**基于RT-Thread的物联小车系统**

**一、任务描述及选题的意义**

智能小车是一种可以通过预先给予其特定的指令，从而使其完成预期目标的设备。智能小车可以在很大程度上减少人员的工作量，又因其自身可扩展、高适应度、高定制化的功能，所以被广泛地应用于军事、医疗、物流、家居等领域。智能小车在智能制造领域的未来发展中有着不可或缺的地位。

本物联小车系统就是智能小车的一种，其目的是针对各种用户需要的功能，提供一个可供扩展的小车底座平台系统，研发阶段采用履带小车，项目成熟后可适配各种类型的小车。

用户可将自己需要的模块安装在本物联小车上，从而实现各种各样的功能。例如在小车上安装摄像头模块，便可完成巡检安防等功能，安装货物架，便可实现运输功能等等。

基于通用性的目标，本系统完成的功能聚焦于小车的控制之上。总结来说，本系统完成的功能如下：

**1.分用户控制功能**

在本物联小车系统中，使用了自行构建的、定制化的私有云平台，可满足不同用户注册账号之后绑定自己的对应小车，实现对自己对应小车的控制。支持用户信息更改，同时也支持改变绑定小车的功能，从而实现单一用户切换控制对应小车，而每一辆小车对应的控制用户是唯一的。

**2.小车移动控制功能**

本系统使用了自行构建的、定制化的私有云平台，通过手机APP给云平台发送控制指令，嵌入式小车获取云平台控制指令，实现小车的移动，便可实现在各种各样的应用场景中对不同小车的应用。

**3.OTA升级功能**

本系统使用了自行构建的、定制化的私有云平台，本系统的私有云平台支持OTA版本管理及OTA固件下载，嵌入式小车从本系统的私有云平台获取版本信息，进行判断是否有新版本，并且进行下载更新，从而实现OTA升级功能

**二、资料搜集**

经过线上资料搜集，发现市场上有许多所谓的智能小车系统，可满足一些特定的功能等，但是所谓的智能并不智能，其大多是通过蓝牙控制的小车系统，配合一些避障功能，价格昂贵且通用性不强。而且由于是蓝牙控制，控制距离与效果十分不理想。

于是对于智能小车，更先进的方式是引入云平台技术，使用物联网技术，把嵌入式小车作为网络中的边缘设备，参与到物联网路系统中，实现更高效、控制距离更远、功能更丰富的物联小车系统，本系统就是采用物联网技术构建的物联小车系统。

关于云平台，成熟易用的共有云平台由很多，可以直接使用他们，同时也可以自行开发一个云平台（即私有云），而自行开发的云平台较为定制化，只需满足业务的特殊需求即可。物联网云平台采用的接入通讯协议一般有mqtt、http等协议，而流行的物联网云平台（如阿里云物联网平台等）一般同时支持mqtt和http协议接入。而本系统就采用了自行构建的云平台的策略，同时自行构建的云平台采用的是http通讯协议。

同时对于小车系统，电机控制算法必不可少，目前广泛使用的电机控制算法为PID控制算法，本系统就采用了PID电机控制算法。

**三、实施内容**

**1.系统整体架构图**

**2.系统功能展示**

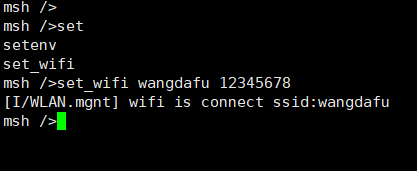
嵌入式小车载入时，显示小车ID（每个小车的唯一标识），以及小车的固件版本信息

****

若未连接wifi或者连接wifi失败，就会出现如下的配网提示信息

****

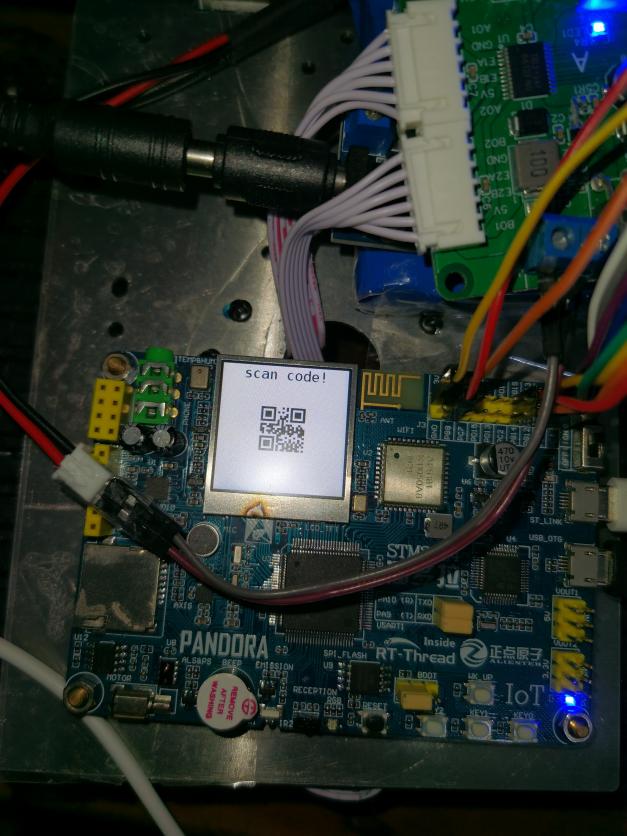
此时需要通过小车上主控开发板的串口，使用命令行进行配网（串口波特率115200），使用set\_wifi <ssid> <password> 命令进行配网。



配网成功或者已经进行过配网，并且成功连接wifi后会出现wifi连接成功提示



若小车未被用户绑定，则小车会出现扫码绑定界面

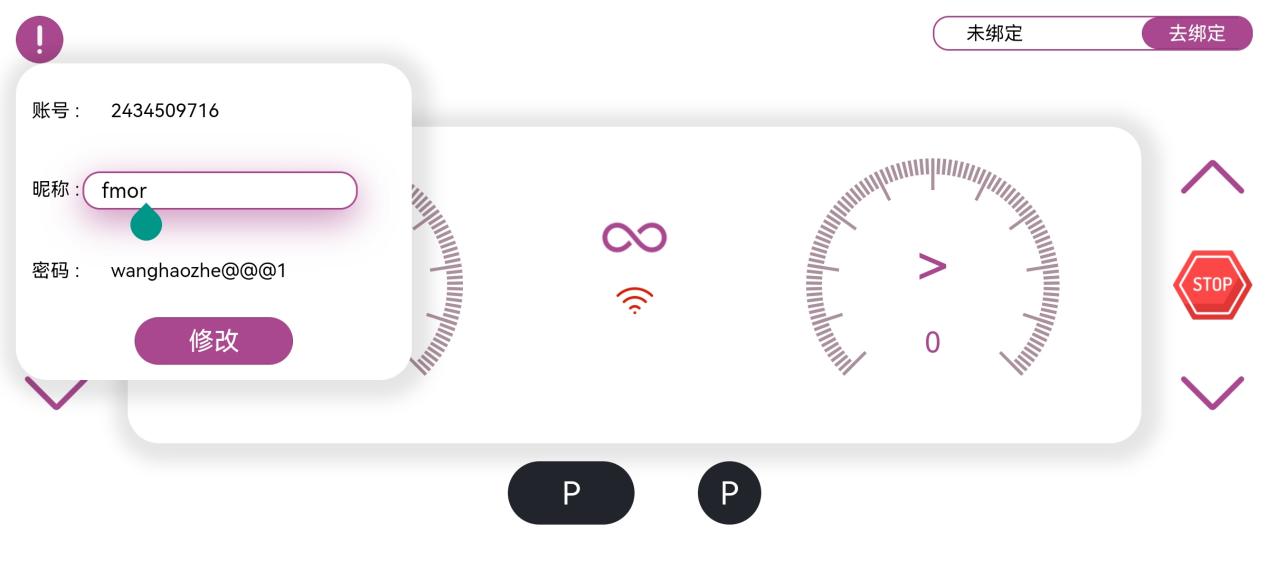


用户需要在手机APP上注册账号，并登录。用户可根据需要修改个人信息。用户ID是用户的唯一标识，不可重复注册。









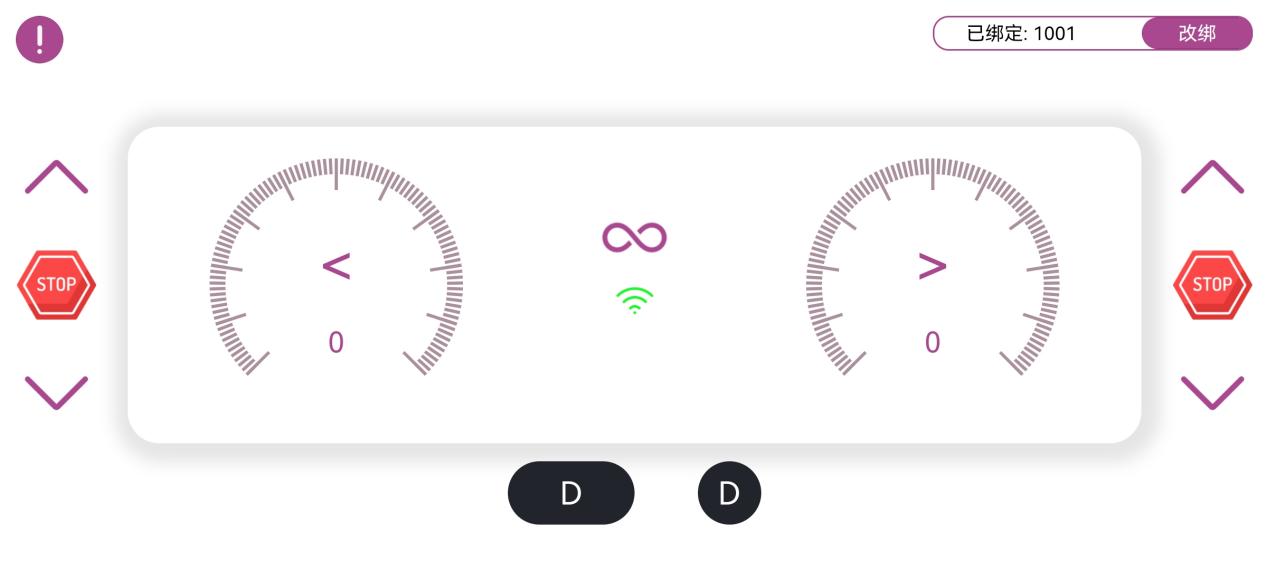
此时点击去绑定，即可扫描小车的二维码，进行小车绑定



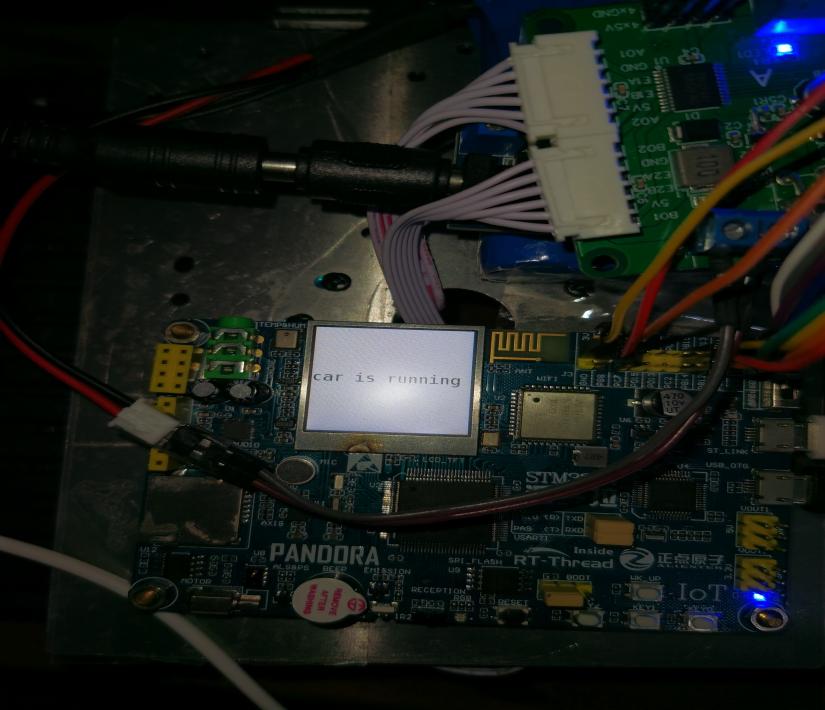
绑定成功



小车若在线，则信号图标变为绿色，否则为红色



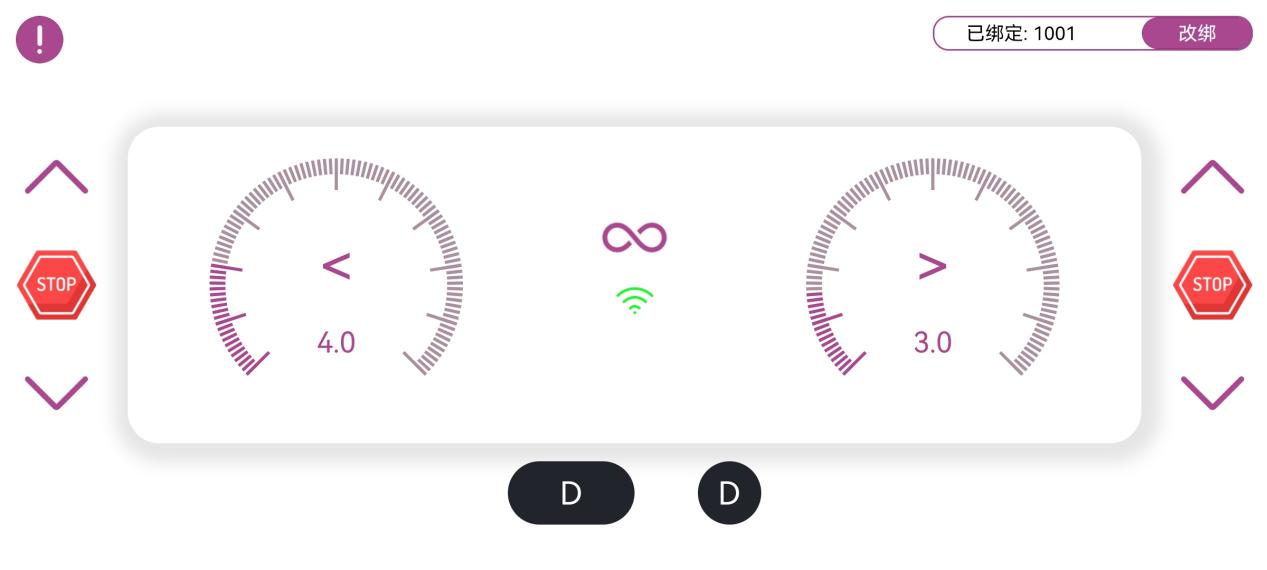
绑定成功后小车进入运行状态

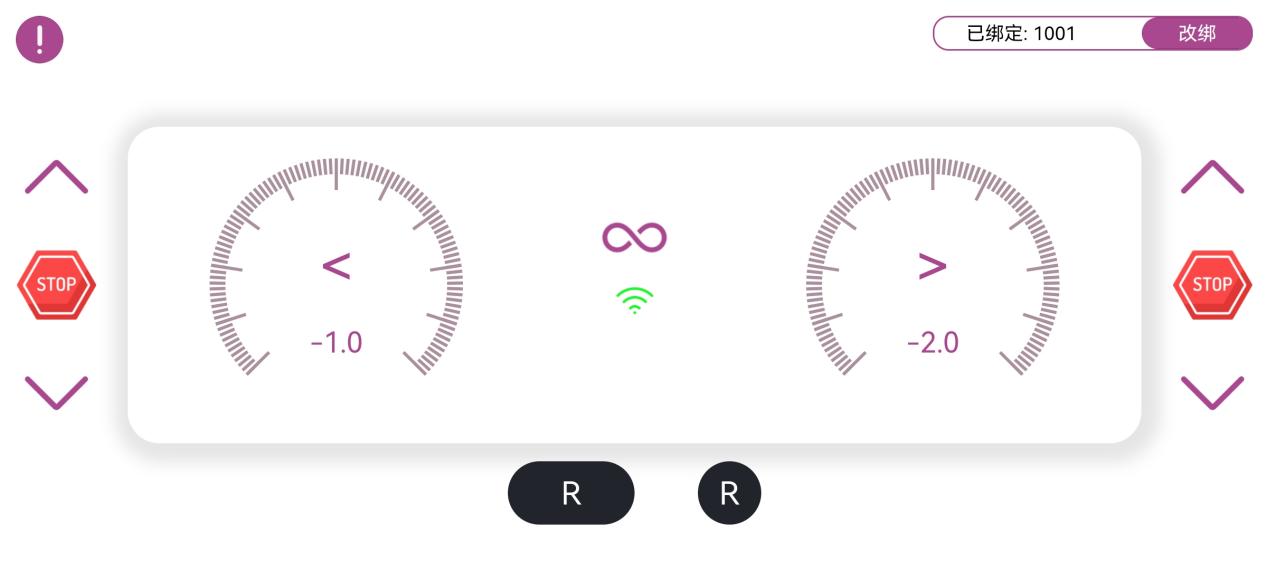


用户便可通过手机APP，控制小车进行运动，满足各种场景的需求。同时小车的左右履带运行速度信息等会实时在APP上显示，前进速度为正，后退速度为负。

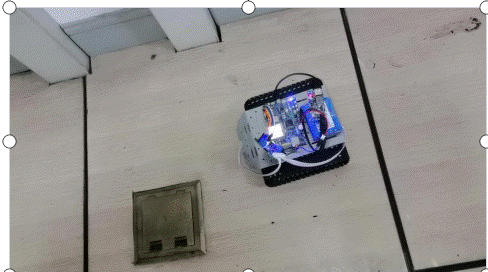
提供四种挡位，即P（泊车）、R（后退）、N（空挡）、D（前进）

其中P档时，电机处于制动状态，N档时，电机处于放松状态

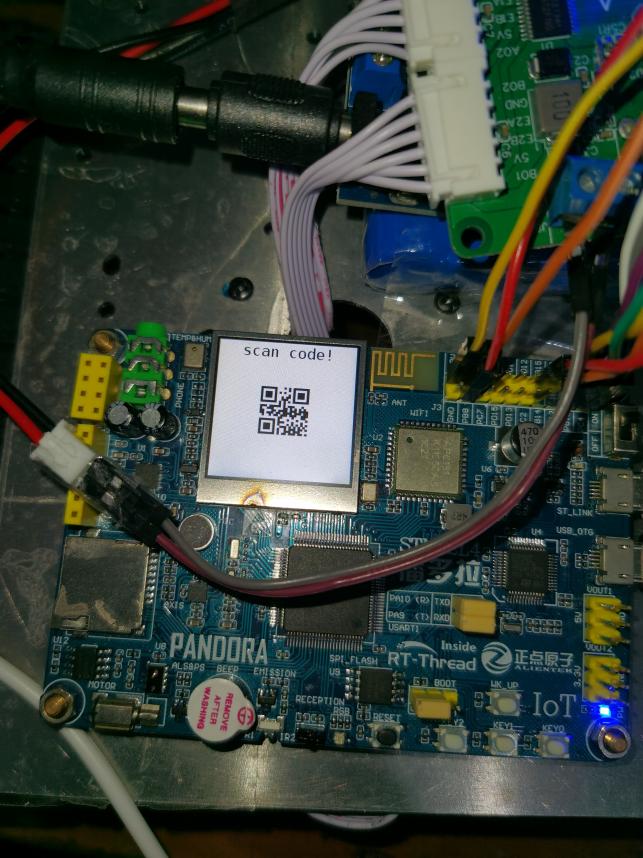




小车运行



改变绑定小车的功能（单一用户可切换控制对应小车，而每一辆小车对应的控制用户是唯一的），按下主控开发板的KEY0键进入更改绑定界面，KEY1键退出更改绑定界面。



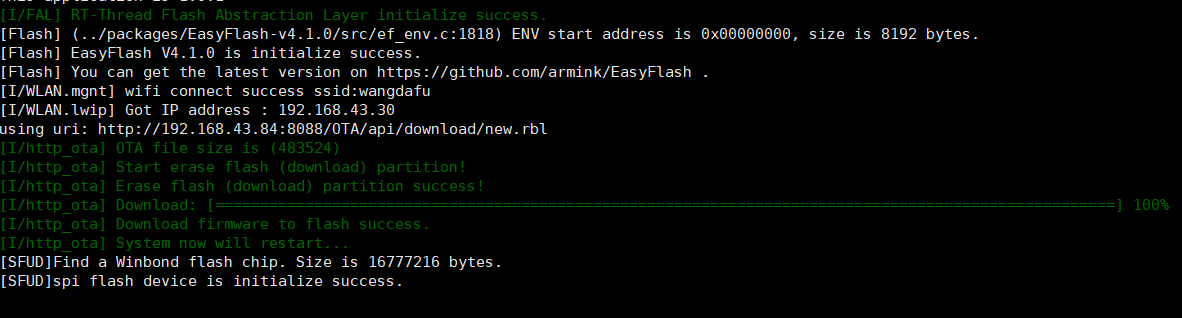
OTA空中升级功能，按下KEY1键，会检查服务器上是否有新版本存在，若已经是最新版本，则提示版本已经最新

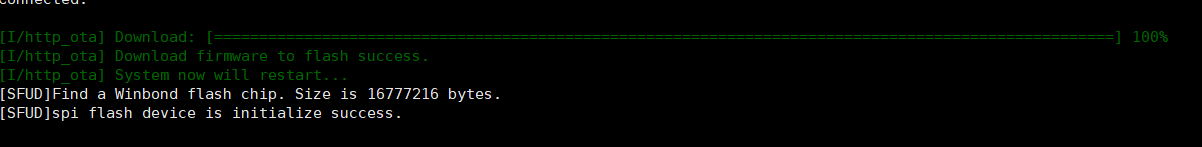


若有更新的固件版本，则进行下载，并更新。更新过程会显示OTA进行中。

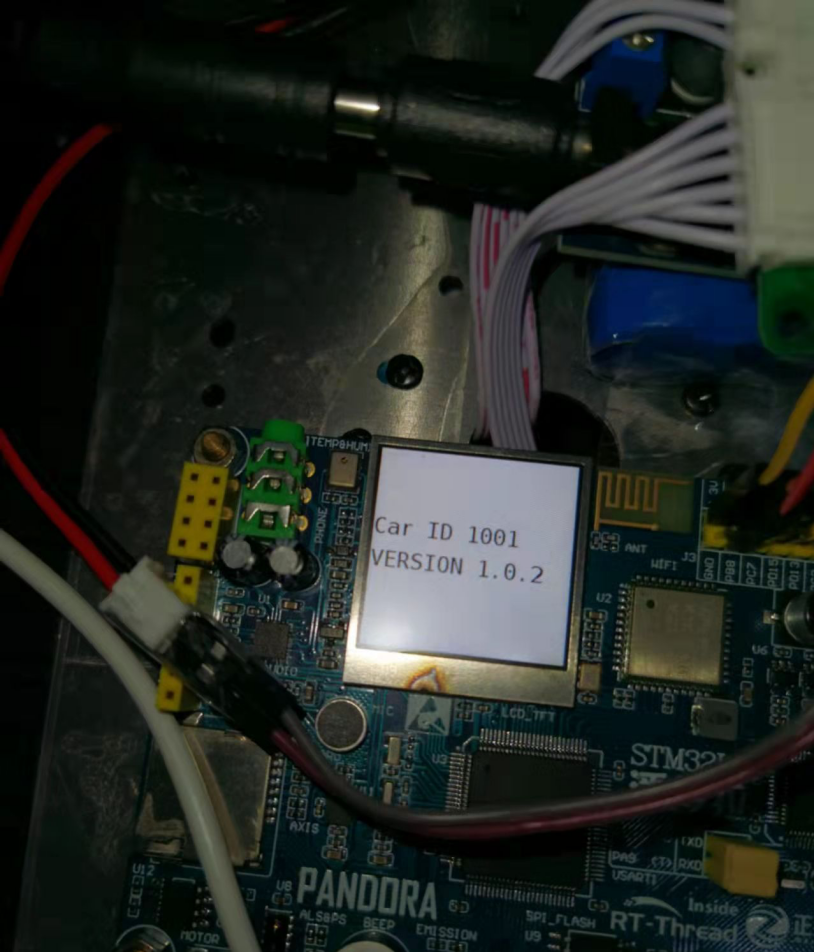


同时串口命令行输出更新的信息





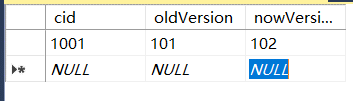
更新完成后，小车会重启。本次更新是从1.0.1版本更新到1.0.2版本，则开机载入界面显示的版本信息变为1.0.2



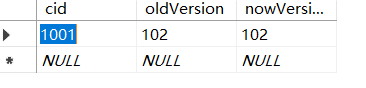
按key1检查更新显示版本已经最新



版本管理在开发阶段直接更改数据库中的版本信息字段，项目成熟后会使用后台管理页面进行版本管理。当然，日后的小车系统版本管理及其他后台管理与用户无关，是应由运维人员进行的



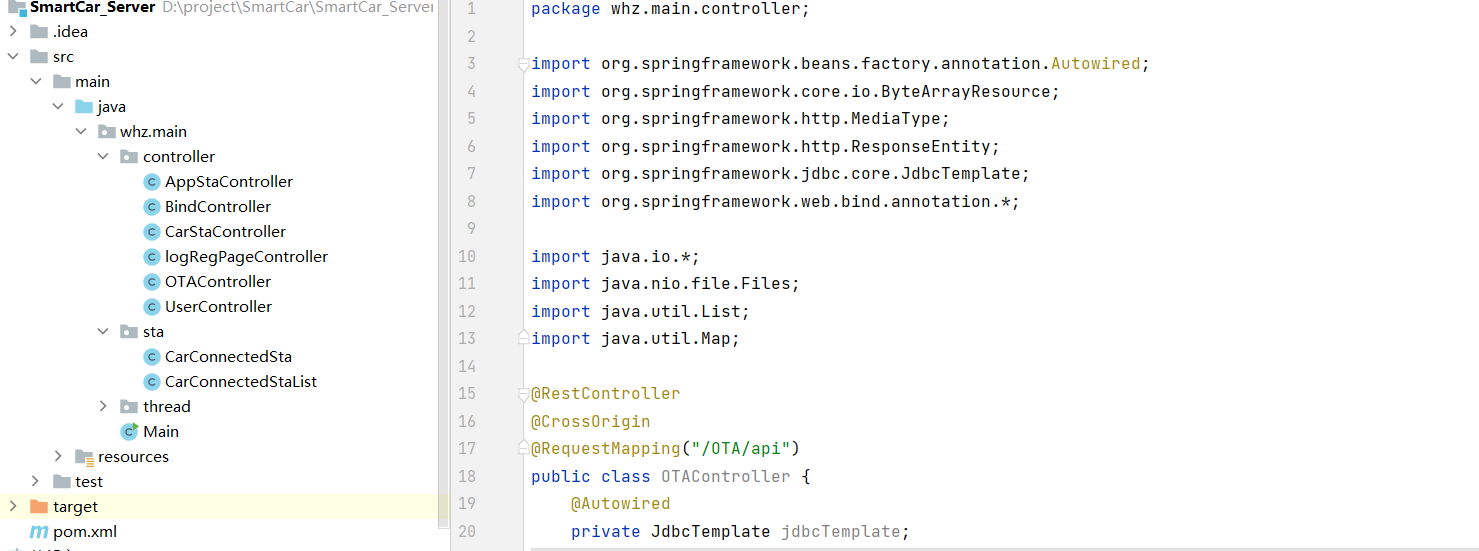
更新完成后版本号自动同步至最新

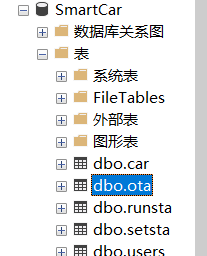


**3.私有云平台、手机APP简介**

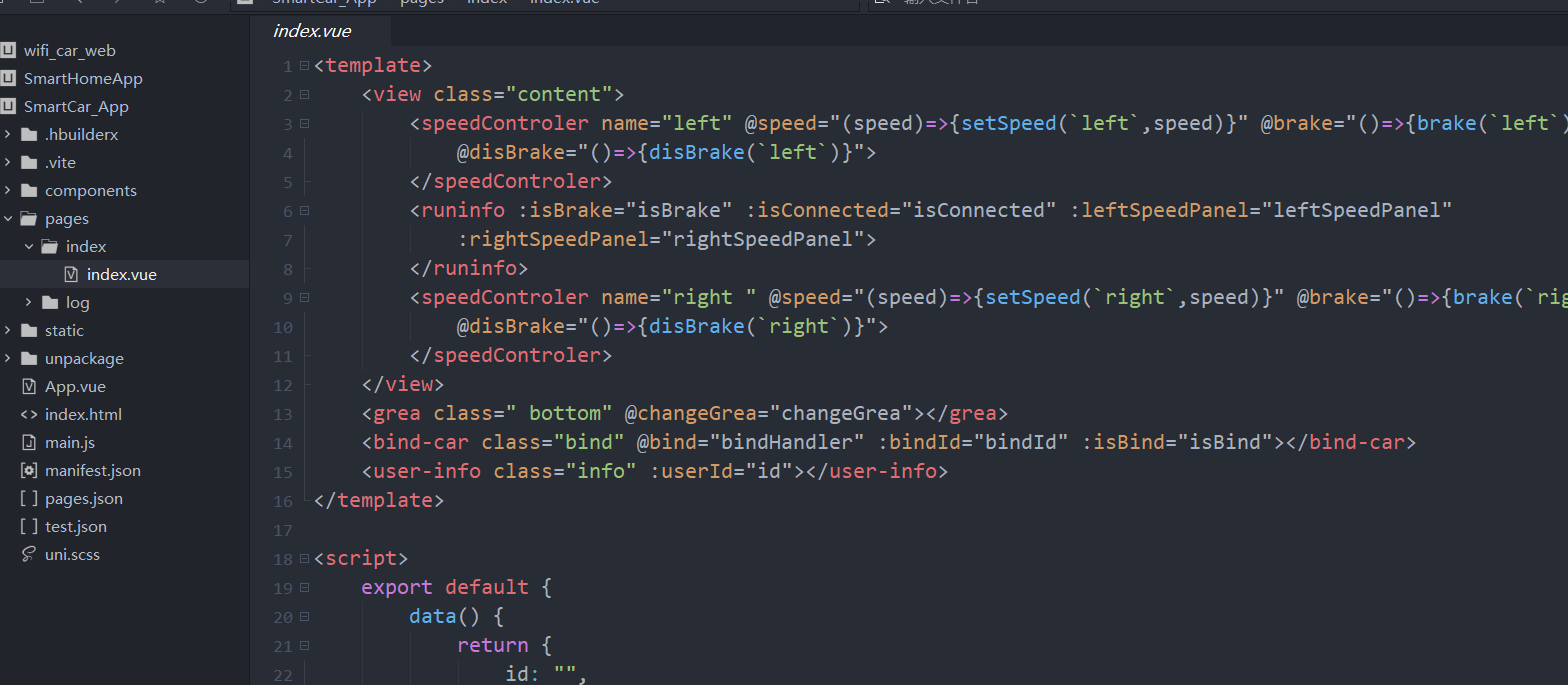
由于是嵌入式系统应用的设计报告，这两部分便不进行详细说明，只进行简述。

本系统使用的云平台为自行构建的基于http协议的，使用springboot构建服务端，mssql构建数据库的私有云平台。支持OTA、在线状态判断、数据通信等一般物联网云平台所具有的功能，但是相较公有云的通用性，本系统的私有云平台更加小巧、定制化。





本系统的手机APP使用了混合开发技术，使用国产uni-app框架进行手机app开发，此框架基于vue，只需要一套代码，便可编译成为小程序、网页、IOS APP以及Android APP。本系统开发测试阶段只使用其开发完成安卓APP，项目成熟之后只需进行编译工作便可进行多端的推广

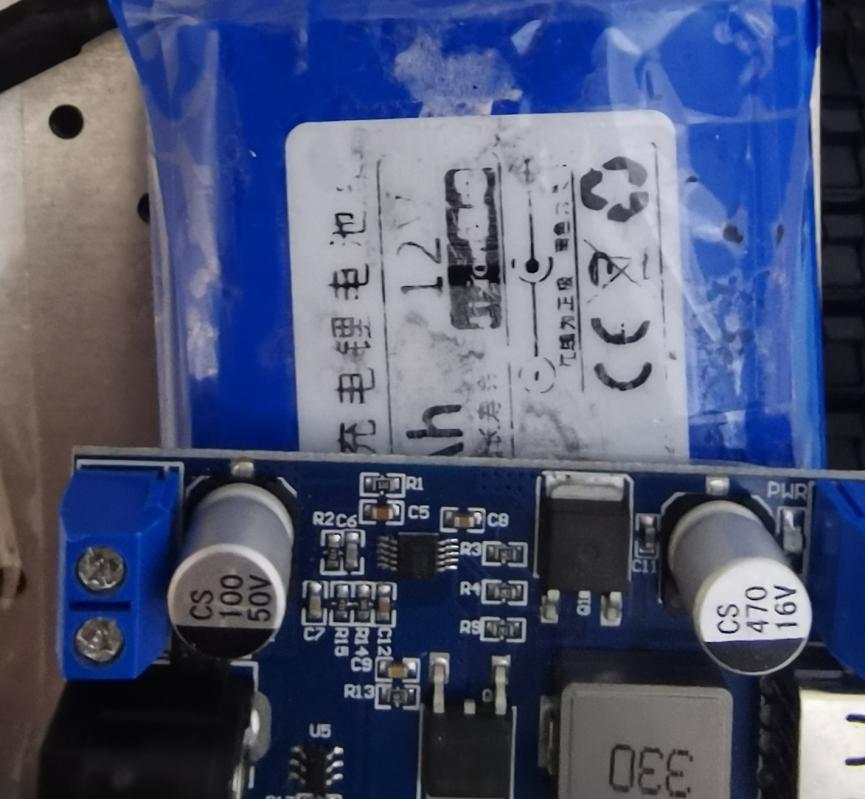


**4、嵌入式小车系统详情**

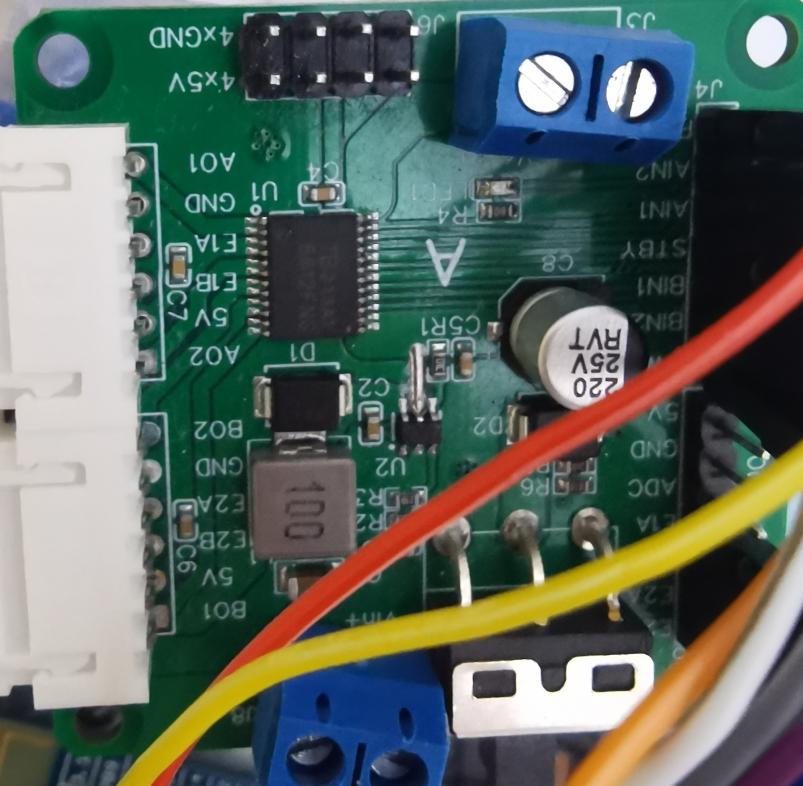
嵌入式小车系统为本系统的精华部分，下面进行详解

4.1硬件组成

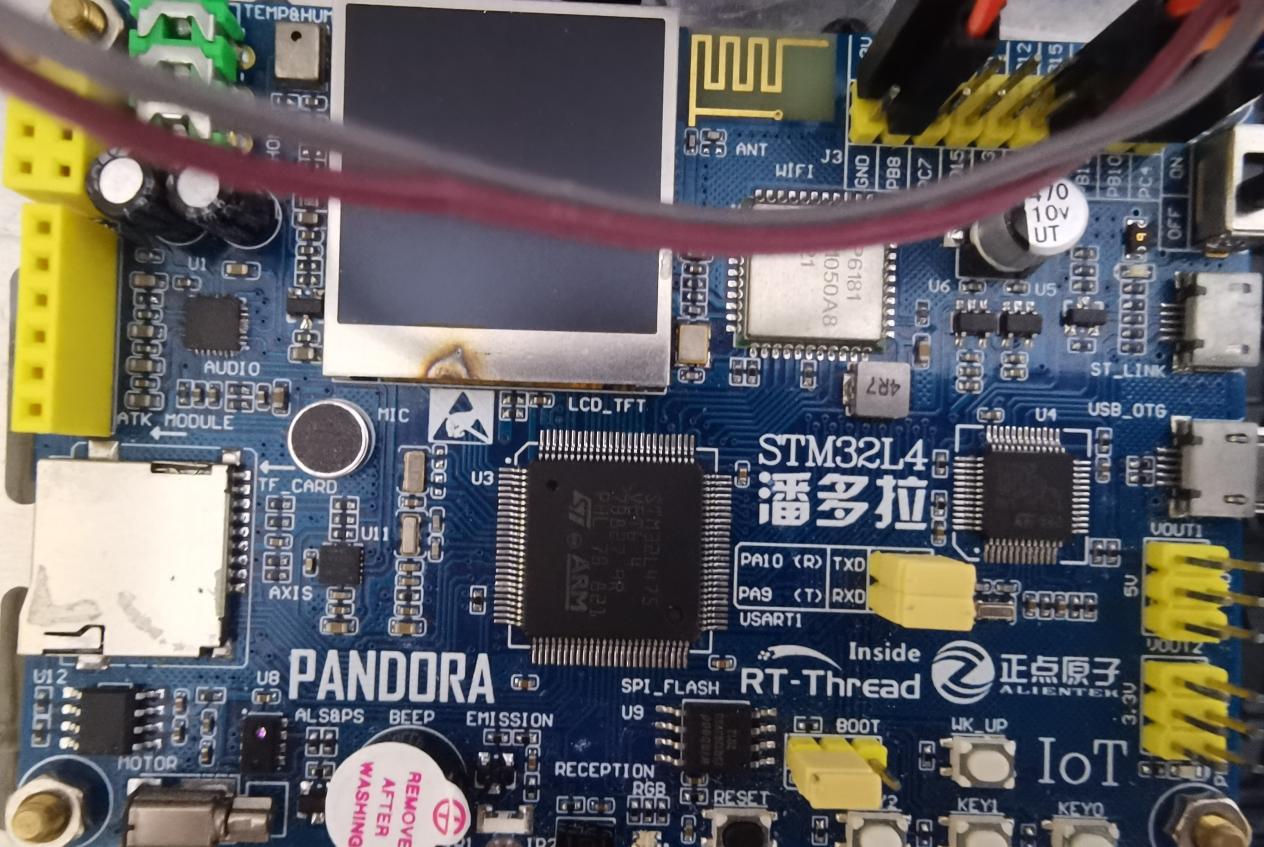
电源部分由降压模块和12v可充电电池组成，负责给mcu开发板、电机供电



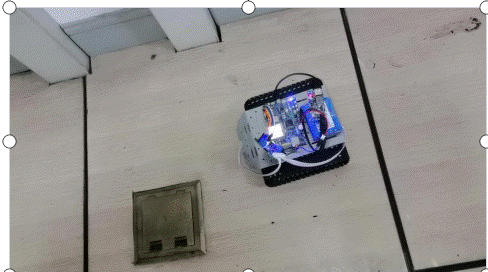
TB6612电机驱动芯片，负责MG513电机的驱动



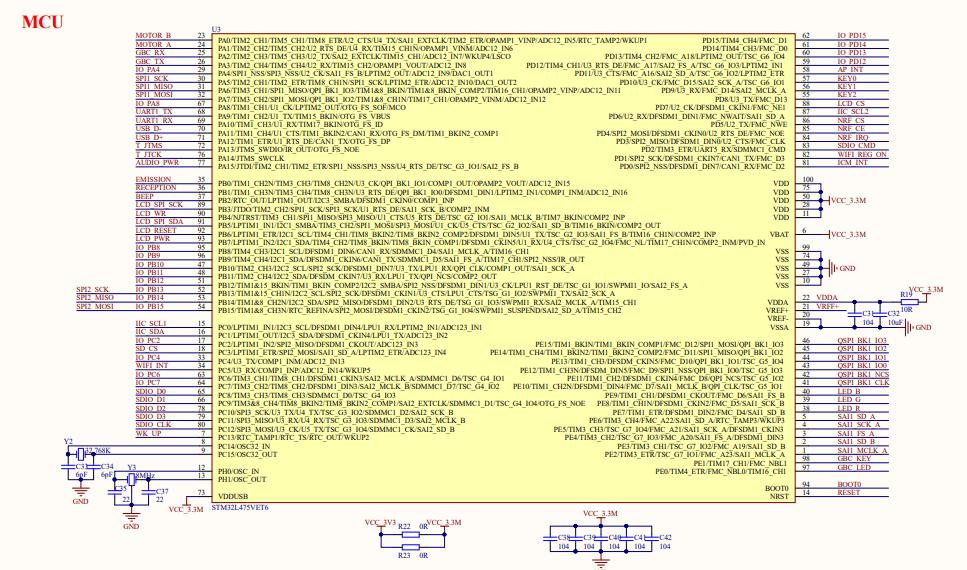
mcu主控开发板，负责小车相关的控制及网络功能

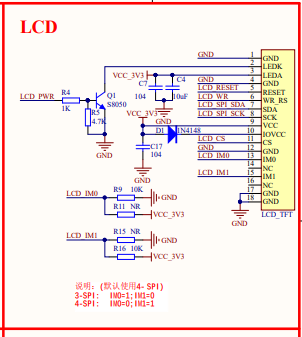


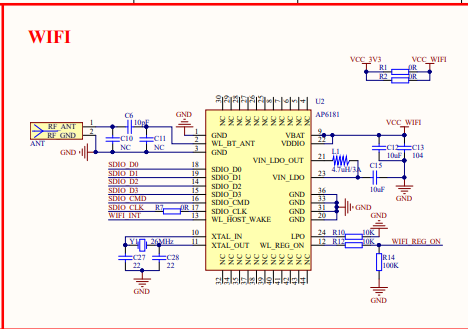
成品小车

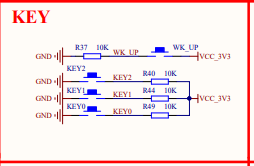


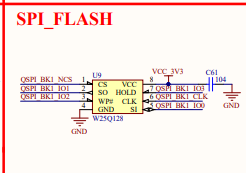
**4.2硬件原理图**

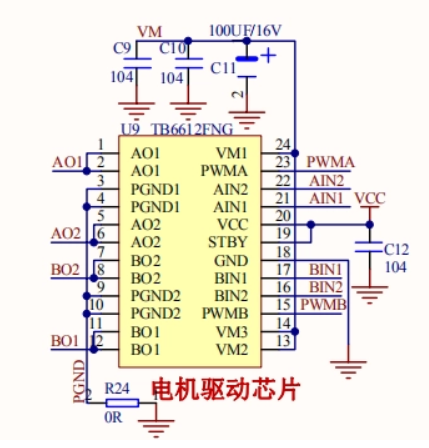








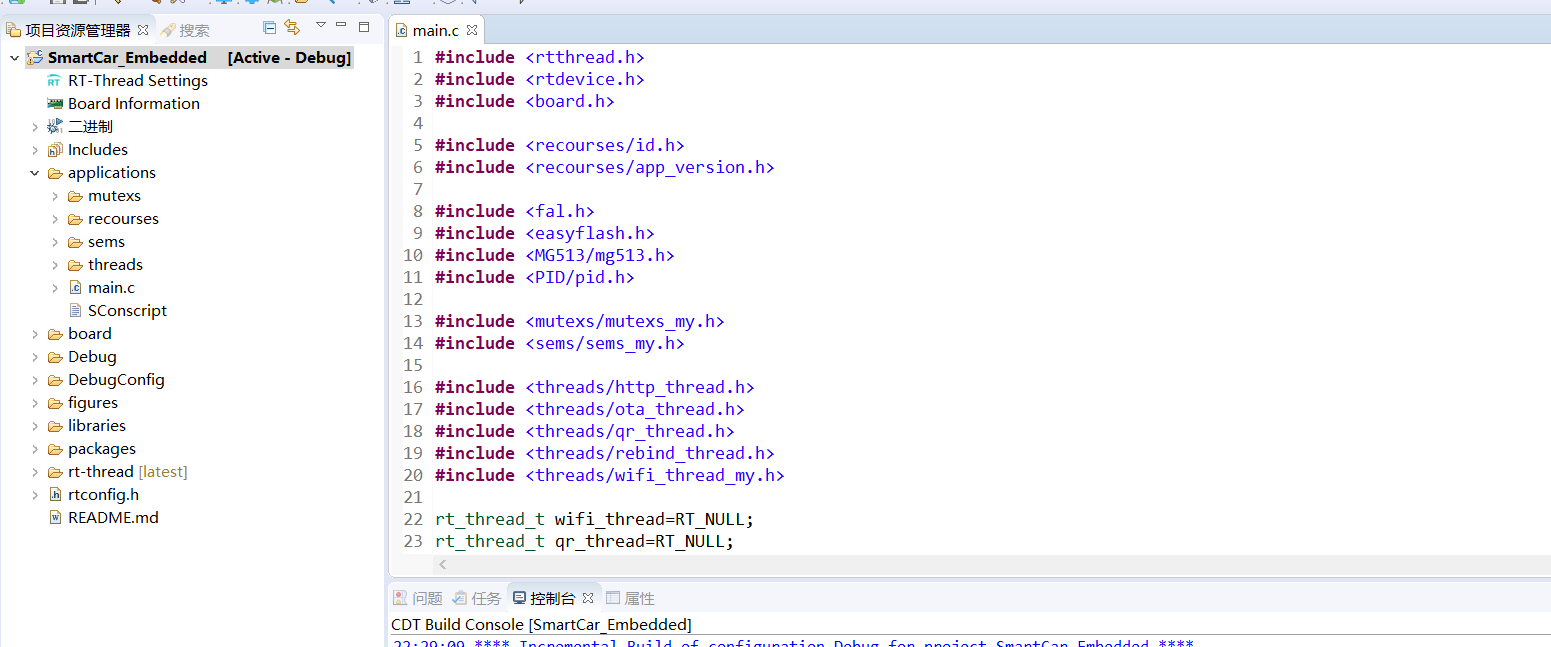




**4.3软件开发环境介绍**

嵌入式小车使用了RT-Thread进行软件开发，使用了RT-Thread提供的BSP进行开发，其提供的BSP支持了基本上全部的底层硬件驱动（如SPI、IIC、PWM等），于是其就很像本学期学习的CMSIS Driver。同时也提供了许多用于网络操作的软件包和用于LCD屏幕、SPI　FLASH等的驱动，也提供了移植好的文件分区以及虚拟文件系统

嵌入式小车的IDE选择了RT-Thread提供的官方构建工具RT-Thread Studio，如下图



**4.4嵌入式小车系统框图**

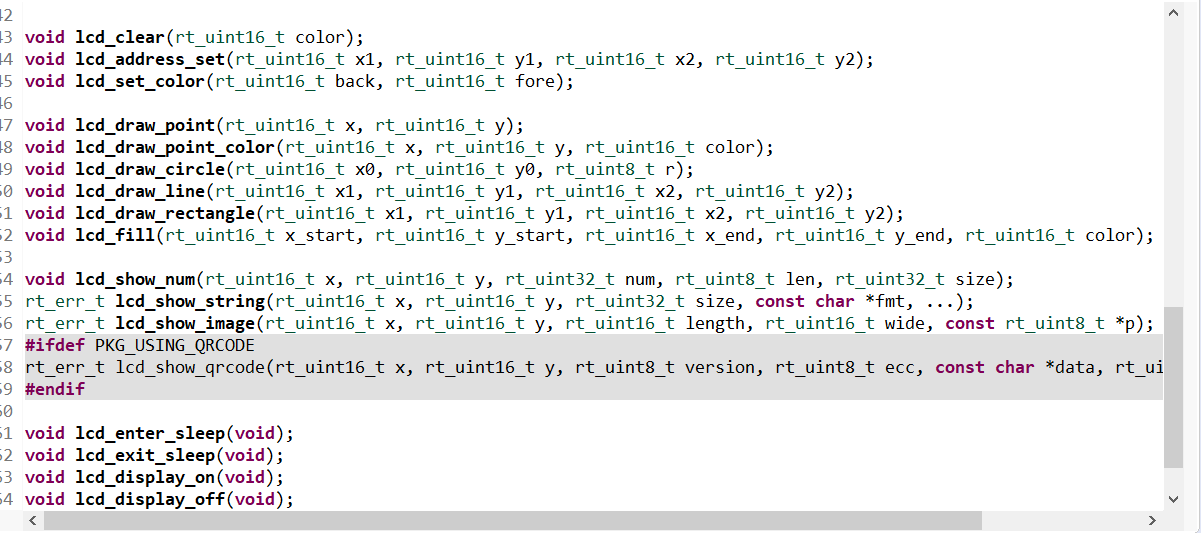


**4.5驱动层**

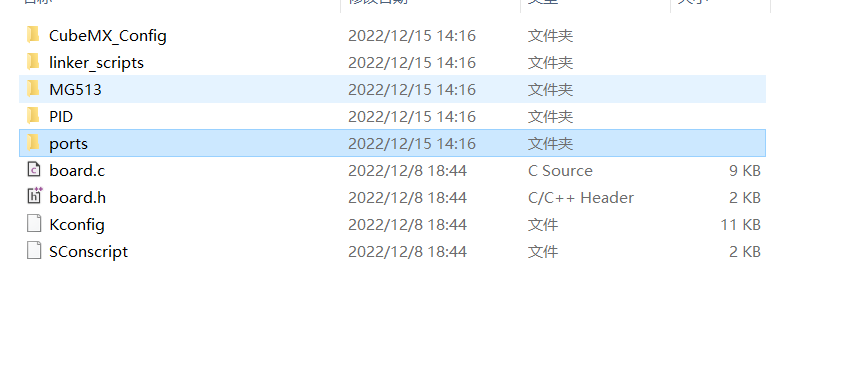
lcd驱动使用了BSP移植提供的，基于RT-Thread SPI设备驱动的LCD驱动，使用了SPI总线，在board/ports文件夹中

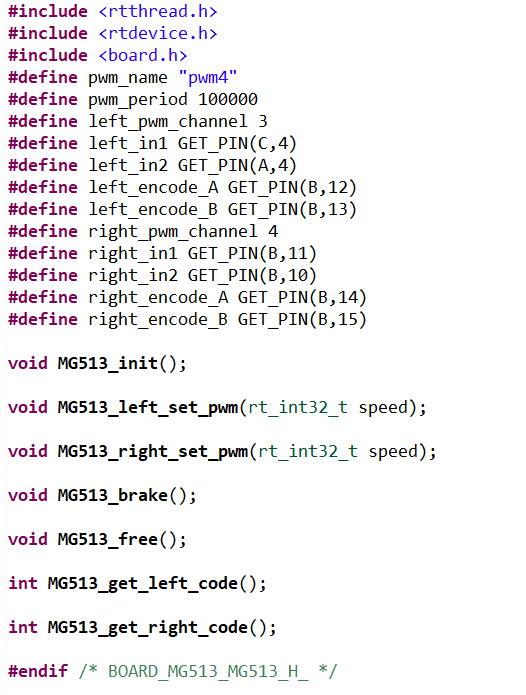




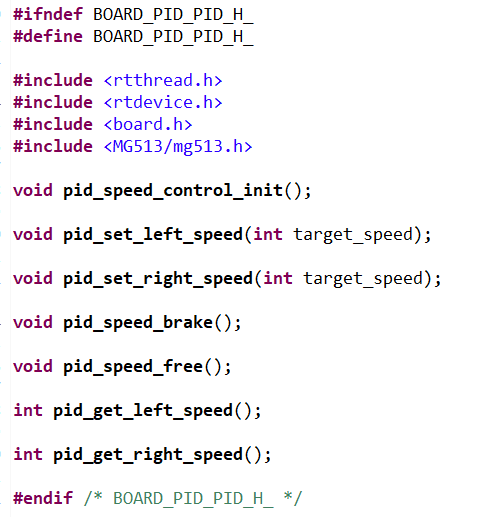
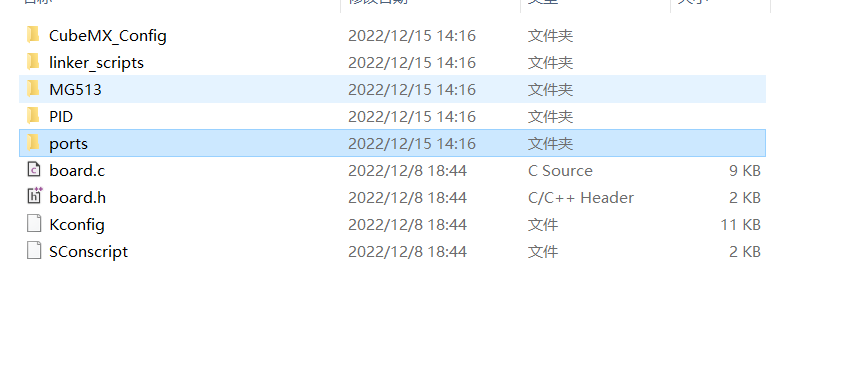


小车的MG513电机是一款带编码器的电机，通过TB6612芯片驱动，源码在board/MG513文件夹。由于开发板外接引脚资源限制，编码器的捕获无法通过编码器硬件设备进行，于是不得不使用GPIO模拟的方式捕获编码器





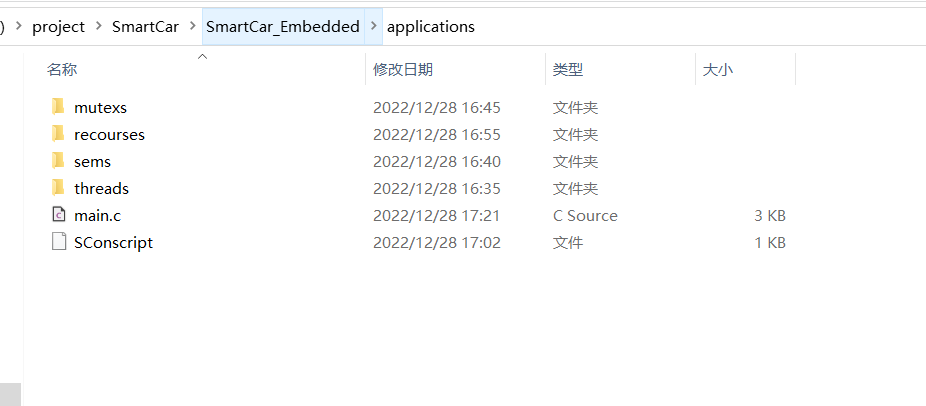
PID电机控制驱动，源码在board/PID中



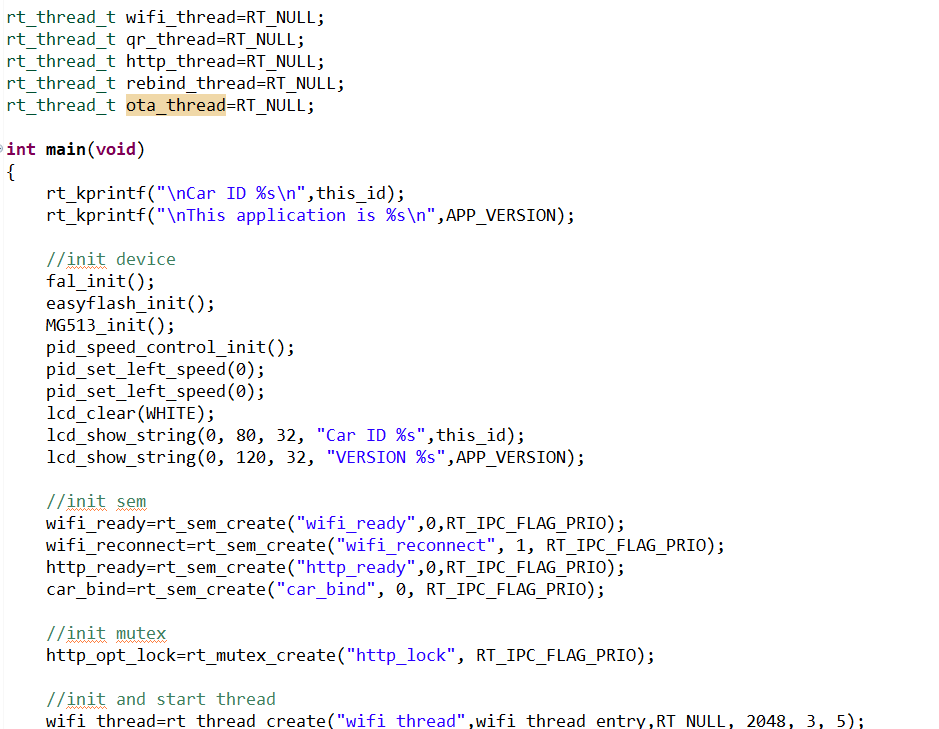
网络的操作全部由RT-Thread提供的wlan框架以及网络组件完成。WIFI芯片是通过sdio总线与mcu通讯的，然后RT-Thread已经做好了lwip堆栈的移植，我们只需启用便可实现轻松的嵌入式设备上网。而后RT-Thread又提供HTTP通讯的软件包，于是网络相关操作便十分简单。

**4.6应用层**

源码在applications中

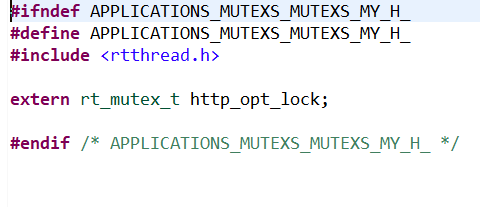


main 线程，完成了设备、信号量、互斥量初始化，线程初始化和启动

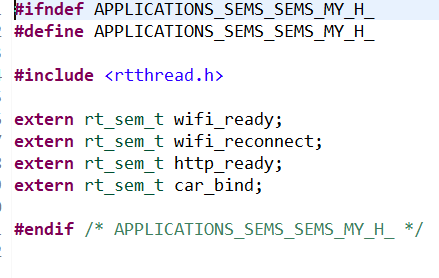




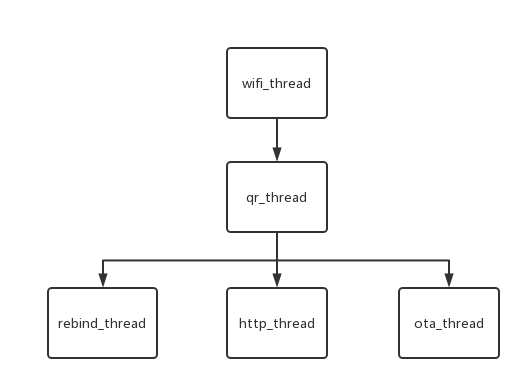
mutex文件夹中存放互斥量，由于每次http请求是非原子性的，若遇到http请求缓慢等情况，下一个http请求又以及开始，就会会出现一些连接性问题，导致请求失败。于是http\_opt\_lock就保证每一次http请求都要完成。



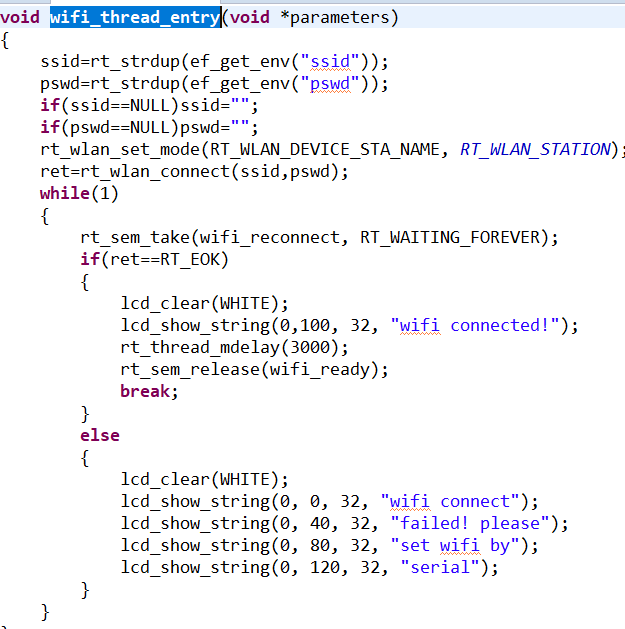
sems文件夹存放信号量

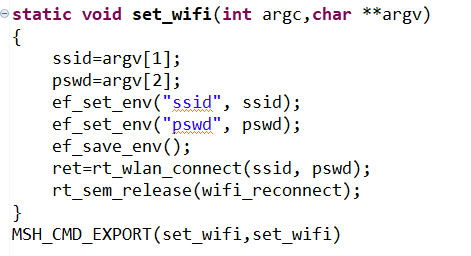


通过信号量控制，线程的执行流程为



wifi\_thread通过easyflash获取储存的ssid pswd并进行连接，若连接失败则提示进行配网，使用配网函数set\_wifi进行配网。连接成功释放wifi\_ready信号量。RT-Thread会在串口启动系统命令行，可导出函数到系统命令行成为系统命令，于是配网函数就是成为了系统命令。配网函数会释放wifi\_reconnect信号量，以进行重复的连接判断。



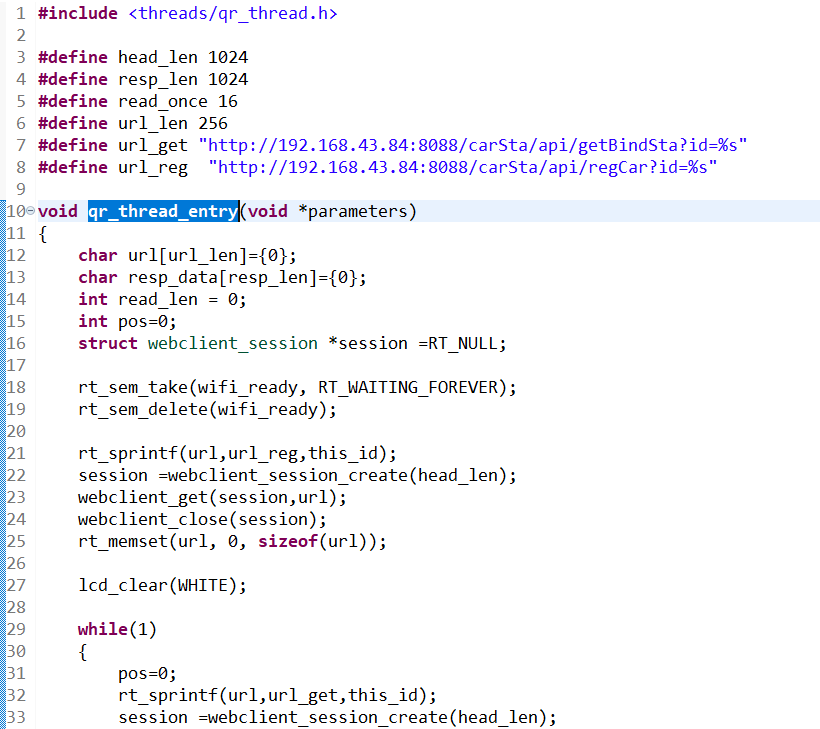


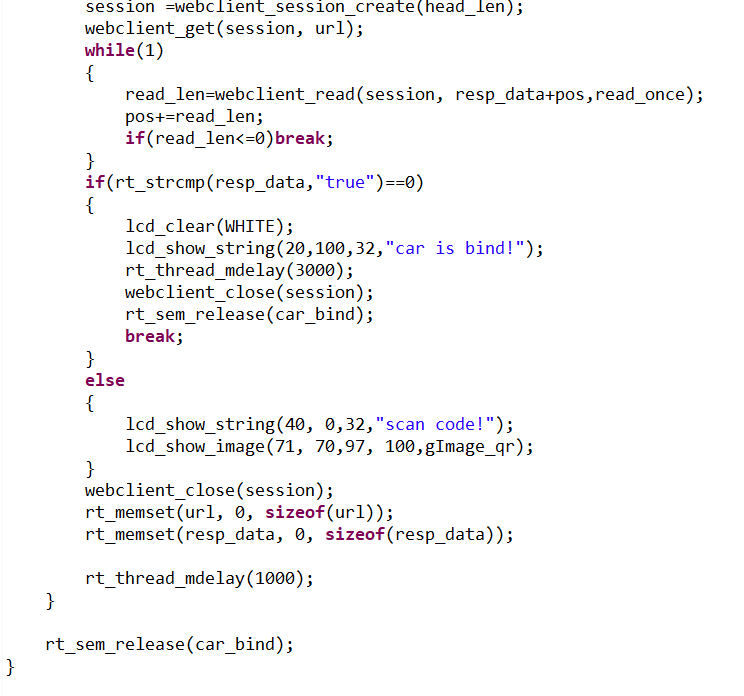
qr\_thread接收wifi\_ready信号量，并通过

<http://192.168.43.84:8088/carSta/api/regCar?id=%s> API 注册小车到云平台

通过

http://192.168.43.84:8088/carSta/api/getBindSta?id=%s API获取小车是否被用户绑定，已经绑定则释放car\_bind信号量，若没有被绑定，则显示绑定页面，循环获取绑定状态，直到用户绑定后释放car\_bind信号量





http\_thread接收car\_bind信号量，释放两次http\_ready信号量

通过每30ms一次的

<http://192.168.43.84:8088/carSta/api/getSta?id=%s>

API请求，获取云平台的小车控制数据，通过sta\_parse进行解析和两个履带的速度控制

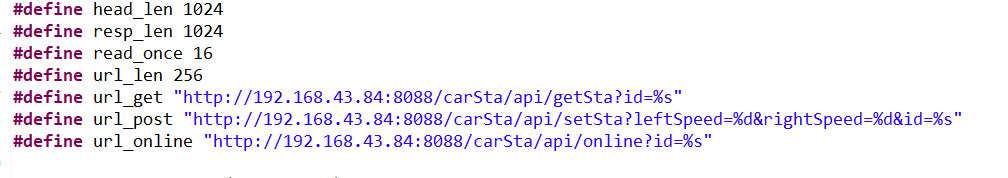
通过每30ms一次的

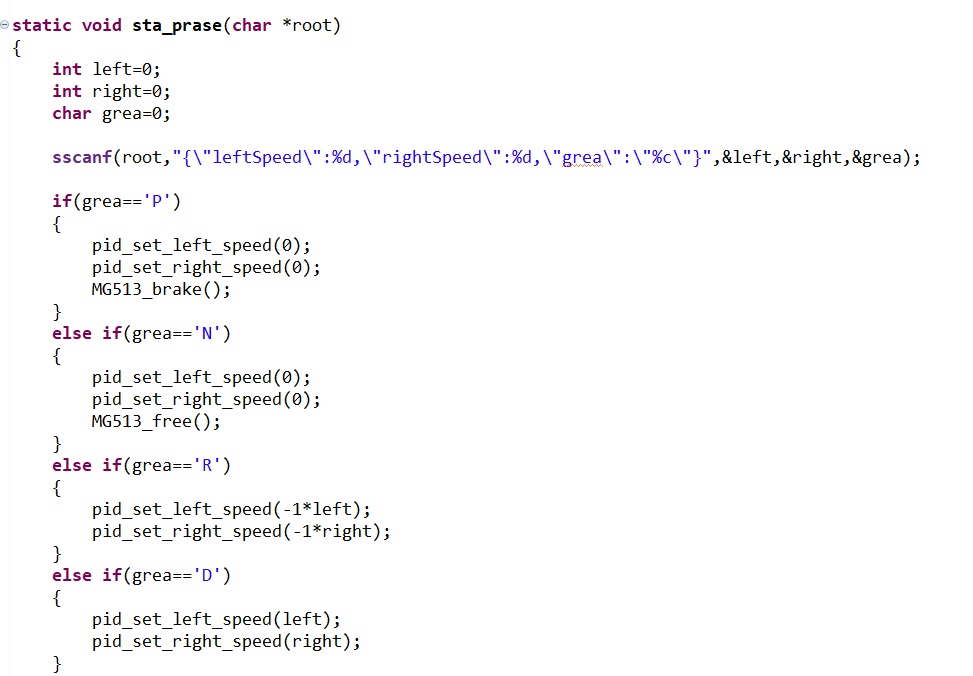
<http://192.168.43.84:8088/carSta/api/setSta?leftSpeed=%d&rightSpeed=%d&id=%s> API请求，向云平台发送小车的运行数据

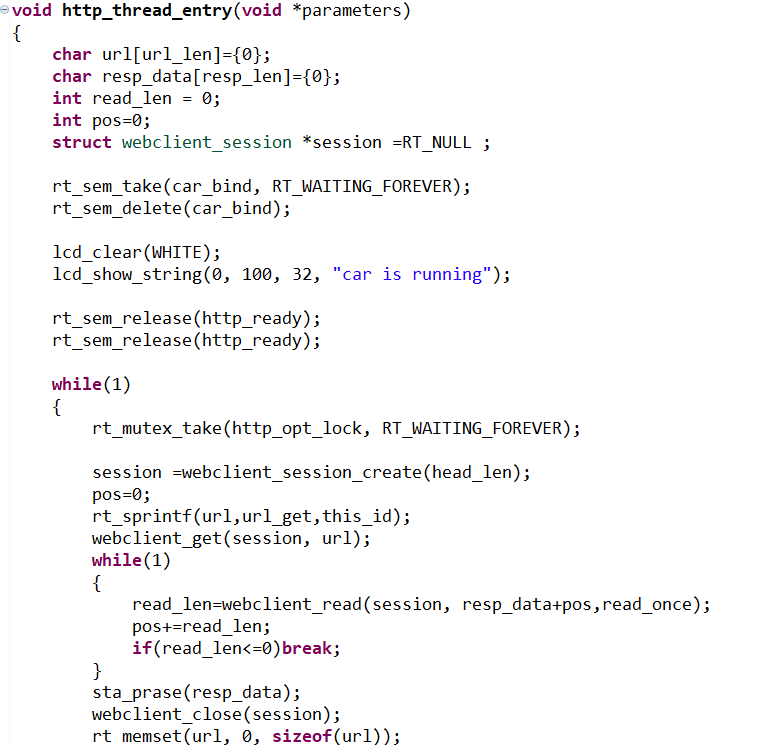
通过每30ms一次的

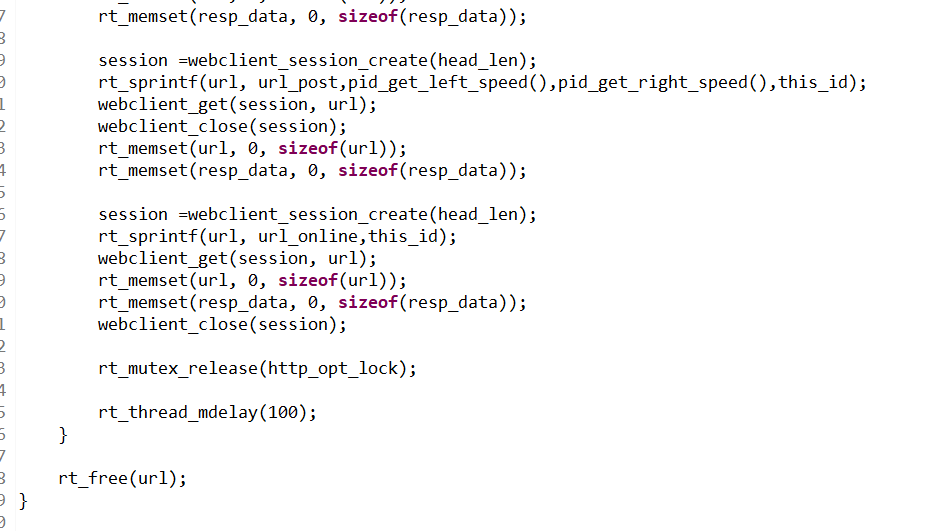
<http://192.168.43.84:8088/carSta/api/online?id=%s>

API请求，向云平台发送在线心跳包。若2s内没有发送该心跳包，则云平台判断离线

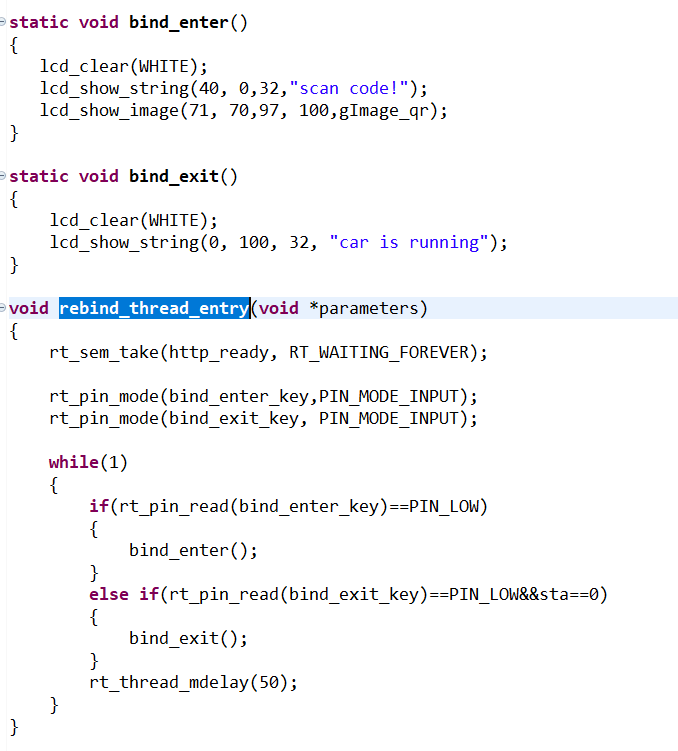








rebind\_thread接收http\_ready信号量，循环检测按键，按下key0进入改变绑定界面，在绑定界面中按下key2退出绑定界面。由于GPIO端口被编码器捕获的中断所冲突，于是不得已使用循环检测方式判断按键状态



ota\_thread接收http\_ready信号量，循环检测按键，key1按下则通过

<http://192.168.43.84:8088/OTA/api/checkUpdateByCid?cid=%s>

API判断云平台上是否有新版本的固件

若无，则显示当前固件版本，提示版本已经是最新

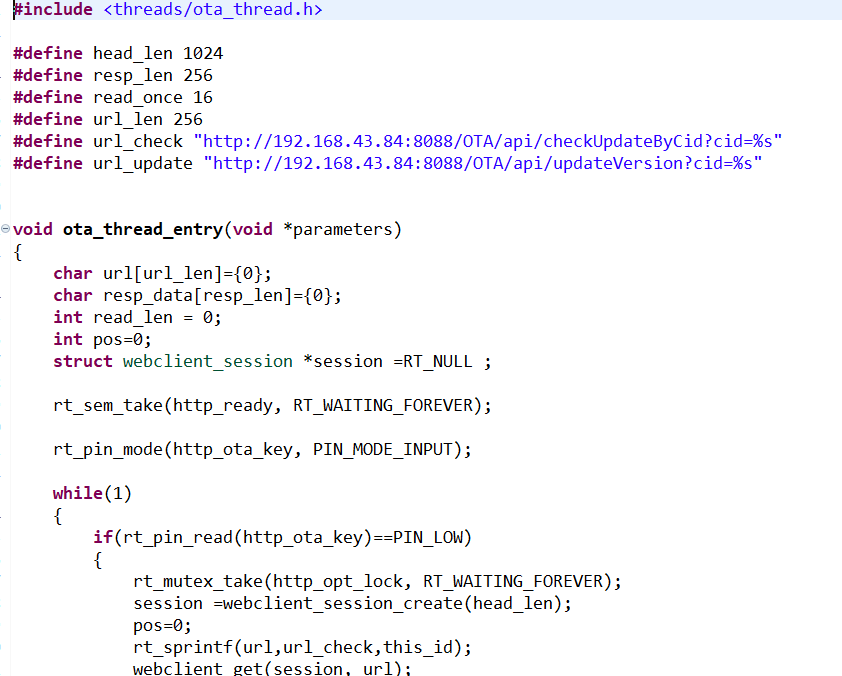
若有，则通过

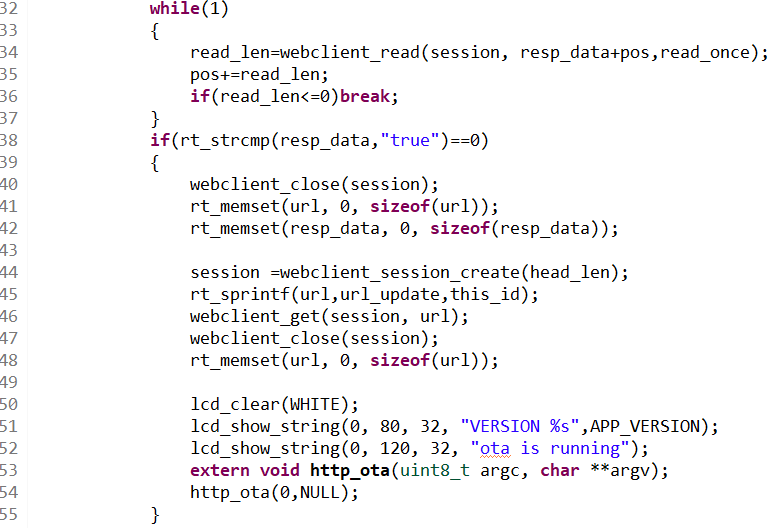
<http://192.168.43.84:8088/OTA/api/updateVersion?cid=%s>

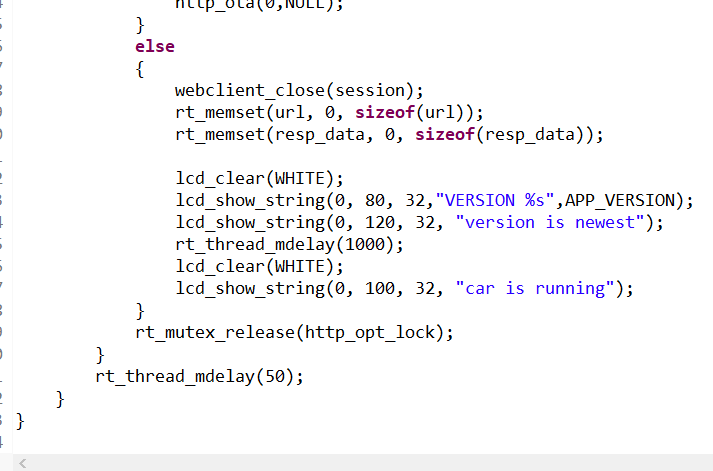
API自动更新小车的当前版本记录，调用RT-Thread提供的 http\_ota接口，从

[http://192.168.43.84:8088/OTA/api/download](http://192.168.43.84:8088/OTA/api/updateVersion?cid=%s)/new.rbl

下载新版固件到spi flash中的文件分区downlaod中，并重启，通过bootloader更新app分区的固件

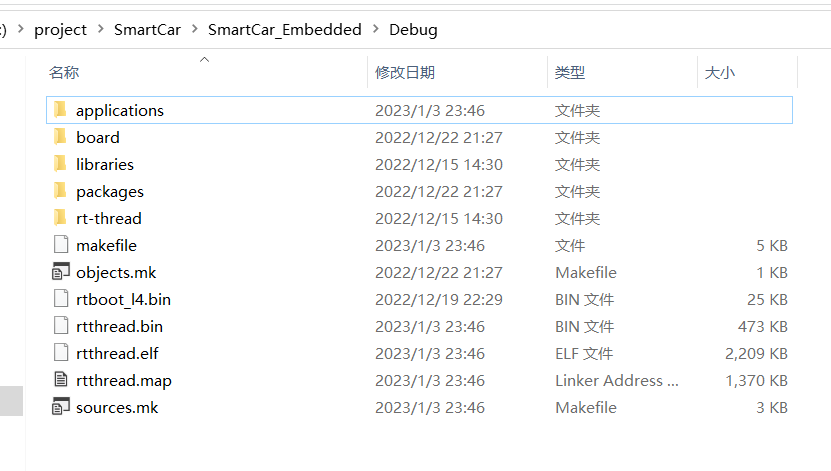






**4.7bootloader+OTA介绍**

本嵌入式小车系统使用的bootloader为RT-Thread官方提供的bootloader，可在RT-Thread官网获取，下图rtboot\_l4.bin即为bootloader

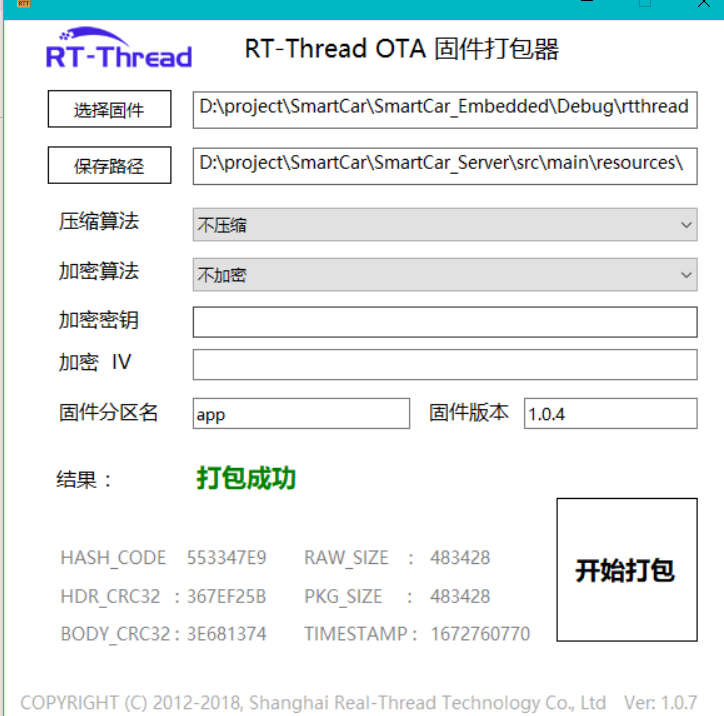


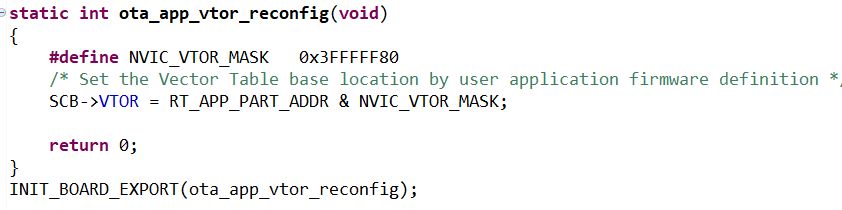
使用该bootloader时需要确保APP固件已经在片内Flash创建app分区，在片外SPI Flash创建download分区。OTA下载的固件都会存在download分区。本嵌入式小车系统的OTA下载方式为HTTP下载

bootloader会对比APP分区中的固件与download分区中的固件是否有差异，若有差异则进行搬运，将download分区中的固件搬运到APP分区并运行

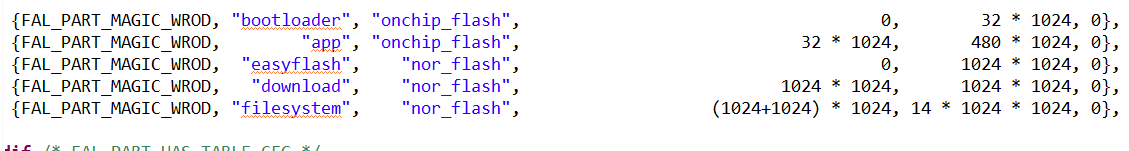
APP固件需要INIT\_BOARD\_EXPORT(ota\_app\_vtor\_reconfig)，即在系统板级初始化之前把中断向量表地址跳转到APP分区

APP固件完成后，需要通过工具完成打包编译生成的bin文件方可供OTA升级使用，需使用打包工具，并存放到指定路径：





本嵌入式小车系统片内flash中0KB—32KB空间为bootloader，32KB-480KB为APP。SPI FLASH中1024KB-2048KB空间为download分区。



**四、总结**

本系统较为复杂，初步估算嵌入式系统的代码工作量就有约3000行，再加上其他手机APP以及私有云平台的开发工作，整个系统的工作量比较巨大。所以由于时间关系，本系统仍然有许多不足的地方。问题即解决方法如下：

1.PID控制算法调好的参数依然不完美，有时导致控制过冲、速度不一致的问题。对于这个问题，在初步的研发工作完成后将进行进一步的PID调参工作，以让小车平稳运行。

2.使用网络的连接方式的问题，要想实现远距离的控制，使用WiFi是不行的，应该使用NBIOT等网络连接方式。对于这个问题，在初步的研发工作完成后，将使用NBIOT模块替代WIFI芯片，而RT-Thread也支持了NBIOT的软件包，可以直接在NBIOT模块上使用HTTP或者MQTT协议。

3.还缺少后台管理界面，进行用户、小车、小车固件版本等后台管理需要。对于这个问题，将在初步的研发工作完成后，开发一个后台管理界面，以进行用户、小车、小车固件版本等的后台管理。

4.小车控制器件尺寸不匹配小车，对于这个问题，将在初步的研发工作完成后，自行设计PCB，完成一个合理的，适合小车尺寸的主控板

总体来说，本物联小车系统还是完成的较为完善的，已经具备了初步应用能力。而本系统基本是从0开始的，我完成了几乎所有的开发工作，不只是嵌入式小车部分，还有私有云平台、手机APP的开发工作，这使我初步具备了全栈开发的能力，同时更使我对嵌入式系统有了更深的了解，也了解了关于嵌入式文件系统、嵌入式设备网络操作、嵌入式设备bootloader等高深的知识，同时也更深层次了解了诸如UART、SPI等通讯总线，也了解了编码器的使用，同时更加深层次的了解了RT-Thread这一物联网操作系统，了解其在这个物联时代，相对于传统嵌入式实时操作系统的优势所在。

总的来说，本次项目完成度好，成品功能丰富完善，我也获益颇多。