ICRA 2018 DJI RoboMaster

人工智能挑战赛裁判系统

规范手册

V1.1 2018.04

ROBOMASTER

免责声明

在使用之前，请仔细阅读本声明，一旦使用，即被视为对本声明全部内容的认可和接受。请严格遵守手册、产品说明和相关的法律法规、政策、准确安装和使用该产品。在使用产品过程中，用户承诺对自己的行为及因此而产生的所有后果负责。因用户不当使用、安装、改装造成的任何损失，大疆™ 创新（ DJI™ ）将不承担法律责任。DJI 和 RoboMaster™ 是深圳市大疆创新科技有限公司及其关联公司的商标或注册商标。本文出现的产品名称、品牌等，均为其所属公司的商标或注册商标。本产品及手册，包括与裁判系统配合使用的 RoboMaster Client、RoboMaster Assistant、RoboMaster Server 软件及 DJI WIN 驱动程序，为大疆创新版权所有。未经许可，不得以任何形式修改、复制、翻印或传播。本文档及本产品所有相关的文档最终解释权归 DJI 所有。所有内容，以最新版本号手册为准。

产品使用注意事项

1. 使用前请检查机器人端裁判系统监控装置已安装正确且牢固。
2. 使用前请确保连线正确。
3. 使用前请检查零部件是否完好，如有部件老化或损坏，请及时更换新部件。

阅读提示

符号说明

⚠ 重要注意事项　　操作、使用提示　　词汇解释、参考信息

前置参考阅读

1. RoboMaster 裁判系统用户手册
2. 裁判系统各模块说明书

修改日志

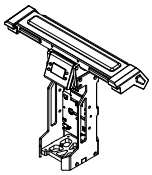
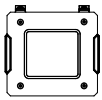
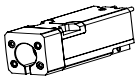
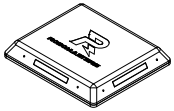
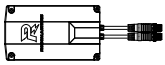
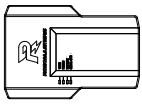
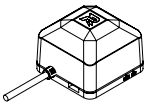
日期	版本	改动记录
2017.11.01	1.0	安装规范以及 ICRA 规则相关的功能描述
2018.04.16	1.1	修改裁判系统接口协议说明（P19）

目录

裁判系统使用规范说明

为保证 RoboMaster 2018 ICRA 的比赛公平公正，机器人对抗结果的评判完全由电子裁判系统自动执行。各参赛队必须严格遵守使用规范的各个事项，正确安装裁判系统，如果违反使用规范则无法于参赛前通过检录，后果由参赛队自行承担。

裁判系统组成

模块名称	外型	功能概述
主控模块		机器人监控装置的核心控制单元，可以监控整个裁判系统的运行状态，集成电源管理、人机交互，无线组网通信，状态显示以及机器人血量结算等功能。
装甲模块		感知弹丸的击打状态。
17mm 测速模块		实时检测弹丸的发射速度和发射频率数据。
场地交互模块		通过近场通信技术实现与场地模块的信息交互。
相机图传模块（发送端）		通过相机采集高清实时视频数据，通过无线方式传输到接收端。
相机图传模块（接收端）		接收高清实时视频数据，并通过 HDMI 接口进行输出。
定位模块		实时的室内定位模块，输出针对比赛场地的二维坐标信息。

模块配置表

必须安装模块	
模块名称	数量
主控模块	1
装甲模块（小）	4
17mm 测速模块	1
场地交互模块	1
选装模块	
模块名称	数量
相机图传模块（发送端）	1
定位模块	1

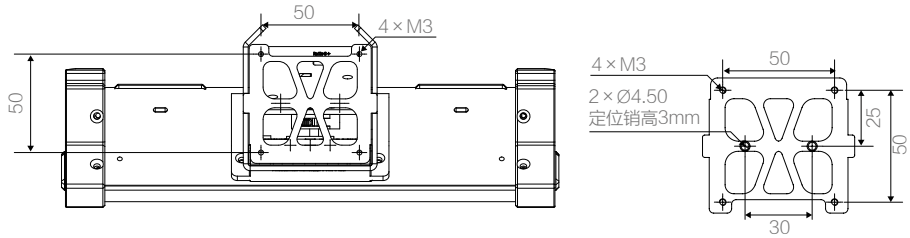
安装规范

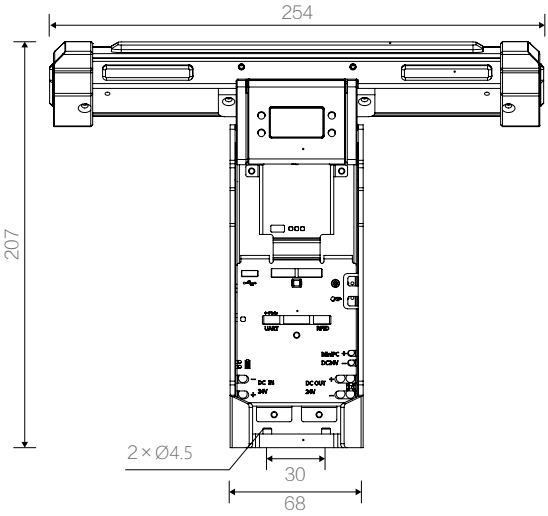
裁判系统是由 RM2018 组委会官方提供，可记录机器人在比赛中被攻击的情况，如血量值、弹丸发射速度、底盘功率，并将实时信息发送到对应操作间电脑以及裁判系统服务器，自动判定比赛胜负，确保比赛的公平性。参赛队设计的机器人需保留好机械和电气接口以便安装裁判系统。

主控模块

安装

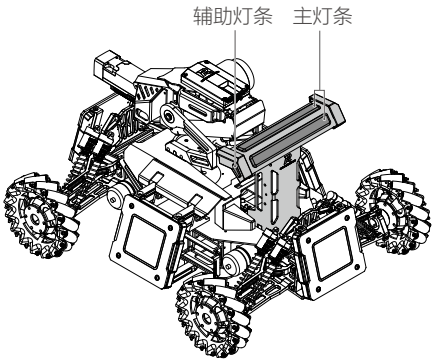
1. 参考主控模块尺寸，在机器人底盘预留安装孔位。主控模块固定座可以进行上下翻转，其中一面有两个距离 30mm，直径为 4.5mm 的定位销，如下图所示，可根据需求选择使用。





单位：mm

2. 使用 4 颗 M3 螺丝固定主控模块至底盘。
3. 主控模块的安装，必须保证左右辅助灯条的连线与地面平行；必须保证至少从一个方向直视机器人时，可以完整看到顶部主灯条的状态。

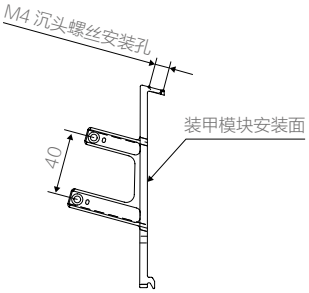


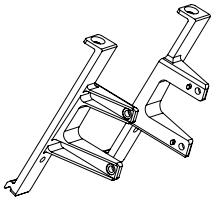
装甲模块

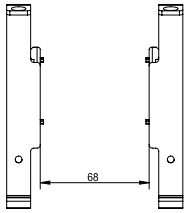
说明

装甲模块需要通过装甲支撑架安装至机器人。装甲支撑架包含两种，装甲支撑架 A 和装甲支撑架 B。

装甲支撑架 A，有左右之分，如下图所示：

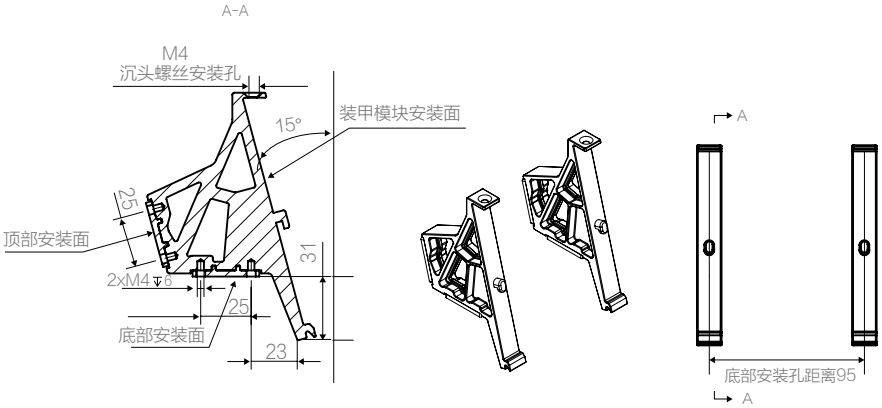




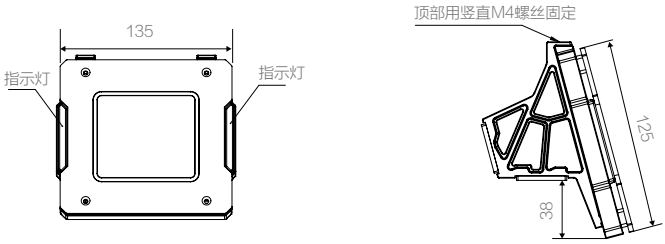


支撑架 A（左）　支撑架 A（右）

装甲支撑架 B 如下图所示：



小装甲模块如图所示：



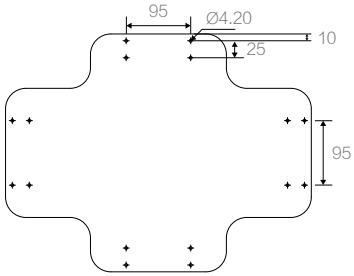
使用底部安装面进行安装

单位：mm

图中标注对于支撑架 A 和 B 同样适用，上图以支撑架 B 为例。

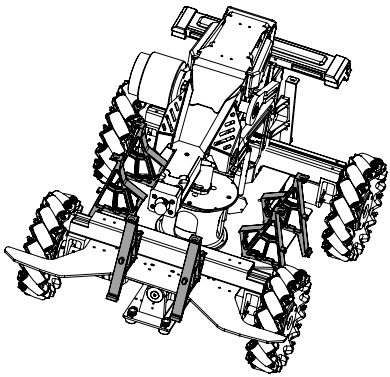
安装步骤

1. 按照下图尺寸，在底盘预留安装孔位，四个安装孔位大小及相对位置保持一致。

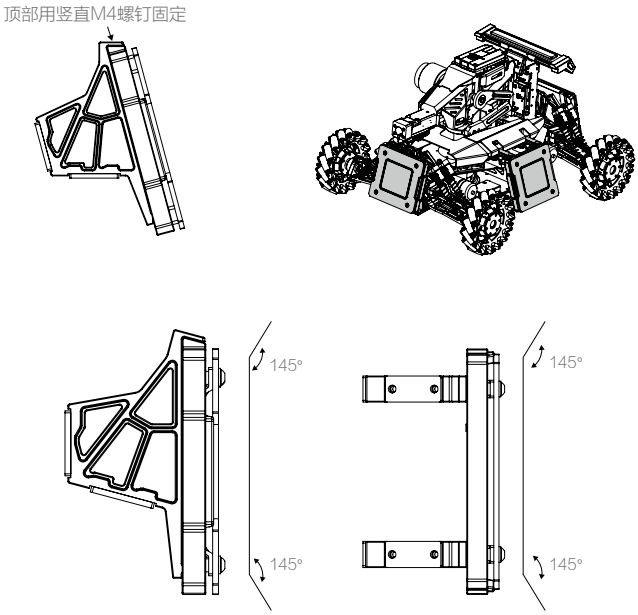


单位：mm

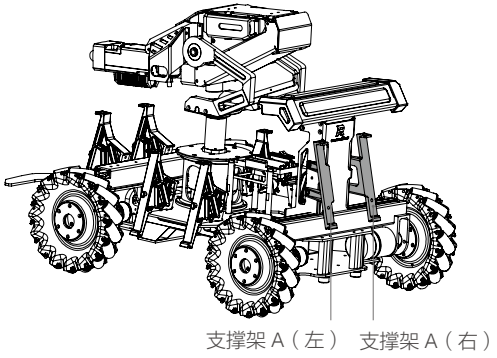
2. 使用 M4 螺丝固定支撑架 B 至三个面的底盘上。



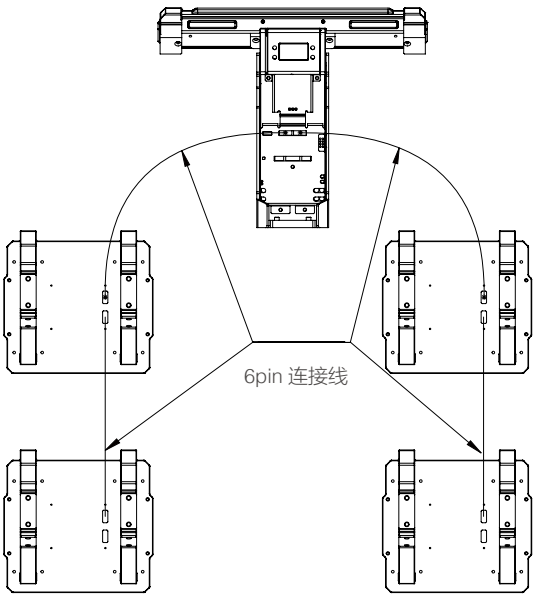
3. 安装装甲模块至支撑架 B，并使用 M4 螺丝固定，装甲支撑架顶部螺纹孔不和支撑架顶面垂直，在正确安装支撑架的情况下顶面螺纹孔与水平面垂直。装甲模块受攻击面 145° 内不得被遮挡。



4. 选装：可使用主控模块两侧的 M4 螺栓将装甲支撑架 A 固定至机器人后部。此时需先安装装甲模块至支撑架 A，再把安装好的整体安装到主控模块上。安装时需区分支撑架 A（左）和支撑架 A（右），注意支撑架的安装方向以保证其与主控模块紧固贴合。



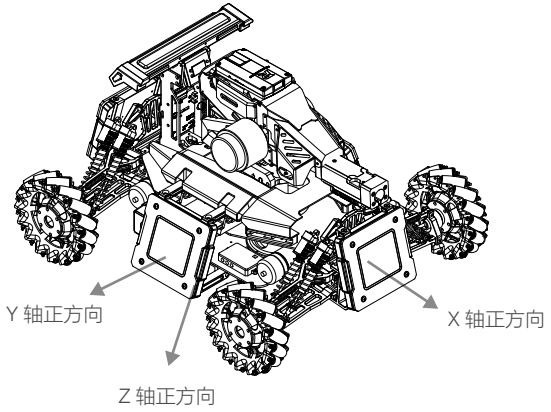
5. 使用包装内提供的 6pin 连接线串联各装甲模块至主控面板的装甲模块接口。装甲模块的两个 6pin 接口为等效接口，连接时建议均分主控面板两个接口上串联的装甲模块个数，以均分该接口的电流。



⚠ • 机器人装甲模块下边沿距离地面高度必须在 50mm-200mm 范围。

安装规范及要求

下文中的讨论中，机器人机体坐标系是标准的 X，Y，Z 笛卡尔坐标系，坐标原点为机器人的质量中心，如下图所示：



机器人本身的运动学方程须建立以笛卡尔坐标系为参考的机体坐标系下。如果参赛机器人使用非笛卡尔坐标系建立运动学模型，则机体坐标系定义为：机器人最大口径的发射机构初始状态下射出弹丸的方向向量投影到 XY 平面作为 X 轴，根据 X 轴和指向地心的 Z 轴按照右手定则生成 Y 轴，原点为机器人的质量中心。

侧面安装

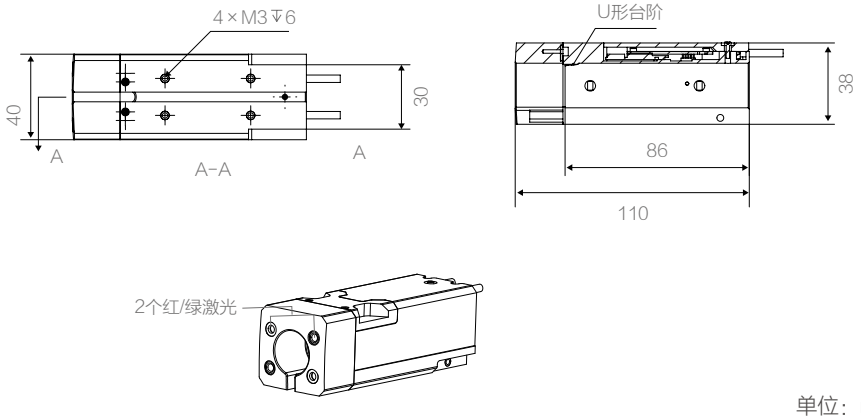
机器人进行侧面安装时装甲模块的受力面和支撑架必须稳固连接。装甲模块的支撑架底部连接面必须与 XY 平面平行，使得装甲模块受力面所在平面的法向量所在直线与 Z 轴负方向所在直线的锐角夹角为 75° 。装甲模块不含指示灯的两条边与 XY 平面保持平行。装甲模块安装好之后必须具备良好的刚性。定义一块安装好的装甲模块受力面所在平面的法向量（与 Z 轴负方向夹角为锐角）在 XY 平面上的投影为该装甲模块的方向向量。4 块装甲模块的方向向量的单位向量必须分别等于机器人机体坐标系的正 X 轴、负 X 轴，正 Y 轴，负 Y 轴（方向向量和对应坐标轴向量之间的角度误差不能超过 5° ）。机器人本身的运动学方程也必须建立在上述作为参考机体坐标系下。装甲模块的安装方式必须与机器人本身的结构特性或者运动学特性共享同一个参考坐标系。X 轴上安装的装甲模块几何中心点连线与 Y 轴上安装的装甲模块几何中心点连线要互相垂直，且连线穿过机器人的几何中心，X,Y 轴的装甲模块允许偏离几何中心正负 50mm。

⚠ • 自行设计的保护装甲，不能与官方提供的装甲模块有任何接触。
• 请勿对官方装甲模块进行任何修改和装饰。

测速模块

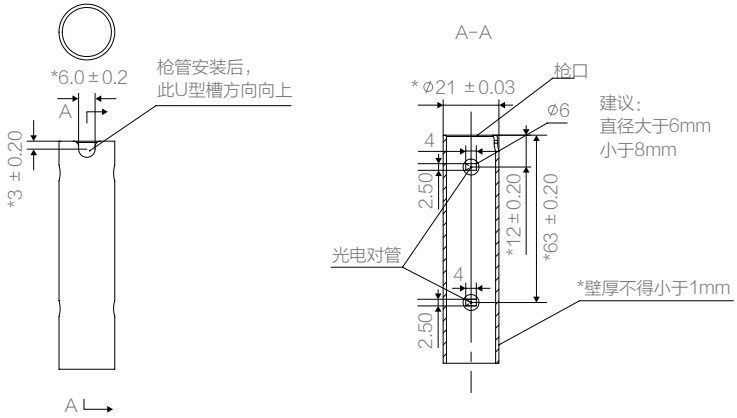
安装

17mm 测速模块：



单位：mm

17mm 弹丸枪管尺寸限制：



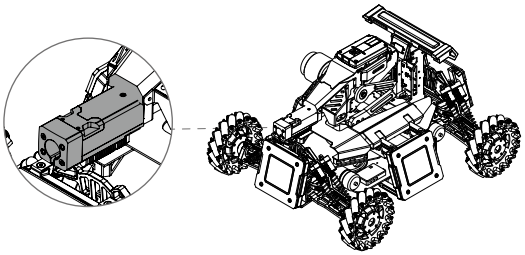
单位：mm

枪管要求：

1. 枪管长必须大于 90mm。
2. 加 * 号为参赛选手需要重点掌控尺寸。
3. 保证光电管不被遮挡。
4. 禁止使用透明材料。

安装步骤：

1. 把测速模块套在枪管上，使 U 形台阶对齐枪管的 U 形槽，连线一端朝向主控模块。
2. 使用 M3 螺丝穿过测速模块后部的螺丝孔以夹紧枪管。
3. 连接测速模块与主控面板上测速接口的航空插头。安装完成后的效果图如下所示：

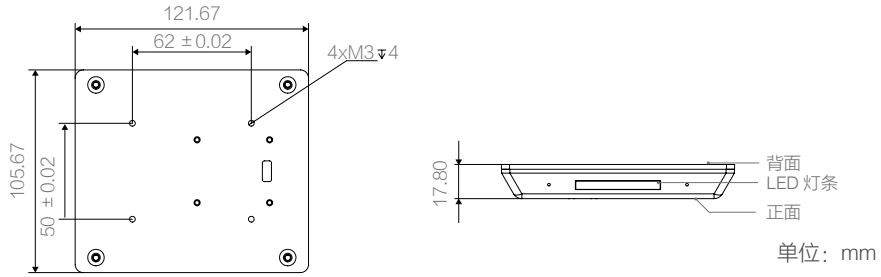


⚠ • 注意红外对管的安装孔位一定不要遮挡。否则会导致测速模块自检不过。
• 注意测速模块要固定牢固，确保使用过程中测速模块和枪口不能发生相对移动。

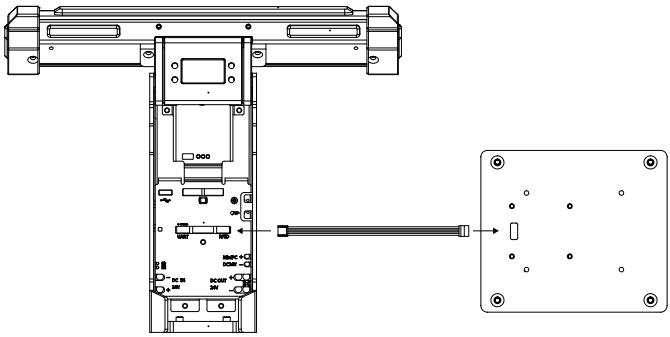
场地交互模块

安装

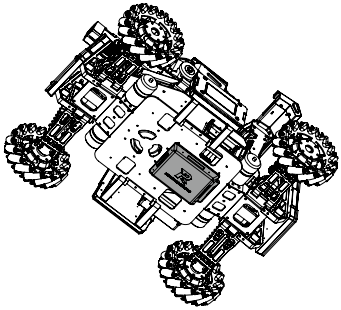
1. 参考场地交互模块结构尺寸和安装接口在底盘预留安装孔位。



2. 使用包装内提供的 4pin 连接线将裁判系统场地交互模块连接到裁判系统主控模块的主控面板上的 RFID 接口。



3. 使用 M3 螺丝固定裁判系统场地交互模块至底盘，安装时切勿压到连接线，并注意保持交互模块与地面有适当的距离。

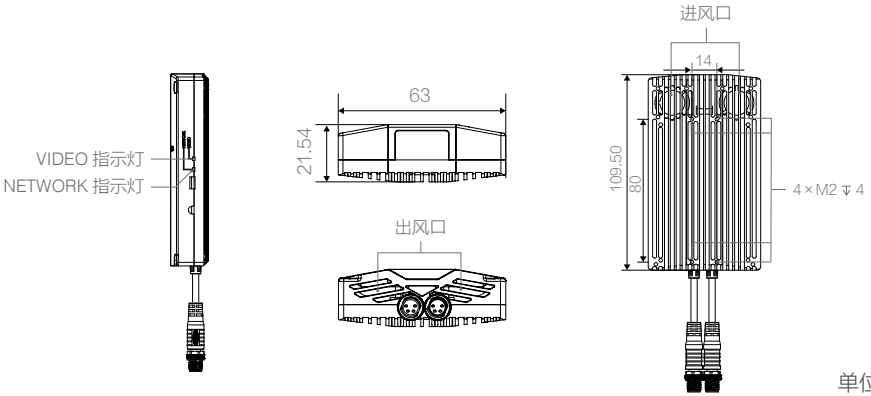


⚠ • 确保场地交互模块有 Logo 的面安装后没有金属遮挡，安装后实际检测距离以测试为准，如果有效检测距离缩短，请检查安装是否合理。
• 赠送的场地交互测试卡可实现增加机器人攻击力 50% 功能。

相机图传模块

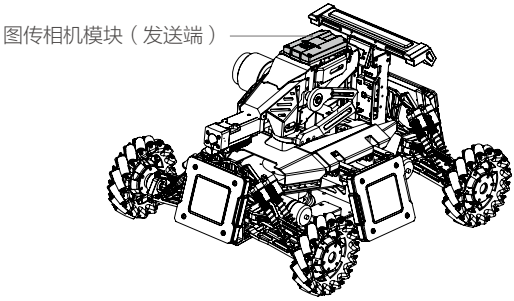
发送端安装

1. 参考发送端结构尺寸和安装接口在所需位置预留安装孔位。



单位：mm

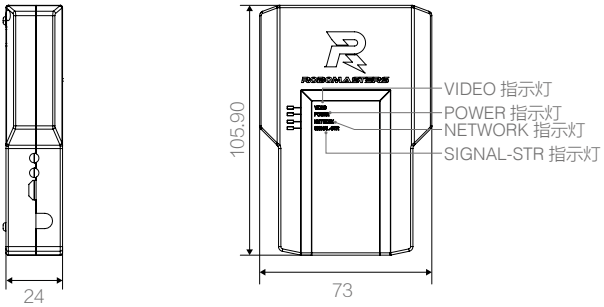
使用 4 颗 M2 螺丝固定发送端至适当位置。安装位置不能遮挡相机图传模块的进风口与出风口；相机图传模块的天线在模块顶部，因此顶部不能有任何金属遮挡。不按要求安装，会导致图传模块工作异常。



2. 发送端的航空插头与主控面板上图传接口的航空插头相连接。

接收端安装

相机图传模块的接收端可以使用配送的安装夹进行固定。固定的位置可以是显示器或者其它支撑物，需要保证固定位置离地高度不低于 1m, 且没有金属遮挡，具体的安装位置，可以通过查看接收图像质量确认。接收端模块如下图所示：

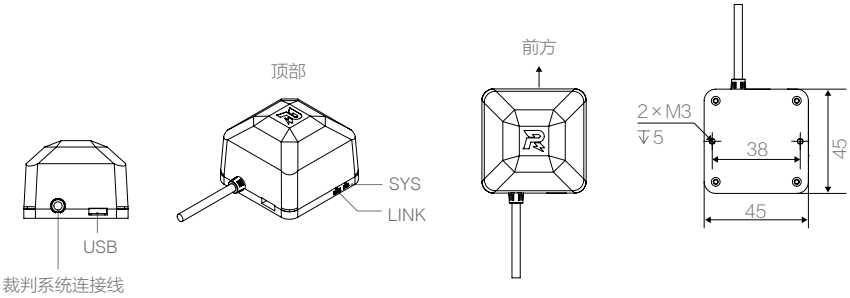


单位：mm

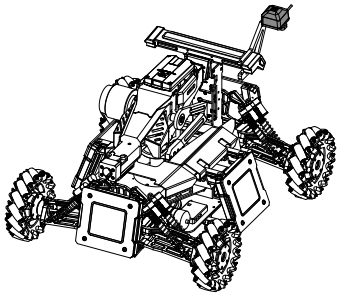
定位模块

安装

1. 参考定位模块尺寸在特定位置预留安装孔位。



2. 使用 2 颗 M3 螺丝固定定位模块至特定位置。定位模块的前方必须与机器人的前方保持一致，并且顶部朝上水平安装。



3. 使用包装内的航空插头对接线连接定位模块至相机图传模块的航空插头。

⚠ • 相机图传模块（发送端）、测速模块、定位模块的航空插头和主控模块上的航空插头均为等效接口。
• 定位基站固定在场地四周围栏的顶部，在机器人运动过程中，须全程保证定位模块和各个基站之间的直线连线中间不能有自身遮挡。推荐将定位模块安装在最高点。
• 安装位置离电机、相机图传模块、带磁性或运行过程中会产生强烈磁场的部件距离推荐在 20cm 以上，最短不能少于 10cm。

功能概述及使用规范

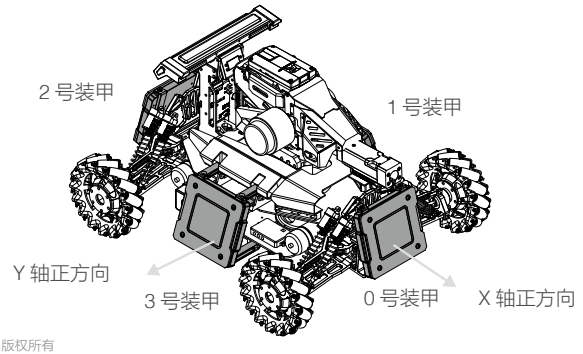
机器人参数

参数类型	数值
机器人初始血量	2000
最大射速	20m/s
最大射频	10Hz
单发弹丸伤害	50

装甲模块 ID 设置规范

裁判系统各模块通信使用 CAN 网络，因此每个模块均需要有唯一的 ID 设置，才能保证正常的通信。裁判系统在生产的过程中，对装甲模块均设置了一个默认的 ID 号。因此在第一次连接多块装甲模块，或者重新更新了新装甲模块后，需要对装甲模块进行 ID 设置，ID 设置的步骤如下：

1. 在裁判系统交互主页面下，长按“确认按键”，进入裁判系统“功能页面”。
2. 在“功能页面”下，短按“上下翻按键”选中“System Setup”选项，进入“系统设置页面”。
3. 在“系统设置页面”，短按“上下翻按键”选中“Armor ID Setup”，然后短按“确认按键”将进入装甲 ID 设置模式，选中“Armor ID Reset”，进入装甲 ID 重置状态，此时装甲指示灯以一定频率闪烁（如果机器人 ID 为红方，则红灯闪烁，如果机器人 ID 为蓝方，则蓝灯闪烁）。
4. 以一定力度依次敲击装甲模块，装甲指示灯停止闪烁，表明该装甲 ID 设置成功。首次被敲击的装甲 ID 编号为 0，并且装甲 ID 编号根据敲击的先后顺序依次递增。完成以上操作后，可以通过查询装甲模块的版本号来确认装甲 ID 设置是否成功。如果读取的有效装甲模块数和实际安装的装甲模块数量相同，则表示装甲模块的 ID 设置成功。机器人装甲模块的 ID 编号有严格要求，标识如下图所示：



⚠ • 比赛前一定要提前设置好装甲模块的 ID，如果设置异常，裁判系统自检过程中，会检测不到设置错误的装甲模块 ID 号，在比赛过程中会判断为装甲模块离线，自动扣除机器人血量。

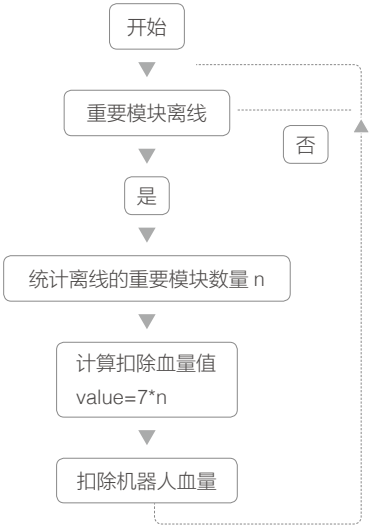
模块状态监测

裁判系统各模块正常工作，是保证比赛公平公正的前提条件。因此，裁判系统在上电启动之后，会对各个模块进行自检。根据对比赛局势的影响程度，对各模块进行了重要程度的两级分类：重要模块与一般模块。

重要模块：装甲模块、测速模块、主控模块

一般模块：场地交互模块、相机图传模块、定位模块

重要模块会影响比赛的公平性，如装甲模块，一旦出现异常，将无法检测到敌方攻击。正式比赛中，裁判系统以 2HZ 的频率对各模块进行自检，一旦检测到重要模块离线，将会自动扣除机器人相应血量值。扣血计算方式如下流程图所示：



一般模块离线本身会对己方不利，因此裁判系统不做任何处理，只会通过裁判系统主控模块进行辅助灯黄色闪烁进行提醒。

射速射频限制

类型	上限值	惩罚
射速	20m/s	a. 超速比例小于等于 10%，则扣除机器人总血量的 10% b. 超速比例大于 10% 小于等于 20%，则扣除机器人总血量的 20% c. 超速比例大于 20%，则扣除机器人总血量的 40%
射频	10HZ	射频超过 10HZ，则每超一颗扣除机器人总血量的 10%

主控灯条状态说明

正常状态		
主灯条状态	辅助灯条状态	描述
部分红色或者蓝色 LED 常亮，部分 LED 熄灭	红色或者蓝色常亮	红色或者蓝色表示机器人身份，常亮 LED 部分表示机器人血量百分比。全部 LED 常亮表示满血状态
红色或者蓝色常亮	白色快速闪烁	机器人获得攻击力 50% 加成
警告或者异常		
黄灯常亮	黄灯常亮	非比赛中，重要模块离线
正常状态	黄灯闪烁	非比赛中，一般模块离线

赛前 20s 系统自检

在两分钟比赛准备阶段结束后，会进入 20 秒的裁判系统自检阶段，自检结束后才正式开始比赛。自检过程中，比赛服务器会自动检测客户端连接状态，比赛机器人无线连接状态，机器人模块状态，场内道具状态等，若状态不符合开始比赛需求，如客户端离线，机器人离线，场内道具离线等，比赛自检倒计时将会暂停，待修复好故障设施后，由裁判恢复自检，自检倒计时继续。进入 20s 系统自检时，比赛服务器会恢复所有机器人血量，确保正式比赛开始时，所有机器人为满血状态。20s 裁判系统自检阶段，参赛选手只能待在比赛操作间，在这期间选手可以检查用于比赛使用的电脑鼠标、键盘是否功能正常。

比赛信息接口

为了方便参赛选手更好的做自动控制以及获得比赛的实时状态，裁判系统设计了一路 UART 输入输出口，输出比赛中机器人以及比赛战场的部分数据；同时可以透传参赛选手自定义的部分数据，显示在客户端的 UI 上。输出的信息包含比赛剩余时间、机器人剩余血量、机器人的定位数据、被攻击的装甲 ID 号；开放的上行数据接口，保留了三个浮点类型数据，参赛选手可以根据“裁判系统接口协议说明”进行实现，上行数据最终会在对应的客户端 UI 页面呈现。

附 录

裁判系统接口协议说明

通信协议格式

FrameHeader(5-Byte)	CmdID(2-Byte)	Data(n-Byte)	FrameTail(2-Byte, CRC16)

FrameHeader 格式

域	偏移位置	大小（字节）	详细描述
SOF	0	1	帧起始字节，固定值为 0xA5
DataLength	1	2	数据段 Data 长度
Seq	3	1	包序号
CRC8	4	1	帧头 CRC8 校验

命令码 ID

简介

命令码	数据段长度（Byte）	功能说明
0x0001	8	比赛机器人状态，10Hz 频率周期发送
0x0002	1	伤害数据，收到伤害时发送
0x0003	6	实时射击数据，发射弹丸时发送
0x0005	2	实时场地交互数据，检测到 RFID 卡时 10Hz 周期发送
0x0006	1	比赛结果数据，比赛结束时发送一次
0x0007	2	Buff 状态，任意 Buff 状态改变时发送一次
0x0008	16	机器人位置朝向信息，50Hz 频率周期发送
0x0100	13	参赛队自定义数据，用于显示在操作界面，限频 10Hz

详细说明

比赛机器人状态 (0x0001)

字节偏移	大小	说明
0	2	当前阶段剩余时间，单位 s

2	1	当前比赛阶段 0：未开始比赛 1：准备阶段 2：自检阶段 3：5s 倒计时 4：对战中 5：比赛结算中
3	1	保留
4	2	机器人当前血量
6	2	机器人满血量

结构体定义：

```
typedef __packed struct
{
  uint16_t  stageRemianTime;
  uint8_t   gameProgress;
  uint8_t   reserved;
  uint16_t  remainHP;
  uint16_t  maxHP;
}extGameRobotState_t;
```

伤害数据 (0x0002)

字节偏移	大小	说明
0	1	0-3bits：若变化类型为装甲伤害时，标识装甲 ID 0x0：0 号装甲（前） 0x1：1 号装甲（左） 0x2：2 号装甲（后） 0x3：3 号装甲（右） 其他保留 4-7bits：血量变化类型 0x0：装甲伤害（受到攻击） 0x1：模块掉线

结构体定义：

```
typedef __packed struct
{
  uint8_t  armorType : 4;
  uint8_t  hurtType  : 4;
}extRobotHurt_t;
```


实时射击数据 (0x0003)

字节偏移	大小	说明
0	1	保留
1	1	弹丸射频
2	4	弹丸射速

结构体定义：

```
typedef __packed struct
{
  uint8_t  reserved;
  uint8_t  bulletFreq;
  float    bulletSpeed;
}extShootData_t;
```

场地交互数据（0x0005）

字节偏移	大小	说明
0	1	卡类型 11：大能量机关打击点卡 其他：保留
1	1	卡索引号，可用于区分不同区域

结构体定义：

```
typedef __packed struct
{
  uint8_t  cardType;
  uint8_t  cardIdx;
}extRfidDetect_t;
```


比赛胜负数据（0x0006）

字节偏移	大小	说明
0	1	比赛结果 0：平局 1：红方胜 2：蓝方胜

结构体定义：

```
typedef __packed struct
{
  uint8_t  winner;
}extGameResult_t;
```

Buff 状态（0x0007）

字节偏移	大小	说明
0	2	Buff 类型，1 表示有效： bit 13：ICRA 己方获得小能量机关 bit 14：ICRA 敌方获得小能量机关 其他：保留

结构体定义：

```
typedef __packed struct
{
  uint16_t  buffMusk;
} extGetBuff_t;
```

机器人位置朝向信息（0x0008）

字节偏移	大小	说明
0	4	位置 X 坐标值
4	4	位置 Y 坐标值
8	4	位置 Z 坐标值
12	4	保留

结构体定义：

```
typedef __packed struct
{
  float  x;
  float  y;
  float  z;
  float  reserved;
}extGameRobotPos_t;
```


参赛队自定义数据（0x0100）

字节偏移	大小	说明
0	4	自定义数据 1
4	4	自定义数据 2
8	4	自定义数据 3
12	1	自定义数据 4

结构体定义：

```
typedef __packed struct
{
  float  data1;
  float  data2;
float  data3;
uint8_t mask;
}extShowData_t;
```


CRC 校验代码示例

```
//crc8 generator polynomial:G(x)=x8+x5+x4+1
const unsigned char CRC8_INIT = 0xff;
const unsigned char CRC8_TAB[256] =
{
0x00, 0x5e, 0xbc, 0xe2, 0x61, 0x3f, 0xdd, 0x83, 0xc2, 0x9c, 0x7e, 0x20, 0xa3, 0xfd, 0x1f, 0x41,
0x9d, 0xc3, 0x21, 0x7f, 0xfc, 0xa2, 0x40, 0x1e, 0x5f, 0x01, 0xe3, 0xbd, 0x3e, 0x60, 0x82, 0xdc, 0x23,
0x7d, 0x9f, 0xc1, 0x42, 0x1c, 0xfe, 0xa0, 0xe1, 0xbf, 0x5d, 0x03, 0x80, 0xde, 0x3c, 0x62, 0xbe, 0xe0,
0x02, 0x5c, 0xdf, 0x81, 0x63, 0x3d, 0x7c, 0x22, 0xc0, 0x9e, 0x1d, 0x43, 0xa1, 0xff, 0x46, 0x18, 0xfa,
0xa4, 0x27, 0x79, 0x9b, 0xc5, 0x84, 0xda, 0x38, 0x66, 0xe5, 0xbb, 0x59, 0x07, 0xdb, 0x85, 0x67, 0x39,
0xba, 0xe4, 0x06, 0x58, 0x19, 0x47, 0xa5, 0xfb, 0x78, 0x26, 0xc4, 0x9a, 0x65, 0x3b, 0xd9, 0x87, 0x04,
0x5a, 0xb8, 0xe6, 0xa7, 0xf9, 0x1b, 0x45, 0xc6, 0x98, 0x7a, 0x24, 0xf8, 0xa6, 0x44, 0x1a, 0x99, 0xc7,
0x25, 0x7b, 0x3a, 0x64, 0x86, 0xd8, 0x5b, 0x05, 0xe7, 0xb9,
0x8c, 0xd2, 0x30, 0x6e, 0xed, 0xb3, 0x51, 0x0f, 0x4e, 0x10, 0xf2, 0xac, 0x2f, 0x71, 0x93, 0xcd, 0x11,
0x4f, 0xad, 0xf3, 0x70, 0x2e, 0xcc, 0x92, 0xd3, 0x8d, 0x6f, 0x31, 0xb2, 0xec, 0x0e, 0x50, 0xaf, 0xf1,
0x13, 0x4d, 0xce, 0x90, 0x72, 0x2c, 0x6d, 0x33, 0xd1, 0x8f, 0x0c, 0x52, 0xb0, 0xee, 0x32, 0x6c, 0x8e,
0xd0, 0x53, 0x0d, 0xef, 0xb1, 0xf0, 0xae, 0x4c, 0x12, 0x91, 0xcf, 0x2d, 0x73, 0xca, 0x94, 0x76, 0x28,
0xab, 0xf5, 0x17, 0x49, 0x08, 0x56, 0xb4, 0xea, 0x69, 0x37, 0xd5, 0x8b, 0x57, 0x09, 0xeb, 0xb5, 0x36,
0x68, 0x8a, 0xd4, 0x95, 0xcb, 0x29, 0x77, 0xf4, 0xaa, 0x48, 0x16, 0xe9, 0xb7, 0x55, 0x0b, 0x88, 0xd6,
0x34, 0x6a, 0x2b, 0x75, 0x97, 0xc9, 0x4a, 0x14, 0xf6, 0xa8,
0x74, 0x2a, 0xc8, 0x96, 0x15, 0x4b, 0xa9, 0xf7, 0xb6, 0xe8, 0x0a, 0x54, 0xd7, 0x89, 0x6b, 0x35,
};
unsigned char Get_CRC8_Check_Sum(unsigned char *pchMessage,unsigned int dwLength,unsigned
char ucCRC8)
{
unsigned char ucIndex;
while (dwLength--)
{
ucIndex = ucCRC8^(*pchMessage++);
ucCRC8 = CRC8_TAB[ucIndex];
}
return(ucCRC8);
}
/*
** Descriptions: CRC8 Verify function
** Input: Data to Verify,Stream length = Data + checksum
** Output: True or False (CRC Verify Result)
*/
```



```
unsigned int Verify_CRC8_Check_Sum(unsigned char *pchMessage, unsigned int dwLength)
{
unsigned char ucExpected = 0;
if ((pchMessage == 0) || (dwLength <= 2)) return 0;
ucExpected = Get_CRC8_Check_Sum (pchMessage, dwLength-1, CRC8_INIT);
return ( ucExpected == pchMessage[dwLength-1] );
}
/*
** Descriptions: append CRC8 to the end of data
** Input: Data to CRC and append,Stream length = Data + checksum
** Output: True or False (CRC Verify Result)
*/
void Append_CRC8_Check_Sum(unsigned char *pchMessage, unsigned int dwLength)
{
unsigned char ucCRC = 0;
if ((pchMessage == 0) || (dwLength <= 2)) return;
ucCRC = Get_CRC8_Check_Sum ( (unsigned char *)pchMessage, dwLength-1, CRC8_INIT);
pchMessage[dwLength-1] = ucCRC;
}

uint16_t CRC_INIT = 0xffff;
const uint16_t wCRC_Table[256] =
{
0x0000, 0x1189, 0x2312, 0x329b, 0x4624, 0x57ad, 0x6536, 0x74bf,
0x8c48, 0x9dc1, 0xaf5a, 0xbed3, 0xca6c, 0xdbe5, 0xe97e, 0xf8f7,
0x1081, 0x0108, 0x3393, 0x221a, 0x56a5, 0x472c, 0x75b7, 0x643e,
0x9cc9, 0x8d40, 0xbfdb, 0xae52, 0xdaed, 0xcb64, 0xf9ff, 0xe876,
0x2102, 0x308b, 0x0210, 0x1399, 0x6726, 0x76af, 0x4434, 0x55bd,
0xad4a, 0xbcc3, 0x8e58, 0x9fd1, 0xeb6e, 0xfae7, 0xc87c, 0xd9f5,
0x3183, 0x200a, 0x1291, 0x0318, 0x77a7, 0x662e, 0x54b5, 0x453c,
0xbdcb, 0xac42, 0x9ed9, 0x8f50, 0xfbef, 0xea66, 0xd8fd, 0xc974,
0x4204, 0x538d, 0x6116, 0x709f, 0x0420, 0x15a9, 0x2732, 0x36bb,
0xce4c, 0xdfc5, 0xed5e, 0xfcd7, 0x8868, 0x99e1, 0xab7a, 0xbaf3,
0x5285, 0x430c, 0x7197, 0x601e, 0x14a1, 0x0528, 0x37b3, 0x263a,
0xdecd, 0xcf44, 0xfddf, 0xec56, 0x98e9, 0x8960, 0xbbfb, 0xaa72,
0x6306, 0x728f, 0x4014, 0x519d, 0x2522, 0x34ab, 0x0630, 0x17b9,
0xef4e, 0xfec7, 0xcc5c, 0xddd5, 0xa96a, 0xb8e3, 0x8a78, 0x9bf1,
0x7387, 0x620e, 0x5095, 0x411c, 0x35a3, 0x242a, 0x16b1, 0x0738,
0xffcf, 0xee46, 0xdcdd, 0xcd54, 0xb9eb, 0xa862, 0x9af9, 0x8b70,
```



```
0x8408, 0x9581, 0xa71a, 0xb693, 0xc22c, 0xd3a5, 0xe13e, 0xf0b7,
0x0840, 0x19c9, 0x2b52, 0x3adb, 0x4e64, 0x5fed, 0x6d76, 0x7cff,
0x9489, 0x8500, 0xb79b, 0xa612, 0xd2ad, 0xc324, 0xf1bf, 0xe036,
0x18c1, 0x0948, 0x3bd3, 0x2a5a, 0x5ee5, 0x4f6c, 0x7df7, 0x6c7e,
0xa50a, 0xb483, 0x8618, 0x9791, 0xe32e, 0xf2a7, 0xc03c, 0xd1b5,
0x2942, 0x38cb, 0x0a50, 0x1bd9, 0x6f66, 0x7eef, 0x4c74, 0x5dfd,
0xb58b, 0xa402, 0x9699, 0x8710, 0xf3af, 0xe226, 0xd0bd, 0xc134,
0x39c3, 0x284a, 0x1ad1, 0x0b58, 0x7fe7, 0x6e6e, 0x5cf5, 0x4d7c,
0xc60c, 0xd785, 0xe51e, 0xf497, 0x8028, 0x91a1, 0xa33a, 0xb2b3,
0x4a44, 0x5bcd, 0x6956, 0x78df, 0x0c60, 0x1de9, 0x2f72, 0x3efb,
0xd68d, 0xc704, 0xf59f, 0xe416, 0x90a9, 0x8120, 0xb3bb, 0xa232,
0x5ac5, 0x4b4c, 0x79d7, 0x685e, 0x1ce1, 0x0d68, 0x3ff3, 0x2e7a,
0xe70e, 0xf687, 0xc41c, 0xd595, 0xa12a, 0xb0a3, 0x8238, 0x93b1,
0x6b46, 0x7acf, 0x4854, 0x59dd, 0x2d62, 0x3ceb, 0x0e70, 0x1ff9,
0xf78f, 0xe606, 0xd49d, 0xc514, 0xb1ab, 0xa022, 0x92b9, 0x8330,
0x7bc7, 0x6a4e, 0x58d5, 0x495c, 0x3de3, 0x2c6a, 0x1ef1, 0x0f78
};
/*
** Descriptions: CRC16 checksum function
** Input: Data to check,Stream length, initialized checksum
** Output: CRC checksum
*/
uint16_t Get_CRC16_Check_Sum(uint8_t *pchMessage,uint32_t dwLength,uint16_t wCRC)
{
Uint8_t chData;
if (pchMessage == NULL)
{
return 0xFFFF;
}
while(dwLength--)
{
chData = *pchMessage++;
(wCRC) = ((uint16_t)(wCRC) >> 8) ^ wCRC_Table[((uint16_t)(wCRC) ^ (uint16_t)(chData)) & 0x00ff];
}
return wCRC;
}

/*
** Descriptions: CRC16 Verify function
```



```
** Input: Data to Verify,Stream length = Data + checksum
** Output: True or False (CRC Verify Result)
*/
uint32_t Verify_CRC16_Check_Sum(uint8_t *pchMessage, uint32_t dwLength)
{
uint16_t wExpected = 0;
if ((pchMessage == NULL) || (dwLength <= 2))
{
return __FALSE;
}
wExpected = Get_CRC16_Check_Sum ( pchMessage, dwLength - 2, CRC_INIT);
return ((wExpected & 0xff) == pchMessage[dwLength - 2] && ((wExpected >> 8) & 0xff) ==
pchMessage[dwLength - 1]);
}

/*
** Descriptions: append CRC16 to the end of data
** Input: Data to CRC and append,Stream length = Data + checksum
** Output: True or False (CRC Verify Result)
*/
void Append_CRC16_Check_Sum(uint8_t * pchMessage,uint32_t dwLength)
{
uint16_t wCRC = 0;
if ((pchMessage == NULL) || (dwLength <= 2))
{
return;
}
wCRC = Get_CRC16_Check_Sum ( (U8 *)pchMessage, dwLength-2, CRC_INIT );
pchMessage[dwLength-2] = (U8)(wCRC & 0x00ff);
pchMessage[dwLength-1] = (U8)((wCRC >> 8)& 0x00ff);
```

⚠ • Uart 通信配置，波特率 115200，数据位 8，停止位 1，校验位无，流控制无。

联系我们

邮箱 : robomaster@dji.com

ICRA2018 官网：icra2018.org/

RoboMaster 官网：www.robomaster.com/en-US

联系电话 : 0755-36383255（周一至周五 10:00-19:00，北京时间）

微信

RoboMasterNews

微博

RoboMaster