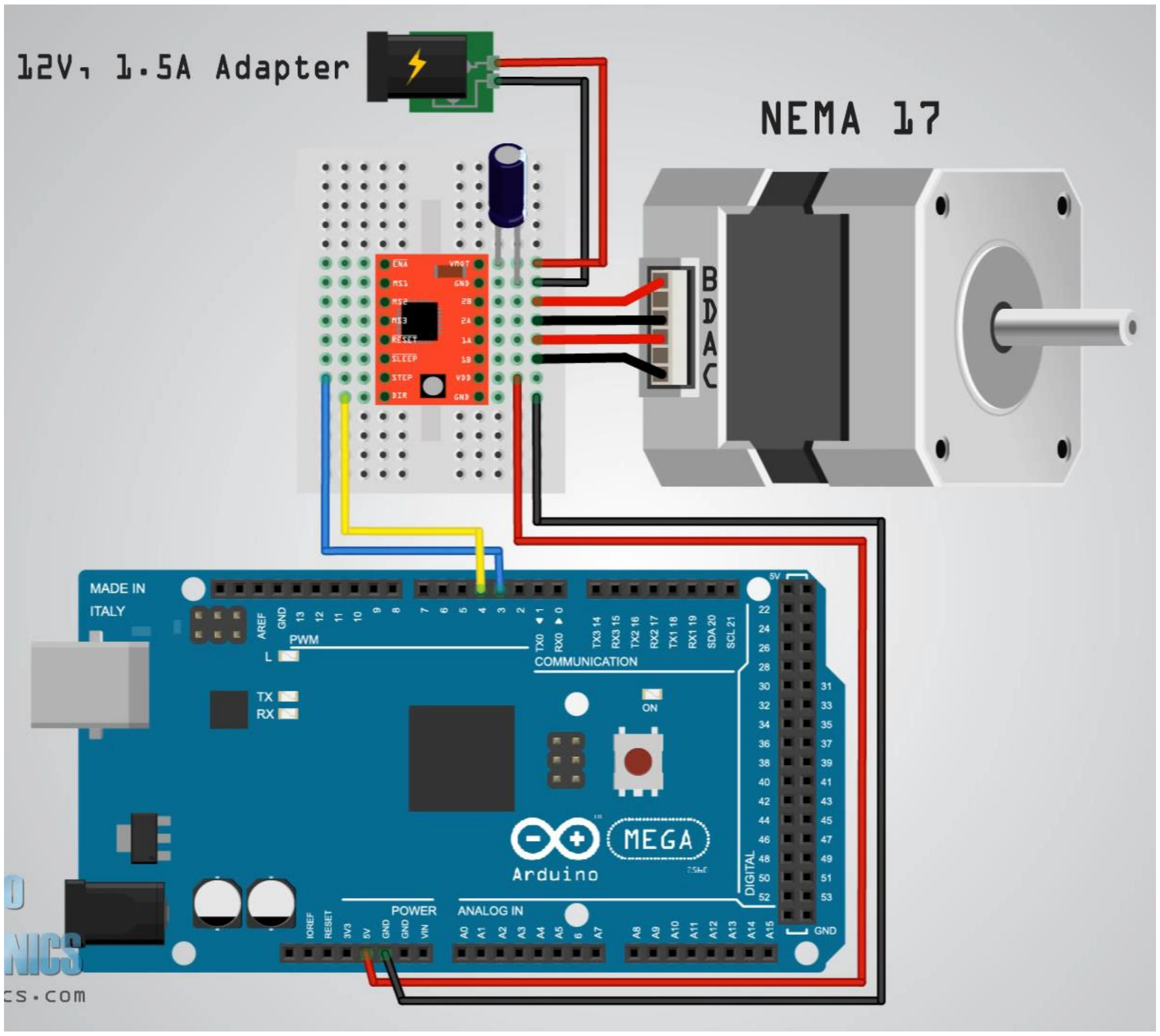


Figure 12. Schematics

Code

```
// defines pins numbers
const int stepPin = 3;
const int dirPin = 4;

void setup() {
// Sets the two pins as Outputs
pinMode(stepPin,OUTPUT);
pinMode(dirPin,OUTPUT);
}
void loop() {
digitalWrite(dirPin,HIGH); // Enables the motor to move in a particular direction
// Makes 200 pulses for making one full cycle rotation
for(int x = 0; x < 200; x++) {
digitalWrite(stepPin,HIGH);
delayMicroseconds(500);
digitalWrite(stepPin,LOW);
delayMicroseconds(500);
}
delay(1000); // One second delay

digitalWrite(dirPin,LOW); //Changes the rotations direction
// Makes 400 pulses for making two full cycle rotation
for(int x = 0; x < 400; x++) {
digitalWrite(stepPin,HIGH);
delayMicroseconds(500);
digitalWrite(stepPin,LOW);
delayMicroseconds(500);
}
delay(1000);
}
```