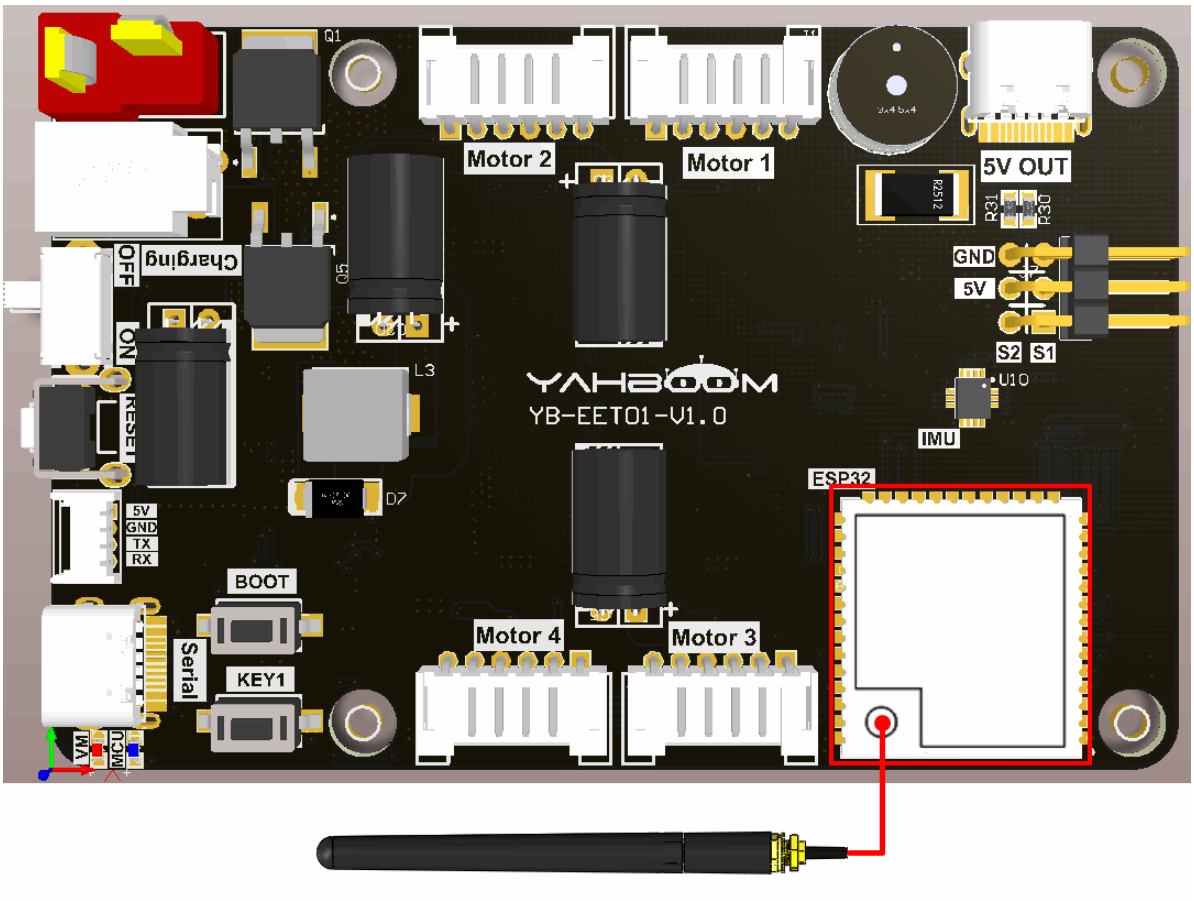
Bluetooth communication

1. Experimental purpose

Use the ESP32S3 core module of the microROS control board to learn the ESP32 Bluetooth communication function.

2. Hardware connection

As shown in the figure below, the microROS control board integrates the ESP32-S3-WROOM core module, which comes with low-power Bluetooth function. The ESP32-S3 core module needs to be connected to the antenna, and the type-C data cable needs to be connected to the computer and microROS control. The board functions as a firmware burner.

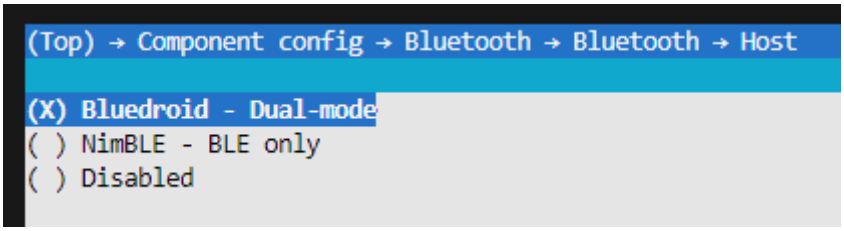


3. Core code analysis

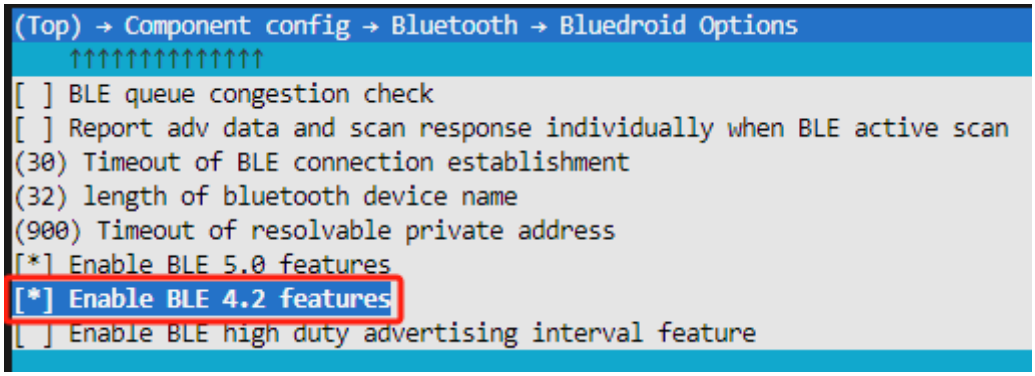
The virtual machine path corresponding to the program source code is as follows

```
~/esp/Samples/esp32_samples/bluetooch
```

First of all, the Bluetooth function is not enabled by default in ESP-IDF, so Bluetooth needs to be enabled in the configuration file.



Since only the Bluetooth BLE 5.0 feature is enabled by default, in order to pass the compilation, you need to manually enable the Bluetooth BLE 4.2 feature.



Customize the Bluetooth name. Here, the Bluetooth name is "YahBoom_BL" as an example.If you need to modify the Bluetooth name, change the SAMPLE_DEVICE_NAME macro definition to the Bluetooth name you need, then split the Bluetooth name into characters in spp_adv_data and fill it in the line below "Complete Local Name in advertising" . Among them, 0x0B represents the data length and needs to be modified according to the name length. 0x09 represents the broadcast data type and does not need to be modified. Starting from the third character are the characters of the Bluetooth name.

```
#define SAMPLE_DEVICE_NAME    "YahBoom_BL"

static const uint8_t spp_adv_data[] = {
    /* Flags */
    0x02, 0x01, 0x06,
    /* Complete List of 16-bit Service Class UUIDs */
    0x03, 0x03, 0xF0, 0xAB,
    /* Complete Local Name in advertising */
    0x0B, 0x09, 'Y', 'a', 'h', 'B', 'o', 'o', 'm', '_', 'B', 'L'
};
```

Bluetooth receiving data processing, here the received data is printed out through ESP_LOGI. If you need to add a communication protocol later, you can add it in the BLE_Parse_Rx_Data function.

```
static void BLE_Parse_Rx_Data(char *rx_data, uint16_t len)
{
    ESP_LOGI(TAG, "BLE RX: %s", rx_data);
}
```

Call the function to initialize Bluetooth in app_main, and the Bluetooth task will run in the background.

```
void app_main(void)
{
    printf("hello yahboom\n");
    ESP_LOGI(TAG, "Nice to meet you!");

    BLE_Init();
}
```

4. Compile, download and flash firmware

Use a Type-C data cable to connect the virtual machine/computer and the microROS control board. If the system pops up, choose to connect to the virtual machine.

Activate the ESP-IDF development environment. Note that every time you open a new terminal, you need to activate the ESP-IDF development environment before compiling the firmware.

```
source ~/esp/esp-idf/export.sh
```

Enter the project directory

```
cd ~/esp/Samples/esp32_samples/bluetooch
```

Compile, flash, and open the serial port simulator

```
idf.py build flash monitor
```

If you need to exit the serial port simulator, press **Ctrl+]**.

5. Experimental results

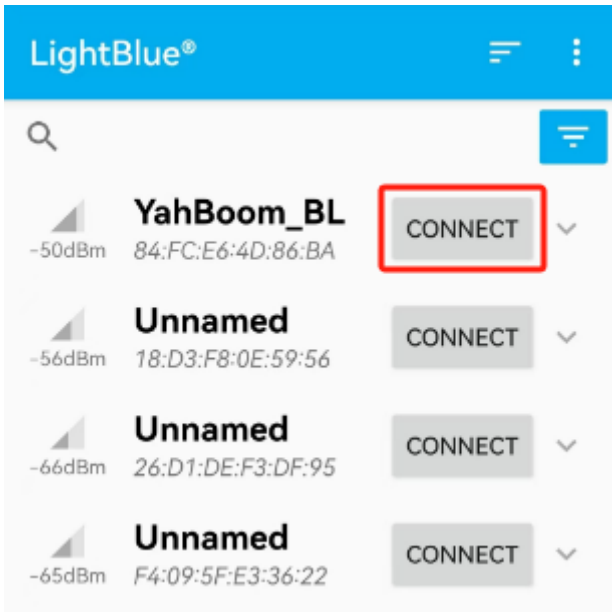
The serial port simulator prints the "hello yahboom" welcome message and prints "BLE_SERVER: BLE_Init init bluetooth" to indicate that Bluetooth BLE is started successfully.

```
I (357) coexist: coexist rom version e7ae62f
I (362) app_start: Starting scheduler on CPU0
I (367) app_start: Starting scheduler on CPU1
I (367) main_task: Started on CPU0
I (377) main_task: Calling app_main()
hello yahboom
I (382) BT_MAIN: Nice to meet you!
I (401) BLE_INIT: BT controller compile version [59725b5]
I (403) BLE_INIT: Bluetooth MAC: 84:fc:e6:4d:86:ba

I (404) phy_init: phy_version 620,ec7ec30,Sep  5 2023,13:49:13
I (451) BLE_SERVER: BLE_Init init bluetooth

I (474) main_task: Returned from app_main()
```

At this time, open the Bluetooth BLE debugging assistant and connect to Bluetooth.

Note: Since there are many Bluetooth BLE debugging assistant apps, here we take LightBlue as an example. You can download the LightBlue software from the mobile app store.

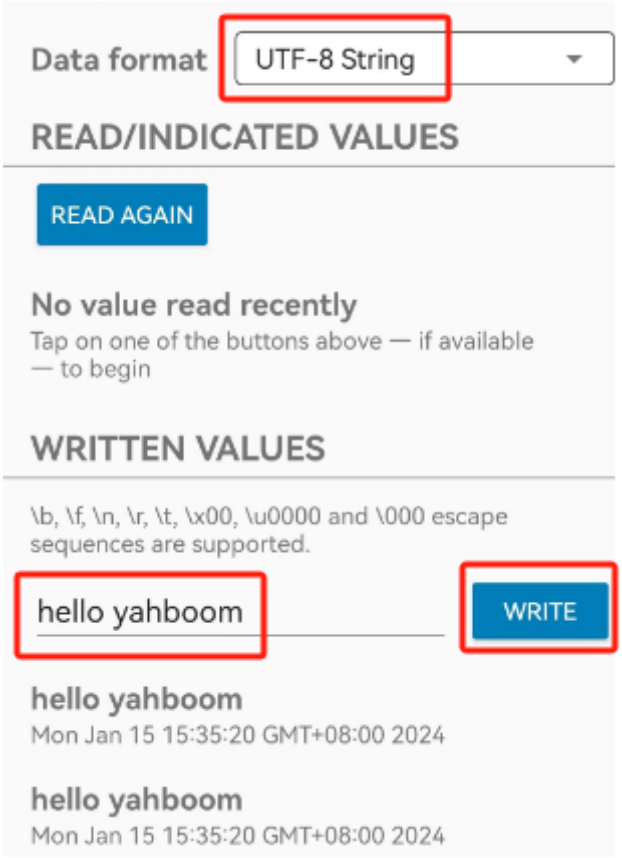
Open LightBlue and search for nearby Bluetooth devices, find the device named "Yahboom_BL", and click "CONNECT" on the right to connect to "Yahboom_BL".



After the connection is completed, the connection status and some basic information will be displayed.

Scroll down, find the service 0000ffe1, and click to enter the 0000ffe1 service.

Swipe down and find the Data format. For the convenience of display, change it to UTF-8 String format. Fill in the content to be sent in the WRITTEN VALUES column and in the dialog box below, and then click the "WRITE" button to send the data. For example, "hello yahboom" is sent twice here.



At this time, you can view the information printed by the serial port simulator, and the received "hello yahboom" has been printed.

```
I (1139396) BLE_SERVER: BLE RX: hello yahboom
I (1140446) BLE_SERVER: BLE RX: hello yahboom
```