

## 基于 Jetson Nano 的智能无人车控制器设计

### 摘 要

本文主要介绍了智能无人车控制器设计及开发过程。以研究基于 CAN 总线的线控无人车底盘的控制器设计原理为目的，在 Jetson Nano 边缘计算平台开展无人车控制器控制软件设计。

智能车底盘采用松灵机器人公司 SCOUT MINI 四轮差速底盘，控制器采用英伟达 Jetson Nano a02 计算平台，以其作为整个系统信息处理和命令发出的操作核心。以 CAN 总线通讯作为通讯方式，附带 GPS、摄像头、CAN-USB、降压电源等模块以保持其工作状态以及智能化功能。在 Jetson Nano 内接无线网卡，通过 VNC 远程连接进行无线网络控制。以大量查阅文章文献等为基础研究线控无人车底盘控制器设计原理、及 CAN 总线工作方式。通过编写 Python 程序由 python-can 库进行代码到 CAN 报文的转换，由 Jetson Nano 将命令发送至智能车底盘，由底盘内部芯片进行数据接收从而控制电机转动。文章主要介绍设计方案选取和分析、系统硬件设计及软件设计，还介绍系统调试方法策略。经测试表明，可以通过 ROS (Robot Operating System) 进行电脑键盘来控制智能车移动以及速度控制，也可以通过 python 程序来控制智能车移动。信息采集后，可以通过 GPS 信号进行路径规划。

**关键词：**智能无人车；控制器；Jetson Nano；CAN 总线通讯

## **Design of intelligent unmanned vehicle controller based on Jetson Nano**

### **Abstract**

This paper focuses on the design and implementation of the intelligent vehicle controller. the goal is to research the controller design principle of the can bus-based in-line unmanned vehicle chassis, the unmanned vehicle controller control software design was carried out on the Jetson Nano edge computing platform.

The intelligent car chassis adopts the SCOUT MINI four-wheel differential chassis of Songling Robotics Company, and the controller adopts the NVIDIA Jetson Nano a02 computing platform, which is used as the operational core of the entire system information processing and command issuance. CAN bus communication is used as a communication method, and modules such as GPS, camera, CAN-USB, and buck power supply are provided to maintain their working state and intelligent functions. Connect the wireless card inside the Jetson Nano and control the wireless network via the VNC remote connection. Based on a large number of article literature, the design principle of the chassis controller of the unmanned vehicle by wire control and the working mode of the CAN bus are studied. By writing a Python program, the code is converted from the python-can library to the CAN message, and the command is sent to the smart car chassis by Jetson Nano, and the data is received by the chip inside the chassis to control the motor rotation. This paper focuses on the selection of the system, the design of the hardware and software of the system, and the debugging methods and strategies of the system. Tests have shown that the smart car movement and speed control can be controlled through the ROS (Robot Operating System) computer keyboard, and the smart car movement can also be controlled through the Python program. After the information is collected, path planning can be carried out by gps signaling.

**Key Words: Intelligent unmanned vehicle; Controller; Jetson Nano; CAN bus communication**

## 目 录

基于 Jetson Nano 的智能无人车控制器设计 .....	I
摘 要 .....	I
Abstract .....	II
第 1 章 概述 .....	1
1.1 选题背景 .....	1
1.2 发展现状 .....	1
1.3 设计概况 .....	3
第 2 章 方案选取与分析 .....	5
2.1 智能车底盘 .....	5
2.2 Jetson Nano 控制器 .....	6
2.3 CAN 总线 .....	8
2.3.1 CAN 总线简介 .....	8
2.3.2 CAN 总线技术 .....	8
2.4 附加模块 .....	10
2.4.1 电源模块 .....	10
2.4.2 CAN 模块 .....	11
2.4.3 GPS 模块 .....	12
2.5 无线网络通讯 .....	13
第 3 章 系统硬件选型 .....	14
3.1 总体设计 .....	14
3.2 智能车底盘框架搭建 .....	14
3.3 电源电路设计 .....	15
3.4 GPS 模块 .....	16
3.4.1 GPS 定位原理 .....	16
3.4.2 GPS 模块工作原理 .....	17
3.5 摄像头 .....	18
第四章 软件设计 .....	19
4.1 软件设计思想 .....	19
4.2 软件控制程序设计 .....	20
4.3 GPS 定位程序设计 .....	22
4.4 物体识别程序设计 .....	23
第 5 章 系统调试 .....	26

5.1 硬件调试.....	26
5.2 软件调试.....	27
5.2.1 基本环境配置 .....	27
5.2.2 ROS 系统 .....	28
5.2.3 ROS 仿真 .....	29
5.2.4 开源 SDK.....	30
5.3 软硬联调.....	31
5.4 成果展示.....	33
5.4.1 实物图.....	33
5.4.2 运动控制 .....	34
5.4.3 路径规划和无线网络的通信远程控制 .....	34
结 论 .....	36
参考文献 .....	37
附 录 .....	39
致 谢 .....	40

## 第1章 概述

### 1.1 选题背景

无人车 (Unmanned Vehicle) 可以依靠计算机系统和车载传感器感知周围环境，并根据传感器信息和自身状态自主规划并完成所分配的任务。由于搭载平台不同，通常也称为智能车辆或自主移动机器人。由于无人车自身具有强大的环境感知和实时规划能力，同时在执行任务过程中不完全依靠操作者，因此在很多领域有良好应用前景。实现无人车底盘运动控制功能的无人车控制器是无人车系统的一个关键部件。课题针对线控无人车底盘的控制器开展研究工作，设计一种基于 Jetson Nano 边缘计算平台的无人车控制器，并开展相关实验验证其有效性。

在现代科技的飞速发展下，汽车的电子化、信息化、网络化、智能化是汽车发展的必然方向。无人机正是这种发展的最好代表。无人机分为远程控制和自动控制两大类。目前常用的各种控制方法包括基于计算机的自动引导、基于 ZigBee 的无线网络、CAN 总线的有线控制等<sup>[1]</sup>。

智能汽车控制器是智能汽车的“神经中枢”，它对汽车的整体性能有着举足轻重的地位，它可以根据 PWM 的输出来实现对汽车的控制，从而实现汽车的自动驾驶；通过对智能汽车的车速进行模糊控制，并在智能汽车行驶时，通过对道路状况的分析，确定出相应的车速，实现车速的调整。

### 1.2 发展现状

国内外智能车控制器已得到飞速发展，早在2007年，“飞思卡尔杯”全国大学生智能车大赛举行，该比赛统一采用MC9S12DG128B型单片机作为智能车控制器，并获得广泛关注。

现如今智能车控制器也已更新换代，如特斯拉公司的产品Model X中控制器之一的MCU控制器，采用了TI公司的TMS320F26611P8K0芯片，为了使其拥有良好的运算能力和运行速度，还搭配了一块ACTE的LA3P125VQG100型号芯片配合其使用，以确保系统的稳定性和可靠性。再如飞思卡尔公司的FlexRay协议中采用MFR4200控制器，适用于智能车底盘的控制、动力调节等<sup>[2]</sup>。

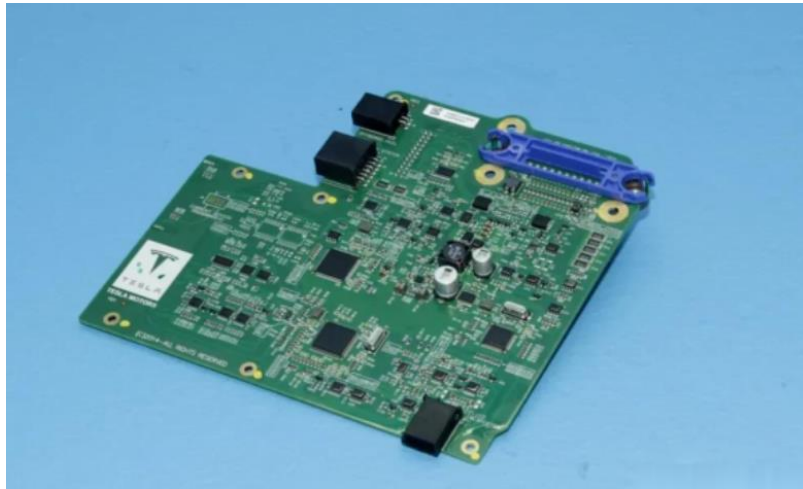


图1-1 特斯拉前驱控制器中的控制板

控制器有很多种类型，包含一系列的内核结构。常用的控制器如飞思卡尔公司的16核控制器，ARM7、ARM9微控制器，其中国内大学智能车比赛常用到STM32，因其在性能、消耗、片内外设上的优势。

表1-1 国内外智能车域控制器产品

公司名称	域控制器名称	备注
奥迪	zFAS	Audi A8
英伟达	DrivePX2	Tesla AV 2.0
Renesas	HADP/R-Car HAD Solution Kit	TTa Drive w/Renesas
博世	AI Car Computer	Nvidia-based
恩智浦	Blue-box	Full HW Stack
德尔福	CSLP Platform w/Mobileye	Prototype
ZF	ZF PRO AI (passenger/commercial)	Nvidia-based
英特尔	GO AV Platform(w/Mobileye vision)	Full HW Stack

我国在 80 年代后期才开展了无人驾驶车辆控制器的研发工作，尽管国内对智能型汽车控制器的研发起步较迟，在很多方面还落后于国外，但通过自己的努力，也已经有了一定的成绩。国内有不少著名的高校，都在从事这一领域的工作。

一九九三年，中国科学院终于研发出了真正意义上的智能型汽车。这款智能型汽车的主要控制系统由计算机、现代精密传感设备和传动系统构成，在此基础上，实现了原有的自动驾驶，并由主控系统来实现自动驾驶。

西安交通大学、北京科技大学、军事交通大学、武汉大学等对智能车控制器都有所研究；近几年，中科院合肥材料所等对汽车智能化控制系统进行了深入的研究，并已有一些成果。然而，在我国，对于智能汽车的研发还处在试验性的阶段，把智能汽车放在常规的车流量中，就象是人工行驶的汽车；遵守交通法规，熟悉周围的环境，特别是周围的车辆（车灯、喇叭等）的反应，并作出反应，最后在市区和市郊公路上安全、顺畅的行驶。这是因为，由于智能型汽车对周围车辆的驾驶行为和意图的识别和了解，缺少基于对车辆的正确反应和反应的判断，也就是说，智能型汽车缺少与周围车辆的互动和协作<sup>[12]</sup>。

由于汽车工业的发展，交通拥堵和环境污染逐渐成为城市交通系统中最严重的问题。因此，各国纷纷展开智能网联汽车（ICV）研究。ICV 使用传感器，控制器和其他设备与车辆的外部环境共享信息和数据。同时，他们可以作出明智的决策，并结合车辆状况输出共享数据。ICV 是具有部分或完全自动驾驶功能的车辆。

作为汽车架构中最重要的组件之一，车辆控制器单元（VCU）是分析驾驶员和车辆之间驾驶需求的处理中心。车辆控制器主要实现车辆行驶需求、CAN 网络管理、故障诊断和能源管理等功能。作为车辆的控制核心，车辆控制器负责通过 CAN 总线的信号采集能力监控车辆中的电池系统、电机系统等模块。外部电路中的数字、模拟和开关信号用于分析驾驶员的驾驶意图，以获得满足功率要求的输出转矩，从而满足车辆的内部控制要求。

### 1.3 设计概况

课题针对线控无人车底盘的控制器开展研究工作，设计一种基于 Jetson Nano 边缘计算平台的无人车控制器，查阅现有智能无人车系统文献，了解国内外线控无人车控制器发展趋势，研究基于 CAN 总线的线控无人车底盘的控制器设计原理，在实验室已有的线控无人车底盘上开展基于 Jetson Nano 边缘计算平台的无人车控制器

设计工作，并完成无人车控制器的调试工作；在 Jetson Nano 边缘计算平台开展无人车控制器控制软件设计，实现基于 GPS 信号的路径规划和无线网络的通信远程控制；在线控无人车底盘上开展控制器控制实验，测试基于 GPS 定位信息的无人车无线网络控制功能，并对实验结果进行分析。

由此可以提出四个研究问题：1. 智能车底盘的控制器如何控制车体移动；2. 什么是CAN总线，CAN总线如何通讯<sup>[13]</sup>；3. 如何通过Jetson Nano与无人车控制器相连接通讯；4. 关于智能车智能化问题，如何进行无线网络控制、GPS的路径规划。

由以上四个问题展开分析、文献翻阅以及实践等，逐步完成各项设计。



## 第2章 方案选取与分析

### 2.1 智能车底盘

智能车底盘采用松灵机器人公司的产品SCOUT MINI。SCOUT MINI智能移动底盘是一种四轮四驱差速底盘，和其他底盘相比更加轻便小巧。而且类比同类型底盘，SCOUT MINI延续了前一代 SCOUT的四轮驱动，独立悬挂，可以原地旋转，车轮马达的设计也是创新性的，它的最大转向半径是0米，爬坡角度几乎是30°。

SCOUT MINI拥有出色的越野能力，其尺寸较SCOUT减小了近半，并具有20公里/小时的速度和精确的操控能力。SCOUT MINI开发平台具有独立的控制中心，能够实现CAN总线的标准化通信，能够访问CAN总线通信，同时还可以通过各种外设来实现对ROS的二次开发，对更先进的CAN和更先进的机器人进行开发，这正符合本课题的要求。配置有标准航模航空器，采用24 V@15Ah的锂电池，行驶距离可达到10公里。为实现多种功能，立体相机、激光雷达、GPS、IMUS、机械手等可供安装到SCOUT MINI的扩展口上。SCOUT MINI可以广泛地用于无人巡检，越野，科研，大学生竞赛，运输等领域。

表2-1 性能参数

参数类型	项目	指标
机械参数	长 x 宽 x 高 (mm)	627 × 550 × 252
	轴距 (mm)	452
	前 / 后轮距 (mm)	450
	车体重量 (Kg)	20
	电池类型	锂电池 24V 15aH
	电机	直流无刷 4 X 150W
	驱动形式	四轮独立驱动
	悬架	摇臂独立悬架
	转向	四轮差速转向
	安全装备	伺服刹车/防撞管
性能参数指针	空载最高车速 (km/h)	≤ 20

	最小转弯半径	可原地转弯
	最大爬坡能力	$\geq 30^\circ$
	最小离地间隙 (mm)	107
控制参数	控制模式	遥控控制 控制指令模式
	遥控器	2.4G / 极限距离 1Km
	通讯接口	CAN

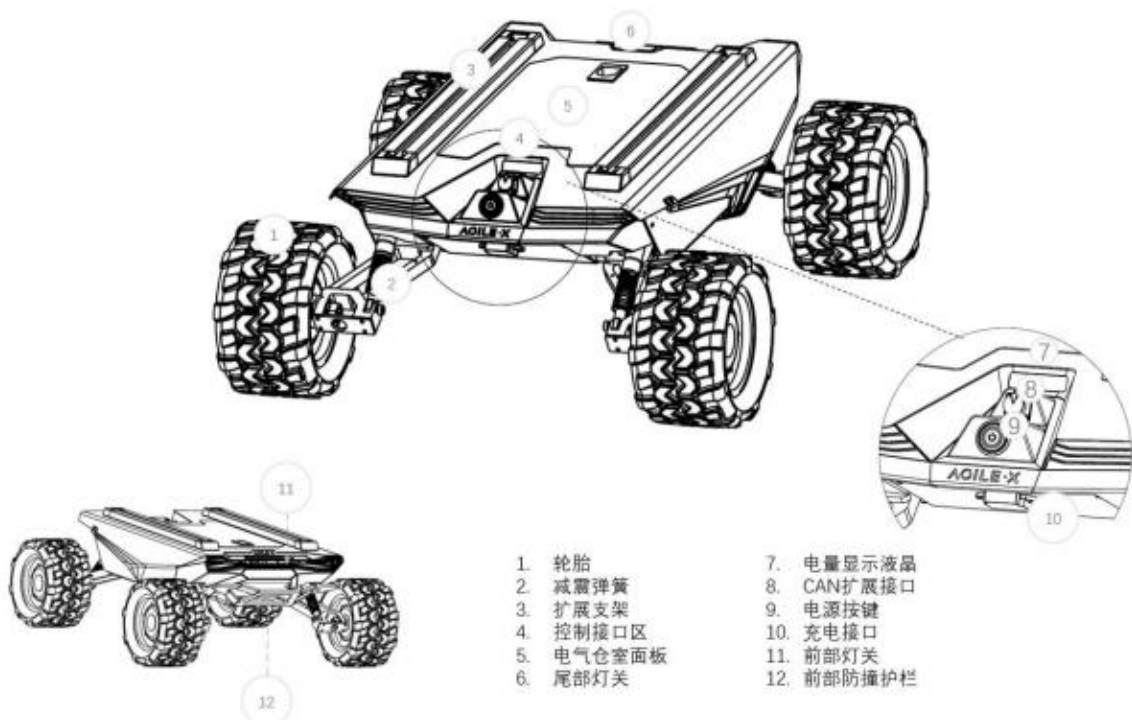


图2-1 底盘概览视图

## 2.2 Jetson Nano 控制器

本课题是在 Jetson Nano 边缘计算平台开展无人车控制器控制软件设计，实现基于 GPS 信号的路径规划和无线网络的通信远程控制；在线控无人车底盘上开展控制器控制实验，测试基于 GPS 定位信息的无人车无线网络控制功能。研究基于 Jetson Nano 的智能无人车控制器设计，Jetson Nano 体积小巧、算力强大，是 AI 嵌入式开发板中性价比较高的一种，由英伟达公司于 2019 年 3 月推出。该开发板

预装 18.04Ubuntu 桌面系统，GPU 采用英伟达公司 128 核的 Maxwell 芯片，能够以高算力快速实现 AI 技术，同时可搭配各种智能设备。

Jetson Nano 相当于一个计算机，给智能车的舵机、电机等发送指令，从而实现智能车的前进后退等动作，并且可以运行完整的操作系统，如 Linux，这意味着我们可以使用自己熟练的语言，如 Python、Java 等和熟悉的库来对物体识别软件进行开发。Jetson Nano 运算能力强大，这同样方便了程序的编写和调试。Jetson Nano 自带的接口比较全面，让接线方式变得灵活可变，可以满足较复杂的控制要求。Jetson Nano 的体积小，可以恰当地搭载到小车上<sup>[14]</sup>。大体的硬件布局，Jetson Nano 除了之前提到的核心板分离式设计（J2），还包括了一个 M.2 接口，可以用来外接无线网卡。除此之外，Jetson Nano 有与树莓派兼容的外设接口（J41）；风扇接口（J15）；摄像头接口（J13）；以及 USB 和 HDMI。另外 J40 是按键接口，类似 PC 主板上的接口。Jetson Nano 推出了 2GB 版本，内存从 4GB 降为 2GB，其他参数不变，但价格降为 4GB 版本的一半。本文基于 Jetson Nano 4GB 版本，但因为 2GB 版本使用完全相同的核心和系统，所以内容同样适用于 2GB 版本<sup>[3]</sup>。由于我也使用过树莓派 4b，对比其与 Jetson Nano，二者都是人工智能嵌入式开发板，而 Jetson Nano 则具有更高的计算能力，处理器、GPU 都更强大，同时因为搭配了 Jetpack 开发包而更加容易上手。

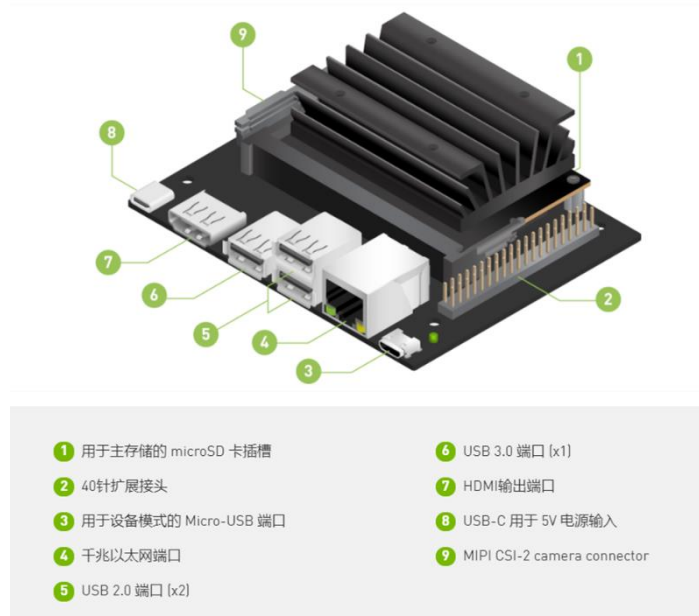


图2-2 Jetson Nano

## 2.3 CAN 总线

### 2.3.1 CAN 总线简介

CAN的全名是“Controller Area Network”，也就是对本地局域网的控制。CAN总线是当今世界上使用最多的一种现场总线。CAN具有多种很好的特性，可以让用户更容易地进行选择<sup>[4]</sup>。

它的特点是：①费用低廉②总线的使用量非常大③数据传送范围长（10 KM）④数据传送速度快（1Mbit/s）⑤可以按照报文ID确定消息的接收或掩码⑥正确的差错处理和误差检测反馈功能⑦在被损坏后可以重新传送⑧当发生重大差错时，该结点可以自动离开总线⑨消息不包含源地址或目的地址报文不含源地址或目标地址，仅用标志符指示功能信息、优先级信息<sup>[11]</sup>。

CAN是全球应用最广泛的现场总线之一，最初由德国波什公司推出。它是一种广泛应用于车辆内部测量和执行单元之间交换数据的串行数据通信总线。由于采用了非破坏性仲裁技术，CAN总线具有更高的可靠性。CAN总线直接通信最大范围为10km，最高通信速度为1MB/s<sup>[13]</sup>。

随着现代异构中电子控制器和仪表数量的增加，车辆可靠性很大程度上受控制系统中部署电路复杂性的影响。此外，维护工作难以进行。<sup>[15]</sup>从布局上看，传统的系统采用点对点的通信方式，这种单一的通信方式必然会导致大型布线问题。因此，高品质的车辆使用CAN控制器局域网、总线系统将系统中的所有控制装置连接起来，实现统一管理。这导致不同控制系统之间的轻松数据共享和互操作性<sup>[16]</sup>。

CAN总线控制网络已应用于车辆、航空航天和军事设备领域。由于CAN总线传输数据涉及车辆控制系统的实时采集，以及车辆、部件和集成车辆的运行状态、设置参数和运行指令等因素，因此，对车载CAN性能测试方法的研究显然是必要的<sup>[17]</sup>。

目前的测试设备采用总线接口芯片（模块）的设计。这种测试设备获得的数据框架是不完整的，因为缺乏时间信息。因此，无法实现多总线之间数据帧传输的相互关系。因此，设计另一种能够独立于总线接口芯片（模块）传输数据帧的测试方案是必要的。该方案涉及通过以下方式测量总线帧间空间的技术指标，直接采集物理层上的数据传输波形、解码软件等。

### 2.3.2 CAN 总线技术

目前，CAN总线测试采用各种特殊协议芯片，解码后只能得到完整的数据帧，不

能覆盖总线中运行的实际整个传输数据（包括不正确的帧）<sup>[18]</sup>。不正确的数据帧包括不正确的位，并验证哪些是由总线传输波形数据中的失真和干扰引起的，并通过特殊的总线协议芯片进行过滤。实际上，这些过滤后的数据对于测试数据传输网络性能和故障位置至关重要<sup>[19]</sup>。目前，故障定位无法依靠那些异常数据来找出，因为测试设备过滤了那些异常数据帧。

由于通过相应的总线专用协议芯片获得的总线数据帧信息没有统一的时间参考，因此从相应的总线专用协议芯片收集到的只是顺序而准确的时间信息<sup>[20]</sup>。因此，总线命令帧和响应帧之间的帧间空间的测量无法实现。特别是在对多个总线数据帧进行综合分析时，由于缺乏相关的时间信息，在判断系统的工作性能时无法将其考虑在内。

目前，车辆总线测试是通过操作特殊的接口协议芯片来完成的，这些芯片是耦合到总线的类型。但是，要通过软件分析数据帧，应通过转换的物理层数据波来获取传输帧数据。直接从集体波数据中获取数据帧的工作，除非解决一些关键理论，如数据帧定位的初始时间，测量脉冲宽度，自动传输码率计算以及基于CAN协议分析数据帧解码等<sup>[21]</sup>。

软件解码总线数据帧的关键技术是，首先计算从高速数据中采集的波数据的脉冲宽度；其次，获取车辆总线传输数据的波特率；第三，从采集到的波形中获取帧的起点，第四，根据耦合总线帧格式，通过平移波解码得到对应的数据，最后，通过软件显示波形和平移日期，以供后期测试。以及以下工作，在计算波特率时处理有效因素，如随机脉冲波的干扰，立即将集体波数据转换为CAN数据帧，解码集成数据帧（包括误差帧），以便为以后分析误差帧提供可靠的帧数据并准确获取误差帧数量，最终为总线性能评估和故障定位提供依据<sup>[22]</sup>。

操作软件要转码时，应计算从高速数据采集的波数据的脉冲宽度。它包括这些步骤，获取总线传输数据的波特率，从集合波形中获取帧的起点，根据测试的总线数据协议对代码进行贪婪编码，并对波形代码进行转码，得到相应的结果<sup>[23]</sup>。

在总线数据帧脉冲宽度和传输码率的自动计算中，采用平均最小宽度法。该总线的传输速率可以通过统计计算帧脉冲宽度来计算。根据相应的总线协议，在设置总线数据帧的开始时间时采用双阈值方法，以便根据启动脉冲特征和信息进行全面的判断。根据响应总线协议，软件对数据帧的解码是基于将数据帧的格式从脉冲波形转换为数据位字符0、1位，最后解释帧日期<sup>[24]</sup>。解码过程完成后，对数据帧进行校准并准

确存储数据帧的时间戳，开始时间。因此，精确时间戳可以是微秒级。数据帧传输波形和软件解码的数据帧结果可以同时显示在查看屏幕上<sup>[5]</sup>。

CAN总线通讯依靠发送CAN报文来实现，由发送单元开始，传送数据的帧，到接收单元，采用十六进制报文形式，报文的种类分为数据帧、远程帧、错误帧、过载帧、帧间隔等。常用Motorola大端报文形式，即低位数据存放在高位地址。CAN报文的组成部分有序号、传输方向、第几路CAN、时间标识、帧ID、帧格式、帧类型、数据长度、数据等。十六进制ID由29位标识符转换而来，帧ID有以下组成：P，优先级，3位，可以有8个优先级（0-7）；R，保留位，1位，固定为0；DP，数据页，1位，固定为0；PF，报文代码，8位；PS，报文目标地址（报文接收方），8位；SA，源地址<sup>[26]</sup>。

根据需要的ID找到相应的数据帧				需要解析的部分				
序号	传输方向	第几路CAN	时间标识	帧ID	帧格式	帧类型	数据长度	数据
0x00000000	接收	0	0x000380ce	181056F4	数据帧	扩展帧	0x05	ce181a0e01
0x00000001	接收	0	0x000381d7	1812F456	数据帧	扩展帧	0x06	2a18a00f0600
0x00000002	接收	0	0x00038300	181056F4	数据帧	扩展帧	0x05	ce181a0e01
0x00000003	接收	0	0x000383da	1812F456	数据帧	扩展帧	0x06	2a18a00f0600
0x00000004	接收	0	0x00038532	181056F4	数据帧	扩展帧	0x05	ce181a0e01

图2-3 CAN报文的组成

## 2.4 附加模块

由于需要智能车的各类功能，如基本移动、基于GPS信号的路径规划和无线网络的通信远程控制等，需要在Jetson Nano和底盘上附加各种需要的模块。

### 2.4.1 电源模块

智能车实现移动时，需要为Jetson Nano供电。为了其简便性，选择使用智能车底盘所带的航空扩展接口上的电源组进行供电。由于智能车底盘SCOUT MINI所配置的航空扩展接口所带的一组电源电压为23V-29.2V，与Jetson Nano所需要的5V电压不匹配，所以需要有一个降压电源模块。SCOUT MINI电气外部扩展电源电流不超过5A，总功率不超过120W；当系统检测到电池电压低于安全电压以后，外部电源扩展会被

主动切换，所以如果外部扩展设备涉及到重要数据的存储且无掉电保护。

降压模块选择Daygreen公司B30-1224-05型24V转5V直流降压器，可以有效将底盘输出的23V-29.2V DC降压为5V输出至Jetson Nano。



图2-4 Daygreen降压模块

#### 2.4.2 CAN 模块

研究基于 CAN 总线的线控无人车底盘控制器的设计，由于 Jetson Nano 无论是 a02 版本还是 b01 版本，均没有 CAN 通讯的引脚，所以需要外加 CAN 模块。但是由《DIA Jetson Linux 开发人指南》得知“A CAN controller is not available in Jetson Nano devices, but you can use an MCP251x chip(SPI to CAN interface), which works with the SPI interface on Jetson Nano.”所以需要—个 MCP25x 系列 CAN 模块。初始选定微雪公司的 MCP2515 芯片，因为其基于 Raspberry Pi 40 引脚 GPIO 接口，适用于树莓派系列主板，因为 Jetson Nano 与 Raspberry Pi 的 40 pin 引脚大体相同，所以其也适用于 Jetson Nano。而且支持 2 路 CAN 接口通信，可以通过 CAN 收发器与车体上的 CANH 和 CANL 相连，从而实现基于 CAN 总线的无人车底盘控制。

通过外部 CAN 总线接口可以控制底盘的移动的线速度以及旋转的角速度。协议包含系统状态回馈帧、运动控制回馈帧、控制帧。控制帧包含了模式控制、故障清除指令、线速度控制开度、角速度控制开度以及检验和。控制器将指令符传送给 CAN 收发信机，并通过接收指令字来启动内线中断，并利用接收指令字进行对应的操纵和转速控制，从而达到操纵汽车的目的<sup>[28]</sup>。本系统可按要求将汽车的状况经由 CAN 总线传输至 CAN 远程控制及数据收集 PC，用于数据的存储与加工<sup>[29]</sup>。



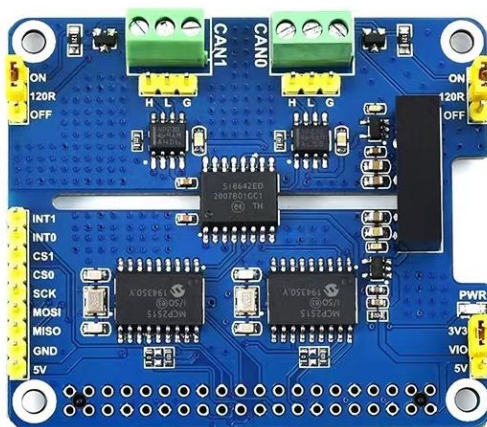


图2-5 微雪 CAN总线模块

后来收到智能车底盘松灵机器人公司邮寄的CAN-USB模块，由于Jetson Nano上带有USB接口，可以直接使用CAN-USB模块，只需要将CAN-USB和底盘接头的CAN\_H和CAN\_L分别相接即可实现CAN通讯。



图2-6 USB-CAN模块

### 2. 4. 3 GPS 模块

为实现智能车的基于GPS信号的路径规划功能，需要在Jetson Nano上附加GPS模块。GPS模块选择微雪公司树莓派GNSS扩展板，同理由于Jetson Nano的40pin引脚与树莓派大体一致，所以同样适用于Jetson Nano。选择其原因是其基于MAX MBQ，支持



北斗、GPS、GLONASS等GNSS，具有精准、快速、防干扰等优点，可以轻松实现全球定位，适合课题中所要实现的基于GPS信号的路径规划功能<sup>[27]</sup>。

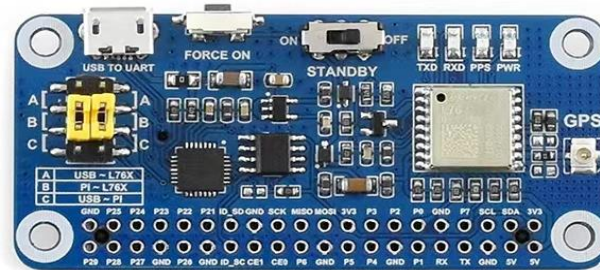


图2-7 GNSS扩展板

## 2.5 无线网络通讯

由于在智能车进行移动时，Jetson Nano由于连接USB-CAN，需要搭载在底盘上随智能车底盘一同移动，所以需要软件对Jetson Nano进行无线网络远程通讯。远程的图形管理方式让实验变得简单方便，因为我们无法在调试时仍然进行有线通讯，智能车一旦移动，连接线就有可能断开，所以我们必须使用无线网络通讯。

VNC (Virtual Network Computing)，是一种利用RFB技术实现屏幕共享与遥控的软件，所以选择VNC作为无线网络通讯的方案<sup>[25]</sup>。

首先在Jetson Nano上安装VNC软件vino-server，再进行文件改写和重新编译，Jetson Nano上VNC便配置完毕，随后在控制端设备安装VNC viewer，获得并填入Jetson Nano的IP地址，创建主机连接，便可以远程控制了。

同时由于Jetson Nano需要外接独立的键盘和鼠标进行控制，每次开机启动VNC时都需要把键盘鼠标也接入，十分麻烦。所以直接在Jetson Nano的开机启动项中加入启动VNC服务的指令，就可以使Jetson Nano开机便开启VNC服务，在上位机上进行无线网络通讯。

但是VNC并不是完全同步的，上位机与Jetson Nano之间有短暂的延迟，可以通过降低分辨率来降低延迟。

## 第3章 系统硬件选型

### 3.1 总体设计

系统硬件总体设计围绕底盘和Jetson Nano，将底盘上扩展通讯接口的VCC和GND与Daygreen降压模块相连，再将降压模块的output和GND和Jetson Nano的5V和GND连接。再将扩展通讯接口上的CAN\_H和CAN\_L与USB-CAN模块上的CAN\_H和CAN\_L分别相连，再将USB口插入Jetson Nano上的USB接口。

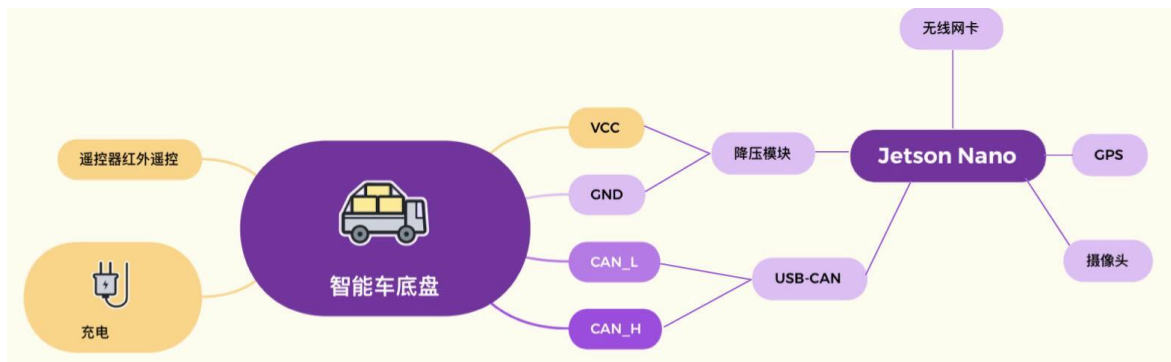


图3-1 系统硬件总体设计

对于其他硬件GPS、无线网卡、摄像头等均安装到Jetson Nano上，GPS安装到40pin引脚上，无线网卡安装到Jetson Nano模组下方的M.2 Key E接口，摄像头安装到Jetson Nano的CSI接口。

### 3.2 智能车底盘框架搭建

松灵机器人SCOUT MINI智能车自带扩展支架，搭建铝合金框架，固定框架后，在平台上把Jetson Nano系列硬件固定。选好硬件后，简单搭建了一个简单的平台，就可以开始软硬件调试和后续工作。通过简单的连接就可以进行基本的实验，通过不同的外设，还可以进行功能扩展。



图3-2 底盘框架

### 3.3 电源电路设计

Jetson Nano有三种供电方式：

1. USB供电。使用数据线，连接Jetson nano的MicroUSB接口进行供电，拔掉J48跳线帽。
2. 使用DC供电，官方认证的DC电源是5V4A(4000mA) switching power supply，台达5V6A的DC电源同样好使。使用时要在J48插上跳线帽。
3. 使用引脚供电。

在此我们使用引脚供电，将底盘上扩展通讯接口的VCC和GND与Daygreen降压模块相连，再将降压模块的output和GND和Jetson Nano的5V和GND连接。



图3-3 电源模块实拍

### 3.4 GPS 模块

#### 3.4.1 GPS 定位原理

GPS通信系统由3个部分组成：其中包含空间部分、地面支撑系统和用户设备部分。空间部分即工作卫星和备用卫星，地面支撑系统为监测站、注入站、主控站等。

[30]

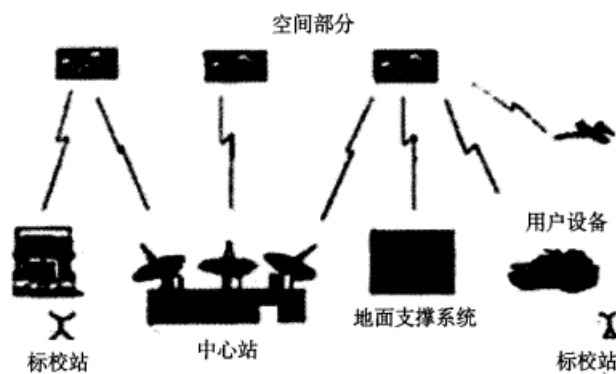


图3-4 卫星定位通信组成

GPS的基本定位原理是：以高速移动的卫星的瞬时位置为初始数据，由卫星不断传送其自身的时间信息和星历值，使用者在收到该信息后，再进行时空相交；求出了三维位置、三维方向和速度和时间信息<sup>[31]</sup>。

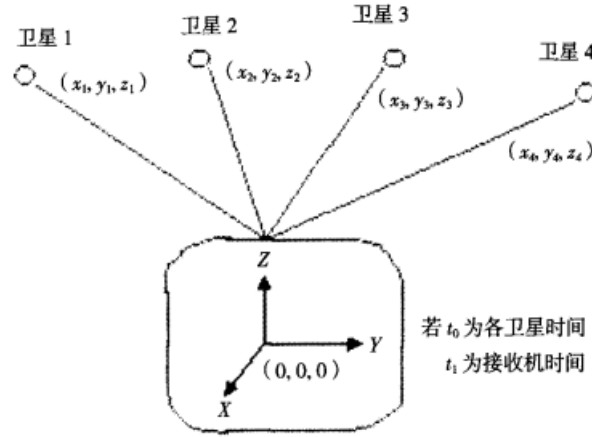


图3-5 四颗卫星观测

由图可得以下四个方程式<sup>[36]</sup>:

$$(x_1 - x)^2 + (y_1 - y)^2 + (z_1 - z)^2 + c^2 \cdot (t - t_{01})^2 = d_1^2 \quad (3-1)$$

$$(x_2 - x)^2 + (y_2 - y)^2 + (z_2 - z)^2 + c^2 \cdot (t - t_{02})^2 = d_2^2 \quad (3-2)$$

$$(x_3 - x)^2 + (y_3 - y)^2 + (z_3 - z)^2 + c^2 \cdot (t - t_{03})^2 = d_3^2 \quad (3-3)$$

$$(x_4 - x)^2 + (y_4 - y)^2 + (z_4 - z)^2 + c^2 \cdot (t - t_{04})^2 = d_4^2 \quad (3-4)$$

上述4个方程式中待测点坐标 $x$ 、 $y$ 、 $z$ 和 $t$ 为未知参数<sup>[36]</sup>， $c$ 为GPS信号的传播速度（即光速）。其中： $d_1 \sim d_4$ 也是卫星1~4到定位点的距离，其值可通过信号从卫星到达定位点的时间乘以电波速度得到，即 $d_i = c \cdot t_a$ （ $i=1, 2, 3, 4$ ）<sup>[32]</sup>； $x$ 、 $y$ 、 $z$ 是每个卫星的星历参数，即卫星的轨道坐标， $t$ 是接收机的时钟差； $t_{01} \sim t_{04}$ 是各个卫星的时钟差解此方程即可得到需定位点的坐标，从而得到了定位<sup>[6]</sup>。

### 3.4.2 GPS 模块工作原理

GPS模块就相当于GPS通信系统三个独立部分中的用户设备部分，全球定位系统的人造卫星不断地向全球定位系统（GNSS模块）提供自己的时间和地点的无线电波。GPS的电脑和导航资讯产生器可以准确地知道其运行的轨迹和时间<sup>[35]</sup>，而Global网络也会不断追踪这些卫星的运行轨迹和系统的时序，在科罗拉多州的施里弗航空基地，与其运行部分共同为每个GPS卫星提供一次轨道定位和钟修正，这些资料是通过精密的数学计算得到的<sup>[33]</sup>。GPS接收机（GNSS）的立体定位需要接收4个以上的卫星发射的信号，三维位置根据空间三角勾股定理和四元一次方程计算得到<sup>[34]</sup>。

### 3.5 摄像头

为了使智能车更加具有智能化的特点，在Jetson Nano上安装了摄像头，使其能够实现物体识别的功能。摄像头采用Pi Camera V2 树莓派无夜视摄像机，其参数如下：

感光芯片：索尼 IMX219

像素：800W像素

CCD：1/4英寸

光圈（F）：2.0

焦距：3.04MM

视场角：73°

尺寸：25×23.86×9（mm）

图片分辨率：3280×2464（静态）

传感器：Sony IMX219PQ CMOS传感器

支持手动调焦

支持1080p30, 720p60 and 640x480p60/90摄像



图3-6 Pi Camera V2 树莓派无夜视摄像机

如图所示，摄像头安装在Jetson Nano内置的CSI接口上，图中Jetson Nano安装了金属外壳，外壳上带有摄像头支架。因为其视场角大，所以随Jetson Nano一同放在底盘的电气仓室面板上，镜头朝向智能车底盘前进方向。

## 第四章 软件设计

### 4.1 软件设计思想

在智能车辆的控制系统中，其软件系统主要可以包括：道路辨识、转向控制、后车轮驱动电机、检测车速偏移的计算方法。主控芯片的控制功能框图如图 4-1 所示：

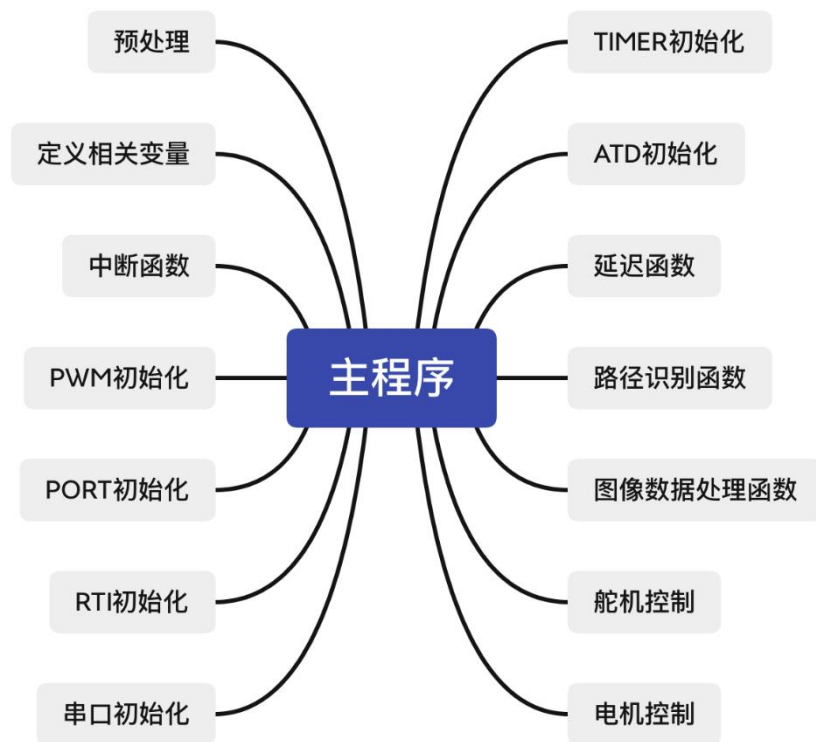


图4-1 主控芯片的控制功能框图

智能汽车的控制算法部分是整个研究内容的核心部分，控制算法能够确保智能汽车尽可能快速地完成路径跟踪的任务，而不会脱离指定路径。然而，在保证尽可能快的完成路径追踪和智能汽车速度提升相互制约这两个问题上，如何协调二者之间的直接关系，却是研究的难点所在<sup>[7]</sup>。

前视跟踪和 PID 控制是智能车辆的最主要的控制手段。尽管在科学和计算机模拟中，预瞄跟踪技术的应用取得了显著的成效，但是针对人一车一程的闭环模型的发展仍取得了较大进展。然而，现实中，由于车辆行驶速度极高，而且车辆行驶路

线变化迅速，加之各种原因，导致车辆在行驶过程中会产生偏差。所以，单纯依靠前瞄准跟踪的方法是不适合于汽车智能化的。另外，常规PID控制器具有结构简单、控制性能好、运行稳定等优点，因此在实际生产中得到了广泛的使用；但是，由于智能车辆技术的不断发展，使得在高速状态下，采用常规PID技术无法实现对目标运动的精确追踪。

20年代末，波德鲁布尼提出了分数级控制器，并提出在所设计的控制系统中，所采用的控制方法是以积分为单位的。21年代初，杨全辰先生就采用FlatPhase（FlatPhase）技术，对该系统进行了深入的研究。在此基础上，吴振宇等以小数阶PID控制与前视跟踪技术为代表的智能车辆进行了系统的仿真研究，并获得了较好的结果。在智能汽车中采用分数阶PID控制器，必将改善汽车智能汽车的控制性能，保证汽车智能化系统的稳定运行<sup>[8]</sup>。

## 4.2 软件控制程序设计

CAN总线通讯需要CAN报文以帧为单位传送，虽然我们可以直接通过Jetson Nano使用“cansend”指令发送帧ID，但由于SCOUT MINI底盘的超时保护机制，在收到一帧通讯协议以后，若超过500ms智能车底盘未收到下一条帧控制指令，底盘则会进入通讯保护状态，速度为零，所以CAN报文的发送必须是周期性发布。由于普通的“cansend”无法发送周期性指令，需要时钟函数进行定时发布帧控制指令。

由于松灵机器人官方已经有二次开发示例，其中有python示例，我们可以通过python-can库进行python语言和CAN报文的转换，以下是python-can的使用：首先加载os模块，调用shell命令，加载python-can模块，设置can接口名字，并且设定波特率1M，设置can的启动函数和can停止函数，尝试发送信息和接收信息

根据SCOUT\_MINI\_UserManual得知底盘的CAN接口协议，其中重要的就是运动控制指令控制帧。



指令名称	控制指令			
发送节点	接收节点	ID	周期 (ms)	接收超时(ms)
决策控制单元	底盘节点	0x130	20ms	500ms
数据长度	0x08			
位置	功能	数据类型	说明	
byte [0]	控制模式	unsigned int8	0x00 遥控模式 0x01 CAN 指令控制模式 <sup>(1)</sup> 0x02 串口控制模式	
byte [1]	故障清除指令	unsigned int8	详见备注 2*	
byte [2]	线速度百分比	signed int8	最大速度 5.0m/s,值域为 (-100, 100)	
byte [3]	角速度百分比	signed int8	最大速度 0.5235rad/s,值域为 (-100, 100)	
byte [4]	保留	-	0x00	
byte [5]	保留	-	0x00	
byte [6]	计数校验 (count)	unsigned int8	0-255 循环计数, 每发送一条指令计数加一次	
byte [7]	校验位(checksum)	unsigned int8	校验位	

图4-2 运动控制指令控制帧

根据CAN接口协议可得知,若要智能车底盘以0.5/s的速度前进,则可以向0x130发送01000a0000000044;若要智能车以0.07853rad/s旋转,则可以向0x130发送0100000a000000044。数据校验位为每一帧 CAN 消息的数据段最后一个有效字节,其校验和的计算方法 $\text{checksum} = (\text{ID\_H} + \text{ID\_L} + \text{data\_length} + \text{can\_msg.data}[0] + \text{can\_msg.data}[1] + \text{can\_msg.data}[2] + \text{can\_msg.data}[3] + \text{can\_msg.data}[4] + \dots + \text{can\_msg.data}[n]) \& 0\text{xFF}$ 。

ID\_H 与 ID\_L 为 ID 分别是帧 ID 的高八位和低八位。比如 ID 为 0x540,那么对应的ID\_H 为 0x05, ID\_L 为 0x40;

Data\_length 为数据长度为一帧 CAN 消息中数据段有效数据长度,包含校验和这个字节; can\_msg.data[n]为有效数据段中具体每个字节的具体内容,计数校验位是需要参与校验和计算的,校验和本身不参与计算。

```
/**
 * @brief CAN message checksum example code
 * @param[in] id : can id
 * @param[in] *data : can message data struct pointer
 * @param[in] len : can message data length
 * @return the checksum result
 */
static uint8 Agilex_CANMsgChecksum(uint16 id, uint8 *data, uint8 len)
{
    uint8 checksum = 0x00;
    checksum = (uint8)(id & 0x00ff) + (uint8)(id >> 8) + len;
    for(uint8 i = 0 ; i < (len-1); i++)
    {
        checksum += data[i];
    }
    return checksum;
}
```

图4-3 CAN信息校验算法

### 4.3 GPS 定位程序设计

首先进行GPS环境的配置：

1. 把GNSS模块安装到Jetson Nano上后，安装Python数据库：
2. 修改gpsd参数
4. 查看端口获取数据
5. 进入相应python目录，执行例程

```
waveshare-zsh@wavesharezsh-desktop:~/Documents/MAX-XXX_GNSS_HAT_Code/f
enter$ sudo python3 main.py
gps device make wgs84 coordinate
gcj02 coordinate is for amap or google map
bd09 coordinate is for baidu map
Please press Ctrl+c if want to exit
Module dont ready,please move the antenna to outdoors
Module dont ready,please move the antenna to outdoors
altitude      = 52.2 M
wgs84 lon,lat = 114.0787845 , 22.541397167
google lon,lat = 22.538707501,114.083912986
amap lon,lat  = 114.083912986,22.538707501
bd09 lon,lat  = 114.090416556,22.544596777
speed         = 0.03 KM/H
Update after 4 secondss
```

图4-4 执行例程

环境配置好后，利用GPS系统，获得智能车当前所在位置的经纬度和即将运行路径点的经纬度，但是经纬度坐标系不是一个恰当使用的坐标系方式，我们可以把其转化成平面坐标系。

首先参数初始化，之后假定智能车即将沿一个四边形移动，输入四边形的四个顶点x, y坐标，得到四边形的最大平行四边形顶点的x, y坐标，坐标点数组建立，从上往下，从左往右；随后获得智能车底盘当前的x, y坐标，找出距离智能车最近的顶点，让机器人运动到该顶点，设定智能车按照列方向走，并记录智能车依次要走的数数据点的x, y坐标。而后就是算法实现的过程。

#### 4.4 物体识别程序设计

在本试验中，汽车的检测方法是单一的，仅对一个具体的物体进行探测，而这一试验以瓶(瓶)为研究对象。选择了YOLOV4-TINY作为目标的探测方法，并根据YOLOV4-TINY. Weights进行加权。YOLO是一种以全卷积神经网络为基础的对象探测方法。YOLO算法的关键在于将整个图形作为网络的一个输入，仅通过一个神经网络即可获得边缘块的所在区域及其分类，满足了对图像进行实时探测的需要。使用YOLOV4-Tiny的Python界面，当摄像机侦测到目标时，会传回“detection”类，该类包含目标的分

类等级、目标的可探测置信范围、几何框 Box（Box具体地包括左上、左下和右上）；在左下方四个点的定位信息），Jetson Nano在获得这些数据后，在正确的情况下，把 Box输入到下一层以进行靶向位移探测。

在摄像机的视线的正中，将该点设定为（X0,Y0）。首先，基于YOLOV4-TINY所提供的标识对象方块，获得所述标识对象的中央（XT、YT），然后将所述对象的中央位置与所述摄像机的所述视场的正中央作差异，从而获得所述对象的所述垂直和横向的所述相对位移的所述YT-Y0、XT-X0。当所述相对偏差为正时，所述物体在所述微型车辆的左边，所述相对移动的所述物体处于所述微型汽车的右边，相对偏移量的绝对值越大表示目标距离小型车的偏移量越大<sup>[9]</sup>。

基于对象的探测，通过对物体的相对偏差进行分析，从而实现了车辆的轨迹追踪。PID控制器在汽车轨迹运动中得到广泛应用。

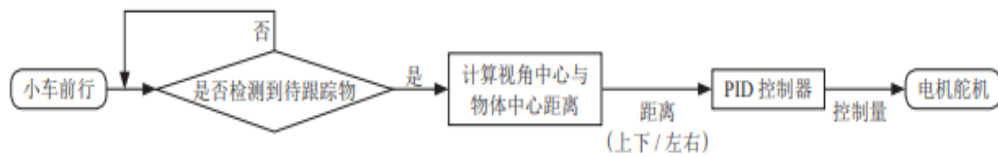


图4-5 智能车跟踪逻辑

通过计算跟踪目标的中心坐标并找到最接近图像中心的目标，我们尝试使检测到的目标最接近视野中心，然后引导智能无人车移动到目标。为了将目标保持在视野的中心，我们将自动驾驶汽车设置为以固定角度进行调整。但机器人无法准确跟踪目标，且旋转范围过大，很容易导致目标在自动驾驶车的视野中丢失。机器人来回摆动，无法平稳地移动到目标。所以我们引入PID算法，让机器人更流畅地跟踪物体。PID算法通过比例、积分和导数，我们使用PID算法来计算控制的控制量<sup>[39]</sup>。

$$C=K_p \times er + K_d \times er_{rate} + K_i \times \int_0^T er * dt \quad (4-1)$$

PID为比例（P），积分（I）和微分（D）的控制。但并非一定要三个都有，或者PD，PI，或者P算法。比例、积分、微分的控制方法均有各自的功能：比例，反应体系的基础（目前）偏离  $e(t)$ ，其因子较大，能加速调整，减少错误；但是，如果比率太高，则会降低系统的稳定性，从而导致不稳定；通过对系统进行积分、反馈，可以有效地排除稳定的误差，改善系统的错误程度；由于存在错误，调整的过程直到没有错误为止；差分法，它是一种能够预测偏离的改变的速率  $e(t) - e(t-1)$ ，它可

以预测出偏差的改变的倾向，从而起到提前的控制效果；在产生误差以前，通过差动调整的方法得到了去除，从而提高了系统的动力学特性。然而，差分法具有对噪声的放大效应，而增强差分法则会对系统的抗干扰性产生负面影响。

加载PID算法后，自动驾驶汽车,可以稳定地跟踪目标。

## 第5章 系统调试

### 5.1 硬件调试

智能车底盘SCOUT MINI自带富斯遥控器作为SCOUT MINI产品选配配件，使用遥控器可以轻松控制SCOUT MINI通用机器人底盘，拨杆SWB切换控制模式，SWC为手动灯光控制开关，SWD控制速度模式，左摇杆控制前进后退，右遥控控制车子左旋转和右旋转。值得注意的是，在内部控制上移动底盘是根据百分比映射的，因此当摇杆处于同一个位置时，其速度是恒定的。正常情况下，SCOUT MINI 若可以通过要遥控器控制正常情况下，说明底盘运动控制正常，可以接受到底盘的反馈帧，说明 CAN 扩展链路正常。检查发送的 CAN 控制帧，看数据校验是否正确，控制模式中是否置为指令控制模式，可以通过底盘反馈的状态帧中错误位中校验错误标志的状态情况。

在遥控器控制模式下，遥控器左摇杆往前推动则为往X正方向运动，遥控器左摇杆往后推动则往X负方向运动，遥控器左摇杆推动至最大值时，往X方向运动速度最大，遥控器左摇杆推动至最小值时，往X方向负方向运动速度最大；遥控器右摇杆左右控制车体的旋转运动，遥控器右摇杆往左推动车体则由X轴正反向往Y正方向旋转，遥控器右摇杆往右推动车体则由X轴正方向往Y负方向旋转，遥控器右摇杆往左推动至最大值时，逆时针方向旋转线速度最大，遥控器右摇杆往右推动至最大值时，顺时针旋转线运动速度最大。

在控制指令模式下，线速度的正值表示往X轴正方向运动，线速度的负值表示往X轴负方向运动；角速度的正值表示车体由X轴正方向往Y轴正方向运动，角速度的负值表示车体由X轴正方向往Y轴负方向运动。

按下SCOUT MINI电源按键，等待数秒；将SWB拨至中间，选址要控制控制挡位；可尝试手动切换灯光模式，确定模式选择时候正确；尝试将左边摇杆轻轻往前推，推一小部分即可，可见小车缓慢速度往前移动；尝试将左边摇杆轻轻往后推，推一小部分即可，可见小车缓慢速度往后移动；释放左边摇杆，小车停下；尝试将右边摇杆轻轻往左推，推一小部分即可，可见小车缓慢往左旋转；尝试将右边摇杆轻轻往右推，推一小部分即可，可见小车缓慢往右旋转；释放右边摇杆，小车停下；在相对空旷的区域自由控制，多次尝试而后熟悉车辆移动速度。

阅读了松灵机器人产品 SCOUT MINI 产品手册，了解智能车底盘基本接口和控

制说明，大致了解 CAN 接口的开发功能，继续扩大检索范围。查询了 CAN 收发器及 CANH 和 CANL 引脚的用法，如果运用 Jetson Nano a02 则使用 CAN 收发器来进行通讯，如果是 Jetson Nano b01 则直接有 CAN\_TX 和 CAN\_RX 接口。同时查询了相关 GPS 模块，但是精度太低，决定采用差分 GPS。继续阅读了松灵机器人产品 SCOUT MINI 产品手册及在 CSDN 上检索相关文章。尝试理解 CAN 通讯协议，打算使用 USB 转 CAN 接口连接电脑通过 CAN 调试助手发送指令。从 Jetson Nano a02 改用 b01，做基本配置，软件安装。查询后发现 b01 没有需要的 CAN\_TX 和 CAN\_RX 接口，通过检索开发人指南得出 CAN controller is not available in Jetson Nano devices，需要一个 MCP251x 芯片。经查询得到 Seeed 的 MCP2517 和 MCP2518 以及微雪的 MCP2515 拓展板可以实现 CAN 通讯。对比了引脚以及电压等最后选定微雪模块。同时查询 GPS 模块，引脚是否冲突、电压是否合适等。

## 5.2 软件调试

### 5.2.1 基本环境配置

通过学习英伟达 Nano 教学视频，学习 JetsonNano 的相关教程，对 JetsonNano 进行实际操作：在电脑上准备 JetsonNano 开发者套件，烧录镜像至 microSD 卡，安装开发者套件并尝试交互，首次启动 Nano 查看并接受 NVIDIAJetson 软件 EULA。在 CSDN 上找到关于 JetsonNano 视觉跟随小车以及 Jetbot 智能车的设计和开发环境测试的教程。同时安装 Ubuntu 等相关软件，为 Nano 程序和代码的编写做准备。在计算机上通过 putty 软件使 Jetson Nano 开机，但是效果并不理想，连接几次只有一次可以成功显示。于是我用一块 7 寸显示器和 HDMI 线连接 Jetson Nano，成功开机运行 Ubuntu18.04 桌面系统。



图 5-1 显示器和 HDMI 线连接 Jetson Nano

开机之后，由于英伟达 Jetson Nano 默认配置的环境源为国外的源，下载或更新时速度慢，可以把源换成清华或者中科大的国内源，但是要先备份原本的源。我们这里选择清华的源进行更新。

### 5.2.2 ROS 系统

打开终端，首先输入命令进行系统更新。

而后基本设置如关屏时间、汉化语言、拼音输入等，方便自己平时的使用，配置基本环境。安装一些常用软件如 Code OSS、pycharm 等。在本课题中，首先尝试安装 ROS 系统。

由于我的 Jetson Nano 预装的是 Ubuntu18.04 系统，对应的是 ROS 的 Melodic 版本，之后配置 ROS 的环境，安装 rosinstall 和其他依赖，对 ROS 进行测试：首



先打开 ROS 系统，运行 roscore，重新打开一个新的终端，输入以下内容：

```
roslaunch turtlesim turtlesim_node
```

再在终端重新打开一个新的终端标签并输入：`$ roslaunch turtlesim turtlesim_teleop_key`，然后就可以看到并能控制一只小海龟。

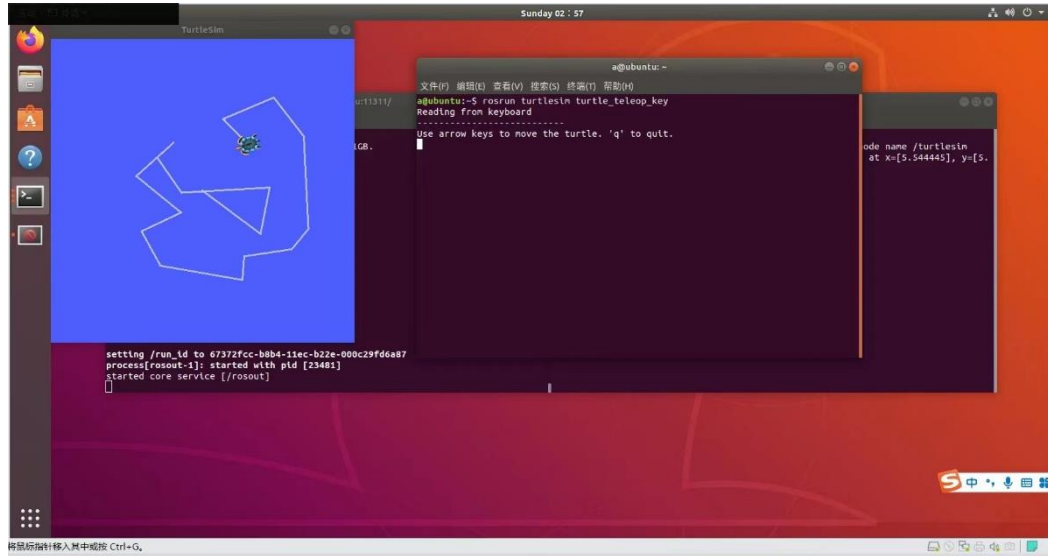


图 5-2 ROS 测试

### 5.2.3 ROS 仿真

ROS 测试成功之后调试 SCOUT MINI 的仿真，官方提供的仿真环境是 ROS-melodic+gazebo9，首先新建工作空间文件夹，并使用 catkin 初始化工作空间：在 .devcontainer 中新建 devcontainer.json 和 Dockerfile 两个文件，之后对 Dockerfile 文件进行编辑。

随后进行仿真环境配置，安装 ROS 库，然后初始化工作空间并安装依赖，即可正常编译与运行。

随后添加自定义传感，但是官方的仿真模型中并未添加现成传感器，这也很好理解——车辆上本来就没有，无故添加反而会在实际使用中让人疑惑。

scout 系列车型的 urdf 描述文件位于 scout\_description/urdf 中，在这里我们并未直接使用 urdf 文件进行描述，而是从 xacro（xml macro）描述文件中生成 urdf 信息。使用 xacro 的好处在于我们可以像编程一样实现复用一些 urdf 节点以及将不同的组件拆分等功能。

在对一切环境进行配置之后运行一下 Rviz 查看我们修改后的模型，可以看到

车辆头顶出了一个红色小方块，这就是我们的简易摄像头模型

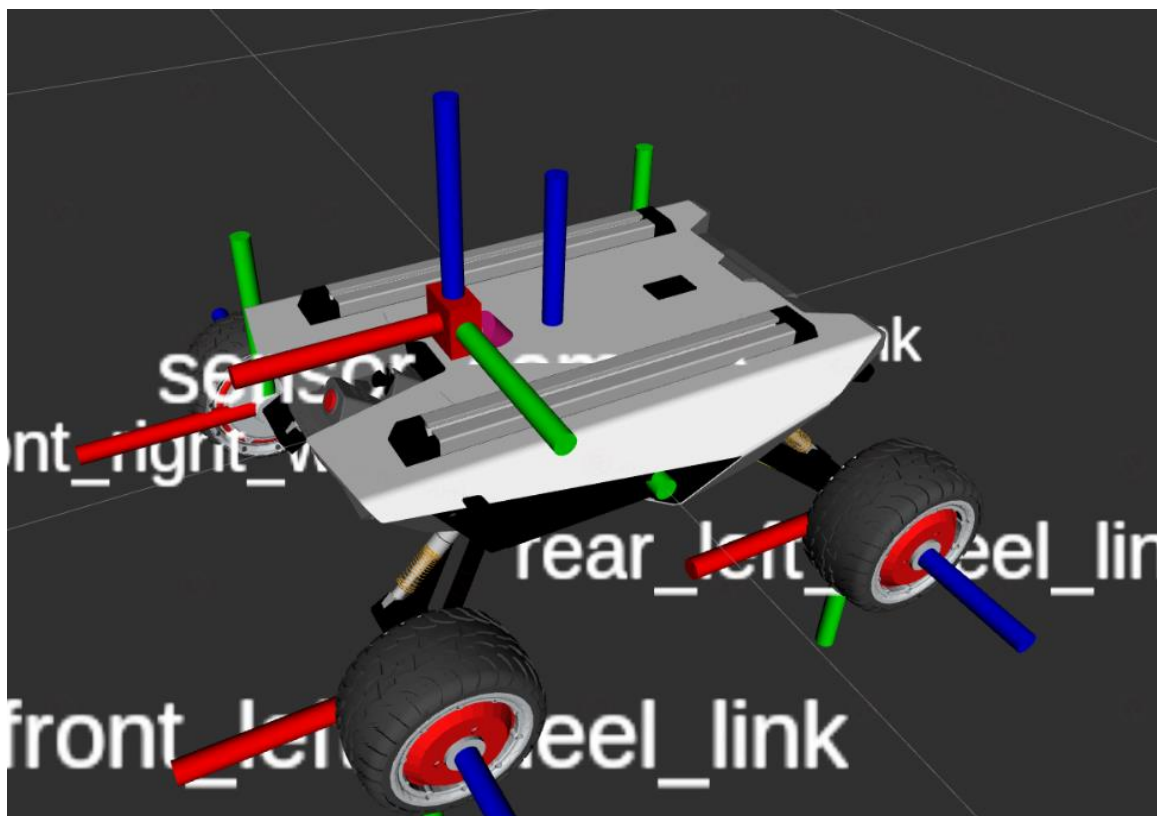
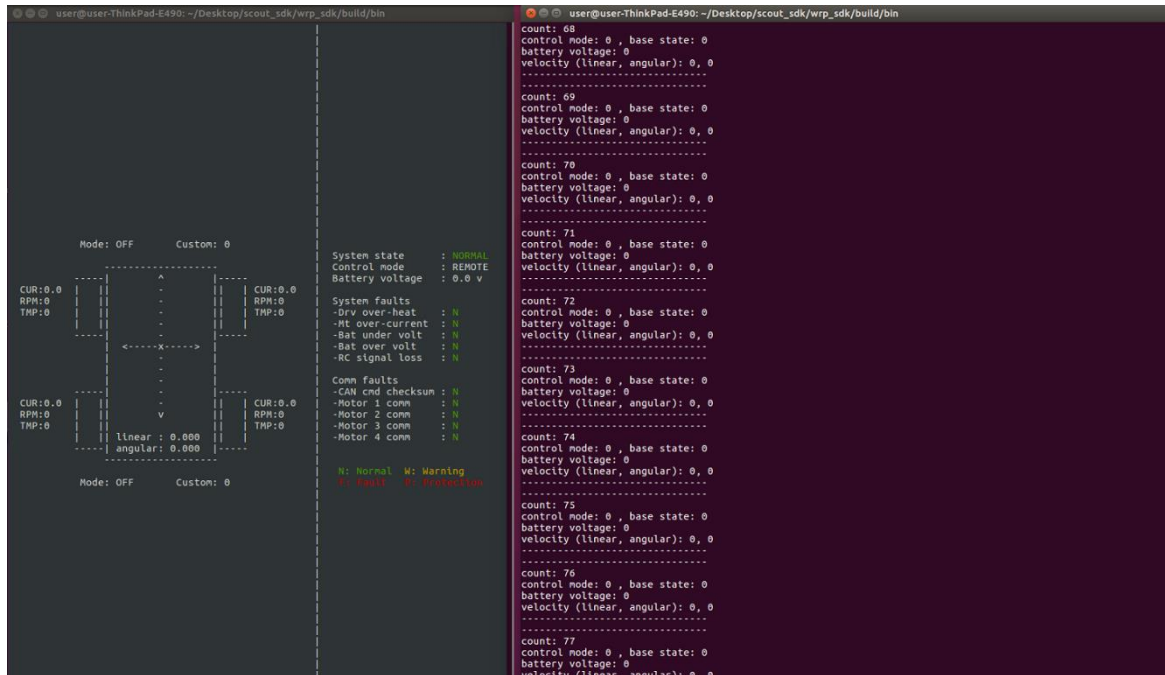


图 5-3 Rviz 仿真模型

也可以在 gazebo 中仿真并用 Rviz 接收图像。

#### 5.2.4 开源 SDK

而后通过开源 SDK 控制 SCOUT MINI 的底盘，首先安装编译工具，构建 TUI 监控工具，需安装 libncurses 配置和构建，设置一个用于 CAN 总线模式的参数，之后运行结果如下：



```
user@user-ThinkPad-E490: ~/Desktop/scout_sdk/wrp_sdk/build/bin
Mode: OFF Custom: 0
CUR:0.0 RPM:0 TMP:0
CUR:0.0 RPM:0 TMP:0
linear: 0.000 angular: 0.000
Mode: OFF Custom: 0

System state : NORMAL
Control mode : REMOTE
Battery voltage : 0.0 V

System faults
-Drv over-heat : N
-Mt over-current : N
-Bat under volt : N
-Bat over volt : N
-RC signal loss : N

Comm faults
-CAN cmd checksum : N
-Motor 1 comm : N
-Motor 2 comm : N
-Motor 3 comm : N
-Motor 4 comm : N

N: Normal W: Warning
F: Fault E: Protection

count: 68
control mode: 0, base state: 0
battery voltage: 0
velocity (linear, angular): 0, 0
count: 69
control mode: 0, base state: 0
battery voltage: 0
velocity (linear, angular): 0, 0
count: 70
control mode: 0, base state: 0
battery voltage: 0
velocity (linear, angular): 0, 0
count: 71
control mode: 0, base state: 0
battery voltage: 0
velocity (linear, angular): 0, 0
count: 72
control mode: 0, base state: 0
battery voltage: 0
velocity (linear, angular): 0, 0
count: 73
control mode: 0, base state: 0
battery voltage: 0
velocity (linear, angular): 0, 0
count: 74
control mode: 0, base state: 0
battery voltage: 0
velocity (linear, angular): 0, 0
count: 75
control mode: 0, base state: 0
battery voltage: 0
velocity (linear, angular): 0, 0
count: 76
control mode: 0, base state: 0
battery voltage: 0
velocity (linear, angular): 0, 0
count: 77
control mode: 0, base state: 0
battery voltage: 0
velocity (linear, angular): 0, 0
```

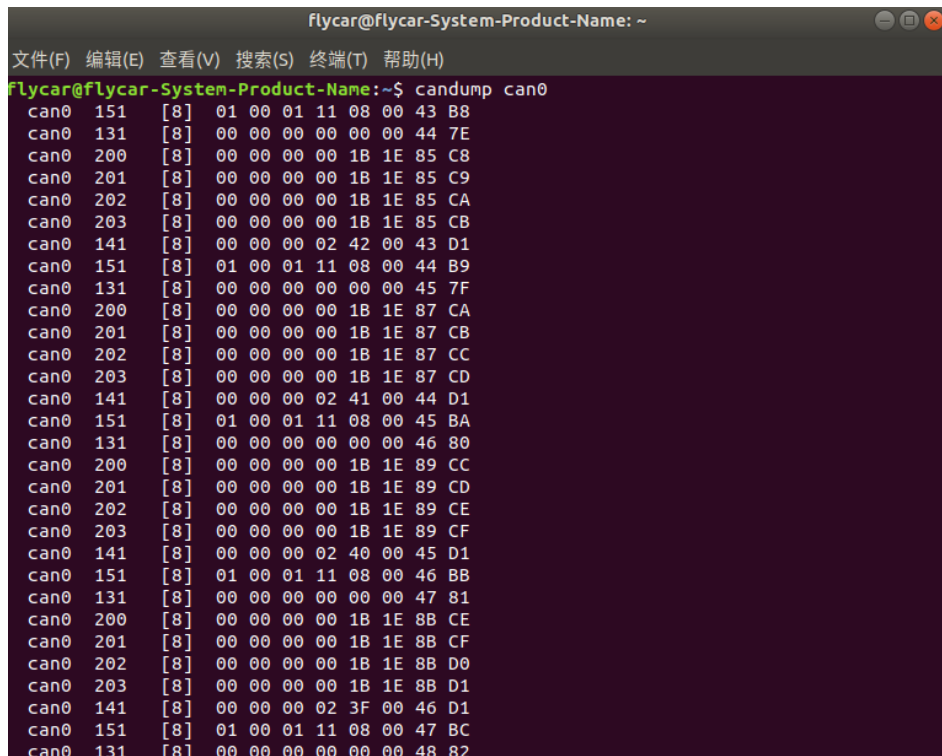
图 5-3 开源 SDK 控制底盘

通过开源 SDK，可以快速通过指令的形式与小车底盘进行控制，省去自主适配协议的过程，做到拿来即用的状态，省去开发的过程，缩短开发时间。

### 5.3 软硬联调

在分别配置好硬件和软件之后，再进行软硬件联合的调试，Jetson Nano接入USB-CAN，再把USB-CAN的CAN\_H和CAN\_L与底盘航空头的CAN\_H和CAN\_L分别相接，开启底盘电源，Jetson Nano开机。

首先设置CAN转USB适配器，启用gs\_usb内核模块；设置can设备参数，设置波特率；使用指令查看can设备；查看CAN设备后应发现“CAN0”得知CAN成功连接上；安装和使用can-utils来测试硬件之后测试指令\$ candump can0



```
flycar@flycar-System-Product-Name: ~  
文件(F) 编辑(E) 查看(V) 搜索(S) 终端(T) 帮助(H)  
flycar@flycar-System-Product-Name:~$ candump can0  
can0 151 [8] 01 00 01 11 08 00 43 B8  
can0 131 [8] 00 00 00 00 00 00 44 7E  
can0 200 [8] 00 00 00 00 1B 1E 85 C8  
can0 201 [8] 00 00 00 00 1B 1E 85 C9  
can0 202 [8] 00 00 00 00 1B 1E 85 CA  
can0 203 [8] 00 00 00 00 1B 1E 85 CB  
can0 141 [8] 00 00 00 02 42 00 43 D1  
can0 151 [8] 01 00 01 11 08 00 44 B9  
can0 131 [8] 00 00 00 00 00 00 45 7F  
can0 200 [8] 00 00 00 00 1B 1E 87 CA  
can0 201 [8] 00 00 00 00 1B 1E 87 CB  
can0 202 [8] 00 00 00 00 1B 1E 87 CC  
can0 203 [8] 00 00 00 00 1B 1E 87 CD  
can0 141 [8] 00 00 00 02 41 00 44 D1  
can0 151 [8] 01 00 01 11 08 00 45 BA  
can0 131 [8] 00 00 00 00 00 00 46 80  
can0 200 [8] 00 00 00 00 1B 1E 89 CC  
can0 201 [8] 00 00 00 00 1B 1E 89 CD  
can0 202 [8] 00 00 00 00 1B 1E 89 CE  
can0 203 [8] 00 00 00 00 1B 1E 89 CF  
can0 141 [8] 00 00 00 02 40 00 45 D1  
can0 151 [8] 01 00 01 11 08 00 46 BB  
can0 131 [8] 00 00 00 00 00 00 47 81  
can0 200 [8] 00 00 00 00 1B 1E 8B CE  
can0 201 [8] 00 00 00 00 1B 1E 8B CF  
can0 202 [8] 00 00 00 00 1B 1E 8B D0  
can0 203 [8] 00 00 00 00 1B 1E 8B D1  
can0 141 [8] 00 00 00 02 3F 00 46 D1  
can0 151 [8] 01 00 01 11 08 00 47 BC  
can0 131 [8] 00 00 00 00 00 00 48 82
```

图5-4 查看CAN0接收的数据

显示如图5-3说明底盘正不停通过CAN0向Jetson Nano发送回馈帧ID。之后测试运动控制指令控制帧，可以通过指令开启指令模式：

cansend can0 130#0100000000000003a: 指令模式；

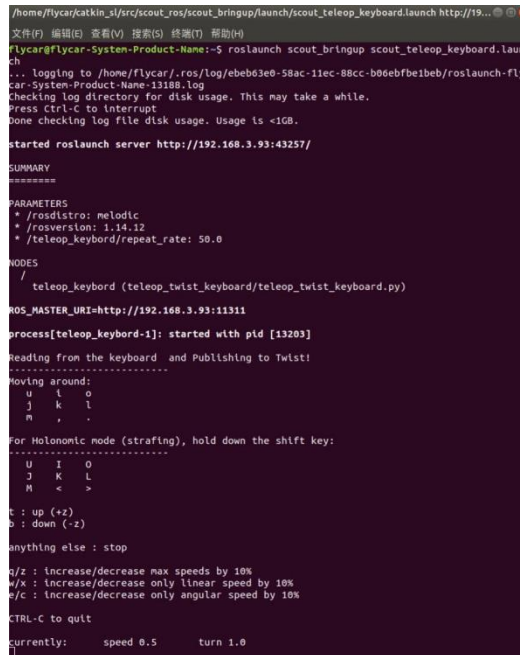
cansend can0 130#01000a0000000044: 看到智能车0.5/s 前进，但由于SCOUT MINI底盘的超时保护机制，在收到一帧通讯协议以后，若超过500ms智能车底盘未收到下一条帧控制指令，底盘则会进入通讯保护状态，速度为零，所以CAN报文的发送必须是周期性发布。

cansend can0 130#0100000a00000044: 智能车旋转。

之后进行ROS packages依赖的安装，将scout\_ros package 下载至catkin 工作空间，并进行编译。

在catkin\_ws下打开终端编译ROS工作空间下的所有功能包。

开启键盘控制：



```
/home/flycar/catkin_ws/src/scout_ros/scout_bringup/launch/scout_teleop_keyboard.launch http://192.168.3.93:11311
flycar@flycar-System-Product-Name:~$ roslaunch scout_bringup scout_teleop_keyboard.launch
... logging to /home/flycar/.ros/log/eb63e0-58ac-11ec-88cc-b06ebf61ebf/roslaunch-flycar-System-Product-Name-13188.log
checking log directory for disk usage. This may take a while.
press ctrl-c to interrupt
done checking log file disk usage. Usage is <1GB.

started roslaunch server http://192.168.3.93:43257/

SUMMARY
*****
PARAMETERS
 * /roscpp: melodic
 * /rosversion: 1.14.12
 * /teleop_keyboard/repeat_rate: 50.0

NODES
 /
   teleop_keyboard (teleop_twist_keyboard/teleop_twist_keyboard.py)

ROS_MASTER_URI=http://192.168.3.93:11311

process[teleop_keyboard-1]: started with pid [13203]

Reading from the keyboard and Publishing to Twist!
-----
Moving around:
  u    i    o
  j    k    l
  m    ,    .

For Holonomic mode (strafing), hold down the shift key:
-----
  U    I    O
  J    K    L
  M    <    >

t : up (+z)
b : down (-z)

anything else : stop

q/z : increase/decrease max speeds by 10%
w/x : increase/decrease only linear speed by 10%
e/c : increase/decrease only angular speed by 10%

CTRL-C to quit

currently:  speed 0.5      turn 1.0
```

图5-5 ROS下的键盘控制

开启键盘控制后，就可以通过外接键盘或远程VNC所在的电脑键盘进行智能车控制，I键前进，<键后退，J、L键进行左右控制，U、O、M、>键可以进行左前、右前、左后、右后方运动，T键加速，B键减速，由此即可进行无线网络通信的远程控制。

## 5.4 成果展示

### 5.4.1 实物图

基于Jetson Nano的智能无人车控制器实物图如下图所示：



图5-6 实物图

#### 5.4.2 运动控制

使用遥控器可以轻松控制SCOUT MINI通用机器人底盘，拨杆SWB切换控制模式，SWC为手动灯关控制开关，SWD控制速度模式，左摇杆控制前进后退，右遥控控制车子左旋转和右旋转。在内部控制上移动底盘是根据百分比映射的，因此当摇杆处于同一个位置时，其速度是恒定的。

使用ROS系统，配置ROS环境后开启键盘控制，就可以通过外接键盘或远程VNC所在的电脑键盘进行智能车控制，I键前进，<键后退，J、L键进行左右控制，U、O、M、>键可以进行左前、右前、左后、右后方运动，T键加速，B键减速，由此即可进行无线网络通信的远程控制。

#### 5.4.3 路径规划和无线网络的通信远程控制

在车辆自动驾驶中，利用坐标比对技术进行车辆自动驾驶。在主机设置好目标后，主机自动产生导航路线。<sup>[37]</sup>在此路线上，每隔10米或碰到拐角时，都会有一个坐标，然后由GPS定位模组的讯息读出，以求出目前的坐标，也就是目前的坐标。将目前的位置与所获取的坐标进行对比，以求车辆行驶的方位<sup>[38]</sup>，并通过与目标坐标之间的对比来判断目前与目标之间的距离，直至目前和结束座标之间的差异处于设置区域之内，此时可以判定为到达目的地，停止移动，结束导航<sup>[10]</sup>。

首先各模块初始化，设定目的地，生成导航路径，读取GPS数据，而后设置循环：智能车位置和导航路径中当前点位置的距离是否在设定范围内，如果不在，则智能车继续沿当前方向前进，继续查看位置和导航路径中当前点位置的距离是否在设定范围内，直到前进到设定范围内，切换到下一个导航路径点。之后比较智能车位置和终点坐标位置的距离是否在设定范围内，如果不是则继续沿当前方向前进，继续比较位置和导航路径中当前点位置的距离是否在设定范围内，到下一个路径点直到智能车位置和终点坐标位置的距离在设定范围内，则视为已到达目的地，结束导航。

在试验中，利用GPS定位和导航技术对车辆进行定位，并将其实时数据传送给主机。在给出了目的地后，通过计算机程序对路线进行优化，把路线的信息传递到Jetson Nano中，从而达到自主导航，并且误差在10m以内<sup>[40]</sup>。

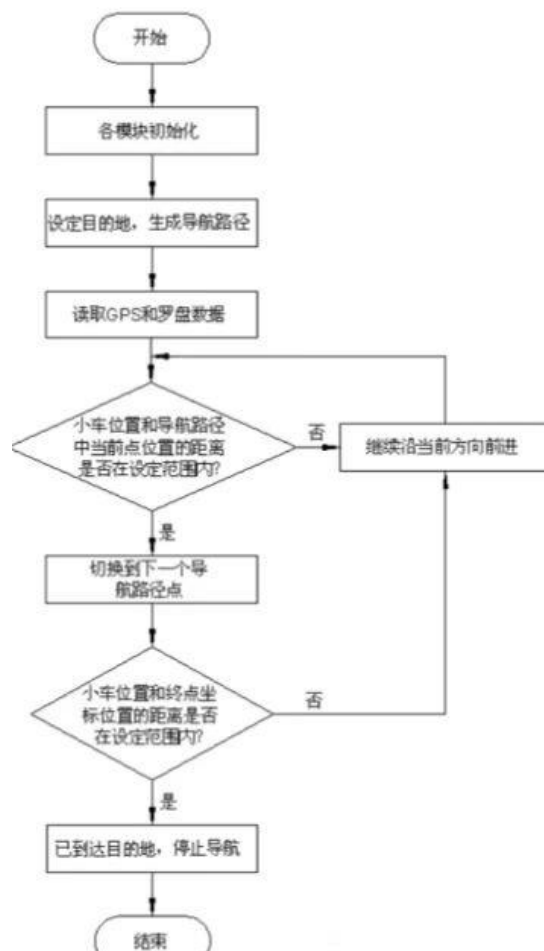


图 5-7 控制程序流程图

## 结 论

本文详细介绍了基于 Jetson Nano 的智能无人车控制器设计方案，该方案中，硬件智能车底盘采用松灵机器人 SCOUT MINI 四轮差速底盘，Jetson Nano 作为核心控制器，通讯方式采用 CAN 总线通讯，通过 VNC 远程控制和 ROS 系统下的键盘控制实现无线网络下的远程控制，在定位系统中采用坐标比较实现智能车的路径规划和自主导航。

文中介绍了方案选取分析和软硬件的设计，各个主要模块的工作原理和设计思路，并给出了开发工具、软件、指令以及各种不同的测试方式。硬件方面主要是智能车底盘、各模块的设计，软件部分主要分为基础运动控制程序和 GPS 定位程序等。

本设计具有前瞻性和总结性。Jetson nano，作为 2019 年上市的计算机，可以提供 AI 功能、强大的算力、完整的软件可编程性。并且在 2020、2021 各国的智能车比赛中，Jetson Nano 也得到了广泛地使用，在英伟达官方网站有完备的 Jetson Nano 教程，借助 Jetson 开发者套件以及为开发者、学生和教育工作者免费提供的在线培训，边实践边学习 AI，使其技术可行性和经济可行性得到极大的提高。其所用到的 CAN 总线通讯方式也具有极高的总线利用率、传输距离长、速率快等优点。

总结整个设计过程，控制器的制作，对内嵌软件及联机软件的使用要求，要求能自行设计并设计马达驱动器电路，控制微控制系统，软件编程等，涉及到广泛的专业知识，对我的学习生涯提供了很好的实践和科研机会。



## 参考文献

- [1] 邱钊鹏. 无人驾驶车辆控制方式研究[D]. 北京:北京工业大学,2009.
- [2] 飞思卡尔半导体(中国)有限公司. 面向未来汽车应用的通信控制器飞思卡尔MFR4200加快FlexRay先进汽车网络发展进程[J]. 半导体技术,2005,30(10):73-75. DOI:10.3969/j.issn.1003-353X.2005.10.020.
- [3] 黄坤. 基于Jetson Nano人型机器人辅助行走天轨实时跟踪系统的实现[D]. 重庆:重庆邮电大学,2021.
- [4] 刘溢,罗惠琼. CAN总线综述[J]. 福建电脑,2006(4):26-27,138. DOI:10.3969/j.issn.1673-2782.2006.04.016.
- [5] HUIMING WU, LING LI, HAO LI. Technical Research On CAN Bus Software Decode And Performance Index Measurement Of Inter-Frame Space[C]. //2017年第2届联合国际信息技术、机械与电气工程国际会议(JIMEC2017)论文集. 2017:101-104.
- [6] 易鸣镝,顾洪夫,陈广飞. GPS定位原理浅析及误差分析[J]. 中国数据通信,2005,7(3):25-27. DOI:10.3969/j.issn.1673-4866.2005.03.006.
- [7] 李欣然. 基于MC9S12DG128的自寻迹智能车控制器设计[D]. 黑龙江:哈尔滨理工大学,2012. DOI:10.7666/d.Y2280449.
- [8] 薛猛. 基于分数阶PID控制器的智能车鲁棒性校正方法研究[D]. 吉林:长春理工大学,2013.
- [9] 赖润平,周鹏程,张梓赫,等. 基于Jetson Nano的目标跟踪小车的设计与实现[J]. 现代信息科技,2021,5(4):181-183,187. DOI:10.19850/j.cnki.2096-4706.2021.04.046.
- [10] 许金钢,常计东,刘思聪,等. 基于GPS定位的自动引导小车设计[J]. 电子测试,2016(19):7-8. DOI:10.3969/j.issn.1000-8519.2016.19.004.
- [11] 付晓.CANBUS总线在组合开关及其监控系统中的应用[J].电气开关,2012,50(05):67-68.
- [12] 张乃龙,崔建国.无线遥控车辆仿真平台的开发[J].机械设计与制造,2014(09):168-170+174.DOI:10.19356/j.cnki.1001-3997.2014.09.050.
- [13] 罗平,邓云海,邓希来,延伟.基于CAN总线的专用车辆遥控系统控制方案设计[J].时代汽车,2018(11):100-102.
- [14] 黄坤. 基于Jetson Nano人型机器人辅助行走天轨实时跟踪系统的实现[D].重庆邮电大学,2021.DOI:10.27675/d.cnki.gcydx.2021.000574.
- [15] 宋长权,蒋小冬.基于CAN总线的频率测试系统设计与实现[J].福建电脑,2011,27(10):121-122.
- [16] 毛开凯,朱晓锦,康宛琪.车载远程动态跟踪系统数据采集与传输[J].仪表技术,2013(06):7-9.DOI:10.19432/j.cnki.issn1006-2394.2013.06.003.
- [17] 戴国强,李金广.基于LPC2109的冷藏车CAN总线温度采集系统的设计[J].电子产品世界,2012,19(07):48-50.
- [18] 张迎春,朱林涛,马鹏飞,黄帅.CAN总线在太阳能电池测试分选机中的应用[J].电子工业专用设备,2012,41(09):14-17.
- [19] 李瑞歌.CAN通信在汽车电控单元中的应用[J].科技创新导报,2011(08):61.DOI:10.16660/j.cnki.1674-098x.2011.08.067.
- [20] 郝志廷.基于CAN总线技术在车载网络中的应用[J].安徽电子信息职业技术学院学报,2011,10(06):22-24.
- [21] 滕文,封平安.极薄煤层机载电牵引采煤机电控系统的智能模块化设计探讨[J].煤矿机

- 电,2012(05):70-71+74.DOI:10.16545/j.cnki.cmet.2012.05.030.
- [22] 杜吉龙.浅谈CAN总线的应用[J].半导体技术,2003(01):58-59+65.
- [23] 张树贵.CAN总线分析[J].电子工业专用设备,2013,42(05):42-45.
- [24] 王绍武.基于CAN总线的煤矿井下安全监测系统浅析[J].科技广场,2012(03):76-78.
- [25] 刘堃.KVM OVER IP等远程控制技术介绍及其在监测台中的应用[J].科技资讯,2014,12(20):20-21.DOI:10.16661/j.cnki.1672-3791.2014.20.034.
- [26] 迟洪鹏,战凯,李建国,何建成.基于CAN总线的地下自主铲运机控制系统设计[J].有色金属,2011,63(02):260-263.
- [27] 韩斌,高公如,丁荣诚,林海波.基于轨迹规划的挖掘机遥控系统研究与开发[J].现代制造技术与装备,2022,58(01):1-8.DOI:10.16107/j.cnki.mmte.2022.0001.
- [28] 王建新.现场总线技术在汽车电子中的应用初探[J].合肥工业大学学报(自然科学版),2005(07):787-791.
- [29] 吴亚玲,霍亮生.CAN总线技术在汽车车速表和里程表中的应用[J].新技术新工艺,2005(08):16-18.
- [30] 沈达,张立新.基于北斗/GPS定位装置的研制[J].电子世界,2013(23):67+72.
- [31] 李云飞.GPS数据采集功能的实现[J].中国水运,2015(04):48-50.DOI:10.13646/j.cnki.42-1395/u.2015.04.019.
- [32] 王东,张海辉,路艳巧.基于GPS的公交车自动报站系统的设计[J].微型机与应用,2010,29(24):86-88+92.DOI:10.19358/j.issn.1674-7720.2010.24.029.
- [33] 刘广.浅谈3D GPS控制技术在工程机械中的应用[J].价值工程,2014,33(28):36-37.DOI:10.14018/j.cnki.cn13-1085/n.2014.28.017.
- [34] 李宝俊,闫琪珉.GPS在临策工程线路测量中的应用初探[J].内蒙古科技与经济,2009(11):112-113.
- [35] 黄礼平,邓利华,肖瑶,旦增次成.多功能电子高原登山辅助仪的设计与实现[J].电子设计工程,2015,23(15):106-108+114.DOI:10.14022/j.cnki.dzsjgc.2015.15.033.
- [36] 王玉龙.GPS-RTK技术在地质找矿测量中的应用分析[J].华北国土资源,2015(03):59-60.
- [37] 徐云航.三峡枢纽河段CGCS2000坐标系测量控制网布设方法研究[J].中国水运,2021(10):139-143.DOI:10.13646/j.cnki.42-1395/u.2021.10.050.
- [38] 张海林,张卿杰,张逸飞,王健羽.嵌入式北斗定位便携终端的设计与实现[J].电子设计工程,2017,25(12):189-193.DOI:10.14022/j.cnki.dzsjgc.2017.12.045.
- [39] RUOYU FANG, CHENG CAI. Computer vision based obstacle detection and target tracking for autonomous vehicles[C]. //2020 2nd International Conference on Computer Science Communication and Network Security (CSCNS2020)(2020年第二届计算机科学,通信和网络安全国际学术会议)论文集. 2020:1-5.
- [40] 周哲,胡钊政,李娜,等.面向智能车的地下停车场环视特征地图构建与定位[J].测绘学报,2021,50(11):1574-1584.

## 附 录

智能车底盘尺寸：

