

微处理器原理与应用

实验报告

|  |  |
| --- | --- |
| **姓名:** | 陶桢 |
| **专业(方向):** | 通信工程 |
| **学号:** | 9191040G04G0428 |
| **指导老师:** | 朱红 |

# 微处理器原理与应用综合实验

# 1 实验要求说明

## 1.1 考核说明

分三步进行考核

1.检测开关性能和LED灯的性能

1）点亮LED灯，可以是跑马灯的形式闪烁

2）能够用两个开关分别控制LED灯，控制点亮的时间间隔或灯的颜色

2.测试轮子转动的功能，小车可以开动，最好速度方向可以由开关调节

3.测试循线功能、避障功能和碰撞转向功能，三个功能根据完成情况原则上分为中、良、优，即完成一个为中，两个为良、全部完成为优。同时最终成绩要结合前两次的测评和实验报告和视频质量综合给出。

# 2 小车硬件和软件原理

## 2.1 小车硬件

本次实验为设计智能小车的控制系统，使用的是TI公司出品的TI-RSLK套件。

MSP432是TI公司于2015年3月推出的嵌入式单片机，内核使用32位的Cortex-M4F内核，具有32位数据总线，32位地址总线和32位存储器接口。实验使用的单片机型号是MSP432P401R。

MSP432单片机采用哈佛结构。与冯·诺依曼结构处理器相比，哈佛结构处理器有两个明显的特点：使用两个独立的存储器模块，分别存储指令和数据，每个存储模块都不允许指令和数据并存；使用两条独立的总线，分别作为CPU与每个存储器之间的专用通信路径，而这两条总线之间毫无关联。

## 2.2 软件原理

本次实验使用到的软件包括Code　composer　studio（CCS）和Ｖisual　studio　code（VS code），其中CCS可以将我们的程序烧录进开发板并进行调试，VS code可以大幅度提高代码编写速度，提高开发效率。

### 2.2.1 Code composer studio

Code Composer Studio 包含一整套用于开发和调试嵌入式应用的工具。它包含适用于每个 TI 器件系列的编译器、源码编辑器、项目构建环境、调试器、描述器、仿真器以及多种其它功能。CCS IDE 提供了单个用户界面，可帮助您完成应用开发流程的每个步骤。借助于精密的高效工具，用户能够利用熟悉的工具和界面快速上手并将功能添加至他们的应用。

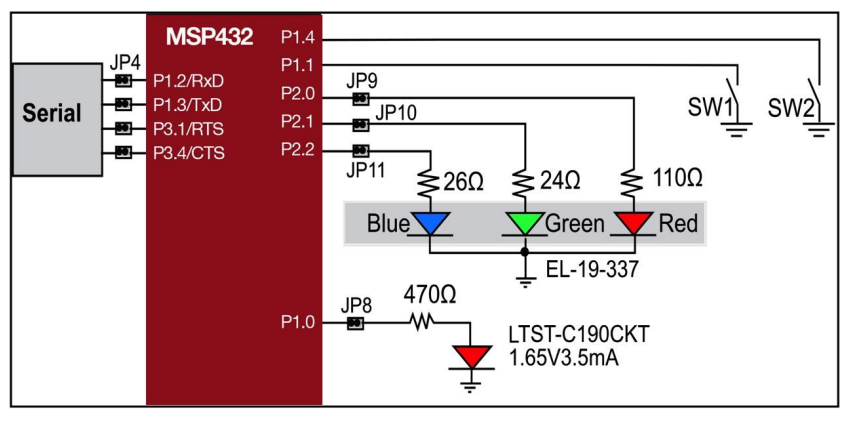
### 2.2.1 Visual studio code

Visual Studio Code（简称“VS Code”）是Microsoft在2015年4月30日Build开发者大会上正式宣布一个运行于 Mac OS X、Windows和 Linux 之上的，针对于编写现代Web和云应用的跨平台源代码编辑器，可在桌面上运行，并且可用于Windows，macOS和Linux。它具有对JavaScript，TypeScript和Node.js的内置支持，并具有丰富的其他语言（例如C++，C＃，Java，Python，PHP，Go）和运行时（例如.NET和Unity）扩展的生态系统。

# 3 实验原理及模块说明

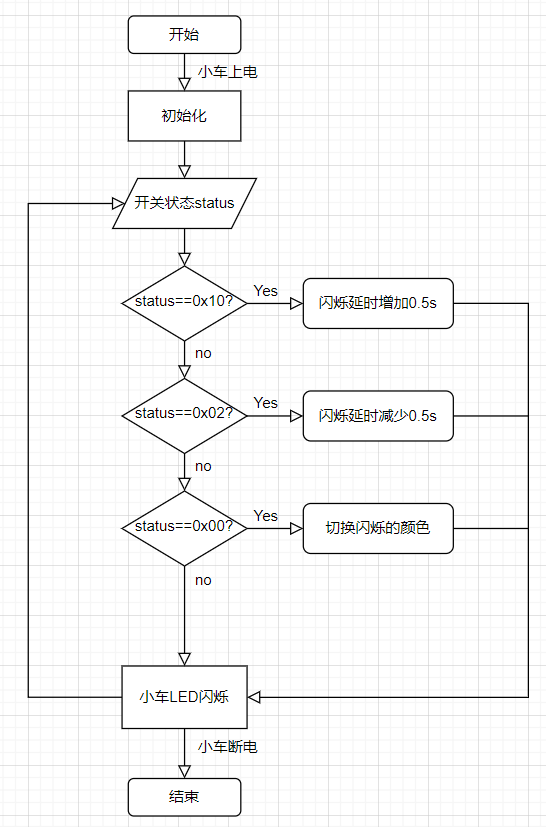
## 3.1 实验一

### 3.1.1 实验原理



初始化P2寄存器后，只要将P2的输出寄存器配置为相应的高电平就可以控制三色LED灯的颜色。同时，初始化P1寄存器后，当开关SW1或者SW2按下的时候，相应的P1.1和P1.4变成低电平，可以通过读取P1寄存器的状态来判断开关的通断情况。

### 3.1.2 实验流程图



### 3.1.3 程序清单

#include "msp.h"

#include "Clock.h"

#include<stdio.h>

//宏定义颜色和开关

#define SW1 0x10

#define SW2 0x02

#define DARK 0

#define RED 0x01

#define GREEN 0x02

#define BLUE 0x04

#define YELLOW 0x03

#define SKY\_BLUE 0x06

#define WHITE 0x07

#define PINK 0x05

//初始化P1

void Port1\_Init**(**void**){**

P1**->**SEL0 **=** 0x00**;**

P1**->**SEL1 **=** 0x00**;**

P1**->**DIR **=** 0x01**;**

P1**->**REN **=** 0x12**;**

P1**->**OUT **=** 0x12**;**

**}**

//输入P1

uint8\_t Port1\_Input**(**void**){**

**return** **(**P1**->**IN**&**0x12**);**

**}**

//初始化P2

void Port2\_Init**(**void**)**

**{**

P2**->**SEL0 **=** 0x00**;**

P2**->**SEL1 **=** 0x00**;**

P2**->**DS **=** 0x07**;**

P2**->**DIR **=** 0x07**;**

P2**->**OUT **=** 0x00**;**

**}**

//输出P1

void Port1\_Output**(**uint8\_t data**){** // write all of P1.0 outputs

P1**->**OUT **=** **(**P1**->**OUT**&**0xFE**)|**data**;**

**}**

//输出P2

void Port2\_Output**(**uint8\_t data**)**

**{**

P2**->**OUT **=** data**;**

**}**

//主函数

void main**(**void**)**

**{**

uint8\_t status**;** //储存开关的状态

uint8\_t colors**[]** **=** **{**RED**,**GREEN**,**BLUE**,**YELLOW**,**PINK**};** //储存颜色的数组

int flag **=** 0**;** //颜色的选择标志

int delay\_time **=** 1**;** //延时的秒数

Port1\_Init**();**

Port2\_Init**();**

**while(**1**)**

**{**

status **=** Port1\_Input**();**

Port1\_Output**(**1**);** //左边的红灯先亮0.5秒

Clock\_Delay1ms**(**500**);**

Port1\_Output**(**0**);**

Port2\_Output**(**colors**[**flag**]);** //右边的LED灯亮0.5秒

**switch(**status**)**

**{**

**case** SW1**:** //按下左边的开关，闪烁延时0.5秒

**{**

delay\_time**++;**

**break;**

**}**

**case** 0x00**:** //按下两边的开关，切换颜色

**{**

**if(**flag**++** **==** 4**)** //循环颜色

flag **=** 0**;**

Port2\_Output**(**colors**[**flag**]);**

**break;**

**}**

**case** SW2**:** //按下右边的开关，速度增加0.5秒

**{**

**if(**delay\_time **==** 1**)**

**break;**

delay\_time**--;**

**break;**

**}**

**}**

Clock\_Delay1ms**(**500**);**

Port2\_Output**(**DARK**);**

Clock\_Delay1ms**(**delay\_time**\***500**);** //闪烁的间隔控制

**}**

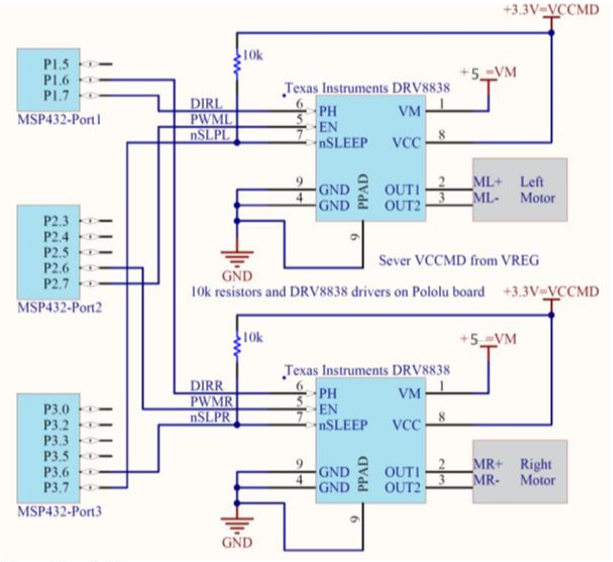
**}**

## 3.2 实验二

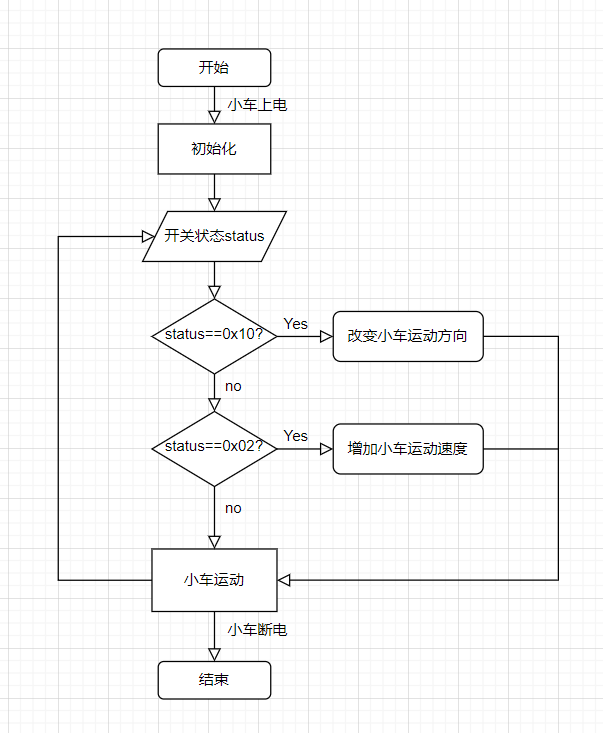
### 3.2.1 实验原理

编码减速电机组所使用是减速直流电机，与之相匹配的是TI的DRV8838驱动器的H桥电机驱动器，连接到CPU的P1.6、P2.6、P3.6、P1.7、P2.7、P3.7的六个GPIO口，该驱动器允许向前或向后转动每个电机。驱动器通过MSP432P401R产生的PWM波控制电机的状态，实现小车的启动、左右转弯、前进后退、掉头、停止等功能。

DRV8838输出驱动器块由一个配置为H桥的N通道功率MOSFET组成，以驱动电机绕组。它能够提供高达1.8A的输出电流。它运行在0至11V 之间的电机电源电压，以及1.8V至7.0V范围内的器件电源电压上。DRV8838具有一个PH/EN 输入接口，可以提供用于短路保护、过流保护、过热保护和欠压闭锁的内部关断功能。



### 3.2.2 实验流程图



### 3.2.3 程序清单

#include "msp.h"

#include "Motor.h"

#include "Clock.h"

#define SW1 0x10

#define SW2 0x02

#define RED 0x01

#define GREEN 0x02

#define BLUE 0x04

#define YELLOW 0x03

//初始化P1

void Port1\_Init**(**void**){**

P1**->**SEL0 **=** 0x00**;**

P1**->**SEL1 **=** 0x00**;**

P1**->**DIR **=** 0x01**;**

P1**->**REN **=** 0x12**;**

P1**->**OUT **=** 0x12**;**

**}**

//输入P1

uint8\_t Port1\_Input**(**void**){**

**return** **(**P1**->**IN**&**0x12**);**

**}**

//初始化P2

void Port2\_Init**(**void**)**

**{**

P2**->**SEL0 **=** 0x00**;**

P2**->**SEL1 **=** 0x00**;**

P2**->**DS **=** 0x07**;**

P2**->**DIR **=** 0x07**;**

P2**->**OUT **=** 0x00**;**

**}**

//输出P1

void Port1\_Output**(**uint8\_t data**){**

P1**->**OUT **=** **(**P1**->**OUT**&**0xFE**)|**data**;**

**}**

//输出P2

void Port2\_Output**(**uint8\_t data**)**

**{**

P2**->**OUT **=** data**;**

**}**

void main**(**void**)**

**{**

uint8\_t status**;** //储存开关的状态

int direction **=** 0**;**

Port1\_Init**();**

Port2\_Init**();**

Clock\_Init48MHz**();**

Motor\_Init**();**

int right**=**3000**;**

int left**=**3000**;**

**while(**1**)**

**{**

status **=** Port1\_Input**();**

**switch(**status**)**

**{**

**case** SW1**:**

**{**

direction**++;**

**break;**

**}**

**case** SW2**:**

**{**

Port1\_Output**(**1**);**

Clock\_Delay1ms**(**500**);**

Port1\_Output**(**0**);**

right **+=** 1000**;**

left **+=** 1000**;**

**break;**

**}**

**}**

**if(**direction **>=** 4**)**

direction **=** 0**;**

**switch(**direction**)**

**{**

**case** 0**:**

**{**

Motor\_Forward**(**left**,**right**);**

Port2\_Output**(**RED**);**

**break;**

**}**

**case** 1**:**

**{**

Motor\_Left**(**left**,**right**);**

Port2\_Output**(**GREEN**);**

**break;**

**}**

**case** 2**:**

**{**

Motor\_Right**(**left**,**right**);**

Port2\_Output**(**BLUE**);**

**break;**

**}**

**case** 3**:**

**{**

Motor\_Backward**(**left**,**right**);**

Port2\_Output**(**YELLOW**);**

**break;**

**}**

**}**

Clock\_Delay1ms**(**1000**);**

**}**

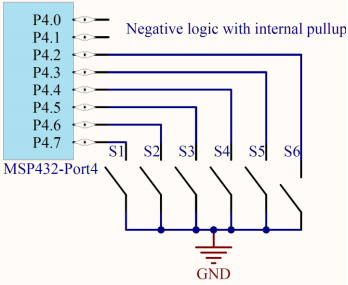
**}**

## 3.3 实验三

### 3.3.1 实验原理

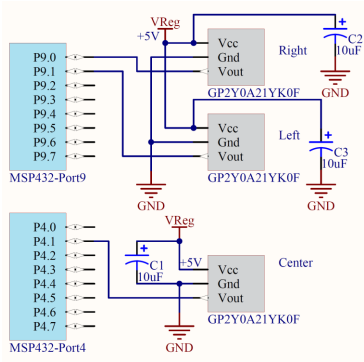
(1) 碰撞模块

为了保护小车，套件中包含了6个防撞开关，防撞开关从左到右分别连接CPU的P4.7、P4.6、P4.5、P4.4、P4.3、P4.2的六个GPIO口（图2.12）。防撞开关的作用除了可以在碰撞时起缓冲作用，还可以通过对MSP432P401R返回信息，使MSP432P401R做出相应的响应，修正运动线路。本设计利用边缘触发中断来检测碰撞传感器的碰撞。小车与物体的碰撞触发边缘中断，在中断服务程序中读取防撞开关的状态，一旦检测到小车发生碰撞，MSP432P401R则做出相应的相应，改变小车的前进方向，从而实现碰撞转弯功能。



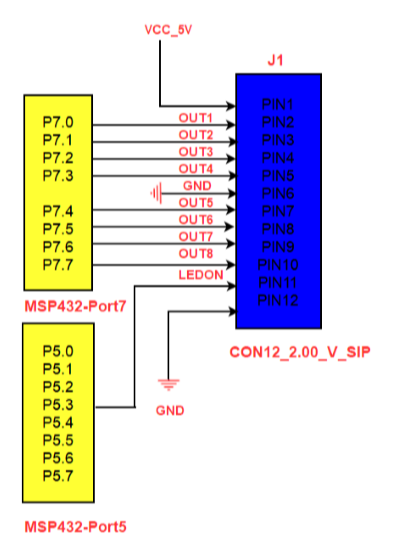
(2) 红外测距模块

在防撞开关的基础上，本设计还扩展了红外测距传感器以避开障碍物。该套件使用的为3个Sharp DP2Y0A21YK0F传感器，分别连接CPU的P9.1、P9.0、P4.1的三个GPIO口（图2.13），探测范围 20cm-150cm，其输出电压与距离成反比。探测原理为三角测量原理：红外发射器以一定的角度发射红外光束，当遇到物体以后，光束会被反射回来，如图2.14所示。反射回来的红外光线被CCD检测器检测到以后，会获得一个偏移值L，利用三角关系，在知道了发射角度a，偏移距L，中心矩X，以及滤镜的焦距f以后，传感器到物体的距离D就可以通过几何关系计算出来了，从而判别是否进行避障改变前进方向，实现避障功能。

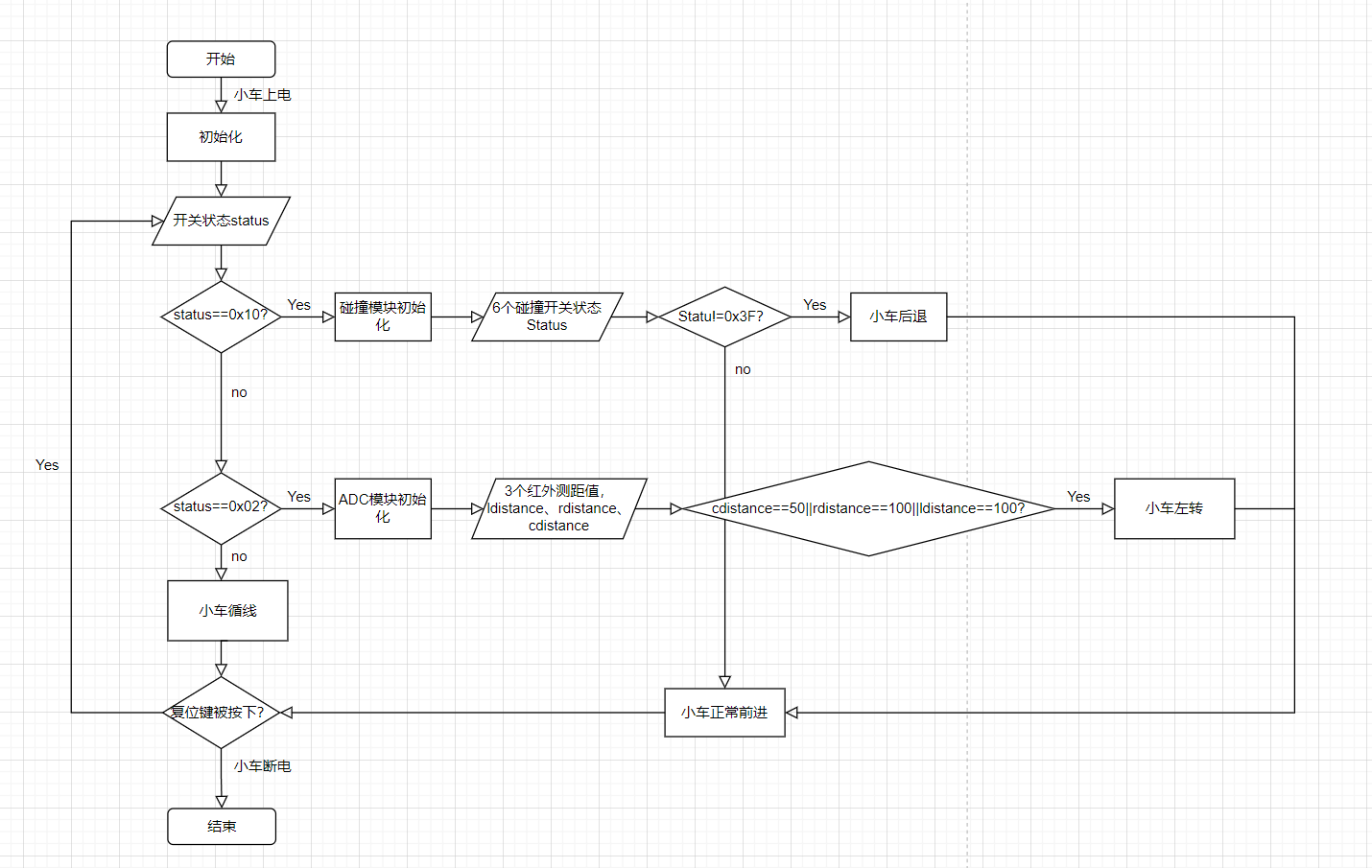


(3) 循迹模块

本设计采用JN\_LSA\_2RSLKBO1线阵传感器，包含8个传感器，每个传感器都对应二进制数，如果看到黑色则返回1，如果看到白色则返回0。如果小车正确地定位在线上，中间传感器将看到黑色；如果小车向左或向右偏离，则一个或多个外侧传感器将看到黑色；如果小车完全脱离轨迹，则所有传感器都将看到白色。MSP432P401R通过对传感器返回的数据进行分析，从而控制小车的行进状态，修正小车运动路径。



### 3.3.2 实验流程图



### 3.3.3 程序清单

#include <stdio.h>

#include "msp.h"

#include "ADC14.h"

#include "Clock.h"

#include "Bump.h"

#include "Reflectance.h"

#include "PWM.h"

#include "Motor.h"

#include "IRDistance.h"

#include "CortexM.h"

#define SW1 0x10

#define SW2 0x02

uint32\_t nr**,**nc**,**nl**;**

int32\_t cdistance**,**ldistance**,**rdistance**;**

uint8\_t Data**;**

uint8\_t Status**;**

void Port1\_Init**(**void**){**

P1**->**SEL0 **=** 0x00**;**

P1**->**SEL1 **=** 0x00**;**

P1**->**DIR **=** 0x01**;**

P1**->**REN **=** 0x12**;**

P1**->**OUT **=** 0x12**;**

**}**

uint8\_t Port1\_Input**(**void**){**

**return** **(**P1**->**IN**&**0x12**);**

**}**

void Port1\_Output**(**uint8\_t data**){**

P1**->**OUT **=** **(**P1**->**OUT**&**0xFE**)|**data**;**

**}**

void Walk**(**uint8\_t Data**)**

**{**

**if(**Data **==** 0x18**){**

Motor\_Forward**(**5000**,**5000**);**

**}**

**else** **if((**Data **&** 0xF0**)** **!=** 0**){**

Motor\_Forward**(**5000**,**200**);**

**}**

**else** **if((**Data **&** 0x0F**)** **!=** 0**){**

Motor\_Forward**(**200**,**5000**);**

**}**

**else{**

Motor\_Forward**(**0**,**0**);**

**}**

Clock\_Delay1ms**(**10**);**

**}**

void main**(**void**)**

**{**

uint8\_t status**;**

Port1\_Init**();**

Clock\_Init48MHz**();**

Motor\_Init**();**

**while(**1**)**

**{**

status **=** Port1\_Input**();**

**switch(**status**)**

**{**

**case** SW1**:{**

Bump\_Init**();**

**while(**1**)**

**{**

Status **=** Bump\_Read**();**

**if(**Status **!=** 0x3F**)**

**{**

Motor\_Backward**(**5000**,**5000**);**

Clock\_Delay1ms**(**1000**);**

**}**

**else**

**{**

Motor\_Forward**(**5000**,**5000**);**

Clock\_Delay1ms**(**10**);**

**}**

**}**

**}**

**case** SW2**:{**

ADC0\_InitSWTriggerCh17\_12\_16**();**

**while(**1**)**

**{**

ADC\_In17\_12\_16**(&**nr**,&**nc**,&**nl**);**

cdistance **=** CenterConvert**(**nc**);**

ldistance **=** LeftConvert**(**nl**);**

rdistance **=** RightConvert**(**nr**);**

**if(**cdistance **==** 50**||**ldistance **==** 100**||**rdistance **==** 100**)**

**{**

Motor\_Backward**(**5000**,**5000**);**

Clock\_Delay1ms**(**200**);**

Motor\_Left**(**5000**,**5000**);**

Clock\_Delay1ms**(**200**);**

**}**

Motor\_Forward**(**5000**,**5000**);**

Clock\_Delay1ms**(**10**);**

**}**

**}**

**default:{**

Reflectance\_Init**();**

Data **=** Reflectance\_Read**(**1000**);**

**switch(**Data**)**

**{**

**case** 0x1F**:**

**case** 0x1E**:**

**case** 0x0F**:{**

Motor\_Forward**(**5000**,**5000**);**

Clock\_Delay1ms**(**350**);**

Motor\_Left**(**5000**,**5000**);**

Clock\_Delay1ms**(**500**);**

**break;**

**}**

**case** 0xF0**:**

**case** 0xF8**:**

**case** 0x78**:{**

Motor\_Forward**(**5000**,**5000**);**

Clock\_Delay1ms**(**350**);**

Data **=** Reflectance\_Read**(**1000**);**

**if(**Data **==** 0x00**)**

**{**

Motor\_Right**(**5000**,**5000**);**

Clock\_Delay1ms**(**500**);**

**}**

**else{**

Motor\_Forward**(**5000**,**5000**);**

Clock\_Delay1ms**(**10**);**

**}**

**break;**

**}**

**case** 0xFF**:{**

Motor\_Forward**(**5000**,**5000**);**

Clock\_Delay1ms**(**350**);**

Motor\_Left**(**5000**,**5000**);**

Clock\_Delay1ms**(**500**);**

Data **=** Reflectance\_Read**(**500**);**

**break;**

**}**

**}**

Walk**(**Data**);**

**}**

**}**

**}**

**}**

# 4 调试过程

## 4.1 实验中遇到的问题

在进行本次实验的过程中，我碰到了一些阻碍。

当测试红外测距模块的时候，小车无法按照预估的现象行动，当程序烧录进去的时候，小车保持一动不动的情况，造成这种现象的情况有很多种，我无法确定究竟是哪一步出现了问题，这个时候就使用到了CCS 的调试功能，通过给代码下断点，查看寄存器的值和变量的值来一步步寻找问题的根源，最终也发现了问题的所在并成功解决。

成功解决了小车无法成功红外测距的问题之后，虽然小车可以测距，但是小车有时候会在很远检测到有障碍物就开始避障，这与我所期望的结果是不符合的。通过分析我发现，是由于代码的逻辑问题造成的这个问题，源代码逻辑是当红外测距的值大于等于某个值开始进行避障功能，这样就造成了一个问题，当障碍物在很远的时候，被小车检测到就会开始避障，同理，小于等于某个值也会造成小车可能已经碰撞到了某个物体才开始避障的尴尬情况。所以，如果想要小车在红外测距到某个确定值的时候就开始避障就需要使用等于等于。

进行循线功能的调试的时候，会出现小车底部传感器检测失误的问题，起初我以为是代码的逻辑问题，在经过反复测试与调试之后，推断出可能是寝室的地板上面的黑点会干扰小车底部的传感器检测黑胶带，造成误判。

验收小车的时候，小车循线跑到一半会停住，可能是由于场地摩擦力比较大以及贴黑胶带的白纸上有很多凸起的小山坡造成的，应该将小车轮子的转速调大一点来克服这个问题。

# 5 实验结果

## 5.1 实验成果

实验结果请观看提交的视频。

## 5.2 收获与感悟

在本次实验中，我负责主程序和各个模块程序的编写和调试。

通过这次微机原理与应用的实验，我收获颇多。通过这次实验，不仅让我学习到了很多有关于单片机嵌入式系统设计与开发的知识，同时也让我对于微机原理与应用这门课程的知识有了更深层次的认识。同时，这次实验也锻炼了我独自分析问题和解决问题的能力，我学会了如何给代码下断点，怎么通过CCS看变量的值与寄存器的值，也增加了我对于专业学习的兴趣，让我的本科生学习更加丰富多彩。