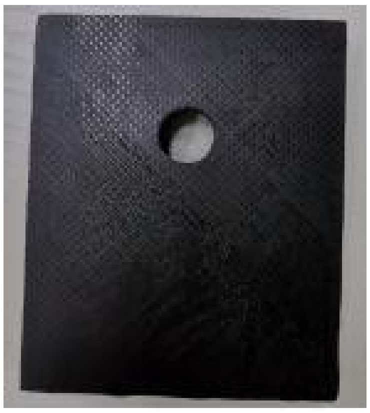
Lesson 2 Light Detection and Placement

Please prepare a light-blocking prop for this game. You can use 3D printer to print a prop shown in the following image or use hard material to make one.



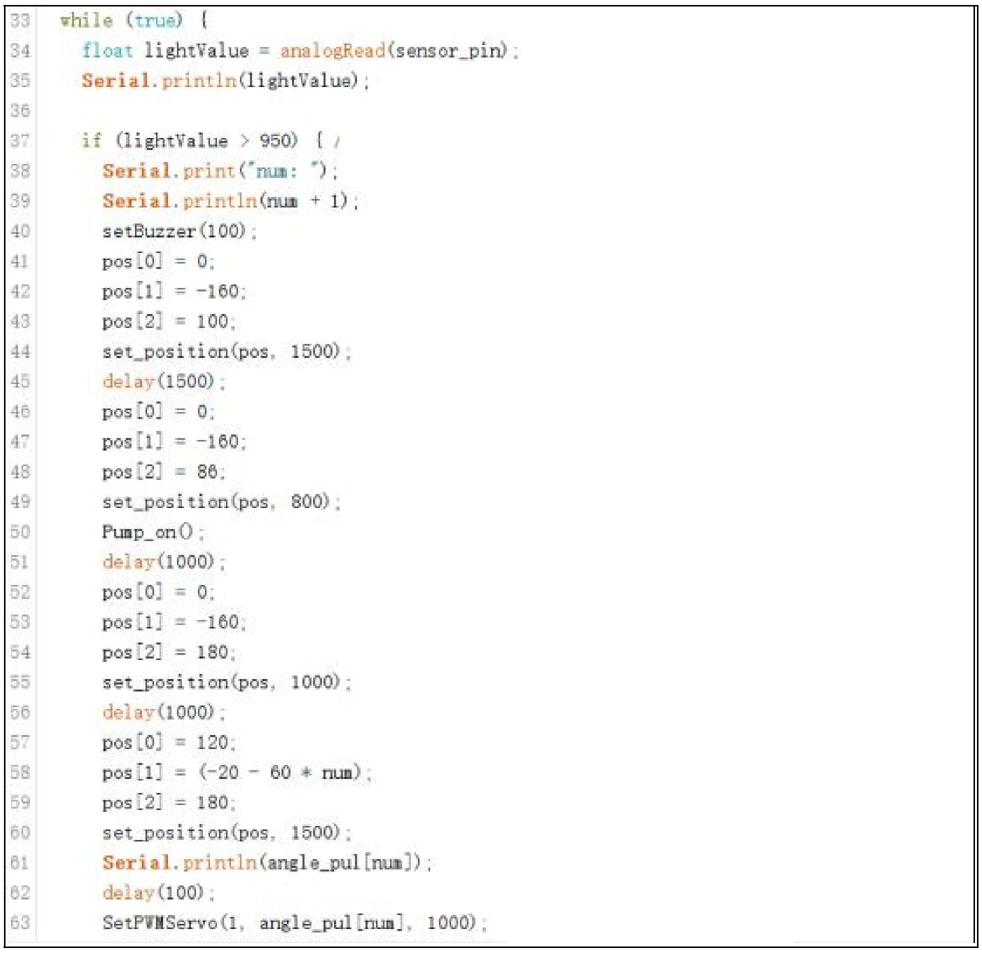
1. Project Principle

This game uses light sensor to sense the ambient light intensity to judge if it is covered by object. Then robot arm is controlled to place object in the corresponding area.

A photosensitive sensor is a sensitive device that responds or converts to external light signals or light radiation. The sensor mainly contains a QT523C (photodiode) and an LM358 chip (voltage comparator).

When it works, sensor uses QT523C to convert light signal to an electrical signal output, which is then converted to a voltage ranging 0 to 5V and received by the data collector after A/D conversion in the range of 0-1023. The greater the external brightness intensity, the smaller the output voltage, so the brightness is inversely proportional to the output voltage.

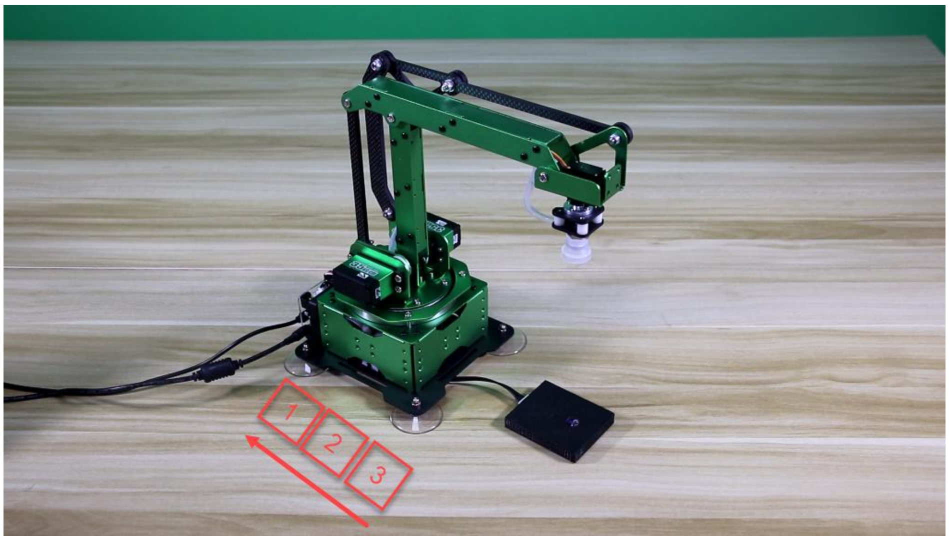
The path of the program file: “6. Secondary Development/ Sensor-extension Game/ Arduino Development/ Program Files/ Light Detection and Placement/ LightSensor_put/LightSensor_put.ino ”



```
while (true) {
  float lightValue = analogRead(sensor_pin);
  Serial.println(lightValue);

  if (lightValue > 950) { /
    Serial.print("num: ");
    Serial.println(num + 1);
    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);
    pos[2] = 180;
    set_position(pos, 1500);
    Serial.println(angle_pul[num]);
    delay(100);
    SetPWMServo(1, angle_pul[num], 1000);
```

Use light sensor to detect the light intensity, and then send the signal to microcontroller for processing. When the light intensity is lower than the set threshold in program, buzzer will sound for responding, and MaxArm will perform the corresponding action.

The first detected object will be placed at position 1. The second detected object will be placed at position 2. The third detected object will be place at position 3. As the figure shown below:



Then, execute the functions for controlling action, buzzer and air pump to control robot arm to move object and place it to the corresponding position on the left side. (Take robotic arm as the first person view)

Note:

1. Since light sensor is susceptible to the ambient light, you may need to adjust the potentiometer of sensor based on the actual situation.

2. 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.

3. When strong light is detected, the blue LED light on sensor will keep on. As the light becomes weak, the blue LED light will be off. Please note that this has nothing to do with the brightness of the sensor being covered by the object.

---

2. Preparation

2.1 Hardware

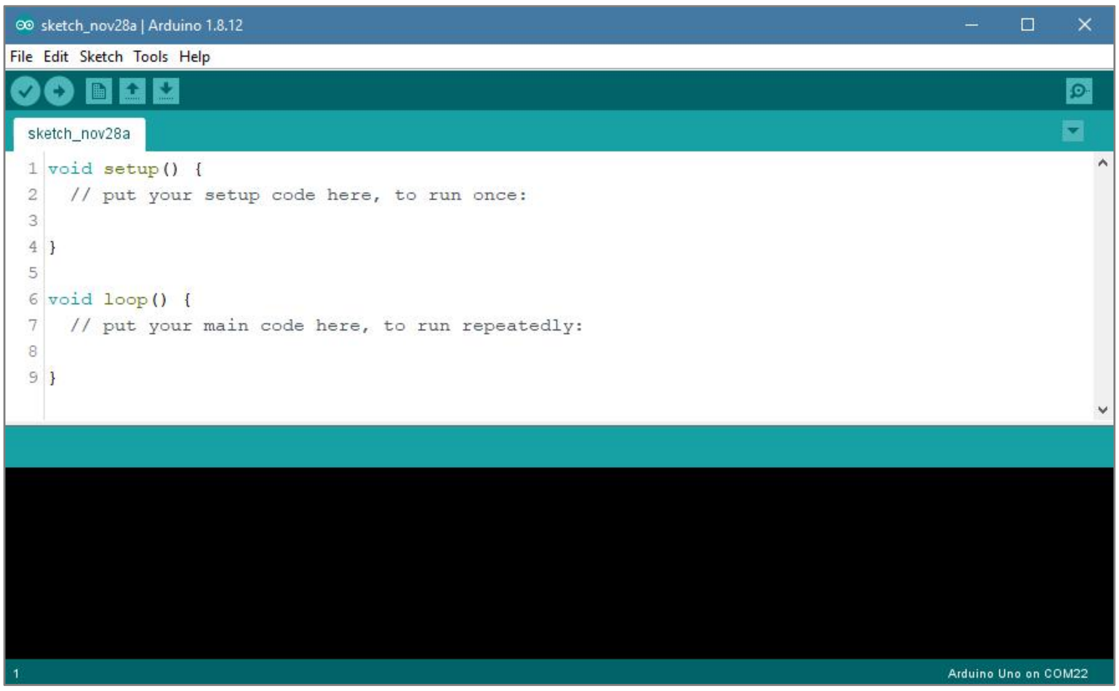
Please assemble light sensor to the corresponding position on MaxArm according to the tutorial in folder “Lesson 1 Sensor Assembly” in the same directory.

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”.

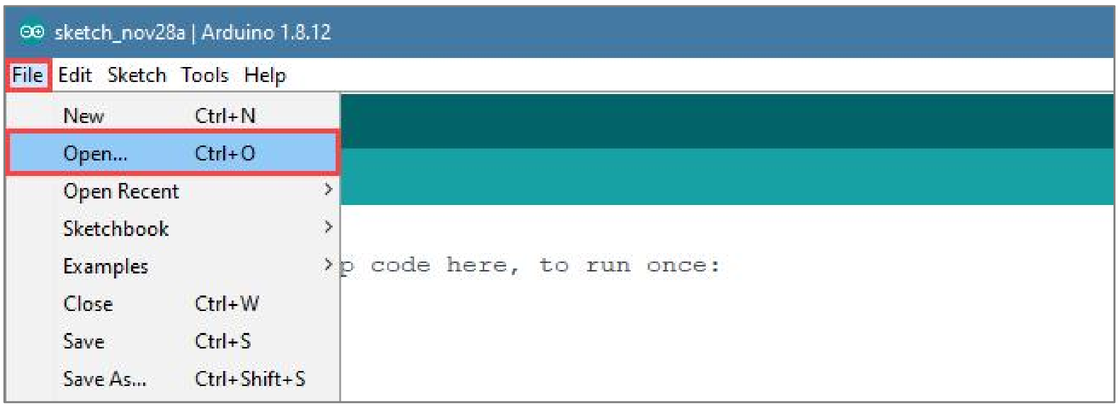
3. Program Download

1) Click on icon to open Arduino IDE.



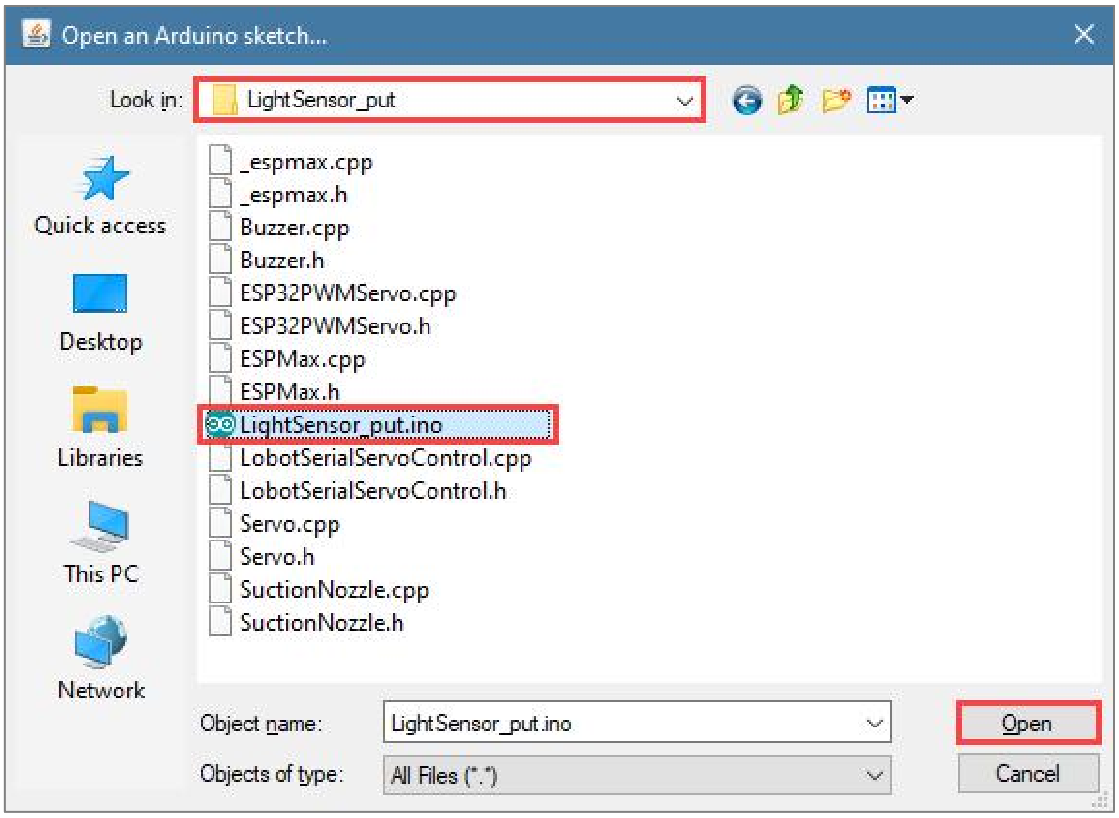
```
void setup() {
  // put your setup code here, to run once:

}

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 “LightSensor_put.ino” in the folder “6.Secondary Development/ Sensor-extension Game/Arduino Development/Program Files/ Light Detection and Placement/ LightSensor_put”.



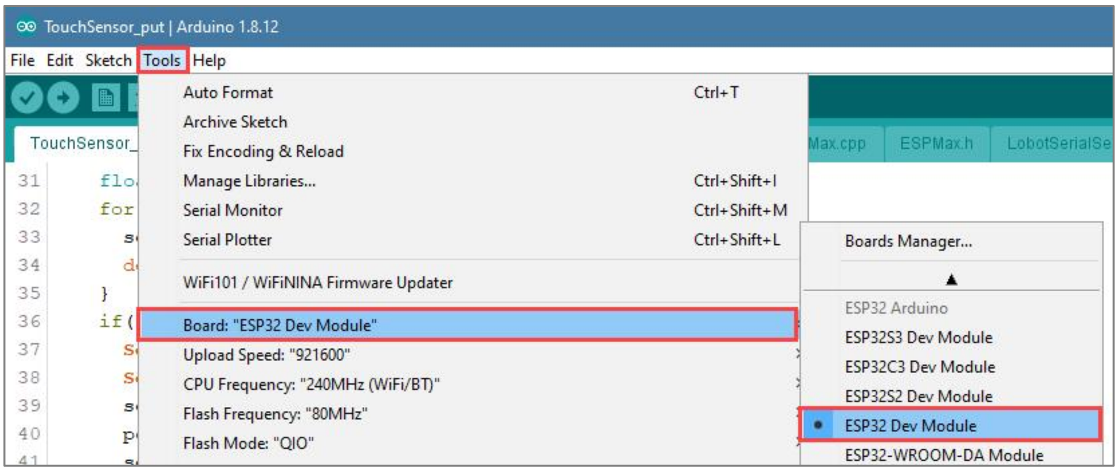
Look in: LightSensor_put

_espmax.cpp
_espmax.h
Buzzer.cpp
Buzzer.h
ESP32PWMServo.cpp
ESP32PWMServo.h
ESPMax.cpp
ESPMax.h
LightSensor_put.ino
LobotSerialServoControl.cpp
LobotSerialServoControl.h
Servo.cpp
Servo.h
SuctionNozzle.cpp
SuctionNozzle.h

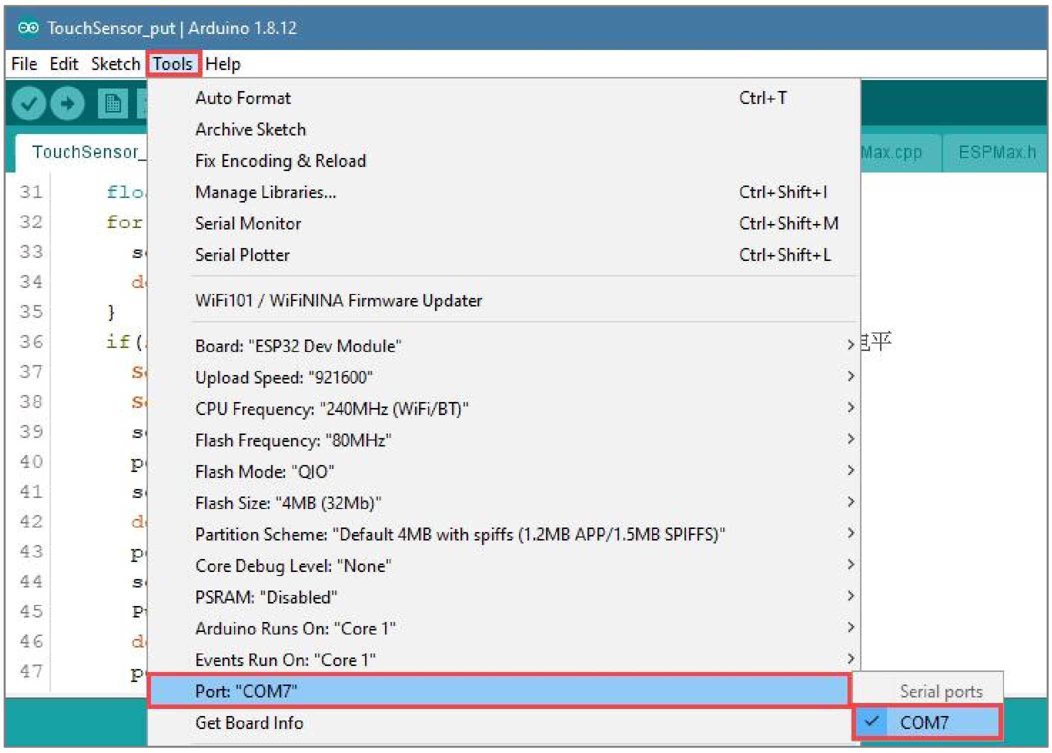
Object name: LightSensor_put.ino

Objects of type: All Files (*.*)

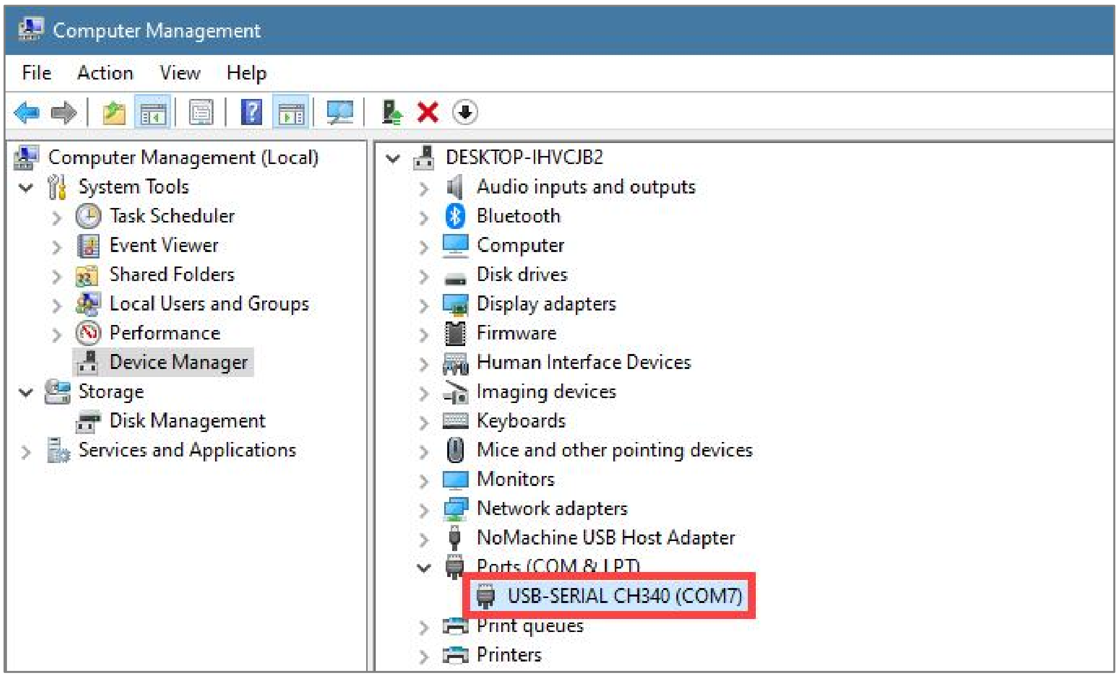
4) Select the model of development board. Click “Tools-> Board” and select “ESP 32 Dev Module” (If the model of the development board has been configured when setting the development environment, you can skip this step).


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

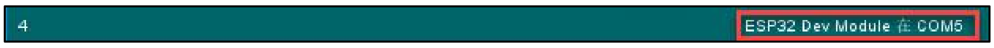
5) Select the corresponding port of Arduino controller in “Tools->Port”. (Here take the port “COM5” as example. Please select the port based on your computer. If COM1 appears, please do not select because it is the system communication port but not the actual port of the development port.)


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

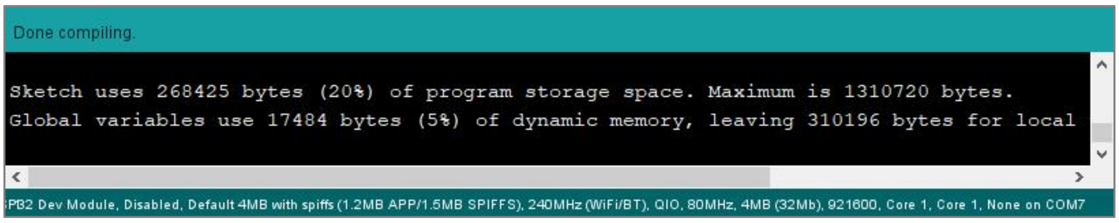
6) If you’re not sure about the port number, please open the “This PC” and click “Properties->Device Manger” in turns to check the corresponding port number (the device is with CH340). Then select the correct port on Arduino editor.



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).

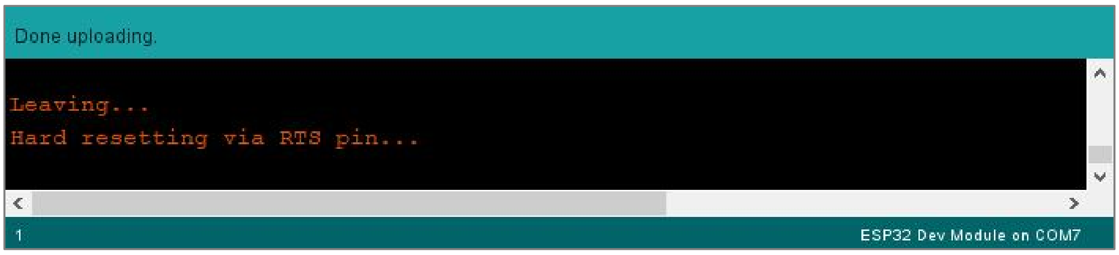


8) Then click on icon to verify the program. If no error, the status area will display “Compiling->Compile complete” in turn. After compiling, the information such as the current used bytes, and occupied program storage space will be displayed.



```
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

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.


Done uploading.

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

1 ESP32 Dev Module on COM7

4. Project Outcome

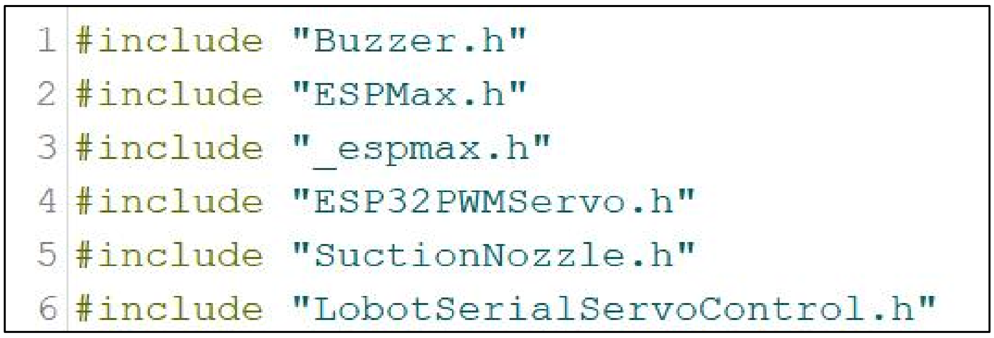
When light sensor starts to detecting light, it covered by object resulting in the light intensity is lower than the threshold set in program. At this time, microcontroller will receive a command, and then robot arm will turn on air pump and suck the object and place it to corresponding position. After that, turn off the air pump, reset and wait for the next command.

5. Program Instruction

5.1 Import function library and Initialize

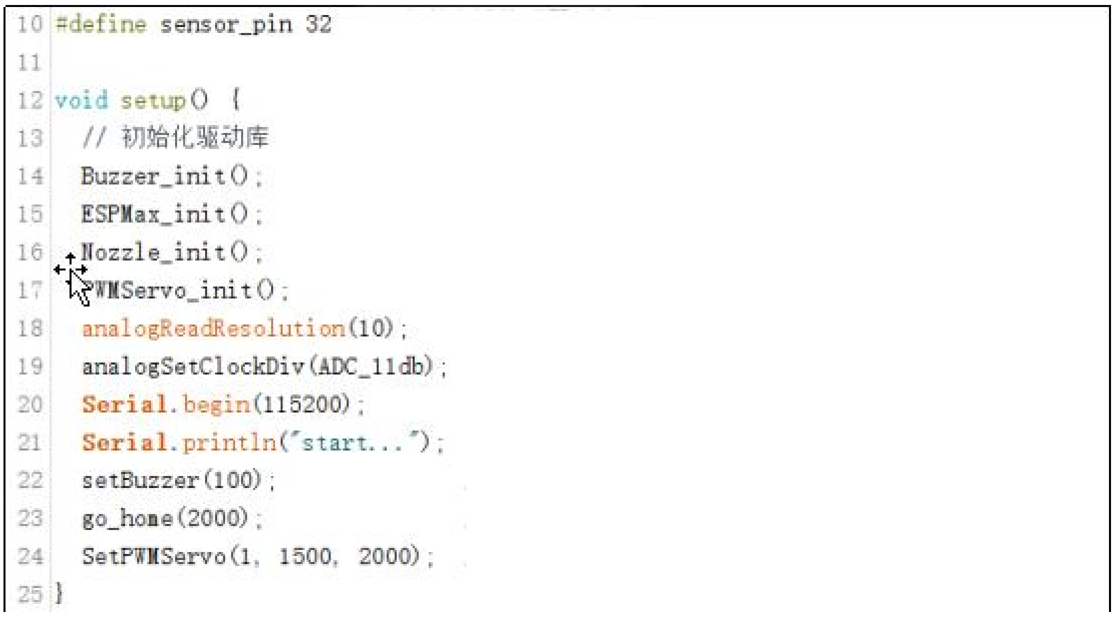
The path of the program file: "6. Secondary Development/ Sensor-extension Game/ Arduino Development/ Program Files/ light Detection and Placement/ LightSensor_put/LightSensor_put.ino". If the program is modified, you can find the backup file in Appendix.

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"
```

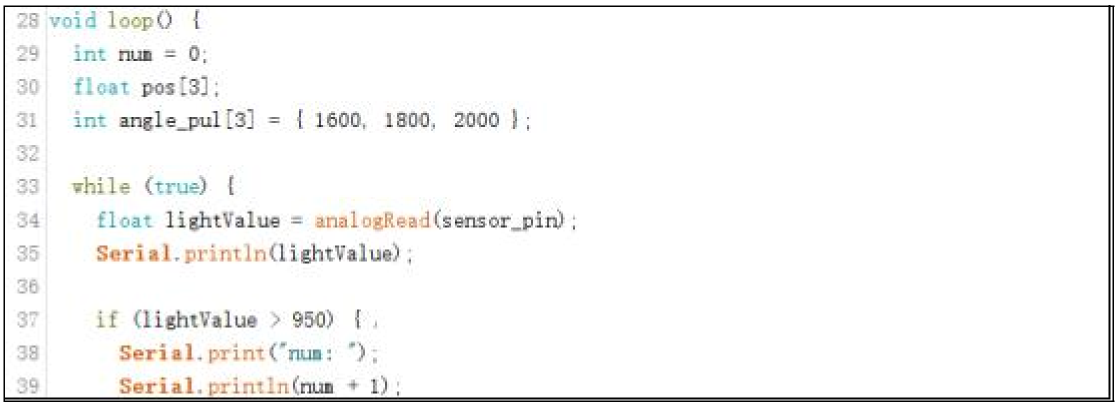
Then, initialize the library files and robotic arm.



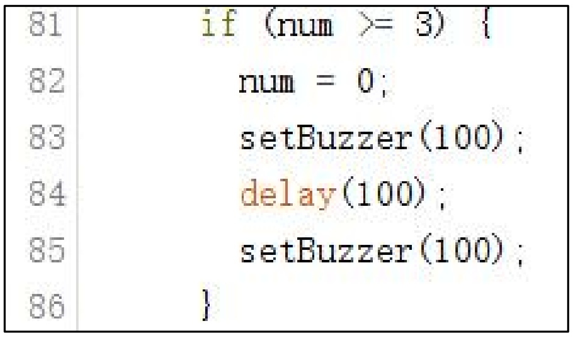
```
#define sensor_pin 32

void setup() {
  // 初始化驱动库
  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 Light Detection

Use analogRead() function to read the analog amount of detected light intensity. When the analog amount reaches the set threshold, the program will place object at position 1. When the next object is detected, it will be placed at position 2. After recognizing and placing three time, game will enter the next round.



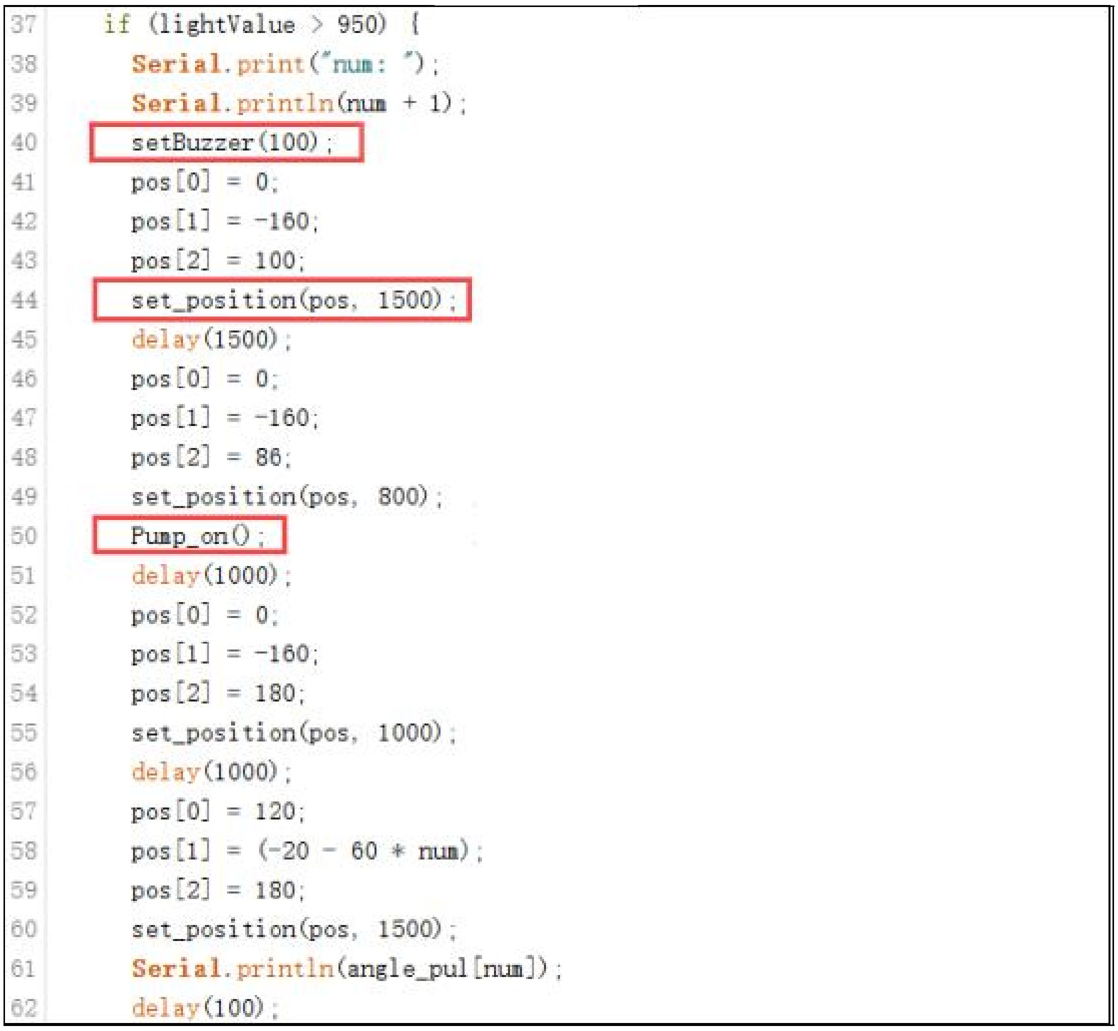
```
void loop() {
  int num = 0;
  float pos[3];
  int angle_pul[3] = { 1600, 1800, 2000 };

  while (true) {
    float lightValue = analogRead(sensor_pin);
    Serial.println(lightValue);

    if (lightValue > 950) {
      Serial.print("num: ");
      Serial.println(num + 1);
```



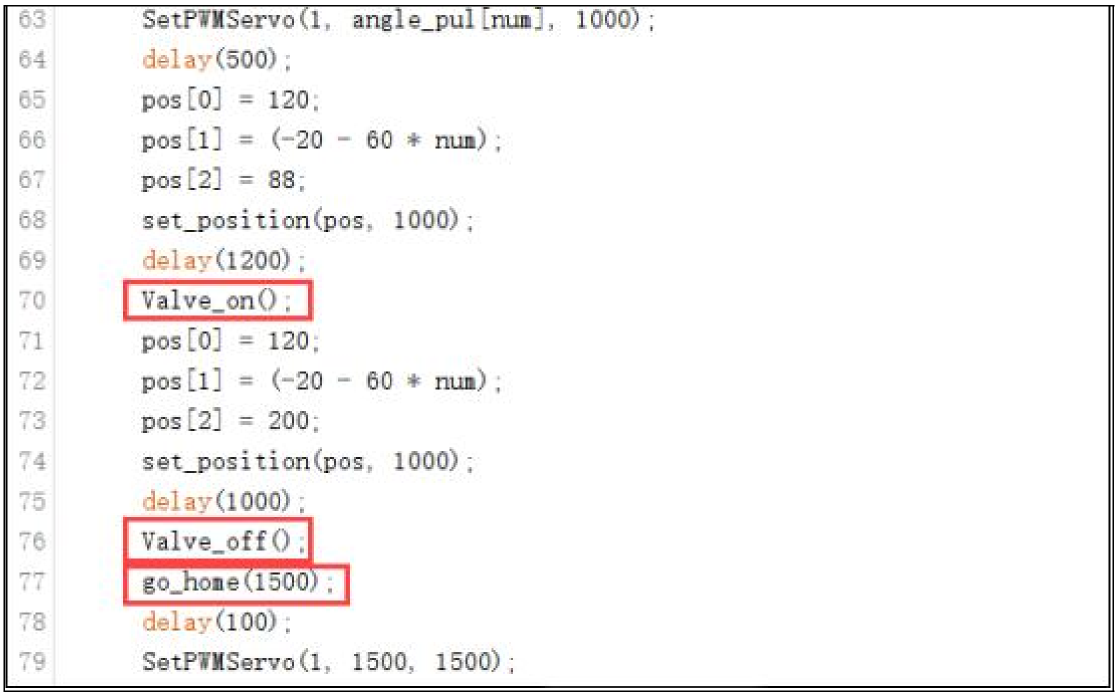
```
      if (num >= 3) {
        num = 0;
        setBuzzer(100);
        delay(100);
        setBuzzer(100);
      }
```

5.3 Light Detection Feedback

When cover is detected by sensor, MaxArm will execute the corresponding action.



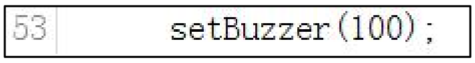
```
    if (lightValue > 950) {
      Serial.print("num: ");
      Serial.println(num + 1);
      setBuzzer(100);
      pos[0] = 0;
      pos[1] = -160;
      pos[2] = 100;
      set_position(pos, 1500);
      delay(1500);
      pos[0] = 0;
      pos[1] = -160;
      pos[2] = 86;
      set_position(pos, 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);
      pos[2] = 180;
      set_position(pos, 1500);
      Serial.println(angle_pul[num]);
      delay(100);
```



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

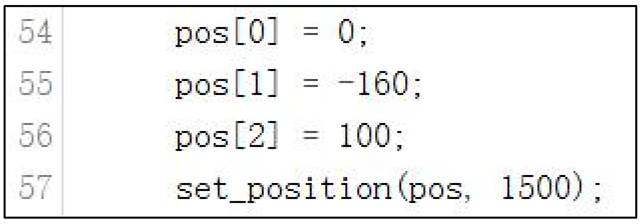
Take setBuzze() function, set_position() function, Pump_on() function and go_home() for example.

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.



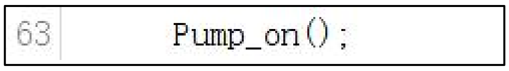
```
        setBuzzer(100);
```

The set_position() is a function to call robotic arm. Call “**ESPMax.h**” function in the same folder as “**LightSensor_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.

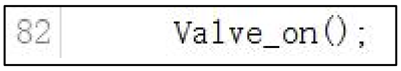


```
        pos[0] = 0;
        pos[1] = -160;
        pos[2] = 100;
        set_position(pos, 1500);
```

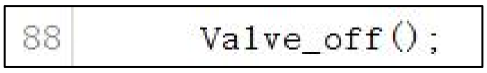
The Pump_on() is a function to control air pump. Pump_on () function is to control the air pump on; Valve_on () function is used to turn off air pump and open solenoid valve; Valve_off () function to turn off the solenoid valve. Call "LightSensor_put.ino" function in the same folder as the "SuctionNozzle.h"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



```
        Pump_on();
```

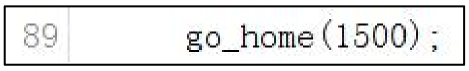


```
Valve_on();
```



```
Valve_off();
```

The go_home() is a function to reset robotic arm. Call “ESPMax.h” function in the same folder as “LightSensor_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 the game and wait for the next recognition signal.

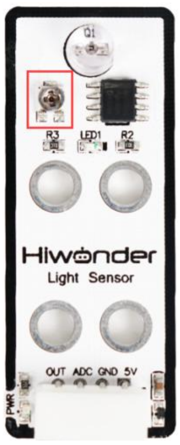


```
go_home(1500);
```

6. Adjust Sensitivity

There is an adjustable potentiometer knob on light sensor for adjusting the measuring distance. When performing the related games, if the measurement effect it not good enough, the measurement sensitivity of sensor can be adjusted by adjusting the knob.

It is recommended to use phillips screwdriver. Rotate the knob clockwise, as the figure shown below, to increase the measurement distance; rotate it counterclockwise to decrease the measurement distance.



Please take notice of the following two tips:

1) Since the warm light (incandescent light, sunlight, etc) contains more infrared component, there will be a certain interference the detection result. When using it, special attention should be paid to the surrounding environment. Therefore, you can not adjust the sensitivity alone.

2) The sensitivity adjustment has is set a threshold, which you can regard it as a critical point. If the sensitivity exceeds the critical point, its value will return to the initial state.

3) Sensitivity adjustment needs to be based on the actual needs of the project. It is better to be adjusted to the most suitable conditions.