**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%3Dsr_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 $V\_{IN}$(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

Blah blah: show photos of the setup; need to annotate photos

Congratulations! You’ve written your first LEGO Labview Program!

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

* 1. Accepts temperatures in Celsius and displays its Fahrenheit equivalent (try pointer slides)
	2. Accepts 3 numbers and displays the average
	3. Accepts 2 numbers and displays which is the maximum

**Exercise 2:** In LabVIEW create programs for the following:

* 1. Given an input in inches, the equivalent in centimeters is displayed (by default) and the equivalent in feet is displayed (if the switch is toggled)
	2. The maximum or minimum of two numbers is displayed based on toggle position

**Concept 3:** While Loops

Loops that repeat until a condition is met are found in every programming language and LabVIEW is no exception. The While-loop repeats commands in its block control while the condition is TRUE. The While-loop control also has an iteration terminal that increases each time the block is executed.

**Step 1:** This LabVIEW VI counts the number of times the switch is toggled. Create the following front panel. Save it as nxtLabviewWhileStatement.vi.



Note: the toggle switch should have its mechanical action set to “Latch when released”.

**Step 2:** Create the following block diagram. A case structure control is inside a while-loop control. When the case is TRUE, a variable (shift register) increases, thereby displaying the number of toggles. The while-loop control stops when the user hits the Stop button.



**Step 3:** Add the False condition in the conditional terminal. Note that there are no contents (i.e. do nothing) in the case structure



**Step 4:** Run your program and toggle the switch. Also, try clicking on the Highlight Execution (light bulb icon) button from the Block Diagram menu bar and notice what happens when you execute your program.

**Exercise 3:** In LabVIEW create programs for the following:

* 1. The display increases by 2 each time the switch is toggled
	2. Increments or decrements depending on the users choice (i.e. using a 2nd toggle switch)
	3. Create a for-loop example.