前言

所实现的功能:小车按黑线循迹左手法则走出迷宫.到终点碰撞物体后停止.按按键后返回.以最优路径返回起点.

首先十分十分十分感谢 https://bbs.nuedc-training.com.cn/thread-384-1-1.html 论坛上分享的代码.整个项目有了十分十分十分可靠的参考.我们的整个项目也是建立在这些东西之上的,再次感谢.

其次感谢 http://patrickmccabemakes.com/tutorials/Maze_Solving/ 让我们算法的思路有了很大的启发.整个记忆优化路径的部分有了很大的帮助.

还要感谢网上找的提供字符串替换算法的老哥,忘了网址的很遗憾.

实际上,TI-RSLK 小车走迷宫机器人在逻辑思维上并不十分困难, ,困难之处在于如何调那些不知所谓的接口.你并不需要掌握多么高深的专业知识,那些完全没有必要,反而会让人更加困惑.你只需要学了 c 语言,一切就易如反掌了.

http://edu.21ic.com/lesson/1984 登陆这个网站,有视频教程,在原本的 TI 官网也能找到英文的版本.但是这个外国老头讲的虽然实在是糟糕透顶云里雾里,但是我还是建议你看完第一单元的部分. 输入链接 www.ti.com/lit/zip/SLAC768,下载整个做小车走迷宫需要的实验例子之类的工程包包,(这个藏在第一单元的一个 pdf 里).还是那句话,建议你看视频,那会很有用.,看不懂了再来看我这个.

我先对教程视频来一个观看优先度的总结吧.

必看!!!:1 4 6 9 10 12 13 14. 而其他的不是很重要啦

必备知识

一 输入输出部分

虽然我说会了 c 语言万事大吉,但你打开文件后也会被 P2->DS = 0x07;这样的句子所困扰.那我们就来解答

P 是指端口 port,但他并非一个引脚,而是一组引脚,比如 port1.0,port1.1,port1.7 像是这样.

一般一个 Port 有 8 个(0-7)引脚.

0x 是指十六进制的数,比如 0xFF 实际上就是 11111111.

~是指非 &是与 |是或

我们 c 语言学 a+=1 的意思是 a=a+1

那像是这样 P1->OUT |= 0x12;也就是 P1->OUT=P1->OUT| 0x12

显然 P1 的 1 位和 4 位就被弄成了 1 OUT 就是 output 就是输出啦.

P1->DIR &= ~0x12;像是这样的意思就是

P1->DIR= P1->DIR& ~0x12 那么 1 和 4 位就被弄成了 0,DIR 就是方向 direction

0 代表输入,1 代表输出.

想必看到这里一定会困惑,怎么设置一个引脚还这么麻烦?因为这样设置不会影响到同一 port 中其他不相干的位置.这样就很友好!

P1->SEL0 &= ~0x13;

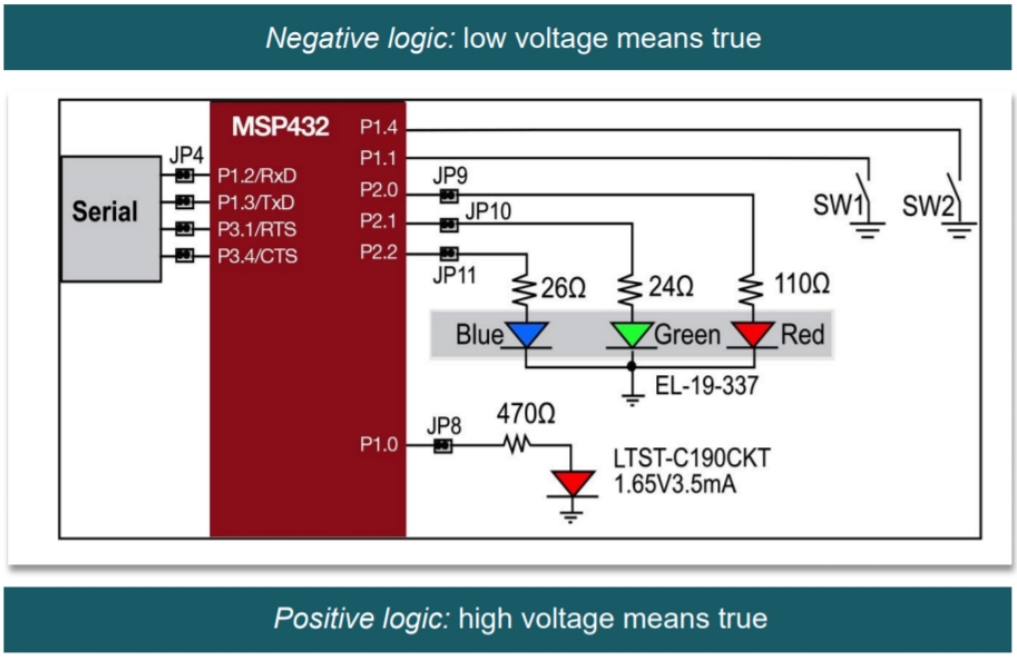
P1->SEL1 &= ~0x13;

SEL0 和 SEL1 就是设置模式,不必在这里纠结究竟在做什么,一般都弄成 0 啦

P1->REN |= 0x12;　设置上拉电阻啦 P2->DS |= 0x07;这个啦,都只出现了一次,完全不用知道究竟是做什么的.

Ok,那么输入输出部分就到这里.在所下载的工程包中有一个 InputOutput 的项目,看完了本章的内容后,就可以完全理解这个项目了.那么赶快按锤子编译该项目,再按小绿虫写到板子上,按开始键,开始调试吧.

下面这个图是就是 InputOutput 引脚的分配啦



顺嘴一提,在所下载的工程包包中,Lab 开头的是要你自己补充完全的,没有的像是 InputOutput 就是可做参考的例子.

顺嘴二提,在/inc/文件夹中我们会看到好多 c 文件和 h 文件。那么他们是做什么的呢？其实啊，同名的 c 文件和 h 文件是一对。函数的定义写在 h 文件中，方便我们 include。具体内容就在 c 文件中啦。建议你将我附带源码中大部分写好的 inc 文件夹覆盖到原工程包包的那个没有写好的 inc 文件夹.这样本教程所提到的 lab 就可以运行了

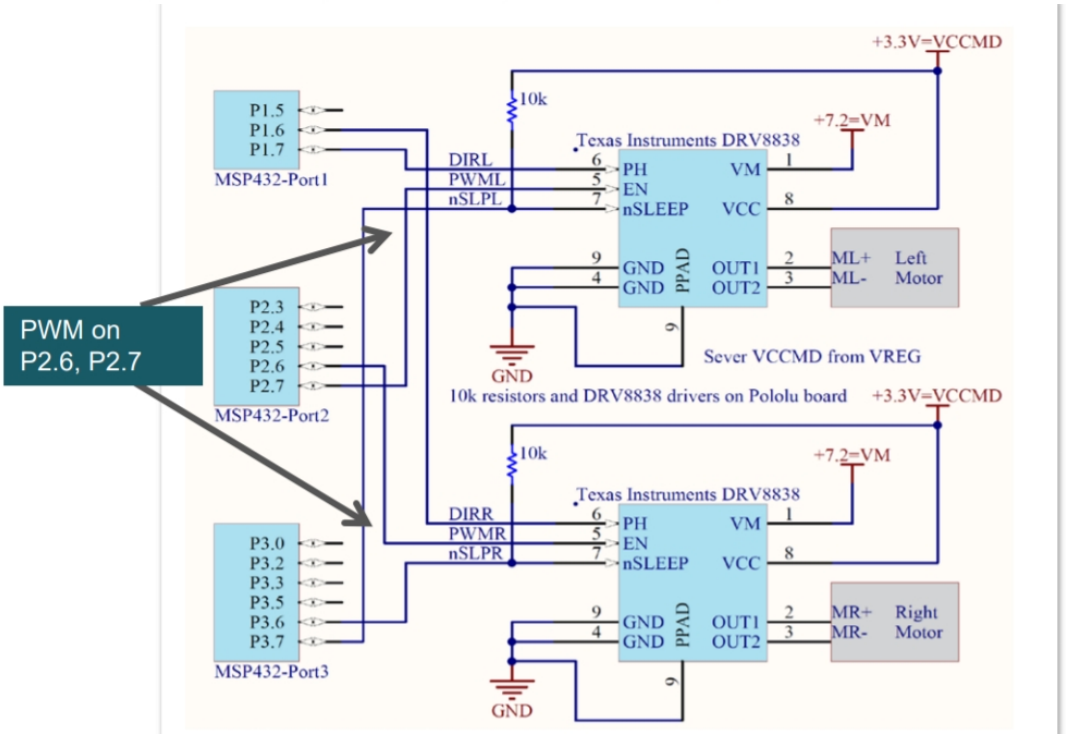
顺嘴三提,原工程包包中的 c 文件中即便等待你去补全,但也会有一些提示和有用的说明

二 让小车动起来

拿到一个小车,我们第一个想到的事情当然就是如何让它动起来.可教程视频在第 12 课才告诉我们如何让小车动起来,在第 13 课才告诉我们如何通过可控

变速的方式来让他动起来,实在是可恶.于是乎,我在第二部分就教教大家如何让小车动起来吧.小车最终动起来的方式在 13 课,所以我们就以 13 课为例.

LaunchPad	MDPDB	DRV8838	Description
P1.6	DIRR	PH	Right Motor Direction
P3.6	nSLPR	nSLEEP	Right Motor Sleep
P2.6	PWMR	EN	Right Motor PWM
P1.7	DIRL	PH	Left Motor Direction
P3.7	nSLPL	nSLEEP	Left Motor Sleep
P2.7	PWML	EN	Left Motor PWM



PWM 是什么?你不用这么困惑,你可以通过观看那个章节的视频来了解,百度一下稍微了解一下,或者完全不管他!小车动起来,说明要让他的电动机转起来,小车的电动机就是一个直流电动机,就和玩具四驱车用的那种是一个意思.只不过他能够受我们的控制罢了!

好了,看到这些图我们就知道要来做一些初始化的东西

看看原工程包包中的 Lab13_Timersmain.c,参考我附带源码中写好的 PWM.c Motor.c 并通过复制粘贴的方式来实现 Lab13 的内容,把 Lab13_Timersmain.c 中 program13.1 的内容拖到 main 中,这些是我们要实现的效果.

Bump 还没有讲到,可以先忽略删掉那一条.

```
Clock_Init48MHz();
      LaunchPad_Init(); // built-in switches and LEDs
      Motor_Init();     // your function
while(1){
```

```
        TimedPause(4000);
        Motor_Forward(7500,7500);  // your function
        TimedPause(2000);
        Motor_Backward(7500,7500); // your function
        TimedPause(3000);
        Motor_Left(5000,5000);     // your function
        TimedPause(3000);
        Motor_Right(5000,5000);    // your function
    }
```

注意啦,到了调试状态后要按左或者右的两个开关来让小车前动后动左动右动,因为 TimedPause 这个函数中的定义是

```
void TimedPause(uint32_t time){
  Clock_Delay1ms(time);          // run for a while and stop
  Motor_Stop();
  while(LaunchPad_Input()==0);  // wait for touch
  while(LaunchPad_Input());     // wait for release
}
```

他是在等你按完按钮再去执行下一个函数的.

做一些小小的说明吧,PWM.c 中的设置方式完全就是对 12 的依葫芦画瓢照搬照抄到 34 上面去.这一部分可能会有看不懂的东西比如

```
TIMER_A0->CCTL[0] = 0x0080;
```

这不用担心.通过照抄的方式先实现 lab13 的内容,让小车能动起来.只要理解 Motor.c 中的内容就可以了,pwm 可以先不管.

Motor.c 中的内容就是给左右轮子不同的 duty 值来控制转速,轮子前后方向不同来左转右转比如

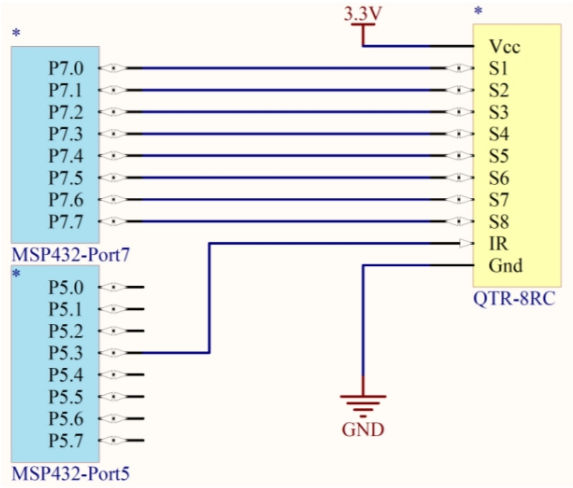
```
  P1->OUT&=~0xC0;// write this as part of Lab 13
  PWM_Duty3(rightDuty);
  PWM_Duty4(leftDuty);
  P3->OUT|=0xC0;
```

看了上述三个文件我们可以发现 leftduty rightduty 还有 period 显然前两者与第三者的比例就是控制转速的所在!

我所参考基于的方法完全摒弃了让人困惑的中断也就是 Interrupt,而是通过延时的方式来控制动作的时长.

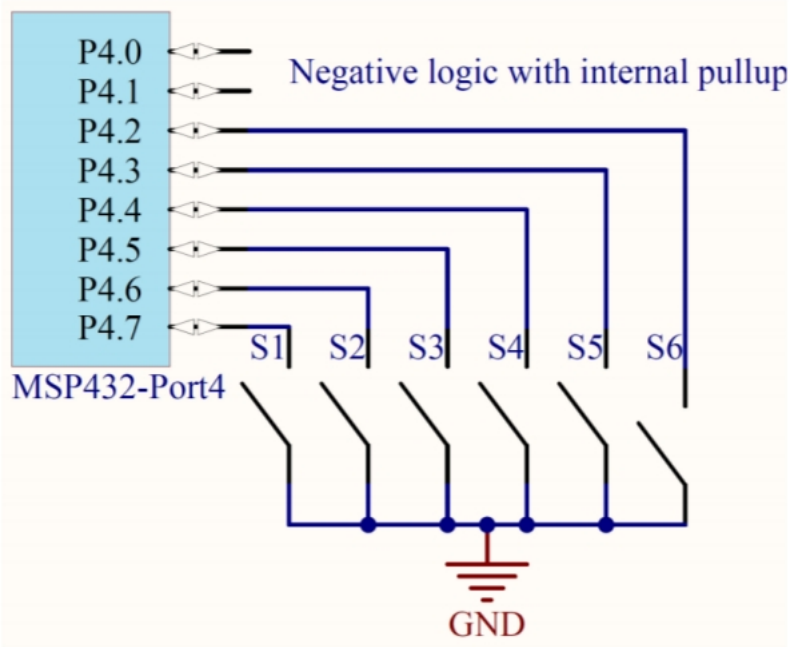
三 红外,碰撞模块

红外模块是视频教程的第 6 章 GPIO 或说是通用输入输出.是 P7.0-7.7 P5.3 就是一个激活啦让他工作的.注意了,扫描出来的低



位在左边,而高位在右边.

就是 0 位在左,7 位在右.也就是说 0x F0 表示的就是一个左边没路,右边有路的路口.反之 0x 0F 就是左边有路而右边没有路.十字路口或 T 字路口,就是 0xff,然而他们怎么区分呢.这我们之后会讲到.这一部分的函数代码在 inc/reflectance.c 里面.



碰撞对应 10 还有 14 章,我相信大家能够自己体会出来.在我们的应用中也不是十分重要的功能.这一部分的函数代码在 inc/bump.c 里面.大家自己看看吧.

四 如何走出迷宫,如何以最佳路径返回

左手算法

解迷宫的步骤是什么？基本上有两个步骤。首先是穿过迷宫，找到迷宫的尽头。第二步是优化这条路径，这样你的机器人就可以穿越迷宫，但要做到完美，就要走出任何死角。

机器人如何找到迷宫的尽头？我使用了一种叫做“左手法则”的算法。想象你在迷宫里，左手一直放在墙的边缘。这样做最终会让你走出一个无回路的迷宫。本教程只处理不包含循环的迷宫。

这种左手算法可以简化为以下条件：

如果你能左转，那就左转

否则，如果你能继续直行，那就直行。

再否则如果你能右转，那么右转

如果你处于死胡同，那就掉头去。

机器人必须在十字路口做出这些决定。十字路口是迷宫中任何一个你有机会转弯的地方。如果机器人遇到了转弯的机会而没有转弯，那么就考虑直行。在交叉路口或转弯时进行的每一次移动都必须存储起来。

举个例子(L=LEFT R=RIGHT S=STRAIGHT B=BACK)

机器人左转，然后掉头，再左转，那么就是直走。所以我们可以说 lbl=s。这个替换是机器人用来优化路径的。这是一个例子，但下面是整个列表：

LBR= B　　因此用了字符串替换函数来实现”LBR”到”B”

LBS＝R

RBL＝B

SBL＝R

SBS＝B

LBL＝S

来一整个例子,机器人走了 LBLLBSR,按着上面的式子化简完成之后就是 SRR.而我们是从终点到起点所以先取反 SLL 再从后向前就应该是 LLS.

因为我使用的红外传感器。使得不同的路口可能看起来一样(比如十字和 T 字都是 0xff)。所以我的机器人每次检测到交叉点都会向前移动一点，这样它就可以看到交叉点后的情况。最主要的例子是右转。如果你可以直走而不是右转，那么你应该直走，但唯一知道你是否可以直走的方法是向前行驶，看看线路是否继续。