Product parameter list

Item	Index
Size	length: 175mm, width: 170mm, high: 145mm
Net weight	≤3KG (including battery pack)
Battery	18650 power 4 series, rated voltage: 14.8V, capacity: 2600mAH
Communication interface	2.4GHz, Bluetooth 4.0, USB
Integrated linear speed	3.6m/s(max)
Angular velocity	500rad/s
Shoot	velocity: 8.5m/s(max)
Chip	height: 650mm、distance: 2000mm
Chassis motor	speed:3000r/min、power: 30W
Ball control motor	speed: 20000r/min、power: 12W
Total power	150W

Introduction of equipment components

This section describes the various components of the robot. Before the operation of the robot is recommended, please be familiar with the various components.

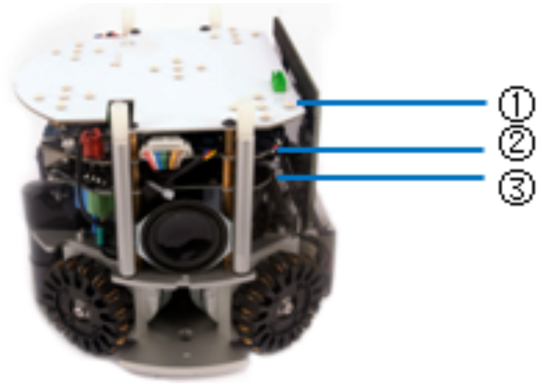
Schematic diagram of the front and back

Figure 1-1 robot closes the front and back sides

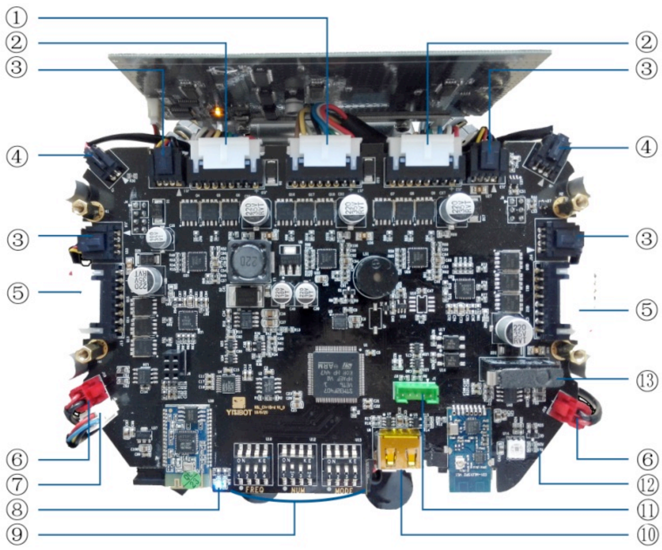
1.Color card
2.LED
3.Shell
4. The Universal wheel
5. The-IR
6. Charging interface
7. Power switch

Robot in the shell

Turn off the power before removing the robot and pull out the battery



1.LED 2. Control 3.Power



■①Ball control motor interface

■②Chassis motor interface

■③encode interface

■④Infrared sensor interface

■⑤Chassis motor interface

■⑥Speaker interface

■⑦LED interface

■⑧Bluetooth Indicator light：The indicator light alternate flashes on behalf of Bluetooth to start working, The right indicator light flashes on behalf of Bluetooth connection success

■⑨Mode setting button

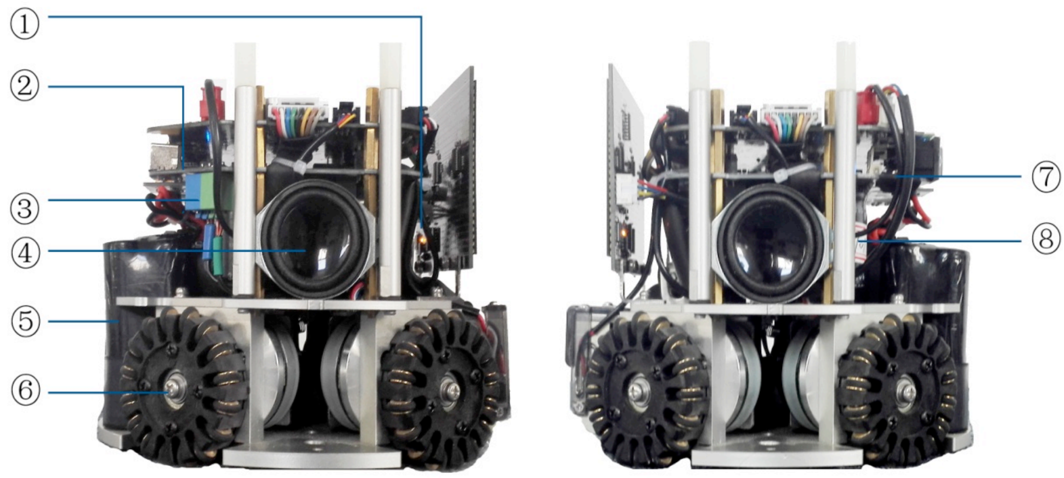
■⑩USB interface

■⑪Download Interface

■⑫RGB LED

■⑬Boost switch

Left and right views



■①LED indicator light： LED indicator light when used in LED
■②Charging indicator light
■③chip and shoot coil interface
■④Bluetooth Speaker
■⑤Battery
■⑥Universal wheel
■⑦On-off indicator: the on-off indicator lights when the power is turned on
■⑧Capacitance

Preparation before use

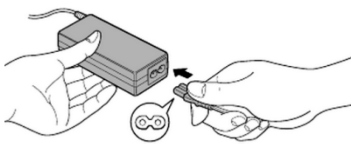
Preparation steps and basic operation

- To charge the battery
- Set the working mode of the robot.
- Expanding function introduction

To charge the battery

Connect the YISI power adapter to charge the battery.

1. Connect the power cord to the power adapter



2. Plug in the output plug of the power adapter into the charging interface. Charging lamp light red.

3. Charging indicator lights red light into green when it is full of electricity.

Charging interface

* Do not use when charging, When the speed of the robot is obviously decreased or the buzzer continues to ring, the battery should be replaced immediately.

Set working mode

Set frequency point

frequency encoder（FERO）：The robot with the same team should be set at the same frequency.。The participating parties in the frequency (4) toggle settings。

Figure 2-3 schematic diagram of the robot frequency point

Set robot number

robot number encoder（NUM）: The setting of robot number is related to binary.， Dial up is 1， On the contrary is 0，So the numbers of right were 1000 (8), 0100 (4), 0010 (2), 0001 (1)。

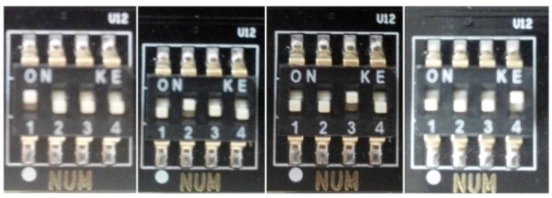


Figure 2-4 schematic diagram of robot number

Set operation mode

mode encoder（MODE）: Confirm the operation mode of the robot,
The right is rhe self checking mode.
The middle is the handle mode.
The left is the game mode.

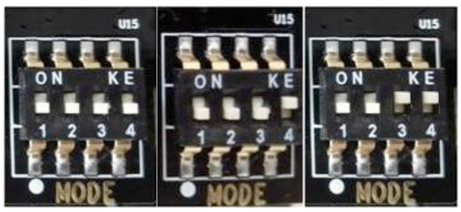


Figure 2-5 Schematic diagram of the robot operation mode

Handle control

In the robot off state, dial "MODE" to "0001", Insert USB into the Key interface on the robot control panel, plug in the battery, and then open the robot, Then we can control the robot by the handle. The function of each button is illustrated in the figure below.

Figure 2-6 Handle button instructions

Expanding function introduction

Bluetooth Speaker

Figure 2-7 Schematic diagram of Bluetooth

1. Turn on the robot, the Bluetooth indicator light
2. Open the phone (or notebook) to be matched.
3. Turn on the phone's Bluetooth switch, use Bluetooth search Bluetooth speaker
4. Robot's Bluetooth number is located on the chassis of the robot.，Bluetooth number is ID after 4 (Figure 2-8)

5、 After docking success by using the phone to play voice, music and other audio files.

Figure 2-8 Robot’s Bluetooth number

communication protocol

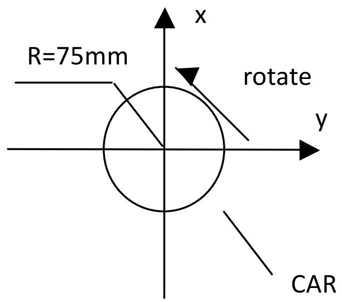
hardware interface： UART

baud rate： 115200

stop bit： 1

Parity checksum： NONE

Vehicle speed command:



1. Structure of Frame Package for competition (Normal):

bit byte	7	6	5	4	3	2	1	0
0 (Header)	1	1	1	1	1	1	1	1
1 (Func)	0				Robot11	Robot10	Robot9	Robot8
2 Robots ID[2]	Robot7	Robot6	Robot5	Robot4	Robot3	Robot2	Robot1	Robot0
3 (misc)	dribble ball forward (0) dribble ball backward (1)	shoot(0) chip(1)	dribble level (0-3)					
4 (velocity_x)	+ x (0) - x (1)	the absolute value of x velocity (lower 7bits)， unit of measurement： 1cm/s						
5 (velocity_y)	+ y (0) - y (1)	the absolute value of y velocity (lower 7bits)， unit of measurement： 1cm/s						
6 (velocity_rotate)	+ w (0) - w (1)	The absolute value of the axial speed (lower 7bits)， unit of measurement： 1/40 rad/s, clockwise is the positive direction						
7 (misc)	dribble ball forward (0) dribble ball backward (1)	shoot(0) chip(1)	dribble level (0-3)					
8 (velocity_x)	+ x (0) - x (1)	the absolute value of x velocity (lower 7bits)， unit of measurement： 1cm/s						

9 (velocity_y)	+ y (0) - y (1)	the absolute value of y velocity (lower 7bits)，unit of measurement：1cm/s						
10 (velocity_rotate)	+ w (0) - w (1)	The absolute value of the axial speed (lower 7bits)，unit of measurement：1/40 rad/s, clockwise is the positive direction						
11 (misc)	dribble ball forward (0) dribble ball backward (1)	shoot(0) chip(1)	dribble level (0-3)					
12 (velocity_x)	+ x (0) - x (1)	the absolute value of x velocity (lower 7bits)，unit of measurement：1cm/s						
13 (velocity_y)	+ y (0) - y (1)	the absolute value of y velocity (lower 7bits)，unit of measurement：1cm/s						
14 (velocity_rotate)	+ w (0) - w (1)	The absolute value of the axial speed (lower 7bits)，unit of measurement：1/40 rad/s, clockwise is the positive direction						
15 (higher 2bits)	Robot0’_Vx8	Robot0’_Vx7	Robot0’_Vy8	Robot0’_Vy7	Robot0’_Vr8	Robot0’_Vr7		
16 (higher 2bits)	Robot1’_Vx8	Robot1’_Vx7	Robot1’_Vy8	Robot1’_Vy7	Robot1’_Vr8	Robot1’_Vr7		
17 (higher 2bits)	Robot2’_Vx8	Robot2’_Vx7	Robot2’_Vy8	Robot2’_Vy7	Robot2’_Vr8	Robot2’_Vr7		
18 Shoot power 1		Robot0’_shootpower						
19 Shoot power 2		Robot1’_shootpower						
20 Shoot power 3		Robot2’_shootpower						

The byte 0 is 0xff

```
transmitPacket[0] = 0xff;
```

The byte 1 and byte 2 means which robot should process this package. For example, if Robot10, Robot7, Robot2 are set to 1, and others are set to 0, it means this package content the command for Robot10, Robot7 and Robot2. (in above table is Robot0', Robot1', Robot2')

3~6 bytes is the command for the Robot10, 7~10 bytes is the command for the Robot7, and 11~14 bytes is the command for Robot2. Each Robot should check its corresponding bit whether equal to 1, and find the sequence of the command

```
// set RobotID
int robotID = 11
if (robotID > 7) {
	transmitPacket[1] = (1 << (robotID - 8)) | 0x00;
	transmitPacket[2] = 0x00;
}
else {
	transmitPacket[1] = 0x00;
	transmitPacket[2] = 1 << (robotID);
}

// first Robot (this example is Robot11)
int shootMode = 1
int dribble = 1
int dribble_level = 3
transmitPacket[3] = (shootMode << 6);
transmitPacket[3] = transmitPacket[3] | (dribble? (dribble_level << 4):0);

// second Robot (this example is Robot7)
int shootMode = 1
```

```
int dribble = 1
int dribble_level = 3
transmitPacket[7] = (shootMode << 6);
transmitPacket[7] = transmitPacket[7] | (dribble? (dribble_level << 4):0);


// first Robot (this example is Robot2)
int shootMode = 1
int dribble = 1
int dribble_level = 3
transmitPacket[11] = (shootMode << 6);
transmitPacket[11] = transmitPacket[11] | (dribble? (dribble_level << 4):0);
```

byte 4 is the lower 7bits of velocity_x of first Robot(this example is Robot11)
byte 5 is the lower 7bits of velocity_y of first Robot(this example is Robot11)
byte 6 is the lower 7bits of velocity_rotate of first Robot(this example is Robot11)

```
transmitPacket[4] = ((velX >= 0)?0:0x80) | (abs(velX) & 0x7f);
transmitPacket[5] = ((velY >= 0)?0:0x80) | (abs(velY) & 0x7f);
transmitPacket[6] = ((velR >= 0)?0:0x80) | (abs(velR) & 0x7f);
if(transmitPacket[4] == char(0xff)) transmitPacket[4] = 0xfe;
if(transmitPacket[5] == char(0xff)) transmitPacket[5] = 0xfe;
if(transmitPacket[6] == char(0xff)) transmitPacket[6] = 0xfe;
```

byte 8 is the lower 7bits of velocity_x of second Robot(this example is Robot7)
byte 9 is the lower 7bits of velocity_y of second Robot(this example is Robot7)

byte 10 is the lower 7bits of velocity_rotate of second Robot(this example is Robot7)

```
transmitPacket[8] = ((velX >= 0)?0:0x80) | (abs(velX) & 0x7f);
transmitPacket[9] = ((velY >= 0)?0:0x80) | (abs(velY) & 0x7f);
transmitPacket[10] = ((velR >= 0)?0:0x80) | (abs(velR) & 0x7f);
if(transmitPacket[8] == char(0xff)) transmitPacket[8] = 0xfe;
if(transmitPacket[9] == char(0xff)) transmitPacket[9] = 0xfe;
if(transmitPacket[10] == char(0xff)) transmitPacket[10] = 0xfe;
```

byte 12 is the lower 7bits of velocity_x of second Robot(this example is Robot2)
byte 13 is the lower 7bits of velocity_y of second Robot(this example is Robot2)
byte 14 is the lower 7bits of velocity_rotate of second Robot(this example is Robot2)

```
transmitPacket[12] = ((velX >= 0)?0:0x80) | (abs(velX) & 0x7f);
transmitPacket[13] = ((velY >= 0)?0:0x80) | (abs(velY) & 0x7f);
transmitPacket[14] = ((velR >= 0)?0:0x80) | (abs(velR) & 0x7f);
if(transmitPacket[12] == char(0xff)) transmitPacket[12] = 0xfe;
if(transmitPacket[13] == char(0xff)) transmitPacket[13] = 0xfe;
if(transmitPacket[14] == char(0xff)) transmitPacket[14] = 0xfe;
```

byte 15 to byte 17 contain the higher 2bits of each command
for first Robot(this example is Robot11)

```
transmitPacket[15] = ((abs(velX) & 0x180) >> 1) | ((abs(velY) & 0x180) >> 3) | ((abs(velR) & 0x180) >> 5);
```

so if Robot11's velX is 320cm/s, then velX = 1 0100 0000 (9 bits), the lower 7bits is in transmitPacket[12], the higher 2bits is in transmitPacket[15]'s first two bits

byte 18 is the first Robot(this example is Robot11)'s shoot power, only lower 7 bits take effects.
transmitPacket[18] = (shoot ? shootPowerLevel:0) & 0x7f;

byte 19 is the second Robot(this example is Robot7)'s shoot power, only lower 7 bits take effects.
byte 20 is the first Robot(this example is Robot11)'s shoot power, only lower 7 bits take effects.

Byte 21 to byte 24 are reserved, and can be set to 0x00.
transmitPacket[21] = transmitPacket[22] = transmitPacket[23] = transmitPacket[24] = 0x00;
Before send the transmitPacket, we should config the transmitter first. To config the transmitter, we need to send to start packet.

startPacket1[25] = {0xff, 0xb0, 0x01, 0x02, 0x03, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x31};
The last byte is 0x31

startPacket2 can set the radio frequency which should equal frequency point on the robot
startPacket2[25] = {0xff, 0xb0, 0x04, 0x05, 0x06};
startPacket2[5] = 0x10 + frequency;   // frequency is the transmitter's radio frequency (0 ~ 9)
startPacket2[24] = CCrc8::calc((unsigned char*)(startPacket2), 24)   // last byte is for Crc8 code
the CCr8::calc function can be found in the attach file
Other bytes are 0x00

Note: If you want to re-config the frequency, you must unplug the transmitter and plug it to computer again, then send start packet

After send the start packet, the transmitPacket contains command can be sent.

Now we can send the packet to the transmitter, (use Qt's QSerialPort for example)

```
QSerialPort serial;
serial.setPortName("COM1"); // serial port name is depended on your system

// use this config params
serial.setBaudRate(QSerialPort::Baud115200);
serial.setDataBits(QSerialPort::Data8);
serial.setParity(QSerialPort::NoParity);
serial.setStopBits(QSerialPort::OneStop);
serial.setFlowControl(QSerialPort::NoFlowControl);

if (serial.open(QIODevice::ReadWrite)) {
        qDebug() << "SerialPort Connected...";
        // send two start packets to config
        serial ->write((startPacket1), 25);    // 25 is packet size
        serial ->write((startPacket2),25);

        while (1) {
               // get command from somewhere and set to transmitPacket
               // ...
               serial -> write(transmitPacket,    25);
        }

}    else {
```

```
    qDebug() << "SerialPort connect failed...";
}
```

The send frequency of transmitPacket should be 60fps (60 packets per second).

Some notes:
When the transmitter is not configured, the two LEDs will blink alternately. When configuration is succeeded, the two LEDs will always light.
And when you are sending the transmitpackets, one of the LEDs will blink one time after 15 packet has been sent.
Because of the transmitpacket only contains 3 robots' command, and each robot should be controlled with about 60 fps, so 120 transmitpackets per second is needed when you want to control 6 robots simultaneously.