ROBOMASTER WINTER CAMP 2019 答辩PPT

Group 7

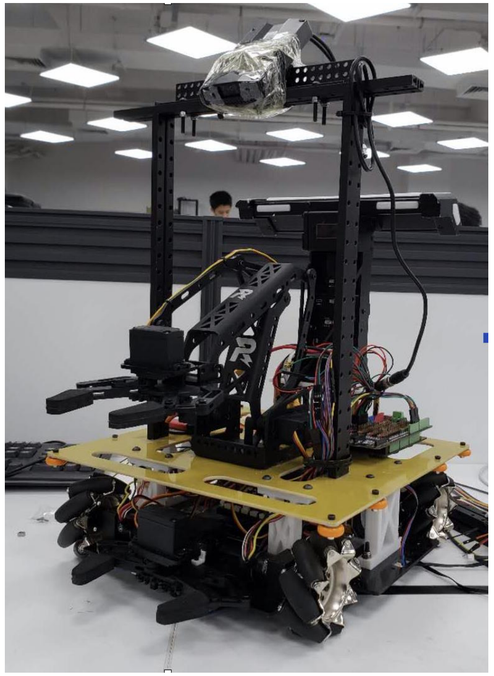
队长：林文韬
队员：李杰相、钟思哲、许臻言、单昱、冯思衡、宋沛昱、姜华祖

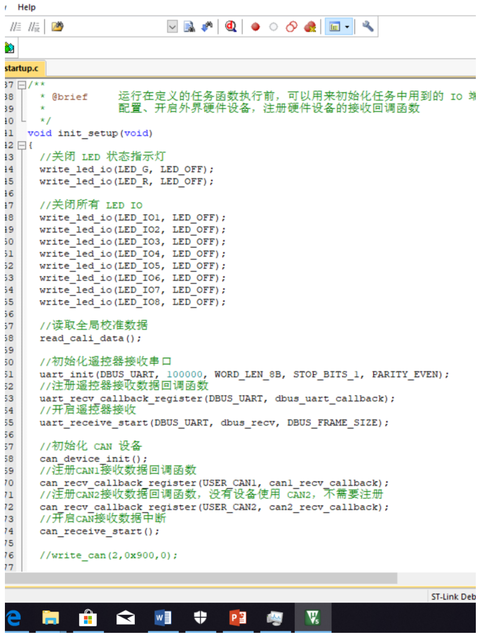
整体分工

2

机械：整体结构设计

目标：让小车能完成指定任务并同时保持结构稳定性。





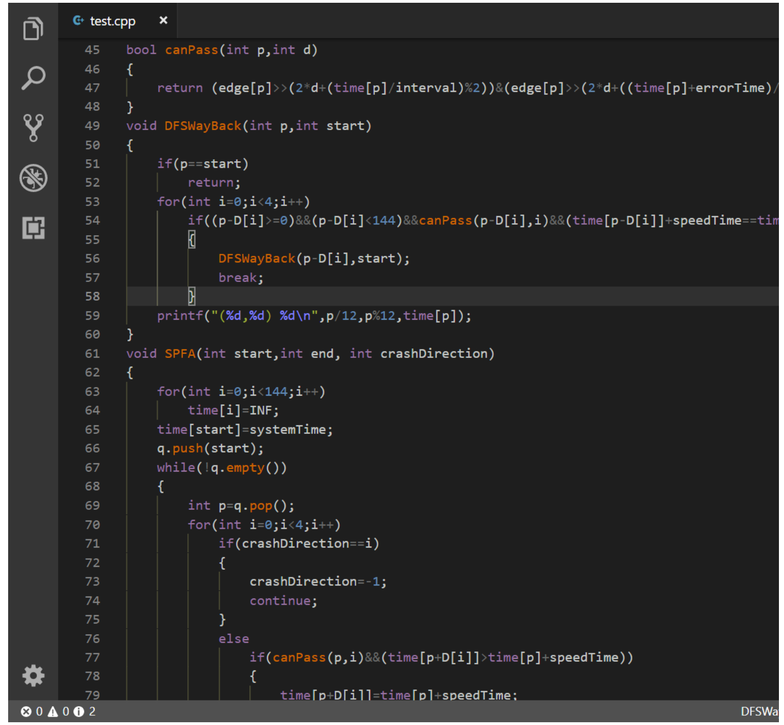
```
Help
startup.c
/**
  * @brief      运行在定义的任务函数执行前，可以用来初始化任务中用到的 IO 等
  *             配置、开启外界硬件设备，注册硬件设备的接收回调函数
  */
void init_setup(void)
{
  //关闭 LED 状态指示灯
  write_led_io(LED_G, LED_OFF);
  write_led_io(LED_R, LED_OFF);

  //关闭所有 LED IO
  write_led_io(LED_IO1, LED_OFF);
  write_led_io(LED_IO2, LED_OFF);
  write_led_io(LED_IO3, LED_OFF);
  write_led_io(LED_IO4, LED_OFF);
  write_led_io(LED_IO5, LED_OFF);
  write_led_io(LED_IO6, LED_OFF);
  write_led_io(LED_IO7, LED_OFF);
  write_led_io(LED_IO8, LED_OFF);

  //读取全局校准数据
  read_cali_data();

  //初始化遥控器接收串口
  uart_init(DBUS_UART, 100000, WORD_LEN_8B, STOP_BITS_1, PARITY_EVEN);
  //注册遥控器接收数据回调函数
  uart_recv_callback_register(DBUS_UART, dbus_uart_callback);
  //开启遥控器接收
  uart_receive_start(DBUS_UART, dbus_recv, DBUS_FRAME_SIZE);

  //初始化 CAN 设备
  can_device_init();
  //注册CAN1接收数据回调函数
  can_recv_callback_register(USER_CAN1, can1_recv_callback);
  //注册CAN2接收数据回调函数，没有设备使用 CAN2，不需要注册
  can_recv_callback_register(USER_CAN2, can2_recv_callback);
  //开启CAN接收数据中断
  can_receive_start();

  //write_can(2,0x900,0);
ST-Link Deb
```

嵌入式：机械结构控制

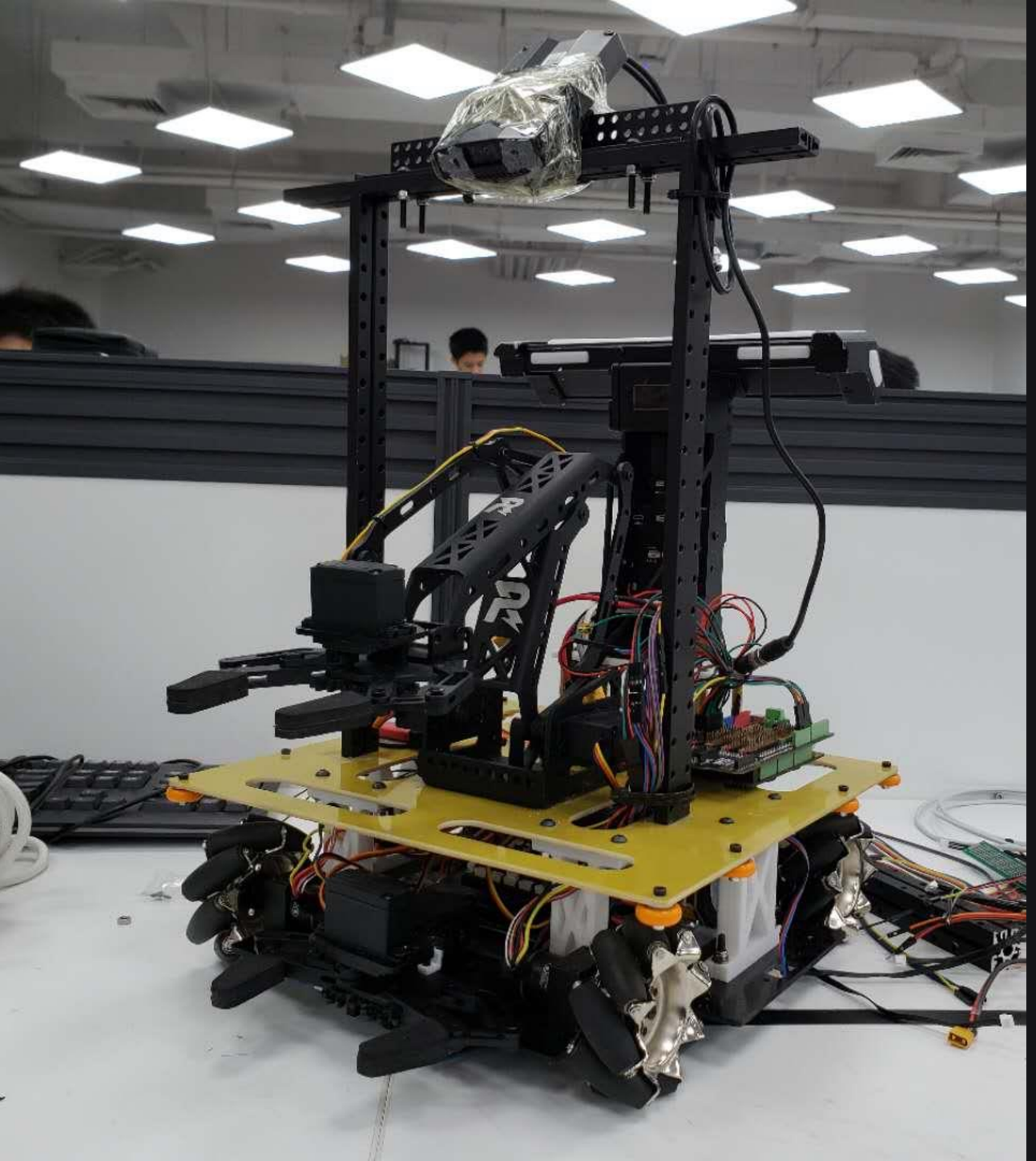
目标：保证机器运行稳定性

算法：生成路径并进行导航

目标：指导工兵自动移动并以最高效率运行

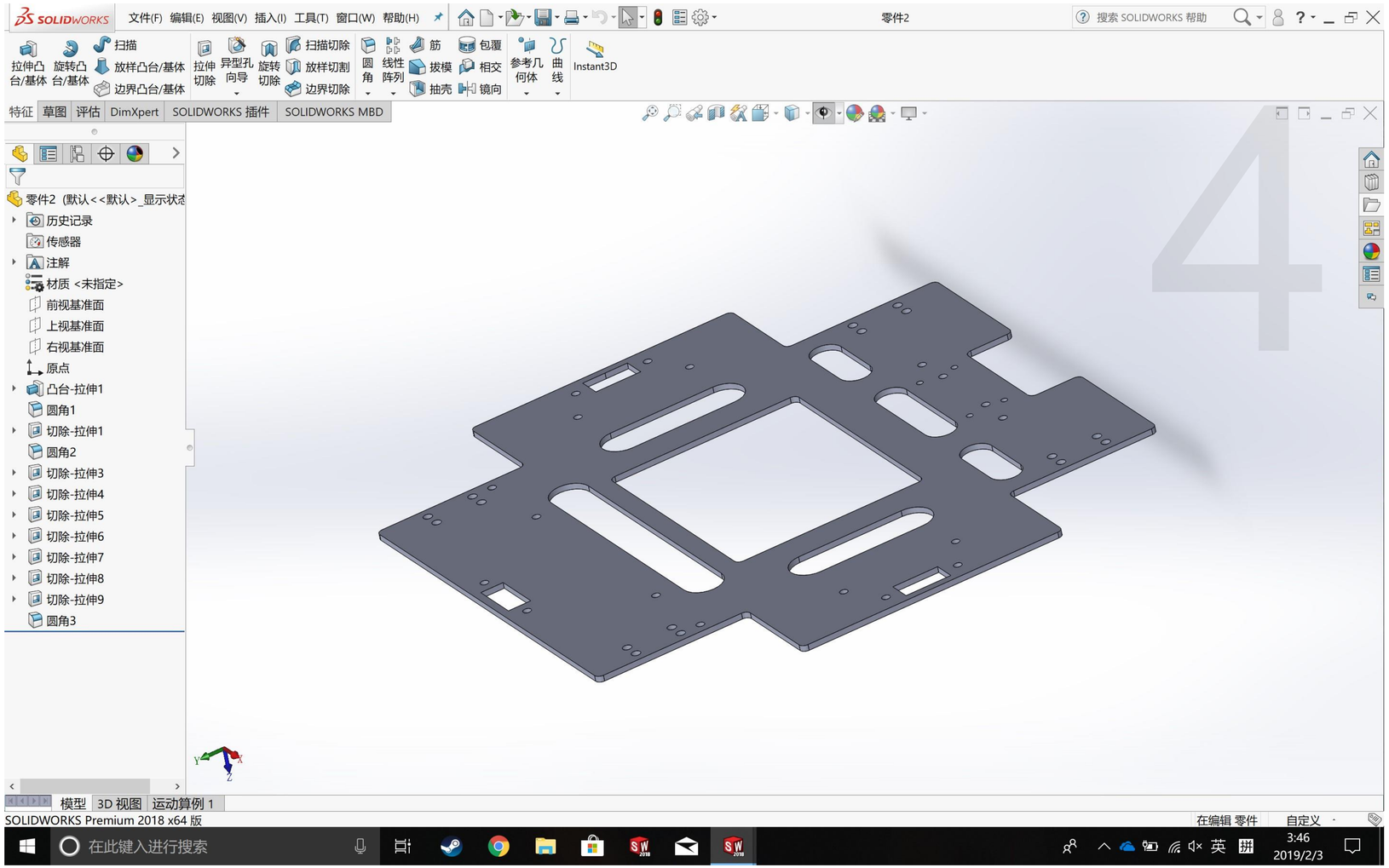


```
test.cpp
45  bool canPass(int p,int d)
46  {
47      return (edge[p]>>(2*d+(time[p]/interval)%2))&(edge[p]>>(2*d+((time[p]+errorTime)/
48  }
49  void DFSWayBack(int p,int start)
50  {
51      if(p==start)
52          return;
53      for(int i=0;i<4;i++)
54          if((p-D[i]>=0)&&(p-D[i]<144)&&canPass(p-D[i],i)&&(time[p-D[i]]+speedTime==tim
55          {
56              DFSWayBack(p-D[i],start);
57              break;
58          }
59      printf("(%d,%d) %d\n",p/12,p%12,time[p]);
60  }
61  void SPFA(int start,int end, int crashDirection)
62  {
63      for(int i=0;i<144;i++)
64          time[i]=INF;
65      time[start]=systemTime;
66      q.push(start);
67      while(!q.empty())
68      {
69          int p=q.pop();
70          for(int i=0;i<4;i++)
71              if(crashDirection==i)
72              {
73                  crashDirection=-1;
74                  continue;
75              }
76              else
77                  if(canPass(p,i)&&(time[p+D[i]]>time[p]+speedTime))
78                  {
79                      time[p+D[i]]=time[p]+speedTime;
0 0 2
DFSWa
```



工兵机器人的设计与考量

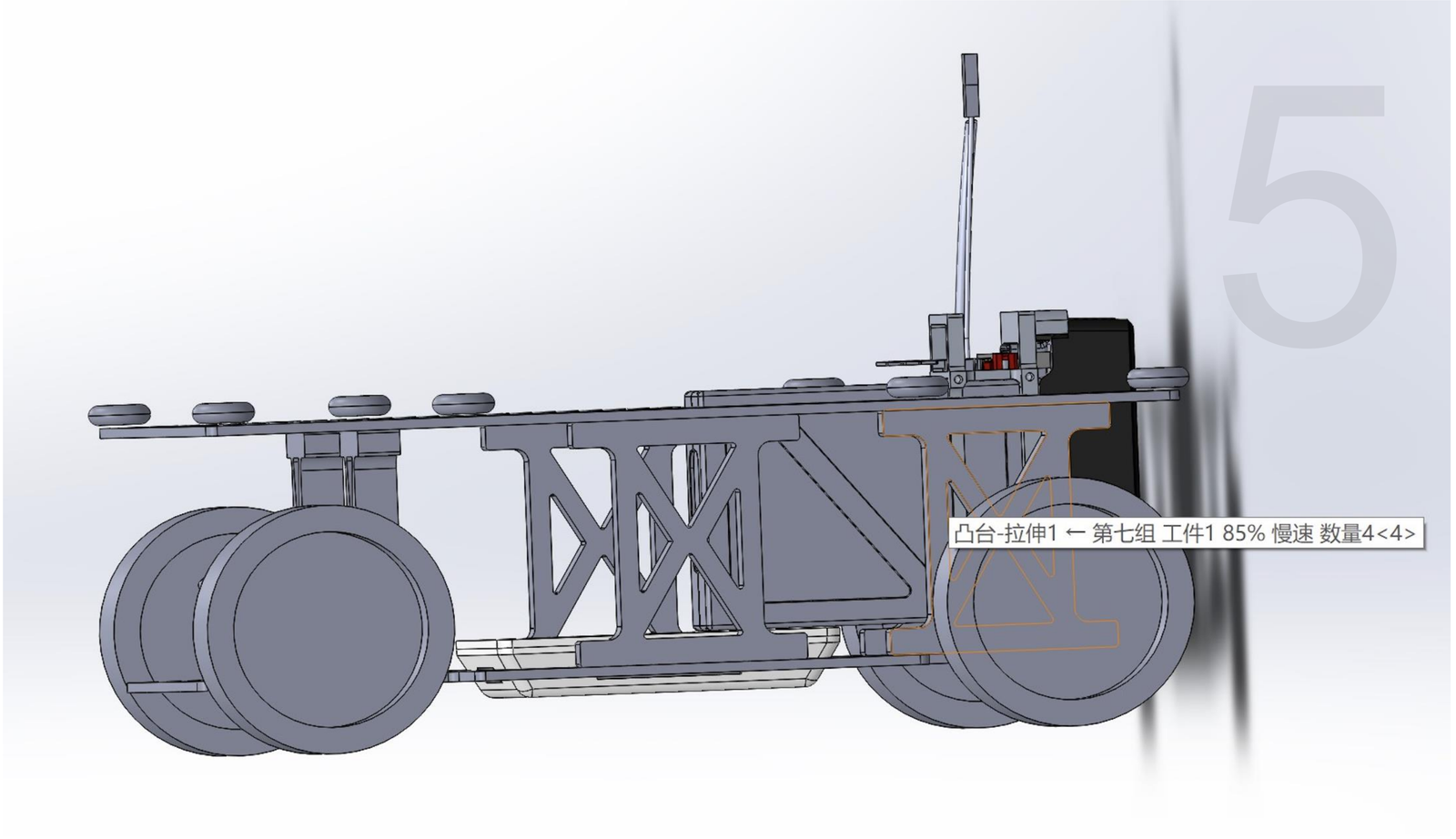
- RFID的安装
- 自定义底盘
- 双重夹子设计
- 正方形底盘



自定义底盘

意义：

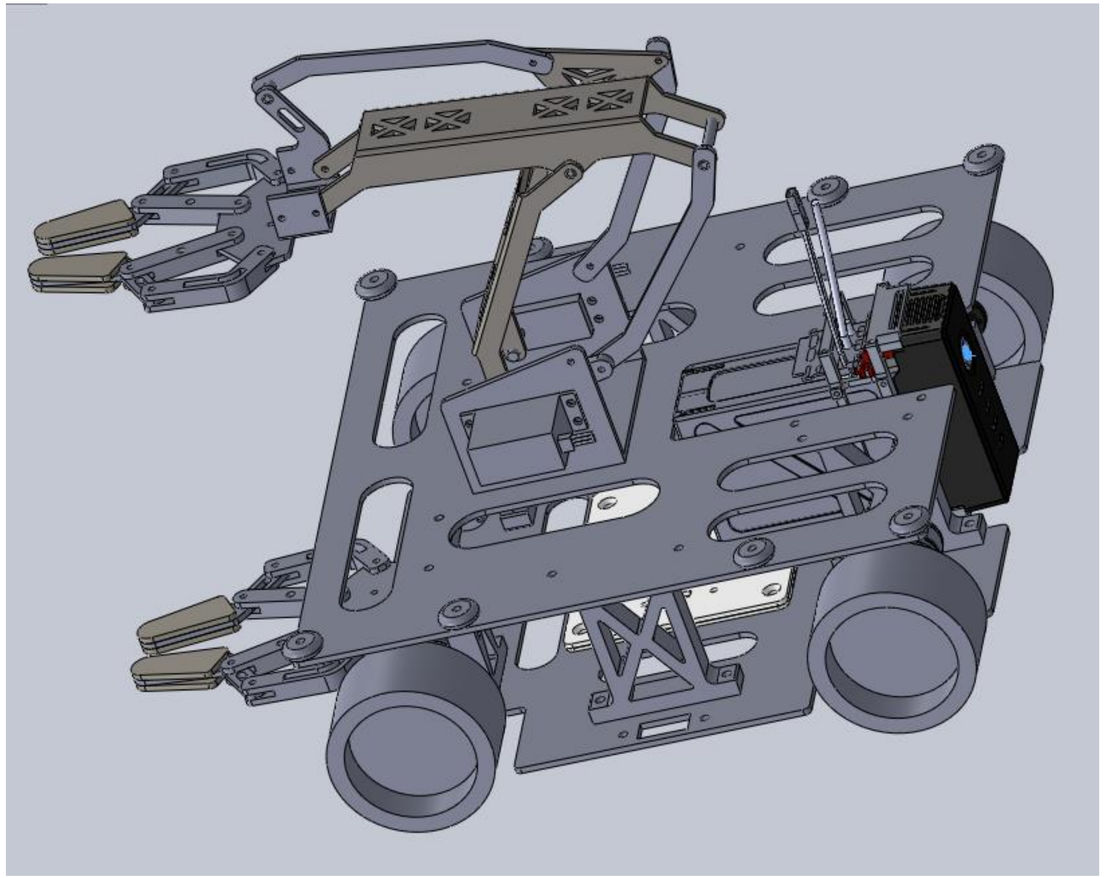
- 提高拓展能力
  - RFID 的安装
  - 裁判系统的安装
  - 双夹的安装
- 提高容错率
  - 减少重新加工板子的时间
  - 便于更改现有零件位置与添加新的传感器
  - 材料强度的控制



RFID

意义：

- 增加底盘与地面的空间以给予小车一定的通过能力
- 保证RFID能被准确识别以支持算法运行
- 走线方便

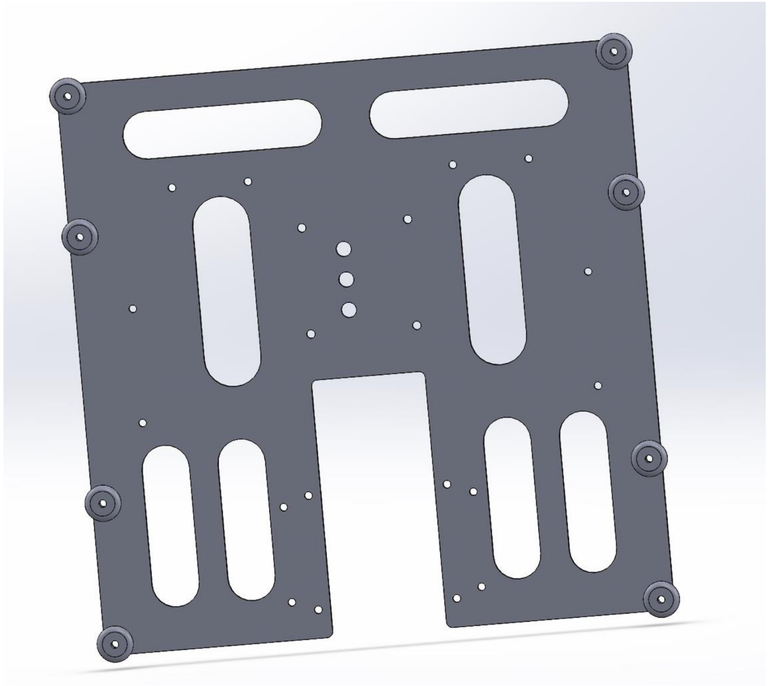


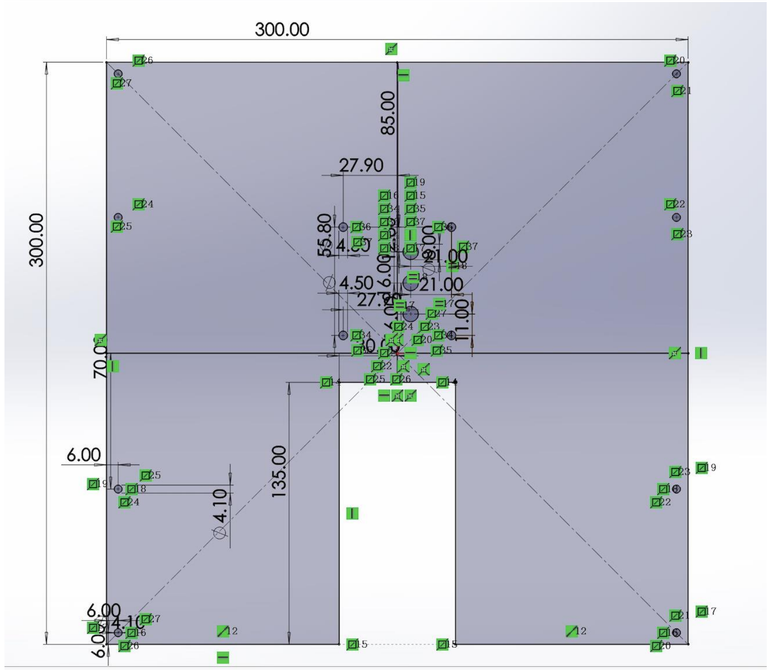
6

双夹设计

意义：

- 提高拓展能力
  - RFID 的安装
  - 裁判系统的安装
  - 双夹的安装
- 提高容错率
  - 减少重新加工板子的时间
  - 便于更改现有零件位置与添加新的传感器



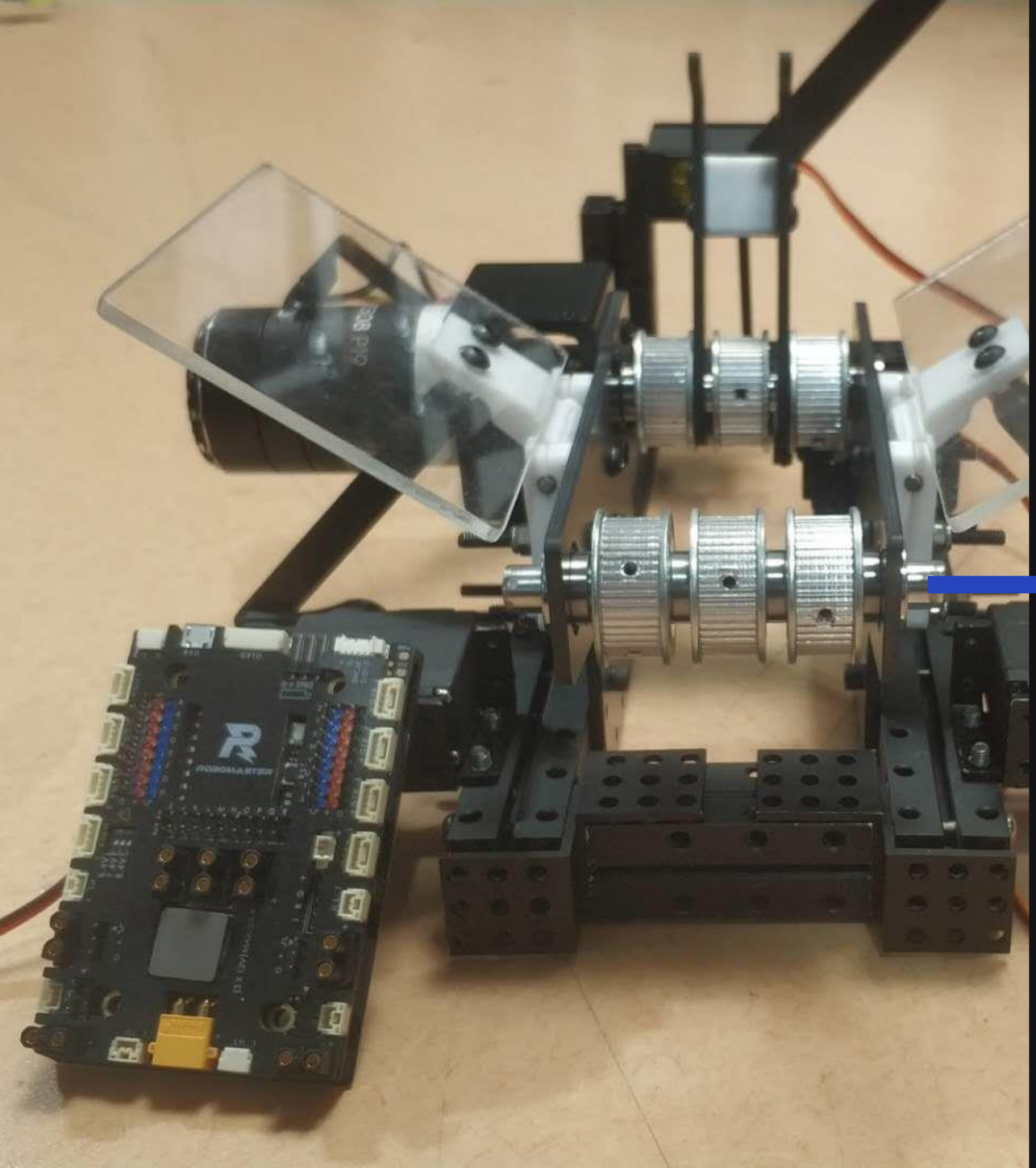


7

正方形底盘与导轮

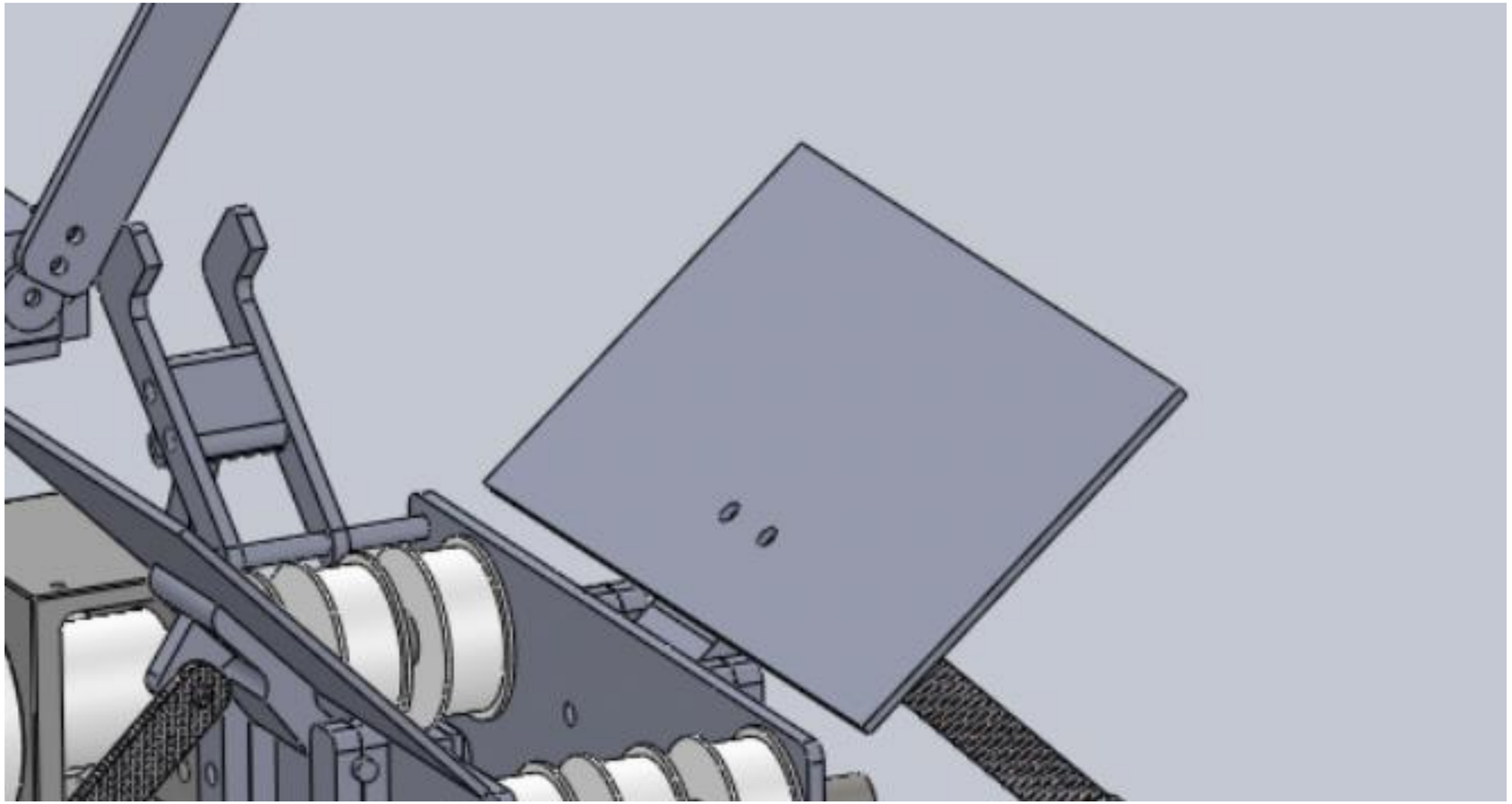
意义：

- 提高通过能力（长宽一致便于操作手操作）
- 减小体积更加简洁紧凑
- 假设意外撞墙导轮可以避免其卡死

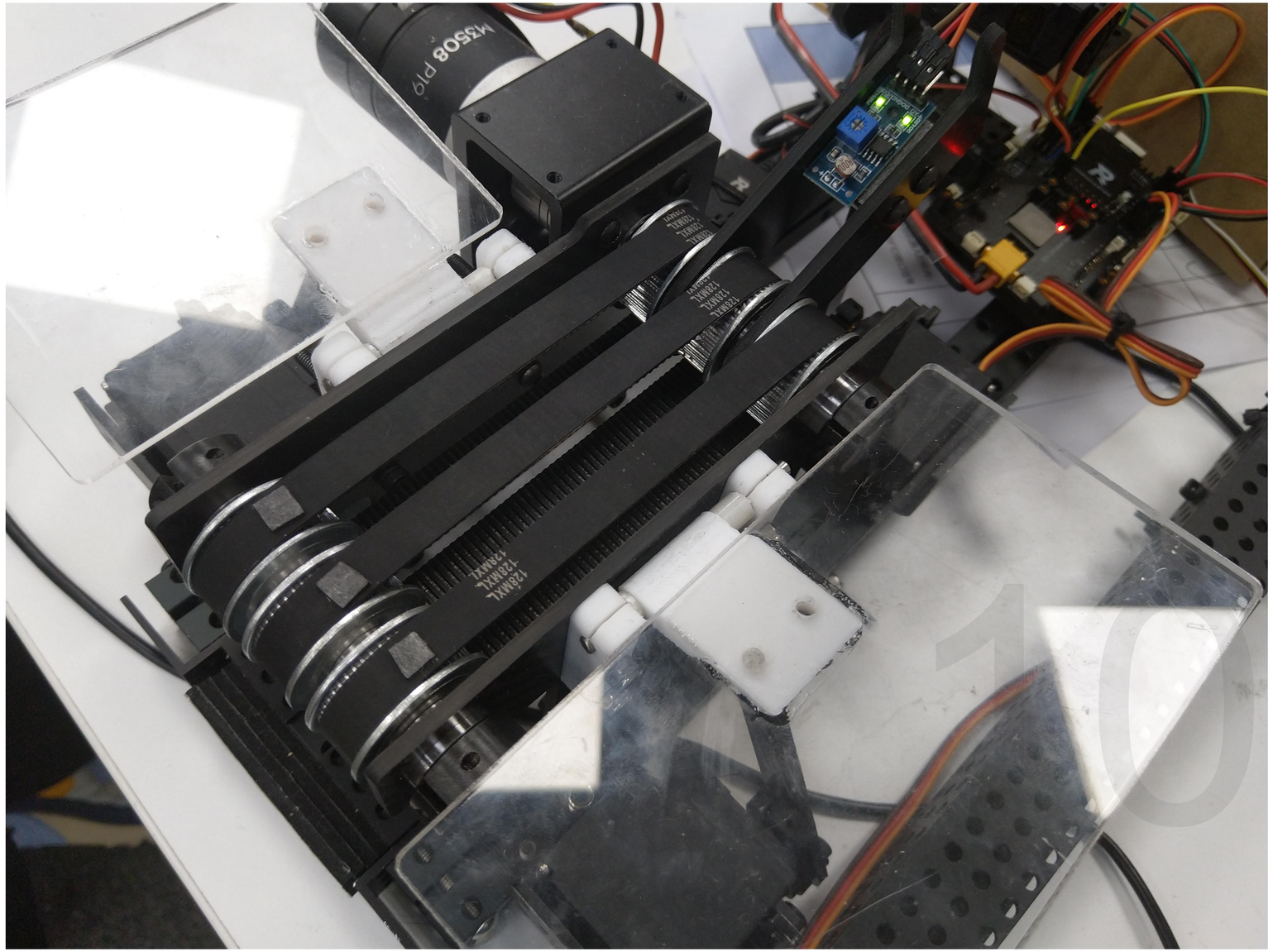


解密机器人

- 体积小
- 同步带设计
- 自动翻转结构
- 透光亚克力



解密结构使用了2块较大的亚克力板作为侧板，方块可以随意的丢在板上，之后通过舵机控制其合起，可将方块稳定的置于传送带上。这个人性化的设计，提高了操作手的容错率，也省去了对不齐造成的的烦恼。同时，亚克力侧板不会遮挡环境光，解密摄像头无需再增加补光灯



2：使用了3条同步带作为方块的传送带，充分利用了官方剩余的同步带，光轴等物资。一方面，三条同步带中间留有足够空隙供之后的方块翻转机构使用；另一方面，三条同步带加上足够间隙后总宽在60MM以内，而方块边长50MM，刚好顺着传送到移动不会被两边的亚克力板所挤压。同时，方块放在三条同步带上，同步带几乎没有下陷。

3：使用了传送带这种结构，不仅是因为他可以调整方块位置便于识别，还因为他搭配了3508电机后，可以较高速的反转将已经解密完成的方块推出，清空解密区使操作手下次携带方块来时，可以直接放入解密区。

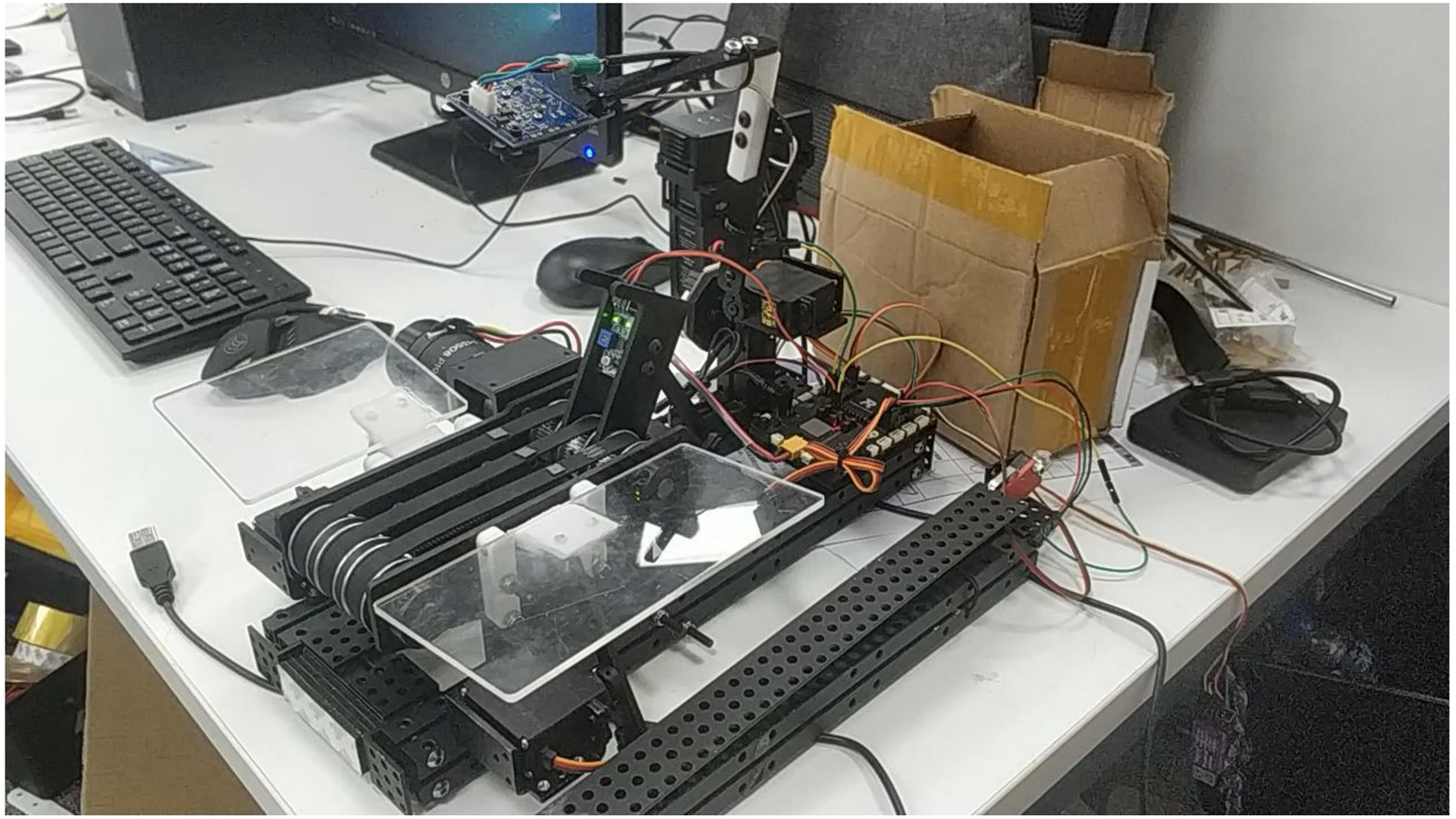
11

4：翻转机构的翻转臂直接以光轴作为其旋转轴，保证可以达到与轨道相切的位置，方块可以比较平滑的滑上翻转臂进行翻面的操作。为了保证每一次做出的翻转动作均为有效动作，我们在翻转臂上添加了一个光电传感器，当方块滑上旋转臂达一定高度后，再进行翻转，即可恰好翻转一个面。

同步带表面较为光滑，提供的摩擦力较小，导致方块有时不易滑上旋转臂。改进在同步带上粘贴了小块的3M胶贴，确保一定可以成功滑上旋转臂进行翻转。

5：使用了舵机带动曲臂转动摄像头的结构，可以使摄像头看到下方，看到左方和右方。看下方时配合翻转机构可以识别到方块的4个面，之后转动到左侧和右侧即可将整个方块全部扫描。

6：改进使用了3M强力胶水连接亚克力板与3D打印件,因为螺丝的突起会影响亚克力板在夹持过程中，造成方块滑出或是加持后阻力过大。改进后，亚克力板面光滑平整，方块滑动及传送不受阻。



7：使用微动开关及杠杆设计了一个奇妙的触发机关，操作手在放置方块后，只需用机械臂轻轻一拍，即可启动整套解密机构。

8：整个解密机构设计结构紧凑小巧，可以完成要求的功能，并为操作手提供较高容错率。

C:\Users\DJI-7\Desktop\666\Project\rm_infantry.uvprojx - µVision

File Edit View Project Flash Debug Peripherals Tools SVCS Window Help

rm_infantry

Project

- Project: rm_infantry
  - rm_infantry
    - user_driver
      - can_device.c
      - uart_device.c
      - calibrate.c
      - rm_hal_lib.h
      - keyboard.c
    - user_algorithm
      - pid.c
      - ramp.c
    - user_app
      - startup.c
      - gimbal_task.c
      - chassis_task.c
      - detect_task.c
      - execute_task.c
      - algorithm_task.c
      - xuanxian_task.c
    - user_custom
      - chassis_custom.c
      - gimbal_custom.c
      - shoot_custom.c
      - sys.h
      - servo.c
      - UART.c
      - GPIO.c
      - CAN2_ctrl.c
      - fifo.c
      - fifo.h
    - lib_rm
      - stm32f4xx_it.c
      - rm_infantry1_3.lib

Project | Books | Functions | Templates

startup.c

```
/**
  * @brief      运行在定义的任务函数执行前，可以用来初始化任务中用到的 IO 端口，
  *             配置、开启外界硬件设备，注册硬件设备的接收回调函数
  */
void init_setup(void)
{
  //关闭 LED 状态指示灯
  write_led_io(LED_G, LED_OFF);
  write_led_io(LED_R, LED_OFF);

  //关闭所有 LED IO
  write_led_io(LED_IO1, LED_OFF);
  write_led_io(LED_IO2, LED_OFF);
  write_led_io(LED_IO3, LED_OFF);
  write_led_io(LED_IO4, LED_OFF);
  write_led_io(LED_IO5, LED_OFF);
  write_led_io(LED_IO6, LED_OFF);
  write_led_io(LED_IO7, LED_OFF);
  write_led_io(LED_IO8, LED_OFF);

  //读取全局校准数据
  read_cali_data();

  //初始化遥控器接收串口
  uart_init(DBUS_UART, 100000, WORD_LEN_8B, STOP_BITS_1, PARITY_EVEN);
  //注册遥控器接收数据回调函数
  uart_recv_callback_register(DBUS_UART, dbus_uart_callback);
  //开启遥控器接收
  uart_receive_start(DBUS_UART, dbus_recv, DBUS_FRAME_SIZE);

  //初始化 CAN 设备
  can_device_init();
  //注册CAN1接收数据回调函数
  can_recv_callback_register(USER_CAN1, can1_recv_callback);
  //注册CAN2接收数据回调函数，没有设备使用 CAN2，不需要注册
  can_recv_callback_register(USER_CAN2, can2_recv_callback);
  //开启CAN接收数据中断
  can_receive_start();

  //write_can(2,0x900,0);
```

13

嵌入式

- 舵机调试
- 循线模块
- PID
- 串口通讯

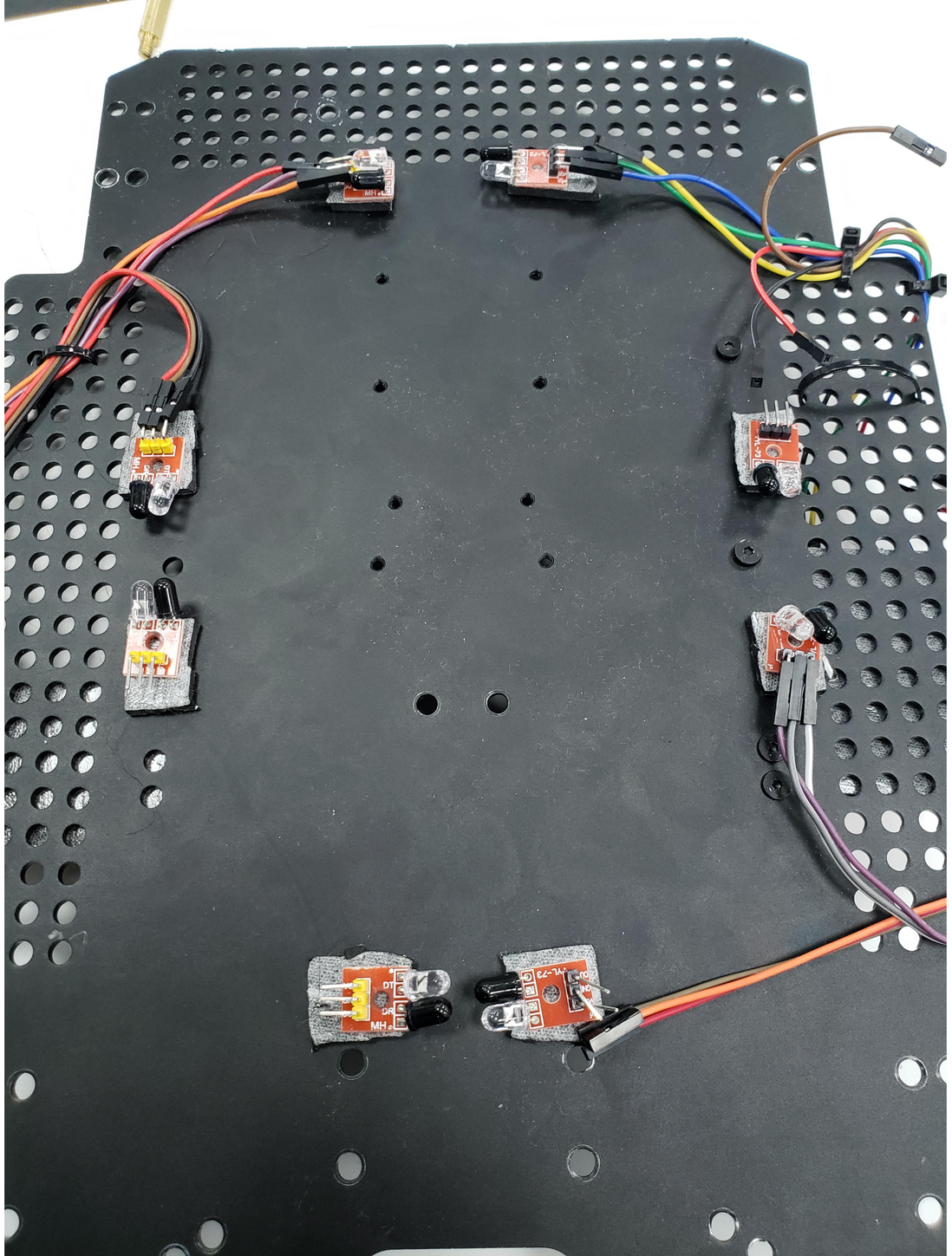
ST-Link Debugger | L:1 C:1 | CAP NUM SCRL OVR R/W

在此键入进行搜索

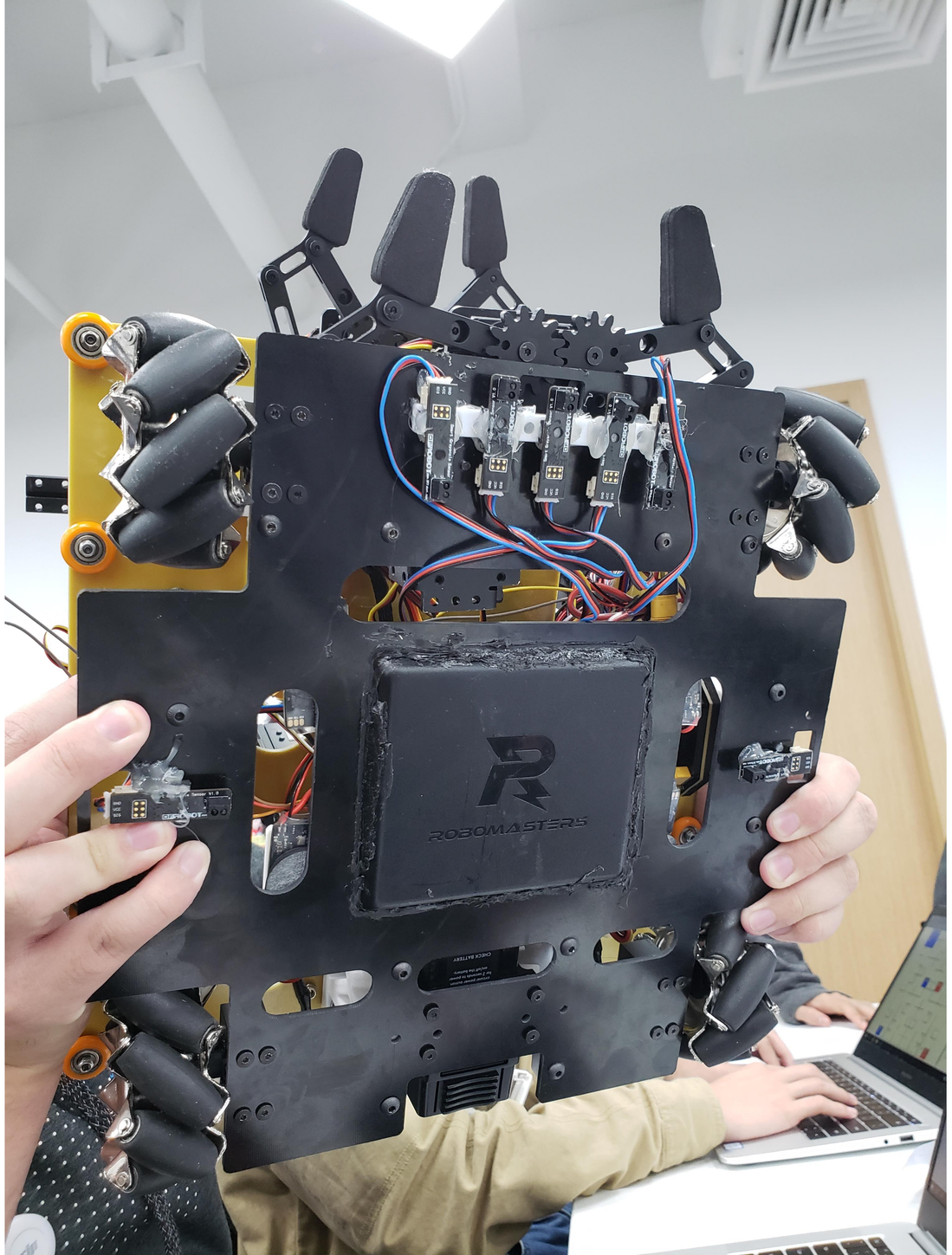
9:35
2019/2/3

我们之前没有接触过360°舵机，对于这种舵机非常不熟悉。并且教育套件的说明书对于这种舵机的使用说明也语焉不详。因此我们在开始调试时也碰到了很多BUG和坑。

在舵机的测试过程中，我们花费了大量的时间在测试舵机的实际角度与程序中的参数的对应关系上，直到最后调出大概的对应关系后才能顺畅的使用舵机



一开始，我们使用的是四路双红外传感巡线。这种思路的好处是可以完全不管地上的ID卡的颜色，只需要在经过ID卡时将运行方向后侧的双传感器暂时停止功能，就可以保持巡线的稳定性。唯一的问题是当机器偏移过大（比如被撞飞了），光靠底盘现有传感器难以矫正回来，因此我还引入了陀螺仪进行辅助矫正。但因为我们在更换底盘时的操作失误，造成了传感器损坏的情况。由于找不到足够的替换用的传感器，我们只得更换了巡线的设计思路。



巡线:

任务:

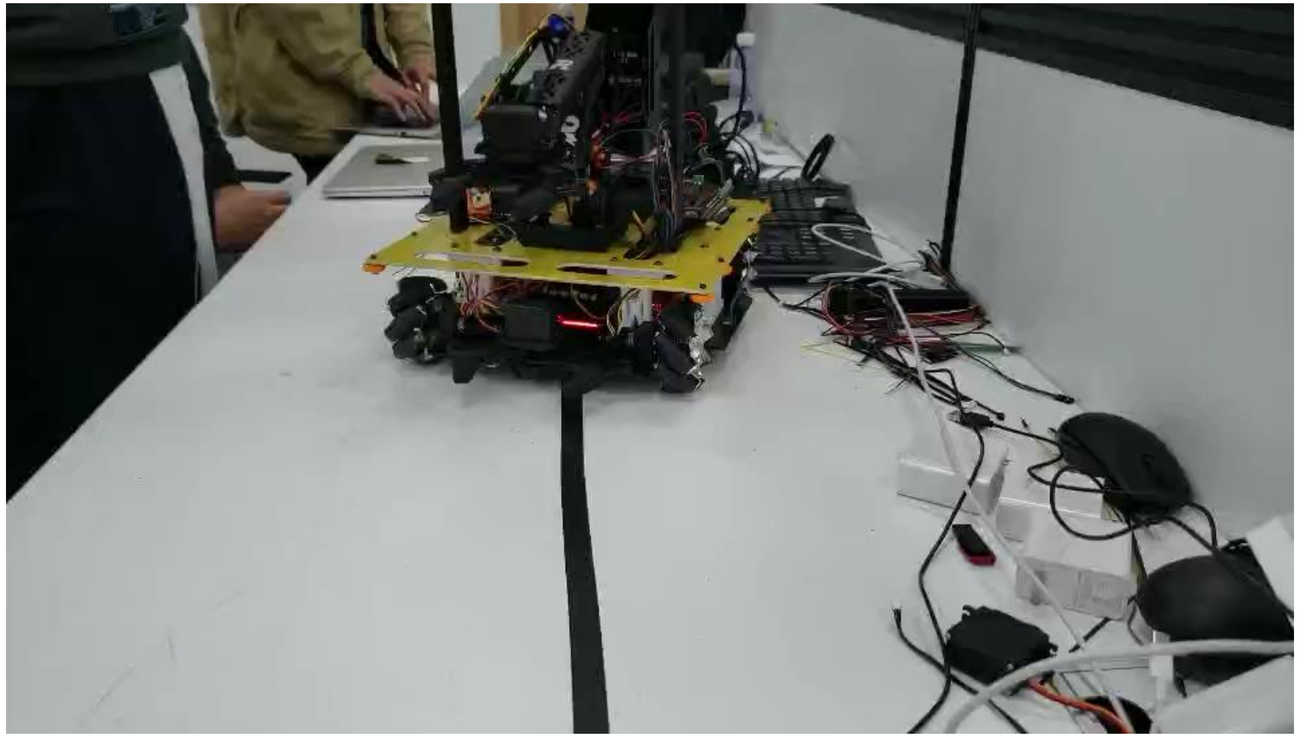
沿着黑线沿着某一条固定的路线行走
并保持图传指向正前方

方案:

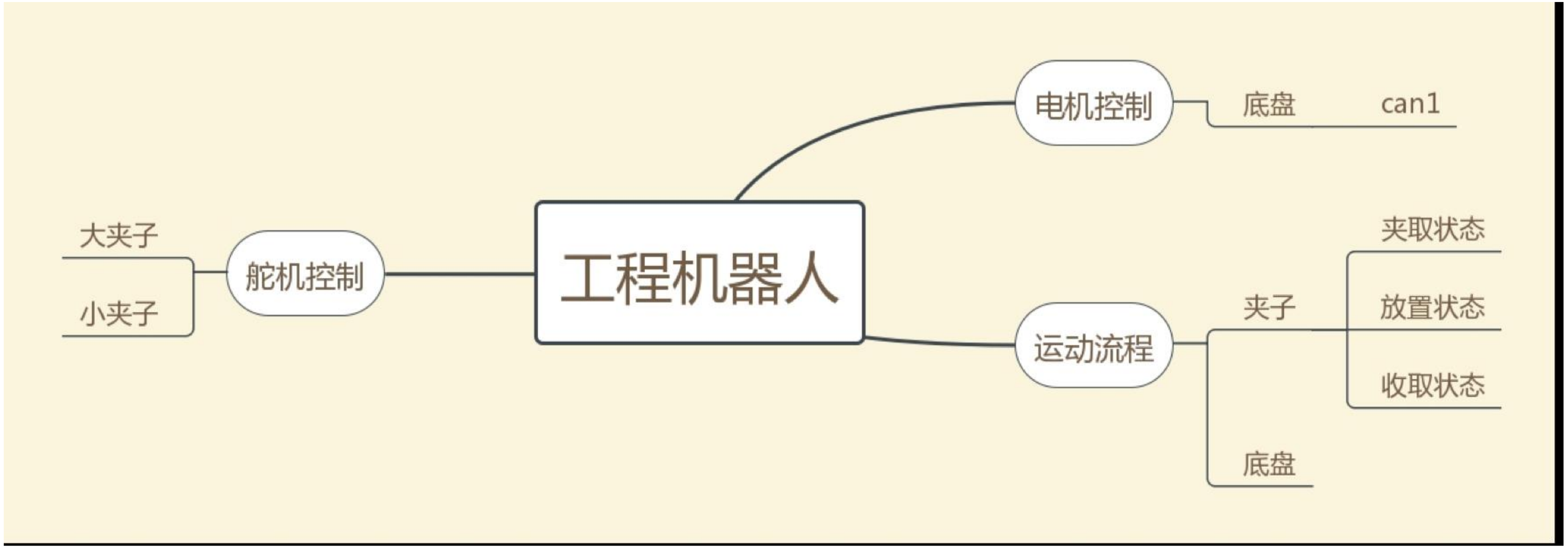
* 7灰度传感器 (3+2+2)

```
[|==1 234 5==|]
 |      - - -     |
 |6  RF-ID  7  |
 |      - - -     |
[|==+ - - - +==|]
```

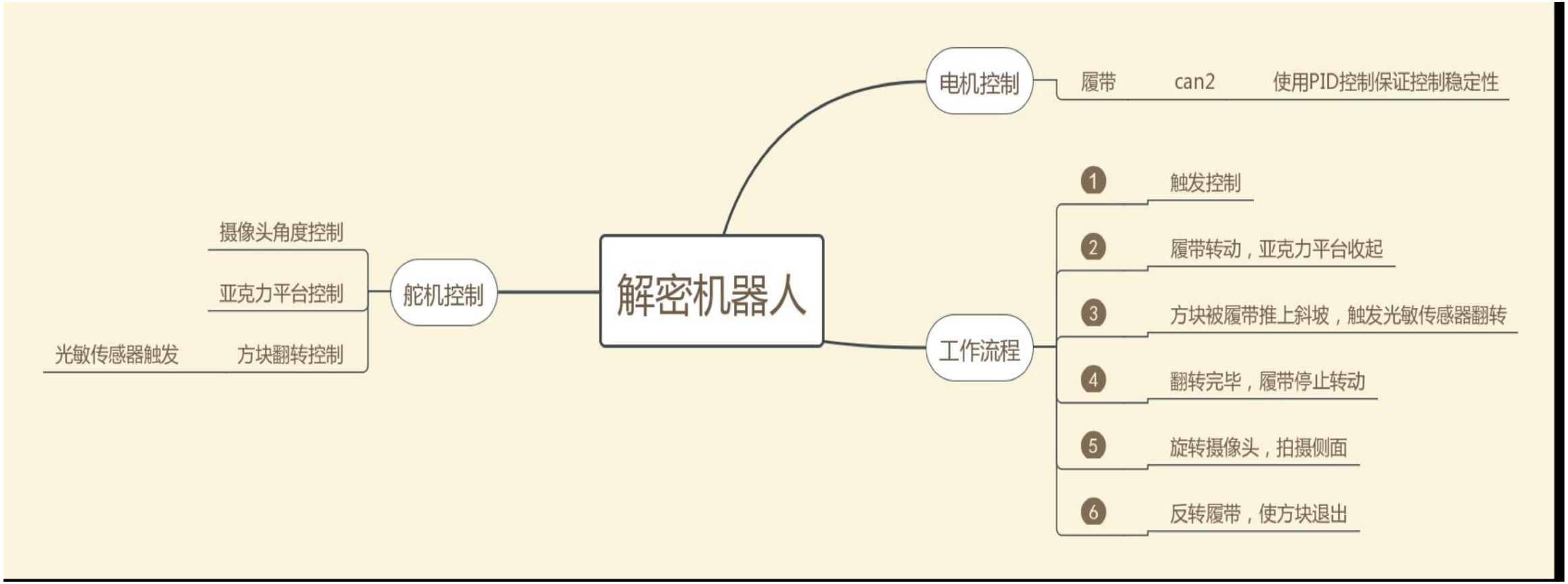
* 2, 4提供小范围旋转矫正
* 1, 5可以在偏度较大时平移车体, 可以实现迅速归位
* 6, 7可以用于十字路口定位辅助, 因为RFID范围较大, 难以实现精确的定位
  > 方便转向, 不容易卡到墙
  > 辅助视觉识别



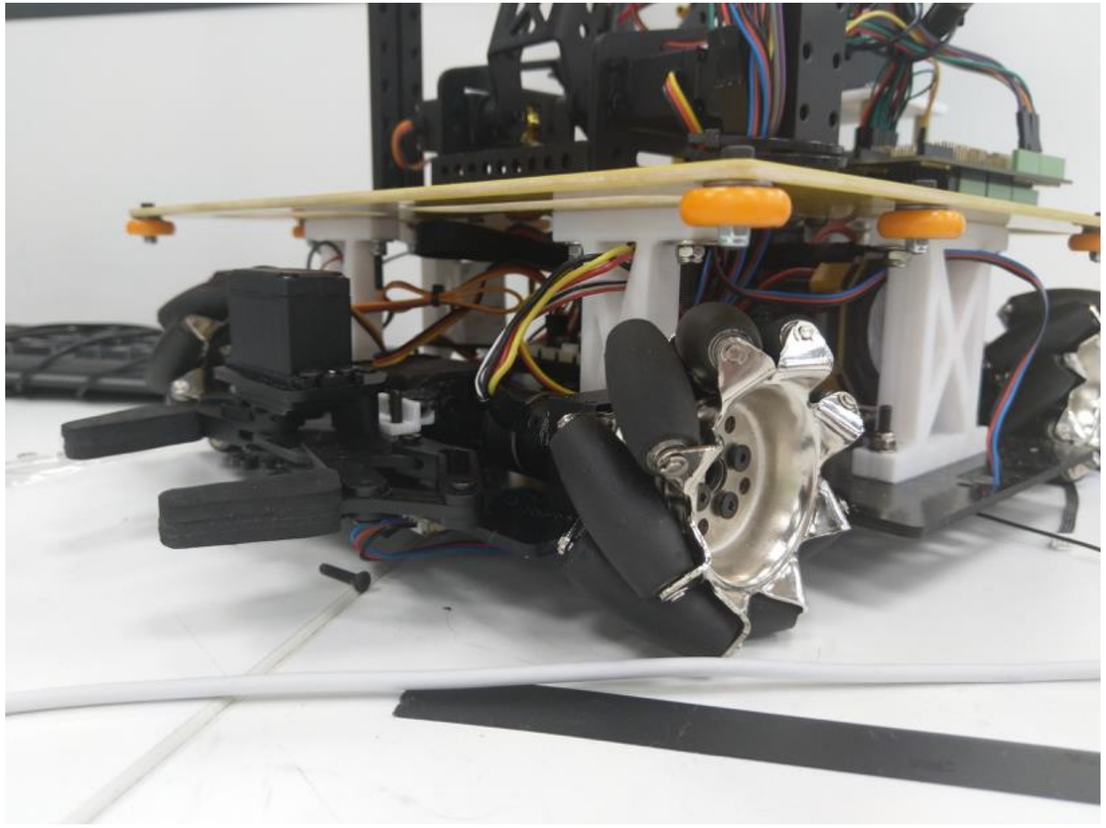
工程机器人控制系统结构



工解密机器人控制系统结构

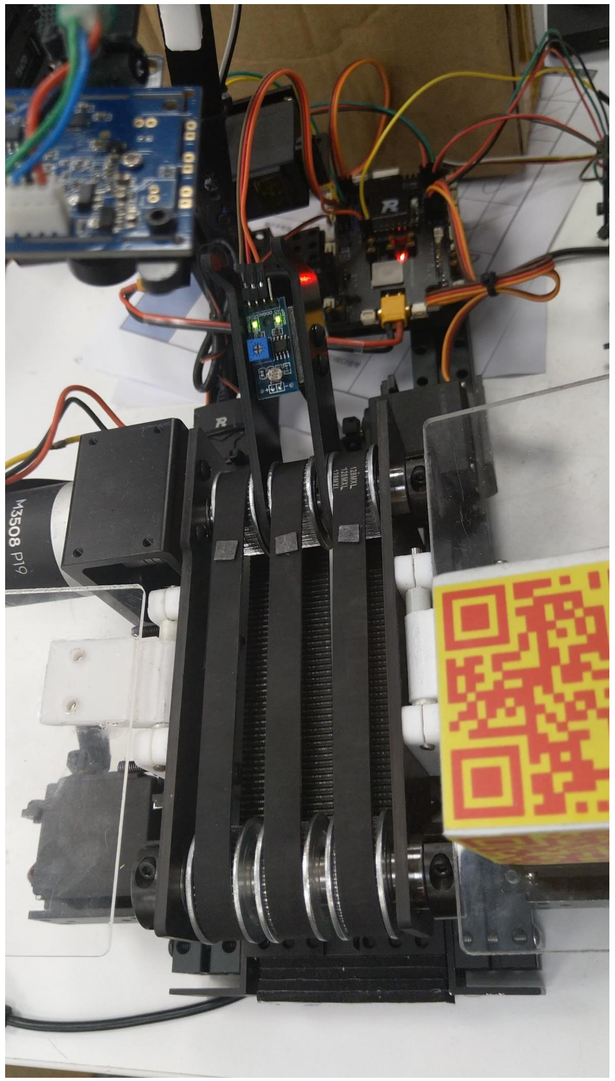


PID 控制



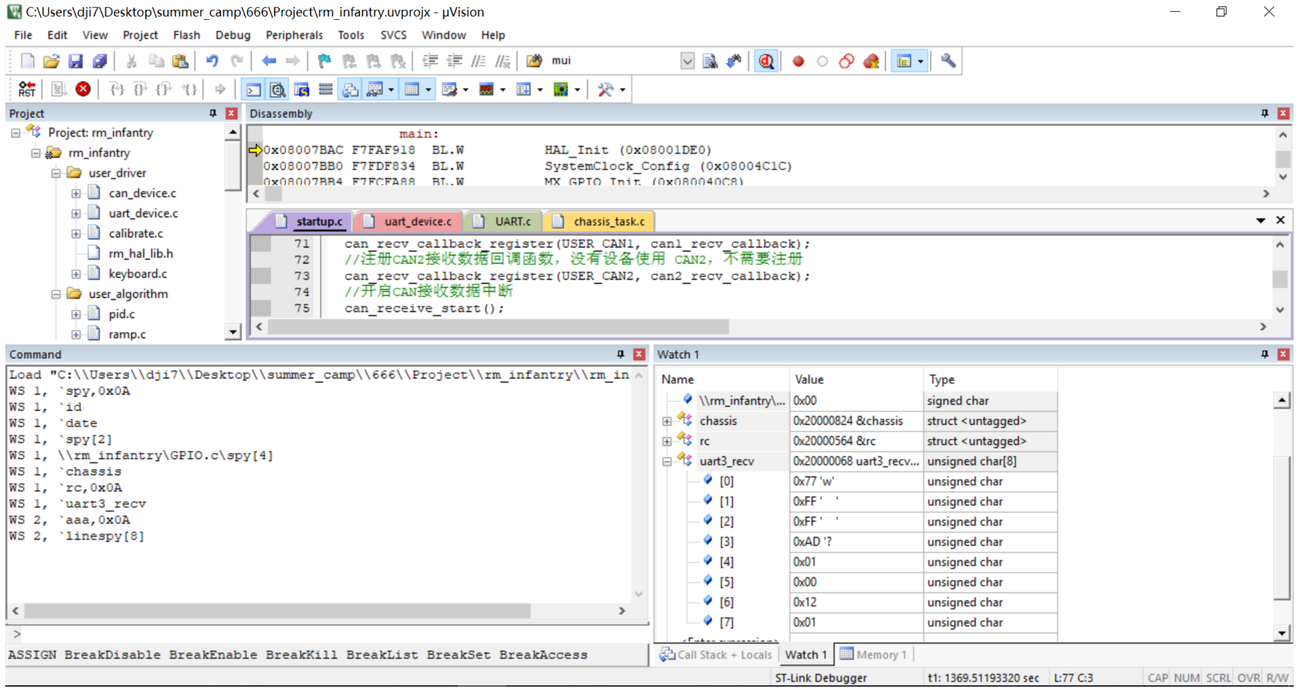
•工程机器人底盘
•底盘的稳定是比赛的基础
•我们通过PID控制减轻底盘在急刹车时的打滑，限制总输出防止超速。满杆转弯时可以无碰撞传过“发夹弯

PID 控制



- 解密机器人履带
- 履带速度稳定奠定翻转方块的基础
- 我们通过在翻转时履带停止，大大增加了翻转的成功率，而在这个过程中，PID控制是不可或缺的

串口通信

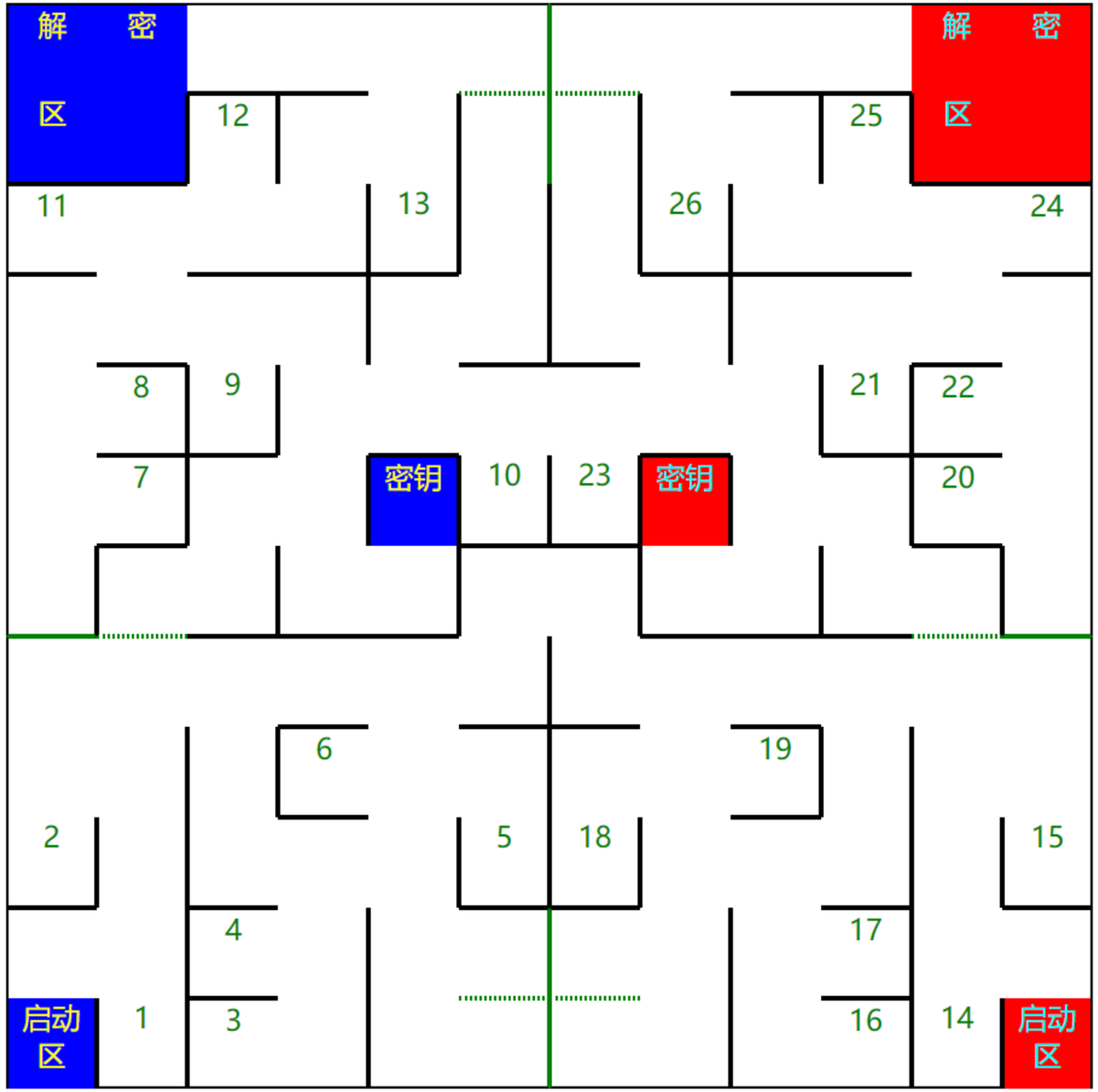


我们在妙算和STM32中搭建了UART通信，将妙算1搭载在工程机器人上，并运行需求性能高的导航算法。

工程机器人上的妙算1

•我们在妙算1运行导航算法，将路径通过UART输出给STM32上。以此实现自动运行。

•我们还准备了冗余方案，与妙算1组成网络，操作手可以指定目标或路径控制机器人运行



算法

- 导航算法
- 路径规划
- 视觉识别

导航算法

- Version 1: 基于Floyd算法预处理出的任意两点间最短路来进行导航。优点为运算量少（时间复杂度为O(n^3+t)），速度快。缺点为该算法无法考虑旋转门带来的影响。
- Version 2.0: 基于SPFA算法，实时更新最短路径。该算法优点在于易于实现，且出现任何问题时都可以立刻刷新，缺点在于它占用了较多运算资源（时间复杂度为O(t*n^2)），且在会车时会出现问题（退一格后掉头走老路）。
- Version 2.1: 改用A*算法求最短路。由于A*是搜索算法，它可以通过在某些地方绕小圈等待旋转门状态切换。而SPFA被设计用于求静态图最短路，可以以局部最优作全局最优，故无法通过绕小圈或者停车等待旋转门。（时间复杂度为O(t*n^2*log n)）

导航算法

- Version 3: 预处理出全图最短路树并使用LCT维护。将旋转门状态变化视为删边和加边操作。随后发现此算法有误（最短路树删边后变化不仅是重新连一条新边）。（时间复杂度为O((n+t)log n)）
- Version 4.0: 每次计算路径时只运行一次SPFA算法，通过留出一些误差时间，消除因车速不均带来的影响（在误差时间内始终可以通过的旋转门才可以通过）。同时实现会车时重新导航。（时间复杂度为O(n^2)）
- Version 4.1: 将SPFA算法换为A*算法。原因同Version 2.1。（时间复杂度为O(n ^2*log n)）

导航算法

- 对于路径规划，我们提出了四种思路：
- 第一种思路是先捡密钥，然后每次选择路程最近的可能存在方块的位置，每次取满方块后就把它放回解密区。这种方法非常简单，但在后期剩下的点可能非常散乱，路径很长
- 第二种思路是使用启发式搜索或者模拟退火，找遍历所有点的最短路径。但是估价函数比较难找，且无法证明这个方案是否最优（我们并不知道密文方块的位置）。
- 第三种思路是人工预设路径。这种方式基本保证路径的优秀性，但是这样非常耗费时间。

导航算法

- 第四种思路是手动设定一部分遍历序列作为，其余的让机器自己找。这种做法糅合了第一种思路和第三种思路，然而实现起来相当繁琐。
- 隐藏思路：用无监督型机器学习，实现真·智能路径规划。但是机器学习数据量极大（粗估10^10^），且算力不足，未被采纳。
- 我们原计划使用第四种思路。但是由于我们的机械爪需要先抓取密钥，再抓取密文。经过一番讨论与尝试，我们最终选择了第一种思路。
- 对于会车时的处理，我们选择在会车时临时单向删去会车边，然后继续前往原定目的地。这样做可以“强迫”寻找一条新的路径，而不是退一格后又调头回来。

视觉

- 任务:
  - 自动识别红蓝双方的方块
- 方案:
  - * 巡线辅助定点拍摄
  - * 裁剪图像，并转换至HSV(Hue-Satuation-Value)空间
  - * 统计红色&蓝色像素点的数量从而确定箱子颜色