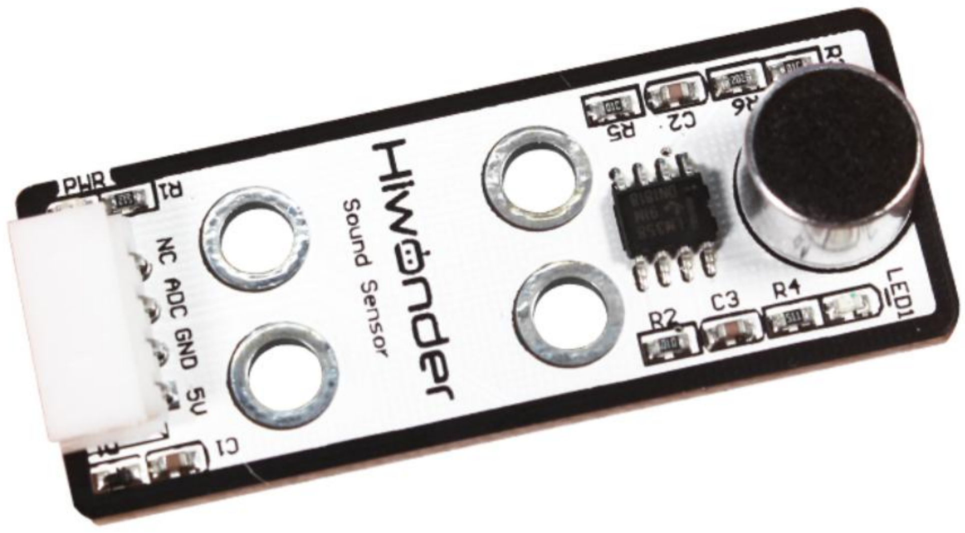
Lesson 2 Sound Detection and Placement

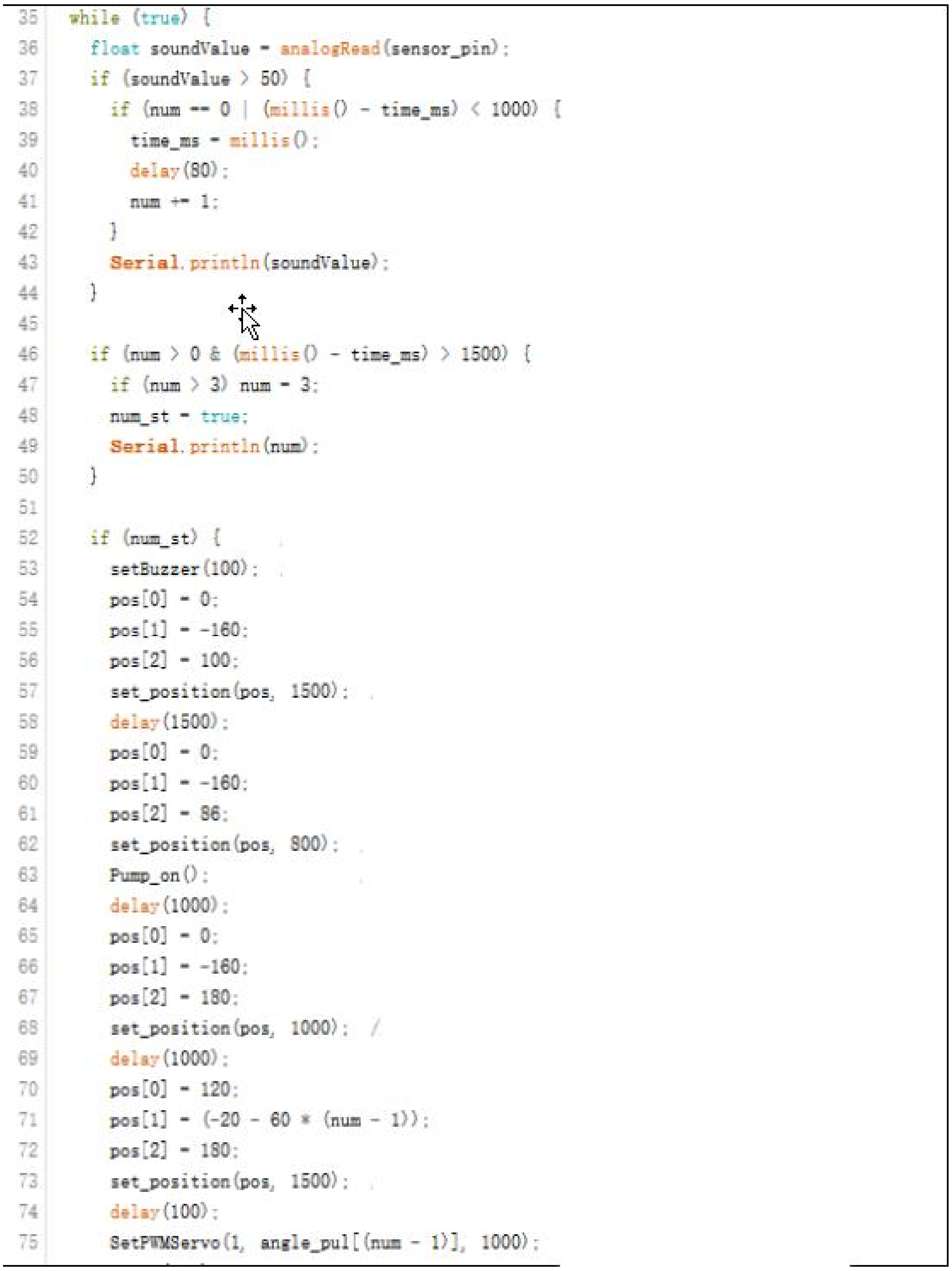
1. Project Principle

Sound sensor has a in-built capacitive microphone. Sound waves cause the diaphragm in the microphone to vibrate, resulting in capacitance change, which in turn produces a correspondingly small change in voltage. This voltage is then converted into a voltage in the range of 0-5 V and compared with an adjustable voltage with adjusted sensitivity by means of a comparator.

Then it is received by the data collector through the A/D conversion of module, and the range is 0-1023. The value is higher as the sound intensity increases, so the detected sound is proportional to the output analog quantity.



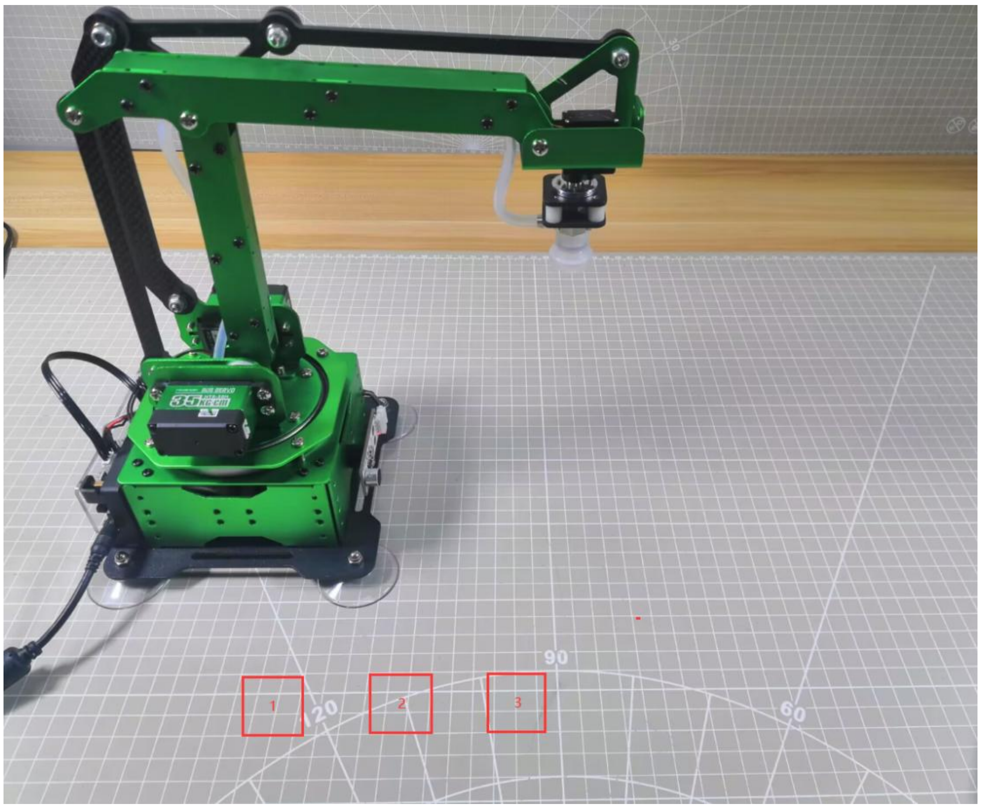
The path of the program file: “6. Secondary Development/ Arduino Development/Sensor-extension Game/Program Files/ Sound Detection and Placement/ SoundSensor_put/SoundSensor_put.ino”



```
while (true) {
  float soundValue = analogRead(sensor_pin);
  if (soundValue > 50) {
    if (num == 0 | (millis() - time_ms) < 1000) {
      time_ms = millis();
      delay(80);
      num += 1;
    }
    Serial.println(soundValue);
  }

  if (num > 0 & (millis() - time_ms) > 1500) {
    if (num > 3) num = 3;
    num_st = true;
    Serial.println(num);
  }

  if (num_st) {
    setBuzzer(100);
    pos[0] = 0;
    pos[1] = -160;
    pos[2] = 100;
    set_position(pos, 1500);
    delay(1500);
    pos[0] = 0;
    pos[1] = -160;
    pos[2] = 86;
    set_position(pos, 800);
    Pump_on();
    delay(1000);
    pos[0] = 0;
    pos[1] = -160;
    pos[2] = 180;
    set_position(pos, 1000);
    delay(1000);
    pos[0] = 120;
    pos[1] = (-20 - 60 * (num - 1));
    pos[2] = 180;
    set_position(pos, 1500);
    delay(100);
    SetPWMServo(1, angle_pul[(num - 1)], 1000);
```

This game uses a sound sensor to detect and identify the number of knocks, and then send the signal to the microcontroller for processing. When a knock is detected, the blue LED on sound sensor will flash, and the buzzer will sound, and then the robot arm will execute the corresponding action.

As shown in the figure below, the robot arm will pick up and place the object to the first position when knock on table once; when knock on table twice, the object will be placed to the second position; when knock on table third, the object will be placed to the third position.



Then execute the functions for controlling action, buzzer, air pump to pick up and place object to the corresponding position in the left side (Take robotic arm as the first person view).

Note:

1. If you knock on table over three times, object will be placed in the third position.

2. The interval between the knocks must be within 1s. If the interval is over 1.5s, program will re-calculate and execute the following process.

3. Due to the swing of the robot arm, the specific object landing point may be deviated, but the overall range will be as shown above.

---

2. Preparation

2.1 Hardware

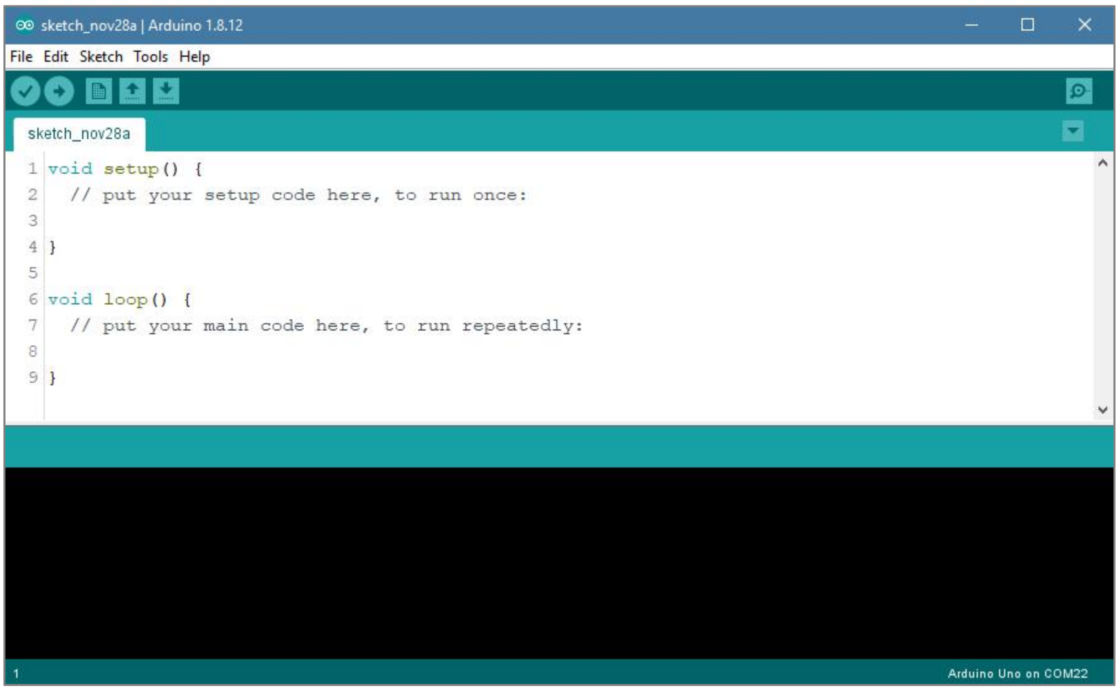
Please assemble sound sensor to the corresponding position on MaxArm according to the tutorial in folder “Lesson 1 Sensor Assembly” under the same directory.

2.2 Software

Please connect MaxArm to Arduino editor according to the tutorial in folder “4. Underlying Program Learning/Arduino Development/Lesson 1 Set Development Environment”.

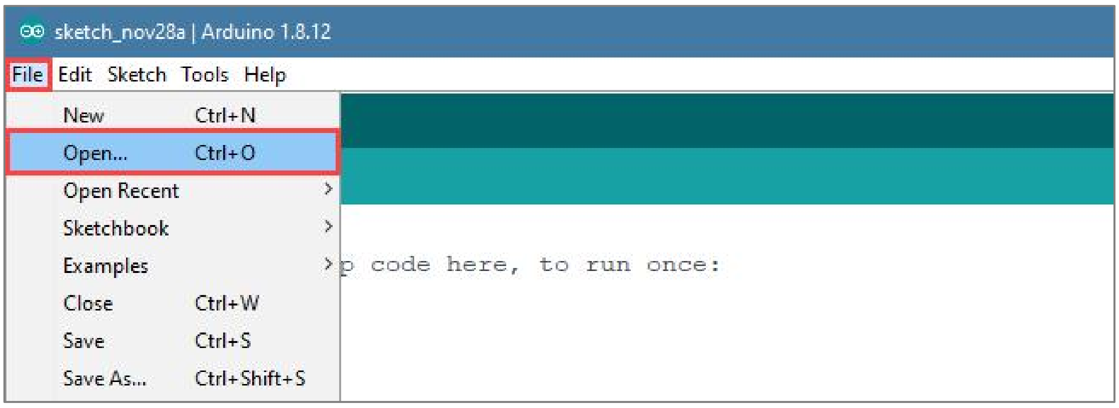
3. Program Download

1) Click on icon to open Arduino IDE.



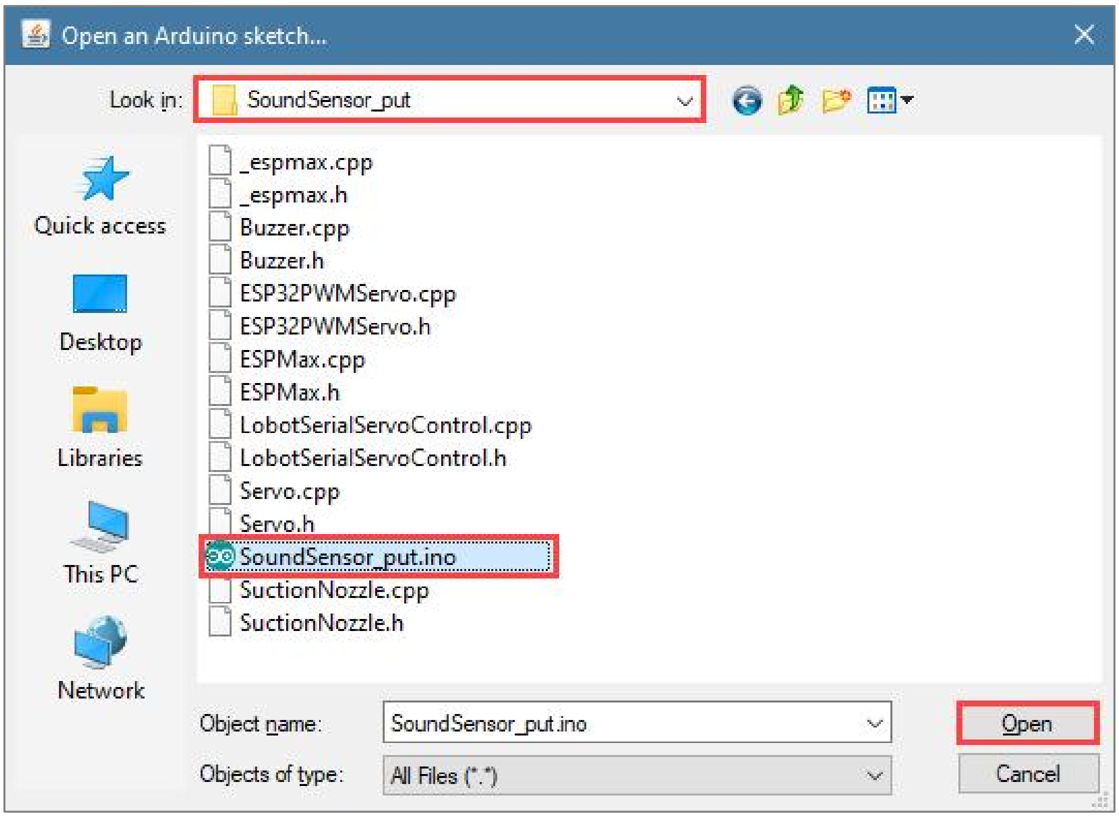
```
void setup() {
  // put your setup code here, to run once:

}

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

}
```

2) Click “File->Open” in turn.

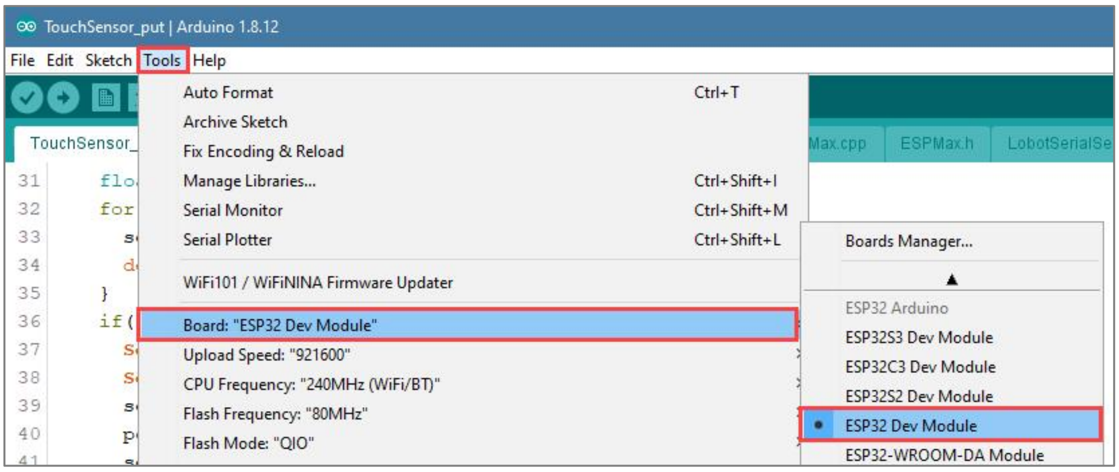


```
p code here, to run once:
```

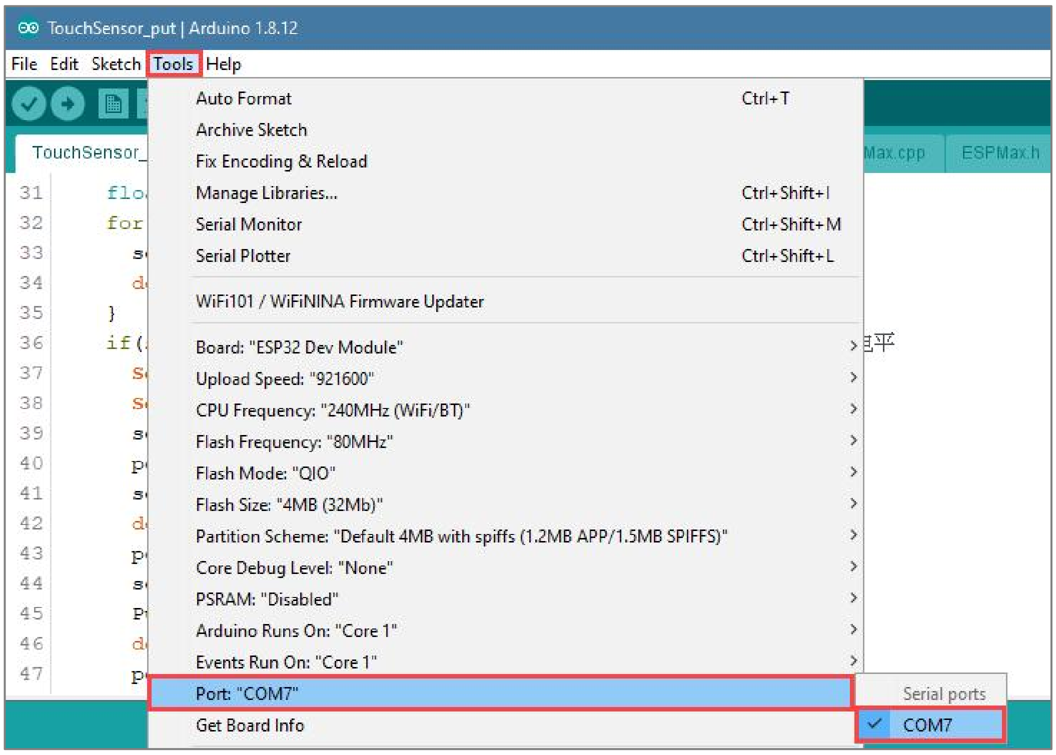
3) Open the program “SoundSensor_put.ino” in the folder “6.Secondary Development/Sensor-extension Game/Arduino Development/Program Files/ Sound Detection and Placement/ SoundSensor_put”.



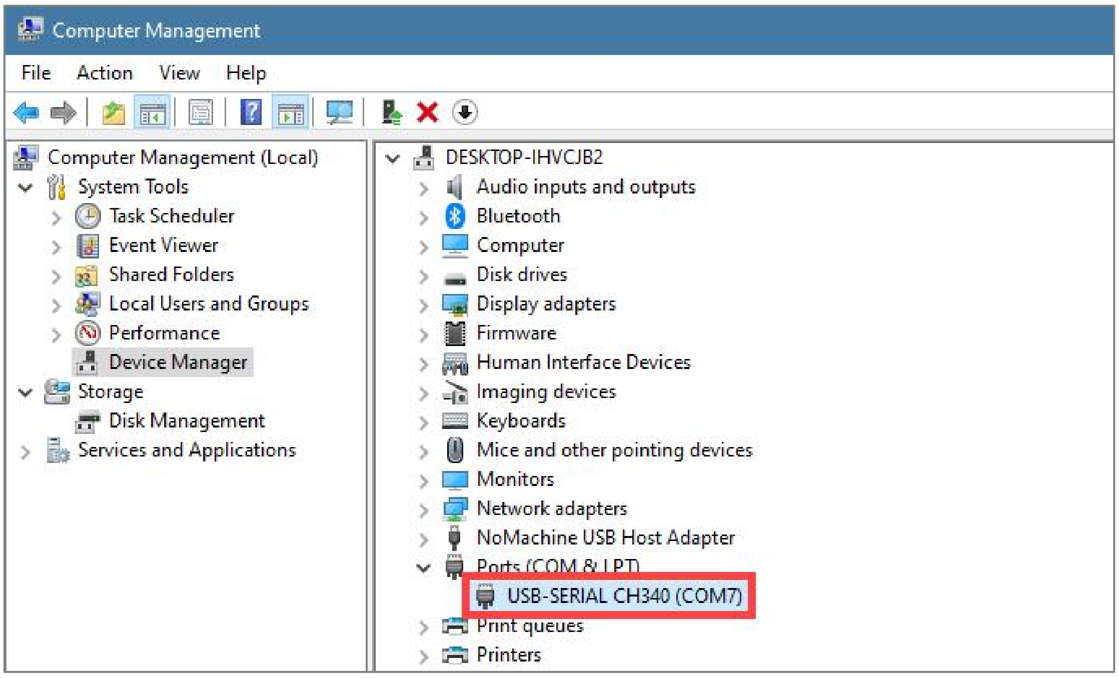
4) Select the model of development board. Click “Tools-> Board” and select “ESP 32 Dev Module” (If the model of the development board has been configured when setting the development environment, you can skip this step).


TouchSensor_put | Arduino 1.8.12
File Edit Sketch Tools Help
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
Board: "ESP32 Dev Module"
Upload Speed: "921600"
CPU Frequency: "240MHz (WiFi/BT)"
Flash Frequency: "80MHz"
Flash Mode: "QIO"
Boards Manager...
ESP32 Arduino
ESP32S3 Dev Module
ESP32C3 Dev Module
ESP32S2 Dev Module
ESP32 Dev Module
ESP32-WROOM-DA Module

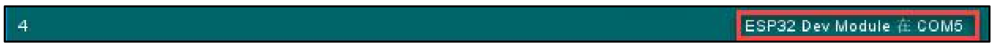
5) Select the corresponding port of Arduino controller in “Tools->Port”. (Here take the port “COM5” as example. Please select the port based on your computer. If COM1 appears, please do not select because it is the system communication port but not the actual port of the development port.)


TouchSensor_put | Arduino 1.8.12
File Edit Sketch Tools Help
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
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"
Get Board Info
Serial ports
COM7

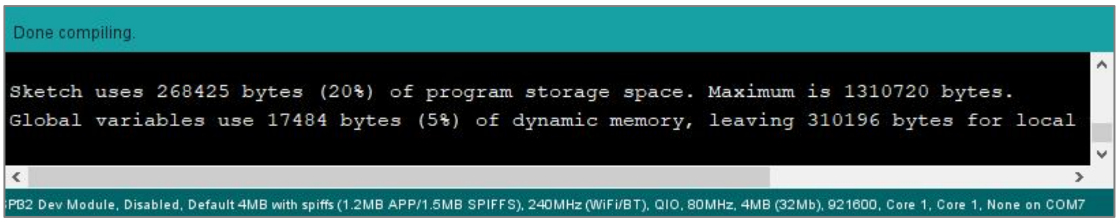
6) If you’re not sure about the port number, please open the “This PC” and click “Properties->Device Manger” in turns to check the corresponding port number (the device is with CH340). Then select the correct port on Arduino editor.



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).



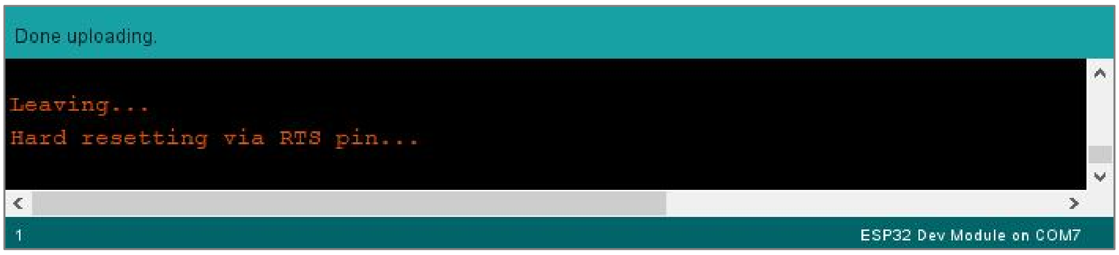
8) Then click on icon to verify the program. If no error, the status area will display “Compiling->Compile complete” in turn. After compiling, the information such as the current used bytes, and occupied program storage space will be displayed.


Done compiling.

```
Sketch uses 268425 bytes (20%) of program storage space. Maximum is 1310720 bytes.
Global variables use 17484 bytes (5%) of dynamic memory, leaving 310196 bytes for local
```

```
P32 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.



```
Done uploading.

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

1                                                    ESP32 Dev Module on COM7
```

4. Project Outcome

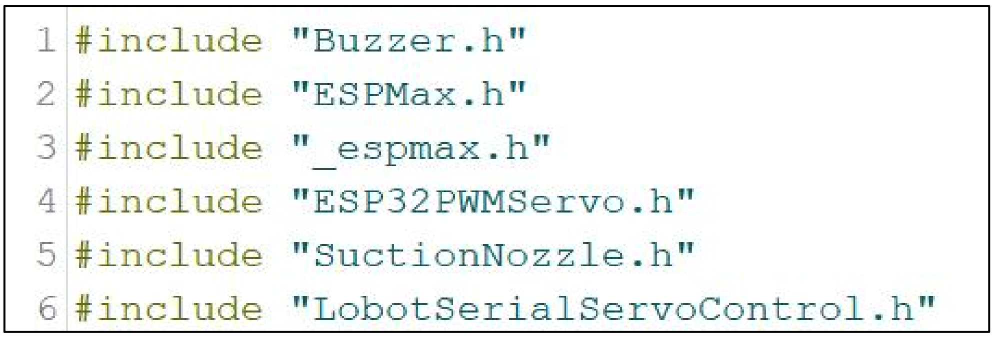
After the knock sound is detected, the blue LED light on sensor will flash corresponding to the number of knocks, then the robot arm rotates to the front and turns on the air pump to suck the block, moves and paces it to the position 1, 2, 3 of the placement area corresponding to the number of knocks. After that, turn off the air pump, reset and wait for the next command.

5. Program Instruction

5.1 Import function library and Initialize

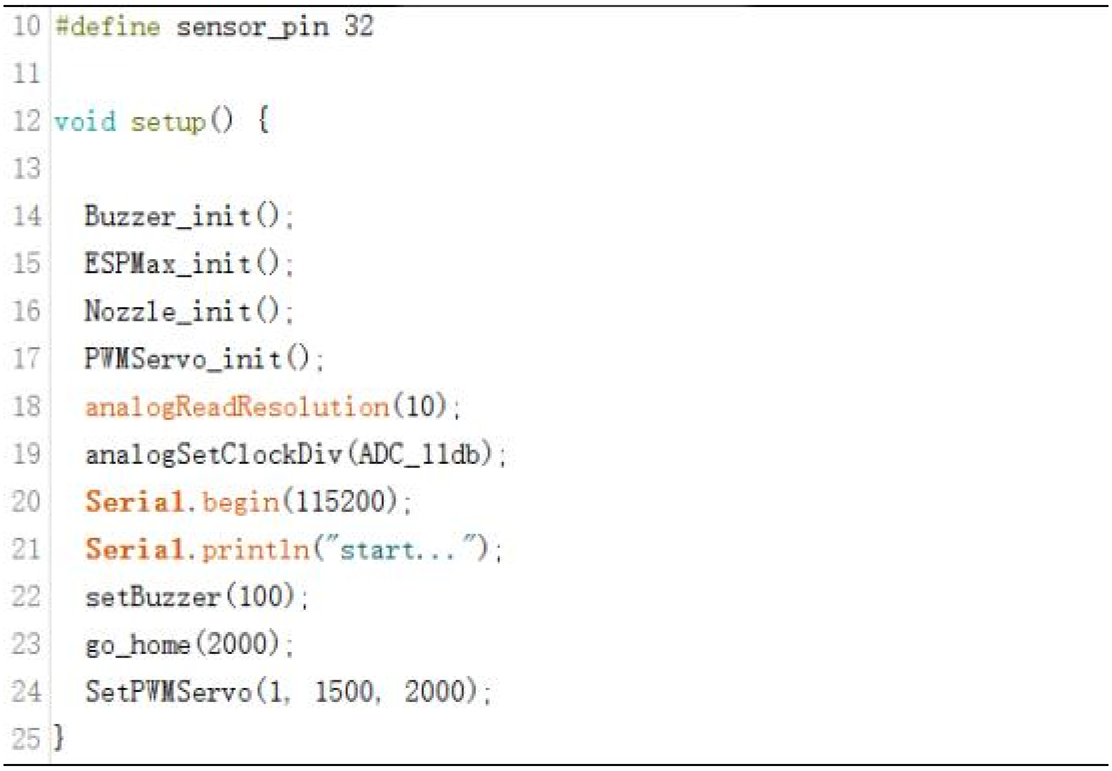
The path of the program file: “6. Secondary Development/Sensor-extension Game/ Arduino Development/Program Files/ Sound Detection and Placement/ SoundSensor_put\SoundSensor_put.ino”. If the program is modified, you can find a backup file in Appendix.

Before running the program, the buzzer, kinematics, kinematics encapsulation library, PWM servo, suction nozzle and other related library files need to be imported first.



```
#include "Buzzer.h"
#include "ESPMax.h"
#include "_espmax.h"
#include "ESP32PWMServo.h"
#include "SuctionNozzle.h"
#include "LobotSerialServoControl.h"
```

Then, initialize the library file and robotic arm.



```
#define sensor_pin 32

void setup() {

  Buzzer_init();
  ESPMax_init();
  Nozzle_init();
  PWMServo_init();
  analogReadResolution(10);
  analogSetClockDiv(ADC_11db);
  Serial.begin(115200);
  Serial.println("start...");
  setBuzzer(100);
  go_home(2000);
  SetPWMServo(1, 1500, 2000);
}
```

5.2 Sound Detection

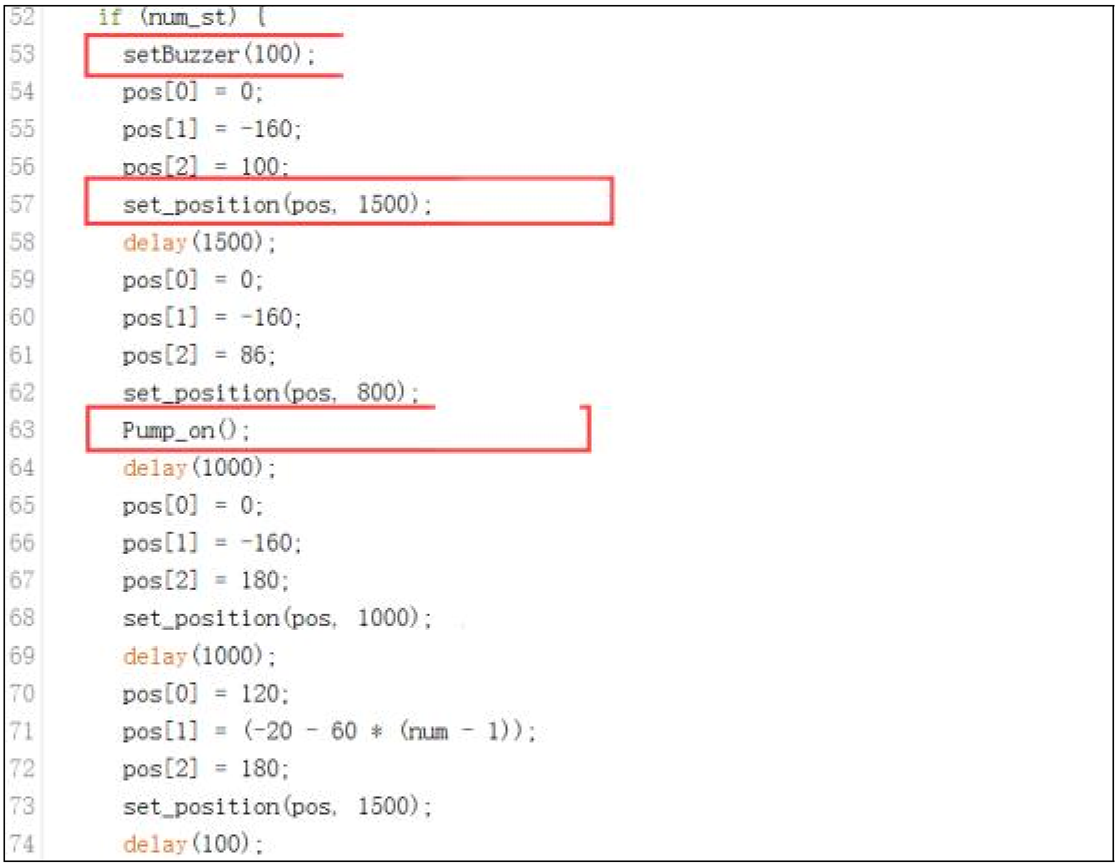
Use analogRead() function to read the analog amount of detected knocks, and millis() function to read the sound interval time. if the analog amount reaches the set threshold and the knock interval is less than 1000ms, then the number of knocks plus one. If the number of detected knocks is greater than 3, then it will also be determined as only three knocks.

```
void loop() {
  int num = 0;
  float pos[3];
  bool num_st = false;
  long int time_ms = millis();
  int angle_pul[3] = { 1800, 2000, 2200 };

  while (true) {
    float soundValue = analogRead(sensor_pin);
    if (soundValue > 50) {
      if (num == 0 | (millis() - time_ms) < 1000) {
        time_ms = millis();
        delay(80);
        num += 1;
      }
      Serial.println(soundValue);
    }

    if (num > 0 & (millis() - time_ms) > 1500) {
      if (num > 3) num = 3;
      num_st = true;
      Serial.println(num);
    }
```

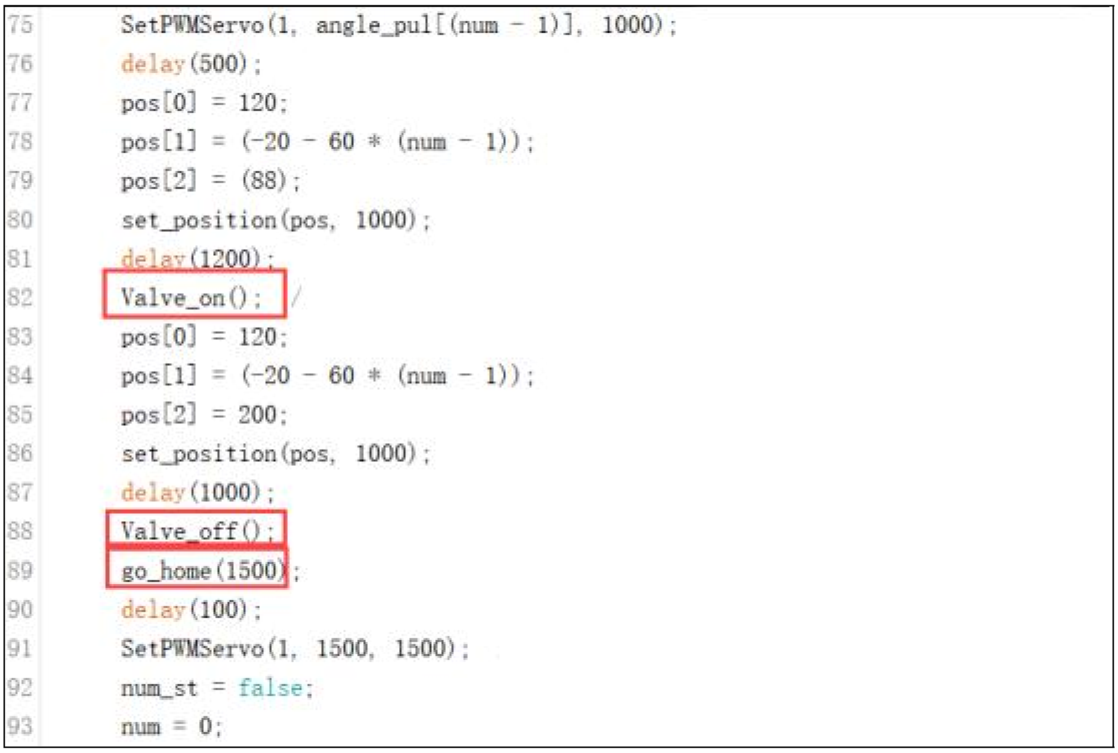
5.3 Sound Detection Feedback

When the knock sound is detected by sensor, the blue LED will flash corresponding times, and then MaxArm will execute the corresponding action.

When the sensor detects a knock, the blue LED will flash the corresponding number of times, and then MaxArm will perform the corresponding action.



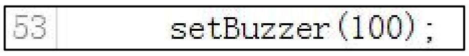
```
  if (num_st) {
    setBuzzer(100);
    pos[0] = 0;
    pos[1] = -160;
    pos[2] = 100;
    set_position(pos, 1500);
    delay(1500);
    pos[0] = 0;
    pos[1] = -160;
    pos[2] = 86;
    set_position(pos, 800);
    Pump_on();
    delay(1000);
    pos[0] = 0;
    pos[1] = -160;
    pos[2] = 180;
    set_position(pos, 1000);
    delay(1000);
    pos[0] = 120;
    pos[1] = (-20 - 60 * (num - 1));
    pos[2] = 180;
    set_position(pos, 1500);
    delay(100);
```



```
    SetPWMServo(1, angle_pul[(num - 1)], 1000);
    delay(500);
    pos[0] = 120;
    pos[1] = (-20 - 60 * (num - 1));
    pos[2] = (88);
    set_position(pos, 1000);
    delay(1200);
    Valve_on(); /
    pos[0] = 120;
    pos[1] = (-20 - 60 * (num - 1));
    pos[2] = 200;
    set_position(pos, 1000);
    delay(1000);
    Valve_off();
    go_home(1500);
    delay(100);
    SetPWMServo(1, 1500, 1500);
    num_st = false;
    num = 0;
```

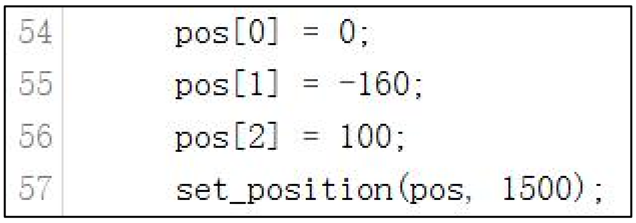
Take setBuzze() function, set_position() function, Pump_on() function and go_home() for example.

The setBuzze() is a function to control buzzer. Call “Buzzer.h” function in the same directory as “InfraredSensor_sorting.ino” program. Fill in the parentheses with the duration time of buzzer, and the unit is ms. The code "setBuzzer(100)" means that the buzzer will respond for 100ms after the infrared sensor detects an object, and then the next set_potision () function will be executed.



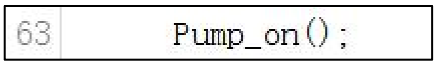
```
        setBuzzer(100);
```

The set_position() is a function to call robotic arm. Call “**ESPMax.h**” function in the same folder as “**SoundSensor_put.ino**” program. Fill in the parentheses with the coordinate values and duration time. Take the code "set_position(pos,1500)" for example, pos[0], pos[1], pos[2] is the representative of the robot arm corresponding to the position on the XYZ axis, 1500 is the running time, the unit is milliseconds (ms). After that, the Pump_on() function is executed in the next step.

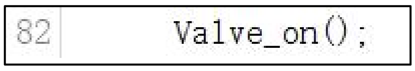


```
        pos[0] = 0;
        pos[1] = -160;
        pos[2] = 100;
        set_position(pos, 1500);
```

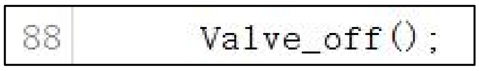
The Pump_on() is a function to control air pump. Pump_on () function is to control the air pump on; Valve_on () function is used to turn off air pump and open solenoid valve; Valve_off () function to turn off the solenoid valve. Call "**SoundSensor_put.ino**" function in the same folder as the "SuctionNozzle.h" function, after the implementation of the next go_home () function.



```
        Pump_on();
```

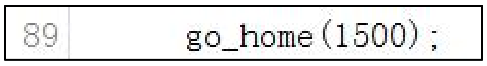


```
        Valve_on();
```



```
        Valve_off();
```

The go_home() is a function to reset robotic arm. Cal “ESPMax.h” function in the same folder as “SoundSensor_put.ino” function.Fill in the parentheses with the action duration time, and the unit is ms. Take the code “go_home(1500)” for example. The parameter “1500” is the time it takes to complete the reset action. When this function is finished, the program will execute 1500ms delay function and then end this recognition and sorting actions and wait for the next recognition signal.



```
        go_home(1500);
```