四路巡线传感器

1. 简介

此巡线传感器是红外探测法为工作原理的传感器。这个传感器上有 4 个探头，每个探头有一个接收管和一个发射管。

当红外反射越强(白色)时，输出越大，红外放射越弱（黑色），输出越小。探测器离黑色线越近输出越小，由此可以通过输出模拟量判断黑线的距离远近，数值越小的传感器离黑线越近。传感器的灵敏度可以通过调节传感器上的旋钮来调整阈值，顺时针转动可以增强灵敏度，逆时针则减弱灵敏度。

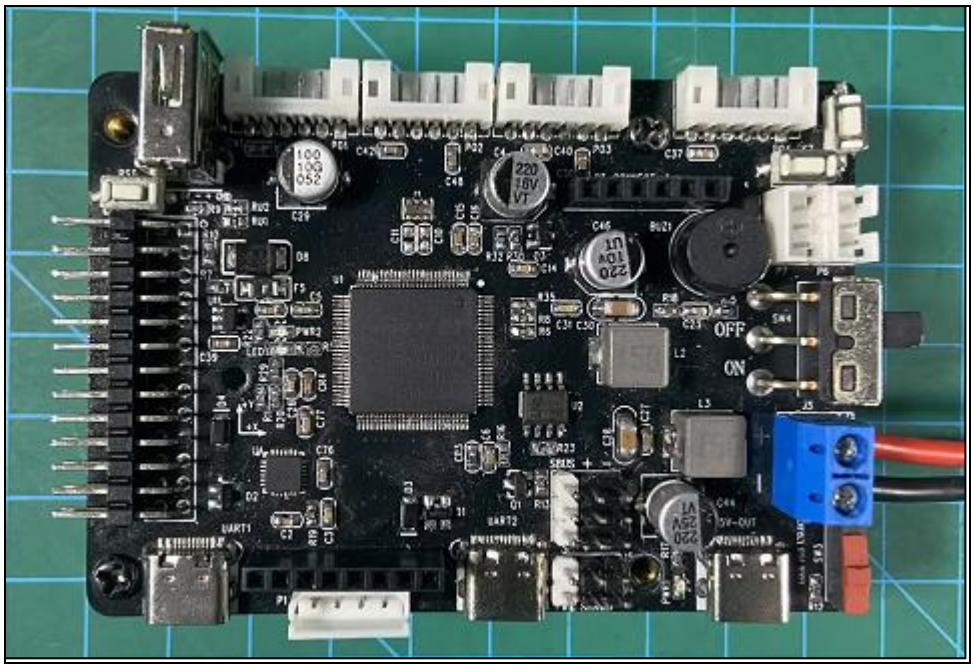
因此这款传感器可以识别黑白线，应用于智能小车、机器人的巡线运动，且可巡较为复杂的路线。同时，这款此传感器的四路巡线探头合理优化间距布局，内侧两路在黑线内精准识别、外侧两路可提前探测到大弯道的黑线并提前预警；具有探测速度快、适应性良好等优点，下图为传感器的示意

1.1 程序路径

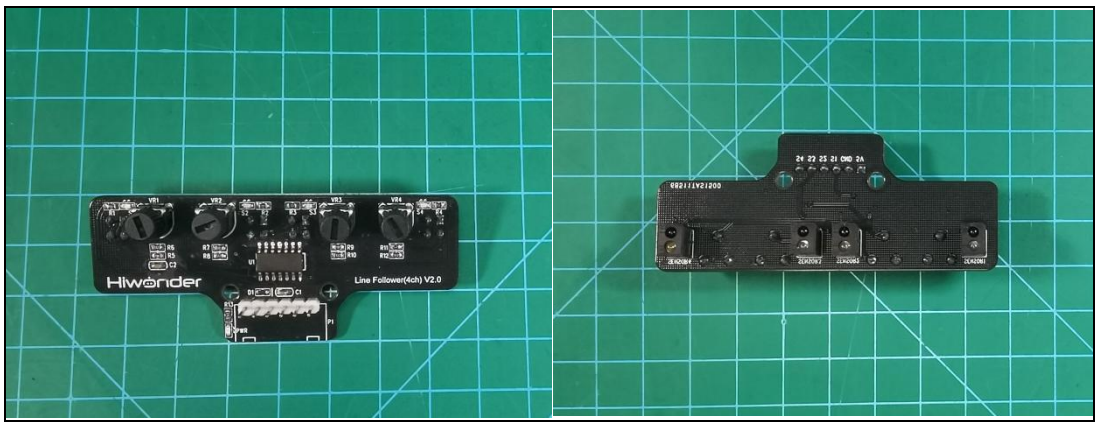
本小节内容实现的源码位于“**四路巡线\程序附录\RosRobotControllerM4-line_follower**”的 Hiwonder/System 的“line_follower.c ”使用 keil5（具体的安装方式可以参考之前的内容）打开工程文件。

1.2 硬件连接说明

① stm32 主控板；



② 四路巡线传感器；

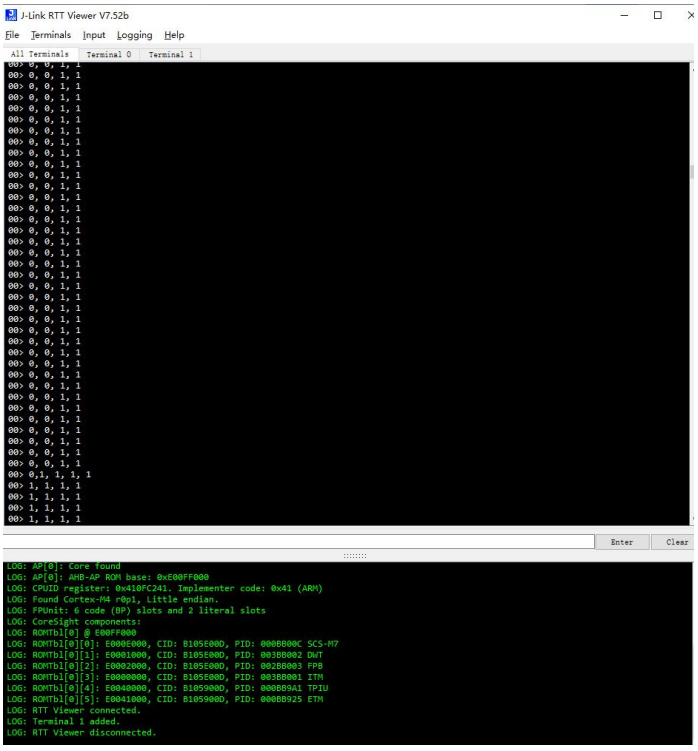


1.3 接口说明：

引脚	引脚说明
5V	电源输入
GND	电源地
S1	PC1
S2	PC0
S3	PE3
S4	PE2

1.4 实例结果：

使用“J-Link RTT Viewer”软件进行打印，如下图所示：



打印的内容 0、1 为四路巡线的接收到黑线，白线。

2. 参数说明：

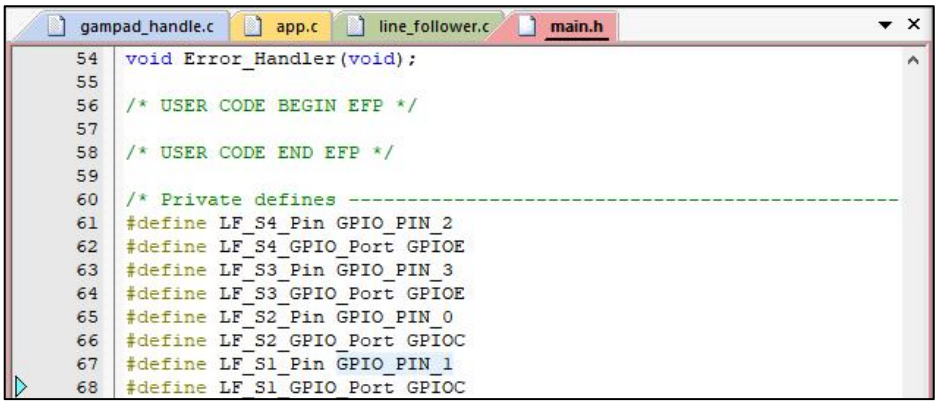
2.1 规格参数

（该资料适合用于麦轮，差速，履带，阿克曼小车）

工作电压	DC5V
工作电流	140mA
工作温度	0° C~50° C
灵敏度调节	微型电位器调节
通讯方式	GPIO 输入
PWR 说明	传感器供电后亮起
具有 4 个探头，每个探头都有一个红外发射器和一个红外接收器。	
4 个探头分别独立工作。	

2.3 源码分析

在这里我们运用了 STM32F4 HAL 库设置了 STM32F407VET6 的 PC0、PC1、PE2、PE3 为输入检测，如下图所示：



通过 PID 控制：

```
float line_folower(float kp, float ki, float kd)
{
    static float error = 0, last_error = 0, integral = 0;
    bool s1 = HAL_GPIO_ReadPin(LF_S1_GPIO_Port, LF_S1_Pin);
    bool s2 = HAL_GPIO_ReadPin(LF_S2_GPIO_Port, LF_S2_Pin);
    bool s3 = HAL_GPIO_ReadPin(LF_S3_GPIO_Port, LF_S3_Pin);
    bool s4 = HAL_GPIO_ReadPin(LF_S4_GPIO_Port, LF_S4_Pin);
```

这个程序是读取每个巡线传感器的状态。将每个传感器状态存储在一个布尔变量中（s1，s2，s3，s4）

函数 line_follower **三个参数**，分别代表 PID 控制器的比例、积分和微分系数。

定义三个静态浮点变量：

error（**当前的错误**）；

last_error（**上一次的错误**）；

integral（**错误的积分**）

。

```
if( !s1 && s2 && !s3 && !s4) { /* 0 1 0 0 */
    error = -1;
} else if(s1 && s2 && !s3 && !s4) { /* 1 1 0 0 */
    error = -2;
} else if(s1 && !s2 && !s3 && !s4) { /* 1 0 0 0 */
    error = -6;
} else if(!s1 && !s2 && s3 && !s4) { /* 0 0 1 0 */
    error = 1;
} else if(!s1 && !s2 && s3 && s4) { /* 0 0 1 1 */
    error = 2;
} else if(!s1 && !s2 && !s3 && s4) { /* 0 0 0 1 */
    error = 6;
} else {
    error = 0;
}
```

这个条件（if-else）结构，用于根据四路巡线传感器的状态计算错误值。

每种情况都对应于一个特定的传感器组合，给出不同的错误值。（**详情可以看最下面有详细分析**）

```
    integral += error;
    integral = integral > 80 ? 80 : (integral < -80 ? -80 : integral);
    float output = kp * error + ki * integral + kd * (error - last_error);
      return output;
}
```

这个程序通过返回 PID 控制器的输出值，根据这个值用于调整底盘的角速度以跟踪线路。

这个是计算错误的积分（integral += error;）。然后，对积分进行限制，确保其值在-80 到 80 之间。

```
integral = integral > 80 ? 80 : (integral < -80 ? -80 : integral);
```

这个部分程序对积分值进行了限制，防止积分过大，将积分值被限制在-80 到 80 之间。

```
float output = kp * error + ki * integral + kd * (error - last_error);
  return output;
```

这个程序计算了 PID 控制的输出，kp、ki、kd 是比例、积分、微分的系数。然后返回 PID 控制器的输出，根据输出可以调整小车底盘的行驶方向。

kp * error 计算的是比例项，它使得控制器的输出和当前的位置偏移成正比；

ki * integral 计算的是积分项，它使得控制器的输出和过去所有的位置偏移的累积成正比；

kd * (error - last_error)计算的是微分项，它使控制器的输出和位置偏移的变化速度成正比。

```
if( !s1 && s2 && !s3 && !s4) { /* 0 1 0 0 */
    error = -1;
} else if(s1 && s2 && !s3 && !s4) { /* 1 1 0 0 */
    error = -2;
} else if(s1 && !s2 && !s3 && !s4) { /* 1 0 0 0 */
    error = -6;
} else if(!s1 && !s2 && s3 && !s4) { /* 0 0 1 0 */
    error = 1;
} else if(!s1 && !s2 && s3 && s4) { /* 0 0 1 1 */
    error = 2;
} else if(!s1 && !s2 && !s3 && s4) { /* 0 0 0 1 */
    error = 6;
} else {
    error = 0;
}
```

接上面程序的详细拆分这部分代码是根据四路巡线传感器的读数（s1, s2, s3, s4）来确定当前小车底盘的位置偏移，也即“错误”(error)。这个错误值将被用于 PID 控制，以调整小车的行驶方向。

传感器的读数是布尔值，true 表示该传感器上检测到了线路，false 则表示没有检测到，在这个程序中，我们四路巡线传感器 s2 和 s3 在机器人的正中，s1 和 s4 分别在机器人的左侧和右侧。

下面是对每个条件的详细解析：

```
if( !s1 && s2 && !s3 && !s4) { /* 0 1 0 0 */
    error = -1;
```

这表示当只有 s2 检测到线路时，小车底盘偏离线路的方向是向左，所以设定一个负的错误值-1。

```
else if(s1 && s2 && !s3 && !s4) { /* 1 1 0 0 */
  error = -2;
```

这表示当 s1 和 s2 检测到线路时，小车底盘更偏离线路的方向是向左，所以设定一个更大的负错误值-2。

```
else if(s1 && !s2 && !s3 && !s4) { /* 1 0 0 0 */
  error = -6;
```

这表示当只有 s1 检测到线路时，小车底盘严重地偏离线路的方向是向左，所以设定一个更大的负错误值-6。

```
else if(!s1 && !s2 && s3 && !s4) { /* 0 0 1 0 */
  error = 1;
```

这表示当只有 s3 检测到线路时，小车底盘偏离线路的方向是向右，所以设定一个错误值-1。

```
else if(!s1 && !s2 && s3 && s4) { /* 0 0 1 1 */
  error = 2;
```

这表示当 s3 和 s4 检测到线路时，小车底盘更偏离线路的方向是向右，所以设定一个更大的错误值+2。

```
else if(!s1 && !s2 && !s3 && s4) { /* 0 0 0 1 */
  error = 6;
```

这表示当只有 s4 检测到线路时，小车底盘严重地偏离线路的方向是向右，所以设定一个更大的错误值+6。

```
} else {
    error = 0;
}
```

这表示当其他情况，如所有传感器都没有检测到线路，或者所有传感器都检测到线路时，程序认为小车底盘没有偏离线路，所以设定错误值为 0。

注意：【切勿正负极接反】