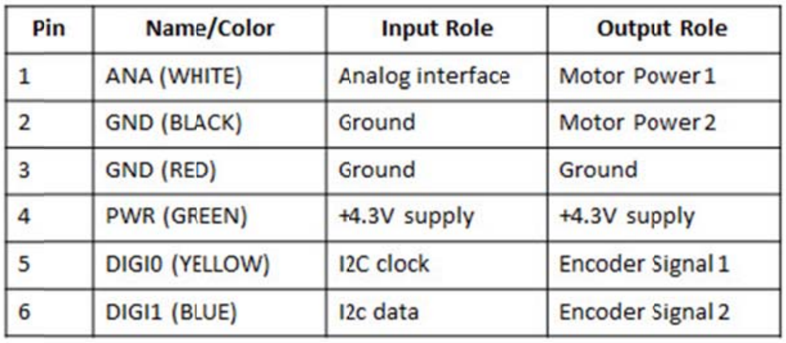
# Hands-on Lab

## Lego Touch Sensor, Voltage Divider and Voltage Supply

This lab will introduce the Lego NXT Brick's connector pinout. This enables one to build their own sensors as well as generate voltages to power actuators. This is important because it expands the capabilities of the Brick; one can go beyond Lego and 3rd party sensors and actuators to customize solutions.

## Concept 1 – NXT DIY 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**.



| Pin | Name/Color | Input Role | Output Role |
|---|---|---|---|
| 1 | ANA (WHITE) | Analog interface | Motor Power 1 |
| 2 | GND (BLACK) | Ground | Motor Power 2 |
| 3 | GND (RED) | Ground | Ground |
| 4 | PWR (GREEN) | +4.3V supply | +4.3V supply |
| 5 | DIGI0 (YELLOW) | I2C clock | Encoder Signal 1 |
| 6 | DIGI1 (BLUE) | I2c data | Encoder Signal 2 |

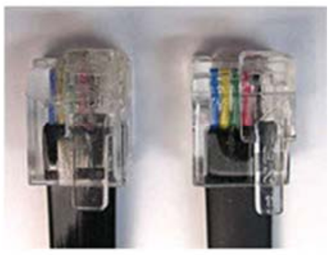


**Figure 1A:** An NXT cable has six wires with roles assigned above

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**).

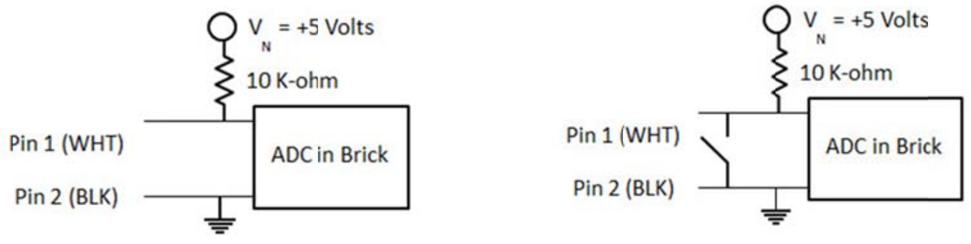


**Figure 1B:** When Pins 1 and 2 are open, then, the ADC will read +5V (left). If the switch closes (right), then Pins 1 and 2 are shorted; the path of least resistance forces the ADC to read 0V.

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

There are several 3rd party companies that sell NXT adapters (**Figure 1C**). The adapter uses a 6-pin male header that can be plugged into a solderless breadboard. Alternatively, one can simply use female-female and or male-female jumper wires.

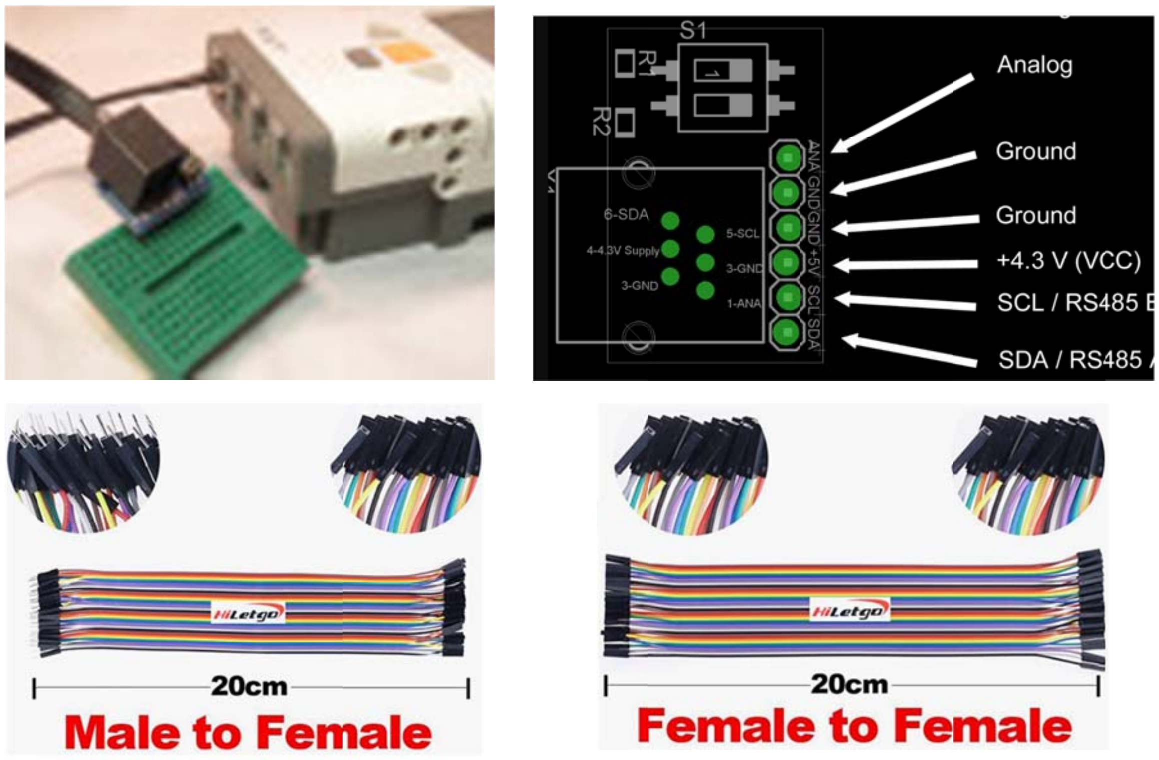


**Figure 1C**: A standard NXT cable plugs into an NXT breadboard adapter (top left) which is available from several vendors. This adapter brings the 6 wires (see Figure 1A) to pin outs (top right). One can plug male-to-female (bottom left) or female-to-female (bottom-right) jumper wires directly to the NXT adapter's male header pin.

**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:

1-1 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**.

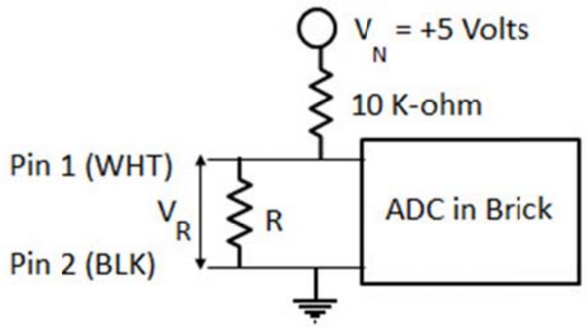


**Figure 2A:** Insert a random resistor $R$in between Pins 1 and 2.

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

$$V_R = \frac{R}{10000\,\Omega + R} V_N \tag{1}$$

**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 $raw$), 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 $2^{10} - 1 = 1023$. Thus, we can calculate the unknown resistor that lies between Pins 1 and 2 with the formula

$$R = \frac{10000}{1023 - raw} raw \tag{2}$$

So, this homemade ohmmeter can detect resistances between $\approx 9\Omega$ and 10,220,000Ω.

## 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 – NXT Voltage Source

Motors, relays, and pneumatic valves are examples of actuators. As such, actuators are critical components in any robot. Actuators need a power source to adjust motor speed, relay rates, and valve displacements. The NXT Brick's **Ports A, B, and C** can be programmed as an adjustable power source so that the actuator's state can be controlled (see **Figure 1A**)

**Step 1:** Connect an NXT motor to **Port A**. Run the following (sanity check) program

```
// FILE: voltOut1_0.nxc - WORKS!
// DATE: 08/18/16 19:33
// AUTH: P.Oh
// DESC: Port A (Motor port) WHITE (Line 1) is M1.  BLACK (Line 2) is M2
//       Can attach NXT motor (sanity check); piezo buzzer; and E10 7.5V lamp
// VERS: 1.0 - simple program: voltage output (0 to 9V)

task main() {

  // button variables
  bool orangeButtonPushed, rightArrowButtonPushed, leftArrowButtonPushed;
  bool greyButtonPushed;

  int powerLevel; // 0 to 100 will be sent to Port A for corresponding 0V to 9V

  // Prompt user to begin
  TextOut (0, LCD_LINE1, "Orange Btn starts");
  do {
     orangeButtonPushed = ButtonPressed(BTNCENTER, FALSE);
  } while(!orangeButtonPushed);

  ClearScreen();
  TextOut(0, LCD_LINE1, "Grey Btn Stops");
  TextOut(0, LCD_LINE3, "<- keys ->");
  powerLevel = 50; // middle of range... this is about 3.9V
  // 10 = about 0.6V; 70 = about 5V; 90 = about 7V

  do {
     leftArrowButtonPushed = ButtonPressed(BTNLEFT, FALSE);
     rightArrowButtonPushed = ButtonPressed(BTNRIGHT, FALSE);
     greyButtonPushed = ButtonPressed(BTNEXIT, FALSE);

     if(leftArrowButtonPushed) powerLevel = powerLevel - 10;
     if(rightArrowButtonPushed) powerLevel = powerLevel + 10;

     if(powerLevel <= 10) powerLevel = 10; // set saturation minimum
     if(powerLevel >= 90) powerLevel = 90; // set saturation maximum

     TextOut(0, LCD_LINE4, FormatNum("Power = %3d", powerLevel));
     OnFwd(OUT_A, powerLevel);
     Wait(250); // need some delay so that buttons can be read properly

   } while(!greyButtonPushed);  // end do-while

  StopAllTasks();

} // end main
```

**Code Explanation:** The Brick's buttons are used to adjust the amount of power (i.e. voltage) being sourced out of Port A. Each time the left arrow button is pushed, the power level decrements by 10 and vice-versa for the right arrow button. This results in controlling the NXT motor's rotational speed.

The power supply comes from the batteries inside the Brick. Thus, the amount of voltage the port can provide is determined by the voltage-levels of the Brick's batteries. The maximum voltage would be 9 Volts (six 1.5 Volt AA batteries). But often, the batteries (especially rechargeable ones) will have lower voltages.

The NXT Brick uses *pulse-width modulation* (PWM) to adjust the voltage coming out of Ports A, B, and C. Brick specs say that the PWM cycle is 128 microseconds (or 7800 Hz). Port A can source 800 mA while Ports B and C can source 500 mA. PWM is the ratio of the times when a signal is on and off. This results in efficiency. Think about a bicyclist. The bike's speed is a ratio of how the bicyclist pedals verses coasts. A bicyclist pedaling 100% of the time will likely get tired quickly. By contrast, pedaling 0% of the time, the bike won't move.

**Exercise 4:**

4-1: Replace the motor with a voltmeter. Connect the voltmeter's positive cable to Pin 1 and negative cable to Pin 2 of Port A. Rerun your `voltOut1_0.nxc` program. What power level corresponds to +5V? What are the voltages at power levels 10 and 90?

4-2: Replace the voltmeter with a DC motor, 7.2V lamp and/or a 3-28V piezo buzzer. What happens when you run `voltOut1_0.nxc`?

4-3: Replace the voltmeter with an oscilloscope and run voltOut1_0.nxc. What happens to the wave when you increment/decrement the power level? Sketch the wave forms. What is the frequency of the wave? What are the voltage levels of the waves? What is the ratio of the on to off times when the power level is 10, 50, 90?