Bluetooth remote control

The tutorial mainly demonstrates the remote control function of the balance car combined with the Bluetooth module.

```
The tutorial only introduces the standard library project code
```

Hardware connection

Bluetooth module	Development board
VCC	VCC
TXD	RXD5
RXD	TXD5
GND	GND

Control principle

The mobile phone connects to the Bluetooth module (slave) on the balance car through the built-in Bluetooth (host). The Bluetooth module receives the data of the balance car APP and sends it to the development board for analysis.

Master-slave mode

The host is responsible for managing the communication process and controlling data transmission, while the slave performs corresponding operations according to the host's instructions.

Host mode

The controller in Bluetooth communication is responsible for initiating connection requests and controlling the entire communication process.

Slave mode

Passively wait for the host's connection request and accept the host's control.

Communication protocol

The Bluetooth module sends the data transmitted by the mobile phone APP to the development board through TXD. The development board then parses the received data and obtains the corresponding command data. Later, the movement state of the car can be controlled according to this command data.

```
For detailed information, please refer to "Bluetooth Remote Control Communication Protocol"
```

Main code

The tutorial mainly explains the code for implementing some Bluetooth remote control functions. For detailed code, please refer to the corresponding project file.

> **deal_bluetooth**

Receive data sent by the Bluetooth module.

```
void deal_bluetooth(uint8_t rxbuf)
{
    u8 uartvalue = rxbuf;
    if(uartvalue == '$')
    {
      startBit = 1;
        num = 0;
    }
    if(startBit == 1)
    {
        inputString[num] = uartvalue;
    }
    if (startBit == 1 && uartvalue == '#')
    {
        newLineReceived = 1;
        startBit = 0;
        int9num = num;
    }
    num++;
    if(num >= 80)
    {
        num = 0;
        startBit = 0;
        newLineReceived = 0;
    }
}
```

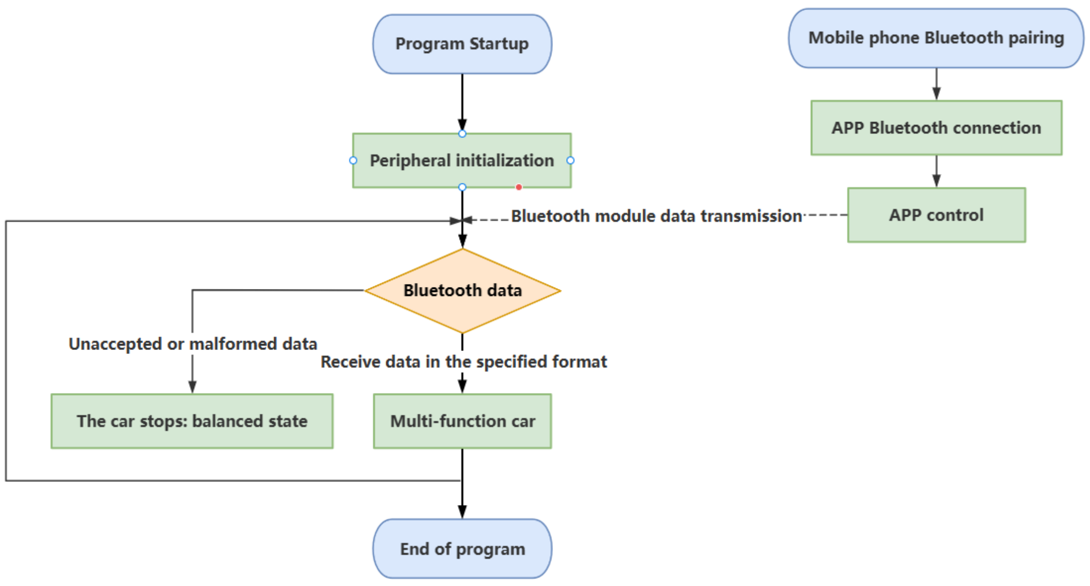
Protocol

Parse the Bluetooth module data according to the protocol.

```
void Protocol(void)
{
    switch (ProtocolString[1])
    {
        case run_car:    g_newcarstate = enRUN; break;
        case back_car:  g_newcarstate = enBACK; break;
        case left_car:  g_newcarstate = enLEFT; break;
        case right_car: g_newcarstate = enRIGHT; break;
        case stop_car:  g_newcarstate = enSTOP; break;
        default: g_newcarstate = enSTOP; break;

    }
    if (ProtocolString[3] == '1')
    {
        g_newcarstate = enTLEFT;
    }

    if (ProtocolString[3] == '2')
    {
        g_newcarstate = enTRIGHT;
    }

    if(strlen((const char *)ProtocolString)<21)
    {
        newLineReceived = 0;
        memset(ProtocolString, 0x00, sizeof(ProtocolString));
        UART5_Send_Char("$ReceivePackError#");
        return;
    }

    if(ProtocolString[5]=='1')
    {
        ResetPID();
        ProtocolGetPID();
//  }
//  else if(ProtocolString[5]=='2')
//  {
        UART5_Send_Char("$OK#");
    }

    if(ProtocolString[7]=='1')
    {
            g_autoup = 1;
            UART5_Send_Char("$OK#");
    }
    else if(ProtocolString[7]=='2')
    {
            g_autoup = 0;
            UART5_Send_Char("$OK#");
    }
    if (ProtocolString[9] == '1')
    {
        //$0,0,0,0,1,1,AP23.54,AD85.45,VP10.78,VI0.26#
        int pos,z;
        char apad[20] = {0},apvalue[8] = {0},advalue[8] = {0};
        pos = StringFind((const char *)ProtocolString, (const char *)"AP");
        if(pos == -1) return;
        memcpy(apad,ProtocolString+pos,int9num-pos);
        //AP23.54,AD85.45,VP10.78,VI0.26#
        z = StringFind(apad, ",");
        if(z == -1) return;
        memcpy(apvalue, apad+2, z-2);
        Balance_Kp = atof(apvalue)*100;
        memset(apad, 0x00, sizeof(apad));
        memcpy(apad, ProtocolString + pos + z + 1, int9num - (pos + z));
        z = StringFind(apad, ",");
        if(z == -1) return;
        memcpy(advalue,apad+2, z-2);
        Balance_Kd=atof(advalue);
        UART5_Send_Char("$OK#");
    }

    if(ProtocolString[11] == '1')
    {
        int pos,z;
        char vpvi[20] = {0},vpvalue[8] = {0},vivalue[8] = {0};
        pos = StringFind((const char *)ProtocolString, (const char *)"VP");
        if(pos == -1) return;
        memcpy(vpvi, ProtocolString+pos, int9num-pos);
        //y=strchr(apad,'AP');
        //AP23.54,AD85.45,VP10.78,VI0.26#
        z = StringFind(vpvi, ",");
        if(z == -1) return;
        memcpy(vpvalue, vpvi+2, z-2);
        Velocity_Kp = atof(vpvalue) *100;
        memset(vpvi, 0x00, sizeof(vpvi));
        memcpy(vpvi, ProtocolString + pos + z + 1, int9num - (pos + z));
        z = StringFind(vpvi, "#");
        if(z == -1) return;
        memcpy(vivalue,vpvi+2, z-2);
        Velocity_Ki=atof(vivalue);
        UART5_Send_Char("$OK#");
    }
    newLineReceived = 0;
    memset(ProtocolString, 0x00, sizeof(ProtocolString));
}
```

Program flow chart

Briefly introduce the process of function implementation:


Program Startup
Mobile phone Bluetooth pairing
Peripheral initialization
APP Bluetooth connection
Bluetooth module data transmission
APP control
Bluetooth data
Unaccepted or malformed data
Receive data in the specified format
The car stops: balanced state
Multi-function car
End of program

Experimental phenomenon

Software code

The bluetooth_control.hex file generated by the project compilation is located in the OBJ folder of the bluetooth_control project. Find the bluetooth_control.hex file corresponding to the project and use the FlyMcu software to download the program into the development board.

```
Product supporting information source code path: Attachment → Source code summary
→ 4.Balanced_Car_base → 04.bluetooth_control
```

Experimental phenomenon

After the program is started, press the KEY1 button according to the OLED prompt to start the balance car Bluetooth remote control function: OLED will display the balance car inclination angle in real time!

```
The program has voltage detection. If the voltage is less than 9.6V, a low
voltage alarm is triggered and the buzzer will sound.
Common situations that trigger voltage alarms:
1. The power switch of the development board is not turned on, and only the Type-
C data cable is connected for power supply
2. The battery pack voltage is lower than 9.6V and needs to be charged in time
```

Bluetooth remote control

Download Bluetooth remote control APP

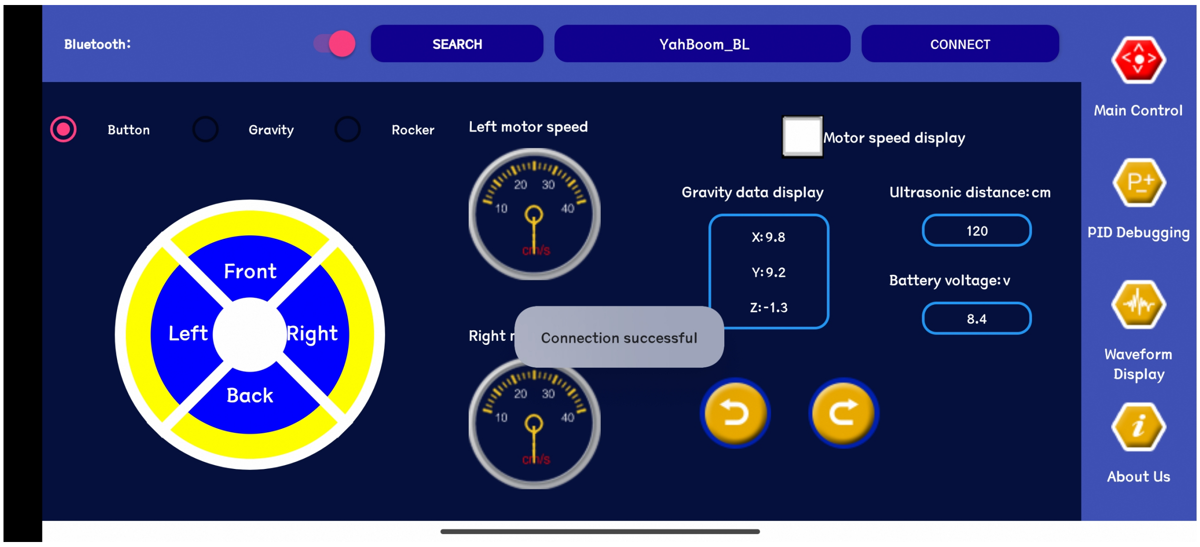
Use the mobile browser to scan the QR code to download the APP (only supports Android)

Bluetooth connection

When you open the BalanceBot APP for the first time, it will automatically search and connect to nearby Bluetooth devices. We need to bring the phone close to the car, and the BalanceBot APP will automatically connect; if the connection is successful, a "Connection successful" prompt will appear.

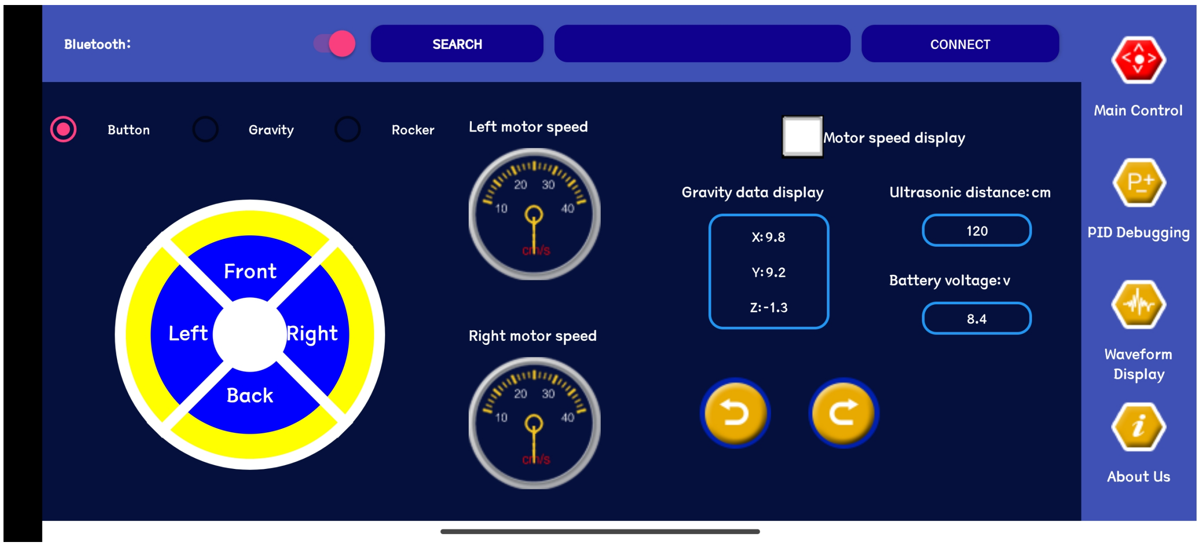
If no search is found, you can click the Bluetooth search and connection options in turn.

```
The mobile phone needs to turn on the Bluetooth function and allow the Bluetooth remote control APP related permissions
```


Connection successful

Main control interface

Used for basic control and data display of the balance car.



Direction control

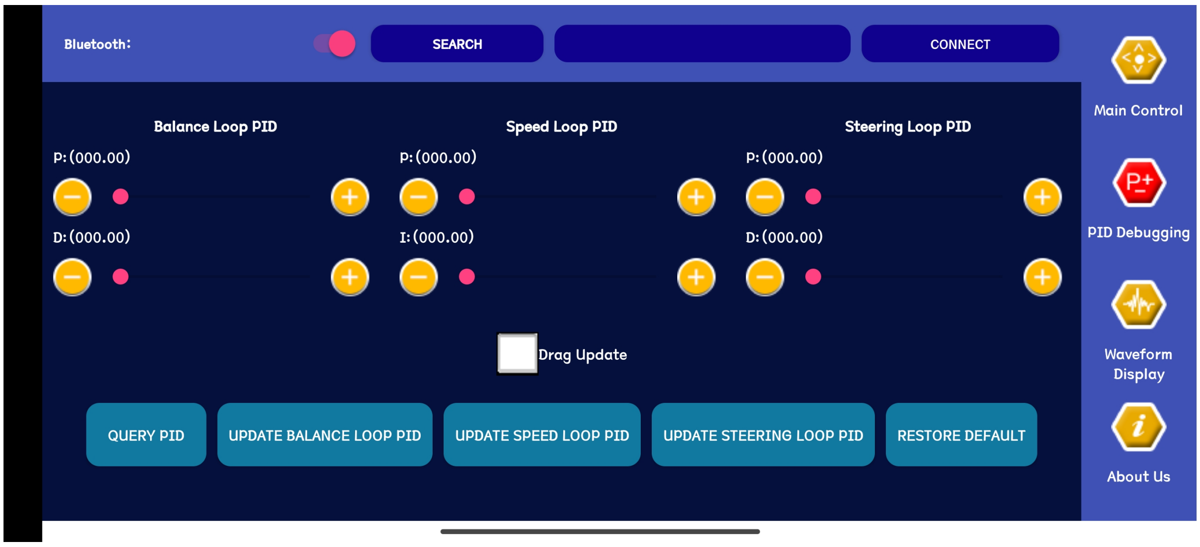
The main control interface can control the movement of the balance car through buttons, gravity and joystick.

Data display

The main control interface will automatically obtain gravity, ultrasonic distance, battery voltage, and left and right motor speed display.

PID debugging

Used for online query and adjustment of PID parameters to achieve the best effect.



Vertical ring

Adjust the vertical ring PD parameters, support dragging and keystrokes to update data.

Speed ring

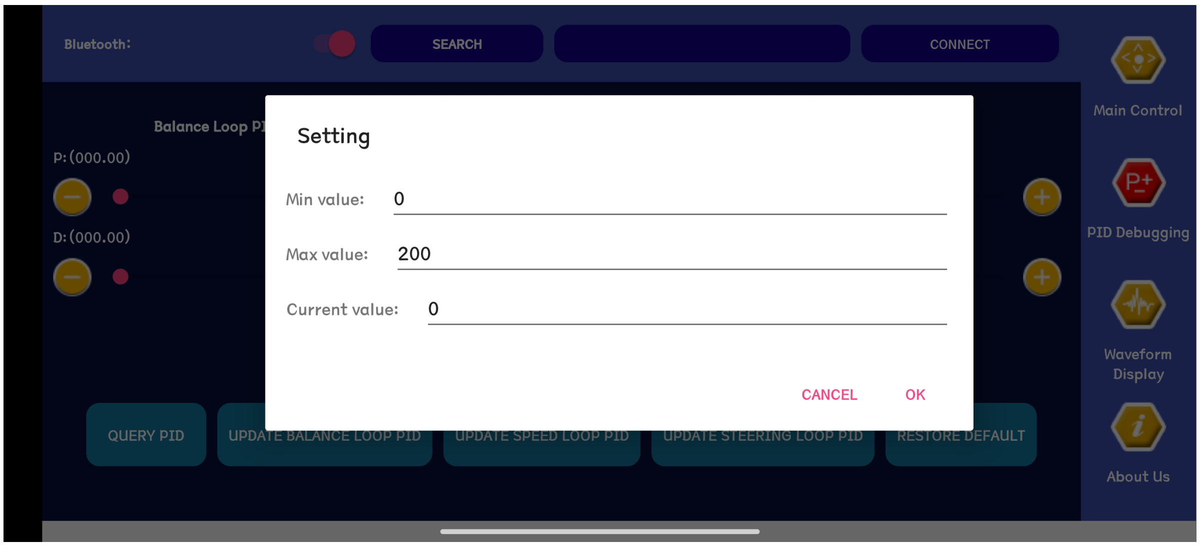
Adjust the speed ring PI parameters, support dragging and keystrokes to update data.

Steering ring

Adjust the steering ring PD parameters, support dragging and keystrokes to update data.

PID parameter range

Click the P, I, D options on the interface to adjust the adjustment range of each PID parameter.


Bluetooth:

SEARCH

CONNECT

Balance Loop PI

P:(000.00)

D:(000.00)

Setting

Min value: 0

Max value: 200

Current value: 0

CANCEL OK

QUERY PID

UPDATE BALANCE LOOP PID

UPDATE SPEED LOOP PID

UPDATE STEERING LOOP PID

RESTORE DEFAULT

Main Control

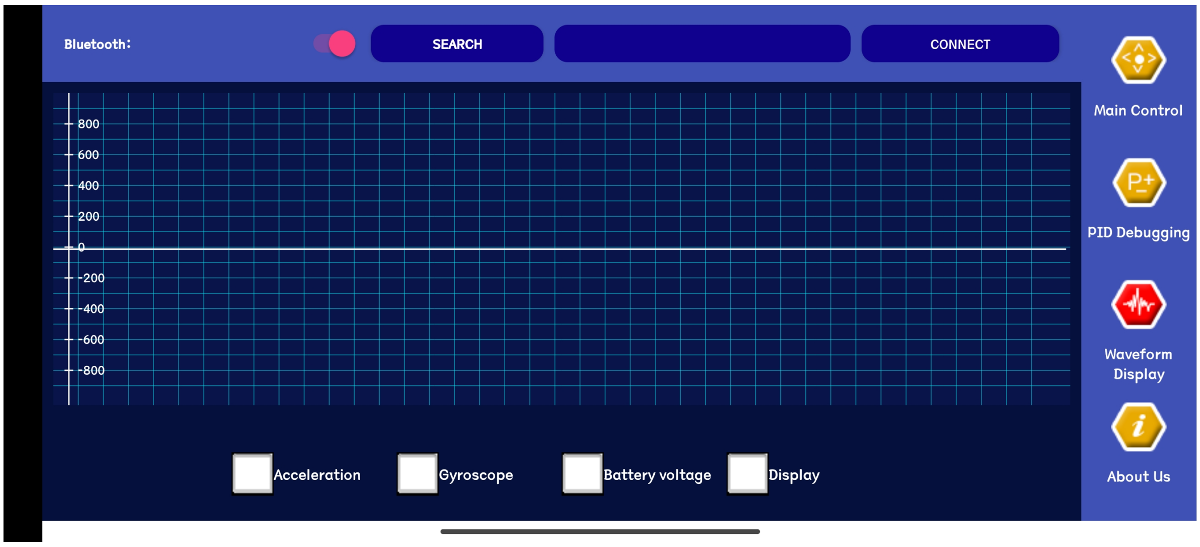
PID Debugging

Waveform Display

About Us

Waveform display

Check the corresponding option to display the dynamic changes of acceleration, gyroscope and battery voltage data.


Bluetooth:
SEARCH
CONNECT
800
600
400
200
0
-200
-400
-600
-800
Acceleration
Gyroscope
Battery voltage
Display
Main Control
PID Debugging
Waveform Display
About Us