Mechatronics Lab
Experiment 10 report
Stepper Motor Motion control using a Arduino Micro-controller

Purvag lapsiwala UIN:662689378
Date: 7/Feb/2018

Contents

1 Summary

Designing a system for motion control of a stepper motor using the PIC micro-controller and a step motor control integrated circuit(IC). Write a code using the Arduino IDE to control-

- Magnitude of speed
- Direction of speed
- Run the motor in full-step mode and half step mode.

2 List of Components

1.Arduino Uno

2.LED (4)

3.Switch

4.Transistor

5.Stepper motor

6.Diode

7.ULN 2003A transistor array

3 Description of the experiment

3.1 Purpose and Objective

Purpose of the experiment is to learn the Operating principle of stepper motor and its control in using the micro-controller. Second part of the experiment to control the speed of the stepper motor using the Stepper.h library in the arduino IDE. In the last part of the experiment we have coded the stepper motor to function in the half step and full step mode and compare the two modes in torque and speed.

3.2 Theory

3.2.1 Stepper Motor

A stepper motor rotates one step per change in the energized state of its stator windings. The stepper motor used in the experiment is a unipolar, 2-phase (center tapped), 48 steps/revolution, 5 VDC stepper motor The 28BYJ-48 is a small, cheap, 5 volt geared stepping motors. These stepping motors are apparently widely used to control things like automated blinds, A/C units and are mass produced. stepper01Due to the gear reduction ratio of *approximately* 64:1 it offers decent torque for its size at speeds of about 15 rotations per minute (RPM). With some software trickery to accelerate gradually and a higher voltage power source (I tested them with 12 volts DC) I was able to get about 25+ RPM. These little steppers can be purchased together with a small breakout board for the Arduino compatible ULN2003 stepper motor driver for less than 5 dollars. The low cost and small size makes the 28BYJ-48 an ideal option for small robotic applications, and an excellent introduction to stepper motor control with Arduino. Hereare

the detailed specs of the 28BYJ-48 stepper motor.

- Step angle:
  Half-step mode: 8 step control signal sequence (recommended) 5.625 degrees per step / 64 steps per one revolution of the internal motor shaft
  Full Step mode: 4 step control signal sequence 11.25 degrees per step / 32 steps per one revolution of the internal motor shaft
- Gear ratio:
  Manufacturer specifies 64:1. But on the Arduino forums have disassembled the gear train of these little motors and determined that the exact gear ratio is in fact 63.68395:1. My observations confirm their findings. These means that in the recommended half-step mode we will have:64 steps per motor rotation x 63.684 gear ratio = **4076 steps** per full revolution (approximately).

4 Circuits Diagrams and Images

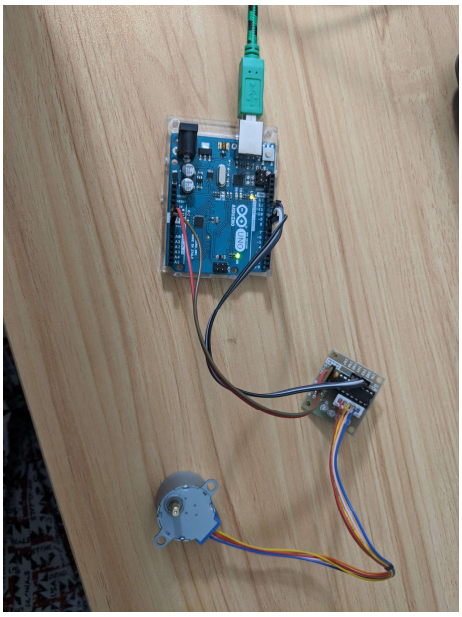

Figure 1: Here in this circuit instead of using the EDE1200 we have attached the ARDUINO UNO directly to the transistor array

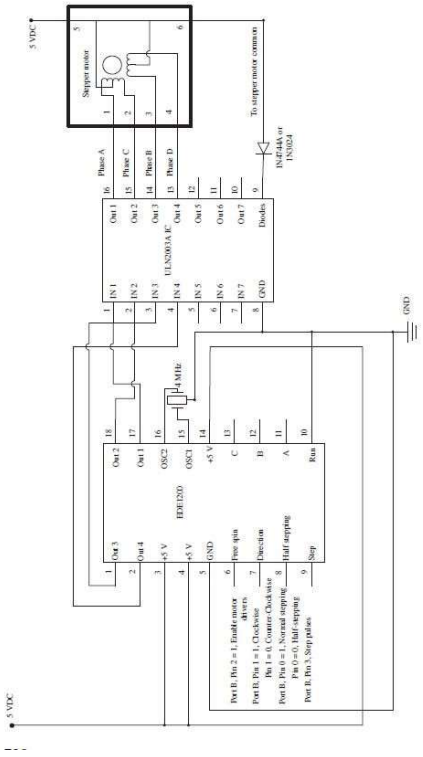

Figure 2: This is the actual circuit diagram

5 Procedure

1. The first step of this experiment is to identify the wiring connection of the stepper motor.
2. Second step is to connect the stepper motor to the transistor array and the transistor array to the stepper motor as shown in the circuit diagram. i.e pins 8,9,10 ad 11 are connected to the transistor array.
3. Third step is to program a the stepper motor using the stepper.h library and control the speed of the motor and direction of the motor.
4. Next step is to implement the half step (micro-stepping) using direct coding with out any library.
   for this procedure we used Digitalwrite HIGH and LOW of the pins 8,9,10 and 11.
5. we have compared the standing torque and running torque difference between the full step and half step mode using the code given below

6 Results and Code

6.1 Full step using stepper.h

Full step one direction is programmed using the stepper.h library

```
#include <Stepper.h>
Stepper myStepper(200 , 8, 9, 10, 11);

void setup() {

  myStepper.setSpeed(100);
}

void loop() {
myStepper.step(48*48);
}
```

Figure 3: Code with stepper.h library

6.2 Direction Control and speed control

Direction of the stepper can be controller using stepper.h by giving the negative step but the direction of the stepper can also be changed by change the pins 8,9,10 and 11 to 11,10,9 and 8 respectively.
To control the speed of the motor in stepper library we have to change the speed by calling '.setSpeed' method of the instance created in using the stepper library. Here in the example both have speed control method.

```
#include <Stepper.h>


Stepper myStepper(200 , 8, 9, 10, 11);

void setup() {

  myStepper.setSpeed(100);
}

void loop() {
myStepper.step(48*48);

 delay(500);

myStepper.step(-48*48);

  delay(500);

}
```

Figure 4: Code used for direction control with negative step

```
#include <Stepper.h>
Stepper myStepper(200 , 11, 10, 9, 8);

void setup() {

  myStepper.setSpeed(100);
}

void loop() {
myStepper.step(48);
}
```

Figure 5: Code for the direction control by changing the pins

6.3 Half step

Half step (also known as micro-stepping) is a technique of energizing the phases in the stepper to achieve better precision compared to half step,here the half step is done using two ways first is the code in basic HIGH and LOW of the pins are made to get the required combination. Next type is more complex but the basic idea remains same but in this code both the direction and speed can be controlled in the. Fig 8 and Fig 9 gives the two ways of achieving this.

```
void setup() {
  pinMode(8,OUTPUT);
  pinMode(9,OUTPUT);
  pinMode(10,OUTPUT);
  pinMode(11,OUTPUT);
}

void loop() {
  digitalWrite(8,HIGH);
  digitalWrite(9,LOW);
  digitalWrite(10,LOW);
  digitalWrite(11,LOW);
  delay(10);
  digitalWrite(8, HIGH);
  digitalWrite(9,HIGH);
  digitalWrite(10,LOW);
  digitalWrite(11,LOW);
  delay(10);
  digitalWrite(8,LOW);
  digitalWrite(9,HIGH);
  digitalWrite(10,LOW);
  digitalWrite(11,LOW);
  delay(10);
  digitalWrite(8,LOW);
  digitalWrite(9,HIGH);
  digitalWrite(10,HIGH);
  digitalWrite(11,LOW);
  delay(10);
  digitalWrite(8,LOW);
  digitalWrite(9,LOW);
  digitalWrite(10,HIGH);
  digitalWrite(11,LOW);
  delay(10);
  digitalWrite(8,LOW);
  digitalWrite(9,LOW);
  digitalWrite(10,LOW);
  digitalWrite(11,HIGH);
  delay(10);
  digitalWrite(8,HIGH);
  digitalWrite(9,LOW);
  digitalWrite(10,LOW);
  digitalWrite(11,HIGH);
}
```

Figure 6: Basic code for the half step.

```
#define IN1 8
#define IN2 9
#define IN3 10
#define IN4 11
const int NBSTEPS = 2048;
const int STEPTIME = 900;
int Step = 0;
boolean Clockwise = true;

int arrayDefault[4] = {LOW, LOW, LOW, LOW};

int stepsMatrix[8][4] = {
  {LOW, LOW, LOW, HIGH},
  {LOW, LOW, HIGH, HIGH},
  {LOW, LOW, HIGH, LOW},
  {LOW, HIGH, HIGH, LOW},
  {LOW, HIGH, LOW, LOW},
  {HIGH, HIGH, LOW, LOW},
  {HIGH, LOW, LOW, LOW},
  {HIGH, LOW, LOW, HIGH},
};

unsigned long lastTime = 0L;
unsigned long time = 0L;

void writeStep(int outArray[4]);
void stepper();
void setDirection();
```

```
void setup() {
  //Arduino UNO
  Serial.begin(9600);
  Serial.println("Starting...");
  pinMode(IN1, OUTPUT);
  pinMode(IN2, OUTPUT);
  pinMode(IN3, OUTPUT);
  pinMode(IN4, OUTPUT);
}

void loop() {
  unsigned long currentMicros;
  int stepsLeft = NBSTEPS;
  time = 0;
  lastTime = micros();
  while (stepsLeft > 0) {
    currentMicros = micros();
    if (currentMicros - lastTime >= STEPTIME) {
      stepper();
      time += micros() - lastTime;
      lastTime = micros();
      stepsLeft--;
    }
    delay(1);
  }
  Serial.println(time);
  Serial.println("Wait...!");
  delay(400);
  Clockwise = !Clockwise;
```

Figure 7: Code for half step and also direction change

```
  stepsLeft = NBSTEPS;
}

void writeStep(int outArray[4]) {
  digitalWrite(IN1, outArray[0]);
  digitalWrite(IN2, outArray[1]);
  digitalWrite(IN3, outArray[2]);
  digitalWrite(IN4, outArray[3]);
}

void stepper() {
  if ((Step >= 0) && (Step < 8)) {
    writeStep(stepsMatrix[Step]);
  } else {
    writeStep(arrayDefault);
  }
  setDirection();
}

void setDirection() {
  (Clockwise == true) ? (Step++) : (Step--);

  if (Step > 7) {
    Step = 0;
  } else if (Step < 0) {
    Step = 7;
  }
}
```

Figure 8: continued Code for half step and also direction change

7 Conclusion

In Conclusion, in this experiment we understood the basic working principle of the unipolar stepper motor and also we create code to perform direction and speed control of the stepper motor. We perform the task in two ways first using the library called stepper.h and next by changing the pin configuration in software. In the next part of experiment we manually code the logic of the half step. We have also used logical operations and coded micro-stepping of the stepper motor and achieved direction control. We understood the how stepper motor can provide added controllability when used in the control system.

8 Videos

Please visithereor visit the pagehttps://photos.app.goo.gl/eeqOoW1igr2kaNaz1

9 References

1.Lab manual.