CMPT 433 - How To Guide
Team FID
Ian Fong
Fred Shih
David Baik

How to Use the BeagleBone Black to Control the NXT Brick Through Bluetooth

Why Write your Own Library

Unfortunately, there simply isn't a high quality bluetooth NXT library available in C. The more libraries are implemented in python (Python-NXT) and .NET (MindSqualls). The closest third party library available is for C++. This is due to several reasons, the first is that it's tough to write a cross platform library because of the way the packets are designed, the second is that it is a tedious and frustrating experience to do so. However, even though the networking is difficult, the NXT platform does make hardware design and implementation much simpler, and even though it is difficult, it is not impossible to get communication working reliably. This guide focuses on showing you the general steps to implement commands from LEGO's bluetooth development manual.

Connecting the NXT Brick to BBB

Pairing the Devices via Command Line

1. Turn NXT Brick's Bluetooth on
    a. Navigate to bluetooth > set on/off > on

2. Pair up the 2 devices through bluetooth
    a. On BBB, install the Bluetooth libraries:
    ```
    # sudo apt-get install bluetooth
    # sudo apt-get install bluez
    ```
    b. Search up the NXT's MAC address:
    ```
    # hcitool scan
    Scanning ...
        00:16:53:07:36:B4    NXT
    ```
    note: your MAC address will be different for your NXT Brick

    c. Pair up the two devices:
    ```
    # bluez-simple-agent hci0 <NXT's MAC address>
    ```

The console will prompt for a PIN
Default pin on the NXT is 1234
To reset default pin on NXT:
Bluetooth > Use Default Pswd > yes

d. Register the NXT as a trusted device:

```
# bluez-test-device trusted <NXT's MAC address> yes
```

To test if the NXT is trusted, run:

```
#bluez-test-device trusted <NXT's MAC address>
```

Returns 1 if the device is trusted

e. Connect the devices:

```
# bluez-test-input connect <NXT's MAC address>
```

The NXT should now be paired up with the BBB

Connecting to the Brick Through C

1. Required libraries:

```
#include <bluetooth/bluetooth.h>
#include <bluetooth/rfcomm.h>
#include <sys/socket.h>
```

2. Pass in your MAC address as a string, then convert the address to a Bluetooth address using str2ba()
3. Create an address of struct sockaddr_rc
4. Initialize a socket using the socket() function

```
socket(AF_BLUETOOTH, SOCK_STREAM, BTPROTO_RFCOMM);
```

5. Set the socket type to bluetooth, and select a channel:

```
<address struct>.rc_family = AF_BLUETOOTH;
<address struct>.rc_channel = (uint8_t) 1;
```

6. Connect using the address and socket

```
connect(socket, (<struct sockaddr *)&
<address struct>, sizeof(<address struct>));
```

A successful Bluetooth connection to the NXT will show a “<” next to the Bluetooth symbol

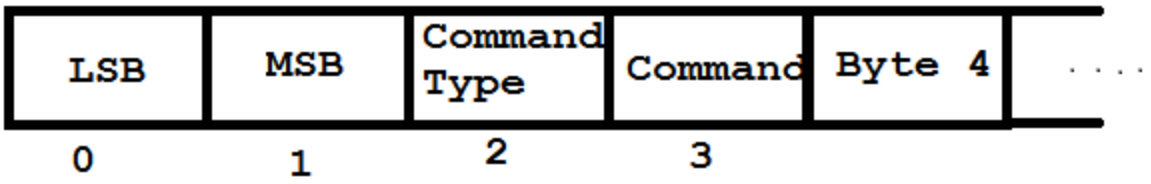
Demo of Initializing and Connecting via Socket:

```
void init_bluetooth(const char *bt_addr, int *socket) {
      struct sockaddr_rc addr = {0};
      int status;

      *socket = socket(AF_BLUETOOTH, SOCK_STREAM,
            BTPROTO_RFCOMM);

      addr.rc_family = AF_BLUETOOTH;
      addr.rc_channel = (uint8_t) 1;
      str2ba(bt_addr, &addr.rc_bdaddr);
      status = connect(*nxt_socket, (struct sockaddr *)&addr,
            sizeof(addr));
      if(status < 0) {
            //error connecting to Bluetooth
      }
}
```

Communicating with the NXT

The NXT Brick Command and Response packet formats can be obtained by reading Appendix 2 of LEGO's NXT Bluetooth Developer manual. The fields values are usually given as **hexadecimal, <u>not</u> decimal!** The basic layout of packets are as follow, each block represents one byte:

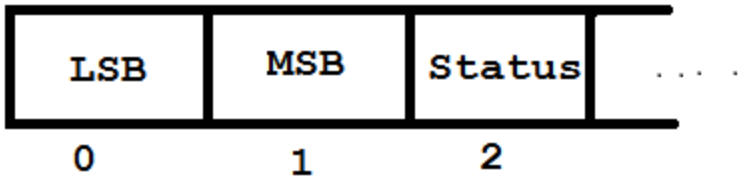
a. Command packets:



Command type: 0x00 to have the NXT return a response
0x80 to have the NXT not return a response

The LSB and MSB compose the length of the packet, excluding the LSB and fields. Because no command is ever larger than 255 bytes, the MSB field is **always zero**. Command specific bytes always come after the command field.

b. Response packets:



Both Command and Response 's first two bytes

(LSB and MSB) indicate the length of the package

Like the command packets, the response packets also have the LSB and MSB fields which specifies the byte size of the response packet (excluding the LSB and MSB fields themselves). There is also a status byte which follows the MSB byte. A value of 0x00 denotes success, and anything else is a mapped error code. **Note:** you will only get a response if the command type is set to 0x00.

c. **Note**: NXT commands are little endian

2. Commands are sent to the NXT through the socket via the write() function. Note that the length from the lsb/msb fields is different from the size of the packet. The size parameter for the write command is the size of the struct or array you chose to represent the command.

```
write(socket, command, size of packet)
```

3. Responses can be read through the socket using the read() function. Note that this is a blocking wait, which means that if you forget to ask for a response in your initial command packet, the program will end up hanging at read.

```
read(socket, response, size of response)
```

Demo of Sending a Command to Motor A via Socket:

```
//struct to a Motor Command
//note the packed attribute
typedef __attribute__((__packed__)) motor_cmd_t {
     uint8_t lsb;
     uint8_t msb;
     uint8_t cmd_type;
     uint8_t cmd;
     uint8_t motor;
     uint8_t speed;
     uint8_t mode;
     uint8_t regulation;
     uint8_t ratio;
     uint8_t state;
     union {
          uint8_t tacho_limit[4];
          uint32_t tacho;
     };
} SetMotor;
```

```
void run_motor_A(const int *nxt_socket) {
      //initialize the Motor Command
      SetMotor cmd = {
            0x0C,           //LSB
            0x00,           //MSB
            0x80,           //Command Type
            0x04,           //Command
            0x00,           //Motor A
            0x64,           //Speed 100
            0x01,           //Mode
            0x00,           //Regulation off
            0x00,           //Turn Ratio
            0x20,           //Run State
            {{ 0,0,0,0 }}   //Tacho Limit
      };

      //send command to the NXT Brick
      if (write(socket, cmd, sizeOf(cmd)) < 0) {
            perror("error sending command");
            abort();
      }
}

int main() {
      int socket = 0;
      init_bluetooth("00:16:53:07:36:B4", &socket);

      run_motor_A(socket);

      //Close Bluetooth connection
      close(socket);
}
```

Note: If a struct is used over a byte array, you *must* tell the compiler to pack the fields. On GCC and Clang, this is accomplished by adding the packed attribute (as shown above). If you do not do this, the compiler will most likely add padding, which will give you result in garbage transmissions between BBB and NXT.

Quick Reference Guide to NXT Packets

Because LEGO frequently changes the URL for the bluetooth development sdk, we have embedded a table of useful packets in this guide.

a. Motor Commands

Byte	**Type**	**Description**
0	LSB	Least significant byte 0x0C set command 0x03 read command
1	MSB	0x00 Most significant byte
2	Command Type	0x00 with response (useful when reading motor values) 0x80 no response
3	Command	0x04 to set command to motor 0x06 to read values from motor
4	Motor	Specify motor of use 0x00 Motor A 0x01 Motor B 0x02 Motor C 0xFF All motors
5	Speed/Power	Desired speed to set to motor Between -0x64 (-100) to 0x64 (100)
6	Mode	0x01 to turn on motor 0x02 to turn use brakes when stopping motor 0x04 to turn on regulation

7	Regulation	0x00 no regulation 0x01 to run motor at the specified Speed 0x02 to set 2 or more motors to a synchronized speed
8	Turn Ratio	Between -0x64 (-100) to 0x64 (100)
9	Run State	0x00 turn off motor 0x20 turn on motor
10 - 14	Tacho Limit	the amount of degrees to rotate the motors

b. Motor Response

Byte	**Type**	**Description**
0	LSB	0x19 Least significant byte
1	MSB	0x00 Most significant byte
2	Response Byte	0x02
3	Response Byte	0x06
4	Status	Status of the motor
5	Motor	Motor of the returned values
6	Speed/Power	Current speed of motor
7	Mode	Current mode
8	Regulation	Current regulation
9	Turn Ratio	Current turn ratio
10	Run State	Current Runstate
11 - 14	Tacho Limit	Current Tacho Limit
15 - 18	Tacho Count	Distance the motor has rotated from last reset position
19 - 22	Block Tacho Count	Position of the motor from most recent command
23 - 26	Rotation Count	Position of the motor from last reset position

c. Motor Reset Command

Byte	Type	Description
0	LSB	0x04 Least significant byte
1	MSB	0x00 Most significant byte
2	Command Type	0x00 with response 0x80 no response
3	Command	0x0A
4	Motor	Specify motor of use 0x00 Motor A 0x01 Motor B 0x02 Motor C
5	Relativity	TRUE to set position relative to last movement FALSE to set to absolute position

8. Sensor Commands and Responses
    a. Sensor Command

Byte	**Type**	**Description**
0	LSB	Least significant byte 0x05 to send command to sensor 0x03to read sensor
1	MSB	0x00 Most significant byte
2	Command Type	0x05 to set command to sensor 0x07 to read values from sensor
3	Command	0x00 with response (useful when reading motor values) 0x80 no response
4	Port	Specify sensor port 0x00 Port 1 0x01 Port 2 0x02 Port 3 0x03 Port 4
5	Sensor Type	values are from LEGO MINDSTORMS NXT documentations in the references 0x0B Low speed 9V
6	Sensor Mode	values are from LEGO MINDSTORMS NXT documentations in the references 0x00 Raw Mode

b. Sensor Response

Byte	**Type**	**Description**
0	LSB	0x0F Least significant byte
1	MSB	0x00 Most significant byte
2	Response Byte	0x02
3	Response Byte	0x07
4	Status	Status of sensor
5	Port	Port of the sensor
6	Validity	TRUE if data is valid
7	Calibration	TRUE if data has been calibrated
8	Sensor Type	Current Sensor type
9	Sensor Mode	Sensor Mode
10 - 11	Raw value	Raw value
12 - 13	Normalized Value	Normalized value
14 - 15	Scaled Value	Scaled value
16 - 17	Calibrated Value	Calibrated value

Cited Guides and Documentation

1. Lego mindstorm bluetooth reference manual v1.00
2. Gentoo Wiki - Device Pairing: http://wiki.gentoo.org/wiki/Bluetooth#Device_pairing
3. NXT LCD Image: http://nnxt.blogspot.ca/2012/10/motorcontrol-nxt.html