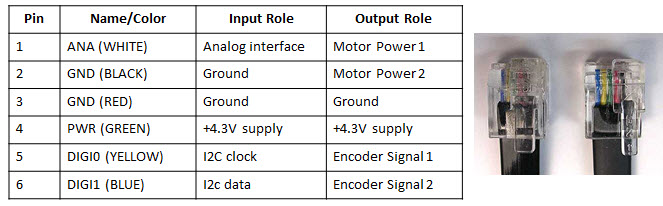
**Hands-on Lab**

**Lego Sensing – Analog-to-Digital Basics**

The Lego NXT contains a 10-bit analog-to-digital (ADC) convertor. This lab will develop sensors. This is important because sensors are a critical component for any robot.

**Concept 1 – NXT ADC:** Homemade touch sensor

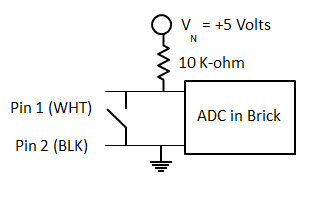
Ports 1 to 4 on an NXT Brick are connected to 10-bit ADC. First, the port’s connector uses a 6-line ribbon cable. The cable can used to connect sensors (i.e. input) or actuators (i.e. output). Since we are interested in the ADC, each wire’s role is defined in **Figure 1A**.

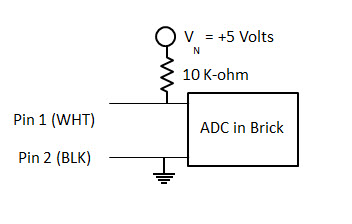


**Figure 1A:** An NXT cable has six wires with roles assigned above

Ports 1 to 4 each are connected to a 10 kilo-ohm resistor and 5 Volt supply which go into a 10-bit ADC (see **Figure 1B**).

**Figure 1B:** When Pins 1 and 2 are open, then, the ADC will read +5V (left). If the switch closes (right), then Pins 1 and 2 are shorted; the path of least resistance forces the ADC to read 0V.





**Step 1:** Create a circuit that reflects **Figure 1B** (right).

A solderless breadboard is perhaps the easiest method to construct the circuit. The switch can be simply made with some wire.

**Step 2:** Write the following NxC program and execute

// FILE: touch1\_0.nxc

// DATE: 08/18/16 01:17

// AUTH: P.Oh

// DESC: Homemade touch sensor; sensor port 1

// VERS: 1.0

task main() {

int touchSensorValue;

string strTouchSensorValue; // store integer value of touch sensor as string

string strMessageAndValue; // To display touch sensor value

SetSensorTouch(IN\_1); // homemade touch sensor on Brick Port 1

do {

touchSensorValue = Sensor(IN\_1);

strTouchSensorValue = NumToStr(touchSensorValue);

strMessageAndValue = StrCat("Touch reads:", strTouchSensorValue);

TextOut(10, LCD\_LINE4, strMessageAndValue);

Wait(100);

} while(true); // endless do-while loop

StopAllTasks();

} // end main

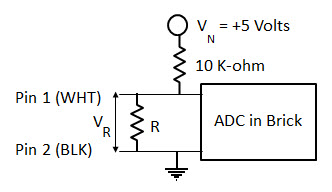
**Code Explanation:** The NxC statement SetSensorTouch(IN\_1)prepares Port 1 for inputs – by setting Pins 1 (White) and 2 (Black) for reading. The Sensor(IN\_1) statement then reads Port 1 and returns a value. This value is stored in the variable touchSensorValue. If the value is 1, in means Pins 1 and 2 are shorted (i.e. switch is closed). If the value is 0, then the two pins are not connected (i.e. switch is open).

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

* 1. Brick displays “”Touch sensor is: “, with “ON = 1” when the switch is closed and “OFF = 0” when the switch is open. If the switch is closed, then play a tone. Use statements like TextOut and PlayTone. Call this program touch1\_1.nxc.

**Concept 2 – Voltage Divider:** Homemade ohmmeter

Expanding upon **Figure 1B**, one can create insert a resistor between Pins 1 and 2. This is shown in **Figure 2A**.



**Figure 2A:** Insert a random resistor in between Pins 1 and 2.

Recall, **Figure 2A** is a voltage divider where we have the voltage across the resistor as:

(1)

**Step 1:** Build the circuit given in **Figure 2A**.

**Step 2:** Write and execute the following NxC program

// FILE: ohm1\_0.nxc

// DATE: 08/18/16 02:07

// AUTH: P.Oh

// DESC: Homemade ohm sensor; sensor port 1

// Uses Brick's Port 1's WHITE (AN) and BLACK (GND) lines

// Display value of unknown resistor connected between WHITE and BLACK lines

// Treats WHITE and BLACK lines as input into Brick's internal 10-bit ADC

// VERS: 1.0 - simple program

task main() {

int touchSensorRawValue; // a number between 0 and 1023 (10-bit ADC)

float ohmValue;

SetSensorTouch(IN\_1); // homemade touch sensor on Brick Port 1

do {

TextOut(0, LCD\_LINE1, "Raw value:");

touchSensorRawValue = SensorRaw(IN\_1); // read raw value at port

TextOut(0, LCD\_LINE2, FormatNum("%d", touchSensorRawValue));

ohmValue = ((10000)\*touchSensorRawValue) / (1023-touchSensorRawValue);

TextOut(0, LCD\_LINE3, "Ohm value is:");

TextOut(0, LCD\_LINE4, FormatNum("%3.3f", ohmValue));

Wait(100);

ClearScreen();

} while(true); // endless do-while loop

StopAllTasks();

} // end main

**Code Explanation:** To read the actual ADC value (called ), one uses the NxC statement touchSensorRawValue = SensorRaw(IN\_1). Recall that we have a 10-bit ADC, so the raw value will range from 0 to Thus, we can calculate the unknown resistor that lies between Pins 1 and 2 with the formula

(2)

So, this homemade ohmmeter can detect resistances between and 10,220,000.

**Exercise 2:**

2-1: Derive the equation (2) above and calculate the min and max resistances that can detected

2-2: Replace a fixed resistor with a potentiometer and show with a real ohmmeter, that your NxC program works

**Concept 3 – ADC Voltages:** Build a voltmeter

Recall that a 10-bit ADC results in (raw) decimal values ranging from 0 to 1023. The ADC is connected to a +5V power supply inside the NXT Brick, Thus, the (raw) decimal values corresponding to 0 and 1023 for 0V and 5V respectively. Or, a formula:

(3)

**Exercise 3:**

3-1. Write an NxC program that implements equation (3). Use the NxC statement SensorRaw(IN\_1) for your program to report raw values that digitally represent a voltage across Pins 1 and 2. Call your program volt1\_0.nxc – to represent your homemade voltmeter.

3-2: Connect a 1.5V battery or variable power supply to Port 1. The +’ve part of the battery or power supply connects to Pin 1 (AN). The –‘ve part goes into Pin 2 (i.e. GND). Run your volt1\_0.nxc so that it displays the voltage of the battery or power supply. Compare the Brick’s value with a real voltmeter.

3-3. From equation (3), what is calculated resolution (in volts) of the Brick’s 10-bit ADC?