Getting Started with the HM-10 BLE Module

Working with the HM-10 BLE Module using UART

Contents

List of Figures

1 HM-10 BLE Module Serial Communication

Figure 1: HM-10 Module from DSD Tech

The HM-10 Module is a Bluetooth 4.0 Module that includes Bluetooth Low Energy (BLE). This module transmits over the 2.4 GHz ISM band like traditional Bluetooth, however BLE uses considerably less power while still maintaining its effective communication range. This makes BLE a viable option for IoT communication between devices.

1.1 Overview

The BLE device communicates with a host microcontroller using UART. Depending on the supplier the device is purchased through will determine the number of pins. For this tutorial, the four middle pins will be used: `RXD`, `TXD`, `GND` and `VDD`.

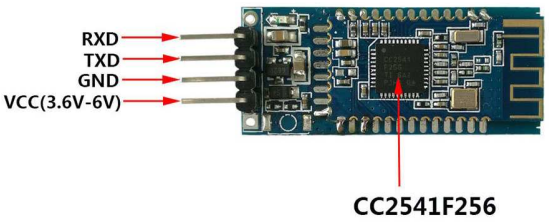

Figure 2: HM-10 Pins

The commands to control this device have to be understood before it is plugged into a microcontroller. To do this, an FTDI module can be used to connect the UART pins of the HM-10 directly to a computer. There, a serial connection can be established with the module and will give the user an easier time testing and debugging commands.

Any FTDI module will work, but be sure to purchase one with a genuine FTDI chip. Most computers have firmware installed already that will make these devices plug-and-play ready. However if the FTDI chip is not a genuine one, the modern firmware will be able to tell and ".*" the counterfeit chip, rendering it unusable.

The FTDI component used here is the SH-U09F, produced by DSD Tech as well.

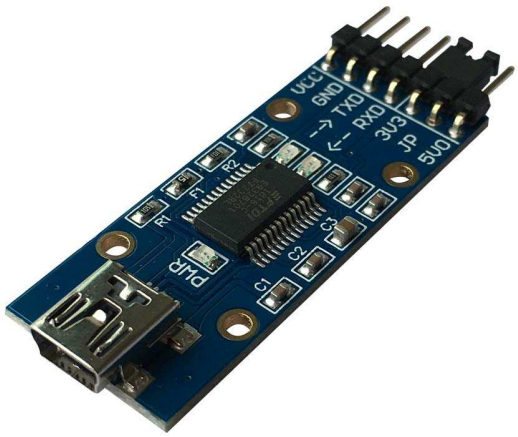

Figure 3: SH-U09F FTDI Module

The current drivers should be downloaded and installed before plugging the FTDI component into the computer. The current FTDI drivers can be found at `http://www.ftdichip.com/Drivers/VCP.htm`. There you can select the correct firmware for your operating system. Once installed plug the device into your computer. The SH-U09F module here uses a micro USB cable to connect to the host computer.

After the computer finishes configuring the FTDI module, unplug the device. Next the FTDI module will be connected to the BLE module. The HC-10 device uses 3.3 V logic and power, so on the SH-U09F module the jumper between the 5V0 pin and JP pin should be changed to connect the 3V3 and JP pins instead. This changes the logic and VCC voltages to 3.3 V. Next, the pins of the devices can be connected with jumper wire. The following diagram should be followed:

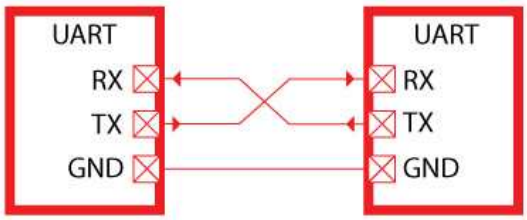

Figure 4: UART Communication Configuration

In UART communication, the receive line (Rx) on one device is connected to the transmit (Tx) of the other, and visa-versa. VCC and GND should also be connected.

Once connected a serial communication program can be used to send and receive data from the user's computer to the end device on the other end of the FTDI module. The program used here to send and receive the UART data is Realterm: Serial Terminal: `https://realterm.sourceforge.io/index.html#downloads_Download`. Go to the link and click the download link at the top of the page to get the install executable.

Once installed and opened, the program will connect automatically to a device with a COM port. So plug in the FTDI module again, then open the **Device Manager** by clicking the *Start* menu, then type *Device Manager* in the *Search* text box. Open the Device Manager once it pops up.

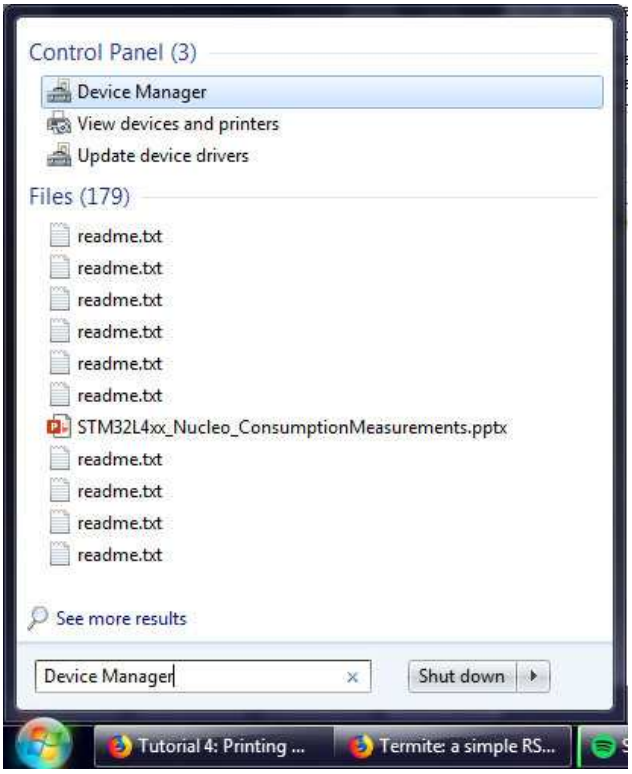

Figure 5: Selecting Device Manager from the Start Menu

Once opened, go to the *Ports* list to identify the COM number of the FTDI module. In this case it is *COM11*.

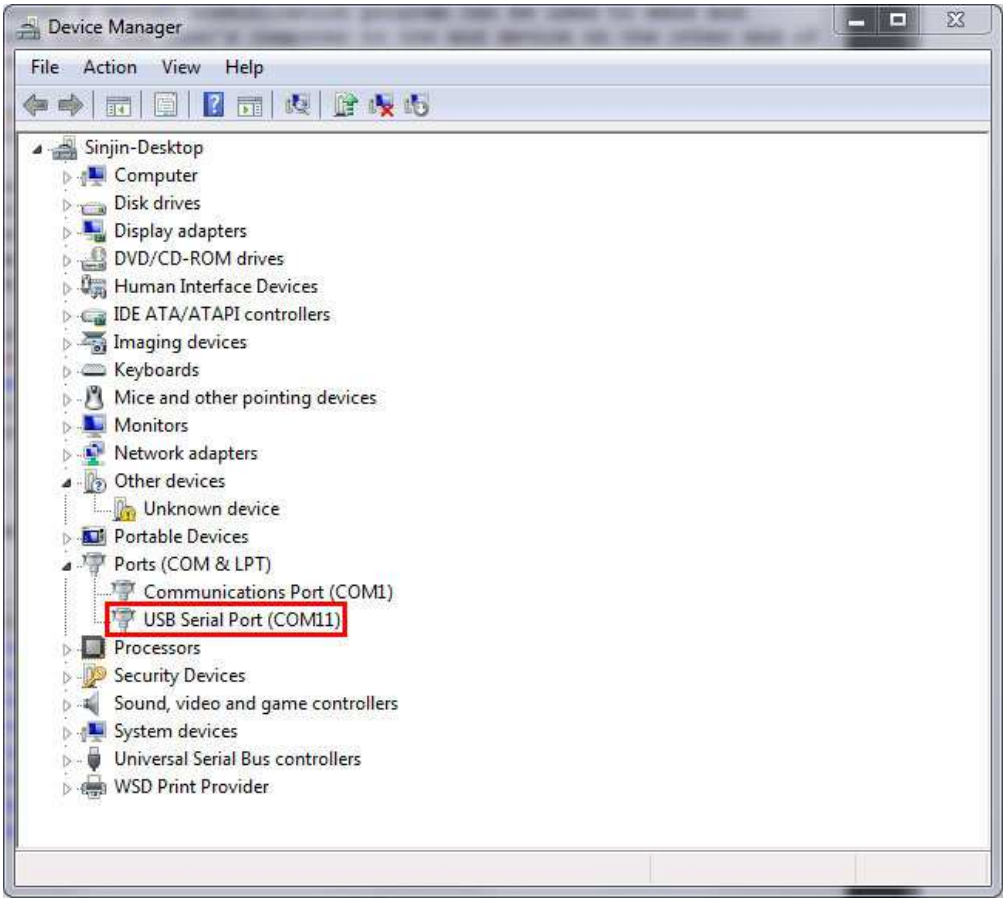

Figure 6: Device Manager

If there are many devices connected and it is difficult to tell which is the correct one, then right click on one of them at a time a go to **Properties**. Under *Manufacturer* should be *FTDI* for the correct module.

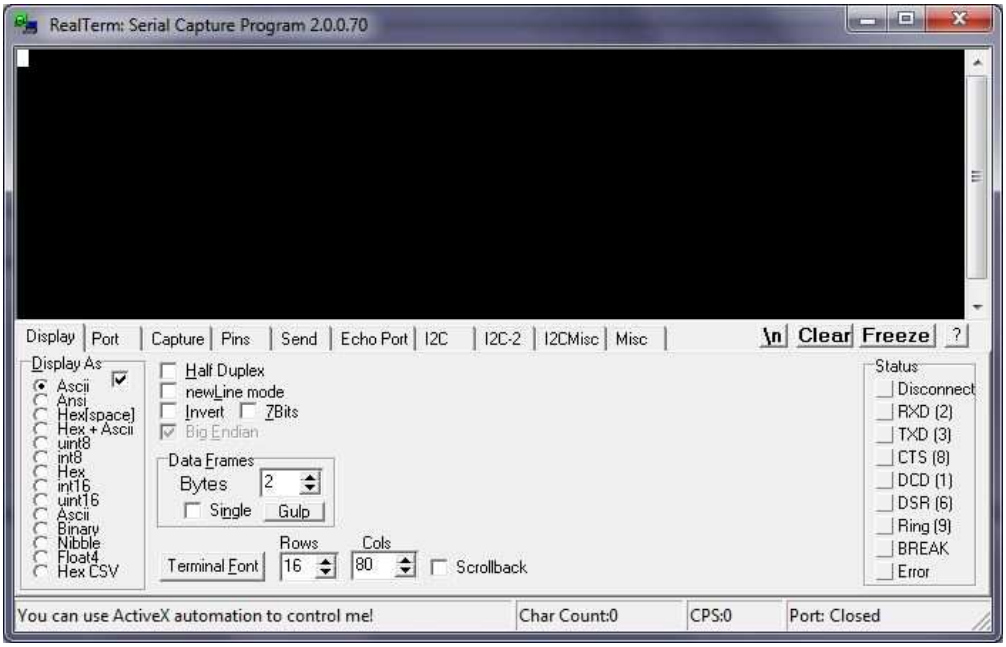

Figure 7: Realterm: Serial Monitor when it first starts.

The first tab displayed in Realterm is just how the serial information is displayed. Leave the *Display As* option as *ASCII*. Next, under the *Port* tab, select the correct port number for the FTDI device and a Baud of 9600. Then click on the *Open* button to connect to the module

Figure 8: Realterm Port Settings for FTDI Device.

The FTDI device will now be ready to talk to. To start, click on the *Send* tab. There will

be two identical lines with a text input and buttons reading *Send Numbers* and *Send ASCII*. Either of the two lines can be used to send data. There will be cases where you'll want to send raw binary/hexadecimal data to the device where the *Send Numbers* will be useful. But for configuring the FTDI module ASCII encoded characters are what we need. Also, to have responses on separate lines, click the *After* check box in the $\backslash n$ section. Otherwise all the FTDI responses will run on the same line.

Now the connection can be tested. In the text box type **"AT"** (without quotes) and click *Send ASCII*. If the connection is successful there should be a response of **"OK"** in the window.

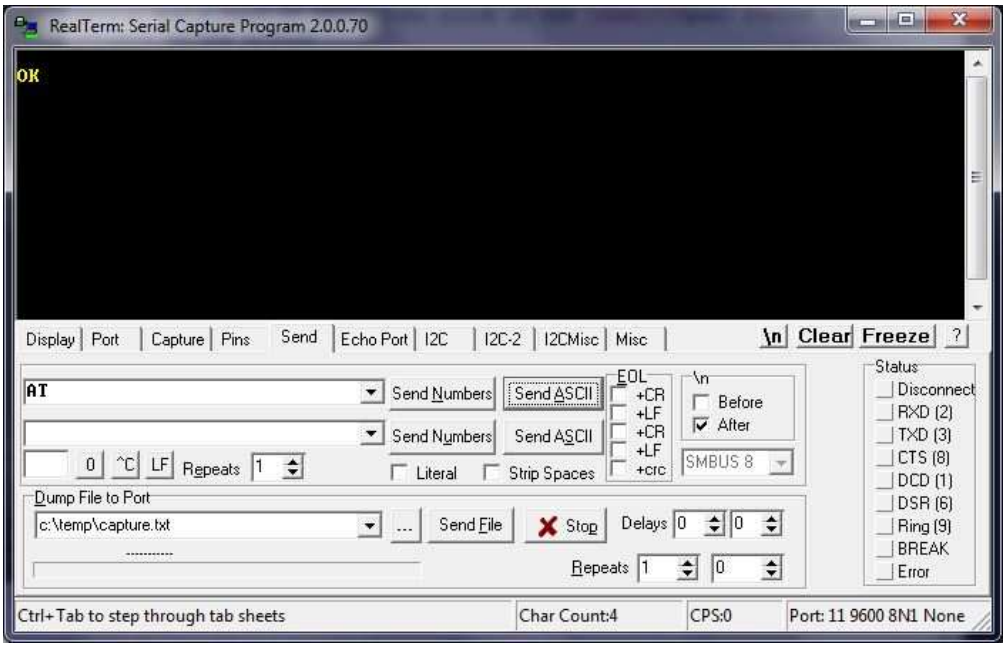

Figure 9: Successful FTDI Communication in Realterm.

1.2 HM-10 Module Configuration

Some default settings in the HM-10 module need to be changed after the module is wired up and communication is verified in Realterm.

While in Realterm, type: **"AT+IMME?"**. If the response is **"OK+Get:0"**, then type **"AT+IMME1"** to change the settings. This setting is used to tell the module to start in *command mode* or *work mode* when the device starts up. In command mode (changed by sending *"AT+IMME1"*), the component will only perform what the user asks based on the **AT** commands. In work mode (*"AT+IMME1"*), the device will automatically perform actions like connecting to neighboring devices, enabling discovery, etc. based on its current settings on start up.

Next, we'll want to see what message the device can display, so the next command will turn on notifications: **"AT+NOTI1"**. Then, type **"AT+NOTP1"** to show MAC

addresses related to the notification.

The HM-10 device likely has a default name (not the same as the MAC address). So give the device a unique name by typing **"AT+NAME[name]"** where "[name]" is a string. For example, if you want to name the device *Bulldogs*, then the command would be **"AT+NAMEBulldogs"**.

Next, all Bluetooth Low Energy devices use some UUID value to identify what type of information to expect in a standardized manner. To change the UUID value sent the command: **"AT+UUID[uuid]"** where "[uuid]" is a valid UUID value in hexadecimal with a hexadecimal prefix. For example, if you wanted to use UUID value 0x181A, which is the Bluetooth assigned number for the *Environment Sensing* service. For more information on the Bluetooth protocol, read the *Bulldog BLE Handbook* (coming soon).

Similarly, the service characteristic value can also be set on the HM-10. Under *Environmental Sensing Service* there are several characteristics, one of which is *Temperature* with a UUID of 0x2A6E. Use the command **"AT+CHAR[char]"** where "[char]" is the UUID value of the characteristic to set the device's characteristic.

While the device is connected to another Bluetooth device, no AT-commands will be valid until the connection is broken. This can be done manually by simply typing **"AT"**. The module should return **"OK+LOST"** when successfully disconnected. This step should not be necessary if *AT+TCON* and *AT+IMME* are set.

1.3 Connecting Two HM-10 Devices

Two HM-10 devices can be connected together once the settings listed previously are changed. Actually the devices *could* be connected prior to running those commands, but to save a head-ache or two its advisable to run them first.

Multiple instances of Realterm can be run on the same computer so this process can be completed on a single computer with two HM-10 devices connected to it. Whether you decide to connect the other HM-10 to the same computer or not, perform the same steps in the *Overview* and *HM-10 Module Configuration* sections of this document to set up the second FTDI and HM-10 device separately.

Once both devices are setup and connected to the computer, then they can be paired and bonded. To do so takes a few steps.

First, one device must be selected as the master and the other the slave device. In Bluetooth Point-to-Point protocols, the slave device is the one with data from sensor readings and the master device is the one that connected to the slave to gather that data.

To set the master/slave roles for each device use the command **"AT+ROLE"**.

1.3.1 Setting up Slave Device

To change the role of the module to a slave use the command: **"AT+ROLE0"**. Doing this starts the module advertising its presence to BLE master devices. As a note, some datasheets call the slave device the Peripheral Node.

Next, to change the interval in which the slave sends an advertisement use the command: **"AT+ADVI[P1]"**, where **[P1]** is a value 0 through F and represents different time intervals in Table 1:

[P1]	Advertising Interval (ms)	**[P1]**	Advertising Interval (ms)
0	100	8	1022.5
1	152.5	9	1285
2	211.25	A	2000
3	318.75	B	3000
4	417.5	C	4000
5	546.25	D	5000
6	760	E	6000
7	852.5	F	7000

Table 1: ADVI Values

The default value of 1285 ms is fine for our purposes.

Also to keep in mind, there are different advertising types under the **"AT+ADTY"** command. The default allows advertising, scanning response and makes the device openly connectable. To set the device back to its default type use: **"AT+ADTY0"**.

Lastly, some device wont advertise unless the advertising data flag is set. To do this the *AT+FLAG* command is used. Type **"AT+FLAG0"** and click *Send ASCII*. This command can use values from 0 through FF, however using any value will get the device to advertise.

Now the slave device is advertising its presence and ready to be connected to.

1.3.2 Setting up Master Device

Setting up the master is a little simpler than the slave device. The command to set the device to master is **"AT+ROLE1"**. The master is also called the Central Node in some datasheets.

Once set the device can immediately begin scanning for slave devices. Type **"AT+DISC?"** to do so. If the slave is configured correctly the terminal for the master should read: **"OK+DISCS"** (starting discovery), then **"OK+DIS0:[MACADDR]"** (where **[MACADDR]** is the MAC address of the slave module), and finally **"OK+DISCE"** (for ending discovery process). Likely all of these messages will be on the same line.

If there were multiple BLE slave devices available and in range there will be a **"OK+DIS1:[MACADD** The important thing here is the value after *OK+DIS*. This value represents the array index of the connectable BLE devices. So it will be important to know the MAC address of the slave HM-10 module to identify which device the master should connect to.

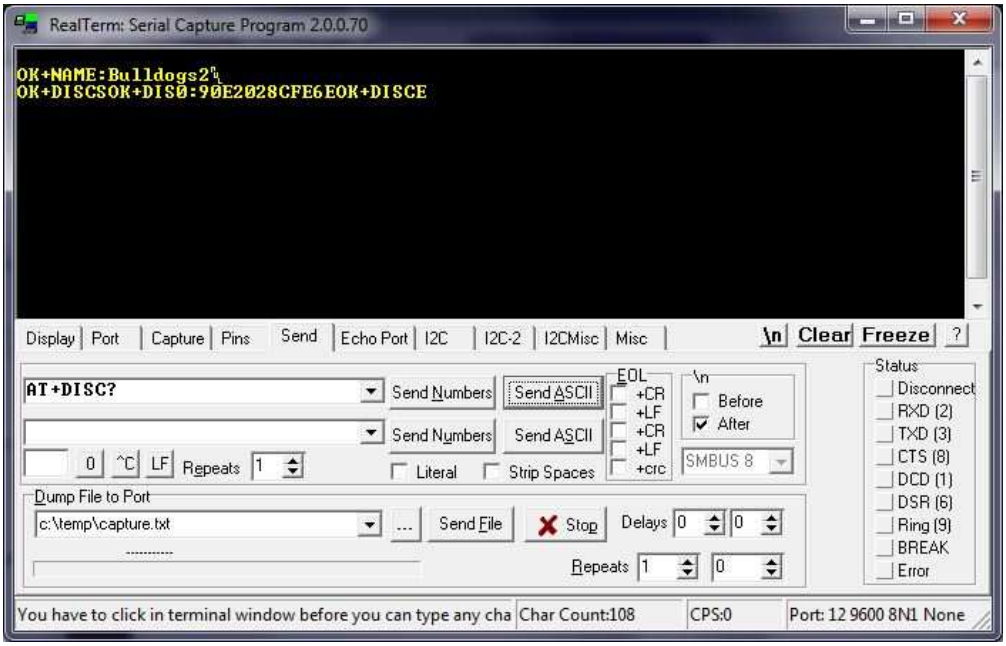

Figure 10: Master Discovery Readings.

1.3.3 Discovery Connection

From Figure 10, the slave device is identified by **"OK+DIS0:90E2028CFE6E"** meaning that the device with MAC address 90:E2:02:8C:FE:6E can be connected to using index 0. So now the master can connect to this discovery device using the command **"AT+CONN[DISC]"** where **[DISC]** is the array index. So in the case of Figure 10 the command would be **"AT+CONN0"**. Doing this connects to the slave device and shows in both devices terminals that they are connected to each others MAC addresses.

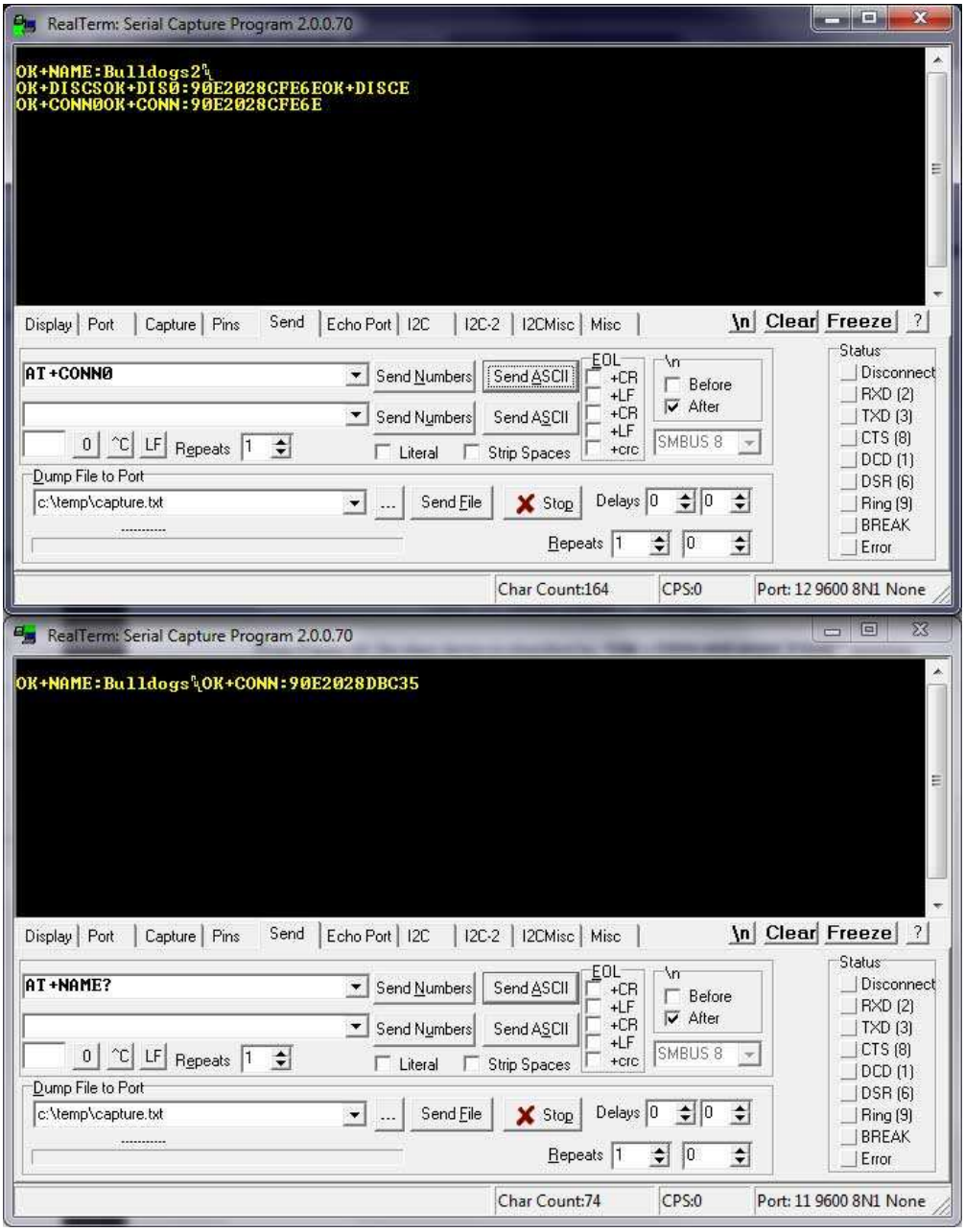

Figure 11: Master (Bulldogs2) and Slave (Bulldogs) Devices Connected.

Now simple strings or numbers can be transmitted from slave to master and visa-versa. In transmit mode, none of the *AT* commands will be recognized except the **"AT"** command, which disconnects both devices.

Note: Once the slave device is disconnected the advertising flag needs to be set again before it can be discovered.

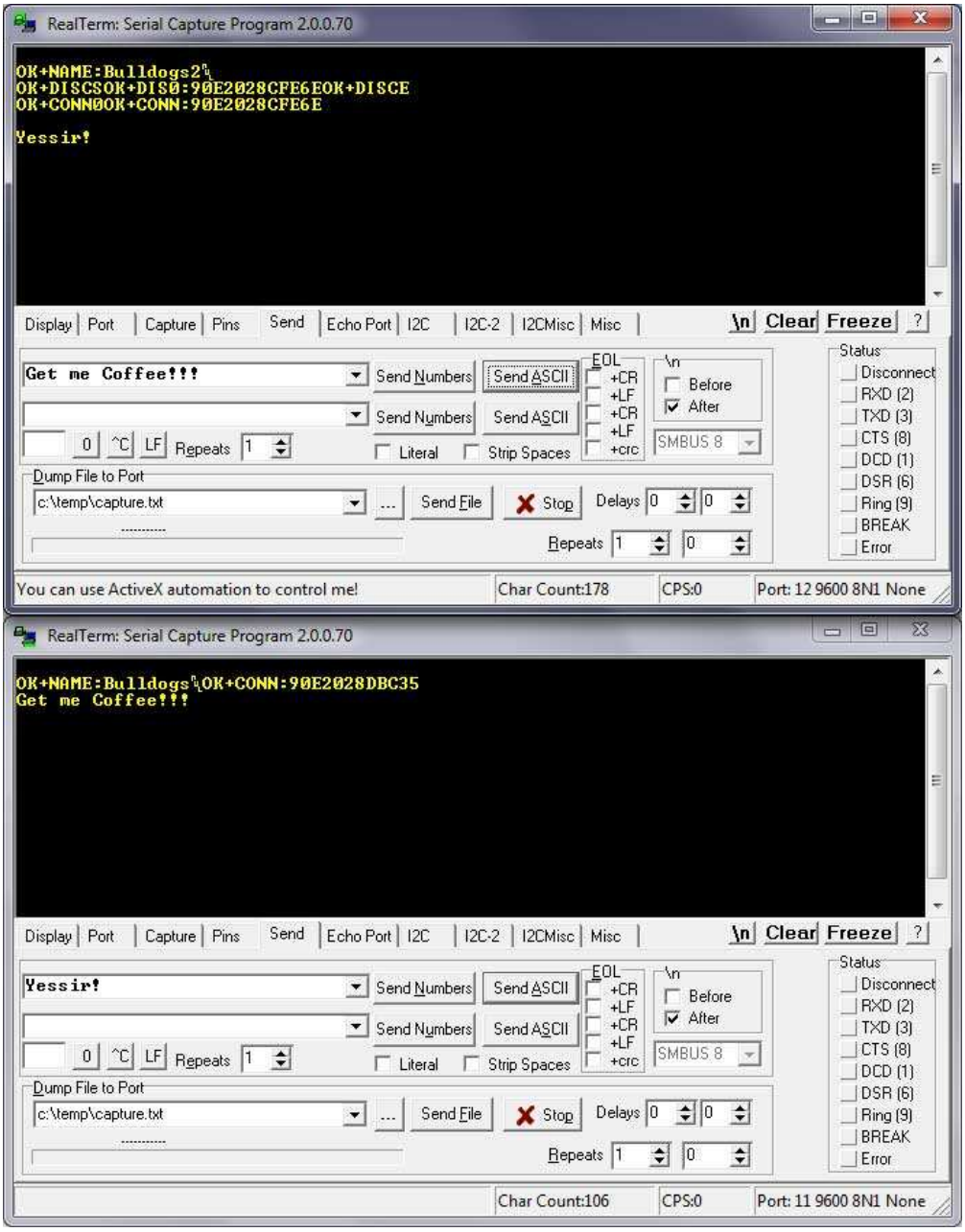

Figure 12: Master and Slave Sending Strings.

When connecting to a device its sometimes more desirable to connected to a device based on its MAC address instead of discovery array index. The command **"AT+CON[MACADDR]"** can be used to so this from the master. The discovery command should still be run first to know if the slave device is available to be connected to first.

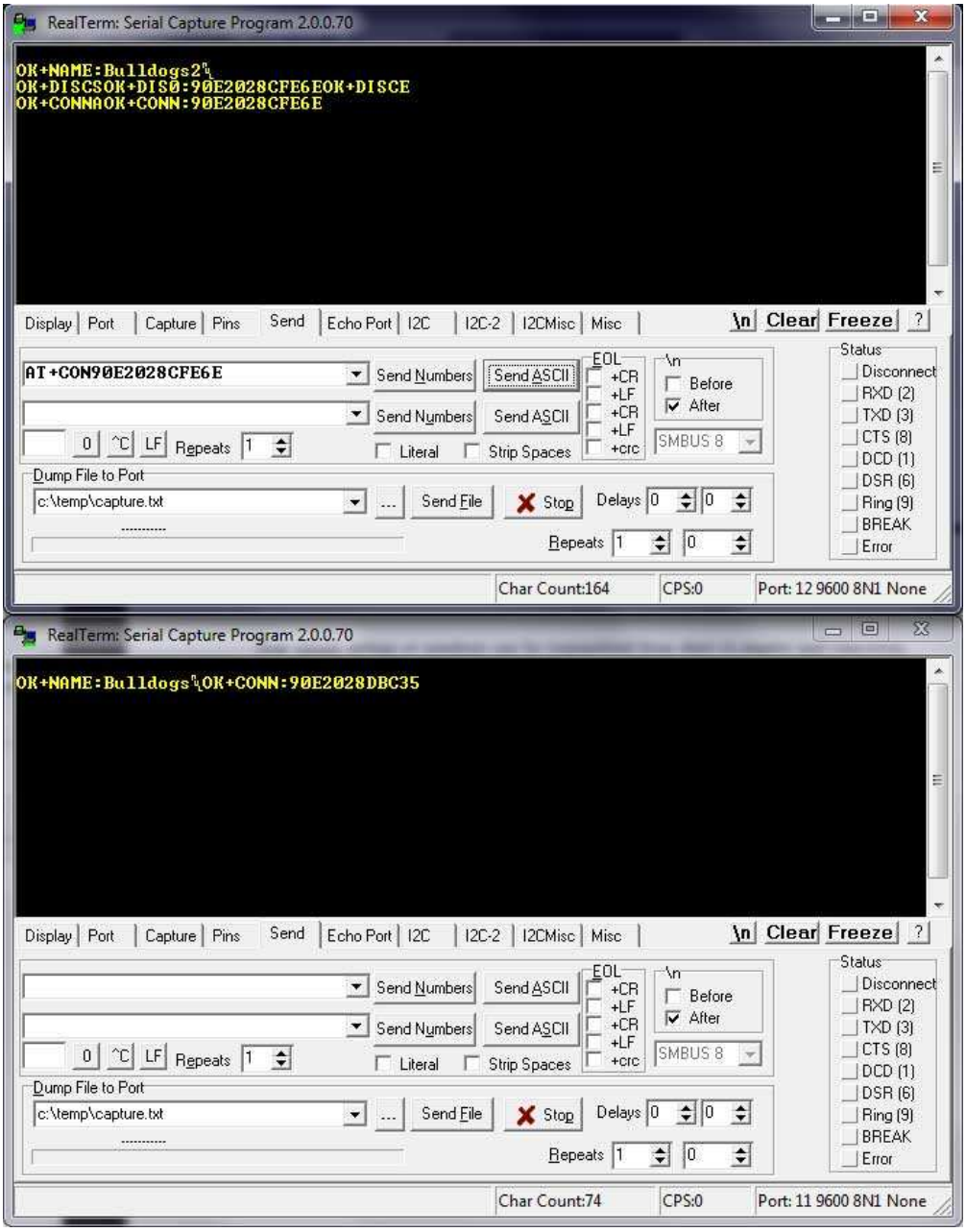

Figure 13: Master Connecting to Slave Based on MAC Address.

1.4 Connecting to HM-10 from Android Phone

In the last part, two HM-10 devices were connected, one being the master and the other the slave. In this part, one HM-10 module will be used as a slave device and a smart phone will be used as the master device. This part completely depends on the capabilities of your smart phone. Make sure your phone has BLE capabilities to be able to do this.

If the phone does have BLE capabilities, the *nRF Connect* can be downloaded from the Google Play Store. nRF Connect is an app by Nordic Semiconductor used to connect and communication to any BLE device.

First however, one of the HM-10 modules should be set up as a slave device as shown in the last section. In addition, the UUID for the slave device's service and characteristic should be configured as described in the *HM-10 Module Configuration*. Make sure to reset the module after making changes to apply them using the **"AT+RESET"** command.

Once the slave device is set up and advertising its presence, the Android device can detect and connect to it. To do so, install and open the nRF Connect app. If the Bluetooth adapter is disabled it there will be a red bar instructing you to enable it.

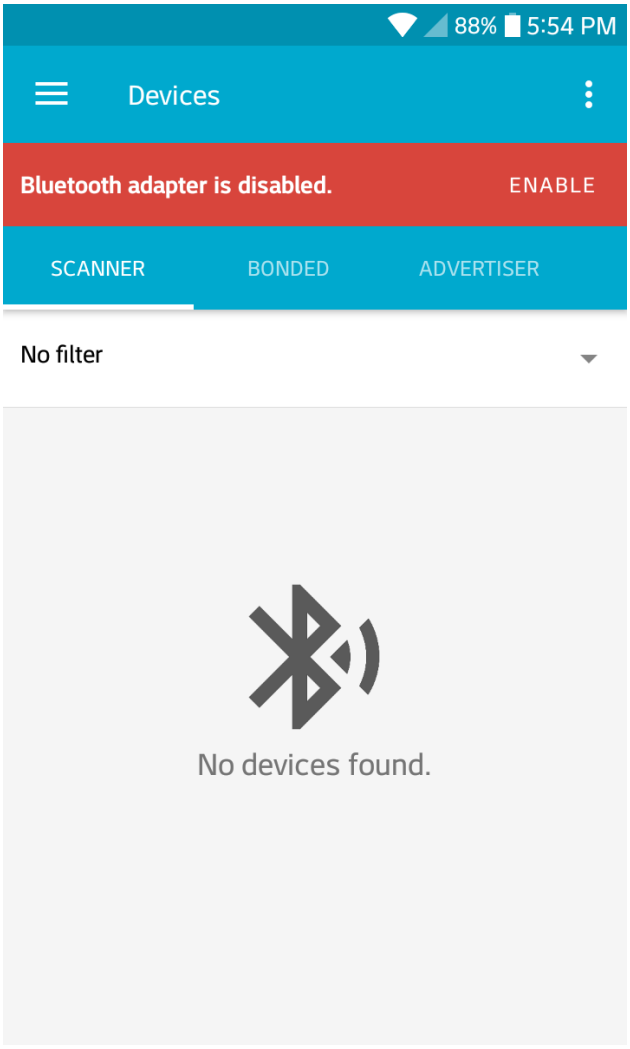

Figure 14: Bluetooth Adapter Disabled.

Once enabled, click the *Scan* button on the top right. After a moment, the HM-10 device should show up with its name.

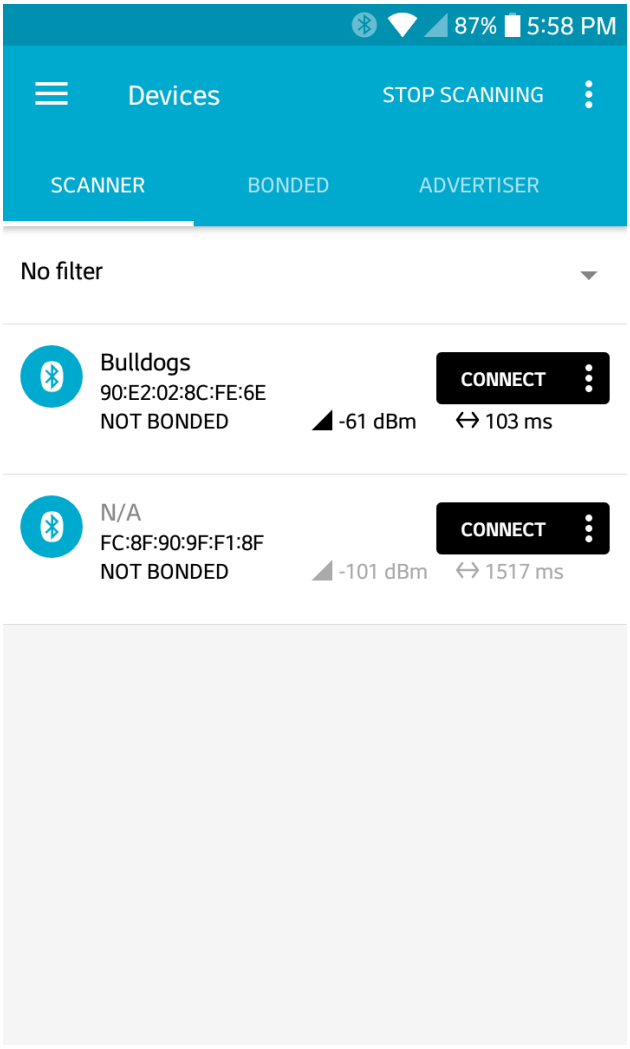

Figure 15: Scanning for BLE Devices.

Click on the *Connect* button on the HM-10 device. Doing this will open a new tab will open with the name of the module. In this tab there are three main services, two generic services and one *Environmental Sensing*. Click on the Environmental Sensing service. If the UUIDs were set up correctly, the values used in the configuration were for the Bluetooth Environment Sensing Service (ESS) and Temperature Characteristic within the ESS.

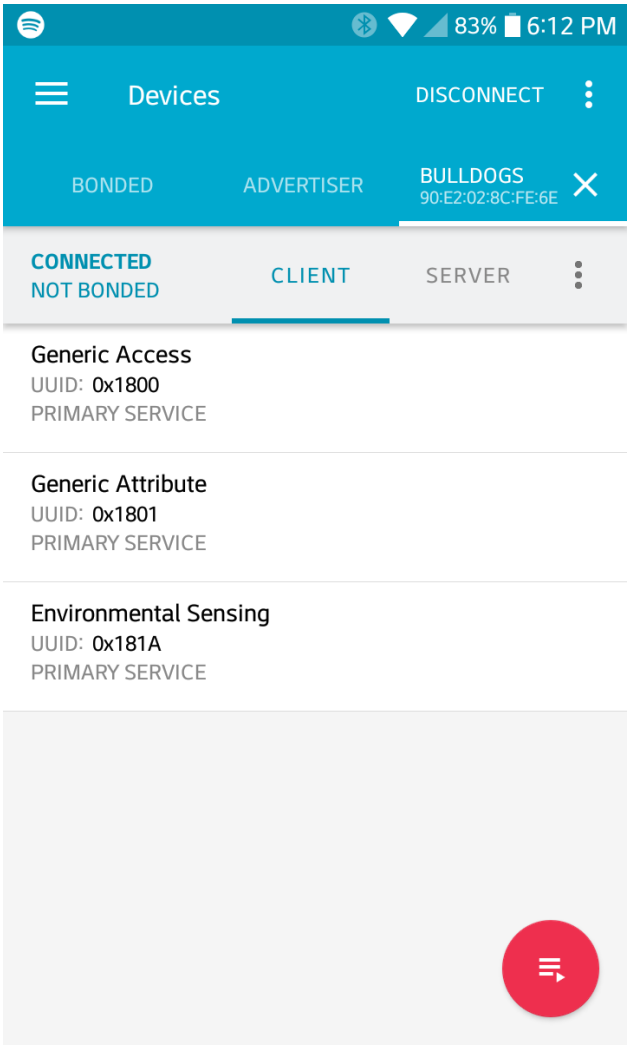

Figure 16: HM-10 Services.

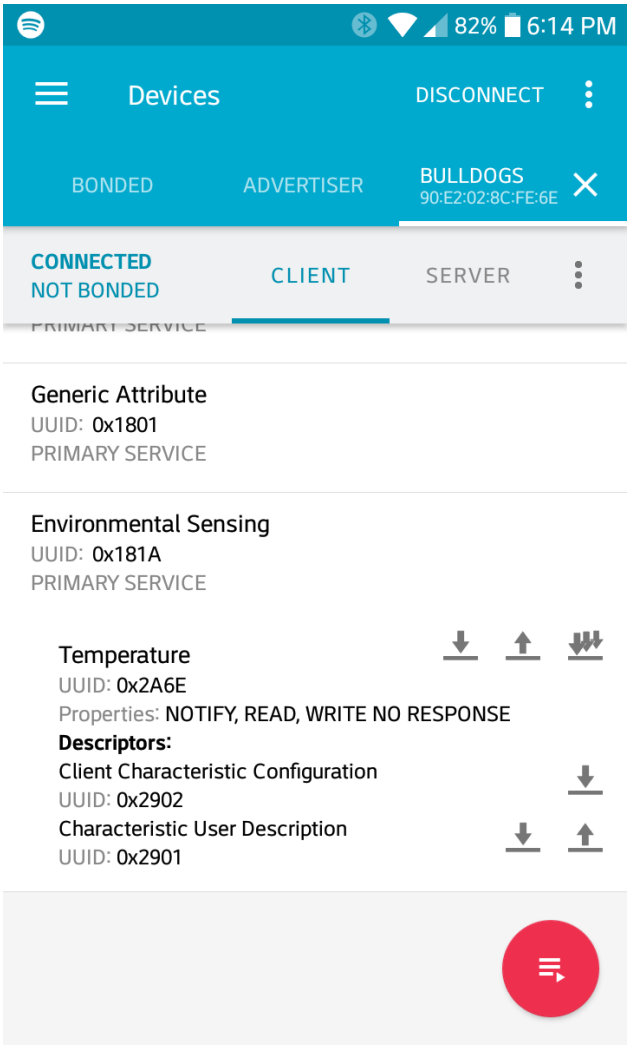

Figure 17: HM-10 ESS Characteristics.

However, there is no data for the temperature reading yet. We don't have an actual temperature sensor connected to the BLE module, so we'll have to fake it for now.

The temperature value can be transmitted from Realterm in the form of raw hexadecimal values. Bluetooth gives specifications for formatting data for all characteristics. The ESS Specification for the Temperature Characteristic allows a degree resolution down to 0.01°C. Also, the data type of the value is an `sint16`, or 16-bit signed-integer. To fix the decimal point problem that would be lost in using a integer the temperature value needs to be multiplied by 10^2 before being converted into signed-integer. So a temperature reading of 34.56°C would be 3456 and 0x0D80 in hexadecimal. Also, signed-integers used Two's Compliment to represent negative values. Thus, a temperature reading of -10.35°C would be multiplied by 10^2 and converted to Two's Compliment to get a result of `0xFBF5`.

So lets use the temperature value of 34.56°C to send to the Android device connected to the HM-10. In Realterm we can send these values in their signed-integer, hexadecimal form

with a couple things to consider. During the UART transmission, the Endianness of the bytes of the numbers are reversed, so the byte order will need to be fixed manually. Also in Realterm, each byte has to be expressed individually with a prefix, so 0x0D80 would need to be sent as 0x0D and 0x80. And if we take into consideration the byte reversal then 0x0D80 becomes 0x80 and 0x0D.

So with this in mind, go to Realterm and in the *Send* tab, type **"0x80 0x0D"** in the input box. Then instead of clicking *Send ASCII*, click *Send Numbers* instead to send the raw hexadecimal values.

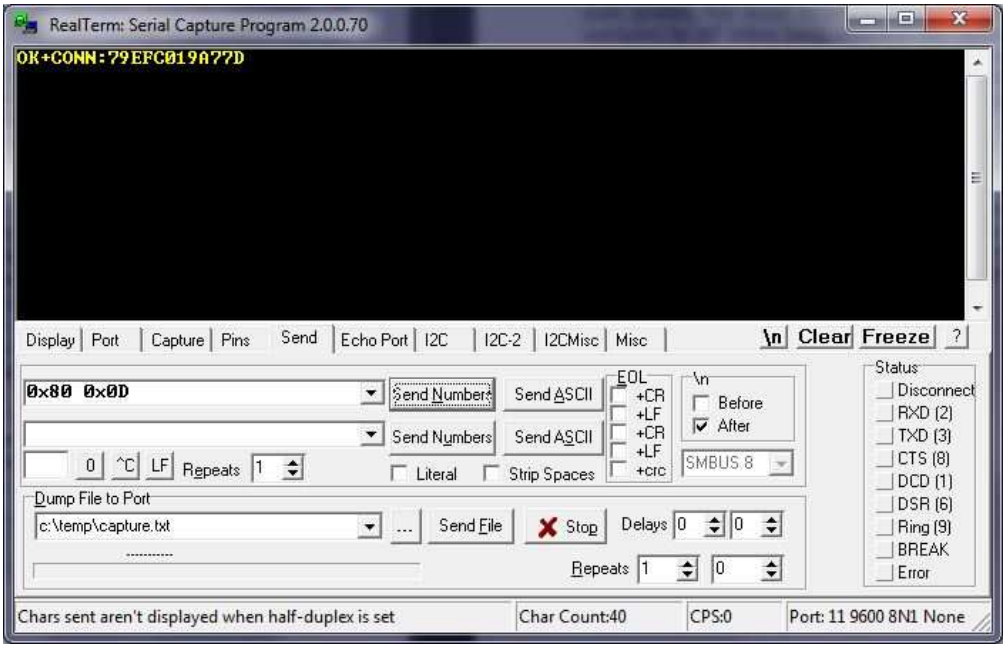

Figure 18: Sending Temperature Data.

Now checking the Andriod device, it can be seen that a *Value* line appears with the temperature reading of 34.56°C:

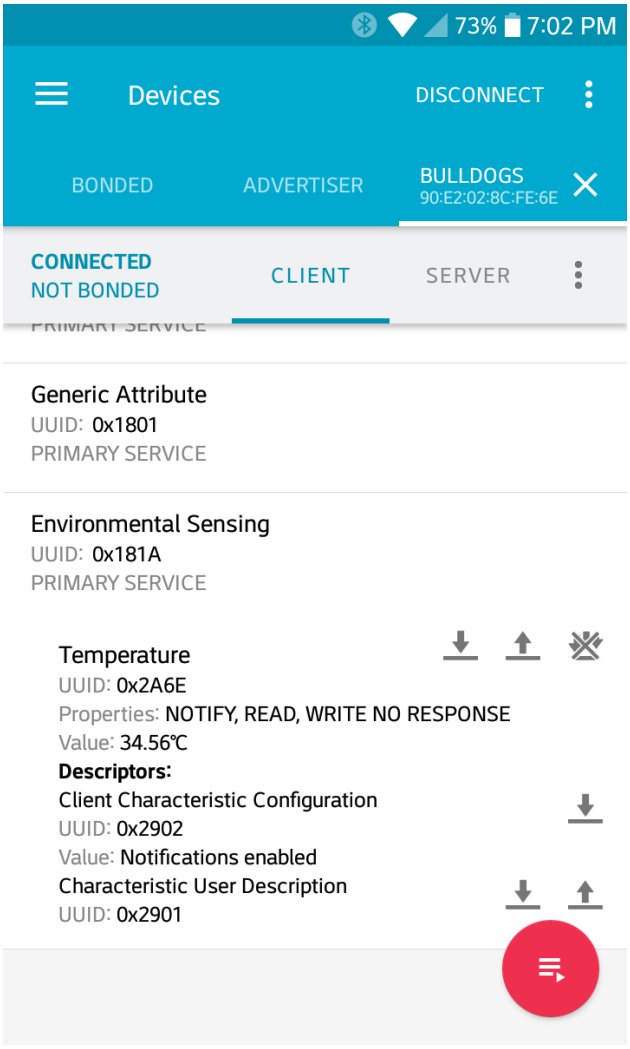

Figure 19: Temperature Reading in nRF Connect.

References

- JNHuaMao Technology Company, "HM Bluetooth module datasheet," Bluetooth 4.0 BLE module Datasheet, Mar. 2015.
- "GATT Services", *Bluetooth.com*, 2019. [Online]. Available: https://www.bluetooth.com/specifications/gatt/services/.
- M. Currey, "HM-10 Bluetooth 4 BLE Modules", *Martyncurrey.com*, 2017. [Online]. Available: http://www.martyncurrey.com/hm-10-bluetooth-4ble-modules/.