

About This Manual

Purpose

This document describes the basic concepts and functions of the AUBO SDK, how to install and use the AUBO SDK, and how to diagnose and resolve issues and errors that arise when using the SDK, aiming to help users understand and quickly master the functions and usage of the AUBO SDK.

Structure of the document

The structure of this document is as follows:

- [Overview](#): An introduction to the AUBO SDK is and its main functions.
- [Introduction to SDK Function Modules](#)
- Protocol
 - [Introduction to JSON-RPC Protocol](#)
 - [Introduction to RTDE Protocol](#)
 - [Introduction to Script Protocol](#)
- Prepare: A brief introduction to the components, running environment and download and installation methods of the AUBO SDK, and how developers use the AUBO SDK.
 - [Introduce](#)
 - [Operating Environment Requirements](#)
 - [Download and Installation](#)
 - [Version Control](#)
- Quick Start: A brief introduction on how to quickly run the example codes provided by the AUBO SDK.
 - [Introduce](#)
 - [SDK PC and Robot Communication Setup](#)
 - [Examples of Running C++ SDK](#)
 - [Examples of Running Python SDK](#)
 - [Examples of Running Java SDK](#)

- [Troubleshooting](#): Brief introduction of the tools used for fault localization and troubleshooting in the AUBO SDK, including error codes, logs, and exception handling.
- [Introduction to ServoJ Function](#)
- [F & Q](#)
- Appendix:
 - [Error Code](#)
 - [RTDE Menu](#)

Overview

AUBO SDK, a software development kit (SDK) developed by AUBO (Beijing) Robotics Technology Co., Ltd., provides a rich set of API interfaces for programming and deploying applications for AUBO robots, aiming to help developers quickly and flexibly build their own applications to control and utilize AUBO robot arms to meet the needs in various application scenarios.

AUBO SDK supports multiple programming languages, including C++, Python, Lua, and JavaScript. It is compatible with both the Windows and Linux operating systems, allowing developers to choose according to their needs. Developers can develop their own applications through the SDK, enabling interaction and communication with AUBO robots.

In addition, AUBO SDK also provides extensive development documents and example codes, allowing developers to get started as beginners and develop their own robot applications more quickly.

Functional Modules of SDK

RPC, RTDE, and SCRIPT are the three major functional modules of AUBO SDK. They are independent of each other and can be used in combination to achieve more flexible and efficient robot control solutions.

RPC Module

RPC (Remote Procedure Call) is a remote procedure call protocol used to achieve remote command transmission and data communication between external devices (such as computers) and the robot controller. To view the introduction to the **JSON-RPC protocol**, please read the document [JSON-RPC Protocol Description](#).

The RPC module in the SDK uses the JSON-RPC 2.0 protocol for data exchange. It supports the client/server mode and request/response mode, featuring simplicity, lightweight, cross-platform, extensibility, etc. Moreover, it supports multiple programming languages and operating systems.

The RPC module supports various types of commands, such as motion commands, status commands, and IO commands, meeting various needs in different application scenarios. External devices (such as computers) can send command requests to the controller through the RPC module in the SDK, and the controller can perform corresponding actions based on the content of the command, such as motion, pause and stop, and return the execution results to the external devices (such as computers).

At the same time, the RPC module can also work with the RTDE module or SCRIPT module to implement more complex robot applications. For example, users can use the RPC module to call motion command interfaces (such as `moveJoint` and `moveLine`), and subscribe to obtain the robot's real-time status, such as actual position and attitude, through the RTDE module.

RTDE Module

RTDE (Real-Time Data Exchange) is a real-time data exchange protocol used to achieve real-time data transmission between external devices (such as computers) and the robot controller, including the robot's joint angle, speed, acceleration, and other status information. The RTDE module in the AUBO SDK achieves real-time data exchange between the controller and

external devices based on the RTDE protocol. In addition, the RTDE module can be used simultaneously with the RPC module or SCRIPT module without compromising the system's real-time performance. To view the introduction to the **RTDE protocol**, please read the document [RTDE Protocol Description](#).

The RTDE module in the SDK can transfer data from the controller to external devices (such as computers) and can also transfer data from external devices to the controller. External devices can obtain real-time data such as the robot's joint angle, speed and acceleration through the RTDE module, and can also write data to the controller's registers via the RTDE module.

The RTDE menu is a file format used to configure the RTDE data stream. It defines information such as the names and types of data, for example, the robot's position, joint angle, speed, current, etc., and supports various data types including boolean, integer, floating-point, etc. To view the definitions, please read the document [RTDE Menu](#).

SCRIPT Module

The SCRIPT module in the SDK supports sending local script programs to the robot controller, which will then load and execute the script program. To view the introduction to the **script protocol**, please read the document [Script Protocol Description](#).

The SCRIPT module supports Lua scripts, including basic statements, control structures, function definitions, etc. It provides a rich set of robot control APIs, such as motion control, algorithms, and obtaining robot status. In addition, the SCRIPT module offers debugging and error handling features for script programs, such as setting breakpoints.

The SCRIPT module can also be used in conjunction with the RTDE module or RPC module. For example, users can write script programs locally and send the local script programs to the robot controller for execution via the SCRIPT module, while using the RTDE module to monitor the robot's status in real-time.

Introduction to JSON-RPC Protocol

Introduction

JSON-RPC (JavaScript Object Notation Remote Procedure Call) is a remote procedure call protocol. It uses JSON as the data format and can communicate between clients and servers through transport protocols such as TCP Socket, HTTP, and WebSocket.

JSON-RPC has the following characteristics:

- **Simplicity:** Use JSON as the data exchange format, which is concise, easy to understand, and easy to parse.
- **Lightweight:** Use a text-based JSON format for data exchange, with simple data and protocol structures, bringing small data transmission size, fast speed, and high performance.
- **Cross-language and cross-platform:** The JSON-RPC protocol supports communication between multiple programming languages (such as C++ and Python) and different platforms.
- **Multiple transport protocols supported:** The JSON-RPC protocol can use various transport protocols for data transmission, such as HTTP, TCP Socket and WebSocket, making it suitable for various network environments.
- **Error handling:** The JSON-RPC protocol defines the specifications for error objects and error codes, allowing the client to perform debugging and handle issues based on the error information returned from the server.

The JSON-RPC protocol implements remote procedure calls between the client and the server by the request-response method. The client sends a JSON formatted request to the server, for example `{"jsonrpc":"2.0","method":"getRobotNames","params":[],"id":1}`. After receiving the request, the server parses the JSON object and extracts the values of fields such as "method", "params", and "id". The server executes the method corresponding to "method" and uses the provided parameters. Based on the execution result, the server generates a response object, serializes the response object into JSON format, for example `{"id":1,"jsonrpc":"2.0","result":["rob1"]}`, and then sends the JSON formatted response to the client. Finally, the client receives the response transmitted from the server.

Supported Transport Protocols and Port Numbers

The RPC function module in the AUBO SDK supports TCP Socket, HTTP, and WebSocket transport protocols, with the corresponding port numbers as follows:

Protocol	Port number
TCP Socket	30004
HTTP	9012
WebSocket	9012

JSON-RPC 2.0 Specifications

Request Object

The RPC request object contains the following members:

- **jsonrpc**: The version number of the JSON-RPC protocol, being a fixed value of 2.0
- **method**: The name of the method being called, being a string
- **params**: The parameter value, being a structured value
- **id**: Identifier

Response Object

Success and failure of RPC calls are described below:

1. When the RPC call is successful, the RPC response object contains the following members:
 - **jsonrpc**: The version number of the JSON-RPC protocol, being a fixed value of 2.0
 - **result**: The return value of the executed function
 - **id**: Identifier, which must be the same as the identifier of the request object
2. When an error occurs during the RPC call, the RPC response object contains the following members:
 - **jsonrpc**: The version number of the JSON-RPC protocol, being a fixed value of 2.0
 - **error**: The error object, which contains the following members

- code means the error code
- message means a brief error description
- data means additional information about the error, which may be ignored.
- id: Identifier, which must be the same as the identifier of the request object

The error codes of the error object are shown in the table below:

Error code	Message	Description
-32700	Parse error.	The server receives invalid JSON. An error occurs while the server was parsing the JSON text.
-32600	Invalid Request.	The sent JSON is not a valid request object.
-32601	Method not found.	The method does not exist/unavailable.
-32602	Invalid params.	Invalid method parameters.
-32603	Internal error.	JSON-RPC internal error.
-32099..-32000	Server error.	It is reserved for implementation-defined server errors.

Example

An example of websocket protocol transmission written in Python is as follows:

localhost is the IP address of the robot

Python

```
import json
import websocket

# WebSocket Server Address
server_address = "ws://localhost:9012"

# Create the WebSocket Connection
ws = websocket.create_connection(server_address)

# Construct the JSON-RPC Request
request = {
```

```
"jsonrpc": "2.0",
"method": "getRobotNames",
"params": [],
"id": 1
}

# Convert the Request into a JSON String
request_json = json.dumps(request)

# Send the Request
ws.send(request_json)

# Receive the Response
response_json = ws.recv()

# Parse the Response
response = json.loads(response_json)

print("Received Response:", response)

# Close the WebSocket Connection
ws.close()
```

Program running result:

Python

```
Received Response: {'id': 1, 'jsonrpc': '2.0', 'result': ['rob1']}
```

Introduction to RTDE Protocol

Introduction

The RTDE (Real-Time Data Exchange) protocol allows for real-time data exchange between the robot controller and external devices. Through the RTDE protocol, external devices can obtain the robot's status information, such as joint angle, end-of-arm tooling's position and attitude, movement speed, and acceleration. In addition, external devices can send control commands to the robot via the RTDE protocol, for example, moving the robot to a specified position by writing codes to the register.

The RTDE protocol has the following characteristics:

- **Real-time:** When the robot controller and external devices exchange data through the RTDE protocol, the data transmission is fast and does not compromise the real-time performance of the robot controller.
 - **Bi-directional communication:** The RTDE protocol supports bi-directional data exchange between the robot controller and external devices, that is, external devices can not only receive robot data, but also send commands.
 - **Cross-language and cross-platform:** The RTDE protocol supports communication between multiple programming languages (such as C++ and Python) and different platforms.
 - **Multiple transport protocols supported:** The RTDE protocol can use various transport protocols for data transmission, such as TCP Socket and WebSocket, making it suitable for various network environments.
-

Supported Transport Protocols and Port Numbers

The RTDE function module in the AUBO SDK supports TCP Socket and WebSocket transport protocols, with the corresponding port numbers as follows:

Protocol	Port number
TCP Socket	30010
WebSocket	9013

RTDE Protocol Specifications

RTDE Menu

RTDE Menu is a menu that defines the content and format of the RTDE data synchronization packet; it is a specification. It describes the input and output fields that can be synchronized and their data types. The RTDE Menu includes the input menu and output menu. The input menu is for the client to push data to the server, while the output menu is for the server to push data to the client. To know the list of topics and data types in the RTDE Menu, please refer to [RTDE Menu](#).

Request-Response

The client can subscribe to topics, publish topics, and unsubscribe from topics to the server.

Subscribe to Topics

When the client subscribes to a topic, it sends a data packet to the server. The first element of the data packet is the index value, fixed at 100, and the second element is an object that contains the following members: channel, frequency, segments, to_server, and trigger. Among them, the value of to_server is fixed as false, and segments are the members of the output menu in the [RTDE Menu](#). For the definitions of these members, refer to the following RtdeRecipe structure:

```

struct RtdeRecipe
{
    bool to_server;           ///< Input/Output, to_server=false means
    int chanel;               ///< Channel The channel number only supp
    double frequency;         ///< Update frequency
    int trigger;              ///< Trigger method (this feature is not y
    std::vector<std::string> segments; ///< Field list, indicating the topics tha
};
  
```

After that, the client will receive the data packet in response from the server. The first element of the data packet is the channel, which is the same value as the channel in subscription. The second and subsequent elements are the data of the topic in subscription.

For example, if you want to subscribe to the topic "R1_actual_q" on channel 0, the data packet sent by the client to the server is as follows:

bash

```
[100, {"chanel":0, "frequency":50.0, "segments":["R1_actual_q"], "to_server":false, "t
```

The response of the server received by the client is as follows:

bash

```
[0, [-0.027531569059195515, -0.28143723119618147, 1.6961810563264004, 0.4177235477634
```

In the above example, the first element in the data packet means channel 0, and the second element is the value of the topic "R1_actual_q" in subscription, which represents the actual values of the six joint angles.

Unsubscribe from Topics

After subscribing to a topic, the client will continuously receive data sent from the server. To cancel the push, the client can unsubscribe from the topic. The data packet format for unsubscribing from a topic is the same as that for subscribing to a topic, but frequency shall be set to 0, channel shall be set to the channel of the topic to be canceled, and segments shall be empty.

For example, to unsubscribe from the topic on channel 0,

bash

```
[100, {"chanel":0, "frequency":0.0, "segments":[], "to_server":false, "trigger":0}]
```

Publish Topics

When the client publishes a topic, it sends a data packet to the server. The first element of the data packet is the index value, fixed at 100, and the second element is an object that contains the following members: channel, frequency, segments, to_server, and trigger. Among them, the value of to_server is fixed as true, and segments are the members of the input menu in the [RTDE Menu](#). For the definitions of these members, refer to the following RtdeRecipe structure:

C++

```
struct RtdeRecipe
{
    bool to_server;           ///< Input/Output, to_server=false means
    int chanel;               ///< Channel The channel number only sup
    double frequency;         ///< Update frequency
    int trigger;              ///< Trigger method (this feature is not y
```

```
std::vector<std::string> segments; ///< Field list, indicating the topics tha
};
```

Then, the client pushes data to the server. The first element of the data packet is the channel, the same value as the channel for publishing the topic. The second and subsequent elements are the data of the published topic.

For example, to publish the topic "input_float_registers_0" and "input_double_registers_1" on channel 1, the data packet sent from the client to the server would be:

```
[100, {"chanel":1, "frequency":1.0, "segments":["input_float_registers_0", "input_dou
```

bash

Then, the client pushes data to the server:

```
[1, 3.1, 4.1]
```

bash

In the above example, the first element in the data packet represents channel 1, and the second and third elements represent the values of "input_float_registers_0" and "input_double_registers_1", respectively.

To check if the data is successfully pushed, you can subscribe to the topics "input_float_registers_r0" and "input_double_registers_r1":

```
[100, {"chanel":1, "frequency":1.0, "segments":["input_float_registers_r0", "input_do
```

bash

If the data is successfully pushed, the response received by the client from the server will be:

```
[1, 3.09999999046325684, 4.1]
```

bash

Example

An example of websocket protocol transmission written in Python is as follows:

localhost is the IP address of the robot

```
import asyncio
import websockets

async def rtde_communication():
    uri = "ws://localhost:9013"
    async with websockets.connect(uri) as websocket:
        # Publish Topics
        publish_topic = '[100,{"chanel":1,"frequency":1.0,"segments":["input_floa'
                                '"input_double_registers_1"],"to_server":true,"trigger":0
        await websocket.send(publish_topic)

        # Push Data to Server
        publish_data = '[1,3.3,4.4]'
        await websocket.send(publish_data)

        # Subscribe to Topics
        subscribe_topic = '[100,{"chanel":1,"frequency":50.0,"segments":["input_f'
                                '"input_double_registers_r1"],"to_server":false,"trigge
        await websocket.send(subscribe_topic)

        # Receive the RTDE Data Packet
        while True:
            data_packet = await websocket.recv()
            print("Received data packet:", data_packet)

asyncio.get_event_loop().run_until_complete(rtde_communication())
```

Program running result:

```
Received data packet: [1,3.299999952316284,4.4]
Received data packet: [1,3.299999952316284,4.4]
Received data packet: [1,3.299999952316284,4.4]
Received data packet: [1,3.299999952316284,4.4]
Received data packet: [1,3.299999952316284,4.4]
Received data packet: [1,3.299999952316284,4.4]
Received data packet: [1,3.299999952316284,4.4]
```

Introduction to Script Protocol

Introduction

The script protocol supports external devices in sending script strings to the robot's controller, which will load the script and execute the commands.

Supported Transport Protocols and Port Numbers

The SCRIPT function module in the AUBO SDK supports TCP Socket, HTTP, and WebSocket transport protocols, with the corresponding port numbers as follows:

Protocol	Port number
TCP Socket	30002
HTTP	30003
WebSocket	30003

Script Protocol Specification

Script strings sent from external devices to the robot controller via **TCP Socket protocol** must end with `\r\n\r\n`.

Example of Sending Script Strings

The following introduces examples of sending scripts via **TCP Socket**, **HTTP**, and **WebSocket** communication. The script string sent in the example is as follows:

Note: This script string is written for the **i5 model** robot arm.

```
return function(api)
  local _ENV = require('aubo').sched.select_robot(1)
  home = { -0.000003, -0.127267, -1.321124, 0.37694, -1.570796, -0.000008 }
  move1 = { -0.400318, 0.064315, 0.547598, 3.14, 0.0, -2.63782 }
  move2 = { -0.400318, 0.064315, 0.379989, 3.14, 0.0, -2.63782 }
```

```

move3 = { -0.400318, 0.064315, 0.547598, 3.14, 0.0, -2.63782 }
move4 = { -0.400319, -0.298610, 0.547598, 3.14, 0.435471, -1.57 }
move5 = { -0.400319, -0.243865, 0.429931, 3.14, 0.435471, -1.57 }
move6 = { -0.400319, -0.298610, 0.547598, 3.14, 0.435471, -1.57 }
-- Set the Tool Center Point (TCP offset relative to the flange center)
setTcpOffset({0,0,0,0,0,0})
-- Set the speed ratio
setSpeedFraction(0.75)
-- Joint movement
moveJoint(home, 3.14, 3.14, 0.0, 0)
-- Linear movement
moveLine(move1, 2, 1, 0.0, 0)
moveLine(move2, 2, 1, 0.0, 0)
moveLine(move3, 2, 1, 0.0, 0)
moveLine(move4, 2, 1, 0.02, 0)
moveLine(move5, 2, 1, 0.0, 0)
moveLine(move6, 2, 1, 0.02, 0)
end

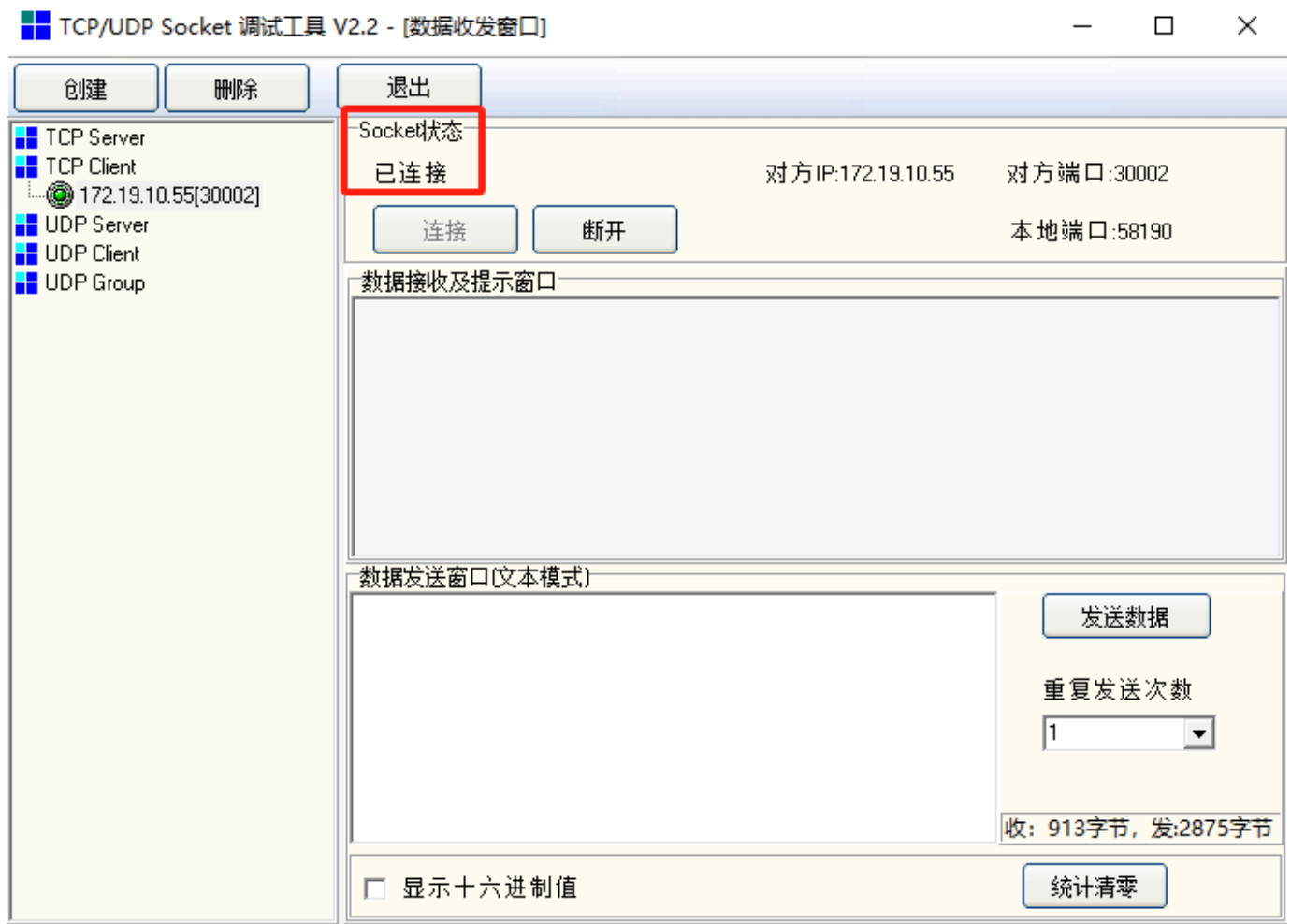
```

Send Scripts Based on TCP Socket Communication

The **IP address** for sending scripts via TCP Socket is the robot's IP, and the **port number** is **30002**.

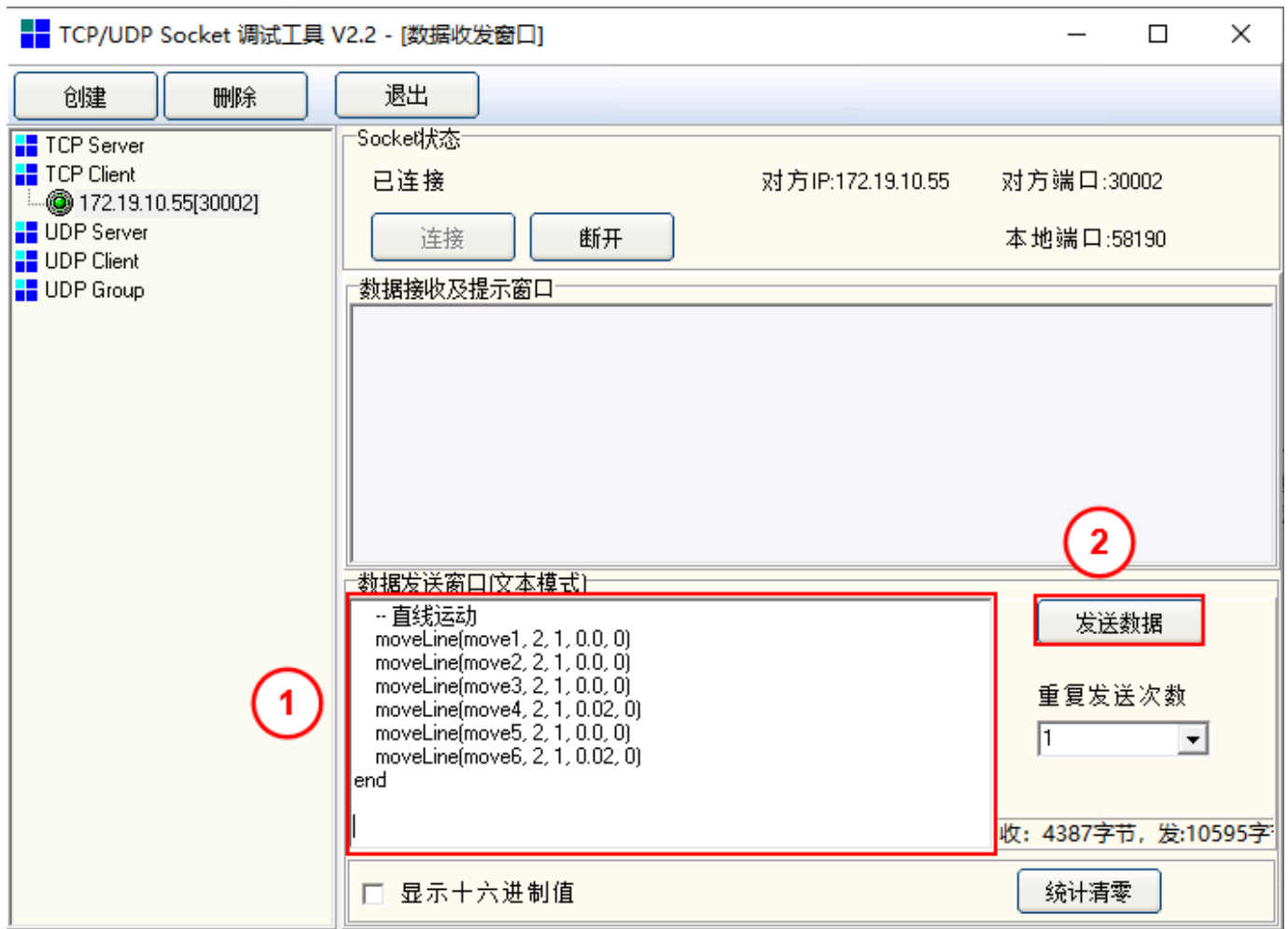
The operation steps for sending script strings via the **TCP Debug Assistant** are described briefly below:

1. After entering the robot's IP address and port number, establish a connection. Once connected successfully, the Socket status shows connected.



2. Enter the script string, press the Enter key twice at the end of the script, and click "Send Data".

Note: The script string sent to the robot controller via the TCP Socket protocol must end with `\r\n\r\n`. `\r\n` represents the carriage return and line feed character. Therefore, after pasting the script into the TCP debugging assistant, it is necessary to press the Enter key at least twice at the end.



3. After successful sending, the received data will be displayed.

Send Scripts Based on WebSocket Communication

The **URL address** for sending scripts via WebSocket is **ws://localhost:30003**. Here, localhost represents the robot's IP address, and 30003 is the port number for WebSocket.

The operation steps for sending script strings via the **web-based WebSocket testing tool** are described briefly below:

1. WebSocket online testing tool: [EasySwoole-WebSocket Online Testing Tool]
(<http://www.easyswoole.com/wstool.html>)
2. After entering the URL address, establish a connection. Once connected successfully, the server configuration status shows that the connection is successful.

WEBSOCKET 在线测试工具

服务器配置 状态: 连接成功

服务地址 ws://172.19.10.55:30003

关闭连接

发包设置

每隔

1

秒发送内容

PING

开始发送

需要发送到服务端的内容

☐ 发包清空输入

发送到服务端

调试消息

3. Enter the script string and click "Send to Server".

WEBSOCKET 在线测试工具

服务器配置 状态: 连接成功

服务地址 ws://172.19.10.55:30003

关闭连接

发包设置

每隔

1

秒发送内容

PING

开始发送

```
moveLine(move6, 2, 1, 0.02, 0)
end
```

☐ 发包清空输入

发送到服务端

4. After successful sending, the received message will be displayed.

服务器配置 状态: 连接成功

服务地址 ws://172.19.10.55:30003

关闭连接

发包设置

每隔 1 秒发送内容 PING

开始发送

```
moveLine(move6, 2, 1, 0.02, 0)
end
```

☐ 发包清空输入

发送到服务端

调试消息

消息记录 清空消息

☐ 收包清空记录 ☐ 收包JSON解码 ☐ 暂停接收

```
setTcpOffset({0,0,0,0,0,0}) — 设置速度比例 setSpeedFraction(0.75) — 关节运动
moveJoint(home, 3.14, 3.14, 0.0, 0) — 直线运动 moveLine(move1, 2, 1, 0.0, 0)
moveLine(move2, 2, 1, 0.0, 0) moveLine(move3, 2, 1, 0.0, 0) moveLine(move4, 2,
1, 0.02, 0) moveLine(move5, 2, 1, 0.0, 0) moveLine(move6, 2, 1, 0.02, 0) end
```

收到消息 15:03:20

```
{"name":"home","event":"updateVariable","type":"table","value":[-3e-06,-0.127267,-1.321124,0.37694,-1.570796,-8e-06]}
```

收到消息 15:03:20

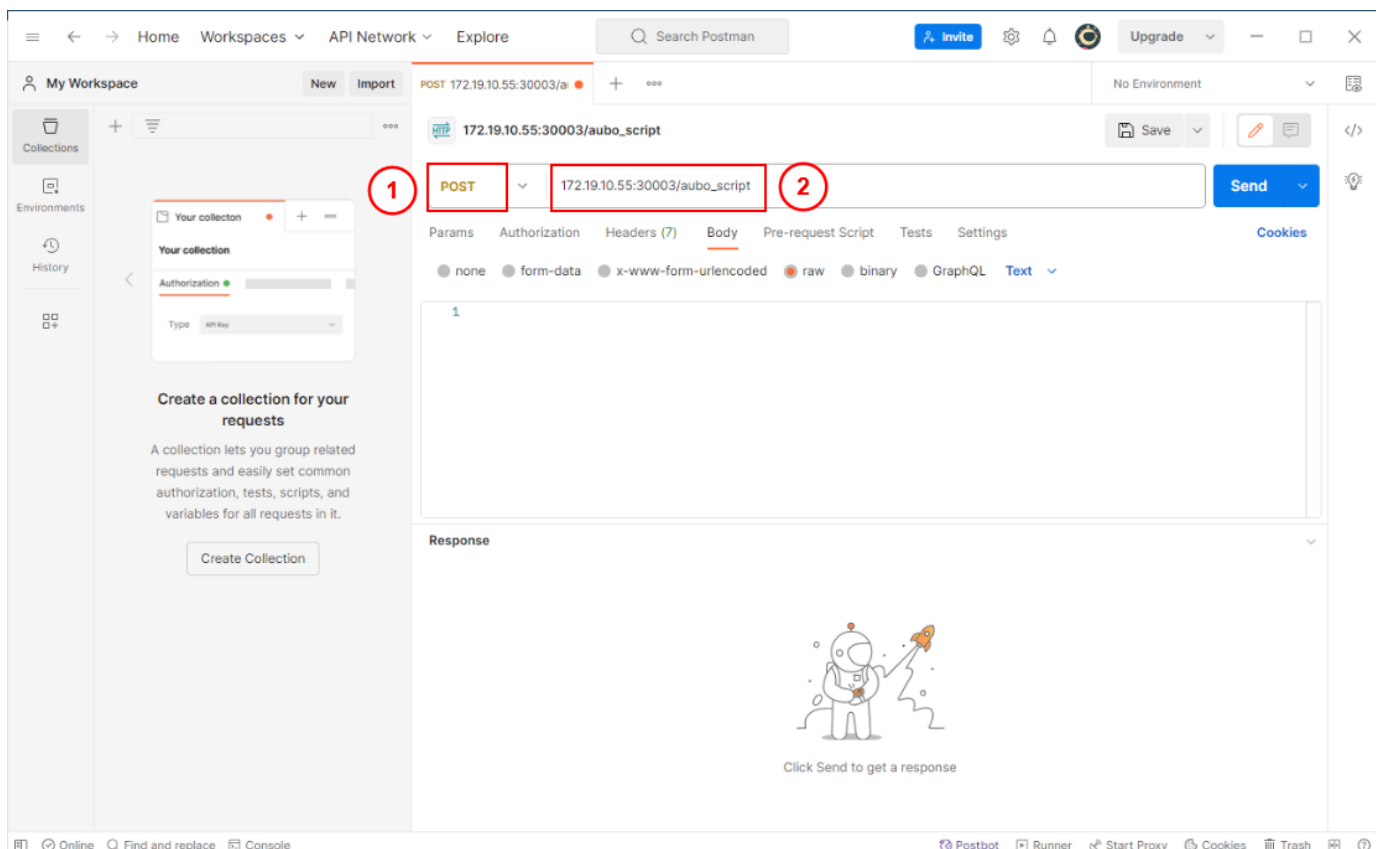
```
{"name":"move1","event":"updateVariable","type":"table","value":[-0.400318,0.064315,0.547598,3.14,0.0,-2.63782]}
```

Sending Script Based on HTTP Communication Example

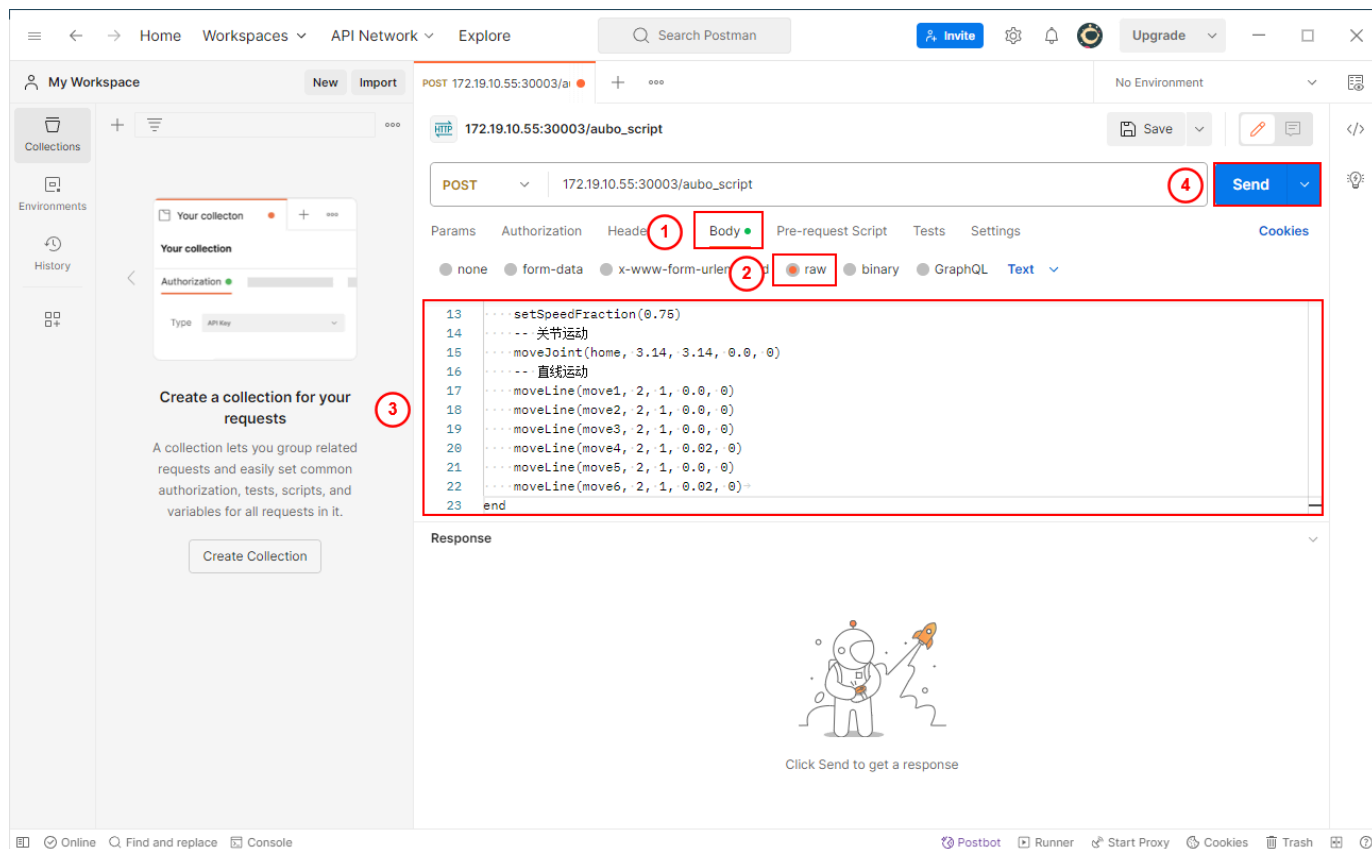
The **URL address** for sending the script via HTTP is **localhost:30003/aubo_script**. Here, localhost represents the robot's IP address, and 30003 is the port number for HTTP.

The operation steps for sending script strings via the **Postman** are described briefly below:

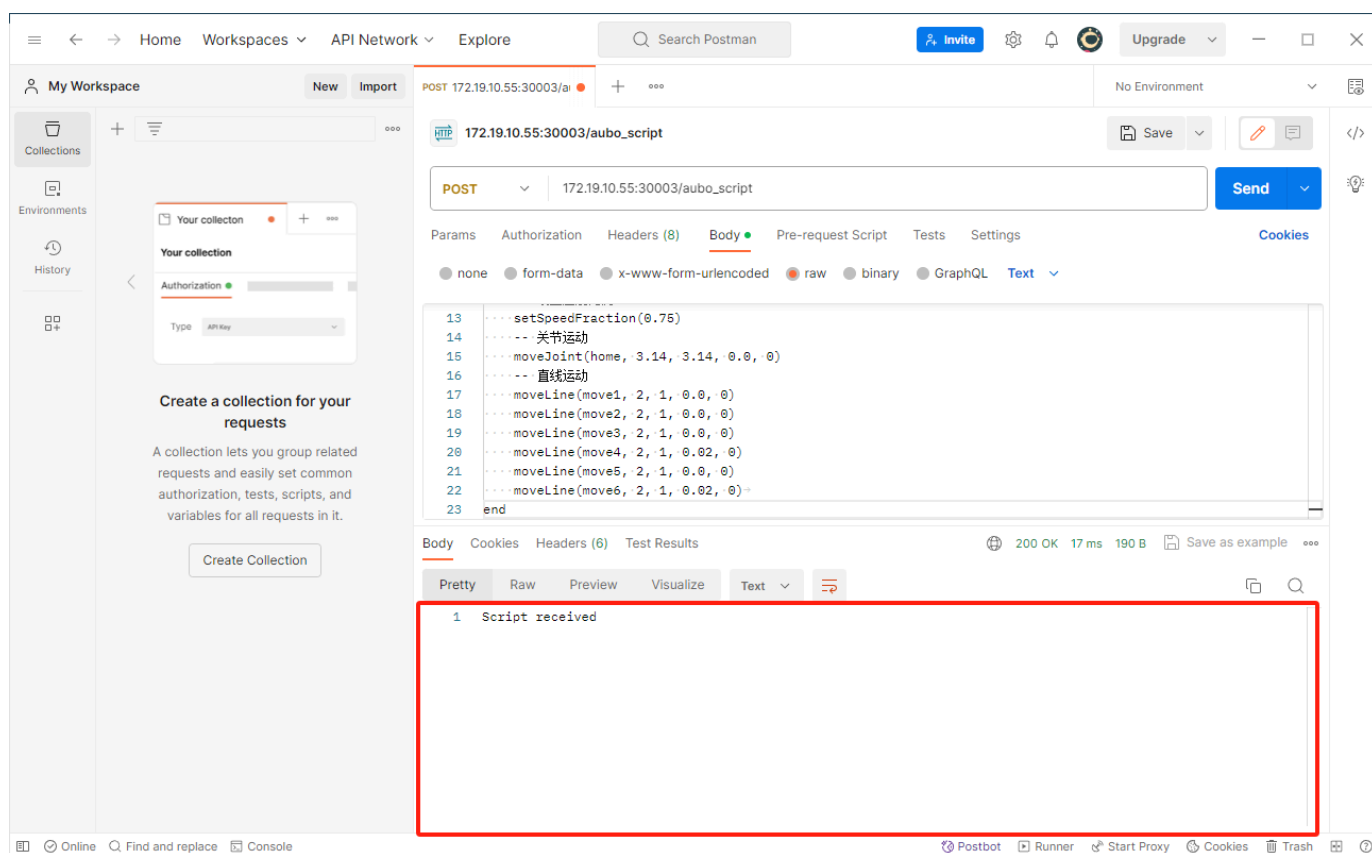
1. Open Postman, select the request action as **POST**, and enter the URL address.



2. In the Body section, enter the script string and click "Send".



3. After successful sending, the received message will be displayed.



SDK Development

The AUBO SDK provides header files, library files, and numerous code examples to help developers quickly develop applications. The AUBO SDK contains the following folders:

- include folder: Header files, which include the API interfaces, data types, and macro definitions provided for users.
 - lib folder: Library files used during compilation.
 - share folder: Code examples and error code descriptions.
-

How to Use

To use the SDK, generally, it is necessary to follow the steps below:

1. [Configure the development environment](#)
2. [Download and install the SDK](#)
3. Create a new project in the IDE.
4. Add the library files and header files provided in the AUBO SDK to the project.
5. Write codes.
6. Connect the Aubo robot or [Aubo Sim virtual machine](#).
7. Compile and run the codes.

The specific steps may vary depending on the project. Please make adjustments according to the actual situation.

Requirements for Running Environment

C++ SDK

- Supported operating systems: Linux 64-bit and Windows 64-bit
- Supported hardware platforms: x86 and ARM embedded platforms
- Requirements for compiler:
 - Linux: G++-7/GCC-7
 - Windows: MSVC 2019 and above

Python SDK

- Supported operating systems: Linux 64-bit and Windows 64-bit
- Supported hardware platforms: x86 and ARM embedded platforms
- Requirements for Python version:
 - Linux x86: Python 3.6 ~ Python 3.11
 - Linux ARM: Python 3.6
 - Windows: Python 3.6 ~ Python 3.11

Java SDK

- Supported operating systems: Linux 64-bit and Windows 64-bit
- Supported hardware platforms: x86 and ARM embedded platforms
- Linux and Windows:
 - JDK 1.8 (Java 8) or above
 - Maven 3.6 or above

Download and Installation

Prerequisites

Before installing the Java SDK, please ensure the runtime environment is correctly configured. For more information on environment configuration, please refer to [AUBO SDK Running Requirements](#).

SDK

Download link for the SDK: [Index of /sdk \(aubo-robotics.cn\)](#).

The SDK in this link includes C++ header files and C++ SDK library files, which can be used for secondary development of the C++ SDK.

pyaubo_sdk library file

To use the Python SDK, it is necessary to download and install the pyaubo_sdk library file.

Download and Installation

- Method 1: Download the pyaubo_sdk from the official website of PyPI
 - Open the PyPI website: [PyPI • The Python Package Index] (<https://pypi.org/>).
 - Enter and search for the pyaubo-sdk.



- Download the pyaubo-sdk.

- Click on "Download files", find the .whl file of the corresponding version in the Built Distributions section, and click to download.

The screenshot shows the PyPI page for the `pyaubo_sdk` package. On the left sidebar, the 'Download files' button is highlighted with a red box and a red circle with the number 1. The main content area shows the 'Download files' section with a link to 'installing packages'. Below this is the 'Source Distributions' section, which states that no source distribution files are available. The 'Built Distributions' section is highlighted with a red box and a red circle with the number 2. It lists various wheel files for different Python versions and platforms, including Windows and Linux. The bottom of the page shows the 'Verified details' section, which includes the maintainer 'aubo-robotics' and the license 'MIT License'.

- After the download is completed, switch to the directory where the .whl file is located and install the `pyaubo-sdk` using the `pip` command. For example, to install the `pyaubo_sdk` library for the 64-bit Python version 3.11 on the Windows system, enter the command below.

```
pip install pyaubo_sdk-0.22.2-cp311-cp311-win_amd64.whl
```

- After successful installation, the interface will look like the figure below.

```
E:\>pip install pyaubo_sdk-0.22.2-cp311-cp311-win_amd64.whl
Processing e:\pyaubo_sdk-0.22.2-cp311-cp311-win_amd64.whl
Installing collected packages: pyaubo-sdk
Successfully installed pyaubo-sdk-0.22.2
```

- Method 2: Download and install the pyaubo_sdk using the pip command

- Install the command

```
pip install pyaubo_sdk
```

- Install the specified version

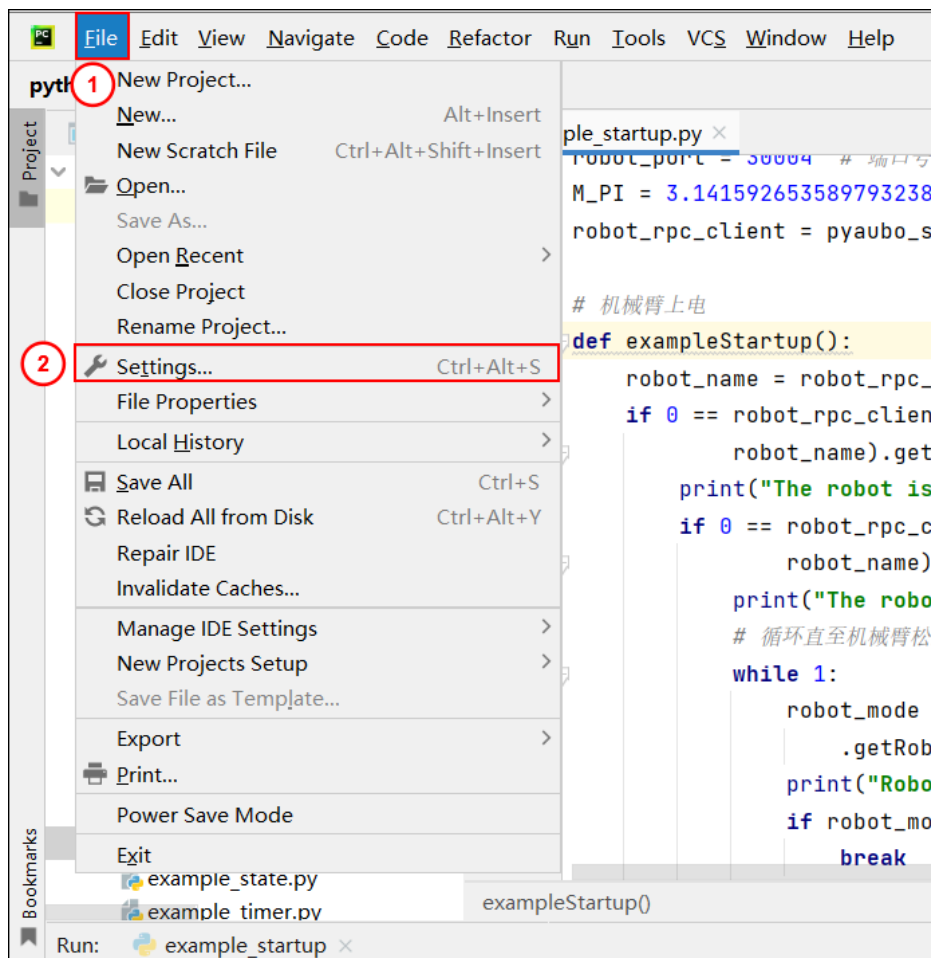
```
pip install pyaubo_sdk==x.x.x
```

For example, download and install version 0.22.2

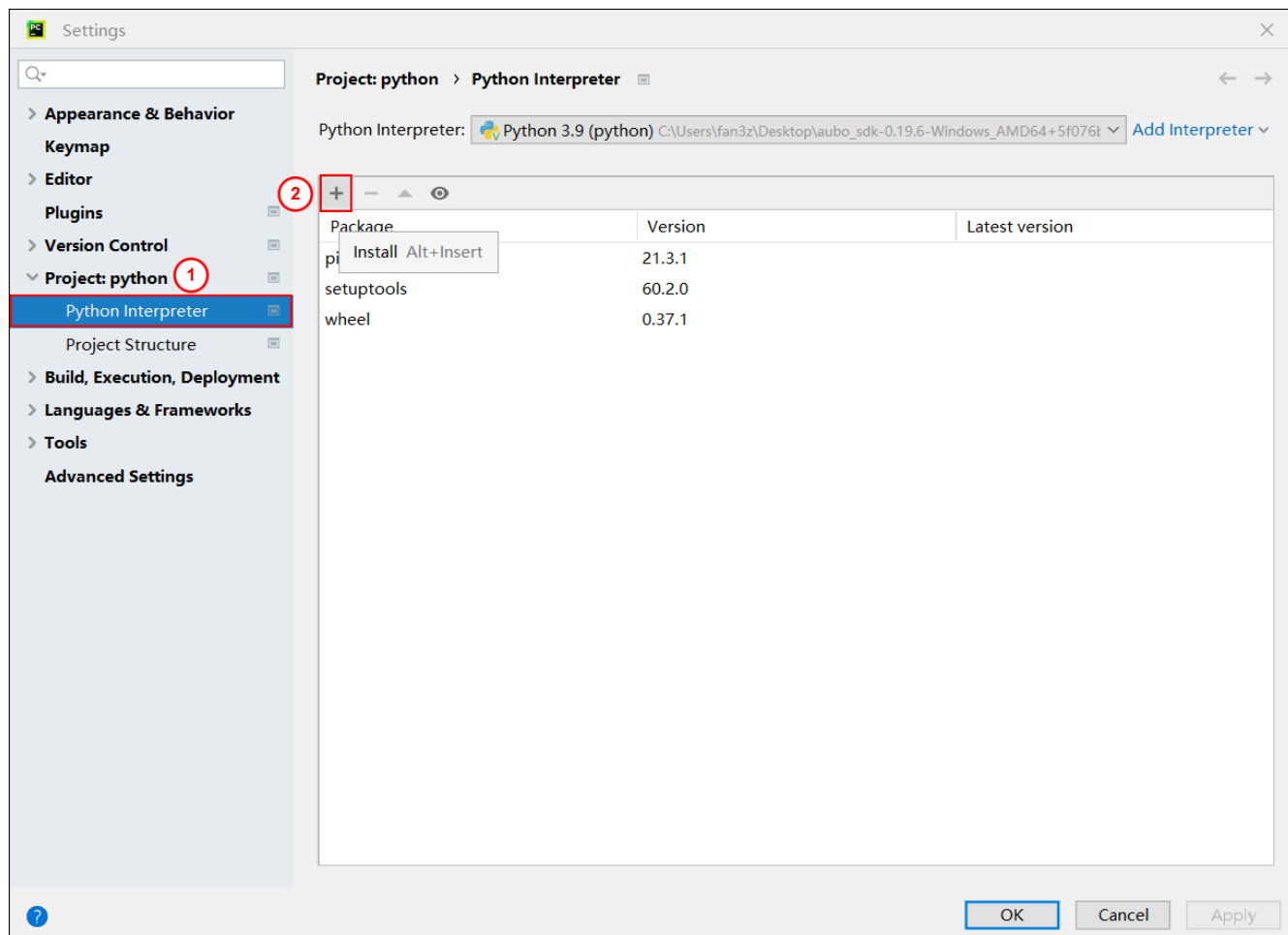
```
pip install pyaubo_sdk==0.22.2
```

- Method 3: Download and install the pyaubo_sdk via PyCharm

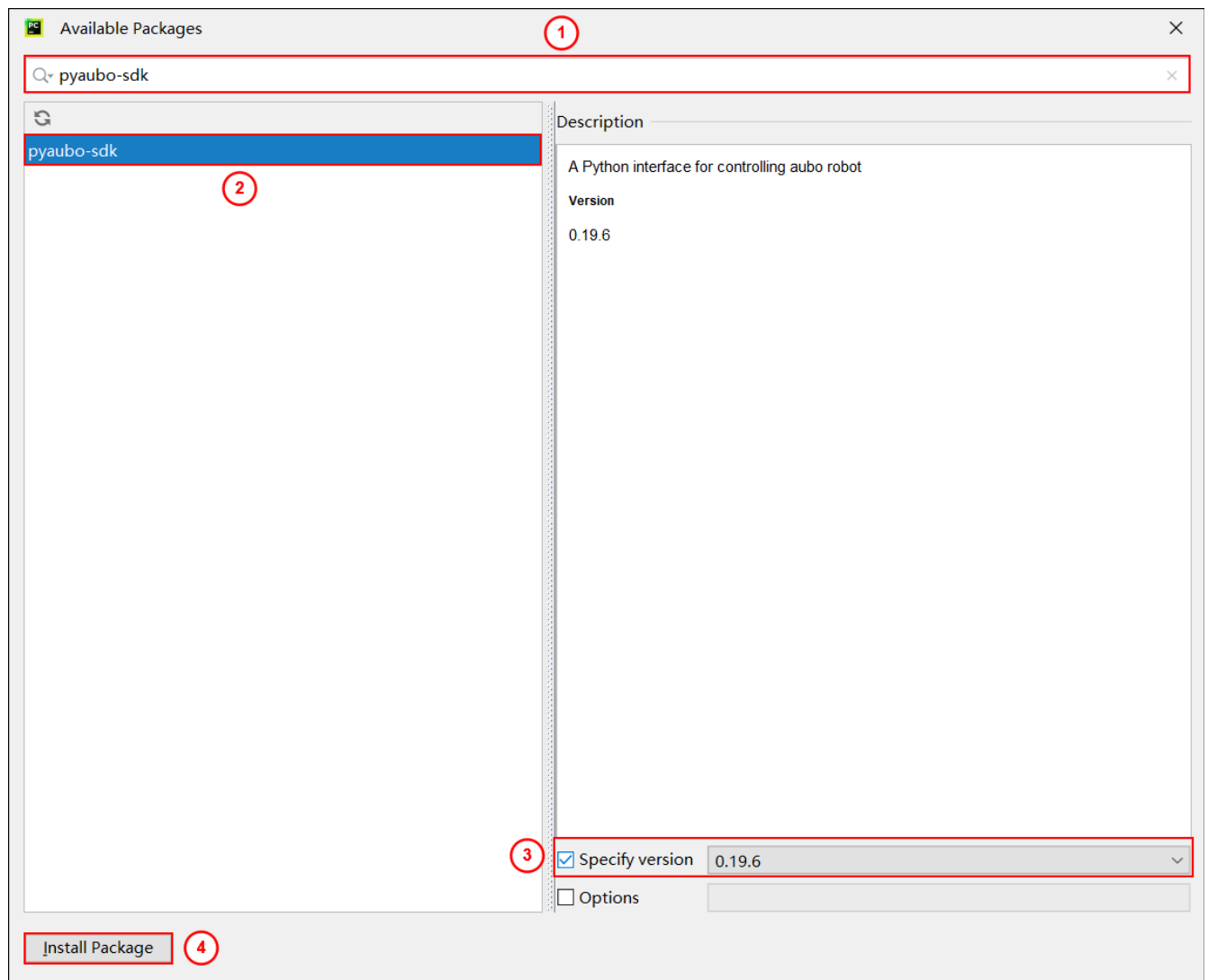
1. File -> Settings



2. On the Python Interpreter page, click the "+" icon.



3. Search for the pyaubo-sdk, select the version, and click "Install Package".



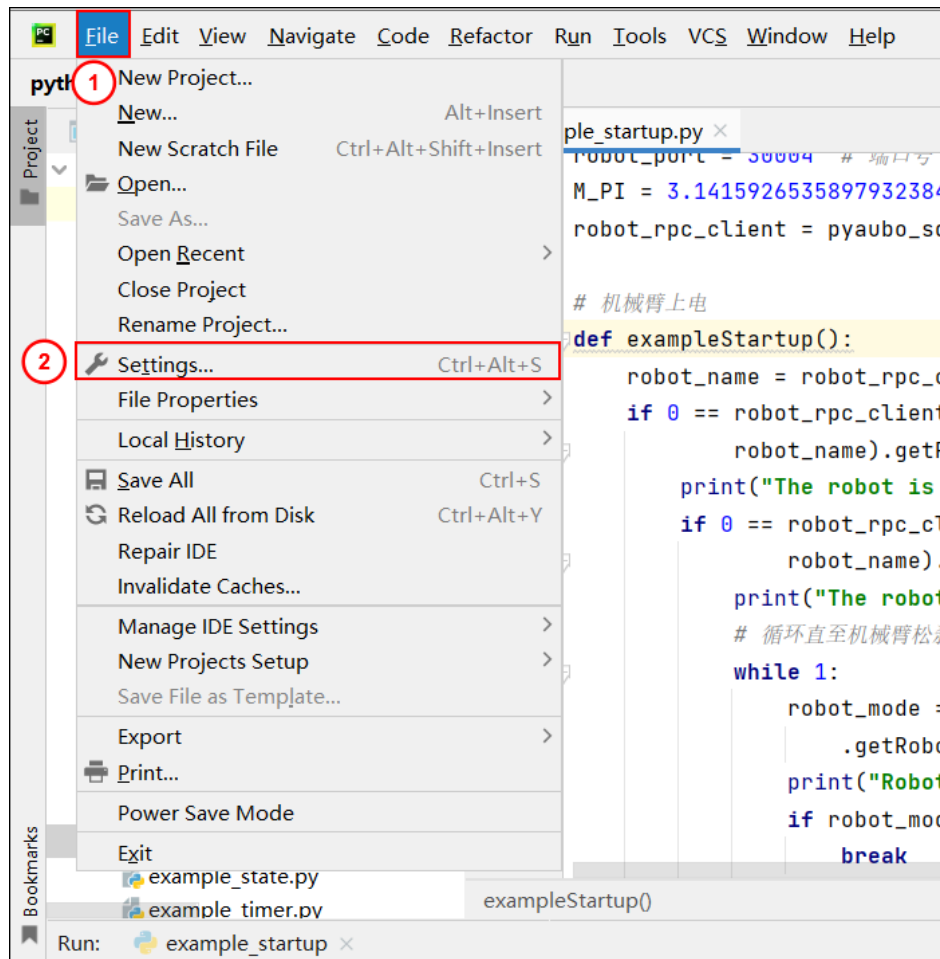
Uninstallation

Method 1: Uninstall the pyaubo-sdk using the pip command

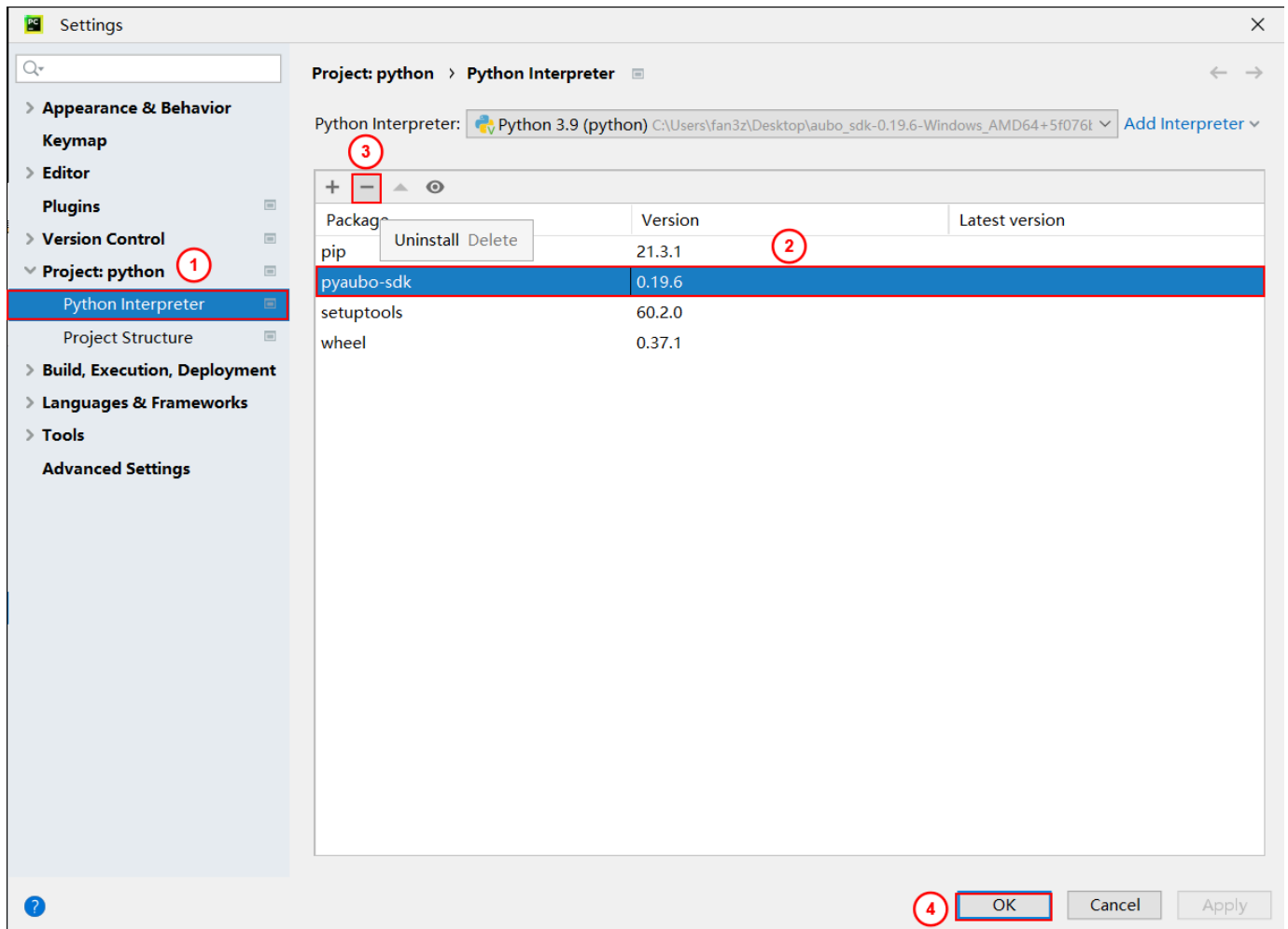
```
pip uninstall pyaubo_sdk
```

Method 2: Uninstall the pyaubo-sdk via PyCharm

1. File -> Settings



2. In the Python Interpreter interface, select the pyaubo-sdk, click the "—" icon to start uninstallation. After the uninstallation is completed, click "OK".



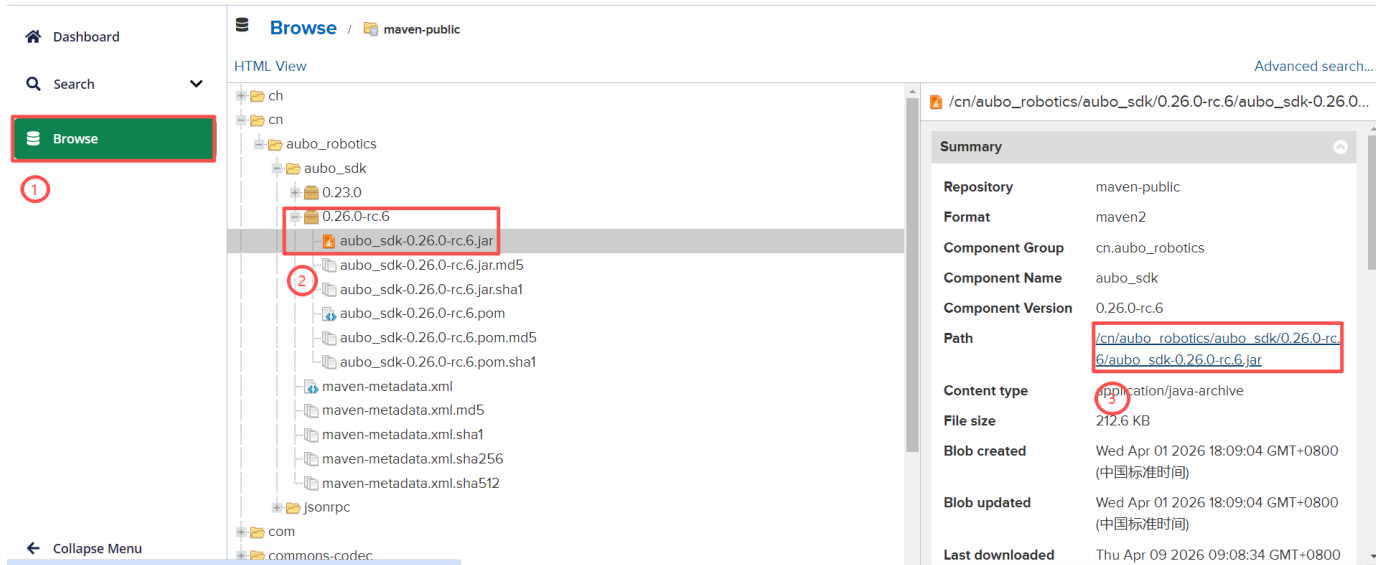
Java SDK Development Package

Java SDK installation can be done in two ways: **local installation** and **remote dependency installation**.

Local Installation

Obtaining the SDK

- Java SDK download link: Download the latest `aubo_sdk-x.x.x.jar` file from the official mirror site [aubo sdk \(aubo-robotics.cn\)](http://aubo-sdk(aubo-robotics.cn)).
- Current version: `aubo_sdk-0.26.0-rc.6.jar`.



SDK Installation

1. Open the command line terminal in the directory where the **JAR** file is located.
2. Run the following command:

Note:

Replace `aubo_sdk-0.26.0-rc.6.jar` with the corresponding SDK version.

```
mvn install:install-file -Dfile=aubo_sdk-0.26.0-rc.6.jar -DgroupId=cn.aubo_robotics
```

3. When **BUILD SUCCESS** appears, the installation is complete.

```

Microsoft Windows [version 10.0.22631.4169]
(c) Microsoft Corporation. All rights reserved.

D:\netdisk\edgedownload>mvn install:install-file -Dfile=aubo_sdk-0.26.0-rc.6.jar
-DgroupId=cn.aubo_robotics -DartifactId=aubo_sdk -Dversion=0.26.0-rc.6 -Dpackag
ing=jar
[INFO] Scanning for projects...
[INFO]
[INFO] -----< org.apache.maven:standalone-pom >-----
[INFO] Building Maven Stub Project (No POM) 1
[INFO] -----[ pom ]-----
[INFO]
[INFO] --- install:3.1.4:install-file (default-cli) @ standalone-pom ---
[INFO] Installing D:\netdisk\edgedownload\aubo_sdk-0.26.0-rc.6.jar to C:\Users\A
dministrator\.m2\repository\cn\aubo_robotics\aubo_sdk\0.26.0-rc.6\aubo_sdk-0.26.
0-rc.6.jar
[INFO] Installing C:\Users\Administrator\AppData\Local\Temp\aubo_sdk-0.26.0-rc.6
2200293260427101485.pom to C:\Users\Administrator\.m2\repository\cn\aubo_robotic
s\aubo_sdk\0.26.0-rc.6\aubo_sdk-0.26.0-rc.6.pom
[INFO]
[INFO] BUILD SUCCESS
[INFO]
[INFO] Total time: 0.596 s
[INFO] Finished at: 2026-04-09T18:03:27+08:00
[INFO]
D:\netdisk\edgedownload>

```

豆包AI生成

Remote Dependency Installation

Maven Remote Dependency Integration

Adding Dependencies

Add the following dependency to the project configuration file `pom.xml` :

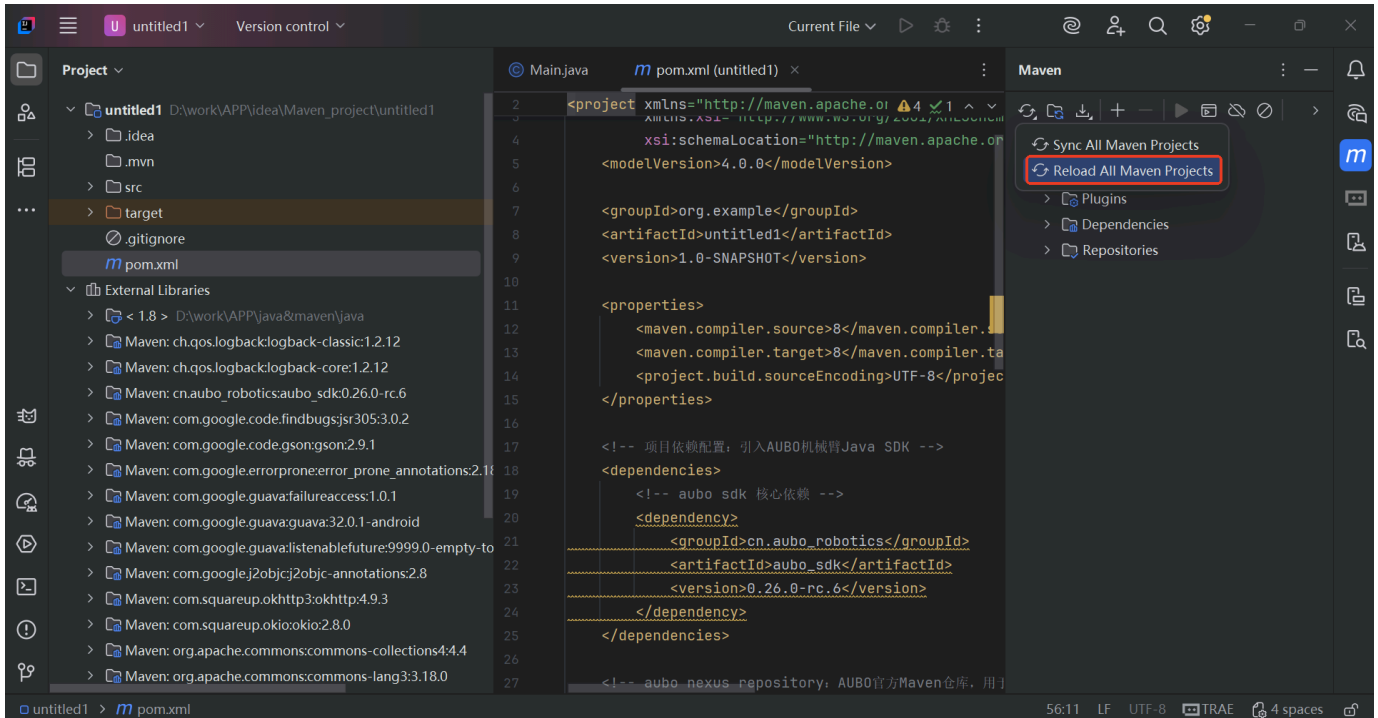
Note:

Replace `<version>0.26.0-rc.6</version>` with the target SDK version.

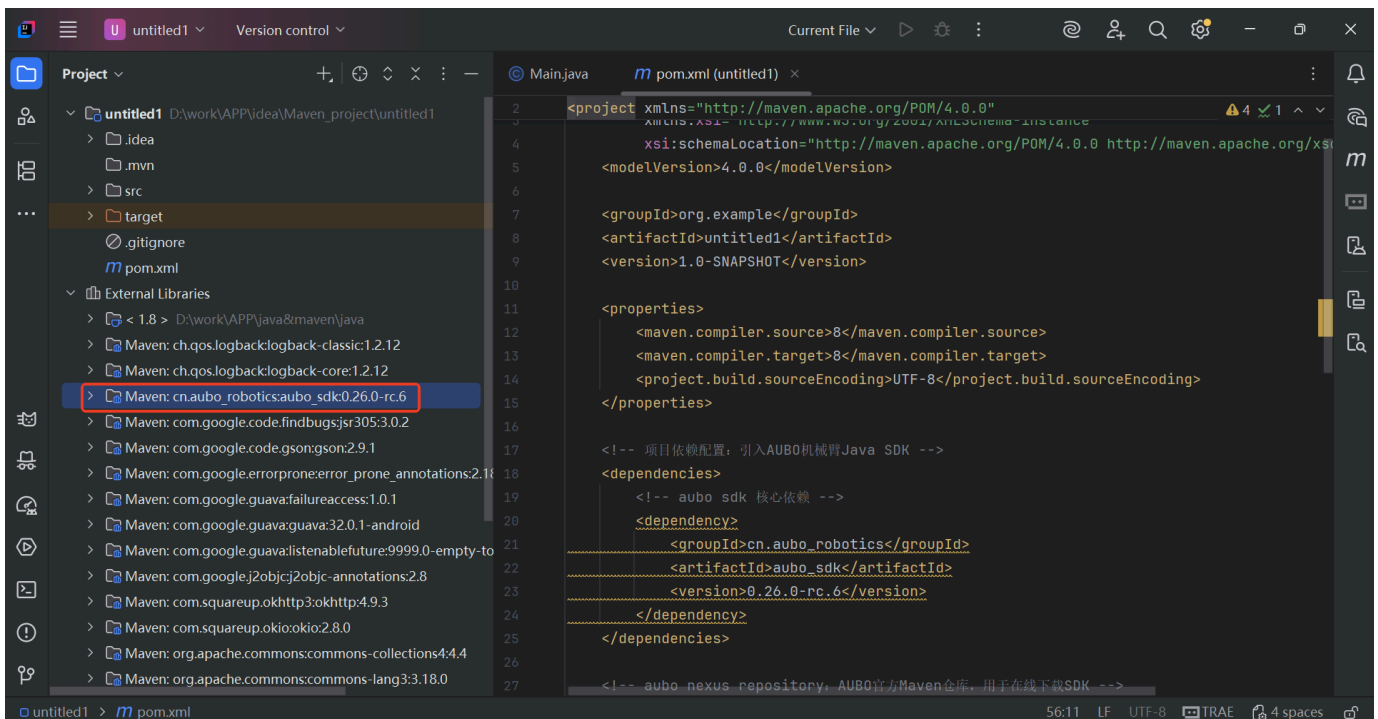
► Click to view dependencies:

IDE Refresh and Synchronization

1. In IntelliJ IDEA, click [Reload All Maven Projects].



2. When `aubo_sdk-0.26.0-rc.6.jar` appears in External Libraries, the installation is successful.



Android Project Repository Configuration

This section demonstrates how to import and verify AUBO Java SDK for Android projects using Android Studio.

Note:

For Android projects, it is recommended to integrate the SDK via **Gradle + Nexus Maven** to easily synchronize the SDK and its related dependencies for unified version management.

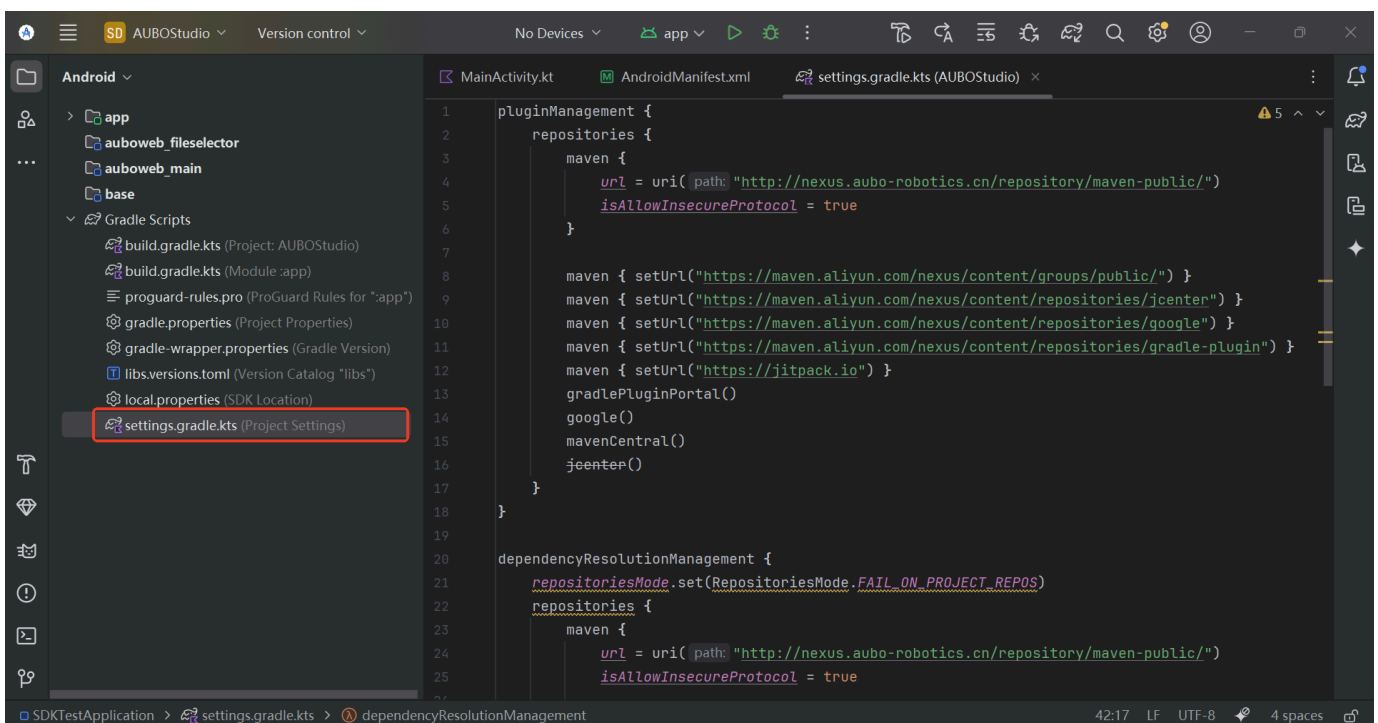
Adding Dependencies

1. Configure `pluginManagement` and `dependencyResolutionManagement` in the Android project's `settings.gradle.kts`.

Note

If the project already has relevant repository configurations, you only need to merge the AUBO private repository section to avoid duplicate configuration.

► Click to view configuration:

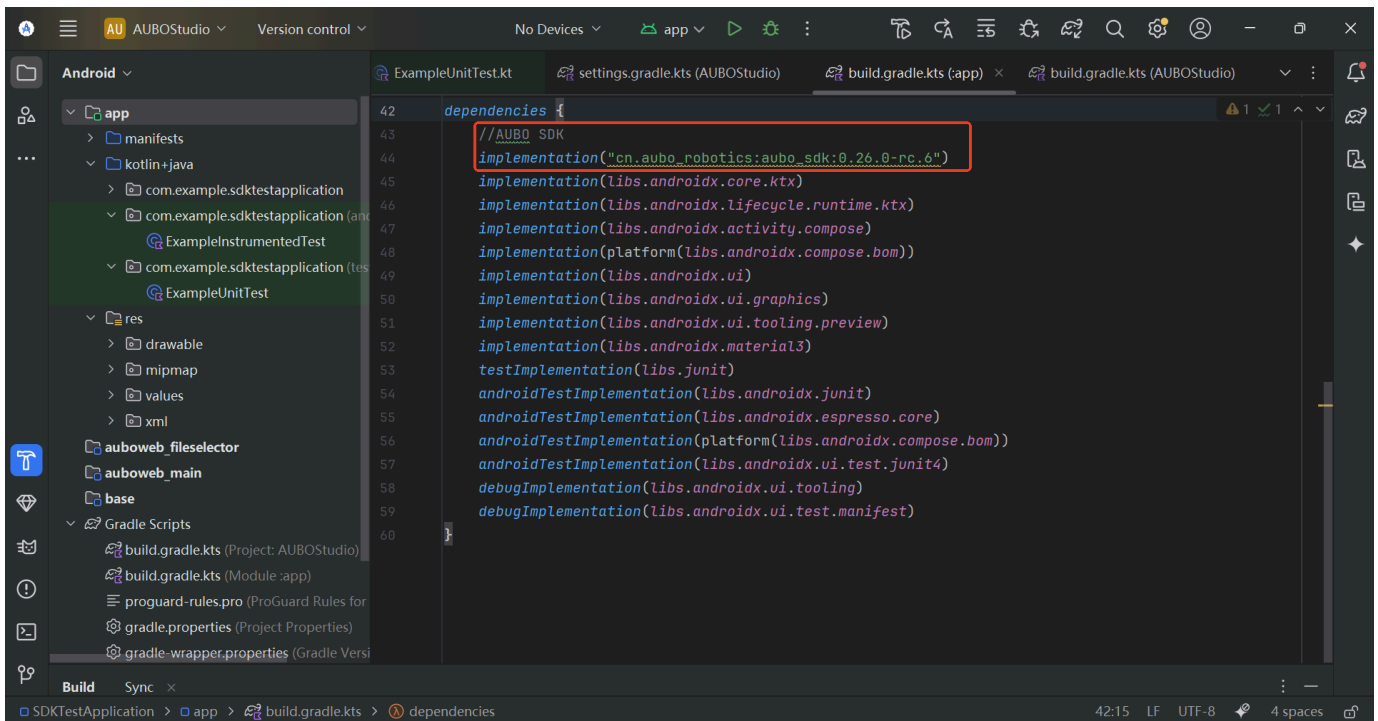


2. Add AUBO SDK dependency in the App module's `build.gradle.kts`.

```

dependencies {
    //AUBO SDK
    implementation("cn.aubo_robotics:aubo_sdk:0.26.0-rc.6")
}
  
```

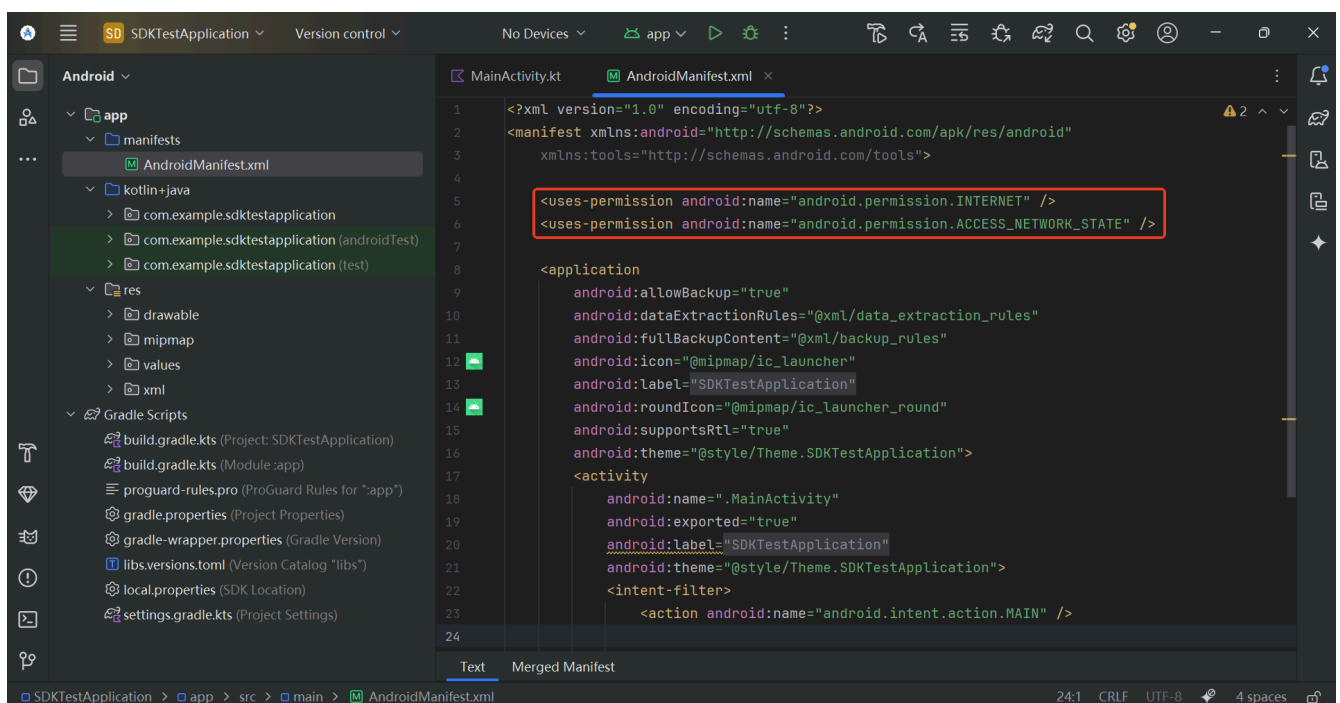
groovy



3. AndroidManifest permission configuration

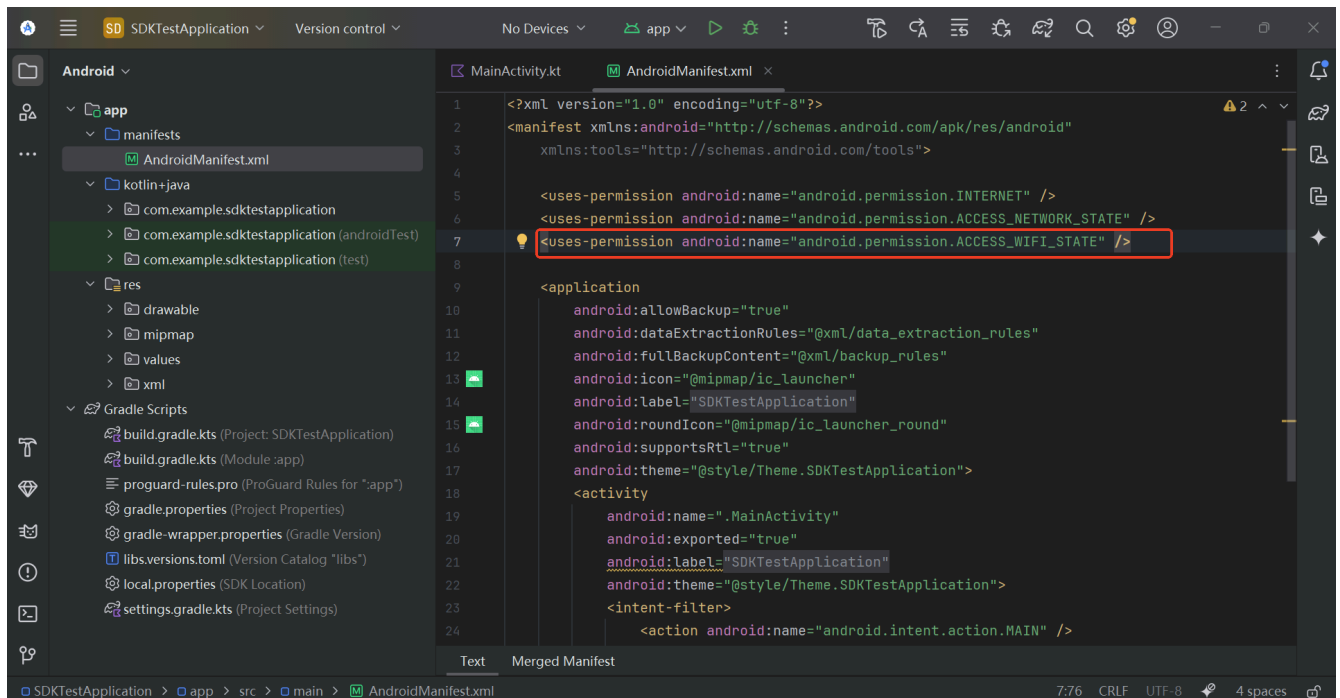
- Required permissions: The SDK requires network communication with the robot controller. Be sure to add the following 2 network permissions in `AndroidManifest.xml`, otherwise issues such as connection failure and network request exceptions may occur:

```
<uses-permission android:name="android.permission.INTERNET" />
<uses-permission android:name="android.permission.ACCESS_NETWORK_STATE" />
```



- Optional permissions: If the project needs to identify the current Wi-Fi network environment (e.g., auto-scan the robot controller's LAN, verify Wi-Fi connection status), you can add the following permission:

```
<uses-permission android:name="android.permission.ACCESS_WIFI_STATE" />
```



4. Android 9+ cleartext traffic instructions

Note

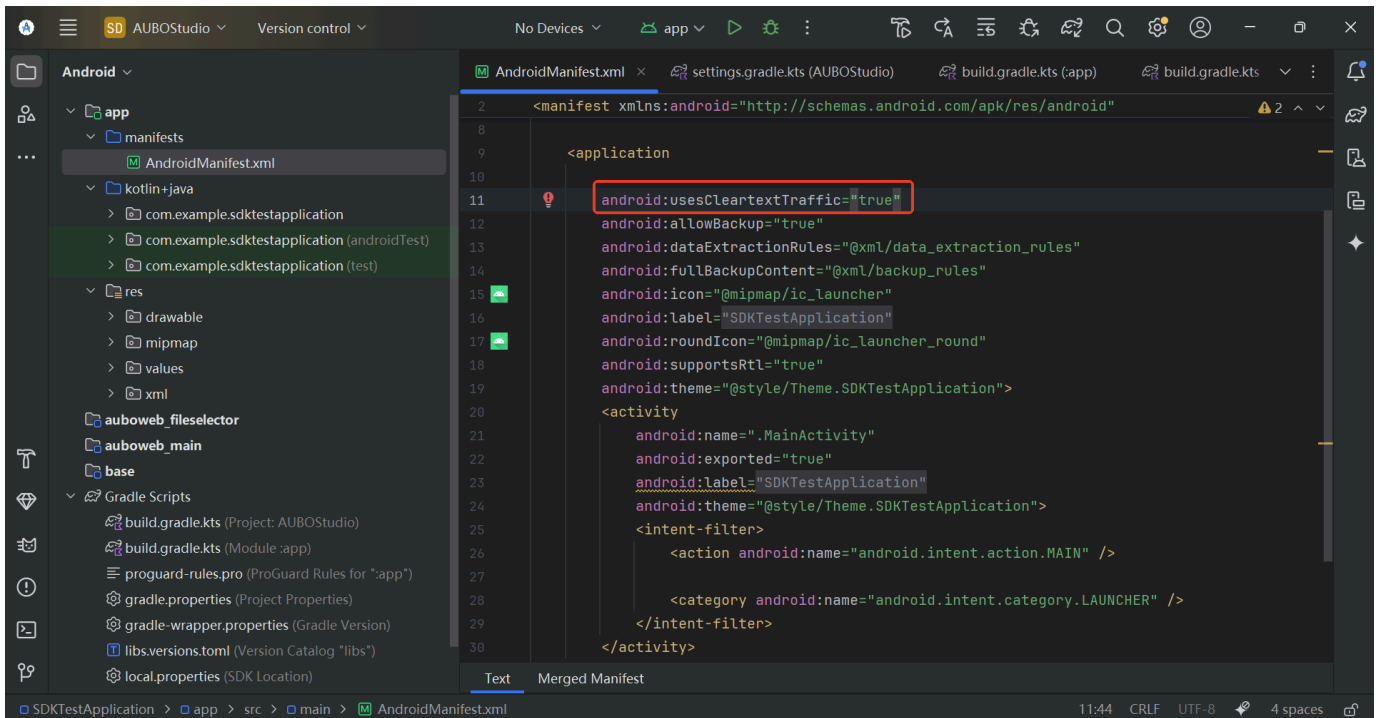
- Configure this section when connecting to the robot using HTTP/WebSocket cleartext protocols.
- If the current network policy of the project already meets the requirements, you can skip this section.

Android 9.0 and above systems restrict cleartext network traffic by default for enhanced security. Direct connections may cause robot connection failures or blocked requests.

Add cleartext access support to the `<application>` node in the project's `AndroidManifest.xml` file.

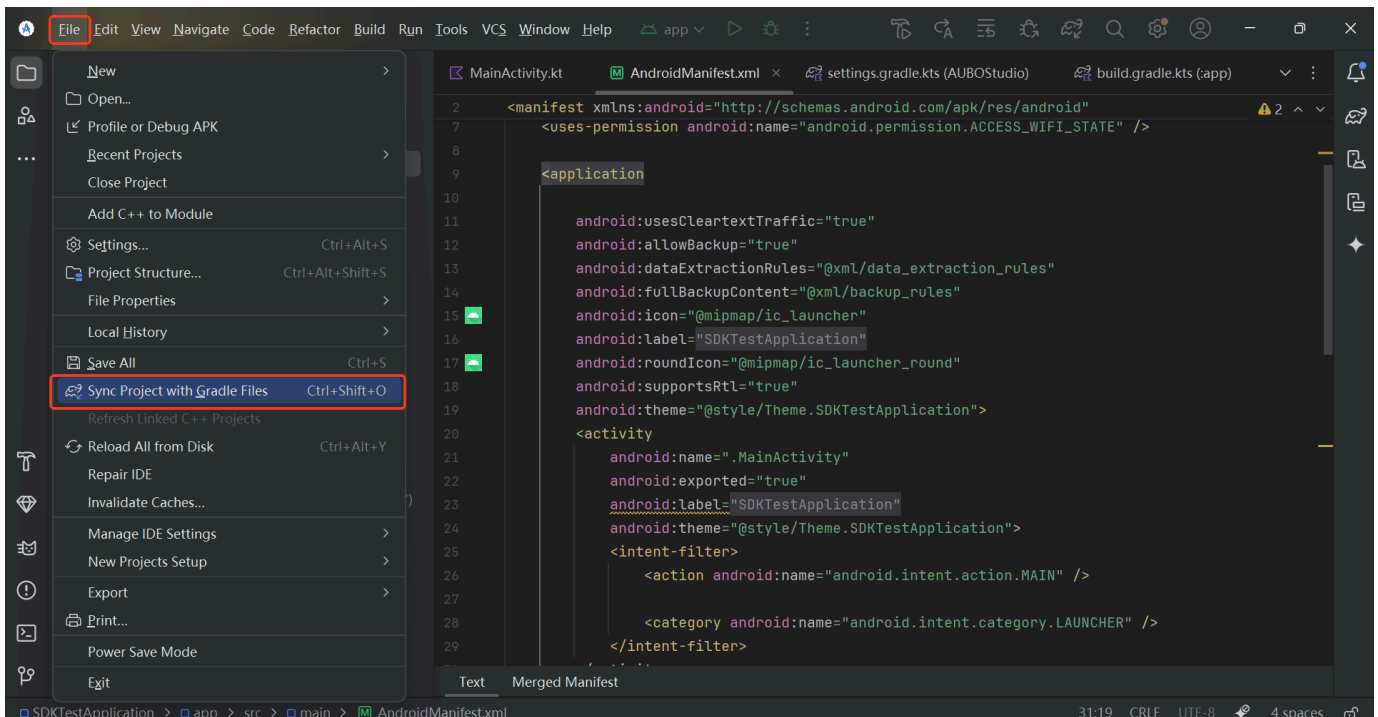
```
<application
    android:usesCleartextTraffic="true"
```

```
... >
</application>
```

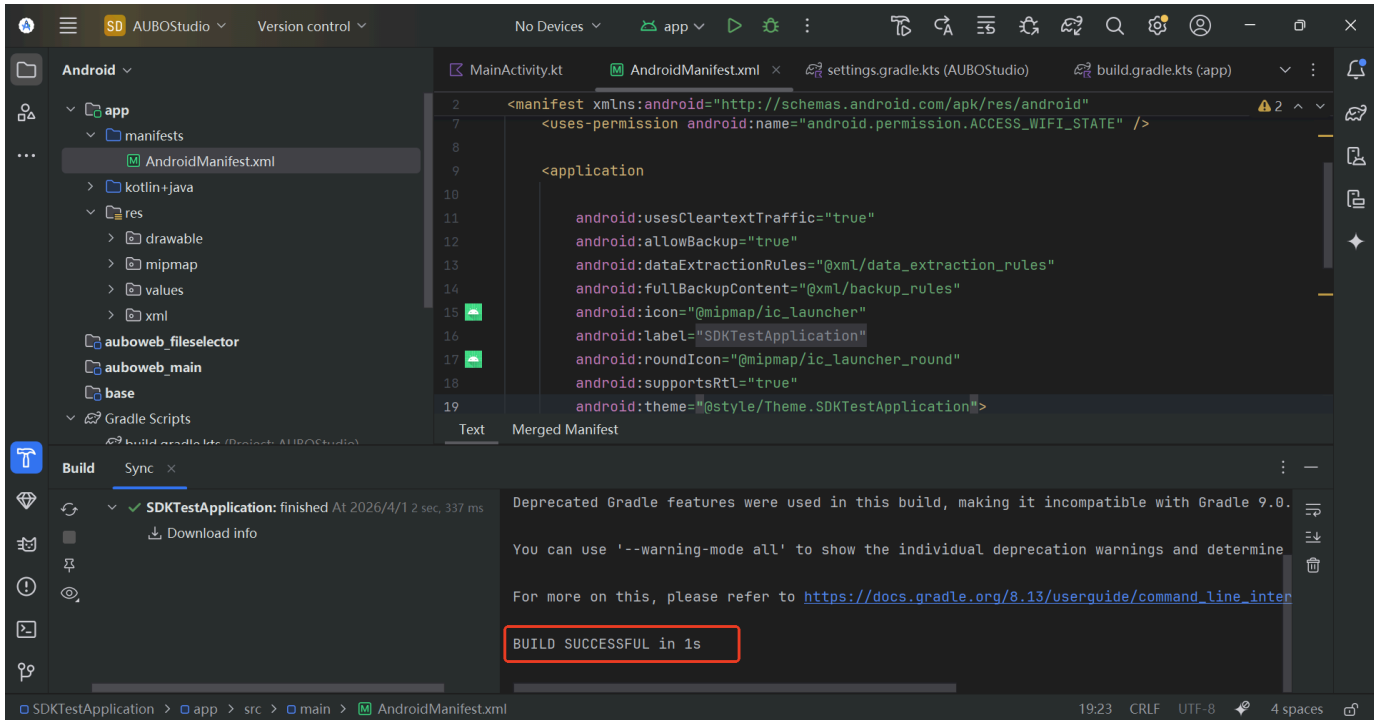


Gradle Sync Verification

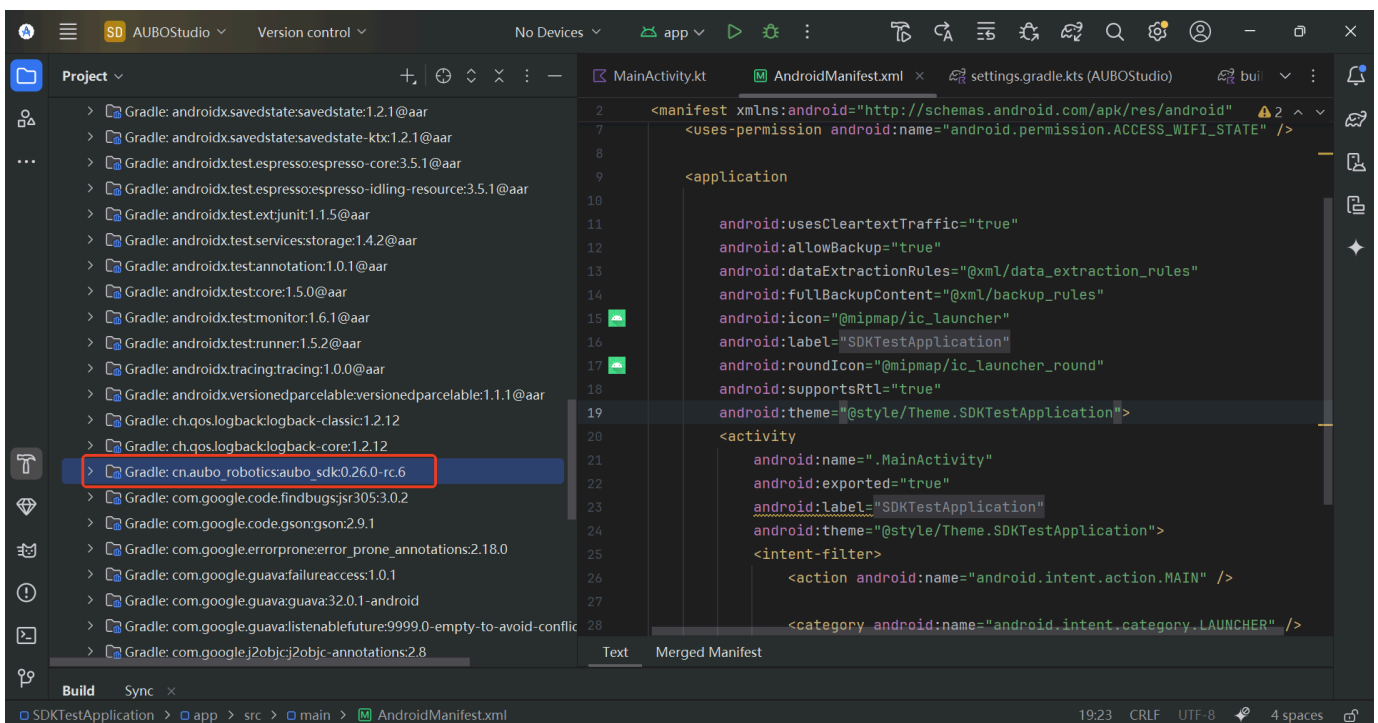
1. In Android Studio, click "File > Sync Project / Gradle Sync".



When **BUILD SUCCESSFUL** appears, the dependency download is complete.



2. When `aubo_sdk-0.26.0-rc.6.jar` appears in External Libraries, the installation is successful.



Version Management

The ARCS SDK follows the specifications in the [Semantic Version Control \(SemVer\)](#).

SDK Version Release Specifications

Normal Version

The normal version number consists of a major version number (MAJOR), a minor version number (MINOR), and a patch version number (PATCH). For example, if the normal version number is 0.18.0, the major version number is 0, the minor version number is 18, and the patch version number is 0.

Pre-release Version

The pre-release version number is denoted by appending a hyphen "-" and a series of dot separated identifiers immediately following the normal version number. Identifiers must consist of ASCII alphanumeric characters and hyphens [0-9A-Za-z-], without whitespace. For example, 1.0.0-alpha, 1.0.0-alpha.1.

The pre-release version numbers have a lower precedence than the associated normal version number. A version marked with a pre-release version number indicates that this version is not stable and may not meet the expected compatibility requirements.

Build Metadata

Build Metadata refers to additional information, such as build time, serial number, and sha checksum. It is denoted by appending a hyphen "-" and a series of dot separated identifiers immediately after the revision or pre-release version number. For example, 1.0.0-alpha+001, 1.0.0+20130313144700, and 1.0.0-beta+exp.sha.5114f85.

When two versions have the same version number and only differ in the build metadata, they have the same preference.

Quick Start

This section introduces how to directly run the example codes in the AUBO SDK.

- [SDK PC Communication Setup with Robot](#)
 - Set up wired and wireless communication between the SDK PC and the robot.
- [C++ Example](#)
 - Before running the C++ example, it is necessary to:
 - Configure the compilation run environment. Please refer to [Running Environment Requirements](#).
 - Install the AUBO SDK locally. Please refer to [Installation Guide](#).
- [Python Example](#)
 - Before running the Python example, it is necessary to:
 - Configure the compilation run environment. Please refer to [Running Environment Requirements](#).
 - Install the aubo_sdk and the pyaubo_sdk library locally. Please refer to [Installation Guide](#).
- [Java Example](#)
 - Before running the Java example, it is necessary to:
 - Configure the compilation run environment. Please refer to [Running Environment Requirements](#).
 - Install the aubo_sdk development package and the java_sdk library locally. Please refer to [Installation Guide](#).

SDK PC Communication Settings with Robot

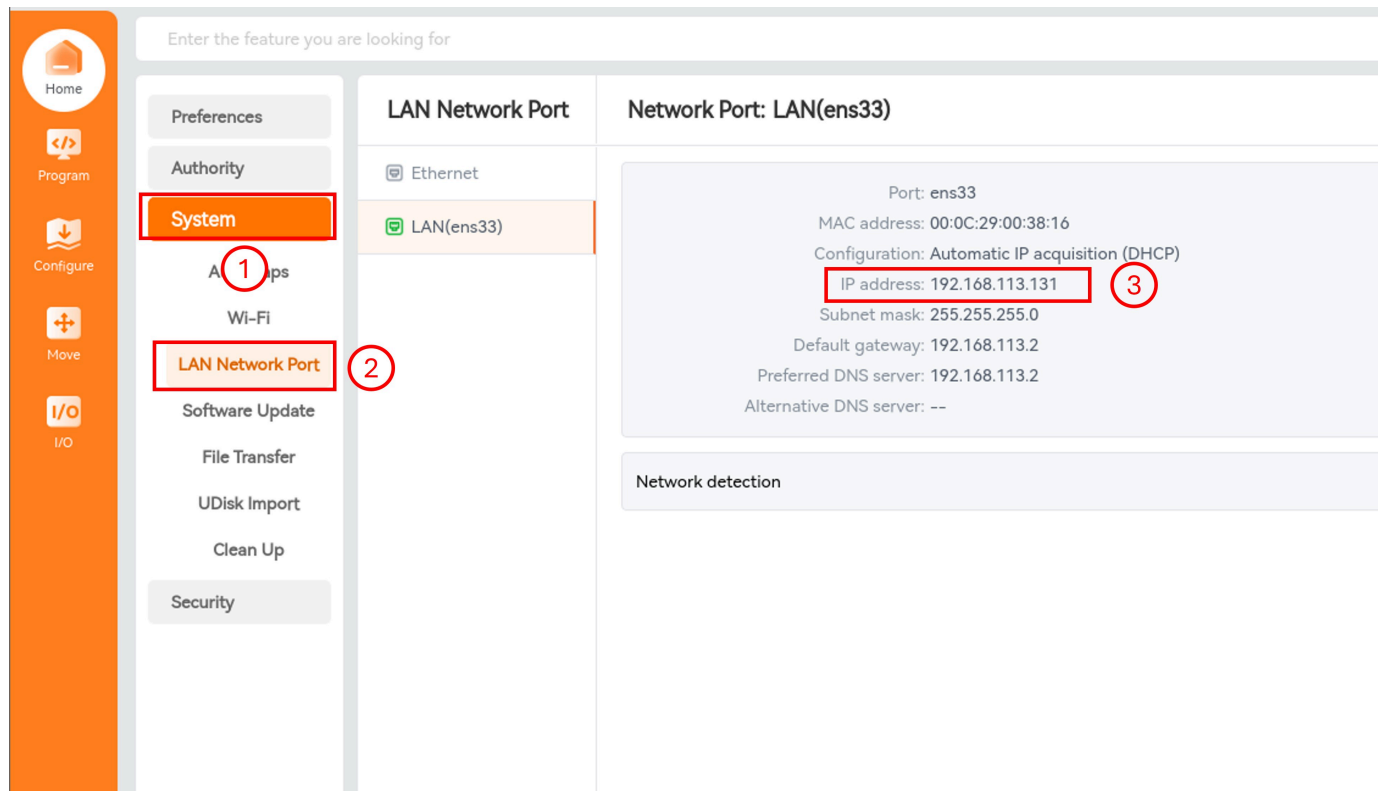
The SDK PC can communicate with the robot in two ways:

1. **Wireless Network Communication:** Wireless network communication is a method of communicating with the robot via a wireless local area network (Wi-Fi). To use the wireless network communication method, it is necessary to configure the corresponding network settings of the PC and the robot so that they are on the same wireless local area network (Wi-Fi). The PC can send commands to the robot, receive data, and interact with it via a wireless network.
 2. **Wired Network Communication:** Wired network communication is a method of communicating with the robot via a wired connection (Ethernet). To use this communication method, it is necessary to connect the PC to the control box with a network cable and set the IP address, subnet mask, and gateway so that they are on the same subnet. The PC can send commands to the robot, receive data, and interact with it via a wired network.
-

Wireless Network Communication

You can achieve wireless network communication between the robot and the PC by referring to the following steps:

1. Connect the control box to the router using an Ethernet cable.
2. Inside Settings, go in System and use Automatic IP acquisition (DHCP). Then in **LAN Network Port** you can see the IP address of the robot in the network details.



3. Connect the PC to the same Wi-Fi as the robot.

4. Test the network connectivity on the PC.

- **Windows operating system:**

- Method 1: Use the ping command. Open the cmd and enter ping 192.168.113.131 (what ever your robot's ip address is). As shown in the figure below, the communication is successful.

```
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

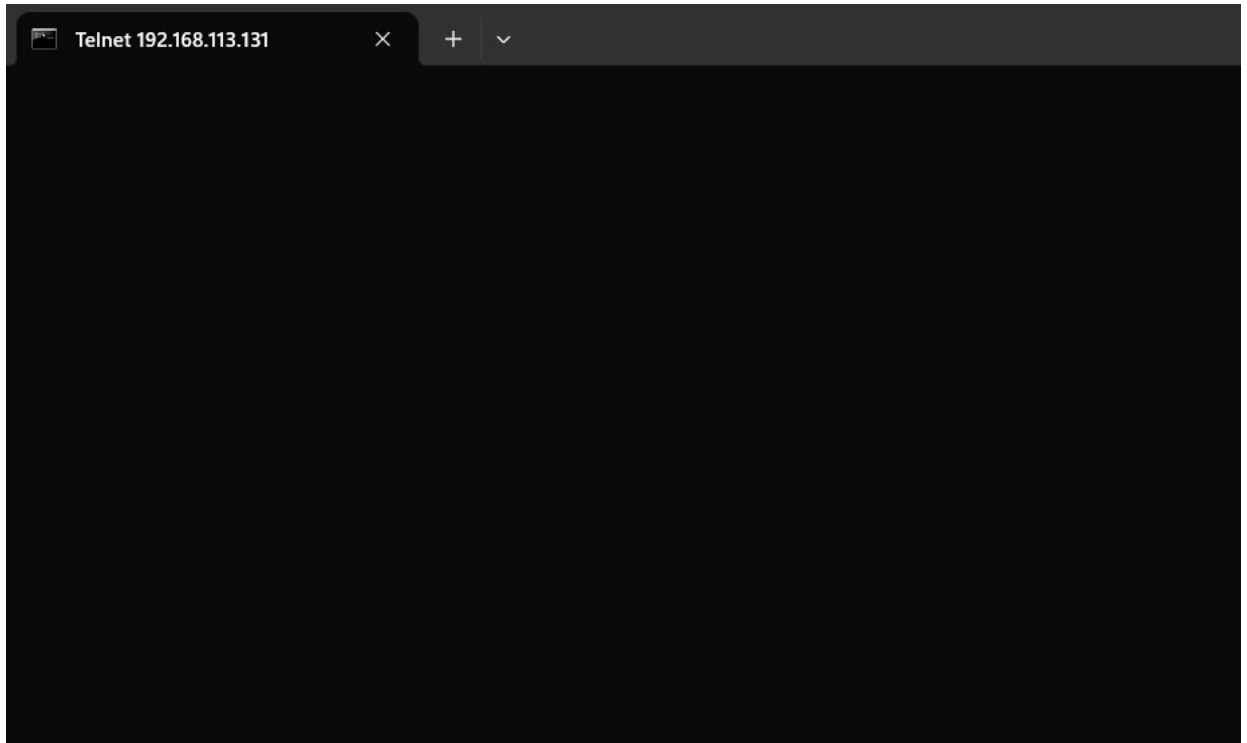
Install the latest PowerShell for new features and improvements! https://aka.ms/PSWindows

PS C:\Users\> ping 192.168.113.131

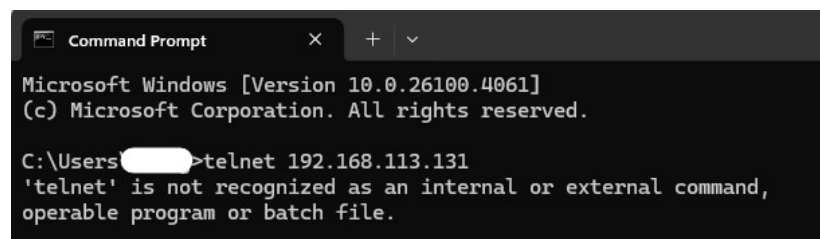
Pinging 192.168.113.131 with 32 bytes of data:
Reply from 192.168.113.131: bytes=32 time<1ms TTL=64
Reply from 192.168.113.131: bytes=32 time=3ms TTL=64
Reply from 192.168.113.131: bytes=32 time=1ms TTL=64
Reply from 192.168.113.131: bytes=32 time=1ms TTL=64

Ping statistics for 192.168.113.131:
    Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
    Approximate round trip times in milli-seconds:
        Minimum = 0ms, Maximum = 3ms, Average = 1ms
```

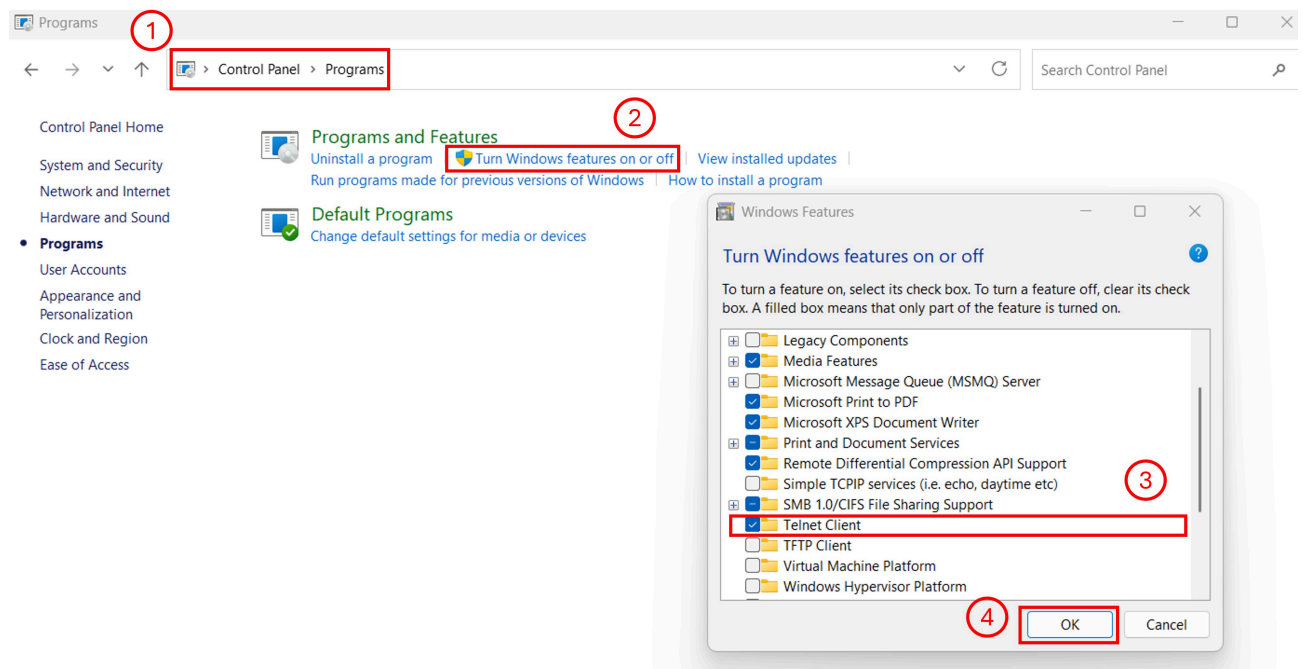
- Method 2: Use the telnet command. Open the cmd and enter telnet 192.168.113.131 30004. As shown in the figure below, the communication is successful.



If the prompt shown in the figure below appears, it means telnet is not enabled.

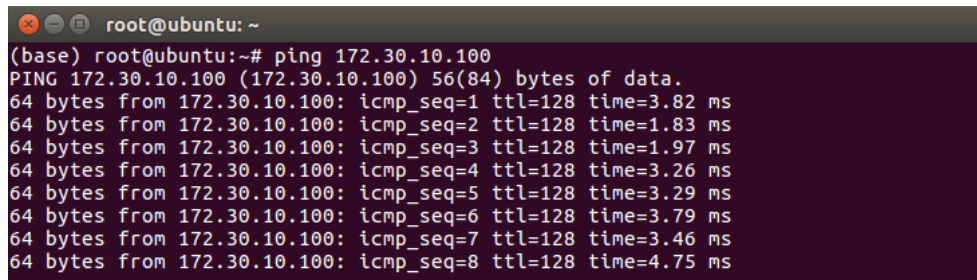


Follow the steps shown in the figure below to enable telnet.



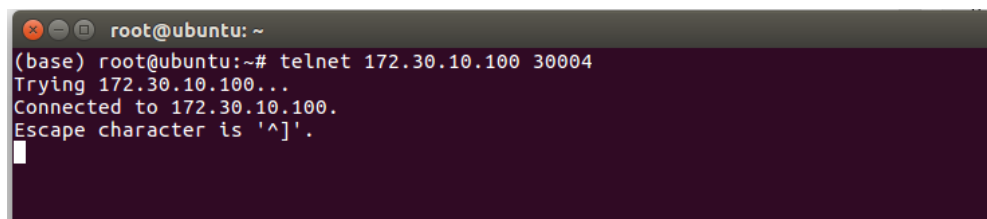
- Linux operating system:

- Method 1: Use the ping command. Open the terminal and enter ping 172.30.10.100 (what ever your robot's ip address is). As shown in the figure below, the communication is successful.



```
root@ubuntu: ~  
(base) root@ubuntu:~# ping 172.30.10.100  
PING 172.30.10.100 (172.30.10.100) 56(84) bytes of data.  
64 bytes from 172.30.10.100: icmp_seq=1 ttl=128 time=3.82 ms  
64 bytes from 172.30.10.100: icmp_seq=2 ttl=128 time=1.83 ms  
64 bytes from 172.30.10.100: icmp_seq=3 ttl=128 time=1.97 ms  
64 bytes from 172.30.10.100: icmp_seq=4 ttl=128 time=3.26 ms  
64 bytes from 172.30.10.100: icmp_seq=5 ttl=128 time=3.29 ms  
64 bytes from 172.30.10.100: icmp_seq=6 ttl=128 time=3.79 ms  
64 bytes from 172.30.10.100: icmp_seq=7 ttl=128 time=3.46 ms  
64 bytes from 172.30.10.100: icmp_seq=8 ttl=128 time=4.75 ms
```

- Method 2: Use the telnet command. Open the terminal and enter telnet 172.30.10.100 30004. As shown in the figure below, the communication is successful.

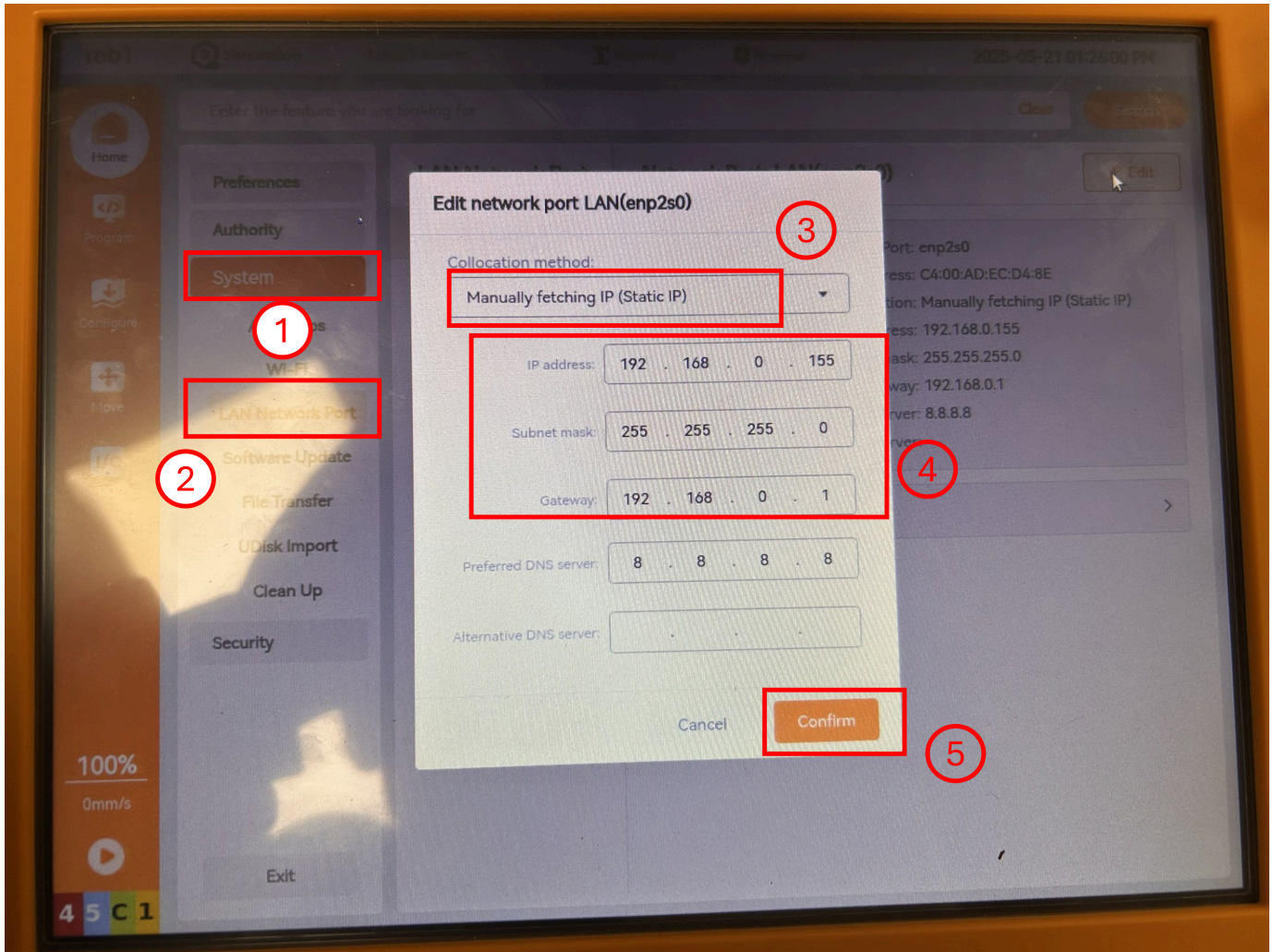


```
root@ubuntu: ~  
(base) root@ubuntu:~# telnet 172.30.10.100 30004  
Trying 172.30.10.100...  
Connected to 172.30.10.100.  
Escape character is '^['.
```

Wired Network Communication

You can achieve wired network communication between the robot and the PC by referring to the following steps:

1. Connect the control box to the PC using an Ethernet cable.
2. Go into Settings, System, LAN Network and set the **Static IP** in the teach pendant. The **IP address** is the robot's IP. The robot's IP address in the figure below is 192.168.0.155. The **subnet mask** must be set to 255.255.255.0.
The host address part of the IP address of the **gateway** is usually set to 1. For example, if the **IP address** is 192.168.0.155, its **default gateway** should be set to 192.168.0.1.



3. Set the **IP address**, **subnet mask**, and **gateway** on the PC to make it in the same subnet as the robot.

The **IP address prefixes** of the robot and the PC must be the same. For example, if the robot's IP address is 192.168.0.155, its **IP address prefix** is 192.168.0, and the PC's **IP address prefix** must also be 192.168.0.

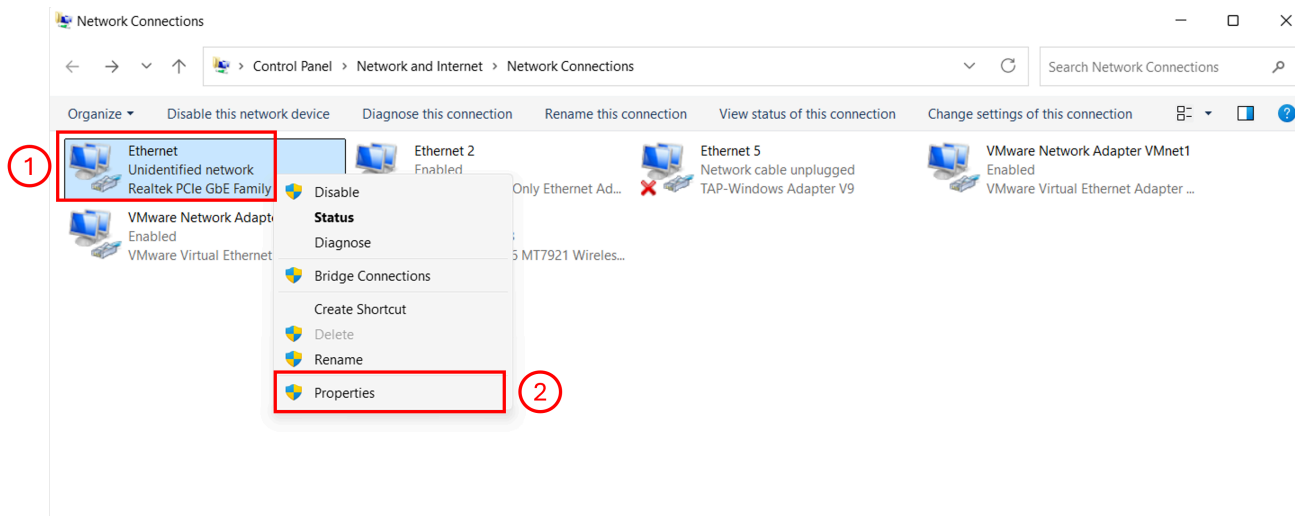
Both the **subnet masks** of the robot and the PC must be 255.255.255.0.

The **gateways** of the robot and the PC must be the same, and the host address part of the gateway's IP address is usually set to 1. For example, if the IP address is 192.168.0.155, the gateway should be set to 192.168.0.1.

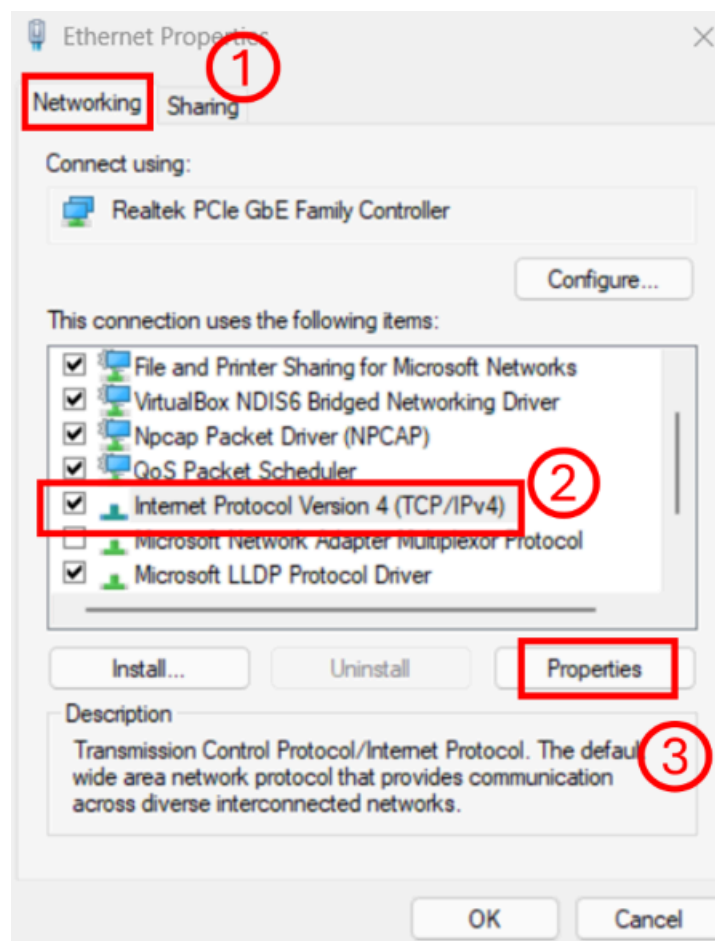
The following describes how to configure in [Windows Operating System] (#windows) and [Linux Operating System] (#Linux).

- [Windows operating system:](#)

- Find **Ethernet** on the PC ---> Select it ---> Right-click ---> Click **Properties**.

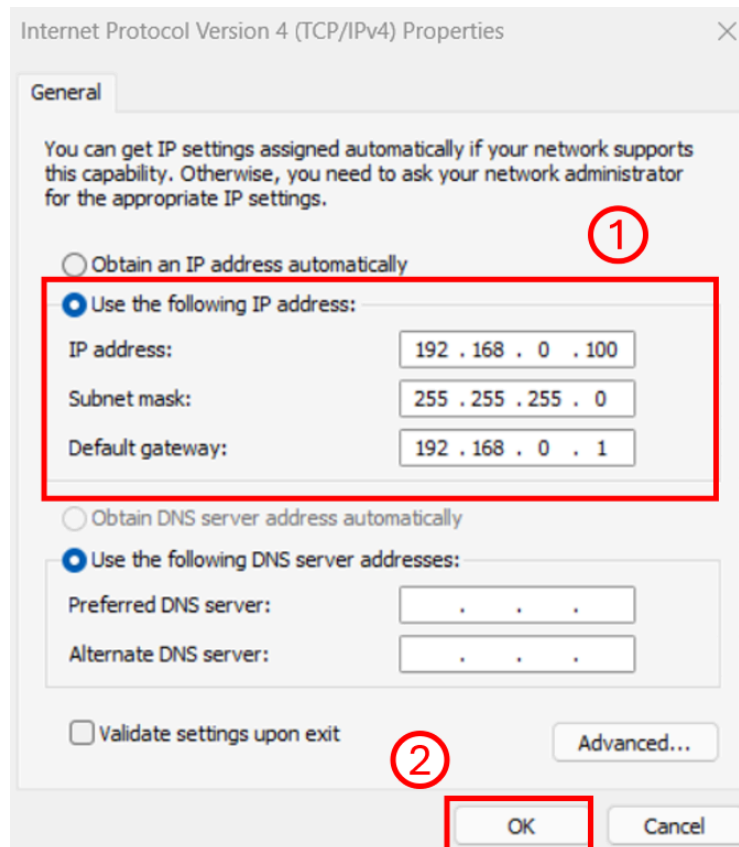


- Select **Internet Protocol Version 4 (TCP/IPv4)** ----> Click **Properties**.



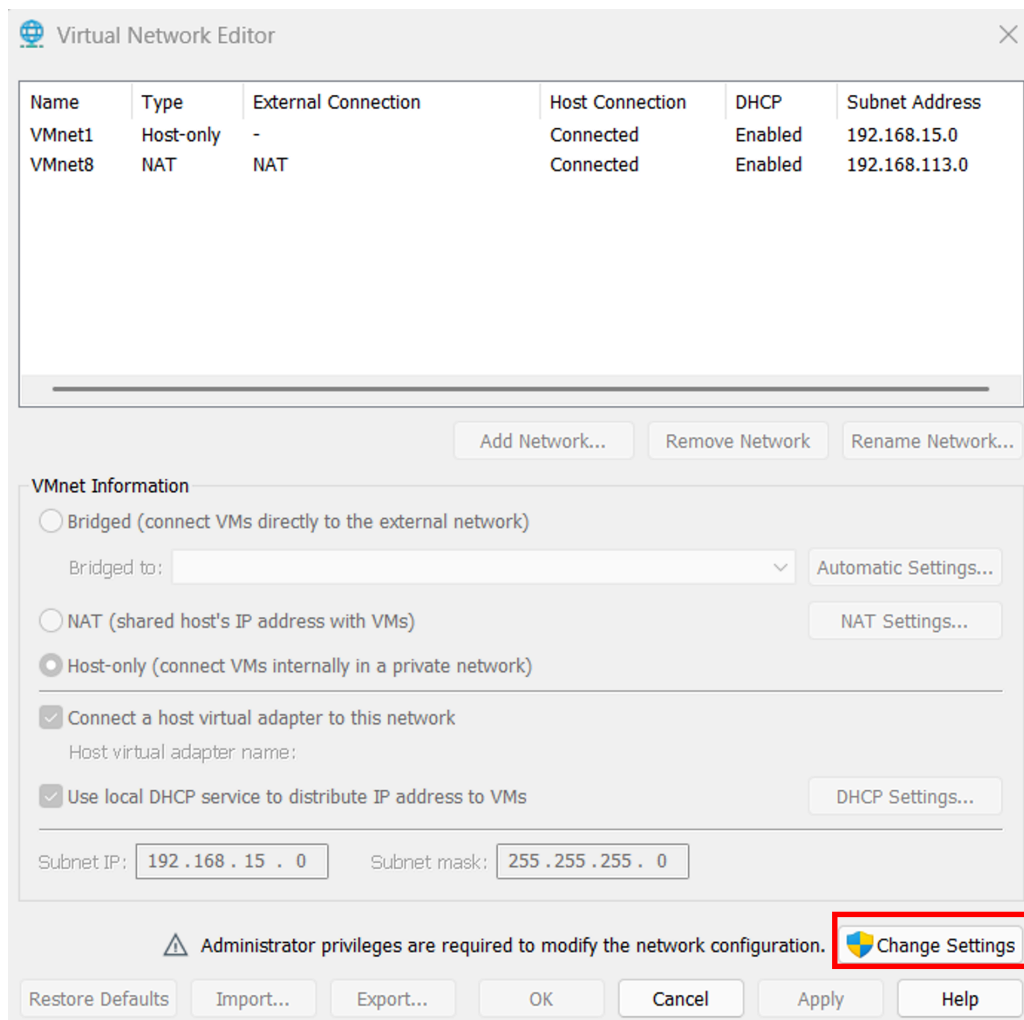
- As shown in the figure below, set the **IP address**, **subnet mask**, and **gateway** ----> Click **OK**.

In the figure, the host's IP address is 192.168.0.100, and the robot's IP address is 192.168.0.155. They are in the same subnet.

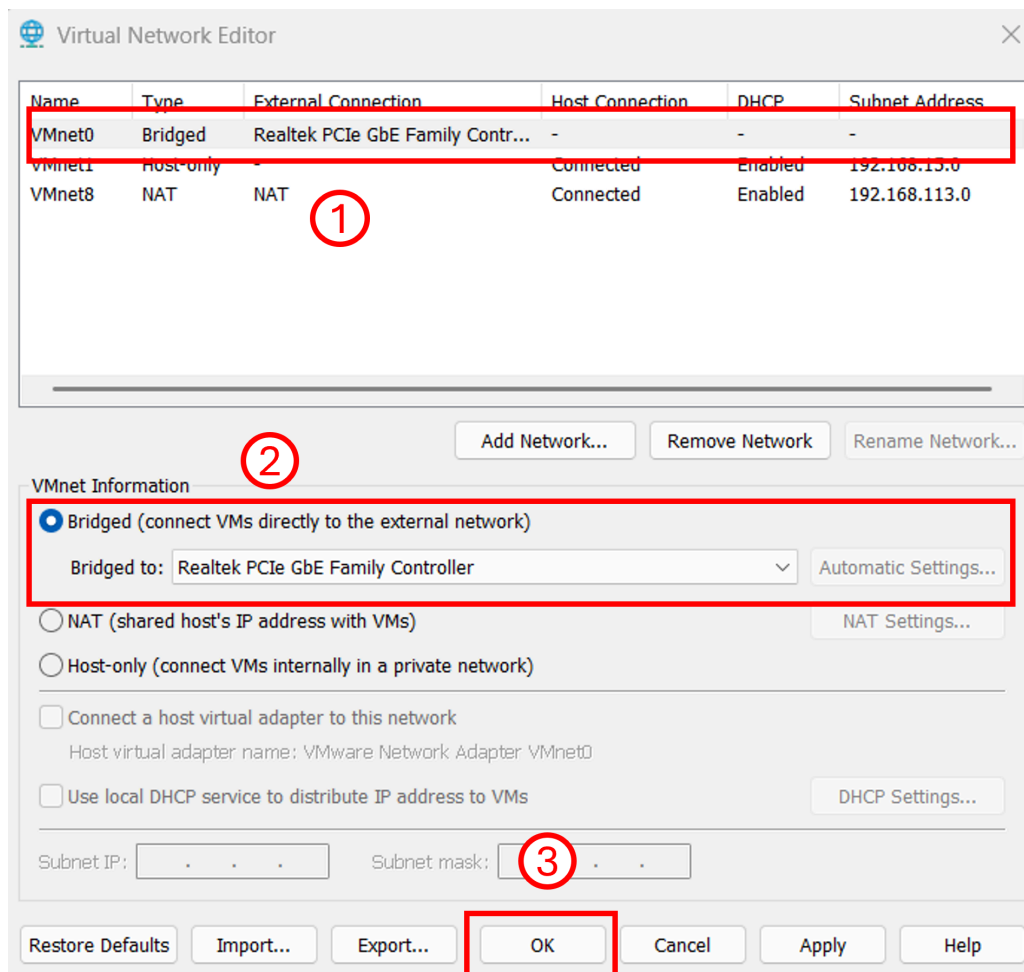


- [VMware virtual machine:](#)

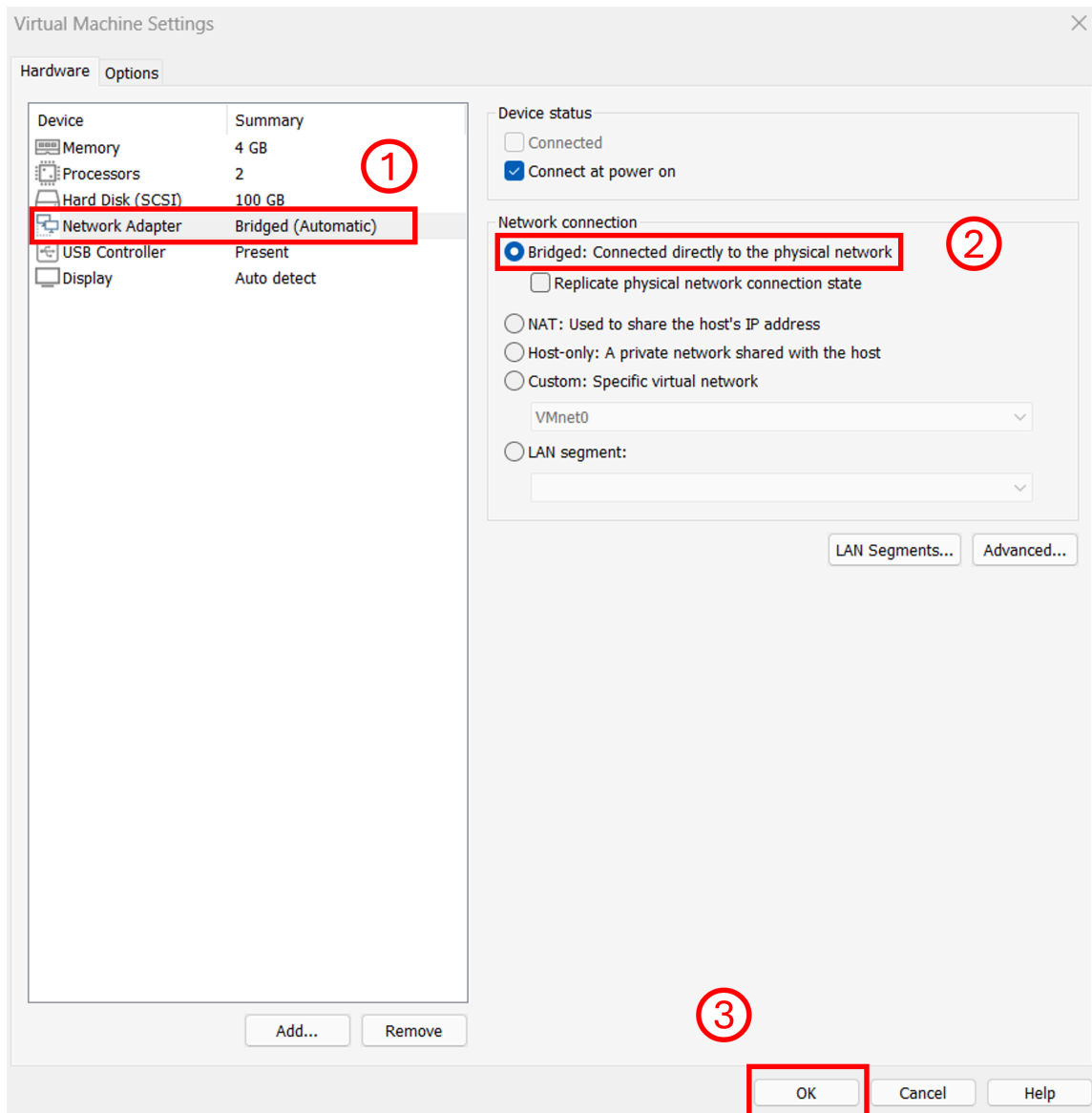
- On the **PC**, set the Ethernet's **IP address**, **subnet mask**, and **gateway** so that the host and the robot are in the same subnet.
- Open the **Virtual Network Editor**, select VMnet0 if possible (or which ever one available), and click **Change Settings**.



- Select **VMnet0**, choose to bridge to **Realtek PCIe GbE Family Controller**, and click **OK**.



- Open the **virtual machine settings**, set the **Network connection** in the **Network Adapter** to the **Bridged**.



- Set the virtual machine's IP address to be in the same subnet as the robot.
In the figure below, the virtual machine's IP address is 192.168.0.120, and the robot's IP address is 192.168.0.155. They are in the same subnet.

```

Terminal - root@aubo-2605431136: ~/Desktop
File Edit View Terminal Tabs Help
root@aubo-2605431136:~/Desktop# ifconfig
ens33: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500
    inet 192.168.0.120 netmask 255.255.255.0 broadcast 192.168.0.255
    inet6 fe80::da25:9f17:3977:7553 prefixlen 64 scopeid 0x20<link>
    ether 00:0c:29:00:38:16 txqueuelen 1000 (Ethernet)
    RX packets 219 bytes 19142 (19.1 KB)
    RX errors 0 dropped 0 overruns 0 frame 0
    TX packets 184 bytes 21106 (21.1 KB)
    TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

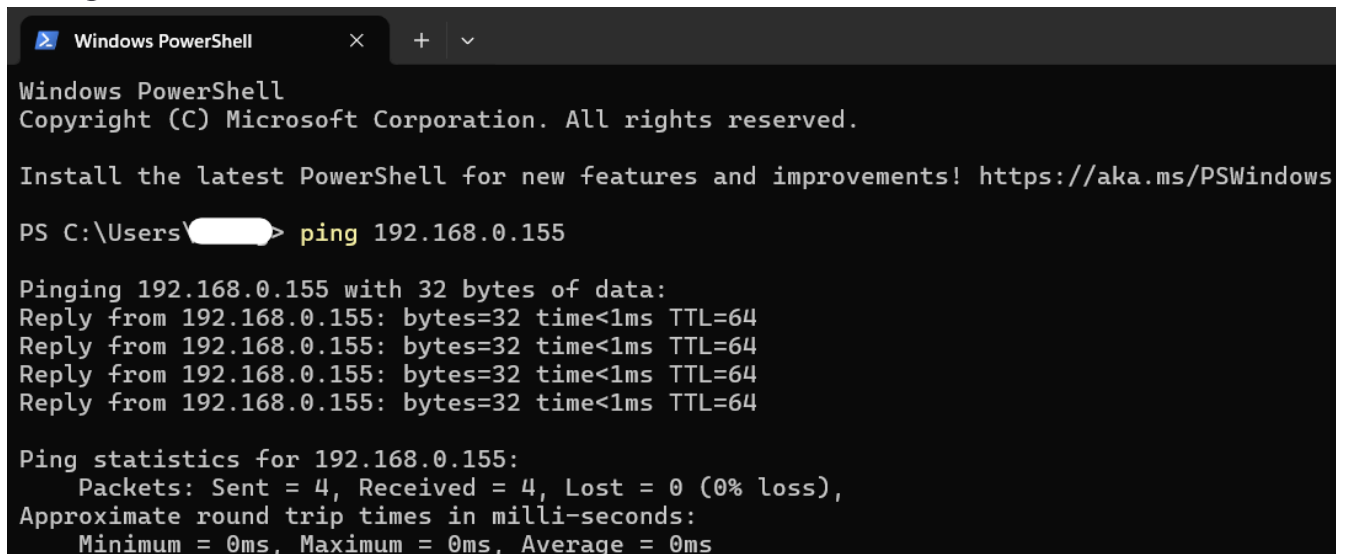
lo: flags=73<UP,LOOPBACK,RUNNING> mtu 65536
    inet 127.0.0.1 netmask 255.0.0.0
    inet6 ::1 prefixlen 128 scopeid 0x10<host>
    loop txqueuelen 1000 (Local Loopback)
    RX packets 82079 bytes 13237106 (13.2 MB)
    RX errors 0 dropped 0 overruns 0 frame 0
    TX packets 82079 bytes 13237106 (13.2 MB)
    TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

root@aubo-2605431136:~/Desktop#

```

4. Test the network connectivity on the PC.

- **Windows operating system:** Open the cmd and enter ping 192.168.0.155. As shown in the figure below, the communication is successful.



```
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

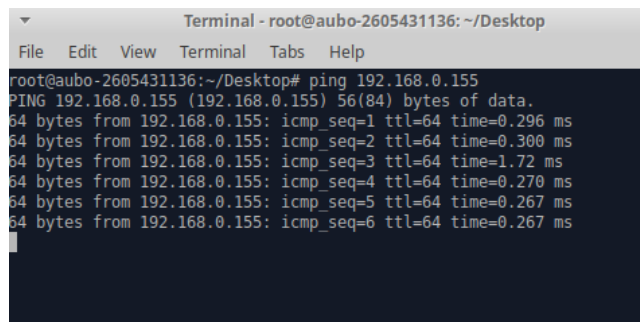
Install the latest PowerShell for new features and improvements! https://aka.ms/PSWindows

PS C:\Users\> ping 192.168.0.155

Pinging 192.168.0.155 with 32 bytes of data:
Reply from 192.168.0.155: bytes=32 time<1ms TTL=64
Reply from 192.168.0.155: bytes=32 time<1ms TTL=64
Reply from 192.168.0.155: bytes=32 time<1ms TTL=64
Reply from 192.168.0.155: bytes=32 time<1ms TTL=64

Ping statistics for 192.168.0.155:
    Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
    Approximate round trip times in milli-seconds:
        Minimum = 0ms, Maximum = 0ms, Average = 0ms
```

- **Linux operating system:** Open the terminal and enter ping 192.168.0.155. As shown in the figure below, the communication is successful.



```
Terminal - root@aubo-2605431136: ~/Desktop
File Edit View Terminal Tabs Help

root@aubo-2605431136:~/Desktop# ping 192.168.0.155
PING 192.168.0.155 (192.168.0.155) 56(84) bytes of data.
64 bytes from 192.168.0.155: icmp_seq=1 ttl=64 time=0.296 ms
64 bytes from 192.168.0.155: icmp_seq=2 ttl=64 time=0.300 ms
64 bytes from 192.168.0.155: icmp_seq=3 ttl=64 time=1.72 ms
64 bytes from 192.168.0.155: icmp_seq=4 ttl=64 time=0.270 ms
64 bytes from 192.168.0.155: icmp_seq=5 ttl=64 time=0.267 ms
64 bytes from 192.168.0.155: icmp_seq=6 ttl=64 time=0.267 ms
```

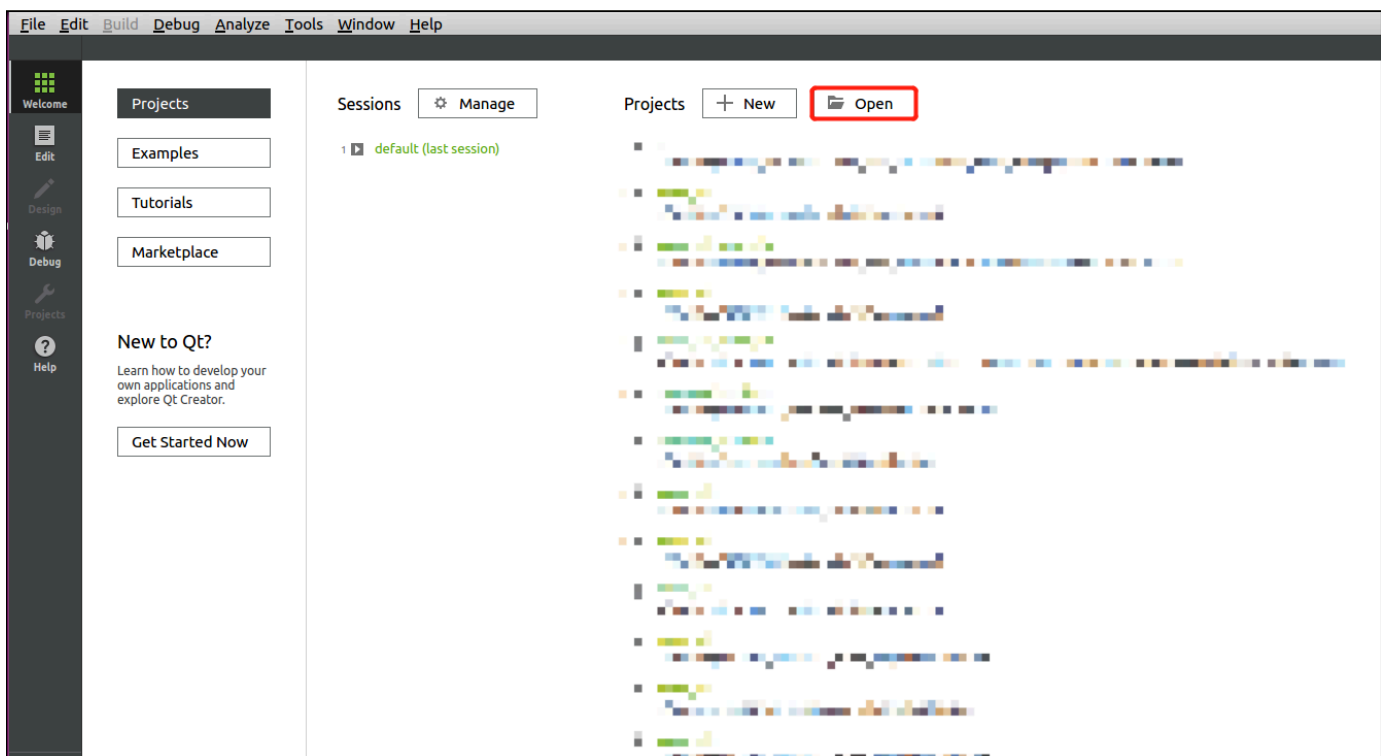
Examples of Running C++ SDK

Compile and Run CMake Projects with QtCreator in Linux Environment

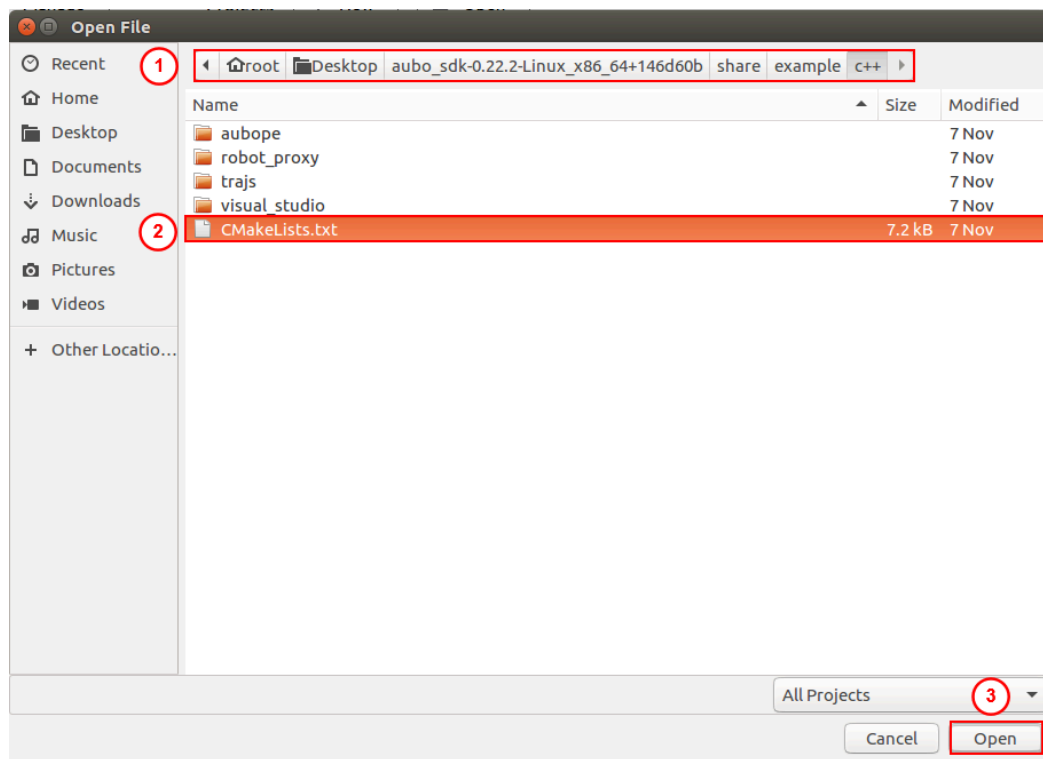
The following describes how to compile and run the `example_startup.cpp` example in the SDK using the QtCreator. After successful compilation and running, the teach pendant will be powered on successfully.

****Note:** If the client of the SDK is connected to a real robot arm, the IP address in the code must be set to the robot arm's IP address. If the project is run on the Aubo Sim virtual machine and the client of the SDK is connected to the virtual machine, the IP address should be 127.0.0.1. ******

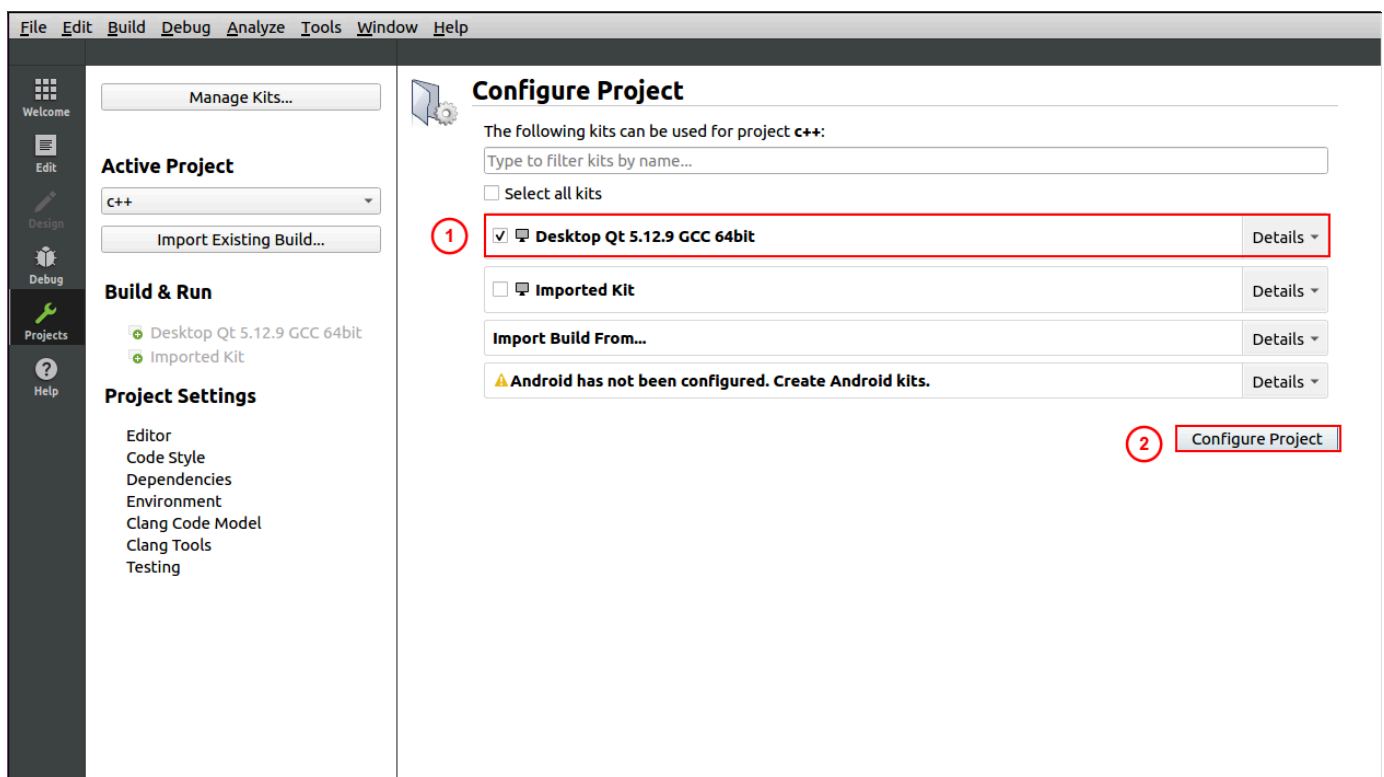
1. Open the QtCreator and click "Open".



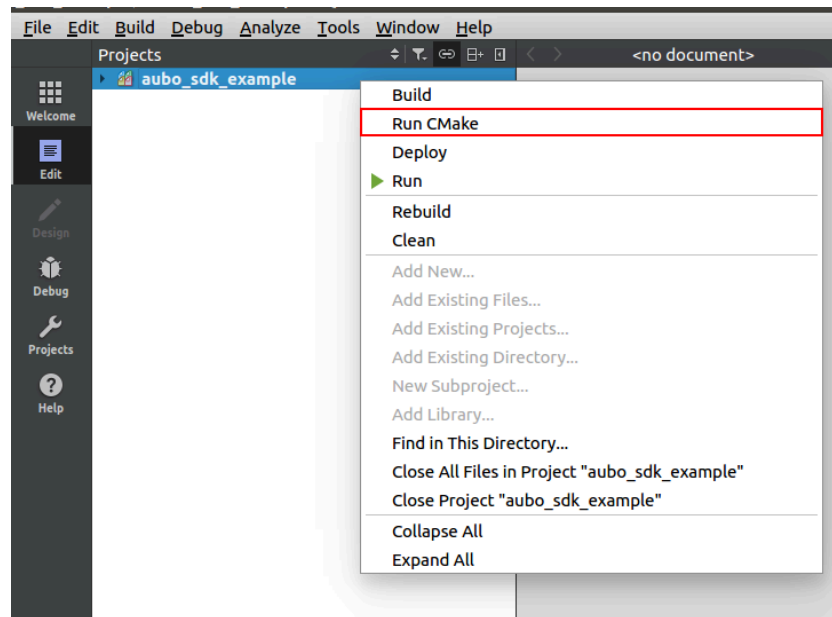
2. Locate the path of the C++ example in the SDK, select the `CMakeLists.txt`, and click "Open".



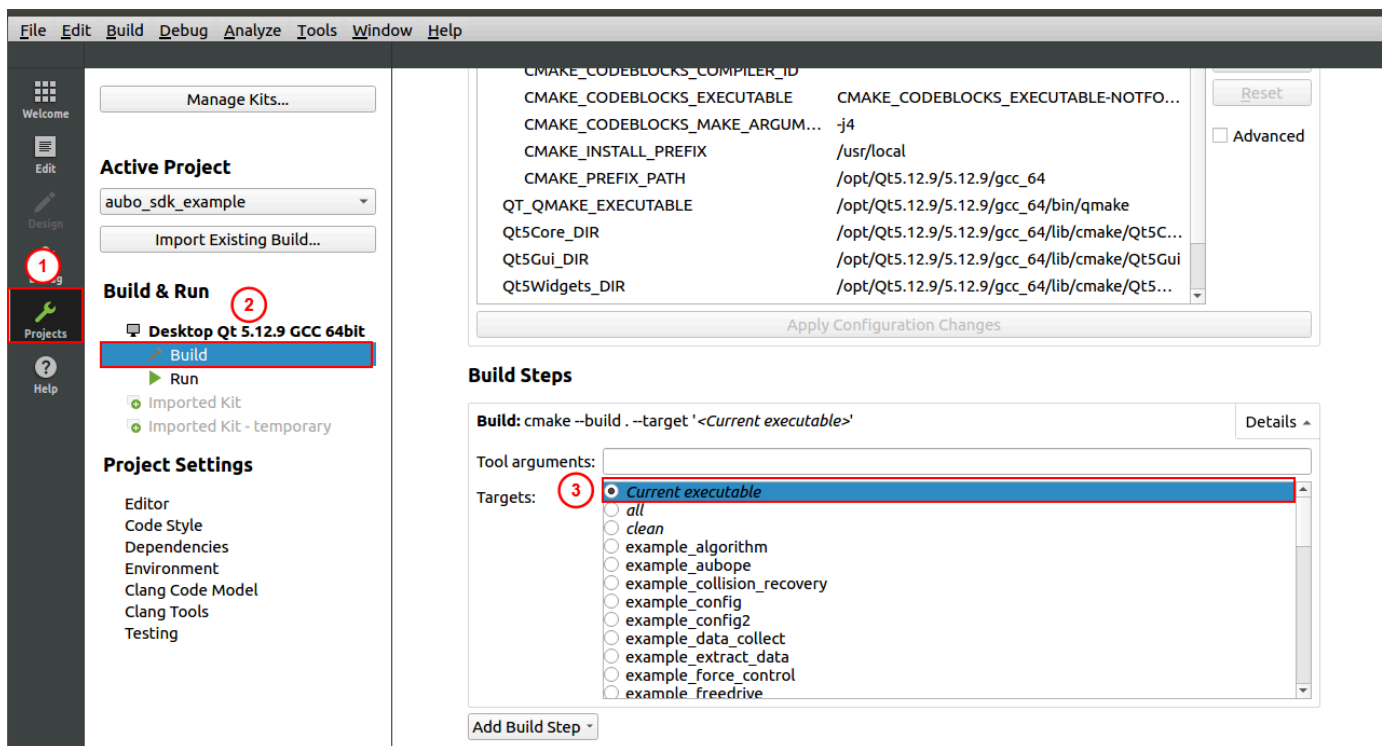
3. Configure the compiler.



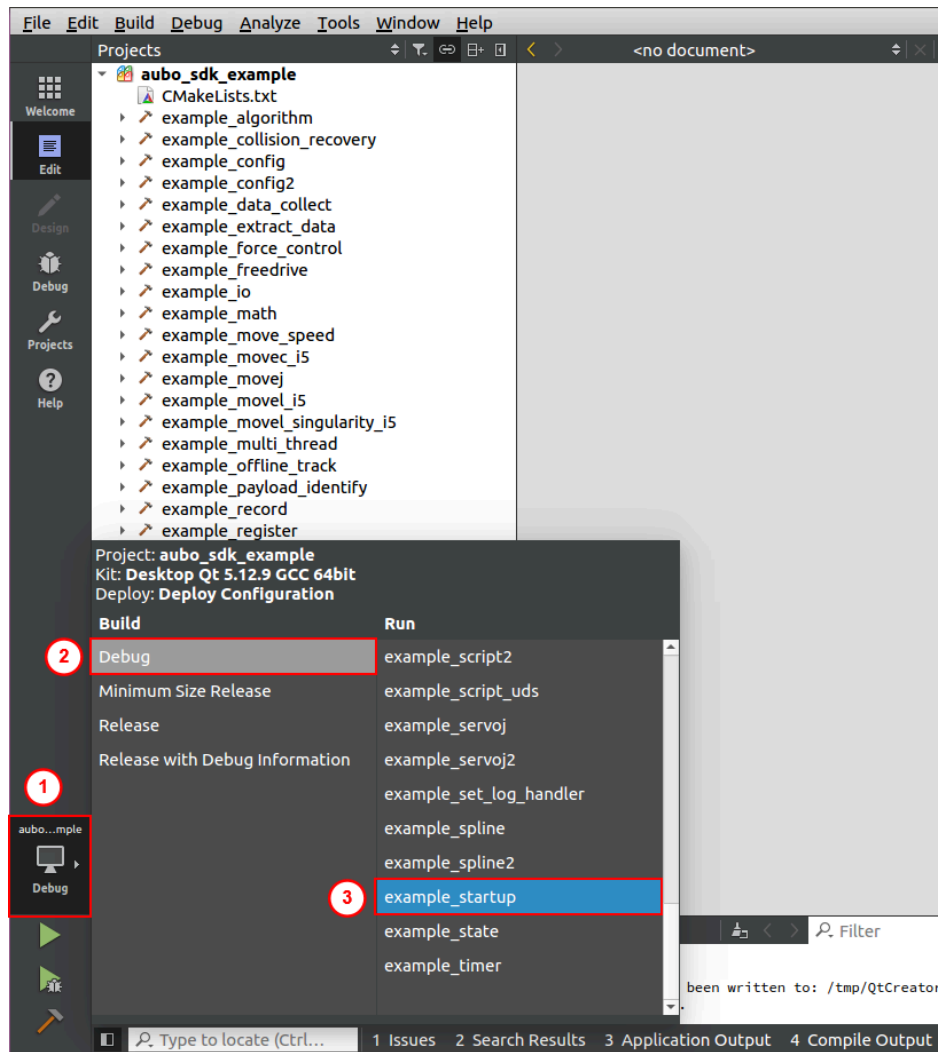
4. Select and right-click the aubo_sdk_example, and click "Run CMake".



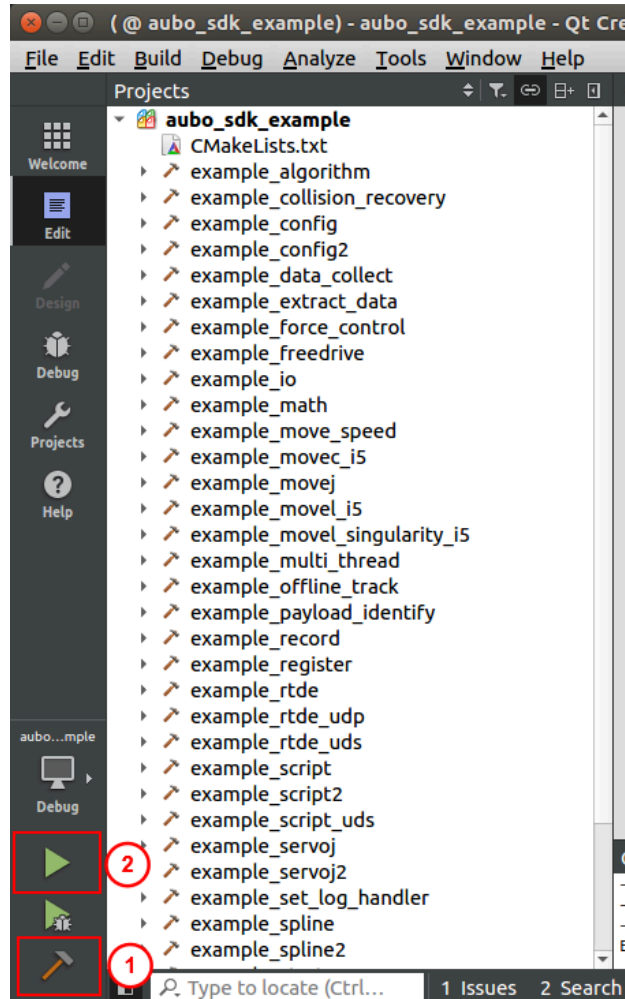
5. In the Build step, select "Current executable" in the Targets column.



6. Select the example_startup.



7. Compile and run the codes.



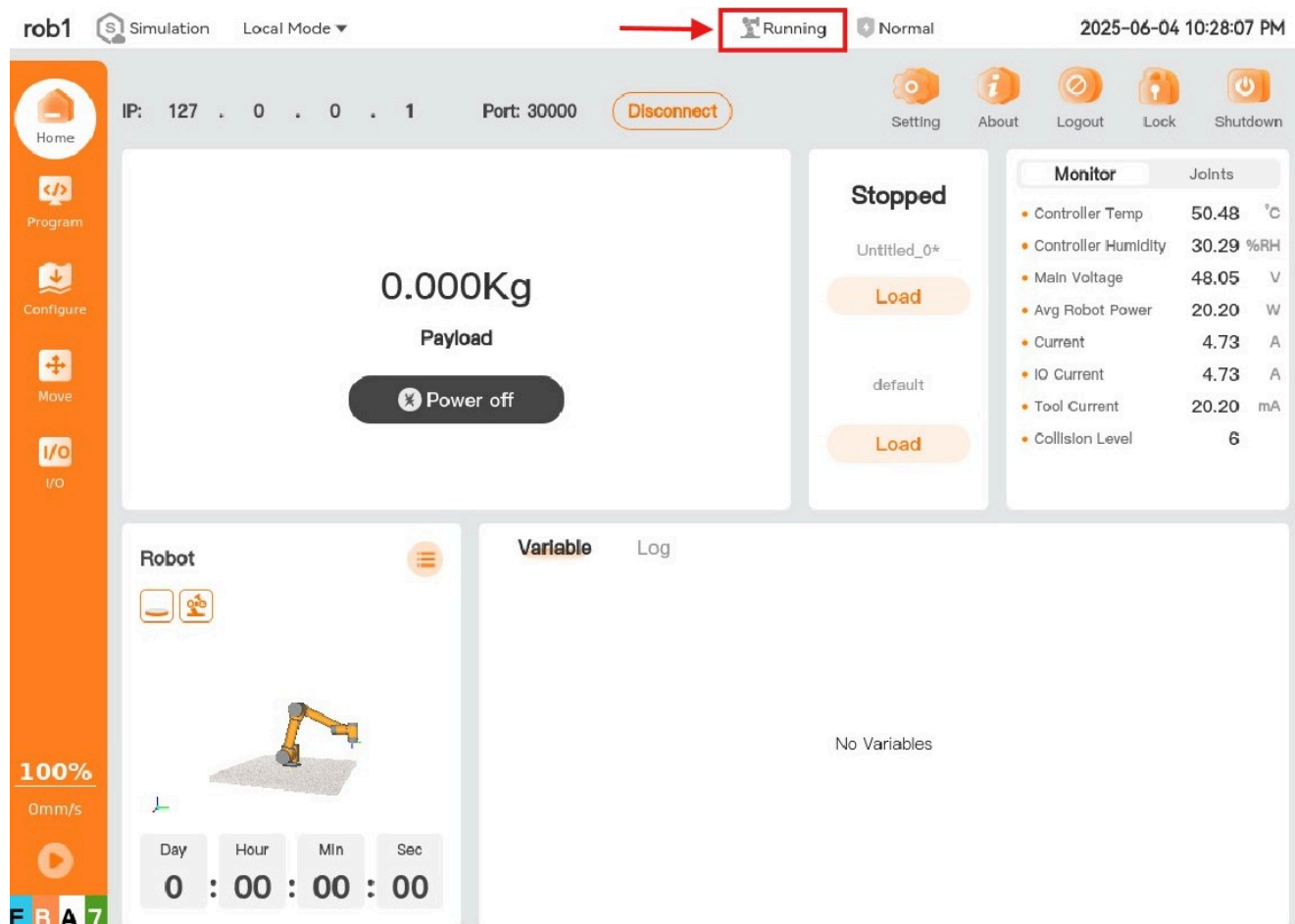
8. The terminal prints information, indicating that the robot arm is powered on and the brake is released successfully.

```

Terminal
机械臂当前模式:RobotModeType.Bootin
机械臂当前模式:RobotModeType.Bootin
机械臂当前模式:RobotModeType.Bootin
机械臂当前模式:RobotModeType.Bootin
机械臂当前模式:RobotModeType.Bootin
机械臂当前模式:RobotModeType.Bootin
机械臂当前模式:RobotModeType.Bootin
机械臂当前模式:RobotModeType.Bootin
机械臂当前模式:RobotModeType.Bootin
机械臂当前模式:RobotModeType.Bootin
机械臂上电成功, 当前模式:RobotModeType.Idle
机械臂当前模式:RobotModeType.BrakeReleasing
机械臂当前模式:RobotModeType.BrakeReleasing
机械臂当前模式:RobotModeType.BrakeReleasing
机械臂当前模式:RobotModeType.BrakeReleasing
机械臂当前模式:RobotModeType.BrakeReleasing
机械臂当前模式:RobotModeType.BrakeReleasing
机械臂松刹车成功, 当前模式:RobotModeType.Running
[Info] [connector_asio.h:175] RPC Client logout
[Info] [connector_asio.h:201] RPC client disconnected.
Press <RETURN> to close this window...

```

9. The status of the robot arm changes to **Running**, indicating that the robot arm is powered on and the brake is released successfully.

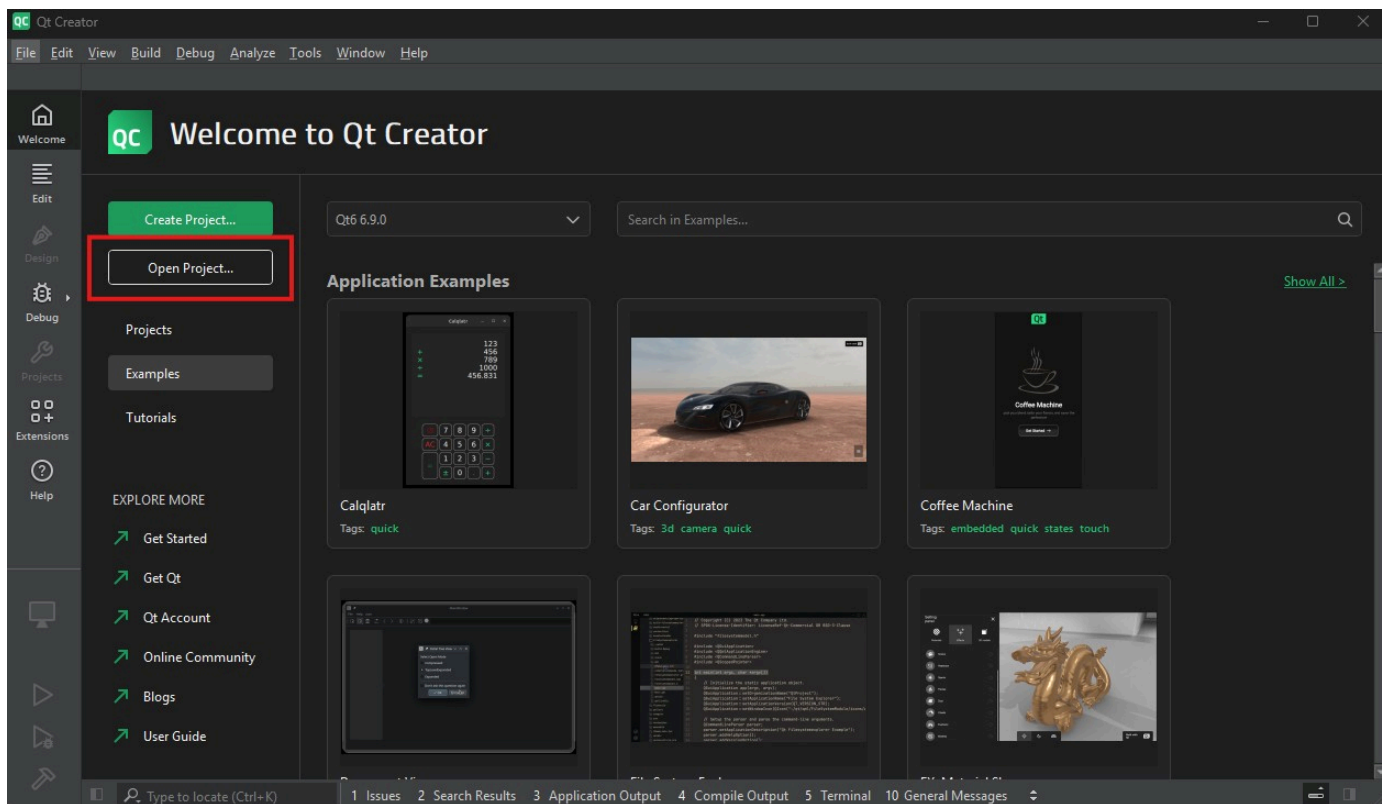


Compile and Run CMake projects with QtCreator in Windows Environment

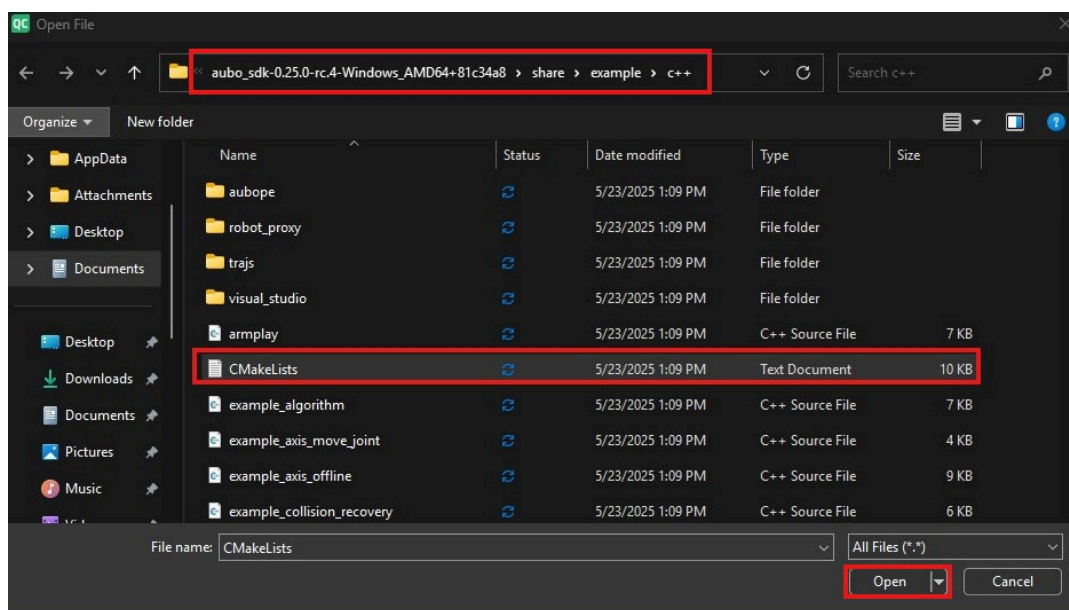
The following describes how to compile and run the `example_startup.cpp` example in the SDK using the QtCreator. After successful compilation and running, the teach pendant will be powered on successfully.

****Note:** To ensure successful communication between the client of the SDK and the robot arm, the IP address in the code must be changed to the IP address of the Aubo Sim virtual machine or the real robot arm before compilation and running. ******

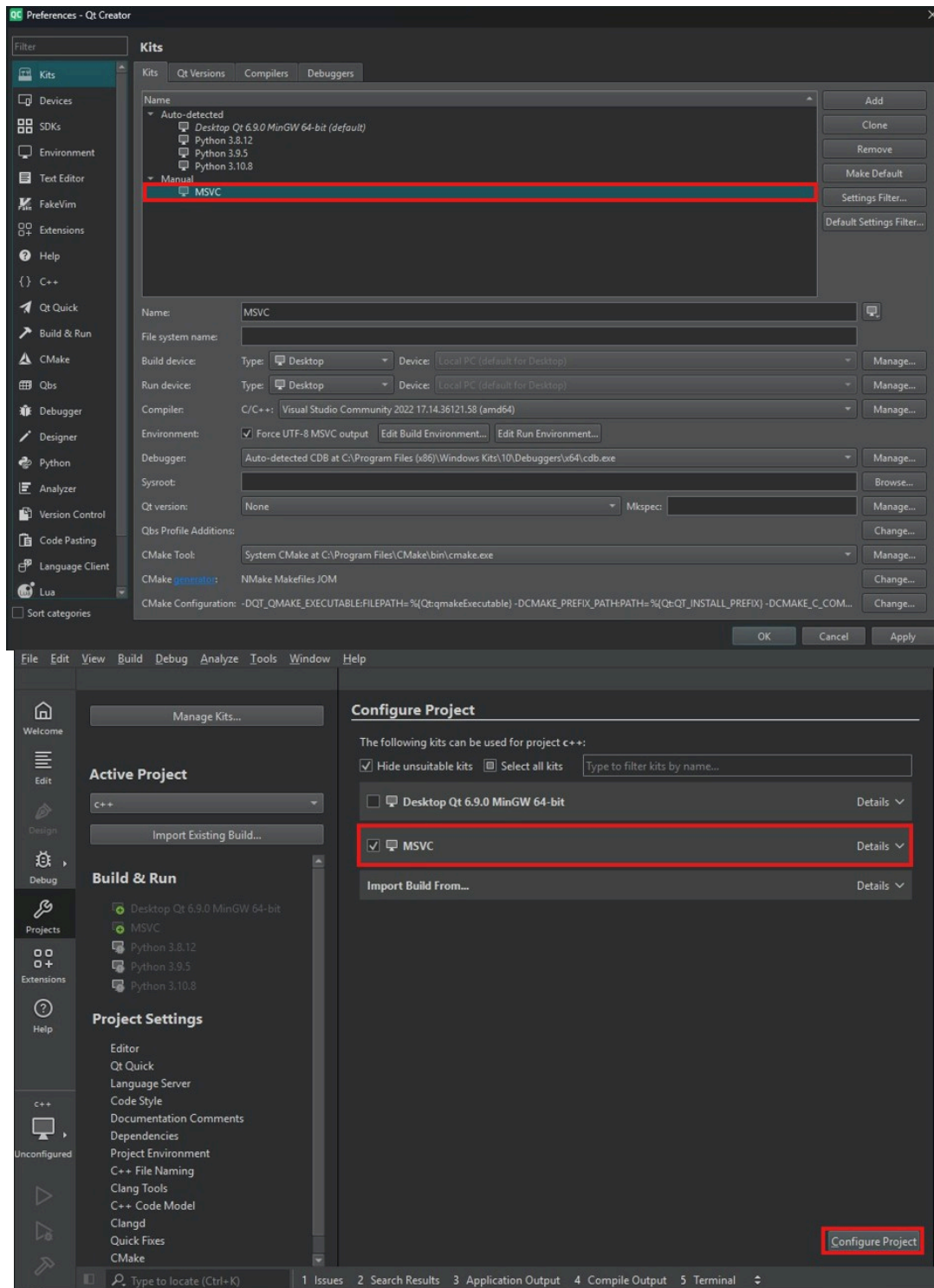
1. Open the QtCreator and click "Open".



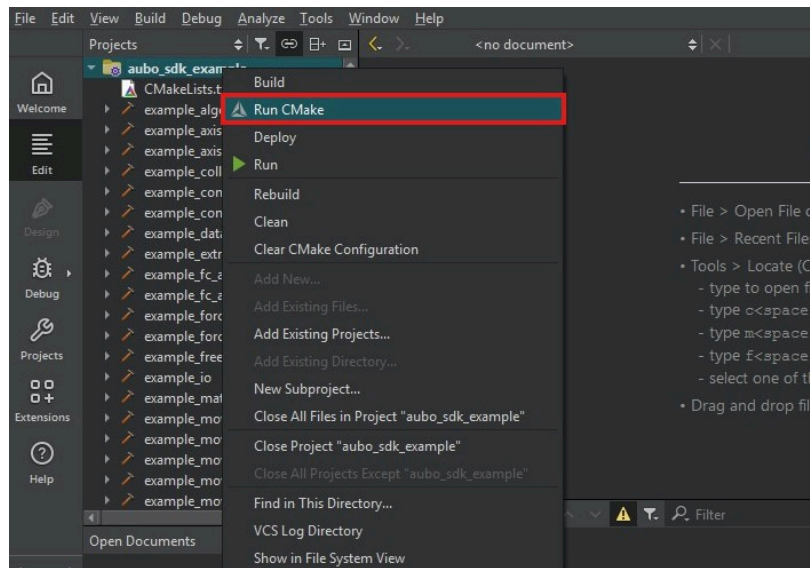
2. Locate the path of the C++ example in the SDK, select the CMakeLists.txt, and click "Open".



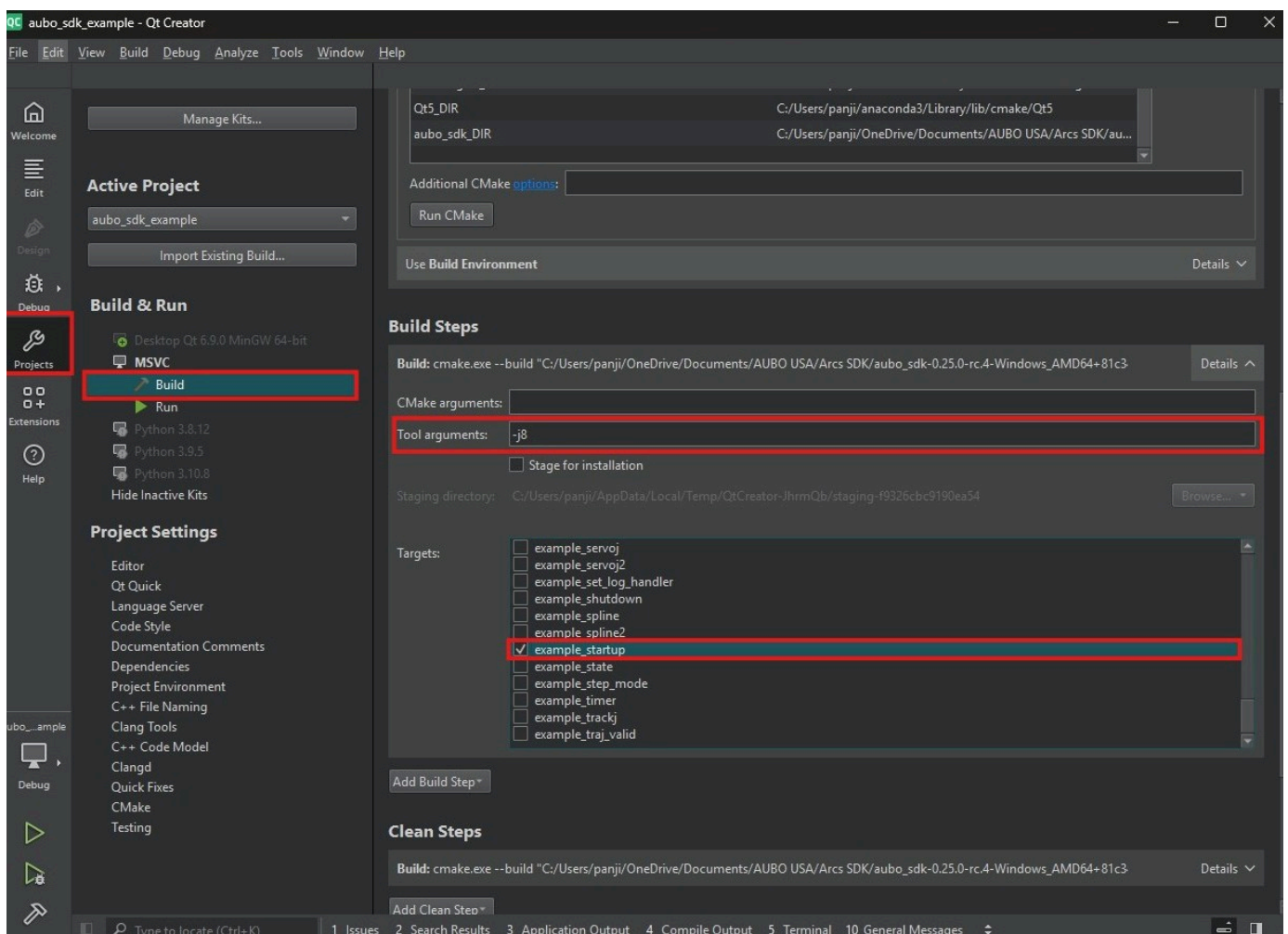
3. Configure the compiler.



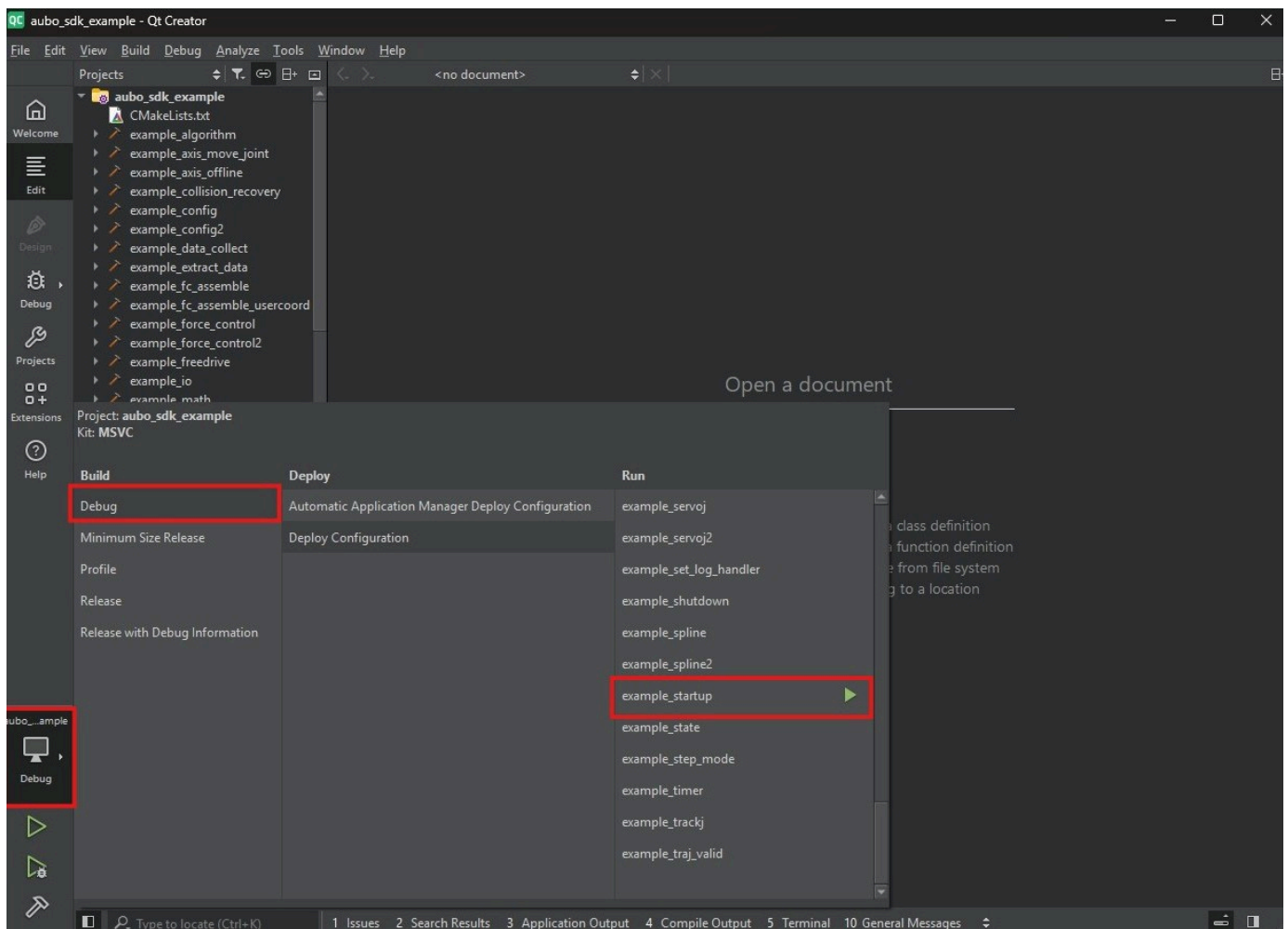
4. Select and right-click the `aubo_sdk_example`, and click "Run CMake".



5. In the Build step, enter "-j8" in the Tool arguments field and select example_startup.



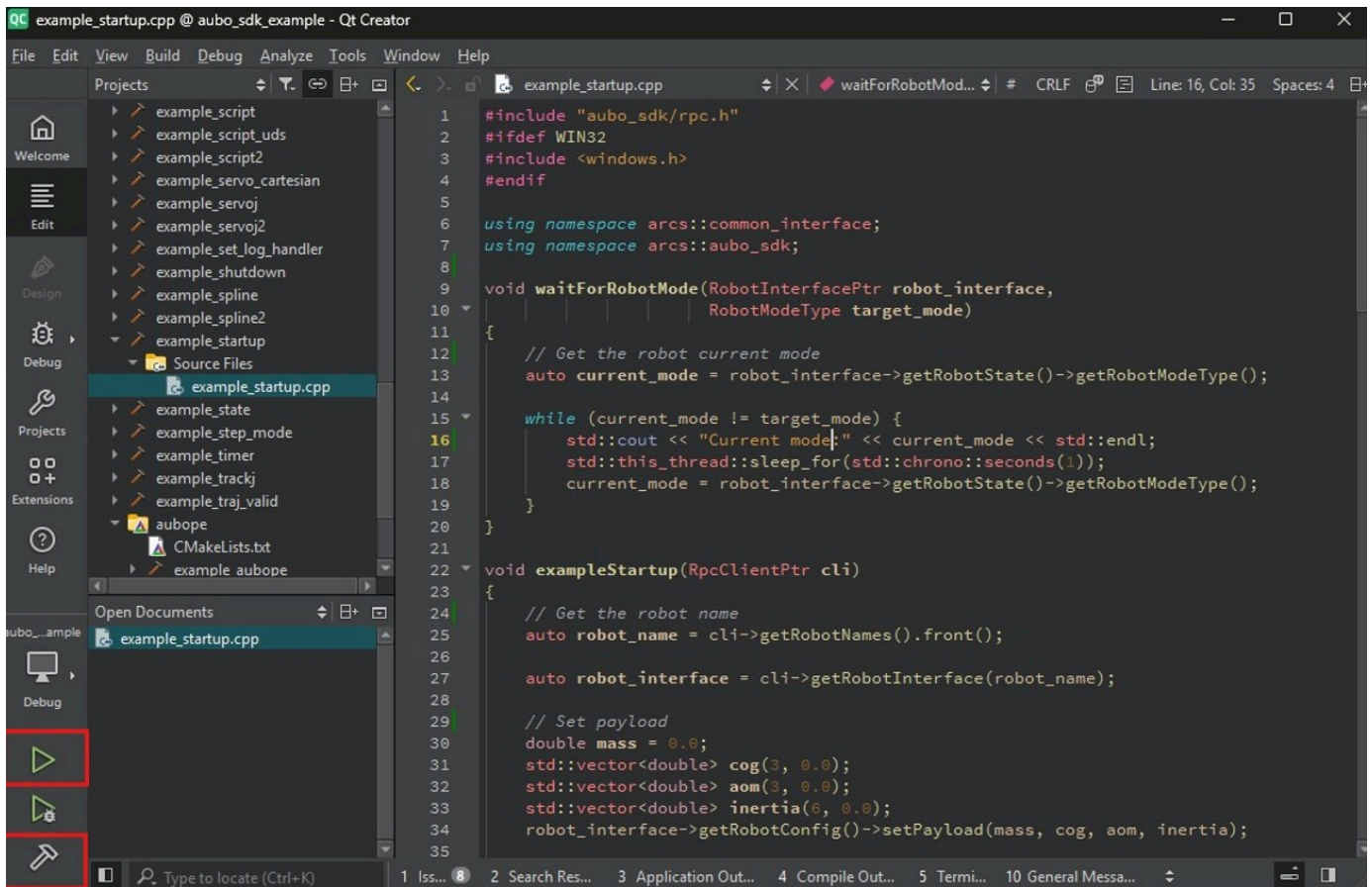
6. Select the example_startup.



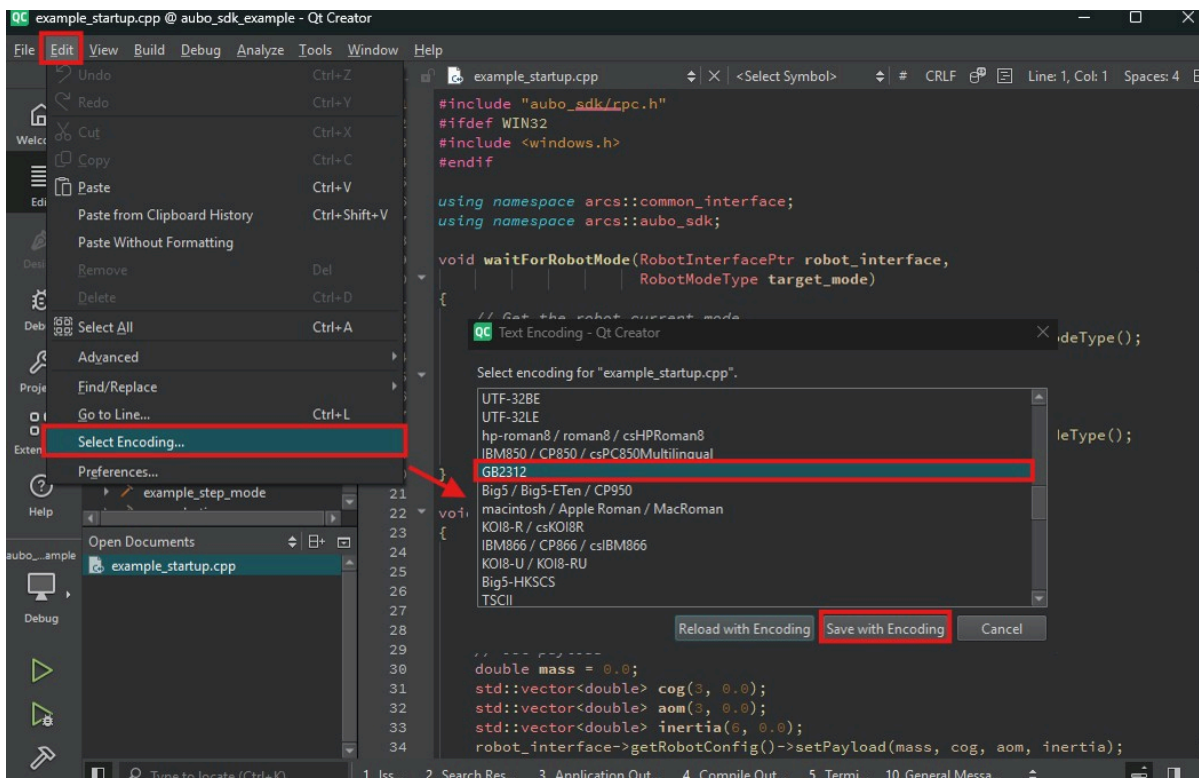
7. Compile and run the codes.

```
Terminal
[Info] [rpc.cpp:413] SDK INTERFACE VERSION: 0.19.6
[Info] [rpc.cpp:414] SERVER INTERFACE VERSION: 0.19.6
[Info] [rpc.cpp:139] RPC Client login
正在上电:3
正在上电:2
正在上电:4
上电成功:5
正在松刹车:5
正在松刹车:6
松刹车成功:8
Press <RETURN> to close this window...
```

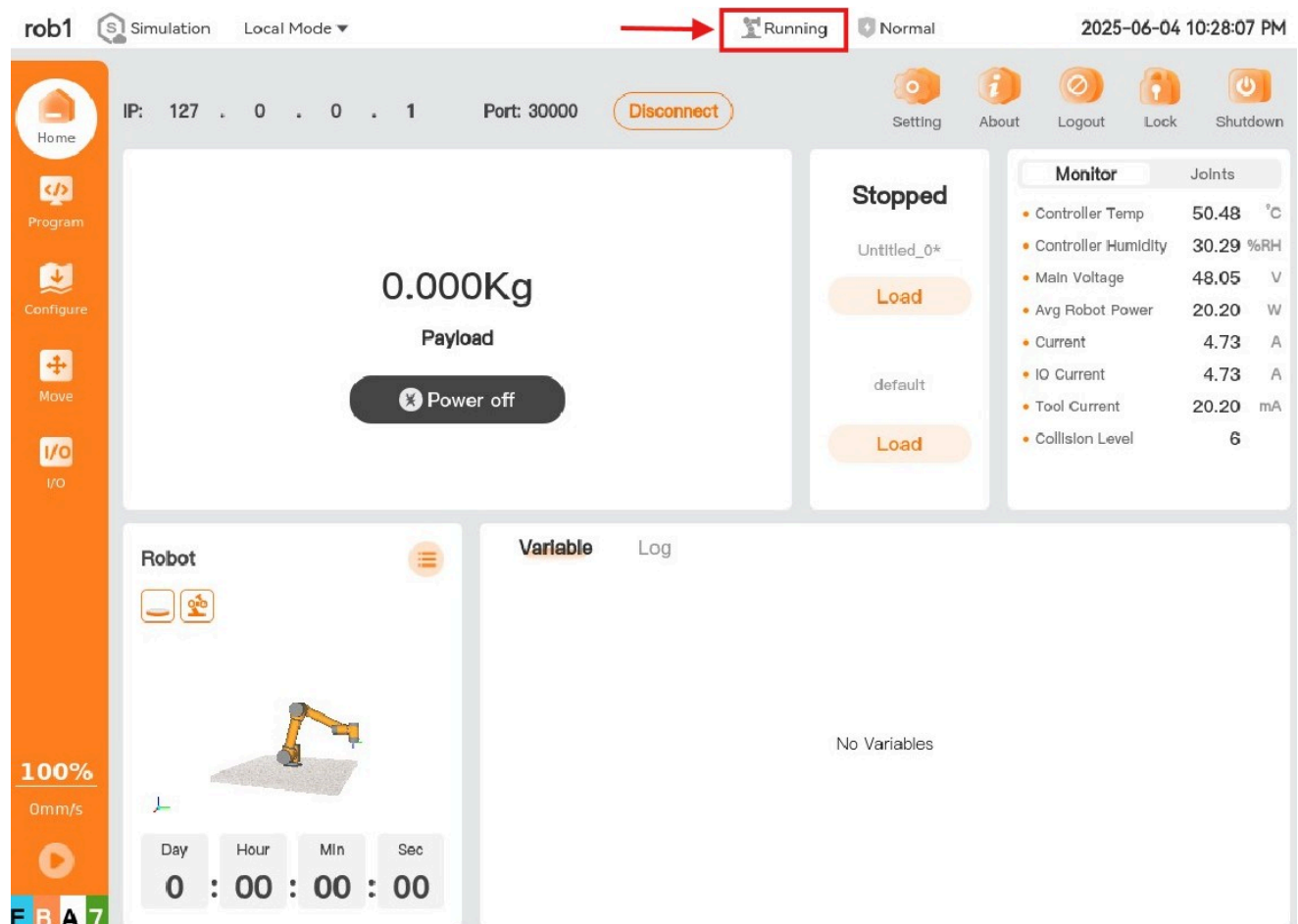
8. The console prints information.



If the information printed by the console is garbled Chinese characters, please set the codes as shown in the figure below, and then perform compilation and running again.



9. The teach pendant shows successful power-on.

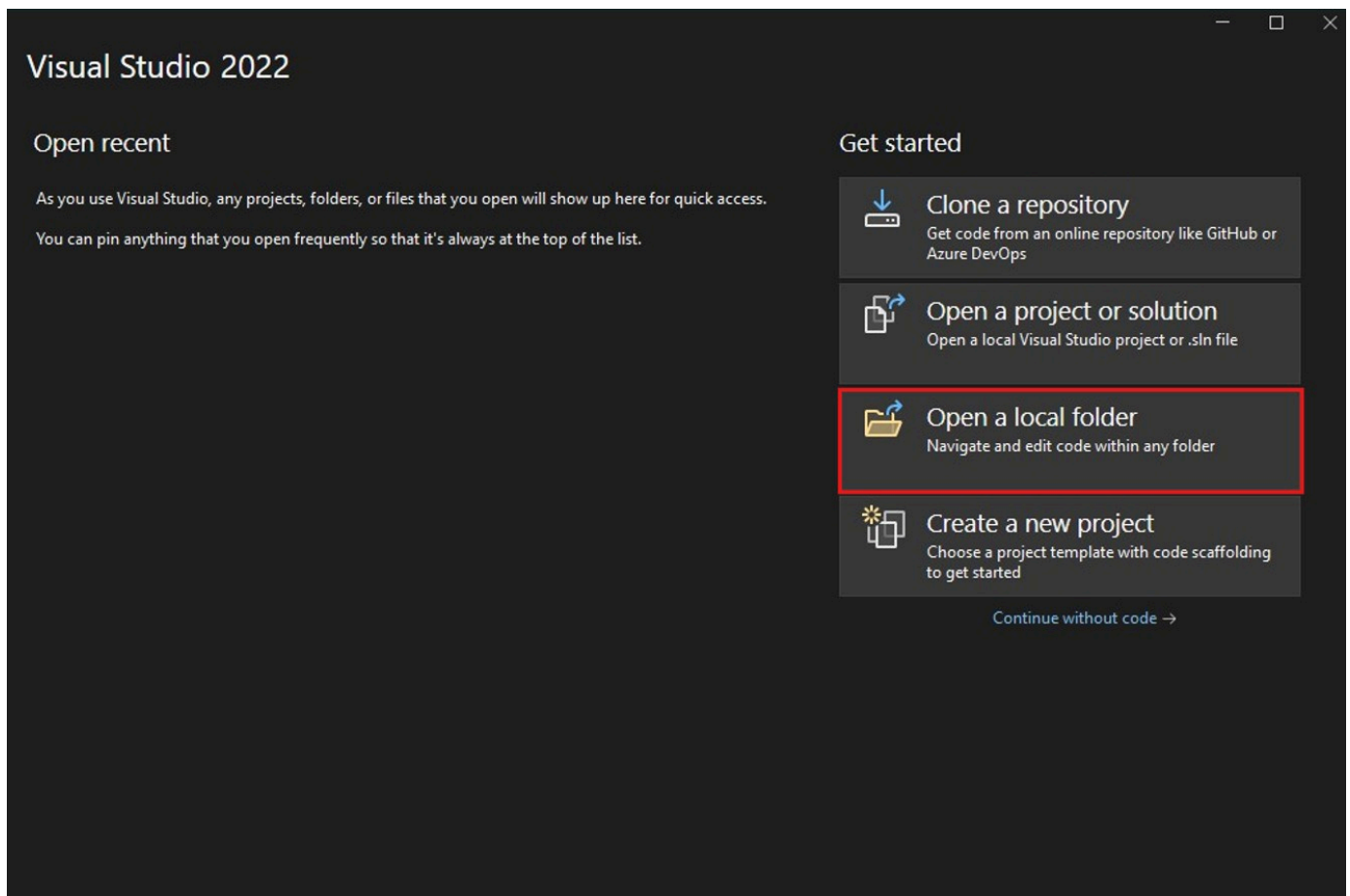


Compile and Run CMake Projects with Visual Studio in Windows Environment

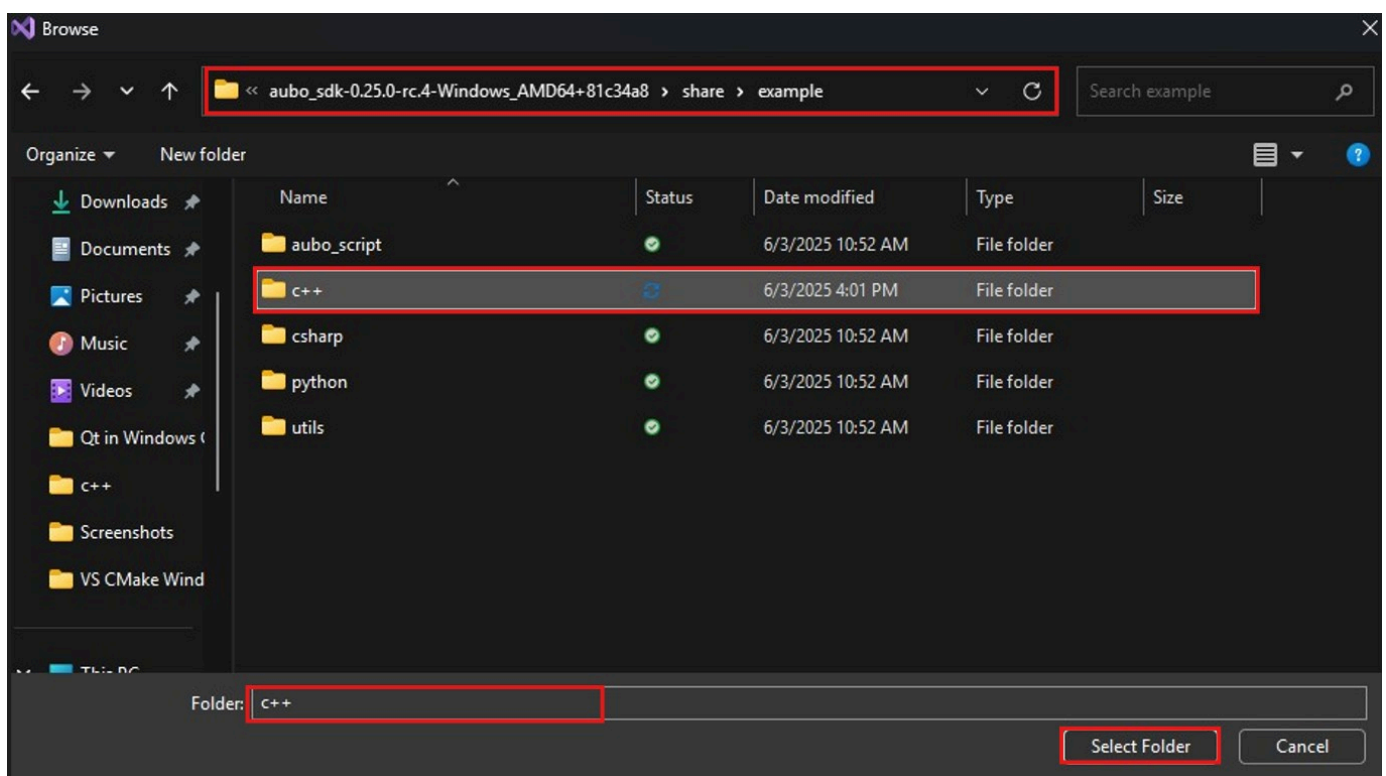
The following describes how to compile and run the `example_startup.cpp` example in the SDK using the Visual Studio. After successful compilation and running, the teach pendant will be powered on successfully.

****Note:** To ensure successful communication between the client of the SDK and the robot arm, the IP address in the code must be changed to the IP address of the Aubo Sim virtual machine or the real robot arm before compilation and running. ******

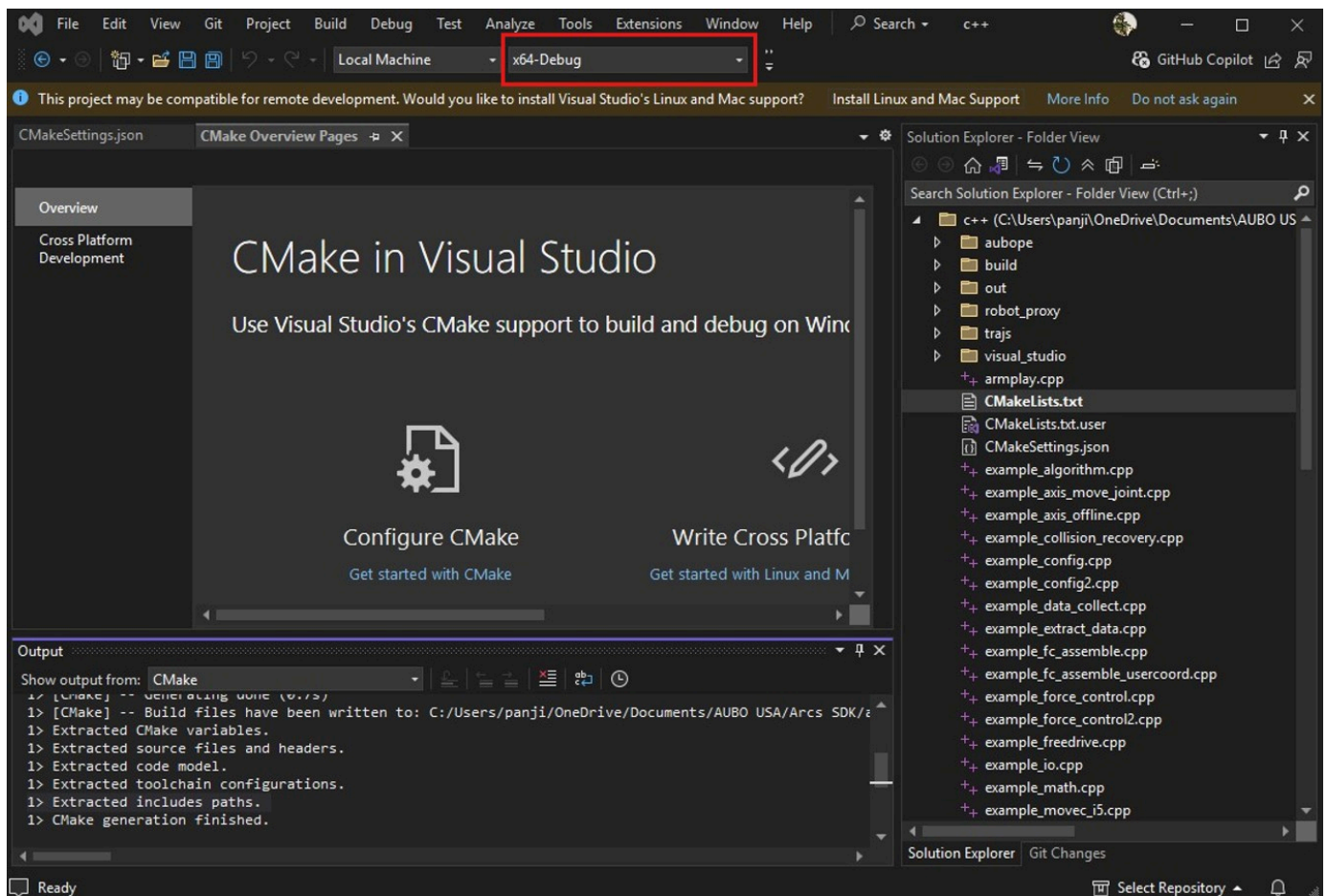
1. Open the Visual Studio and click "Open Local Folder" .



2. Find the path where the C++ examples are located in the SDK, select the “c++” folder, and click "Select Folder".

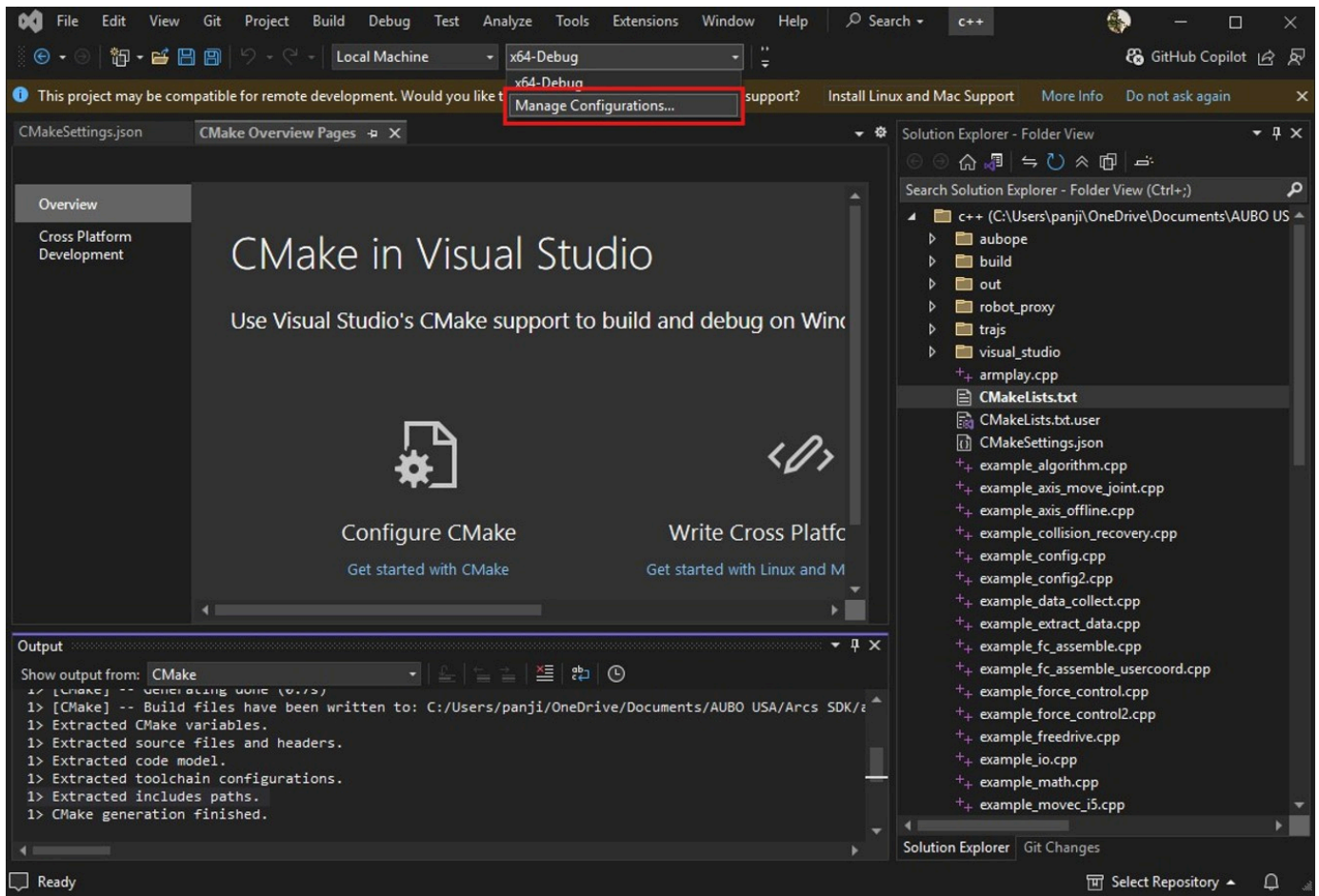


3. Build the configuration, and select "x64-Release".

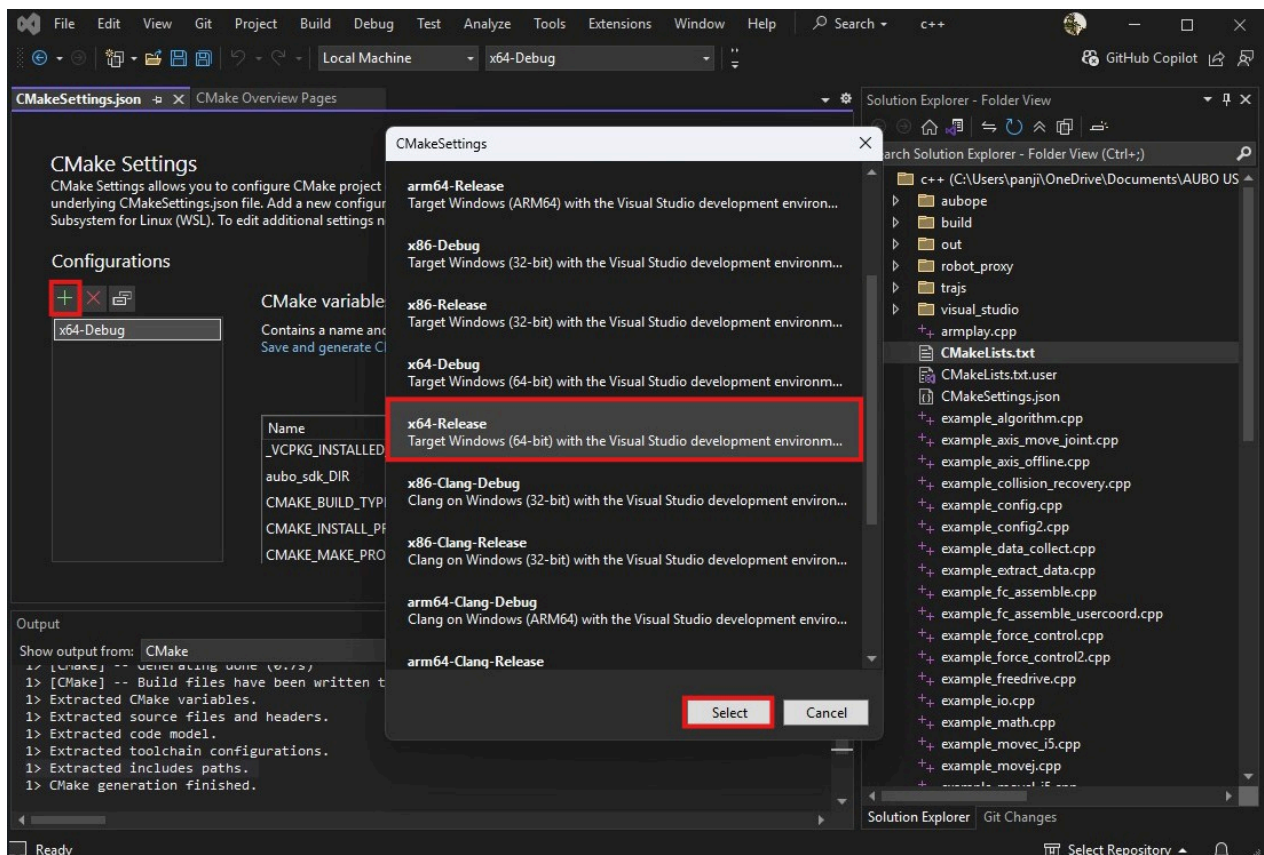


If there is no Release mode in the VS, follow the steps below:

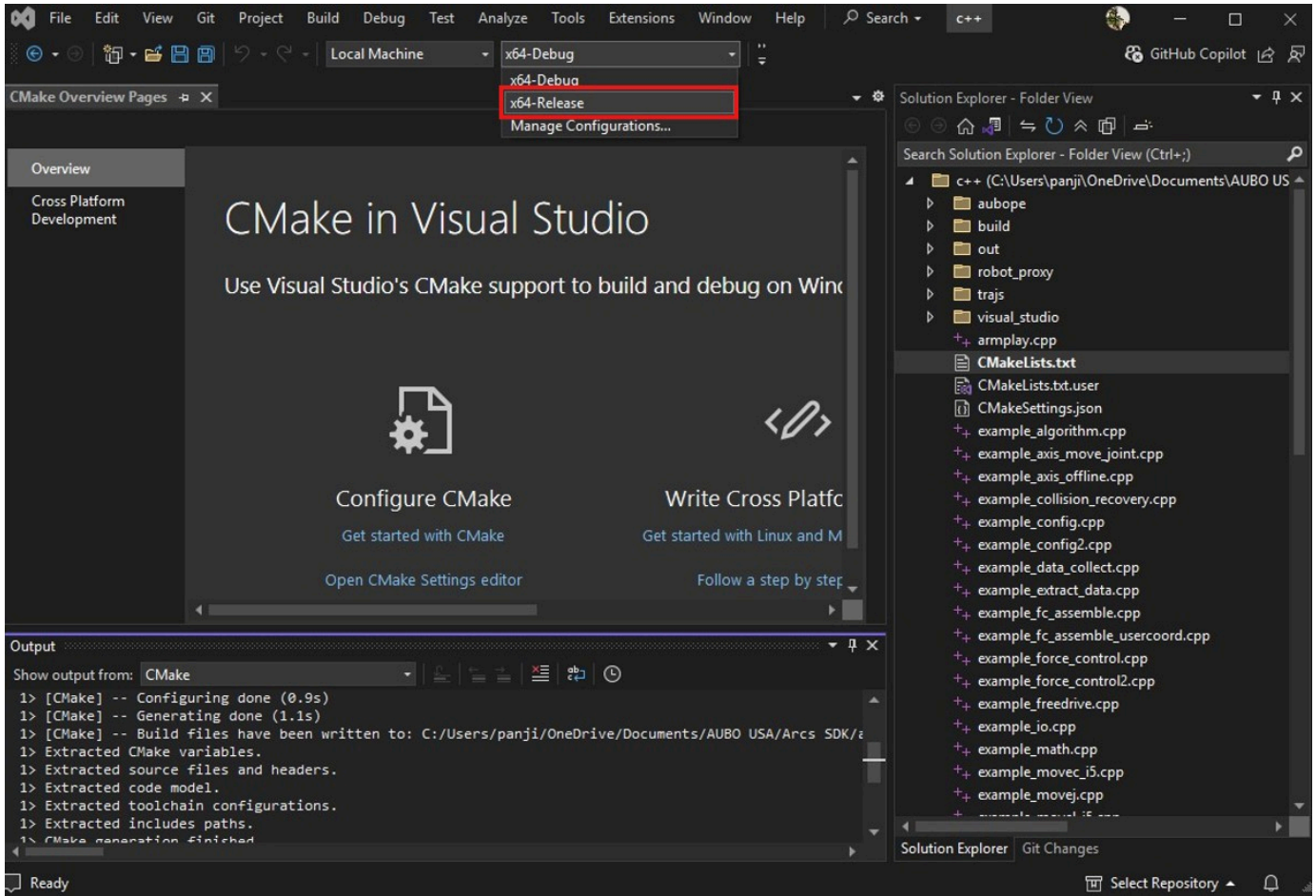
Step 1: Click "Manage Configuration".



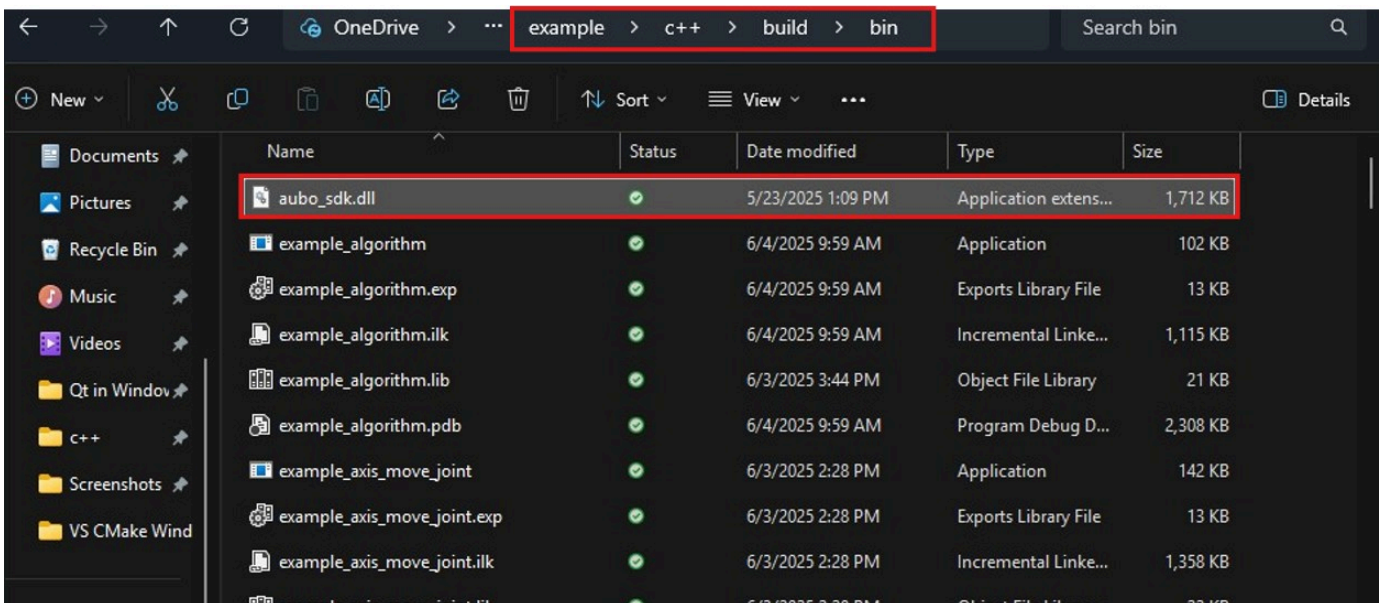
Step 2: In the CMakeSettings.json file, first click the "+" icon, then select "x64-Release", and click "Select".



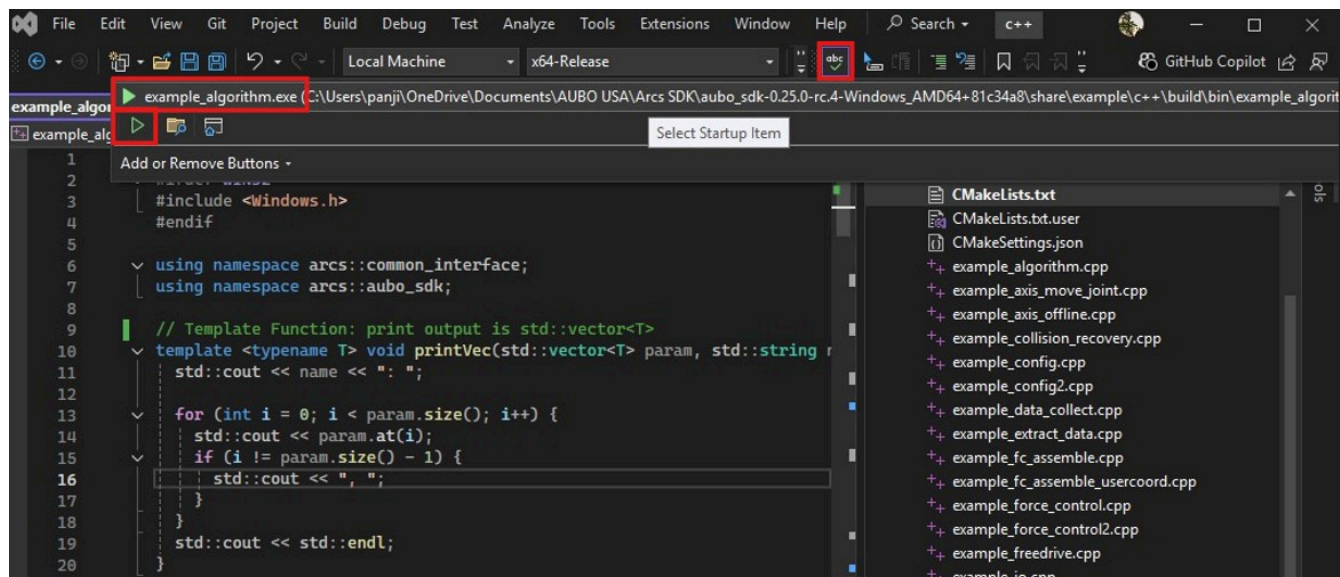
As shown in the figure below, the "x64-Release" configuration is successful.



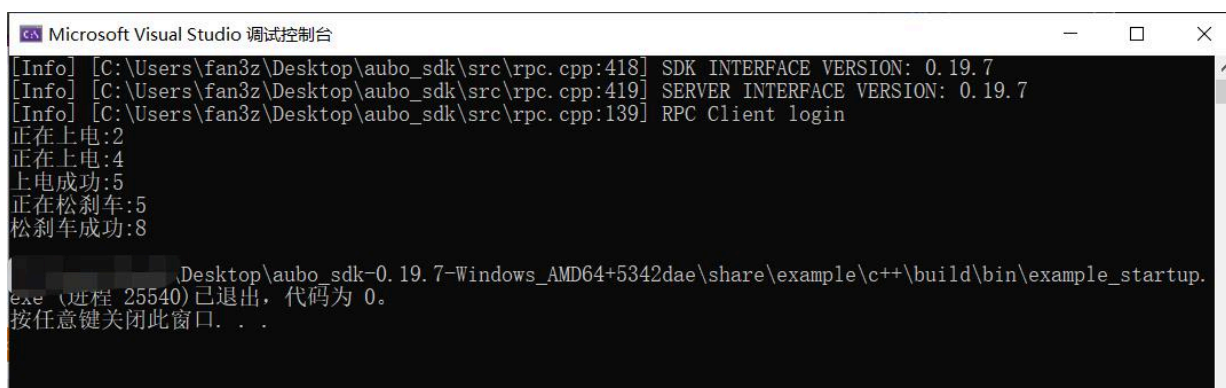
4. If an error stating that the aubo_sdk.dll file cannot be found is reported during the compilation, move the aubo_sdk.dll to the executable program directory, that is, ../example/c++/build/bin.



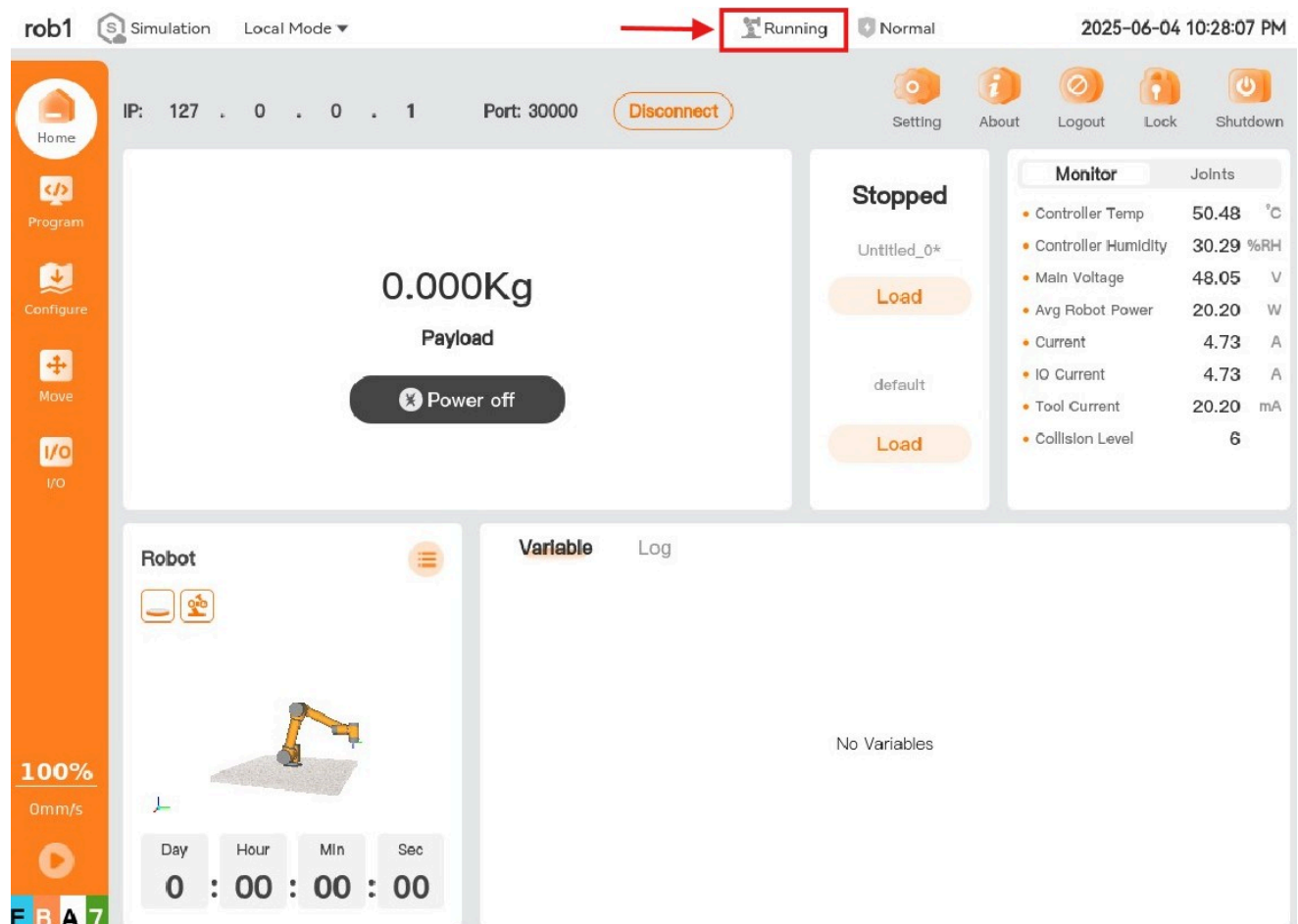
5. Select the example_startup.exe and perform compilation and running.



6. The console prints information.



7. The teach pendant shows successful power-on.

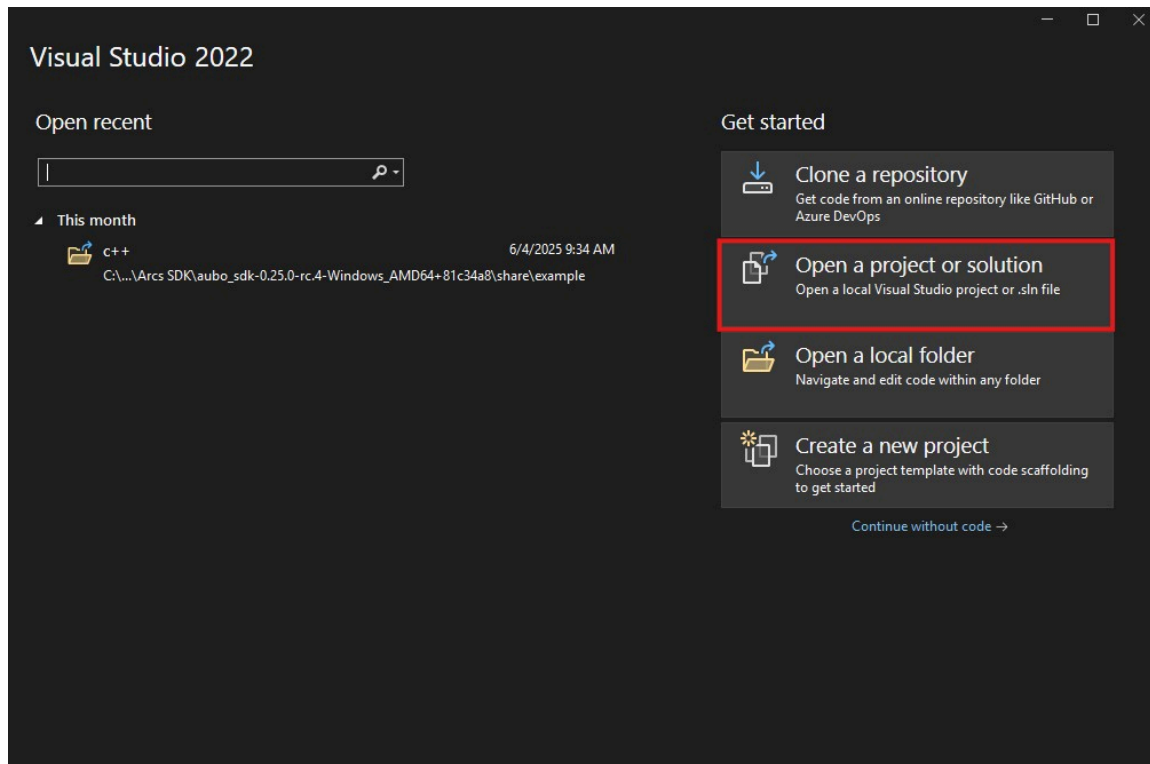


Compile and Run non-CMake Projects with Visual Studio in Windows Environment

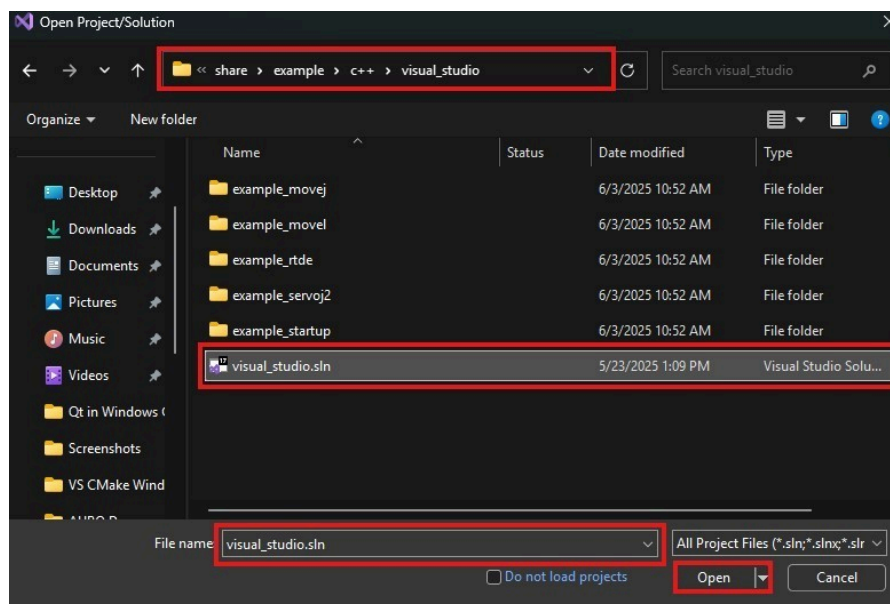
The following describes how to compile and run the example_startup.cpp example in the SDK using the Visual Studio. After successful compilation and running, the teach pendant will be powered on successfully.

****Note:** To ensure successful communication between the client of the SDK and the robot arm, the IP address in the code must be changed to the IP address of the Aubo Sim virtual machine or the real robot arm before compilation and running. ******

1. Open the Visual Studio and click "Open Project or Solution".

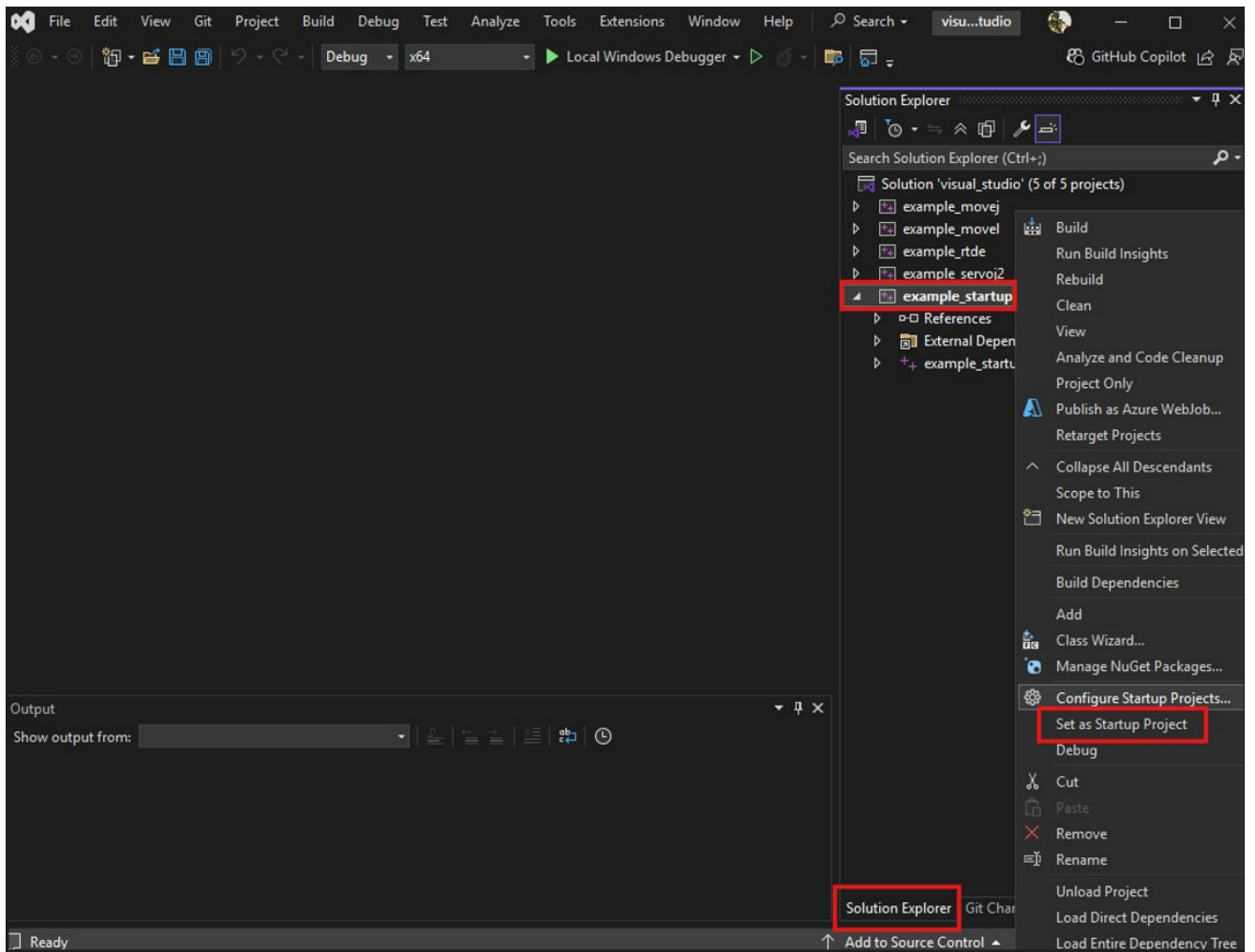


2. Find the path where the visual_studio.sln is located in the SDK, select the visual_studio.sln, and click "Open".

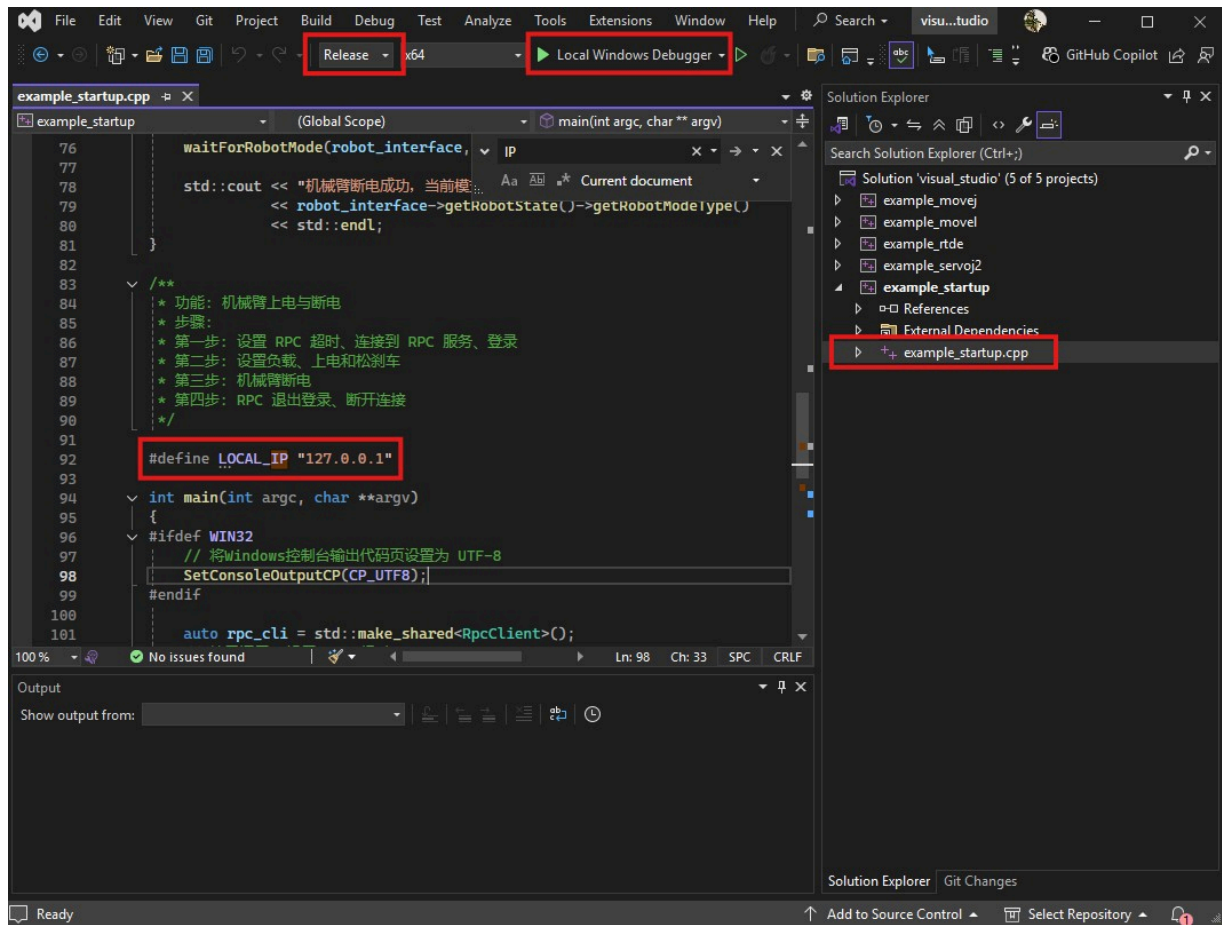


3. Set the project to be run as the startup project.

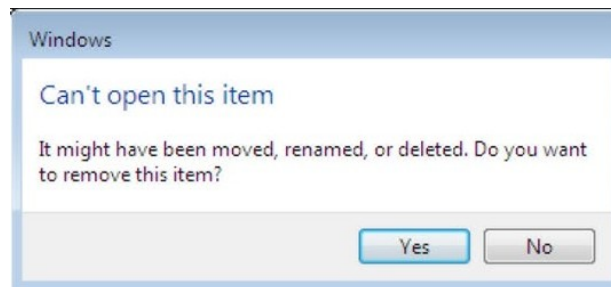
Taking the example_startup project as an example, in the "Solution Explorer", select the example_startup, right-click it, and click "Set as Startup Project".



4. Open the `example_startup.cpp` file, modify the robot's IP address, select "Release" or "Debug", and perform running and debugging.

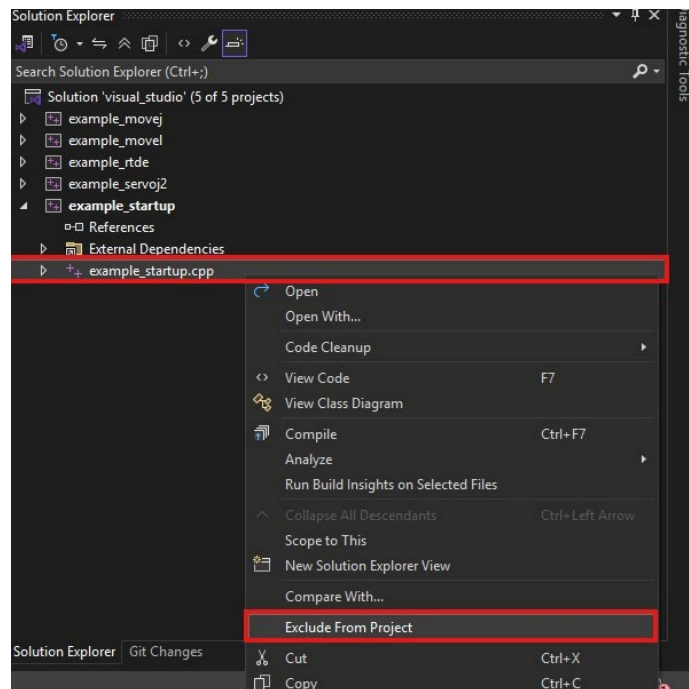


If the following error occurs during the opening of the example_startup.cpp file:

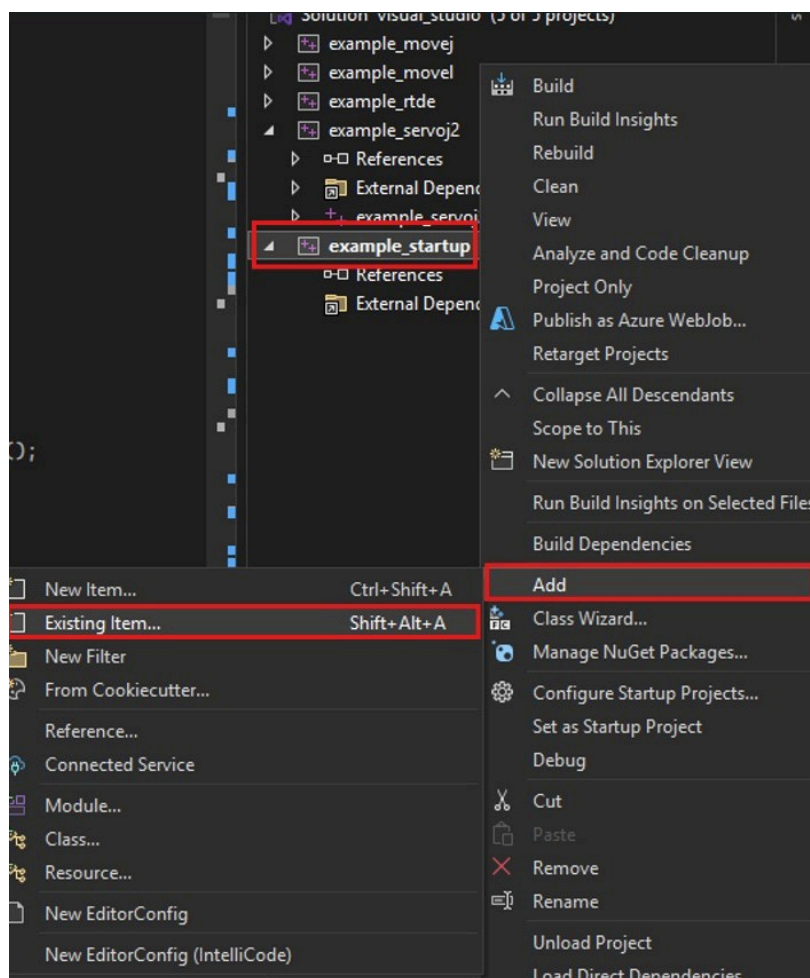


Re-add the example_startup.cpp file using the following method.

Select the example_startup.cpp file, right-click it, and click "Exclude from Project".



Select the example_startup.cpp file, right-click it, click "Add", and then click "Existing Item".



Find and select the example_startup.cpp file, and click "Add".

rob1

Simulation

Local Mode

Running

Normal

2025-06-04 10:28:07 PM

Home

Program

Configure

Move

I/O

100%

0mm/s

F

B

A

7

IP: 127 . 0 . 0 . 1

Port: 30000

Disconnect

Setting

About

Logout

Lock

Shutdown

0.000Kg

Payload

Power off

Stopped

Untitled_0*

Load

default

Load

Monitor

Joints	
Controller Temp	50.48 °C
Controller Humidity	30.29 %RH
Main Voltage	48.05 V
Avg Robot Power	20.20 W
Current	4.73 A
IO Current	4.73 A
Tool Current	20.20 mA
Collision Level	6

Robot

Day

Hour

Min

Sec

0

:

00

:

00

:

00

Variable

Log

No Variables

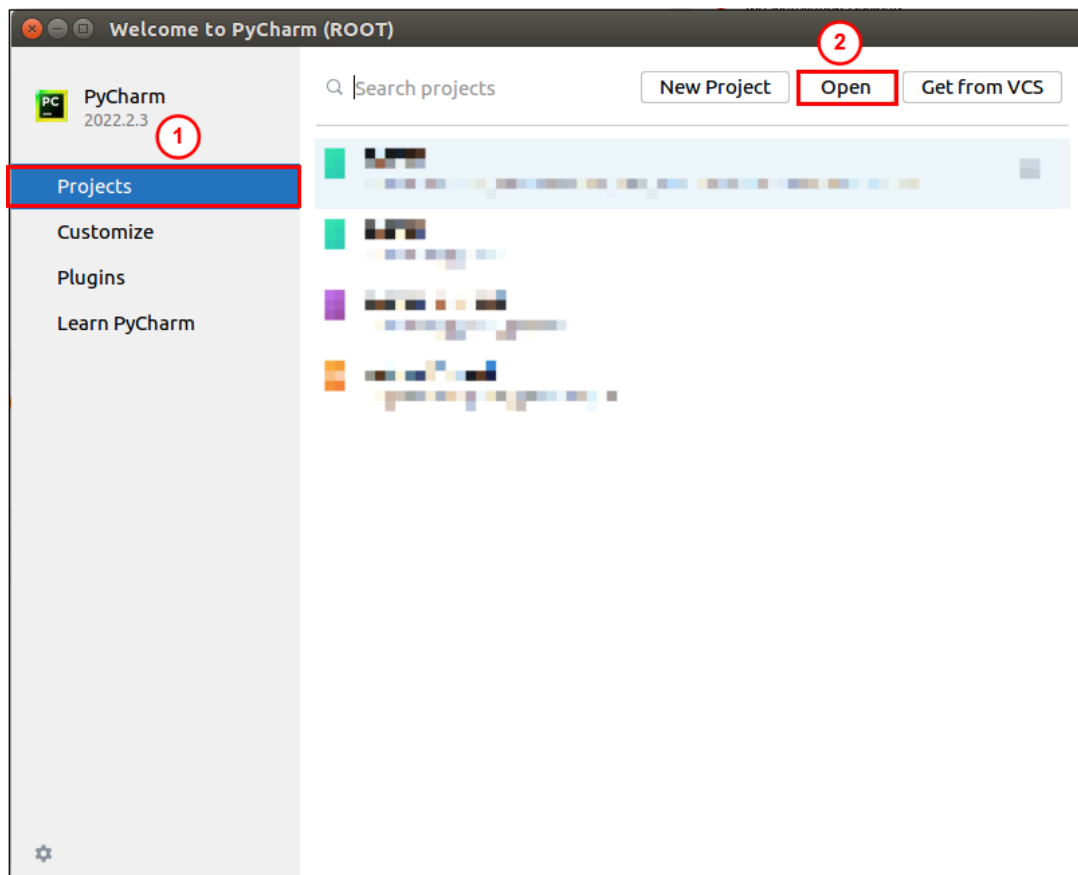
Examples of Running Python SDK

In Linux Environment

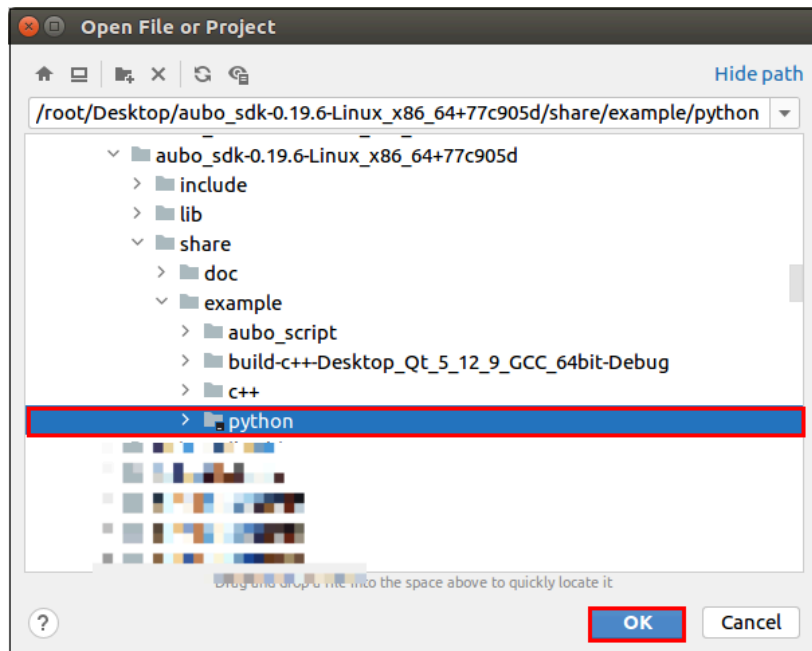
The following describes how to compile and run the `example_startup.py` example in the SDK using the PyCharm. After successful running, the teach pendant will be powered on and then powered off.

****Note:** If the client of the SDK is connected to a real robot arm, the IP address in the code must be set to the robot arm's IP address. If the project is run on the Aubo Sim virtual machine and the client of the SDK is connected to the virtual machine, the IP address should be 127.0.0.1. ******

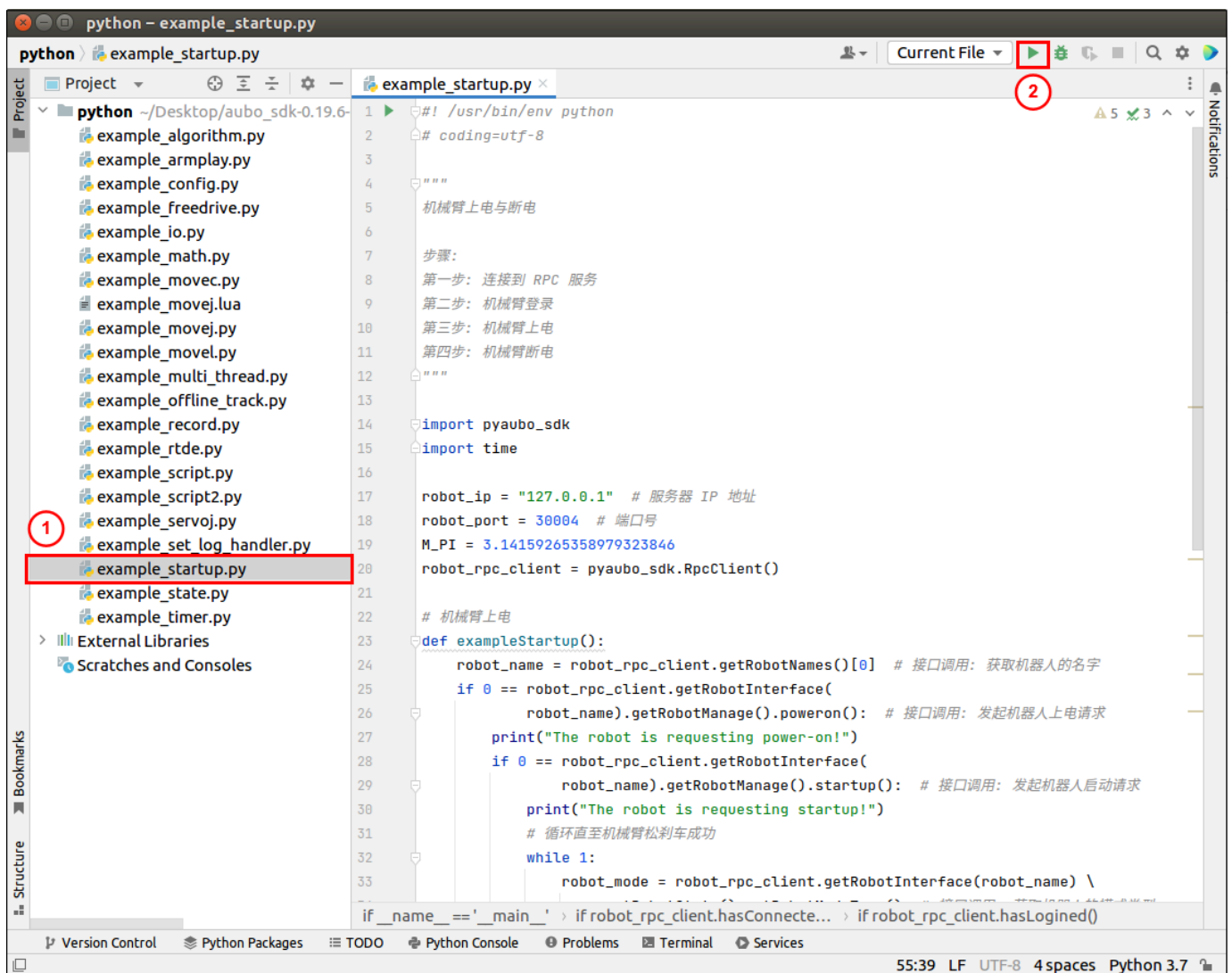
1. Open the PyCharm.



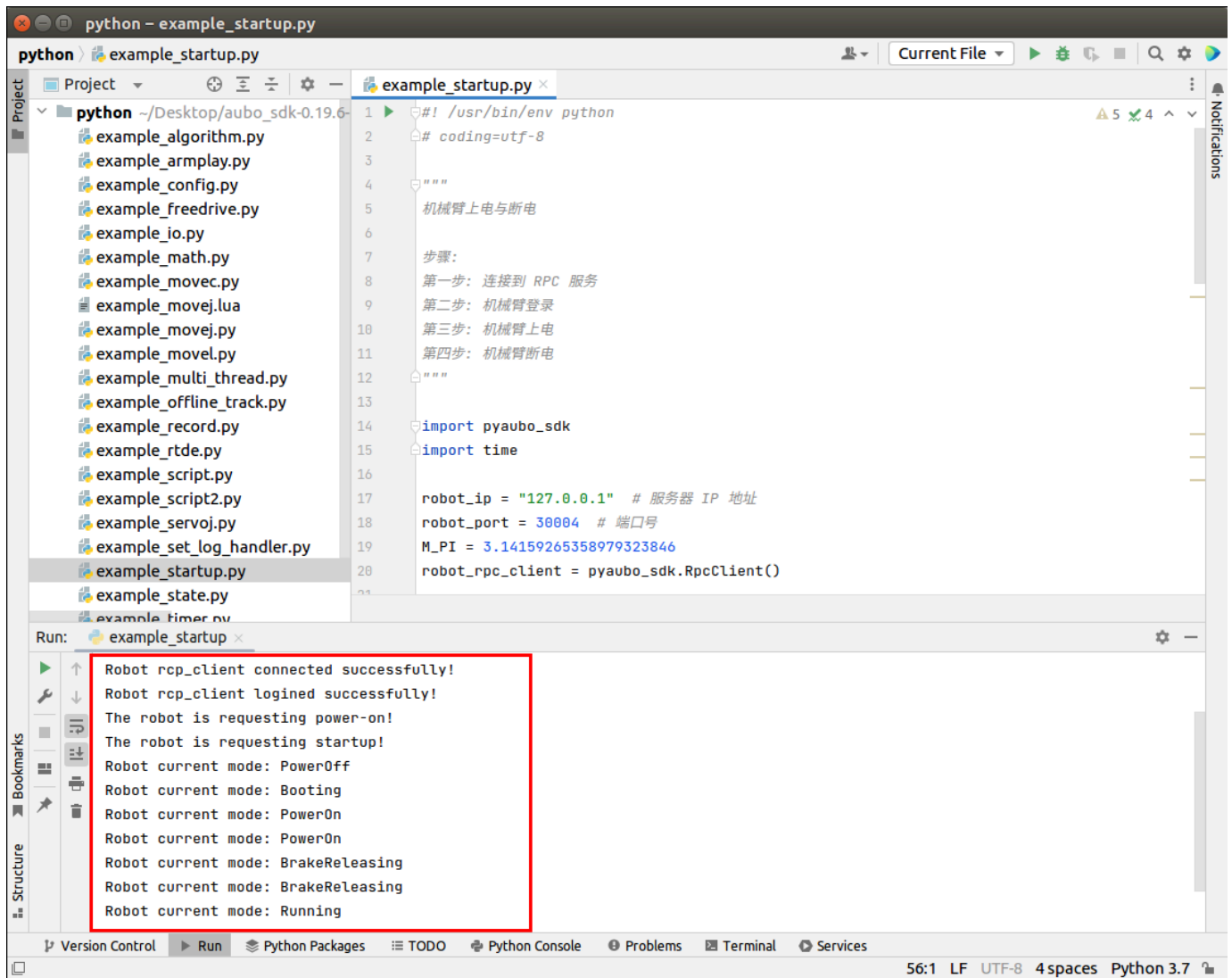
2. Locate the directory where the Python examples are stored, select the python, and click "OK".



3. Select the example_startup.py file, and click the "Run" button.



4. Check the running results of the program.



```
python - example_startup.py
python ~/Desktop/aubo_sdk-0.19.6
example_algorithm.py
example_armplay.py
example_config.py
example_freedrive.py
example_io.py
example_math.py
example_movec.py
example_movej.lua
example_movej.py
example_model.py
example_multi_thread.py
example_offline_track.py
example_record.py
example_rtde.py
example_script.py
example_script2.py
example_servoj.py
example_set_log_handler.py
example_startup.py
example_state.py
example_timer.py

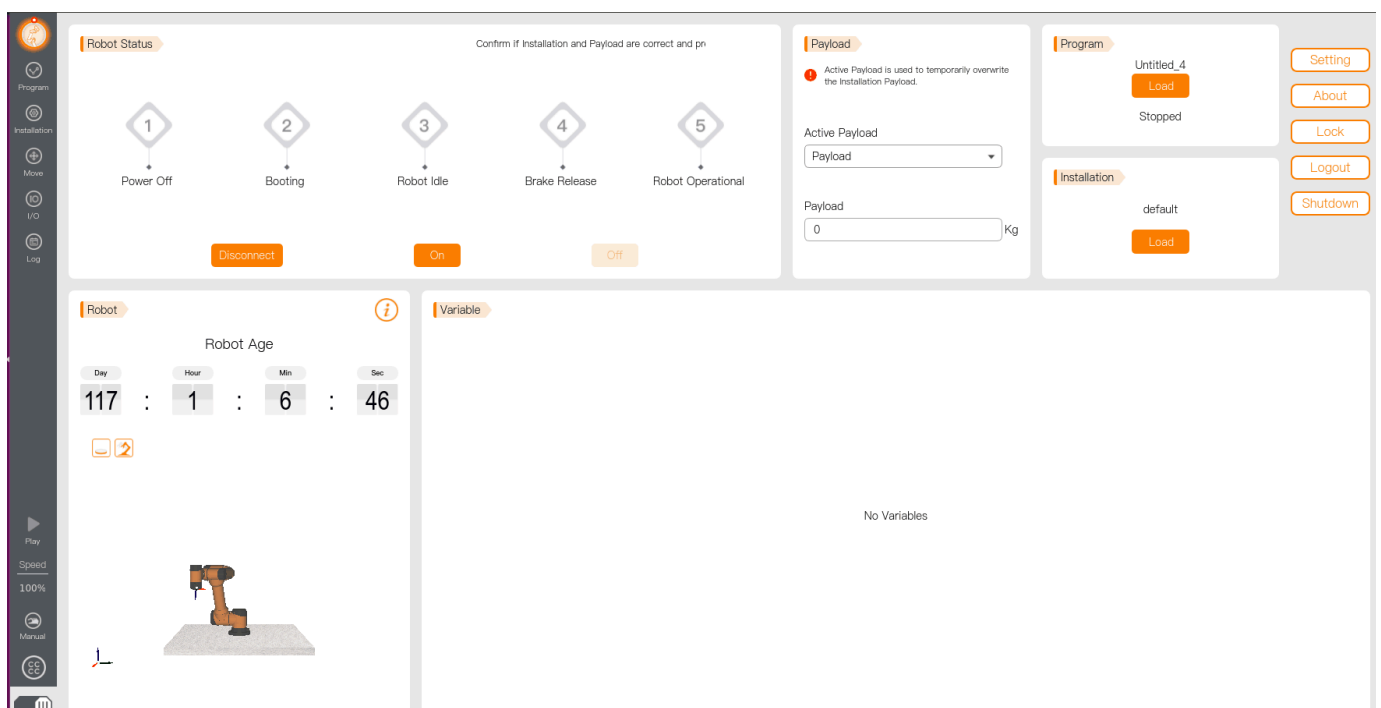
1  #!/usr/bin/env python
2  # coding=utf-8
3
4  """
5  机械臂上电与断电
6
7  步骤:
8  第一步: 连接到 RPC 服务
9  第二步: 机械臂登录
10  第三步: 机械臂上电
11  第四步: 机械臂断电
12  """
13
14  import pyaubo_sdk
15  import time
16
17  robot_ip = "127.0.0.1" # 服务器 IP 地址
18  robot_port = 30004 # 端口号
19  M_PI = 3.14159265358979323846
20  robot_rpc_client = pyaubo_sdk.RpcClient()
```

Run: example_startup x

```
Robot rpc_client connected successfully!
Robot rpc_client logged successfully!
The robot is requesting power-on!
The robot is requesting startup!
Robot current mode: PowerOff
Robot current mode: Booting
Robot current mode: PowerOn
Robot current mode: PowerOn
Robot current mode: BrakeReleasing
Robot current mode: BrakeReleasing
Robot current mode: Running
```

56:1 LF UTF-8 4 spaces Python 3.7

5. The robot arm is powered on first, and all the indicator lights on the teach pendant come on. Then the robot arm is powered off, and all the indicator lights go out.

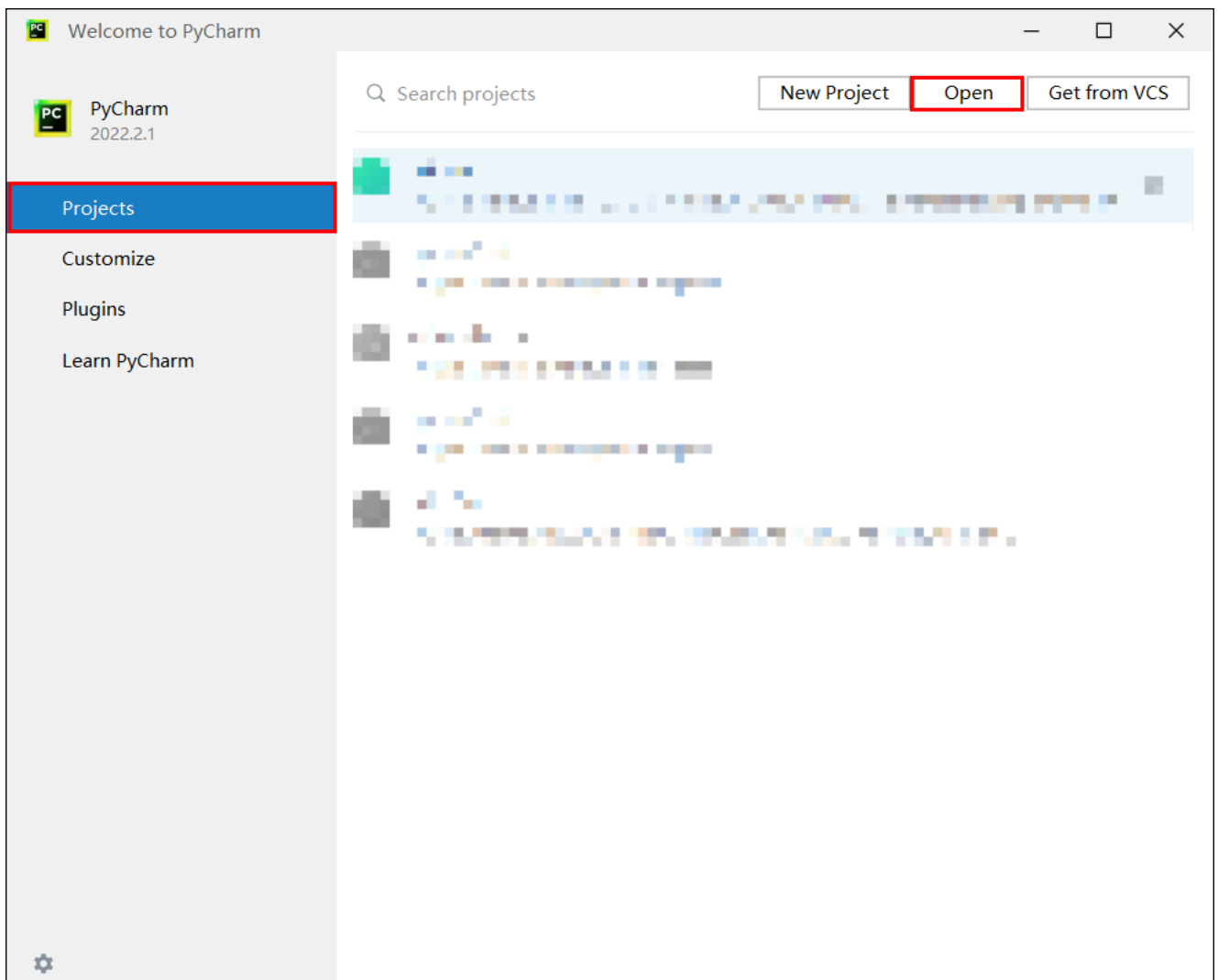


In Windows Environment

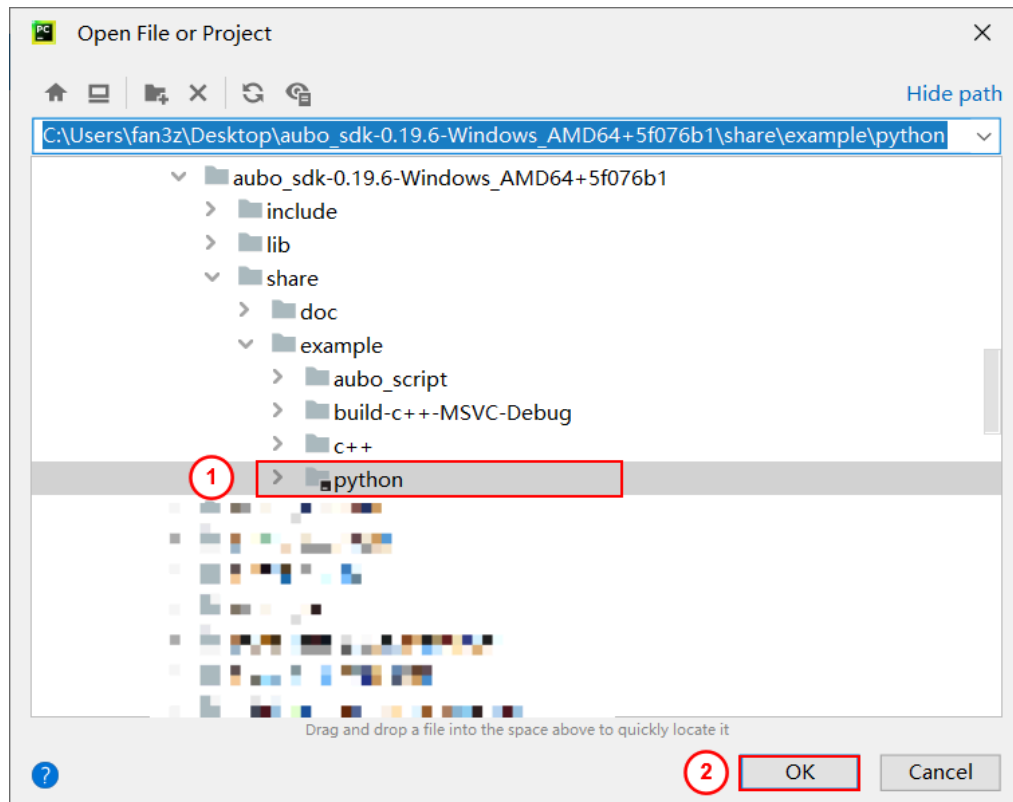
The following describes how to compile and run the `example_startup.py` example in the SDK using the PyCharm. After successful running, the teach pendant will be powered on and then powered off.

****Note:** To ensure successful communication between the client of the SDK and the robot arm, the IP address in the code must be changed to the IP address of the Aubo Sim virtual machine or the real robot arm before compilation and running. ******

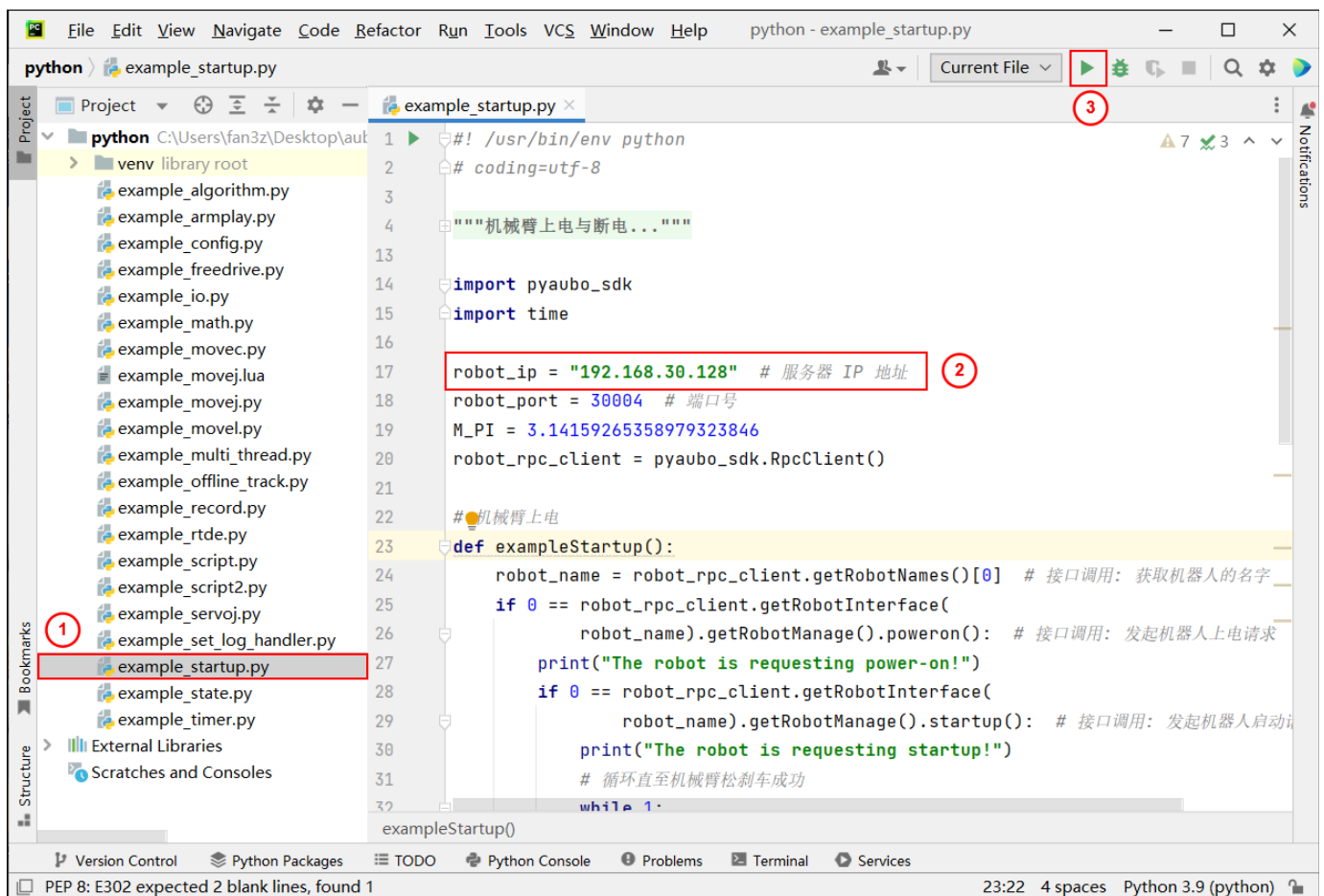
1. Open the PyCharm.



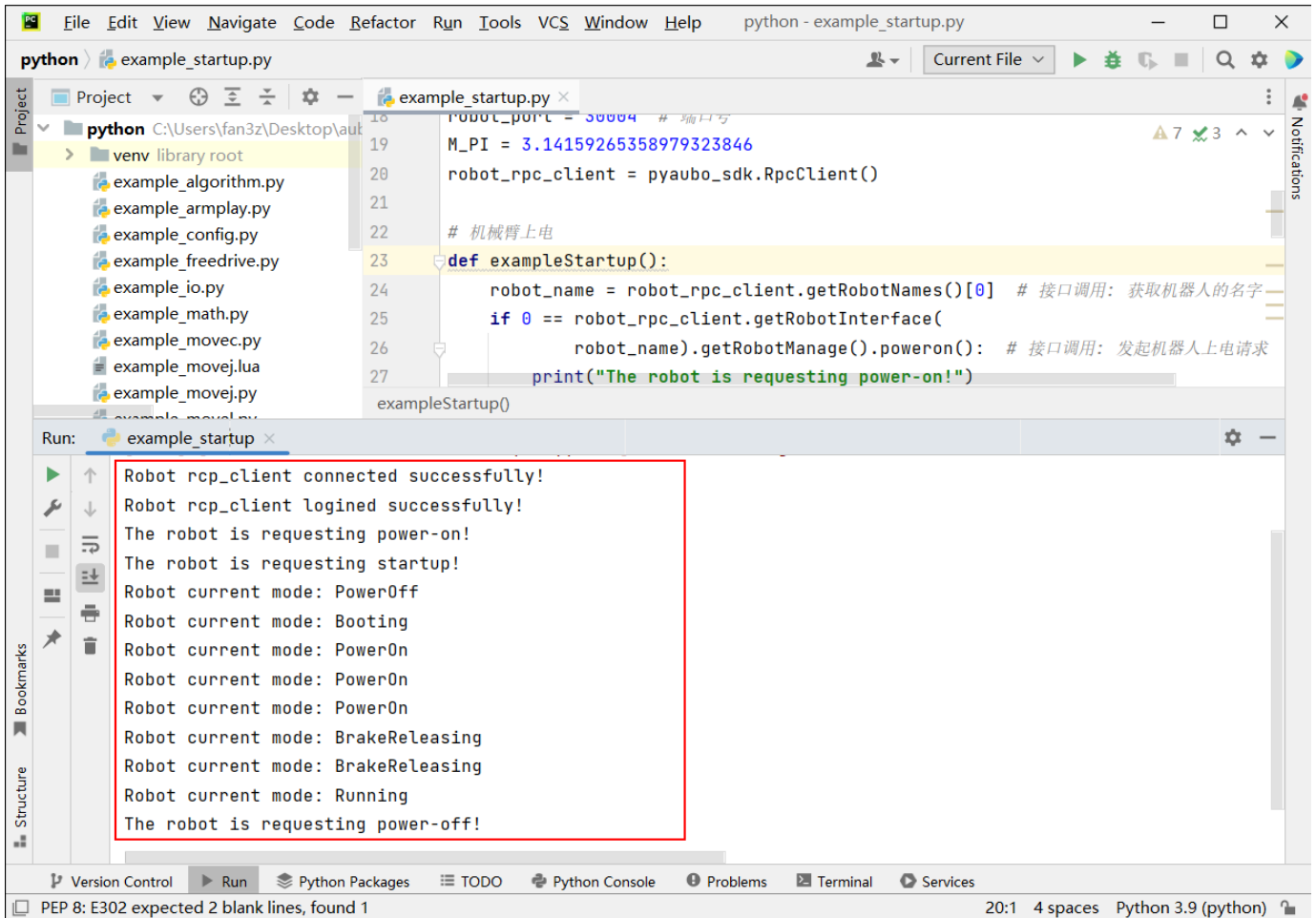
2. Locate the directory where the Python examples are stored, select the python, and click "OK".



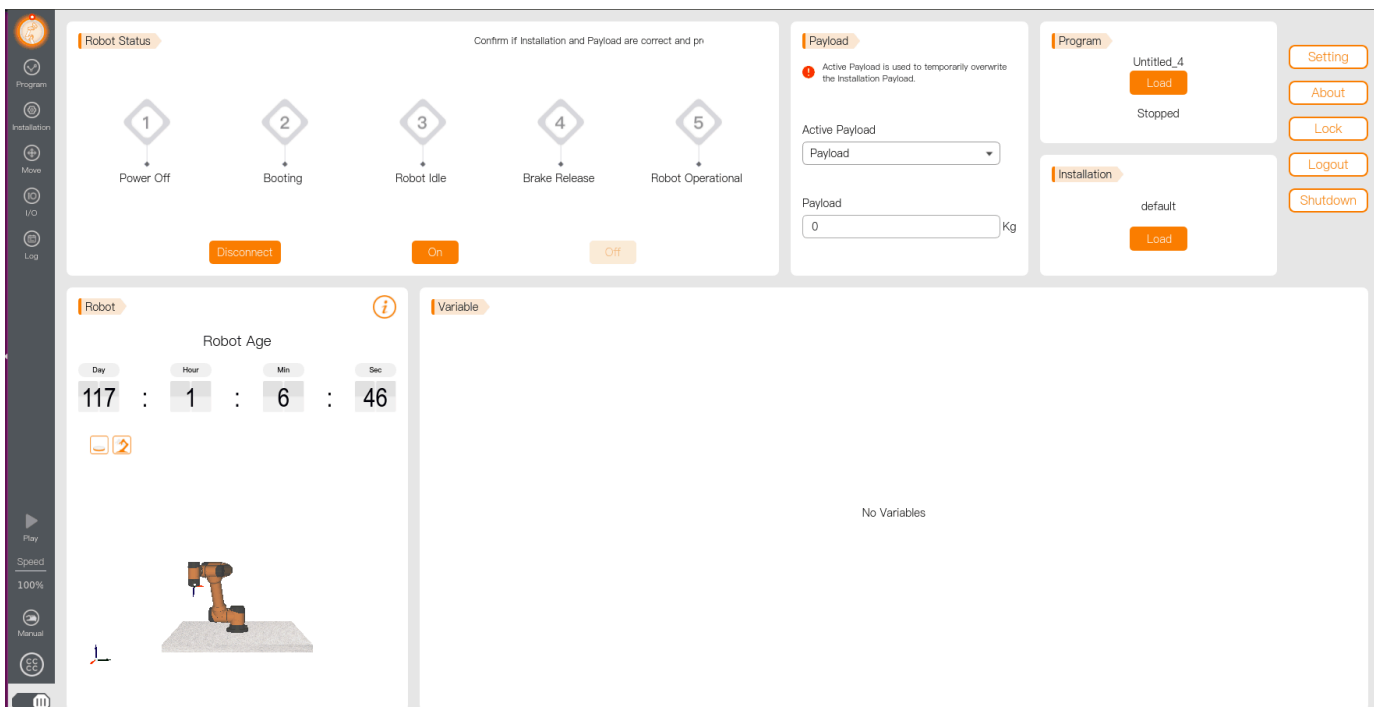
3. Select the example_startup.py file, and click the "Run" button.



4. Check the running results of the program.



5. The robot arm is powered on first, and all the indicator lights on the teach pendant come on. Then the robot arm is powered off, and all the indicator lights go out.



Examples of Running Java SDK

In Windows Environment

This section describes how to run Java SDK examples in the Windows environment, covering standard Maven projects and Android projects.

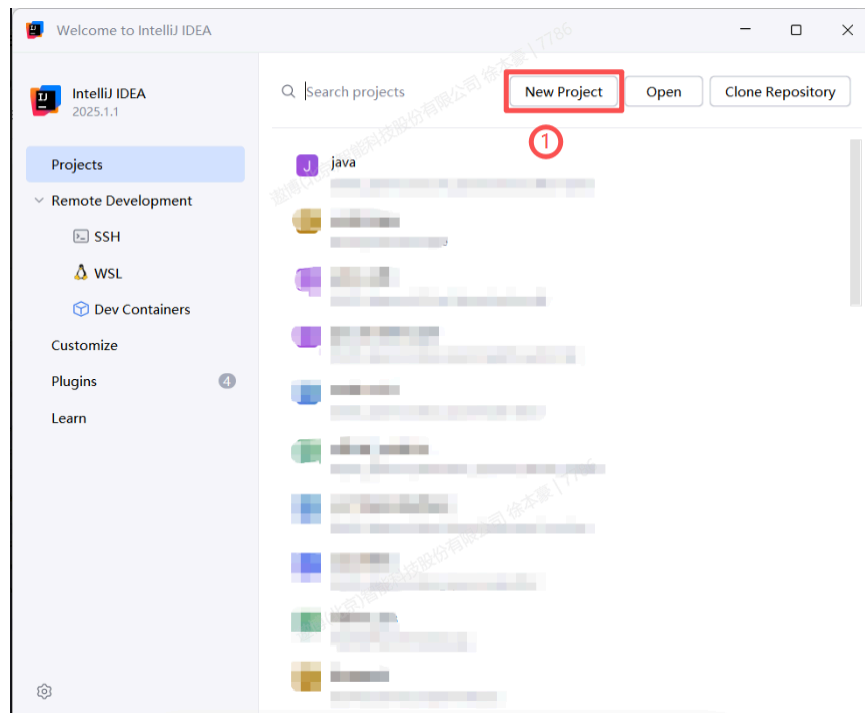
Note

- If the SDK client connects to a real robot arm, you must set the IP address in the code to the robot arm's IP address.
- If the project runs on the Aubo Sim virtual machine and the SDK client connects to the virtual machine, the IP address should be the virtual machine's IP address.

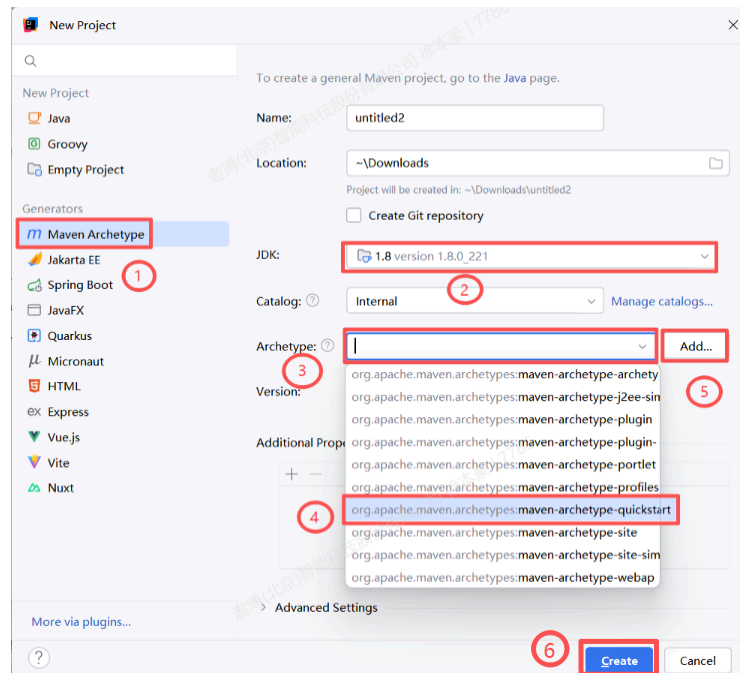
Java SDK Maven Project Example

This section describes how to use IntelliJ IDEA to power on the Aubo robot arm via the Java SDK in the Windows environment.

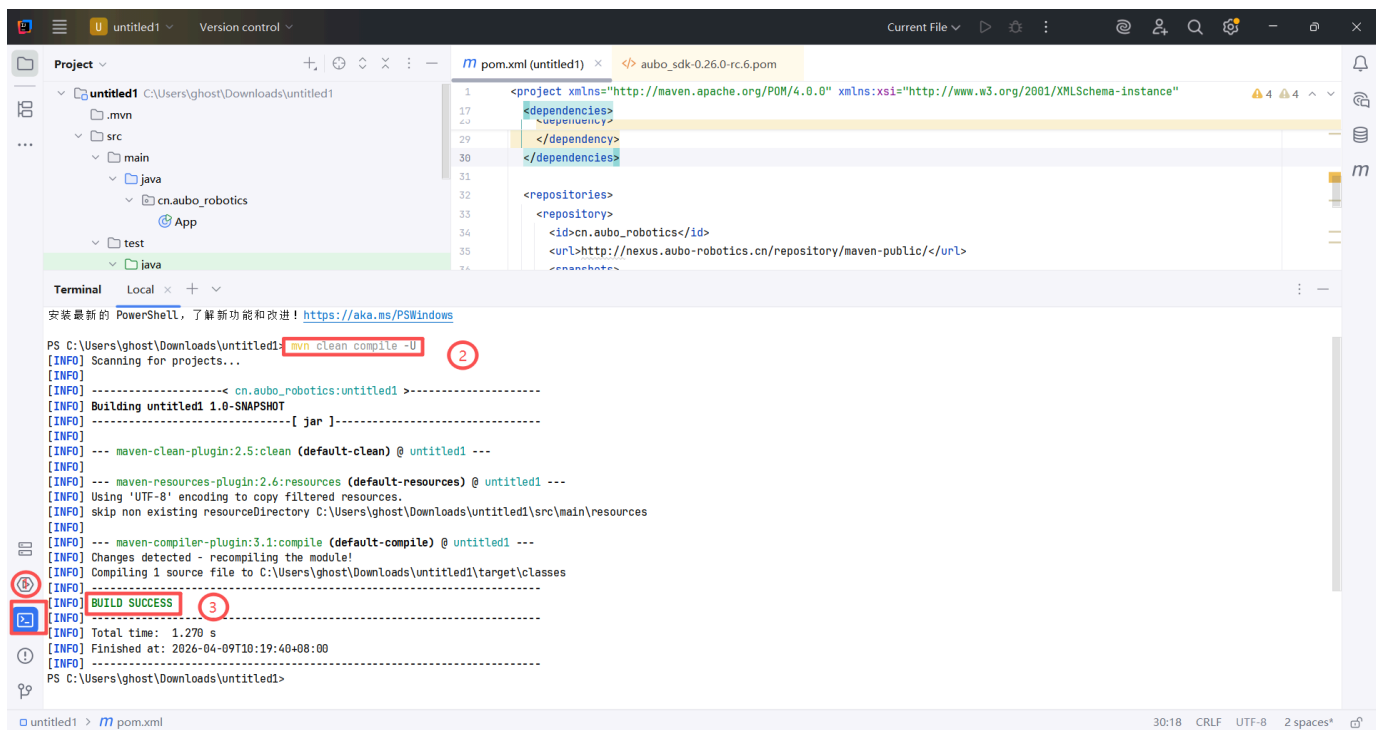
1. Open IntelliJ IDEA and click **New Project**.



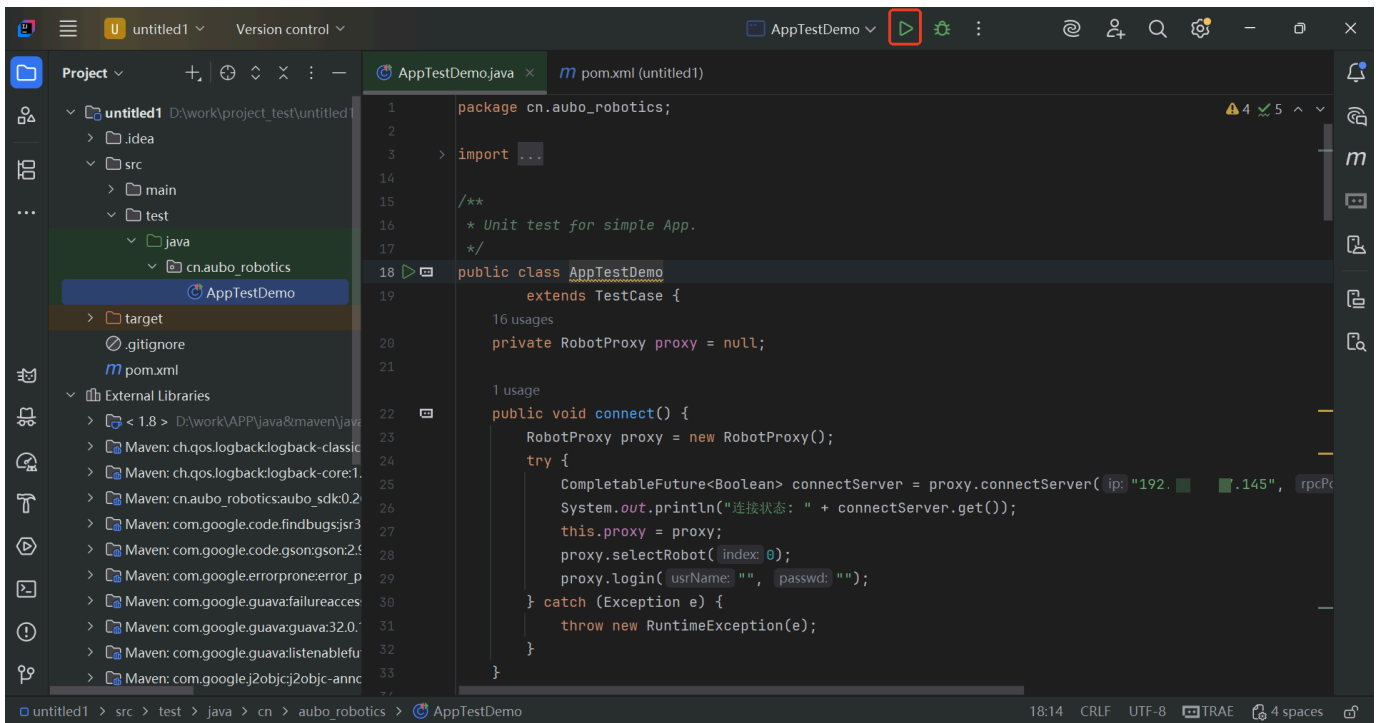
2. Create a Maven Archetype project and select the relevant options as shown in the steps below.



3. Refer to the [Installation Guide](#) to configure the Maven project repository, then run the Maven build command in the terminal: `mvn clean compile -U`.

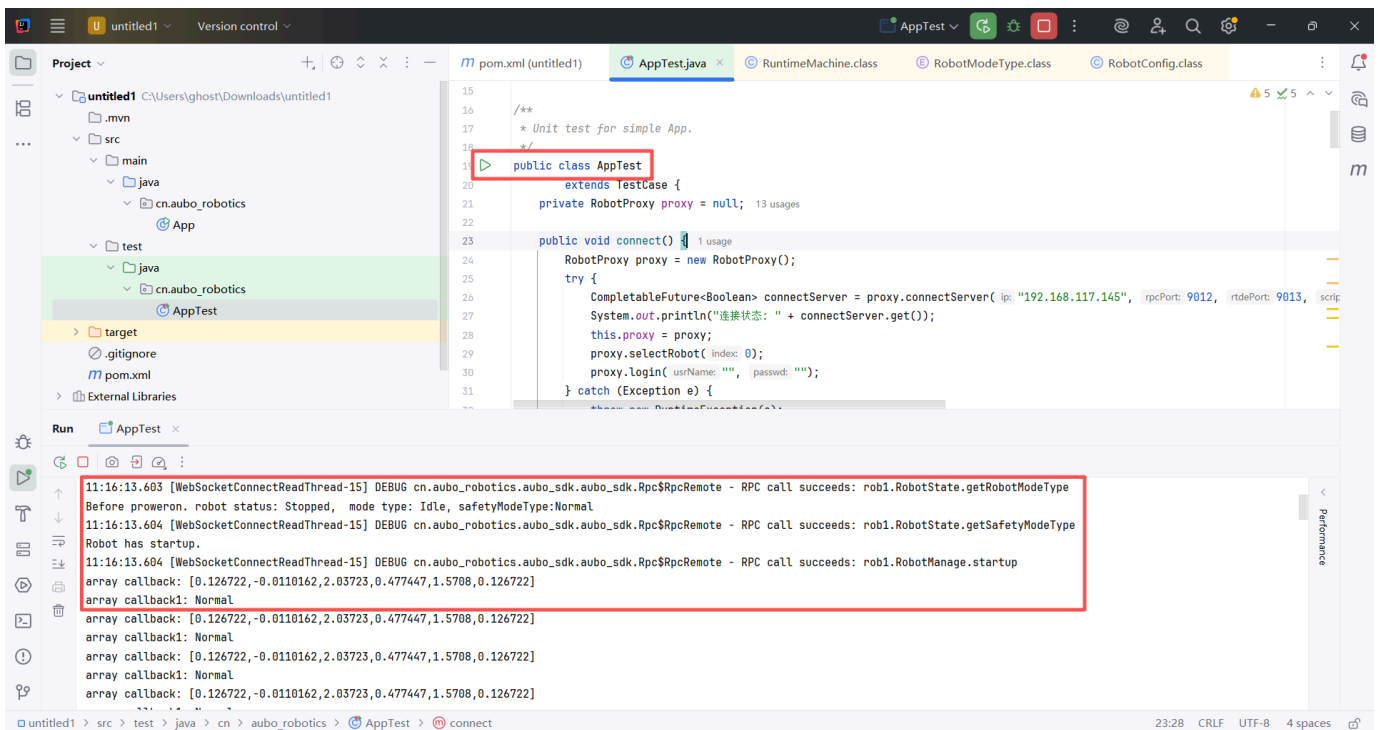


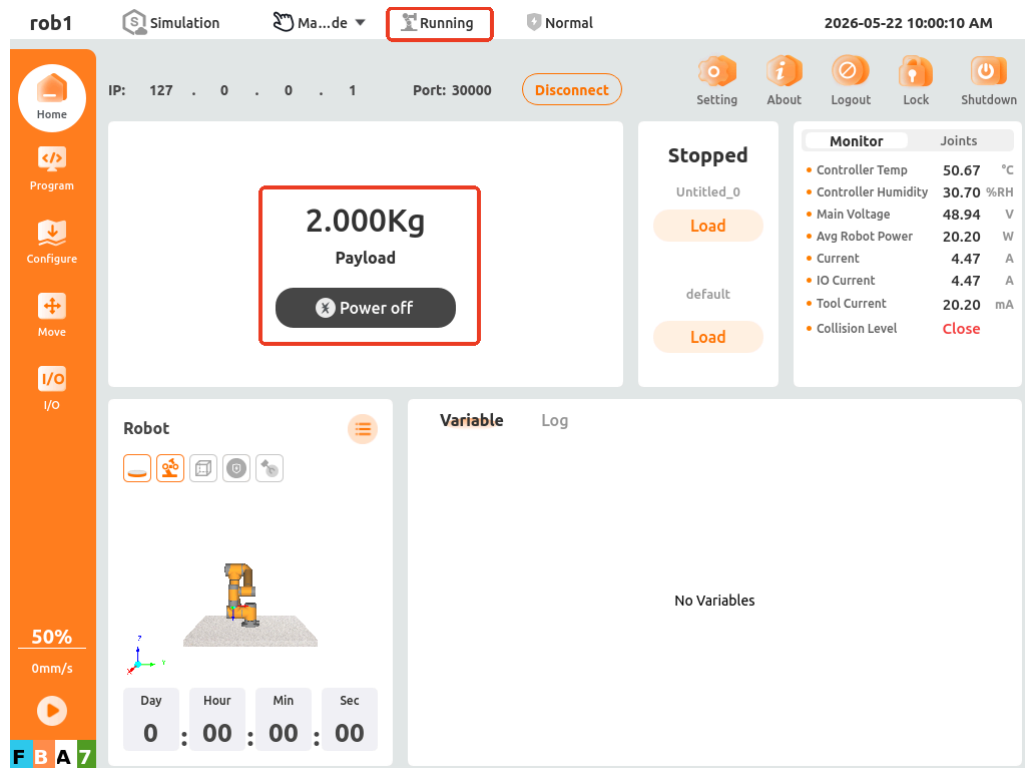
4. Create a test class `AppTest.java` in the test directory and run it.



► Click to view sample code

5. After running the `AppTest.java` file, check the runtime logs. Upon success, you can see the robot arm has been powered on in the virtual machine.

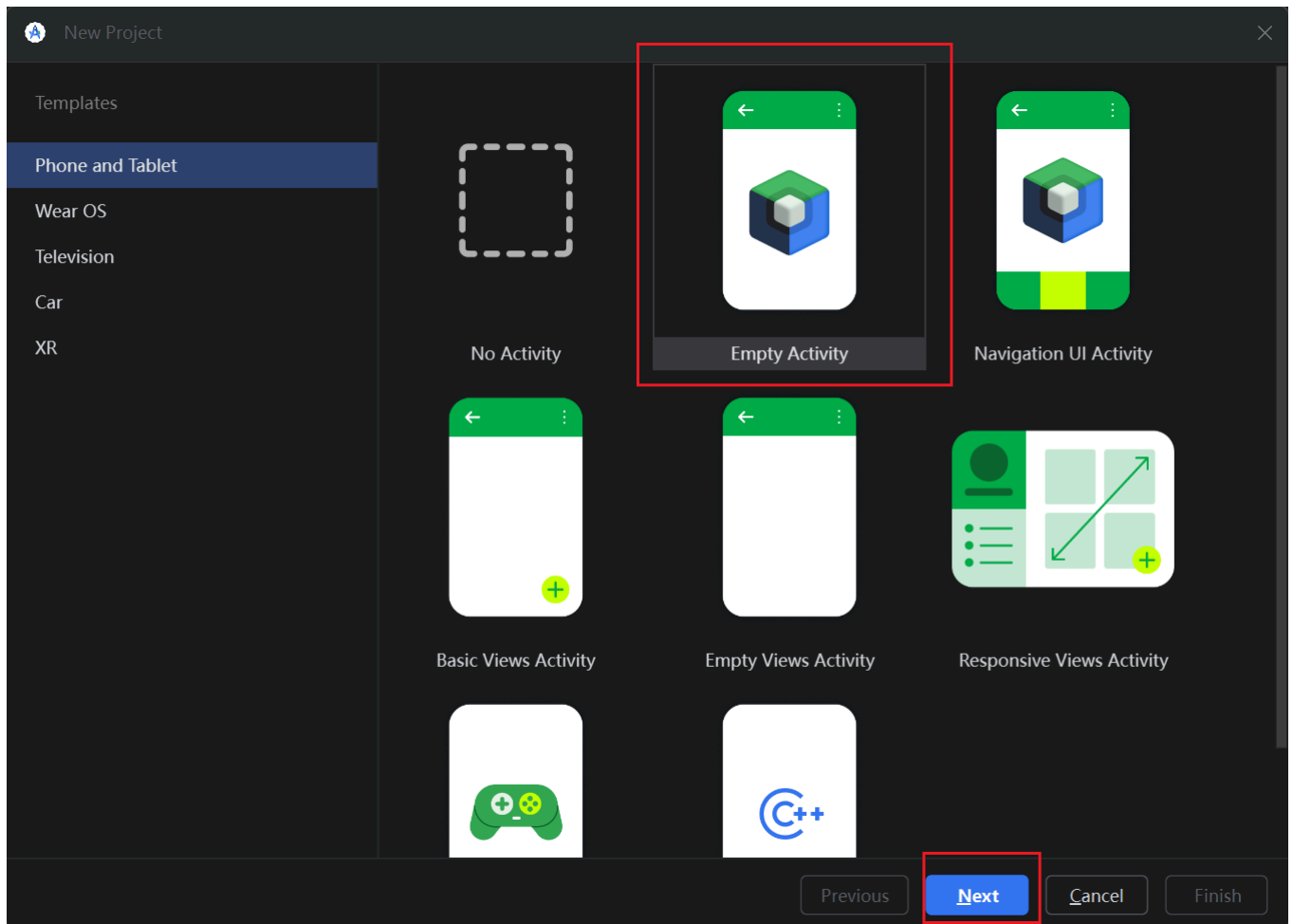




Java SDK Android Project Example

This section describes how to use Android Studio to power on the Aubo robot arm via the Java SDK in the Windows environment.

1. Open Android Studio and navigate through "New Project > Empty Activity > Next".



2. Configure the project name as shown in the image.

New Project

Empty Activity

Create a new empty activity with Jetpack Compose

Name

Package name

Save location

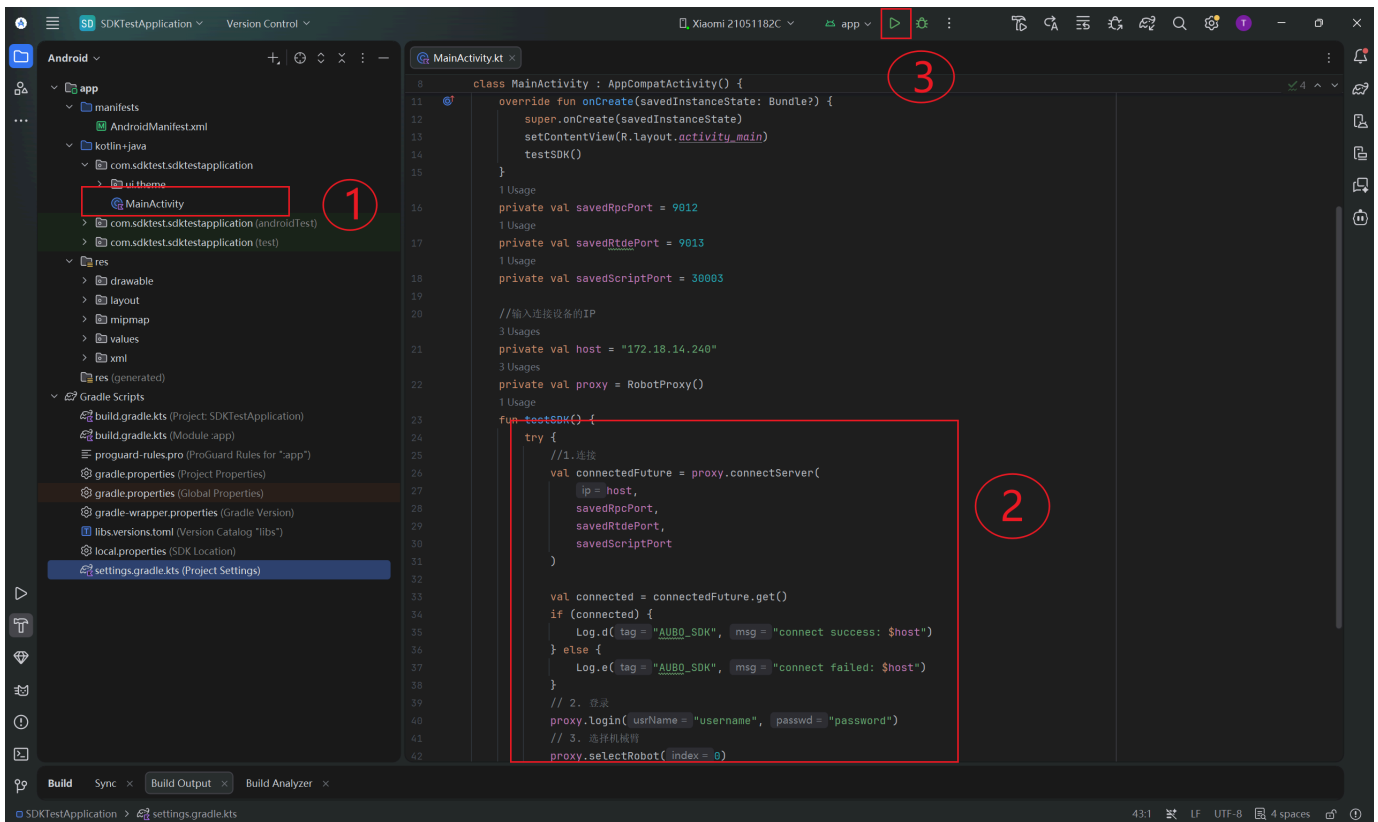
Minimum SDK

ⓘ Your app will run on approximately **99.2%** of devices.
[Help me choose](#)

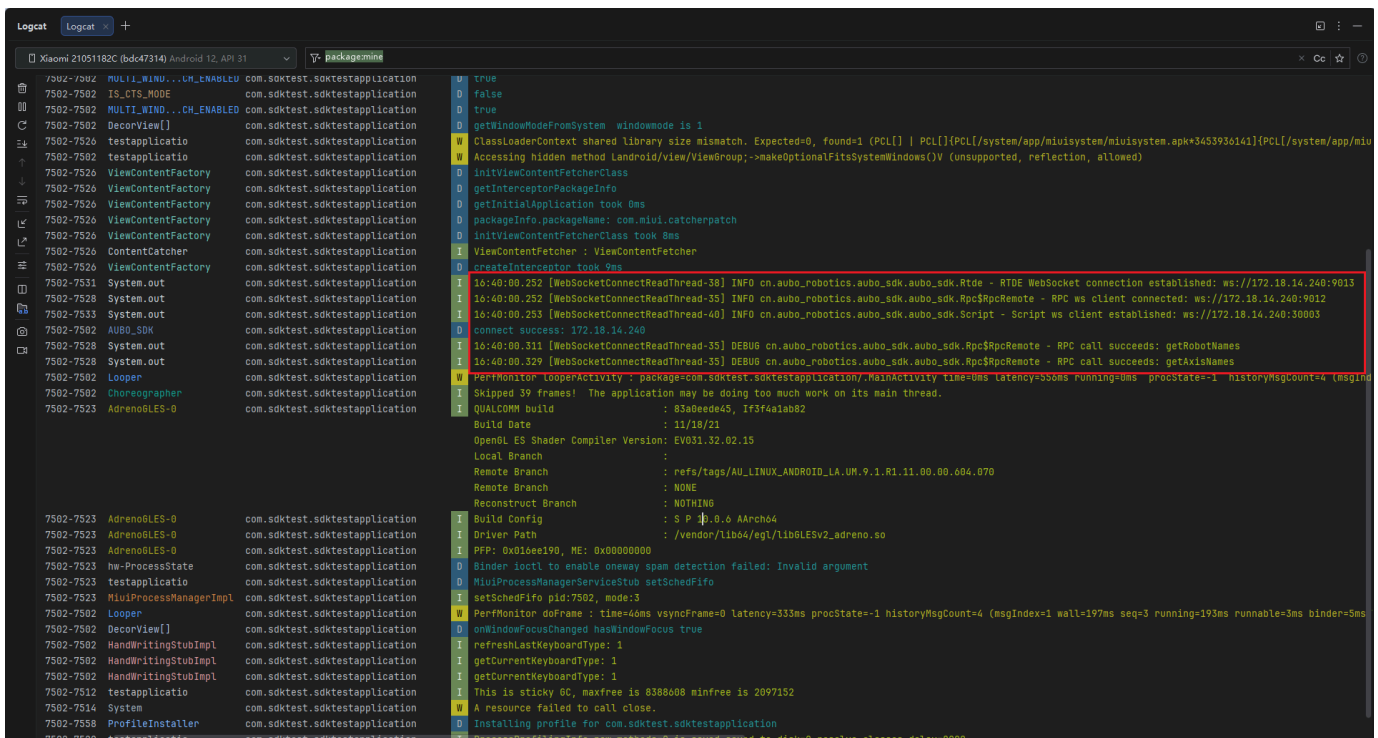
Build configuration language

3. Refer to the [Installation Guide](#) to configure the Android project repository.
4. Write the code in the `MainActivity` file, connect the Android device, and run the Android program.

► [Click to view sample code](#)



5. After running the `MainActivity` file, check the runtime logs in Logcat as shown. Upon success, you can see the robot arm has been powered on in the virtual machine.



rob1

Simulation

Ma...de

Running

Normal

2026-05-22 10:00:10 AM

Home

Program

Configure

Move

I/O

I/O

50%

0mm/s

Day

Hour

Min

Sec

0 : 00 : 00 : 00

IP: 127 . 0 . 0 . 1

Port: 30000

Disconnect

2.000Kg

Payload

Power off

Stopped

Untitled_0

Load

default

Load

Monitor

Controller Temp

50.67

°C

Controller Humidity

30.70

%RH

Main Voltage

48.94

V

Avg Robot Power

20.20

W

Current

4.47

A

IO Current

4.47

A

Tool Current

20.20

mA

Collision Level

Close

Robot

Variable

Log

No Variables

Troubleshooting

AUBO SDK provides **error codes**, **logs**, and **exception handling**, which are important tools for analyzing, judging, and resolving issues or errors of the SDK. They can help developers better debug and manage codes, improving the maintainability, reliability, and stability of codes. Logs record detailed information about the application in running process, including error codes and descriptions when an error occurs during running of the program. Exception handling can be used to diagnose and handle abnormal situations that arise during running of the program, such as program termination or crashes. Developers can choose appropriate methods to perform debugging based on specific scenarios and needs.

Error Codes

Error codes fall into three types: system error code, hardware abstraction layer error code, and runtime error code, defined in the header files `system_error.h`, `hal_error.h`, and `rtm_error.h` respectively. To view error codes and their corresponding descriptions, please refer to the document [Error Codes](#).

Log

AUBO SDK provides logs to record detailed information about events, errors, etc. during program running. Logs perform recording in text format, including timestamps, log levels, event descriptions, error information, etc. They also provide a history that can help developers or technical support personnel quickly locate issues when problems arise and provide detailed information about program running.

Log Level

Log level indicates the severity of the log, ranging from 0 to 5.

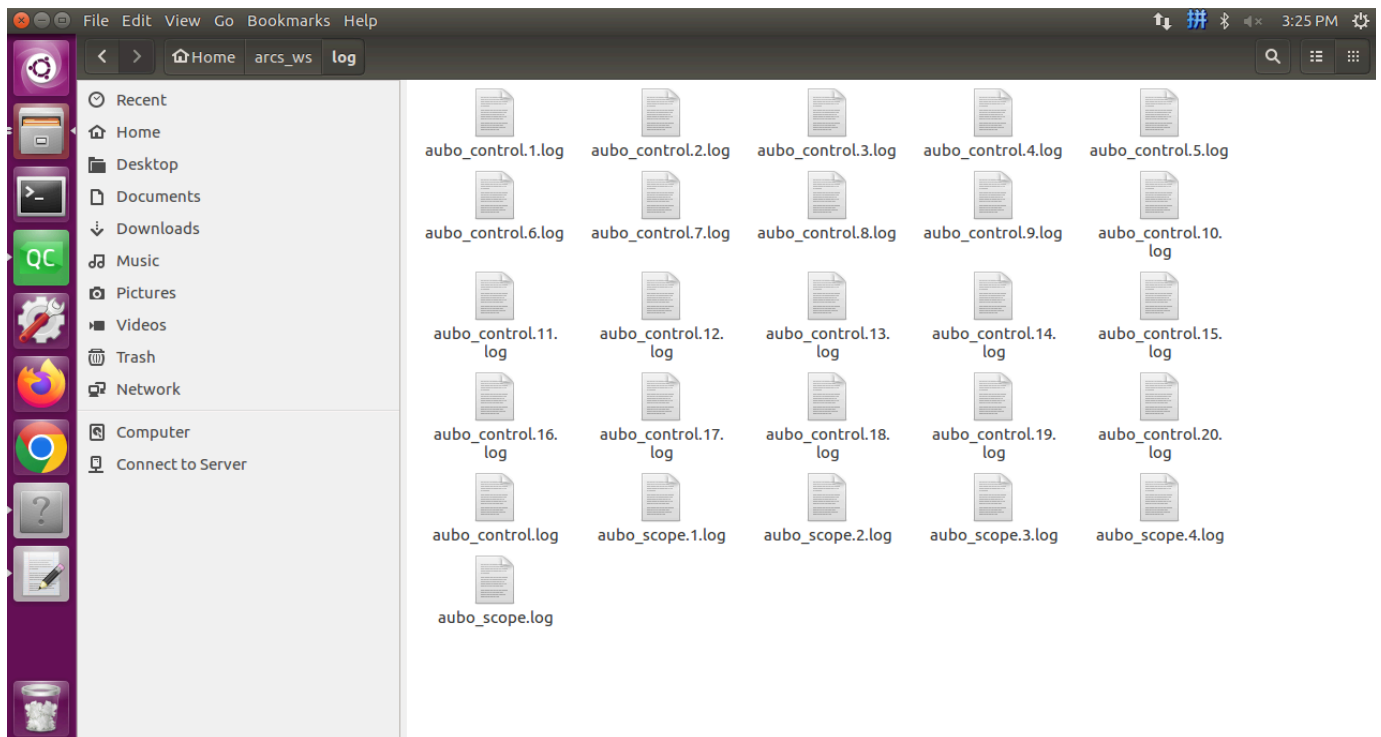
LogLevel	Value	Description
FATAL	0	Severe
ERROR	1	Error
WARNING	2	Warning

INFO	3	Notification
DEBUG	4	Debug
BACKTRACE	5	Backtrace

View Logs

Users can view logs in the following two ways:

1. **Log Files:** On the AUBO Sim virtual machine or teach pendant, users can find the aubo_control and aubo_scope log files. The aubo_control log file represents the server and the aubo_scope log file represents the teach pendant. The file path is **/root/arcs_ws/log**, as shown in the figure below:



2. **Terminal Real-time Log Printing:** Open the terminal on the AUBO Sim virtual machine or teach pendant, switch to the folder path **/root/arcs_ws/log** where the logs are located, and enter **tail -f aubo_control.log** or **tail -f aubo_scope.log**. While running the program, users can see the log printing information of the server or teach pendant in real time on the terminal, monitor the program's running status, reproduce error issues, or find the causes of errors.

Exception

AUBO SDK supports exception handling, and the **AuboException** class is defined in its header file **type_def.h**. Errors may occur during program running, leading to abnormal situations such as program termination or crashes. Developers can add codes to the program to catch exceptions to avoid issues like crashes or unresponsiveness, and to locate the position or cause of the error.

AuboException Class

In C++, the **AuboException** class includes three member functions: **type()**, **code()**, and **what()**.

type() is used to return the exception type, that is, an enumerated value of **error_type** that identifies the specific type of exception, such as `pars_error`, `invalid_request` and `method_not_found`.

Enumerated values of **error_type** are defined as follows:

```
enum error_type
{
    parse_error = -32700,      ///< Parsing error
    invalid_request = -32600,  ///< Invalid request
    method_not_found = -32601, ///< Method not found
    invalid_params = -32602,   ///< Invalid parameters
    internal_error = -32603,   ///< Internal error
    server_error,              ///< Server error
                               ///< This type is returned when -32099<= code_<=-3
    invalid                    ///< Invalid
                               ///< This type is returned when code_ is not within
};
```

code() is used to return the error code of the exception, that is, an enumerated value of **ExceptionCode** that represents the specific error situation of the exception.

Enumerated values of **ExceptionCode** are defined as follows:

```
enum ExceptionCode
{
    EC_DISCONNECTED = -1,      ///< Disconnected
    EC_NOT_LOGINED = -2,      ///< Not logged in
};
```

```

    EC_INVALID_SOCKET = -3,        ///< Invalid socket
    EC_REQUEST_BUSY = -4,         ///< Request busy
    EC_SEND_FAILED = -5,          ///< Sending failed
    EC_RECV_TIMEOUT = -6,         ///< Receiving timeout
    EC_RECV_ERROR = -7,           ///< Receiving error
    EC_PARSE_ERROR = -8,          ///< Parsing error
    EC_INVALID_REQUEST = -9,      ///< Invalid request
    EC_METHOD_NOT_FOUND = -10,    ///< Method not found
    EC_INVALID_PARAMS = -11,      ///< Invalid parameters
    EC_INTERNAL_ERROR = -12,      ///< Internal error
    EC_SERVER_ERROR = -13,        ///< Server error
    EC_INVALID = -14              ///< Invalid
};

```

what() is used to return a detailed error message of the exception object.

Example of C++ Exception Catch

When writing a program in C++, use **try/catch** to catch and handle potential exceptions. The codes are as follows:

```

try{
    ...// Code that causes an exception
}
catch(const arcs::aubo_sdk::AuboException& e) {
    // Print exception information
    std::cout << "An exception occurs: " << e.what() << std::endl;
    // Print the error code
    std::cout << "Exception code: " << e.code() << std::endl;
    // Print the exception type
    std::cout << "Exception type: " << e.type() << std::endl;
}

```

Example of Python Exception Catch

When developing with Python, use **try/except** for exception handling. The codes are as follows:

```
try:
    ...# Code that causes an exception
except pyaubo_sdk.AuboException as ex:
    print("An exception occurs: ", ex)
```


Introduction to ServoJ Function

`ServoJoint` refers to the method of precisely controlling the movement of the robot arm joints through servo motors with built-in control systems. The `ServoJoint` interface represents the servo position in joint space (linear).

1. ServoJ Interface

1.1 Parameters of Interface

```
// ServoJ Joint Space Servo

/**
 * Parameter definitions:
 * q: Target value, that is, the radians of the six joints.
 * a: Acceleration, not yet implemented.
 * v: Velocity, not yet implemented.
 * t: Expected movement time of the robot arm controlled by the command [s]
 * lookahead_time: Specifies the movement duration of the robot before starting to
 * gain: Proportional gain for tracking the target position [100, 200], used to co
 */

int servoJoint(const std::vector<double> &q, double a, double v, double t,
               double lookahead_time, double gain);
```

1.2 Parameter Adjustment Method

1. Set the `t` value ($t \geq 0.005$). The larger the `t` value, the slower the robot arm moves; the smaller the `t` value, the faster the robot arm moves.
2. Run the `ServoJoint` example and adjust the parameters of `std::this_thread::sleep_for()` to simulate the network delay.

Note: Currently, the available parameters for the `ServoJoint` interface are `q` and `t`. `q` represents the target trajectory point sent to the robot arm, and `t` represents the expected running time from the previous point to the current point. When this `t` value is set as the

interval time for continuously calling the `ServoJoint` interface, the effect is theoretically the best.

1.3 Usage Conditions

To ensure real-time performance, please use the `Linux` operating system and use a **wired network** for connection.

1.4 Automatic Exit Mechanism

The `ServoJ` function has a mechanism to automatically exit the `servo` mode. After exiting the `servo` mode, this function cannot control the movement of the robot arm anymore. To execute such control again, this function must re-enter the `servo` mode.

The automatic exit mechanism includes:

- When any of interfaces such as "stopJoint/stopLine/abort" is called through the "SDK", the `servo` mode will automatically exit;
- Once the drag mode is enabled, the `servo` mode will automatically exit;
- If no `ServoJoint` command is received for a long time (5 times the expected movement time, `t`), the `servo` mode will automatically exit;

After the `servo` mode exits, to continue using the `ServoJ` function, `setServoMode(true)` must be called through the "SDK" to re-enter the `servo` mode;

1.3 Typical Application Scenarios

- This function is suitable for applications that require precise and stable motion control, such as assembly, cell separation and electronic component assembly;
- This function is suitable for applications that require high torque and precision, such as machining and automotive manufacturing;
- This function is suitable for applications that need to adapt to various workpieces in fixed and non-fixed shapes, such as mold manufacturing and material handling;

1.4 Problem Handling

Problem 1: Robot Arm Shaking

- Check if the network communication is normal, and use a **wired network** for connection;
- To improve the real-time performance, use a **Linux** operating system with a **real-time kernel**;
- Check if the time intervals of the issued trajectory points are uniform and continuous, and appropriately increase the **t** value.

Problem 2: Robot Arm Stops Moving After a Period of Time

The following log may appear:

```
11/09/23 17:37:57.101697 [934] INFO rob1::alg - aral::initMovementAttribute - trajectory_setup.cpp:732 max_vel: 3.11049, 3.11049, 3.11049, 3.11049, 3.11049, 3.11049 max_acc: 31.4, 31.4, 31.4, 31.4, 31.4 max_jerk: 251.2, 251.2, 251.2, 251.2, 251.2, 251.2
11/09/23 17:37:57.101761 [934] INFO rob1::alg - aral::updateMotion - trajectory_tracking.cpp:391 当前运动更新成关节空间位置跟踪模式, 目标关节角: -0.994666, 0.0860886, -0.773704, 0.937372, -1.54666, -1.8872
11/09/23 17:37:57.116099 [934] INFO rob1::alg - aral::computeTimeStamps - velocity_profile_otg.cpp:131 addLineVel 计算速度轮廓成功, duration: 0.005, maxV: 3.11049, 3.11049, 3.11049, 3.11049, 3.11049, 3.11049, maxA: 31.4, 31.4, 31.4, 31.4, 31.4, 31.4, maxJ: 251.2, 251.2, 251.2, 251.2, 251.2, 251.2
11/09/23 17:37:57.116120 [934] WARNING rob1::alg - aral::computeTimeStamps - velocity_profile_otg.cpp:137 addLineVel计算速度轮廓超时 with TypeV OTG, 耗时(us): 3135.71 超时时间(us): 2000
11/09/23 17:37:57.116447 [934] INFO rob1::alg - aral::add - trajectory_queue.cpp:449 轨迹队列中增加的路径id -1:-1 该节点的初始速度: 0 0 0 0 0 0 目标速度: 0 0 0 0 0 0 起始加速度: 0 0 0 0 0 0 目标加速度: 0 0 0 0 0 0 maxV: 3.11049 3.11049 3.11049 3.11049 3.11049 3.11049 maxA: 31.4 31.4 31.4 31.4 31.4 31.4 maxJ: 251.2 251.2 251.2 251.2 251.2 251.2
11/09/23 17:37:57.116940 [934] INFO rob1::alg - aral::step - trajectory_queue.cpp:289 Finish one trajectory unit, path id: -1
11/09/23 17:37:57.117079 [934] INFO rob1::alg - aral::updatePlanStartPoint - trajectory_setup.cpp:346 设置规划器下一段路径起始关节角为: -0.994666, 0.0860886, -0.773704, 0.937372, -1.54666, -1.8872 关节速度为: -4.80493e-20, 6.66708e-24, 5.84346e-26, 1.10556e-19, 5.2145e-24, 1.57369e-23 关节加速度为: 4.99682e-17, 0, 0, 3.40033e-17, 0, 0
11/09/23 17:37:57.126277 [934] INFO rob1 - Exiting servo mode without calling the servoJoint interface for 0.026 seconds
11/09/23 17:37:57.126440 [934] INFO rob1::alg - aral::updatePlanStartPoint - trajectory_setup.cpp:346 设置规划器下一段路径起始关节角为: -0.994666, 0.0860886, -0.773704, 0.937372, -1.54666, -1.8872 关节速度为: 0, 0, 0, 0, 0, 0 关节加速度为: 0, 0, 0, 0, 0, 0
11/09/23 17:37:57.126590 [953] DEBUG rob1 - >>> servoJoint [-0.9946664989950161, 0.08608861804099621, -0.7737044900137799, 0.9373719010486864, -1.5466607023079022, -1.8872040933408862]
11/09/23 17:37:57.126660 [953] DEBUG aubo_comm - jsonrpc_asio.h:70 [conn_id:2] req: {"id":"visZSIYz-Wq4P-hUqP","jsonrpc":"2.0","method":"rob1.MotionControl.servoJoint","params":{"a":0.2,"gain":200.0,"lookahead_time":0.1,"q":[-0.9946664989950161,0.08608861804099621,-0.7737044900137799,0.9373719010486864,-1.5466607023079022,-1.8872040933408862],"t":0.005,"v":0.2}} rep: {"id":"visZSIYz-Wq4P-hUqP","jsonrpc":"2.0","result":0}
11/09/23 17:37:57.127091 [934] INFO rob1::alg - aral::scaleVelAndAcc - trajectory_speed_adjust.cpp:14 用户设置的速度比例: 1 加速度比例: 1 缓冲点个数: 0
11/09/23 17:37:57.127169 [934] DEBUG rob1::alg - trajectory_plan.hpp:248 tpScaleVelocityAndAcceleration 1
11/09/23 17:37:57.127209 [934] INFO rob1::alg - aral::setEndEffectorFrameInFlange - robot_parameter_setup.cpp:77 设置机械臂末端执行点, 相对于法兰坐标系的位姿为: [0, 0, 0, 0, 0, 0]
11/09/23 17:37:57.127250 [934] DEBUG rob1::alg - trajectory_plan.hpp:261 Init last target -0.994666, 0.0860886, -0.773704, 0.937372, -1.54666, -1.8872
```

- Due to the uneven call time of the **servoJoint** interface executed by the customer's own program, the interval time of a certain call is too long (exceeding t^*5s). As a result, the automatic exit mechanism is triggered. Appropriately increase the **t** value and run the program again;

2. Test

Considerations: CPU processing capability, memory size, bandwidth, latency, and other factors of the system.

System Platform Test:

- **windows** platform (without live patching)
- **linux ubuntu16.04** platform (without live patching)
- **linux ubuntu16.04** platform (with live patching)

Performance Test:

Use benchmarking methods: Cycle time comparison, maximum payload test, transmission efficiency test, and real-time performance test.

Note: When running the `servoJ` example, wait till the program execution is completed. Do not interrupt it midway. Otherwise, after the `servo` mode is enabled, the `servoJ` example will not be closed.

2.1 Preparation before Test

2.1.1 Network and Firewall Settings

ARCS Software Network Configuration:

https://docs.aubo-robotics.cn/application_notes/26-arcs-networkconfig/

A wired static network configuration is used on the `linux ubuntu16.04` platform (with live patching).

2.1.2 Check `linux ubuntu16.04` Platform for Live Patching

Steps:

1. Open the terminal.
2. Run the command:

```
bash
uname -r # This command displays the currently running kernel version number
```

3. If the kernel version number ends with `-rt` plus more characters (for example: `4.14.195-rt94`), it indicates that the live patching of the kernel is installed; If there is no `-rt`, there is no live patching on the current machine.
-

2.2. Test Using `ServoJ` Example in `SDK`

2.2.1 Test Process

Note:

When performing test on various system platforms, bind the routine to be tested to an isolated `CPU`, call the `ServoJ` interface, adjust parameters, and analyze the collected data through plotting and observe the robot arm's shaking phenomenon by touching the robot arm to determine the test results.

Example Code of `example_servoj2`

Execution Result:

```

设置调度策略 = [1]
[Info] [connector_asio.h:108] Connect to 172.16.3.191:30004
[Info] [rpc.cpp:85] SDK INTERFACE VERSION: 0.20.0
[Info] [rpc.cpp:86] SERVER INTERFACE VERSION: 0.20.0
[Info] [connector_asio.h:152] RPC Client login
servoJoint finish!
Servoj end
按 <RETURN> 来关闭窗口...
  
```

2.2.1.1 windows Platform (without Live Patching)

1. Configure the network and test the connectivity of the network.
2. Execute the test example of `example_servoj2`.
3. Record the results.

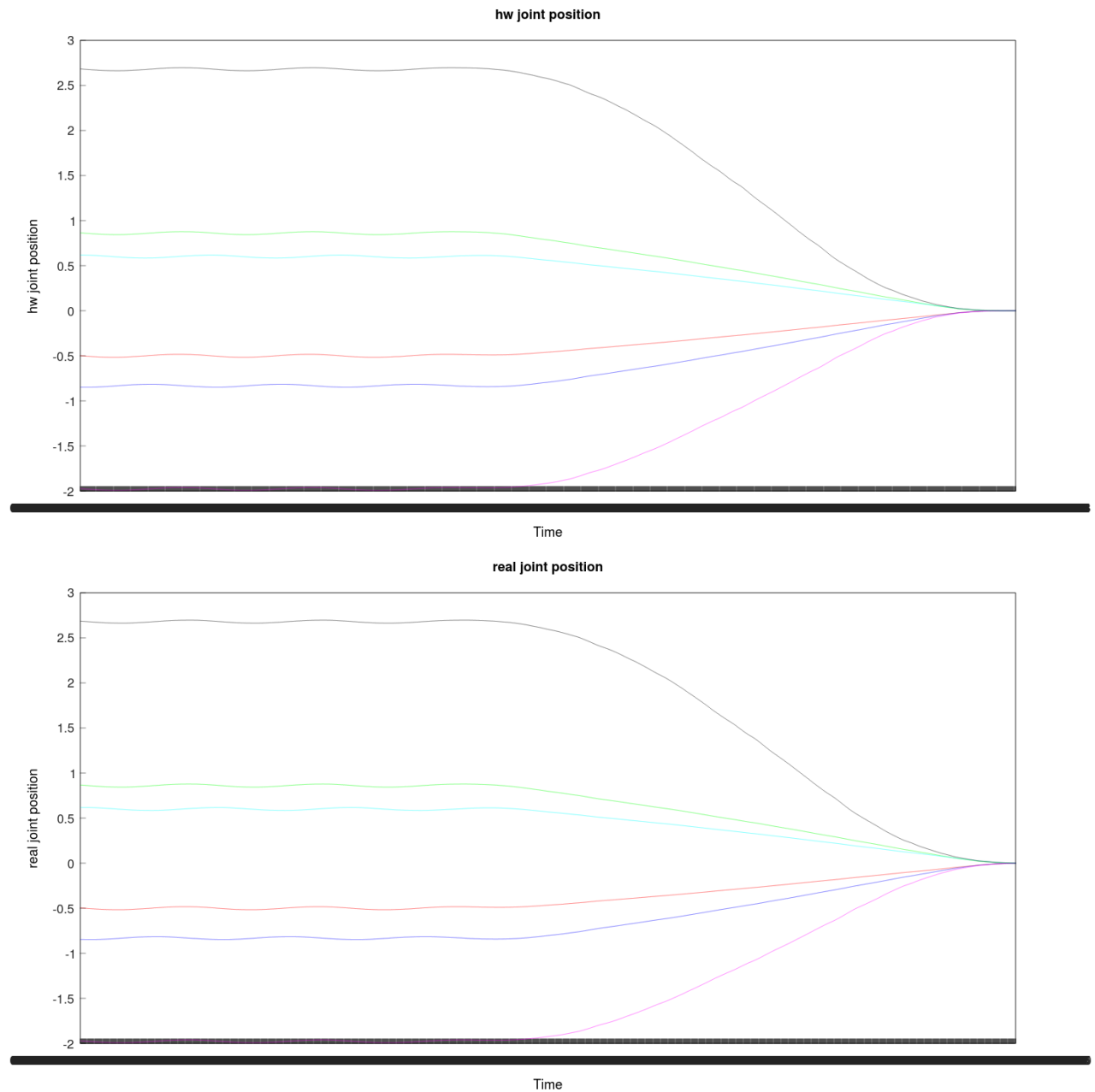
Use a real-time system, set up real-time processes, bind the real-time processes to an isolated `CPU`, and call `servoj with a period of 5ms``.

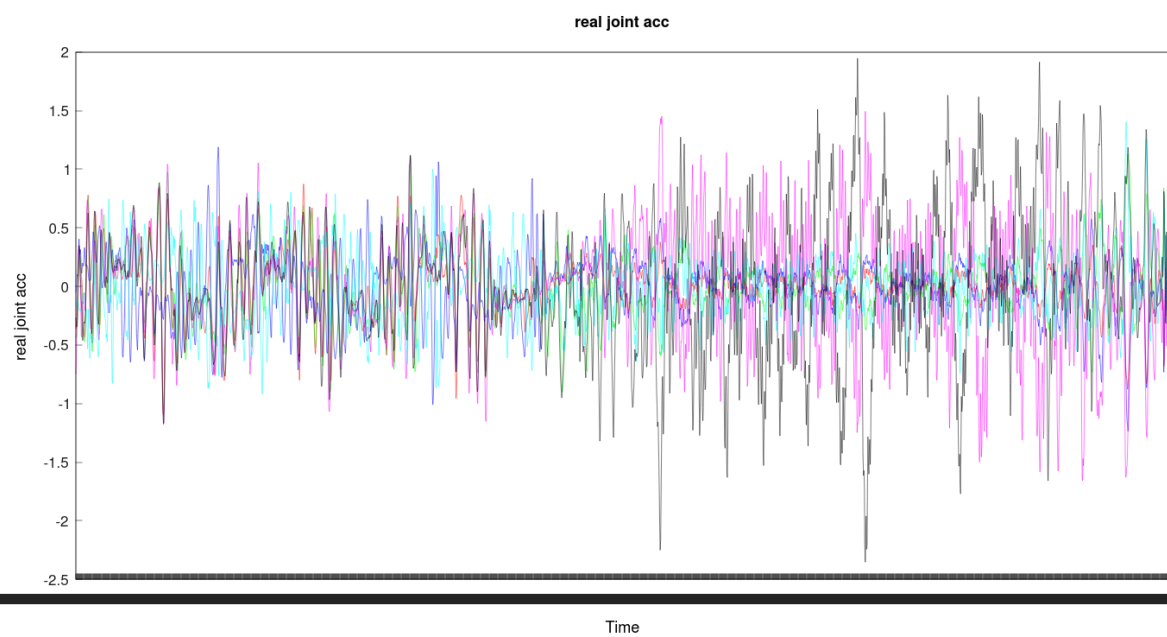
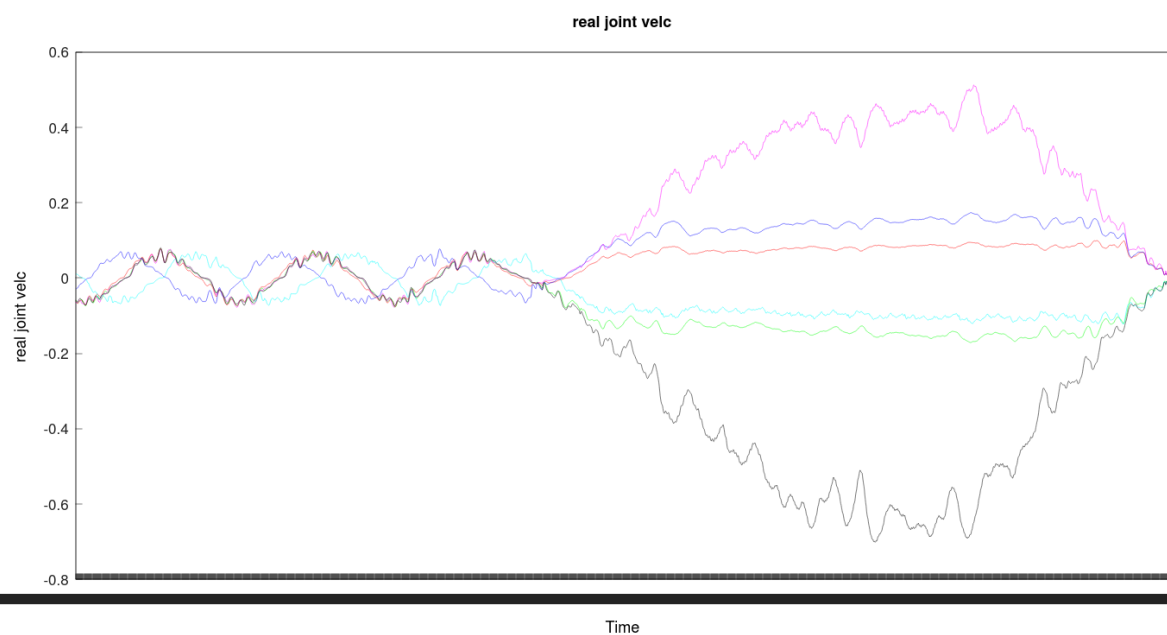
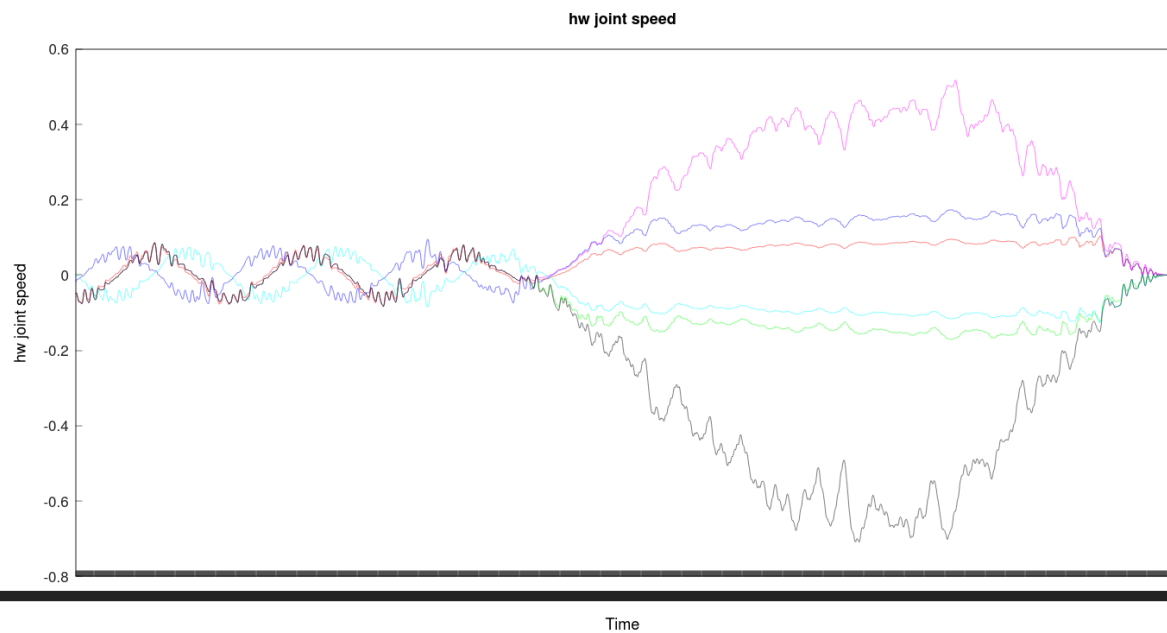
- Qualitative analysis

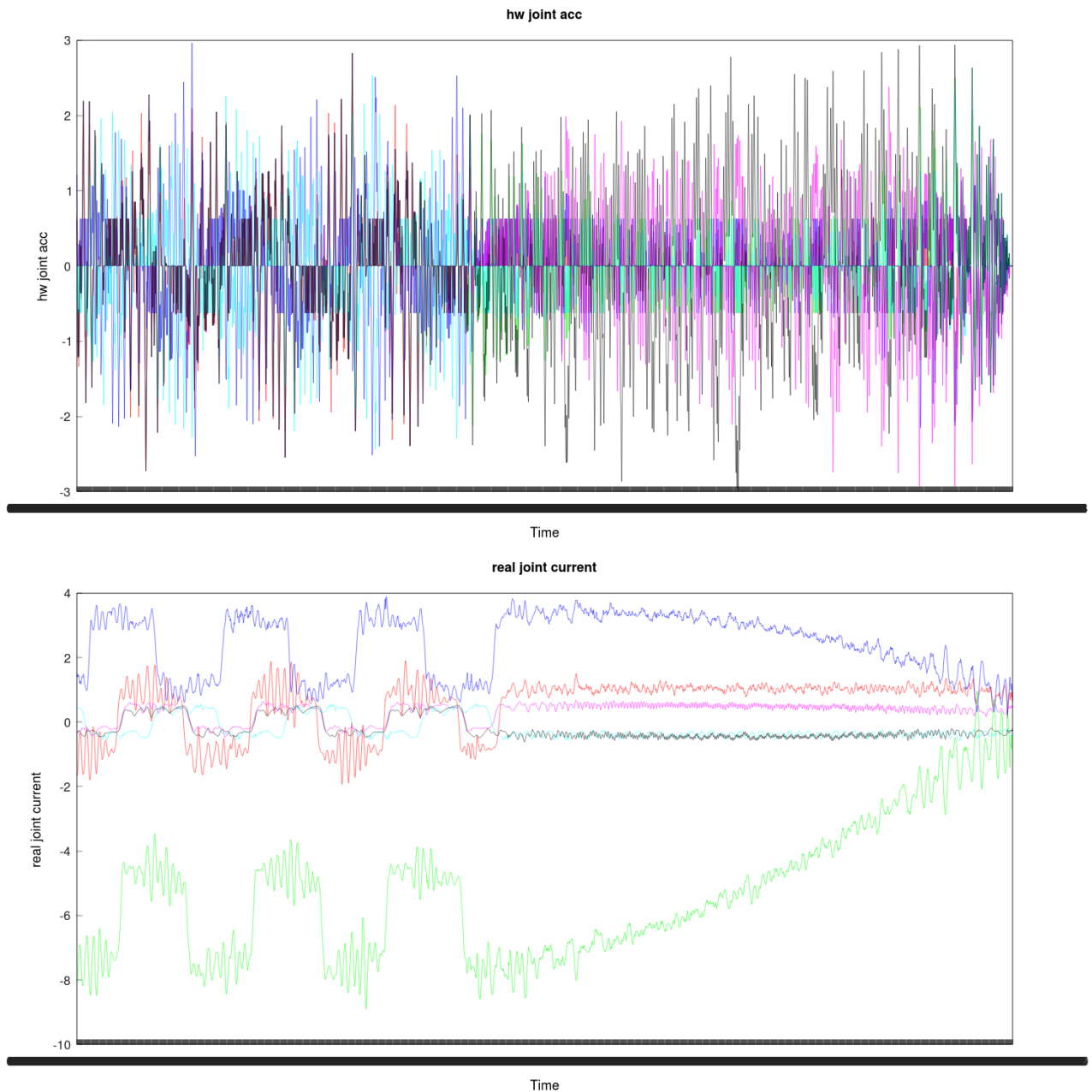
Touch the robot arm by hand to feel it, which shakes obviously, seen with naked eyes.

- Quantitative analysis

From the figure below, it can be analyzed that the robot arm has significant vibration.







2.2.1.2 linux ubuntu16.04 Platform (without Live Patching)

1. Configure the network and test the connectivity of the network.
2. Execute the test example of `example_servoj2`.
3. Record the results.

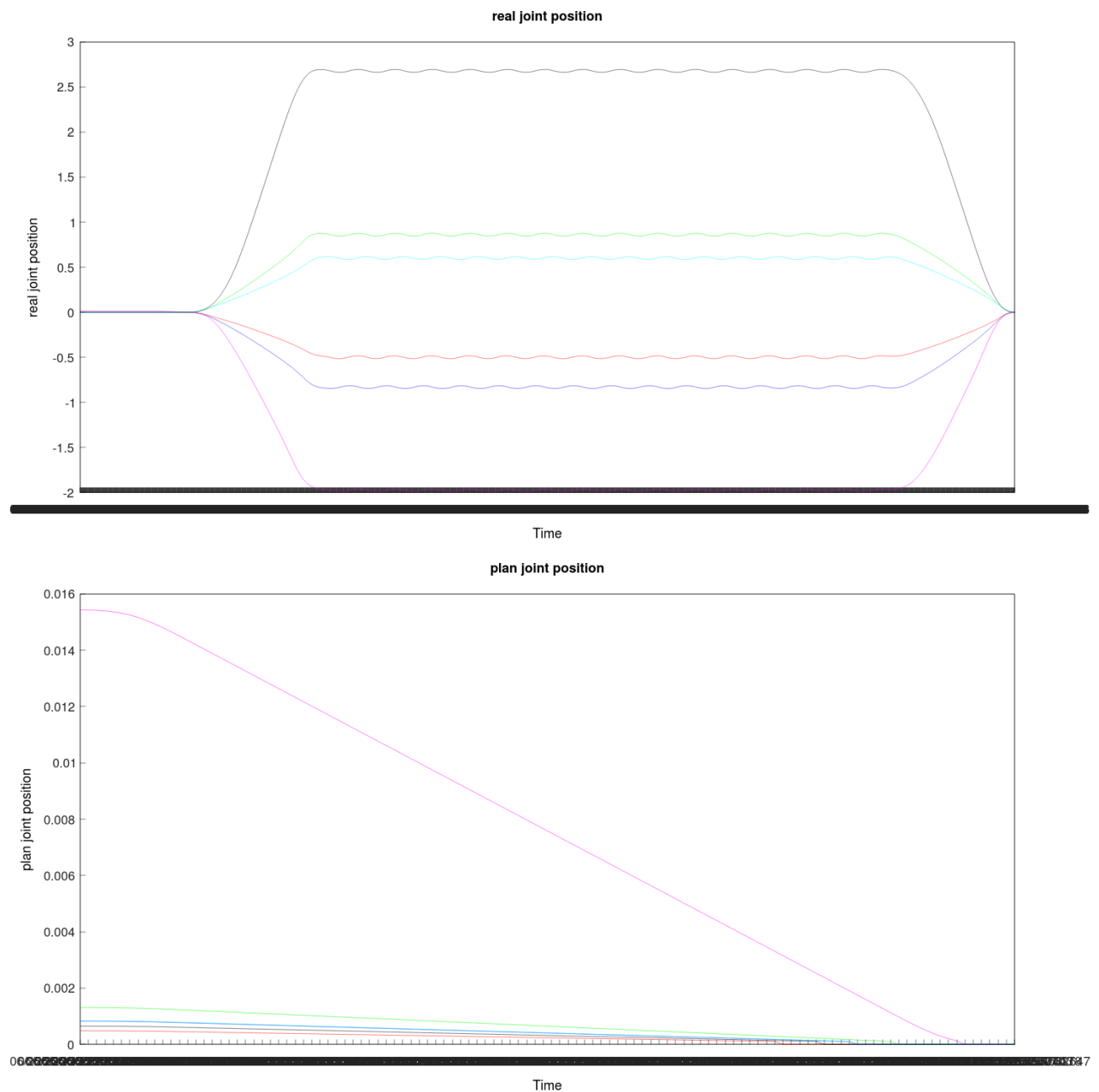
Use a real-time system, set up real-time processes, bind the real-time processes to an isolated `CPU`, and call `servoj` with a period of `5ms`.

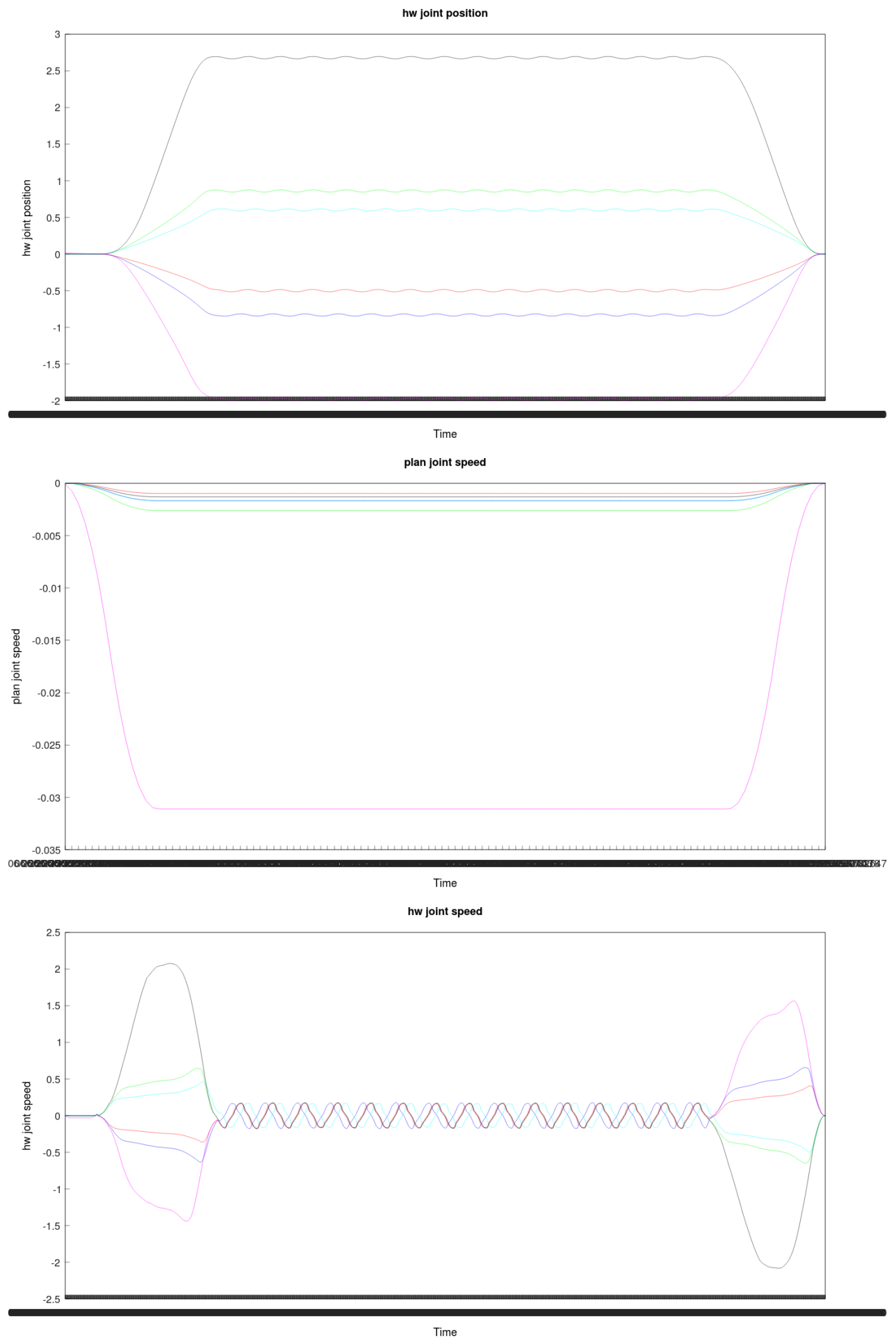
- Qualitative analysis

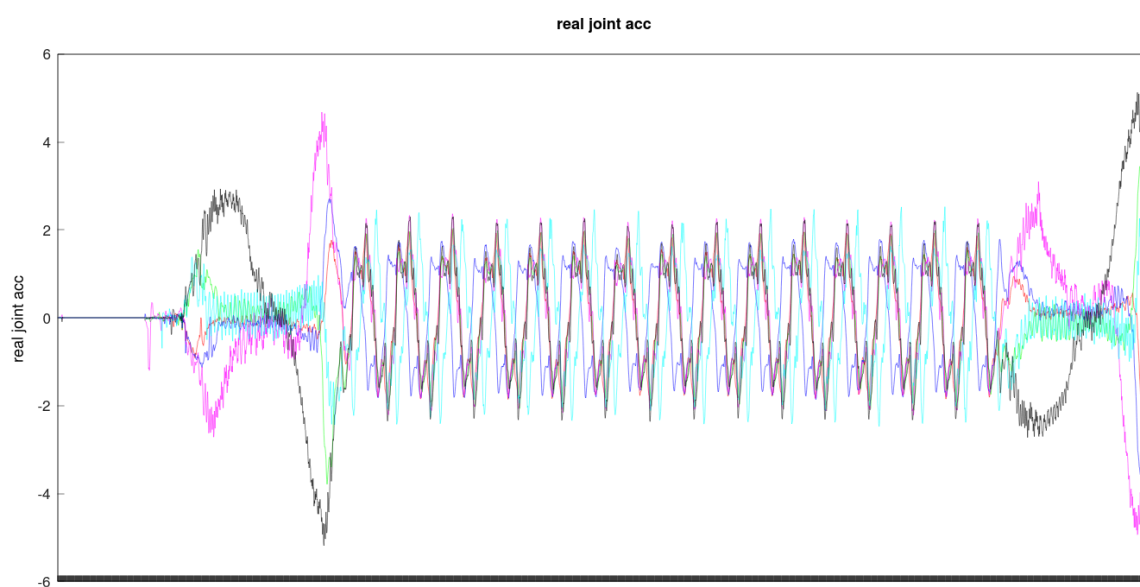
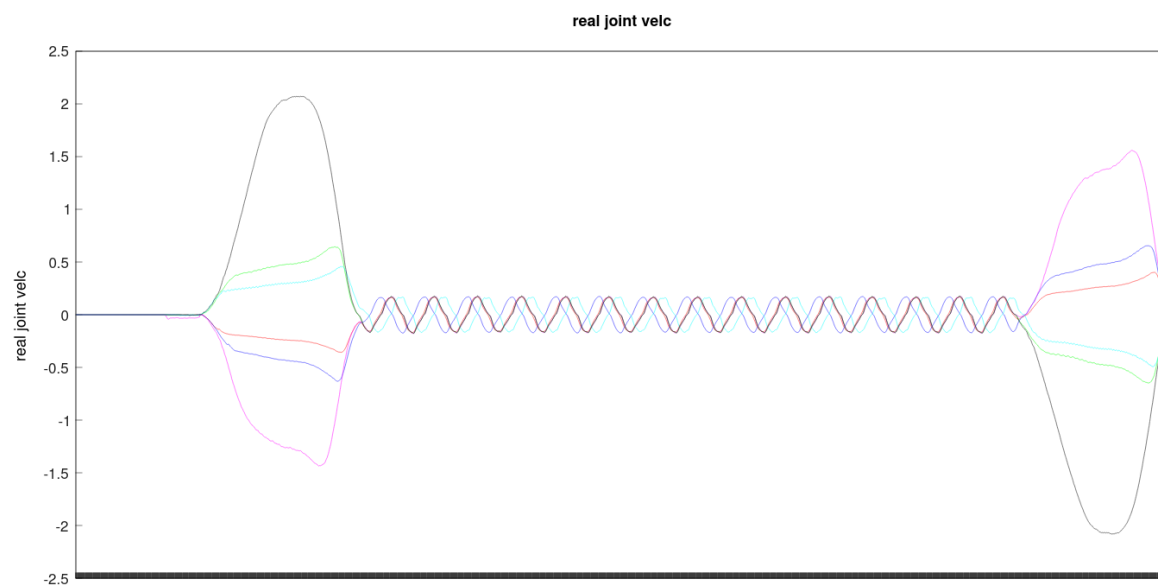
Touch the robot arm by hand to feel it. There is slight shaking during the movement, which is more noticeable compared to the [linux ubuntu16.04](#) platform (with live patching). However, the shaking is not obvious during the ascent and descent.

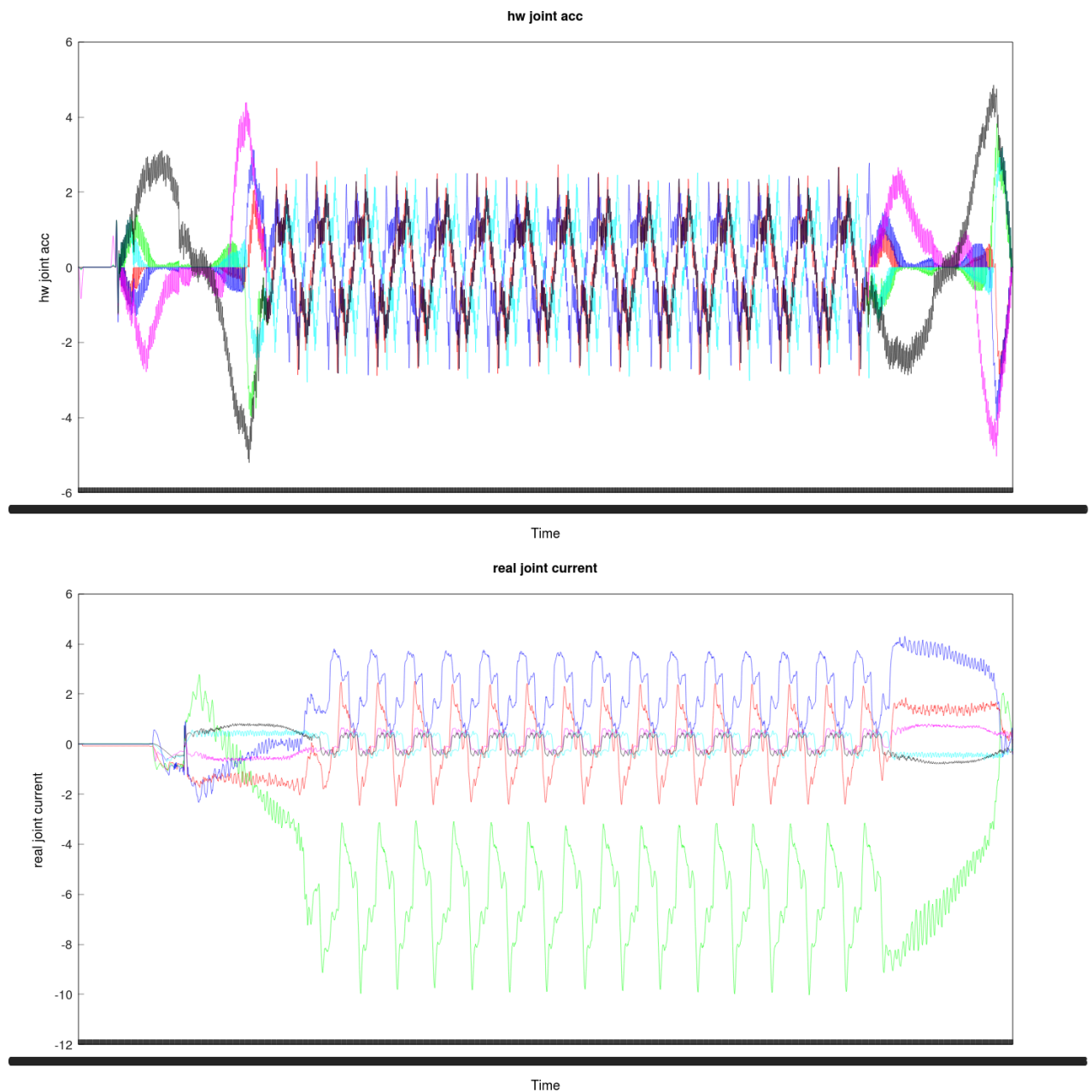
- Quantitative analysis

From the figure below, it can be analyzed that the robot arm basically does not vibrate.









2.2.1.3 linux ubuntu16.04 Platform (with Live Patching)

Steps:

1. Configure the network and test the connectivity of the network.
2. Execute the test example of `example_servoj2`.
3. Record the results.

Use a real-time system, set up real-time processes, bind the real-time processes to an isolated `CPU`, and call `servoj` with a period of `5ms`.

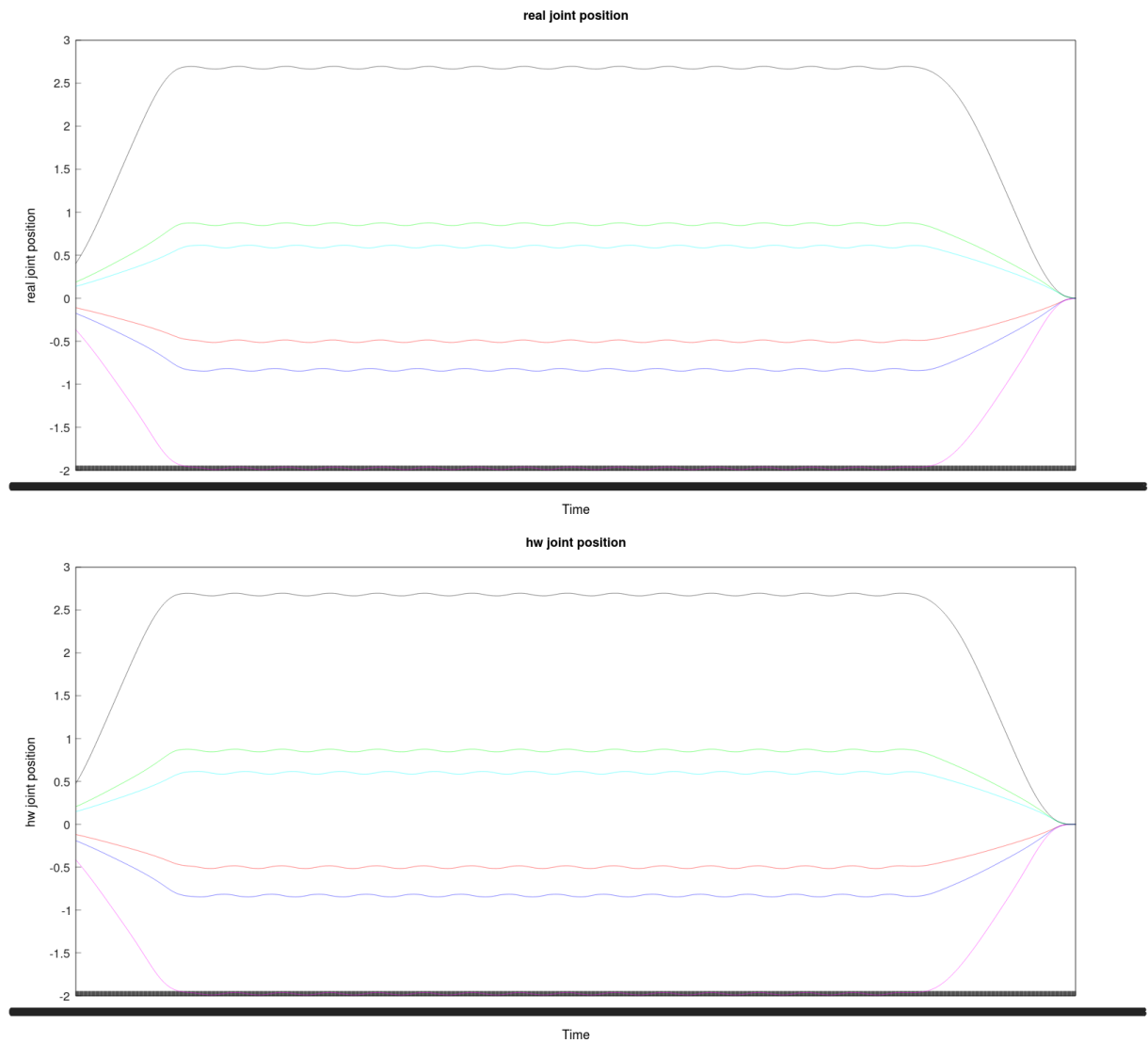
- **Qualitative Analysis**

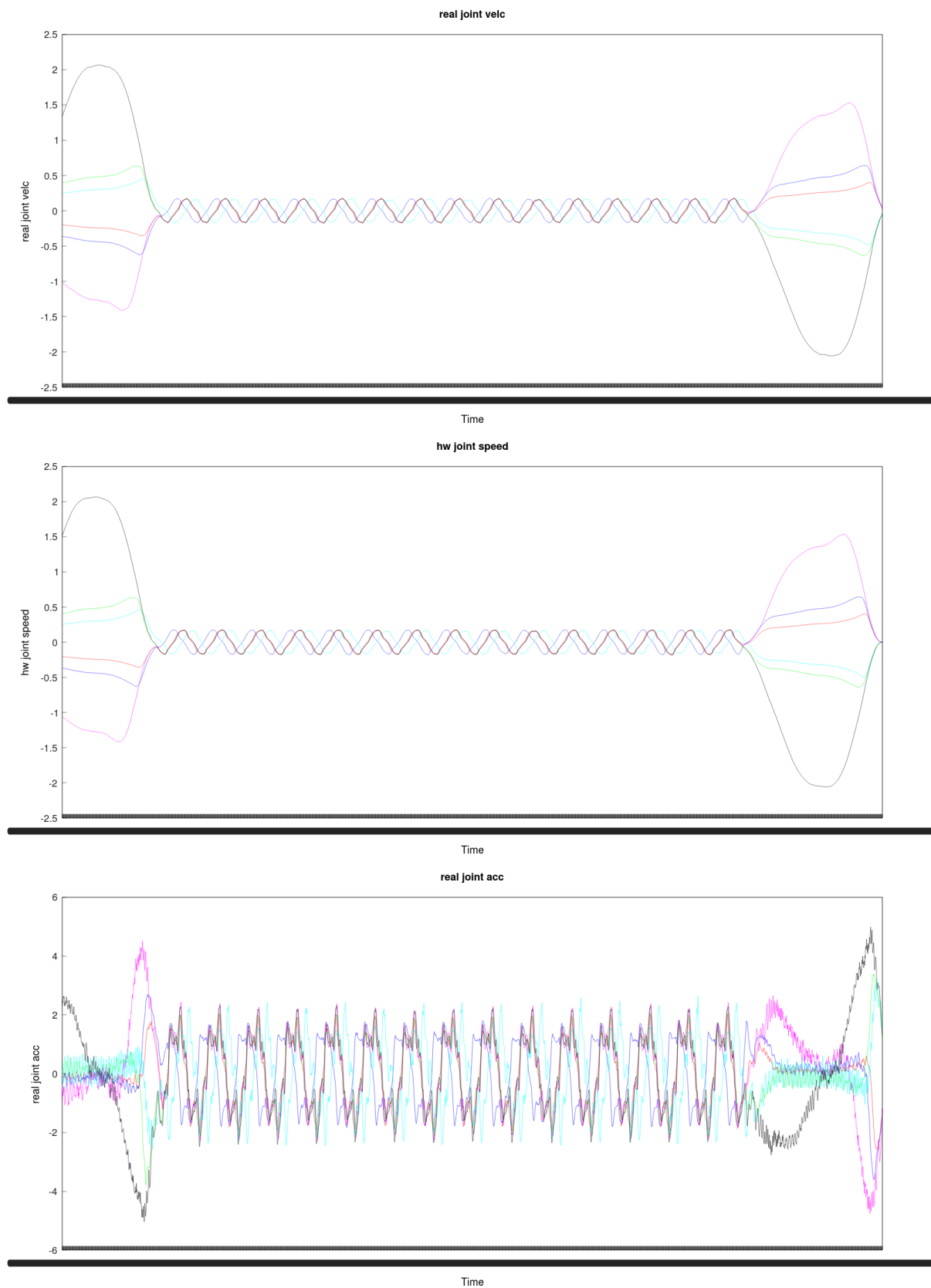
Touch the robot arm by hand to feel it. There is some shaking during the ascent and descent of the robot arm, but there is no shaking phenomenon during the motion except ascent and descent.

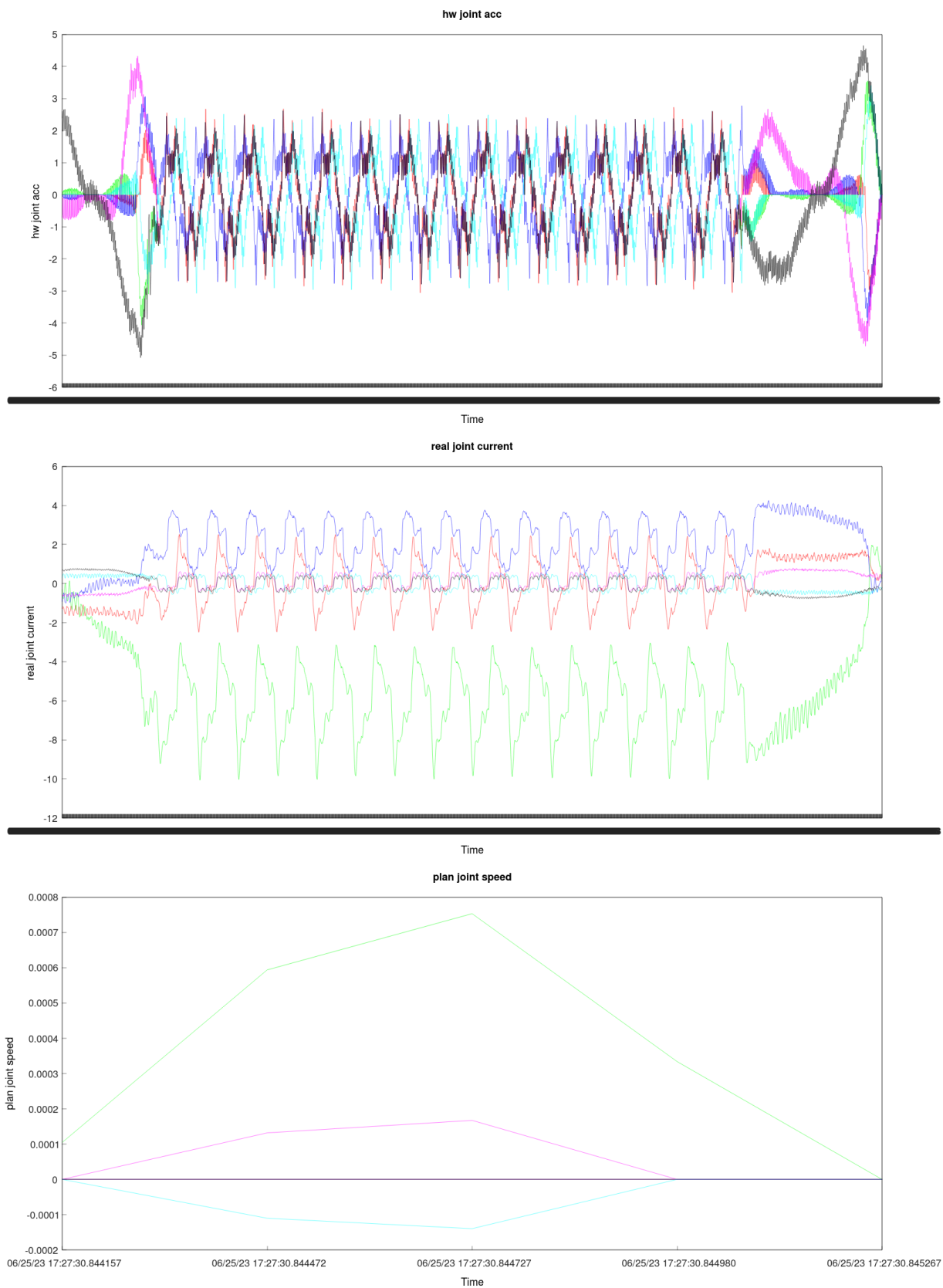
- Quantitative Analysis

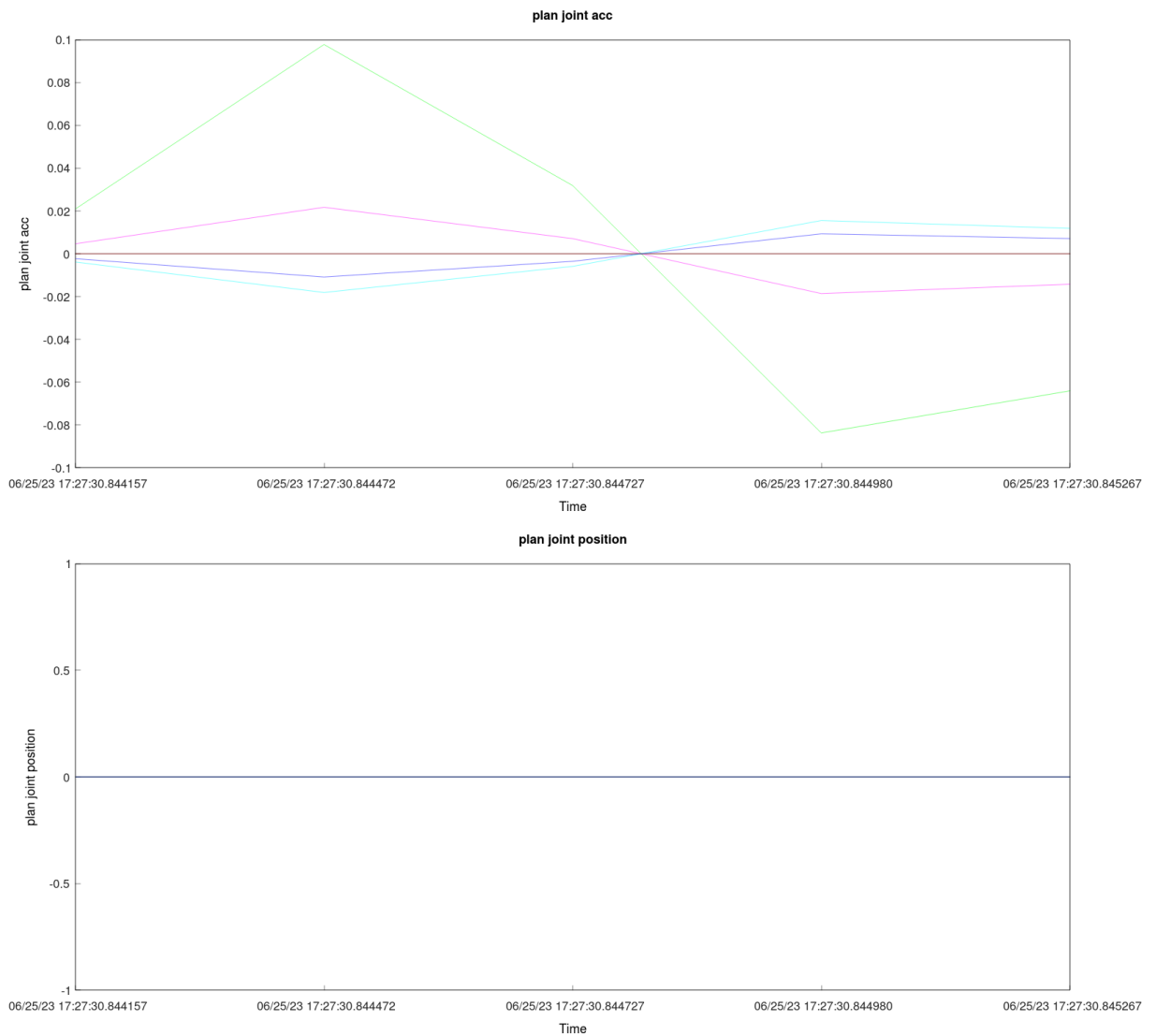
Real Robot Arm (running `aubo_control1` in the control box)

From the figure below, it can be analyzed that the robot arm basically does not vibrate.



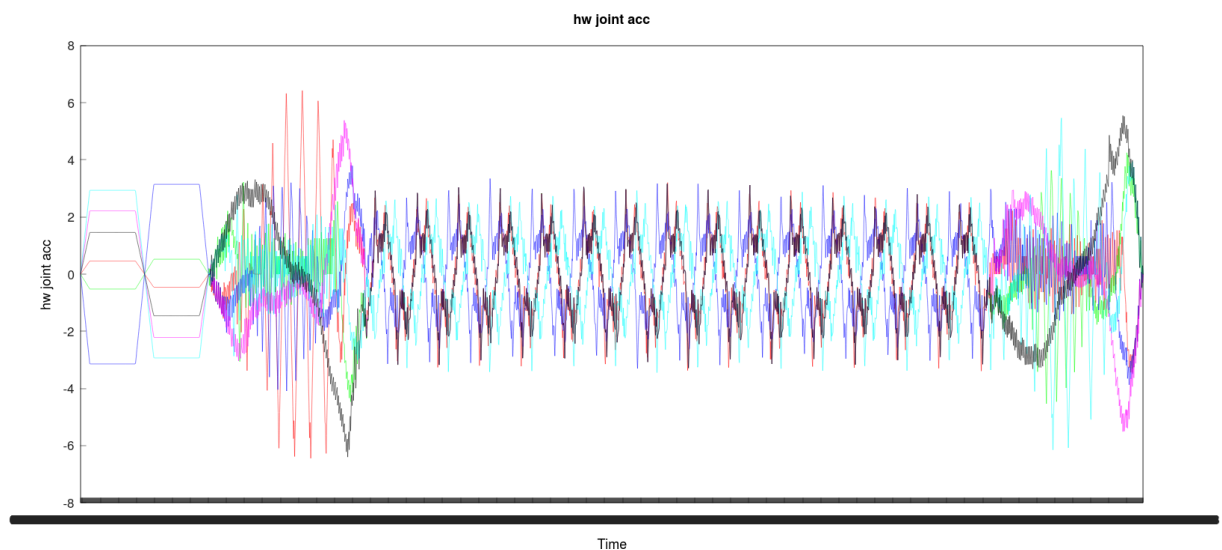


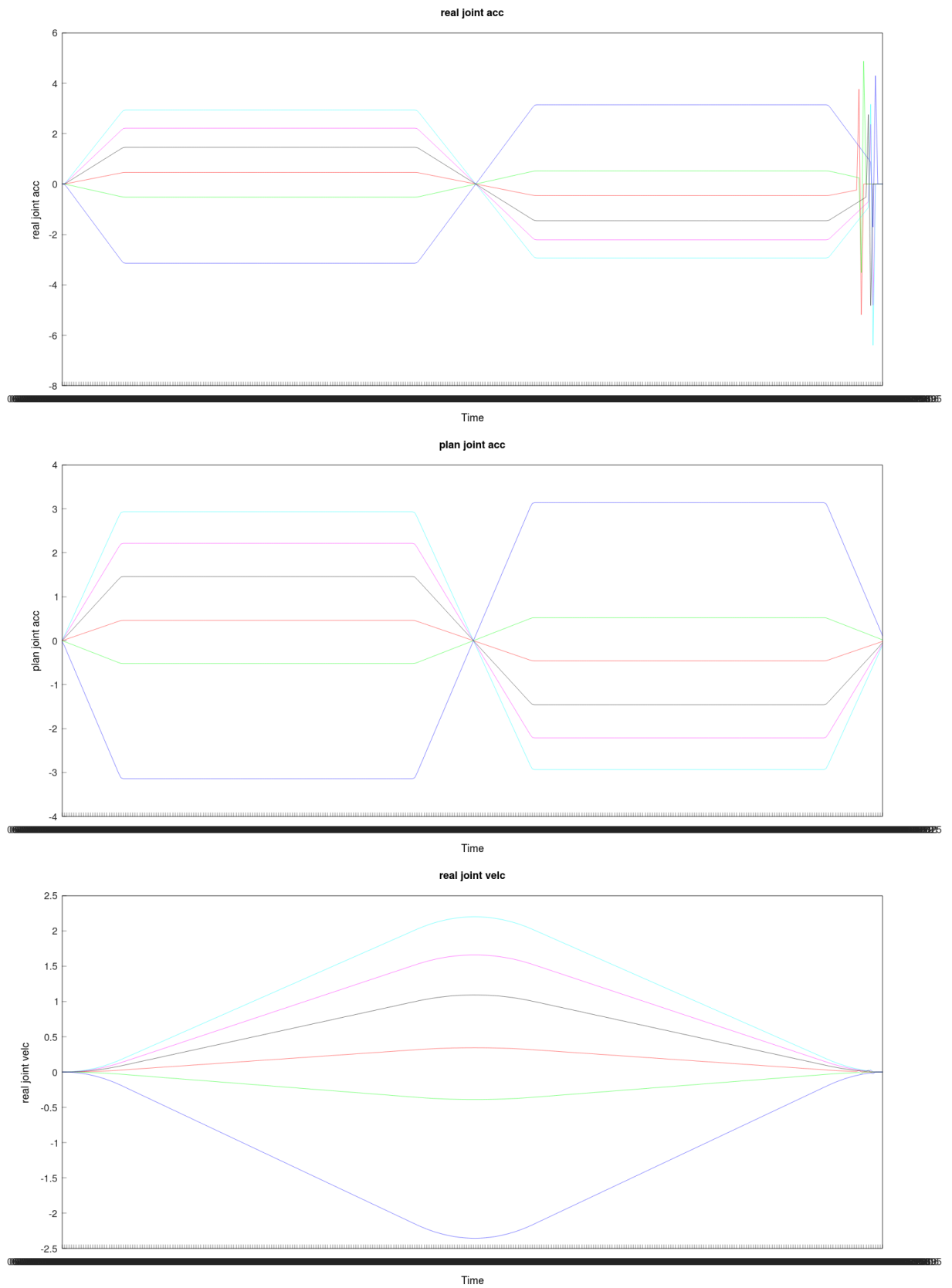


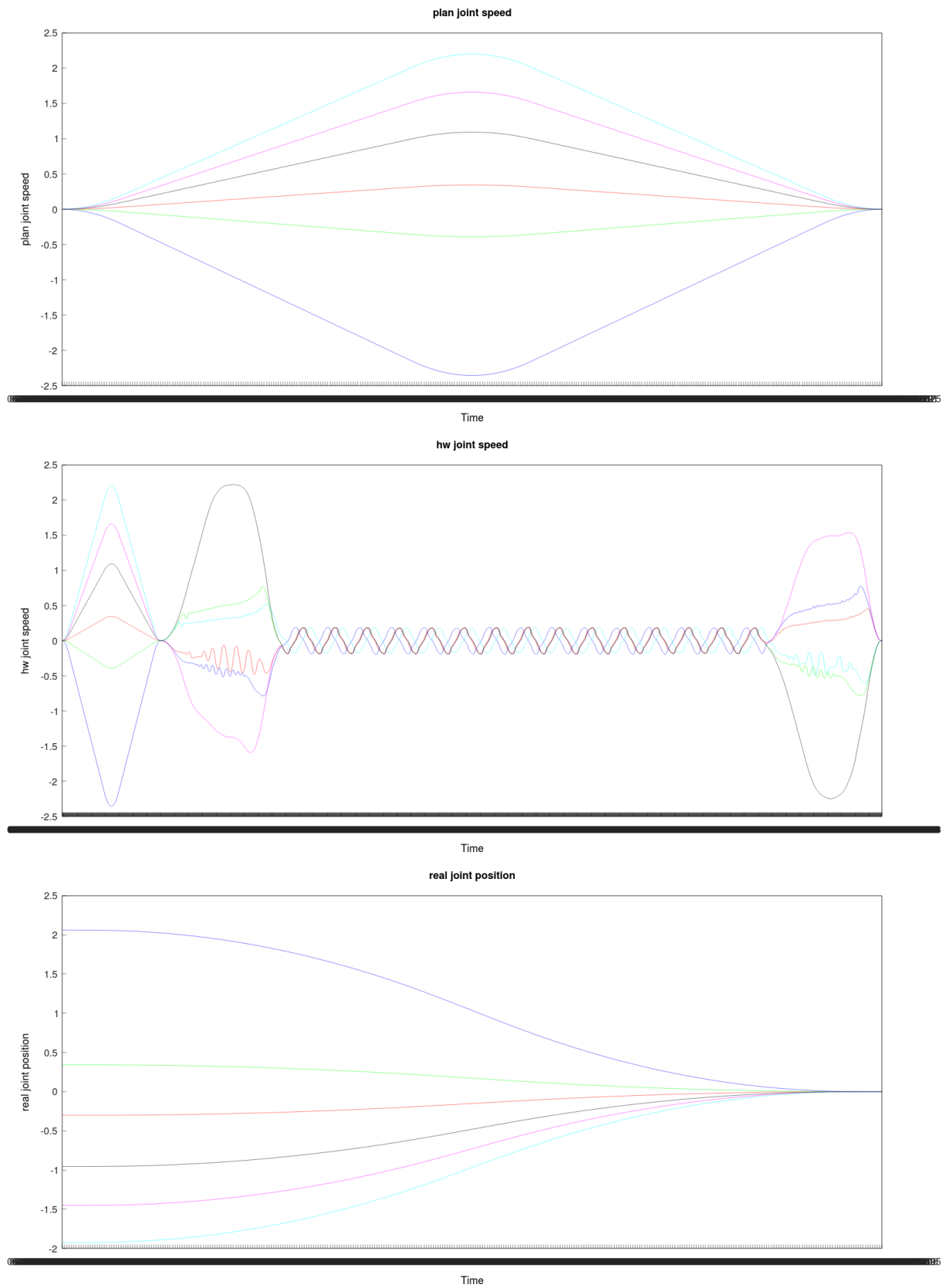


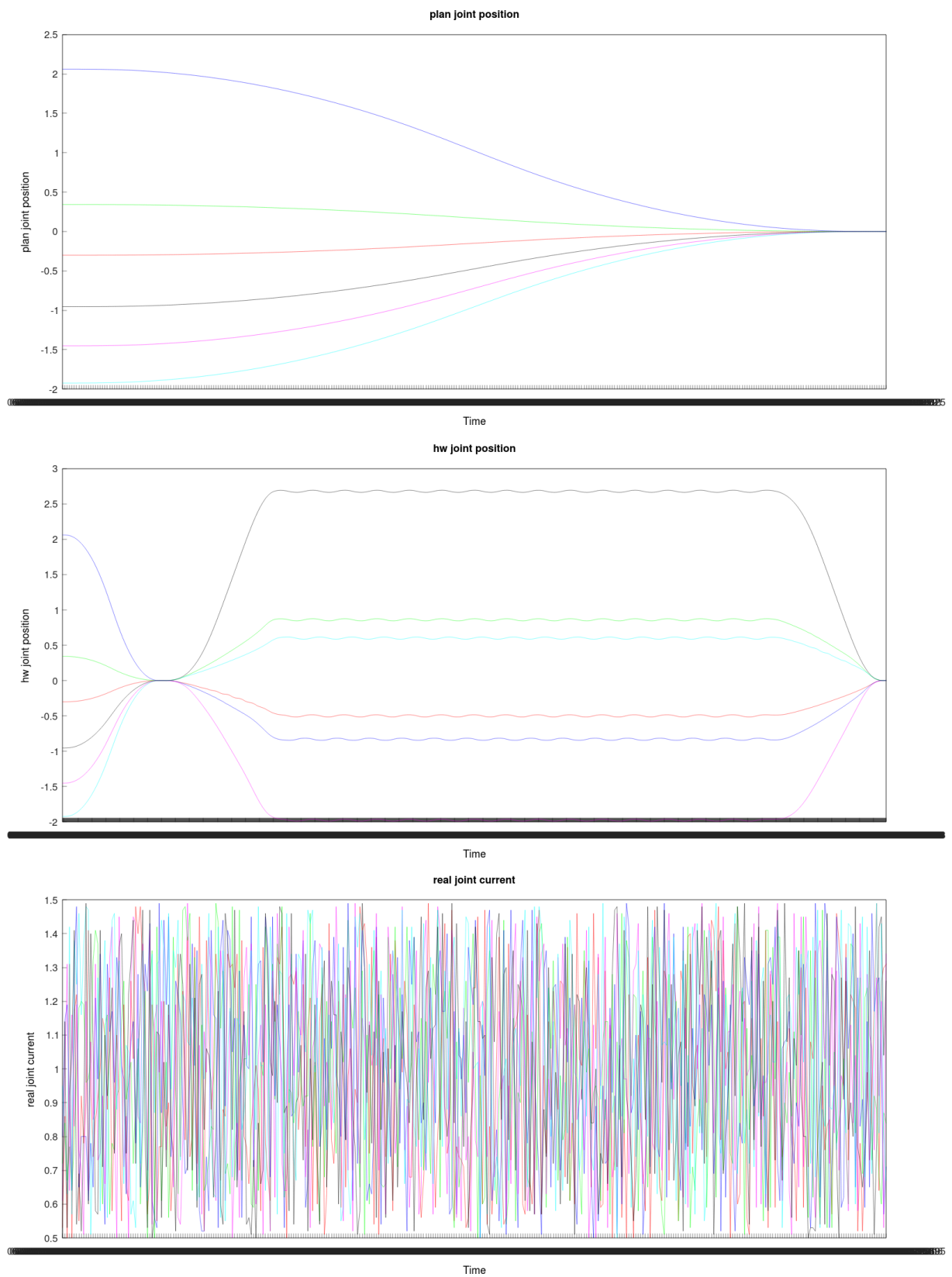
Simulated Robot Arm (running "aubo_control" on the virtual machine)

From the figure below, it can be seen that in the `hw joint speed` graph, the robot arm has some shaking during the ascent and descent.









2.2.2 Test Results

System Platform	CPU	Test Result
-----------------	-----	-------------

<code>windows</code> platform (without live patching)	<code>11th Gen Intel(R) Core(TM) i7-1165G7 @ 2.80GHz</code>	Significant vibration
<code>linux ubuntu16.04</code> platform (without live patching)	<code>11th Gen Intel(R) Core(TM) i7-1165G7 @ 2.80GHz</code>	Slight vibration
<code>linux ubuntu16.04</code> platform (with live patching)	Control box (real robot arm): <code>Intel(R) Celeron(R) CPU N3160 @ 1.60GHz</code> virtual machine (simulated robot arm): <code>11th Gen Intel(R) Core(TM) i7-1165G7 @ 2.80GHz</code>	Minimal vibration

2.2.3 Comparative Analysis of Test Results

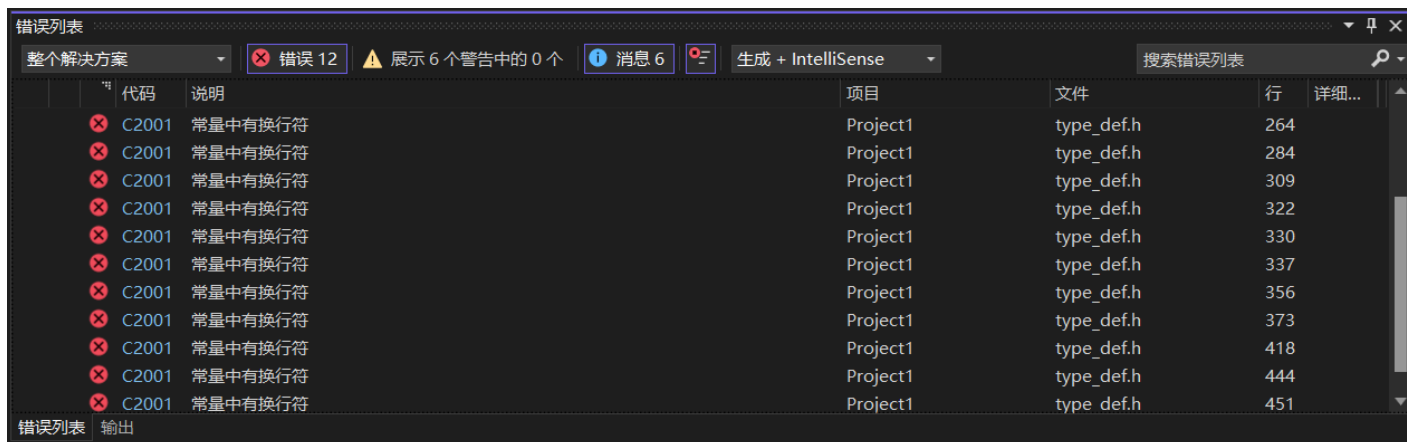
Known from the analysis above, the degree of vibration of the robot arm is as follows:

`windows` platform (without live patching) > `linux ubuntu16.04` platform (without live patching) > `linux ubuntu16.04` platform (with live patching). Among them, the `linux ubuntu16.04` platform (with live patching) has the best effect, the difference in effect between `linux ubuntu16.04` platform (without live patching) and `linux ubuntu16.04` platform (with live patching) is not significant, and the `windows` platform (without live patching) has the worst effect, with significant vibration.

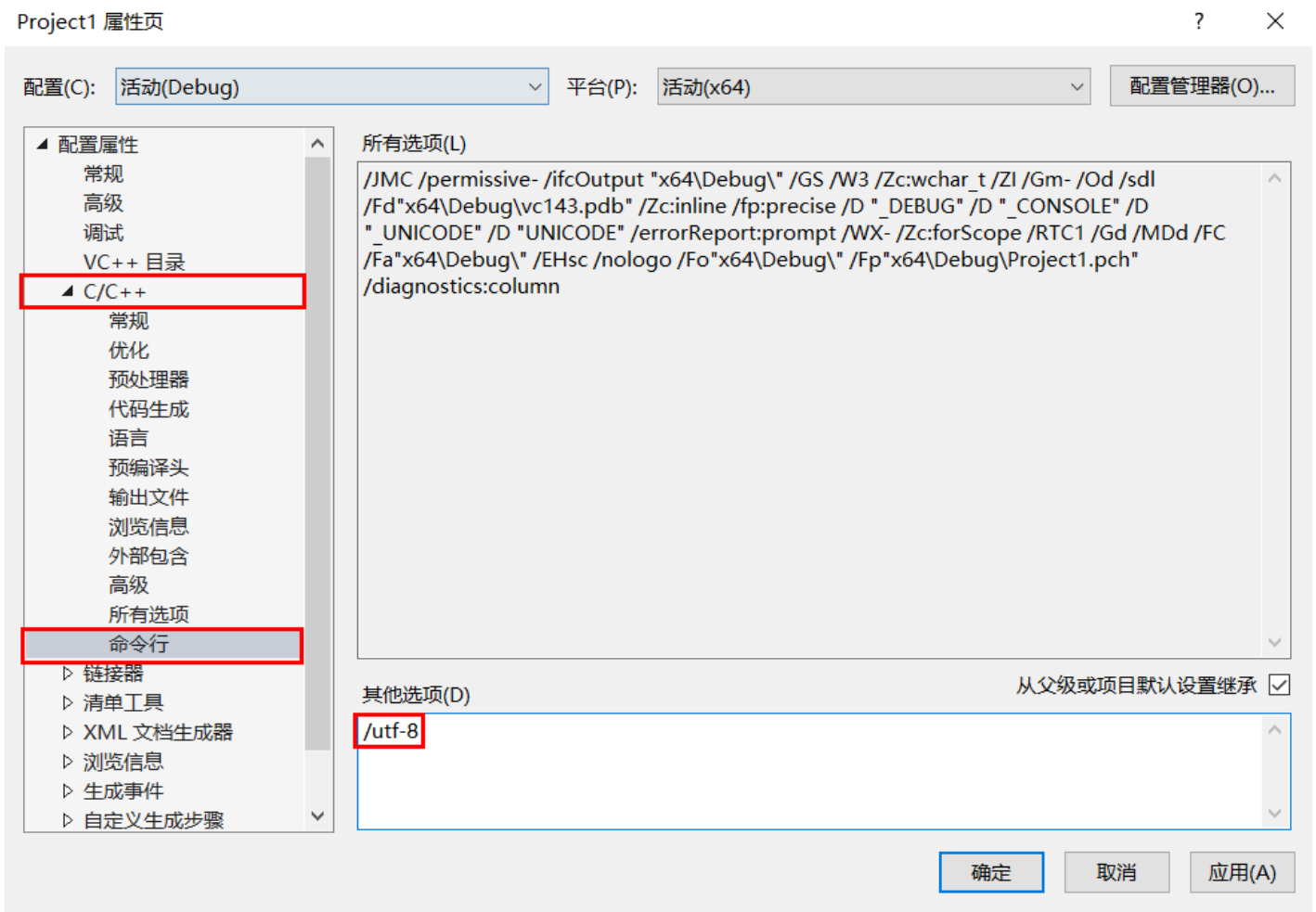
Frequently Asked Questions

Q1. When creating a new C++ project in Visual Studio, after including the AUBO SDK header and library files, a compilation error occurs in type_def.h "newline in constant".

Issue Screenshot:



Solution: Set the compiler option /utf-8 in your project settings. **Note:** If there are multiple parameters, please separate them with **spaces**.



Q2. In Visual Studio, the /utf-8 option has been set, but the code contains Chinese characters, and a compilation error occurs: "newline in constant".

Issue Screenshot:


```

65  ...// 接口调用: 获取机器人的名字
66  ...auto robot_name = cli->getRobotNames().front();
67  ...auto robot_interface = cli->getRobotInterface(robot_name);
68
69  ...// 接口调用: 设置机械臂的速度比率
70  ...robot_interface->getMotionControl()->setSpeedFraction(0.3);
71
72  ...// 关节角, 单位: 弧度
73  ...std::vector<double> joint_angle = {
74  ...0.0 * (M_PI / 180), -15.0 * (M_PI / 180), 100.0 * (M_PI / 180),
75  ...25.0 * (M_PI / 180), 90.0 * (M_PI / 180), 0.0 * (M_PI / 180)
76  ...};
77
78  ...// 接口调用: 关节运动
79  ...robot_interface->getMotionControl()->moveJoint(
80  ...joint_angle, 80 * (M_PI / 180), 60 * (M_PI / 180), 0, 0);
81  ...// 阻塞
82  ...int ret = waitArrival(robot_interface);
83  ...if (ret == 0)
84  ...{
85  ...std::cout << "关节运动到初始位置成功" << std::endl;
86  ...}
87  ...else
88  ...{
89  ...std::cout << "关节运动到初始位置失败" << std::endl;
90  ...}
91
92  ...// 接口调用: 开启servo模式
93  ...robot_interface->getMotionControl()->setServoMode(true);
94
95  ...// 等待进入 servo 模式
96  ...int i = 0;
97  ...while (!robot_interface->getMotionControl()->isServoModeEnabled()) {
98  ...if (i++ > 5) {

```

99 % 未找到相关问题 行: 54 字符: 6 空格 CRLF

错误列表

整个解决方案 错误 2 展示 3607 个警告中的 0 个 展示 31 个消息中的 0 个 生成 + IntelliSense 搜索错误列表

代码	说明	项目	文件	行	禁止显示状态
C2001	常量中有换行符	TestAUBORobot	TestZGAutoPath.cpp	89	
C2143	语法错误: 缺少";" (在")"的前面)	TestAUBORobot	TestZGAutoPath.cpp	90	

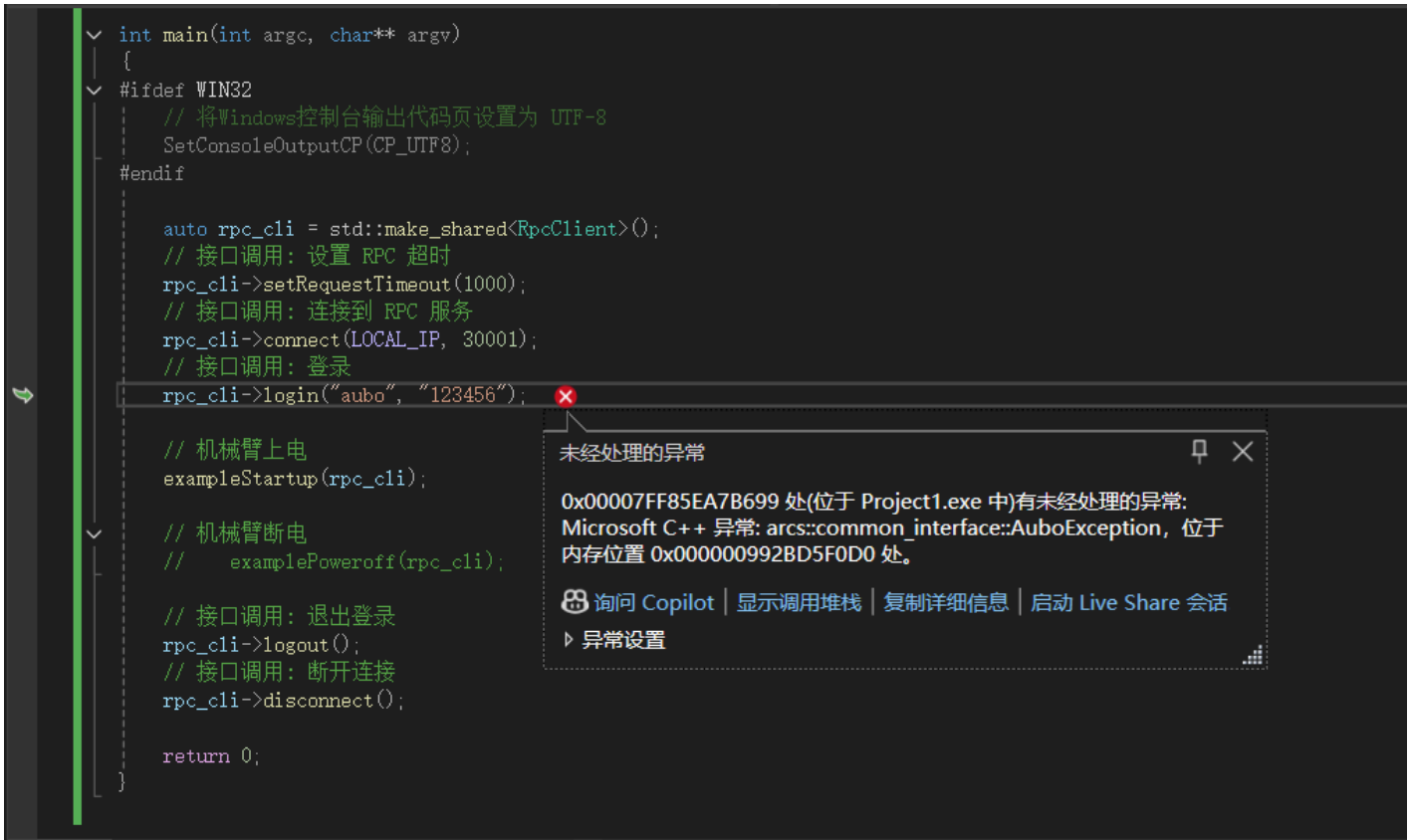
错误列表 输出 查找符号结果

Solution:

- Solution 1: End Chinese string with a space.
- Solution 2: Change Chinese string to an English string.

Q3. When running the program in Visual Studio, an unhandled AuboException is thrown, causing the program to crash.

Issue Screenshot:



Solution: Use `AuboException` for exception handling in the code. For specific methods, please refer to [Exception Handling](#).

Q4. RPC connection failed, SDK log error "Connect to <IP_ADDRESS>:<PORT> FAILED".

Issue Screenshot:

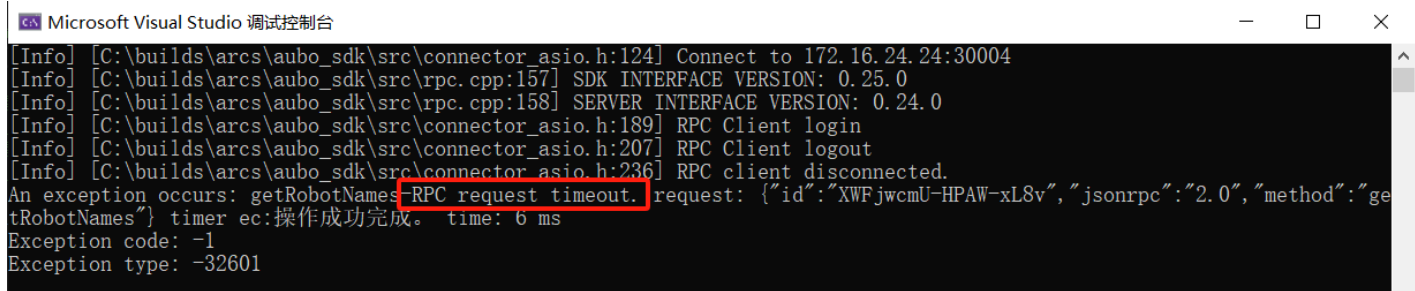


Solution:

1. The <IP_ADDRESS>:<PORT> shown in the console SDK logs indicates the target robot's IP address and RPC port number. Please note that the RPC port number is fixed at 30004.
2. Check if the robot IP address and RPC port number in the code are correct.
3. Check the network connection between the local computer and target robot. You can use the ping command to test. For detailed instructions, refer to [SDK PC to Robot Communication Setup](#).

Q5. The try/catch block caught an AuboException with the message "RPC request timeout".

Issue Screenshot:



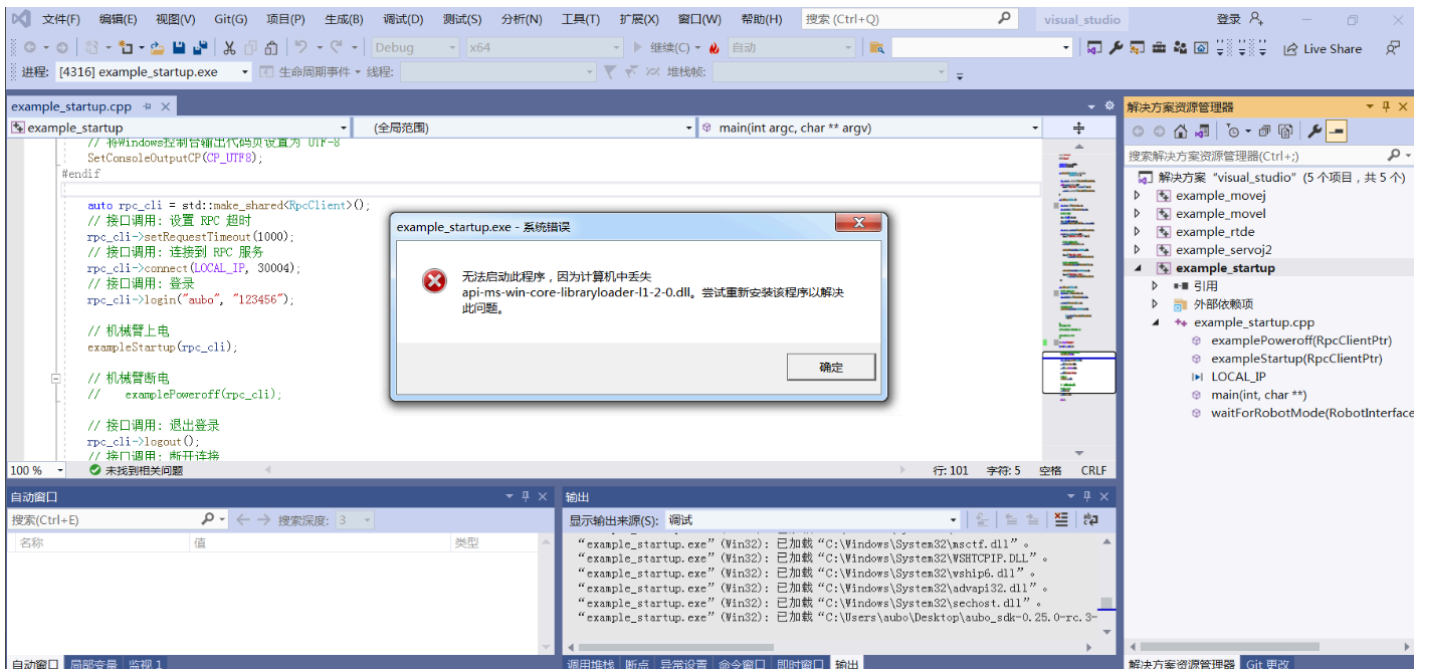
```
Microsoft Visual Studio 调试控制台
[Info] [C:\builds\arcs\aubo_sdk\src\connector_asio.h:124] Connect to 172.16.24.24:30004
[Info] [C:\builds\arcs\aubo_sdk\src\rpc.cpp:157] SDK INTERFACE VERSION: 0.25.0
[Info] [C:\builds\arcs\aubo_sdk\src\rpc.cpp:158] SERVER INTERFACE VERSION: 0.24.0
[Info] [C:\builds\arcs\aubo_sdk\src\connector_asio.h:189] RPC Client login
[Info] [C:\builds\arcs\aubo_sdk\src\connector_asio.h:207] RPC Client logout
[Info] [C:\builds\arcs\aubo_sdk\src\connector_asio.h:236] RPC client disconnected.
An exception occurs: getRobotNames-RPC request timeout. request: {"id":"XWFjwcmU-HPAW-xL8v","jsonrpc":"2.0","method":"getRobotNames"} timer ec:操作成功完成。 time: 6 ms
Exception code: -1
Exception type: -32601
```

Solution:

1. Ensure that you have called `setRequestTimeout` to set the request timeout before invoking the RPC client's connect interface. If not, please add `setRequestTimeout(1000)`.
2. If `setRequestTimeout` is already called, try to increase the time. For example, change the input to 3000 to improve the request success rate.

Q6. When running the C++ example from the SDK package on Windows 7, the error occurs: "This program can't run because api-ms-win-core-libraryloader-l1-2-0.dll is missing from the computer."

Issue Screenshot:



Cause of Issue: The cryptbase.dll file in the lib folder of the SDK package is not compatible with the cryptbase.dll file that comes with the Windows 7 system.

Solution:

- Solution 1: Only import aubo_sdkd.dll, aubo_sdk.dll, aubo_sdkd.lib and aubo_sdk.lib from the lib folder.
- Solution 2: Delete cryptbase.dll from the lib folder.
- Solution 3: Replace the cryptbase.dll file in the lib folder with the local computer's C:\Windows\System32\cryptbase.dll.

Q7. When doing development with the C++ SDK, is it necessary to load all the DLL files in the lib folder of the SDK package?

Solution: Only aubo_sdkd.dll and aubo_sdk.dll are needed.

Q8. What to do when Gradle sync verification fails to resolve dependencies?

Solution:

1. Check the private repository address: Verify the Maven repository configuration in `settings.gradle.kts`, ensuring the address is accessible and free of typos.

2. Test Nexus network connectivity: Verify that the current network can properly access the AUBO internal Nexus repository, and check for proxy/firewall blocks.
3. Confirm `settings.gradle.kts` loading status: Ensure the file has correct syntax and paths, and there are no loading failure logs during Gradle sync.

Aubo Sdk Error Code

Abbreviations

- JNT: joint
- PDL: pedstral
- TP: teach pendant
- COMM: communication
- ENC: encoder
- CURR: current
- POS: position
- PKG: package
- PROG: program

Error Codes

Error Name	Error Code	Description	Recommend
DEBUG	0	Debug message {}	suggest..
POPUP	1	Popup title: {}, msg: {}, mode: {}	suggest..
POPUP_DISMISS	2	{}	suggest..
SYSTEM_HALT	3	{}	suggest..
INV_ARGUMENTS	4	Invalid arguments.	suggest..
USER_NOTIFY	5	{}	suggest..
MODBUS_SIGNAL_CREATED	10	Modbus signal {} created.	suggest..
MODBUS_SIGNAL_REMOVED	11	Modbus signal {} removed.	suggest..

MODBUS_SIGNAL_VALUE_CHANGED	12	Modbus signal {} value changed to {}	suggest..
RUNTIME_CONTEXT	13	tid: {} lineno: {} index: {} comment: {}	suggest..
INTERP_CONTEXT	14	tid: {} lineno: {} index: {} comment: {}	suggest..
PROGRAM_LOADED	15	program loaded: {}	suggest..
TASK_DELETED	16	tid: {}	was delete
MODBUS_SLAVE_BIT	20	Modbus slave address: {} value {} 	suggest..
MODBUS_SLAVE_REG	21	Modbus slave address: {} value {} 	suggest..
PNIO_SLAVE_SLOT_VALUE	30	PNIO slot: {} subslot {} index {} value {}	suggest..
PNIO_CONNECT_STATUS	31	PNIO connection status changed to {}	suggest..
PNIO_DEVICE_NAME	32	PNIO device name changed to {}	suggest..
PNIO_IP	33	PNIO ip {} mask {} gateway {}	suggest..
ICM_SERVER_STATUS	40	ICM server status changed to {}	suggest..
EIP_X	50	ICM slot: {} index {} subindex {}	suggest..

LOG_PROGRAM_SUCCESS	100	[[{}]] Load program {} successful	suggest..
LOG_PROGRAM_FAILED	101	[[{}]] Load program {} failed, file not found	suggest..
LOG_PROGRAM_FAILED2	102	[[{}]] Load program {} failed, configuration file (.ins) does not match	suggest..
JOINT_ERR_OVER_CURRENET	10001	joint{} error: over current	(a) Check for circuit. (b) Complete reboot sequence. (c) happens more than two times in a row replace joint
JOINT_ERR_OVER_VOLTAGE	10002	joint{} error: over voltage	(a) Do a Complete rebooting sequence (b) Check 48 V supply, current distributor, error eater and Control Board for issues
JOINT_ERR_LOW_VOLTAGE	10003	joint{} error: low voltage	(a) Do a Complete rebooting sequence (b) Check for circuit in robot (c) Check 48 V supply, current distributor, error eater and Control Board for issues
JOINT_ERR_OVER_TEMP	10004	joint{} error: over temperature	(a) Check robot environment make sure the robot is operating within recommended

			(b) Do a Com rebooting seq
JOINT_ERR_HALL	10005	joint{} error: hall	suggest..
JOINT_ERR_ENCODER	10006	joint{} error: encoder	Check enco connectio
JOINT_ERR_ABS_ENCODER	10007	joint{} error: abs encoder	suggest..
JOINT_ERR_Q_CURRENT	10008	joint{} error: detect current	suggest..
JOINT_ERR_ENC_POLL	10009	joint{} error: encoder pollustion	suggest..
JOINT_ERR_ENC_Z_SIGNAL	10010	joint{} error: enocder z signal	suggest..
JOINT_ERR_ENC_CAL	10011	joint{} error: encoder calibrate	suggest..
JOINT_ERR_IMU_SENS	10012	joint{} error: IMU sensor	suggest..
JOINT_ERR_TEMP_SENS	10013	joint{} error: TEMP sensor	suggest..
JOINT_ERR_CAN_BUS	10014	joint{} error: CAN bus error	suggest..
JOINT_ERR_SYS_CUR	10015	joint{} error: system current error	suggest..
JOINT_ERR_SYS_POS	10016	joint{} error: system position error	suggest..
JOINT_ERR_OVER_SP	10017	joint{} error: over speed	suggest..
JOINT_ERR_OVER_ACC	10018	joint{} error: over accelerate	suggest..
JOINT_ERR_TRACE	10019	joint{} error: trace accuracy	suggest..

JOINT_ERR_TAG_POS_OVER	10020	joint{} error: target position out of range	suggest..
JOINT_ERR_TAG_SP_OVER	10021	joint{} error: target speed out of range	suggest..
JOINT_ERR_COLLISION	10022	joint{} error: collision	suggest..
JOINT_ERR_COMMON	10023	joint{} error: unkown error. Check communication with joint.	suggest..
JOINT_ERR_SWITCH_SERVO_MODE	10024	joint{} error: switch servo mode timeout.	suggest..
JOINT_ERR_MOTOR_STUCK	10025	joint{} error: motor stucked.	suggest..
JOINT_ERR_REDUCER_OVER_TEMP	10026	joint{} error: reducer over temperature	(a) Check rob environment make sure the is operating v recommended (b) Do a Com rebooting seq
JOINT_ERR_REDUCER_NTC	10027	joint{} error: reducer TEMP sensor failure	(a) Check rob environment make sure the is operating v recommended (b) Do a Com rebooting seq
JOINT_ERR_ABS_MULTITURN	10028	joint{} error: absolute encoder multiturn error	(a) Check rob environment make sure the is operating v recommended (b) Do a Com rebooting seq

JOINT_ERR_ADC_ZERO_OFFSET	10029	joint{} error: ADC zero offset failure	(a) Check rob environment make sure the is operating v recommended (b) Do a Com rebooting seq
JOINT_ERR_SHORT_CIRCUIT	10030	joint{} error: short circuit	(a) Check rob environment make sure the is operating v recommended (b) Do a Com rebooting seq
JOINT_ERR_PHASE_LOST	10031	joint{} error: motor phase lost	(a) Check rob environment make sure the is operating v recommended (b) Do a Com rebooting seq
JOINT_ERR_BRAKE	10032	joint{} error: brake failure	(a) Check rob environment make sure the is operating v recommended (b) Do a Com rebooting seq
JOINT_ERR_FIRMWARE_UPDATE	10033	joint{} error: firmware update failure	(a) Check rob environment make sure the is operating v recommended (b) Do a Com rebooting seq
JOINT_ERR_BATTERY_LOW	10034	joint{} error: battery low	(a) Check rob environment make sure the is operating v recommended (b) Do a Com rebooting seq

JOINT_ERR_PHASE_ALIGN	10035	joint{} error: phase align	(a) Check rob environment make sure the is operating v recommended (b) Do a Com rebooting seq
JOINT_ERR_CAN_HW_FAULT	10036	joint{} error: CAN bus hw fault	(a) Check rob environment make sure the is operating v recommended (b) Do a Com rebooting seq
JOINT_ERR_POS_DISCONTINUOUS	10037	joint{} error: target position discontinuous	(a) Check rob environment make sure the is operating v recommended (b) Do a Com rebooting seq
JOINT_ERR_POS_INIT	10038	joint{} error: position initiallization failure	(a) Check rob environment make sure the is operating v recommended (b) Do a Com rebooting seq
JOINT_ERR_TORQUE_SENSOR	10039	joint{} error: torqure sensor failure	(a) Check rob environment make sure the is operating v recommended (b) Do a Com rebooting seq
JOINT_ERR_OFFLINE	10040	joint{} error: joint may be offline	(a) Check joi hardware. (b) joint's id
JOINT_ERR_BOOTLOADER	10041	joint{} error: The joint is in bootloader mode. Retry	suggest..

		firmware update.	
JOINT_ERR_SLAVE_OFFLINE	10042	slave joint{} error: slave joint may be offline	(a) Check slave hardware. (b) slave joint's
JOINT_ERR_SLAVE_BOOTLOADER	10043	slave joint{} error: The slave joint is in bootloader mode. Retry firmware update.	suggest..
IFB_ERR_ROBOTTYPE	20001	Robot error type!	suggest..
IFB_ERR_ADXL_SENS	20002	Acceleration sensor error!	suggest..
IFB_ERR_EN_LINE	20003	Encoder line error!	suggest..
IFB_ERR_ENTER_HDG_MODE	20004	Robot enter handguide mode!	suggest..
IFB_ERR_EXIT_HDG_MODE	20005	Robot exit handguide mode!	suggest..
IFB_ERR_MAC_DATA_BREAK	20006	MAC data break!	suggest..
IFB_ERR_DRV_FIRMWARE_VERSION	20007	Motor driver firmware version error!	suggest..
INIT_ERR_EN_DRV	20008	Motor driver enable failed!	suggest..
INIT_ERR_EN_AUTO_BACK	20009	Motor driver enable auto back failed!	suggest..
INIT_ERR_EN_CUR_LOOP	20010	Motor driver enable current loop failed!	suggest..

INIT_ERR_SET_TAG_CUR	20011	Motor driver set target current failed!	suggest..
INIT_ERR_RELEASE_BRAKE	20012	Motor driver release brake failed!	suggest..
INIT_ERR_EN_POS_LOOP	20013	Motor driver enable position loop failed!	suggest..
INIT_ERR_SET_MAX_ACC	20014	Motor set max accelerate failed!	suggest..
SAFETY_ERR_PROTECTION_STOP_TIMEOUT	20015	Protective stop timeout!	suggest..
SAFETY_ERR_REDUCED_MODE_TIMEOUT	20016	Reduced mode timeout!	suggest..
SYS_ERR_MCU_COM	20017	Robot system error: mcu communication error!	suggest..
SYS_ERR_RS485_COM	20018	Robot system error: RS485 communication error!	suggest..
IFB_ERR_DISCONNECTED	20019	Interface board may be disconnected. Please check connection between IPC and Interface board.	suggest..
IFB_ERR_PAYLOAD_ERROR	20020	Payload error.	suggest..
IFB_OFFLINE	20021	ifaceboard error: ifaceboard may be offline	(a) Check ifaceboard hardware. (b) Check ifaceboard's power supply.

IFB_ERR_BOOTLOADER	20022	interface board error: The interface board is in bootloader mode. Retry firmware update.	suggest..
IFB_SLAVE_OFFLINE	20023	interface slave board error: interface slave board may be offline	(a) Check interface slave board hardware. (b) Check interface slave board's ic
IFB_SLAVE_ERR_BOOTLOADER	20024	interface slave board error: The interface slave board is in bootloader mode. Retry firmware update.	suggest..
ROBOT_BE_PULLING	30001	Something is pulling the robot.	Please check configuration, payload and mount settings
PSTOP_ELBOW_POS	30002	Protective Stop: Elbow position close to safety plane limits.	Please move Elbow joint away from the safety
PSTOP_STOP_TIME	30003	Protective Stop: Exceeding user safety settings for stopping time.	(a) Check speed accelerations program (b) Check usage of TCP, payload and CoG correction. Check external equipment activation if correctly
PSTOP_STOP_DISTANCE	30004	Protective Stop: Exceeding user safety settings for stopping distance.	(a) Check speed accelerations program (b) Check usage of TCP, payload and CoG correction. Check external

			equipmentacti if correctly
PSTOP_CLAMP	30005	Protective Stop: Danger of clamping between the Robot' s lower arm and tool.	(a) Check spee accelerations program (b) C usage of TCP,p and CoG corre Check exter equipmentacti if correctly
PSTOP_POS_LIMIT	30006	Protective Stop: Position close to joint limits	suggest..
PSTOP_ORI_LIMIT	30007	Protective Stop: Tool orientation close to limits	suggest..
PSTOP_PLANE_LIMIT	30008	Protective Stop: Position close to safety plane limits	suggest..
PSTOP_POS_DEVIATE	30009	Protective Stop: Position deviates from path	Check paylo center of gravi acceleration se
JOINT_CHK_PAYLOAD	30010	Joint {}: Check payload, center of gravity and acceleration settings. Log screen may contain additional information.	suggest..
PSTOP_SINGULARITY	30011	Protective Stop: Position in singularity.	Please use Mo change the m
PSTOP_CANNOT_MAINTAIN	30012	Protective Stop: Robot cannot maintain its position, check	suggest..

		if payload is correct	
PSTOP_WRONG_PAYLOAD	30013	Protective Stop: Wrong payload or mounting detected, or something is pushing the robot when entering Freedrive mode	Verify that the configuration mounting in the installation is correct
PSTOP_JOINT_COLLISION	30014	Protective Stop: Collision detected by joint {}	Make sure no obstacles are in the path of the robot and resume the program
PSTOP_POS_DISAGREE	30015	Protective stop: The robot was powered off last time due to a joint position disagreement.	(a) Verify that the robot position matches the 3D graphics model. (b) Verify that the real robot position is correct. (c) Verify that the encoders function correctly before releasing the brakes. Stand back and monitor the robot performance. (d) If the first program cycle is not as expected. (e) If the position is not correct, the robot must be repaired. (f) In this case, click Off Robot. (g) If the position is correct, please tick the box below the 3D graphics and Robot Position Verified.
TARGET_JOINT_SPEED_EXCEED	30016	Target joint speed exceed limits	suggest..
TARGET_POS_SUDDEN_CHG	30017	Sudden change in target	suggest..

		position	
SUDDEN_STOP	30018	Sudden stop.	To abort a move, use "stopj" or "stopl" script command to generate a smooth deceleration by using "wait". After aborting motion, there is a delay between waypoints with blend.
ROBOT_STOP_ABNORMAL	30019	Robot has not stopped in the allowed reaction and braking time	suggest..
PROG_INVALID_SETP	30020	Robot program resulted in invalid setpoint.	Please review waypoints in program
BLEND_INVALID_SETP	30021	Blending failed and resulted in an invalid setpoint.	Try changing blend radius or contact technique for support
APPROACH_SINGULARITY	30022	Robot approaching singularity – Acceleration threshold failed.	Review waypoints in the program using MoveJ instead of MoveL in position close to singularity
TSPEED_UNMATCH_POS	30023	Target speed does not match target position	suggest..
INCONSIS_TPOS_SPD	30024	Inconsistency between target position and speed	suggest..

JOINT_TSPD_UNMATCH_POS	30025	Target joint speed does not match target joint position change – Joint {}	suggest..
FIELDBUS_INPUT_DISCONN	30026	Fieldbus input disconnected.	Please check fieldbus connections (ModBus, EtherCAT and Profinet) and disable the fieldbus watchdog feature if the installation is not correct. Check RTI watchdog feature. Check if a URCL is using this feature.
OPMODE_CHANGED	30027	Operational mode changed: {}	suggest..
NO_KIN_CALIB	30028	No Kinematic Calibration found (calibration.conf file is either corrupt or missing).	A new kinematic calibration may be needed if the robot needs to improve its kinematic accuracy. Otherwise, ignore the message.
KIN_CALIB_UNMATCH_JOINT	30029	Kinematic Calibration for the robot does not match the joint(s).	If moving a program from a different robot to this one, rekinematic calibration is needed for the second robot to improve kinematic accuracy. Otherwise, ignore the message.
KIN_CALIB_UNMATCH_ROBOT	30030	Kinematic Calibration does not match the robot.	Please check the serial number of the robot arm matches the Control Panel.
JOINT_OFFSET_CHANGED	30031	Large movement of the robot detected while	suggest..

		it was powered off. The joints were moved while it was powered off, or the encoders do not function	
OFFSET_CHANGE_HIGH	30032	Change in offset is too high	suggest..
JOINT_SPEED_LIMIT	30033	Close to joint speed safety limit.	Review prog speed an acceleratio
TOOL_SPEED_LIMIT	30034	Close to tool speed safety limit.	Review prog speed an acceleratio
MOMENTUM_LIMIT	30035	Close to momentum safety limit.	Review prog speed an acceleratio
ROBOT_MV_STOP	30036	Robot is moving when in Stop Mode	suggest..
HAND_PROTECTION	30037	Hand protection: Tool is too close to the lower arm: {} meter.	(a) Check w position. (b) \ mounting (c) Complete rebo sequence (d) U software (e) Co your local AI Robots serv provider f assistanc
WRONG_SAFETYMODE	30038	Wrong safety mode: {}	suggest..
SAFETYMODE_CHANGED	30039	Safety mode changed: {}	suggest..
JOINT_ACC_LIMIT	30040	Close to joint acceleration safety limit	suggest..

TOOL_ACC_LIMIT	30041	Close to tool acceleration safety limit	suggest..
JOINT_TEMPERATURE_LIMIT	30042	Joint {} temperature too high(>{}°C)	suggest..
CONTROL_BOX_TEMPERATURE_LIMIT	30043	Control box temperature too high(>{}°C)	suggest..
ROBOT_EMERGENCY_STOP	30044	Robot emergency stop	suggest..
ROBOTMODE_CHANGED	30045	Robot mode changed: {}	suggest..
ROBOTMODE_ERROR	30046	Wrong robot mode: {}	suggest..
POSE_OUT_OF_REACH	30047	Target pose [{}] out of reach	suggest..
TP_PLAN_FAILED	30048	Trajectory plan FAILED.	suggest..
START_FORCE_FAILED	30049	Start force control failed, because force sensor does not exist.	suggest..
OVER_SAFE_PLANE_LIMIT	30050	{} axis exceeds the safety plane limit (Move_type:{} id:{}).	Please move robot to the s plane rang
POWERON_FAIL_VIOLATION	30051	Failed to power on because the robot safety mode is in violation	suggest..
POWERON_FAIL_SYSTEMEMERGENCYSTOP	30052	Failed to power on because the robot safety mode is in	suggest..

		system emergency stop	
POWERON_FAIL_ROBOTEMERGENCYSTOP	30053	Failed to power on because the robot safety mode is in robot emergency stop	Pop up the emergency stop button on the pendant when the robot is in a range of motion
POWERON_FAIL_FAULT	30054	Failed to power on because the robot safety mode is in fault	suggest..
STARTUP_FAIL_VIOLATION	30055	Failed to startup because the robot safety mode is in violation	suggest..
STARTUP_FAIL_SYSTEMEMERGENCYSTOP	30056	Failed to startup because the robot safety mode is in system emergency stop	suggest..
STARTUP_FAIL_ROBOTEMERGENCYSTOP	30057	Failed to startup because the robot safety mode is in robot emergency stop	Pop up the emergency stop button on the pendant when the robot is in a range of motion
STARTUP_FAIL_FAULT	30058	Failed to startup because the robot safety mode is in fault	suggest..
BACKDRIVE_FAIL_VIOLATION	30059	Failed to backdrive because the robot safety mode is in violation	suggest..
BACKDRIVE_FAIL_SYSTEMEMERGENCYSTOP	30060	Failed to backdrive	suggest..

		because the robot safety mode is in system emergency stop	
BACKDRIVE_FAIL_ROBOTEMERGENCYSTOP	30061	Failed to backdrive because the robot safety mode is in robot emergency stop	Pop up the emergency stop button on the pendant when the robot is in a range of motion
BACKDRIVE_FAIL_FAULT	30062	Failed to backdrive because the robot safety mode is in fault	suggest..
SETSIM_FAIL_VIOLATION	30063	Switch sim mode failed because the robot safety mode is in violation	suggest..
SETSIM_FAIL_SYSTEMEMERGENCYSTOP	30064	Switch sim mode failed because the robot safety mode is in system emergency stop	suggest..
SETSIM_FAIL_ROBOTEMERGENCYSTOP	30065	Switch sim mode failed because the robot safety mode is in robot emergency stop	Pop up the emergency stop button on the pendant when the robot is in a range of motion
SETSIM_FAIL_FAULT	30066	Switch sim mode failed because the robot safety mode is in fault	suggest..
FREEDRIVE_FAIL_VIOLATION	30067	Enable handguide	suggest..

		mode failed because the robot safety mode is in violation	
FREEDRIVE_FAIL_SYSTEMEMERGENCYSTOP	30068	Enable handguide mode failed because the robot safety mode is in system emergency stop	suggest..
FREEDRIVE_FAIL_ROBOTEMERGENCYSTOP	30069	Enable handguide mode failed because the robot safety mode is in robot emergency stop	Pop up the emergency stop button on the pendant when robot is in a range of motion
FREEDRIVE_FAIL_FAULT	30070	Enable handguide mode failed because the robot safety mode is in fault	suggest..
UPFIRMWARE_FAIL_VIOLATION	30071	Firmware update failed because the robot safety mode is in violation	suggest..
UPFIRMWARE_FAIL_SYSTEMEMERGENCYSTOP	30072	Firmware update failed because the robot safety mode is in system emergency stop	suggest..
UPFIRMWARE_FAIL_ROBOTEMERGENCYSTOP	30073	Firmware update failed because the	Pop up the emergency stop button on the

		robot safety mode is in robot emergency stop	pendant when robot is in a range of mo
UPFIRMWARE_FAIL_FAULT	30074	Firmware update failed because the robot safety mode is in fault	suggest..
SETPERSOSTENT_FAIL_VIOLATION	30075	Set persistent parameter failed because the robot safety mode is in violation	suggest..
SETPERSOSTENT_FAIL_SYSTEMEMERGENCYSTOP	30076	Set persistent parameter failed because the robot safety mode is in system emergency stop	suggest..
SETPERSOSTENT_FAIL_ROBOTEMERGENCYSTOP	30077	Set persistent parameter failed because the robot safety mode is in robot emergency stop	Pop up the emergency stop button on the pendant when robot is in a range of mo
SETPERSOSTENT_FAIL_FAULT	30078	Set persistent parameter failed because the robot safety mode is in fault	suggest..
SETPERSOSTENT_FAIL_PARAM_ERR	30079	Set persistent parameter failed	(a) Check the parameter for whether all floating point numbers
ROBOT_CABLE_DISCONN	30080	Robot cable not connected	(a) Make sure cable between Control Box and Robot Arm is correctly connected and

			no damage. (b) for loose connection (c) Do a Complete rebooting sequence (d) Update software (e) Contact your AUBO Robots service provider for assistance. Contact your local AUBO Robots service provider for assistance.
TP_TOO_SHORT	30081	The generated trajectory is ignored because it is too short	(a) Please check added waypoints coincident (b) an arc mover please check whether the three points are collinear
INV_KIN_FAIL	30082	Inverse kinematics solution failed. The target pose may be in a singular position or exceed the joint limits	(a) Change the pose and try again
FREEDRIVE_ENABLED	30083	Freedrive status changed to {}	suggest..
TP_INV_FAIL_REFERENCE_JOINT_OUT_OF_LIMIT	30084	Inverse kinematics solution failed. Reference angle [{}] exceeds joint limit [{}].	suggest..
TP_INV_FAIL_NO_SOLUTION	30085	Inverse kinematics solution failed. The reference angle [{}] and the target angle	suggest..

		[{}] are used as parameters. there is no solution in the calculation of the inverse solution process.	
SERVO_FAIL_VIOLATION	30086	Switch servo mode failed because the robot safety mode is in violation	suggest..
SERVO_FAIL_SYSTEMEMERGENCYSTOP	30087	Switch servo mode failed because the robot safety mode is in system emergency stop	suggest..
SERVO_FAIL_ROBOTEMERGENCYSTOP	30088	Switch servo mode failed because the robot safety mode is in robot emergency stop	Pop up the emergency stop button on the pendant when the robot is in a range of motion
SERVO_FAIL_FAULT	30089	Switch servo mode failed because the robot safety mode is in fault	suggest..
FREEDRIVE_FAIL_NO_RUNNING	30090	Enable handguide mode failed because the robot mode type is {}(not running)	suggest..
RUNTIME_MACHINE_ERROR	30091	The state of the running machine is {},	suggest..

		not {}. {} function execution failed because the state is wrong.	
RESUME_FAR_PAUSE_PT	30092	Cannot resume from joint position [{}].\nToo far away from paused point [{}].	suggest..
PAYLOAD_LIGHTER_ERROR	30093	The payload setting is too small!	suggest..
PAYLOAD_OVERLOAD_ERROR	30094	The payload setting is too large!	suggest..
PAUSE_FAIL_NOT_POSITION_PLAN_MODE	30095	This motion does not support the pause function. The motion is stopping.	suggest..
TP_PLAN_FAILED_CIRCULAR_WAYPOINTS_COINCIDE	30096	The planning failed because the three waypoints of the arc were determined to coincide.	Check the cir waypoints to sure they a different
SERVO_WRONG_SAFETYMODE	30097	Switch servo mode failed because the robot safety mode is in {}.	Check the cir waypoints to sure they a different
SET_PERSTPARAM_WRONG_SAFETYMODE	30098	Set persistent parameter failed because the robot safety mode is in {}	suggest..

SET_KINPARAM_WRONG_SAFETYMODE	30099	Set Kinematics Compensate parameters failed because the robot safety mode is in {}	suggest..
SET_ROBOT_ZERO_WRONG_SAFETYMODE	30100	Set current joint angles to zero failed because the robot safety mode is in {}	suggest..
UPFIRMWARE_WRONG_SAFETYMODE	30101	Firmware update failed because the robot safety mode is in {}	suggest..
POWERON_WRONG_SAFETYMODE	30102	Failed to power on because the robot safety mode is in {}	suggest..
STARTUP_WRONG_SAFETYMODE	30103	Failed to startup because the robot safety mode is in {}	suggest..
BACKDRIVE_WRONG_SAFETYMODE	30104	Failed to backdrive because the robot safety mode is in system emergency stop	suggest..
SETSIM_WRONG_SAFETYMODE	30105	Switch sim mode failed because the robot safety mode is in violation	suggest..
FREEDRIVE_WRONG_SAFETYMODE	30106	Enable handguide mode failed because the	suggest..

		robot safety mode is in wrong safety mode: {}	
TP_PLAN_FAILED_JOINT_JUMP_BIGGER	30107	Inverse kinematics solution failed. The target point and the current point are in different robot configuration spaces.	Add a few more points between target point and current point
RUN_PROGRAM_FAILED	30108	Run program {} failed.	suggest..
FREEDRIVE_FAIL_WRONG_RTMSTATE	30109	Unable to enter the HandGuide mode as the robot is not currently in a stopped or paused state.	suggest..
SAFEGUARDSTOP_CONFIGURABLE_INPUT	30110	Configurable safety input is triggered.	suggest..
SAFEGUARDSTOP_3PE	30111	3PE is triggered.	suggest..
SAFEGUARDSTOP_SI	30112	SI0/SI1 is triggered.	suggest..
ROBOT_TYPE_CHANGED	30200	Robot type changed to '{}', and robot subtype changed to '{}'	suggest..
LINKMODE_CHANGED	30201	Link mode changed to {}	suggest..
ROBOT_SELF_COLLISION	30301	Detect risk of robot self collision	suggest..

CONSTANT_INVALID	30302	Joint torque constants are invalid. HandGuide will be disabled, and the collision protection may be triggered by mistake.	suggest..
GRAVITY_INVALID	30303	Abnormal value of gravity acceleration sensor. HandGuide will be disabled, and the collision protection may be triggered by mistake.	suggest..
DYNAMICS_INVALID	30304	Robot dynamics parameters are invalid. HandGuide will be disabled, and the collision protection may be triggered by mistake.	suggest..
FRICTION_INVALID	30305	Joint friction parameters are invalid. HandGuide will be disabled, and the collision protection may be triggered by mistake.	suggest..
HANDGUIDE_UNDER_DEVELOP	30306	Robot type of {} function under development.	suggest..

		HandGuide will be disabled, and the collision protection may be triggered by mistake.	
SLOW_DOWN_INFO	30307	Slow down level changed to {} ({}%)	suggest..
WRONG_JOINT_DESIGNED_LIMIT	30308	Joint designed ranges exceeds ranges read from hardware interface.	suggest..
FREEDRIVE_IN_SIMULATION	30309	Enable handguide mode failed because the robot is in simulation mode.	suggest..
ROBOT_STOPPING_TIMEOUT	30310	Robot stopping timeout.	suggest..
PSTOP_INCORRECT_FORCE_OFFSET	30311	Protective Stop: Sudden change in force control target position. Force sensor offset may be incorrect or force sensor fault.	suggest..
WRONG_JOINT_SAFETY_LIMIT	30312	Joint safety ranges exceeds designed ranges.	suggest..
PSTOP_TCP_PLANE_VIOLATION	30401	Protective Stop: TCP position close to safety plane limits.	suggest..

PSTOP_ELBOW_PLANE_VIOLATION	30402	Protective Stop: elbow position close to safety plane limits.	suggest..
PSTOP_JOINT_TORQUE_VIOLATION	30403	Protective Stop: joint{} exceeds torque limit.	suggest..
PSTOP_JOINT_POSITION_VIOLATION	30404	Protective Stop: joint{} exceeds position limit.	suggest..
PSTOP_JOINT_SPEED_VIOLATION	30405	Protective Stop: joint{} exceeds speed limit.	suggest..
PSTOP_TCP_SPEED_VIOLATION	30406	Protective Stop: TCP speed close to safety limits.	suggest..
PSTOP_ELBOW_SPEED_VIOLATION	30407	Protective Stop: elbow speed close to safety limits.	suggest..
PSTOP_TCP_FORCE_VIOLATION	30408	Protective Stop: TCP force close to safety limits.	suggest..
PSTOP_ELBOW_TORQUE_VIOLATION	30409	Protective Stop: elbow torque close to safety limits.	suggest..
PSTOP_POWER_VIOLATION	30410	Protective Stop: robot power close to safety limits.	suggest..
PSTOP_MOMENTUM_VIOLATION	30411	Protective Stop: robot momentum close to safety limits.	suggest..
PSTOP_TCP_CUBE_VIOLATION	30412	Protective Stop: TCP position	suggest..

		close to safety cube.	
PSTOP_ELBOW_CUBE_VIOLATION	30413	Protective Stop: TCP position close to safety cube.	suggest..
REDUCE_ELBOW_PLANE_TRIGGER	30414	Reduce mode: elbow close to safety plane triggers reduction mode.	suggest..
REDUCE_TCP_PLANE_TRIGGER	30415	Reduce mode: TCP close to safety plane triggers reduction mode.	suggest..
PSTOP_MOVE_OUT_RANGE	30416	Joint {} has exceeded the limit, please do not continue to move out of the range	suggest..
RESUME_PAUSE_FAILED	30417	Resume Failed: Safety mode type is {}	suggest..
FIRMWARE_UPDATE_FAIL_EMERGENCYSTOP	30418	Failed to firmware update because the robot safety mode is in {}	Release emergency stop when the is in a safe rare motion
TOOL_SENSOR_CHANGED	30419	Tool sensor type changed to {}	suggest..
TOOL_SENSOR_REMOVED	30420	Tool sensor is removed.	suggest..
CAL_TARGET_CURRENT_ERR	30421	The calculation of the target current failed.	suggest..

		Please try again later.	
CONVEYOR_MODE_CHANGED	30422	Conveyor{}: track mode changed to {}, track item id is {}	suggest..
CONVEYOR_ENQUEUE	30423	Conveyor{}: the queue has been changed, item{} is enqueue	suggest..
CONVEYOR_DEQUEUE_FINISH	30424	Conveyor{}: the queue has been changed, item{} dequeue due to track finished	suggest..
CONVEYOR_DEQUEUE_STARTWINDOW	30425	Conveyor{}: the queue has been changed, item{} dequeue due to exceeds startwindow	suggest..
CONVEYOR_DEQUEUE_LIMIT	30426	Conveyor{}: the queue has been changed, item{} dequeue due to exceed limit area	suggest..
CONVEYOR_DEQUEUE_CLEAR	30427	Conveyor{}: item queue is cleared	suggest..
CONVEYOR_NEXT_TRACK	30428	Conveyor{}: item{} inside the start window that can be tracked	suggest..
CONVEYOR_EXCEED_LIMIT	30429	Conveyor{}: item{} exceeds the limit area during tracking	suggest..

WRONG_POWER_SAFETY_LIMIT	30430	Robot power safety value exceeds designed value.	suggest..
WRONG_POWER_DESIGNED_LIMIT	30431	Power designed value exceeds value read from hardware interface.	suggest..
TOOL_SENSOR_STATUS_CHANGED	30432	Tool sensor status changed to {}	suggest..
TOOL_FLASH_VERIFY_FAILED	40001	Flash write verify failed	suggest..
TOOL_PROGRAM_CRC_FAILED	40002	Program flash checksum failed during bootloading	suggest..
TOOL_PROGRAM_CRC_FAILED2	40003	Program flash checksum failed at runtime	suggest..
TOOL_ID_UNDEFINED	40004	Tool ID is undefined	suggest..
TOOL_ILLEGAL_BL_CMD	40005	Illegal bootloader command	suggest..
TOOL_FW_WRONG	40006	Wrong firmware at the joint	suggest..
TOOL_HW_INVALID	40007	Invalid hardware revision	suggest..
TOOL_SHORT_CURCUIT_H	40011	Short circuit detected on Digital Output: {} high side	suggest..
TOOL_SHORT_CURCUIT_L	40012	Short circuit detected on	suggest..

		Digital Output: { } low side	
TOOL_AVERAGE_CURR_HIGH	40013	10 second Average tool IO Current of { } A is outside of the allowed range.	suggest..
TOOL_POWER_PIN_OVER_CURR	40014	Current of { } A on the POWER pin is outside of the allowed range.	suggest..
TOOL_DOUT_PIN_OVER_CURR	40015	Current of { } A on the Digital Output pins is outside of the allowed range.	suggest..
TOOL_GROUND_PIN_OVER_CURR	40016	Current of { } A on the ground pin is outside of the allowed range.	suggest..
TOOL_RX_FRAMING	40021	RX framing error	suggest..
TOOL_RX_PARITY	40022	RX Parity error	suggest..
TOOL_48V_LOW	40031	48V input is too low	suggest..
TOOL_48V_HIGH	40032	48V input is too high	suggest..
TOOL_ERR_OFFLINE	40033	tool error: tool may be offline	(a) Check to hardware. (b) joint's id
TOOL_ERR_BOOTLOADER	40034	tool error: The tool is in bootloader mode. Retry firmware update.	suggest..

PKG_LOST	50001	Lost package from pedestal	suggest..
PEDSTRAL_OFFLINE	50002	pedestal error: pedestal may be offline	(a) Check pedestal hardware. (b) pedestal's
PEDESTAL_ERR_BOOTLOADER	50003	pedestal error: The pedestal is in bootloader mode. Retry firmware update.	suggest..
HW_SCB_SETUP_FAILED	60001	Setup of Interface Board failed	suggest..
HW_PKG_CNT_DISAGREE	60002	Packet counter disagreements	suggest..
HW_SCB_DISCONNECT	60003	Connection to Interface Board lost	suggest..
HW_SCB_PKG_LOST	60004	Package lost from Interface Board	suggest..
HW_SCB_CONN_INIT_FAILED	60005	Ethernet connection initialization with Interface Board failed	suggest..
HW_LOST_JOINT_PKG	60006	Lost package from joint {}	suggest..
HW_LOST_TOOL_PKG	60007	Lost package from tool	suggest..
HW_JOINT_PKG_CNT_DISAGREE	60008	Packet counter disagreement in packet from joint {}	suggest..
HW_TOOL_PKG_CNT_DISAGREE	60009	Packet counter disagreement in	suggest..

		packet from tool	
HW_JOINTS_FAULT	60011	{ } joint entered the Fault State	suggest..
HW_JOINTS_VIOLATION	60012	{ } joint entered the Violation State	suggest..
HW_TP_FAULT	60013	Teach Pendant entered the Fault State	suggest..
HW_TP_VIOLATION	60014	Teach Pendant entered the Violation State	suggest..
HW_JOINT_MV_TOO_FAR	60021	{ } joint moved too far before robot entered RUNNING State	suggest..
HW_JOINT_STOP_NOT_FAST	60022	Joint Not stopping fast enough	suggest..
HW_JOINT_MV_LIMIT	60023	Joint moved more than allowable limit	suggest..
HW_FT_SENSOR_DATA_INVALID	60024	Force-Torque Sensor data invalid	suggest..
HW_NO_FT_SENSOR	60025	Force-Torque sensor is expected, but it cannot be detected	suggest..
HW_FT_SENSOR_NOT_CALIB	60026	Force-Torque sensor is detected but not calibrated	suggest..
HW_RELEASE_BRAKE_FAILED	60030	Robot was not able to brake	suggest..

		release, see log for details	
HW_OVERCURR_SHUTDOWN	60040	Overcurrent shutdown	suggest..
HW_ENERGEY_SURPLUS	60050	Energy surplus shutdown	suggest..
HW_IDLE_POWER_HIGH	60060	Idle power consumption to high	suggest..
HW_ENTER_COLLISION_TIMEOUT	60071	Enter collision stop procedure timeout	suggest..
HW_POWERON_TIMEOUT	60072	Poweron robot timeout	suggest..
HW_NO_NIC_FOUND	60073	No network cards found.	suggest..
HW_IFB_NOT_FOUND	60074	No Interface Board found.	suggest..
HW_IFB_BOOTLOAD	60075	The Interface Board is in bootloader mode. Update firmware firstly.	suggest..
HW_TOOL_NOT_FOUND	60076	No Tool Board found.	suggest..
HW_BASE_NOT_FOUND	60077	No Base Board found.	suggest..
HW_BRINGUP_TIMEOUT	60078	Poweron robot timeout	suggest..
HW_COLLISION_RECOVERY_FAILED	60079	Collision recovery failed	suggest..
ARCS_MAX_ERROR_CODE	-1	Max error code	suggest..

RTDE Menu

[Input Menu](#) is the data for publishing, that is, the menu to push data to the server from the client. [Output Menu](#) is the data for subscribing, that is, the menu to push data to the client from the server.

R1 represents interaction with robot1, R2 represents interaction with robot2, and so on. Currently, only R1 is supported.

[Input Menu](#)

Name	Data Type	Description
set_recipe	RtdeRecipe	Set
input_bit_registers0_to_31	int	General purpose bits This range of the boolean input registers is reserved forFieldBus/PLC interface usage.
input_bit_registers32_to_63	int	General purpose bits This range of the boolean input registers is reserved forFieldBus/PLC interface usage.
input_bit_registers64_to_127	int64_t	64 general purpose bits X: [64..127] - The upper range of the boolean input registers canbe used by external RTDE clients (i.e AUBOCAPS).
input_int_registers_0	int	48 general purpose integer registers X: [0..23] - The lower range of the integer input registers isreserved for FieldBus/PLC interface usage. X: [24..47] - Theupper range of the integer input registers can be used byexternal RTDE clients (i.e AUBOCAPS).
input_float_registers_0	float	48 general purpose integer registers X: [0..23] - The lower range of the integer input registers isreserved for FieldBus/PLC

		interface usage. X: [24..47] - The upper range of the integer input registers can be used by external RTDE clients (i.e AUBOCAPS).
input_double_registers_0	double	48 general purpose double registers X: [0..23] - The lower range of the double input registers is reserved for FieldBus/PLC interface usage. X: [24..47] - The upper range of the double input registers can be used by external RTDE clients (i.e AUBOCAPS).
input_int16_registers_0	double	
input_int16_registers0_to_63	std::vector<double>	
R1_speed_slider_mask	double	0 = don't change speed slider with this input = use speed_slider_fraction to set speed slider value
R1_speed_slider_fraction	double	new speed slider value
R1_standard_digital_output_mask	uint32_t	Standard digital outputs
R1_configurable_digital_output_mask	uint32_t	Configurable digital outputs
R1_standard_digital_output	uint32_t	Standard digital outputs
R1_configurable_digital_output	uint32_t	Configurable digital outputs
R1_tool_digital_output	uint32_t	Tool digital outputs its 0-1: output state, remaining bits are reserved for future use
R1_standard_analog_output_type	std::vector<int>	Output domain {0=current[A], 1=voltage[V]} Bits 0-1: standard_analog_output_0 standard_analog_output_1
R1_standard_analog_output_mask	uint32_t	Standard analog output 0 (ratio) [0..1]
R1_standard_analog_output	std::vector<double>	Standard analog output 1 (ratio) [0..1]

R1_debug	uint32_t	Debug for internal use
R1_tool_digital_output_mask	uint32_t	Tool digital outputs mask
R1_rtde_input_max	int	
R2_speed_slider_mask	double	0 = don't change speed slider with this input = use speed_slider_fraction to set speed slider value
R2_speed_slider_fraction	double	new speed slider value
R2_standard_digital_output_mask	uint32_t	Standard digital outputs
R2_configurable_digital_output_mask	uint32_t	Configurable digital outputs
R2_standard_digital_output	uint32_t	Standard digital outputs
R2_configurable_digital_output	uint32_t	Configurable digital outputs
R2_tool_digital_output	uint32_t	Tool digital outputs its 0-1: output state, remaining bits are reserved for future use
R2_standard_analog_output_type	std::vector<int>	Output domain {0=current[A], 1=voltage[V]} Bits 0-1: standard_analog_output_0 standard_analog_output_1
R2_standard_analog_output_mask	uint32_t	Standard analog output 0 (ratio) [0..1]
R2_standard_analog_output	std::vector<double>	Standard analog output 1 (ratio) [0..1]
R2_debug	uint32_t	Debug for internal use
R2_tool_digital_output_mask	uint32_t	Tool digital outputs mask
R2_rtde_input_max	int	
R3_speed_slider_mask	double	0 = don't change speed slider with this input = use speed_slider_fraction to set speed slider value
R3_speed_slider_fraction	double	new speed slider value

R3_standard_digital_output_mask	uint32_t	Standard digital outputs
R3_configurable_digital_output_mask	uint32_t	Configurable digital outputs
R3_standard_digital_output	uint32_t	Standard digital outputs
R3_configurable_digital_output	uint32_t	Configurable digital outputs
R3_tool_digital_output	uint32_t	Tool digital outputs its 0-1: output state, remaining bits are reserved for future use
R3_standard_analog_output_type	std::vector<int>	Output domain {0=current[A], 1=voltage[V]} Bits 0-1: standard_analog_output_0 standard_analog_output_1
R3_standard_analog_output_mask	uint32_t	Standard analog output 0 (ratio) [0..1]
R3_standard_analog_output	std::vector<double>	Standard analog output 1 (ratio) [0..1]
R3_debug	uint32_t	Debug for internal use
R3_tool_digital_output_mask	uint32_t	Tool digital outputs mask
R3_rtde_input_max	int	
R4_speed_slider_mask	double	0 = don't change speed slider with this input = use speed_slider_fraction to set speed slider value
R4_speed_slider_fraction	double	new speed slider value
R4_standard_digital_output_mask	uint32_t	Standard digital outputs
R4_configurable_digital_output_mask	uint32_t	Configurable digital outputs
R4_standard_digital_output	uint32_t	Standard digital outputs
R4_configurable_digital_output	uint32_t	Configurable digital outputs
R4_tool_digital_output	uint32_t	Tool digital outputs its 0-1: output state, remaining bits are reserved for future use

R4_standard_analog_output_type	std::vector<int>	Output domain {0=current[A], 1=voltage[V]} Bits 0-1: standard_analog_output_0 standard_analog_output_1
R4_standard_analog_output_mask	uint32_t	Standard analog output 0 (ratio) [0..1]
R4_standard_analog_output	std::vector<double>	Standard analog output 1 (ratio) [0..1]
R4_debug	uint32_t	Debug for internal use
R4_tool_digital_output_mask	uint32_t	Tool digital outputs mask
R4_rtde_input_max	int	

Output Menu

Name	Data Type	Description
timestamp	double	Time elapsed since the controller was started [s]
line_number	int	line number set by setPlanContext
runtime_state	RuntimeState	Program state
output_bit_registers_0_to_63	int64_t	64 [000..063] General purpose bits
output_bit_registers_64_to_127	int64_t	64 [064..127] general purpose bits
output_int_registers_0	int	48 general purpose integer registersX: [0..23] - The lower range of the integer output registers is reserved for FieldBus/PLC interface usage. X: [24..47] - The upper range of the integer output registers can be used by external

		RTDE clients (i.e AUBOCAPS).
output_float_registers_0	int	48 general purpose integer registersX: [0..23] - The lower range of the integer output registers isreserved for FieldBus/PLC interface usage. X: [24..47] - The upper range of the integer output registers can be used by external RTDE clients (i.e AUBOCAPS).
output_double_registers_0	double	48 general purpose double registersX: [0..23] - The lower range of the double output registers isreserved for FieldBus/PLC interface usage. X: [24..47] - The upper range of the double output registers can be used by external RTDE clients (i.e AUBOCAPS).
input_bit_registers_r0_to_63	int64_t	[0..63] General purpose bits This range of the boolean outputregisters is reserved for FieldBus/PLC interface usage.
input_bit_registers_r64_to_127	int64_t	64 [64..127] general purpose bits
input_int_registers_r0	int	(([0 .. 48]) 48 general purpose integer registersX: [0..23] - The lower range of the integer input registers isreserved for FieldBus/PLC interface usage. X: [24..47] - Theupper range of the

		integer input registers can be used by external RTDE clients (i.e. AUBOCAPS).
input_float_registers_r0	int	([0 .. 48]) 48 general purpose integer registers X: [0..23] - The lower range of the integer input registers is reserved for FieldBus/PLC interface usage. X: [24..47] - The upper range of the integer input registers can be used by external RTDE clients (i.e. AUBOCAPS).
input_double_registers_r0	double	([0 .. 48]) 48 general purpose double registers X: [0..23] - The lower range of the double input registers is reserved for FieldBus/PLC interface usage. X: [24..47] - The upper range of the double input registers can be used by external RTDE clients (i.e. AUBOCAPS).
modbus_signals	std::vector<int>	Modbus signals from connected modbus slaves
modbus_signals_errors	std::vector<int>	Modbus signals request status from connected modbus slaves
input_int16_registers_r0	int16_t	input_int16_registers_r0
input_int16_registers_0_to_63	std::vector<int16_t>	
runtime_context	std::vector<int>	

R1_message	RobotMsg	Robot message from controller
R1_target_q	std::vector<double>	Target joint positions
R1_target_qd	std::vector<double>	Target joint velocities
R1_target_qdd	std::vector<double>	Target joint accelerations
R1_target_current	std::vector<double>	Target joint currents
R1_target_moment	std::vector<double>	Target joint moments (torques)
R1_actual_q	std::vector<double>	Actual joint positions
R1_actual_qd	std::vector<double>	Actual joint velocities
R1_actual_current	std::vector<double>	Actual joint currents
R1_joint_control_output	std::vector<double>	Joint control currents
R1_joint_temperatures	std::vector<double>	Temperature of each joint in degrees Celsius
R1_actual_joint_voltage	std::vector<double>	Actual joint voltages
R1_joint_mode	std::vector<JointStateType>	Joint control modes Please see Remote Control Via TCP/IP - 16496
R1_actual_execution_time	double	Controller real-time thread execution time
R1_robot_mode	RobotModeType	Robot mode Please see Remote Control Via TCP/IP - 16496
R1_safety_mode	SafetyModeType	Safety mode Please see Remote Control Via TCP/IP - 16496
R1_safety_status	unknown	Safety status
R1_robot_status_bits	unknown	Bits 0-3 0: Is power on 1: Is program running 2: Is teach button

		pressed 3: Is power button pressed
R1_safety_status_bits	unknown	Bits 0-10 0: Is normal mode 1: Is reduced mode 2: Is protectivestopped 3: Is recovery mode 4: Is safeguard stopped 5: Is systememergency stopped 6: Is robot emergency stopped 7: Is emergencystopped 8: Is violation 9: Is fault 10: Is stopped due to safety
R1_speed_scaling	double	Speed scaling of the trajectory limiter
R1_target_speed_fraction	double	Target speed fraction
R1_actual_TCP_pose	std::vector<double>	Actual Cartesian coordinates of the tool: (x,y,z,rx,ry,rz), where rx, ry and rz is a rotationvector representation of the tool orientation
R1_actual_TCP_speed	std::vector<double>	Actual speed of the tool given in Cartesian coordinates
R1_actual_TCP_force	std::vector<double>	Generalized forces in the TCP
R1_target_TCP_pose	std::vector<double>	Target Cartesian coordinates of the tool: (x,y,z,rx,ry,rz), where rx, ry and rz is a rotationvector representation of the tool orientation

R1_target_TCP_speed	std::vector<double>	Target speed of the tool given in Cartesian coordinates
R1_elbow_position	std::vector<double>	Position of robot elbow in Cartesian Base Coordinates
R1_elbow_velocity	std::vector<double>	Velocity of robot elbow in Cartesian Base Coordinates
R1_actual_momentum	std::vector<double>	Norm of Cartesian linear momentum
R1_tcp_force_scalar	std::vector<double>	TCP force scalar [N]
R1_future_path_points	std::vector<std::vector<double>>	Get future path points joint
R1_actual_main_voltage	unknown	Safety Control Board: Main voltage
R1_actual_robot_voltage	unknown	Safety Control Board: Robot voltage (48V)
R1_actual_robot_current	unknown	Safety Control Board: Robot current
R1_joint_torque_sensor	std::vector<double>	Joint torque sensor
R1_operationalModeSelectorInput	OperationalModeType	Current state of the operational mode selector input
R1_threePositionEnablingDeviceInput	unknown	
R1_masterboard_temperature	unknown	
R1_standard_digital_input_bits	uint64_t	Current state of the standard digital inputs.
R1_tool_digital_input_bits	uint64_t	Current state of the tool digital inputs and outputs.
R1_configurable_digital_input_bits	uint64_t	Current state of the safety inputs.

R1_link_digital_input_bits	uint64_t	Current state of the link digital inputs.
R1_standard_digital_output_bits	uint64_t	Current state of the standard digital outputs.
R1_tool_digital_output_bits	uint64_t	Current state of the tool digital inputs and outputs.
R1_configurable_digital_output_bits	uint64_t	Current state of the safety outputs.
R1_link_digital_output_bits	uint64_t	Current state of the link digital outputs.
R1_standard_analog_input_values	std::vector<double>	Current values of the standard analog inputs.
R1_tool_analog_input_values	std::vector<double>	Current values of the tool analog inputs.
R1_standard_analog_output_values	std::vector<double>	Current values of the standard analog outputs.
R1_tool_analog_output_values	std::vector<double>	Current values of the tool analog outputs.
R1_is_simulation_enabled	bool	
R1_collision_level	int	
R1_master_io_current	unknown	I/O current [A]
R1_euromap67_input_bits	unknown	Euromap67 input bits
R1_euromap67_output_bits	unknown	Euromap67 output bits
R1_euromap67_24V_voltage	unknown	Euromap 24V voltage [V]
R1_euromap67_24V_current	unknown	Euromap 24V current [A]
R1_tool_mode	unknown	Tool mode Please see Remote Control Via TCP/IP - 16496
R1_tool_output_mode	unknown	The current output mode

R1_tool_output_voltage	unknown	Tool output voltage [V]
R1_tool_output_current	unknown	Tool current [A]
R1_tool_voltage_48V	unknown	
R1_tool_current	unknown	
R1_tool_temperature	unknown	Tool temperature in degrees Celsius
R1_actual_tool_accelerometer	unknown	Tool x, y and z accelerometer values
R1_motion_progress	unknown	Trajectory running progress
R1_actual_qdd	unknown	Actual joint accelerations
R1_controlbox_humidity	double	Controbox humidity
R1_actual_tool_pose	std::vector<double>	Actual Cartesian coordinates of the tool(without TCP)
R1_rtde_output_max	int	
R1_actual_TCP_force_sensor	std::vector<double>	TCP force sensor
R1_fc_cond_fullfiled	bool	
R1_actual_payload	Payload	Actual payload
R1_tool_button_status	bool	Tool button status
R2_message	RobotMsg	Robot message from controller
R2_target_q	std::vector<double>	Target joint positions
R2_target_qd	std::vector<double>	Target joint velocities
R2_target_qdd	std::vector<double>	Target joint accelerations
R2_target_current	std::vector<double>	Target joint currents
R2_target_moment	std::vector<double>	Target joint moments (torques)

R2_actual_q	std::vector<double>	Actual joint positions
R2_actual_qd	std::vector<double>	Actual joint velocities
R2_actual_current	std::vector<double>	Actual joint currents
R2_joint_control_output	std::vector<double>	Joint control currents
R2_joint_temperatures	std::vector<double>	Temperature of each joint in degrees Celsius
R2_actual_joint_voltage	std::vector<double>	Actual joint voltages
R2_joint_mode	std::vector<JointStateType>	Joint control modes Please see Remote Control Via TCP/IP - 16496
R2_actual_execution_time	double	Controller real-time thread execution time
R2_robot_mode	RobotModeType	Robot mode Please see Remote Control Via TCP/IP - 16496
R2_safety_mode	SafetyModeType	Safety mode Please see Remote Control Via TCP/IP - 16496
R2_safety_status	unknown	Safety ststus
R2_robot_status_bits	unknown	Bits 0-3 0: Is power on 1: Is program running 2: Isteach button pressed 3: Is power button pressed
R2_safety_status_bits	unknown	Bits 0-10 0: Is normal mode 1: Is reduced mode 2: Is protectivestopped 3: Is recovery mode 4: Is safeguard stopped 5: Is systememergency stopped 6: Is robot emergency stopped

		7: Is emergency stopped 8: Is violation 9: Is fault 10: Is stopped due to safety
R2_speed_scaling	double	Speed scaling of the trajectory limiter
R2_target_speed_fraction	double	Target speed fraction
R2_actual_TCP_pose	std::vector<double>	Actual Cartesian coordinates of the tool: (x,y,z,rx,ry,rz), where rx, ry and rz is a rotation vector representation of the tool orientation
R2_actual_TCP_speed	std::vector<double>	Actual speed of the tool given in Cartesian coordinates
R2_actual_TCP_force	std::vector<double>	Generalized forces in the TCP
R2_target_TCP_pose	std::vector<double>	Target Cartesian coordinates of the tool: (x,y,z,rx,ry,rz), where rx, ry and rz is a rotation vector representation of the tool orientation
R2_target_TCP_speed	std::vector<double>	Target speed of the tool given in Cartesian coordinates
R2_elbow_position	std::vector<double>	Position of robot elbow in Cartesian Base Coordinates
R2_elbow_velocity	std::vector<double>	Velocity of robot elbow in Cartesian Base Coordinates
R2_actual_momentum	std::vector<double>	Norm of Cartesian linear momentum

R2_tcp_force_scalar	std::vector<double>	TCP force scalar [N]
R2_future_path_points	std::vector<std::vector<double>>	Get future path points joint
R2_actual_main_voltage	unknown	Safety Control Board: Main voltage
R2_actual_robot_voltage	unknown	Safety Control Board: Robot voltage (48V)
R2_actual_robot_current	unknown	Safety Control Board: Robot current
R2_joint_torque_sensor	std::vector<double>	Joint torque sensor
R2_operationalModeSelectorInput	OperationalModeType	Current state of the operational mode selector input
R2_threePositionEnablingDeviceInput	unknown	
R2_masterboard_temperature	unknown	
R2_standard_digital_input_bits	uint64_t	Current state of the standard digital inputs.
R2_tool_digital_input_bits	uint64_t	Current state of the tool digital inputs and outputs.
R2_configurable_digital_input_bits	uint64_t	Current state of the safety inputs.
R2_link_digital_input_bits	uint64_t	Current state of the link digital inputs.
R2_standard_digital_output_bits	uint64_t	Current state of the standard digital outputs.
R2_tool_digital_output_bits	uint64_t	Current state of the tool digital inputs and outputs.
R2_configurable_digital_output_bits	uint64_t	Current state of the safety outputs.
R2_link_digital_output_bits	uint64_t	Current state of the link digital outputs.

R2_standard_analog_input_values	std::vector<double>	Current values of the standard analog inputs.
R2_tool_analog_input_values	std::vector<double>	Current values of the tool analog inputs.
R2_standard_analog_output_values	std::vector<double>	Current values of the standard analog outputs.
R2_tool_analog_output_values	std::vector<double>	Current values of the tool analog outputs.
R2_is_simulation_enabled	bool	
R2_collision_level	int	
R2_master_io_current	unknown	I/O current [A]
R2_euromap67_input_bits	unknown	Euromap67 input bits
R2_euromap67_output_bits	unknown	Euromap67 output bits
R2_euromap67_24V_voltage	unknown	Euromap 24V voltage [V]
R2_euromap67_24V_current	unknown	Euromap 24V current [A]
R2_tool_mode	unknown	Tool mode Please see Remote Control Via TCP/IP - 16496
R2_tool_output_mode	unknown	The current output mode
R2_tool_output_voltage	unknown	Tool output voltage [V]
R2_tool_output_current	unknown	Tool current [A]
R2_tool_voltage_48V	unknown	
R2_tool_current	unknown	
R2_tool_temperature	unknown	Tool temperature in degrees Celsius
R2_actual_tool_accelerometer	unknown	Tool x, y and z accelerometer values
R2_motion_progress	unknown	Trajectory running progress

R2_actual_qdd	unknown	Actual joint accelerations
R2_controlbox_humidity	double	Controbox humidity
R2_actual_tool_pose	std::vector<double>	Actual Cartesian coordinates of the tool(without TCP)
R2_rtde_output_max	int	
R2_actual_TCP_force_sensor	std::vector<double>	TCP force sensor
R2_fc_cond_fullfiled	bool	
R2_actual_payload	Payload	Actual payload
R2_tool_button_status	bool	Tool button status
R3_message	RobotMsg	Robot message from controller
R3_target_q	std::vector<double>	Target joint positions
R3_target_qd	std::vector<double>	Target joint velocities
R3_target_qdd	std::vector<double>	Target joint accelerations
R3_target_current	std::vector<double>	Target joint currents
R3_target_moment	std::vector<double>	Target joint moments (torques)
R3_actual_q	std::vector<double>	Actual joint positions
R3_actual_qd	std::vector<double>	Actual joint velocities
R3_actual_current	std::vector<double>	Actual joint currents
R3_joint_control_output	std::vector<double>	Joint control currents
R3_joint_temperatures	std::vector<double>	Temperature of each joint in degrees Celsius
R3_actual_joint_voltage	std::vector<double>	Actual joint voltages

R3_joint_mode	std::vector<JointStateType>	Joint control modes Please see Remote Control Via TCP/IP - 16496
R3_actual_execution_time	double	Controller real-time thread execution time
R3_robot_mode	RobotModeType	Robot mode Please see Remote Control Via TCP/IP - 16496
R3_safety_mode	SafetyModeType	Safety mode Please see Remote Control Via TCP/IP - 16496
R3_safety_status	unknown	Safety status
R3_robot_status_bits	unknown	Bits 0-3 0: Is power on 1: Is program running 2: Is teach button pressed 3: Is power button pressed
R3_safety_status_bits	unknown	Bits 0-10 0: Is normal mode 1: Is reduced mode 2: Is protectivestopped 3: Is recovery mode 4: Is safeguard stopped 5: Is systememergency stopped 6: Is robot emergency stopped 7: Is emergencystopped 8: Is violation 9: Is fault 10: Is stopped due to safety
R3_speed_scaling	double	Speed scaling of the trajectory limiter
R3_target_speed_fraction	double	Target speed fraction
R3_actual_TCP_pose	std::vector<double>	Actual Cartesian coordinates of the tool:

		(x,y,z,rx,ry,rz), where rx, ry and rz is a rotationvector representation of the tool orientation
R3_actual_TCP_speed	std::vector<double>	Actual speed of the tool given in Cartesian coordinates
R3_actual_TCP_force	std::vector<double>	Generalized forces in the TCP
R3_target_TCP_pose	std::vector<double>	Target Cartesian coordinates of the tool: (x,y,z,rx,ry,rz), where rx, ry and rz is a rotationvector representation of the tool orientation
R3_target_TCP_speed	std::vector<double>	Target speed of the tool given in Cartesian coordinates
R3_elbow_position	std::vector<double>	Position of robot elbow in Cartesian Base Coordinates
R3_elbow_velocity	std::vector<double>	Velocity of robot elbow in Cartesian Base Coordinates
R3_actual_momentum	std::vector<double>	Norm of Cartesian linear momentum
R3_tcp_force_scalar	std::vector<double>	TCP force scalar [N]
R3_future_path_points	std::vector<std::vector<double>>	Get future path points joint
R3_actual_main_voltage	unknown	Safety Control Board: Main voltage
R3_actual_robot_voltage	unknown	Safety Control Board: Robot voltage (48V)
R3_actual_robot_current	unknown	Safety Control Board: Robot current

R3_joint_torque_sensor	std::vector<double>	Joint torque sensor
R3_operationalModeSelectorInput	OperationalModeType	Current state of the operational mode selector input
R3_threePositionEnablingDeviceInput	unknown	
R3_masterboard_temperature	unknown	
R3_standard_digital_input_bits	uint64_t	Current state of the standard digital inputs.
R3_tool_digital_input_bits	uint64_t	Current state of the tool digital inputs and outputs.
R3_configurable_digital_input_bits	uint64_t	Current state of the safety inputs.
R3_link_digital_input_bits	uint64_t	Current state of the link digital inputs.
R3_standard_digital_output_bits	uint64_t	Current state of the standard digital outputs.
R3_tool_digital_output_bits	uint64_t	Current state of the tool digital inputs and outputs.
R3_configurable_digital_output_bits	uint64_t	Current state of the safety outputs.
R3_link_digital_output_bits	uint64_t	Current state of the link digital outputs.
R3_standard_analog_input_values	std::vector<double>	Current values of the standard analog inputs.
R3_tool_analog_input_values	std::vector<double>	Current values of the tool analog inputs.
R3_standard_analog_output_values	std::vector<double>	Current values of the standard analog outputs.
R3_tool_analog_output_values	std::vector<double>	Current values of the tool analog outputs.

R3_is_simulation_enabled	bool	
R3_collision_level	int	
R3_master_io_current	unknown	I/O current [A]
R3_euromap67_input_bits	unknown	Euromap67 input bits
R3_euromap67_output_bits	unknown	Euromap67 output bits
R3_euromap67_24V_voltage	unknown	Euromap 24V voltage [V]
R3_euromap67_24V_current	unknown	Euromap 24V current [A]
R3_tool_mode	unknown	Tool mode Please see Remote Control Via TCP/IP - 16496
R3_tool_output_mode	unknown	The current output mode
R3_tool_output_voltage	unknown	Tool output voltage [V]
R3_tool_output_current	unknown	Tool current [A]
R3_tool_voltage_48V	unknown	
R3_tool_current	unknown	
R3_tool_temperature	unknown	Tool temperature in degrees Celsius
R3_actual_tool_accelerometer	unknown	Tool x, y and z accelerometer values
R3_motion_progress	unknown	Trajectory running progress
R3_actual_qdd	unknown	Actual joint accelerations
R3_controlbox_humidity	double	Controbox humidity
R3_actual_tool_pose	std::vector<double>	Actual Cartesian coordinates of the tool(without TCP)
R3_rtde_output_max	int	
R3_actual_TCP_force_sensor	std::vector<double>	TCP force sensor

R3_fc_cond_fullfiled	bool	
R3_actual_payload	Payload	Actual payload
R3_tool_button_status	bool	Tool button status
R4_message	RobotMsg	Robot message from controller
R4_target_q	std::vector<double>	Target joint positions
R4_target_qd	std::vector<double>	Target joint velocities
R4_target_qdd	std::vector<double>	Target joint accelerations
R4_target_current	std::vector<double>	Target joint currents
R4_target_moment	std::vector<double>	Target joint moments (torques)
R4_actual_q	std::vector<double>	Actual joint positions
R4_actual_qd	std::vector<double>	Actual joint velocities
R4_actual_current	std::vector<double>	Actual joint currents
R4_joint_control_output	std::vector<double>	Joint control currents
R4_joint_temperatures	std::vector<double>	Temperature of each joint in degrees Celsius
R4_actual_joint_voltage	std::vector<double>	Actual joint voltages
R4_joint_mode	std::vector<JointStateType>	Joint control modes Please see Remote Control Via TCP/IP - 16496
R4_actual_execution_time	double	Controller real-time thread execution time
R4_robot_mode	RobotModeType	Robot mode Please see Remote Control Via TCP/IP - 16496
R4_safety_mode	SafetyModeType	Safety mode Please see Remote Control Via TCP/IP - 16496

R4_safety_status	unknown	Safety status
R4_robot_status_bits	unknown	Bits 0-3 0: Is power on 1: Is program running 2: Is teach button pressed 3: Is power button pressed
R4_safety_status_bits	unknown	Bits 0-10 0: Is normal mode 1: Is reduced mode 2: Is protectively stopped 3: Is recovery mode 4: Is safeguard stopped 5: Is system emergency stopped 6: Is robot emergency stopped 7: Is emergency stopped 8: Is violation 9: Is fault 10: Is stopped due to safety
R4_speed_scaling	double	Speed scaling of the trajectory limiter
R4_target_speed_fraction	double	Target speed fraction
R4_actual_TCP_pose	std::vector<double>	Actual Cartesian coordinates of the tool: (x,y,z,rx,ry,rz), where rx, ry and rz is a rotation vector representation of the tool orientation
R4_actual_TCP_speed	std::vector<double>	Actual speed of the tool given in Cartesian coordinates
R4_actual_TCP_force	std::vector<double>	Generalized forces in the TCP
R4_target_TCP_pose	std::vector<double>	Target Cartesian coordinates of the tool:

		(x,y,z,rx,ry,rz), where rx, ry and rz is a rotationvector representation of the tool orientation
R4_target_TCP_speed	std::vector<double>	Target speed of the tool given in Cartesian coordinates
R4_elbow_position	std::vector<double>	Position of robot elbow in Cartesian Base Coordinates
R4_elbow_velocity	std::vector<double>	Velocity of robot elbow in Cartesian Base Coordinates
R4_actual_momentum	std::vector<double>	Norm of Cartesian linear momentum
R4_tcp_force_scalar	std::vector<double>	TCP force scalar [N]
R4_future_path_points	std::vector<std::vector<double>>	Get future path points joint
R4_actual_main_voltage	unknown	Safety Control Board: Main voltage
R4_actual_robot_voltage	unknown	Safety Control Board: Robot voltage (48V)
R4_actual_robot_current	unknown	Safety Control Board: Robot current
R4_joint_torque_sensor	std::vector<double>	Joint torque sensor
R4_operationalModeSelectorInput	OperationalModeType	Current state of the operational mode selector input
R4_threePositionEnablingDeviceInput	unknown	
R4_masterboard_temperature	unknown	
R4_standard_digital_input_bits	uint64_t	Current state of the standard digital inputs.
R4_tool_digital_input_bits	uint64_t	Current state of the tool digital inputs and

		outputs.
R4_configurable_digital_input_bits	uint64_t	Current state of the safety inputs.
R4_link_digital_input_bits	uint64_t	Current state of the link digital inputs.
R4_standard_digital_output_bits	uint64_t	Current state of the standard digital outputs.
R4_tool_digital_output_bits	uint64_t	Current state of the tool digital inputs and outputs.
R4_configurable_digital_output_bits	uint64_t	Current state of the safety outputs.
R4_link_digital_output_bits	uint64_t	Current state of the link digital outputs.
R4_standard_analog_input_values	std::vector<double>	Current values of the standard analog inputs.
R4_tool_analog_input_values	std::vector<double>	Current values of the tool analog inputs.
R4_standard_analog_output_values	std::vector<double>	Current values of the standard analog outputs.
R4_tool_analog_output_values	std::vector<double>	Current values of the tool analog outputs.
R4_is_simulation_enabled	bool	
R4_collision_level	int	
R4_master_io_current	unknown	I/O current [A]
R4_euomap67_input_bits	unknown	Euomap67 input bits
R4_euomap67_output_bits	unknown	Euomap67 output bits
R4_euomap67_24V_voltage	unknown	Euomap 24V voltage [V]
R4_euomap67_24V_current	unknown	Euomap 24V current [A]
R4_tool_mode	unknown	Tool mode Please see Remote Control Via

		TCP/IP - 16496
R4_tool_output_mode	unknown	The current output mode
R4_tool_output_voltage	unknown	Tool output voltage [V]
R4_tool_output_current	unknown	Tool current [A]
R4_tool_voltage_48V	unknown	
R4_tool_current	unknown	
R4_tool_temperature	unknown	Tool temperature in degrees Celsius
R4_actual_tool_accelerometer	unknown	Tool x, y and z accelerometer values
R4_motion_progress	unknown	Trajectory running progress
R4_actual_qdd	unknown	Actual joint accelerations
R4_controlbox_humidity	double	Controbox humidity
R4_actual_tool_pose	std::vector<double>	Actual Cartesian coordinates of the tool(without TCP)
R4_rtde_output_max	int	
R4_actual_TCP_force_sensor	std::vector<double>	TCP force sensor
R4_fc_cond_fullfiled	bool	
R4_actual_payload	Payload	Actual payload
R4_tool_button_status	bool	Tool button status