**首届长三角青少年人工智能奥林匹克挑战赛**

**主赛道——无人驾驶**

**技术说明文档**

|  |  |
| --- | --- |
| 所在省市 |  |
| 所在区 |  |
| 参赛单位 |  |
| 参赛队员 |  |

1. **项目分工**

|  |  |  |  |
| --- | --- | --- | --- |
| **序号** | **队员姓名** | **年级** | **项目中的分工及成果** |
| 1 |  |  |  |
| 2 |  |  |  |
| 3 |  |  |  |

1. **项目时间节点及说明**

|  |  |  |
| --- | --- | --- |
| **序号** | **时间节点** | **主要工作** |
| 1 |  |  |
|  |  |  |
|  |  |  |
|  |  |  |

1. **软硬件清单**

**1.硬件清单**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **序号** | **硬件名称** | **品牌/型号** | **硬件图片** | **项目中的作用** |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |

*说明：如是自己设计制作的硬件，品牌/型号栏中写“自制”。*

**2.软件开发工具**

|  |  |  |  |
| --- | --- | --- | --- |
| **开发工具名称** |  | **版本** |  |
| **编程方式类型** | 代码（语言类型： C语言 ） | | |

1. **结构设计方案**
2. **车体结构**

*说明：请图文并茂地展示。*

1. **小车正确行驶、红绿灯识别相关结构**

*说明：请图文并茂地展示。*

要使得小车正确行驶，首先就是要能正确判断行驶的道路，而脚下的道路，由2部分组成，一部分是黑色的“路面”，另一部分是路口地面上白色的“白线”。在此，我们通过安装在车底下的“光电传感器”进行检测。【配图：光电传感器】

光电传感器分为2排，前排有4个在中间，后排有2个在两侧。光电传感器可以返回1个数值，当物体的反光强度越大，则这个数值越小。

因此，经过我们的检测，当小车在黑色路面上时，6个传感器的数值几乎始终高于450，而当小车在白色路面上时，则会低于300。

因此，我们最初的设计是通过光电传感器阈值来判断小车是否到了路口，停留在了白线上。

图形用户界面, 文本, 应用程序

描述已自动生成

这段代码的好处是判断非常容易，当有2个传感器检测到光电数值低于阈值的时候就返回2，否则，为了防止程序长时间阻塞，在调用之前可以传入(int TriesLimited)来确保在尝试一定次数后，即便没有检测到白线，主程序也可以继续执行。

而对于阈值，我们使用了一个常量，因为我们发现在白线上时，几乎所有的光电返回值都小于230。

//For the use of eadcTreshold1

const int eadcTreshold1 = 230;

但是在经过一段时间的测试之后，我们发现了一定的问题：当车辆从停车区开出时，由于每个停车区【配图：颜色不一样的停车区】使用的颜色不一样。在黄色这类高反光性的颜色上，光电传感器输出的数值有可能会明显差异，甚至直接小于我们设定的阈值230。【配图：小车在彩色路面上光电数值小于230】 面对这个问题，我们想了一个改进方法：

我们只要判断小车是否在黑色路面上，当离开停车区进入黑色路面之后，当传感器数值再次减小，我们就认为上了白线。

//For the use of eadcTreshold1

const int eadcTreshold1 = 450;

因此我们拉高的传感器阈值，略小于黑色路面的输出值480，选择了450作为阈值。同时也对代码进行了修改，增加了是否上到黑色路面的检测，这样一来，当上到黑色路面之后，函数返回，主函数便可以执行下一个命令。

int ToBlackRoad(int TriesLimited) {

for (int a = 0; a < TriesLimited; a++) {

cls();

UpdateeadcSensor();

int count = 0;

if (eadcSensor[1] > eadcTreshold1) count++;

if (eadcSensor[2] > eadcTreshold1) count++;

if (eadcSensor[3] > eadcTreshold1) count++;

if (eadcSensor[4] > eadcTreshold1) count++;

//Can edit the Sensor nunmber of detected

if (count >= 2) {

return 1;

}

}

return 0;

}

经过修改之后，小车便可以顺利从停车区开出，沿着马路行驶，找到路口的白线，并且停车执行相应的操作了。

int LookupWhiteLineWithSenior(void) {

motor(1, 0); motor(2, 0);

UpdateeadcSensor();

if (eadcSensor[0] > eadcTreshold1 && eadcSensor[1] > eadcTreshold1 && eadcSensor[2] > eadcTreshold1 &&

eadcSensor[3] > eadcTreshold1 && eadcSensor[4] > eadcTreshold1 && eadcSensor[5] > eadcTreshold1) {

return 0;

}

LookupStarted:

if (eadcSensor[2] <= eadcTreshold1) {

if (eadcSensor[4] > eadcTreshold1) {

motor(2, 180);

//Tries should have A TEST! (Value of a)

for (int a = 0; a < 10000; a++) {

if (geteadc(5) <= eadcTreshold1) {

motor(2, 0);

return 1;

}

}

motor(2, -180);

for (int a = 0; a < 200; a++) {

if (geteadc(5) <= eadcTreshold1) {

motor(2, 0);

return 1;

}

}

}

}

else if (eadcSensor[4] <= eadcTreshold1) {

if (eadcSensor[1] > eadcTreshold1) {

motor(1, 200);

//Tries should have A TEST! (Value of a)

for (int a = 0; a < 50; a++) {

if (geteadc(2) <= eadcTreshold1) {

motor(1, 0);

return 1;

}

}

motor(2, -200);

for (int a = 0; a < 100; a++) {

if (geteadc(2) <= eadcTreshold1) {

motor(1, 0);

return 1;

}

}

}

}

else {

go(100, 100);

for (int a = 0; a < 50; a++) {

UpdateeadcSensor();

if (eadcSensor[1] <= eadcTreshold1 || eadcSensor[4] <= eadcTreshold1) {

goto LookupStarted;

}

}

}

motor(1, 0); motor(2, 0);

return 0;

}

接下来在测试过程中，我们发现，即便可以检测到白线，由于有时车辆在停车区域放置的时候由于默写原因放歪了，或是车辆由于打滑和两侧马达动力不同等原因，在行驶过程中有些偏离。由于路比较宽，可能并没有太大问题，但是当到了路口时，如果车辆不是正对路口，则有可能出现检测不到红绿灯或是转弯角度不对等情况。

为此，我们需要对白线进行对齐。这是我们一开始的思路，大概就是检测6个光电中已经低于阈值的光电，如果大多数左边的光电低于阈值但右边没有，那就继续使右侧轮前进，直到右侧光电也低于阈值；反之如果大多数右边的光电低于阈值但左边没有，那就继续使左侧轮前进，直到左侧光电也低于阈值。

这个思路看起来比较容易，但事实上操作起来依然不是很可行，每次车辆都还是歪的，甚至会出现校准之后比原来更歪的情况。

经过我们再三调试，我们找到了以下几个问题：

1. 每个光电的输出值会有一定误差，甚至在同一条白线上最低的光电值和最高的光电值可能要相差接近60。当我们使用检测白线时，由于我们会对所有光电取平均或是检测一定数量的光电小于阈值就执行判断，误差之间可以互相弥补，不至于出现太过明显错误。但当我们仅使用一个或两个光电进行对位时，误差会被明显放大。【附图：误差】
2. 即便光电没有到白线上，光电输出的数值已经会开始变小，很有可能明明已经低于阈值，光电其实并没有到白线上。这点非常令人难受。【附图：没到达白线但是光电减小】

对此，我们给出了一定的解决方案：

首先，我们对每个光电单独测量其完全到达白线上的阈值，并且只使用最左、最右两个固定的光电传感器进行检测，来确保当触发判断时，这两个传感器已经完全停留在了白线上。

同时我们发现，即便一侧轮子不转，另一侧轮子前进时也会带动整个小车前进，造成另一侧停留在白线上时原来一侧轮子已经越线，我们在另一侧轮子前进时进行了使另侧后退，尽可能使车辆原地自转。代码如下：

int LookupWhiteLineWithSensor(void) {

UpdateeadcSensor();

if (eadcSensor[0] > eadcTreshold1 && eadcSensor[1] > eadcTreshold1 && eadcSensor[2] > eadcTreshold1 &&

eadcSensor[3] > eadcTreshold1 && eadcSensor[4] > eadcTreshold1 && eadcSensor[5] > eadcTreshold1) {

return 0;

}

motor(1, 220); motor(2, 220);

char flagL = 0;

char flagR = 0;

while (1) {

UpdateeadcSensor();

if (eadcSensor[0] <= 190) {

flagL = 1;

motor(1, -180);

}

if (eadcSensor[5] <= 235) {

flagR = 1;

motor(2, -180);

}

if (flagL == 1 && flagR == 1) {

wait(0.1);

motor(1, 0); motor(2, 0);

UpdateeadcSensor();

printf("%d %d", eadcSensor[0], eadcSensor[5]);

return 1;

}

}

}

1. **物资收取、投放相关结构**

*说明：请图文并茂地展示。*

1. **获取通信装置编号装置相关结构**

*说明：请图文并茂地展示。*

1. **整体算法逻辑结构**

*说明：可通过文字描述或框架图来表示。*

1. **创新亮点说明**

*说明：设计、编程或调试中的创新亮点说明，可图文并茂地展示，将作为评审重点关注的部分。*

1. **备赛感悟**

*说明：学习及备赛过程中非具体技术方面的感受和感想。*