Tracker Sensor 用户手册

产品概述

Tracker Sensor 是红外循迹传感器，常用于制作循迹智能小车。

Tracker Sensor 采用 ITR20001/T 红外反射传感器，ITR2001/T 传感器的红外发射二极管不断发射红外线，当发射出的红外线被物体反射时，被红外接收器就收，并输出模拟值。输出模拟值和物体距离以及物体颜色有关。通过计算五路输出的模拟值，判断循迹线位置。

规格

工作电压：3.3V ~ 5V
产品尺寸：78mm × 18mm
探头间距：16mm
固定孔尺寸：3mm
感应距离：1cm ~ 5cm

主要用途

智能小车或机器人寻线，避悬崖防跌落等

小车循迹原理简介

Tracker Sensor 有五路模拟输出，模拟量输出和距离，物体颜色有关。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。相对于其他只能输出高低电平的轨迹传感器而言，本产品的 5 路模拟量输出可以反馈黑线的距离，反馈更为准确。
小车算法可以分为 3 部分：

第一部分：归一化校准

相同的颜色，距离，不同的探测器输出会有所不同，而且不同环境输出的模拟量范围也不一样。如果采用 10 位 AD 采集。理论上输出范围会在 0~1023 之间。但是实际上输出最小 Min 会大于 0，

版本：V1.1，日期：2016 年 12 月 1 日

最大值 Max 会比 1023 小。由此为了减少传感器自身以及环境的影响，作归一化处理，归一化处理实际上就是将 Min~Max 的数值范围转为 0~1 范围线性转换如下：

y = (x - Min) / (Max - Min)

（其中 x 为探测器输出值，y 为转换后的值，Max、Min 分别为输出的最大值和最小值）

为了数据处理方便可以将数据放大 1000 倍。则

y = (x - Min) * 1000 / (Max - Min)

转换后数据范围为 0~1000.其中 1000 表示探测器远离黑线，0 表示探测器在黑线正中。

其中 Min 和 Max 的值是由程序在运行过程中多次采集传感器的值而获得，并保存起来，采集过程中不断移动小车，使得采集到的最大最小值更加接近实际情况。

第二部分：加权平均

将五个探测器输出数据通过归一化处理后，得到五个反应探测器离黑线相对距离的数据。为了尽可能准确地确定路线的中心线。通过加权平均的方式将这 5 个数值转变成一个数值。公式如下：

y = (0 * value0 + 1000 * value1 + 2000 * value2 + 3000 * value3 +4000 * value4) / (value0 + value1 + value2 + value3 + value4)

其中 0, 1000, 2000, 3000, 4000 分别为从左到右 5 个探测器的权，value0~value4 为探测器归一化后的数据。

则经过出后的数值范围为 0~4000。代表黑线的位置。例如 2000 则表示黑线在模块的正中间。0 表示在黑线在模块的最左侧，4000 表示黑线在最右侧。

为了使模块探测的精度更高，对模块的高度和黑线有所要求。黑线应该等于或略小于探测器的距离(16mm)。高度为黑线在两个传感器正中间时，两个传感器都能够刚好探测到为宜。

第三部分：PID 控制

由第二部分可得到黑线的位置，为了使小车一直沿着黑线走，则必须保证黑线在小车正下方，此时加权平均后的输出应该为 2000。为了使小车走的更加平滑，减少左右摇摆。采用位置式 PID 控制。关于 PID 算法网上有很多介绍，此处不再详细讲解，只是大概介绍一下。

PID 是指通过比例(P)，积分(I)，微分(D)对误差进行反馈调节。主要算法如下：

proportional = position - 2000;

derivative = proportional - last_proportional;

integral += proportional;

last_proportional = proportional;

power_difference = proportional * Kp + integral * Ki + derivative * Kd;

其中：

理想情况下，加权平均后的输出数据为 2000，即黑线在正中间。

比例项(proportional)为当前的位置(Position)减去目标位置(2000)，表示位置误差，正数表示小车偏右，负数表示小车偏左。

积分项(integral)数据为每次误差的总和，绝对值越大，表示误差累积值越大，表示小车越偏越远了。

微分项(derivative)为当前误差和上次误差的差值，反映小车的响应速度，数值越大，响应速度越快。

调节 Kp, Ki, kd 三个参数可获得最佳性能。先调节 Kp 参数，Ki, Kd 设为 0，不断调整 Kp 值使得小车可以循线，然后调节 Ki 和 Kd 值，参数可以设小一点，或者为 0。

AlphaBot 循迹模块示例程序分析

下面结合 AlphaBot 智能小车讲解一下 Tracker Sensor 循迹程序。这里以 Arduino 的程序作为示例，Tracker Sensor 的库文件主要包含 TRSensors.cpp 和 TRSensors.h 两个文件。

TRSensors.cpp 主要包含下面这几个函数

```
TRSensors();

void AnalogRead(unsigned int *sensor_values);

void calibrate();

void readCalibrated(unsigned int *sensor_values);
```

```
int readLine(unsigned int *sensor_values, unsigned char white_line = 0);
```

其中 TRSensors()为初始化函数，初始化相应的管脚，以及申请内存用作存储各个传感器的 Max, Min 值。

AnalogRead()函数为读取五路探测器的模拟值，AlphaBot 是通过 TLC1543 AD 芯片进行 AD 转换，而非 Arduino 芯片的 AD 管脚。如果将 Tracker Sensor 接到 Arduino 的 A0~A4 管脚需修改此函数。

calibrate()函数为校准函数，通过多次采集数据，确定 Max，Min 值。故校准阶段时，小车需在黑线中紧贴地面左右摇晃。确保取得的 Max,Min 准确。

readCalibrated()函数为归一化校准，对应原理的第一部分，通过归一化线性转换，将数据转为 0~1000 范围内，其中 1000 表示探测器远离黑线，0 表示探测器在黑线正中。

readLine()函数为读取循迹线的位置，对应原理中的第二部分。通过加权平均算出寻迹线的位置。数值范围为 0~4000，0 表示在黑线在模块的最左侧，4000 表示黑线在最右侧。

原理中第三部分在循迹主程序 Infrared_Line_Tracking.ino 中实现，主要代码如下。

```
// Get the position of the line.  Note that we *must* provide
// the "sensors" argument to read_line() here, even though we
// are not interested in the individual sensor readings.
unsigned int position = trs.readLine(sensorValues);
//  for (unsigned char i = 0; i < NUM_SENSORS; i++)
//  {
//    Serial.print(sensorValues[i]);
//    Serial.print('\t');
//  }
//Serial.println(position); // comment this line out if you are using raw values

// The "proportional" term should be 0 when we are on the line.
int proportional = (int)position - 2000;
derivative = proportional - last_proportional
// improve performance.
int power_difference = proportional/15 + derivative/100;
```

通过 readline()函数读取寻迹线的位置，然后算出比例项 proportional，微分项 derivative，

此处只用了 PD 算法，没有积分项，而非 PID。可以修改对应的 PD 参数，使性能更佳。

```
// Compute the actual motor settings.  We never set either motor
// to a negative value.
const int maximum =100;

if (power_difference > maximum)
  power_difference = maximum;
if (power_difference < - maximum)
  power_difference = - maximum;

if (power_difference < 0)
{
  analogWrite(ENB,maximum + power_difference);
  analogWrite(ENA,maximum);
}
else
{
  analogWrite(ENB,maximum);
  analogWrite(ENA,maximum - power_difference);
}
```

最后一步就是通过 PD 算得的修改项 power_difference 去调节小车左右轮的 PWM 值，实现小车沿着寻迹线运动。

操作与现象

本节只是介绍模块的简单测试程序，关于 AlphaBot 使用 Tracker Sensor 实现循迹的操作以及示例程序可以参考 AlphaBot 的相关资料。

下面以 XNUCLEO-F103RB，UNO PLUS 开发板为例。

① 将配套程序下载到相应的开发板中。
② 将串口线和模块接入开发板，给开发板上电，打开串口调试软件。
模块与开发板连接如下表所示：

端口	XNUCLEO_F103RB 引脚
IR1~IR5	A0~A4
GND	GND
VCC	3.3V

表1. 模块接入 STM32 开发板

端口	Arduino 引脚
IR1~IR5	A0~A4
GND	GND
VCC	5V

表2. 模块接入 Arduino

串口配置如下表所示：

Baud rate	9600
Data bits	8
Stop bit	1
Parity bit	None

表3. 串口配置

③ 串口会显示 5 串数据，分别对应 IR1~IR5 传感器。数据随着反射距离的远近而改变。当模块没有遮挡时，输出的大概为几十，当模块接近台面时，输出为八九百左右。