Table of Contents

Introduction

1 myCobot - From 0 to 1 1.1
2 Product 1.2
3 How to Read 1.3
4 Use Cases 1.4
5 Quick Start 1.5
6 myCobot Pi 1.6

pymycobot 1.6.1
1. arm_swing 1.6.1.1
2. arm_route_plan 1.6.1.2
3. arm_safety_control 1.6.1.3
4. testing_arm 1.6.1.4
5. arm_dance 1.6.1.5
6. use_arm 1.6.1.6
7. calibration_manipulator_arm 1.6.1.7
8. control_suction_pump 1.6.1.8

myblockly 1.6.2
Myblockly Introduction 1.6.2.1
1. relax_fixed_arm 1.6.2.2
2. testing_jaw 1.6.2.3
3. set_move_time 1.6.2.4
4. mechanism_control 1.6.2.5
5. advanced_control_arm 1.6.2.6

Ros 1.6.3

Preparation

1 Background Knowledge 2.1
  1.1 Industrial Robots 2.1.1
  1.2 Software 2.1.2
    1.2.1 github 2.1.2.1
    1.2.2 arduino 2.1.2.2
  1.3 Eletronics 2.1.3
  1.4 Mechanics 2.1.4
Development

Software Platform and API

1 arduino

1.1 api

1.2 Test

2 uiflow

3 python

3.1 api

4 ROS&Moveit

4.1 Install environment

4.2 Install mycobot_ros

4.3 Control and Follow

4.4 Keyboard control

4.5 moveit

5 RoboFlow

6 Communication Protocols

7 Accessories

7.1 End-effectors

7.1.1 Gripper

7.1.2 Suction Pump

7.1.3 Clamps

7.2 Support

7.2.1 G-Shape Support

7.2.2 Suction Support

7.3 Parst

8 Computer Vision

8.1 Environments

8.2 Color Detection

8.3 Shape Detection
4 Others

1 Maintenance 4.1
2 FAQ 4.2
3 Resources 4.3
myCobot: From 0 to 1

1.1 Why do we design myCobot

An entry-level collaborative robot arm that everyone can learn and play

The original design of myCobot is to help friends who are interested in 6-axis series robot to learn it from entry to master, creating unprecedented experience and teaching value.

What you can learn

Robotics is based on rigid body kinematics and dynamics, but also an interdisciplinary subject that combines hardware, software, algorithm and control.

With myCobot, you can learn that

- **Hardware**
  - Embedded Microcontroller Based on ESP32
  - Motor and Steering Gear
  - M5Stack Basic/ Atom
- **Software**
  - Arduino
  - C++
  - Python
• ROS, MoveIt
• Communication Data
• Virtual Machines & Linux (visual system)

• Algorithm
  • Series Manipulator
  • Coordinate and Coordinate Transformation
  • DH Parameters
  • Kinematics
  • Manipulator Algorithm (e.g. dynamics)

• Machine Vision (Vision Set)
  • Color Recognition
  • Image Recognition
  • Hand-Eye Calibration
  • See and Grab

• Extended Applications
  • End-effector: gripper, suction pump, etc.
  • Robot Suit & Industry 4.0 Applications

Parts of Gitbook

View the directory on the left to jump

There are four major parts of Gitbook:

• Introduction & Quick Start
  • Introduction -- introduce what myCobot is and its main features, etc.
  • How to Read -- help you read Gitbook efficiently according to your learning level and knowledge background
  • Use Cases -- you can know exactly what use cases you can accomplish with
  • Quick Start -- learn the unboxing of your myCobot, and its first boot and use

• Preparation before Development
• **Background Knowledge** -- learn about tools, industrial robots, algorithms, software, hardware, etc.

• **Hardware Learning** -- learn about embedded hardware, structural components, electronic components, etc.

• **Purpose of Use** -- identify the purpose you want to use it for, and complete the study related to your task

• **Development and Use**
  • **Development Environment** -- learn to use Arduino, ROS, uiFlow, roboFlow, python and others development environment to develop myCobot
  • **Accessories** -- learn to use myCobot with different accessories, such us bases, grippers, suction pumps and so on
  • **Machine Vision** -- learn to control myCobot under the guidance of machine vision
  • **Robot Modification** -- learn how to modificate myCobot into a 4 or 5 axis manipulator

• **myCobot Suit**
  • **Intelligent Warehouse**: learn how to use myCobot to carry different objects
  • **Artificial Intelligence**: learn how to control myCobot to grasp objects intelligently under the guidance of machine vision
  • **Industry 4.0**: learn how to grasp and place objects intelligently by simulating production line

---

**Information Source**

- **Official website**: www.elephantrobotics.com
- **Tutorial video**: https://www.youtube.com/channel/UC68l2RaRF2Mp8fpzpCTzNBfA
- **Shop website**: https://shop.elephantrobotics.com

---

**Contact Us**

If you have any other questions, you can contact us as follows.

We will answer you as soon as possible (working day 9:30-18:30)

- **Twitter**: myCobot Official@CobotMy
- **Facebook**: https://www.facebook.com/MyCobot-116558893805177
- **Mail**: support@elephantrobotics.com
myCobot Introduction

1 Design Background

Upholding the mission of “Enjoy Robots World”, Elephant Robotics designed and developed myCobot, the world’s smallest and lightest collaborative robot, retaining most functions of industrial robots. With compact and elegant industrial design, excellent and powerful performance, and huge software and hardware development space, myCobot has unlimited possibilities in applications.

The design prototype of myCobot is from All-in-one Robot launched by Elephant Robot in China in 2018. As the first all-in-one collaborative robot in China, it has won “2019 CAIMRS Industrial Robot Innovation Award” and 2019 High-tech Robot Annual "Innovation Technology Award", and has been sold to more than 30 countries at home and abroad, receiving unanimous praise and recognition from the world's top 500 enterprises.

2 Introduction
myCobot is the world's **smallest and lightest six-axis collaborative robot**, jointly produced by **Elephant Robotics** and **MSSTACK**. It is more than a productivity tool full of imaginations, can carry on the secondary development according to the demands of users to achieve personalized customization.

With a weight of 850g, a payload of 250g and an arm length of 350mm, myCobot is compact but powerful, can not only be matched with a variety of end effectors to adapt to different kinds of application scenarios also support the secondary development of multi-platforms software to meet the needs of various scenarios such as **scientific research and education, smart home, and commercial applications.**

3 Framework
4 Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Data</th>
</tr>
</thead>
<tbody>
<tr>
<td>Model</td>
<td>myCobot-280</td>
</tr>
<tr>
<td>Working radius</td>
<td>280mm</td>
</tr>
<tr>
<td>Payload</td>
<td>250g</td>
</tr>
<tr>
<td>Arm Span</td>
<td>350mm</td>
</tr>
<tr>
<td>Repeatability</td>
<td>±0.5mm</td>
</tr>
<tr>
<td>Weight</td>
<td>850g</td>
</tr>
<tr>
<td>Power input</td>
<td>8V, 5A</td>
</tr>
<tr>
<td>Working Condition</td>
<td>-5° ~ 45°</td>
</tr>
<tr>
<td>Communication</td>
<td>USB Type-C</td>
</tr>
</tbody>
</table>

5 Accessories

- myCobot
- End Effectors
  - Parallel Gripper
Adaptive Gripper
Angled Gripper
Suction Pump

• Base
  • G Base
  • Flap Base

• Others
  • Rocker
  • Battery Case

6 Features

• Unique Industrial Design & Extremely Compact
  myCobot is an integrated modular design and only weighs 850g which is very easy to carry. Its overall body structure is compact with less spare parts and can be quickly disassembled and replaced to realize plug and play.

• High configuration & Equipped with 2 Screens
  myCobot contains 6 high-performance servo motors with fast response, small inertia and smooth rotation, and carries 2 screens supporting fastLED library to show the expanded application scene more easily and clearly.

• Lego Connector & Thousands of M5STACK Ecological Application
  The base and end of myCobot are equipped with Lego Connector, which is suitable for the development of various miniature embedded equipment. Its base is controlled by M5STACK Basic, and thousands of application cases can be use directly.

• Bloky Programming & Supporting Industrial ROS
  Using UIFlow visual programming software, programming myCobot is simple and easy for everyone. You can also Arduino, ROS, or other multiple functional modules of open source system, even RoboFlow, software of industrial robots from Elephant Robotics.

• Track Recording & Learn by hand
  Get rid of the traditional point saving mode, myCobot supports drag teaching to record the saved track and can save up to 60mins different tracks making it easy and fun for new players to learn.
**7 Patents**

myCobot is protected by patents

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Collaborative robotic arm</td>
<td>2020030683471.3</td>
</tr>
<tr>
<td>2</td>
<td>Mechanical arm linkage and mechanical arm</td>
<td>CN 208196791 U</td>
</tr>
<tr>
<td>3</td>
<td>Mechanical arm joint connector and mechanical arm</td>
<td>CN 208196840 U</td>
</tr>
<tr>
<td>4</td>
<td>Method and system for robot posture maintaining, dragging and teaching</td>
<td>ZL 2018 1 1634649.3</td>
</tr>
<tr>
<td>5</td>
<td>A robot online collision detection method and system based on momentum model</td>
<td>ZL 2019 1 0030748.9</td>
</tr>
<tr>
<td>6</td>
<td>A Kind of Robot Dynamic Parameter Identification Method Independent of Joint Angular Acceleration</td>
<td>ZL 2019 1 0773865.4</td>
</tr>
</tbody>
</table>
外观设计专利证书

外观设计名称：机械臂（S1）
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智谷大道南山云谷
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登记注册。专利权自授予公告之日起生效。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的法律状况。专利权的许可使用、质押、转让、终止、恢复和专利权人的姓名及名称、国籍、地址变更等事项登记在专利登记簿上。

局长
申广雨
2019年01月01日
Reading Instruction

Reading Objectives

Gitbook is designed to help you to achieve goals

<table>
<thead>
<tr>
<th>Major Goals</th>
</tr>
</thead>
<tbody>
<tr>
<td>- Understand the basic use of mechanics, electronics and software related to myCobot</td>
</tr>
<tr>
<td>- Understand the basic principle, joints, coordinates, terms, control of mechanical arm, and master simple forward and inverse kinematics calculations</td>
</tr>
<tr>
<td>- Understand the basic operation of API, uiFlow Visual Programming</td>
</tr>
<tr>
<td>- Have all skills to complete the intelligent warehouse suit, and can use myCobot for simple waypoint movement, handling, control, etc.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Extended Goals</th>
</tr>
</thead>
<tbody>
<tr>
<td>- Understand the image recognition algorithms related to machine vision</td>
</tr>
<tr>
<td>- Understand the construction of robot visual scene and the coordination methods and strategies between vision and manipulator</td>
</tr>
<tr>
<td>- Have all skills to complete the AI suit</td>
</tr>
</tbody>
</table>

Your Knowledge Level

Gitbook needs to be read according to your actual learning level and knowledge background. We divide them into three levels:
<table>
<thead>
<tr>
<th>Level</th>
<th>Learning Background</th>
<th>Skills and Qualifications Requires</th>
<th>Estimated Learning Time</th>
<th>Suggested Development Platform</th>
</tr>
</thead>
<tbody>
<tr>
<td>Freshman</td>
<td>Major in Information, Electronics, Mechanical, Automation</td>
<td>Have a knowledge of programming languages Basic knowledge of electronics</td>
<td>100h</td>
<td>uiFlow</td>
</tr>
<tr>
<td>Superior</td>
<td>Knowledge of Arduino or other similar hardware products Knowledge of steering gear and programming Knowledge of IO interface, etc.</td>
<td>Know how to debug API interface Have a knowledge of communication</td>
<td>50h</td>
<td>Arduino</td>
</tr>
<tr>
<td>Professional</td>
<td>Experience with at least one kind of industrial robot or consumer robot Ability to develop hardware and software</td>
<td>Understand the Cartesian coordinates Understand joint control Understand the basic use of robots</td>
<td>30h</td>
<td>all</td>
</tr>
</tbody>
</table>

## Learning Schedule
<table>
<thead>
<tr>
<th>NO</th>
<th>Target</th>
<th>Theory</th>
<th>Practice</th>
<th>Hour</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Unboxing</td>
<td>1. track recording &amp; learn by hand</td>
<td>1. accessories of myCobot 2. drive myCobot to track recording</td>
<td>1h</td>
</tr>
<tr>
<td>2</td>
<td>Background Knowledge Learning</td>
<td>1. application background of industrial robot 2. learning of coordinate and space, cartesian 3d coordinate and rotation, xyz 3. joint and coordinate control of industrial robot</td>
<td>1. joint control and reproduction 2. speed control 3. coordinate point position control and circulation</td>
<td>5h</td>
</tr>
<tr>
<td>3</td>
<td>Hardware Learning</td>
<td>1. principle and operation of embedded electronics 2. principles of steering gear and motor 3. actuator learning</td>
<td>1. control and drive of basic/atom 2. drive and motion of steering gear 3. study of robot accessories</td>
<td>5h</td>
</tr>
<tr>
<td>4</td>
<td>Software: Firmware and Updates</td>
<td>1. identify different software platforms and its purposes for use 2. principles of firmware loading and adaptation</td>
<td>1. choose the platform that works for you 2. download and update the firmware</td>
<td>2h</td>
</tr>
<tr>
<td>5</td>
<td>Software: Development Environment</td>
<td>1. building of arduino 2. library file download and update of arduino 3. have a knowledge of serial communication</td>
<td>1. be familiar with arduino 2. load the library 3. program and run the first line of code</td>
<td>2h</td>
</tr>
<tr>
<td>6</td>
<td>Learning and Development of Robot Library</td>
<td>1. basic communication and operation types of robots 2. common operation of robot 3. control of direction mode and coordinate mode</td>
<td>1. communicate with myCobot 2. control myCobot to move 3. operate I0 port, gripper and other signals</td>
<td>5h</td>
</tr>
<tr>
<td>NO</td>
<td>Target</td>
<td>Theory</td>
<td>Practice</td>
<td>Hour</td>
</tr>
<tr>
<td>----</td>
<td>------------</td>
<td>------------------------------------------------------------------------</td>
<td>--------------------------------------------------------------------------------------------</td>
<td>------</td>
</tr>
<tr>
<td>7</td>
<td>uiFlow</td>
<td>1. understand the basic architecture and relationships of visual programming interface: sensors, actuators and processes 2. variables, loops, and judgments 3. control method of manipulator arm</td>
<td>1. displays different fonts in basic 2. make myCobot move in different positions according to the three buttons of basic 3. control myCobot to cycle multiple points</td>
<td>10h</td>
</tr>
<tr>
<td>8</td>
<td>roboFlow</td>
<td>1. learning of common industrial operating system for robots 2. common modules of roboFlow: point position, fast movement, IO control and input 3. advanced modules of roboFlow: looping, judging, and pallet procedures</td>
<td>1. control the movement of myCobot 2. basic control of IO input and output 3. cycle control and judgment</td>
<td>5h</td>
</tr>
<tr>
<td>9</td>
<td>Intelligent Warehouse</td>
<td>1. learning of robot point motion 2. rules for the placement of pallets 3. principle of end-effector control</td>
<td>1. operate the robot to move at different points 2. operate the robot to grab and place 3. control and recognition of gripper</td>
<td>10h</td>
</tr>
<tr>
<td>NO</td>
<td>Target</td>
<td>Theory</td>
<td>Practice</td>
<td>Hour</td>
</tr>
<tr>
<td>----</td>
<td>--------------------------------------------------</td>
<td>------------------------------------------------------------------------</td>
<td>--------------------------------------------------------------------------</td>
<td>------</td>
</tr>
<tr>
<td>10</td>
<td>Algorithm about Image Recognition</td>
<td>1. introduction to the visual sensor of stickV</td>
<td>1. build virtualbox environment</td>
<td>20h</td>
</tr>
<tr>
<td></td>
<td></td>
<td>2. introduction and programming environment of maixpy</td>
<td>2. read different colors</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>3. methods and strategies of common color recognition</td>
<td>3. recognition of different shapes</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>4. methods and strategies of common shape recognition</td>
<td>4. data transfer and reading in uiFlow</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>5. methods and strategies of common area identification</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>6. json data transfer</td>
<td></td>
<td></td>
</tr>
<tr>
<td>11</td>
<td>Vision and Robotics Control</td>
<td>1. correlation of world and camera coordinates</td>
<td>1. operate myCobot to camera coordinate system</td>
<td>10h</td>
</tr>
<tr>
<td></td>
<td></td>
<td>2. standardization of qr code image</td>
<td>2. the motion of myCobot in the camera coordinate system</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>3. movement and correction</td>
<td>3. re-calibration and setting</td>
<td></td>
</tr>
<tr>
<td>12</td>
<td>Artificial Intelligence</td>
<td>1. learn and make the flow chart</td>
<td>1. sensor connection</td>
<td>20h</td>
</tr>
<tr>
<td></td>
<td></td>
<td>2. learn and make electrical connection diagram</td>
<td>2. connection and drive of gripper</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>3. description and operation of shape classification</td>
<td>3. robot actuator &amp; joint debugging of uiFlow</td>
<td></td>
</tr>
</tbody>
</table>

In addition, you can also buy our **Industry 4.0 suit** to learn how to build and use the simulation industrial applications.

---

**Service Support**

If the above schedule cannot meet your needs, you can contact the **custom service** for further communication. We provide software and hardware customization service.

We will answer you as soon as possible (working day 9:30-18:30)

- Twitter: myCobot Official\@CobotMy
• Facebook: https://www.facebook.com/MyCobot-116558893805177
• Mail: support@elephantrobotics.com
4 Use cases

the mycobot can be used for both personal learning applications and educational applications.

Personal Applications

Leisure & entertainment
dance with rhythm, play an instrument, take pictures and video, write painting, play games and so on

Intelligent furniture
remote control of myCobot to grab and carry objects with gripper and suction pump, such as mixing coffee, grabbing bread, etc.

Teaching AIDS
simulating the teaching of industrial robot, K12 teaching through entertainment experience, cost-effective research laboratory assistant

Scientific and technological equipment
carry AGV car to realize automatic navigation and handling, combined with vision to do infrared thermal imager, face tracking, etc.

Commercial Exhibition
cooperate with other products to do commercial scene demonstration

Educational Applications

1 Maker Education /K12 Education

- Learning Platform: myCobot + Arduino/ROS + UIFlow
- Learning Purpose: independent development, creative learning
- Corresponding Suit: basic suit
2 Artificial Intelligence Course

- Learning Platform: myCobot + StickV camera
- Learning Purpose: robot algorithm, visual recognition algorithm
- Corresponding Suit: artificial intelligence suit
- Suit Content: visual system, feeding table

3 Vocational Education/Higher Education

- Learning platform: myCobot + RoboFlow
- Learning Purpose: robot algorithm, robot movement, robot simulation
- Corresponding Suit: artificial intelligence suit
- Suit Content: visual system, conveyor belt, feeding table, material block
4 Quick Start

Step 1: What’s in the Box

After the packaging box is in place, please confirm that the robot packaging is intact and undamaged. If there is any damage, please contact the logistics company and the local supplier in time.

1 myCobot 280 【standard set】

- myCobot-280
- Brochure
- Power Supply
- USB-Type C
- Jumper
- M4*35, stainless steel screw
- Hexagon wrench

2 Operating Environment and Conditions
Please install the robot system in an environment that meets the conditions described in the table in order to exert and maintain the performance of the machine and use it safely.

<table>
<thead>
<tr>
<th>Environment</th>
<th>Target</th>
</tr>
</thead>
<tbody>
<tr>
<td>Temperature</td>
<td>-10°C~45°C</td>
</tr>
<tr>
<td>Relative Humidity</td>
<td>20%~70%</td>
</tr>
<tr>
<td>Indoor/Outdoor</td>
<td>Indoor</td>
</tr>
<tr>
<td>Another Environmental Requirement</td>
<td>-Avoid sunlight.  -Keep away from dust, oily smoke, salt, iron filings, etc.  -Keep away from flammable and corrosive liquids and gases.  -Do not contact with water.  -Does not transmit shock, vibration, etc.  -Keep away from strong electromagnetic interference sources.</td>
</tr>
</tbody>
</table>

**Step 2: Fix the robot**

You can fix the robot by our base, or design your own customized base to fix.

The specific fixation method can refer to the fixation of this机械臂的固定

**Step 3: Power Supply**

myCobot must use an external power source to provide enough electricity.
- **Rated Voltage:** 7-9V
- **Rated Current:** 3-5A
- **The type of plug:** DC 5.5mm×2.1

Note: You can’t simply supply with typeC inserted into Basic.

Please use the official power supply to avoid damage to myCobot.

### Step 4: Drag Teaching Demonstration

**Before Recording:**
- After entering the recording mode, select the recording storage location
  - Button A: Store to Ram
  - Button B: Store to Memory Card
  - Button C: Exit the Recording Mode

**Start Recording:**
- Click record and select the storage location, then manually drag the robotic arm to complete your target action and remember to save it, the action will be recorded and stored.
- Note: The default recording time is 100s. If the recording time is too long that will exceed the memory, you can customize it by modifying the code, or record it on the computer.

**Play:**
- Click Button A to play the recorded action
  - Button A: Start Playing the Recorded Action
  - Button B: Pause
  - Button C: Exit Playback

**Open source code:** The code above is open, named MainControl, you can download and customize the code via github, refer to the link below:

https://github.com/elephantrobotics/myCobot/tree/main/Arduino/MycobotBasic/examples

**Video Tutorials:** https://youtu.be/WzrbOrdQop0
Raspberry myCobot

Begin your myCobot

Firstly, you need to connect power supply, display, keyboard and mouse as shown in the following image, then initiate myCobot by the red switch.

The use of python

Start quickly

We've pre-loaded the python API package in the Raspberry Pi version of myCobot pymycobot, just use it in your code.

You can create a python file at any position, for example: light_led.py, then write the following code in the file:

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

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

After saving, run it. You'll see the light board on top of myCobot lit in red.

Update the library

If our library is updated, it can also be easily synchronized on Raspberry Pi.

python2 update:

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

python3 update:

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

More examples

In addition to the above content, we provide more examples to help users use the python API.

You can download it at the following address:

The use of ROS

We've pre-installed ROS kinetic in the Raspberry version of myCobot and provide the package `myCobotROS`, you can use it easily.

1. Go to the working directory:

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

1. Go to myCobotROS’s redele, follow `3. Visualization in RViz`, the address is as follows:

https://github.com/elephantrobotics/myCobotROS#3-visualization-in-rviz
Python API of myCobot

这是用于与mycobot进行串行通信并对其进行控制的python API。

Installation

Notes:

Make sure that you flash Atom to the top Atom. flash Transponder to the Basic. Firmware Atom and Transponder Download link: https://github.com/elephantrobotics/myCobot/tree/main/Software You can also use myStudio to update. download link of myStudio: 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. **The robot arm swings left and right**

**Introduction of API**

API used to control the robot arm swings left and right is:

1. `MyCobot(port)`

   Function: Initialize a MyCobot object.

   Parameter Description: `port` - The type of data is String, is the port number that controls the robot arm, and the windows system can view it at the port in Device Manager.

2. `get_angles()`

   Function: Get the angle of the six joint points of the robot arm

   Return Value: The type of return value is list, with six elemental data, corresponding to joints 1 to 6.

3. `send_angles(degrees, speed)`

   Function: Set the angle of six joint points at a time.

   Parameter Description:

   - `degrees`: The type of parameter is list. The angle data for the six joint points must be included. The angle value range of the six joint points is -180 to 180.
   - `speed`: The type of data is int, value range 0~100. Represents the speed at which the robot arm runs to the specified position, and the higher the value, the greater the speed.

4. `send_angle(id, degree, speed)`

   Function: Set the angle of a single joint.

   Parameter Description

   - `id`: Represents the joints of the robotic arm, a total of six joints, with a specific representation. For example, joint1 can be: `Angle.J1.value`.
   - `degree`: Represents the angle of the joint, value range -180 to 180.
   - `speed`: Represents the speed of the robot arm

5. `release_all_servos()`

   Function: Release the robot arm, let it swing manually at will.

**Content of code**
```python
from pymycobot.mycobot import MyCobot
from pymycobot.genre import Angle
from pymycobot import PI_PORT, PI_BAUD  # When using the Raspberry Pi version of MyCobot
import time

# Initialize a MyCobot object
mc = MyCobot(PI_PORT, PI_BAUD)

# Get the coordinates of the current location
angle_datas = mc.get_angles()
print(angle_datas)

# Pass coordinate parameters with a number of columns, let the robot arm move to the specified position
mc.send_angles([0, 0, 0, 0, 0, 0], 50)
print(mc.is_paused())

# Set the wait time to ensure that the robot arm has reached the specified position
while not mc.is_paused():
    time.sleep(2.5)

# Move joint 1 to the position of 90
mc.send_angle(Angle.J1.value, 90, 50)

# Set the wait time to ensure that the robot arm has reached the specified position
time.sleep(2)

# Set the number of cycles
num = 5

# Let the robot arm swing left and right
while num > 0:
    # Move joint 2 to the position of 50
    mc.send_angle(Angle.J2.value, 50, 50)

    # Set the wait time to ensure that the robot arm has reached the specified position
    time.sleep(1.5)

    # Move joint 2 to the position of -50
    mc.send_angle(Angle.J2.value, -50, 50)

    # Set the wait time to ensure that the robot arm has reached the specified position
    time.sleep(1.5)

    num -= 1

# Let the robot arm shrink. You can swing the robot arm manually and then use the get_angles function to get the robot arm to where you want it to be.
mc.send_angles([88.68, -138.51, 155.65, -128.05, -9.93, -15.29], 50)

# Set the wait time to ensure that the robot arm has reached to the specified position
time.sleep(2.5)

# Release the robot arm, let it swing manually at will
mc.release_all_servos()
```

Effects as shown
2. The head of the robot arm intelligently plans the route

Knowledge preparation for API

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

is used to control the movement of the head of the robot arm to a specified point in a specified manner. It is primarily used to enable intelligent planning of the head of the robot arm from one position to another. X, Y, Z represents the position of the head of the robot arm in space (The coordinate system is a right-angle coordinate system). rx, ry, rz represents the posture of the head of the robot arm at that point. (The coordinate system is Euler coordinates). The implementation of the algorithm and the representation of Euler coordinates require a certain degree of academic knowledge, we won’t make too much explanation here, we only need to understand the right angle coordinate system can we use this function well.

Introduction of API

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

   Function: Intelligently plan the route to move the head of the robot arm from the original point to the specified point.

   Parameter Description

   `[x,y,z,rx,ry,rz]`: The parameters are combined into a spatial right-angle coordinate system with x, y, z. The bottom of the robot arm is as the origin, x-positive axis in front, y-positive axis on the right and z-axis on the top. `x, y, z` represented as the position of the head of the robot arm. `[rx, ry, rz]` indicate the posture of the head of the robot arm. You can adjust the posture of the head of the robot arm and then use `get_coords()` to get the posture in the position, In this way you don’t need to know the Euler coordinate system can you also understand the function.

   `speed`: Represents the speed of the robot arm. The range of values is 0 to 100, the higher the value, the faster the speed.

   `model`: Values are limited to 0 and 1. 0 indicates that the path to the head of the robot arm is nonlinear, means the route is randomly planned, as long as the head of the robot arm moves to the specified point in a specified manner. 1 indicates the movement of the head of the robot arm is linear, means the intelligent planning route allows the head of the robot arm to move in a straight line to the specified point.

2. `get_coords()`

   Function: Get the spatial coordinates of the head of the robot arm at this time and the current posture.
Return value: The returned type is a list collection of six float elements, the first three coordinates x, y, z representing the coordinates of the head of the robot arm, and the last three coordinates rx, ry, rz representing the posture of the head of the robot arm.

3. `send_coord(id, coord, speed)`

Function: Only one of the coordinates of the x, y, z axis in the head space coordinates of the robot arm is modified separately.

Parameter Description:

- `id`: The type of data is `genre.Coord`, it represents the X, Y, Z axis of the head of the robot arm. For example: `Coord.X.value`.

- `coord`: The type of data is float, Represents modifying the coordinate value.

- `speed`: Represents the speed of the robot arm. Value range 0~100, the higher the value, the faster it is.

Content of code

```python
from pymycobot.mycobot import MyCobot
from pymycobot.genre import Coord
from pymycobot import PI_PORT, PI_BAUD  # When using the Raspberry Pi version of myCobot
import time

# Initialize a myCobot object
mc = MyCobot(PI_PORT, PI_BAUD)

# Gets the angle and posture of the current head
coords = mc.get_coords()
print(coords)

# Intelligently plan your route, let the head linearly arrive at the coordinate [59.9, -65.8, 250.7, -50.99, 83.14, -52.42]
mc.send_coords([59.9, -65.8, 250.7, -50.99, 83.14, -52.42], 80, 1)
# Set the wait time
time.sleep(1.5)

# Intelligently plan your route, let the head linearly arrive at the coordinate [59.9, -65.8, 350.7, -50.99, 83.14, -52.42]
mc.send_coords([59.9, -65.8, 350.7, -50.99, 83.14, -52.42], 80, 1)
# Set the wait time
time.sleep(1.5)

# Change only the x coordinates of the head, set the x coordinate of the head to -40.
mc.send_coord(Coord.X.value, -40, 70)
```

Effects as shown
3. **Safety control of the robot arm**

**Introduction of API**

1. **is_power_on()**
   
   Function: Determine whether the robot arm is powered.
   
   Return value: 1 indicates powered, 0 indicates outage, -1 indicates an error

2. **power_on()**
   
   Function: Power the robot arm

3. **power_off()**
   
   Function: Power is lost to the robot arm and all functions will fail.
   
   Note: The robot arm cannot be relaxed after outage, means `set_free_mode()` is invalid.

4. **pause()**
   
   Function: Pause the movement of the robot arm.

5. **resume()**
   
   Function: Restore the movement of the robot arm.

6. **stop()**
   
   Function: The robot arm stops moving

7. **is_in_position(data, flag)**
   
   Function: Determines whether the robot arm has reached the specified position
   
   Parameter Description:
   
   - **data**: A list collection of six elements, representing an angle collection, or a collection of robot arm head data.
   - **flag**: 1 Represents the head data of the robot arm. 0 Represents the angle collection data of the robot arm.

8. **is_paused()**
   
   Function: Determine if the robot arm is suspended
   
   Return value: 1 indicates a pause, 0 indicates that it is not paused, and -1 indicates an error.

**Content of code**
from pymycobot.mycobot import MyCobot
from pymycobot import PI_PORT, PI_BAUD  # When using the Raspberry Pi version of MyCobot
import time

# Initialize a MyCobot object
mc = MyCobot(PI_PORT, PI_BAUD)
# Determine whether the robot arm is powered or not, and if there is no power supply,
if not mc.is_power_on():
    # Power the robot arm
    mc.power_on()
# The robot arm arrive at position[0,0,0,0,0,0] at 30 speeds
mc.send_angles([0, 0, 0, 0, 0, 0], 30)
# Get the current time
start = time.time()
# Check whether the robot arm has arrived at the position [0,0,0,0,0,0] or not
while not mc.is_in_position([0, 0, 0, 0, 0, 0], 0):
    # Restore the robot arm movement
    mc.resume()
    # Let the robot arm move 0.5s
    time.sleep(0.5)
    # Pause the robot arm movement
    mc.pause()
    # Determine if the movement timed out
    if time.time() - start > 9:
        # Stop movement of the robot arm
        mc.stop()
        break
# Get the current time
start = time.time()
# The robot arm arrive at position[88.68, -138.51, 155.65, -128.05, -9.93, -15.29] at
mc.send_angles([88.68, -138.51, 155.65, -128.05, -9.93, -15.29], 30)
# Check whether the robot arm has arrived at the position [88.68, -138.51, 155.65, -12
while not mc.is_in_position([88.68, -138.51, 155.65, -128.05, -9.93, -15.29], 0):
    # Restore the movement of the robot arm
    mc.resume()
    # Let the robot arm move 0.5s
    time.sleep(0.5)
    # Pause the movement of the robot arm You can use isPaused() to check whether the
    mc.pause()
    # Check if the movement timed out
    if time.time() - start > 9:
        # Stop movement of the robot arm
        mc.stop()
        break
# Power off the robot arm after operation
mc.power_off()
4. Test the robot arm

Introduction of API

1. is_all_servo_enable()

Function: Determine if six joint points are working properly.
Return parameter: 1 indicates working properly, 0 means that it doesn't work, -1 indicates an error alarm.

2. jog_angle(joint_id,direction,speed)

Function: Let a joint point move continuously.
Parameter Description:
- joint_id: value range 1-6, six integers represent joint points 1 to 6.
- direction: 1 indicates increase in angle, 0 indicates decrease in angle.
- speed: Indicates the speed of increase and decrease.

3. jog_stop()

Function: Stop the movement of joint points.

4. release_servo(servo_id)

Function: Relax the specified joint points.
Parameter Description:
- servo_id: value range 1-6, six integers represent joint points 1 to 6.

5. focus_servo(servo_id)

Function: Secure the specified joint points
Parameter Description
- servo_id: Value range 1-6, six integers represent joint points 1 to 6.

Content of code
from pymycobot.mycobot import MyCobot
from pymycobot import PI_PORT, PI_BAUD  # When using the Raspberry Pi version of myCobot
import time

## Initialize a MyCobot object
mc = MyCobot(PI_PORT, PI_BAUD)
# Determine whether the robot arm is powered or not, and if there is no power supply,
if not mc.is_power_on():
    # Power the robot arm
    mc.power_on()

# Check that the six joints are working properly
# You can also use is_servo_enable(serv_id) to change single calibration
if mc.is_all_servo_enable():
    # Power off the robot arm
    mc.power_off()
    # Check whether the robot arm is power off
    if mc.is_all_servo_enable() == -1:
        print("机臂")

## Effects as shown
5、Dance of the robot arm

Introduction of API

1. set_color(R, G, B)

Function: Set the color of the headlights of the robot arm

Parameter Description:
R, G, B: Corresponding to the value of color array[R, G, B]

Content of code

```python
from pymycobot.mycobot import MyCobot
from pymycobot import PI_PORT, PI_BAUD  # When using the Raspberry Pi version of myCobot
import time

if __name__ == '__main__':
    # Initialize a MyCobot object
    mc = MyCobot(PI_PORT, PI_BAUD)
    # Set the start time
    start = time.time()
    # Let the robot arm move to the specified position
    mc.send_angles([-1.49, 115, -153.45, 30, -33.42, 137.9], 80)
    # Check whether it move to the specified position
    while not mc.is_in_position([-1.49, 115, -153.45, 30, -33.42, 137.9], 0):
        # Restore the movement of the robot arm
        mc.resume()
        time.sleep(0.5)
        # Pause the movement of the robot arm
        mc.pause()
        # Check if the movement timed out
        if time.time() - start > 3:
            break
    # Set start time
    start = time.time()
    # Let the movement last 30 seconds
    while time.time() - start < 30:
        # Let the robot arm reach this position quickly
        mc.send_angles([-1.49, 115, -153.45, 30, -33.42, 137.9], 80)
        # Set the color of the light to[0,0,50]
        mc.set_color(0, 0, 50)
        time.sleep(0.7)
        # Let the robot arm reach this position quickly
        mc.send_angles([-1.49, 55, -153.45, 80, 33.42, 137.9], 80)
        # Set the color of the light to[0,50,0]
        mc.set_color(0, 50, 0)
        time.sleep(0.7)
        # mc.release_all_servos()
```

Effects as shown
6. Use of gripper

Introduction of API

1. is_gripper_moving()

Function: Determine if the gripper is running.

Return parameter:
1 means running, 0 means no, and -1 indicates an error

2. set_encoder(joint_id, encoder)

Function: Turn the specified joint point to the specified position

Parameter Description

- joint_id: Value range 1-7, Represents 1 to 6 joint points and grippers respectively.
- encoder: Value range 0~4096, 2048 means 0 when the value range is -180~180

3. set_encoders(encoders, sp)

Function: Let the robot arm move to the specified position.

Parameter Description:

- encoders: A list collection of six int elements, with the order of six encoder data representing the position of 1 to 6 joint points.
- sp: Indicates the rotate speed of the robot arm.

4. get_encoder(joint_id)

Function: Gets encoder data for the specified joint point.

Parameter Description

- joint_id: Value range 1-7, reprents 1-6 joint points and grippers respectively.

Return value:

- encoder: Indicate the encoder data information of this joint

5. set_gripper_value(value, speed)

Function: Let the gripper turn to the specified position at the specified speed.

Parameter Description:

- value: Indicates where the claws are to be reached, value range 0~4096.
- speed: Indicates how much speed it turns, value range 0~100.

6. get_gripper_value()
Function: Get the encoder data information of gripper

Return value:

encoder: the data information of gripper

7. set_gripper_state(flag, speed)

Function: Let the gripper reach the specified state at the specified speed.

Parameter Description

flag: 1 means claws are closed and 0 means claws are open.

speed: Indicates how quickly the specified state has been reached, value range 0~100.

Content of code
from pymycobot import PI_PORT, PI_BAUD  # When using the Raspberry Pi version of myCobot
from pymycobot.mycobot import MyCobot
import time

def gripper_test(mc):
    print("Start check IO part of api\n")
    # Check if the gripper is moving
    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()
    # Set joint point 1, let it rotates to the position 2048
    mc.set_encoder(1, 2048)
    time.sleep(1)
    # Set a joint position, let the robot arm rotates to the position at 20 speeds.
    mc.set_encoders([1024, 1024, 1024, 1024, 1024, 1024], 20)
    time.sleep(1)
    # Get location information for joint point 1
    print(mc.get_encoder(1))
    # Set the gripper rotate to the position 2048
    mc.set_encoder(7, 2048)
    time.sleep(1)
    # Set the gripper rotate to the position 1300
    mc.set_encoder(7, 1300)
    time.sleep(1)

    # Set the gripper to the state 2048 at 70 speeds.
    mc.set_gripper_value(2048, 70)
    time.sleep(1)
    # Set the gripper to the state 1500 at 70 speeds.
    mc.set_gripper_value(1500, 70)
    time.sleep(1)

    # Set the state of gripper. Let it open its paws quickly at 70 speeds
    mc.set_gripper_state(0, 70)
    time.sleep(1)
    # Set the state of gripper. Let it close its paws quickly at 70 speeds
    mc.set_gripper_state(1, 70)
    time.sleep(1)

    # Get the value of gripper
    print("")
    print(mc.get_gripper_value())

if __name__ == "__main__":
    # Initialize a MyCobot object
    mc = MyCobot(PI_PORT, PI_BAUD)
    # Let it move to the zero position
    mc.set_encoders([2048, 2048, 2048, 2048, 2048, 2048], 20)
    time.sleep(1)
    gripper_test(mc)

Effects as shown
7、Calibration of the robot arm

Introduction of API

1. set_servo_calibration(servo_no)

Function: When the position is adjusted to zero, there is impalement of the port of the robot arm, and the wrong joint can be calibrated using this method.

Parameter Description

- **servo_no**: Value range 1~6, represents 6 joint points.

2. is_controller_connected()

Function: Determines whether the current robot arm is in a writeable state

Return value: 1 means writeable, 0 means you can't write, -1 indicates an error.

3. set_gripper_ini()

Function: Set 6 joints to the initial position.

Content of code

```python
from pymycobot.mycobot import MyCobot
from pymycobot import PI_PORT, PI_BAUD  # hen using the Raspberry Pi version of myCobot

# Initailize a MyCobot object
mc = MyCobot(PI_PORT, PI_BAUD)

# Detect whether the robot arm can burn into the program
if mc.is_controller_connected() != 1:
    print("Connect the robot arm correctly for program writing")
    exit(0)

# Fine-tune the robot arm to ensure that all snaps in the adjusted position are aligned
# Based on the alignment of the robot arm port, only a case is given here
mc.send_angles([0, 0, 18, 0, 0, 0], 20)

# Calibrate the position at this time, and the angle position after calibration is represented
# The for loop is equivalent to the set_gripper_ini() method
for i in range(1, 7):
    mc.set_servo_calibration(i)
```

---

44
8、Control suction pump

Connection

Firstly, we need to power the suction pump as shown in the picture.

Next we need to connect suction pump and myCobot, using control cables. Connect 2 and 5 in the pin port. As shown in the picture.

Introduction of API

```
set_basic_output(pin_no, pin_signal)
```

Function: By connecting `pin_no` and control signal `pin_signal` to control external devices.

Parameter Description:
Pin_no: The int type parameter, the number of the label at the bottom of the device takes only the numeric portion.

Pin_signal: 1 means stop running, 0 means run.

Content of code

```python
from pynycobot.mycobot import MyCobot
from pynycobot import PI_PORT, PI_BAUD  # When using the Raspberry Pi version of myCobot
import time

# Initialize a MyCobot object
mc = MyCobot(PI_PORT, PI_BAUD)

# Open suction pump
def pump_on():
    # Let 2 work
    mc.set_basic_output(2, 0)
    # Let 5 work
    mc.set_basic_output(5, 0)

# Stop suction pump
def pump_off():
    # Let 2 stop working
    mc.set_basic_output(2, 1)
    # Let 5 stop working
    mc.set_basic_output(5, 1)

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

Use Instruction of Myblockly

Interfacce introduction

As 1-1 indicates, ① represents the Puzzle Toolbar, which contains logical control puzzles, variable settings puzzles, mathematical function puzzles, text type puzzles, and control robot arm method puzzles. ② represents a jigsaw puzzle board, pulls the method module in the puzzle toolbar into the puzzle board, and the method module will appear in the drawing board. ③ represents the code display area, and the method module stitched into the drawing board automatically generates python code in the code display area.

Introduction of save and load

After programming, you can click ① save in the picture1-2 to save, the effect after clicking is shown in the picture1-3.
In particular, it is important to add a suffix .xml when defining a saved file name, otherwise the saved file will not be loaded. Of course, if you accidentally forget to add the suffix.xml. It doesn't matter, just rename the file before loading it and add .xml suffix.

You can click ②Load in the picture1-2 to load the file, the effect after clicking is shown in the picture1-4. The file that is loaded at this time can only be a file with a .xml suffix.
Intrduction of Myblockly

Logic

1. As shown in Figure 1-1, all methods contained in the Logic module are included.

2. Method is explained in detail

   1-2 detailed method (一)

   As 1-2 indicates, ① indicates if (condition) do (program) method. If the conditions are satisfied, execute program 1, otherwise program 2.

Loops

1. 如图2-1所示为Loops模块所包含的所有方法。
2-1 Loops模块

2. Method is explained in detail

2-2 Loops模块方法详细

注：在循环中想使用循环中的变量需要设置一致的变量。

Text

1. 如图3-1所示即为Text模块所包含的所有方法。
3-1 Text Module

2. Method is explained in detail

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

3-2 Text模块方法详细（一）

3-3 Text模块方法详细（二）
Math

1. 如图4-1所示即为Math模块所包含的所有方法。

2. Method is explained in detail

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。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

List

1. 如图5-1所示即为List模块所包含的所有方法。
2. Method is explained in detail

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

Variables

1. 如图6-1所示即为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所示即为创建好的变量。
Functions

![7-1 Functions Module](image1)

As 7-1 indicates, `sleep(time_num)` represents waiting `time_num` seconds before execution.

Time

![8-1 Time Module](image2)

Mycobot
The method of use of this module, please refer to pymycobot.
Release and fixation of the robot arm

Case Introduction

In this case, by using loop call, the method of *focus servo* to fix 6 joint points, and use *time* method to fix it for 10 seconds. Finally use loop call *release* method to release 6 joint points.

二、Content of demo
Gripper detection of the robot arm

Case Introduction

By using `Set_Gripper_State` function of myCobot to let the gripper open and close 10 times, and adjust angle after each group close.

demo
Set the movement time of the robot arm

Case content

The main experimental content of this case is to call the `jog_angle` function to keep the six joints moving continuously through a loop. Stop its motion with the `jog_stop` function. Finally, the robot arm is moved to a safer position and the joint is released and powered off.

Content of program
Control mechanism of the robot arm

Case content

This case mainly calls some of the commonly used control mechanism functions of the robot arm to control the robot arm, such as power off the robot arm, power supply, suspension of motion, restore movement and other control mechanism functions, as well as the control of the headlights of the robot arm.

Demo
Adavanced operation of the robot arm

Case Introduction

Mainly realize that the robot arm intelligent judge the function that the robot arm has already arrived the specified position, based on this function, simply let the robot arm repeat two arrival instructions.

- Firstly use `if do` module to judge if the robot arm is powered, if not, you should power the robot arm. Output the current angle node information.
- Transfer the robot arm to zero. Define `angles` variable. Use the `repeated` method in the creation of `list` type data to assign zero node information to `angles`. Define `limit_time` to determine if the movement times out.
- Pass `angles` into the `is in position` method to determine whether the robot arm has reached the specified position. Use the `repeat` module 0.5s per movement to detect whether the robot arm has reached the specified position and time out the timer, and if more than 7s the robot arm has not yet reached the specified position, determine that it has arrived and execute the next instructions.
- The principle is similar, so no more explanation

demo
When you use the Raspberry Pi version of mycobot, you should already have a Raspberry Pi system equipped with ROS Kinetic.

For detailed usage, please refer to the ROS part of Chapter 3.
1 Background

It is necessary to have an in-depth understanding of myCobot from the aspects of hardware, software and robot algorithm. At the beginning, we can understand the progress of robots by understanding the history of robots.

- Robot
- Software
- Electronics
- Mechanics
- Motor
1.1 Robot

This chapter is excerpted from Introduction to robotics mechanics and control by J.Craig. If you want to read more about it, please buy it online.

1 Background

The history of industrial automation is characterized by the rapid renewal of technological means. The renewal of such automation technology is closely related to the world economy, whether as an inducement or a result of the development of the world economy. Industrial robot in the 1960s is undoubtedly a unique equipment, it will be combined with the computer aided design (CAD) system, computer aided manufacturing (CAM) system application, this is the modern manufacturing automation of the latest development trend. These technologies are leading the transition to a new field of industrial automation.

Manipulator is one of the most important types of industrial robots. Whether the manipulator can be called an industrial robot is controversial. The equipment shown here is generally considered to belong to the category of industrial robots, while CNC (NC) grinders are usually outside this category.

Generally speaking, the research of manipulator mechanism and control theory is not a new science, it is only a synthesis of traditional theory. Mechanical engineering theory provides a methodology for the study of manipulator in static and dynamic environments. The mathematical method is used to describe the spatial motion of the manipulator and its characteristics. Control theory provides various design methods and evaluation algorithms for the realization of desired motion or force. Electrical engineering technology can be used in sensor and industrial robot interface design; Computer technology provides the programming platform needed to perform the desired task.

2 Basic Concepts

Mechanical Arm

Mechanical Arm can also be called industrial robot, cooperative robot, manipulator arm, bionic arm, series robot, etc.

Position & Pose

In robot research, we usually study the position of objects in a three-dimensional space. The objects referred to here include not only the lever, parts and grasping tools of the manipulator, but also other objects in the workspace of the manipulator. Usually these objects can be described by two very important properties: position and pose. Naturally, we will first study how to express and calculate these parameters mathematically.

In order to describe the position and posture of a space object, we usually place the object firmly in a space coordinate system, that is, the reference frame, and then we study the position and posture of the space object in this reference...
coordinate system.

**Direct Kinematics**

Kinematics is the study of the motion of objects without regard to the forces causing such motion. In kinematics, we study higher-order derivatives of position, velocity, acceleration, and position variables with respect to time or other variables. Thus, the research object of manipulator kinematics is all the geometric and temporal characteristics of motion. Almost all manipulators are composed of rigid links, adjacent links connected by joints that allow for relative motion. If it's a revolute joint, its displacement is called the joint Angle. These joints are usually fitted with position sensors to measure the relative position of adjacent bars. If you have a revolute joint, this displacement is called the joint Angle. Some manipulators have sliding (or moving) joints, so the displacement of two adjacent links is a linear motion, which sometimes called the joint offset.

A typical problem in the study of manipulator kinematics is manipulator forward kinematics. It is a static geometry problem to calculate the position and posture of its end-effector of the manipulator. Specifically, given a set of values for joint angles, the forward kinematics problem is to compute the position and posture of the tool coordinate system relative to the base coordinate system. In general, we refer to this process as the representation of manipulator position from joint space description to Cartesian space description.

**Degree of Freedom (DOF)**

The number of DOF is the number of manipulator position variables in the coordinate system (reference frame) with figure 1-5 in the manipulator, which determines the position of all components in the mechanism. DOF is universal to all mechanisms. For example, a four-bar mechanism has only one DOF (although it has three movable rods). For a typical industrial robot, the number of joints is equal to the number of DOF because manipulator arms are mostly open motion chains and each joint position is defined by an independent variable.

**End-effector**

End-effector is installed at the free end of the manipulator. Depending on different applications of robot, it may be a fixture, a welding torch, an electromagnet, or some other devices. We usually describe the position of the manipulator in terms of a tool coordinate system attached to its end-effector, and the corresponding tool coordinate system is the base coordinate system connected to the fixed base of the manipulator.

**Inverse Kinematic**

Given the position and posture of the end-effector of the manipulator, calculating all joint angles that can reach the given position and attitude.

### 3 Space description

**Position**
Once the coordinate system is established, we can locate any point in the world coordinate system with a position vector of 3x1. Since many coordinates are often defined in the world coordinate system, a piece of information must be attached to the position vector indicating which coordinate system is defined. In this book, the position vector has a leading superscript to indicate the coordinate system to which it refers.
We find that it is often necessary not only to represent points in space, but also to
describe the posture of objects in space. For example, if the vector "P" in Figure
2-2 directly determines a point between the fingers of the manipulator hand, the
position of the hand can only be fully determined if the posture of the hand is
known. Assuming that the manipulator has a sufficient number of joints, the
manipulator can be in any position and the position of the points between the
fingers remains constant. To describe the posture of an object, we will fix a
coordinate system on the object and give the representation of this coordinate
system with respect to the reference system. In Figure 2-2, the coordinate system
(B) is known to be fixed to the object in some way. The description in (B) relative
to (A) is sufficient to indicate the attitude of object (A).

Coordinate System

A reference frame can be described in terms of the relation of one coordinate
system with respect to another. The reference frame includes the concepts of
position and posture, which is considered to be a combination of these two
concepts in most cases. The position can be represented by a frame of reference
in which the rotation matrix is the identity matrix and the position vector in this
frame of reference determines the position of the described point. Similarly, if the
position vector in the frame of reference is the zero vector, then it represents the
posture.

4 DH Parameters

Definition

For rotational joint n, set 0=0.0, the direction of X axis is the same as that of X,
axis, the origin position of coordinate system (n) is selected to satisfy d.=0.0. For
prismatic joint n, the direction of axis is set to meet 0.=0.0. When d.=0.0, the
origin of the coordinate system (n) is selected to be located at the intersection of
axis XN-1 and joint axis n.

In the link coordinate system, if the link coordinate system is fixedly attached to
the link as described above, the link parameters can be defined as follows:

- a_i-1 : along x_i-1 : move from the distance of z_i-1 to z_i
- alpha_i-1 : aroundx_i-1 : rotate from the Angle of z_i-1 to z_i
- d_i : along z_i : move from the distance of x_i-1 to x_i
- theta_i : aroundz_i : rotate from the Angle of x_i-1 to x_i

myCobot DH parameter
<table>
<thead>
<tr>
<th>Joint</th>
<th>alpha</th>
<th>a</th>
<th>d</th>
<th>theta</th>
<th>offset</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>131.56</td>
<td>theta_1</td>
<td>0</td>
</tr>
<tr>
<td>2</td>
<td>PI/2</td>
<td>0</td>
<td>0</td>
<td>theta_2</td>
<td>-PI/2</td>
</tr>
<tr>
<td>3</td>
<td>0</td>
<td>-110.4</td>
<td>0</td>
<td>theta_3</td>
<td>0</td>
</tr>
<tr>
<td>4</td>
<td>0</td>
<td>-96</td>
<td>64.62</td>
<td>theta_4</td>
<td>-PI/2</td>
</tr>
<tr>
<td>5</td>
<td>PI/2</td>
<td>0</td>
<td>73.18</td>
<td>theta_5</td>
<td>PI/2</td>
</tr>
<tr>
<td>6</td>
<td>-PI/2</td>
<td>0</td>
<td>48.6</td>
<td>theta_6</td>
<td>0</td>
</tr>
</tbody>
</table>
1.2 Software

For myCobot users, software operation requires basic C/C++ knowledge to drive microcontrollers Basic and Atom.

Meanwhile, the related software is mainly required by GitHub and Arduino

- **GitHub**: download the latest myCobot code and update the related users instructions
- **Arduino**: IDE (integration development environment) of the core development of myCobot need to program in C/C++

You can also directly use Python, ROS, UIFlow, and other development tools, which are described in the chapter of Development and Use.

If you want to learn programming languages, you can learn from books, open classes, online videos, etc. And your development platforms can be Windows, MacOS, or Linux.
Github

GitHub is a hosting platform for open source and private software projects, it houses our software (Python, ROS, Arduino), APP, industrial visual programming software - RoboFlow, firmware, user manual and development guide manual.

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

Please click to download as shown in the figure
Arduino

Arduino is an open source electronic prototyping platform that is convenient, flexible and easy to use, contains hardware (various types of Arduino boards) and software (Arduino IDE). The hardware part (or development board) consists of a microcontroller (MCU), flash memory (FLASH), and a set of general input/output interfaces (GPIO), etc., you can think of it as a microcomputer motherboard. The software part is mainly composed of Arduino IDE on PC, related Board Support Package (BSP) and abundant third-party function library. Users can easily download the BSP related to the development board you own and the library of functions by Arduino IDE to write your programs.

How to install Arduino IDE?

- Driver Installation

M5Core host (including BASIC/GRAY/M5GO/FIRE/FACES). Before burning program, please click the button below to download the corresponding CP210X driver zip package according to the operating system you are using. After unpacking the package, select the installation package corresponding to the operating system number to install.

Download CP2104 driver

- Windows10
- MacOS
- Linux
Arduino IDE

If you need to download Arduino IDE, you can click Arduino官网 to download and install the version corresponding to your computer system.

Configure Arduino development environment required by myCobot

1. Add the required development board

- Open Arduino IDE, select File -> Preferences ->setting Then copy the URL below to link to the attached development board manager

https://dl.espressif.com/download/package_esp32_index.json
Select Tools -> Development Board: -> Development Board Manager

In the new dialog box that pops up, enter and search ESP32, then click Install

(If the search fails, try restarting Arduino as below)

After the installation, select Tools -> Development Board: to check whether it was successful As shown in the figure below:
2. Add Libraries

**Different hardware devices have different case libraries, please choose to download it according to the device you are using.**

- You can use project libraries added by IDE.

Open Arduino IDE, select Project -> Load Library -> Management Library...

Search and install the following project libraries separately: M5Stack、M5Core2、M5Atom、ESP32_Lite_Pack_Library, etc.. Steps are as follows:
- Project libraries that need to be downloaded from GitHub, such as MycobotBasic

Download link: https://github.com/elephantrobotics/myCobot

After downloading, unzip and add it according to the index below
Click open and "The library has been added. Please check the 'Import Library' menu." is displayed in the lower right, then the environment configuration of Arduino is complete.
Introduction

Single-Chip Microcomputer (Microcontrollers/SCM) is a kind of integrated circuit chip, which uses VLSI technology to integrate the central processing unit CPU with data processing capabilities, random access memory RAM, read-only memory ROM, various I/O ports, interrupt systems, and timer/Counter and other functions (may also include display drive circuit, pulse width modulation circuit, analog multiplexer, A/D converter and other circuits) integrated into a silicon chip to form a small and complete microcomputer system, widely used in the field of industrial control. From the 1980s, it developed from the 4-bit, 8-bit microcontroller to the current 300M high-speed microcontroller.

Basic Structure

- Arithmetic Unit

Arithmetic Unit is composed of arithmetic logic unit (ALU), accumulator and register. The function of ALU is to perform arithmetic or logical operations on the incoming data. The input comes from two 8-bit data sources, one from the accumulator and the other from the data register. ALU can perform various operations on the data, such as Addition, Subtraction, AND, OR, Comparison, etc., and finally store the results in the accumulator.

The arithmetic machine has two functions:
• Perform various arithmetic operations.
• Perform various logical operations and perform logical tests, such as a zero-value test or a comparison of two values.
• All operations performed by arithmetic unit are directed by the control signal issued by the controller, and an arithmetic operation produces an operation, and a logical operation produces a decision.

**Controller**

Controller is composed of program counter, instruction register, instruction decoder, timing generator and operation controller, etc. It is the "decision-making body" of issuing commands, namely coordinating and directing the operation of the entire microcomputer system. Its main functions are:

• Extract an instruction from memory and indicates the location of the next instruction in memory.
• Decode and test the instructions, then generate the corresponding operation control signal to facilitate the execution of the specified actions.
• Direct and control the direction of data flow between CPU, memory, and input/output devices.

Microprocessor interconnects ALU, counter, register and control parts through an internal bus, and connects the external memory, input and output interface circuit through an external bus. External bus, also known as system bus, is divided into data bus DB, address bus AB and control bus CB. It can be connected with various peripheral devices through the input and output interface circuit.
1.4 Mechanics Background

Is being developed
Motor & Steering Gear

Motor

According to the type of working power supply, it can be divided into:

- 1). DC (Direct Current) motor
  - 1.1). Brushless DC motor
    - 1.2). Brushed DC motor
  - 1.2.1). Permanent magnet DC motor
  - 1.2.1.1). Rare Earth permanent magnet DC motor
  - 1.2.1.2). DC Ferrite permanent magnet DC motor
  - 1.2.1.3). Aluminium Nickel-cobalt permanent magnet DC motor
  - 1.2.2). Electromagnetic DC motor
  - 1.2.2.1). DC series motor
  - 1.2.2.2). Shunt DC motor
  - 1.2.2.3). Separately Excited DC motor
- 2). AC (Alternating Current) motor
  - 2.1). Single-phase motor
  - 2.2). Three-phase motor
According to the application, it can be divided into:

- **Drive motor:** Electric tools (including drilling, buffing, polishing, grooving, cutting, reaming and other tools) motor, household appliances (including washing machine, electric fan, refrigerator, air conditioner, tape recorder, video recorder, DCD, vacuum cleaner, camera, hair dryer, electric shaver, etc.) motor, and other general small mechanical equipment (including all kinds of small machine tools, small machinery, medical equipment, electronic instruments, etc.) motor
- **Control motor:**
  - Stepping motor
  - Servo motor

**Servo Motor**
The steering gear is actually a servo motor, just like the rudder shaft control in the model aircraft and other equipment, so these lightweight servo motor is called steering gear.

**Servo Motor**

Servo Motor refers to the engine that controls the operation of mechanical components in the servo system, it is a kind of indirect speed change device for auxiliary motor.
Hardware of myCobot

The hardware of myCobot is composed of **electronic parts** and **mechanical parts**. Electronic parts include PCBA, controller, steering gear, charger, etc. Structural parts include solid plastic casing, LEGO-mounted ports, flanges, fastener bearings, etc.

Basic & Atom

The microcontroller of myCobot is mainly composed of **BASIC installed at the base** and **Atom installed at the end**. The reason for the design is to separate the motion program from the application program, so that users can program and control myCobot by himself while making it has some real-time performance.

**Basic - M5Stack Basic**

- It is mainly used for the application side of myCobot, which can carry out Arduino programming, UIFlow programming, communication or various software defined by users themselves. Basic related programs are open source for everyone. Check out our GitHub for more information.

**ATOM - M5Stack Atom**

- It is mainly used for the kinematics algorithm control of myCobot, including forward and inverse kinematics, solution selection, acceleration and deceleration, speed synchronization, multiple square interpolation, coordinate transformation, real-time control and multi-threading, etc.. Atom related programs are not open source yet.

**Note**: Basic and Atom can communicate with each other through a communication protocol that is open to all users simultaneously.
**myCobot Electronic Components**

myCobot Electronics

- **Base** (gray shell)
  - Charging Board PCBA: charging and voltage protection function
  - PCBA Substrate: control signal conversion function
  - M5 Basic: main controller
  - Servo Motor No. 1

- **Body** (white shell)
  - Servo Motor No. 2-6
  - Pinboard of Servo Motor

- **End**
  - M5 Atom: 2nd controller
  - Peripheral - Charger

**Structural Parts**

- **Base**
  - Through-Hole
  - Lego Interface

- **Body**
  - White plastic shell
  - Fixed parts, fasteners, bearings, etc

- **End**
DH Parameter & Coordinate System

DH parameter of myCobot is the modified DH parameter, and the specific parameter values are as follows.

<table>
<thead>
<tr>
<th>Joint</th>
<th>alpha</th>
<th>a</th>
<th>d</th>
<th>theta</th>
<th>offset</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>131.56</td>
<td>theta_1</td>
<td>0</td>
</tr>
<tr>
<td>2</td>
<td>PI/2</td>
<td>0</td>
<td>0</td>
<td>theta_2</td>
<td>-PI/2</td>
</tr>
<tr>
<td>3</td>
<td>0</td>
<td>-110.4</td>
<td>0</td>
<td>theta_3</td>
<td>0</td>
</tr>
<tr>
<td>4</td>
<td>0</td>
<td>-96</td>
<td>66.39</td>
<td>theta_4</td>
<td>-PI/2</td>
</tr>
<tr>
<td>5</td>
<td>PI/2</td>
<td>0</td>
<td>73.18</td>
<td>theta_5</td>
<td>PI/2</td>
</tr>
<tr>
<td>6</td>
<td>-PI/2</td>
<td>0</td>
<td>43.6</td>
<td>theta_6</td>
<td>0</td>
</tr>
</tbody>
</table>
The coordinate system represented by DH parameter of myCobot is as follows.

Maintenance

If there are some problems with your myCobot, you can contact our customer service for maintenance.
BASIC

Description

M5Stack BASIC Kit, like its namesake, is a starter kit among the M5Stack development kit series. It's a modular, stackable, scalable, and portable device which is powered with an ESP-32 core, which makes it open source, low cost, full-function, and easy for developers to handle new product development on all stages including circuit design, PCB design, software, mold design and production. This Basic kit provides a friendly price and full-featured resources which makes it a good starter kit for you to explore IoT.

If you want to explore the fastest way of IoT prototyping, M5Stack development board is the perfect solution. Not like others, M5Stack development board is highly efficient, covered with industrial grade case and ESP32-based development board. It integrates with Wi-Fi & Bluetooth modules and contains a dual-core and 16MB of SPI Flash. Together with 30+ M5Stack stackable modules, 40+ extendable units and different levels of program language, you can create and verify your IoT product in a very short time.

Supportive development platforms and programming languages: Arduino, Blockly language with UIFlow, Micropython. Regardless of what level programming skill you have, M5Stack would guide you in every step of the way to realize your idea as well as to the final productization. If you ever played with ESP8266, you would realize that ESP32 is a perfect upgrade out of ESP8266. In comparison, ESP32 has more GPIOs, more analog inputs and two analog outputs, multiple extra peripherals (like a spare UART). Official developing platform ESP-IDF has transplanted with FreeRTOS. With dual-core and real time OS you can get more organized code and much high speed processor.
Pin description of myCobot

Features

- ESP32-based
- Built-in Speaker, Buttons, Color LCD, Power/Reset button
- TF card slot (16G Maximum size)
- Magnetic suction at back
- Extendable Pins & Holes
- M-Bus Socket & Pins
- Program Platform: UIFlow, MicroPython, Arduino

Applications

- Internet of things terminal controller
- Stem education product
- DIY creation
- Smart home equipment

Parameters
<table>
<thead>
<tr>
<th>Resources</th>
<th>Parameter</th>
</tr>
</thead>
<tbody>
<tr>
<td>ESP32</td>
<td>240MHz dual core, 600 DMIPS, 520KB SRAM, Wi-Fi, dual mode Bluetooth</td>
</tr>
<tr>
<td>Flash Memory</td>
<td>16MB</td>
</tr>
<tr>
<td>Power Input</td>
<td>5V@ 500mA</td>
</tr>
<tr>
<td>Port</td>
<td>TypeC x 1, GROVE(I2C+I/O+UART) x 1</td>
</tr>
<tr>
<td>Core Bottom Port</td>
<td>PIN (G1, G2, G3, G16, G17, G18, G19, G21, G22, G23, G25, G26, G35, G36)</td>
</tr>
<tr>
<td>IPS Screen</td>
<td>2 inch, 320x240 Colorful TFT LCD, ILI9342C, max brightness 853nit</td>
</tr>
<tr>
<td>Button</td>
<td>Custom button x 3</td>
</tr>
<tr>
<td>Battery</td>
<td>110mAh@ 3.7V</td>
</tr>
<tr>
<td>Antenna</td>
<td>2.4G 3D Antenna</td>
</tr>
<tr>
<td>Operating</td>
<td>32°F to 104°F (0°C to 40°C)</td>
</tr>
<tr>
<td>Temperature</td>
<td></td>
</tr>
<tr>
<td>Net weight</td>
<td>47.2g</td>
</tr>
<tr>
<td>Gross weight</td>
<td>93g</td>
</tr>
<tr>
<td>Product Size</td>
<td>54 x 54 x 18mm</td>
</tr>
<tr>
<td>Package Size</td>
<td>95 x 65 x 25mm</td>
</tr>
<tr>
<td>Case Material</td>
<td>Plastic (PC)</td>
</tr>
</tbody>
</table>

**LCD & TF card**

**LCD : 320x240 TF card Maximum size 16GB**

<table>
<thead>
<tr>
<th>ESP32 Chip</th>
<th>GPIO23</th>
<th>GPIO19</th>
<th>GPIO18</th>
<th>GPIO14</th>
<th>GPIO27</th>
</tr>
</thead>
<tbody>
<tr>
<td>ILI9342C</td>
<td>MOSI/MISO</td>
<td>/</td>
<td>CLK</td>
<td>CS</td>
<td>DC</td>
</tr>
<tr>
<td>TF</td>
<td>MOSI</td>
<td>MISO</td>
<td>CLK</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Button**

<table>
<thead>
<tr>
<th>ESP32 Chip</th>
<th>GPIO39</th>
<th>GPIO38</th>
<th>GPIO37</th>
<th>GPIO25</th>
</tr>
</thead>
<tbody>
<tr>
<td>Button Pin</td>
<td>BUTTON A</td>
<td>BUTTON B</td>
<td>BUTTON C</td>
<td>/</td>
</tr>
</tbody>
</table>

**GROVE Port A & IP5306**

<table>
<thead>
<tr>
<th>ESP32 Chip</th>
<th>GPIO22</th>
<th>GPIO21</th>
<th>5V</th>
<th>GND</th>
</tr>
</thead>
<tbody>
<tr>
<td>GROVE A</td>
<td>SCL</td>
<td>SDA</td>
<td>5V</td>
<td>GND</td>
</tr>
<tr>
<td>IP5306</td>
<td>SCL</td>
<td>SDA</td>
<td>5V</td>
<td>GND</td>
</tr>
</tbody>
</table>
IP5306 charging/discharging Voltage parameter

<table>
<thead>
<tr>
<th>Charging</th>
<th>Discharging</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.00~3.40V -&gt; 0%</td>
<td>4.20~4.07V -&gt; 100%</td>
</tr>
<tr>
<td>3.40~3.61V -&gt; 25%</td>
<td>4.07~3.81V -&gt; 75%</td>
</tr>
<tr>
<td>3.61~3.88V -&gt; 50%</td>
<td>3.81~3.55V -&gt; 50%</td>
</tr>
<tr>
<td>3.88~4.12V -&gt; 75%</td>
<td>3.55~3.33V -&gt; 25%</td>
</tr>
<tr>
<td>4.12~/ -&gt; 100%</td>
<td>3.33~0.00V -&gt; 0%</td>
</tr>
</tbody>
</table>

ESP32 ADC/DAC

<table>
<thead>
<tr>
<th>ADC1</th>
<th>ADC2</th>
<th>DAC1</th>
<th>DAC2</th>
</tr>
</thead>
<tbody>
<tr>
<td>8 channels</td>
<td>10 channels</td>
<td>2 channels</td>
<td>2 channels</td>
</tr>
</tbody>
</table>

Charging current measured value

<table>
<thead>
<tr>
<th>Charging current</th>
<th>Fully charged current (Power OFF)</th>
<th>Fully charged current (Power ON)</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.55A</td>
<td>-</td>
<td>0.066A</td>
</tr>
</tbody>
</table>

Related Link

Datasheet

- ESP32
- IP5306

API

- Arduino API

pcba

- pcba.pdf
M5 Stack Atom

Description

ATOM Matrix, which has a size of only 24 * 24mm, is the most compact development board in the M5Stack development kit series. It provides more GPIO pins and is very suitable for handy and miniature embedded device development. The main control adopts the ESP32-PICO-D4 chip, which comes integrated with Wi-Fi and Bluetooth technologies and has 4MB of integrated SPI flash memory. The Atom board provides an Infra-Red LED along with the 5 * 5 RGB LED matrix on the panel, a built-in IMU sensor (MPU6886), and a HY2.0 interface. A general purpose programmable button is provided below the RGB Led matrix to enable users to add input support to their various projects. The on-board USB interface (Type-C) enables rapid program uploading and execution. One M2 screw hole is provided on the back for mounting the board.

Note: When using FastLED lib, the recommended brightness of RGB LED is 20. Please do not set it to a high brightness value to avoid damage to the LED and acrylic screen. (In ATOM lib, we have mapped its appropriate brightness range to 0~100)

Product Features
• ESP32 PICO-based
• Programmable button
• 5 * 5 RGB LED matrix panel(WS2812C)
• Built-in Infra-red LED
• Built-in MPU6886 Inertial Sensor
• Extendable Pins & Holes
• Program Platform: Arduino, UIFlow

Applications

• Internet of things terminal controller
• IoT node
• Wearable peripherals

Specification

<table>
<thead>
<tr>
<th>Resources</th>
<th>Parameter</th>
</tr>
</thead>
<tbody>
<tr>
<td>ESP32</td>
<td>240MHz dual core, 600 DMIPS, 520KB SRAM, Wi-Fi, dual mode Bluetooth</td>
</tr>
<tr>
<td>Flash</td>
<td>4MB</td>
</tr>
<tr>
<td>Power Input</td>
<td>5V@ 500mA</td>
</tr>
<tr>
<td>Port</td>
<td>TypeC x 1, GROVE(I2C+I/0+UART) x 1</td>
</tr>
<tr>
<td>PIN Port</td>
<td>G19, G21, G22, G23, G25, G33</td>
</tr>
<tr>
<td>RGB LED</td>
<td>WS2812C 2020 x 25</td>
</tr>
<tr>
<td>MEMS</td>
<td>MPU6886</td>
</tr>
<tr>
<td>IR</td>
<td>Infrared transmission</td>
</tr>
<tr>
<td>Button</td>
<td>Custom button x 1</td>
</tr>
<tr>
<td>Antenna</td>
<td>2.4G 3D Antenna</td>
</tr>
<tr>
<td>Operating Temperature</td>
<td>32°F to 104°F ( 0°C to 40°C )</td>
</tr>
<tr>
<td>Net weight</td>
<td>3g</td>
</tr>
<tr>
<td>Gross weight</td>
<td>14g</td>
</tr>
<tr>
<td>Product Size</td>
<td>24<em>24</em>14mm</td>
</tr>
<tr>
<td>Package Size</td>
<td>43<em>43</em>20mm</td>
</tr>
<tr>
<td>Case Material</td>
<td>Plastic ( PC )</td>
</tr>
</tbody>
</table>
3.4 Motors and Servos

Specification

<table>
<thead>
<tr>
<th>Specification</th>
<th>Parameter</th>
</tr>
</thead>
<tbody>
<tr>
<td>Dimension</td>
<td>45.2X24.7X35mm</td>
</tr>
<tr>
<td>Locked-rotor torque</td>
<td>19.5 kg·cm@ 7.4V</td>
</tr>
<tr>
<td>Locked-rotor speed</td>
<td><a href="mailto:52RPM@7.4V">52RPM@7.4V</a></td>
</tr>
<tr>
<td>Feedback</td>
<td>Load/position/speed/voltage/current/temperature</td>
</tr>
<tr>
<td>Electronic protection</td>
<td>Overheat/overcurrent/overvoltage/overload protection</td>
</tr>
</tbody>
</table>

Structure Feature

- The shell adopts engineering plastic shell with higher strength, which optimizes the center point alignment distance and makes the overall structure more compact.
- Steering gear adopts 1:345 copper tooth combination, which makes greater torque.
- Under the condition of the same torque, it will appear shorter (5mm) than the size of the standard steering gear.
- The body adopts double-axis structure design, round lining three-dimensional structure characteristics with metal main and auxiliary steering wheel and double wire wiring, suitable for quadruped robot, snake robot, desktop robot, humanoid robot, mechanical arm applications.

Electric Control Function

- **Acceleration Start & Stop**
  Speed and acceleration can be set, the movement is better and softer.

- **High Precision**
  360-degree absolute position 4096 bit accuracy, the highest position resolves 0.088 degrees. If you control 90 degrees, input 4096/36090=1024; if you control 180 degrees, input 4096/360180=2048; the rest can be done in the same manner.

- **Working Modes**
  - **Mode 0**: Location mode, which is the default one. 360 degree absolute Angle control can be achieved in this mode and. Support acceleration motion.
  - **Mode 1**: Speed closed loop. In the programming interface, the operation mode is set to 1, you can switch to speed closed loop mode, and enter the corresponding speed under the speed bar to run.
• **Mode 2**: Speed open loop, in the programming interface, the operation mode is set to 2, you can switch to speed open loop mode, and enter the corresponding time under the time bar to run.

**A key to Calibrate**

Install in any position with 360°Angle, {enter 128 (decimal) at address number 40 (decimal}) calibrate the current position to the center with one
Learn Structure and fixation of myCobot

How to fix myCobot

The actual weight of myCobot collaborative robot is 850g. Considering the movement of the robot, the center of gravity will move as the robot moves. Therefore, the robot needs to be fixed on a solid base to be used normally.

Common Fix: myCobot There are two common ways to fix myCobot:

1. Secure the lego connector on the base with lego interface.
   Types of bases: Flap Base and G Base, you can find them in myCobot's peripheral base. myCobot Support
2. Use 4 screws to pass through myCobot base, secure to a threaded base.
   The position of the 4 screw holes on the base can be referenced in the myCobot screw position diagram below.

Interface size of robot base

The pedestal fixing hole is the interface that fixes the robot to other bases or planes. The specific hole size is shown as following. It is 4 through holes with a diameter of 4.5mm, which can be fixed with M4 bolts.
Make sure that there is a corresponding threaded hole on the fixed base before installing. **Before you officially install, please confirm:**

- The environment to be installed complies with the requirements above.
- The installation position is not less than the working range of the robot, and there is enough space for installation, use, maintenance and repair.
- Place the stand in the proper position.
- Installation related tools are ready, such as screws, wrenches, etc.

After confirming the above, move the robot to the mounting surface of the base, adjust the position of the robot, and align the fixing hole of the robot base with the hole on the mounting surface of the base.

**Note:** When adjusting the position of the robot on the mounting base, please avoid pushing the robot directly on the mounting surface of the base to avoid scratches. When manually moving the robot, please try to avoid applying external force to the weak part of the robot body to avoid unnecessary damage to the robot.

**End assembly diagram**
The end of the robot arm is compatible with both lego connector holes and screw threaded holes.

MyCobot Urdf Mode
myStudio

The design of myStudio

- myStudio is a one-stop platform for robots of myRobot/myCobot.
- It is convenient for users to choose different firmware and download it according to their own usage scenarios, and learn relevant textbooks and videos.

Supported platforms and versions

- Version 1.2
- windows, mac, linux

The main functions

- Update the firmware;
- Provide video tutorials on how to use the robot;
- Provide maintenance and repair information (such as video tutorials, Q&A, etc.).

Select your use purpose

Please download different **PC, BASIC** firmware and **Atom** firmware depending on your development environment.
<table>
<thead>
<tr>
<th>开发使用环境</th>
<th>PC所需Library</th>
<th>Basic所需零件</th>
<th>Atom所需零件</th>
</tr>
</thead>
<tbody>
<tr>
<td>默认程序 Default Program</td>
<td>N/a</td>
<td>mainControl</td>
<td>atomMain</td>
</tr>
<tr>
<td>UI Flow可程式化编程 Visual Programming</td>
<td>UIFLOW</td>
<td>UIFLOW读取器由MS提供</td>
<td>atomMain</td>
</tr>
<tr>
<td>RoboFlow 机器人可程式化控制软件 Industrial Programming Software</td>
<td>RoboFlow库</td>
<td>Transponder文件</td>
<td>atomMain</td>
</tr>
<tr>
<td>Arduino 开发板 Maker!</td>
<td>Arduino IDE  + MSStack Lib 套库</td>
<td>各类程序自定义 All Examples</td>
<td>atomMain</td>
</tr>
<tr>
<td>API 开发软件接口 on Desktop</td>
<td>Python/ C+</td>
<td>Transponder文件</td>
<td>atomMain</td>
</tr>
<tr>
<td>ROS Development ROS 开发</td>
<td>ROS库</td>
<td>Transponder文件</td>
<td>atomMain</td>
</tr>
<tr>
<td>通信协议 - USB/TxRx(V1/03)</td>
<td>通信协议阅读 Read Protocol</td>
<td>Transponder文件</td>
<td>atomMain</td>
</tr>
<tr>
<td>蓝牙或无线连接 Bluetooth 蓝牙直接连接Basic</td>
<td>通信协议阅读 Read Protocol</td>
<td>BT_Transponder文件</td>
<td>atomMain</td>
</tr>
<tr>
<td>手机控制器 phoneApp</td>
<td>myCube 手机APP</td>
<td>BT_Transponder文件</td>
<td>atomMain</td>
</tr>
</tbody>
</table>
Download and Loading myStudio

Download myStudio

myStudio https://github.com/elephantrobotics/myStudio

Select the latest version

Then select different versions for different systems
Note: Don’ t install in a folder with a space directory.

**Download Basic & Atom**

**Burn Basic**

- First, connect the BASIC development board with USB, the connection window of myStudio will display the connected development board, select and click “Connect”
Then there are the basic-related firmware in the Basic and tools. Select the firmware you want to burn, and click to burn.
Burn Atom

- Burn Atom is same as Burn Basic, with USB connection at the end of the Atom
- ATOM can be selected in the Board, firmware of Atom will appear
- There is only one firmware of Atom, click to burn

Usage of myStudio

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

Q&A

Q: When you click on the Tools it will be stuck in the side bar for the first time?
A: Please make sure your network status is good.
Software Platform & API

Before developing, make sure that the Basic and Atom in your myCobot are using the latest firmware and suitable for your environment.

You can update the firmware through MyStudio

We currently support the development of the following API

**Arduino**
- Suitable for maker development, you can use all kinds of Arduino program library.

**uiFlow**
- Suitable for beginners to develop, based on Visual programming, can be adapted to the full M5 education kit.

**python**
- Suitable for users with a certain level of Python development. Based on python3.

**ROS&MoveIt**
- Suitable for professional development, can undertake the simulation and calculation of mechanical arm, trajectory planning, etc.

**roboFlow**
- Suitable for industrial development, It is an industrial oriented operating system launched by Elephant Robot, adapted all series of robotic arms of Elephant Robot

**Communication Protocols and Messages**
- Suitable for serial port development, can directly communicate by sending message

In addition, we are also developing C# and other software interface API for development.
<table>
<thead>
<tr>
<th>开发使用环境</th>
<th>PC所需Library</th>
<th>Basic所需固件</th>
<th>Atom所需固件</th>
</tr>
</thead>
<tbody>
<tr>
<td>默认程序 Default Program</td>
<td>N/a</td>
<td>mainControl</td>
<td>atomMain</td>
</tr>
<tr>
<td>UI Flow可程式化编程 Visual Programming</td>
<td>UIFLOW</td>
<td>UIFLOW转换器 由M5提供</td>
<td>atomMain</td>
</tr>
<tr>
<td>RoboFlow 工業級可程式化編程软件 Industrial Programming Software</td>
<td>RoboFlow库</td>
<td>Transponder文件</td>
<td>atomMain</td>
</tr>
<tr>
<td>Arduino 创客 DIY Maker!</td>
<td>Arduino IDE + M5Stack Lib 库</td>
<td>各类程序自定义 All Examples</td>
<td>atomMain</td>
</tr>
<tr>
<td>API 开发软件接口 on Desktop</td>
<td>Python/ C++</td>
<td>Transponder文件</td>
<td>atomMain</td>
</tr>
<tr>
<td>ROS Development ROS 开发</td>
<td>ROS库</td>
<td>Transponder文件</td>
<td>atomMain</td>
</tr>
<tr>
<td>通信协议 - USB/TTL/485/483</td>
<td>通信协议阅读 Read Protocol</td>
<td>Transponder文件</td>
<td>atomMain</td>
</tr>
<tr>
<td>蓝牙或无线连接 BlueTooth</td>
<td>通信协议阅读 Read Protocol</td>
<td>BT_Transponder文件</td>
<td>atomMain</td>
</tr>
<tr>
<td>手机控制器 phoneApp</td>
<td>myCobot 手机</td>
<td>BT_Transponder文件</td>
<td>atomMain</td>
</tr>
</tbody>
</table>
Arduino

1. Preparation before development

1.1 Connected Device

- Connect the Basic on the base of myCobot with PC terminal by Type-C data line

1.2 Requirement

- ATOM: Burn the latest version of ATOMMain (At least 2.7 Version)
- Basic: None

1.3 Check for Connection

- After connection with computer, open device manager to check if there is a device
- If the device is not detected, please replace the USB cable. If it indicates that the device cannot be used, please install and download CP210X. After downloading, unzip it and install the required version of driver.
- Open Arduino IDE -> tools -> port to check if there is a device. If the device is not detected, please replace the USB cable to test, or test whether the driver has been installed successfully

2. Start development

2.1 Burn an official demo

- Open Arduino IDE, select -> file -> Examples -> mycobotbasic, then you can see all Examples about myCobot
- Burn an demo -> SetRGB.ino.
Open SetRGB from Examples

Select the development board: M5Stack-Core-ESP32 and COM

Click to download

Wait until the bottom right shows upload success, which means that the application has been downloaded
Then you’ll see the Atom screen loop with red, green and blue lights
Arduino API

1. Overall Status

`powerOn();`
- Function: atom power on (open by default)
- Return Value: none

`powerOff();`
- Function: atom power off
- Return Value: none

`isPoweredOn();`
- Function: atom status inquiry, return Atom link status
- Return Value: power on is TRUE, power off is FALSE

`setFreeMove();`
- Function: All joints close torsion output
- Return Value: none

2. MDI Mode and Robot Control (Manual Data Input)

`getAngles();`
- Function: Read all joint angles, when used one Angles should be defined to receive data that was read. Angles are defined in terms of variables or functions built into library functions. We can define a memory space that is 6 angles to store Angle variables, it is used in the same way as arrays.
- Return Value: Angles type of array

`writeAngle(int joint, float value, int speed);`
- Function: Send a single joint Angle
- Parameter Specification:
  Joint Number = joint, range from 1 to 6 Specified Angle Value = value, range approximately from -160° ~ ± 160° Specified Speed = value, range from 0~100
- Return Value: none

`writeAngles(Angles angles, int speed);`
- Function: Synchronize joint angles, send joint angles at the same time. Specified Angles is a container with a capacity of 6 data, can be viewed as an array. Use a for loop to assign values, or assign values separately.
- Angles[0] = Specified Angle, Angles[2] = Specify Angle, range from 0 – 90 (the value range should be the same as writeAngle) unit°
  Movement Speed = speed, range from 0 – 100 unit°
- Return Value: none
getCoords();

- Function: Read x,y,z,rx,ry,rz of the end of myCobot, a Coords tempcoords should be defined when used to received angles that was read. Coords are defined in terms of variables or functions built into library functions. We can define a memory space that is 6 tempcoords to store Angle variables, it is used in the same way as arrays.
- Return Value: An array of type Coords. You need to define variables of type Coords.

isInPosition (Coords coord,bool is_linear);

- Function: Read x, y, z, rx, ry, rz at the end of the current robot arm to test whether the specified point has been reached, you should define a Coords tempcoords to receive the read angle when you use it. Coords is a variable number or function definition of a library function that defines a storage space of 6 memory, tempcoords, which is used in the same way as an array.
- Return value: An array under the Coords type that needs to define variables of the Coords type

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

- Function: Send the specific value of the individual coordinate parameters x/y/z, the ends are going to move in a single direction.
- Parameter Specification:
  Value of Moving Path Coordinate = value range from -300 – 300 ( The position coordinates of axis=Axis::X, aixs=Axis::Y and axis=Axis::Z are respectively X,Y,Z, the units would be mm. Position coordinate value range is not uniform, axis=Axis::RX, aixs=Axis::RY and axis=Axis::RZ are respectively RX,RY,RZ ranging from-180°~180°, if the value is beyond the range it will return the clue "inverse kinematics no solution" )
  Specified Speed = speed range from 0~100 unit %
- Return Value: none

writeCoords(Coords coords, int speed);

- Function: Send the specified coordinate parameter, the type of parameter is Coords, need to declares a variable of type COORDS, it is used in the same way as arrays.
- Parameter Specification
  coords[0] = X, coords[1] = Y, coords[2] = Z, X,Y,Z range from -300-300.00 ( Value range is not defined. If the value is beyond the range, the clue "inverse kinematics no solution" will be given ) unit mm
  RX,RY,RZ range from -180~180
  Specified Speed = speed, range from 0~100 unit %
- Return Value: none

checkRunning();

- Function: Check whether the equipment is in motion
- Return Value: In motion is TRUE, on the contrary it's FALSE

setEncoder(int joint, int encoder);
• Function: Set a single joint to rotate to a specified potential value
  • Parameter Specification:
    Joint Number = joint, range from 1-6
    Potential Value of Servo Motor = encoder, range from 0-4096 (The range should be positively related to the range of each joint)
  • Return Value: none

getEncoder(int joint);

• Function: Get the specified joint potential value
  • Parameter Specification: Servo Motor Number = joint range from 1-6
  • Return Value: int type, range from 0-4096

setEncoders(Angles angleEncoders, int speed);

• Function: Set the six joints to run synchronously to the specified position
  • Parameter Specification: Need to define a variable of type Angles: angleEncoders, it is used in the same way as arrays. Assign a value to the array angleEncoders, values range from 0 to 4096 (The range should be positively related to the range of each joint), the length range of the array is 6. Specified Speed = speed, range from 0~100unit %
  • Return Value: none

3. JOG Mode

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

• Function: Control the movement of a single joint in one direction
  • Parameter Specification:
    Joint/Servo Motor
    • Number = joint, range from 1-6
    • Direction of Joint Motion = Direction, range from -1/1
    • Specified Speed = speed, range from 0~100unit %
  • Return Value: none

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

• Function: Control myCobot moves in one direction in Cartesian space
  • Parameter Specification:
    • Direction Selection = axis, range from X,Y,Z,RX,RY,RZ
    • Direction of Joint Motion = Direction, range from -1/1
    • Specified Speed = speed, range from 0~100unit %
  • Return Value: none

jogStop();

• Function: Stops the specified direction of motion that has started
  • Return Value: none

pause();

• Function: Program pause
  • Return Value: none

resume();
• Function: Program continues to run
  • Return Value: none
  
  stop();

• Function: Program stop
  • Return Value: none

4. Running Status and Settings

getSpeed();

• Function: read the current running speed
  • Return Value: int tape, range from 0-100, unit %

setSpeed(int percentage);

• Function: set the running speed
  • Parameter Specification: percentage, range from 0-100, unit %

getFeedOverride(); (Not open at this time)

• Function: read FeedOverride
  • Return Value: float type of the array

sendFeedOverride(float feedOverride); (Not open at this time)

• Function: send FeedOverride

getAcceleration();(Not open at this time)

• Function: read acceleration
  • Return Value: float type of the array

setAcceleration(float acceleration); (Not open at this time)

• Function: set acceleration
  • Parameter Specification: acceleration range from 0-100 (Value range is not defined)

getJointMin(int joint);

• Function: read the joint minimal limit Angle
  • Parameter Specification: Joint Number = joint, range from 1-6
  • Return Value: float type of the array

getJointMax(int joint);

• Function: read the joint maximal limit Angle
  • Parameter Specification: Joint Number = joint, range from 1-6
  • Return Value: float type of the array

setJointMin(int joint, float angle);

• Function: set the joint minimal limit Angle
  • Parameter Specification: joint/servo motor number = joint, range from 1-6
  • Return Value: none

setJointMax(int joint, float angle);
5. Joint Servo Control

**isServoEnabled(int joint);**
- Function: Check whether the joint is properly connected
- Parameter Specification: Joint Number = joint, range from 1-6
- Return Value: Normal links return TRUE, on the contrary return FALSE

**isAllServoEnabled();**
- Function: Check whether all joins are properly connected
- Return Value: Normal links return TRUE, on the contrary return FALSE

**getServoData(int joint, byte data_id);**
- Function: Read the data of the specified address of joint
- Parameter Specification:
  - Joint Number = joint, range from 1-6
  - Data Address = data_id, refer to the following Figure for address
- Return Value: refer to the following Figure for the value range

**setServoCalibration(int joint);**
- Function: Calibrate the current position of the joint to zero Angle, the corresponding potential value is 2048
- Parameter Specification: joint number = joint, range from 1 – 6

**jointBrake();**
- Function: brake a single stand-alone
- Parameter Specification: joint number = joint, range from 1 – 6

**setPinMode(byte pin_no, byte pin_mode);**
- Function: set atom specified pin mode
- Parameter Specification:
  - Pin Number = pin_no, range from:19, 22, 23, 26, 32, 33
  - Pin mode = pin_mode, range from:0, 1
- Return Value: none

6. Atom IO Control

**setLEDRGB(byte r, byte g, byte b);**
- Function: set the color of the RGB lights of atom
- Parameter Specification:
  - Parameter of Red Light = r, range from 0x00 – 0xFF;
  - Parameter of Green Light = g, range from 0x00 – 0xFF;
  - Parameter of Blue Light = b, range from 0x00 – 0xFF;
- Return Value: none
setGripper(int data);

- Function: Set the jaw opening and closing
- Parameter Specification: data 0 is open, 1 is close

7. Coordinate Control Mode

setToolReference(Coords coords);

- Function: set coordinate system of tool
- Parameter Specification:
  - X,Y,Z range from -300-300.00 (Value range is not defined. If the value is beyond the range, the clue "inverse kinematics no solution" will be given)
  - RX,RY,RZ range from -180~180
- Value: none

setWorldReference(Coords coords);

- Function: set coordinate system of world
- Parameter Specification:
  - X,Y,Z range from -300-300.00 (Value range is not defined. If the value is beyond the range, the clue "inverse kinematics no solution" will be given)
  - RX,RY,RZ range from -180~180
- Return Value: none

getToolReference();

- Function: get coordinate system of tool
- Parameter Specification: none
- Return Value:
  - X,Y,Z range from -300-300.00 (Value range is not defined. If the value is beyond the range, the clue "inverse kinematics no solution" will be given)
  - RX,RY,RZ range from -180~180

getWorldReference();

- Function: get coordinate system of world
- Parameter Specification: none
- Return Value:
  - X,Y,Z range from -300-300.00 (Value range is not defined. If the value is beyond the range, the clue "inverse kinematics no solution" will be given)
  - RX,RY,RZ range from -180~180

setReferenceFrame(RFType rftype);

- Function: set coordinate system of frame
- Parameter Specification: RFType::BASE takes the robot base as the base coordinate, RFType::WORLD takes the coordinate
**getReferenceFrame();**

- **Function:** get coordinate system of flange
- **Parameter Specification:** none
- **Return Value:**
  - X,Y,Z range from -300-300.00 (Value range is not defined. If the value is beyond the range, the clue "inverse kinematics no solution" will be given)
  - unit mm
  - RX,RY,RZ range from -180~180

**setEndType(EndType end_type);**

- **Function:** set coordinate system of the end
- **Parameter Specification:** EndType::FLANGE set the end as flange, EndType::TOOL set the end to the tool end
- **Return Value:** none

**getEndType();**

- **Function:** get coordinate system of the end
- **Parameter Specification:** none
- **Return Value:**
  - X,Y,Z range from -300-300.00 (Value range is not defined. If the value is beyond the range, the clue "inverse kinematics no solution" will be given)
  - unit mm
  - RX,RY,RZ range from -180~180

**setGripperValue();**

- **Function:** Set the electric potential of the gripper. Get the current electric potential of the gripper before use
- **Parameter Specification:** Input Value (0-4095)
- **Return Value:** none

**setGripperIni();**

- **Function:** Set the gripper to zero
- **Parameter Specification:** none
- **Return Value:** none

**getGripperValue();**

- **Function:** get the value of the current electric potential
- **Parameter Specification:** none
- **Return Value:** return the value of the current electric potential, range from 0-4085

**setGripperState();**

- **Function:** Set the opening and closing of the gripper
- **Parameter Specification:** 0 represents open, 1 represents close
- **Return Value:** none
setDigitalOutput;

- Function: Set the working state of IO pins
- Parameter Specification: 0 input; 1 output; 2 pull_up_input
- Return Value: none

getDigitalInput;

- Function: Read input
- Parameter Specification: none
- Return Value: none

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

- Function: Set the PWM signal of the ATOM end IO output that specifies the duty-through ratio
- Parameter description: pin_no: IO serial number freq: clock frequency pin_write: ratio: 0 ~ 256; 128 indicates 50%.
- Return value: none
Test Program

Functional Specification

Burn the connect-test firmware, which can detect the connection of the device, and the Basic screen will show the linked device

Requirement

- Use MyStudio Basic -> connect-test
- Use MyStudio Atom -> AtomMain

Flow Chart

- Burn AtomMain for Atom
- Burn connect-test for Basic

Press Button C1
If there is any problem:
5.2 UIFlow

Press the web URL of uiFlow: https://flow.m5stack.com/

Operate as shown below:

Then you can start programming

Make the Atom screen green and adjust the angles of the six joints as shown in the figure.
5.3 Python

Programming in Python Environment

Identify development goals

PyMycobot is a Python package that communicates with myCobot on a serial port. It supports Python2, Python3.5 and later versions.

If you want to control myCobot by programming in Python, that is your choice.

How to install and use

Before using PyMycobot, make sure you have a myCobot and have it ready.

Pre-preparation:

- Make sure the top Atom burns into Atom and the bottom BASIC burns into Transponder.
- The firmware Atom and Transponder download address:
  https://github.com/elephantrobotics/myCobot/tree/main/Software

Pip

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

Notes:

- Only Atom2.6 and later versions are currently supported; if you are using previous versions, please install PyMycobot 1.0.7.

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

Installing from Source

```
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 serves as a secondary directory

If you don't want to install it, you can use it. First, you need to go to GitHub and download it locally.
Download Path:  https://github.com/elephantrobotics/pymycobot

Then, put the Pymycobot file in your project so that you can import it and use it.
pymycobot API

Class:

- MyCobot
  - Overall status
  - MDI mode and operation
  - JOG mode and operation
  - Running status and Settings
  - Servo control
  - Atom IO
- Angle
- Coord

MyCobot

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

Note: If no parameter is given, there is no parameter; if no return value is given, there is no return value

Overall status

**power_on()**

- **Description:** Robot arm power up.

**power_off()**

- **Description:** Robot arm power down.

**is_power_on()**

- **Description:** Adjust robot arm whether power on.
- **Returns**
  - 1: power on
  - 0: power off
  - -1: error

**set_free_mode()**

- **Description:** Set robot arm into free moving mode.
MDI mode and operation

get_angles()

- **Description**: Get the degree of all joints.
- **Returns**: list: A float list of all degree.

send_angle()

- **Description**: Send one degree of joint to robot arm.
- **Parameters**
  - id: Joint id (genre.Angle)
  - degree: degree value (float)
  - speed: (int) 0 ~ 100
- **Example**
  ```python
  from pymycobot.mycobot import MyCobot
  from pymycobot.genre import Angle
  mycobot = MyCobot('/dev/ttyUSB0')
  mycobot.send_angle(Angle.J2.value, 10, 50)
  ```

send_angles()

- **Description**: Send the degrees of all joints to robot arm.
- **Parameters**
  - degrees: a list of degree value (List[float])
  - speed: (int)
- **Example**
  ```python
  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()

- **Description**: Get the radians of all joints.
- **Returns**: list: A float list of radian.

send_radians()

- **Description**: Send the radians of all joint to robot arm.
- **Parameters**
degrees: a list of radian value (List[float])
speed: (int) 0 ~ 100

Example

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

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

get_coords()

- **Description**: Get the Coords from robot arm, coordinate system based on base.
- **Returns**: list: A float list of coord - [x, y, z, rx, ry, rz]

send_coord()

- **Description**: Send one coord to robot arm.
- **Parameters**
  - id: Coord id (genre.Coord)
  - coord: coord value (float)
  - speed: (int) 0 ~ 100
- **Example**

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

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

send_coords()

- **Description**: Send all coords to robot arm.
- **Parameters**
  - coords: a list of coords value (List[float])
  - speed: (int) 0 ~ 100
  - mode: (int): 0 - angular, 1 - linear
- **Example**

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

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

sync_send_angles()
• **Description:** Send the angle in synchronous state and return when the target point is reached

• **Parameters**
  - `id`: Joint id (genre.Angle)
  - `degree`: degree value (float)
  - `speed`: (int) 0 ~ 100

`sync_send_coords()`

• **Description:** Send the coord in synchronous state and return when the target point is reached

• **Parameters**
  - `coords`: a list of coords value (List[float])
  - `speed`: (int) 0 ~ 100
  - `mode`: (int): 0 - angular, 1 - linear

`pause()`

• **Description:** Pause movement.

`resume()`

• **Description:** Recovery movement.

`stop()`

• **Description:** Stop moving.

`is_paused()`

• **Description:** Judge whether the manipulator pauses or not.

• **Returns**:
  - 1 - paused
  - 0 - not paused
  - -1 - error

`is_in_position()`

• **Description:** Judge whether in the position.

• **Parameters**
  - `data`: A data list, angles or coords.
  - `flag`: Tag the data type, 0 - angles, 1 - coords.

• **Returns**
  - 1 - true
  - 0 - false
JOG mode and operation

jog_angle()

- **Description**: Jog control angle
- **Parameters**
  - joint_id: (int) 1 ~ 6
  - direction: 0 - decrease, 1 - increase
  - speed: 0 ~ 100

jog_coord()

- **Description**: Jog control coord.
- **Parameters**
  - coord_id: (int) 1 ~ 6
  - direction: 0 - decrease, 1 - increase
  - speed: 0 ~ 100

jog_stop()

- **Description**: Stop jog moving.

Running status and Settings

get_speed()

- **Description**: Get speed.
- **Returns**: speed: (int)

set_speed()

- **Description**: Set speed.
- **Parameters**: speed: (int) 0 ~ 100

get_joint_min_angle()

- **Description**: Gets the minimum movement angle of the specified joint
- **Parameters**: joint_id: (int)
- **Returns**: angle value (float)

get_joint_max_angle()
Servo control

is_servo_enable()

- Description: Determine whether all steering gears are connected
- Parameters: servo_id (int) 1 ~ 6
- Returns
  - e: disable
  - 1: enable
  - -1: error

is_all_servo_enable()

- Description: Determine whether the specified steering gear is connected
- Returns
  - e: disable
  - 1: enable
  - -1: error

release_servo()

- Description: Power off designated servo
- Parameters: servo_id: 1 ~ 6

focus_servo()

- Description: Power on designated servo
- Parameters: servo_id: 1 ~ 6

Atom IO

set_color()

- Description: Set the color of the light on the top of the robot arm.
- Parameters
  - r: 0 ~ 255
  - g: 0 ~ 255
  - b: 0 ~ 255
set_pin_mode()

- **Parameters**
  - pin_no (int):
  - pin_mode (int): 0 - input, 1 - output, 2 - input_pullup

set_digital_output()

- **Parameters**
  - pin_no (int):
  - pin_signal (int): 0 / 1

get_digital_input()

- **Parameters**: pin_no (int)
- **Return**: signal value

get_gripper_value()

- **Description**: Get gripper value
- **Return**: gripper value (int)

set_gripper_state()

- **Description**: Set gripper switch state
- **Parameters**
  - flag (int): 0 - open, 1 - close
  - speed (int): 0 ~ 100

set_gripper_value()

- **Description**: Set gripper value
- **Parameters**
  - value (int): 0 ~ 4096
  - speed (int): 0 ~ 100

set_gripper_ini()

- **Description**: Set the current position to zero, set current position value is 2048.

is_gripper_moving()

- **Description**: Judge whether the gripper is moving or not
- **Returns**
- 0: not moving
- 1: is moving
- -1: error data

**Angle**

```pythonrom pymycobot.genre import Angle
```

**Description**

Instance class of joint. It's recommended to use this class to select joint.

**Coord**

```pythonrom pymycobot.genre import Coord
```

**Description**

Instance class of coord. It's recommended to use this class to select coord.
ROS Introduction

ROS is abbreviation of Robot Operating System. ROS is a highly flexible software architecture for writing robotic software programs.

ROS icon:

![ROS.org](https://example.com/ROS.png)

ROS is open source, a post-operating system, or secondary operating system for robot control. It provides features similar to those provided by the operating system, including hardware abstract descriptions, underlying driver management, execution of common functionality, inter-program messaging, and package management, as well as tool programs and libraries for acquiring, building, writing, and running multi-machine integrated programs.

ROS's primary design goal is to increase code reuse in the field of robotics development. ROS is a framework of distributed processes (or "nodes") that are encapsulated in packages and feature packs that are easy to share and publish. ROS also supports a joint system similar to a code repository, which can also enable engineering collaboration and release. This design enables the development of an engineering implementation to make completely independent decisions (without ROS restrictions) from the file system to the user interface. At the same time, all projects can be integrated with the basic tools of ROS.

MoveIt Introduction

MoveIt! is the most advanced software for robotic arm movement operations and is used on more than 100 robots. It combines the latest results in motion planning, control, 3D perception, operations control, control and navigation, provides an easy-to-use platform for developing advanced robotics applications, and provides an integrated software platform for the design and evaluation of new robot products in industrial, commercial and research and development fields.

MoveIt icon:

![MoveIt](https://example.com/MoveIt.png)
Basic development environments require the installation of robotic operating systems ROS, MoveIt, and git version managers, which are described below.

**ROS Installation**

1. **Choose version**

   ROS has a one-to-one relationship with ubuntu, and different versions of ubuntu correspond to different versions of ROS, the website below: [http://wiki.ros.org/Distributions](http://wiki.ros.org/Distributions)

   If the version is different, the download will fail. The system we selected here is Ubuntu 16.04, corresponding to ros version ROS Kinetic Kame.

2. **Begin installation**

   2.1. **Add source**

      There are no ROS software sources in the list of Ubuntu's own software sources, so you need to configure the ROS software source into the software list repository first, then you can download ROS and open a console terminal (Ctrl+Alt+T), Enter the following instructions:

      ```
      sudo sh -c 'echo "deb http://packages.ros.org/ros/ubuntu $(lsb_release -sc) main" > /etc/apt/sources.list.d/ros-latest.list'
      
      The results are as follows (the user password is required here, just enter the user password set when Ubuntu is installed)
      ```

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

   2.2. **Set the key**

      Configure the public network key, this step is to let the system confirm that our path is safe, so that download file is no problem, otherwise the download will be deleted immediately:

      ```
      sudo apt-key adv --keyserver hkp://keyserver.ubuntu.com:80 --recv-key C1CF6E31E6BADE
      
      The results of the execution are as follows:
      ```

      ```
      $ jerry@ubuntu$: sudo apt-key adv --keyserver hkp://keyserver.ubuntu.com:80 --recv-key C1CF6E31E6BADE
      
      
      $ sudo requesting key AB17C654 from hkp server keyserver.ubuntu.com
      
      ggpg key AB17C654: "Open Robotics <info@osrfoundation.org>" not changed
      
      ggpg Total number processed: 1
      
      ggpg Unchanged: 1
      ```
2.3. Installation

After adding a new software source, update the list of software sources:

```
sudo apt-get update
```

install ROS:

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

It is recommended to install a complete ROS to prevent the loss of libraries and dependencies.

The installation process takes a long time and requires patience

2.4. Configure the ROS environment to the system

`rosdep` allows you to easily install the source code that you want to compile, or be dependent on some system that the ROS core components need, and execute the following commands in turn at the terminal. Initialize `rosdep`:

```
sudo rosdep init
rosdep update
```

Once initialization is complete, in order to avoid having to re-take effect on the ROS functional path after each shutdown of the terminal window, we can configure the path into the environment variable so that the ROS functional path automatically takes effect each time a new terminal is opened. At the terminal, the following commands are executed in turn:

**Bash**

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

**Zsh**

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

2.5. Install the ROS extra dependence

Enter the following command at the terminal:

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

3. Verify and installation

The start-up of the ROS system requires a ROS Master, the node manager, and we can enter the roscore instruction at the terminal to start the ROS Master, to verify that the ROS was installed successfully, the following commands are executed at the terminal:

```
roscore
```

When the following interface is displayed, means the ROS installation is successful.

For more detailed installation instructions, you can refer to the official installation instructions, website: http://wiki.ros.org/ROS/Installation

**MoveIt Installation**

MoveIt is the component of a series of mobile operations in ros, including motion planning, collision detection, kinematics, 3D awareness, operation control and other functions.

1. Update the list of software sources

Enter the following command in the terminal window to update the list of software sources:

```
sudo apt-get update
```

2. Install MoveIt

Enter the following command in the terminal window to perform the installation of MoveIt:
git installation

1. Add a software source

Add the git-installed software source to the list of Ubuntu's software sources and enter the following command in the terminal window:

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

2. Update the list of software sources

Enter the following command in the terminal window to update the list of software sources:

```
sudo apt-get update
```

3. Install git

Enter the following command in the terminal window to perform the installation of git:

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

4. Verify and installation
Read git version, Enter the following command in the terminal window:

```
$ git --version
```

The git version number can be displayed in the terminal, as follows, for a successful installation.
Introduction

mycobot_ros is from ElephantRobotics, fits its desktop six-axis robotic arm mycobot.

The project address: [http://github.com/elephantrobotics/mycobot_ros](http://github.com/elephantrobotics/mycobot_ros)

Installation

**NOTE:**

- This package relies on ROS and MoveIT. Ensure a successful installation of ROS and MoveIT before use.
- The interaction of this package with the real robot arm is depend on PythonApi - pymycobot
- The project site of Api is: [https://github.com/elephantrobotics/pymycobot](https://github.com/elephantrobotics/pymycobot)
- Fast installation: `pip install pymycobot --upgrade`
- The way you install it depends on Git, make sure Git is installed on your computer.

The following text will refer to the ROS workspace path in your computer with `<ros-workspace>`, and make sure that you replace `<ros-workspace>` with your native’s real path when you execute the following command.

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

Open a command line and run:

```
roslaunch mycobot_ros mycobot_slider.launch
```

It will open the rviz and a slider assembly, and you'll see as the following:

You can then control the movement of the model in the rviz by dragging the slider. If you want the real mycobot to move with you, you need to open another command line and run:

```
rosrun mycobot_ros slider_control.py
```

It publishes the angle to mycobot in real time. The script supports the setting of port numbers and baud rates, with defaults of `"/dev/ttyUSB0"` and `115200`.

Model follow

In addition to the controls above, we can also allow the model to follow the real robot arm movement. Open a command line to run:

```
rosrun mycobot_ros follow_display.py
```

It will release the angle of the real robotic arm. The script supports setting port numbers and baud rates, which are defaulted to `"/dev/ttyUSB0"` and `115200`.

Then open another command line and run:

```
roslaunch mycobot_ros mycobot_follow.launch
```
It opens the rviz display model to follow the effect.
Keyboard control

Keyboard control was added to the mycobot_ros package and synchronized in real time in rviz. This function relies on pythonApi, so make sure to connect to the real robot arm.

Open a command line and run:

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

Due to the need for real robotic arm communication, the launch supports the setting of port numbers and Baud rates, which default to "/dev/ttyUSB0" and 115200.

The running results are as follows:

The information of myCobot is output from the command line as follows:
PARAMETERS
* /mycobot_services/baud: 115200
* /mycobot_services/port: /dev/ttyUSB0
* /robot_description: <?xml version="1.0...)
* /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-b8c8-d0c63728b379
process[rosout-1]: started with pid [1340]
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] [1620882819.435778]: ready

Next, open another command line and run:

rosrun mycobot_ros teleop_keyboard.py  
#or
rosrun mycobot_ros teleop_keyboard.py _speed:=70

You'll see the following output on the command line:
Mycobot Teleop Keyboard Controller

---------------------------

Moving 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

The parameters supported by the script:

- `_speed`: The movement speed of the robot arm
- `_change_percent`: The percentage of the movement distance.
MoveIt

`mycobot_ros` is now integrated into the MoveIt section

Open the command line to run:

```
roslaunch mycobot_ros mycobot_moveit.launch
```

The running results are as follows:

![Image of running results]

You can plan and execute to demonstrate the effect:

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

The script supports setting port numbers and baud rates, which are defaulted to "`/dev/ttyUSB0`" and 115200.
Please look forward to more...
1. Introduction to the RoboFlow

The RoboFlow is the operating system of the Elephant Collaborative Robot. It provides a human-computer interaction interface, which is convenient for operators to interact with the elephant robot and use the elephant robot correctly. That is to say, when the user uses the robot, most of the time is achieved by using the RoboFlow operating system.

For example, since the RoboFlow operating system runs in the teach pendant, the user can use the carrier of the teach pendant to perform manual robotics, programming, and other operations. The operating system OS can also be used to communicate with other robots or devices. All in all, with the advantages of friendly interface and rich functions, the appearance of the RoboFlow operating system makes it easier for users to start using the elephant robot. It makes everyone a commander of robots.

2 Main interface introduction

2.1 User Login Interface

When the controller is powered up and the system startup button on the teach pendant is pressed, the login page is entered. Figure 2-1 shows the login page of the OS3 operating system.
As shown in the login page, "TAUGHT BY PEOPLE, PERFORMED BY ROBOT", this is the concept that the elephant robotics has always insisted on making the operator become the commander of the robot. Let robots replace people with simple but repetitive tasks, work in harsh working conditions, and work that people can't do well (such as scenes with very high operational accuracy).

There are two types of login users for the OS3 operating system, one is the administrator and the other is the operator. The administrator has the highest authority to perform all operations, programming and setup. The operator can only load and run existing programs and check the statistical data information.

Administrators can add and modify multiple accounts in the settings, including operator accounts.

By clicking on the "Shutdown" button, the OS3 operating system can be turned off, and then the power supply can be turned off, thus the robot system can be shut down.

2.2 Main Menu

When the login is successful, it will go to the main menu page. The main menu of the OS3 operating system is shown in Figure 2-2.

On the left side of the main menu, there are four different options available:

- **Run Program**
  - Load an existing program directly and control the program to run. In this window, the user is not allowed to edit the program, but can only control the program running (such as control program running, pausing, stopping). At the same time, you can view the log and other related information during the running process of the program.

- **Edit Program**
- Users can choose to load an existing program in this window for modification, or they can choose to create a new blank program for editing. This window is the most frequently used function window for users. Besides programming, it can also perform other operations, such as manual manipulation of robots with "fast moving" function, forced control of IO signals, new variables, etc.

- **Statistics**
  - In this window, users can not only view the existing running data of the system, but also view related information saved before.

- **Settings**
  - In this window, the user can make basic settings for the robot. Such as robot open, robot off, account management, default program settings, etc.

In addition to these four main options, in the right window of the main menu, the user can see and open the most recently run program files. It is convenient for users to quickly find the most recently run program and control the program to run.

Click the "Shutdown" button to close the OS3 operating system; click the "Logout" button to log out.

### 2.3 Run Program

If the user selects "Run Program" in the main menu, it will enter the Run Program window. The running program window of the OS3 operating system is shown in Figure 2-3.
Users can enter the program window by loading the program they need to run. In this window, users can:

1. Get the basic information of the current (ready) running program, including program name, running status, user type.
2. Understand the statistical information of the current running program, such as the total number of runs and the rhythm, etc.
3. Read the relevant information of the current running program through the display window, such as IO, variables, logs, etc.
4. The most important thing is that the running program window is the channel for the user to load and run the program that has been debugged.

2.4 Edit Program

As shown in Figure 2-4, if the user selects "Write Program" in the main menu, two options will appear in the right window. The first is to create a program (optional blank or template) and the second is to load the program.

![Figure 2-4 Program Editing Window Options](image-url)
When first entering the program edit page, the user sees the initial page as shown in figure 2-5. In this page, common tools, initialization group and file management functions are provided. The role of the initialization group is to make it easy for the user to set the program content to run at the beginning of the program and run only once. For example, set the initial point, Io State, and so on before the robot starts formal work. File management provides users with a way to manage files. Users can manage program files here, and can copy them to the U disk, or from the U disk to the system memory. If the user wants to go back to the initial page during the programming process, click "Back".

**Function bar** As shown in Figure 2-6, the function bar has seven sub-options, which are divided into two categories, one is the program editing toolbar, and the other is the function editing column.
Program editing toolbar: Includes file option bar, edit option bar, and tool options bar. File: As shown in Figure 2-7, you can edit the program file. There are several operation options: Save, Save As, New, Load, Rename, and Exit.

Edit: As shown in Figure 2-8, you can edit the specific command content in the program file. There are cut, copy, paste, delete, disable, delete all, redo, undo options.

Tool options bar: As shown in Figure 2-9, it is a shortcut toolbar. When editing a robot program, the user often uses other tools to operate the robot. The tool options bar provides tools commonly used in program editing. Tools provided include: Quickmove, install, input and output, variables, logs, basic settings. For example, when editing a motion command, the user needs to manually operate the robot to a working position and teach the point. Then, the “Quickmove” tool in the toolbar can be selected to manually operate the robot to move to the position.

Functional Editing Window The OS3 operating system provides a rich set of features that allow users to perform complex functions with simple operations. Simple, but not simple functions, thus reducing the time workers to learn
programming, efficient accomplish their goals. The function editing bar includes basic functions, logic functions, advanced functions, and extended functions.

**Basic functions**: As shown in Figure 2-10, the basic functions include Waypoint, Gripper, Wait, Set, and Group, which are some basic functions commonly used by users.

![Figure 2-10 Basic functions](image)

- **Waypoint**: “Create new waypoints → Manually operate the robot to move the robot to the target point → Save current point → Running program”. With this series of operations, the user completes the goal of controlling the movement of the robot to the target point. If you create multiple waypoints, the motion of the robot will form a trajectory when you run the program.
- **Gripper**: The user can use this function to set the end effector. For example, it holds the workpiece or releases the workpiece.
- **Wait**: Users can use this function to delay, or wait for signals, conditions, and so on.
- **Set**: Users can use this function to set the input and output signals and custom conditions.
- **Group**: Users can use this function to edit the programs in the group.

**Logic function**: As shown in Figure 2-11, the logic functions include Loop, If/Else, Subprogram, Thread, Halt, Switch, to complete the program running process control.

![Figure 2-11 Logic function](image)

- **Loop**: The user can use this function to set a block to run cyclically multiple times.
- **If/Else**: The user can use this function to make conditional judgments, such as the determination of an input signal.
- **Subprogram**: The user can use this function to call a subroutine.
- **Thread**: Users can use this function to achieve robot multi-thread control.
- **Halt**: The user can use this function to control the program to pause, stop, restart, and pop up the window to display the corresponding prompt information.
- **Switch**: The user can use this function to make a condition selection and determine the content to be executed according to the value of the selected object.

**Advanced function**: As shown in Figure 2-12, advanced functions include Pallet, Assign to Var, Script, Popup, and Sender, all of which perform more complex operations.

![Figure 2-12 Advanced function](image)
• **Pallet**: Users can use this function to realize the robot to perform regular point movements. For example, the handling of workpieces in pallets, palletizing, etc. It is also possible to implement the fixed but irregular rendezvous motion of the robot in sequence.

• **Assign to Var**: Users can use this function to implement the assignment of a variable.

• **Script**: With the scripting feature, users can use the other common functions to achieve simple tasks while using the elephant robot, and can also use script programming to complete more complex tasks.

• **Popup**: Users can use this function to customize the pop-up window to display related information. This helps the operator to analyze the status of the current robot running program.

• **Sender**: Users can use this function to achieve TCP/IP communication between the elephant robot and other devices.

• **Extended function**: To adapt to different application scenarios, the OS3 operating system provides some extension functions, and even customizes functions according to important application scenarios proposed by users.

---

**Program Display Window** On the left side of the program editing page, there is a program display window as shown in Figure 2-14. The upper part is the name of the currently open program file, and the lower part is the program tree, which records the specific instructions and related information.
On the right side of the program editing page, there is a function editing window as shown in Figure 2-15, which shows the specific contents of the function instructions.

The user can make specific settings for the function instructions in this window. Quick control and current command renaming, deletion, and disabling are also provided here.

At the bottom of the program editing page, there is a program running control bar as shown in Figure 2-16. When debugging a program, users can use it to run, pause, stop and limit the running speed of the program.

When users use elephant robots, they can not only program and control the robot to complete the corresponding tasks, but also get some valuable statistical data in the statistical report window for analysis and statistics.

The statistical report window is divided into four sub-windows.

As shown in Figure 2-17, the general class counts the total running time, the number of active programs, and the specific information of active programs.
As shown in Figure 2-18, the program class counts the total running time and times of different programs.

As shown in Figure 2-19, the log lists the general information, warning information and error information recorded by the system during the user's use of OS3 operating system. This information helps users to determine what changes and feedback the system has made during the operation of the OS3 operating system.

In particular, error information can help users quickly locate the possible causes of errors, so as to solve problems according to error information and resume normal use.
Figure 2- 19 Log statistics

As shown in Figure 2-20, security statistics can help users to count security-related information, such as collision information, number of stops, etc.

Figure 2- 20 Security statistics

2.6Settings

In the configuration center, users can configure the robot. For example, power the robot, turn off the robot, set the load, time, network and so on. Initialization

The initialization configuration page is shown in Figure 2-21.

When robot movement is required, the user needs to enter the configuration center → initialize the robot, or shut down the robot. In the initialization page, you can also set the load and installation, these two are important configuration
content before other operations, such as configuration errors may cause unexpected situations.

Figure 2-21 Initialization

Default program

Figure 2-22 shows the default program settings page.

Figure 2-22 Default program

This function allows the user to set a default running program. As long as the system starts, the robot directly enters the running program window, and can start running the program and perform corresponding actions to complete the specified task.
If the user does not want the system to start and the startup program starts running, you can choose not to run.

Version update Figure 2-23 shows the version update settings page.

![Version update settings page](image)

Figure 2-23 Version update

This page allows users to update the OS3 operating system in two ways, one for local file updates and one for network updates. Figure 2-24 shows the account management page.

![Account management page](image)

Figure 2-24 Account management

Users can add new users, delete expired users, or change passwords on this page. On this page, the user can get all the account information. Language and unit The language and unit settings page are shown in Figure 2-25. At present,
the OS3 operating system supports Chinese and English and metric units. Other languages and units are increasing, so stay tuned!

Figure 2-25 Language and unit

Time Figure 2-26 shows the time setting page.

Figure 2-26 Time

The user can set the system time on the current page. If the "24-hour system" is not checked, the time display format defaults to 12-hour system. Touch screen calibration

Figure 2-27 shows the touch screen calibration instructions. The user clicks on "Start to Calibrate" to enter the calibration interface. The calibration interface will appear in sequence with four circles, as shown in the figure. The user needs to click the center of the circle with a touch pen, and each time the button is clicked,
the next circle will appear until all four circles appear. A pop-up window will appear indicating that the calibration is complete, and you can exit the calibration screen after confirming the pop-up.

If the calibration times out or the steps are wrong, a pop-up prompts the calibration failure. At this point, you can confirm to exit the calibration interface and return to the page in Figure 2-27 to recalibrate.

![Image: Touch Screen](image1)

**Figure 2-27 Touch Screen**

**About us** As shown in Figure 2-28, it is about our page.

![Image: About Us](image2)

**Figure 2-28 About us**

This page shows basic information about the operating system of the OS3 operating system. For example, the model of the robot used is the Elephant series, version information, and so on.
3 Introduction to common tools

3.1 Quickmove

Quickmove is a tool that users use frequently when they operate the robot quickly and manually. Therefore, every user must be very familiar with the use of Quickmove using methods. The wrong operation may result in damage to the robot and its peripheral equipment, and even injuries to personnel.

As shown in Figure 3-1, Quickmove are mainly composed of 11 parts, which are described below.

![Quickmove Diagram](image)

**Figure 3-1 Quickmove**

Motion Control Mode in Cartesian Coordinate System

As shown in Figure 3-2, over-fixed-point O, three axes perpendicular to each other, all with O as the origin and generally with the same unit of length. These three axes are called x-axis (horizontal axis), y-axis (vertical axis), and z-axis (vertical axis), which are collectively referred to as coordinate axes. The x-axis and y-axis are usually arranged on a horizontal plane, while the z-axis is a plumb line. Their positive direction is in accordance with the right-hand rule, that is, holding the z-axis with the right hand. When the four fingers of the right hand turn from the positive x-axis to the positive y-axis from the $\pi/2$ angle, the thumb is
pointed to the positive direction of the z-axis. Such three axes form a spatial Cartesian coordinate system, and point O is called the coordinate origin. This constitutes a Cartesian coordinate.

There are three planes in the three-dimensional Cartesian coordinate system, XY-plane, YZ-plane, and XZ-plane. These three planes divide the three-dimensional space into eight parts, called octant spaces. The three coordinates of each point of the first limit are positive values.

Figure 3-2 Cartesian coordinate system Direction callout diagram

As shown in Figure 3-3, the robot can be controlled to move in the direction of the Cartesian coordinate system by clicking the key corresponding to the direction of the Cartesian coordinate system.

Figure 3-3 Cartesian coordinate system motion control mode button

3D View This window marks the direction of movement of the six joints of the robot.

- User coordinate system
- Motion mode switching

There are two main motion modes for manual manipulation robots.

- **Continuous motion mode**: The user presses the motion control button and allows the robot to move until the user releases the button and the robot stops. For example, if you press the + X direction motion control button, you need to hold the button all the time. The time of pressing the motion control key determines the distance of the robot in the + X direction.

- **Stepping motion mode**: Manual manipulation robot step motion, click "step motion" and open the step setting window as shown in Figure 3-4. Then the user chooses the step in this window and clicks the key of the target control.
direction. Every time he clicks, the robot takes a step. For example, choose a 1 mm step, click the X-direction movement control button, and every time you click the button, the robot will move 1 mm in the + X direction.

![Step-by-step motion step setting window](image)

**Figure 3-4** Step-by-step motion step setting window

**Speed** As shown in Figure 3-5, the control speed of the manual manipulator can be set here. Speed can be set from 0 to 100%.

![Speed setting window](image)

**Figure 3-5** Speed setting window

**Move to origin** By selecting the icon shown in Figure 3-6, the robot can be controlled to return to its original position and posture.

![Move to Origin](image)

**Figure 3-6** Move to origin
**Freemove** Select the icon shown in Figure 3-7 to switch to the drag mode.

![Freemove](image)

**Return** Click on the icon shown in Figure 3-8 to return to the programming operation window.

![Back](image)

**Joint control** Serial robot is an open kinematic chain of the robot. It is formed by a series of connecting rods connected in series with a rotating joint or a moving joint. The elephant cooperative robot belongs to a 6-axis serial robot. It drives the relative motion of the connecting rod by using motor drivers to drive the movement of 6 joints, allowing the end operator to reach the right posture. The Joint control window shown in Figure 3-9 provides the keys used by the operator to manually manipulate the robot and control the robot for joint movement using the instructor. The control buttons for each joint are divided into 2 directions, and
the angle data of each axis can be seen.

![Joint Control](image)

**Figure 3-9 Joint motion mode control window**

**Coordinate position** As shown in Figure 3-10, this window displays the coordinate position corresponding to the coordinate control.

![TCP (Unit: mm)](image)

**Figure 3-10 Coordinate position display window**

Status display button: The button has two states, "OK" (displays green) and "Reset" (displays red). When the display is normal, it indicates that the robot is working properly, and when the reset is displayed, the robot is abnormal, the anomaly needs to be lifted and the key is clicked for reset.

**3.2 Installation**
As shown in Figure 3-11, there are three submenus inside the installation tool. It is used to implement the loading/saving installation configuration, security configuration, and network configuration of the elephant robot.

Figure 3-11 Load/save installation

**Security configuration:** As shown in Figure 3-12, set the torque limit and brake control of the elephant robot.

Figure 3-12 Security configuration
Network Configuration: As shown in Figure 3-13, configure the IP address and port number of the Ethernet communication here.

![Figure 3-13 Network settings](image)

### 3.3 Input and output configuration

The robot system has a total of 16 digital input signals and 16 digital output signals. As shown in Figure 3-14, the input and output signals can be configured and monitored in this window, and the output signals can be forcibly output. IO configuration files can also be saved and loaded on this page. As shown in Figure 3-15, it is an input/output interface description corresponding to the page shown in Figure 3-14.

![Figure 3-14 Input and output configuration](image)
It should be noted that the input public terminal needs to be connected to 24V power supply. It can be determined whether the input is active high or active low according to the common configuration (hardware connection determines 24V or 0V). As shown in Figure 3-16, when the common terminal is connected to 24V, once an external device inputs 0V, the input signal is in the “High” state, otherwise it is in the “Low” state; vice versa.

As shown in Figure 3-17, the output is 24V when there is no output. Once the output is turned on (that is, the output is High), the output is 0V.
3.4 Variable

As shown in Figure 3-18, in the variable editing window, you can add, edit, and delete variables.

As shown in Figure 3-19, there are 5 types of editable variable types. They are string variables, pose variables, floating point variables, integer variables, and Boolean variables. On this page, you can edit the variable name and initial value.

3.5 Log
As shown in Figure 3-20, you can view information about the robot running status, error information, and alarm information in the running log window. Click the "Information", "Warning" and "Error" buttons to sort the corresponding logs.

Users can save logs to a local folder. Log files are a record of how the system is performing, helping users to have a clearer understanding of the system and also helping to troubleshoot errors.

Figure 3-20 Running log

3.6 Basic Settings
As shown in Figure 3-21, the basic settings page provides a common setting channel, allowing the user to quickly set up some functions, such as free movement related parameter settings, even when leaving the programming window while writing the program.

![Figure 3-21 Basic Settings](image)

**4 Function instruction**

**4.1 Basic function**

**4.1.1 Waypoint**

There are four types of waypoints: Absolute points, Relative points, Shared points, and Variables. These four types are side-by-side. Under one waypoint command, you can only choose one.

**Absolute point**: The absolute point is a description of the actual pose of the robot.

- That is, as long as the robot records the absolute point, the next time the instruction is executed, regardless of the position of the robot (other settings unchanged), will reproduce the original teaching of the absolute point of posture.

The specific configuration page for absolute points is shown in Figure 4-1.
As shown in Figure 4-2, there are two formats for the representation of absolute points, namely Cartesian coordinate system coordinate values and joint angles. Among them, the Cartesian coordinate system coordinate value records the position and attitude of the robot TCP relative to the base coordinate system (in mm). The joint angle is a direct record of the actual angle of each axis (in degree, degrees).

Waypoint control Save current point This button is used to save the current pose data of the robot. Move to this point If you need to verify the teaching point or move to the teaching point for some operations, press and hold the button until the robot moves to the current teaching point. If the current teaching point is no longer needed, this button is used to clear the current teaching point. Advanced Features Shared configuration: This feature is being debugged, so stay tuned! Advanced configuration As shown in Figure 4-3, in the advanced configuration page, the user can set the movement mode, proximity mode, command speed, and torque limit.
Relative point: The relative point is used in a situation where a certain displacement is required based on a corresponding point of the movement instruction on the robot/ an absolute point/ variable point offset. The displacement can be a distance in a single direction, or a superposition of displacements in multiple directions, and can also teach a segment to offset. Figure 4-4 shows the specific configuration page of the relative point.
Direct input (relative movement) As shown in Figure 4-5, you can directly enter the coordinate value / joint angle.

Regardless of whether the coordinate value or the joint angle is input, one or more of the six values are selected according to the offset requirement, and not every value is required to be input.

For example, as shown in Figure 4-6, in the actual pickup and placement process, it is necessary to set a transition point above the target placement position. At this time, we can set a path command as absolute point, control the robot (at this time the robot should be holding the state of the workpiece) to move to the placement point, click to save the current point, which generates the command line 2 shown in Figure 4-6. Then click on the basic function - waypoint: select the relative point, set the z-direction of the icon to increase the relative point of 50mm, then the robot will move to the position of the transition point after running the last sentence. In the actual pickup and placement process, other instructions, such as setting instructions, may be added between the two instructions to open the gripper.
In addition to offset based on the position of the last motion instruction, relative point instruction can also be offset based on a Waypoint or Variable point.

The "Move to this" button verifies the offset motion, and "Clear Saved Points" clears the currently entered content.

Reference move: By teaching two points, a path is generated, based on the current point, and the track is reproduced.

Advanced Features: The advanced configuration of the absolute point is not repeated here.

*Shared point:* The share point can use the location of other waypoints. Figure 4-7 shows the specific configuration page of the share point.
Shared point: Select the point you want to share in the box, you can keep pressing the "Move to this point" button to control the robot to move to that point. If you click "Clear Saved Points" to clear the current share point.

Advanced Features: The advanced configuration of the absolute point is not repeated here.

Variable: The waypoint can be assigned by a variable. The user can use the communication method to obtain the waypoint location from other devices.

Figure 4-8 shows the specific configuration page of the variable point.

Variable assignment: The user can select the associated pose variable, and "Move to this point" can check whether the pose is the target pose.

Advanced Features: The advanced configuration of the absolute point is not repeated here.

![Variable Point Configuration Page](image)

4.1.2 Gripper Figure 4-9 shows the specific configuration page of the gripper.
Figure 4-9 Gripper

The user defines and controls the gripper through a simple function.

- 1 Select gripper
- 2 Set existing grippers
- 3 Select the gripper, you can edit or delete the existing gripper.
- 4 Define new grippers 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。
Set the saved state

- Fully open: The option in the execution gripper definition is the "open" state.
- Completely off: The option in the execution gripper definition is "off" status.
- Debug control
- Open the gripper: Manual operation performs the option of the "Open" state in the gripper definition.
- Close the gripper: Manual operation performs the option of the "Close" state in the gripper definition.

4.1.3 Wait As shown in Figure 4-11, there are four modes for waiting for instructions.

- Waiting time: The delay time can be set in seconds.
- Waiting for the input signal: The state of the input signal is judged and waits until it meets the set input signal state condition.
- Waiting for the output signal: The state of the output signal is judged and waits until it meets the set output signal state condition.
- Waiting conditions: You can customize the wait condition and wait until the wait condition is met.
4.1.4 **Set**  As shown in Figure 4-12, the setup command has four modes of selection.

- **Set IO**: Set the state of the output signal. In addition to selecting the set output signal to determine whether it is on or off, you can also set the time that the signal is held.
- **Set conditions**: Customize the content of the settings.
- **Set TCP** (i.e. tool center point).
- **Set the load**.

4.1.5 **Group**  As shown in Figure 4-13, the group instructions provide common combination templates, such as grabbing and placing combinations.
When users use group instruction, such as grabbing and placing combinations, they can modify parameters and teach Waypoints directly on the basis of template programs, or they can add or delete instructions freely according to their needs.

The user can simplify the process of finding instructions by using the group instruction. And it is more convenient and quicker to complete the programming of the corresponding project.

4.2 Logic function

4.2.1 Loop Loop instructions can repeat all instructions within a loop for a certain number of times. As shown in Figure 4-14, the number of loops can be represented by a constant or a variable or an expression.
4.2.2 If/Else

Judging the set conditions allows the program to read the data, determine and determine what to do next.

If/Else can be used to determine the I/O signal and can also be used to determine other conditions.

If/Else consists of three parts: If, Else If, and Else. The relationship between these three parts is as follows:

Except that If is an integral part, the remaining two are optional parts;

If both If, Else If, and Else exist, the program will first read If, then read Else If, Else If ... Else. The relationship between the three is shown in Figure 4-15:

![Figure 4-15 Relationship between If, Else If, and Else](image)

There can be more than one Else If, but there is only one If, and if you choose to add Else, you can only have one Else.

You can delete Else If or Else, but if you delete If, delete all Else If and Else.

Figure 4-16 shows the setting page of the conditional judgment command.
Figure 4-17 If/Else

As shown in the figure above, if the condition following "If" is met, the robot will move to waypoint 1; if it meets the condition followed by "Else if", it will move to waypoint 2; if both conditions are not met, the "Else" corresponding block will be executed, that is, the robot will move to the waypoint 3.

4.2.3 Subprogram

As shown in Figure 4-17, other subroutines can be called using this instruction. The main program can use multiple subroutines, but there are no subroutines in the subprogram.
As shown in Figure 4-18, you can view and edit subroutines in the main program. If you edit the subroutine, please note that it will not take effect until it is saved.

Figure 4-18 Display subroutine

4.2.4 Thread The thread runs along the main program. It is used to check signals such as emergency buttons or safety light curtains. As shown in Figure 4-19, you can set the running interval of threads.

Note that motion instructions are not allowed in threads. 4.2.5 Halt The pause command is used to control the robot to pause, stop, and resume. Figure 4-20 shows the specific configuration page for the pause command.

- When setting the pause and stop status, you can also select “Show Popup” to customize the contents displayed by the popup.
- Set the restart state. When the program runs to this instruction, it will start running again from the first instruction at the beginning.
4.2.6 Switch As shown in Figure 4-21, the conditional selection instruction is used to judge the value of a variable.

Corresponding to different conditional values, how many conditional values need to be judged and how many cases, you can open each case, increase the corresponding execution instructions. For example, to judge the integer variable A, set two cases, if A is 1, execute the first route instruction, if A is 2, execute the second route instruction.

If only a few variables are judged, and other cases are handled uniformly, we need to select "default" and add corresponding instructions to the switch.
4.3 Advanced function

4.3.1 Pallet  The pallet instruction allows the user to teach only a few points, through which the position of the other points can be calculated by the robotic system. Running this instruction can control the movement of the robot to these points. As shown in Figure 4-22, you can select a line, plane, cube, discrete point.

As shown in Figure 4-23, after you select line, select the number of points, and the line will be split evenly based on the number of points. These points are the split point. The user determines this line by teaching two points.

Figure 4- 22 Pallet type Selection

Figure 4- 23 Line
As shown in Figure 4-24, after selecting "Plane", select the number of points of the two axes, and the plane is divided equally. These points are the dividing points. This plane is determined by teaching four points.

Figure 4-24 Plane

As shown in Figure 4-25, after selecting "Cube", select the number of points of the three axes, and the cube is divided equally. These points are the dividing points. Determine this cube by teaching eight points.

Figure 4-25 Cube

As shown in Figure 4-26, when "Discrete Point" is selected, the number of points is selected to teach different points. That is, a discrete point is a collection of multiple points.
4.3.2 **Assign to var** As shown in Figure 4-27, this command can assign values to integer variables and string variables. You can also use the "set variables" to directly set the value of the variable according to the instruction.

4.3.3 **Script** Script instructions can be used to edit complex instructions, providing a richer set of functional instructions. Figure 4-28 shows the specific configuration page of the script command. There are two types of setup scripts, one is a single-line expression and the other is a multi-line script.
4.3.4 Popup

The pop-up command allows the user to customize the pop-up window. In other words, when this command is executed, a pop-up window appears, and the pop-up content is user-defined content. As shown in Figure 4-29, there are three types of pop-up windows: information, warnings, and errors. The user selects one and customizes the pop-up content.

There are also three kinds of pop-up window control: continue the program (logging), that is, do not pop the window, just display the contents of the pop-up window to the log, and the program continues to run; When the window is popped, the program is paused, that is, the pop-up window appears, and the program is suspended; When the window is popped, the program stops, and the pop-up window appears, and the program stops running.
4.3.5 Sender

If TCP/IP communication is to be performed, the robot system must set the IP and port number as a client or server to communicate with other devices.

The sender allows the user to set up a TCP/IP connection. Figure 4-30 shows the specific configuration page of the Sender instruction.

If the robot system acts as a client, the IP address filled in is the IP address of the external device of the server, and the port number corresponds to the port number assigned to the robot system by the server. When the server is in the state of monitoring, it can communicate with the server by clicking the "connection" button.

If the robot system serves as a server, the IP address filled in is the local IP address, and the port number corresponds to the port number assigned to the client device. Click on the "monitor" button, at which point the client device can connect to the robot system. In the client list, you can view the IP addresses and port numbers of all clients.

After establishing communication, data can be sent and received.

![Sender Configuration](image)

Figure 4-30 Sender

5 Quickly create a new runnable project

5.1 Flow Description

5.1.1 Ready to work

Precondition

- Complete robot system
- No problem

Prepare content
- Plug the power cord into the board that provides the AC 220V.
- Turn on the power switch. **Press the start button on the teach pendant.**

5.1.2 **Flow chart** As shown in Figure 5-1, it is the program editing flowchart.

5.2 **Specific steps**

5.2.1 **Login** After the system is successfully started, it will enter the login interface of the OS3 operating system as shown in Figure 5-2.
Select the login user name "Admin" or other administrator user name (only administrator permissions are allowed to edit and debug the program), click on the password box will pop up as shown in Figure 5-3.

The login password corresponding to the default administrator user “Admin” is “aaa” (if the other administrator user name is selected, enter the corresponding login password), enter the password and click “OK”, and return to the interface of Figure 5-2. Then click "Login" to log in successfully. **5.2.2 Power on** After the login is successful, the main menu interface shown in Figure 5-4 will be displayed.
In the main menu interface, select “Settings”, it will enter the interface as shown in Figure 5-5 (this time has not been powered).

To ensure that the emergency stop knob is not pressed, click on the “Start Robot” button as shown in Figure 5-5. The interface will change and the “Powering On” icon as shown in Figure 5-6 will appear. If the power is turned on successfully, the “I’m OK!” status shown in Figure 5-7 will appear. If it fails, check if you are missing any steps.

After completing the previous step, return to the main menu by pressing the motor “<Main Menu” button in the configuration center.
5.2.3 New blank program As shown in Figure 5-8, click “Program Robot” and then select “Empty Program”.

![Figure 5- 6 Powering up](image1)

![Figure 5- 7 Power on](image2)
After performing the previous step, enter the program editing interface as shown in Figure 5-9.

5.2.4 Add and edit instructions

As shown in Figure 5-10, add two waypoints: absolute point, and teach two points (that is, use the Quickmove to manually operate the robot, control the robot to move to a certain pose, return, click "Save Current Point" The teaching steps of the two points are the same. To verify the save point, press and hold the "Move to this point" button to manually control the robot to move to the teaching point.).
After editing is complete, please note that the program file is saved. Click “Save” in the file option bar as shown in Figure 5-10, and the window shown in Figure 5-11 will pop up. Click on “File Name” and the input keyboard shown in Figure 5-12 will appear. After entering the file name, click “OK”. Then go back to the save interface, click “OK”, the program file is saved successfully. After the save is successful, as shown in Figure 5-13, the program name in the upper left corner of the program editing interface will be changed.

Figure 5-10 Program editing

Figure 5-11
5.2.5 Program Debugging  As shown in Figure 5-14, in addition to the "Next" and "Run" functions provided in the program run control bar, click "Advanced" to enter the more settings interface.

Among them, the "Next" function corresponds to step by step execution of the program, click to run only one step at a time, if you need to continue to run, continue to click "Next." The "Run" function corresponds to automatically running the program once.

In "Advanced ", you can set the number of cycles to run, or you can run in an infinite loop. You can also control whether the program runs in automatic or manual mode. In the automatic mode, you can use "Next", "Run" and cycle operation. In the interface shown in Figure 5-14, select "Manual Run Mode" and then select "Run" or "Infinite Loop" in the loop run. You can enter the running interface in manual operation mode as shown in Figure 5-15.
If you use manual mode to debug the program, you need to keep pressing the "Press Down" button to continue running. If you release the button, the program pauses and presses again to continue.

5.2.6 **Save and run the program**

If debugging is complete, make sure you have saved the debugged program. After returning to the main menu, select “Run Program”. The pop-up window shown in Figure 5-16 will appear. Select the program to complete the debugging and click “OK”.

![Figure 5-14 Program debugging](image1)

![Figure 5-15 Manual mode to debug the program](image2)
After selecting the program, it will enter the running program interface as shown in Figure 5-17. In this interface, you can run the program to view the program running information.

If you are sure that the program will continue to run in the near future, you can also select it in the Settings - Default Program. In this way, as long as the system is started, it will automatically jump to the “Run Program” interface. After the power is turned on successfully, click “Run” to run the program.
6 Communications and messages

Note: When use a communication protocol to communicate directly you need to burn Transponder in Basic and the latest version of AtomMain in Atom.

6.1 Communication Settings

Make sure your communication Settings are as follows

- Bus Interface: USB Type-C
- Baud Rate: 115200
- Date Bits: 8
- Parity Check: none
- Stop Bit: 1

6.2 Command Frame Description & Single Instruction Parsing

The main BASIC sends data to the Atom, and the Atom parses the data after receiving it. For example, an instruction containing Return Value will be returned to the BASIC within 500ms.

6.3 Format for Sending and Receiving Command Frames

All commands are hexadecimal, and the format of send and receive is the same.

Each communication command must contain the following five parts, part 3 and 4 of which can be null.

- **Command Frame Head: 0xFE 0xFE**
  - Fixed
  - Included
- **Effective Command Length : 0x02 ~ 0x10**
  - The length of all the following commands
  - Included
- **Command Sequence Number: 00 ~ 8F**
  - Various commands have been developed
  - Null
- **Command Content: some**
  - Null
- **End of the Command: 0XFA**
  - Fixed
  - Included

### 6.4 Single Instruction Parsing

The main BASIC sends data to the Atom, and the Atom parses the data after receiving it. For example, an instruction containing Return Value will be returned to the BASIC within 500ms.

<table>
<thead>
<tr>
<th>Type</th>
<th>Data Description</th>
<th>Data Length</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Command frame</td>
<td>First byte 0</td>
<td>1</td>
<td>The frame head identification, 0xfe</td>
</tr>
<tr>
<td></td>
<td>First byte 1</td>
<td>1</td>
<td>The frame head identification, 0xfe</td>
</tr>
<tr>
<td></td>
<td>Data length byte</td>
<td>1</td>
<td>Different instructions correspond to different lengths of data</td>
</tr>
<tr>
<td></td>
<td>Command byte</td>
<td>1</td>
<td>Depends on different commands</td>
</tr>
<tr>
<td>Data frame</td>
<td>Data</td>
<td>0-16</td>
<td>Command attached data, depends on different commands</td>
</tr>
<tr>
<td>End frame</td>
<td>End the byte</td>
<td>1</td>
<td>Stop bit, 0xfa</td>
</tr>
</tbody>
</table>

1). **Atom Power On**

<table>
<thead>
<tr>
<th>Data field</th>
<th>Description</th>
<th>Data</th>
</tr>
</thead>
<tbody>
<tr>
<td>Data[0]</td>
<td>Identify the frame</td>
<td>0xfe</td>
</tr>
<tr>
<td>Data[1]</td>
<td>Identify the frame</td>
<td>0xfe</td>
</tr>
<tr>
<td>Data[2]</td>
<td>Data length frame</td>
<td>0x02</td>
</tr>
<tr>
<td>Data[3]</td>
<td>Instruction frame</td>
<td>0x10</td>
</tr>
<tr>
<td>Data[4]</td>
<td>End frame</td>
<td>0xfa</td>
</tr>
</tbody>
</table>

Serial port sending example: FE FE 02 10 FA

No Return Value

2). **Atom Power Off**
<table>
<thead>
<tr>
<th>Data field</th>
<th>Description</th>
<th>Data</th>
</tr>
</thead>
<tbody>
<tr>
<td>Data[0]</td>
<td>Identify the frame</td>
<td>0xfe</td>
</tr>
<tr>
<td>Data[1]</td>
<td>Identify the frame</td>
<td>0xfe</td>
</tr>
<tr>
<td>Data[2]</td>
<td>Data length frame</td>
<td>0x02</td>
</tr>
<tr>
<td>Data[3]</td>
<td>Instruction frame</td>
<td>0x11</td>
</tr>
<tr>
<td>Data[4]</td>
<td>End frame</td>
<td>0xfa</td>
</tr>
</tbody>
</table>

Serial port sending example: FE FE 02 11 FA

**NO Return Value**

3). Atom Status Inquiry

<table>
<thead>
<tr>
<th>Data field</th>
<th>Description</th>
<th>Data</th>
</tr>
</thead>
<tbody>
<tr>
<td>Data[0]</td>
<td>Identify the frame</td>
<td>0xfe</td>
</tr>
<tr>
<td>Data[1]</td>
<td>Identify the frame</td>
<td>0xfe</td>
</tr>
<tr>
<td>Data[2]</td>
<td>Data length frame</td>
<td>0x02</td>
</tr>
<tr>
<td>Data[3]</td>
<td>Instruction frame</td>
<td>0x12</td>
</tr>
<tr>
<td>Data[4]</td>
<td>End frame</td>
<td>0xfe</td>
</tr>
</tbody>
</table>

Serial port sending example: FE FE 02 12 FA

**Return Value**

<table>
<thead>
<tr>
<th>Data field</th>
<th>Description</th>
<th>Data</th>
</tr>
</thead>
<tbody>
<tr>
<td>Data[0]</td>
<td>Return frame header</td>
<td>0xFE</td>
</tr>
<tr>
<td>Data[1]</td>
<td>Return frame header</td>
<td>0xFE</td>
</tr>
<tr>
<td>Data[2]</td>
<td>Return data length frame</td>
<td>0X02</td>
</tr>
<tr>
<td>Data[3]</td>
<td>Return instruction frame</td>
<td>0X12</td>
</tr>
<tr>
<td>Data[4]</td>
<td>Power on/off</td>
<td>0X01/0X00</td>
</tr>
<tr>
<td>Data[5]</td>
<td>End frame</td>
<td>0XFA</td>
</tr>
</tbody>
</table>

Serial port sending example: FE FE 02 12 00 FA

4). Read Angle (read position information)

<table>
<thead>
<tr>
<th>Data field</th>
<th>Description</th>
<th>Data</th>
</tr>
</thead>
<tbody>
<tr>
<td>Data[0]</td>
<td>Identify the frame</td>
<td>0xfe</td>
</tr>
<tr>
<td>Data[1]</td>
<td>Identify the frame</td>
<td>0xfe</td>
</tr>
<tr>
<td>Data[2]</td>
<td>Data length frame</td>
<td>0x02</td>
</tr>
<tr>
<td>Data[3]</td>
<td>Instruction frame</td>
<td>0x20</td>
</tr>
<tr>
<td>Data[4]</td>
<td>End frame</td>
<td>0xfa</td>
</tr>
</tbody>
</table>

Serial port sending example: FE FE 02 12 00 FA
Serial port sending example: FE FE 02 20 FA

Returns a data structure from Atom

<table>
<thead>
<tr>
<th>Data field</th>
<th>Description</th>
<th>Data</th>
</tr>
</thead>
<tbody>
<tr>
<td>Data[0]</td>
<td>Return frame header</td>
<td>0xFE</td>
</tr>
<tr>
<td>Data[1]</td>
<td>Return frame header</td>
<td>0xFE</td>
</tr>
<tr>
<td>Data[2]</td>
<td>Return data length frame</td>
<td>0x0E</td>
</tr>
<tr>
<td>Data[3]</td>
<td>Return instruction frame</td>
<td>0X20</td>
</tr>
<tr>
<td>Data[4]</td>
<td>Servo 1 high Angle</td>
<td>Angle1_low</td>
</tr>
<tr>
<td>Data[5]</td>
<td>Servo 1 low Angle</td>
<td>Angle1_high</td>
</tr>
<tr>
<td>Data[6]</td>
<td>Servo 1 high Angle</td>
<td>Angle2_low</td>
</tr>
<tr>
<td>Data[7]</td>
<td>Servo 1 low Angle</td>
<td>Angle2_high</td>
</tr>
<tr>
<td>Data[8]</td>
<td>Servo 1 high Angle</td>
<td>Angle3_low</td>
</tr>
<tr>
<td>Data[9]</td>
<td>Servo 1 low Angle</td>
<td>Angle3_high</td>
</tr>
<tr>
<td>Data[10]</td>
<td>Servo 1 high Angle</td>
<td>Angle4_low</td>
</tr>
<tr>
<td>Data[11]</td>
<td>Servo 1 low Angle</td>
<td>Angle4_high</td>
</tr>
<tr>
<td>Data[12]</td>
<td>Servo 1 high Angle</td>
<td>Angle5_low</td>
</tr>
<tr>
<td>Data[13]</td>
<td>Servo 1 low Angle</td>
<td>Angle5_high</td>
</tr>
<tr>
<td>Data[14]</td>
<td>Servo 1 high Angle</td>
<td>Angle6_low</td>
</tr>
<tr>
<td>Data[15]</td>
<td>Servo 1 low Angle</td>
<td>Angle6_high</td>
</tr>
<tr>
<td>Data[16]</td>
<td>End frame</td>
<td>0xFA</td>
</tr>
</tbody>
</table>

Serial port sending example: FE FE 0E 20 06 E6 EA 4E C4 81 0B BD EA C0 02 B6 FA

How to get the angle of joint 1 ?

temp = angle1_low+angle1_high*256

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

(The rest are the same)

5). Send Individual Angles
<table>
<thead>
<tr>
<th>Data field</th>
<th>Description</th>
<th>Data</th>
</tr>
</thead>
<tbody>
<tr>
<td>Data[0]</td>
<td>Identify the frame</td>
<td>0xfe</td>
</tr>
<tr>
<td>Data[1]</td>
<td>Identify the frame</td>
<td>0xfe</td>
</tr>
<tr>
<td>Data[2]</td>
<td>Data length frame</td>
<td>0x06</td>
</tr>
<tr>
<td>Data[3]</td>
<td>Instruction frame</td>
<td>0x21</td>
</tr>
<tr>
<td>Data[4]</td>
<td>Joint number</td>
<td>Joint_no</td>
</tr>
<tr>
<td>Data[6]</td>
<td>high Angle</td>
<td>Angle_high</td>
</tr>
<tr>
<td>Data[7]</td>
<td>low Angle</td>
<td>Angle_low</td>
</tr>
<tr>
<td>Data[8]</td>
<td>End frame</td>
<td>0xfa</td>
</tr>
</tbody>
</table>

Serial port sending example: FE FE 06 21 00 00 00 20 FA

The value of joint NO ranges from 0 to 5

Angle High: Data type Byte

Calculation: The Angle value is multiplied by 100 first to int and then to take the high hexadecimal byte

Angle Low: Data type Byte

Calculation: The Angle value is multiplied by 100 first to int and then to take the high hexadecimal byte

No Return Value

6). Send All Angles
<table>
<thead>
<tr>
<th>Data field</th>
<th>Description</th>
<th>Data</th>
</tr>
</thead>
<tbody>
<tr>
<td>Data[0]</td>
<td>Identify the frame</td>
<td>0xfe</td>
</tr>
<tr>
<td>Data[1]</td>
<td>Identify the frame</td>
<td>0xfe</td>
</tr>
<tr>
<td>Data[2]</td>
<td>Data length frame</td>
<td>0x0f</td>
</tr>
<tr>
<td>Data[3]</td>
<td>Instruction frame</td>
<td>0x22</td>
</tr>
<tr>
<td>Data[4]</td>
<td>Angle 1 is in low_byte</td>
<td>Angle1_low</td>
</tr>
<tr>
<td>Data[5]</td>
<td>Angle 1 is in high_byte</td>
<td>Angle1_high</td>
</tr>
<tr>
<td>Data[6]</td>
<td>Angle 2 is in low_byte</td>
<td>Angle2_low</td>
</tr>
<tr>
<td>Data[7]</td>
<td>Angle 2 is in high_byte</td>
<td>Angle2_high</td>
</tr>
<tr>
<td>Data[8]</td>
<td>Angle 3 is in low_byte</td>
<td>Angle3_low</td>
</tr>
<tr>
<td>Data[9]</td>
<td>Angle 3 is in high_byte</td>
<td>Angle3_high</td>
</tr>
<tr>
<td>Data[10]</td>
<td>Angle 4 is in low_byte</td>
<td>Angle4_low</td>
</tr>
<tr>
<td>Data[11]</td>
<td>Angle 4 is in high_byte</td>
<td>Angle4_high</td>
</tr>
<tr>
<td>Data[12]</td>
<td>Angle 5 is in low_byte</td>
<td>Angle5_low</td>
</tr>
<tr>
<td>Data[13]</td>
<td>Angle 5 is in high_byte</td>
<td>Angle5_high</td>
</tr>
<tr>
<td>Data[14]</td>
<td>Angle 6 is in low_byte</td>
<td>Angle6_low</td>
</tr>
<tr>
<td>Data[15]</td>
<td>Angle 6 is in high_byte</td>
<td>Angle6_high</td>
</tr>
<tr>
<td>Data[16]</td>
<td>Specified speed</td>
<td>Sp</td>
</tr>
<tr>
<td>Data[17]</td>
<td>End frame</td>
<td>0xfa</td>
</tr>
</tbody>
</table>

Serial port sending example: FE FE 0F 22 06 E6 EA 4E C4 81 0B BD EA C0 02 B6 FA

Angle High: Data type Byte

Calculation: The Angle value of Joint 1 is multiplied by 100 first to int and then to take the high hexadecimal byte

Angle Low: Data type Byte

Calculation: The Angle value of Joint 1 is multiplied by 100 first to int and then to take the high hexadecimal byte

(The rest are the same)

No Return Value

7). Read All Coordinates
<table>
<thead>
<tr>
<th>Data field</th>
<th>Description</th>
<th>Data</th>
</tr>
</thead>
<tbody>
<tr>
<td>Data[0]</td>
<td>Identify the frame</td>
<td>0xfe</td>
</tr>
<tr>
<td>Data[1]</td>
<td>Identify the frame</td>
<td>0xfe</td>
</tr>
<tr>
<td>Data[2]</td>
<td>Data length frame</td>
<td>0x02</td>
</tr>
<tr>
<td>Data[3]</td>
<td>Instruction frame</td>
<td>0x23</td>
</tr>
<tr>
<td>Data[4]</td>
<td>End frame</td>
<td>0xfa</td>
</tr>
</tbody>
</table>

Serial port sending example: FE FE 02 23 FA

Returns a data structure from Atom

<table>
<thead>
<tr>
<th>Data field</th>
<th>Description</th>
<th>Data</th>
</tr>
</thead>
<tbody>
<tr>
<td>Data[0]</td>
<td>Return frame header</td>
<td>0XFE</td>
</tr>
<tr>
<td>Data[1]</td>
<td>Return frame header</td>
<td>0XFE</td>
</tr>
<tr>
<td>Data[2]</td>
<td>Return data length frame</td>
<td>0X0E</td>
</tr>
<tr>
<td>Data[3]</td>
<td>Return instruction frame</td>
<td>0X23</td>
</tr>
<tr>
<td>Data[4]</td>
<td>Specified x is in a low coordinate</td>
<td>x_high</td>
</tr>
<tr>
<td>Data[5]</td>
<td>Specified x is in a high coordinate</td>
<td>x_low</td>
</tr>
<tr>
<td>Data[6]</td>
<td>Specified y is in a low coordinate</td>
<td>y_low</td>
</tr>
<tr>
<td>Data[7]</td>
<td>Specified y is in a high coordinate</td>
<td>y_high</td>
</tr>
<tr>
<td>Data[8]</td>
<td>Specified z is in a low coordinate</td>
<td>z_low</td>
</tr>
<tr>
<td>Data[9]</td>
<td>Specified z is in a high coordinate</td>
<td>z_high</td>
</tr>
<tr>
<td>Data[10]</td>
<td>Specified rx is in a low coordinate</td>
<td>rx_low</td>
</tr>
<tr>
<td>Data[11]</td>
<td>Specified rx is in a high coordinate</td>
<td>rx_high</td>
</tr>
<tr>
<td>Data[12]</td>
<td>Specified ry is in a low coordinate</td>
<td>ry_low</td>
</tr>
<tr>
<td>Data[13]</td>
<td>Specified ry is in a high coordinate</td>
<td>ry_high</td>
</tr>
<tr>
<td>Data[14]</td>
<td>Specified rz is in a low coordinate</td>
<td>rz_low</td>
</tr>
<tr>
<td>Data[15]</td>
<td>Specified rz is in a high coordinate</td>
<td>rz_high</td>
</tr>
<tr>
<td>Data[16]</td>
<td>End frame</td>
<td>0XFA</td>
</tr>
</tbody>
</table>

How to get x-coordinate?

\[ \text{temp} = x_{\text{low}} + x_{\text{high}} \times 256 \]

\[ \text{x-coordinate} = \left(\text{temp} > 33000 \ ?(\text{temp} - 65536) : \text{temp}\right) / 10 \]

(same as the y/z-coordinate)

How to get rx-coordinate?

\[ \text{temp} = rx_{\text{low}} + rx_{\text{high}} \times 256 \]

\[ \text{rx-coordinate} = \left(\text{temp} > 33000 \ ?(\text{temp} - 65536) : \text{temp}\right) / 10 \]
(same as the ry/rz-coordinate )

8). Send Individual Coordinates

<table>
<thead>
<tr>
<th>Data field</th>
<th>Description</th>
<th>Data</th>
</tr>
</thead>
<tbody>
<tr>
<td>Data[1]</td>
<td>Identify the frame</td>
<td>0xfe</td>
</tr>
<tr>
<td>Data[2]</td>
<td>Data length frame</td>
<td>0x06</td>
</tr>
<tr>
<td>Data[3]</td>
<td>Instruction frame</td>
<td>0x24</td>
</tr>
<tr>
<td>Data[4]</td>
<td>Specified coordinate</td>
<td>X/y/z/rx/ry/rz</td>
</tr>
<tr>
<td>Data[5]</td>
<td>Specified xyz/rxryrz is in a low parameter</td>
<td>Xyz/ rxryrz_low</td>
</tr>
<tr>
<td>Data[6]</td>
<td>Specified xyz/txrytz is in a high parameter</td>
<td>Xyz/txrytz_high</td>
</tr>
<tr>
<td>Data[7]</td>
<td>Specified speed</td>
<td>Sp</td>
</tr>
<tr>
<td>Data[8]</td>
<td>End frame</td>
<td>0xfa</td>
</tr>
</tbody>
</table>

Set the x-coordinate to be 100 and the target speed to be 20

Serial port sending example: FE FE 06 24 00 00 64 20 FA

Specify coordinate axis: data type Byte

Value range: 0~5

xyz_high: Data type Byte

Calculation: The x/y/z coordinate value is multiplied by 10 and then to take the high hexadecimal byte

xyz_low: Data type Byte

Calculation: The x/y/z coordinate value is multiplied by 10 and then to take the low hexadecimal byte

rxyz_high: Data type Byte

Calculation: The rx/ry/rz coordinate value is multiplied by 10 and then to take the high hexadecimal byte

rxyz_low: Data type Byte

Calculation: The rx/ry/rz coordinate value is multiplied by 10 and then to take the low hexadecimal byte

No Return Value

9). Send All Coordinates
<table>
<thead>
<tr>
<th>Data field</th>
<th>Description</th>
<th>Data</th>
</tr>
</thead>
<tbody>
<tr>
<td>Data[0]</td>
<td>Identify the frame</td>
<td>0xfe</td>
</tr>
<tr>
<td>Data[1]</td>
<td>Identify the frame</td>
<td>0xfe</td>
</tr>
<tr>
<td>Data[2]</td>
<td>Data length frame</td>
<td>0x10</td>
</tr>
<tr>
<td>Data[3]</td>
<td>Instruction frame</td>
<td>0x25</td>
</tr>
<tr>
<td>Data[4]</td>
<td>Specified x is in a low coordinate</td>
<td>X_low</td>
</tr>
<tr>
<td>Data[5]</td>
<td>Specified x is in a high coordinate</td>
<td>X_high</td>
</tr>
<tr>
<td>Data[6]</td>
<td>Specified y is in a low coordinate</td>
<td>Y_low</td>
</tr>
<tr>
<td>Data[7]</td>
<td>Specified y in a high coordinate</td>
<td>Y_high</td>
</tr>
<tr>
<td>Data[8]</td>
<td>Specified z is in a low coordinate</td>
<td>Z_low</td>
</tr>
<tr>
<td>Data[9]</td>
<td>Specified z is in a high coordinate</td>
<td>Z_high</td>
</tr>
<tr>
<td>Data[10]</td>
<td>Specified rx is in a low coordinate</td>
<td>Rx_low</td>
</tr>
<tr>
<td>Data[11]</td>
<td>Specified rx is in a high coordinate</td>
<td>Rx_high</td>
</tr>
<tr>
<td>Data[12]</td>
<td>Specified ry is in a low coordinate</td>
<td>Ry_low</td>
</tr>
<tr>
<td>Data[13]</td>
<td>Specified ry in a high coordinate</td>
<td>Ry_high</td>
</tr>
<tr>
<td>Data[14]</td>
<td>Specified rz is in a low coordinate</td>
<td>Rz_low</td>
</tr>
<tr>
<td>Data[15]</td>
<td>Specified rz is in a high coordinate</td>
<td>Rz_high</td>
</tr>
<tr>
<td>Data[16]</td>
<td>Specified speed</td>
<td>Sp</td>
</tr>
<tr>
<td>Data[17]</td>
<td>Mode</td>
<td>Mode</td>
</tr>
<tr>
<td>Data[18]</td>
<td>End frame</td>
<td>0xfa</td>
</tr>
</tbody>
</table>

Set the target point at the end of the robot (-14, -27, 275, -89.5, 0.7, -90.7) set the target speed to be 20

Serial port sending example: FE FE 10 25 FF 74 FE EE 0A C1 DD 05 00 48 DC 95 32 01 FA

x_high: Data type Byte

Calculation: The x coordinate value is multiplied by 10 and then to take the high hexadecimal byte

x_low: Data type Byte

Calculation: The x coordinate value is multiplied by 10 and then to take the low hexadecimal byte

(rx_same as the y/z-coordinate )

rx_high: Data type Byte

Calculation: The rx coordinate value is multiplied by 10 and then to take the high hexadecimal byte
rx_low: Data type Byte
Calculation: The rx coordinate value is multiplied by 10 and then to take the low hexadecimal byte
(same as the ry/rz-coordinate )
No Return Value

10). Specified Point Arrival Detection (under development)

<table>
<thead>
<tr>
<th>Data field</th>
<th>Description</th>
<th>Data</th>
</tr>
</thead>
<tbody>
<tr>
<td>Data[0]</td>
<td>Identify the frame</td>
<td>0xfe</td>
</tr>
<tr>
<td>Data[1]</td>
<td>Identify the frame</td>
<td>0xfe</td>
</tr>
<tr>
<td>Data[2]</td>
<td>Data length frame</td>
<td>0x10</td>
</tr>
<tr>
<td>Data[3]</td>
<td>Instruction frame</td>
<td>0x25</td>
</tr>
<tr>
<td>Data[4]</td>
<td>x is in a low coordinate</td>
<td>X_low</td>
</tr>
<tr>
<td>Data[5]</td>
<td>x is in a high coordinate</td>
<td>X_high</td>
</tr>
<tr>
<td>Data[6]</td>
<td>y is in a low coordinate</td>
<td>Y_low</td>
</tr>
<tr>
<td>Data[7]</td>
<td>y in a high coordinate</td>
<td>Y_high</td>
</tr>
<tr>
<td>Data[8]</td>
<td>z is in a low coordinate</td>
<td>Z_low</td>
</tr>
<tr>
<td>Data[9]</td>
<td>z in a high coordinate</td>
<td>Z_high</td>
</tr>
<tr>
<td>Data[10]</td>
<td>rx is in a low coordinate</td>
<td>Rx_low</td>
</tr>
<tr>
<td>Data[11]</td>
<td>rx is in a high coordinate</td>
<td>Rx_high</td>
</tr>
<tr>
<td>Data[12]</td>
<td>ry is in a low coordinate</td>
<td>Ry_low</td>
</tr>
<tr>
<td>Data[13]</td>
<td>ry in a high coordinate</td>
<td>Ry_high</td>
</tr>
<tr>
<td>Data[14]</td>
<td>rz is in a low coordinate</td>
<td>Rz_low</td>
</tr>
<tr>
<td>Data[15]</td>
<td>rz is in a high coordinate</td>
<td>Rz_high</td>
</tr>
<tr>
<td>Data[16]</td>
<td>Is_linear type</td>
<td></td>
</tr>
<tr>
<td>Data[17]</td>
<td>End frame</td>
<td>0xfa</td>
</tr>
</tbody>
</table>

Determine whether the manipulator has reached the specified point

Serial port sending example: FE FE 10 25 FF 74 FE EE 0A C1 DD 05 00 48 DC 95 32 01 FA

x_high: Data type Byte
Calculation: The x coordinate value is multiplied by 10 and then to take the high hexadecimal byte

x_low: Data type Byte
Calculation: The x coordinate value is multiplied by 10 and then to take the low hexadecimal byte
(same as the y/z-coordinate)

rx_high: Data type Byte
Calculation: The rx coordinate value is multiplied by 10 and then to take the high hexadecimal byte

rx_low: Data type Byte
Calculation: The rx coordinate value is multiplied by 10 and then to take the low hexadecimal byte

(same as the ry/rz-coordinate)

Type: Data type Byte (Not yet in use)
Returns a data structure

<table>
<thead>
<tr>
<th>Data field</th>
<th>Description</th>
<th>Data</th>
</tr>
</thead>
<tbody>
<tr>
<td>Data[0]</td>
<td>Return frame</td>
<td>0xFE</td>
</tr>
<tr>
<td>Data[1]</td>
<td>Return frame</td>
<td>0xFE</td>
</tr>
<tr>
<td>Data[2]</td>
<td>Return data length frame</td>
<td>0X03</td>
</tr>
<tr>
<td>Data[3]</td>
<td>Return instruction frame</td>
<td>0X2A</td>
</tr>
<tr>
<td>Data[4]</td>
<td>InPosition/noInPosition</td>
<td>0X01/0X00</td>
</tr>
<tr>
<td>Data[5]</td>
<td>End frame</td>
<td>0xFA</td>
</tr>
</tbody>
</table>

It has reached a point;
Serial port sending example: FE FE 03 2A 00 FA

11). Motion Detection

<table>
<thead>
<tr>
<th>Data field</th>
<th>Description</th>
<th>Data</th>
</tr>
</thead>
<tbody>
<tr>
<td>Data[0]</td>
<td>Identify the frame</td>
<td>0xFE</td>
</tr>
<tr>
<td>Data[1]</td>
<td>Identify the frame</td>
<td>0xFE</td>
</tr>
<tr>
<td>Data[2]</td>
<td>Data length frame</td>
<td>0X02</td>
</tr>
<tr>
<td>Data[3]</td>
<td>Instruction frame</td>
<td>0X2B</td>
</tr>
<tr>
<td>Data[4]</td>
<td>End frame</td>
<td>0xFA</td>
</tr>
</tbody>
</table>

Check whether the robot is moving
Serial port sending example: FE FE 02 2B FA

Returns a data structure
<table>
<thead>
<tr>
<th>Data field</th>
<th>Description</th>
<th>Data</th>
</tr>
</thead>
<tbody>
<tr>
<td>Data[0]</td>
<td>Return frame</td>
<td>0xFE</td>
</tr>
<tr>
<td>Data[1]</td>
<td>Return frame</td>
<td>0xFE</td>
</tr>
<tr>
<td>Data[2]</td>
<td>Return data length frame</td>
<td>0x02</td>
</tr>
<tr>
<td>Data[3]</td>
<td>Return instruction frame</td>
<td>0x2B</td>
</tr>
<tr>
<td>Data[4]</td>
<td>Not running/no data - running</td>
<td>0x00/0x01</td>
</tr>
<tr>
<td>Data[5]</td>
<td>End frame</td>
<td>0xFA</td>
</tr>
</tbody>
</table>

Serial port sending example: FE FE 02 2B 00 FA

12). Jog-Direction Motion

<table>
<thead>
<tr>
<th>Data field</th>
<th>Description</th>
<th>Data</th>
</tr>
</thead>
<tbody>
<tr>
<td>Data[0]</td>
<td>Identify the frame</td>
<td>0xFE</td>
</tr>
<tr>
<td>Data[1]</td>
<td>Identify the frame</td>
<td>0xFE</td>
</tr>
<tr>
<td>Data[2]</td>
<td>Data length frame</td>
<td>0x05</td>
</tr>
<tr>
<td>Data[3]</td>
<td>Instruction frame</td>
<td>0x30</td>
</tr>
<tr>
<td>Data[4]</td>
<td>Joint number</td>
<td>Joint</td>
</tr>
<tr>
<td>Data[5]</td>
<td>Joint direction</td>
<td>Direction</td>
</tr>
<tr>
<td>Data[6]</td>
<td>Specified speed</td>
<td>Sp</td>
</tr>
<tr>
<td>Data[7]</td>
<td>End frame</td>
<td>0xFA</td>
</tr>
</tbody>
</table>

Set Joint 1 to rotate clockwise at 50% speed

Serial port sending example: FE FE 05 30 01 01 32 FA

Joint No. ranges from 1~6

di: Data type Byte, ranges from 0~1

sp: Data type Byte, ranges from 0-100%

No Return Value

13). Jog-Coordinate Motion
### Data field

<table>
<thead>
<tr>
<th>Data field</th>
<th>Description</th>
<th>Data</th>
</tr>
</thead>
<tbody>
<tr>
<td>Data[0]</td>
<td>Identify the frame</td>
<td>0xfe</td>
</tr>
<tr>
<td>Data[1]</td>
<td>Identify the frame</td>
<td>0xfe</td>
</tr>
<tr>
<td>Data[2]</td>
<td>Data length frame</td>
<td>0x05</td>
</tr>
<tr>
<td>Data[3]</td>
<td>Instruction frame</td>
<td>0x33</td>
</tr>
<tr>
<td>Data[4]</td>
<td>Specified coordinates</td>
<td>X1 y2 z3 rx4 ry5 rz6</td>
</tr>
<tr>
<td>Data[5]</td>
<td>Joint direction</td>
<td>Direction</td>
</tr>
<tr>
<td>Data[6]</td>
<td>Specified speed</td>
<td>Sp</td>
</tr>
<tr>
<td>Data[7]</td>
<td>End frame</td>
<td>0xfa</td>
</tr>
</tbody>
</table>

Set the end to move at a speed of 50% counterclockwise toward the X-axis

Serial port sending example: FE FE 05 32 01 00 32 FA

Axis number: Data type Byte (x = 0, y, z, rx, ry, rz) Range from 1~6

di: Data type Byte, ranges from 0~1

sp: Data type Byte, ranges from 0-100%

No Return Value

### 14). jog stop

<table>
<thead>
<tr>
<th>Data field</th>
<th>Description</th>
<th>Data</th>
</tr>
</thead>
<tbody>
<tr>
<td>Data[0]</td>
<td>Identify the frame</td>
<td>0xfe</td>
</tr>
<tr>
<td>Data[1]</td>
<td>Identify the frame</td>
<td>0xfe</td>
</tr>
<tr>
<td>Data[2]</td>
<td>Data length frame</td>
<td>0x02</td>
</tr>
<tr>
<td>Data[3]</td>
<td>Instruction frame</td>
<td>0x34</td>
</tr>
<tr>
<td>Data[4]</td>
<td>End frame</td>
<td>0xfa</td>
</tr>
</tbody>
</table>

Jog stop to move

Serial port sending example: FE FE 02 34 FA

No Return Value

### 15). send potential value
<table>
<thead>
<tr>
<th>Data field</th>
<th>Description</th>
<th>Data</th>
</tr>
</thead>
<tbody>
<tr>
<td>Data[0]</td>
<td>Identify the frame</td>
<td>0xfe</td>
</tr>
<tr>
<td>Data[1]</td>
<td>Identify the frame</td>
<td>0xfe</td>
</tr>
<tr>
<td>Data[2]</td>
<td>Data length frame</td>
<td>0x05</td>
</tr>
<tr>
<td>Data[3]</td>
<td>Instruction frame</td>
<td>0x3a</td>
</tr>
<tr>
<td>Data[4]</td>
<td>Joint number</td>
<td>Joint</td>
</tr>
<tr>
<td>Data[5]</td>
<td>Encoder is in a Low position</td>
<td>Encoder_low</td>
</tr>
<tr>
<td>Data[6]</td>
<td>Encoder is in a High position</td>
<td>Encoder_high</td>
</tr>
<tr>
<td>Data[7]</td>
<td>End frame</td>
<td>0xfa</td>
</tr>
</tbody>
</table>

1. Return data structure

Example, set joint 2 to 2249 potential.

Example of serial port sending: FE FE 05 3A 01 08 C9 FA

Range of joint serial numbers: 0~5

byte Joint: data types
byte Encoder_high: data types

Calculation method: high bit value (hexadecimal) is taken

byte Encoder_low: data types

Calculation method: take potential value (hexadecimal) low position

Return Value: No

16. Gets the potential value;)

213
<table>
<thead>
<tr>
<th>Data domain</th>
<th>Description</th>
<th>Data</th>
</tr>
</thead>
<tbody>
<tr>
<td>Data[0]</td>
<td>Recognition frame</td>
<td>0xFE</td>
</tr>
<tr>
<td>Data[1]</td>
<td>Recognition frame</td>
<td>0xFE</td>
</tr>
<tr>
<td>Data[2]</td>
<td>Data length frame</td>
<td>0X03</td>
</tr>
<tr>
<td>Data[3]</td>
<td>Instruction frame</td>
<td>0X3B</td>
</tr>
<tr>
<td>Data[4]</td>
<td>Joint serial number</td>
<td>joint</td>
</tr>
<tr>
<td>Data[5]</td>
<td>End frame</td>
<td>0xFA</td>
</tr>
</tbody>
</table>

Get the potential value of steering gear 1

Example of serial port sending: FE FE 03 3B 00 FA

Range of joint serial numbers: 1~6

Return data structure

<table>
<thead>
<tr>
<th>Data domain</th>
<th>Description</th>
<th>Data</th>
</tr>
</thead>
<tbody>
<tr>
<td>Data[0]</td>
<td>Return recognition frame</td>
<td>0xFE</td>
</tr>
<tr>
<td>Data[1]</td>
<td>Return recognition frame</td>
<td>0xFE</td>
</tr>
<tr>
<td>Data[2]</td>
<td>Returns data length frames</td>
<td>0X04</td>
</tr>
<tr>
<td>Data[3]</td>
<td>Returns the instruction frame</td>
<td>0X3B</td>
</tr>
<tr>
<td>Data[4]</td>
<td>High value of steering gear potential</td>
<td>Encoder_high</td>
</tr>
<tr>
<td>Data[5]</td>
<td>Low potential of steering gear</td>
<td>Encoders_low</td>
</tr>
<tr>
<td>Data[6]</td>
<td>End frame</td>
<td>0xFA</td>
</tr>
</tbody>
</table>

Example of serial port return: FE FE 04 3B 08 C9 FA

Potential =2249

How to calculate the potential value

Potential value = low potential value + high potential value * 256

1. Transmission of potential values for six steering gear
<table>
<thead>
<tr>
<th>Data domain</th>
<th>Description</th>
<th>Data</th>
</tr>
</thead>
<tbody>
<tr>
<td>Data[0]</td>
<td>Recognition frame</td>
<td>0XFE</td>
</tr>
<tr>
<td>Data[1]</td>
<td>Recognition frame</td>
<td>0XFE</td>
</tr>
<tr>
<td>Data[2]</td>
<td>Data length frame</td>
<td>0X15</td>
</tr>
<tr>
<td>Data[3]</td>
<td>Instruction frame</td>
<td>0X3C</td>
</tr>
<tr>
<td>Data[4]</td>
<td>1 steering gear with high potential</td>
<td>encoder_1_high</td>
</tr>
<tr>
<td>Data[5]</td>
<td>Low byte potential of steering gear 1</td>
<td>encoder_1_low</td>
</tr>
<tr>
<td>Data[6]</td>
<td>2 steering gear with high potential</td>
<td>encoder_2_high</td>
</tr>
<tr>
<td>Data[7]</td>
<td>Low byte potential of steering gear 2</td>
<td>encoder_2_low</td>
</tr>
<tr>
<td>Data[8]</td>
<td>3 steering gear with high potential</td>
<td>encoder_3_high</td>
</tr>
<tr>
<td>Data[9]</td>
<td>Low byte potential of steering gear 3</td>
<td>encoder_3_low</td>
</tr>
<tr>
<td>Data[10]</td>
<td>4 steering gear with high potential bytes</td>
<td>encoder_4_high</td>
</tr>
<tr>
<td>Data[11]</td>
<td>Low byte potential of steering gear 4</td>
<td>encoder_4_low</td>
</tr>
<tr>
<td>Data[12]</td>
<td>5 steering gear with high potential</td>
<td>encoder_5_high</td>
</tr>
<tr>
<td>Data[13]</td>
<td>Low byte potential of steering gear 5</td>
<td>encoder_5_low</td>
</tr>
<tr>
<td>Data[14]</td>
<td>6 steering gear with high potential</td>
<td>encoder_6_high</td>
</tr>
<tr>
<td>Data[15]</td>
<td>Low byte potential of steering gear 6</td>
<td>encoder_6_low</td>
</tr>
<tr>
<td>Data[16]</td>
<td>Specify speed</td>
<td>Sp</td>
</tr>
<tr>
<td>Data[17]</td>
<td>End frame</td>
<td>0XFA</td>
</tr>
</tbody>
</table>

Send potential values for all motors

Example of serial port sending: FE FE 15 3C 00 00 00 00 00 00 00 00 00 00 00 00 00 20FA
(separate potential values sent over reference)

byte encoder_1_high: data types
Calculation method : 1 steering gear potential value converted to int type and then hexadecimal high byte

byte encoder_1_low: data types
Calculation method : 1 steering gear potential value converted to int type and then hexadecimal low byte

(Other Same)

Sp: data type byte range 0~100

Return Value: No
1. Read speed

<table>
<thead>
<tr>
<th>Data domain</th>
<th>Description</th>
<th>Data</th>
</tr>
</thead>
<tbody>
<tr>
<td>Data[0]</td>
<td>Recognition frame</td>
<td>0xFE</td>
</tr>
<tr>
<td>Data[1]</td>
<td>Recognition frame</td>
<td>0xFE</td>
</tr>
<tr>
<td>Data[2]</td>
<td>Data length frame</td>
<td>0x02</td>
</tr>
<tr>
<td>Data[3]</td>
<td>Instruction frame</td>
<td>0x40</td>
</tr>
<tr>
<td>Data[4]</td>
<td>End frame</td>
<td>0xFA</td>
</tr>
</tbody>
</table>

Example of serial port sending: FE FE 02 40 FA

Return data structure

<table>
<thead>
<tr>
<th>Data domain</th>
<th>Description</th>
<th>Data</th>
</tr>
</thead>
<tbody>
<tr>
<td>Data[0]</td>
<td>Return recognition frame</td>
<td>0xFE</td>
</tr>
<tr>
<td>Data[1]</td>
<td>Return recognition frame</td>
<td>0xFE</td>
</tr>
<tr>
<td>Data[2]</td>
<td>Returns data length frames</td>
<td>0x03</td>
</tr>
<tr>
<td>Data[3]</td>
<td>Returns the instruction frame</td>
<td>0x40</td>
</tr>
<tr>
<td>Data[4]</td>
<td>Specify speed</td>
<td>Sp</td>
</tr>
<tr>
<td>Data[5]</td>
<td>End frame</td>
<td>0xFA</td>
</tr>
</tbody>
</table>

Example of serial port return: FE FE 03 40 32 FA

1. Set speed

<table>
<thead>
<tr>
<th>Data domain</th>
<th>Description</th>
<th>Data</th>
</tr>
</thead>
<tbody>
<tr>
<td>Data[0]</td>
<td>Recognition frame</td>
<td>0xFE</td>
</tr>
<tr>
<td>Data[1]</td>
<td>Recognition frame</td>
<td>0xFE</td>
</tr>
<tr>
<td>Data[2]</td>
<td>Data length frame</td>
<td>0x02</td>
</tr>
<tr>
<td>Data[3]</td>
<td>Instruction frame</td>
<td>0x41</td>
</tr>
<tr>
<td>Data[4]</td>
<td>Specify speed</td>
<td>sp</td>
</tr>
<tr>
<td>Data[5]</td>
<td>End frame</td>
<td>0xFA</td>
</tr>
</tbody>
</table>

Example of serial port sending: FE FE 02 41 32 FA

Return Value: No

1. Read FeedOverride (not open yet)
Data domain | Description | Data  
---|---|---
Data[0] | Recognition frame | 0XFE  
Data[1] | Recognition frame | 0XFE  
Data[2] | Data length frame | 0X02  
Data[3] | Instruction frame | 0X42  
Data[4] | End frame | 0XFA  

Return Value: No

1. Read acceleration (not yet open)

| Data domain | Description | Data  
---|---|---
Data[0] | Recognition frame | 0XFE  
Data[1] | Recognition frame | 0XFE  
Data[2] | Data length frame | 0X02  
Data[3] | Instruction frame | 0X44  
Data[4] | End frame | 0XFA  

1. Read the minimum angle of the joint

| Data domain | Description | Data  
---|---|---
Data[0] | Recognition frame | 0XFE  
Data[1] | Recognition frame | 0XFE  
Data[2] | Data length frame | 0X03  
Data[3] | Instruction frame | 0X4A  
Data[4] | Joint steering gear serial number | Joint_number  
Data[5] | End frame | 0XFA  

Read

Example of serial port sending: FE FE 03 4A 00 FA

joint_no range: 0~5

Return data structure
<table>
<thead>
<tr>
<th>Data domain</th>
<th>Description</th>
<th>Data</th>
</tr>
</thead>
<tbody>
<tr>
<td>Data[0]</td>
<td>Return recognition frame</td>
<td>0xFE</td>
</tr>
<tr>
<td>Data[1]</td>
<td>Return recognition frame</td>
<td>0xFE</td>
</tr>
<tr>
<td>Data[2]</td>
<td>Returns data length frames</td>
<td>0x04</td>
</tr>
<tr>
<td>Data[3]</td>
<td>Returns the instruction frame</td>
<td>0x4A</td>
</tr>
<tr>
<td>Data[4]</td>
<td>High angle of steering gear</td>
<td>Angle_high</td>
</tr>
<tr>
<td>Data[5]</td>
<td>Low angle of steering gear</td>
<td>Angle_low</td>
</tr>
<tr>
<td>Data[6]</td>
<td>End frame</td>
<td>0xFA</td>
</tr>
</tbody>
</table>

Example of serial port return: FE FE 04 4A 01 44 FA

Angle =90

How to get the minimum angle

\[ \text{temp} = \text{angle1\_low} + \text{angle1\_high} \times 256 \]

\[ \text{Angle1} = \frac{\text{temp} \div 33000}{\text{temp} - 65536} \times 10 \]

Calculation method: low angle value + high angle value multiplied by 256 to determine whether it is greater than 33000 if it is greater than 33000 then subtract 65536 and divide 10 if directly divided by 10

1. Read the maximum angle of the joint

<table>
<thead>
<tr>
<th>Data domain</th>
<th>Description</th>
<th>Data</th>
</tr>
</thead>
<tbody>
<tr>
<td>Data[0]</td>
<td>Recognition frame</td>
<td>0xFE</td>
</tr>
<tr>
<td>Data[1]</td>
<td>Recognition frame</td>
<td>0xFE</td>
</tr>
<tr>
<td>Data[2]</td>
<td>Data length frame</td>
<td>0x03</td>
</tr>
<tr>
<td>Data[3]</td>
<td>Instruction frame</td>
<td>0x4B</td>
</tr>
<tr>
<td>Data[4]</td>
<td>Joint steering gear serial number</td>
<td>joint_number</td>
</tr>
<tr>
<td>Data[5]</td>
<td>End frame</td>
<td>0xFA</td>
</tr>
</tbody>
</table>

Example of serial port sending: FE FE 03 4B 01 FA

Return data structure
Data domain | Description                  | Data  
---|---|---
Data[0] | Return recognition frame | 0XFE  
Data[1] | Return recognition frame | 0XFE  
Data[2] | Returns data length frames | 0X04  
Data[3] | Returns the instruction frame | 0X4B  
Data[4] | High angle of steering gear | Angle_high  
Data[5] | Low angle of steering gear | Angle_low  
Data[6] | End frame | 0XFA  

Example of serial port return: FE FE 04 4B 01 44 FA  

joint_no range 0~5  

How to get the maximum angle of the joint  

\[ \text{temp} = \text{angle1_low} + \text{angle1_high} \times 256 \]  

\[ \text{Angle1} = \text{temp} \div 33000? (\text{temp} - 65536): \text{temp}/10 \]  

Calculation method: low angle value + high angle value multiplied by 256 to determine whether it is greater than 33000 if it is greater than 33000 then subtract 65536 and divide 10 if directly divided by 10  

1. View connection  

Data domain | Description                  | Data  
---|---|---
Data[0] | Recognition frame | 0XFE  
Data[1] | Recognition frame | 0XFE  
Data[2] | Data length frame | 0X03  
Data[3] | Instruction frame | 0X50  
Data[4] | End frame | 0XFA  

Example of serial port sending: FE FE 03 50 FA  

Return data structure  

Data domain | Description                  | Data  
---|---|---
Data[0] | Return recognition frame | 0XFE  
Data[1] | Return recognition frame | 0XFE  
Data[2] | Returns data length frames | 0X03  
Data[3] | Instruction frame | 0X50  
Data[4] | Connected/unconnected | 0X01/0X00  
Data[5] | End frame | 0XFA  

219
Example of serial port return: FE FE 03 50 00 FA

1. Check to see if the steering gear is all powered up

<table>
<thead>
<tr>
<th>Data domain</th>
<th>Description</th>
<th>Data</th>
</tr>
</thead>
<tbody>
<tr>
<td>Data[0]</td>
<td>Recognition frame</td>
<td>0xFE</td>
</tr>
<tr>
<td>Data[1]</td>
<td>Recognition frame</td>
<td>0xFE</td>
</tr>
<tr>
<td>Data[2]</td>
<td>Data length frame</td>
<td>0x02</td>
</tr>
<tr>
<td>Data[3]</td>
<td>Instruction frame</td>
<td>0x51</td>
</tr>
<tr>
<td>Data[4]</td>
<td>End frame</td>
<td>0xFA</td>
</tr>
</tbody>
</table>

Example of serial port sending: FE FE 02 51 FA

Return data structure

<table>
<thead>
<tr>
<th>Data domain</th>
<th>Description</th>
<th>Data</th>
</tr>
</thead>
<tbody>
<tr>
<td>Data[0]</td>
<td>Return recognition frame</td>
<td>0xFE</td>
</tr>
<tr>
<td>Data[1]</td>
<td>Return recognition frame</td>
<td>0xFE</td>
</tr>
<tr>
<td>Data[2]</td>
<td>Returns data length frames</td>
<td>0x03</td>
</tr>
<tr>
<td>Data[3]</td>
<td>Instruction frame</td>
<td>0x51</td>
</tr>
<tr>
<td>Data[4]</td>
<td>Electricity on/off</td>
<td>0x01/0x00</td>
</tr>
<tr>
<td>Data[5]</td>
<td>End frame</td>
<td>0xFA</td>
</tr>
</tbody>
</table>

Example of serial port return: FE FE 03 51 00 FA

1. Read steering gear status

<table>
<thead>
<tr>
<th>Data domain</th>
<th>Description</th>
<th>Data</th>
</tr>
</thead>
<tbody>
<tr>
<td>Data[0]</td>
<td>Recognition frame</td>
<td>0xFE</td>
</tr>
<tr>
<td>Data[1]</td>
<td>Recognition frame</td>
<td>0xFE</td>
</tr>
<tr>
<td>Data[2]</td>
<td>Data length frame</td>
<td>0x04</td>
</tr>
<tr>
<td>Data[3]</td>
<td>Instruction frame</td>
<td>0x53</td>
</tr>
<tr>
<td>Data[4]</td>
<td>Joint steering gear serial number</td>
<td>joint_number</td>
</tr>
<tr>
<td>Data[5]</td>
<td>Data</td>
<td>data_id</td>
</tr>
<tr>
<td>Data[6]</td>
<td>End frame</td>
<td>0xFA</td>
</tr>
</tbody>
</table>

Read Parameter of position P ratio of steering gear 1

Example of serial port sending: FE FE 04 53 00 21 FA

joint_no range 0~5
byte Data_id: data types

The values are taken in the table below

<table>
<thead>
<tr>
<th>Address</th>
<th>Function</th>
<th>value range</th>
<th>initial value</th>
<th>value analysis</th>
</tr>
</thead>
<tbody>
<tr>
<td>20</td>
<td>LED alarm</td>
<td>0-254</td>
<td>0</td>
<td>10 = open or close LED alarm</td>
</tr>
<tr>
<td>21</td>
<td>Speed loop P</td>
<td>0-254</td>
<td>123joint8,456joint5</td>
<td>control the proportional coefficient of the motor</td>
</tr>
<tr>
<td>22</td>
<td>Position ring I</td>
<td>0-254</td>
<td>123joint20,456joint13</td>
<td>control the differential coefficient of the motor</td>
</tr>
<tr>
<td>23</td>
<td>Position ring D</td>
<td>0-254</td>
<td>0</td>
<td>control the integral coefficient of the motor</td>
</tr>
<tr>
<td>24</td>
<td>minimum starting force</td>
<td>0-1000</td>
<td>0</td>
<td>set the minimum output torque1000 = 100%</td>
</tr>
</tbody>
</table>

Return data structure

<table>
<thead>
<tr>
<th>Data domain</th>
<th>Description</th>
<th>Data</th>
</tr>
</thead>
<tbody>
<tr>
<td>Data[0]</td>
<td>Return recognition frame</td>
<td>0XFE</td>
</tr>
<tr>
<td>Data[1]</td>
<td>Return recognition frame</td>
<td>0XFE</td>
</tr>
<tr>
<td>Data[2]</td>
<td>Returns data length frames</td>
<td>0X03</td>
</tr>
<tr>
<td>Data[3]</td>
<td>Returns the instruction frame</td>
<td>0X53</td>
</tr>
<tr>
<td>Data[4]</td>
<td>Return data</td>
<td>data_state</td>
</tr>
<tr>
<td>Data[5]</td>
<td>End frame</td>
<td>0XFA</td>
</tr>
</tbody>
</table>

Example of serial port return: FE FE 03 53 00 FA

1. Set steering gear zero
### Data domain

<table>
<thead>
<tr>
<th>Data domain</th>
<th>Description</th>
<th>Data</th>
</tr>
</thead>
<tbody>
<tr>
<td>Data[0]</td>
<td>Recognition frame</td>
<td>0xFE</td>
</tr>
<tr>
<td>Data[1]</td>
<td>Recognition frame</td>
<td>0xFE</td>
</tr>
<tr>
<td>Data[2]</td>
<td>Data length frame</td>
<td>0x03</td>
</tr>
<tr>
<td>Data[3]</td>
<td>Instruction frame</td>
<td>0x55</td>
</tr>
<tr>
<td>Data[4]</td>
<td>Joint steering gear serial number</td>
<td>joint_number</td>
</tr>
<tr>
<td>Data[5]</td>
<td>End frame</td>
<td>0xFA</td>
</tr>
</tbody>
</table>

Example of serial port sending: FE FE 03 55 00 FA

Return Value: No

1. Brake single motor (not yet open)

### Data domain

<table>
<thead>
<tr>
<th>Data domain</th>
<th>Description</th>
<th>Data</th>
</tr>
</thead>
<tbody>
<tr>
<td>Data[0]</td>
<td>Recognition frame</td>
<td>0xFE</td>
</tr>
<tr>
<td>Data[1]</td>
<td>Recognition frame</td>
<td>0xFE</td>
</tr>
<tr>
<td>Data[2]</td>
<td>Data length frame</td>
<td>0x03</td>
</tr>
<tr>
<td>Data[3]</td>
<td>Instruction frame</td>
<td>0x56</td>
</tr>
<tr>
<td>Data[4]</td>
<td>Joint steering gear serial number</td>
<td>joint_number</td>
</tr>
<tr>
<td>Data[5]</td>
<td>End frame</td>
<td>0xFA</td>
</tr>
</tbody>
</table>

Example of serial port sending: FE FE 03 56 00 FA

Return Value: No

---

1. Setting atom Pin Mode

### Data domain

<table>
<thead>
<tr>
<th>Data domain</th>
<th>Description</th>
<th>Data</th>
</tr>
</thead>
<tbody>
<tr>
<td>Data[0]</td>
<td>Recognition frame</td>
<td>0xFE</td>
</tr>
<tr>
<td>Data[1]</td>
<td>Recognition frame</td>
<td>0xFE</td>
</tr>
<tr>
<td>Data[2]</td>
<td>Data length frame</td>
<td>0x04</td>
</tr>
<tr>
<td>Data[3]</td>
<td>Instruction frame</td>
<td>0x60</td>
</tr>
<tr>
<td>Data[4]</td>
<td>Number of pins</td>
<td>pin_no</td>
</tr>
<tr>
<td>Data[5]</td>
<td>Pin mode</td>
<td>pin_mode</td>
</tr>
<tr>
<td>Data[6]</td>
<td>End frame</td>
<td>0xFA</td>
</tr>
</tbody>
</table>

atom pin16 set to output mode 0

Example of serial port sending: FE FE 04 60 19 00 FA

byte Pin_no: data types
byte Pin_mode: data types

Return Value: No

1. The program paused

<table>
<thead>
<tr>
<th>Data domain</th>
<th>Description</th>
<th>Data</th>
</tr>
</thead>
<tbody>
<tr>
<td>Data[0]</td>
<td>Recognition frame</td>
<td>0xFE</td>
</tr>
<tr>
<td>Data[1]</td>
<td>Recognition frame</td>
<td>0xFE</td>
</tr>
<tr>
<td>Data[2]</td>
<td>Data length frame</td>
<td>0x02</td>
</tr>
<tr>
<td>Data[3]</td>
<td>Instruction frame</td>
<td>0x26</td>
</tr>
<tr>
<td>Data[4]</td>
<td>End frame</td>
<td>0xFA</td>
</tr>
</tbody>
</table>

Example of serial port sending: FE FE 02 26 FA

Return Value: No

1. The program continues to run

<table>
<thead>
<tr>
<th>Data domain</th>
<th>Description</th>
<th>Data</th>
</tr>
</thead>
<tbody>
<tr>
<td>Data[0]</td>
<td>Recognition frame</td>
<td>0xFE</td>
</tr>
<tr>
<td>Data[1]</td>
<td>Recognition frame</td>
<td>0xFE</td>
</tr>
<tr>
<td>Data[2]</td>
<td>Data length frame</td>
<td>0x02</td>
</tr>
<tr>
<td>Data[3]</td>
<td>Instruction frame</td>
<td>0x27</td>
</tr>
<tr>
<td>Data[6]</td>
<td>End frame</td>
<td>0xFA</td>
</tr>
</tbody>
</table>

Example of serial port sending: FE FE 02 27 FA

Return Value: No

1. The program stops running

<table>
<thead>
<tr>
<th>Data</th>
<th>Description</th>
<th>Data</th>
</tr>
</thead>
<tbody>
<tr>
<td>Data[0]</td>
<td>Recognition frame</td>
<td>0xFE</td>
</tr>
<tr>
<td>Data[1]</td>
<td>Recognition frame</td>
<td>0xFE</td>
</tr>
<tr>
<td>Data[2]</td>
<td>Data length frame</td>
<td>0x02</td>
</tr>
<tr>
<td>Data[3]</td>
<td>Instruction frame</td>
<td>0x28</td>
</tr>
<tr>
<td>Data[4]</td>
<td>End frame</td>
<td>0xFA</td>
</tr>
</tbody>
</table>

Example of serial port sending: FE FE 02 28 FA

Return Value: No
1. Set steering gear status

<table>
<thead>
<tr>
<th>Data domain</th>
<th>Description</th>
<th>Data</th>
</tr>
</thead>
<tbody>
<tr>
<td>Data[0]</td>
<td>Recognition frame</td>
<td>0XFE</td>
</tr>
<tr>
<td>Data[1]</td>
<td>Recognition frame</td>
<td>0XFE</td>
</tr>
<tr>
<td>Data[2]</td>
<td>Data length frame</td>
<td>0X05</td>
</tr>
<tr>
<td>Data[3]</td>
<td>Instruction frame</td>
<td>0X52</td>
</tr>
<tr>
<td>Data[4]</td>
<td>Joint steering gear serial number</td>
<td>servo_no</td>
</tr>
<tr>
<td>Data[5]</td>
<td>Steering gear status</td>
<td>servo_state</td>
</tr>
<tr>
<td>Data[6]</td>
<td>Data</td>
<td>servo_data</td>
</tr>
<tr>
<td>Data[7]</td>
<td>End frame</td>
<td>0XFA</td>
</tr>
</tbody>
</table>

Set position P ratio Parameter 1

Example of serial port sending: FE FE 05 52 00 21 01 FA

Return Value: No


1. Robot free mode (turn off all torque output)

<table>
<thead>
<tr>
<th>Data domain</th>
<th>Description</th>
<th>Data</th>
</tr>
</thead>
<tbody>
<tr>
<td>Data[0]</td>
<td>Recognition frame</td>
<td>0XFE</td>
</tr>
<tr>
<td>Data[1]</td>
<td>Recognition frame</td>
<td>0XFE</td>
</tr>
<tr>
<td>Data[2]</td>
<td>Data length frame</td>
<td>0X02</td>
</tr>
<tr>
<td>Data[3]</td>
<td>Instruction frame</td>
<td>0X13</td>
</tr>
<tr>
<td>Data[4]</td>
<td>End frame</td>
<td>0XFA</td>
</tr>
</tbody>
</table>

Example of serial port sending: FE FE 02 13 FA

Return Value: No

1. Set atom screen RGB light color
<table>
<thead>
<tr>
<th>Data domain</th>
<th>Description</th>
<th>Data</th>
</tr>
</thead>
<tbody>
<tr>
<td>Data[0]</td>
<td>Recognition frame</td>
<td>0xFE</td>
</tr>
<tr>
<td>Data[1]</td>
<td>Recognition frame</td>
<td>0xFE</td>
</tr>
<tr>
<td>Data[2]</td>
<td>Data length frame</td>
<td>0x05</td>
</tr>
<tr>
<td>Data[3]</td>
<td>Instruction frame</td>
<td>0x6A</td>
</tr>
<tr>
<td>Data[4]</td>
<td>R</td>
<td>R</td>
</tr>
<tr>
<td>Data[5]</td>
<td>G</td>
<td>G</td>
</tr>
<tr>
<td>Data[6]</td>
<td>B</td>
<td>B</td>
</tr>
<tr>
<td>Data[7]</td>
<td>End frame</td>
<td>0xFA</td>
</tr>
</tbody>
</table>

RGB set to blue

Example of serial port sending: FE FE 05 6A 00 00 FF FA

Return Value: No

1. Setting Claw Angle

<table>
<thead>
<tr>
<th>Data domain</th>
<th>Description</th>
<th>Data</th>
</tr>
</thead>
<tbody>
<tr>
<td>Data[0]</td>
<td>Recognition frame</td>
<td>0xFE</td>
</tr>
<tr>
<td>Data[1]</td>
<td>Recognition frame</td>
<td>0xFE</td>
</tr>
<tr>
<td>Data[2]</td>
<td>Data length frame</td>
<td>0x03</td>
</tr>
<tr>
<td>Data[3]</td>
<td>Instruction frame</td>
<td>0x66</td>
</tr>
<tr>
<td>Data[4]</td>
<td>Claw Data</td>
<td>Gripper_data</td>
</tr>
<tr>
<td>Data[5]</td>
<td>End frame</td>
<td>0xFA</td>
</tr>
</tbody>
</table>

Example of serial port sending: FE FE 03 66 00 FA

Gripper_data: data type byte range 0-1

Return Value: No

1. Set FeedOverride (not open yet)
<table>
<thead>
<tr>
<th>Data domain</th>
<th>Description</th>
<th>Data</th>
</tr>
</thead>
<tbody>
<tr>
<td>Data[0]</td>
<td>Recognition frame</td>
<td>0xFE</td>
</tr>
<tr>
<td>Data[1]</td>
<td>Recognition frame</td>
<td>0xFE</td>
</tr>
<tr>
<td>Data[2]</td>
<td>Data length frame</td>
<td>0x04</td>
</tr>
<tr>
<td>Data[3]</td>
<td>Instruction frame</td>
<td>0x43</td>
</tr>
<tr>
<td>Data[4]</td>
<td>Feed_OVERRIDE high</td>
<td>Feed_OVERRIDE_high</td>
</tr>
<tr>
<td>Data[5]</td>
<td>Feed_OVERRIDE low</td>
<td>Feed_OVERRIDE_low</td>
</tr>
<tr>
<td>Data[6]</td>
<td>End frame</td>
<td>0xFA</td>
</tr>
</tbody>
</table>

1. Set acceleration (not open yet)

<table>
<thead>
<tr>
<th>Data domain</th>
<th>Description</th>
<th>Data</th>
</tr>
</thead>
<tbody>
<tr>
<td>Data[0]</td>
<td>Recognition frame</td>
<td>0xFE</td>
</tr>
<tr>
<td>Data[1]</td>
<td>Recognition frame</td>
<td>0xFE</td>
</tr>
<tr>
<td>Data[2]</td>
<td>Data length frame</td>
<td>0x04</td>
</tr>
<tr>
<td>Data[3]</td>
<td>Instruction frame</td>
<td>0x45</td>
</tr>
<tr>
<td>Data[4]</td>
<td>High acceleration values</td>
<td>acceleration_high</td>
</tr>
<tr>
<td>Data[5]</td>
<td>Acceleration low</td>
<td>acceleration_low</td>
</tr>
<tr>
<td>Data[6]</td>
<td>End frame</td>
<td>0xFA</td>
</tr>
</tbody>
</table>

byte acceleration_high: data types
Calculation method: the acceleration value multiplied by 10 is converted to int format and then hexadecimal high byte

byte acceleration_low: data types
Calculation method: the acceleration value multiplied by 10 is converted to int format and then hexadecimal low byte

1. Set minimum angle of joint
<table>
<thead>
<tr>
<th>Data domain</th>
<th>Description</th>
<th>Data</th>
</tr>
</thead>
<tbody>
<tr>
<td>Data[0]</td>
<td>Recognition frame</td>
<td>0XFE</td>
</tr>
<tr>
<td>Data[1]</td>
<td>Recognition frame</td>
<td>0XFE</td>
</tr>
<tr>
<td>Data[2]</td>
<td>Data length frame</td>
<td>0X05</td>
</tr>
<tr>
<td>Data[3]</td>
<td>Instruction frame</td>
<td>0X4C</td>
</tr>
<tr>
<td>Data[4]</td>
<td>Joint steering gear serial number</td>
<td>Joint</td>
</tr>
<tr>
<td>Data[5]</td>
<td>High angle</td>
<td>Angle_high</td>
</tr>
<tr>
<td>Data[6]</td>
<td>Low angle</td>
<td>Angle_low</td>
</tr>
<tr>
<td>Data[7]</td>
<td>End frame</td>
<td>0XFA</td>
</tr>
</tbody>
</table>

Set the minimum angle to 90

Example of serial port sending: FE FE 05 4C 00 01 44 FA

Joint range 0~5

byte angle_high: data types

Calculation: angle value multiplied by 10 converted to int form and then hexadecimal high byte

byte angle_low: data types

Calculation: angle value multiplied by 10 converted to int form and then hexadecimal low byte

Return Value: No

1. Set the maximum angle of the joint

Example of serial port sending: FE FE 05 4D 00 01 44 **FA

Example of serial port sending: FE FE 02 12 FA
Return Value: Yes

Joint range 0~5

byte of <<<<<<<<HEAD angle_high: data types

**Calculation method: angle value multiplied by 10 converted to int form and then hexadecimal high byte**

Example of serial port sending: FE FE 02 20 FA

Return Value: Yes

Example of serial port return: FE FE 0E 20 06 E6 EA C4 81 0B BD EA C0 02 B6 FA

byte angle_low: data types

Calculation: angle value multiplied by 10 converted to int form and then hexadecimal low byte

Return Value: No

---

1. Set the Tool Coordinate System

<table>
<thead>
<tr>
<th>Data domain</th>
<th>Description</th>
<th>Data</th>
</tr>
</thead>
<tbody>
<tr>
<td>Data[0]</td>
<td>Recognition frame</td>
<td>0XFE</td>
</tr>
<tr>
<td>Data[1]</td>
<td>Recognition frame</td>
<td>0XFE</td>
</tr>
<tr>
<td>Data[2]</td>
<td>Data length frame</td>
<td>0X14</td>
</tr>
<tr>
<td>Data[3]</td>
<td>Instruction frame</td>
<td>0X81</td>
</tr>
<tr>
<td>Data[4]</td>
<td>X coordinates high byte</td>
<td>x_high</td>
</tr>
<tr>
<td>Data[5]</td>
<td>X coordinates low byte</td>
<td>x_low</td>
</tr>
<tr>
<td>Data[6]</td>
<td>Y coordinates high byte</td>
<td>y_high</td>
</tr>
<tr>
<td>Data[7]</td>
<td>Y coordinates low byte</td>
<td>y_low</td>
</tr>
<tr>
<td>Data[8]</td>
<td>Z coordinates high byte</td>
<td>z_high</td>
</tr>
<tr>
<td>Data[9]</td>
<td>Z coordinates low byte</td>
<td>z_low</td>
</tr>
<tr>
<td>Data[10]</td>
<td>RX coordinates high byte</td>
<td>rx_high</td>
</tr>
<tr>
<td>Data[11]</td>
<td>RX coordinates low byte</td>
<td>rx_low</td>
</tr>
<tr>
<td>Data[12]</td>
<td>RY coordinates high byte</td>
<td>ry_high</td>
</tr>
<tr>
<td>Data[13]</td>
<td>RY coordinates low byte</td>
<td>ry_low</td>
</tr>
<tr>
<td>Data[14]</td>
<td>RZ coordinates high byte</td>
<td>rz_high</td>
</tr>
<tr>
<td>Data[15]</td>
<td>RZ coordinates low byte</td>
<td>rz_low</td>
</tr>
<tr>
<td>Data[16]</td>
<td>End frame</td>
<td>0XFA</td>
</tr>
</tbody>
</table>
Set tool coordinate system (-14,-27,275,-89.5,0.7,-90.7),

Example of serial port sending: FE FE 14 81FF 74FE EE 0A C1DD 05 00 48DC 95FA

byte x_high: data types

Calculation: x coordinates multiplied by 10 and then high bytes in hexadecimal

byte x_low: data types

Calculation: x coordinates multiplied by 10 and then hexadecimal low bytes

(y axis coordinates z axis coordinates are the same)

byte rx_high: data types

Calculation: rx coordinate value multiplied by 100 and then hexadecimal high byte

byte rx_low: data types

Calculation: rx coordinate value multiplied by 100 and then hexadecimal low byte

(ry axis coordinates rz axis coordinates are the same)

Return Value: No

1. Setting the world coordinate system
<table>
<thead>
<tr>
<th>Data domain</th>
<th>Description</th>
<th>Data</th>
</tr>
</thead>
<tbody>
<tr>
<td>Data[0]</td>
<td>Recognition frame</td>
<td>0XFE</td>
</tr>
<tr>
<td>Data[1]</td>
<td>Recognition frame</td>
<td>0XFE</td>
</tr>
<tr>
<td>Data[2]</td>
<td>Data length frame</td>
<td>0X14</td>
</tr>
<tr>
<td>Data[3]</td>
<td>Instruction frame</td>
<td>0X83</td>
</tr>
<tr>
<td>Data[4]</td>
<td>X coordinates high byte</td>
<td>x_high</td>
</tr>
<tr>
<td>Data[5]</td>
<td>X coordinates low byte</td>
<td>x_low</td>
</tr>
<tr>
<td>Data[6]</td>
<td>Y coordinates high byte</td>
<td>y_high</td>
</tr>
<tr>
<td>Data[7]</td>
<td>Y coordinates low byte</td>
<td>y_low</td>
</tr>
<tr>
<td>Data[8]</td>
<td>Z coordinates high byte</td>
<td>z_high</td>
</tr>
<tr>
<td>Data[9]</td>
<td>Z coordinates low byte</td>
<td>z_low</td>
</tr>
<tr>
<td>Data[10]</td>
<td>RX coordinates high byte</td>
<td>rx_high</td>
</tr>
<tr>
<td>Data[11]</td>
<td>RX coordinates low byte</td>
<td>rx_low</td>
</tr>
<tr>
<td>Data[12]</td>
<td>RY coordinates high byte</td>
<td>ry_high</td>
</tr>
<tr>
<td>Data[13]</td>
<td>RY coordinates low byte</td>
<td>ry_low</td>
</tr>
<tr>
<td>Data[14]</td>
<td>RZ coordinates high byte</td>
<td>rz_high</td>
</tr>
<tr>
<td>Data[15]</td>
<td>RZ coordinates low byte</td>
<td>rz_low</td>
</tr>
<tr>
<td>Data[16]</td>
<td>End frame</td>
<td>0XFA</td>
</tr>
</tbody>
</table>

Set the world coordinate system (-14,-27,275,-89.5,0.7,-90.7),

Example of serial port sending: FE FE 14 83 FF FE EE 0A C1 DD 05 00 48 DC 95 FA

byte x_high: data types
Calculation: x coordinates multiplied by 10 and then high bytes in hexadecimal

byte x_low: data types
Calculation: x coordinates multiplied by 10 and then hexadecimal low bytes

(y axis coordinates z axis coordinates are the same)

byte rx_high: data types
Calculation: rx coordinate value multiplied by 100 and then hexadecimal high byte

<<<<<<<<<HEAD
<<<<<<<<< HEAD

byte rx_low: data types
Example of serial port sending: FE FE 05 30 01 01 32 FA Range of joint serial numbers 1~6 di: data type byte value range 0 and 1 sp: data type range 0-100%

no Return Value

Calculation: rx coordinate value multiplied by 100 and then hexadecimal low byte

(ry axis coordinates rz axis coordinates are the same)

Return Value: No

1. Gets the tool coordinate

<table>
<thead>
<tr>
<th>Data domain</th>
<th>Description</th>
<th>Data</th>
</tr>
</thead>
<tbody>
<tr>
<td>Data[0]</td>
<td>Recognition frame</td>
<td>0xFE</td>
</tr>
<tr>
<td>Data[1]</td>
<td>Recognition frame</td>
<td>0xFE</td>
</tr>
<tr>
<td>Data[2]</td>
<td>Data length frame</td>
<td>0x02</td>
</tr>
<tr>
<td>Data[3]</td>
<td>Instruction frame</td>
<td>0x82</td>
</tr>
<tr>
<td>Data[4]</td>
<td>End frame</td>
<td>0xFA</td>
</tr>
</tbody>
</table>

Example of serial port sending: FE FE 02 82 FA

Return data structure
<table>
<thead>
<tr>
<th>Data domain</th>
<th>Description</th>
<th>Data</th>
</tr>
</thead>
<tbody>
<tr>
<td>Data[0]</td>
<td>Return recognition frame</td>
<td>0XFE</td>
</tr>
<tr>
<td>Data[1]</td>
<td>Return recognition frame</td>
<td>0XFE</td>
</tr>
<tr>
<td>Data[2]</td>
<td>Returns data length frames</td>
<td>0X14</td>
</tr>
<tr>
<td>Data[3]</td>
<td>Returns the instruction frame</td>
<td>0X82</td>
</tr>
<tr>
<td>Data[4]</td>
<td>X coordinates high byte</td>
<td>x_high</td>
</tr>
<tr>
<td>Data[5]</td>
<td>X coordinates low byte</td>
<td>x_low</td>
</tr>
<tr>
<td>Data[6]</td>
<td>Y coordinates high byte</td>
<td>y_high</td>
</tr>
<tr>
<td>Data[7]</td>
<td>Y coordinates low byte</td>
<td>y_low</td>
</tr>
<tr>
<td>Data[8]</td>
<td>Z coordinates high byte</td>
<td>z_high</td>
</tr>
<tr>
<td>Data[9]</td>
<td>Z coordinates low byte</td>
<td>z_low</td>
</tr>
<tr>
<td>Data[10]</td>
<td>RX coordinates high byte</td>
<td>rx_high</td>
</tr>
<tr>
<td>Data[11]</td>
<td>RX coordinates low byte</td>
<td>rx_low</td>
</tr>
<tr>
<td>Data[12]</td>
<td>RY coordinates high byte</td>
<td>ry_high</td>
</tr>
<tr>
<td>Data[13]</td>
<td>RY coordinates low byte</td>
<td>ry_low</td>
</tr>
<tr>
<td>Data[14]</td>
<td>RZ coordinates high byte</td>
<td>rz_high</td>
</tr>
<tr>
<td>Data[15]</td>
<td>RZ coordinates low byte</td>
<td>rz_low</td>
</tr>
<tr>
<td>Data[16]</td>
<td>End frame</td>
<td>0XFA</td>
</tr>
</tbody>
</table>

Example of serial port return: FE FE 14 82 FF 74 EE 0A C1DD 05 00 48 DC 95 FA

byte x_high: data types

Mode of calculation: x coordinates multiplied by 10 are converted to int type and then hexadecimal high bytes

byte x_low: data types

Mode of calculation: x coordinates multiplied by 10 are converted to int type and then hexadecimal low bytes

(y axis coordinates z axis coordinates are the same)

byte rx_high: data types

Mode of calculation: rx coordinates multiplied by 100 are converted to int type and then hexadecimal high bytes

byte rx_low: data types

Mode of calculation: rx coordinates multiplied by 100 are converted to int type and then hexadecimal low bytes

(ry axis coordinates rz axis coordinates are the same)
1. Get the world coordinate system

<table>
<thead>
<tr>
<th>Data domain</th>
<th>Description</th>
<th>Data</th>
</tr>
</thead>
<tbody>
<tr>
<td>Data[0]</td>
<td>Recognition frame</td>
<td>0xFE</td>
</tr>
<tr>
<td>Data[1]</td>
<td>Recognition frame</td>
<td>0xFE</td>
</tr>
<tr>
<td>Data[2]</td>
<td>Data length frame</td>
<td>0x02</td>
</tr>
<tr>
<td>Data[3]</td>
<td>Instruction frame</td>
<td>0x84</td>
</tr>
<tr>
<td>Data[4]</td>
<td>End frame</td>
<td>0xFA</td>
</tr>
</tbody>
</table>

Example of serial port sending: FE FE 02 84 FA

Return data structure

<table>
<thead>
<tr>
<th>Data domain</th>
<th>Description</th>
<th>Data</th>
</tr>
</thead>
<tbody>
<tr>
<td>Data[0]</td>
<td>Return recognition frame</td>
<td>0xFE</td>
</tr>
<tr>
<td>Data[1]</td>
<td>Return recognition frame</td>
<td>0xFE</td>
</tr>
<tr>
<td>Data[2]</td>
<td>Returns data length frames</td>
<td>0x14</td>
</tr>
<tr>
<td>Data[3]</td>
<td>Returns the instruction frame</td>
<td>0x84</td>
</tr>
<tr>
<td>Data[4]</td>
<td>X coordinates high byte</td>
<td>x_high</td>
</tr>
<tr>
<td>Data[5]</td>
<td>X coordinates low byte</td>
<td>x_low</td>
</tr>
<tr>
<td>Data[6]</td>
<td>Y coordinates high byte</td>
<td>y_high</td>
</tr>
<tr>
<td>Data[7]</td>
<td>Y coordinates low byte</td>
<td>y_low</td>
</tr>
<tr>
<td>Data[8]</td>
<td>Z coordinates high byte</td>
<td>z_high</td>
</tr>
<tr>
<td>Data[9]</td>
<td>Z coordinates low byte</td>
<td>z_low</td>
</tr>
<tr>
<td>Data[10]</td>
<td>RX coordinates high byte</td>
<td>rx_high</td>
</tr>
<tr>
<td>Data[11]</td>
<td>RX coordinates low byte</td>
<td>rx_low</td>
</tr>
<tr>
<td>Data[12]</td>
<td>RY coordinates high byte</td>
<td>ry_high</td>
</tr>
<tr>
<td>Data[13]</td>
<td>RY coordinates low byte</td>
<td>ry_low</td>
</tr>
<tr>
<td>Data[14]</td>
<td>RZ coordinates high byte</td>
<td>rz_high</td>
</tr>
<tr>
<td>Data[15]</td>
<td>RZ coordinates low byte</td>
<td>rz_low</td>
</tr>
<tr>
<td>Data[16]</td>
<td>End frame</td>
<td>0xFA</td>
</tr>
</tbody>
</table>

Example of serial port return: FE FE 14 84 FF 74 FE EE 0A C1 DD 05 00 48 DC 95 FA

1. Set up flange base coordinate system
**Example of serial port sending:** FE FE 03 85 00FA

**byte RFType:** data types

Value range: 0~1 BASE =0; WORLD BASE =1;

The RFType: BASE is to take the robot base as the base coordinate, RFType: WORLD the world coordinate system as the base coordinate.

---

1. Get the flange base coordinate system

<table>
<thead>
<tr>
<th>Data domain</th>
<th>Description</th>
<th>Data</th>
</tr>
</thead>
<tbody>
<tr>
<td>Data[0]</td>
<td>Recognition frame</td>
<td>0xFE</td>
</tr>
<tr>
<td>Data[1]</td>
<td>Recognition frame</td>
<td>0xFE</td>
</tr>
<tr>
<td>Data[2]</td>
<td>Data length frame</td>
<td>0x02</td>
</tr>
<tr>
<td>Data[3]</td>
<td>Instruction frame</td>
<td>0x86</td>
</tr>
<tr>
<td>Data[4]</td>
<td>RFType</td>
<td>0x00/0x01</td>
</tr>
<tr>
<td>Data[5]</td>
<td>End frame</td>
<td>0xFA</td>
</tr>
</tbody>
</table>

**Example of serial port sending:** FE FE 02 86 FA

**Return data structure**

<table>
<thead>
<tr>
<th>Data domain</th>
<th>Description</th>
<th>Data</th>
</tr>
</thead>
<tbody>
<tr>
<td>Data[0]</td>
<td>Return recognition frame</td>
<td>0xFE</td>
</tr>
<tr>
<td>Data[1]</td>
<td>Return recognition frame</td>
<td>0xFE</td>
</tr>
<tr>
<td>Data[2]</td>
<td>Returns data length frames</td>
<td>0x03</td>
</tr>
<tr>
<td>Data[3]</td>
<td>Returns the instruction frame</td>
<td>0x86</td>
</tr>
<tr>
<td>Data[4]</td>
<td>RFType</td>
<td>0x00/0x01</td>
</tr>
<tr>
<td>Data[5]</td>
<td>End frame</td>
<td>0xFA</td>
</tr>
</tbody>
</table>

**Example of serial port return:** FE FE 03 86 00 FA

---

1. Set the terminal coordinate system
<table>
<thead>
<tr>
<th>Data domain</th>
<th>Description</th>
<th>Data</th>
</tr>
</thead>
<tbody>
<tr>
<td>Data[0]</td>
<td>Return recognition frame</td>
<td>0XFE</td>
</tr>
<tr>
<td>Data[1]</td>
<td>Return recognition frame</td>
<td>0XFE</td>
</tr>
<tr>
<td>Data[2]</td>
<td>Returns data length frames</td>
<td>0X03</td>
</tr>
<tr>
<td>Data[3]</td>
<td>Returns the instruction frame</td>
<td>0X89</td>
</tr>
<tr>
<td>Data[4]</td>
<td>EndType</td>
<td>0x00/0x01</td>
</tr>
<tr>
<td>Data[5]</td>
<td>End frame</td>
<td>0XFA</td>
</tr>
</tbody>
</table>

Example of serial port sending: FE FE 03 89 00 FA

Return Value: No

byte EndType: data types

Value range: 0~1 FLANGE =0; TOOL FLANGE =1;

EndType: FLANGE to set the end to flange, EndType: TOOL to set the end to tool end.

1. Get the terminal coordinate system

<table>
<thead>
<tr>
<th>Data domain</th>
<th>Description</th>
<th>Data</th>
</tr>
</thead>
<tbody>
<tr>
<td>Data[0]</td>
<td>Recognition frame</td>
<td>0XFE</td>
</tr>
<tr>
<td>Data[1]</td>
<td>Recognition frame</td>
<td>0XFE</td>
</tr>
<tr>
<td>Data[2]</td>
<td>Data length frame</td>
<td>0X02</td>
</tr>
<tr>
<td>Data[3]</td>
<td>Instruction frame</td>
<td>0X8A</td>
</tr>
<tr>
<td>Data[4]</td>
<td>End frame</td>
<td>0XFA</td>
</tr>
</tbody>
</table>

Example of serial port sending: FE FE 02 8A FA

Return data structure

<table>
<thead>
<tr>
<th>Data domain</th>
<th>Description</th>
<th>Data</th>
</tr>
</thead>
<tbody>
<tr>
<td>Data[0]</td>
<td>Return recognition frame</td>
<td>0XFE</td>
</tr>
<tr>
<td>Data[1]</td>
<td>Return recognition frame</td>
<td>0XFE</td>
</tr>
<tr>
<td>Data[2]</td>
<td>Returns data length frames</td>
<td>0X03</td>
</tr>
<tr>
<td>Data[3]</td>
<td>Returns the instruction frame</td>
<td>0X8A</td>
</tr>
<tr>
<td>Data[4]</td>
<td>EndType</td>
<td>0x00/0x01</td>
</tr>
<tr>
<td>Data[5]</td>
<td>End frame</td>
<td>0XFA</td>
</tr>
</tbody>
</table>

Example of serial port return: FE FE 03 8A 00 FA

1. Single motor shutdown
**Data domain** | **Description** | **Data**
---|---|---
Data[0] | Recognition frame | 0xFE
Data[1] | Recognition frame | 0xFE
Data[2] | Data length frame | 0x03
Data[3] | Instruction frame | 0x56
Data[4] | Steering gear serial number | Servo_no
Data[5] | End frame | 0xFA

Turn down the first steering gear

Example of serial port sending: FE FE 03 56 01 FA

Return Value: No

1. Single motor powered on

**Data domain** | **Description** | **Data**
---|---|---
Data[0] | Recognition frame | 0xFE
Data[1] | Recognition frame | 0xFE
Data[2] | Data length frame | 0x02
Data[3] | Instruction frame | 0x57
Data[4] | Steering gear serial number | Servo_no
Data[5] | End frame | 0xFA

Power the steering gear one

Example of serial port sending: FE FE 03 57 01 FA

Return Value: No

1. Setting Atom IO port level state

**Data domain** | **Description** | **Data**
---|---|---
Data[0] | Recognition frame | 0xFE
Data[1] | Recognition frame | 0xFE
Data[2] | Data length frame | 0x04
Data[3] | Instruction frame | 0x61
Data[4] | Number of pins | Pin_no
Data[5] | Level signal | 0x00/0x01
Data[6] | End frame | 0xFA

set pin P22 to high level
Example of serial port sending: FE FE 04 61 22 01 FA

Return Value: No

1. Read Atom IO port level state

<table>
<thead>
<tr>
<th>Data domain</th>
<th>Description</th>
<th>Data</th>
</tr>
</thead>
<tbody>
<tr>
<td>Data[0]</td>
<td>Recognition frame</td>
<td>0XFE</td>
</tr>
<tr>
<td>Data[1]</td>
<td>Recognition frame</td>
<td>0XFE</td>
</tr>
<tr>
<td>Data[2]</td>
<td>Data length frame</td>
<td>0X03</td>
</tr>
<tr>
<td>Data[3]</td>
<td>Instruction frame</td>
<td>0X62</td>
</tr>
<tr>
<td>Data[4]</td>
<td>Number of pins</td>
<td>pin_no</td>
</tr>
<tr>
<td>Data[5]</td>
<td>End frame</td>
<td>0XFA</td>
</tr>
</tbody>
</table>

Read pin P22 level state

Example of serial port sending: FE FE 03 62 22 FA

Return data structure

<table>
<thead>
<tr>
<th>Data domain</th>
<th>Description</th>
<th>Data</th>
</tr>
</thead>
<tbody>
<tr>
<td>Data[0]</td>
<td>Return recognition frame</td>
<td>0XFE</td>
</tr>
<tr>
<td>Data[1]</td>
<td>Return recognition frame</td>
<td>0XFE</td>
</tr>
<tr>
<td>Data[2]</td>
<td>Returns data length frames</td>
<td>0X03</td>
</tr>
<tr>
<td>Data[3]</td>
<td>Returns the instruction frame</td>
<td>0X62</td>
</tr>
<tr>
<td>Data[4]</td>
<td>Level state</td>
<td>0X00/0X01</td>
</tr>
<tr>
<td>Data[5]</td>
<td>End frame</td>
<td>0XFA</td>
</tr>
</tbody>
</table>

1. Set Atom pin PWM mode

<table>
<thead>
<tr>
<th>Data domain</th>
<th>Description</th>
<th>Data</th>
</tr>
</thead>
<tbody>
<tr>
<td>Data[0]</td>
<td>Recognition frame</td>
<td>0XFE</td>
</tr>
<tr>
<td>Data[1]</td>
<td>Recognition frame</td>
<td>0XFE</td>
</tr>
<tr>
<td>Data[2]</td>
<td>Data length frame</td>
<td>0X04</td>
</tr>
<tr>
<td>Data[3]</td>
<td>Instruction frame</td>
<td>0X63</td>
</tr>
<tr>
<td>Data[4]</td>
<td>Number of pins</td>
<td>pin_no</td>
</tr>
<tr>
<td>Data[5]</td>
<td>Channels</td>
<td>channel</td>
</tr>
<tr>
<td>Data[6]</td>
<td>End frame</td>
<td>0XFA</td>
</tr>
</tbody>
</table>

Example of serial port sending: FE FE 04 63 22 01 FA
Return Value: No

1. Set Atom pin PWM output

<table>
<thead>
<tr>
<th>Data domain</th>
<th>Description</th>
<th>Data</th>
</tr>
</thead>
<tbody>
<tr>
<td>Data[0]</td>
<td>Recognition frame</td>
<td>0xFE</td>
</tr>
<tr>
<td>Data[1]</td>
<td>Recognition frame</td>
<td>0xFE</td>
</tr>
<tr>
<td>Data[2]</td>
<td>Data length frame</td>
<td>0x04</td>
</tr>
<tr>
<td>Data[3]</td>
<td>Instruction frame</td>
<td>0x64</td>
</tr>
<tr>
<td>Data[4]</td>
<td>Channels</td>
<td>channel</td>
</tr>
<tr>
<td>Data[5]</td>
<td>Duty cycle</td>
<td>Pin_val</td>
</tr>
<tr>
<td>Data[6]</td>
<td>End frame</td>
<td>0xFA</td>
</tr>
</tbody>
</table>

Example of serial port sending: FE FE 04 64 01 20 FA

Return Value: No

1. Reading Claw Angle

<table>
<thead>
<tr>
<th>Data domain</th>
<th>Description</th>
<th>Data</th>
</tr>
</thead>
<tbody>
<tr>
<td>Data[0]</td>
<td>Recognition frame</td>
<td>0xFE</td>
</tr>
<tr>
<td>Data[1]</td>
<td>Recognition frame</td>
<td>0xFE</td>
</tr>
<tr>
<td>Data[2]</td>
<td>Data length frame</td>
<td>0x02</td>
</tr>
<tr>
<td>Data[3]</td>
<td>Instruction frame</td>
<td>0x65</td>
</tr>
<tr>
<td>Data[4]</td>
<td>End frame</td>
<td>0xFA</td>
</tr>
</tbody>
</table>

Example of serial port sending: FE FE 02 65 FA

Return data structure

<table>
<thead>
<tr>
<th>Data domain</th>
<th>Description</th>
<th>Data</th>
</tr>
</thead>
<tbody>
<tr>
<td>Data[0]</td>
<td>Return recognition frame</td>
<td>0xFE</td>
</tr>
<tr>
<td>Data[1]</td>
<td>Return recognition frame</td>
<td>0xFE</td>
</tr>
<tr>
<td>Data[2]</td>
<td>Returns data length frames</td>
<td>0x04</td>
</tr>
<tr>
<td>Data[3]</td>
<td>Instruction frame</td>
<td>0x65</td>
</tr>
<tr>
<td>Data[4]</td>
<td>High angle</td>
<td>Value_high</td>
</tr>
<tr>
<td>Data[5]</td>
<td>Low angle</td>
<td>Value_low</td>
</tr>
<tr>
<td>Data[6]</td>
<td>End frame</td>
<td>0xFA</td>
</tr>
</tbody>
</table>

1. Set Claw Mode
<table>
<thead>
<tr>
<th>Data domain</th>
<th>Description</th>
<th>Data</th>
</tr>
</thead>
<tbody>
<tr>
<td>Data[0]</td>
<td>Recognition frame</td>
<td>0XFE</td>
</tr>
<tr>
<td>Data[1]</td>
<td>Recognition frame</td>
<td>0XFE</td>
</tr>
<tr>
<td>Data[2]</td>
<td>Data length frame</td>
<td>0X04</td>
</tr>
<tr>
<td>Data[3]</td>
<td>Instruction frame</td>
<td>0X66</td>
</tr>
<tr>
<td>Data[4]</td>
<td>Claw opening and closing</td>
<td>0X00/0X01</td>
</tr>
<tr>
<td>Data[5]</td>
<td>Speed</td>
<td>Sp</td>
</tr>
<tr>
<td>Data[6]</td>
<td>End frame</td>
<td>0XFA</td>
</tr>
</tbody>
</table>

Set the claw to open at 20

Example of serial port sending: FE FE 04 66 00 20 FA

Return Value: No

1. Set Claw Angle

<table>
<thead>
<tr>
<th>Data domain</th>
<th>Description</th>
<th>Data</th>
</tr>
</thead>
<tbody>
<tr>
<td>Data[0]</td>
<td>Recognition frame</td>
<td>0XFE</td>
</tr>
<tr>
<td>Data[1]</td>
<td>Recognition frame</td>
<td>0XFE</td>
</tr>
<tr>
<td>Data[2]</td>
<td>Data length frame</td>
<td>0X05</td>
</tr>
<tr>
<td>Data[3]</td>
<td>Instruction frame</td>
<td>0X67</td>
</tr>
<tr>
<td>Data[4]</td>
<td>High angle</td>
<td>Angle_high</td>
</tr>
<tr>
<td>Data[5]</td>
<td>Low angle</td>
<td>Angle_low</td>
</tr>
<tr>
<td>Data[6]</td>
<td>Speed</td>
<td>Sp</td>
</tr>
<tr>
<td>Data[7]</td>
<td>End frame</td>
<td>0XFA</td>
</tr>
</tbody>
</table>

Sample serial port sending:

Return Value: No

1. Claw setting zero

<table>
<thead>
<tr>
<th>Data domain</th>
<th>Description</th>
<th>Data</th>
</tr>
</thead>
<tbody>
<tr>
<td>Data[0]</td>
<td>Recognition frame</td>
<td>0XFE</td>
</tr>
<tr>
<td>Data[1]</td>
<td>Recognition frame</td>
<td>0XFE</td>
</tr>
<tr>
<td>Data[2]</td>
<td>Data length frame</td>
<td>0X02</td>
</tr>
<tr>
<td>Data[3]</td>
<td>Instruction frame</td>
<td>0X68</td>
</tr>
<tr>
<td>Data[4]</td>
<td>End frame</td>
<td>0XFA</td>
</tr>
</tbody>
</table>
Set the current position of the claw to zero

Example of serial port sending: FE FE 02 68 FA

1. Check the movement of the claw

<table>
<thead>
<tr>
<th>Data domain</th>
<th>Description</th>
<th>Data</th>
</tr>
</thead>
<tbody>
<tr>
<td>Data[0]</td>
<td>Recognition frame</td>
<td>0xFE</td>
</tr>
<tr>
<td>Data[1]</td>
<td>Recognition frame</td>
<td>0xFE</td>
</tr>
<tr>
<td>Data[2]</td>
<td>Data length frame</td>
<td>0x02</td>
</tr>
<tr>
<td>Data[3]</td>
<td>Instruction frame</td>
<td>0x69</td>
</tr>
<tr>
<td>Data[4]</td>
<td>End frame</td>
<td>0xFA</td>
</tr>
</tbody>
</table>

Set the current position of the claw to zero

Example of serial port sending: FE FE 02 69 FA

Return data structure

<table>
<thead>
<tr>
<th>Data domain</th>
<th>Description</th>
<th>Data</th>
</tr>
</thead>
<tbody>
<tr>
<td>Data[0]</td>
<td>Return recognition frame</td>
<td>0xFE</td>
</tr>
<tr>
<td>Data[1]</td>
<td>Return recognition frame</td>
<td>0xFE</td>
</tr>
<tr>
<td>Data[2]</td>
<td>Returns data length frames</td>
<td>0x03</td>
</tr>
<tr>
<td>Data[3]</td>
<td>Instruction frame</td>
<td>0x69</td>
</tr>
<tr>
<td>Data[4]</td>
<td></td>
<td>0x00/0x01</td>
</tr>
<tr>
<td>Data[5]</td>
<td>End frame</td>
<td>0xFA</td>
</tr>
</tbody>
</table>

Appendix:

A coordinate transformation program is added to the ATOM library and the kinematics library, which is implemented as follows:

1. change the terminal coordinate system
2. can set the end coordinate system by setEndType and getEndType functions,
   EndType: : FLANGE to set the end to flange, EndType: : TOOL to set the end to tool end.
3. can set the coordinate information of the reading tool by setToolReference and getToolReference functions. The flange coordinate system is set as the relative coordinate system, and the tool end information is relative to the flange coordinate system.
4. set the EndType to FLANGE, both the GetCoords and the WriteCoords methods are calculated by the flange position.
5. set the EndType to TOOL, both the GetCoords and the WriteCoords methods are calculated at the end position of the tool.
6. Change the base coordinate system

7. base coordinate system can be set by setReferenceFrame function,
   \[ \text{RFType} : \BASE \text{ the robot base as the base coordinate and the world}
   \] coordinate system as the base coordinate. getReferenceFrame function is to
   read the current base coordinate system type.

8. read base coordinate system information can be set by setWorldReference
    and getWorldReference functions. When set, the world coordinate system is
    used as the relative coordinate system, and the position information of the
    base of the robot relative to the world coordinate system is input.

9. when the base coordinate system is the base, the GetCoords and
    WriteCoords methods take the base as the reference coordinate system.

10. When the base coordinate system is the world coordinate system, both the
    GetCoords and WriteCoords methods use the world coordinate system as
    the reference coordinate system.

Communications related changes (temporary)

The setting and reading of the terminal coordinate system, the setting and reading
of the world coordinate system, the setting and reading of the current reference
coordinate system, the setting and reading of the terminal type, the setting and
reading of the moving mode, and the sending and receiving of the manipulator
information are added.

These communications are temporarily set to 0x80 to 0x8A

The new roboticMessages space in the ParameterList.h file is used to add the
manipulator communication information.

MOVEL function simple design idea is as follows:

The Euclidean distance between the initial point and the target point is obtained,
and an interpolation point is inserted every 10 mm based on the Euclidean
distance. If the interpolation point has no inverse solution, search position
invariant three directions attitude positive and negative PI/30 adjacent space
whether there are inverse solutions, mainly to avoid singular values and some
special positions that can not be solved.

The point transmission interval between MOVEL and JOG is changed to dynamic
time. The moving time is calculated according to the maximum joint moving
distance between two points, and then the moving time minus the specific time is
taken as the time interval.
7 Accessories

myCobot accessories includes

**End Effectors**
- Parallel Gripper
- Adaptive Gripper
- Opening Angle Gripper
- Suction Pump

**Bases**
- G Base
- Flat Base

**Accessories**
- JoyStick
- Battery Box
7.1 End Effectors

End effectors of myCobot now in mass production

- **Parallel Gripper:** Use of Clamping Objects
- **Suction Pump:** Use of adsorption objects
7.1.1 Gripper

Gripper

- 1. fix the gripper to the top of the robotic arm with a lego tech and connect to the M5STACK Basic end-effector extension interface (described in the unboxing video)
- 2. grippers can be used in all development environments, such as ROS, Arduino, UIFlow and RoboFlow.

Applicable objects

- small box
- small ball
- long strip

Tips you can paste rubber at the fingertips of the gripper for better friction.
API control you can control gripper by downloading the latest myCobot API.
1. Control the opening and closing degree of the claw: please read the claw position first, then set the range. The value range is 0-4096; the actual range is near 2000.

2. Set the initial point of the claw: the initial point corresponds to a closing degree of 2048.

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

3. Read gripper position

   getGripperValue() // return gripper encoder value (from 0 to 4096)

4. Set gripper state: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 Suction pump

Applicable object

- Paper/plastics
- Smooth planar object
- Cards, etc

Product Description

Installation schematic
Notes

- Please ensure that the product is connected successfully according to the instructions
- Make sure the product is powered by attached adapter
- Please ensure the access direction of positive and negative electrodes

Use of suction pump in Arduino

1. connect the suction pin to the Basic pin
   - solenoid valve control pin link Basic -G2 pin
   - Pump control pin link Basic -G5 pin
2. create a new Arduino program to copy the following:
set pin 2 to high level and close solenoid valve
// please confirm solenoid valve connection G2 pin, pump link G5 pin
// connection completed, high level off, low level open

void setup()
{
  // put your setup code here, to run once:
  Serial.begin(9600); // open serial port, baud rate 9600
  pinMode(2, OUTPUT); // set pin G2 to output state
  pinMode(5, OUTPUT); // set pin G5 to output state
  delay(100);
  digitalWrite(2, 1); // set pin 2 to high level and close solenoid valve
  digitalWrite(5, 1); // set pin 5 to high level and shut down pump
}

void loop()
{
  digitalWrite(5, 0); // set pin 5 as low level and open the pump
  delay(200); // delay 200ms
  digitalWrite(2, 0); // set pin 2 as low level and open the solenoid valve
  delay(2000); // delay 2000ms, releasing the object
  digitalWrite(2, 1); // set pin 2 as high level and close the solenoid valve
  delay(200); // delay 200ms
  digitalWrite(5, 1); // set pin 5 as high level and close the pump
  delay(200); // delay 200ms
}

Use of suction pumps in Uiflow

1. connect the suction pin to the Baisc pin
   - Solenoid valve control pin link Basic -G2 pin
   - Pump control pin link Basic -G5 pin
2. Create a new Uiflow program to add the following content:
7.1.3 Pen holder

Is being developed and written...ss
7.2 Base

myCobot current support base:

- **G Base**: Conveniently secured to table
- **Sucking Base**: Easy to fix directly on smooth table
7.2.1 G Base

G Base - fixed to the edge of the table

- 1. Tighten the horn screws at both ends and insert a rubber sleeve into the he
- 2. Fix the base on the edge of the table with a G clip
- 3. The base and the bottom of the arm with the attached Lego tech
- 4. Make sure it’s stable before use
7.2.2 Sucking Base

Suitable for smooth surfaces

- 1. Install the sucker at the four corners of the base and tighten them
- 2. Use the attached Lego piece to connect the sucking base and the bottom of the manipulator
- 3. Fix the four suckers to a smooth and flat surface before use

Tips

A small amount of non-conductive liquids can be added under the sucker to fill the gap between the sucker and the desktop to obtain the best adsorption effect.
7.3 Accessories

Is being developed and written...
8 Machine Vision Development

What is image recognition?

1. **principle of image recognition**
   - Image recognition refers to the technology of using computers to process, analyze and understand images in order to identify targets and objects of various modes. It is a practical application of deep learning algorithms.

2. **Application scenarios of image recognition**
   - At the present stage, image recognition technology is generally divided into face recognition and commodity recognition. Face recognition is mainly used in security check, identity verification and mobile payment. Product identification is mainly used in the process of commodity circulation, especially in unmanned retail areas such as unmanned shelves and intelligent retail cabinets.

3. **Artificial intelligence application of image recognition**
   - Image recognition is an important field of artificial intelligence. Different image recognition models have been proposed in order to make computer programs simulating human image recognition activities. An example is the template matching model. This model holds that to recognize an image, one must have a memory pattern of the image in the past experience, also known as the template. If the current stimulus matches the template in the brain, the image is recognized. For example, if there is a letter “A”, the letter “A” is recognized if the size, orientation, and shape of the letter “A” are exactly the same as the template of “A” in the mind. The model is simple and straightforward, and easy to be applied in practice. However, this model emphasizes that the image must be completely consistent with the template in the brain before it can be recognized. In fact, people can not only recognize the image that is completely consistent with the template in the brain, but also recognize the image that is not completely consistent with the template. For example, people can identify not only a specific letter “A”, but also printed, handwritten, misaligned, and different-sized letters “A”. At the same time, people can recognize a large number of images, if the recognition of each image has a corresponding template in the brain, it is impossible.
   - In order to solve the problem of template matching model, Gestalt psychologists put forward a prototype matching model. According to this model, what is stored in long-term memory is not the innumerable templates to be recognized, but some "similarity" of the images. The "similarity" abstracted from the image can be used as a prototype to test the image to be recognized. If a similar prototype can be found, the image is identified. This model is better than template-matching models,
both in the process of neural and memory exploration, and it can also explain the recognition of images that are irregular, but in some ways similar to the prototype. However, this model does not explain how people can identify and process similar stimuli, and it is difficult to implement in a computer program. Therefore, a more complex model is proposed, that is, the "pan-demonic" recognition model.

- In industrial applications, pictures are usually taken by industrial cameras, and then processed by software according to the grayscale difference of the picture to identify useful information. The representative of the image recognition software is Connex.

4. development of image recognition

- The development of image recognition has experienced three stages: character recognition, digital image processing and recognition, and object recognition. The research of character recognition began in 1950. It is generally used to recognize letters, numbers and symbols. It is widely used from printed character recognition to handwritten character recognition.

- The research of digital image processing and recognition began in 1965. Compared with analog images, digital images have great advantages such as storage, convenient transmission and compression, not easy distortion in transmission and convenient processing, which provide a powerful impetus for the development of image recognition technology. Object recognition mainly refers to the perception and understanding of the object and environment of the three-dimensional world, which belongs to advanced computer vision. It is based on the digital image processing and recognition of the combination of artificial intelligence, systems science and other disciplines research, its research results have been widely used in a variety of industrial and exploring robots. One of the shortcomings of modern image recognition technology is the poor adaptive performance. Once the target image is polluted by strong noise or the target image has large imperfection, there will not be an ideal identification result.

- The mathematical nature of image recognition is a mapping problem from pattern space to category space. At present, there are mainly three recognition methods: statistical pattern recognition, structural pattern recognition and fuzzy pattern recognition in the development of image recognition. Image segmentation is a key technology in image processing. Since the 1970s, its research has a history of several decades and has been highly valued by people. Up to now, thousands of segmentation algorithms have been proposed with the help of various theories, and the research in this field is still being actively carried out.

- There are many kinds of existing image segmentation methods, including threshold segmentation, edge detection, region extraction, and the segmentation method combined with specific theoretical tools. From the type of image to include: gray image segmentation, color image segmentation and texture image segmentation. As early as 1965, someone proposed the edge detection operator, which resulted in many classical edge detection algorithms. However, in the past 20 years, with
the rapid development of image segmentation based on histogram and wavelet transform, computing technology and VLSI technology, the research on image processing has made great progress. Image segmentation methods combine some specific theories, methods and tools, such as image segmentation based on mathematical morphology, segmentation based on wavelet transform, and segmentation based on genetic algorithm.

Use StivckV + Maixpy - IDE to image recognition development

Development Platform
- Maixpy - IDE

Development Environment
- Windows
- Linux

Developer Components
- M5Stack - StickV
Description

M5Stick-V RISC-V AI Camera

M5Stack recently launched the new AIoT(AI+IoT) Camera powered by Kendryte K210 -an edge computing system-on-chip(SoC) with dual-core 64bit RISC-V CPU and advanced neural network processor..

M5StickV AI Camera possesses machine vision capabilities, equips OmniVision OV7740 image sensor, adopts OmniPixel®3-HS technology, provides optimum low light sensitivity, supports various vision identification capabilities. (e.g. Real-time acquisition of the size, type and coordinates of the detected target ) In addition to an OV7740 sensor, M5StickV features more hardware resources such as a speaker with built-in I2S Class-D DAC, IPS screen, 6-axis IMU, 200mAH Li-po battery, and more.

It is able to perform convolutional neural network calculations at low power consumption, so M5StickV will be a good zero-threshold machine vision embedded solution. It is in support with MicroPython, which makes your code to be more concise when you use M5stick-V for programming.

Product Features

- Dual-Core 64-bit RISC-V RV64IMAFDC (RV64GC) CPU / 400Mhz(Normal)
- Dual Independent Double Precision FPU
- Neural Network Processor(KPU) / 0.8Tops
- Field-Programmable IO Array (FPIOA)
- Dual hardware 512-point 16bit Complex FFT
- SPI, I2C, UART, I2S, RTC, PWM, Timer Support
- AES, SHA256 Accelerator
- Direct Memory Access Controller (DMAC)
- Micropython Support
- Firmware encryption support
- Case Material: PC + ABS

Applications

- Face recognition/detection
- Object detection/classification
- Obtaining size and coordinates of the target in real-time
- Obtaining the type of detected target in real-time
- Shape recognition
- Video/Display
- Game simulator

USB Drive problems

- M5StickV may not work without driver in some systems. Users can manually install FTDI to fix this problem.
<table>
<thead>
<tr>
<th>Resources</th>
<th>Parameter</th>
</tr>
</thead>
<tbody>
<tr>
<td>Kendryte K210</td>
<td>Dual core 64-bit RISC-V RV64IMAFDC (RV64GC) CPU / 400Mhz (Normal)</td>
</tr>
<tr>
<td>SRAM</td>
<td>8MiB</td>
</tr>
<tr>
<td>Flash</td>
<td>16M</td>
</tr>
<tr>
<td>Power input</td>
<td>5V @ 500mA</td>
</tr>
<tr>
<td>KPU parameter</td>
<td>5.5MiB-5.9MiB</td>
</tr>
<tr>
<td>size of neural</td>
<td></td>
</tr>
<tr>
<td>network</td>
<td></td>
</tr>
<tr>
<td>Port</td>
<td>TypeC x 1, GROVE (I2C + I/O + UART) x 1</td>
</tr>
<tr>
<td>RGB LED</td>
<td>RGBW x 1</td>
</tr>
<tr>
<td>Button</td>
<td>Custom button x 2</td>
</tr>
<tr>
<td>IPS screen</td>
<td>1.14 TFT, 135 * 240, ST7789</td>
</tr>
<tr>
<td>Camera</td>
<td>OV7740 (30w pixels)</td>
</tr>
<tr>
<td>FOV</td>
<td>55deg</td>
</tr>
<tr>
<td>PMU</td>
<td>AXP192</td>
</tr>
<tr>
<td>Battery</td>
<td>200mAh</td>
</tr>
<tr>
<td>External storage</td>
<td>TF-card (microSD)</td>
</tr>
<tr>
<td>MEMS</td>
<td>MPU6886</td>
</tr>
<tr>
<td>Net weight</td>
<td>23g</td>
</tr>
<tr>
<td>Gross weight</td>
<td>82g</td>
</tr>
<tr>
<td>Product Size</td>
<td>48.24.22mm</td>
</tr>
<tr>
<td>Package Size</td>
<td>144.44.43mm</td>
</tr>
<tr>
<td>Case Material</td>
<td>Plastic (PC)</td>
</tr>
</tbody>
</table>

**TF-card (microSD) test**

- M5StickV not currently recognize all types of TF-card(microSD). We have tested some common TF-card(microSD). The test results are as follows.
<table>
<thead>
<tr>
<th>Brand</th>
<th>Storage</th>
<th>Type</th>
<th>Class</th>
<th>Format</th>
<th>Test Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>Kingston</td>
<td>8G</td>
<td>HC</td>
<td>Class4</td>
<td>FAT32</td>
<td>ok</td>
</tr>
<tr>
<td>Kingston</td>
<td>16G</td>
<td>HC</td>
<td>Class10</td>
<td>FAT32</td>
<td>ok</td>
</tr>
<tr>
<td>Kingston</td>
<td>32G</td>
<td>HC</td>
<td>Class10</td>
<td>FAT32</td>
<td>no</td>
</tr>
<tr>
<td>Kingston</td>
<td>64G</td>
<td>XC</td>
<td>Class10</td>
<td>FAT32</td>
<td>ok</td>
</tr>
<tr>
<td>SanDisk</td>
<td>16G</td>
<td>HC</td>
<td>Class10</td>
<td>FAT32</td>
<td>ok</td>
</tr>
<tr>
<td>SanDisk</td>
<td>32G</td>
<td>HC</td>
<td>Class10</td>
<td>FAT32</td>
<td>ok</td>
</tr>
<tr>
<td>SanDisk</td>
<td>64G</td>
<td>XC</td>
<td>Class10</td>
<td></td>
<td>no</td>
</tr>
<tr>
<td>SanDisk</td>
<td>128G</td>
<td>XC</td>
<td>Class10</td>
<td></td>
<td>no</td>
</tr>
<tr>
<td>XIAKE</td>
<td>16G</td>
<td>HC</td>
<td>Class10</td>
<td>FAT32</td>
<td>ok（紫色）</td>
</tr>
<tr>
<td>XIAKE</td>
<td>32G</td>
<td>HC</td>
<td>Class10</td>
<td>FAT32</td>
<td>ok</td>
</tr>
<tr>
<td>XIAKE</td>
<td>64G</td>
<td>XC</td>
<td>Class10</td>
<td></td>
<td>no</td>
</tr>
<tr>
<td>XIAKE</td>
<td>32G</td>
<td>HC</td>
<td>Class10</td>
<td></td>
<td>no</td>
</tr>
</tbody>
</table>

**Functional Description**

**Kendryte K210**

The Kendryte K210 is a system-on-chip (SoC) that integrates machine vision. Using TSMC’s ultra-low-power 28-nm advanced process with dualcore 64-bit processors for better power efficiency, stability and reliability. The SoC strives for "zero threshold" development and to be deployable in the user’s products in the shortest possible time, giving the product artificial intelligence

- Machine Vision
- Better low power vision processing speed and accuracy
- KPU high performance Convolutional Neural Network (CNN) hardware accelerator
- Advanced TSMC 28nm process, temperature range -40°C to 125°C
- Firmware encryption support
- Unique programmable IO array maximises design flexibility
- Low voltage, reduced power consumption compared to other systems with the same processing power
- 3.3V/1.8V dual voltage IO support eliminates need for level shifters

**CPU**
The chip contains a high-performance, low power RISC-V ISA-based dual core 64-bit CPU with the following features:

- Core Count: Dual-core processor
- Bit Width: 64-bit CPU 400MHz
- Frequency: 400MHz
- ISA extensions: IMAFDC
- FPU: Double Precision
- Platform Interrupts: PLIC
- Local Interrupts: CLINT
- I-Cache: 32KiB x 2
- D-Cache: 32KiB x 2
- On-Chip SRAM: 8MiB

OV7740

- support for output formats: RAW RGB and YUV
- support for image sizes: VGA, QVGA, CIF and any size smaller
- support for black sun cancellation
- support for internal and external frame synchronization
- standard SCCB serial interface
- digital video port (DVP) parallel output interface
- embedded one-time programmable (OTP) memory
- on-chip phase lock loop (PLL)
- embedded 1.5 V regulator for core
- Sophisticated Edge Rate Control Enables Filterless Class D Outputs
- 77dB PSRR at 1kHz
- Low RF Susceptibility Rejects TDMA Noise from GSM Radios
- Extensive Click-and-Pop Reduction Circuitry
- array size: 656 x 488
- power supply: – core: 1.5VDC ± 5% – analog: 3.3V ± 5% – I/O: 1.7 ~ 3.47V
- temperature range: – operating: -30° C to 70°C – stable image: 0° C to 50° C
- output format: – 8-/10-bit raw RGB data – 8-bit YUV
- lens size: 1/5”
- input clock frequency: 6 ~ 27 MHz
- max image transfer rate: VGA (640x480): 60 fps – QVGA (320 x 240): 120 fps
- sensitivity: 6800 mV/(Lux-sec)
- maximum exposure interval: 502 x tROW
- pixel size: 4.2 μm x 4.2 μm
- image area: 2755.2 μm x 2049.6 μm
- package/die dimensions: – CSP3: 4185 μm x 4345 μm – COB: 4200 μm x 4360 μm

MAX98357

- Single-Supply Operation (2.5V to 5.5V).
- 3.2W Output Power into 4Ω at 5V
- 2.4mA Quiescent Current
- 92% Efficiency (RL = 8Ω, POUT = 1W)
- 22.8µVRMS Output Noise (AV = 15dB)
- Low 0.013% THD+N at 1kHz
- No MCLK Required
- Sample Rates of 8kHz to 96kHz
- Supports Left, Right, or (Left/2 + Right/2) Output
- Sophisticated Edge Rate Control Enables Filterless Class D Outputs
- 77dB PSRR at 1kHz
- Low RF Susceptibility Rejects TDMA Noise from GSM Radios
- Extensive Click-and-Pop Reduction Circuitry

**AXP192**

- Operation Voltage: 2.9V - 6.3V(AMR: -0.3V~15V)
- Configurable Intelligent Power Select system
- Current and voltage limit of adaptive USB or AC adapter input
- The resistance of internal ideal diode lower than 100mΩ

**MPU6886**

Gyroscope features

- Digital-output X-, Y-, and Z-axis angular rate sensors (gyroscopes) with a user-programmable full-scale range of ±250 dps, ±500 dps, ±1000 dps, and ±2000 dps and integrated 16-bit ADCs
- Digitally-programmable low-pass filter
- Low-power gyroscope operation
- Factory calibrated sensitivity scale factor
- Lens size: 1/5"
- Self-test

Accelerometer features

- Digital-output X-, Y-, and Z-axis accelerometer with a programmable full scale range of ±2g, ±4g, ±8g and ±16g and integrated 16-bit ADCs
- User-programmable interrupts
- Wake-on-motion interrupt for low power operation of applications processor
- Self-test

**SPI / I2C dual communication mode**
Note: There are two versions of M5StickV currently released by M5Stack. When programming, users need to configure differently according to their corresponding pin mapping. The specific differences are as follows.

- In the M2StickV circuit design of the I2C single-mode (blue PCB) version, MPU6886 only supports the user to configure its communication mode to I2C, and its pin mapping is SCL-28, SDA-29.
- In the SPI/I2C dual mode (black PCB) version of the M5StickV circuit design, MPU6886 supports the user to configure its communication mode to SPI or I2C, and its pin mapping is SCL-26, SDA-27. When using, you can switch CS Pin level to switch modes (high level 1 is I2C mode, low level 0 is SPI mode).
- The specific pin mapping is shown below:

![Pin Mapping Diagram]

Links

datasheet
- MPU6888
- SH200Q

Web page
- sipeed

GITHUB
- API

schematic
K210-CAM

Procedure

- Maixpy reference example


8.1 Set up MaixPy environment

1.1 What is MaixPy?

MaixPy is a project transplanted Micropython into K210 (a 64-bit dual-core with hardware FPU, convolution accelerator, FFT, Sha256 of RISC-V CPU), supporting MCU regular operations, it also incorporates hardware acceleration AI machine vision and a microphone array, 1TOPS computing power core module is less than ¥50, which quickly develops very low cost and volume of practical AIOT field of smart applications.

The first thing you need to know is Maixpy uses MicroPython scripting syntax, so it doesn't require compilation like C language, it can be used without an IDE: using the serial terminal tool which has been installed previously.

Using IDE, you can edit scripts in real time on the computer and upload them to the development board, which is convenient to execute scripts directly on the development board, view camera images in real time on the computer, save files to the development board and so on.

However, using an IDE will compress and transport some resources, so the performance will be reduced, and if MaixPy goes down, it's hard to find problems like serial terminal.

![Image of MaixPy IDE](image)

For more information, please click official website to view the official instructions.

1.2 What can MaixPy be used for?
2. How to set up the usage environment?

2.1 Preparation of the development environment

2.1.1 Install Driver

- Before using MaixPy, we need to install the serial driver before we can proceed to the next step of development and use. The board is connected to the computer via a USB-to-serial device (K210 does support USB hardware), please install the driver according to the USB to serial port chip model of the board.
- Click to download and install the serial driver
- When operating a serial port on Linux or Mac, if you don't want to use “sudo” every time, perform “sudo usermod -a -G dialout $(whoami)” to add yourself to the “dialout” user group, which may require a logout or reboot.

2.1.2 Burn Firmware

- The firmware must be V0.3.1 or above to use the MaixPy IDE, otherwise the MaixPy IDE will not be connected. Try to check the firmware version and IDE version before use, and update it to the latest version to ensure normal use.
- Click to download the firmware burner Kflash, a compressed package after downloading.

Kflash_gui is cross-platform and can be used on multiple systems (including Windows, Linux, MacOS, and even Raspberry Pi).

Windows version of Kendryte may be unable to download. Please use the software kflash_gui to download.

- Click to download and install firmware
Burn firmware

After downloading, use the firmware burner to burn the firmware to M5StickV:
2.2 Download and install software

2.2.1 Click to download MaixPy-IDE

Note: Please refer to the readme.txt file in the latest version folder for the list of files. If the download speed is slow, please use the cdn link to download.

Install software

Download the file, directly double-click the exe file to run the installation program in Windows

Test connection
Open MaixPy IDE and select the model of the development board from the top toolbar. Select M5Stickv to connect.

Tool-> Select Board (Tools -> Select Development Board)

Click "connect" to connect MaixPy IDE

When the connection is successful, it will change from green to red.

Below the connect button is the run button, which executes the py file in the current edit area.
Click the Run button again (red) to stop running the current code.

3. Troubleshooting Guide of USB Serial Port

If you do not see a serial port, check for hardware problems in the following order.

- Insert the computer, whether there is a ding-dong sound, such as the sound of the USB drive loading when inserting the U disk. If there isn't, it means that there is a problem with the serial chip on the hardware.
- Replace the cable and try again. Replace the USB port of the computer and try again. It still won't load out, change the computer to confirm.

If you cannot burn firmware, check for hardware problems in the following order.

- Use the serial port tool to see if the MAIXPY firmware exists in the hardware.
- Set 115200 baud rate to connect the serial port, press the reset key (RST) to receive the data of the chip. Whatever it happens means the serial port chip is working normally, but if there is nothing, it means the hardware is abnormal.
- Based on the above, burn the firmware again. Before burning, press the BOOT key of the hardware and press reset, and then release the BOOT key, which means the burning is processing normally. If not, it means that Flash is damaged, you can try to burn to SRAM. If the burning fails, it means that the serial port chip is abnormal.
- If you get to this step and still can't fix the problem, then the hardware does have a defect.

3.1 Burning mechanism introduction of K210
We often call this a key download circuit, can easily complete the control of BOOT and RST pin and enter the burning mode through the control of the serial port RST and DTR. As described above, the hardware circuit is expected to replace human to automatically perform the operation of pressing RST and BOOT, which is strongly dependent on hardware implementation. Only based on this can data transmission of TX and RX be carried out, so we need to use functional pins of UART serial port.

Kflash can be divided into a variety of versions of a variety of burning triggers. We can simply divide it into several categories, 115200 baud rate at low speed and 1500000 baud rate at high speed, taking the matching burning mode of these two types of baud rate as the difference point.

If it is found that the download process fails, the baud rate can be reduced appropriately, because the serial port chip is not stable. The selection of the type in the tool will only affect the trigger of the first burning mode, and after that the burning firmware will be burned at the configured baud rate, usually not exceed the communication burning speed with flash, commonly seen in 50~60 KB/S.

If you find that no matter how to replace the burning mode can not enter, either the burning version does not match, or the serial chip DTR RST pin problem (physical).
8.2 Color Recognition

Preparation for development

- Complete the device link
- Complete firmware burning
- Complete software environment

Example code
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, -30, -20],
                     [55, 64, 52, -9, 3],
                     [29, 65, -109, -18, 36, 99],
                     [38, 68, -25, -10, 92, 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)
uart_Port.write(str(blobs))

def show_fps():
    fps = clock.fps()
    img.draw_string(200, 1, ('%2.2fps: %d') % (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_threshold=4096)
        if blobs[i]:
            blobs_output(blobs[i])

    lcd.display(img)

---

**Functional Assignment**
- Identify four colors
- The serial port outputs data
- The display screen displays the recognized color blocks

Define the color you want to recognize

Open the MaixPy IDE and select Tool -- Machine Vision -- Curve Editor

Open source image location and select frame buffer.
Adjust the Lab tie down value mainly in the binary image field, and white pixel is the tracked pixel

**Related knowledge**

**MaixPy 机械视觉 API**

**Full Knowledge of Lab color space**

- **Name** Before we begin, let's clarify the name of the Lab Color Space: The full name of Lab is CIELAB, sometimes also written CIE Lab*. CIE stands for the International Commission on Illumination, an International authority on lighting, colour, etc..

- **Channel** Lab is composed of a brightness channel and two color channels. In Lab color space, each color is represented by three numbers, L, a, b, and the meaning of each component is as follows: L stands for brightness, a is the component from green to red, b is the component from blue to yellow.

- **Perceptual uniform** Lab is designed based on people's perception of colors. More specifically, it is perceptual uniform. Perceptual uniform means that if the number (L, a, b) changes equally, then it brings about a similar degree of visual change. Lab is more consistent with human vision than RGB and CMYK color space, and is easier to adjust. If you want to adjust the brightness (regardless of the Helmholtz -- Kohlrausch effect, refer to the note below), adjust the L channel, and if you want to adjust only the color balance, adjust a and b respectively. Note: Helmholtz–Kohlrausch effect is an illusion in the human eyes -- that colors appear brighter when they are saturated.

- **Device-independent** Lab has a nice feature -- Device-independent. That is, given a white point in the color space (the figure below represents a white spot in a color space), the color space can clearly determine how each color is created and displayed, regardless of the display medium used.
For example, when you want to convert an RGB image on the screen to a CMYK image for printing, you can first convert it from RGB to Lab and then convert the Lab image to CMYK mode. Because gamut of Lab is larger than RGB and CMYK (the gamut of Lab is so large that a large part of it is beyond the range of human vision that it cannot be called "color"). It is important to note that Lab defines the color relative to the white point. We will not know the other colors until we define the color of the white point (e.g., it is defined as CIE Standard Illuminant D50).

- **Range of value** In theory, L, a, and b are all real numbers, but in practice they are confined to a range of integers: The larger the L is, the higher the brightness is. When L is 0, it represents black, and when L is 100, it represents white. a and b are both gray when they are 0. When a goes from a negative number to a positive number, the corresponding color will change from green to red. When b goes from a negative number to a positive number, the corresponding color will change from blue to yellow. In practical application, we often use the range of color channel between -100~+100 or -128~127.

- **Visualize** Lab* has three components in total, so it can be presented in three-dimensional space. In two-dimensional space, the Chromaticity Diagram is often used to visualize it, that is, to fix the brightness L, to see the change of a and b. Note that these visualizations are not accurate, it's just helpful to understand.

- **CIELUV** There is a color space similar to Cielab, called Cie 1976 (L, U, V), also known as Cieluv. The L of the color space is the same as the CIELAB, but the color component is different.

- **Conversion between LAB and RGB and CMYK** Since RGB and CMYK are both device related, they cannot be directly converted to Lab. So before the
conversion, you must define an absolute color space, such as sRGB or Adobe RGB. Conversion from RGB to sRGB is device independent, but the subsequent conversion are device independent.
8.3 Shape Recognition

Preparation for development

- Complete the device link
- Complete firmware burning
- Complete software environment

Example Code

```python
# identify the rectangle
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(threshold = 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)
```
# identify the circle
```python
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_margin
        r_min = 5, r_max = 45, r_step = 20):
        img.draw_circle(c.x(), c.y(), c.r(), color = (255, 0, 0))
        print("r \%f" % c.r())
        print("FPS \%f" % clock.fps())
lcd.display(img)
```

### Functional Assignment

- Identify the specified shape
- Return the corresponding data

### Relate knowledge

MaixPy 机械视觉 API
8.4 Face Recognition

Preparation for development

- Complete the device link
- Complete firmware burning
- Complete software environment

Burn specified firmware

We use the MaixPy IDE to run this routine. First, we need to download the model we need from the MaixHub, which need us to provide the machine code, so we download the bin file that can get the machine code, then burn it to token_gen.bin file address

Get Machine code

Use kfalsh_gui to burn ken_gen.bin file, click it to download.
View the machine code in the serial output

After the firmware is burned, connect the computer, open the serial assistant, then open the serial port.

Please Send Bellow Data to Sipeed --> support@sipeed.com:

https://blog.csdn.net/gmq_syy
Download model

With the machine code we can start to download the corresponding model files.

Model download address

List of files to extract:

<table>
<thead>
<tr>
<th>Name</th>
<th>Size</th>
<th>Type</th>
<th>Date</th>
<th>CRC32</th>
</tr>
</thead>
<tbody>
<tr>
<td>HUBK7sXdaT7448k85w3W.model</td>
<td>235223</td>
<td>Flash</td>
<td>2020/7/17</td>
<td>54534XXZ</td>
</tr>
<tr>
<td>flash_kit.json</td>
<td>859</td>
<td>JSON</td>
<td>2020/11/11</td>
<td>3M2A6756</td>
</tr>
<tr>
<td>HUBK7sXdaT7448k85w3W.bin</td>
<td>682432</td>
<td>bin</td>
<td>2019/11/29</td>
<td>984F0171</td>
</tr>
</tbody>
</table>

The blue part is the model file, and the yellow part is the bin file of MaixPy. This is the compact version, and the size is relatively small, just more than 600 KB. Inside the json file is the configuration, which is about where these files should be downloaded to Flash and whether they need to be checked.
Burn model files

Download the Kkfpkg file to our development board using kfalsh_gui and run it
Run example code

code address
import sensor, image, lcd  # import 相关库
import KPU as kpu
import time
from Maix import 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.clk()  # 初始化系统时钟，计算帧率

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

    # 初始化lcd
    lcd.init()  
    sensor.reset()  #初始化sensor 芯片
    sensor.set_pixformat(sensor.RGB565)  
    sensor.set_framesize(sensor.QVGA)  
    sensor.set_vflip(1)  #设置垂直镜像
    sensor.set_hmirror(0)  #设置水平镜像
    sensor.run(1)  #开摄像头

    anchor = (1.089, 2.5245, 2.0465, 3.04056, 3.09987, 5.3658, 5.15437, 5.92275, 6.713875)

dst_point = [(64, 59), (64, 59), (64, 59), (64, 59), (64, 59), (64, 59), (64, 59)]  #standface key point posit
    a = kpu.init_yolo2(task_fd, a, b, 1, 0.3, 0.3, 5, anchor)  #初始化人脸检测模型

    # 设置显示
    img_lcd=image.Image()  
    img_lcd=image.Image()  # 设置显示buf

    # 设置人脸图片butf
    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[0]  #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，读出来
    names = ['Mr.1', 'Mr.2', 'Mr.3', 'Mr.4', 'Mr.5', 'Mr.6', 'Mr.7', 'Mr.8', 'Mr.9', 'Mr.10']  #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 == 0:  # 如果检测到人脸
        for i in code:  # 迭代坐标
            a = img.draw_rectangle(i.rect())  # 在屏幕上显示方框
            anchor = (1.089, 2.5245, 2.0465, 3.04056, 3.09987, 5.3658, 5.15437, 5.92275, 6.713875)

    # Landmark for face 5 points
        fmap = kpu.forward(task_fd, 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
        le1(i.x()+int(plist[1]*i.w()) - 10), i.y()+int(plist[3]*i.h()))  # 计算左眼位置
        re1(i.x()+int(plist[1]*i.w()) - 10), i.y()+int(plist[3]*i.h()))  # 计算右眼位置
        nose(i.x())+int(plist[3]*i.w()), i.y()+int(plist[3]*i.h()))  # 计算鼻子位置
        lm1(i.x()+int(plist[1]*i.w()) - 10), i.y()+int(plist[3]*i.h()))  #计算左嘴角位置
        rm1(i.x()+int(plist[1]*i.w()) - 10), i.y()+int(plist[3]*i.h()))  #计算右嘴角位置

        a = img.draw_circle(le1, le1, 4)  
        a = img.draw_rectangle(le1, le1, 4)  
        a = img.draw_rectangle(nose, nose, 4)  
        a = img.draw_rectangle(5, 5, 4)
Run the program with MaixPy IDE

Program running as shown in figure
Press BUTTON A to record the face. After the face is recorded, the name will be assigned in order and displayed when the face is recognized.

**Related knowledges**
8.5 QR code Identification

Preparation for development

- Complete the device link
- Complete firmware burning
- Complete software environment
- Use MaixPy to generate recognizable QR codes Select -> Tools -> Machine Vision -> AprilTag -> TAG36H11 to generate the recognizable code

example code
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。

# 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）乘以图像中的像素。
# 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）乘以图像中的像素。
# 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_rotation(),
                       # 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 Maintenance

After-sales Service

Return service is limited to goods not opened within 7 days after the receipt date of logistics of the products. The freight or other risks incurred in return shall be borne by the customer.

Customers should provide the purchasing invoice and warranty card as the warranty certification when a warranty is being asked.

**Elephant Robotics will be responsible for the hardware faults of products caused by the normal using during the warranty period.**

The warranty period starts from the date of purchase or the receipt date of the logistics.

The faulty parts from the products will be owned by Elephant Robotics, and the appropriate cost will be charged if necessary.

If you need to apply for warranty service, please contact our customer service first to confirm the detailed information. The following is warranty terms of detailed components:

**Note: If there is a conflict with the Product Brochure, the User Manual shall prevail.**

**Servos**

<table>
<thead>
<tr>
<th>Warranty Period</th>
<th>Warranty Services</th>
</tr>
</thead>
<tbody>
<tr>
<td>≤1 months</td>
<td>Elephant Robotics offers a free new sever motor and bear the freight.</td>
</tr>
<tr>
<td>1-3 months</td>
<td>Elephant Robotics offers a free new sever motor, customs shall bear the freight.</td>
</tr>
<tr>
<td>≥3 months</td>
<td>Customers need to buy it themselves.</td>
</tr>
</tbody>
</table>
### Electrical Parts (M5 Hardware)

<table>
<thead>
<tr>
<th>Warranty Period</th>
<th>Warranty Services</th>
</tr>
</thead>
<tbody>
<tr>
<td>≤3 months</td>
<td>Customers need to send it back after disassembly, Elephant Robotics shall send a new one for free and bear the freight out and home.</td>
</tr>
<tr>
<td>3-6 months</td>
<td>Customers need to send it back after disassembly and bear the freight out and home, Elephant Robotics shall send a new one for free.</td>
</tr>
<tr>
<td>≥6 months</td>
<td>Customers need to buy it themselves.</td>
</tr>
</tbody>
</table>

### Structure Parts, including Shell Parts

<table>
<thead>
<tr>
<th>保修期限</th>
<th>保修服务</th>
</tr>
</thead>
<tbody>
<tr>
<td>≤1 years</td>
<td>Elephant Robotics offers free new components once, customs shall bear the freight.</td>
</tr>
<tr>
<td>≥1 years</td>
<td>Customers need to buy it themselves.</td>
</tr>
</tbody>
</table>

During the warranty period of the delivered product, the company only repairs the malfunctions that occur during normal use of the robot for free. However, in the following cases, the customer will be charged for repairs (even during the warranty period):

- Damage or malfunction caused by incorrect use and improper use different from the manual.
- Failure caused by unauthorized disassembly by the customer.
- Damage caused by improper adjustment or unauthorized repairs.
- Damage caused by natural disasters such as earthquakes and floods.

Please strictly follow the instructions in this manual and related manual to operate the robot.

### Limit

<table>
<thead>
<tr>
<th>Joint</th>
<th>Range of Motion/Angle</th>
</tr>
</thead>
<tbody>
<tr>
<td>Joint 1</td>
<td>-165 ~ +165</td>
</tr>
<tr>
<td>Joint 2</td>
<td>-165 ~ +165</td>
</tr>
<tr>
<td>Joint 3</td>
<td>-165 ~ +165</td>
</tr>
<tr>
<td>Joint 4</td>
<td>-165 ~ +165</td>
</tr>
<tr>
<td>Joint 5</td>
<td>-165 ~ +165</td>
</tr>
<tr>
<td>Joint 6</td>
<td>-175 ~ +175</td>
</tr>
</tbody>
</table>
After we receive myCobot, we should pay attention to the limit of each joint of myCobot, and the rotation angle of each axis cannot exceed its own maximum physical limit.

- It should be turned at a small angle and gently. After reaching the limit, you should not continue turning.
- Be careful not to touch the robot arm itself when the joint is running so as not to knock it out.

### Connectivity issues (November 2020 batch)

**Problem description:** when the arm suddenly loses force and can not be driven, open the J5 joint shell to check whether the connection line is off, then need to cut off the longer pin, attach insulation tape and then put back to the steering gear, and cover the cover plate. **Replacement steps**

- Need to remove back cover housing
- Cable adjustment and adhesive tape Youtube-video link: https://youtu.be/1wq0kTJVqw4

### Shell Disassembly

Youtube-[Video Link: https://youtu.be/wHzFsExkYrE]  
**Steps**

- Prepare the items as shown, loosen the screws of the corresponding shell, force properly, and gently remove the shell from the left and right.
- After removing the shell, check that there is no damage inside, you can install the new shell.
- Fasten the new shell, the shell and the manipulator have a clear touch, after fastening, the screws are placed and tightened to complete the shell removal.

**Calibration**

Youtube-video Link: https://youtu.be/vGznxW4OF10
Burning calibration on myStudio

When the zero position of the manipulator is aligned with the zero position calibration groove of each joint

Adjust the zero position calibration groove manually and press the A key (leftmost, set the steering gear zero) to calibrate each joint in turn

After calibration, the basic screen will show that all steering gear bits have been set.

Press the B key (middle key) to test the steering gear, the test is finished and the correction is over.

If not, press the C key (right-click) to reset the steering gear zero and repeat the above steps.
2 常见问题FAQ

Hardware

Q: why it appear small jitter when I use the vertical state, but not when running?
A: please check whether it is vertical, vertical state is not affected by gravity, mechanical voids lead to small jitter, after the use of this state will not occur. The recommended speed in this state is 400-500.

Q: Will ROS system charge later?
A: ROS is open source and will update on our github. There is no charge for firmware upgrades.

Q: myCobot joint hard limit
A: The first axis and fifth axis have limit, the first axis clockwise about 160°, counterclockwise about 160°. the fifth axis clockwise and counterclockwise rotation about 160°. Note: when turning the robot arm, it should be small angle and gently, after reaching the limit. you can not continue to rotate.

Q: what is the unit of velocity of the robotic arm?
A: speed 180 degrees per second

Q: Is it just not support M5STACK's steering gear, will it support other M5 sensors?
A: M5STACK Basic currently supports all sensors.

Q: Do Atom support sensors in the future?
A: atom support needs to further open the interface, within our plan, need to wait

Software

Q: why can't my compiler find the corresponding device?
A: You need to build the development environment and install the corresponding project library to develop the equipment.

Q: why my compiler can't compile the sample program properly?
A: the required project library is not installed or the project inventory is in conflict, it is recommended to check that the project library is installed correctly. If the installation is correct and still can not be compiled, please reinstall the arduino development environment.

Q: why did I burn the firmware to the ATOM terminal and the device didn't work properly?
ATOM terminal firmware needs to use our factory firmware, you cannot change other unofficial firmware in use. If the device burns other firmware accidentally, you can use "myCobot firmware burner" to select ATOM terminal-select serial port-select ATOMMAIN firmware to burn the ATOM terminal.

UIFlow

Q: UI Flow doesn't support the latest firmware? A: Need M5STACK update, we have asked M5 to help us update, the time cycle is unknown.

RoboFlow

Q: Can I use robotStudio software programming? A: Our own industrial programming software RoboFlow can be used, robotStudio is ABB company, it cannot communicate with us.

ROS

Q: Does the ROS system operate with a computer attached to the robot arm? A: Yes, connect with the computer is ok.
Q: Ros folder downloaded from github runs only display controls but not display myCobot 3D models. A: You need to open rviz,rosrun rviz rviz manually
Q: Range out or error[101] occurs when in runtime A: Check that the serial port is correct, basic and atom firmware are correct.
Q: operation mode A: It need to open 3 terminal for current operation

myStudio

Q: Does myStudio replace the UIFlow? A: No, it insteads of the current downloader.

myCobot phone controller

Q: I have a few questions to ask about APP: 1) Bluetooth has two controls on the right, one is back to zero, the other is? 2) The sixth axis has no mechanical zero, how to get the robot back to zero? Rz the Euler angle is 0. 3) Press the control button of the joint and end shaft, then loose, will it keep moving? A: 1. Another free model.

2. When the robot arm is in calibration correction, the current position is set to zero by default.

3. After release, you should stop. APP use the jog control. However, due to the instability of Bluetooth connection, there may be a stop instruction but arm do not received.

Firmware Burner

Q: Motor don't work A: Use the latest version 1.3 firmware burner, burn atommain to the atom. Only if computer link to the end of the robotic arm typeC interface can you update the firmware to atom.
Q: After burn, how to return to the factory settings? A: Download firmware burner or arduino program, burn main firmware.

<table>
<thead>
<tr>
<th>Others</th>
</tr>
</thead>
<tbody>
<tr>
<td>Q: What is the difference between MainControl and Transponder?</td>
</tr>
<tr>
<td>A: MainControl is the factory with its own, transponder is the transfer program, after burning can be directly sent packet protocol control.</td>
</tr>
<tr>
<td>Q: default software - Drag to teach, action can only be saved to ram can not be saved to flash A: old version will be released with the Chinese version mainControl after the bug adjustment</td>
</tr>
<tr>
<td>Q: each Data is a hexadecimal number, right, and then converted to hex characters to send to the machine A: right, serial communication sends and receives data in HEX form</td>
</tr>
</tbody>
</table>

**Others**

Q: Can I get real-time data while executing instructions

A: It is not allowed for the time being, the bus can not be disturbed when working

Q: The end interface rudder model has a limited end of the control card, what is the relationship with the M5stack? Is it independent?

A: End of the device needs to be able to adapt to our controller, it can support PWM mode control of the steering gear.

Q: what does the speed parameter 0-100 in the WriageAngle mean

A: speed parameter 0-100 is the percentage of speed

Q: Can I use GetAngles to get the angle information of 6 axes?

The return value of the

A: GetAngles is a float array with a length of 6. The index starts at 0, followed by 6 joints

Q: Is servo or stepper motor?

A: myCobot carry six servo motors.

Q: repeated positioning accuracy

A: +/-0.5mm

Q: what is the reducer structure?

A: reducer is gear set.

Q: what is the gear?

A: metal tooth structure

Q: what is this programmed with?
A: myCobot is open ROS, now supports most platforms on the market including UIFLOW, Arduino, microPython, FreeROTS

Q: How is the Real Time
A: Uart communications, maximum 50 ms delay, normal 5 ms

Q: where can the detailed documentation, sample code of the Arduino library and API interface be seen?
A: check our github: https://github.com/elephantrobotics/myCobot

Q: How much time continuous operation / motor life
A: 300-500 hours, rest 15-30 minutes before use will prolong the life of the arm;

Q: shell material
A: plastic; photosensitive resin; SLA

Q: Do the robot arm have torque sensors
A: most collaborative robots don’t have torque sensors

Q: Brand of motor
A: self-made motor, external processing

Q: Turn the joint tape encoder?
A: yes

Q: not harmonics?
A: notss

Q: I don’t understand servo steering gear, is it a steering gear like a model?
A: it is a steering gear like a model, but we customize and modify a lot, is more suitable for the manipulator. Six-axis linkage, three-power interpolation, not as uniform as industrial robots, the final price here.

Q: motor parameters
A: large steering gear: rated load 10 kgcm, maximum speed 60 rpm, voltage 7.4 V, encoder 12 bit magnetic knitting.
small steering gear: rated load 2 kgcm, maximum speed 80 rpm, voltage 7.4 V, encoder 12 bit magnetic knitting.

Q: ROS version
I rosdistro: kinetic //
I rosversion: 1.12.17//

Q: Angle Accuracy
I 360/4096=0.0879°

Q: power supply range
I: 6~9V
3~5A

Q: IO input, output, voltage, how to debug; why marked 5V?
A: 3.3 V, atom of IO control is gradually open; there are 5V of power; debugging needs to see which platform to develop, arduino, uiFlow, python is different

Q: IO port connection problem
A: IO according to the position, it is divided into the IO, on the basic of the manipulator base and the IO; IO at the end of the manipulator, which is divided into digital signal input and output DIO, analog signal, communication I2C, SPI, power grounding.

Q: Are encoders absolute or incremental?
A: absolute value (steering gear with high precision encoder)

Q:asic software brush uiFlow, later with your burning software brush back maincontrol, why the robot arm track recording software can not be used
A: atom firmware version is not compatible, can only use the same version of firmware; update the atom version can

Q: why did it remind TimeOut when burning?
A: power up again or press the reset button when burning

Q: why did lights go out when I burn Bluetooth or Transponder atom
A: need command control to reopen LED, close by default

Q: why my equipment can't be used after burning the demo program
A: need to check your version number, corresponding to the version number of firmware can, different versions of the program is incompatible

Q: the accuracy range deviation of the manipulator is a little large
A: the current accuracy, we need more precision to view our industrial arm

Q: I can't drag to teach after burning
A: First check whether the basic and atom are burned; whether the burned firmware corresponds to the requirements to be achieved; and whether the burning is the latest firmware bottom Basic burning atommain at the top

Q: can't control robotic arm when burn transponder
A: Atom also need to burn, burn firmware atom2.1alpha; Basic burn transponder

Q: by which of the six steering gear
A: Atom control at top

Q: python API sample tutorial
A: currently have demo, test code under github Test folder
Q: Do you need to start again after drag the instruction

A: because when re-recording joint motor lock position can not be cut off, click on the release can make the arm joint loose, can be the next drag teaching; power failure can also start again

Q: Does instruction be achieved by reading the set potential value (drag instruction principle)

A: principle is like this

Q: What information does the send to the joint motor through the ROS? Can feedback encoder information

A: potential value; read potential value

Q: show six values in Get coords

A: x,y,z,rx,ry,rz, translation plus rotation; z-y-x compliance

Q: AC flash red light, robotic arm screen not on

A: if the motor doesn't move, may be Basic is broken. Ac frequently flash red lights mean power supply, then power off and power supply. It could be some pcba broken.
3 Resources

Website

Official Website
https://www.elephantrobotics.com/en/

myCobot github-software
https://github.com/elephantrobotics/myCobot

M5 UI Flow
https://docs.m5stack.com/#/zh_CN/quick_start/m5core/m5stack_core_get_started MicroPython

Videos

youtube

- maintenance
  - connecting line https://youtu.be/1wq0kJVqw4
  - disassembly https://youtu.be/wHzFsExkYrE
  - limit https://youtu.be/PUrU-mynljw
  - calibration https://youtu.be/vGzmxW4OF10

- Tutorials unboxing https://youtu.be/Lwi8UoihzNc
  - free move https://youtu.be/WzrbOrdQop0
  - Maincontrol https://youtu.be/VKd8b989M8g
  - ROS https://youtu.be/-Jo_J8RaXc
  - myStudio https://youtu.be/Kr9l6ZPF4w

- 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