

# 创新实验课

## 课题中期报告

课题名称：基于树莓派的智能移动“充电桩”小车

课题名称（英文）：Intelligent mobile“charging station”car based On Raspberry Pi

课题组长：闫康 2022210566

课题组成员：苏珈磊 2022210560 孙常鑫 2022210581 薛皓林  
2022210571

报告时间： 2024 年 9 月 05 日

# 1. 当前选题与立项时的差异

我们的当前选题与我们之前在立项时候的选题是不太冲突差异不大的，先来重新叙述一下我们在立项时的选题，我们项目的名称是基于树莓派的移动“充电桩”智能小车，我们的项目描述或者说是系统具有的功能是：在室外时，小车通过 GPS 固定巡逻轨迹，用户通过小程序界面可以实时定位小车的位置，用户可以选择让小车立即原地停车或者是让小车追踪到用户所在位置在停车，待用户从小车悬挂栏取下充电宝后，用户确认后，小车拍照作为记录，并存储，拍照后返回充电桩原待位位置，等待人为补充或者归还充电宝，随后再次出发。之所以说差距不大是因为项目的主要内容都是一致的，唯一需要变动的是对于室内的部分我们可能不会再考虑，因为室外项目的调试与完成都将是很大的考验，未来我们的选题不会发生变化，我们将持续在这条道路上走下去。

## 2. 课题工作进展

### 2.1 课题方案

#### 2.1.1 硬件

##### 1、硬件架构

1. **主控制单元**:本项目利用树莓派 4B 作为主控制单元，作为核心处理单元，负责运行操作系统，处理传感器数据，执行控制算法。



图 1 利用树莓派 4B 作为主控制单元和 TF 卡存储

2. **数据存储架构**:利用树莓派 4B 内置的 32GTF/SD 卡作为存储单元，存储烧录的系统，代码，软件，程序和系统相关设置以及图像等。

3. **结构件**:选用了稳固的车体框架以及车载固件悬挂系统，以用来保证小车在各种地形上的，稳定性和耐用性，安装轮子时充分考虑防滑设计，同时选用了合适的驱动轴，与轮子相搭配以用来保证与地面良好的附着力和通过性。本车采用了金属齿轮，铜制马达齿轮，钢化大齿轮，坚固耐磨，持

久耐用，金属中狗骨，同时采用碳纤底盘，耐磨耐撞。

4. 整个硬件架构旨在确保智能小车在执行移动“充电服务”的同时，能够稳定和安全地运行，并提供可靠的用户交互体验。通过精心设计的硬件架构，项目旨在实现高效的能量管理、精确的导航和灵活的用户操作。

5. 油压减震系统，小车配备可调式油压减震器最大程度地减缓小车在行进过程当中抖动。利用液体（通常是油）来吸收和阻尼震动的装置。它通常由一个充满油液的密闭缸体、一个或多个活塞、以及控制油流的阀门组成。配备油压减震器还有很多作用，油压减震系统能够吸收由于不平坦路面引起的震动和冲击，减少对小车结构和车载电子设备的损害。通过有效控制小车的弹跳和震动，油压减震系统可以提高小车在各种路面条件下的行驶稳定性。对于类似本项目携带敏感设备或进行精确操作的小车，油压减震系统可以提供更平滑的行驶体验。在恶劣的地形条件下，油压减震系统可以减少机械磨损，提高小车的耐用性和使用寿命。



图 2 小车油压减震器实物图

6. 基于 Donkeycar 的硬件框架，基于一种名称为 Donkeycar 的智能小车，下面的硬件介绍也包含 Donkeycar 小车集成的硬件，或自身携带的硬件。

## 2、传感器

1. 集成 MPU6050 加速度和陀螺仪传感器，用于实时检测小车的运动状态和方向。来弥补 GPS 传感器精度不足所造成的误差，进而从硬件方面来弥补“路径跟随算法”的精度。我们项目中的 MPU6050 也被称之为“惯导模块”，MPU6050 能够提供小车在三维空间中的姿态信息，包括俯仰角(Pitch)、横滚角(Roll)和偏航角(Yaw)，这对于保持小车行驶方向和稳定性至关重要，尤其是在小车直线行驶的保持方面，通过内置的 3 轴加速度计和 3 轴陀螺仪，MPU6050 可以实时监测小车的运动状态，包括线性加速度和角速度，帮助小车实现精确的运动控制，同时可以辅助 GPS 进行定位，在 GPS 信号弱或丢失的环境中，如室内或城市峡谷，MPU6050 可以提供短期的运动数据，辅助 GPS 进行位置推算，确保小车导航的连续性，并且将自身测量到的数据与 GPS 数据融合，

通过传感器融合算法（如卡尔曼滤波）提高定位的精度和可靠性。在某些应用场景中，MPU6050 可以作为主要的导航传感器，使小车能够在没有外部导航信号的情况下进行自主导航。相比于其他高精度的导航设备，MPU6050 具有成本效益，使得智能小车项目在预算有限的情况下也能实现高级的导航功能。MPU6050 的加入提高了系统的鲁棒性，即使在复杂多变的环境中也能保持小车的正常运行。

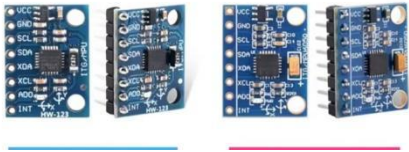


图 3 本项目所采取的 MPU6050 硬件图

2. 搭载微雪树莓派 GPS 天线和高精度拓展板，来实时获取智能小车的实时 GPS 位点信息，进而为“路径跟随算法”和自动导航算法搜集数据，奠定基础。选取本拓展板的原因是专门为树莓派所设计，接口连接简单，直接对应插入就可以。支持多种定位系统，如 GPS、GLONASS 和北斗。它提供快速定位和高精度坐标数据，适用于需要精确位置信息的项目，如自动驾驶、地理测绘和户外导航。同时网络资源比较丰富，很多人都倾向于使用，进而为我们的 GPS 部分的开发能够减轻复杂度，推进项目进度。



图 4 本项目所采取的 GPS 拓展板和 GPS 天线硬件图

3. 准备搭载红外传感器和 5 路灰度传感器以及超声波传感器，进而用其实现避障和灰度寻迹等附加功能(待定)。

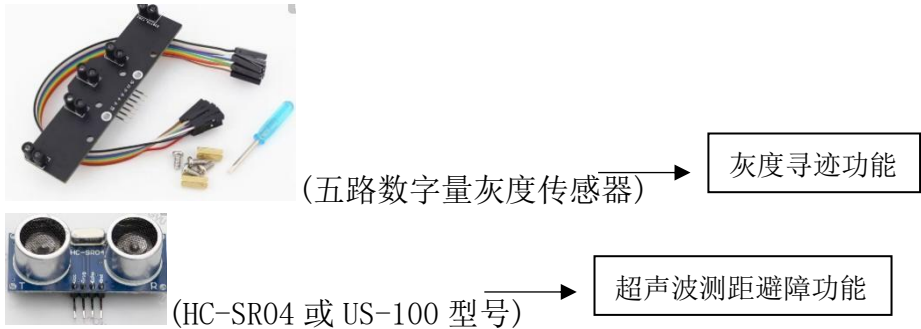


图 5 红外传感器和灰度寻迹硬件描述

### 3、外设

1. PCA 电机驱动芯片 9685(舵机驱动板)：PCA9685 是一款基于 IIC 总线通信的 12 位精度 16 通道 PWM 波输出的芯片，可用于控制舵机、led、电机等设备，I2c 通信，节省主机资源。在本项目中 PCA9685 驱动板用于驱动控制小车转向的舵机和电机电调。同时 PCA9685 内置时钟振荡器，无需外部时钟信号，方便使用。驱动板支持睡眠模式，降低功耗，适用于电池供电的应用，输出逻辑可以配置为开漏或者推挽输出，以用来适应不同的负载需求，同时具有过流保护和短路保护，提高系统的可靠性。而且驱动板大小较小非常适合我们的智能小车等类似的项目。我们利用 PCA9685 来实现舵机控制，实现小车的转向和复杂动作；同时驱动电机，控制小车速度。



图 6 本项目采用的 PCA9685 电机/舵机驱动板硬件图

2. 集成电子调速器(ESC)和接收机,利用这两个外设来实现 RC(遥控)的功能,ESC 是遥控小车中用于控制电机转速的设备。它接收来自遥控器的信号,并将其转换为电机的转速控制。ESC 通常与电池和电机直接相连,根据遥控器的指令调节电机的功率,从而控制小车的速度和方向。接收机是 RC 小车用来接收遥控器信号的设备。它将无线信号解码,并将指令发送给 ESC 和其他控制单元。接收机通常与遥控器配对,使用不同的频率或协议来确保信号的准确性和抗干扰能力。接收机的设计需要考虑信号的稳定性和可靠性,以保证小车在各种环境下都能准确接收指令。通过精确调节电机的转速,ESC 和接收机的组合使得小车可以进行精确的速度和方向控制,有利于本项目小车搜集路径的 GPS 位点信息。

3. 有刷电调(BESC)和碳刷电机,是一种用于控制有刷直流电机转速的电子设备,广泛应用于遥控模型车、无人机、机器人以及其他需要精确电机控制的领域。在智能小车项目中,有刷电调扮演着至关重要的角色。有刷电调通过改变施加在电机上的电压大小和方向来控制电机的转速和转向。它包含一个电子调速器,用于调节电机的转速。本项目小车采用 1 高品质的有刷防水电调进行双向调节,同时配备特种散热片进行散热,连接碳刷电机,采用高性能防水碳刷电机,发热小,寿命长,功率大,性能稳定。电调主要有两个作用:一是将电池降压,适合接收机和其他舵机的工作电压;二是从接收机获得油门信号,控制马达的转速,从而改变小车轮子的速度。





图 7 本项目采取的有刷电调和碳刷电机硬件图

4. 舵机，舵机的工作原理：舵机常用的控制信号是一个周期为 20 毫秒左右，宽度为 1 毫秒到 2 毫秒的脉冲信号。当舵机收到该信号后，会马上激发出一个与之相同的，宽度为 1.5 毫秒的负向标准的中位脉冲。之后二个脉冲在一个加法器中进行相加得到了所谓的差值脉冲。输入信号脉冲如果宽于负向的标准脉冲，得到的就是正的差值脉冲。如果输入脉冲比标准脉冲窄，相加后得到的肯定是负的脉冲。此差值脉冲放大后就是驱动舵机正反转的动力信号。舵机电机的转动，通过齿轮组减速后，同时驱动转盘和标准脉冲宽度调节电位器转动。直到标准脉冲与输入脉冲宽度完全相同时，差值脉冲消失时才会停止转动。本校车中，舵机和电机会利用固定连接杆来连接轮子。



图 8 本项目采取的舵机和连接杆部分

5. USB 摄像头，本项目中将 USB 摄像头与树莓派 4B 主板相连接，来完成小车的视觉功能部分，重点的是路线监控以及拍照功能，我们选用的是 720P 的海康威视摄像头。



图 9 本项目选取的海康威视 720P 摄像头硬件图

6. USB 音箱，本项目中将 USB 音箱与树莓派 4B 主板相连接，来完成小车的语音输出部分。



图 10 本项目选取的树莓派 4BUSB 扬声器硬件图

7. USB 机械键盘和树莓派 4B 适配显示屏，两者作为项目的调试工具，例如辅助完成树莓派 4B 与计算机的远程连接，查看树莓派 IP 等辅助操作的完成，显示屏需要搭配高清 HDMI 传输线的显示屏配备电源线。



图 11 本项目调试所用的机械键盘和树莓派显示屏

## 4、供电

1. Donkeycar 小车供电系统，小车的舵机电机等采用外接锂电池充电。具体的电池的型号是 HSP 1/16 1/18 车模原装电池 7.2V 1100 毫安镍氢电池组(小田宫插头)，同时配备同型号的充电器。



图 12 本项目采用的 HSP 型号电池组硬件图

2. 对于树莓派 4B 的主板供电，我们利用树莓派 Type-C 供电口来给树莓派进行供电，其中移动电源选择 XiaoMi 充电宝，为了满足树莓派 4B 的 5V 的供电要求，我们选取的充电宝的型号是:小米充电宝 10000 海岸大容量 22.5W 轻薄小巧便携快充移动电源 PD20W。



图 13 本项目采用的 XiaoM 充电宝硬件图

3. 对于 GPS 拓展板以及 PCA9685 和摄像头音箱等其他外设，我们利用树莓派

的相应的管脚进行供电。最后总结也就是说，我们项目小车的供电总来源还是基于这个锂电池和这个移动充电宝。

## 5、接口及连接:(接口之间都采用杜邦线进行连接)

1. 小车固有配套硬件的连接，这里我们将详细描述小车的电调（Electronic Speed Control, ESC）与电机之间的连接；电调与电池的连接，电调与 PCA 驱动芯片(接收机的连接)。

电调与电机的连接	电调的输出线（有刷两根、无刷三根）与电机连接；（在我们用到的有刷模型中，是红色和黑色的两根比较粗的线）
电调与接收机的连接	电调的信号线（三根红色，白色和黑色的较细的线）与接收机（PCA 驱动芯片）连接。
电调与电池的连接	电调的输入线与电池连接即可，红色线对应红色线，黑色线对应黑色线，如果接反了会接入不进去。

2. 舵机与 PCA9685 驱动板的连接，舵机和电调的连接。

舵机与 PCA9685 驱动板的连接	将输出的三线与对应的 PCA 驱动板引脚连接。电调和舵机都是标准 3 线母插头连接只要按照对应的引脚插入驱动板就可以了。（地线一般为黑色或棕色、信号线一般为黄色或白色）。
舵机与电调的连接	电调和舵机都是标准 3 线母插头连接只要按照对应的引脚插入驱动板就可以了。（地线一般为黑色或棕色、信号线一般为黄色或白色，所以对应黑色的线要插在 GND 对应的针头）。



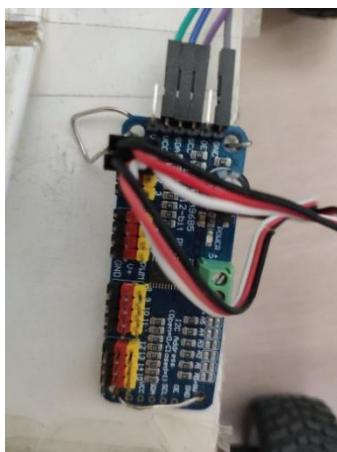


图 14 PCA9685 与舵机三母线和电调三母线的连接  
(注意 Channel\_0 和 Channel\_1 哪接的是电调，哪个是舵机)

### 3. 树莓派 4B 的 TF 存储卡等连接。

我们下面给出树莓派 4B 的主板的详细图解以及相关接口的定义：



图 15 树莓派 4B 主板的详细图解

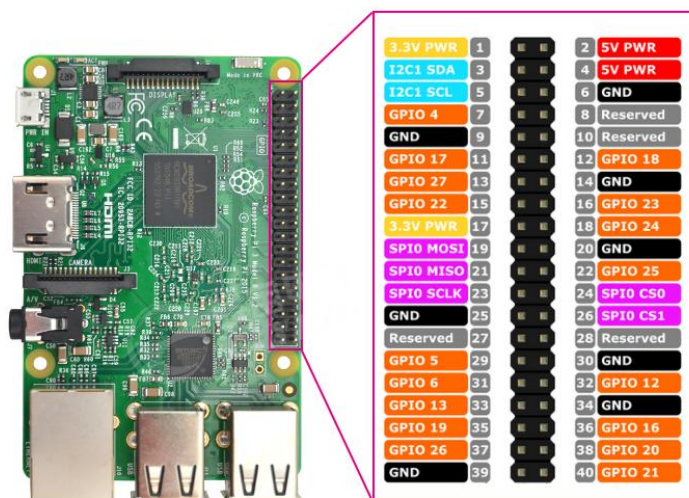


图 16 树莓派 4B 的相关接口的定义

树莓派 4B 和移动充电宝的连接	USB 电源接口接入充电宝电源，要利用到一根 Type-C 充电线，通过 USB-C 接口连接充电宝 USB 接口 (microusb) 供电 (Power supply)。
树莓派 4B 和 USB 摄像头的连接	树莓派 4B USB 2 或者 USB 3 中的四个接口中任意选择一个连接 USB 摄像头的 USB。
树莓派 4B 和 USB 音箱的连接	树莓派 4B USB 2 或者 USB 3 中的四个接口中任意选择一个连接树莓派 4B USB 音箱的 USB。
树莓派 4B 与 PCA9856 的连接	只连四根线，3.3V，引脚 SDA 和 SCL 和地线 Ground。 <div style="margin-left: 150px;"> <math>\left\{ \begin{array}{l} \text{GND} \rightarrow \text{RPi GND (9 脚)} \\ \text{SCL} \rightarrow \text{RPi SCL1 (5 脚)} \\ \text{SDA} \rightarrow \text{RPi SDA1 (3 脚)} \\ \text{VCC} \rightarrow \text{RPi 3.3V (1 脚)} \\ \text{V+} \rightarrow \text{RPi 5V} \end{array} \right.</math> </div>



图 17 树莓派 4B 和 PCA9856 的实物连接图

树莓派 4B 主板上考虑安装低高适配树莓派风扇的散热风扇和散热板，用来给树莓派进行供电，电风扇红色线和蓝色线接树莓派 2, 4 5VPWR 管脚，黑色地线接 6 号 GND 管脚。

4. 树莓派和微雪 GPS 拓展板高精度之间的连接以及 GPS 拓展板和 GPS 天线之间的连接。这个接入比较简单，但是 GPS 拓展板需要提前将配套的双排排针进行焊接，焊接时要细致入微，由于管脚之间的间隔非常小，因此要防止将管脚焊在一起，同时电烙铁要注意温度和位置，以防止将板子上贴片电容

等元器件烫坏，天线接口比较脆弱，建议用热熔机和胶布固定。防止接口撕裂。

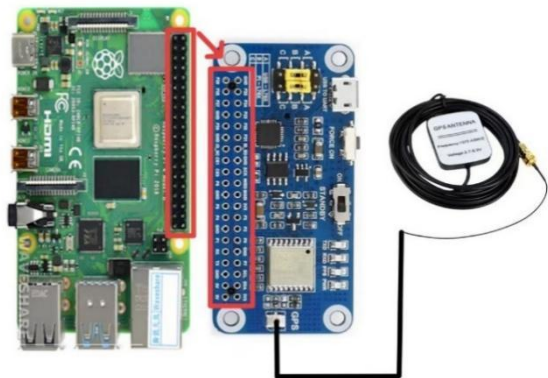


图 18 树莓派 4B 主控制板和 GPS 扩展板以及天线之间的连接

5. 中层主控系统总体电路图我们给出如下所示：

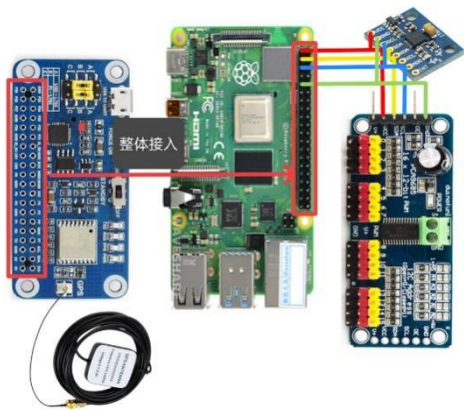


图 19 中层主控系统总体电路图(接线)

6、购买清单：(相关硬件介绍已经在前方介绍过)

硬件名称	价 格 /元	链接/网址
树莓派 4 代 B 型 RaspberryPi4 4B 8GB 开发板编程 AI 入门套 件 Python (8GB)	464	<a href="https://item.taobao.com/item.htm?id=619656726932">https://item.taobao.com/item.htm?id=619656726932</a>
DonkeyCar 树莓派 4B 深 度学习 AI 智能小车	1718	<a href="https://item.taobao.com/item.htm?id=717535501606">https://item.taobao.com/item.htm?id=717535501606</a>

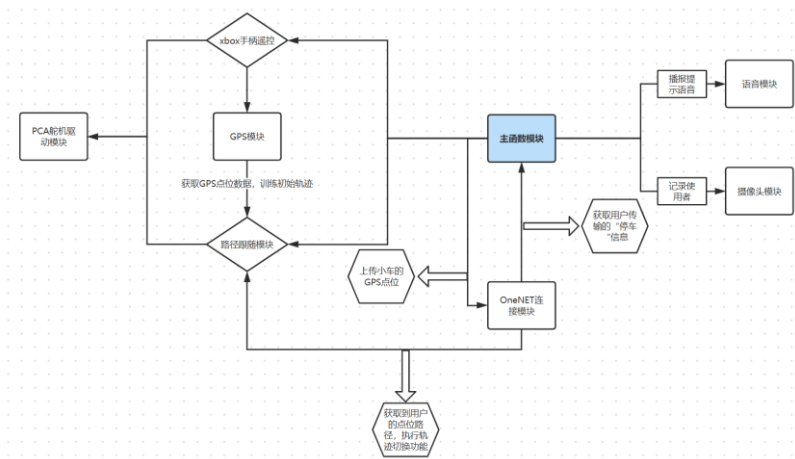
PCA9685 16 路 PWM Servo 舵机驱动板机器 人控制器 IIC 接口驱动 器模块	14. 50	<a href="https://item.taobao.com/item.htm?id=597498156699">https://item.taobao.com/item.htm?id=597498156699</a>
杜邦线 母对母 公对母 公对公 40P 彩色排线连 接线 10/20/30CM	视情 况所 需	<a href="https://item.taobao.com/item.htm?id=597119754200">https://item.taobao.com/item.htm?id=597119754200</a>
微雪 树莓派双频 RTK 厘米级高精度定位 GPS 模块 LC29H GNSS 扩展 板(型号 DA)	355	<a href="https://item.taobao.com/item.htm?id=736388381095">https://item.taobao.com/item.htm?id=736388381095</a>
海康威视 USB 摄像头 720P 基础性价款	74	<a href="https://item.taobao.com/item.htm?id=644467575802">https://item.taobao.com/item.htm?id=644467575802</a>
树莓派 4B 散热器 铝合 金散热风扇 Raspberry Pi 4B PWM 调速风扇	19. 9	<a href="https://item.taobao.com/item.htm?id=732140193028">https://item.taobao.com/item.htm?id=732140193028</a>
闪迪内存卡手机 32g/64g/128g/高速 tf 行车记录仪存储卡 sd switch 卡	29. 9	<a href="https://item.taobao.com/item.htm?id=43096943840">https://item.taobao.com/item.htm?id=43096943840</a>
树莓派扬声器 Raspberry Pi 4B/3B+ 小喇叭扬声器音响功放 板免驱动	21. 5	<a href="https://item.taobao.com/item.htm?id=721130671671">https://item.taobao.com/item.htm?id=721130671671</a>
小米充电宝 10000 毫安 大容量 22. 5w 轻薄小巧 便携迷你快充移动电源 pd20w 适用于苹果无线 大功率双向适配	66. 9	<a href="https://item.taobao.com/item.htm?umpChannel=tttj&amp;u_channel=tttj&amp;id=741004637452">https://item.taobao.com/item.htm?umpChannel=tttj&amp;u_channel=tttj&amp;id=741004637452</a>
MPU6050 模块 三维角度 传感器 6DOF 三轴加速 度计电子陀螺仪 GY- 521	11. 67	<a href="https://item.taobao.com/item.htm?id=729656168752">https://item.taobao.com/item.htm?id=729656168752</a>
HSP 1/16 1/18 车模原 装电池 7. 2V 1100 毫 安镍氢电组 小田宫插 头(配套充电器)	60	<a href="https://item.taobao.com/item.htm?id=529096266978">https://item.taobao.com/item.htm?id=529096266978</a>
创乐博树莓派 10 寸高	260	<a href="https://item.taobao.com/item.htm?id=599265218230">https://item.taobao.com/item.htm?id=599265218230</a>

清显示屏(附 HDML 线)		
飞利浦有线键盘鼠标套装 USB 办公游戏台式电脑笔记本通用打字机械	29.9	<a href="https://item.taobao.com/item.htm?id=732968109631">https://item.taobao.com/item.htm?id=732968109631</a>
5 路数字量循迹传感器 五路灰度巡线模块光电寻黑白红线识别比赛用	68	<a href="https://item.taobao.com/item.htm?id=642516141119">https://item.taobao.com/item.htm?id=642516141119</a>
HC-SR04 US-100 US-015 超声波模块距离测距模块超声波传感器电子	5	<a href="https://item.taobao.com/item.htm?id=41248598447">https://item.taobao.com/item.htm?id=41248598447</a>

### 2.1.2 软件

#### 1、软件模块

我们的小车是基于 DonKey Car 框架进行开发的,其中软件部分主要包括以下模块:  
PCA 电机驱动模块、GPS 模块、摄像头模块、语音模块、OneNET 连接模块、路径跟随模块以及主程序模块。具体连接情况如下:



#### 2、模块间接口

主函数与 OneNET 连接模块的接口:
<pre>g = get(path, origin_reset) g.start_polling() print("get success")</pre>

```
V.add(g, threaded=True, inputs=[], outputs=['utm_x_series', 'utm_y_series', 'signal'])
```

这里我们通过调用 `get` 函数建立与 OneNET 的连接，随后调用 `g.start_polling()` 创建一个新的线程来定时获取 `signal` 信号，若 `signal` 为 1，则说明用户点击了“叫车”按键，则需要根据 `output` 中的 `utm_*_series` 来进行路径跟踪，实现递送功能。

主函数与路径跟随模块的接口：

```
# This will use path and current position to output cross track error
cte = CTE(look_ahead=cfg.PATH_LOOK_AHEAD, look_behind=cfg.PATH_LOOK_BEHIND,
num_pts=cfg.PATH_SEARCH_LENGTH)
V.add(cte, inputs=['path', 'pos/x', 'pos/y', 'cte/closest_pt'], outputs=['cte/error',
'cte/closest_pt'],
run_condition='run_pilot')

# This will use the cross track error and PID constants to try to steer back towards the
path.
pid = PIDController(p=cfg.PID_P, i=cfg.PID_I, d=cfg.PID_D)
pilot = PID_Pilot(pid, cfg.PID_THROTTLE, cfg.USE_CONSTANT_THROTTLE,
min_throttle=cfg.PID_THROTTLE)
V.add(pilot, inputs=['cte/error', 'throttles', 'cte/closest_pt'], outputs=['pilot/steering',
'pilot/throttle'],
run_condition="run_pilot")
```

其中 CTE 主要用于计算小车当前方向与理想方向的偏差，并给出误差因子 **error**，接着传递给 `PID_Pilot` 函数进行计算，得到的 `steering` 值用于控制舵机进行转向，实现路径跟随。

PCA 舵机驱动模块与路径跟随模块的接口：

```
left_motor_speed = throttle
right_motor_speed = throttle

if steering < 0:
    left_motor_speed *= (1.0 - (-steering))
elif steering > 0:
    right_motor_speed *= (1.0 - steering)
```



这里就是路径跟随模块驱动多级的主要代码，其中，**steering** 值是我们之前调用函数计算而得出的转向值，在这里，我们判断转向值的正负，从而修正小车的运动轨迹。

主函数模块与摄像头模块和语音模块的接口：
<pre># 与摄像头模块的接口 add_camera(V, cfg, camera_type) # 与语音模块的接口 add_yuyiin()</pre>

对于摄像头，因为在初始化时已经配置了相关参数，所以我们只需调用函数给出配置参数和摄像头的类型，便可进行与摄像头模块的连接，后续可继续调用其拍照功能。对于语音模块，因为我们的车只需在固定情况下说固定的话即可（这里也可以考虑优化，不过应先确保基础功能实现）所以我们只需调用即可，无需传参。

### 3、软件流程

首先，我们在驱动模块中配置所有模块的初始化参数，如初始化舵机时，我们需要在 `config.py` 中配置与其相关的舵机转向参数和油门值参数，这里我们给出一些配置相关代码，以作说明：

### 路径跟随相关配置
<pre>PATH_FILENAME = "/home/yk/mycar/donkeycar.csv" # /home/yk/mycar/record/1.csv # the path will be saved to this # filename as comma separated x,y values PATH_DEBUG = True # True to log x,y position PATH_SCALE = 10.0 # the path display will be # scaled by this factor in the web page PATH_OFFSET = (255, 255) # 255, 255 is the center of the # map. This offset controls where the origin is displayed. PATH_MIN_DIST = 0.1 # after travelling this distance # (m), save a path point PATH_SEARCH_LENGTH = None # number of points to search # for closest point, None to search entire path PATH_LOOK_AHEAD = 2 # number of points ahead of the # closest point to include in cte track PATH_LOOK_BEHIND = 1 # number of points behind the # closest point to include in cte track</pre>

```

PID_P = 0.02          # proportional mult for PID path
follower
PID_I = 0.0           # integral mult for PID path
follower
PID_D = 0.0           # differential mult for PID path
follower
PID_THROTTLE = 0.4    # constant throttle value during
path following
USE_CONSTANT_THROTTLE = True    # whether or not to use the
constant throttle or variable throttle captured during path
recording
PID_D_DELTA = 0.25    # amount the inc/dec function
will change the D value
PID_P_DELTA = 0.25    # amount the inc/dec function
will change the P value

```

### ### gps 模块初始化

```

HAVE_GPS = True      # True to read gps position
GPS_SERIAL = '/dev/ttyUSB0' # serial device path, like
'/dev/ttyAMA1' or '/dev/ttyUSB0'
GPS_SERIAL_BAUDRATE = 115200
GPS_NMEA_PATH = None    # File used to record gps, like
"nmea.csv".

                        # If this is set then when waypoints are
recorded then

                        # the underlying NMEA sentences will
also be saved to

                        # this file along with their time
stamps. Then when

                        # the path is loaded and played in auto-
pilot mode then

                        # the NMEA sentences that were recorded
will be played back.

                        # This is for debugging and tuning the
PID without having

                        # to keep driving the car.
GPS_DEBUG = False    # set to True to log UTM position (beware; Lots
of Logging!)

```

### 舵机初始化参数

```
PWM_STEERING_THROTTLE = {  
    "PWM_STEERING_PIN": "PCA9685.1:40.1",    # PWM output pin for  
    steering servo  
    "PWM_STEERING_SCALE": 1.0,                # used to compensate for  
    PWM frequency differents from 60hz; NOT for adjusting steering  
    range  
    "PWM_STEERING_INVERTED": False,           # True if hardware requires  
    an inverted PWM pulse  
    "PWM_THROTTLE_PIN": "PCA9685.1:40.0",    # PWM output pin for ESC  
    "PWM_THROTTLE_SCALE": 1.0,                # used to compensate for  
    PWM frequence differences from 60hz; NOT for increasing/limiting  
    speed  
    "PWM_THROTTLE_INVERTED": False,           # True if hardware requires  
    an inverted PWM pulse  
    "STEERING_LEFT_PWM": 470,                 #pwm value for full left  
    steering  
    "STEERING_RIGHT_PWM": 310,                #pwm value for full right  
    steering  
    "THROTTLE_FORWARD_PWM": 450,              #pwm value for max forward  
    throttle  
    "THROTTLE_STOPPED_PWM": 380,              #pwm value for no movement  
    "THROTTLE_REVERSE_PWM": 220,              #pwm value for max reverse  
    throttle  
}
```

随后, 我们开始与 OneNET 建立通信连接, 并实时获取其上的参数作为变量, 其中包括:

- **signal** 信号作为用户是否点击“叫车”按钮标志, 用来确定小车的路径
- **stop** 信号作为用户是否点击“停车”按钮标志, 用来控制小车停止或行驶
- **utm\_x\_series** 数组作为小车获取到路径的经度坐标
- **utm\_y\_series** 数组作为小车获取到路径的纬度坐标

下面是相关部分代码:

```
def update(self):  
    while True:  
        self.poll()  
        time.sleep(self.poll_delay)
```

```

def poll(self):
    try:
        response = self.session.get(self.download_url,
headers=self.headers)
        response.raise_for_status()
        params = response.json()
        print("connect success")
        a=params['data']
        #print(params)
        if 'error' not in a:
            for datastream in a:
                if datastream['id'] == 'Location':
                    self.signal =
int(datastream['current_value']['signal'])
                    print(self.signal)
                elif datastream['id'] == 'xunhangline':
                    self.xunhangline_data =
datastream['current_value']['road_message']
                    print(self.xunhangline_data)
                # 检查经纬度数组是否同步更新，然后转换为 UTM 坐标
                # 更新 last_signal 状态

                self.last_signal = self.signal
    except requests.RequestException as e:
        print(f"请求出错: {e}")

def run_threaded(self):
    if self.xunhangline_data:
        # 提取 x 和 y 坐标的列表
        utm_x_series = [utm.from_latlon(point['latitude'],
point['longitude'])[0] for point in self.xunhangline_data]
        utm_y_series = [utm.from_latlon(point['latitude'],
point['longitude'])[1] for point in self.xunhangline_data]
        return utm_x_series, utm_y_series, self.signal
    else:
        return [], [], self.signal

```

```

def start_polling(self):
    thread = threading.Thread(target=self.update)
    thread.daemon = True
    thread.start()
    print("get start")

def shutdown(self):
    self.session.close()

```

与 OneNET 建立连接后，我们的小车会在初始规划路径上自动巡航，这就需要我们使用驱动模块来控制 PCA 舵机驱动模块，加载我们提前规划好的.csv 路径文件进行巡航，相关代码如下：

```

def load_path():

if signal==1:

path.load("onenet.csv")

print("The path was loaded was loaded from onenet.csv ")

else:

if os.path.exists(cfg.PATH_FILENAME) and
path.load(cfg.PATH_FILENAME):

print("The path was loaded was loaded from ", cfg.PATH_FILENAME)

# we loaded the path; also load any recorded gps readings

if gps_player:

gps_player.stop().nmea.load()

gps_player.start()

```

从代码中也可以得知，默认情况下，小车的巡航轨迹为“donkeycar.csv”，而一旦用

户按下“叫车”按键，相应 signal 的切换就会让小车切换轨迹到“onenet.csv”，这个文件就是我们从 OneNET 上获取到的到用户点位的规划路径，切换路径后，小车就会通过路径跟随算法，进行自动寻迹，如下是路径跟随相关代码：

```
def nearest_pt(self, path, x, y, from_pt=0, num_pts=None):
    from_pt = from_pt if from_pt is not None else 0
    num_pts = num_pts if num_pts is not None else len(path)
    num_pts = min(num_pts, len(path))
    if num_pts < 0:
        logging.error("num_pts must not be negative.")
    return None, None, None

    min_pt = None
    min_dist = None
    min_index = None
    for j in range(num_pts):
        i = (j + from_pt) % len(path)
        p = path[i]
        d = dist(p[0], p[1], x, y)
        if min_dist is None or d < min_dist:
            min_pt = p
            min_dist = d
            min_index = i
    return min_pt, min_index, min_dist

def nearest_two_pts(self, path, x, y):
    if path is None or len(path) < 2:
        logging.error("path is none; cannot calculate nearest points")
    return None, None

    distances = []
    for iP, p in enumerate(path):
        d = dist(p[0], p[1], x, y)
        distances.append((d, iP, p))
    distances.sort(key=lambda elem : elem[0])

    # get the prior point as start of segment
    iA = (distances[0][1] - 1) % len(path)
    a = path[iA]
```



```

# get the next point in the path as the end of the segment
iB = (iA + 2) % len(path)
b = path[iB]

return a, b

def nearest_waypoints(self, path, x, y, look_ahead=1,
look_behind=1, from_pt=0, num_pts=None):
    """
    Get the path elements around the closest element to the given
    (x,y)
    :param path: list of (x,y) points
    :param x: horizontal coordinate of point to check
    :param y: vertical coordinate of point to check
    :param from_pt: index start start search within path
    :param num_pts: maximum number of points to search in path
    :param look_ahead: number waypoints to include ahead of nearest
    point.
    :param look_behind: number of waypoints to include behind nearest
    point.
    :return: index of first point, nearest point and last point in
    nearest path segments
    """
    if path is None or len(path) < 2:
        logging.error("path is none; cannot calculate nearest points")
        return None, None

    if look_ahead < 0:
        logging.error("look_ahead must be a non-negative number")
        return None, None
    if look_behind < 0:
        logging.error("look_behind must be a non-negative number")
        return None, None
    if (look_ahead + look_behind) > len(path):
        logging.error("the path is not long enough to supply the
        waypoints")
        return None, None

```

```

_pt, i, _distance = self.nearest_pt(path, x, y, from_pt, num_pts)

# get start of segment
a = (i + len(path) - look_behind) % len(path)

# get the end of the segment
b = (i + look_ahead) % len(path)

return a, i, b

def nearest_track(self, path, x, y, look_ahead=1, look_behind=1,
from_pt=0, num_pts=None):
    """
    Get the line segment around the closest point to the given (x,y)
    :param path: list of (x,y) points
    :param x: horizontal coordinate of point to check
    :param y: vertical coordinate of point to check
    :param from_pt: index start start search within path
    :param num_pts: maximum number of points to search in path
    :param look_ahead: number waypoints to include ahead of nearest
    point.
    :param look_behind: number of waypoints to include behind nearest
    point.
    :return: start and end points of the nearest track and index of
    nearest point
    """

    a, i, b = self.nearest_waypoints(path, x, y, look_ahead,
look_behind, from_pt, num_pts)

    return (path[a], path[b], i) if a is not None and b is not None
    else (None, None, None)

def run(self, path, x, y, from_pt=None):
    """
    Run cross track error algorithm
    :return: cross-track-error and index of nearest point on the path
    """

```

```

"""
cte = 0.
i = from_pt

a, b, i = self.nearest_track(path, x, y,
look_ahead=self.look_ahead, look_behind=self.look_behind,
from_pt=from_pt, num_pts=self.num_pts)

if a and b:
logging.info(f"nearest: ({a[0]}, {a[1]}) to ({x}, {y})")
a_v = Vec3(a[0], 0., a[1])
b_v = Vec3(b[0], 0., b[1])
p_v = Vec3(x, 0., y)
line = Line3D(a_v, b_v)
err = line.vector_to(p_v)
sign = 1.0
cp = line.dir.cross(err.normalized())
if cp.y > 0.0 :
sign = -1.0
cte = err.mag() * sign
else:
logging.info(f"no nearest point to ({x},{y})")
return cte, i

```

在小车到达用户所在地时，用户点击“停车按键”后，小车停止，随后开始播报“欢迎使用”（之类的常用语言）。这时，用户即可取走小车上的充电宝，并且，在这个过程中，我们会调用摄像头对用户进行记录，以防一些不法行为，相关代码如下：

### ### 语音模块

```

def yuyin():
    args = docopt(__doc__)
    cfg = dk.load_config()
    tubs = args['--tubs']
    model = args['--model']
    model_type = args['--type']
    comment = args['--comment']
    train(cfg, tubs, model, model_type, comment)

```

### ### 摄像头模块

```
def UartReceiveDate():
    global receive1
    global receive2
    global receive3
    global receive4
    global data
    data[0] = ser.read(1) # 读取一个字节的数据
    data[1] = ser.read(1) # 读取一个字节的数据
    data[2] = ser.read(1) # 读取一个字节的数据
    data[3] = ser.read(1) # 读取一个字节的数据
    data[4] = ser.read(1) # 读取一个字节的数据
    data[5] = ser.read(1) # 读取一个字节的数据
    if data[0] == b'\xFF' and data[5] == b'\xFE': # 检测这次发送过
来的帧的帧头和帧尾
        receive1 = data[1] # 后面根据需要名字随便改
        receive2 = data[2]
        receive3 = data[3] # 后面根据需要名字随便改
        receive4 = data[4]
        print(receive1, receive2, receive3, receive4)
        #if(receive1 != b'\x00' or receive2 != b'\x00'):
        print("接收成功")
        ser.flushInput() # 清空缓存

cap = cv2.VideoCapture(0)

cap.set(3,640)
cap.set(4,480)

#ret, frame = cap.read()
#rows, cols, channels = frame.shape
#print(cols, rows, channels)
name = 220
```

至此，流程结束。

## 4、软件安装及运行环境

在配置 DonKey Car 时，我们参考了其[官方文档](#)。树莓派所使用的 Python 环境为 3.11.2。

具体按照以下步骤进行：

首先创建虚拟环境安装所需要的库

```
python3 -m venv env --system-site-packages ↵  
echo "source ~/env/bin/activate" >> ~/.bashrc ↵  
source ~/.bashrc↵
```

安装所需的库↵

```
sudo apt install libcap-dev libhdf5-dev↵
```

安装 Donkeycar Python 代码，用户安装，↵

使用临时指定清华镜像源的方法↵

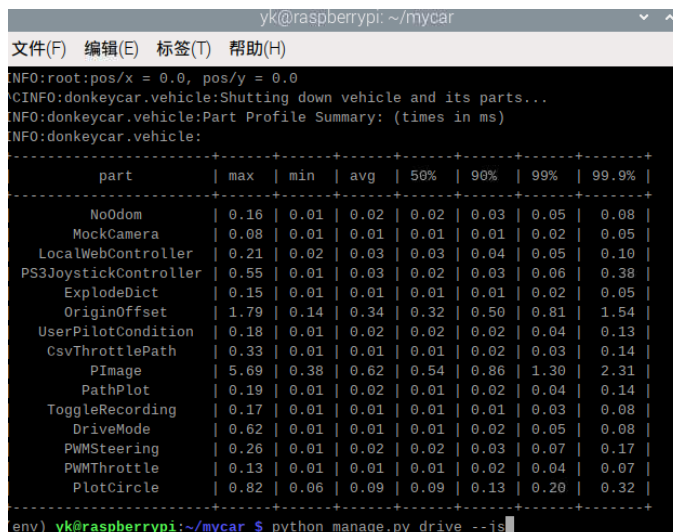
```
pip install donkeycar[pc]  
https://pypi.tuna.tsinghua.edu.cn/simple↵
```

然后我们创建 GPS 路径跟踪模板的汽车；

```
donkey createcar --template=path_follow --path ~/mycar↵
```

然后我们就开始小车的校正。

配置成功并且驱动小车成功：



```
yk@raspberrypi: ~/mycar  
文件(F) 编辑(E) 标签(T) 帮助(H)  
INFO:root:pos/x = 0.0, pos/y = 0.0  
INFO:donkeycar.vehicle:Shutting down vehicle and its parts...  
INFO:donkeycar.vehicle:Part Profile Summary: (times in ms)  
INFO:donkeycar.vehicle:  
+-----+-----+  
| part | max | min | avg | 50% | 90% | 99% | 99.9% |  
+-----+-----+  
| NoOdom | 0.16 | 0.01 | 0.02 | 0.02 | 0.03 | 0.05 | 0.08 |  
| MockCamera | 0.08 | 0.01 | 0.01 | 0.01 | 0.01 | 0.02 | 0.05 |  
| LocalWebController | 0.21 | 0.02 | 0.03 | 0.03 | 0.04 | 0.05 | 0.10 |  
| PS3JoystickController | 0.55 | 0.01 | 0.03 | 0.02 | 0.03 | 0.06 | 0.38 |  
| ExplodeDict | 0.15 | 0.01 | 0.01 | 0.01 | 0.01 | 0.02 | 0.05 |  
| OriginOffset | 1.79 | 0.14 | 0.34 | 0.32 | 0.50 | 0.81 | 1.54 |  
| UserPilotCondition | 0.18 | 0.01 | 0.02 | 0.02 | 0.02 | 0.04 | 0.13 |  
| CsvThrottlePath | 0.33 | 0.01 | 0.01 | 0.01 | 0.02 | 0.03 | 0.14 |  
| PImage | 5.69 | 0.38 | 0.62 | 0.54 | 0.86 | 1.30 | 2.31 |  
| PathPlot | 0.19 | 0.01 | 0.02 | 0.01 | 0.02 | 0.04 | 0.14 |  
| ToggleRecording | 0.17 | 0.01 | 0.01 | 0.01 | 0.01 | 0.03 | 0.08 |  
| DriveMode | 0.62 | 0.01 | 0.01 | 0.01 | 0.02 | 0.05 | 0.08 |  
| PWMSteering | 0.26 | 0.01 | 0.02 | 0.02 | 0.03 | 0.07 | 0.17 |  
| PWMThrottle | 0.13 | 0.01 | 0.01 | 0.01 | 0.02 | 0.04 | 0.07 |  
| PlotCircle | 0.82 | 0.06 | 0.09 | 0.09 | 0.13 | 0.20 | 0.32 |  
+-----+-----+  
env) yk@raspberrypi:~/mycar $ python manage.py drive --js
```

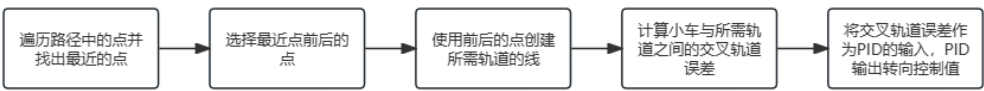
### 2.1.3 算法

#### 1、路径跟随算法

##### 功能

路径跟随算法的主要功能是使智能小车能够沿着预先设定或动态生成的 GPS 路径进行行驶。首先，该算法解析路径点列表，并根据小车的当前位置与目标点的距离和方

向，计算行驶的方向和速度。通过方向调整和速度控制，小车能够逐步接近每一个目标点，直到到达最终目标。算法还包括到达检测机制，以判断是否已经到达终点或用户指定位置。此外，路径跟随算法支持动态路径更新，并具备异常处理能力，如在 GPS 信号丢失或小车偏离路径时进行纠正，从而确保小车能够稳定、安全地完成任任务。如下是路径跟随算法的流程图：



## 输入输出

- ◇ **输入：**
- **路径点列表：** 小车需要沿行的路径，由一系列 GPS 坐标点组成
  - **小车当前 GPS 坐标：** 通过 GPS 模块实时获取，用于确定小车的当前位置
  - **目标点信息：** 算法中当前目标点的 GPS 坐标，即小车正在前往的路径点
  - **用户指令（“叫车/停车”信号）：** 用于动态更新路径或触发停止命令。
  - **新路径更新：** 在接收到新路径时，进行“停车”的操作，并沿新路径重新开始路径跟随。
- ◇ **输出：**
- **方向调整：** 根据小车当前位置与目标点的方向差异，输出调整后的行驶方向。
  - **到达指示：** 输出是否已到达目标点或终点，用于切换下一个目标点或停止小车。

## 数据集

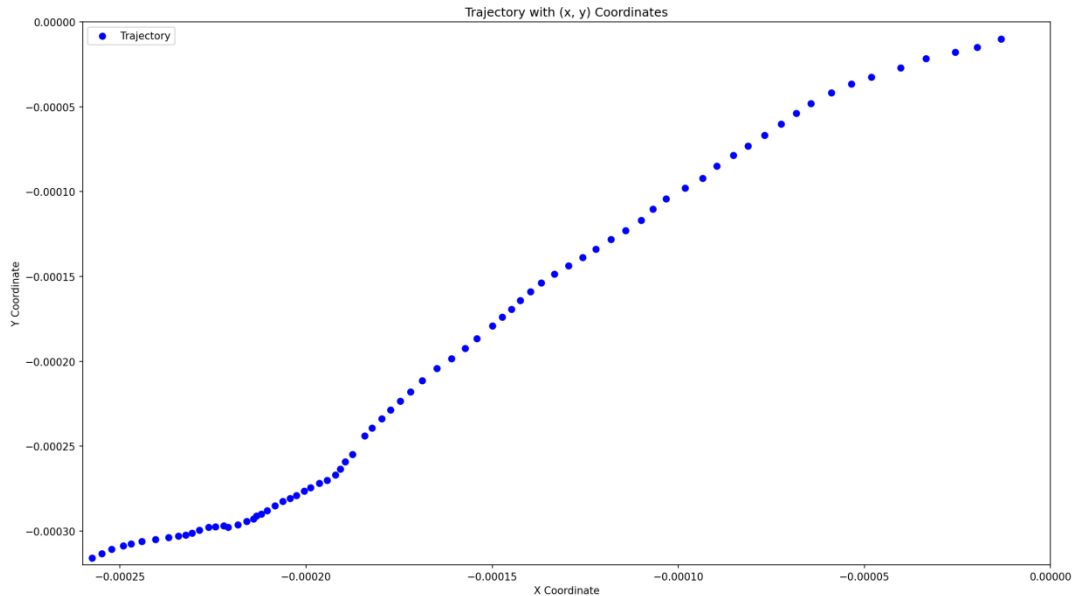
目前我们所使用的数据是通过手柄控制进行记录的路径点，并保存在一个.csv 文件中，方便后续训练，如下是一些测试的数据：

x 坐标	y 坐标	油门值
0	0	0
-1.97E-05	-1.48E-05	0.2
-2.57E-05	-1.78E-05	0.4
-3.35E-05	-2.17E-05	0.4
-4.03E-05	-2.70E-05	0.4
-4.82E-05	-3.25E-05	0.235066
-5.35E-05	-3.67E-05	0.4
-5.88E-05	-4.18E-05	0.387646
-6.43E-05	-4.80E-05	0.329917



-6.83E-05	-5.37E-05	0.37528
-7.23E-05	-6.02E-05	0.367028
-7.68E-05	-6.68E-05	0.387646
-8.13E-05	-7.32E-05	0.4
-8.52E-05	-7.85E-05	0.4
-8.97E-05	-8.48E-05	0.362902
-9.35E-05	-9.20E-05	0.338157
-9.82E-05	-9.80E-05	0.362902
-0.000103333	-0.000104167	0.387646
-0.000106833	-0.000110333	0.371154
-0.00011	-0.000117	0.391772
-0.000114167	-0.000122833	0.4
-0.000118167	-0.000128167	0.4
-0.000122167	-0.000134	0.391772
-0.000125667	-0.000138833	0.4
-0.0001295	-0.000143667	0.074233
-0.000133333	-0.0001485	0.4
-0.000136833	-0.000153667	0.4
-0.000139667	-0.000158833	0.4
-0.0001425	-0.000164167	0.4
-0.000144833	-0.000169167	0.379394
-0.000147333	-0.000173833	0.4
-0.00015	-0.000179	0.4
-0.000154167	-0.000186667	0.395898
-0.000157333	-0.000192333	0.371154
-0.000161	-0.0001985	0.367028
-0.000164833	-0.000204167	0.4
-0.000168833	-0.000211333	0.4
-0.000172	-0.000217833	0.346409
-0.000174667	-0.0002235	0.4
-0.000177333	-0.000228667	0.334031
-0.000179667	-0.000233833	0.4
-0.000182333	-0.000239167	0.4
-0.000184333	-0.000244	0.395898

对应路径图如下：



随后，我们需要将这个.csv 文件加载到小车的运行路径当中，通过路径跟随算法训练小车，从而得到合适的 PID 值。

## 性能

### ◇ 路径跟随精度

在路径跟随算法中，PID 控制器通过调节比例（P）、积分（I）、微分（D）三个参数，来调整小车的方向和速度。其中，**P 值**决定了小车对路径偏差的敏感度，较高的 P 值可以快速纠正误差，但过高会导致震荡；**I 值**通过累积误差来修正长期偏差，适合消除系统的稳态误差，但过高可能导致系统反应过慢；**D 值**通过预测误差的变化趋势来减缓快速变化的误差，有助于减少超调现象，但过高会使系统对噪声过于敏感。在代码中，具体配置如下：

```
#
# 路径跟随相关配置
#
PATH_FILENAME = "/home/yk/mycar/donkeycar.csv" #/home/yk/mycar/record/1.csv # the p
PATH_DEBUG = True # True to log x,y position
PATH_SCALE = 10.0 # the path display will be scaled by this factor
PATH_OFFSET = (255, 255) # 255, 255 is the center of the map. This offset
PATH_MIN_DIST = 0.1 # after travelling this distance (m), save a path poi
PATH_SEARCH_LENGTH = None # number of points to search for closest point,
PATH_LOOK_AHEAD = 2 # number of points ahead of the closest point to i
PATH_LOOK_BEHIND = 1 # number of points behind the closest point to in
PID_P = 0.02 # proportional mult for PID path follower
PID_I = 0.0 # integral mult for PID path follower
PID_D = 0.0 # differential mult for PID path follower
PID_THROTTLE = 0.4 # constant throttle value during path following
USE_CONSTANT_THROTTLE = True # whether or not to use the constant throttle or
PID_D_DELTA = 0.25 # amount the inc/dec function will change the D
PID_P_DELTA = 0.25 # amount the inc/dec function will change the P
```

从目前所训练的情况来看，我们小车的路径跟随精度还是比较高的，可以很好

地按照设定的 GPS 路径进行行驶。通过 PID 参数的调整，小车在预设路径上表现稳定，能够迅速响应并纠正偏差，不会产生过度震荡。同时，在实际测试中，小车能够平稳完成目标点的到达，且在动态路径更新后，能够快速重新规划路径并继续跟随。总体而言，现有的控制算法已经具备较好的精度和稳定性，能够满足项目的基本要求。

#### ✧ 响应时间

在目前的测试中，小车的响应时间表现良好。因为我们的整体系统使用的是多线程并进的结构，这就使得小车能够迅速反应从 OneNET 上获取到的用户指令，调整行驶路径。经过反复测试，小车能够在数秒内完成路径的重新规划并开始沿新路径行驶。同时，对于用户通过微信小程序发出的指令，小车也能快速给出反馈。总体来看，当前算法的响应时间符合项目预期要求，能够提供较好的用户体验。

#### ✧ 算法效率

在路径跟随算法的实现中，算法效率表现出色。对于树莓派这样资源有限的微型计算机来说，一旦算法的 CPU 和内存占用率较高，就会出现程序卡顿、树莓派芯片过热等情况，甚至是程序控制失灵（这些都是我们在测试时实际遇到过的，所以后来才买了一个风扇降温）。对于我们的路径跟随算法，其 CPU 和内存占用率可以接受，在程序运行时可以长时间保持高效的路径跟随能力。

## 2.1.4 系统外观及 UI

### 1、系统外观

### 2、操作 UI

我们的用户界面（UI）是基于小程序构建的，旨在优化用户交互体验。在设计过程中，我们精心考虑了界面的交互性、功能性和美观性，确保能够全面满足用户的使用需求。该界面不仅是用户与小车之间沟通的桥梁，更是实现双方互动的关键。用户通过小程序向小车发送指令，小车接收到指令后将执行相应的操作，从而实现用户与小车的无缝交互。此外，我们深入考虑了小车在实际使用中的多种场景。为了应对多个用户同时使用小车的情况，我们特别设计了一套预约排队系统。当小车正在被一个用户使用，其他用户可以通过预约系统进行排队，待当前用户使用完毕后，系统将自动安排下一个用户与小车进行交互。这一设计不仅提高了小车的使用效率，还确保了用户体验的连贯性和满意度。通过我们的精心设计，用户可以享受到更加便捷、高效的服务体验。

操作 UI 主要分为两个部分，一个主页面和一个子页面

**主页面：**主页面按照小车可不可以使用表现为两种不同的状态，主页面主要有三部分组成，小车可以使用和不可以使用时显示的文字，小车巡航路线地图，和小

## 车功能控制按钮

- 1、小车可以使用，页面对应的图片会亮起，并显示小车距离用户的距离和大致到达时间具体代码如下表所示：

小车可以使用文字显示

```
<!-- 小车巡航 -->

<view wx:if="{{car_can_use}}" style="background-
color: #EFEFEF;padding: 5px 10px;margin-left: 10px;margin-
right: 80px;color: #7a2929;margin-top: 5px;">
  小车状态： 小车巡航中,欢迎使用！
</view>

<!-- 可以使用文字显示 -->
<view wx:if="{{car_can_use}}">
  <view style="display: flex;margin-left: 20px;justify-content: space-
between;margin-bottom: 10px;">
    <view style="color: #9A9A9A;">
      小车距离您
    </view>
    <view style="color: #6C6C6C;margin-right: 10px;direction: rtl;">
      {{distance}}m
    </view>
  </view>
  <view style="display: flex;margin-left: 20px;justify-content: space-between;margin-
bottom: 10px;">
    <view style="color: #9A9A9A;">
      小车预计到达所需时间
    </view>
    <view style="color: #6C6C6C;margin-right: 10px;direction: rtl;">
      {{WaitTime}}分钟
    </view>
  </view>
</view>

</view>
```

小车不能被使用时或正在被使用时，主页面会显示当前用户需要等待的人数，和预约按钮，具体见下列代码。

```
<view wx:if="{{!car_can_use}}" style="background-
color: #EFEFEF;padding: 5px 10px;margin-left: 10px;margin-
right: 40px;color: #7a2929;margin-top: 5px;">
  小车状态： 小车送宝中，请排队预约哟！
</view>

<!-- 不可以使用文字显示 -->
<view wx:if="{{!car_can_use}}">
  <view style="display: flex;margin-left: 20px;justify-content: space-
```

```

between;margin-bottom: 10px;">
  <view style="color: #9A9A9A;">
    您当前所需等待人数
  </view>
  <view style="color: #6C6C6C;margin-right: 10px;direction: rtl;">
    {{needWaitPeople}}人
  </view>
</view>
<view wx:if="{{!car_can_use}}" style="display: flex;">
<!-- 预约按钮 -->
  <button size="mini" type="primary" bind:tap="bookingCar" style="border-
radius: 30px">{{identification==2?'取消预约':'点击预约'}}</button>
</view>
</view>

```

同时小程序主页面还会根据小车的状态显示不同的图片，来使得小程序的交互界面更加有趣，丰富，这里就不展示具体代码。

- 2、显示玩小车具体信息的下方是显示小车，用户，充电桩位置坐标，以及小车巡航路线的地图。采用 map 组件编写，先确定地图的中心点，以确保能在图中正确显示小车和用户之间的图标，后根据腾讯地图后台规划的 GPS 路线点，画出小车固定的巡航路线。

```

<map

  id="myMap"

  style="width: 100%; height: 300px;"

  latitude="{{latitude}}"

  longitude="{{longitude}}" 中心经纬度

  bindmarkertap="markertap"

  bindcallouttap="callouttap"

  bindlabeltap="labeltap"

  markers="{{markers}}"

  polyline="{{polyline}}" 路线

  scale="16"

>

</map>

```

- 3、地图下方是小车的功能卡选项，用户点击对应的功能卡，小程序会执行相应的

函数，对应功能为送还，巡航，停止，结构与送还结构类似，所以只展示送还结构代码

```
<view class="scene" bind:tap="tonav" style="margin-left: 0px;">

  <view style="width: 50px;height: 50px;display: flex;flex-
direction: column;justify-content: center;align-items: center;">

    <image src="cloud://xuehaolin-9gpfu9n5283fbe3e.7875-xuehaolin-
9gpfu9n5283fbe3e-
1314104796/bao/keyong.gif" style="height: 55px;width: 55px;" />

  </view>

  <view class="scene_text" style="margin-left: 10px;">

    送/还

  </view>
```

- 4、用户一进入主页面，需要显示地图，小车和用户的位置坐标，判断小车状态，获取用户列表等等操作，所以我们需要在监听函数 onload 函数里一一调用这些函数，初始化小程序，并获取必要的信息。一进入小程序主页面系统自动调用 Onload 函数，onload 函数通过云函数获取当前用户的 openId，再通过监听函数 onChange 调用小车状态判断函数，可以实时刷新小车状态。然后通过微信位置接口获取用户当前所在地的经纬度坐标，成功后调用一系列函数上传数据，画出路线和坐标点，最后通过轮询的方式，实时获取小车的坐标位置。

```
onLoad(e) {
  // 获取当前登录用户的_openid
  wx.cloud.callFunction({
    name: 'getUserInfo'
  }).then(res => {
    this.setData({
      openId: res.result.openid
    })
    // 判断用户页面的展示情况
    var that = this
    const watcher = db.collection('waitCarList').doc('user').watch({
      onChange: function (snapshot) {
        that.checkCarStatus()
      },
      onError: function (err) {
        console.error('the watch closed because of error', err)
      }
    })
  })
}
```



```

    })
    this.checkCarStatus()
  }).catch(err => {
    console.error('云函数调用失败', err)
  })

  // 获取用户所在位置的经纬度信息
  var that = this
  wx.getLocation({
    type: 'wgs84',
    isHighAccuracy: true,
  }).then(res => {
    // 修改展示页面的经纬度展示
    that.setData({
      // you_lon: baiduInstance.Lon_Google,
      // you_lat: baiduInstance.Lat_Google
    }),
    this.addMarker(); //添加地图坐标
    this.uploadzhuang(); //上传充电桩信息
    this.xunhangxian(); //绘制巡航线
    this.getDistanceAndTime(); //获取用户与小车之间的距离和时间
    this.getcarstation(); //获得小车位置
    wx.hideLoading();
    //循环调用位置信息函数 this.getcarstation(), 可以实时更新小车位置信息
    const interval = () => {
      timer = setTimeout(() => {
        // 执行代码块
        this.getcarstation()
        this.addMarker()
        interval()
      }, 500)
    }
    interval()
  })
},

```

- 5、小车状态判断函数，用户进入小程序后，通过 onload 函数调用小车状态判断函数，进入状态判断函数，先获取云端数据库中的用户列表 userList，然后根据用户列表是否为空，改变小车的状态。用户列表为空，设置小车为可用和巡航状态；用户列表不为零，则会在用户列表里寻找当前用户的 openId，并返回其在用户列表中的位置。当前用户不在其中时，设置小车为不可用和送宝状态，当前用户在第一位的时候，设置小车为不可用和送宝中，当前用户处于其他位置时，设置小车为不可用和预约中。小车状态判断函数通过 onchange 函数绑定了数据库，只要数据库中内容发生变化，就会调用此函数，更新小车的状态，前段主页面会根据小车的状态显示不同的视觉效果。

```

wx.showLoading({
  title: '信息获取中...',
})
wx.cloud.database().collection('waitCarList').where({
  _openid: 'oJPFn4o841lCQyL4lxd45YqqRDQl'
})
.get()
.then(res => {
  this.data.userList = res.data[0].userList
  if (!this.data.userList.length) {
    this.setData({
      car_can_use: 1,
      identification: 0
    })
    wx.hideLoading()
  } else {
    const signal = this.data.userList.findIndex(e => e === this.data.openId)
    if (signal == -1) {
      this.setData({
        car_can_use: 0,
        identification: 1,
        needWaitPeople: this.data.userList.length
      })
      wx.hideLoading()
    } else if (signal == 0) {
      this.setData({
        car_can_use: 0,
        identification: 1
      })
      wx.hideLoading()
    } else {
      var index = this.data.userList.findIndex(e => e === this.data.openId)
      this.setData({
        car_can_use: 0,
        identification: 2,
        needWaitPeople: index
      })
      wx.hideLoading()
    }
  }
  wx.hideLoading()
})

```

- 6、送宝函数的实现，点击下方送还按钮，会调用此函数。此函数先判断小车状态，如果小车处于巡航状态，那么会将当前用户的 openId 添加进用户列表然

后上传到云数据库，并将小车的状态设置为不可用和送宝中。

```
if (!this.data.identification) {
  this.data.userList.push(this.data.openId)
  db.collection('waitCarList').where({
    _openid: 'oJPFn4o841lCQyL4lxd45YqqRDQI',
  })
  .update({
    data: {
      userList: this.data.userList
    }
  }).then(res => {
    this.setData({
      identification: 1,
      car_can_use: 0
    })
  })
}
```

- 7、预约函数的实现：先判断当前的小车状态，因为一进入小程序已经调用过小车判断函数，所以已经获得了小车的状态和用户列表，所以可以直接判断。当小车处于排队状态时，从用户列表中移除当前用户的 openId，并更新到云数据库，实现取消预约的操作，当小车处于其他状态时，会将用户的 openId 添加到用户列表里，并且同时上传到云数据库，成功之后将小车的状态设置为预约中，这样用户下一次点击预约按钮，就会执行取消预约的操作。

```
// 是否预约小车
bookingCar(e) {
  console.log('identification:', this.data.identification)
  if (this.data.identification == 2) {
    this.data.userList = this.data.userList.filter(item => item !== this.data.openId)
    wx.cloud.database().collection('waitCarList').where({
      _openid: 'oJPFn4o841lCQyL4lxd45YqqRDQI'
    })
    .update({
      data: {
        userList: this.data.userList
      }
    })
    .then(res => {
      this.checkCarStatus()
      this.setData({
        identification: 1
      })
    })
  } else {
```

```

this.data.userList.push(this.data.openId)
wx.cloud.database().collection('waitCarList').where({
  _openid: 'oJpfn4o841lCQyL4lxd45YqqRDQl'
})
.update({
  data: {
    userList: this.data.userList
  }
})
.then(res => {
  this.checkCarStatus()
  this.setData({
    identification: 2
  })
})
}
},

```

- 8、绘制巡航线函数：进入小程序，通过 onload 函数调用执行此函数。由在腾讯地图控制台获得实现规划好的巡航线经纬度坐标点，按照固定键值对格式添加进数组中后传入 map 组件的 polyline 参数，设置好地图显示的中心点，这样之后，就能在主页面显示出小车的巡航路线了

```

//绘制巡航线信息
xunhangxian(e){
  var _this = this
  // 生成路径经纬度信息点
  var route_list = this.data.PL
  var pl = []
  // 将解压后的坐标放入点串数组 pl 中
  for (var i = 0; i < route_list.length; i += 2) {
    pl.push({
      latitude: route_list[i],
      longitude: route_list[i + 1]
    })
  }
  this.data.datapoint_xunhang["datastreams"][0]["datapoints"][0]["value"]["road_message"] = pl
  this.setData({
    // 将路线的起点设置为地图中心点
    latitude: pl[0].latitude,
    longitude: pl[0].longitude,
    // 绘制路线
    polyline: [{
      points: pl,

```

```

        color: '#54dde7',
        width: 6,
        borderColor: '#2f693c',
        arrowLine: true,
        borderWidth: 2
      }]
    })
  },

```

- 9、小车与用户距离时间函数：调用微信小程序位置信息请求接口，获取用户的经纬度坐标，然后通过 onenet 获取当前的小车经纬度坐标，然后通过调用腾讯地图 API，将小车的坐标点设置为起点，用户的坐标设置为重点，最终获取小车从当前位置出发到达用户所在点所需要的时间和与用户之间的距离。最终会显示在主页面上。

```

//获取用户与小车之间的距离
getDistanceAndTime(e) {
  wx.getLocation({
    type: 'wgs84',
    isHighAccuracy: true,
  }).then(res => {
    const baiduInstance = new mapTransfrom();
    baiduInstance.Google_Coordinates(res.latitude, res.longitude)
    var start = baiduInstance.Lat_Goodle + ',' + baiduInstance.Lon_Goodle
    console.log('start:', start)
    // 2、获取小车实时的经纬度信息
    wx.request({
      url: 'https://api.heclouds.com/devices/1100048518/datapoints',
      header: {
        "api-key": "d4Rs4uU=tusmD3Dmh3Kl1OcPvzg=" //自己的 api-key
      },
      method: "GET",
      success: res => {
        var end = res.data.data.datastreams[1].datapoints[0].value + ',' + res.data.data
        .datastreams[3].datapoints[0].value
        console.log('end:', end)
        // 3、发送 api 调用请求，获取当前的小车和用户的距离以及预计到达时间
        wx.request({
          url: 'https://apis.map.qq.com/ws/direction/v1/walking',
          data: {
            "from": start,
            "to": end,
            // "from": '39.961652,116.355849',
            // "to": '39.961850,116.357252',
            "key": "JOJBZ-EUPYQ-RUG5V-BK7Q6-ANOYV-3HFC6"
          },
        })
      }
    })
  })
}

```

```

    success: res => {
      this.setData({
        distance: res.data.result.routes[0].distance,
        WaitTime: res.data.result.routes[0].duration,
      })
      wx.hideLoading()
    }
  })
},
fail: err => {
  console.log(err, '获取数据失败')
}
})
}
},

```

- 10、下方功能列表：点击送换按钮会执行送宝函数并跳转到子页面，点击巡航和函数会向 onenet 发送小车的控制信息，控制小车的行动。

**子页面：**主要由一个地图和出发按钮组成，界面主要有几张图片装饰

- 1、出发按设计：出发按钮用一张图片代替按钮，点击出发按钮会先调用腾讯地图的 API，设置小车的经纬度坐标为起始点，人的经纬度坐标为终点，成功后按照固定的键值对格式将经纬度路线传入 p1 数组，在地图上画出规划后的小车行进路径，并把规划好的经纬度路径点 p1 传入 onenet 云平台上方便小车读取。

```

cargo(e){
  var _this = this
  // 生成路径经纬度信息点
  wx.request({
    url: 'https://apis.map.qq.com/ws/direction/v1/driving',
    data: {

      "from": this.data.road_start,
      "to": this.data.road_end,
      // "from": '39.961652,116.355849',
      // "to": '39.961850,116.357252',
      "speed": '0.001',
      "accuracy": '0.00001',
      "key": "JOJBZ-EUPYQ-RUG5V-BK7Q6-ANOYV-3HFC6",
    },
  },
  success: res => {
    console.log(res)
    this.getlonlaufrompi()
  }
}

```

```

var route_list = res.data.result.routes[0].polyline
var pl = [
  {latitude: Number(this.data.car_lat),
   longitude: Number(this.data.car_lon)}
]
// 坐标解压（返回的点串坐标，通过前向差分进行压缩）
var kr = 1000000;
for (var i = 2; i < route_list.length; i++) {
  route_list[i] = Number(route_list[i - 2]) + Number(route_list[i]) / kr;
}
// 将解压后的坐标放入点串数组 pl 中
for (var i = 0; i < route_list.length; i += 2) {
  pl.push({
    latitude: route_list[i],
    longitude: route_list[i + 1]
  })
}
pl.push({
  latitude: this.data.you_lat,
  longitude: this.data.you_lon
  // latitude: 39.961850,
  // longitude: 116.357252,
})
this.data.datapoint_xunhang["datastreams"][0]["datapoints"][0]["value"]["road_message"] = pl
console.log(pl)
this.uplonlaufrompi();
this.setData({
  // 将路线的起点设置为地图中心点
  latitude: pl[0].latitude,
  longitude: pl[0].longitude,
  waitTime : res.data.result.routes[0].duration,
  // 绘制路线
  polyline: [{
    points: pl,
    color: '#54dde7',
    width: 6,
    borderColor: '#2f693c',
    arrowLine: true,
    borderWidth: 2
  }]
})
},
})

```

```
}
```

## 2.1.5 验收时的演示方案

### 1、场地

我们预期将演示场地选在操场。因为在整个校园内，操场是少数几个开阔的露天场地之一，非常适合 GPS 模块的导航使用。同时，操场宽阔且平坦的地面为小车的高速行驶和机动测试提供了充足空间，能够最大程度上减少复杂地形对测试结果的干扰。此外，操场相对封闭的环境也有助于减少外界因素（如车辆等）的干扰，也确保了测试过程的安全性和精确性。

### 2、时间

截止目前，我们的实验进展还处于可控范围内，与理想中的进度相差不。在接下来的实验中，如果一切顺利的话，我们预期将在**第 6 周**进行验收。

### 3、必备条件要求

- **环境控制：**我们计划选择在天气晴朗、无强风的日子进行测试，尽量减少自然因素对 GPS 模块精度的影响，确保测试数据的可靠性。
- **设备检查：**在测试前，我们会确保所有硬件设备准备就绪，包括小车、电脑、鼠标、显示屏、手柄、电池等关键设施，避免因设备故障导致测试中断。
- **通信测试：**在测试开始前，我们会提前建立与 OneNET 的连接，进行通信测试，以确保数据传输的稳定性和可靠性，为正式测试提供保障。
- **巡航路径：**在测试前，我们会预先设计准备好小车的初始巡航路线，并对该路径进行可行性检查，确保路径符合实际环境条件，从而为测试的顺利进行做好充分准备。
- **GPS 信号监测：**在测试过程中，我们会实时监测 GPS 信号的强度和质量，避免因信号波动或干扰导致的小车偏离路径或失控，确保导航精度。
- **安全预案：**为确保测试安全，我们会提前制定应急预案，包括在突发情况下手动控制小车或终止测试的措施，避免可能出现的安全隐患。

### 4、预期流程

我们设想的测试流程如下：首先，将小车放置在起点位置，并检查所有硬件设置是

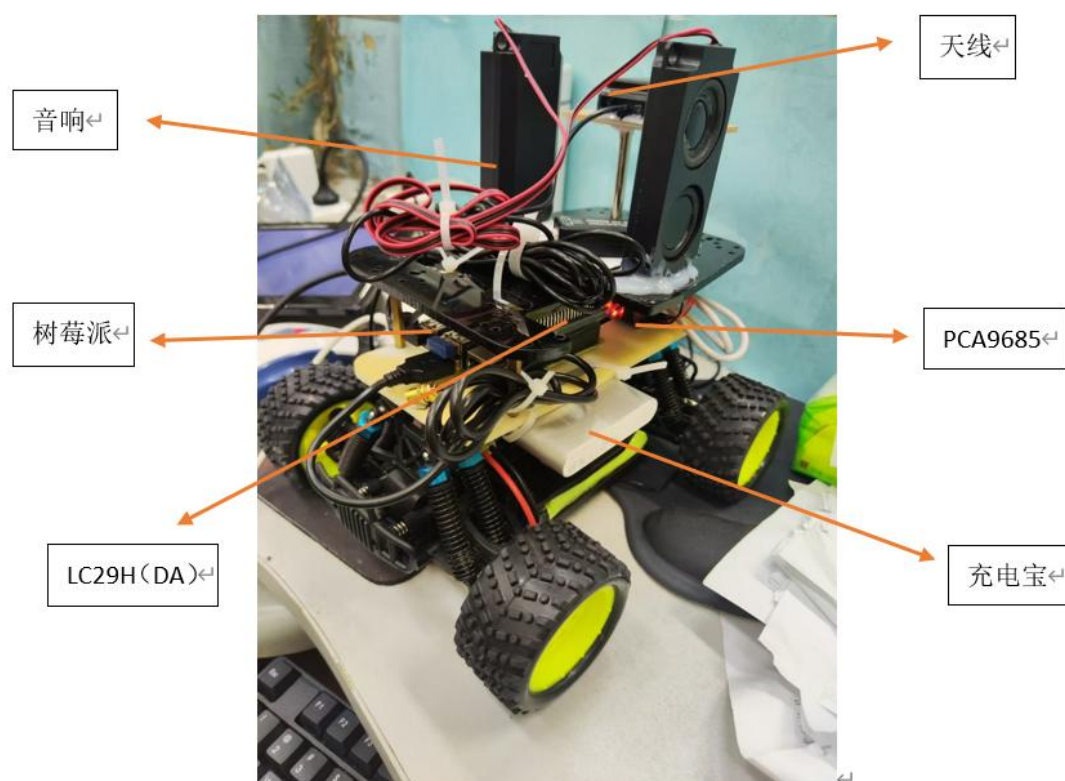


否正常。确认无误后，启动小车，小车将自动按照预设的轨道开始巡航。此时，GPS 模块会实时获取小车的当前位置，用于路径跟随，并将数据上传至 OneNET 平台。在巡航功能正常的情况下，用户可以通过微信小程序点击“叫车”按钮，预期情况下，小车将会停止当前巡航，转而根据从 OneNET 获取的新路径进行导航，前往用户所在位置。到达用户附近后，用户点击“停车”按钮，小车将停止行驶，进行语音播报，并对用户进行拍照并保存照片。在用户成功取走充电宝后，整个流程结束。

## 2.2 方案完成情况

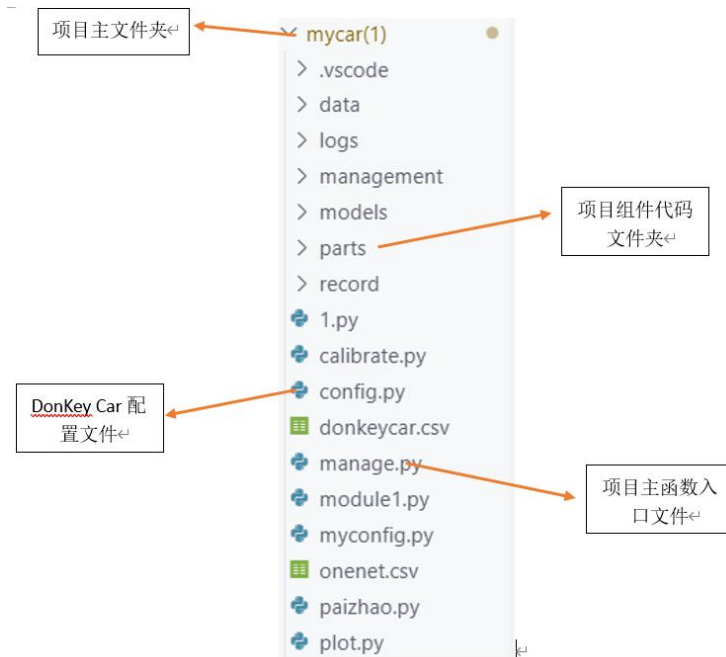
### 2.2.1 硬件

我们小组目前已经成功搭建了基础智能小车，整个硬件系统包括舵机驱动模块、树莓派、充电宝以及通过杜邦线连接的电调，这些核心部件确保了小车的基本动力和控制能力。此外，我们还为小车额外配备了 GPS 模块，实现精确的地理位置跟踪；添置了音响和风扇等附加组件，为系统提供了额外的交互能力和视觉效果。目前，小车硬件部分的搭建已经基本完成，所有组件的连接和基础测试也都顺利通过。接下来的工作将集中在进一步调试和优化这些硬件设备，确保各模块能够稳定运行，同时与我们的软件系统无缝对接，为后续的功能集成奠定基础。下面给出我们小车的成品图：

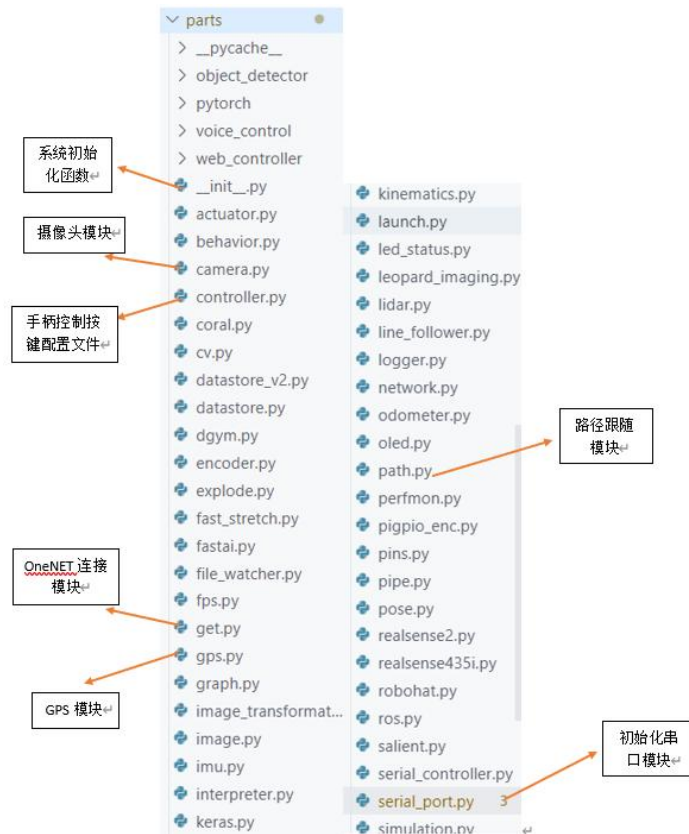


## 2.2.2 软件

软件部分，我们已经成功实现了树莓派与 OneNET 平台的连接模块，以及 GPS 模块、语音模块、摄像头模块和舵机驱动模块的代码编写工作。同时，我们通过将这些模块嵌入到主函数模块中，实现了各个功能模块之间的有效耦合和级联。此外，为了提升系统的运行效率和响应速度，我们构建了一个多线程系统，确保各个模块能够并行执行。这样设计可以有效提升整体系统的性能，为后续功能的进一步扩展打下了坚实基础。这里我们给出项目代码的整体架构，具体代码在上面已经提及，这里就不再赘述。



图：项目整体框架



图：项目组件代码文件夹

### 2.2.3 算法

算法部分，我们已经成功实现了路径跟随算法，该算法能够让小车按照预设路径点进行导航。经过测试表明，在 GPS 模块的精度足够的情况下（在静态条件下，GPS 模块的精度能够勉强满足要求），小车可以较为准确地执行路径跟随，从而实现基本的巡航功能。然而，在动态环境下，GPS 模块的精度会有所降低，导致路径跟随的准确性受到一定影响。我们接下来将针对这一问题进行进一步优化，可能会结合如卡尔曼滤波等其他误差校正算法和 RTK 等差分定位技术，来进一步提高路径跟随的稳定性和精度。

2.2.4 系统外观及 UI

最终用户交互小程序界面效果如下图所示：



图：巡航中的主

图：预约中的主页面

图：小程序子页面

2.3 合作分工

成员	负责的任务	完成情况
闫康	主要负责路径跟随算法的开发与优化，包括 GPS 数据的处理、路径规划以及控制算法的实现，确保小车能够准确跟随预定路线	已经完成了路径跟随算法代码的编写，并在初步的测试中取得了不错的反响。目前正在致力于研究提高 GPS 精度的算法。
孙常鑫	负责算法和系统的调试与测试，验证各模块功能的稳定性与准确性，进行故障排除和性能优化，确保系统在实际环境下能够顺利运行	目前正在进行系统集成测试，已经排除了一些潜在问题，但还是存在一些线程冲突的问题，目前正在进一步优化。
苏珈磊	负责小车硬件部分的搭建与调试，包括舵机驱动模块、	硬件搭建已完成，所有模块功能均通过了基本测试。目

	GPS 模块、摄像头、语音模块和其他附加组件。负责撰写项目文档，详细记录项目进展、技术实现和调试过程。	前正在进一步优化硬件功能并撰写项目文档，实时跟进记录项目进展。
薛皓林	负责微信小程序的设计、开发和调试，实现用户与小车的交互功能，包括实时位置监控、路线规划、停止车辆和跟随功能等。	小程序的基本功能已经实现，目前正在进行 OneNET 连接等功能的调试，确保与小车系统的无缝对接。

## 3. 后续工作计划

### 3.1 课题整体安排

#### 3.1.1 进度汇总

截止目前（第二周），我们小组已经完成了项目启动与需求分析阶段，明确了项目目标、需求以及各个模块的技术实现，并制定了详细的时间表和成员分工。在硬件搭建方面，我们完成了小车的基础结构，包括舵机驱动模块、树莓派、充电宝、电调的连接，并为小车配备了 GPS 模块、音响、摄像头和风扇灯等组件。与此同时，微信小程序已进入开发阶段，初步实现了用户与小车的交互功能。在软件部分，树莓派的各个模块代码编写已初步完成，并实现了路径跟随算法的基本功能。我们还完成了部分系统耦合，并进行了初步测试，验证了系统的基本稳定性和可行性。

#### 3.1.2 未来规划

在项目的第三至第六周，我们计划逐步推进并优化完善项目的各个模块。

在第三周，我们的工作重点将是优化算法，特别是针对 GPS 定位不精准的问题。我们将进行深入研究和测试，以提高 GPS 的精度。同时，我们将继续完善微信小程序的开发，确保用户界面和功能能够全面满足用户需求。在此期间，我们还将加强系统模块之间的耦合性，并推进更多的集成测试工作，以确保各个模块能够无缝对接。

进入第四周，我们计划完成 GPS 模块的优化工作，确保小车在动态情况下能够精准地进行路径跟随。同时，我们将持续进行系统集成和功能性测试，以确保所有模块的稳定运行。此外，我们将着重调整和优化硬件部分，特别是小车载件的布局，以确保整个系统的稳定性和可靠性。

第五周，我们将专注于进行全系统的综合测试，确保所有功能，包括路径跟随和用户交互，都能顺利运行。在此期间，我们还将准备验收展示所需的材料，包括演示文稿、

视频展示以及书面报告的初稿。

最终,在第六周,我们将完成最终的系统验收展示,这将包括现场演示和答辩环节。根据验收过程中得到的反馈,我们会进行必要的调整。此外,我们将最终完善报告和文档,以确保项目的完整性和呈现效果。

通过这一阶段性的工作,我们期望能够实现项目目标,同时确保系统的高性能和用户满意度。我们相信,通过团队的共同努力,最终一定能够成功地完成这一挑战,并交付一个高质量的项目成果。

### 3.2 小组各成员任务分解

成员	任务分解
闫康	专注于 GPS 定位优化算法,确保路径跟随精度,调整和优化现有算法,确保在不同环境下的稳定性。
孙常鑫	负责系统的整体测试,尤其是硬件与软件的集成测试,发现并解决潜在问题,确保系统在不同情况下的稳定性和可靠性。
苏珈磊	完善硬件部分的调试和优化,解决潜在的硬件问题,编写和整理项目文档,包括硬件搭建说明、系统架构设计和最终报告。
薛皓林	继续完善用户界面和功能,确保应用程序的稳定性和易用性,与其他模块进行接口调试,确保数据传输和控制命令正常运行。

## 4. 目前遇到的主要问题及后续工作难点

### 4.1 GPS 模块精度较低

在我们的项目中,我们最初使用的是 LC29H(DA) GPS 模块,但是在测试过程中,我们发现这个模块存在精度不足的问题。具体来说,该模块提供的坐标点实际上是半分钟前的位置信息,这种延迟对于需要实时精确定位的自动驾驶小车项目来说是不可接受的。无法实时获取小车的精确位置,高精度的路径跟随变得不可能实现,这对我们的项目构成了重大挑战。

面对这一挑战,我们的团队并没有选择被动等待,而是积极寻求解决方案以提高 GPS 的定位精度。首先,我们尝试更换了精度更高的 NEO-M8T 模块。然而,实际应用中我们发现,尽管理论上精度有所提高,但实际效果并不明显。更令人困扰的是,由于该模块的高精度特性,即使小车处于静止状态,获取到的 GPS 位点仍然在不断变化,这种变化虽然在一定范围内波动,但无疑增加了误差的可能性。

在更换模块的尝试未能取得预期效果之后,我们转而探索更优的算法解决方案。我们初

步考虑采用卡尔曼滤波算法对 GPS 模块收集到的位点数据进行处理，以期获得更加精确的路径点。卡尔曼滤波器是一种高效的递归滤波器，能够从一系列含有噪声的测量中估计动态系统的状态，这对于减少 GPS 数据的噪声和提高定位精度具有潜在的帮助。

此外，我们也在尝试采用实时运动测量（RTK）技术来进一步提升 GPS 的精度。RTK 技术通过使用一个固定的基准站来校正移动 GPS 接收器的误差，从而实现厘米级的定位精度。我们相信，通过引入 RTK 技术，能够显著提高小车的定位精度，从而满足项目的需求。

如果这些措施仍然不能满足我们对精度的要求，我们也会继续考虑其他可能的方案，包括使用差分 GPS 技术、引入 GLONASS 或其他卫星导航系统辅助定位等。差分 GPS 通过比较两个 GPS 接收器的信号来消除误差，而 GLONASS 系统则提供了更多的卫星资源，有助于在城市环境中获得更好的定位效果。