得分：98
评语：介绍完整，完成度高

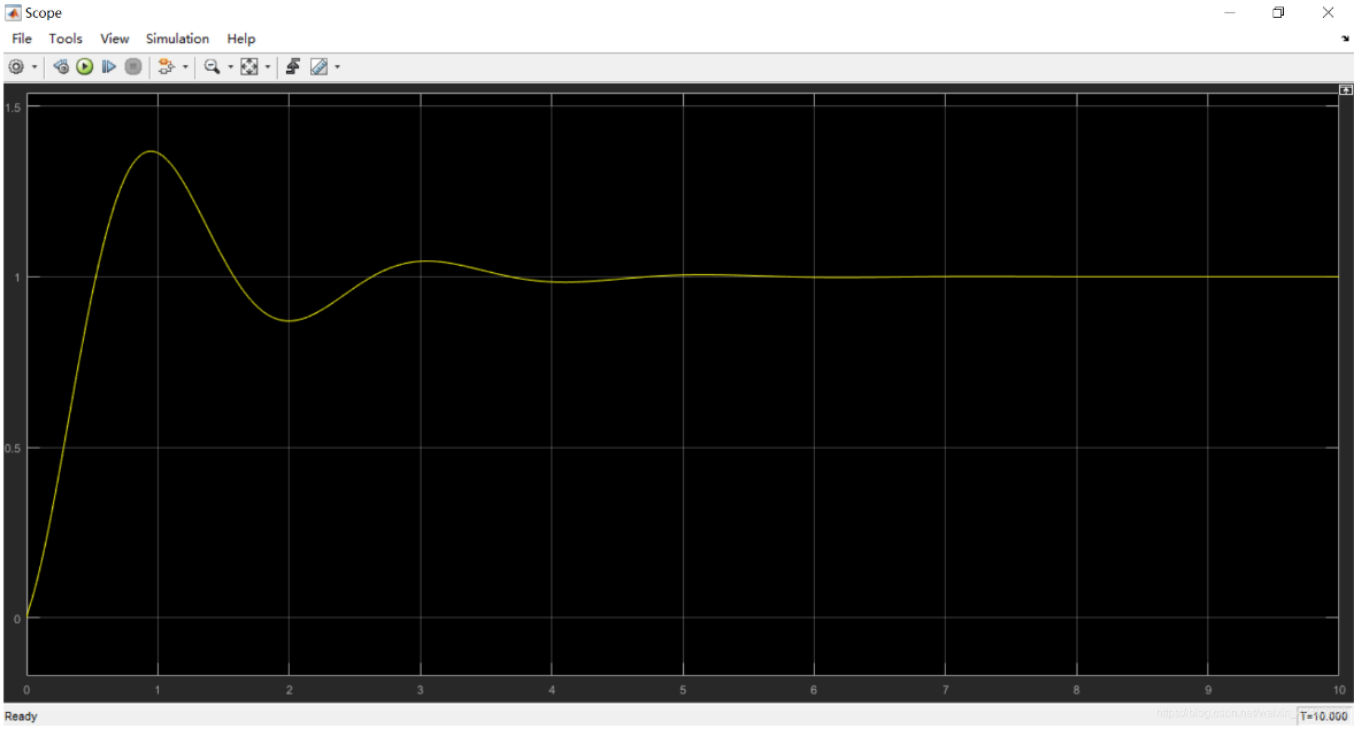
第三次汇报

一、pid控制

参考代码：

```
static float pid_cal_l(pidStructdef_l pid,float destination_l,float
current_l,uint8_t mode)
{
    static float Integral_error_l;
    if(mode == SPEED)//增量式速度环
    {
        pid.error = destination_l - current_l;
        pid.L_L_error = pid.error - pid.L_error;

        pid.out = pid.kp * pid.L_L_error + pid.ki * pid.error;

        pid.L_error = pid.error;
//      printf("pid.out=%f\r\n",pid.out);

        if(pid.out<0) pid.out=0;
        if(pid.out>1000) pid.out=1000;
        return pid.out;
    }
    else//位置环
    {
        pid.error = destination_l - current_l;
        pid.L_L_error = pid.error - pid.L_error;
        Integral_error_l+=pid.error;

        pid.out = pid.kp * pid.error + pid.ki*Integral_error_l + pid.kd *
pid.L_L_error;

        pid.L_error = pid.error;
//      printf("pid.out=%f\r\n",pid.out);

        if(pid.out<0) pid.out=0;
        if(pid.out>1000) pid.out=1000;
        return pid.out;
    }
}
```

1.可以先仿真



2.采用上位机仿真，通过无线串口传输数据，观看上位机曲线

3.看曲线超调量和收敛速度，进行参数调整

调参心得：

```
/*************************************************************/
// 参数整定找最佳，从小到大顺序查；
// 先是比例后积分，最后再把微分加；
// 曲线振荡很频繁，比例度盘要放大；
// 曲线漂浮绕大湾，比例度盘往小扳；
// 曲线偏离回复慢，积分时间往下降；
// 曲线波动周期长，积分时间再加长；
// 曲线振荡频率快，先把微分降下来；
// 动差大来波动慢。微分时间应加长；
// 理想曲线两个波，前高后低四比一；
// 一看二调多分析，调节质量不会低；

// 若要反应增快，增大P减小I；

// 若要反应减慢，减小P增大I；

// 如果比例太大，会引起系统震荡；

// 如果积分太大，会引起系统迟钝。
/*************************************************************/
```

4.pid作用：

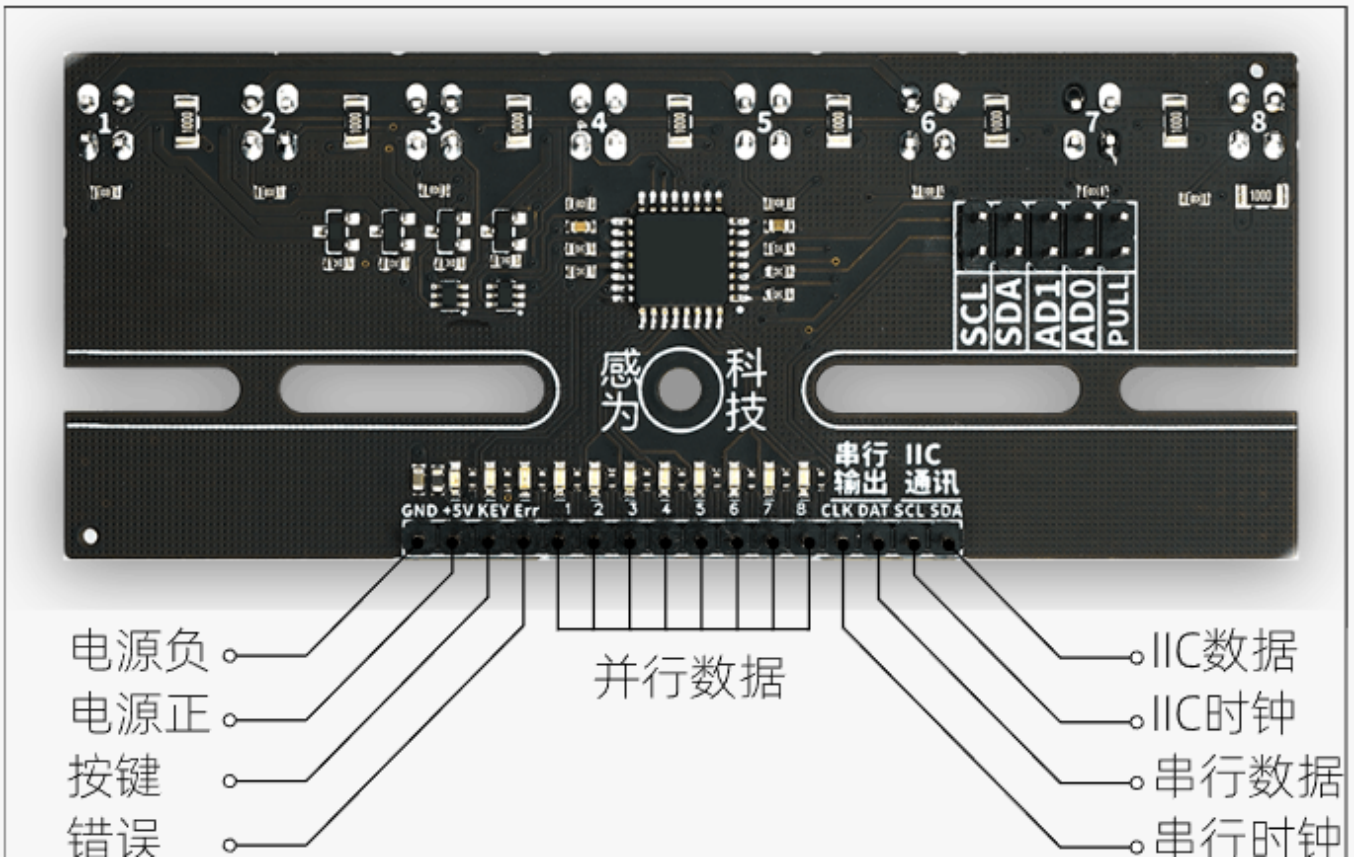
稳定性增强：

PID控制器通过调整系统的输出，以减少误差，增强系统的稳定性。它能够使系统在面对外部扰动或参数变化时，保持稳定运行。

鲁棒性：

PID控制器具有一定的鲁棒性，能够在一定范围内容忍系统参数的变化和外部干扰，使得小车更加稳定。

二、感为灰度



感为提供了两种提供信号反馈的方式：

- IO口
- i2C传输打包成一个8位char类型的数组

命令	0xB1	0xB2	0xB3	0xB4	0xB5	0xB6	0xB7	0xB8
模拟数据	1 路	2 路	3 路	4 路	5 路	6 路	7 路	8 路

实践对比：

1.IO口输入逻辑更简单，但会造成代码比较冗长，接线多

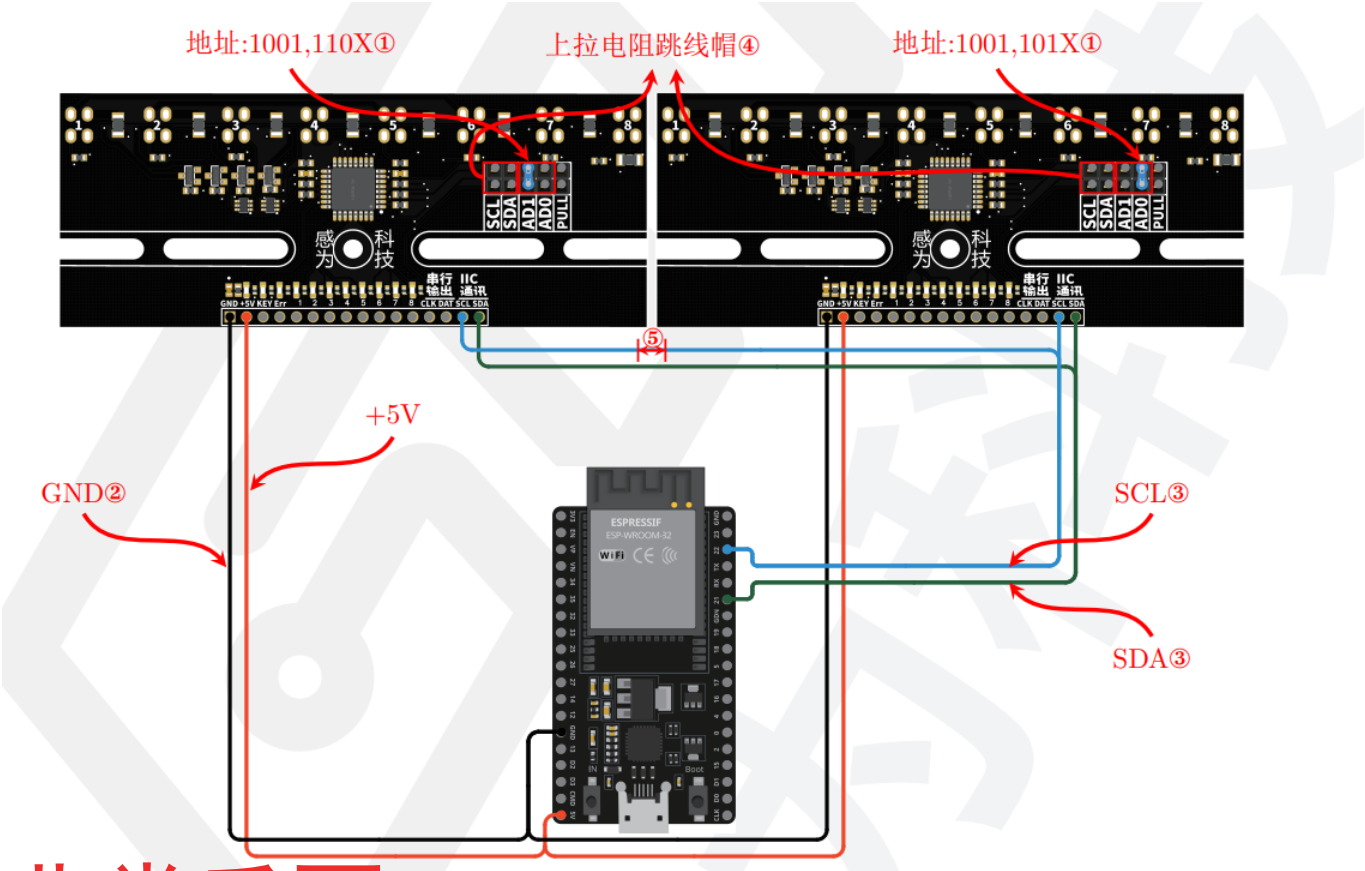
```
#define READ_FIND_LINE_OUT1 HAL_GPIO_ReadPin(GPIOA,GPIO_PIN_5)
#define READ_FIND_LINE_OUT2 HAL_GPIO_ReadPin(GPIOC,GPIO_PIN_1)
#define READ_FIND_LINE_OUT3 HAL_GPIO_ReadPin(GPIOC,GPIO_PIN_2)
#define READ_FIND_LINE_OUT4 HAL_GPIO_ReadPin(GPIOC,GPIO_PIN_3)
#define READ_FIND_LINE_OUT5 HAL_GPIO_ReadPin(GPIOA,GPIO_PIN_7)
```

```
#define READ_FIND_LINE_OUT6 HAL_GPIO_ReadPin(GPIOA,GPIO_PIN_6)
#define READ_FIND_LINE_OUT7 HAL_GPIO_ReadPin(GPIOC,GPIO_PIN_10)
#define READ_FIND_LINE_OUT8 HAL_GPIO_ReadPin(GPIOC,GPIO_PIN_13)

uint8_t get_LedFind_Scan(void)
{
    low_pin_count = 0;
    if (READ_FIND_LINE_OUT1 == GPIO_PIN_RESET) low_pin_count++;
    if (READ_FIND_LINE_OUT2 == GPIO_PIN_RESET) low_pin_count++;
    if (READ_FIND_LINE_OUT3 == GPIO_PIN_RESET) low_pin_count++;
    if (READ_FIND_LINE_OUT4 == GPIO_PIN_RESET) low_pin_count++;
    if (READ_FIND_LINE_OUT5 == GPIO_PIN_RESET) low_pin_count++;
    if (READ_FIND_LINE_OUT6 == GPIO_PIN_RESET) low_pin_count++;
    if (READ_FIND_LINE_OUT7 == GPIO_PIN_RESET) low_pin_count++;
    if (READ_FIND_LINE_OUT8 == GPIO_PIN_RESET) low_pin_count++;

    if (low_pin_count>=4) final_intersection++;
  else if   (low_pin_count==0) final_intersection ++;
    else final_intersection=0;
    if  (final_intersection>10){ intersection = 1; final_intersection=0;}

    if(READ_FIND_LINE_OUT4 == 0)
    {
        return 3;
    }
    else if(READ_FIND_LINE_OUT5 == 0)
    {
        return 4;
    }
    else if(READ_FIND_LINE_OUT6 == 0)
    {
        return 5;
    }
    else if(READ_FIND_LINE_OUT3 == 0)
    {
        return 2;
    }
    else if(READ_FIND_LINE_OUT2 == 0)
    {
        return 1;
    }
    else if(READ_FIND_LINE_OUT7 == 0)
    {
        return 6;
    }
    else if(READ_FIND_LINE_OUT1 == 0)
    {
        return 0;
    }
    else if(READ_FIND_LINE_OUT8 == 0)
    {
        return 7;
    }
    else
```

```
    {
        return 3;
    }
```

2.i2C传输，逻辑会比较复杂，但是代码精简可读性高。接线少，而且会更加稳定。

```
if (GET_NTH_BIT(recv_value, 1) == 0) {
            // 第1个探头数据为0

        } else {
            // 第1个探头数据为1

        }

        if (GET_NTH_BIT(recv_value, 2) == 0) {
            // 第2个探头数据为0

        } else {
            // 第2个探头数据为1
        }


        if (GET_NTH_BIT(recv_value, 8) == 0) {
            // 第8个探头数据为0

        } else {
            // 第8个探头数据为1
        }
//对比整排
        if (recv_value == 0xE7) { // = 0b11100111
            // 中间两个45探头数据为0 其他为1
        }
```



非常重要：

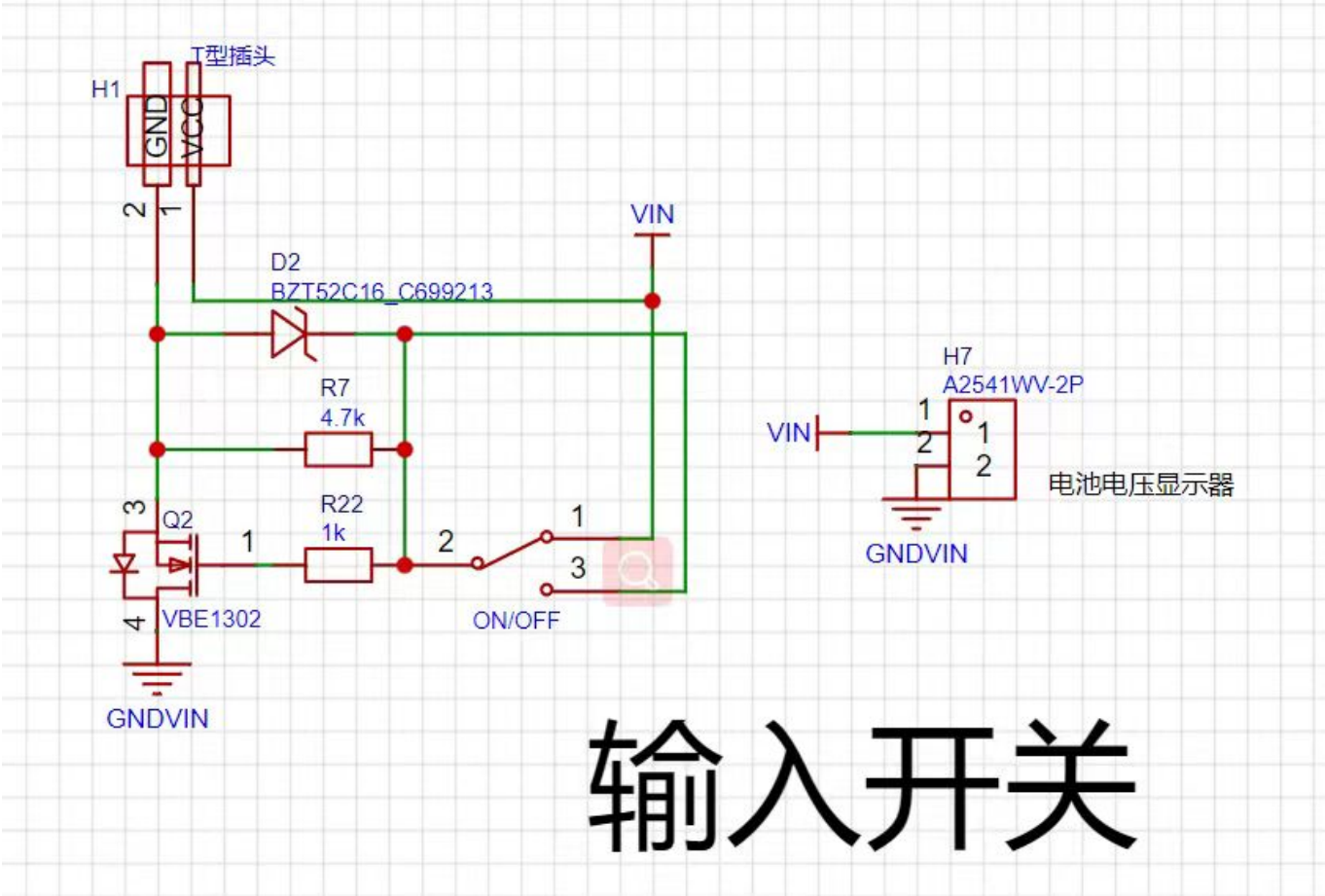
②： 主控必须与灰度传感器共地，共地才能建立通讯。

三、串口屏

四、小车主板

1.电源输入模块

考虑到电池电压为12V，改进了单个开关组成的输入开关，减少开关过程中的点火损耗。也可以直接并一个铝电解电容缓冲一下电压，使电压能够较为平稳地输入。



2.稳压电路

在直流稳压芯片里分为两种：LDO（低压线差性稳压）和开关稳压器（DCDC）。这张图是他们的差异总结：

	LDO（低压线差性稳压）	开关稳压器（DCDC）
输出功率	适合低压差小电流	适合高压差大电流
纹波	纹波小	纹波大
效率	效率低	效率高
发热	发热高	发热低
成本	成本低	成本高
易用性	简单	复杂
静态功耗	静态功耗低	静态功耗高

CSDN @大淼同学

在这里我们的电源只起到三个作用：芯片供电、外部模块供电、电机供电，后续可能还有舵机供电等。 在设计稳压电路时除了要考虑器件所需电压，还需考虑电流需求。 一些普通外设可能电流只要求几十或几百mA,所以只需要选择一款稳定且输出电流在1A左右的稳压芯片。但考虑到以后的拓展需求，可能会增加舵机，以普通996R舵机为例，正常工作电流在800mA左右，但堵转电流可达1.5A，所以舵机电源需采用输出电流2A以上的稳

压芯片。我们选择的是TPS5430DDAR，其输出电流连续3A，峰值可达4A。

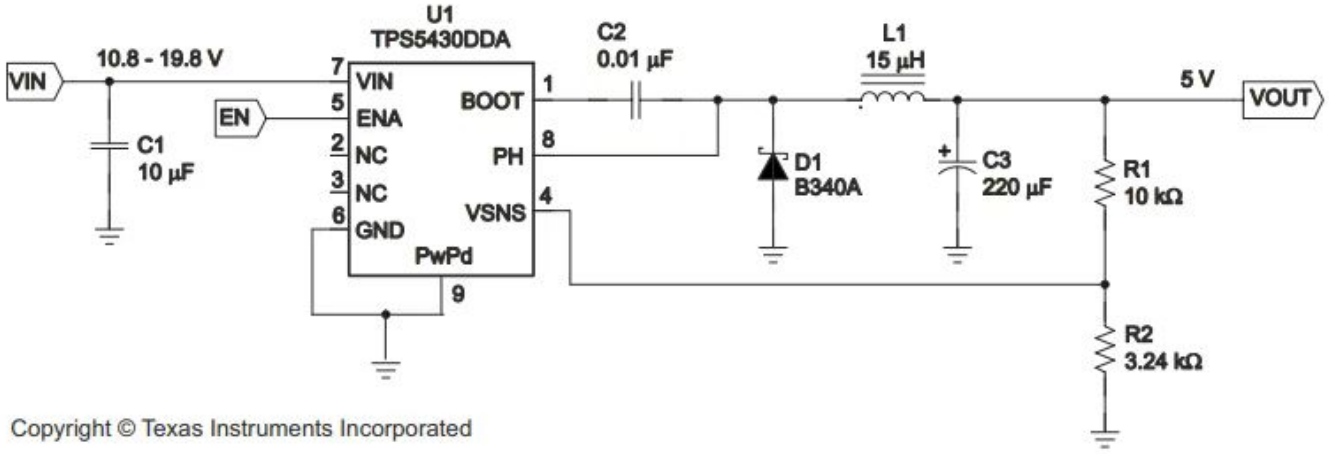


图 8-1. 应用电路，12V 输入到 5.0V 输出