ServoSDK Manual

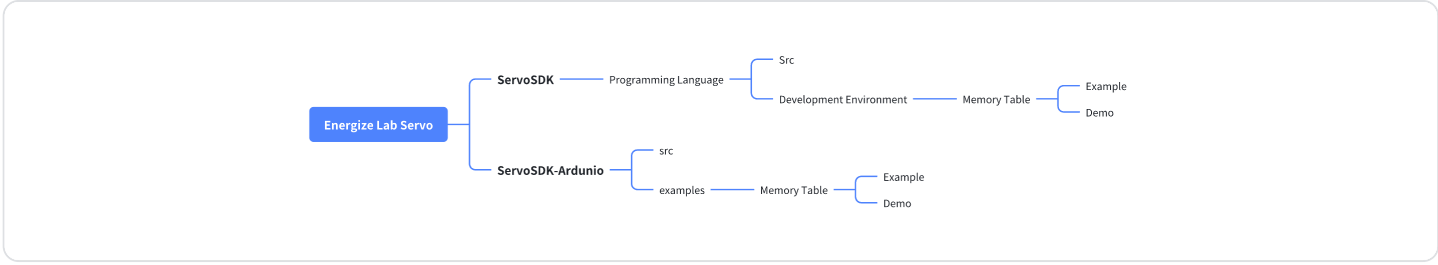
1. Introduction

Welcome to ServoSDK, an open-source software development kit designed specifically for the Energize Lab Servo. This SDK provides multi-language, multi-platform servo usage examples and motion control demos, aimed at helping users easily and quickly master the use of various models, including the EM-2030. It is strongly recommended to refer to the corresponding memory table parameter for understanding and verification.

- Supported Protocols: Energize Lab Servo Communication Protocol
- Supported Memory Tables: Primary Memory Table
- Supported Servos: EM Series, EH-3030
- Supported Programming Languages:
  - Arduino: Includes source code, with compatible examples and demos across multiple platforms.
  - C: Includes source code, with examples and demos based on Windows, STC89C52, and STM32F103C8T6.
  - C++: Includes source code, with examples and demos based on Windows.
  - MicroPython: Includes source code, with examples and demos based on ESP32.
  - Python: Includes source code, with examples and demos based on Windows.

2. Repository

2.1 File Structure



- Src/src: Source code that includes instruction generation and parsing for the entire memory table of servo.

- Example: Demonstrate the use of Ping instruction, read data instruction, write data instruction, sync write instruction, parameter reset instruction, factory reset instruction, and reboot instruction.
- Demo: Demonstrate the motion of the servo in different control modes.

2.2 Download Repository

- Method 1: Download using the Git command. This offers the most features and is suitable for developers and users who need to update code frequently.
- Method 2: Download using a browser.
  - Open with GitHub Desktop. A graphical interface, suitable for users who are not familiar with the command line.
  - Download ZIP. The simplest option, suitable for users who only need to obtain the code once.

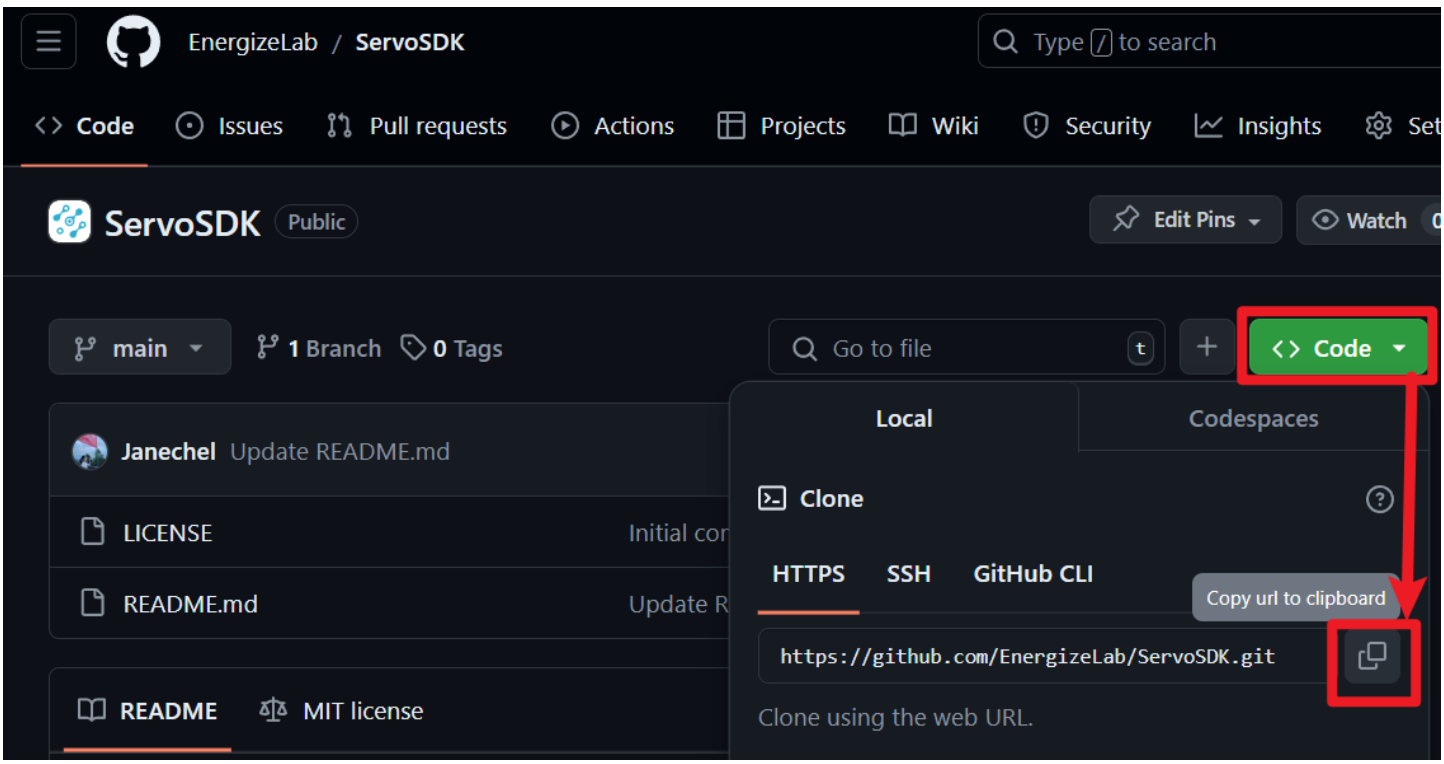
2.2.1 Use Git Command

Step 1: Install Git. Visit https://git-scm.com/downloads and select the appropriate version of Git to download and install based on your computer’ s operating system.

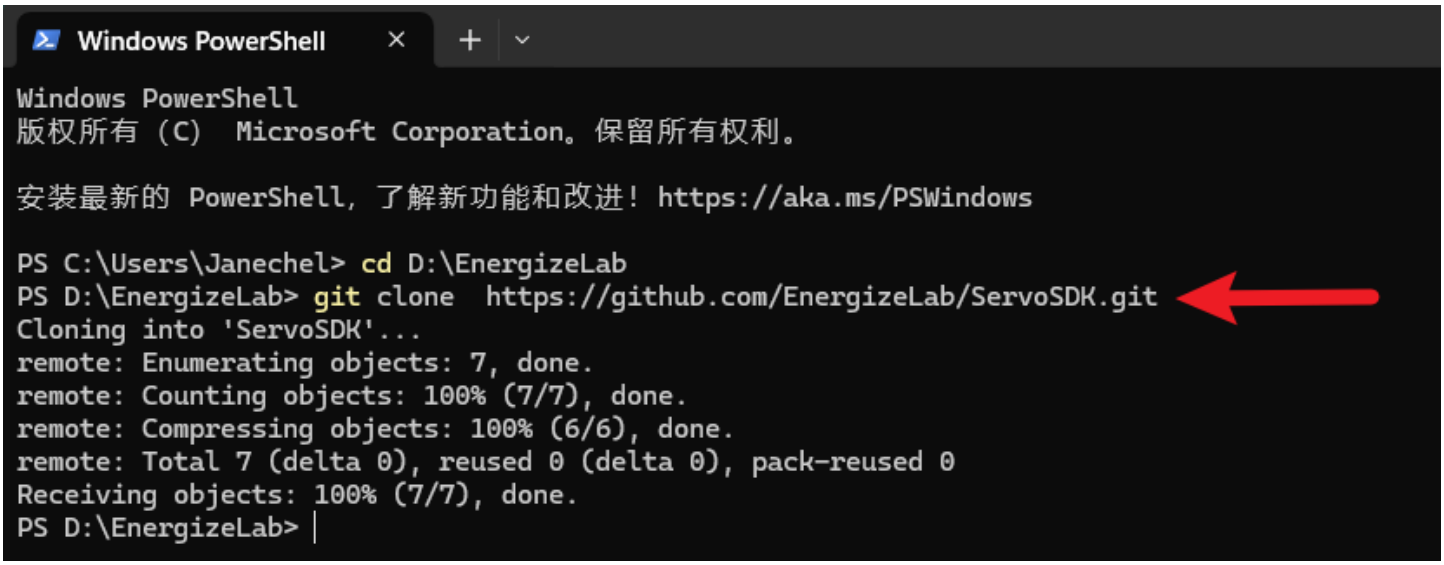
Step 2: Open the Terminal. Open the terminal on your computer and use the 'cd' command to enter the local folder directory where you want to download the repository. For example, type 'cd D:\EnergizeLab' and press Enter.

Taking Windows 11 as an example

Step 3: Clone the Repository. After entering the local directory, type 'git clone' followed by the target repository's 'URL'. For example, type 'git clone https://github.com/EnergizeLab/ServoSDK.git ' and press Enter.

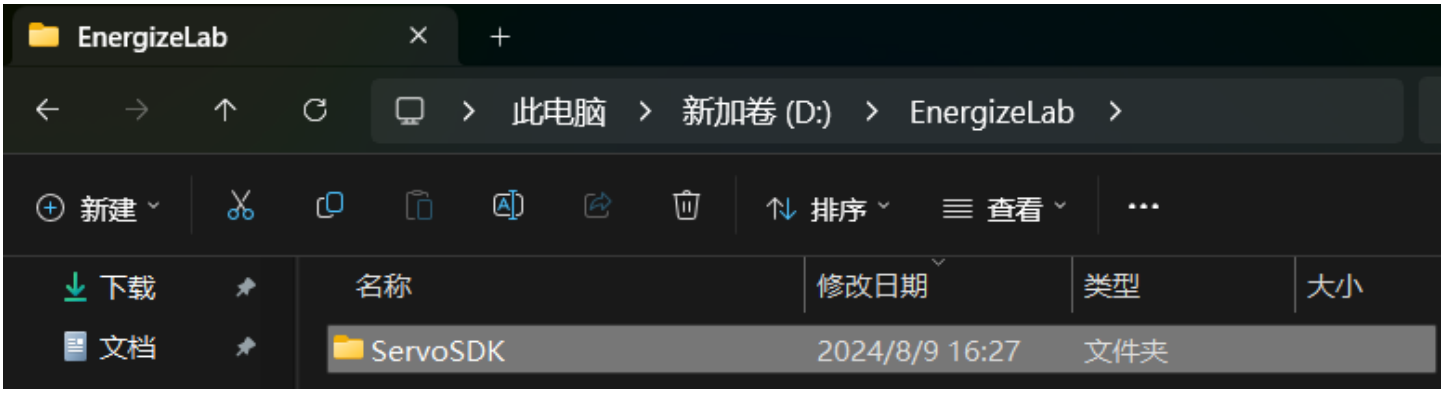


Method to obtain the target repository's 'URL'



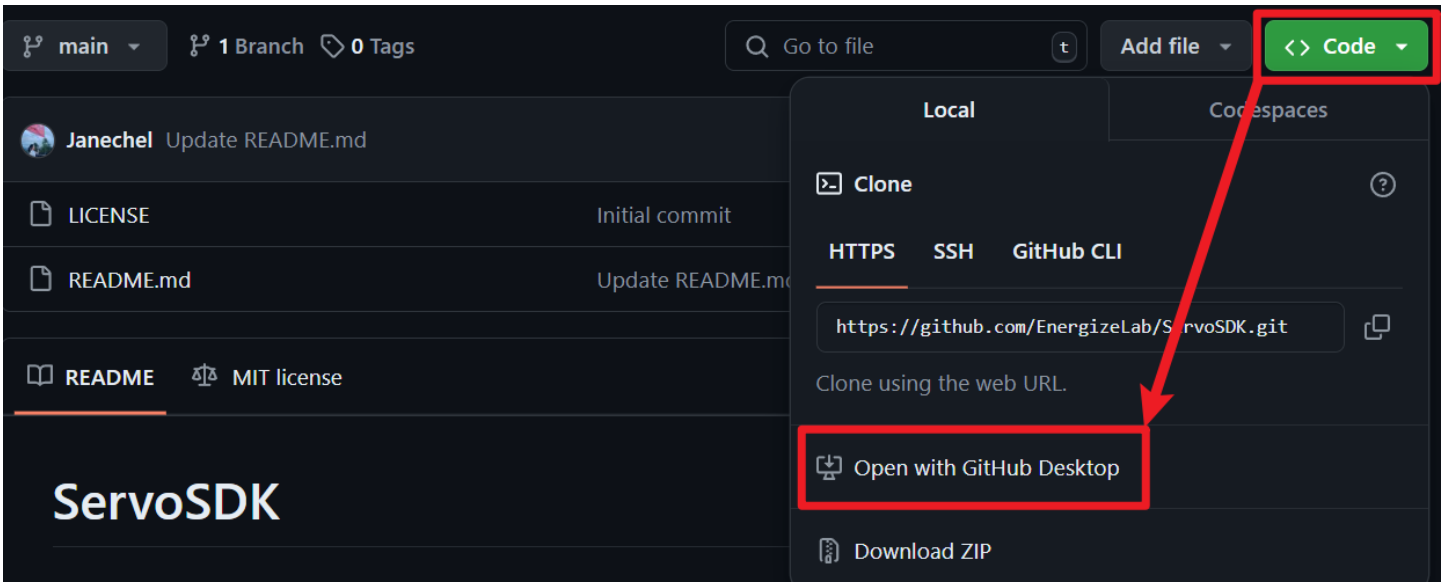
Taking Windows 11 as an example

After the download is complete, you can view the repository in the local folder.

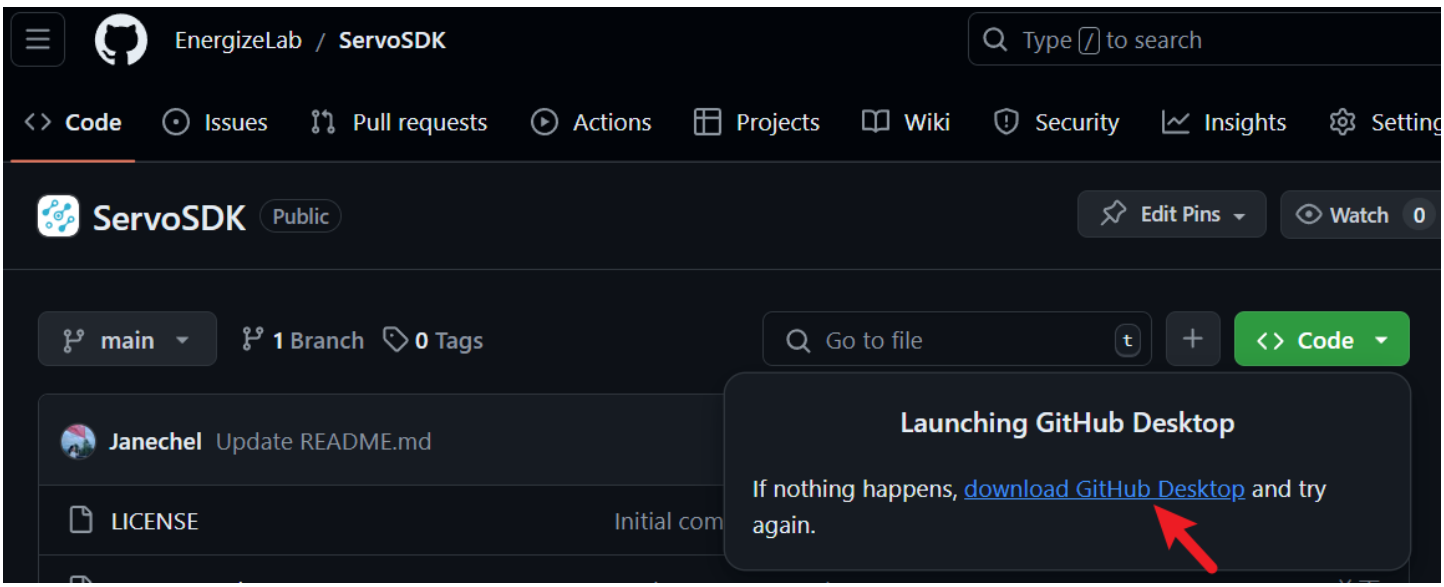


2.2.2 Open with GitHub Desktop

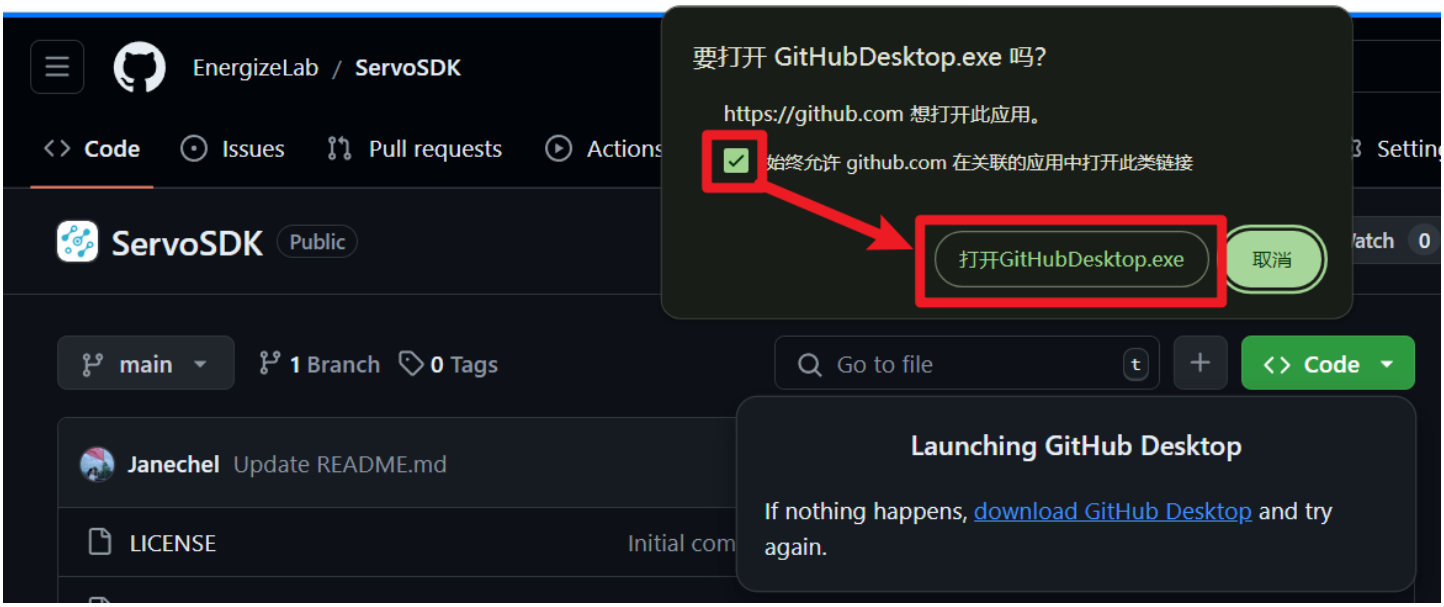
Step 1: Visit https://github.com/EnergizeLab/ServoSDK, click 'Code,' and select ' Open with GitHub Desktop.'



Step 2: Install GitHub Desktop. Click the hyperlink and follow the prompts to complete the installation.

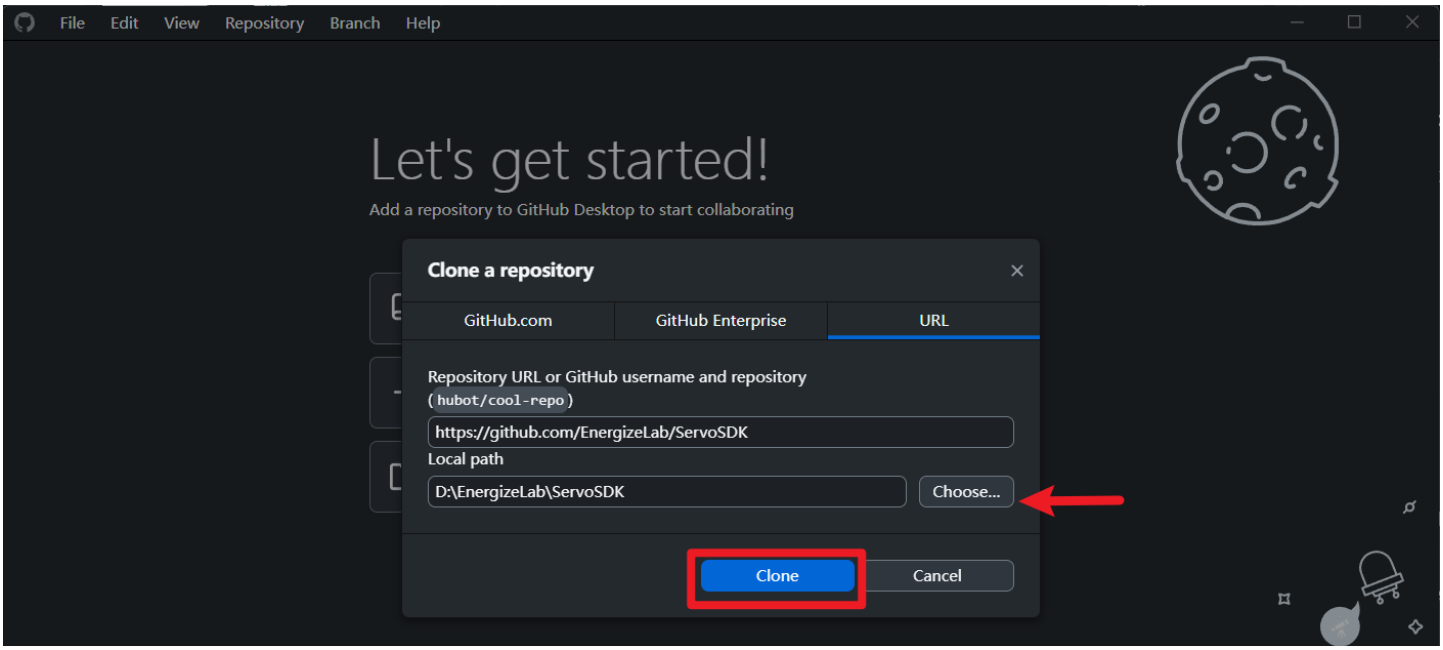


Step 3: Open with GitHub Desktop. Again, select and check "Always allow... to open this type of link," then click to open.



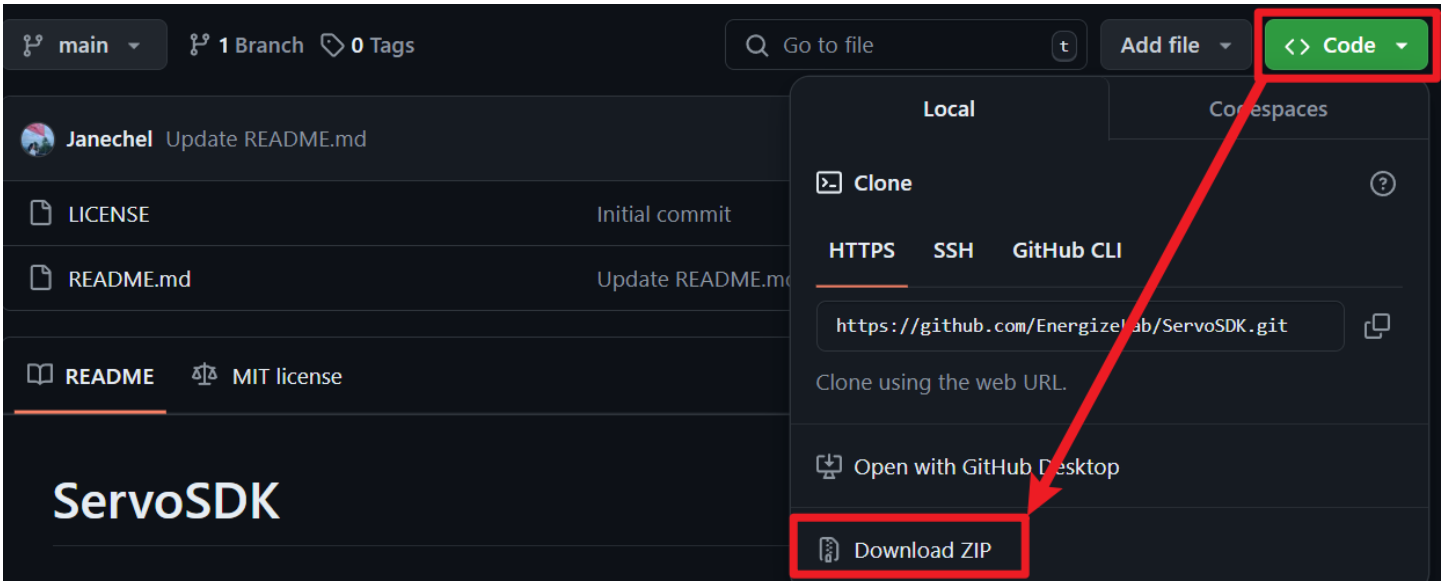
Taking Windows 11 as an example

Step 4: Clone the Repository. Choose the local folder where you want to download the repository and click ' Clone.'

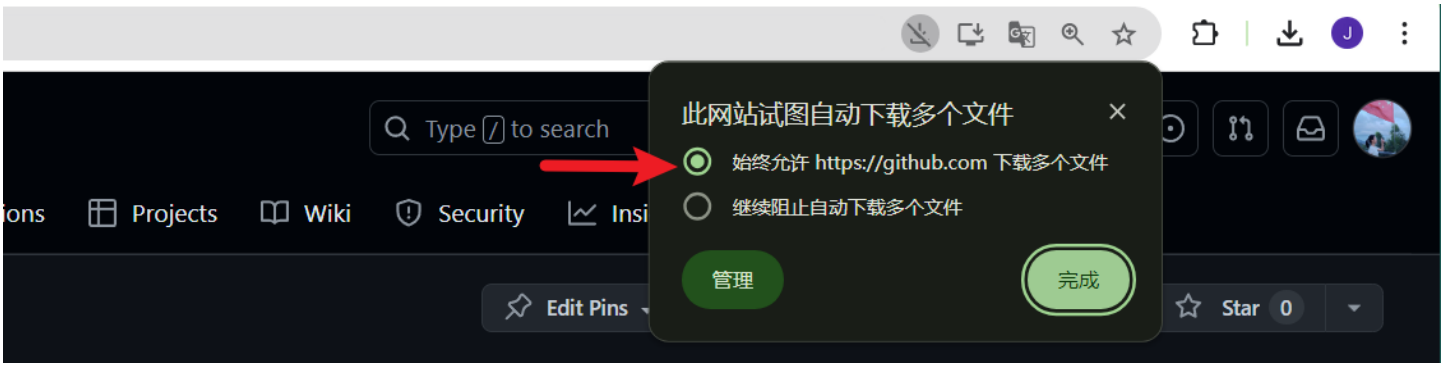


2.2.3 Download ZIP

Step 1: Visit https://github.com/EnergizeLab/ServoSDK, click 'Code', and select 'Download ZIP'.

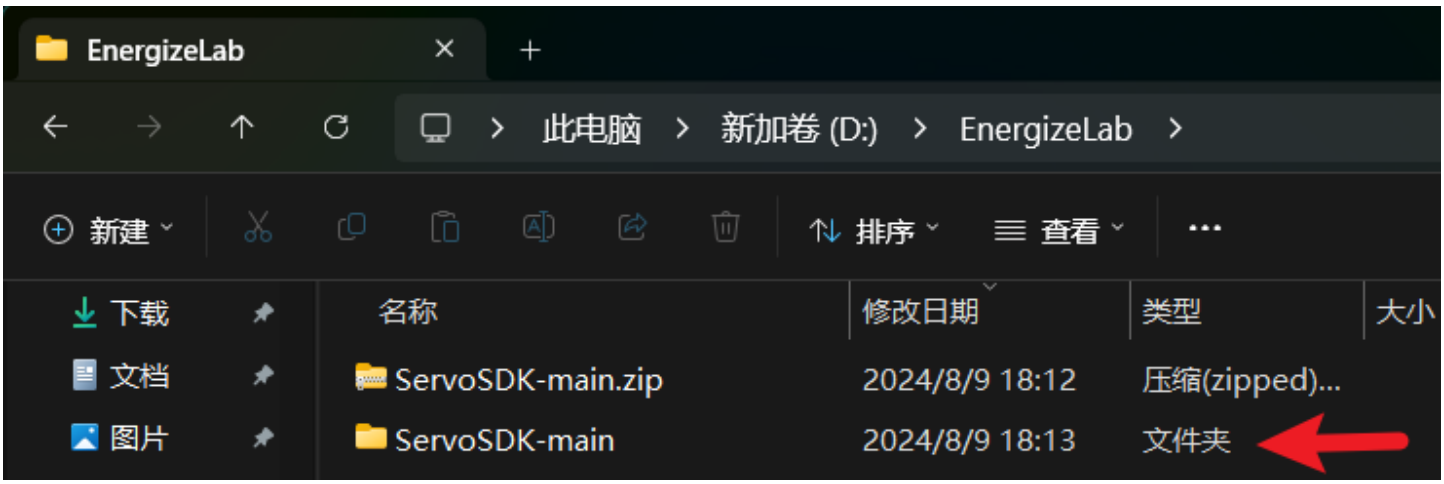


Step 2: Allow multiple file downloads. Check 'Always allow... to download multiple files' and click 'Done'.



Step 3: Clone the repository. Select the local folder where you want to download the repository and click "Save."

Step 4: Extract. After the download is complete, you need to extract the files.



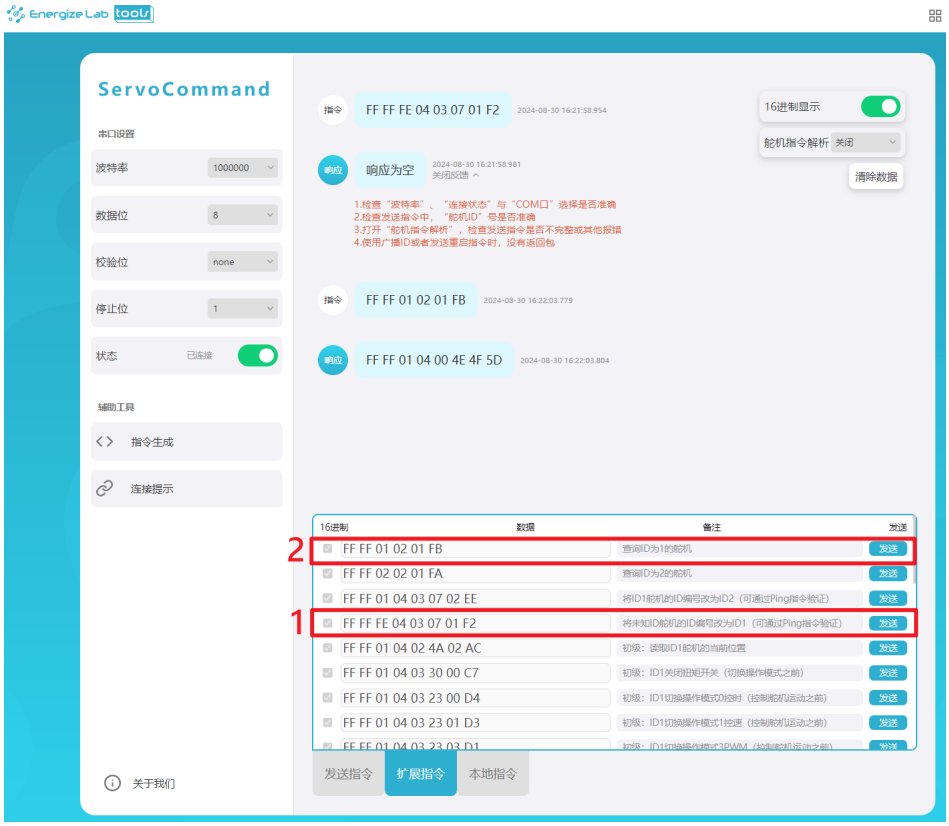
3. Servo

3.1 Preparation

To run the ServoSDK Example and Demo, you need to change the following parameters of the servo:

- ID = 1 (for testing Sync Write instruction, use 1 and 2 respectively)
- Baud Rate = 1 Mbps (the corresponding value in the memory table is 7, which usually doesn't need to be changed)

To change the servo parameters, please use ServoStudio or ServoCommand. Refer to the corresponding user manual for details.

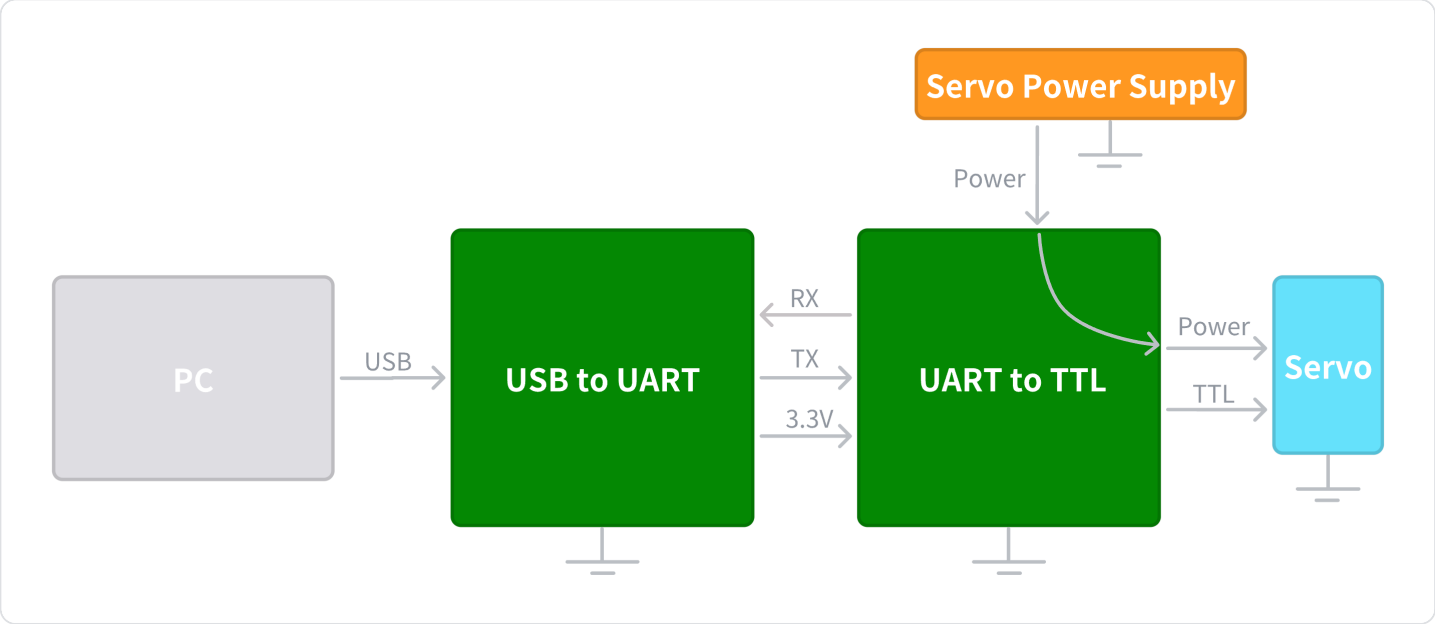


Taking modifying a single servo ID = 1 as an example

3.2 Communication Circuits

3.2.1 Servo and PC Communication

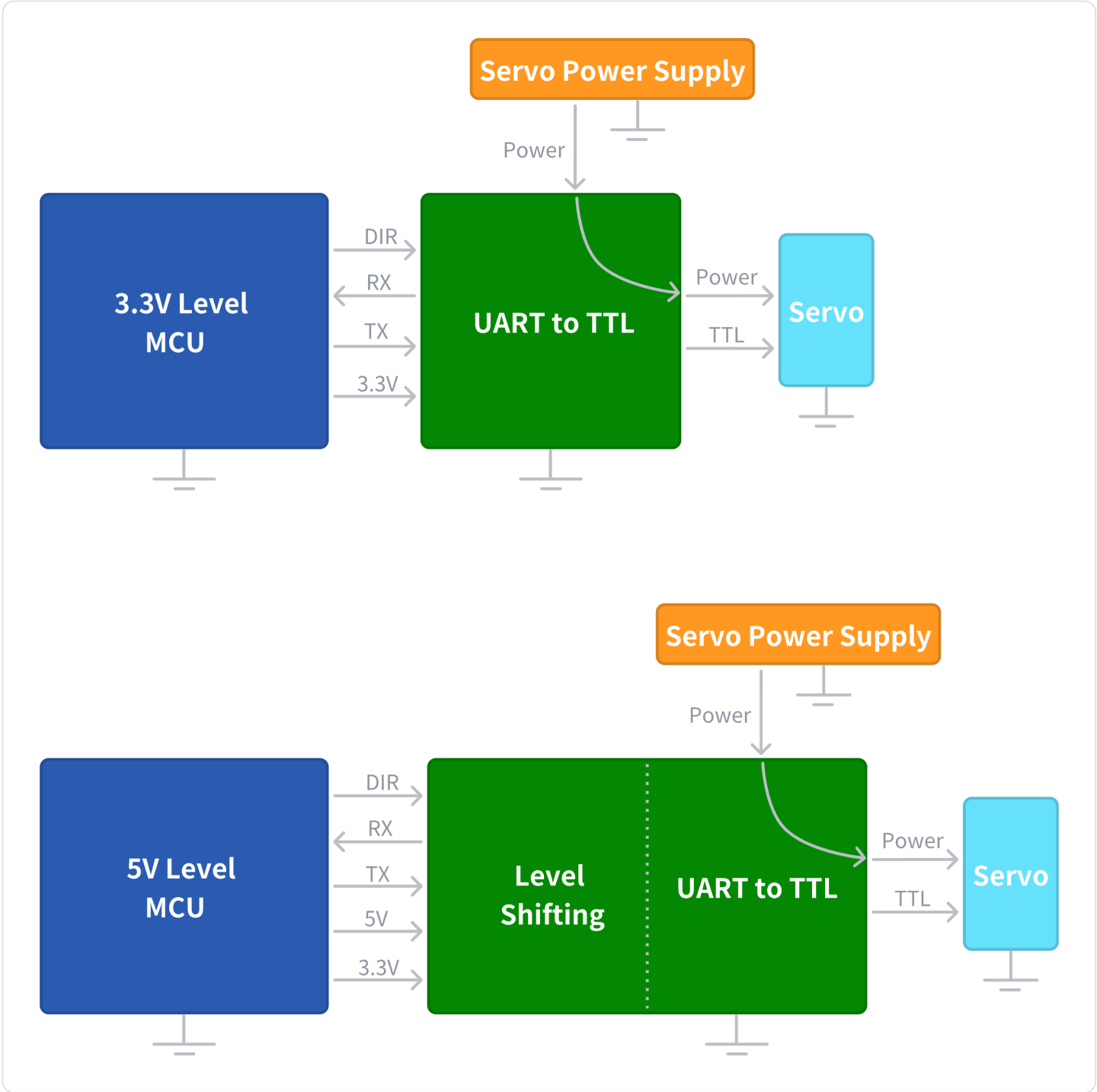
To enable communication between the Energize Lab Servo and a PC, a UART to TTL module and a USB to UART module is required, as illustrated in the diagram below:



Servo and PC Connection Diagram

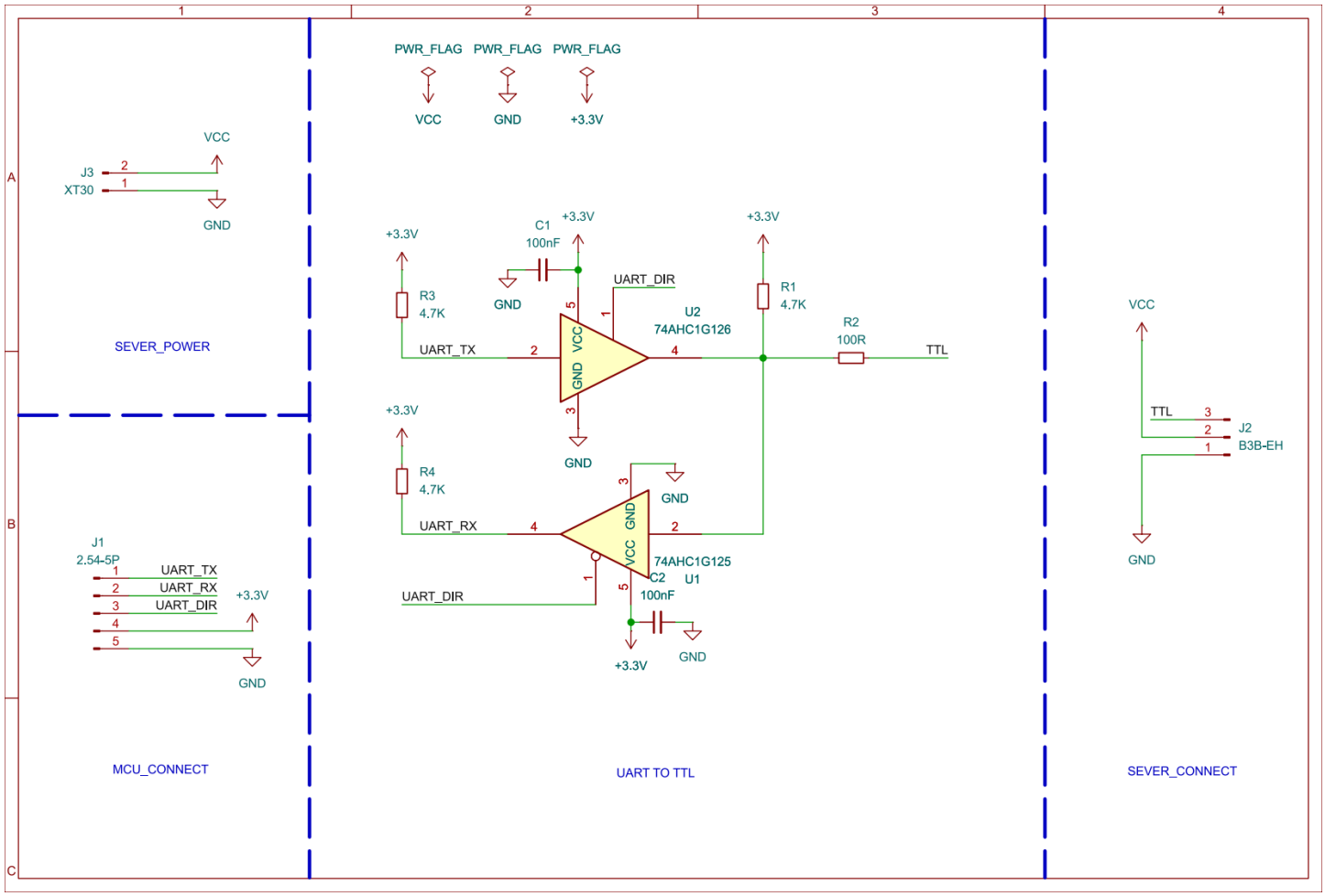
3.2.2 Servo and MCU Communication

To enable communication between the Energize Lab Servo and the MCU, a UART to TTL module is needed; if the MCU operates at 5V, a level shifting module is also necessary, as illustrated in the diagram below:

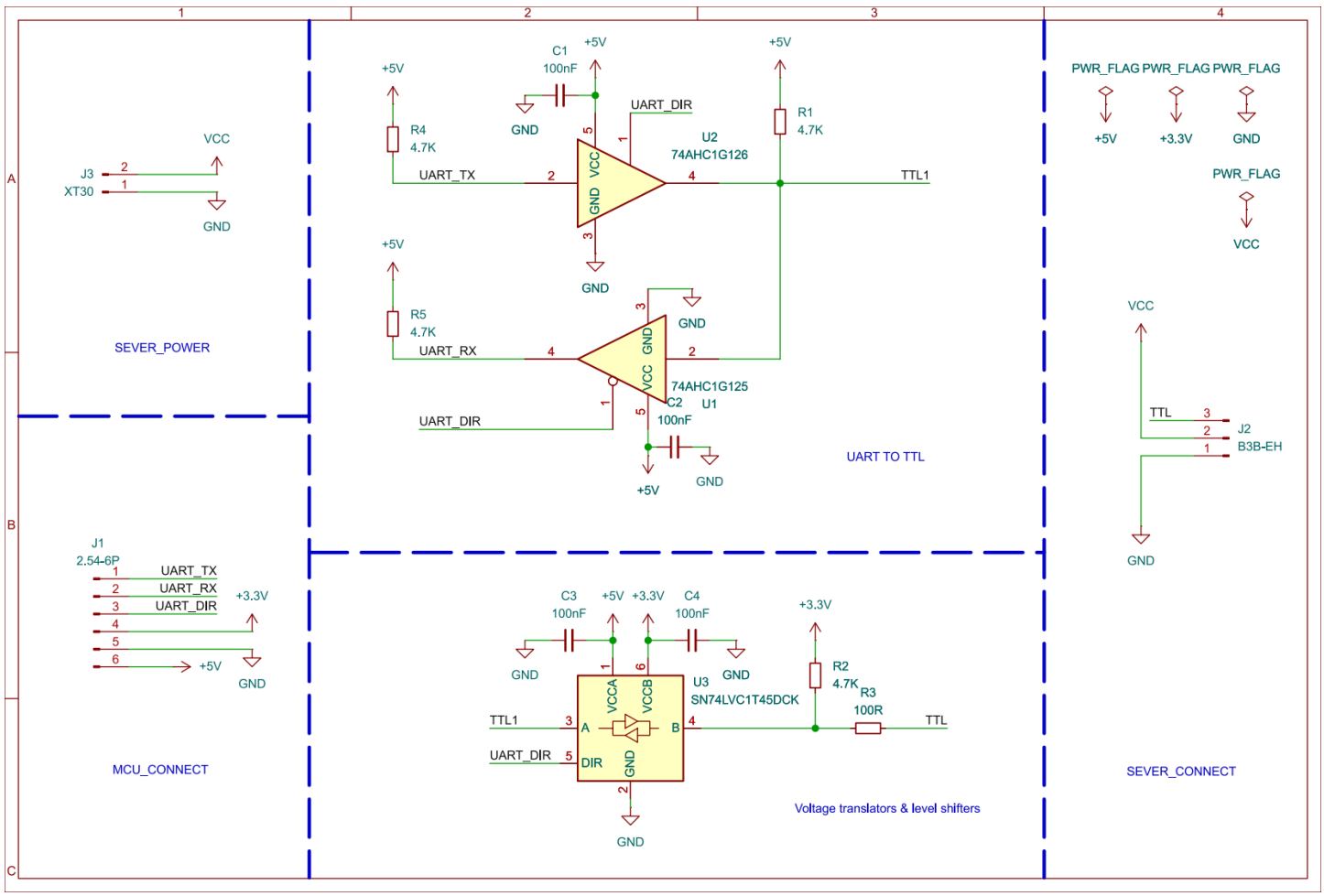


Servo and MCU Connection Diagram

UART to TTL Circuit Diagram



UART to TTL Module Circuit Diagram



Level Shifting & UART to TTL Module Circuit Diagram

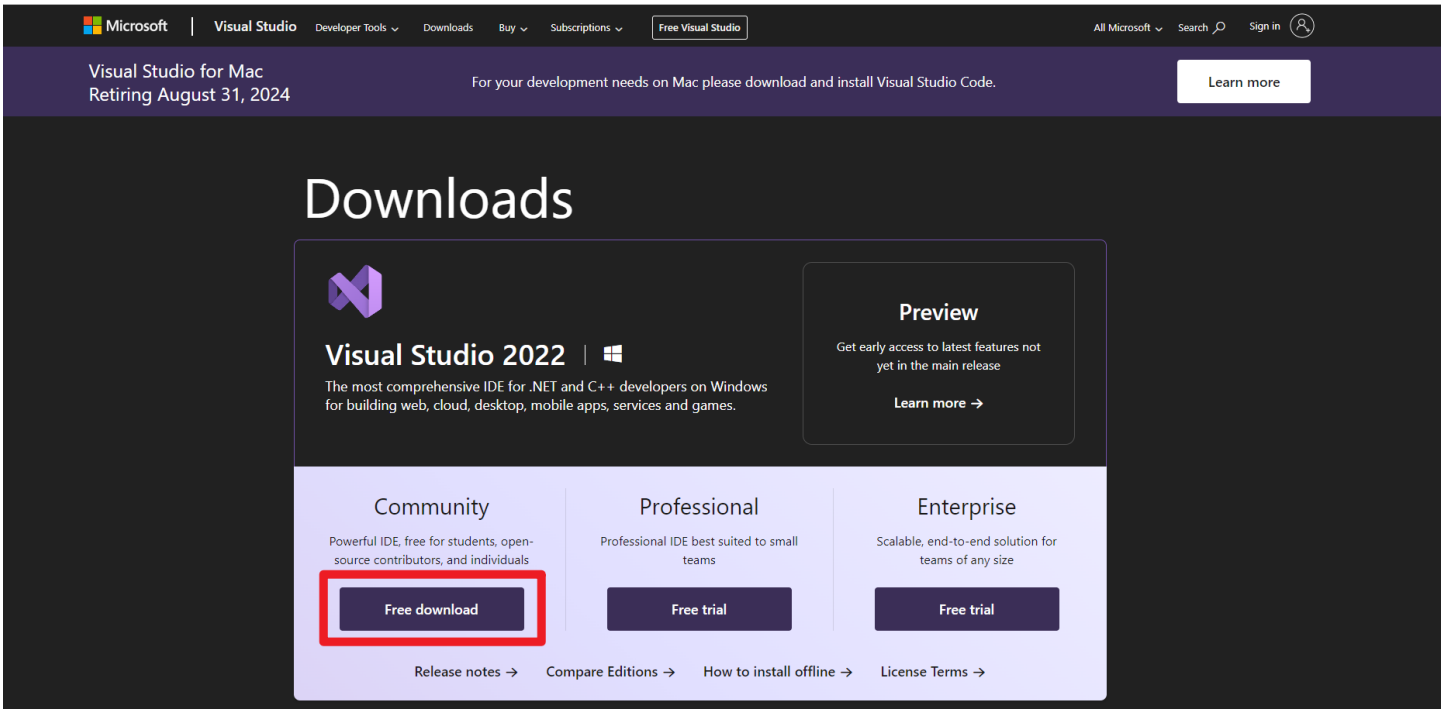
4. C

4.1 Windows

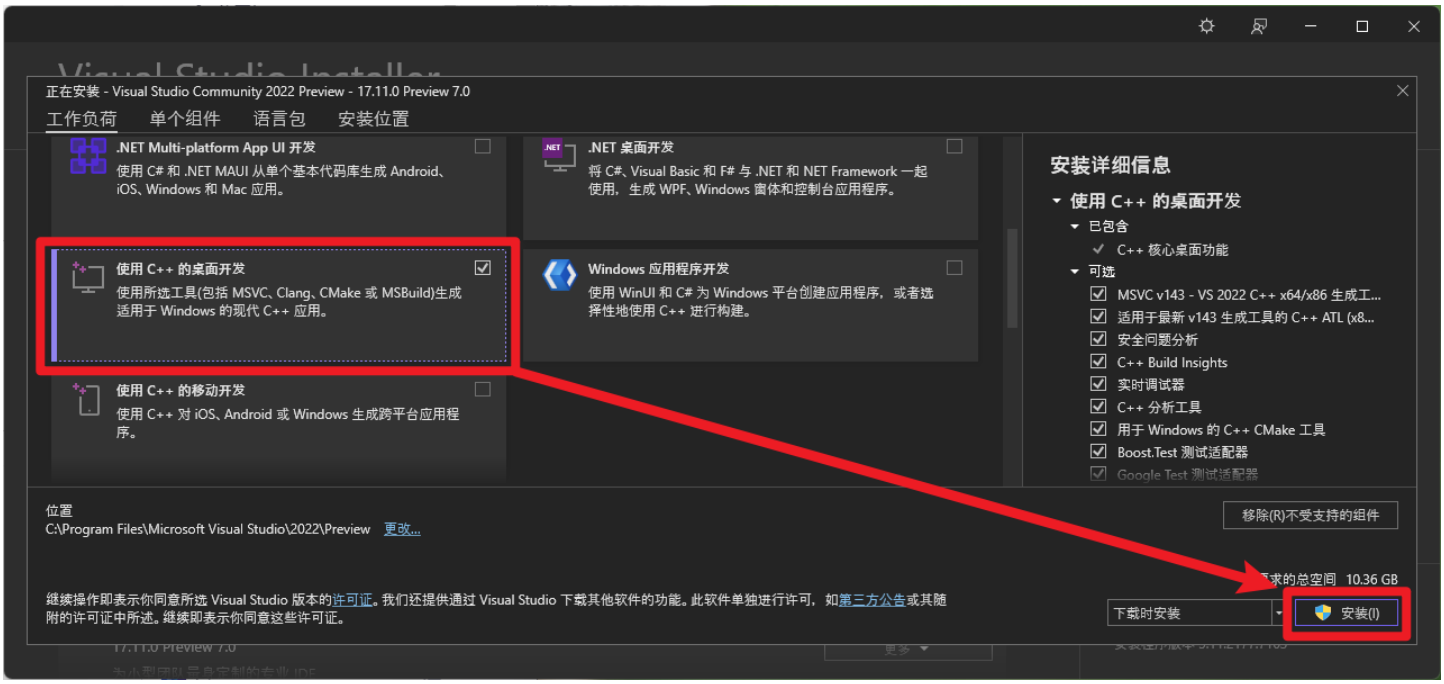
4.1.1 Environment Setup

Set up a C programming environment using the IDE, taking Visual Studio 2022 as an example.

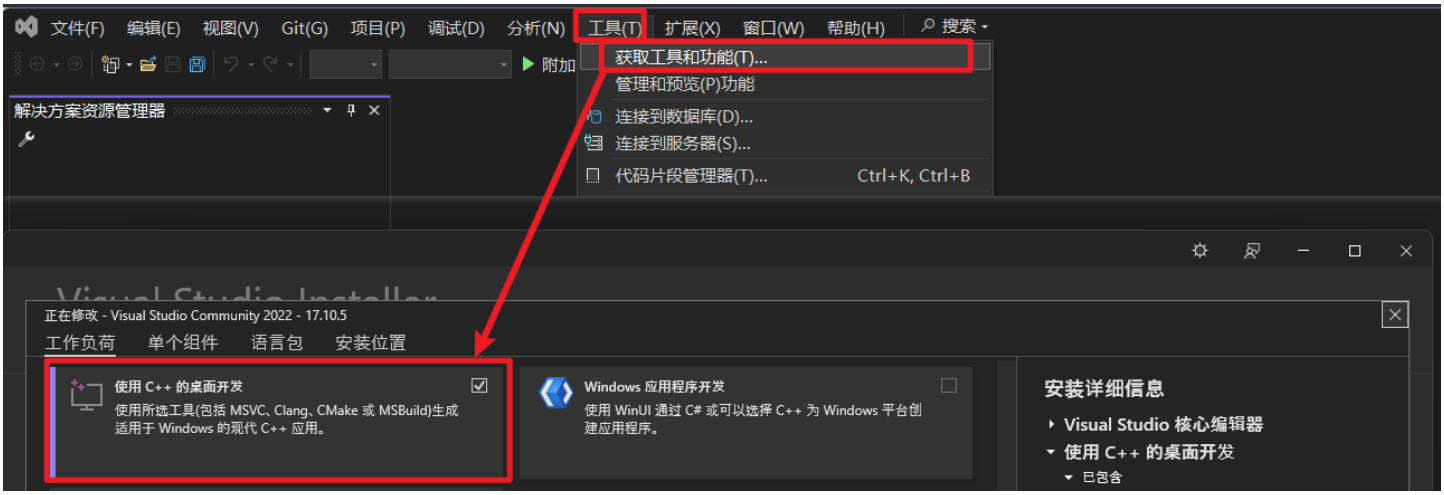
Step 1: Install Visual Studio 2022. Visit https://visualstudio.microsoft.com/downloads/ and select the appropriate version to download and install based on your computer’ s operating system.



Step 2: Install C desktop development. C is included under C++, so select 'Desktop development with C++' in the workload options.

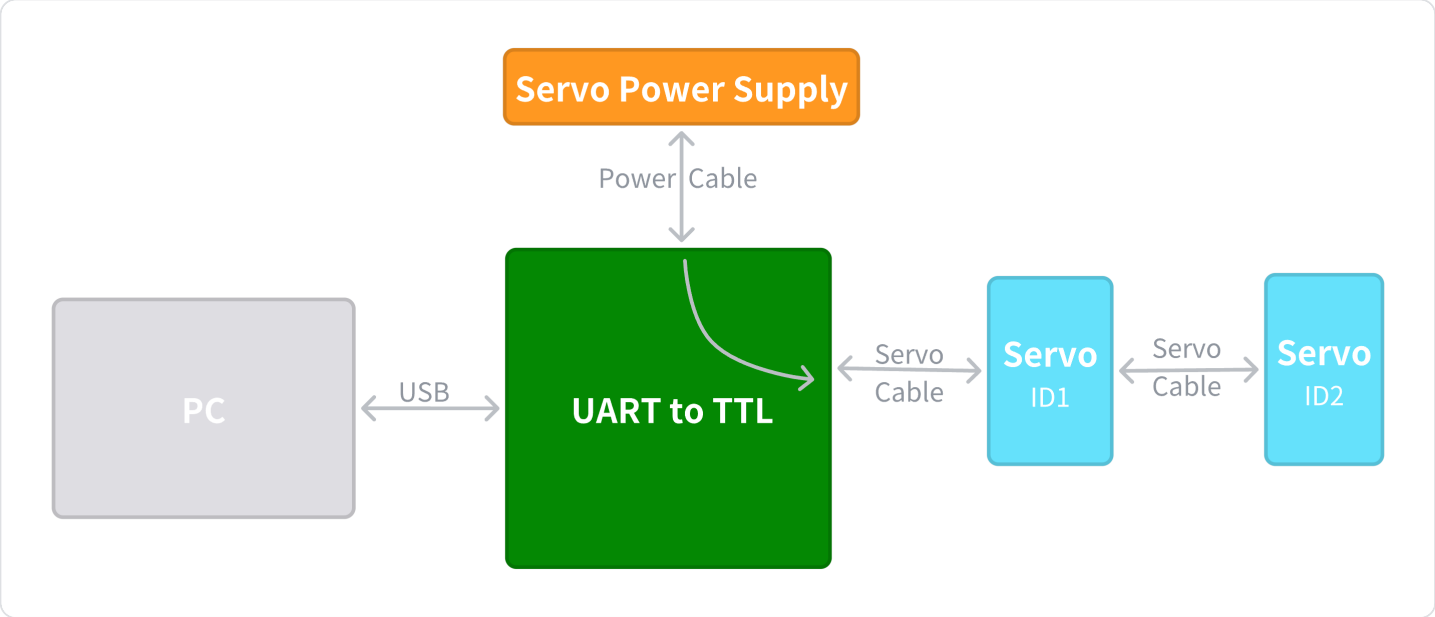


When installing the software, check 'Desktop development with C++' directly in the workload.



If not checked, you can also install it through 'Tools - Get tools and features'.

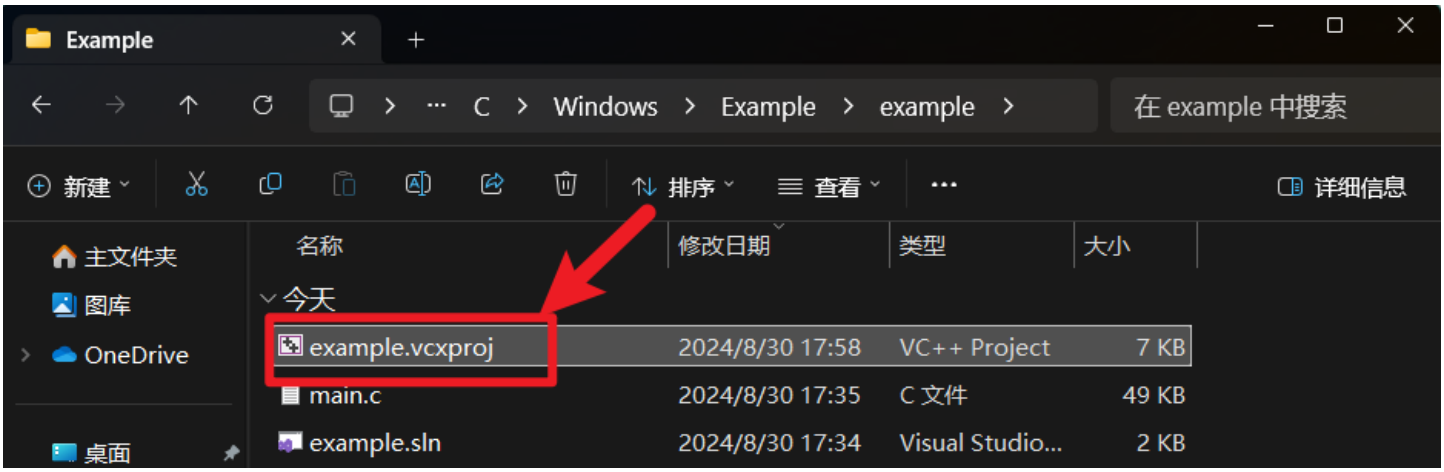
4.1.2 Hardware Connection



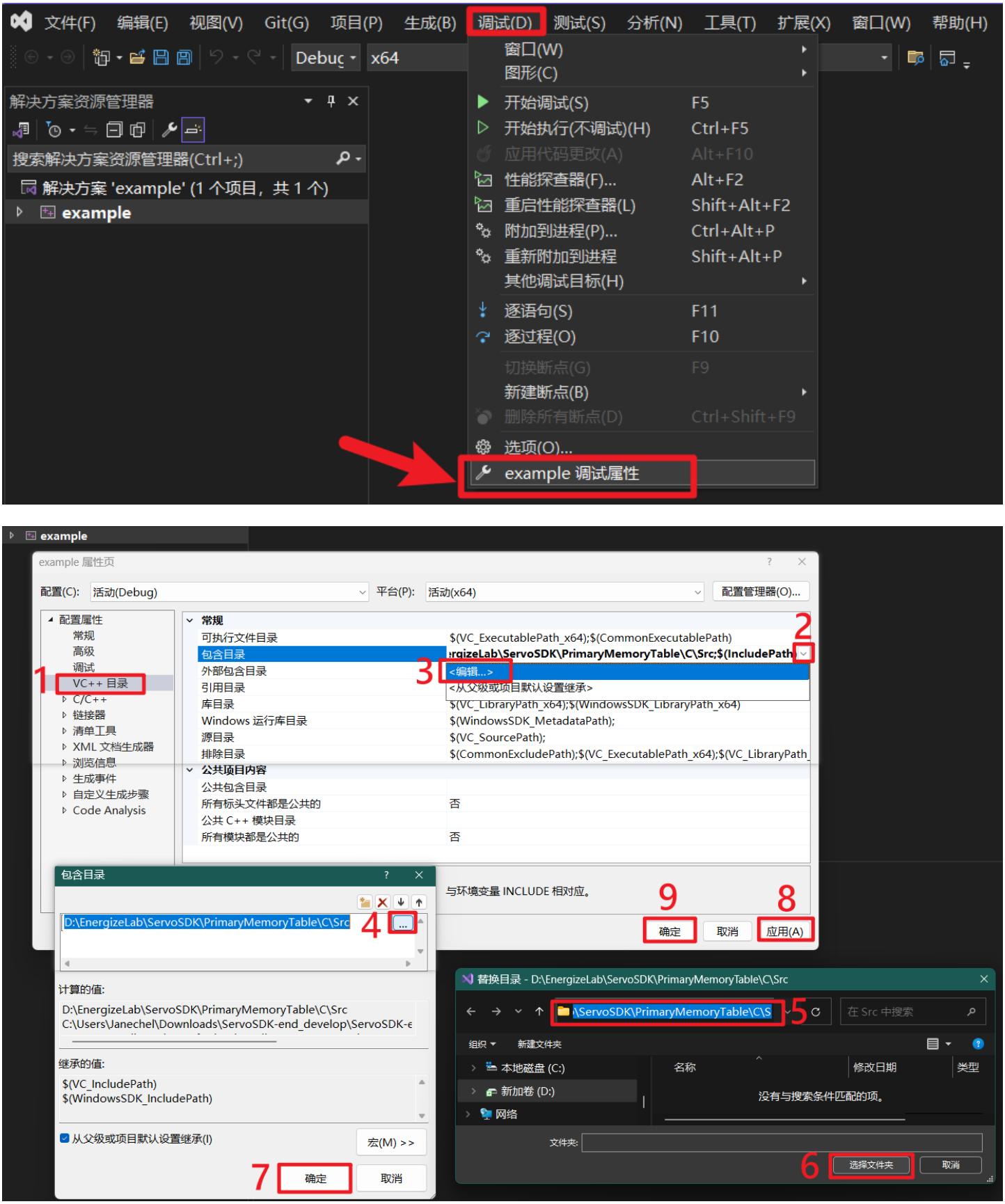
PC and Servo Connection Diagram

4.1.3 Run the Example

Step 1: Open the example.vcxproj file. Taking EM-2030 as an example, double-click to open the example.vcxproj located in ServoSDK\C\Windows\PrimaryMemoryTable\Example\example.



Step 2: Replace the include directory with the local Src path. Navigate through 'Debug -> Example debug properties -> VC++ Directories -> Include Directories dropdown -> Edit -> More -> Find local Src -> Select folder -> Confirm -> Apply -> OK'.



To avoid issues with missing .h files, it is recommended to set the Configuration (C) and Platform (P) to 'All Configurations' and 'All Platforms'

Step 3: Change the actual USBLink port number. In line 121 of the main.c file, change '\\\\.\\COM18' to the actual USBLink port number.

Note: In C/C++, since Microsoft’ s predefined standard devices only include 'COM1-COM9', if the user connects to a hardware port 'COM10 or above', a specific format must be used, such as

'\\\\.\\COM10'. Otherwise, after starting debugging, you will encounter 'Fail to open port'.

```
//Taking COM6 as an example
HANDLE hSerial = CreateFile("COM6", GENERIC_READ | GENERIC_WRITE, 0, NULL,
OPEN_EXISTING, FILE_ATTRIBUTE_NORMAL, NULL);
```

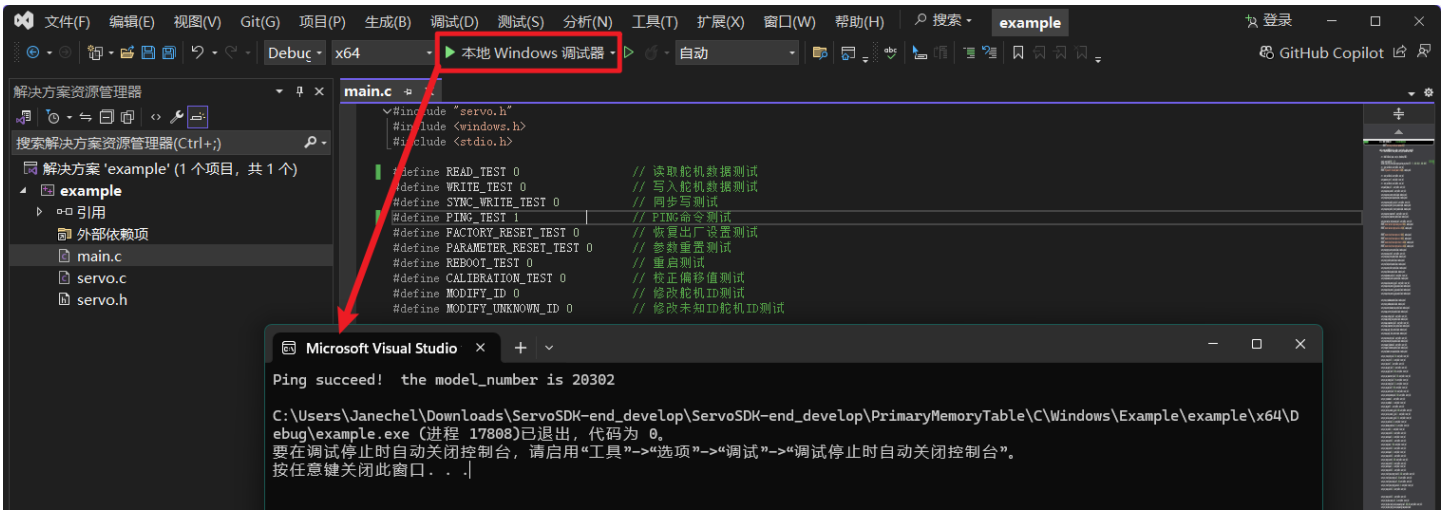
Step 4: Change the module value you want to test to 1. In lines 5-14 of the main.c file, change the module value from '0' to '1' for the module you want to test.

```
//Taking test the Ping module as an example
#define READ_TEST 0                 //Read Servo Data Test
#define WRITE_TEST 0                //Write Servo Data Test
#define SYNC_WRITE_TEST 0           //Sync Write Test
#define PING_TEST 1                 //PING Instruction Test
#define FACTORY_RESET_TEST 0        //Factory Reset Test
#define PARAMETER_RESET_TEST 0      //Parameter Reset Test
#define REBOOT_TEST 0               //Reboot Test
#define CALIBRATION_TEST 0          //Calibration Test
#define MODIFY_ID 0                 //Change Known Servo ID Test
#define MODIFY_UNKNOWN_ID 0         //Change Unknown Servo ID Test
```

Step 5: Change the maximum number of servos for sync write and the print output enable as needed. In lines 4-5 of the PrimaryServo.h file, MAX_SERVERS indicates the maximum supported number of sync write servos. If the actual number of servos exceeds 20, change it accordingly; it should be greater than or equal to the actual number of servos. If print output is not needed, set PRINTF_ENABLE to '0'.

```
//Taking 1 servo and print enable as an example (no change is required)
#define MAX_SERVERS 20              //Maximum number of servos for sync write.
#define PRINTF_ENABLE 1             //Print output enable.
```

Step 6: Run the program. Click the 'Local Windows Debugger' button in VS to run the program.



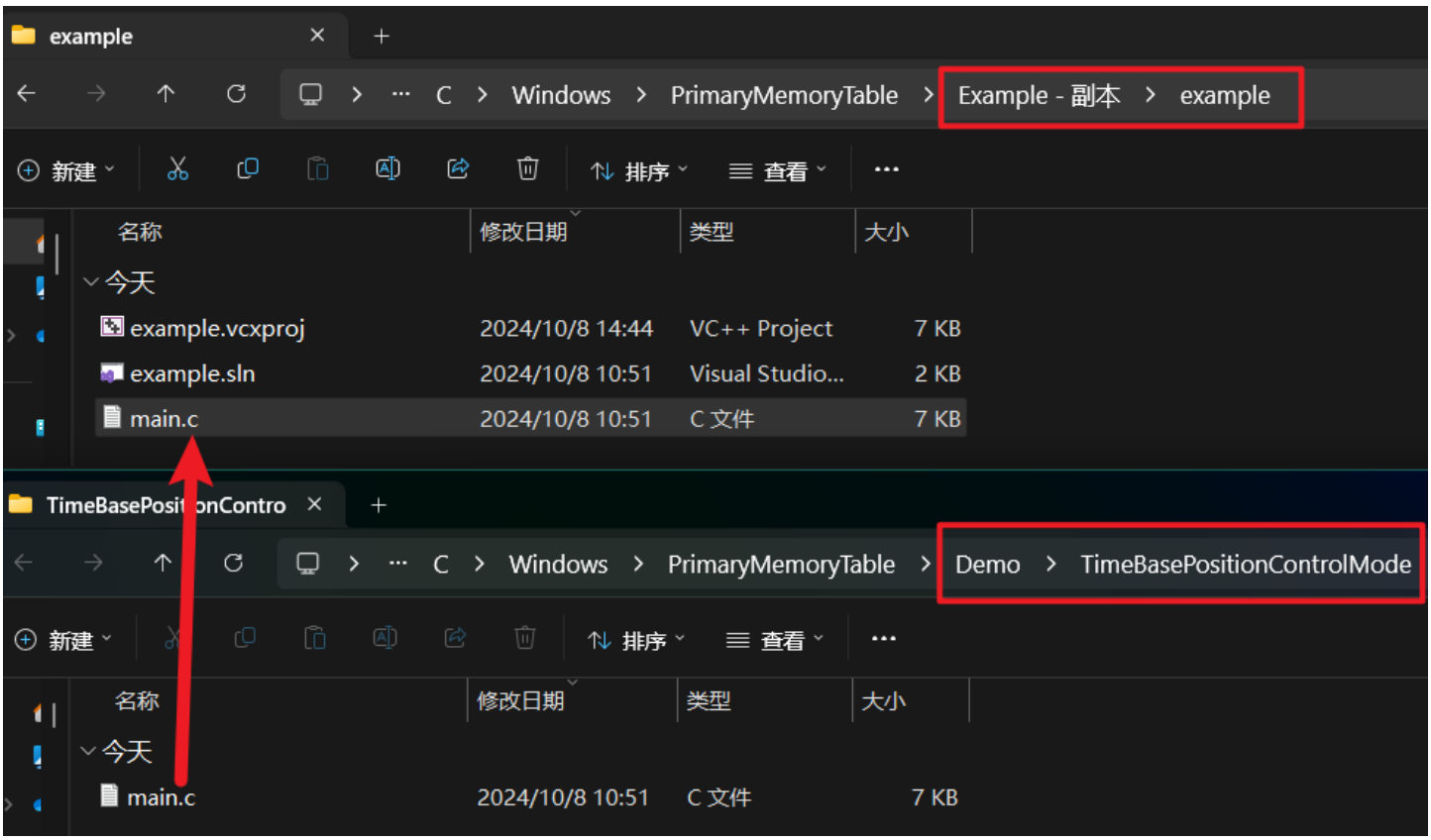
Taking test the Ping module as an example

4.1.4 Run the Demo

In the same programming language, the servo operation libraries (Src) across different platforms are identical. To run the Demo, one simply needs to replace the main file in the platform's Example with a main file from the Demo that corresponds to a specific motion control.

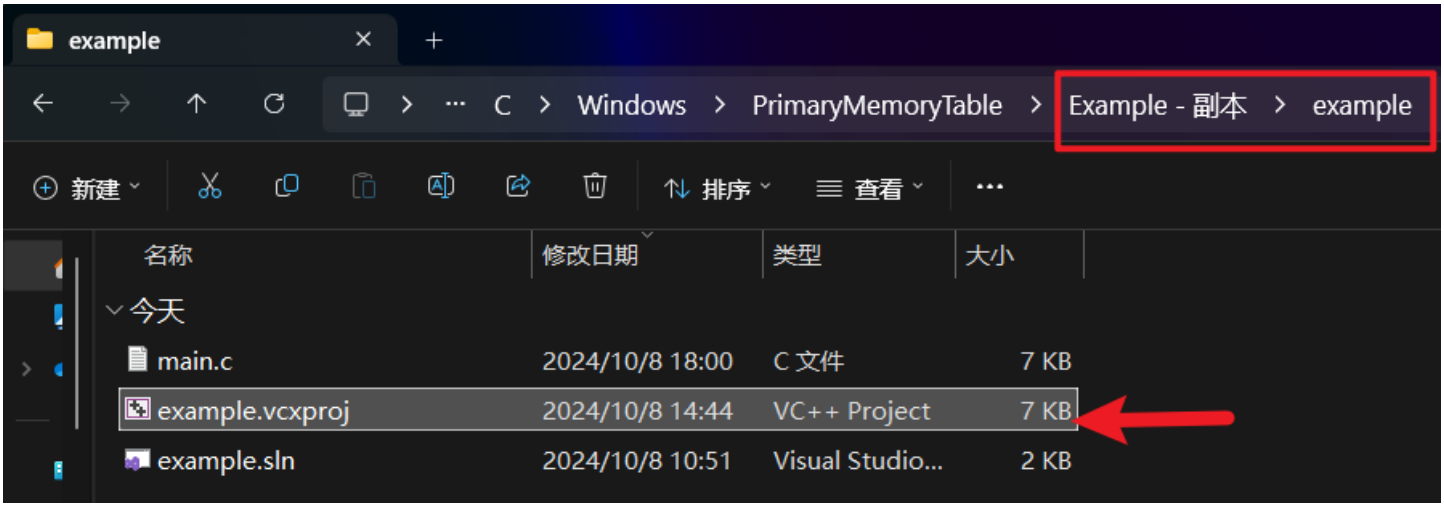
For instance, taking the C language, Windows platform, and the EM-2030 time base position control Demo as an example.

Step 1: Create a copy of the Example and replace the main file. Replace the main file in the copy with the main file from a specific motion control Demo.



delete the main file in the copy and copy the main file from the Demo into the copy

Step 2: Open the project file in the copy.



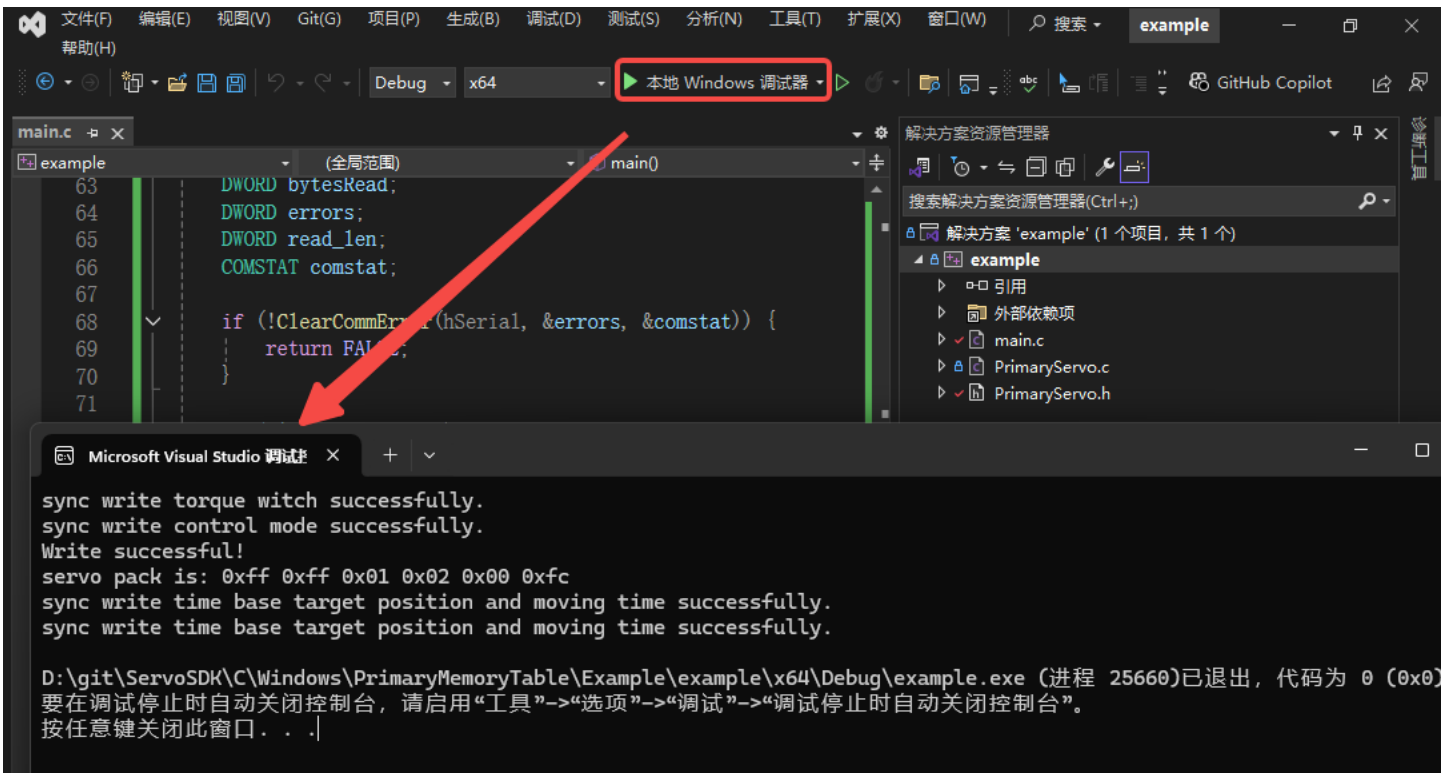
Taking open the `example.vcproj` file as an example

Step 3: Replace the included directories with the local Src. Refer to *section 4.1.3 Run the Example*; if it has already been replaced, you can skip this step.

Step 4: Change the actual USBLink port number. In the `main.c` file, change the code on line 105 from `"\\\\.\\COM16"` to the actual USBLink port number.

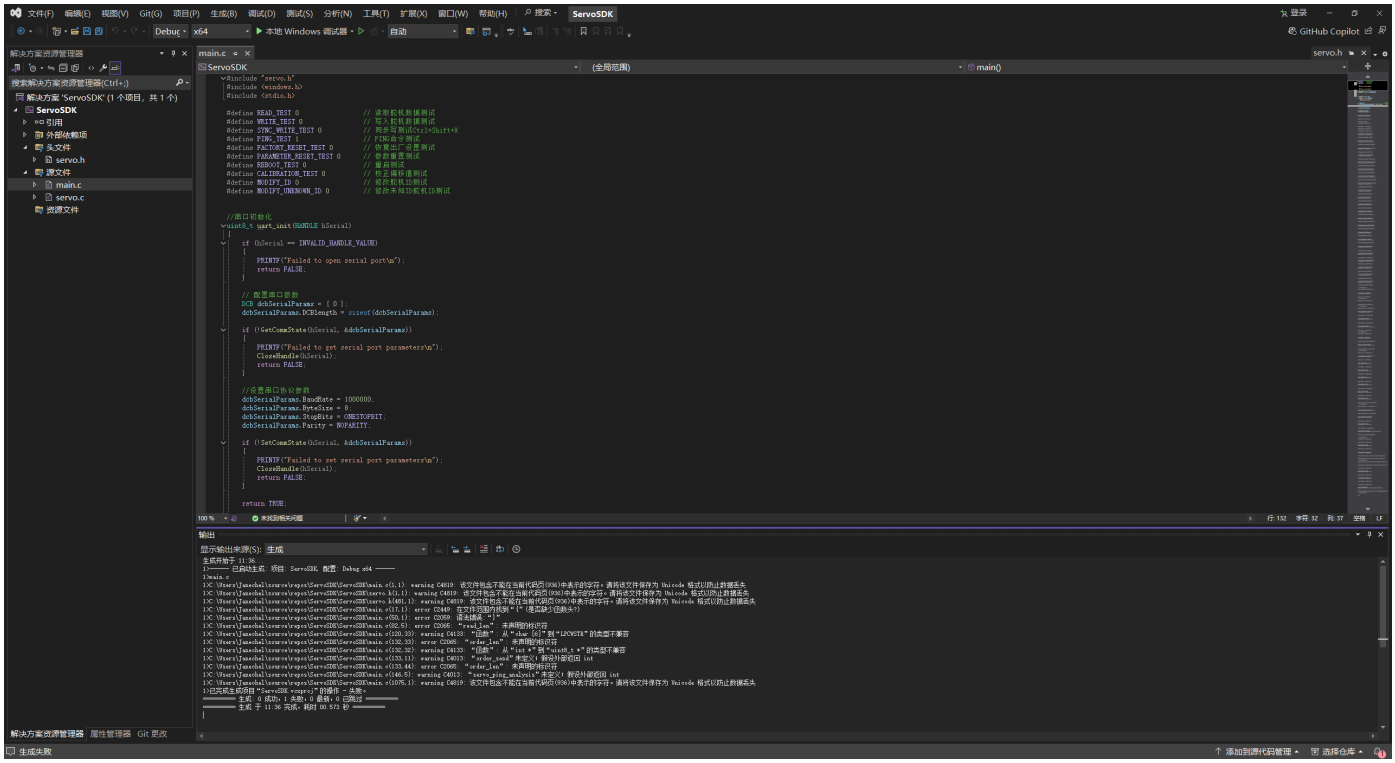
```
1 //Taking COM8 as an example
2 HANDLE hSerial = CreateFile("COM8", GENERIC_READ | GENERIC_WRITE, 0,
  NULL,OPEN_EXISTING, FILE_ATTRIBUTE_NORMAL, NULL);
```

Step 5: Run the program.

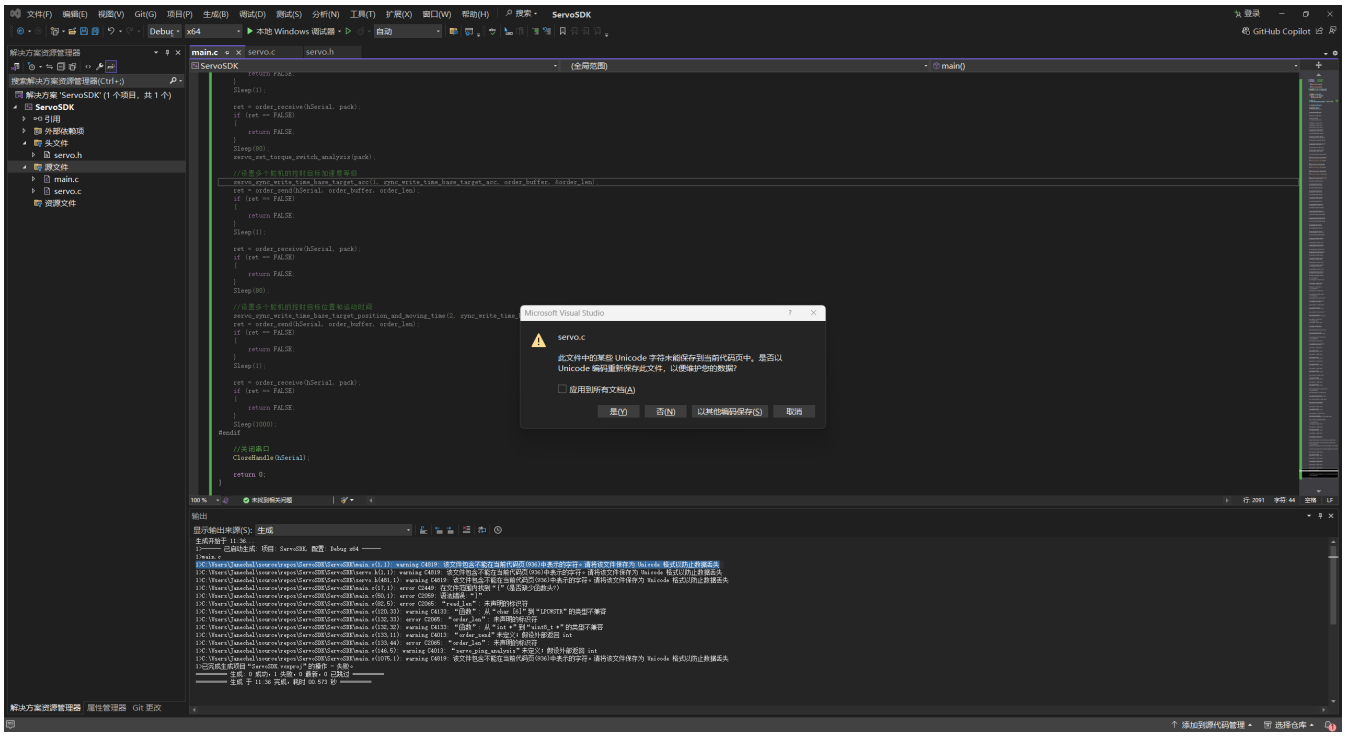


4.1.5 FAQ

- If users are debugging the program by selecting "Create New Project," directly copying main.c, PrimaryServo.c, and PrimaryServo.h from C\Windows\Example into their newly created project folder, and then adding these three files to the project using "Add Existing Item," they might encounter errors such as "This file contains characters that cannot be represented in the current code page."



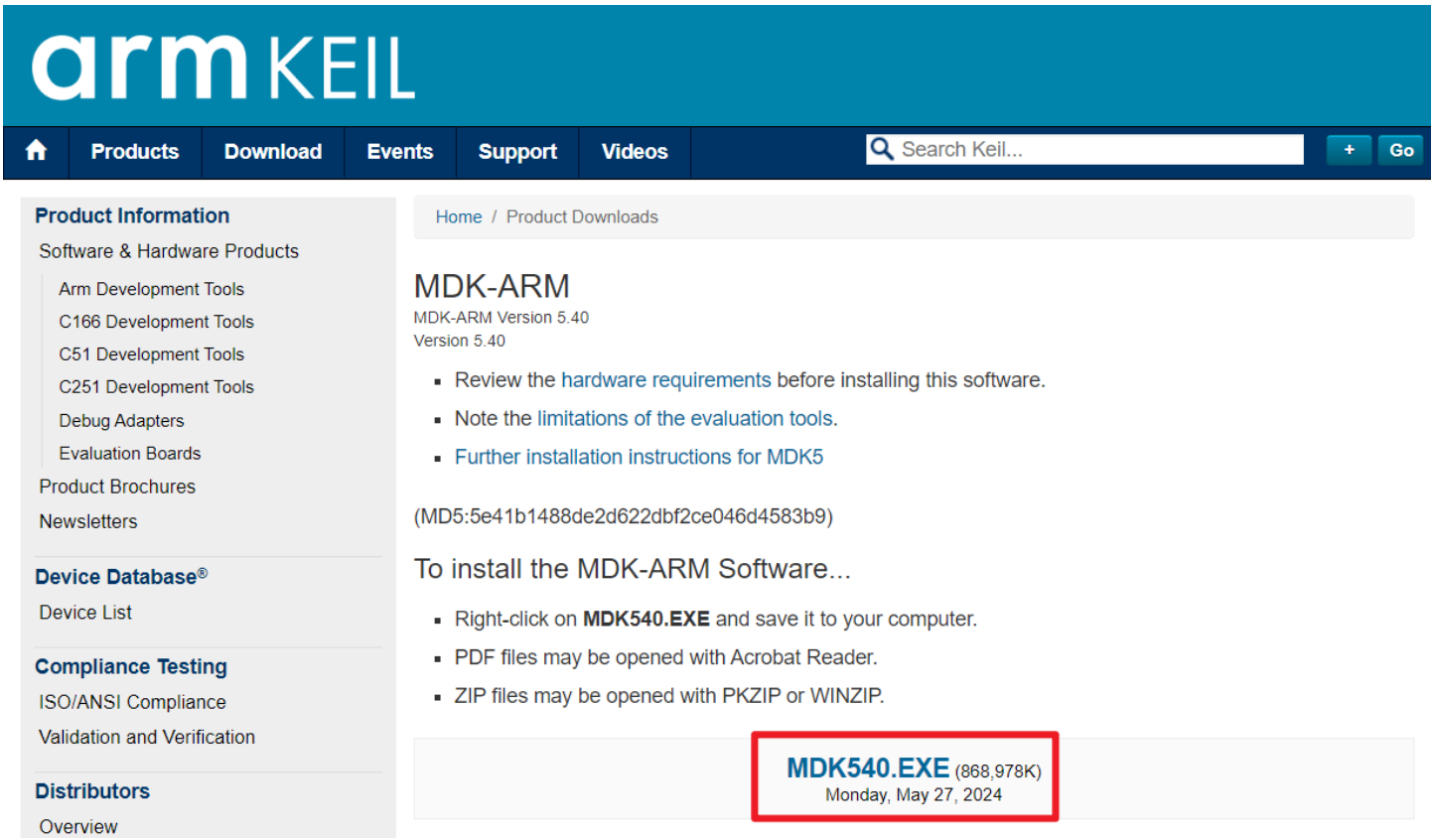
  - The correct approach is to use "Add New Item," select the appropriate file type, rename the files, and confirm their addition. Next, the corresponding code from the three files in the SDK should be copied into the newly created files. Once the code modifications are complete, click "Start Debugging." If a pop-up appears stating "Some Unicode characters in this file cannot be saved in the current code page. Would you like to save this file with Unicode encoding to maintain your data?", click "Yes." At this point, the execution should proceed normally.



4.2 STM32 Standard Library

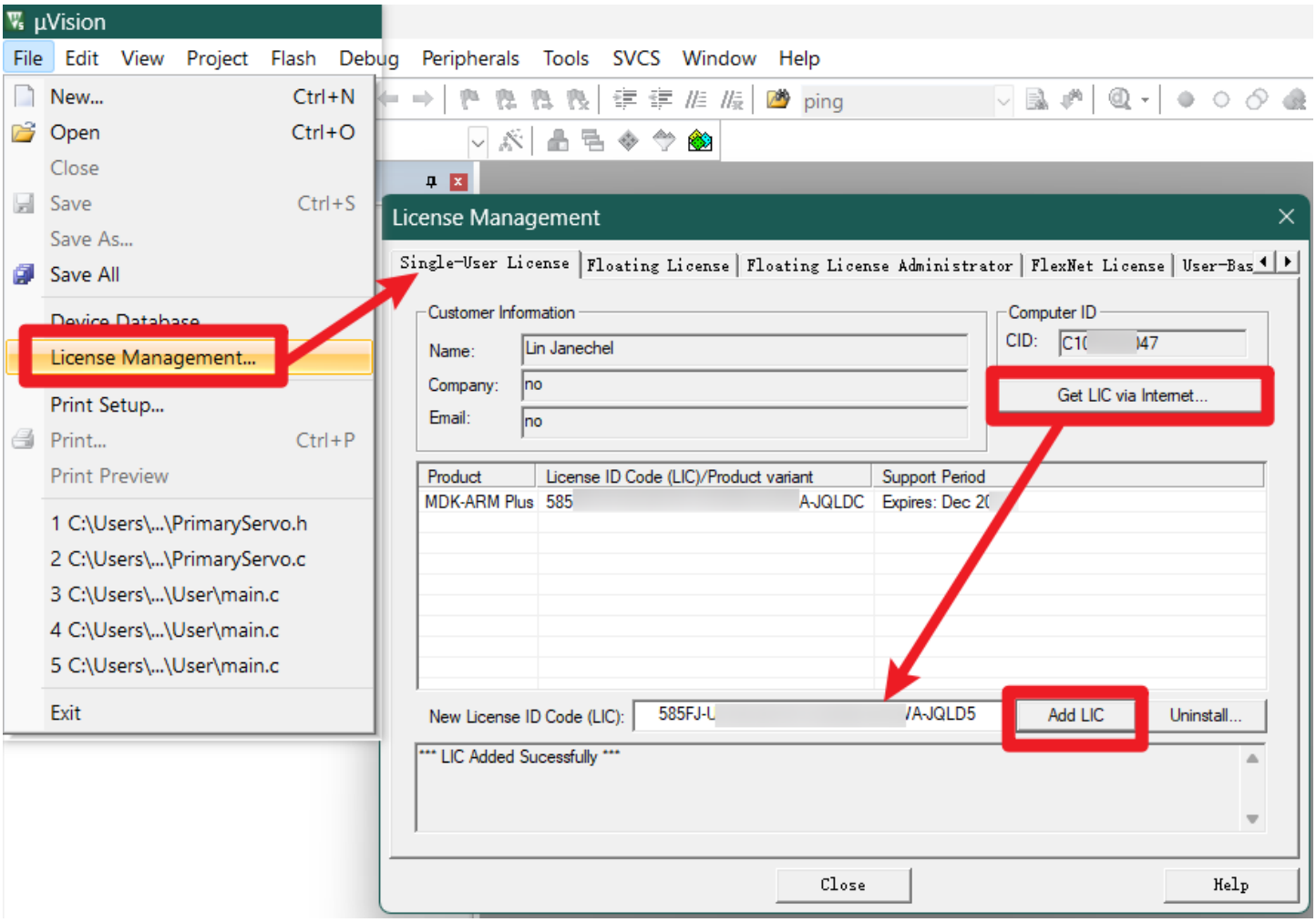
4.2.1 Environment Setup

Step 1: Install Keil5. Visit http://www.keil.com/download/product/, select 'MDK-ARM', and follow the prompts to download and install.

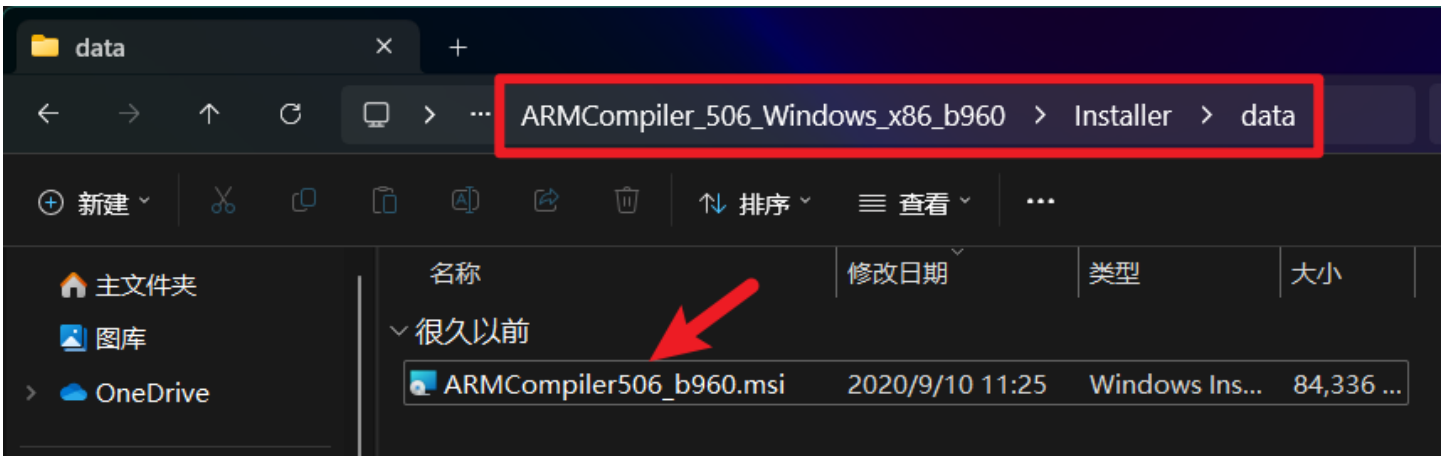


Note: To avoid issues, it is recommended not to select a folder that requires administrator privileges (e.g., Program Files) for the installation path

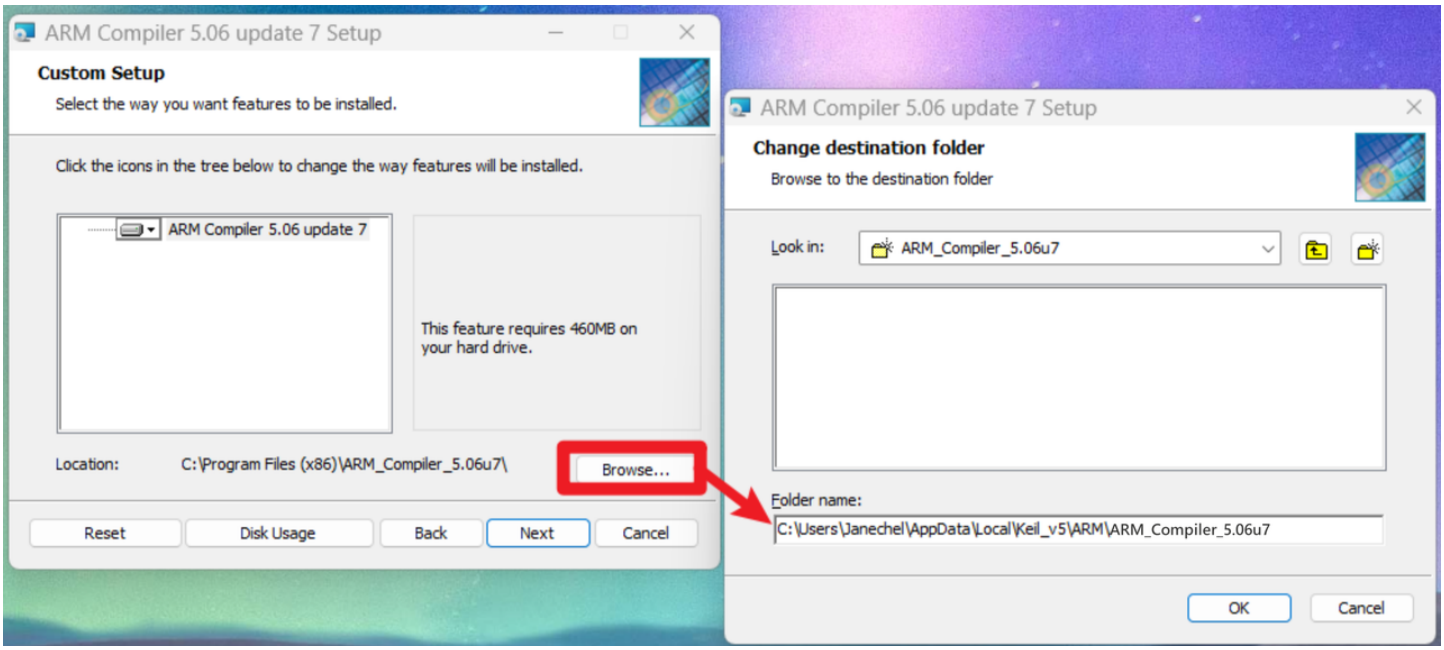
Step 2: Register Keil5 MDK. Obtain the license key through official channels, then click on File -> License Management -> Single-User, enter the license key, and click Add LIC to complete the registration.



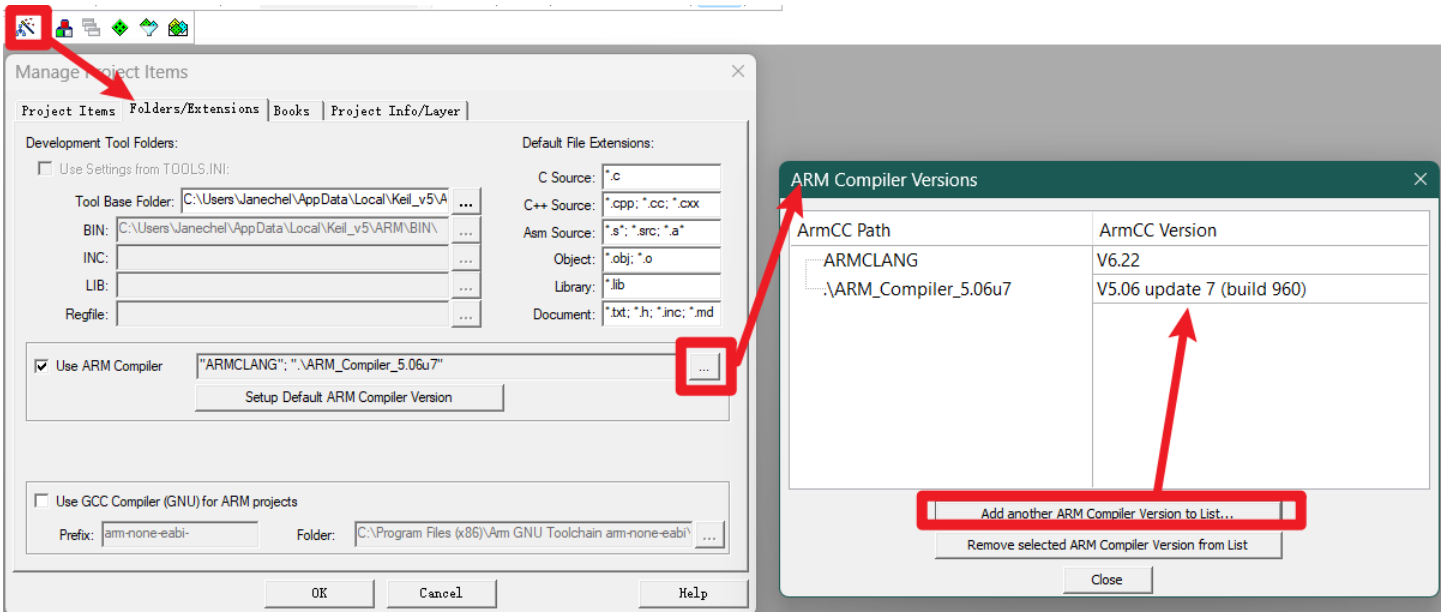
Step 3: Install Compiler Version 5. Visit https://developer.arm.com/downloads/view/ACOMP5 and follow the instructions to download and install it.



Extract the installation package and run the .msi file.

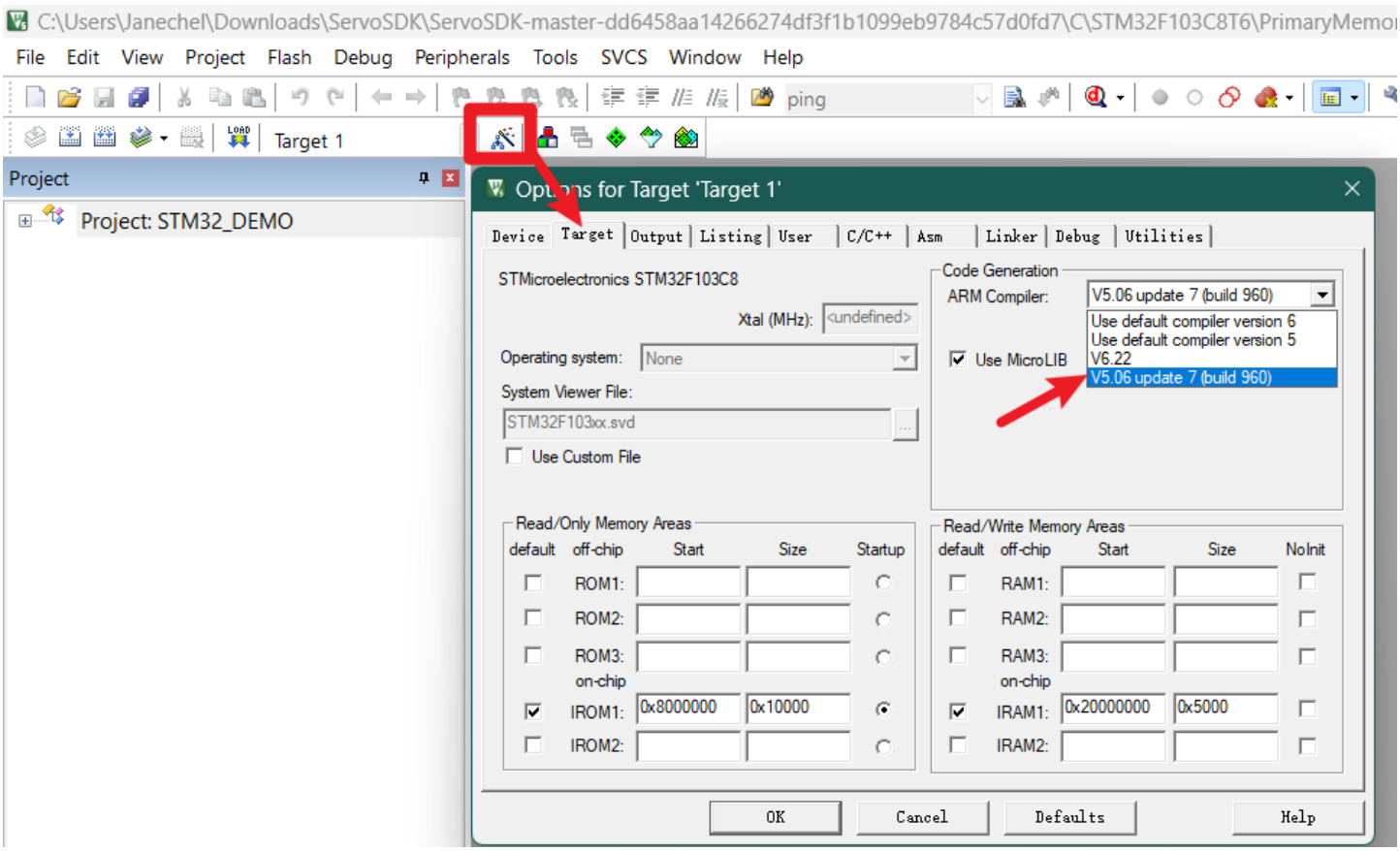


Locate the Keil_v5 installation directory using the desktop shortcut, and replace it with the previously noted path.



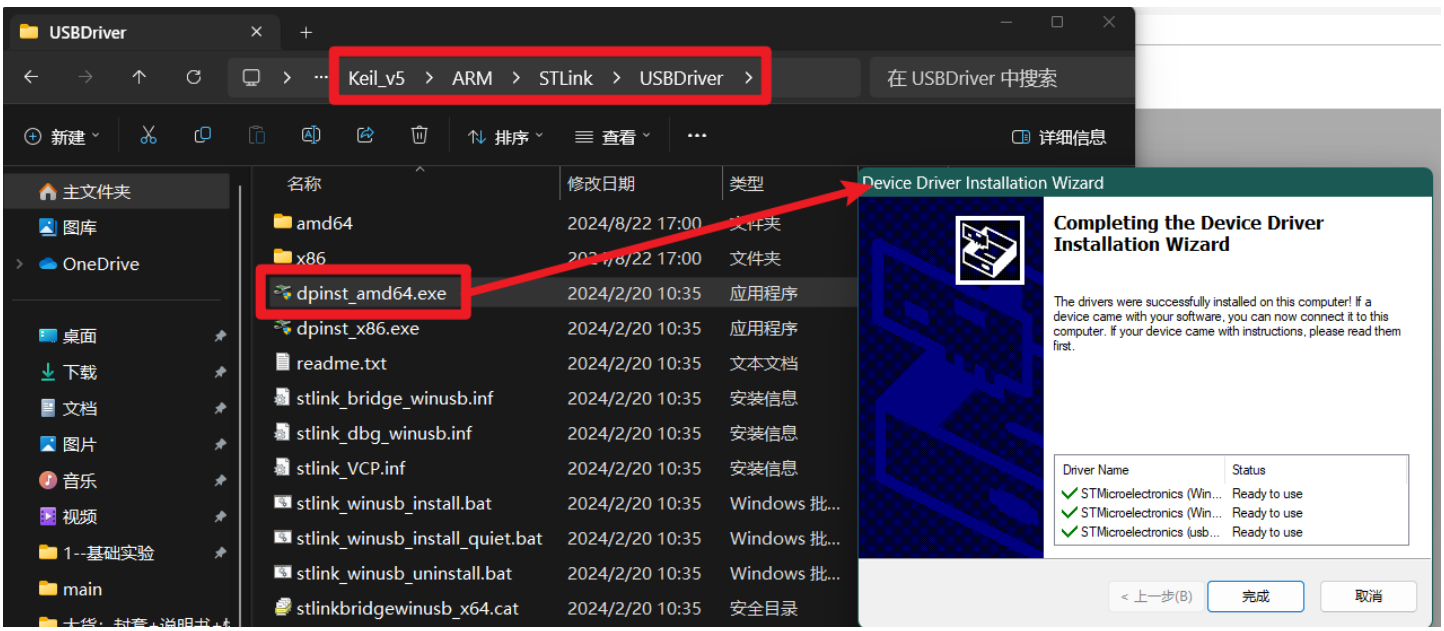
Open Keil5 to configure the compiler.

Step 4: Select the V.506 update 7 compiler. Open Keil5 and navigate to Options for Target -> Target -> ARM Compiler -> V.506 update 7 (build 960), then click OK to confirm the changes.



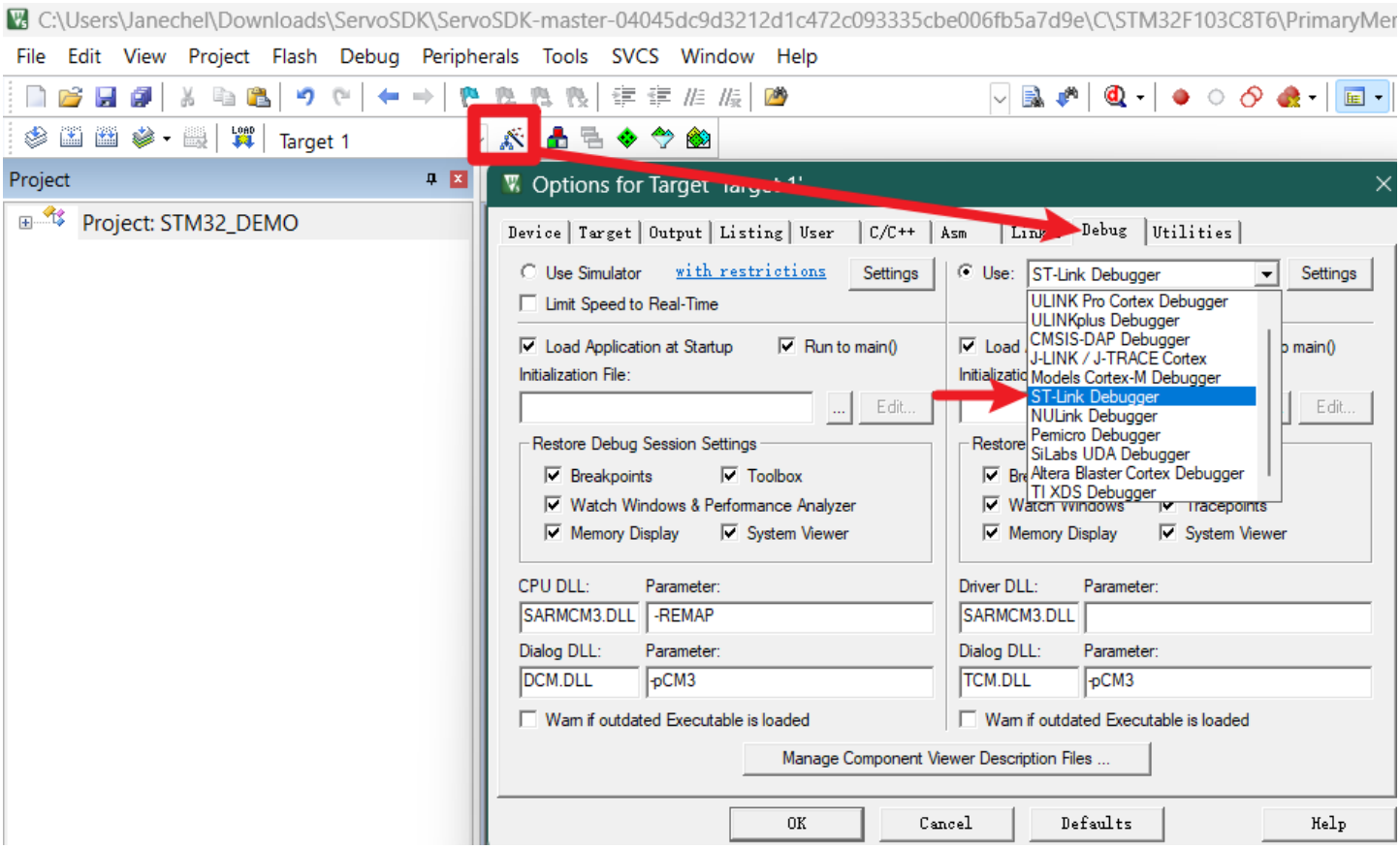
Select V.506 update 7(build 960)

Step 5: Install the ST-LINK driver. Open the local folder Keil5 -> ARM -> STLink -> USBDriver and run dpinst_amd64.exe, then follow the prompts to complete the installation.



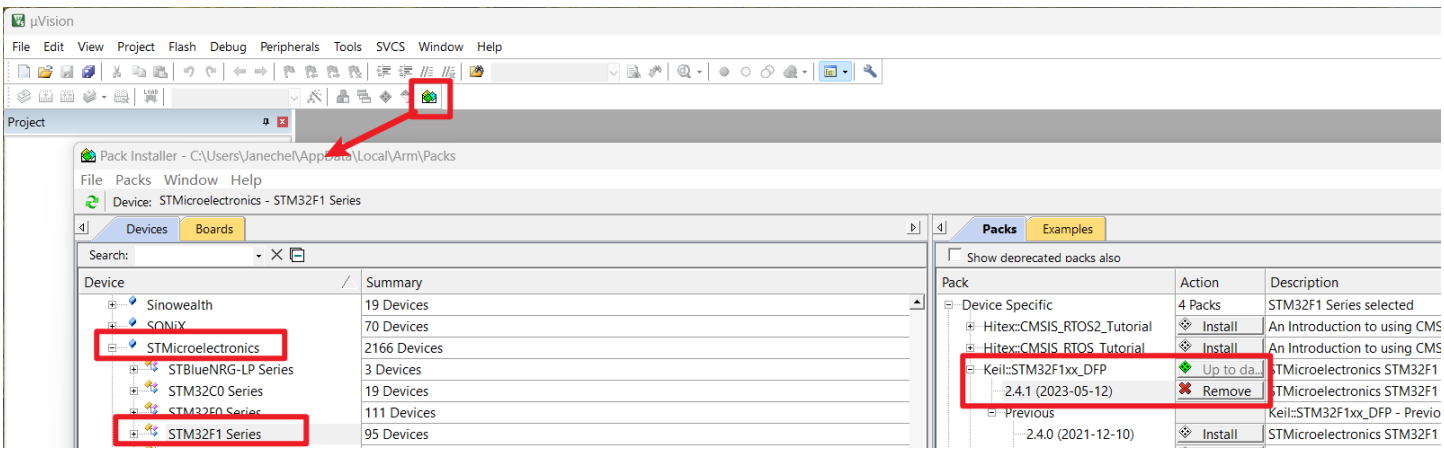
Note: If using a different debugger, the operation and selection are similar.

Step 6: Select the ST-Link Debugger. Open Keil5 and navigate to Options for Target -> Debug -> Use -> ST-Link Debugger, then click OK to confirm the changes.

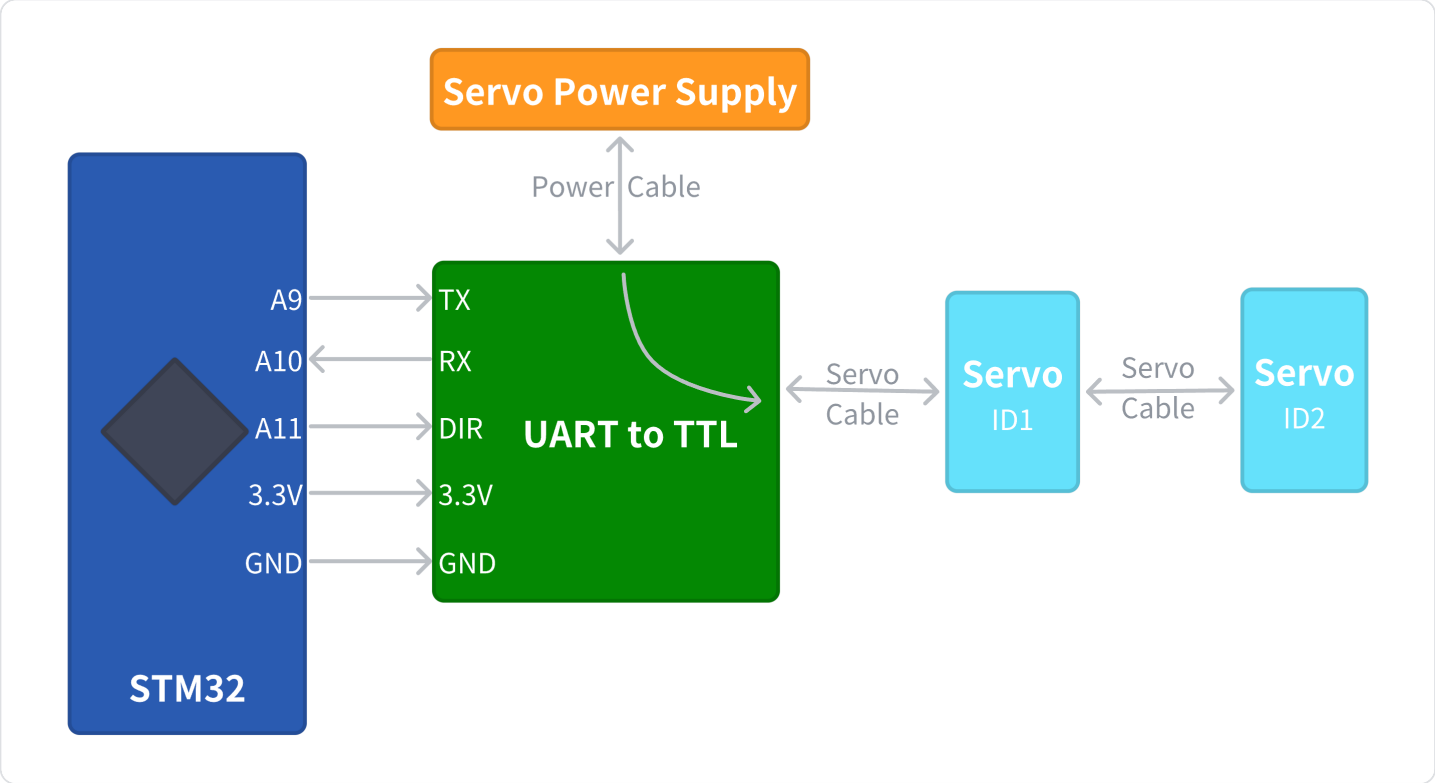


Select ST-Link Debugger

Step 7: Install the STM32F1xx_DFP chip package. Open the Keil5 chip packages, then select STMicroelectronics -> STM32F1 Series -> Keil::STM32F1xx_DFP -> 2.4.1, and click Install.



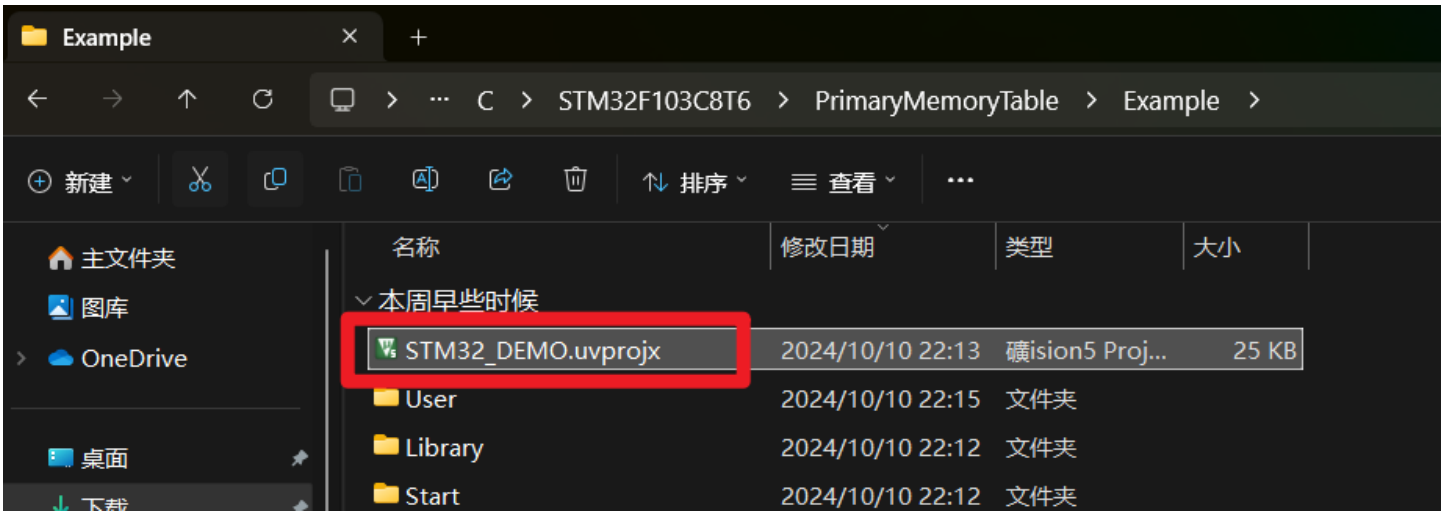
4.2.2 Hardware Connection



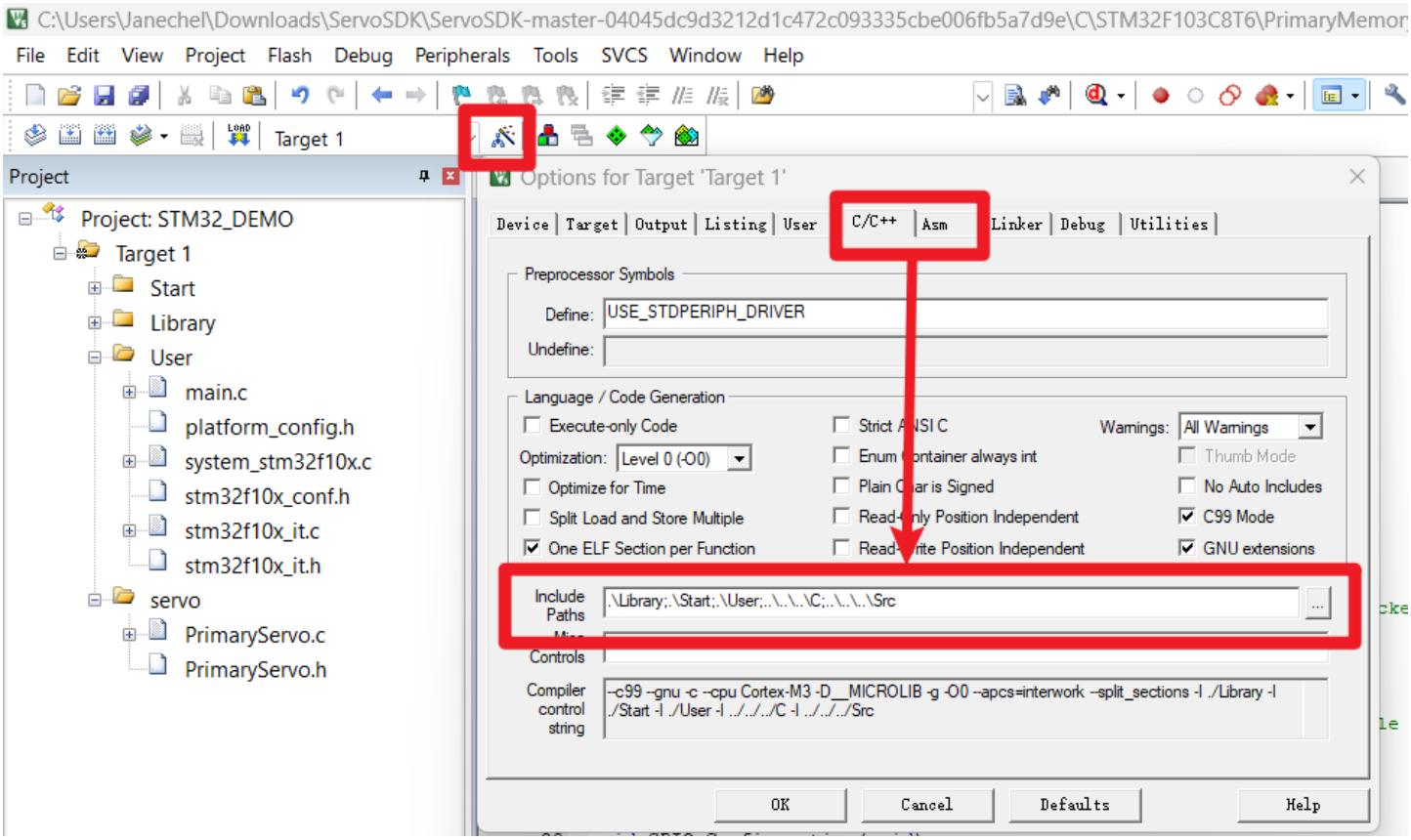
STM32 and Servo Connection Diagram

4.2.3 Run the Example

Step 1: Open the STM32_DEMO.uvprojx file. Take EM-2030 as an example. double-click to open the STM32_DEMO.vcxproj located in ServoSDK\C\STM32F103C8T6\PrimaryMemoryTable\Example.



Step 2: Replace the C/C++ and ASM including directories with the local Src path.



Step 3: Change the module value you want to test to 1. In the main.c file, change the module value from "0" to "1" in lines 7-16.

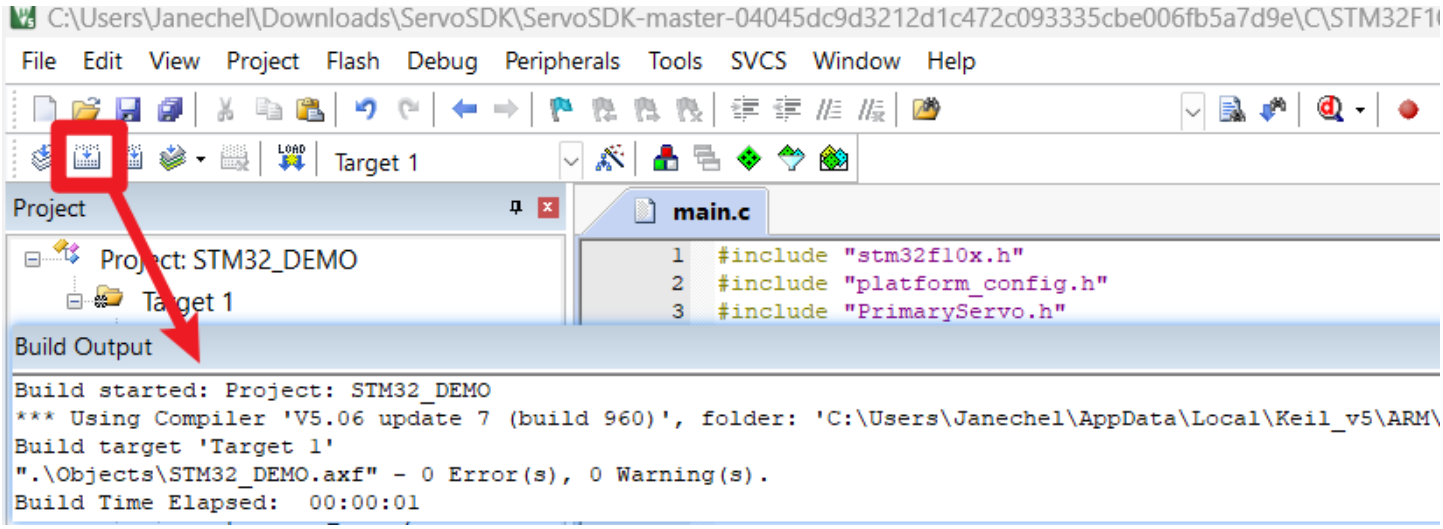
```
1  //Taking test the Ping module as an example
2  #define READ_TEST 0                    //Read Servo Data Test
3  #define WRITE_TEST 0                   //Write Servo Data Test
4  #define SYNC_WRITE_TEST 0              //Sync Write Test
5  #define PING_TEST 1                    //PING Instruction Test
6  #define FACTORY_RESET_TEST 0           //Factory Reset Test
7  #define PARAMETER_RESET_TEST 0         //Parameter Reset Test
8  #define REBOOT_TEST 0                  //Reboot Test
9  #define CALIBRATION_TEST 0             //Calibration Test
10 #define MODIFY_ID_TEST 0               //Change Known Servo ID Test
11 #define MODIFY_UNKNOWN_ID_TEST 0       //Change Unknown Servo ID Test
```

Step 4: Change the maximum number of servos for sync write and enable printing output as needed. In the PrimaryServo.h file, lines 4-5, MAX_SERVERS represents the upper limit of the maximum supported sync write servos. If the actual number of servos exceeds 20, adjust this value accordingly; it should be set to at least the actual number of servos. If printing output is not needed, change PRINTF_ENABLE to "0".

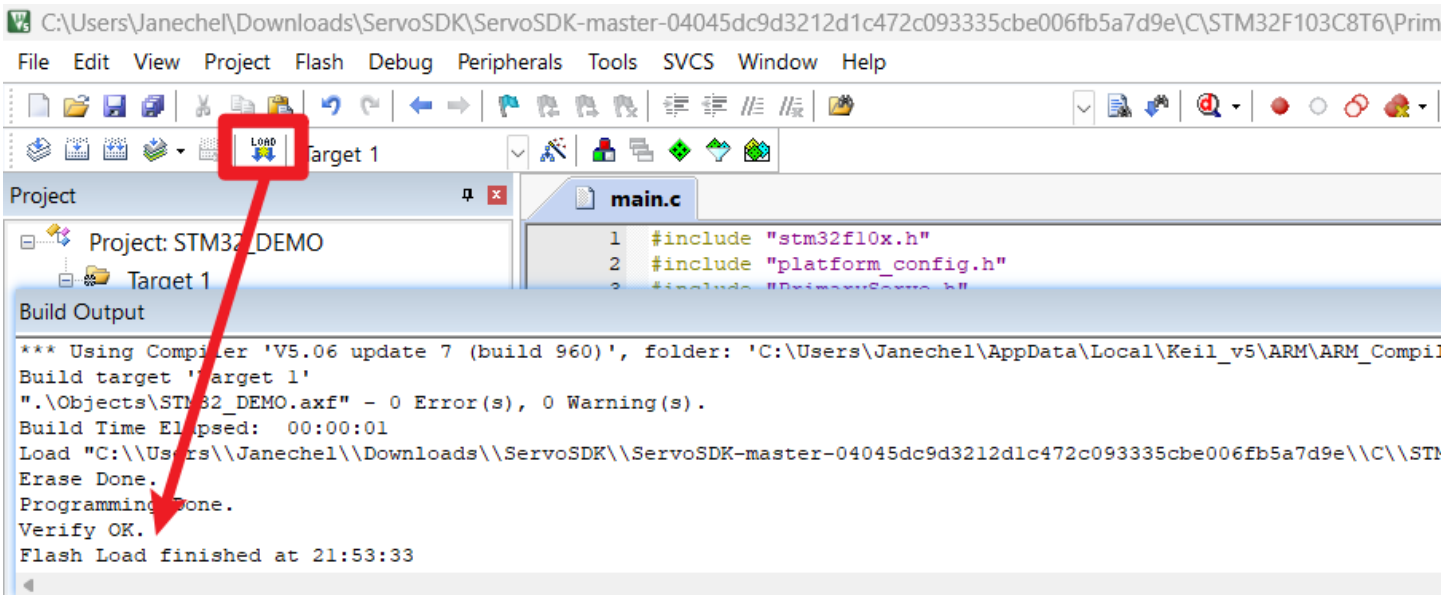
```
1  //Taking one servo and print enable as an example (no changes needed).
2  #define MAX_SERVERS 20                 //Maximum number of servos for sync write.
```

```
3 #define PRINTF_ENABLE 1            //Print output enable.
```

Step 5: Compile and download. Click Build to compile the project; once there are no errors, click Download to transfer to the microcontroller.

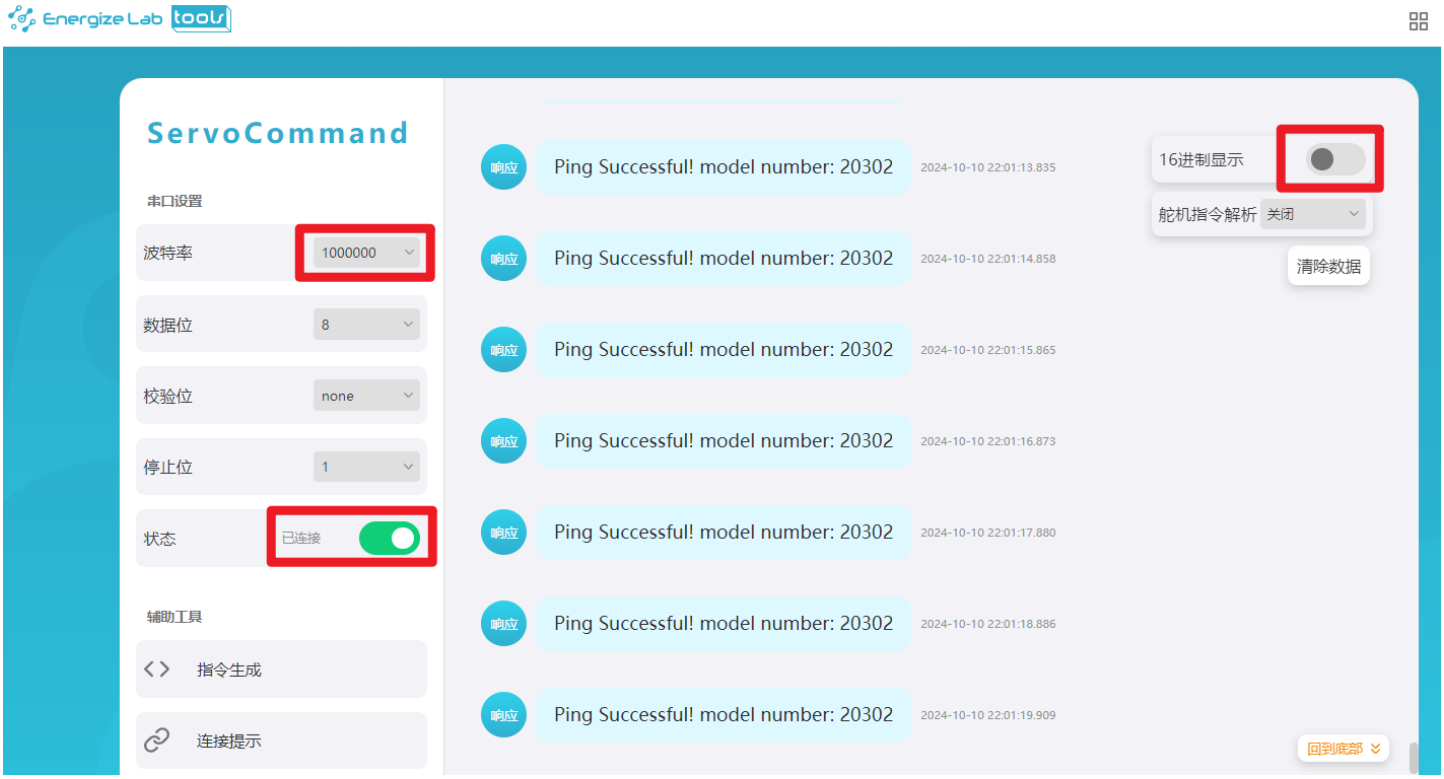


Compilation completed without errors



Download finished

Step 6: Run the program. Short press the Reset button on the microcontroller to restart the program. You can check the printed information using ServoCommand.

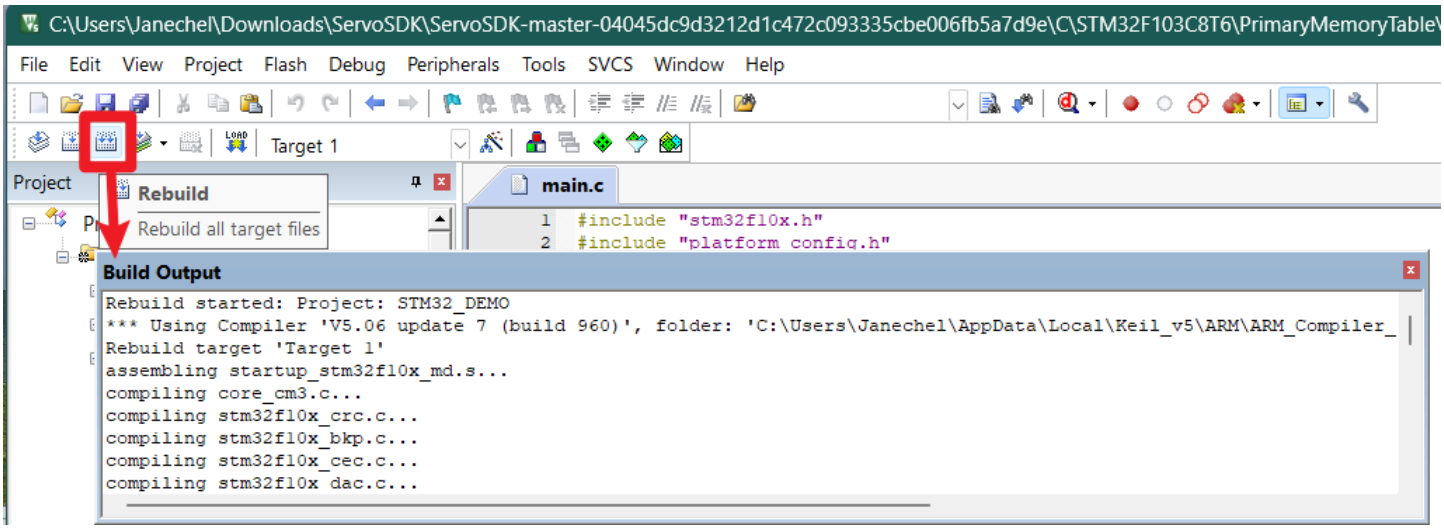


Taking test the Ping module as an example

4.2.4 Run the Demo

For details, refer to *section 4.1.4 Run the Demo*.

Note: After replacing the main file, it is recommended to click 'Rebuild' to recompile the project before downloading and running the program.



4.3 STM32 HAL Library

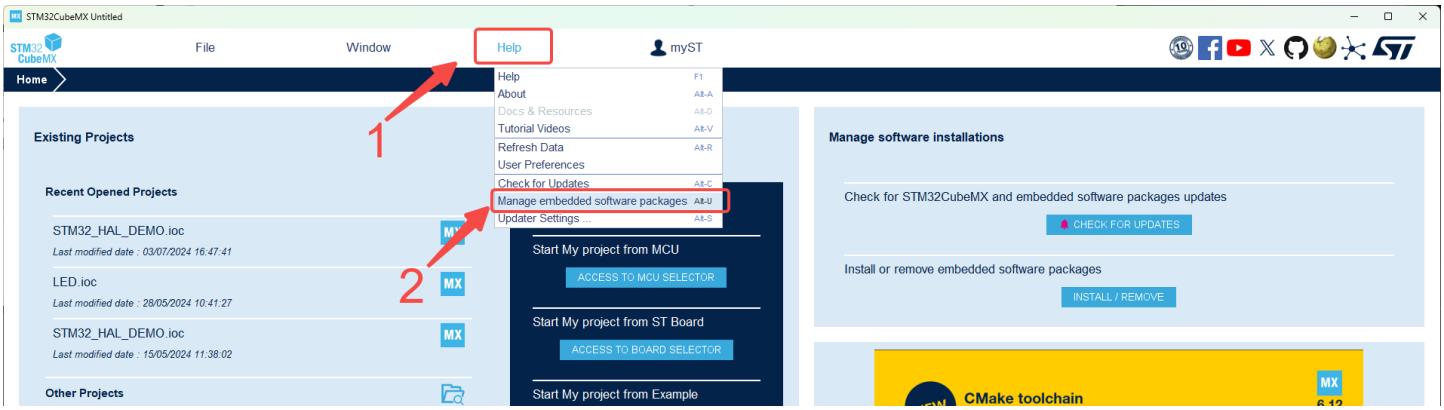
4.3.1 Environment Setup

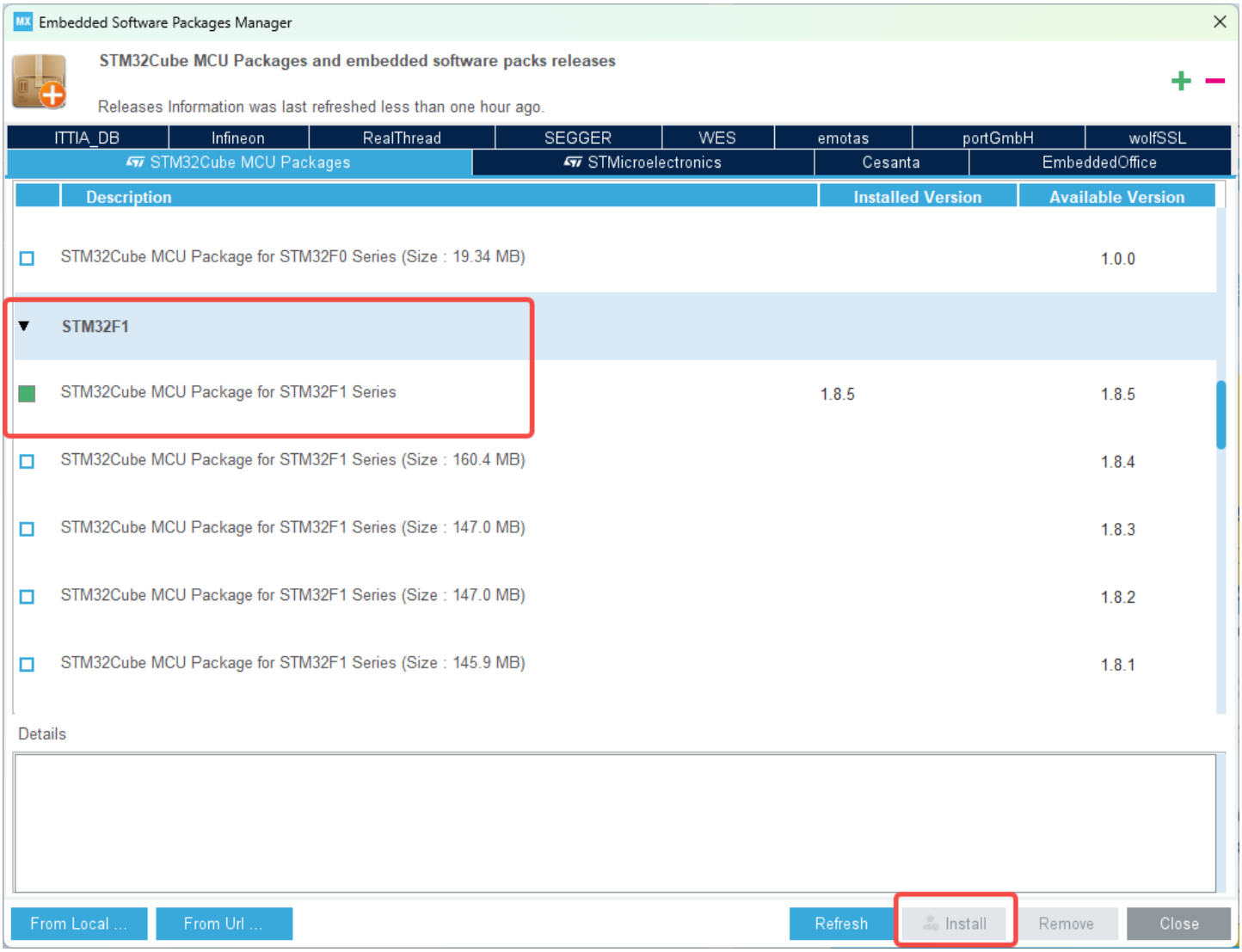
Step 1: Install and configure Keil5 MDK. For details, refer to *section 4.2.1 Environment Setup*.

Step 2: Install STM32CubeMX. Visit https://www.st.com.cn/zh/development-tools/stm32cubemx.html#get-software, select 'STM32CubeMX-Win', and follow the prompts to

download and install.

Step 3: Install the STM32F1 chip package. Open STM32CubeMX, then navigate to Help -> Manage Embedded Software Packages -> STM32F1 -> Latest Version, and click Install.





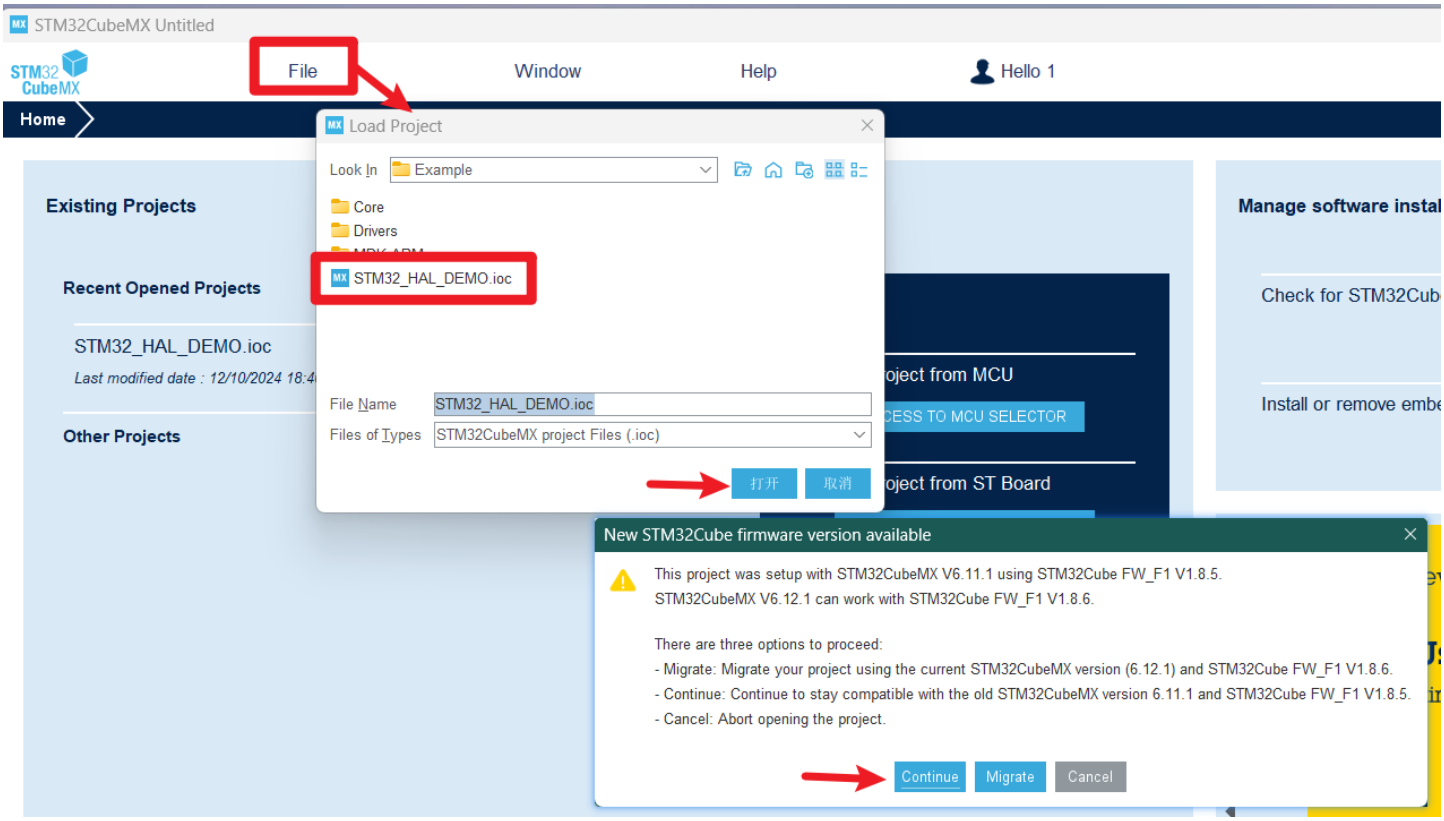
4.3.2 Hardware Connection

For details, refer to *section 4.2.2 Hardware Connection* for details.

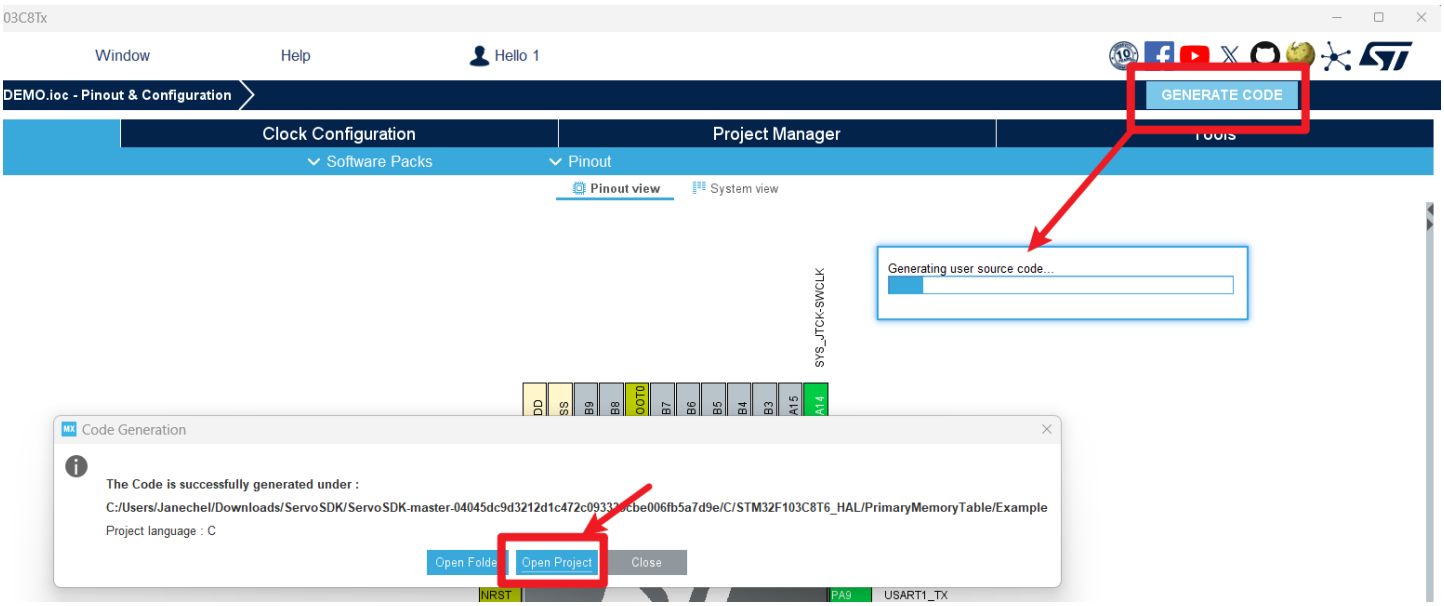
4.3.3 Run the Example

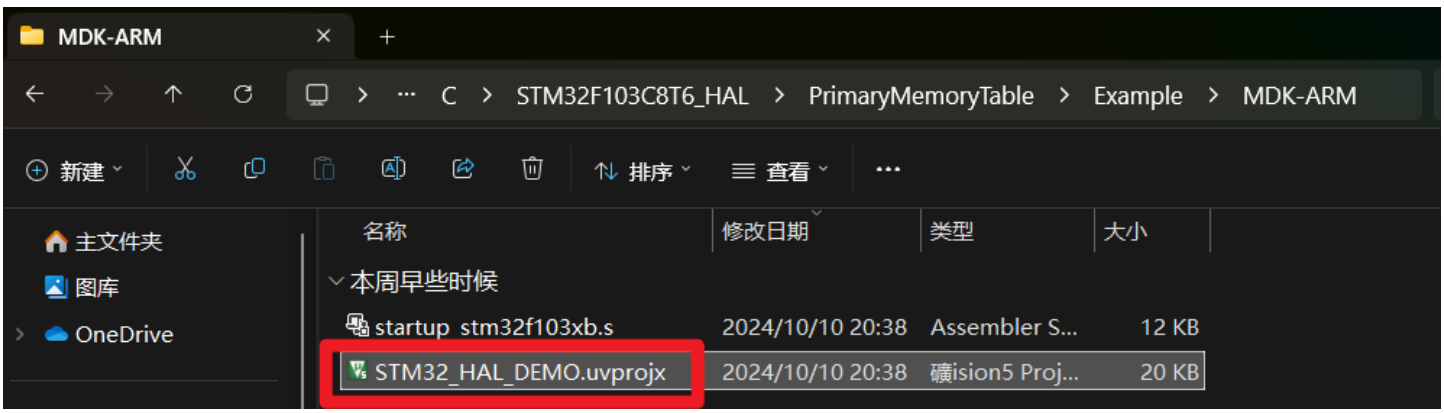
Note: When using this project for the first time, it is recommended to open the project file with STM32CubeMX to avoid encountering the "Unknown compiler" error after compilation.

Step 1: Open the STM32_HAL_DEMO.ioc file. Launch STM32CubeMX and select File -> Load Project -> STM32_HAL_DEMO.ioc -> Open -> Continue -> GENERATE CODE -> Open Project.



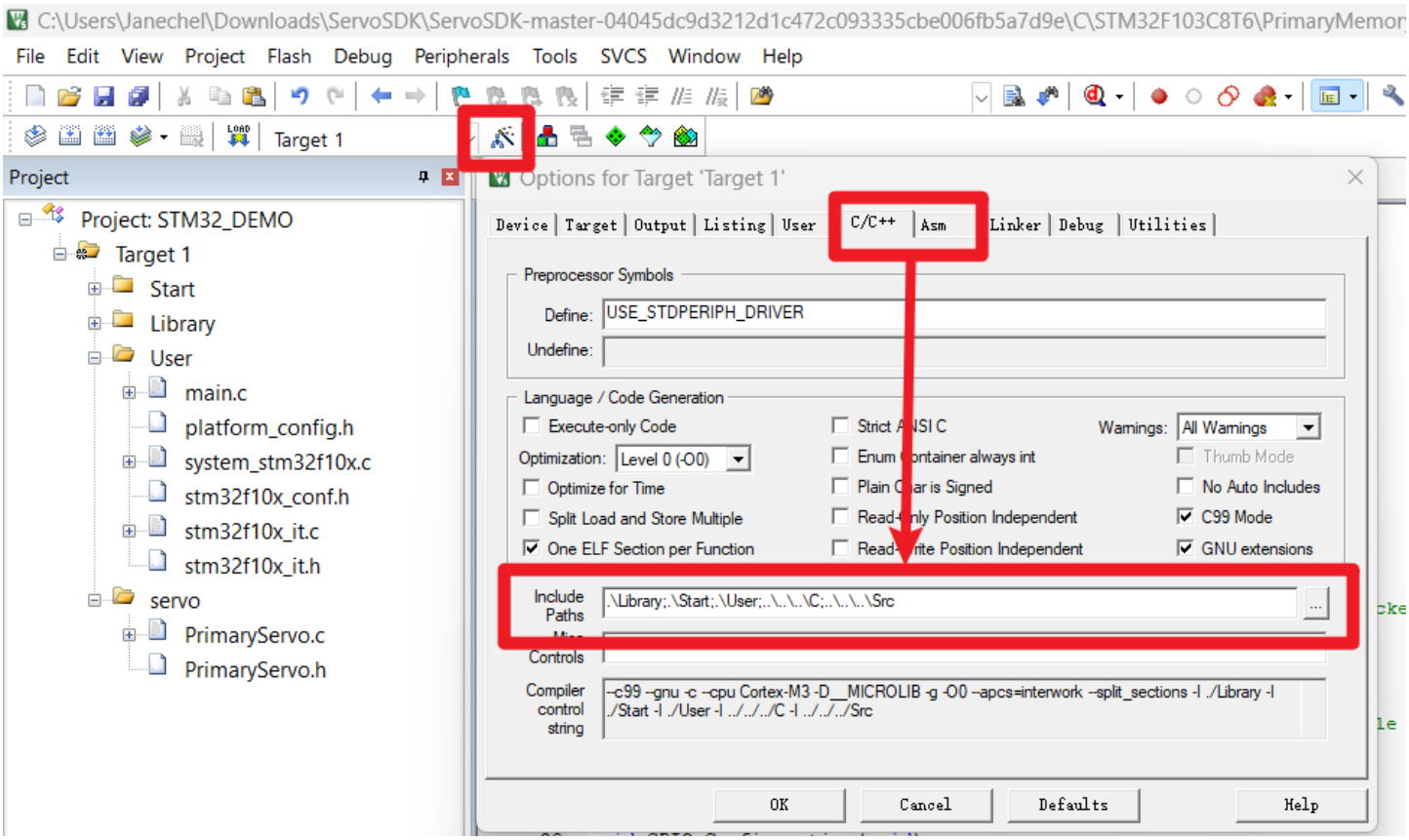
Taking EM-2030 as an example, STM32_HAL_DEMO.ioc in C\STM32F103C8T6_HAL\PrimaryMemoryTable\Example\MDK-ARM





STM32_HAL_DEMO.uvprojx file can be directly opened, if this is not the first time it has been accessed.

Step 2: Replace the C/C++ and ASM including directories with the local Src path.



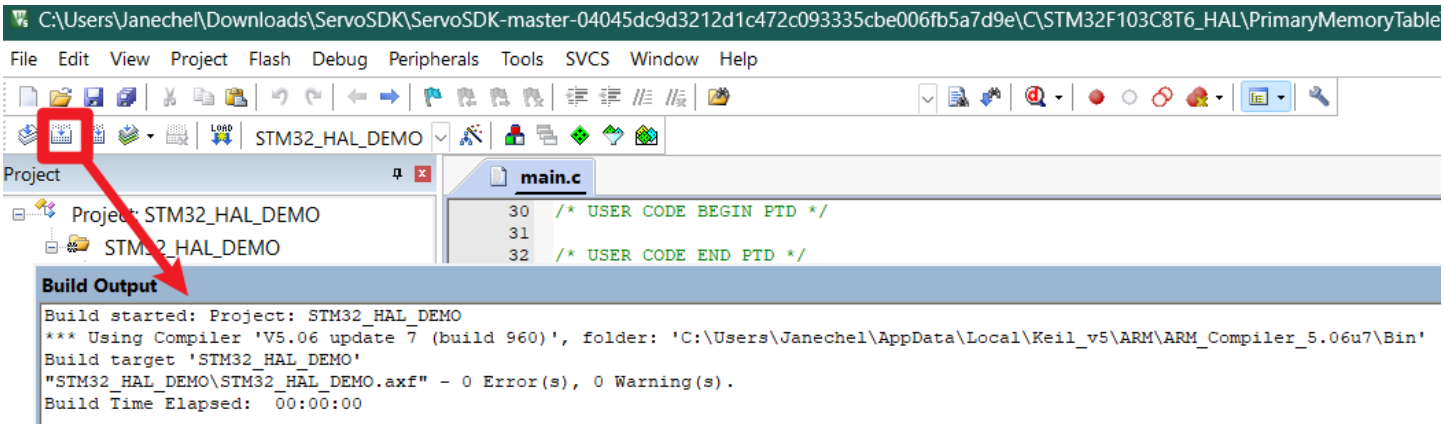
Step 3: Change the module value you want to test to 1. In the main.c file, modify the module value from "0" to "1" in lines 61-70.

```
1  //Taking test the Ping module as an example
2  #define READ_TEST 0                  //Read Servo Data Test
3  #define WRITE_TEST 0                 //Write Servo Data Test
4  #define SYNC_WRITE_TEST 0            //Sync Write Test
5  #define PING_TEST 1                  //PING Instruction Test
6  #define FACTORY_RESET_TEST 0         //Factory Reset Test
7  #define PARAMETER_RESET_TEST 0       //Parameter Reset Test
8  #define REBOOT_TEST 0                //Reboot Test
9  #define CALIBRATION_TEST 0           //Calibration Test
10 #define MODIFY_ID 0                  //Change Known Servo ID Test
11 #define MODIFY_UNKNOWN_ID 0          //Change Unknown Servo ID Test
```

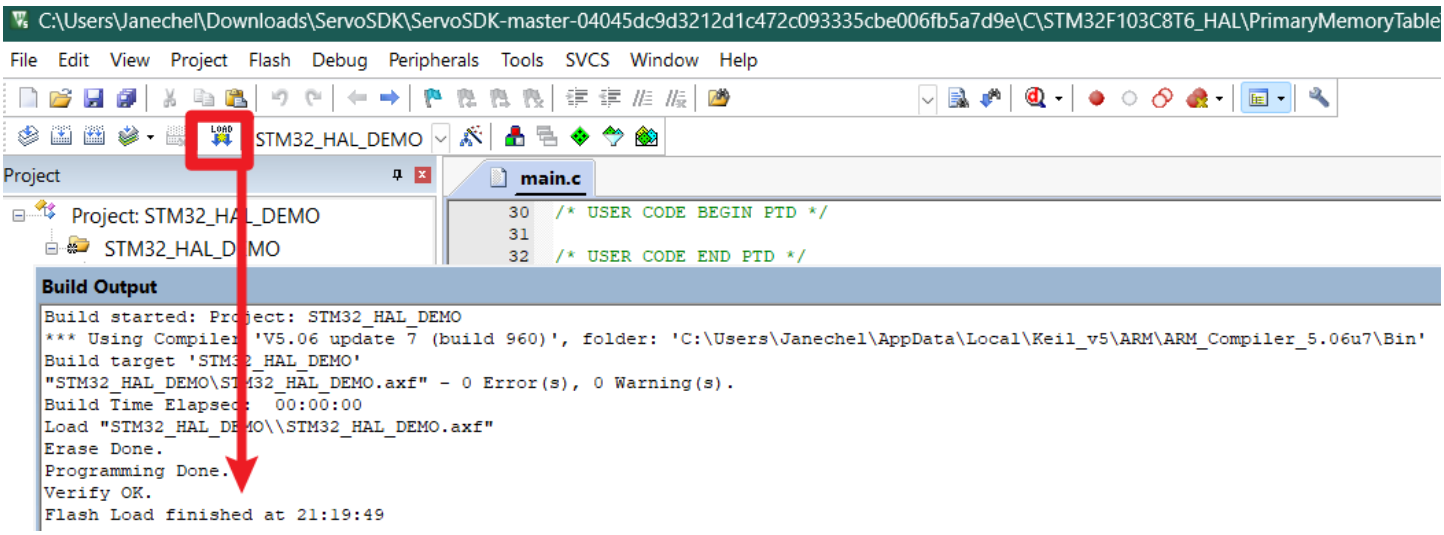
Step 4: Change the maximum number of servos for sync write and enable printing output as needed. In the PrimaryServo.h file, lines 4-5, MAX_SERVERS represents the upper limit of the maximum supported sync write servos. If the actual number of servos exceeds 20, adjust this value accordingly; it should be set to at least the actual number of servos. If printing output is not needed, change PRINTF_ENABLE to "0".

```
//Taking one servo and print enable as an example (no changes needed).
#define MAX_SERVERS 20              //Maximum number of servos for sync write.
#define PRINTF_ENABLE 1             //Print output enable.
```

Step 5: Compile and download. Click Build to compile the project; once there are no errors, click Download to transfer to the microcontroller.

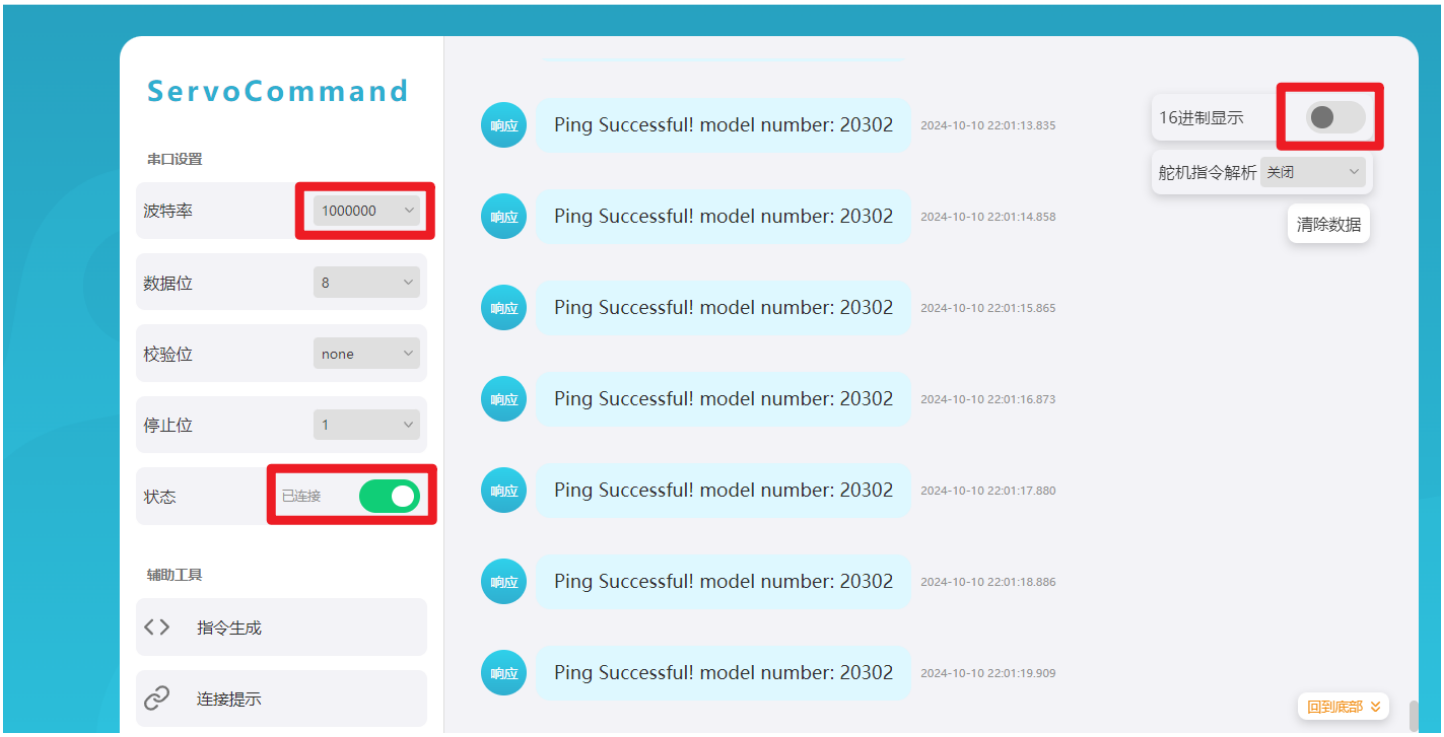


Compilation completed without errors



Download finished

Step 6: Run the program. Short press the Reset button on the microcontroller to restart the program. You can check the printed information using ServoCommand.

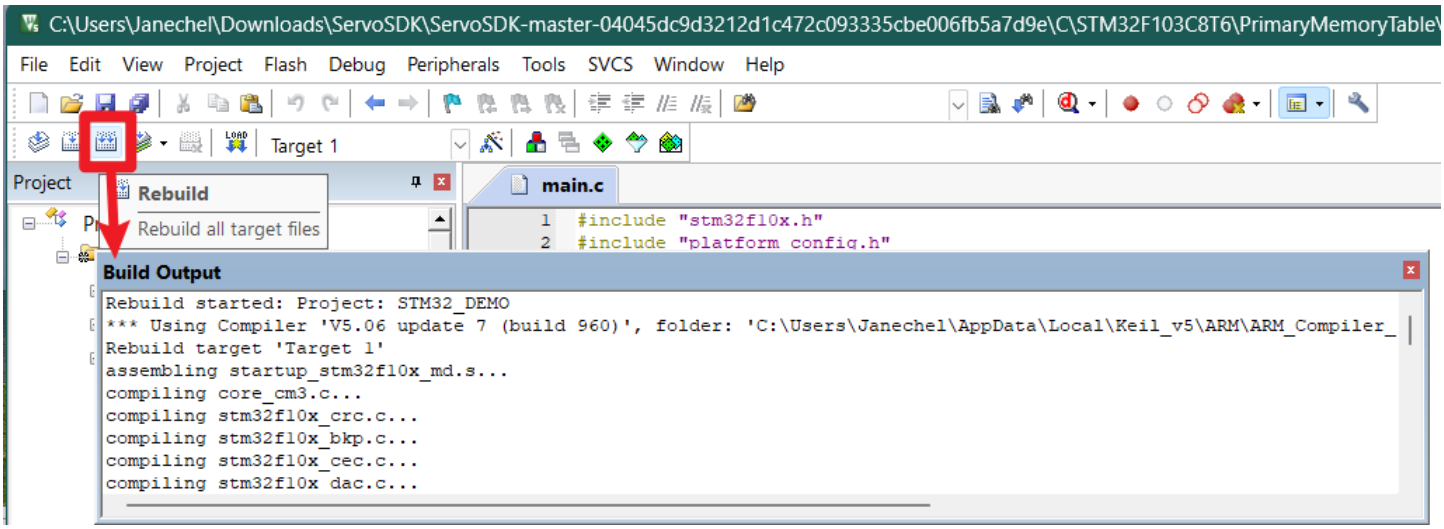


Taking test the Ping module as an example

4.3.4 Run the Demo

For details, refer to *section 4.1.4 Run the Demo*.

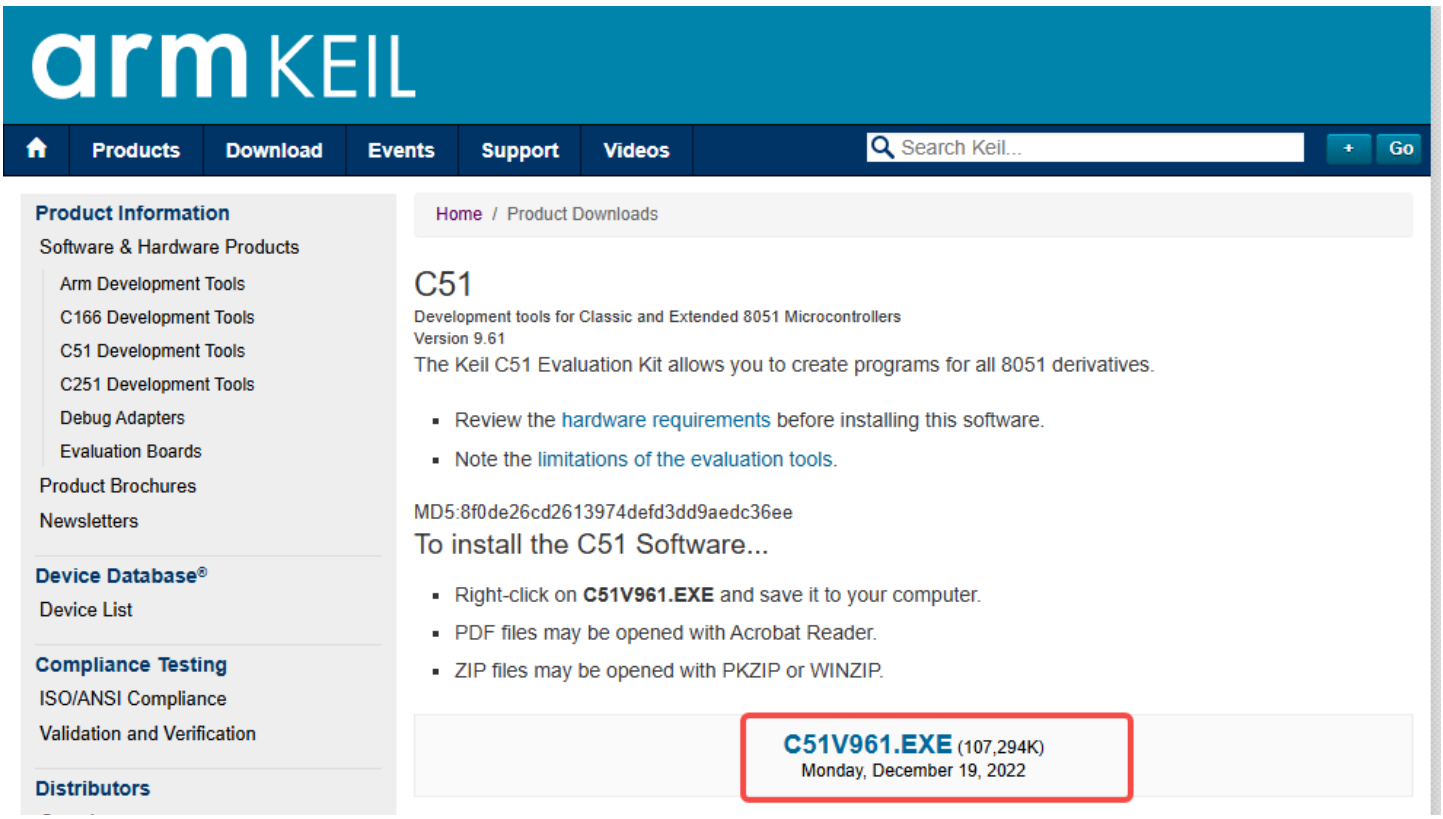
Note: After replacing the main file, it is recommended to click 'Rebuild' to recompile the project before downloading and running the program.



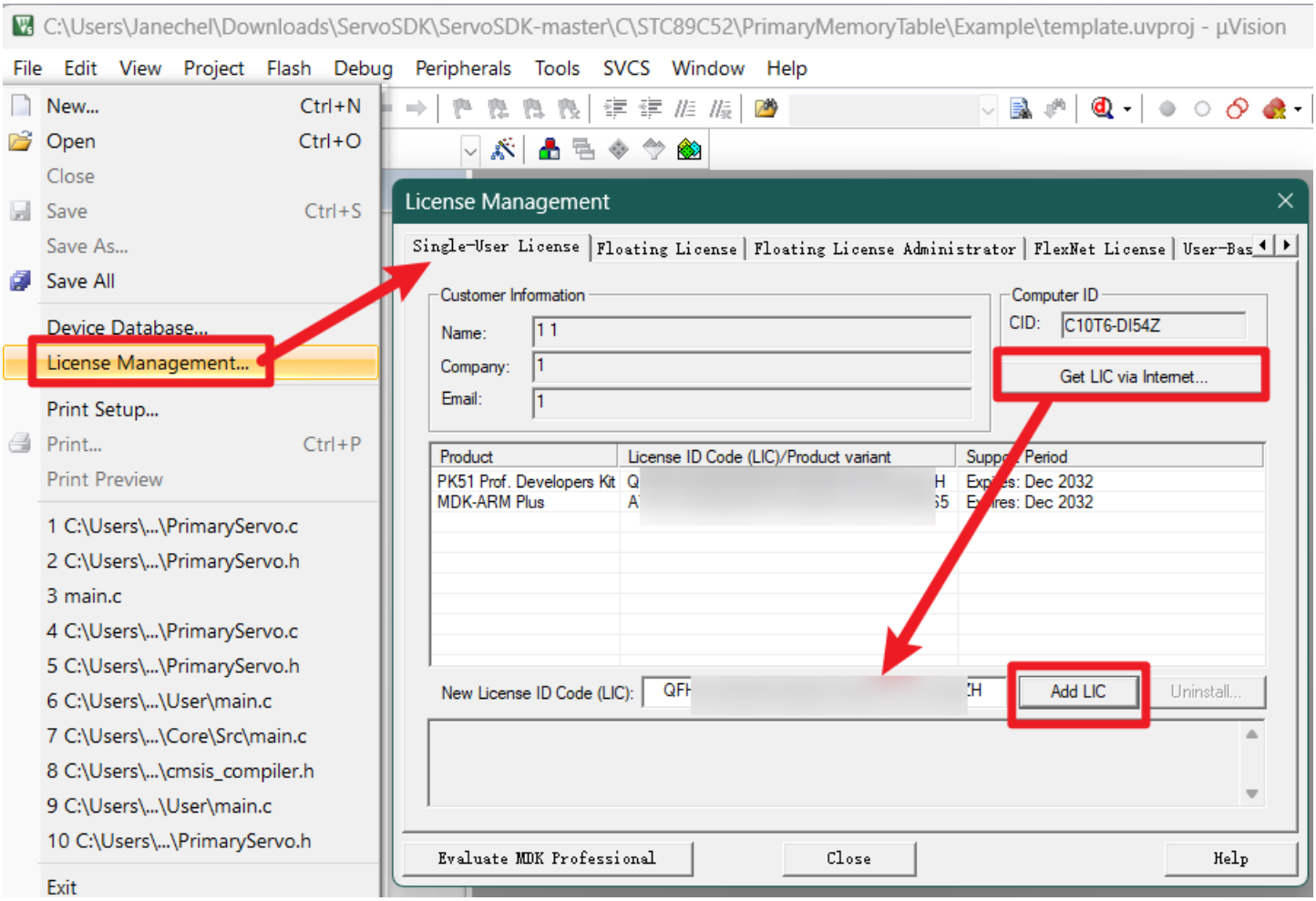
4.4 STC89C52

4.4.1 Environment Setup

Step 1: Install Keil5 C51. Visit http://www.keil.com/download/product/, select 'C51', and follow the prompts to download and install.

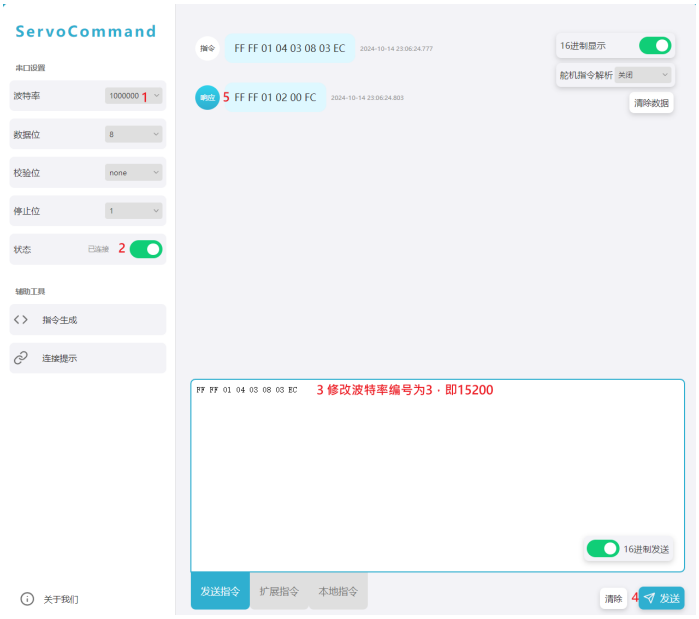


Step 2: Register Keil5 C51. Obtain the license key through official channels, then click File -> License Management -> Single-User, enter the license key, and click Add LIC to complete the registration.

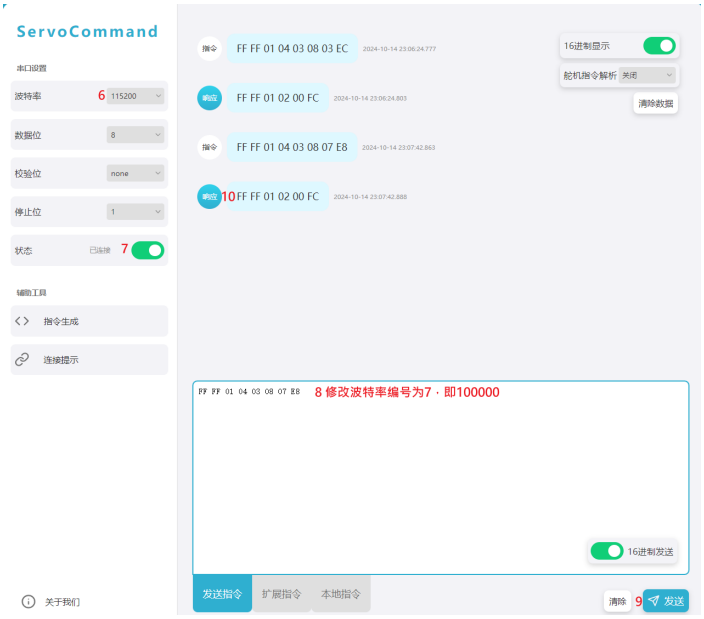


Step 3: Install the download tool. For example, to install PZ-ISP.exe, visit http://www.prechin.cn/gongsixinwen/208.html, click on the link for the 51 Microcontroller Series - PZ - 2&3&4 (A2&A3&A4) Development Board, and then select 5 - Development Tools -> 3 - Programming Software -> PZ-ISP -> PZ-ISP.exe.

Step 4: Change the servo baud rate. Since the microcontroller hardware oscillator frequency used for this test is low, the highest baud rate that can be provided is 115200. Therefore, before testing, the baud rate for servos ID1 and ID2 needs to be changed to 3 (which corresponds to 115200). You can use ServoStudio or ServoCommand to make this change; please refer to the respective user manual for details.

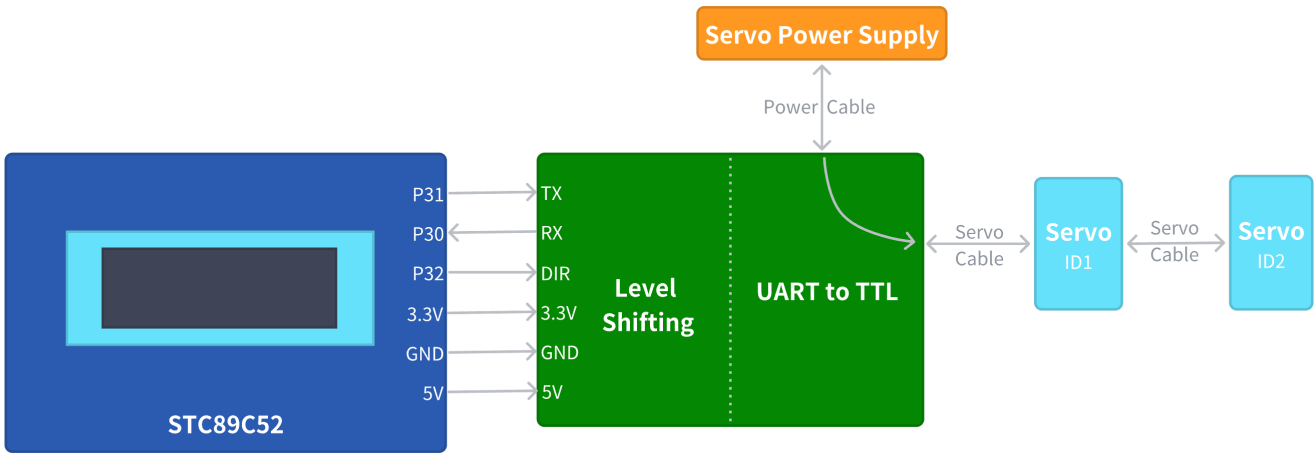


Change the baud rate of ID1 to 3



Change the baud rate of ID1 back to 7

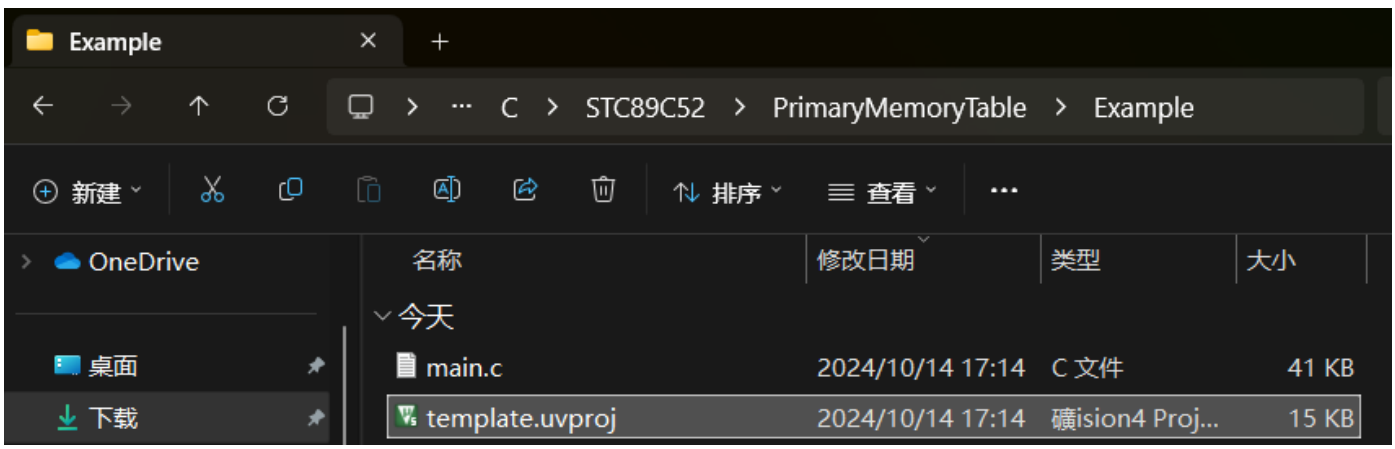
4.4.2 Hardware Connection



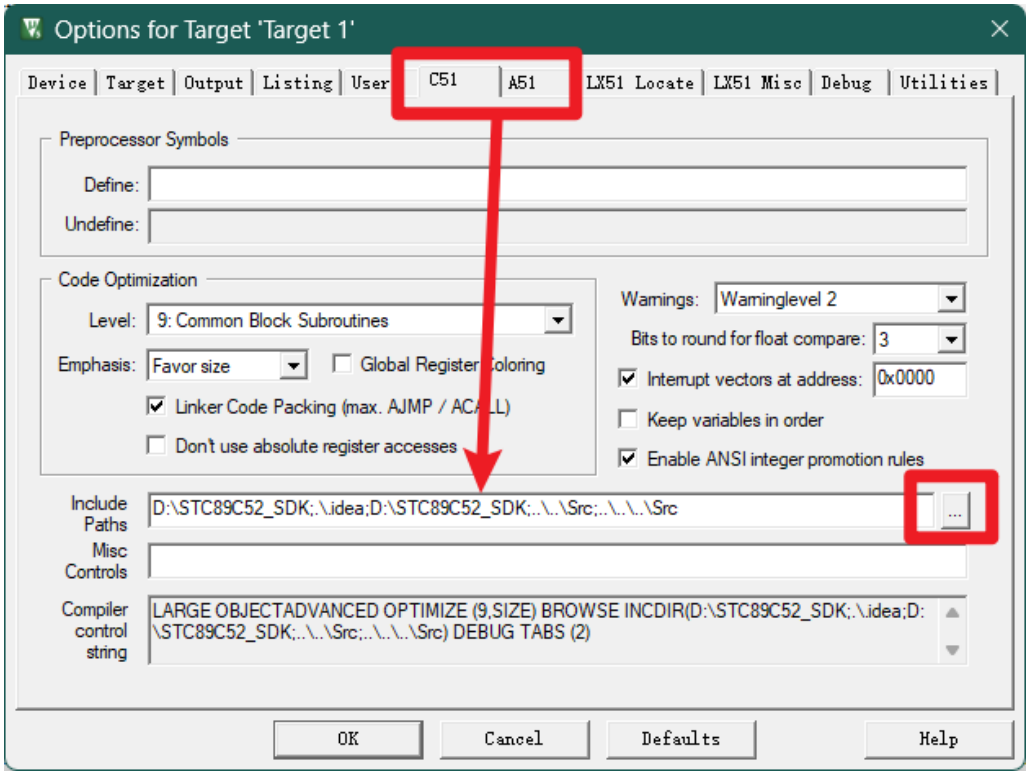
STC89C52 and Servo Connection Diagram

4.4.3 Run the Example

Step 1: Open the template.uvproj file. Taking EM-2030 as an example, double-click to open the template.uvproj located in ServoSDK\C\STC89C52\PrimaryMemoryTable\Example.



Step 2: Replace the C51 and A51 including directories with the local Src path.



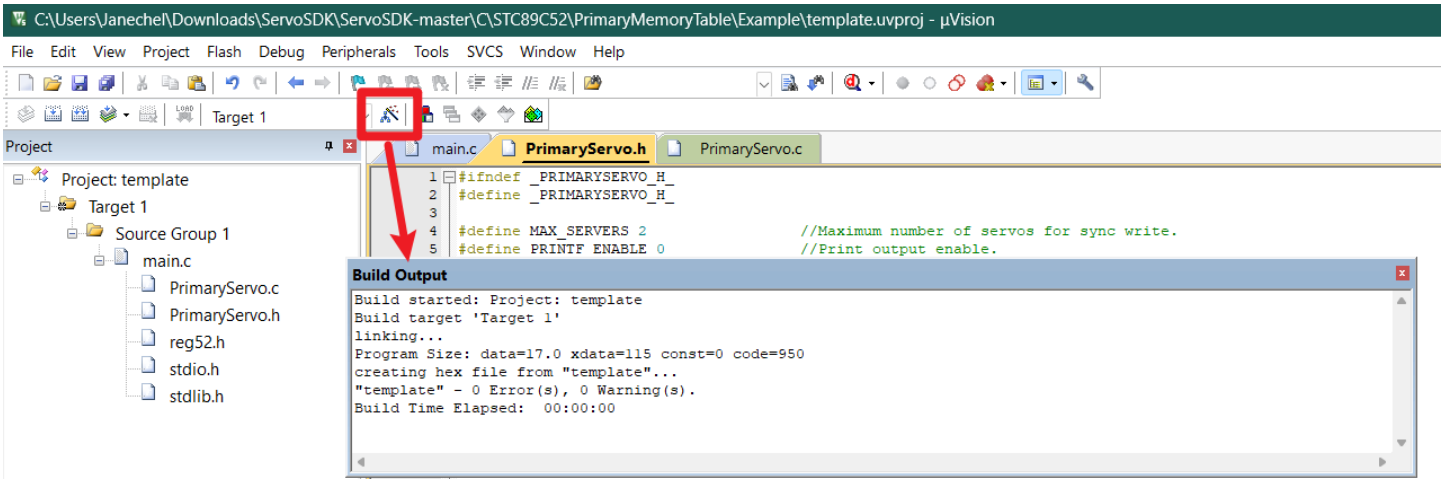
Step 3: Change the module value you want to test to 1. In the main.c file, modify the module value from "0" to "1" in lines 4-13.

```
1  //Taking test the SYNC WRITE module as an example
2  #define READ_TEST 0                  //Read Servo Data Test
3  #define WRITE_TEST 0                 //Write Servo Data Test
4  #define SYNC_WRITE_TEST 1            //Sync Write Test
5  #define PING_TEST 0                  //PING Instruction Test
6  #define FACTORY_RESET_TEST 0         //Factory Reset Test
7  #define PARAMETER_RESET_TEST 0       //Parameter Reset Test
8  #define REBOOT_TEST 0                //Reboot Test
9  #define CALIBRATION_TEST 0           //Calibration Test
10 #define MODIFY_ID 0                  //Change Known Servo ID Test
11 #define MODIFY_UNKNOWN_ID 0          //Change Unknown Servo ID Test
```

Step 4: Change the servo output and quantity configuration. Due to the hardware limitations of the STC89C52 test board, change the MAX_SERVERS in PrimaryServo.h (lines 4-5) to 2, and set PRINTF_ENABLE to 0 (disable).

```
1 //Taking one servo and print disabled as an example (no changes needed).
2 #define MAX_SERVERS 2                //Maximum number of servos for sync write.
3 #define PRINTF_ENABLE 0              //Print output enable.
```

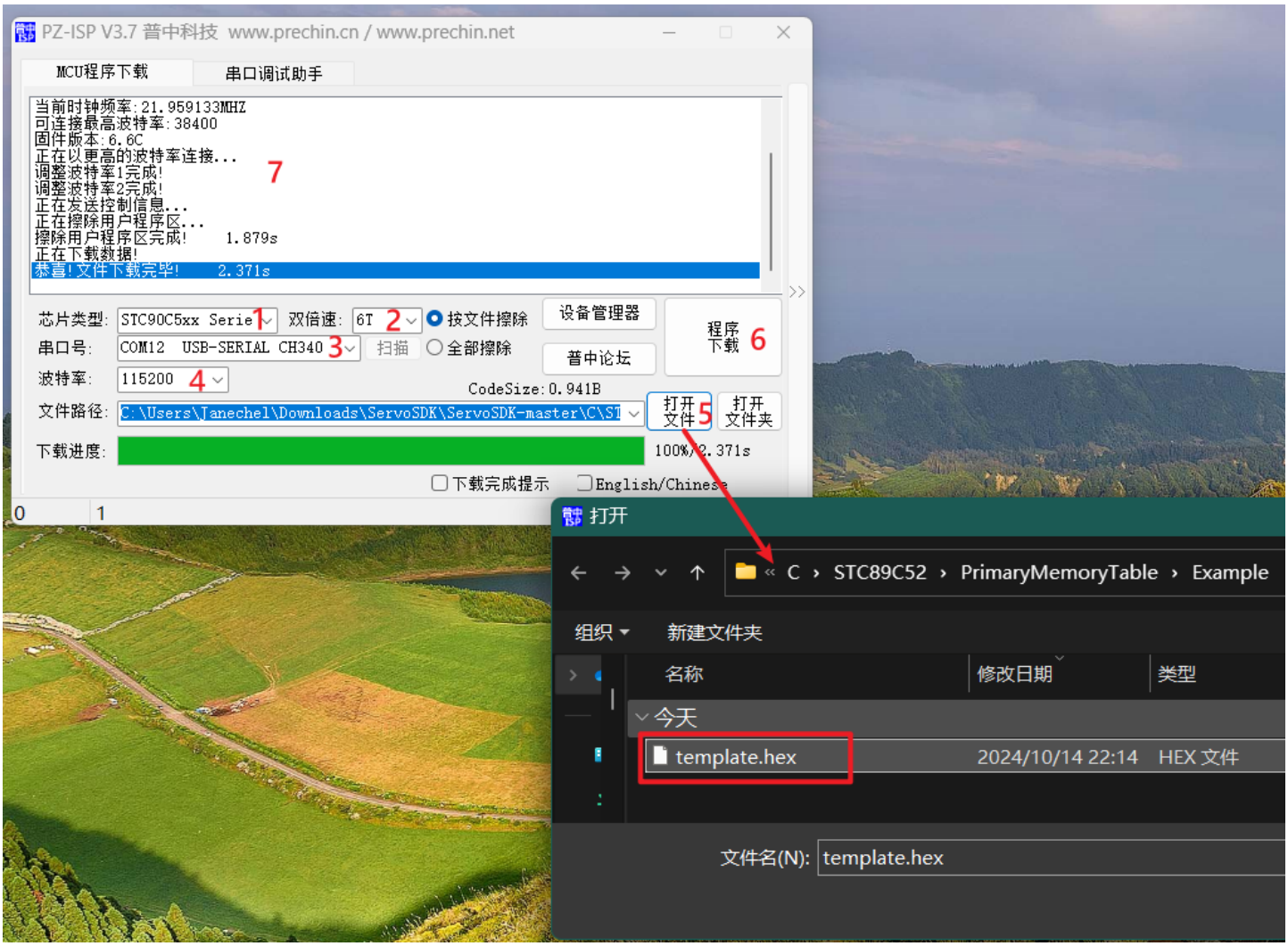
Step 5: Compile the project. Click Build to compile, ensuring there are no errors.



Compilation completed without errors

Step 6: Download the template.hex file. For example, with PZ-ISP.exe, use double-speed 6T for programming.

Note: To avoid write timeout issues, disconnect the wire connected to pin P31 before clicking "Program Download." After the file download is complete, reconnect the wire to pin P31.



Step 7: Run the program. Short press the Reset button on the microcontroller to restart the program. Since printing is disabled, you can only verify successful operation by observing the

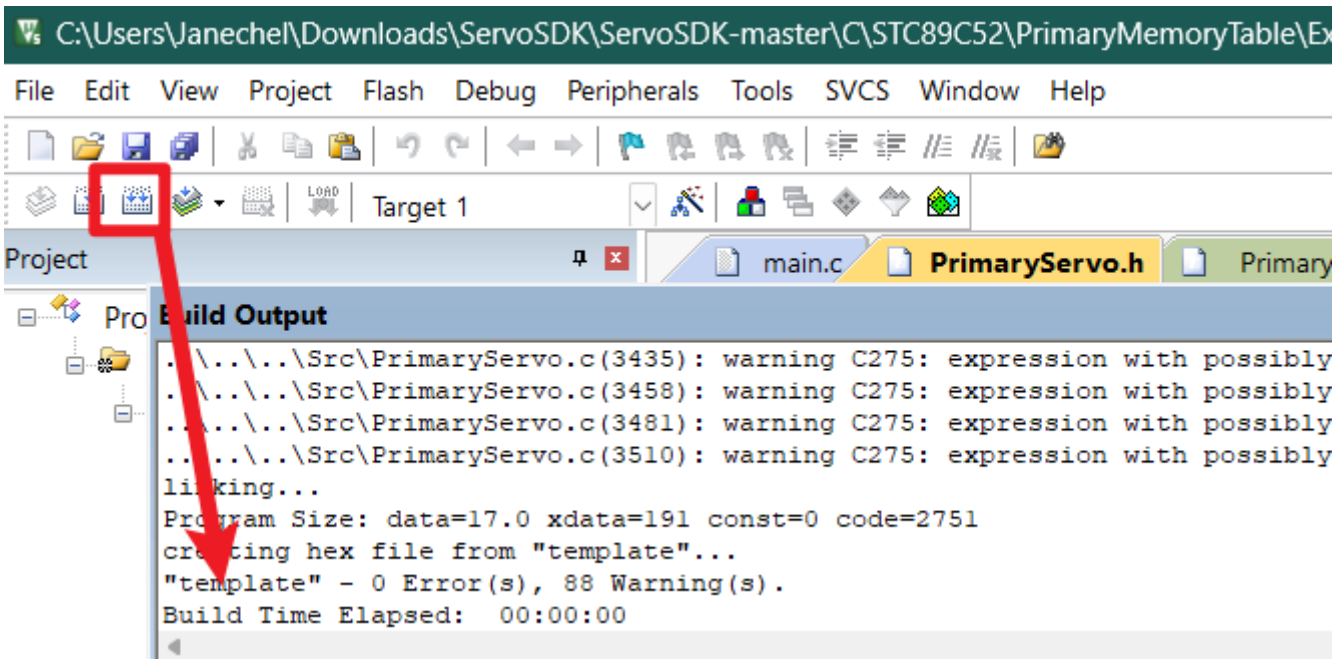
servo's movement.

Note: After testing, remember to change the baud rate back to 7 (which corresponds to 1,000,000) to avoid affecting servo testing on other platforms.

4.4.4 Run the Demo

For details, refer to *section 4.1.4 Run the Demo*.

Note: After replacing the main file, click 'Rebuild' to recompile the project before downloading and running the program.



4.4.5 FAQ

- Printing output enabled causing the microcontroller to hang
  - Since the 51 microcontroller only has one serial port for communication with the servo and no extra port for printing output, PRINTF_ENABLE needs to be set to 0 to disable it.
- PZ-ISP not set to double-speed 6T
  - Due to hardware limitations, this 51 microcontroller cannot reach a baud rate of 1,000,000; therefore, double-speed 6T is enabled to achieve the maximum baud rate of 115,200.
- PZ-ISP download failure
  - Try disconnecting the wire from pin P31 before downloading again.

5. C++

5.1 Windows

5.1.1 Development Environment Setup

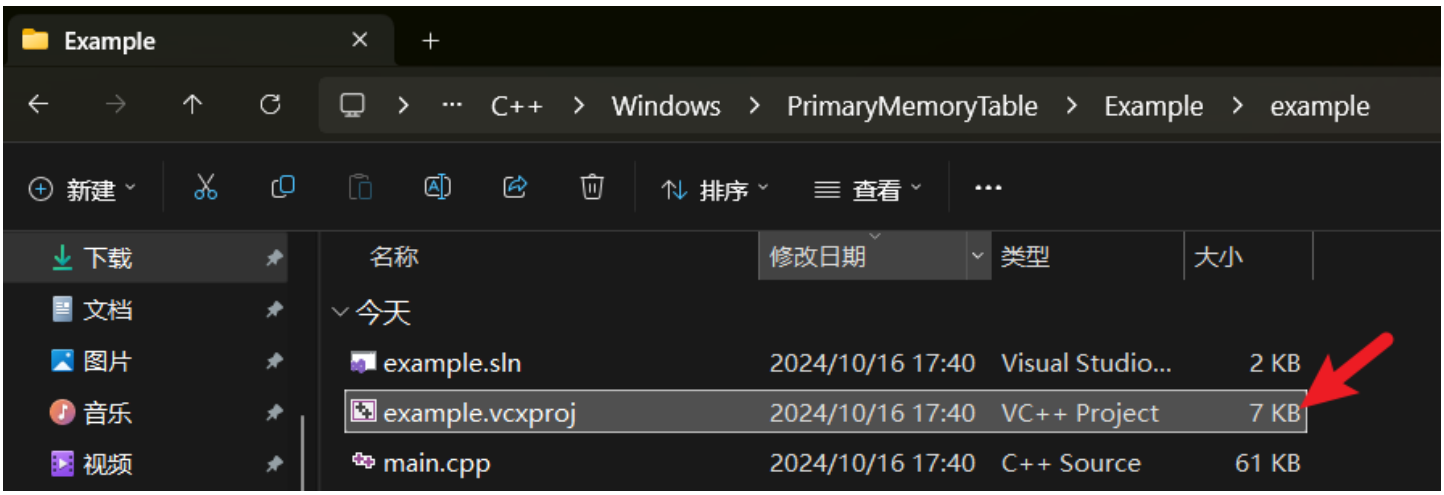
For details, refer to section *4.1.1 Environment Setup.*

5.1.2 Hardware Connection

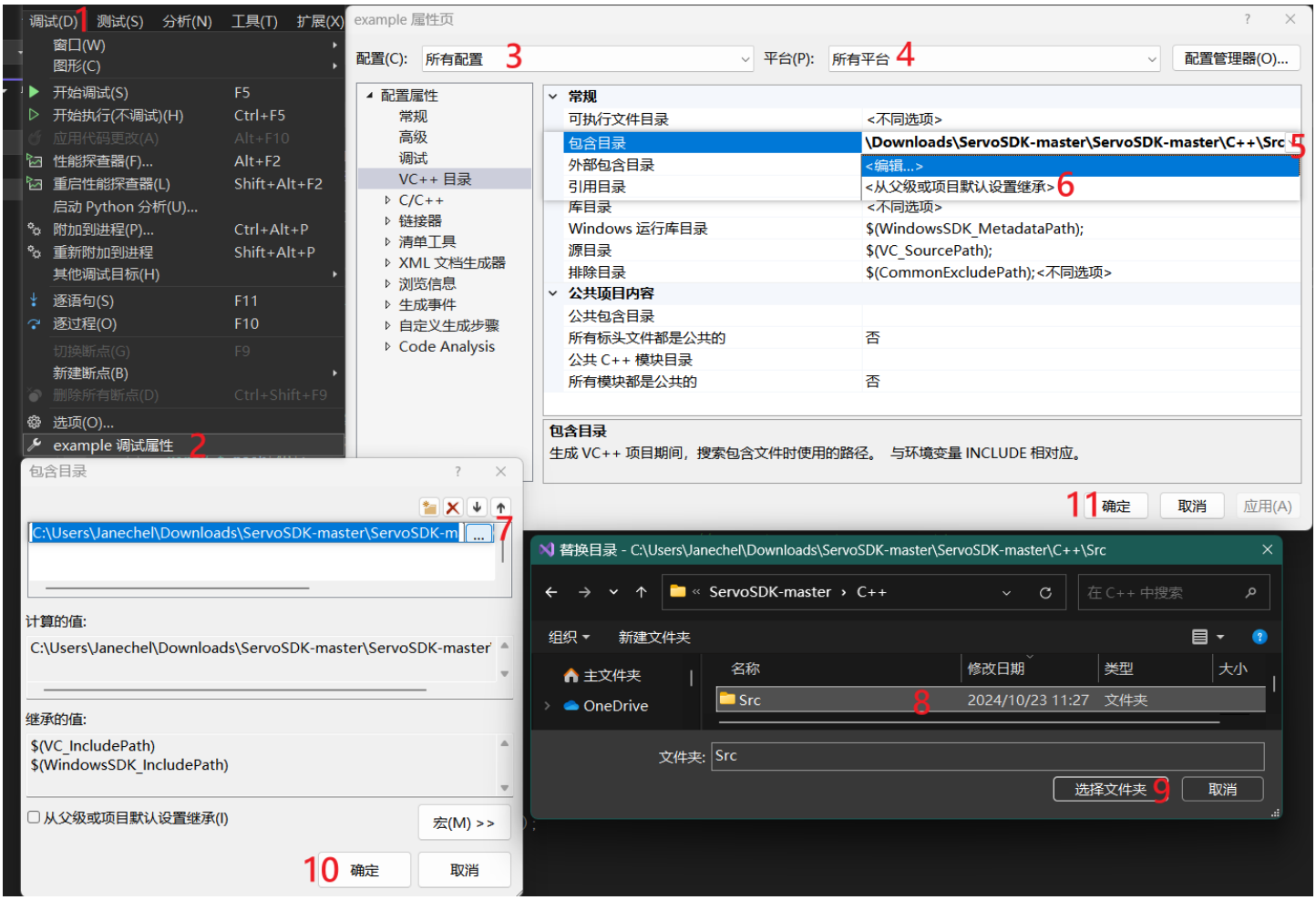
For details, refer to section *4.1.2 Hardware Connection*.

5.1.3 Run the Example

Step 1: Open the example.vcxproj file. Taking EM-2030 as an example, double-click to open the example.vcxproj located in ServoSDK\C++\Windows\PrimaryMemoryTable\Example\example.



Step 2: Replace the include directory with the local src path. Navigate through 'Debug -> Example debug properties ->All Configurations -> All Platforms -> VC++ Directories -> Include Directories dropdown -> Edit -> More -> Find local Src -> Select folder -> Confirm -> Apply -> OK'.



To avoid issues with missing .h files, it is recommended to set the Configuration (C) and Platform (P) to 'All Configurations' and 'All Platforms'

Step 3: Modify the actual USBLink port number. In the main.c file, change the code on line 35 from '12' to the actual USBLink port number.

```
    //Taking COM9 as an example
    if (serialPort.Open(9, 1000000))
```

Step 4: Change the module value you want to test to 1. In the main.cpp file, change the module value from "0" to "1" in lines 4-13.

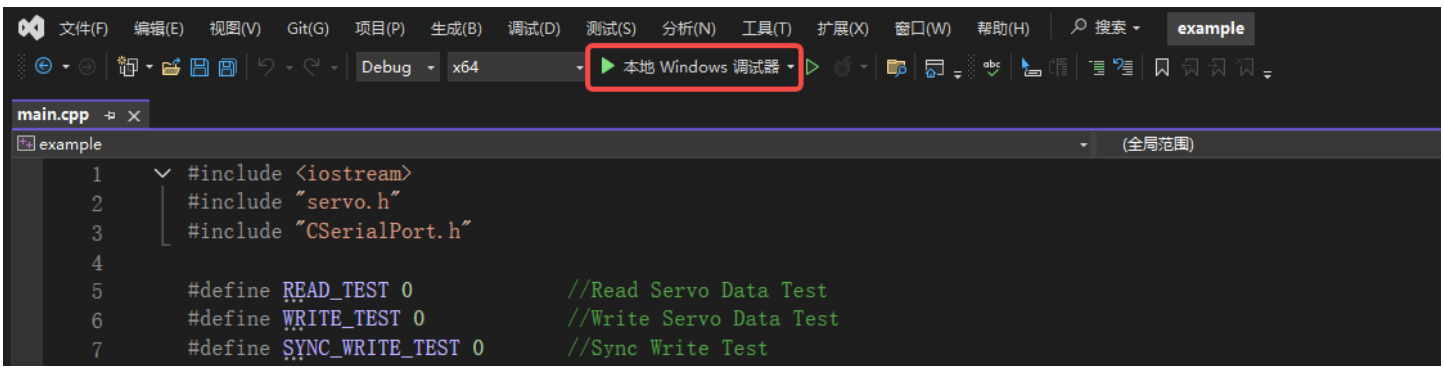
```
//Taking test the Ping module as an example
#define READ_TEST 0                 //Read Servo Data Test
#define WRITE_TEST 0                //Write Servo Data Test
#define SYNC_WRITE_TEST 0           //Sync Write Test
#define PING_TEST 1                 //PING Instruction Test
#define FACTORY_RESET_TEST 0        //Factory Reset Test
#define PARAMETER_RESET_TEST 0      //Parameter Reset Test
#define REBOOT_TEST 0               //Reboot Test
#define CALIBRATION_TEST 0          //Calibration Test
```

```
#define MODIFY_ID 0                 //Change Known Servo ID Test
#define MODIFY_UNKNOWN_ID 0         //Change Unknown Servo ID Test
```

Step 5: Change the printing output enable as needed. In the PrimaryServo.h file, change line 6. If printing output is not needed, set PRINTF_ENABLE to "0".

```
//Taking print enable as an example (no changes needed).
#define PRINTF_ENABLE 1             //Print output enable.
```

Step 6: Run the program. Click the "Local Windows Debugger" button in Visual Studio to run the program.



5.1.4 Run the Demo

For details, refer to *section 4.1.4 Run the Demo*.

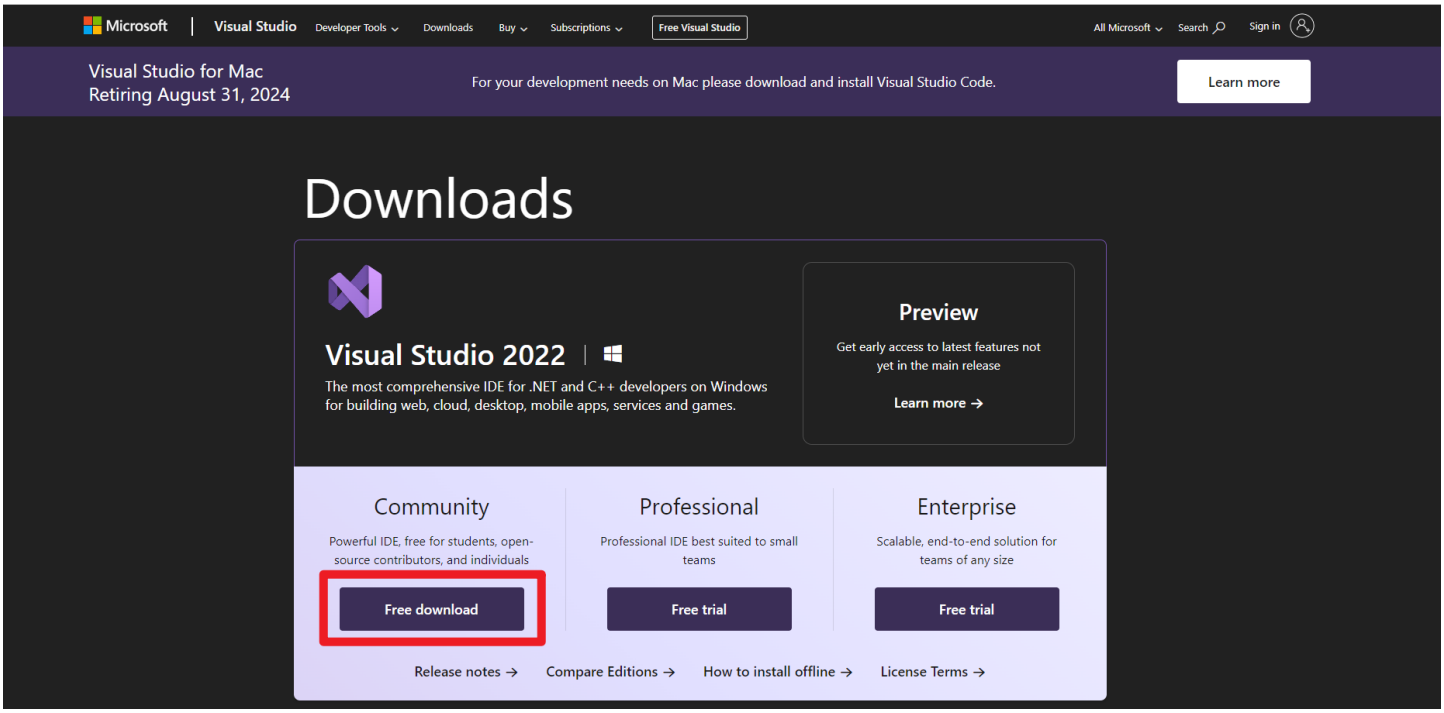
6. Python

6.1 Windows

6.1.1 Environment Setup

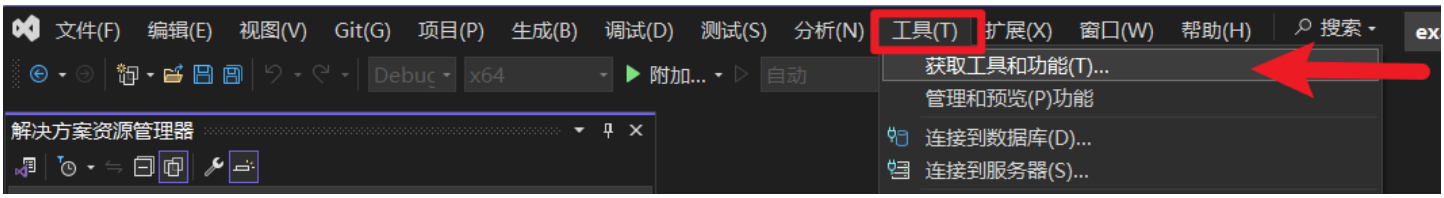
Set up a Python programming environment using an IDE, taking Visual Studio 2022 as an example.

Step 1: Install Visual Studio 2022. Visit https://visualstudio.microsoft.com/downloads/ and select the appropriate version for your operating system to download and install.



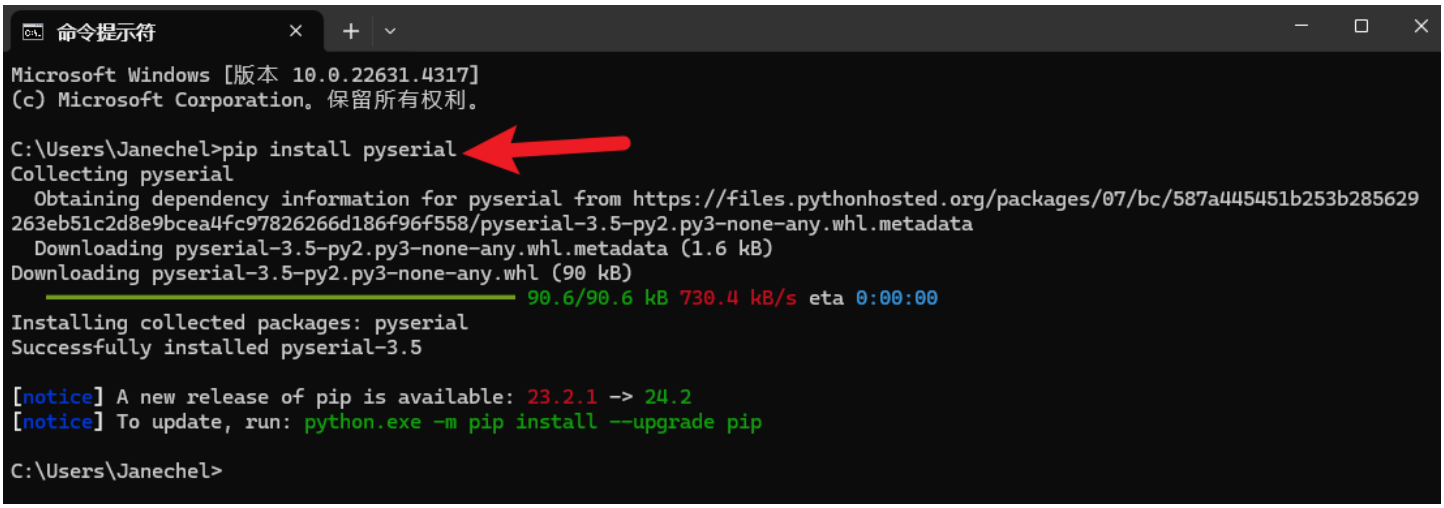
Step 2: Install Python development.

When installing the software, simply check the 'Python development' workload.



If you haven't checked it, you can also install it through the 'Menu Bar - Tools - Get Tools and Features.'

Install the pyserial module. In the desktop search box, enter the cmd command to open the Command Prompt. Then, type pip install pyserial and press Enter to install the module.



6.1.2 Hardware Connection

For details, refer to section *4.1.2 Hardware Connection*.

6.1.3 Run the Example

Step 1: Open the example.pyproj file. Taking EM-2030 as an example, double-click to open the example.pyproj located in ServoSDK\Python\Windows\PrimaryMemoryTable\Example\example.

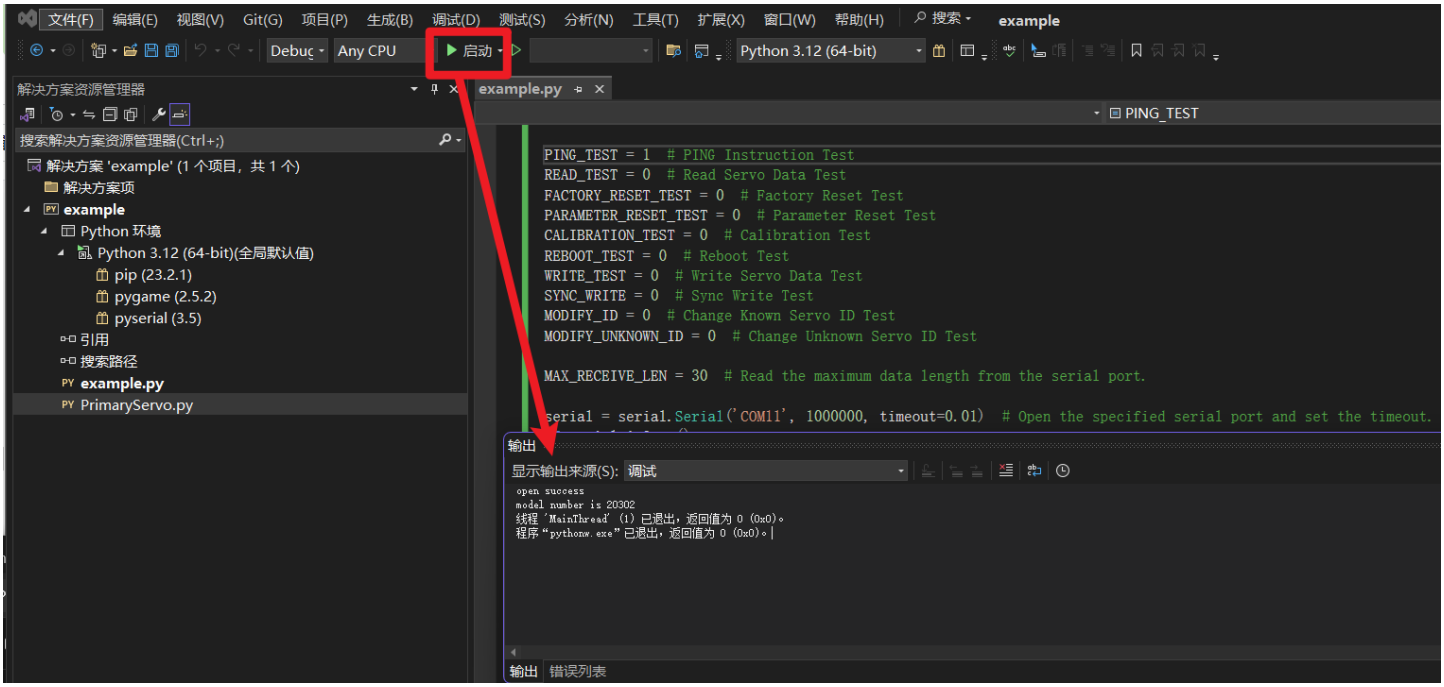
Step 2: Change the actual USBLink port number. In the example.py file, change the code on line 18 from "18" to the actual USBLink port number.

```
#Taking COM9 as an example.
serial = serial.Serial('COM11', 1000000, timeout=0.01)  # Open the specified serial port and set the timeout.
```

Step 3: Change the module value you want to test to 1. In the example.py file, change the module value from "0" to "1" in lines 5-14.

```
#Taking test the Ping module as an example
PING_TEST = 1  # PING Instruction Test
READ_TEST = 0  # Read Servo Data Test
FACTORY_RESET_TEST = 0  # Factory Reset Test
PARAMETER_RESET_TEST = 0  # Parameter Reset Test
CALIBRATION_TEST = 0  # Calibration Test
REBOOT_TEST = 0  # Reboot Test
WRITE_TEST = 0  # Write Servo Data Test
SYNC_WRITE = 0  # Sync Write Test
MODIFY_ID = 0  # Change Known Servo ID Test
MODIFY_UNKNOWN_ID = 0  # Change Unknown Servo ID Test
```

Step 4: Run the program. Click the "Start" button in VS to run the program.



Taking test the Ping module as an example

6.1.4 Run the Demo

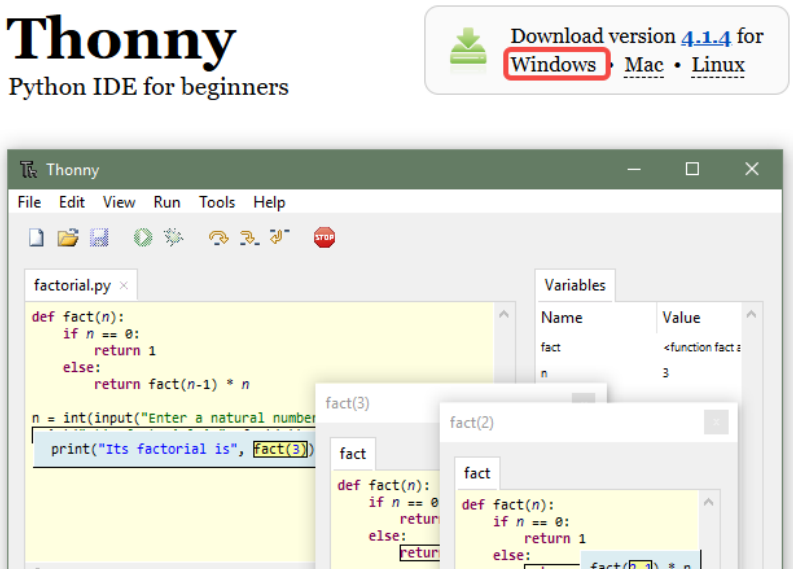
For details, refer to *section 4.1.4 Run the Demo*.

7. MicroPython

7.1 ESP32

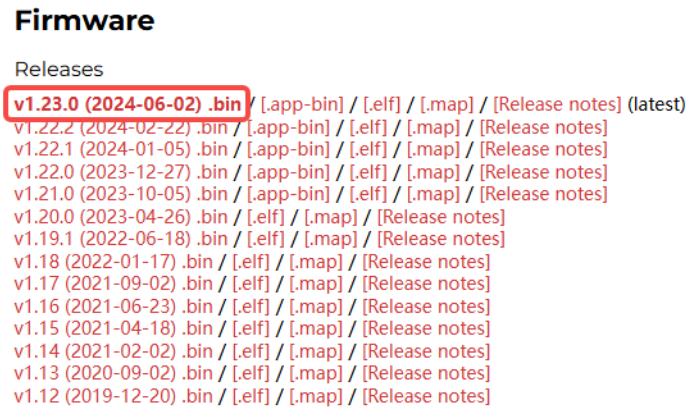
7.1.1 Environment Setup

Step 1: Install Thonny. Visit https://thonny.org/ and select the appropriate version for your operating system to download and install Thonny.



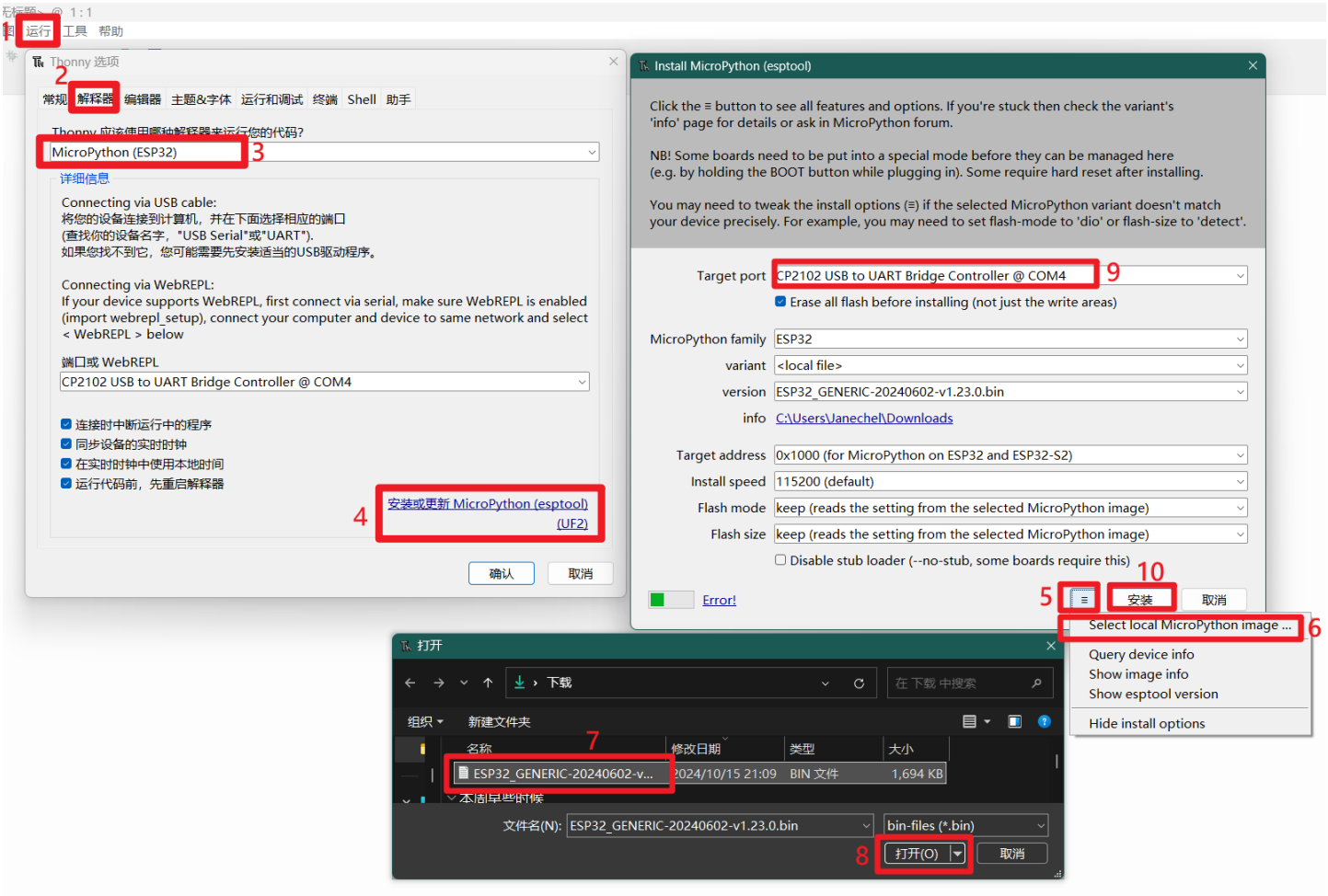
Taking Windows as an example

Step 2: Download the MicroPython firmware. Visit https://micropython.org/download/ESP32_GENERIC/ and download the latest firmware.

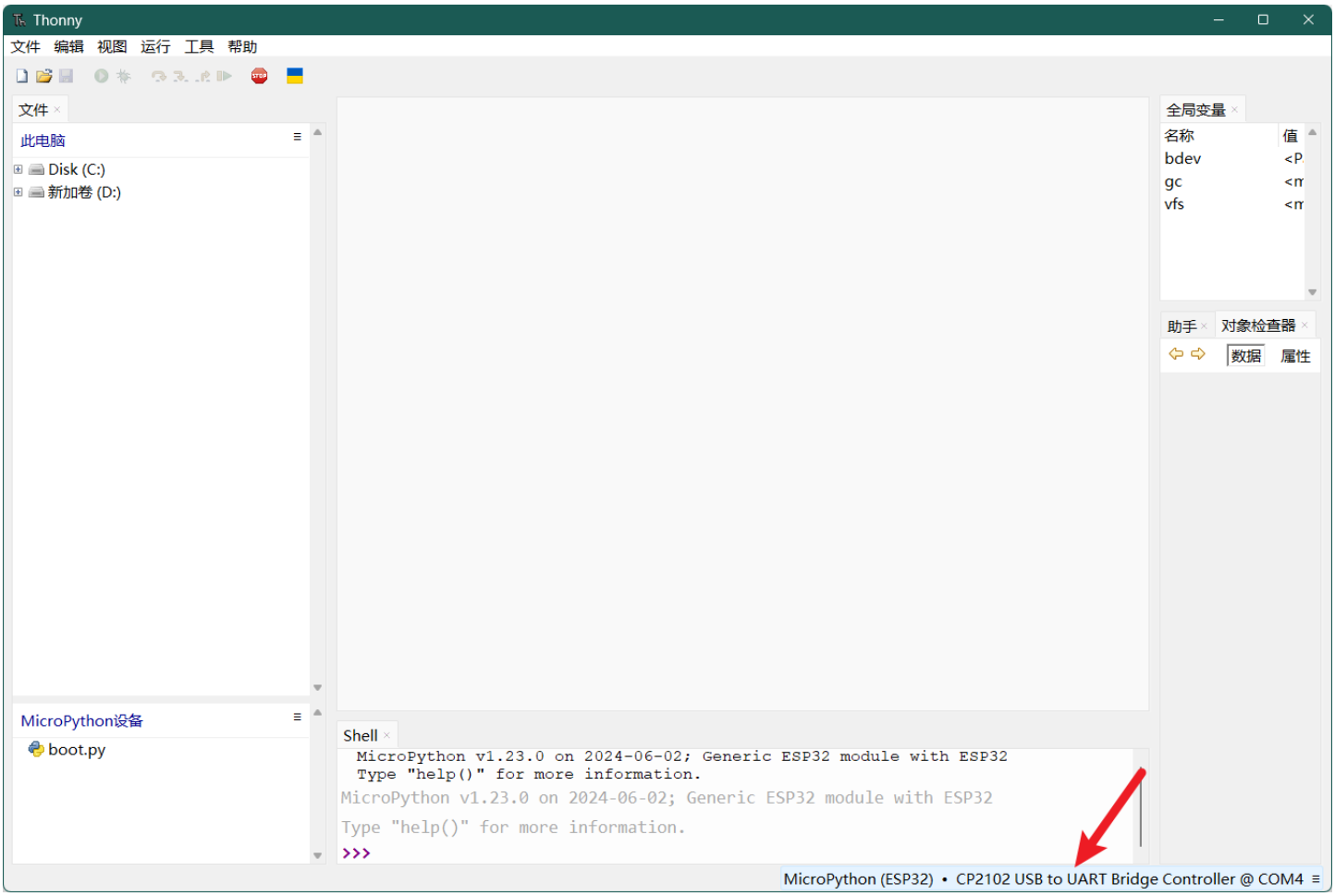


Step 3: Connect the device. Use a USB cable to connect the ESP32 to the PC, which will automatically detect and install the driver. If the device is not recognized, you will need to manually download the CP210x driver and update it.

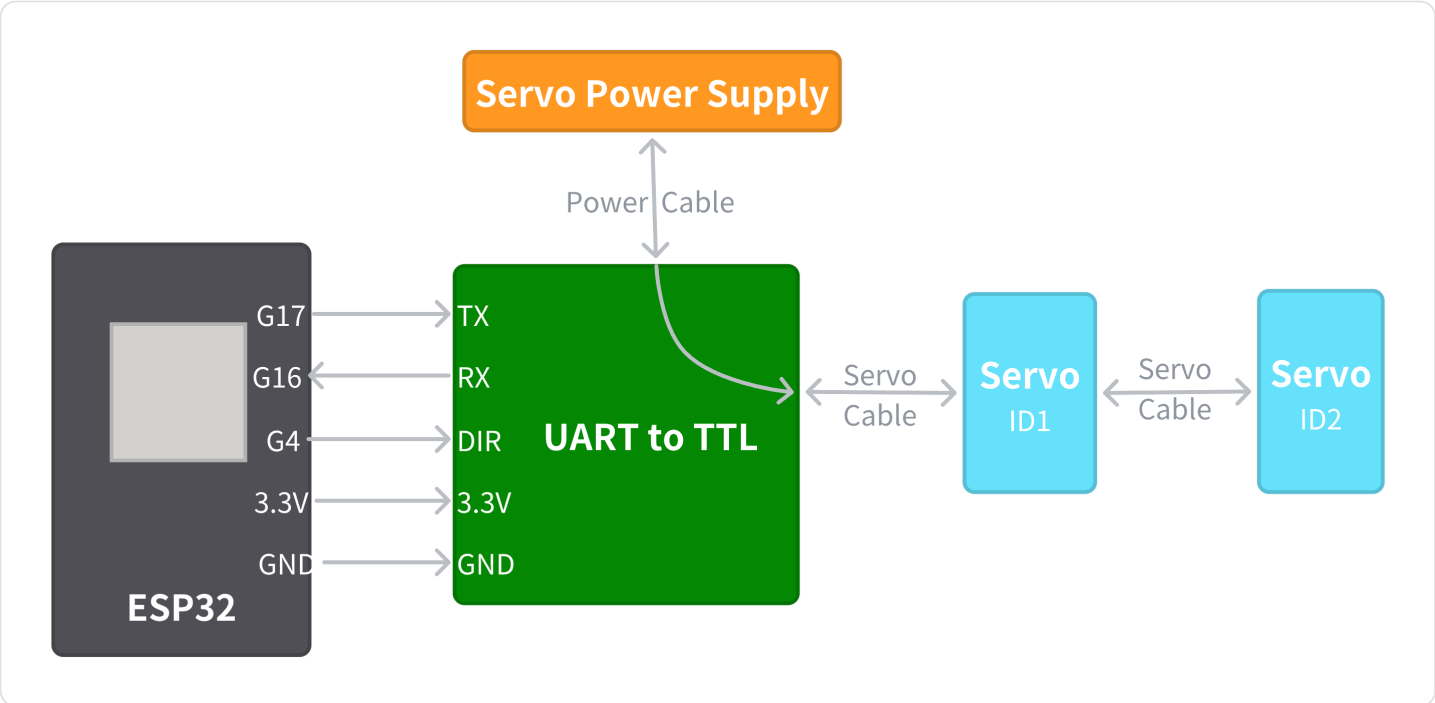
Step 4: Upload the MicroPython firmware. Open Thonny and navigate to Run -> Configure Interpreter -> MicroPython (ESP32) -> Install or Update MicroPython -> More -> Select Local MicroPython Image. Locate the downloaded .bin file, open it, select the target port, and click Install.



Step 5: Select the configurator.



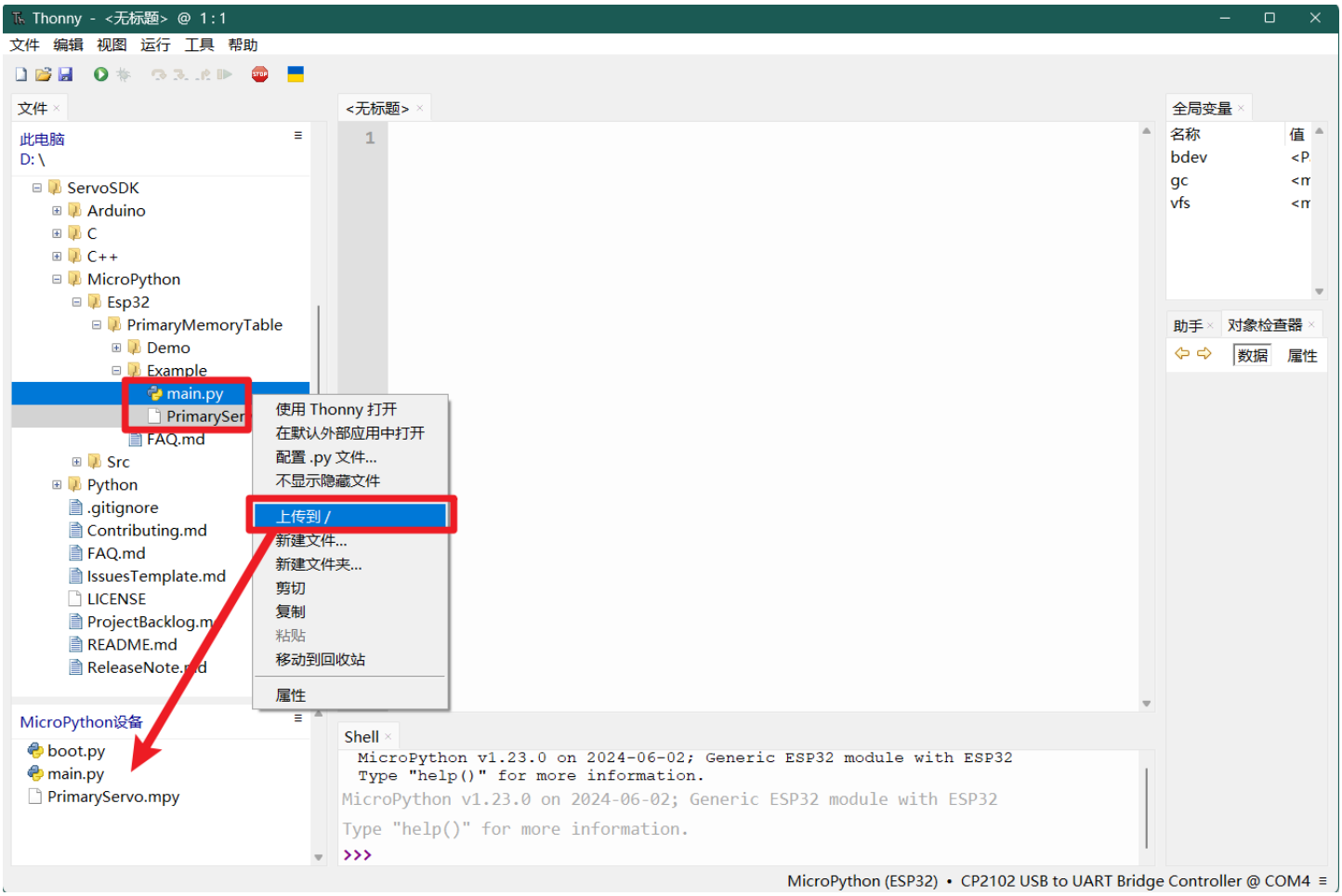
7.1.2 Hardware Connection



ESP32 and Servo Connection Diagram

7.1.3 Run the Example

Step 1: Upload main.py and PrimaryServo.mpy to the ESP32 device. Taking EM-2030 as an example, open Thonny, in the file window, locate main.py and PrimaryServo.mpy under ServoSDK\MicroPython\ESP32\PrimaryMemoryTable\Example, and upload them to the MicroPython device.



If the MicroPython device does not have a boot.py file, please reinstall the MicroPython firmware.

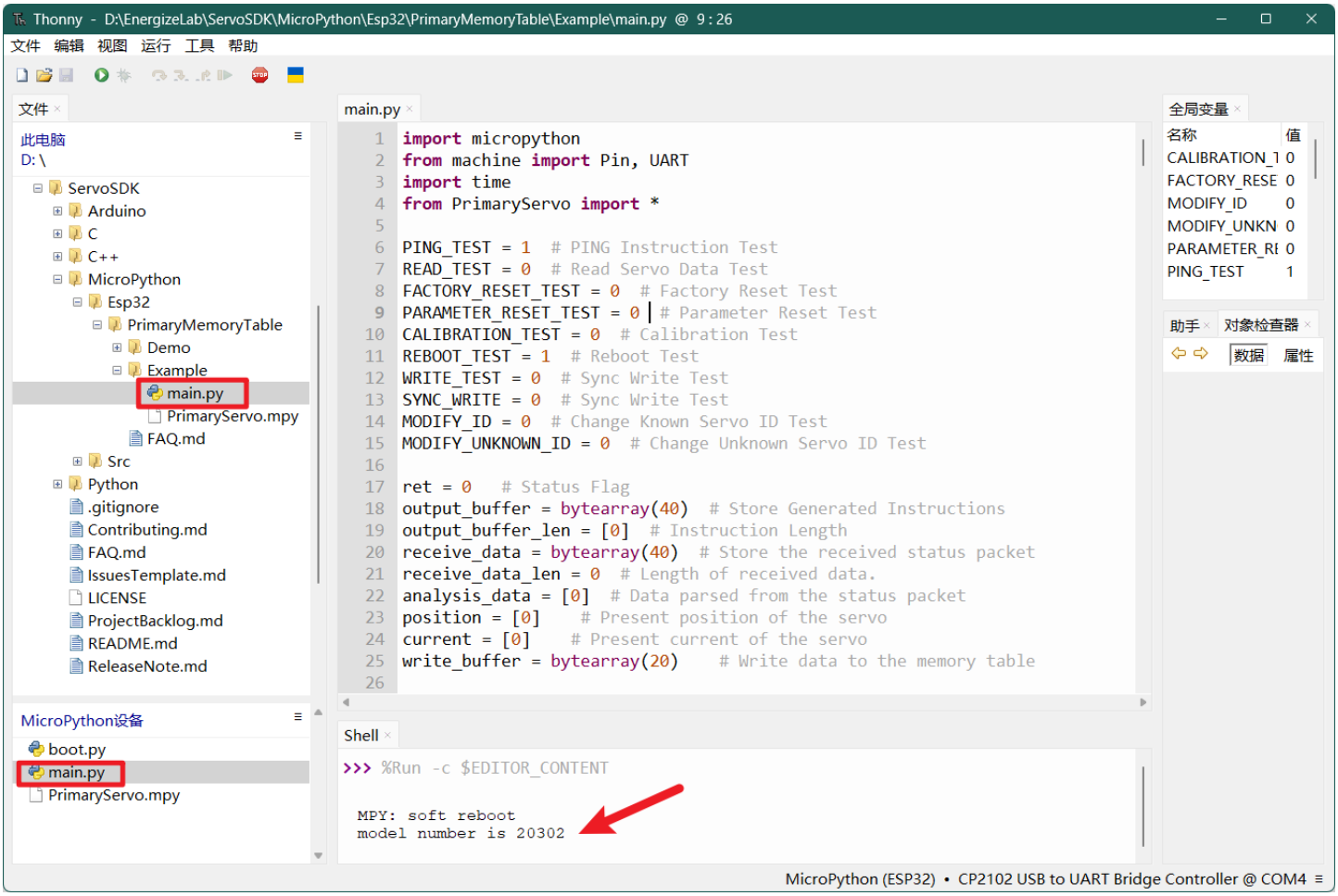
Step 2: Change the module value you want to test to 1. In the main.py file on the MicroPython device, modify lines 6-15 to change the functionality module value from "0" to "1."

```
#Taking test the Ping module as an example
PING_TEST = 1  # PING Instruction Test
READ_TEST = 0  # Read Servo Data Test
FACTORY_RESET_TEST = 0  # Factory Reset Test
PARAMETER_RESET_TEST = 0  # Parameter Reset Test
CALIBRATION_TEST = 0  # Calibration Test
REBOOT_TEST = 0  # Reboot Test
WRITE_TEST = 0  # Sync Write Test
SYNC_WRITE = 0  # Sync Write Test
MODIFY_ID = 0  # Change Known Servo ID Test
MODIFY_UNKNOWN_ID = 0  # Change Unknown Servo ID Test
```

Step 3: Run the current script. If you want to retain the settings after power loss, you need to edit the main file on the device.

Note: Due to the large size of the library files, importing them at once may fail because of memory fragmentation. To address this, convert the PrimaryServo.py file into a

PrimaryServo.mpy file using bytecode freezing. This file will not affect the normal import of modules.



7.1.4 Run the Demo

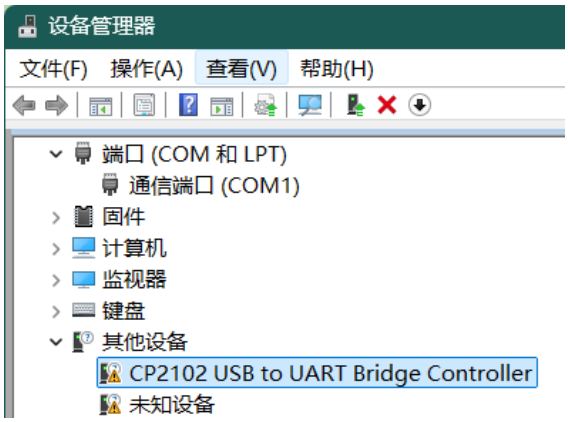
For details, please refer to *section 4.1.4 Demo.*

7.1.5 FAQ

- An index out of bounds error occurs in the get_check() function for the byte array.

```
Traceback (most recent call last):
  File "<stdin>", line 29, in <module>
  File "servo.py", line 2698, in servo set torque switch analysis
  File "servo.py", line 310, in servo unpack
  File "servo.py", line 224, in get check
IndexError: bytearray index out of range
```

  - The servo may not be powered on; be mindful of supplying power to the servo.
- Exclamation mark appears next to the device in device manager after connecting ESP32

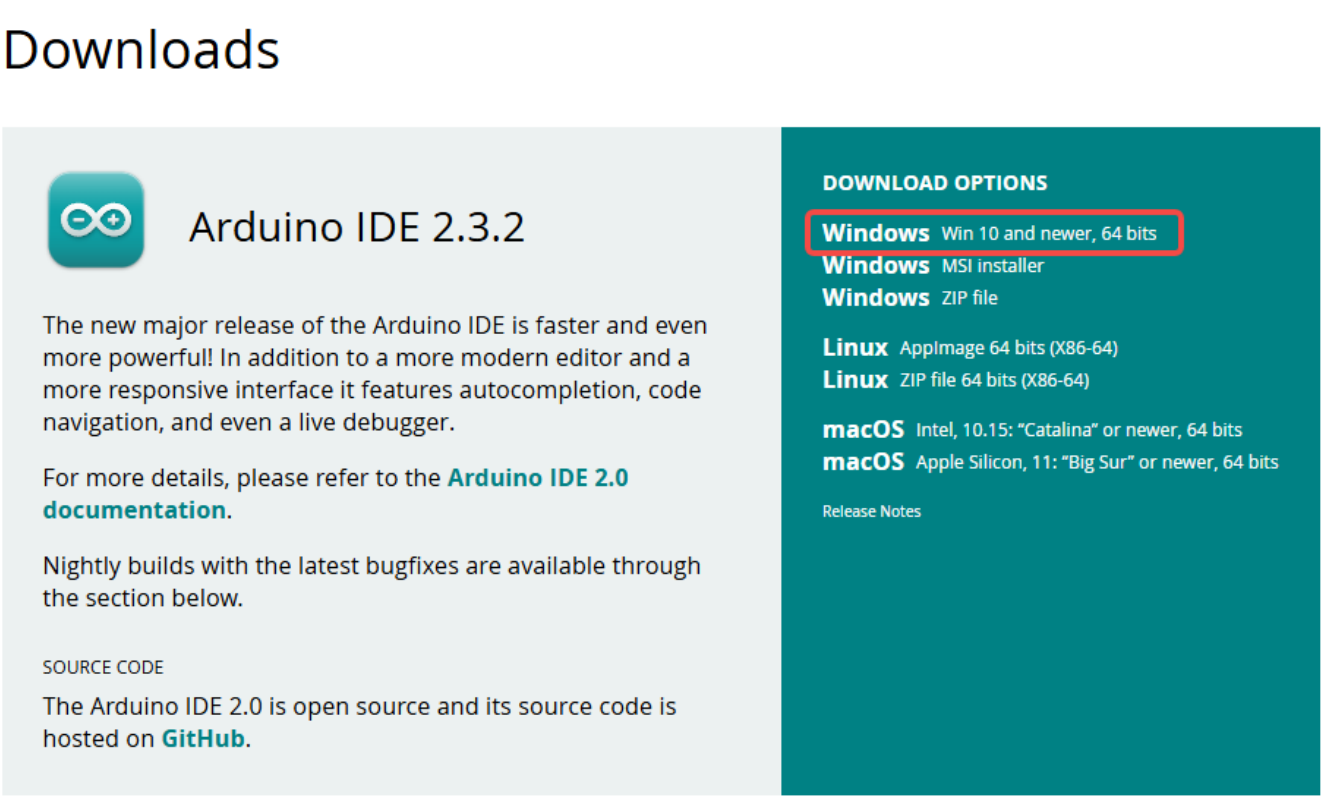


- Step 1: Download the driver. Visit https://cn.silabs.com/developers/usb-to-uart-bridge-vcp-drivers?tab=downloads and select the appropriate version for your operating system to download and install.
- Step 2: Right-click on the device -> Update Driver -> Browse my computer for drivers -> Browse to the path where you downloaded the driver -> Next -> Search automatically for updated driver software.

8. Arduino

8.1 Environment Setup

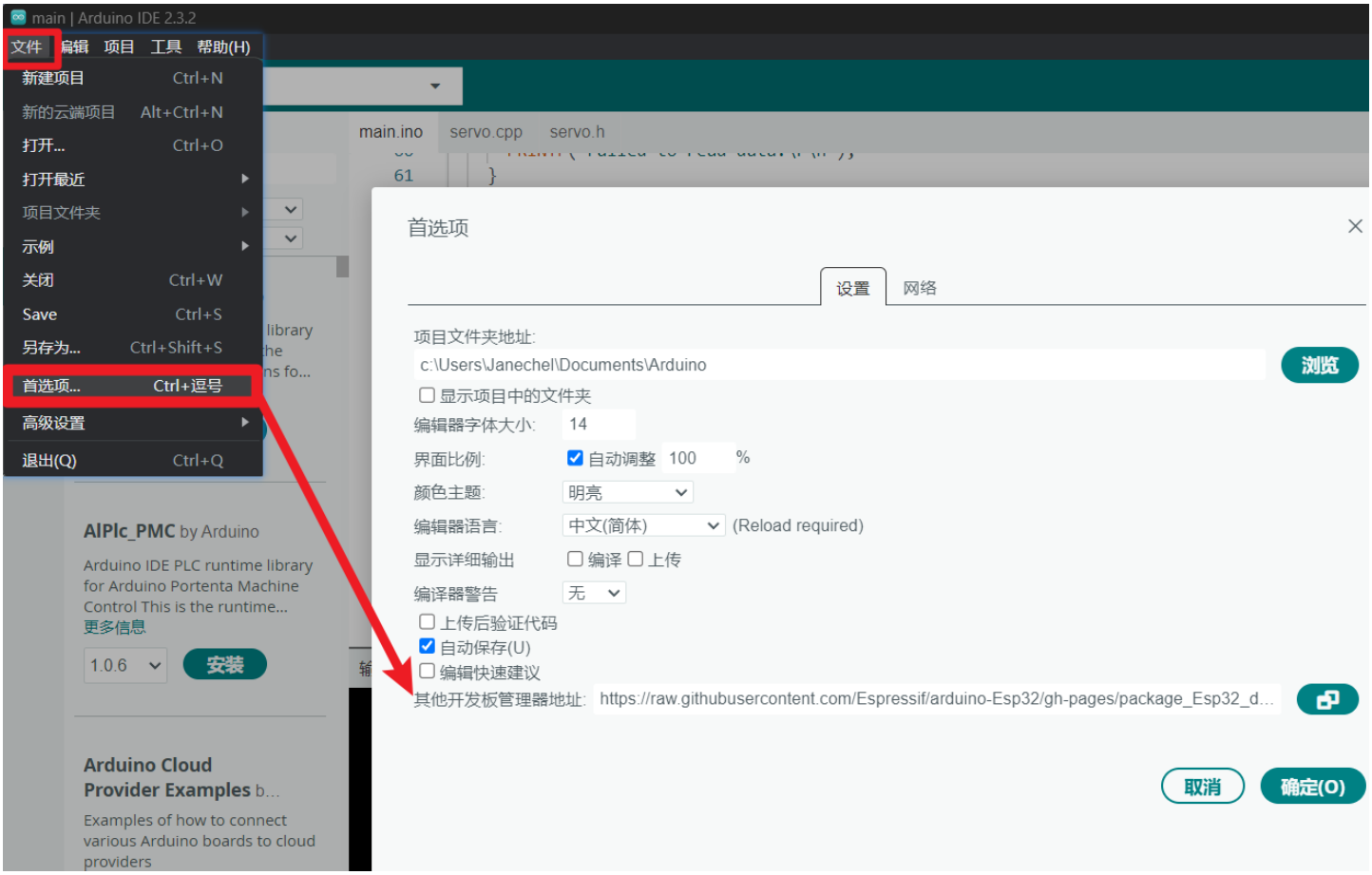
Step 1: Install Arduino. Visit https://www.arduino.cc/en/software, and select the appropriate version for your operating system to download and install.



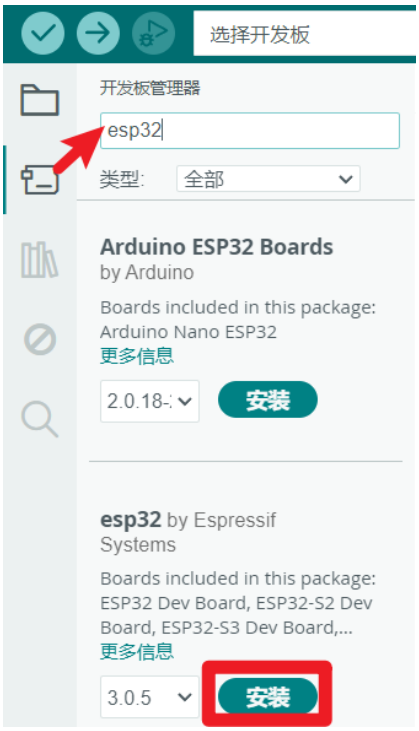
Taking Windows as an example

Step 2: Add the ESP32 development board. Open Arduino, then select File -> Preferences. In the Additional Board Manager URLs field, paste the following link:

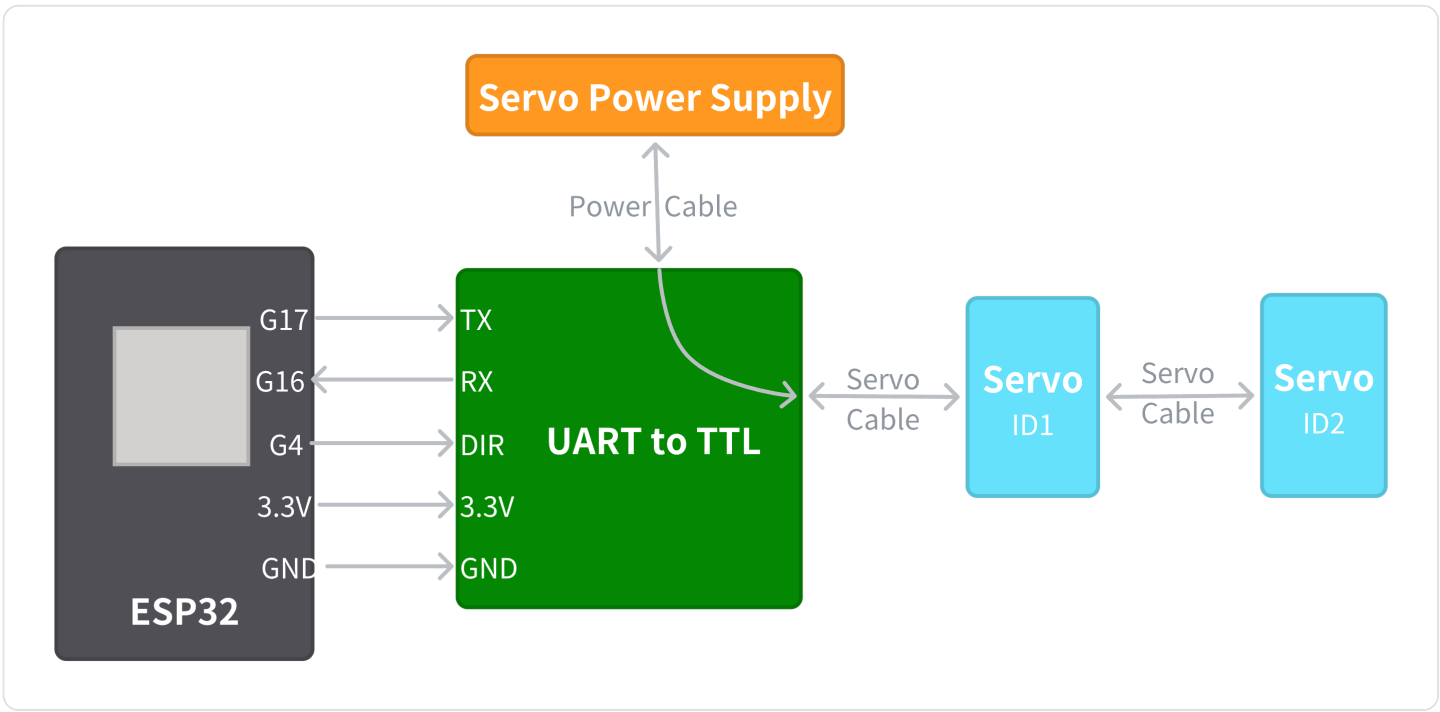
```
https://raw.githubusercontent.com/Espressif/arduino-Esp32/gh-pages/package_Esp32_dev_index.json
```



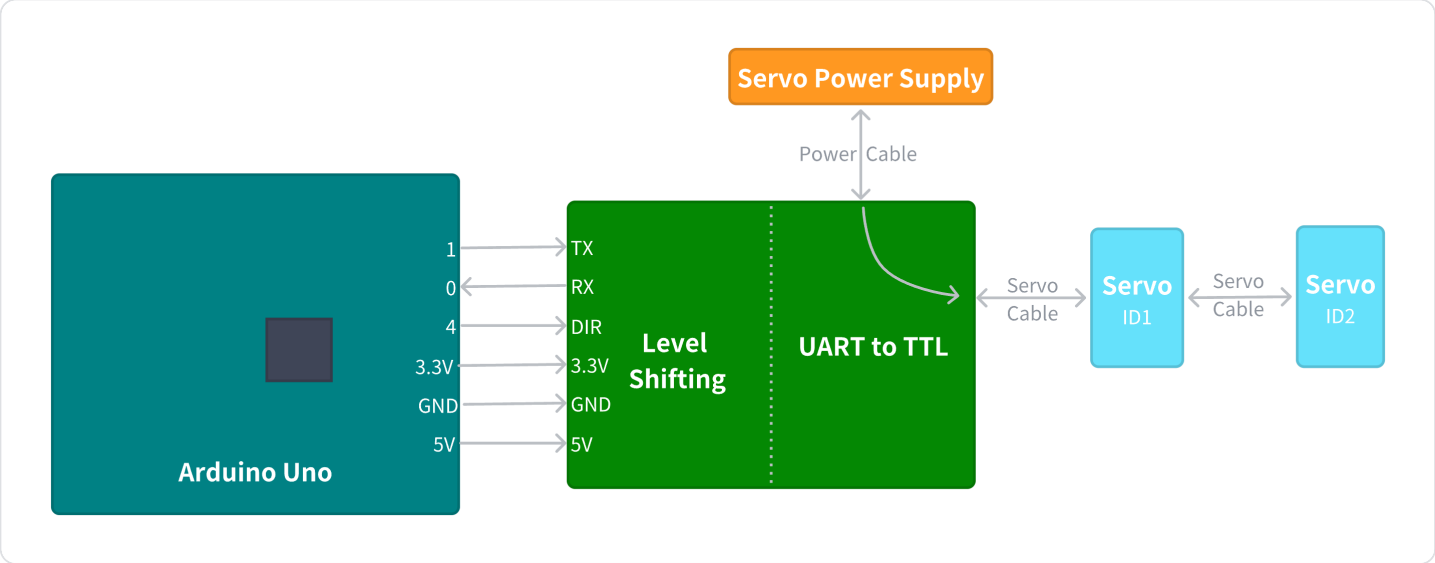
Step 3: Install the ESP32 development board. Open Arduino, click on the Boards Manager, enter 'esp32' , select the latest version, and complete the installation.



8.2 Hardware Connection



ESP32 and Servo Connection Diagram

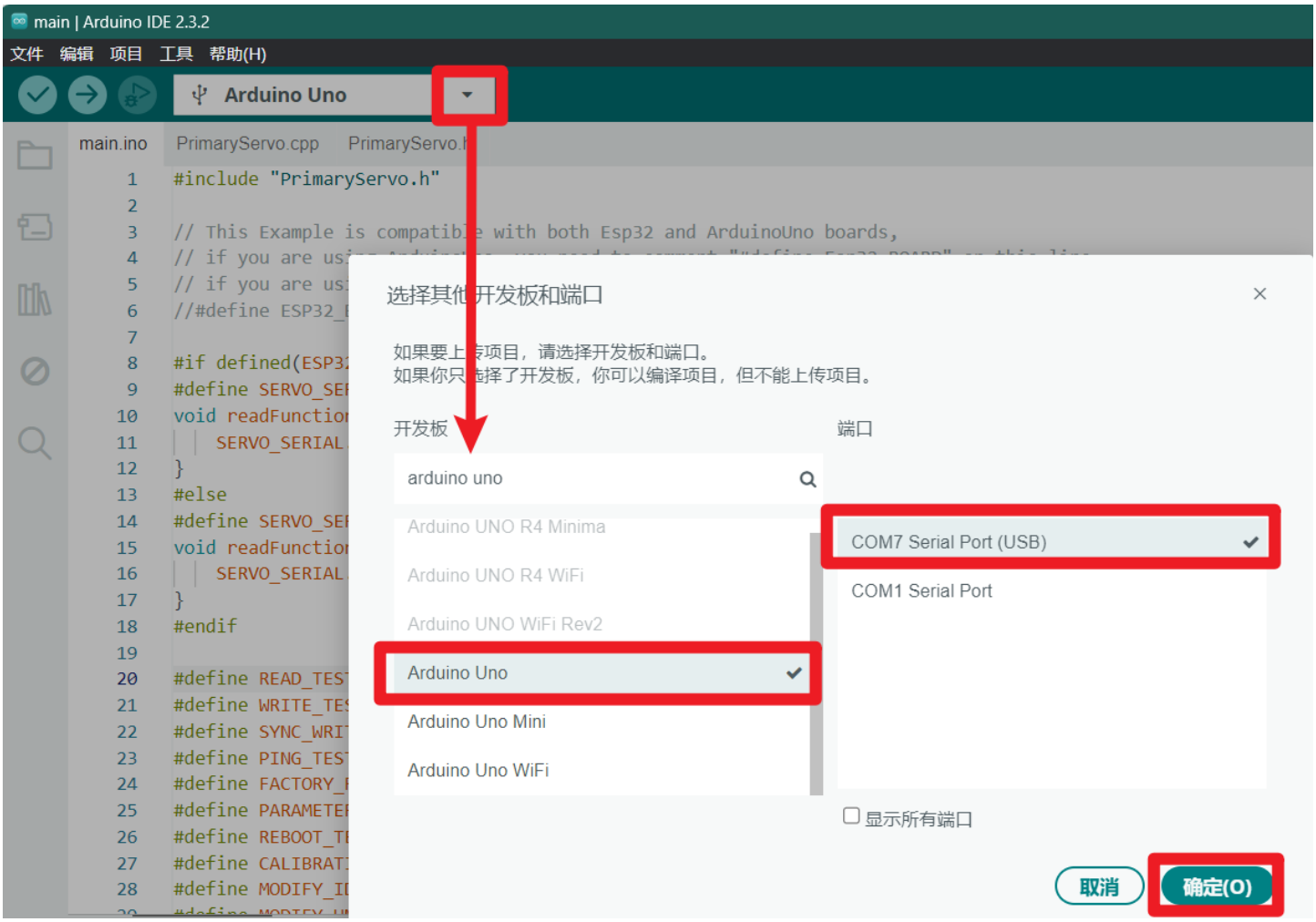


Arduino Uno and Servo Connection Diagram

8.3 Run the Example

Step 1: Download the servo control library. Open the Arduino software and download the Energize Lab Servo library. Click on "..." and select the corresponding example to open.

Step 2: Select the serial port and board type. Choose the appropriate serial port and board type based on the development board you are using.



Taking Arduino Uno as an example

Step 3: Comment out Esp32_BOARD based on the board selection. This example is compatible with both Esp32 and ArduinoUno boards. If using ArduinoUno, comment out line 6 of main.ino, "#define Esp32_BOARD". If using Esp32, no changes are needed.

```
1  // This Example is compatible with both Esp32 and ArduinoUno boards,
2  // if you are using ArduinoUno, you need to comment "#define Esp32_BOARD" on
   this line,
3  // if you are using Esp32 it does not need to change.
4
5  //Taking Arduino Uno as an example
6  //#define ESP32_BOARD
7
8  #if defined(ESP32_BOARD)
9  #define SERVO_SERIAL Serial2
10 void readFunction(uint8_t* pack, uint8_t pack_len) {
11     SERVO_SERIAL.read(pack, pack_len);
12 }
13 #else
14 #define SERVO_SERIAL Serial
15 void readFunction(uint8_t* pack, uint8_t pack_len) {
16     SERVO_SERIAL.readBytes(pack, pack_len);
```

```
}
#endif
```

Step 4: Change the module value you want to test to 1. In lines 17-26 of main.ino, change the module value "0" to "1" for the module you want to test.

Note: It is recommended to test only one module at a time to avoid upload errors.

```
//Taking test the Ping module as an example
#define READ_TEST 0              //Read Servo Data Test
#define WRITE_TEST 0             //Write Servo Data Test
#define SYNC_WRITE_TEST 0        //Sync Write Test
#define PING_TEST 1              //PING Instruction Test
#define FACTORY_RESET_TEST 0     //Factory Reset Test
#define PARAMETER_RESET_TEST 0   //Parameter Reset Test
#define REBOOT_TEST 0            //Reboot Test
#define CALIBRATION_TEST 0       //Calibration Test
#define MODIFY_ID 0              //Change Known Servo ID Test
#define MODIFY_UNKNOWN_ID 0      //Change Unknown Servo ID Test
```

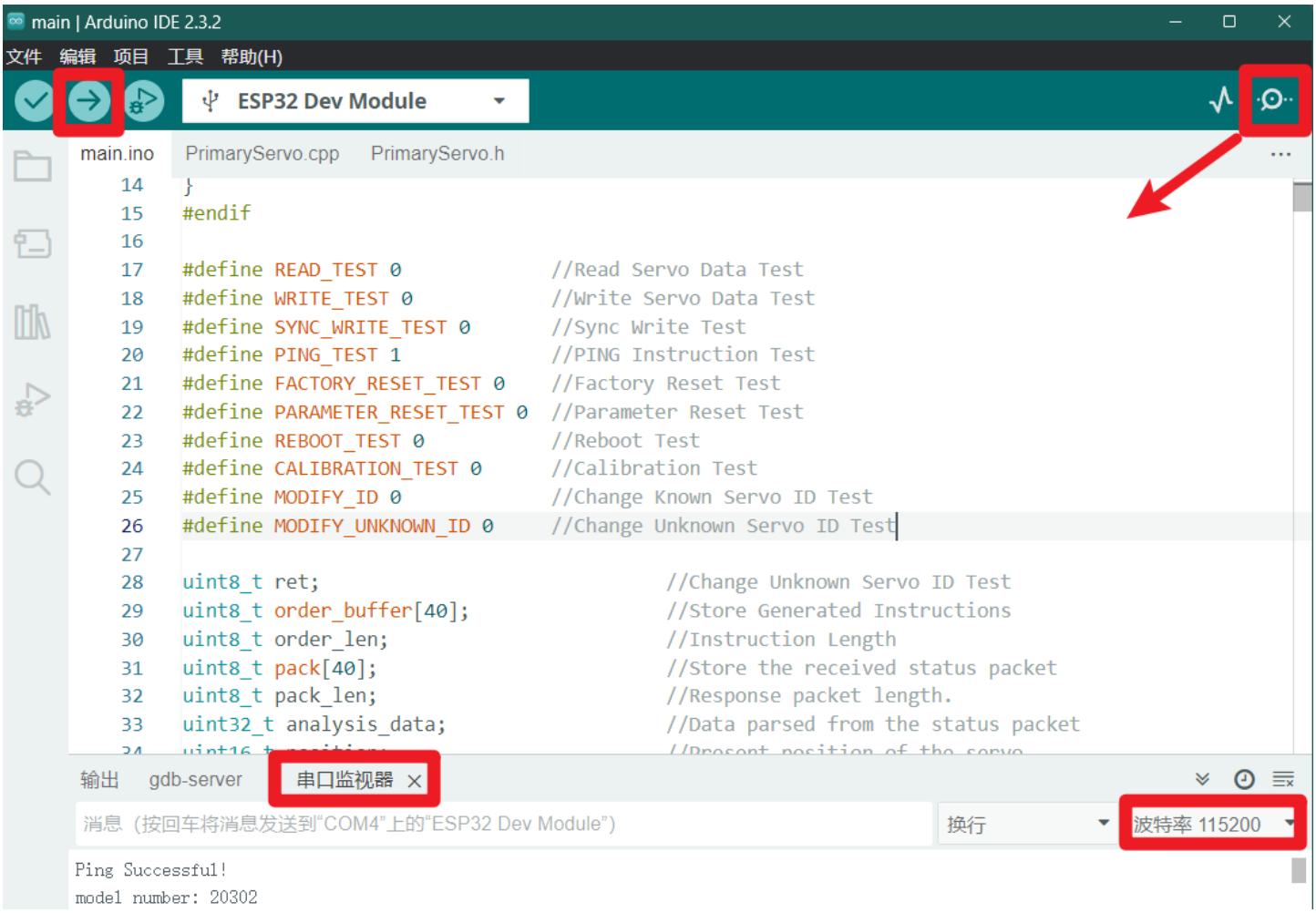
Step 5: Change the maximum number of servos for sync write and enable printing output as needed. In lines 6-8 of the PrimaryServo.h file, `MAX_SERVOS` defines the maximum number of synchronized servos supported, with a default of writing data for 6 servos simultaneously. `Serial` is the output serial port for printing actual information, which is the same for both ESP32 and Arduino, and does not need to be modified. If you do not need to print output results, change the value of `PRINTF_ENABLE` to `0`.

Note: Due to memory limitations of different boards, the maximum number of synchronized servos supported may vary. For example, Arduino Uno supports up to 6 servos for synchronized writing in the provided examples, while ESP32 can support more

```
//Taking one servo and print enable as an example (no changes needed).
#define ServoPrintf Serial          //A serial port for output information.
#define PRINTF_ENABLE 1             //Print output enable.
#define MAX_SERVERS 6              //Maximum number of servos for sync write.
```

Step 6: Compile and upload. To avoid upload failures, unplug the RX and TX pin wires before clicking 'Upload.'

Step 7: Check the information. After compilation is complete, reconnect the RX and TX wires, open the Serial Monitor, and adjust the baud rate to view the complete debug information.



The baud rate for ESP32 is 115200, while for Arduino Uno it is 1000000

8.4 Run the Demo

For details, refer to *section 4.1.4 Run the Demo*.

Note: If you are using Arduino Uno, you need to comment out line 6 of the .ino file, which is //#define ESP32_BOARD.

8.5 FAQ

- Due to the memory limitations of Uno, it may be necessary to appropriately reduce the maximum number of sync write servos.
- Arduino library files are in read-only status within the Arduino software; if editing is needed, locate the local library files, open them, and then edit.

To open the .h file: Move the mouse to the file location, copy the path from the popup window, and open it in the file manager.