Hands-on Lab

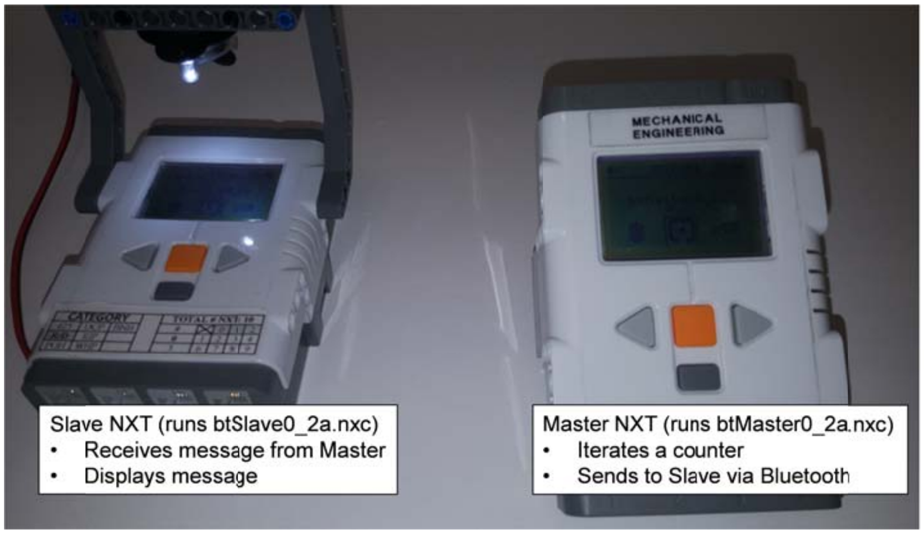
XL-320 NXC Programming – Bluetooth

NXC programs are introduced to have two NXT Bricks communicate via Bluetooth. This is useful because it enables distributed computing. For example, a motor could be attached to a ~~Slave~~ Follower Brick that would receive messages. A sensor might be attached to a ~~Master~~ Leader Brick that would then send messages containing desired motor speeds. Distributing task amongst multiple Bricks relieves computational expense.

Preliminary: Enable Bluetooth on the NXT Brick

A YouTube search yields many videos on establishing NXT Brick Bluetooth connections. One example is https://youtu.be/CN3iXGsK9YM. To customize the Brick's name, use BrixCC's `Tools – Diagnostics` and in the pop-up box, edits the `Name` field.

Concept 1: Master sending Bluetooth Messages to Slave



Slave NXT (runs btSlave0_2a.nxc)
- Receives message from Master
- Displays message

Master NXT (runs btMaster0_2a.nxc)
- Iterates a counter
- Sends to Slave via Bluetooth

Concept: Two NXT Bricks; left is ~~Slave~~ Follower and right is ~~Master~~ Leader. YouTube demonstration: https://youtu.be/s9aWIpGIYZk

Step 1: Write, compile and download the ~~Slave~~ Follower NXC program **`btSlave0_2a.nxc`**

Figure 1A shows the full NXC program to be run on the ~~Slave~~ Follower NXT. The header file **`protocol0_2a.h`** was authored by Daniele Benedettelli, a famed Lego developer and author. This H-file has functions to send or receive messages between the ~~Master~~ Leader and the ~~Slave~~ Follower NXTs; these use NXC's Bluetooth functions like `BluetoothWrite` and `ReceiveMessage`. Also, the H-file has error checking and wait-states and employs NXC's functions like `btchannelcheck`, `btwaitfor`, and `BluetoothStatus`. The goal of this concept is simply to pass messages from the ~~Master~~ Leader to the ~~Slave~~ Follower. So, an in-depth discussion of the H-file will not be explored here.

The program begins by a call to `slavecheck()` to check on the Bluetooth connection. By design, the H-file defines the ~~Slave~~ Follower and ~~Master~~ Leader channels are 1 and 0 respectively and the mailbox for Bluetooth messages is set to 0.

Next an endless `for` loop is entered. Here, `receivefrommaster` is called. Any data in the mailbox is then stored in the string variable `stringFromMaster`. The number of characters in that string is also stored in variable `j`.

```
// FILE: btSlave0_2a.nxc - Works!
// DATE: 02/24/20 14:47
// AUTH: P.Oh
// DESC: Read message from Master and display it
//       Message contains a number (as string).  Perform math on that number
// REFS: Works with btMaster0_1a.nxc

#include "protocol0_2a.h"

task main() {

  string stringFromMaster; // store string from Master
  int j;                   // store length value of received string
  int intR, mathResult;    // int form of string and math performed on that number

  slavecheck();  // initialize NXT running this program as the Slave
  TextOut(0, LCD_LINE1, "Slave" );

  for(;;) {
    stringFromMaster = receivefrommaster();
    j = StrLen(stringFromMaster);

    // -- print to screen only if there is a message
    if(j!=0) {
      TextOut(0, LCD_LINE3, stringFromMaster);
    };

    intR = StrToNum(stringFromMaster); // Master's message contains a number, so convert it
    mathResult = 10*intR;              // Perform simple math to prove it's a number
    // TextOut(0, LCD_LINE4, FormatNum("math = %5d" , mathResult));
    NumOut(0, LCD_LINE4, mathResult);

    Wait(500); // min is 10 msec, but 500 msec makes easier to see on Brick
    ResetSleepTimer(); // don't time out and shut off Brick
  } // end for
} // end main
```

Figure 1A: Listing of **`btSlave0_2a.nxc`**

As will be shown in Step 2, the ~~Master~~ Leader will send messages containing *numerical characters*. One observes the line `intR = StrToNum(stringFromMaster).` The purpose is to convert the received string to *numerical values*. The ~~Slave~~ Follower Brick will display the product of the number and 10. Before looping back, the program calls the `Wait` function. The value of 500 milliseconds helps to see what is displayed on the Brick before the next iteration.

Step 2: Write, compile and download the ~~Master~~ Leader NXC program **`btMaster0_2a.nxc`**

Similar to the ~~Slave~~ Follower program, **Figure 1B** shows the NXC code for the ~~Master~~ Leader. After checking the Bluetooth connection with a call to `mastercheck`, an endless `for` loop is entered.

```
for(;;) {
    stringFromSlave = receivefromslave(); // read message (if any) from slave
    i++; // i will be the number Master wishes to send
    strI = NumToStr(i); // must convert numbers into string

    NumOut(0, LCD_LINE2, i); // Row 2 displays actual number
    TextOut(0, LCD_LINE3, strI); // Row 3 displays string version of number
    sendtoslave(strI); // Master sends string to Slave

    Wait(500); // min is 10 msec.  But wish to view the string on Brick
    ResetSleepTimer(); // keep Brick from sleeping and turning off Bluetooth connection

  } // end for
```

In this loop, a counter called `i` is incremented and then converted to a string. This string is displayed on the ~~Master~~ Leader Brick and then `sendtoslave(strI)` sends this string via Bluetooth, to the ~~Slave~~ Follower Brick.

```
// FILE: btMaster0_2a.nxc - Works!
// DATE: 02/24/20 14:01
// AUTH: P.Oh
// DESC: Master sends message to Slave; message displayed on Slave
// VERS: Clean up btMaster0_1a.nxc
// REFS: Works with btSlave0_2a.nxc

#include "protocol0_2a.h"
#define NAP 10  // milliseconds

task main() {

  string stringFromSlave;  // any messages from slave
  int i;                  // index
  string strI;            // string version of index

  TextOut(0, LCD_LINE1, "Master" );
  mastercheck(); // check Master bluetooth connection

  for(;;) {
    stringFromSlave = receivefromslave(); // read message (if any) from slave
    i++; // i will be the number Master wishes to send
    strI = NumToStr(i); // must convert numbers into string

    NumOut(0, LCD_LINE2, i); // Row 2 displays actual number
    TextOut(0, LCD_LINE3, strI); // Row 3 displays string version of number
    sendtoslave(strI); // Master sends string to Slave

    Wait(500); // min is 10 msec.  But wish to view the string on Brick
    ResetSleepTimer(); // keep Brick from sleeping and turning off Bluetooth connection

  } // end for
} // end main
```

Figure 1B continued: Listing for **`btMaster0_2a.nxc`**

Congratulations! Your ~~Master~~ Leader NXT Brick can send strings via Bluetooth to a ~~Slave~~ Follower NXC Brick.

Exercises

1-1. Write NXC programs to detect a Master's button push states as follows. Pushing the Master's left or right arrow buttons sends via Bluetooth, a 1 or 2 respectively. The Slave receives these numbers and displays on its LCD screen the messages "Left" or "Right" respectively.