对开源机械臂

Niryo One 的国产化研究和制作

指导教师：王文琛

邝弘亮

摘要

本论文主要包括了本课题研究的出发点，即学习掌握六轴桌面机械臂Niryo One的结构，通讯协议以及ROS系统，在此基础上，通过寻找能替代的国产零部件，以及通过开源源代码推算没公开的部分，完成了步进电机控制板的移植工作；并运用3D打印技术生产所需零部件，最终完成搭建机械臂的工作。

关键字：机械臂，CAN BUS，3D 打印，ROS

目录

1 引言 …………………………………………………………………………………… 1

2 正文

2.1 准备树莓派

2.1.1 安装Raspbian操作系统

2.1.2 安装ROS Kinetic系统 ………………………………………………………… 2

2.1.3 安装开发环境 …………………………………………………………………… 3

2.2 步进电机控制板的国产化替代板 ………………………………………………… 4

2.2.1 磁传感器移植 …………………………………………………………………… 5

2.2.2 电机驱动移植 …………………………………………………………………… 7

2.2.3 CAN bus移植

2.3 制作树莓派扩展板 ………………………………………………………………… 11

2.4 3D打印机械臂组件………………………………………………………………… 12

2.4.1桌面级打印机的流程 …………………………………………………………… 13

2.4.2桌面3d打印实践和遇到的问题 ……………………………………………… 14

2.4.3对打印成品的后续加工 ………………………………………………………… 15

2.4.4成品的保存方式

2.5 Niryo One 组装

2.6 ROS 操作系统 …………………………………………………………………… 17

3.结论与展望 ………………………………………………………………………… 19

4.参考文献 …………………………………………………………………………… 19

1 引言

工业机器人可以说是现代工业的基石。凭借着与多样化传感器的协同及多轴的运动，它可以实现复杂而又精准的各种操作。然而在桌面机械臂的领域，目前的大量产品都是面向于教育领域的，在操作的精度上有所欠缺，同时部分为节约成本而做出的结构设计也存在缺陷。针对这一现状，我们提出了这一课题。我们意图采用法国Niryo公司研发的开源机械臂Niryo One 的硬件设计，通过自主创新，针对国内能买到的零部件进行改造，修正它的控制软件以适应国产化需求。使其能作为教育人士、创客、小公司的机械臂操作平台。

2.正文

本次课题研究，实施流程分成多个阶段。

2.1 准备树莓派

2.1.1 安装Raspbian操作系统

Niryo One 提供了一份针对Raspberry Pi 3的SD卡镜像文件供下载。但由于我们需要改动源代码，因此需要搭建完整的树莓派开发环境：

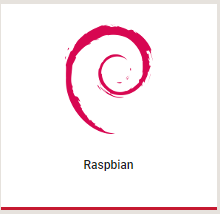
通过对Niryo One Raspberry Pi 3 image v2.2.0恢复到树莓派并研究，决定我们的树莓派采用最适合的官方Raspbian操作系统。(图2.1)

图 2.1 Raspbian 图标

镜像文件从<https://downloads.raspberrypi.org/raspbian_full_latest>下载。

用Win32 Disk Imager烧录Raspbian镜像文件到TF卡上。

烧录完成后，树莓派正常开机然后需要做以下几个特殊配置：

1. Wifi 设置

设置wifi成热点，以便外部计算机连接并控制。

1. 系统设置

打开 raspi-config

选择 A1 Expand filesystem 分区扩展到整张TF卡。

1. 创建Niryo用户

命令行创建新用户 niryo, 密码 robotics，并赋予权限。

2.1.2 安装ROS Kinetic系统

ROS(Robot Operating System)是面向机器人的开源的元操作系统(meta-operating system)。它能够提供类似传统操作系统的诸多功能,如硬件抽象、底层设备控制、常用功能实现、进程间消息传递和程序包管理等。此外,它还提供相关工具和库,用于获取、编译、编辑代码以及在多个计算机之间运行程序完成分布式计算。

先期准备工作：

$ sudo sh -c 'echo "deb http://packages.ros.org/ros/ubuntu $(lsb\_release -sc) main" > /etc/apt/sources.list.d/ros-latest.list'

$ sudo apt-key adv --keyserver hkp://ha.pool.sks-keyservers.net:80 --recv-key C1CF6E31E6BADE8868B172B4F42ED6FBAB17C654

更新Debian软件包：

$ sudo apt-get update

$ sudo apt-get upgrade

安装Bootstrap依赖关系包：

$ sudo apt-get install -y python-rosdep python-rosinstall-generator python-wstool python-rosinstall build-essential cmake

安装rosdep：

$ sudo rosdep init

$ rosdep update

创建catkin工作空间：

$ mkdir -p ~/ros\_catkin\_ws

$ cd ~/ros\_catkin\_ws

核心包：

$ rosinstall\_generator ros\_comm --rosdistro kinetic --deps --wet-only --tar > kinetic-ros\_comm-wet.rosinstall

$ wstool init src kinetic-ros\_comm-wet.rosinstall

安装ROS额外包：

sudo apt-get install ros-kinetic-robot-state-publisher ros-kinetic-moveit ros-kinetic-rosbridge-suite ros-kinetic-joy ros-kinetic-ros-control ros-kinetic-ros-controllers ros-kinetic-tf2-web-republisher

2.1.3 安装开发环境

安装额外的Python模块：

sudo -H pip install jsonpickle

进入catkin工作空间并下载Niryo One ROS源码：

cd ~/catkin\_ws/src

git clone https://github.com/NiryoRobotics/niryo\_one\_ros.git .

编译源代码包：

cd ~/catkin\_ws

catkin\_make

点命令生成执行点

source /opt/ros/kinetic/setup.bash

source ~/catkin\_ws/devel/setup.bash

至此树莓派主控制器安装完毕

2.2 步进电机控制板的国产化替代板

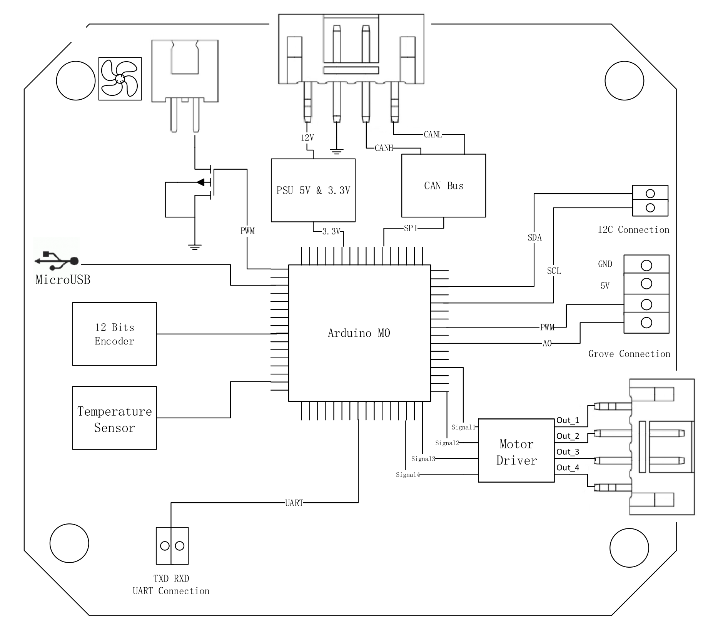
虽说Niryo One是个开源项目但它所用的步进电机控制板只有简单的原理图。(图2.2)

图 2.2 Niryo One 步进电机控制器框架图

因此我们找到具有相同功能，主控片同样也是SAMD21G的开源步进电机控制板Misfittech的Nano Zero(图2.3)来替代它。它们之间的区别如下：(表2.1)

|  |  |  |
| --- | --- | --- |
|  | Niryo Stepper | Nano Zero Stepper |
| 主控片 | SAMD21G | SAMD21G |
| 电机驱动 | A4954 | A4954 |
| 磁传感器 | AS5600 | AS5047D |
| 通讯 | CAN bus | 串口 |

表2.1 Niryo Stepper 和Nano Zero Steppre 硬件比较表

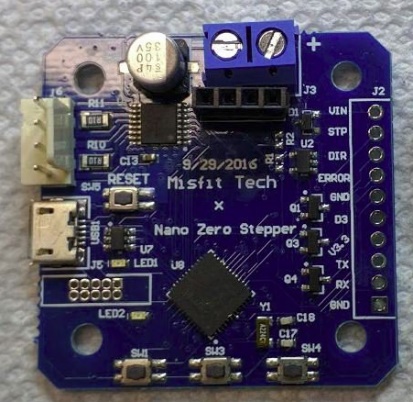
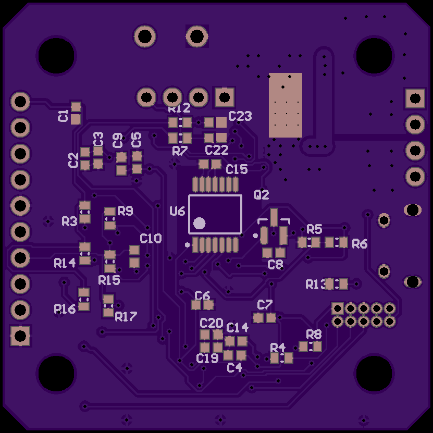
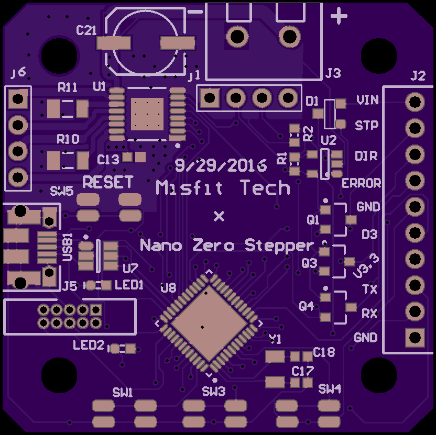


图2.3 Misfittech 的 Nano Zero 电机控制板以及实物图

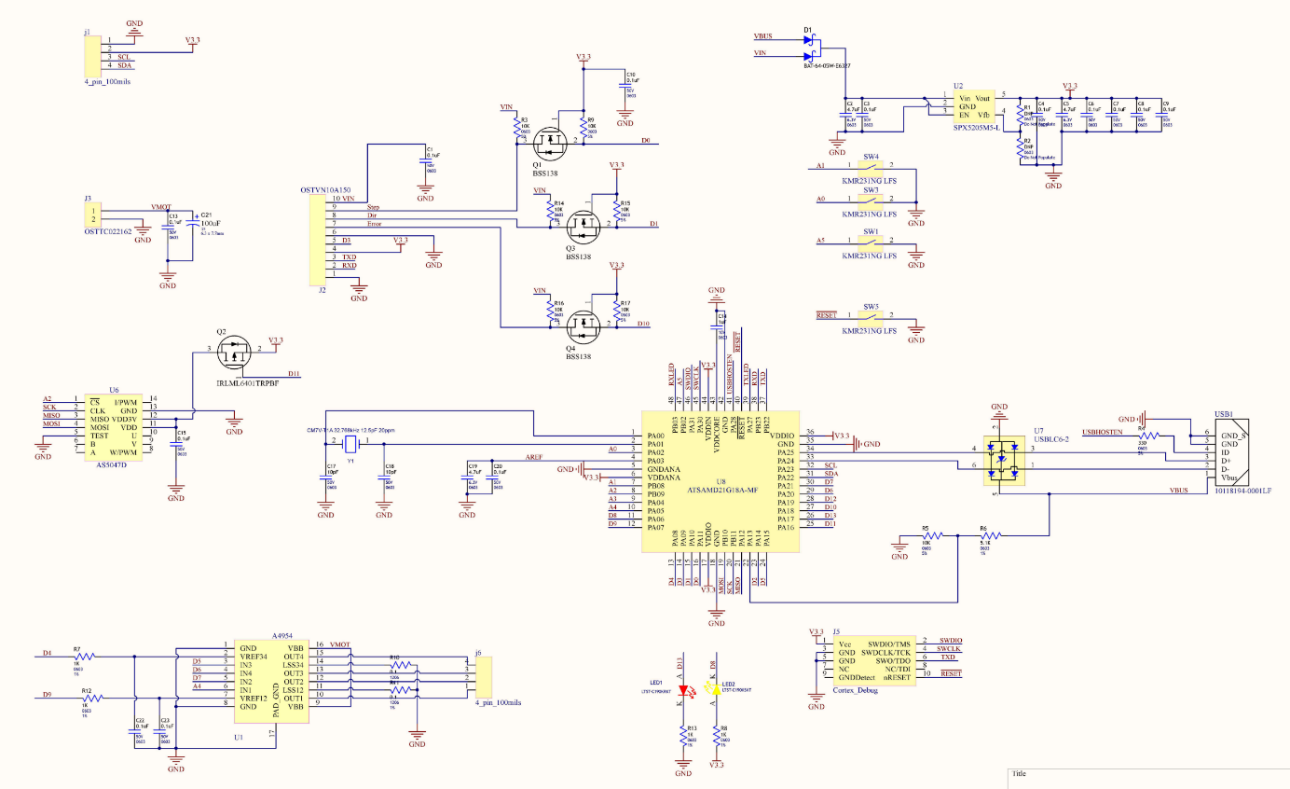
2.2.1 磁传感器移植

图2.3 Nano Zero 电机控制板电路图

由于niryo stepper中原生旋转位置传感器AS5600与SAM D21G微控器之间采用的是I2C协议进行的通信，而nano stepper中的AS5047旋转位置传感器采用spi协议通讯，且AS5600为12位的输出分辨率，AS5047为14位的输出分辨率。

首先，将AS5047D中的spi初始化代码SPISettings settingsA(5000000, MSBFIRST, SPI\_MODE1);移植到niryo stepper AS5600代码中，

其中函数void init\_position\_sensor();函数int read\_encoder();

函数void update\_current\_position(int microsteps)

在niryo stepper的主程序中被调用，于是将AS5600中通过I2C获取位置代码Wire.requestFrom(AS5600\_ADDRESS, 2);改为AS5047d中通过SPI获取位置的代码(uint32\_t)readAddress(AS5047D\_CMD\_ANGLEUNC)。同理，实现通过AS5047D获取位置信息被niryo stepper主代码调用。

对于AS5600与AS5047输出分辨率的差别，在整合的代码中对AS5047获取的介于0x0000与0xFFFF的数据右移3位，将14位数据降至12位。通过于niryo one studio控制下机械臂的角度核对并矫正14位值与角度值之间转换率。

2.2.2 电机驱动移植

虽然niryo stepper与nano stepper步进电机闭环控制器中都使用了A4954步进电机驱动模块，但是对照nano stepper的电路原理图与niryo stepper的代码后发现A4954步进电机驱动模块与SAM D21G微控制器的引脚并不相同，于是对代码做了以下改动(表2.2)

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Niryo stepper | | | Nano Zero stepper | | |
| IN\_4 | (8) | PA06 | PIN\_A4954\_IN3 | (5) | PA15 |
| IN\_3 | (3) | PA09 | PIN\_A4954\_IN4 | (6) | PA20 |
| IN\_2 | (17) | PA04 | PIN\_A4954\_IN2 | (7) | PA21 |
| IN\_1 | (15) | PB08 | PIN\_A4954\_IN1 | (18) | PA05 |
| VRCF\_2 | (4) | PA08 | PIN\_A4954\_VRCF34 | (4) | PA08 |
| VRCF\_3 | (9) | PA07 | PIN\_A4954\_VRCF12 | (9) | PA07 |

表2.2 Niryo stepper 和 Nano Zero stepper A5954引脚区别

2.2.3 CAN bus移植

对于电机控制板和树莓派主机之间通讯，Niryo One是用CAN bus通讯，针对替代板Nano Zero电机控制板，我们曾尝试用单线Modbus协议建立通讯。但考虑到不能像CAN bus那样很好地处理中断，因此还是改回使用CAN bus协议，转换芯片用MC2515。

由于Nano Stepper中微处理器原有的SPI口被AS5047D所占用，并且SAMD21G引脚与AS5047D之间的电路无法引出跳线，所以缺乏连接CAN bus转换片MCP2515的spi引线。通过查阅SAMD21G有关资料，发现能开放第2个SPI口。结合Nano Zero步进电路板的特殊布局，通过几次探索性飞线，最后确定唯一能用的引出口是Nano Stepper中SDA 、SCL、 step、 dir、 enable、 d3口。

图2.4 SAMD21G 引脚关系图

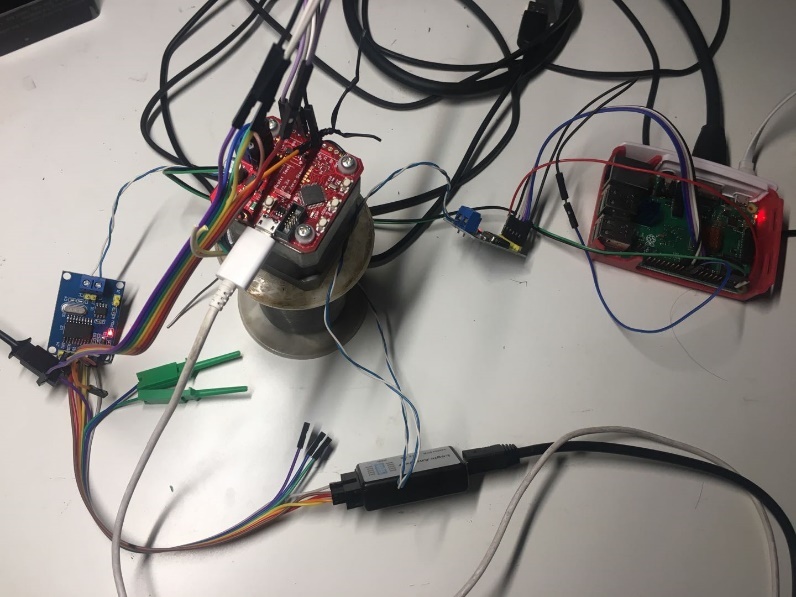
于是在mcp\_can的代码中添加此新spi口SPIClass mySPI (&sercom3, 20, 21, 10, SPI\_PAD\_0\_SCK\_1, SERCOM\_RX\_PAD\_2)。

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | Pin | Arduino ‘Pin’ | SERCOM | SERCOM alt | Pin |
| 1 | PA22 | D20 / SDA | SERCOM3.0 | SERCOM5.0 | SPI\_MISO |
| 2 | PA23 | D21 / SCL | SERCOM3.1 | SERCOM5.1 | SPI\_SCK |
| 3 | PA18 | D10 | SERCOM1.2 | SERCOM3.2 | SPI\_MOSI |

表2.3 第2个SPI接口引脚关系表

在Nano Stepper步进电机闭环控制器上找到所有对应的引脚接口，通过跳线与MCP2515 CAN bus转换芯片连接，使MCP2515的SPI引脚与在Nano Stepper上SAMD21G模拟出的SPI接口一一对应。

尝试通过连接的MCP2515转换芯片与装有MCP2515的树莓派主机板通信，并通过多通道数字逻辑分析仪监视spi通道中的信号流，检查所有焊线与接线，更正接线。(图2.5)正在联调的步进控制器和树莓派。

 图2.5 步进控制与树莓派通讯调试

而对比使用的MCP2515模块与niryo的mcp2515相关代码后发现，niryo原厂MCP2515使用的是16mhz的晶振，而国内MCP2515使用的是8mhz的晶振，所以需要在程序中将16mhz改为8mhz（图2.6）。

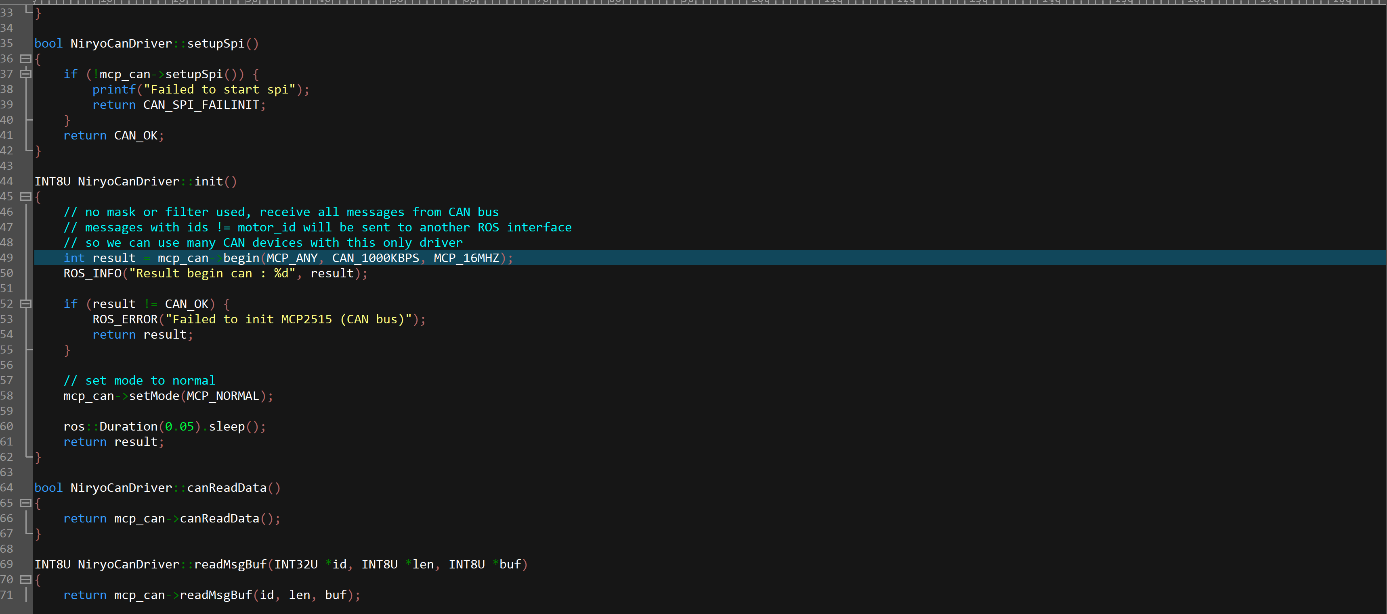


图2.6 MCP频率更改

除此之外，国产的MCP2515模块将MCP2515与TJA1050的供电同时使用了5v，而MCP2515的参考输入电压为3.3v，所以需要从树莓派与nano stepper上跳线引出3.3v。

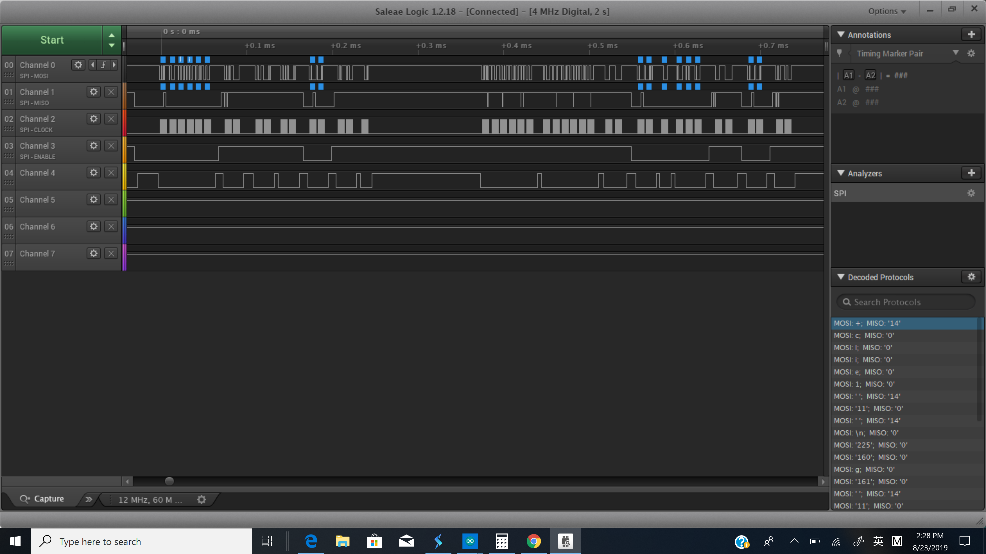
联调中，发现通讯时好时坏，于是通过通过逻辑分析仪对CAN总线进行采集分析，发现CAN bus上的数据有时会出现干扰，而在使用2色双绞线后通讯相对稳定。

图 2.7 MCP2515的SPI接口逻辑分析图

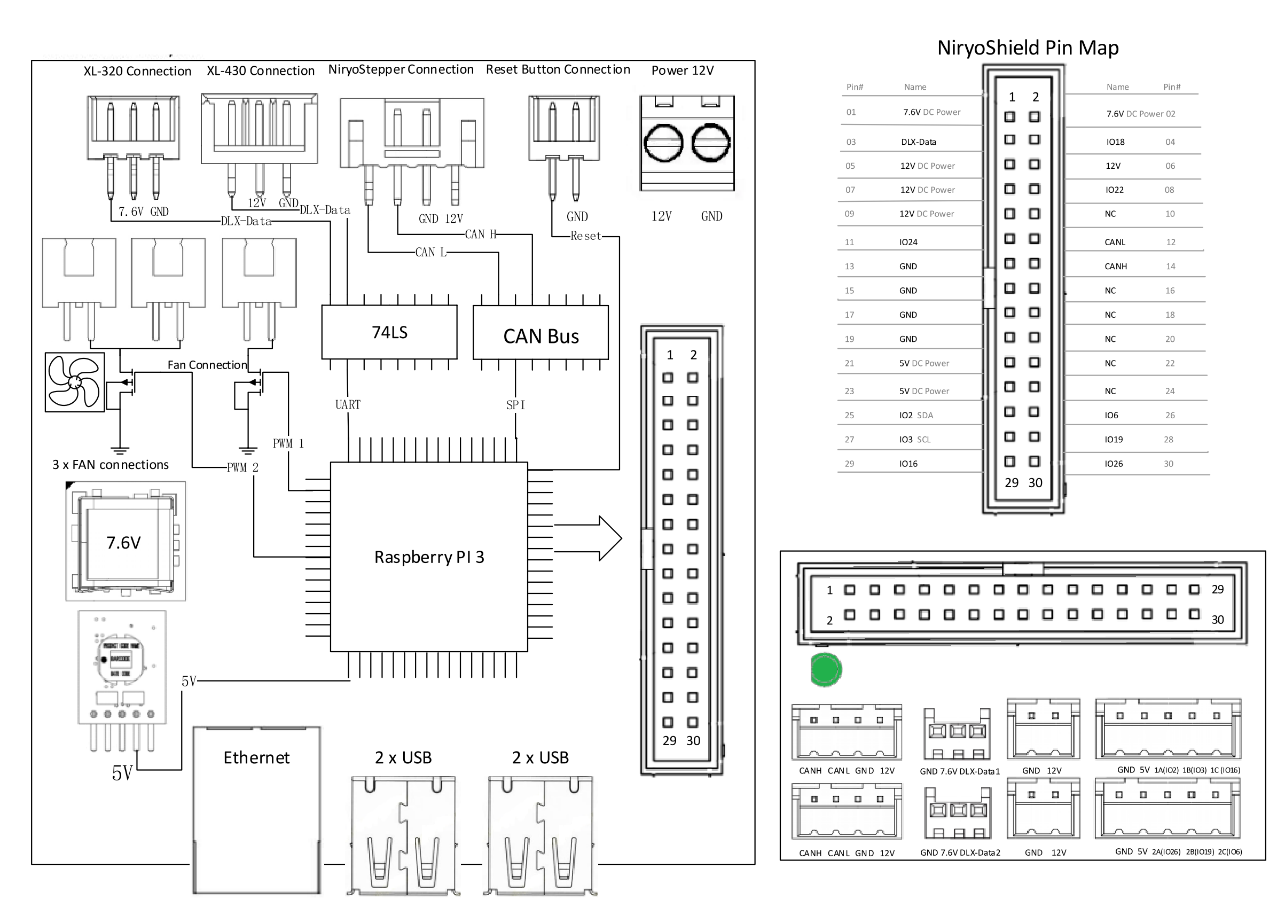
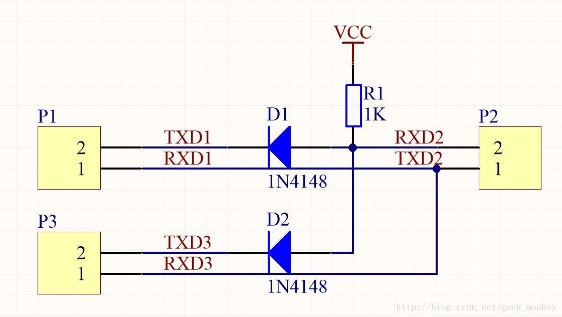
2.3 制作树莓派扩展板

图2.8 Niryo One 树莓派扩展板

Niryo One树莓派扩展板(图2.8)主要完成和CAN bus通讯，和DYNAMIXEL舵机通讯，以及电源分配。通过对niryo\_one\_ros-master代码的解读，和CAN bus通讯采用MCP2515转换，和DYNAMIXEL舵机通讯是1-wire专用协议，用树莓派的硬串口。

看似简单，但制作过程遇到不少挫折：

1. 用了国产的Dynamixel转接板，但不成功，原因不明。
2. 通过分析Niryo One用的是74LS241三态缓冲，应用上后不能通讯。通过逻辑分析仪分析，串口TX发出的包舵机能正常接受，但舵机发回的包串口RX不能正常接受。请教了电子工程师，建议把TTL的74LS241改为CMOS的74HC241，问题得到解决，由此看出DYNAMIXEL信号输出驱动力弱。
3. 由于DYNAMIXEL舵机用了2种XL320和XL430，两者的电压不同，一个是7.2v另一个是12v，因此需要分开，串口需要驱动2组转接，简单的并联会引发不稳定。需要用二极管做单向导通。(图2.9)

图 2.9 串口并联原理图

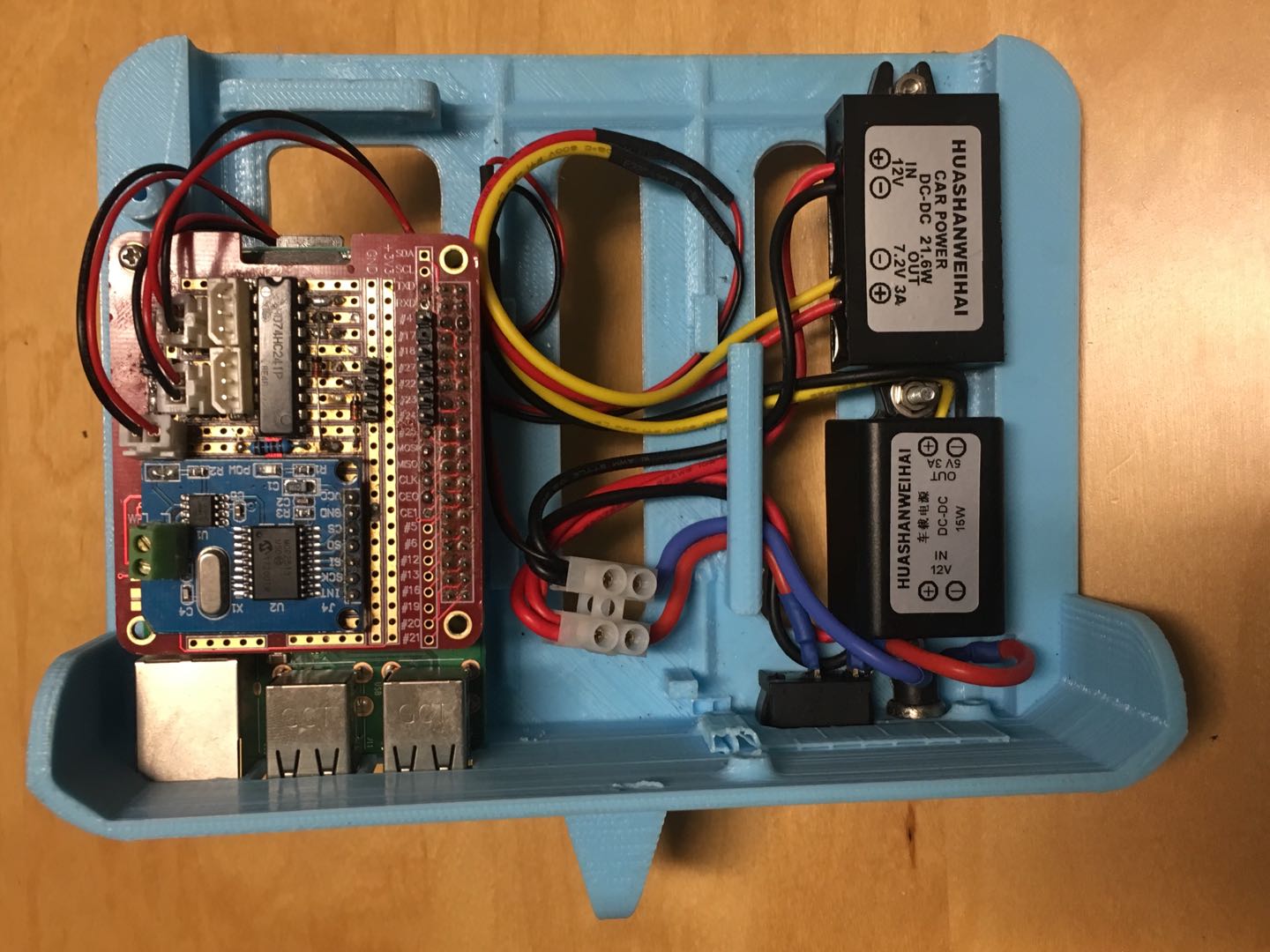
 通过努力，完成了所有的协议通讯，通过了测试，从中我们也学到了很多硬件和通讯协议的知识。(图2.10)是组装完成后的扩展板、电源等底板部件

图2.10 Niryo One 机械臂主控板

2.4 3D打印机械臂组件

零件生产问题是此阶段研究围绕的主要问题。在学校中有桌面级的FDM原理3D打印机，我与课题组成员选择使用3d打印技术将机械臂的部件打印下来。而在3D打印中，我们选择了桌面级3D打印机，原因主要如下：

（1）精度问题: 我们课题组采用机械臂的结构设计来源于Niryo-One，属于一款入门程度的机械臂，对于零件的精度要求不是过于苛刻，市售精度较好的3D打印机的精度可以满足。

（2）速度问题：需要3D打印总量较小，对速度要求较小，桌面级3D打印即可完成任务。

（3）材料问题：机械部件的主要构成部分（速干型稳定熔点塑料）相同。

（4）成本问题：价格低廉，在可考虑范围内，符合课题力求在控制成本的情况下达到最优精度的目标。

（5）技术问题：技术较普及，且本课题指导教师有开设相关课程。

2.4.1桌面级打印机的流程

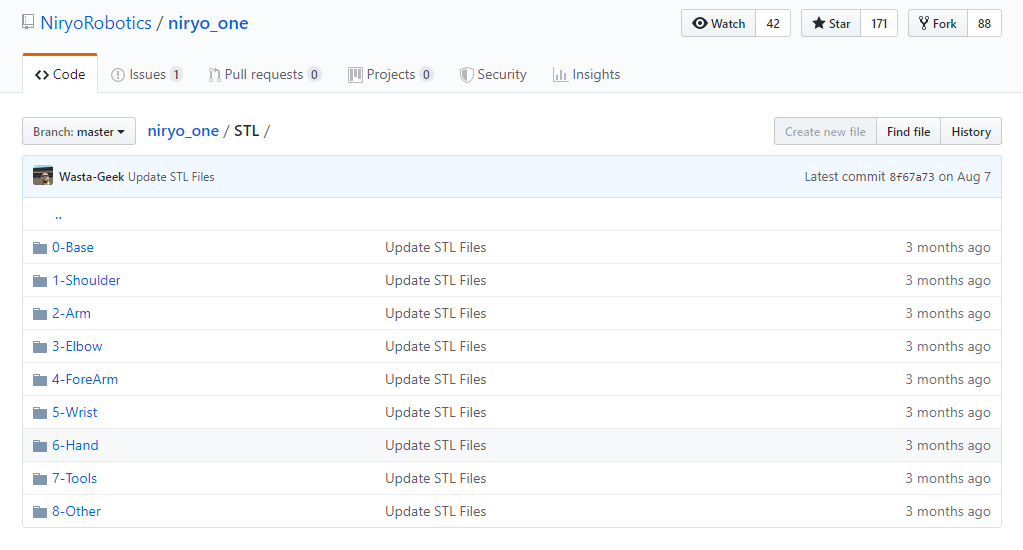
（1）·建模：在本课题中，我们直接在GITHUB上下载Niryo的结构件三维图纸

图 2.11 github 上的niryo one 3D 打印文件

（2）·格式转化：一般的建模软件的软件类型都是STL文件，这种格式是不被3d打印机所接受的，桌面级3d打印机一般只接受Gcode格式。本次我们使用CURA进行切片与格式转化

（3）·打印设置：通过SD卡或者USB优盘把它拷贝到3D打印机中，进行预热（对应所需材料的打印温度和所需底板温度），开始打印

（4）·等待打印结束。

2.4.2桌面3d打印实践和遇到的问题：

在我们的打印过程中，我们也发现了一定的问题，主要有大致以下几方面的问题：

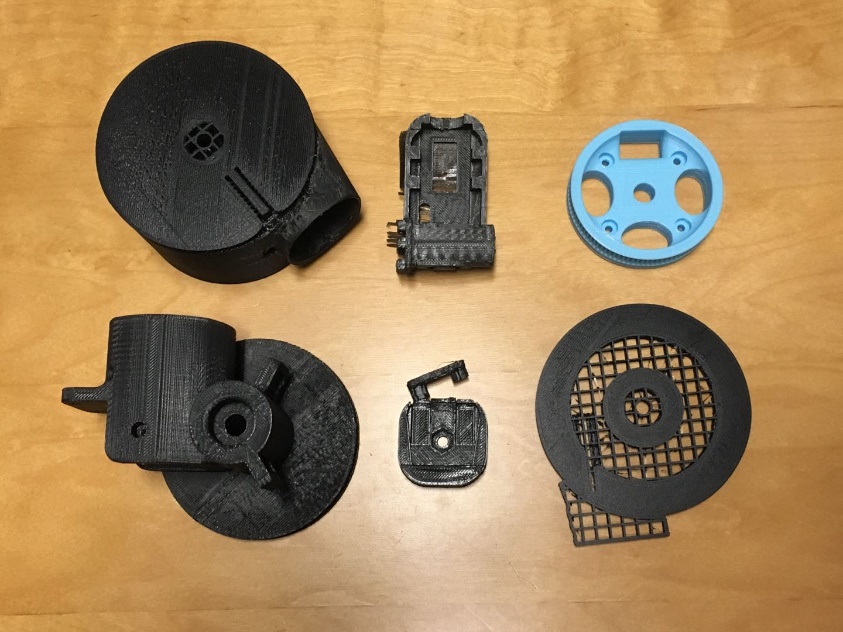
（1）打印部件太高无法打印：这是因为打印机型号难以支撑过高的部件造成的，过高的结构部件对于硬度和稳定性有着很大的需求，这也是目前阶段最难解决的问题，

解决方法：在三维建模软件中予以分段，分段打印后进行拼接。但这一过程中也存在缺陷：

（2）喷嘴吐丝不均匀：这种情况是喷嘴温度和塑料的融融温度不匹配的结果，一般地，温度要是低于需要温度，材料难以融化，会导致吐丝不均匀，后果就是打印的叠加出现差错和粘丝，导致最后打印失败。

解决方案：精确设置好每一个打印的需要温度（不同的机械部件所需要的材料也有差别，需要区别来逐个调节），并在打印前进行充分预热。甚至可以多次试验，寻找最佳温度，提升打印质量。

（3）喷料不粘底板：这种情况是是因为底板温度过高，或者喷嘴与底板间间距过大或过小。导致打印不成形。

解决方案：使用水平仪，调节底板水平，并在试打同时调节底板，让打印能够顺利进行（这是至今为止我与课题成员亲自解决的最大的问题之一。）

2.12 部分打印失败的零部件

2.4.3对打印成品的后续加工

这一部分没有什么困难复杂的程序和操作，但需要十分小心谨慎。在打印一些镂空的机械部件时，我们会对这种难以保持稳定成型的部件进行支撑，也就是在空心部分打入较为稀疏的支撑部分，在打印完成品以后，要将里面这些细丝剪去，在这过程中，不小心的用力过度就会破坏整个部件，所以一定要小心至极。同时，对于一些稳定性不足的机械部件，我们要用万能胶水对其进行固定，因为仪器精密的需要，在进行这一步的时候，必须两人配合，这充分考验了同学间的配合能力。

2.4.4成品的保存方式

因为打印材料的低熔点，打印完的成品的保存大致有以下几点注意事项：

（1）避免阳光直射，因为这种塑料的特殊性，过多的阳光照射会让材料部分分解，导致打印的机械部件损坏。

（2）不要存放在湿度过大的地方：部分的外层塑料会吸水膨胀（这种吸水是不明显的，但对于螺帽的地方，轻微的膨胀会导致机械部件无法拼装）。

（3）避免过高温度，防止部件融化，再度粘合导致精度下降。

（4）同时，耗材在不使用的情况下也需要尽量密封，因为氧化性会让内部结构变化，

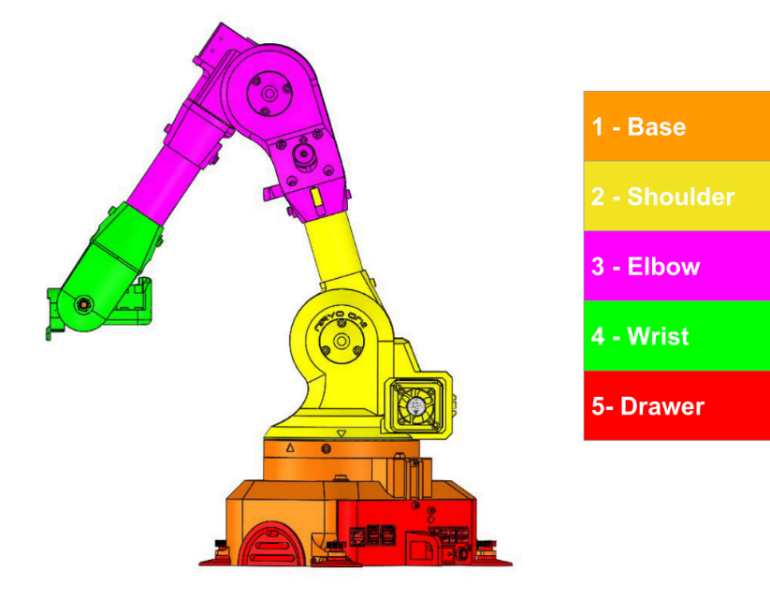
2.5 Niryo One 组装

图2.13 Niryo One 组件构成

Niryo One 设计得非常巧妙，每一步的安装都有严格的顺序。组装的过程中，还需要倒推一些没有公布的零件尺寸，比如铝合金空心圆柱的规格尺寸等。在耐心地不断尝试下，发挥了超常想象力，终于完成了全部零部件的组装工作，精度也达到设计要求。

图2.14 组装好的机械臂

2.6 ROS 操作系统

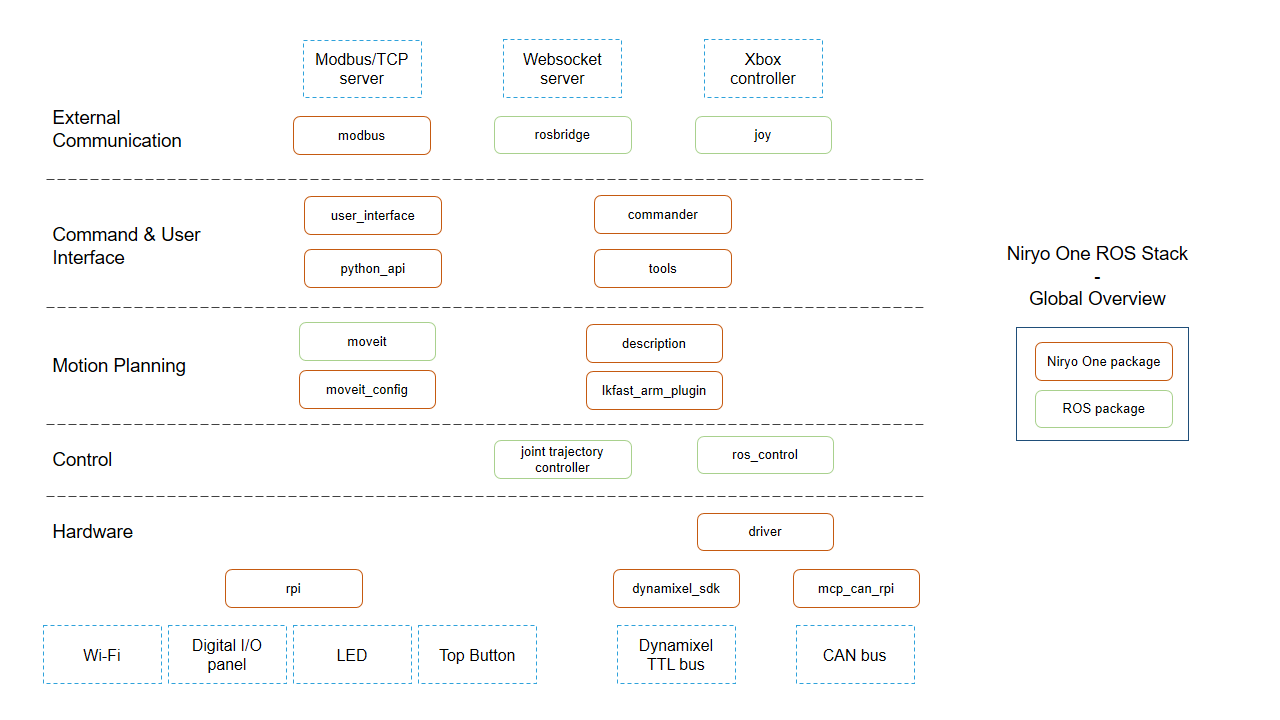
 图2.15 Niryo One 机械臂ROS操作系统框架

图2.14 是Niryo One 机械臂ROS操作系统框架，它并非是一种操作系统，它只是一种通信框架，一种基于消息传递通信的分布式多进程框架。ROS的主要组成包括ROS Mastar 、 ROS Node、ROS Service等。不同的功能可以由不同的节点实现，节点之间可以通过发布和订阅话题来传递消息，它的本质是基于TCP/IP的Socket通信机制。不同的模块可以被单独设计，在运行时松散耦合，它执行若干种类型的通信，比如基于服务的同步RPC通信、基于话题的异步数据流通信以及基于参数服务器的数据存储等。

它的主要组成部分为：

（1）节点

节点就是一些直行运算任务的进程。ROS利用规模可增长的方式是代码模块化：一个系统就是典型的由很多节点组成的。在这里，节点也可以被称之为“软件模块”。我们使用“节点”使得基于ROS的系统在运行的时候更加形象化：当许多节点同时运行时，可以很方便的将端对端的通讯绘制成一个图表，在这个图表中，进程就是图中的节点，而端对端的连接关系就是其中弧线连接。

（2）消息

节点之间是通过传送消息进行通讯的。每一个消息都是一个严格的数据结构。原来标准的数据类型（整型，浮点型，布尔型等等）都是支持的，同时也支持原始数组类型。消息可以包含任意的嵌套结构和数组（很类似于C语言的结构structs）。

（3）主题

消息以一种发布/订阅的方式传递。一个节点可以在一个给定的主题中发布消息。一个节点针对某个主题关注与订阅特定类型的数据。可能同时有多个节点发布或者订阅同一个主题的消息。总体上，发布者和订阅者不了解彼此的存在。

（4）服务

虽然基于话题的发布/订阅模型是很灵活的通讯模式，但是它广播式的路径规划对于可以简化节点设计的同步传输模式并不适合。在ROS中，我们称之为一个服务，用一个字符串和一对严格规范的消息定义：一个用于请求，一个用于回应。这类似于web服务器，web服务器是由URIs定义的，同时带有完整定义类型的请求和回复文档。需要注意的是，不像话题，只有一个节点可以以任意独有的名字广播一个服务：只有一个服务可以称之为“分类象征”，比如说，任意一个给出的URI地址只能有一个web服务器。

在上面概念的基础上，需要有一个控制器可以使所有节点有条不紊的执行，这就是一个ROS的控制器（ROS Master）。

ROS Master 通过RPC（Remote Procedure Call Protocol，远程过程调用）提供了登记列表和对其他计算图表的查找。没有控制器，节点将无法找到其他节点，交换消息或调用服务。

ROS 是个复杂的大系统，短暂的学习并不能很好地理解各含义。好在我们有了这个Niryo One机械臂平台，为我们将来深入研究打下基础。

3.结论与展望

在为期一学年的课题研究中，我们小组针对目前功能较完善的开源机械臂系统Niryo One的控制系统，进行了对于获取较困难的控制板的更换，同时移植了属于工业现场总线的CANBUS总线，通过双向控制保证了机械臂上六个步进电机移动的精准性，同时也使得机械臂具有了与工业机器人一致的可操作性。下一步，我们希望能够在多台同种机械臂上实现多台协同控制，并进一步丰富它的功能，使其同时具有更好的教学性与实用性。

4.参考文献

[1]Niryo Robotics. Niryo-one[website]. <https://niryo.com/>, Quoted on 11/14/2019

[2]renarded,SarraELGHALI,Wasta-Geek,smaassen,avkudr,alfonsosanchezbeato. Niryo One ROS stack[website]. <https://github.com/NiryoRobotics/niryo_one_ros>, Last commit on 09/25/2019

[3]The Raspberry Pi Foundation. Download Raspbian for Raspberry Pi[website]. <https://www.raspberrypi.org/downloads/raspbian/>, Last update on 09/26/2019

[4]balenaEtcher. balenaEtcher-Home[website]. <https://www.balena.io/etcher/>, Quoted on 11/14/2019

[5]TullyFoote. ROS Kinetic Kame[website]. <http://wiki.ros.org/kinetic>, Update on 01/08/2018

[6]drogon. Raspberry Pi | Writing | Download & Install | Wiring Pi[website]. <http://wiringpi.com/download-and-install/>, Update on August 2019

[7]DrTrigon. Arduino 1-wire Sniffer: 4 steps[website]. <https://www.instructables.com/id/Arduino-1-wire-sniffer/>, Quoted on 11/14/2019

[8]giggls. 4 different Methods of 1-wire access on Raspberry Pi[website]. <http://blog.gegg.us/2013/03/4-different-methods-of-1-wire-access-on-raspberry-pi/>, Released on 03/29/2013

[9]thiagohersan. How to Drive Dynamixel AX-12A Servos (with a RaspberryPi)[website]. <https://www.instructables.com/id/How-to-drive-Dynamixel-AX-12A-servos-with-a-Raspbe/>, Quoted on 11/14/2019

[10]Industrial Robotics. Beta Testing Dynamixel XL-320 Servos | Robotics Tomorrow[website]. <https://www.roboticstomorrow.com/article/2014/03/beta-testing-dynamixel-xl-320-servos/248>, Released on 03/27/2014

[11]lady ada. Overview | Using ATSAMD21 SERCOM for more SPI, I2C and Serial ports[website]. <https://learn.adafruit.com/using-atsamd21-sercom-to-add-more-spi-i2c-serial-ports/overview>, Quoted on 11/14/2019