AI Farm

EXPERIMENT

Programming Platform：Arduino
Hardware：Magician Lite+Sensor

DOBOT

Shake Hands With The Future

CATALOGUE

Experiment 1

Smart Sowing

Team name: Team member: Date:

Overview

In conventional farming, farmers generally have to bend to plough and sow. When sowing, they have to squat on the ground and place the seeds in the soil at an even distance, which is slow and tiring work. Nowadays, science and technology are becoming more and more advanced, people have invented high-tech sowers to overturn traditional sowing methods, as shown in Figure 1.1.

Figure 1.1 Two sowing methods

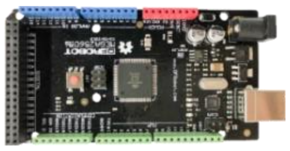
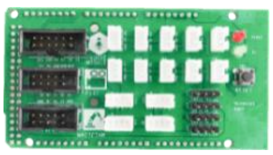
This experiment will control the robotic arm to automatically sow. Let's explore how to achieve it!

Objective

- Learn how to use the user interface (UI) of Arduino IDE with Arduino IDE.
- Learn how to use the Magician Lite-related APIs by programming the robotic arm to automatically sow three times.
- Learn how to use the judgment structure by programming the robotic arm to automatically sow three times.

- Learn how to use the loop structure by programming the robotic arm to automatically sow nine times.

Equipment

Equipment Picture	Name	Quantity
	Dobot Magician Lite robotic arm	1
	Suction cup kit	1
	Tape-C cable	1
	Power adapter	1
	Arduino Mega 2560 control board	1
	Arduino shield expansion board	1
	USB square port cable (Type-B cable)	1
	10-Pin DuPont adapter cable	1

Equipment Picture	Name	Quantity
	Farmland	1
	Seeds	Handful

Requirements

- ➢ Take care when using electricity.
- ➢ Before the experiment, check whether the experiment equipment is complete and intact. If there is any omission or damage, please report to the teacher.
- ➢ Any specific operations in the experiment shall be performed according to the experiment manual. If you have any questions, please promptly ask the teacher.
- ➢ During the experiment, the joints will start to work as the robotic arms are powered on. In that case, do not move the joints of the robotic arms hard if you do not press the unlock key.
- ➢ Report any device fault during the experiment to your teacher in a timely manner, and do not handle it yourself.
- ➢ Arrange all devices after the experiment. You shall not leave the lab before check by the group leader.

Task 1: Sow Three Times

The robotic arm automatically sows three times. For this, we can think that it repeats sowing three times. Of course, during programming, note that the sowing position will change each time.

1. Analysis

Analyze the steps to sow once, and think about any method of sowing three times. Then, fill in the blanks in the table below according to your analysis.

Step 1: Prepare Hardware.

Step 2: Open Arduino IDE and select the corresponding development board and port.

Step 3: Design the program: Move to the seed grabbing position $\rightarrow$ Open the suction cup to suck the seed $\rightarrow$

(Follow the prompts to complete the remaining steps)

Step 4: Implement the method for sowing three times:

2. Steps

(1) Prepare Hardware

Step 1: Place the equipment on the corresponding positions, as shown in Figure 1.2.

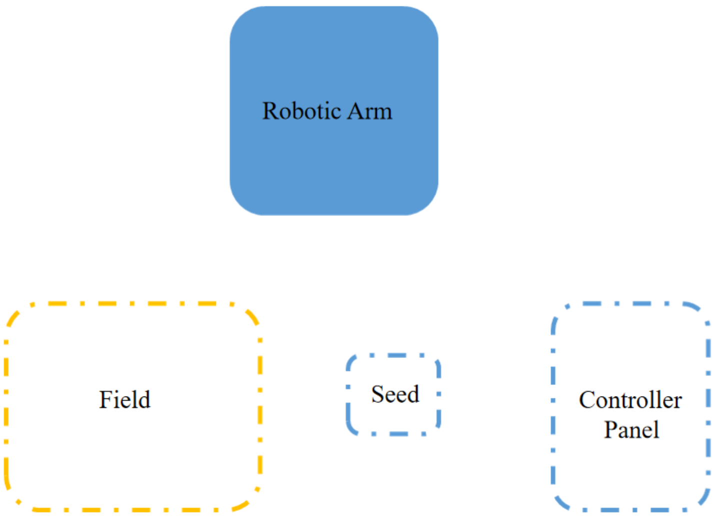


Figure 1.2 Placement positions

Step 2: Connect the components of the robotic arm sowing system, as shown in Figure 1.3.

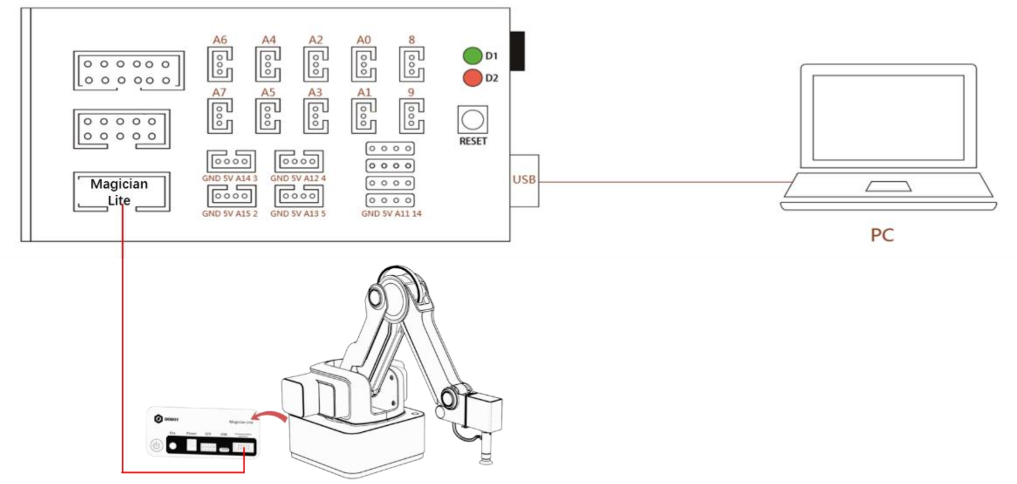


Figure 1.3 Connection diagram of the robotic arm sowing system

Step 3: Prepare the related equipment, and power on the robotic arm, as shown in Figure 1.4. Connect the Arduino Mega 2560 control board with Arduino shield expansion board to the computer, as shown in Figure 1.5. The connection with the robotic arm is shown in Figure 1.6.

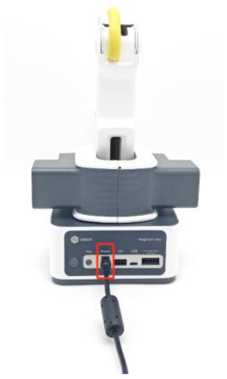


Figure 1.4 Connecting the robotic arm to a power supply

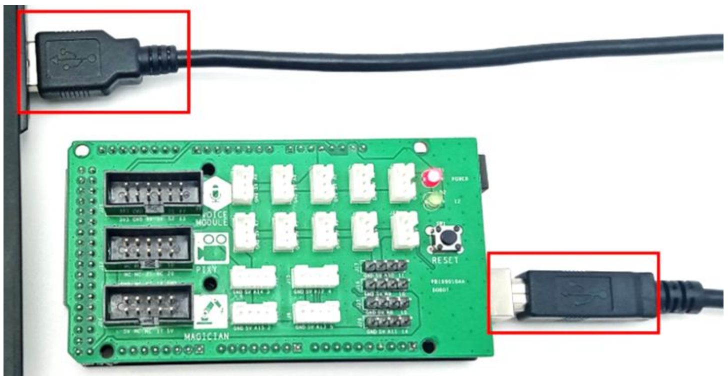


Figure 1.5 Connecting the control board to the PC

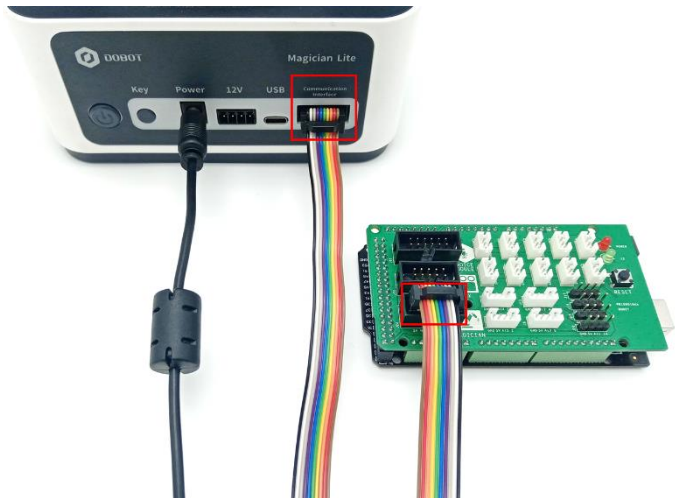


Figure 1.6 Connecting the control board and the robotic arm

(2) Design Program

Step 1: Check the port, connect the data cable, right-click "This PC", and click **Manage** in the pop-up window, as shown in Figure 1.7.

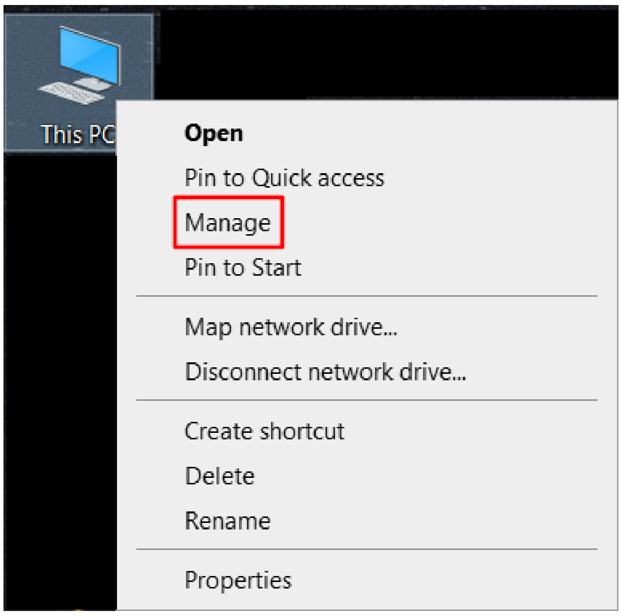


Figure 1.7 Clicking **Computer Management**

Step 2: Click **Manage**, and a window appears, as shown in Figure 1.8. Click **Device Manager** on the left and find **Ports (COM & LPT)**. Click the drop-down triangle to display the information. You can see that the COM port pops up in brackets.

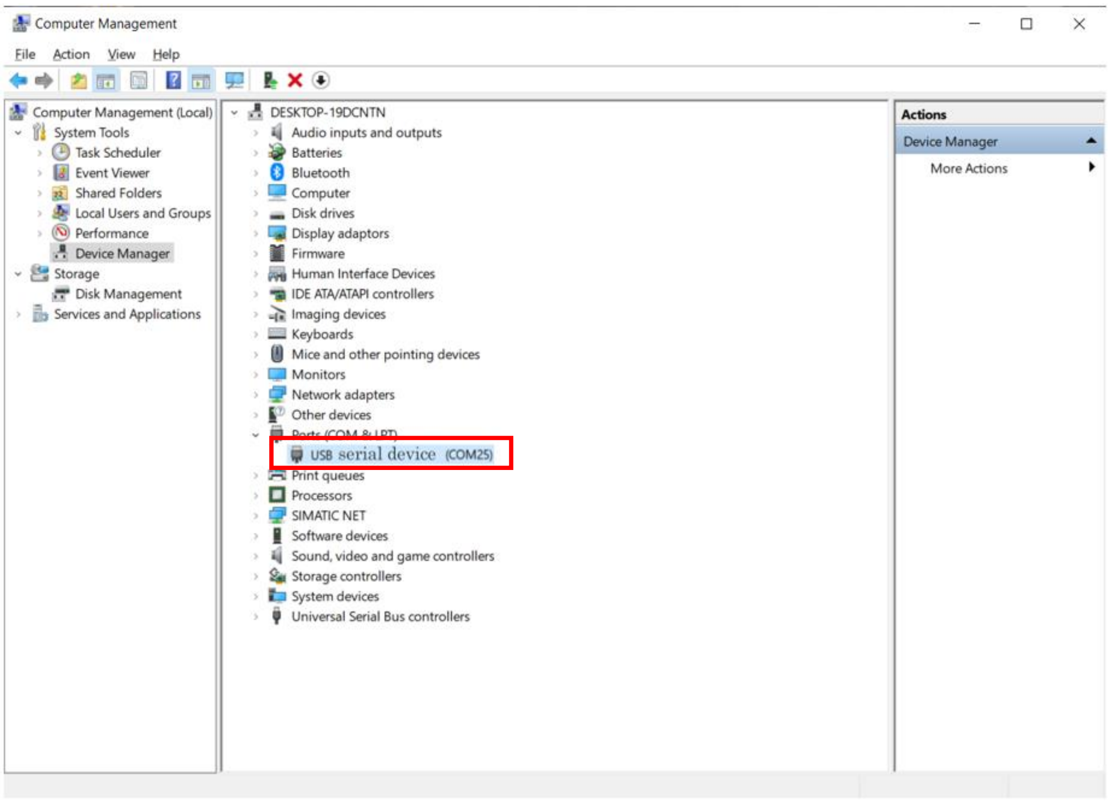
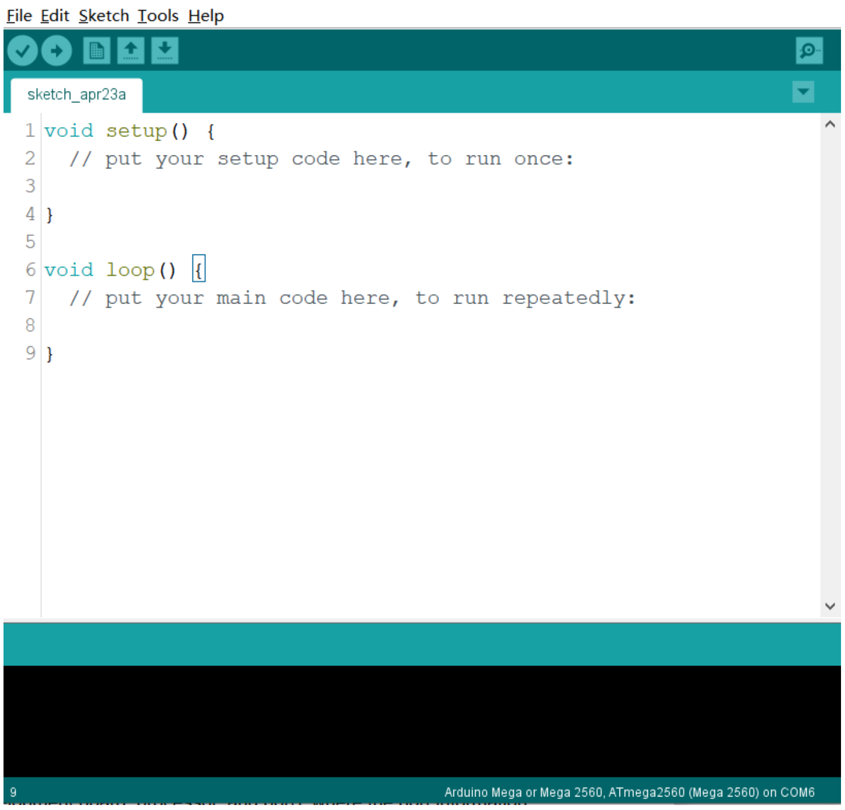


Figure 1.8 COM port

Step 3: Enable the Arduino IDE. The main interface of the software is shown in Figure 1.9.

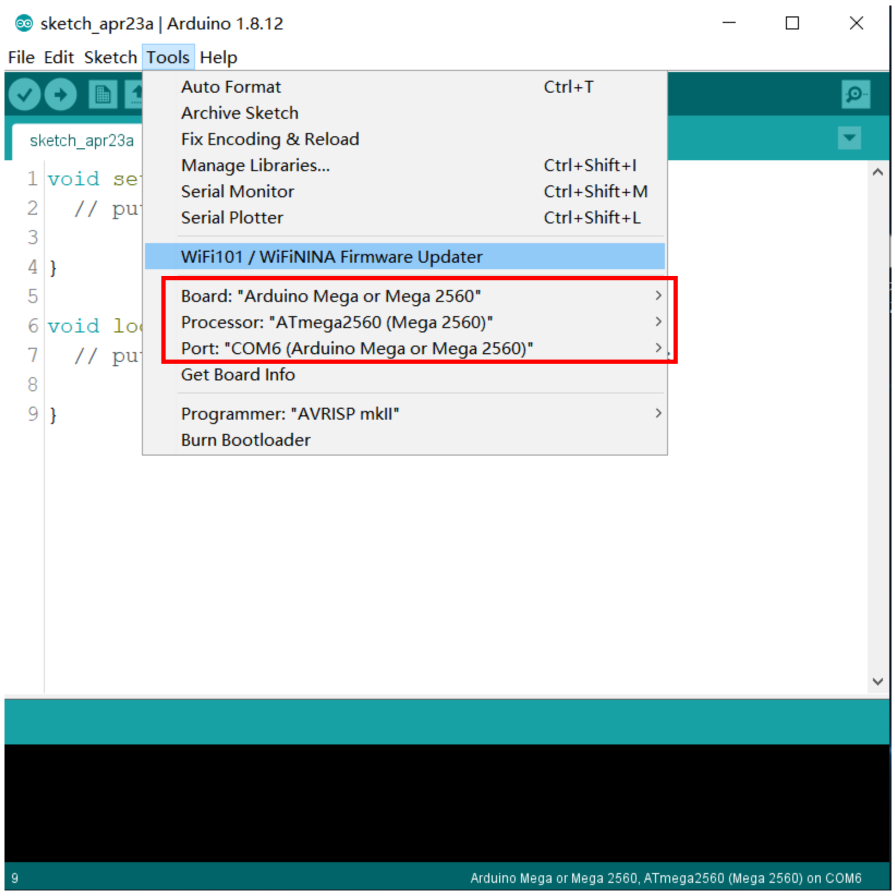


```
void setup() {
  // put your setup code here, to run once:

}

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

}
```

9 Arduino Mega or Mega 2560, ATmega2560 (Mega 2560) on COM6

Figure 1.9 Arduino IDE interface

Step 4: In the menu bar of the software interface, click **Tools**, and modify the information in the red box in Figure 1.10 (respectively involving the board, processor, and port). Here, the port information is the COM port value you have just queried in **Device Manager**.


sketch_apr23a | Arduino 1.8.12
File Edit Sketch Tools Help
sketch_apr23a
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
Programmer: "AVRISP mkII"
Burn Bootloader

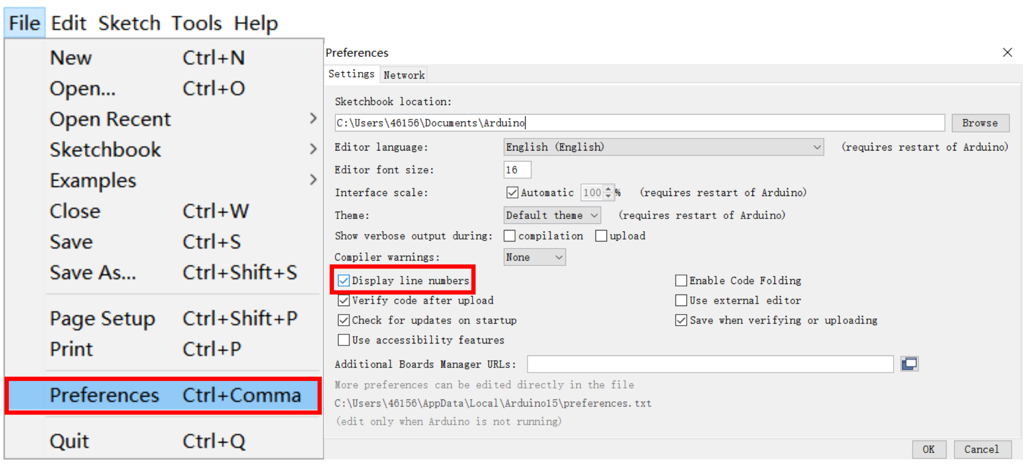
Board: "Arduino Mega or Mega 2560"
Processor: "ATmega2560 (Mega 2560)"
Port: "COM6 (Arduino Mega or Mega 2560)"

```
void se
  // pu
}

void lo
  // pu
}
```

Arduino Mega or Mega 2560, ATmega2560 (Mega 2560) on COM6

Figure 1.10 Software settings

Step 5: Click **File** in the menu bar, select **Preferences**, and tick **Display line numbers** in the preferences, as shown in Figure 1.11. Thus, as you write code, the left side of the software interface displays the code line number, as shown in Figure 1.12.



Preferences

Settings | Network

Sketchbook location:
C:\Users\46156\Documents\Arduino [Browse]

Editor language: English (English) (requires restart of Arduino)

Editor font size: 16

Interface scale: [✓] Automatic 100 % (requires restart of Arduino)

Theme: Default theme (requires restart of Arduino)

Show verbose output during: [ ] compilation [ ] upload

Compiler warnings: None

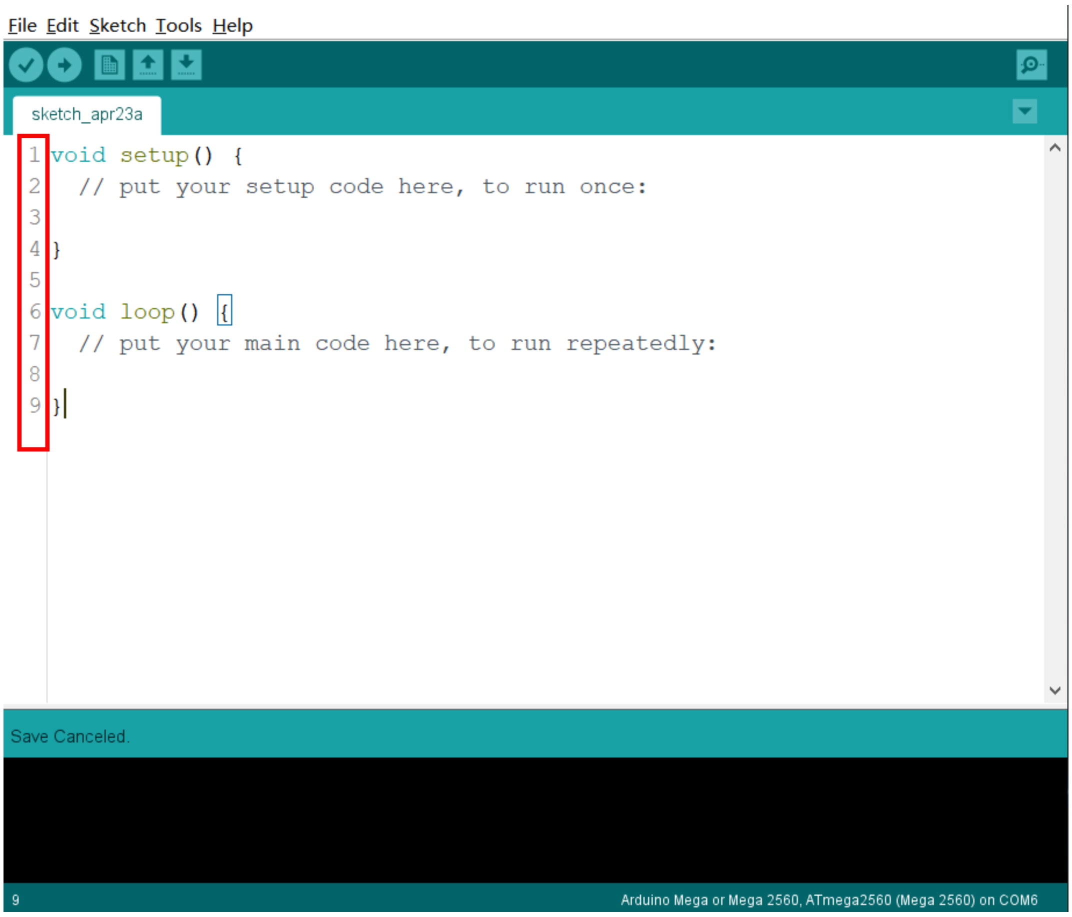
[✓] Display line numbers
[ ] Enable Code Folding
[✓] Verify code after upload
[ ] Use external editor
[✓] Check for updates on startup
[✓] Save when verifying or uploading
[ ] Use accessibility features

Additional Boards Manager URLs:

More preferences can be edited directly in the file
C:\Users\46156\AppData\Local\Arduino15\preferences.txt
(edit only when Arduino is not running)

[OK] [Cancel]

Figure 1.11 Displaying line numbers



```
1 void setup() {
2   // put your setup code here, to run once:
3 
4 }
5 
6 void loop() {
7   // put your main code here, to run repeatedly:
8 
9 }
```

Save Canceled.

9 Arduino Mega or Mega 2560, ATmega2560 (Mega 2560) on COM6

Figure 1.12 Displaying the code line number

Step 6: Analyze how the robotic arm automatically sows three times. Then, draw the program flow chart, as shown in Figure 1.13.

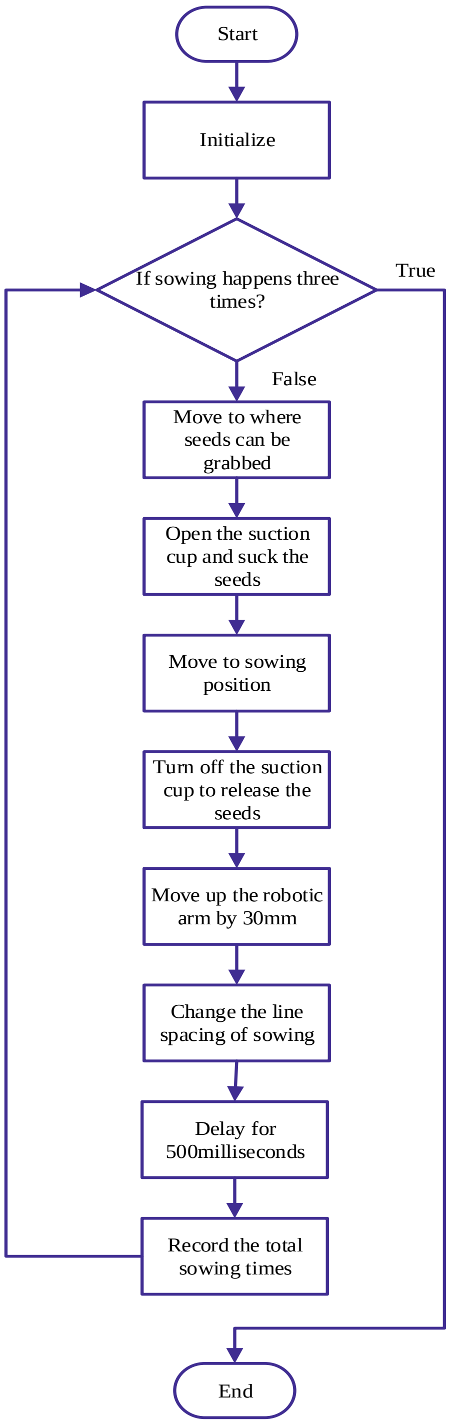

Start
Initialize
If sowing happens three times?
True
False
Move to where seeds can be grabbed
Open the suction cup and suck the seeds
Move to sowing position
Turn off the suction cup to release the seeds
Move up the robotic arm by 30mm
Change the line spacing of sowing
Delay for 500milliseconds
Record the total sowing times
End

Figure 1.13 Flow chart of sowing three times

Step 7: Set the header file. The header of the program is the header file. The header file as the carrier file contains the functions and data interface declarations, and plays the role of a toolbox. Therefore, the header file must have some statements. The header file of the robotic arm is: #include <MagicianLite.h>, and the programming method is as follows:

```
#include<MagicianLite.h>//header file
```

Step 8: Define a variable. Define the integer variable *num*. Its initial value is *0*. After it loops once, the value increases by 60. Here, 60 indicates the sowing spacing.

```
int num=0;
```

Step 9: Set the setup function.

1) Initialize the robotic arm. The corresponding program statement is:

```
voidsetup(){
MagicianLite_Init();              //Initialize the robotic arm
}
```

2) Set both movement speed percentage and acceleration percentage of the robotic arm to 80.

```
MagicianLite_SetPTPCommonParams(80,80);
   //Set movement speed and acceleration percentage of the robotic arm
```

3) Make the robotic arm move above the seed position.

```
voidsetup(){
MagicianLite_Init();              //Initialize the robotic arm
MagicianLite_SetPTPCommonParams(80,80);
   //Set movement speed and acceleration percentage of the robotic arm
   MagicianLite_SetPTPCmd(JUMP_XYZ, 275,80,-43+30,0);
   //Move above the seed grabbing position
}
```

Step 10: Set the main function of the loop.

1) "if" statement: If you control the robotic arm to sow three times automatically, the sowing spacing is 60 mm. Here, *num = 0* indicates the first sowing, *num = 60* the second sowing, *num = 120* the third sowing, and *num = 180* sowing stoppage.

```
voidloop(){
if(num<180)                       //Sowing three times
}
```

2) Sowing process: Move to the seed grabbing position → Open the suction cup → Move to the sowing position → Close the suction cup. The programming method is as follows:

```
MagicianLite_SetPTPCmd(JUMP_XYZ, 275,80,-43,0);//Move to seed grabbing
position
   MagicianLite_SetEndEffectorSuctionCup(true);      //Open the suction cup
   delay(200);
   MagicianLite_SetPTPCmd(JUMP_XYZ, 327-num,-39, -9, 0);//Sowing position
   MagicianLite_SetEndEffectorSuctionCup(false);    //Close the suction cup
   delay(200);
```

Step 11: Raise the robotic arm by 30 mm each time it finishes sowing.

```
   MagicianLite_SetPTPCmd(JUMP_XYZ, 327,-39,-9+30,0);
   //The robotic arm is lifted upward by 30 mm
```

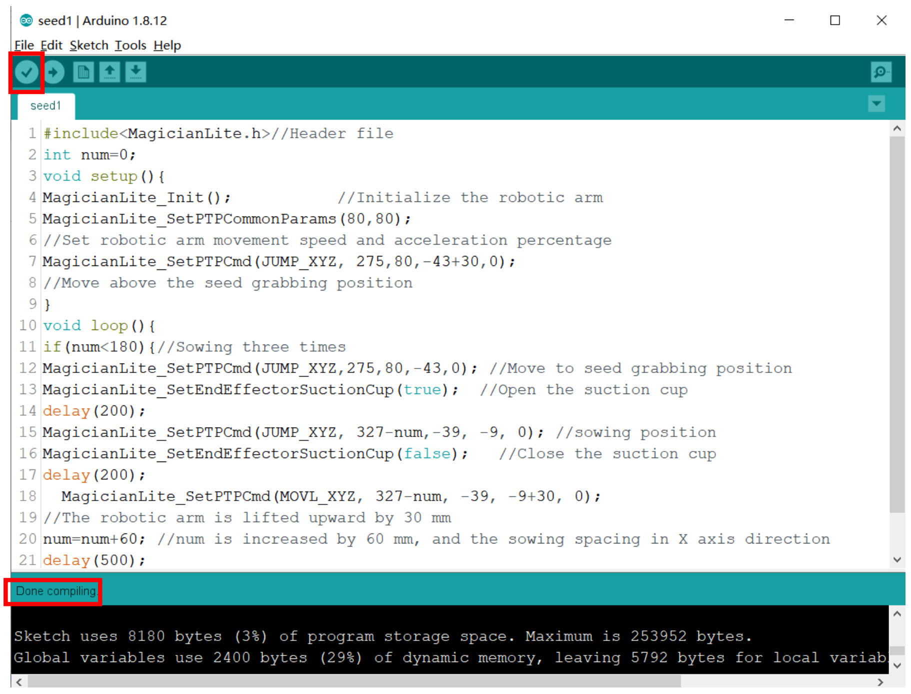
Step 12: Increase the variable *num* by 60.

```
   num=num+60;      //num increased by 60 mm, sowing spacing in X axis direction
   delay(500);
```

Step 13: According to the analysis of the above steps, integrate the program.

```
#include<MagicianLite.h>//Header file
int num=0;
Void setup(){
MagicianLite_Init();                    //Initialize the robotic arm
MagicianLite_SetPTPCommonParams(80,80);
   //Set robotic arm movement speed and acceleration percentage
   MagicianLite_SetPTPCmd(JUMP_XYZ, 275,80,-43+30,0);
   //Move above the seed grabbing position
}
Void loop(){
if(num<180){//Sowing three times
      MagicianLite_SetPTPCmd(JUMP_XYZ,275,80,-43,0); //Move to seed grabbing position
       MagicianLite_SetEndEffectorSuctionCup(true);    //Open the suction cup
   delay(200);
   MagicianLite_SetPTPCmd(JUMP_XYZ, 327-num,-39, -9, 0); //sowing position
   MagicianLite_SetEndEffectorSuctionCup(false);      //Close the suction cup
   delay(200);
      MagicianLite_SetPTPCmd(MOVL_XYZ, 327-num, -39, -9+30, 0);
      //The robotic arm is lifted upward by 30 mm
      num=num+60; //num is increased by 60 mm, and the sowing spacing in X axis direction
   delay(500);
   }
}
```

Step 14: Write the code, compile the program, click the “√” symbol in the upper left corner of the Arduino interface, and then check the compilation, as shown in Figure 1.14.



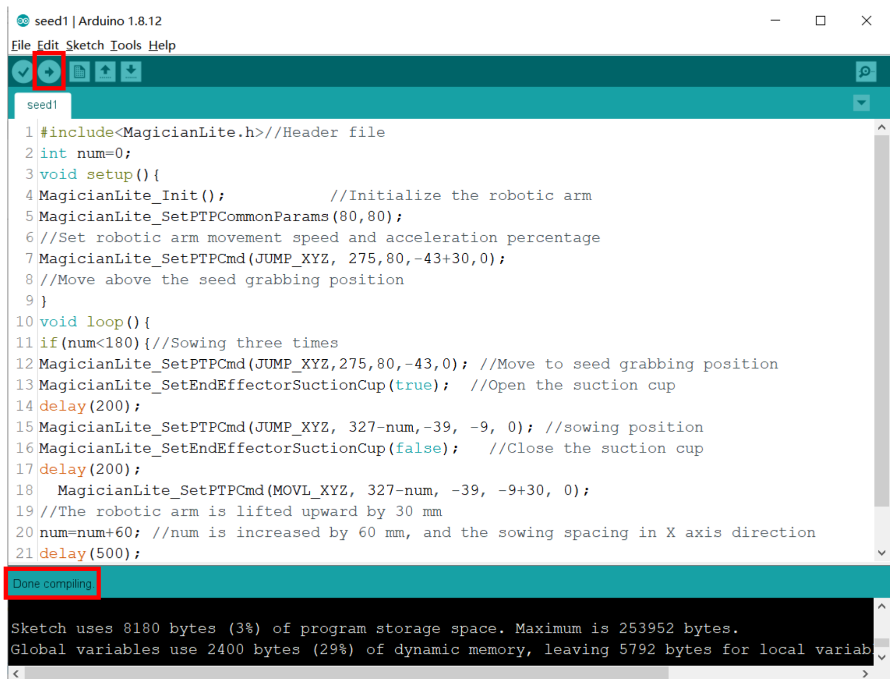
```
#include<MagicianLite.h>//Header file
int num=0;
void setup(){
MagicianLite_Init();            //Initialize the robotic arm
MagicianLite_SetPTPCommonParams(80,80);
//Set robotic arm movement speed and acceleration percentage
MagicianLite_SetPTPCmd(JUMP_XYZ, 275,80,-43+30,0);
//Move above the seed grabbing position
}
void loop(){
if(num<180){//Sowing three times
MagicianLite_SetPTPCmd(JUMP_XYZ,275,80,-43,0); //Move to seed grabbing position
MagicianLite_SetEndEffectorSuctionCup(true);  //Open the suction cup
delay(200);
MagicianLite_SetPTPCmd(JUMP_XYZ, 327-num,-39, -9, 0); //sowing position
MagicianLite_SetEndEffectorSuctionCup(false);   //Close the suction cup
delay(200);
  MagicianLite_SetPTPCmd(MOVL_XYZ, 327-num, -39, -9+30, 0);
//The robotic arm is lifted upward by 30 mm
num=num+60; //num is increased by 60 mm, and the sowing spacing in X axis direction
delay(500);
```

Done compiling

```
Sketch uses 8180 bytes (3%) of program storage space. Maximum is 253952 bytes.
Global variables use 2400 bytes (29%) of dynamic memory, leaving 5792 bytes for local variab
```

Figure 1.14 Compiling the program

Step 15: Upload the program to the main control board if there is no error in the code. Click the "→" symbol in the upper left corner of the Arduino interface, and then check the upload status. When the status bar displays "Uploaded successfully", your program is burned, as shown in Figure 1.15. From the board you can view that a small green LED flashes every 500 ms, as set by the program.



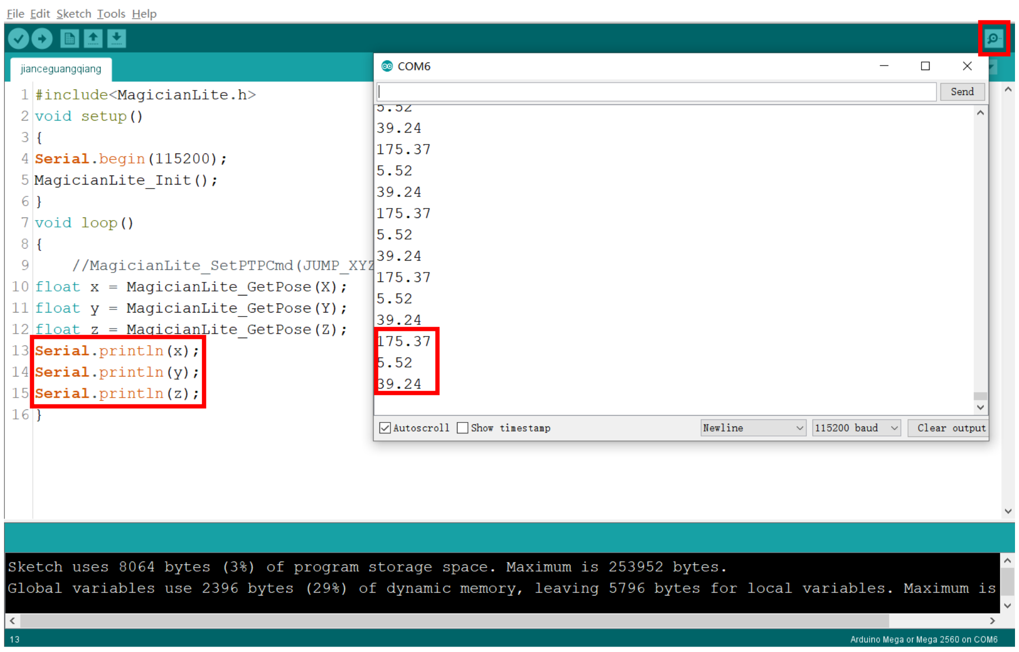
```
#include<MagicianLite.h>//Header file
int num=0;
void setup(){
MagicianLite_Init();            //Initialize the robotic arm
MagicianLite_SetPTPCommonParams(80,80);
//Set robotic arm movement speed and acceleration percentage
MagicianLite_SetPTPCmd(JUMP_XYZ, 275,80,-43+30,0);
//Move above the seed grabbing position
}
void loop(){
if(num<180){//Sowing three times
MagicianLite_SetPTPCmd(JUMP_XYZ,275,80,-43,0); //Move to seed grabbing position
MagicianLite_SetEndEffectorSuctionCup(true);  //Open the suction cup
delay(200);
MagicianLite_SetPTPCmd(JUMP_XYZ, 327-num,-39, -9, 0); //sowing position
MagicianLite_SetEndEffectorSuctionCup(false);   //Close the suction cup
delay(200);
  MagicianLite_SetPTPCmd(MOVL_XYZ, 327-num, -39, -9+30, 0);
//The robotic arm is lifted upward by 30 mm
num=num+60; //num is increased by 60 mm, and the sowing spacing in X axis direction
delay(500);
```

```
Sketch uses 8180 bytes (3%) of program storage space. Maximum is 253952 bytes.
Global variables use 2400 bytes (29%) of dynamic memory, leaving 5792 bytes for local variab
```

Figure 1.15 Uploading the program and viewing the upload result

Get-point method: Call the API "MagicianLite_GetPose ()", and the serial port monitor can return the real-time coordinate point of the robotic arm.

```
#include<MagicianLite.h>
Void setup()
{
Serial.begin(115200);
MagicianLite_Init();
}
Void loop()
{
    //MagicianLite_SetPTPCmd(JUMP_XYZ,200,0,30,0);
float x = MagicianLite_GetPose(X);
float y = MagicianLite_GetPose(Y);
float z = MagicianLite_GetPose(Z);
Serial.println(x);
Serial.println(y);
Serial.println(z);
}
```



```
#include<MagicianLite.h>
void setup()
{
Serial.begin(115200);
MagicianLite_Init();
}
void loop()
{
    //MagicianLite_SetPTPCmd(JUMP_XYZ
float x = MagicianLite_GetPose(X);
float y = MagicianLite_GetPose(Y);
float z = MagicianLite_GetPose(Z);
Serial.println(x);
Serial.println(y);
Serial.println(z);
}
```

```
5.52
39.24
175.37
5.52
39.24
175.37
5.52
39.24
175.37
5.52
39.24
175.37
5.52
39.24
```

```
Sketch uses 8064 bytes (3%) of program storage space. Maximum is 253952 bytes.
Global variables use 2396 bytes (29%) of dynamic memory, leaving 5796 bytes for local variables. Maximum is
```

13 Arduino Mega or Mega 2560 on COM6

Single-line comment: Single-line comments are usually used to explain a line of code in a program, which is indicated by a "//" symbol, followed by the content of the comment. The sample code is as follows:

```
int ledPin = 2; // the number of the LED pin
```

Multi-line comments: Multi-line comments, starting with the symbol "/*" and ending with the symbol "*/". The sample code is as follows:

```
/*Serial.print (" *C ");
Serial.print (hif);
Serial.println (" *F");*/
```

3. Summary

(1) How do we check the port number on the computer when the data cable is connected?____

____

(2) What actions shall the robotic arm perform during sowing? ______

____

(3) Which APIs can open and close the suction cup?_______________

____

4. Self-Assessment

Check the completed content in the experiment task. Tick（√） the completed items, and circle (◯) the uncompleted items.

Assessment Content	**Completion Status**
I've built the hardware	
I've checked the port	
I've completed the task of displaying the line number of the code	
I've set header files and defining variables	

Assessment Content	Completion Status
I've set up the setup function	
I've programmed the robotic arm to automatically sow three times	

Task 2: Sow Multiple Times

In the last task we completed sowing three times. The farmland in the experiment was sowed according to three rows and three columns. That is, we completed one row of sowing in Task 1. In this task, we must sow nine times according to three rows and three columns.

1. Analysis

Review the method of sowing three times and think about the steps to sow nine times. Then, fill in the blanks in the table below according to your analysis.

Step 1: Prepare Hardware.

Step 2: Open Arduino IDE and select the corresponding development board and port.

Step 3: Design the program: Move to the seed grabbing position. → Open the suction cup to suck the seeds → Move to the sowing position, → close the sucker and lower the seed, → change the spacing between the sowing rows, → repeat the sowing three times (implementation steps for three sowings)

Step 4: Think about the steps to achieve sowing nine times:

2. Steps

(1) Prepare Hardware

Step 1: Place the equipment in the corresponding positions. See Task 1 for the position diagram.

Step 2: See Task 1 for the connection diagram of the robotic arm sowing system.

Step 3: Connect the Arduino Mega 2560 control board with the Arduino shield expansion board, and power on the robotic arm. See Task 1 for the physical connection diagram.

(2) Design Program

Step 1: Analyze the steps to sow nine times, and draw the program flow chart, as shown in Figure 1.16.

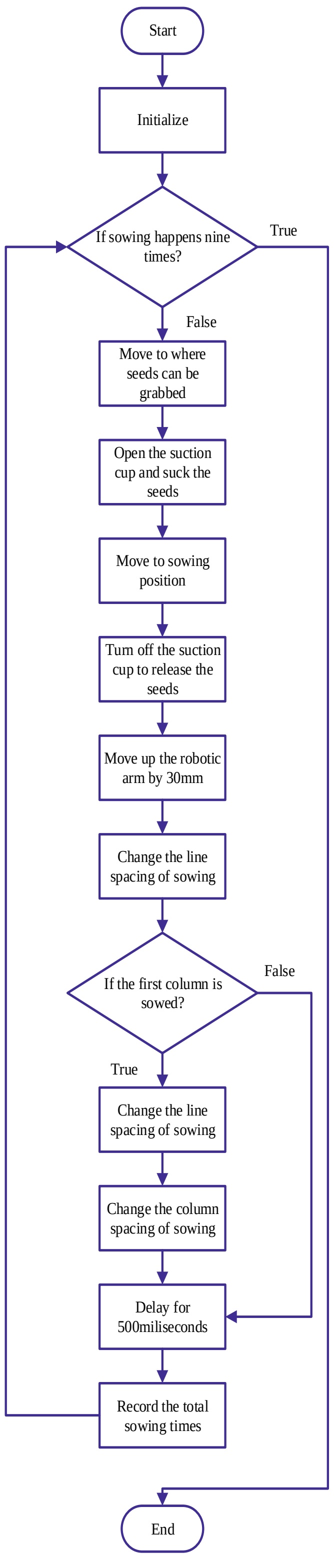

Start
Initialize
If sowing happens nine times?
True
False
Move to where seeds can be grabbed
Open the suction cup and suck the seeds
Move to sowing position
Turn off the suction cup to release the seeds
Move up the robotic arm by 30mm
Change the line spacing of sowing
If the first column is sowed?
False
True
Change the line spacing of sowing
Change the column spacing of sowing
Delay for 500miliseconds
Record the total sowing times
End

Figure 1.16 Flow chart of sowing nine times

Step 2: Set the header file. For this, see Task 1.

Step 3: Define variables. Define an integer variable *i*, and record the number of sowing times; define the integer variable *num* to control the sowing distance in the X axis direction; define the integer variable *j* to control the sowing distance in the Y axis direction.

```
inti=0;//Record the number of sowing times
int num=0;//Control the sowing spacing in X-axis
int j=0;//Control the sowing spacing in Y-axis
```

Step 4: Set up the setup function. For this, see Task 1. Write the code yourselves.

Step 5: Set the main function of the loop.

1) Use the “while” loop statement to sow nine times. Therefore, sowing must loop nine times.

```
voidloop(){
  While(i<9)
}
```

2) Write the code to sow.

```
      MagicianLite_SetPTPCmd(JUMP_XYZ, 275,80,-43,0); //Move to seed
grabbing position
MagicianLite_SetEndEffectorSuctionCup(true);   //Open the suction cup
delay(200);
MagicianLite_SetPTPCmd(JUMP_XYZ, 327-num, -39-j, -9, 0);//sowing position
MagicianLite_SetEndEffectorSuctionCup(false);     //Close the suction cup
      delay(200);
      MagicianLite_SetPTPCmd(MOVL_XYZ, 327-num,-39-j,-9+30, 0);
//The robotic arm is lifted upward by 30 mm
```

3) Set the variable. *Num* increases by 60 for each sowing, and each column has three sowing positions, so when *num* is *120*, the robotic arm sows three times; when *num* is *180*, *num* is set to *0*, j increases by 60, and the robotic arm will sow in the next column. The variable *i* records the total number of sowing times, which does not exceed 9

```
        num=num+60;//Increase num by 60, sowing spacing in X axis direction
        if(num==180){
    num=0;
    j=j+60;
        }
        delay(500);
        i=i+1;
    }
}
```

Step 6: According to the analysis of the above steps, integrate the program.

```
#include<MagicianLite.h>// Header file
inti=0;       //Control the robotic arm sow in X-axis
intnum=0;     //The variable num is used to control the sowing spacing
intj=0;     //Record the number of sowing times voidsetup()
{
MagicianLite_Init();                              //Initialize the robotic arm
MagicianLite_SetPTPCommonParams(80,80); //Set the robotic arm motion ratio
MagicianLite_SetPTPCmd(JUMP_XYZ,275,80,-43+30,0);
//Robotic arm moves above seed position
}
voidloop()
{
while(i<9)                                         //Sowing nine times
    {
MagicianLite_SetPTPCmd(JUMP_XYZ, 275, 80, -43, 0);
// Move to the seed grabbing position
MagicianLite_SetEndEffectorSuctionCup(true); //Open the suction cup
delay(200);
MagicianLite_SetPTPCmd(JUMP_XYZ, 327-num, -39-j, -9, 0);
// Sowing position
MagicianLite_SetEndEffectorSuctionCup(false);//Close the suction cup
delay(200);
MagicianLite_SetPTPCmd(MOVL_XYZ, 327-num,-39-j,-9+30, 0);
//The robotic arm is lifted upward by 30 mm
         num=num+60;      //Increase num by 60, sowing spacing in X axis direction
if(num==180)
         {
            num=0;
            j=j+60;
}
delay(500);
i=i+1;
      }
}

```

Step 7: Compile the program and upload the program to the Arduino control board.

Supposing that the farmland is sowed according to four rows and four columns, totalling 16 sowing positions, and the sowing spacing is still 60, how do you modify this program?

3. Summary

(1) Meanings of the three variables *i*, *j* and *num*: ________________
____

(2) Summarize the usage of the “while” loop statement:____________
____

(3) Summarize the usage of the judgment structure “if” statement: __
____

(4) API for point-to-point motion of the end of the robotic arm:______
____

4. Self-Assessment

Check the completed content in the experiment task. Tick（√） the completed items, and circle (◯) the uncompleted items.

Assessment Content	**Completion Status**
I’ve written the steps to sow nine times	
I’ve set header files and defined variables	
I’ve set the setup function	
I’ve allowed the robotic arm to automatically sow nine times	

Experiment 2
Automatic Supplementary Lighting

Team name: Team member: Date:

Overview

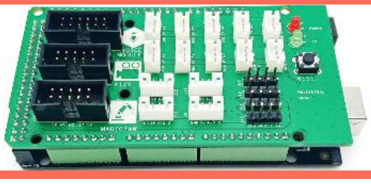
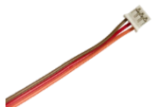
We know that plant growth depends on photosynthesis. In the farm, when light is insufficient, the chlorophyll formation of crops is hindered, and photosynthesis is affected, resulting in weak, yellowing, fallen leaves, and falling flowers. When light is too strong, it is easy to cause physiological damage to crops. For example, cucumbers, eggplants, tomatoes and so on are prone to leaf burns when light is too strong. This experiment allows crops to receive sufficient and appropriate light intensity by detecting and adjusting the light intensity of the farm. Next, let's make the LED running lights and breathing light to detect and adjust light intensity.

Objective

- Learn the control instructions of LED lights by controlling the LED to turn on and off.
- Explore the lighting sequence of LED running lights by making LED running lights.
- Explore the control method of LED breathing lights by making breathing lights.
- Understand how light sensors work by using light sensors to detect light intensity.
- Explore the relationship between light intensity and LED light brightness by using a light sensor to adjust light intensity.

- Learn the analog data and understand its range by adjusting light intensity through programming,

Equipment

Equipment Picture	Name	Quantity
	Arduino intelligent control board	1
	USB square port cable	1
	3-pin DuPont head adapter	Several
	Light sensor	1
	Red LED light	1
	Green LED light	1
	Blue LED light	1

Requirements

- Take care when using electricity.
- Before the experiment, check whether the experiment equipment is complete and intact. If there is any omission or damage, please report to the teacher.

- Any specific operations in the experiment shall be performed according to the experiment manual. If you have any questions, please promptly ask the teacher.
- During the experiment, the joints will start to work as the robotic arms are powered on. In that case, do not move the joints of the robotic arms hard if you do not press the unlock key.
- Report any device fault during the experiment to your teacher in a timely manner, and do not handle it yourself.
- Arrange all devices after the experiment. You shall not leave the lab before check by the group leader.

Task 1: Make the LED Running Light

With the development of science and technology, people are more and more aware of the importance of advertising. More and more colorful and innovative LED advertising films are flooding public places in cities, adding dazzling and bright colors. The dynamic display of these LED advertising billboards is based on the principle of running lights. Next, let's try making LED running lights.

1. Analysis

Try analyzing the sequence of turning on/off LED running lights.

(1) Turn on the red LED and turn off ______________________________ .

(2) ____ .

(3) ____ .

2. Steps

(1) Prepare Hardware

Step 1: Connect the LED running lights, as shown in Figure 2.1.

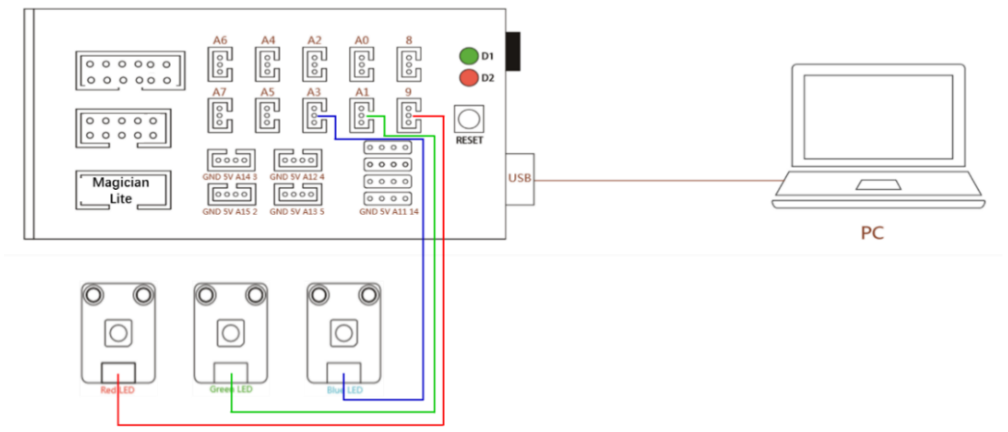


Figure 2.1 Equipment connection diagram

Step 2: Prepare the experiment equipment and connect it. The physical connection is shown in Figure 2.2.

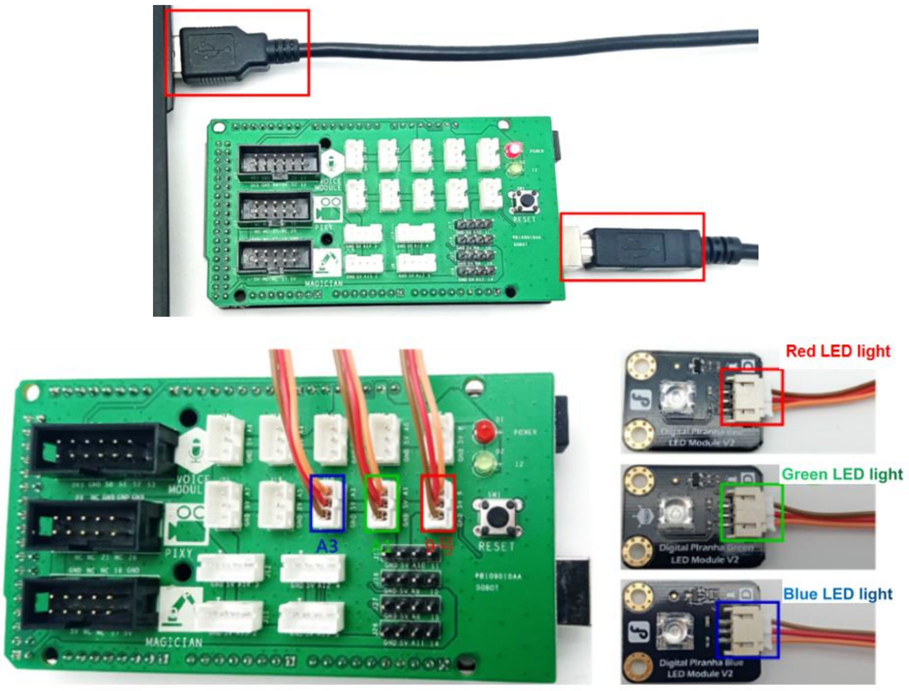


Figure 2.2 Physical connection diagram

(2) Design Program

Step 1: Analyze the implementation method of LED running lights, and read their flow chart, as shown in Figure 2.3.

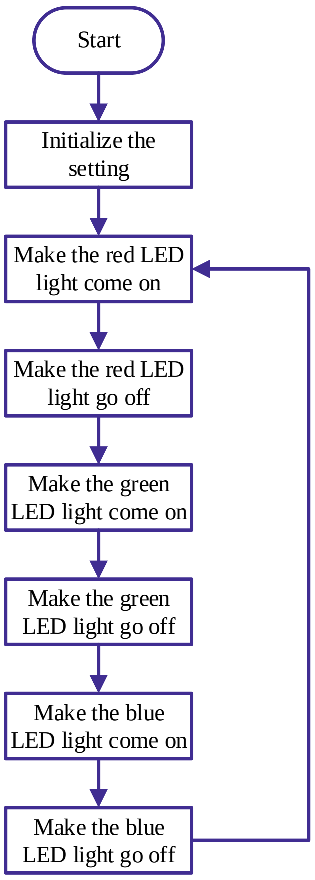

Start
Initialize the setting
Make the red LED light come on
Make the red LED light go off
Make the green LED light come on
Make the green LED light go off
Make the blue LED light come on
Make the blue LED light go off

Figure 2.3 Flow chart of LED running lights

Step 2: Perform Initial Setup.

1) Set the baud rate of the serial port to 115200, and allow signals to have the same frequency of sending and receiving. That is, ensure the consistent frequency between the computer and the Arduino control board. The programming method is as follows:

```
voidsetup()
{
Serial.begin(115200);   // Set serial baud rate
```

2) Set the input/output mode of the LED light port. Set each LED light port to output mode, that is, set each LED light port to OUTPUT mode. The programming method is as follows:

```
pinMode(Red_LED, OUTPUT);            //Set No.9 port to output mode
pinMode(Green_LED, OUTPUT);          // Set A1 port to output mode
pinMode(Blue_LED, OUTPUT);           //Set A3 port to output mode
```

(1) What is the difference between input and output modes?

(2) What are common input and output devices?

(3) Can you change the value of Red_LE to 9 in the programming, Green_LED to A1, and Blue_LED to A3? why?

3) Set the initial state of the LED light. There are two types of level modes: HIGH and LOW. When the LED light uses the high mode, it is on; when it uses the low mode, it is off. We set the initial state of the LED light to low mode, the programming method is as follows:

```
digitalWrite(Red_LED,LOW);          // Set No.9 port to low
digitalWrite(Green_LED,LOW);       // Set A1 port to low
digitalWrite(Blue_LED,LOW);        // Set 31 port to low
}
```

Step 3: Control the red LED to be on/off. In the loop function, program the red LED to turn it on, wait 500ms (0.5s), and then turn off it. The programming method is as follows:

```
void loop()
{
digitalWrite(Red_LED,HIGH);        //Turn on the red LED
delay(500);                        // Delay 0.5 seconds
digitalWrite(Red_LED,LOW);          //Turn off the red LED light
```

Step 4: Turn on/off the green LED light and blue LED light.

In the loop function, program the green and blue LED lights to turn them on/off.

Step 5: According to the analysis of the above steps, integrate the program of the LED running light. The programming method is as follows:

```
voidsetup()
{
    Serial.begin (115200); // Set serial baud rate
digitalWrite(Red_LED,LOW);          // Set No.9 port to low
digitalWrite(Green_LED,LOW);      // Set A1 port to low
digitalWrite(Blue_LED,LOW);       // Set 31 port to low
}
void loop()
{
digitalWrite(Red_LED,HIGH);       //Turn on the red LED
delay(500);                       // Delay 0.5 seconds
digitalWrite(Red_LED,LOW);        //Turn off the red LED light

  digitalWrite(Green_LED,HIGH);   //Turn on the green LED
   delay(500);                    //Delay 0.5 seconds
digitalWrite(Green_LED,LOW);      //Turn off the green LED light

digitalWrite(Blue_LED,HIGH);      //Turn on the blue LED
   delay(500);                    //Delay 0.5 seconds
digitalWrite(Blue_LED,LOW);       //Turn off the blue LED light
}
```

Step 9: Compile the program and upload it to Arduino main control board.

3. Summary

(1) Describe what you think of the setup function and the loop function:____

____

(2) Design a sequence of LED running lights yourself, and use Arduino programming:___________________________________

____

4. Self-Assessment

Check the completed content in the experiment task. Tick（√） the completed items, and circle (◯) the uncompleted items.

Assessment Content	Completion Status
I've used the control instructions for the LED lights	
I've controlled the red, green and blue LED lights to turn on and off through programming	

Assessment Content	Completion Status
I’ve programmed LED running lights	

Task 2: Make the Breathing Light

Common light shows, decorative lights, etc. can not only change to different colors, but also gradually change from light to dark, just as they can "breathe". "Breathing" is a physical activity necessary for us, and we know it very well. Then, how do we make a “breathing” light? Next, let's try doing it.

1. Analysis

Try analyzing the sequence in which the LED breathing lights become brighter and darker.

(1) Mode 1

When making an LED breathing light, first enable the LED light to change from dark to bright. When its brightness reaches ________,

(Fill in: the brightest and darkest), let it ______________.

(2) Mode 2

2. Steps

(1) Prepare Hardware

Step 1: Connect the LED breathing light, as shown in Figure 2.4.

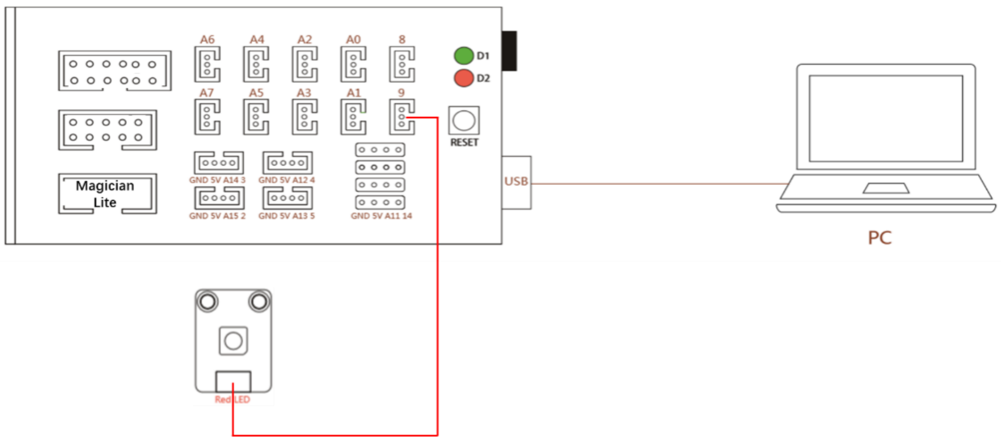


Figure 2.4 Device connection diagram

Step 2: Prepare the experiment equipment and connect it, as shown in Figure 2.5.

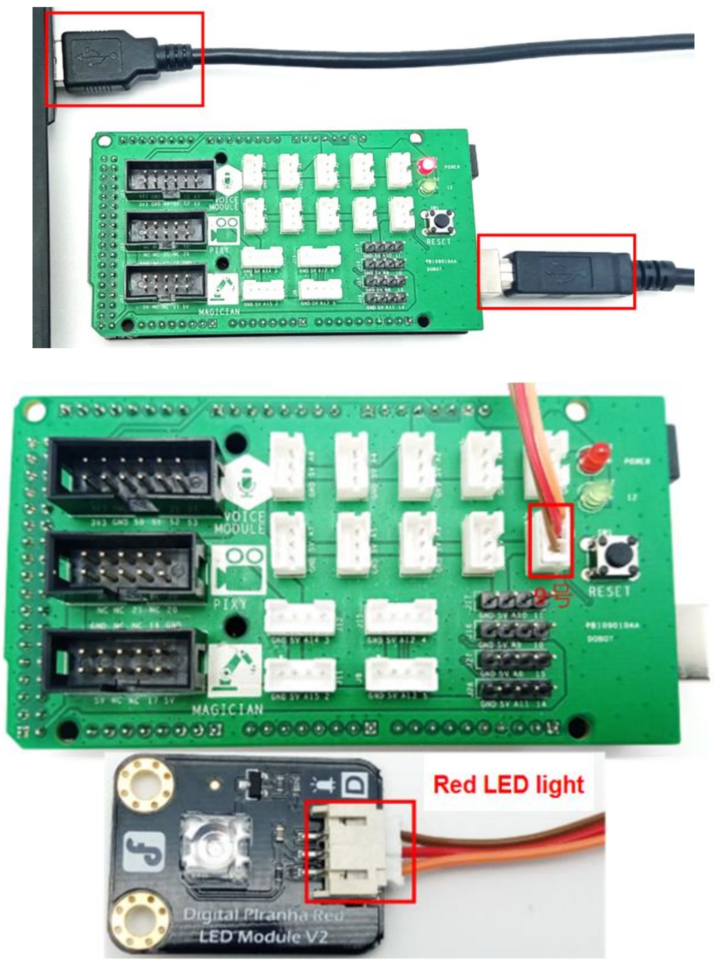


Figure 2.5 Physical connection diagram

(2) Design Program

Step 1: Analyze the method for making the breathing light, and draw a flow chart, as shown in Figure 2.6.

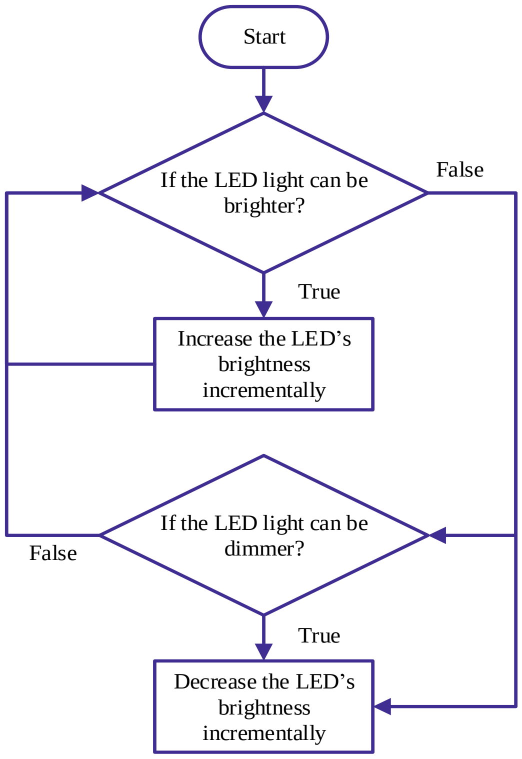

Start
If the LED light can be brighter?
False
True
Increase the LED's brightness incrementally
If the LED light can be dimmer?
False
True
Decrease the LED's brightness incrementally

Figure 2.6 Flow chart of making breathing lights

Step 2: Use the "for" loop statement because the breathing light is to gradually change from bright to dark and vice versa.

In the loop function, set variables, judgment conditions, and variable transformation formulas for the "for" loop statement.
The general form of the "for" statement is:

for (Expression 1; Expression 2; Expression 3)
Statement;

Example: "for" statement pseudocode.

```
for(i=1; i<4; i++)
{
    Move to the first point;
    Move to the second point;
}
```

Define an integer variable *i*, and set its initial value to *0* (that is, i = 0); the judgment condition of the loop statement is: i <= 255;

When the judgment condition is satisfied, the variable *i* increases by 1 from the original value (that is, i = i + 1); when the judgment condition is not satisfied, skip directly to the next loop statement. The programming method is as follows:

```
voidsetup()
{
// The setup part is not processed
}

void loop()
{
  for (int i = 0; i <= 255; i + = 1) {// From dark to bright, increment by 1 each time
```

Step 3: Write the analog value of the LED. When *i* gradually increases, so does the brightness of the LED light. Read the change of the variable *i* through the analogWrite () function. In that case, the brightness of the LED light changes, too. The programming method is as follows:

```
analogWrite(LED, i);                    // Write the analog value of LED
delay(300);                             //Wait 300ms to observe the gradient effect
  }
```

The analogWrite() function is the analog value (pulse signal) of the analog written pin. The syntax is analogWrite(pin,value);
Pin corresponds to the analog output pin, and the value indicates the duty cycle of the output pulse signal, ranging from 0 to 255.

Step 4: Skip to the next for loop statement when the judgment condition i <= 255 is not satisfied.

1) Define the integer variable *i*, and set its initial value to 255 (that is, i = 255);

2) The judgment condition of the loop statement is: i> = 0; when the judgment condition is satisfied, the variable *i* decreases by 1 from the original value (that is, *i* = *i*-1); when the judgment condition is not satisfied, the variable *i* returns to the first “for” loop statement again. The programming method is as follows:

```
    for (int i = 255; i> = 0; i-= 1) {// From bright to dark, it gradually decreases by 1
each time
```

Step 5: Write the analog value of the LED. When *i* decreases gradually, the brightness of the LED light will gradually become darker. Read the change of the variable *i* through the analogWrite () function. In that case, the brightness of the LED light changes, too. The programming method is as follows:

```
        analogWrite(LED, i);                    // Write the analog value of LED
        delay (300); // Wait 300ms to observe the gradual change effect
    }
}
```

Step 7: Compile the program and upload it to the Arduino control board.

3. Summary

(1) Write the general form of the "for" loop statement: ______________

(2) Within the parentheses of the "for" loop statement, the meaning of each expression is: ____

____

(3) Make the breathing light brighten and darken according to the following sequence: ____

4. Self-Assessment

Check the completed content in the experiment task. Tick（√） the completed items, and circle (◯) the uncompleted items.

Assessment Content	**Completion Status**
I've designed the sequence of making a LED light brighter and darker	
I've known the programming method of "for" loop statement	
I've learned to write the function of the LED analog value	
I've programmed to make a breathing light	

Task 3: Detect Light Intensity

We know that light intensity plays an important role in plant growth and morphology. Plants have different requirements for light intensity. Next, let's try detecting light intensity.

1. Analysis

We analyze the key steps to detect light intensity.

(1) To detect light intensity, use the equipment: ________________ . (Fill in the sensor name)

(2) Write the Arduino program, upload it to the Arduino intelligent control board.

(3) View the light intensity data in __________________.

2. Steps

(1) Prepare Hardware

Step 1: Connect the equipment to detect the light intensity, as shown in Figure 2.7.

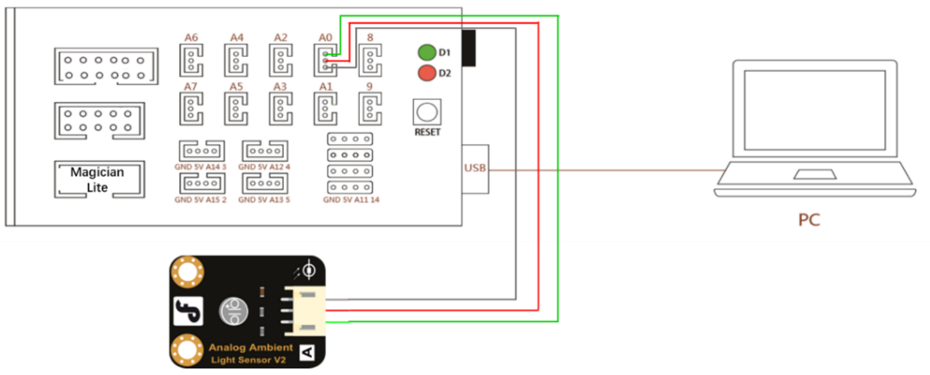


Figure 2.7 Equipment connection diagram

Step 2: Prepare the experiment equipment, and connect it, as shown in Figure 2.8.

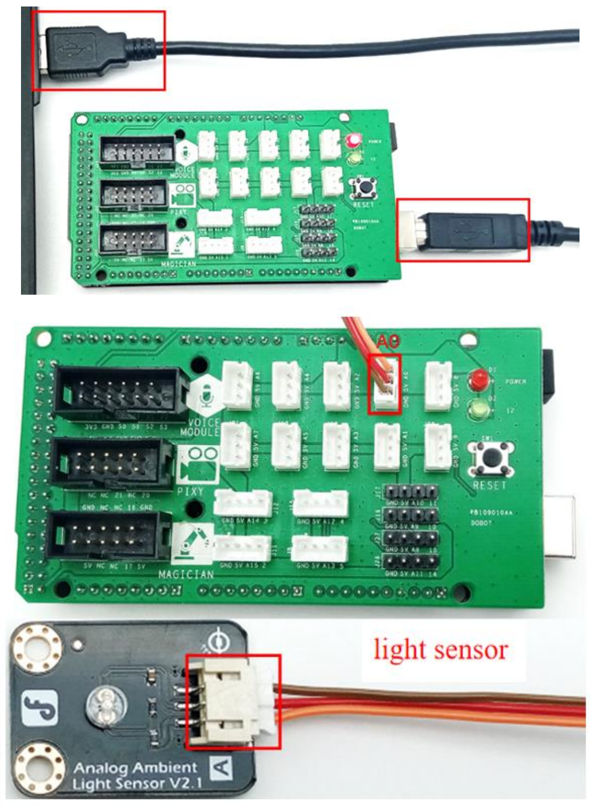


Figure 2.8 Physical connection diagram

(2) Design Program

Step 1: Analyze the method of detecting light intensity, and read the flow chart, as shown in Figure 2.9.

$$\begin{tikzcd}
\text{Start} \arrow[d] \\
\text{Initialize the setting} \arrow[d] \\
\text{Read the value of the light sensor} \arrow[d] \\
\text{Print the brightness data} \arrow[u, bend right]
\end{tikzcd}$$

Figure 2.9 Flow chart of detecting light intensity

Step 2: Perform initial setup.

1) Set the serial port baud rate to 115200.

In Arduino, try writing a program to set the serial port baud rate.

2) Set the input and output modes of the LED light port. The programming method is as follows:

```
Void setup() {
   Serial.begin (115200); // Set serial baud rate
pinMode (Lightsensor, INPUT); // Set the connection port A0 of the light sensor to the input mode
}
```

Step 3: In the loop function, define the integer variable *Light* and use it when reading the value of the light sensor in real time in the surface program. The programming method is as follows:

```
Void loop() {
   int Light; // Define the integer variable Light
```

Step 4: Read the value of the light sensor. Assign the variable *Light* to the current value of the light sensor. The programming method is as follows:

```
   Light = analogRead (Lightsensor); // Read the current value of the light sensor
```

The analogRead () function indicates the data value read from the specified analog pin, and the corresponding syntax is: analogRead (pin);

Step 5: Print the light intensity data. The light can read the value returned by the light sensor in real time, so you just need to print the value of the variable *Light*. The programming method is as follows:

```
Serial.println(Light);                    //Print light intensity data
}
```

Read and output the analog value of the pin, and send the data to a computer or a serial port receiving device. The serial port prints the syntax:

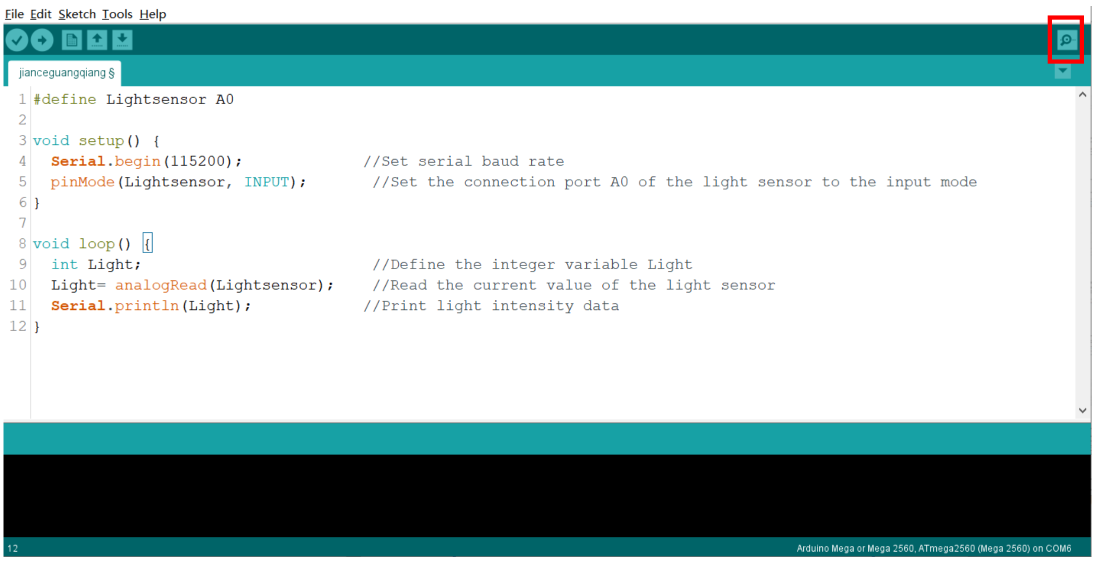
(1) Serial.print(val);   (2)Serial.println(val);

The differences between the two functions above are as follows:

Function	Serial.print(val)	Serial.println(val);
Serial display Data results	308307308309	308 307 308 309

Step 6: Compile the program and upload it to the Arduino control board.

Step 7: Open the serial port monitor, and view the light intensity data. In the upper right corner of Arduino main interface, click the **Search** symbol, and then enter the serial monitor interface, as shown in Figure 2.10.

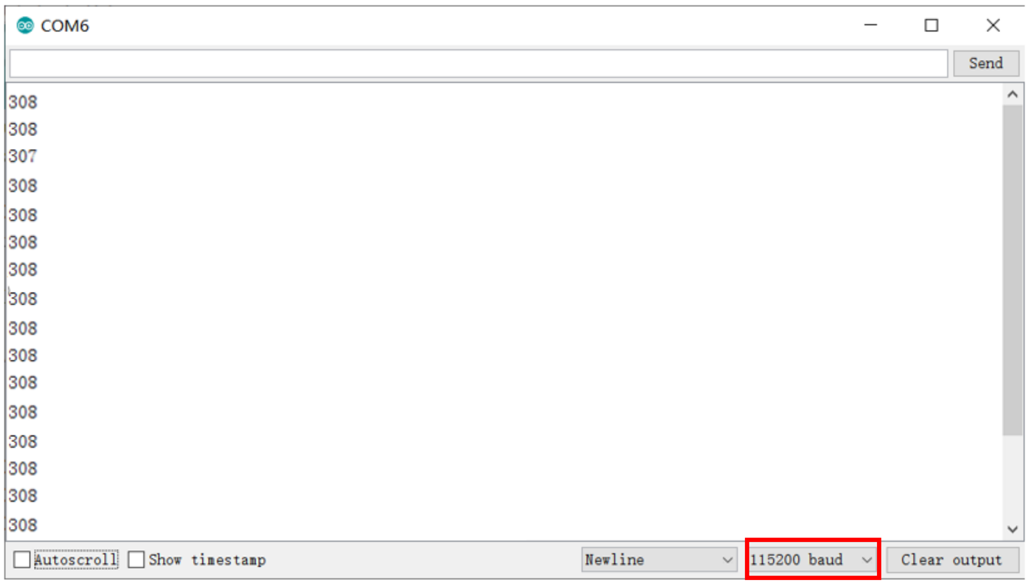

File Edit Sketch Tools Help
jianceguangqiang §

```
#define Lightsensor A0

void setup() {
  Serial.begin(115200);              //Set serial baud rate
  pinMode(Lightsensor, INPUT);        //Set the connection port A0 of the light sensor to the input mode
}

void loop() {
  int Light;                          //Define the integer variable Light
  Light= analogRead(Lightsensor);     //Read the current value of the light sensor
  Serial.println(Light);             //Print light intensity data
}
```

12
Arduino Mega or Mega 2560, ATmega2560 (Mega 2560) on COM6

Figure 2.10 Opening the serial monitor

Step 8: View the real-time data of the light intensity. See Figures 2.11 and 2.12 for the sensor data of strong light and no light respectively.

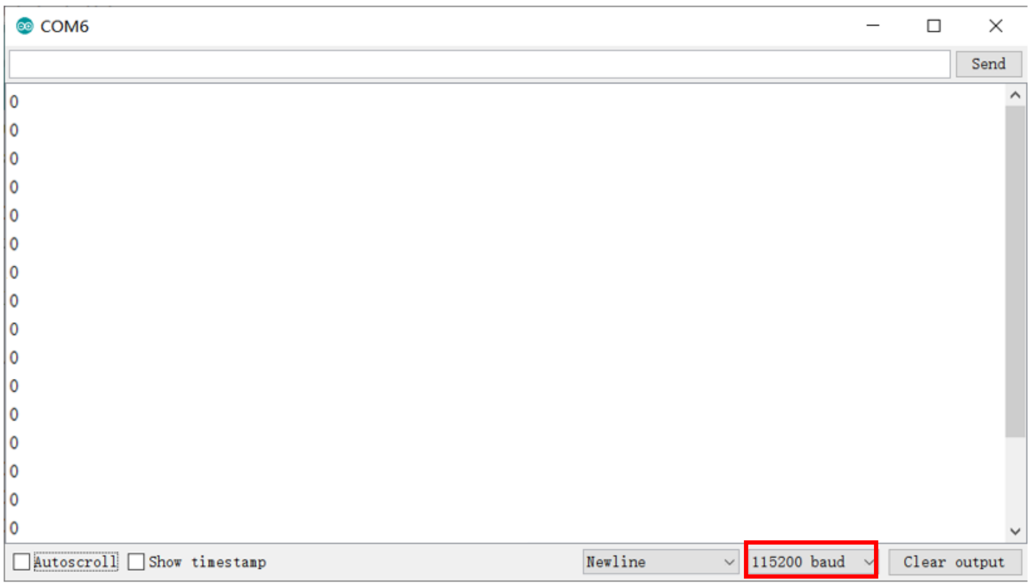

COM6

Send

```
308
308
307
308
308
308
308
308
308
308
308
308
308
308
308
308
```

Autoscroll Show timestamp Newline 115200 baud Clear output

Figure 2.11 Sensor values under strong light


COM6

Send

```
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
```

Autoscroll Show timestamp Newline 115200 baud Clear output

Figure 2.12 Sensor values under zero light intensity value

3. Summary

(1) The similarities between the two serial print functions: __________
____

(2) The similarities between the two serial print functions: __________
____

4. Self-Assessment

Check the completed content in the experiment task. Tick（√） the completed items, and circle (◯) the uncompleted items.

Assessment Content	Completion Status
I’ve learned the function of data read from analog pins	
I’ve learned the instructions for serial printing	
I’ve known how to view light intensity data from a serial monitor	
I’ve programmed light intensity detection	

Task 4: Adjust Light Intensity

We are now able to detect and view light intensity data. Next, let’s try adjusting the brightness of a light. Under different light intensity conditions, adjust the light intensity to make plants get more suitable light.

1. Analysis

We analyze the key steps to adjust the light intensity of a light.

2. Steps

(1) Prepare Hardware

Step 1: Connect the equipment to adjust light intensity, as shown in Figure 2.13.

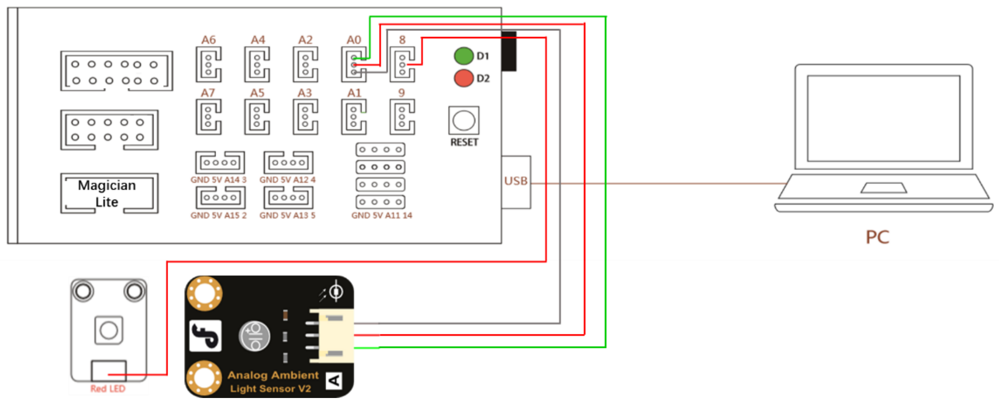


Figure 2.13 Equipment connection diagram

Step 2: Prepare the experiment equipment and connect it, as shown in Figure 2.14.

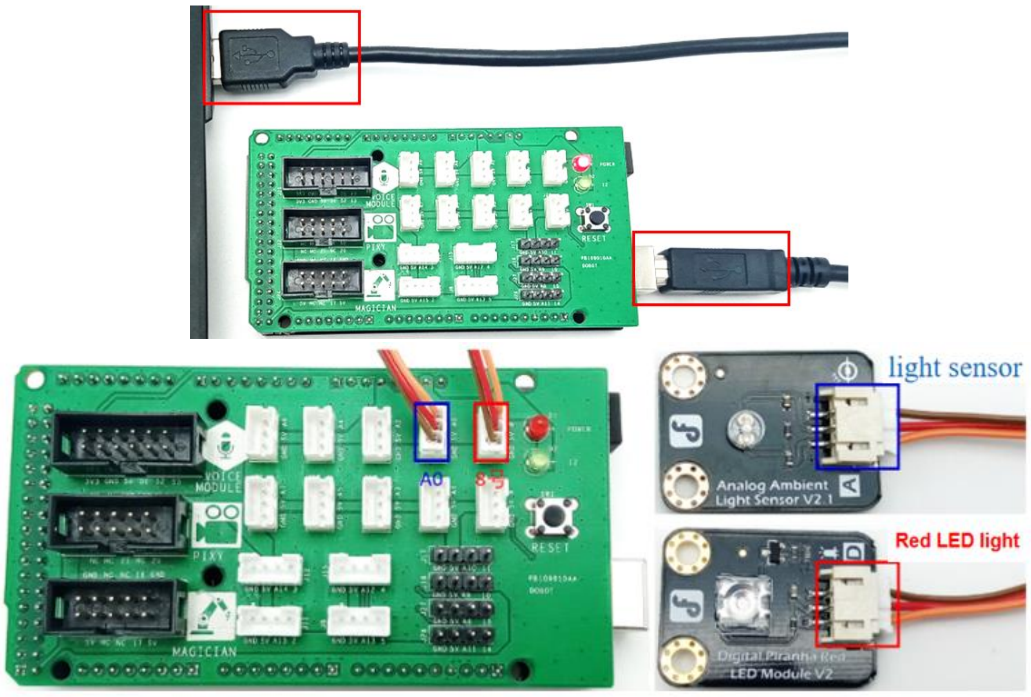

light sensor
A0
8 9
Analog Ambient Light Sensor V2.1
Red LED light
Digital Piranha Red LED Module V2

Figure 2.14 Physical connection diagram

(2) Design Program

Step 1: Analyze the method of adjusting light intensity, and read the flow chart, as shown in Figure 2.15.

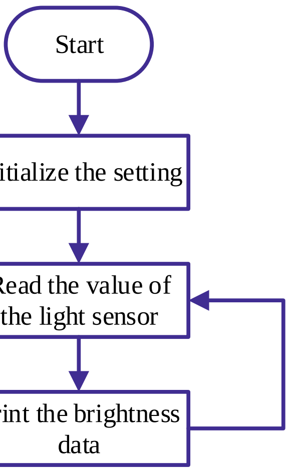


Figure 2.15 Flow chart of adjusting light intensity

Step 2: Define two global variables *MaxLight* and *MinLight* to record the maximum and minimum values of the light sensor output. Generally, set the maximum light intensity to 1023, and the minimum value to 0. The programming method is as follows:

```
int MaxLight = 1023;          //Output value of the light sensor module when it is fully exposed under strong light
int MinLight = 0;             //Output value of the light sensor module when fully shielded
```

During the programming, variables fall into global variables and local variables; global variables can be referenced by all objects or functions in the program. Local variables can only be referenced by some objects or functions, and cannot be referenced by other objects or functions.

The light sensor can automatically detect light intensity in the environment. The value read by the light sensor varies with the light intensity of the environment. Generally, the output value is recorded as 1023 and 0 respectively when the light sensor module is completely exposed under strong light (the actual value is smaller) and shielded (the actual value is slightly greater).

Step 3: Set the connection port mode.

Review: Programming modes for setting the connection port mode.

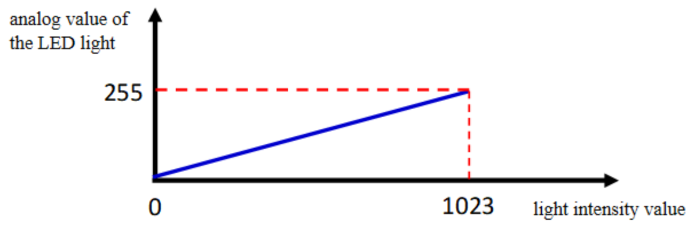
Try writing a connection port program. Among them, the connection port 8 of the LED light is in the output mode, and the connection port A0 in the input mode.

Step 4: In the loop function, define the local variable *Light* and assign a value to it, and thus read the current value of the light sensor. The programming method is as follows:

```
voidsetup() {
pinMode(LED, OUTPUT);                //Set LED connection port 8 to output mode
pinMode (Lightsensor, INPUT); // Set the connection port A0 of the light sensor to the input mode
}

voidloop() {
int Light;
  Light = analogRead(Lightsensor);        //Read the current value of the light sensor
```

Step 5: Define a local variable *i* and assign a value to it. The value of *i* ranges from 0 to 255. The value read by the light sensor ranges from 0 to 1023, then how does the value correspond to the analog value of the LED (that is, the value of *i*)? For this, we use the map () function.

map () function syntax: map (x, in_min, in_max, out_min, out_max)
Wherein x: value to be mapped
in_min: minimum interval before being mapped   in_max: maximum interval before being mapped
out_min: minimum interval after being mapped out_max: maximum interval after being mapped
The mapping relationship between the light intensity value and the analog value of the LED light is shown in the following figure:

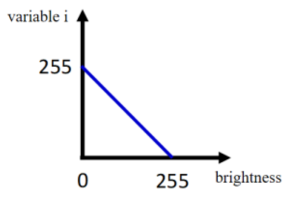


Define the integer variable i, and write the proportional mapping function. The programming method is as follows:

```
int i;
   i = map(Light, MinLight, MaxLight, 0, 255);   //Map the value of the light sensor
to 0-255
```

Step 6: Output Brightness.

When the light sensor finds light intensity relatively weak (i.e., the smaller the variable *Light* is, the smaller the variable *i* is), we need to increase the brightness of the LED light; when the light sensor finds light intensity strong (i.e., the greater the variable *Light* is, the greater the variable *i* is), we need to decrease the brightness of the LED light. The relationship between the variable *i* and the LED brightness is shown in the following figure:



The output brightness is programmed as follows:

```
analogWrite(LED,255-i);             //Output brightness, where the less light the
light sensor gets, the brighter the LED
delay(100);                          // Delay100ms
}
```

Step 7: Compile the program and upload it to the Arduino control board, and observe the changes in the light intensity and brightness of the LED light.

3. Summary

(1) Understand local and global variables: ______________________

____

(2) Relationship between ambient light intensity and LED brightness to be adjusted: ______________________________

____

____

4. Self-Assessment

Check the completed content in the experiment task. Tick（√） the completed items, and circle (◯) the uncompleted items.

Assessment Content	Completion Status
I’ve used a light sensor to detect light intensity	
I’ve known the mapping relationship between the light intensity value and the analog value of the LED light	
I’ve known the relationship between ambient light intensity and LED light brightness to be adjusted	
I’ve programmed the adjustment of light intensity	

Experiment 3

Smart Temperature Control

Team name: Team member: Date:

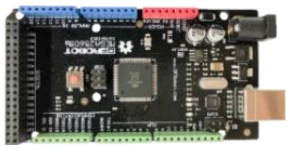
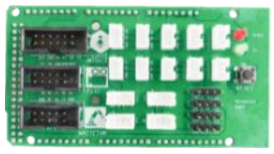
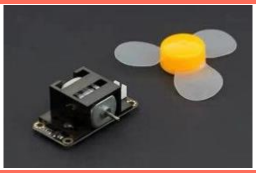
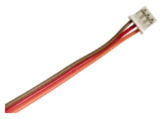
Overview

In hot summers, cold winters, humid springs, and dry autumns, temperature and humidity change too much, which hinders plant growth. In the smart farm, we have a temperature control system that can accurately detect the temperature, warm and cool crops, and ensure their better growth. Let's learn about the smart farm temperature control system.

Objective

1. Master the difference between integer variables and float variables by the instruction to define variables.
2. Master the control method of the temperature and humidity sensor by the relevant instructions of the temperature and humidity sensor.
3. Master the control method of the LCD display by the relevant instructions of the LCD display.
4. Master the control method of DC Motor by DC Motor related instructions.
5. Master the control method of the RGB module by the instructions of the RGB module.
6. Understand the multi-branch structure by programming the temperature control system.
7. Use the temperature and humidity sensor, the fan module, and the RGB light to simulate and build the temperature control system for a smart farm.

Equipment

Equipment Picture	Name	Quantity
	Arduino Mega2560 control board	1
	Arduino shield expansion board	1
	Digital temperature and humidity sensor (including data cable)	1
	Digital RGB full color LED module	1
	I^{2}C LCD character LCD blue screen	1
	DC motor fan	1
	USB square port cable	1
	3-pin DuPont head adapter	1
	Dupont line	Several

Requirements

- Take care when using electricity.
- Before the experiment, check whether the experiment equipment is complete and intact. If there is any omission or damage, please report to the teacher.
- Any specific operations in the experiment shall be performed according to the experiment manual. If you have any questions, please promptly ask the teacher.
- During the experiment, the joints will start to work as the robotic arms are powered on. In that case, do not move the joints of the robotic arms hard if you do not press the unlock key.

Task 1: Read the Values of the Temperature and Humidity Sensor

To adjust the temperature and humidity, understand the specific data of temperature and humidity. You can use the temperature and humidity sensor to detect the temperature and humidity of the environment, and display the temperature and humidity data on the serial monitor.

1. Analysis

Request the teacher to demonstrate how to read values of a temperature and humidity sensor. Observe the data printed on the serial port, analyze the process of reading the temperature and humidity sensor values, and fill in the blanks in the table below.

Step 1: Connect the equipment, control board, temperature and humidity sensor, and PC.

Step 2: Define the sensor ________.

Step 3: Create a variable,___________.

Step 4: Read the values of the temperature and humidity sensor.

Step 5: Use the function to display temperature and humidity data on the serial monitor.

2. Steps

(1) Prepare Hardware

Step 1: Connect the Arduino control board, DHT11 temperature and humidity sensor, and PC, as shown in Figure 3.1.

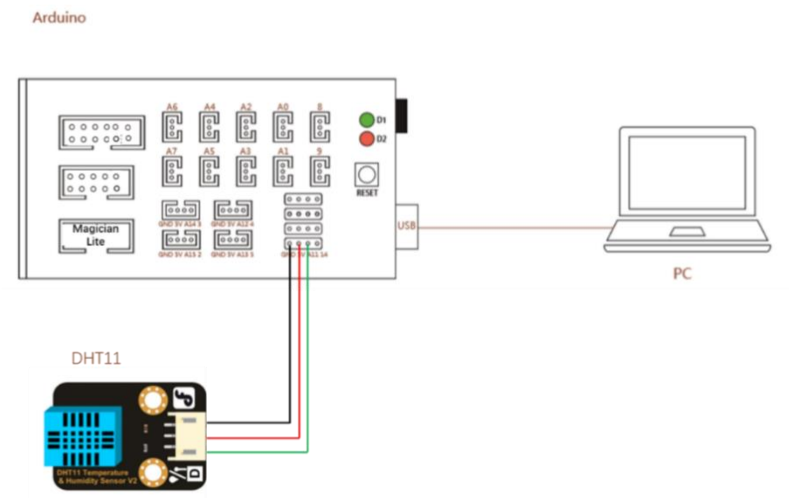


Figure 3.1 Equipment connection diagram

The temperature and humidity sensor is a temperature and humidity composite sensor with a calibrated digital signal output. It applies special digital module acquisition technology and temperature and humidity sensing technology to ensure that the product has extremely high reliability and excellent long-term stability. The sensor includes a resistive humidity sensing element and an NTC temperature measuring element, and is connected to a high-performance 8-bit microcontroller. Therefore, this product has the advantages of excellent quality, ultra-fast response, strong anti-interference ability, and high cost performance.

(2) Design Program

Step 1: According to the experiment analysis, read the program flow chart of displaying temperature and humidity data on the serial port of Arduino IDE, as shown in Figure 3.2.

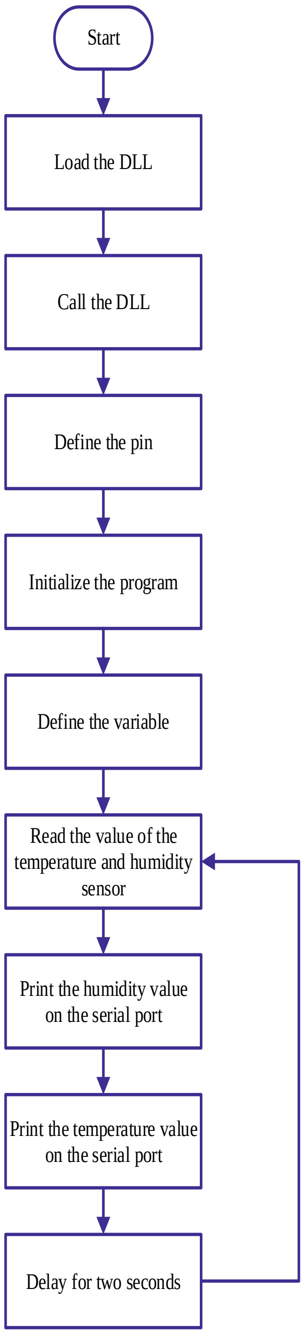

Start
Load the DLL
Call the DLL
Define the pin
Initialize the program
Define the variable
Read the value of the temperature and humidity sensor
Print the humidity value on the serial port
Print the temperature value on the serial port
Delay for two seconds

Figure 3.2 Flow chart

Step 2: Call the library file and define the pins. The program is shown below.

```
// Call the temperature and humidity sensor library file
#include<dht11.h>
dht11 DHT;
#define DHT11_PIN A11 // Select the interface of the sensor
```

Step 3: Initialize the program, set the baud rate, print character humidity and temperature. The program is shown below.

```
//Initialization
voidsetup()
{
Serial.begin(115200);   //Baud rate
  }
```

Step 4: Define the variables, store the data of the temperature and humidity sensor data, and read the data. The program is shown below.

```
//Loop
voidloop(){
intTem;
Serial.print("DHT11, \t");
Tem = DHT.read(A11);     // Read data
```

int is an integer variable, and *float* is a floating-point variable. Why do we define an integer variable here?

Step 5: Display the data. Display the humidity and temperature of the temperature and humidity sensor, with a delay of 2 seconds. The program is shown below.

```
// Display data
Serial.print(DHT.humidity,1); // Print the read humidity data
Serial.println(DHT.temperature,1);   // Print the read temperature data
delay(2000);
}
```

Step 6: Integrate the program, as shown below.

```
// Call the temperature and humidity sensor library file
#include<dht11.h>
dht11 DHT;
#define DHT11_PIN A11 // Select the interface of the sensor

//Initialization
voidsetup()
{
Serial.begin(115200);   //Baud rate
}

```

```
//Loop
voidloop(){
   int Tem; // Create a variable
Serial.print("DHT11, \t");
Tem = DHT.read(A11);     // Read data

// Display data
Serial.print(DHT.humidity,1); // Print the read humidity data
Serial.println(DHT.temperature,1);    // Print the read temperature data
delay(2000);
}
```

3. Summary

(1) The instructions to define the variables are: ____________________

____

(2) The instructions to read the temperature and humidity sensor values are: ____

____

(3) The instructions to display humidity data are: ____________________

____

(4) The instructions to display temperature data are: ________________

____

4. Self-Assessment

Check the completed content in the experiment task. Tick（√） the completed items, and circle (◯) the uncompleted items.

Assessment Content	Completion Status
I've created an integer variable	
I've used the instruction to read the values of the temperature and humidity sensor	
I've programmed to read the values of the temperature and humidity sensor	

Task 2: Display the Parameters of the Temperature and Humidity Sensor through the LCD

To read the values of the temperature and humidity sensor, view the serial port monitor of the computer. If it is not easy to do so, you can use the LCD to display the sensor values, and directly view them after the program runs.

1. Analysis

Ask the teacher to show the values of the temperature and humidity sensor on the LCD. Observe the results, analyze the steps to read the parameters of the temperature and humidity sensor from the LCD, and complete the steps in the blanks in the table below.

Step 1: Connect the device, motherboard, temperature and humidity sensor, LCD display, and computer.

Step 2: Define temperature and humidity sensors and LCD display pins.

Step 3: Create a variable.

Step 4: Read the values of the temperature and humidity sensor.

Step 5: Set the display position of ________, and display the current environment ________ on the LCD screen.

The LCD liquid crystal display uses the physical characteristics of liquid crystal and controls its display area through voltage, i.e., it can show corresponding characters, such as the number 123, upper and lower case letters Aa, and the symbol %.

2. Steps

(1) Prepare Hardware

Step 1: Connect Arduino control board, DHT11 temperature and humidity sensor, LCD liquid crystal module, and PC, as shown in Figure 3.3.

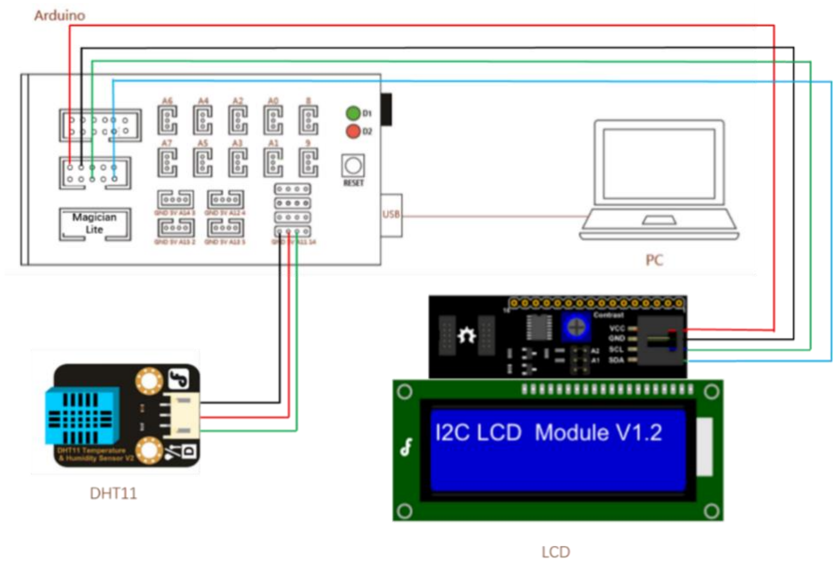


Figure 3.3 Equipment connection diagram

The wiring of the LCD module in the Mage2560 control board is SCL-21 and SDA-20 respectively.

(2) Program design

Step 1: According to the results of the experiment analysis, read the program flow chart of showing the temperature and humidity data on the LCD, as shown in Figure 3.4.

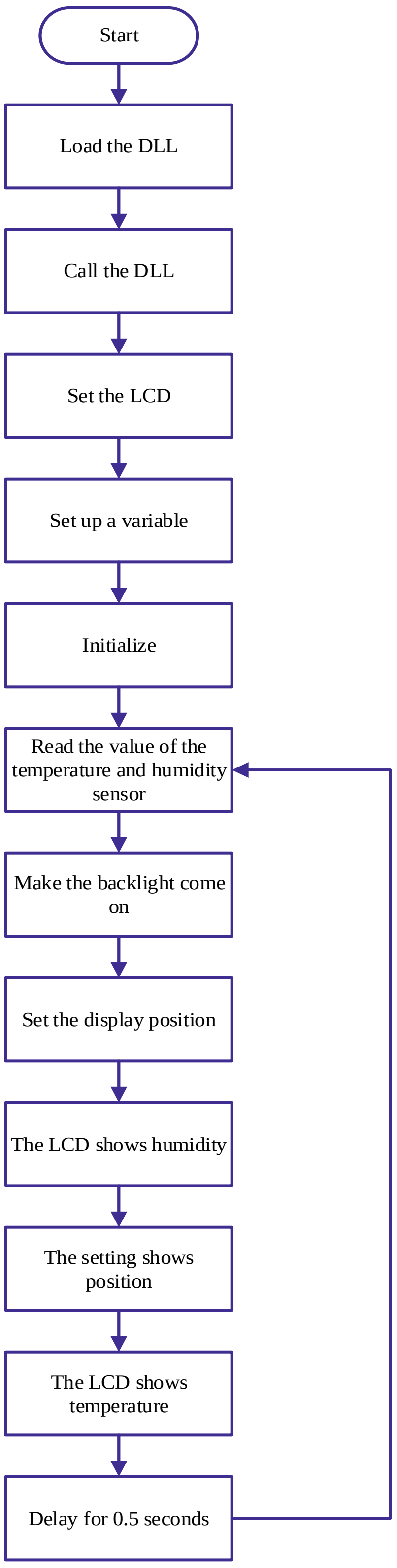


Figure 3.4 Flow chart

Step 2: Call the library file. In this experiment project, we need to use the LCD and the temperature and humidity sensor, call the library file of the display and the temperature and humidity sensor. The program is shown below.

```
// Call library file
#include<Wire.h>
#include<LiquidCrystal_I2C.h>
#include<dht11.h>
```

Step 3: Set the LCD to 2 lines and 16 characters, set the functions of the temperature and humidity sensor, and create a variable program as follows.

```
LiquidCrystal_I2C lcd (0x20,16,2); // Set the LCD to 2 lines with 16 characters each
line
  dht11 DHT;
intchk;
```

The LCD module is displayed in the form of a dot matrix, and the content of each byte corresponds to the light and dark of the corresponding position on the display. For example, 2 lines with 16 characters per line are shown in the figure below.

Step 4: Perform initialization. Set the pins and baud rate of the temperature and humidity sensor pins, and initialize the LCD. The program is shown below.

```
void setup()
{
  #define DHT11_PIN A11
Serial.begin(115200); // Set baud rate
lcd.init();                              // Initializelcd
  }
```

Step 5: Read the data of the temperature and humidity sensor and turn on the LCD backlight. The program is shown below.

```
Tem = DHT.read (A11); // Read temperature and humidity sensor data
lcd.backlight (); // Turn on the LCD backlight
```

☛ Do not turn on the LCD backlight, but observe the results displayed by the LCD.

Step 6: Show the read data of the temperature and humidity sensor. The humidity appears on the first line of the LCD and the temperature on the second line. The program is shown below.

```
// Display humidity
lcd.setCursor (0,0); // Show the cursor position
lcd.print("H:");
lcd.print(DHT.humidity);

  // Display temperature
lcd.setCursor (0,1); // Show cursor position
lcd.print("T:");
lcd.print(DHT.temperature);
lcd.print("C");
```

Step 7: Integrate the program. The program is shown below.

```
// Call the library file
#include<Wire.h>
#include<LiquidCrystal_I2C.h>
#include<dht11.h>

LiquidCrystal_I2C lcd (0x20,16,2); // Set the LCD to 2 lines with 16 characters each
line
dht11 DHT;
intTem;

void setup()
{
  #define DHT11_PIN A11
Serial.begin(115200); // Set baud rate
lcd.init (); // Initialize lcd
```

```
 }
Void loop()
{
Tem = DHT.read (A11); // read temperature and humidity sensor data
lcd.backlight (); // Turn on the LCD backlight
// Display humidity
lcd.setCursor (0,0); // Show cursor position
lcd.print("H:");
lcd.print(DHT.humidity);

  // Display temperature
lcd.setCursor (0,1); // Show cursor position
lcd.print("T:");
lcd.print(DHT.temperature);
lcd.print("C");

delay(500);
}
```

3. Summary

(1) Briefly describe the display principle of the LCD:________________

____

(2) The instructions to set the cursor position of the LCD are: _______

____

(3) The instructions to display characters on the LCD are: __________

____

4. Self-Assessment

Check the completed content in the experiment task. Tick（√） the completed items, and circle (◯) the uncompleted items.

Assessment Content	Completion Status
I've known how the LCD works	
I've set the cursor position of the LCD	
I've used the instructions to display characters on the LCD	

I've programmed to display the temperature and humidity sensor parameters on the LCD	

Task 3: Design the Cooling System

As surveys show, the suitable growth temperature is about 20°C for radishes, 20-25°C for germination, 18-22°C for leaves, and 18-22°C for stems. If the temperature is too high, plants grow slowly and are prone to diseases and insect pests, the worst of which are aphids and viruses; if the temperature is too low, plants grow slowly. During leaf growth, we should uncover the film to ventilate and reduce humidity, enhance ventilation and increase air flow.

1. Analysis

During leaf growth, when the temperature of the temperature and humidity sensor exceeds 23°C on the LCD, start the small fan module to accelerate the air velocity in the greenhouse to reduce the temperature and humidity. Analyze the steps to run the cooling system. Then, complete the steps in the blanks in the table below.

Step 1: Connect the device, control board, temperature and humidity sensor, LCD display, fan module, and PC.

Step 2: Temperature and humidity sensor, LCD display, and fan module pins.

Step 3: Create a variable, ______________.

Step 4: Read the values of the temperature and humidity sensor.

Step 5: Set the display position of ______, and show ________the current environment on the LCD display.

Step 6: Determine whether the temperature exceeds 23°C.

Step 7: When the temperature exceeds 23°C, ___________, ventilate; when the temperature is below 23°C, you do not need to ventilate ________.

According to surveys, during the growth and development of crops, their required temperature and moisture range varies with their types and varieties. The most suitable temperature for radish seed germination is 20-25°C, and the initial germination temperature is 2-3°C. Radish seedlings can withstand the temperature of up to 25°C, and the temperature from -2 to -3°C for a short time. The optimum growth temperature is 18-22°C for leaves, and 15-18°C for endoplasmic roots. At a temperature of over 25°C, this plant grows weak with poor quality. Therefore, its suitable growth temperature is high in the early period and low in the later period, but this order is reverse for radish planting in high mountains. The use of mulch can prevent radish planting from being affected by temperature. Korean cold-resistant varieties have a wide range of adaptability. As long as the temperature at the early stage is not lower than 10°C, their growth and quality are not affected. Therefore, in this experiment we set 23°C as the critical point of high temperature.

2. Steps

(1) Prepare Hardware

Step 1: Connect Arduino control board, DHT11 temperature and humidity sensor, LCD liquid crystal module, DC Motor, PC, as shown in Figure 3.5.

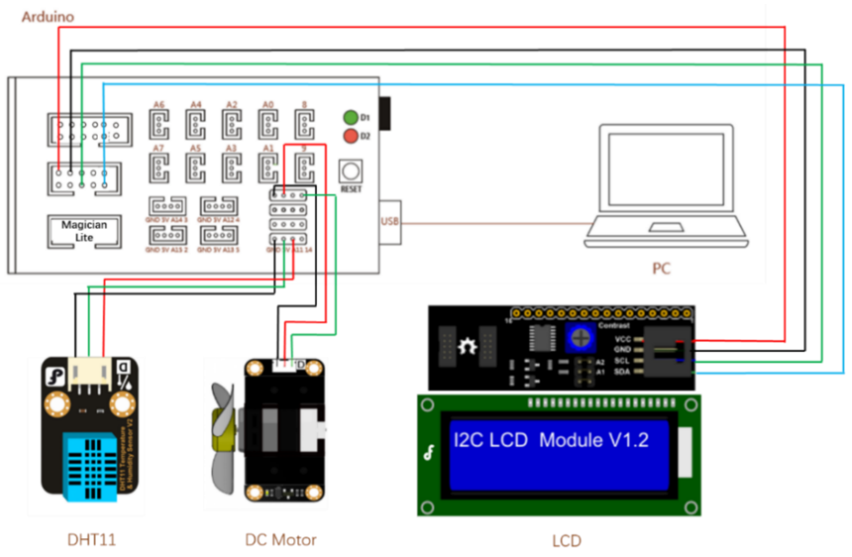


Figure 3.5 Equipment connection diagram

(2) Design Program

Step 1: According to the experiment analysis, read the flow chart of the cooling system, as shown in Figure 3.6.

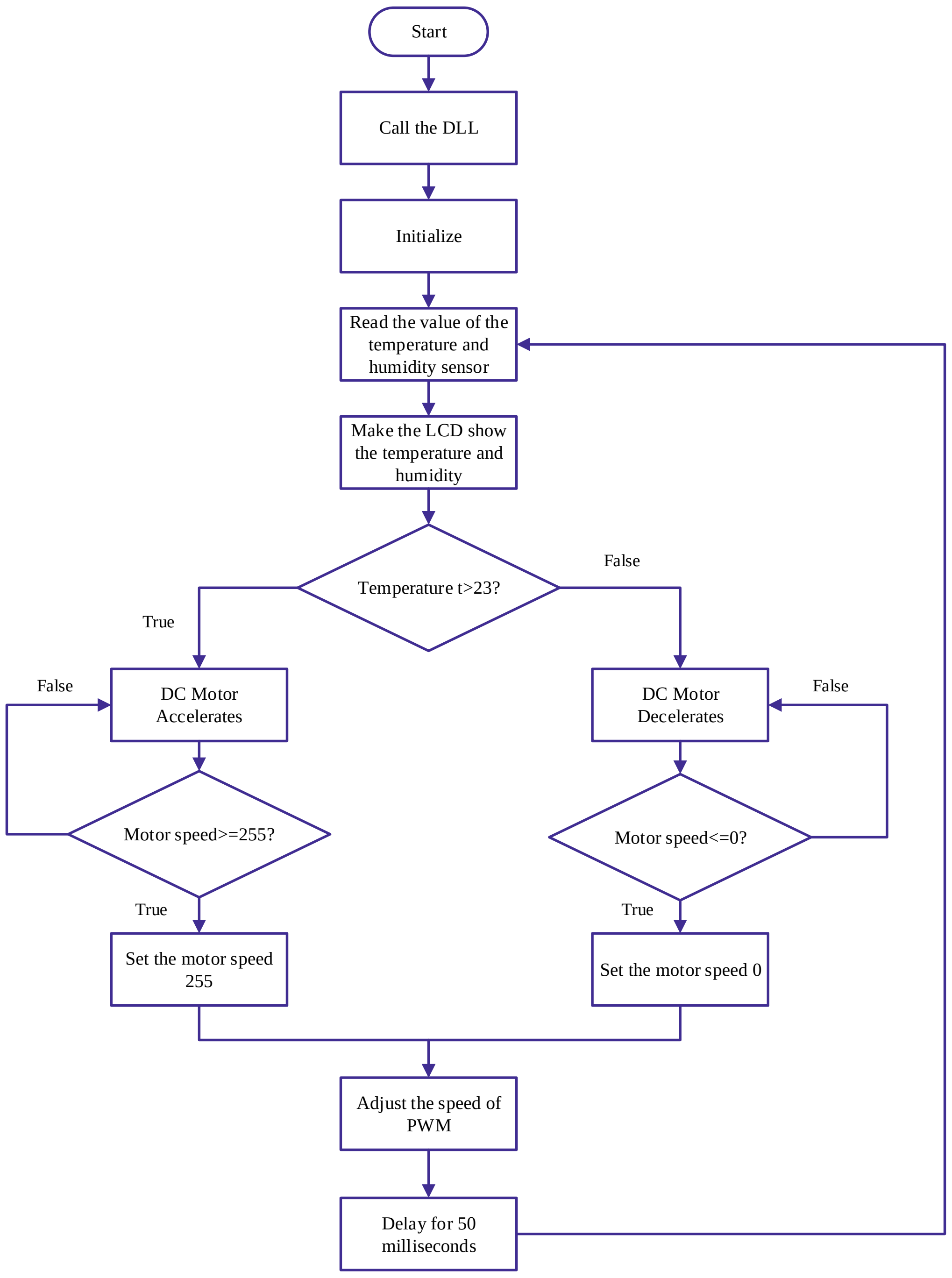


Figure 3.6 Flow chart of the cooling system

Step 2: Provide no library file the DC motor. For this, see Task 2.

Step 3: Perform initialization. Initialize the LCD, set the display address, motor pins, and temperature and humidity sensor pins, create variables of motor speed and temperature, and set the baud rate. The program is shown below.

```
dht11 DHT;
LiquidCrystal_I2C lcd (0x20,16,2); // Set the LCD to 2 lines and 16 characters
int motorSpeed; //Motor speed variable
int motorPin = 11; //Motor drive pin 11
#define DHT11_PIN A11 // Set the DHT pin to A11
intchk;
int t; //temperature variable

voidsetup()
{
Serial.begin (115200); // Set the baud rate
lcd.init (); // Initialize lcd
  }
```

Step 4: Repeatedly read the data of the temperature and humidity sensor and the humidity and temperature on the LCD. The program is shown in task 2.

Step 5: Use the variable *t* to store temperature data to determine whether the temperature exceeds 23°C. The program is shown below.

```
    t=DHT.temperature;
if(t >23)
{
```

Step 6: At a temperature of over 23℃, the DC motor accelerates, and starts the fan, which keeps rotating at the maximum speed of 255, thus cooling the system. The program is shown below.

```
motorSpeed+=5;
if(motorSpeed>=255)
motorSpeed=255;
}
```

Step 7: At a temperature of lower than 23°C, the DC motor decelerates to a speed of 0. The program is shown below.

```
  else
    {
motorSpeed-=5;
if(motorSpeed<=0)
motorSpeed=0;
        }
```

Step 8: Adjust the motor speed through PWM, with a delay of 500 milliseconds. The program is shown below.

```
analogWrite(motorPin, motorSpeed);     //PWM speed regulation
delay(50);
```

Step 9: Integrate the program. The cooling system program is shown below.

```
// Call library file
#include<Wire.h>
#include<LiquidCrystal_I2C.h>
#include<dht11.h>

dht11 DHT;
LiquidCrystal_I2C lcd (0x20,16,2); //Set the LCD to display in 2 lines and 16
characters
int motorSpeed; // Variable for the motor speed
int motorPin = 11; // Motor drive pin 11
#define DHT11_PIN A11 // Set the DHT pin to A11
intchk;
int t; // Variable for temperature

voidsetup()
{
Serial.begin (115200); // Set the baud rate
lcd.init (); // Initialize lcd
  }

voidloop()
{

chk = DHT.read(DHT11_PIN);     // Read temperature and humidity sensor data
lcd.backlight (); // Turn on the LCD backlight
// Display humidity
lcd.setCursor (0,0); // Show cursor position
lcd.print("H:");
lcd.print(DHT.humidity,1);
lcd.print("%");

  // Display temperature
lcd.setCursor (0,1); // Show cursor position
lcd.print("T:");
    lcd.print(DHT.temperature,1);
    lcd.print("C");
    t=DHT.temperature;
if(t >23)
{
motorSpeed+=5;
if(motorSpeed>=255)
motorSpeed=255;
    }
else
    {
motorSpeed-=5;
if(motorSpeed<=0)
motorSpeed=0;
        }
analogWrite(motorPin, motorSpeed);      //PWM speed regulation
delay(50);
}
```

3. Summary

(1) The instructions to set the acceleration and deceleration of the motor are: ____

____

(2) The instructions to adjust the motor speed through PWM are: ___

____

4. Self-Assessment

Check the completed content in the experiment task. Tick（√） the completed items, and circle (◯) the uncompleted items.

Assessment Content	Completion Status
I've used the instructions to set the acceleration of the motor	
I've used the instructions to set the deceleration of the motor	
I've known the instructions to adjust the motor speed through PWM	

Assessment Content	Completion Status
I’ve designed the cooling program design	

Task 4: Design the Heating System

To make the seedlings grow faster, the cultivation farm usually uses a sunshine lamp to heat the seedling growth environment to ensure the yield of seedlings. The power of the sunlight lamp is too large to suit the use in the classroom. In this experiment, we will use RGB colored lights instead of the sunlight lamp to simulate the temperature rise.

1. Analysis

When the temperature of the temperature and humidity sensor on the LCD is lower than 10°C, start the RGB colored lights to simulate the heating of the sunlight. Analyze the implementation steps of the heating system. Complete the steps are in the blanks in the table below.

Step 1: Connect the device, control board, temperature and humidity sensor, LCD display, RGB lights, and PC.

Step 2: Connect the temperature and humidity sensor, LCD and RGB colored light pins.

Step 3: Create a variable, __________.

Step 4: Read the values of the temperature and humidity sensor.

Step 5: Set the display position of ______, and display ______the current environment on the LCD.

Step 6: Determine whether the temperature is below 10°C.

Step 7: When the temperature is lower than 10°C, ___________; when the temperature exceeds 10°C, _______________.

Step 8: Set the RGB color function.

2. Steps

(1) Prepare Hardware

Step 1: Connect Arduino control board, DHT11 temperature and humidity sensor, LCD module, RGB module, PC, as shown in Figure 3.7.

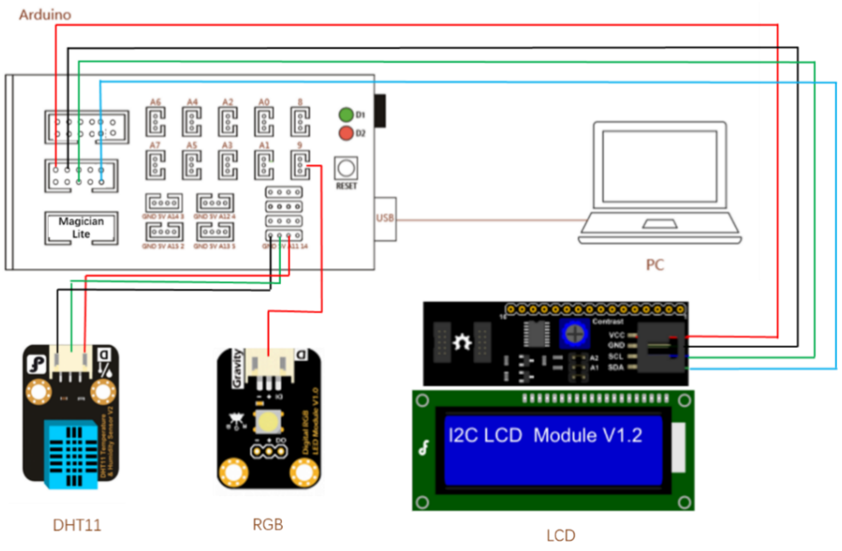


Figure 3.7 Equipment connection diagram

The full-color RGB module offers R, G, and B three-color output, and achieve the full-color effect by mixing the three colors: R, G, and B. Multiple RGB modules can be connected in series to achieve marvelous lights, blinking, rainbow transformation, and even cool effects such as text, letters, pictures and animations.

(2) Design Program

Step 1: According to the experiment analysis, read the flow chart of the heating system, as shown in Figure 3.8.

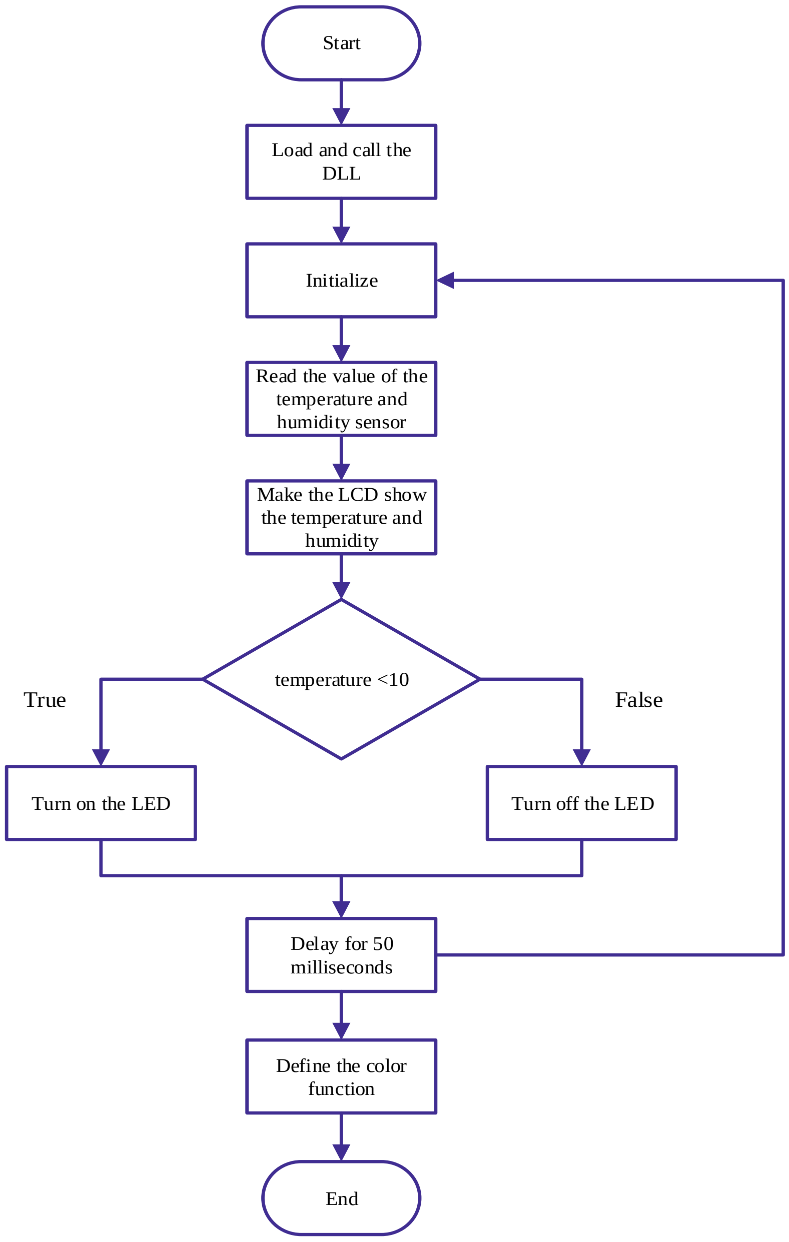

Start
Load and call the DLL
Initialize
Read the value of the temperature and humidity sensor
Make the LCD show the temperature and humidity
temperature <10
True
False
Turn on the LED
Turn off the LED
Delay for 50 milliseconds
Define the color function
End

Figure 3.8 Flow chart of the heating system

Step 2: Call the library file. In this experiment project, we need to use the LCD, the temperature and humidity sensor, and the RGB module, and call their library files. The program is shown below.

```
// Call the library file
#include<Wire.h>
#include<LiquidCrystal_I2C.h>
#include<dht11.h>
#include <Adafruit_NeoPixel.h>
```

Step 3: Initialize the LCD and DHT temperature and humidity sensor in the same way as in Task 2. In this experiment, we need to initialize the RGB module. The program is shown below.

```
// Define the number of RGB pins and lights
#define PIN_LED 9
#define NUM_LED 1
// Define objects of RGB module
Adafruit_NeoPixelRGB_Strip = Adafruit_NeoPixel(NUM_LED, PIN_LED,
NEO_GRB + NEO_KHZ800);
```

Step 4: Repeatedly read the values of the temperature and humidity sensor and the humidity and temperature on the LCD. The program is shown in Task 2.

Step 5: Determine whether the temperature is lower than 10°C. If the temperature is lower than 10°C, start the RGB module to heat up; otherwise, turn off the RGB module. The program is shown below.

```
if(t < 10)
   {
  //Turn on the lights
colorWipe(RGB_Strip.Color(255, 255, 255), 1000);
RGB_Strip.setBrightness(128);
   }
else
   {
  //Turn off the lights
colorWipe(RGB_Strip.Color(255, 255, 255), 1000);
RGB_Strip.setBrightness(0);
         }
```

Step 6: Define the color function colorWipe(RGB_Strip.Color (R, G, B), time). The program is shown below.

```
  // Define the color function
Void colorWipe(int c, int wait) {
   for (uint16_t i = 0; i<RGB_Strip.numPixels(); i++) {
RGB_Strip.setPixelColor(i, c);
RGB_Strip.show();
  delay(wait);
   }
}
```

Step 7: Integrate the program. The heating system program is shown below.

```
// Call library file
#include<Wire.h>
#include<LiquidCrystal_I2C.h>
#include<dht11.h>
#include <Adafruit_NeoPixel.h>

dht11 DHT;
LiquidCrystal_I2Clcd(0x20,16,2);    // Set the LCD to 2 lines and 16 characters

int Tem;
int t; // Variable for temperature
// Define the number of RGB pins and lights
#define PIN_LED 9
#define NUM_LED 1
// Define the object of RGB module
Adafruit_NeoPixelRGB_Strip = Adafruit_NeoPixel(NUM_LED, PIN_LED,
NEO_GRB + NEO_KHZ800);
Void setup()
{
#define DHT11_PIN A11 // Define the pins of the temperature and humidity sensor

Serial.begin (115200); // Set the baud rate
lcd.init();                                   // Initialize lcd
RGB_Strip.begin();
RGB_Strip.show();

  }
Void setup()
{
Serial.begin(115200); // Set baud rate
lcd.init();                                   // Initialize lcd
  }

Void loop()
{

Tem = DHT.read (A11); // Read temperature and humidity sensor data
lcd.backlight (); // Turn on the LCD backlight
// Display humidity
lcd.setCursor (0,0); // Show cursor position
lcd.print("H:");
lcd.print(DHT.humidity);
```

```

  // Display temperature
lcd.setCursor (0,1); // Show cursor position
lcd.print("T:");
lcd.print(DHT.temperature);
lcd.print("C");
    t=DHT.temperature;
if(t < 10)
{
// Turn on the lights
colorWipe(RGB_Strip.Color(255, 255, 255), 1000);
RGB_Strip.setBrightness(128);
    }
else
    {
        // Turn off the lights
colorWipe(RGB_Strip.Color(255, 255, 255), 1000);
RGB_Strip.setBrightness(0);
            }
        delay(50);
}
// define the color function
voidcolorWipe(int c, int wait) {
    for (uint16_t i = 0; i<RGB_Strip.numPixels(); i++) {
RGB_Strip.setPixelColor(i, c);
RGB_Strip.show();
  delay(wait);
    }
        }
```

3. Summary

(1) The instructions to turn the lights on and off are: ________________
____

(2) The way to define a function is: ______________________________
____

4. Self-Assessment

Check the completed content in the experiment task. Tick（√） the completed items, and circle (◯) the uncompleted items.

Assessment Content	Completion Status
I’ve used the RGB light	
I’ve used the instruction to turn on the light	
I’ve used the instruction to turn off the light	
I’ve defined the function to set the color of the RGB light	
I’ve programmed the heating system	

Task 5: Design the Temperature Control

According to the requirements of different time and temperature of radish growth, people have designed a temperature control system to automatically adjust the temperature in the greenhouse. For example, during seedling cultivation, there must have a high temperature (make sure the temperature is higher than 10°C); during leaf growth, there must have an optimal growth temperature (18-22°C), which may not exceed 23°C.

1. Analysis

When the temperature of the temperature and humidity sensor on the LCD is lower than 10°C, perform the operation to raise temperature (using a RGB colored light instead of a sunlight for simulation); when such a temperature is over 23°C, the fan module starts to cool down and ventilate; when such a temperature is between 10°C and 26°C, neither the fan nor the RGB light starts. Analyze the implementation steps of the temperature control system.

Step 1: Connect the equipment, such as the control board, the temperature and humidity sensor, the LCD, the RGB light, the fan module, and the PC.

Step 2: Connect the pins of the temperature and humidity sensor, the LCD, and the RGB colored light.

Step 3: Create a variable, __________.

Step 4: Read the values of the temperature and humidity sensor.

Step 5: Set the display position of _______, and display _______the current environment on the LCD.

Step 6: Determine whether the temperature is below 10°C.

Step 7: When the temperature is lower than10°C, ________________;

Step 8: Set the function of the RGB colored light.

Step 9: Determine whether the temperature exceeds 23°C.

Step 10: When the temperature exceeds 23 °C, _______________, and blow air for ventilation;

Step 11: Determine whether the temperature is between 10°C and 23°C. If so, you do not need to provide ventilation,____________; do not need to raise temperature, ___________.

2. Steps

(1) Prepare Hardware

Step 1: Connect Arduino control board, DHT11 temperature and humidity sensor, the LCD module, the RGB module, the DC Motor, and the PC, as shown in Figure 3.9.

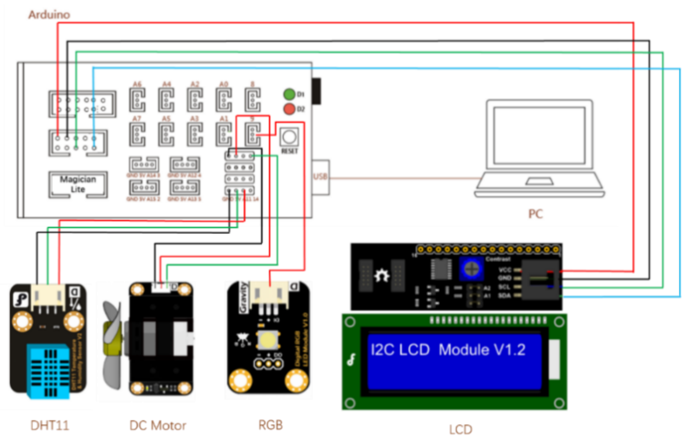


Figure 3.9 Equipment connection diagram

(2) Design Program

Step 1: According to the experiment analysis, read the flow chart of the temperature control system, as shown in Figure 3.10.

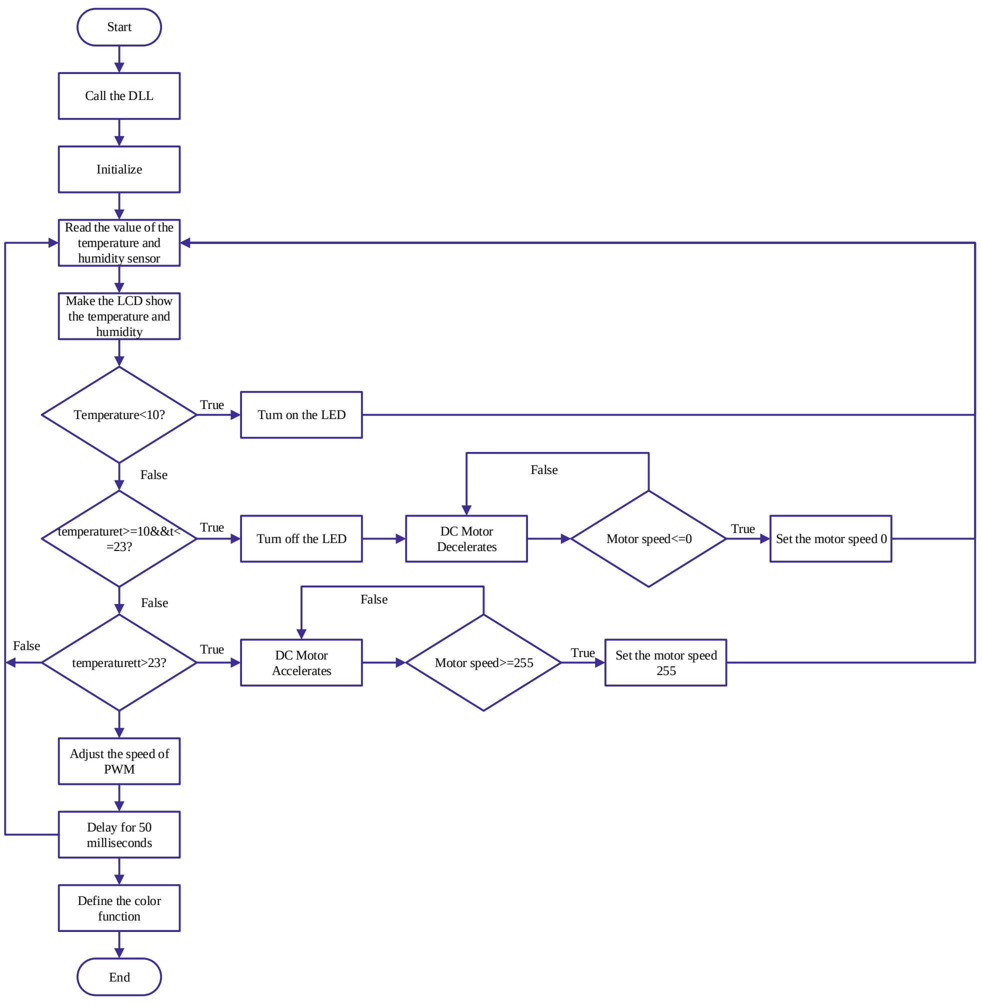


Figure 3.10 Flow chart of the temperature control system

Step 2: Call the library file. Since DC Motor does not have a library file, the library file called in this experiment is the same as task 4.

Step 3: Initialize the equipment. Initialize the LCD and DHT temperature and humidity sensors as you do in Task 2, the DC Motor as you do in Task 3, and the RGB module as you do in Task 4.

👉 Try programming the initialization for this experiment.

Step 4: Repeatedly read the values of the temperature and humidity sensor and the humidity and temperature on the LCD. The program is shown in task 2.

Step 5: Determine whether the temperature is lower than 10°C. If yes, start the RGB module to heat up; the program is shown in task 4.

Step 6: Determine whether the temperature is within 10-23°C. If yes, turn off the RGB module and let the DC Motor slow down. The program is shown below.

```
else if(t=>10&&t<=23)
  {
// Turn off the lights
colorWipe(RGB_Strip.Color(255, 255, 255), 1000);
RGB_Strip.setBrightness(0);
// Turn off the fan
motorSpeed-=20;
if(motorSpeed<=0)
motorSpeed=0;
      }
```

Step 7: Determine whether the temperature exceeds 26°C. If yes, start the fan module. The program is shown in task 3.

Step 8: Adjust the motor speed through PWM, with 500 ms delay. The program is shown in Task 3.

Step 9: Define the color function colorWipe (RGB_Strip.Color (R, G, B), time). The program is shown in Task 4.

Step 10: Integrate the program. The temperature control system program is as follows.

```
// Call the library file
#include<Wire.h>
```

```
#include<LiquidCrystal_I2C.h>
#include<dht11.h>
#include <Adafruit_NeoPixel.h>

dht11 DHT;
LiquidCrystal_I2C lcd (0x20,16,2); // Set the LCD to 2 lines and 16 characters
int motorSpeed; // Variable for motor speed
int motorPin = 11; // Motor drive pin 11
#define DHT11_PIN A11 // Set the DHT pin to A11
int Tem;
int t; // Variable for temperature
// Define the number of RGB pins and lights
#define PIN_LED 9
#define NUM_LED 1
// Define the objects of RGB module
Adafruit_NeoPixelRGB_Strip = Adafruit_NeoPixel(NUM_LED, PIN_LED,
NEO_GRB + NEO_KHZ800);
Void setup()
{
#define DHT11_PIN A11 // Define the pins of the temperature and humidity sensor

Serial.begin (115200); // Set the baud rate
lcd.init (); // Initialize the LCD
RGB_Strip.begin();
RGB_Strip.show();

  }
Void setup()
{
Serial.begin (115200); // Set the baud rate
lcd.init (); // Initialize LCD
  }

Void loop()
{

Tem = DHT.read (A11); // Read the data of the temperature and humidity sensor
lcd.backlight (); // Turn on the LCD backlight
// Display humidity
lcd.setCursor (0,0); // Show cursor position
lcd.print("H:");
lcd.print(DHT.humidity);

```

```
  // Display temperature
lcd.setCursor (0,1); // Show cursor position
lcd.print("T:");
lcd.print(DHT.temperature);
lcd.print("C");
    t=DHT.temperature;
if(t < 10)
{
// Turn on the lights
colorWipe(RGB_Strip.Color(255, 255, 255), 1000);
RGB_Strip.setBrightness(128);
    }
else if(t=>10&&t<=23)
    {
        // Turn off the lights
colorWipe(RGB_Strip.Color(255, 255, 255), 1000);
        RGB_Strip.setBrightness(0);
        // Turn off the fan
        motorSpeed-=5;
if(motorSpeed<=0)
motorSpeed=0;
            }
      else if(t >23)
{
        // start the fan
motorSpeed+=5;
      if(motorSpeed>=255)
motorSpeed=255;
    }
        analogWrite(motorPin, motorSpeed);      //PWM speed regulation
        delay(50);
        }
// define the color function
Void colorWipe(int c, int wait) {
    for (uint16_t i = 0; i<RGB_Strip.numPixels(); i++) {
RGB_Strip.setPixelColor(i, c);
RGB_Strip.show();
  delay(wait);
    }
        }
```

3. Summary

(1) The expression of double judgment is:________________________
____

(2) The characteristics of the multi-branch statement are:__________
____

4. Self-Assessment

Check the completed content in the experiment task. Tick（√） the completed items, and circle (◯) the uncompleted items.

Assessment Content	Completion Status
I've known how to express the double judgment statement	
I've used the multi-branch structure	
I've designed the programming for a temperature control system	

Experiment 4

Watering

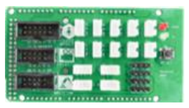
Overview

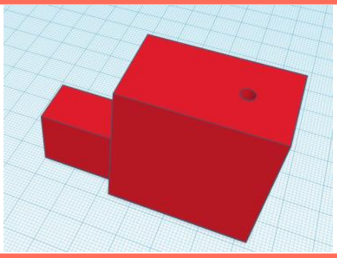
Crops cannot grow without sunlight, air and water. Irrigation is to water farmland. Too much or little moisture in soil is not good for crop growth. Therefore, it is important to monitor soil moisture and water crops in time, because these measures ensure their growth and harvest. In this experiment, we simulate a scenario of automatic watering in an intelligent farm. Through a sensor, the system automatically learns about soil moisture. As soil moisture is lower than the related index, the system can control the robot to automatically water farmland.

Objective

- By making the intelligent watering system, master how to apply the soil humidity sensor.
- By designing the new end of the robotic arm, master how to apply the 3D model.
- By making the intelligent watering system, learn how to use the LCD1602 display.
- By making the intelligent watering system, learn how to control the pump
- By making the intelligent watering system, learn how to control the relay.

Equipment

Equipment Picture	Name	Quantity
	Dobot Magician Lite robotic arm	1
	Type-C cable	1
	Power adapter	1
	Arduino Mage 2560 control board	1
	Arduino shield expansion board	1
	Soil humidity sensor	1
	Pump with a water pipe	1
	LCD1602 display	1

Equipment Picture	Name	Quantity
	Digital relay	1
	Battery box for four AA size batteries	1
	AA size battery	4
	Water pipe clamp	1

Requirements

- Take care when using electricity.
- Before the experiment, check whether the experiment equipment is complete and intact. If there is any omission or damage, please report to the teacher.
- Any specific operations in the experiment shall be performed according to the experiment manual. If you have any questions, please promptly ask the teacher.
- During the experiment, the joints of the robotic arms will start to work as the arms are powered on. In that case, do not move the joints if students do not press the unlock key.
- Report any device fault during the experiment to your teacher in a timely manner, and do not handle it yourself.
- Arrange all devices after the experiment. You shall not leave the lab before check by the group leader.

Task 1: Obtain the Soil Humidity

When should the intelligent watering system work? To answer this question, we need to measure the moisture content in soil. As moisture content is lower than the set target value, the robot will automatically water soil. In this experiment, we will test soil moisture content with a soil humidity sensor.

1. Analysis

Get the soil moisture value from a soil humidity sensor, and analyze the completed steps.

Arduino mage2560 is equipped with the digital I\O port and the simulation I\O port. The soil humidity sensor needs to be connected to port of Arduino mage2560.

2. Steps

(1) Prepare Hardware

Step 1: According to the hardware connection diagram, connect the circuit, as shown in Figure 4.1.

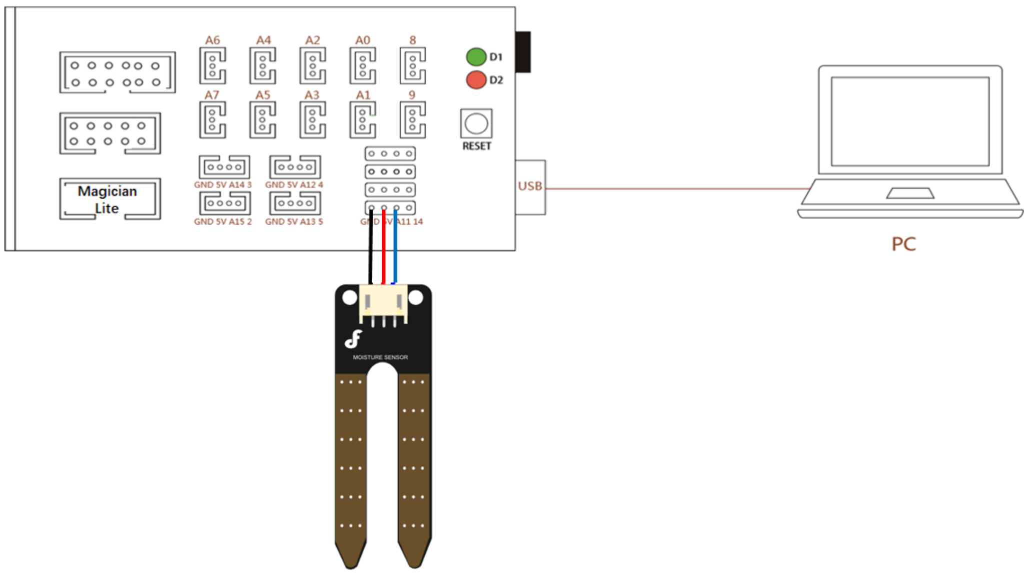


Figure 4.1 Wiring diagram of the soil humidity sensor

(2) Design Program

Step 1: Read the flow chart about how to get the soil humidity value, as shown in Figure 4.2.

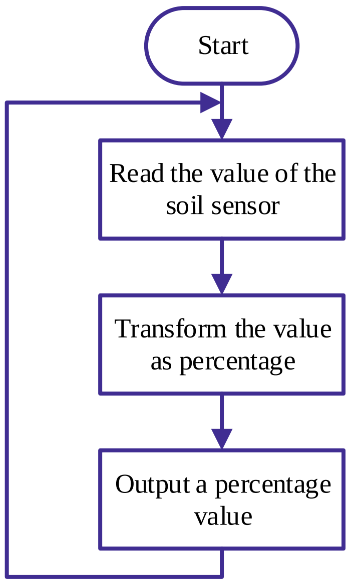

Start
Read the value of the soil sensor
Transform the value as percentage
Output a percentage value

Figure 4.2 Reading the soil humidity flow chart

Step 2: Enable Arduino programming software. Define the pin name.

```
#define HumidityPin A11    //Define the pin of the soil humidity sensor
```

Step 3: Define the variable *HumidityValue* to store the soil humidity values tested by the soil humidity sensor.

```
intHumidityValue;    //Store the soil humidity values, with a range from 0 to 1023
```

Step 4: Define the variable *HumidityPercent* to store the soil humidity percent values.

```
intHumidityPercent;    //Store the soil humidity values, with a range from 0 to 100
```

Step 5: Set the baud rate.

```
Void setup()
{
Serial.begin(115200); //Set the baud rate
}
```

Step 6: Read the value from the sensor.

```
Void loop()
{
HumidityValue = analogRead(HumidityPin); //Read the value from the sensor
```

Step 7: Convert the value from the sensor into a percent, because moisture is indicated by a percent.

```
HumidityPercent = map(300, 1023, 0, 100, HumidityValue);
        //Set the humidity value of dry soil to 300
        if(HumidityPercent>100)
        {
        HumidityPercent = 100;
        }
```

Step 8: Print and output the soil humidity percent.

```
                Serial.print("Humidity:");
                Serial.print(HumidityPercent);
                Serial.println("%");
}
```

Step 9: Compile and upload the program.

Step 10: Insert a soil humidity sensor into soil with different humidity, and switch on the serial interface monitor to observe whether the tested value changes.

3. Summary

(1) The method of getting the soil humidity value:___________________

____

____

____

(2) How we output the soil value in the percent form?________________

____

____

4. Self-Assessment

Check the completed content in the experiment task. Tick（√）the completed items, and circle (◯) the uncompleted items.

Assessment Content	Completion Status
I've learned how to use the soil humidity sensor	
I've learned how to test the soil moisture content value	

Task 2: Display the Soil Humidity on LCD1602

From the above experiment task we have learned how to get the soil moisture content, and how to show characters through LCD1602 display. Today, we will learn to show the tested soil moisture values on LCD1602 display.

1. Observation

Observe how the teacher conducts the experiment. Then, complete the steps to show the soil humidity values on LCD1602.

Step 1: Obtain the soil humidity value.

Step 2:

2. Steps

(1) Prepare Hardware

Step 1: According to the hardware connection diagram, connect the circuit, as shown in Figure 4.3.

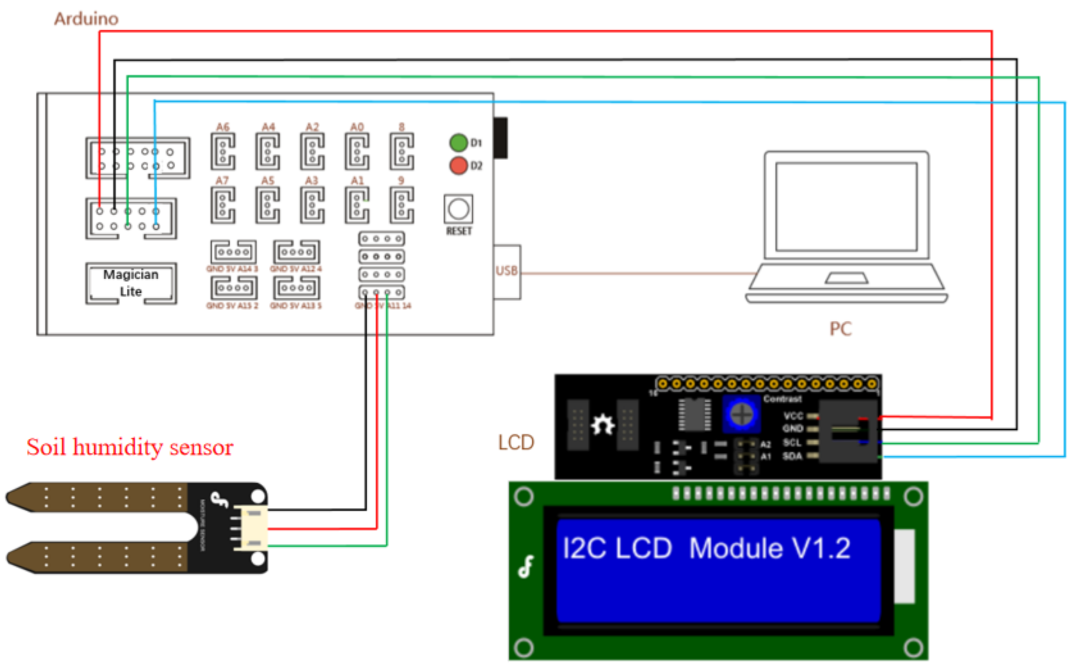


Figure 4.3 Connection diagram of the soil humidity sensor and LCD1602

(2) Design Program

Step 1: Draw the program flow chart in which LCD1602 shows soil humidity, as shown in Figure 4.4.

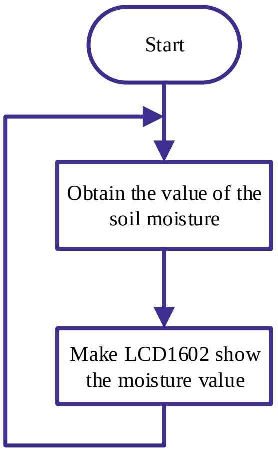

Start
Obtain the value of the soil moisture
Make LCD1602 show the moisture value

Figure 4.4 Flow chart of showing soil humidity

Step 2: Enable Arduino software, and import the library files of I2C communication of LCD1602 display.

```
#include <Wire.h>
#include <LiquidCrystal_I2C.h>
```

Step 3: Set the display parameter of LCD1602.

```
LiquidCrystal_I2C lcd(0x20,16,2);
```

Step 4: Define the value variable and percent variable of soil humidity.

```
intHumidityValue;
intHumidityPercent;
```

Step 5: Initialize LCD1602 in setup() function.

```
Void setup()
{
lcd.init();      //Initialize LCD
lcd.backlight();
}
```

Step 6: Read the value from the soil humidity sensor, and convert it into a percent. According to Task 1, independently complete this step.

```
Void loop()
{

```

```
 
 
 
```

Step 7: Allow LCD1602 display to show the percent value.

```
lcd.home();
lcd.print("Humidity:");
lcd.print(HumidityPercent);
lcd.print("%");
delay(50);
}
```

Step 8: Compile and upload the program.

Step 9: Place a soil humidity sensor in soil with different humidity, and observe the change in the value on LCD1602 display.

3. Summary

The steps to show the soil humidity value on LCD1602 in the percent form are:____
____
____
____
____

4. Self-Assessment

Check the completed content in the experiment task. Tick（√） the completed items, and circle (◯) the uncompleted items.

Assessment Content	Completion Status
I've learned how to comprehensively apply the soil humidity sensor and LCD1602 display	
I've drawn the program flow chart of how LCD1602 shows the soil humidity value	
I've programmed to allow LCD1602 to show the soil humidity value	

Task 3: Control the Pump to Draw Water

An important step for the automatic watering system is how it control the water amount. Too much water corrupts the roots of plants, while too little water is not enough for normal growth of plants. To better control the water amount, we use the pump to draw water in this task, ensuring that water amount is intelligently controlled.

1. Observation

Observe how the teacher draws water with a pump. Then, record the key steps of this process, and fill in the form.

In the hardware circuit connection, connect the control port of the relay to Arduino port 9, the COM port of the relay to the anode of the battery box, the anode of the pump to ____, and the cathode of the pump to ______.

2. Steps

(1) Prepare Hardware

Step 1: According to the hardware connection diagram, connect the circuit, as shown in Figure 4.5.

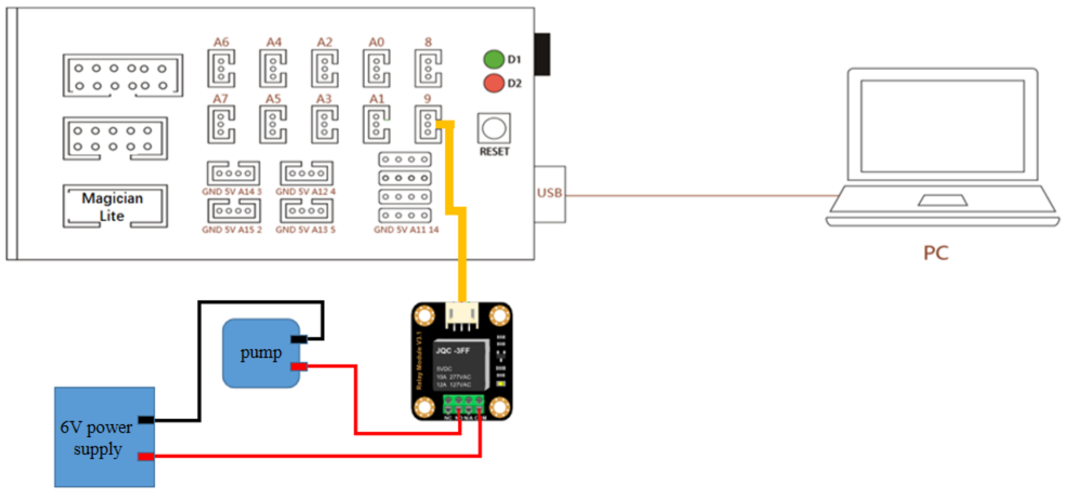


Figure 4.5 Pump wiring diagram

(2) Design Program

Step 1: Read the flow chart of pump control program, as shown in Figure 4.6.

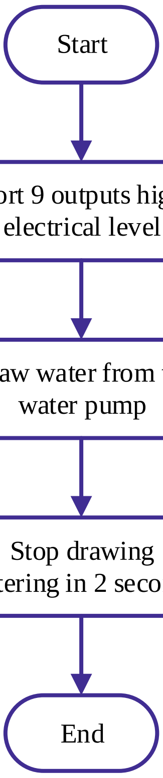

Start
Port 9 outputs high electrical level
Draw water from the water pump
Stop drawing watering in 2 seconds
End

Figure 4.6 Pump control flow chart

Step 2: Enable Arduino programming software, and define the control port of the relay.

```
int Relay = 9;
```

Step 3: Set the control port of the relay to the output mode.

```
voidsetup()
{
pinMode(Relay,OUTPUT);
}
```

Step 4: Control the pump to draw water. Input high level into the relay, and ensure the connectivity between the NO port and the COM port, thus connecting the pump to its power supply.

```
voidloop()
{
digitalWrite(Relay, HIGH);
```

Step 5: Shut down the pump, with a delay of 2 seconds.

```
delay(2000);
digitalWrite(Relay, LOW);
```

```
while(1);
}
```

Step 8: Compile and upload the program.

3. Summary

When the relay is input high level, ____ is connected to the COM port; when the relay is input low level, ________is disconnected with the COM port.

4. Self-Assessment

Check the completed content in the experiment task. Tick（√） the completed items, and circle (◯) the uncompleted items.

Assessment Content	**Completion Status**
I've learned how to use the relay	
I've learned how to use the pump	
I've understood the pump circuit diagram	
I've programmed the pumping control.	

Task 4: Design the End of the Robotic Arm

To improve pump utilization, we can use the robotic arm to control the water pipe in watering plants from different places. But the current end of the robotic arm fails to fix the water pipe, so we need to design a new end to do so.

1. Observation

Observe how the teacher designs the end of the robotic arm. Then, record the key steps of this process, and fill in the blanks below.

In designing a water pipe clamp, we need to use the basic geometric bodies: cuboid,____

____

____

In designing a water pipe clamp, we use these drawing instructions: alignment,____

____

2. Steps

Step 1: Log on to Tinkercad, and create a new project. Make the external shape of a steering engine box, and draw out a cuboid with 40 mm in length, 24 mm in width and 40 mm in height, as shown in Figure 4.7.

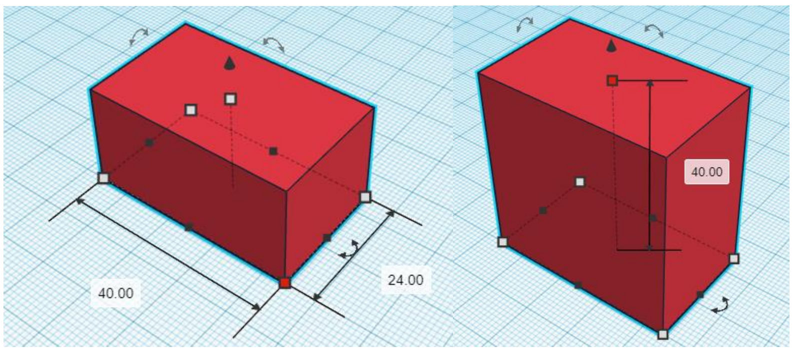


Figure 4.7 Setting the length, width and height of the cuboid

Step 2: Measure the size of the interface at the end of the robotic arm, and add to the steering engine box the section that is connected with the interface of the robotic arm. Then, drag out another cuboid, with length, width and height set to 23 mm, 17 mm and 14 mm respectively, as shown in Figure 4.8.

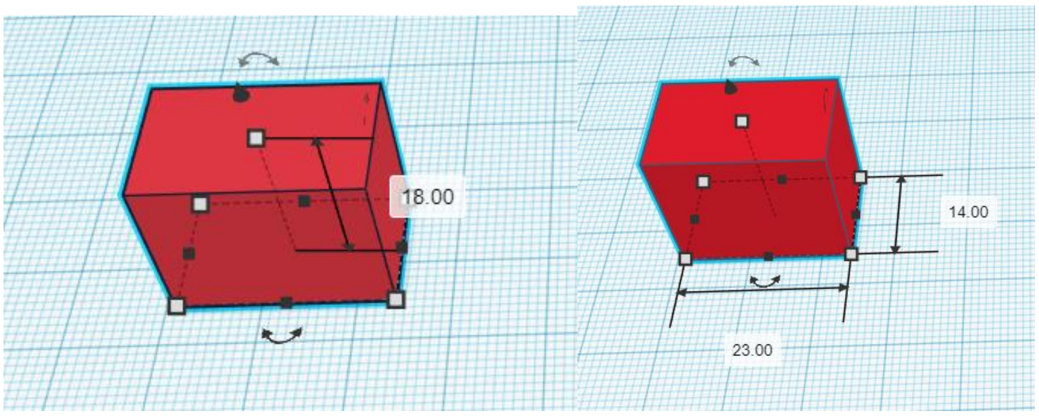


Figure 4.8 Setting the the length, width and height of the small cuboid

Step 3: Align the two objects forward and backward, move the small one rightward until it touches the large one, as shown in Figure 4.9.

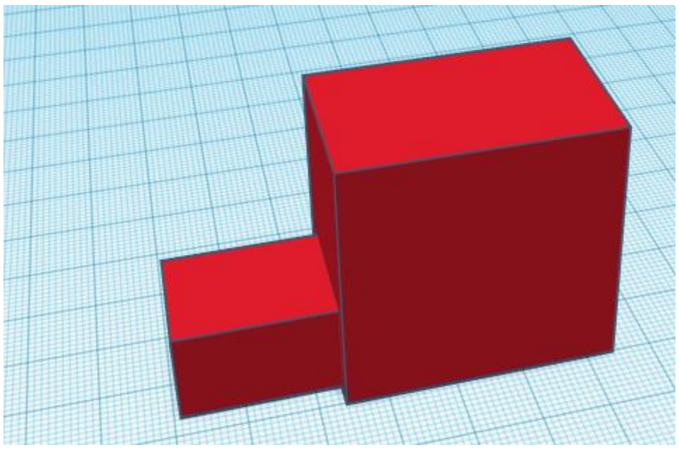


Figure 4.9 Aligning and moving the small cuboid

Step 4: Combine the two objects. Select the two objects with the cursor, and then click the **Group** icon, as shown in Figure 4.10.

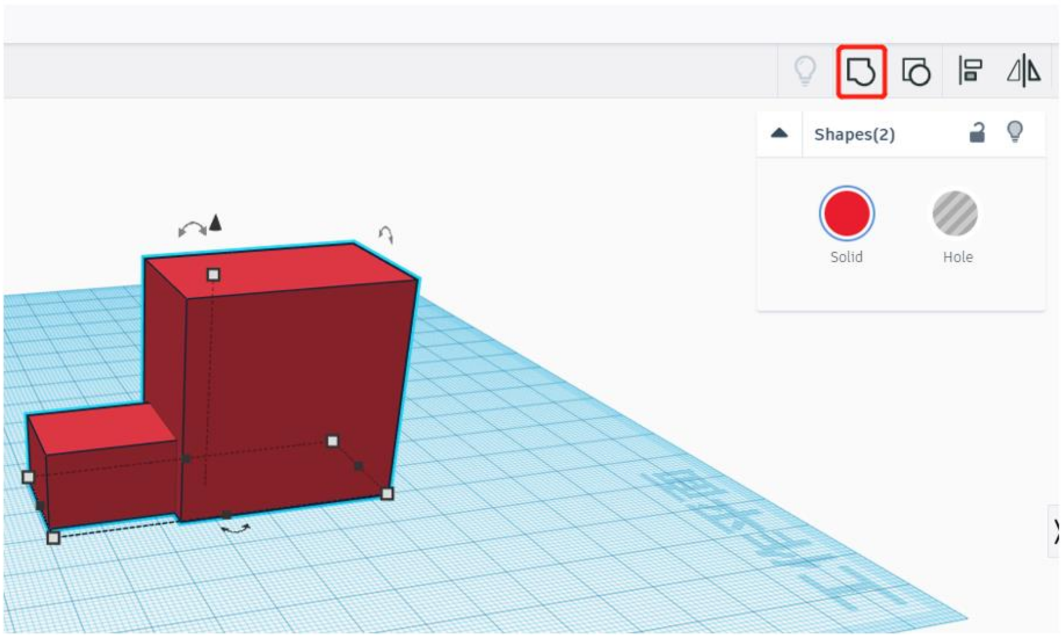


Figure 4.10 Combination

Step 5: Establish a cylinder, which is 50 mm in height, and is 1 mm less than the real water pipe in diameter. Measure the pipe diameter based on the real water pipe size. In this experiment, the water pipe has a 5 mm diameter. We can set the cylinder diameter to 4 mm, because the size 1 mm less than the real size can lock the water pipe, as shown in Figure 4.11.

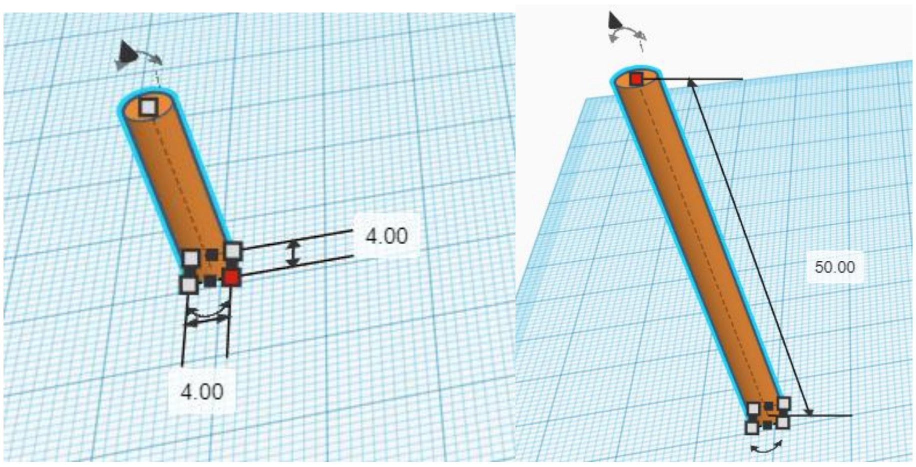
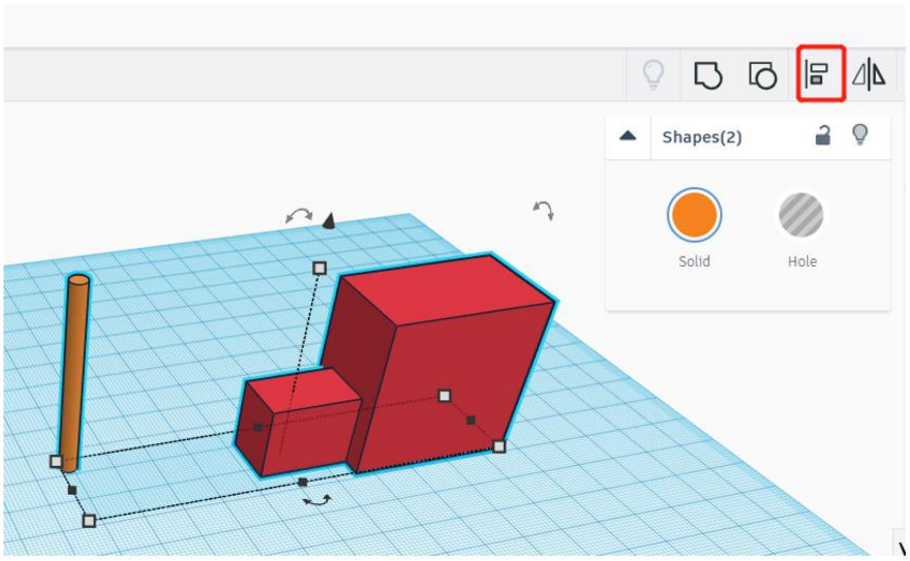


Figure 4.11 Establishing the cylinder

Step 6: Align the cylinder with the end of the designed robotic arm. Click the cylinder, press the **Shift** key, then click the end of the designed robotic arm, and finally click the **Alignment** icon in the menu in the upper right corner. After that, nine alignment points appear, and click the proper alignment points, as shown in Figure 4.12.



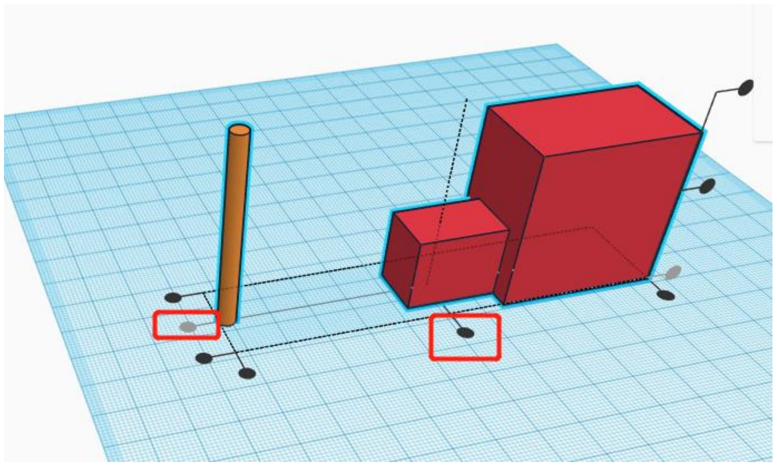


Figure 4.12 Alignment

Step 7: Move the cylinder rightward. Click the cylinder, and press the **right** key in the keyboard to move the cylinder, as shown in Figure 4.13.

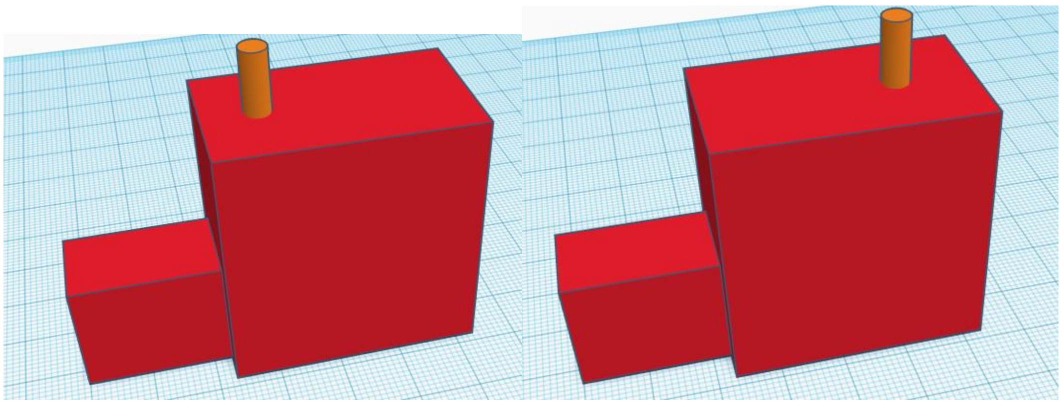
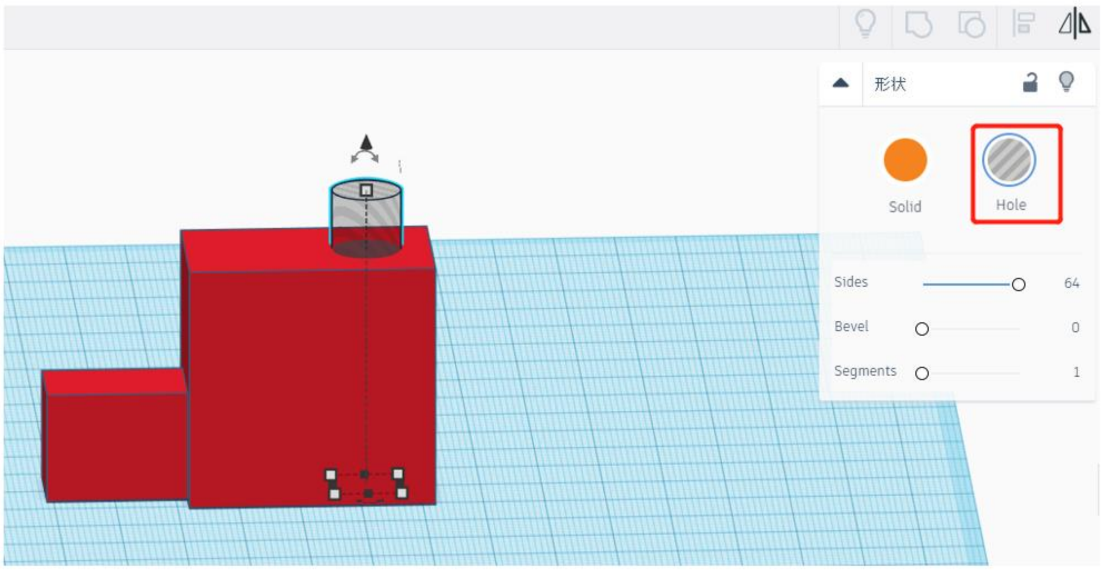
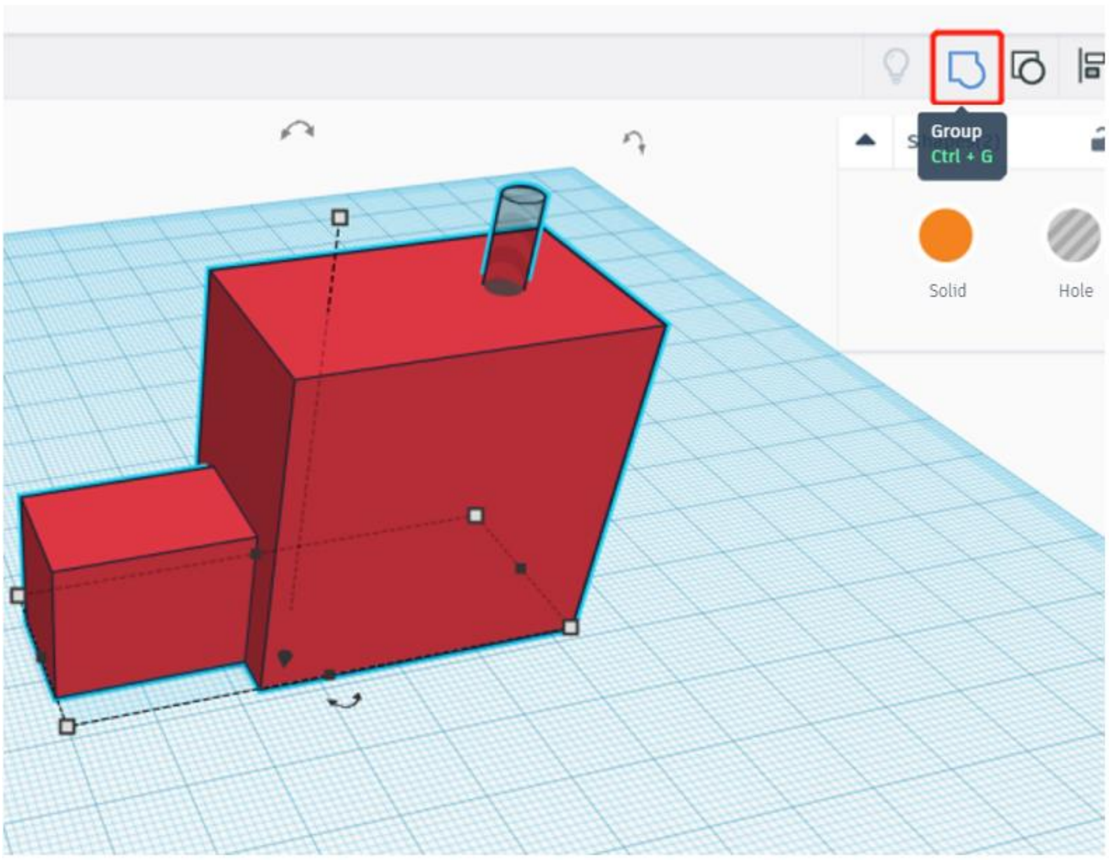


Figure 4.13 Moving the cylinder

Step 8: Cut the end of the designed robotic arm with the cylinder, and dig a hole at that end to fix the water pipe. Set the cylinder to hole, and combine it with such end, as shown in Figure 4.14.



Sides 64
Bevel 0
Segments 1



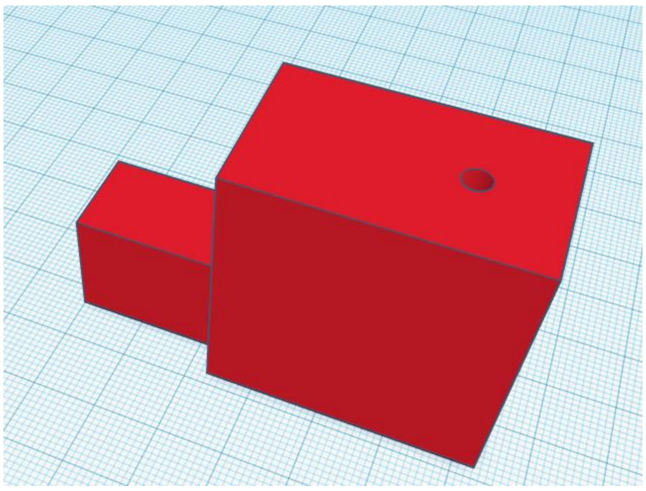


Figure 4.14 Cutting the end

3. Summary

Which basic objects do you need in designing a water pipe clamp?_____
____

Which instructions do you need in designing a water pipe clamp? _____
____

4. Self-Assessment

Check the completed content in the experiment task. Tick（√） the completed items, and circle (◯) the uncompleted items.

Assessment Content	Completion Status
I've learned how to implement the absorption function	
I've learned how to implement the combination function	
I've learned how to implement the subtraction function	

Task 5: Build the Automatic Watering System

From the above task we have made and tested each part of the automatic watering system. Now, let's build the system.

1. Observation

Observe how the teacher builds the automatic watering system. Then, record the key steps of this process, and fill in the blanks below.

Step 1: Get soil moisture through ________.
Step 2: When soil moisture is lower than 30%, the robotic arm will control the water pipe and move it to the position above the pot plant.
Step 3: ______ level into the relay, and then the pump starts to draw water.

2. Steps

(1) Prepare Hardware

Step 1: According to the hardware connection diagram, connect the circuit, as shown in Figure 4.15.

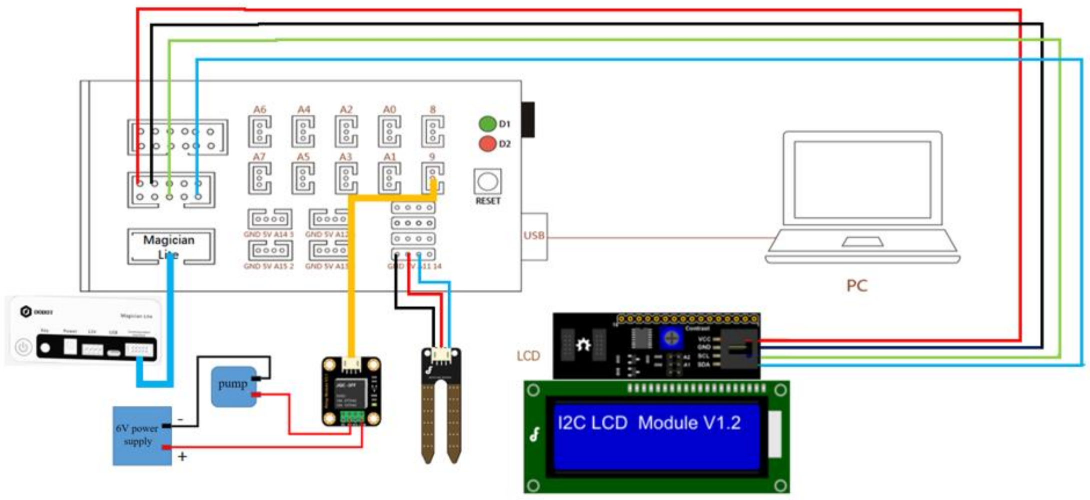


Figure 4.15 Hardware connection diagram

Step 2: Design the experiment map, and according to it set the equipment, as shown in Figure 4.16.

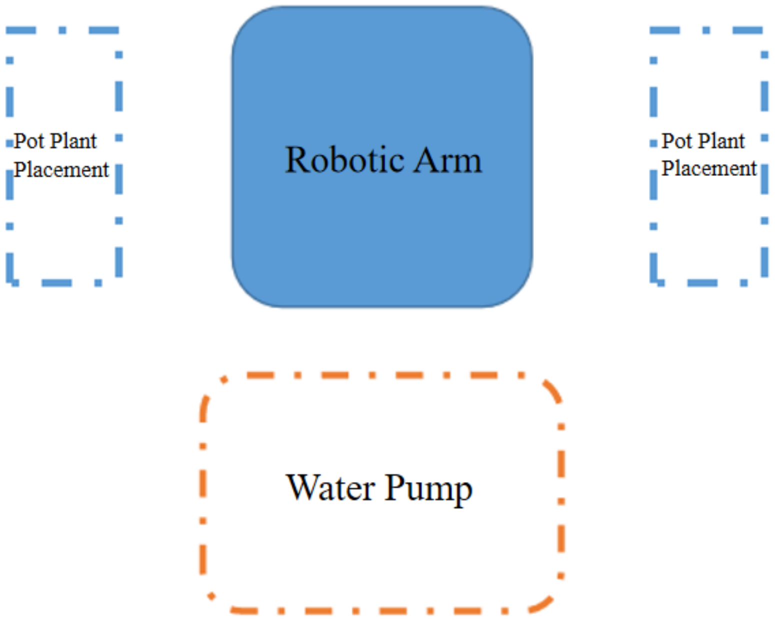


Figure 4.16 Experiment map

Step 3: Read the automatic watering flow chart, as shown in Figure 4.17.

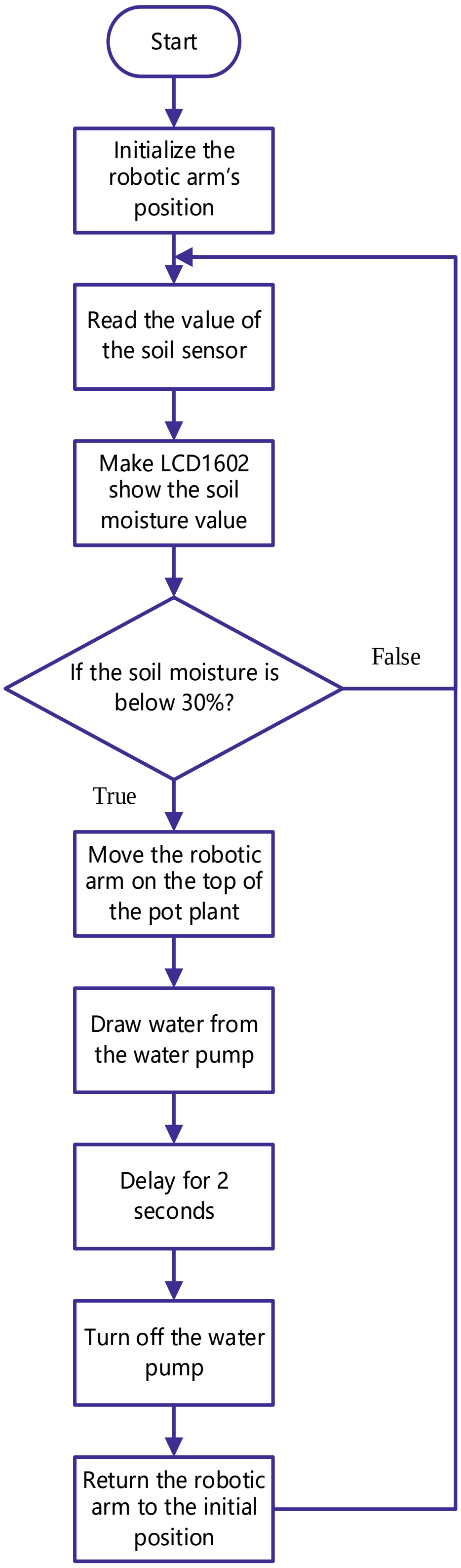

Start
Initialize the robotic arm's position
Read the value of the soil sensor
Make LCD1602 show the soil moisture value
If the soil moisture is below 30%?
False
True
Move the robotic arm on the top of the pot plant
Draw water from the water pump
Delay for 2 seconds
Turn off the water pump
Return the robotic arm to the initial position

Figure 4.17 Automatic watering flow chart

Step 4: Enable Arduino programming software. Import the library files of the robotic arm and LCD1602.

```
#include <Wire.h>
#include <LiquidCrystal_I2C.h>
#include <MagicianLite.h>
```

Step 5: Define the object and the variable.

```
LiquidCrystal_I2Clcd(0x20,16,2); //Define the LCD object
#define HumidityPin A11 //Define the name of the pin of the soil humidity sensor
Int HumidityValue; //Define the value variable of the soil humidity sensor
Int HumidityPercent;     //Define the percent variable of the value of the soil
humidity sensor
int Relay = 9;     //Define the pin of the relay
```

Step 6: Initialize all parameters.

```
Void setup()
{
pinMode(Relay,OUTPUT); //Set the port that controls the relay to output
MagicianLite_Init(); //Initialize the robotic arm
MagicianLite_SetPTPCmd(JUMP_XYZ,200,0,50,0);//Initialize the position of the
robotic arm
lcd.init();     //Initialize LCD
lcd.backlight();
}
```

Step 7: Obtain the soil humidity value, and display it on LCD1602. Based on the above task, independently encode the current step.

```
Void loop()
{
```

Step 8: Decide whether the soil humidity is lower than 30%. If yes, move the robotic arm to the position above the pot plant. From practical operation, obtain the coordinates of the robotic arms. The coordinates of the robotic arm in the program below is for reference.

```
if(HumidityPercent<=30)
{
MagicianLite_SetPTPCmd(JUMP_XYZ,14,-90,50,0);//Move the water pipe to the
position of the pot plant
delay(1000);
```

Step 9: Switch on the pump to draw water. Two seconds later, switch off it.

```
digitalWrite(Relay, HIGH);
delay(2000);
digitalWrite(Relay, LOW);
```

Step 10: Move the robotic arm to its original position.

```
MagicianLite_SetPTPCmd(JUMP_XYZ,200,0,50,0);//The robotic arm returns to
its original position
}
Delay (50) ; //Sampling interval of the temperature and humidity sensor</1639>
}
```

Step 11: Compile and upload the program.

3. Summary

The automatic watering steps:____
____
____
____

4. Self-Assessment

Check the completed content in the experiment task. Tick（√） the completed items, and circle (◯) the uncompleted items.

Assessment Content	Completion Status
I've connected the hardware of the automatic watering system	
I've programmed automatic watering	

Experiment 5

Intelligent Picking

Team Name: Team member: Date:

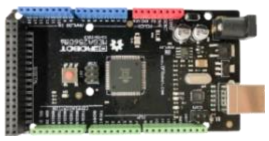
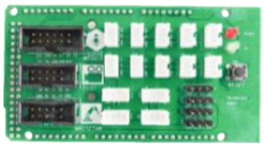
Overview

With the advancement of science and technology, agriculture has gradually realized mechanized production. From sowing and watering to automatic temperature control and automatic light control, mechanized production has become increasingly diverse. This experiment simulates the use of robotic arm picking instead of manual picking, thereby improving picking efficiency and reducing picking costs. Next we will pick one radish, one row of radishes, one field of and two fields of radishes.

Objective

- ➢ Skillfully use movement instructions by controlling the movement of the robotic arm.
- ➢ Learn the instructions corresponding to different states of the flexible gripper by controlling the flexible gripper to grasp and place objects.
- ➢ Learn the “while” statement by controlling the robotic arm, pick one radish.
- ➢ Review the "for” statement and derive the numerical change formula by controlling the robotic arm to pick one row of radishes.
- ➢ Review the loop structure by controlling the robotic arm to pick one field of and two fields of radishes.

Equipment

Equipment Picture	Name	Quantity
	Dobot Magician Lite Robotic arm	1
	Gripper kit	1
	Power adapter	1
	USB type-C interface cable	1
	Arduino Mega 2560 control board	1
	Arduino shield expansion board	1
	USB square port cable	1
	10Pin-DuPont head adapter cable	1
	Radish field	2

	Red and white radishes	Several
	basket	2

Requirements

- ➢ Take care when using electricity.
- ➢ Before the experiment, check whether the experiment equipment is complete and intact. If there is any omission or damage, please report to the teacher.
- ➢ Any specific operations in the experiment shall be performed according to the experiment manual. If you have any questions, please promptly ask the teacher.
- ➢ During the experiment, the joints will start to work as the robotic arms are powered on. In that case, do not move the joints of the robotic arms hard if you do not press the unlock key.
- ➢ Report any device fault during the experiment to your teacher in a timely manner, and do not handle it yourself.
- ➢ Arrange all devices after the experiment. You shall not leave the lab before check by the group leader.

Task 1: Pick One Radish

When radishes are ripe, people usually pick them manually or by machine. Then, how should we do so with the robotic arm? Next, we will first try picking one radish with the robotic arm.

5. Analysis

Analyze the methods and steps to pick one radish.

To pick one radish: Pick _____with the grippers of the robotic arm.

Steps to pick one radish:

First move the robotic arm to _______, pull _______ out; then move the robotic arm to _________, and place the radish on ________.

6. Steps

(1) Prepare Hardware

Step 1: Set the positions for picking one radish, as shown in Figure 5.1.

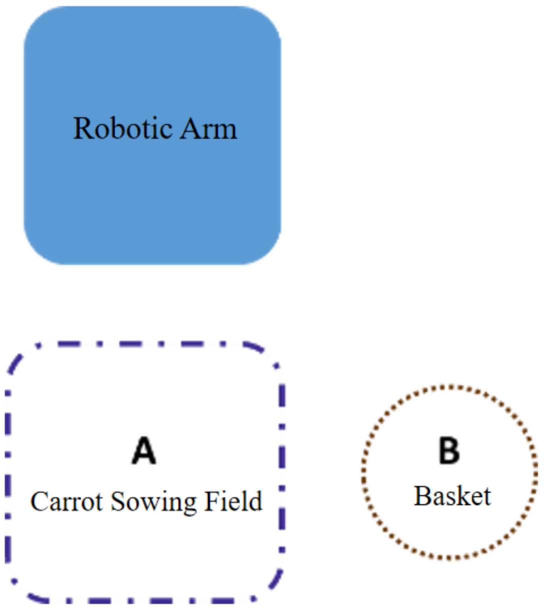


Figure 5.1 Placement position diagram

Step 2: Connect the equipment for picking one radish, as shown in Figure 5.2.

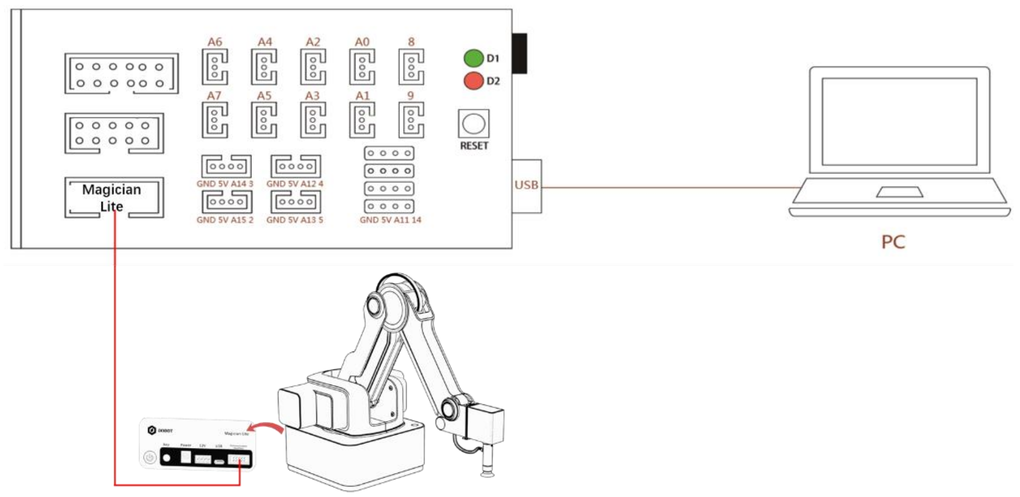


Figure 5.2 Connection diagram

Step 3: Prepare the experiment equipment ourselves, and connect it.

(2) Design Program

Step 1: Analyze how to pick one radish with the robotic arm. Then, read the flow chart, as shown in Figure 5.3.

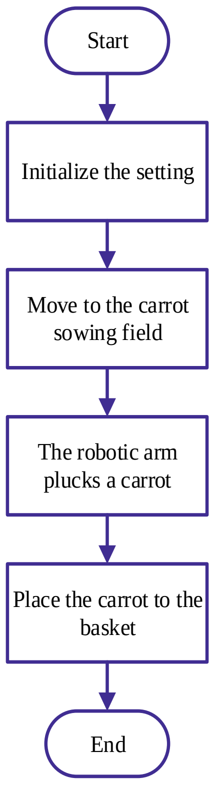

Start
Initialize the setting
Move to the carrot sowing field
The robotic arm plucks a carrot
Place the carrot to the basket
End

Figure 5.3 Experiment map

Step 2: Set the header file. At the beginning of the program, there is a header file, which contains the port definition and common functions of the robotic arm. The programming method is as follows.

```
#include<MagicianLite.h>// Set header file
```

Step 3: Initialize the settings. In the setup function, initialize the robotic arm, setting its movement ratio and the lifting height of the JUMP movement. The programming method is as follows.

```
void setup() {
   MagicianLite_Init();                //Initialize the robotic arm
MagicianLite_SetPTPCommonParams (80,80); // Set the robotic arm movement ratio
MagicianLite_SetPTPJumpParams (80); // Set the lifting height of the JUMP movement
}
```

Step 4: In the loop function, move the machine to the designated position of the radish field. The programming method is as follows.

```
void loop(){
   MagicianLite_SetPTPCmd (JUMP_XYZ, 260,0,15,0); // moved to the radish field
```

Step 5: Pull out the radish using the robotic arm. After the robot arm moves to the radish field, it starts to pick radishes. Here, use the flexible grippers to grab the radish, and set the state of the gripper to grab. Delay a while after grasping it. The programming method is as follows.

```
MagicianLite_SetEndEffectorGripper (true, true); // Grab the radish
   delay(200);
}
```

The control method of the flexible gripper at the end is similar to that of the suction cup, except that the programming statement of the control gripper contains two function parameters, and the programming method is as follows:

MagicianLite_SetEndEffectorGripper(boolisEnable,boolisGriped)

As the gripper contains three states, grabbing, opening, and closing, the return values of the two parameters are different, and the state of the gripper is different. The relationship between the gripper state and the parameter return value is as follows:

Gripper state	Grab	Open	Close
First parameter	true	true	false
Second parameter	true	false	false

Step 6: Put the radish in the basket.

(1) After the robotic arm pulls out the radish, JUMP moves above the basket position, and the coordinate values of the X, Y, and Z axes are 240, 200, and 0.

(2) Put the radish in the basket, keeping the gripper open.

(3) Delay for 200ms.

(4) Write the related program.

Step 7: Set the gripper to close.

After placing the radish, keep the grippers closed and write a program to set the grippers closed

Step 8: According to the analysis of the above steps, summarize the program as follows.

```
#include<MagicianLite.h>// Set header file
voidsetup() {
  MagicianLite_Init();//Initialize the robotic arm
MagicianLite_SetPTPCommonParams (80,80); // Set the robotic arm movement ratio
MagicianLite_SetPTPJumpParams (80); // Set the lifting height of the JUMP movement
}
void loop(){
  MagicianLite_SetPTPCmd (JUMP_XYZ, 260,0,15,0); // Move to the radish field
MagicianLite_SetEndEffectorGripper (true, true); // Grab the radish
  delay(200);
  MagicianLite_SetPTPCmd (JUMP_XYZ, 240,200,20,0); // JUMP moves upon the basket
  MagicianLite_SetEndEffectorGripper (true, false); // Place the radish
  delay(200);
  MagicianLite_SetEndEffectorGripper (false, false); // The gripper is in a natural state
  MagicianLite_SetHOMECmd (); // Robot returns to zero
}
```

Step 9: Compile the program and upload the program to Arduino control board, and observe how the robotic arm picks the radish.

Observation shows that the robot arm picks radishes more than once. Why?

This is because the entire picking program is in the loop function. In that case, the robotic arm will pick the radish repeatedly.

Step 10: Pick one radish. Above the setup function, define an integer variable *i* and set the initial value to 0 to calculate the number of executions of the robotic arm; in the loop function, place the entire picking program in the while loop. The judgment condition for the while loop is *i* <1. Upon picking completion, the variable *i* increases by 1. The programming method is as follows.

```
#include<MagicianLite.h>// Set the header file
inti = 0;
voidsetup() {
  MagicianLite_Init();//Initialize the robotic arm
MagicianLite_SetPTPCommonParams (80,80); // Set the robotic arm movement ratio
MagicianLite_SetPTPJumpParams (80); // Set the lifting height of the JUMP movement
}
void loop(){
  while(i < 1){
    MagicianLite_SetPTPCmd (JUMP_XYZ, 260,0,15,0); // Move to the radish field
MagicianLite_SetEndEffectorGripper (true, true); // Grab the radish
    delay(200);
    MagicianLite_SetPTPCmd (JUMP_XYZ, 240,200,20,0); // JUMP moves above the basket
    MagicianLite_SetEndEffectorGripper (true, false); // Place the radish
    delay(200);
    MagicianLite_SetEndEffectorGripper (false, false); // The gripper is in a natural state
    MagicianLite_SetHOMECmd (); // Robot returns to zero
    i++;
  }
}
```

Step 11: Compile the program, upload the program to Arduino control board, and observe how the robotic arm picks the radish.

7. Summary

(1) In this experiment, the role of the “while” statement is:

(2) The relationship between the state of the flexible gripper and the parameter return value

Gripper state	Grab	Open	Close
First parameter			
Second parameter			

8. Self-Assessment

Check the completed content in the experiment task. Tick（√） the completed items, and circle (◯) the uncompleted items.

Assessment Content	Completion Status
I’ve skillfully used the instructions to move the robotic arm	
I’ve used the “while” statement	
I’ve learned how to control the flexible gripper	
I’ve used different instructions to control the flexible gripper	
I’ve programmed to pick one radish	

Task 2: Pick One Row of Radishes

We are able to pick one radish with the flexible gripper of the robotic arm. Next, we will try picking one row of (3) radishes with the robotic arm.

1. Analysis

Analyze the main steps to pick one radish.

Pick the first radish: Move to the first radish position, pull out the radish, and place it in the basket.

Pick the second radish: ____

Pick the third radish: ____

2. Steps

(1) Prepare Hardware

Step 1: Set the positions for picking a row of radishes, as shown in Figure 5.4.

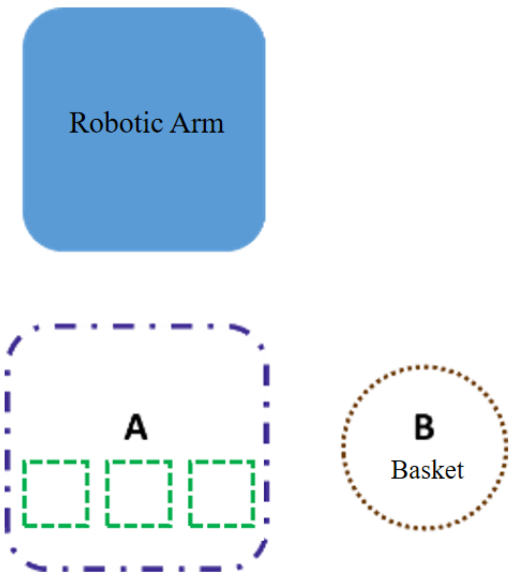


Figure 5.4 Placement location diagram

Step 2: Connect the hardware for picking one row of radishes in the same way in Task 1, as shown in Figure 5.2.

Step 3: Prepare the experiment equipment ourselves and connect the equipment.

(2) Design Program

Step 1: Analyze the process of picking one row of radishes, and read the flow chart, as shown in Figure 5.5.

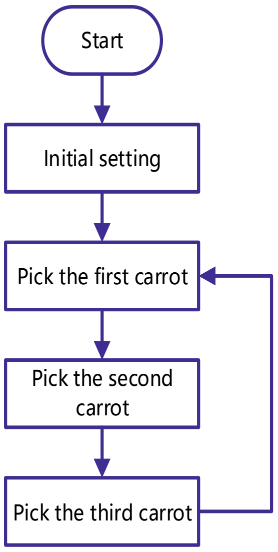


Figure 5.5 Flow chart of picking a row of radishes

Step 2: Set the header file. The programming method is as follows:

```
#include<MagicianLite.h>// Set header file
```

Step 3: Initialize the equipment. In the setup function, initialize the robotic arm, set the ratio of the movement speed and acceleration of the movement of the robotic arm, and set the lifting height of the JUMP movement. The programming method is as follows.

```
Void setup() {
MagicianLite_Init();//Initialize the robotic arm
MagicianLite_SetPTPCommonParams (80,80); // Set the robotic arm movement ratio
MagicianLite_SetPTPJumpParams (80); // Set the lifting height of the JUMP movement
}
```

Step 4: In the loop function, allow the robotic arm to JUMP above the initial position of the radish field. The programming method is as follows.

```
void loop() {
MagicianLite_SetPTPCmd (JUMP_XYZ, 220, -60,100,0); // JUMP moves to the
initial position of the radish planting land
```

Step 5: Pick one row of radishes. According to Task 1, program the robotic arm to pick a row of radishes as follows:

```
MagicianLite_SetPTPCmd (JUMP_XYZ, 220, -60,20,0); // First radish position
MagicianLite_SetEndEffectorGripper (true, true); // Grab the radish
delay(200);
MagicianLite_SetPTPCmd (JUMP_XYZ, 240,200,30,0); // JUMP moves above the basket
MagicianLite_SetEndEffectorGripper (true, false); // Place the radish
delay(200);
        MagicianLite_SetEndEffectorGripper (false, false); // The gripper is in a natural state

MagicianLite_SetPTPCmd (JUMP_XYZ, 220,0,20,0); // The second radish position
MagicianLite_SetEndEffectorGripper (true, true); // Grab the radish
delay(200);
MagicianLite_SetPTPCmd (JUMP_XYZ, 240,200,30,0); // JUMP moves to the position above the
basket
MagicianLite_SetEndEffectorGripper (true, false); // Place the radish
delay(200);
        MagicianLite_SetEndEffectorGripper (false, false); // The gripper is in a natural state

MagicianLite_SetPTPCmd (JUMP_XYZ, 220,60,20,0); // The third radish position
MagicianLite_SetEndEffectorGripper (true, true); // Grab the radish
delay(200);
MagicianLite_SetPTPCmd (JUMP_XYZ, 240,200,30,0); // JUMP moves to the position above the
```

```
basket
MagicianLite_SetEndEffectorGripper (true, false); // Place the radish
delay(200);
        MagicianLite_SetEndEffectorGripper (false, false); // The gripper is in a natural state
```

According to the above program, we find that when the robotic arm picks the first, second, and third radishes, they have different initial positions.

Since there are almost same steps and program for picking a row of radishes, we can use a loop structure. Here we use the "for" statement.

Step 6: Pick a row of radishes when the judgment condition (a <3) is satisfied in the "for" loop.

See the coordinate axis in Figure 5.6. When the robotic arm picks from left to right, the X axis remains unchanged, and the Y axis coordinate value gradually increases. Describe the change in the Y-axis coordinate during the picking process. Try writing a formula for the change in the Y-axis coordinate (note that the formula change is based on the actual picking spacing).

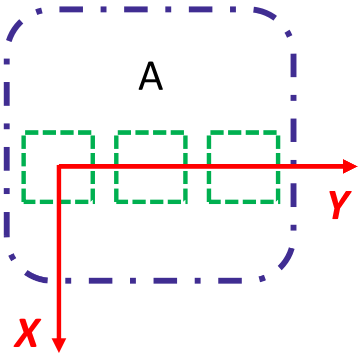


Figure 5.6 Coordinate axes

The programming method for picking one row of radishes is as follows:

```
for(int a = 0; a<3; a +=1){
MagicianLite_SetPTPCmd (JUMP_XYZ, 220, -60 + a * 60,20,0); // Radish spacing 60mm
MagicianLite_SetEndEffectorGripper (true, true); // Grab the radish
delay(200);
MagicianLite_SetPTPCmd (JUMP_XYZ, 240,200,30,0); // JUMP moves to the position above the
basket
MagicianLite_SetEndEffectorGripper (true, false); // Place the radish
delay(200);
MagicianLite_SetEndEffectorGripper (false, false); // The gripper is in a natural state
  }
```

Step 7: According to the analysis of the above steps, summarize the program for picking one row of radishes as follows:

```
#include<MagicianLite.h>// Set header file
void setup() {
MagicianLite_Init();//Initialize the robotic arm
MagicianLite_SetPTPCommonParams (80,80); // Set the robotic arm motivement ratio
MagicianLite_SetPTPJumpParams (80); // Set the lifting height of the JUMP move
}
void loop() {
MagicianLite_SetPTPCmd (JUMP_XYZ, 220, -60,100,0); // JUMP moves to the initial position
of the radish field
 for(int a = 0; a<3; a +=1){
MagicianLite_SetPTPCmd (JUMP_XYZ, 220, -60 + a * 60,20,0); // Radish spacing 60mm
MagicianLite_SetEndEffectorGripper (true, true); // Grab the radish
delay(200);
MagicianLite_SetPTPCmd (JUMP_XYZ, 240,200,30,0); // JUMP moves to the position above the
basket
MagicianLite_SetEndEffectorGripper (true, false); // Place the radish
delay(200);
MagicianLite_SetEndEffectorGripper (false, false); // The gripper is in a natural state
  }
MagicianLite_SetHOMECmd (); // Robotic arm returns to zero
}
```

Step 8: Compile the program, upload it to Arduino control board, and observe how the robotic arm executes the picking task.

3. Summary

(1) In this experiment, the role of the “for” statement is: ____________
____

(2) Changes in the Y-axis coordinates when the robot arm picks a row of radishes____
____

Try writing the formula for change in the Y-axis coordinate ___________

4. Self-Assessment

Check the completed content in the experiment task. Tick（√） the completed items, and circle (◯) the uncompleted items.

Assessment Content	Completion Status
I’ve skillfully used the instructions to control the flexible gripper	
I’ve learned to derive the formula for change in the Y-axis coordinate	
I’ve used the “for” statement	
I’ve programmed to pick one row of radishes	

Task 3: Pick One Field of Radishes

We are able to pick one row (3) of radishes with the flexible gripper of the robotic arm. Next, we will try picking one field of radishes (three rows and three columns) with the robotic arm.

1. Analysis

Analyze the main steps to pick one field of radishes.

Pick the first row of radishes:

Pick the second row of radishes:

Pick the third row of radishes:

2. Steps

(1) Prepare Hardware

Step 1: Set the positions for picking one field of radishes, as shown in Figure 5.7.

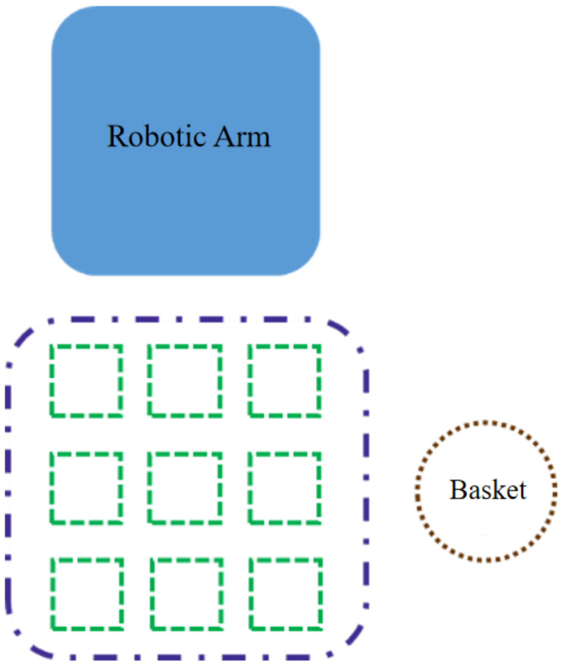


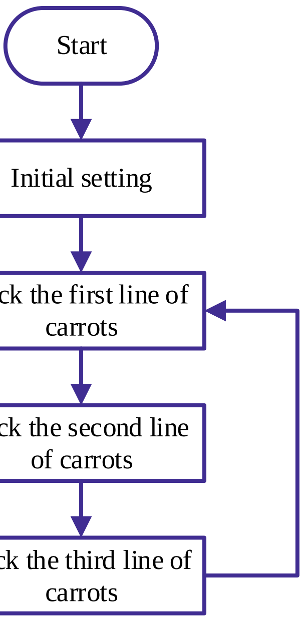
Figure 5.7 Placement location diagram

Step 2: Connect the equipment for picking one field of piece of radishes in the same way in Task one, as shown in Figure 5.2.

Step 3: Prepare the experiment equipment ourselves and connect it.

(2) Design Program

Step 1: Analyze how the robotic arm picks one field of radishes, and read the flow chart, as shown in Figure 5.8.


Start
Initial setting
Pick the first line of carrots
Pick the second line of carrots
Pick the third line of carrots
Figure 5.8 Implementation flow chart

Step 2: Observe the characteristics of the radish field. We know that radishes are grown in three rows and three columns. When picking radishes, we can first pick one row, then the next, and so on.

Review the formula for change in the Y-axis coordinate picking one row of radishes in Task 2.

In Arduino, write a program to pick one row of radishes.

Step 3: Pick one field of radishes. According to Task 2, program the robotic arm to picking one field of radishes as follows:

```
for (int a = 0; a <3; a + = 1) {// picking the first row of radishes
MagicianLite_SetPTPCmd (JUMP_XYZ, 220, -60 + a * 60,20,0); // Radish spacing 60mm
MagicianLite_SetEndEffectorGripper (true, true); // Grab the radish
delay(200);
MagicianLite_SetPTPCmd (JUMP_XYZ, 240,200,30,0); // Move above the basket
MagicianLite_SetEndEffectorGripper (true, false); // Place the radish
delay(200);
MagicianLite_SetEndEffectorGripper (false, false); // The gripper is in a natural state
  }
MagicianLite_SetHOMECmd (); // Robotic arm returns to zero

```

```
for (int a = 0; a <3; a + = 1) {// Pick the second row of radishes
MagicianLite_SetPTPCmd (JUMP_XYZ, 340, -60 + a * 60,20,0); // Radish spacing 60mm
MagicianLite_SetEndEffectorGripper (true, true); // Grab the radish
delay(200);
MagicianLite_SetPTPCmd (JUMP_XYZ, 240,200,30,0); // Move above the basket
MagicianLite_SetEndEffectorGripper (true, false); // Place the radish
delay(200);
MagicianLite_SetEndEffectorGripper (false, false); // The gripper is in a natural state
  }
MagicianLite_SetHOMECmd (); // Robotic arm returns to zero

for (int a = 0; a <3; a + = 1) {// Pick the third row of radishes
MagicianLite_SetPTPCmd (JUMP_XYZ, 220, -60 + a * 60,20,0); // Radish spacing 60mm
MagicianLite_SetEndEffectorGripper (true, true); // Grab the radish
delay(200);
MagicianLite_SetPTPCmd (JUMP_XYZ, 240,200,30,0); // Move above the basket
MagicianLite_SetEndEffectorGripper (true, false); // Place the radish
delay(200);
MagicianLite_SetEndEffectorGripper (false, false); // The gripper is in a natural state
  }
MagicianLite_SetHOMECmd (); // Robot returns to zero
```

According to the above program, we find that when the robotic arm picks the first, second, and third rows of radishes, the three rows have different initial positions.

Since there are almost same steps and program for picking a row of radishes, we can use a nested loop structure. Here we use the "for" statement again.

Loop nesting is a common method in logic programs. The inclusion of another loop statement in a loop body statement is called loop nesting.

Step 4: Observe the change in the coordinates when the robotic arm picks three rows of radishes. According to Figure 5.9, neatly place the three rows of radishes to be picked in parallel in front of the robotic arm.

When the robotic arm picks each row of radishes from back to front, the corresponding Y-axis and X-axis coordinate values gradually increase.

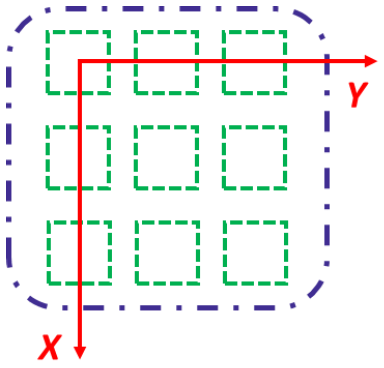


Figure 5.9 Coordinate axes

Describe the change in the X-axis coordinate when the robotic arm picks three rows of radishes.

Try writing the X-axis coordinate change formula (note that the formula change is based on the actual picking spacing).

When the robotic arm picks one field of radishes (three rows and three columns), the programming method is as follows:

```
 for(int b = 0; b<3; b +=1){
for(int a = 0; a<3; a +=1){
MagicianLite_SetPTPCmd (JUMP_XYZ, 220 + b * 60,60 + a * 60,20,0); // Radish spacing is 60mm
MagicianLite_SetEndEffectorGripper (true, true); // Grab the radish
delay(200);
MagicianLite_SetPTPCmd (JUMP_XYZ, 240,200,30,0); // Move above the basket
MagicianLite_SetEndEffectorGripper (true, false); // Place the radish
delay(200);
MagicianLite_SetEndEffectorGripper (false, false); // The gripper is in a natural state
}
  }
MagicianLite_SetHOMECmd (); // Robotic arm returns to zero
}
```

Step 5: According to the analysis of the above steps, summarize the program for picking one row of radishes is as follows:

```
#include<MagicianLite.h>// Set header file
void setup() {
MagicianLite_Init();//Initialize the robotic arm
MagicianLite_SetPTPCommonParams (80,80); // Set the robotic arm motivement ratio
MagicianLite_SetPTPJumpParams (80); // Set the lifting height of the JUMP movement
}
void loop() {
MagicianLite_SetPTPCmd (JUMP_XYZ, 220, -60,100,0); // JUMP moves to the initial position
of the radish field
  for(int b = 0; b<3; b +=1){
for(int a = 0; a<3; a +=1){
MagicianLite_SetPTPCmd (JUMP_XYZ, 220 + b * 60,60 + a * 60,20,0); // Radish spacing is
        60mm
MagicianLite_SetEndEffectorGripper (true, true); // Grab the radish
delay(200);
MagicianLite_SetPTPCmd (JUMP_XYZ, 240,200,30,0); // moved above the basket
MagicianLite_SetEndEffectorGripper (true, false); // Place the radish
delay(200);
MagicianLite_SetEndEffectorGripper (false, false); // The gripper is in a natural state
}
   }
MagicianLite_SetHOMECmd (); // Robotic arm returns to zero
}
```

Step 5: Compile the program, upload it to Arduino control board, and observe how the robotic arm picks radishes.

3. Summary

(1) In this experiment, the role of the “for” statement is: ____________

(2) Change in the X-axis coordinate when the robotic arm picks three rows of radishes ______________________________

Try writing the formula for change in the X-axis coordinate ___________

4. Self-Assessment

Check the completed content in the experiment task. Tick（√） the completed items, and circle (◯) the uncompleted items.

Assessment Content	Completion Status
I've learned to derive the formula for change in the X-axis and Y-axis coordinates	
I've known about nested loops and their role	
I've used the "for" statement multiple times	
I've programmed to pick one field of radishes	

Task 4: Pick Two Fields of Radishes

We are able to pick one field of radishes (three rows and three columns) with the flexible gripper of the robotic arm. Now, we will try picking two fields of radishes with the robotic arm.

1. Analysis

Analyze the main steps to pick two fields of radishes.

Pick the first field of radishes:

Pick the second field of radishes:

2. Steps

(1) Prepare Hardware

Step 1: Set the positions for picking two fields of radishes, as shown in Figure 5.10.

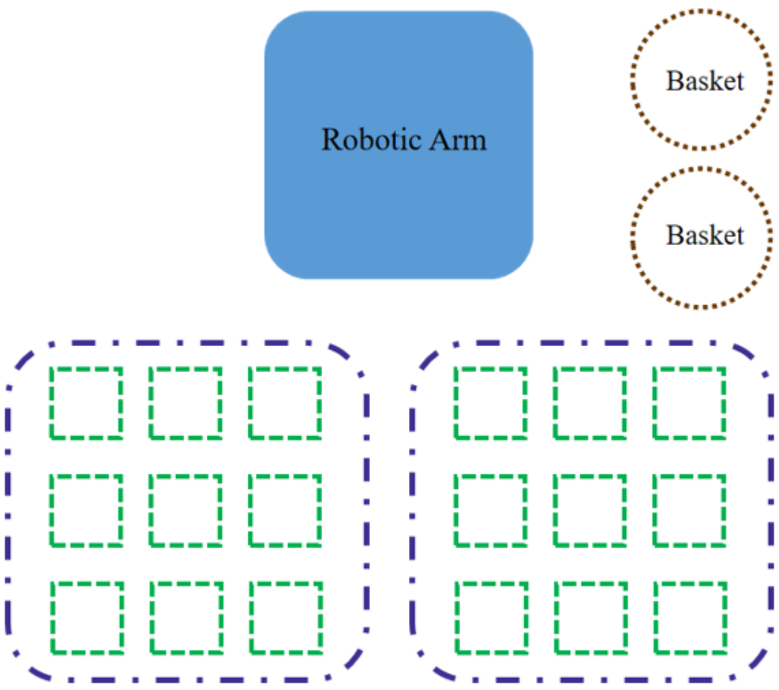


Figure 5.10 Placement location diagram

Step 2: Connect the equipment for picking two fields of radishes in the same way as in Task 1, as shown in Figure 5.2.

Step 3: Prepare the experiment equipment ourselves and connect it.

(2) Design Program

Complete this task yourselves. See Figure 5.11 for the flow chart of picking two fields of radishes.

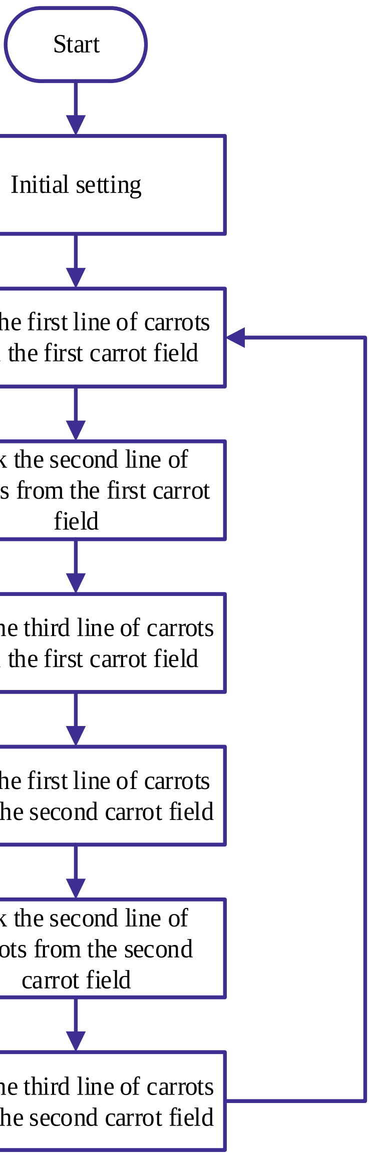

Start
Initial setting
Pick the first line of carrots from the first carrot field
Pick the second line of carrots from the first carrot field
Pick the third line of carrots from the first carrot field
Pick the first line of carrots from the second carrot field
Pick the second line of carrots from the second carrot field
Pick the third line of carrots from the second carrot field

Figure 5.11 Flow chart of picking two fields of radishes

3. Summary

(1) Change in the Y-axis coordinate when the robotic arm picks a row of radishes____

____

Try writing the formula for change in the Y-axis coordinate __________

(2) Change in the X-axis coordinate when the robotic arm picks three rows of radishes ____________________________________

____

Try writing the formula for change in the X-axis coordinate __________

4. Self-Assessment

Check the completed content in the experiment task. Tick（√） the completed items, and circle (◯) the uncompleted items.

Assessment Content	Completion Status
I’ve skillfully derived formulas for changes in the X-axis and Y-axis coordinates	
I’ve used the nested loop	
I’ve used the “for” statement multiple times	
I’ve programmed to pick two fields of radishes	

Experiment 6

Automatic Sorting

Team name: Team member: Date:

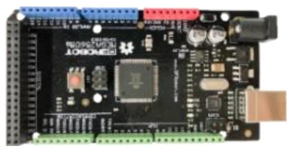
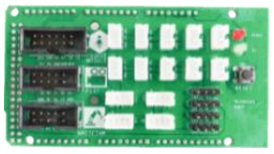
Overview

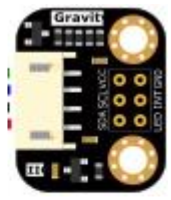
Because of the better weather, the farm's radishes have a good harvest. Seeing baskets of white and red radishes, everyone feels happy but troubled. When so many radishes are picked, they are not separated. To manually sort them, you will take long time and cost. Can you help find a solution? The solution proposed in this experiment is to simulate the radish sorting in the laboratory. First, build white radish and red radish models. Then build a radish basket model to hold the sorted radishes, and use the color sensor to print the red and white RGB values. Finally, compare the RGB values to complete the sorting task.

Objective

1. Learn the drawing function in Tinkercad by designing radish leaves.
2. Master the method of basic shape-based modeling by designing radish roots.
3. Master the programming modeling skills in Tinkercad by learning the array function in Tinkercad.
4. Understand how to control a color sensor by controlling the color sensor.
5. Master the print () function by displaying the RGB value through the serial port.
6. By programming the robotic arm to sort the radishes, master how to use the color sensor to determine colors, and how to use the robotic arm to grab radishes.

Equipment

Equipment Picture	Name	Quantity
	Dobot Magician Lite Robotic arm	1
	Gripper kit	1
	USB Type-C interface cable	1
	Power adapter	1
	Arduino Mega 2560 Control board	1
	Arduino shield expansion board	1
	USB square port cable (Type-B cable)	1
	10Pin-DuPont adapter cable	1

Equipment Picture	Name	Quantity
	Color sensor	1
	1Pin-DuPont Line (with male & female connector)	Some
	Basket	2
	Radish	Some
	3D printer	1
	3D printing material PLA	4

Requirements

- Take care when using electricity.
- Before the experiment, check whether the experiment equipment is complete and intact. If there is any omission or damage, please report to the teacher.
- Any specific operations in the experiment shall be performed according to the experiment manual. If you have any questions, please promptly ask the teacher.

- During the experiment, the joints will start to work as the robotic arms are powered on. In that case, do not move the joints of the robotic arms hard if you do not press the unlock key.
- Report any device fault during the experiment to your teacher in a timely manner, and do not handle it yourself.
- Arrange all devices after the experiment. You shall not leave the lab before check by the group leader.

Task 1: Design the Radish Model

To simulate the scenario of radish sorting, we first need radishes. Use Tinkercad to design the radish model.

1. Analysis

Analyze the steps to design the radish model based on the shape and structure of the radish. Then, complete the design steps in the blanks in the table below.

Step 1: Analyze the shape of radish, which can be divided into upper part the leaves and lower part the roots.

Step 2: Select the drawing function to draw the radish leaves.

Step 3: Select shape, design.

(Hint: how to design the root of the lower part of the radish)

Step 4:.

(Hint: Radish is a whole, and leaves and roots cannot be separated)

2. Steps

Step 1: Create a new 3D design project, as shown in Figure 6.1.

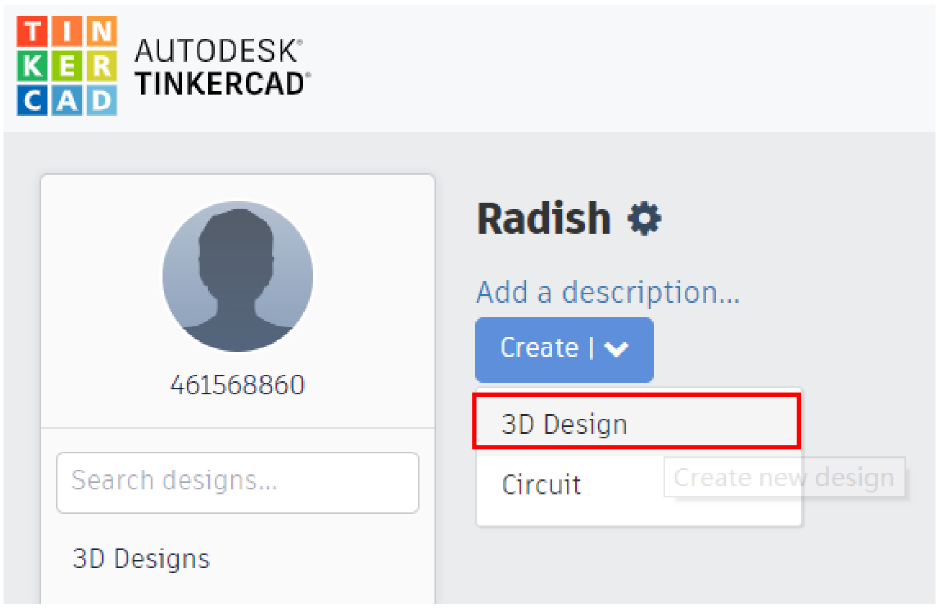


Figure 6.1 Creating the project

Step 2: Select **Scribble** in the basic shapes, as shown in Figure 6.2.

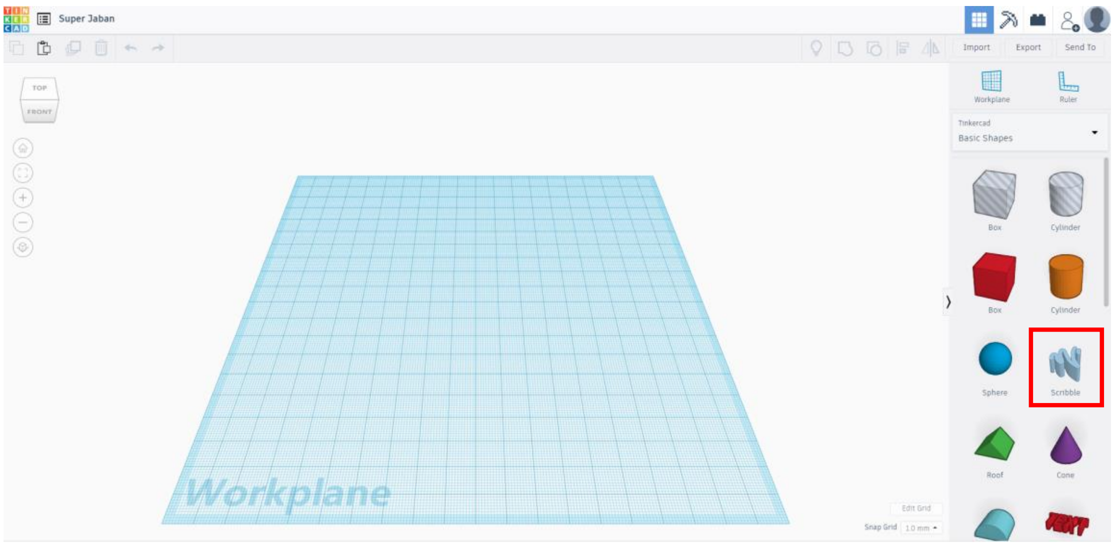


Figure 6.2 Choosing **Scribble**

Step 3: Draw the leaves of the upper half of the radish, as shown in Figure 6.3.

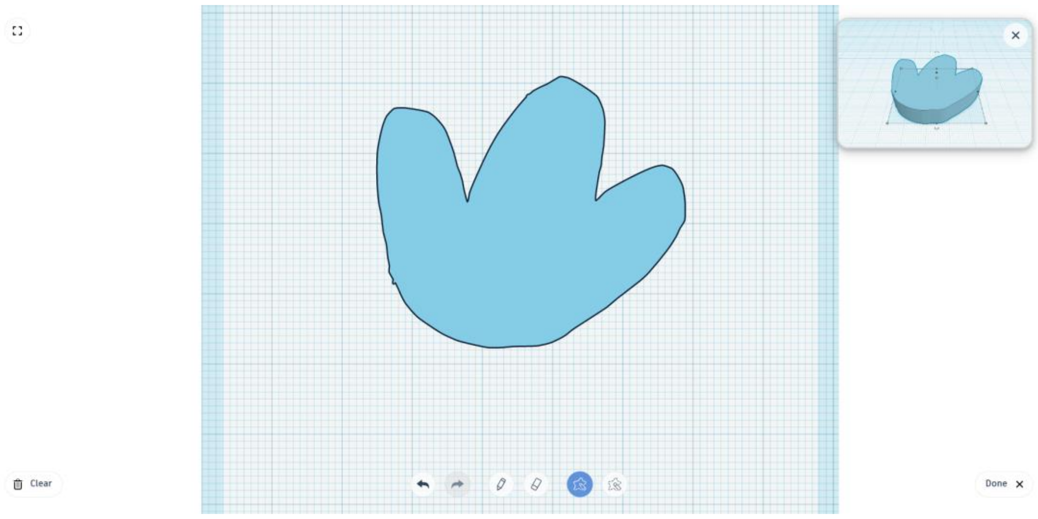


Figure 6.3 Leaf of the radish

Step 4: Set the height of the upper part of the radish to 8 mm, and the length and width to about 20 mm and 18 mm, as shown in Figure 6.4.

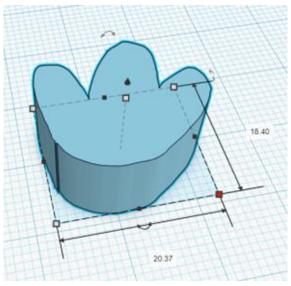


Figure 6.4 Setting the length, width, and height

Step 5: Analyze the root of the radish to be long-horned, cylindrical, select **Sphere**, set the height to 18 mm, the length and width to 20 mm and 45 mm, as shown in Figure 6.5.

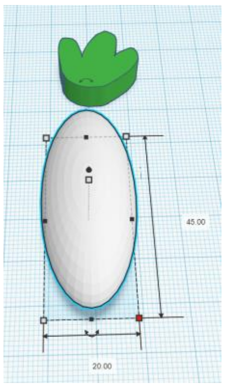
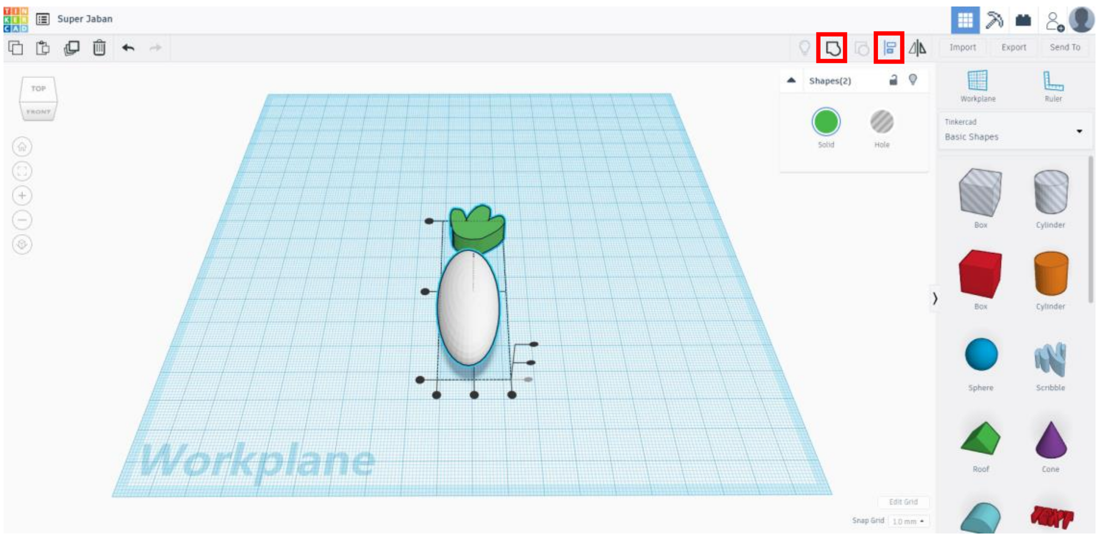
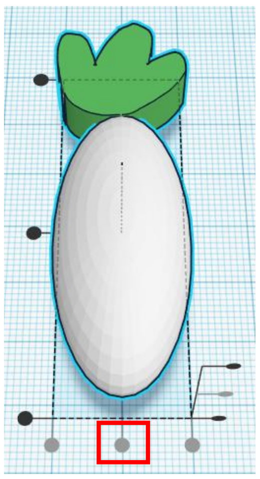
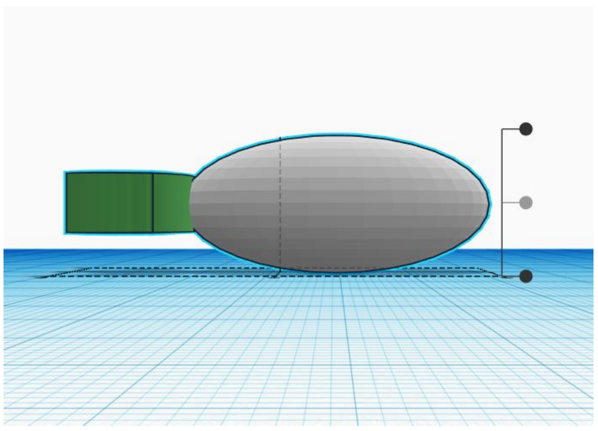


Figure 6.5 Setting the length, width, and height of the sphere

Step 6: Move the leaves and roots of the radish to appropriate positions, select one part, hold down the **Shift** key to select another part, and then click **Align**. The alignment method is shown in Figure 6.6.







Default vision

Left view

Figure 6.6 Alignment

Step 7: After aligning, click **Group** to complete the radish modeling. Export the model and slice it for 3D printing.

Design the radish size based on the farmland. You can set the size according to the actual situation.

3. Summary

(1) The structure of the radish can be divided into: ________________

The function of the designed radish leaf is: ______________________ ,

The shape of the radish root is: ________________________________

(2) In this test task, the grouping function is: _____________________

____

4. Self-Assessment

Check the completed content in the experiment task. Tick（√） the completed items, and circle (◯) the uncompleted items.

Assessment Content	Completion Status
I've observed the structure of the radish	
I've used the drawing function	
I've designed the radish leaf	
I've known the shape of the radish root	
I've designed the radish model	

Task 2: Design the Radish Basket Model

After designing the radish model, we also need to design a radish basket for sorted radishes. The basket size needs to match the radish model, so the size requirements are high. We can use Tinkercad to complete the modeling task and design more accurate size. Let's try building a model through programming.

1. Analysis

Ask the teacher to show the printed radish basket model. Observe the radish basket model, analyze the steps to design the basket, and complete the steps in the blanks in the table below.

Step 1: Measure the designed radish model and calculate its size.

Step 2: Select the ring as the top edge of the basket and the cone as the main body.

Step 3: Create a new hollow cone and use _______ to turn the solid basket into ______. (Hint: fill in the function of Tinkercad and the shape of the basket)

Step 4: Design the pattern module of the basket.

Step 5: Pattern module, arrange the pattern around the basket. (Fill in the function of Tinkercad)

Step 6: __________basket and pattern module. (Fill in the function of Tinkercad)

2. Steps

The first step: Design the size. Measure the size of the radish model, and calculate the appropriate size of the basket based on the actual radish to basket size ratio.

In addition to allowing a basket to hold as many radishes as possible, when designing the ratio of radish to basket you have to consider the weight and volume for transport convenience, and achieve the best ratio.

Step 2: Log in to your Tinkercad account and select **Codeblocks** to create a new code block, as shown in Figure 6.7.

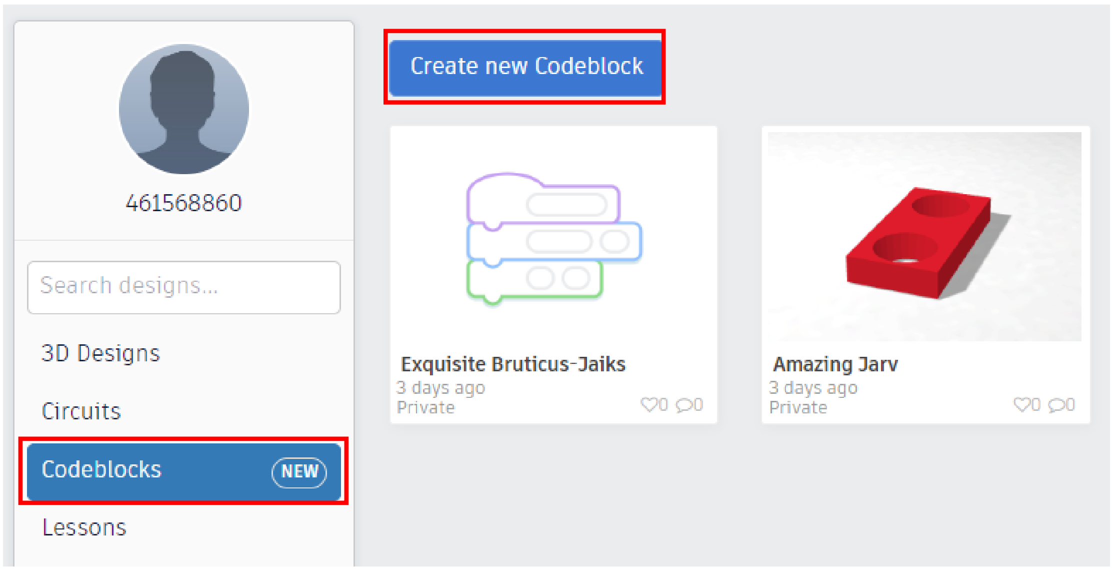


Figure 6.7 Creating a new code block

Step 3: Read the radish basket modeling flow chart based on the experiment analysis, as shown in Figure 6.8.

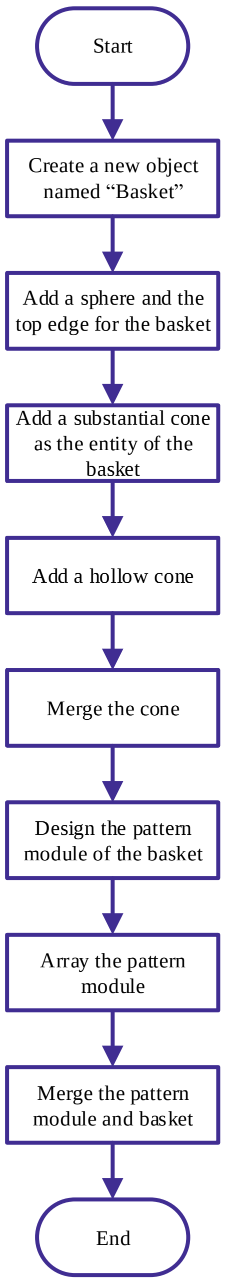

Start
Create a new object named "Basket"
Add a sphere and the top edge for the basket
Add a substantial cone as the entity of the basket
Add a hollow cone
Merge the cone
Design the pattern module of the basket
Array the pattern module
Merge the pattern module and basket
End

Figure 6.8 Radish basket modeling flow chart

Step 4: Create a new object, click the **Modify** tab, and drag out the "create new object object10" block, and create a new name "basket", as shown in Figure 6.9.

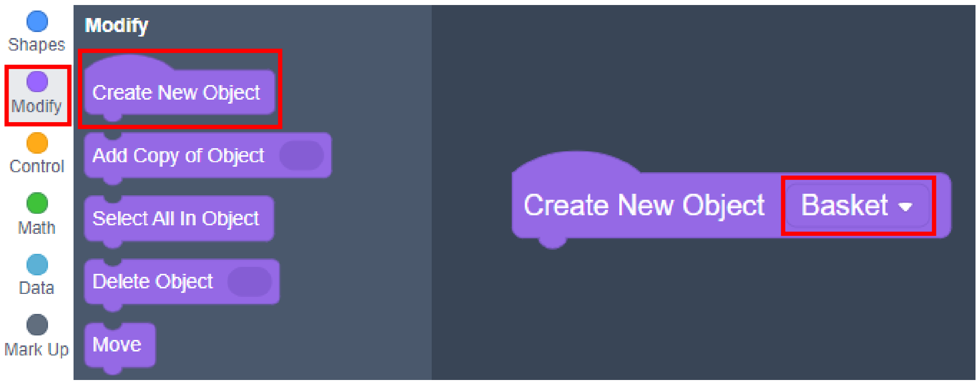


Figure 6.9 Creating the new object

Step 5: Add the torus as the top edge of the basket, click the **Shapes** tab, drag out the "torus" block, select solid, and fill in the parameter radius of 45 mm, as shown in Figure 6.10.

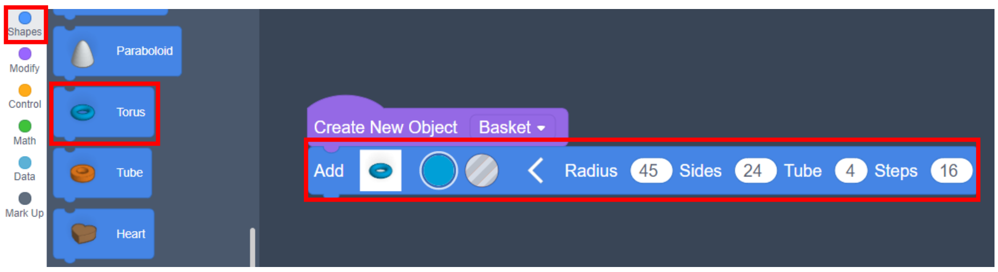


Figure 6.10 Adding the torus

Step 6: Move the torus to the coordinate point (0,0,30), as shown in Figure 6.11.

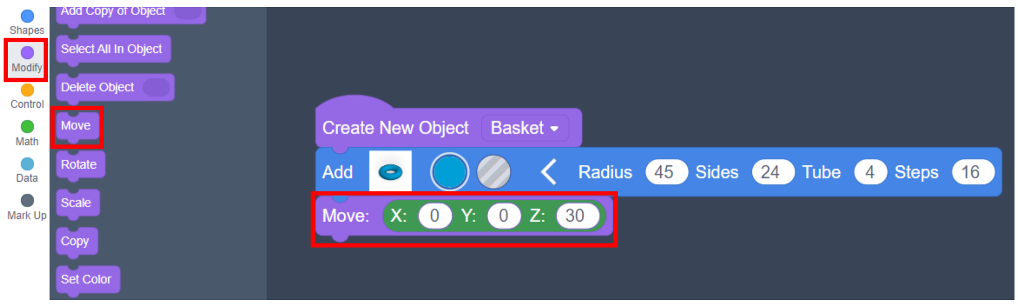
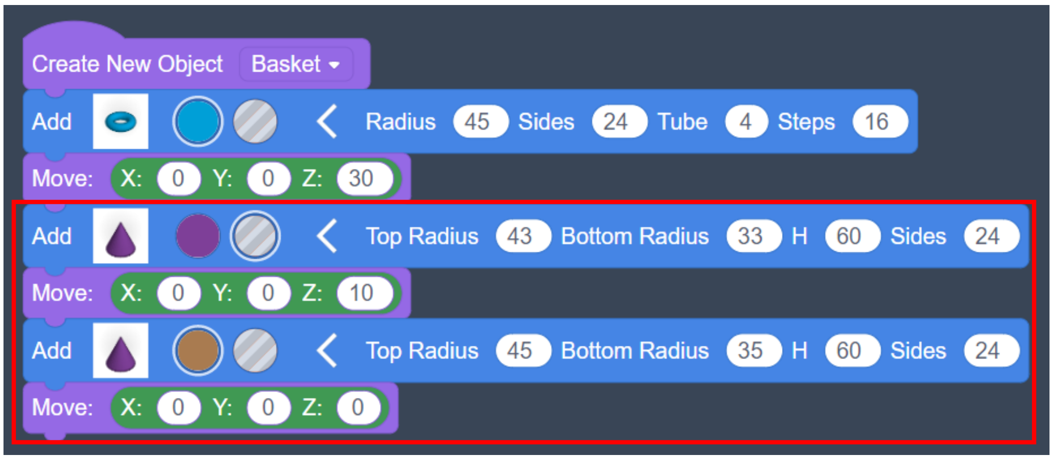


Figure 6.11 Moving

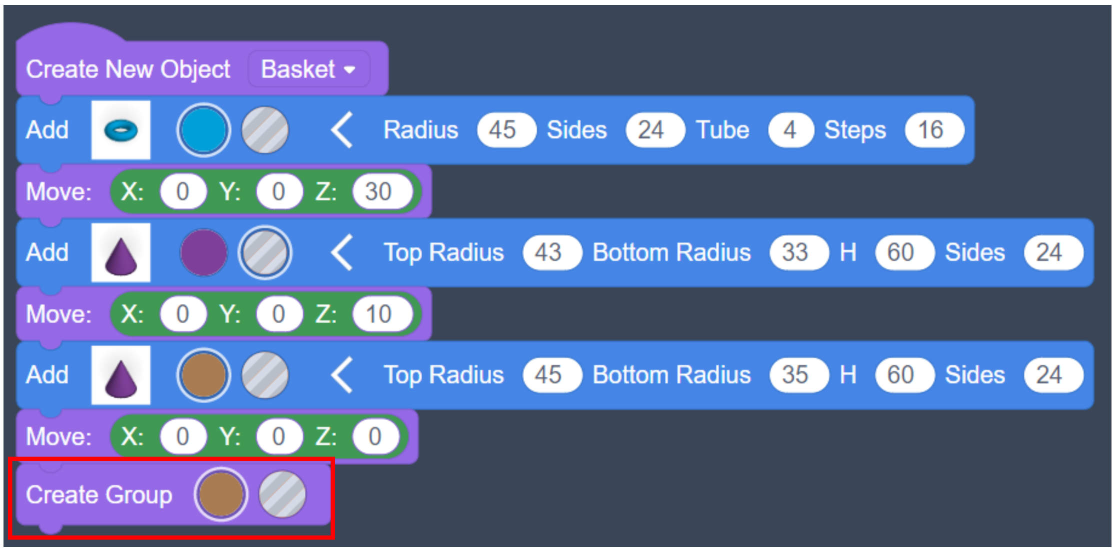
Step 7: Add two cones as the main body of the basket. The first cone is hollow, with the top radius of 43 mm, the bottom radius of 33 mm, and the height (H) of 60 mm. Move the first cone to the position (0, 0, 10). The second cone is solid, with the top radius of 45 mm, the bottom radius of 35 mm, and the height (H) of 60 mm. Move the second cone to the position (0, 0, 0), as shown in Figure 6.12.



Create New Object Basket
Add Radius 45 Sides 24 Tube 4 Steps 16
Move: X: 0 Y: 0 Z: 30
Add Top Radius 43 Bottom Radius 33 H 60 Sides 24
Move: X: 0 Y: 0 Z: 10
Add Top Radius 45 Bottom Radius 35 H 60 Sides 24
Move: X: 0 Y: 0 Z: 0

Figure 6.12 Adding two cones

Step 8: Create a group and select solid, as shown in Figure 6.13.

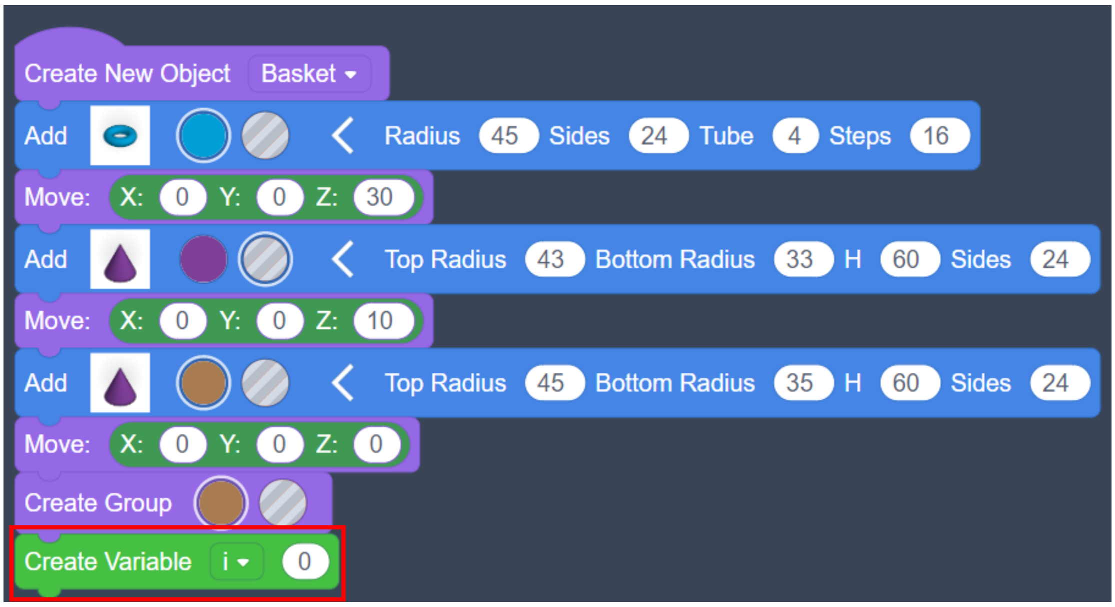


Create New Object Basket
Add Radius 45 Sides 24 Tube 4 Steps 16
Move: X: 0 Y: 0 Z: 30
Add Top Radius 43 Bottom Radius 33 H 60 Sides 24
Move: X: 0 Y: 0 Z: 10
Add Top Radius 45 Bottom Radius 35 H 60 Sides 24
Move: X: 0 Y: 0 Z: 0
Create Group

Figure 6.13 Creating the group (solid)

Calculate the basket size based on the size of the hollow cone and the solid cone.

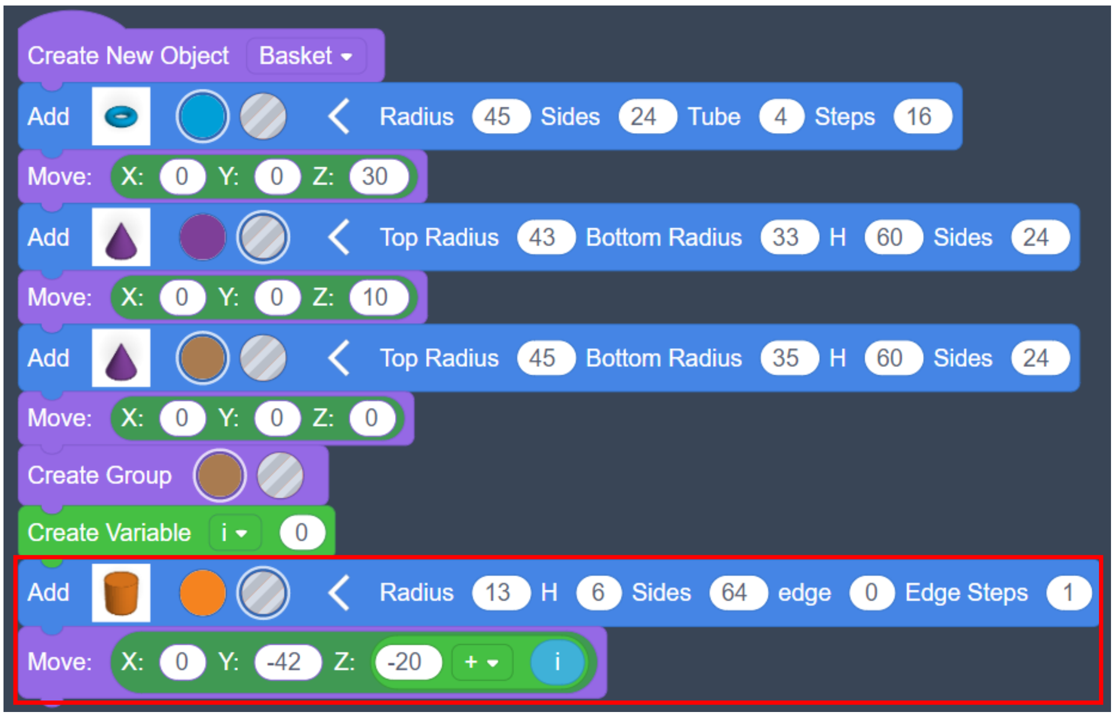
Step 9: Design the pattern module. Create the position of the variable control module and set the initial value of the variable to "0", as shown in Figure 6.14.



```
Create New Object  Basket ▾
Add  ◯ ◯  <  Radius 45  Sides 24  Tube 4  Steps 16
Move:  X: 0  Y: 0  Z: 30
Add  ◯ ◯  <  Top Radius 43  Bottom Radius 33  H 60  Sides 24
Move:  X: 0  Y: 0  Z: 10
Add  ◯ ◯  <  Top Radius 45  Bottom Radius 35  H 60  Sides 24
Move:  X: 0  Y: 0  Z: 0
Create Group  ◯ ◯
Create Variable  i ▾  0
```

Figure 6.14 Creating the variable

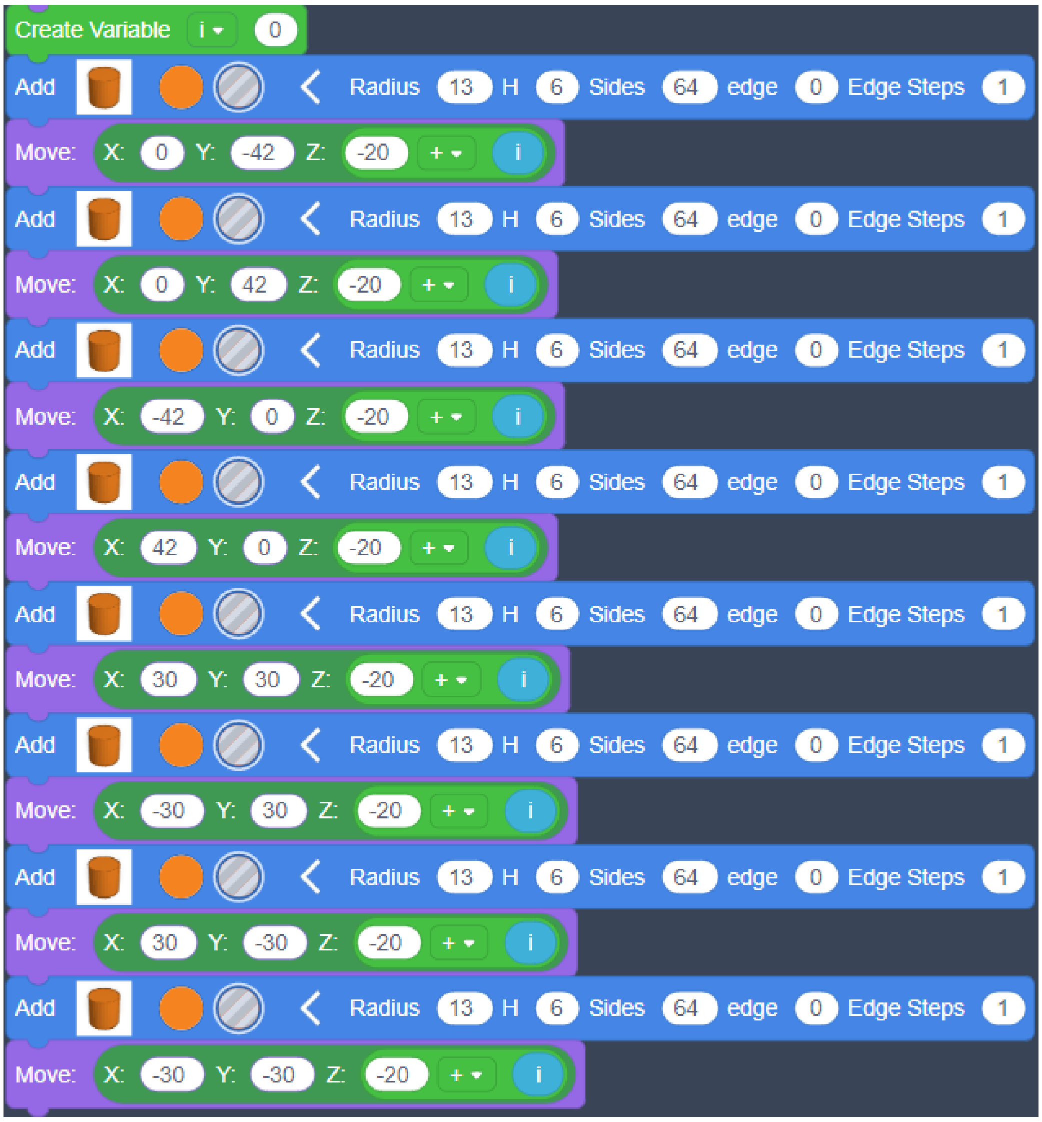
Step 10: Add a cylinder, set it as hollow, with the radius of 13 mm and the height (H) of 6 mm, and move the cylinder to the position (X: 0, Y: -42, Z: -20 + i), as shown in the figure 6.15.



```
Create New Object  Basket ▾
Add  ◯ ◯  <  Radius 45  Sides 24  Tube 4  Steps 16
Move:  X: 0  Y: 0  Z: 30
Add  ◯ ◯  <  Top Radius 43  Bottom Radius 33  H 60  Sides 24
Move:  X: 0  Y: 0  Z: 10
Add  ◯ ◯  <  Top Radius 45  Bottom Radius 35  H 60  Sides 24
Move:  X: 0  Y: 0  Z: 0
Create Group  ◯ ◯
Create Variable  i ▾  0
Add  ◯ ◯  <  Radius 13  H 6  Sides 64  edge 0  Edge Steps 1
Move:  X: 0  Y: -42  Z: -20 + ▾ i
```

Figure 6.15 Adding the cylinder and setting its movement coordinates

Step 11: Add seven more cylinders and set the movement coordinates of each cylinder, as shown in Figure 6.16.

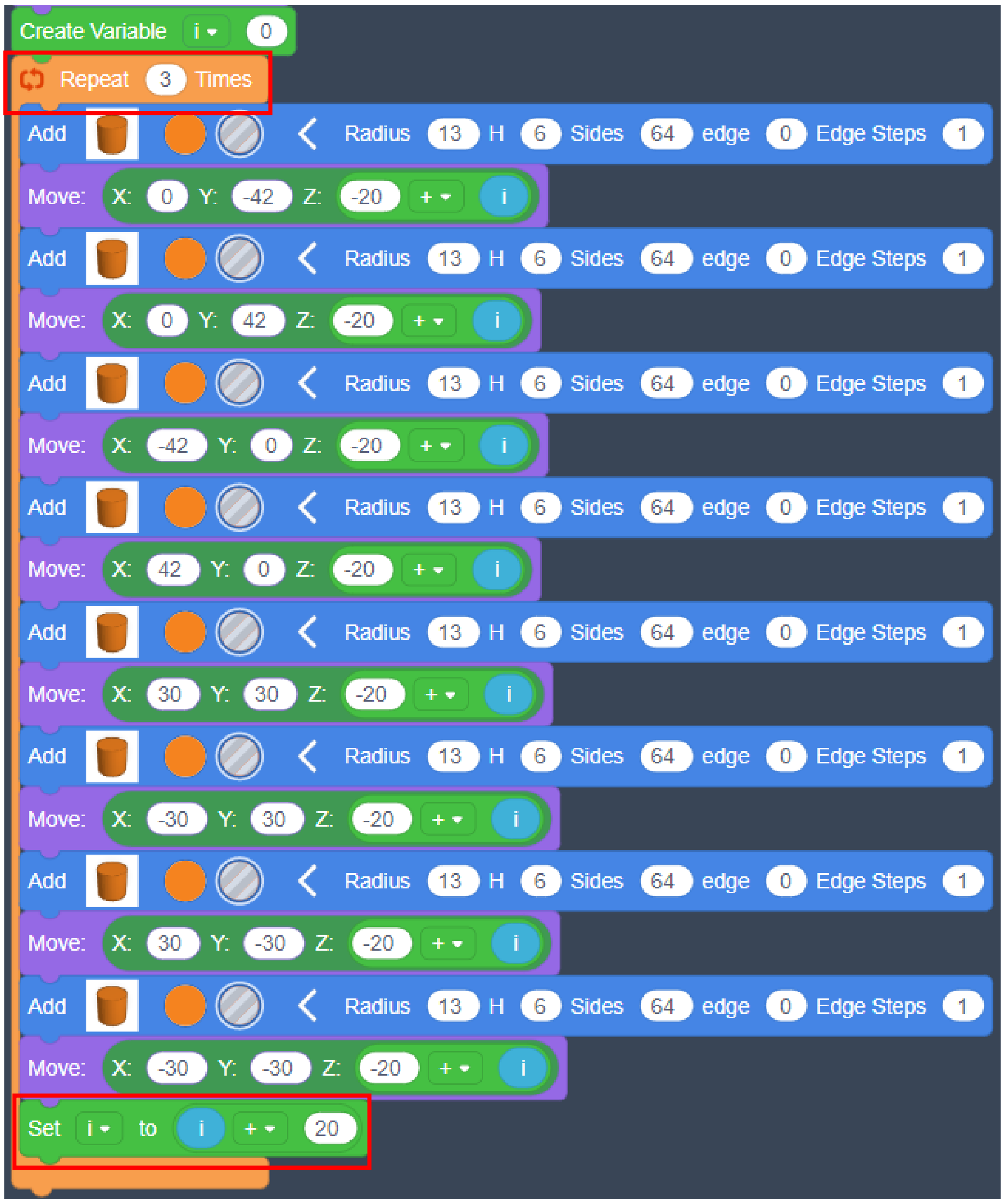


```
Create Variable i 0
Add Radius 13 H 6 Sides 64 edge 0 Edge Steps 1
Move: X: 0 Y: -42 Z: -20 + i
Add Radius 13 H 6 Sides 64 edge 0 Edge Steps 1
Move: X: 0 Y: 42 Z: -20 + i
Add Radius 13 H 6 Sides 64 edge 0 Edge Steps 1
Move: X: -42 Y: 0 Z: -20 + i
Add Radius 13 H 6 Sides 64 edge 0 Edge Steps 1
Move: X: 42 Y: 0 Z: -20 + i
Add Radius 13 H 6 Sides 64 edge 0 Edge Steps 1
Move: X: 30 Y: 30 Z: -20 + i
Add Radius 13 H 6 Sides 64 edge 0 Edge Steps 1
Move: X: -30 Y: 30 Z: -20 + i
Add Radius 13 H 6 Sides 64 edge 0 Edge Steps 1
Move: X: 30 Y: -30 Z: -20 + i
Add Radius 13 H 6 Sides 64 edge 0 Edge Steps 1
Move: X: -30 Y: -30 Z: -20 + i
```

Figure 6.16 Adding another seven cylinders and setting their movement coordinates

You can design pattern modules yourselves.

Step 12: Set the array pattern module, set *i* to i + 20, and the Z coordinate increases by 20 mm each time. Add the cylinder part and repeat the addition 3 times, as shown in Figure 6.17.

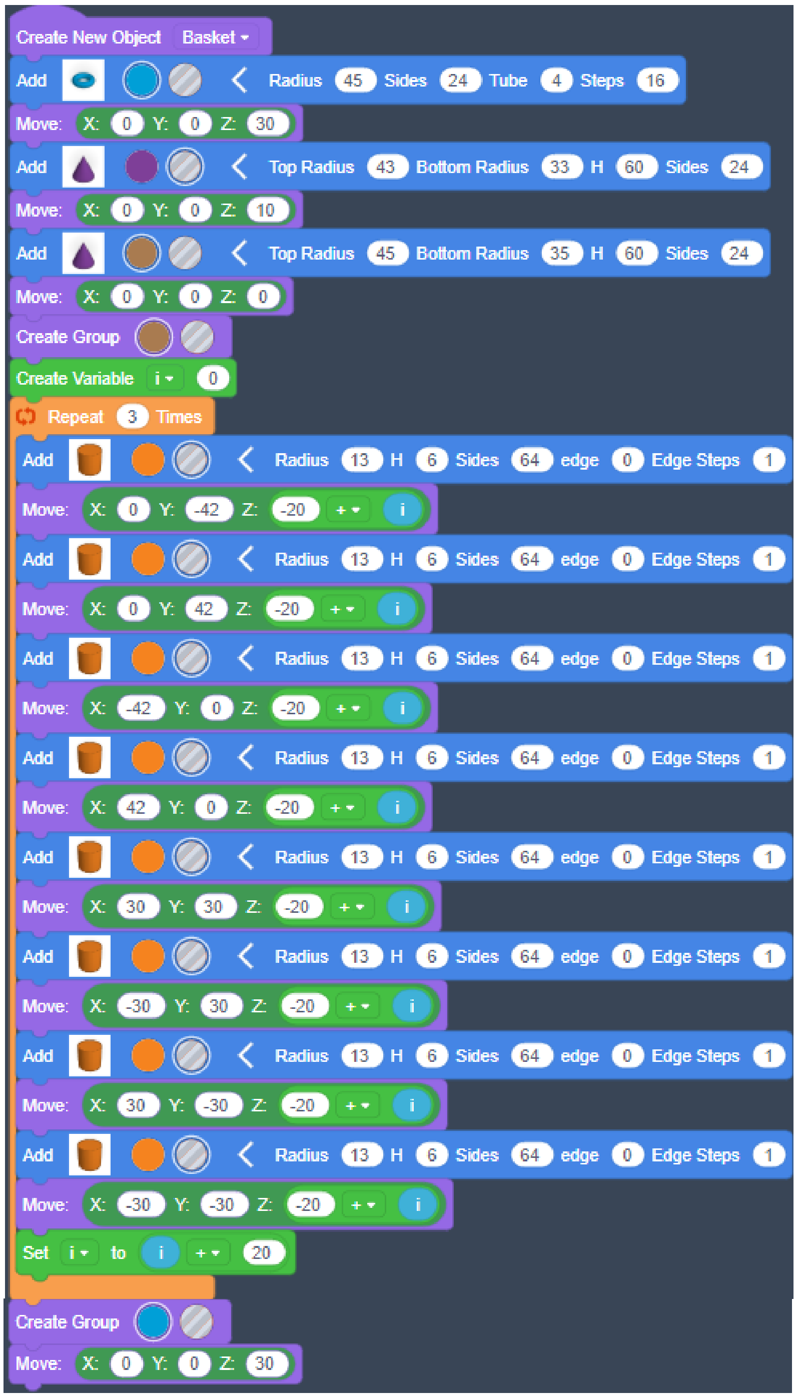


```
Create Variable i ▾ 0
Repeat 3 Times
  Add  < Radius 13 H 6 Sides 64 edge 0 Edge Steps 1
  Move: X: 0 Y: -42 Z: -20 + ▾ i
  Add  < Radius 13 H 6 Sides 64 edge 0 Edge Steps 1
  Move: X: 0 Y: 42 Z: -20 + ▾ i
  Add  < Radius 13 H 6 Sides 64 edge 0 Edge Steps 1
  Move: X: -42 Y: 0 Z: -20 + ▾ i
  Add  < Radius 13 H 6 Sides 64 edge 0 Edge Steps 1
  Move: X: 42 Y: 0 Z: -20 + ▾ i
  Add  < Radius 13 H 6 Sides 64 edge 0 Edge Steps 1
  Move: X: 30 Y: 30 Z: -20 + ▾ i
  Add  < Radius 13 H 6 Sides 64 edge 0 Edge Steps 1
  Move: X: -30 Y: 30 Z: -20 + ▾ i
  Add  < Radius 13 H 6 Sides 64 edge 0 Edge Steps 1
  Move: X: 30 Y: -30 Z: -20 + ▾ i
  Add  < Radius 13 H 6 Sides 64 edge 0 Edge Steps 1
  Move: X: -30 Y: -30 Z: -20 + ▾ i
  Set i ▾ to i + ▾ 20
```

Figure 6.17 Setting the variable and repeating the addition 3 times

Why do we need to repeat the steps to include all the blocks that create the pattern module?

Step 13: Add **Create Group** outside the loop body, select solid, select the final color, and move the entire built model to the position (0, 0, 30), as shown in Figure 6.18.



```
Create New Object  Basket
Add  Radius 45  Sides 24  Tube 4  Steps 16
Move: X: 0  Y: 0  Z: 30
Add  Top Radius 43  Bottom Radius 33  H 60  Sides 24
Move: X: 0  Y: 0  Z: 10
Add  Top Radius 45  Bottom Radius 35  H 60  Sides 24
Move: X: 0  Y: 0  Z: 0
Create Group
Create Variable  i  0
Repeat 3 Times
  Add  Radius 13  H 6  Sides 64  edge 0  Edge Steps 1
  Move: X: 0  Y: -42  Z: -20 + i
  Add  Radius 13  H 6  Sides 64  edge 0  Edge Steps 1
  Move: X: 0  Y: 42  Z: -20 + i
  Add  Radius 13  H 6  Sides 64  edge 0  Edge Steps 1
  Move: X: -42  Y: 0  Z: -20 + i
  Add  Radius 13  H 6  Sides 64  edge 0  Edge Steps 1
  Move: X: 42  Y: 0  Z: -20 + i
  Add  Radius 13  H 6  Sides 64  edge 0  Edge Steps 1
  Move: X: 30  Y: 30  Z: -20 + i
  Add  Radius 13  H 6  Sides 64  edge 0  Edge Steps 1
  Move: X: -30  Y: 30  Z: -20 + i
  Add  Radius 13  H 6  Sides 64  edge 0  Edge Steps 1
  Move: X: 30  Y: -30  Z: -20 + i
  Add  Radius 13  H 6  Sides 64  edge 0  Edge Steps 1
  Move: X: -30  Y: -30  Z: -20 + i
  Set  i  to  i + 20
Create Group
Move: X: 0  Y: 0  Z: 30
```

Figure 6.18 Combination model

Step 14: Run the program and view the modeling animation, as shown in Figure 6.19.

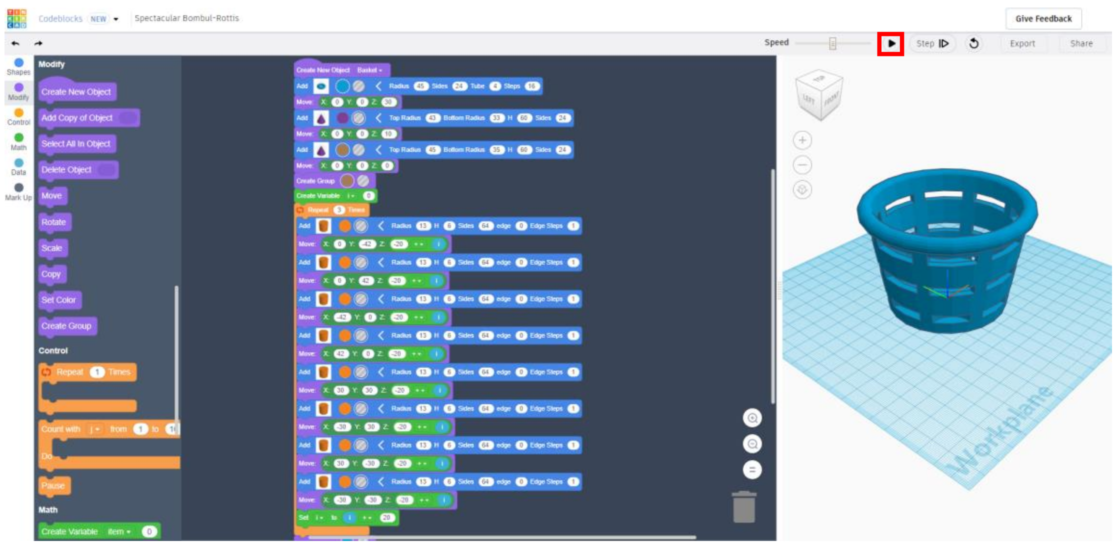


Figure 6.19 Running the program

What effect can we achieve by repeating the steps three times to add eight cylinders and change the movement coordinates?

3. Summary

(1) Set the movement coordinates: ______________________________
____

(2) Describe the role of "create group": __________________________
____

(3) The array functions in Tinkercad are implemented through _______
____

4. Self-Assessment

Check the completed content in the experiment task. Tick（√）the completed items, and circle (◯) the uncompleted items.

Assessment Content	Completion Status
I’ve used the movement module	
I’ve used the variable module	
I’ve used the repeated execution module	

I’ve implemented the array function	
I’ve designed the radish basket model	

Task 3: Display the RGB Value on the Serial Port

A color sensor is a sensing device that compares the color of an object with a demonstrated reference color to detect the color. In this experiment, we will rely on the serial port monitor to display the RGB value of the color of the light reflected by the red radish and white radish to the sensor, and analyze the difference between the RGB values of red radish and white radish.

When we select a color filter, it only allows a certain primary color to pass, and prevents other primary colors from passing. For example: a red filter only allows red in the incident light to pass, but blocks blue and green. Thus, we can obtain the maximum value of red light intensity. By comparing the values of the three light intensity values RGB, we can analyze the color of the light reflected on the sensor, and further determine the color of the object.

1. Analysis

Analyze the steps to obtain the RGB values. Then, complete the steps in the table below.

Step 1: Determine whether the color sensor is connected properly.

Step 2: If the connection is normal, the serial port displays _________; otherwise, stop and check the connection of the color sensor.

Step 3: Get _______ of the color sensor.

Step 4: of the color sensor.

2. Steps

(1) Prepare Hardware

Step 1: Connect the hardware, as shown in Figure 6.20.

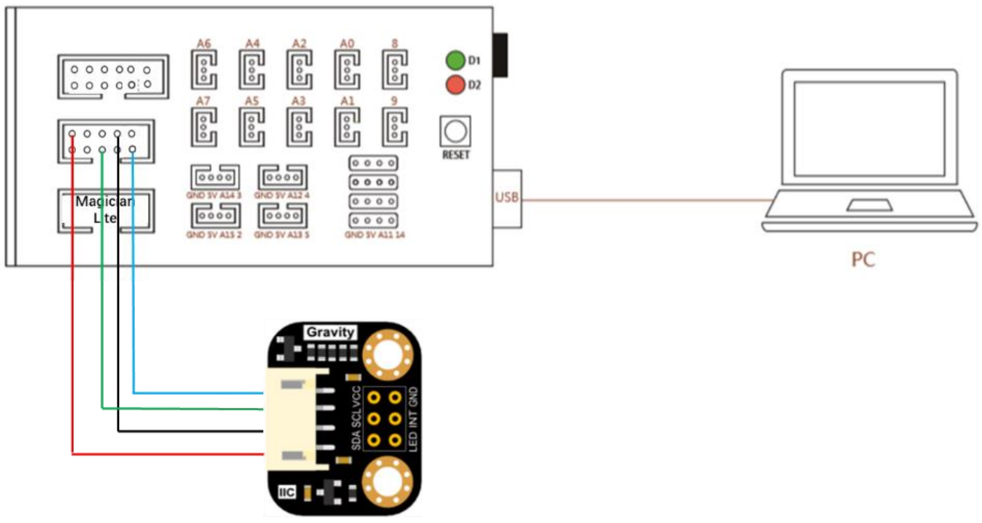


Figure 6.20 Hardware connection diagram

Step 2: Prepare the experiment equipment. Connect it yourselves.

(2) Design Program

Step 1: Analyze and print the color sensor data implementation process, and draw a flow chart, as shown in Figure 6.21.

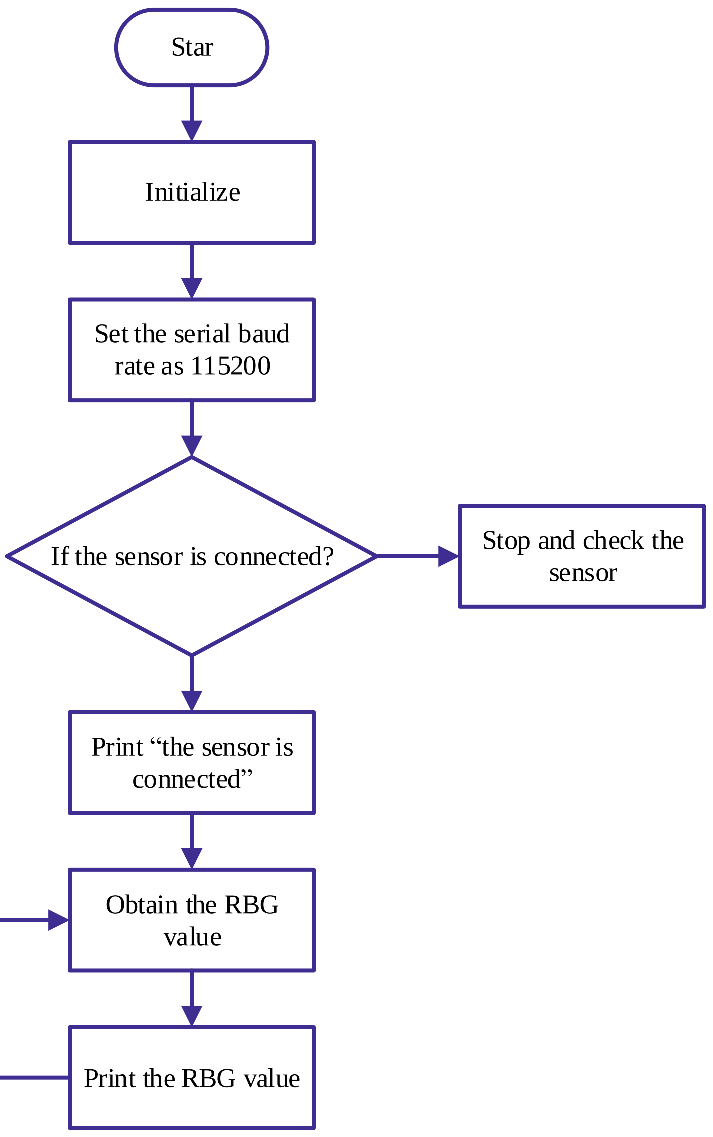

Star
Initialize
Set the serial baud rate as 115200
If the sensor is connected?
Stop and check the sensor
Print “the sensor is connected”
Obtain the RBG value
Print the RBG value

Figure 6.21 Flow chart

Step 2: Set the header file and get an object. The programming method is as follows:

```
#include<Wire.h>//Header file
#include"DFRobot_TCS34725.h"//Header file
DFRobot_TCS34725 tcs = DFRobot_TCS34725
(TCS34725_INTEGRATIONTIME_50MS, TCS34725_GAIN_4X)；// Get an object
```

Step 3: Set the setup function and set the baud rate to 115200 to judge whether the sensor is connected normally.

```
void setup(){
Serial.begin(115200);                    // Set the baud rate to 115200
if(tcs.begin()){                  // Detect the sensor
```

```
        Serial.println ("Sensor connection is normal"); // Print "Sensor connection is normal"
    }
    else{
    Serial.println("Please check the sensor connection");
    while (1); // End
    }
}
```

Step 4: Set the main loop function.

1) Get the RGB value.

```
voidloop(){
intclear, R, G, B;
tcs.getRGBC(&R, &Gn, &B, &clear); // Get RGB value
}
```

2) Print the RGB values.

```
Serial.print("\tR:\t"); Serial.print(R);      // Print the value of R
Serial.print("\tG:\t"); Serial.print(G); // Print the value of G
Serial.print("\tB:\t");Serial.print(B);    // Print the value of B
Serial.println("\t");
```

Step 7: Integrate the program.

```
#include<Wire.h>//Header file
#include "DFRobot_TCS34725.h"//Header file
DFRobot_TCS34725 tcs = DFRobot_TCS34725
(TCS34725_INTEGRATIONTIME_50MS, TCS34725_GAIN_4X)； // Get an object
voidsetup(){
Serial.begin(115200);                // Set the baud rate to 115200
if(tcs.begin()){                  // Detecting sensor
        Serial.println("Sensor connection is normal");// Print "Sensor connection is normal"
    }
    else{
    Serial.println("Please check the sensor connection");
    while (1); // End
    }
}
voidloop(){
intclear, R, G, B;
tcs.getRGBC(&R, &G, &B, &clear); // Get RGB value
Serial.print("\tR:\t"); Serial.print(R);      // Print the value of R
Serial.print("\tG:\t"); Serial.print(G); // Print the value of G
Serial.print("\tB:\t");Serial.print(B);    // Print the value of B
```

```
    Serial.println("\t");
}
```

Observe the relationship between the RGB values of red radish and white radish to find the maximum value.

3. Summary

(1) The conditions for judging that the color sensor is normally connected are: ______________________________________
____

(2) The instruction to obtain the RGB value of the color sensor is: ___
____

(3) The instruction to display the RGB value of the color sensor is: __
____

(4) The maximum value in red radish RGB is:____________ The maximum value in white radish RGB is:______________________
____

4. Self-Assessment

Check the completed content in the experiment task. Tick（√）the completed items, and circle (◯) the uncompleted items.

Assessment Content	Completion Status
I've used the color sensor	
I've judged whether the color sensor is connected normally	
I've used the instruction to obtain the RGB values of the color sensor	
I've implemented the program to display RGB value on the serial port	

I've compared the relationship between the RGB values of red radishes and white radishes	

Task 4: Sort the Radish

Compare the RGB values of red radishes and white radishes, and find the conditions for judging red radishes. Thus, we can start to sort radishes. Let's look at how to use the robotic arm to automatically do so.

1. Analysis

Analyze the steps to sort radishes and then complete the steps in the blanks in the table below.

Step 1: Determine the number of radishes to grab.

Step 2:The first radish.

Step 3: Move the radish to position.

Step 4: Judge.

Step 5: If it is red radish, then, otherwise,.

Step 6: Calculate the number of radishes captured and re-grab them.

2. Steps

(1) Prepare Hardware

Step 1: Set the scenario of sorting radishes, as shown in Figure 6.22.

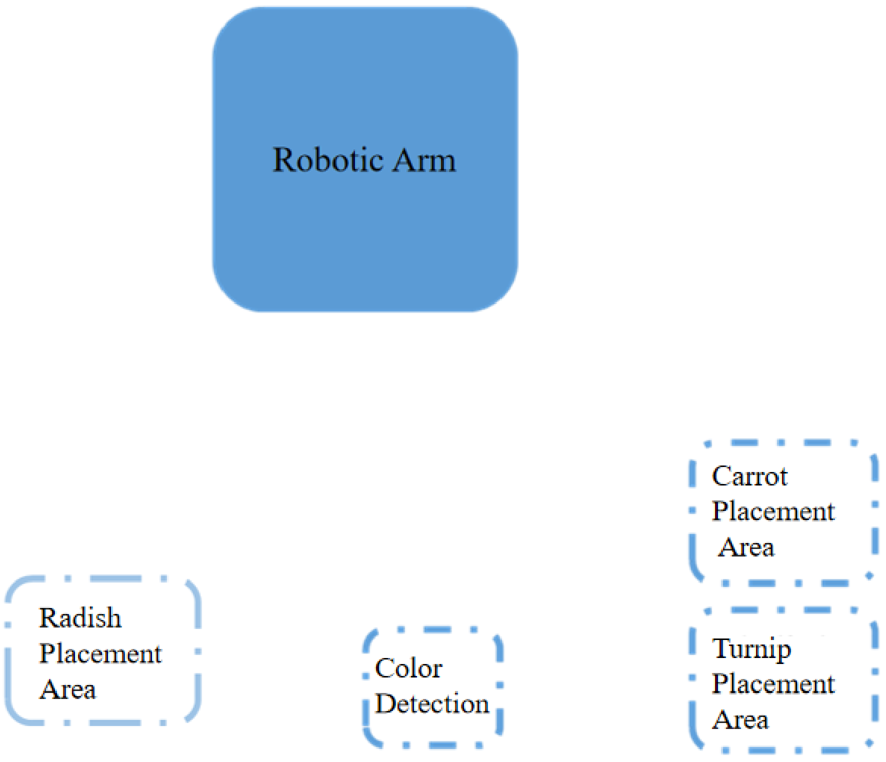


Figure 6.22 Scenario placement diagram

Step 2: Connect the hardware, as shown in Figure 6.23.

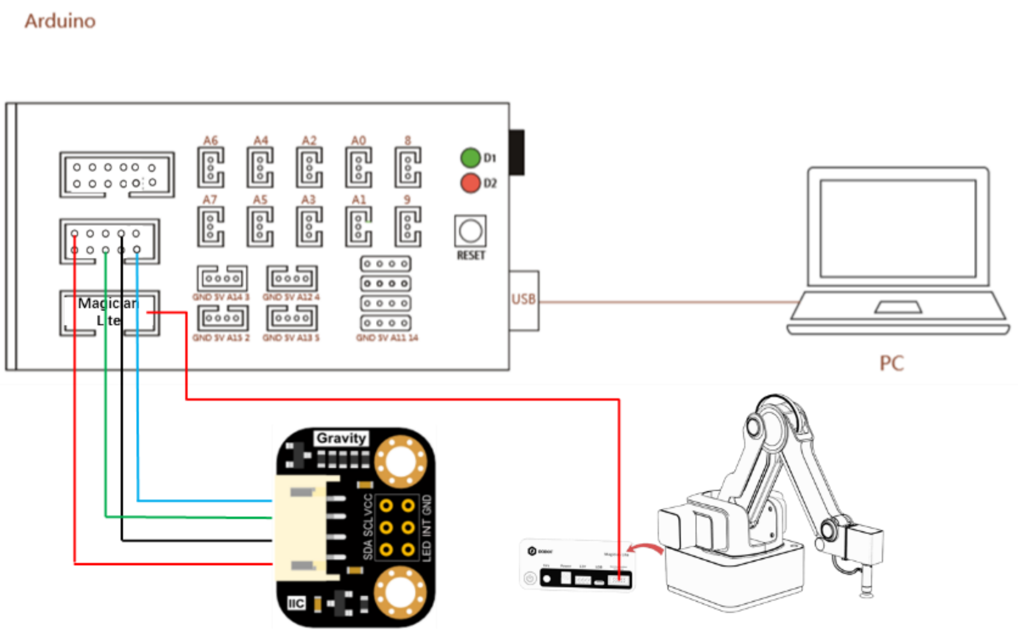


Figure 6.23 Hardware connection diagram

(2) Design Program

Step 1: Analyze the process of sorting red and white radishes, and draw a flowchart, as shown in Figure 6.24.

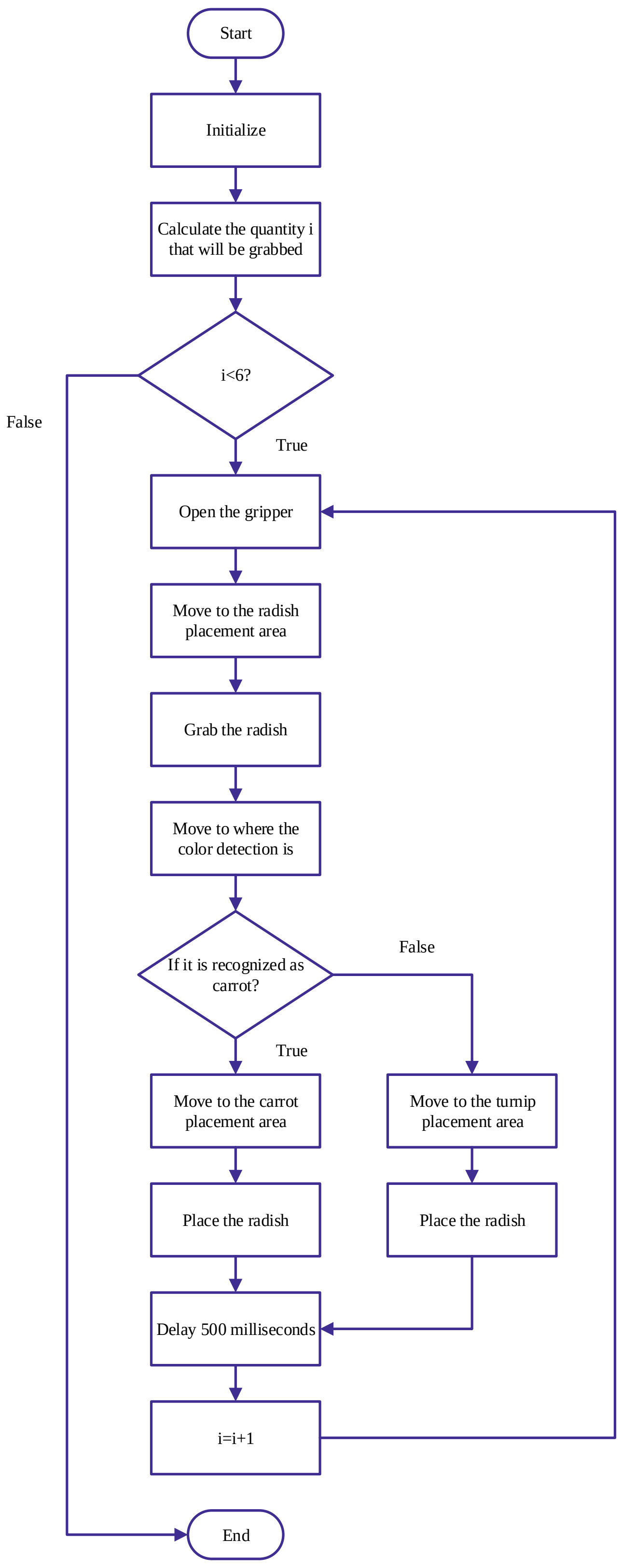

Start
Initialize
Calculate the quantity i that will be grabbed
i<6?
False
True
Open the gripper
Move to the radish placement area
Grab the radish
Move to where the color detection is
If it is recognized as carrot?
False
True
Move to the carrot placement area
Move to the turnip placement area
Place the radish
Place the radish
Delay 500 milliseconds
i=i+1
End

Figure 6.24 Radish sorting flow chart

Step 2: Set the header file and get the color sensor object. This task requires MagicianLite, so you need to add the header file of MagicianLite.

```
#include<MagicianLite.h>// Set header file
#include<Wire.h>//Header file
#include"DFRobot_TCS34725.h"//Header file
DFRobot_TCS34725 tcs = DFRobot_TCS34725
(TCS34725_INTEGRATIONTIME_50MS, TCS34725_GAIN_4X);
```

Step 3: Define variables, define the integer variable *i*, and record the number of picks.

```
int i=0;    // Record the number of grab times
```

Step 4: Set up the setup function.

1) Initialize the control board, set the movement ratio of the robotic arm and move the end of the robotic arm to the position above the radish. Try writing the corresponding code yourself.

2) Set the baud rate to 115200 and judge whether the sensor is connected normally. As per Task 3, write the code yourselves.

Step 5: Set the main loop function.

1) While loop: for example, to grab six radishes, loop six times; before grabbing them, define the related variables. The programming method is as follows:

```
voidloop(){
intclear, R, G, B;
   While(i<6)
}
```

2) Program the robotic arm to grab the radish. First, open your gripper and move it to the radish basket. Close the gripper to grab the radish and move the robotic arm upward.

```
        MagicianLite_SetEndEffectorGripper(true,false)   // Open gripper
        //Move to the radish grabbing position
        MagicianLite_SetPTPCmd(JUMP_XYZ, 318.18,-58.45,36.90,0);
        MagicianLite_SetEndEffectorGripper(true,true); // Grab the radish
        MagicianLite_SetPTPCmd(MOVL_INC, 0,0,40,0);      // Move upward
        delay(200);
```

3) After catching the radish, move it to the monitoring color position. The color sensor judges whether the radish is red radish or white radish, then, put the radish in the corresponding basket.

```
        // Detect color position
        MagicianLite_SetPTPCmd(JUMP_XYZ, 242.79, 178.85, 29.42, 0);
        tcs.getRGBC(&red, &green, &blue, &clear);         // Get the RGB value of the radish
        delay(2000);                                       // Detection time is 2000ms
        if(red>green & red>blue ){                        //Determine if it is red
// Move to the place where the red radish is placed
          MagicianLite_SetPTPCmd(JUMP_XYZ, 242.79, 178.85, 29.42, 0);
        MagicianLite_SetEndEffectorGripper(true,false);// Put down red radish
        }
        else{
          // Move to the place where the white radish is placed
          MagicianLite_SetPTPCmd(JUMP_XYZ, 232.72, 87.26, 25.43, 0);
          MagicianLite_SetEndEffectorGripper(true,false);    // Put down white radish
        }
        delay(200);
        MagicianLite_SetEndEffectorGripper(false,false);// The gripper is in natural state
```

Step 7: Sort the radish, count with the variable *i*, and record the total number of grabs, which is not greater than 6.

```
        delay(500);
        i=i+1;
```

Step 8: According to the step analysis, integrate the program.

```
#include<MagicianLite.h>// Set header file
#include<Wire.h>//Header file
#include"DFRobot_TCS34725.h"//Header file
```

```
DFRobot_TCS34725 tcs = DFRobot_TCS34725
(TCS34725_INTEGRATIONTIME_50MS, TCS34725_GAIN_4X);
int i=0;    // Record the number of grabs

voidsetup(){
MagicianLite_Init();                                    //Initialize the control board
MagicianLite_SetPTPCommonParams(80,80);      //Set the robotic arm movement ratio
   //Move robotic arm above the seed position
   MagicianLite_SetPTPCmd(JUMP_XYZ, 318.18, -58.45, 76.90, 0);
   Serial.begin(115200);

voidloop(){
intclear, R, G, B;
   While(i<6){
      MagicianLite_SetEndEffectorGripper(true,false)   // Open the gripper
       //Move to the radish grab position
       MagicianLite_SetPTPCmd(JUMP_XYZ, 318.18,-58.45,36.90,0);
       MagicianLite_SetEndEffectorGripper(true,true); //Grab the radish
       MagicianLite_SetPTPCmd(MOVL_INC, 0,0,40,0);      // Move upward
      delay(200);
       // Detect the color position
       MagicianLite_SetPTPCmd(JUMP_XYZ, 242.79, 178.85, 29.42, 0);
       tcs.getRGBC(&red, &green, &blue, &clear);            // Get the RGB value of the radish
       delay(2000);                                            // Detection time is 2000ms
       if(red>green & red>blue ){                             // Judge if it is red
// Move to the place where the red radish is placed
         MagicianLite_SetPTPCmd(JUMP_XYZ, 242.79, 178.85, 29.42, 0);
       MagicianLite_SetEndEffectorGripper(true,false);// Put down red radish
       }
       else{
         // Move to the place where the white radish is placed
         MagicianLite_SetPTPCmd(JUMP_XYZ, 232.72, 87.26, 25.43, 0);
         MagicianLite_SetEndEffectorGripper(true,false);     // Put down the white radish
       }
       delay(200);
      MagicianLite_SetEndEffectorGripper(false,false);// The gripper is in the natural state
delay(500);
        i=i+1;
       }
}
```

Step 8: Upload and run the program on the equipment.

Summarize how to control the gripping API "MagicianLite_SetEndEffectorGripper ()".

3. Summary

(1) The radish grabbing process through the robotic arm is: __________
____

(2) The reason for placing radishes above the color sensor is: ________
____

(3) The conditions for judging red radishes are:____________________
____

4. Self-Assessment

Check the completed content in the experiment task. Tick（√） the completed items, and circle (◯) the uncompleted items.

Assessment Content	Completion Status
I've known the radish grabbing process	
I've used the instructions to open, close, and place naturally the gripper	
I've known the conditions for judging red radishes	
I've programmed to sort radishes	