2022 年全国大学生电子设计竞赛

**具有自动泊车功能的电动车（B题）**

参赛队号：20220139



**2022** 年 **10** 月 **16** 日

**具有自动泊车功能的电动车**

摘要

该系统以正点原子STM32F407VET6作为核心板，由OpenMV模块、循迹检测模块、蜂鸣器模块和前轮转向式四轮车构成。该自动泊车系统的设计采用了模块化思想，完成了色块识别模块及自动泊车模块的设计，达到集中检测车库、自动泊车于一体的目的，实现了自动泊车功能。

系统主要运用色块识别方法以及自动泊车模块完成自动泊车功能。色块识别方法是对摄像头采集的图像中的感兴趣区域（ROI）中的黑色像素数进行检测和统计，当像素数大于设定的阈值，发送高电平信号，标志检测完成；自动泊车模块是利用STM32的定时器输出PWM波控制电机速度以及舵机旋转角度，通过编码器确定小车运行轨迹，当车身进入车库内通过循迹避障模块检测地面库线，待小车全部进入车库内，最终实现自动泊车功能。

关键词：自动泊车，色块识别，STM32

目录

[1.引言 1](#_Toc116840559)

[2.方案论证 1](#_Toc116840560)

[2.1供电方案比较与选择 1](#_Toc116840561)

[2.2电机驱动方案比较与选择 1](#_Toc116840562)

[2.3摄像头方案比较与选择 2](#_Toc116840563)

[3.理论分析与计算 2](#_Toc116840564)

[3.1电机与舵机PWM波配置 2](#_Toc116840565)

[3.2 OpenMV检测车库方法 3](#_Toc116840566)

[4.电路与程序设计 3](#_Toc116840567)

[4.1系统总体设计 3](#_Toc116840568)

[4.2单元电路设计 4](#_Toc116840569)

[4.2.1单片机最小系统设计 4](#_Toc116840570)

[4.2.2电机驱动模块设计 4](#_Toc116840571)

[4.2.3独立按键模块设计 5](#_Toc116840572)

[4.2.4巡线避障模块设计 5](#_Toc116840573)

[4.3系统程序设计 6](#_Toc116840574)

[5.测试方案与测试结果 6](#_Toc116840575)

[5.1测试仪器 6](#_Toc116840576)

[5.2测试方案 6](#_Toc116840577)

[5.2.1“邻库有车”情况停车测试 6](#_Toc116840578)

[5.2.2“邻库无车”情况停车测试 7](#_Toc116840579)

[5.3测试结果 7](#_Toc116840580)

[5.4测试结果分析 8](#_Toc116840581)

[6.结论 8](#_Toc116840582)

[附录1.部分程序代码 1](#_Toc116840583)

[1.1主程序代码 1](#_Toc116840584)

[1.2定时器代码 5](#_Toc116840585)

[1.3电机控制代码 7](#_Toc116840586)

[1.4编码器代码 8](#_Toc116840587)

[1.5舵机控制代码 9](#_Toc116840588)

[1.6色块识别代码 9](#_Toc116840589)

[附录2.电路连接原理 13](#_Toc116840590)

[附录3.实验结果照片 14](#_Toc116840591)

# 1.引言

题目要求设计一个具有自动泊车功能的电动车，具备以下功能：（1）能够在相邻车库有车的情况下完成倒车入库；（2）能够在相邻车库有车的情况下完成侧方位入库；（3）能够在邻库无车的情况下完成倒车入库；（4）能够在邻库无车的情况下完成侧方位入库；（5）能够连续完成（1）、（2）两项功能；（6）能够连续完成（3）、（4）两项功能。

为了设计实现该功能的电动车，本设计使用STM32F407作为核心处理器，利用色块识别算法和自动泊车算法，通过OpenMV利用色块检测方法识别车库，检测到车库之后，声光显示，然后通过自动泊车算法进行倒车，倒车过程中通过循迹避障模块检测地面边缘库线，直至小车完整入库。

# 2.方案论证

## 2.1供电方案比较与选择

方案一、电源直接给系统供电

由于电源直接接入系统会一直放电，电量损耗快，影响电动车的稳定行驶，导致测试结果出现偏差，同时也无法实时监测电源的输出电压，容易造成电源过放。

方案二、电源接入稳压模块给系统供电

将电源接入稳压模块，通过稳压模块给系统供电，保证了电源电压的输出稳定，同时模块上板载的数码管模块可以实时显示电源输出电压，方便及时充电保护。

经过综合考虑，最终选用方案二。

## 2.2电机驱动方案比较与选择

方案一、使用继电器

该方案电路简单可靠，但不容易实现精细控制。

方案二、使用分立元件构成电机驱动电路

该方案结构简单，价格低廉，在实际中应用广泛，缺点是该方案工作性能不够稳定，不满足比赛的需要。

方案三、使用DRV8701E作为电机驱动芯片

DRV8701E具有抗电磁干扰，驱动力强，响应频率高，体积较小等优良性能，同时带有使能控制端，方便使用。同时控制电机正反转无需两路PWM，只需一路PWM加高低电平即可实现正反转，节省单片机PWM资源。使用该芯片作为电机驱动，操作方便，稳定性好，性能优良。

结合比赛精度需求的考虑，最终选用方案三。

## 2.3摄像头方案比较与选择

方案一、使用MT9V034神眼摄像头

MT9V034具有全局快门、高动态成像等优点，使用过程中能以60fps的形式输出。根据车库编写特征识别函数，让电动车在行驶过程中通过摄像头识别车库相应特征，做出相应动作。但是特征识别函数编写工作量大，同时摄像头在工作时容易产生图像畸变，导致无法识别或误识别情况发生，识别精度低。

方案二、使用OpenMV

OpenMV以STM32F427CPU为核心，集成了OV7725摄像头芯片，能够高效的实现核心机器视觉算法。可以通过摄像头对车库边缘色块的检测，在设定好的检测区域内，一旦符合条件的像素点个数达到设定值，即可认定为检测到车库，检测精度高。

综合考虑系统设计的需要，本系统选用方案二。

# 3.理论分析与计算

本系统将小车自动泊车任务分解为小车驱动以及车库识别两个模块。通过输出PWM波驱动直流电机与舵机控制小车运动，小车行进过程中进行车库识别。输出PWM波以及车库识别的分析与计算过程如下。

## 3.1电机与舵机PWM波配置

为了驱动电机以及舵机转动需要配置PWM波，该系统使用STM32F407的定时器2以及定时器3调制PWM波，调整PWM波占空比控制电机转速以及舵机旋转角度。定时器3和定时器2主频为84MHZ，PWM周期=(psc+1)\*（arr+1）/主频，其中，psc为定时器计数值，arr为分频系数。定时器2的psc为19999，arr为83，因此，其PWM波周期为0.02s；定时器3的psc为9999，arr为83，因此其PWM波周期为0.01s。

## 3.2 OpenMV检测车库方法

为了精确检测到车库位置，对摄像头采集到的视频进行实时的色块检测。对视频图像进行二值化处理并设定一个30×140的ROI区域，小车前进过程中不断检测并统计该区域中黑色像素点个数，当像素数达到设定阈值即表示检测到了车库交接处，通过IO口发送高电平信号给单片机，完成车库检测。车库检测方法示意图，如图3.1所示。



图3.1车库检测示意图

# 4.电路与程序设计

## 4.1系统总体设计

本系统以STM32F407作为核心处理器，主要由OpenMV模块、循迹检测模块、蜂鸣器模块、直流电机、舵机等部分构成。该系统总体设计图以及系统设计清单如下图所示：



图4.1系统总体设计图

表4.1系统设计元件清单

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 元件 | 数量 | 元件 | 数量 | 元件 | 数量 |
| 四位独立按键 | 1 | 星瞳OpenMV | 1 | 有源蜂鸣器 | 1 |
| 电源 | 1 | 红外发射传感器 | 2 | STM32F4芯片 | 1 |
| 稳压模块 | 1 | 编码器 | 1 | 单片机最小系统板 | 1 |
| 直流电机 | 1 | 舵机 | 1 |  |  |

## 4.2单元电路设计

### 4.2.1单片机最小系统设计

微处理器选择STM32F407单片机，其运算速度非常快，具有丰富的外设接口，适合电动车的运行，其主要性能指标如下：

1. 工作频率：168MHZ
2. 时钟配置： STM32F407的SYSCLK最高频率是168MHz；HCLK最高频率为168MHz；内部RC振荡器，可产生16MHz的时钟信号；
3. 14个定时器，包含：高级定时器 TIM1、TIM8 ；通用定时器 TIM2、TIM5、TIM3、TIM4，TIM9-TIM14；基本定时器 TIM6、TIM7 。
4. 6个UART（串口），可灵活地与外部设备全双工数据交换。
5. 丰富的IO口，方便连接外设。

STM32F407单片机核心板电路图如附录所示，其中包括MCU、晶振和复位电路以及外接设备模块连接情况。

### 4.2.2电机驱动模块设计

该驱动模块采用DRV8701E芯片，该芯片具有驱动力强，响应频率高，体积较小等优良性能，同时带有使能控制端。同时控制电机正反转无需两路PWM，只需一路PWM加高低电平即可实现正反转，节省单片机PWM资源。驱动模块电路原理图如图4.2所示。

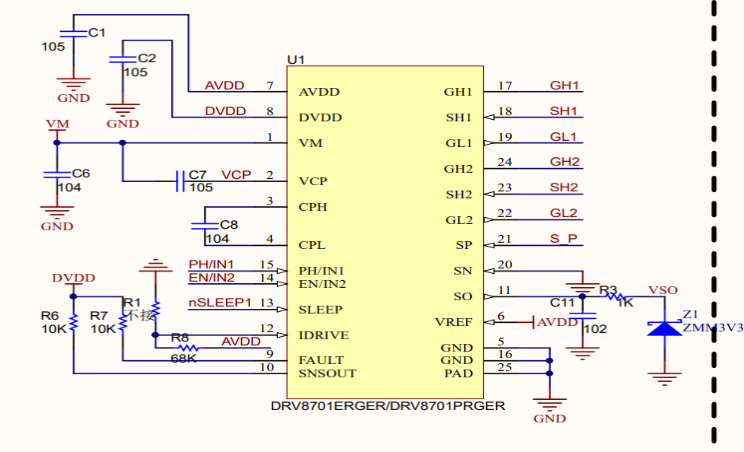


图4.2驱动模块原理图

### 4.2.3独立按键模块设计

四位独立按键一端接地，另一端接单片机IO口，将单片机IO输入上拉，当按键按下时IO口被接地，IO口电平反翻，用于按键扫描检测。该模块原理图如图4.3所示。

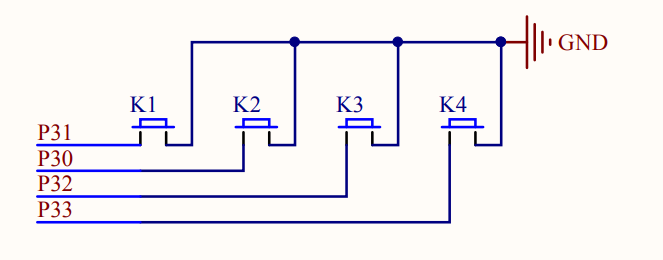


图4.3独立按键模块原理图

### 4.2.4巡线避障模块设计

通过该模块检测车库边缘库线。使用TCRT5000传感器，当检测到黑线时反射回的红外线强度不够大，红外接收管关闭，输出端为高电平，指示灯熄灭。该模块电路原理图如图4.4所示。

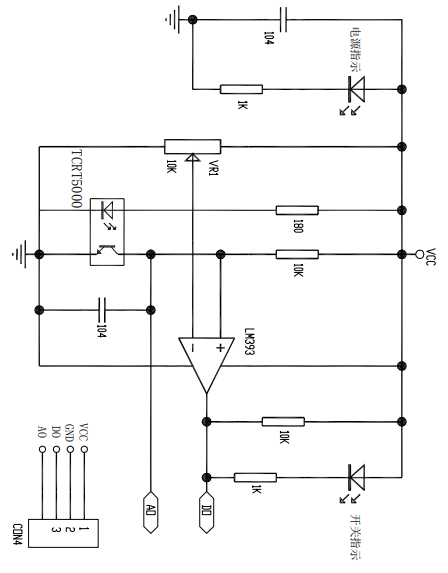


图4.4巡线避障模块原理图

## 4.3系统程序设计



图4.5系统软件总体流程图

# 5.测试方案与测试结果

## 5.1测试仪器

电动车、秒表、比赛赛道。

## 5.2测试方案

### 5.2.1“邻库有车”情况停车测试

将库1、库3、库4、库6分别放入车辆，分三次将电动车分别置于距起点5cm、15cm、30cm处，启动电动车，小车蜂鸣器第一次响起（即开始倒车）时按下秒表，待小车蜂鸣器第二次响起（即倒车完成）停止计时，记录时间。分别观察小车倒车入库以及侧方位入库的完成情况，记录小车停止后与车库边的夹角。

### 5.2.2“邻库无车”情况停车测试

移除库1、库3、库4、库6中的车辆，分三次将电动车分别置于距起点5cm、15cm、30cm处，启动电动车，小车蜂鸣器第一次响起（即开始倒车）时按下秒表，待小车蜂鸣器第二次响起（即倒车完成）停止计时，记录时间。分别观察小车倒车入库以及侧方位入库的完成情况，记录小车停止后与车库边的夹角。

## 5.3测试结果

表5.1电动车“邻库有车”倒车入库情况测试结果

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 小车距起点距离（cm） | 是否完成倒车 | 是否压线 | 倒车所用时间（s） | 小车停止后与库边夹角（**°**） |
| 5 | 是 | 否 | 2.87 | 7 |
| 15 | 是 | 否 | 2.45 | 9 |
| 30 | 是 | 否 | 2.73 | 10 |

表5.2电动车“邻库有车”侧方位入库情况测试结果

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 小车距起点距离（cm） | 是否完成倒车 | 是否压线 | 倒车所用时间（s） | 小车停止后与库边夹角（**°**） |
| 5 | 是 | 否 | 2.25 | 9 |
| 15 | 是 | 否 | 2.46 | 11 |
| 30 | 是 | 否 | 2.15 | 10 |

表5.1电动车“邻库无车”倒车入库情况测试结果

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 小车距起点距离（cm） | 是否完成倒车 | 是否压线 | 倒车所用时间（s） | 小车停止后与库边夹角（**°**） |
| 5 | 是 | 否 | 2.45 | 9 |
| 15 | 是 | 否 | 2.96 | 9 |
| 30 | 是 | 否 | 2.28 | 10 |

表5.2电动车“邻库无车”侧方位入库情况测试结果

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 小车距起点距离（cm） | 是否完成倒车 | 是否压线 | 倒车所用时间（s） | 小车停止后与库边夹角（**°**） |
| 5 | 是 | 否 | 2.64 | 8 |
| 15 | 是 | 否 | 2.27 | 10 |
| 30 | 是 | 否 | 2.84 | 11 |

## 5.4测试结果分析

根据“邻库有车”情况停车测试结果，可以看出电动车在任意位置发车，能够识别到空车库，并完成倒车入库和侧方位入库，车身位置与库边夹角在10°左右；倒车所用时间均在30s以内。

根据“邻库无车”情况停车测试结果，可以看出电动车在任意位置发车，能够识别到目标车库，并完成倒车入库和侧方位入库，车身位置与库边夹角在10°左右；倒车所用时间均在30s以内。

综上，该系统能够完成题目所给任务。

# 6.结论

该系统以正点原子STM32F407VET6作为核心板，由OpenMV模块、循迹检测模块、蜂鸣器模块和前轮转向式四轮车构成。该自动泊车系统的设计采用了模块化思想，完成了色块识别模块及自动泊车模块的设计，达到集中检测车库、自动泊车于一体的目的，实现了自动泊车功能，较好的完成了设计要求。

通过测试方案检验，本系统在邻库有车和邻库无车的情况下，均能完成自动泊车功能。

# 附录1.部分程序代码

## 1.1主程序代码

1. #include "stm32f4xx.h"
2. #include "usart.h"
3. #include "delay.h"
4. #include "sys.h"
5. #include "usart.h"
6. #include "pwm.h"
7. #include "timer.h"*//OPENMV\_OUT() C4*
8. #include "led.h"
9. #include "key.h"*//A1 A2 A3 A4*
10. #include "sensor.h"*////PC1  Sensor1()  PC2   Sensor2()*
11. #include "beer.h"*//C3*
12. #include "speed.h"*//DIR A7  PWM A6*
13. #include "ste.h"*//A0*
14. #include "Encoder.h" *//E0*
15. int Flag\_finish=0;
16. int count\_j=0;
17. int main(void)
18. {
19. LED\_Init();
20. LED\_Close();
21. delay\_init(168);
22. delay\_ms(7000);*//等待openmv完成初始化*
23. TIM5\_Int\_Init(2000,8399);
24. uart\_init(115200);
25. Beer\_Init();
26. speed\_Init();
27. Sensor\_Init();
28. ste\_Init();
29. KEY\_Init();
30. TIM\_ETR\_Init();
31. LED\_Open();*//各初始化完成标志*
32. while(1)
33. {
34. if(KEY\_Num==0)
35. {
36. KEY\_Num=KEY\_Scan(0);
37. count\_j=0;
38. }
39. if(KEY\_Num!=0&&count\_j==0)*//有输入时延时2s*
40. {
41. if(KEY\_Num==1)
42. {
43. LED\_Open();
44. Beer\_Open();
45. delay\_ms(500);
46. Beer\_Close();
47. delay\_ms(1500);
48. count\_j++;
49. }
50. else if(KEY\_Num==2)
51. {
52. Beer\_Close();
53. Beer\_Open();
54. delay\_ms(500);
55. Beer\_Close();
56. delay\_ms(1500);
57. count\_j++;
58. }
60. }
61. *//前进识别*
62. if((KEY\_Num==1||KEY\_Num==2)&&Flag\_back==0)
63. {
64. Forward\_PWM(3000);
65. }
67. if(KEY\_Num==1&&Flag\_back==1)*//倒车入库*
68. {
70. Backward\_PWM(0);
71. delay\_ms(100);
72. STE\_PWM(0);
73. TIM\_SetCounter(TIM4,0);
74. Backward\_PWM(3000);
75. while(TIM\_GetCounter(TIM4)<170);
76. TIM\_SetCounter(TIM4,0);*//向后*
78. STE\_PWM(-300);
79. Backward\_PWM(0);
80. delay\_ms(100);
81. Forward\_PWM(4000);

84. while(TIM\_GetCounter(TIM4)<400);
85. Forward\_PWM(0);
86. STE\_PWM(400);
87. delay\_ms(100);
88. Backward\_PWM(4000);
89. Beer\_Open();
90. delay\_ms(500);
91. Beer\_Close();
92. while(Sensor1()==0);
93. while(sensor2()==0);
94. delay\_ms(200);
95. STE\_PWM(0);
96. TIM\_SetCounter(TIM4,0);
97. while(TIM\_GetCounter(TIM4)<280);
98. Forward\_PWM(0);
99. TIM\_SetCounter(TIM4,0);*//已入库*
101. Beer\_Open();
102. delay\_ms(500);
103. Beer\_Close();
104. delay\_ms(4500);
106. Forward\_PWM(4000);*//出库*
107. *//   while(Sensor1()==0&&sensor2()==0);*
108. TIM\_SetCounter(TIM4,0);
109. while(TIM\_GetCounter(TIM4)<250);
110. STE\_PWM(400);
111. TIM\_SetCounter(TIM4,0);
112. while(TIM\_GetCounter(TIM4)<1050);
113. *//进入完成阶段标志*
114. Flag\_finish=1;
115. }
117. else if(KEY\_Num==2&&Flag\_back==1)*//侧方位*
118. {
119. TIM\_SetCounter(TIM4,0);
120. Forward\_PWM(3000);
121. while(TIM\_GetCounter(TIM4)<400);
122. TIM\_SetCounter(TIM4,0);
123. Forward\_PWM(0);
124. Beer\_Open();
125. STE\_PWM(200);
126. Backward\_PWM(4000);
127. delay\_ms(500);
128. Beer\_Close();
129. while(Sensor1()==0);
130. STE\_PWM(-325);
131. TIM\_SetCounter(TIM4,0);
132. while(TIM\_GetCounter(TIM4)<600);*//已入库*
133. Backward\_PWM(0);
134. Beer\_Open();
135. delay\_ms(500);
136. Beer\_Close();
137. delay\_ms(4500);
139. Forward\_PWM(4000);*//出库*
140. while(sensor2()==0);
141. delay\_ms(400);
142. STE\_PWM(325);
143. TIM\_SetCounter(TIM4,0);
144. while(TIM\_GetCounter(TIM4)<770);
145. *//进入完成阶段标志*
146. Flag\_finish=1;
147. }
149. if(Flag\_finish==1&&Flag\_back==1)*//完成阶段标志*
150. {
151. STE\_PWM(0);
152. TIM5\_Int\_Init(2000,8399);
153. TIM\_SetCounter(TIM4,0);
154. Forward\_PWM(3000);
155. while(TIM\_GetCounter(TIM4)<500);
156. Forward\_PWM(0);
157. TIM\_SetCounter(TIM4,0);
158. Flag\_finish=0;
159. Flag\_back=0;
160. KEY\_Num=0;
162. }
163. }
164. }

## 1.2定时器代码

1. void TIM3\_PWM\_Init(u32 arr,u32 psc)
2. {
3. *//此部分需手动修改IO口设置*
5. GPIO\_InitTypeDef GPIO\_InitStructure;
6. TIM\_TimeBaseInitTypeDef  TIM\_TimeBaseStructure;
7. TIM\_OCInitTypeDef  TIM\_OCInitStructure;
9. RCC\_APB1PeriphClockCmd(RCC\_APB1Periph\_TIM3,ENABLE);   *//TIM3时钟使能*
10. RCC\_AHB1PeriphClockCmd(RCC\_AHB1Periph\_GPIOA, ENABLE);  *//使能PORTA时钟*
12. GPIO\_PinAFConfig(GPIOA,GPIO\_PinSource6,GPIO\_AF\_TIM3); *//GPIOA6复用为定时器3*
14. GPIO\_InitStructure.GPIO\_Pin = GPIO\_Pin\_6;           *//GPIOFA*
15. GPIO\_InitStructure.GPIO\_Mode = GPIO\_Mode\_AF;        *//复用功能*
16. GPIO\_InitStructure.GPIO\_Speed = GPIO\_Speed\_100MHz; *//速度100MHz*
17. GPIO\_InitStructure.GPIO\_OType = GPIO\_OType\_PP;      *//推挽复用输出*
18. GPIO\_InitStructure.GPIO\_PuPd = GPIO\_PuPd\_UP;        *//上拉*
19. GPIO\_Init(GPIOA,&GPIO\_InitStructure);              *//初始化PA6*
21. TIM\_TimeBaseStructure.TIM\_Prescaler=psc;  *//定时器分频*
22. TIM\_TimeBaseStructure.TIM\_CounterMode=TIM\_CounterMode\_Up; *//向上计数模式*
23. TIM\_TimeBaseStructure.TIM\_Period=arr;   *//自动重装载值*
24. TIM\_TimeBaseStructure.TIM\_ClockDivision=TIM\_CKD\_DIV1;
26. TIM\_TimeBaseInit(TIM3,&TIM\_TimeBaseStructure);*//初始化定时器3*
28. *//初始化TIM14 Channel1 PWM模式*
29. TIM\_OCInitStructure.TIM\_OCMode = TIM\_OCMode\_PWM1; *//选择定时器模式:TIM脉冲宽度调制模式2*
30. TIM\_OCInitStructure.TIM\_OutputState = TIM\_OutputState\_Enable; *//比较输出使能*
31. TIM\_OCInitStructure.TIM\_OCPolarity = TIM\_OCPolarity\_High; *//输出极性:TIM输出比较极性低*
32. TIM\_OC1Init(TIM3, &TIM\_OCInitStructure);  *//根据T指定的参数初始化外设TIM1 4OC1*
33. TIM\_OC1PreloadConfig(TIM3, TIM\_OCPreload\_Enable);  *//使能TIM3在CCR1上的预装载寄存器*
35. TIM\_ARRPreloadConfig(TIM3,ENABLE);*//ARPE使能*
37. TIM\_Cmd(TIM3, ENABLE);  *//使能TIM3*

40. }
41. *//A0*
42. void TIM2\_PWM\_Init(u32 arr,u32 psc)
43. {
44. GPIO\_InitTypeDef GPIO\_InitStructure;
45. TIM\_TimeBaseInitTypeDef TIM\_TimeBaseStructure;
46. TIM\_OCInitTypeDef TIM\_OCInitStructure;
48. RCC\_APB1PeriphClockCmd(RCC\_APB1Periph\_TIM2,ENABLE);*//使能定时器时钟*
50. RCC\_AHB1PeriphClockCmd(RCC\_AHB1Periph\_GPIOA, ENABLE);*//使能IO口时钟*
52. GPIO\_PinAFConfig(GPIOA,GPIO\_PinSource0,GPIO\_AF\_TIM2);*//PA0复用为定时器2*

55. GPIO\_InitStructure.GPIO\_Pin = GPIO\_Pin\_0;
56. GPIO\_InitStructure.GPIO\_Mode = GPIO\_Mode\_AF;        *//复用功能*
57. GPIO\_InitStructure.GPIO\_Speed = GPIO\_Speed\_100MHz; *//速度100MHz*
58. GPIO\_InitStructure.GPIO\_OType = GPIO\_OType\_PP;      *//推挽复用输出*
59. GPIO\_InitStructure.GPIO\_PuPd = GPIO\_PuPd\_UP;        *//上拉*
60. GPIO\_Init(GPIOA,&GPIO\_InitStructure);               *//初始化PA0*

63. TIM\_TimeBaseStructure.TIM\_Prescaler=psc;                  *//定时器分频*
64. TIM\_TimeBaseStructure.TIM\_CounterMode=TIM\_CounterMode\_Up;*//向上计数模式*
65. TIM\_TimeBaseStructure.TIM\_Period=arr;                  *//自动重装载值*
66. TIM\_TimeBaseStructure.TIM\_ClockDivision=TIM\_CKD\_DIV1;
67. TIM\_TimeBaseInit(TIM2,&TIM\_TimeBaseStructure);           *//初始化定时器*

70. TIM\_OCInitStructure.TIM\_OCMode = TIM\_OCMode\_PWM1;            *//选择PWM模式*
71. TIM\_OCInitStructure.TIM\_OutputState = TIM\_OutputState\_Enable;*//比较输出使能*
72. TIM\_OCInitStructure.TIM\_OCPolarity = TIM\_OCPolarity\_High;    *//输出极性低*
73. TIM\_OC1Init(TIM2, &TIM\_OCInitStructure);                     *//初始化定时器通道*
75. */\*定时器主模式设置\*/*
76. TIM\_SelectMasterSlaveMode(TIM2, TIM\_MasterSlaveMode\_Enable);
77. TIM\_SelectOutputTrigger(TIM2, TIM\_TRGOSource\_OC1Ref);
79. TIM\_Cmd(TIM2, ENABLE);
81. TIM\_OC1PreloadConfig(TIM2, TIM\_OCPreload\_Enable);*//使能定时器在ccr上的预装载寄存器*
83. TIM\_ARRPreloadConfig(TIM2,ENABLE);*//ARPE使能*
84. TIM\_SetCompare1(TIM2,1000);
85. }

## 1.3电机控制代码

1. void speed\_Init(void)
2. {
3. GPIO\_InitTypeDef  GPIO\_InitStructure;
4. RCC\_AHB1PeriphClockCmd(RCC\_AHB1Periph\_GPIOA, ENABLE);*//使能GPIOC时钟*
5. GPIO\_InitStructure.GPIO\_Pin = GPIO\_Pin\_7;
6. GPIO\_InitStructure.GPIO\_Mode = GPIO\_Mode\_OUT;
7. GPIO\_InitStructure.GPIO\_OType =GPIO\_OType\_PP;
8. GPIO\_InitStructure.GPIO\_Speed = GPIO\_Speed\_100MHz;*//100M*
9. GPIO\_InitStructure.GPIO\_PuPd =  GPIO\_PuPd\_NOPULL;*//*
10. GPIO\_Init(GPIOA, &GPIO\_InitStructure);*//初始化*
11. TIM3\_PWM\_Init(9999,83);
12. Forward\_PWM(0);
13. }
14. void Forward\_PWM(u16 PWM)
15. {
16. GPIO\_ResetBits(GPIOA,GPIO\_Pin\_7);
17. TIM\_SetCompare1(TIM3,PWM);
19. }
20. void Backward\_PWM(u16 PWM)
21. {
22. GPIO\_SetBits(GPIOA,GPIO\_Pin\_7);
23. TIM\_SetCompare1(TIM3,PWM);
25. }

## 1.4编码器代码

1. void TIM\_ETR\_Init(void)
2. {
3. TIM\_TimeBaseInitTypeDef TIM\_TimeBaseStructure;
4. GPIO\_InitTypeDef GPIO\_InitStructure;
5. RCC\_APB1PeriphClockCmd(RCC\_APB1Periph\_TIM4,ENABLE);
6. RCC\_AHB1PeriphClockCmd(RCC\_AHB1Periph\_GPIOE, ENABLE);
7. GPIO\_InitStructure.GPIO\_Pin = GPIO\_Pin\_0;
8. GPIO\_InitStructure.GPIO\_Mode = GPIO\_Mode\_AF;
9. GPIO\_InitStructure.GPIO\_PuPd = GPIO\_PuPd\_UP;
10. GPIO\_InitStructure.GPIO\_Speed = GPIO\_Speed\_100MHz;
11. GPIO\_Init(GPIOE,&GPIO\_InitStructure);
12. GPIO\_PinAFConfig(GPIOE,GPIO\_PinSource0,GPIO\_AF\_TIM4);   *//开启第二功能*
13. TIM\_TimeBaseStructure.TIM\_Period = 0xFFFF;
14. TIM\_TimeBaseStructure.TIM\_Prescaler = 0;
15. TIM\_TimeBaseStructure.TIM\_ClockDivision = TIM\_CKD\_DIV1;
16. TIM\_TimeBaseStructure.TIM\_CounterMode = TIM\_CounterMode\_Up;
17. TIM\_TimeBaseInit(TIM4, &TIM\_TimeBaseStructure);
18. TIM\_ETRClockMode2Config(TIM4,TIM\_ExtTRGPSC\_OFF,TIM\_ExtTRGPolarity\_NonInverted,0);  *//外部时钟源模式2*
19. TIM\_Cmd(TIM4, ENABLE);
20. TIM\_GetCounter(TIM4);
21. TIM\_SetCounter(TIM4,0);
22. }

## 1.5舵机控制代码

1. void ste\_Init(void)
2. {
3. TIM2\_PWM\_Init(19999,83);
4. TIM\_SetCompare1(TIM2,STE\_Median\_PWM);
5. }
6. void STE\_PWM(int PWM)
7. {
8. if(PWM<0)
9. {
10. PWM=PWM\*1.62;
11. }
12. if(PWM>400)
13. {
14. PWM=400;
15. }
16. else if(PWM<-648)
17. {
18. PWM=-648;
19. }
20. TIM\_SetCompare1(TIM2,PWM+STE\_Median\_PWM);
21. }

## 1.6色块识别代码

1. *# Single Color Grayscale Blob Tracking Example*
2. *#*
3. *# This example shows off single color grayscale tracking using the OpenMV Cam.*
4. import sensor, image, time, math
5. import pyb
6. from pyb import Pin
7. *# Color Tracking Thresholds (Grayscale Min, Grayscale Max)*
8. *# The below grayscale threshold is set to only find extremely bright white areas.*
9. thresholds = (0, 74)
10. sensor.reset()
11. sensor.set\_pixformat(sensor.GRAYSCALE)
12. sensor.set\_framesize(sensor.QVGA)
13. sensor.skip\_frames(time = 2000)
14. sensor.set\_auto\_gain(False) # must be turned off for color tracking
15. sensor.set\_auto\_whitebal(False) # must be turned off for color tracking
16. sensor.set\_vflip(1)
17. sensor.set\_hmirror(1)
18. clock = time.clock()
19. PIN1=Pin('P0',Pin.OUT\_PP)
20. PIN1.low()
21. led = pyb.LED(3)
22. led1 = pyb.LED(1)
23. led.off()
24. count\_i=0
25. count\_k=0
26. count\_j=0
27. *# Only blobs that with more pixels than "pixel\_threshold" and more area than "area\_threshold" are*
28. *# returned by "find\_blobs" below. Change "pixels\_threshold" and "area\_threshold" if you change the*
29. *# camera resolution. "merge=True" merges all overlapping blobs in the image.*
30. while(True):
31. clock.tick()
32. img = sensor.snapshot()
33. blobs = img.find\_blobs([thresholds], roi= [145, 120, 30, 140],pixels\_threshold=100, area\_threshold=100, merge=True)
34. for blob in blobs:
35. count\_k=count\_k+1
36. if count\_k<2:
37. count\_j=1
38. count\_k=0
39. else:
40. count\_k=0
41. count\_j=0
42. for blob in blobs:
43. *# These values depend on the blob not being circular - otherwise they will be shaky.*
44. if blob.elongation() > 0.5:
45. img.draw\_edges(blob.min\_corners(), color=0)
46. img.draw\_line(blob.major\_axis\_line(), color=0)
47. img.draw\_line(blob.minor\_axis\_line(), color=0)
48. *# These values are stable all the time.*
49. img.draw\_rectangle(blob.rect(), color=127)
50. img.draw\_cross(blob.cx(), blob.cy(), color=127)
51. *#print(blob.pixels())*
52. print(count\_j)
53. if blob.pixels()>1000 and count\_j==1:
54. count\_i=count\_i+1
55. led.on()
56. if count\_i<=2:
57. time.sleep\_ms(500)
58. led.off()
59. if count\_i==3:
60. count\_i=0
61. PIN1.high()
62. led1.on()
63. time.sleep\_ms(1000)
64. PIN1.low()
65. led1.off()
67. #if blob.pixels()/last\_num>2:
68. *#print(2)*
69. *#print(blob.pixels())*

# 附录2.电路连接原理

# 附录3.实验结果照片

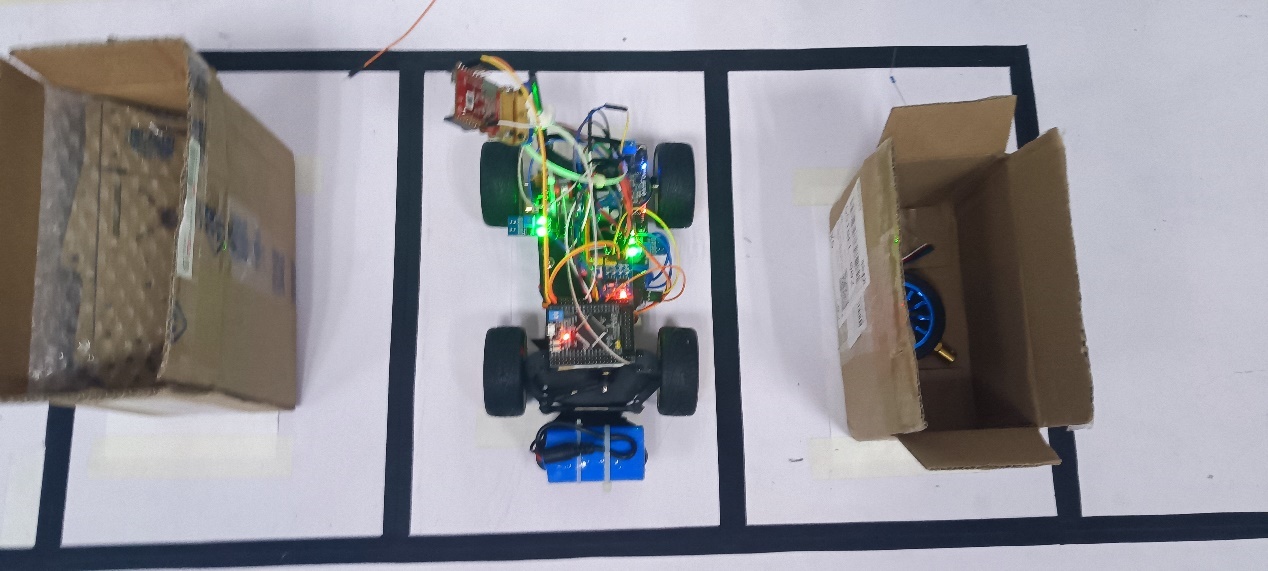


图1.邻库有车倒车入库测试结果图

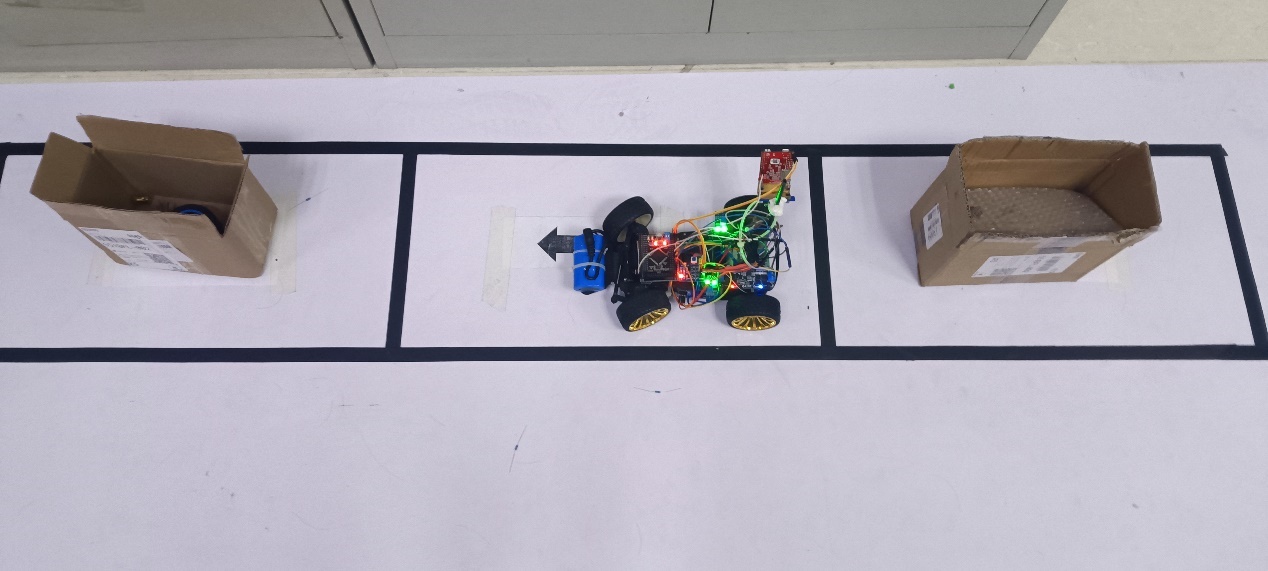


图2.邻库有车侧方位停车测试结果图

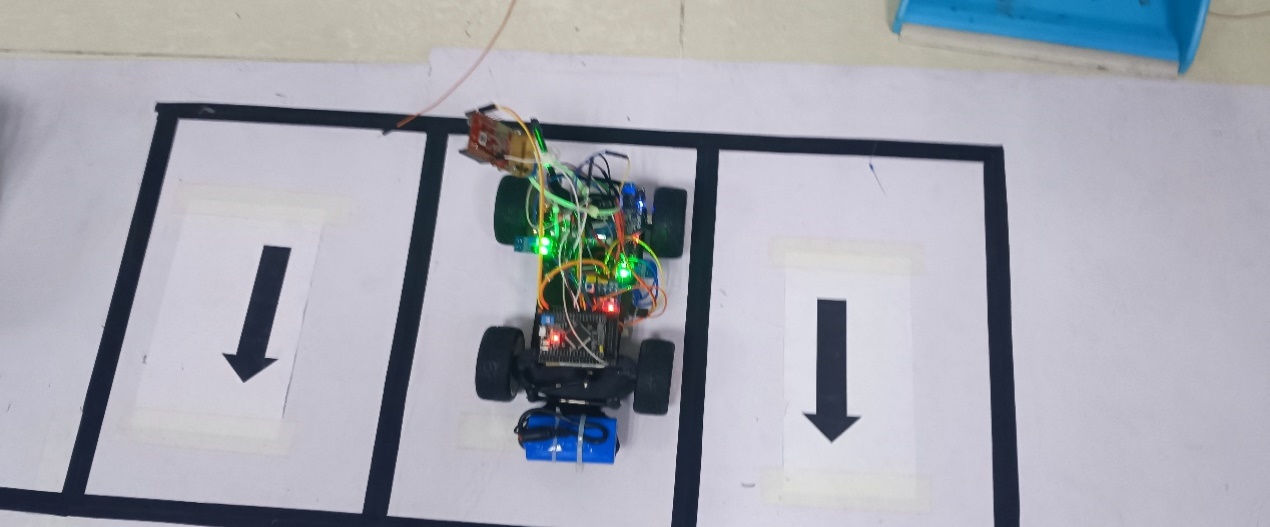


图3.邻库无车倒车入库测试结果图

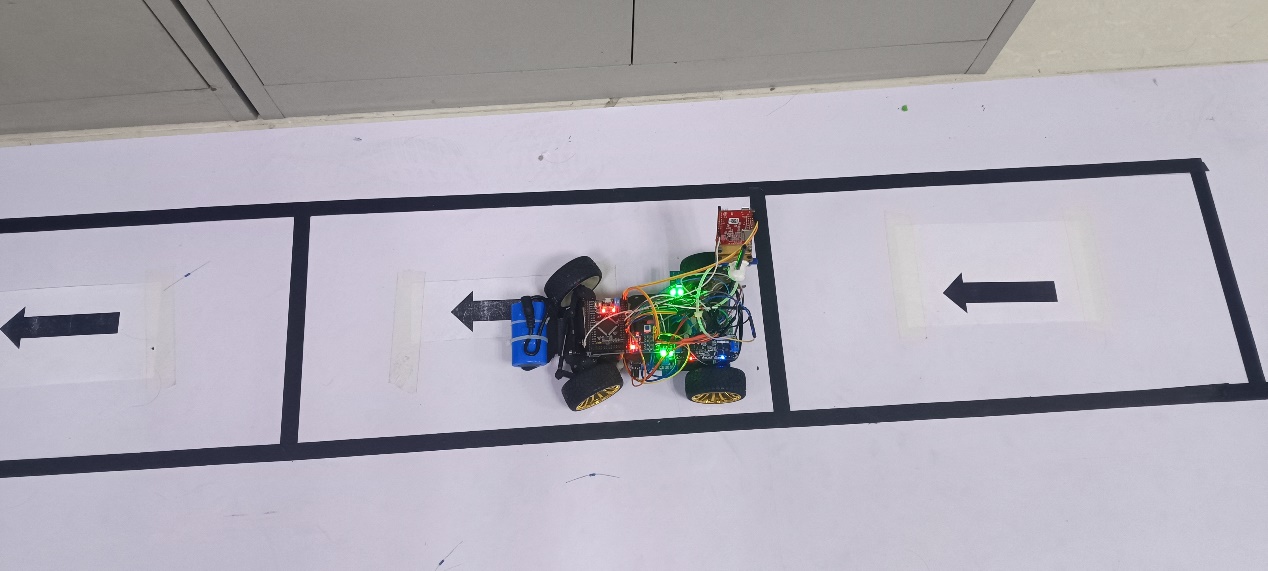


图4.邻库无车侧方位停车测试结果图