Using Bluetooth to Connect the Beaglebone Black to the NXT Brick & Sending Simple Motor Command

By Aman, Cathy, Gener, Jaspal

Commands prefaced with $ are run on the host
Commands prefaced with # are run on the target

Use Bluetooth to Connect BBB to NXT

1. Turn on bluetooth on NXT

2. Install necessary bluetooth packages on host

```
$ sudo apt-get install bluetooth
$ sudo apt-get install bluez
$ sudo apt-get install libbluetooth-dev
```

3. Install necessary bluetooth packages on target'

```
# sudo apt-get install bluetooth
# sudo apt-get install bluez
# sudo apt-get install libbluetooth-dev:armel
```

4. Scan for the NXT

```
# hcitool scan
Scanning …
      00:16:53:09:D3:3C  n/a
```

   The NXT will show up as "n/a" or as "NXT". The address will be different for you. Note the mac address for later use.

5. On the initial connect, confirm the passcode on the NXT
   You are now connected!
   Refer to troubleshooting section if you encountered any problems.

Cross Compiling for BBB

Follow the guide found at the following link http://wiki.beyondlogic.org/index.php?title=Cross_Compiling_BlueZ_Bluetooth_tools_for_ARM. You will need to complete the prerequisite and installing sections. Make sure to install the latest versions as the guide is outdated. This can easily be done with a little research on google.

To run your executive, “-lbluetooth” must be added to your gcc command.

Troubleshooting

Unable to install bluez

Edit the file

```
# nano /etc/init.d/led_aging.sh
```

and replace the contents with

```
#!/bin/sh -e
### BEGIN INIT INFO
# Provides:        led_aging.sh
# Required-Start:  $local_fs
# Required-Stop:   $local_fs
# Default-Start:   2 3 4 5
# Default-Stop:    0 1 6
# Short-Description: Start LED aging
# Description:     Starts LED aging (whatever that is)
### END INIT INFO

x=$(/bin/ps -ef | /bin/grep "[l]ed_acc")
if [ ! -n "$x" -a -x /usr/bin/led_acc ]; then
      /usr/bin/led_acc &
fi
```

Then run

```
# apt-get upgrade
```

Bluetooth Connection via C Code

```
#include <stdio.h>  //These must be added to the include section to work
#include <unistd.h>
#include <sys/socket.h>
#include <bluetooth/bluetooth.h>
#include <bluetooth/rfcomm.h>

int main(int argc, char **argv)
{
      struct sockaddr_rc addr = { 0 };
      int status;
      const char dest[18] = "00:16:53:09:D3:3C";

      // allocate a socket
      s = socket(AF_BLUETOOTH, SOCK_STREAM, BTPROTO_RFCOMM);

      // set the connection parameters (who to connect to)
      addr.rc_family = AF_BLUETOOTH;
      addr.rc_channel = (uint8_t) 1;
      str2ba( dest, &addr.rc_bdaddr );

      // connect to server
      status = connect(s, (struct sockaddr *)&addr,    sizeof(addr));

      // send a message
      printf("status: %d\n", status);

      // send a message
      if( status == 0 ) {
         // connected successfully
         // do main work
      }

      if( status < 0 ) perror("Error");

      close(s);
      return 0;
}
```

*Note: You will still need to run the following command to start the bluetooth-agent

```
# sudo bluetooth-agent <passcode> &
```

Sending Simple Motor Commands

Telegrams are used to send and receive information via bluetooth. They follow a format, which if done incorrectly, could cause errors and send back garbage values. More information on these command formats can be found in the LEGO Mindstorms NXT Direct Commands manual:

http://joanna.iwr.uni-heidelberg.de/projects/NXT_DAME/data/nxt_direct_command.pdf

Writing to Motor

Byte 0: LSB
Byte 1: MSB
Byte 2: Response = 0x00, No Response = 0x80
Byte 3: Command, Writing = 0x04
Byte 4: Port Number, from 0x00 - 0x02, 0xFF for ALL
Byte 5: Motor Power, from 0x9C - 0x64 (-100 to 100)
Byte 6: Motor Mode, motor on = 0x01, brake = 0x02, regulated = 0x04
*Can have any combination (ex. Motor + Reg = 0x05)
Byte 7: Regulation Mode, idle = 0x00, motor speed = 0x01, motor sync 0x02
Byte 8: Turn Ratio, 0x9C - 0x64 (-100 to 100)
*Requires regulation mode to be set to motor sync
Byte 9: Run State, idle = 0x00, ramp up = 0x10, running = 0x20, ramp down = 0x40
Byte 10 - 14: Tachometer, 0x0000 0000 - 0xFFFF FFFF

If specifying a response in Byte 2, the NXT will return a response telegram with the values below:

Return Package for Writing with Response:
Byte 0: 0x03 (LSB)
Byte 1: 0x00 (MSB)
Byte 2: 0x02 (Response)
Byte 3: 0x04 (Write Command)
Byte 4: Status Byte (Will Return 0 if Successful)

Example shown below:

```
Example:
setMotor forwardA = {
            0x0C,   //lsb = 0x0C = 12, size of command being sent
            0x00,   //not including LSB and MSB
            0x80,   //No response, we dont need the NXT to respond
            0x04,   //Write command
            0x00,   //Port 0x00 = first port
            0x64,   //0x64 = 100, sends 100% power to the motor
            0x03,   //0x03 == 0x01 + 0x02, which means motor on with brake
            0x01,   //Motor speed is regulated by NXT software
            0x00,   //No need to sync multiple motors for turning
            0x20,   //Want the motor to just run with power specified
            0x68,   //Tachometer is set to 0x0168 == 360, which is 360 ticks
            0x01,   //or one full rotation of the motor
            0x00,
            0x00
        };
```

Figure 1 - Sample Write

Reading from Motor

Byte 0: LSB
Byte 1: MSB
Byte 2: Response = 0x00, No Response = 0x80
Byte 3: Command, Reading = 0x06
Byte 4: Port Number, from 0x00 - 0x02,

Return Package for Reading with Response:
Byte 0: LSB
Byte 1: MSB
Byte 2: 0x02 (Response)
Byte 3: 0x06 (Read Command)
Byte 4: Status Byte (Will Return 0 if Successful)
Byte 5: Port number (from 0 to 2)
Byte 6: Motor Power (from -100 to 100)
Byte 7: Motor Mode (See modes above)
Byte 8: Regulation Mode (See reg modes above)
Byte 9: Turn Ratio (See turn codes above)
Byte 10: Run State (See turn codes above)
Byte 11-14: Tachometer Limit
Byte 15-18: Tachometer Count
Byte 19-22: Block Tachometer Count
Byte 22-25: Rotation Count

```
readMotor motorA = {
            0x03,   //LSB
            0x00,   //MSB
            0x00,   //We want a response so we write 0x00
            0x06,   //0x06 is the command to read
            0x00    //Port number
          };
```

Figure 1 - Sample Read

Reading from the motor is useful to ensure that the proper commands are being written.

References

http://eionix.blogspot.ca/2015/02/beagleboneblack-apt-get-upgrade-error.html
http://www.cedtnsit.in/connectx/pdf/Bluetooth_BBB.pdf
https://www.cs.sfu.ca/CourseCentral/433/bfraser/other/2014-student-howtos/LegoNXTBrick.pdf
http://www.robotappstore.com/Knowledge-Base/Introduction-To-Lego-NXT-Programming/32.html
http://joanna.iwr.uni-heidelberg.de/projects/NXT_DAME/data/nxt_direct_command.pdf