

Developing an application communicating with BLE devices on Dusun's gateways

Bluetooth Low Energy (BLE) is intended to provide considerably reduced power consumption and cost while maintaining a similar communication range with Bluetooth. BLE is suitable for IoT applications. Many operating systems support BLE and thus BLE is very popular. The Dusun IoT gateways support most connectivity protocols including BLE (4.x/5.0/Mesh), ZigBee 1.2/3.0, Z-Wave, TCP/UDP, etc. Users can connect the gateway with various BLE devices. As the Dusun gateway is running based on the Linux system, how to implement an application over Bluez stack to 'talk' with BLE devices on Dusun gateways will be presented here.

1. Product Feature Summary

- **System :**
- OS: Linux@ OpenWrt
- Processor: MTK7620A (MIPS24KEc(580MHZ))
- RAM: 128MB
- Flash: 64MB
- **Wireless protocol :**
- LTE-M1
- Bluetooth Low Energy
- Wi-Fi

- Zigbee3.0
- Z-Wave
- Ethernet
- WLAN
- LAN

2. System block diagram

Dusun's gateways run Linux OpenWrt system. As depicted in figure 1, Linux support BLE protocol through running the Bluez stack. BlueZ provides support for the core Bluetooth layers and protocols. It is flexible, efficient and uses a modular implementation. The BlueZ stack supports all core Bluetooth protocols and layers now. Users can write BLE applications using Bluez APIs to manipulate BLE devices.

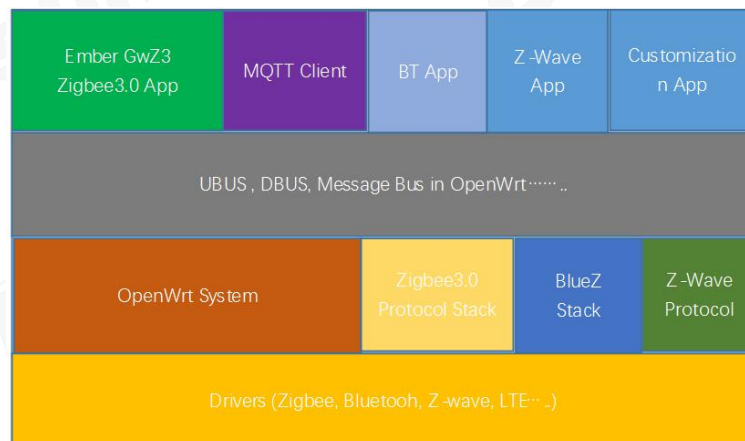


Figure 1 Dusun Gateway system architecture

3. System configuration

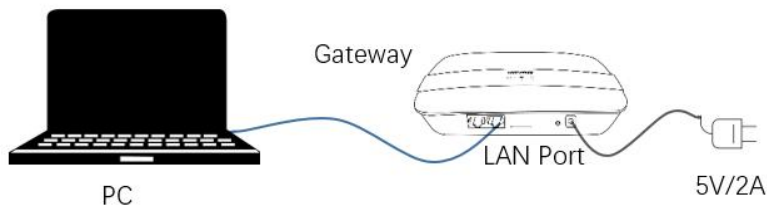
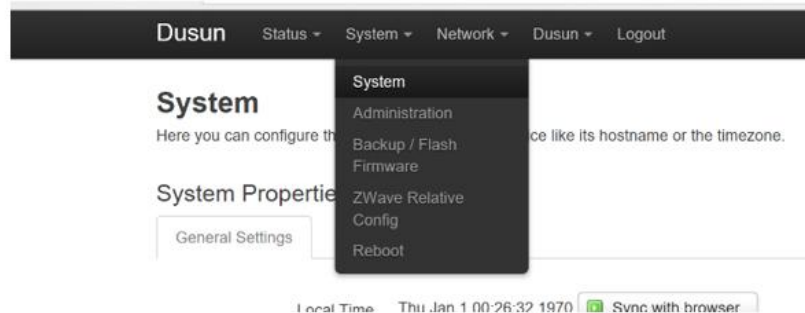


Figure 2 gateway->PC connection diagram

The gateway can be configured following these steps:

1. Connect the gateway to the PC and power up, according to figure 2;
2. Open a web browser on PC, Input Gateway IP Address: 192.168.66.1 ; Enter the username and Password (Username: root Password: root) , login to the gateway;

3. Make system configuration at the system menu, the timezone, password and others can be revised here.



4. Configure network at the network sub-menu.

Dusun Status System Network Dusun Logout AUTO REFRESH ON

Interface Overview

Network	Status	Actions
LAN br-lan	Uptime: 0h 47m 39s MAC-Address: 30:AE:7B:2B:4D:68 RX: 238.91 KB (2839 Pkts.) TX: 505.31 KB (2735 Pkts.) IPv4: 192.168.66.1/24 IPv6: fd02:cf79:968::1/60	Connect Stop Edit Delete
WAN eth0.2	Uptime: 0h 0m 0s MAC-Address: 30:AE:7B:2B:4D:69 RX: 0 B (0 Pkts.) TX: 331.27 KB (991 Pkts.)	Connect Stop Edit Delete
WAN6 eth0.2	Uptime: 0h 0m 0s MAC-Address: 30:AE:7B:2B:4D:69 RX: 0 B (0 Pkts.) TX: 331.27 KB (991 Pkts.)	Connect Stop Edit Delete
WANPPP 3g-wanPPP	RX: 0 B (0 Pkts.) TX: 0 B (0 Pkts.)	Connect Stop Edit Delete

Add new interface...

Global network options

IPv6 ULA-Prefix:

Save & Apply Save Reset

Dusun Status System Network Dusun Logout AUTO REFRESH ON

WAN WAN6 WANPPP LAN

Interfaces - LAN

On this page you can configure the network interfaces. You can bridge several interfaces by ticking the "bridge interfaces" field and enter the names of several network interfaces separated by spaces. You can also use VLAN notation INTERFACE.VLANID (e.g.: eth0.1).

Common Configuration

General Setup Advanced Settings Physical Settings Firewall Settings

Status: br-lan

Uptime: 1h 18m 52s
MAC-Address: 30:AE:7B:2B:4D:68
RX: 572.65 KB (6380 Pkts.)
TX: 1.17 MB (6287 Pkts.)
IPv4: 192.168.66.1/24
IPv6: fd02:cf79:968::1/60

Protocol: Static address

IPv4 address:

IPv4 netmask: 255.255.255.0

IPv4 gateway:

IPv4 broadcast:

Use custom DNS servers:

When the configuration is completed please reboot the gateway. Then users can log in the gateway by inputting the following commands in a terminal, and then input your password. IP_addr is the IP addr which user configured above. The default IP is 192.168.66.1. The below figure shows the login scene.

```
ssh root@IP_addr
```

```
guojie@guojie-Inspiron-5577:~$ ssh root@192.168.66.1
root@192.168.66.1's password:
```

BusyBox v1.23.2 (2019-03-12 19:08:20 CST) built-in shell (ash)

```

+-----+
|         |   +-----+   +-----+   +-----+   +-----+
|         |   |         |   |         |   |         |   |         |
|         |   |         |   |         |   |         |   |         |
|         |   |         |   |         |   |         |   |         |
+-----+   +-----+   +-----+   +-----+   +-----+
```

W I R E L E S S F R E E D O M

CHAOS CALMER (Chaos Calmer, r47202)

* 1 1/2 oz Gin	Shake with a glassful
* 1/4 oz Triple Sec	of broken ice and pour
* 3/4 oz Lime Juice	unstrained into a goblet.
* 1 1/2 oz Orange Juice	
* 1 tsp. Grenadine Syrup	

```
root@dusun:~# █
```

4. Connecting a BLE sensor with Bluetoothctl

The Dusun gateway is installed Bluez version v5.50. Bluez has provided many command utilities for controlling BLE devices. Bluetoothctl is the main command for configuring Bluetooth devices on Linux. Users can use this utility to know more about their BLE devices and are familiar with the connecting steps. An oximeter BLE sensor is used for living example. The steps for connecting and managing the oximeter BLE sensor is shown below:

1. Open Bluetoothctl and scan Bluetooth devices to find the oximeter sensor(whose name is my oximeter).

```
root@Dusun:~# bluetoothctl
Agent registered
[bluetooth]# power on
Changing power on succeeded
[CHG] Controller 00:02:5B:00:A5:A5 Powered: yes
[bluetooth]# scan on
Discovery started
[CHG] Controller CC:2F:71:E1:9C:21 Discovering: yes
[NEW] Device A4:C1:38:DC:4D:C5 My Oximeter
[NEW] Device A4:E6:15:CA:13:FC SWTV
[NEW] Device 53:71:73:C7:82:51 53-71-73-C7-82-51
[bluetooth]# scan off
Discovery stopped
```

2. Pair the sensor and connect it.

```
[bluetooth]# pair A4:C1:38:DC:4D:C5
Attempting to pair with A4:C1:38:DC:4D:C5
[CHG] Device A4:C1:38:DC:4D:C5 Connected: yes
Failed to pair: org.bluez.Error.AuthenticationFailed
[CHG] Device A4:C1:38:DC:4D:C5 Connected: no
[bluetooth]# connect A4:C1:38:DC:4D:C5
Attempting to connect to A4:C1:38:DC:4D:C5
[CHG] Device A4:C1:38:DC:4D:C5 Connected: yes
Connection successful
[NEW] Primary Service
/org/bluez/hci0/dev_A4_C1_38_DC_4D_C5/service0008
0000180a-0000-1000-8000-00805f9b34fb
Device Information
[NEW] Characteristic
/org/bluez/hci0/dev_A4_C1_38_DC_4D_C5/service0008/char0009
00002a50-0000-1000-8000-00805f9b34fb
PnP ID
[NEW] Primary Service
/org/bluez/hci0/dev_A4_C1_38_DC_4D_C5/service000b
cdeacb80-5235-4c07-8846-93a37ee6b86d
Vendor specific
[NEW] Characteristic
/org/bluez/hci0/dev_A4_C1_38_DC_4D_C5/service000b/char000c
cdeacb81-5235-4c07-8846-93a37ee6b86d
Vendor specific
[NEW] Characteristic
/org/bluez/hci0/dev_A4_C1_38_DC_4D_C5/service000b/char0010
cdeacb82-5235-4c07-8846-93a37ee6b86d
Vendor specific
```

3. List attributes and select the characteristic for getting data. Use `select-attribute` and then `notify on`, the data will be pushed to the command line as the below figure shows.

```
[My Oximeter]# menu gatt
Menu gatt:
Available commands:
-----
list-attributes [dev]          List attributes
select-attribute <attribute/UUID> Select attribute
attribute-info [attribute/UUID] Select attribute
read                          Read attribute value
write <data=xx xx ...>        Write attribute value
acquire-write                 Acquire Write file descriptor
release-write                 Release Write file descriptor
acquire-notify                Acquire Notify file descriptor
release-notify                Release Notify file descriptor
notify <on/off>               Notify attribute value
register-application [UUID ...] Register profile to connect
unregister-application         Unregister profile
register-service <UUID>        Register application service.
unregister-service <UUID/object> Unregister application service
register-characteristic <UUID> <Flags=read,write,notify...> Register application characteristic
unregister-characteristic <UUID/object> Unregister application characteristic
register-descriptor <UUID> <Flags=read,write...> Register application descriptor
unregister-descriptor <UUID/object> Unregister application descriptor
back                          Return to main menu
version                       Display version
quit                          Quit program
exit                          Quit program
help                          Display help about this program

[My Oximeter]# list-attributes
Primary Service
/org/bluez/hci0/dev_A4_C1_38_DC_4D_C5/service0008
0000180a-0000-1000-8000-00805f9b34fb
Device Information
```



```

b/char000c]# attribute-info /org/bluez/hci0/dev_A4_C1_38_DC_4D_C5/service000
Characteristic - Vendor specific
UUID: cdeacb81-5235-4c07-8846-93a37ee6b86d
Service: /org/bluez/hci0/dev_A4_C1_38_DC_4D_C5/service000b
Notifying: no
Flags: notify
00b/char000c]# select-attribute /org/bluez/hci0/dev_A4_C1_38_DC_4D_C5/service0
[My Oximeter:/service000b/char000c]# notify on
[CHG] Attribute /org/bluez/hci0/dev_A4_C1_38_DC_4D_C5/service000b/char000c Notifying: yes
Notify started
[CHG] Attribute /org/bluez/hci0/dev_A4_C1_38_DC_4D_C5/service000b/char000c Value:
80 0d 0c 0b 09 08 08 0a 0d 11 16 .....
[CHG] Attribute /org/bluez/hci0/dev_A4_C1_38_DC_4D_C5/service000b/char000c Value:
80 1b 1f 23 26 29 2a 2c 2c 2d 2d ...#&)*,--
[CHG] Attribute /org/bluez/hci0/dev_A4_C1_38_DC_4D_C5/service000b/char000c Value:
80 2d 2d 2e 2e 2e 2e 2e 2d 2c 2b .....*
[CHG] Attribute /org/bluez/hci0/dev_A4_C1_38_DC_4D_C5/service000b/char000c Value:
80 2a 28 26 23 21 1f 1e 1c 1a 19 ..*(&#!.....
[CHG] Attribute /org/bluez/hci0/dev_A4_C1_38_DC_4D_C5/service000b/char000c Value:
81 41 62 25 ..Ab%

```

After these steps, user can know something about the sensor and its BLE profiles which can also be seen from the sensor user manual. Then we will show how to program by using Bluez Dbus APIs to get the sensor data.

5. A sample: communicate with an oximeter BLE sensor using Bluez

We have provided a sample application to get the BLE oximeter sensor data. They include the Bluez Dbus library, and several c program files. The oximeter sensor operating functions is contained in the Oximeter.c. The main function in the main.c is shown as below.

```

int main(int argc, char *argv[])
{
    GError *error = NULL;
    GDBusClient *client;

```

```
INIT_LOG("HL");

/*init global data for bluez operation*/

init_global_data();

init_curl();

/*register the drivers you have written*/

load_healt_drivers();

/*create a main loop object*/

main_loop = g_main_loop_new(NULL, FALSE);

/*set up the dbus connection*/

dbus_conn = g_dbus_setup_bus(DBUS_BUS_SYSTEM, NULL, NULL);

/*create a bluez client for dbus connection object*/

client = g_dbus_client_new(dbus_conn, "org.bluez", "/org/bluez");

/* set connect/disconnect/signal handler function*/

g_dbus_client_set_connect_watch(client, connect_handler, NULL);

g_dbus_client_set_disconnect_watch(client, disconnect_handler, NULL);

g_dbus_client_set_signal_watch(client, message_handler, NULL);

/* set proxy handlers*/

g_dbus_client_set_proxy_handlers(client, proxy_added, proxy_removed,

property_changed, NULL);

/* set ready */
```



```
g_dbus_client_set_ready_watch(client, client_ready, NULL);

/*running in a loop*/

g_main_loop_run(main_loop);

/*release the resources*/

g_dbus_client_unref(client);

dbus_connection_unref(dbus_conn);

g_main_loop_unref(main_loop);

return 0;
}
```

Users should note the function `load_healt_drivers();`, in which the function `register_health_driver(&Oximeter_driver)` is invoked to register the callback functions which process the upcoming events and then get the data the sensor reported. The `UUID_Oximeter_SERVICE` is the primary service that its characteristic can be read to get oximeter values.

The `HealthDriver` and `BluetoothDeviceCallbacks` is defined as

```
typedef struct HealthDriver_
{
    char * name;
```

```

char * service_uuid;

struct BluetoothDeviceCallbacks_ * callbacks;

}HealthDriver;

typedef struct BluetoothDeviceCallbacks_
{
    void (*init) (BluetoothDevice * btdev);

    void (*exit) (BluetoothDevice * btdev);

    void (*scan_found)(BluetoothDevice * btdev);

    void (*connect_state_changed) (BluetoothDevice * btdev, bool connected);

    void (*service_added) (BluetoothDevice * btdev, GattService * service);

    void (*characteristics_added) (BluetoothDevice * btdev, GattCharacteristic *
characteristics);

    void (*characteristics_notify)(BluetoothDevice * btdev, GattCharacteristic *
characteristics,
const unsigned char * value, int len);

    //void (*property_changed)();
} BluetoothDeviceCallbacks;

```

In Oximeter.c a BluetoothDeviceCallbacks instance and a HealthDriver instance are defined:

```
static BluetoothDeviceCallbacks Oximeter_callbacks =
```

```
{
    .init = Oximeter_init,
    .connect_state_changed = Oximeter_connect_state_changed,
    .scan_found = Oximeter_scan_found,
    .characteristics_added = Oximeter_characteristics_added,
    .characteristics_notify = Oximeter_characteristics_notify,
};

#define UUID_Oximeter_SERVICE "cdeacb80-5235-4c07-8846-93a37ee6b86d"

HealthDriver Oximeter_driver =
{
    .name = " Oximeter detector",
    .service_uuid = UUID_Oximeter_SERVICE,
    .callbacks = &Oximeter_callbacks,
};
```

`Oximeter_init, Oximeter_connect_state_changed, Oximeter_scan_found,`

`Oximeter_characteristics_added, Oximeter_characteristics_notify` are functions to process connection events when the sensor is connected to the gateway.

`Oximeter_characteristics_notify` process the data notified. Users should write their own callback functions according to their sensor characteristics to control the sensor.

6. Run the oximeter sensor sample on the gateway

- 1) Get the Openwork toolchain and copy it to a Linux PC. Get the cross compile

OpenWrt-Toolchain from Dusun, whose name is

openwrt-sdk-ramips-mt7620_gcc-4.8-linaro_uClibc-0.9.33.2. Linux-x86_64.tar.bz2.

Decompress the downloaded OpenWrt Toolchain to a local folder (E.g.: home/software/OpenWrt-SDK).

- 2) Compiled the attached code files.

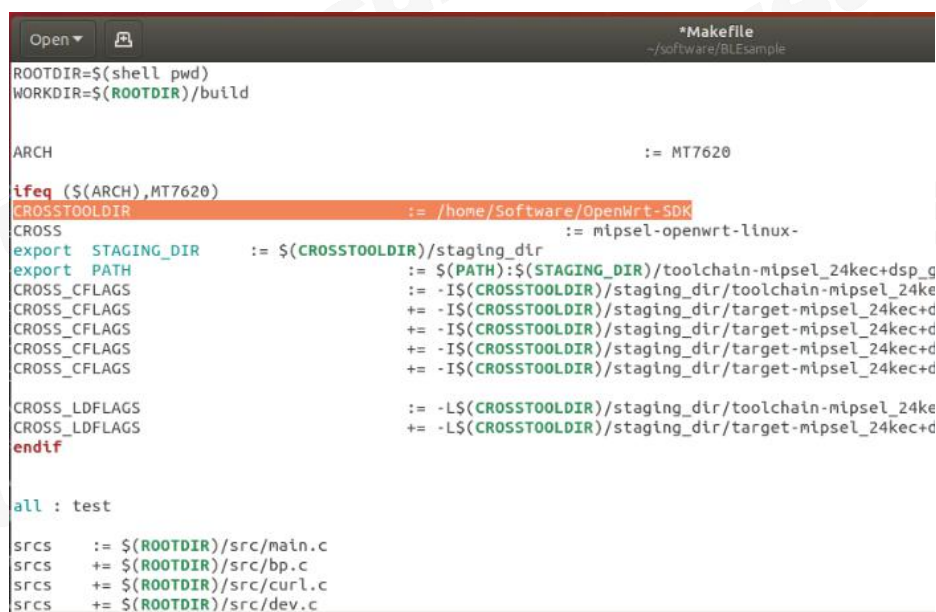
View our GitHub for the code sample:

https://github.com/dusuniot/Dusun_BLE_Code_Sample

BLE_code_sample_for_development_on_dusun_gateway

Copy the above code file to the Linux PC and decompress it to a folder (E.g.: home/software/test). Open ./BLEsample/example/ folder, and edit the Makefile file to revise the CROSSTOOLDIR to the OpenWrt toolchain directory you created above.

This is depicted in Figure 3.



```

*Makefile
~/software/BLEsample

ROOTDIR=$(shell pwd)
WORKDIR=$(ROOTDIR)/build

ARCH                               := MT7620

ifeq ($(ARCH),MT7620)
CROSSTOOLDIR                       := /home/Software/OpenWrt-SDK
CROSS                              := mipsel-openwrt-linux-
export STAGING_DIR                 := $(CROSSTOOLDIR)/staging_dir
export PATH                        := $(PATH):$(STAGING_DIR)/toolchain-mipsel_24kec+dsp_g
CROSS_CFLAGS                       := -I$(CROSSTOOLDIR)/staging_dir/toolchain-mipsel_24ke
CROSS_CFLAGS                       += -I$(CROSSTOOLDIR)/staging_dir/target-mipsel_24kec+d
CROSS_CFLAGS                       += -I$(CROSSTOOLDIR)/staging_dir/target-mipsel_24kec+d
CROSS_CFLAGS                       += -I$(CROSSTOOLDIR)/staging_dir/target-mipsel_24kec+d
CROSS_CFLAGS                       += -I$(CROSSTOOLDIR)/staging_dir/target-mipsel_24kec+d
CROSS_LDFLAGS                     := -L$(CROSSTOOLDIR)/staging_dir/toolchain-mipsel_24ke
CROSS_LDFLAGS                     += -L$(CROSSTOOLDIR)/staging_dir/target-mipsel_24kec+d
endif

all : test

srcs := $(ROOTDIR)/src/main.c
srcs += $(ROOTDIR)/src/bp.c
srcs += $(ROOTDIR)/src/curl.c
srcs += $(ROOTDIR)/src/dev.c
  
```

Figure 3 change the CROSSTOOL path

```
guojie@guojie-Inspiron-5577:~$ cd software/BLEsample/
guojie@guojie-Inspiron-5577:~/software/BLEsample$ ls
build Makefile src
guojie@guojie-Inspiron-5577:~/software/BLEsample$ sudo make
[sudo] password for guojie:
make: Warning: File '/home/guojie/software/BLEsample/src/main.c' has modified
mipsel-openwrt-linux-gcc -c /home/guojie/software/BLEsample/src/main.c -I/h
Example/src/monitor -I/home/guojie/Software/OpenWrt-SDK/staging_dir/toolcha
home/guojie/Software/OpenWrt-SDK/staging_dir/target-mipsel_24kec+dsp_uClibc
dir/target-mipsel_24kec+dsp_uClibc-0.9.33.2/usr/include/glib-2.0 -I/home/gu
bc-0.9.33.2/usr/include/dbus-1.0 -I/home/guojie/Software/OpenWrt-SDK/stagin
lude/ -MMD -MP -MF/home/guojie/software/BLEsample/build/src/main.d -o /home
mipsel-openwrt-linux-gcc -c /home/guojie/software/BLEsample/src/main.c -I/h
Example/src/monitor -I/home/guojie/Software/OpenWrt-SDK/staging_dir/toolcha
home/guojie/Software/OpenWrt-SDK/staging_dir/target-mipsel_24kec+dsp_uClibc
dir/target-mipsel_24kec+dsp_uClibc-0.9.33.2/usr/include/glib-2.0 -I/home/gu
bc-0.9.33.2/usr/include/dbus-1.0 -I/home/guojie/Software/OpenWrt-SDK/stagin
lude/ -MMD -MP -MF"/home/guojie/software/BLEsample/build/src/main.d" -o /ho
mipsel-openwrt-linux-gcc /home/guojie/software/BLEsample/build/src/main.o /
e/BLEsample/build/src/curl.o /home/guojie/software/BLEsample/build/src/dev.
ware/BLEsample/build/src/lock.o /home/guojie/software/BLEsample/build/src/p
e/software/BLEsample/build/src/wt.o /home/guojie/software/BLEsample/build/s
mainloop.o /home/guojie/software/BLEsample/build/src/gdbus/object.o /home/g
ware/BLEsample/build/src/gdbus/watch.o -ldbus-1 -ljson-c -lcurl -lssl -lcr
ing_dir/toolchain-mipsel_24kec+dsp_gcc-4.8-linaro_uClibc-0.9.33.2/usr/lib -
+dsp_uClibc-0.9.33.2/usr/lib/ -o /home/guojie/software/BLEsample/build/te
/home/guojie/Software/OpenWrt-SDK/staging_dir/target-mipsel_24kec+dsp_uClib
T 'utime' is discouraged, use 'utimes'
/home/guojie/Software/OpenWrt-SDK/staging_dir/target-mipsel_24kec+dsp_uClib
scent, use getnameinfo() instead.
make: warning: Clock skew detected. Your build may be incomplete.
guojie@guojie-Inspiron-5577:~/software/BLEsample$ cd build/
guojie@guojie-Inspiron-5577:~/software/BLEsample/build$ ls
src test
```

Figure 4 the test application compilation

Then open a terminal on the Linux PC, and type the following commands:

```
cd BLEsample; sudo make
```

Finally, the test bin file which can be run in the gateway has been compiled. (Figure 4).

- 3) Copy the compiled test bin file into the gateway and run it. There are some ways one can do it. Under Linux PC, you can use SCP command (scp local_file remote_username @remote_ip:remote_folder) to do it. Make sure the gateway is connected to the same router with PC, then run the following commands:

```
scp test root@192.168.66.1:/root
```

```
guojie@guojie-Inspiron-5577:~/software/BLEsample/build$ scp test root@192.168.66.1:/root
root@192.168.66.1's password:
test
100% 141KB 131.7KB/s 00:01
guojie@guojie-Inspiron-5577:~/software/BLEsample/build$
```



```
guojie@guojie-Inspiron-S577:~$ ssh root@192.168.66.1
root@192.168.66.1's password:
```

BusyBox v1.23.2 (2019-03-12 19:08:20 CST) built-in shell (ash)

```

  _   _   _   _   _   _   _   _   _   _   _   _   _   _   _   _
 |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |
 |___|___|___|___|___|___|___|___|___|___|___|___|___|___|___|
      W I R E L E S S       F R E E D O M

```

CHAOS CALMER (Chaos Calmer, r47202)

```
* 1 1/2 oz Gin           Shake with a glassful
* 1/4 oz Triple Sec     of broken ice and pour
* 3/4 oz Lime Juice     unstrained into a goblet.
* 1 1/2 oz Orange Juice
* 1 tsp. Grenadine Syrup
```

```
root@Dusun:~# ls
mqtt.log  test          ucmd.sh
root@Dusun:~# ./test
HL[3672]: Gateway mac: 30:AE:7B:63:F3:56
HL[3672]: add proxy: org.bluez.AgentManager1
HL[3672]: add proxy: org.bluez.ProfileManager1
HL[3672]: add proxy: org.bluez.Adapter1
HL[3672]: add proxy: org.bluez.GattManager1
HL[3672]: add proxy: org.bluez.LEAdvertisingManager1
```

Then remote login to the gateway using SSH commands or SSH client (windows: putty or SecureCrt) and run the copied bin file. The login password can be revised following the configuration steps in the above sections. Finally, we power up the oximeter device and run the test file, we can see the data the oximeter notified to the program as the figure 5 shows. The data can be analyzed according to its manual.

Figure 6/7 show the sensor and gateway for testing respectively.

```
HL[3672]: Oximeter_characteristics_notify: 81 44 61 17
HL[3672]: Notify started
HL[3672]: Oximeter_characteristics_notify: 80 09 0d 10 14 16 18 19 1a 1b 1b
HL[3672]: Oximeter_characteristics_notify: 80 1a 1a 19 18 17 16 16 17 17 18
HL[3672]: Oximeter_characteristics_notify: 80 19 1a 1a 1a 1a 19 18 17 17 18
HL[3672]: Oximeter_characteristics_notify: 80 16 15 14 13 12 11 10 0e 0d 0b
HL[3672]: Oximeter_characteristics_notify: 80 0a 08 07 08 0a 0e 13 18 1d 20
HL[3672]: Oximeter_characteristics_notify: 81 44 61 16
```

Figure 5 the printed messages when notify callback functions invoked



Figure 6 the oximeter device for testing



Figure 7 the Dusun Gateway for testing