Autolabor Pro1 (AP1)

机器人移动平台

Version: 2020-07-20

产品介绍

产品概述

Autolabor Pro1(AP1)机器人移动开发平台是一款集模块化、简洁化、可定制为一身的机器人底盘，适用于机器人教育培训、科学研究和产品开发等。具有通过性强、负载能力大、精度高、动力足、续航长和扩展性高等特点，可跨平台开发，支持多种应用场景。

主要特性

- 可靠耐用，操作简单，无需组装，开机直接使用
- 高通过性，爬坡能力十足，续航能力强
- 高精度工业级编码器，定位精准
- 高负载，负重可达50KG
- 适应场景广泛，室内外均可使用
- 兼容多个系统，支持多种开发平台、控制方式
- 支持ROS开发，提供ROS驱动包与运动模型
- 雷达、摄像头、惯导等多种传感器随意组合搭配，拆装方便
- 使用文档齐全，配备后续用户支持，并不断更新开发教程

产品清单

名称	数量	备注
AP1移动平台	1辆	
AP1充电器	1个	
AP1控制手柄	1套	含电池
串口数据线	1根	
20cm支架	2套	
30cm支架	2套	
辅助电源供电线	3个	
配套工具	1套	

安全守则

使用提醒

- 产品边缘锋利，请小心接触，避免划伤
- 产品表面为金属材质，请勿与电路板直接接触
- 操控平台时避免速度过快，引起碰撞
- 搬运时以及设置作业时，请勿落下或倒置
- 非专业人员，请不要私自拆卸
- 不使用非原厂标配的电池、电源、充电座
- 运行时请勿用手触碰
- 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
- 不要放置在加热器或者大型卷线电阻器等发热体周围
- 切勿将电机直接与商用电源连接

电池安全

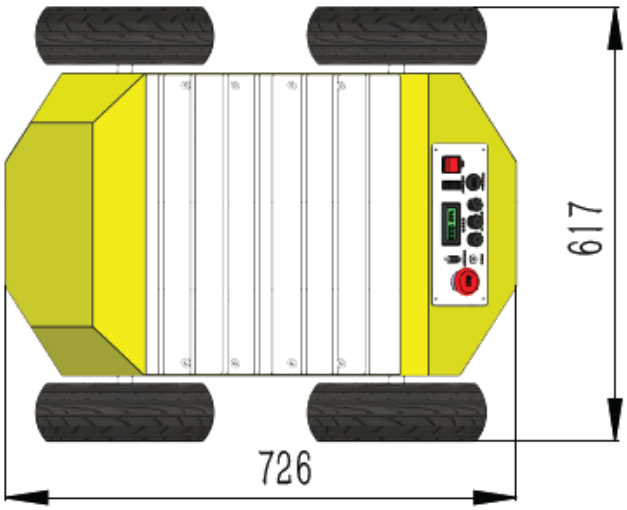
- 请在有人看护的状态下充电，若人员离开请拔掉充电插头
- 充电器在充电工作时，会向外界散发一定的热量，充电器与产品应放在通风干燥的环境中使用
- 正常充电时，充电指示灯为红色，当转为绿色时为充满
- 停止充电时，应先拔下插头，然后取下电池端插头
- 产品长期不用，需三个月至半年补充一次电
- 产品电池不可将电完全用完，否则会严重受损，容易造成不可修复

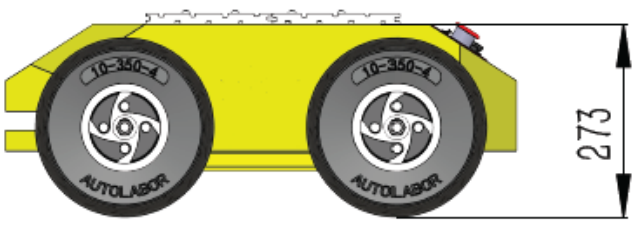
产品规格

产品参数

尺寸	726*617*273mm	净重	35kg
负载	直线50kg/原地转弯30KG	电池	24V DC锂聚合物电池
电池容量	10Ah/18Ah（选配）	续航时间	2.5h/4h
辅助电源	3*12V 1*5V	最大速度	0.8m/s
驱动方式	四驱	转向方式	差速转向
通信接口	串口	编码器精度	400线
PID控制频率	50Hz	适用地形	全地形
垂直越障能力	8cm	爬坡能力	25°
支持系统	Windows/Linux	支持平台	X86
手柄控制	20m	手柄通讯	2.4Ghz

尺寸参数





控制手柄参数

该手柄有自动休眠功能，长时间不操作，省电模块被激活后自动进入休眠模式，按下START键即可唤醒手柄。

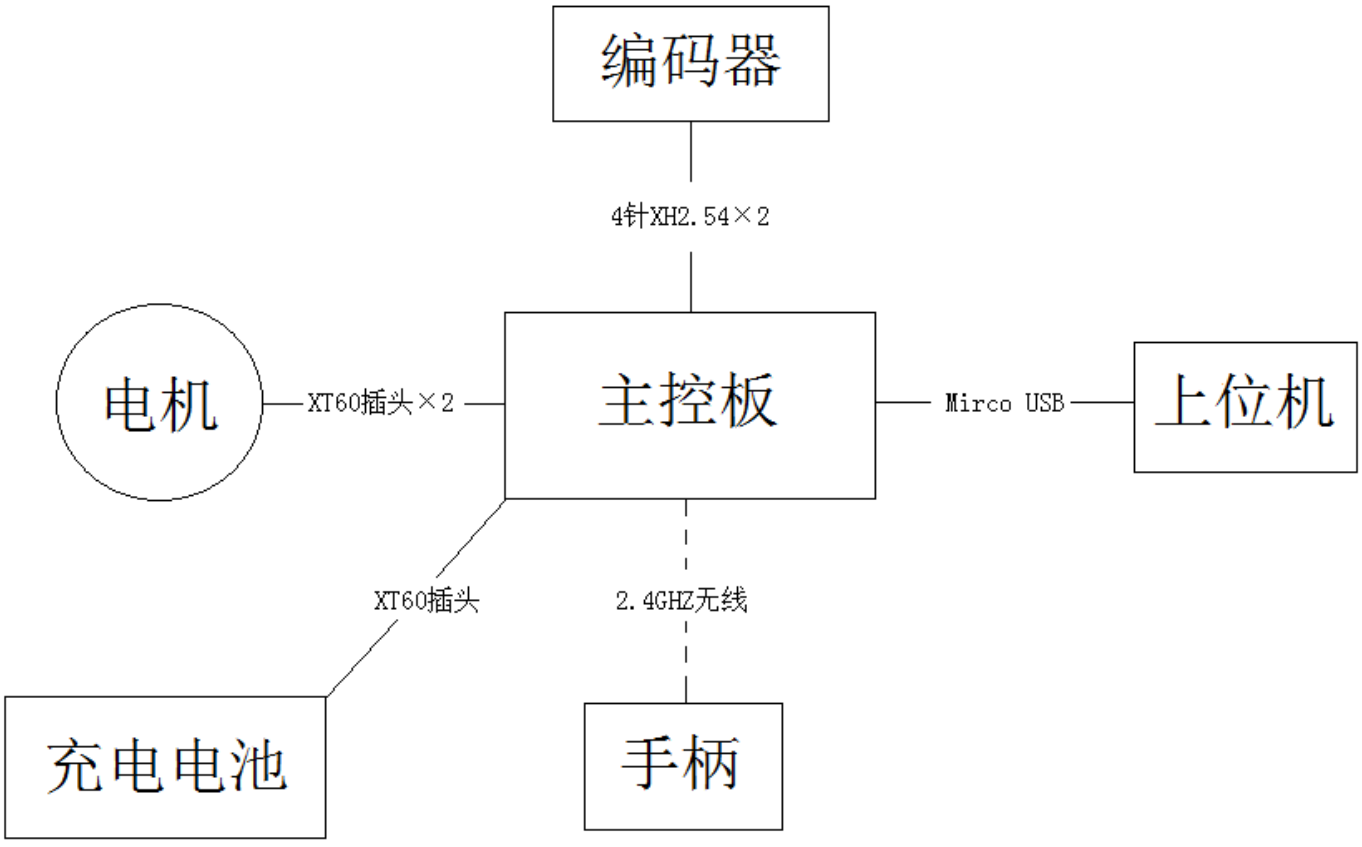
参数名称	参数内容
电池	AAA（7号）电池*2
使用时间	约10小时
无线频率	2.4GHz
接收范围	20m

控制面板使用

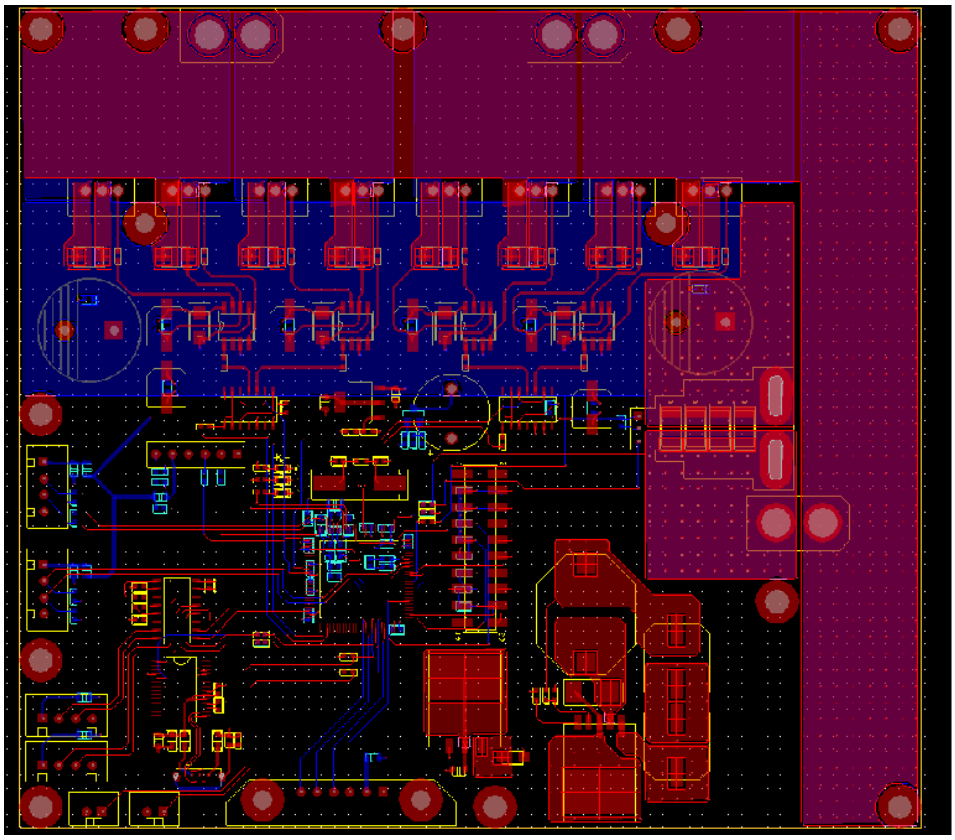
控制面板清单：

名称	个数	说明
紧急停止	1	在紧急情况下请按下该紧急停止按钮
电源	1	电源开启关闭按钮
充电口（IN）	1	为AP1充电
电量显示	1	以百分比的形式显示AP1电池电量
12V电源输出（OUT）	3	12V内置电源输出接口
5V电源输出（OUT）	1	5V内置电源输出接口
上位机控制	1	切换手柄控制与上位机控制按钮
RS232串口	1	使用串口数据线连接下位机与上位机，进行通信

硬件结构

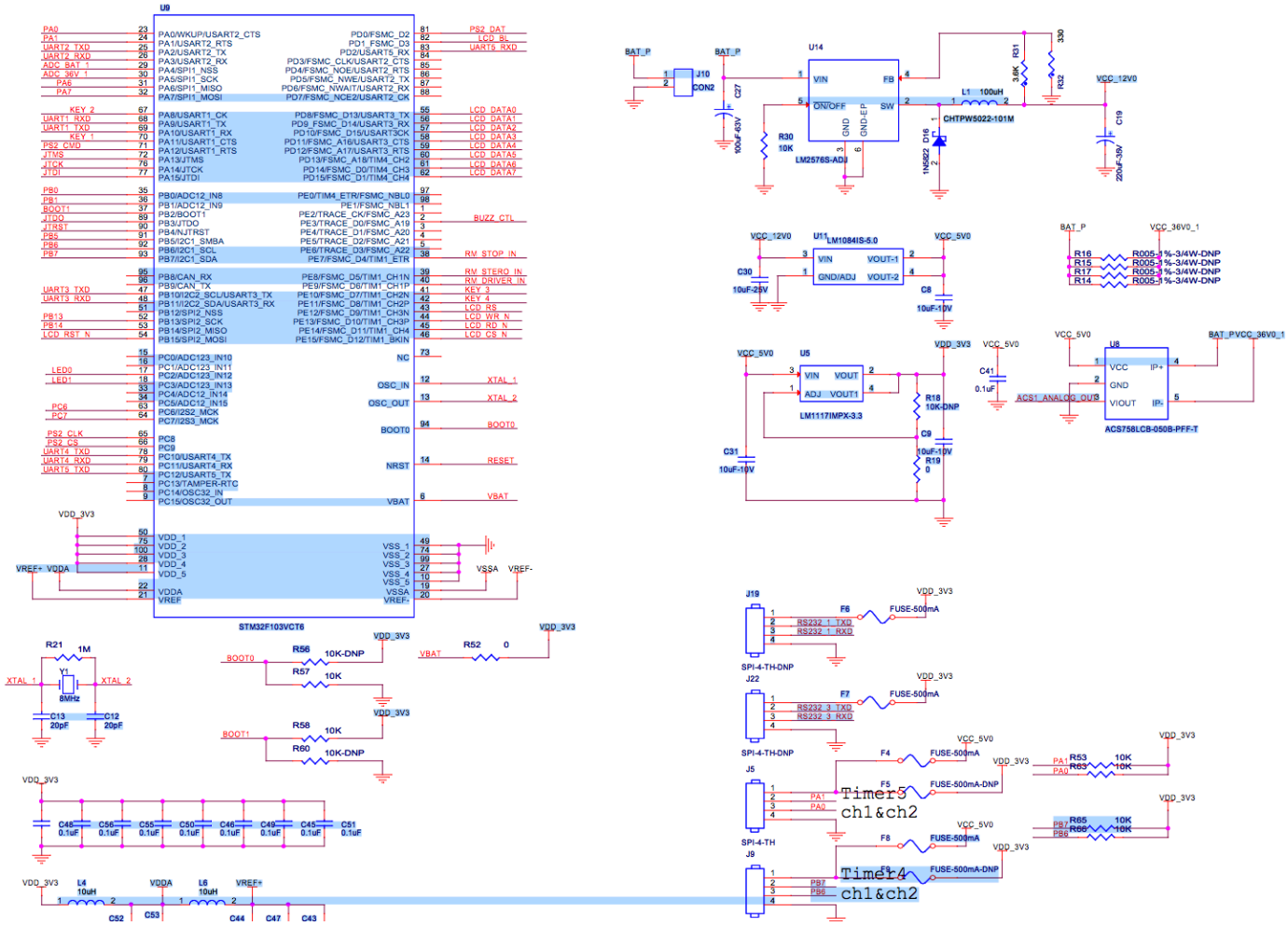


电路板图



电路原理图

下载



产品使用

控制方式

AP1可以使用手柄控制和上位机控制两种方式来控制，下面将分别概述这两种控制方式。

手柄控制

概述

我们配送一个控制手柄用于控制AP1移动。

使用说明

AP1出厂默认控制模式为手柄控制，在控制面板中可切换控制模式，将AP1控制面板中的上位机控制按钮置于关闭状态切换到手柄控制模式，即可操作手柄的按键控制移动平台。

手柄操作步骤

1. 打开手柄电源，切换至ON模式，手柄顶部POWER指示灯亮起
2. 使用MODE键开启控制模式，顶部MODE LED指示灯为红色时，才能使用手柄进行控制
3. 先按住手柄顶部两个【1】键，然后按照下文使用说明操作

手柄按键说明

我们对于AP1的手柄控制设计有几种速度档位，如下表。

档位	0档	1档	2档	3档
速度	25%	50%	75%	100%

按键	功能	备注
左区4控制键	前/后/左/右控制运动方向	
SELECT/最低速度档	设置运动速度为最低档位	设置速度档位为0档
START/最高速度档	设置运动速度为最高档位	设置速度档位为3档
左侧摇杆键	控制AP1的运动方向和速度	使用摇杆灵活的控制运动角度与速度，不必受限于正前/后/左/右，控制摇杆推动的力度来控制速度档位的调节，将摇杆向前方直接推到低，则直接是3档，向后方直接推到底，则直接设置为0档
右侧摇杆键	无功能	现阶段暂未使用
右区控制键-上	线速度增加	调整速度档位，每按一下，速度提高一档，最高不得高于最高档，如3档
右区控制键-下	线速度减少	每按一下，速度降低一档，最低不得高于最底档，如0档
右区控制键-左/右	角速度增加/减少	同线速度
顶部4键	无功能	现阶段暂未使用

上位机控制

概述

使用AP1配送的串口数据线与上位机相连，按照预定义的协议规则，向下位机（底层硬件平台）发送指令，控制移动平台。

使用说明

将AP1控制面板中的上位机控制按钮置于开启状态，使用接口线将下位机与上位机连接起来，向下位机发送指令。

指令发送方式

用户可按照自己的开发场景，选择不同的方式发送指令：

- 使用**串口调试助手**直接向下位机发送指令
- 基于ROS开发可使用我们提供的**ROS驱动包**与下位机通信发送接收指令

协议规则

协议概述

本协议是一种用于AP1上位机与下位机之间通信的自定义通信协议，波特率为115200，以16进制格式传输。上位机向下位机发送请求（commands），下位机在收到来自上位机的请求后，作出相应的反应并返回应答（feedback）至上位机，帧结构说明如下。

帧结构说明

数据帧分为5个部分：起始标志位/帧头（Header），数据长度（Length），序列号（Sequence），有效载荷数据（Payload），检验码（Checksum）。

数据帧结构如下表所示：

Name	Header	Length	Sequence	Payload	Checksum
Size	2 Byte	1 Byte	1 Byte	N Bytes	1 Byte

起始标志位/帧头（Header）

起始标志位，即为我们常说的帧头，以固定不变的“55AA”作为起始标志位，标志着一帧的开始。

数据长度（Length）

数据长度，其值表示数据包Payload的长度。

序列号（Sequence）

帧的序列号从0开始，范围为0~255，消息的发送端每发送一个帧将该字段的值加1，接收端可以根据该字段是否连续，判断是否有丢包的情况发生。

有效载荷数据（Payload）

有效载荷数据，即实际数据内容，考虑到数据传输效率与可扩展性，本协议将Payload的长度设计为非固定长度，可适应不同的消息类型。

Payload分为两部分：MsgID和PARAM。

Payload	
MsgID	PARAM
指令的类型	指令的内容

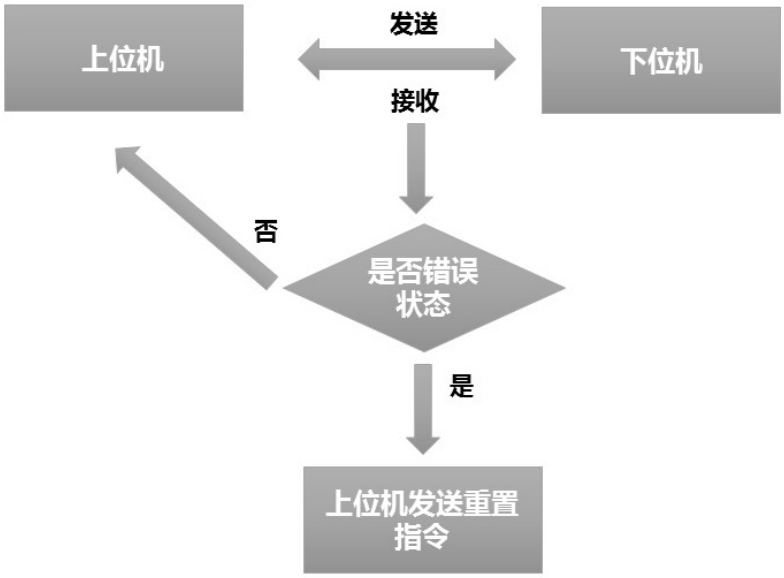
Payload指令如下表所示:

Name	Type	MsgID	PARAM	Description

<table>
<tr><td rowspan="4">车轮速度指令</td><td>发送</td><td>01</td><td>00
00</td><td>00
00</td><td>00
00</td><td>00
00</td><td rowspan="2">AP1是2电机4轮驱动模式，左/右两侧的两个轮子各由一个电机驱动。因此左边的两个轮子的速度是相同的，在发送左/右轮速度时，仅需发送一个轮子的速度，每个轮速是2个字节
PARAM中前2个字节设置左轮的速度，接着2个字节设置右轮的速度，后面4个字节全部置0，用于后续拓展四轮控制
当我们改变车轮的速度的时候，其实是改变车轮编码器的计数，例如给左轮一个0.1m/s前进的速度，即给出的指令是0004，则需要通过计算将车轮速度转化为编码器的计数，后面会讲到如何计算</td></tr>
<tr><td>左轮</td><td>右轮</td><td colspan="2">空字段，用于后续扩展</td></tr>
<tr><td>接收</td><td>01</td><td>00
04</td><td>00
00</td><td>00
00</td><td>00
00</td><td rowspan="2">返回的数据是当前时刻车轮编码器的累计计数</td></tr>
<tr><td>左轮</td><td>右轮</td><td colspan="2">空字段，用于后续扩展</td></tr>
<tr><td>电池电量指令</td><td>发送</td><td>02</td><td colspan="4">00</td><td>请求电池电量</td></tr>
<tr><td></td><td>接收</td><td>02</td><td colspan="4">32</td><td>返回的数据为0~100电量区间的16进制表达，例如电量为50，则返回32</td></tr>
<tr><td rowspan="3">重置状态指令</td><td>发送</td><td>05</td><td colspan="4">00</td><td>注：当上位机在发送指令后，接收到错误状态信息FF时，必须发送重置指令，重置后才能恢复对下位机的控制</td></tr>
<tr><td rowspan="2">接收</td><td>05</td><td colspan="4">01</td><td>返回操作成功</td></tr>
<tr><td>05</td><td colspan="4">00</td><td>返回操作失败</td></tr>
<tr><td rowspan="3">清除编码器计数指令</td><td>发送</td><td>06</td><td colspan="4">00</td><td>对编码器计数清零</td></tr>
<tr><td rowspan="2">接收</td><td>06</td><td colspan="4">01</td><td>返回操作成功</td></tr>
<tr><td>06</td><td colspan="4">00</td><td>返回操作失败</td></tr>
<tr><td></td><td></td><td>FF</td><td colspan="4">01</td><td>电池没电</td><td>当上位机向发送</td></tr>
</table>

错误状态指令	接收	FF	02	电流过大	请求，下位机发生错误无法执行指令时，会向上位机返回相应的错误信息
		FF	03	串口通信故障	
		FF	04	车轮卡死	

错误状态处理流程图：



检验码(Checksum)

为保证上位机与下位机所传输数据的无误性与一致性，本协议采用异或（XOR）校验的方式，根据具体发送的指令生成异或校验码，校验的数据包括帧头55AA，用户可以使用在线的异或校验计算器来计算，如下图所示**计算结果（Hex）**即时我们所需的异或校验码。

指令说明：

注意：

- 在使用串口发送指令时，底层超过 **1s** 没有接收到上位机的指令，会向上位机返回错误状态码FF (error状态:连接超时），此时用户如果想要重新控制车辆，需要发送Reset命令
- 在调试发送指令时，先发一条Reset命令，然后接着发指令，如:

```
0x 55 AA 02 02 05 00 FA
0x 55 AA 09 00 01 00 04 00 00 00 00 00 00 F3
0x 55 AA 02 02 05 00 FA
0x 55 AA 09 00 01 00 04 00 00 00 00 00 00 F3
...
```

注意：指令与指令之间的间隔时间不能超过1s。

串口测试速度指令样例

左转

```
55 AA 02 02 05 00 FA 55 AA 09 00 01 00 00 00 04 00 00 00 00 F3
```

右转

```
55 AA 02 02 05 00 FA 55 AA 09 00 01 00 04 00 00 00 00 00 00 F3
```

前进

```
55 AA 02 02 05 00 FA 55 AA 09 00 01 00 04 00 04 00 00 00 00 F7
```

后退

```
55 AA 02 02 05 00 FA 55 AA 09 00 01 FF FA FF FA 00 00 00 00 F7
```

下面列出几个常用的控制指令示例：

控制车轮速度指令

假设请求左车轮速度为0.1m/s，传输数据帧内容为：

```
0x 55 AA 09 00 01 00 04 00 00 00 00 00 00 F3
```

Header	Length	Sequence	Payload					Checksum
			MsgID	PARAM				
55 AA	09	00	01	00 04	00 00	00 00	00 00	F3
起始标志位	有效数据的长度为9个字节	第一个指令	控制车轮速度指令	左车轮	右车轮	固定不变为0		异或校验码

如上文所说，通常我们所说的速度是以m/s为单位的速度，而指令中车轮速度的参数实际上是单位时间内期望编码器的计数，在此需要将指令的速度V（m/s）结合AP1下位机的物理参数进行计算，下面是车轮速度换算方法。

AP1下位机物理参数如下表：

名称	参数	说明
wheel_diameter	0.25	车轮直径，单位：米
encoder_resolution	1600	编码器转一圈产生的脉冲数
PID_RATE	50	PID调节PWM值的频率

假设我们给AP1左轮V=0.1m/s的速度（右轮速度的计算指令与此相同），计算方法如下：

车轮转动一圈，移动的距离为轮子的周长：

```
distance_one_round
=wheel_diameter*π
=wheel_diameter*3.1415926
=0.25*3.1415926
```

车轮转动一圈，编码器产生的脉冲数为：

```
wheel_encoder_resolution
=1*encoder_resolution
=1*1600
=1600
```

注：编码器与车轮连接在一起，车轮转1圈，编码器转1圈；编码器的脉冲数可以理解为编码器计数，编码器自转一圈计数1600，则车轮转一圈编码器计数为1600。

所以AP1每移动1米产生脉冲数（编码器的计数）为：

```
ticks_per_meter
=(1/distance_one_round)*wheel_encoder_resolution
=1/(0.25*3.1415926)*1600
=2037.18
```

又因为PID的频率是1秒钟50次，所以指令的参数计算方法为:

```
int(V*ticks_per_meter/PID_RATE)
=int(0.1*2037.18/50)
=4
```

用户可直接使用0.1m/s速度的计算结果来对应自己期望的速度换算成相应的数值，如0.2m/s为8，1m/s为40等。

接着将计算出来的4换算为16进制为：

```
00 04
```

则速度指令的PARAM为

```
00 04 00 00 00 00 00 00
```

注：如果用户发送的速度超过了最大速度，则会按最大速度行进。

获取电池电量指令

```
0x 55 AA 02 01 02 00 FE
```

Header	Length	Sequence	Payload		Checksum
			MsgID	PARAM	
55 AA	02	01	02	00	FE
起始标志位	有效数据的长度为2个字节	第二个指令	小车的电量请求	参数为00	异或校验码

重置状态指令

当接收到下位机发送的错误状态码FF时，需要将AP1状态重置（Reset)，才能重新恢复控制，指令如下：

```
0x 55 AA 02 02 05 00 FA
```

Header	Length	Sequence	Payload		Checksum
			MsgID	PARAM	
55 AA	02	02	05	00	FA
起始标志位	有效数据的长度为2个字节	第三个指令	小车的电量请求	参数为00	异或校验码

清除编码器计数指令

```
0x 55 AA 02 03 06 00 F8
```

Header	Length	Sequence	Payload		Checksum
			MsgID	PARAM	
55 AA	02	03	06	00	F8
起始标志位	有效数据的长度为2个字节	第四个指令	请求清除编码器计数	参数为00	异或校验码

ROS驱动包(ROS Driver Package)使用

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)。

订阅的话题

/cmd_vel (geometry_msgs/Twist)
控制下位机运动的速度指令

发布的话题

/wheel_odom (nav_msgs/Odometry)
轮速里程计
/remaining_battery (std_msgs::Int32)
剩余电池电量
/current(std_msgs::Float32)
电流信息
/voltage(std_msgs::Float32)
电压信息

参数

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

串口名称

~odom_frame (str, default: odom)

odom

~base_frame (str, default: base_link)

base_link

~baud_rate (int, default: 115200)

波特率

~control_rate (int, default: 10)

控制频率

~sensor_rate (int, default: 5)

请求电池电量频率

~reduction_ratio (double, default: 1)

减速比

~encoder_resolution (double, default: 1600)

编码器的编码数

~wheel_diameter (double, default: 0.25)

车轮直径

~model_param_cw (double, default:0.8)

顺时针旋转运动模型参数　　注：2018.5.9前购买产品的用户，此参数为0.83

~model_param_acw (double, default:0.8)

逆时针旋转运动模型参数　　注：2018.5.9前购买产品的用户，此参数为0.83

~pid_rate (double, default:50.0)

PID控制周期

TF转换

odom → base_link

驱动示例

```
<launch>
    <node name="autolabor_driver" pkg="autolabor_pro1_driver" type="autolabor_pro1
_driver" output="screen">
        <param name="port_name" value="/dev/ttyUSB0" />
        <param name="odom_frame" value="odom" />
        <param name="base_frame" value="base_link" />

        <param name="baud_rate" value="115200" />
        <param name="control_rate" value="10" />
        <param name="sensor_rate" value="5" />

        <param name="reduction_ratio" value="1.0" />
        <param name="encoder_resolution" value="1600.0" />

        <param name="wheel_diameter" value="0.25" />
        <param name="model_param_cw" value="0.8" />
        <param name="model_param_acw" value="0.8" />
        <param name="pid_rate" value="50.0" />
         <param name="maximum_encoding" value="32.0" />
   </node>
</launch>
```

驱动启动方法

1. 将ROS驱动放入工作空间catkin_ws/src
2. 回到catkin_ws根目录下
3. 打开terminal
4. 执行

   ```
   source /opt/ros/kinetic/setup.bash
   ```

5. 执行

   ```
   caktin_make
   ```

6. 执行

   ```
   source devel/setup.bash
   ```

7. 执行

   ```
   roslaunch autolabor_pro1_driver driver.launch
   ```

驱动启动后，可以给小车发送速度指令。此时须将底盘架起来，车轮悬空。

速度指令发送

1. 打开一个terminal
2. 执行 rostopic list 查看有无/cmd_vel 话题
3. 如果有的话，执行：

```
rostopic pub -r 10 /cmd_vel geometry_msgs/Twist -- '[0.0,0.0,0.0]' '[0.0,0.0,1
]'
```

此时车轮会开始转动，如果想要停止的话，必须在执行命令的terminal中执行Ctrl+C，停止发送。

发送指令的详细参数，请参考ROS官网的介绍说明【Using rostopic pub】
http://wiki.ros.org/ROS/Tutorials/UnderstandingTopics

常见问题

- 如果启动的时无法找到端口，请使用以下命令，查看是否存在该目录 /dev/ttyUSB0

  ```
  ll /dev/ttyUSB0
  ```

  - 若不存在，请检查：

    1. 串口线是否插好
    2. 小车电源是否打开
    3. 如果上述都没有问题，请查询是否是其他的USB端口，如/dev/ttyUSB1或/dev/ttyUSB2

  - 如果存在目录，但启动显示权限不足，请使用以下命令进行赋权

    ```
    sudo chmod 666 /dev/ttyUSB0
    ```

ROS驱动调试助手

该驱动还为用户提供了一个调试的节点，可以使用该节点直接向驱动板发送串口指令，进行指令速度与实际运行速度的校准或调整模型等工作。

使用步骤

1. 创建并启动launch文件，示例如下

```
<launch>
    <node name="driver" pkg="autolabor_pro1_driver" type="sim_autolabor_pro1_d
river" output="screen">
        <param name="port_name" value="/dev/ttyUSB0" />
        <param name="baud_rate" value="115200" />
        <param name="control_rate" value="10" />
        <param name="pid_rate" value="50" />
    </node>
</launch>
```

2. 打开新终端，输入以下命令，打开操作界面（如下图）

```
rosrun rqt_reconfigure rqt_reconfigure
```

3. 调整 left_wheel 与 right_wheel的数值，可独立控制左/右轮转速，勾选 run_flag，即可向小车发送速度指令

注：调整的数值为一个PID控制周期下，预期编码器的变化数，详细的计算方法可参见控制车轮速度指令中的编码器计数计算方法

服务支持

在线文档请访问

http://www.autolabor.com.cn

微信客户服务平台