Bluetooth remote control

1. Opening instructions

Please read the "Introduction to Motors and Usage" in the four-way motor driver board information first to understand the motor parameters, wiring methods, and power supply voltage you are currently using. To avoid burning the motherboard or motor.

Motor: The case and code take the 310 motor of our store as an example.

2. Experimental preparation

National Race chassis V2 four-wheel drive version, 4*310 motor, 7.4V lithium battery, Bluetooth 5.0 module, STM32F103C8T6 core board.

The relationship between the 4 motor interfaces and the car is as follows:

M1 -> upper left motor (left front wheel of the car)

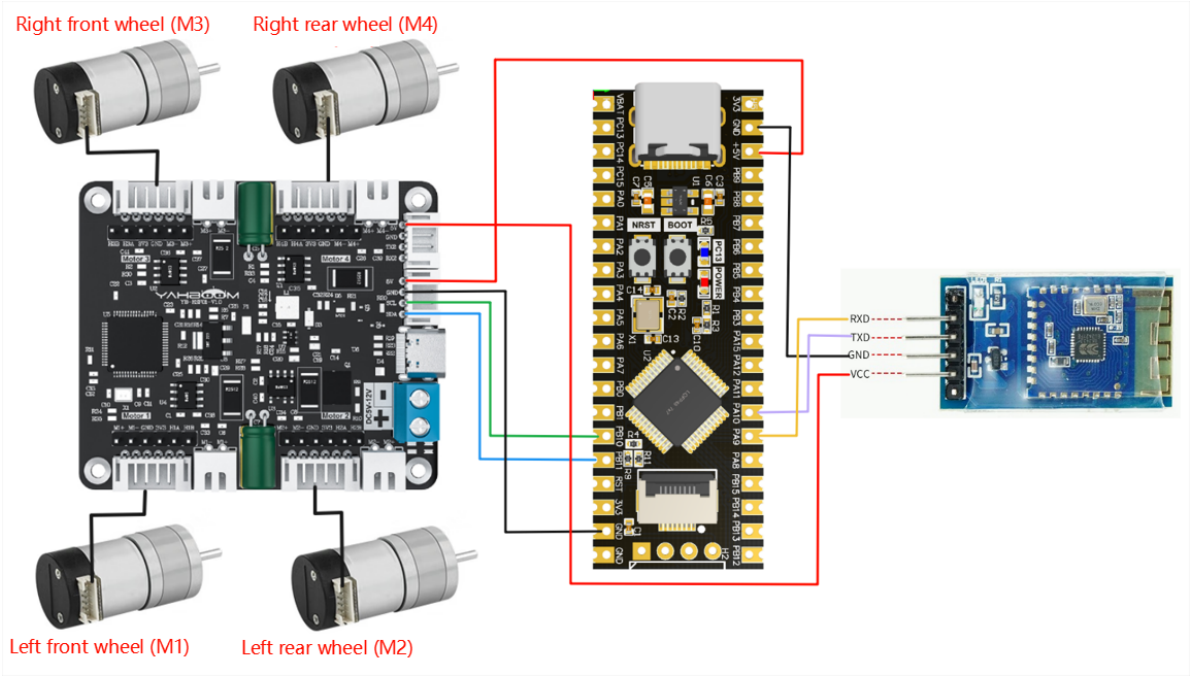
M2 -> lower left motor (left rear wheel of the car)

M3 -> upper right motor (right front wheel of the car)

M4 -> lower right motor (right rear wheel of the car)

Hardware wiring:

Overall wiring



Wiring pins

Four-way motor driver board	STM32C8T6
5V	5V
GND	GND
SCL	PB10
SDA	PB11

Take M1 motor as an example below, and other motors are similar.

Motor	Four-way motor driver board (Motor)
M2	M1-
VCC	3V3
A	H1A
B	H1B
GND	GND
M1	M1+

Bluetooth module	STM32C8T6
VCC	3V3
GND	GND
RXD	PA9
TXD	PA10

3. Key code analysis

- bsp_motor_iic.c

```
//配置电机  Configure the motor
void Set_motor_type(uint8_t data)
{
    i2cWrite(Motor_model_ADDR,MOTOR_TYPE_REG,2,&data);
}

//配置死区  Configuring Dead Zone
void Set_motor_deadzone(uint16_t data)
{
    static uint8_t buf_tempzone[2];

    buf_tempzone[0] = (data>>8)&0xff;
    buf_tempzone[1] = data;

    i2cWrite(Motor_model_ADDR,MOTOR_DeadZONE_REG,2,buf_tempzone);
}

//配置磁环线 Configuring magnetic loop
void Set_Pluse_line(uint16_t data)
{
    static uint8_t buf_templine[2];

    buf_templine[0] = (data>>8)&0xff;
    buf_templine[1] = data;

    i2cWrite(Motor_model_ADDR,MOTOR_PluseLine_REG,2,buf_templine);
}

//配置减速比 Configure the reduction ratio
void Set_Pluse_Phase(uint16_t data)
{
    static uint8_t buf_tempPhase[2];

    buf_tempPhase[0] = (data>>8)&0xff;
    buf_tempPhase[1] = data;

    i2cWrite(Motor_model_ADDR,MOTOR_PlusePhase_REG,2,buf_tempPhase);
}


//配置直径  Configuration Diameter
void Set_Wheel_dis(float data)
{
    static uint8_t bytes[4];

    float_to_bytes(data,bytes);

    i2cWrite(Motor_model_ADDR,WHEEL_DIA_REG,4,bytes);
}

...

//控制带编码器类型的电机   Control the motor with encoder type
```

```
//传入参数:4个电机的pwm PWM of 4 motors
//此函数可以结合实时编码器的数据，来实现control_speed的功能   This function can combine the data of real-time encoder to realize the function of control_speed
void control_pwm(int16_t m1,int16_t m2 ,int16_t m3,int16_t m4)
{
    static uint8_t pwm[8];

    pwm[0] = (m1>>8)&0xff;
    pwm[1] = (m1)&0xff;

    pwm[2] = (m2>>8)&0xff;
    pwm[3] = (m2)&0xff;

    pwm[4] = (m3>>8)&0xff;
    pwm[5] = (m3)&0xff;

    pwm[6] = (m4>>8)&0xff;
    pwm[7] = (m4)&0xff;

    i2cWrite(Motor_model_ADDR,PWM_Control_REG,8,pwm);

}
```

Define functions for writing configuration parameters to the four-way motor driver board and motor control functions to set key parameters such as motor type, dead zone, number of magnetic ring lines, reduction ratio, and wheel diameter, and control four motors. Send PWM signals to the motor driver board through i2c communication to adjust the speed and direction of the motor.

- Motor.c

```
void Forward(void)
{
    control_pwm(1500,1500,1500,1500);
}
void Backward(void)
{
    control_pwm(-1500,-1500,-1500,-1500);
}
void Turnleft(void)
{
    control_pwm(0,0,2000,2000);
}
void Turnright(void)
{
    control_pwm(2000,2000,0,0);
}
void Stop(void)
{
    control_pwm(0,0,0,0);
}
void SpinLeft(void)
{
    control_pwm(-1500,-1500,1500,1500);
}
void SpinRight(void)
{
```

```
    control_pwm(1500,1500,-1500,-1500);
}

void Data_Analyse(void)//解析串口中断串口接收的数据
{
        if(rxd_flag == 1)
        {
            switch(rxd_buf[0])
            {
                case '1':
                    printf("Forward!\n");
                    Car_state = 1;
                    break;
                case '2':
                    printf("Backward!\n");
                    Car_state = 2;
                    break;
                case '3':
                    printf("Left!\n");
                    Car_state = 3;
                    break;
                case '4':
                    printf("Right!\n");
                    Car_state = 4;
                    break;
                case '0':
                    printf("Stop!\n");
                    Car_state = 0;
                    break;
            }
            switch(rxd_buf[2])
            {
                case '1':
                    printf("SpinLeft!\n");
                    Car_state = 5;
                    break;
                case '2':
                    printf("SpinRight!\n");
                    Car_state = 6;
                    break;
            }
            rxd_flag = 0;
        }
}

void Car_Function(unsigned int Car_state)//控制小车不同状态
{
    switch(Car_state)
    {
            case 0:
                    printf("Stop\n");
                    Stop();
                    break;
            case 1:
                    printf("Forward\n");
                    Forward();
                    break;
            case 2:
```

```
                    printf("Backward\n");
                    Backward();
                    break;
            case 3:
                    printf("Turnleft\n");
                    Turnleft();
                    break;
            case 4:
                    printf("Turnright\n");
                    Turnright();
                    break;
            case 5:
                    printf("SpinLeft\n");
                    SpinLeft();
                    break;
            case 6:
                    printf("SpinRight\n");
                    SpinRight();
                    break;
    }
}
```

Several motion control functions are defined. `control_pwm()` controls four motors to realize the car's forward, backward, turn, stop and rotate on the spot. `Data_Analyse()` parses serial port instructions and updates `Car_state`. `Car_Function()` calls the corresponding motion function according to `Car_state`, so that the car performs the corresponding action according to the received instructions.

- main.c

```
#include "stm32f10x.h"
#include "SysTick.h"
#include "Uart.h"
#include "bsp_motor_iic.h"
#include "Motor.h"

#define MOTOR_TYPE 2   //1:520电机 2:310电机 3:测速码盘TT电机 4:TT直流减速电机 5:L型520电机
                       //1:520 motor 2:310 motor 3:speed code disc TT motor 4:TT DC reduction motor 5:L type 520 motor

int main(void)
{
    delay_init();       //配置定时器Configure the timer
    Uart1_Configuration();          //配置UART1串口Configure UART1 serial port
    Uart1_NVIC_Configuration(); //配置UART1串口中断Configure UART1 serial port interrupt

    IIC_Motor_Init();

    #if MOTOR_TYPE == 1
    ...

    #elif MOTOR_TYPE == 2
    Set_motor_type(2);//配置电机类型  Configure motor type
    delay_ms(100);
```

```
    Set_Pluse_Phase(20);//配置减速比 查电机手册得出 Configure the reduction ratio.
Check the motor manual to find out
    delay_ms(100);
    Set_Pluse_line(13);//配置磁环线 查电机手册得出  Configure the magnetic ring wire.
Check the motor manual to get the result.
    delay_ms(100);
    Set_Wheel_dis(48.00);//配置轮子直径,测量得出      Configure the wheel diameter
and measure it
    delay_ms(100);
    Set_motor_deadzone(1300);//配置电机死区,实验得出  Configure the motor dead zone,
and the experiment shows
    delay_ms(100);

    ...

    #endif

    while(1)
    {
            Data_Analyse();//解析串口中断接收的数据    Analyze data received by the
serial port interrupt
            Car_Function(Car_state);//控制小车不同状态  Control different states of
the car
    }
}
```

`MOTOR_TYPE`: used to set the type of motor used. Modify the corresponding number according to the comments based on the motor you are currently using.

Initialize the timer, UART serial port and IIC motor driver module, and use `Set_motor_type()` and other functions to set the motor type and parameters according to `MOTOR_TYPE`. In the main loop, call `Data_Analyse()` to continuously parse the data received by the serial port, and call `Car_Function()` to control the movement state of the car according to the analysis results, such as forward, backward, turn or stop.

4. Experimental phenomenon

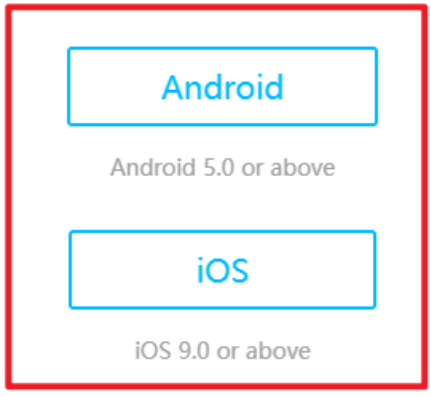
After connecting the car, burning the program to STM32, put the car on the ground and plug in the power supply. Open the app, select the 4WD model, and after connecting Bluetooth, you can control the car to move forward and backward, turn left and right, and rotate left and right through the app.

APP download: http://www.yahboom.net/software/app

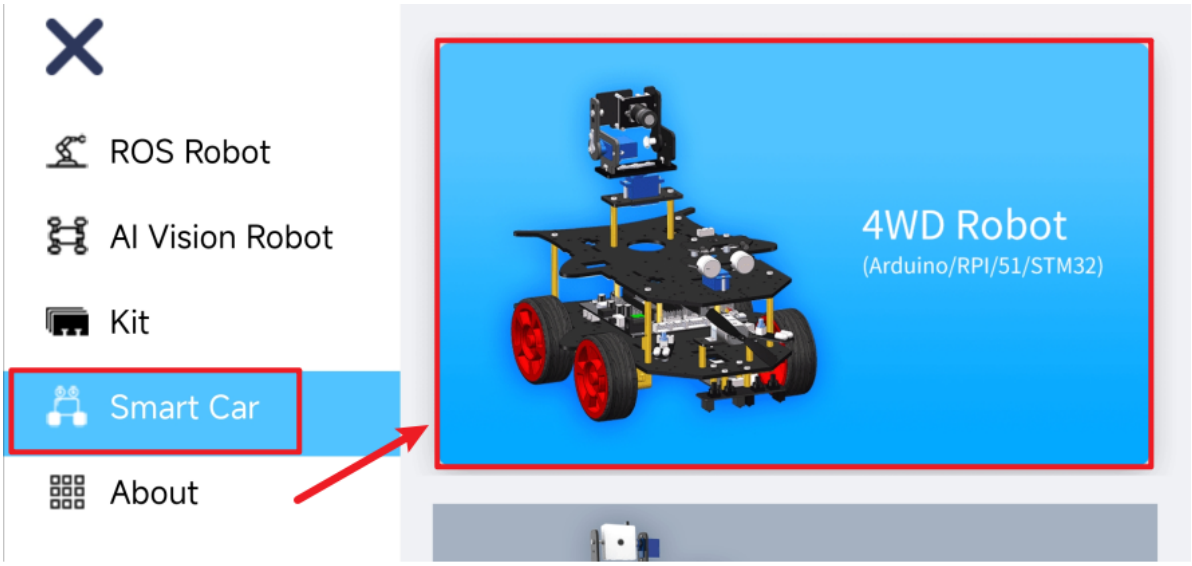
Specific operation steps of YahboomRobot APP

① Download the corresponding App

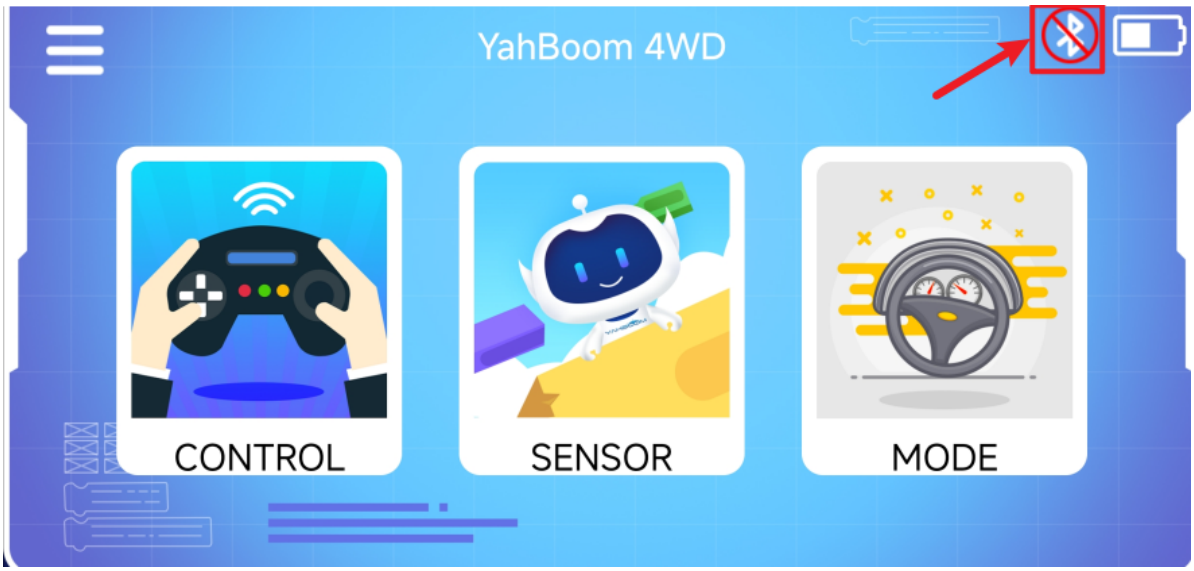
Android users scan the QR code by browser to download APP.
iOS users scan the QR code by browser or camera to download APP. Or search "YahBoomRobot" in App Store to download APP.



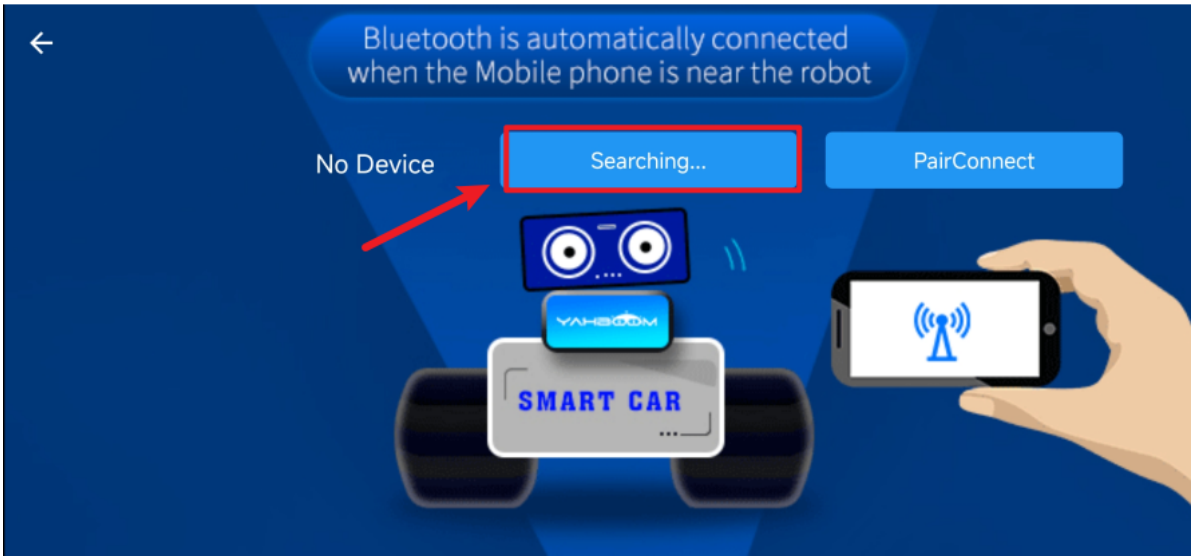
② Open the App and select 4WD Robot



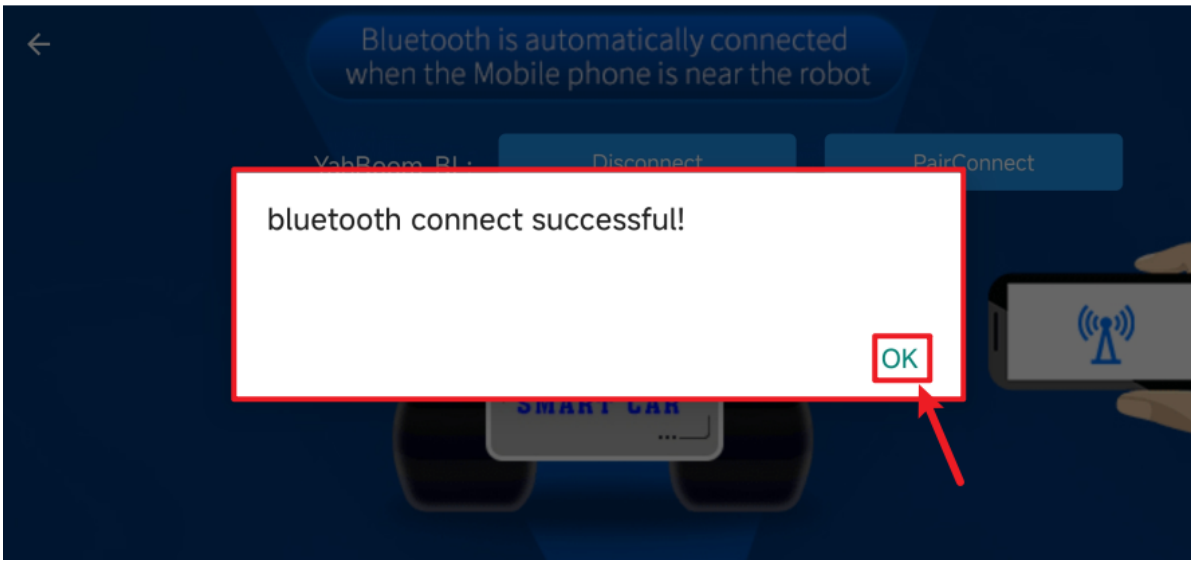
③ Open the Bluetooth interface and enable location permissions, etc.



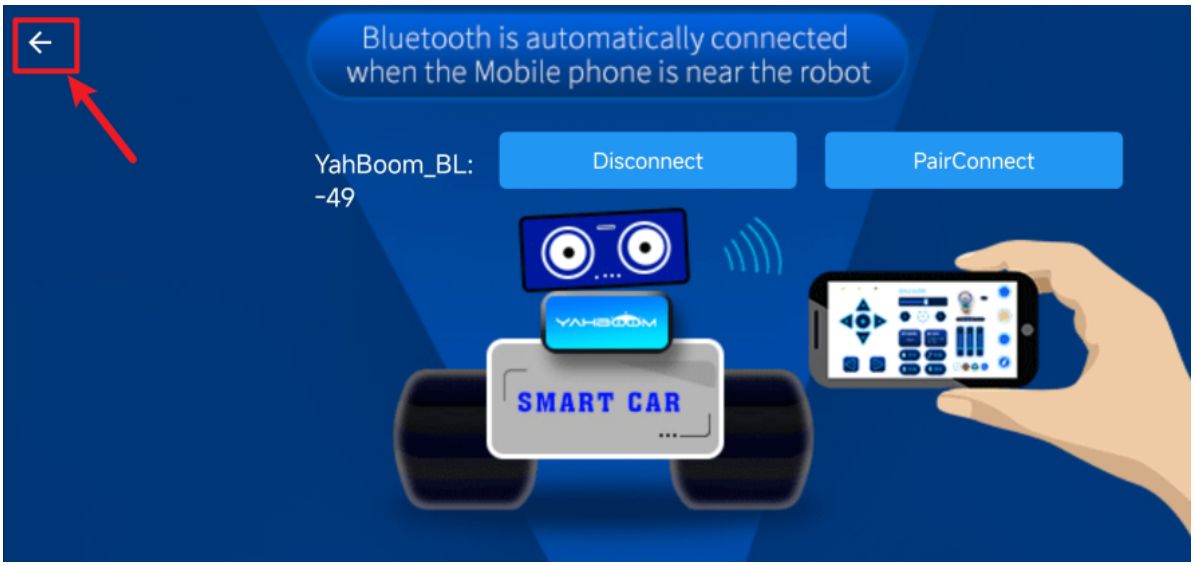
④Search for Bluetooth



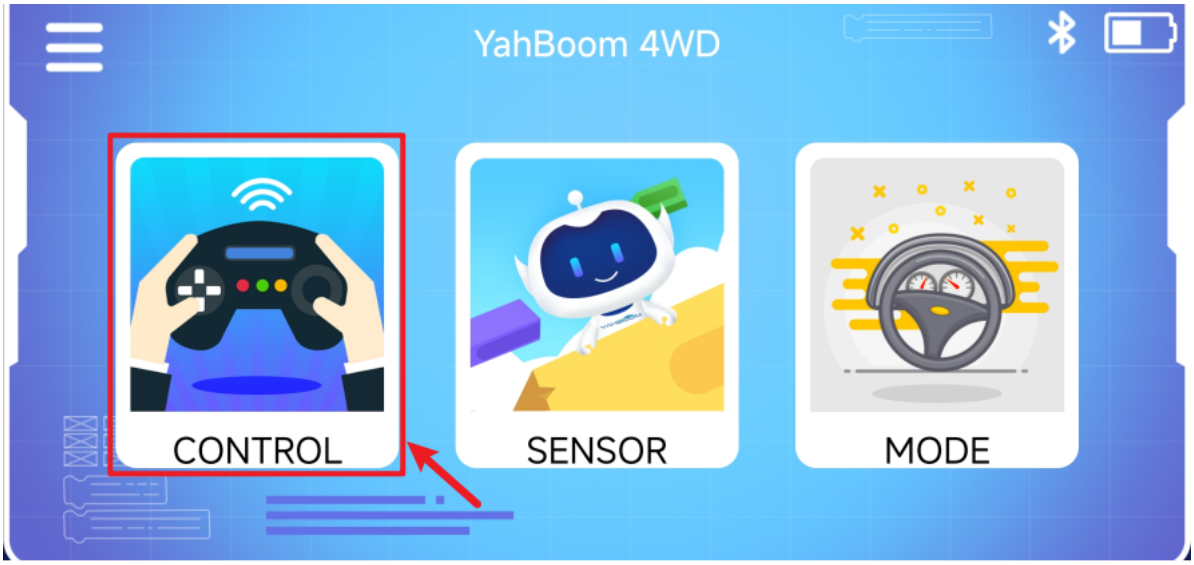
⑤Connect Bluetooth



⑥Return to function selection page



⑦Control interface



⑧Areas supporting function control