University of Hull
Department of Computer Science
C4DI
Fun with Steppers

Vsn. 1.0 Rob Miles 2014

Introduction

Welcome to our Arduino hardware sessions.

Please follow the instructions carefully. If you get the wiring wrong your programs will not work and there is a good chance that you will destroy the delicate circuitry in the device that you are using. We are using RedBoard devices from SparkFun. These are interchangeable with Arduino devices, and so when you read Arduino I really mean the SparkFun RedBoard.

In this session you will learn a bit about electronics and how to control simple circuits using an Arduino device. Here are a few conventions used in the text.

This indicates a warning to be careful about this bit. If you get it wrong it might be time to buy a new device.

This indicates an activity you should perform in at this point in the text. You may be given precise instructions, or you may have to work something out for yourself.

This indicates something that you may want to think about later.

There are two parts to this work. We have to make the circuit (build the hardware) and then we have to create a program to use the devices (write the software).

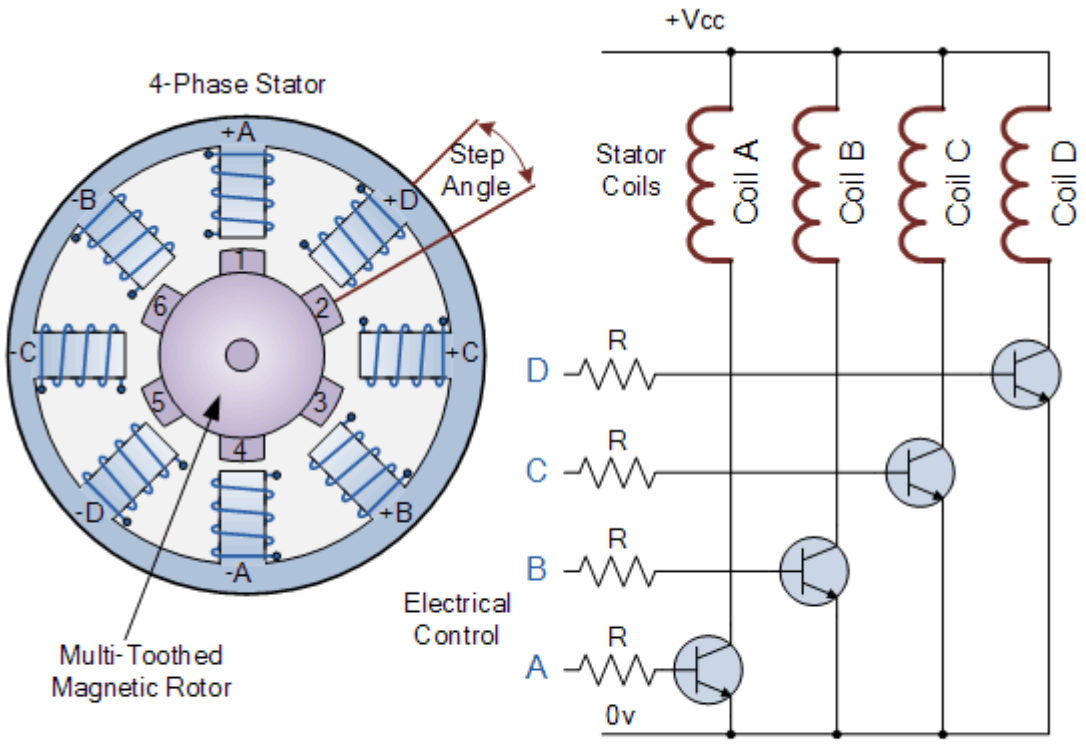
So far we have just created programs that make lights or sounds. Now we want some real, physical action. So we are going to take a look at stepper motors.

Stepper Motors

A stepper motor is an electric motor that moves in, er, steps. An electric motor works by using magnetic attraction. A magnetic field in a coil is attracted to another magnetic field. In a cheap Direct Current (dc) motor one of the magnets is a permanent magnet and the other is a coil wrapped around the part in the middle that can rotate. There is usually more than one coil inside the magnetic field, so that by switching the power through each coil in turn we can get a continuous movement of the rotor. This switching is performed by a series of brushes which transfer power into the rotating coil. The switching is performed by a component called *commutator*.

The motor that starts your car works a bit like this, but it contains extra coils to replace the weaker permanent magnets and make a stronger magnetic field around the rotor.

Stepper motors use magnetic attraction to move, but they turn the design inside out. Rather than having the coils in the middle of the motor, they put them round the outside and then put a magnet in the middle.

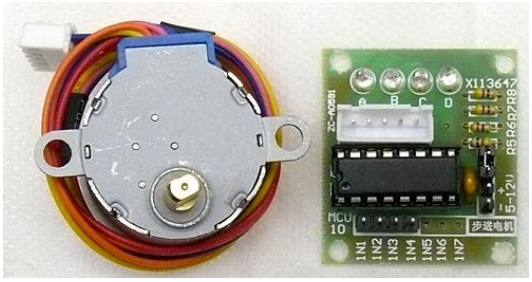


This figure shows how a stepper motor works. The coils are arranged around the outside of the motor, with a permanent magnet rotor in the middle. Each of the coils can be magnetised. If you do this in the right sequence you can make the rotor in the middle "chase" the moving magnetic field and turn.

This means that something has to switch the signals going into the coils to make the rotor spin. If the signals are switched too quickly, or in the wrong order this will not actually do any damage, but the rotor will not turn.

Wiring up the Motor Driver

The Arduino pins do not have enough power to drive the coils in the motor directly, we need a circuit to amplify them. On the right of the diagram you can see roughly what the circuit looks like. The good news is that the board we have will perform the amplification for us.



You should have a motor and a board like the ones above. The driver board may look slightly different, but it should have pins labelled IN1, IN2, IN3 and IN4. These correspond to the signals for the four coils you can see above. The board should also have a pair of pins that you can use for power input. One should be marked + and the other – The board also has some LEDs which will confirm when the coils are being driven. The motor plugs into the board.

The connections we are going to use are as follows:

Connection	Arduino Pin
IN1	8
IN2	9
IN3	10
IN4	11
- (ground)	GND
+	5 Volts

Connect the driver board to your Arduino.

You will need to use jumper cables that have a socket on one end (for the driver board) and a plug on the other (for the Arduino).

Make sure you get the wiring right, particularly the power connection, as otherwise you may have to buy a new Arduino/stepper motor driver.

Writing the Software

We can now use a program that will switch the signals at an appropriate rate and in the correct sequence to make the motor turn.

```
// This Arduino example demonstrates bidirectional operation of a
// 28BYJ-48, using a ULN2003 interface board to drive the stepper.

// The 28BYJ-48 motor is a 4-phase, 8-beat motor, geared down by

// a factor of 68. One bipolar winding is on motor pins 1 & 3 and

// the other on motor pins 2 & 4. The step angle is 5.625/64 and the
 // operating Frequency is 100pps. Current draw is 92mA.
 ////////////////////////////////////////////////

//declare variables for the motor pins
 int motorPin1 = 8;    // Blue   - 28BYJ48 pin 1
 int motorPin2 = 9;    // Pink   - 28BYJ48 pin 2
 int motorPin3 = 10;    // Yellow - 28BYJ48 pin 3
 int motorPin4 = 11;    // Orange - 28BYJ48 pin 4
                         // Red    - 28BYJ48 pin 5 (VCC)

int motorSpeed = 1200;  //variable to set stepper speed
int count = 0;          // count of steps made
int countsperrev = 512; // number of steps per full revolution
int lookup[8] = {B01000, B01100, B00100, B00110, B00010, B00011, B00001,
B01001};

////////////////////////////////////////////////////////////////////////////
//////
```

```
 void setup() {
   //declare the motor pins as outputs
   pinMode(motorPin1, OUTPUT);
   pinMode(motorPin2, OUTPUT);
   pinMode(motorPin3, OUTPUT);
   pinMode(motorPin4, OUTPUT);
   Serial.begin(9600);
 }

////////////////////////////////////////////////////////////////////////////
//////
 void loop(){
   if(count < countsperrev )
     clockwise();
   else if (count == countsperrev * 2)
     count = 0;
   else
     anticlockwise();
   count++;
 }

////////////////////////////////////////////////////////////////////////////
//////
 //set pins to ULN2003 high in sequence from 1 to 4
 //delay "motorSpeed" between each pin setting (to determine speed)
 void anticlockwise()
 {
   for(int i = 0; i < 8; i++)
   {
     setOutput(i);
     delayMicroseconds(motorSpeed);
   }
 }

void clockwise()
 {
   for(int i = 7; i >= 0; i--)
   {
     setOutput(i);
     delayMicroseconds(motorSpeed);
   }
 }

void setOutput(int out)
 {
   digitalWrite(motorPin1, bitRead(lookup[out], 0));
   digitalWrite(motorPin2, bitRead(lookup[out], 1));
   digitalWrite(motorPin3, bitRead(lookup[out], 2));
   digitalWrite(motorPin4, bitRead(lookup[out], 3));
 }
```

This program will make the motor turn first one way, and then the other.

Enter the program and run it..

To make it easier to see the shaft turning we have some labels you can stick onto it.

Investigating the Motor Driver

The code contains a number of constants that control how the rotor is made to rotate. There is also an array of bit patterns that gives the sequence in which the motors should be switched on and off. These have been set for the motor that you have, it might be fun to try and see what they do.

What do you think would be the effect of changing the value of the `motorSpeed` delay? Try the values 1000 and 1600

If the number is made larger the motor will move more slowly. If the number is decreased this doesn't actually make the motor faster as the rotor is not able to move from one pole to the next and the motor just buzzes.

Making a Robotic Finger of Doom

The `countsperrev` variable does what it says on the tin. It combines the number of steps required for the motor to rotate with the gearing which steps down the motor movement to give you a number that is the number of steps that are required for a full revolution.

We could make a "finger of doom" that turned the rotor a random amount to select a particular victim stood around the stepper. The Arduino provides a random method that will produce a number in a given range:

```
long doomValue;
doomValue = random(512,1000);
```

This would produce a value in the range 512 to 1000. If you used this to control the movement of the motor it would spin once and then stop at a random position.

Create a doom based game.

Rob Miles 2014