关于 RM 夏令营机器人的技术报告大纲(16 组)

组员及分工：机械：杨曦、俞沐开、罗芷晴

嵌入式：武馨怡、王昱潼

算法：唐朝龙、鲜昊甫

一、 需求分析（abstract）

1、根据比赛主题和规则的描述，我们通过讨论决定制作四面都可跨越一层方块的机器人，并且在机器内部进行方块的夹取和储存。

2、比赛以第一人称视角操作，顾我们需要图像传感器来进行传输图像，且视野要尽可能的广泛，所以需要舵机控制图像传感器，控制其旋转角度。

3、实现机器基本功能：底盘系统，抬升系统，夹子系统。

4、抬升装置需要限位开关 防止下降到底后皮带打滑

5、由于麦轮本身的原因 在自动时期轮子打滑可能会导致底盘走的不直 需要用陀螺仪进行纠正 同样旋转也可利用陀螺仪来控制底盘旋转

6、第一阶段算法运用插头动态规划在联通 16 个颜色格子的情况下尽可能让路线占满整张地图，并且每个格子到与之颜色对应的资源区的距离和最小。

7、第二阶段算法使用贪心策略求出局部最优解，在每次铺设堡垒时找出地图上能够提高分数最多的位置，每当地图发生变化时，都更新一次最佳的放置堡垒的位置。

8、UI 要能够明显的显示出每个格子要铺设什么颜色的方块，使操作手能够快速的看懂。

二 所需技术点，关键词

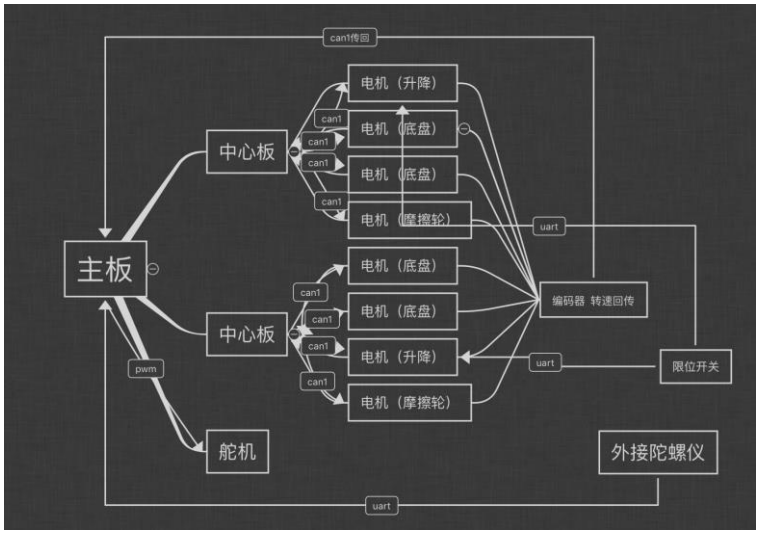
机械：气动装置、传动装置

嵌入式：舵机控制、PID 闭环控制、陀螺仪矫正、限位开关的合理使用

算法：插头状态压缩动态规划、贪心算法、局部最优解

三 总体方案

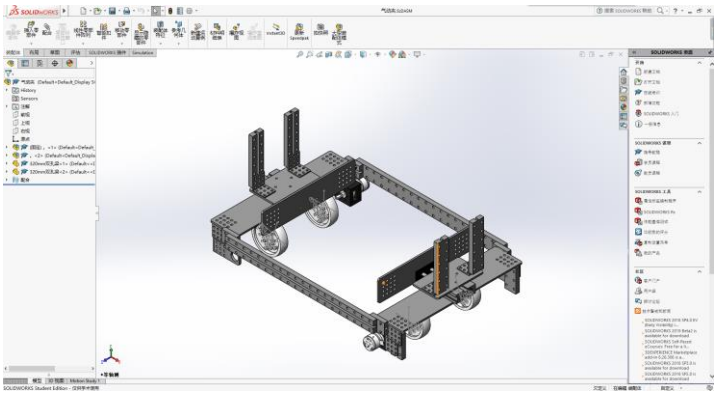
给出各系统之间的相互联系关系，比如，电池驱动主控板，电机，电调，舵机，主控板，编码器，传感器和电机等的信息传递。



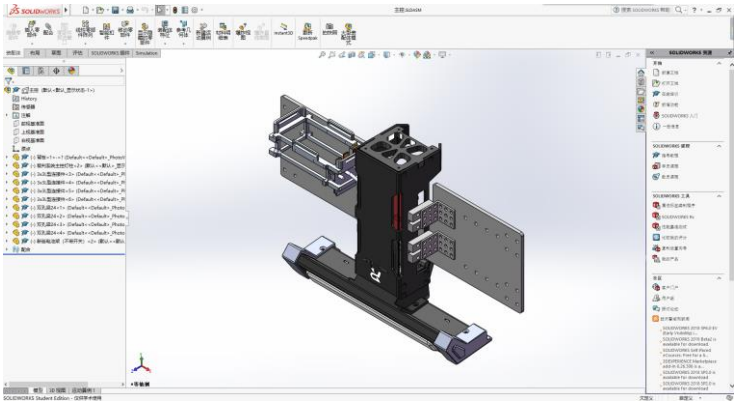
四 各模块方案

1、 机械部分：

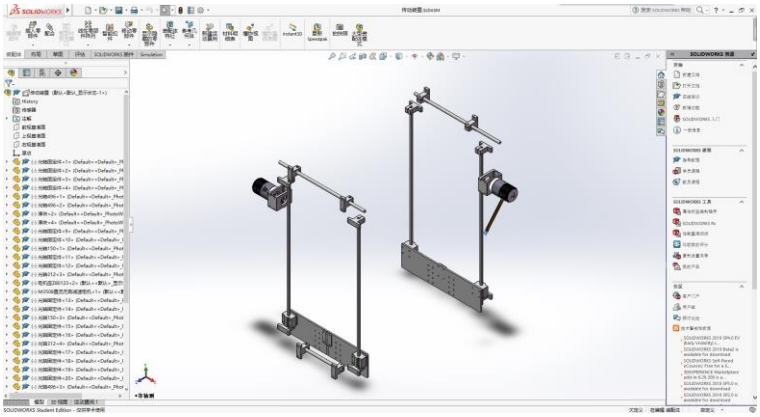
（1） 气动夹



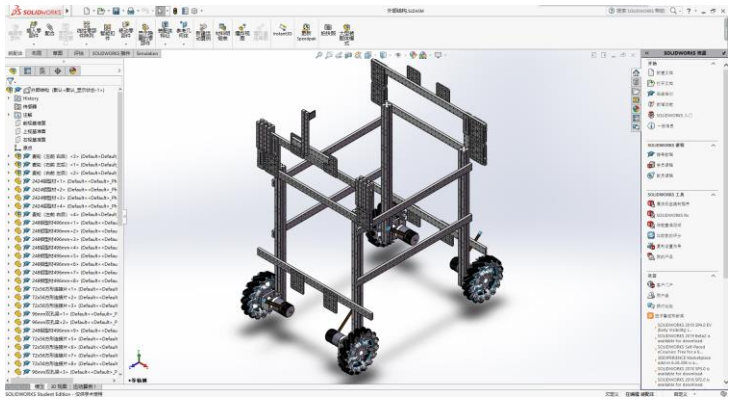
（2）.主控



（3）.传动装置



（4）.外部结构



嵌入式部分：

（一）通讯模块

1、can 通讯----驱动电机----上机测试----调整 pid 与速度 功率（俞沐开、武馨怡）

2、板间通讯（uart 通讯）：把外接主控板的的数据传回 stm32（王昱潼、唐朝龙）

发送数据：

```
#pragma config(UART_Usage, UART1, uartUserControl, baudRate4800, IOPins, None, None)
#pragma config(Sensor, in1,    gyro_vex,       sensorGyro)
//*!!Code automatically generated by 'ROBOTC' configuration wizard               !!*//
void sendgyrodata(int a)
{
  sendChar(UART1, a);
  sendChar(UART1, a >> 8);
  sendChar(UART1, a >> 16);
  sendChar(UART1, a >> 24);
  wait1Msec(50);
}
task sendgyrodata_task()
{
  int gyro_vexda,bbb=0;
  setBaudRate(UART1, baudRate4800);
  configureSerialPort(UART1, uartUserControl);
  while(1)
  {
    gyro_vexda = SensorValue[gyro_vex];
    sendgyrodata(gyro_vexda);
  }
}
task main()
{
  startTask(sendgyrodata_task);
}
```

接收数据：

```
#include "execute_task.h"
#include "can_device.h"
#include "uart_device.h"
#include "cmsis_os.h"
#include "calibrate.h"
#include "pid.h"
#include "sys.h"
#include "string.h"

int32_t gyro_vex, enc_L_vex, enc_R_vex,judge;
uint8_t temp[4];

void getdataback(void)
{
  judge=1;//To determine whether or not to enter the callback function
  memcpy(&gyro_vex, temp, 4);
}

void execute_task(const void* argu)
{
  judge=0;
  temp[0]=0;
  temp[1]=0;
  temp[2]=0;
  temp[3]=0;
  uart_init(USER_UART3,4800,WORD_LEN_8B,STOP_BITS_1,PARITY_NONE);
  uart_recv_callback_register(USER_UART3,getdataback);
  uart_receive_start(USER_UART3, temp, 4);

  while(1)
    osDelay(5);
}
```

（二）执行器模块（武馨怡、俞沐开）

1、pwm 控制----调整舵机角度

```
uint16_t position = 1500, speed = 3;

void camera_task(const void* argu)
{
  set_pwm_group_param(PWM_GROUP1,20000);
  set_pwm_param(PWM_IO8,20000);
  start_pwm_output(PWM_IO8);

  while(1)
    {
        position += rc.ch4/110*speed;
        VAL_LIMIT(position, 1200, 1750);
        set_pwm_param(PWM_IO8,position);

        osDelay(20);

    }
}
```

2、继电器控制电磁阀

3、利用中心板分线

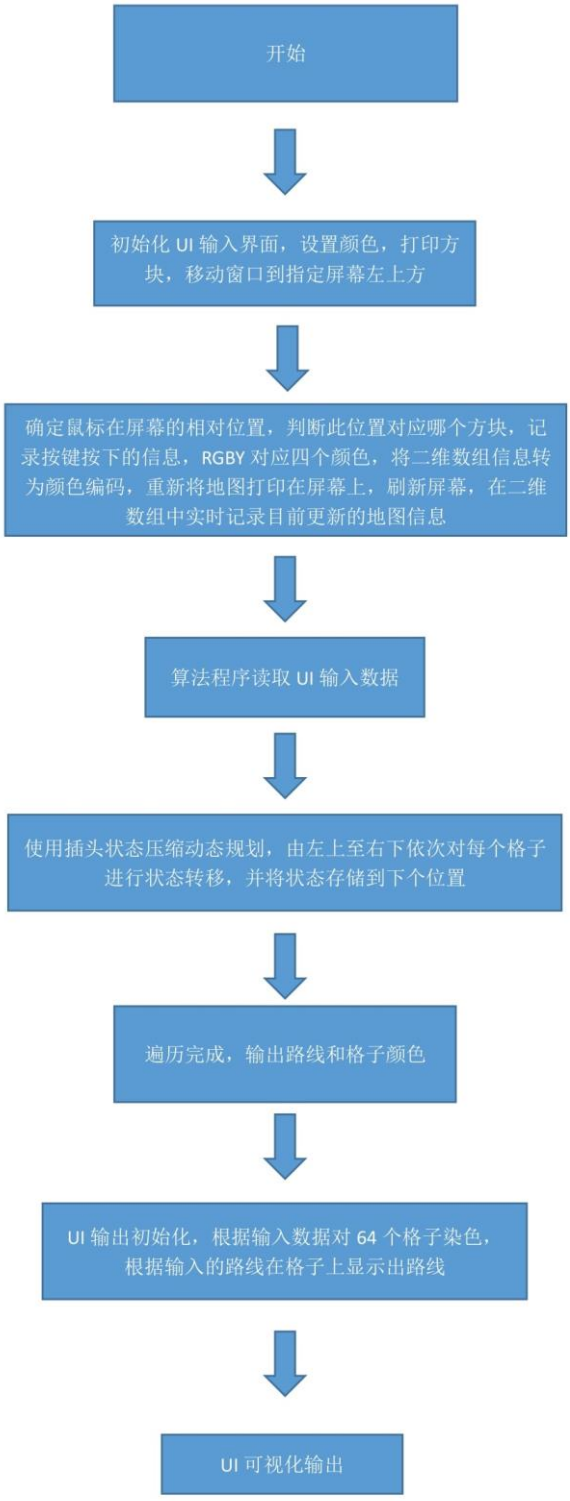
（三）传感器模块（王昱潼）

1、限位开关的使用（抬升限位）

2、陀螺仪对底盘矫正（第二阶段 陀螺仪接在外接的板子上）

算法部分：

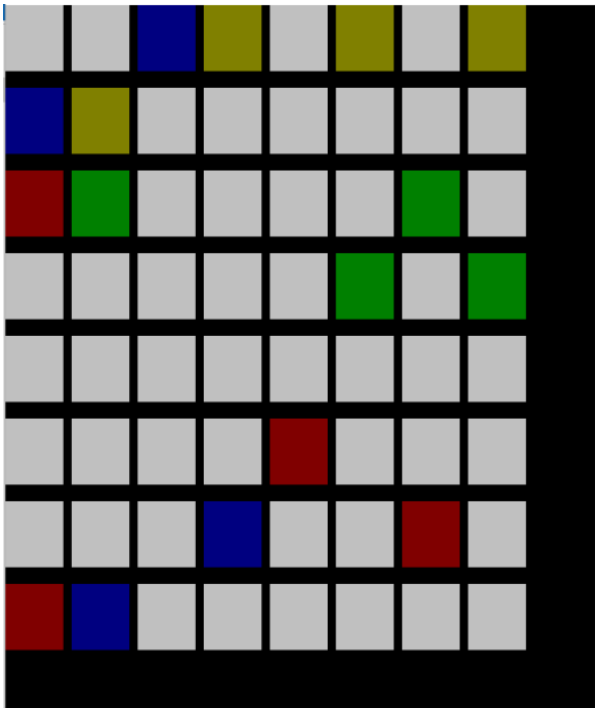
1. 流程图：

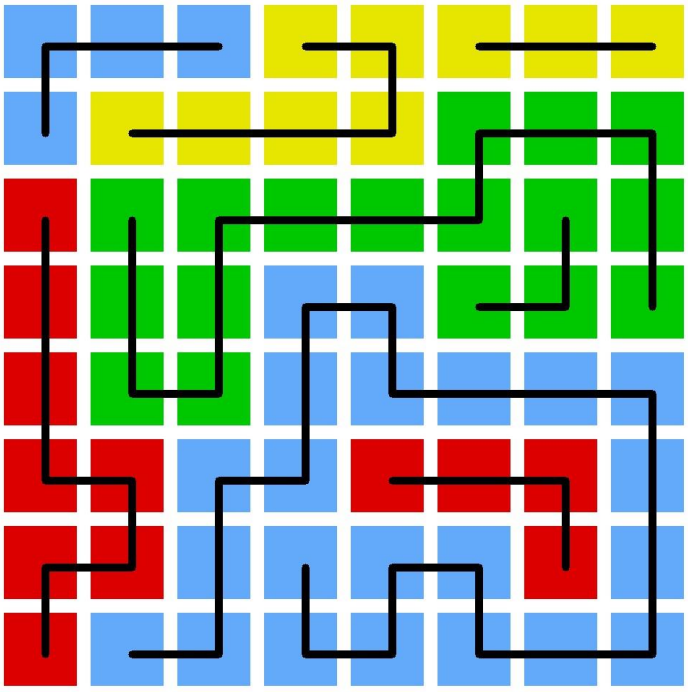


2. 思路：

算法部分用的是插头状态压缩动态规划的一个方法，由左上至右下依次对每个格子进行状态转移，并将状态存储到下个位置，轮廓线长度为 9，轮廓线上方的格子即为确定的路线，在路线穿过轮廓线的地方，为一个插头。当轮廓线上方路线的两端都穿过轮廓线时，左边的插头称为左插头，右边的插头称为右插头。当完成对地图的遍历后，即可求出路线。

3. UI 设计：





五 理论分析（anaylsis）

机械：3508 功率（加电调）：240W

2006 功率（加电调）：72W

气缸：TN10*15

轮距：393.10mm

轴距：543.40mm

高度：855.06mm

长度：685.78mm

宽度：542.20mm

资源块间距（资源区内）（方块中心距）：248.00mm

算法：

因为使用了插头动态规划，算法能够完美的求出解，即在联通 16 个颜色格子的情况下路线能够完全填满整张地图。插头动态规划的求解速度经测试大约为 0.05 秒，整个程序包括 UI 输入，求解路线和 UI 输出大约为 0.25 秒。

嵌入式：

1、板间通讯中，通讯中断时间为 50 毫秒。

2、

给出控制参数。

写出遇到的困难和问题，重点写出走过的弯路，例如没有进行负载的估计，导致动力选型不合理等（介绍各部分任务的完成者）

五 制作与测试流程

这一部分主要记录制作机器人与进入场地测试的经历，讲讲初始规划中与实际偏差较大的部分，分析问题原因，以及解决方案（讲讲是谁提出了解决问题的方案）

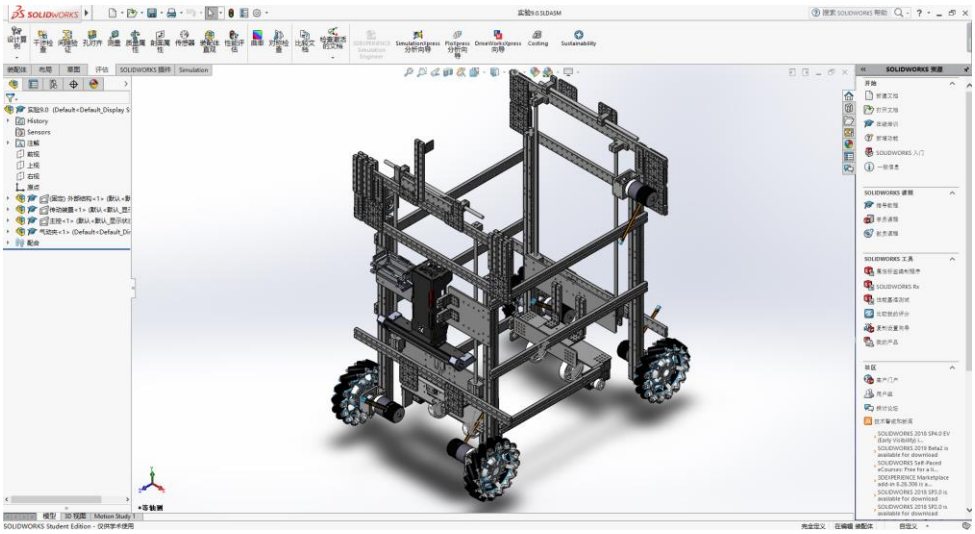
六 结果与评价

优势：可以拆卸城堡

劣势：储存搭建城堡的速度较慢

创新点：拆卸装置和夹取装置合在一起

七 附录（Appendix）请将代码及机械制图等粘贴在附录中



```
#pragma config(UART_Usage, UART1, uartUserControl, baudRate4800, IOPins, None, None)
#pragma config(Sensor, in1,    gyro_vex,       sensorGyro)
//*!!Code automatically generated by 'ROBOTC' configuration wizard               !!*//
void sendgyrodata(int a)
{
  sendChar(UART1, a);
  sendChar(UART1, a >> 8);
  sendChar(UART1, a >> 16);
  sendChar(UART1, a >> 24);
  wait1Msec(50);
}
task sendgyrodata_task()
{
  int gyro_vexda,bbb=0;
  setBaudRate(UART1, baudRate4800);
  configureSerialPort(UART1, uartUserControl);
  while(1)
  {
    gyro_vexda = SensorValue[gyro_vex];
    sendgyrodata(gyro_vexda);
  }
}
task main()
{
  startTask(sendgyrodata_task);
}
```

```
#include "execute_task.h"
#include "can_device.h"
#include "uart_device.h"
#include "cmsis_os.h"
#include "calibrate.h"
#include "pid.h"
#include "sys.h"
#include "string.h"

int32_t gyro_vex, enc_L_vex, enc_R_vex,judge;
uint8_t temp[4];

void getdataback(void)
{
  judge=1;//To determine whether or not to enter the callback function
  memcpy(&gyro_vex, temp, 4);
}

void execute_task(const void* argu)
{
  judge=0;
  temp[0]=0;
  temp[1]=0;
  temp[2]=0;
  temp[3]=0;
  uart_init(USER_UART3,4800,WORD_LEN_8B,STOP_BITS_1,PARITY_NONE);
  uart_recv_callback_register(USER_UART3,getdataback);
  uart_receive_start(USER_UART3, temp, 4);

  while(1)
    osDelay(5);
}
```

```
uint16_t position = 1500, speed = 3;

void camera_task(const void* argu)
{
  set_pwm_group_param(PWM_GROUP1,20000);
  set_pwm_param(PWM_IO8,20000);
  start_pwm_output(PWM_IO8);

  while(1)
    {
      position += rc.ch4/110*speed;
      VAL_LIMIT(position, 1200, 1750);
      set_pwm_param(PWM_IO8,position);

      osDelay(20);

    }
}
```

这个大纲从理论上来说是一个小项目初级的研发流程，按照这个流程你们会省去很多不必要的麻烦，规避很多不必要的风险。你们通过这一段时间的经历应该对这个感触很深（如果你们真正投入了这个夏令营的话）当然大家也可以自由发挥，想什么写什么。把自己的看法都加进去，这个会是你们之后拿出去给别人展示的一份技术报告。