**Hands-on Lab:**

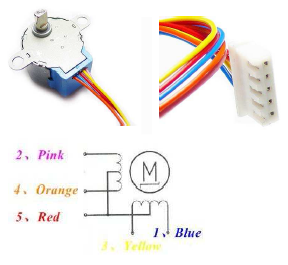
**LEGO NXT, NXC and Stepper Motors**

This lab uses NXC to actuate a stepper motor (4-phase, unipolar, 5-wire) through the LEGO NXT Brick.

**Preamble:** DC motor rotation is continuous. By contrast, steppers rotate discretely. Such stepping makes rotating a motor to fixed angle easier. Moreover, stepper rotation is practically instantaneous and accurate. This is significant because it eliminates (in a practical sense) rise-time and use of angle encoders. The net effect is that steppers are often used for precision movement applications. Examples include the positioning of 3D printer heads, CNC machine tool cutters, and robotic manipulators. Given the broad range and market for such positioning, steppers come in many different configurations to meet technical requirements such as a torque, rotational granularity, rotational speed, and size, weight, and power.

This lab introduces a commonly found 4-phase, 5-wire stepper. Specifically, the 28BYJ-48 stepper will be used. There are many vendors and online notes (e.g. using this motor with Arduino) but moreover, it often comes sold with a ULN2003 motor driver board (**Figure A left**).

**Figure A:** The 28BYJ-48 is commonly sold with a ULN2003 motor controller board with many vendors like [Amazon](https://www.amazon.com/HiLetgo-ULN2003-28BYJ-48-Stepper-4-phase/dp/B00LPK0E5A/ref=sr_1_5?crid=MBOE4C7K5CVL&keywords=28byj-48+stepper+motor+with+driver&qid=1552761082&s=gateway&sprefix=28BYJ-48+%2Caps%2C217&sr=8-5) (left). The wiring diagram from the manufacturing shows that 2 pairs of coils: Pink-and-Orange and Blue-and-Yellow, with power to the Red wire.





The [28BYJ-48 data sheet](https://www.mouser.com/ds/2/758/stepd-01-data-sheet-1143075.pdf) (**Figure A right**) shows there are 5-wires for this stepper. Moreover, there are 2 pairs of coils. One coil terminates in Orange and Pink wires, while the other coil terminates in Blue and Yellow. This coil configuration results in a 4-phase stepper; current must cycle through the 4 wires. Current is provided through the 5th wire (Red) which is connected the positive end of a voltage source (i.e. unipolar). The net effect is that the 28BYJ-48 is a *4-phase, 5-wire, unipolar* stepper motor.

**Figure B** depicts the sequence to issue current thru the coils. The key principle with a 4-phase unipolar stepper is that members of a pair (e.g. Orange/Pink or Blue/Yellow) are never energized at the same time.



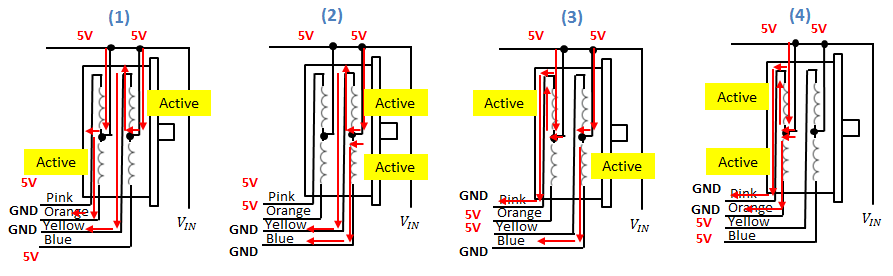
**Figure B:** Recall that members of a pair e.g. Orange/Pink or Blue/Yellow are always opposite.

For example, referring to **Figure B**, Step 1 has the Orange/Pink pair at 0 and 5 Volts respectively and the Blue/Yellow at 5 and 0 Volts respectively.

Embedded computers often have digital output lines. For example, the input port of a Lego NXT Brick can connect to a PCF8574 8-bit input/output expander via I2C communications. This provides the PCF8574 eight digital lines (D0 to D7). The stepper’s 4 wires connect to D0 to D4 as denoted in **Figure B**. The least significant bit (LSB) is D0 and the most significant bit (MSB) is D3.

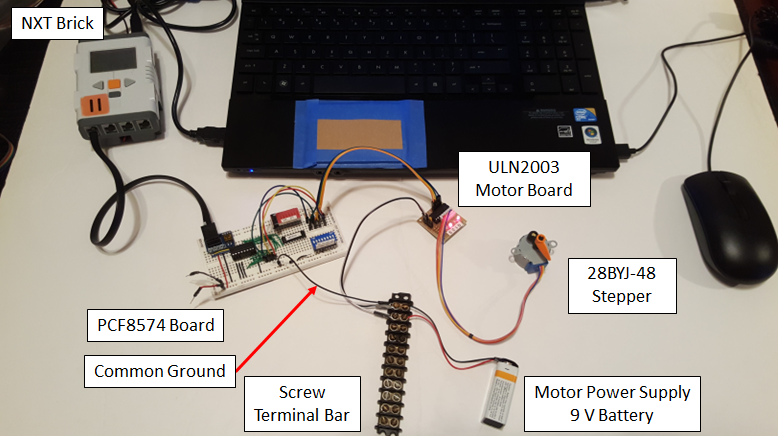
For example, in Step 1 of **Figure B**, Orange/Pink needs 0 and 5 Volts respectively and Blue/Yellow to 5 and 0 Volts respectively. Binary numbers assign 0 for 0 Volts and 1 for 5 Volts. Thus Blue-Yellow-Orange-Pink (D3-D2-D1-D0) would be 1-0-0-1 binary or 9 decimal. One would thus issue 9 decimal thru the embedded computer’s digital lines. In the case of the NXT Brick, 9 decimal would be issued via the PCF8574.

**Figure C** helps to see where the sequence in **Figure B** is derived from.

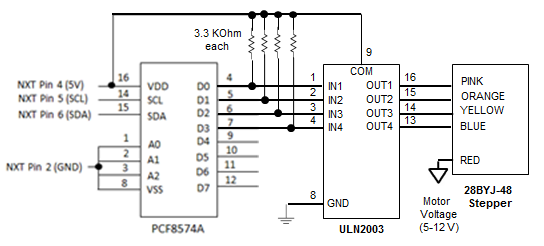


**Figure C:** The sequence of 4 steps (left to right). The red arrows show the current flow from the motor voltage supply (set to 5V to 12V), thru the 4 coils and out to the 4 colored wires. One notices that this matches the step diagram in **Figure B**. The yellow box highlights the coil that is activated.

**Concept 1:** The ULN2003 Darlington Transistor



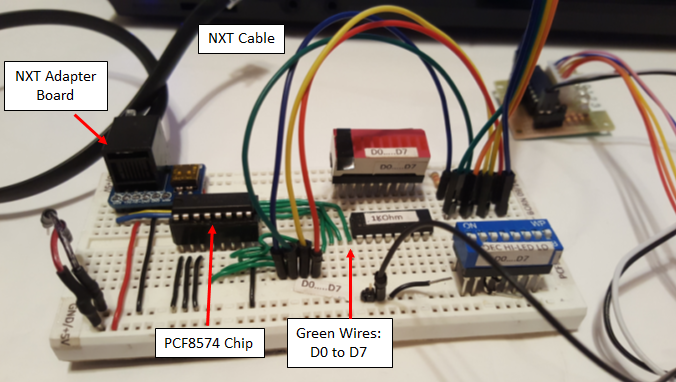
**Figure 1-0A:** Overall setup of the various components to run the 28BYJ-48 unipolar 4-phase 5-wire stepper through a PCF8574 digital input/output expander connected to Port 1 of the NXT Brick. The PCF8574 digitally commands the stepper through the ULN2003 motor board.



**Figure 1-0B:** Schematic showing PCF8574’s digital lines D0 to D3 connected into the ULN2003’s inputs IN1 to IN4. The 28BYJ-48 stepper’s 5 wires connect to the ULN2003’s outputs OUT1-OUT4 and a motor supply voltage (e.g. 9V battery).

**Step 1:** NXT Brick to PCF8574 Connections

In previous labs, the NXT Brick was connected to a PCF8574 digital input/output (DIO) expander chip. This chip provides the Brick with 8 digital lines (D0-D7) as shown in **Figure 1-0B**. The chip connects to an output port on the Brick (e.g. Port 1) via an NXT cable. **Figure 1-1** is a photo of the physical construction.



**Figure 1-1:** NXT to PCF8574 connection. There are 8 green wires for the 8 digital lines D0 to D7. For the 28BYJ-48 stepper, only 4 digital lines will be used (D0 to D3).

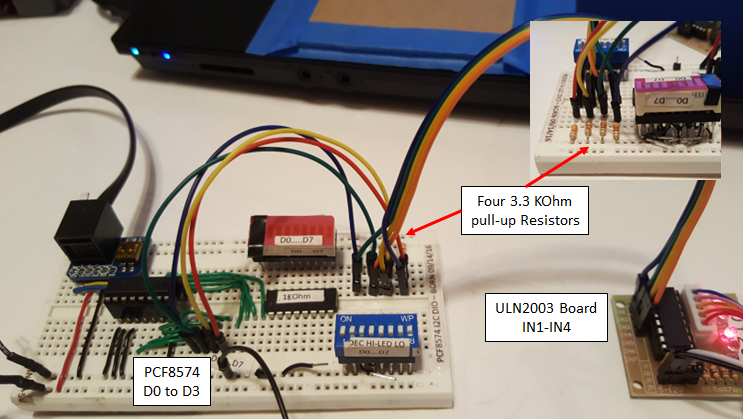
**Step 2:** PCF8574 to ULN2003 Connections

Current stemming from the digital output lines of the PCF8574 is relatively small (about 20 mA). However, steppers like the 28BJY-48 demand higher current. A simple method is to attach transistors (e.g. MOSFETs) to these digital output lines to amplify current. Since the 28BYJ-48 has 4 coils, one would need 4 such transistors. While simple, such additional hardware and wiring adds both financial and time cost to implement.

The ULN2003 is a chip that contains 7 (Darlington) transistors. As such, it serves as a single component alternative to amplify current. Implementing such a chip (rather than 4 separate MOSFETs) saves financial and time cost. **Figure 1-0B** shows the connections for this 16-pin chip. The PCF8574’s D0 to D3 lines connect into the ULN2003’s IN1 to IN4 pins.

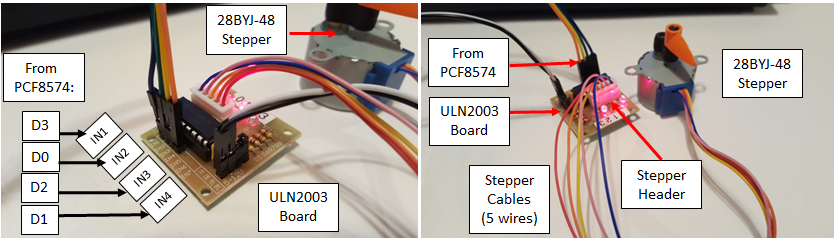
Pull-up resistors are needed. **Figure 1-0B** also shows 3.3 KOhm resistors that serve this need; when a digital line is high (i.e. 5 Volts), these pull-up resistors steer current to flow from Pin 4 and into the ULN2003’s input line (hence *pulling-up* current).

Another advantage of using the ULN2003 is that it packaged as a board and sold together with the 28BYJ-48 stepper (see **Figure A**). This makes connections even easier (Figure 2-1). The board features male header pins for IN1 to IN4 so that male-female jumper cables can be employed.



**Figure 2-1:** Four digital lines (D0 to D3) from the PCF8574 feed into the four inputs (IN1 to IN4) of the ULN2003 Board via jumper cables. Four 3.3 KOhm pull-up resistors are used. The inset photo shows them more clearly (top right corner).

**Step 3:** ULN2003 to 28BYJ-48 Stepper Connections



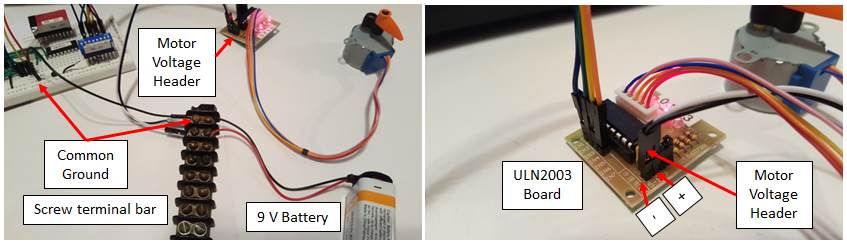
**Figure 3-1:** Four jumper wires connect the digital lines D0-D3 from the PCF8574 to the inputs (IN1-IN4) of the ULN2003 motor board (left). The 28BYJ-48 stepper has a 5-connector female header which has mates into the ULN2003 motor board. This mate is slotted (i.e. fool-proof) such that the header can be inserted in only one way.

Referring to coil sequencing diagram (**Figure B**), the digital lines D0-D4 need to be fit into the ULN2003 input lines (IN1-IN4) in a specific manner (**Figure 3-1 left**).

The 28BYJ-48 stepper’s five wires terminate in a 5-pin female header. This header can only fit into the ULN2003 motor board in a specific way (i.e. fool-proof, biased mating) as seen in **Figure 3-1 right**.

**Step 4:** Voltage Supply to 28BYJ-48 Stepper Connection

**Figure 1.0B** shows that the 28BYJ-48 stepper’s red-colored wire connects to a voltage source. This voltage source can range from 5 to 12 Volts. As such, a 9 Volt battery is used. To facilitate connections, **Figure 1-4** shows a screw terminal bar. The battery’s ends and jumper cables are securely tied down to the bar with a screwdriver.



**Figure 1-4:** A screw terminal bar (left) facilitates connections of the battery and 28BYJ-48 stepper. Note: tie the grounds of the PCF8574 circuit and the 9 Volt battery’s negative end. A 2-wire female jumper cable connects to the male 2-pin header (right), noting the polarity.

**Step 5:** NXC Programming

**Figure 1-5** shows nxcStepper1\_0e.nxc, a program to actuate the stepper according to the sequence in **Figure 1.0B**. The code for writing data through the PCF8574 was explained in a previous lab. In essence, one first defines the NXT Brick’s Port 1 for I2C communications with the PCF8574 set at address 40 Hex (64 decimal). In main, variables are declared for writing and reading data and monitoring the NXT Brick’s buttons.

// FILE: nxcStepper1\_0e.nxc - Works! Hallelujah!

// AUTH: P.Oh

// DATE: 03/15/19 19:08

// VERS: 1.0e - PCF8574A and 5-wire stepper 28YBJ-48 motor and ULN2003 driver board

// NOTE: Uses PCF8574 chip (hence address A2-A1-A0 set to 0-0-0 hence 0x40

#define I2Cport S1 // Port number

#define I2CAddr8574 0x40 // I2C address x040 8574 or 0x70 for 8574A (or 0x40 TI's PCF8574N)

task main() {

// array variables (since NXC's I2C functions take array variables

byte WriteBuf[2]; // Data written to PCF8574. Declares a two one-byte variables

byte ReadBuf[]; // Data received from PCF8574. Data won’t be read but needed for I2CBytes

int RdCnt = 1; // Number of bytes to read

// button variables

bool orangeButtonPushed, rightArrowButtonPushed, overflowFlag;

// Counting variables

int decimalNumber; // values from 0 to 255

int timeDelay;

SetSensorLowspeed (I2Cport); // PCF8574A connect to NXT on S1

// Prompt user to begin

// First, set address with first I2CWrite. Recall, WriteBuf[1] has address 0xF0 0x00

WriteBuf[1] = 0x00; // i.e. write zeros to port sets up PCF8574A for writing

WriteBuf[0] = I2CAddr8574; // i.e. address is 0x70

I2CBytes(S1, WriteBuf, RdCnt, ReadBuf);

**Figure 1-5:** NXC Program for actuating the 28BYJ-48 Stepper with the Lego NXT Brick

// Lets start with all LEDs on. This means making the port LO

WriteBuf[1] = 0x00; // Port lines are LO; LEDs should be on

WriteBuf[0] = I2CAddr8574; // i.e. address is 0x70

I2CBytes(S1, WriteBuf, RdCnt, ReadBuf);

TextOut (0, LCD\_LINE1, "Right Btn starts");

do {

rightArrowButtonPushed = ButtonPressed(BTNRIGHT, FALSE);

} while(!rightArrowButtonPushed);

TextOut(0, LCD\_LINE1, "Orange BTN quits");

decimalNumber = 0;

timeDelay = 2;

do {

orangeButtonPushed = ButtonPressed(BTNCENTER, FALSE);

// If pressed, then orange button becomes TRUE otherwise it is FALSE

// 1. Write 9

WriteBuf[1] = 9;

I2CBytes(S1, WriteBuf, RdCnt, ReadBuf);

decimalNumber = 9;

TextOut (0, LCD\_LINE3, FormatNum("Value Out: %3d" , decimalNumber));

Wait(timeDelay);

// 2. Write 3

WriteBuf[1] = 3;

I2CBytes(S1, WriteBuf, RdCnt, ReadBuf);

decimalNumber = 3;

TextOut (0, LCD\_LINE3, FormatNum("Value Out: %3d" , decimalNumber));

Wait(timeDelay);

// 3. Write 6

WriteBuf[1] = 6;

I2CBytes(S1, WriteBuf, RdCnt, ReadBuf);

decimalNumber = 6;

TextOut (0, LCD\_LINE3, FormatNum("Value Out: %3d" , decimalNumber));

Wait(timeDelay);

// 4. Write 12

WriteBuf[1] = 12;

I2CBytes(S1, WriteBuf, RdCnt, ReadBuf);

decimalNumber = 12;

TextOut (0, LCD\_LINE3, FormatNum("Value Out: %3d" , decimalNumber));

Wait(timeDelay);

// TextOut (0, LCD\_LINE3, FormatNum("Value Out: %3d" , decimalNumber));

// Wait(250); // wait 250 millsec

} while(!orangeButtonPushed && !overflowFlag);

TextOut(0, LCD\_LINE5, "Finished!");

PlaySound(SOUND\_DOUBLE\_BEEP);

} // end main

**Figure 1-5 continued:** NXC Program for actuating the 28BYJ-48 Stepper with the Lego NXT Brick

The first do-while loop waits for the user to push the NXT Brick’s right-arrow button to commence stepper actuation. The variable timeDelay sets the length of time a coil is energized. The lowest value for this motor is 2 milliseconds and thus the faster rotational speed the stepper can achieve.

The second do-while loop writes the decimal values of 9, 3, 6, and 12 as per **Figure 1.0B**. Each writing of a decimal value results in setting the digital lines D0 thru D3 with the binary equivalent. For example, 9 decimal sets digital lines D3-D2-D1-D0 to 1001 binary. This results in D3 and D0 being 5 Volts and D2 and D1 to 0 Volts. Referring to **Figure 3-1**, D3 is connect to IN1 (the stepper’s Blue wire) and D0 is connected to IN2 (the stepper’s Pink wire). Referring to Figure 1.0B, the Blue and Pink wires will receive 5 V. Likewise, the Orange and Yellow wires will be 0 Volts.

Once a decimal value is written, the program waits 2 milliseconds before writing the next decimal value. The do-while exits when the user pushes the NXT Brick’s Orange button. Upon exit, the Brick displays a finished message on the LCD, sounds a beep and terminates.

Congratulations on actuating a 28BYC-48 stepper with a LEGO NXT Brick!

**Exercise 1:** In NXC create programs for the following:

* 1. In nxcStepper1\_0e.nxc the stepper rotates in one direction (e.g. clockwise). Modify the program so the stepper rotates in the opposite direction (i.e. counter-clockwise).
  2. Allows the user to control stepper rotation by pressing the left arrow button (CW) and right arrow button (CCW)
  3. Initiate time delay to 500 milliseconds. Each moment the user pushes the NXT’s right arrow button, decrease the time delay by 100 milliseconds (and hence increase the motor’s rotational speed). Vice-versa, if the NXT’s left arrow button is pressed, then decrease the motor’s speed. NB: time delay should not go below 2 milliseconds.
  4. Rotates the stepper 90 degrees CW and stops