铁人战队

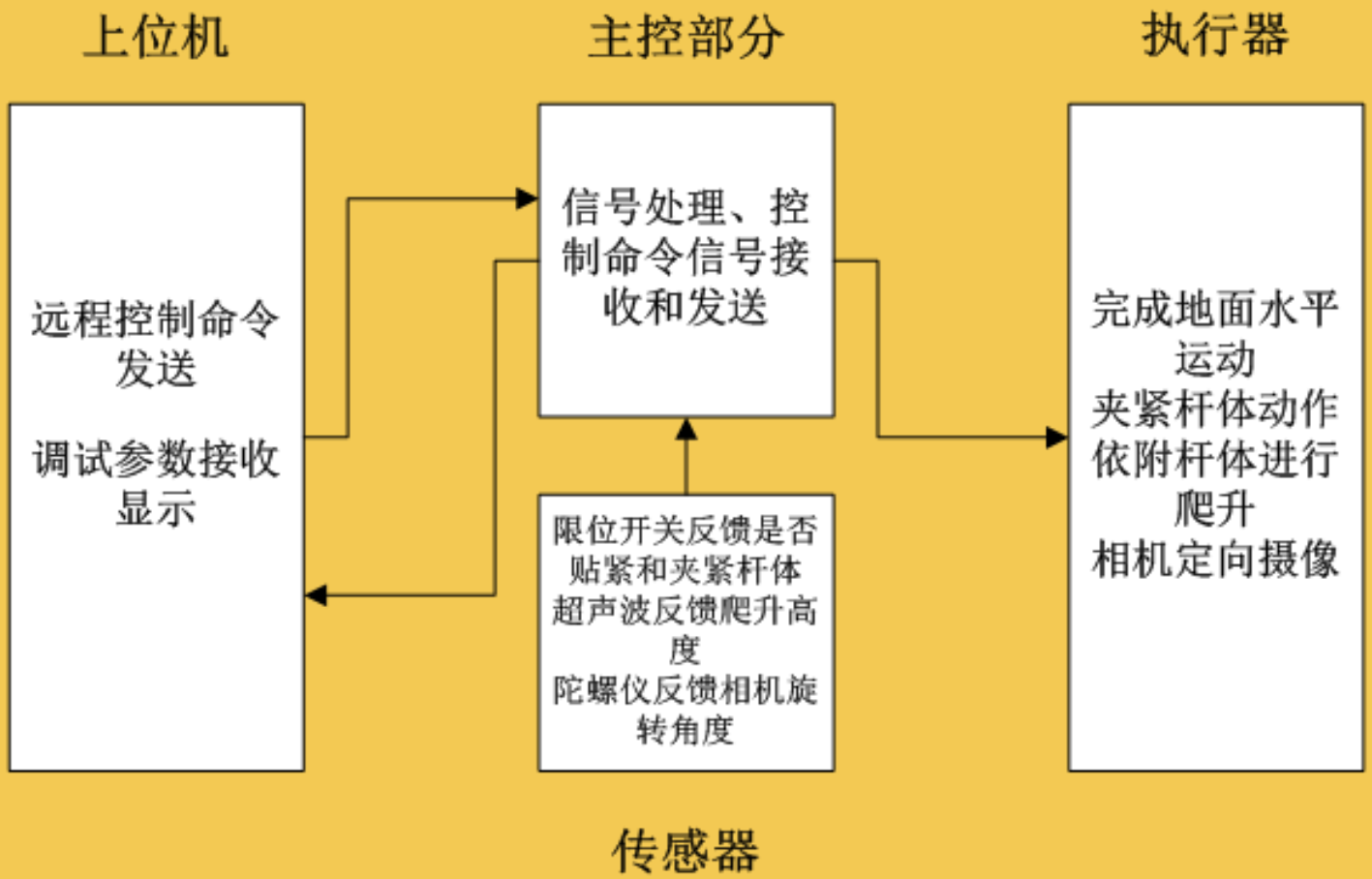
特色奖—最牛哨兵机器人

出品单位：西南石油大学

目 录

1 系统方案设计

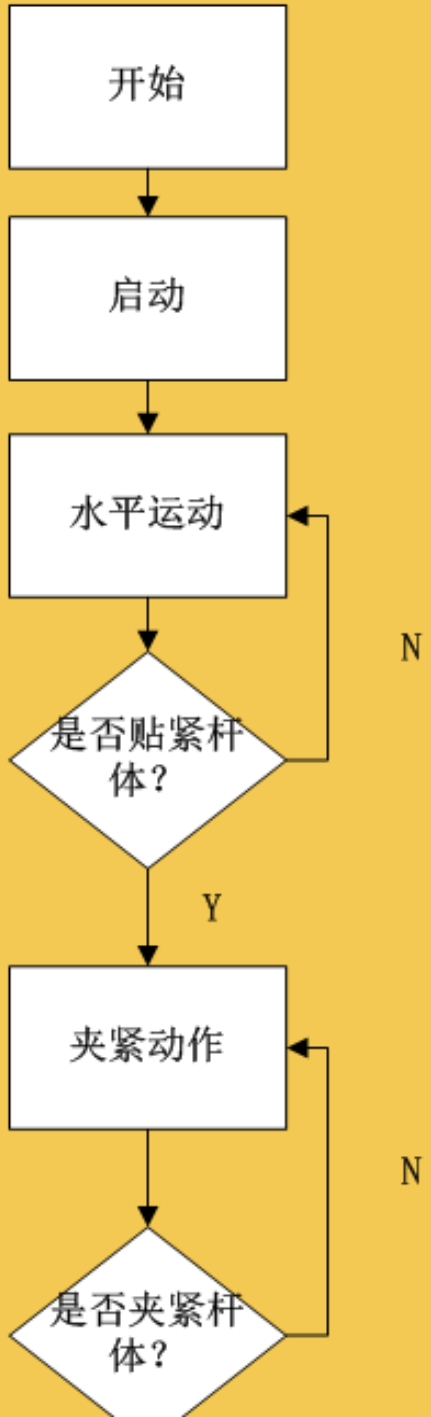
- 系统总体方案设计
- 硬件方案设计
- 软件方案设计
- 控制方案设计

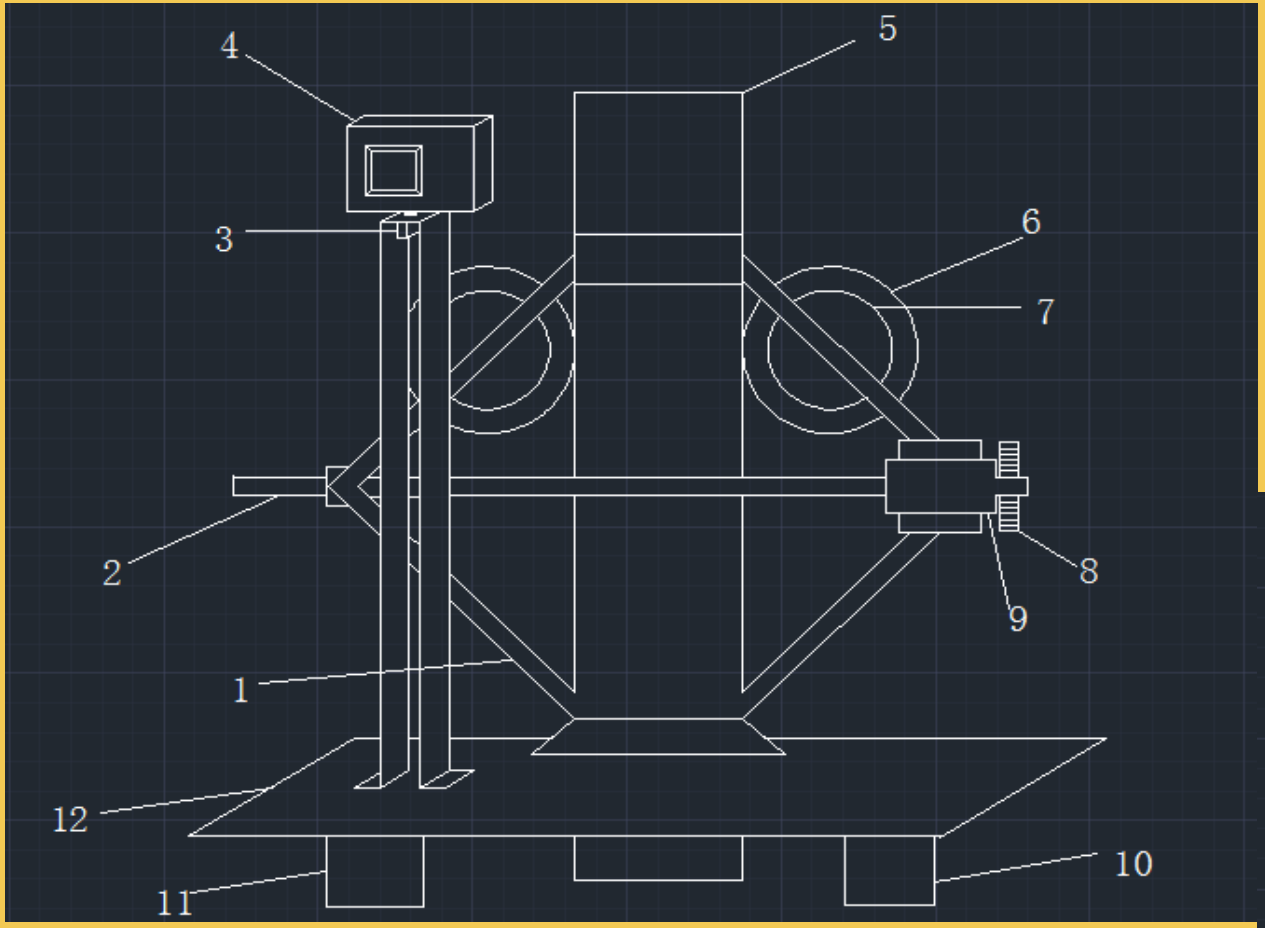


系统总体设计方案图

- **整体系统由主控部分、上位机、执行器、传感器、剪式千斤顶、小车、运动摄像机、无线图传、电源等组成。**

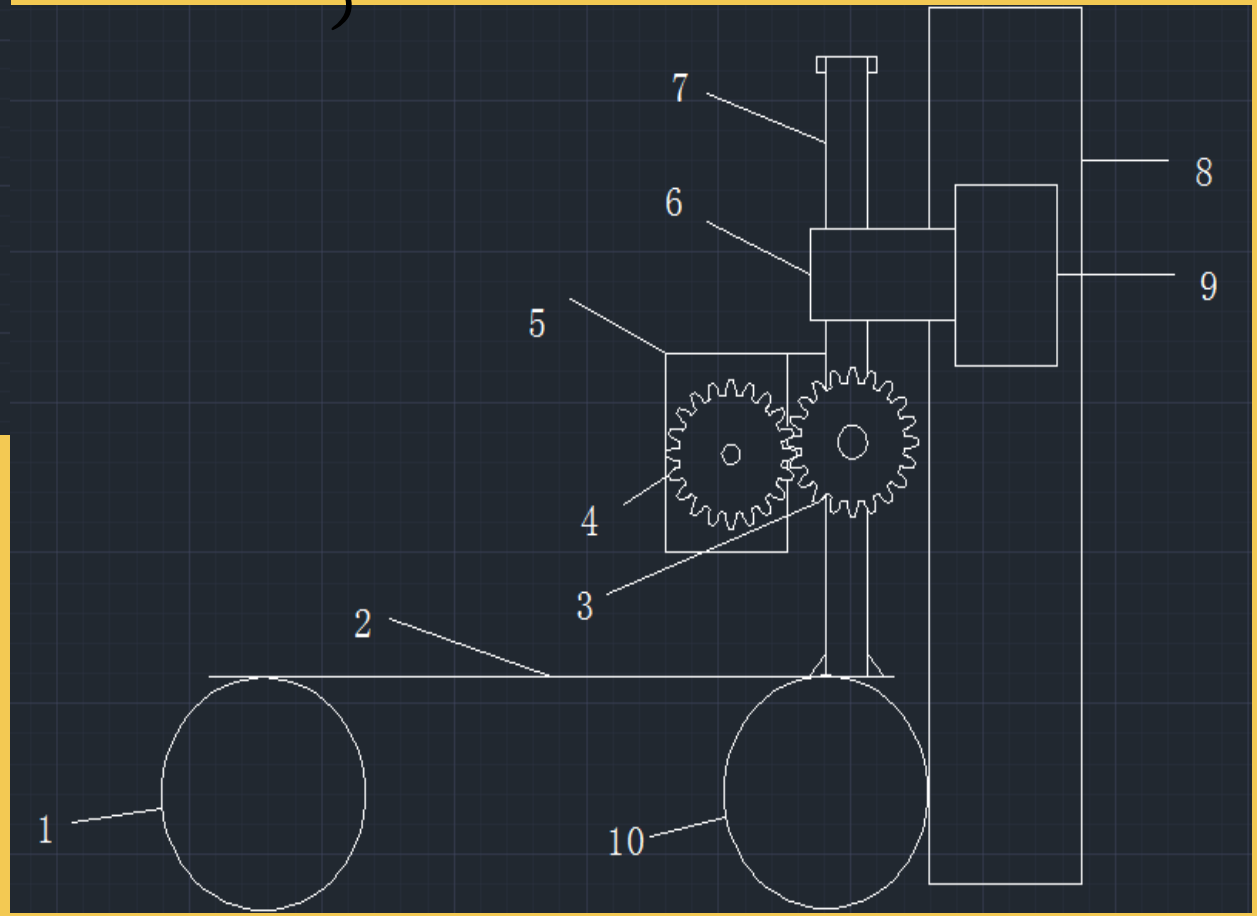
系统运行流程图



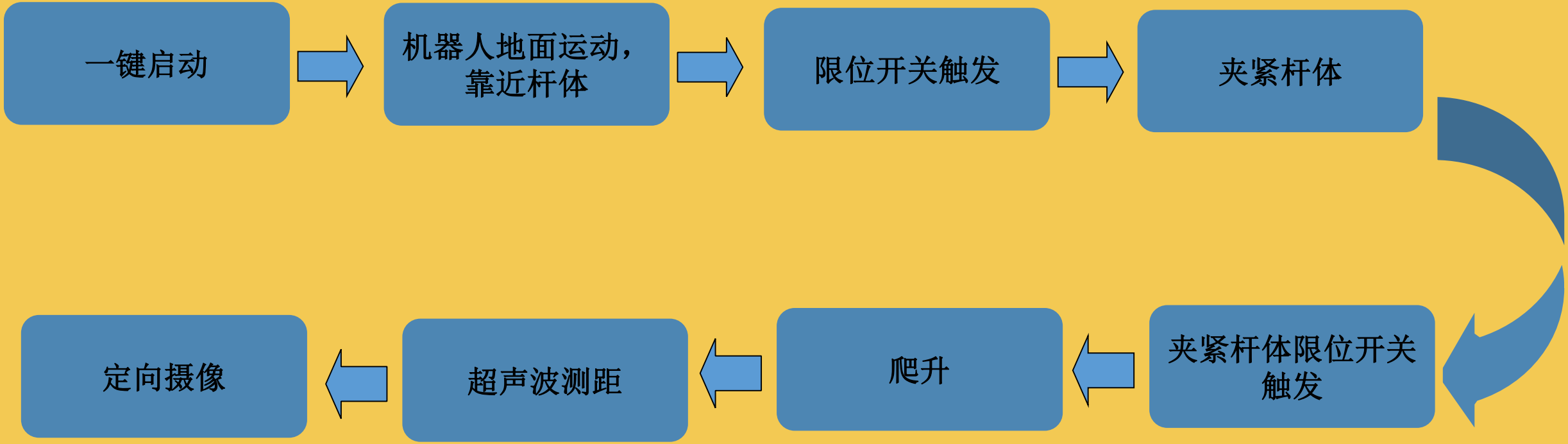


机械结构设计（主视图）

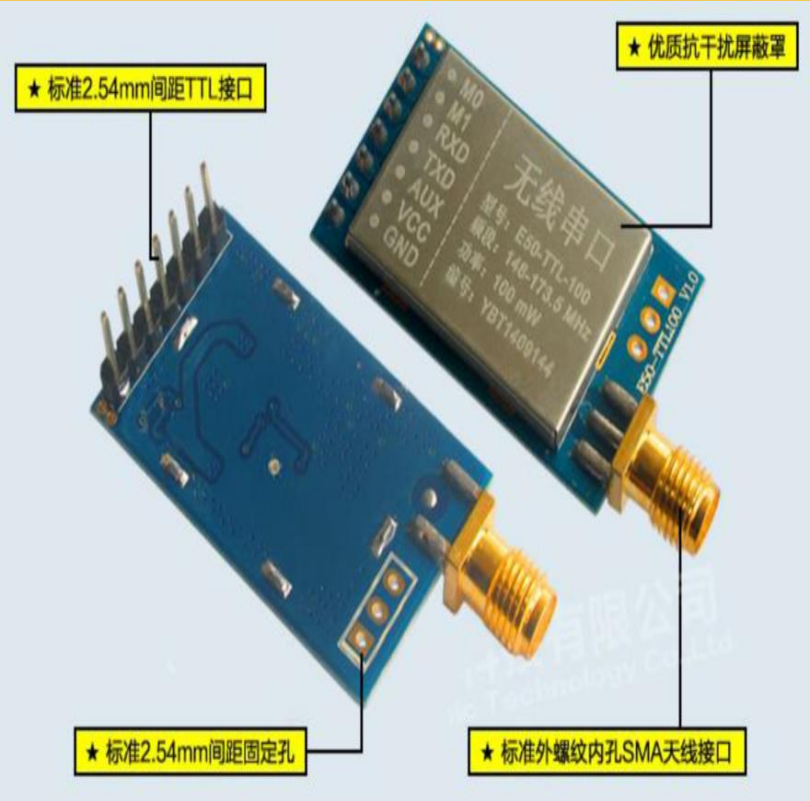
机械结构设计（右视图）



- **硬件机械部分是基础！**



- **软件部分是血液！运动分解为水平动作、夹紧动作、爬升动作、相机定向动作四个部分。**



无线串口模块

● 控制部分，乃是头脑！选择用无线串口远程控制，发送控制命令，并利用其设计一较为简易的上位机，接收显示调试参数。

E50-TTL-100 工作模式

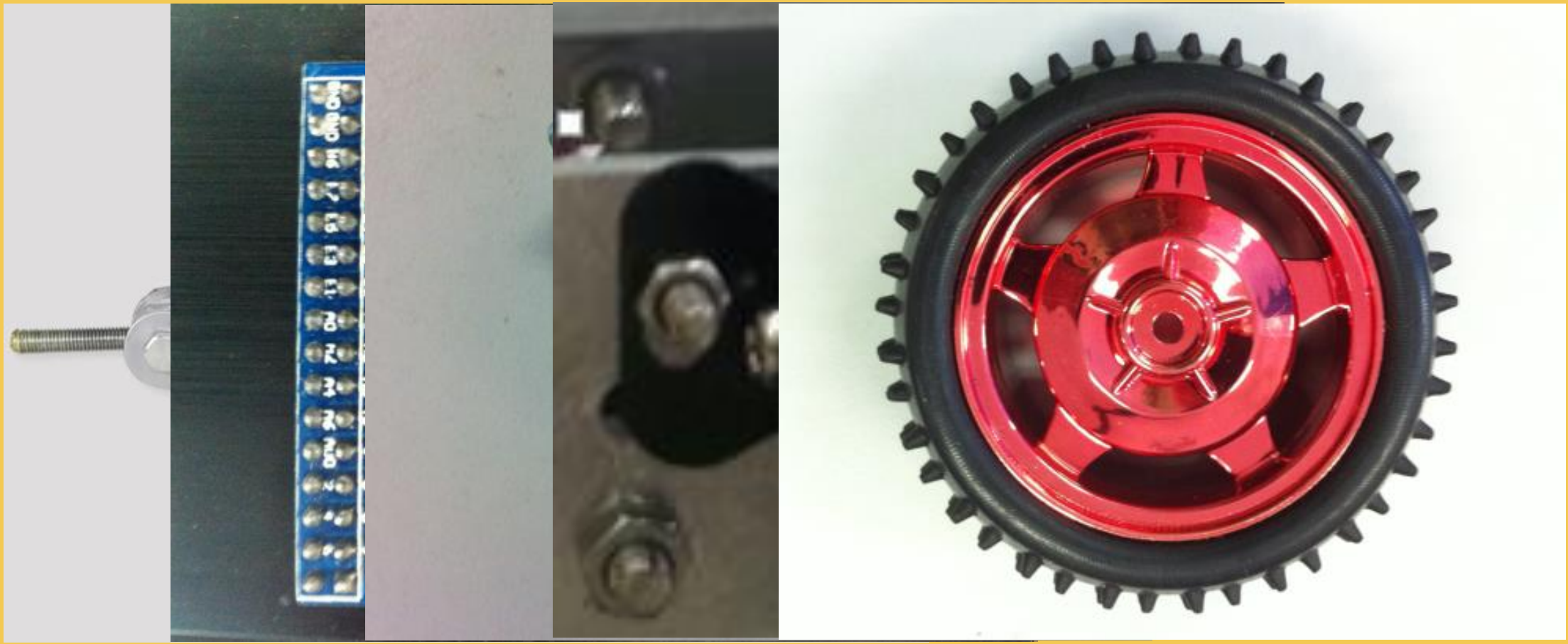
模式（0-3）	M1	M2	模式介绍	备注
0：一般模式	0	0	串口打开，无线打开，透明传输。	接收方必须是模式 0、1。
1：唤醒模式	0	1	串口打开，无线打开，和模式 0 唯一区别：数据包发送之前，自动增加唤醒码，这样才能唤醒工作方式 2 的接收方。	接收方必须是模式 0、1、2。
2：省电模式	1	0	串口接收关闭，无线处于空中唤醒模式，接收到无线数据后，打开串口发出数据。	发射方必须是模式 1，该模式下不能发射。
3：休眠模式	1	1	模块进入休眠，可以接收参数设置命令。	

E50-TTL-100 引脚功能描述

引脚序号	引脚名称	引脚方向	引脚用途
1	M0	输入（极弱上拉）	和 M1 配合，决定模块的四种工作模式，不可悬空。
2	M1	输入（极弱上拉）	和 M0 配合，决定模块的四种工作模式，不可悬空。
3	RXD	输入	TTL 串口输入，连接到外部 TXD 输出引脚，可配置为漏极开路或上拉输入。
4	TXD	输出	TTL 串口输出，连接到外部 RXD 输入引脚，可配置为漏极开路或推挽输出。
5	AUX	输出	用于指示模块工作状态，用户唤醒外部 MCU，上电自检初始化期间输出高电平，可配置为漏极开路或推挽输出。
6	VCC		模块电源正极，电压范围：2.1V-5.5V DC
7	GND		模块地线

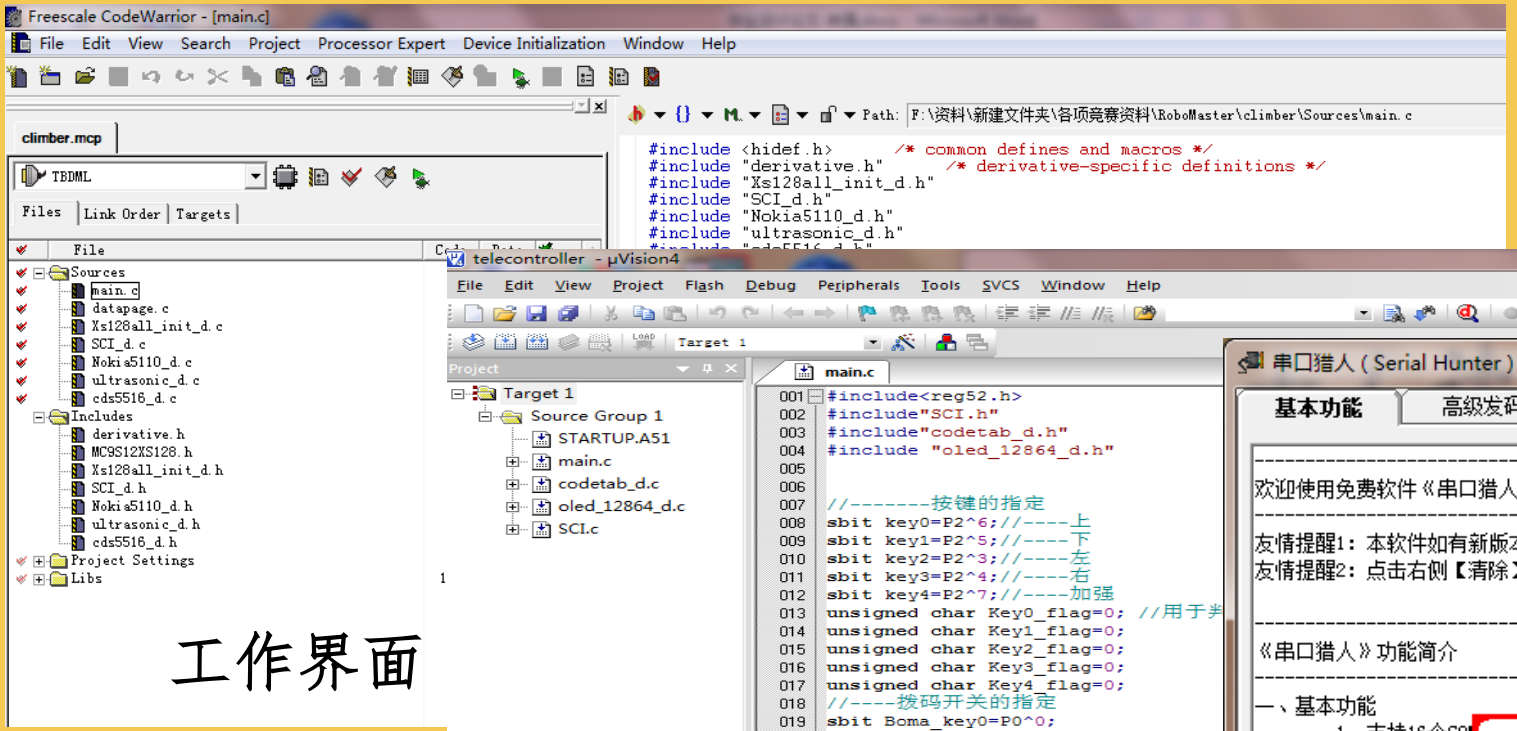
2 方案论证分析

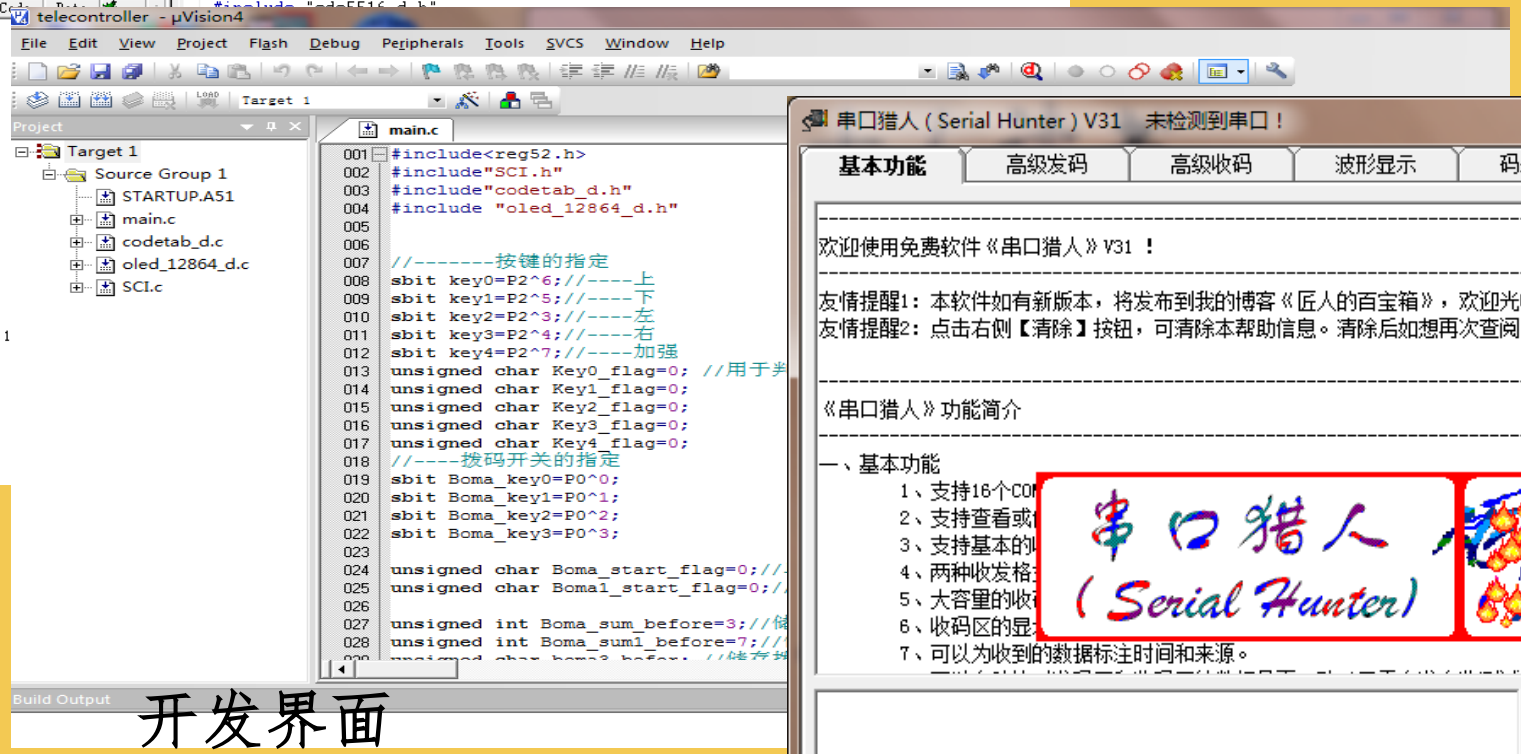
- 硬件设计方案论证分析
- 软件设计方案论证分析

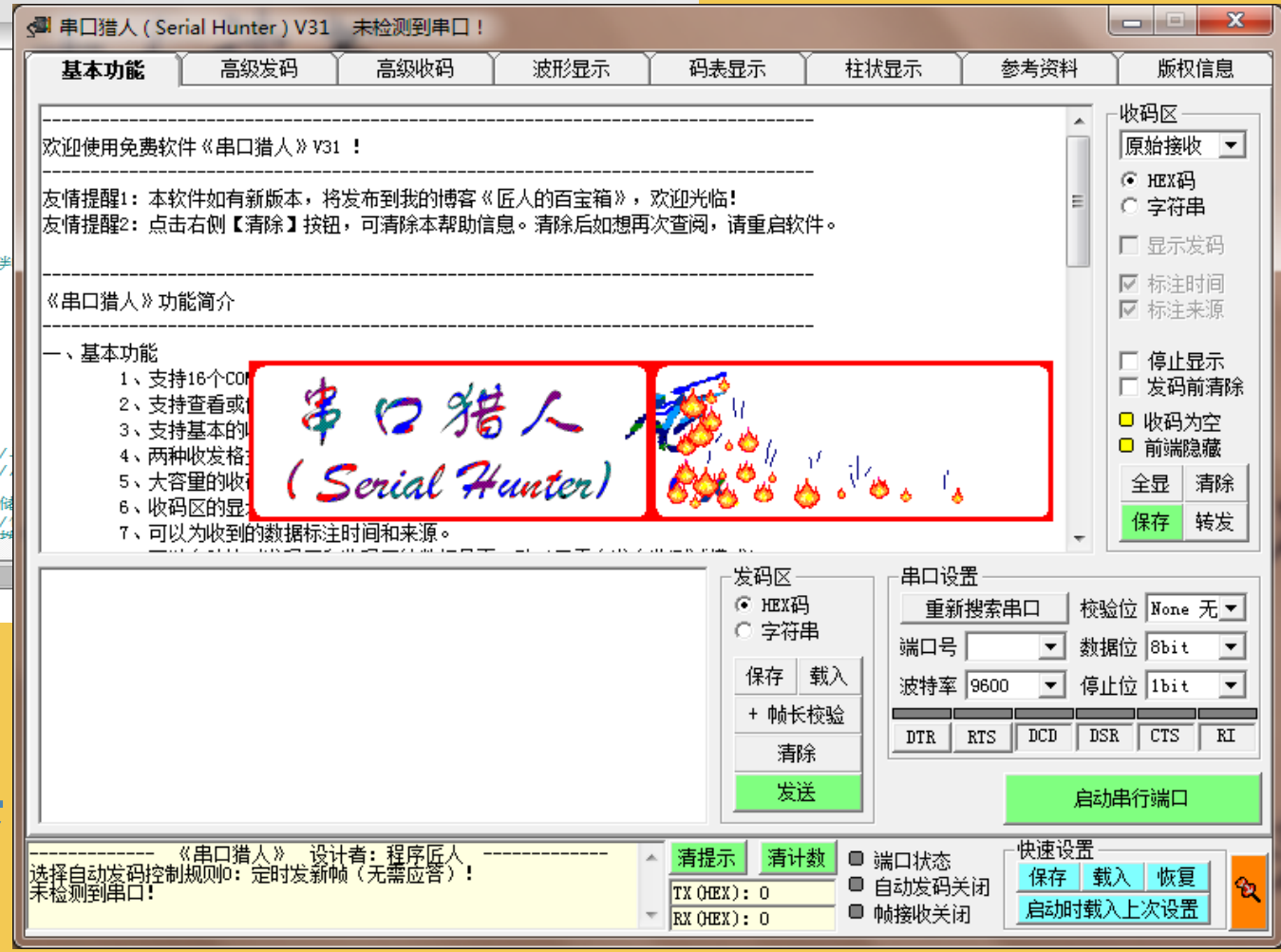


剪式千斤顶　XS128单片机最小系统板　最小系统板　水平和爬升运动动力车轮

- 硬件方面主要包括夹紧变形结构选择论证分析，主控芯片论证分析，水平和爬升运动论证分析，贴紧、夹紧杆体和爬升高度论证定位分析，相机定向摄像论证分析，无线图像传输论证分析





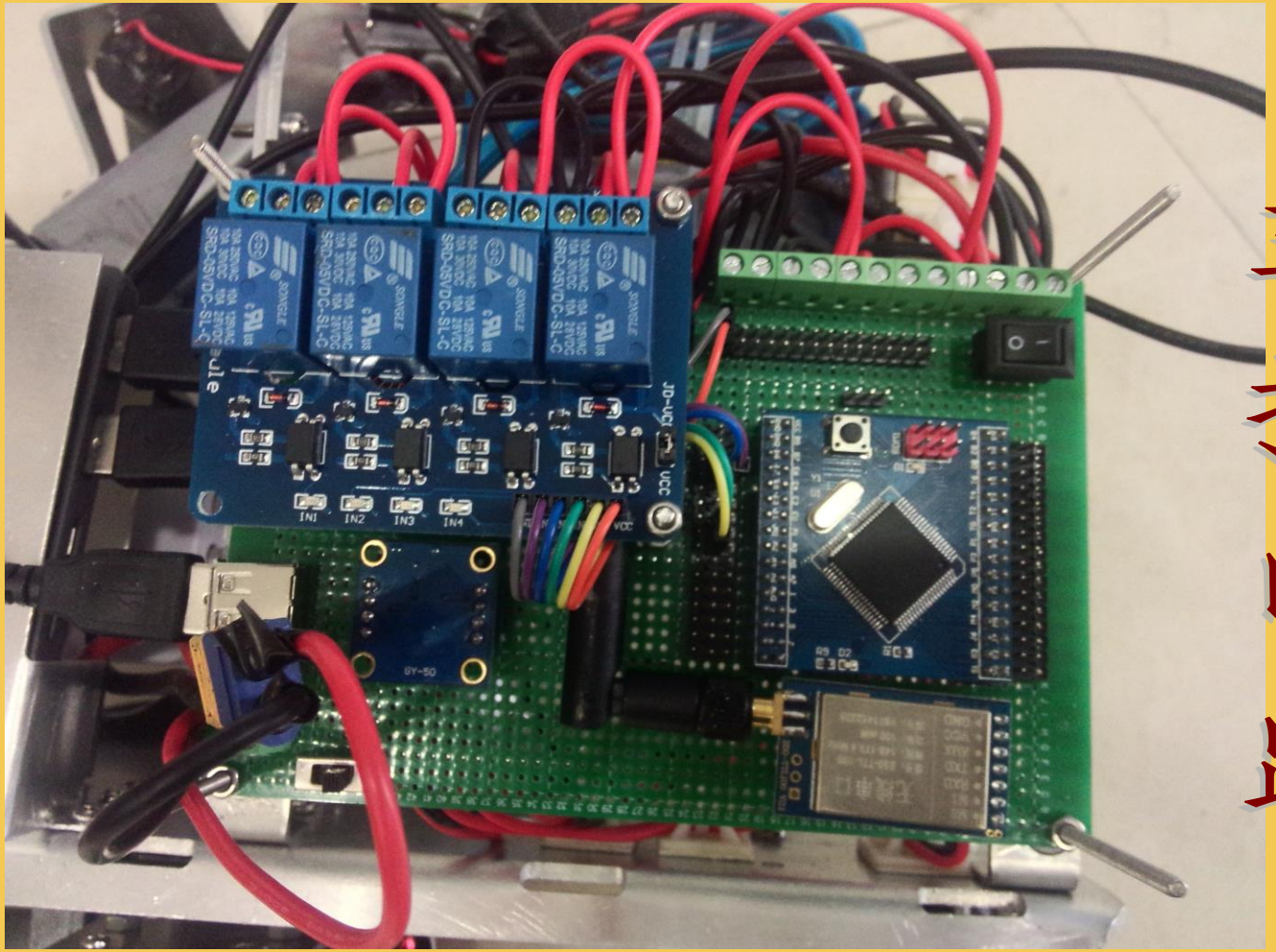


- **软件控制系统主控芯片采用的是飞思卡尔MC9S12XS128单片机，所以选择CodeWarrior IDE v5.9为开发环境**

3 硬件设计制作

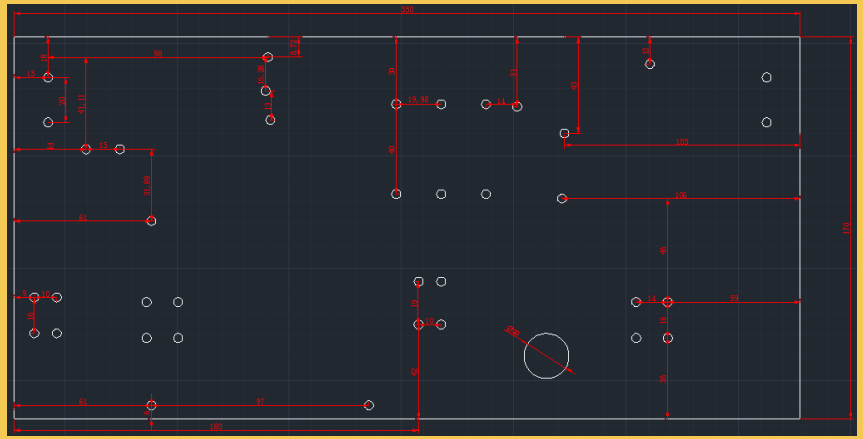
- 控制电路设计制作
- 机械机构设计制作

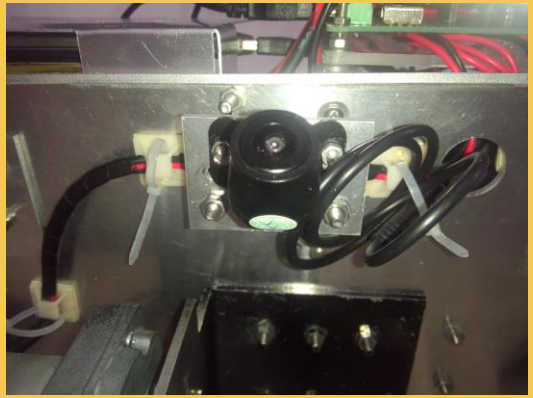
- 根据XS128最小系统原理图，以及各传感器，执行器占用资源情况，焊接电路。

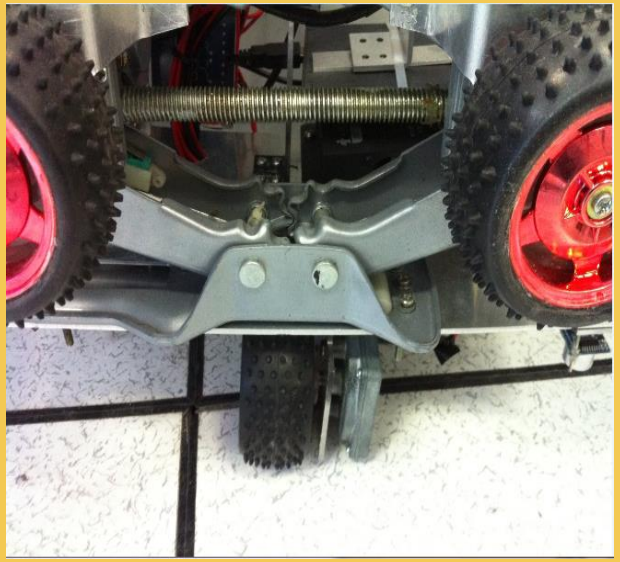


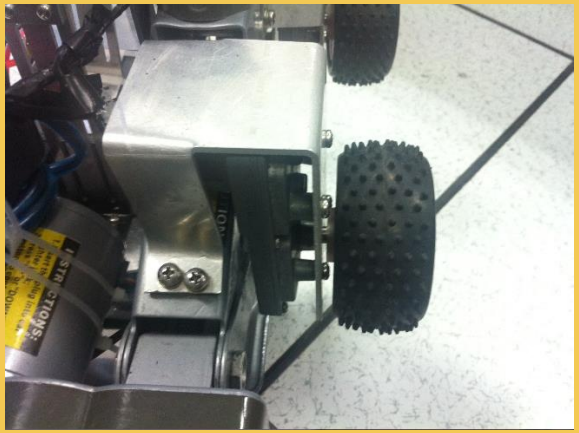
主控电路

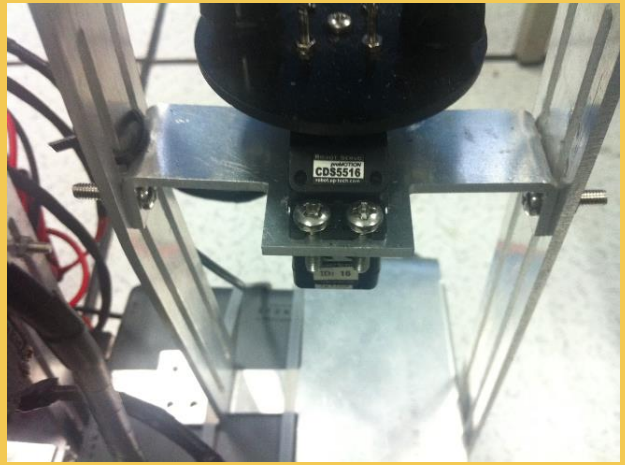
- 机械结构设计制作包括小车底盘设计，电机、舵机安装，千斤顶固定，限位开关安装，超声波安装固定，相机安装。











4 软件控制与设计

- 电机控制
- 限位开关状态读取
- 超声波定位高度
- 陀螺仪旋转角度测量
- 数字舵机控制
- Nokia 5110液晶显示

- 软件控制设计部分的主要内容是利用编写的程序对各部分硬件进行控制，需要熟练的运用掌握

```
//-----限位开关------//
#define advance_xianwei PORTB_PB3
#define tight_xianwei    PORTB_PB4
#define xianwei_action 0
if(step==0)
    {
        if(advance_xianwei==xianwei_action)   //贴紧杆体 进
        {
         robot_state=state_tight;
         duoji_weizhi=1850;
         step=1;
        }
    }
if(step==1)
```

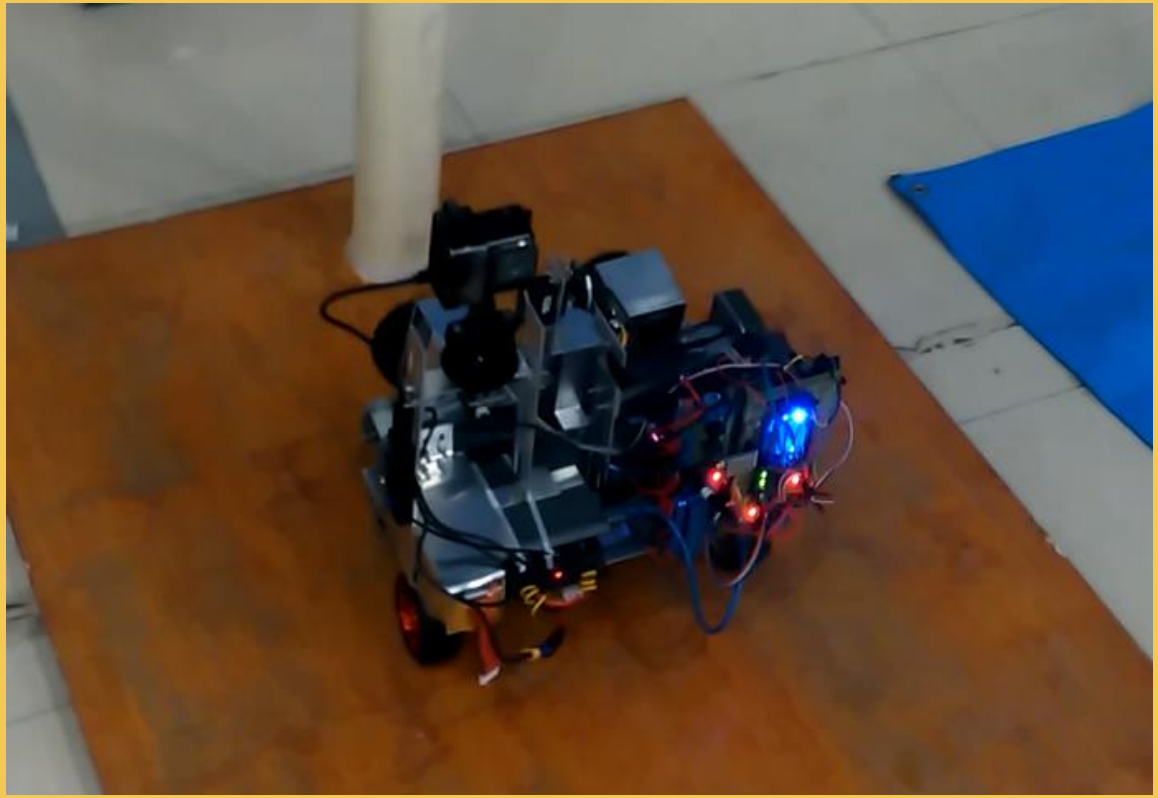
```
    unsigned char Receive_data_buf[30];//多字节读时数据缓存区
    #include"I2C_xs128.c"
    /*void    I2C_write_reg(unsigned    char    I2C_slave_address,unsigned    char REG_address,unsigned char REG_data)
    通过 I2C 总线向某一寄存器写入一个字节数据
    I2C_slave_address:器件地址
    REG_address:寄存器地址
    EG_data:写入寄存器数据
    unsigned  char  I2C_read_reg(unsigned  char  I2C_slave_address,  unsigned  char REG_address)
    通过 I2C 总线读出某一寄存器的------(单字节数据) I2C_slave_address:器件地址
    REG_address:寄存器地址
    返回:Receive_data
    示例:m= I2C_read_reg(0x38, 0x22)
    void  I2C_multiple_read_reg(unsigned  char  I2C_slave_address,unsigned  char REG_address,unsigned char MAX_byte)
    通过 I2C 总线读出某一寄存器的------(多字节数据)
    I2C_slave_address:器件地址
    REG_address:寄存器地址
    MAX_byte:连续读数据个数的最大值
    无返回:数据存于全局变量 Receive_data_buf[]中适用于器件寄存器地址自动加的场合
    示例:I2C_read_reg(0x38, 0x22);m=Receive_data_buf[0];*/
    通过 I2C 总线读取到陀螺仪角速度后，再对其进行软件积分，中断服务子程序如下：
    #pragma CODE_SEG __NEAR_SEG NON_BANKED
    void interrupt 66 timer0(void)      //10us 定时
    {
```

```
    #define    ECHO1    PORTA_PA4     //接收
    #define    TRIG2    PORTA_PA2     //触发
    #define    ECHO2    PORTA_PA0     //接收
    #define    ultrasonic_amount    2    //超声波模块的数量
    extern volatile unsigned int dis1;//0 号超声波测得的距离
    extern volatile unsigned int dis2;//1 号超声波测得的距离
    void ultrasonic_measure(void); //超声波测量函数
    #endif
    void ultrasonic_measure(void)
    {
      static unsigned int Time_50ms;
      static unsigned int time1;
      static unsigned int time2;
      static unsigned int ultrasonic_change=0;
      Time_50ms++;        //并不是标准的 50ms,但这样能保证定时触发,虽然对
号计数时,也会计数,但影响并不大
      if(Time_50ms==5000)   //    50ms 触发一次超声波 例如超声波数量为 2,
上为 100ms 触发一次
        {
          Time_50ms=0;
          ultrasonic_change++;
          if(ultrasonic_change>ultrasonic_amount)   ultrasonic_change=1;
          ultrasonic_choose(ultrasonic_change);
        }
        //-------第 0 组-------//
        if(Trig_flag1==1&&ECHO1==1)
```

```
        Weishi_L=(unsigned char)(Weishi_limit;
        can16_send(0xff);
        can16_send(0xff);
        can16_send(ID);
        can16_send(0x07);
        can16_send(0x03);
        can16_send(0x06);
        can16_send(0x00);
        can16_send(0x00);
        can16_send(Weishi_H);
        can16_send(Weishi_L);
        Jiaoyan_sum=~(ID+0x10+Weishi_H+Weishi_L);
        can16_send(Jiaoyan_sum);
    }
    void Yibu_write(unsigned char ID,unsigned int Weishi,int Sudu,unsigned char action)  //
    {
        unsigned char Weishi_L,Weishi_H,Sudu_L,Sudu_H,Jiaoyan_sum,flag=0;
        char i;
        Weishi=Weishi+150; //将初始 0 位置校为 150 度
        Weishi=Weishi*3.41;
        if(Weishi>1023)   Weishi=1023;//限幅限制
        if(Sudu>1023)     Sudu=1023;
        if(Sudu<-1023)    Sudu=-1023;
        if(Sudu<0) {Sudu=-Sudu;flag=1;}
        Weishi_H=(unsigned char)(Weishi>>8);   //高八位与低八位分离(char 型占 1 个字节)
        Weishi_L=(unsigned char)(Weishi;
        Sudu=(unsigned int)Sudu;
        Sudu_H=(unsigned char)(Sudu>>8);
        if(flag==1)   {Sudu_H=Sudu_H+0x04;flag=0;}   //改变速度方向
```

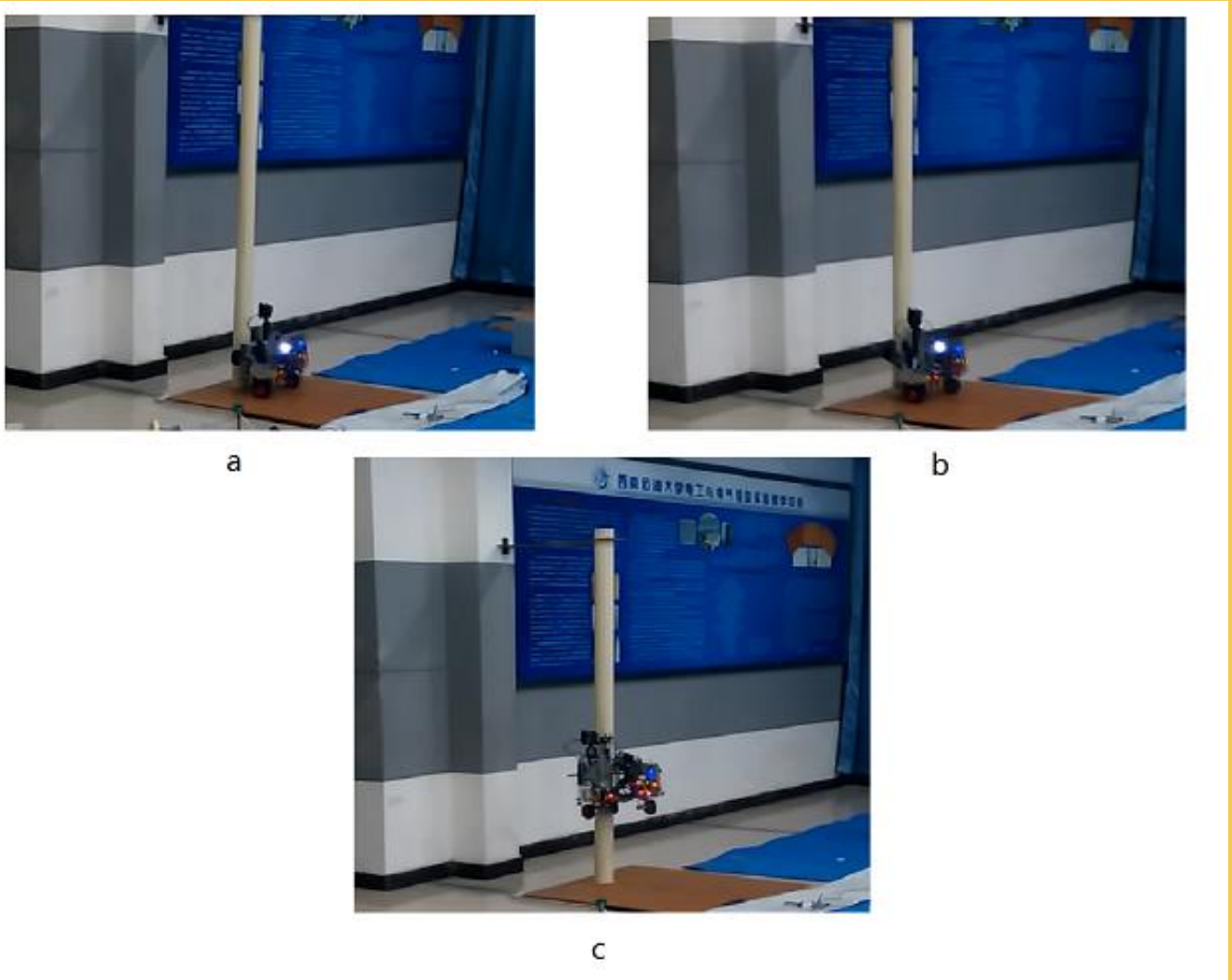
5 联机调试

开机等待一键启动



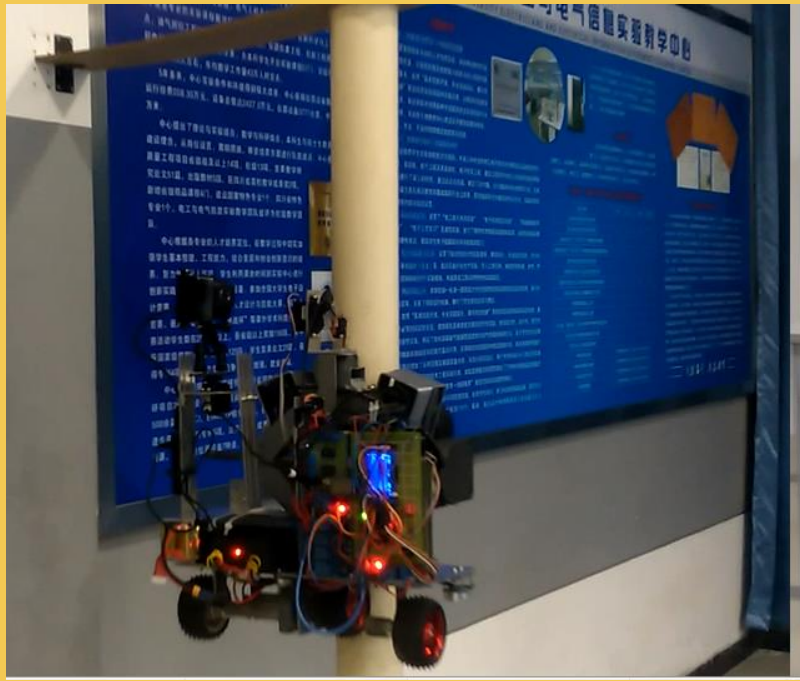
按下启动按钮

机器人运动状态

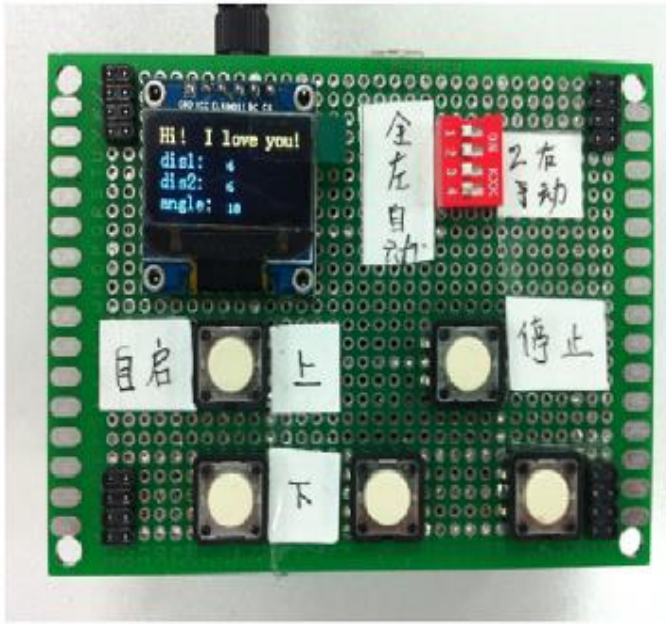


- a图为机器人水平向杆体靠近
- b图为机器人夹紧动作
- c图为机器人爬升动作

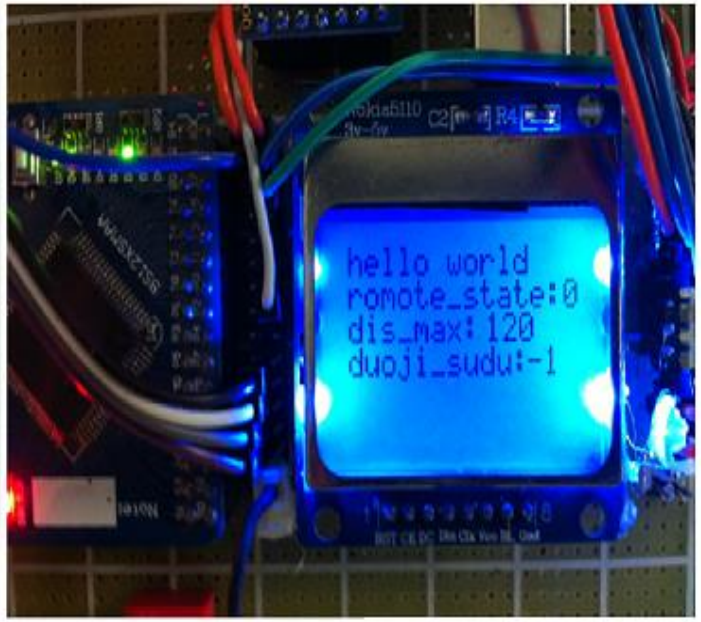
定向摄像



监测调试


Hi! I love you!
dis1: 4
dis2: 4
angle: 10
全左自动
2右手动
自启
上
停止
下

a


hello world
romote_state:0
dis_max:120
duoji_sudu:-1

b

6 总结

在基于现有丰硕的研究成果上，经历了很多次失败，尝试了很多种方案，经过优劣对比以及性能优化，最终选定目前所使用的方案，也实现了快速爬升、定位准确、及时传回信息等所需要实现的重要功能的哨兵机器人。

感谢结局，更感谢过程，一路走来，收获颇丰！


THANK YOU !

QUESTION

Q

&

AND

A

ANSWER