**2024-25学年格创杯设计竞赛**

**平衡小车设计报告**

Godzilla队

楼学之 张炜鹏 邱智齐

摘要

本文使用了STM32F04C8T6芯片设计了一个可以保持平衡状态且能完成特定运动任务的二轮小车。IDE为Keil，使用的库函数为标准外设库。系统主要通过三环PID进行闭环控制保持平衡、前进与转向，使用MPU6050模块进行位置状态获取，通过外部中断实现模式转换，利用蓝牙模块进行远程控制。该系统还应用PCB版，以实现较高集成度。实验结果表明，小车具有良好的性能，可适应多种现实场景，具有较高的实用价值。

关键词：PID控制 平衡小车 蓝牙远程控制 STM32F103C8T6

1. 系统方案
   1. 主控模块的论证与选择

方案一： 使用STM

32L432KC 开发板

STM32L432KC是一款32pin UFQFPN封装(非常适合开发智能便携产品)，基于ARM Cortex-M4 主频最高80MHZ，拥有256-KB Flash 64-KB SRAM，兼容Arduino naon接口的高性能开发板

优点：课程使用，较为熟悉；mbed库配套，开发相对快速简易；功能强大

缺点：社区支持少；不支持标准外设库；价格昂贵

方案二：使用STM32F104C8T6 最小系统板

STM32F104C8T6集成了工作频率为72 MHz的高性能Cortex-M3 32位RISC内核、高速嵌入式存储器（高达128 KB的Flash存储器和20 KB的SRAM存储器）提供2个12位ADC、3个16位通用定时器、2个PWM定时器以及标准和高级通信接口：多达2个I2C和SPI、3个USART、1个USB和1个CAN

优点：社区广大，支持文档多，学习资源丰富；支持标准外设库，贴近底层原理，有更好的学习意义；价格相对低廉

综合考虑采用方案二，因为社区大，资源丰富，支持标准外设库，外设资源与算力基本满足需求，符合本组实际情况

* 1. 硬件控制方案的论证与选择

方案一：利用电机PID控制，实现平衡及特定运动任务。

优点：方案相对成熟，技术相对简单，参考资料丰富

缺点：缺乏新意

方案二：前后二轮，利用舵机控制笼头角度实现平衡，类似自行车的平衡原理

优点：创新，转向更为灵活。

缺点：技术复杂；机械结构精致，实现有一定难度，损坏率较高；软件实现较为复杂；资料文档少

方案三：利用角动量守恒原理实现平衡

优点：方案新颖，角度出众

缺点：实现难度大；需要不停转动，能量消耗大；

综合考虑选择了方案一。由于其简单易上手，教程丰富，各种模块驱动移植完善，比较符合本队基本情况。

* 1. 开发模式选择

IDE选择

方案一：利用MbedOS线上开发

优点：简单易行，贴近课程，较为熟悉

缺点：抽象层次高，难以进行精细化调整；网上参考资源少

方案二：利用 Keil 开发

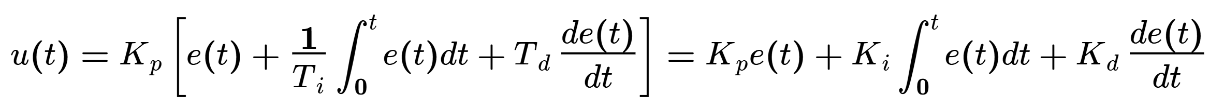
优点：主流开发软件，网上教程多

缺点：相对复杂，学习成本高

综合考虑选择了方案二。由于其相比于Mbed更有利于获取帮助支持。

1. 理论分析与计算
   1. PID控制部分

采用三级PID控制。PID是比例积分微分控制（proportional-integral-derivative control）的简称。其计算公式为



其中：

* + 1. 平衡环PD控制

代码实现：

int Balance(float Goal\_Angle , float Real\_Angle , float Real\_gyro\_Y)

{

int temp;

temp=Balance\_Kp \* (Real\_Angle - Goal\_Angle) + Balance\_Kd \* Real\_gyro\_Y;

return temp;

}

对于输入角度（P）和输入加速度（D）进行计算，返回计算后数值。

* + 1. 速度环PI控制

代码实现：

int Speed(int Goal\_Speed , int L\_Encoder , int R\_Encoder)

{

static int Err\_LowOut\_Last;

static int Encoder\_Intergal;

static float LowOut\_Parameter;

int Err , Err\_LowOut , temp;

Err = (L\_Encoder+R\_Encoder)-Goal\_Speed;

Err\_LowOut = (1 - LowOut\_Parameter) \* Err + LowOut\_Parameter \* Err\_LowOut\_Last ;

Err\_LowOut\_Last = Err\_LowOut;

Encoder\_Intergal += Err\_LowOut;

if(Encoder\_Intergal > 20000)

{

Encoder\_Intergal = 20000;

}

if(Encoder\_Intergal < -20000)

{

Encoder\_Intergal = -20000;

}

if(Flag\_Sleep == 1)

{

Encoder\_Intergal = 0;

}

temp=Err\_LowOut \* Speed\_Kp + Encoder\_Intergal \* Speed\_Ki;

return temp;

}

通过计数器获得编码器转速，再进行低通滤波得到速度。离散型数据累加代替积分（I），同时注意积分限幅，再结合比例（P）算法，最后计算出数据。

* + 1. 转向环PD控制

代码实现：

int Turn(float gyro\_Z , int Goal\_Turn)

{

int temp;

temp=Turn\_Kp \* Goal\_Turn + Turn\_Kd \* gyro\_Z;

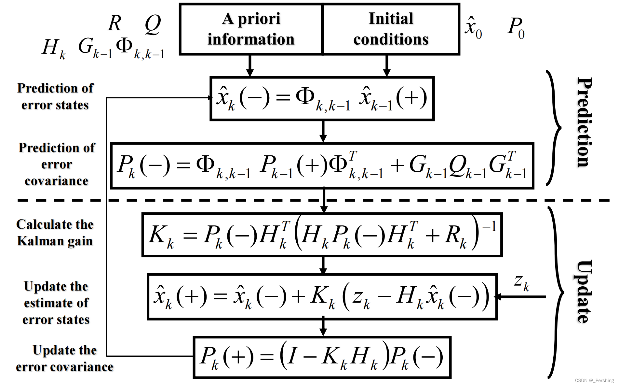
return temp;

}

传入目标转向参数和角速度，计算出相应数据，最后差分地加载在输出的PWM波形上。

* 1. MPU6050 数据处理部分
     1. 理论

本实验采用卡尔曼滤波处理MPU6050 获得的姿态数据。



* + 1. 代码部分

鉴于卡尔曼滤波原理较为复杂，本组采用了开源的库函数进行滤波。

1. 电路设计
   1. 片上资源配置

|  |  |  |  |
| --- | --- | --- | --- |
| IO编号 | 资源说明 | 外设 | 备注 |
| A12 | 普通IO | MPU6050的INT引脚 | 检测数据中断 |
| B8,B9 | 普通IO | MPU6050 | IIC 通信 |
| B10, B11 | 串口3 | 蓝牙模块 | 遥控小车 |
| A8, B15, B14 | PWMA IO | 左轮电机 | 控制方向，大小 |
| A11, B12, B13 | PWMB IO | 右轮电机 | 控制方向，大小 |
| A0, A1 | TIM2\_CH1,CH2 | 左轮编码器 | 速度 |
| B6, B7 | TIM4\_CH1,CH2 | 右轮编码器 | 速度 |
| A4 | 普通IO | LED | 指示状态 |
| A15, B5 | 普通IO | OLED |  |
| A13, A14 | SWD 调试 | SWD |  |

* 1. PCB板设计