Bluetooth Adapter Guide

Connecting and transmitting data from a Bluetooth device to a BeagleBone Green.

By Megan Duclos, Laura Flood, Tanya Nookut, and Kaleigh Toering
Created December 2022

Guide Reasoning

A new Bluetooth guide is needed due to updated Bluetooth libraries and commands. This guide also includes initial Bluetooth testing and troubleshooting using Linux commands and data transmission to the BeagleBone.

This document guides the user through:

1. Using Bluetooth libraries from the Linux command line to detect and connect to Bluetooth devices.
2. C code to detect Bluetooth devices.
3. C code to read data to a BeagleBone server from a Bluetooth client.

Table of Contents

Formatting

1. Linux commands for the host start with `(host) $`.
   `(host) $ echo "Hello World from Host!"`
2. Linux commands for the target start with `(bbg) $`.
   `(bbg) $ echo "Hello World from Target!"`
3. Commands in the Bluetooth terminal include `[bluetooth]#` after host or target notation.
   `(bbg) $ [bluetooth]# power on`
4. Commands in the Bluetooth terminal for a connected device include `[device-name]#` after host or target notation.
   `(bbg) $ [device-name]# info`
5. Assume all commands are case-sensitive.

1. Bluetooth Basics

Bluetooth allows for wireless detection, connection, and communication between devices in close proximity. The device which scans and initiates connection in the Bluetooth relationship is called the client and the device which accepts the connection request is the server. In this guide, the BeagleBone Green is the client and the user's device is the server.

To facilitate Bluetooth communication, the Media Access Control (MAC) address of both the client and the server is needed. Every device has a unique MAC address which is 12 characters long with every two characters separated by usually a colon or hyphen.

```
01:23:45:67:89:YZ
01-23-45-67-89-YZ
```

Depending on the device's operating system, the MAC address may be referred to as the physical or hardware address as well.

Note: The BeagleBone Green can detect non-android devices and may be able to connect. However, an android device is recommended and used for this guide.

2. Bluetooth Connection Using Linux Command Line

This step-by-step guide lays out how to detect and connect to a Bluetooth device from the BeagleBone Green using the Linux command line. A Bluetooth USB adapter compatible with Linux is required for the BeagleBone Green.

2.1. Setting Up the IDE for Bluetooth

1. Install Bluetooth libraries on the target.
   ```
   (bbg) $ sudo apt update
   (bbg) $ sudo apt install bluetooth bluez bluez-tools rfkill
   ```
2. Check the Bluetooth adapter is connected. A device with the brand of the USB adapter should be listed.
   ```
   (bbg) $ lsusb
   ```
3. Ensure Bluetooth is not disabled.
   ```
   (bbg) $ rfkill
   ```
   a. If Bluetooth is blocked, unblock it.
      ```
      (bbg) $ rfkill unblock bluetooth
      ```
4. Start the Bluetooth terminal and power on the Bluetooth adapter.
   ```
   (bbg) $ bluetoothctl
   (bbg) $ bluetoothctl show
   (bbg) $ [bluetooth]# agent KeyboardOnly
   (bbg) $ [bluetooth]# default-agent
   (bbg) $ [bluetooth]# power on
   ```
5. Troubleshooting
   a. If you are unable to install the Bluetooth libraries, ensure the target is connected to the internet.
   b. If the USB adapter is not listed as connected to the BeagleBone Green, try unplugging and replugging it back in.

2.2. Pairing and Connecting to a Bluetooth Device[1]

1. Scan for nearby devices on the target.
   ```
   (bbg) $ [bluetooth]# scan on
   ```
2. Find the name of the desired device in the list and note its MAC address (ie. `01:23:45:67:89:YZ`).
3. Pair the target to the user's device.
   ```
   (bbg) $ [bluetooth]# pair 01:23:45:67:89:YZ
   (bbg) $ [bluetooth]# trust 01:23:45:67:89:YZ
   ```
4. Connect to the user's device.
   ```
   (bbg) $ [bluetooth]# connect 01:23:45:67:89:YZ
   ```
5. Check device info of paired device. Device should be paired, trusted, and connected.
   ```
   (bbg) $ [device-name]# info
   ```
6. When done, disconnect from the device and exit the Bluetooth terminal.
   ```
   (bbg) $ [device-name]# disconnect
   (bbg) $ [bluetooth]# exit
   ```
7. Troubleshooting

[1] Note this section may be replaced with simply connecting to the BeagleBone Green on the user's device if only confirming connection is possible is desired.

a. If your device does not show up when scanning for devices:
    i. Ensure your device's Bluetooth is on and discoverable. Most devices are automatically discoverable when Bluetooth is on.
    ii. If communicating with the BeagleBone Green using a Virtual Machine and trying to connect to your native operating system, ensure the device's Bluetooth is connected to the native operating system and not the Virtual machine.
b. If you cannot connect to the device, ensure it is actually paired.
    `(bbg) $ [bluetooth]# paired-devices`

3. Bluetooth Device Detection using C Code

To run the program simply call the initialization and cleanup function in a main function with a desired delay.
Note: The `hcitool` and its commands can be used for finding MAC addresses and general troubleshooting although is not required to complete the guide.

3.1. Initialization and Cleanup

```
// Assume device already detectable to other devices
// sudo hciconfig hci0 piscan
// Assume MAC address of client and server known
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <sys/socket.h>
#include <bluetooth/bluetooth.h>
#include <bluetooth/hci.h>
#include <bluetooth/hci_lib.h>
#include <bluetooth/rfcomm.h>
#include <pthread.h>
#include <stdbool.h>

// Initialize thread IDs
static pthread_t deviceDiscoveryId;
static pthread_mutex_t mutex;

// Initialize module variables
static bdaddr_t bluetoothAdapterID = {{0xAB, 0xCD, 0xEF, 0xGH, 0xIJ, 0xKL}};
// client MAC address
char userIDString[17] = "01:23:45:67:89:YZ";  // server MAC address
static bool userInRange = false;
static bool isDoneRunning = false;
```

```
// Initializes the Bluetooth module and starts thread. Must be run before
other functions.
void BluetoothServer_init(void)
{
    pthread_create(&deviceDiscoveryId, NULL, deviceDiscoveryThread, NULL);
    return;
}

// Shuts downBlueTooth module and closes thread
void BluetoothServer_cleanup(void)
{
    isDoneRunning = true;
    pthread_join(deviceDiscoveryId, NULL);
    return;
}
```

3.2. Range Output, Device Detection, and Thread

```
// Checks if the user's device is within range
static bool BluetoothServer_isUserInRange(void)
{
    bool status = false;
    pthread_mutex_lock(&mutex);
    status = userInRange;
    pthread_mutex_unlock(&mutex);
    return status;
}

// This function looks up the Bluetooth device name from the known device
address.
// If successful, function returns 0. If unsuccessful, function returns -1;
static int discoverDevice(void)
{
    int dev_id = 0;
    dev_id = hci_get_route(&bluetoothAdapterID);

    // Open a Bluetooth socket to connect BBG to the Bluetooth adapter
    int sock = 0;
    sock = hci_open_dev(dev_id);

    // Error checking
    if (dev_id < 0 || sock < 0)
```

```
    {
        perror("opening socket");
        exit(1);
    }

    char name[248] = {0};

    bdaddr_t userID;
    str2ba(userIDString, &userID);

    // Read name from device ID
    int rsp_code = hci_read_remote_name(sock, &userID, sizeof(name), name, 0);

    close(sock);

    return rsp_code;
}

// This thread continuously scans if the user is within range every 1 sec
static void *deviceDiscoveryThread(void *_)
{
    int scanPeriod = 1; // s
    while (!isDoneRunning)
    {
        int rsp_code = 0;
        rsp_code = discoverDevice();

        if (rsp_code < 0)
        {
            pthread_mutex_lock(&mutex);
            userInRange = false;
            pthread_mutex_unlock(&mutex);
        }
        else
        {
            pthread_mutex_lock(&mutex);
            userInRange = true;
            pthread_mutex_unlock(&mutex);
        }
        sleep(scanPeriod);
    }
    return NULL;
```

```
}
```

4. Cross-Compiling with Bluetooth Setup

This step-by-step guide lays out how to set up your environment to compile and run C code that uses Bluetooth.

1. Install Bluetooth libraries on the host.
   ```
   (host) $ sudo apt-get update
   (host) $ sudo apt-get install libbluetooth-dev
   ```
2. On the target check if the Bluetooth library is installed.
   ```
   (bbg) $ cd /usr/lib/arm-linux-gnueabihf
   (bbg) $ ls libbluetooth*
   ```
   a. If libbluetooth.so not listed, install the library.
      ```
      (bbg) $ sudo apt-get install libbluetooth-dev
      ```
3. The host needs a copy of the .so file from the target in order to cross compile. Copy the file to the shared NFS mounted folder.
   ```
   (host) $ mkdir ~/cmpt433/public/bluetooth_lib_BBG
   (host) $ chmod a+rw ~/cmpt433/public/bluetooth_lib_BBG
   (bbg) $ cd /usr/lib/arm-linux-gnueabihf
   (bbg) $ cp libbluetooth.so /mnt/remote/bluetooth_lib_BBG
   ```
4. Add a flag for the Bluetooth to your Makefile.
   ```
   LFLAGS = -L$(HOME)/cmpt433/public/bluetooth_lib_BBG
   ```
5. Add the flag and linker to your existing Makefile line with your other flags.
   ```
   $(LFLAGS) -lbluetooth
   ```

5. Bluetooth Server Connection and Data Transmission using C Code

To run the program simply call the overall initialization and cleanup function in a main function with a desired delay.
Note: Overall initialization, cleanup, and device detection functions the same as Section 3.1.
Note: The client can be built in any desired language, for an example in C code refer to the BeagleBone Textbook listed in References (Section 5).

5.1. Transmission Communication Initialization and Cleanup

```
// Initialize data transmission variables
static bool commInProgress = false;
static struct sockaddr_rc adapter_addr = {0}, user_addr = {0};
static int bluetoothSocket = 0;
static int client = 0;

// Connects BlueTooth to user's device
static void bluetoothCommInit(void)
{
    // Allocate socket for Bluetooth communication with RFCOMM protocol
```

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

    // Initialise RFCOMM socket for the Bluetooth adapter a
    adapter_addr.rc_family = AF_BLUETOOTH;
    // Since there is only one Bluetooth adapter, any Bluetooth adapter
address works
    adapter_addr.rc_bdaddr = *BDADDR_ANY;
    adapter_addr.rc_channel = (uint8_t)1;

    // Bind Bluetooth adapter socket to port 1 (allocate port 1 to the
bluetoothSocket)
    bind(bluetoothSocket, (struct sockaddr *)&adapter_addr,
sizeof(adapter_addr));

    // Put bluetoothSocket into listening mode, allow only 1 connection
    listen(bluetoothSocket, 1);
    printf("Listen for a connection.....................................\n");

    // Accept one client connection
    socklen_t usaddr_len = sizeof(user_addr);
    // Blocking call that waits for incoming connection request
    client = accept(bluetoothSocket, (struct sockaddr *)&user_addr,
&usaddr_len);

    char buffer[1024] = {0};
    ba2str(&user_addr.rc_bdaddr, buffer);
    fprintf(stderr, "Accepted connection from %s\n", buffer);
    memset(buffer, 0, sizeof(buffer));

    return;
}

// Closes the BlueTooth socket
static void bluetoothCommCleanup(void)
{
    // Close connection
    close(client);
    close(bluetoothSocket);
}
```

5.2. Read Data over Bluetooth

```
// This function continuously read data
static void bluetoothCommRead(void)
{
    int numBytesRead = 0;
    char buffer[1024] = {0};

    // Read data into a buffer
    numBytesRead = read(client, buffer, sizeof(buffer));

    // Parse data from the buffer
    if (numBytesRead > 0)
    {
        // Display transmitted data
        printf("User's device says %s.\n", buffer);
    }
}
```

5.3. Updated Bluetooth Thread

```
// This thread continuously scans if the user is within range every 1 sec
static void *deviceDiscoveryThread(void *_)
{
    int scanPeriod = 1; // s
    while (!isDoneRunning)
    {
        int rsp_code = 0;
        rsp_code = discoverDevice();

        if (rsp_code < 0)
        {
            pthread_mutex_lock(&mutex);
            userInRange = false;
            pthread_mutex_unlock(&mutex);

            // If a bluetoothCommThread exists, destroy thread
            if (commInProgress)
            {
                commInProgress = false;
                bluetoothCommCleanup();
            }
```

```
        }
        else
        {
            pthread_mutex_lock(&mutex);
            userInRange = true;
            pthread_mutex_unlock(&mutex);

            // If communication is not in progress, initialise new socket
connection
            if (!commInProgress)
            {
                bluetoothCommInit();
            }
            if (client > 0)
            {
                commInProgress = true;
                bluetoothCommRead();
            }
        }
        sleep(scanPeriod);
    }
    return NULL;
}
```

6. References

Old Bluetooth Guide from CMPT433 Fall 2015
https://opencoursehub.cs.sfu.ca/bfraser/grav-cms/cmpt433/links/files/2015-student-howtos/BBBluetoothGuide.pdf

How To Connect To Bluetooth Device from Linux Terminal by Computing for Geeks
https://computingforgeeks.com/connect-to-bluetooth-device-from-linux-terminal/

Exploring BeagleBone 2nd Ed, by Derek Molloy, 2019.

Bluetooth Essentials for Programmers 1st Edition by Albert S. Huang