中国科学技术大
2012robogame
技术报告

队伍名称：晴天霹雳

队员：蒋肖杰，薛亚军，张宇航，肖琪，李合壁

指导老师：邵长星

目录 Table Of Contents

第一章 机械部分

1.1 整体认识与简介

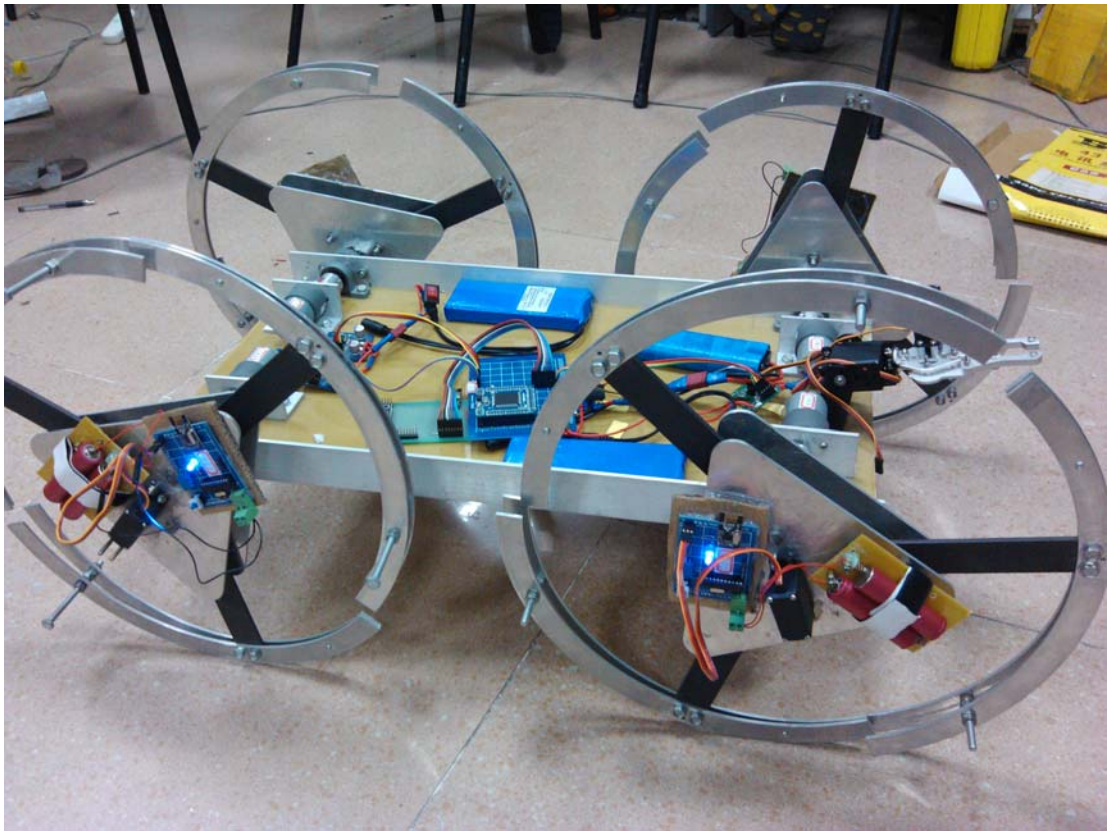


图 1-1

如图 1-1，这是我们按照计划书制作出的献花机器人。机器人主要由四个变形轮、车身板、机械臂构成，平路上行进时，舵机的自锁能够使车轮保持圆形状态，这样便于在平地上平稳地前行；当需要爬楼时，舵机转动一定角度，车轮三叶在车轮的传动下改变了形态，使得车轮叶互相错开，这样便能牢牢地卡主楼梯阶层，从而达到爬楼的目的。

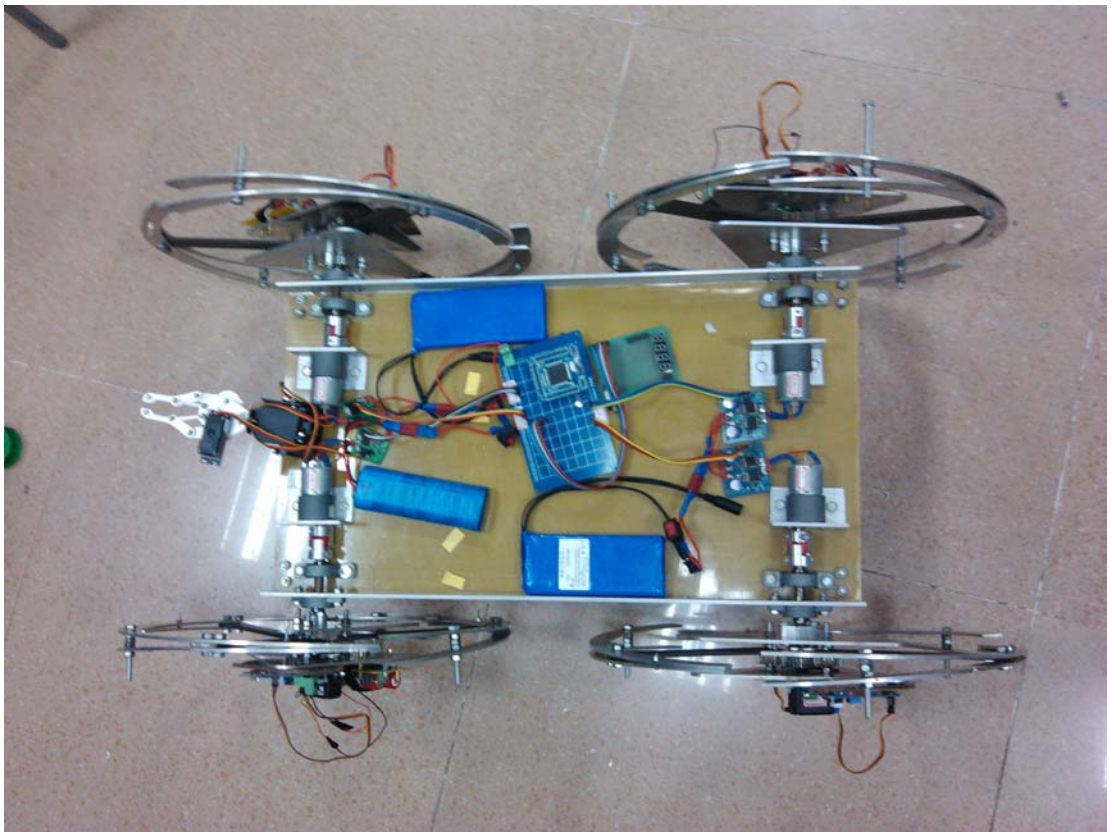
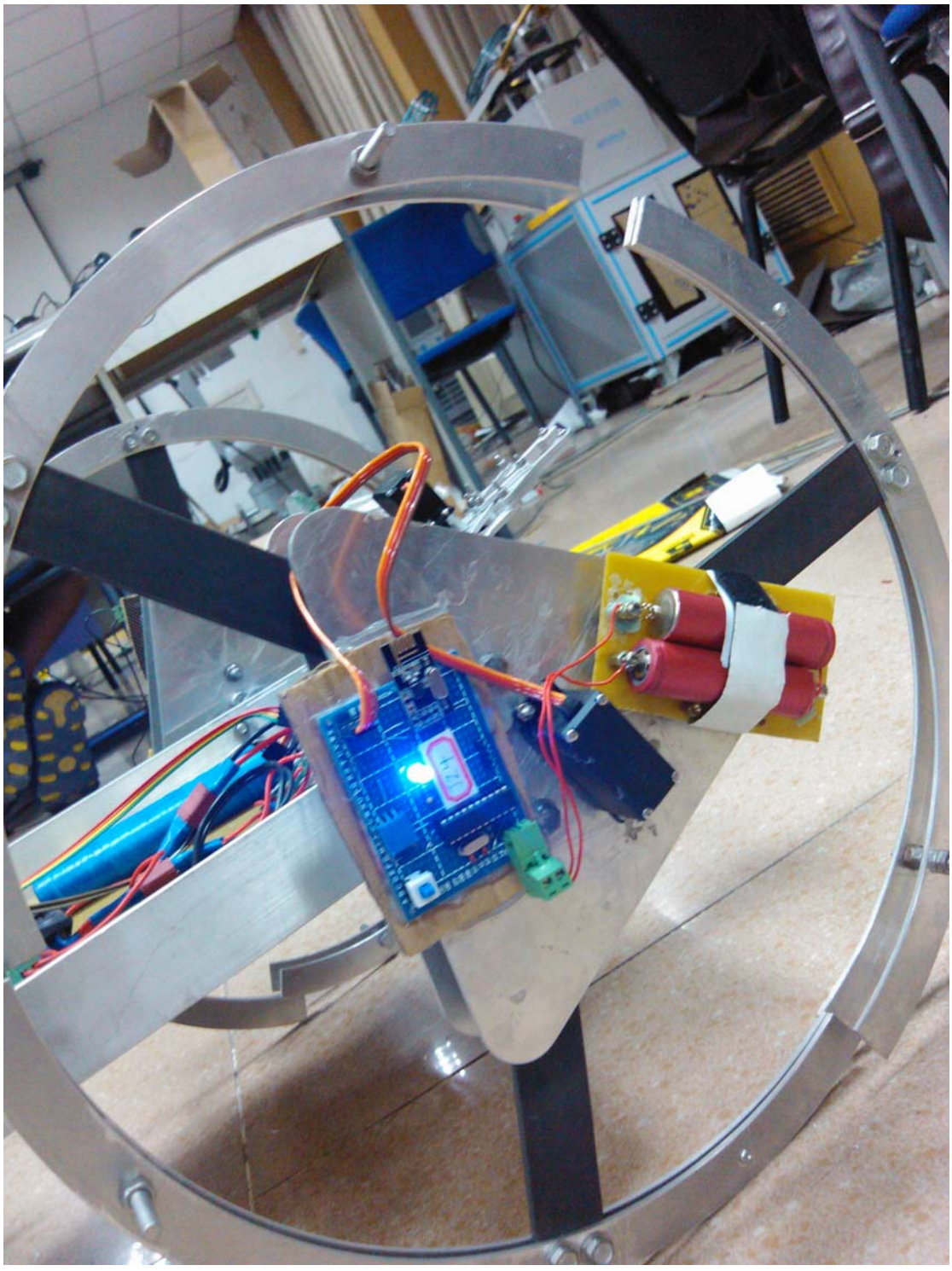


图 1-2

1.2 机器人部分机械的详解

1.2.1 变形轮的设计与制作



1.2.1.1、变形轮材料的选择

考虑到我们使用的是拥有特殊功能的变形轮，市场上没有成品卖，因此我们必须自己设计与制作。从加工的难易、材质的硬度、与地面的摩擦、质量、耐磨性等多个角度出发考虑，我们选择了铝材。

最初，直接对厚 2.5mm 的铝片进行切割打磨，手工制成变形轮，等到成型后发现这样的车轮并不能支撑起整辆车，行动起来左右摇摆，而且呈现出一定的变形趋势，可见这种纯铝在硬度、厚度上都不能满足机器人的需求，需要重新选材。经过与其他组的交流以及网上查询，我们选择了 3.5mm 的铝合金，并使用了机械

加工，事实证明这种材质完全能够满足机器人的需求，而且从外观上来看，具有一定的可欣赏性。

1.2.1.2、齿轮的传动问题

我们的设计精髓在于变形轮能在爬楼前变换形态使之能够爬楼，这样就要求有那么一个结构能够将舵机的转动量传递给变形轮三个车叶。我们考虑将一个大齿轮与舵机固定住，分别用三个小齿轮与三根车叶固定，然后通过齿轮间的磨合使得大齿轮带动小齿轮转动，达到预期的目的。齿轮与舵机、车叶的固定要求很高，不能轻易地改变形态。我们使用了厚达十几毫米的钢铁齿轮，在上面打两个对称的孔，通过螺丝的紧固将齿轮与舵机舵盘、车叶固定在一起。

1.2.1.3、舵机的安装

舵机在变形轮中起提供动力的关键作用，但舵机的安装却是一个技术性问题。起初，我们考虑过使用电机控制变形轮变形，但电机的一个比较突出的问题是控制不精确，加上自锁力度不够，我们很快放弃电机的使用。确定使用舵机后，我们先是考虑将舵机固定在车身，通过传动带将转动量传递给齿轮。这样的话，舵机的传动就会与变形轮整体的转动有冲突，要解决这一冲突，我们得将舵机与变形轮同步转动。因此，把舵机直接安装在变形轮的外侧三角盘上比较合适。

在舵机安装过程中我发现舵机的连线是一个问题，由于舵机与变形轮同步转动，导致三根导线也会转动，而导线的末端如果固定在车身上必然导致导线缠绕在轴上，限于导线的长度，最终导线会缠死阻止车轮转动。经过组内讨论，产生两种想法:一是采用离合器，二是采用电刷。综合考虑成本、安装难易、运行难易等方面因素，确定使用电刷方案，也就是将导线与电刷滚轴固定，接上三处导体，通过三根电刷将其引伸出去。且不说电刷的安装非常纠结，最后的效果不是很满意。这时，组员提出使用无线模块，也就是在变形轮上安装舵机和电源，通过无线模块向其发出相应的指令信号。这个方案确实是变形轮舵机安装的最好解决方案，我们一直使用到比赛前。

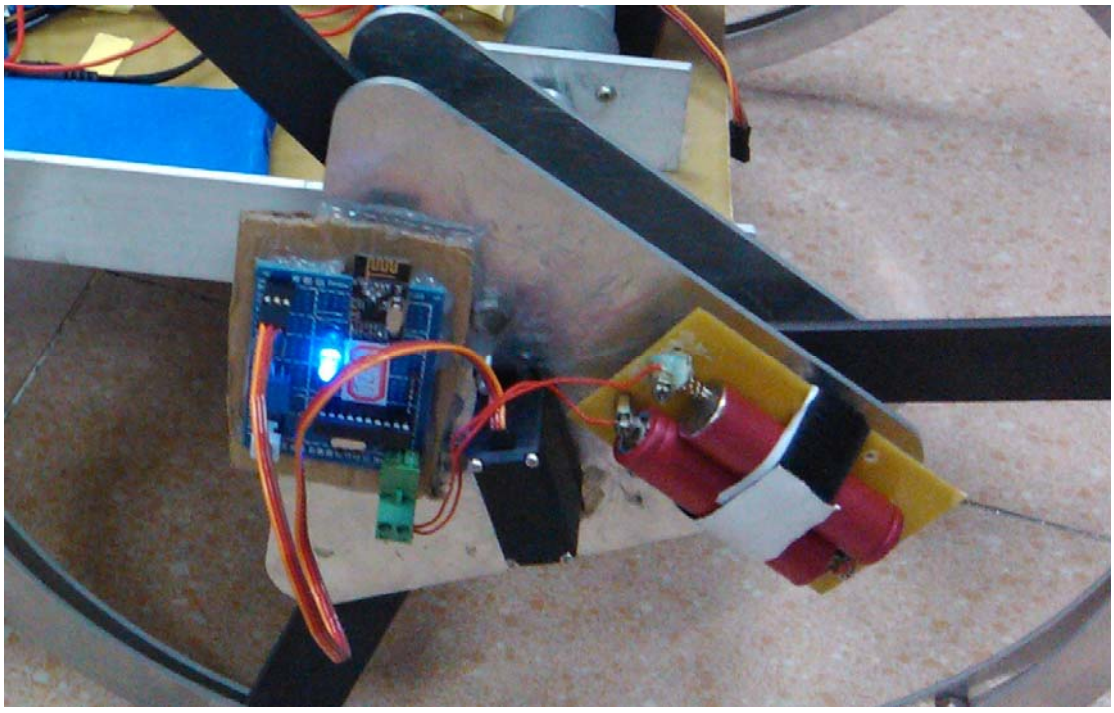


图 1-2

1.2.1.4、传动齿轮的固定

既要保证齿轮能够紧紧地固定在三角盘上，又要确保其能顺畅地转动，滚轴是最佳的选择。经过仔细地寻找与测量，我们找到了与齿轮能很好粘合的滚轴，下面就是要找出轴，这根轴要恰好与滚轴内径相当，还有能与三角盘固定。滚轴的内径大约 6.2mm，而我们只有 6mm 的螺丝，看似能凑合使用，然而这种配合的效果是不令人满意的，由于 0.2mm 的误差，车轮能够在轴上左右一定程度地摇晃，这是我们不愿接受的。为了取得尽可能完美地效果，我决定对 7mm 的均匀铝棒进行打磨，使其尺寸近于 6.2mm，同时借助螺纹工具能够在铝棒的两端做出螺纹来，这样既解决了摇晃问题，也解决了固定问题。

1.2.1.5、车叶

最初的想法就是将一个环形铝截成均匀截成三段，制成三个车叶。在反复测试和思考后，认为车叶必须达到两个标准：硬和稳。如果只用单片薄铝片的话，车在行进过程中极可能对车叶产生压力破坏其应有的形状。因此，我们使用了两层铝片，构成 V 字形，这样既稳固又坚硬，能够取得良好的效果。

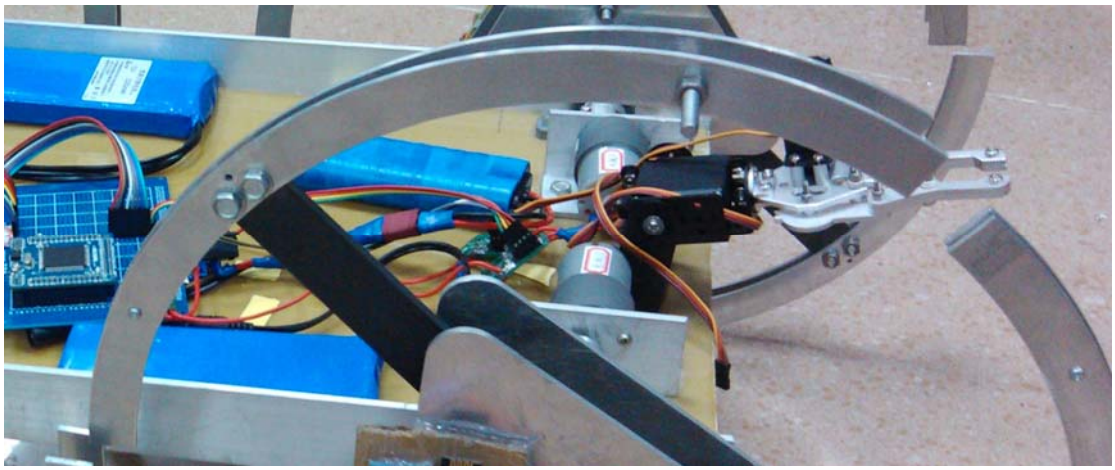


图 1-3

1.2.2 电机的安装与联轴器的使用

1.2.2.1、电机部分

电机的选择实在很重要，这不仅是我们队的感慨，更是广大献花队的感慨，电机选不好小则影响最终成绩，重则直接导致一些队伍退出比赛，实在可惜。刚开始我们并未对电机的选用重视，只是简单购买了四个电机，然而到调试阶段问题突出：机器人爬不上楼！于是我们重新认真考虑电机，从扭矩等各种参数逐一分析其性能，最终确定一款性价比较高的墨西哥进口电机。图 2-1 是原电机，图 2-2 是墨西哥进口电机。

图 2-1

图 2-2

由于电机上带有螺丝孔，因此我们固定电机比较轻松，直接制作一角铝将其与车身固定。

1.2.2.2 联轴器的使用

如何将电机与车轮连起来？显然是使用联轴器，现买的联轴器也是实现这么一个功能的，但在实际使用过程中，我们发现联轴器本身自带的螺丝并不能把电机转子、车轮轴紧紧固定。由于电机会产生较大的力而车轮同时因转动需要也产生很大阻力，这样对联轴器的摩擦要求很高，不能打滑。然而事实证明仅靠摩擦来固定两轴是不现实的，因为行动过程不断产生磨损使得摩擦力不断减小。那么我们就不得不从机械角度考虑如何加固的问题了，思考分析后，在联轴器上纵向打孔，同时在两轴上也打相应的孔，使用 M4 或 M3 螺丝穿过，这样借助机械的力量能够使两轴良好地“连”在一起。

图 2-3

1.2.3 车身

车身的设计相对来说比较简单。起初我们想简单地使用一块铝板做车身，然而这带来了两个问题：一是铝板较重，对电机造成一些压力；二是由于电机与车身直接固定，铝板的导电性质致使电机漏电。为了解决这两个问题，我们将铝板取代以电木板，达到了理想的目标。

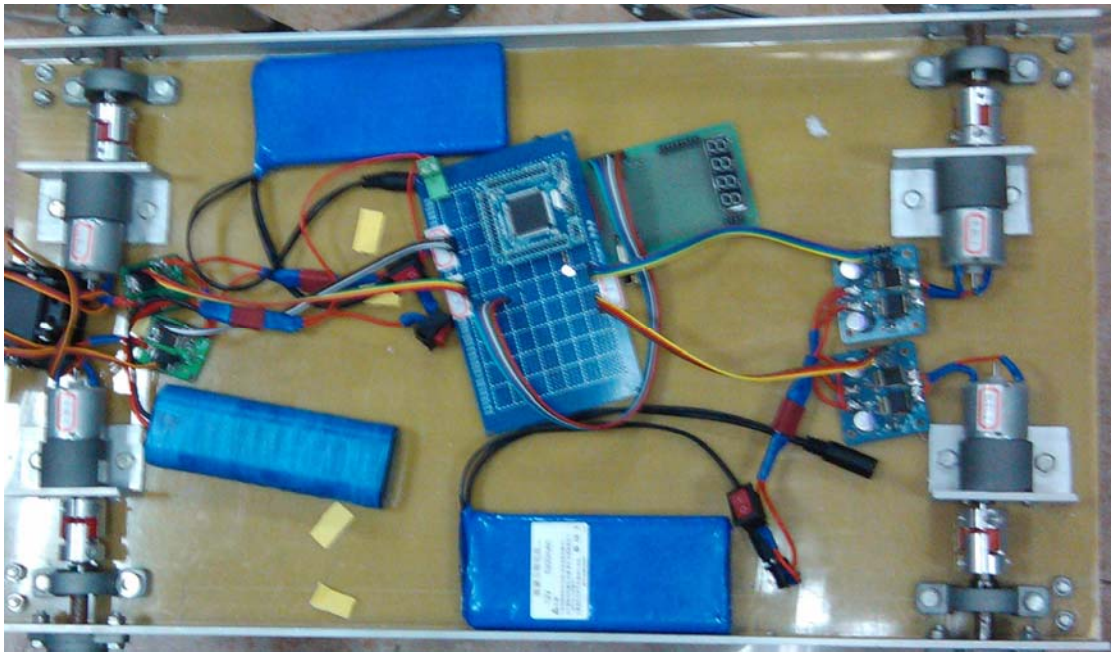


图 2-4

1.2.4 机械臂

一开始我们直接使用成品机械臂进行拿花，然而取得的效果非常不好。由于车在每次巡线过程都会产生一定误差，这样每次取花的位置都有些偏差，而成品的机械臂对位置要求非常高，最好每次都在同一位置，这显然是不可能。参考了其

他队的方案，我们决定使用简单而高效的上下咬合式机械夹持装置。

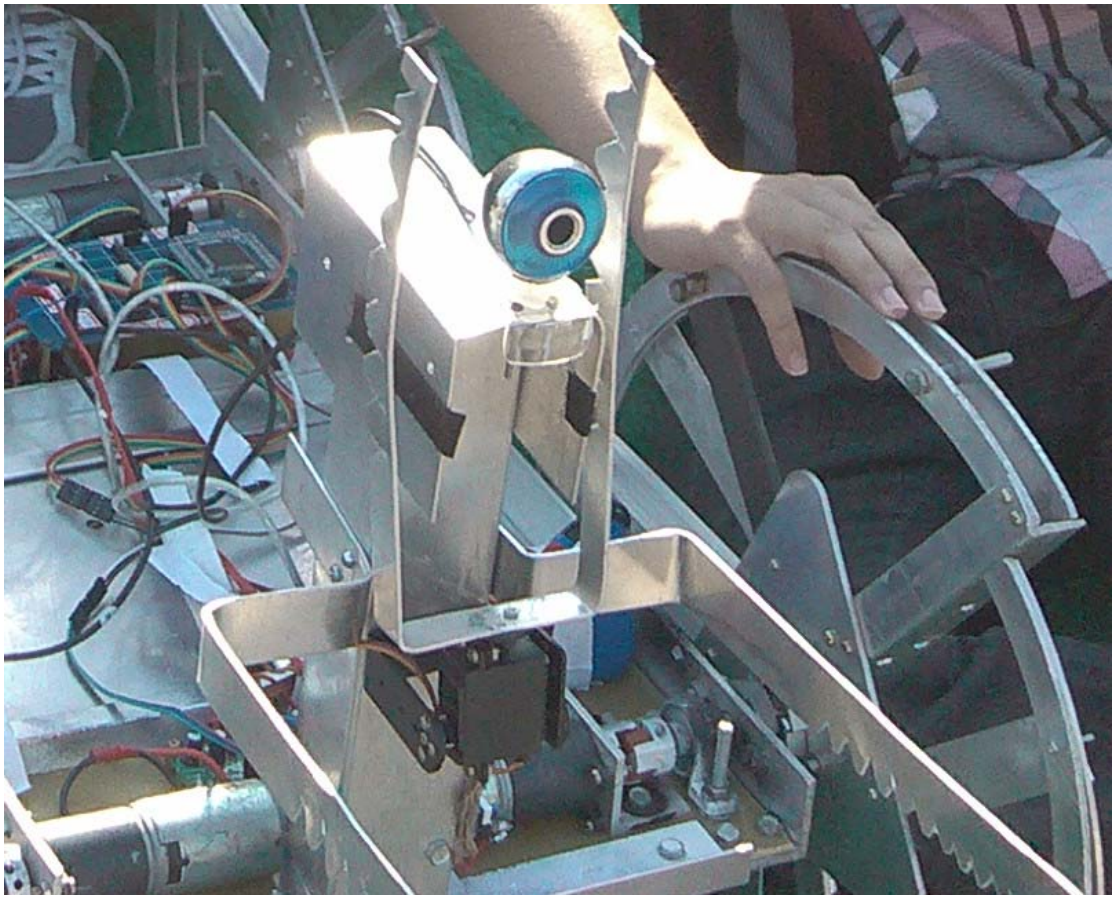


图 2-5

舵机通过轴与车身固定，本身与下颌装置固定，这样，舵机能带动下颌装置上下咬合，从而达到夹花的目的。

1.3 最后的整改

我们不得不承认，问题是在一次次调试后产生的。原先我们想象变形轮能够在平地上以圆形状态前进，上楼梯前变换形态直至放花。然而在比赛前几天我们发现舵机并不能支持车身，也就是说舵机被损坏，支持不了车身的压力。当然我们知道选用更好的舵机能够解决问题，但是，经费和时间是摆在我们面前最大的坎。经过激烈的讨论，我们决定放弃使用变形轮方案，直接采用车叶差错轮，这样，平地上能够在允许的振动下实现前进、倒退和转弯；上楼时，能利用车叶间的错位卡住台阶，从而借助电机的力量爬楼。

图 2-6

事实以及最后比赛的场景证明了我们方案改变的正确性。

1.4 机械部分的总结

可以这么说，机械是此次献花的重点，如果机械不能做得很好，整个比赛是不能顺利完成的，同时，机械也是最费时、费钱、费力的环节。

机械的制作需要以下几个必要因素：1、队员的合作与支持，一个人是不可能独立搞定机械的；2、积极的思考与分析，尽量采取有效简单地机械结构；3、巧用机械知识，善于使用工具。

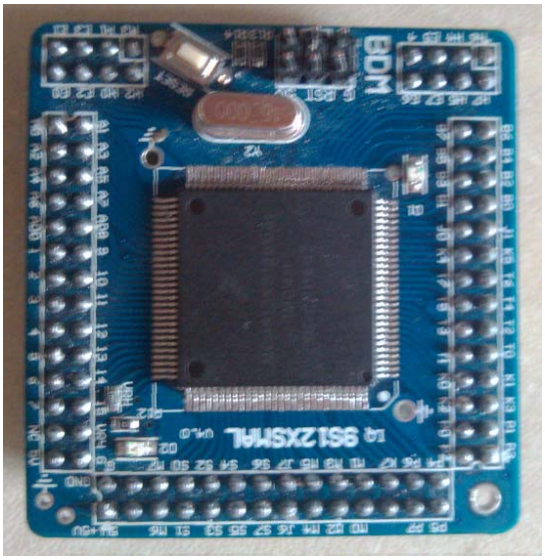
第二章 电路

2.1 概述

我们的电路主要分成四个部分：最小系统板，主板，电机驱动板，舵机控制板。最小系统板是从淘宝上买的 PCB 板；主板上包含电源控制电路，串口通信电路，光耦隔离电路，电机、舵机控制引脚和一些方便调试的电路；电机驱动板是做飞思卡尔智能车用的 PCB 板；舵机控制板主要由一块 STC12C2052AD 及其外围电路组成。

2.2 最小系统板

我们没有采用大家常用的 AVR 和 51 系列单片机，因为做了飞思卡尔智能车，就继续沿用我们熟悉而又强大的 MC9S12XS128。这款单片机只有 QFP 和 LQFP 两种形式的封装，焊接起来比较麻烦，我们就从淘宝上买了现成的 PCB 板，轻巧而稳定。



XS128 最小系统板实物图

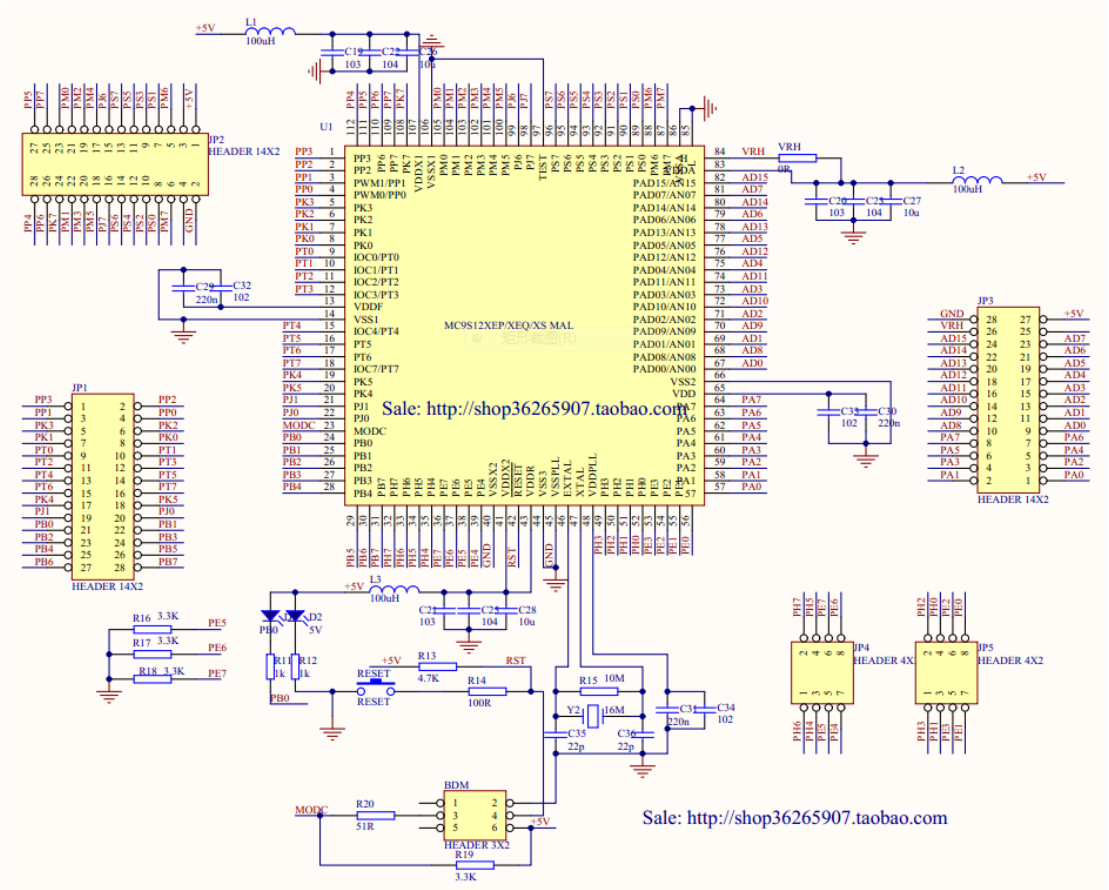


图 x-x　　XS128 最小系统原理图

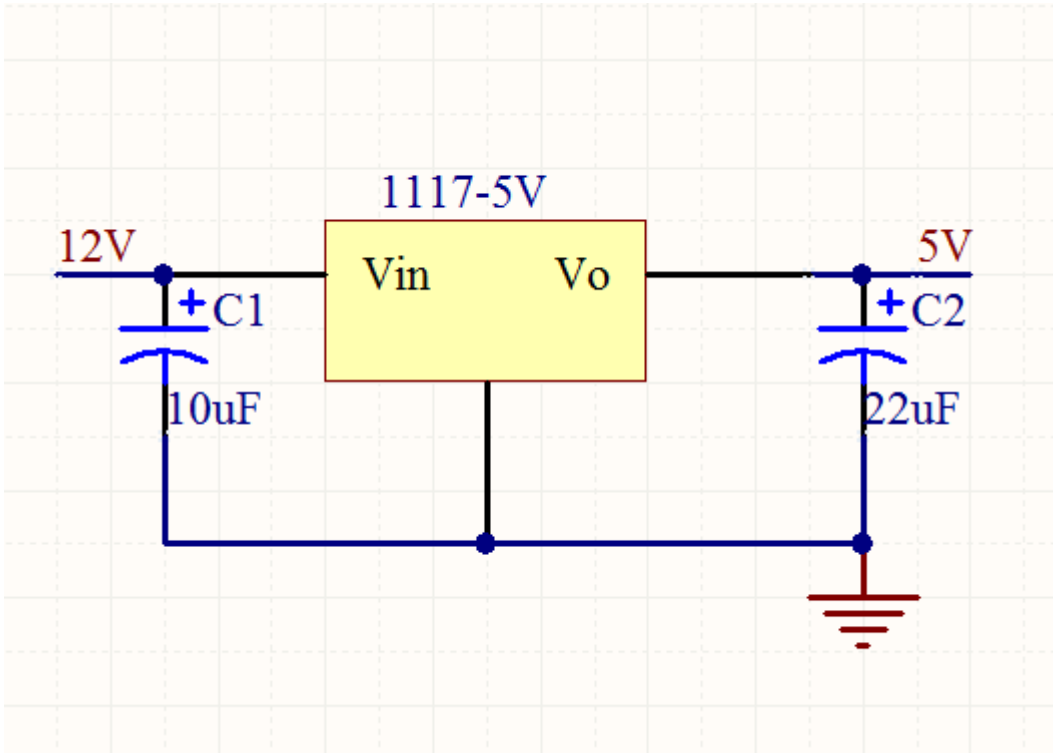
2.3　主板

这是我们机器人的核心控制板。包含了各种模块和接口。

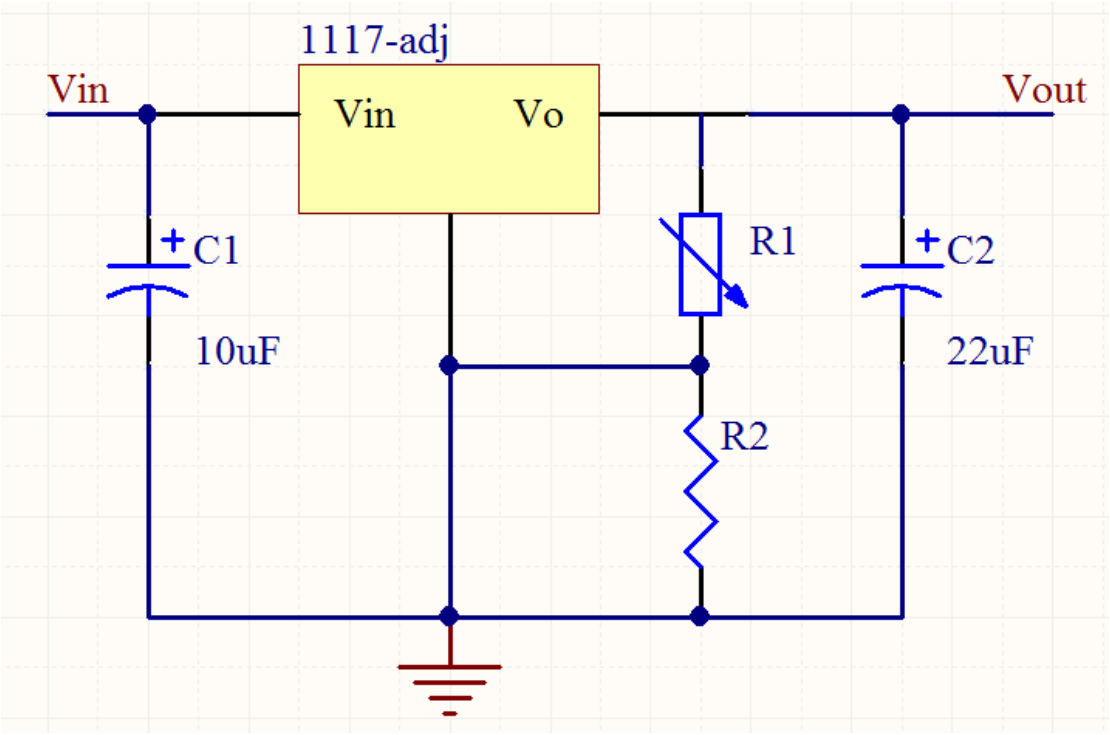
2.3.1 电源模块

机器人总电源是从淘宝上买的 12V 锂电池，可以直接给 12V 电机供电，并通过稳压模块给单片机，舵机等供电。其中给单片机一路供电的为固定 5V 输出，给舵机供电的一路采用 1117-adj 做成可调电压输出。

12V 锂电池



5V 固定输出

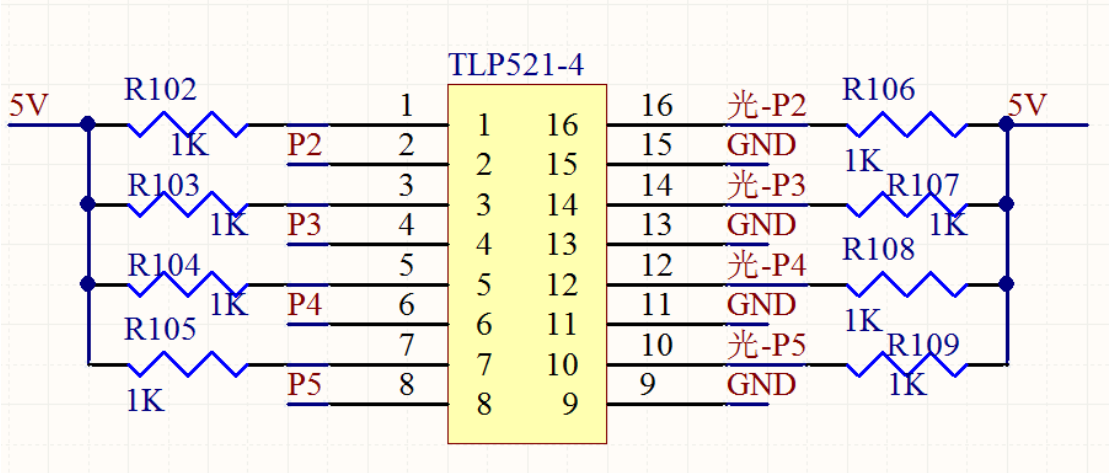


电压可调输出

$$V_{out}=\frac{R1+R2}{R1}\times(1.25V)+(50uA)\times R2$$

2.3.2 光耦电路

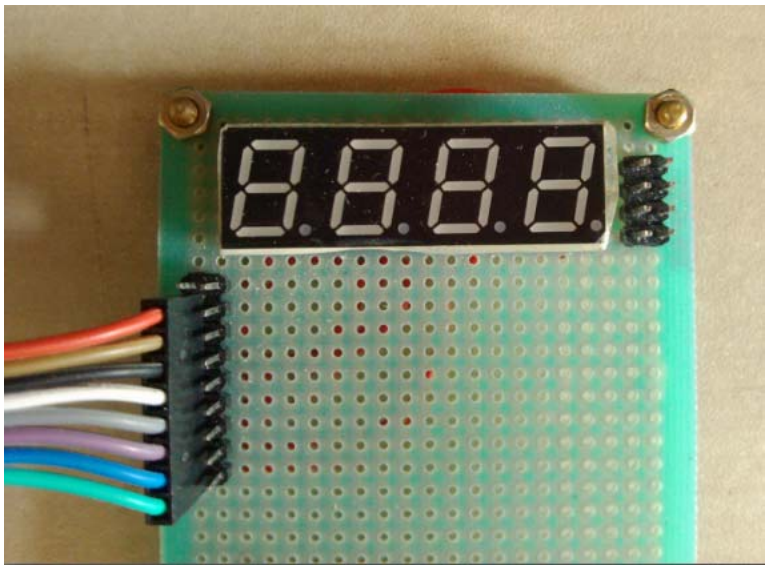
为了防止电机控制电路中大电流对单片机的影响，我们在单片机的输出控制口和控制电路间加了光耦隔离电路，以使单片机能稳定工作。



光耦电路原理图

2.3.3 IO 接口和数码管显示

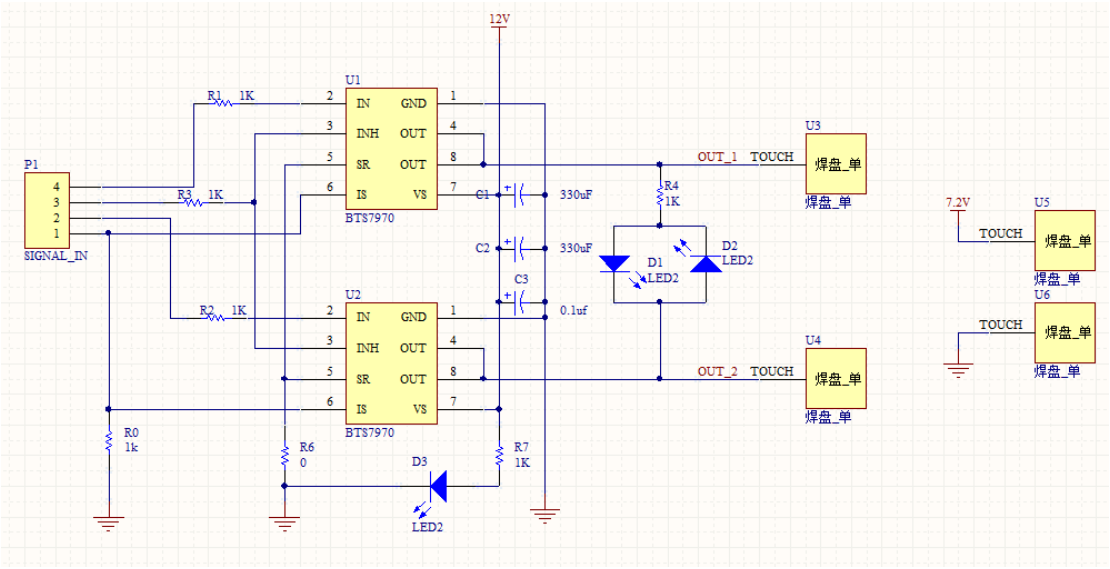
将单片机各种输入输出接口引出，如舵机、电机控制接口，串口通信接口，舵机控制板控制接口，数码管控制接口等。增加数码管显示，为了方便调试。



2.4 电机驱动模块

电机是整个机器人的动力核心，由于机器人要上楼梯，需要电机提供很大的力，机器人电机功率很大，流过的电流很大，这就需要一个好的电机驱动板的支持。

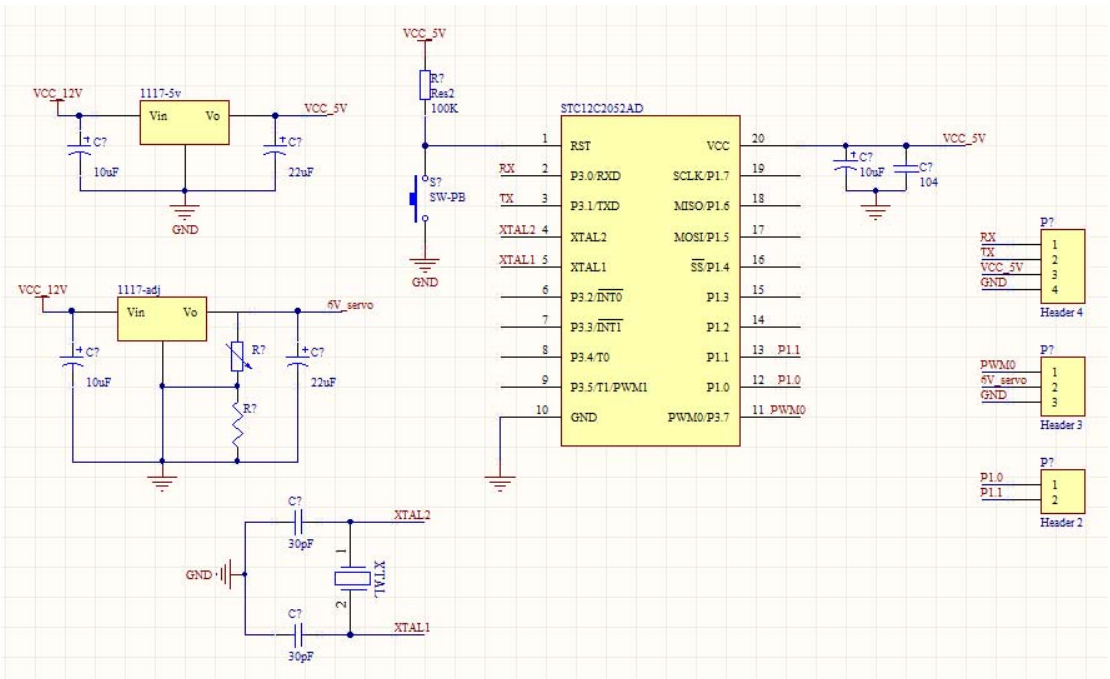
大家普遍采用的一种电机驱动板是 L298N，电路简单而且价格也较便宜。但是考虑到我们机器人电机功率很大，采用 L298N，最大只能提供 2A 电流，而且转化效率低，发热量大。因此我们采用了半桥芯片 BTS7960，用两片组成全桥，控制一路电机。它最大能提供 43A 的电流，转化效率高，发热小，完全满足我们的需要。其电路原理图如下，



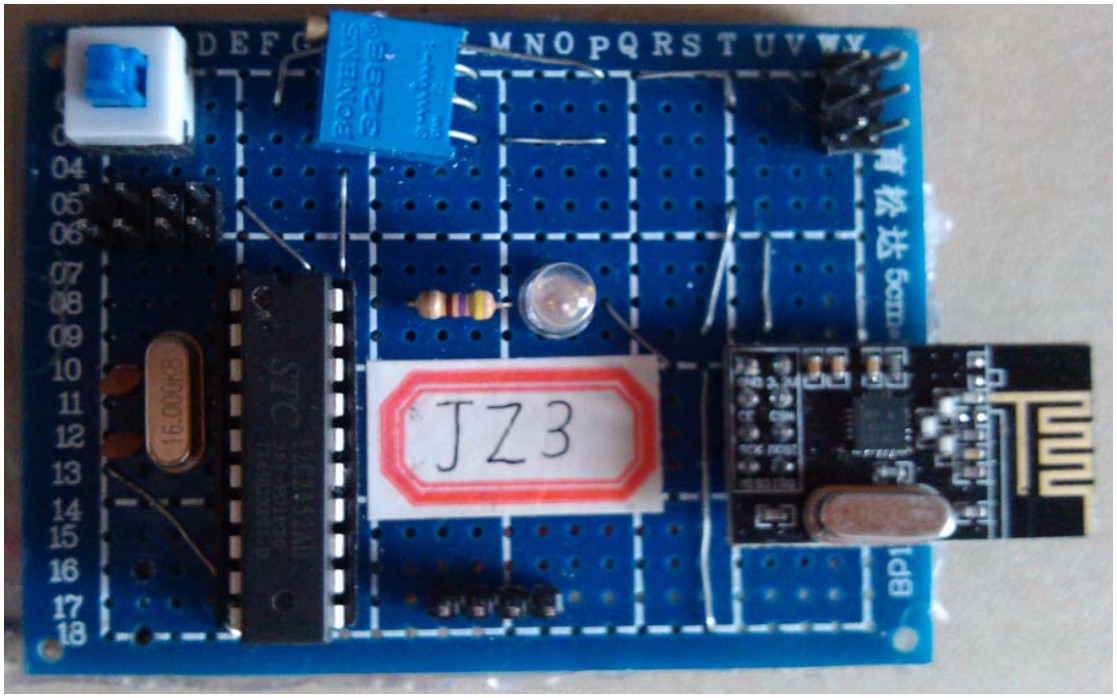
电机驱动原理图

2.5 舵机控制模块

因为我们的核心最小系统板上只能输出 8 路 PWM，恰好控制四个电机，还需要 1 路来控制机械臂上的舵机，如果采用普通 IO 口模拟 PWM 输出，会使我们的控制程序有些麻烦。于是我们采用一块 STC12C2052AD 作一个舵机控制板，接收 XS128 的控制指令，来控制舵机做出相应的响应。原理图如下：



舵机控制板原理图



舵机控制板实物图(包含了没用到的通信模块)

2.6 总结

以上就是我们经过不断的测试和改进，最终确定的电路部分。从中学到了很多知识，也吸取了很多教训。如为了防止短路，我们用塑料螺钉把电路板固定在机器人底板上；舵机控制板后还加了一层绝缘材料。这些都可以提高系统的稳定性，为程序的完美运行打下了坚实的基础。

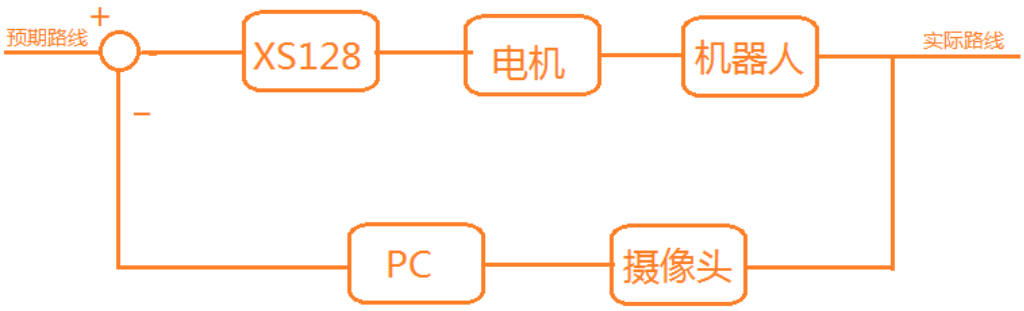
第三章 控制程序

3.1 概述

根据本次比赛的要求，机器人需要自主运行按照预定的路线，完成献花任务。我们使用摄像头采集道路和路标信息，用 PC 来做图像处理，把处理得到的结果发给单片机，完成相应的控制任务，如转弯，加速，减速，上下楼梯，拿花，放花等。

3.2 控制流程

摄像头负责采集信息，PC 负责图像处理任务，XS128 负责机器人的控制，PC 和 XS128 之间通过串口通信。流程图如下：



简化控制流程图

3.3 控制策略

主要根据图像处理得到的结果来实现闭环反馈控制。因为路线是确定的，控制过程主要分为三阶段：启动下楼，寻路标走，上楼献花。

第一阶段，启动下楼：

因为上下楼梯对机械损耗很大，令又考虑到稳定性，下楼时速度较低，保证机器人能平稳顺利的下楼。

第二阶段，寻路标走：

根据比赛要求，机器人需要按照预定路线完成任务。在寻路标过程中，主要根据摄像头采集的信息，PC 经过图像处理，识别机器人与路标的位置偏差，通过反馈控制来缩小偏差，以使机器人能按照预定路线行走。

由于路面不平、机械结构不完善、摄像头反应不够灵敏和一些外来的干扰因素等，在机器人行走过程中，摄像头会在某一时刻丢失路标，下一刻又会找到，如果不加处理，机器人很容易跑偏，这不是我们所期望的。于是，我们又加入了“丢帧处理”程序，使机器人抗干扰性更强，系统更加稳定。

第三阶段，上楼献花：

根据我们机器人的机械结构，当上楼梯时，机器人的重心偏后，主要靠后轮电机来使机器人爬上楼梯。经过不断的调试，我们最终确定了上楼梯时，机器人前后电机的参数，使机器人能稳定快速的上楼梯，完成献花任务。

3.4 控制程序

Init.c，初始化相关寄存器

```
/************************************************************
* File Name:       init.c
* Discription:         Initial related registers
* Writer:          Xiao Chongerfei
```

```
* Last Edit Time:  2012-9-14
************************************************************/
#include "derivative.h"

#define   LED     PORTB_PB0
#define   TX0     PORTB_PB1

/************初始化锁相环*****************/
void PLL_Init(void)
{
              // PLLCLK     = 2*OSCCLK*(SYNR+1)/(REFDV+1)
              // 锁相环时钟 = 2*16*(4+1)/(0+1)=160MHz
              // 总线频率        = PLLCLK/2=80MHz
     CLKSEL_PLLSEL = 0; // 选择时钟源为OSCCLK(外部晶振)
     PLLCTL_PLLON  = 0; // 禁用锁相环
     SYNR_VCOFRQ        = 0b11; // 选择VCO频率范围为最大
     SYNR_SYNDIV        = 4; // 设置SYNR寄存器值为
     REFDV_REFFRQ  = 0b11;   // 选择REF频率范围为最大
     REFDV_REFDIV  = 0; // 设置REFDV寄存器的值为
     PLLCTL_PLLON  = 1; // 启用锁相环
     _asm nop                // 汇编空指令
     _asm nop                // +1
     _asm nop                // +2
     _asm nop                // +3
     while(!CRGFLG_LOCK);    // 等待锁相环稳定
     CLKSEL_PLLSEL = 1; // 选择时钟源为PLLCLK(锁相环)
}
/*****************************************/

/************PIT初始化******************/
void PIT_Init()
{
     PITCFLMT  = 0x00;       // 关闭PIT模块
     PITFLT         = 0x01;       // 0路定时器加载倒计时数值
     PITCE          = 0x01;       // 0路定时器使能
     PITMUX         = 0x00;       // 16位计数器基于微计数器

     PITMTLD0  = 159;        // 8位递减计数值
     PITLD0         = 49999; // 16位递减计数值
                         // frequncy=BusClk/160/50000=10
                         // cycle= 100ms

     PITINTE        = 0x01;      //使能路中断
     PITCFLMT  = 0x81;     // 使能PIT模块,使能微计数器
```

```
}
/************************************/

/*****************PWM初始化*********************/
void PWM_Init(void)
{
    PWME          =   0x00;    // 禁用所有PWM通道
    PWMPOL        =   0xFF;    // 全部极性设置为高
    PWMCAE        =   0x00;    // 全部对齐方式设置为左对齐
    PWMPRCLK      =   0x22;    // 预分频寄存器设置为A=B=80M/4=20M
    PWMSCLA       =   50;      // 时钟设置为SA=A/2/10=200K
    PWMSCLB       =   50;      // 时钟设置为SB=B/2/50=200K
    //电机(0通道,1通道)
    PWMCLK_PCLK0  =   1;       // 通道时钟源选择为SA
    PWMCLK_PCLK1  =   1;       // 通道时钟源选择为SA
    PWMPER0       =   101; // 0通道周期寄存器设置为Frequency=SA/100=2k
    PWMDTY0       =   0;       // 占空比寄存器设置占空比%
    PWMPER1       =   101; // 1通道周期寄存器设置为Frequency=SA/100=2k
    PWMDTY1       =   0;       // 占空比寄存器设置占空比%
    //电机(2通道,3通道)
    PWMCLK_PCLK2  =   1;       // 通道时钟源选择为SB
    PWMCLK_PCLK3  =   1;       // 通道时钟源选择为SB
    PWMPER2       =   101; // 2通道周期寄存器设置为Frequency=SB/100=2k
    PWMDTY2       =   0;       // 占空比寄存器设置占空比%
    PWMPER3       =   101; // 3通道周期寄存器设置为Frequency=SB/100=2k
    PWMDTY3       =   0;       // 占空比寄存器设置占空比%
    //电机(4通道,5通道)
    PWMCLK_PCLK4  =   1;       // 通道时钟源选择为SA
    PWMCLK_PCLK5  =   1;       // 通道时钟源选择为SA
    PWMPER4       =   101; // 4通道周期寄存器设置为Frequency=SA/100=2k
    PWMDTY4       =   0;       // 占空比寄存器设置占空比%
    PWMPER5       =   101; // 5通道周期寄存器设置为Frequency=SA/100=2k
    PWMDTY5       =   0;       // 占空比寄存器设置占空比%
    //电机(6通道,7通道)
    PWMCLK_PCLK6  =   1;       // 通道时钟源选择为SB
    PWMCLK_PCLK7  =   1;       // 通道时钟源选择为SB
    PWMPER6       =   101; // 6通道周期寄存器设置为Frequency=SB/100=2k
    PWMDTY6       =   0;       // 占空比寄存器设置占空比%
    PWMPER7       =   101; // 7通道周期寄存器设置为Frequency=SB/100=2k
    PWMDTY7       =   0;       // 占空比寄存器设置占空比%

    PWME          =   0xFF;    // 使能所有PWM通道
}
/************************************************/
```

```
/***************SCI初始化*******************/
void SCI_Init()
{
    SCI0BDH = 0x02;
    SCI0BDL = 0x08;     // BaudRate = BusClk/16/SBR[12:0] = 9615 = 9600
    SCI0CR1   = 0x00;       // 正常8 位模式,无奇偶校验
    SCI0CR2   = 0x2C;       // 接收中断使能,可发可收模式
}
/*****************************************/

/*************相关端口初始化*****************/
void PORT_Init(void)
{
  DDRA  = 0x00;
  DDRB  = 0xFF;
  PORTB = 0xFF;

  LED   = 0;
  TX0   = 0;
}
/*****************************************/

/*****************初始化所有*********************/
void Init_All(void)
{
  PLL_Init();
  PWM_Init();
  PIT_Init();
  SCI_Init();
  PORT_Init();
}
/*****************************************/
```

Main.c，主函数和相关中断函数

```
/*************************************************************
* File Name:       main.c
* Discription:          Call the Init_All function
*                       to initial all registers
*                       and enable interrupts.
*                       End by a dead loop.
* Writer:          Xiao Chongerfei
* Last Edit Time:  2012-9-14
```

```
************************************************************/
#include <hidef.h>      /* common defines and macros */
#include "derivative.h"      /* derivative-specific definitions */

#include "init.h"
#include "moto.h"

#define   LED               PORTB_PB0

unsigned char Speed         = 70;      //设定的基础速度
unsigned char dir           = 255;   //接收图像数据,方向标志
unsigned char stage         = 0;     //记录运行阶段
unsigned char flag          = 0;     //标志状态,用于丢帧处理
unsigned char flag_old      = 0;     //上次的状态
unsigned char stage_4_flag  = 0;     //第四阶段的一个标志
unsigned char sci_status    = 0;     //用于读串口状态

unsigned int  temp          = 0;     //临时变量,方便调试

unsigned long time          = 0;     //定时器计数,每ms加
unsigned long TimeSec       = 0;     //用于计数,每s加

void Delay_ms(word ms)                //延时ms
{
     word i,j;
     for(i=0; i<ms; i++)
          for(j=0; j<13350; j++);
}

void main(void)
{
    DisableInterrupts;
    Init_All();         //初始化
    Delay_ms(3000);         //延时s
    EnableInterrupts;   //开总中断
     for(;;);
}

#pragma CODE_SEG NON_BANKED

/***************定时中断******************/
void interrupt VectorNumber_Vpit0 PIT0(void)   // every 100ms
{
    PITTF_PTF0=1;   //清标志
```

```
    time++;
    if(time%10 == 0)
    {
        TimeSec++;    //每s加
        LED = ~LED;   //闪烁,系统正在运行...
        PORTB_PB4 = ~PORTB_PB4;
    }
    Moto_Control();   //控制程序
}
/*************************************/

/**************串口中断****************/
void interrupt VectorNumber_Vsci0 SCI0(void)    //串口中断
{
     PITINTE = 0x00;      // 屏蔽路定时器中断

    sci_status = SCI0SR1_RDRF;  //读接收完成标志位
    dir = SCI0DRL;              //读数据

    PITINTE   = 0x01;    // 使能路定时器中断
}
```

Moto.c，控制函数，包含主要的控制策略

```
/************************************************************
* File Name:       moto.c
* Discription:          the key controlling program
* Writer:          Xiao Chongerfei
* Last Edit Time:  2012-9-14
************************************************************/
#include <hidef.h>      /* common defines and macros */
#include "derivative.h"      /* derivative-specific definitions */

void moto_run(char SpeedL,char SpeedR); //函数声明
void Come_Up(void);

#define     TX0     PORTB_PB1   //发送指令接口,控制拿花


unsigned char  P    = 8;      //转弯速度系数

char Left  = 0;
char Right = 0;
```

```
unsigned int count_0 = 0;

extern unsigned char Speed;    //基础速度
extern unsigned int temp;      //临时变量,用于调试
extern unsigned char flag;      //标志状态
extern unsigned char flag_old;  //上次的状态
extern unsigned char stage_4_flag; //第四阶段运行的一个标志
extern unsigned char stage;       //记录运行阶段
extern unsigned char dir;         //偏差
extern void Delay_ms(word ms);

void Moto_Control(void)
{
    if(dir == 0xF1)             //下楼
    {
        //PITINTE      = 0x00;      //关闭路中断
        PITCFLMT_PITE   = 0;          //关闭路定时器
        SCIOCR2_RE      = 0;          //串口接收停止

        stage           = 0xF1;
        moto_run(30,30);
        Delay_ms(5000);          //延时s
        //Delay_ms(2000);
        dir = 255;
        moto_run(0,0);              //停s
        Delay_ms(1000);
        TX0 = 1;                    //打开爪子

        SCIOCR2_RE      = 1;             //串口接收使能
        //PITINTE         = 0x01;      //打开路中断
        PITCFLMT_PITE   = 1;          //打开路定时器

    }
    else if(dir == 0xF2)
    {
        //PITINTE    = 0x00;      //关闭路中断
        PITCFLMT_PITE = 0;      //关闭路定时器
        SCIOCR2_RE = 0;         //串口接收停止

        stage = 0xF2;
        moto_run(0,0);          //停s,准备拿花
        Delay_ms(1000);
        TX0 = 0;                //拿花
```

```
    Delay_ms(1000);          //再停s
    moto_run(-30,-30);       //后退,准备转弯
    Delay_ms(3000);
    dir = 255;

    SCI0CR2_RE = 1;          //串口接收使能
    //PITINTE     = 0x01;      //打开路中断
    PITCFLMT_PITE = 1;       //打开路定时器
}
else if(dir == 0xF3)
{
    stage = 0xF3;
    dir = 255;
}
else if(dir == 0xF4)
{
    stage = 0xF4;
    dir = 255;
}
else if(dir == 0xF5)
{
    stage = 0xF5;
    //moto_run(0,0);
    //PITCFLMT_PITE = 0;         //关闭路定时器
    //Delay_ms(1000);
    //PITCFLMT_PITE = 1;         //打开路定时器
    dir = 255;
}
else if(dir == 0xF0)
{
    stage = 0xF0;
    dir = 255;
}

if(stage != 0xF5 && temp >= 85)
    //moto_run(0,0);
    Come_Up();       //上楼

else if(!dir)                 //路标不在视野内
    {
        flag = 0;

        if((!flag)&&flag_old)   //丢帧处理
        {
```

```
    moto_run(0,0);
    //PITINTE      = 0x00;      //关闭路中断
    PITCFLMT_PITE = 0;         // 关闭路定时器
    Delay_ms(500);
    //PITINTE      = 0x01;      //打开路中断
    PITCFLMT_PITE = 1;      // 打开路定时器
}
else
{
    if(stage == 0xF1)        //第一阶段
    {
        moto_run(60,-20);    //右转,直到第一个路标出现
    }
    else if(stage == 0xF2)   //第二阶段
    {
        moto_run(-20,60);    //左转,直到第二个路标出现
    }
    else if(stage == 0xF3)   //第三阶段
    {
        moto_run(-20,60);    //左转,直到第三个路标出现
    }
    else if(stage == 0xF4)   //第四阶段
    {
        stage_4_flag = 1;
        moto_run(60,-20);    //右转,直到第四个路标出现
    }
    else if(stage == 0xF5)
    {
        PITCFLMT_PITE = 0;          //关闭路定时器
        count_0++;
        if(count_0 == 1)
        {
            moto_run(50,50);          //向前走一点
            Delay_ms(1000);
            TX0 = 1;                  //放下花,结束
        }
        moto_run(0,0);
        //PITCFLMT_PITE = 1;        // 打开路定时器
    }
    else if(stage == 0xF0)
    {
        moto_run(-20,60);
    }
}
```

```
            flag_old = flag;
        }
        else if((dir >= 26)&&(dir <= 34))  //途中......
        {
            if(stage_4_flag)temp++;
            flag  = 1;
            flag_old  = flag;
            Left  = (dir - 30)*P;
            Right = (30 - dir)*P;
            if(temp < 85 || stage == 0xF5)
                moto_run(Left + Speed,Right + Speed);
        }
        else if(dir == 255)
        {
            //Come_Up();
            moto_run(0,0);
        }
}

void moto_run(char SpeedL,char SpeedR)
{
    if(SpeedL > 99)     SpeedL = 99;        //限速
    if(SpeedL < -99)    SpeedL = -99;

    if(SpeedR > 99)     SpeedL = 99;
    if(SpeedR < -99)    SpeedL = -99;

    if(SpeedL>=0)
    {
        PWMDTY0 = SpeedL;       //电机,左前
         PWMDTY1 = 0;

         PWMDTY6 = 0;            //电机,左后
     PWMDTY7 = SpeedL;
    }
    else
    {
        PWMDTY0 = 0;            //电机,左前
        PWMDTY1 = (-SpeedL);

        PWMDTY6 = (-SpeedL);    //电机,左后
     PWMDTY7 = 0;
    }
```

```
    if(SpeedR >= 0)
    {
        PWMDTY2 = 0;            //电机,右前
     PWMDTY3 = SpeedR;

     PWMDTY4 = SpeedR;       //电机,右后
     PWMDTY5 = 0;
    }
    else
    {
        PWMDTY2 = (-SpeedR);    //电机,右前
     PWMDTY3 = 0;

     PWMDTY4 = 0;            //电机,右后
     PWMDTY5 = (-SpeedR);
    }
}

void Come_Up(void)
{
    PWMDTY0 = 55;       //电机,左前
     PWMDTY1 = 0;

     PWMDTY2 = 0;        //电机,右前
    PWMDTY3 = 55;

    PWMDTY6 = 0;        //电机,左后
    PWMDTY7 = 95;

    PWMDTY4 = 100;       //电机,右后
    PWMDTY5 = 0;
}
```

第四章 图像处理

4.1 概述

今年 robogame 首次将场地移到室外进行，并且使用图像处理寻找色柱定位，对图像处理提出了新的要求。随着 robogame 的继续进行，相信图像部分会越来越成为比赛的焦点之一。其广泛的用途和对综合素质的考察也让我们不得不重视图像方面的学习与研究。未来必将会有难度大的多的题目，故而先抛砖引玉，将自己在做图像处理时对处理方法的感悟记录下来。

4.2 图像基本知识。

我们处理的图像是摄像头捕捉到的图像，其本质是由一系列像素点构成。图像有很多种表达形式，相对应的很多色彩空间。常用的是 bgr 空间和 hsv 空间。摄像头直接捕捉到的是一个三通道 bgr 图。30 万像素的摄像头捕捉到的是 640×480 的三维数组，每个点包含三组值，分别为 bgr 三者，范围是 0~255。可以通过变换把这三组数值变为 hsv 空间的三组数值。Bgr 的好处是直观。Hsv 的好处是 h 分量对光不敏感。

4.3 图像处理工具。

Opencv 是处理图像的最佳选择之一。

4.4 工具的使用。

Opencv 为我们提供了很多库函数。但是，我所用到的，也是我认为 opencv 对于我们的最大价值在于，它提供了图像的数据结构 IplImage。用 opencv 只是一个获取摄像头信息的方式，当把一幅图像的三维坐标取出时，暴露在面前的只剩下数据而已。对数据的处理才是程序中最重要的算法层面。工具总是为算法服务的。

4.5 调试过程的感悟与经验技巧。

4.5.1 算法：

把图像横向分为 320 等份，计算每条上从上倒下所有符合 b，g，r 的数目，如果三者的数目都大于特定值且三者的纵坐标中心位置排列顺序符合当前柱子颜色顺序时，认为找到目标。

4.5.2 待调试参数：

B，g，r 三者的阈值。

等分的份数。

算法中的“特定值”。

等。

最重要的必然是 bgr 的阈值。

4.5.3 我的调试过程如下：

4.5.3.1 确定需要的阈值种类。

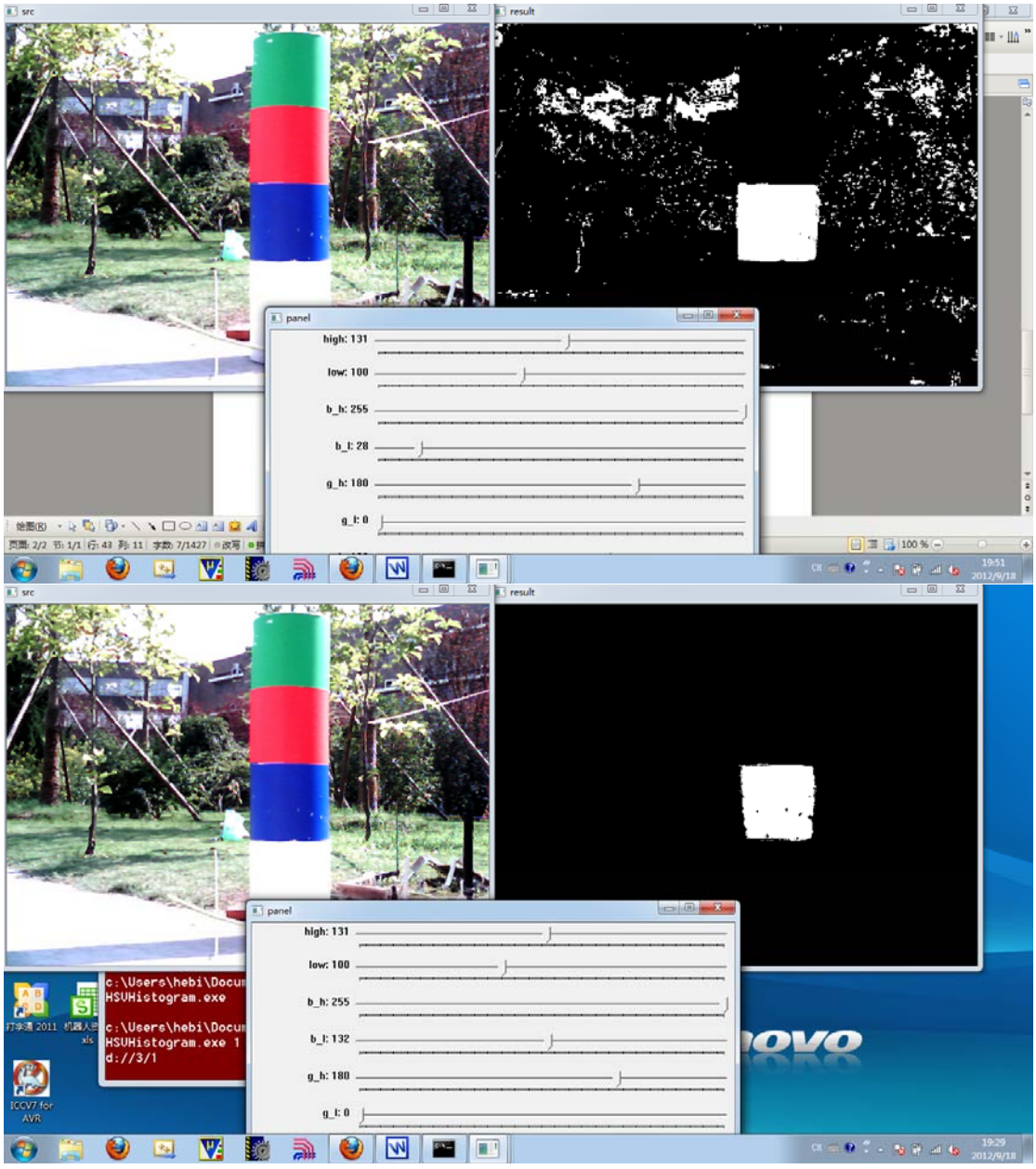
Bgr 直观但易受到天气的影响。Hsv 单独使用背景噪声太大。故选择

bgr 配合 hsv 的参数组合。每种颜色有六种参数，每种参数有上限和下限，故每种颜色需要调整的参数有 12 个。这是相当多的。故而需要很好的调试手段。如下所述。

4.5.3.2 优化调试工具。

工欲善其事，必先利其器。当面临 12×3=36 个参数的阈值调整，显然不能每次在程序中改变参数，然后编译运行监测效果。使用 thredhold 组件是很好的方法。

下图是用滑动条调试的截图。



4.5.3.3 实际视频调试。

先拍一段实际的视频，然后截图处理，对每一张图做阈值分析，调试出对于每张图都有很好识别效果而且噪声不是很大的参数。噪声是不可避免的，尤其是户外图像处理中。所以不要指望参数能准确无误的把所有图片的颜色都识别出来。滤波可以搞定剩下的一切。

4.5.3.4 滤波。

效果最好的滤波是在实际问题上的专门的滤波算法。不可能要求用求平均等方法把噪点去除干净。而针对本问题，上述算法已然达到上佳的滤波效果，理由是：在同一小条图像上能使 bgr 三种颜色都大于某一特定值而且顺序还按指定顺序排列，概率非常小。这样基本不用考虑噪点太多的问题。当然，还可以检测 bgr 是否连续来进一步滤波，但是无关紧要。

4.5.3.5 不同天气环境的阈值调整。

最后剩下的这项工作是最令人头疼的工作，因为你会发现在阳光下你的努力似乎有些苍白。需要做的是把所有天气都拍个视频处理，阈值调松（不可调小），宁多识别，不放过。这样才能保证识别的灵敏性。至于多的部分，滤波而已。

4.6 总结。

图像处理是一门很大的课程。只有对处理算法深入研究，调试过程兢兢业业，才能做到准确灵敏。我们组调试图像的时间略显紧张，而且调试过程中一直阴天，比赛前一天突然放晴，对阳光环境下参数的调整做到不够到位，导致在第三个路标上耽误了点时间，略感遗憾。不过对图像的认识也更深了一步。

在文档最后附上图像处理代码。上位机与下位机的协同工作协议见单片机程序部分。

附录：图像处理源码

```
#include<stdio.h>
#include<cv.h>
#include<cvaux.h>
#include<highgui.h>
#include<math.h>
#include<windows.h>


#define uchar unsigned char
#define uint unsigned int



#define DEC 320

    double fps = 20;

    char flagx = 0;
//用来遍历图像的循环计数量
    int x,y;
    int i,j,k;
    int notfound = 0;
    int start = 0;
    int lhb = 1;
    int widthflag = 0;
    int on = 0;

    int found = 0;
    int oning = 0;
    bool foundornot = 0;
    bool oningornot = 0;

    int mid1 = 320;
    int mid2 = 320;
    int mid3 = 320;
    int mid4 = 320;

    bool flag1 = 1;
    bool flag2 = 1;
    bool flag3 = 1;
    bool flag4 = 1;
```

```
    bool flag5 = 1;

    bool _BGR = 0;
    bool _BRG = 0;
    bool _GBR = 0;
    bool _GRB = 0;
    bool _RBG = 0;
    bool _RGB = 0;
    bool ok = 0;

    int b_num=0, g_num=0, r_num=0;

    int com = 8;

    double b_y = 0;
    double g_y = 0;
    double r_y = 0;
int thred_r_h_h = 255;
int thred_r_h_l = 0;
int thred_r_b_l = 0;
int thred_r_b_h = 160;
int thred_r_g_l = 0;
int thred_r_g_h = 160;
int thred_r_r_l = 120;
int thred_r_r_h = 255;
int thred_r_s_h = 255;
int thred_r_s_l = 115;
int thred_r_v_h = 255;
int thred_r_v_l = 125;

int thred_g_h_h = 95;
int thred_g_h_l = 60;
int thred_g_b_l = 0;
int thred_g_b_h = 255;
int thred_g_g_l = 50;
int thred_g_g_h = 255;
int thred_g_r_l = 0;
int thred_g_r_h = 143;
int thred_g_s_h = 255;
int thred_g_s_l = 130;
int thred_g_v_h = 255;
int thred_g_v_l = 47;

int thred_b_h_h = 128;
```

```
int thred_b_h_l = 105;
int thred_b_b_l = 25;
int thred_b_b_h = 255;
int thred_b_g_l = 0;
int thred_b_g_h = 180;
int thred_b_r_l = 0;
int thred_b_r_h = 160;
int thred_b_s_h = 255;
int thred_b_s_l = 130;
int thred_b_v_h = 255;
int thred_b_v_l = 0;
#endif

    CvSize g_size;
//函数声明
    void myHandle(IplImage *src, IplImage* frame);
    //dat 可以是:
    //0x66 ------ b
    //int a = 102 ----- b  (0x66 的十进制数)
    //uchar a = 0x66  ----- b
    int sentToUART(uchar dat, int comnum);
//  void get_loc_rec(IplImage *src);
    void getloc(IplImage *frame, IplImage* hue, int bgr);
    bool B(uchar* frame_ptr, uchar* hsv_ptr, int pos);
    bool G(uchar* frame_ptr, uchar* hsv_ptr, int pos);
    bool R(uchar* frame_ptr, uchar* hsv_ptr, int pos);
//主函数
int main(int argc, char** argv)
{
    IplImage* frame;
    CvCapture* capture;
    uint key = 0;
    IplImage* hsv;
    cvNamedWindow("frame");
    capture = cvCaptureFromCAM(1);
    //capture = cvCaptureFromFile("d://5/r.avi");
    CvSize  size  =  cvSize((int  )cvGetCaptureProperty(capture,
CV_CAP_PROP_FRAME_WIDTH),
    (int)cvGetCaptureProperty( capture, CV_CAP_PROP_FRAME_HEIGHT));
    CvVideoWriter * writer;
    if(argc == 3)
        writer          =          cvCreateVideoWriter(argv[2],
CV_FOURCC('M','J','P','G'), fps, size, 1);
```

```
    frame = cvQueryFrame(capture);
    //frame = cvLoadImage("d://2/a.jpg");
    flagx = argv[1][0];
    if(argv[1][0] == '2')
    {
        flag1 = 0;
//      flag22 = 1;
    }
    if(argv[1][0] == '3')
    {
        flag1 = 0;
        flag2 = 0;
    }
    if(argv[1][0] == '4')
    {
        flag1 = 0;
        flag2 = 0;
        flag3 = 0;
    }
    g_size = cvGetSize(frame);
    hsv = cvCreateImage(g_size, 8, 3);
    while(1)
    {
        //time1 = GetTickCount();
        frame = cvQueryFrame(capture);
        if(argc == 3)
        {//printf("hello");
            cvWriteFrame(writer, frame);
        }
        if(flag3 == 0)
        {
thred_r_h_h = 255;
thred_r_h_l = 0;
thred_r_b_l = 0;
thred_r_b_h = 210;
thred_r_g_l = 0;
thred_r_g_h = 210;
thred_r_r_l = 120;
thred_r_r_h = 255;
thred_r_s_h = 255;
thred_r_s_l = 50;
thred_r_v_h = 255;
thred_r_v_l = 172;
```

```
thred_g_h_h = 95;
thred_g_h_l = 60;
thred_g_b_l = 0;
thred_g_b_h = 255;
thred_g_g_l = 50;
thred_g_g_h = 255;
thred_g_r_l = 0;
thred_g_r_h = 193;
thred_g_s_h = 255;
thred_g_s_l = 70;
thred_g_v_h = 255;
thred_g_v_l = 47;

thred_b_h_h = 128;
thred_b_h_l = 100;
thred_b_b_l = 25;
thred_b_b_h = 255;
thred_b_g_l = 0;
thred_b_g_h = 180;
thred_b_r_l = 0;
thred_b_r_h = 160;
thred_b_s_h = 255;
thred_b_s_l = 130;
thred_b_v_h = 255;
thred_b_v_l = 80;
        }

        cvCvtColor(frame, hsv, CV_BGR2HSV);
        myHandle(frame, hsv);
//1 bgr    2 brg  3 gbr 4 grb   5 rbg 6 rgb
        cvShowImage("frame", frame);
        key = cvWaitKey(6);
        if(key==27)              //按 esc 退出
        {
            return 0;
        }
        //printf("%d\n", GetTickCount()-time1);
    }
    cvReleaseCapture(&capture);
    return 0;
}

void myHandle(IplImage *frame, IplImage * hsv)
{
```

```
//1 bgr    2 brg  3 gbr 4 grb   5 rbg 6 rgb
if(flag1)
{
    if(lhb == 1)
    {
        sentToUART(0xf1, com);  //阶段 1 开始
        cvWaitKey(10);
        lhb = 0;
        start = 0;
        printf("/t/t1\n");
    }
// printf("hello");
    getloc(frame, hsv, 4);
}
else if(flag2)
{
    /*if(flagx == '2')
    {
        lhb = 0;
        flagx = 0;
    }*/
    if(flagx == '2')
    {
        sentToUART(0xf0, com);
        cvWaitKey(10);

        flagx = 0;
        printf("/t/t0\n");
        lhb = 1;
    }

    if(lhb == 0)
    {
        sentToUART(0xf2, com);
        cvWaitKey(10);
        lhb = 1;
        start = 0;
        printf("\t\t2\n");
    }
    getloc(frame, hsv, 1);
}
else if(flag3)
{
    if(flagx == '3')
```

```
            {
                lhb = 1;
                flagx = 0;
            }
            if(lhb == 1)
            {
                sentToUART(0xf3, com);
                cvWaitKey(10);
                lhb = 0;
                start = 0;
                printf("\t\t3\n");
            }
            getloc(frame, hsv, 6);
        }
        else if(flag4)
        {
            if(flagx == '4')
            {
                lhb = 0;
                flagx = 0;
            }
            if(lhb == 0)
            {
                printf("\t\t4\n");
                sentToUART(0xf4, com);
                cvWaitKey(10);
                lhb = 1;
                start = 0;
            }
            getloc(frame, hsv, 5);
        }
    }

    bool B(uchar* frame_ptr, uchar* hsv_ptr, int pos)
    {
        if( hsv_ptr[3*pos+0]       >=thred_b_h_l  &&  hsv_ptr[3*pos+0]
<=thred_b_h_h
            &&  hsv_ptr[3*pos+1]    >=thred_b_s_l  &&  hsv_ptr[3*pos+1]
<=thred_b_s_h
            &&  hsv_ptr[3*pos+2]    >=thred_b_v_l  &&  hsv_ptr[3*pos+2]
<=thred_b_v_h
            &&            frame_ptr[3*pos+0]>=thred_b_b_l            &&
frame_ptr[3*pos+0]<=thred_b_b_h
            &&            frame_ptr[3*pos+1]>=thred_b_g_l            &&
```

```
frame_ptr[3*pos+1]<=thred_b_g_h
          &&          frame_ptr[3*pos+2]>=thred_b_r_l          &&
frame_ptr[3*pos+2]<=thred_b_r_h
          )
          return TRUE;
       else
          return FALSE;
   }
   bool G(uchar* frame_ptr, uchar* hsv_ptr, int pos)
   {
       if(frame_ptr[3*pos+0]                    >=thred_g_b_l      &&
frame_ptr[3*pos+0]<=thred_g_b_h
          &&          frame_ptr[3*pos+1]>=thred_g_g_l          &&
frame_ptr[3*pos+1]<=thred_g_g_h
          &&          frame_ptr[3*pos+2]>=thred_g_r_l          &&
frame_ptr[3*pos+2]<=thred_g_r_h
          &&  hsv_ptr[3*pos+0]   >=thred_g_h_l  &&  hsv_ptr[3*pos+0]
<=thred_g_h_h
          && (hsv_ptr[3*pos+0] <= 10 || hsv_ptr[3*pos+0] >= 30)
          &&  hsv_ptr[3*pos+1]   >=thred_g_s_l  &&  hsv_ptr[3*pos+1]
<=thred_g_s_h
          &&  hsv_ptr[3*pos+2]   >=thred_g_v_l  &&  hsv_ptr[3*pos+2]
<=thred_g_v_h
          )
          return TRUE;
       else
          return FALSE;
   }
   bool R(uchar* frame_ptr, uchar* hsv_ptr, int pos)
   {
       if(frame_ptr[3*pos+0]                    >=thred_r_b_l      &&
frame_ptr[3*pos+0]<=thred_r_b_h
          &&          frame_ptr[3*pos+1]>=thred_r_g_l          &&
frame_ptr[3*pos+1]<=thred_r_g_h
          &&          frame_ptr[3*pos+2]>=thred_r_r_l          &&
frame_ptr[3*pos+2]<=thred_r_r_h
          &&  hsv_ptr[3*pos+0]   >=thred_r_h_l  &&  hsv_ptr[3*pos+0]
<=thred_r_h_h
          &&  hsv_ptr[3*pos+1]   >=thred_r_s_l  &&  hsv_ptr[3*pos+1]
<=thred_r_s_h
          &&  hsv_ptr[3*pos+2]   >=thred_r_v_l  &&  hsv_ptr[3*pos+2]
<=thred_r_v_h
          )
          return TRUE;
```

```
        else
            return FALSE;
    }
    //1 bgr    2 brg  3 gbr 4 grb   5 rbg 6 rgb
    void getloc(IplImage *frame, IplImage* hsv, int bgr)//bgr 为要寻找的柱子顺序（从上到下）
    {
        uchar* hsv_ptr;
        uchar* frame_ptr;
        uchar* result_ptr;
        int yes[DEC] = {0};
        int num = 0;
        int count = 0;
        int interval = 640/DEC;
        int center = 0;
        int centernum = 0;
        for(num=0;num<DEC;num++)
        {
            for(y=0;y<frame->height;y++)
            {
                frame_ptr                                                    =
(uchar*)(frame->imageData+y*frame->widthStep);
                hsv_ptr = (uchar*)(hsv->imageData+y*hsv->widthStep);
                for(x=num*interval;x<(num+1)*interval;x++)
                {
                    if(B(frame_ptr, hsv_ptr, x))
                    {
                        b_num++;
                        b_y += y;

                    }

                    if(G(frame_ptr, hsv_ptr, x))
                    {
                        g_num++;
                        g_y += y;
                    }

                    if(R(frame_ptr, hsv_ptr, x))
                    {
                        r_num++;
                        r_y += y;
                    }
                }
```

```
            }
            b_y /= b_num;
            g_y /= g_num;
            r_y /= r_num;
            if(b_num>=20 && g_num>=20 && r_num>=20)          //      参
数!!!!!!!!!!!!!!!!!!!!!!!
            {
                switch(bgr)
                {
                case 1:{if(b_y<g_y&&g_y<r_y)    ok = 1;break;}    //bgr
                case 2:{if(b_y<r_y&&r_y<g_y)    ok = 1;break;}    //brg
                case 3:{if(g_y<b_y&&b_y<r_y)    ok = 1;break;}    //gbr
                case 4:{if(g_y<r_y&&r_y<b_y)    ok = 1;break;}    //grb
                case 5:{if(r_y<b_y&&b_y<g_y)    ok = 1;break;}    //rbg
                case 6:{if(r_y<g_y&&g_y<b_y)    ok = 1;break;}    //rgb
                }
                if(ok)
                {
                    center += (num*interval);
                    centernum++;
                    yes[num] = 1;
                    cvRectangle(frame,     cvPoint(num*interval,     0),
cvPoint((num+1)*interval, 479), cvScalar(97, 0, 255));
                }
            }
            ok = 0;
            b_num = 0;
            g_num = 0;
            r_num = 0;
            b_y = 0;
            g_y = 0;
            r_y = 0;
        }
    // printf("%d\n", centernum);
        if(centernum == 0)// && start >= 10)
        {
            found = 0;
            on = 0;
            if(flag3 == 0 && foundornot == 1)
            {
                oning ++;
                if(oning>=10 && !oningornot)
                {
                    oningornot = 1;
```

```
                printf("oning\n");
            }
        }
        notfound++;
        if(notfound>=5 && start >= 10 && widthflag == 1)
        {
            widthflag = 0;
            notfound = 0;
            if(flag1 == 1)
                flag1 = 0;
            else if(flag2 == 1)
                flag2 = 0;
            else if(flag3 == 1)
                flag3 = 0;
            //else if(flag4 == 1)
            //  flag4 = 0;
        }
        else
        {
            sentToUART(0x00, com);
            //printf("0\n");
        }
    }
    if(centernum != 0)
    {
        //oning = 0;
        if(flag3 == 0 && !foundornot)
        {
            found ++;
            if(found >= 20)
            {
                foundornot = 1;
                printf("found\n");
            }
        }

        if(oningornot == 1 && flag5 == 1)            //上楼梯与
否!!!!!!!!!!!!!!!!!!!!!!!!!
        {
            //if(centernum >= 24)
            //{
            //  on ++;
            //}
            //else
```

```
                //{
                // on = 0;
                //}
                on ++;
                if(on >= 20)
                {
                    printf("on\n");
                    oningornot = 0;
                    flag5 = 0;
                    sentToUART(0xf5, com);
                    cvWaitKey(10);
                    printf("\t\t5\n");
                }
            }
            if(centernum >= 20)      //参数!!!!!!!!!!!!!!!!!!!!!!!!!!!!
                widthflag = 1;
            start++;
            notfound = 0;
            center /= centernum;
            center += interval/2;
            if(flag1 == 1)
                center = (center-mid1)/50+30;      //3      个      参
数!!!!!!!!!!!!!!!!!!!!!!!!!
            else if(flag2 == 1)
                center = (center-mid2)/50+30;
            else if(flag3 == 1)
                center = (center-mid3)/50+30;
            else if(flag4 == 1)
                center = (center-mid4)/50+30;
        }
        sentToUART(center, com);
    }



    int sentToUART(uchar dat, int comnum)
    {
        HANDLE hCom;
        unsigned char buf[1];  //要传输串口数据缓冲区
        char com[5] = {"COM"};
        com[4] = '\0';
        switch(comnum)
        {
        case 1:{com[3] = '1';break;}
```

```
case 2:{com[3] = '2';break;}
case 3:{com[3] = '3';break;}
case 4:{com[3] = '4';break;}
case 5:{com[3] = '5';break;}
case 6:{com[3] = '6';break;}
case 7:{com[3] = '7';break;}
case 8:{com[3] = '8';break;}
case 9:{com[3] = '9';break;}
}

//把串口作为文件打开

hCom=CreateFile( com, //L"COM6", //com6 口

    GENERIC_READ|GENERIC_WRITE, //读\写

    0,
    NULL,
    OPEN_EXISTING,  //已存在文件[只能这个]
    0,
    NULL
);

if (hCom==(HANDLE)-1)//打开串口失败返回
{

    return 0;
}

COMMTIMEOUTS TimeOuts;  //串口超时结构体设置

TimeOuts.ReadIntervalTimeout=1000;        //填充 TimeOuts
TimeOuts.ReadTotalTimeoutMultiplier=500;
TimeOuts.ReadTotalTimeoutConstant=5000;
TimeOuts.WriteTotalTimeoutMultiplier=500;
TimeOuts.WriteTotalTimeoutConstant=2000;

if(!SetCommTimeouts(hCom,&TimeOuts))  //与串口绑定 失败返回
    return 0;

DCB dcb;
if(!GetCommState(hCom,&dcb)) //获取当前 DCB
    return 0;
dcb.BaudRate = CBR_9600;   //波特率设置 19200
```

```
        dcb.ByteSize = 8;             //数据位 8 位

        dcb.Parity = NOPARITY;
        dcb.StopBits = ONESTOPBIT;  //1 个停止位

        dcb.fBinary=1;
        dcb.fParity=0;

        if (!SetCommState(hCom, &dcb))  //DCB 绑定失败返回
            return 0;

        if (!SetupComm(hCom,1024,1024))//设置串口读写缓冲区[1024]byte
失败返回
        {
            return 0;
        }

        PurgeComm(hCom, PURGE_TXABORT | PURGE_RXABORT | PURGE_TXCLEAR |
PURGE_RXCLEAR);
        DWORD ComWriteReturn;

        buf[0] = dat;
        WriteFile(hCom, &buf, 1, &ComWriteReturn, NULL);
        //printf("hi\n");
        CloseHandle(hCom);
    }
```