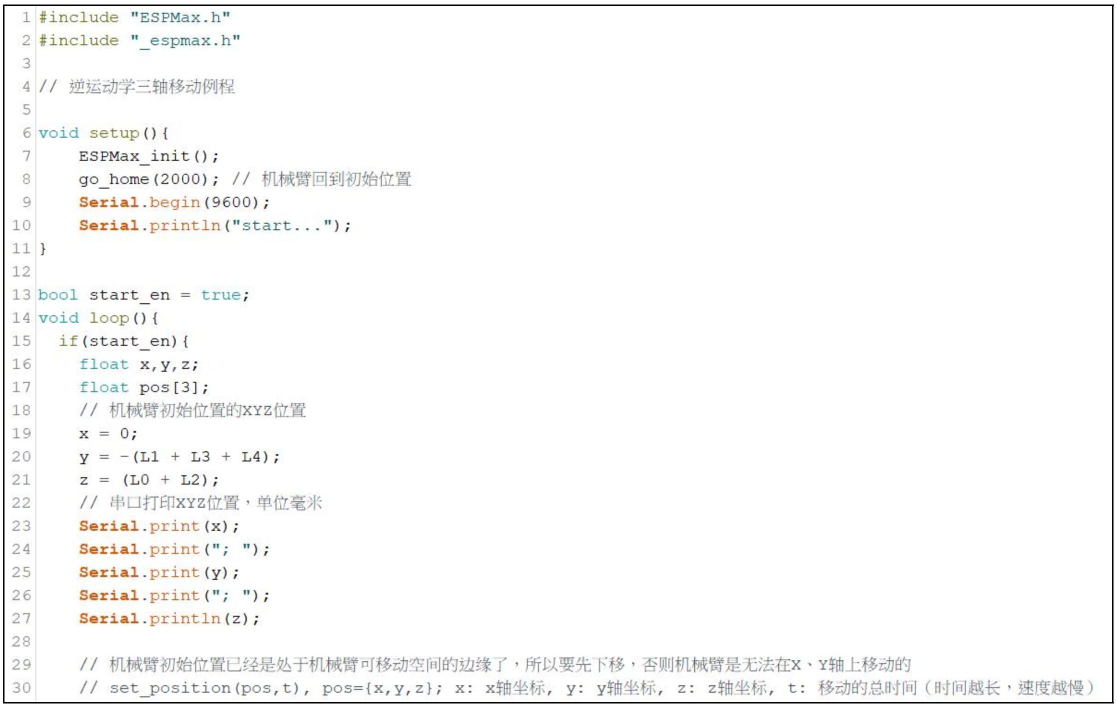
Lesson 4 Move on XYZ Axis

1. Working Principle

Call the related kinematics functions by calling “espmax” kinematics encapsulation library to realize the movement of robotic arm on xyz axes.

The path to the program of the source code is 7. Inverse Kinematics Basics and Application/Arduino Development/Lesson 4 The Movement on XYZ Axes/Program File/kinematics_move/kinematics_move.ino.



```
#include "ESPMax.h"
#include "_espmax.h"

// 逆运动学三轴移动例程

void setup(){
    ESPMax_init();
    go_home(2000); // 机械臂回到初始位置
    Serial.begin(9600);
    Serial.println("start...");
}

bool start_en = true;
void loop(){
  if(start_en){
    float x,y,z;
    float pos[3];
    // 机械臂初始位置的XYZ位置
    x = 0;
    y = -(L1 + L3 + L4);
    z = (L0 + L2);
    // 串口打印XYZ位置，单位毫米
    Serial.print(x);
    Serial.print("; ");
    Serial.print(y);
    Serial.print("; ");
    Serial.println(z);

    // 机械臂初始位置已经是处于机械臂可移动空间的边缘了，所以要先下移，否则机械臂是无法在X、Y轴上移动的
    // set_position(pos,t), pos={x,y,z}; x: x轴坐标, y: y轴坐标, z: z轴坐标, t: 移动的总时间（时间越长，速度越慢）
```

2. Preparation

2.1 Hardware

MaxArm robotic arm, power adapter, USB cable.

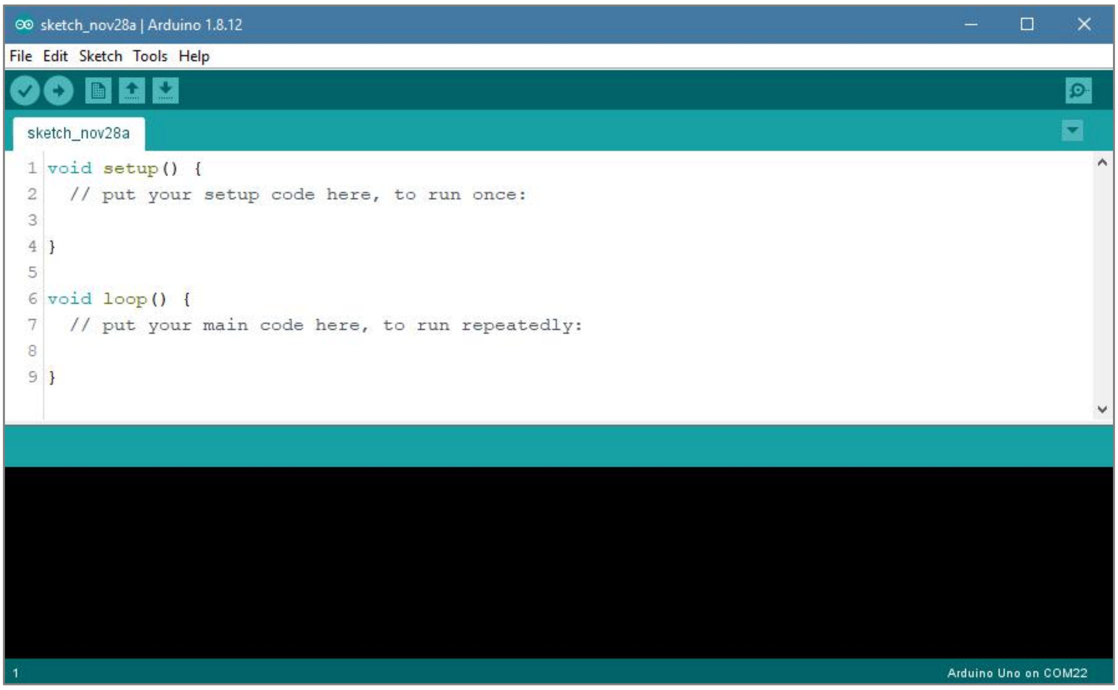
2.2 Software

Please connect MaxArm to Arduino editor according to the tutorial in folder “4.

MaxArm Underlying Program/Python Development/Lesson 1 Set Development Environment”.

3. Program Download

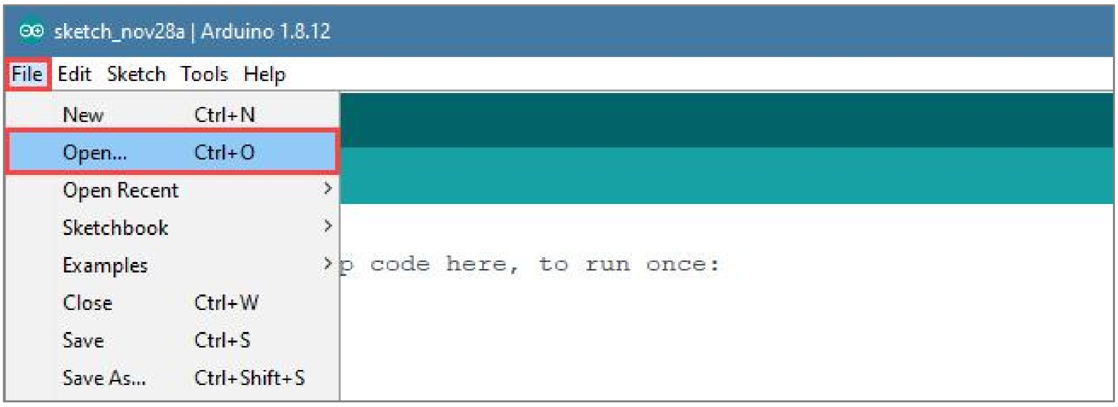
1) Click on ![arduino] icon to open Arduino IDE.



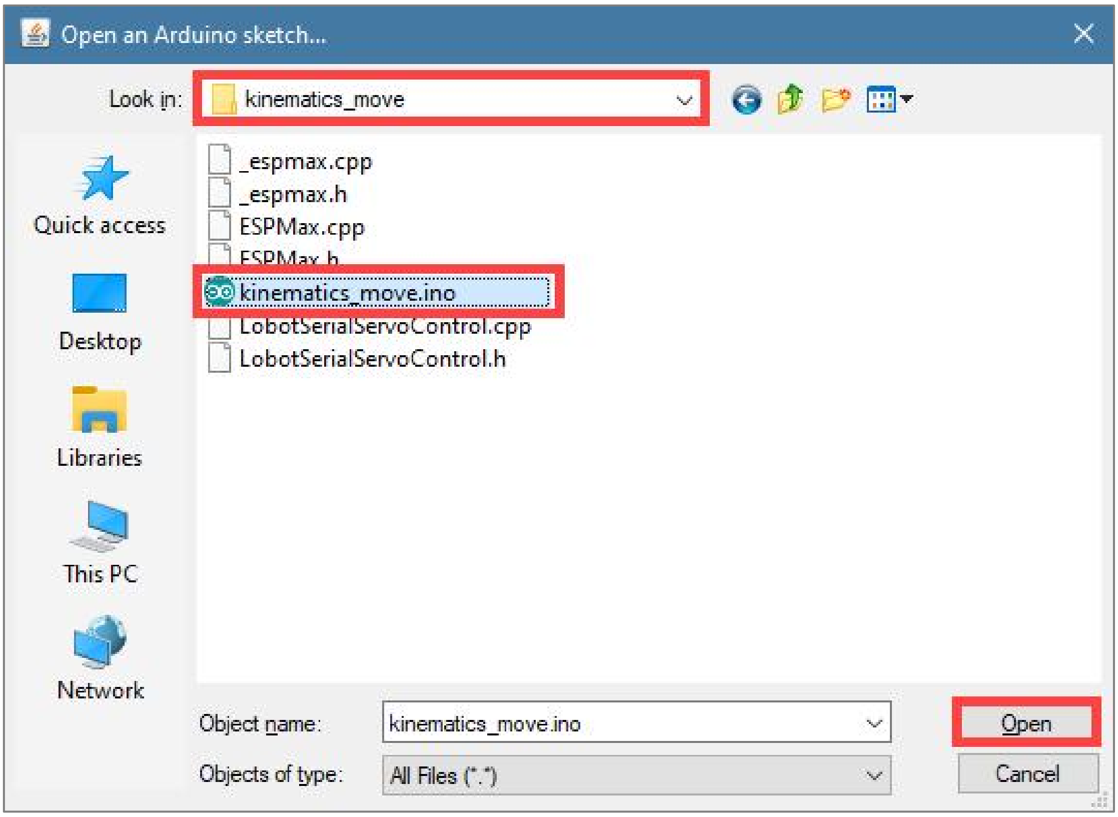
sketch_nov28a | Arduino 1.8.12

```
void setup() {
  // put your setup code here, to run once:

}

void loop() {
  // put your main code here, to run repeatedly:

}
```

2) Click “File->Open” in turn.



3) Select the program "kinematics_move.ino" in the folder " 7. Inverse Kinematics Basics and Application/Arduino Development/Lesson 4 The Movement on XYZ Axes/Program File/kinematics_move".



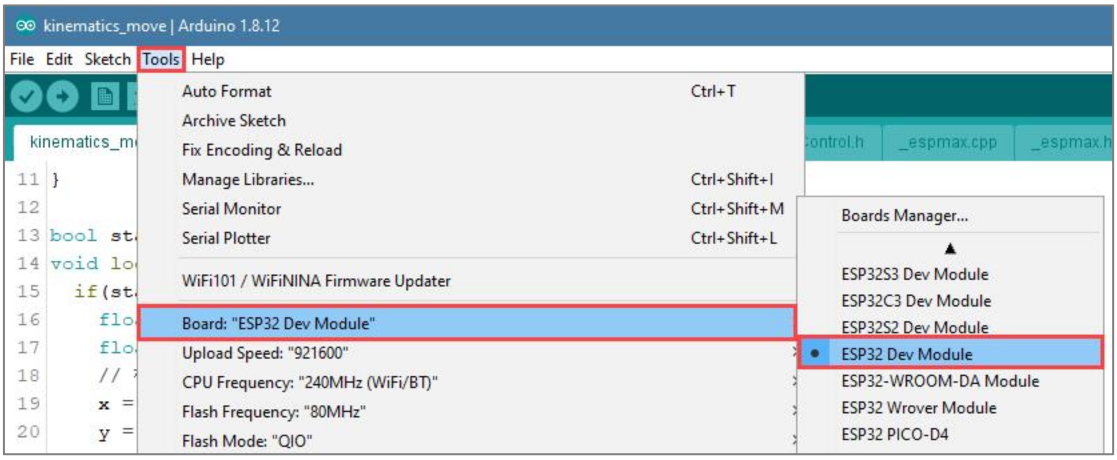
Open an Arduino sketch...

Look in: kinematics_move

Object name: kinematics_move.ino

Objects of type: All Files (*.*)

4) Select the model of the development board. Click "Tools-> Board" and select "ESP 32 Dev Module" (If the model of the development board has been configured when setting the development environment, you can skip this step).

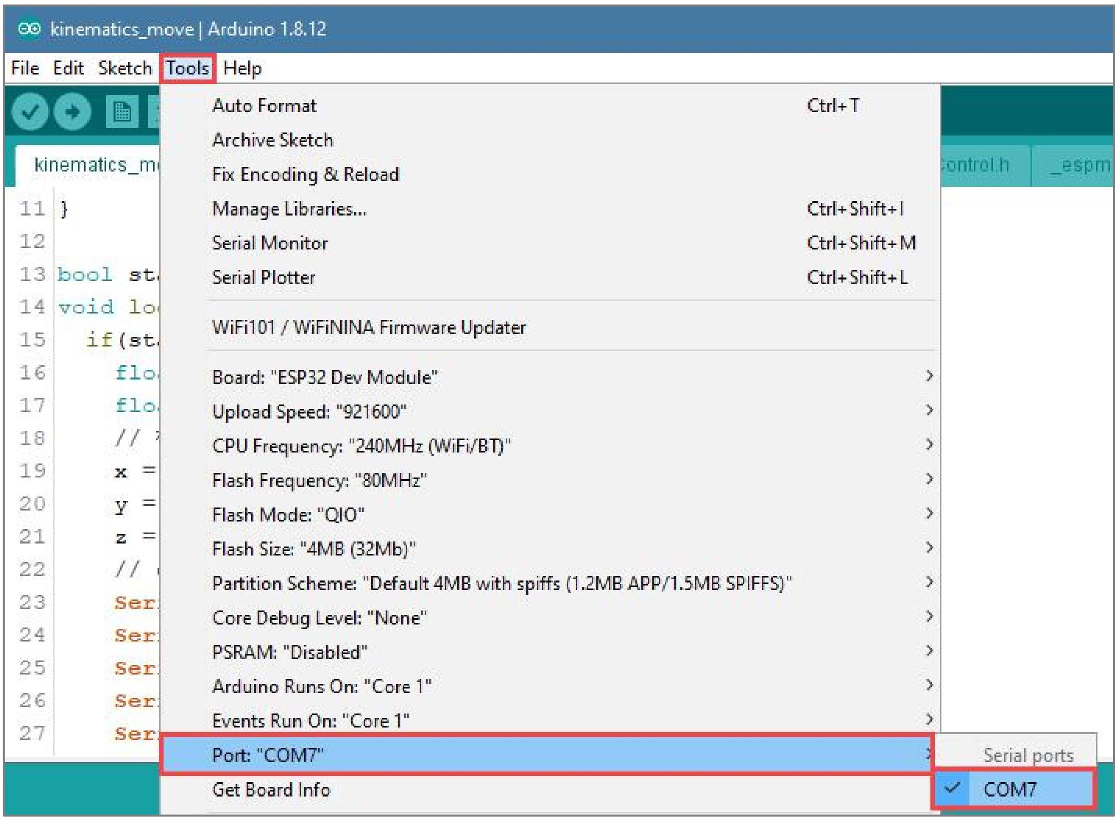


kinematics_move | Arduino 1.8.12

```
}

bool st
void lo
  if(st
    flo
    flo
    //
    x =
    y =
```

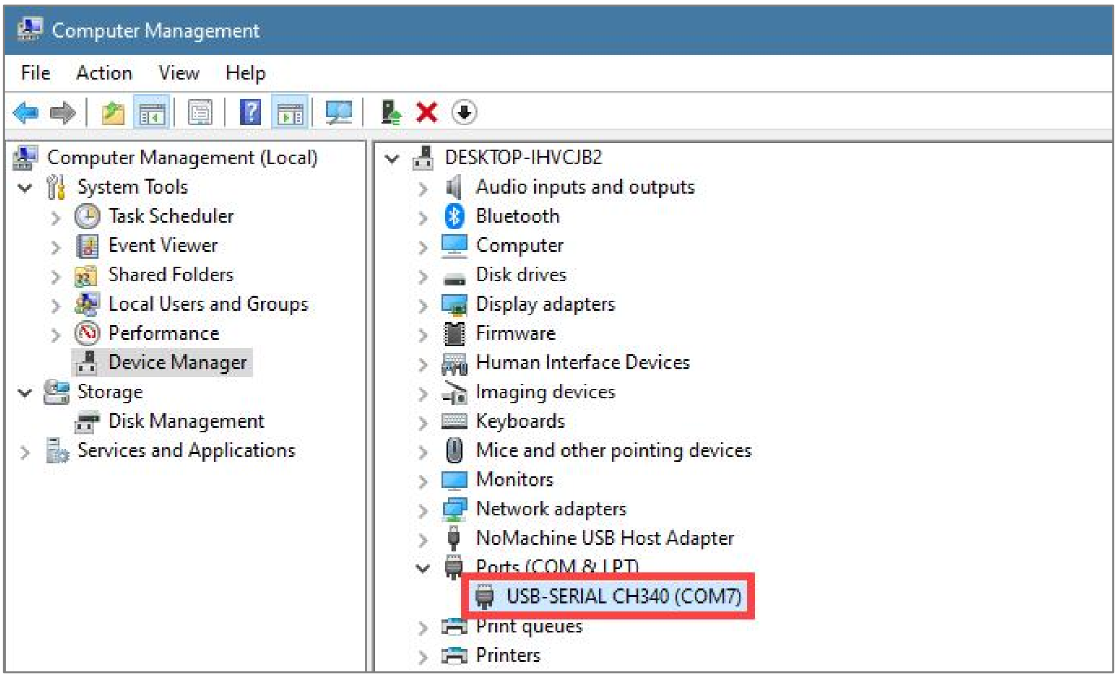
Board: "ESP32 Dev Module"
Upload Speed: "921600"
CPU Frequency: "240MHz (WiFi/BT)"
Flash Frequency: "80MHz"
Flash Mode: "QIO"

5) Select the corresponding port of Arduino controller in “Tools->Port”. (Here take the port “COM5” as example. Please select the port based on your computer. If COM1 appears, please do not select because it is the system communication port but not the actual port of the development port.)

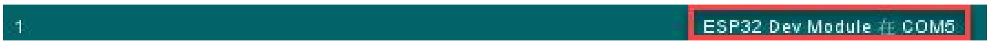

kinematics_move | Arduino 1.8.12
File
Edit
Sketch
Tools
Help
kinematics_m
control.h
_espm
Auto Format Ctrl+T
Archive Sketch
Fix Encoding & Reload
Manage Libraries... Ctrl+Shift+I
Serial Monitor Ctrl+Shift+M
Serial Plotter Ctrl+Shift+L
WiFi101 / WiFiNINA Firmware Updater
Get Board Info
Serial ports
✓ COM7

Board: "ESP32 Dev Module"
Upload Speed: "921600"
CPU Frequency: "240MHz (WiFi/BT)"
Flash Frequency: "80MHz"
Flash Mode: "QIO"
Flash Size: "4MB (32Mb)"
Partition Scheme: "Default 4MB with spiffs (1.2MB APP/1.5MB SPIFFS)"
Core Debug Level: "None"
PSRAM: "Disabled"
Arduino Runs On: "Core 1"
Events Run On: "Core 1"
Port: "COM7"

6) If you’re not sure about the port number, please open the “This PC” and click “Properties->Device Manger” in turns to check the corresponding port number (the device is with CH340). Then select the correct port on Arduino editor.

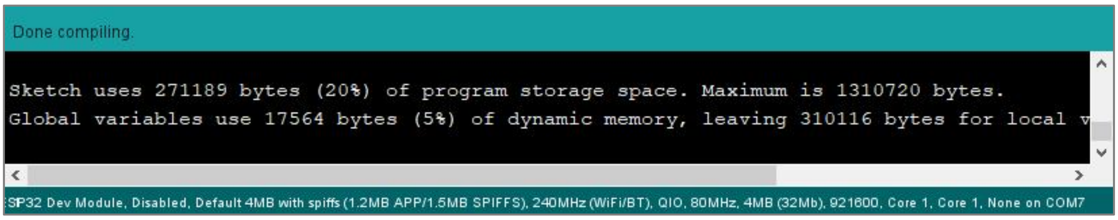


7) After selecting, confirm the board “ESP32 Dev Module” in the lower right corner and the port number “COM5” (it is an example here, please refer to the actual situation).



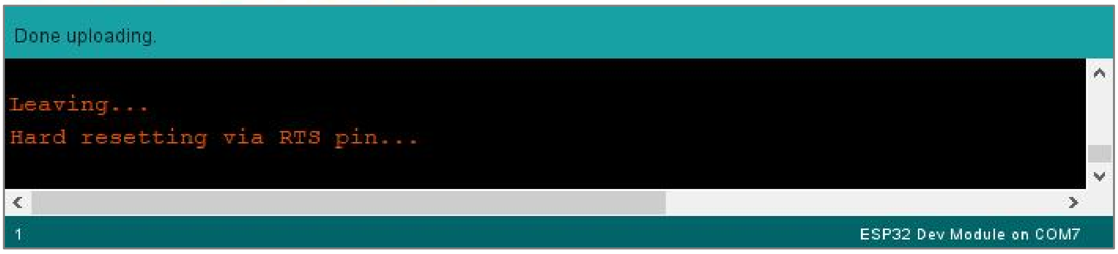
```
1                                    ESP32 Dev Module 在 COM5
```

8) Then click on ✓ icon to verify the program. If no error, the status area will display “Compiling->Compile complete” in turn. After compiling, the information such as the current used bytes, and occupied program storage space will be displayed.



```
Done compiling.

Sketch uses 271189 bytes (20%) of program storage space. Maximum is 1310720 bytes.
Global variables use 17564 bytes (5%) of dynamic memory, leaving 310116 bytes for local v

SP32 Dev Module, Disabled, Default 4MB with spiffs (1.2MB APP/1.5MB SPIFFS), 240MHz (WiFi/BT), QIO, 80MHz, 4MB (32Mb), 921600, Core 1, Core 1, None on COM7
```

9) After compiling, click on ➔ icon to upload the program to the development board. The status area will display “Compiling->Uploading->Complete” in turn. After uploading, the status area will stop printing the uploading information.



```
Leaving...
Hard resetting via RTS pin...
```

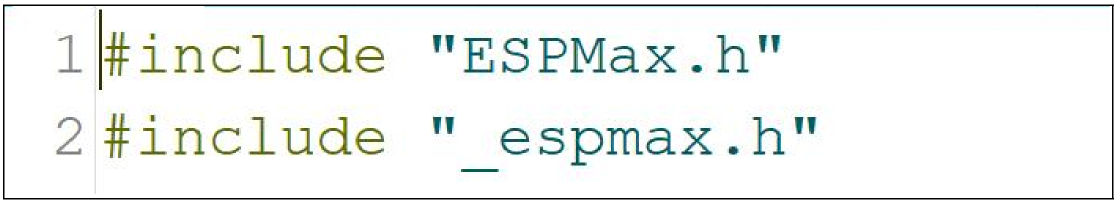
4. Project Outcome

When running the program, the robotic arm will move up and down along z axis, then move to the left and the right along x axis, and then move forwards and backwards along y axis. After the program stops, exit automatically.

5. Program Instruction

5.1 Import Library File

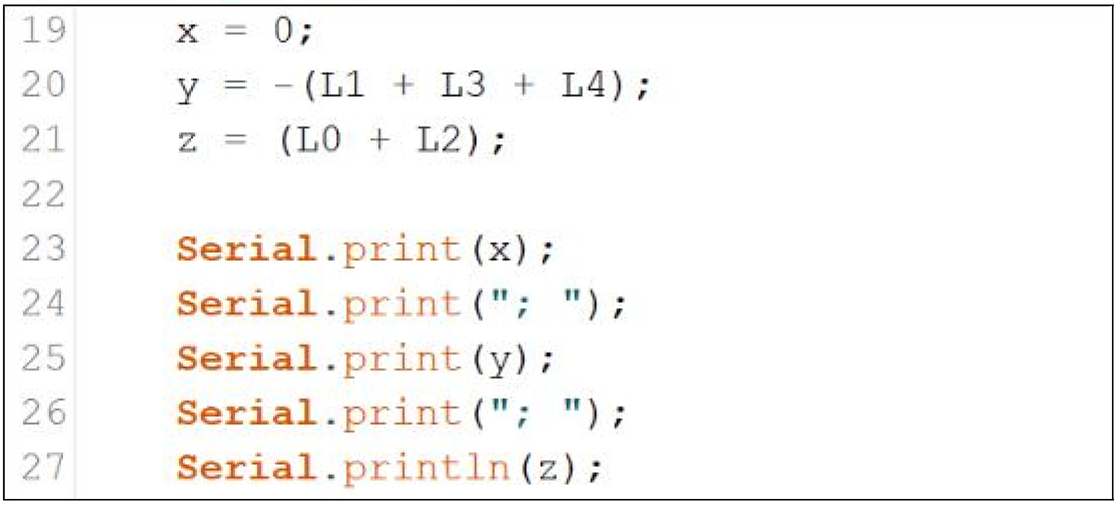
Before the robotic arm starts to move, the Python function libraries related to the kinematics and bus servo need to be imported.



```
#include "ESPMax.h"
#include "_espmax.h"
```

5.2 Read position

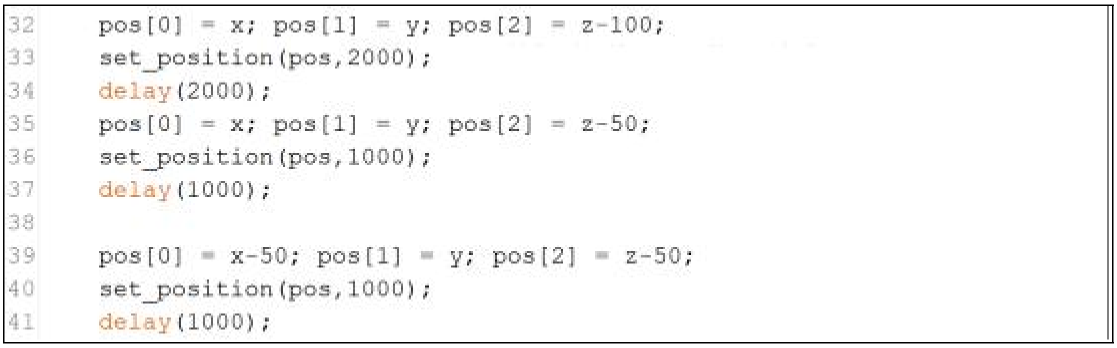
The initial position xyz of the robotic arm is calculated, and the value of “xyz” is printed.



```
    x = 0;
    y = -(L1 + L3 + L4);
    z = (L0 + L2);

    Serial.print(x);
    Serial.print("; ");
    Serial.print(y);
    Serial.print("; ");
    Serial.println(z);
```

5.3 Control Robotic Arm

Robotic arm starts to move on xyz axes.



```
    pos[0] = x; pos[1] = y; pos[2] = z-100;
    set_position(pos,2000);
    delay(2000);
    pos[0] = x; pos[1] = y; pos[2] = z-50;
    set_position(pos,1000);
    delay(1000);

    pos[0] = x-50; pos[1] = y; pos[2] = z-50;
    set_position(pos,1000);
    delay(1000);
```

The robotic arm is controlled by set_position() function. Take the code “set_position(pos,2000)” as example:

The first parameter “pos” represents the position of the robotic arm on x, y and z axes. Among them, pos[0] represents the coordinate of x axis, pos[1] represents the coordinate of y axis, and pos[2] represents the coordinate of z-axis.

The second parameter “200” represents the running time and the unit is ms.