Experiment #4

Controlling Stepper Motor

Birzeit University
Faculty of Engineering and Technology
Department of Electrical and Computer Engineering

Abstract

This experiment aims at understanding and testing the main concepts of stepper motor operation using Arduino to control or positioning Stepper Motor.

PART I Theoretical and Technical Introduction

1.1 Introduction to Stepper Motors

The stepping motor is a device which can transfer the incoming pulses to stepping motion of a predetermined angular displacement. By using suitable control circuitry, the angular displacement can be made proportional to the number of pulses. Using microcomputer, one can have better control of the angular displacement resolution and angular speed of a stepping motor. In the past few years the stepping motor has improved in size reduction, speed and precision. Stepping motor already have and will continue to have wide applications in industry.

Stepping motors are suitable for translating digital inputs into mechanical motion. In general, there are three types of stepping motor (For detailed Comparison look at Table 1):

1. VR (Variable Reluctance) stepping motors
2. Hybrid Stepping motors
3. PM (Permanent Magnet) Stepping motors

Table 1: Stepping Motors Characteristics Comparison

Motor Type Characteristics	PM	VR	Hybrid
Efficiency	High	Low	High
Rotor Inertia	High	Low	Low
Speed	High	High	Low
Torque	Fair	Low	High
Power O/P	High	Low	Low
Damping	Good	Poor	Poor
Typical Step Angle	1.8°, 15°, 30°	7.5°, 15°, 30°	0.18°, 0.45°

A stepper motor has several coils arranged in a circle, and the rotating shaft is inside the circle. Whenever a coil is excited, the shaft is attracted towards the coil and rotates to point to the coil. Therefore, if there are n coils, there will be n positions that the shaft can rotate to, depending on which coil is excited. If two adjacent coils are excited, the shaft will rotate to in between the two coils. So, there are n more positions that the shaft can rotate to. Altogether, if there are n coils, there are 2n positions.

One-Phase Excitation is when only one coil is excited at a time, so that the shaft rotates from one coil to another. **Two-Phase Excitation** is when two adjacent coils are excited together, so that the coil rotates to positions between the coils. **1-2 Phase Excitation** is when the coils are excited in such a manner that the shaft rotates from a coil to between two coils and vice versa (it uses both 1-phase and 2-phase excitation).

1.2 Single-phase excitation

Figure 1 is used to explain the operation of simplified stepping motor (90°/step). Here the **A** coil and **B** coil are perpendicular to each other. If either **A** or **B** coil is excited (a condition which is known as single-phase excitation), the rotor can be moved to 0°, 90°, 180°, 270°-degree position depending on the current's ON/OFF conditions in the coils. Table 1 clarifies more the stepping motor operation.

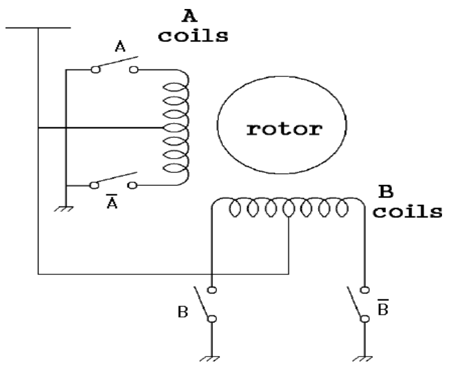
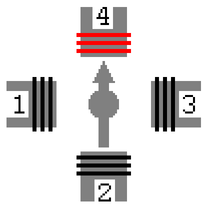
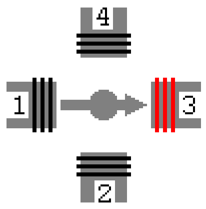
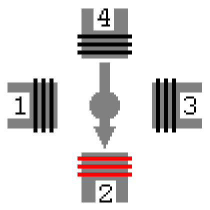
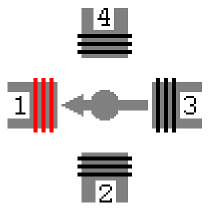


Figure 1 Bipolar Stepper motor Coil schematic

Table 2 Stepper Motor Clockwise Rotation One-Phase

Step	**Coil 4**	**Coil 3**	**Coil 2**	**Coil 1**	
a.1	**On**	off	Off	off	
a.2	Off	**on**	off	off	

a.3	Off	off	**on**	off	
a.4	Off	off	off	**on**	

From Table 2 we can derive stepper motor excitation values. Table 2 shows excitation **values** for stepper motor with starting position at coil A (clockwise). Fill Table3 (Pre-lab)

Table 3: Excitation table of One-Phase excitation (clockwise)

	A	B	A'	B'	HEX
Step1	0	1	1	1	07H
Step2	1	0	1	1	0BH
Step3	1	1	0	1	0DH
Step4	1	1	1	0	0EH

1.3 Two–Phase Excitation

Two-Phase excitation is when two adjacent coils are excited together, so that the coil rotates to positions between the coils. In **Two-Phase** excitation, both coils have current flowing at the same time, then the rotor positions can be 45°, 135°, 225°, 315° degrees as shown in Figure 3. This is known as two-phase exception.

In 2-1 Phase excitation, the excitation alternates between 1-phase and 2-phase, then the motor will rotate according to 0°, 45°, 90°, 135°, 180°, 225°, 270°, 315° sequence. This is 1-2 phase excitations; each step distance is only half of step movement of either One-Phase or Two-Phase excitation.

Figure 2 illustrates clockwise of Two-Phase excitation.

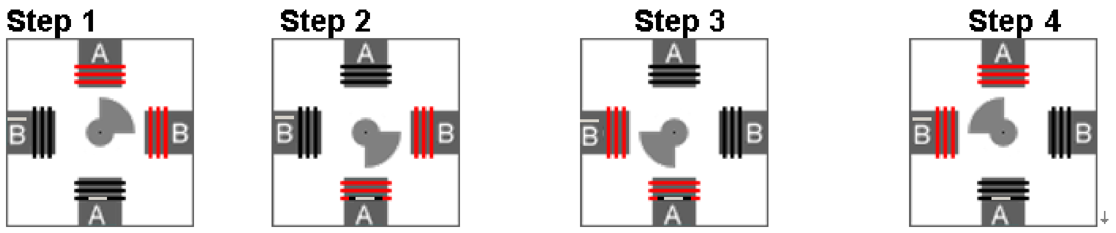


Figure 2 Clockwise Two-Phase Excitation

Commercial stepping motor uses multi-motor rotor, the rotor features two gear like PM cylinders that are turned one-half of tooth spacing. One gear is South Pole; the other gear is North Pole. If a 50-tooth rotor gear is used, the following movement sequence will proceed.

A. Single-phase excitation:

The stepping position will be 0°, 1.8°, 3.6°, ..., 358.2°, total 200 steps in one round.

B. Two-phase excitation:

The stepping position will be 0.9°, 2.7°, 4.5°, ..., 359.1°, total 200 steps in one round.

C. Single-phase and two-phase excitations combined:

The stepping position will be 0°, 0.9°, 1.8°, 2.7°, 3.6°, 4.5°, ..., 358.2°, 359.1°, total 400 steps in one round.

Since stepping motor makes step-by-step movement and each step is equidistant, the rotor and stator magnetic fields must be synchronous. During start-up and stopping, the two fields may not be synchronous, so it is suggested to slowly accelerate and decelerate the stepping motor during the start-up or stopping period.

PART II Practices:

PRACTICE I: Controlling and Positioning Stepper Motor using LED's

Step1: Connect the circuit shown in Figure 3 below using Tinkercad such that each led represent one coil from stepper motor.

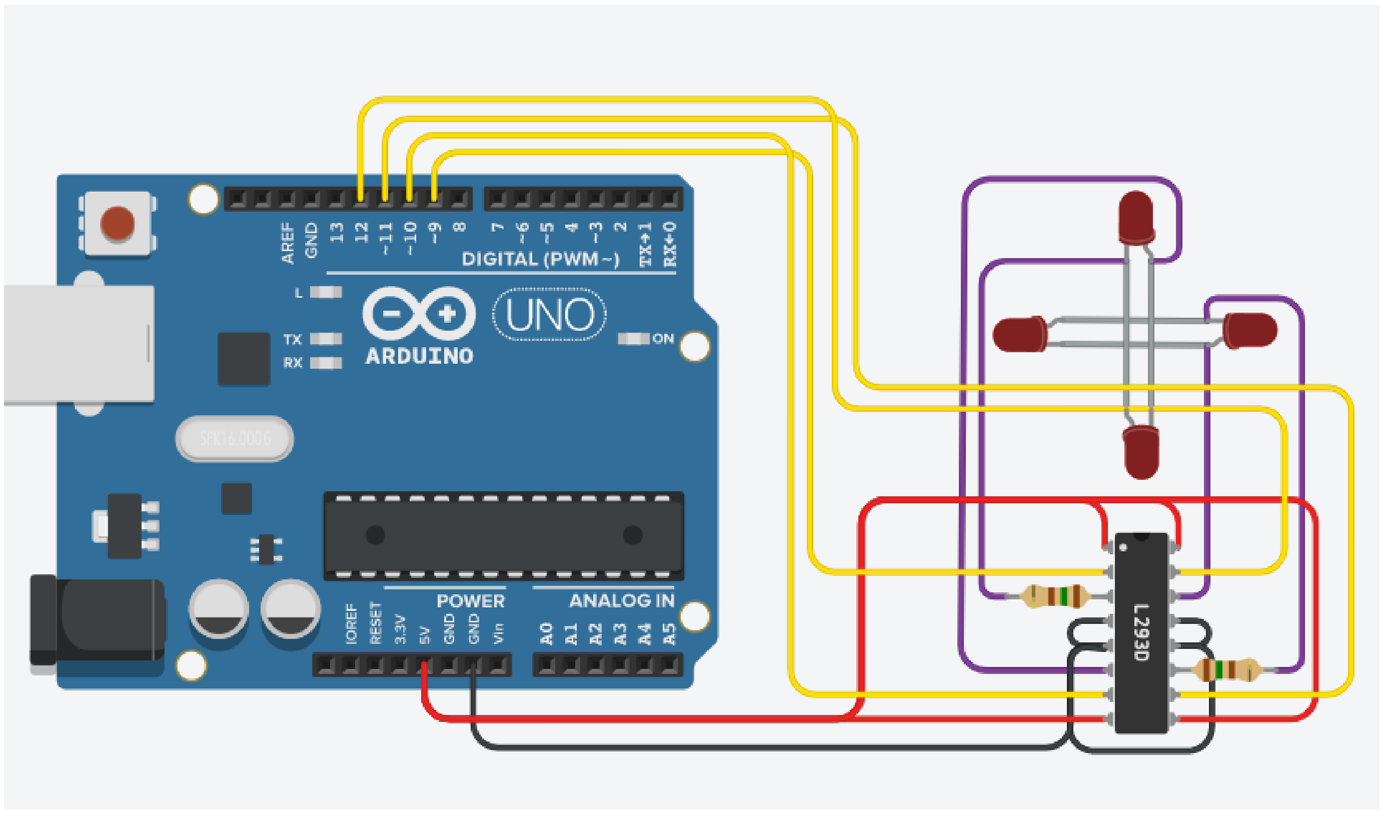


Figure3: stepper motor using Arduino

Step2: Write the following code and run it.

```
// command pins for stepper motor windings (A+, A-, B+, B-)
#define ApPin 9
#define AmPin 10
#define BpPin 11
#define BmPin 12

// activation period in µs
#define stepperPeriod 250000 // 0.25 s == 4 steps/s

// phases count

unsigned long stepperNextDate;        // Next rendez-vous in µs
char stepperPhase = 0; // memorize which stepper motor coil to supply
```

```
void setup() {
  pinMode(ApPin,OUTPUT); pinMode(AmPin,OUTPUT);
  pinMode(BpPin,OUTPUT); pinMode(BmPin,OUTPUT);
  unsigned long now = micros(); // initialize next rendezvous
  stepperNextDate = now + stepperPeriod;
}

void loop() {
  bool doStepper;
  do { // wait for the upcoming rendezvous
    long int now = micros();
    doStepper = (signed long)(now - stepperNextDate) >= 0;
  } while (!doStepper);

  if (doStepper) { // stepper motor processing
    // set the date of the next stepper rendezvous
    stepperNextDate += stepperPeriod;
    // update the stepping phase
    stepperPhase = stepperPhase + 1;
    if (stepperPhase > 3) stepperPhase = 0;
    // command the stepper motor coils accordingly
    switch (stepperPhase) {
      case 0: { digitalWrite(BmPin,LOW); digitalWrite(ApPin,HIGH); } break;
      case 1: { digitalWrite(ApPin,LOW); digitalWrite(BpPin,HIGH); } break;
      case 2: { digitalWrite(BpPin,LOW); digitalWrite(AmPin,HIGH); } break;
      case 3: { digitalWrite(AmPin,LOW); digitalWrite(BmPin,HIGH); } break;
    } // switch
  } // doStepper
} //loop()
```

TASKS:

1. Explain what does this code do?
2. In which excitation phase does the motor operate?
3. Modify the code let the stepper motor rotate in opposite direction (clockwise)

PRACTICE III: Tinkercad Stepper motor

Step1: connect the circuit bellow (in fig4) in Tinkercad.
Component used:
1. Potentiometer.
2. DC motor with Encoder.
3. Arduino.
4. Breadboard.
5. Wires.

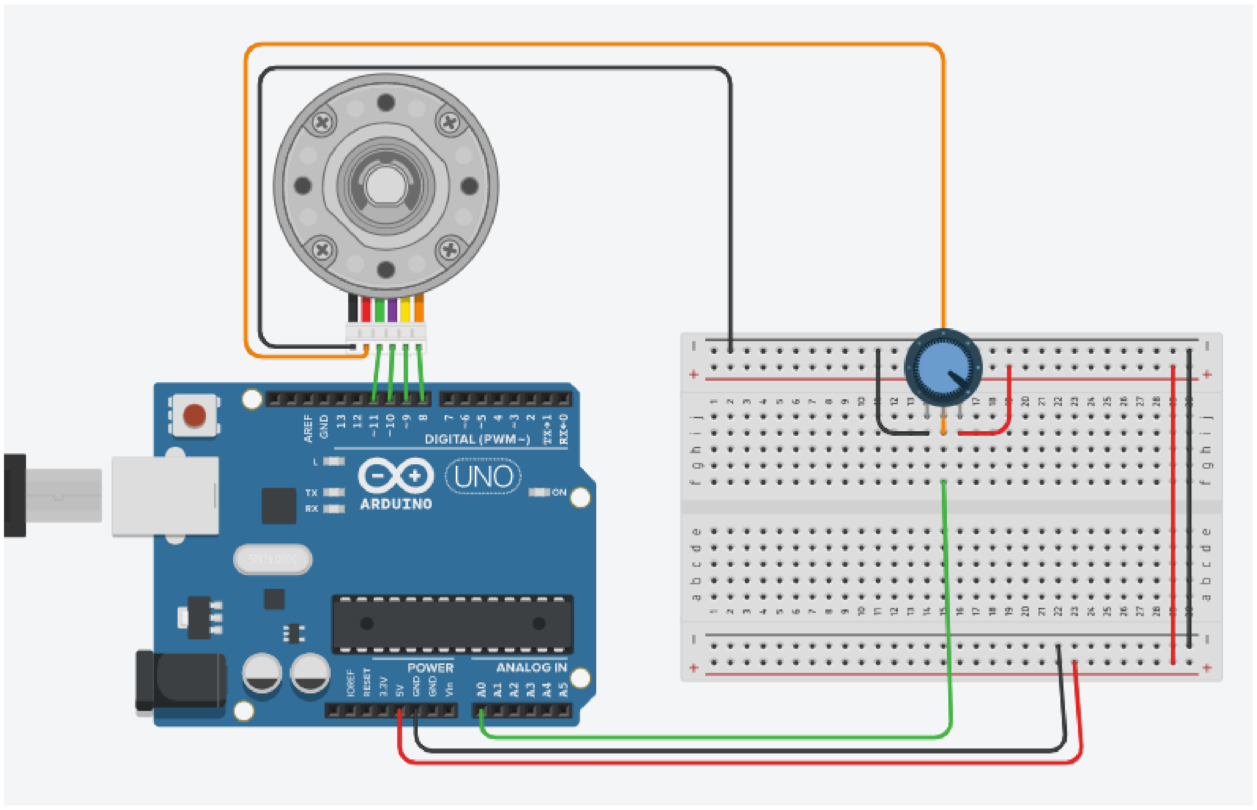


Figure4 : Tinkercad Stepper motor.

Step2: Write the following code and run it.

```
#include <Stepper.h>

const int stepsPerRevolution = 200;  // change this to fit the number of steps per revolution
// for your motor


// initialize the stepper library on pins 8 through 11:
Stepper myStepper(stepsPerRevolution, 8, 9, 10, 11);

int stepCount = 0;  // number of steps the motor has taken
```

```
void setup() {
  // nothing to do inside the setup
}

void loop() {
  // read the sensor value:
  int sensorReading = analogRead(A0);
  // map it to a range from 0 to 100:
  int motorSpeed = map(sensorReading, 0, 1023, 0, 100);
  // set the motor speed:
  if (motorSpeed > 0) {
    myStepper.setSpeed(motorSpeed);
    // step 1/100 of a revolution:
    myStepper.step(stepsPerRevolution / 100);
  }
}
```