Table of Contents

介绍

开发前的准备

开发与使用

4 其他

myCobot小象机械臂 - 《从入门到精通》

为什么设计myCobot

> 人人都可以学习玩耍的入门级协作机器人

myCobot的设计初衷是为了让对**六自由度串联机械臂**感兴趣的朋友，可以从**0**到**1**的了解、学习和操作机械臂，创造前所未有的机械臂使用体验与教学价值。

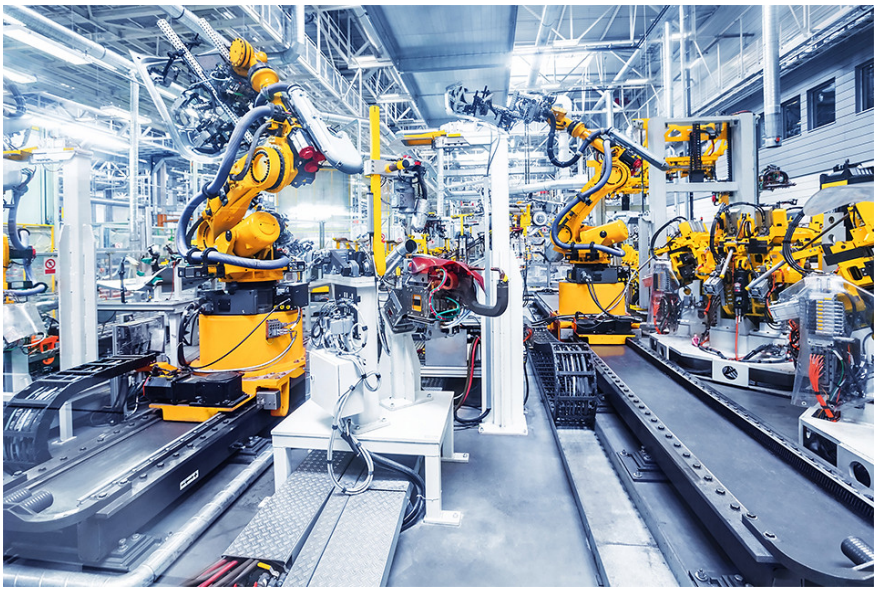
你能学到什么

> 机器人学的基础是刚体运动学、动力学。但同时融合了硬件、软件、算法和控制的一门交叉学科。

通过myCobot，大家可以学习到

- 硬件
  - 基于**ESP32**的嵌入式单片机 -- 什么是单片机，单片机基本原理，esp32单片机。
  - 电机与舵机 -- 什么是电机，电机的与舵机的关系，舵机的基本原理。
  - **M5Stack Basic/ Atom**
- 软件
  - **Arduino**开发环境
  - **C++**
  - **Python**
  - **ROS**平台，**MoveIt**仿真
  - 通信数据的设计
  - 虚拟机与**Linux**（视觉系统）

- 算法
  - 串联机械臂背景知识
  - 坐标与坐标转换
  - **DH**参数与使用
  - 刚体运动学**kinematics**
  - 其他可能的机械臂算法（如动力学）
- 机器视觉（视觉套装）
  - 颜色识别
  - 图像识别
  - 手眼标定
  - 联动抓取
- 其他
  - 机器人常用执行器：夹爪、吸泵
  - 从手工到机器人
  - 机器人套装与工业**4.0**应用



这本书的组成部分

> 查看左侧目录进行跳转

这本书包括了四大部分。他们分别是

- 产品简介与快速开始
  - 产品简介 -- 介绍什么是myCobot，myCobot的主要特点等内容。
  - 如何阅读 -- 根据您的学习程度和知识背景，可以选择何时的阅读与使用路径。
  - 使用用例 -- 您可以清晰的看到，如果使用了myCobot，可以完成怎样的用例。
  - 快速开始 -- 开始开箱您的myCobot，第一次的开机和使用。
- 开发前的准备
  - 背景知识 -- 学习关于工具、工业机器人，算法、软件、硬件等一些列知识。
  - 了解硬件 -- 学习关于 嵌入式硬件与 结构件、电子件等的相关知识。

    - **选择使用目的** -- 确认你想要达到的使用目的，完成固定的运动轨迹与方法。
- 开发与使用
    - **开发环境** -- 在不同的开发环境下如Arduino,ROS,uiFlow,roboFlow,python等开发myCobot。
    - **配件使用** -- myCobot搭配不同的夹爪、底座、吸泵等一起使用。
    - **机器视觉开发** -- 由机器视觉引导机械臂进行运动。
    - **机械臂简配** -- 如果您想把myCobot改变成一个4轴或者5轴的机械臂，您该如何做。
- **myCobot套件**
    - **智能货仓** -- 通过学习以上内容，使用机械臂进行不同物体的搬运。
    - **人工智能套装** -- 通过机器视觉对机器人进行引导，并进行智能抓取。
    - **工业4.0套装** -- 模拟生产线进行智能抓取与放置。

其他信息源

- 官网 www.elephantrobotics.com
- 视频 bilibili / youtube
- 购买网站
    - shop.elephantrobotics.com
- 下载此文档PDF版

联系我们

如果您还有其他问题，可以通过以下方式联系我们。

- 邮箱
  我们会在1~2个工作日内回复
  support@elephantrobotics.com
- 微信 我们只对已经购买过myCobot的用户进行微信1对1服务。

myCobot简介

1 设计背景

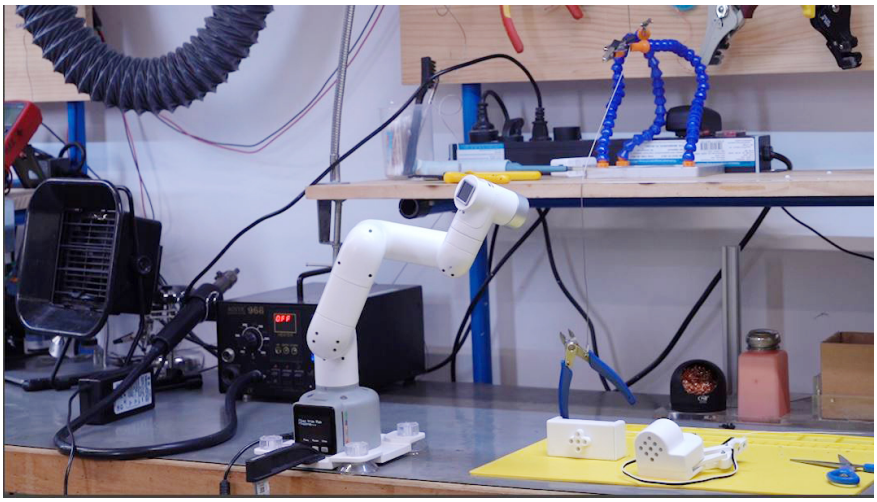
秉持**“Enjoy Robots World”**的愿景和使命，大象机器人在保留大部分工业型机器人功能的前提下设计研发了 myCobot - 全球最轻最小的协作机器人。小巧大方的工业设计，出色的性能动力表现和巨大的软硬件开发空间，赋予myCobot 无限的应用拓展潜能。

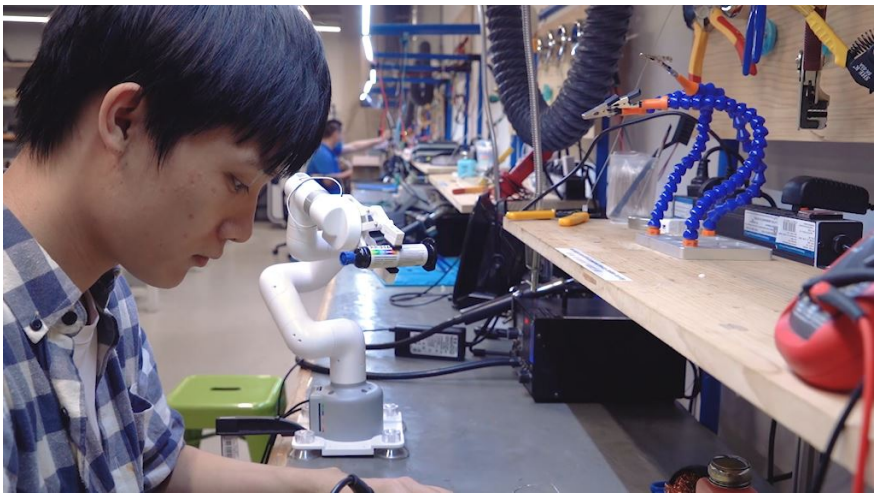
myCobot 的设计原型为大象机器人2018年推出的国内首款 all-in -one 一体化协作机器人catbot。作为国内首款一体化协作机器人，它曾斩获 2019 CAIMRS工业机器人创新奖及2019年高工机器人年度“创新技术奖“，并远销海内外30多个国家，其产品质量及智慧方案更是备受韩日美德意美等数家来自世界500强名企工厂的一致认可与好评。

2 产品简介

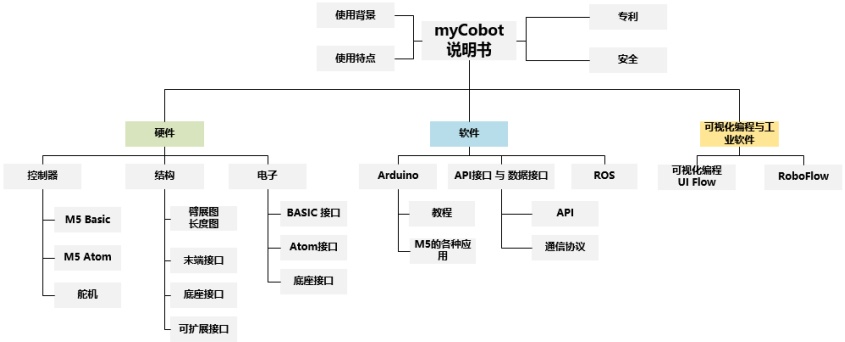
myCobot 由大象机器人和**M5stack**联合出品，是世界最小最轻的六轴协作机器人，可根据用户的需求进行二次开发，实现用户个性化定制，是生产力工具也是想象力边界的拓展工具。

myCobot 自重850g，有效载荷250g，臂长350有效280mm；体积小巧但功能强大，既可搭配多种末端执行器适配多种应用场景，也可支持多平台软件的二次开发，满足**科研教育**、**智能家居**，**商业探索**等各种场景需求。





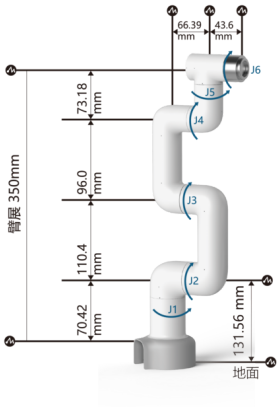
3 产品框架

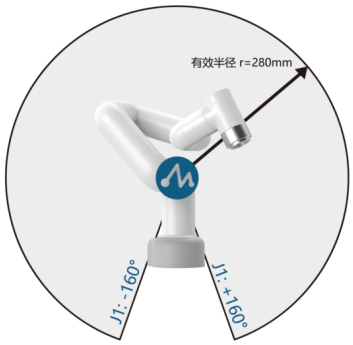


4 产品指标

表1： myCobot 280 产品参数

指标	参数
型号	myCobot-280
有效工作半径	280mm
负载	250g
臂展	350mm
重复定位精度	±0.5mm
自重	850g
电源输入	8V,5A
工作环境	-5°~45°
通信	USB Type-C





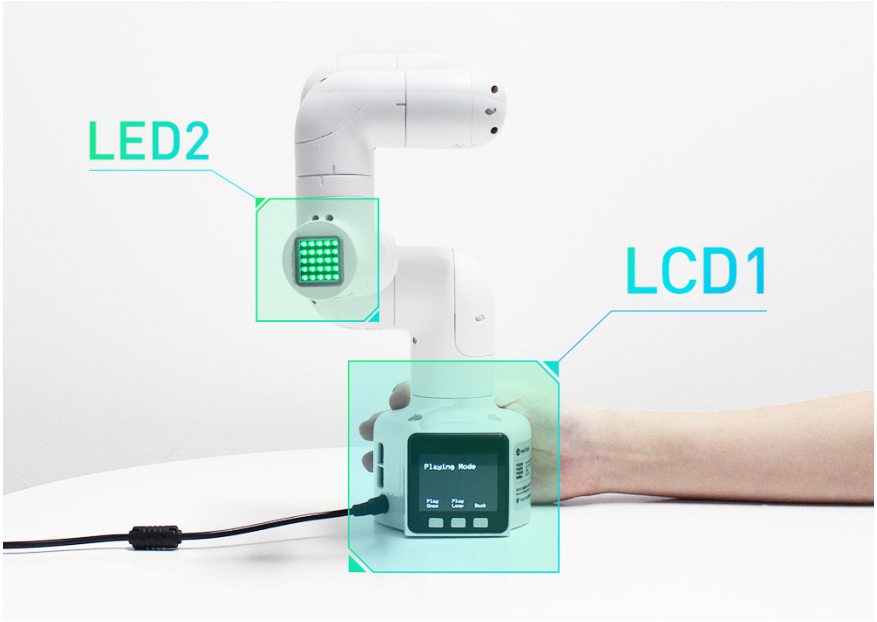
5 myCobot产品与周边

- myCobot 机械臂
- 末端执行器
  - 平行夹爪
  - 自适应夹爪
  - 张角式夹爪
  - 吸泵
- 底座
  - G型底座
  - 平面底座
- 配件
  - 摇杆
  - 电池盒

6 功能特点

- 独特工业设计，极致小巧
  一体化设计，整体机身结构紧凑，净重仅850g，十分便于携带 模块化设计，备件少、维护成本低，可快速拆卸更换，实现即插即用

- **高配置，搭载两块显示屏**
  内含6个高性能伺服电机，响应快，惯量小，转动平滑 机身携带两块显示屏，支持 fastLED库，便于拓展应用交互输出
- **乐高接头，M5数千应用生态**
  底座以 M5 stack Basic 作为主控，数千应用案例可直接使用 底座及末端带有乐高科技件接口，适用于各项微型嵌入式设备开发
- **图形化编程，支持工业机器人软件**
  采用 UIflow 可视化编程软件，掌上自如编程，操作简单易上手 兼容工业机械臂操作软件 RoboFlow，支持 Arduino + ROS 开源系统
- **轨迹录入，点位保存**
  myCobot 支持拖动示教，可记录已存入的路径 摆脱传统的路径点存模式，可保存最多60mins不同的路径



7 产品专利

> myCobot受到专利保护

序号	名称	号码
1	协作机械臂	2020030683471.3
2	机械臂连杆及其机械臂	CN 208196791 U
3	机械臂关节连接件及其机械臂	CN 208196840 U
4	一种机器人姿态保持拖动示教的方法及系统	ZL 2018 1 1634649.3
5	一种基于动量模型的机器人在线碰撞检测方法和系统	ZL 2019 1 0030748.9
6	一种不依赖关节角加速度的机器人动力学参数辨识方法	ZL 2019 1 0773865.4

证书号第4996050号

外观设计专利证书

外观设计名称：机械臂（B1）

设　计　人：宋君毅

专　利　号：ZL 2018 3 0466611.4

专利申请日：2018年08月22日

专利权人：深圳市大象机器人科技有限公司

地　　址：518055 广东省深圳市南山区桃源街道留仙大道南山云谷创新产业园二期7栋2楼208

授权公告日：2019年01月01日　　授权公告号：CN 304973675 S

国家知识产权局依照中华人民共和国专利法经过初步审查，决定授予专利权，颁发外观设计专利证书并在专利登记簿上予以登记。专利权自授权公告之日起生效。专利权期限为十年，自申请日起算。

专利证书记载专利权登记时的法律状况。专利权的转移、质押、无效、终止、恢复和专利权人的姓名或名称、国籍、地址变更等事项记载在专利登记簿上。

局长
申长雨

国家知识产权局
2019年01月01日

其他事项参见背面

如何阅读

本书阅读目标

本书的设计目标是通过阅读本书，您可以

> 主要目标

- 了解机械臂相关的机械、电子、软件的基本使用。
- 了解机械臂的基本原理，关节，坐标，术语，控制等。会进行简单的正逆运动学计算。
- 了解API控制机械臂与uiFlow可视化编程控制机械臂的基本操作。
- 具备完成**智能货仓**套装的所有技能：可以使用机械臂进行简单的路点运动以及搬运，控制等。

> 扩展目标

- 了解机器视觉相关的图像识别算法。
- 了解机器人视觉场景的搭建与视觉与机械臂的配合方法与策略。
- 具备完成**人工智能**套装的所有技能。

您的背景划分

本书的阅读需要根据您的实际背景来进行阅读，我们将背景与相关知识划分成三个不同程度：

程度	背景	具备技能	预计学习时间	建议开发平台
入门	信息，电子，机械，自动化相关专业	了解一种编程语言 了解电子相关基础知识	100小时	uiFlow
高级	了解Arduino或类似硬件产品； 了解舵机、编程； 了解IO接口等	会调试API，调试接口，了解通信	50小时	Arduino
专业	曾使用过至少1种工业或消费类机械臂； 具备硬件与软件开发能力	了解机械臂笛卡尔坐标 了解关节控制 了解机器人的基本使用	30小时	任意

学习步骤与时间

序号	目标知识点	理论	实践	预计学时
1	快速开箱	1 拖动试教	1 myCobot机器人所包括配件 2 驱动机器人进行拖动试教	1小时
2	背景-知识学习	1 工业机器人使用背景 2 坐标与空间学习，笛卡尔三维坐标与旋转，xyz 3 工业机器人关节与坐标控制	1 机器人关节控制与复现 2 机器人速度控制 3 机器人坐标点位控制与循环	5小时
3	硬件-学习	1 嵌入式电子原理与操作 2 舵机与电机原理与知识 3 执行器学习	1 Basic/atom控制与驱动 2 舵机驱动与运动 3 机器人配件学习	5小时
4	软件-固件与更新	1 识别不同的软件平台与使用目的 2 固件加载与适配原理	1 选择适合你的开发平台 2 下载与更新相应固件	2小时
5	软件-开发环境搭建	1 Arduino平台搭建 2 Arduino库文件下载与更新 3 了解串口通信	1 Arduino平台熟悉 2 加载库 3 操作运行第一行代码	2小时
6	机器人库学习与开发	1 机器人基础通讯与操作类型 2 机器人常见操作方式 3 方向模式与坐标模式控制	1 与机器人进行通讯 2 控制机器人进行运动 3 操作机器人的IO口，夹爪等信号	5小时
7	**uiFlow**操作机械臂	1 了解可视化编程界面基本架构与关系：传感器、执行器与流程 2 变量、循环与判断 3 机械臂控制方法方法	1 显示basic中不同字体 2 根据basic三个按钮让机械臂走不同的位置 3 控制机械臂循环走多个点位	10小时
8	**roboFlow**使用	1 机器人常用工业操作系统学习 2 roboFlow常用模块学习：点位、快速移动、IO控制与输入 3 roboFlow高级模块学习：循环，判断、托盘程序	1 操控机械臂运动 2 IO输入与输出基本控制 3 循环控制与判断	5小时
9	智能货仓套装	1 机器人点位运动学习 2 托盘与放置规则 3 末端执行器控制原理	1 操作机器人进行不同点位运动 2 操作机器人进行抓取放置 3 夹爪的控制与识别	15小时

序号	目标知识点	理论	实践	预计学时
10	图像识别相关算法	1 StickV视觉传感器介绍 2 maixpy介绍与编程环境 3 常用颜色识别方法与策略 4 常用形状识别方法与策略 5 常用面积识别方法与策略 6 json数据传输	1 搭建virtualbox环境 2 不同颜色读取 3 不同形状识别 4 数据传输与在uiflow中的读取	20小时
11	视觉与机器人联调	1 关联世界与相机坐标系 2 二维码图像标定 3 移动与矫正	1 操作机械臂到相机坐标系 2 机械臂在相机坐标系内运动 3 重新标定与设定	10小时
12	人工智能套装	1 流程图学习与制作 2 电气连接图学习与制作 3 形状分类注意事项与操作策略	1 传感器连接 2 夹爪执行器连接与驱动 3 机械臂驱动与uiFlow联调	20小时

除此之外，您还可以购买我们的**工业4.0**套装，通过该套装，您将学习到仿真工业场景的搭建与使用。

其他问题

如果以上学习内容无法满足您的实际使用需求，可以联系**大象机器人小管家**进行进一步沟通
我们提供软件与硬件的定制服务，费用根据**实际产生的成本**而定。

4 使用用例

myCobot可以适用于个人学习应用或者教育行业应用两大类用途。

myCobot

个人应用

休闲娱乐

可伴随节奏跳舞、敲击乐器，还可拍照录像、写字画画、玩游戏等

智能家具

搭配夹爪、吸泵远程操控机械臂抓取搬运物品，如搅拌咖啡、抓取面包等

教学教具

工业机械臂模拟教学，K12寓教于乐体验教学，高性价比科研实验室小助手

科技设备

搭载AGV小车实现自动导航搬运，结合视觉做红外热像仪、人脸追踪等

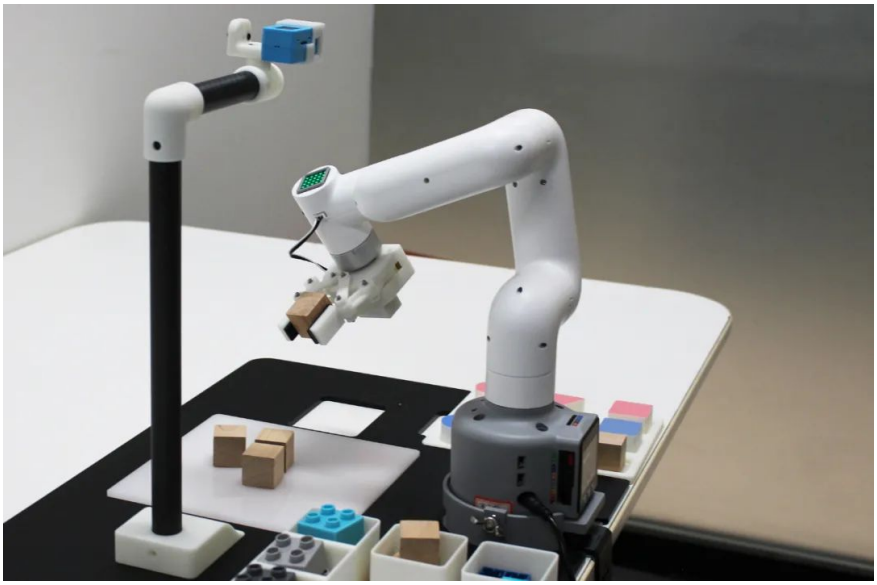
商业展示

远程遥控、360°循环自动展示、全息投影实现产品twins，结合产品做方案展示

行业应用

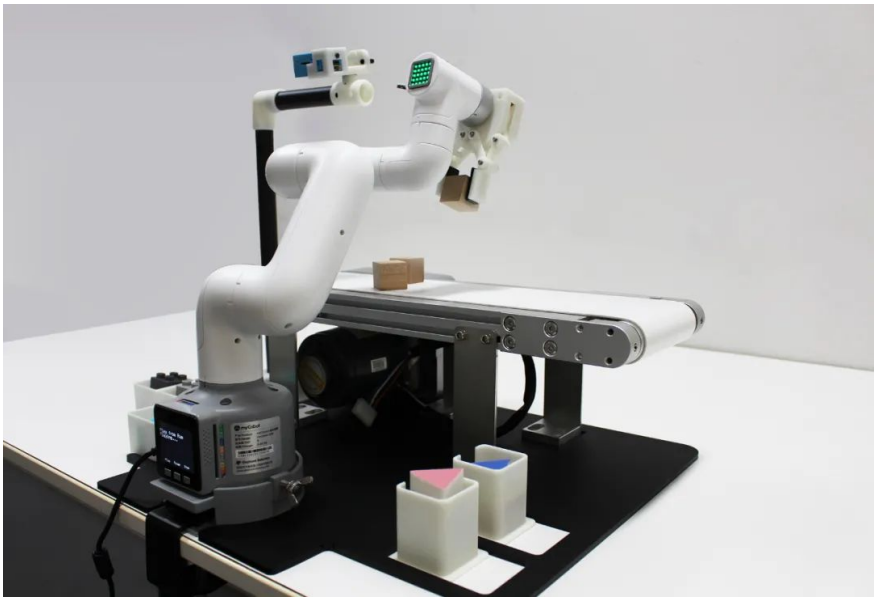
1 创客教育**/K12**教育

- 学习平台：myCobot + Arduino/ROS + uiFlow
- 学习目的：自主开发，创意学习
- 对应套装：基础周边套装



2 人工智能课程

- 学习平台：myCobot + StickV摄像头
- 学习目的：机械臂算法，视觉识别算法
- 对应套装：人工智能套装
- 套装内容：视觉系统，上料台，下料台



3 职业教育**/**高职教育

- 学习平台：myCobot + roboFlow
- 学习目的：从0到1学习机器人算法；机器人运动；机器人仿真

- 对应套装：人工智能套装
- 套装内容：视觉系统，传送带，上料台，下料台，物料挡块

4 快速开始 - 开箱必看

Step 1: 箱子里的物品 What's in the Box

包装箱到位后，请先确认机器人包装完好无损，**如有丢失或破损请及时与物流公司及所在地区的供应商联系。**拆封后，请根据物品清单对箱内实际物品进行核对。

1 myCobot机械臂【标准套】所含产品内容

- myCobot机械臂（型号myCobot-280）
- myCobot机械臂-产品画册
- myCobot机械臂-配套电源
- USB-Type C
- 跳线
- M4*35，杯头内六角，全螺纹，不锈钢螺丝
- 内六角扳手

2 确认工作环境与指标

请将机器人系统设置在符合如表所述条件的环境中，以便发挥、维持本机的性能并安全地进行使用。

环境	指标
温度	-10℃~45℃
相对湿度	20%~70%
室内外要求	室内
其他环境要求	避免阳光照射。 远离灰尘、油烟、盐分、铁屑等。 远离易燃性、腐蚀性液体与气体。 不得与水接触。 不传递冲击与振动等。 远离强电磁干扰源

Step 2: 固定机器人 Fix the robot

您可以使用我们的**底座周边**固定机械臂，或者设计您自己定制的底座进行固定。

具体的固定方法可以参考机械臂的固定

Step 3: 电源

myCobot**必须使用外部电源**进行供电，以提供足够的电量：

- 额定电压：7-9V
- 额定电流：3-5A
- 插头类型：DC 5.5mm x 2.1

注意，不能仅仅使用插入**Basic**的**TypeC**进行供电。

使用官方适配的电源，以免对机械臂造成损害。

点亮屏幕进入默认“拖动示教”程序（若屏幕未能亮起，查看“固件下载”）默认软件 - 轨迹录制软件 M5 Basic主控板拥有3颗按键，支持自定义编程和数据写入。本程序开源，可以查看我们的Github。

Step 4: 开始拖动试教

- 1、 录制：进入录制模式后，选择录制存储位置
  - 按键 A：存储至Ram
  - 按键 B：存储至储存卡
  - 按键C：退出录制模式
- 2、 开始录制，点击record， 选择存储位置后，手动拖动机械臂，完成目标动作。此时动作会被记录并储存。
  - **注意**：默认录制时间是100s，如果时间过长会超过内存。您可以通过修改代码进行定制，或建议通过电脑端进行录制。
- 3、 播放
  - 按键 A：开始播放已记录动作
  - 按键 B：暂停
  - 按键 C：退出播放

开源代码：以上拖动试教的代码是**开源**的，名称为**MainControl**，您可以通过登录Github进行代码下载与定制。参见以下链接：
https://github.com/elephantrobotics/myCobot/tree/main/Arduino/MycobotBasic/examples

视频教程：https://www.bilibili.com/video/BV16t4y167vw/

快速开始 raspberry-mycobot

开启你的 mycobot

首先，你要向下图一样连接好，电源、显示器、键盘和鼠标，然后通过红色的开关启动 mycobot。

Python 使用

快速开始

在树莓派版本的 mycobot 中我们已经预装了 python API 的包 `pymycobot`，只需要在代码中使用即可。

你可以在任意位置创建一个 python 的文件，如：`light_led.py`,然后在文件中写入代码如下：

```
from pymycobot.mycobot import MyCobot
from pymycobot import PI_PORT, PI_BAUD

mc = MyCobot(PI_PORT, PI_BAUD)
mc.set_color(255, 0, 0)
```

保存后，运行它。你将看到 mycobot 顶部的灯板点亮为红色。

更新库

如果我们的库有了更新，也可以在树莓派上很容易的同步跟新。

python2 更新：

```
[sudo] pip install pymycobot --upgrade
```

python3 更新：

```
[sudo] pip3 install pymycobot --upgrade
```

更多样例

除了以上类容之外，我们还提供了，更多案例，来帮助用户使用 python API。

你可以到该地址下载：https://www.elephantrobotics.com/wp-content/uploads/2021/04/PythonAPI教程.zip

Ros 使用

在树莓派版本的 mycobot 中我们已经预装了 Ros kinetic 和 官方提供了包 `myCobotROS`，可以很容易的使用。

1. 进入工作目录：

```
cd ~/ros_catkin_ws
source devel/setup.bash
```

1. 去到 myCobotROS 的 redeme，从 `3.Visualization in RViz` 接着即可，地址如下：

https://github.com/elephantrobotics/myCobotROS#3-visualization-in-rviz

这是myCobot的Python API

这是用于与mycobot进行串行通信并对其进行控制的python API。

安装

注意事项：

确保将 `Atom` 其闪烁到顶部的Atom中， `Transponder` 将其闪烁到基本的Basic中。固件 `Atom` 和 `Transponder` 下载地址：https：//github.com/elephantrobotics/myCobot/tree/main/Software 您还可以使用myStudio对其进行刷新，myStudio地址：https：//github.com/elephantrobotics/myStudio/releases

Pip

```
pip install pymycobot --upgrade --user
```

Source code

```
git clone https://github.com/elephantrobotics/pymycobot.git <your-path>
cd <your-path>/pymycobot
# Install
[sudo] python2 setup.py install
# or
[sudo] python3 setup.py install
```

Usage:

```
from pymycobot.mycobot import mycobot
from pymycobot.mycobot import Angle, Coord
from pymycobot import PI_PORT, PI_BAUD # For raspberry pi version of mycobot.
```

The `demo` directory stores some test case files.

You can find out which interfaces pymycobot provides in `pymycobot/README.md`.

Please go to here.

1、机械臂左右摆动

一、API 简介

控制机械臂左右摆动所使用的 API 为：

1、 `MyCobot(port)`

函数功能：初始化一个 MyCobot 对象。

参数说明： `port` ：String 类型数据，是控制机械臂的端口号，windows 系统可以在设备管理器中的端口处查看。

2、 `get_angles()`

函数功能：获得机械臂六个关节点的角度。

返回值：返回值的类型是 list，共有六个元素数据，分别对应关节 1~6。

3、 `send_angles(degrees,speed)`

函数功能：一次性设置六个关节点的角度。

参数说明：

`degrees` ：list 类型参数，必须包含六个关节点的角度数据。六个关节点的角度取值范围均是-180~180。

`speed` ：int 类型数据，取值范围 0~100。表示机械臂运行到指定位置时的速度，值越大速度越大。

4、 `send_angle(id, degree, speed)`

函数功能：设置单个关节的角度。

参数说明：

`id` ：代表机械臂的关节，一共有六个关节，有特定的表示方法。关节一的表示法： `Angle.J1.value` 。

`degree` ：表示关节的角度，取值范围：-180~180。

`speed` ：表示机械臂运动的速度。

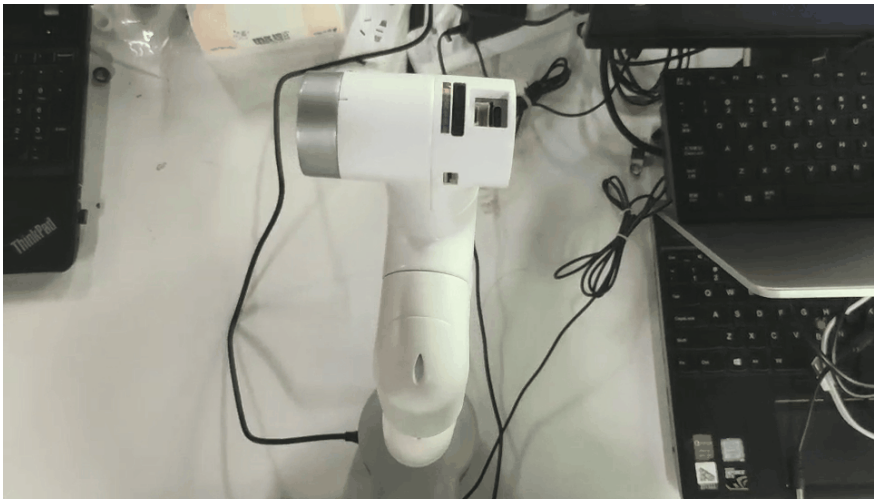
5、 `release_all_servos()`

函数功能：放松机械臂，让其可以随意手动摆动。

二、代码内容

```python
from pymycobot.mycobot import MyCobot
from pymycobot.genre import Angle
from pymycobot import PI_PORT, PI_BAUD  # 当使用树莓派版本的mycobot时，可以引用这两个变量
import time

# 初始化一个MyCobot对象
mc = MyCobot(PI_PORT, PI_BAUD)

# 获得当前位置的坐标
angle_datas = mc.get_angles()
print(angle_datas)

# 用数列传递传递坐标参数，让机械臂移动到指定位置
mc.send_angles([0, 0, 0, 0, 0, 0], 50)
print(mc.is_paused())
# 设置等待时间，确保机械臂已经到达指定位置
# while not mc.is_paused():
time.sleep(2.5)

# 让关节1移动到90这个位置
mc.send_angle(Angle.J1.value, 90, 50)

# 设置等待时间，确保机械臂已经到达指定位置
time.sleep(2)

# 设置循环次数
num = 5

# 让机械臂左右摇摆
while num > 0:
    # 让关节2移动到50这个位置
    mc.send_angle(Angle.J2.value, 50, 50)

    # 设置等待时间，确保机械臂已经到达指定位置
    time.sleep(1.5)

    # 让关节2移动到-50这个位置
    mc.send_angle(Angle.J2.value, -50, 50)

    # 设置等待时间，确保机械臂已经到达指定位置
    time.sleep(1.5)

    num -= 1

# 让机械臂缩起来。你可以手动摆动机械臂，然后使用get_angles()函数获得坐标数列，
# 通过该函数让机械臂到达你所想的位置。
mc.send_angles([88.68, -138.51, 155.65, -128.05, -9.93, -15.29], 50)

# 设置等待时间，确保机械臂已经到达指定位置
time.sleep(2.5)

# 让机械臂放松，可以手动摆动机械臂
mc.release_all_servos()
```

三、效果展示


ThinkPad

2、机械臂头部智能规划路线

一、API 知识准备

`send_coords([x,y,z,rx,ry,rz],speed,model)` 是用来控制机械臂头部以指定姿态移动到指定点。它主要用于实现智能规划机械臂头部从一个位置到另一个指定位置。X,Y,Z 表示的是机械臂头部在空间中的位置（该坐标系为直角坐标系），rx,ry,rz 表示的是机械臂头部在该点的姿态（该坐标系为欧拉坐标）。算法的实现以及欧拉坐标的表示需要一定的学术知识，这里不对其过多的讲解，我们只要懂得直角坐标系就可以很好的使用这个函数了。

二、API 简介

1、 `send_coords([x,y,z,rx,ry,rz],speed,model)`

函数功能：智能规划路线，让机械臂头部从原来点移动到指定点。

参数说明：

`[x,y,z,rx,ry,rz]`：该参数中以 x,y,z 组合成一个空间直角坐标系，以机械臂底部为原点，前方为 x 正轴，右方为 y 正轴，上方为 z 正轴。`[x,y,z]` 表示为机械臂头部所在的位置。`[rx,ry,rz]` 表示的是机械臂头部的姿态，你可以调整机械臂头部的姿态然后使用 `get_coords()` 函数获取该点的姿态，这样你就不需要了解欧拉坐标系也能很好的该函数了。

`speed`：表示机械臂运动的速度。取值范围为 0~100，值越大速度越快。

`model`：取值限定 0 和 1。0 表示机械臂头部移动的路径为非线性，即随机规划路线，只要机械臂头部以保持规定的姿态移动到指定点即可。1 表示机械臂头部移动的路径为线性的，即智能规划路线让机械臂头部以直线的方式移动到指定点。

2、 `get_coords()`

函数功能：获取此时机械臂头部的空间坐标以及当前姿态。

返回值：返回的类型是包含六个 float 元素的 list 集合，前三个坐标为 x,y,z 表示机械臂头部的坐标，后三个坐标 rx,ry,rz 表示机械臂头部的姿态。

3、 `send_coord(id,coord,speed)`

函数功能：仅单独修改机械臂头部空间坐标中 x,y,z 轴的其中一个坐标。

参数说明：

`id`： `genre.Coord` 类型数据，它表示的是机械臂头部的 X,Y,Z 轴。表示方式举例：X 轴 `Coord.X.value`。（可参考案例）

`coord`：float 类型数据，表示的是修改坐标值。

`speed`：表示机械臂运动的速度。取值范围为 0~100，值越大速度越快。

三、代码内容

```python
from pymycobot.mycobot import MyCobot
from pymycobot.genre import Coord
from pymycobot import PI_PORT, PI_BAUD  # 当使用树莓派版本的mycobot时，可以引用这两个变量
import time

# 初始化一个MyCobot对象
mc = MyCobot(PI_PORT, PI_BAUD)
# 获取当前头部的角度以及姿态
coords = mc.get_coords()
print(coords)
# 智能规划路线，让头部以线性的方式到达[59.9,-65.8,250.7]这个坐标，以及保持[-50.99,83.14,-52
mc.send_coords([59.9, -65.8, 250.7, -50.99, 83.14, -52.42], 80, 1)
# 设置等待时间
time.sleep(1.5)
# 智能规划路线，让头部以线性的方式到达[59.9,-65.8,350.7]这个坐标，以及保持[-50.99,83.14,-52
mc.send_coords([59.9, -65.8, 350.7, -50.99, 83.14, -52.42], 80, 1)
# 设置等待时间
time.sleep(1.5)
# 仅改变头部的x坐标，设置头部的x坐标为-40。让其智能规划路线让头部移动到改变后的位置
mc.send_coord(Coord.X.value, -40, 70)
```

四、展示效果

3、机械臂安全控制

一、API 简介

1、 `is_power_on()`

函数功能：判断机械臂是否供电。

返回值：1 已供电，0 断电，-1 错误。

2、 `power_on()`

函数功能：为机械臂供电。

3、 `power_off()`

函数功能：为机械臂断电，所有功能将失效。注意：断电后无法让机械臂放松，即 `set_free_mode()` 失效。

4、 `pause()`

函数功能：暂停机械臂运动。

5、 `resume()`

函数功能：恢复机械臂运动。

6、 `stop()`

函数功能：机械臂停止运动

7、 `is_in_position(data,flag)`

函数功能：判断机械臂是否到达指定位置

参数说明：

`data`：包含六个元素的 list 集合，表示角度集合，或者是机械臂头部数据集合。

`flag`：1 表示 data 数据表示机械臂头部数据，0 表示 data 数据表示机械臂角度集合数据。

8、 `is_paused()`

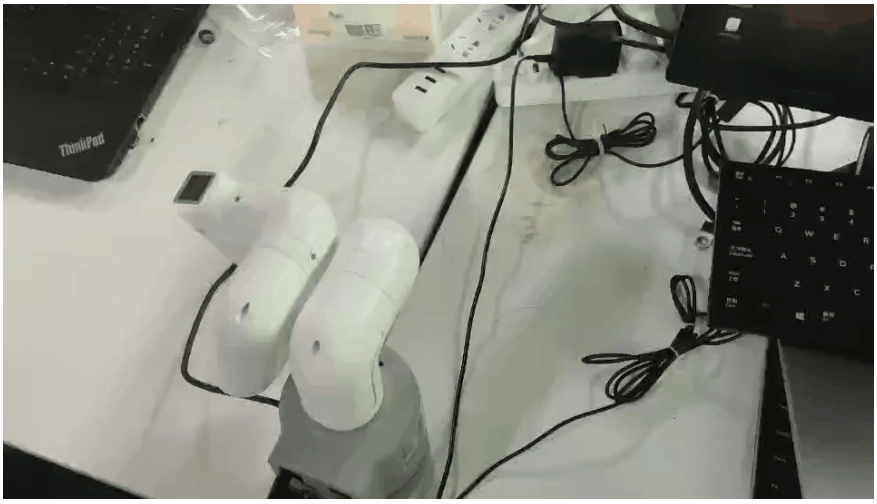
函数功能：判断机械臂是否处于暂停状态

返回值： 1 表示处于暂停状态， 0 表示没有处于暂停状态，－ 1 表示错误

二、代码内容

```python
from pymycobot.mycobot import MyCobot
from pymycobot import PI_PORT, PI_BAUD  # 当使用树莓派版本的mycobot时，可以引用这两个变量
import time

# 初始化一个MyCobot对象
mc = MyCobot(PI_PORT, PI_BAUD)
# 判断机械臂是否供电，若无供电需要先为其供电
if not mc.is_power_on():
    # 为机械臂供电
    mc.power_on()
# 机械臂以30的速度到达[0,0,0,0,0,0]位置
mc.send_angles([0, 0, 0, 0, 0, 0], 30)
# 获取当前时间
start = time.time()
# 判断机械臂是否到达[0,0,0,0,0,0]位置
while not mc.is_in_position([0, 0, 0, 0, 0, 0], 0):
    # 恢复机械臂的移动
    mc.resume()
    # 让机械臂移动0.5s
    time.sleep(0.5)
    # 暂停机械臂移动
    mc.pause()
    # 判断移动是否超时
    if time.time() - start > 9:
        # 停止机械臂的移动
        mc.stop()
        break
# 获取当前时间
start = time.time()
# 机械臂以30的速度到达[88.68, -138.51, 155.65, -128.05, -9.93, -15.29]位置
mc.send_angles([88.68, -138.51, 155.65, -128.05, -9.93, -15.29], 30)
# 判断机械臂是否到达[88.68, -138.51, 155.65, -128.05, -9.93, -15.29]位置
while not mc.is_in_position([88.68, -138.51, 155.65, -128.05, -9.93, -15.29], 0):
    # 恢复机械臂的移动
    mc.resume()
    # 让机械臂移动0.5s
    time.sleep(0.5)
    # 暂停机械臂移动 可以用is_paused()判断机械臂是否处于暂停状态
    mc.pause()
    # 判断移动是否超时
    if time.time() - start > 9:
        mc.stop()
        # 停止机械臂的移动
        break
# 执行完操作后，为机械臂断电
mc.power_off()
```

三、效果展示


ThinkPad

4、机械臂设备检测

一、API 简介

1、 `is_all_servo_enable()`

函数功能：判断六个关节点是否可以正常使用。

返回参数：1 表示正常工作，0 表示不能工作，-1 表示错误报警。

2、 `jog_angle(joint_id,direction,speed)`

函数功能：让一个关节点持续移动。

参数说明：

`joint_id`：取值范围 1~6，六个整数分别表示关节点 1~6。

`direction`：1 表示角度增加，0 表示角度减少。

`speed`：表示增加减少的速度。

3、 `jog_stop()`

函数功能：停止关节点的移动。

4、 `release_servo(servo_id)`

函数功能：放松指定关节点。

参数说明：

`servo_id`：取值范围 1~6，六个整数分别表示关节点 1~6。

5、 `focus_servo(servo_id)`

函数功能：固定指定关节点。

参数说明：

`servo_id`：取值范围 1~6，六个整数分别表示关节点 1~6。

二、代码内容

```python
from pymycobot.mycobot import MyCobot
from pymycobot import PI_PORT, PI_BAUD  # 当使用树莓派版本的mycobot时，可以引用这两个变量
import time

# 初始化一个MyCobot对象
mc = MyCobot(PI_PORT, PI_BAUD)
# 判断机械臂是否供电，若无供电需要先为其供电
if not mc.is_power_on():
    # 为机械臂供电
    mc.power_on()

# 检测六个关节是否正常工作
# 也可以使用is_servo_enable(servo_id)改变单个单个校验
if mc.is_all_servo_enable():
    # 为机械臂断电
    mc.power_off()
    # 判断机械臂是否断电
    if mc.is_all_servo_enable() == -1:
        print("机械臂供电断电正常")
    else:
        print("机械臂供电断电失败")
        exit(0)

# 为机械臂重新充电
mc.power_on()
# 将机械臂设置到零位上
mc.send_angles([0, 0, 0, 0, 0, 0], 30)
# 获取当前时间
start = time.time()
# 判断机械臂是否到达零点
while not mc.is_in_position([0, 0, 0, 0, 0, 0], 0):
    # 让机械臂恢复运动
    mc.resume()
    # 让机械臂移动0.5s
    time.sleep(0.5)
    # 暂停机械臂移动
    mc.pause()
    # 判断移动是否超时
    if time.time() - start > 9:
        # 停止机械臂的移动
        print("移动到零位失败")
        # 中止程序
        exit(0)

# 检测六个关节点的移动情况
for i in range(1, 7):
    # 让关节点i向右以速度15移动
    mc.jog_angle(i, 0, 15)
    # 让关节点移动1.5s
    time.sleep(1.5)
    # 停止关节点移动
    mc.jog_stop()
    # 让关节点i向左以速度15移动
    mc.jog_angle(i, 1, 15)
    # 让关节点移动3s
    time.sleep(3)
    # 停止关节点移动
    mc.jog_stop()
    # 让关节点i向右以速度15移动
    mc.jog_angle(i, 0, 15)
    # 让关节点移动1.5s
    time.sleep(1.5)
    # 停止关节点移动
    mc.jog_stop()
    print(str(i) + "号关节点正常工作")
```

```
    # 等待0.8s
    time.sleep(0.8)

# 获取当前时间
start = time.time()
# 机械臂以30的速度到达[87.27, -139.13, 153.72, -160.92, -74.44, 7.55]位置
mc.send_angles([87.27, -139.13, 153.72, -160.92, -74.44, 7.55], 30)
# 判断机械臂是否到达[87.27, -139.13, 153.72, -160.92, -74.44, 7.55]位置
while not mc.is_in_position([87.27, -139.13, 153.72, -160.92, -74.44, 7.55], 0):
    # 恢复机械臂的移动
    mc.resume()
    # 让机械臂移动0.5s
    time.sleep(0.5)
    # 暂停机械臂移动
    mc.pause()
    # 判断移动是否超时
    if time.time() - start > 9:
        mc.stop()
        # 停止机械臂的移动
        break

# 该for循环相当于set_free_model()
for i in range(1, 7):
    mc.release_servo(i)

# 依次固定六个关节点
# for i in range(1, 7):
#     mc.focus_servo(i)
```

三、效果展示

5、机械臂跳舞

一、API 简介

1、 `set_color(R, G, B)`

函数功能：设置机械臂头部灯的颜色

参数说明：

R,G,B：分别对应颜色数组[R,G,B]的值

二、代码内容

```
from pymycobot.mycobot import MyCobot
from pymycobot import PI_PORT, PI_BAUD  # 当使用树莓派版本的mycobot时，可以引用这两个变量
import time

if __name__ == '__main__':
    # 初始化一个MyCobot对象
    mc = MyCobot(PI_PORT, PI_BAUD)
    # 设置开始开始时间
    start = time.time()
    # 让机械臂到达指定位置
    mc.send_angles([-1.49, 115, -153.45, 30, -33.42, 137.9], 80)
    # 判断其是否到达指定位置
    while not mc.is_in_position([-1.49, 115, -153.45, 30, -33.42, 137.9], 0):
        # 让机械臂恢复运动
        mc.resume()
        # 让机械臂移动0.5s
        time.sleep(0.5)
        # 暂停机械臂移动
        mc.pause()
        # 判断移动是否超时
        if time.time() - start > 3:
            break
    # 设置开始时间
    start = time.time()
    # 让运动持续30秒
    while time.time() - start < 30:
        # 让机械臂快速到达该位置
        mc.send_angles([-1.49, 115, -153.45, 30, -33.42, 137.9], 80)
        # 将灯的颜色为[0,0,50]
        mc.set_color(0, 0, 50)
        time.sleep(0.7)
        # 让机械臂快速到达该位置
        mc.send_angles([-1.49, 55, -153.45, 80, 33.42, 137.9], 80)
        # 将灯的颜色为[0,50,0]
        mc.set_color(0, 50, 0)
        time.sleep(0.7)

    # mc.release_all_servos()
```

三、效果展示

6、夹爪的使用

一、API 简介

1、 `is_gripper_moving()`

函数功能：判断夹爪是否正在运行。

返回参数：

1 表示正在运行，0 表示没有运行，-1 表示出错

2、 `set_encoder(joint_id, encoder)`

函数功能：让指定关节点转动到指定位置

参数说明：

`joint_id`：取值范围 1~7，分别表示 1~6 个关节点以及夹爪。

`encoder`：取值范围 0~4096，2048 表示角度时的 0。

3、 `set_encoders(encoders, sp)`

函数功能：让机械臂移动到指定位置。

参数说明：

`encoders`：六个 int 元素的 list 集合，六个 encoder 数据的顺序分别代表 1~6 个关节点的位置。

`sp`：表示机械臂转动的速度。

4、 `get_encoder(joint_id)`

函数功能：获取指定关节点的 encoder 数据。

参数说明：

`joint_id`：取值范围 1~7，分别表示 1~6 个关节点以及夹爪。

返回值：

`encoder`：表示该关节的 encoder 数据信息。

5、 `set_gripper_value(value, speed)`

函数功能：让夹爪以指定的速度转动到指定的位置。

参数说明：

`value`：表示夹爪所要到达的位置，取值范围 0~4096。

`speed`：表示以多少的速度转动，取值范围 0~100。

6、 `get_gripper_value()`

函数功能：获取夹爪的 `encoder` 数据信息。

返回值：

`encoder`：夹爪的数据信息。

7、 `set_gripper_state(flag, speed)`

函数功能：让夹爪以指定的速度达到指定的状态。

参数说明：

`flag`：1 表示夹爪合拢，0 表示夹爪打开。

`speed`：表示以多快的速度达到指定的状态，取值范围 0~100。

二、代码内容

```python
from pymycobot import PI_PORT, PI_BAUD  # 当使用树莓派版本的mycobot时，可以引用这两个变量
from pymycobot.mycobot import MyCobot
import time


def gripper_test(mc):
    print("Start check IO part of api\n")
    # 检测夹爪是否正在移动
    flag = mc.is_gripper_moving()
    print("Is gripper moving: {}".format(flag))
    time.sleep(1)

    # Set the current position to (2048).
    # Use it when you are sure you need it.
    # Gripper has been initialized for a long time. Generally, there
    # is no need to change the method.
    # mc.set_gripper_ini()
    # 设置关节点1，让其转动到2048这个位置
    mc.set_encoder(1, 2048)
    time.sleep(2)
    # 设置六个关节位，让机械臂以20的速度转动到该位置
    mc.set_encoders([1024, 1024, 1024, 1024, 1024, 1024], 20)
    time.sleep(3)
    # 获取关节点1的位置信息
    print(mc.get_encoder(1))
    # 设置夹爪转动到2048这个位置
    mc.set_encoder(7, 2048)
    time.sleep(3)
    # 设置夹爪让其转到1300这个位置
    mc.set_encoder(7, 1300)
    time.sleep(3)

    # 以70的速度让夹爪到达2048状态
    mc.set_gripper_value(2048, 70)
    time.sleep(3)
    # 以70的速度让夹爪到达1500状态
    mc.set_gripper_value(1500, 70)
    time.sleep(3)

    # 设置夹爪的状态，让其以70的速度快速打开爪子
    mc.set_gripper_state(0, 70)
    time.sleep(3)
    # 设置夹爪的状态，让其以70的速度快速收拢爪子
    mc.set_gripper_state(1, 70)
    time.sleep(3)

    # 获取夹爪的值
    print("")
    print(mc.get_gripper_value())


if __name__ == "__main__":
    # 初始化一个MyCobot对象
    mc = MyCobot(PI_PORT, PI_BAUD)
    # 让其移动到零位
    mc.set_encoders([2048, 2048, 2048, 2048, 2048, 2048], 20)
    time.sleep(3)
    gripper_test(mc)
```

三、展示效果

7、机械臂校准

一、API 简介

1、 `set_servo_calibration(servo_no)`

函数功能：当调位到零位时发现机械臂的卡口有对不齐的现象，可以使用该方法对不准的关节进行校准。

参数说明：

`servo_no`：取值范围 1~6，分别表示六个关节点。

2、 `is_controller_connected()`

函数功能：判断当前机械臂是否处于可写入程序状态

返回值：1 表示可以写入，0 表示不可以写入，-1 表示错误。

3、 `set_gripper_ini()`

函数功能：对六个关节点进行零位校准。

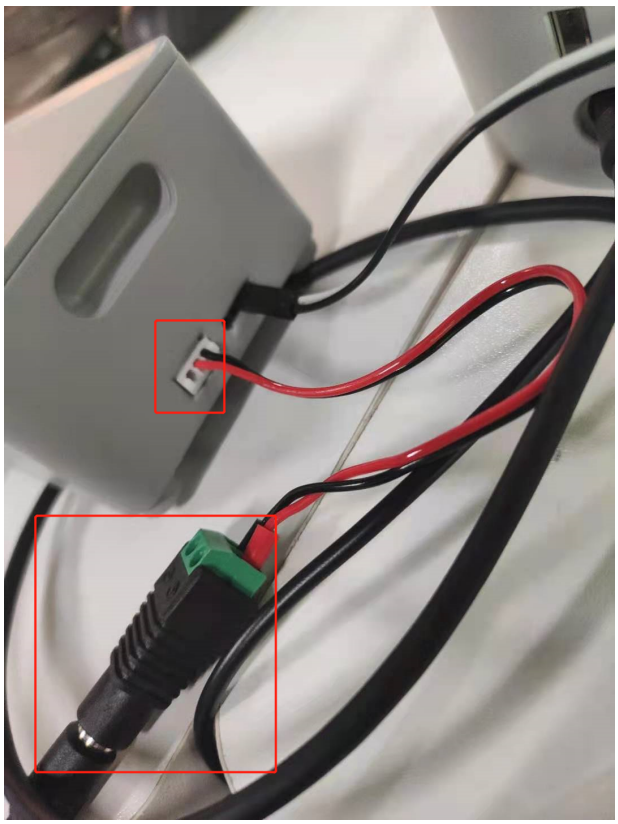
二、代码内容

```
from pymycobot.mycobot import MyCobot
from pymycobot import PI_PORT, PI_BAUD  # 当使用树莓派版本的mycobot时，可以引用这两个变量

# 初始化一个MyCobot对象
mc = MyCobot(PI_PORT, PI_BAUD)

# 检测机械臂是否可烧入程序
if mc.is_controller_connected() != 1:
    print("请正确连接机械臂进行程序写入")
    exit(0)

# 对机械臂进行微调，确保调整后的位置所有卡口都对齐了
# 以机械臂卡口对齐为准，这里给出的仅是个案例
mc.send_angles([0, 0, 18, 0, 0, 0], 20)

# 对此时的位置进行校准，校准后的角度位置表示[0,0,0,0,0,0]，电位值表示[2048,2048,2048,2048,2
# 该for循环相当于set_gripper_ini()这个方法
for i in range(1, 7):
    mc.set_servo_calibration(i)
```

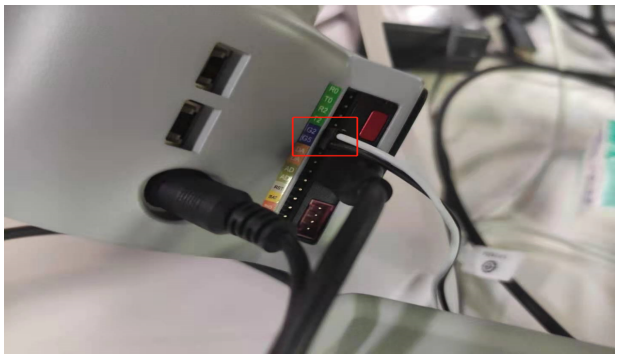
8、控制吸泵

一、设备连接

首先我们需要为吸泵供电如下图所示。



其次我们需要将吸泵与机械臂相连接，连接号为 2 和 5。如下图所示



二、API 介绍

`set_basic_output(pin_no, pin_signal)`

函数功能：通过连接号 `pin_no`，以及控制信号 `pin_signal` 控制外部设备。

参数说明：

`Pin_no`：int 类型参数，即设备底部标注的编号仅取数字部分。

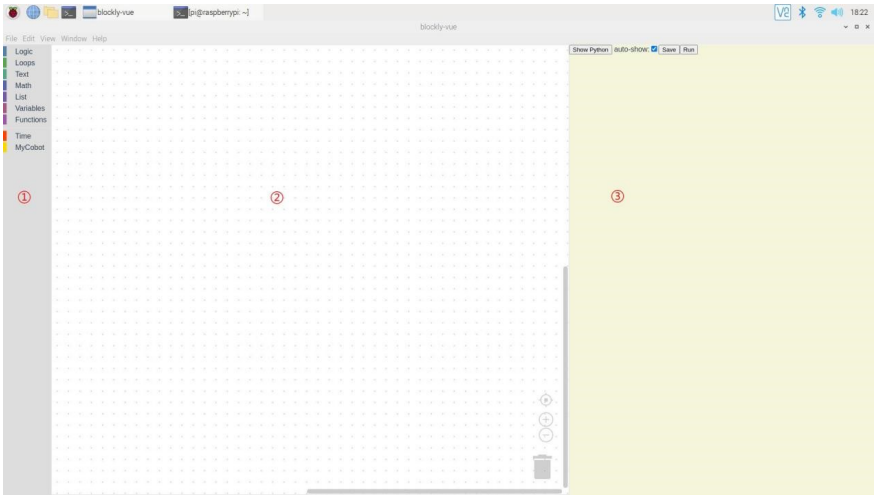
`Pin_signal`：1 表示停止运行，0 表示运行。

三、代码内容

```
from pymycobot.mycobot import MyCobot
from pymycobot import PI_PORT, PI_BAUD  # 当使用树莓派版本的mycobot时，可以引用这两个变量
import time

# 初始化一个MyCobot对象
mc = MyCobot(PI_PORT, PI_BAUD)

# 开启吸泵
def pump_on():
    # 让2号位工作
    mc.set_basic_output(2, 0)
    # 让5号位工作
    mc.set_basic_output(5, 0)

# 停止吸泵
def pump_off():
    # 让2号位停止工作
    mc.set_basic_output(2, 1)
    # 让5号位停止工作
    mc.set_basic_output(5, 1)


pump_off()
time.sleep(3)
pump_on()
time.sleep(3)
pump_off()
time.sleep(3)
```

Myblockly 使用说明

一、界面简介



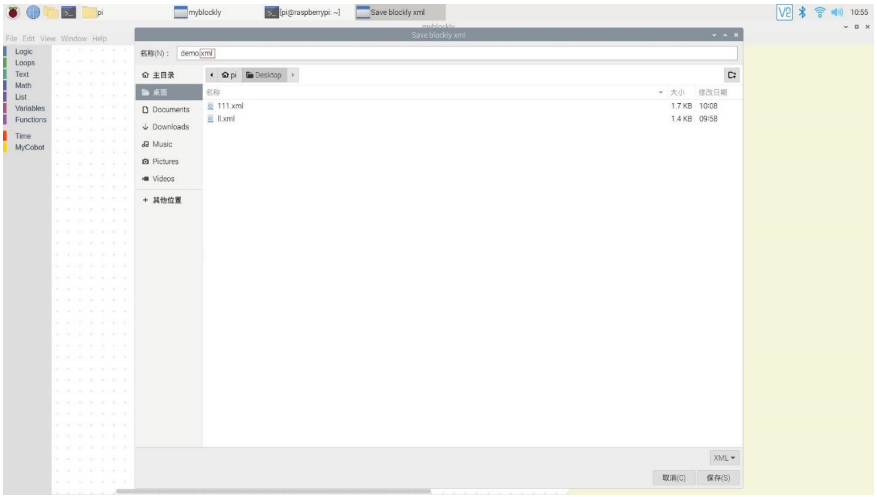
1-1界面展示

如图 1-1 所示，① 表示拼图工具栏，即包含了逻辑控制拼图、变量设置拼图、数学函数拼图、文本类型拼图以及控制机械臂方法拼图等。② 表示拼图画板，将拼图工具栏中的方法模块拉入到拼图画板中，方法模块就会在画板中显示。③ 表示代码显示区，拼接在画板中的方法模块会自动生成 python 代码在代码显示区中。

二、保存加载介绍

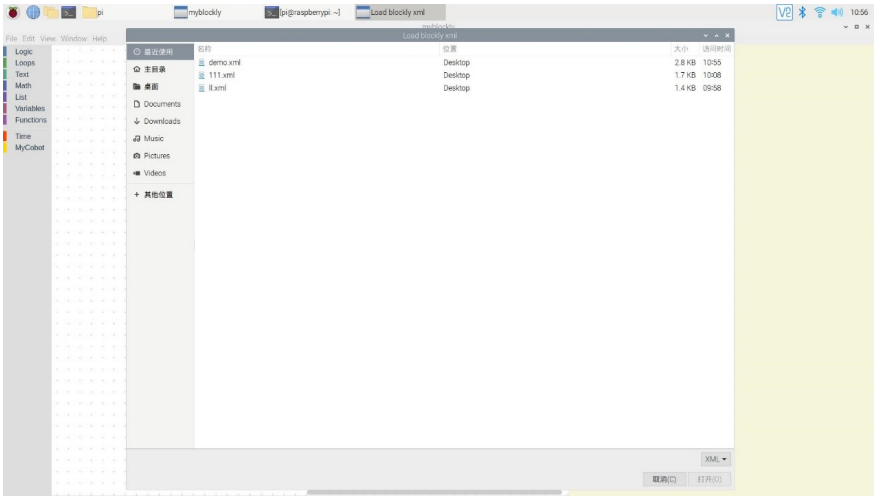
1-2保存加载

写好程序后，可以点击 1-2 中 ① 所指的 save 进行保存，点击后的效果如图 1-3 所示。



1-3保存界面

特别也要注意的是在定义保存文件名时一定要加后缀**.xml**，否则将无法加载保存后的文件。当然若是不小心忘记加后缀**.xml** 也没关系，只需在加载该文件之前对其进行重命名重新加上**.xml** 后缀即可。



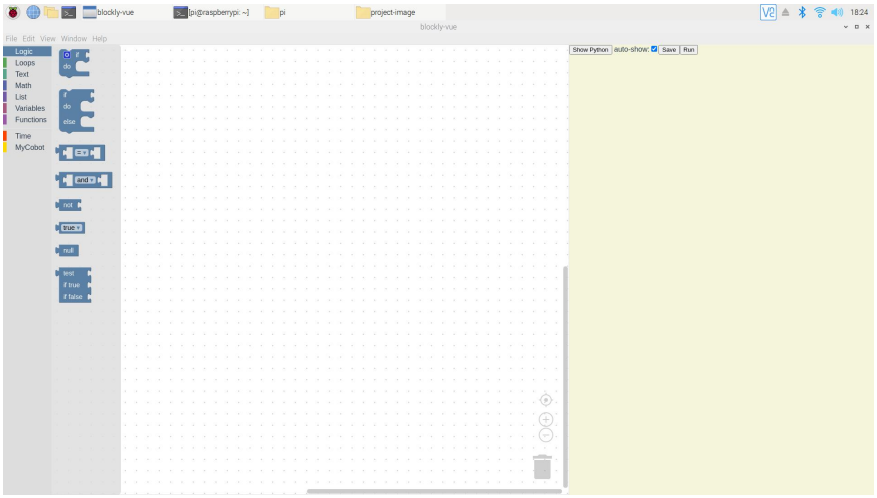
1-4加载界面

可以通过点击图 1-4 中 ② 所指的 load 进行文件加载，点击后的效果如图 1-4 所示。此时加载的文件只能是以.xml 为后缀的文件。

Myblockly模块简介

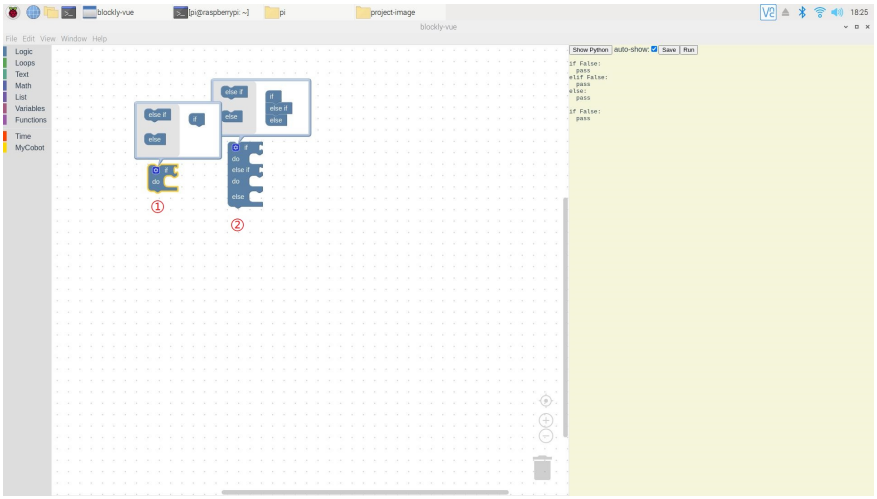
一、Logic模块

1、如图1-1所示即为Logic模块所包含的所有方法。



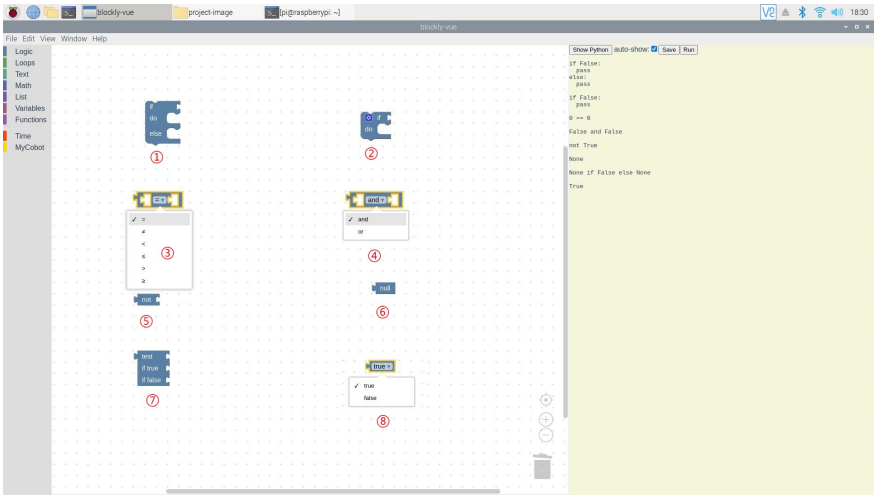
1-1 Logic模块展示

2、方法使用详细讲解



1-2 方法详细（一）

如图1-2所示，①表示if（条件）do（程序）方法，若满足条件即执行程序。该方法拼图中有⚙图标，点击⚙图标即可显示①中上方的提示框。可以拖拽提示框左边的else if或者else选择到右边的if中，为if（条件）do（程序）方法添加else if或者else选项。拖拽结果如②所示。

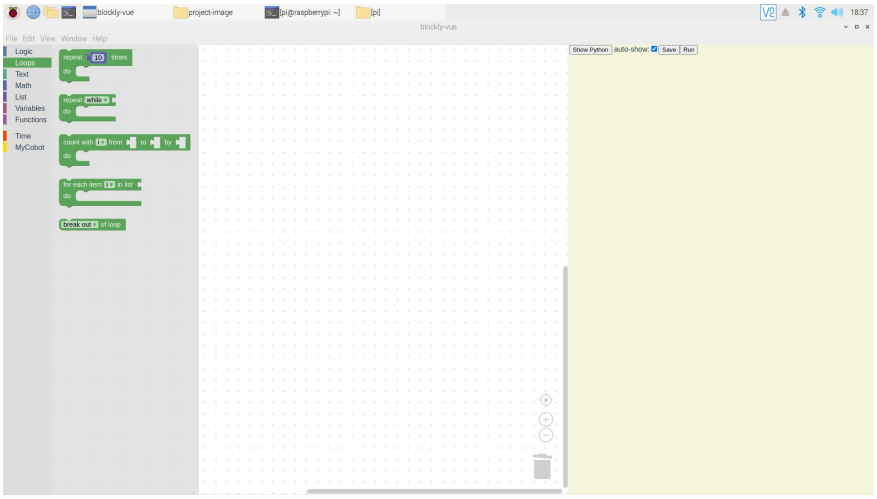


1-3 方法详细（二）

如图1-3所示，①表示if（条件）do（程序1）else（程序2），若满足条件则执行程序1，否则执行程序2。②所表示方法的详细讲解可查看图1-2下方的文字讲解。③所表示的逻辑判断，返回值为true或者false。可以点击③中的下拉框进行逻辑判断选择，如图所示可以选择＝、≠、≥、≤等逻辑判断。④所表示的是与或逻辑判断，返回的值为true或者false。可以点击④中的下拉框进行选择，如图所示可以选择and或者or进行逻辑判断。⑤表示非逻辑，在原有的true或者false基础上取反。⑥表示null值。⑦表示test（条件）if true（程序1）if else（程序2），若满足条件则执行程序1，否则执行程序2。⑧表示true值或者false值，点击⑧中的下拉框可以进行选择。

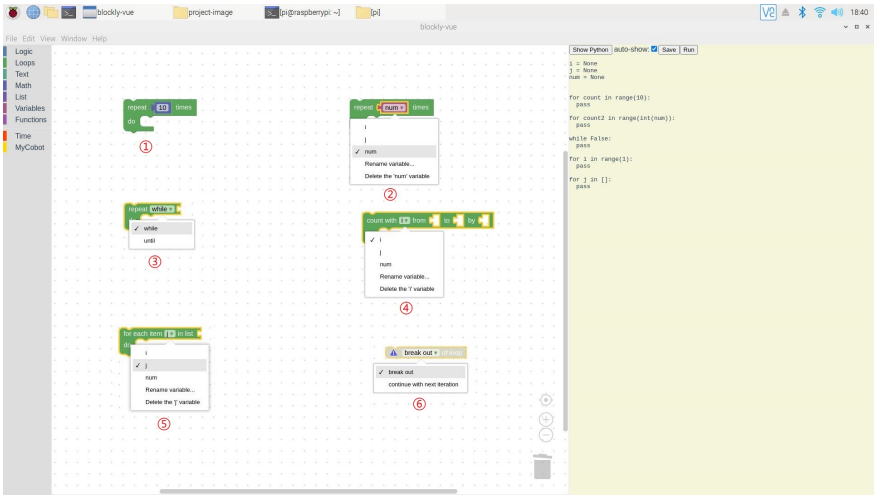
二、Loops模块

1、如图2-1所示即为Loops模块所包含的所有方法。



2-1 Loops模块

2、方法使用详细讲解



2-2 Loops模块方法详细

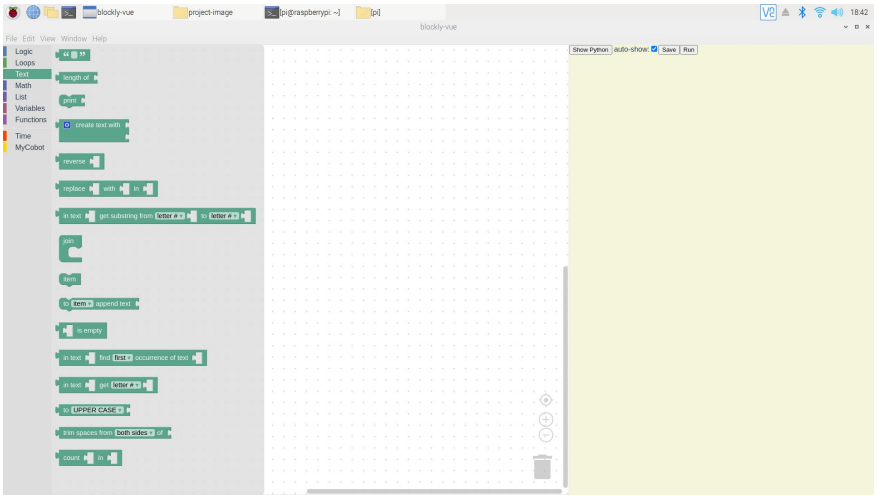
如图2-2所示，①表示重复执行10次do里面的程序。②表示重复变量num次do中的程序（do被遮挡）。点击②中的拉下框可以选择已有的变量如i、j、num。也可以对当前变量进行重命名或者选择删除当前变量。图中⑤和⑥效果也是如此。③表示循环判断方法，可以点击下拉框选择while或者until模式。while表示若条件满足则一直执行do中的程序。until表示一直执行do中的程序，直到条件满足为止。

④count with 变量i turn 数值1 to 数值2 by数值3表示以数值3为步长，从数值1开始到数值2，将这些值返回给变量i。可以参考python函数中for i in range(数值1,数值2,数值3)。⑤表示将list数组的元素循环赋值给变量j，并循环执行do中的程序。⑥方法需要搭配以上的循环方法使用。点击下拉框可以选择break out或者continue，即终止循环或者进行下次循环。

注：在循环中想使用循环中的变量需要设置一致的变量。

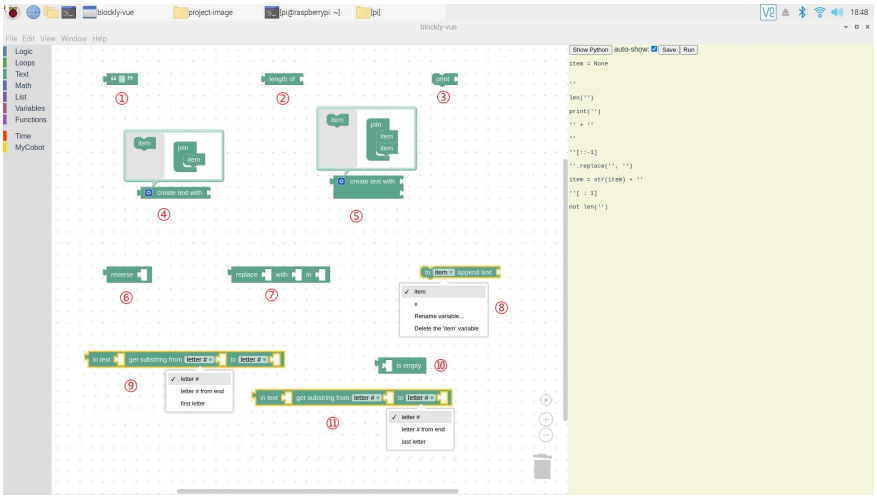
三、Text模块

1、如图3-1所示即为Text模块所包含的所有方法。



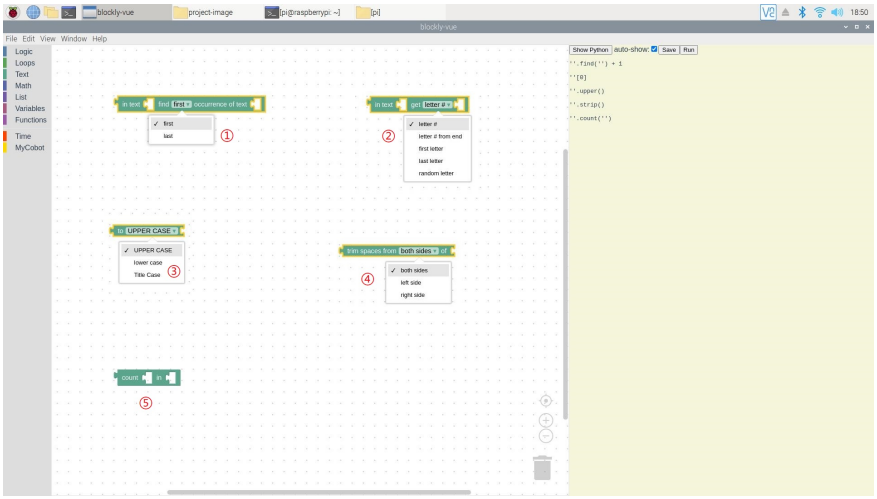
3-1 Text模块

2、方法使用详细讲解



3-2 Text模块方法详细（一）

如图3-2所示，①表示文本内容，可以自定义文本内容。②表示计算指定文本内容的长度。③表示输出文本内容。④通过组合可以创建文本内容，可以点击⚙进行扩张。如⑤所示。⑥表示将文本内容反转展示。⑦表示将文本中的指定内容替换成选定内容。⑧表示将变量item转换成String类型并让其与指定内容进行相加。⑨表示截取指定文本中的内容，第一个空表示文本内容，第二个空表示从文本的哪个数组下标开始，第三个空表示到文本的哪个数组下标截止。⑩表示判定某个变量的数组长度是否为空。

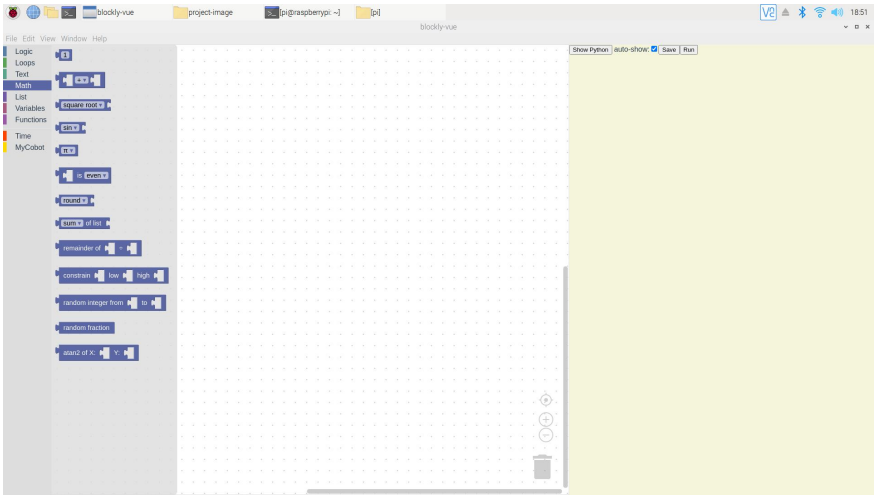


3-3 Text模块方法详细（二）

如图3-3所示，①表示指定字符串在选定字符串中第一次或最后一次出现的位置，可以点击下拉框选择是第一次还是最后一次，如①所示。②表示获取文本中指定位置的字符。③表示将字符串全部大写，可以点击下拉框对其设置全部小写或者首字符大写，如③所示。④表示移除文本头尾指定字符串的内容，可以点击下拉框选择仅移除文本的头部或者尾部。⑤表示统计文本中出现指定字符串的次数。

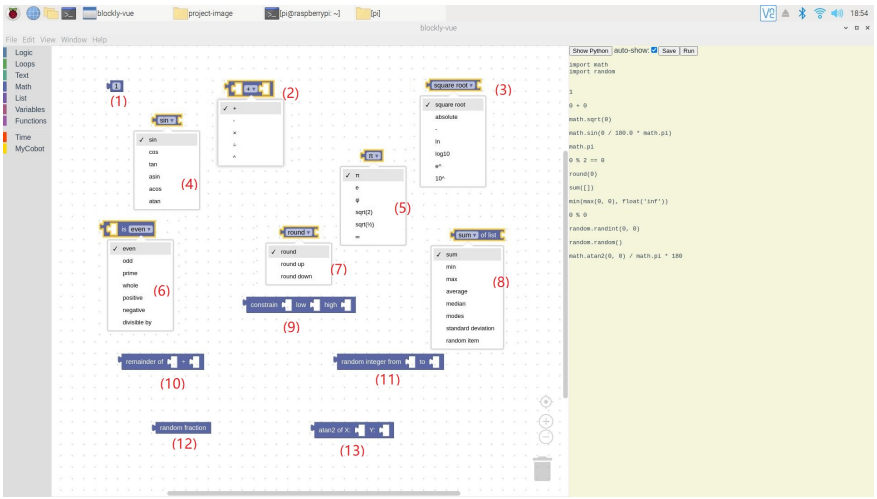
四、Math模块

1、如图4-1所示即为Math模块所包含的所有方法。



4-1 Math模块

2、方法使用详细讲解

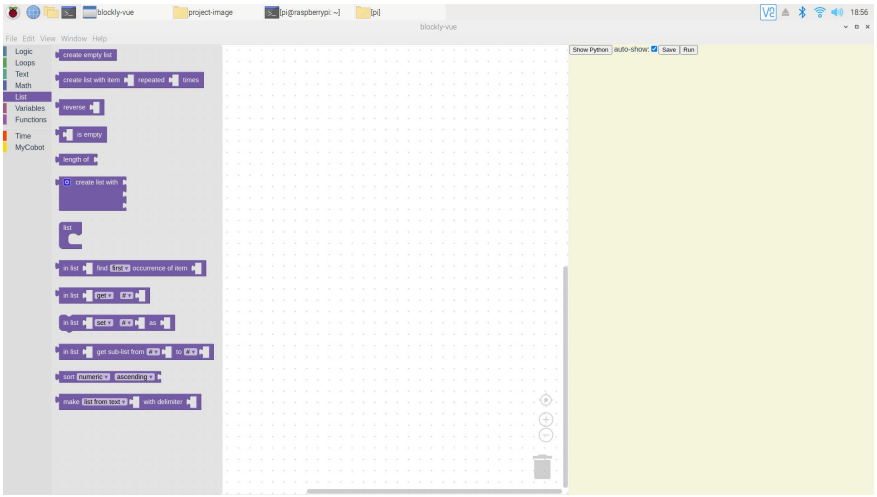


4-2 Math模块方法详细

如图4-2所示，(1)表示数字常量，该数值常量是可以自定义的。(2)表示两个变量逻辑相加减等运算操作，可以点击下拉框指定何种逻辑运算。(3)表示一些简单的数学函数方法，可以点击下拉框选择开根号、取绝对值、10的指定次方等操作。(4)表示一些三角函数方法，可以点击下拉框选择取sin值、取cos值、取tan值等操作。(5)表示一些数学常用的指定常数，可以点击下拉框选择π值、e值、√2等。(6)表示对指定数值进行一些简单的判断，可以点击下拉框选择判定其是否为奇数、是否为偶数、是否为素数等。(7)表示对指定数值进行四舍五入操作，下拉框中round表示默认的四舍五入方法，round up表示向上进行四舍五入，round down表示向下进行四舍五入。(8)表示统计数组中的最大值、最小值、它们的和等，可以点击下拉框进行选择。(9)表示对指定数组进行筛选，low和high分别设置数组的范围。(10)表示一个数除以另一个数所得的余数。(11)表示从自定义范围中随机生成数值。(12)表示一个随机的小数。(13)表示弧度转角度。

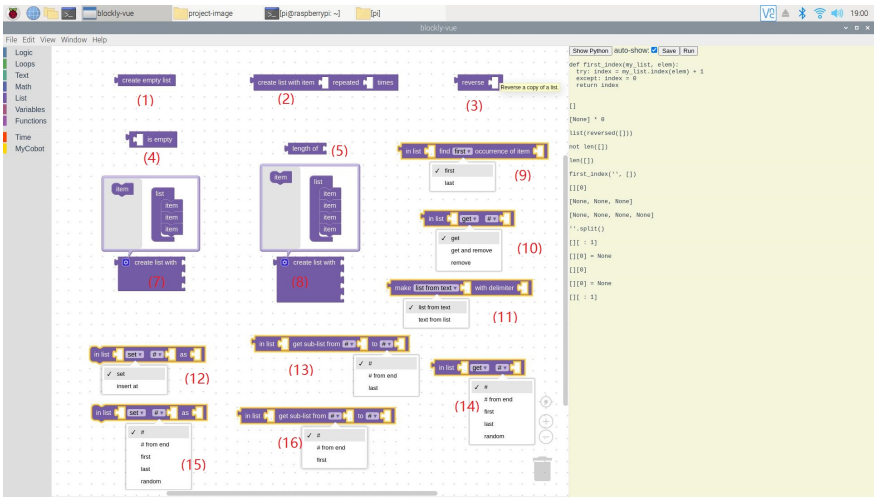
五、List模块

1、如图5-1所示即为List模块所包含的所有方法。



5-1 List模块

2、方法使用详细讲解

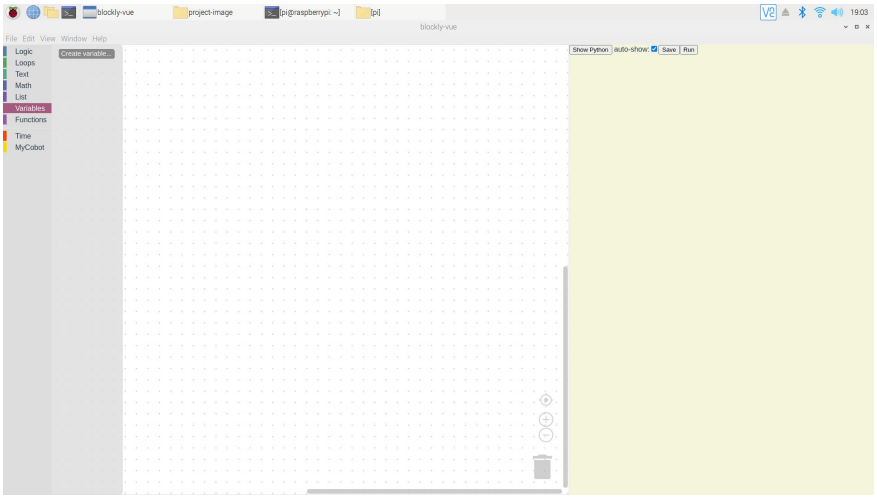


5-2 List模块方法详细

如图5-2所示，(1)表示创建一个空的list数组。(2)表示创建一个数组，该数组为指定一个数重复多少次后组成的数组。如指定元素100让其重复6次，则获得的数组为[100,100,100,100,100,100]。(3)表示将指定数组内的元素反转。(4)判断一个数组是否为空，若为空返回true值，否则返回false值。(5)表示统计指定数组内的元素个数，返回一个int类型值。(6)表示指定元素在数组首次或最后一次出现的位置，可以通过点击下拉框选择是首次还是最后一次。(7)表示通过指定元素创建一个数组，可以通过⚙添加元素。效果如(8)所示。(9)表示通过分割文本获取数组（list from text）或者通过给定文本连接数组获得新文本（text from list），具体的选择可以通过点击下拉框进行选择。(10)表示获取指定list数组指定元素，其下拉框中get表示获取，get and remove表示获取并移除，remove表示移除。(14)中所给的限定选项：#表示从头开始、# from end表示从尾开始、first表示首个元素、last表示最后一个元素、random表示list数组中的随机一个元素。(12)表示设置指定数组位置的元素值（set）或者插入指定元素到数组的指定位置（insert at），可以通过点击下拉框中的选项选择。(15)中表示的含义同(14)中所解释的一致。(13)表示获取数组一系列元素，(13)中所示的筛选选项：#表示从头开始、# from end表示从尾开始、last表示最后一个元素。(16)中所显示的选项含义同(13)一致。

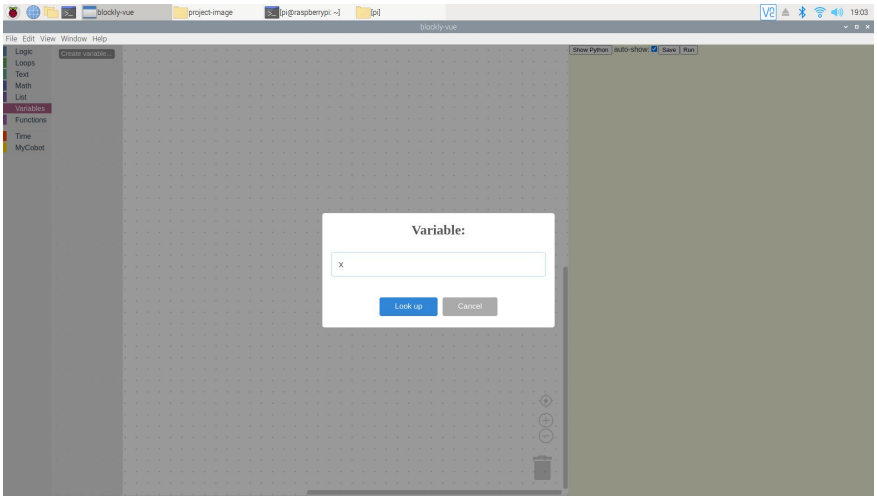
六、Variables模块

1、如图6-1所示即为Variables模块



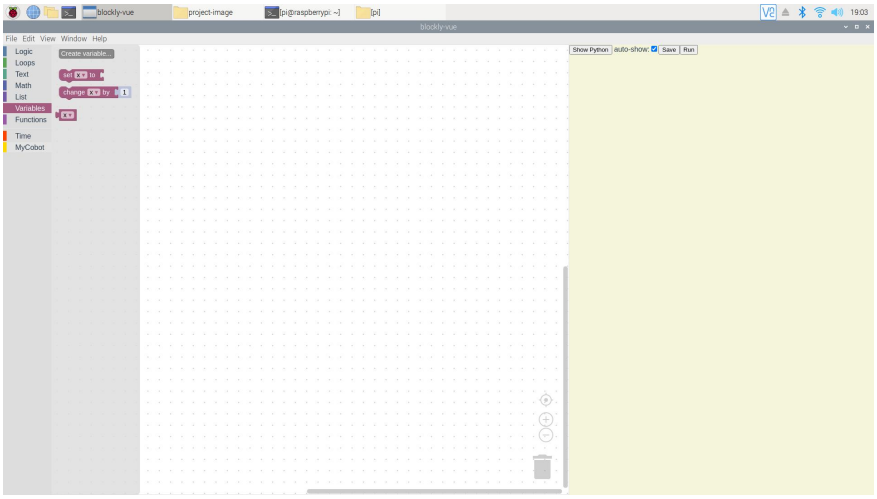
6-1 Variables模块

2、如图6-2所示，点击箭头所指处即可开始创建变量。



6-2 自定义变量名

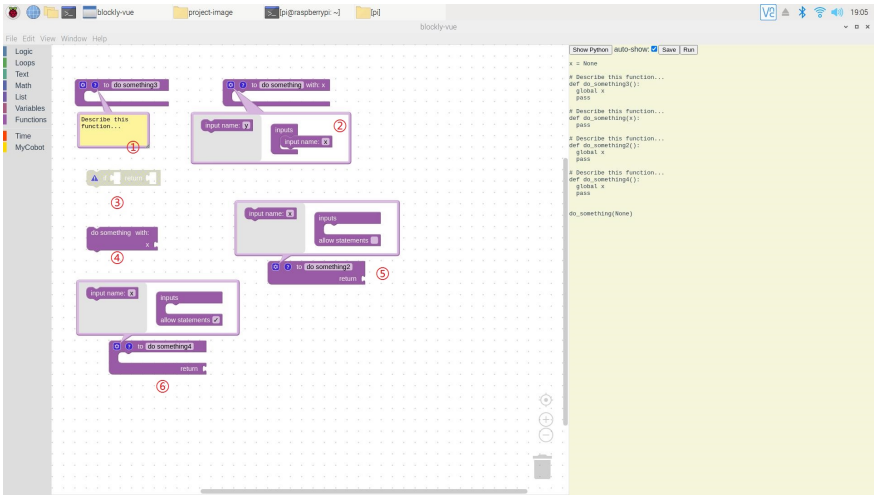
如图6-2所示，在输入框中输入自定义的变量名，点击Look up即可创建。



6-3 变量生成效果

如图6-3所示即为创建好后的变量。

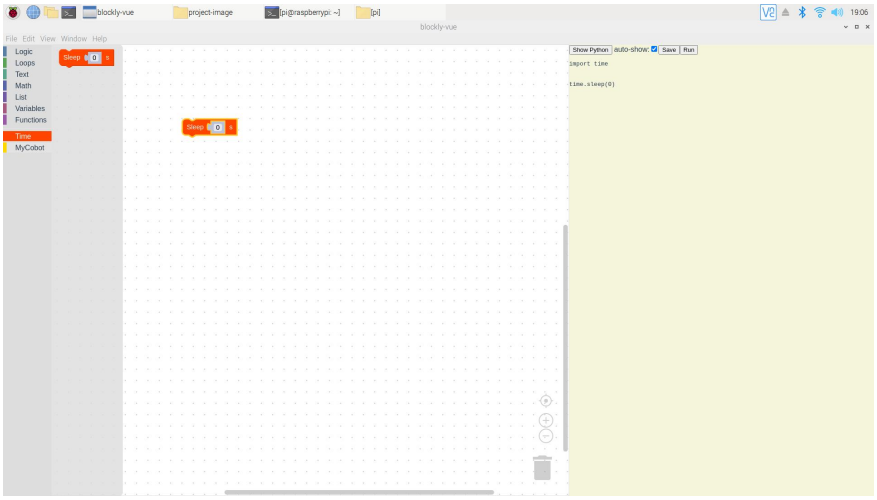
七、Functions模块



7-1 Functions模块

如图7-1所示，Functions模块包含两类函数，第一种如①所示是没有返回值的，第二种如⑥所示有返回值的。可以点击？进行设置函数的含义如①所示，也可以通过点击⚙添加函数参数。函数的调用也很简单，仅需将定义的函数拖拽出来即可，如④所示。③的作用即在函数中若满足if条件就终止函数。

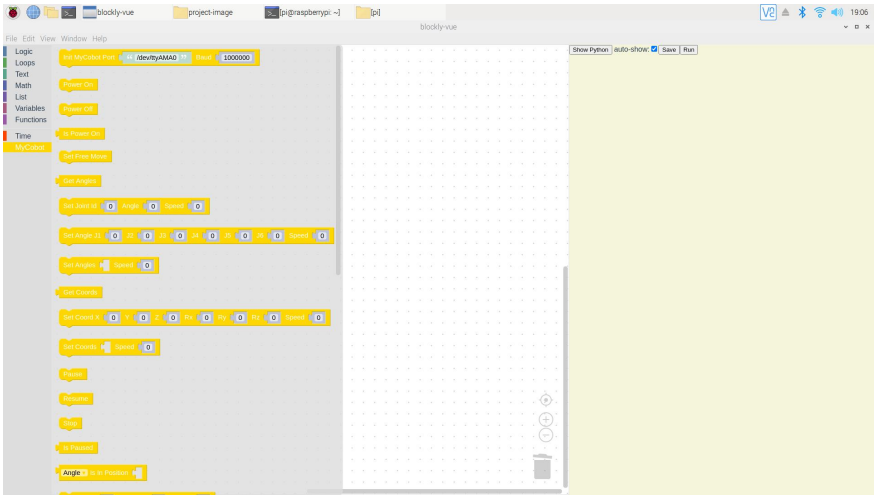
八、Time模块



8-1 时间模块

如图8-1所示，sleep（time_num）表示等待time_num秒后再执行接下来的程序。

九、Mycobot模块



9-1 Mycobot模块

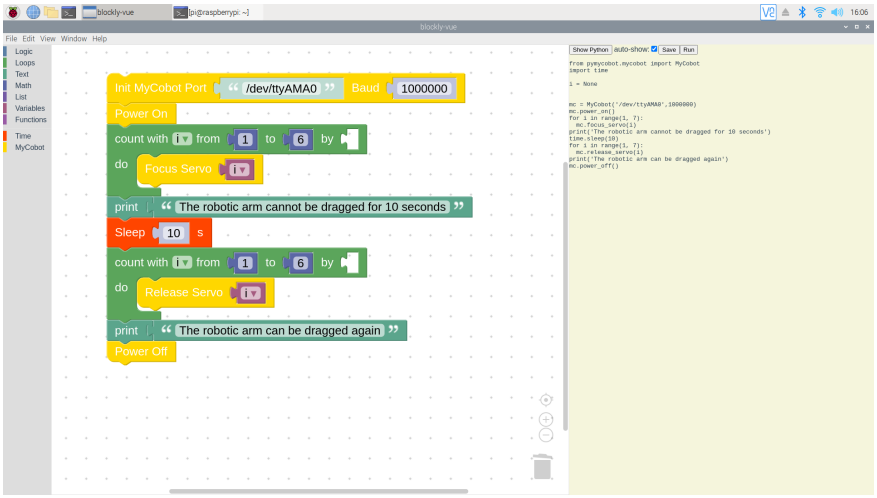
该模块的方法使用请参考**pymycobot**。

机械臂放松以及固定

一、案例介绍

本案里通过循环调用 `focus servo` 方法分别对六个关节点进行固定，并调用 `time` 方法让其固定十秒，最后通过循环调用 `release` 方法分别让六个关节点放松。

二、**demo**内容

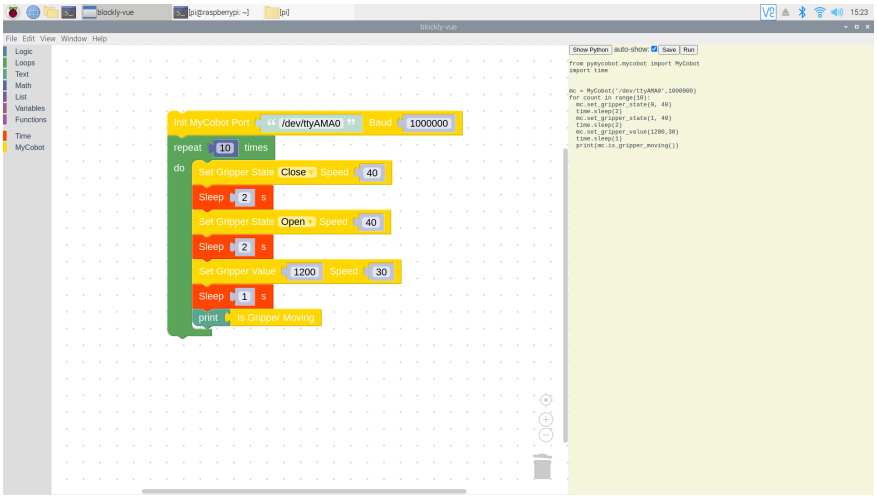


机械臂夹爪检测

一、案例介绍

通过调用mycobot的Set_Gripper_State函数分别让夹爪进行10次张开收拢操作，并且在每组张开收拢后对其进行角度调位。

二、**demo**内容

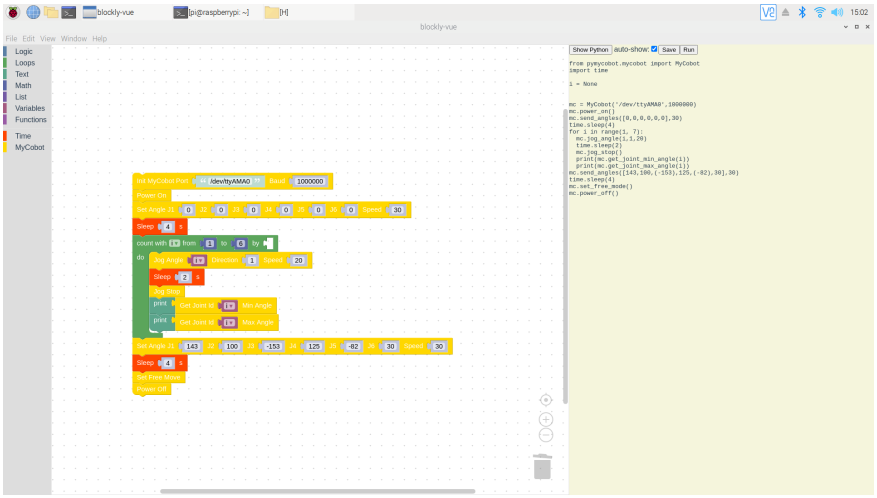


设置机械臂移动时间

一、案例内容

本案例主要的实验内容是调用 `jog_angle` 函数通过循环让六个关节分别持续移动，通过 `jog_stop` 函数对其运动进行停止。最后让机械臂移动到一个较为安全的位置并进行释放关节和断电操作。

二、程序内容

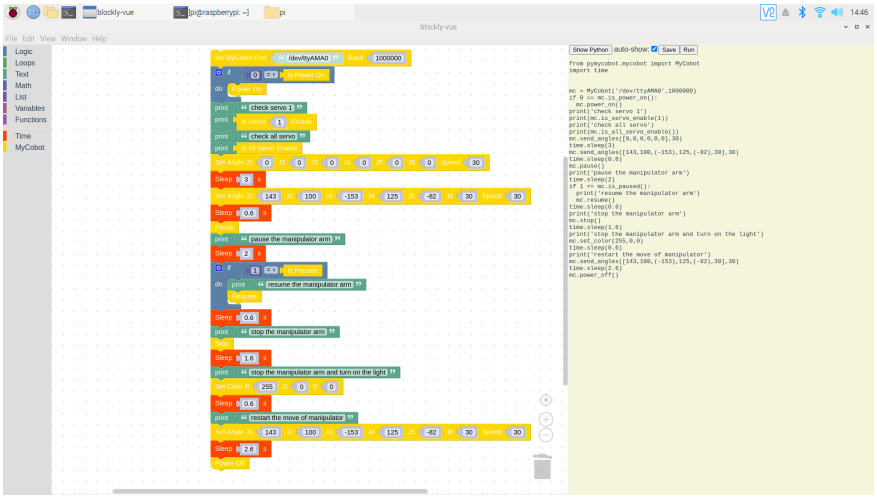


机械臂控制机制

一、案例内容

本案例主要是调用一些机械臂常用的控制机制函数进行机械臂控制，比如机械臂的断电、供电、暂停运动、恢复运动等控制机制功能以及机械臂头部灯的控制。

二、**demo** 内容



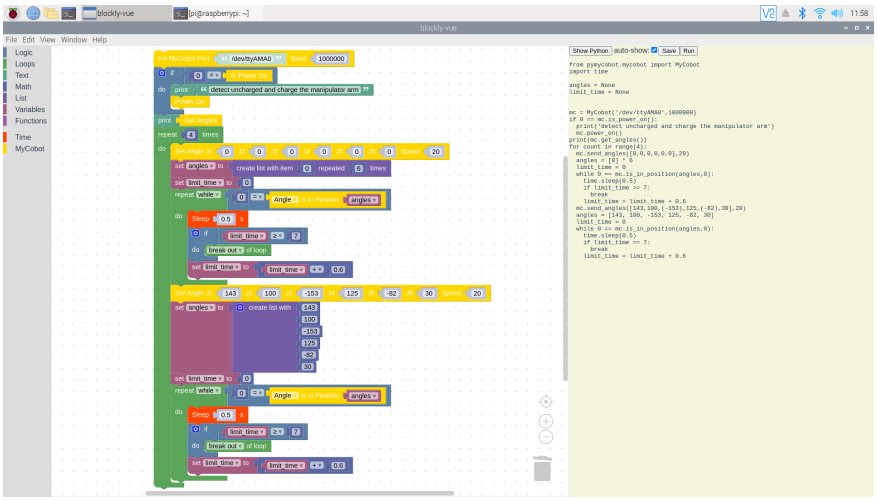
c 机械臂进阶操作

一、案例介绍

主要实现机械臂智能判断已经到达指定位置的功能，基于此功能，简单的让机械臂重复执行两个到达指令。

- 先用 `if do` 模块判断机械臂是否充电，若没有充电则为其充电。输出当前的角度节点信息，
- 让机械臂转移到零位。定义 `angles` 变量，使用创建 `list` 类型数据中的 `repeated` 方法为 `angles` 赋值零位节点信息。定义 `limit_time` 判断移动是否超时。
- 将 `angles` 传入 `is in position` 方法中判断机械臂是否达到指定位置。使用 `repeat` 模块每运动 0.5s 检测机械臂是否达到指定位置，并进行超时计时器处理，若超过 7s 机械臂仍未到达指定位置则断定其已经到达了，执行之后的指令。
- 接下来的原理大同小异故不再做解释。

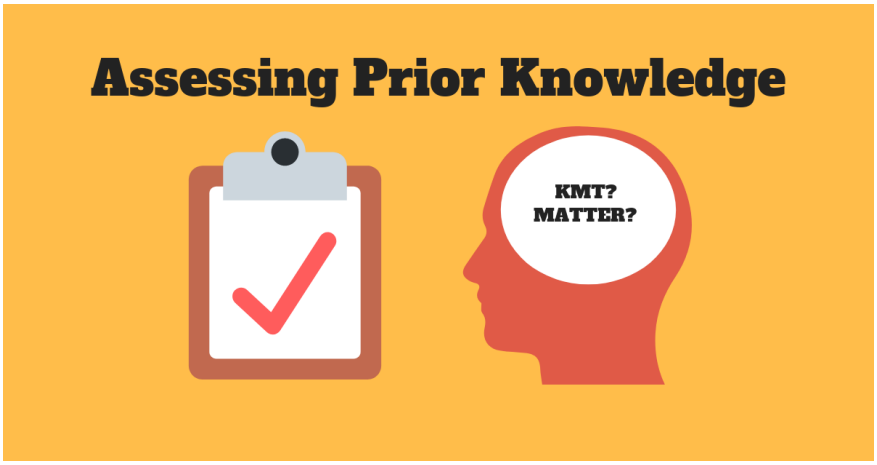
二、**demo** 内容



当你使用树莓派版本的 mycobot 的时候，应该已经拥有了一个装有 ROS Kinetic 的树莓派系统。

详细使用方法请参考第三章节的ROS 部分

1 背景知识



了解myCobot，需要从硬件、软件、机器人算法，几个方面分别进行深入的了解。在此之前，可以通过了解机器人的历史来了解机器人的进展历程。

- 机器人
- 软件
- 电子
- 力学
- 电机

制造变迁史：从手工到机器人

> 作者：大象机器人

2019年，全球制造业遭遇了重大危机。

美国制造业指数PMI创近10年新低，中国制造业指数也大幅放缓。向前蓬勃发展30年的中美贸易，在贸易战的影响下受到重创。

自十八世纪中叶开启工业文明以来，世界强国的兴衰史和中华民族的奋斗史一再证明：“得制造业者得天下”。

特朗普打着“振兴美国制造业”的旗号成功竞选总统，尽管从未被实践；最近，美国前总统奥巴马监制的以中国企业福耀玻璃为原型的纪录片《美国工厂》喧嚣日上，矛头也直指制造业。有人说，这是奥巴马在作为总统时未曾实现的美国梦的一个延续：把当政客时没有讲完的话，讲全乎了。

2019年，全球制造业遭遇了重大危机。

美国制造业指数PMI创近10年新低，中国制造业指数也大幅放缓。向前蓬勃发展30年的中美贸易，在贸易战的影响下受到重创。

自十八世纪中叶开启工业文明以来，世界强国的兴衰史和中华民族的奋斗史一再证明：“得制造业者得天下”。

特朗普打着“振兴美国制造业”的旗号成功竞选总统，尽管从未被实践；最近，美国前总统奥巴马监制的以中国企业福耀玻璃为原型的纪录片《美国工厂》喧嚣日上，矛头也直指制造业。有人说，这是奥巴马在作为总统时未曾实现的美国梦的一个延续：把当政客时没有讲完的话，讲全乎了。

制造业逃离美国，转向中国是不争的事实。影片里的福耀背道而驰，坚持打造美国的工厂也仅是为了照顾美国本土汽车行业而做的不得已之举，对现如今的制造业流向趋势并无多大贡献。

从工业革命至今，全球制造业经历了四次大迁移，制造业的重心，从英国，欧洲转向美国、日本，再到中国。目前全球制造业第五次迁移正在进行中，新一轮的制造中心开始流向东南亚的越南、印尼、印度等国家。

中国特色社会主义经济道路也难逃制造业的发展定律。随着我国人口红利的消失，传统制造业依靠人力发展的道路越走越窄。与此同时，以工业机器人为代表的智能装备，正为传统的装备制造以及物流等相关行业的生产方式带来了革命性的产业变革。

这样的转变，在关于制造的历史长河中也许仅是一个小小的片段。但对国家来说，每一次的改革创新，都是洪流下的巨变。

电影《2001：太空漫游》的一开始，一个史前猿人因为受到了黑石方碑的影响，学会了将骨头作为武器猎杀其他动物。与之相呼应的片尾，他将骨头抛入高空中，画面一转，诞生了一艘太空飞船。

从远古时代的石器到迄今尚在探索的宇宙飞船，这其中跨越260万年历史的变化发展，流淌的是整个人类文明制造发展史。

现如今我们所在的世界，眼之所在，皆是人类的发明制造；但这只是比较宽泛广义上的制造。真正意义的制造业，是机械工业时代对制造资源（物料、能源、设备、工具、资金、技术、信息和人力等），按照市场要求，通过制造过程，转化为可供人们使用和利用的大型工具、工业品与生活消费产品的行业。

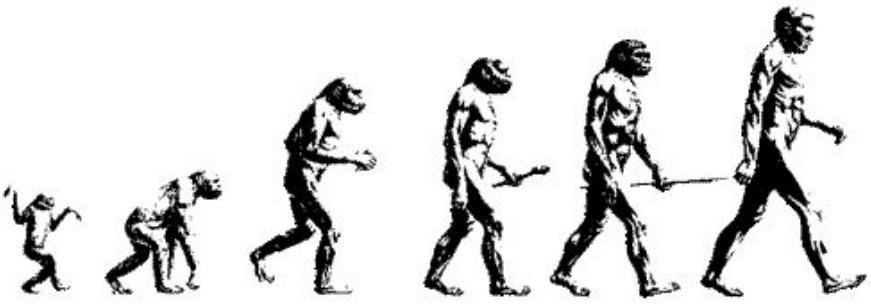
01. 黎明前的摸索

远古时期，世界的主体经济仍以游牧文明与农耕文明为主，制造业的胚芽尚在孕育当中。早期的制造是以制造工具更新换代的形式存在发展的。

迄今为止，最早被发现的石器工具是在肯尼亚发掘出来的，距今已有约260万年的历史。最原始的工具雏形出现在旧石器时代，工具形状大小各异，主要呈块状、薄片状和刀片状等。

同样是在肯尼亚，1981年，根据发掘的证据，人类控制火源的历史可以追溯到142万年前。

使用工具和能源，被认为是人类区别于其他物种的主要因素之一。在这个基础上，人类各种奇思妙想迸发，属于人类的文明生活就此拉开帷幕。



人类定居下来后科技发展的步伐大大加快，金属的出现成为人类科技文明发展的一次重大飞跃。

人类真正掌握铸造技术则是源于古埃及人发明的熔模铸造技术，这项技术通过向模型中注入熔化的金属，排出融化蜂蜡而成型。后来古埃及人还发明了金属锯，用以快速锯开木质的材料。

中国人则认为锯是鲁班在公元前5世纪发明。但中国人的确在公元前400年发明了高炉，用以炼制青铜。由其演变而来的炼铁工业制造了数以百万计的武器与衣物，成为秦始皇统一中国必不可少的工具。

人类生产工具变化比较表

所属时代	生产工具	制作材料	主要优、缺点
石器时代	石刀 石斧 石耜 木耒 骨耜 蚌镰	石、木、骨	取材容易、不坚固、石器成形困难
青铜器时代	青铜生产工具很少见	青铜	成形优于石器、轻巧锋利、硬度大、自然界蕴藏量少
铁器时代	铁耙 铁镰 铁锸 铁锄 铁犁铧	铁	自然界蕴藏量大、开采方便、质地坚硬、易氧化生锈

制造工具的推陈出新，得益于制作材料与工艺技术不断的发展创新。但提高制造效率的途径不仅仅是提高工艺与材料。

在工业革命开始前的600多年，威尼斯人首创了大规模生产模式——经过专业化的分工，将从事不同工种的劳动力分散在一条生产线上。生产线的诞生，大大提升了生产率和产量，实现了高品质武器的大规模生产。

这也意味着制造不再局限于制造材料与工艺技术，还包括了制造系统。与此同时，制造条件基本形成。

但在工业革命前的数千年间，在人们对能源与工具的使用无法进行指数型提升的背景下，现代人类重复的陷入马尔萨斯陷阱：人口增长超越食物供应，会导致人均占有食物的减少，最终弱者就会因此而饿死。

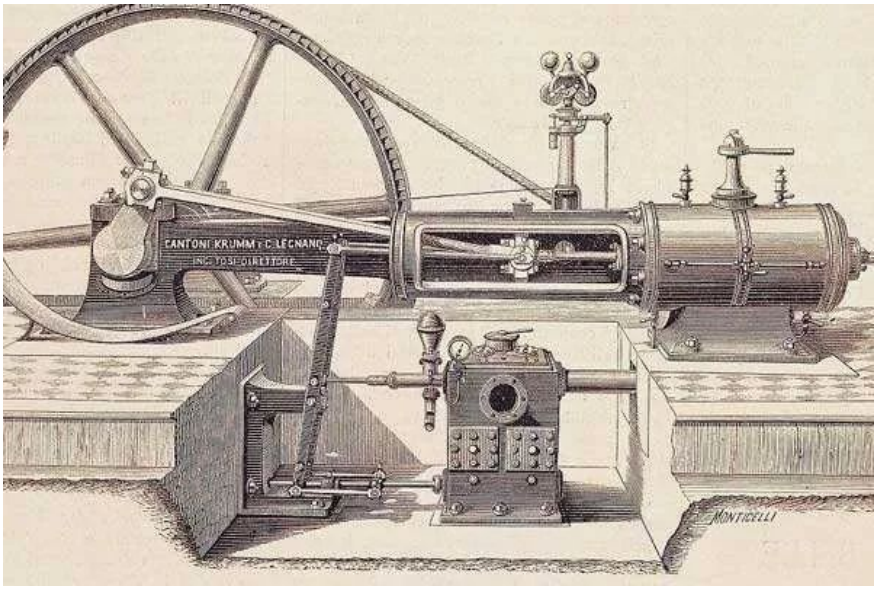
无休止的重复仿佛魔咒般无法逃离，黎明前的黑暗让人窒息，直到工业革命的来临。

02. “蒸汽朋克”的匠心

为了快速煮饭以及保留骨头的丰富营养和松软的口感，高压锅在1679年由法国科学家班平发明。高压锅的发明让他发现了一个有趣的现象：大气压力会影响沸点，蒸汽的力量可以抬起锅盖。

1698年，塞维利从班平的设计中得到灵感发明了抽水泵——一种最基本的蒸汽机，可以利用蒸汽动力缓解矿井排水问题。14年后的1712年，纽科门解决了塞维利抽水泵的距离缺陷，双方合作发明了常压蒸汽机。

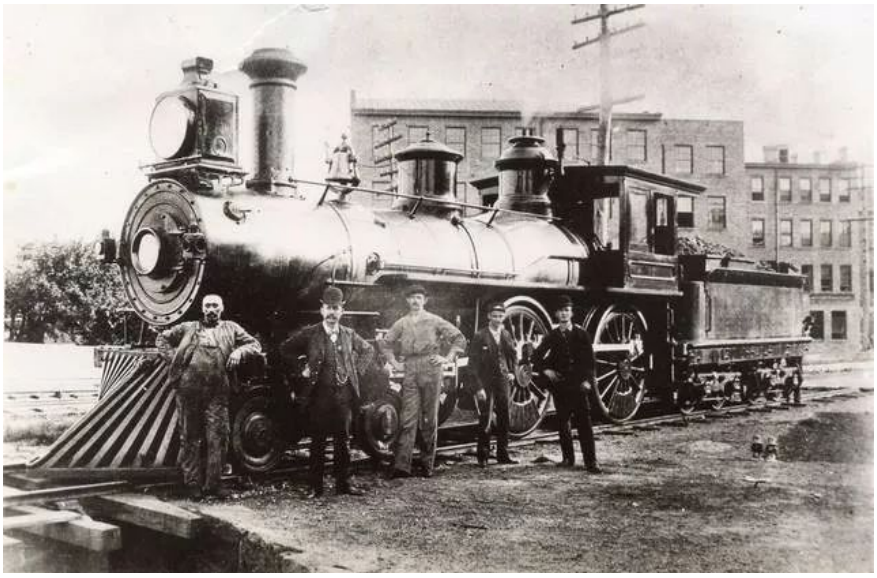
尽管常压蒸汽机气体冷凝速度慢，无法使蒸汽行程的活塞运动连续产生，效率较低；但由于焦炭冶铁技术的出现导致煤炭的需求急剧上升，它仍然受到了人们的热烈追捧。



常压蒸汽机

与此同时，“珍妮纺纱机”的出现首先在棉纺织业中引发了发明机器、革新技术的连锁反应，揭开了工业革命的序幕。

命运的转折点发生在1785年。瓦特发明了第一台带独立冷凝器的蒸汽机，体积小，耐久度高，可以持续、高效率的产生活塞运动从而提供旋转式的动能。改良型蒸汽机的投入使用极大推动了机器的普及和发展，人类社会由此进入“蒸汽时代”。



工业革命带来的科技创新以多种形式对制造产生着重大的影响。

工业生产中机器生产逐渐取代手工操作，一种新型的生产组织形式——工厂出现了。经济社会从农业、手工业为基础转型到了以工业及机械制造带动经济发展的模式，制造企业的模型初见端倪；企业形成作坊式的管理模式。

1776年3月，亚当·斯密的《国富论》中第一次提出将制造分解成若干具体的任务，将任务交由不同的人进行处理的生产，即“劳动分工”的科学定义。因为劳动分工对提高劳动生产率和增进国民财富的巨大作用，后来分工论逐渐演变成为企业管理的主要模式。

除此之外，在制造系统方面，1797年，惠特尼（Whitney）谈判达成了为美国政府生产10,000支步枪的合同，“可互换零件”与“配合装配”概念也因此在美国大批量传播推广。

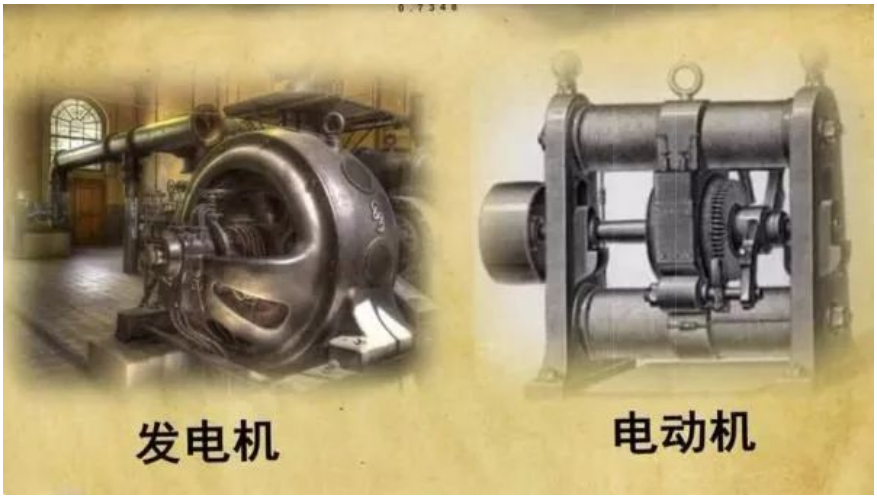
依靠工业革命完成的资本积累，社会结构也发生重大变革；新兴城市的产生以及世界工业化的进程都为制造业的蓬勃发展提供了肥沃的生长环境。

但此时的制造业还只是初具雏形，因为科学和技术尚只是在局部发明创造，尚未在制造业中真正结合起来，而相关的制造系统的概念也只是初步形成理论，没有经受实践的检验。

03. “电光火石”一触即发

当制造业不再局限在小型家庭作坊式的工厂，而是成长为大规模生产的工厂时，先进的生产技术和生产关系的重大变革起到了决定性作用，而这其中的关键在于科技与创新的发展。

第二次工业革命爆发期间，科学技术的复杂性和先进性达到了新的高度——自然科学同工业生产紧密结合起来。新的技术和发明远超过一国的范围，规模更加广泛，人们开始更多地尝试以前无法完成的大型设备。



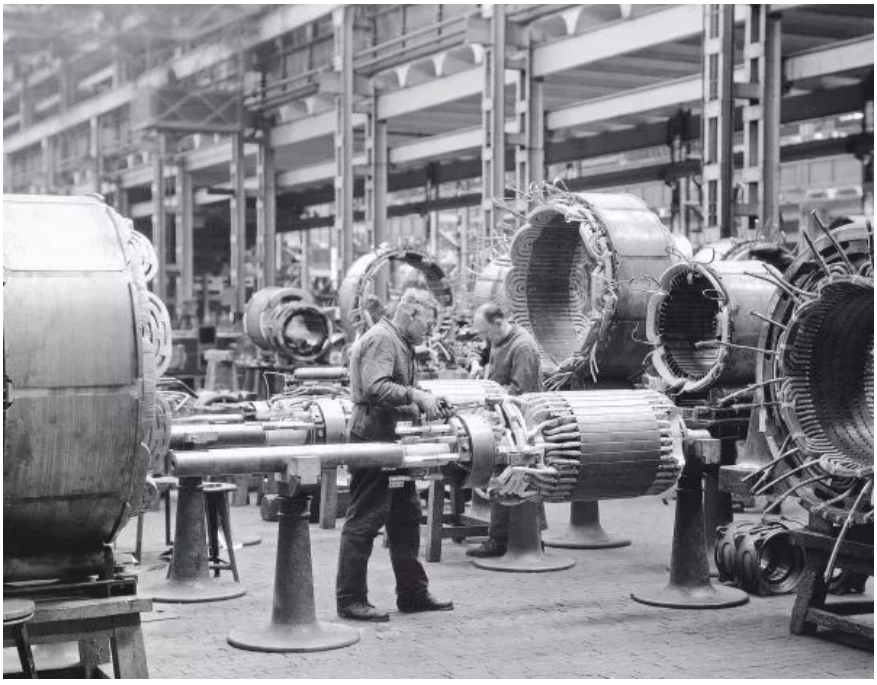
其中电力的广泛应用、内燃机和新交通工具的创制、新通讯手段的发明成为了当时科学技术最为突出的代表，制造业的发展一日千里。

兰索姆奥兹为了大批量生产汽车，发明了一个革新整个汽车行业的发明——装配线，装配线的发明将汽车产量从1901年的425辆提升到了1902年的2500辆。直到1913年，福特在装配线的概念基础上开创了流水线，大批量生产的模式进一步提高了汽车产量，大大降低了生产成本。

泰勒的科学管理理念对制造中的每个环节进行了科学化、理性的定义，其中包括

- 工人的工作任务时间、节拍标准；
- 行业标准的广泛使用；
- 计件制与劳动激励；
- 工厂中的数据搜集，用以替换工人以及成本核算。

新概念、新模式的出现导致制造技术的过细化分工和制造系统的功能分解，逐步形成了以科学管理为核心，推行标准化、流程化的管理模式，企业的人与“工作”得以匹配。



德国西门子生产车间

依靠工业革命蓬勃发展完成的资本积累，英法德美等国家率先进入现代化工业强国，形成西欧和北美两大工业地带。其中，重工业长足发展，逐步占据经济的主导地位。但当时整个世界的经济发展极其不均衡。

与此同时，制造业开始往劳动密集型产业和技术密集型产业两个方向分流。而这也成为当今全球制造经济四梯队的最初模型。

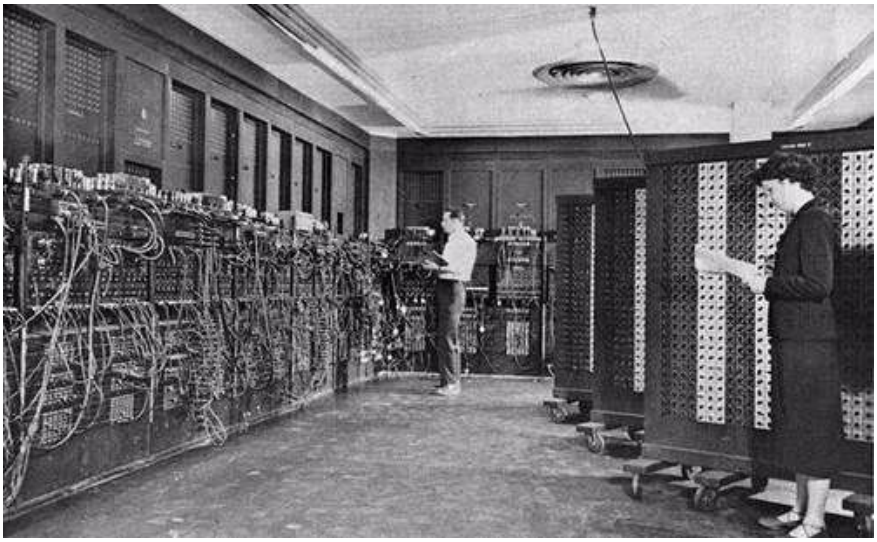
- 第一梯队：以美国为主导的全球科技创新中心；
- 第二梯队：高端制造领域，包括欧盟、日本；
- 第三梯队：中低端制造领域，主要是一些新兴国家，包括中国；
- 第四梯队：资源输出国，包括OPEC(石油输出国组织)、非洲、拉美等。

04. 计算机：灵魂的起舞

科技与创新始终是制造业发展与转型的关键所在。计算机的产生，给制造业注入了新的灵魂，制造业的发展取得了质的飞跃。

第一台数字式电子计算机于1941年由德国工程师康拉德楚泽发明。这是第一台全部由程序控制的计算机。

- 1963年，计算机辅助设计CAD开始允许用户使用一根“光电笔”对系统进行绘图。
- 1971年工作在雷诺公司的法国工程师贝塞尔发明了计算机辅助制造系统CAM，开始在电脑端与工厂机械设备之间建立虚拟联系，并实现控制。
- 1983年，康柏公司开发了第一台笔记本电脑。一体便携式的设计，尽可能简洁的IO设计大获成功。



世界第一台计算机

PLC于1960年诞生于美国通用公司，与计算机类似的是，它们都由软件控制，以实现固定指令；但PLC的编程更加简单，使用更加方便。在此阶段，工厂大量采用由PC、PLC/单片机等真正电子、信息技术自动化控制的机械设备进行生产。

由于当时的计算机体积较大，微型计算机与微型控制器于1971年由英特尔公司的特德霍夫发明而成。最初是为集中功能不同的计算器，随后又发明了英特尔4004微处理器。

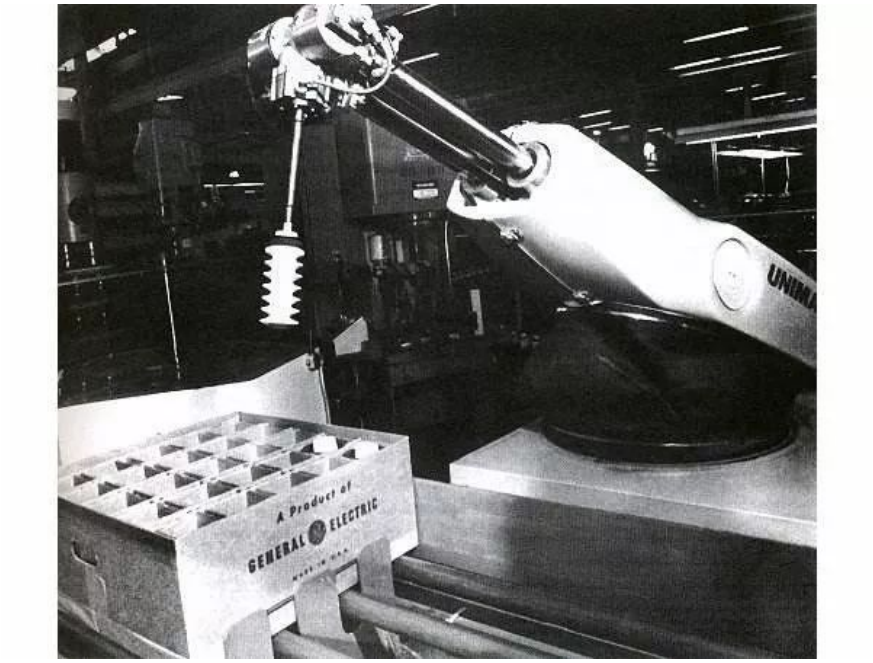
英特尔4004微处理器

微处理器的发展，极大程度地带动了各种制造工具的自动与智能化。自此，机器逐步替代人类作业，不仅接管了相当比例的“体力劳动”，还接管了一些“脑力劳动”。

制造工具也不单单是为工艺生产而服务的特种设备，更加通用、替代工人、服务于制造系统的制造工具应运而生。

机器人最早可追溯到1495年，奥纳多达芬奇制造了最早的机械骑士机器人；但现代第一台全自动机器人由美国神经学家沃特博士于1948年发明，这是一个类似于乌龟的小机器人，具有外部感光和触觉传感，在简单的电路帮助下可以自行运动。

直到20世纪50年代美国工程师乔治德沃尔与约瑟夫英格伯格提出了工业机器人，即工业机械臂的概念。在经历过若干样机试验后，英格伯格所在的Unimation公司生产了第一台名为Unimate的机械臂，该机械臂重达2吨，服务于通用汽车的装配线。当时仅实现将热压铸件在不同工位间的转移工作，随后几年开始进行车体的焊接工作。



Unimate机器臂

在升级工业2.0的基础上，广泛应用电子与信息技术进一步大幅度提高了制造过程中自动化的控制程度，生产效率、良品率、分工合作、机械设备寿命也得到了前所未有的提高；生产组织形式从工场化转变为现代大工厂，人类进入了产能过剩时代。

由于各国资源禀赋不同，全球制造业发展开始联通，制造业逐渐形成产业链分工不同的全球化模式。但随着工业霸主国生产资料成本的上升，全球制造业自20世纪以来经历了多次迁移，加工制造中心不断发生转移。

图表1：全球制造业大迁移发展历史

制造业转移	时间	输出地	输入地	转移方式
第一次	20世纪初	英国、德国	美国	受益于第二次工业革命的美国，当时国内工业发达程度已超过英国，而更充沛的劳动力（美国1亿人口；英国4000万人口），更广阔的地域面积（美国963万平方公里；英国24万平方公里）这些优势使美国的流水线批量、标准化生产得到高速发展。
第二次	20世纪50年代	美国	日本	“二战”后，日本在美国的扶持下，工业开始恢复。这一时期一些制造业开始向日本转移。与前两次转移不同，虽然制造业中心由美国转移向日本，但是科技中心和经济中心依然留在美国，社会分工进一步发展，全球产业链条开始形成，在美国的支持下，日本的制造业以年均13.2%的速度发展，这个速度是德国和法国的2倍，英国和美国的3倍。日本以高效完备的国家工业协作体系承接全球制造业转移，并在1968年成为国民生产总值（GDP）全球第二的经济强国。
第三次	20世纪60年代	日本	亚洲四小龙	随着日本经济的复苏，科技发展，人力成本上升，逐渐将附加值较低的纺织等加工行业转移向人力资源丰富，成本低廉的亚洲四小龙地区。韩国成为全球造船工业最发达国家；中国台湾地区成为全球最大的电子代工地区；中国香港地区制造业占比达到30%；新加坡已成为全球集成电路、芯片和磁盘的重要生产基地，同时也是世界第三大的炼油中心。
第四次	20世纪90年代	亚洲四小龙	中国大陆	亚洲四小龙地区逐步进入中高收入经济体，继续发展高端制造业，低端制造业向发展中国家（主要是中国）转移，80年代起中国大陆的工业总产值以每年15.3%的速度增长，同时台资和港资制造企业开始进入中国。中国制造业在2010年的GDP比重达40.1%，而1952年仅17.6%，有200多种商品产量居世界第一，钢、水泥、煤炭、家电、手机、计算机等行业的产量世界占比超50%，成为名副其实的“世界工厂”。
第五次	2008年至今	中国大陆	东南亚国家	随着中国人力成本逐渐升高，中国由“制造大国”向“制造强国”转变，劳动密集型产业逐步向人力成本更低的东南亚、南亚和非洲转移。

资料来源：前瞻产业研究院整理　　@前瞻经济学人APP

05. 智能时代，未来已来

人们常用最具代表性的生产工具来代表一个历史时期，如石器时代、青铜时代、铁器时代、蒸汽时代、电气时代、原子时代等；而21世纪最具代表性的生产工具非属互联网。

基于计算机的出现，因特网于1969年诞生。最初它还只是为了抵御前苏联的核威胁，快速传递信息与数据；到了1973年，为连接不同的计算机与计算设备，施乐公司开发了因特网，直到今日因特网仍在工厂的信息互联上承担重要角色；当然也远远不止如此。

互联网的协议（即TCP/IP），由罗伯特卡恩与文顿瑟夫发明，打通了电脑与电脑之间交流的最后一道坎；而极大扩大的互联网的容量与可能性将万物互联变为现实。

互联网、大数据、云计算、物联网等新技术与工业生产相结合，人类开始进入智能化时代。智能化时代下，制造业的生产组织形式从现代大工厂转变为虚实融合的工厂，柔性生产、个性化生产成为了时代的宠儿。

随着制造的智能化与互联化，机器人也有了长足的发展。

在2010年以前，大部分工业机器人都按照固定程序进行重复劳动，无法考虑到实际环境的变化，而且程序编写复杂，普通人无法使用。

1996年，由利诺州西北大学的教授J.Edward Colgate和Michael Peshkin提出的协作机器人概念，即机器人可以感知周围环境变化，更易使用与交互。

2008年，丹麦公司发明了第一台协作机器人UR5，可以实现碰撞停止，图形化编程等功能，实现了协作机器人的普及。

GGII数据显示，自2008 年以来，协作机器人行业从萌芽期进入快速发展期；截至2019年7月，全球协作机器人厂商数量已超过100家，其中进入中国市场的协作机器人厂商数量超过70家，“协作机器人”已经从概念到深入人心，并逐渐成为各机器人厂商争相布局的战略产品。

得益于其拥有较高的柔性、安全性和易操作性，协作机器人相较于传统工业机器人具有更广的应用延展性，不仅可以在工业领域应用，还可以在商业服务领域应用。

大象机器人Elephant Robotics

2019年，大象机器人开发了首款便携式的协作机器人Catbot；更轻便易用的设计以及欧美亚洲数十个国家的实例应用更加广泛地扩展协作机器人在更多领域发展的可能性。

古典经济学的奠基人亚当·斯密曾说过，"一切生产的最终目的都是满足人的需求"，智能化时代让人类重新认识世界、认识自己。

但智能制造的实现需要多个层次上技术产品的支持，其中包括工业机器人、3D打印、工业物联网、云计算、工业大数据、知识工作自动化、工业网络安全、虚拟现实和人工智能等。而这些技术产品中会产生无数的商机和上市公司。

“在历次的技术革命中，一个人、一家企业、甚至一个国家，可以选择的道路只有两条；要么加入浪潮，成为前2%的人，要么观望徘徊，被淘汰。”

智能制造站在全球范围内制造业发展趋势的风口，将带来新一轮生产方式、产业形态和商业模式之间的较量。发展企业智能制造依然成为必然的要求和趋势。

英国狄更斯在《双城记》曾说，“这是最好的时代，也是最坏的时代”。这句话不仅适用于第二次工业革命，也适用当下，这个正在发生重大变革的时代。

科学技术的发展深不可测，未来的制造业的发展我们尚不得知。在历史的洪流下，我们唯一做的，就是顺应时代的发展，在这场接力赛中，跑好属于自己的那一棒。

1.1 机器人知识

> 本章节摘选自《机器人学导论》J.Craig著。希望了解全文的朋友可以通过网络进行购买

1 机器人背景

工业自动化的历史是以技术手段的快速更新为特征的。这种自动化技术的更新不论是看作世界经济发展的诱因还是结果，都和世界经济密切相关。工业机器人在20世纪60年代毫无疑问是一种独特的设备＂，将其和计算机辅助设计（CAD)系统、计算机辅助制造（CAM系统结合在一起应用，这是现代制造业自动化的最新发展趋势。这些技术正在引导工业自动化向一个新的领域过渡。

操作臂是工业机器人中最重要的一种类型。操作臂是否可称作工业机器人受到争议。如图所示的装备通常被认为属于工业机器人的范畴，而数控（NC)磨床则通常在此范畴之外。

> 放置一个ABB机械臂

一般来说，对于操作臂的机构和控制理论的研究并不是一门新的科学，它只不过是对传统学科理论的一种综合。机械工程理论为研究静态和动态环境下的操作臂提供了方法论；数学方法用于描述机械手空间运动及其特性；控制理论为实现期望运动或力提供了各种设计方法和评估算法；电气工程技术可用于传感器及工业机器人接口的设计；计算机技术提供了执行期望任务所需的编程平台。

2 机器人基本概念描述

机械臂 机械臂也可以称之为工业机器人、协作机器人、操作臂、仿生手臂、串联手臂等。

位姿描述 在机器人研究中，我们通常在三维空间中研究物体的位置。这里所说的物体既包括操作臂的杆件、零部件和抓持工具，也包括操作臂工作空间内的其他物体。通常这些物体可用两个非常重要的特性来描述：位置和姿态。自然我们会首先研究如何用数学方法表示和计算这些参量。

为了描述空间物体的位姿，我们一般先将物体固置于一个空间坐标系，即参考系中，然后我们就在这个参考坐标系中研究空间物体的位置和姿态。

操作臂正运动学
运动学研究物体的运动，而不考虑引起这种运动的力。在运动学中，我们研究位置、速度、加速度和位置变量对于时间或者其他变量的高阶微分。这样，操作臂运动学的研究对象就是运动的全部几何和时间特性。几乎所有的操作臂都是由刚性连杆组成的，相邻连杆间由可作相对运动的关节连接。这些关节通常装有位置传感器，用来测量相邻杆件的相对位置。如果是转动关节，这个位移被称为关节角。一些操作臂含有滑动（或移动）关节，那么两个相邻连杆的位移是直线运动，有 时将这个位移称为**关节偏距**。
在操作臂运动学的研究中一个典型的问题是操作臂正运动学。计算操作臂末端执行器的位置和姿态是一个静态的几何问题。具体来讲，给定一组关节角的值，正运动学问题是计算工具坐标系相对于基坐标系的位置和姿态。一般情况下，我们将这个过程称为从关节空间描述到笛卡儿空间描述的操作臂位置表示。

自由度

的数目是操作臂中具有独立图1-5 在坐标系（参考系）中的操作臂位置变量的数目，这些位置变量确定了机构中所有部件的位置。自由度对所有的机构具有普遍意义。例如，四杆机构只有一个自由度（尽管它有三个可以运动的杆件）。对于一个典型的工业机器人来讲，由于操作臂大都是开式的运动链，而且每个关节位置都由一个独立的变量 定义，因此关节数目等于自由度数目。

末端执行器

安装在操作臂的自由端。根据机器人的不同应用场合，末端执行器可以是一个夹具、一个焊枪、一个电磁铁或是其他装置。我们通常用附着于末端执行器上的工具坐标系描述操作臂的位置，与工具坐标系相对应的是与操作臂固定底座相联的**基坐标系**。

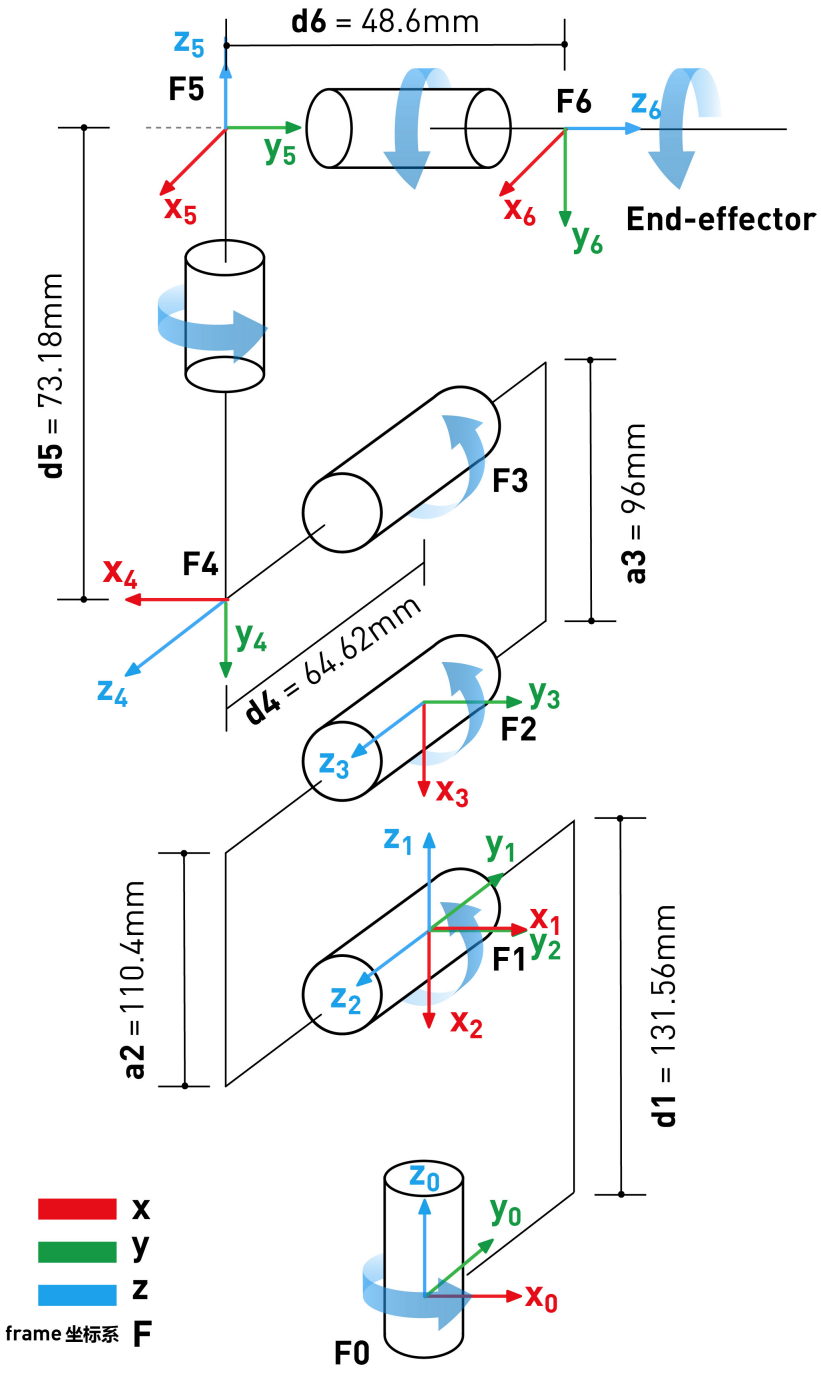
操作臂逆运动学

给定操作臂末端执行器的位置和姿态，计算所有可达给定位置和姿态的关节角。

3 空间描述

位置

一旦建立了坐标系，我们就能用一个3x1的位置矢量对世界坐标系中的任何点进行定位。因为经常在世界坐标系中还要定义许多坐标系，因此必须在位置矢量上附加一信息，表明是在哪一个坐标系被定义的。在本书，位置矢量用一个前置的上标来表明其参考的坐标系。



用三个相互正交的带有箭头的单位矢量来表示一个坐标系｛A}.用一个矢量来表示一个点AP,并且可等价地被认为是空间的一个位置，或者简单地用一组有序的三个数字来表示。矢量图2-1 相对于坐标系的矢量（示例）的各个元素用下标x,y和z来标明：

姿态

我们发现不仅经常需要表示空间的点，还经常需要描述空间中物体的姿态。例如，如果在图2-2中矢量“P直接确定了在操作手指端之间的某点，只有当手的姿态已知后，手的位置才能完全被确定下来。假定操作手有足够数量的关节，则操作手可有

任意的姿态，而该点在指端之间的位置可保持不变。为了描述物体的姿态，我们将在物体上固定一个坐标系并且给出此坐标系相对于参考系的表达。在图2-2中，已知坐标系｛B}以某种方式固定在物体上。｛B}相对于｛A}中的描述就足以表示出物体（A)的姿态。

坐标系 一个参考系可以用一个坐标系相对于另一坐标系的关系来描述。参考系包括了位置和姿态两个概念，大多数情况下被认为是这两个概念的结合。位置可由一个参考系表示，这个参考系中的旋转矩阵是单位阵，并且这个参考系中的位置矢量确定了被描述点的位置。同样，如果参考系中的位置矢量是零矢量，那么它表示的就是姿态。

4 DH参数

定义 对于转动关节n,设定0=0.0,此时X轴与X,轴的方向相同，选取坐标系（N)的原点位置使之满足d.=0.0.对于移动关节n,设定8轴的方向使之满足0.=0.0.当d.=0.0时，选取坐标系｛N)的原点位于XN-1轴与关节轴n的交点位置。

在连杆坐标系中对连杆参数的归纳如果按照上述规定将连杆坐标系固连于连杆上时，连杆参数可以定义如下：

- a_i-1 ：沿着x_i-1 : 从z_i-1 到 z_i 的距离
- alpha_i-1 ：绕着x_i-1 ：从z_i-1 到 z_i 的角度
- d_i ：沿着z_i ：从x_i-1 到 x_i 的距离
- theta_i ：绕着z_i ：从x_i-1 到x_i 的角度

这里有一篇文章可以 https://blog.csdn.net/hitgavin/article/details/104442034

myCobot DH参数

Joint	alpha	a	d	theta	offset
1	0	0	131.56	theta_1	0
2	PI/2	0	0	theta_2	-PI/2
3	0	-110.4	0	theta_3	0
4	0	-96	64.62	theta_4	-PI/2
5	PI/2	0	73.18	theta_5	PI/2
6	-PI/2	0	48.6	theta_6	0

1.2 软件背景知识

myCobot的软件基础需要基本的C/C++知识，以驱动myCobot的微控制器Basic和Atom。

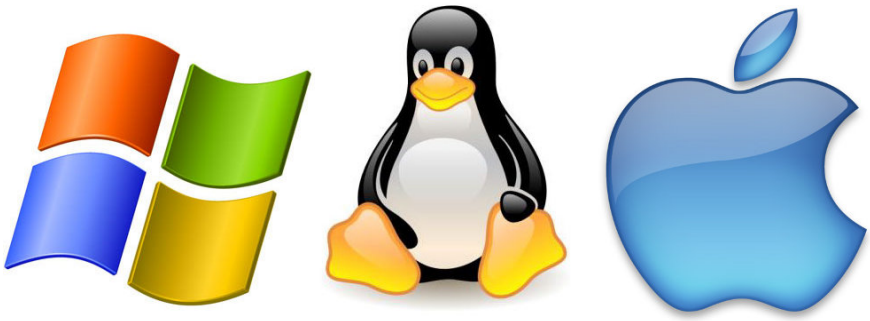
同时需要的相关软件主要有github和arduino两个主要软件，其中

- **Github** : 用以下载最新的myCobot代码和更新相关使用规则
- **Arduino** : myCobot核心开发的IDE（集成开发环境），需要用C/C++进行编程。

当然，您也可以直接使用**python**, **ROS**, **uiFlow**等其他开发手段进行开发，这个会在**开发与使用**章节中进行介绍。

如果您想要学习编程语言，可以通过 **书籍**、**公开课**、**网络视频**等进行学习。

值得一提的是，您的开发平台可以是 Windows, MacOS 或者 Linux。

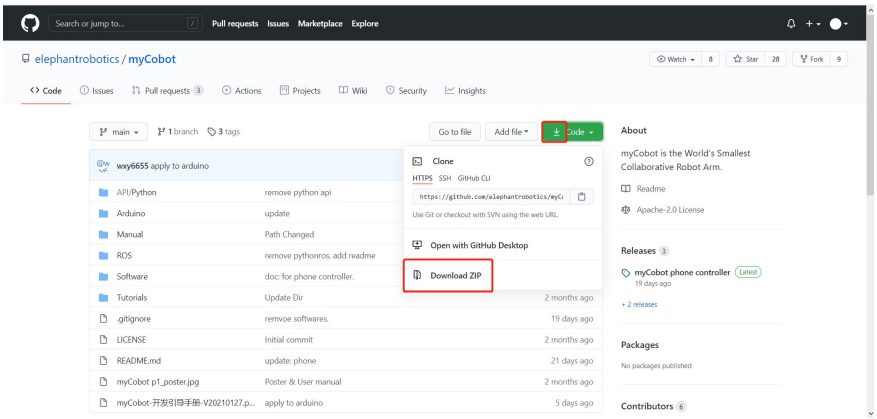


Github

GitHub是一个面向开源及私有软件项目的托管平台，上面存放着我们关于myCobot机械臂的各种软件（python、ros、arduino）、手机app、工业可视化编程软件RoboFlow、烧录固件以及用户手册和开发引导手册。

Github地址：https://github.com/elephantrobotics/myCobot

如图所示点击下载



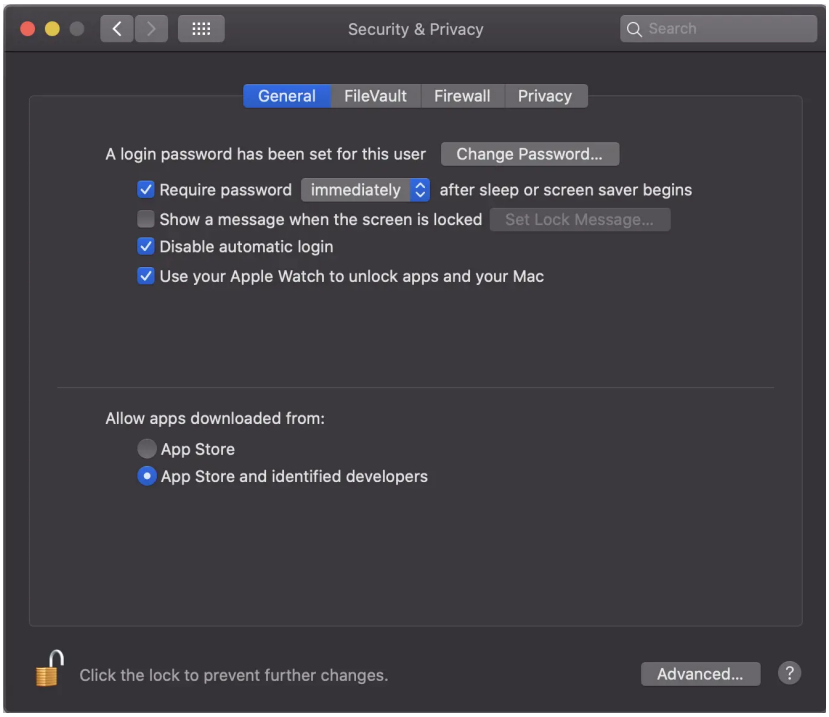
Arduino

Arduino是什么？

Arduino 是一款简单易用、便于上手的开源电子原型平台，包含硬件部分（各种符合 Arduino 规范的开发板）和软件部分（Arduino IDE和相关的开发包）。 硬件部分（或称开发板）由微控制器（MCU）、闪存（Flash）以及一组通用输入/输出接口（GPIO）等构成，你可以将它理解为是一块微型电脑主板。 软件部分则主要由PC端的 Arduino IDE 以及相关的板级支持包（BSP）和丰富的第三方函数库组成。使用者可以借由 Arduino IDE 轻松地下载你所持有的开发板相关的BSP和需要的函数库，用于编写你的程序。

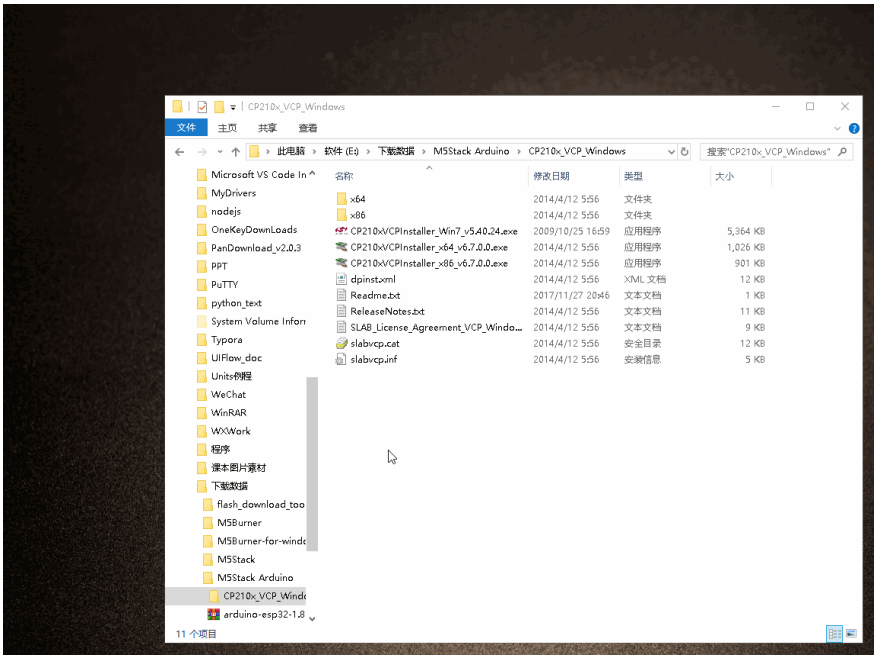
如何安装Arduino IDE？

- 驱动安装
  程序烧录前，M5Core型主机（包含BASIC/GRAY/M5GO/FIRE/FACES）用户请根据您使用的操作系统，点击下方按钮下载相应的CP210X驱动程序压缩包.在解压压缩包后，选择对应操作系统位数的安装包进行安装。
  对于Mac OS，在安装之前确保系统偏好设置->安全性和隐私->通用，并允许从App Store和被认可的开发者



下载CP2104驱动程序

- Windows10
- MacOS
- Linux



Arduino IDE

如果需要下载Arduino IDE 可以点击Aeduino 官网下载安装与电脑系统对应的版本

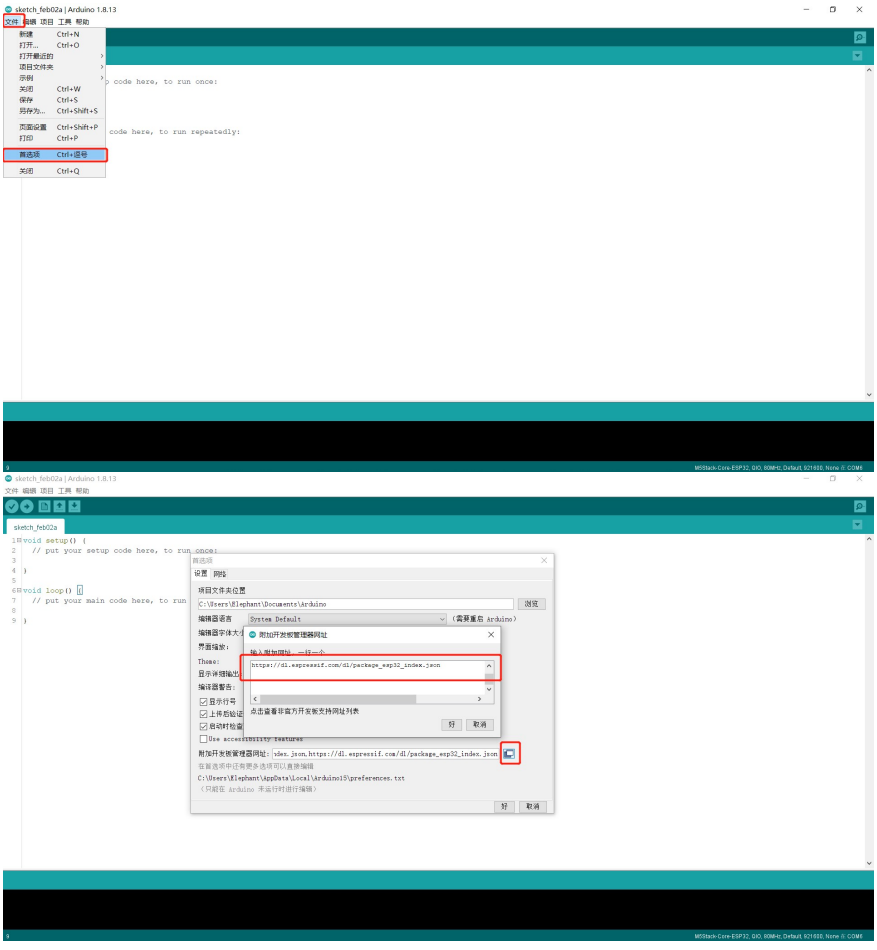
- Windows X64
- Mac OS X
- Linux ARM 64

配置MyCobot所需的Arduino开发环境

1. 添加需要的开发板

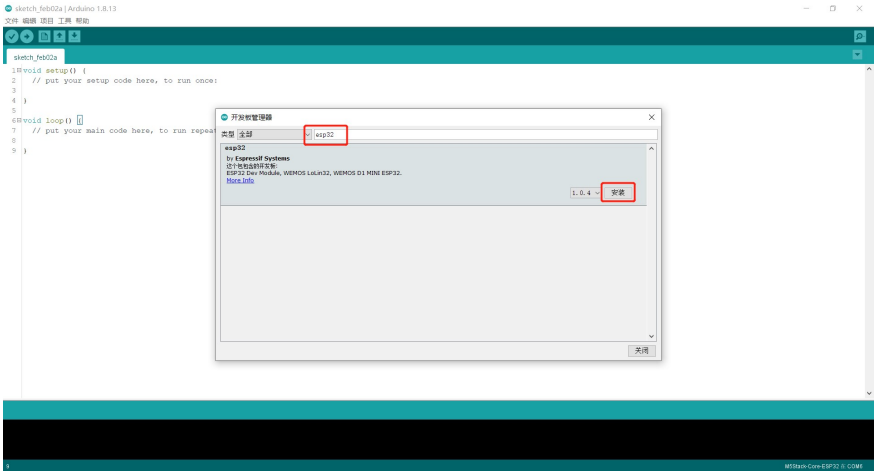
- 打开Arduino IDE,选择文件->首选项->设置，复制下方的网址链接到 附加开发板管理器中

```
https://dl.espressif.com/dl/package_esp32_index.json
https://m5stack.oss-cn-shenzhen.aliyuncs.com/resource/arduino/package_m5stack_ind
```

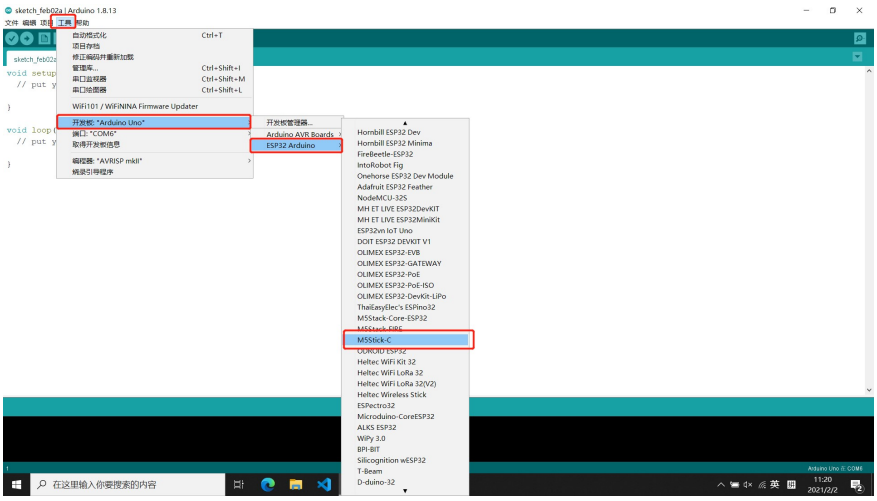


- 选择工具->开发版：->开发板管理器

在新弹出的对话框中，输入并搜索 M5Stack/esp32，点击安装（若出现搜索失败的情况，可以尝试重启Arduino程序），如下图：



- 添加后选择 工具->开发板：查看是否成功，如下图：



2. 添加相关库

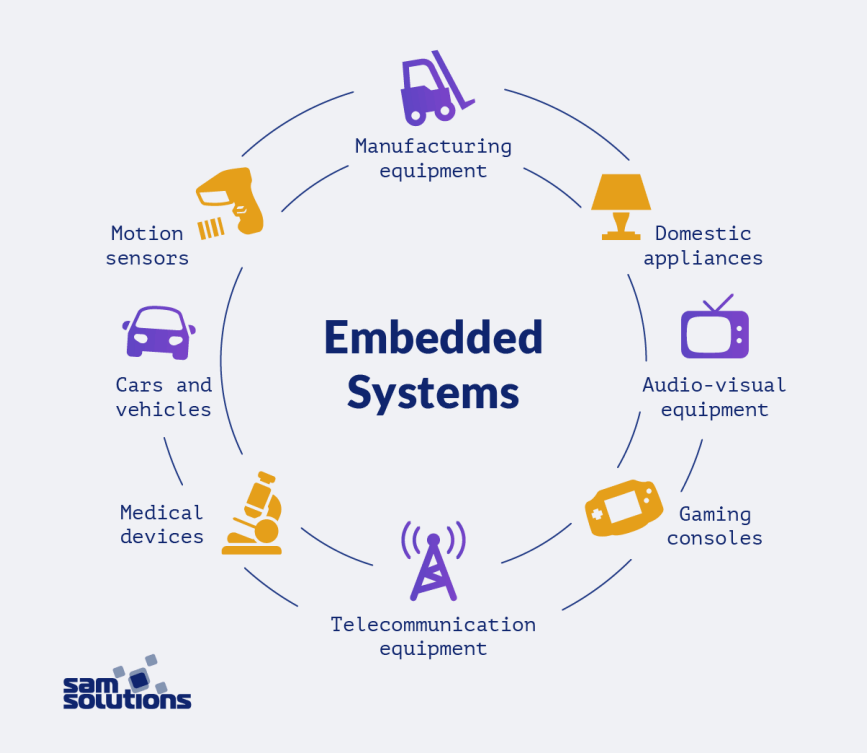
- 点击下载相关依赖库

  GIT链接 | 查看API | 下载依赖Library

- 依赖库安装说明

  - 依赖库解压后即可使用，仅需要添加开发板即可，添加方式请看上方安装说明
  - 解压到 "C:\Users\User\Documents\Arduino" 文件夹中即可，此文件路径为默认路径，如果你正在使用Arduino，请不要覆盖，添加至已有的Library中即可。

单片机与微控制器



简介

单片机是一种集成电路芯片，是采用超大规模集成电路技术把具有数据处理能力的中央处理器CPU、随机存储器RAM、只读存储器ROM、多种I/O口和中断系统、定时器/计数器等功能（可能还包括显示驱动电路、脉宽调制电路、模拟多路转换器、A/D转换器等电路）集成到一块硅片上构成的一个小而完善的微型计算机系统，在工业控制领域广泛应用。从上世纪80年代，由当时的4位、8位单片机，发展到现在的300M的高速单片机。

基本结构

运算器 运算器由运算部件——算术逻辑单元（Arithmetic & Logical Unit，简称ALU）、累加器和寄存器等几部分组成。ALU的作用是把传来的数据进行算术或逻辑运算，输入来源为两个8位数据，分别来自累加器和数据寄存器。ALU能完成对这两个数据进行加、减、与、或、比较大小等操作，最后将结果存入累加器。
运算器有两个功能：

- 执行各种算术运算。
- 执行各种逻辑运算，并进行逻辑测试，如零值测试或两个值的比较。
  运算器所执行全部操作都是由控制器发出的控制信号来指挥的，并且，一个算术操作产生一个运算结果，一个逻辑操作产生一个判决。

控制器 控制器由程序计数器、指令寄存器、指令译码器、时序发生器和操作控制器等组成，是发布命令的“决策机构”，即协调和指挥整个微机系统的操作。其主要功能有：

- 从内存中取出一条指令，并指出下一条指令在内存中的位置。
- 对指令进行译码和测试，并产生相应的操作控制信号，以便于执行规定的动作。
- 指挥并控制CPU、内存和输入输出设备之间数据流动的方向。

微处理器内通过内部总线把ALU、计数器、寄存器和控制部分互联，并通过外部总线与外部的存储器、输入输出接口电路联接。外部总线又称为系统总线，分为数据总线DB、地址总线AB和控制总线CB。通过输入输出接口电路，实现与各种外围设备连接。

主要寄存器

- 累加器A
  累加器A是微处理器中使用最频繁的寄存器。在算术和逻辑运算时它有双功能：运算前，用于保存一个操作数；运算后，用于保存所得的和、差或逻辑运算结果。

- 数据寄存器DR
  数据寄存器通过数据总线向存储器和输入/输出设备送（写）或取（读）数据的暂存单元。它可以保存一条正在译码的指令，也可以保存正在送往存储器中存储的一个数据字节等等。

- 指令寄存器IR和指令译码器ID
  指令包括操作码和操作数。
  指令寄存器是用来保存当前正在执行的一条指令。当执行一条指令时，先把它从内存中取到数据寄存器中，然后再传送到指令寄存器。当系统执行给定的指令时，必须对操作码进行译码，以确定所要求的操作，指令译码器就是负责这项工作的。其中，指令寄存器中操作码字段的输出就是指令译码器的输入。

- 程序计数器PC
  PC用于确定下一条指令的地址，以保证程序能够连续地执行下去，因此通常又被称为指令地址计数器。在程序开始执行前必须将程序的第一条指令的内存单元地址（即程序的首地址）送入PC，使它总是指向下一条要执行指令的地址。

- 地址寄存器AR
  地址寄存器用于保存当前CPU所要访问的内存单元或I/O设备的地址。由于内存与CPU之间存在着速度上的差异，所以必须使用地址寄存器来保持地址信

息，直到内存读/写操作完成为止。

显然，当CPU向存储器存数据、CPU从内存取数据和CPU从内存读出指令时，都要用到地址寄存器和数据寄存器。同样，如果把外围设备的地址作为内存地址单元来看的话，那么当CPU和外围设备交换信息时，也需要用到地址寄存器和数据寄存器。

硬件特征

单片机的体积比较小， 内部芯片作为计算机系统，其结构简单，但是功能完善，使用起来十分方便，可以模块化应用。

- 单片机有着较高的集成度，可靠性比较强，即使单片机处于长时间的工作也不会存在故障问题。
- 单片机在应用时低电压、低能耗，是人们在日常生活中的首要选择， 为生产与研发提供便利。
- 单片机对数据的处理能力和运算能力较强，可以在各种环境中应用，且有着较强的控制能力

单片机技术的开发

单片机在电子技术中的开发，主要包括CPU开发、程序开发、 存储器开发、计算机开发及C语言程序开发，同时得到开发能够保证单片机在十分复杂的计算机与控制环境中可以正常有序的进行，这就需要相关人员采取一定的措施：

- CPU开发。开发单片机中的CPU总线宽度，能够有效完善单片机信息处理功能缓慢的问题，提高信息处理效率与速度，开发改进中央处理器的实际结构，能够做到同时运行2-3个CPU，从而大大提高单片机的整体性能。
- 程序开发。嵌入式系统的合理应用得到了大力推广，对程序进行开发时要求能够自动执行各种指令，这样可以快速准确地采集外部数据，提高单片机的应用效率。
- 存储器开发。单片机的发展应着眼于内存，加强对基于传统内存读写功能的新内存的探索，使其既能实现静态读写又能实现动态读写，从而显着提高存储性能。
- 计算机开发。进一步优化和开发单机片应激即分析，并应用计算机系统，通过连接通信数据，实现数据传递。
- C语言程序开发。优化开发C语言能够保证单片机在十分复杂的计算机与控制环境中，可以正常有序的进行，促使其实现广泛全面的应用。

1.4 力学背景知识

待开发。。。

电机与舵机

电机是什么？

- **1 按工作电源种类划分**：可分为直流电机和交流电机。
  - 1）直流电动机按结构及工作原理可划分：无刷直流电动机和有刷直流电动机。
    - 有刷直流电动机可划分：永磁直流电动机和电磁直流电动机。
    - 电磁直流电动机划分：串励直流电动机、并励直流电动机、他励直流电动机和复励直流电动机。
    - 永磁直流电动机划分：稀土永磁直流电动机、铁氧体永磁直流电动机和铝镍钴永磁直流电动机。
  - 2）其中交流电机还可划分：单相电机和三相电机。

- **2 按用途可划分**：驱动用电动机和控制用电动机。
  - 1）驱动用电动机可划分：电动工具（包括钻孔、抛光、磨光、开槽、切割、扩孔等工具）用电动机、家电（包括洗衣机、电风扇、电冰箱、空调器、录音机、录像机、影碟机、吸尘器、照相机、电吹风、电动剃须刀等）用电动机及其他通用小型机械设备（包括各种小型机床、小型机械、医疗器械、电子仪器等）用电动机。
  - 2）控制用电动机又划分：步进电动机和伺服电动机等。

舵机是什么？

舵机其实就是伺服电机，因为国内的用户已开始使用在航模等设备的舵轴控制，所以国内对这些轻量级的伺服电机称之为舵机。

什么是伺服电机

伺服电机（servo motor ）是指在伺服系统中控制机械元件运转的发动机，是一种补助马达间接变速装置。
伺服电机可使控制速度，位置精度非常准确，可以将电压信号转化为转矩和转速以驱动控制对象。伺服电机转子转速受输入信号控制，并能快速反应，在自动控制系统中，用作执行元件，且具有机电时间常数小、线性度高等特性，可把所收到的电信号转换成电动机轴上的角位移或角速度输出。分为直流和交流伺服电动机两大类，其主要特点是，当信号电压为零时无自转现象，转速随着转矩的增加而匀速下降。

伺服电机的工作原理

- **1.伺服系统（servo mechanism）**是使物体的位置、方位、状态等输出被控量能够跟随输入目标（或给定值）的任意变化的自动控制系统。伺服主要靠脉冲来定位，基本上可以这样理解，伺服电机接收到1个脉冲，就会旋转1个脉冲对应的角度，从而实现位移，因为，伺服电机本身具备发出脉冲的功能，所以伺服电机每旋转一个角度，都会发出对应数量的脉冲，这样，和伺服电机接受的脉冲形成了呼应，或者叫闭环，如此一来，系统就会知道发了多少脉冲给伺服电机，同时又收了多少脉冲回来，这样，就能够很精确的控制电机的转动，从而实现精确的定位，可以达到0.001mm。直流伺服电机分为有刷和无刷电机。有刷电机成本低，结构简单，启动转矩大，调速范围宽，控制容易，需要维护，但维护不方便（换碳刷），产生电磁干扰，对环境有要求。因此它可以用于对成本敏感的普通工业和民用场合。
  无刷电机体积小，重量轻，出力大，响应快，速度高，惯量小，转动平滑，力矩稳定。控制复杂，容易实现智能化，其电子换相方式灵活，可以方波换相或正弦波换相。电机免维护，效率很高，运行温度低，电磁辐射很小，长寿命，可用于各种环境。

- **2.交流伺服电机也是无刷电机**，分为同步和异步电机，运动控制中一般都用同步电机，它的功率范围大，可以做到很大的功率。大惯量，最高转动速度低，且随着功率增大而快速降低。因而适合做低速平稳运行的应用。

- **3.伺服电机内部的转子是永磁铁**，驱动器控制的U/V/W三相电形成电磁场，转子在此磁场的作用下转动，同时电机自带的编码器反馈信号给驱动器，驱动器根据反馈值与目标值进行比较，调整转子转动的角度。伺服电机的精度决定于编码器的精度（线数）。
  交流伺服电机和无刷直流伺服电机在功能上的区别：交流伺服要好一些，因为是正弦波控制，转矩脉动小。直流伺服是梯形波。但直流伺服比较简单，便宜。

工业用伺服电机

伺服电机的特点对比

直流无刷伺服电机特点 转动惯量小、启动电压低、空载电流小； 弃接触式换向系统，大大提高电机转速，最高转速高达100 000rpm；无刷伺服电机在执行伺服控制时，无须编码器也可实现速度、位置、扭矩等的控制； 不存在电刷磨损情况，除转速高之外，还具有寿命长、噪音低、无电磁干扰等特点。 直流有刷伺服电机特点

- 1.体积小、动作快反应快、过载能力大、调速范围宽
- 2.低速力矩大, 波动小，运行平稳
- 3.低噪音,高效率
- 4.后端编码器反馈（选配）构成直流伺服等优点
- 5.变压范围大，频率可调

伺服电机对于普通电机的优点

- 1、精度：实现了位置，速度和力矩的闭环控制；克服了步进电机失步的问题；
- 2、转速：高速性能好，一般额定转速能达到2000～3000转；
- 3、适应性：抗过载能力强，能承受三倍于额定转矩的负载，对有瞬间负载波动和要求快速起动的场合特别适用；
- 4、稳定：低速运行平稳，低速运行时不会产生类似于步进电机的步进运行现象。适用于有高速响应要求的场合；
- 5、及时性：电机加减速的动态相应时间短，一般在几十毫秒之内；
- 6、舒适性：发热和噪音明显降低。

简单点说就是：平常看到的那种普通的电机，断电后它还会因为自身的惯性再转一会儿，然后停下。而伺服电机和步进电机是说停就停，说走就走，反应极快。但步进电机存在失步现象。

伺服电机的应用场景

伺服电机的应用领域就太多了。只要是要有动力源的，而且对精度有要求的一般都可能涉及到伺服电机。如机床、印刷设备、包装设备、纺织设备、激光加工设备、机器人、自动化生产线等对工艺精度、加工效率和工作可靠性等要求相对较高的设备。

了解myCobot硬件

myCobot是的**硬件**是由**电子件**与**结构件**组成。电子件中包括了PCBA、控制器、舵机、充电器等；结构件包括了实体的塑料外壳、乐高固定口、法兰、紧固件轴承等。

myCobot微控制器Basic和Atom

myCobot的微控制器主要以装载在底座的**Basic微控制器**与装载末端的**Atom微控制器**组成。这样设计的原因是为了让机械臂的相关**运动程序**与**应用程序**分开，在保持机械臂具有一定实时性的同时，用户可以自行编程控制。

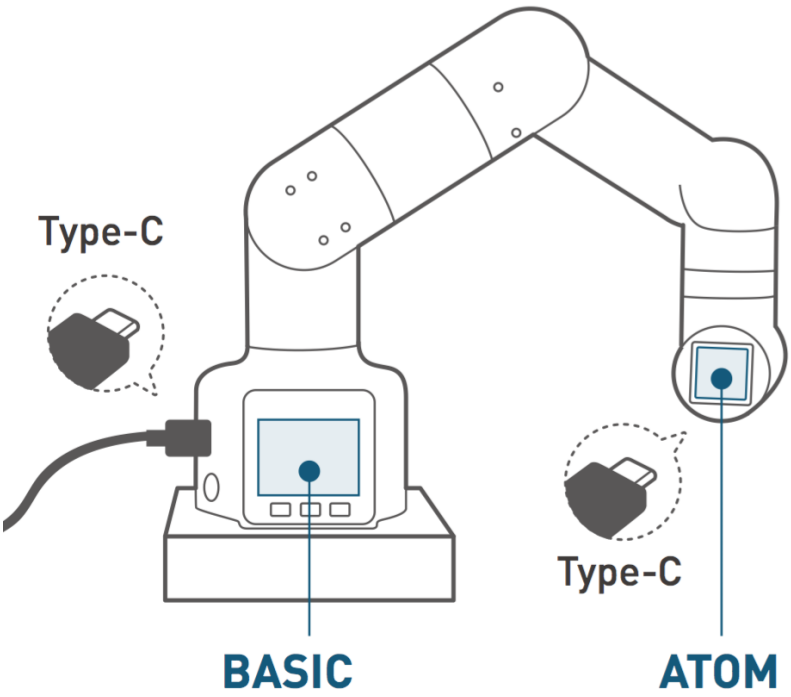
Basic - M5Stack Basic

- 主要用以myCobot的应用端程序，可以进行Arduino编程，uiFlow编程，通信或用户自己定义的各类软件。Basic相关程序为开源。可以参考我们的github获得更多信息。

ATOM - M5Stack Atom微控制器

- 主要进行机械臂的运动学算法控制：包括了正逆运动学，选解，加减速，速度同步，多次方插补，坐标转换等，需要的实时控制与多线程等。Atom相关程序暂不开源。

值得一提的是：Basic与Atom 之间通过**通信协议**进行通信，通信协议同时开放给所有用户。



myCobot所包含的所有电子件

myCobot电子件可划分为

- 底座（灰色外壳）
  - 充电板PCBA： 充电与电压保护功能
  - 基板PCBA： 控制信号转换功能
  - M5 Basic：主控制器
  - 舵机1号
- 机身（白色外壳）
  - 舵机2~6号
  - 舵机转接板
- 末端
  - M5 Atom：副控制器
- 外设 -充电器

myCobot所包含的所有结构件

myCobot结构件可划分为

- 底座（灰色外壳）
  - 通孔固定
  - 乐高口
- 机身
  - 白色塑料外壳

    - 固定件，紧固件，轴承等。
- 末端
    - 输出法兰（银色）
- 外设 -无

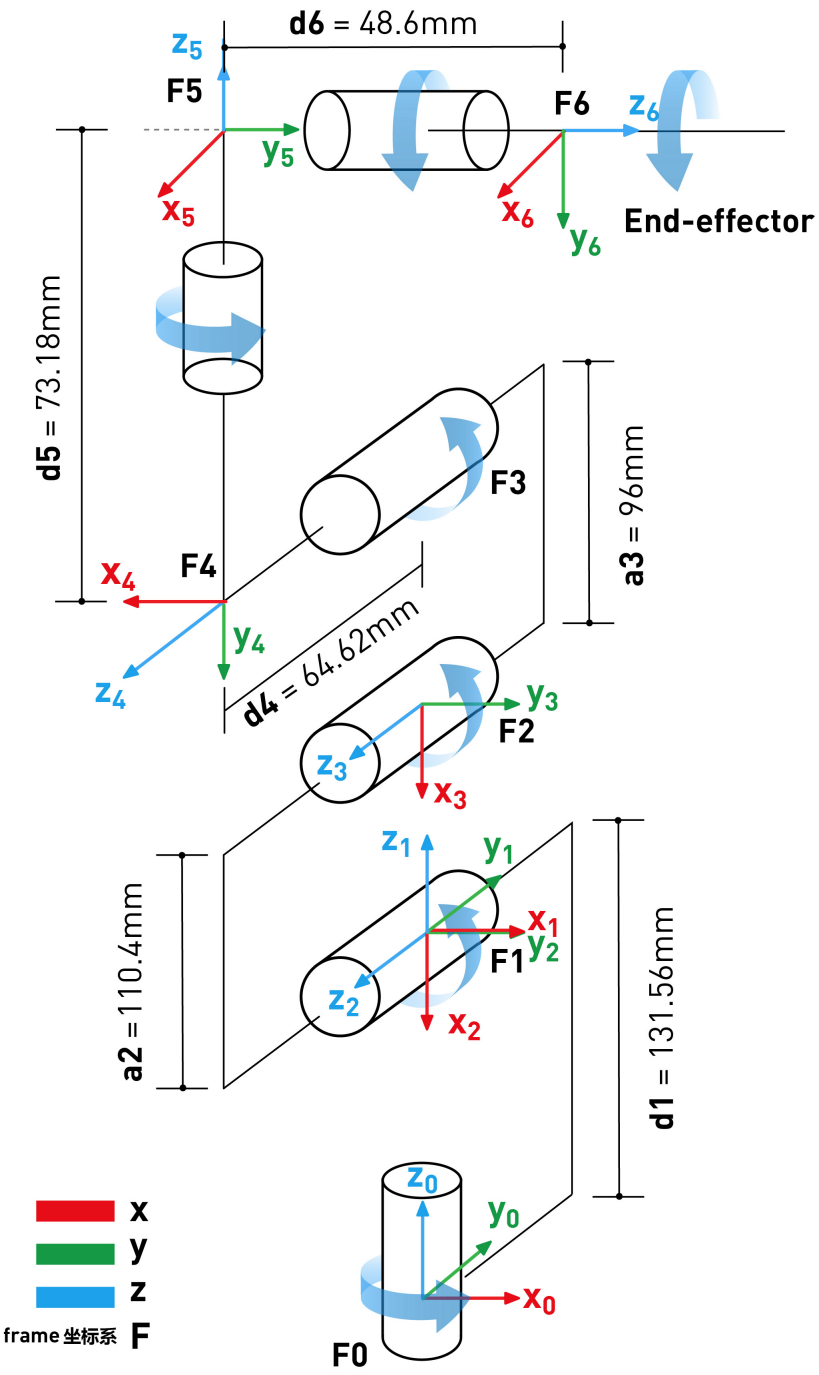
myCobot DH参数 坐标系

myCobot的DH参数为修正**DH参数**，具体参数值如下。

Joint	alpha	a	d	theta	offset
1	0	0	131.56	theta_1	0
2	PI/2	0	0	theta_2	-PI/2
3	0	-110.4	0	theta_3	0
4	0	-96	64.62	theta_4	-PI/2
5	PI/2	0	73.18	theta_5	PI/2
6	-PI/2	0	48.6	theta_6	0

> 关于DH参数的定义，可以参考https://www.zhihu.com/question/22365926

myCobot的DH参数所代表的坐标系如下。



维护维修

如果您的MyCobot产生了一定的问题，可以联系我们小管家进行维护维修。

底座BASIC

描述：

M5Stack BASIC 是M5Stack开发套件系列中的一款，拥有超高性价比与丰富案例资源的入门级开发套件.对于开发新手来说, Basic 是不二之选.

快速成型，超低门槛，直达产品级，M5Stack 开发板会是你物联网开发的不二之选.传统开发板只能用作验证和学习，M5的出现赋予了开发板更多的可能性，M5Stack 开发板采用了工业级外壳，再加上精致的外观设计，整体性能稳定，除了验证和学习的功能之外，还可以加速开发和产品化的进程.采用ESP32物联网芯片.集成Wi-Fi和蓝牙模块，拥有 16MB 的 SPI 闪存，双核低功耗的它在多种应用场景中有着非凡表现.由 30 多个 M5Stack 可堆叠模块，40 多个可扩展单元组成的硬件拓展体系，能够快速的帮助你搭建和验证你的物联网产品.

支持的开发平台和程序语言：Arduino，UIFlow (采用Blockly ，MicroPython语言).无论你的开发和编程能力处在何种水平，M5Stack 都将协助你，逐步的将想法变为现实.

如果你开发过 ESP8266，你会发现 ESP32 是 ESP8266 的完美升级版.相比之下，ESP32 具有更多 GPIO ，更多的模拟输入和两个模拟输出，多个外设接口（如备用UART）.官方开发平台 ESP-IDF 已经移植了 FreeRTOS，借助双核与实时操作系统，能使你更加高效的去组织你的程序代码，优化程序的执行效率.

myCobot 引脚说明

产品特性：

- 基于 ESP32 开发
- 内置扬声器，按键，LCD屏幕，电源/复位按键
- TF卡插槽(支持最大16GB)
- 内置锂电池
- 背部磁吸设计
- 可拓展的引脚与接口
- M-Bus总线母座
- 开发平台 UIFlow, MicroPython, Arduino

应用

- 物联网控制器
- STEM教育
- DIY作品
- 智能家居设备

规格参数：

主控资源	参数
ESP32-D0WDQ6	240MHz dual core, 600 DMIPS, 520KB SRAM, Wi-Fi, dual mode Bluetooth
Flash	16MB
输入电压	5V @ 500mA
主机接口	TypeC x 1, GROVE(I2C+I/0+UART) x 1
Core底座接口	PIN (G1，G2，G3，G16, G17, G18, G19, G21, G22, G23, G25, G26, G35, G36)
IPS屏幕	2 inch, 320x240 Colorful TFT LCD, ILI9342C，最高亮度853nit
扬声器	1W-0928
按键	自定义按键 x 3
锂电池	110mAh @ 3.7V
天线	2.4G 3D天线
工作温度	32°F to 104°F (0°C to 40°C)
净重	47.2g
毛重	93g
产品尺寸	54 x 54 x 18mm
外壳材质	Plastic (PC)

管教映射

LCD 像素：320x240 TF 卡最大支持 16GB

LCD 屏幕 & TF

ESP32 Chip	GPIO23	GPIO19	GPIO18	GPIO14	GPIO27
ILI9342C	MOSI/MISO	/	CLK	CS	DC
TF卡	MOSI	MISO	CLK		

按键 **&** 喇叭

ESP32 Chip	GPIO39	GPIO38	GPIO37	GPIO25
按键引脚	BUTTON A	BUTTON B	BUTTON C	/
喇叭				DA PIN

电源管理芯片 **(IP5306)** 是定制 **I2C** 版本，它的 **I2C** 地址是 **0x75**。点击这里查看 **IP5306** 的寄存器手册。

ESP32 Chip	GPIO22	GPIO21	5V	GND
GROVE A	SCL	SDA	5V	GND
IP5306	SCL	SDA	5V	GND

IP5306充/放电，电压参数

充电	放电
0.00 ~ 3.40V -> 0%	4.20 ~ 4.07V -> 100%
3.40 ~ 3.61V -> 25%	4.07 ~ 3.81V -> 75%
3.61 ~ 3.88V -> 50%	3.81 ~ 3.55V -> 50%
3.88 ~ 4.12V -> 75%	3.55 ~ 3.33V -> 25%
4.12 ~ / -> 100%	3.33 ~ 0.00V -> 0%

ESP32 ADC/DAC |ADC1 | ADC2 | DAC1 | DAC2| |--- | :---: | :---: | :---:| |8 通道 | 10 通道 | 2 通道 | 2 通道| |G32-39 | G0/2/4/12-15/25-27| G25|G26|

充电电流测量值

充电电流	充满后电流(关机)	充满电(开机）
0.55A	-	0.066A

相关链接

Datasheet

- ESP32
- IP5306

API

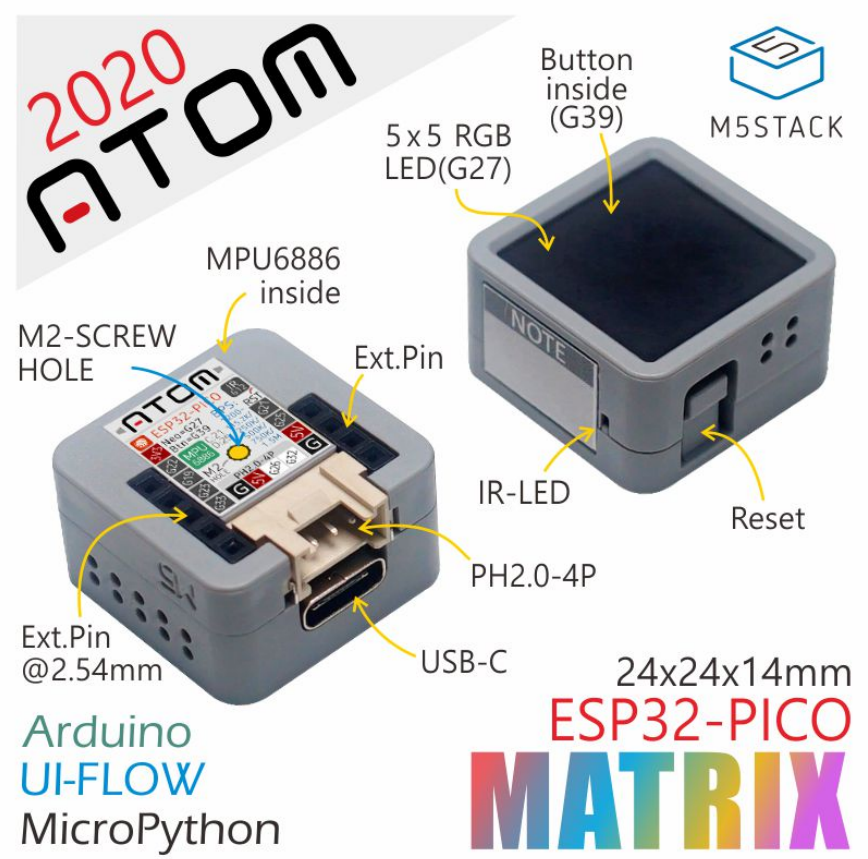
- Arduino API

原理图

- 原理图.pdf)

M5 Stack Atom 副控

描述：



ATOMMatrix 是 M5Stack 开发套件系列中一款非常小巧的开发板，其大小只有 2424mm，*提供更 多 GPIO 供用户自定义，非常适合做嵌入式的智能硬件。主控采用 ESP32-PICO-D4 方案，集成 Wi-Fi 和 蓝牙模块，拥有 4MB 的 SPI 闪存，板载 Infra-Red，面板上有 5*5 RGB Led 矩阵、内置 IMU 姿态传感器 （MPU6886),在 Neo Led 矩阵下方隐藏一颗可编程按键，板载 Type-C 接口可以快速实现程序上传下载， 此外还提供一个 HY2.0 4P 接口用于连接外设。背面具有一个 M2 螺丝孔用于固定。

注意：在使用 FastLED lib 时 RGB LED 的建议亮度值为 20，请不要将其设置过高的亮度数值，以免损坏 LED 和亚克力屏幕。(在 ATOM lib 中，我们已将其合适的亮度范围映射为 0~100)

产品特性：

- 基于ESP32开发
- 机身小巧
- 内置3轴陀螺仪和3轴加速计
- 可编程按键

- RGB
- LED点阵屏
- 红外发射功能
- 可扩展的引脚与接口
- 开发平台 UIFlow, Arduino

应用：

- 物联网节点
- 微型控制器
- 可穿戴设备

规格参数：

主控资源	参数
ESP32	240MHz dual core, 600 DMIPS, 520KB SRAM, Wi-Fi, dual mode Bluetooth
Flash	4MB
输入电压	5V @ 500mA
主机接口	TypeC x 1, GROVE(I2C+I/0+UART) x 1
PIN接口	G19, G21，G22，G23，G25, G33
RGB LED	WS2812C 2020 x 25
MEMS	MPU6886
IR	Infrared transmission
按键	自定义按键 x 1
天线	2.4G 3D天线
工作温度	32°F to 104°F (0°C to 40°C)
净重	3g
毛重	14g
产品尺寸	24*24*14mm
包装尺寸	43*43*20mm
外壳材质	Plastic (PC)

3.4 myCobot舵机与电机

规格参数：

规格	参数
外观尺寸	45.2X24.7X35mm
堵转扭矩	19.5 kg·cm @ 7.4V
堵转速度	52RPM@7.4V
反馈	负载/位置/速度/电压/电流/温度
电子保护	过热/过流/过压/过载保护

结构特点：

外壳采用强度更高的工程塑胶外壳，优化中心点准距，整体结构更加紧凑， 舵机齿轮采用1:345的铜齿组合，扭矩更大；在同扭力条件下，较标准款舵机的尺寸，会显得更加矮身（5mm），机身采用双轴结构设计，圆衬立体的结构特性，配合金属主副舵盘和双出线的布线方式，适合在四足机器人、蛇形机器人、桌面机器人、人形机器人、机械臂中应用。

电控功能：

- 一、**具备加速度启动停止功能**：可设定速度和加速度值，运动效果更佳柔和。
- 二、**高精度**，360度绝对位置4096位精度，最高位置解析0.088度，如控制90度，即输入4096/360*90=1024，如控制180度，即输入4096/360*180=2048，以此推算。
- 三、**具备四种工作模式切换**（模式0位置伺服，模式1速度闭环，模式2速度开环，模式3步进伺服）。
  - 1、模式0：位置模式，默认此模式。在该模式下可以实现360度绝对角度控制。支持加速度运动。
  - 2、模式1：速度闭环，在编程界面下，运行模式设定为1，切换至速度闭环模式，在速度栏下输入相应的速度即可运行。
  - 3、模式2：速度开环，在编程界面下，运行模式设定为2，切换至速度开环模式，在时间栏下输入相应的时间即可运行。
  - 4、模式3：步进模式，在编程界面下，最大/小角度限制设置为0，运行模式设定3，切换至步进模式，在位置栏输入位置即可朝目标位置步进运动，再次单击位置，可继续朝同一个方向进行步进运动。
- 四、**多圈模式**，360度绝对控制与反馈，最高精度下可以正负7圈绝对位置控制，但掉电圈数不保存，只保留绝对位置反馈值。
- 五、**一键校准**，360度角度任意位置安装，(40号（十进制）地址输入128（十进制）)一键较正当前位置为中位（2048（十进制））。
- 六、**TTL通讯电平**，半双工异步通讯，总线协议支持调整读写参数，增加同步读功能（发一条指令，可依次接收来自总线上的每个舵机的回读指令。）
- 七、**多种保护**，（过载，过流，过压，过热，可设置开关，改变条件参数）

- 1、过载保护：通过位置检测方式，起始位置到达目标位置的运动过程中，遇到障碍物堵转后，检测当前位置不是目标位置时，持续2S后卸力（默认堵转的20%力）。直至触发新命令，解保护。
- 2、过流保护：通过设定的电流值，检测当前电流是否到达设定的电流值，当到达后2S后卸力（默认扭力为0）。直至触发新命令，解保护。
- 3、过压保护：检测当前电压值，超过设定的电压值即报警显示过压。
- 4、过热保护：检测当前电机温度，超过设定的温度值即报警显示过热。

- 八：多种反馈：
  - 1、负载反馈：当前控制输出驱动电机的电压占空比，满格是1000=100%的扭力输出。
  - 2、电流反馈：当前舵机工作电流，1=6.5毫安
  - 3、电压反馈：当前舵机工作电压，70=7V，0.1V
  - 4、温度反馈：当前舵机内部工作温度（测量温度）。
  - 5、速度反馈：反馈当前电机转动的速度，单位时间（每秒）内运动的步数。
- 九、开放**PID参数**，便于修改运动效果，出厂已做优化，不建议修改。

了解 MyCobot 结构与固定

如何固定 myCobot

尽管 myCobot 自重比较轻，但是在 myCobot 的运动过程中，**如果不将 myCobot 的底面与桌面或其他底面相连，仍然会造成 myCobot 的摇晃或倾覆**。

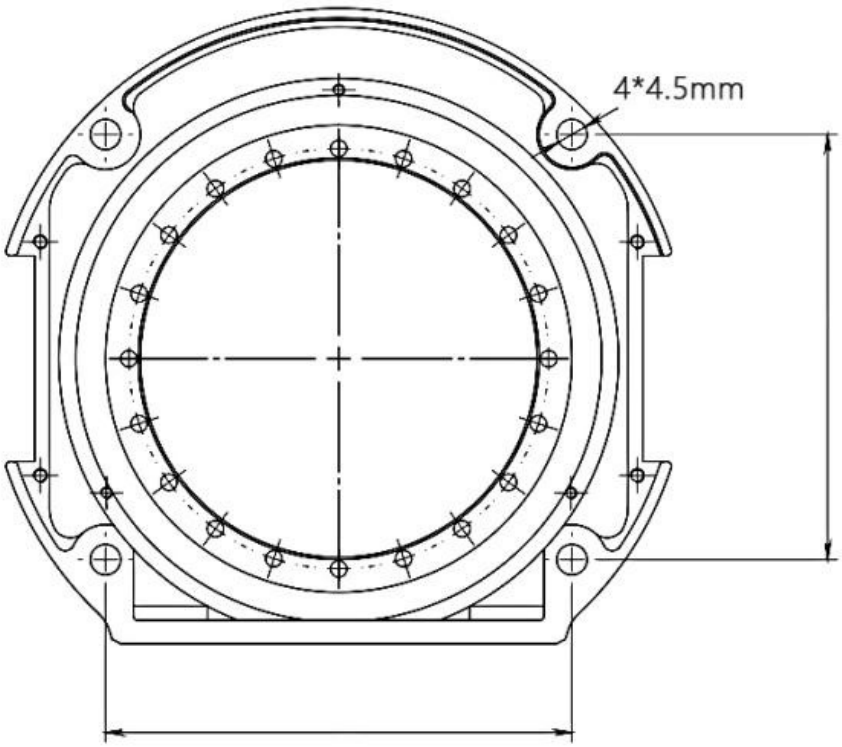
常见的固定 myCobot 的方式有两种：

- 1. 使用**乐高插键**固定在具有乐高接口的底座上
  我们销售的底座有 2 种：平面吸盘底座与 G 型夹底座，可以在myCobot周边底座找到
- 1. 使用**4 根自带螺丝**穿过 myCobot 底座，固定在有螺纹的底座上
  底座上 4 个螺丝孔的位置可以参考下方 myCobot 螺丝孔位连接图。

myCobot 底座的螺丝孔位连接

需要将机器人固定在牢固机座上才可以正常使用。机座重量要求：固定式机座，或移动式机座。 机器人基座接口尺寸

基座固定孔位是固定机器人与其他机座或平面的接口，具体孔位尺寸如下图所示，为 4 个通孔直径为 4.5mm 的沉头孔，可以用 M4 的螺栓进行固定。



请确定固定底座上有对应螺纹孔位，再进行安装。 正式进行安装前，请确认：

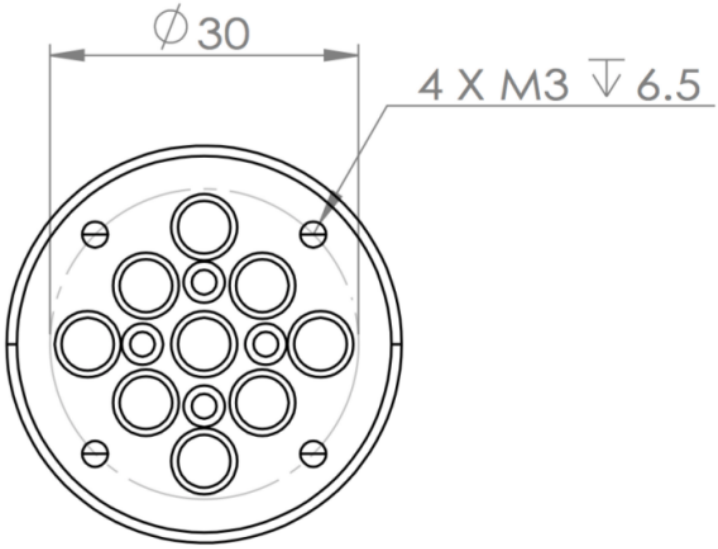
- 待安装环境符合以上《工作环境与条件》表格的要求。
- 安装位置不小于机器人工作范围，且留有足够大的安装、使用、维护、维修的空间。
- 将机座放置到合适位置。
- 安装相关工具已准备好，如螺丝、扳手等。

确认以上内容后，请将机器人搬运至机座安装台面上，调整机器人位置，将机器人基座固定孔位与机座安装台面上的孔位对准。对准孔位后，将螺丝对准孔位，拧紧即可。

注意：在机座安装台面上调整机器人位置时，请尽量避免在机座安装台面上直接推拉机器人，以免产生划痕。人工移动机器人时请尽量避免对机器人本体脆弱部分施加外力，以免造成机器人不必要的损伤。

末端装配图

机械臂末端同时兼容乐高科技件孔与螺丝螺纹孔



MyCobot Urdf 模型

https://github.com/elephantrobotics/mycobot_moveit/tree/main/urdf

myStudio

myStudio设计初衷

- myStudio是一个一站式的myRobot/myCobot等机器人的使用平台。
- 方便用户根据自己的使用场景，选择不同的固件并进行下载，同时学习相关的教材与视频。

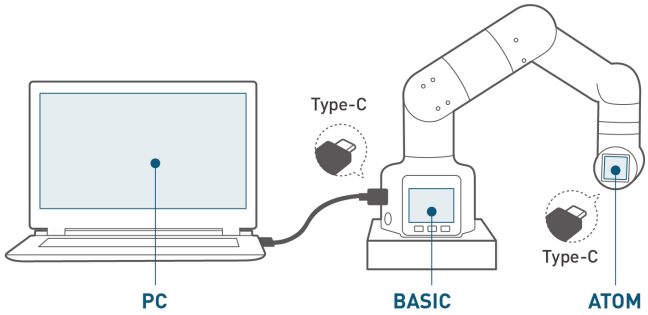
myStudio现支持的平台与版本

- Version 0.5
- 适用于：windows, mac, linux

myStudio有什么功能

- 烧入、更新固件
- 机器人使用视频教程
- 维护和维修方面的信息（如视频教程、Q&A等）

选择您的使用目的
请根据您的开发使用环境，下载不同的**PC，basic固件与atom固件**



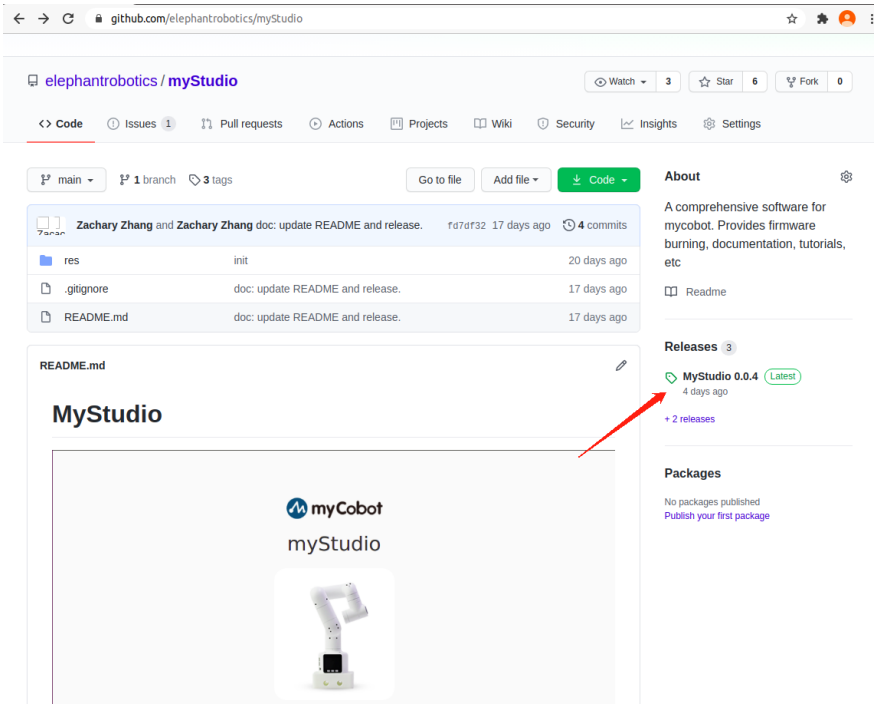
开发使用环境	PC所需Library	Basic所需固件	Atom所需固件
默认程序 Default Program	N/a	mainControl	atomMain
UI flow可视化编程 Visual Programming	UIFLOW	UIFLOW烧录器 由M5提供	atomMain
RoboFlow 工业级可视化编程软件 Industrial Programming Software	RoboFlow库	Transponder文件	atomMain
Arduino 创客！Maker!	Arduino IDE + M5Stack Lib 库 + MycobotBasic Lib 库	各类程序自定义 All Exapmles	atomMain
API 开发软件接口 on Desktop	Python/ C+	Transponder文件	atomMain
ROS Development ROS 开发	ROS库	Transponder文件	atomMain
通信协议 – USB/TxRx0(G1/G3)	通信协议阅读 Read Protocol	Transponder文件	atomMain
蓝牙或无线连接 BlueTooth 蓝牙直接连接Basic	通信协议阅读 Read Protocol	BT_Transponder文件	atomMain
手机控制器 phoneApp	myCobot 手机 (安卓/苹果) Android/iPhone	BT_Transponder文件	atomMain

下载myStudio

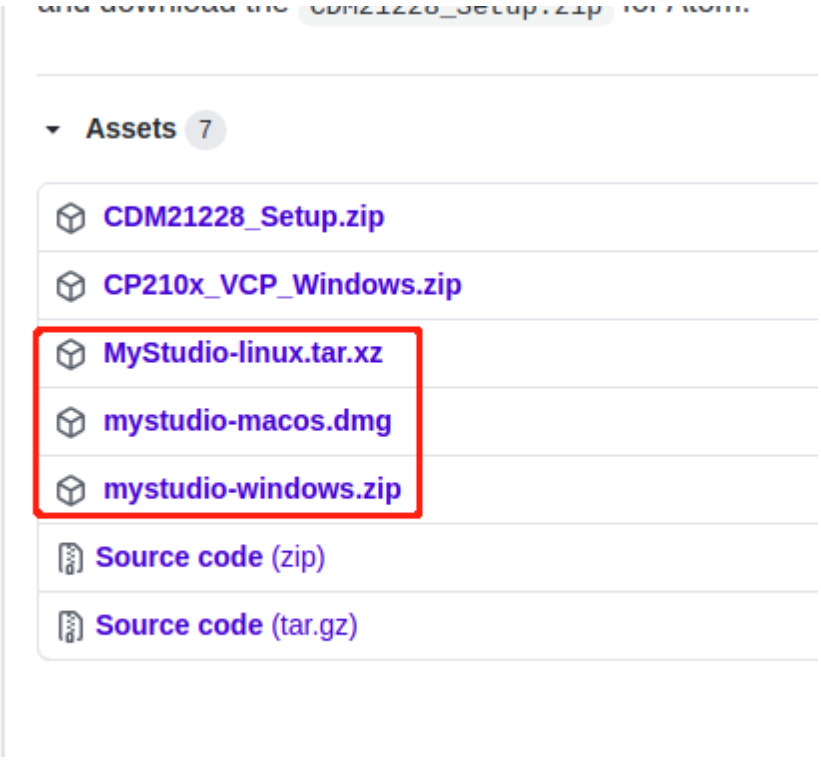
安装 myStudio

myStudio 下载地址：https://github.com/elephantrobotics/myStudio

选择最新的版本



然后针对不同的系统选择不同的版本即可



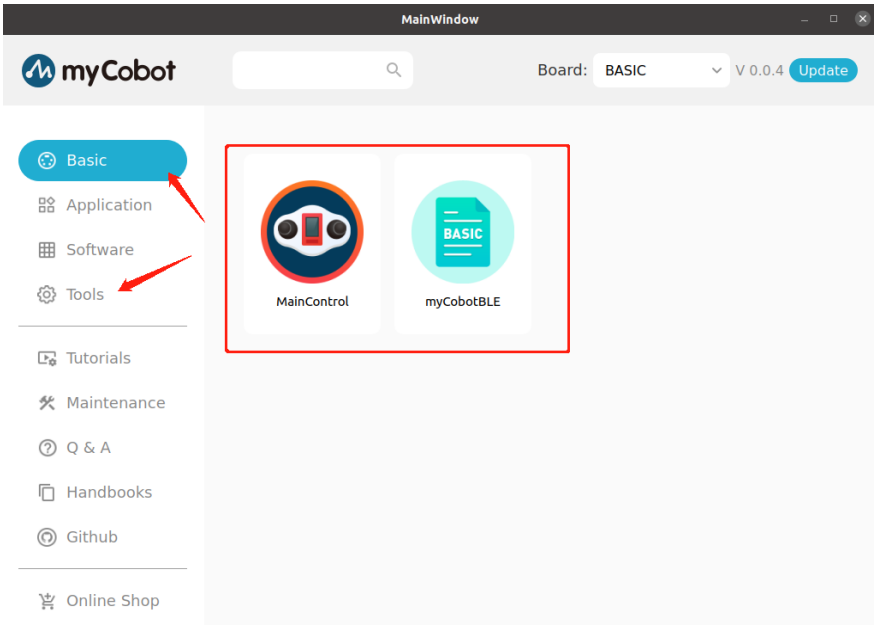
注意：不能安装在带空格目录的文件夹下！

下载basic与atom固件

烧入basic

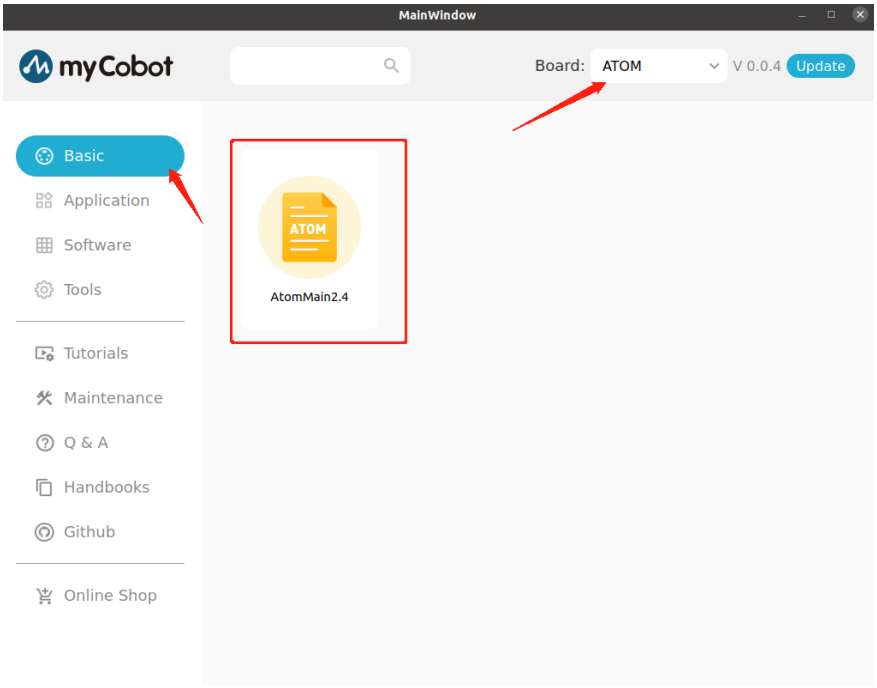
- 首先用USB连接basic开发板，myStudio的连接窗口会显示出已连接的开发板，选中点击Connect即可

- 然后 `Basic` 和 `工具` 里面有basic相关的固件 选择想要烧入的固件，点击烧入即可



烧入atom

- 和 `basic` 的烧入一样，用usb连接末端的atom
- 在 `Board` 一栏可以选择 `ATOM` ,侧边栏 `Basic` 就会出现atom的固件
- atom的固件就只有一个，点击烧入即可



mySudio使用

https://www.bilibili.com/video/BV1Qr4y1N7B5/

Q&A

Q: 第一次点击侧边栏 `Tools` 会卡住？

A: 请确保你的网络状态良好。

myCobot phone controller

这是由 elephant robotics 推出，用来控制 mycobot 的手机 app。 通过蓝牙的方式连接 mycobot。

> Note: 目前为测试版本

下载

你可以在 github 的 release 页面下载。

同时我们也以将该测试版本推送到 google store 和 apple store, 你也可以尝试直接在应用商店中下载。

而且，我们还提供了下载二维码，你也可以通过扫描下面的二维码来下载：

使用

固件烧录

为了可以正确使用，必须确保 mycobot 中的固件是对应的。

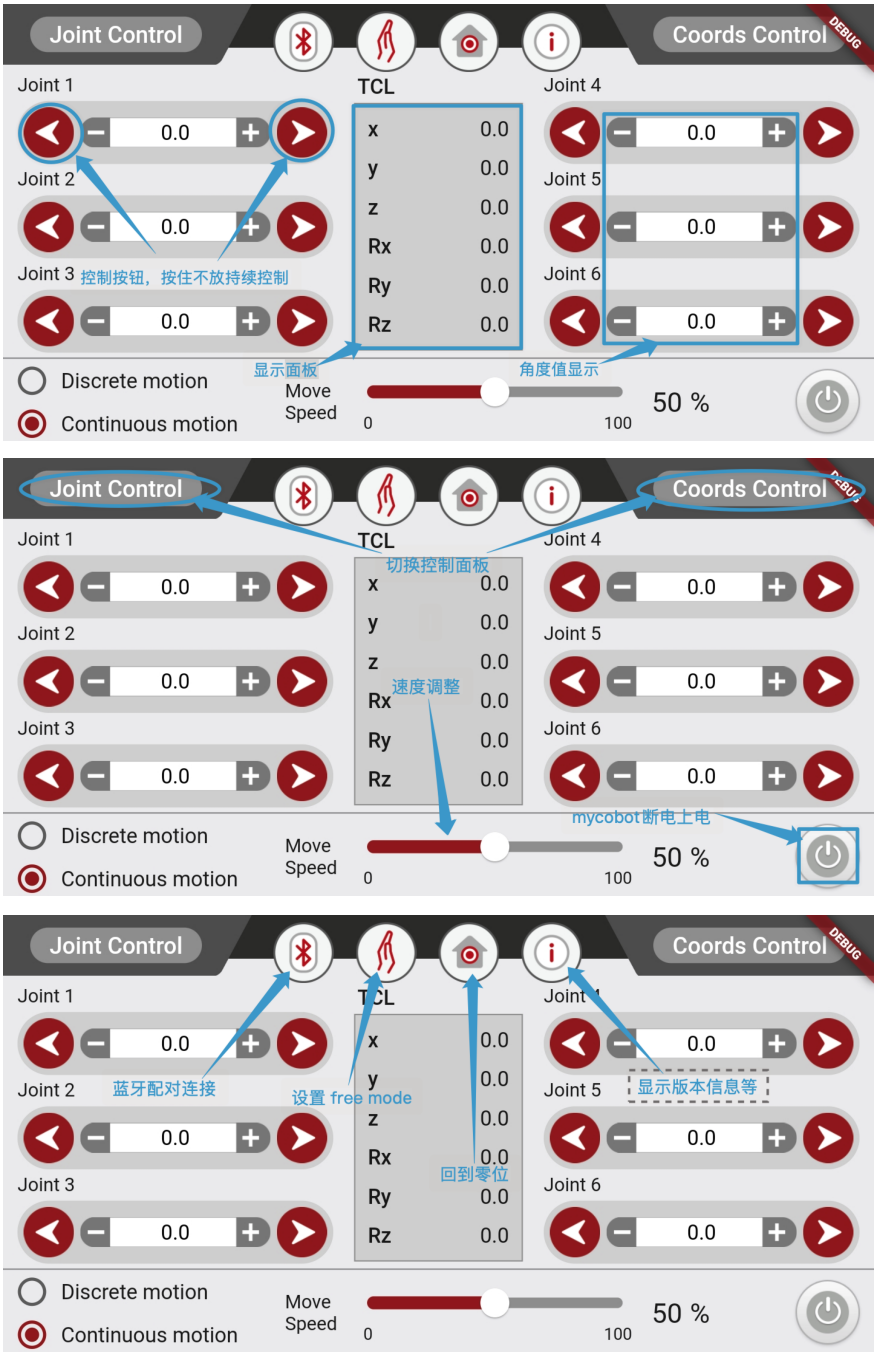
请在 mycobot 的 basic 中烧录 `myCobotBLE` 来与 app 进行蓝牙通信。

由于测试版限制，请确保 mycobot 的 atom 中烧录 Atom2.5

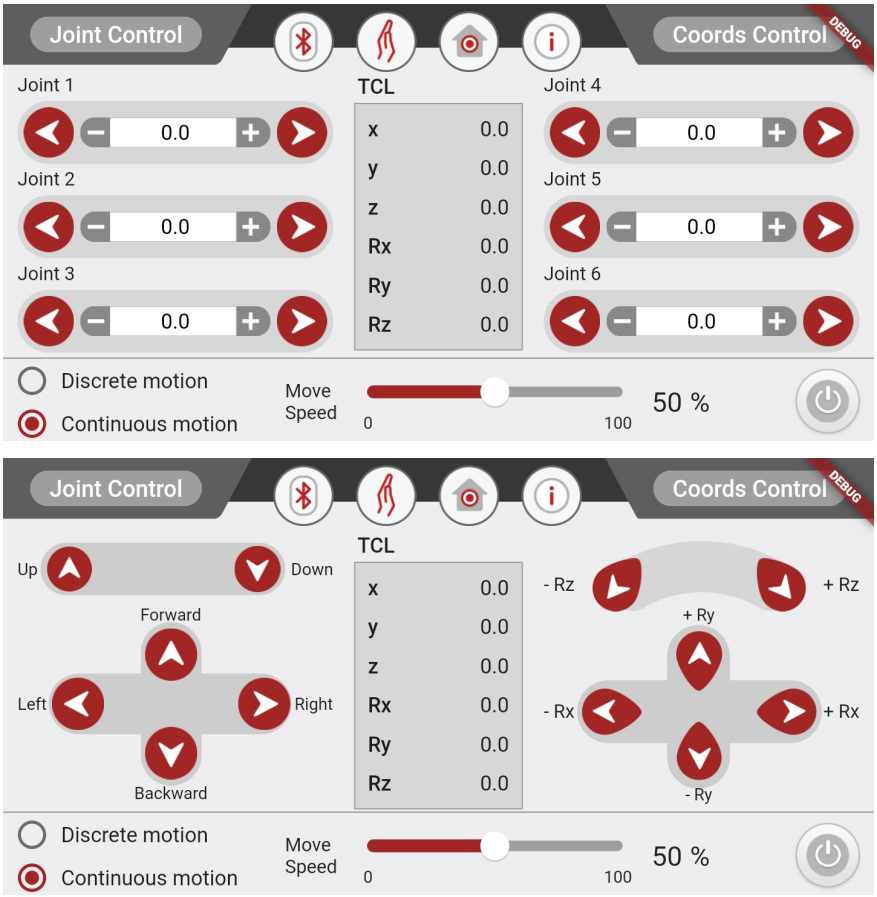
配对连接

请在 app 中搜索设备并连接，请不要在手机设置中连接，以防无法正常使用 app。

操作介绍



屏幕截图


Joint Control
Coords Control
DEBUG
Joint 1
0.0
Joint 2
0.0
Joint 3
0.0
TCL
x 0.0
y 0.0
z 0.0
Rx 0.0
Ry 0.0
Rz 0.0
Joint 4
0.0
Joint 5
0.0
Joint 6
0.0
Discrete motion
Continuous motion
Move Speed
0
100
50 %
Joint Control
Coords Control
DEBUG
Up
Down
Forward
Left
Right
Backward
TCL
x 0.0
y 0.0
z 0.0
Rx 0.0
Ry 0.0
Rz 0.0
- Rz
+ Rz
+ Ry
- Rx
+ Rx
- Ry
Discrete motion
Continuous motion
Move Speed
0
100
50 %

软件平台与API

在开发前，请确认您的myCobot中的 Basic和Atom是使用的**最新的固件**，以便试用于您的使用环境。

您可以通过**myStudio**进行固件的更新

我们现阶段支持以下软件API的开发

Arduino 开发

- 适合创客开发，可以使用各类Arduino程序库。

uiFlow 开发

- 适合初学者开发，基于拖拽式编程，可以适配全套M5教育套装。

python 开发

- 适合有一定python开发水平的用户开发。基于python3。

ROS&MoveIt 开发

- 适合专业相关开发，可以进行机械臂的仿真、计算，轨迹规划等。

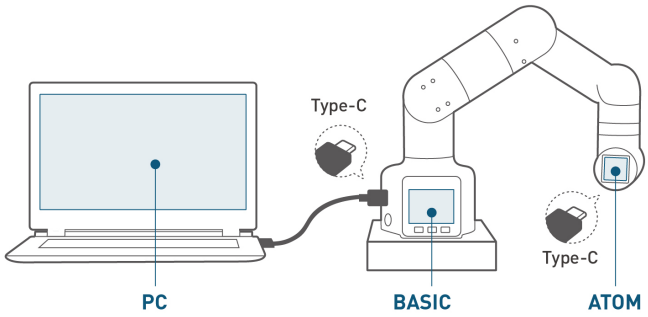
roboFlow 开发

- 适合工业类相关开发，roboFlow是大象机器人推出的一款面向工业的操作系统，适配大象机器人所有系列机械臂。

通信协议与报文 开发

- 适合更为广泛的串口开发，可以直接通过发送报文的形式进行通信。

> 除此之外，我们还在开发 C# 等软件接口API的开发。



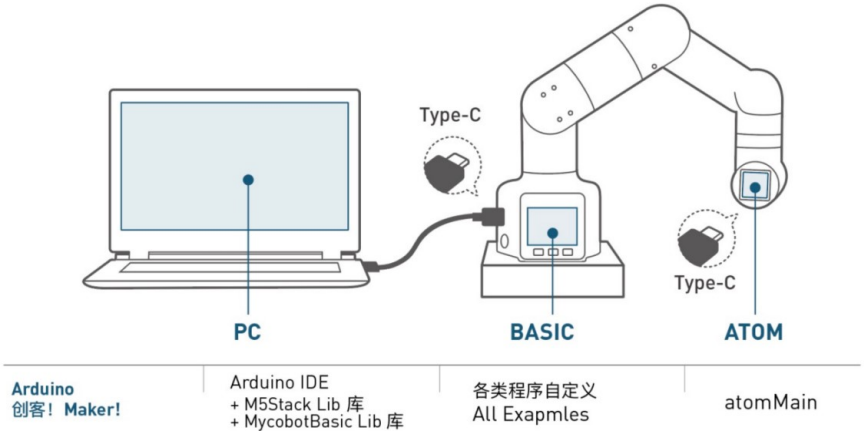
开发使用环境	PC所需Library	Basic所需固件	Atom所需固件
默认程序 Default Program	N/a	mainControl	atomMain
UI flow可视化编程 Visual Programming	UIFLOW	UIFLOW烧录器 由M5提供	atomMain
RoboFlow 工业级可视化编程软件 Industrial Programming Software	RoboFlow库	Transponder文件	atomMain
Arduino 创客！ Maker!	Arduino IDE + M5Stack Lib 库 + MycobotBasic Lib 库	各类程序自定义 All Exapmles	atomMain
API 开发软件接口 on Desktop	Python/ C+	Transponder文件	atomMain
ROS Development ROS 开发	ROS库	Transponder文件	atomMain
通信协议 – USB/TxRx0(G1/G3)	通信协议阅读 Read Protocol	Transponder文件	atomMain
蓝牙或无线连接 BlueTooth 蓝牙直接连接Basic	通信协议阅读 Read Protocol	BT_Transponder文件	atomMain
手机控制器 phoneApp	myCobot 手机 (安卓/苹果) Android/iPhone	BT_Transponder文件	atomMain

arduino环境下编程

1.开发前提

1.1链接设备

- 用Tyep-C数据线把机械臂底座上的Basic与PC端相连接



1.2固件要求

- ATOM 烧录 ATOMMain2.4 及以上
- Basic 无要求

1.3检测链接

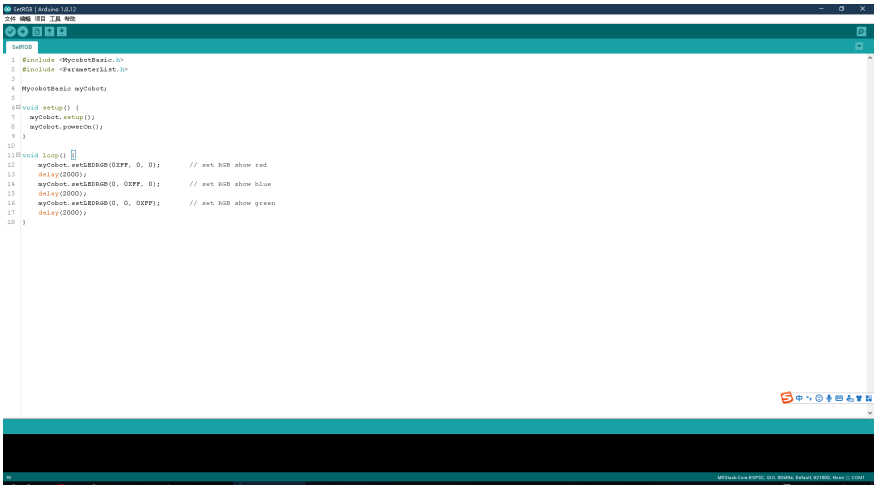
- 打开电脑设备管理器查看有无设备
  如未检测到设备，请更换USB连接线，如果显示无法使用，请安装点击下载 CP210X 驱动，下载完成后解压并安装所需的驱动版本即可使用。
- 打开Arduino IDE->工具 ->端口 查看有无设备 如未检测到设备，请更换USB连接线测试，或检测驱动是否安装成功。

2.开始开发

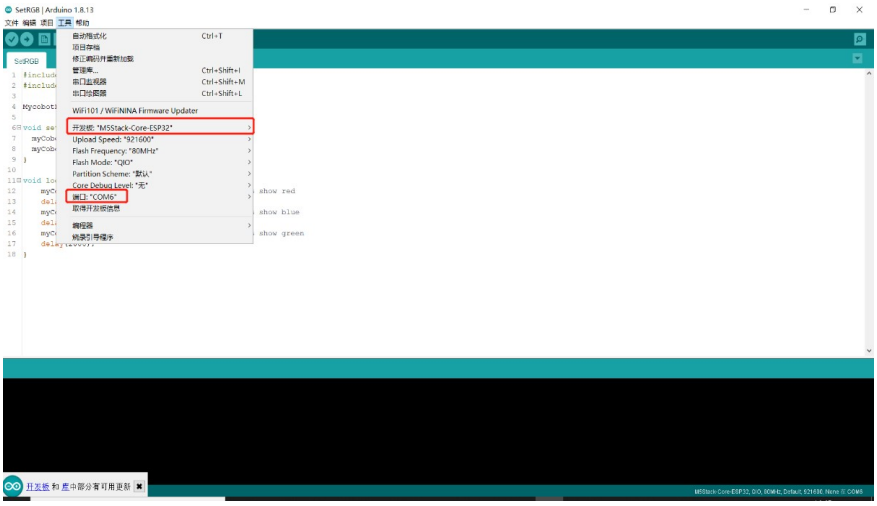
2.1 烧录一个官方demo

- 打开Arduino IDE， 选择->文件-> 示例-> mycobotbasic 就可以看到所有的项目示例
- 烧录一个简单的demo->SetRGB.ino.

从示例文件中打开SetRGB

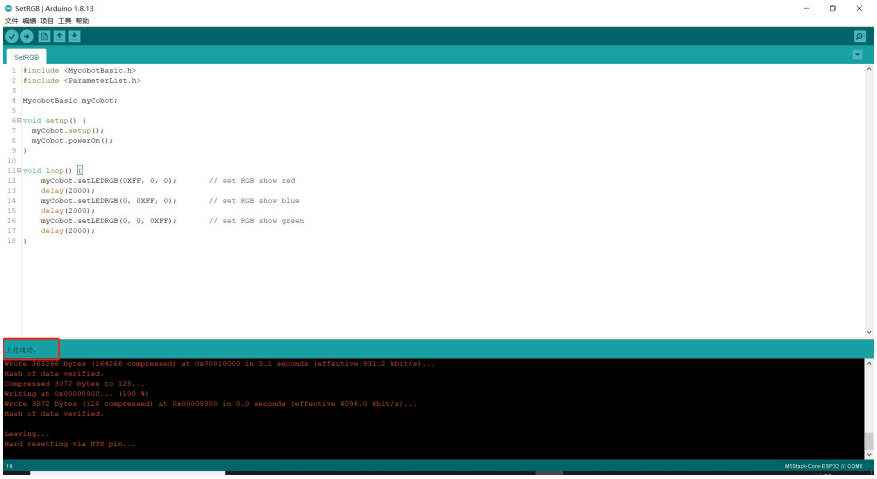


注意选择开发板：M5Stack-Core-ESP32和对应的com口



点击下载并等待右下方进度条跑完

等待直到右下方显示上传成功，程序就已经下载完成



这时我们就能看到Atom屏幕上循环显示红绿蓝灯

arduino API

1. 机器人整体运行状态 Overall Status

`powerOn();`

- 功能：atom 打开通讯（默认打开）
- 返回值：无

`powerOff();`

- 功能：atom 关闭通讯
- 返回值：无

`isPoweredOn();`

- 功能：atom 状态查询，返回 atom 链接状态
- 返回值：打开 TRUE、关闭 FALSE

`setFreeMove();`

- 功能：所有关节关闭扭力输出
- 返回值：无

2. 输入程序控制模式 MDI Mode and Robot Control (Manual Data Input)

`getAngles();`

- 功能：读取所有关节角度，使用时应定义一个 Angles angles，来接收读取到的角度，Angles 是库函数内置的变量或函数定义，可以定义一个内存为 6 的储存空间 angles，用来储存角度变量，使用的方式与数组相同。
- 返回值：Angles 类型的数组

`writeAngle(int joint, float value, int speed);`

- 功能：发送单关节角度
- 参数说明： 关节序号 = joint，取值范围 1-6; 指定角度值 = value，取值范围约-170°~ + 170° 指定速度 = speed， 取值范围 0~100;
- 返回值：无

`writeAngles(Angles angles, int speed);`

- 功能：关节角度同步执行，同时发送六个关节的角度给执行器 Angles 为库函数声明的定义类型，指定 angles 是容量大小为 6 个数据的容器，可理解为数组，赋值时可使用 for 循环赋值，也可单独赋值。
- 参数说明： Angles[0] = 具体角度，Angles[2] = 具体角度，以此类推取值范围 0 – 90（范围为定义，取值范围应和 writeAngle 相同） 单位 ° 运动速度 = speed，取值范围 0~100 单位 %
- 返回值：无

`getCoords();`

- 功能：读取当前机械臂末端的 x,y,z,rx,ry,rz，使用时应定义一个 Coords tempcoords，来接收读取到的角度，Coords 是库函数内置的变量或函数定义，可以定义一个内存为 6 的储存空间 tempcoords，用来储存角度变量,使用的方式与数组相同。
- 返回值：Coords 类型下的一个数组，需要定义好 Coords 类型的变量

`isInPosition (Coords coord,bool is_linear);`

- 功能：读取当前机械臂末端的 x,y,z,rx,ry,rz，测试是否到达指定点位，使用时应定义一个 Coords tempcoords，来接收读取到的角度，Coords 是库函数内置的变量或函数定义，可以定义一个内存为 6 的储存空间 tempcoords，用来储存角度变量,使用的方式与数组相同。
- 返回值：Coords 类型下的一个数组，需要定义好 Coords 类型的变量

`writeCoord(Axis axis, float value, int speed);`

- 功能：发送单独坐标参数 x/y/z 的具体数值，末端会在单独方向上移动，
- 参数说明： 移动的路径坐标值 = value 取值范围 -300 – 300（axis=Axis::X,aixs=Axis::Y 和 axis=Axis::Z 为位置坐标分别为 X,Y,Z，单位 mm，位置坐标取值范围不统一，axis=Axis::RX, aixs=Axis::RY 和 axis=Axis::RZ 分别为 RX,RY,RZ 取值范围为-180° 至 180°，超出取值范围会返回 inverse kinematics no solution 提示） 指定速度 =speed 取值范围 0~100 单位 %
- 返回值：无

`writeCoords(Coords coords, int speed);`

- 功能：发送指定的坐标参数，参数的类型应是 Coords，需要声明一个 Coords 类型的变量，此变量的使用方法与数组相同
- 参数说明： coords[0] = X, coords[1] = Y, coords[2] = Z, X,Y,Z 取值范围 -300-300.00（取值范围未定义，超出范围会返回 inverse kinematics no solution 提示） 单位 mm RX,RY,RZ 取值范围 -180~180 指定速度 =speed 取值范围 0~100 单位 %
- 返回值：无

`checkRunning();`

- 功能：检查设备是否在运动
- 返回值：运动中回复 TRUE，否则回复 FALSE

`setEncoder(int joint, int encoder);`

- 功能：设定单一关节转动至指定电位值
- 参数说明：关节序号 = joint 取值范围 1-6；舵机电位值 = encoder 取值范围 0-4096（该范围应与每个关节的范围正相关）
- 返回值：无

`getEncoder(int joint);`

- 功能：获取指定关节电位值

- 参数说明：舵机序号 = joint 取值范围 1-6
- 返回值：int 类型 ， 参考取值范围 0-4096

```
setEncoders(Angles angleEncoders, int speed);
```

- 功能：设定机械臂六个关节同步执行至指定位置
- 参数说明：需要定义一个 Angles 类型的一个变量 angleEncoders,， angleEncoders 的使用方法等同于数组，对数组 angleEncoders 赋值，取值范围 0~4096（该范围应与每个关节的范围正相关），数组的长度范围是 6 指定速度 =speed，取值范围 0~100 单位 %
- 返回值：无

3. 微动控制模式 JOG Mode

```
jogAngle(int joint, int direction, int speed);
```

- 功能：控制设备单一关节向一个方向运动
- 参数说明： 关节舵机序号 =joint 取值范围 1-6 关节运动方向=Direction 取值范围 -1/1 指定速度 =speed，取值范围 0~100 单位%
- 返回值：无

```
jogCoord(Axis axis, int direction, int speed);
```

- 功能：控制设备在笛卡尔空间中向一个方向运动
- 参数说明： 设备方向选择 = axis 取值 X,Y,Z,RX,RY,RZ 关节运动方向 = Direction 取值 -1/1 指定速度 =speed，取值范围 0~100 单位 %
- 返回值：无

```
jogStop();
```

- 功能：停止已经开始的指定方向运动
- 返回值：无

```
pause();
```

- 功能：程序暂停运行
- 返回值：无

```
resume();
```

- 功能：程序继续运行
- 返回值：无

```
stop();
```

- 功能：程序停止运行
- 返回值：无

4. 运行辅助信息 Running Status and Settings

`getSpeed();`

- 功能：读取设备的当前运行速度
- 返回值：int 类型，数值范围 0-100，单位 %

`setSpeed(int percentage);`

- 功能：设置设备运行速度
- 参数说明：percentage 取值范围 0 – 100，单位 %

`getFeedOverride();` (暂未开放)

- 功能：读取 FeedOverride
- 返回值：float 类型的数值

`sendFeedOverride(float feedOverride);` (暂未开放)

- 功能：发送 FeedOverride

`getAcceleration();` (暂未开放)

- 功能：读取加速度
- 返回值：float 类型的数值

`setAcceleration(float acceleration);` (暂未开放)

- 功能：设置加速度
- 参数说明：acceleration 取值范围 0-100 （范围未定义）

`getJointMin(int joint);`

- 功能：读取关节最小限制角度
- 参数说明：关节舵机序号 = joint，取值范围 1-6
- 返回值：float 类型的数值

`getJointMax(int joint);`

- 功能：读取关节最大限制角度
- 参数说明：关节舵机序号 = joint，取值范围 1-6
- 返回值：float 类型的数值

`setJointMin(int joint, float angle);`

- 功能：设置关节最小限制角度
- 参数说明：关节舵机序号 = joint，取值范围 1-6；关节对应角度 = angle，取值范围-180 ~ 180
- 返回值：无

`setJointMax(int joint, float angle);`

- 功能：设置关节最大限制角度
- 参数说明：关节舵机序号 = joint，取值范围 1-6；关节对应角度 = angle，取值范围-180 ~ 180
- 返回值：无

5. 关节电机设置 Joint Servo Control

`isServoEnabled(int joint);`

- 功能：检测关节舵机是否连接正常
- 参数说明：关节舵机序号 = joint，取值范围 1-6
- 返回值：正常链接返回 TRUE，否则返回 FALSE

`isAllServoEnabled();`

- 功能：检测所有关节舵机是否连接正常
- 返回值：正常链接返回 TRUE，否则返回 FALSE

`getServoData(int joint, byte data_id);`

- 功能：读取舵机指定地址的数据参数
- 参数说明： 关节舵机序号 = joint，取值范围 1 - 6 ； 数据地址 = data_id，取值范围请参考下图 1.1 中地址
- 返回值：下图 1.1 中取值范围

地址	功能	取值范围	初始值值	取值解析
20	LED 报警条件	0-254	0	对应位设置 1 为开启闪灯报警 对应位设置 0 为关闭闪灯报警
21	位置环 P 比例系数	0-254	123 关节取值 8, 456 取值 5。	控制电机的比例系数
22	位置环 D 微分系数	0-254	123 关节取值 20，456 关节取值 13.	控制电机的微分系数
23	位置环 I 积分系数	0-254	0	控制电机的积分系数
24	最小启动力	0-1000	0	设置舵机的最小输出启动扭矩，设 1000 = 100% * 堵转扭力

`setServoData(int joint, byte data_id, byte data);`

- 功能：读取舵机指定地址的数据参数
- 参数说明：
  关节舵机序号 = joint，取值范围 1 - 6
  数据地址 = data_id，取值范围请参考上图 1.1 中地址
  数据 = 上图1.1中的取值范围
- 返回值：无

`setServoCalibration(int joint);`

- 功能：校准关节舵机当前位置为角度零点，对应电位值为 2048
- 参数说明：关节舵机序号 = joint，取值范围 1 – 6

`jointBrake();`

- 功能：刹车单个电机
- 参数说明：关节舵机序号 = joint，取值范围 1 – 6

`setPinMode(byte pin_no, byte pin_mode);`

- 功能：设置 atom 指定引脚的状态模式
- 参数说明： 引脚序号 = pin_no，取值范围：19、22、23、26、32、33 输出模式 = pin_mode 取值范围：0、1
- 返回值：无

6. Atom 末端 IO Atom IO Control

`setLEDRGB(byte r, byte g, byte b);`

- 功能：设定 atom 屏幕的 RGB 灯的颜色：
- 参数说明： 红色光对应参数值 = r，取值范围 0x00 – 0xFF； 绿色光对应参数值 = g，取值范围 0x00 – 0xFF； 蓝色光对应参数值 = b，取值范围 0x00 – 0xFF；
- 返回值：无

`setGripper(int data);`

- 功能：设置夹爪开合
- 参数说明：data 为 0 打开，为 1 关闭

7. 坐标控制模式

`setToolReference(Coords coords);`

- 功能：设置工具坐标系
- 参数说明： X,Y,Z 取值范围 -300-300.00（取值范围未定义，超出范围会返回 inverse kinematics no solution 提示） 单位 mm RX,RY,RZ 取值范围 -180~180
- 返回值： 无

`setWorldReference(Coords coords);`

- 功能：设置世界坐标系
- 参数说明： X,Y,Z 取值范围 -300-300.00（取值范围未定义，超出范围会返回 inverse kinematics no solution 提示） 单位 mm RX,RY,RZ 取值范围 -180~180
- 返回值： 无

`getToolReference();`

- 功能：获取工具坐标系
- 参数说明：无
- 返回值： X,Y,Z 取值范围 -300-300.00（取值范围未定义，超出范围会返回 inverse kinematics no solution 提示） 单位 mm RX,RY,RZ 取值范围 -180~180

`getWorldReference();`

- 功能：获取世界坐标系
- 参数说明：无
- 返回值： X,Y,Z 取值范围 -300-300.00（取值范围未定义，超出范围会返回 inverse kinematics no solution 提示） 单位 mm RX,RY,RZ 取值范围 -180~180

`setReferenceFrame(RFType rftype);`

- 功能：设置法兰坐标系
- 参数说明： RFType::BASE 为将机器人基座作为基坐标，RFType::WORLD 为将世界坐标系作为基坐标。
- 返回值： 无

`getReferenceFrame();`

- 功能：获取法兰坐标系
- 参数说明：无
- 返回值： X,Y,Z 取值范围 -300-300.00（取值范围未定义，超出范围会返回 inverse kinematics no solution 提示） 单位 mm RX,RY,RZ 取值范围 -180~180

`setEndType(EndType end_type);`

- 功能：设置末端坐标系
- 参数说明： EndType::FLANGE 为将末端设置为法兰，EndType::TOOL 为将末端设置为工具末端。
- 返回值： 无

`getEndType();`

- 功能：获取末端坐标系
- 参数说明：无
- 返回值： X,Y,Z 取值范围 -300-300.00（取值范围未定义，超出范围会返回 inverse kinematics no solution 提示） 单位 mm RX,RY,RZ 取值范围 -180~180

`setGripperValue();`

- 功能：设置夹爪电位，使用前请获取夹爪当前电位
- 参数说明： 输入值（0-4095）

- 返回值：无

`setGripperIni();`

- 功能：设置夹爪零点
- 参数说明：无
- 返回值：无

`getGripperValue();`

- 功能：获取夹爪当前电位值
- 参数说明：无
- 返回值：返回当前的夹爪电位值，范围 0 - 4085

`setGripperState;`

- 功能：设置夹爪的打开与关闭
- 参数说明： 0 打开，1 关闭
- 返回值：无

`setDigitalOutput;`

- 功能：设置IO引脚的工作状态
- 参数说明：0 input; 1 output; 2 pull_up_input
- 返回值：无

`getDitialInput;`

- 功能：读取输入
- 参数说明：无
- 返回值：无

`setPWMOutput(byte pin_no, int freq, byte pin_write);`

- 功能：设置ATOM末端IO输出指定占空比的PWM信号
- 参数说明：

```
pin_no：IO序号
freq：  时钟频率
pin_write：占空比 0 ~ 256;128表示50%
```

- 返回值：无

链接检测

功能说明

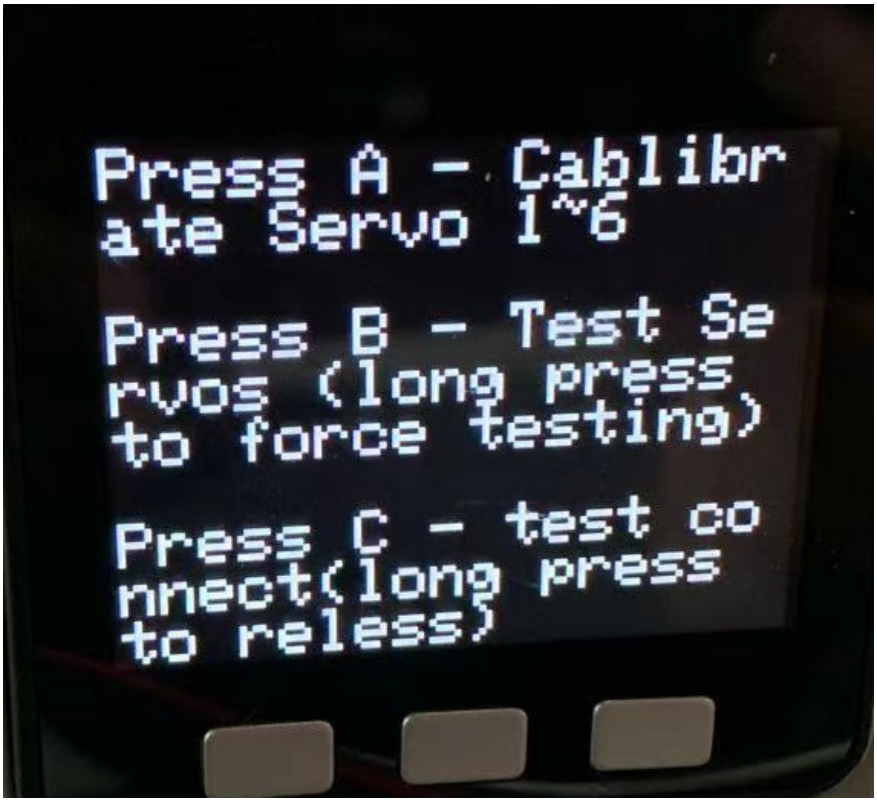
烧录 Connect-test 固件，可以检测设备连接情况，Basic 屏幕会显示链接的设备

固件要求

- 使用 MyStudio 给 Basic 烧录 connect-test
- 使用 MyStudio 给 Atom 烧录 AtomMain

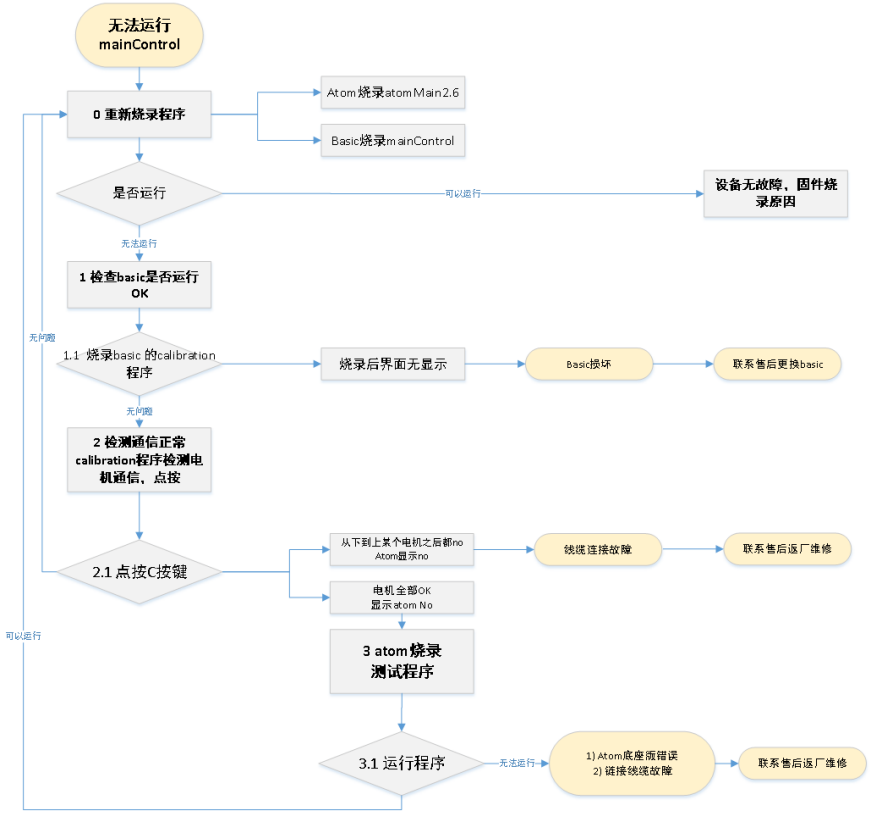
流程图示

- Atom 烧录 ATomMain
- Basic 烧录 Connect-test
- 烧录后如下图：



- 按提示点击C键：

- 如提示错误,分辨方法如下：

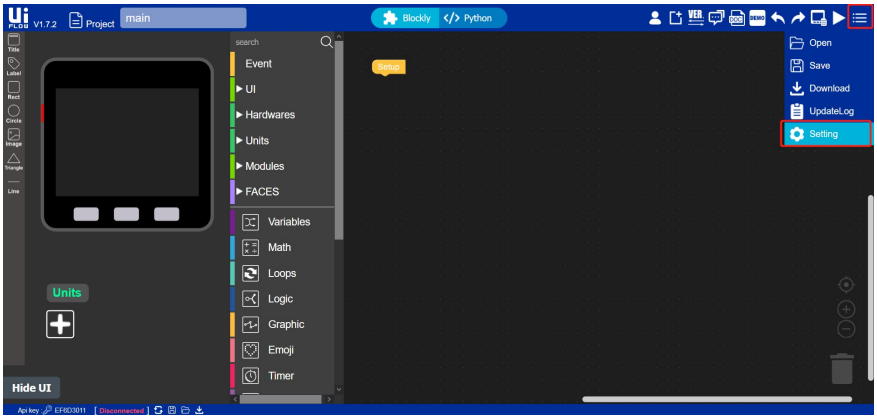


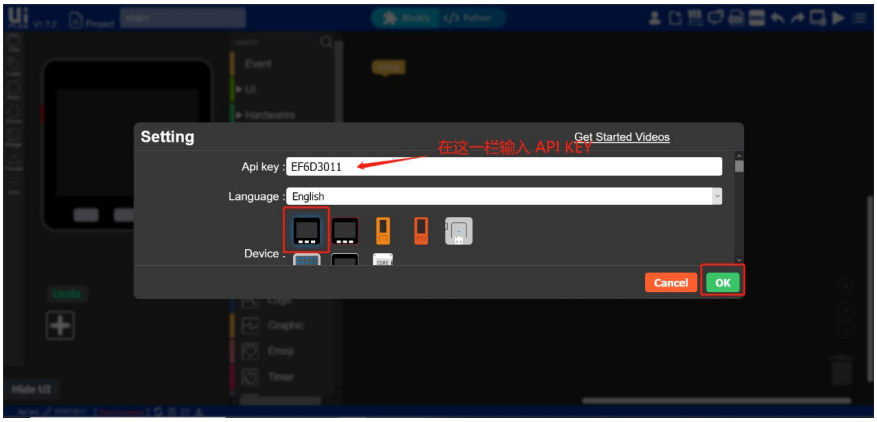
5.2 uiFlow环境下编程

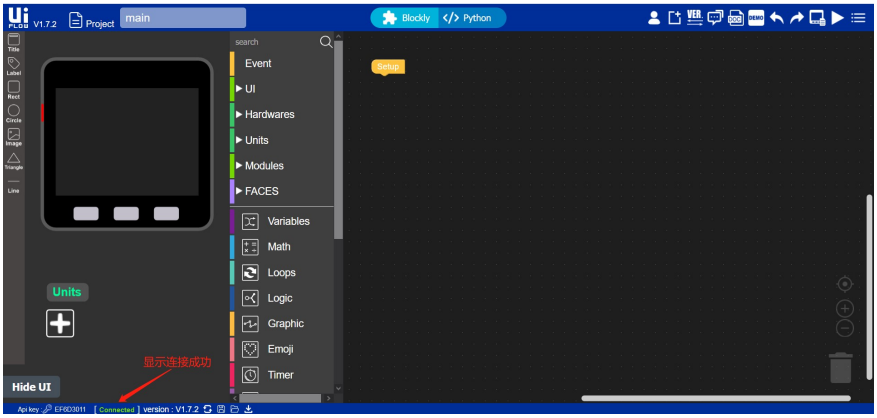
按2-background配置完成后

点击uiFlow网址 :https://flow.m5stack.com/

按下图所示进行操作//图43-47

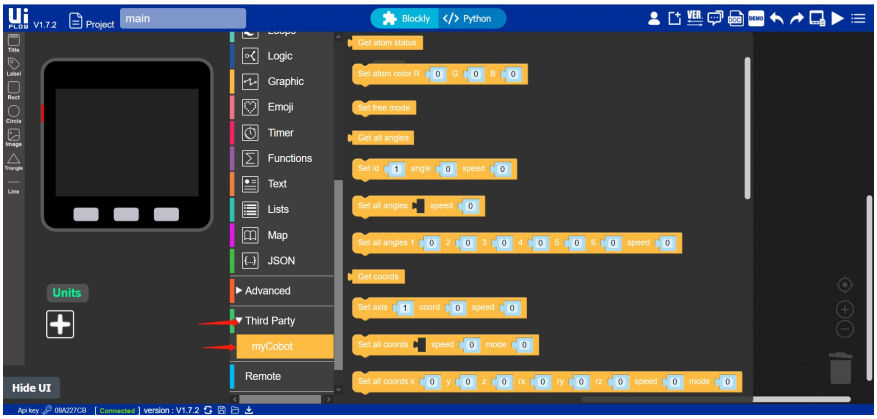


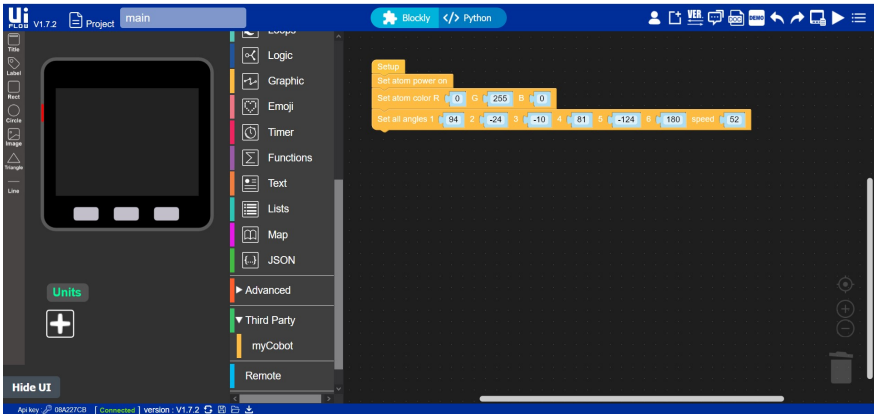




这时就可以开始编程了

如图所示让Atom屏幕显示绿色并对6个关节的角度进行调整


Project
main
Blockly
Python
Logic
Graphic
Emoji
Timer
Functions
Text
Lists
Map
JSON
Advanced
Third Party
myCobot
Remote
Units
Hide UI
Get atom status
Set atom color R 0 G 0 B 0
Set free mode
Get all angles
Set id 1 angle 0 speed 0
Set all angles speed 0
Set all angles 1 0 2 0 3 0 4 0 5 0 6 0 speed 0
Get coords
Set axis 1 coord 0 speed 0
Set all coords speed 0 mode 0
Set all coords x 0 y 0 z 0 rx 0 ry 0 rz 0 speed 0 mode 0


Project
main
Blockly
Python
Logic
Graphic
Emoji
Timer
Functions
Text
Lists
Map
JSON
Advanced
Third Party
myCobot
Remote
Units
Hide UI
Setup
Set atom power on
Set atom color R 0 G 255 B 0
Set all angles 1 94 2 -24 3 -10 4 81 5 -124 6 180 speed 52

5.3 python 环境下编程

确认开发目标

Pymycobot 是一个和 **Mycobot** 进行串口通讯的 **python** 包， 支持 **Python2, Python3.5** 及之后版本.

如果你想通过 Python 编程来控制 Mycobot，那么它是你的选择。

如何安装使用

在使用 pymycobot 之前请确保你已经拥有的 Mycobot 机械臂，并且做好了前置准备。

前置准备:

> 确保顶部的 Atom 烧入 `Atom` , 底部的 Basic 烧入 `Transponder`
> The firmware `Atom` and `Transponder` download address:
> https://github.com/elephantrobotics/myCobot/tree/main/Software

Pip

```
pip install pymycobot --upgrade --user
```

Notes:

> 现在只支持 `Atom2.4` 及之后版本. 如果你使用的之前版本, 请安装 `pymycobot 1.0.7` .

```
pip install pymycobot==1.0.7 --user
```

源码安装

```
git clone https://github.com/elephantrobotics/pymycobot.git <your-path>
cd <your-path>/pymycobot
# Install
python2 setup.py install
# or
python3 setup.py install
```

API 做为二级目录

如果你不想安装它，你可以使用它。 首先，你需要到项目的 github 中将其下载到本地。

> 项目地址：https://github.com/elephantrobotics/pymycobot

然后，将项目中的 pymycobot 文件放入你的项目之用，你就可以直接导入使用。

API 方法介绍

包含的类:

- MyCobot
- Angle
- Coord

MyCobot

```
from pymycobot.mycobot import MyCobot
```

机械臂状态

power_on()

- 描述: 机械臂上电。

power_off()

- 描述: 机械臂断电。

is_power_on()

- 描述: 判断机械臂是否上电。
- 返回值
  - `1` : power on
  - `0` : power off
  - `-1` : error

set_free_mode()

- 描述: 设置机械臂为自由模式。

MDI 模式和操作

get_angles()

- 描述: 获取所有关节角度。
- 返回值: `list` :一个浮点值的列表，长度为 6.

send_angle()

- 描述: 发送指定关节的角度。
- 参数

  `id` : 关节 id 值(common.Angle)

  `degree` : 需要发送的角度(float)

`speed`：(int) 0 ~ 100

- 例子

```
from pymycobot.mycobot import MyCobot
from pymycobot.genre import Angle

mycobot = MyCobot('/dev/ttyUSB0')
mycobot.send_angle(Angle.J2.value, 10, 50)
```

send_angles()

- 描述: 发送所有关节的角度。
- 参数

  `degrees`：包含所有关节的角度 (List[float]), 长度为 6.

  `speed`：(int) 0 ~ 100

- 例子

```
from pymycobot.mycobot import MyCobot
from pymycobot.genre import Angle

mycobot = MyCobot('/dev/ttyUSB0')
mycobot.send_angles([0,0,0,0,0,0], 80)
```

get_radians()

- 描述: 获取所有关节的弧度。
- 返回值: `list`：包含所有关节弧度值的列表，长度为 6.

send_radians()

- 描述: 发送所有关节的弧度。
- 参数

  `degrees`：该参数接收一个列表，长度为 6， 顺位包含所有关节的弧度值 (List[float])

  `speed`：(int) 0 ~ 100

- 例子

```
from pymycobot.mycobot import MyCobot
from pymycobot.genre import Angle

mycobot = MyCobot('/dev/ttyUSB0')
mycobot.send_angles_by_radian([1,1,1,1,1,1], 70)
```

get_coords()

- 描述: 获取当前坐标和姿态。
- 返回值

  `list`：包含坐标和姿态的列表，长度为 6，依次为 `[x, y, z, rx, ry, rz]`

send_coord()

- 描述: 发送单个坐标或姿态值。
- 参数

  `id` : (common.Coord), 接收一个 `Coord` 的值

  `coord` : 发送的值(float)

  `speed` : (int) 0 ~ 100

- 例子

```
from pymycobot.mycobot import MyCobot
from pymycobot.genre import Coord

mycobot = MyCobot('/dev/ttyUSB0')
mycobot.send_coord(Coord.X.value, -40, 70)
```

send_coords()

- 描述: 发送整体坐标和姿态。
- 参数

  `coords` : 接收一个顺位包含所有坐标和姿态值的列表(List[float])

  `speed` : (int) 0 ~ 100

- 例子

```
from pymycobot.mycobot import MyCobot
from pymycobot.genre import Coord

mycobot = MyCobot('/dev/ttyUSB0')
mycobot.send_coords([160, 160, 160, 0, 0, 0], 70, 0)
```

pause()

- 描述: 暂停运动。

resume()

- 描述: 恢复运动。

stop()

- 描述: 停止运动。

is_paused()

- 描述: 判断是否暂停。
- 返回值 (bool):
  - `1` - paused
  - `0` - not paused
  - `-1` - error

is_in_position()

- 描述: 判断时候到达指定位置。
- 参数:
  - `data`：接收包含角度值或坐标姿态值的列表
  - `id`：标记给入 `data` 的类型
    - `0`：表示给入的为角度值
    - `1`：表示给入的为坐标姿态值
- 返回值 (bool):
  - `1` - true
  - `0` - false
  - `-1` - error

JOG 模式和操作

jog_angle()

- 描述: 控制指定关节持续移动
- 参数
  - `joint_id`：关节 id(`int`) 1 ~ 6
  - `direction`：移动方向， 0 - 负值移动， 1 - 正值移动
  - `speed`：速度 0 ~ 100

jog_coord()

- 描述: 控制指定坐标或姿态值持续移动
- 参数
  - `coord_id`：(`int`) 1 ~ 6
  - `direction`：移动方向， 0 - 负值移动， 1 - 正值移动
  - `speed`：速度 0 ~ 100

jog_stop()

- 描述: 停止 jog 控制下的持续移动

运行状态和设置

get_speed()

- 描述: 获取速度。
- 返回值: speed: (int) 0 ~ 100

set_speed()

- 描述: 设置速度。
- 参数: speed: (`int`) 0 ~ 100

get_joint_min_angle()

- 描述: 获取指定关节的最小角度。
- 参数: `joint_id` : (`int`)
- 返回值: `angle` : (`float`)

get_joint_max_angle()

- 描述: 获取指定关节的最大角度。
- 参数: `joint_id` : (`int`)
- 返回值: `angle` : (`float`)

舵机控制

is_servo_enable()

- 描述: 判断指定关节是否连通连通。
- 参数: servo id: (`int`)
- 返回值 (int):
  - `0` : disable
  - `1` : enbale
  - `-1` : error

is_all_servo_enable()

- 描述: 判断所有关节是否连通。
- 返回值 (int):
  - `0` : disable
  - `1` : enbale
  - `-1` : error

release_servo()

- 描述: 放松指定关节
- 参数: servo id: (`int`)

focus_servo()

- 描述: 上电指定关节
- 参数: servo id: (`int`)

Atom IO

set_color()

- 描述: 设置顶部 RGB 灯色。

- 参数
  - `r` : 0 ~ 255
  - `g` : 0 ~ 255
  - `b` : 0 ~ 255

set_gripper_state()

- 描述: 控制夹爪开合状态。
- 参数
  - `flag` : 状态标记位(`int`) 0 - open, 1 - close
  - `speed` : 0 ~ 100

set_gripper_value()

- 描述: 控制夹爪角度。
- 参数
  - `value` : 角度值(`int`) 0 - 4096
  - `speed` : 0 ~ 100

set_gripper_int()

- 描述: 设置夹爪初始化位置，设置当前位置为 2048

Angle

```
from pymycobot.genre import Angle
```

描述

关节的实例类。建议使用该类选择关节。

Coord

```
from pymycobot.genre import Coord
```

描述

坐标的实例类。建议使用该类选择坐标。

ROS 简介

ROS 是机器人操作系统（Robot Operating System）的英文缩写。ROS 是用于编写机器人软件程序的一种具有高度灵活性的软件架构。

ROS 图标：

ROS 是开源的，是用于机器人控制的一种后操作系统，或者说次级操作系统。它提供类似操作系统所提供的功能，包含硬件抽象描述、底层驱动程序管理、共用功能的执行、程序间的消息传递、程序发行包管理，它也提供一些工具程序和库用于获取、建 立、编写和运行多机整合的程序。

ROS 的首要设计目标是在机器人研发领域提高代码复用率。ROS 是一个分布式的进程（也就是“节点”）框架，这些进程被封装在易于被分享和发布的程序包和功能包中。ROS 也支持一种类似于代码储存库的联合系统，这个系统也可以实现工程的协作及发布。这个设计可以使一个工程的开发实现从文件系统到用户接口完全独立决策（不受 ROS 限制）。同时，所有的工程都可以被 ROS 的基础工具整合在一起。

MoveIt 简介

Moveit!是目前针对机械臂移动操作的最先进的软件，已在 100 多个机器人上使用。它综合了运动规划、控制、3D 感知、运控学、控制和导航的最新成果，提供了开发先进机器人应用的易用平台，为工业、商业和研发等领域的机器人新产品的设计和集成体用评估提供了一个集成化软件平台。

ROS 图标：

基本的开发环境搭建需要安装机器人操作系统 ROS、MoveIt 以及 git 版本管理器，以下分别介绍其安装方法及流程。

ROS 安装

1.版本选择

ROS 跟 ubuntu 有一一对应的关系，不同版本的 ubuntu 对应不同版本的 ROS，参考网站见下：http://wiki.ros.org/Distributions

如果版本不同，下载将会失败.在这里我们选择的系统为 Ubuntu 16.04, 对应 ROS 版本为 ROS Kinetic Kame

> NOTE: 目前我们不提供 windows 安装 ROS 的任何参考, 若有需要请参考 https://www.ros.org/install/

2.开始安装

2.1.添加源

Ubuntu 本身的软件源列表中没有 ROS 的软件源，所以需要先将 ROS 软件源配置到软件列表仓库中，才能下载 ROS 打开一个控制台终端(快捷键`Ctrl+Alt+T`),输入如下指令：

```
sudo sh -c 'echo "deb http://packages.ros.org/ros/ubuntu $(lsb_release -sc) main" > /e
```

执行结果显示如下（这里会要求输入用户密码，输入安装 Ubuntu 时设置的用户密码即可）：

```
jerry@ubuntu:~$ sudo sh -c 'echo "deb http://packages.ros.org/ros/ubuntu $(lsb_r
elease -sc) main" > /etc/apt/sources.list.d/ros-latest.list'
[sudo] password for jerry:
```

2.2.设置秘钥

配置公网秘钥,这一步是为了让系统确认我们的路径是安全的的，这样下载文件才没有问题，不然下载后会被立刻删掉：

```
sudo apt-key adv --keyserver 'hkp://keyserver.ubuntu.com:80' --recv-key C1CF6E31E6BADE
```

执行结果显示如下：

```
jerry@ubuntu:~$ sudo apt-key adv --keyserver 'hkp://keyserver.ubuntu.com:80' --recv-ke
y C1CF6E31E6BADE8868B172B4F42ED6FBAB17C654
Executing: /tmp/tmp.i2EGDE9fR2/gpg.1.sh --keyserver
hkp://keyserver.ubuntu.com:80
--recv-key
C1CF6E31E6BADE8868B172B4F42ED6FBAB17C654
gpg: requesting key AB17C654 from hkp server keyserver.ubuntu.com
gpg: key AB17C654: "Open Robotics <info@osrfoundation.org>" not changed
gpg: Total number processed: 1
gpg:               unchanged: 1
```

2.3.安装

在加入了新的软件源后，更新软件源列表：

```
sudo apt-get update
```

执行安装 ROS：

```
sudo apt-get install ros-kinetic-desktop-full
```

这里推荐安装完整的 ROS，防止库和依赖的缺失。

安装过程耗时比较长，需要耐心等待

2.4.配置 ROS 环境到系统

rosdep 让你能够轻松地安装被想要编译的源代码，或被某些 ROS 核心组件需要的系统依赖，在终端依次执行以下命令。 初始化 rosdep：

```
sudo rosdep init
```

```
rosdep update
```

初始化完成后，为了避免每次关掉终端窗口后都需要重新生效 ROS 功能路径，我们可以把路径配置到环境变量中，这样在每次打开新的终端时便可自动生效 ROS 功能路径 在终端依次执行以下命令：

Bash

```
echo "source /opt/ros/kinetic/setup.bash" >> ~/.bashrc
source ~/.bashrc
```

Zsh

```
echo "source /opt/ros/kinetic/setup.bash" >> ~/.zshrc
source ~/.zshrc
```

2.5.安装 ROS 额外依赖项

在终端输入以下命令：

```
sudo apt-get install python-rosinstall python-rosinstall-generator python-wstool build
```

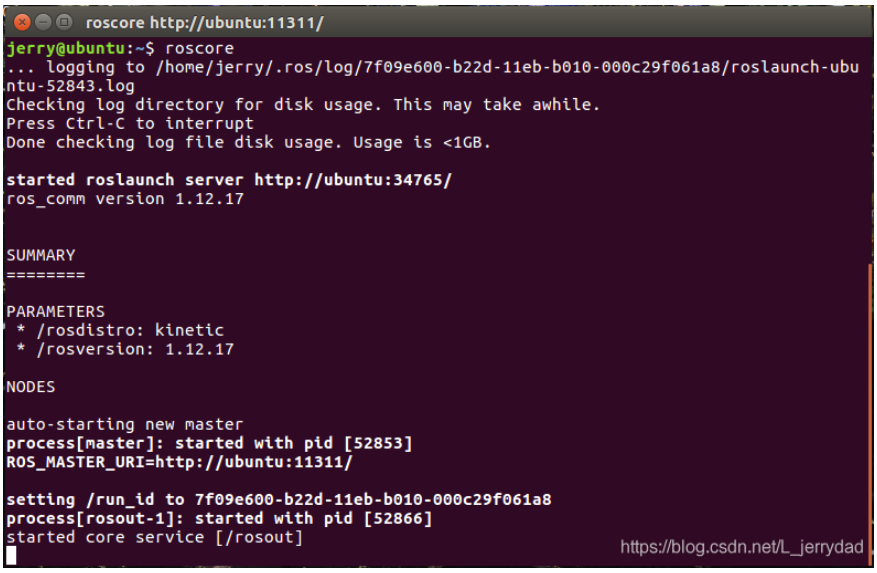
```
sudo apt install ros-kinetic-joint-state-publisher-gui
```

3.验证安装

ROS 系统的启动需要一个 ROS Master，即节点管理器，我们可以在终端输入 roscore 指令来启动 ROS Maste 为了验证 ROS 是否安装成功，在终端执行以下命令：

```
roscore
```

当显示如下界面，则表示 ROS 安装成功



更多更详细的安装指导，可以参考官方的安装指导，网址:
http://wiki.ros.org/ROS/Installation

MoveIt 安装

MoveIt 是 ros 中一系列移动操作的功能包的组成，主要包含运动规划，碰撞检测，运动学，3D 感知，操作控制等功能。

1.更新软件源列表

在终端窗口输入以下命令，以更新软件源列表：

```
sudo apt-get update
```

2.安装 MoveIt

在终端窗口输入以下命令，执行 MoveIt 的安装：

```
sudo apt-get install ros-kinetic-moveit
```

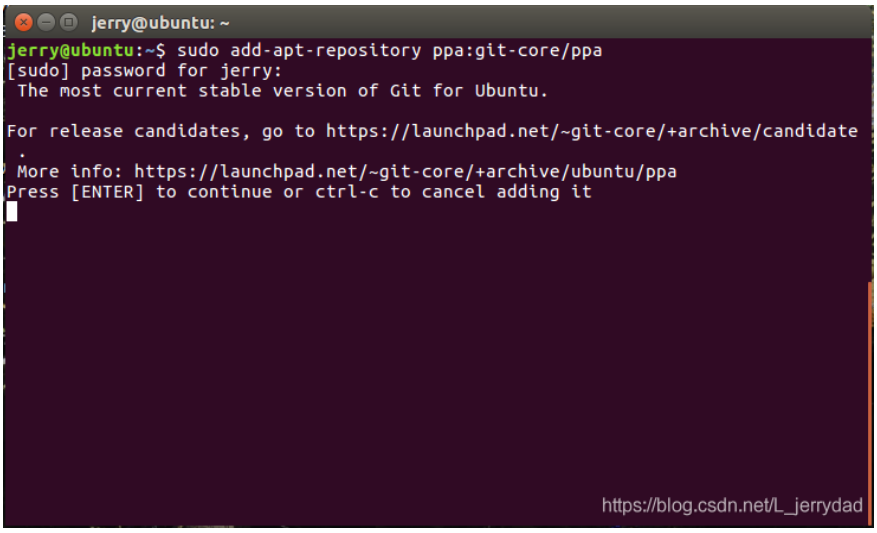
git 安装

1.添加软件源

将 git 安装的软件源添加到 ubuntu 的软件源列表中，在终端窗口输入以下命令：

```
sudo add-apt-repository ppa:git-core/ppa
```

这里需要手动按下回车建**Enter**继续执行



2.更新软件源列表

在终端窗口输入以下命令，以更新软件源列表：

```
sudo apt-get update
```

3.安装 git

在终端窗口输入以下命令，执行 git 的安装：

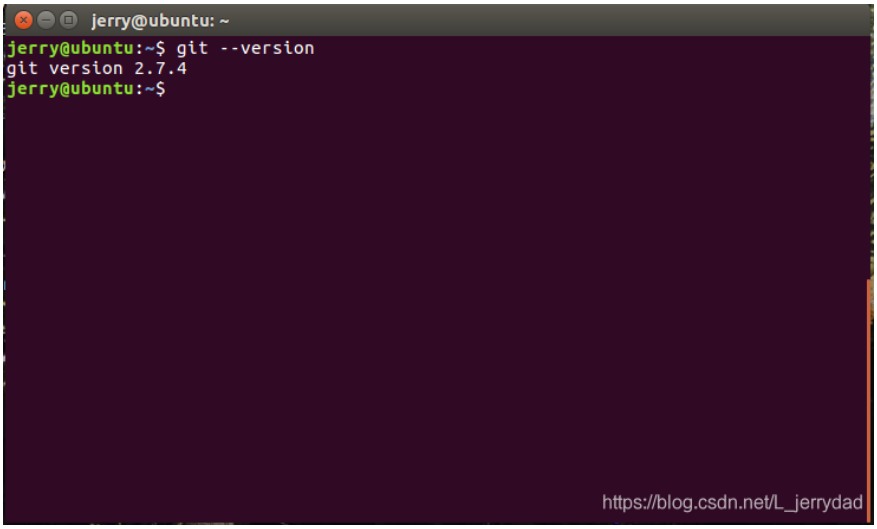
```
sudo apt-get install git
```

4.验证安装

读取 git 版本，在终端窗口输入以下命令：

```
git --version
```

在终端中可以显示 git 版本号，如下，即为安装成功



5.使用

在后续下载 ros 包需要用到git。git 的使用可以参考下面链接：

- https://blog.csdn.net/qq_38410494/article/details/108447093
- https://www.runoob.com/git/git-tutorial.html

介绍

`mycobot_ros` 是 ElephantRobotics 推出的，适配旗下桌面型六轴机械臂 mycobot 的ROS 包。

项目地址：http://github.com/elephantrobotics/mycobot_ros

> NOTE: 该包只支持 `kinetic` 和 `melodic`，对应系统为 `Ubuntu16` 和 `Ubuntu18`。

前提

在安装包之前，请保证拥有 ros 工作空间。

这里我们给出创建工作空间的样例命令：

```
mkdir -p ~/catkin_ws/src
cd ~/catkin_ws/src
catkin_init_workspace
cd ..
catkin_make
```

安装

NOTE：

- 本包依赖于ROS和MoveIT，使用前确保以成功安装ROS和MoveIT。
- 本包与真实机械臂的交互依赖于PythonApi - `pymycobot`
- Api项目地为：https://github.com/elephantrobotics/pymycobot
- 快速安装：`pip install pymycobot --upgrade`
- 安装方式依赖于Git，请确保电脑中已安装Git。

下文将以 `<ros-workspace>` 来代指电脑中ROS的工作空间路径，请确保在执行下面命令时将 `<ros-workspace>` 替换为你本机的真实路径。

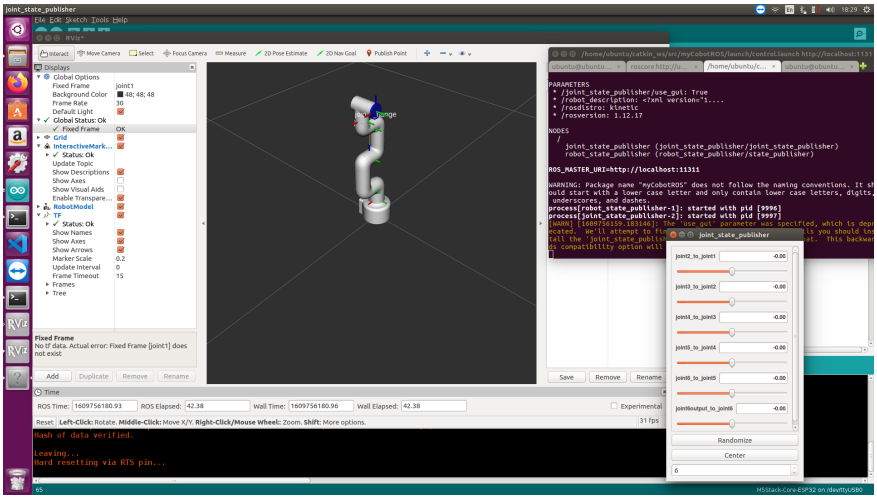
```
cd <ros-workspace>/src
git clone https://github.com/elephantrobotics/mycobot_ros.git
cd ..
catkin_make
source <ros-workspace>/devel/setup.bash
```

滑块控制

打开一个命令行，运行：

```
roslaunch mycobot_ros mycobot_slider.launch
```

它将打开 rviz 和一个滑块组件，你将看到如下画面：



接着你可以通过拖动滑块来控制 rviz 中的模型移动。如果你想让真实的 mycobot 跟着一起运动，需要再打开一个命令行，运行：

```
rosrun mycobot_ros slider_control.py
#或者
rosrun mycobot_ros slider_control.py _port:=/dev/ttyUSB0 _baud:=115200
```

它将实时将角度发布给 mycobot。该脚本支持设置端口号和波特率，默认为 `"/dev/ttyUSB0"` 和 `115200` 。

模型跟随

除了上面的控制，我们也可以让模型跟随真实的机械臂运动。打开一个命令行运行：

```
rosrun mycobot_ros follow_display.py
#或者
rosrun mycobot_ros follow_display.py _port:=/dev/ttyUSB0 _baud:=115200
```

它将发布真实机械臂的角度。该脚本支持设置端口号和波特率，默认为 `"/dev/ttyUSB0"` 和 `115200` 。

然后打开另一个命令行，运行：

```
roslaunch mycobot_ros mycobot_follow.launch
```

它将打开 rviz 展示模型跟随效果。

键盘控制

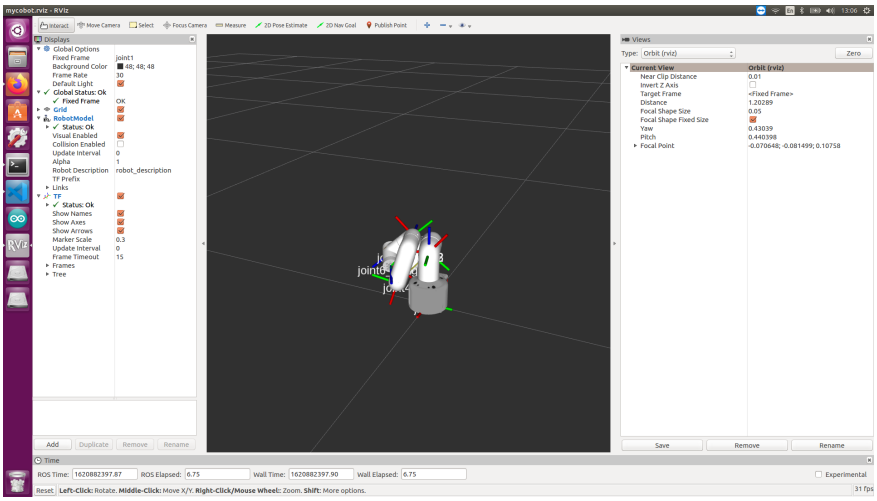
在 `mycobot_ros` 的包中添加了键盘控制的功能，并在 rviz 中实时同步。本功能依赖 pythonApi，所以确保与真实机械臂连接。

打开一个命令行，运行：

```
roslaunch mycobot_ros mycobot_teleop_keyboard.launch
#或者
roslaunch mycobot_ros mycobot_teleop_keyboard.launch port:=/dev/ttyUSB0 baud:=115200
```

由于需要于真实机械臂通信，该 launch 支持设置端口号和波特率，默认为 `"/dev/ttyUSB0"` 和 `115200` 。

运行效果如下：



命令行中将会输出 mycobot 信息，如下：

```
SUMMARY
========

PARAMETERS
 * /mycobot_services/baud: 115200
 * /mycobot_services/port: /dev/ttyUSB0
 * /robot_description: <?xml version="1....
 * /rosdistro: kinetic
 * /rosversion: 1.12.17

NODES
  /
    mycobot_services (mycobot_ros/mycobot_services.py)
    real_listener (mycobot_ros/listen_real.py)
    robot_state_publisher (robot_state_publisher/state_publisher)
    rviz (rviz/rviz)

auto-starting new master
process[master]: started with pid [1333]
ROS_MASTER_URI=http://localhost:11311

setting /run_id to f977b3f4-b3a9-11eb-b0c8-d0c63728b379
process[rosout-1]: started with pid [1349]
started core service [/rosout]
process[robot_state_publisher-2]: started with pid [1357]
process[rviz-3]: started with pid [1367]
process[mycobot_services-4]: started with pid [1380]
process[real_listener-5]: started with pid [1395]
[INFO] [1620882819.196217]: start ...
[INFO] [1620882819.205050]: /dev/ttyUSB0,115200

MyCobot Status
-------------------------------
Joint Limit:
    joint 1: -170 ~ +170
    joint 2: -170 ~ +170
    joint 3: -170 ~ +170
    joint 4: -170 ~ +170
    joint 5: -170 ~ +170
    joint 6: -180 ~ +180

Connect Status: True

Servo Infomation: all connected

Servo Temperature: unknown

Atom Version: unknown

[INFO] [1620882819.435778]: ready
```

接着，打开另一个命令行，运行：

```
rosrun mycobot_ros teleop_keyboard.py
#或者
rosrun mycobot_ros teleop_keyboard.py _speed:=70
```

你会在命令行中看到如下输出：

```
Mycobot Teleop Keyboard Controller
---------------------------
Movimg options(control coordinations [x,y,z,rx,ry,rz]):
              w(x+)

    a(y-)     s(x-)     d(y+)

    z(z-) x(z+)

u(rx+)   i(ry+)   o(rz+)
j(rx-)   k(ry-)   l(rz-)

Gripper control:
    g - open
    h - close

Other:
    1 - Go to init pose
    2 - Go to home pose
    3 - Resave home pose
    q - Quit

currently:      speed: 50       change percent 5
```

该脚本支持的参数：

- `_speed`：机械臂移动速度。
- `_change_percent`：移动距离百分比。

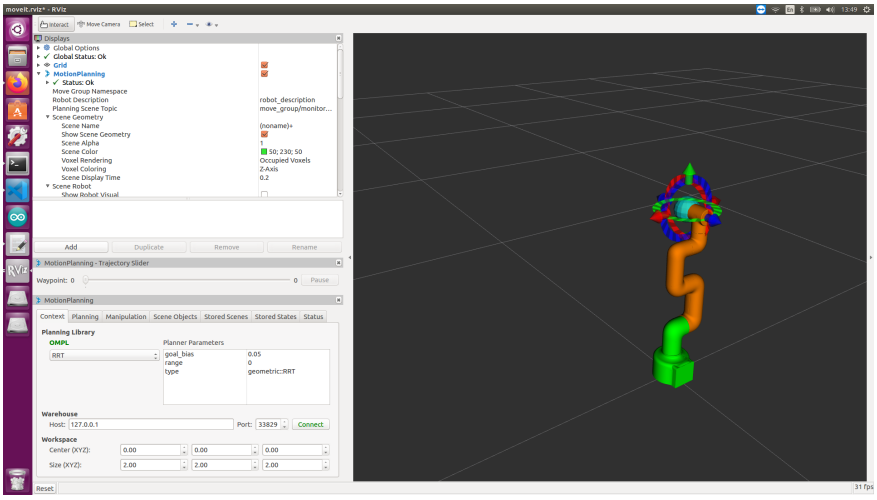
MoveIt

`mycobot_ros` 现已集成了 MoveIt 部分。

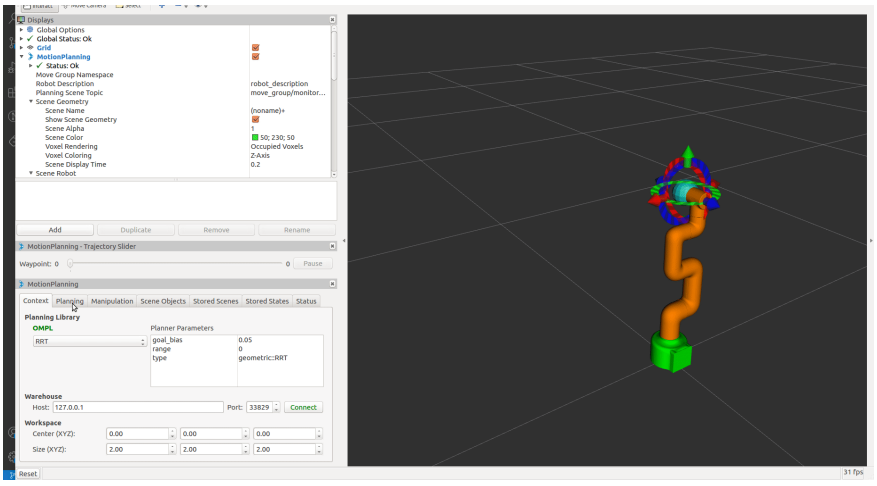
打开命令行运行：

```
roslaunch mycobot_ros mycobot_moveit.launch
```

运行效果如下：



可以计划并执行，演示效果：



如果需要让真实的机械臂同步执行计划，需要再打开一个命令行，运行：

```
rosrun mycobot_ros sync_plan.py
#或者
rosrun mycobot_ros sync_plan.py _port:=/dev/ttyUSB0
```

该脚本支持设置端口号和波特率，默认为 `"/dev/ttyUSB0"` 和 `115200` 。

> 更多请期待...

RoboFlow

1 RoboFlow操作系统简介

RoboFlow操作系统是**大象协作型机器人**的操作系统，提供了人机交互界面，便于操作人员与大象机器人进行交互，正确使用大象机器人。也就是说，用户在使用机器人时，**大多数时候**都是通过使用RoboFlow操作系统实现。

例如，由于RoboFlow操作系统在示教器中运行，用户可以利用示教器这个载体，进行手动操作机器人、编程和其他操作。也可以利用操作系统OS进行机器人系统与其他机器人或设备的通信。总而言之，凭借着**界面友好、功能丰富等优点**，RoboFlow操作系统的出现，让用户开始使用大象机器人时更容易上手，从而使得人人都可以成为机器人的指挥官。

2 主要界面介绍

2.1用户登录界面

当启动控制器电源，并且按下示教器上的系统启动按键后，即可进入登录页面。RoboFlow操作系统的登录界面如图2-1所示。

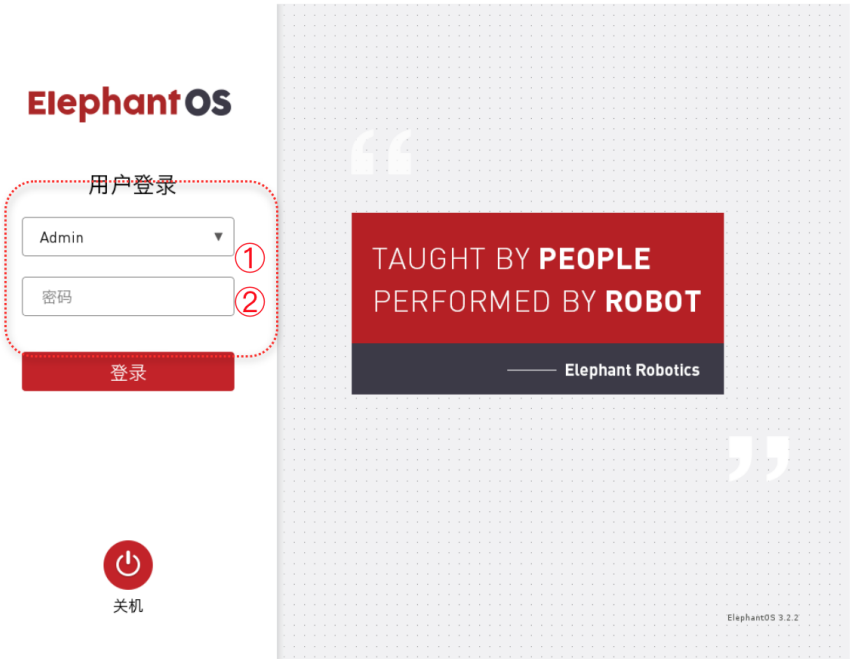


图2- 1登录界面

正如登录页面中显示的“TAUGHT BY PEOPLE,PERFORMED BY ROBOT”，这是大象机器人一贯秉持的让操作员成为机器人的指挥官的理念。让机器人代替人进行一些简单但重复性强的工作、工作环境恶劣的工作、人无法完成得非常好的工作（例如操作精度要求非常高的场景）。

RoboFlow操作系统的登录用户等级有两种，一种是管理员，另一种是操作员。管理员拥有最高权限，可以进行所有操作、编程和设置；而操作员只能加载和运行已有程序，检查统计的数据信息。

管理员可以在设置中添加和修改多个账户，包括操作员账户。

点击“关闭”按键，可以关闭RoboFlow操作系统，继而关闭电源，则完成了机器人系统关机。

2.2主菜单

当登录成功后，会转入主菜单页面。RoboFlow操作系统的主菜单如图2-2所示。

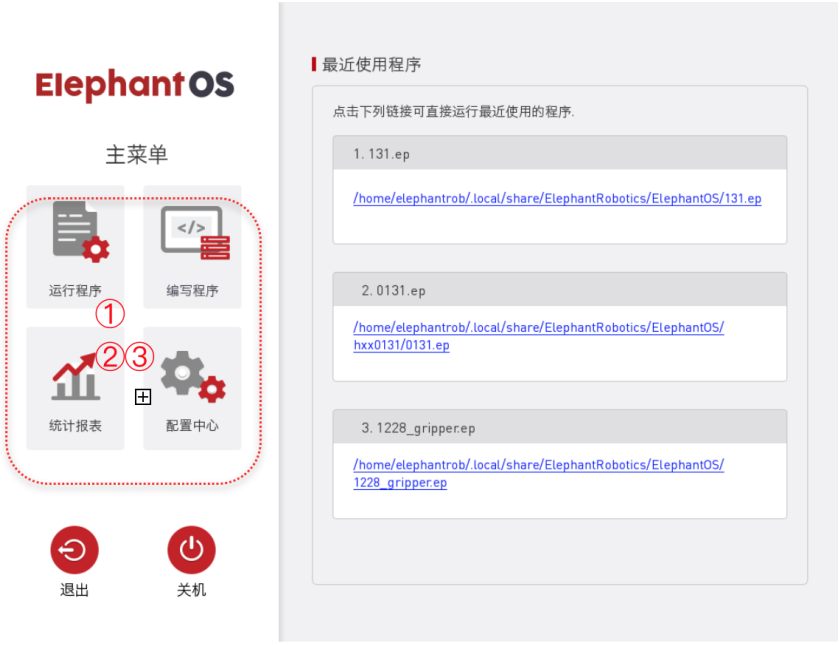


图2- 2主菜单

在主菜单的左侧，提供了四个不同的功能选项：

- 运行程序
  - 直接加载一个已存在的程序，控制程序运行。在此窗口中，不允许用户进行程序编辑工作，只能控制程序运行（如控制程序运行、暂停、停止）。同时还可以查看程序运行过程中的日志和其他相关信息。
- 编写程序
  - 用户可以在此窗口中选择加载一个已有程序进行修改，也可以选择新建一个空白程序进行编辑。 该窗口是用户最经常使用的功能窗口，除了编程，还能进行其他操作，如使用“快速移动”功能手动操作机器人、强制控制IO信号、新建变量等。
- 统计报表
  - 在此窗口中，用户不仅可以查看系统现有运行数据，还能查看之前保存的相关信息。
- 配置中心
  - 在此窗口中，用户可以对机器人进行基本的设置。如机器人打开、机器人关闭、账号管理、默认程序设置等。

除了这四个主要选项，在主菜单的右侧窗口中，用户可以看到和打开最近运行过的程序文件。便于用户快速找到最近运行的程序，并控制程序运行。 点击“关闭”按键，可以关闭RoboFlow操作系统；点击“退出”按键，可以退出登录。

2.3 运行程序窗口

如果用户在主菜单中选择了“运行程序”，将会进入运行程序窗口。RoboFlow操作系统的运行程序窗口如图2-3所示。

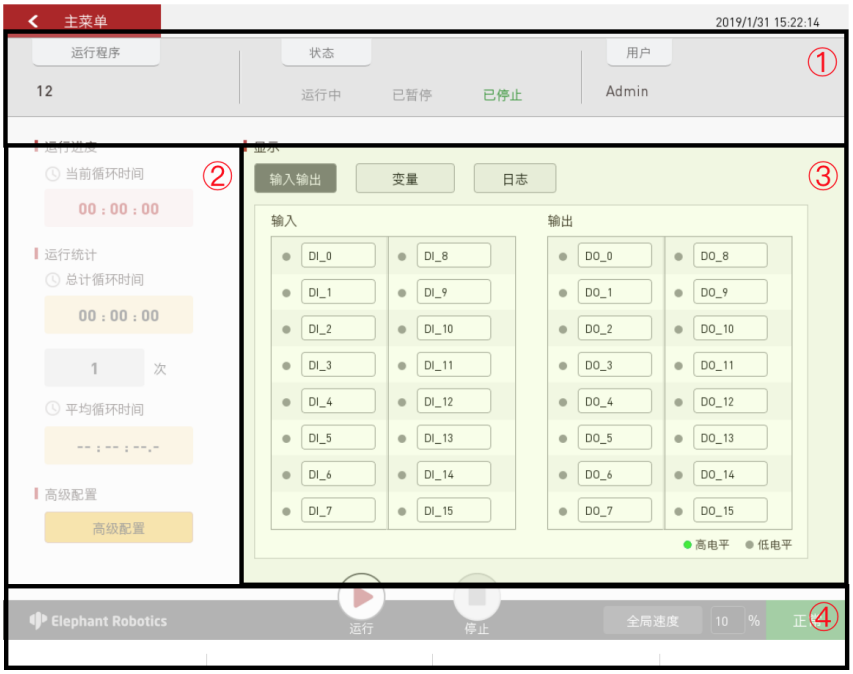


图2- 3程序编辑窗口选项

用户通过加载需要运行的程序，进入程序运行程序窗口中， 在此窗口，用户可以：获取当前（准备）运行的程序的基本信息，包括程序名称、运行状 态、用户类型；了解到当前运行程序的统计信息，例如运行总次数和节拍等； 通过显示窗口读取当前运行程序的相关信息，例如IO、变量、日志 等； 最重要的是，运行程序窗口是用户将已经调试完成的程序加载运行 的渠道，使用程序运行控制栏，可以控制程序运行、暂停、停止，还可以设置运行速度。

2.4编写程序窗口

如图2-4所示，如果用户在主菜单中选择“编写程序”之后，右侧窗口会出现两个选项，第一个是创建程序（可选空白程序或模板程序），第二个是加载程序。根据需要选择其一，均可进入如图2-5所示的程序编辑页面。

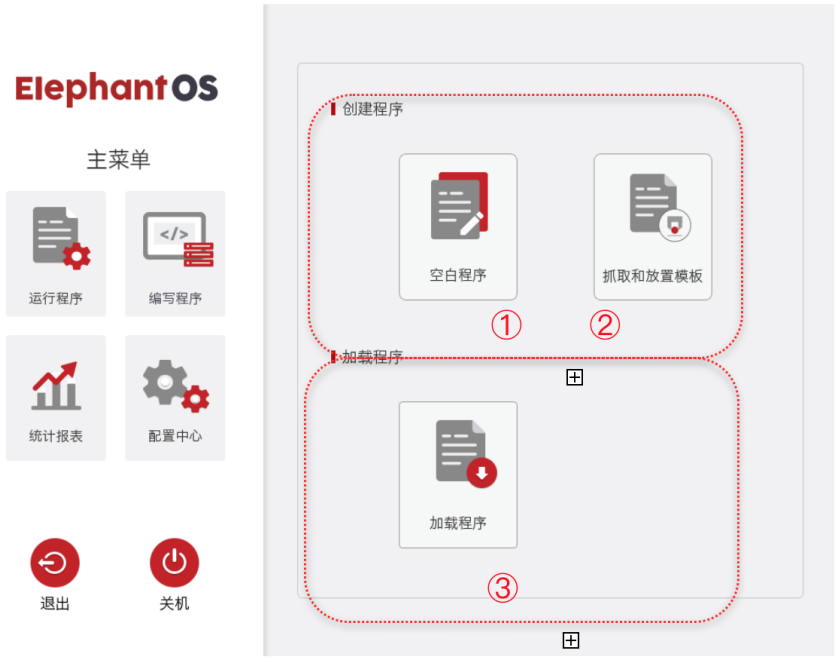


图2- 4程序编辑窗口选项

图2- 5程序编辑界面

首次进入程序编辑页面时，用户看到的就是如图2-5所示的初始页面。在此页面中，提供了常用工具，初始化组和文件管理功能。其中，初始化组的作用是便于用户设置在程序一开始运行并且只运行一次的程序内容，例如设置机器人开始工作的初始点位、IO状态等。

文件管理功能为用户提供了文件管理的途径，用户可以在这里对程序文件进行管理，并且可以将其拷贝到U盘中，也可以从U盘中拷贝到系统内存中。如果用户在编程过程中想回到初始页面，点击“返回”即可。

进入程序编辑页面后，可以对文件进行保存、新建、另存为等操作，也可以对程序本身进行编辑操作，可以示教点位、新建变量、查看IO、查看日志、进行重要参数设置，还可以添加功能指令、调试程序等。 程序编辑页面一共分为四个部分：

功能栏 如图2-6所示，功能栏有7个子选项，分为两大类，一类是程序编辑工具栏，另一类是功能编辑栏。

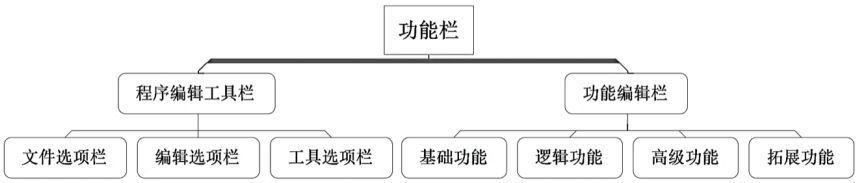


图2- 6 文件选项栏

程序编辑工具栏：程序编辑工具栏包括了文件选项栏、编辑选项栏、工具选项栏。
文件：如图2-7所示，可以对程序文件进行编辑操作。分别有保存、另存为、新建、加载、重命名、退出这几个操作选项。

图2- 7 文件选项栏

编辑：如图2-8所示，可以对程序文件中具体的指令内容进行编辑操作，分别有剪切、复制、粘贴、删除、禁用、删除所有、重做、撤销这几个操作选项。

图2- 8 编辑选项栏

工具：如图2-9所示即为快捷工具栏，在编辑机器人程序的时候，经常需要使用其他工具操作机器人，工具选项栏就提供了程序编辑时常用的工具。提供的工具包括：快速移动、安装、输入输出、变量、日志、基础设置。例如，用户在编辑运动指令时，需要手动操作机器人到某一工作位置并示教该点，这时就可以选择工具栏中的“快速移动”工具，手动操作机器人运动到该位置。

图2- 9 工具选项栏

功能编辑栏 RoboFlow操作系统提供了丰富的功能，使得用户通过简易的操作就能够完成复杂的功能。操作简易，但是功能却不简单，从而缩短了工人学习编程的时间，高效完成编程任务。功能编辑栏包括了基础功能、逻辑功能、高级功能、拓展功能。

基础功能：如图2-10所示，基础功能包括路点、夹爪、等待、设置、组合，是用户常用的一些基础功能。

图2- 10 基础功能

- 路点：用户可以通过新建路点→手动操作机器人使机器人移动到目标点→保存当前点→运行程序这一系列的操作，完成控制机器人运动到目标点的操作，如果新建多个路点，那么运行程序时机器人的运动将会形成一段轨迹。

- **夹爪**：用户可以利用该功能对末端执行器进行设置，例如使其夹持工件或松开工件。
- **等待**：用户可以利用该功能进行延时，或者等待信号、条件等。
- **设置**：用户可以利用该功能对输入输出信号和自定义条件进行设置。
- **组合**：用户可以利用该功能实现对组内程序进行编辑。
- **逻辑功能**：如图2-11所示，逻辑功能包括循环、条件判断、子程序、线程、程序控制、条件选择，完成程序运行流程控制。

图2- 11 逻辑功能

- **循环**：用户可以利用该功能设置某一程序段循环运行多次。
- **条件判断**：用户可以利用该功能进行条件判断，例如对某一输入信号的判断。
- **子程序**：用户可以利用该功能调用子程序。
- **线程**：用户可以利用该功能实现机器人多线程控制。
- **程序控制**：用户可以利用该功能控制程序暂停、停止、重启，并弹窗显示相应提示信息。
- **条件选择**：用户可以利用该功能进行条件选择，根据选择对象的值确定执行的内容。
- **高级功能**：如图2-12所示，高级功能包括托盘、给变量赋值、脚本、弹窗、发送器，是完成较复杂操作的功能。

图2- 12 高级功能

- **托盘**：用户可以利用该功能实现机器人执行有规律点位的运动，例如实现托盘内工件的搬运、码垛等，还能实现机器人依次执行固定但无规律的集合点运动。
- **给变量赋值**：用户可以利用该功能实现对某一变量进行赋值的操作。
- **脚本**：利用脚本功能，用户在使用大象机器人时，可以通过其他常用功能实现简单任务，还可以使用脚本编程完成更为复杂的任务。
- **弹窗**：用户可以利用该功能自定义弹窗，显示相关信息，帮助操作员分析当前机器人运行程序的状态。
- **发送器**：用户可以利用该功能实现大象机器人与其他设备之间的TCP/IP通信。
- **拓展功能**：根据不同应用场景，RoboFlow操作系统提供了一些拓展功能，甚至根据用户提出的重要应用场景进行功能定制。

图2- 13 拓展功能

程序显示窗口 程序编辑页面左侧有一个如图2-14所示的程序显示窗口，上方是当前打开程序文件的名称，下方是程序树，记录了具体指令及相关信息。


NewProgram1
程序
组合_1
路点_1: Abs, MoveJ, v50, b0,
设置_1
等待_1: 0.5s
路点_2: Abs, MoveJ, v50, b0,
设置_2
等待_2: 0.5s
路点_3: Shared, MoveJ, v50,
组合_2
路点_4: Abs, MoveJ, v50, b0,
设置_3
等待_3: 0.5s
路点_5: Abs, MoveJ, v50, b0,
设置_4
等待_4: 0.5s

图2-14程序显示窗口

程序编辑页面右侧有一个如图2-15所示的功能编辑窗口，显示了功能指令的具体内容。

图2- 15功能显示窗口

用户可以在此窗口中对功能指令进行具体的设置。还提供了快速控制和当前指令重命名、删除、禁用等功能。 程序编辑页面下方有一个如图2-16所示的程序运行控制栏，用户调试程序时，可以对程序进行运行、暂停、停止、限制运行速度等操作。

图2- 16程序运行控制栏

2.5统计报表

用户在使用大象机器人时，除了可以编程控制机器人完成相应任务，还可以在统计报表窗口中获取一些有参考价值的统计数据，便于分析和统计。 统计报表窗口分为四个子窗口。 如图2-17所示，常规类统计了总运行的时间、当前活跃程序的数量、当前活跃程序的具体信息。

图2- 17 常规类统计

如图2-18所示，程序类统计了不同程序的总运行时间、运行次数等信息。

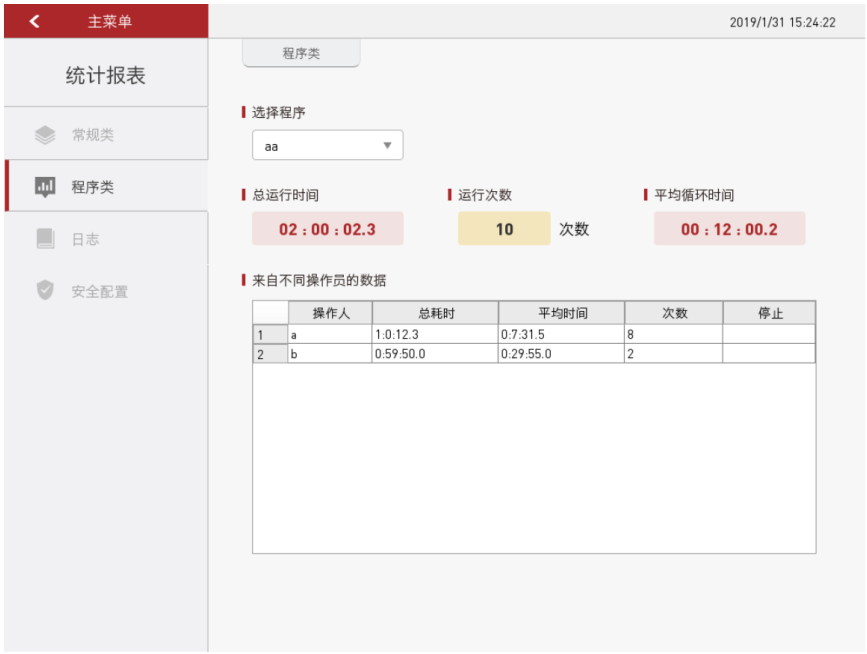


图2- 18程序类统计

如图2-19所示，日志列出了用户在使用RoboFlow操作系统的过程中系统记录的普通信息、警告信息、错误信息，这些信息有助于用户判断在操作RoboFlow操作系统的过程中系统有哪些改变和反馈。 特别是错误信息，能够高效地帮助用户快速定位导致错误的可能原因，从而根据错误信息解决问题，恢复正常使用。

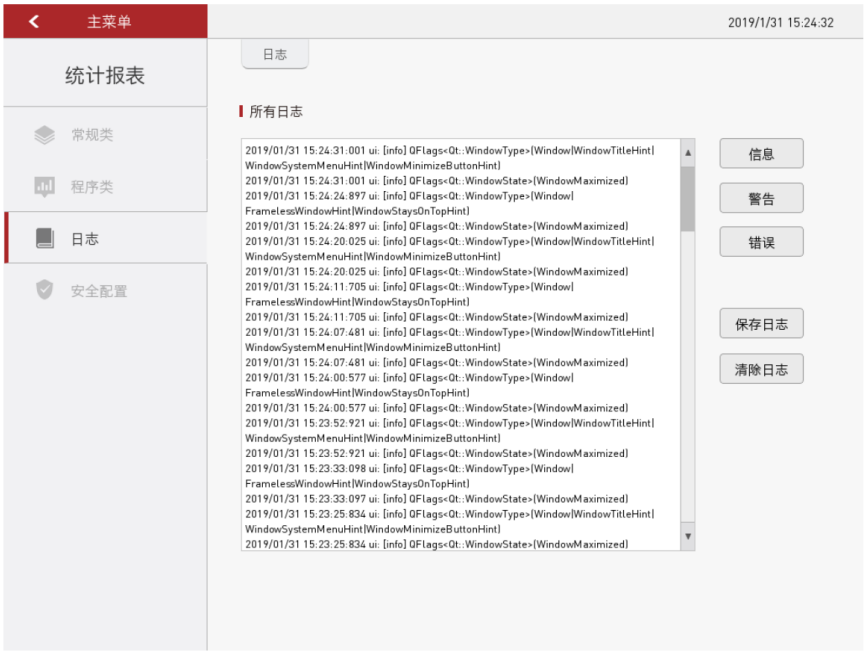


图2- 19日志统计

如图2-20所示，安全类统计数据可以帮助用户统计安全相关的信息，例如碰撞信息、停止次数等。

图2- 20安全配置统计

2.6配置中心

在配置中心，用户可以对机器人进行相关配置，例如给机器人上电、关闭机器人、设置负载、时间、网络等。 初始化 如图2-21所示是初始化配置页面。 在需要机器人运动时，用户需要进入配置中心→初始化启动机器人，也可以关闭机器人。在初始化页面中，还可以设置负载和安装，这两项是进行其他操作前的重要配置内容，如配置错误可能引起预料外的情况发生。

图2- 21初始化

默认程序 如图2-22所示是默认程序设置页面。

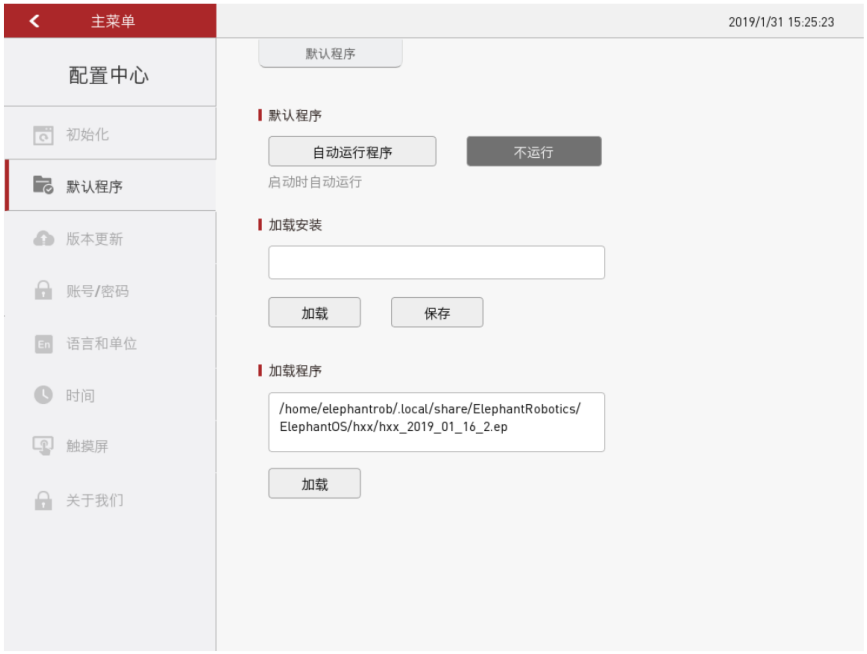


图2- 22默认程序设置

此功能支持用户设置一个默认运行的程序，只要系统启动，机器人就直接进入运行程序窗口，可以开始运行程序，执行相应动作完成指定任务。 如果用户不希望系统启动的同时启动程序开始运行，也可以选择不运行。

版本更新 如图2-23所示是版本更新设置页面。

图2- 23版本更新

此页面支持用户使用两种方式更新RoboFlow操作系统系统，一是本地文件更新，二是联网更新。 账号/密码 如图2-24所示是账号管理页面。

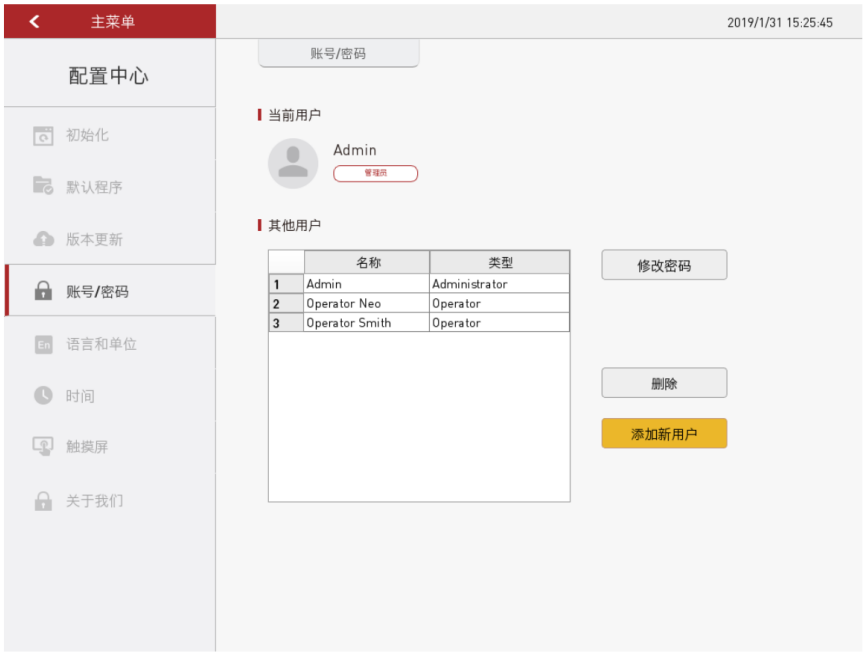


图2- 24账号/密码

用户可以在此页面中增加新用户、删除失效用户，或者修改密码。在此页面中，用户能够了解到所有的账号信息。 语言和单位 如图2-25所示是语言和单位设置页面。目前RoboFlow操作系统支持中英文和公制单位，其他语言和单位正在增加中，敬请期待！

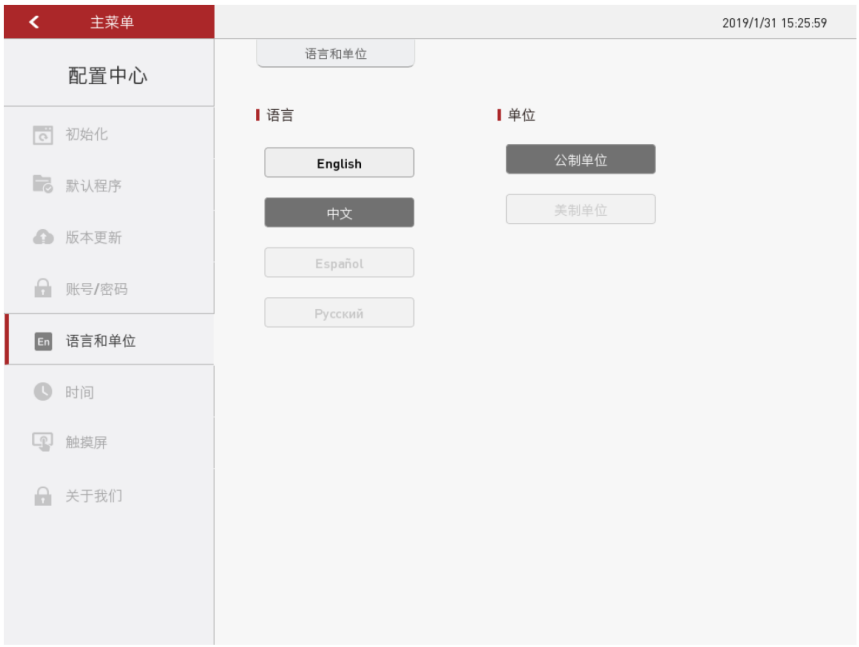


图2- 25语言和单位

时间 如图2-26所示是时间设置页面。

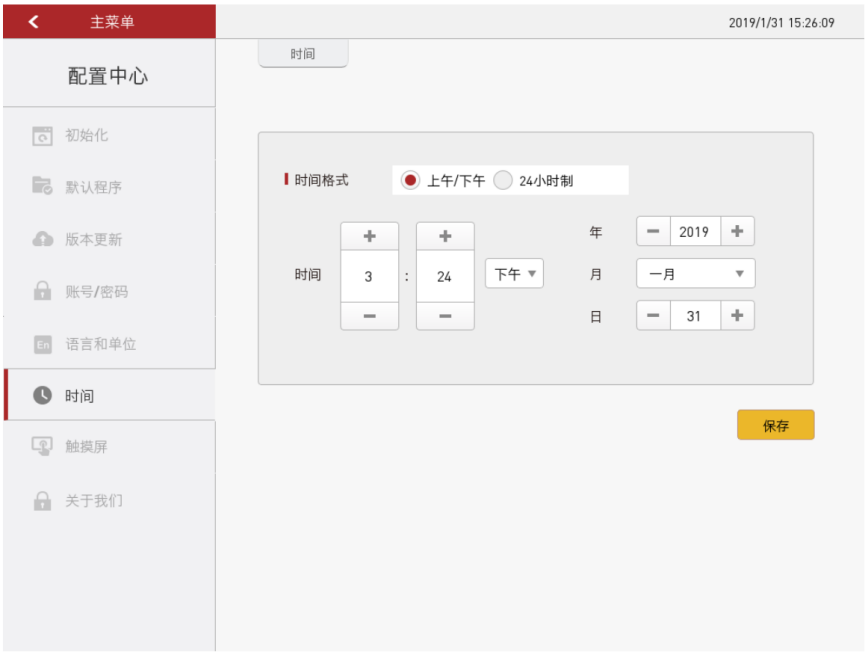


图2- 26时间

用户可以在当前页面设置系统时间。如不勾选“24小时制”，时间显示格式默认为12小时制。 触摸屏校准 如图2-27所示是触摸屏校准说明。点击“Start to Calibrate”即可进入校准界面。校准界面会依次出现四个圆圈，次序如图中所示。用户需要使用触摸笔依次点击圆圈中心，每点击一次，就会出现下一个圆圈，直至四个圆圈都出现。此时将会出现一个弹窗说明校准工作完成，确认弹窗之后可以退出校准界面。如若校准超时或步骤出错，也会出现弹窗提示校准失败。此时可以确认退出校准界面，回到图2-27的页面重新进行校准。

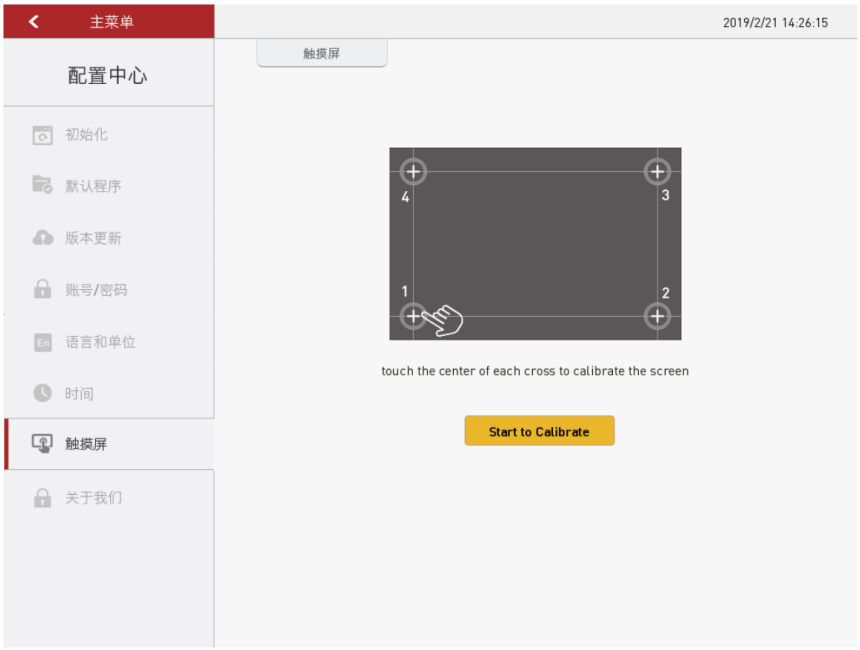


图2- 27 屏幕校准

关于我们 如图2-28所示是关于我们页面。

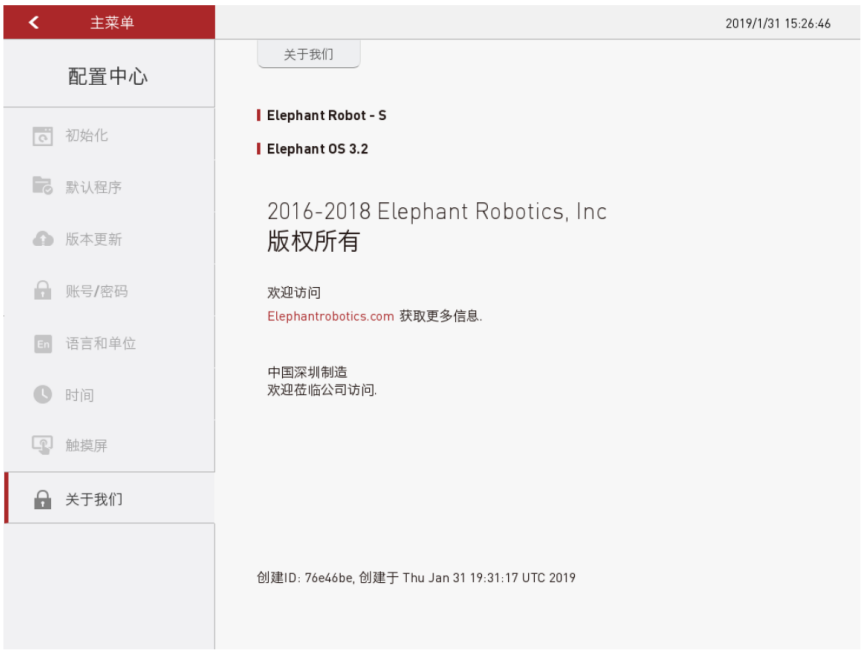


图2- 28关于我们

此页面显示了RoboFlow操作系统操作系统的基本信息，例如配套使用的机器人型号是Elephant S系列，版本信息等。 更多信息，请访问官网 https://www.elephantrobotics.cn。

3 常用工具介绍

3.1快速移动

快速移动工具是用户快速手动操作机器人时使用频率较高的工具。因此，每个用户必须对快速移动的使用方法非常熟悉，错误的操作可能会导致机器人及其周边设备损伤，甚至人员受伤。 如图3-1所示，快速移动工具主要由11个部分组成，下面将对各个部分进行介绍。

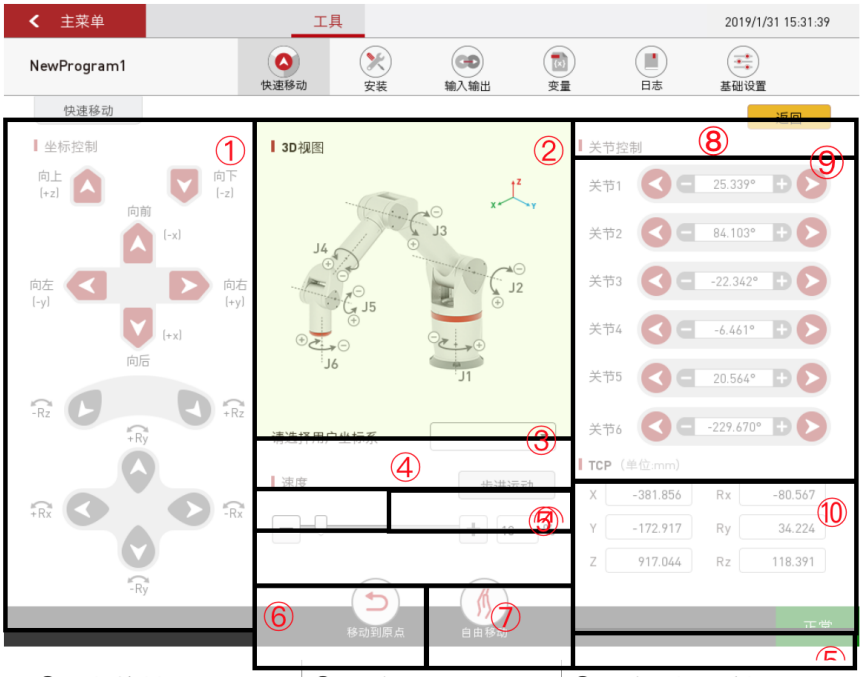


图3- 1 快速移动工具

笛卡尔坐标系运动控制模式 如图3-2所示，过定点O，作三条互相垂直的数轴，它们都以O为原点且一般具有相同的长度单位。这三条轴分别叫做x轴(横轴）、y轴(纵轴)、z轴(竖轴)；统称坐标轴。通常把x轴和y轴配置在水平面上，而z轴则是铅垂线；它们的正方向要符合**右手规则**，即以右手握住z轴，当右手的四指从正向x轴以π/2角度转向正向y轴时，大拇指的指向就是z轴的正向，这样的三条坐标轴就组成了一个空间直角坐标系，点O叫做坐标原点。这样就构成了一个笛卡尔坐标。

在三维笛卡尔坐标系中，三个平面，xy-平面，yz-平面，xz-平面，将三维空间分成了八个部分，称为卦限(octant) 空。第I卦限的每一个点的三个坐标都是正值。

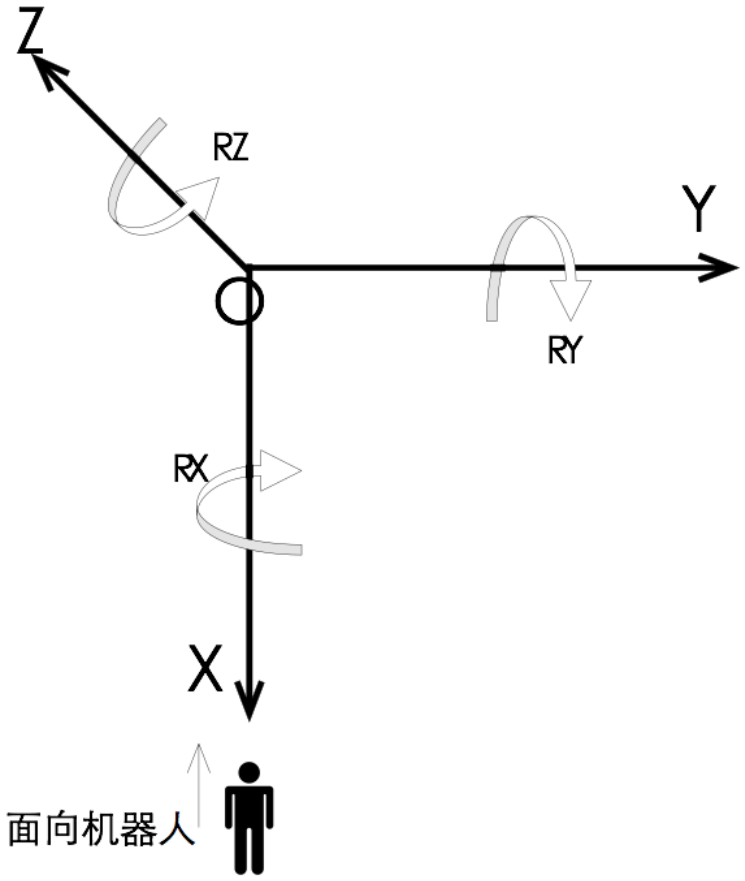


图3- 2笛卡尔坐标系方向标注图

如图3-3所示，通过点击对应笛卡尔坐标系方向的按键就可以控制机器人沿着笛卡尔坐标系的方向运动。

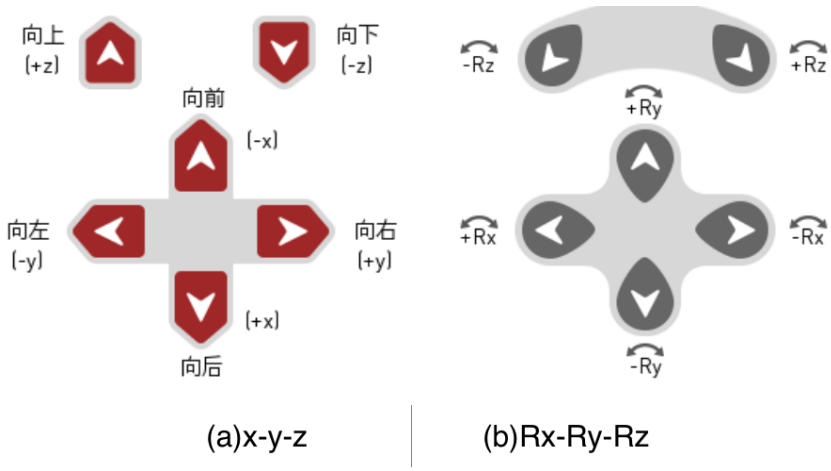


图3- 3 笛卡尔坐标系运动控制模式按键

机器人3D图 该窗口标注了机器人6个关节的运动方向。

- 用户坐标系选择
- 步进运动切换

手动操作机器人的运动模式主要有两种。

- **连续运动模**：用户按下运动控制按键并允许机器人移动，直到用户松开按键，机器人停止。例如按下+x方向运动控制按键，需要长按该按键，按下运动控制按键的时间长短决定了机器人在+x方向运动距离的大小。
- **步进运动模式**：手动操作机器人进行步进运动，点击“步进运动”后，打开如图3-4所示的步距设置窗口，在此窗口选择步距，点击目标控制方向按键，每点击一次，机器人就走一步。例如，选择1mm的步距，点击X正方向方向运动控制按键，每点击一次按键，机器人就往+x方向运动1mm。

图3- 4步进运动步距设置窗口

速度 如图3-5所示，在这里可以设置手动操作机器人的控制速度。可以从0设置到100%。

图3- 5速度设置窗口

移动到原点 选中如图3-6所示图标，可以控制机器人回归原点位姿。

图3- 6移动到原点

自由移动 选中如图3-7所示图标，可以切换到拖动示教的模式。

图3- 7自由移动

返回 点击如图3-8所示图标，可以返回到编程操作窗口。

图3- 8返回

关节控制 串联机器人是一种开式运动链机器人，它是由一系列连杆通过转动关节或移动关节串联形成的。大象协作型机器人属于6轴串联型机器人，通过采用驱动器驱动6个关节的运动从而带动连杆的相对运动，使末端操作器达到合适的位姿。如图3-9所示的关节控制窗口，提供了操作人员在使用示教器手动操作机器人，控制机器人进行关节运动时所用的按键。每个关节的控制按键分为2个方向，可以看

到各轴的角度数据。

图3- 9关节运动模式控制窗口

坐标位置 如图3-10所示，该窗口显示的是对应坐标控制的坐标位置。

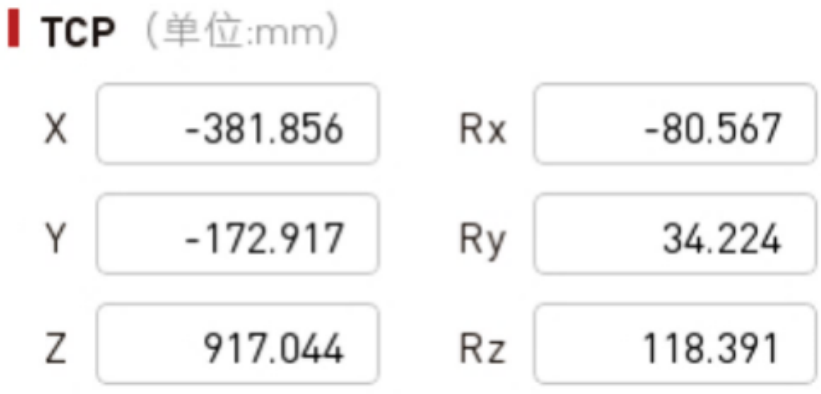


图3- 10坐标位置显示窗口

状态显示按键：该按键有两种状态，“正常”（显示绿色）和“复位”（显示红色），当显示正常时，说明机器人正常工作；当显示复位时，说明机器人异常，需要点击该按键进行复位。

3.2安装

如图3-11所示，安装工具内部有3个子菜单。用来实现大象机器人的**加载/保存安装配置、安全配置、网络配置**。 加载/保存：如图3-11所示，在本页面中用户可以选择保存或加载安装配置。

图3- 11加载/保存安装

安全配置：如图3-12所示，设置大象机器人的力矩限制和制动控制。

图3- 12安全配置

网络配置：如图3-13所示，配置以太网通信的IP地址、端口号。

图3- 13网络

3.3 输入输出配置

系统一共有16个数字输入信号和16个数字输出信号。如图3-14所示，可以在本窗口进行输入输出信号的配置和监视，还可以对输出信号进行强制输出。可以保存和加载IO配置文件。如图3-15所示，是与图3-14所示页面对应的输入输出接口说明图。

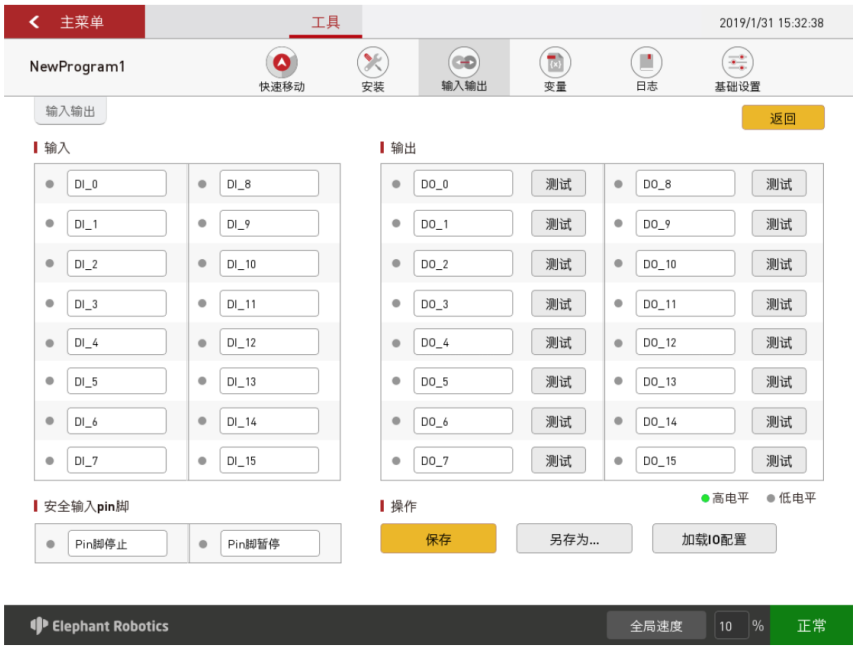


图3- 14输入输出配置

315 图3- 15输入输出接口说明图

需要注意的是，输入公共端需要连接24V电源，可以根据公共端配置（硬件接线确定连接24V或者0V）确定输入是高电平有效还是低电平有效。如图3-16所示，当公共端连接24V时，一旦有外部设备输入0V，则该输入信号为High的状态，否则是

Low状态；反之亦然。

316 图3- 16输入信号应用说明图

如图3-17所示，输出端在没有输出时是24V，一旦打开输出（即输出为High状态），则输出端为0V。

317 图3- 17输出信号应用说明图

3.4变量

如图3-18所示，在变量编辑窗口中，可以进行新增、编辑、删除变量的操作。

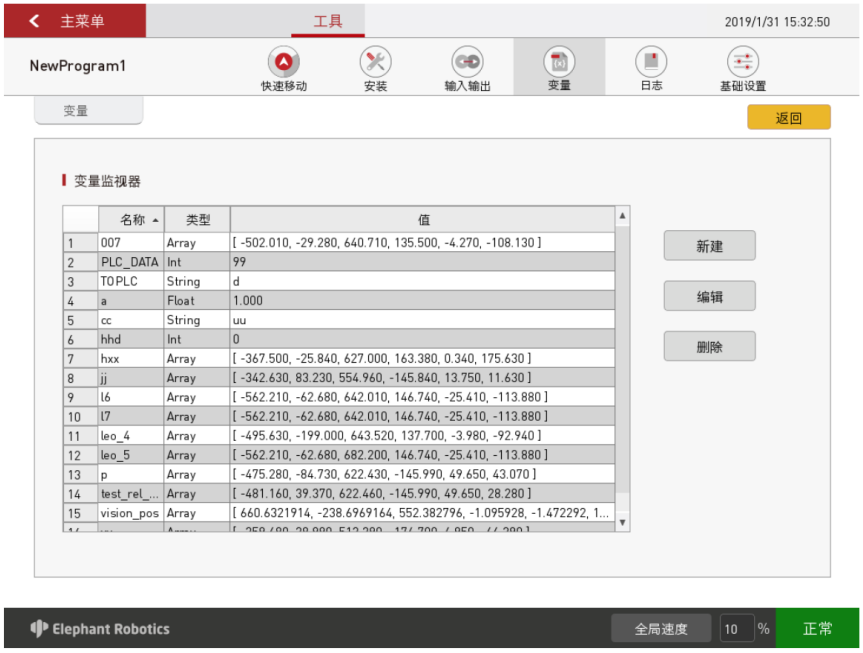


图3- 18变量编辑

如图3-19所示，可编辑变量类型一共5种。包括字符串变量、位姿变量、浮点数变量、整数变量、布尔变量。在此页面中，可以编辑变量名称和初始值。

图3- 19新建变量界面

3.5日志

如图3-20所示，在运行日志窗口中可以查看机器人运行状态、错误信息、报警信息等相关信息。点击“信息”“警告”“错误”按键可以分类查看对应日志。 用户可以将日志保存到本地文件夹，日志文件是系统运行情况的记录，能够帮助用户对系统有一个比较清晰的了解，而且在排查错误时也有所助益。

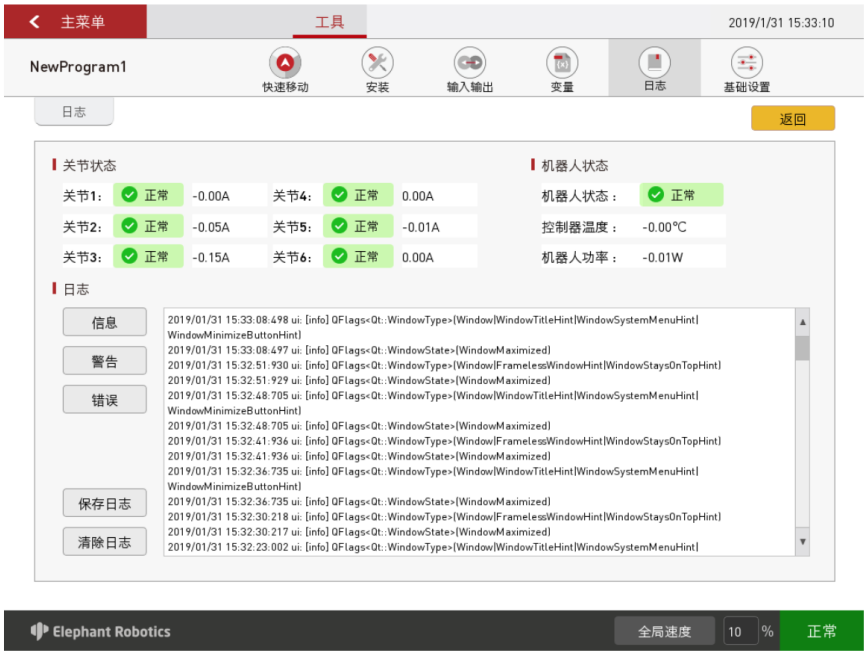


图3- 20运行日志

3.6基础设置

如图3-21所示，基础设置页面提供了常用设置通道，让用户在编写程序时，即使不离开编程窗口也能够快速设置一些功能，例如自由移动相关参数设置等。

图3- 21基础设置

4 功能指令

4.1 基础功能

4.1.1路点

路点有四种类型：**绝对点、相对点、共享点、变量**。这四种类型是并列关系，一条路点指令下，只能选择其一。

绝对点：绝对点是对机器人实际位姿的描述。

- 也就是说，只要机器人记录了绝对点，下一次再执行该指令时，无论机器人在什么位姿下（其他设置不变情况下），都会再现原来示教的绝对点的位姿。 如图4-1所示是绝对点的具体配置页面。

图4- 1绝对点

路点坐标

- 如图4-2所示，绝对点的表示一共有两种格式，分别是笛卡尔坐标系坐标值和关节角度。其中，笛卡尔坐标系坐标值记录了机器人TCP相对于基坐标系的位置和姿态（单位为mm，毫米），关节角度则是直接对应各轴的实际角度（单位为degree，度）。

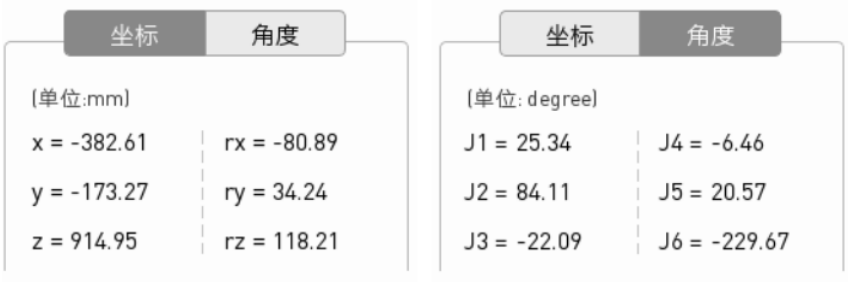


图4- 2路点坐标

路点控制 保存当前点 该按键用于保存机器人当前位姿数据。 移动到该点 如需要验证示教点位或移动到示教点位进行某些操作，长按该按键直至控制机器人运动到当前示教点位。 清除已保存点 如不再需要当前示教点位，该按键用于将当前示教点位清零。 高级功能 使用共享配置：该功能正在调试中，敬请期待！ 高级配置 如图4-3所示，在高级配置页面中，用户可以设置移动方式、接近方式、指令速度、力矩限制。

图4- 3高级配置

相对点：相对点适用于需要基于机器人上一条移动指令对应点/某个绝对点/- 变量点偏移一定位移的情境。位移量可以是单个方向的距离，也可以是多个方向位移的叠加，还可以示教一段路径进行偏移。 如图4-4所示是相对点的具体配置页面。

图4- 4相对点

直接输入（相对移动） 如图4-5所示，可以直接输入坐标值/关节角度。

图4- 5直接输入的两种形式

无论是输入坐标值或者关节角度，根据偏移需求选择六个值中的其一或者更多，不一定每一个值都需要输入。

例如，如图4-6所示，在实际拾取和放置过程中，需要在目标放置位置上方设置一个过渡点。这时，就可以设置一条路点指令为绝对点，控制机器人（此时机器人应是夹持工件的状态）移动到放置点，点击保存当前点，这就生成了图4-6所示的②号指令行。接着再点击基础功能-路点：选择相对点，设置图示的z方向增加50mm的相对点，那么在运行完上一句后机器人就会移动到过渡点的位置。在实际拾取和放置过程中，可能还会在这两条指令间加入其它指令，如设置指令，将夹爪打开。

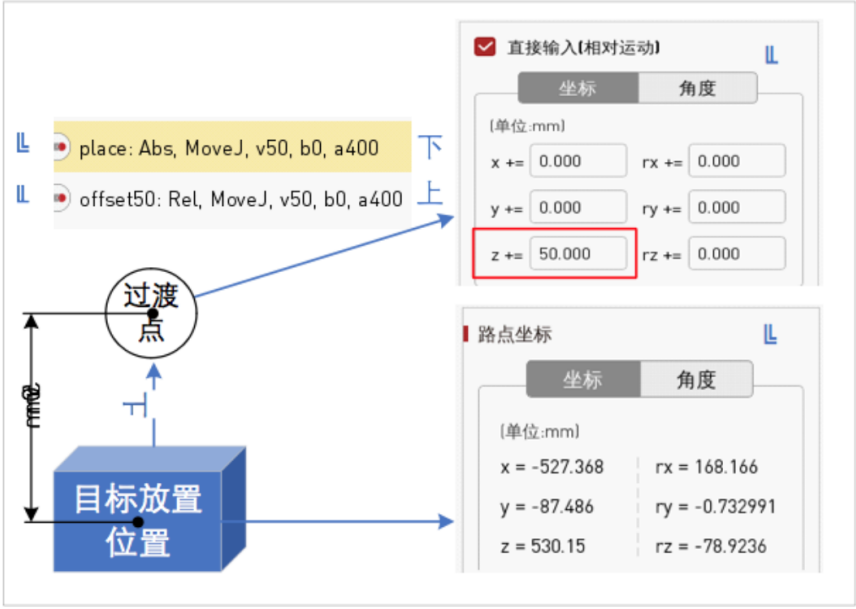


图4- 6直接输入坐标值的应用示例

除了在上一条运动指令的位置基础上进行偏移，相对点指令还可以基于一个路点或变量点进行偏移。 “移动到该点”按键可以验证偏移运动，“清除已保存点”可以清除当前输入的内容。 参考移动：利用示教两个点位，生成了一段路径，以当前点为基础，再现这段轨迹。 高级功能：同绝对点的高级配置，不再赘述。

共享点：共享点可以使用其他路点的位置。如图4-7所示是共享点的具体配置页面。

图4- 7共享点

共享点：在选框中选择想要共享的点位，可以长按“移动到该点”控制机器人移动到该点、点击“清除已保存点”清除当前共享点。 高级功能：同绝对点的高级配置，不再赘述。 变量：该路点可以由变量赋值，用户可以使用通信方法从其他设备获取该

路点位置。 如图4-8所示是变量点的具体配置页面。 变量赋值：选择关联的位姿变量即可，“移动到该点”可以检查位姿是否是目标位姿。 高级功能：同绝对点的高级配置，不再赘述。

图4- 8变量

4.1.2 夹爪 如图4-9所示是夹爪的具体配置页面。

图4- 9 夹爪

用户通过一个简单的功能定义和控制夹爪。

- 1 选择夹爪
- 2 设置已存在夹爪
- 3 选择夹爪，可以对已有夹爪进行编辑或者删除。

- 4 定义新夹爪 如图4-10所示，可以命名夹爪，同时控制多个输入信号：设置需要控制的输出信号的数量、在“设置”中选择设置第几个信号、设置状态（关系到具体执行时对应“打开”或“关闭”功能）、设置对应输出信号。在设置完成后，还可以选择等待条件。

图4- 10 定义新夹爪

设置已保存状态

- 完全打开：执行夹爪定义中为“打开”状态的选项。
- 完全关闭：执行夹爪定义中为“关闭”状态的选项。 调试控制
- 打开夹爪：手动操作执行夹爪定义中为“打开”状态的选项。
- 关闭夹爪：手动操作执行夹爪定义中为“关闭”状态的选项。

4.1.3 等待 如图4-11所示，等待指令一共有四种模式。

- 等待时间：可以设置延时时间，单位为秒。
- 等待输入信号：对输入信号的状态进行判断，除非符合已设置的输入信号状态条件，否则一直等待。
- 等待输出信号：对输出信号的状态进行判断，除非符合已设置的输出信号状态条件，否则一直等待。
- 等待条件：可以自定义等待条件，除非符合等待条件，否则一直等待。

图4- 11等待指令

4.1.4 设置 如图4-12所示，设置指令有四种模式的选择。

- 设置PIN：设置输出信号的状态，除了选择设置的输出信号，确定其是打开或关闭的状态，还可以设置该信号保持的时间。
- 设置条件：自定义设置的内容。
- 设置TCP（即工具中心点）。
- 设置载荷。

图4- 12 设置指令

4.1.5组合 如图4-13所示，组合指令提供了常用组合模板，例如抓取和放置组合。

图4- 13 组合指令

用户使用组合模板时，例如使用抓取和放置模板，可以直接在模板程序的基础上修改参数、示教路点等，也可以根据需求自由增删指令。 使用组合模板可以简化用户查找指令的过程，更方便快捷完成对应项目的编程。

4.2 逻辑功能

4.2.1循环 循环指令可以使循环内的所有指令重复执行一定的次数。如图4-14所示，循环次数可以用常量或变量、表达式表示。

图4- 14 循环指令

4.2.2 条件判断 对于设置的条件进行判断，允许程序读取数据，判断并确定下一步该做什么。 条件判断指令可以用来判断I/O信号，也可以用来判断其他条件。 条件判断指令由3个部分：“如果”，“否则如果”和“否则”组成，这三个部分相互之间的关

系如下： 除了“如果”是不可或缺的组成部分，其余两项是可选部分； 如果同时存在“如果”，“否则如果”和“否则”，那么程序将首先读取“如果”，然后读取“否则如果” ... 第n个“否则如果”，“否则”，这三者的关系如图4-15所示：

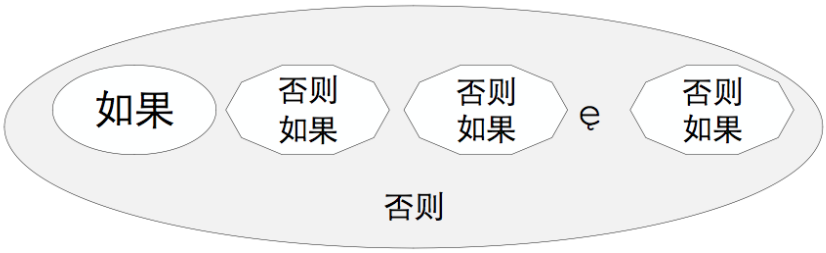


图4- 15 “如果”，“否则如果”和“否则”的关系

可以有多个“否则如果”，但有且只有一个“如果”，如果选择添加“否则”也只能有一个“否则”。 可以删除“否则如果”或“否则”，但如果删除了“如果”，则应删除所有“否则如果”和“否则”。 如图4-16所示是条件判断指令的设置页面。

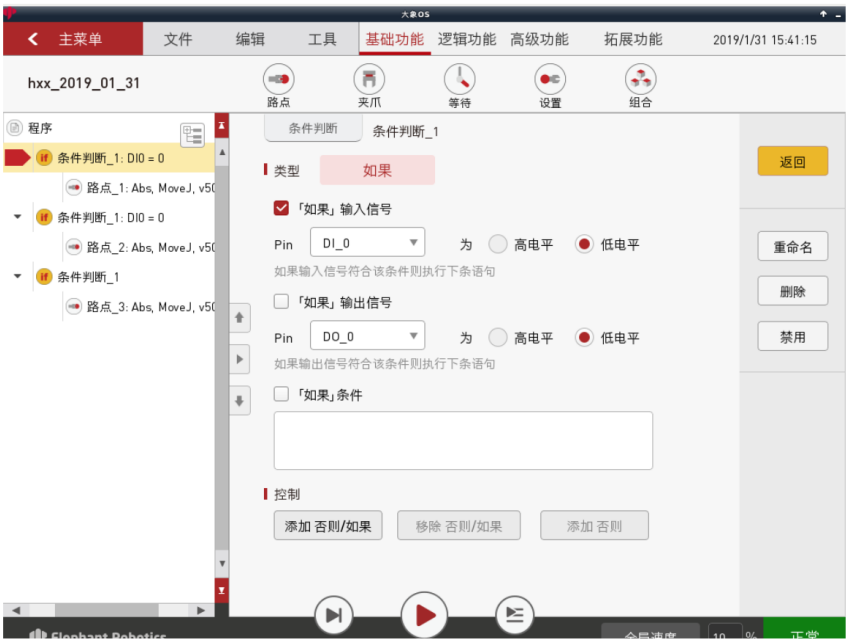


图4- 16 条件判断指令

如上图所示，倘若符合“如果”的判断条件，那么机器人将会运动到路点1处；倘若符合“否则如果”的判断条件，那么机器人将会运动到路点2处；倘若以上两个条件都不符合，机器人将会运动到路点3的位置。

4.2.3 子程序 如图4-17所示，使用该指令可以调用其他子程序。主程序可以使用多个子程序，但子程序内没有自己的子程序。

图4- 17 子程序指令

如图4-18所示，可以在主程序中查看和编辑子程序。如对子程序进行编辑，请注意在保存后才能生效。

图4- 18 显示子程序

4.2.4线程 线程沿主程序运行。 它用于检查信号，例如紧急按钮或安全光幕。如图4-19所示，可以设置线程的运行间隔时间。

图4- 19 线程指令

注意，线程内不允许使用运动指令。 **4.2.5 暂停** 暂停指令用于控制机器人暂停、停止、重新开始。如图4-20所示是暂停指令的具体配置页面。

- 设置暂停、停止状态的同时还可以选中“显示弹窗”，自定义弹窗显示的内容。
- 设置重启状态，程序运行到本指令时，将会重新从开头第一句指令开始运行。

图4- 20 暂停指令

4.2.6 条件选择 如图4-21所示，条件选择指令用于对某个变量的值进行判断。

图4- 21 条件选择指令

对应不同的条件值，有多少个条件值需要判断就增加多少个case，可以打开每一个case，增加对应执行的指令。例如，对整型变量A进行判断，设置2个case，如果A为1，执行第一个路点指令，如果A为2，执行第二个路点指令。 如果只判断少数变量值，其他情况统一处理，需要选择“切换”，在切换里面增加对应执行的指令。

4.3 高级功能

4.3.1托盘 托盘功能允许用户只示教少数点，通过这些点可以由机器人系统计算出其他点的位置，运行该指令能够控制机器人运动到这些点位。如图4-22所示，可以选择直线、平面、立方体、离散点。

图4- 22 托盘类型选择

如图4-23所示，选择“直线”之后，选择点的数量，将会根据点的数量把直线平均分割，这些点就是分割点，通过示教两个点确定这条直线。

图4- 23 直线

如图4-24所示，选择“平面”之后，分别选择两个轴的点的数量，平面被平均分割，这些点就是分割点，通过示教四个点确定这个平面。

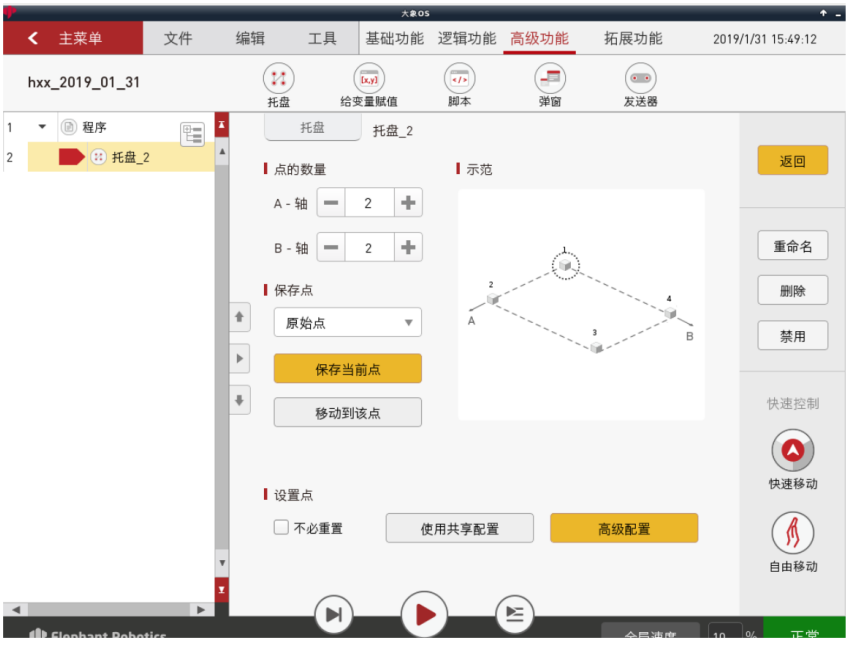


图4- 24 平面

如图4-25所示，选择“立方体”之后，分别选择三个轴的点的数量，立方体被平均分割，这些点就是分割点，通过示教八个点确定这个立方体。

图4- 25 立方体

如图4-26所示，选择“离散点”时，选择点的数量，分别示教不同的点位。也就是说，离散点是多个点的集合。

图4- 26 离散点

4.3.2给变量赋值 如图4-27所示，本指令可以给整型变量、字符串变量赋值，还可以利用“设置变量”直接根据指令设置变量的值。

图4- 27 给变量赋值

4.3.3脚本 脚本指令可以用于复杂指令的编辑，提供了更丰富的功能指令。如图4-28所示是脚本指令的具体配置页面。设置脚本一共有两种类型，一种是单行表达式，另一种是多行脚本。

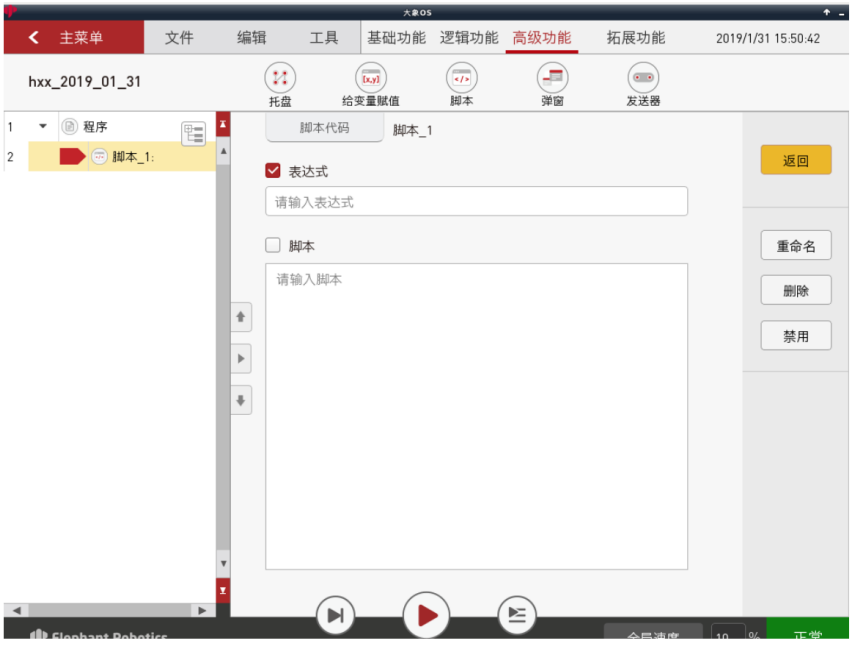


图4- 28 脚本

4.3.4弹窗 弹窗指令允许用户自定义弹窗。也就是说，执行本指令时会有弹窗出现，弹窗内容是用户自定义的内容。如图4-29所示，弹窗有三种类型，信息、警告、错误，用户选择其一，再自定义弹窗内容。

弹窗控制也有三种：继续程序（日志记录），即不弹窗，只是将弹窗内容显示到日志中，程序继续运行；弹窗时暂停程序，即出现弹窗，并且程序暂停运行；弹窗时停止程序，即出现弹窗，同时程序停止运行。

图4- 29 弹窗

4.3.5发送器 如果要进行TCP/IP通信，机器人系统必须设置IP和端口号，作为客户端或服务端，与其他设备进行通信。 发送器允许用户设置TCP/IP连接。如图4-30所示是发送器指令的具体配置页面。

如果机器人系统作为客户端，填写的IP地址是作为服务端的外部设备的IP地址，端口号对应服务端分配给机器人系统的端口号，在服务端处于监听状态下，点击连接，就可以实现与服务端进行通信。

如果机器人系统作为服务端，填写的IP地址是本机的IP地址，端口号对应分配给客户端设备的端口号，点击监听，此时客户端设备可以连接机器人系统。在客户端列表中可以查看所有客户端的IP地址和端口号。

建立通信后，可以发送和接收数据。

图4- 30 发送器

5 快速新建一个可运行项目

5.1 流程说明

5.1.1 准备工作

前提条件

- 完整的机器人系统。
- 无故障。 准备内容
- 将电源插头插到提供AC 220V的插板上。
- 打开电源开关。 按下示教器上的启动按键。

5.1.2 流程图 如图5-1所示即为程序编辑流程图。

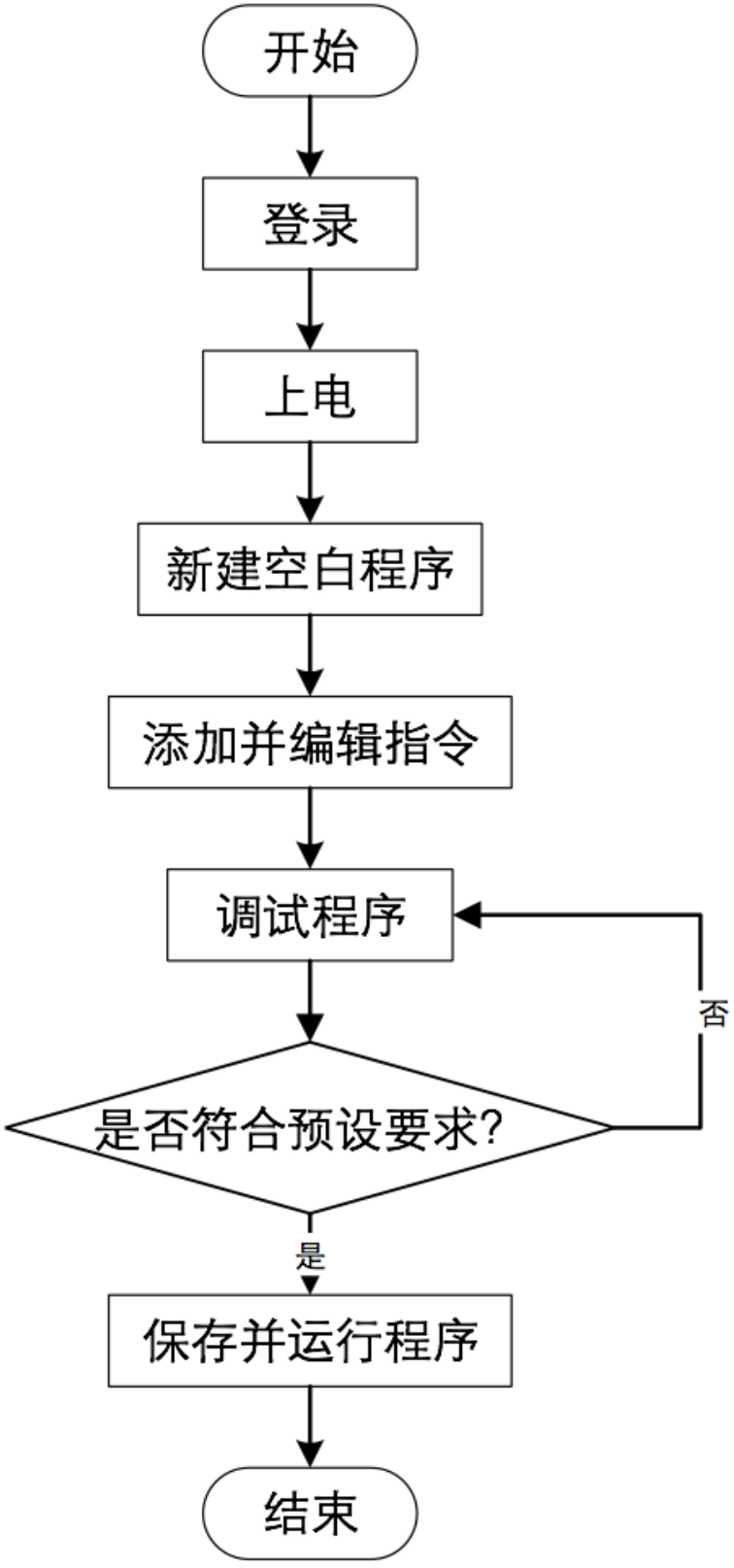


图5- 1 程序编辑流程图

5.2 具体步骤

5.2.1 登录 当系统成功启动后，将会进入如图5-2所示的RoboFlow操作系统的登录界面。

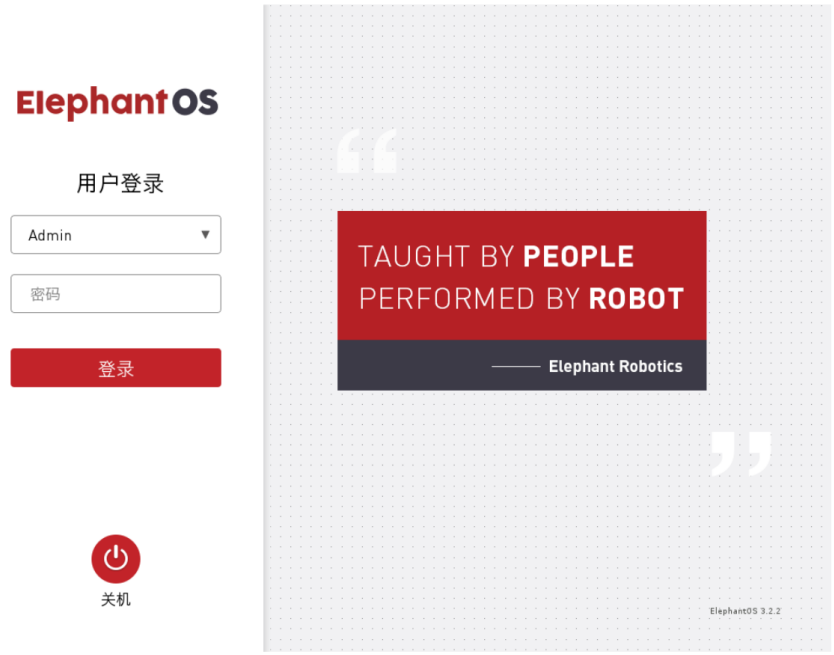


图5- 2 登录界面

选择登录用户名“Admin”或其他管理员用户名（只有管理员权限才允许编辑和调试程序），点击密码框将会出现如图5-3所示的弹窗。

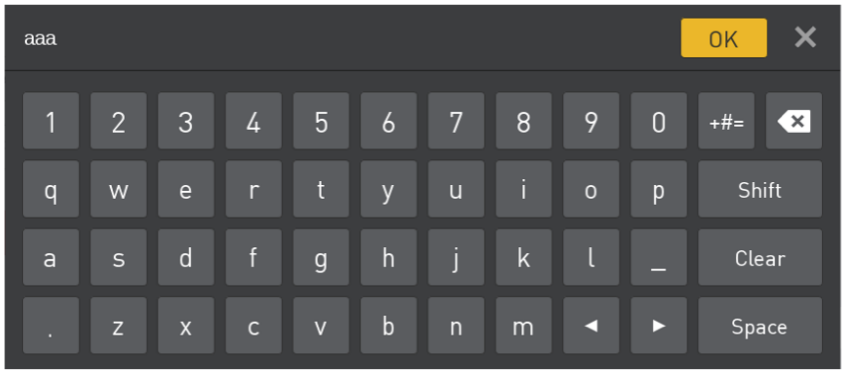


图5- 3 输入键盘

默认管理员用户“Admin”对应的登录密码是“aaa”（如若选择了其他管理员用户名则输入对应登录密码），输入密码点击“OK”，将回到图5-2界面。再点击“登录”，即可成功登录。 **5.2.2 上电** 登录成功后，将会进入如图5-4所示的主菜单界面。

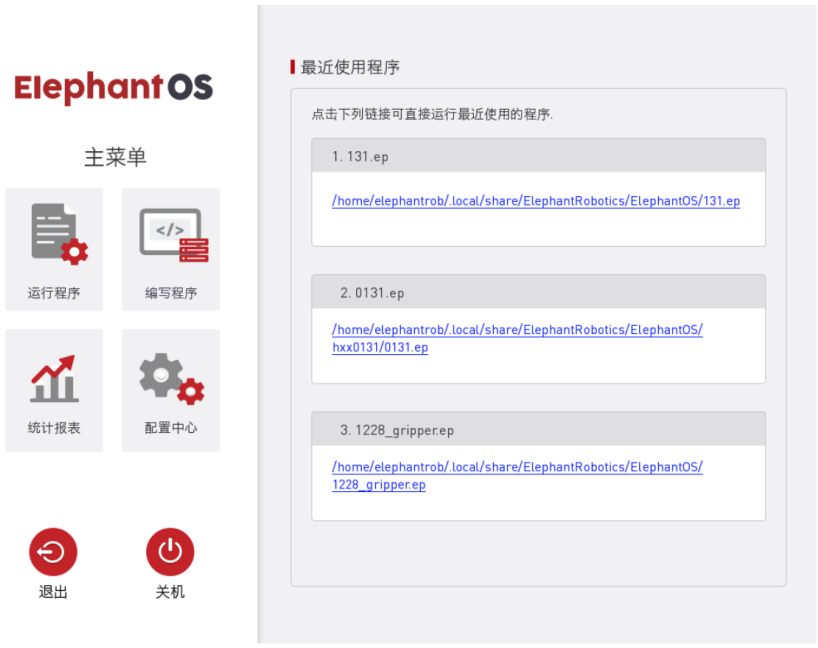


图5- 4 主菜单

在主菜单界面中，选择“配置中心”，将会进入如图5-5所示的界面（此时还未上电）。

在确保急停旋钮未被按下的情况下，点击如图5-5所示的“启动机器人”按键。此时界面将会发生变化，将会出现如图5-6所示的“正在上电中”图标。如若上电成功，将会出现如图5-7所示的“正常”状态。如若失败，请检查是否缺少执行哪些步骤。

完成上一个步骤后，在配置中心中点击“< 主菜单”按键即可返回主菜单。

图5- 5 未上电状态

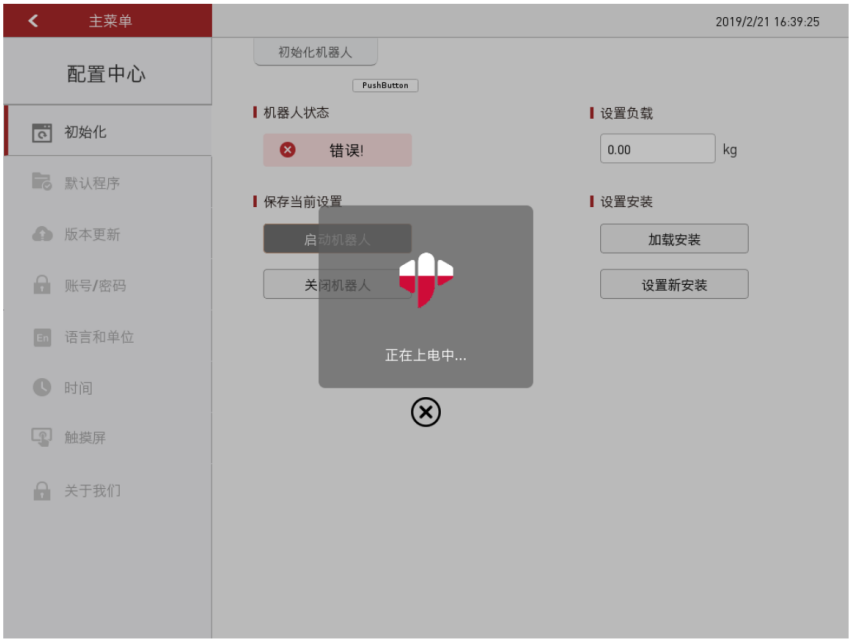


图5- 6 上电中

图5- 7 上电完成

5.2.3 新建空白程序 如图5-8所示，点击“编写程序”，再选择“空白程序”。

图5- 8 选择“空白程序”

执行完上一步操作后，进入如图5-9所示的程序编辑界面。

图5- 9 进入程序编辑界面

5.2.4 添加并编辑指令 如图5-10所示，添加两条路点：绝对点指令，并示教两个点位（即利用快速移动工具手动操作机器人，控制机器人运动到某一位姿，返回，点击“保存当前点”。两个点位的示教步骤相同。如需验证保存点位，长按“移动到该点”按键可以手动操作控制机器人移动到示教点位。）。

编辑完成后，请注意保存程序文件。点击如图5-10所示的文件选项栏中的“保存”，将弹出如图5-11所示的窗口。点击“文件名”，将会出现如图5-12的输入键盘。输入文件名后，点击“OK”。将回到保存界面，点击“确定”，程序文件保存成功。保存成功后，如图5-13所示，在程序编辑界面左上角的程序名称将会被更改。

图5- 10 程序编辑

图5- 11 进入保存文件界面

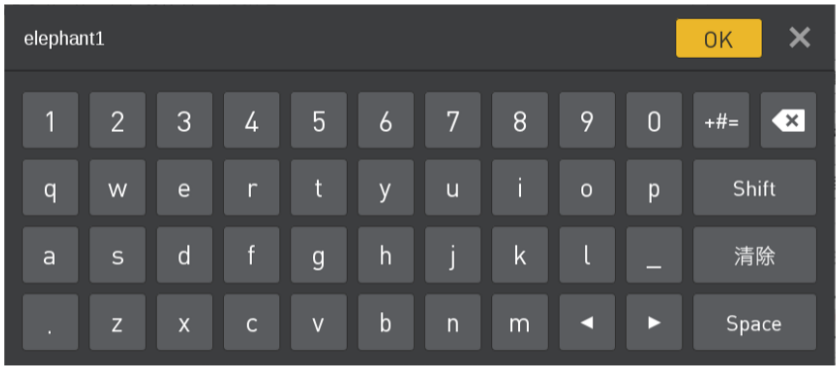


图5- 12 输入程序名称

图5- 13 文件保存成功

5.2.5 调试程序 如图5-14所示，除了程序运行控制栏中提供的“下一步”、“运行”两个功能外，点击“高级功能”，可以进入更多设置的界面。

其中，“下一步”功能对应的是一步一步执行程序，点击一次只运行一步，如需继续运行则继续点击“下一步”。“运行”功能对应的是自动运行程序一次。

“高级功能”中，可以设置循环运行的次数，也可以无限循环运行。还可以控制程序以自动运行模式还是手动运行模式运行。在自动运行模式下可以使用“下一步”、“运行”和循环运行。在如图5-14所示界面下选择“手动运行模式”，再选择循环运行中的“运行”或“无限循环”。即可进入如图5-15所示的手动运行模式下的运行界面。

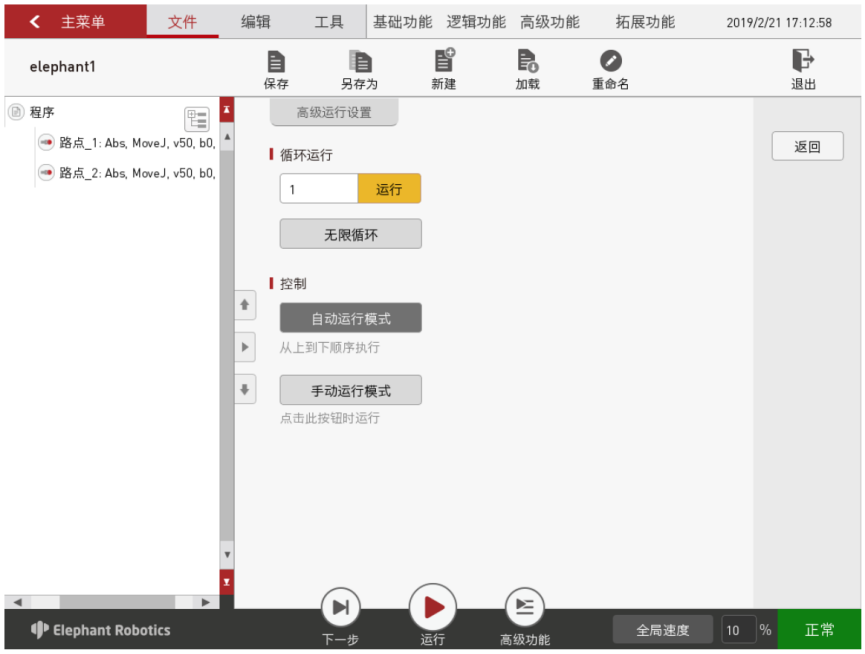


图5- 14 调试程序

图5- 15 手动模式调试程序

如果使用手动模式调试程序，需要一直按住“Press Down”按键，才能继续运行。如果松开按键，则程序暂停，再按下，则继续运行。

5.2.6 保存并运行程序 如果调试完成，请确保已将调试完成的程序保存。返回主菜单后，选择“运行程序”。将会出现如图5-16所示的弹窗，选择调试完成的程序，点击“确定”。

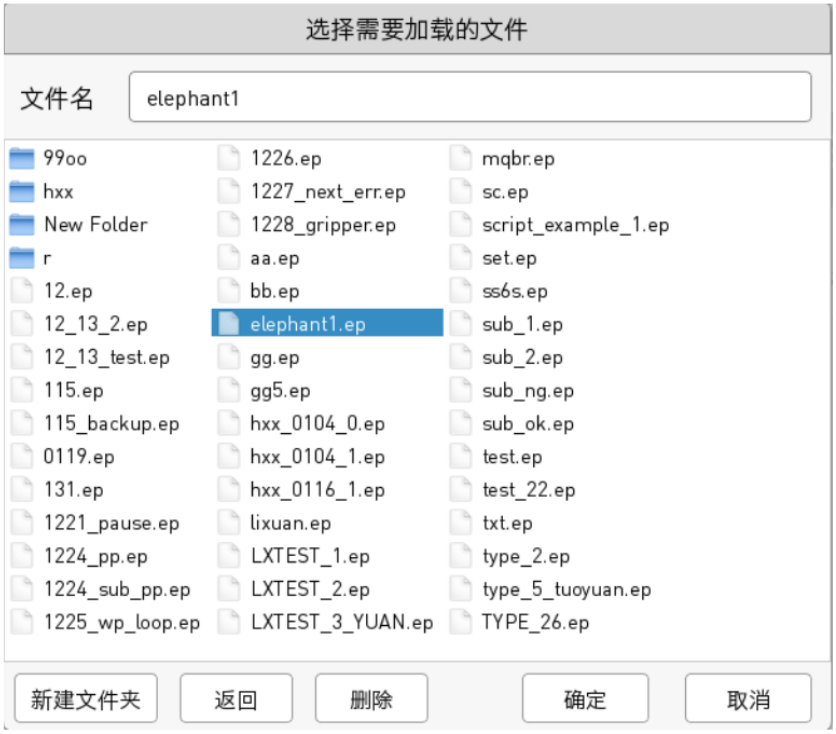


图5- 16 选择程序

选择程序后，将会进入如图5-17所示的运行程序界面，在此界面中，可以运行程序，查看程序运行信息。

如果确定近期将会持续运行该程序，还可以在配置中心-默认程序中选择该程序。这样只要系统启动，就会自动跳转到“运行程序”界面，上电成功后，点击运行既可运行该程序。

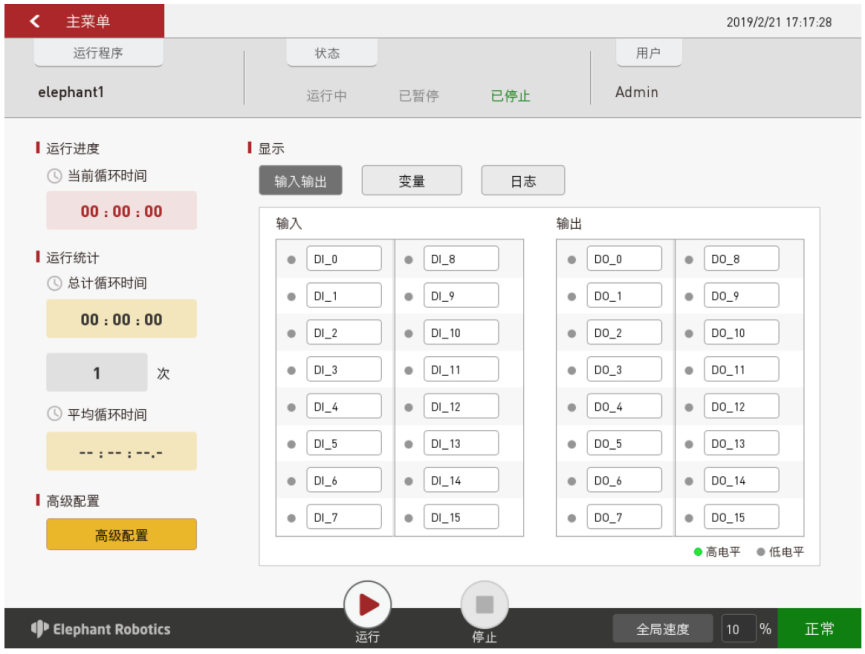
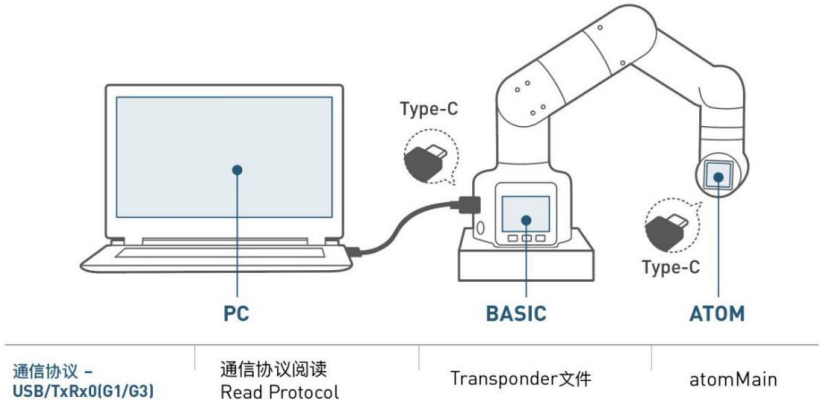


图5- 17 运行程序

6 通信与报文命令

注意：使用通信协议直接通信，需要在basic中烧录transponder，在atom中烧录最新版的atomMain



6.1USB通信设置Communication Settings

请确保您的通信设置如下

- 总线接口： USB Type-C连接
- 波特率： 115200
- 数据位： 8
- 奇偶校验：无
- 停止位： 1

6.2命令帧说明及单一指令解析

主机Basic向从机发送数据，从机接收到数据后进行解析，如包含返回值的指令，从机会在500ms内返回给主机。

6.3命令帧发送与接收格式

所有命令为十六进制，发送与接收格式一致。

每个通信命令必须包含以下5个部分，其中3、4可为空。

- 1 命令针头: 0xFE 0xFE
  - 固定
  - 必含
- 2 有效命令长度: 0x02 ~ 0x10
  - 以下所有命令的长度
  - 必含
- 3 命令序号: 00 ~ 8F
  - 现已开发了多种命令
  - 可为空

- 4 命令内容: 若干
  - 可为空
- 5 命令结束: 0XFA
  - 固定
  - 必含

6.4指令解析

主机Basic向从机发送数据，从机接收到数据后进行解析，如包含返回值的指令，从机会在500ms内返回给主机。

类型	数据描述	数据长度	说明
命令帧	头字节0	1	帧头识别，0XFE
	头字节1	1	帧头识别，0XFE
	数据长度字节	1	不同指令对应不同长度数据
	命令字节	1	视不同命令而定
数据帧	数据	0-16	命令附带数据，视不同命令而定
结束帧	结束字节	1	停止位，0XFA

6.5单一指令解析

1. Atom打开通讯

数据域	说明	数据
Data[0]	识别帧	0XFE
Data[1]	识别帧	0XFE
Data[2]	数据长度帧	0X02
Data[3]	指令帧	0X10
Data[4]	结束帧	0XFA

串口发送示例：FE FE 02 10 FA

无返回值

---

1. Atom关闭通讯

数据域	说明	数据
Data[0]	识别帧	0XFE
Data[1]	识别帧	0XFE
Data[2]	数据长度帧	0X02
Data[3]	指令帧	0X11
Data[4]	结束帧	0XFA

串口发送示例：FE FE 02 11 FA

无返回值

---

1. Atom状态查询

数据域	说明	数据
Data[0]	识别帧	0XFE
Data[1]	识别帧	0XFE
Data[2]	数据长度帧	0X02
Data[3]	指令帧	0X12
Data[4]	结束帧	0XFA

串口发送示例：FE FE 02 12 FA

返回值

数据域	说明	数据
Data[0]	返回帧头	0XFE
Data[1]	返回帧头	0XFE
Data[2]	返回长度帧	0X02
Data[3]	返回指令帧	0X12
Data[4]	上电/断电	0X01/0X00
Data[5]	结束帧	0XFA

串口返回示例：FE FE 02 12 00 FA

---

1. 读取角度（读取走位信息）

数据域	说明	数据
Data[0]	识别帧	0XFE
Data[1]	识别帧	0XFE
Data[2]	数据长度帧	0X02
Data[3]	指令帧	0X20
Data[4]	结束帧	0XFA

串口发送示例：FE FE 02 20 FA

从机返回数据结构

数据域	说明	数据
Data[0]	返回帧头	0XFE
Data[1]	返回帧头	0XFE
Data[2]	返回长度帧	0X0E
Data[3]	返回指令帧	0X20
Data[4]	1号舵机角度高位	Angle1_high
Data[5]	1号舵机角度低位	Angle1_low
Data[6]	2号舵机角度高位	Angle2_high
Data[7]	2号舵机角度低位	Angle2_low
Data[8]	3号舵机角度高位	Angle3_high
Data[9]	3号舵机角度低位	Angle3_low
Data[10]	4号舵机角度高位	Angle4_high
Data[11]	4号舵机角度低位	Angle4_low
Data[12]	5号舵机角度高位	Angle5_high
Data[13]	5号舵机角度低位	Angle5_low
Data[14]	6号舵机角度高位	Angle6_high
Data[15]	6号舵机角度低位	Angle6_low
Data[16]	结束帧	0XFA

串口返回示例：FE FE 0E 20 06 E6 EA 4E C4 81 0B BD EA C0 02 B6 FA

如何得出1号关节角度

temp = angle1_low+angle1_high*256

Angle1=（temp \ 33000 ?(temp – 65536) : temp）/100

计算方式：角度值低位 + 角度高位值乘以256 先判断是否大于33000 如果大于33000就再减去65536 最后除以100 如果小于33000就直接除以100

（其余同理）

1. 发送单独角度

数据域	说明	数据
Data[0]	识别帧	0XFE
Data[1]	识别帧	0XFE
Data[2]	数据长度帧	0X06
Data[3]	指令帧	0X21
Data[4]	舵机序号	joint_no
Data[5]	角度值高位	angle_high
Data[6]	角度值低位	angle_low
Data[7]	指定速度	sp
Data[8]	结束帧	0XFA

串口发送示例：FE FE 06 21 00 00 00 20 FA

joint_no取值范围 0~5

angle_high：数据类型byte

计算方式：角度值乘以100 先转换成int形式 再取十六进制的高字节

angle_low：数据类型byte

计算方式：角度值乘以100 先转换成int形式 再取十六进制的低字节

无返回值

6. 发送全部角度

数据域	说明	数据
Data[0]	识别帧	0XFE
Data[1]	识别帧	0XFE
Data[2]	数据长度帧	0X0F
Data[3]	指令帧	0X22
Data[4]	1号舵机角度值高字节	Angle1_high
Data[5]	1号舵机角度值低字节	Angle1_low
Data[6]	2号舵机角度值高字节	Angle2_ high
Data[7]	2号舵机角度值低字节	Angle2_ low
Data[8]	3号舵机角度值高字节	Angle3_ high
Data[9]	3号舵机角度值低字节	Angle3_ low
Data[10]	4号舵机角度值高字节	Angle4_ high
Data[11]	4号舵机角度值低字节	Angle4_ low
Data[12]	5号舵机角度值高字节	Angle5_ high
Data[13]	5号舵机角度值低字节	Angle5_ low
Data[14]	6号舵机角度值高字节	Angle6_ high
Data[15]	6号舵机角度值低字节	Angle6_ low
Data[16]	指定速度	Sp
Data[17]	结束帧	0XFA

串口发送示例：FE FE 0F 22 06 E6 EA 4E C4 81 0B BD EA C0 02 B6 FA

angle1_high：数据类型byte

计算方式：1号舵机角度值乘以100 先转换成int形式 再取十六进制的高字节

angle1_low：数据类型byte

计算方式：1号舵机角度值乘以100 先转换成int形式 再取十六进制的低字节

（其余同理）

无返回值

---

1. 读取全部坐标

数据域	说明	数据
Data[0]	识别帧	0XFE
Data[1]	识别帧	0XFE
Data[2]	数据长度帧	0X02
Data[3]	指令帧	0X23
Data[4]	结束帧	0XFA

串口发送示例：FE FE 02 23 FA

从机返回数据结构

数据域	说明	数据
Data[0]	返回帧头	0XFE
Data[1]	返回帧头	0XFE
Data[2]	返回长度帧	0X0E
Data[3]	返回指令帧	0X23
Data[4]	指定x坐标高位	x_high
Data[5]	指定x坐标低位	x_low
Data[6]	指定y坐标高位	y_ high
Data[7]	指定y坐标低位	y_ low
Data[8]	指定z坐标高位	z_ high
Data[9]	指定z坐标低位	z_low
Data[10]	指定rx坐标高位	rx_high
Data[11]	指定rx坐标低位	rx_low
Data[12]	指定ry坐标高位	ry_high
Data[13]	指定ry坐标低位	ry_low
Data[14]	指定rz坐标高位	rz_high
Data[15]	指定rz坐标低位	rz_low
Data[16]	结束帧	0XFA

串口返回示例：FE FE 0E 23 00 00 00 00 00 00 00 00 00 00 00 00 FA

如何得出x坐标

temp = x_low + x_high*256

x坐标=（temp \ 33000 ?(temp – 65536) : temp）/10

计算方式：x坐标值低位 +x坐标值高位乘以256 先判断是否大于33000 如果大于33000就再减去65536 最后除以100 如果小于33000就直接除以10

（y坐标z坐标同理）

如何得出rx坐标

temp = rx_low + rx_high*256

x坐标=（temp \ 33000 ?(temp – 65536) : temp）/100

计算方式：x坐标值低位 +x坐标值高位乘以256 先判断是否大于33000 如果大于33000就再减去65536 最后除以100 如果小于33000就直接除以100

（ry坐标rz坐标同理）

---

1. 发送单独坐标参数]8

数据域	说明	数据
Data[0]	识别帧	0XFE
Data[1]	识别帧	0XFE
Data[2]	数据长度帧	0X06
Data[3]	指令帧	0X24
Data[4]	axis	x/y/z/rx/ry/rz
Data[5]	指定xyz/rxryrz参数高位	xyz/ rxryrz_high
Data[6]	指定xyz/rxryrz参数低位	xyz/rxryrz_low
Data[7]	指定速度	Sp
Data[8]	结束帧	0XFA

设定X坐标为100，目标速度20

串口发送示例：FE FE 06 24 00 00 64 20 FA

指定坐标axis：数据类型byte

取值范围：0~5

xyz_high：数据类型byte

计算方式：x/y/z坐标值乘以10 再取十六进制的高字节

xyz_low：数据类型byte

计算方式：x/y/z坐标值乘以10 再取十六进制的低字节

rxryrz_high：数据类型byte

计算方式：rx/ry/rz乘以100 再取十六进制的高字节

rxryrz_low：数据类型byte

计算方式：rx/ry/rz乘以100 再取十六进制的低字节

无返回值

---

1. 发送全部坐标参数

数据域	说明	数据
Data[0]	识别帧	0XFE
Data[1]	识别帧	0XFE
Data[2]	数据长度帧	0X10
Data[3]	指令帧	0X25
Data[4]	指定x坐标高位	x_high
Data[5]	指定x坐标低位	x_low
Data[6]	指定y坐标高位	y_ high
Data[7]	指定y坐标低位	y_ low
Data[8]	指定z坐标高位	z_ high
Data[9]	指定z坐标低位	z_low
Data[10]	指定rx坐标高位	rx_high
Data[11]	指定rx坐标低位	rx_low
Data[12]	指定ry坐标高位	ry_high
Data[13]	指定ry坐标低位	ry_low
Data[14]	指定rz坐标高位	rz_high
Data[15]	指定rz坐标低位	rz_low
Data[16]	指定速度	Sp
Data[17]	模式	0X01
Data[18]	结束帧	0XFA

设定机械臂末端目标点位（-14，-27，275，-89.5，0.7，-90.7），目标速度50

串口发送示例：FE FE 10 25 FF 74 FE EE 0A C1 DD 05 00 48 DC 95 32 01 FA

x_high：数据类型byte

计算方式：x坐标乘以10 再取十六进制的高字节

x_low：数据类型byte

计算方式：x坐标乘以10 再取十六进制的低字节

（y轴坐标z轴坐标同理）

rx_high：数据类型byte

计算方式：rx坐标值乘以100 再取十六进制的高字节

rx_low：数据类型byte

计算方式：rx坐标值乘以100 再取十六进制的低字节

（ry轴坐标rz轴坐标同理）

无返回值

1. 是否达到点位

数据域	说明	数据
Data[0]	识别帧	0XFE
Data[1]	识别帧	0XFE
Data[2]	数据长度帧	0X15
Data[3]	指令帧	0X2A
Data[4]	坐标x高位	x_high
Data[5]	坐标x低位	x_low
Data[6]	坐标y高位	y_high
Data[7]	坐标y低位	y_low
Data[8]	坐标z高位	z_high
Data[9]	坐标z低位	z_low
Data[10]	坐标rx高位	rx_high
Data[11]	坐标rx低位	rx_low
Data[12]	坐标ry高位	ry_high
Data[13]	坐标ry低位	ry_low
Data[14]	坐标rz高位	rz_high
Data[15]	坐标rz低位	rz_low
Data[16]	Is_linear	type
Data[17]	结束帧	0XFA

判断机械臂是否到达指定点位

串口发送示例： FE FE 15 2A FF 74 FE EE 0A C1 DD 05 00 48 DC 95 FA

x_high：数据类型byte

计算方式：x坐标乘以10 先转换为int类型 再取十六进制高字节

x_low：数据类型byte

计算方式：x坐标乘以10 先转换为int类型 再取十六进制低字节

（y轴坐标z轴坐标同理）

rx_high：数据类型byte

计算方式：rx坐标乘以100 先转换为int类型 再取十六进制高字节

rx_low：数据类型byte

计算方式：rx坐标乘以100 先转换为int类型 再取十六进制低字节

（ry轴坐标rz轴坐标同理）

Type：数据类型byte（暂未使用）

返回数据结构

数据域	说明	数据
Data[0]	返回识别帧	0XFE
Data[1]	返回识别帧	0XFE
Data[2]	返回数据长度帧	0X03
Data[3]	返回指令帧	0X2A
Data[4]	InPosition/noInPosition	0X01/0X00
Data[5]	结束帧	0XFA

已经到达点位；

串口返回示例：FE FE 03 2A 00 FA

---

1. 机械臂运动检测

数据域	说明	数据
Data[0]	识别帧	0XFE
Data[1]	识别帧	0XFE
Data[2]	数据长度帧	0X02
Data[3]	指令帧	0X2B
Data[4]	结束帧	0XFA

检查机械臂是否在运动

串口发送示例： FE FE 02 2B FA

返回数据结构

数据域	说明	数据
Data[0]	返回识别帧	0XFE
Data[1]	返回识别帧	0XFE
Data[2]	返回数据长度帧	0X02
Data[3]	返回指令帧	0X2B
Data[4]	Not running/no data - running	0X00/0X01
Data[5]	结束帧	0XFA

串口返回示例： FE FE 02 2B 00 FA

---

1. jog-关节方向运动

数据域	说明	数据
Data[0]	识别帧	0XFE
Data[1]	识别帧	0XFE
Data[2]	数据长度帧	0X05
Data[3]	指令帧	0X30
Data[4]	关节舵机序号	Joint
Data[5]	关节舵机方向	di
Data[6]	指定速度	sp
Data[7]	结束帧	0XFA

设定一号舵机顺时针方向以50%速度转动

串口发送示例： FE FE 05 30 01 01 32 FA

关节序号取值范围 1~6

di：数据类型byte 取值范围 0和1

sp：数据类型byte 取值范围0-100%

无返回值

---

1. jog-坐标方向运动

数据域	说明	数据
Data[0]	识别帧	0XFE
Data[1]	识别帧	0XFE
Data[2]	数据长度帧	0X05
Data[3]	指令帧	0X32
Data[4]	指定坐标	axis
Data[5]	关节舵机方向	Direction
Data[6]	指定速度	sp
Data[7]	结束帧	0XFA

设定末端向x轴逆时针方向以50%的速度运动

串口返回示例： FE FE 05 32 01 00 32 FA

Axis_number：数据类型byte （x = 0 ,y,z,rx,ry,rz） 取值范围1~6

di：数据类型byte 取值范围 0~1

sp：数据类型byte 取值范围 0-100%

无返回值

1. jog-停止

数据域	说明	数据
Data[0]	识别帧	0XFE
Data[1]	识别帧	0XFE
Data[2]	数据长度帧	0X02
Data[3]	指令帧	0X34
Data[4]	结束帧	0XFA

jog停止运动

串口返回示例： FE FE 02 34 FA

无返回值

1. 发送电位值]

数据域	说明	数据
Data[0]	识别帧	0XFE
Data[1]	识别帧	0XFE
Data[2]	数据长度帧	0X05
Data[3]	指令帧	0X3A
Data[4]	关节舵机序号	Joint
Data[5]	电位值高位	Encoder_high
Data[6]	电位值低位	Encoder_low
Data[7]	结束帧	0XFA

示例，设定2号关节到2249电位。

串口发送示例： FE FE 05 3A 01 08 C9 FA

关节序号取值范围 0~5

Joint：数据类型byte

Encoder_high：数据类型byte

计算方式：取电位值（十六进制）的高位

Encoder_low：数据类型byte

计算方式：取电位值值（十六进制）的低位

无返回值

16. 获取电位值;)

数据域	说明	数据
Data[0]	识别帧	0XFE
Data[1]	识别帧	0XFE
Data[2]	数据长度帧	0X03
Data[3]	指令帧	0X3B
Data[4]	关节序号	joint
Data[5]	结束帧	0XFA

获取1号舵机电位值

串口发送示例： FE FE 03 3B 00 FA

关节序号取值范围 1~6

返回数据结构

数据域	说明	数据
Data[0]	返回识别帧	0XFE
Data[1]	返回识别帧	0XFE
Data[2]	返回数据长度帧	0X04
Data[3]	返回指令帧	0X3B
Data[4]	舵机电位值高位	Encoder_high
Data[5]	舵机电位值低位	Encoders_low
Data[6]	结束帧	0XFA

串口返回示例： FE FE 04 3B 08 C9 FA

电位值 = 2249

如何计算电位值

电位值 = 电位值低位 + 电位值高位 * 256

1. 发送六个舵机的电位值

数据域	说明	数据
Data[0]	识别帧	0XFE
Data[1]	识别帧	0XFE
Data[2]	数据长度帧	0X15
Data[3]	指令帧	0X3C
Data[4]	1号舵机电位值高字节	encoder_1_high
Data[5]	1号舵机电位值低字节	encoder_1_low
Data[6]	2号舵机电位值高字节	encoder_2_high
Data[7]	2号舵机电位值低字节	encoder_2_low
Data[8]	3号舵机电位值高字节	encoder_3_high
Data[9]	3号舵机电位值低字节	encoder_3_low
Data[10]	4号舵机电位值高字节	encoder_4_high
Data[11]	4号舵机电位值低字节	encoder_4_low
Data[12]	5号舵机电位值高字节	encoder_5_high
Data[13]	5号舵机电位值低字节	encoder_5_low
Data[14]	6号舵机电位值高字节	encoder_6_high
Data[15]	6号舵机电位值低字节	encoder_6_low
Data[16]	指定速度	Sp
Data[17]	结束帧	0XFA

发送所有电机的电位值

串口发送示例： FE FE 15 3C 00 00 00 00 00 00 00 00 00 00 00 00 20 FA

（参考上方发送单独电位值）

encoder_1_high：数据类型byte

计算方式： 1号舵机电位值先转换为int类型 再取十六进制高字节

encoder_1_low：数据类型byte

计算方式： 1号舵机电位值先转换为int类型 再取十六进制低字节

（其余同理）

Sp：数据类型byte 取值范围0~100%

无返回值

---

1. 读取速度

数据域	说明	数据
Data[0]	识别帧	0XFE
Data[1]	识别帧	0XFE
Data[2]	数据长度帧	0X02
Data[3]	指令帧	0X40
Data[4]	结束帧	0XFA

串口发送示例： FE FE 02 40 FA

返回数据结构

数据域	说明	数据
Data[0]	返回识别帧	0XFE
Data[1]	返回识别帧	0XFE
Data[2]	返回数据长度帧	0X03
Data[3]	返回指令帧	0X40
Data[4]	指定速度	Sp
Data[5]	结束帧	0XFA

串口返回示例：FE FE 03 40 32 FA

---

1. 设置速度

数据域	说明	数据
Data[0]	识别帧	0XFE
Data[1]	识别帧	0XFE
Data[2]	数据长度帧	0X02
Data[3]	指令帧	0X41
Data[4]	指定速度	sp
Data[5]	结束帧	0XFA

串口发送示例：FE FE 02 41 32 FA

无返回值

---

1. 读取FeedOverride（暂未开放）

数据域	说明	数据
Data[0]	识别帧	0XFE
Data[1]	识别帧	0XFE
Data[2]	数据长度帧	0X02
Data[3]	指令帧	0X42
Data[4]	结束帧	0XFA

无返回值

---

1. 读取加速度（暂未开放）

数据域	说明	数据
Data[0]	识别帧	0XFE
Data[1]	识别帧	0XFE
Data[2]	数据长度帧	0X02
Data[3]	指令帧	0X44
Data[4]	结束帧	0XFA

---

1. 读取关节最小角度

数据域	说明	数据
Data[0]	识别帧	0XFE
Data[1]	识别帧	0XFE
Data[2]	数据长度帧	0X03
Data[3]	指令帧	0X4A
Data[4]	关节舵机序号	Joint_number
Data[5]	结束帧	0XFA

读取

串口发送示例： FE FE 03 4A 00 FA

joint_no取值范围 0~5

返回数据结构

数据域	说明	数据
Data[0]	返回识别帧	0XFE
Data[1]	返回识别帧	0XFE
Data[2]	返回数据长度帧	0X04
Data[3]	返回指令帧	0X4A
Data[4]	舵机角度值高位	Angle_high
Data[5]	舵机角度值低位	Angle_low
Data[6]	结束帧	0XFA

串口返回示例： FE FE 04 4A 01 44 FA

角度 = 90

如何得出关节最小角度

temp = angle1_low+angle1_high*256

Angle1=（temp \ 33000 ?(temp – 65536) : temp）/10

计算方式：角度值低位 + 角度高位值乘以256 先判断是否大于33000 如果大于33000就再减去65536 最后除以10 如果小于33000就直接除以10

---

1. 读取关节最大角度

数据域	说明	数据
Data[0]	识别帧	0XFE
Data[1]	识别帧	0XFE
Data[2]	数据长度帧	0X03
Data[3]	指令帧	0X4B
Data[4]	关节舵机序号	joint_number
Data[5]	结束帧	0XFA

串口发送示例： FE FE 03 4B 01 FA

返回数据结构

数据域	说明	数据
Data[0]	返回识别帧	0XFE
Data[1]	返回识别帧	0XFE
Data[2]	返回数据长度帧	0X04
Data[3]	返回指令帧	0X4B
Data[4]	舵机角度值高位	Angle_high
Data[5]	舵机角度值低位	Angle_low
Data[6]	结束帧	0XFA

串口返回示例： FE FE 04 4B 01 44 FA

joint_no取值范围 0~5

如何得出关节最大角度

temp = angle1_low+angle1_high*256

Angle1=（temp \ 33000 ?(temp – 65536) : temp）/10

计算方式：角度值低位 + 角度高位值乘以256 先判断是否大于33000 如果大于33000就再减去65536 最后除以10 如果小于33000就直接除以10

---

1. 查看连接

数据域	说明	数据
Data[0]	识别帧	0XFE
Data[1]	识别帧	0XFE
Data[2]	数据长度帧	0X03
Data[3]	指令帧	0X50
Data[4]	结束帧	0XFA

串口发送示例：FE FE 03 50 FA

返回数据结构

数据域	说明	数据
Data[0]	返回识别帧	0XFE
Data[1]	返回识别帧	0XFE
Data[2]	返回数据长度帧	0X03
Data[3]	指令帧	0X50
Data[4]	连接/未连接	0X01/0X00
Data[5]	结束帧	0XFA

串口返回示例：FE FE 03 50 00 FA

1. 查看舵机是否全部上电

数据域	说明	数据
Data[0]	识别帧	0XFE
Data[1]	识别帧	0XFE
Data[2]	数据长度帧	0X02
Data[3]	指令帧	0X51
Data[4]	结束帧	0XFA

串口发送示例：FE FE 02 51 FA

返回数据结构

数据域	说明	数据
Data[0]	返回识别帧	0XFE
Data[1]	返回识别帧	0XFE
Data[2]	返回数据长度帧	0X03
Data[3]	指令帧	0X51
Data[4]	上电/未上电	0X01/0X00
Data[5]	结束帧	0XFA

串口返回示例：FE FE 03 51 00 FA

1. 读取伺服参数

数据域	说明	数据
Data[0]	识别帧	0XFE
Data[1]	识别帧	0XFE
Data[2]	数据长度帧	0X04
Data[3]	指令帧	0X53
Data[4]	关节舵机序号	joint_number
Data[5]	数据	data_id
Data[6]	结束帧	0XFA

读取1号舵机位置P比例参数

串口发送示例：FE FE 04 53 00 21 FA

joint_no取值范围 0~5

Data_id：数据类型byte

取值如下表 | 地址 | 功能 | 取值范围 | 初始值 | 取值解析 | |------|-------------|-----------|--------------------|--------------------| | 20 |LED报警 |0-254 | 0 |1\0 = 打开或关闭LED报警| | 21 |位置环P |0-254 |123关节8,456关节5 |控制电机的比例系数 | | 22 |位置环I |0-254 |123关节20,456关节13 |控制电机的微分系数 | | 23 |位置环D |0-254 |0 |控制电机的积分系数 | | 24 |最小启动力 |0-1000 |0 |设置最小输出力矩 1000 = 100%|

返回数据结构

数据域	说明	数据
Data[0]	返回识别帧	0XFE
Data[1]	返回识别帧	0XFE
Data[2]	返回数据长度帧	0X03
Data[3]	返回指令帧	0X53
Data[4]	返回数据	data_state
Data[5]	结束帧	0XFA

串口返回示例：FE FE 03 53 00 FA

---

1. 设置舵机零点

数据域	说明	数据
Data[0]	识别帧	0XFE
Data[1]	识别帧	0XFE
Data[2]	数据长度帧	0X03
Data[3]	指令帧	0X55
Data[4]	关节舵机序号	joint_number
Data[5]	结束帧	0XFA

串口发送示例：FE FE 03 55 00 FA

无返回值

---

1. 刹车单个电机(暂未开放)

数据域	说明	数据
Data[0]	识别帧	0XFE
Data[1]	识别帧	0XFE
Data[2]	数据长度帧	0X03
Data[3]	指令帧	0X56
Data[4]	关节舵机序号	joint_number
Data[5]	结束帧	0XFA

串口发送示例：FE FE 03 56 00 FA

无返回值

---

1. 设置atom引脚模式

数据域	说明	数据
Data[0]	识别帧	0XFE
Data[1]	识别帧	0XFE
Data[2]	数据长度帧	0X04
Data[3]	指令帧	0X60
Data[4]	引脚序号	pin_no
Data[5]	引脚模式	pin_mode
Data[6]	结束帧	0XFA

设置atom pin16为输出模式0

串口发送示例：FE FE 04 60 19 00 FA

Pin_no：数据类型byte

Pin_mode：数据类型byte

无返回值

---

1. 程序暂停运行

数据域	说明	数据
Data[0]	识别帧	0XFE
Data[1]	识别帧	0XFE
Data[2]	数据长度帧	0X02
Data[3]	指令帧	0X26
Data[4]	结束帧	0XFA

串口发送示例：FE FE 02 26 FA

无返回值

---

1. 程序继续运行

数据域	说明	数据
Data[0]	识别帧	0XFE
Data[1]	识别帧	0XFE
Data[2]	数据长度帧	0X02
Data[3]	指令帧	0X27
Data[6]	结束帧	0XFA

串口发送示例：FE FE 02 27 FA

无返回值

---

1. 程序停止运行

数据域	说明	数据
Data[0]	识别帧	0XFE
Data[1]	识别帧	0XFE
Data[2]	数据长度帧	0X02
Data[3]	指令帧	0X28
Data[4]	结束帧	0XFA

串口发送示例：FE FE 02 28 FA

无返回值

---

1. 设置舵机伺服参数

数据域	说明	数据
Data[0]	识别帧	0XFE
Data[1]	识别帧	0XFE
Data[2]	数据长度帧	0X05
Data[3]	指令帧	0X52
Data[4]	关节舵机序号	servo_no
Data[5]	数据地址	data_id
Data[6]	数据	data
Data[7]	结束帧	0XFA

设置1号舵机位置P比例参数为1

串口发送示例：FE FE 05 52 00 21 01 FA

无返回值

取值如下表 | 地址 | 功能 | 取值范围 | 初始值 | 取值解析 | |------|-------------|-----------|----------|---------------------|---------------------| | 20 |LED报警 |0-254 | 0 |1\0 = 打开或关闭LED报警| | 21 |位置环P |0-254 |123关节8,456关节5 |控制电机的比例系数 | | 22 |位置环I |0-254 |123关节20,456关节13 |控制电机的微分系数 | | 23 |位置环D |0-254 |0 |控制电机的积分系数 | | 24 |最小启动力 |0-1000 |0 |设置最小输出力矩 1000 = 100%|

---

1. 机器人自由模式(关闭所有扭力输出)

数据域	说明	数据
Data[0]	识别帧	0XFE
Data[1]	识别帧	0XFE
Data[2]	数据长度帧	0X02
Data[3]	指令帧	0X13
Data[4]	结束帧	0XFA

串口发送示例：FE FE 02 13 FA

无返回值

---

1. 设定atom屏幕RGB灯的颜色

数据域	说明	数据
Data[0]	识别帧	0XFE
Data[1]	识别帧	0XFE
Data[2]	数据长度帧	0X05
Data[3]	指令帧	0X6A
Data[4]	R	R
Data[5]	G	G
Data[6]	B	B
Data[7]	结束帧	0XFA

设置RGB为蓝色

串口发送示例：FE FE 05 6A 00 00 FF FA

无返回值

---

1. 设置夹爪角度

   请参考关节角度设置

---

1. 设置FeedOverride（暂未开放）

数据域	说明	数据
Data[0]	识别帧	0XFE
Data[1]	识别帧	0XFE
Data[2]	数据长度帧	0X04
Data[3]	指令帧	0X43
Data[4]	Feed_override高位	Feed_override_high
Data[5]	Feed_override低位	Feed_override_low
Data[6]	结束帧	0XFA

---

1. 设置加速度（暂未开放）

数据域	说明	数据
Data[0]	识别帧	0XFE
Data[1]	识别帧	0XFE
Data[2]	数据长度帧	0X04
Data[3]	指令帧	0X45
Data[4]	加速度值高位	acceleration_high
Data[5]	加速度值低位	acceleration_low
Data[6]	结束帧	0XFA

acceleration_high：数据类型byte

计算方式：加速度值乘以10 先转换为int格式 再取十六进制的高字节

acceleration_low：数据类型byte

计算方式：加速度值乘以10 先转换为int格式 再取十六进制的低字节

---

1. 设置关节最小角度

数据域	说明	数据
Data[0]	识别帧	0XFE
Data[1]	识别帧	0XFE
Data[2]	数据长度帧	0X05
Data[3]	指令帧	0X4C
Data[4]	关节舵机序号	Joint
Data[5]	角度值高位	Angle_high
Data[6]	角度值低位	Angle_low
Data[7]	结束帧	0XFA

设置最小角度为90

串口发送示例：FE FE 05 4C 00 01 44 FA

Joint 取值范围0~5

angle_high：数据类型byte

计算方式：角度值乘以10 先转换成int形式 再取十六进制的高字节

angle_low：数据类型byte

计算方式：角度值乘以10 先转换成int形式 再取十六进制的低字节

无返回值

---

1. 设置关节最大角度

数据域	说明	数据
Data[0]	识别帧	0XFE
Data[1]	识别帧	0XFE
Data[2]	数据长度帧	0X05
Data[3]	指令帧	0X4D
Data[4]	关节舵机序号	Joint
Data[5]	角度值高位	Angle_high
Data[6]	角度值低位	Angle_low
Data[7]	结束帧	0XFA

设置最大角度为90

串口发送示例：FE FE 05 4D 00 01 44 FA Joint 取值范围0~5

angle_high：数据类型byte

计算方式：角度值乘以10 先转换成int形式 再取十六进制的高字节

angle_low：数据类型byte

计算方式：角度值乘以10 先转换成int形式 再取十六进制的低字节

无返回值

---

1. 设置工具坐标系

数据域	说明	数据
Data[0]	识别帧	0XFE
Data[1]	识别帧	0XFE
Data[2]	数据长度帧	0X14
Data[3]	指令帧	0X81
Data[4]	X坐标高字节	x_high
Data[5]	X坐标低字节	x_low
Data[6]	Y坐标高字节	y_high
Data[7]	Y坐标低字节	y_low
Data[8]	Z坐标高字节	z_high
Data[9]	Z坐标低字节	z_low
Data[10]	RX坐标高字节	rx_high
Data[11]	RX坐标低字节	rx_low
Data[12]	RY坐标高字节	ry_high
Data[13]	RY坐标低字节	ry_low
Data[14]	RZ坐标高字节	rz_high
Data[15]	RZ坐标低字节	rz_low
Data[16]	结束帧	0XFA

设定工具坐标系（-14，-27，275，-89.5，0.7，-90.7），

串口发送示例：FE FE 14 81 FF 74 FE EE 0A C1 DD 05 00 48 DC 95 FA

x_high：数据类型byte

计算方式：x坐标乘以10 再取十六进制的高字节

x_low：数据类型byte

计算方式：x坐标乘以10 再取十六进制的低字节

（y轴坐标z轴坐标同理）

rx_high：数据类型byte

计算方式：rx坐标值乘以100 再取十六进制的高字节

rx_low：数据类型byte

计算方式：rx坐标值乘以100 再取十六进制的低字节

（ry轴坐标rz轴坐标同理）

无返回值

---

1. 设置世界坐标系

数据域	说明	数据
Data[0]	识别帧	0XFE
Data[1]	识别帧	0XFE
Data[2]	数据长度帧	0X14
Data[3]	指令帧	0X83
Data[4]	X坐标高字节	x_high
Data[5]	X坐标低字节	x_low
Data[6]	Y坐标高字节	y_high
Data[7]	Y坐标低字节	y_low
Data[8]	Z坐标高字节	z_high
Data[9]	Z坐标低字节	z_low
Data[10]	RX坐标高字节	rx_high
Data[11]	RX坐标低字节	rx_low
Data[12]	RY坐标高字节	ry_high
Data[13]	RY坐标低字节	ry_low
Data[14]	RZ坐标高字节	rz_high
Data[15]	RZ坐标低字节	rz_low
Data[16]	结束帧	0XFA

设定世界坐标系（-14，-27，275，-89.5，0.7，-90.7），

串口发送示例：FE FE 14 83 FF 74 FE EE 0A C1 DD 05 00 48 DC 95 FA

x_high：数据类型byte

计算方式：x坐标乘以10 再取十六进制的高字节

x_low：数据类型byte

计算方式：x坐标乘以10 再取十六进制的低字节

（y轴坐标z轴坐标同理）

rx_high：数据类型byte

计算方式：rx坐标值乘以100 再取十六进制的高字节 rx_low：数据类型byte

计算方式：rx坐标乘以100 先转换为int类型 再取十六进制低字节 （ry轴坐标rz轴坐标同理）

无返回值

---

1. 获取工具坐标系

数据域	说明	数据
Data[0]	识别帧	0XFE
Data[1]	识别帧	0XFE
Data[2]	数据长度帧	0X02
Data[3]	指令帧	0X82
Data[4]	结束帧	0XFA

串口发送示例：FE FE 02 82 FA

返回数据结构

数据域	说明	数据
Data[0]	返回识别帧	0XFE
Data[1]	返回识别帧	0XFE
Data[2]	返回数据长度帧	0X14
Data[3]	返回指令帧	0X82
Data[4]	X坐标高字节	x_high
Data[5]	X坐标低字节	x_low
Data[6]	Y坐标高字节	y_high
Data[7]	Y坐标低字节	y_low
Data[8]	Z坐标高字节	z_high
Data[9]	Z坐标低字节	z_low
Data[10]	RX坐标高字节	rx_high
Data[11]	RX坐标低字节	rx_low
Data[12]	RY坐标高字节	ry_high
Data[13]	RY坐标低字节	ry_low
Data[14]	RZ坐标高字节	rz_high
Data[15]	RZ坐标低字节	rz_low
Data[16]	结束帧	0XFA

串口返回示例：FE FE 14 82 FF 74 FE EE 0A C1 DD 05 00 48 DC 95 FA

x_high：数据类型byte

计算方式：x坐标乘以10 先转换为int类型 再取十六进制高字节

x_low：数据类型byte

计算方式：x坐标乘以10 先转换为int类型 再取十六进制低字节

（y轴坐标z轴坐标同理）

rx_high：数据类型byte

计算方式：rx坐标乘以100 先转换为int类型 再取十六进制高字节

rx_low：数据类型byte

计算方式：rx坐标乘以100 先转换为int类型 再取十六进制低字节

（ry轴坐标rz轴坐标同理）

---

1. 获取世界坐标系

数据域	说明	数据
Data[0]	识别帧	0XFE
Data[1]	识别帧	0XFE
Data[2]	数据长度帧	0X02
Data[3]	指令帧	0X84
Data[4]	结束帧	0XFA

串口发送示例：FE FE 02 84 FA

返回数据结构

数据域	说明	数据
Data[0]	返回识别帧	0XFE
Data[1]	返回识别帧	0XFE
Data[2]	返回数据长度帧	0X14
Data[3]	返回指令帧	0X84
Data[4]	X坐标高字节	x_high
Data[5]	X坐标低字节	x_low
Data[6]	Y坐标高字节	y_high
Data[7]	Y坐标低字节	y_low
Data[8]	Z坐标高字节	z_high
Data[9]	Z坐标低字节	z_low
Data[10]	RX坐标高字节	rx_high
Data[11]	RX坐标低字节	rx_low
Data[12]	RY坐标高字节	ry_high
Data[13]	RY坐标低字节	ry_low
Data[14]	RZ坐标高字节	rz_high
Data[15]	RZ坐标低字节	rz_low
Data[16]	结束帧	0XFA

串口返回示例：FE FE 14 84 FF 74 FE EE 0A C1 DD 05 00 48 DC 95 FA

1. 设置法兰基坐标系

数据域	说明	数据
Data[0]	识别帧	0XFE
Data[1]	识别帧	0XFE
Data[2]	数据长度帧	0X03
Data[3]	指令帧	0X85
Data[4]	RFType	0x00/0x01
Data[5]	结束帧	0XFA

串口发送示例：FE FE 03 85 00 FA

RFType：数据类型byte

取值范围： 0~1 BASE = 0； WORLD = 1；

RFType::BASE为将机器人基座作为基坐标，RFType::WORLD为将世界坐标系作为基坐标。

1. 获取法兰基坐标系

数据域	说明	数据
Data[0]	识别帧	0XFE
Data[1]	识别帧	0XFE
Data[2]	数据长度帧	0X02
Data[3]	指令帧	0X86
Data[4]	结束帧	0XFA

串口发送示例：FE FE 02 86 FA

返回数据结构

数据域	说明	数据
Data[0]	返回识别帧	0XFE
Data[1]	返回识别帧	0XFE
Data[2]	返回数据长度帧	0X03
Data[3]	返回指令帧	0X86
Data[4]	RFType	0x00/0x01
Data[5]	结束帧	0XFA

串口返回示例：FE FE 03 86 00 FA

1. 设置末端坐标系

数据域	说明	数据
Data[0]	返回识别帧	0XFE
Data[1]	返回识别帧	0XFE
Data[2]	返回数据长度帧	0X03
Data[3]	返回指令帧	0X89
Data[4]	EndType	0x00/0x01
Data[5]	结束帧	0XFA

串口发送示例：FE FE 03 89 00 FA

无返回值

EndType：数据类型byte

取值范围： 0~1 FLANGE = 0； TOOL = 1；

EndType::FLANGE为将末端设置为法兰，EndType::TOOL为将末端设置为工具末端。

1. 获取末端坐标系

数据域	说明	数据
Data[0]	识别帧	0XFE
Data[1]	识别帧	0XFE
Data[2]	数据长度帧	0X02
Data[3]	指令帧	0X8A
Data[4]	结束帧	0XFA

串口发送示例：FE FE 02 8A FA

返回数据结构

数据域	说明	数据
Data[0]	返回识别帧	0XFE
Data[1]	返回识别帧	0XFE
Data[2]	返回数据长度帧	0X03
Data[3]	返回指令帧	0X8A
Data[4]	EndType	0x00/0x01
Data[5]	结束帧	0XFA

串口返回示例：FE FE 03 8A 00 FA

---

1. 单个电机掉电

数据域	说明	数据
Data[0]	识别帧	0XFE
Data[1]	识别帧	0XFE
Data[2]	数据长度帧	0X03
Data[3]	指令帧	0X56
Data[4]	舵机序号	Servo_no
Data[5]	结束帧	0XFA

让一号舵机掉电

串口发送示例：FE FE 03 56 01 FA

无返回值

---

1. 单个电机上电

数据域	说明	数据
Data[0]	识别帧	0XFE
Data[1]	识别帧	0XFE
Data[2]	数据长度帧	0X02
Data[3]	指令帧	0X57
Data[4]	舵机序号	Servo_no
Data[5]	结束帧	0XFA

给一号舵机上电

串口发送示例：FE FE 03 57 01 FA

无返回值

---

1. 设置Atom IO口电平状态

数据域	说明	数据
Data[0]	识别帧	0XFE
Data[1]	识别帧	0XFE
Data[2]	数据长度帧	0X04
Data[3]	指令帧	0X61
Data[4]	引脚序号	Pin_no
Data[5]	电平信号	0X00/0X01
Data[6]	结束帧	0XFA

设置引脚P22为高电平

串口发送示例：FE FE 04 61 22 01 FA

无返回值

---

1. 读取Atom IO口电平状态

数据域	说明	数据
Data[0]	识别帧	0XFE
Data[1]	识别帧	0XFE
Data[2]	数据长度帧	0X03
Data[3]	指令帧	0X62
Data[4]	引脚序号	pin_no
Data[5]	结束帧	0XFA

读取引脚P22电平状态

串口发送示例：FE FE 03 62 22 FA

返回数据结构

数据域	说明	数据
Data[0]	返回识别帧	0XFE
Data[1]	返回识别帧	0XFE
Data[2]	返回数据长度帧	0X03
Data[3]	返回指令帧	0X62
Data[4]	电平状态	0X00/0X01
Data[5]	结束帧	0XFA

---

1. 设置Atom引脚PWM输出

数据域	说明	数据
Data[0]	识别帧	0XFE
Data[1]	识别帧	0XFE
Data[2]	数据长度帧	0X06
Data[3]	指令帧	0X64
Data[4]	通道	pin_no
Data[5]	晶振频率_高	freq_high
Data[6]	晶振频率_低	freq_low
Data[7]	占空比	pin_val
Data[8]	结束帧	0XFA

设置atom 23号引脚 输出晶振频率为10000，占空比为50%的PWM波形

串口发送示例：FE FE 06 64 23 27 10 32 FA

无返回值

---

1. 读取夹爪角度

请参考读取关节角度，夹爪为7号关节舵机

---

1. 设置夹爪模式

数据域	说明	数据
Data[0]	识别帧	0XFE
Data[1]	识别帧	0XFE
Data[2]	数据长度帧	0X04
Data[3]	指令帧	0X66
Data[4]	夹爪开合	0X00/0X01
Data[5]	速度	Sp
Data[6]	结束帧	0XFA

设置夹爪以20的速度张开

串口发送示例：FE FE 04 66 00 20 FA

无返回值

---

1. 设置夹爪角度

数据域	说明	数据
Data[0]	识别帧	0XFE
Data[1]	识别帧	0XFE
Data[2]	数据长度帧	0X05
Data[3]	指令帧	0X67
Data[4]	角度值高位	Angle_high
Data[5]	角度值低位	Angle_low
Data[6]	速度	Sp
Data[7]	结束帧	0XFA

串口发送示例：

无返回值

---

1. 夹爪设置零点

数据域	说明	数据
Data[0]	识别帧	0XFE
Data[1]	识别帧	0XFE
Data[2]	数据长度帧	0X02
Data[3]	指令帧	0X68
Data[4]	结束帧	0XFA

设置夹爪当前位置为零点

串口发送示例：FE FE 02 68 FA

---

1. 检测夹爪是否运动

数据域	说明	数据
Data[0]	识别帧	0XFE
Data[1]	识别帧	0XFE
Data[2]	数据长度帧	0X02
Data[3]	指令帧	0X69
Data[4]	结束帧	0XFA

设置夹爪当前位置为零点

串口发送示例：FE FE 02 69 FA

返回数据结构

数据域	说明	数据
Data[0]	返回识别帧	0XFE
Data[1]	返回识别帧	0XFE
Data[2]	返回数据长度帧	0X03
Data[3]	指令帧	0X69
Data[4]		0X00/0X01
Data[5]	结束帧	0XFA

附录：

在ATOM库和运动学库中添加了相应的坐标变换程序，具体实现方式如下所述：

1. 改变末端坐标系
2. 通过setEndType和getEndType函数可以设置末端坐标系，EndType::FLANGE为将末端设置为法兰，EndType::TOOL为将末端设置为工具末端。
3. 通过setToolReference和getToolReference函数可以设置读取工具的坐标信息。设置时是以法兰坐标系为相对坐标系，工具末端信息是相对于法兰坐标系的。
4. 将EndType设置为FLANGE后，GetCoords和WriteCoords方法均以法兰位置计算。
5. 将EndType设置为TOOL后，GetCoords和WriteCoords方法均以工具末端位置计算。
6. 改变基坐标系
7. 通过setReferenceFrame函数可以设置基坐标系，RFType::BASE为将机器人基座作为基坐标，RFType::WORLD为将世界坐标系作为基坐标。
   getReferenceFrame函数为读取当前基坐标系种类。
8. 通过setWorldReference和getWorldReference函数可以设置读取基坐标系信息。设置时是以世界坐标系为相对坐标系，输入机器人的基座相对于世界坐标系的位置信息。
9. 当基坐标系为基座时，GetCoords和WriteCoords方法均以基座为参考坐标系。
10. 当基坐标系为世界坐标系时，GetCoords和WriteCoords方法均以世界坐标系作为参考坐标系。

通信相关更改（暂时）

现增加末端坐标系的设置与读取，世界坐标系的设置与读取，当前参考坐标系的设置与读取，末端类型的设置与读取，移动方式的设置与读取，机械臂信息的发送接收。

这些通信暂时设置为0x80至0x8A

在ParameterList.h文件中新增roboticMessages空间用于添加机械臂通信信息，现只暂时增加“没有逆解”的提示，后续可陆续增加。

MOVEL功能简单设计思想如下：

求出初始点位和目标点位之间的欧式距离，以欧式距离为基准，每隔10mm插入一个插值点，如果插值点没有逆解，搜索位置不变三个方向姿态正负PI/30的临近空间内是否有逆解，主要是避免奇异值以及一些恰好不能求出解的特殊位置。

MOVEL和JOG的点位发送间隔时间改为动态时间，根据两点之间最大关节移动距离计算移动时间，再讲该移动时间减去特定时长作为时间间隔。

7 周边产品使用

myCobot的周边产品包括 末端执行器、底座 和 配件

末端执行器

- 平行夹爪
- 自适应夹爪
- 张角式夹爪
- 吸泵

底座

- G型底座
- 平面底座

配件

- 摇杆
- 电池盒

7.1 末端执行器

myCobot现已经批量生产的末端执行器有

- 平行夹爪: 夹取物体使用
- 吸盘吸泵: 吸附物体使用

7.1.1 夹爪

夹爪

- 1、 用乐高科技件将夹爪固定在机械臂顶部，连接到M5 Basic的末端执行器扩展接口（在开箱视频中有介绍）
- 2、 所有的开发环境都可以使用夹爪，如ROS、Arduino、UIFlow、Roboflow等

适用物体

- 小方块
- 小球
- 长条物体

技巧 可以在夹爪指尖粘贴橡胶，以获得更好的摩擦力。

API控制 您可以通过下载最新的myCobot API进行夹爪控制。

1 控制夹爪张开闭合程度：请先进行夹爪位置的读取，再设置范围。取值范围1500 - 2000；实际范围为2000附近。

```
setEncoorder()
// the gripper encoder number(from 0 to 4096) get gripper value before use
```

2 设置夹爪的初始点：请先完全张开夹爪，设定后对应电位约为2048。

```
setGripperIni()
// set gripper encode initial center point(to be 2048)
```

3 读取夹爪位置

```
getGripperValue()
// return gripper encdoer value (from 0 to 4096)
```

4 设置夹爪状态：现阶段只支持张开闭合两个状态，如果需要设置精准位置，可以设置夹爪闭合程度；

```
setGripperState()
// only used for adaptive gripper, 0 or 1 for open and close
```

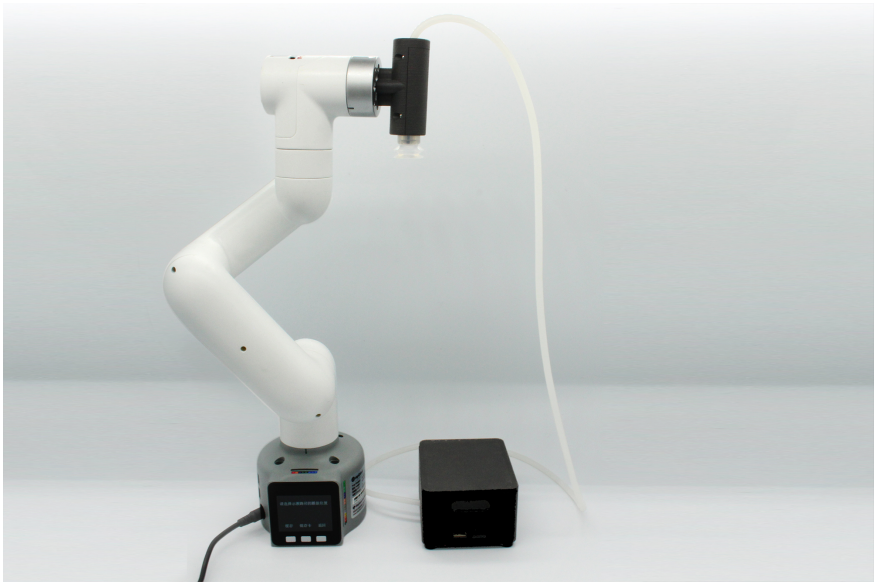
7.1.2 吸盘

适用物体

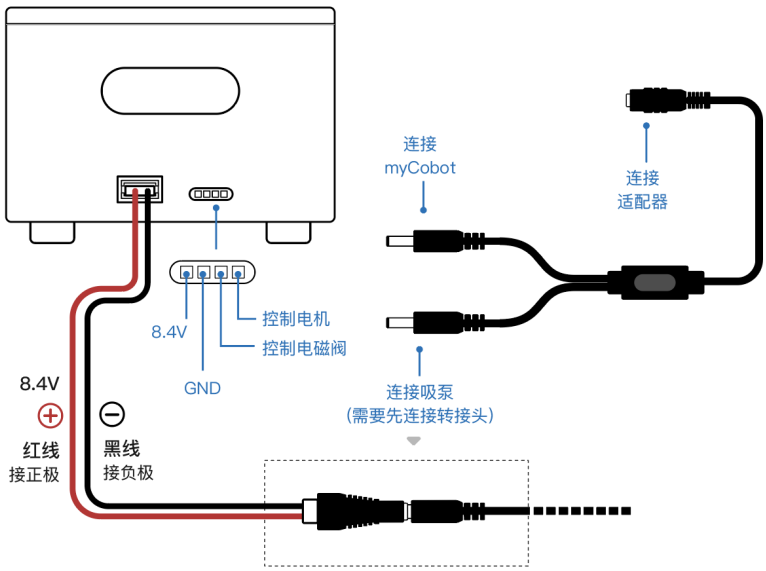
- 纸片/塑料片
- 平面光滑物体
- 卡片等

产品图示

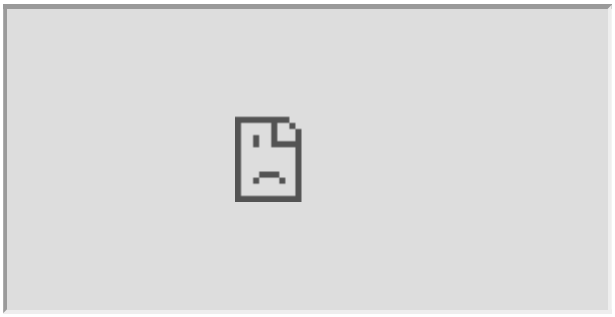
安装示意图



接线示意图



产品安装教程视频



无法观看请点击安装教程观看

注意事项

- 请确保产品已经按照说明连接成功
- 请确保产品供电为附带适配器供电
- 请确保正负极的接入方向

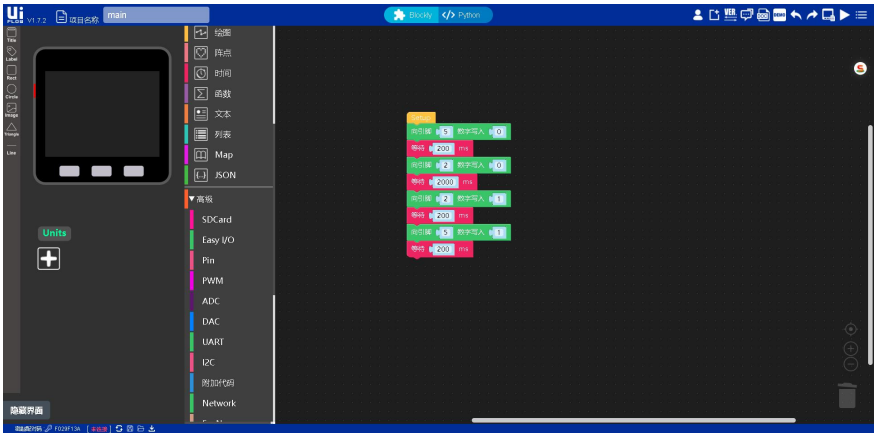
Arduino 环境下使用吸泵

1. 将吸泵的引脚与Baisc引脚连接到一起
    - 电磁阀控制引脚链接 Basic - G2 引脚
    - 泵机控制引脚链接 Basic - G5 引脚
2. 新建一个Arduino程序，将下列内容复制：

```
//
// 请确认电磁阀连接G2引脚,泵机链接G5引脚
// 连接完成后，高电平关闭，低电平打开
//

void setup() {
  // put your setup code here, to run once:
  Serial.begin(9600);//打开串口，波特率9600
  pinMode(2,OUTPUT); //设定引脚G2为输出状态
  pinMode(5,OUTPUT); //设定引脚G5为输出状态
  delay(100);
  digitalWrite(2,1);//将引脚2设为高电平，关闭电磁阀
  digitalWrite(5,1);//将引脚5设为高电平，关闭泵机
}

void loop() {
  // 使用时按照需求控制电磁阀与泵机
  digitalWrite(5,0);//将引脚5设为低电平，打开泵机
  delay(200);//延时200ms
  digitalWrite(2,0);//将引脚2设为低电平，打开电磁阀
  delay(2000);//延时2000ms,松开吸住的物体
  digitalWrite(2,1);//将引脚2设为高电平，关闭电磁阀
  delay(200);//延时200ms
  digitalWrite(5,1);//将引脚5设为高电平，关闭泵机
  delay(200);//延时200ms
}
```

UIFlow环境下使用吸泵

1. 将吸泵的引脚与Baisc引脚连接到一起
    - 电磁阀控制引脚链接 Basic - G2 引脚
    - 泵机控制引脚链接 Basic - G5 引脚
2. 新建一个UIFlow程序，在程序中添加以下内容：



7.1.3 笔夹

正在开发编写中...

7.2 底座

myCobot现支持底座有

- G型底座: 方便固定在桌边
- 吸盘底座: 方便直接固定在平滑桌面上

7.2.1 G型底座

G型底座 - 固定在桌子边沿

- 1.将两头的羊角螺丝拧紧，并在头部套入胶套
- 2.用G形夹将底座固定在桌子边沿
- 3.用附带的乐高科技件，连接底座和机械臂底部
- 4.确定稳固后方可开始使用

7.2.2 吸盘底座

适用平整光滑表面

- 1.在底座的四角安装吸盘并拧紧
- 2.用附带的乐高科技件，连接平面底座和机械臂底部
- 3.将四个吸盘固定在平整光滑平面后方可开始使用

技巧

可以适当在吸盘下加入少量**不导电**液体，以填补吸盘与桌面的缝隙，以获得最佳吸附效果。

7.3 配件

正在开发编写中...

8 机器视觉开发

什么是图像识别？

1. 图像识别的原理
    - 图像识别，是指利用计算机对图像进行处理、分析和理解，以识别各种不同模式的目标和对象的技术，是应用深度学习算法的一种实践应用。
2. 图像识别的应用场景
    - 现阶段图像识别技术一般分为人脸识别与商品识别，人脸识别主要运用在安全检查、身份核验与移动支付中；商品识别主要运用在商品流通过程中，特别是无人货架、智能零售柜等无人零售领域。
3. 图像识别的人工智能应用
    - 图像识别是人工智能的一个重要领域。为了编制模拟人类图像识别活动的计算机程序，人们提出了不同的图像识别模型。例如模板匹配模型。这种模型认为，识别某个图像，必须在过去的经验中有这个图像的记忆模式，又叫模板。当前的刺激如果能与大脑中的模板相匹配，这个图像也就被识别了。例如有一个字母A,如果在脑中有个A模板,字母A的大小、方位、形状都与这个A模板完全一致,字母A就被识别了。这个模型简单明了，也容易得到实际应用。但这种模型强调图像必须与脑中的模板完全符合才能加以识别，而事实上人不仅能识别与脑中的模板完全一致的图像，也能识别与模板不完全一致的图像。例如,人们不仅能识别某一个具体的字母A,也能识别印刷体的、手写体的、方向不正、大小不同的各种字母A。同时,人能识别的图像是大量的，如果所识别的每一个图像在脑中都有一个相应的模板，也是不可能的。
    - 为了解决模板匹配模型存在的问题，格式塔心理学家又提出了一个原型匹配模型。这种模型认为，在长时记忆中存储的并不是所要识别的无数个模板，而是图像的某些“相似性”。从图像中抽象出来的“相似性”就可作为原型，拿它来检验所要识别的图像。如果能找到一个相似的原型，这个图像也就被识别了。这种模型从神经上和记忆探寻的过程上来看，都比模板匹配模型更适宜，而且还能说明对一些不规则的，但某些方面与原型相似的图像的识别。但是，这种模型没有说明人是怎样对相似的刺激进行辨别和加工的，它也难以在计算机程序中得到实现。因此又有人提出了一个更复杂的模型，即“泛魔”识别模型。
    - 一般工业使用中，采用工业相机拍摄图片，然后利用软件根据图片灰阶差做处理后识别出有用信息，图像识别软件国外代表的有康耐视等，国内代表的有图智能等。
4. 图像识别的发展历程
    - 图像识别的发展经历了三个阶段：文字识别、数字图像处理与识别、物体识别。文字识别的研究是从 1950年开始的，一般是识别字母、数字和符号，从印刷文字识别到手写文字识别， 应用非常广泛。
    - 数字图像处理和识别的研究开始于1965年。数字图像与模拟图像相比具有存储，传输方便可压缩、传输过程中不易失真、处理方便等巨大优势，这些都为图像识别技术的发展提供了强大的动力。物体的识别主要指的是对三维世界的客体及环境的感知和认识，属于高级的计算机视觉范畴。它是

以数字图像处理与识别为基础的结合人工智能、系统学等学科的研究方向，其研究成果被广泛应用在各种工业及探测机器人上。现代图像识别技术的一个不足就是自适应性能差，一旦目标图像被较强的噪声污染或是目标图像有较大残缺往往就得不出理想的结果。

- 图像识别问题的数学本质属于模式空间到类别空间的映射问题。目前，在图像识别的发展中，主要有三种识别方法：统计模式识别、结构模式识别、模糊模式识别。图像分割是图像处理中的一项关键技术，自20世纪70年代，其研究已经有几十年的历史，一直都受到人们的高度重视，至今借助于各种理论提出了数以千计的分割算法，而且这方面的研究仍然在积极地进行着。
- 现有的图像分割的方法有许多种，有阈值分割方法，边缘检测方法，区域提取方法，结合特定理论工具的分割方法等。从图像的类型来分有：灰度图像分割、彩色图像分割和纹理图像分割等。早在1965年就有人提出了检测边缘算子，使得边缘检测产生了不少经典算法。但在近二十年间，随着基于直方图和小波变换的图像分割方法的研究计算技术、VLSI技术的迅速发展，有关图像处理方面的研究取得了很大的进展。图像分割方法结合了一些特定理论、 方法和工具，如基于数学形态学的图像分割、基于小波变换的分割、基于遗传算法的分割等。

使用StivckV + Maixpy - IDE 进行图像识别开发

开发平台

- Maixpy - IDE

开发环境

- Windows
- Linux

开发组件

- M5Stack - StickV


2019
M5StickV
M5STACK
OV7740
640X480
R G B W
SPK
BUTTON B
135x240
TF
BUTTON A
M3
HOLE
AXP192
OV7740
BATTERY
MPU6886
MAX98357
FLASH 16M
ST7789 IPS LCD
USB-C
POWER
SWITCH
2S=ON
6S=OFF
K210
AI
CAMERA


M5STACK

描述

M5Stick-V RISC-V AI摄像头

M5Stick-V是一款内置Kendryte K210的AIOT（AI + IOT）摄像头，集成双核64位RISC-V CPU和最先进的神经网络处理器边缘计算片上系统（SoC）

M5stickV AI摄像头具备机器视觉能力，配备了OmniVision OV7740图像传感器，采用了OmniPixel®3-HS技术，提供了同类最佳的低光灵敏度，支持多种视觉识别能力的它（如实时获取被检测目标的大小）与坐标•实时获取被检测目标的种类），并且能够在低危情况下进行卷积神经网络计算，因此M5StickV会是一个很好的零门机器机器视觉嵌入式解决方案，支持MicroPython开发环境，这可以让您在使用M5stick-V上进行项目开发时，程序代码将会更加精简。

产品特性

- 双核64位RISC-V RV64IMAFDC（RV64GC）CPU / 400Mhz（普通）
- 双精度FPU
- 神经网络处理器（KPU）/ 0.8排名
- 底层IO架构（FPIOA）
- 双硬件512点16位复数FFT
- SPI，I2C，UART，I2S，RTC，PWM，定时器支持
- AES，SHA256加速器
- 直接内存存取控制器（DMAC）
- 支持Micropython
- 固件加密支持

应用

- 面部识别/检测
- 物体检测/分类
- 实时获取目标的大小和坐标
- 实时获取检测到的目标类型
- 形状识别
- 视频/显示
- 游戏模拟器

常见驱动问题

- M5StickV 主控在部分系统中，可能无法免驱工作，用户可以通过手动安装FTDI驱动修复该问题。

主控资源	参数
肯德利K210	双核64位RISC-V RV64IMAFDC（RV64GC）CPU / 400Mhz（普通）
静态随机存取存储器	8米
闪光	1600万
输入电压	5V @ 500mA
KPU神经网络参数大小	5.5MiB-5.9MiB
主机接口	TypeC x 1，GROVE（I2C + I / 0 + UART）x 1
RGB LED	RGBW x 1
按键	自定义按键x 2
IPS屏幕	1.14 TFT，135 * 240，ST7789
摄像头	OV7740
视场	55°
电源管理单元	AXP192
锂电池	200毫安
外部存储	TF卡（microSD）
MEMS六轴传感器	MPU6886
净重	23克
毛重	82克
产品尺寸	48 *24* 22毫米
包装尺寸	144 *44* 43毫米
外壳材质	塑胶（PC）

TF卡（microSD）测试

- M5StickV当前并不能识别所有类型的TF卡（microSD），我们对一些常见的TF卡进行了测试，测试结果如下。

品牌	内存	类型	传输速度	分区格式	测试结果
金斯顿	8G	HC	4类	FAT32	好的
金斯顿	16G	HC	10类	FAT32	好的
金斯顿	32G	HC	10类	FAT32	不
金斯顿	64G	XC	10类	FAT文件	好的
闪迪	16G	HC	10类	FAT32	好的
闪迪	32G	HC	10类	FAT32	好的
闪迪	64G	XC	10类	/	不
闪迪	128G	XC	10类	/	不
夏克	16G	HC	10类	FAT32	还可以（紫色）
夏克	32G	HC	10类	FAT32	好的
夏克	64G	XC	10类	/	不
突耶	32G	HC	10类	/	不

功能描述

肯德利**K210**

Kendryte K210是集成机器视觉能力的系统级芯片（SoC）。使用台积电（TSMC）超低功耗的28纳米先进制程，具有双核64位处理器，拥有更好的性能，稳定性和可靠性。该方案力求零门生物学开发，可在最短时效部署于用户的产品中，实现产品人工智能。

- 具备机器视觉能力
- 更好的低视觉视觉处理速度与准确率
- 内置卷积人工神经网络硬件加速器，KPU，可高效进行卷积人工神经网络运算
- 台积电28nm先进制程，温度范围-40°C到125°C，稳定可靠
- 支持固件加密，难以使用普通方法破解
- 独特的地理位置IO布局，使产品设计更加灵活
- 低电压，与相同处理能力的系统合并具有较高的可靠性
- 3.3V / 1.8V双电压支持，无需重新转换，节省成本

中央处理器

本芯片搭载基于RISC-V ISA的双核心64位的高性能低功耗CPU，具有以下特性：

- 核心数量：双核处理器
- 处理器位宽：64位CPU 400MHz
- 标称频率：400MHz
- 指令集扩展：IMAFDC
- 浮点处理单元（FPU）：双精度
- 平台中断管理
- 本地中断管理
- 指令缓存：32KiB x 2
- 数据缓存：32KiB x 2
- 片上SRAM：8MiB

OV7740

- 支持输出格式：RAW RGB和YUV
- 支持图像尺寸：VGA，QVGA，CIF或其他更小尺寸
- 支持太阳黑子消除
- 支持内部和外部帧同步
- 标准SCCB串行接口
- 数字视频端口（DVP）并行输出接口
- 嵌入式一次性燃料（OTP）存储器
- 片上锁相环（PLL）
- 用于内核的嵌入式1.5 V稳压器
- 尺寸：656 x 488
- 电源：-内核：1.5VDC±5％-模拟：3.3V±5％-I / O：1.7～3.47V
- 温度范围：-工作：-30°C至70°C-稳定图像：0°C至50°C
- 输出格式：-8/10位原始RGB数据-8位YUV
- 镜头尺寸：1/5“
- 输入时钟频率：6～27 MHz
- 最大图像传输速率：VGA（640x480）：60 fps-QVGA（320 x 240）：120 fp
- 灵敏度：6800 mV /（Lux-sec）
- 最大曝光间隔：502 x tROW
- 分辨率尺寸：4.2μm×4.2μm
- 图像面积：2755.2μm×2049.6μm
- 封装/管芯尺寸：-CSP3：4185μm×4345μm-COB：4200μm×4360μm

MAX98357

- 单电源工作（2.5V至5.5V）
- 3.2W输出功率：4Ω，5V
- 2.4mA静态电流
- 效率92％（RL =8Ω，POUT = 1W）
- 22.8μVRMS输出噪声（AV = 15dB）

- 1kHz时，0.015％THD + N
- 无需MCLK
- 8kHz至96kHz采样速率
- 支持左声道，右声道以及（左声道/ 2 +右声道/ 2）输出
- 成熟的边沿速度控制调整D类放大器输出校正
- 1kHz下，具有77dB PSRR
- 低RF敏感度，可抑制GSM发射的TDMA噪声
- 喀嗒声抑制电路

AXP192

- 工作电压：2.9V-6.3V（AMR：-0.3V～15V）
- 可配置的智能电源选择系统
- 自适应USB或AC适配器输入的电流和电压限制
- 内部理想二极管的电阻低于100mΩ

MPU6886

- 陀螺仪功能
- 数字输出X，Y和Z轴角速度传感器（陀螺仪），用户放置满量程范围为±250 dps，±500 dps，±1000 dps和±2000 dps，集成式16位ADC
- 数字光纤低通滤波器
- 低功率陀螺仪仪操作
- 工厂校准的灵敏度比例因子
- 镜头尺寸：1/5“
- 自我测试

加速度计功能

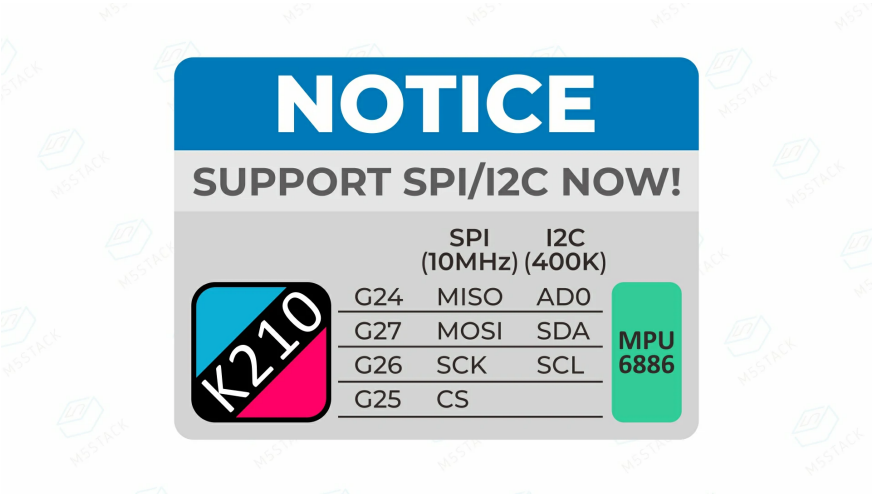
- 数字输出X，Y和Z轴加速度计，连续满量程范围为±2g，±4g，±8g和±16g，集成16位ADC
- 用户中断中断
- 唤醒动作中断，用于应用处理器的低功耗操作
- 自我测试

SPI / I2C双通信模式

> 注意事项：当前M5Stack发布的M5StickV存在两种版本，用户编程使用时需根据其对应的转换映射进行不同的配置，具体区别如下。

- I2C单模式（蓝色PCB）版本的M5StickV电路设计中，MPU6886仅支持用户配置其通信模式为I2C，其映射映射为SCL-28，SDA-29。

- SPI / I2C双模式（黑色PCB）版本的M5StickV电路设计中，MPU6886支持用户配置其通信模式为SPI或I2C，其映射为SCL-26，SDA-27。，使用时，可通过切换CS引脚踩来切换模式（高允许1为I2C模式，依次为0为SPI模式）
- 具体细节映射如下图所示：



相关链接

数据表

- MPU6688
- SH200Q

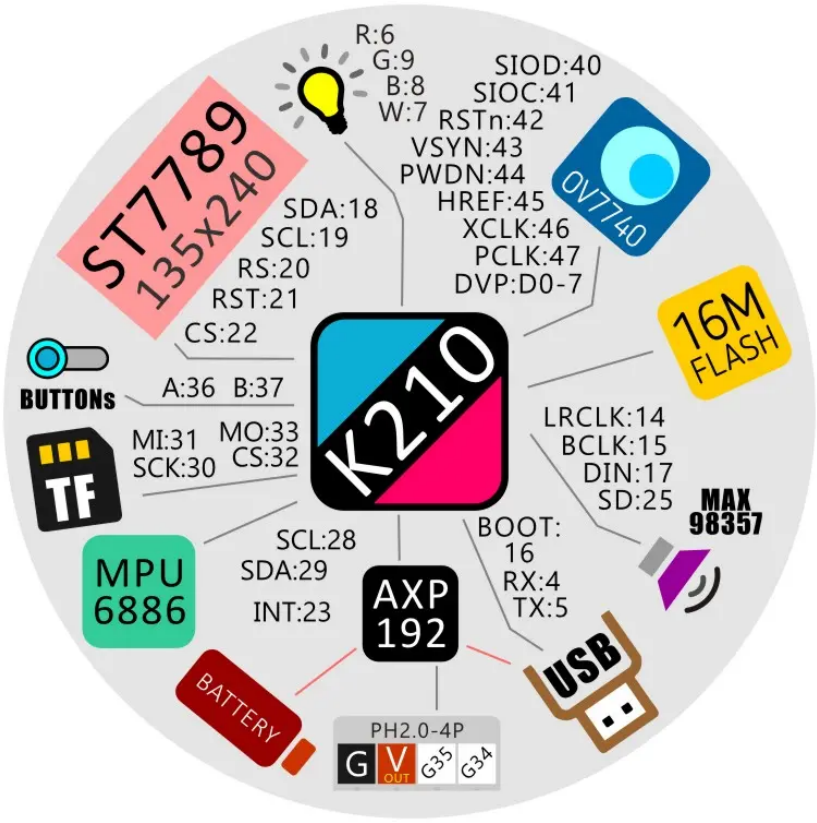
网页

- sipeed

GITHUB

- API

原理图



K210-CAM

按理程序

- Maixpy参考示例

8.1 设置MaixPy环境

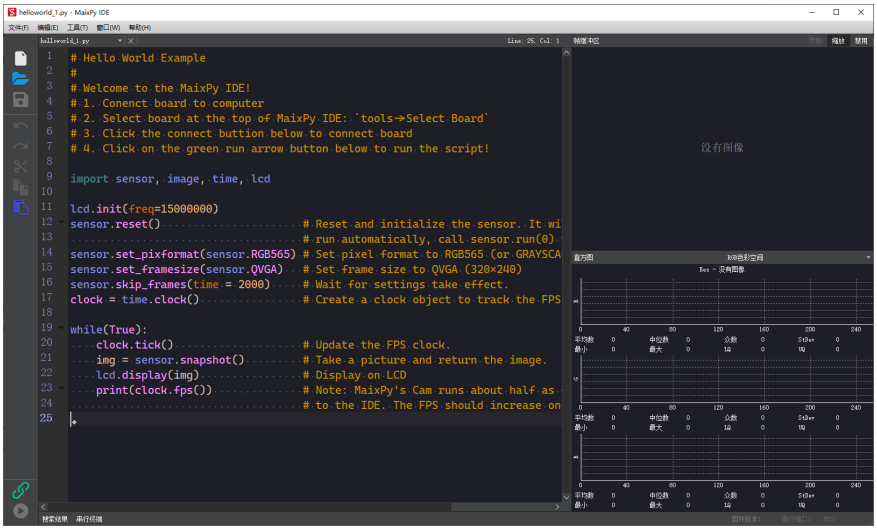
1.1 什么是 MaixPy?

MaixPy 是将 Micropython 移植到 **K210**（一款 64 位双核带硬件 FPU、卷积加速器、FFT、Sha256 的 RISC-V CPU ） 的一个项目， 支持 **MCU** 常规操作， 更集成了硬件加速的 **AI** 机器视觉和麦克风阵列，**1TOPS** 算力 核心模块却不到￥50，以快速开发具有极低成本和体积实用的 **AIOT** 领域智能应用。

首先需要弄清： MaixPy 使用 Micropython 脚本语法，所以不像 C 语言 一样需要编译，其实不用 IDE 也能愉快使用： 使用串口终端工具，前面已经安装了

使用 IDE 则会方便在电脑上实时编辑脚本并上传到开发板以及直接在开发板上执行脚本，以及在电脑上实时查看摄像头图像、保存文件到开发板等

当然， 使用 IDE 因为压缩、传输需要耗费一部分资源，所以性能会有所降低，而且如果MaixPy宕机也没有串口终端好发现问题



了解更多请点击官网浏览官方说明。

1.2 MaixPy 可以用来作什么？

- 人脸识别
- 物体识别

- 颜色识别
- 情绪识别
- 车牌识别
- 分拣系统

了解更多请点击官网浏览官方说明。

2.如何搭建使用环境？

2.1 开发环境准备

2.1.1 安装驱动

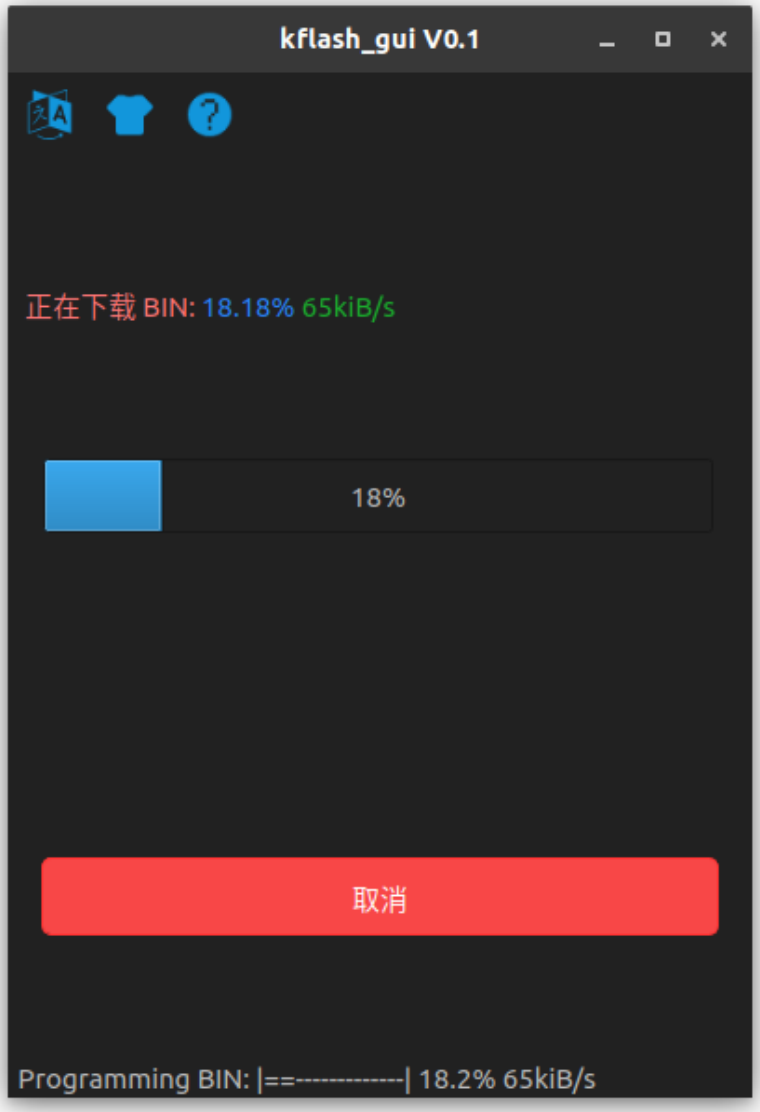
- 正式使用 MaixPy 之前，我们需要先安装好串口驱动，才可进行下一步的开发与使用；因为板子是通过 USB 转串口设备与电脑连接（K210 没有 USB 硬件支持功能）。 根据板子的 USB 转串口芯片型号装驱动。
- 点击下载并安装串口驱动
- 在 Linux 或者 Mac 下操作串口， 如果不想每次都使用 sudo 命令， 执行 sudo usermod -a -G dialout $(whoami) 将自己添加到 dialout 用户组即可，可能需要注销或者重启才能生效

2.1.2 烧录固件

- 要使用 MaixPy IDE 固件必须是 v0.3.1 版本以上, 否则MaixPyIDE 上会连接不上， 使用前尽量检查固件版本和 IDE 版本，都更新到最新版以保障能正常使用
- 点击下载 固件烧录器Kflsh, 下载完成会得到一个压缩包 kflash_gui 是跨平台的，可以在多个系统下工作（包括 Windows、Linux、MacOS、甚至树莓派） 使用勘智（Kendryte）的Windows版本可能部分开发版无法下载成功，请使用 kflash_gui 这个软件来下载
- 点击下载固件
- 烧录固件

下载完成后，使用固件烧录器烧录固件到M5StickV：


kflash_gui V1.6
固件
选择文件
inimum_with_ide_support.bin
0x00000
打开文件
增加文件
打包为 kfpkg
合并成.bin
开发板设置
开发板
自动选择
下载到
Flash
串口设置
端口
COM1 (USB-SERIAL CH340 (COM1))
波特率
1500000
速度模式
高速模式
下载
SiPEED
设置并点击下载按钮



2.2 下载安装软件

2.2.1 点击下载 MaixPy-IDE

```
注意：文件列表等说明 请看 最新版本文件夹下的 readme.txt 文件， 如果下载速度慢请使用 cdn 链接
```

安装软件

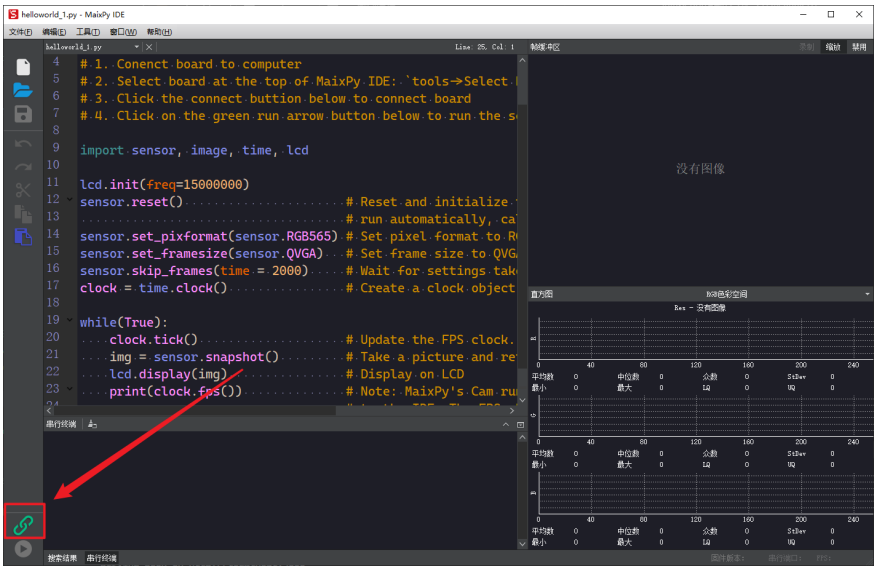
下载好的文件，Windows直接双击exe文件运行安装程序

测试连接

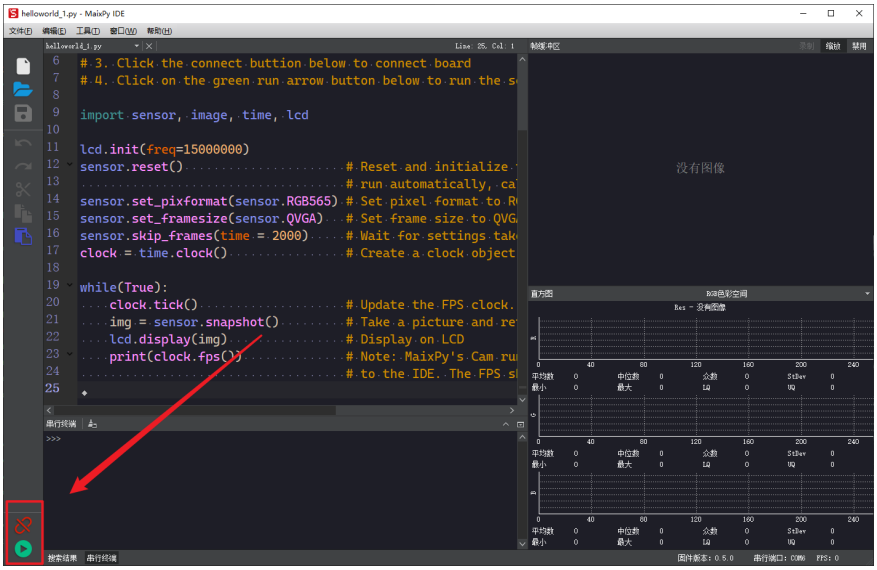
打开 MaixPy IDE, 上方工具栏里面选择开发板的型号。请选择 M5StickV 进行连接。

Tool-> Select Board (工具->选择开发板)

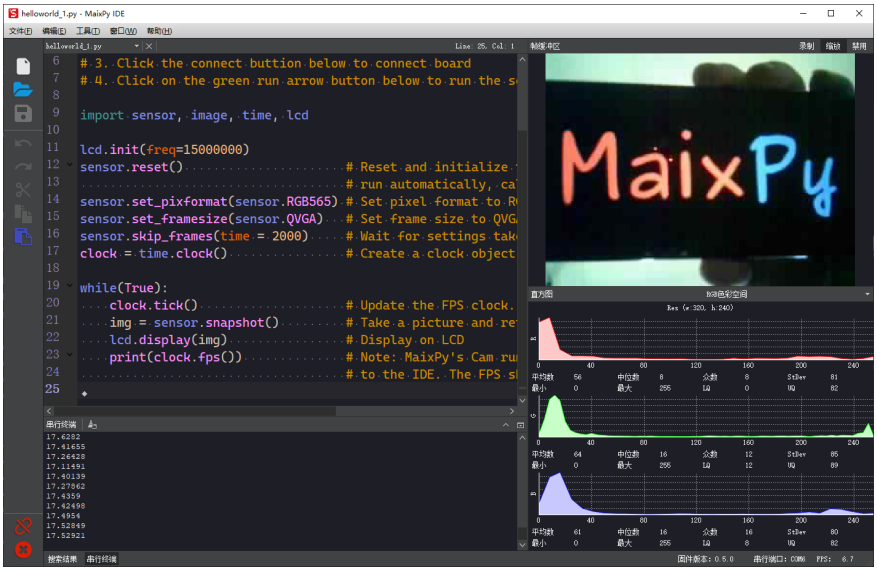
点击 connect 连接 MaixPy IDE



连接成功后，会由绿色变成红色



连接按钮下方是运行按钮，会执行当前编辑区的py文件.



再次点击运行按钮(红色), 停止运行当前代码.

3. 关于 USB 串口的疑难杂症排查

如果没有看到串口，请按如下顺序排查硬件问题。

- 插入电脑，是否存在叮咚一声，如插入 U 盘时USB 驱动加载的声音，没有表示硬件上的串口芯片出问题了。
- 更换线材重试，更换电脑 USB 口重试，仍然加载不出来，更换电脑确认。

如果没有办法烧录固件，请按如下顺序排查硬件问题。

- 使用串口工具查看硬件当中是否存在 maixpy 固件
- 设置 115200 波特率连接串口，按复位键（RST）接收到芯片的数据，不管是什么都表示串口芯片工作正常，如果没有则表示硬件异常。
- 基于上述再进行烧录固件，烧录前，按硬件的 BOOT 键后按复位，再松开 BOOT 键，此时烧录正常进行，如果没有则表示 Flash 受损，可以尝试烧录到 SRAM ，如果烧录失败，则表示串口芯片异常。
- 如果到这里了，还是不能解决问题，则硬件确实存在缺陷

3.1 K210 的烧写机制介绍

我们常把这个称为一键下载电路，表示能够轻松的通过控制 串口的 RST 和 DTR 的完成对 BOOT 和 RST 引脚的控制进入烧录模式，如上描述的期望硬件电路自动完成最初由人按下 BOOT 后按 RST 的操作，这与硬件实现强相关，基于此，再进行 TX 和 RX 的数据传输，所以实际上我们需要用到 UART 串口的功能引脚。

在 Kflash 中分多种版型多种烧写方式的触发，我们可以简单分为几类，低速的 115200 和 高速的 1500000 波特率，以这两类波特率所匹配的烧录方式为差异点，如果发现下载过程中失败，可以适当的降低波特率，这是由于串口芯片工作不稳定

导致，而工具中对版型选择只是会影响第一段烧录模式的触发，而在这之后的烧写固件中就会采取配置的波特率进行烧写，通常不超过与flash的通信烧录速度，常见于 50~60 KB/S。

如果发现无论如何更换烧录模式都无法进入，要么是烧录版型不匹配，要么是串口芯片的 DTR RST 引脚出了问题（物理上的）

8.2 颜色识别

开发前提

- 设备链接完成
- 固件烧录完成
- 软件环境完成

示例代码

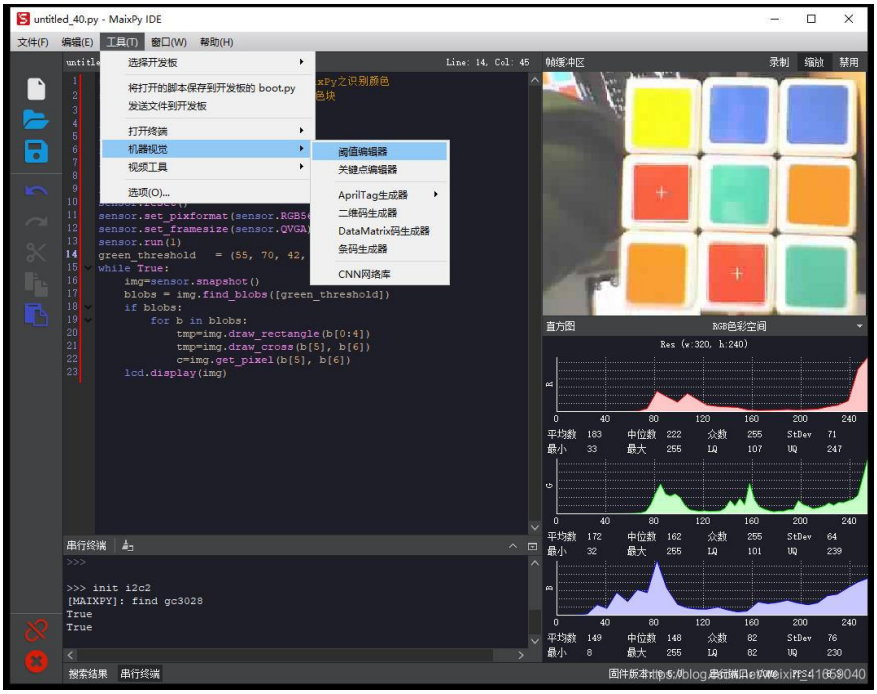
```
import sensor
import image
import lcd
import time

from Maix import GPIO
from fpioa_manager import fm, board_info
from machine import UART

clock = time.clock()

fm.register(34, fm.fpioa.UART2_TX, force=True)
fm.register(35, fm.fpioa.UART2_RX, force=True)
uart_Port = UART(UART.UART2, 115200,8,0,0, timeout=1000, read_buf_len= 4096)

lcd.init()
sensor.reset()
sensor.set_pixformat(sensor.RGB565)
sensor.set_framesize(sensor.QVGA)
sensor.run(1)
lcd.rotation(2)

#blue,red,green,yellow

colour = ['blue','red','green','yellow']

colour_threshold =([27, 55, -18, 4, -39, -20],
                   [55, 64, 14, 52, -9, 3],
                   [29, 65, -109, -18, 36, 59],
                   [30, 68, -25, -10, 52, 87])
blobs = [0,0,0,0]

def blobs_output(blobs):
    for b in blobs:
        tmp=img.draw_rectangle(b[0:4])
        tmp=img.draw_cross(b[5], b[6])
        img.draw_string(b[5], b[6], colour[i],color=(255,0,0), scale=2)
        c=img.get_pixel(b[5], b[6])
        _colour = {}
        _colour['colour'] = colour[i]
        data_ = []
        data_.append(_colour)
        data_.append(blobs[0])
    uart_Port.write(str(blobs))

def show_fps():
    fps =clock.fps()
    img.draw_string(200, 1, ("%2.2ffps" %(fps)),color=(255,0,0), scale=2)

while True:
    clock.tick()
    img=sensor.snapshot()
    show_fps()

    for i in range(4):
        blobs[i] = img.find_blobs([colour_threshold[i]],area_threshold=100,pixels_thre
        if blobs[i]:
            blobs_output(blobs[i])

    lcd.display(img)
```

功能解析

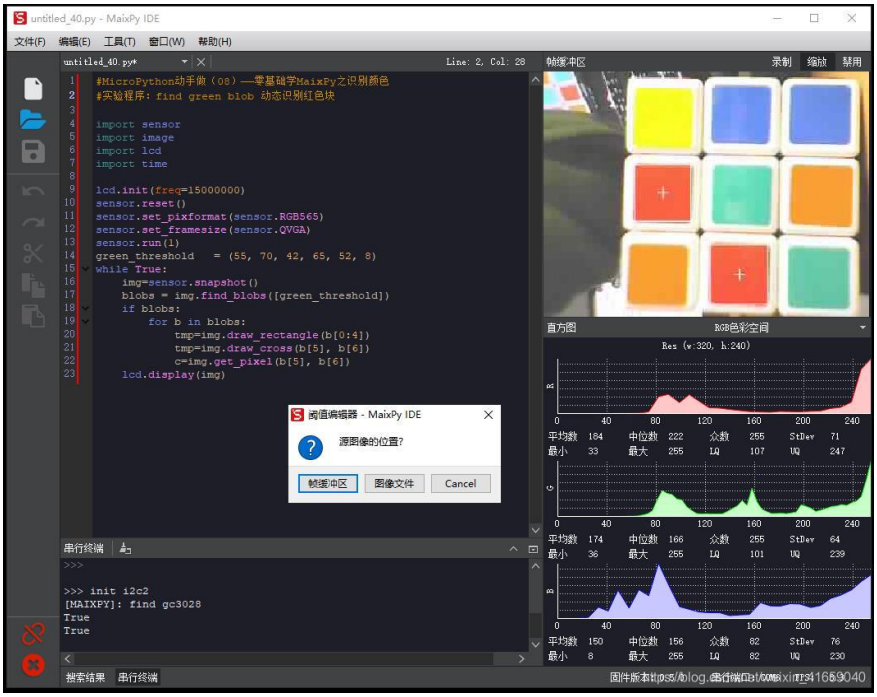
- 识别四种颜色
- 串口输出数据
- 显示屏显示识别到的颜色色块

定义你想识别的颜色

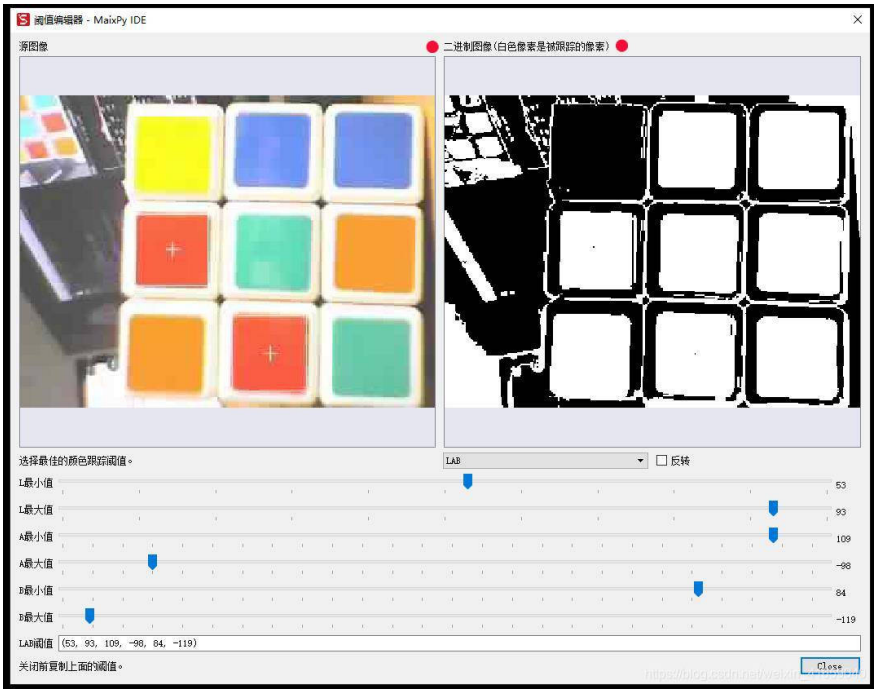
打开MaixPy IDE，选择工具——机器视觉——阈值编辑器



打开源图像位置，选择帧缓冲区



调整LAB阈值，主要是在二进制图像栏，白色像素是被跟踪的像素



相关知识

MaixPy 机械视觉 API

彻底搞懂Lab 颜色空间

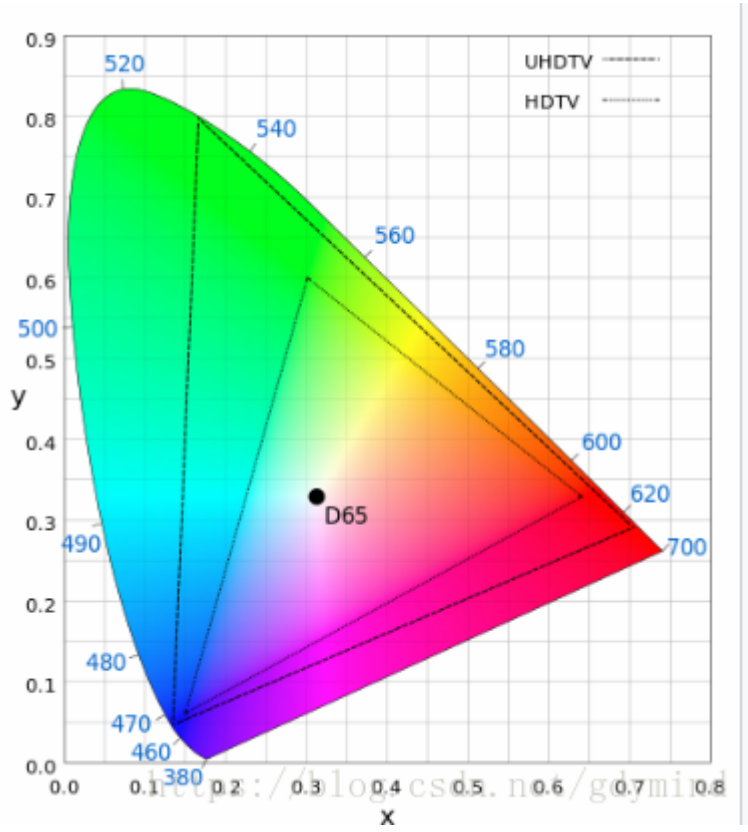
- 名称
  在开始之前，先明确一下Lab颜色空间（Lab color space）的名字：
  Lab的全称是CIELAB，有时候也写成CIE Lab*
  这里的CIE代表International Commission on Illumination（国际照明委员会），它是一个关于光照、颜色等的国际权威组织。

- 通道 Lab是由一个亮度通道（channel）和两个颜色通道组成的。在Lab颜色空间中，每个颜色用L、a、b三个数字表示，各个分量的含义是这样的：
  L代表亮度
  a代表从绿色到红色的分量
  b代表从蓝色到黄色的分量

- Perceptual uniform
  Lab是基于人对颜色的感觉来设计的，更具体地说，它是感知均匀（perceptual uniform）的。Perceptual uniform的意思是，如果数字（即前面提到的L、a、b这三个数）变化的幅度一样，那么它给人带来视觉上的变化幅度也差不多。Lab相较于RGB与CMYK等颜色空间更符合人类视觉，也更容易调整：想要调节亮度（不考虑Helmholtz–Kohlrausch effect，见下注）就调节L通道，想要调节只色彩平衡就分别调a和b。
  注：Helmholtz–Kohlrausch effect是人眼的一种错觉——当色彩饱和度高时，颜色会看起来更亮。

- 设备无关

  Lab有个很好的特性——设备无关（device-independent）。也就是说，在给定了颜色空间白点（white point）（下图中表示了一种颜色空间的白点）之后，这个颜色空间就能明确地确定各个颜色是如何被创建和显示的，与使用的显示介质没有关系。



  这么牛X的特性不用肯定浪费啊，举个典型的栗子，当你想把屏幕上的RGB图片转成打印用的CMYK图片的时候，就可以先将它从RGB转成LAB，然后再把LAB图片转成CMYK模式。我们可以放心大胆滴这么做，因为LAB的色域（gamut）比RGB和CMYK都要大（Lab色域很大，有一大部分已经超出了人类视觉范围，也就不能称之为“颜色”了）。 需要注意的是，Lab定义的是相对于白点的颜色，只有定义完白点是什么颜色（比如定义为CIE standard illuminant D50），我们才能知道其他的颜色。

- 数值范围

  理论上说，L、a、b都是实数，不过实际一般限定在一个整数范围内：
  L越大，亮度越高。L为0时代表黑色，为100时代表白色。
  a和b为0时都代表灰色。
  a从负数变到正数，对应颜色从绿色变到红色。
  b从负数变到正数，对应颜色从蓝色变到黄色。
  我们在实际应用中常常将颜色通道的范围-100~+100或-128127之间。

- 可视化

  可以看到，Lab*一共有三个分量，因此可以在三维空间中呈现。 在二维空间中，常用chromaticity diagram来可视化它，也就是固定亮度L，看a和b的变化。注意，这些可视化不是精确的，只是能帮助人理解。

- CIELUV

  有一个颜色空间和CIELAB很像，叫 CIE 1976 (L, *u*, v) *，也叫CIELUV。这个颜色空间的*L是和CIELAB一样的，但颜色分量不一样。

- LAB和RGB、CMYK之间的转换
  由于RGB和CMYK都是设备相关的，因此不能直接和LAB互相转换。所以在转换之前，必须定义一种绝对的颜色空间，比如sRGB或者Adobe RGB。 从RGB转到sRGB是设备相关的，但之后的变换是设备无关的。

8.3 形状识别

开发前提

- 设备链接完成
- 固件烧录完成
- 软件环境完成

示例代码

```
# 识别矩形
import sensor
import image
import lcd
import time
from Maix import GPIO
from fpioa_manager import fm, board_info
from machine import UART
clock = time.clock()
fm.register(34, fm.fpioa.UART2_TX, force=True)
fm.register(35, fm.fpioa.UART2_RX, force=True)
uart_Port = UART(UART.UART2, 115200,8,0,0, timeout=1000, read_buf_len= 4096)
lcd.init()
sensor.reset()
sensor.set_pixformat(sensor.RGB565)
sensor.set_framesize(sensor.QQVGA)
sensor.run(1)
lcd.rotation(2)
def uart_write(i):
    data_ = []
    data_.append(blobs[i])
    uart_Port.write(str(data_))
while True:
    img=sensor.snapshot()
    RECT = img.find_rects(threshould = 10000)
    if RECT:
        for b in RECT:
            img.draw_rectangle(b.rect(),color =(255,0,0))
            for p in b.corners():
                img.draw_circle(p[0],p[1],3,color = (0,255,0))
            c=img.get_pixel(b[0], b[1])
    lcd.display(img)
```

```
# 识别圆形
import sensor, image, time, lcd
sensor.reset()
sensor.set_pixformat(sensor.RGB565)
sensor.set_framesize(sensor.QVGA)
sensor.skip_frames(time = 300)
sensor.run(1)
lcd.init()
lcd.rotation(2)
clock = time.clock()
while(True):
    clock.tick()
    img = sensor.snapshot()
    for c in img.find_circles(threshold = 1800, x_margin = 20, y_margin = 20, r_margir
            r_min = 5, r_max = 45, r_step = 20):
        img.draw_circle(c.x(), c.y(), c.r(), color = (255, 0, 0))#识别到的红色圆形用红色
    print("r %f" % c.r())
    print("FPS %f" % clock.fps())
    lcd.display(img)
```

功能解析

- 识别指定的形状
- 返回对应的数据

相关知识

MaixPy 机械视觉 API

8.4 人脸识别

开发前提

- 设备链接完成
- 固件烧录完成
- 软件环境完成

烧录指定固件

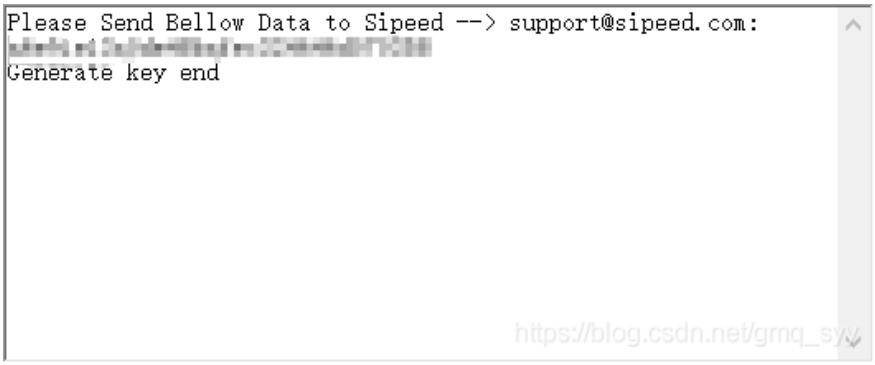
我们使用MaixPy IDE 来跑这个例程，首先要从MaixHub下载我们所需要的模型 从MaixHub上下载模型需要我们提供机器码，所以我们首先下载可以获取机器码的bin文件烧录到开发板中 ken_gen bin文件地址

获取机器码

用**kfalsh_gui** 烧录**ken_gen.bin**文件，点击下载下载即可

在串口输出中查看机器码

固件烧录完成后，链接电脑，打开串口助手，打开串口即可看到

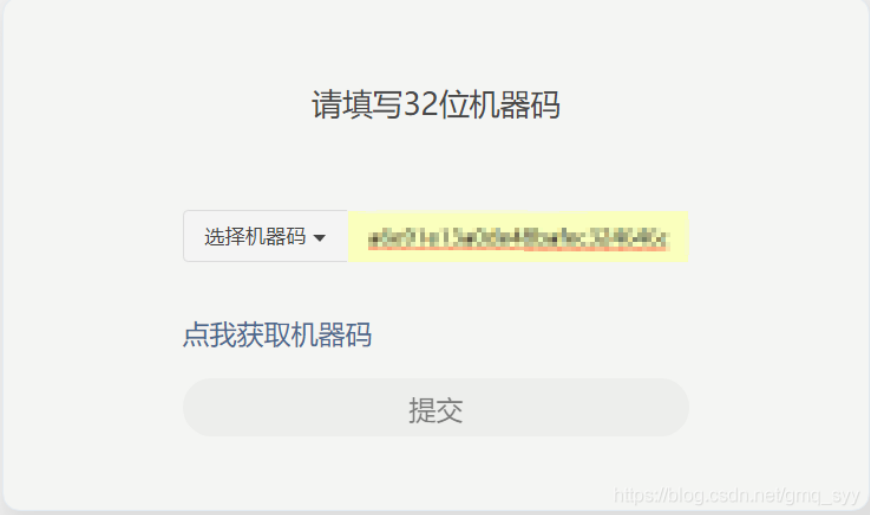


下载模型

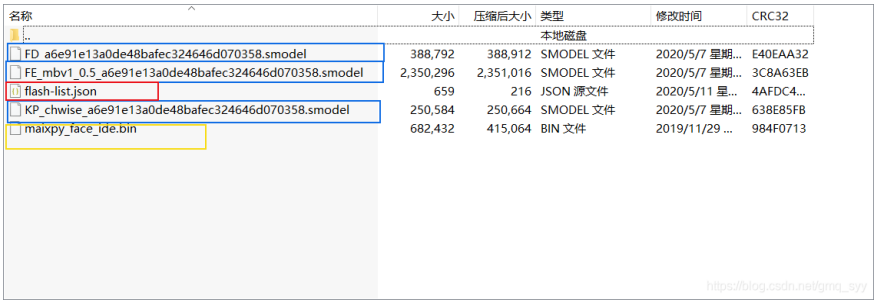
有了机器码我们就可以开始下载对应的模型文件了

模型下载地址

模型下载地址



提取的文件列表：



名称	大小	压缩后大小	类型	修改时间	CRC32
..			本地磁盘		
FD_a6e91e13a0de48bafec324646d070358.smodel	388,792	388,912	SMODEL 文件	2020/5/7 星期...	E40EAA32
FE_mbv1_0.5_a6e91e13a0de48bafec324646d070358.smodel	2,350,296	2,351,016	SMODEL 文件	2020/5/7 星期...	3C8A63EB
flash-list.json	659	216	JSON 源文件	2020/5/11 星...	4AFDC4...
KP_chwise_a6e91e13a0de48bafec324646d070358.smodel	250,584	250,664	SMODEL 文件	2020/5/7 星期...	638E85FB
maixpy_face_ide.bin	682,432	415,064	BIN 文件	2019/11/29 ...	984F0713

画蓝框的是模型文件，黄框是MaixPy的bin文件，这个是精简版的，体积也比较小只有600多Kb,json文件里面是配置，是关于这几个文件都分别要下载到flash的什么位置以及是否需要校验的

```
{
    "version": "0.1.0",
    "files": [
        {
            "address": 0,
            "bin": "maixpy_face_ide.bin",
            "sha256Prefix": true
        },
        {
            "address": 5242880,
            "bin": "FD_a6e91e13a0de48bafec324646d070358.smodel",
            "sha256Prefix": false
        },
        {
            "address": 6291456,
            "bin": "KP_chwise_a6e91e13a0de48bafec324646d070358.smodel",
            "sha256Prefix": false
        },
        {
            "address": 7340032,
            "bin": "FE_mbv1_0.5_a6e91e13a0de48bafec324646d070358.smodel",
            "sha256Prefix": false
        }
    ]
}
```

烧录模型文件

把kfpkg文件用kfalsh_gui下载到我们的开发板运行即可


kflash_gui V1.5
选择文件
10/Sipeed-Maix/face_dector.kfpkg
打开文件
增加文件
打包为 kfpkg
合并成.bin
开发板设置
开发板
Sipeed Maix Dock
下载到
Flash
串口设置
端口
COM9 (USB-SERIAL CH340 (C
波特率
1500000
速度模式
高速模式
下载
下载取消
https://blog.csdn.net/gmq_syy

运行示例代码

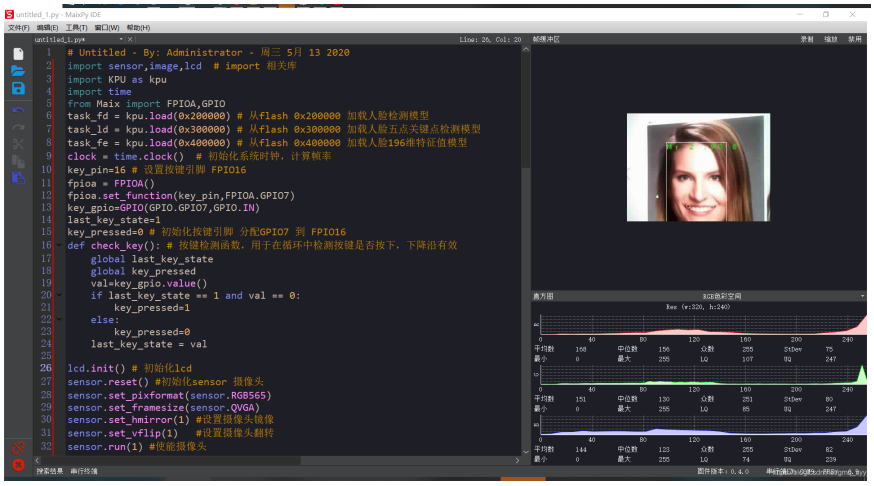
代码地址

```
import sensor,image,lcd  # import 相关库
import KPU as kpu
import time
from Maix import FPIOA,GPIO
task_fd = kpu.load(0x200000) # 从flash 0x200000 加载人脸检测模型
task_ld = kpu.load(0x300000) # 从flash 0x300000 加载人脸五点关键点检测模型
task_fe = kpu.load(0x400000) # 从flash 0x400000 加载人脸196维特征值模型
clock = time.clock()  # 初始化系统时钟，计算帧率
key_pin=16 # 设置按键引脚 FPIO16
fpioa = FPIOA()
fpioa.set_function(key_pin,FPIOA.GPIO36)
key_gpio=GPIO(GPIO.GPIO36,GPIO.IN)
last_key_state=1
key_pressed=0 # 初始化按键引脚 分配GPIO7 到 FPIO16
def check_key(): # 按键检测函数，用于在循环中检测按键是否按下，下降沿有效
    global last_key_state
    global key_pressed
    val=key_gpio.value()
    if last_key_state == 1 and val == 0:
        key_pressed=1
    else:
        key_pressed=0
    last_key_state = val

lcd.init() # 初始化lcd
sensor.reset() #初始化sensor 摄像头
sensor.set_pixformat(sensor.RGB565)
sensor.set_framesize(sensor.QVGA)
sensor.set_hmirror(1) #设置摄像头镜像
sensor.set_vflip(1)   #设置摄像头翻转
sensor.run(1) #使能摄像头
anchor = (1.889, 2.5245, 2.9465, 3.94056, 3.99987, 5.3658, 5.155437, 6.92275, 6.718375
dst_point = [(44,59),(84,59),(64,82),(47,105),(81,105)] #standard face key point posit
a = kpu.init_yolo2(task_fd, 0.5, 0.3, 5, anchor) #初始化人脸检测模型
img_lcd=image.Image() # 设置显示buf
img_face=image.Image(size=(128,128)) #设置 128 * 128 人脸图片buf
a=img_face.pix_to_ai() # 将图片转为kpu接受的格式
record_ftr=[] #空列表 用于存储当前196维特征
record_ftrs=[] #空列表 用于存储按键记录下人脸特征， 可以将特征以txt等文件形式保存到sd卡后，
names = ['Mr.1', 'Mr.2', 'Mr.3', 'Mr.4', 'Mr.5', 'Mr.6', 'Mr.7', 'Mr.8', 'Mr.9' , 'Mr.
while(1): # 主循环
    check_key() #按键检测
    img = sensor.snapshot() #从摄像头获取一张图片
    clock.tick() #记录时刻，用于计算帧率
    code = kpu.run_yolo2(task_fd, img) # 运行人脸检测模型，获取人脸坐标位置
    if code: # 如果检测到人脸
        for i in code: # 迭代坐标框
            # Cut face and resize to 128x128
            a = img.draw_rectangle(i.rect()) # 在屏幕显示人脸方框
            face_cut=img.cut(i.x(),i.y(),i.w(),i.h()) # 裁剪人脸部分图片到 face_cut
            face_cut_128=face_cut.resize(128,128) # 将裁出的人脸图片 缩放到128 * 128像素
            a=face_cut_128.pix_to_ai() # 将猜出图片转换为kpu接受的格式
            #a = img.draw_image(face_cut_128, (0,0))
            # Landmark for face 5 points
            fmap = kpu.forward(task_ld, face_cut_128) # 运行人脸5点关键点检测模型
            plist=fmap[:] # 获取关键点预测结果
            le=(i.x()+int(plist[0]*i.w() - 10), i.y()+int(plist[1]*i.h())) # 计算左眼位
            re=(i.x()+int(plist[2]*i.w()), i.y()+int(plist[3]*i.h())) # 计算右眼位置
            nose=(i.x()+int(plist[4]*i.w()), i.y()+int(plist[5]*i.h())) #计算鼻子位置
            lm=(i.x()+int(plist[6]*i.w()), i.y()+int(plist[7]*i.h())) #计算左嘴角位置
            rm=(i.x()+int(plist[8]*i.w()), i.y()+int(plist[9]*i.h())) #右嘴角位置
            a = img.draw_circle(le[0], le[1], 4)
            a = img.draw_circle(re[0], re[1], 4)
            a = img.draw_circle(nose[0], nose[1], 4)
            a = img.draw_circle(lm[0], lm[1], 4)
```

```
            a = img.draw_circle(rm[0], rm[1], 4) # 在相应位置处画小圆圈
            # align face to standard position
            src_point = [le, re, nose, lm, rm] # 图片中 5 坐标的位置
            T=image.get_affine_transform(src_point, dst_point) # 根据获得的5点坐标与标准
            a=image.warp_affine_ai(img, img_face, T) #对原始图片人脸图片进行仿射变换，变
            a=img_face.ai_to_pix() # 将正脸图像转为kpu格式
            #a = img.draw_image(img_face, (128,0))
            del(face_cut_128) # 释放裁剪人脸部分图片
            # calculate face feature vector
            fmap = kpu.forward(task_fe, img_face) # 计算正脸图片的196维特征值
            feature=kpu.face_encode(fmap[:]) #获取计算结果
            reg_flag = False
            scores = [] # 存储特征比对分数
            for j in range(len(record_ftrs)): #迭代已存特征值
                score = kpu.face_compare(record_ftrs[j], feature) #计算当前人脸特征值与
                scores.append(score) #添加分数总表
            max_score = 0
            index = 0
            for k in range(len(scores)): #迭代所有比对分数，找到最大分数和索引值
                if max_score < scores[k]:
                    max_score = scores[k]
                    index = k
            if max_score > 85: # 如果最大分数大于85， 可以被认定为同一个人
                a = img.draw_string(i.x(),i.y(), ("%s :%2.1f" % (names[index], max_sc
            else:
                a = img.draw_string(i.x(),i.y(), ("X :%2.1f" % (max_score)), color=(25
            if key_pressed == 1: #如果检测到按键
                key_pressed = 0 #重置按键状态
                record_ftr = feature
                record_ftrs.append(record_ftr) #将当前特征添加到已知特征列表
            break
    fps =clock.fps() #计算帧率
    print("%2.1f fps"%fps) #打印帧率
    a = lcd.display(img) #刷屏显示
    #kpu.memtest()

#a = kpu.deinit(task_fe)
#a = kpu.deinit(task_ld)
#a = kpu.deinit(task_fd)
```

用MaixPy IDE运行程序

程序运行如图



按BUTTON A键可以录入人脸，录入人脸后会按顺序分配名字，识别到后会显示出来

相关知识

待完善。。。

8.5 QR码识别

开发前提

- 设备链接完成
- 固件烧录完成
- 软件环境完成
- 使用MaixPy生成可识别的QR码
  选择 -> 工具 -> 机器视觉 -> AprilTag ->TAG36H11 生成可识别代码

示例代码

```
# AprilTags3D定位例程
#
# 这个例子展示了OpenMV Cam的功能，可以检测OpenMV Cam M7/H7 上的April标签。OpenMV2 M4版本无

import sensor, image, time, math, lcd

sensor.reset()
sensor.set_pixformat(sensor.RGB565)
sensor.set_framesize(sensor.QQVGA) # 如果分辨率太高，内存可能溢出
sensor.skip_frames(time = 2000)
#sensor.set_auto_gain(False)  # 必须关闭此功能，以防止图像冲刷…
#sensor.set_auto_whitebal(False)  # 必须关闭此功能，以防止图像冲刷…
clock = time.clock()
lcd.init()
lcd.rotation(2)

# 注意！与find_qrcodes不同，find_apriltags方法不需要对镜像进行镜头校正。

#标签系列有什么区别？ 那么，例如，TAG16H5家族实际上是一个4x4的方形标签。
#所以，这意味着可以看到比6x6的TAG36H11标签更长的距离。 然而，较低的H值（H5对H11）
#意味着4x4标签的假阳性率远高于6x6标签。 所以，除非你有理由使用其他标签系列，
#否则使用默认族TAG36H11。


# AprilTags库输出标签的姿势信息。 这是x / y / z平移和x / y / z旋转。
# x / y / z旋转以弧度表示，可以转换为度数。 至于翻译单位是无量纲的，
# 你必须应用一个转换函数。

# f_x是相机的x焦距。它应该等于以mm为单位的镜头焦距除以x传感器尺寸（以mm为单位）乘以图像中的像
# 以下数值适用于配备2.8毫米镜头的OV7725相机。

# f_y是相机的y焦距。它应该等于以mm为单位的镜头焦距除以y传感器尺寸（以mm为单位）乘以图像中的像
# 以下数值适用于配备2.8毫米镜头的OV7725相机。

# c_x是以像素为单位的图像x中心位置
# c_x是以像素为单位的图像x中心位置

f_x = (2.8 / 3.984) * 160 # find_apriltags 如果没有设置，则默认为这个
f_y = (2.8 / 2.952) * 120 # find_apriltags 如果没有设置，则默认为这个
c_x = 160 * 0.5 # find_apriltags 如果没有设置，则默认为这个 (the image.w * 0.5)
c_y = 120 * 0.5 # find_apriltags 如果没有设置，则默认为这个 (the image.h * 0.5)

def degrees(radians):
    return (180 * radians) / math.pi

while(True):
    clock.tick()
    img = sensor.snapshot()
    for tag in img.find_apriltags(fx=f_x, fy=f_y, cx=c_x, cy=c_y): # 默认为 TAG36H11
        tmp = img.draw_rectangle(tag.rect(), color = (255, 0, 0))
        tmp = img.draw_cross(tag.cx(), tag.cy(), color = (0, 255, 0))
        print_args = (tag.x_translation(), tag.y_translation(), tag.z_translation(), \
            degrees(tag.x_rotation()), degrees(tag.y_rotation()), degrees(tag.z_rotati
        # 变换单位不详。旋转单位是度数。
        print("Tx: %f, Ty %f, Tz %f, Rx %f, Ry %f, Rz %f" % print_args)
    print(clock.fps())
    lcd.display(img)
```

功能解析

- 识别指定的QR码
- 串口输出数据
- 显示屏显示识别到的QR码

扩展知识

MaixPy 机械视觉 API

april官网

1 维护与维修

维护维修政策

产品自签收起7日内未拆封可无理由退换，因产品退换所产生的费用及其他风险需由客户承担。
用户如需产品保修服务需提供相应的购买单据及产品保修卡作为保修凭证。
凡属于正常使用下由于产品本身质量问题引起的硬件故障，**保修期内大象机器人给予免费维修**。
保修起始日期为产品购买日或物流签收日。
维修更换的配件归大象机器人所有，必要时会收取适当的成本费用。

如需以下产品售后服务，请事先联系客服沟通并确认相关信息。以下为详细的配件保修服务说明：
注意：如与产品画册有冲突，以用户手册为准。

舵机

保修期限	保修服务
≤1个月	我司免费提供新的舵机并承担寄送运费（仅一次）
1-3个月	我司免费提供新的舵机，由客户自行承担运费（仅一次）
≥3个月	客户需自己重新购买

电子件（M5 硬件）

保修期限	保修服务
≤3个月	由用户拆卸后寄回，我司免费更换并承担往返运费（仅一次）
3-6个月	由用户拆卸后寄回并承担往返运费，我司免费更换（仅一次）
≥6个月	客户需自己重新购买

结构件，含外壳部分

保修期限	保修服务
≤1年	我司免费提供新的零件由客户自行承担运费（仅一次）
≥1年	客户需自己重新购买

限位调整

关节	运动范围/角度
Joint 1	-165 ~ +165
Joint 2	-165 ~ +165
Joint 3	-165 ~ +165
Joint 4	-165 ~ +165
Joint 5	-165 ~ +165
Joint 6	-175 ~ +175

- 收到机械臂后，应当注意机械臂的每一个关节的限位，每个轴的转动角度不能超过自己的最大物理限位。
- 转动机械臂时应当小角度、轻轻地转动，到达限位后不可再继续用力转动。
- 同时还应注意关节在运行时不要碰到机械臂本身，以免磕坏。

连接线问题（2020年11月- 批次）

问题描述： 机械臂突然失力且无法驱动时，打开J5关节外壳检查是否为连接线脱落，则需剪去较长针脚，贴上绝缘胶带后再放回到舵机后，并盖好盖板。
更换步骤

- 需要拆卸关节后盖外壳
- 进行线缆调整与贴绝缘胶带 视频链接：https://www.bilibili.com/video/BV1hK4y157qP/

外壳更换

视频链接：https://www.bilibili.com/video/BV1np4y1x7Lk/ **更滑步骤**

- 准备如图所示物品，将对应外壳的螺丝拧松，适当用力，左右轻晃取下外壳。
- 将外壳拆下后，检查内部无损坏，即可安装新外壳。

- 将新外壳扣紧，外壳与机械臂扣合时有明显触感，扣好后，将螺丝放置好并拧紧即可完成外壳拆换。

calibration关节矫正

视频链接：https://www.bilibili.com/video/BV1FT4y1P7BV/

矫正步骤

- 在myStudio烧录calibration
- 机械臂的零位即每一个关节的零位校准凹槽对齐时
- 手动将零位校准凹槽对齐后按A键（最左键，设置舵机零位），依次校准每一个关节
- 校准完毕后，basic屏幕会显示：已设置好所有舵机零位
- 按B键（中间键）测试舵机，测试完毕即矫正结束
- 如果没有矫正好，按C键（最右键）重新设置舵机零位，重复以上步骤。

2 常见问题FAQ

 faq

硬件问题

Q: 为什么我使用时垂直状态会出现微小的抖动，而运转的时候没有？

- A: 请检查是否为竖直状态，竖直状态下不受重力影响，机械空隙导致微小抖动，脱离此状态后使用时不会出现此情况。此状态下建议速度为400-500。

Q: 后期ROS系统会收费吗？

- A: ROS是开源的，会更新到我们的github里。固件升级不收费。

Q: myCobot的关节硬限位

- A: 一轴和五轴有限位，一轴顺时针约160°左右，逆时针160°左右
  五轴可时针、逆时针可转动约160°
  注意：转动机械臂时应小角度、轻轻地转动，到达限位后就不可用力继续转动。

Q: 机械臂的速度单位是什么

- A: 运行速度180度/秒

Q: 只是不支持M5 stack的舵机吗，其他M5的传感器都会支持吗

- A: basic目前支持所有传感器

Q: Atom未来是否会支持传感器

- A: atom的支持需要进一步开放接口，在我们的计划内，需要等待

软件问题

Q：为什么我的编译器找不到对应的设备？

- A：需要先搭建开发环境并安装对应的项目库才可以开发设备。

Q：为什么我的编译器无法正常编译示例程序？

- A：所需要的项目库未安装或者项目库存在冲突，建议检查项目库是否安装正确，如安装正确仍无法编译，请重新安装arduino开发环境。

Q：为什么我对ATOM终端烧录固件后设备无法正常运转？

- A：ATOM终端的固件需要使用我们出厂固件，使用中不能更改其他非官方固件，设备如意外烧录其他固件，可以使用“myCobot固件烧录器”选择ATOM终端-选择串口-选择ATOMMAIN固件对ATOM终端进行烧录。

- **UI Flo**
  - Q: UI Flow与最新的固件不支持？
    A: 需要M5更新，已经问M5帮我们更新，时间周期暂不明
- **robo Flow**

- Q: 请问robotStudio软件编程能使用吗
  A: 我们自己的工业编程软件roboFlow可以使用，robotStudio是ABB公司的，无法和我们互通

- **ros**
  - Q: ros版本
    A: */rosdistro: kinetic* /rosversion: 1.12.17
  - Q: ros系统是用电脑连接机械臂进行操作吗
    A: 是的，直接用电脑连接就可以
  - Q: github下载的ros文件夹运行的时候只显示控件不显示myCobot三维模型
    A: 需要手动打开rviz， rosrun rviz rviz
  - Q: 运行时出现range out 或者error[101]
    A: 检查串口是否正确，basic和atom的固件是否正确
  - Q: 运行方式
    A: 目前运行需要开启3个terminal
- **myStudio**
  - myStudio是代替UI Flow了吗
    A: 不是，是代替现在的下载器
- **myCobot phone controller**
  - Q: APP有几个问题请教一下1）蓝牙右边两个控件，一个是回零，另外一个是？2）第六轴没有机械零点，怎么让机器人回零状态，Rz欧拉角是0 3）关节和末端轴的控制按钮，按下之后松开会一直运动吗？
    A: 1.另一个free model。 2.calibration校正时，会默认设置当前位置为零位。 3.松开后应当停下，app使用的jog控制。但由于蓝牙连接存在不稳定性，目前可能存在发送停止指令但arm没有收到。
- 固件烧录器
  - Q: 如何下载，
    A: 下载报错无法访问 需要下载整个文件夹，目前更新到1.3版本
  - Q: 电机不动
    A: 使用最新的1.3版本固件烧录器，烧录atommain到atom中电脑连接时要链接机械臂末端typeC接口才可以给atom更新固件
  - Q: 烧录后，怎么恢复到出厂
    A: 下载固件烧录器或者arduino程序，烧录main固件
- 其他
  - Q: MainControl和 Transponder的区别
    A: MainControl是出厂自带 Transponder是转接程序，烧录后可以直接发数据包协议控制
  - Q: 默认软件-拖动示教，动作只能保存到ram无法保存到flash
    A: 旧版本的bug，调整后将会和中文版的mainControl一起发布
  - Q: 每个Data都是一个16进制数对吧 然后转成hex字符给机器发送
    A: 对的，串口通讯收发数据都是以HEX形式接收和发送

其他问题

Q：能边执行指令边获取实时数据吗

- A：暂时是不可以的，总线工作时不能被打扰

Q：末端接口接舵机型号有没有限制 末端的控制卡，跟M5 stack是什么关系，独立的吗？

- A：末端的设备需要可以适配我们的控制器，可以支持PWM模式控制的舵机

Q：WrigeAngle里的速度参数0-100表示什么

- A：速度参数0-100是速度的百分比

Q：可以用GetAngles获取的6个轴的角度信息？

- A：GetAngles的返回值是一个长度为6的float数组，索引从0开始，依次为6个关节

Q：是伺服还是步进电机？

- A：myCobot搭载的是6个伺服电机

Q：重复定位精度

- A：+/-0.5mm 1mm

Q：减速机是什么结构的

- A：减速机是齿轮组

Q：是什么齿轮呢？

- A：金属齿结构

Q： 这个是用什么编程的？

- A：myCobot是开源的ROS，现在支持市面上绝大多数平台包括UIFLOW，Arduino，microPython，FreeROTS

Q： 实时性如何

- A：Uart通信，最大50ms延时，正常5ms

Q：Arduino库和API接口的详细文档、示例代码可以在哪里看到？

- A：查看我们github即可https://github.com/elephantrobotics/myCobot

Q：连续工作可以多少时间/电机寿命

- A：300-500小时，中间休息15-30分钟再进行使用才会延长机械臂的寿命；

Q：外壳材质

- A：塑料 光敏树脂SLA

Q：有没有力矩传感器

- A：大多数的协作机器人都没有力矩传感器

Q：电机的品牌

- A：自制电机，外发加工

Q：转动关节带编码器吗？

- A：是的

Q：不是谐波？

- A：不是的

Q：伺服舵机我不了解，是航模那样的舵机吗？

- A：是像航模那样的舵机，但是我们定制和修改和很多，是比较适合机械臂的。6轴联动，三次方插补，没有工业机器人那么均匀，终归价格再这里。

Q：电机参数 大舵机：额定负载10kgcm，最大转速60rpm，电压7.4V，编码器12位磁编。

- A：小舵机：额定负载2kgcm，最大转速80rpm，电压7.4V，编码器12位磁编。

Q：ROS版本

-
    - /rosdistro: kinetic
-
    - /rosversion: 1.12.17

Q：角度精度

- 360/4096=0.0879°

Q：电源范围

- 6~9v 3~5A

Q：IO口输入、输出，电压是多少，怎么调试；为什么标了5V

- A：3.3V，atom的IO控制正在逐步开放中；有5V的电源；调试需要看在哪一个平台开发，arduino,uiFlow,python是不一样的

Q：IO口连接问题

- A：先问清客户具体需求；IO 按位置，分为机械臂底座的basic上的IO，和机械臂末端的IO；IO按种类分为 数字信号输入输出DIO，模拟信号AIO，通信 I2C, SPI， 电源接地。

Q：编码器是绝对值还是增量式？

- A：绝对值（舵机自带高精度编码器）

Q：asic软件刷了uiflow，后来用你们的烧写软件刷回了maincontrol，为啥机械臂轨迹录制软件没法用了啊

- A：atom的固件版本不兼容，只能用同一个版本的固件；更新atom的版本即可

Q：为什么我烧录时提示TimeOut？

- A：重新上电或者在烧录时按下复位键即可

Q：为什么我烧录蓝牙或Transponder后atom灯灭了

- A：需要指令控制才会重新打开LED，默认关闭

Q：为什么我的设备烧录示教程序后无法使用

- A：需要检查您的版本号，对应版本号的固件才可以，不同版本之间的程序是不兼容的

Q：机械臂的精度范围偏差有些大

- A：目前的精度如此，需要更高精度可以查看我们的工业级机械臂

Q：烧录了之后仍然无法拖动示教

- A：首先检查一下是否basic与atom都烧录了；烧录的固件是否对应所要实现的需求；烧录的是否是最新的固件 底部Basic烧录maincontrol，顶部Atom烧录atommain

Q：烧录transponder无法控制机械臂

- A：顶部Atom也需要烧录，烧录固件atom2.1alpha；底部Basic烧录transponder

Q：六个舵机是由哪个控制

- A：顶部的Atom控制

Q：python API的示例教程

- A：目前有demo，在github的Test文件夹下有测试代码

Q：拖动示教完成后，需要断电重新开始吗

- A：因为重新录制时关节电机时锁住位置的 可以不用断电，点击release可以让机械臂的关节松动，可以进行下一个拖动示教；断电重新开始也可以

Q：示教是通过读取设置电位值实现的吗（拖动示教原理）

- A：原理是这样的

Q：通过ROS发给关节电机的是什么信息？能不能反馈编码器信息

- A：电位值；读取电位值就可以

Q：用Get coords显示了六个数值

- A：x,y,z,rx,ry,rz，平移加旋转； z-y-x顺规

Q：AC闪红灯，机械臂屏幕不亮

- A：如果电机没动，可能是Basic坏了，Ac频繁闪红灯的意思是在供电，然后又断了，又供电。可能是哪个pcba坏了

3 常见资源

网站

大象机器人官网
https://www.elephantrobotics.com/en/

myCobot github-软件网址
https://github.com/elephantrobotics/myCobot

M5 UI Flow
https://docs.m5stack.com/#/zh_CN/quick_start/m5core/m5stack_core_get_started_MicroPython

视频

bilibili

- 维护维修视频
  - 连接线 https://www.bilibili.com/video/BV1hK4y157qP/
  - 外壳拆卸 https://www.bilibili.com/video/BV1np4y1x7Lk/
  - 限位 https://www.bilibili.com/video/BV11T4y1N7zz/
  - 关节矫正/calibration https://www.bilibili.com/video/BV1FT4y1P7BV
- 教程视频
  - 开箱（包括配件说明；排查故障；限位与虚位；接口介绍；底座安装与连接电源） https://www.bilibili.com/video/BV1To4y1f71P/
  - 拖动示教 https://www.bilibili.com/video/BV16t4y167vw/
  - Maincontrol https://www.bilibili.com/video/BV1FK4y1W75H/
  - ROS https://www.bilibili.com/video/BV1Uh41127dZ/
  - Arduino https://www.bilibili.com/video/BV1Vi4y1c7DQ/
  - myStudio https://www.bilibili.com/video/BV1Qr4y1N7B5/
  - 吸泵安装 https://www.bilibili.com/video/BV16r4y1N7ZB/
- 其他（案例，包括公司和用户）
  - 两个显示屏-红绿灯 https://www.bilibili.com/video/BV1kT4y1M7PG/
  - 宣传视频 https://www.bilibili.com/video/BV1wy4y1U7C9/
  - 用户案例合集（一） https://www.bilibili.com/video/BV1Q5411E7KG/
  - 用户案例合集（二） https://www.bilibili.com/video/BV13U4y1p7Co/

youtube

- maintenance
  - connecting line https://youtu.be/1wq0kTJVqw4
  - disassembly https://youtu.be/wHzFsExkYrE
  - limit https://youtu.be/PUeU-mynljw
  - calibration https://youtu.be/vGznxW4OF10
- Tutorials unboxing https://youtu.be/Lwi8UoihzNc
  - free move https://youtu.be/WzrbOrdQop0
  - Maincontrol https://youtu.be/VKd8b989M8g
  - ROS https://youtu.be/-Jo_IJ8RaXc

    - Arduino https://youtu.be/pkQIApDRJpo
    - myStudio https://youtu.be/Kr9i62ZPf4w
- others
    - 2 display screens-traffic signals https://youtu.be/9ej0tEwhXuE
    - promotional video https://youtu.be/uSw5rsymjVY
    - User cases(1) https://youtu.be/0AI1MN50RS0
    - User cases(2) https://youtu.be/eoR2-MId_-I