3D Printer 多自由度 3D 打印系统控制软件

设计说明书 V1.0

目录

1 引言

1.1 文档编写说明

本文档介绍了基于ROS的多自由度3D打印系统控制软件的硬件和软件基础，详细解释了该系统软件的设计原理，实现方法以及最终效果，同时还说明该软件配置安装和使用的基本流程。

基于 ROS 的 3D 打印系统控制软件可实现的基本功能有初始参数的设置，数据和控制命令的读取，运行状态的检测反馈和动态调整。

本控制系统的人机交互界面上允许用户设置喷头的温度，打印热床的温度，工件的打印密度和打印速度等初始参数，在传统打印机上，上述参数通常情况下设定后直至打印完成是不可更改的，但本控制软件可以实时设定打印喷头和热床的温度，以实现对打印不同阶段打印温度的精准控制。

本控制系统需要读取控制指令文件，控制机械臂六关节连续变化，使热床相对于喷头发生位置和姿态的变化，实现无支撑的 3D 打印，相较于传统 3 轴 3D 打印机使用切片软件生成的 Gcode 来控制电机运动，本系统读取的是自定义点参数文件，该文件由另一款项目中研制的多自由度自适应切片软件来生成，文件规定了机械臂末端的行动路径，读取数据后控制系统会驱动关节沿预定的轨迹运动。

由于该系统应用于实验平台中，所以有丰富的检测反馈功能以反映和记录运动的情况，本系统软件可以实时检测打印喷头和打印热床的温度并在 GUI 界面上显示，还可以实时检测六自由度机械臂平台各个关节的角位置和角速度，实时反馈六自由度平台的沿 x，y，z 方向的线速度和绕 x，y，z 方向的角速度，本系统还带有画图功能，能对上述检测的参数进行作图展示。

1.2 软件设计背景

随着计算机科学和材料工程学的快速发展，3D 打印技术日趋成熟，业内普遍认为 3D 打印技术将对传统的制造业产生革命性变革，促进制造业的快速发展。

在《中国制造 2025》发展规划中，有多处提到对增材制造领域技术的突破需求。在第二部分高档数控机床和机器人发展战略中，指出增材制造是重点发展装备和技术的方向之一，其中指出要重点突破具有系列原创技术的成套装备、专

用材料及工程化关键技术，重点攻克 3D 打印材料的制备、智能软件等瓶颈，打造产业链。在第十部分生物医疗及高性能医疗器械发展战略中，发展医用增材技术作为关键技术被提出，其中具体的方向有重点发展适于 3D 打印技术的可植入材料及装饰技术，重点发展适用于个性化制造的全方面解决方案，包括监测、计算机辅助设计与制造技术。

3D 打印中较为成熟的技术有熔融沉积成形（FDM）、粉末粘接（3DP）、选择性激光烧结（SLS）、分层制造（LOM）和光固化（SLA）等技术。其中 FDM 技术的 3D 打印机被广泛运用于教学，医疗，原型制造和模具制造等领域，在 3D 打印技术中由于很高的普及度。

目前主流的 FDM 打印机是 3 轴的，有 Delta 型和 Cartesian 型，3 自由度的打印系统在打印过程中存在的一些问题有支撑材料过多，可打印模型较小，打印模式单一，交互性不好等不足之处。于是学界提出增加自由度的的方法改善上述问题，多自由度的打印模型的优势在于有更加灵活的打印轨迹设计和更加丰富的机构设计，可以大幅度减少支撑结构甚至可以实现无支撑 3D 打印。

目前可以市面上可以购买的五轴 3D 打印机有英国公司的 5AxisMaker，Q5D Technology，印度公司的 Ethereal Halo 和波兰公司的 Vshaper 5AX，上述产品的售价范围通常在数十万到百万人民币。这些国外生产商因为掌握了核心技术，形成了市场的高竞争力，所以产品的价格昂贵，而且这些最前沿技术的设备有较为严格的市场准入制度，严重制约了我国智能化增材制造领域的发展。因此推进国有智能化 3D 打印控制系统，提高国产数字模型处理软件产品的技术水平、配套能力和市场竞争力，对促进我国智能制造的发展有着重要的意义。

3D 打印技术涵盖了多个学科的前沿技术，包括计算机三维模型处理，运动轨迹的智能规划，人机交互界面，多传感器检测等等，研发团队要有多学科交叉融合的能力，鉴于 3D 打印技术会给传统的制造业带来全新的变革，促进制造向着模块化、个性化、智能化方向发展，推进 3D 打印理论和技术突破对我国制造业的高质量发展有着重要作用。

2 约束条件

2.1 硬件环境约束

本系统运行在装有 lunix 系统的高性能计算机上，该 PC 上位机通过串口数据线连接 Arduino 控制主板，Arduino 主板连接基于温敏电阻和继电器设置的控制电路，实现对喷头温度，热床温度，挤出物料速度控制，同时 PC 上位机通过网络传输数据线与一 UR5 型号的六关节机械臂相连，机械臂末端安装转接头和热床整体结构如下图所示。

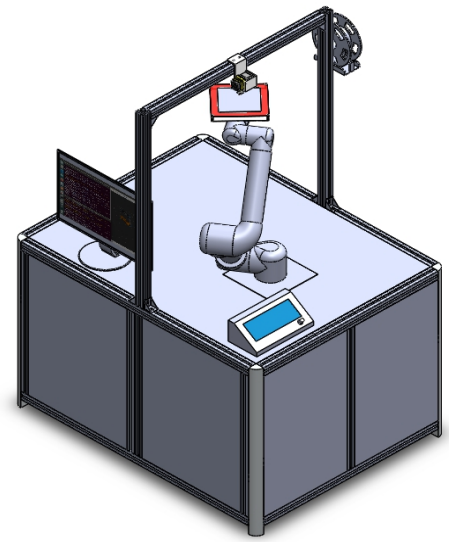

图 1 多自由度 3D 打印控制系统硬件平台

2.2 软件依赖约束

本软件是在 lunix 系统下开发的，用到了开源的 ROS 系统开发的语言主要有 C 语言和 Python 语言。C 语言用于 Arduino 控制单元的内置程序，同时也依赖 Python 里的 PyQt5 函数库制作 GUI 界面，依赖 Serial 函数库用于与单片机串口相连，依赖 Matlibplot 函数库用于绘制图像。特别申明上述依赖的函数库以及功能包等均为开源程序，本系统需要用到部分开源依赖库里的程序，所以在使用本系统软件时需要做相应的依赖环境配置。

2.3 用户界面约束

多自由 3D 打印技术尚处于发展阶段，目前的工业机器人多是采用示教的方式实现运动控制。不能实现对机械臂末端的位置，姿态，速度随时间的精准变化，本系统利用运动学逆解和机器人运动轨迹规划解决这一问题使得机械臂平台能够实现 3D 打印。目前的 3D 打印机温度，打印速度等参数均由单片机嵌入的程序设定，操作者难于实时修改，可拓展性和交互性差，本系统通过串口通信和自定

义数据协议来解决这一问题，使用户可以通过 GUI 界面上的按钮和选项框实现快速而准确的调节。现有的 3D 打印机只包含温度一项信息反馈，容错性和检测性不足，本系统则对 3D 打印过程中的温度，控制驱动，平台运动参数进行了全面的检测，有利于增强系统对加工方式，工件材料，工件结构等的适应性。

3 软件设计原理

3.1 Arduino 单片机的内置程序设计

Arduino 主控板有数据量输入和输出引脚，模拟量输入和输出引脚，这里用数据量输出引脚配合继电器来控制热床和温度的加热，用模拟量输入引脚读取电压值转化为相应的温度参数。

用 PinMode()函数设置引脚的输入和输出模式，digitalWrite()函数来输出引脚的高低电平，程序中定义了 motor_active 和 heat_active 两个 bool 变量来作为步进电机转动和加热功能引脚的使能标志变量，当上述变量的值为 true 时，相应模块功能判断语句才能通过启动。

loop()循环函数里将读取到的模拟量电压值由计算公式转化为相应的摄氏温度值，涉及到的变量转换代码为：

```
heat_voltage = analogRead(TEM_READER_PIN)*0.004883;
heat_tem = 1/(log(heat_voltage/(5-heat_voltage))*0.000253165+0.003354) - 273.15;
bed_voltage = analogRead(BED_READER_PIN)*0.004883;
bed_tem = 1/(log(bed_voltage/(5-bed_voltage))*0.000253165+0.003354) - 273.15;
```

在 loop()循环中还对温度进行了判断，判断计算出的温度是否处于合适的区间，如果判断不成立则做出相应的调整，这部分的判断逻辑为：

```
if( bed_tem > bed_aim-10 && heat_tem > aim_tem-10 && heat_active){
    if(heat_tem < aim_tem+10){
      digitalWrite(HEATER_PIN,HIGH);
    }
    if(heat_tem > aim_tem+20){
      digitalWrite(HEATER_PIN,LOW);
    }
```

```
if(bed_tem < bed_aim+10){
   digitalWrite(BED_HEATER_PIN,HIGH);
}
if(heat_tem > aim_tem+20){
   digitalWrite(BED_HEATER_PIN,LOW);
}
```

如果温度偏低，会将对应引脚输出高电位加热，反而如果温度偏高，则会将对应的引脚输出低电位不加热，从而保证两个目标温度在预定区间变化，只有当检测的温度在预定区间时才输出脉冲控制电机转动，这里设置的是向步进电机输出方波信号，周期为 2000μs。

3.2 Serial 串口通信协议设计

计算机和 Arduino 之间的通信是通过串口来实现的，对于 Arduino 主控板，设定波特率为 9600，通过 Serial.print()函数和 Serial.read()函数来输出信号和读取信号，为了解决同时收发信号造成数据链路堵塞的问题，这里使用了 serialEvent()“伪”中断机制，每个 loop()循环结束后，Arduino 都会执行 serialEvent 里定义的程序，将数据的读取程序方在此函数中就可以将串口数据的输出和读取流程分开来，实现同时信号的输入和输出。

在系统主控界面，可执行操作有预定温度的调节、加热的使能控制，物料挤出的使能控制，设定的通信数据为 utf-8 的字符编码格式，传输的字符判断逻辑示意图如下：

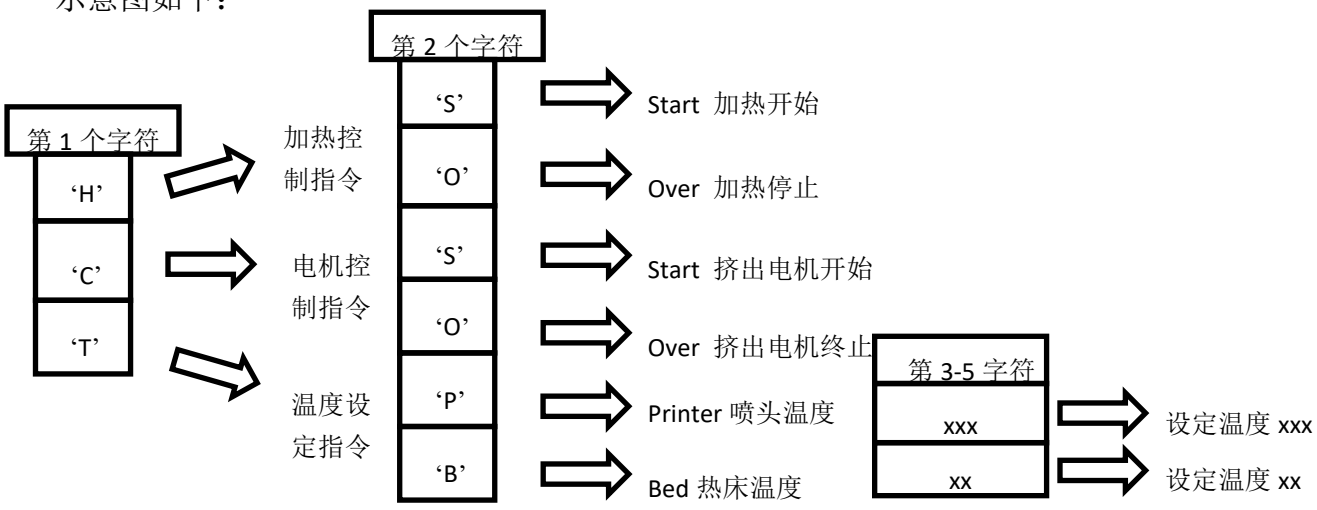

图 2 字符控制指令判断逻辑示意图

3.3 ROS 六自由度机械臂平台运动规划设计

本系统使用 ROS 里的相关功能包来实现对六关节机械臂的控制，首先要获取 UR5 机器人的 urdf 文件，此文件中描述了各关节、连杆的相对位姿和各关节的坐标变换矩阵，其次要使用 Moveit！功能包里的 assistant 工具定义机器人的运动规划关节组，自碰撞矩阵，末端关节等参数配置，最后就可以使用 Moveit Commander 的相关函数来实现部分功能。

在本系统中引入了仿真模拟功能区，仿真模拟是通过子进程中运行程序代码中的 demo.launch 文件打开 rviz 可视化界面，启动 move group 功能包，连接 fake controllers 服务节点来进行工作的，整体结构如下图所示：

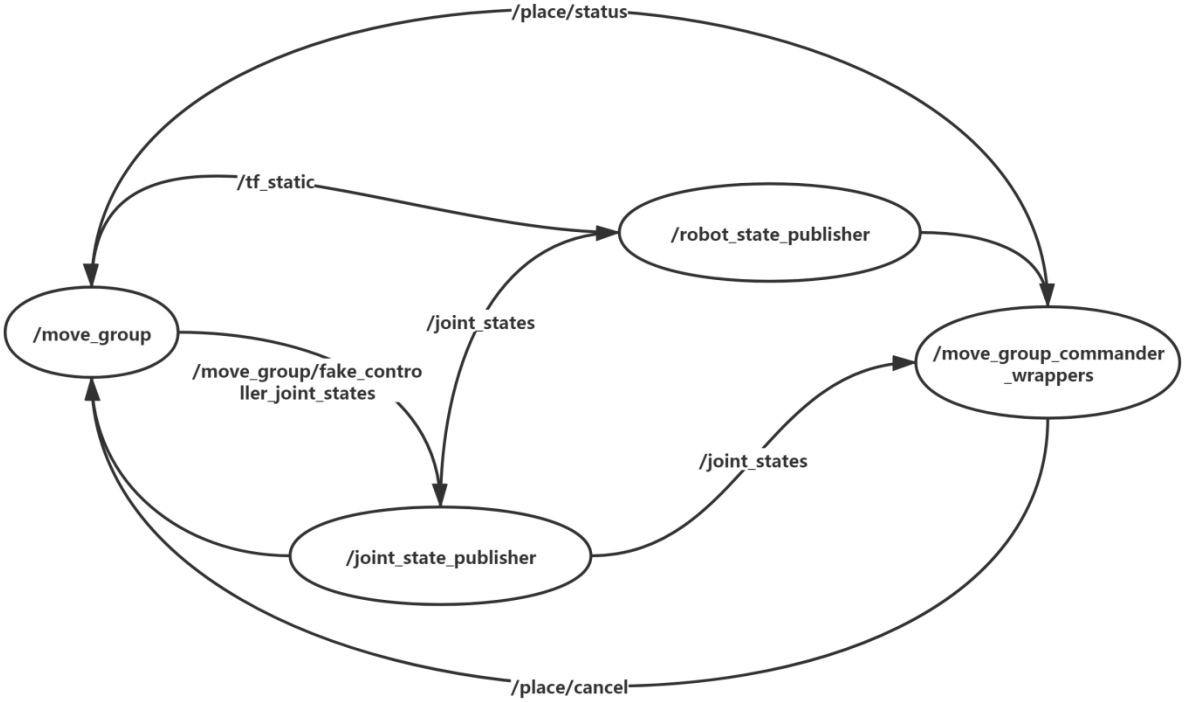

图 3 模拟仿真时节点通信示意图

仿真模拟的优势是可以将切片软件生成的控制代码在虚拟的环境下先仿真一遍，观察机械臂运动的路径是否符合预期，运动过程是否遇到干涉等现象，这项功能可以大大增加算法设计的效率，增强工件打印的安全可靠性。

连接真机的功能是通过子进程中执行程序代码中的 start.launch 文件来实现的，该文件会启动 ur bringup 功能节点连接 UR5 机器人的电机控制器，会启动 rviz 可视化界面，启动 move group 功能包提供机器人运动反解功能。在此系

统的中多自由度平台的运动控制流程为：

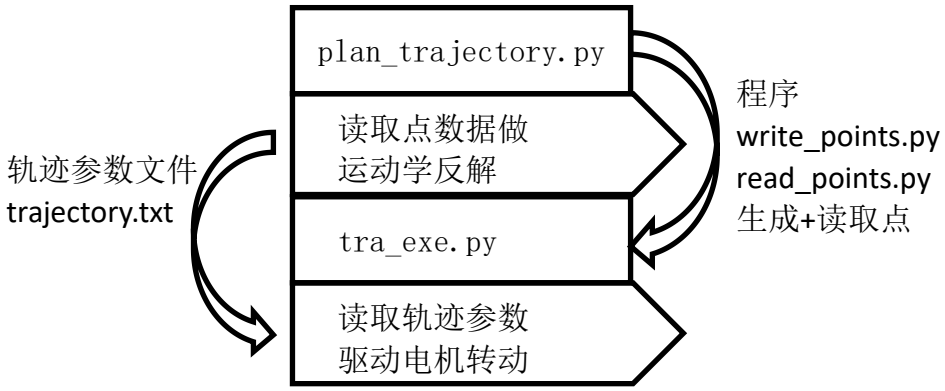

图 5 字符控制指令判断逻辑示意图

在 plan_trajectory.py 程序中，先调用 read_points 函数对 points.txt 点数据文件进行读写操作，将点数据暂时存入到内存中，启动并初始化 Moveit Commander 类，定义规划关节组为“manipulator”，即 Moveit assistant 中配置的关节名称，用 get_end_effector 函数选定机械臂末端关节，用 get_current_ pose 来获得末端关节的位姿，因为运动控制的主体是机械臂末端的热床，所以这里的基本思路是定义一系列末端关节的位姿，然后将这些位姿参数传给 KDL 运动学逆解插件就可以得到对应的关节空间下的角位置值，这里用到了 ROS 的 geometry_msg 定义的 pose 数据结构，其基体结构和参数为：

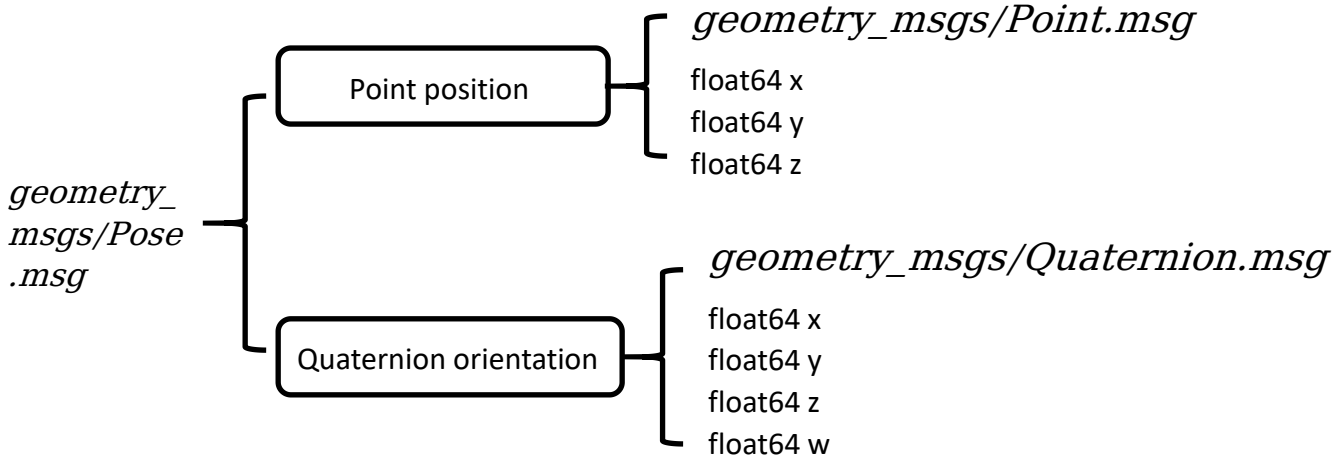

图 6 平台位姿 Pose 数据结构

通过 compute_cartesian_plan 函数，可以得到上述位姿对应的轨迹参数 plan，这里的 plan 是 moveit_msgs/RobotTrajectory 型的数据结构，这里定义

了一个 tra_stroe 的类方法，用于将计算得到的路径写入到名为“trajectory.txt”的文件中去，写入的轨迹最主要信息有一系列的关节控制量，其中包括关节的角位置随时间的变化，关节角速度随时间的变化以及关节力随时间的变化，完成轨迹的运算后，通过 tra_exe.py 程序就可以直接读取上述存储的数据，然后赋值给临时变量将数据通过 action 的通行机制传送至机器人的 controller 控制器里。

需要补充说明的是，因为 3D 打印设备每次加工前都要回到起始位置，俗称的“回零操作”，所以这里定义了回零程序 go_home.py，此函数运用的是 moveit_

Commander 里的关节驱动功能，用到的主要是 set_joint_value_target 函数，此函数输出参数为目标关节位置，返回的是规划好的轨迹，为了防止回零的时候机器人关节与桌面，支撑架等外部物体发生碰撞，本系统采用的是从大关节至小关节依次回零的方式：

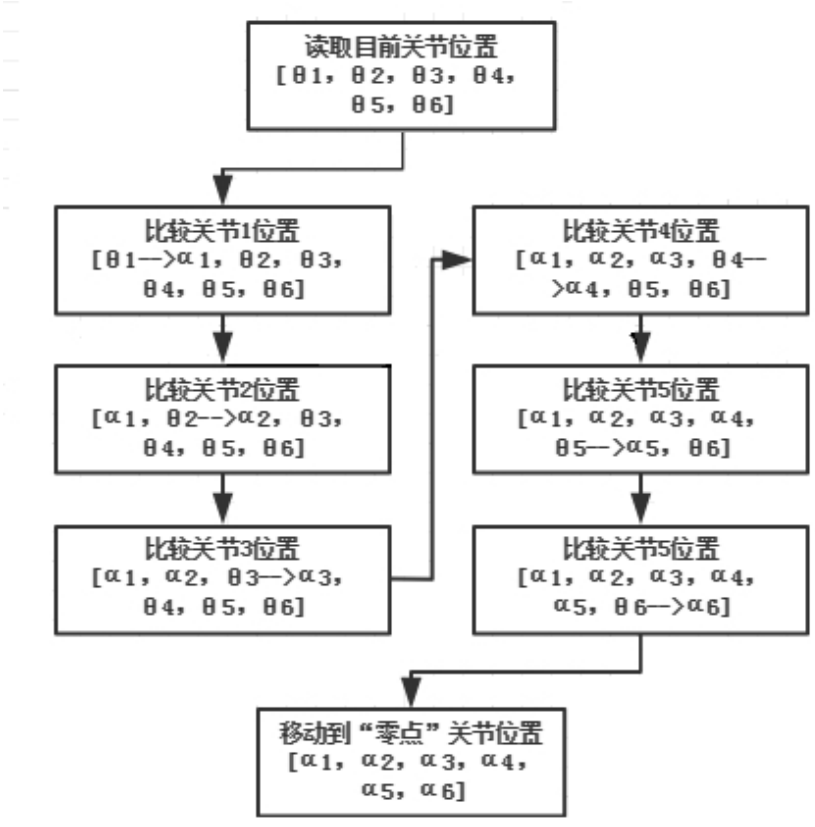

图 7 机械臂平台回零操作原理

3.4 传感信息监测程序设计

3.4.1 边缘处理单元 Arduinio 板检测的温度数据

微机读取电压计算温度的原理上文已经说明，在 Arduinoi 的 loop()主循环函数中，先将温度量进行 utf-8 字符串编码，然后用 Srial.prnt()函数从串口发出，计算机上接受串口信号是通过 Ard.py 代码实现的，需要用到的第三方依赖库是 serial，基本流程是首先用 serial.Serial(port,baudrate)建立串口示例对象，此函数的第一个输入变量是串口连接设备的端口，第二个输入变量是设置匹配的波特率，因为 Arduino 上设置的波特率是 9600，所以在创建此实例是也设置波特率参数为 9600，在 Adr 的类下定义了一个 tem_read()函数，收到传过来数据的处理流程为：

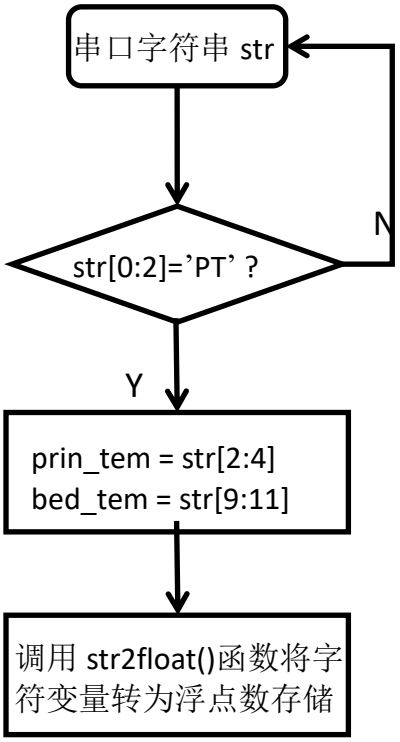

图 8 读取串口传出的温度信息

3.4.2 获取关节角位置和速度

获取机器人各关节的角位置和速度是通过 speed_monitor_ui.py 程序实现的，在类 SpeMon 中首先用 rospy.init_node()函数初始化 ros 节点，调用 movegroup。

commander 连接关节规划组，用 get_curerent_joint_vlues（）函数获得机器人控制器传回的数据，这里传回的数据类型是一个包含六个浮点数的数组，浮点数依次表示各关节的角位置弧度制数值。

在机器人运动的过程中，ros 中 moveit 的 robot_state_publiaher 节点会向外发出/joint_states 话题信息，在 w_m.py 程序中，新建了一个‘w_listener’节点用来监听此数据并将其打印出来传回主程序变量中。

3.4.3 由关节参数计算末端位姿和速度

计算 UR5 的末端位姿和速度时需要用到机器人学中的运动学正解，基本法理论是 DH 参数和 Jacobian 矩阵计算，如图是 UR5 的关节关系简图：

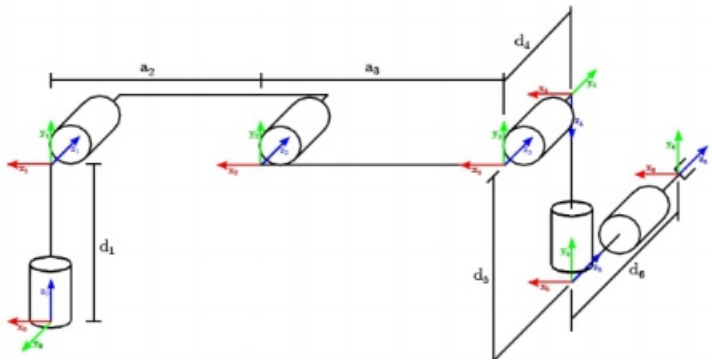

图 9 UR5 各关节几何关系示意图

表 1 UR5 关节 DH 参数表格

i	d_i	a_i	α_i	θ_i
1	d_1=89.459	a_1=0	α_1=π/2	θ_1
2	d_2=0	a_2=-425	α_2=0	θ_2
3	d_3=0	a_3=-392.25	α_3=0	θ_3
4	d_4=109.15	a_4=0	α_4=π/2	θ_4
5	d_5=94.56	a_5=0	α_5=-π/2	θ_5
6	d_6=82.3	a_6=0	α_6=0	θ_6

相邻的两个关节坐标系转换矩阵为：

$$
{}_{i}^{i-1}T = \begin{bmatrix} \theta_i & \sin\theta_i & 0 & 0 \\ \sin\theta_i & \cos\theta_i & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & d_i \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos\alpha_i & -\sin\alpha_i & 0 \\ 0 & \sin\alpha_i & \cos\alpha_i & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & a_i \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}
$$

相乘化简之后得到：

$$ {}^{i-1}_{i}T = \begin{bmatrix} \theta_i & -\sin\theta_i\cos\alpha_i & \sin\theta_i\sin\alpha_i & a_i\cos\theta_i \\ \sin\theta_i & \cos\theta_i\cos\alpha_i & -\cos\theta_i\sin\alpha_i & a_i\sin\theta_i \\ 0 & \sin\alpha_i & \cos\alpha_i & d_i \\ 0 & 0 & 0 & 1 \end{bmatrix} $$

将表 1 中的 D-H 参数代入上式可以得到从关节 1 到关节 6 的坐标变换矩阵：

$$ {}^{0}_{1}T = \begin{bmatrix} \theta_1 & 0 & \sin\theta_1 & 0 \\ \sin\theta_1 & 0 & -\cos\theta_1 & 0 \\ 0 & 1 & 0 & d_1 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad {}^{1}_{2}T = \begin{bmatrix} \theta_2 & -\sin\theta_2 & 0 & a_2\cos\theta_2 \\ \sin\theta_2 & \cos\theta_2 & 0 & a_2\sin\theta_2 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} $$

$$ {}^{2}_{3}T = \begin{bmatrix} \cos\theta_3 & -\sin\theta_3 & 0 & a_3\cos\theta_3 \\ \sin\theta_3 & \cos\theta_3 & 0 & a_3\sin\theta_3 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad {}^{3}_{4}T = \begin{bmatrix} \cos\theta_4 & 0 & \sin\theta_4 & 0 \\ \sin\theta_4 & 0 & -\cos\theta_4 & 0 \\ 0 & 1 & 0 & d_4 \\ 0 & 0 & 0 & 1 \end{bmatrix} $$

$$ {}^{4}_{5}T = \begin{bmatrix} \cos\theta_5 & 0 & -\sin\theta_5 & 0 \\ \sin\theta_5 & 0 & \cos\theta_5 & 0 \\ 0 & -1 & 0 & d_5 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad {}^{5}_{6}T = \begin{bmatrix} \cos\theta_6 & 0 & -\sin\theta_6 & 0 \\ \sin\theta_6 & 0 & \cos\theta_6 & 0 \\ 0 & 0 & 1 & d_6 \\ 0 & 0 & 0 & 1 \end{bmatrix} $$

最终从基坐标系到末端坐标系的矩阵为：

$$ {}^{0}_{6}T = {}^{0}_{1}T \cdot {}^{1}_{2}T \cdot {}^{2}_{3}T \cdot {}^{3}_{4}T \cdot {}^{4}_{5}T \cdot {}^{5}_{6}T $$

上述运算是有源码中的 Forward_kinematics.py 程序实现的，改程序需要输入六个关节角度变量，返回的是末端相对于 Base 坐标系下的位置和相对于 x，y，z 坐标轴的旋转角弧度值。

在做机器人运动学求解时，需要用到矩阵运算，所以这里定义了一个 Matrix 类，类的属性和成员函数声明都在 matrices.py 程序中，并且其中定义了 mult()函数，该函数输入两个矩阵对象，比较第一个矩阵的列和第二个矩阵的行，如果数目相等则进行矩阵相乘运算，不相等则输出不能相乘的提示信息，在求运动学正解时，所用到的矩阵和相乘，都是采取上述方法实现的。

为了获得机器人末端的速度参数，本系统采用了雅各比矩阵理论，由各关节的角速度求出机器人末端热床的线速度和角速度，雅各比矩阵的获取用到了 moveitcommander 里的 get_current_jacobian()函数，将机器人关节控制传回的角速度数据与上述函数得到的雅各比矩阵相乘，就可以得到一个六个元素的数

组，这六个元素分别是，热床沿 x，y，z 方向的线速度以及绕 x 轴，y 轴，z 轴的角速度,将该信息通过 topic 话题机制发送出去,就可以被其他程序接受使用，并最终展现在 GUI 界面上或者图像上。

3.4.4 末端轨迹可视化

考虑到多自由度 3D 打印系统的末端轨迹更多是空间曲线，为了有更好的观察效果,本系统开发了末端轨迹可视化的功能,该功能是通过 show_endeffector_path.py 程序来实现，基本思路是创建新节点“endeffector_visualization”，并发布话题“show_path”，话题的数据类型是 Path，Path 类型主要有两部分组成，一是消息头 header，二是数据 PoseStamped，程序将 header 里的时间戳设定为当前时间，相对坐标系是“/world”，初始化 moveitcommander，使用其中的 get_current_pose 获得机器人末端的位姿 Pose，然后幅值给 Path 中的 PoseStamped，最后按 100hz 的频率将消息发出。

最后 ros 中的 rviz 可视化工具接收 Path 信息并在三维图像上显示,就可以实现机器人末端轨迹可视化的功能。此功能的好处在于可以实时观察末端的运动轨迹，并与预期的设定做比较，在仿真阶段使用该功能可以有效的判断运动过程的有效性。

3.5 GUI 界面设计

3.5.1 界面布局

本系统的 GUI 界面由两个主窗口组成，第一个主窗口主要负责 3D 打印部件参数的设置，文件的导入，机器人的设计、连接、仿真和运动等功能（如图 10），第二个主窗口有两个分页，第一个分页，监测的是机器人六个关节的角位置和角速度（如图），第二个分页监测的是机器人末端热床的位姿变化和速度变换（如图），两个分页上都有启停按钮和画图的控制按钮。

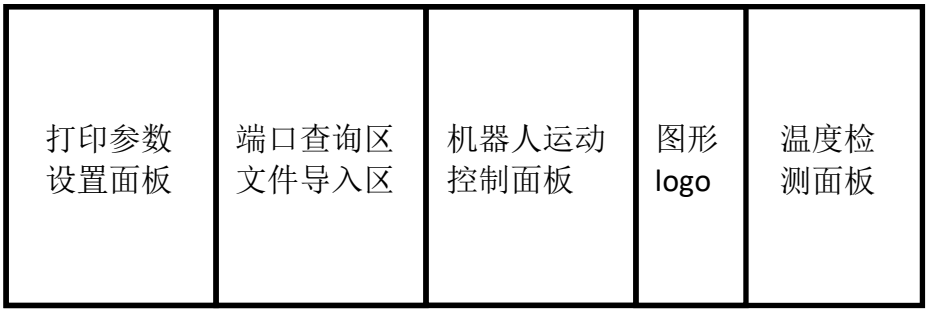

图 10 主窗口 1 布局

关节名称	关节 1-6 的名称
角位置	关节 1-6 的角位置
角速度	关节 1-6 的角速度
启停按钮	绘图按钮

图 11 主窗口 2 分页 1 布局

笛卡尔系位置参数	线速度分量 Vx	线速度分量 Vy	线速度分量 Vz	线速度总量 V
笛卡尔系速度参数	角速度分量 Wx	角速度分量 Wy	角速度分量 Wz	启停按钮 V 绘图 W 绘图

图 12 主窗口 2 分页 2 布局

本系统的 GUI 图形界面是使用 PyQt5 函数库来开发的，界面布局需要用到 QWidget 里的 QHBoxLayout、QVBoxLayout 和 QGridLayout 类，它们分别提供水平布局，垂直布局和网格布局功能，在单一窗口中，各个可视化元件都是放置在这些布局下的,布局功能可以相互嵌套,如在主窗口 1 中各板块之间是水平布局，而板块内部嵌套的是竖直布局，在主窗口 2 中，主体是按 1:3 比例的水平布局，右侧参数状态栏是网格布局，网格布局中放入文本标记或者按钮元件。

3.5.2 信号的反映机制

GUI 界面的主要职能是接收用户发送的指令，包括键盘按键，文本输入，鼠标点击等操作，在 QT 中，每个 GUI 界面执行是都是使用 app.exec_()函数，将窗口运行设置为循环函数，每个循环中都有读取消息缓存的函数，于是便实现对主窗口操作行为的不断监测，QT 将监测到的行为操作定义为 signal 信号，如果当相应信号产生时执行预定的相应程序，就需要设立 slot 槽，signal 由窗口对

象传输至 slot 槽连接的执行程序，便能将用户的操作转化为程序的执行。

本系统需要 GUI 界面显示参数和提供控制按钮，部分参数和控制按钮是与 ROS 程序相连的，通过 Qt 来运行 Ros 程序需要用到 Python 里的 subprocess 函数库，该函数库可以通过程序语句来开启 shell 并输入命令，同时可以读取输出的信息 stdout，将输出的 stdout 信息进行处理便可以得到有效数据存储在窗口类的属性中，并通过 QLabel 类的 setText()函数来显示文本信息。

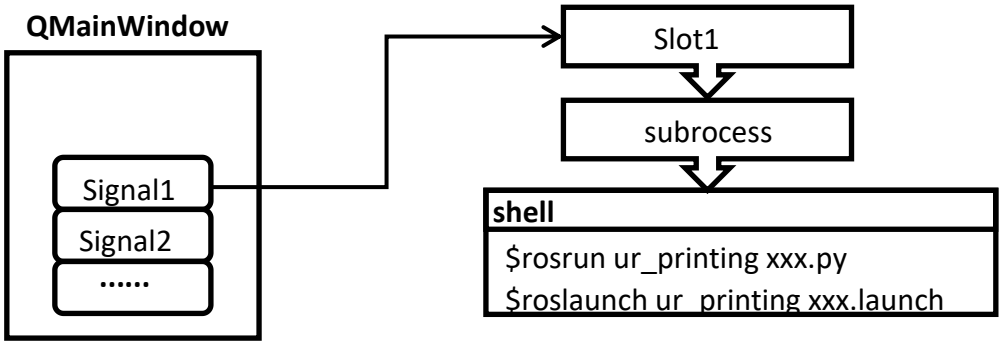

图 13 signal&slot 信号传输示意图

3.5.3 检测数据图像生成

在机器人运动的过程中，关节角位置和角速度通过 robot_state_publisher 节点发出，话题的名称为/joint_states，热床的线速度和角速度通过主窗口节点发出，话题的名称为/plat_sates。

本系统中作图是通过 Ros 里的 rqt_plot 插件来实现的，该插件可以将接收到的话题参数作为纵坐标，以时间作为横坐标，绘制时变函数，函数变化的频率与话题发出时设定的频率相同，在 GUI 界面中，只需要按下按钮就会自动调用内置函数，对关节运动参数和热床末端运动参数进行绘图操作。

4 软件安装配置和使用说明

4.1 软件的安装配置

本系统的搭载在 lunix 系统上，基于 ROS 的 kinetic 版本开发，首先要配置相关环境，unbuntu 系统软件和 ROS 软件都是开源的，可以从官网上下载和安装，ubuntu 官方网站为：https://ubuntu.com/

ROS 官方网站为：https://www.ros.org/

通常情况下，ubuntu 系统自带 python 编译器，如果没有的话可以用命令行：

sudo apt-agt install python 来安装，如果版本较低，可以使用命令行：sudo apt-get update python 来更新软件。

将上述环境配置好了之后，创建 ROS 的工作空间 catkin_ws，新建文件夹 src，将本系统软件复制进该文件夹内，如下图示所示：

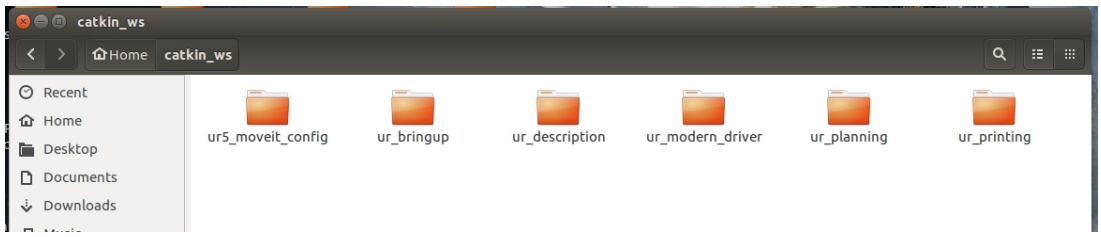

图 14 文件夹布局

上图中的文件夹都是 ROS 中的功能包，ur5_moveit_config 文件夹里包含着 moveit assistant 工具生成的 ur5 机器人配置文件，ur_bringup 功能包可实现 PC 机与 ur 机器人控制器通过网线连接通信，ur_description 功能包包含了 ur5 机器人的模型描述文件和坐标变换矩阵，ur_modern_driver 功能包用于连接和控制 ur5 六个关节驱动器，ur_planning 提供运动规划算法，ur_printing 功能包是本项目开发的多自由 3D 打印系统软件。

4.2 软件的使用说明

4.2.1 控制主窗口

环境配置完成，功能包安装完成之后，就可以正式使用本系统软件，首先用 Ctrl+Att+T 打开 shell 终端，输入“cd catkin_ws/src/ur_printing/scripts”进入系统工作路径，通过“ls”指令可以查看当前路径下的文件，如图 15 所示，然后输入“python ui.py”打开主窗口 1。

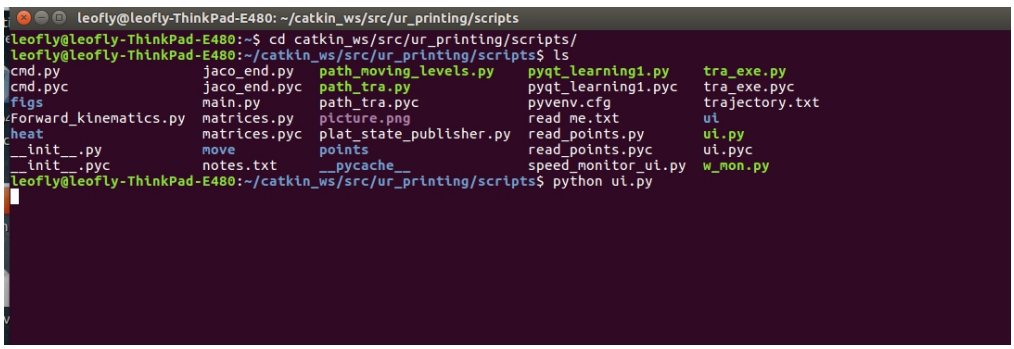

图 15 进入工作路径

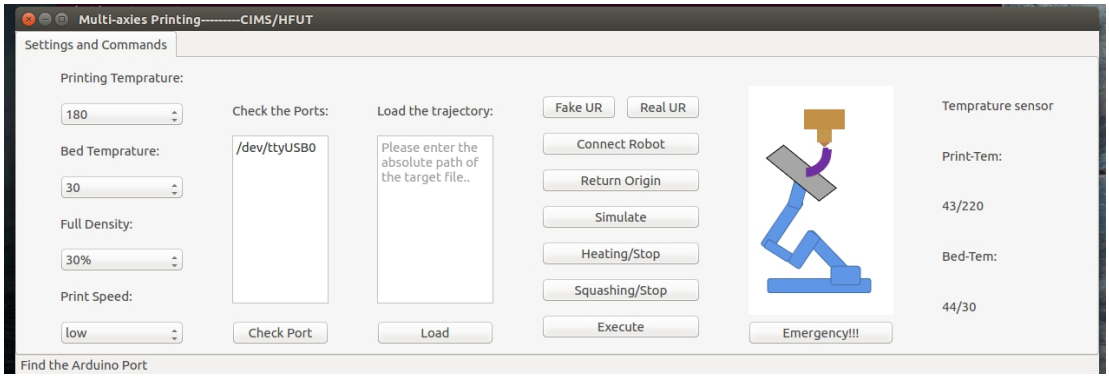

图 16 主窗口 1

主窗口打开默认是在下半个屏幕的位置，可以通过鼠标来移动位置和放大缩小。主窗口的第一列有 Printing Temperature（打印温度）、Bed Temperature（热床温度） 、Full Density（填充密度）和 Print Speed（打印速度）四个参数设定的菜单栏，打印温度的设定范围在 160-240° C，热床温度在 20-60° C，选中相应的数值后，最下方的菜单栏会有提示语句，说明设置完成。

主窗口的二列是 Check the Port（查找端口），由于 Arduino 串口通信时要输出端口名称，所以这一列最下方的 Check Port（查找端口）按钮按下后会自动监测计算机的端口连接信息并返回串口设备的名称，用于创建串口 serial 的对象。这一步操作应该是最先完成的，否则将监测不到 Arduino 主控板，同时注意在 unbuntu 系统中，每个连接设备都用访问权限设置，这里要打开该端口设备的读写权限才能正常访问，基本操作时打开 shell 终端，输入命令行“sudo chmod 777 xxx”，其中 xxx 指的是设备名，然后输入用户密码即可。

主窗口的第三列是输入导入文件的名称，该文件是模型切完片后生成的路点数据文件，用于规划机器人的行动路径。

主窗口第四列有一系列控制本 3D 打印平台的按钮，下面对它们的功能依次展开。

“Fake UR”（虚拟的 UR 机器人），“Real UR”（真实的 UR 机器人），这两个按钮设置真假机器人，按下“Connect Robot”（连接机器人）后，会打开 rviz 可视化界面，如果选择假机器人则打开仿真机器人，连接仿真控制器，如果选择真机器人则连接真实的 UR 机器人，连接真实的关节驱动控制器。

“Return Origin”（回零）按钮按下之后，机器人会回到默认的零点位置，

即打印的起始位置，如图 17 所示，并将系统状态在最下方的状态栏显示出来。本设备正常使用时，都需要做提前仿真生成轨迹文件，按下“Simulate”（仿真）按钮之后，系统就会读取路点文件，根据机器人目前的位置，做路径规划，在路径规划的过程中，用户可以在 rviz 三维面板是哪个看到规划的运动效果。

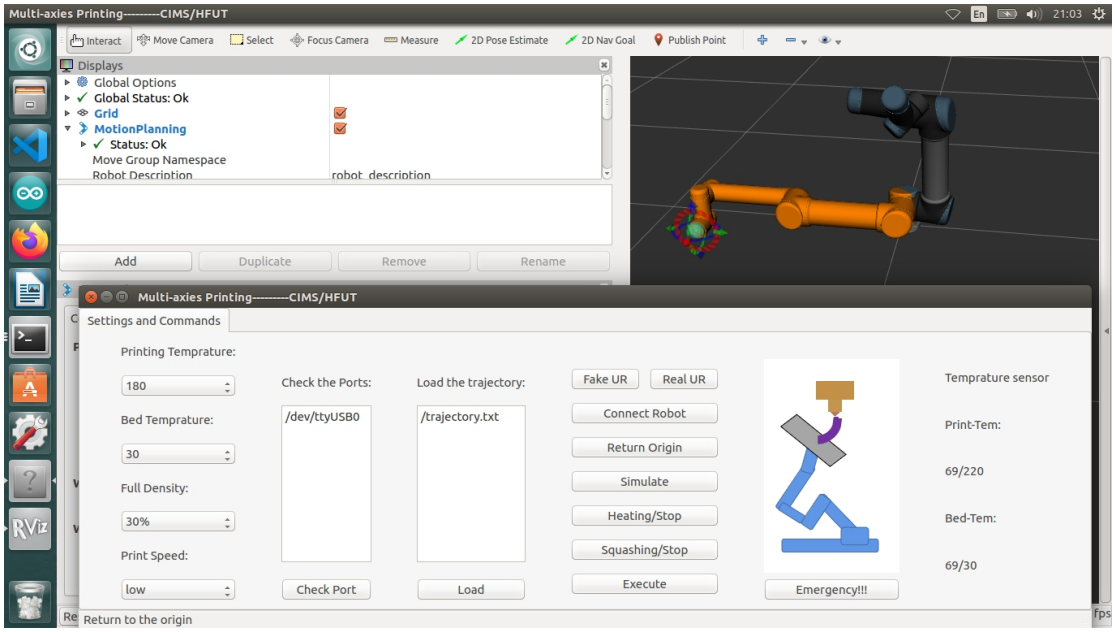

图 17 机器人回零操作

仿真完成之后，就可以按下“Heating/Stop”（加热/停止）对 3D 打印喷头和热床组件进行加热，两部件的温度数据都会在主窗口 1 的第六列实时显示。“Squashing/Stop”（挤出/停止）是对喷头单独调试时的功能键，可以让 3D 打印机的喷头进给物料。温度到达指定范围之后，就可以按下“execute”（执行）按钮，开始正式打印过程。在此过程，机器人末端热床的运动轨迹会实时可视化的显示出来。

当系统在正常工作时，如果想要监测机器人和热床的运动参数，新打开一个 shell 终端，输入“python speed_monitor.py”，打开主窗口 2，也就是运动参数监测窗口，该窗口有两个分页，第一个分页叫做“Robot Arm”（机械臂），用于监测 ur5 机器人六个关节的角位置和角速度（如图 18），第二个分页叫做“Multi-axies Platfrom”（六自由度移动平台），用于检测该平台运动时热床的线速度和角速度（如图 19）。在任一分页按下“Monitor”（监测）按钮，就会开始读数，按下“Graph”（图像）按钮，就会跳出图像窗口，实时反映运动量的变化（如图 20）。

图 18 监测窗口分页 1

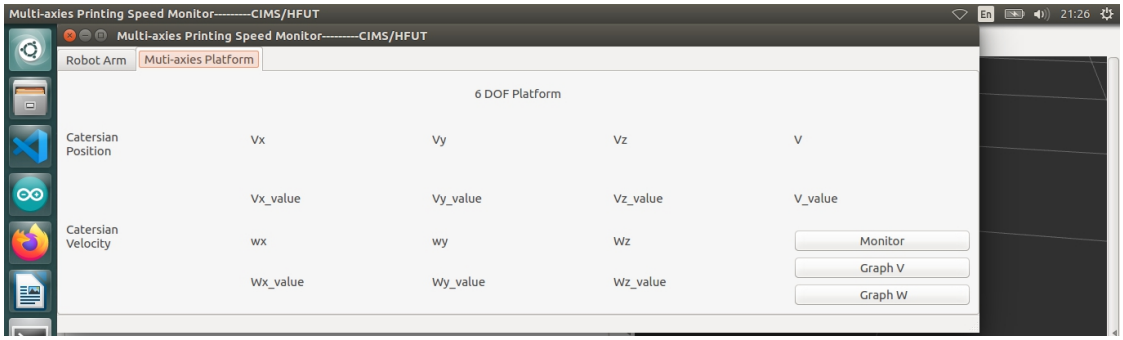

图 19 监测窗口分页 2

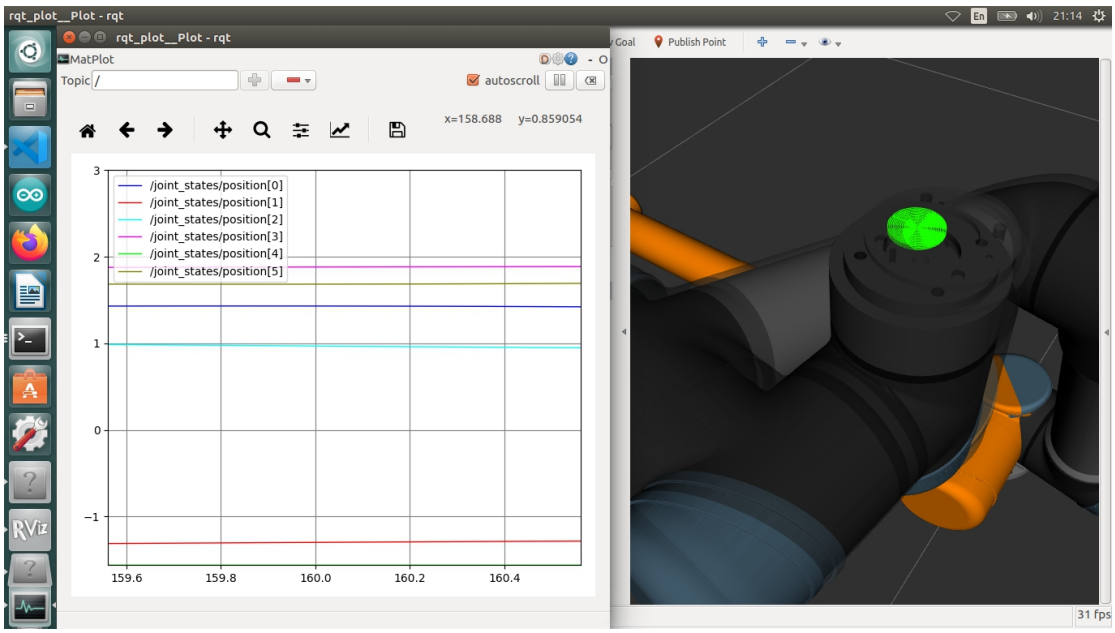

图 20 监测窗口图像