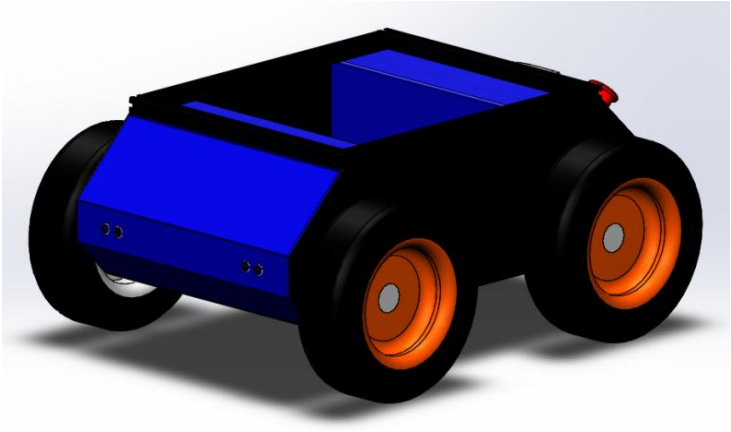
Scholar E600 移动机器人底盘
使用说明手册



北京维尔玛科技有限公司

2020 年 3 月 15 日

目 录

1. 产品介绍

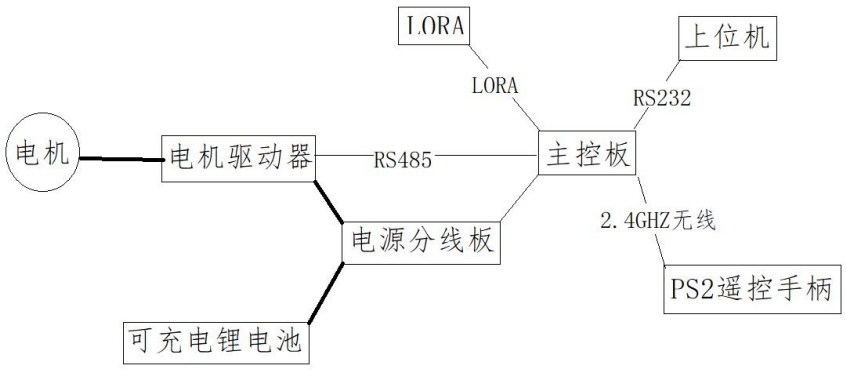
Scholar E600 移动机器人底盘（以下简称机器人底盘）是一款模块化的机器人底盘，适用于机器人教学、科学研究和产品开发等领域。

机器人底盘前后各设置了两路防撞保护声波传感器；当前方 10cm 内有障碍物靠近时，机器人底盘紧急停止不再前进（此时可原地转弯或后退）；当后方 10cm 内有障碍物靠近时，机器人底盘紧急停止不再后退（此时可原地转弯或前进）；

机器人底盘预设了低电量保护功能，当机器人剩余电量低于 10%时，机器人自动进入自由停止状态（请注意防范溜坡），此时急停按钮失效，LORA 进入休眠状态，机器人底盘收到指令包后，只应答低电量报警信息。

机器人底盘集成了 SEMTECH 公司的 LORA 无线扩频模块，最远通信距离可达 5000m（晴朗空旷环境，最大功率，天线增益 5dBi，高度 2m）；空中传输速率 2.4Kbps，发射功率 30dBm。控制端可通过 LORA-4G/5G 与物联网云平台进行连接。

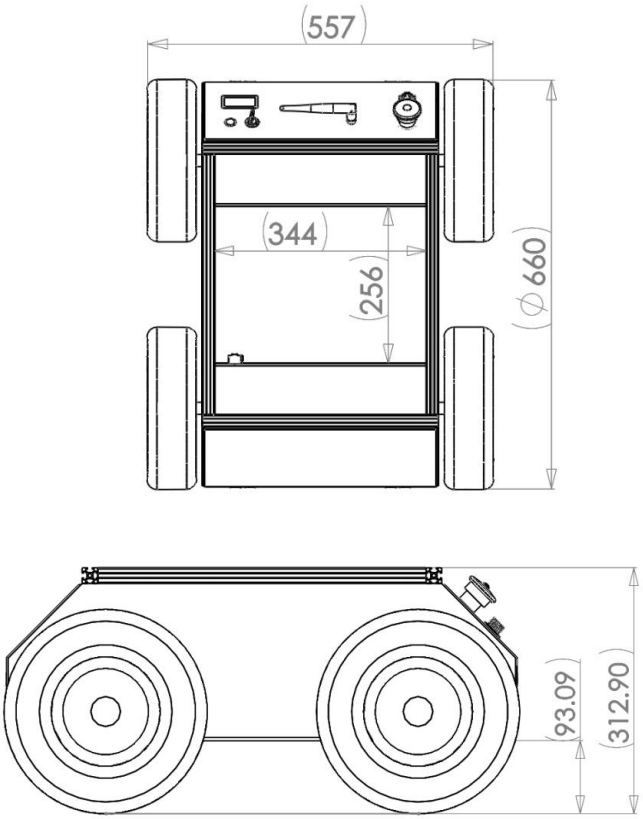
1.1. 硬件结构



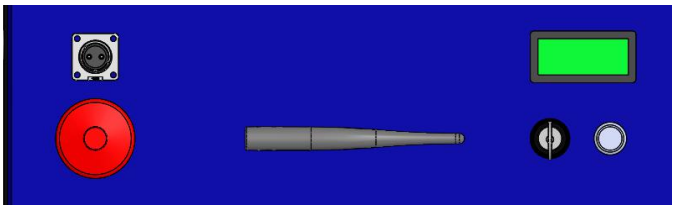
1.2. 规格参数：

Scholar E600 移动机器人底盘本体			
尺寸	660 x 560 x 310 mm	轮胎尺寸	10 寸（260mm）
自重	35Kg	车身最小离地距离	90mm
最大负载	40Kg	通信方式	RS232、LORA、PS2
最大速度	1m/s	输出电源	24V@5A、12V@8A、5V@5A
最大爬坡角度	20°	最小转弯半径	0°
电池容量	24V@20Ah	续航时间	通常：4 小时 重载：1.5 小时
电池充电器输入电压	100-240VAC 50/60Hz	电池充电器输出电压	29.4V@5A
充电时间	4h	电源保护功能	电源保险管预设规格 30A
驱动电机	空心杯直流无刷减速电机	电机额定功率	70W*4
电机减速器规格（减速比）	81.4	驱动器	高性能直流无刷驱动器
低电量保护功能	有	防撞保护功能	有

1.3. 尺寸参数：



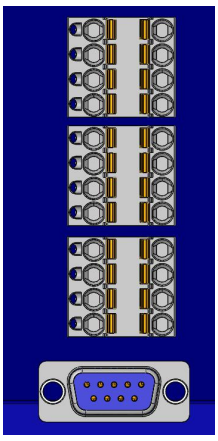
1.4. 后面板介绍



名称	个数	说明及注意事项
急停开关	1 个	按下时，机器人紧急停止
启动开关	1 个	按下时，机器人启动
三挡钥匙开关	1 个	左侧为 PS2 控制模式；中间位置为串口控制模式；右侧位置为 LORA 控制模式
充电口	1 个	给 Scholar E600 机器人电池充电
LORA 胶棒天线	1 个	LORA 通信天线
电量显示屏	1 个	以百分比的形式显示机器人剩余电量

机器人剩余电量低于 10%时，需要给机器人充电；此时，机器人自动进入自由停止状态，急停按钮失效（按下急停按钮，机器人不会紧急停止），LORA 进入休眠状态，串口指令不执行。

1.5. 电源输出及通讯面板介绍



机器人底盘提供 4 路 24V（共 5A）、4 路 12V（共 8A）、4 路 5V（共 5A）电源输出；RS232 通信接口。

1.6. 控制手柄参数

参数名称	参数内容
电池	AAA（7 号）电池*2
使用时间	约 10小时
无线频率	2.4GHz
接收范围	8m

1.7. 发货清单：

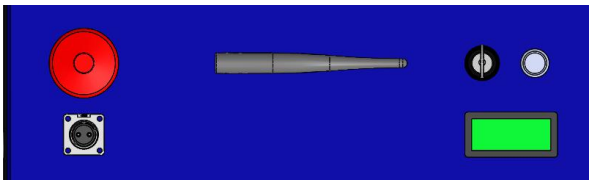
名称	数量
机器人底盘	1 台
RS232-USB 数据	1 根
电池充电器	1 套
遥控手柄	1 个
使用说明书	1 本

2. 产品使用

2.1. 控制方式介绍

Scholar E600 机器人支持 PS2 手柄控制、RS232 串口控制和 LORA 远程控制。通过后面板的**三挡钥匙开关**进行控制方式的选择。

2.2. 手柄控制说明



按下机器人底盘的启动开关（此时，启动开关的环形指示灯会变亮），确保剩余大量大于 10%，确保急停开关没有被按下，确保三挡钥匙开关旋转到左侧状态,确保机器人周边 10cm 内没有障碍物。

2.2.1. 手柄控制步骤：

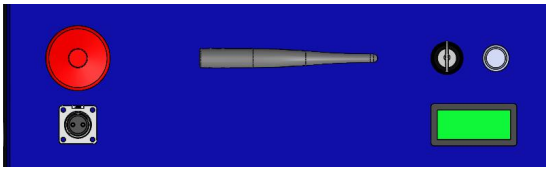
- ✓ 拨动手柄开关到“ON”档位；按“START”按钮，绿色指示灯（“POWER”指示灯）闪烁，配对后常亮。
- ✓ 按“MODE”键切换模式，确保红色指示灯（“MODE LED”指示灯）常亮。
- ✓ 向前推动摇杆，机器人向前运动，反之机器人向后运动，机器人的运动速度与推动摇杆的幅度成正比，最大 1m/s；向左推动摇杆，机器人将向左转，反之机器人右转，机器人的转速与推动摇杆的幅度成正比，最大 3rad/s。

2.3. 上位机串口控制

2.3.1. 概述

机器人底盘采用 RS232 与上位机进行通信。使用 RS232-USB 接口线与上位机（如 PC/工控机等）相连，按照预定义的协议规则，向下位机发送指令，控制移动平台进行运动。

2.3.2. 串口使用说明



按下启动开关（此时，启动开关的环形指示灯会变亮），确保剩余大量大于 10%，确保急停开关没有被按下，确保三挡钥匙开关旋转到中间状态,确保机器人周边 10cm 内没有障碍物。

2.3.3. 协议说明

2.3.3.1. 通信协议概要

机器人底盘与上位机采用问答方式通信，上位机发出指令包，机器人返回应答包。

通信方式采用串行异步方式，8 位数据位，1 位停止位，无奇偶校验位，波特率 115200。

2.3.3.2. 设置机器人速度指令

2.3.3.2.1. 指令包

名称	Header	Length	MsgId	PARAM	Check_Sum
长度	3 Byte	1 Byte	1 Byte	6 Byte	2 Byte

✧Header：即帧头，标志着一帧的开始；固定为 0xFF0301。

✧Length：整帧的长度，即 Header+Length+MsgId+PARAM+Check_Sum 的长度，此处为 0x0D。

✧MsgId：设置机器人速度的 MsgId 为 0x03。

✧PARAM：机器人目标速度，格式为：

PARAM			
线速度		角速度	
线速度方向	线速度大小	角速度方向	角速度大小
1 Byte	2 Byte	1 Byte	2 Byte

线速度方向：

1 Byte；0x00 表示机器人向前运动，0x01 表示机器人向后运动。

线速度大小：

2 Byte，高位在前，线速度的绝对值，单位为 mm/s，最大为 1000mm/s

角速度方向：

1 Byte；0x00 表示机器人向左运动，0x01 表示机器人向右运动。

角速度大小：

2 Byte，高位在前，角速度的绝对值，单位为 mrad/s，最大为 3rad/s

✧Check_Sum：Length+MsgId+PARAM 的 CRC16 校验和，低位在前，计算方法见 2.5.4。

2.3.3.2.2. 应答包

名称	Header	Length	PayLoad	Check_Sum
长度	3 Byte	1 Byte	8 Byte	2 Byte

✧Header：即帧头，标志着一帧的开始；固定为 0xFF0301。

✧Length：整帧的长度，即 Header+Length+Payload+Check_Sum 的长度，此处为 0x0E。

✧Payload：返回值的有效载荷，格式如下：

PayLoad		解释
MsgId	PARAM	
1 Byte	7 Byte	
0x82	00 00 00 00 00 00 00	指令包校验码错误
0x03	00 00 00 00 00 00 00	指令包指令开始执行
0x86	XX 00 00 00 00 00 00	有电机失去联系，XX 为四个电机驱动器的 ID 之和
0x90	00 00 00 00 00 00 XX	低电量报警，XX 为剩余电量

✧Check_Sum:Length+MsgId+PARAM 的 CRC16 校验和，低位在前，计算方法见 2.5.4。

2.3.3.3. 读取机器人速度和电池剩余电量的指令

2.3.3.3.1. 指令包

名称	Header	Length	MsgId	Check_Sum
长度	3 Byte	1 Byte	1 Byte	2 Byte

✧Header：即帧头，标志着一帧的开始；固定为 0xFF0301。

✧Length：整帧的长度，即 Header+Length+MsgId+Check_Sum 的长度，此处为 0x07。

✧MsgId：读取机器人速度和剩余电量的 MsgId 为 0x04。

✧Check_Sum:Length+MsgId 的 CRC16 校验和，低位在前，计算方法见 2.5.4。

2.3.3.3.2. 应答包

名称	Header	Length	PayLoad	Check_Sum
长度	3 Byte	1 Byte	8 Byte	2 Byte

✧Header：即帧头，标志着一帧的开始；固定为 0xFF0301。

✧Length：整帧的长度，即 Header+Length+Payload+Check_Sum 的长度，此处为 0x0E。

✧Payload：返回值的有效载荷，格式如下：

<table>
<tr><th colspan="6">PayLoad</th><th>解释</th></tr>
<tr><td>MsgId</td><td colspan="5">PARAM</td><td></td></tr>
<tr><td>1 Byte</td><td colspan="5">7 Byte</td><td></td></tr>
<tr><td>0x82</td><td colspan="5">00 00 00 00 00 00 00</td><td>指令包校验码错误</td></tr>
<tr><td rowspan="3">0x04</td><td>线速度方向</td><td>线速度大小</td><td>角速度方向</td><td>角速度大小</td><td>剩余电池电量</td><td rowspan="2"></td></tr>
<tr><td>1 Byte</td><td>2 Byte</td><td>1 Byte</td><td>2 Byte</td><td>1 Byte</td></tr>
<tr><td>0x00:机器人向前运动；0x01:机器人向后运动</td><td>高位在前，线速度的绝对值，单位为 mm/s</td><td>0x00：机器人向左运动；0x01：机器人向右运动</td><td>高位在前，标示角速度的绝对值，单位为 mrad/s</td><td>剩余电量百分比</td><td></td></tr>
<tr><td>0x86</td><td colspan="5">XX 00 00 00 00 00 00</td><td>有电机失去联系，XX 为四个电机驱动器的 ID 之和</td></tr>
<tr><td>0x90</td><td colspan="5">00 00 00 00 00 00 XX</td><td>低电量报警，XX 为剩余电量</td></tr>
</table>

✧Check_Sum：Length+Payload 的 CRC16 校验和，低位在前，计算方法见 2.5.4。

2.3.3.4. 读取本地 LORA 参数

2.3.3.4.1. 指令包

名称	Header	Length	MsgId	Check_Sum
长度	3 Byte	1 Byte	1 Byte	2 Byte

✧Header：即帧头，标志着一帧的开始；固定为 0xFF0301。

✧Length：整帧的长度，即 Header+Length+MsgId+Check_Sum 的长度，此处为 0x07。

✧MsgId：读取机器人速度和剩余电量的 MsgId 为 0x05。

✧Check_Sum:Length+MsgId 的 CRC16 校验和，低位在前，计算方法见 2.5.4。

2.3.3.4.2. 应答包

名称	Header	Length	PayLoad	Check_Sum
长度	3 Byte	1 Byte	8 Byte	2 Byte

✧Header：即帧头，标志着一帧的开始；固定为 0xFF0301。

✧Length：整帧的长度，即 Header+Length+Payload+Check_Sum 的长度，此处为 0x0E。

✧Payload：返回值的有效载荷，格式如下：

PayLoad		解释
MsgId	PARAM	
1 Byte	7 Byte	
0x82	00 00 00 00 00 00 00	指令包校验码错误
0x05	详见 2.5.5	
0x86	00 00 00 00 00 00 00	本地 LORA 失去联系
0x90	00 00 00 00 00 00 xx	低电量报警，XX 为当前剩余电量

✧Check_Sum：Length+Payload 的 CRC16 校验和，低位在前，计算方法见 2.5.4。

2.3.3.5. 设置本地 LORA

2.3.3.5.1. 指令包

名称	Header	Length	MsgId	PARAM	Check_Sum
长度	3 Byte	1 Byte	1 Byte	6 Byte	2 Byte

✧Header：即帧头，标志着一帧的开始；固定为 0xFF0301。

✧Length：整帧的长度，即 Header+Length+MsgId+PARAM+Check_Sum 的长度，此处为 0x0E。

✧MsgId：读取机器人速度和剩余电量的 MsgId 为 0x06。

✧Check_Sum:Length+MsgId 的 CRC16 校验和，低位在前，计算方法见 2.5.4。

2.3.3.5.2. 应答包

名称	Header	Length	PayLoad	Check_Sum
长度	3 Byte	1 Byte	8 Byte	2 Byte

✧Header：即帧头，标志着一帧的开始；固定为 0xFF0301。

✧Length：整帧的长度，即 Header+Length+Payload+Check_Sum 的长度，此处为 0x0E。

✧Payload：返回值的有效载荷，格式如下：

PayLoad		解释
MsgId	PARAM	
1 Byte	7 Byte	
0x82	00 00 00 00 00 00 00	指令包校验码错误
0x06	详见 2.5.5	
0x86	00 00 00 00 00 00 00	本地 LORA 失去联系

✧Check_Sum：Length+Payload 的 CRC16 校验和，低位在前，计算方法见 2.5.4。

2.3.3.6. 读取目标 LORA 参数

2.3.3.6.1. 指令包

名称	Header	Length	MsgId	Check_Sum
长度	3 Byte	1 Byte	1 Byte	2 Byte

✧Header：即帧头，标志着一帧的开始；固定为 0xFF0301。

✧Length：整帧的长度，即 Header+Length+MsgId+Check_Sum 的长度，此处为 0x07。

✧MsgId：读取机器人速度和剩余电量的 MsgId 为 0x07。

✧Check_Sum:Length+MsgId 的 CRC16 校验和，低位在前，计算方法见 2.5.4。

2.3.3.6.2. 应答包

名称	Header	Length	PayLoad	Check_Sum
长度	3 Byte	1 Byte	8 Byte	2 Byte

✧Header：即帧头，标志着一帧的开始；固定为 0xFF0301。

✧Length：整帧的长度，即 Header+Length+Payload+Check_Sum 的长度，此处为 0x0E。

✧Payload：返回值的有效载荷，格式如下：

PayLoad		解释
MsgId	PARAM	
1 Byte	7 Byte	
0x82	00 00 00 00 00 00 00	指令包校验码错误
0x07	详见 2.5.5	

✧Check_Sum：Length+Payload 的 CRC16 校验和，低位在前，计算方法见 2.5.4。

2.3.3.7. 设置目标 LORA 参数

2.3.3.7.1. 指令包

名称	Header	Length	MsgId	PARAM	Check_Sum
长度	3 Byte	1 Byte	1 Byte	6 Byte	2 Byte

✧Header：即帧头，标志着一帧的开始；固定为 0xFF0301。

✧Length：整帧的长度，即 Header+Length+MsgId+PARAM+Check_Sum 的长度，此处为 0x08。

✧MsgId：读取机器人速度和剩余电量的 MsgId 为 0x05。

✧Check_Sum:Length+MsgId 的 CRC16 校验和，低位在前，计算方法见 2.5.4。

2.3.3.7.2. 应答包

名称	Header	Length	PayLoad	Check_Sum
长度	3 Byte	1 Byte	8 Byte	2 Byte

✧Header：即帧头，标志着一帧的开始；固定为 0xFF0301。

✧Length：整帧的长度，即 Header+Length+Payload+Check_Sum 的长度，此处为 0x0E。

✧Payload：返回值的有效载荷，格式如下：

PayLoad		解释
MsgId	PARAM	
1 Byte	7 Byte	
0x82	00 00 00 00 00 00 00	指令包校验码错误
0x05	详见 2.5.5	
0x86	00 00 00 00 00 00 00	本地 LORA 失去联系

✧Check_Sum：Length+Payload 的 CRC16 校验和，低位在前，计算方法见 2.5.4。

2.3.4. CRC16 的计算

C 语言 CRC 生成函数如下 **程序** 所示。所有的可能的 CRC 值都被预装在两个数组中，当计算报文内容时可以简单的索引即可。一个数组含有 16 位 CRC 域的所有 256 个可能的高位字节，另一个数组含有低位字节的值。

注意：此函数内部执行高/低 CRC 字节的交换。此函数返回的是已经经过交换的 CRC 值。也就是说，从该函数返回的 CRC 值可以直接放置于报文用于发送。

函数使用两个参数：

unsigned char *puchMsg; 指向含有用于生成 CRC 的二进制数据报文缓冲区的指针。

unsigned short usDataLen; 报文缓冲区的字节数。

程序 CRC16 生成函数程序清单

```
/* 高位字节的 CRC 值*/
static unsigned char auchCRCHi[] = {
0x00, 0xC1, 0x81, 0x40, 0x01, 0xC0, 0x80, 0x41, 0x01, 0xC0, 0x80, 0x41, 0x00, 0xC1, 0x81,
0x40, 0x01, 0xC0, 0x80, 0x41, 0x00, 0xC1, 0x81, 0x40, 0x00, 0xC1, 0x81, 0x40, 0x01, 0xC0,
0x80, 0x41, 0x01, 0xC0, 0x80, 0x41, 0x00, 0xC1, 0x81, 0x40, 0x00, 0xC1, 0x81, 0x40, 0x01,
0xC0, 0x80, 0x41, 0x00, 0xC1, 0x81, 0x40, 0x01, 0xC0, 0x80, 0x41, 0x01, 0xC0, 0x80, 0x41,
0x00, 0xC1, 0x81, 0x40, 0x01, 0xC0, 0x80, 0x41, 0x00, 0xC1, 0x81, 0x40, 0x00, 0xC1, 0x81,
0x40, 0x01, 0xC0, 0x80, 0x41, 0x00, 0xC1, 0x81, 0x40, 0x01, 0xC0, 0x80, 0x41, 0x01, 0xC0,
0x80, 0x41, 0x00, 0xC1, 0x81, 0x40, 0x00, 0xC1, 0x81, 0x40, 0x01, 0xC0, 0x80, 0x41, 0x01,
0xC0, 0x80, 0x41, 0x00, 0xC1, 0x81, 0x40, 0x01, 0xC0, 0x80, 0x41, 0x00, 0xC1, 0x81, 0x40,
0x00, 0xC1, 0x81, 0x40, 0x01, 0xC0, 0x80, 0x41, 0x01, 0xC0, 0x80, 0x41, 0x00, 0xC1, 0x81,
0x40, 0x00, 0xC1, 0x81, 0x40, 0x01, 0xC0, 0x80, 0x41, 0x00, 0xC1, 0x81, 0x40, 0x01, 0xC0,
0x80, 0x41, 0x01, 0xC0, 0x80, 0x41, 0x00, 0xC1, 0x81, 0x40, 0x00, 0xC1, 0x81, 0x40, 0x01,
0xC0, 0x80, 0x41, 0x01, 0xC0, 0x80, 0x41, 0x00, 0xC1, 0x81, 0x40, 0x01, 0xC0, 0x80, 0x41,
0x00, 0xC1, 0x81, 0x40, 0x00, 0xC1, 0x81, 0x40, 0x01, 0xC0, 0x80, 0x41, 0x00, 0xC1, 0x81,
0x40, 0x01, 0xC0, 0x80, 0x41, 0x01, 0xC0, 0x80, 0x41, 0x00, 0xC1, 0x81, 0x40, 0x01, 0xC0,
0x80, 0x41, 0x00, 0xC1, 0x81, 0x40, 0x00, 0xC1, 0x81, 0x40, 0x01, 0xC0, 0x80, 0x41, 0x01,
```

```
0xC0, 0x80, 0x41, 0x00, 0xC1, 0x81, 0x40, 0x00, 0xC1, 0x81, 0x40, 0x01, 0xC0, 0x80, 0x41,
0x00, 0xC1, 0x81, 0x40, 0x01, 0xC0, 0x80, 0x41, 0x01, 0xC0, 0x80, 0x41, 0x00, 0xC1, 0x81,
0x40
} ;
/* 低位字节的 CRC 值*/
static char auchCRCLo[] = {
0x00, 0xC0, 0xC1, 0x01, 0xC3, 0x03, 0x02, 0xC2, 0xC6, 0x06, 0x07, 0xC7, 0x05, 0xC5, 0xC4,
0x04, 0xCC, 0x0C, 0x0D, 0xCD, 0x0F, 0xCF, 0xCE, 0x0E, 0x0A, 0xCA, 0xCB, 0x0B, 0xC9, 0x09,
0x08, 0xC8, 0xD8, 0x18, 0x19, 0xD9, 0x1B, 0xDB, 0xDA, 0x1A, 0x1E, 0xDE, 0xDF, 0x1F, 0xDD,
0x1D, 0x1C, 0xDC, 0x14, 0xD4, 0xD5, 0x15, 0xD7, 0x17, 0x16, 0xD6, 0xD2, 0x12, 0x13, 0xD3,
0x11, 0xD1, 0xD0, 0x10, 0xF0, 0x30, 0x31, 0xF1, 0x33, 0xF3, 0xF2, 0x32, 0x36, 0xF6, 0xF7,
0x37, 0xF5, 0x35, 0x34, 0xF4, 0x3C, 0xFC, 0xFD, 0x3D, 0xFF, 0x3F, 0x3E, 0xFE, 0xFA, 0x3A,
0x3B, 0xFB, 0x39, 0xF9, 0xF8, 0x38, 0x28, 0xE8, 0xE9, 0x29, 0xEB, 0x2B, 0x2A, 0xEA, 0xEE,
0x2E, 0x2F, 0xEF, 0x2D, 0xED, 0xEC, 0x2C, 0xE4, 0x24, 0x25, 0xE5, 0x27, 0xE7, 0xE6, 0x26,
0x22, 0xE2, 0xE3, 0x23, 0xE1, 0x21, 0x20, 0xE0, 0xA0, 0x60, 0x61, 0xA1, 0x63, 0xA3, 0xA2,
0x62, 0x66, 0xA6, 0xA7, 0x67, 0xA5, 0x65, 0x64, 0xA4, 0x6C, 0xAC, 0xAD, 0x6D, 0xAF, 0x6F,
0x6E, 0xAE, 0xAA, 0x6A, 0x6B, 0xAB, 0x69, 0xA9, 0xA8, 0x68, 0x78, 0xB8, 0xB9, 0x79, 0xBB,
0x7B, 0x7A, 0xBA, 0xBE, 0x7E, 0x7F, 0xBF, 0x7D, 0xBD, 0xBC, 0x7C, 0xB4, 0x74, 0x75, 0xB5,
0x77, 0xB7, 0xB6, 0x76, 0x72, 0xB2, 0xB3, 0x73, 0xB1, 0x71, 0x70, 0xB0, 0x50, 0x90, 0x91,
0x51, 0x93, 0x53, 0x52, 0x92, 0x96, 0x56, 0x57, 0x97, 0x55, 0x95, 0x94, 0x54, 0x9C, 0x5C,
0x5D, 0x9D, 0x5F, 0x9F, 0x9E, 0x5E, 0x5A, 0x9A, 0x9B, 0x5B, 0x99, 0x59, 0x58, 0x98, 0x88,
0x48, 0x49, 0x89, 0x4B, 0x8B, 0x8A, 0x4A, 0x4E, 0x8E, 0x8F, 0x4F, 0x8D, 0x4D, 0x4C, 0x8C,
0x44, 0x84, 0x85, 0x45, 0x87, 0x47, 0x46, 0x86, 0x82, 0x42, 0x43, 0x83, 0x41, 0x81, 0x80,
0x40
};
unsigned short CRC16(unsigned char *puchMsg, /* 用于计算 CRC 的报文*/
unsigned short usDataLen /* 报文中的字节数*/
) /* 函数以 unsigned short 类型返回 CRC */
{
unsigned char uchCRCHi = 0xFF; /* CRC 的高字节初始化*/
```

```
unsigned char uchCRCLo = 0xFF; /* CRC 的低字节初始化*/
unsigned uIndex ; /* CRC 查询表索引*/
while (usDataLen--) /* 完成整个报文缓冲区*/
{
uIndex = uchCRCLo ^ *puchMsg++; /* 计算 CRC */
uchCRCLo = uchCRCHi ^ auchCRCHi[uIndex};
uchCRCHi = auchCRCLo[uIndex];
}
return (uchCRCHi << 8 | uchCRCLo);
}
```

2.3.5. LORA PARAM 介绍

PARAM						
ADDH	ADDL	NETID	REG0	Reg1	Reg2	REG3
地址高位	地址低位	网络地址，用于区分网络； 相互通信时，应保持一致	固定为 0x63	固定为 0x00	信道	固定为 0x03

默认机器人本地 LORA 参数为：0x00 ,0x00,0x00,0x63,0x00,0x28,0x03；即地址为 0x0000；网络地址为：0x00；信道：0x28。

默认与本地LORA进行通信的LORA的参数为0x00 ,0x0A,0x00,0x63,0x00,0x28,0x03；即地址：0x000A;网络地址：0x00;信道：0x28。

2.3.6. ROS 驱动包使用说明

ROS 驱动包(ROS Driver Package)是为使用ROS 开发的用户，提供上位机与下位机通讯的驱动包，下载地址：http://www.verma-robot.com/xz。

2.3.6.1. 订阅话题：

/cmd_vel (geometry_msgs/Twist)

2.3.6.2. 发布话题：

/odom (nav_msgs/Odometry)

/battery (std_msgs::Int32)

2.3.6.3. 参数：

~port_name (str, default: /dev/ttyUSB0)

2.3.6.4. 发布的 tf 转换

odom → base_link

2.3.6.5. 权限

2.3.6.5.1. 查看端口名称：

```
ls -l /dev/ttyUSB*
```

假设返回值为：/dev/ttyUSB0

2.3.6.5.2. 修改端口权限：

sudo chmod 666 /dev/ttyUSB0

2.3.6.6. 修改启动文件 （bringup.launch）

```
<launch>
    <node pkg="scholar_bringup" name="scholar_bringup"  type="scholar_bringup"
output="screen"  required="true" >
        <param name="port_name" value="/dev/ttyUSB0" />
    </node>
</launch>
```

2.3.6.7. 启动机器人底盘

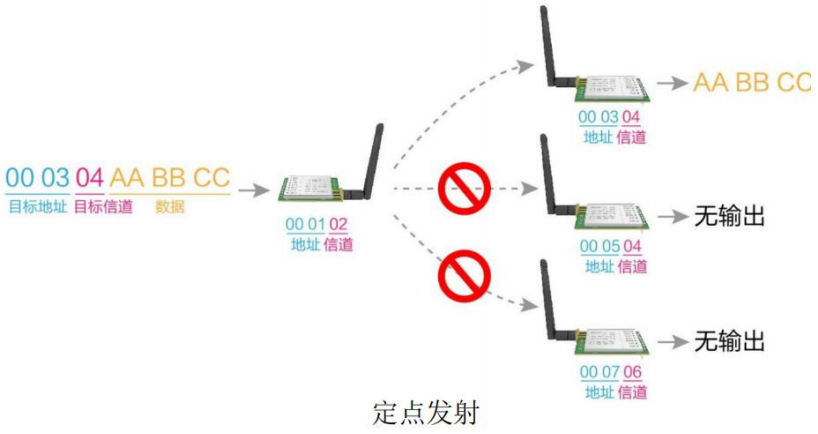
```
        roslaunch scholar_bringup bringup.launch
```

2.4. LORA 远程控制

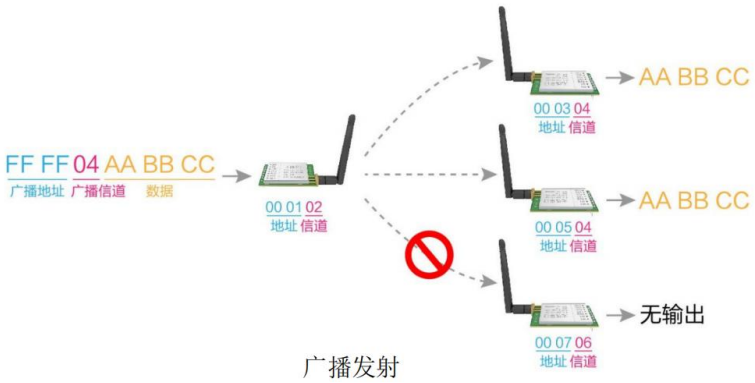
2.4.1. 概述

机器人底盘集成了 SEMTECH 公司的 SX1262 射频-USART 芯片，采用全新一代的 LoRa 扩频技术，最远通信距离可达 **5000m**（晴朗空旷环境，最大功率，天线增益 5dBi，高度 2m）；空中传输速率默认 2.4Kbps，发射功率 30dBm。

支持定点发射和广播发射（机器人默认采用定点发送方式；本机地址为 0x0000，本机信道 0x28；通信目标地址为 0x000A,目标信道为 0x28）

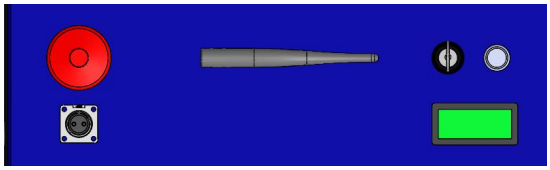


定点发射



广播发射

2.4.2. LORA 使用说明



按下启动开关（此时，启动开关的环形指示灯会变亮），确保剩余大量大于 10%，确保急停开关没有被按下，确保三挡钥匙开关旋转到右侧状态,确保机器人周边 10cm 内没有障碍物。

2.4.3. LORA 通信规则

LORA 模式下,只支持 LORA 与机器人底盘之间的设置机器人速度指令与读取机器人速度和电池剩余电量的指令；通信的协议规则与串口规则一致

3. 使用注意事项

3.1. 提醒

- ✓ 操控机器人底盘时避免速度过快，引起碰撞 。
- ✓ 请勿将机器人底盘倒置或掉落。
- ✓ 非专业人员，请不要私自拆卸 。
- ✓ 不使用非原厂标配的充电器
- ✓ 不要在有水的地方，存在腐蚀性、易燃性气体的环境内和靠近可燃性物质的地方使用。
- ✓ 不要放置在加热器等发热体周围

3.2. 电池安全

- ✓ 请在有人看护的状态下充电，若人员离开请拔掉充电插头。
- ✓ 正常充电时，充电指示灯为红色，当转为绿色时为充满 。
- ✓ 停止充电时，应先拔下插头，然后取下电池端插头 。
- ✓ 产品长期不用，需三个月至半年补充一次电 。
- ✓ 充电器应放在通风干燥的环境中使用。