**using a Stepper motor with arduino controller**

Introduction

A stepper motor is a unique type of motor where the stator windings can be separately excited to precisely control the motion of rotor made of permanent magnets. Since there are several separate windings and they each have their own control, there are several wires connected to a stepper motor. Depending on how one chooses to excite the windings a stepper motor can be used for continuous rotation or precise positioning with digital signals from a microcontroller.

It can be controlled in an open-loop system configuration with no feedback indicating its position. This is in contrast to a servo motor system that does provide positional (or speed) feedback.

You will:

* Interface your stepper to the Arduino board
* Interface your joystick to the Arduino board
* Use stick to control motor speed/direction
* Use the push button to increment stepper one step

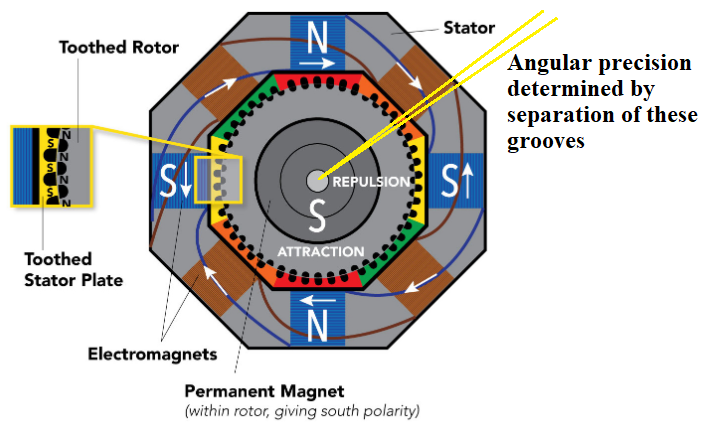
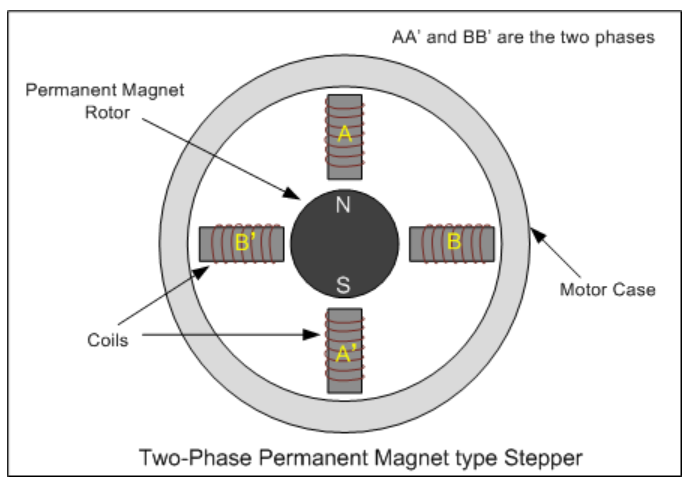
MATERIALS

* Makerspace Kit
  + Datasheet for 28YBJ-48 Stepper Motor
  + Datasheet for ULN2003 Stepper Motor Driver
  + Joystick
* Optional: O-scope
* Optional: Power Supply.

Background

Stepper Motor

The internal arrangement of a stepper motor is shown in figure 1. Activation of the stator windings applies a torque to the permanent magnet or iron core rotor. The sequence of excitation windings must be coordinated in such a way that the rotor moves between the smallest units. A mcro controller can ensure that the sequence is precisely what it needs to be.



1. b.

Figure 1. a. Simplified two pole stepper. Only 90 degree control b. Cross section of motor showing how precise angular control can be achieved.

There are different configurations of stator winding: bipolar and unipolar. The unipolar uses a center tap to maximize the number of excitation coils while minimizing the number of winding connections. As shown in figure 2. there are 5 wires required to energize 4 windings. In order to energize the coils in the sequence A – B – C – D, then pins 1, 2, 3 and 4 would need to be activated in that sequence. Any other order would cause the motor to stall or move in the opposite direction or simply jerk beyond the minimum angular rotation angle. Current can only be in one direction through the motor. The 5V source provides the power and current is sunk in each of the output coils.

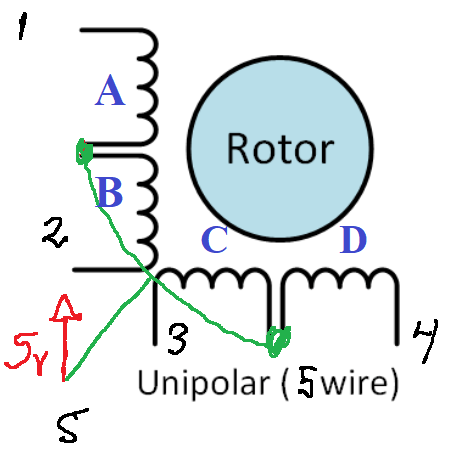


Figure 2. 5-wire unipolar stepper motor winding arrangement.

https://www.youtube.com/watch?v=eyqwLiowZiU

Website reference for stepper.

<https://www.aranacorp.com/en/control-a-stepper-motor-with-arduino/>

An amplifier is required to run the stepper. The micro controller can neither source the required current nor sink it. The windings can require several hundred milliamps and that’s beyond the capability of the MCU.

The circuit diagram for our stepper motor is shown in figure 3.

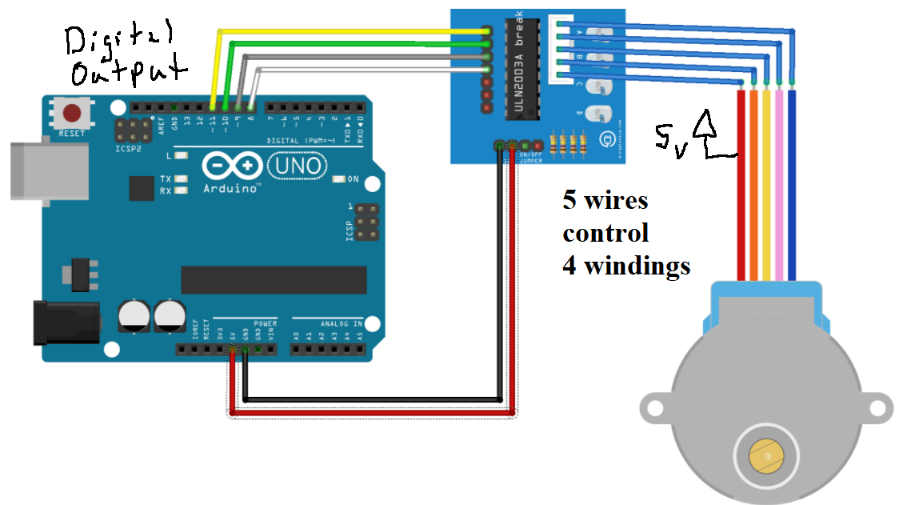


Figure 3. Wiring for stepper motor and MCU.

Joystick

The operation for a simple joystick is shown in figure 4. VRx,y outputs are effectively potentiometers/voltage dividers. They can use analog inputs to measure the position of the stick. We’ll use only MapX to control the motor.

The stick can also be pushed like a push button with output a SW. We’ll use that to move the motor one unit forward.

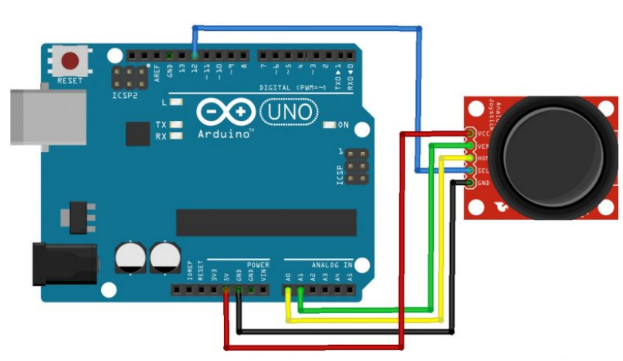
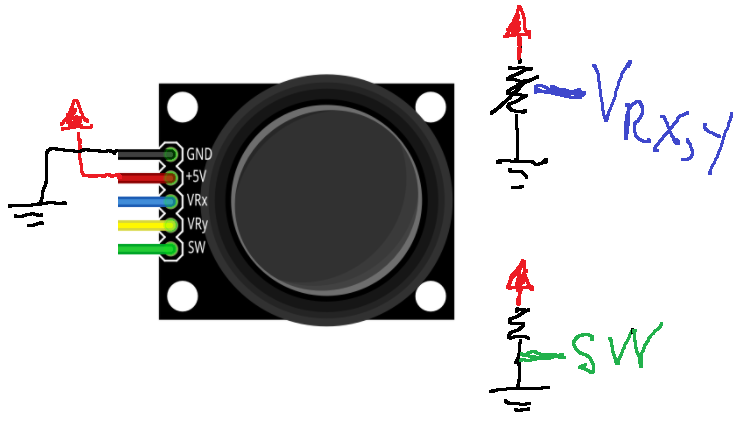


Figure 4. Joystick operation. Each axis is connected to a potentiometer generating an analog signal. The stick can also be pushed like a pushbutton. The analog signals are sent to analog inputs while the switch is connected to a digital input.

Exercises

**Downloads:** Download from Bb the stepperJoyStick.zip code. Within the folder are two stand-alone .ino files: one for the stepper and the other for the joystick. You don’t need to include any new libraries!

stepperCodeStarter.ino

joystickStarter.ino

**Highlighted portions of text described in video**

**https://streaming.uvm.edu/private/videos/N0wElyd/**

**Part I. Establish contact with stepper**

In this section you will connect to the stepper motor using the code in stepperCodeStarter.ino.

* Connect your stepper according to figure 3. Use a breadboard to connect power and ground in your construction. You will be adding parts as in figure 4 and it will make things easier if you have a separate power/ground connections.
* Upload the code to your board As written, the code should rotate the shaft of the stepper 1 full revolution CW and CCW.
* Study the code. Note:
  + A full rotation results from executing stepper(512);
  + The stepper(512) function executes a series of patterns on the output pins, IN1, IN2, IN3, IN4. What happens if you modify the values of the pins in the #define statements right at the beginning? Not working
  + Within the stepper(512) function there is a delayMicroseconds(1000); What happens when you change its value. <1000 not working, 10000 rotates slower
  + Modify the function so that it does not simply make a number of steps but runs continuously with the speed as an argument. So instead of telling the motor to run 512 steps you tell it to run with a speed between

-10<speed<10 with 0 being at rest? ?

200 rotation smaller, 600 rotation larger

* + Is there a maximum speed beyond which the motor cannot respond?

**Part II. Establish contact with the joystick**

The goal in this part is to study the function of the joystick.

* Build the circuit as in Figure 4 without disconnecting the stepper motor. Make sure to use the same analog inputs and digital input.
* Use the serial monitor to watch the value of xPosition and yPosition as you move the joystick.
* Note the function of the mapping function. Use this to control the range of the values you’re using. MapX can be made to vary between any value.
* Modify it so that it returns a -10 < MapX < 10 where MapX = 0 is not adjustment on the joystick

**Part III. Integrate the Stepper and joystick.**

Create an application that uses the joystick to control the speed and direction of the stepper motor. Your final application should have these characteristics:

* X position of the joystick controls the stepper motor.
* The stepper should be motionless when the joystick is in its home position.
* It should continuously increase stepper speed as you adjust MapX eventually reaching full speed when the stick is fully pushed to one side.
* It should do this in both directions.
* Pushing the button slows the motion by a factor of 2 but otherwise does not change how the system behaves.

This task is made more difficult because your values for MapX are supposed to be proportional to speed but the motor speed is controlled by a delay, or 1/MapX. In order to do this you might consider making a lookup table or LUT. The table might have these values. You will need to fill in the rest.

|  |  |  |
| --- | --- | --- |
| MapX | Delay |  |
| 0 | Turn off motor | Motor should be still |
| 1 | 10000 |  |
| 2 | 5000 |  |
| 3 | 3300 |  |
|  |  |  |
| 9 |  |  |
| 10 | 1000 | full speed |

In code, the table information could be put into an array.

myMapDelay[10] = {…..,2000,…..,1000}

Submit a short report demonstrating and describing only part III code.