**密封纸页（需写对应参赛队编码）**

**报告正文长度严格限制为A4纸8页以内，首页另附300字以内的设计报告中文摘要，正文采用小四号宋体字，行距固定值22磅，标题字号自定，纵向打印。设计报告包括：密封纸页（需写对应参赛队编码）、封面页、摘要页、正文。《设计报告》每页上方必须留出3cm以上空白，空白区域内不得有任何文字，每页右下端注明页码。**

**2022“数智中原”河南省大学生电子设计大赛**

**（自动绕障行走小车\_I题）**

**【本科组】**



**2022年11月9日**

自动绕障行走小车（I题）

摘 要

本系统以arduino UNO 板芯微控制器为主控芯片，三轮小车套件为模型，通过机器视觉模块OpenMV获取与障碍物的表面图像，通过图像信息处理得到两者的距离与偏转信息发送给单片机，使小车能够在合适的位置进行转弯避障操作。

关键词：arduino UNO；机器视觉模块；三轮小车套件；

Abstract

The system takes the arduino UNO board core microcontroller as the main control chip and the three-wheeled car firmware as the model. It obtains the surface image with the obstacles through the machine vision module OpenMV. Through the image information processing, the distance and deflection information of the two are obtained and sent to the MCU, so that the car can turn and avoid the obstacle operation in the appropriate position.

Key words:arduino UNO;Machine vision module;Three-wheeled trolley firmware;

目录

[自动绕障行走小车（I题） 2](#_Toc121945016)

[1系统方案 1](#_Toc121945017)

[1.1系统总体方案设计 1](#_Toc121945018)

[**1.2系统各模块的选择与比较** 1](#_Toc121945019)

[1.2.1控制模块的选择 1](#_Toc121945020)

[1.2.2电机驱动模块的选择 1](#_Toc121945021)

[1.2.3视觉模块的选择 1](#_Toc121945022)

[1.2.4稳压模块的选择 1](#_Toc121945023)

[1.2.5电源模块的选择 1](#_Toc121945024)

[2系统理论分析与计算 1](#_Toc121945025)

[2.1电路分析 1](#_Toc121945026)

[2.2距离计算 1](#_Toc121945027)

[3软件程序设计 1](#_Toc121945028)

[4测试方案与测试结果 1](#_Toc121945029)

[4.1测试方案 1](#_Toc121945030)

[4.1.1检查电路 1](#_Toc121945031)

[4.1.2通电观察 1](#_Toc121945032)

[4.1.3观察波形与数据 1](#_Toc121945033)

[4.1.4实验条件和器具 1](#_Toc121945034)

[4.2 测试结果 1](#_Toc121945035)

[4.3分析与结论 1](#_Toc121945036)

[附录1：源程序 1](#_Toc121945037)

[openmv检测部分源代码： 1](#_Toc121945038)

[主控部分源程序： 1](#_Toc121945039)

# 1系统方案

## 1.1系统总体方案设计

根据设计任务要求，可以将该系统分为四个部分，第一个部分为单片机控制模块，由arduino UNO R3单片机、单片机最小系统、串口通信电路和相应辅助电路组成，并主要负责各个子模块的控制工作，从而保证各自模块正常有序的完成各自任务；第二个部分为电机驱动模块，主要由tb6612fnc模块组成，它的主要作用是驱动小车直流电机，使小车运动；第三个部分为视觉模块，它主要用于对小车运行前方障碍物进行探测和对障碍物距离进行判断；第四个部分为稳压模块，主要作用是稳定降低电源电压，使小车的供电稳定。

**1.2系统各模块的选择与比较**

### 1.2.1控制模块的选择

第一种方案：STM32 性能较高，对于计算或控制要求较高可以优先考虑，但是开发时间较长，且对于本项目性能有所富裕，成本较高。

第二种方案：Arduino UNO代码的封装比较完整，开发更加快速，由于两种方案对于本项目性能都足够，本着开发时间短，成本低的原则，我们选择Arduino UNO 作为本项目的主要控制模块。

### 1.2.2电机驱动模块的选择

综合市面上的驱动模块，和我们的实际情况。我们选用TB6612FNG这款驱动模块，该模块每通道能输出最高1A的连续驱动电流，启动峰值电流达2A／3A(连续脉冲／单脉冲)，有着4种电机控制模式：正转／反转／制动／停止，PWM支持频率高达100 kHz；

Tb6612fnc驱动模块能够同时输出两路pwm波，可以同时驱动两个电机，而且可以配合减速电机上面的霍尔编码器读取电机的转速以实现闭环控制。满足该项目的需求。

### 1.2.3视觉模块的选择

第一种方案：使用K210 作为视觉识别模块，在算力上K210更大，但K210 的 IDE（MaixPy IDE）的优化相对于 OpenMV IDE 较差，IDE 也很久没有人维护了。尽管 OpenMV 和 K210 都采用 Python语言编程，但有些 OpenMV 上较为重要的函数在 K210 上不能使用，并且K210目前主要被用来作为ai应用的硬件，与本项目的需求有些冲突。

第二种方案：OpenMV的图像更加清楚，支持的视频格式更多，最高可支持SXGA，而K210 最高支持VGA，在本项目的需求方面，OpenMV更强，且OpenMV与arduino的串口通讯更方便。因此我们选择OpenMV作为视觉模块使用。

### 1.2.4稳压模块的选择

稳压模块选择了LM2596S DC-DC可调降压模块，可提供 3.3V、5V、 12V 固定输出电压和可调节输出电压版本，足够为本项目中控制模块与电机驱动模块提供稳定的电压。

### 1.2.5电源模块的选择

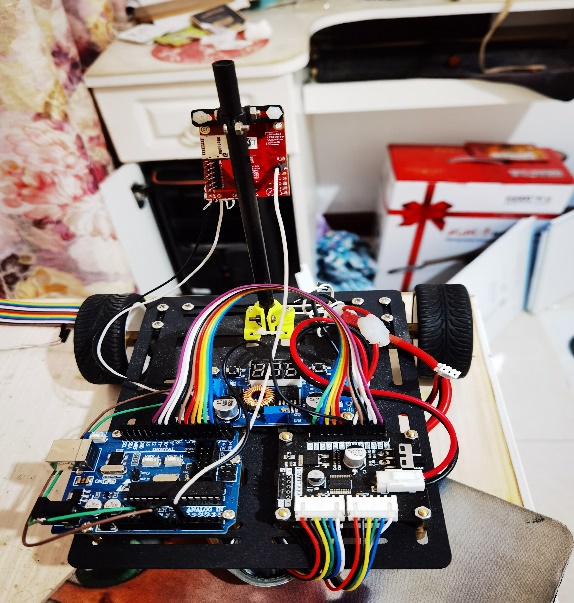
本项目的电机和驱动模块需要7v到12v的电压来进行驱动，综合市场和我们自身的经验，选择了输出电流能力更强的2s锂电池作为本项目的电源。

# 2系统理论分析与计算

## 2.1电路分析

本作品通过2s锂电池给各个模块进行供电，首先将电池连接在tb6612fnc电机驱动上驱动电机运行，同时通过驱动模块将电压降到5v给openmv和主控模块进行供电。这样可以保证系统中每个模块的正常运行，同时保证主控模块和openmv不受到剧烈的电流冲击导致系统出现异常。

通信电路主要将openmv和主控芯片连接起来，将两个模块对应的串口引脚用杜邦线连接保证两个模块之间的正常通讯。



主要电路连接图



Openmv部分电路图

## 2.2距离计算

我们通过将摄像头安装到支架上保证摄像头的视野。通过不断检测路障在图像中的大小与预设的值相比较，就可以控制小车相对于路障的距离。



计算距离部分代码

K为预设值，cw和ch为检测到的图像中的路障的参数，通过这两个之间的换算就可以让小车在离路障有一定距离时停下。

# 3软件程序设计

## 3.1主控程序

## 3.2openmv程序

Openmv先通过预设的阈值检测拍到的图片中路障和停止标志的色块大小，经过滤波选出最大的那个色块作为路障或是停止色块，获得该色块的参数，最后判断该色块是停止标志还是路障。返回对应的参数给主控模块。

# 4测试方案与测试结果

## 4.1测试方案

### 4.1.1检查电路

我们在所有电路通电调试之前仔细的检查核对电路，看是否有连线错误的的部分。最重要的是正负极有没有接反，短接，然后是元器件的正负极，还有元器件的引脚有没有焊实。

### 4.1.2通电观察

检查电路无误后，首先要做的不是用仪器观测波形和数据，二是先观察整个电路是否有冒烟、异常气味、发光放电、元器件发烫等现象。如果有，先断电， 继续检查电路，找出故障。再重新通电。

### 4.1.3观察波形与数据

通电观察无异常后，用电压表和示波器测试输入信号和输出信号是否和期望， 代码编写的结果一致。

### 4.1.4实验条件和器具

我们根据题目要求搭建符合标准的场地进行测试，测试工具主要有万用表，示波器，串口转usb模块，电脑，蓝牙模块等。

## 4.2 测试结果

|  |  |  |  |
| --- | --- | --- | --- |
| 次数 | 完成时间/s | 碰撞障碍物次数 | 中心标志距离障碍物的距离超过 40cm次数 |
| 测试1 | 39 | 0 | 0 |
| 测试2 | 38 | 0 | 0 |
| 测试3 | 30 | 0 | 0 |
| 测试4 | 24 | 0 | 0 |
| 测试5 | 23 | 0 | 0 |
| 测试6 | 28 | 0 | 0 |
| 测试7 | 27 | 0 | 0 |
| 测试8 | 25 | 0 | 0 |
| 测试9 | 26 | 0 | 0 |

## 4.3分析与结论

通过不断的调节pwm的输出控制电机的速度和摄像头模块的检测阈值，改变转弯的半径、转弯的角度，优化减少碰撞障碍物次数和中心标志距离障碍物的距离，加快完成速度。通过测试，本作品可以顺利完成题目要求。

# 附录1：源程序

## openmv检测部分源代码：

from sys import flags

import sensor, image, time,math,pyb

from pyb import UART,LED ###串口要用

import json

import ustruct ###串口要用

# 为了使色彩追踪效果真的很好，你应该在一个非常受控制的照明环境中。

uart = UART(1,9600) # 定义串口

def find\_max(blobs): #定义寻找色块面积最大的函数 ###串口要用

max\_size=0

for blob in blobs:

if blob.pixels() > max\_size:

max\_blob=blob

max\_size = blob.pixels()

return max\_blob

green\_threshold = (60, 39, 40, 82, 73, 42)

black\_threshold = (10, 20, -20, 13, -20, 20)

sensor.reset() # 初始化sensor

sensor.set\_pixformat(sensor.RGB565) # use RGB565.

#设置图像色彩格式，有RGB565色彩图和GRAYSCALE灰度图两种

sensor.set\_framesize(sensor.QVGA) # 使用QQVGA的速度。

#设置图像像素大小

sensor.skip\_frames(10) # 让新的设置生效。

sensor.set\_auto\_whitebal(False) # turn this off.

#关闭白平衡。白平衡是默认开启的，在颜色识别中，需要关闭白平衡。

clock = time.clock() # 跟踪FPS帧率

K=1000#the value should be measured

a\_1=a\_2=0.

flaga=0;

flagb=0;

while(True):

clock.tick() # 追踪两个snapshots()之间经过的毫秒数.

img = sensor.snapshot() # 拍一张照片并返回图像。

cx=0;cy=0;

blobs = img.find\_blobs([green\_threshold])

blobsb = img.find\_blobs([black\_threshold])

#find\_blobs(thresholds, invert=False, roi=Auto),thresholds为颜色阈值，

#是一个元组，需要用括号［ ］括起来。invert=1,反转颜色阈值，invert=False默认

#不反转。roi设置颜色识别的视野区域，roi是一个元组， roi = (x, y, w, h)，代表

#从左上顶点(x,y)开始的宽为w高为h的矩形区域，roi不设置的话默认为整个图像视野。

#这个函数返回一个列表，[0]代表识别到的目标颜色区域左上顶点的x坐标，［1］代表

#左上顶点y坐标，［2］代表目标区域的宽，［3］代表目标区域的高，［4］代表目标

#区域像素点的个数，［5］代表目标区域的中心点x坐标，［6］代表目标区域中心点y坐标，

#［7］代表目标颜色区域的旋转角度（是弧度值，浮点型，列表其他元素是整型），

#［8］代表与此目标区域交叉的目标个数，［9］代表颜色的编号（它可以用来分辨这个

#区域是用哪个颜色阈值threshold识别出来的）。

if blobs:

#如果找到了目标颜色

max\_b = find\_max(blobs);

# Draw a rect around the blob.

img.draw\_rectangle(max\_b[0:4]) # rect

#用矩形标记出目标颜色区域

img.draw\_cross(max\_b[5], max\_b[6]) # cx, cy

#img.draw\_cross(160, 120) # 在中心点画标记

#在目标颜色区域的中心画十字形标记

cx=max\_b[5];

cy=max\_b[6];

cw=max\_b[2];

ch=max\_b[3];

flaga=1;

if blobsb:

#如果找到了目标颜色

max\_b1 = find\_max(blobsb);

# Draw a rect around the blob.

img.draw\_rectangle(max\_b1[0:4]) # rect

#用矩形标记出目标颜色区域

img.draw\_cross(max\_b1[5], max\_b1[6]) # cx, cy

#img.draw\_cross(160, 120) # 在中心点画标记

#在目标颜色区域的中心画十字形标记

cx1=max\_b1[5];

cy1=max\_b1[6];

cw1=max\_b1[2];

ch1=max\_b1[3];

flagb=1;

if flaga==1 and flagb!=1:

a\_1=cx-160

Lm = (cw+ch)/2

length = K/Lm

data={

"length":length,

"a\_1":a\_1

}

data\_out=json.dumps(data)

uart.write(data\_out); #必须要传入一个字节数组

print("距离：",length)

print("偏移量：",a\_1)

img.draw\_string(0,0, "length:", color=(0,255,0), scale = 2)

img.draw\_string(0,20, "Offset:", color=(0,255,0), scale = 2)

img.draw\_string(120,0, str(length), color=(255,0,255), scale = 2)

img.draw\_string(110,20, str(a\_1), color=(255,0,255), scale = 2)

img.draw\_cross(160, 120, size=5, color=(0,255,0))

img.draw\_line((160, 120,cx1, cy1), color=(255,0,255))

flaga=0;

flagb=0;

elif flaga!=1 and flagb==1:

a\_1=cx1-160

Lm = (cw1+ch1)/2

length = K/Lm

data={

"length":length,

"a\_1":a\_1

}

data\_out=json.dumps(data)

uart.write(data\_out); #必须要传入一个字节数组

print("距离：",length)

print("偏移量：",a\_1)

img.draw\_string(0,0, "length:", color=(0,255,0), scale = 2)

img.draw\_string(0,20, "Offset:", color=(0,255,0), scale = 2)

img.draw\_string(120,0, str(length), color=(255,0,255), scale = 2)

img.draw\_string(110,20, str(a\_1), color=(255,0,255), scale = 2)

img.draw\_cross(160, 120, size=5, color=(0,255,0))

img.draw\_line((160, 120,cx1, cy1), color=(255,0,255))

flaga=0;

flagb=0;

elif flaga==1 and flagb==1:

if max\_b.pixels()>max\_b1.pixels():

a\_1=cx-160

Lm = (cw+ch)/2

length = K/Lm

data={

"length":length,

"a\_1":a\_1

}

data\_out=json.dumps(data)

uart.write(data\_out); #必须要传入一个字节数组

print("距离：",length)

print("偏移量：",a\_1)

img.draw\_string(0,0, "length:", color=(0,255,0), scale = 2)

img.draw\_string(0,20, "Offset:", color=(0,255,0), scale = 2)

img.draw\_string(120,0, str(length), color=(255,0,255), scale = 2)

img.draw\_string(110,20, str(a\_1), color=(255,0,255), scale = 2)

img.draw\_cross(160, 120, size=5, color=(0,255,0))

img.draw\_line((160, 120,cx1, cy1), color=(255,0,255))

flaga=0;

flagb=0;

else:

a\_1=cx1-160

Lm = (cw1+ch1)/2

length = K/Lm

data={

"length":length,

"a\_1":a\_1

}

data\_out=json.dumps(data)

uart.write(data\_out); #必须要传入一个字节数组

print("距离：",length)

print("偏移量：",a\_1)

img.draw\_string(0,0, "length:", color=(0,255,0), scale = 2)

img.draw\_string(0,20, "Offset:", color=(0,255,0), scale = 2)

img.draw\_string(120,0, str(length), color=(255,0,255), scale = 2)

img.draw\_string(110,20, str(a\_1), color=(255,0,255), scale = 2)

img.draw\_cross(160, 120, size=5, color=(0,255,0))

img.draw\_line((160, 120,cx1, cy1), color=(255,0,255))

flaga=0;

flagb=0;

## 主控部分源程序：