1.Configure the Arduino

1.1 What is Arduino?

Arduino is an open-source electronics platform based on easy-to-use hardware and software. Arduino boards are able to read inputs - light on a sensor, a finger on a button, or a Twitter message - and turn it into an output - activating a motor, turning on an LED, publishing something online. You can tell your board what to do by writing the program code in the IDE and sending the instructions to the microcontroller on the board. To do so you use the Arduino programming language (based on Wiring), and the Arduino Software (IDE), based on Processing.

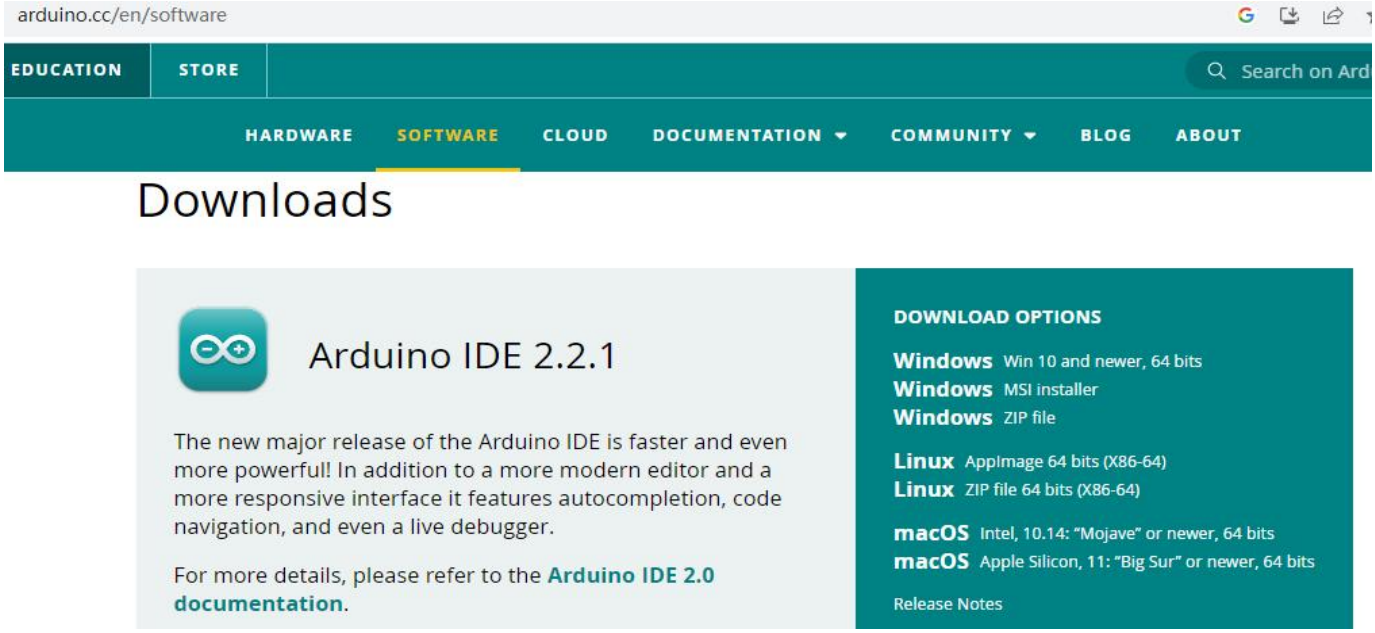
1.2 Install the Arduino IDE for Windows

1. Visit https://www.arduino.cc/en/software to download the latest Arduino IDE version for your computer’s operating system. There are versions for Windows, Mac, and Linux systems.

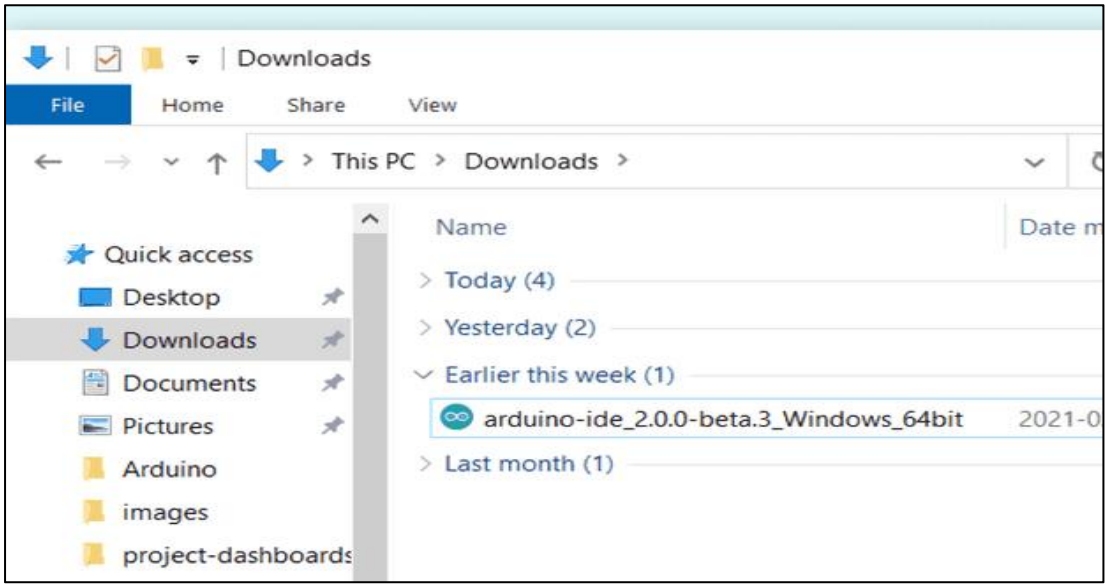
The Arduino IDE 2

The Arduino IDE 2 is a big step from its sturdy predecessor, Arduino IDE 1.x, and comes with revamped UI, improved board & library manager, debugger, autocomplete feature and much more.
Here we will show how to download and install the Arduino IDE 2.21 on your Windows
You can choose between the Installer (.exe) and the Zip packages. We suggest you use the first one that installs directly everything you need to use the Arduino Software (IDE), including the drivers. With the Zip package you need to install the drivers manually. The Zip file is also useful if you want to create a portable installation.

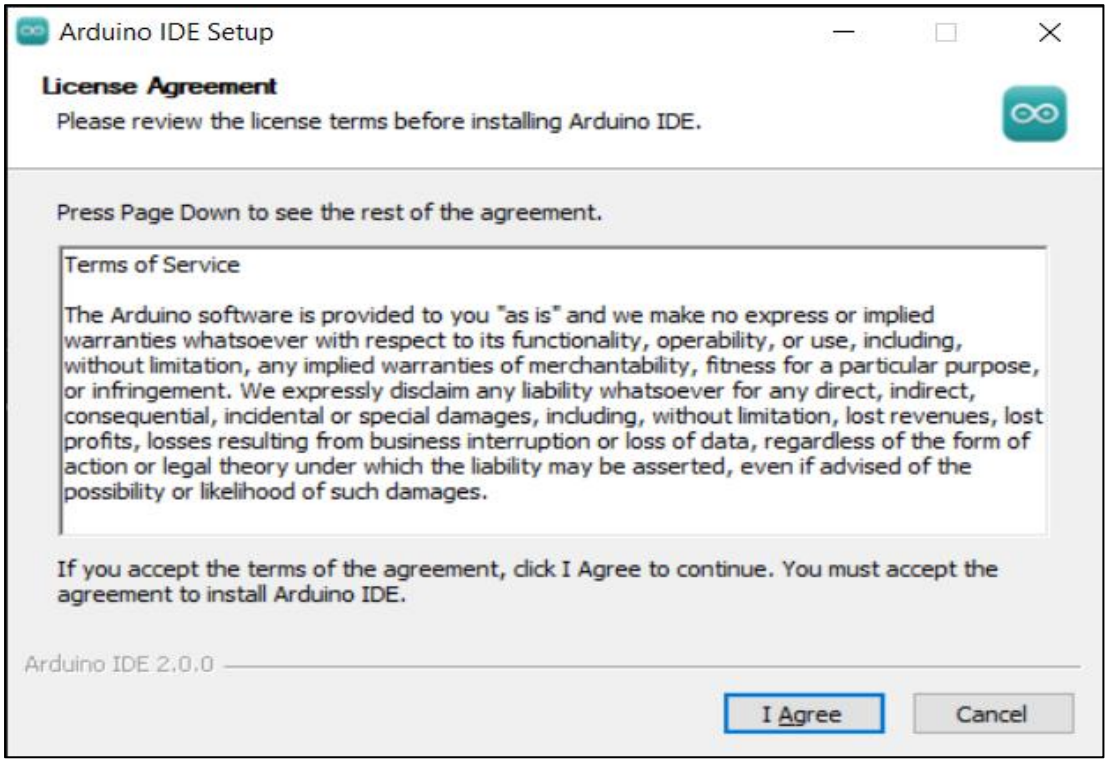
1.Here we click on the “**WindowsWin 10 and newer, 64 bits**” option for the easiest installation.



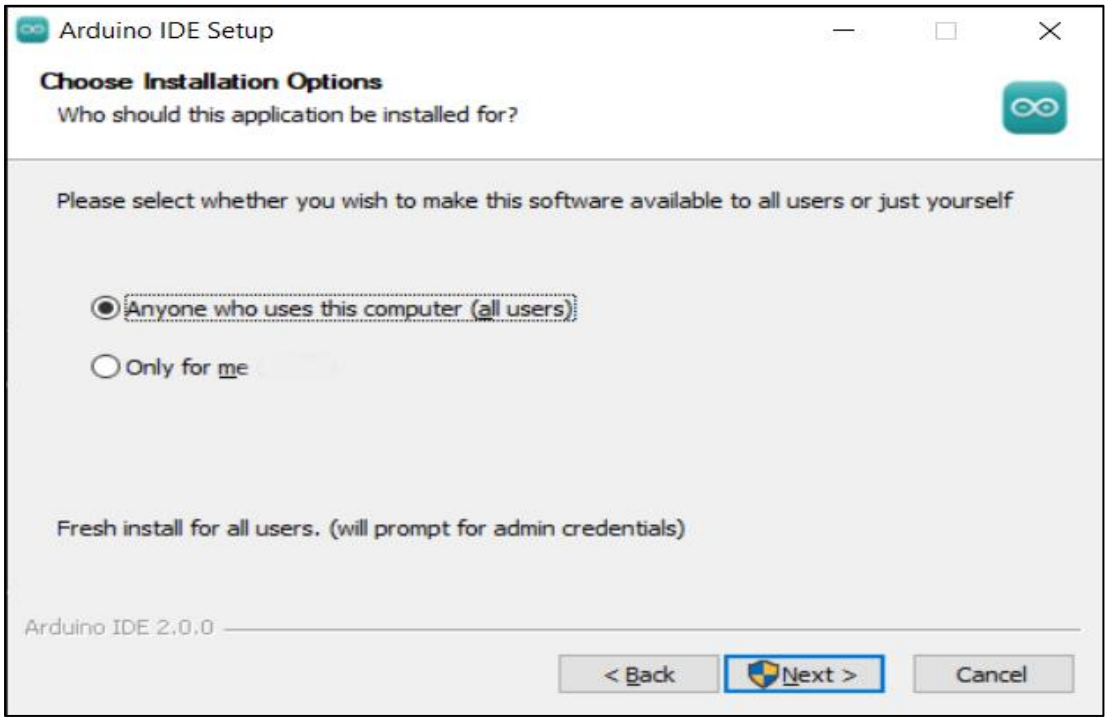
2. Save the .exe file downloaded from the software page to your hard drive and simply run the file .



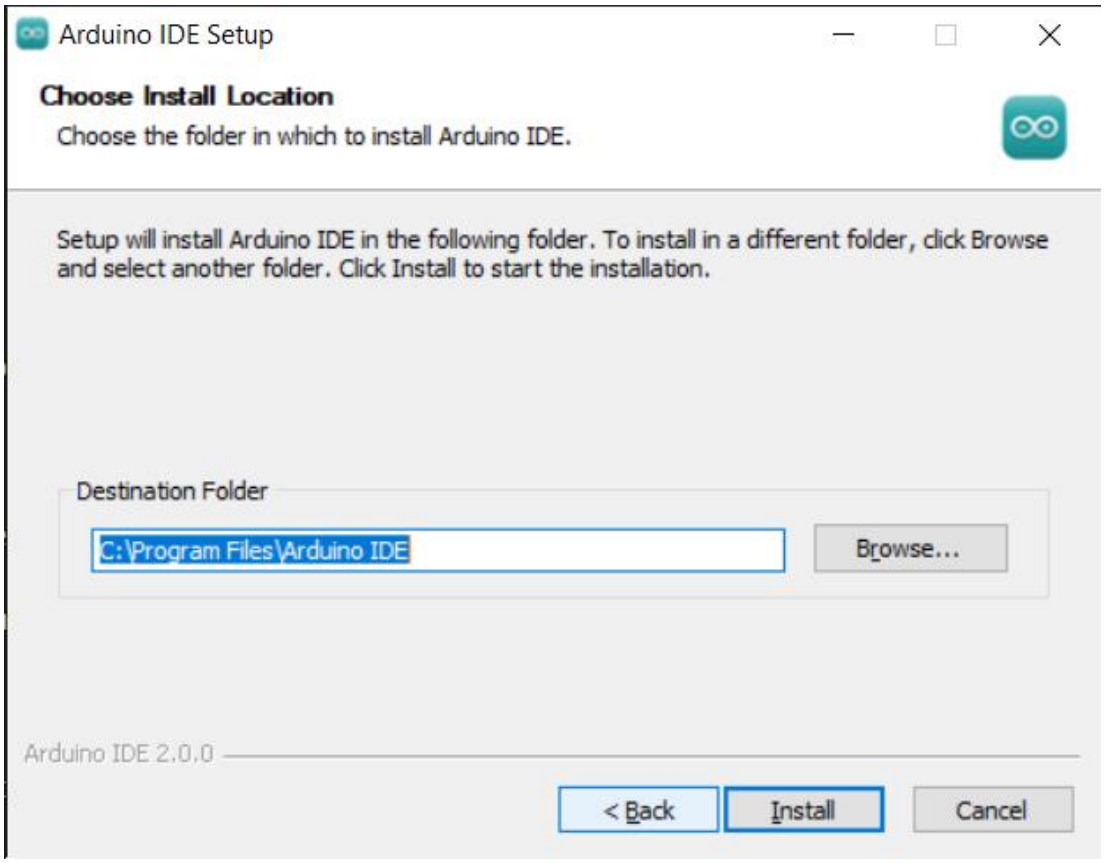
3. Read the License Agreement and agree it.



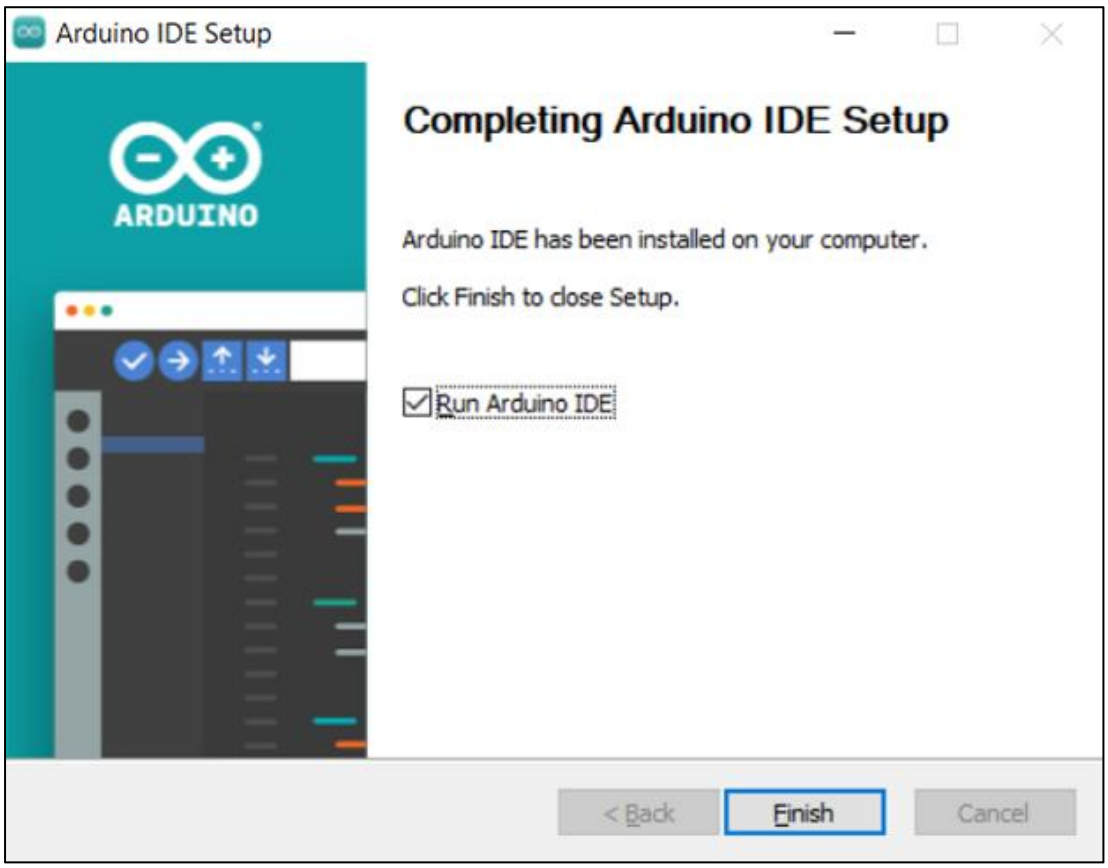
4. Choose the installation options.



5. Choose the install location.



6. Click finish and run Arduino IDE

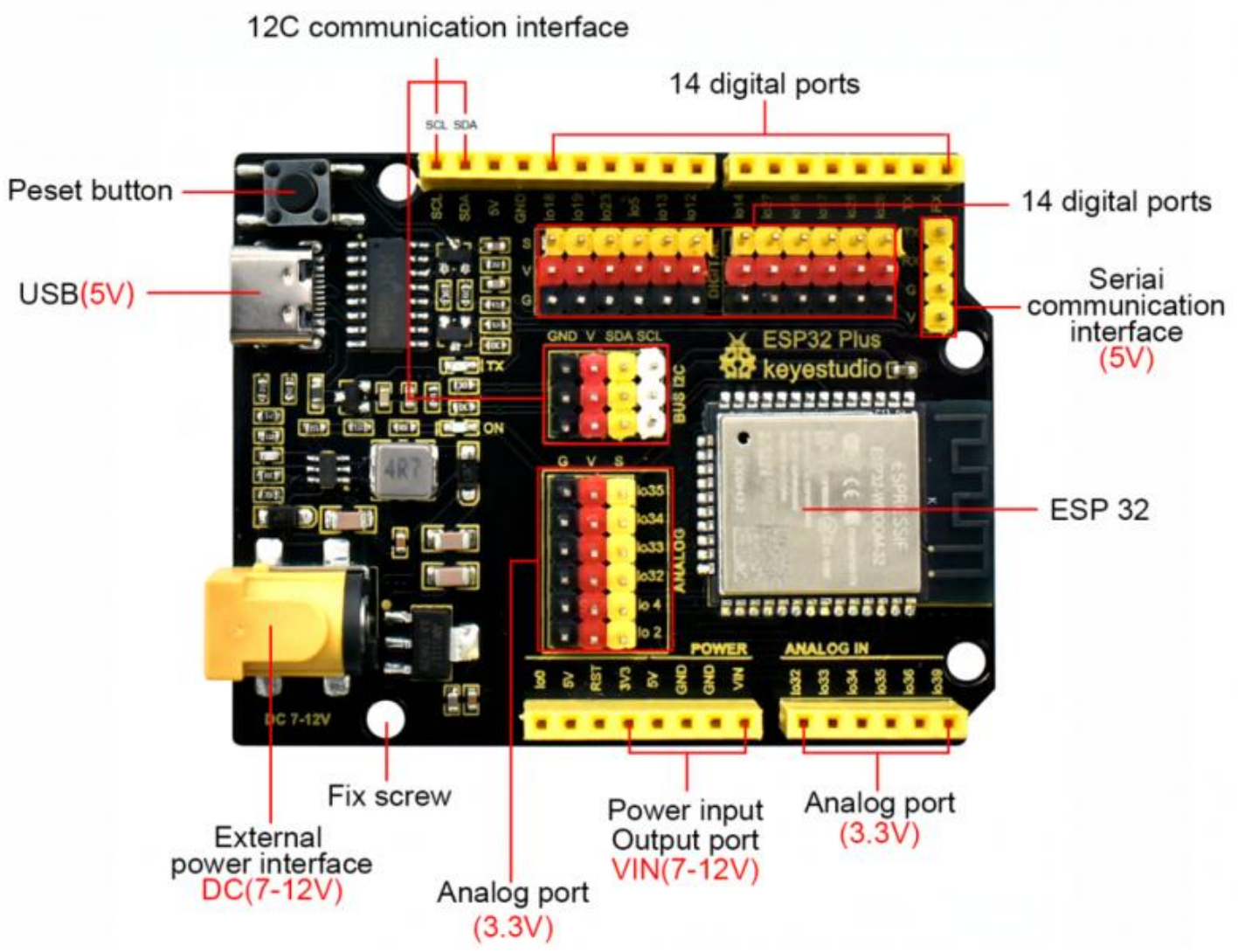


1.3 Install Driver for KEYESTUDIO ESP32 PLUS Board

KEYESTUDIO ESP32 PLUS Board a universal WIFI plus Bluetooth development board based on ESP32, integrated with ESP32-WOROOM-32 module and compatible with Arduino.
It has a hall sensor, high-speed SDIO/SPI, UART, I2S as well as I2C. Furthermore, equipped with freeRTOS operating system, which is quite suitable for the Internet of things and smart home.

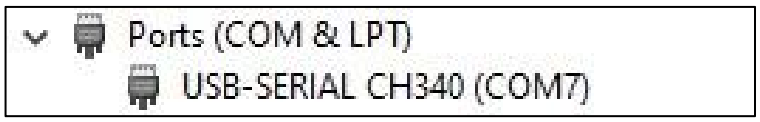
Specifications

Voltage: 3.3V-5V
Current Output: 1.2A(maximum)
Maximum power Output: 10W
Working temperature: -10℃~50℃
Dimension: 69*54*14.5mm
Weight: 25.5g
Environmental protection attributes: ROHS



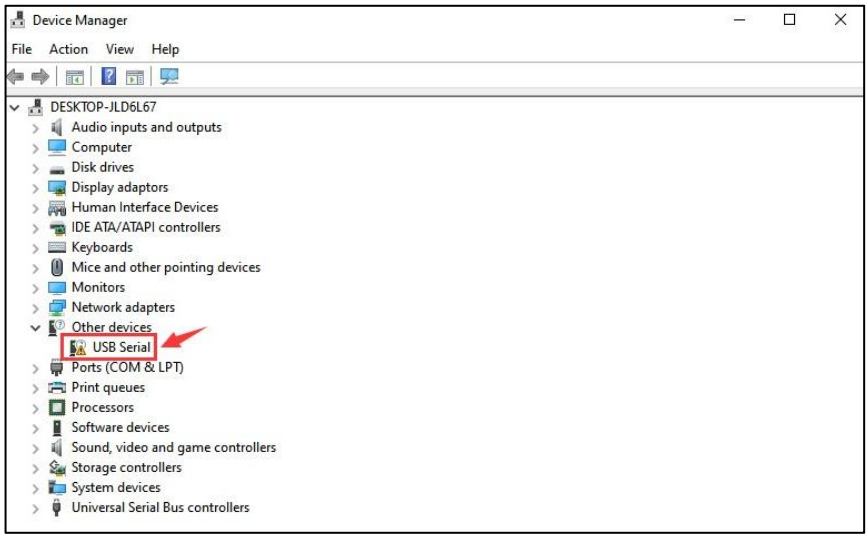
Install driver

Connect the ESP32 board to the computer and wait for Windows to begin its driver installation process. Often CH340 driver will be automatically installed by your system when using Arduino. You can check the Device Manager or the port of the Arduino IDE to see if the driver is successfully installed.

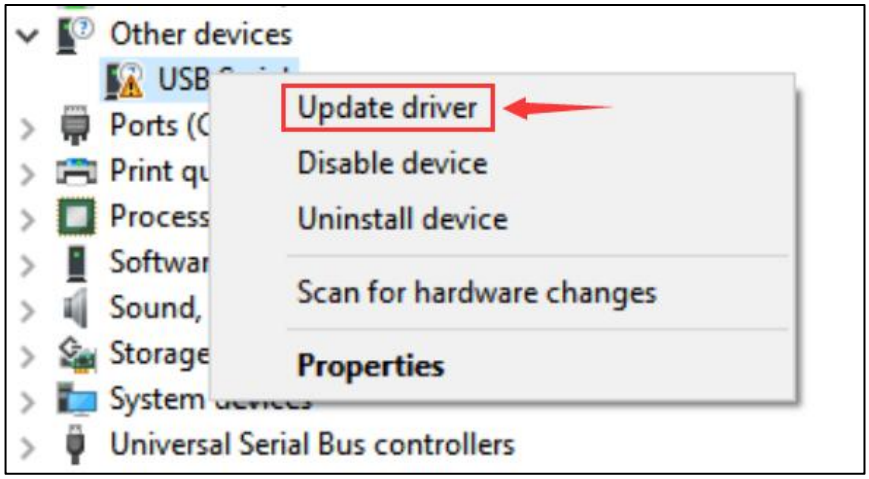


If the CH340 driver is not installed automatically, we need to install it manually. We have provided the CH340 driver in the tutorial package for you to use.

1.Open the **Device Manager** by right clicking “**My PC**” and selecting **Properties**.
Look under **Other devices**. You should see an open port named **USB Serial**

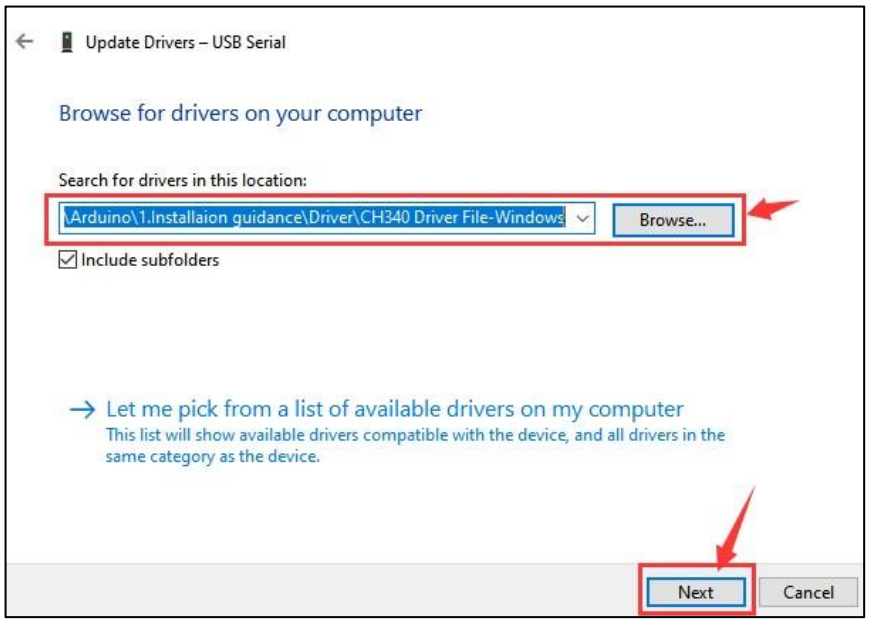


2.Right click on the "**USB Serial**" and choose the "**Update Driver**" option.

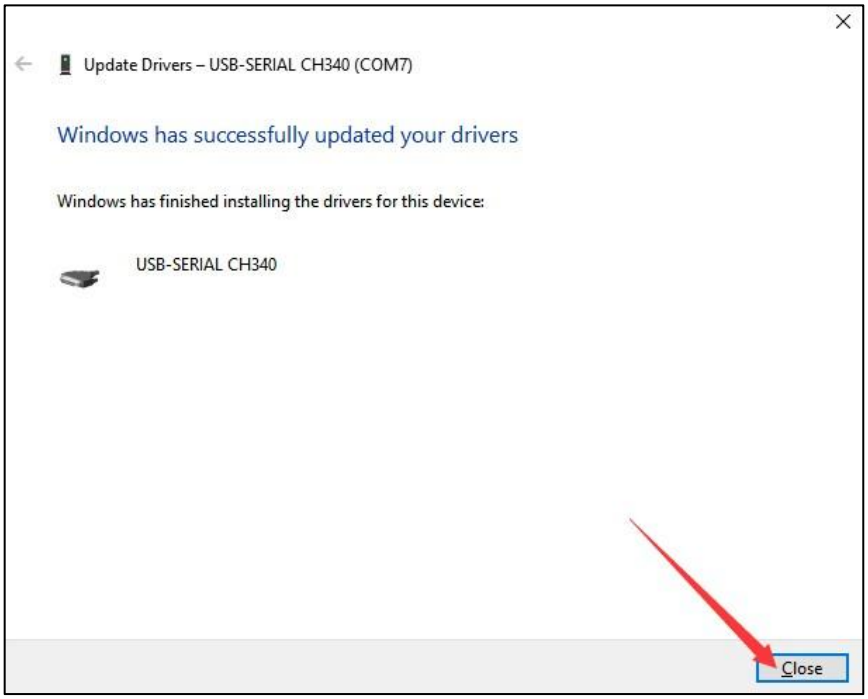


3.Choose the "**Browse my computer for Driver software**" option.

4. Select the driver file named "**CH340 Driver File-Windows**", located in the Driver Folder of the tutorial package.



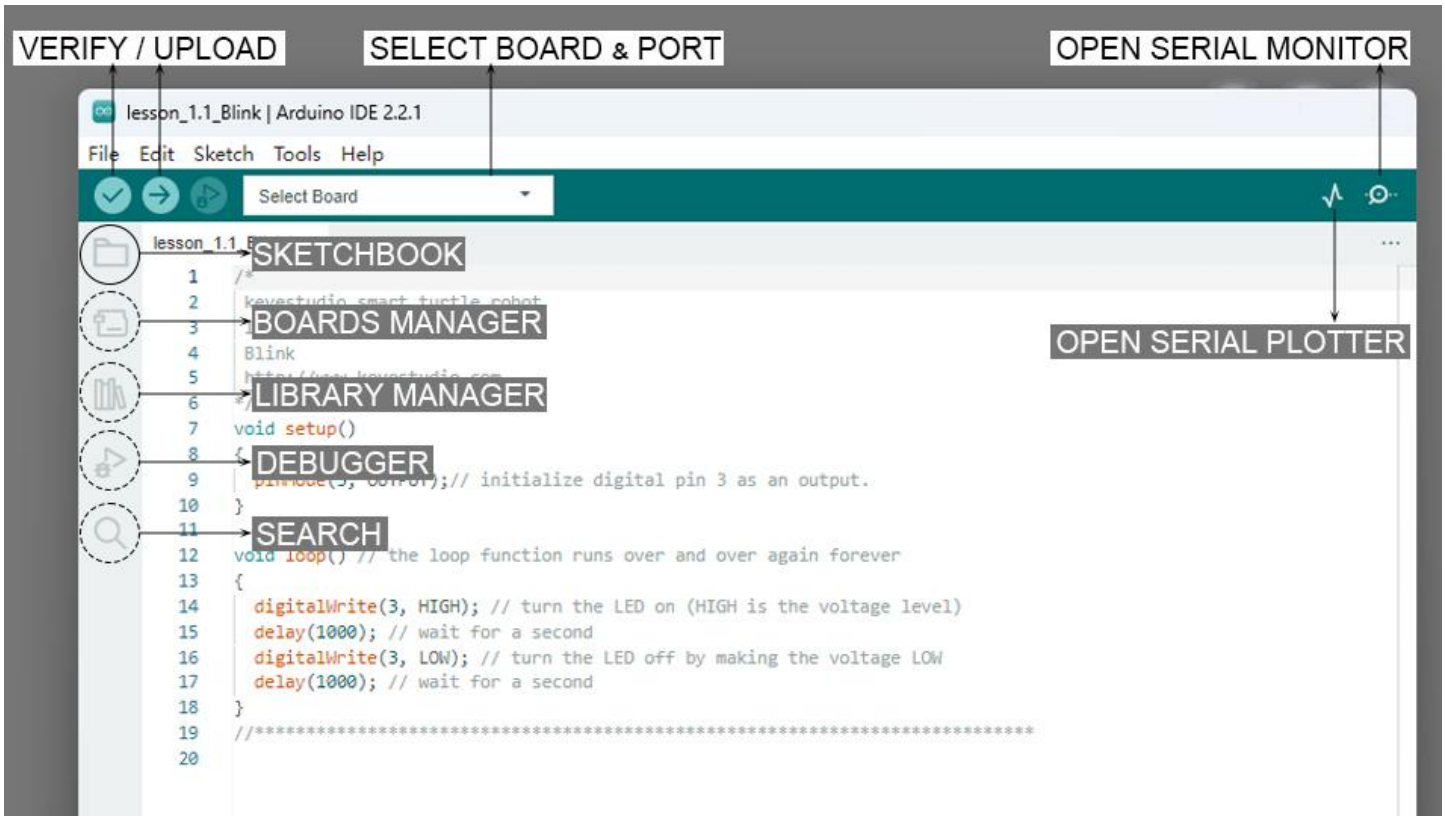
5.Driver successfully installed.



5. Device Manager will automatically refresh. Look under Ports (COM & LPT). You should see an open port named "**USB-SERIAL CH340(COM8)**"

6.Click **Tools>Port** at Arduino IDE, you can find the same COM port as the CH340 driver in the device manager.

1.4 Introduce of Arduino IDE 2.0



Verify / Upload - compile and upload your code to your Arduino Board.

Select Board & Port - detected Arduino boards automatically show up here, along with the port number.

Sketchbook - here you will find all of your sketches locally stored on your computer. Additionally, you can sync with the Arduino Cloud, and also obtain your sketches from the online environment.

Boards Manager - browse through Arduino & third party packages that can be installed. For example, using a MKR WiFi 1010 board requires the Arduino SAMD Boards package installed.

Library Manager - browse through thousands of Arduino libraries, made by Arduino & its community.

Debugger - test and debug programs in real time.

Search - search for keywords in your code.

Open Serial Monitor - opens the Serial Monitor tool, as a new tab in the console.

If you want to learn more about Arduino IDE, please refer to this document： Getting Started with Arduino IDE 2

1.5 Add Libraries to Arduino IDE

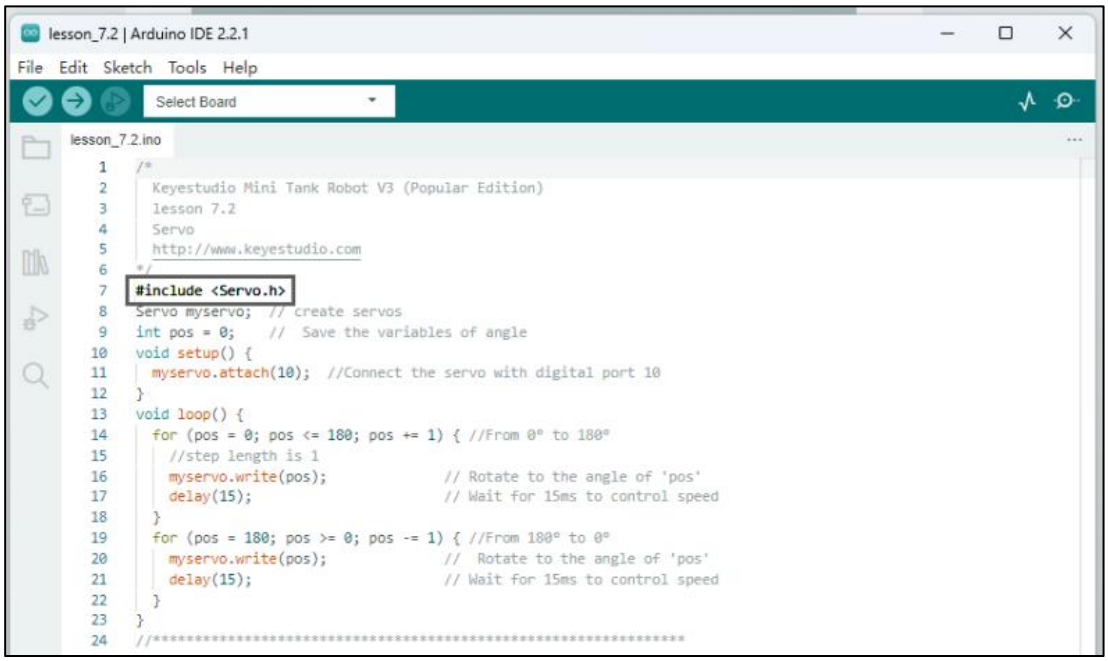
Why Use Libraries?

Libraries are incredibly useful when creating a project of any type. They make our development experience much smoother, and there almost an infinite amount out there. They are used to interface with many different sensors, RTCs, Wi-Fi modules, RGB matrices and of course with other components on your board.

Including a Library in the sketch

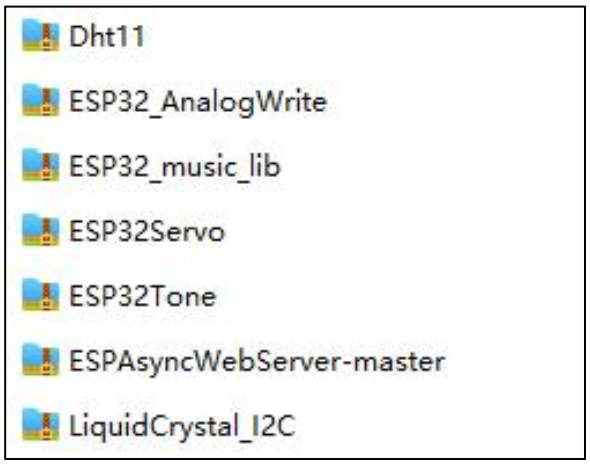
To use a library, you first need to include the library at the top of the sketch.

If you find a line of code in the format of #include <library name> at the beginning of the code when using our code, it means that you need to add this library file to arduino IDE first before you can successfully upload this code.



To make the smart farm kit work, we will need to **add these library files to the Arduino IDE.**

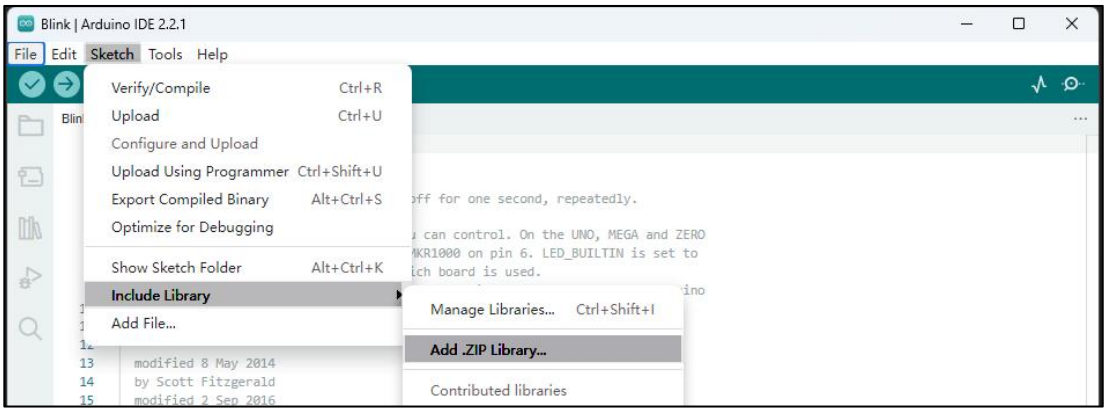
You can find them in the tutorial package.



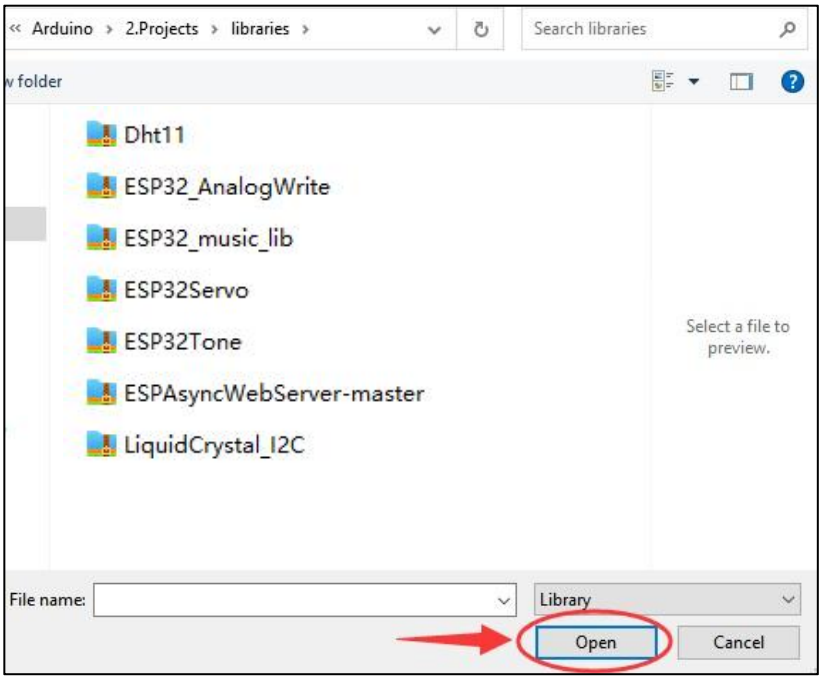
Importing a .zip Library

In the menu bar, go to **Sketch > Include Library > Add .ZIP Library**...

You will be prompted to select the library you want to add.

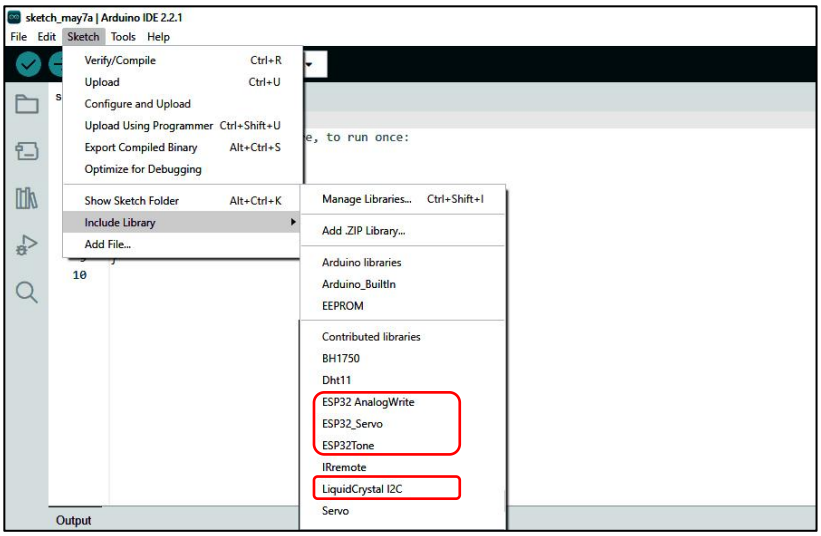


Navigate to the .zip file's location and open it.



You may need to restart the Arduino IDE for the library to be available.

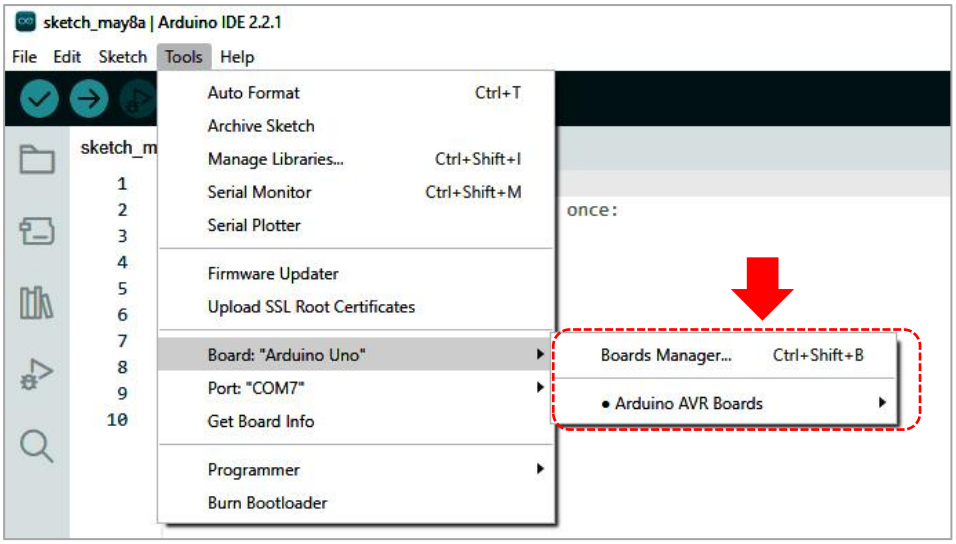
After successfully installing the library file, you will see them in the list.



1.6 Configure the development environment for ESP32

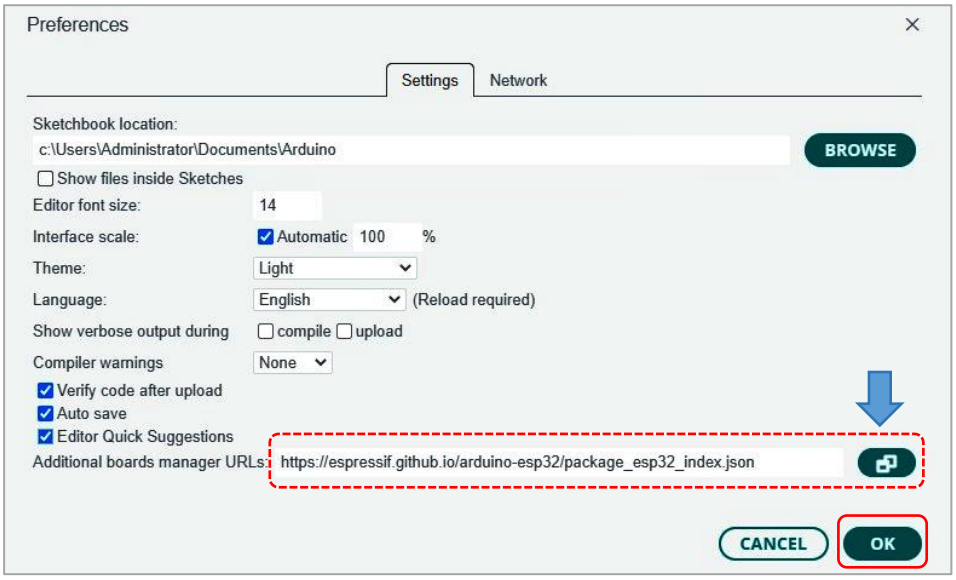
Before using Arduino IDE to program the smart farm, you need to configure the Arduino IDE, select the correct board type (**ESP32 Dev Module**) for the ESP32 Plus board, and select the **COM port** that is assigned in the device manager.

There is no option for ESP32 in Arduino's default board list, so we need to **install it manually**.



Click **File > Preferences**.

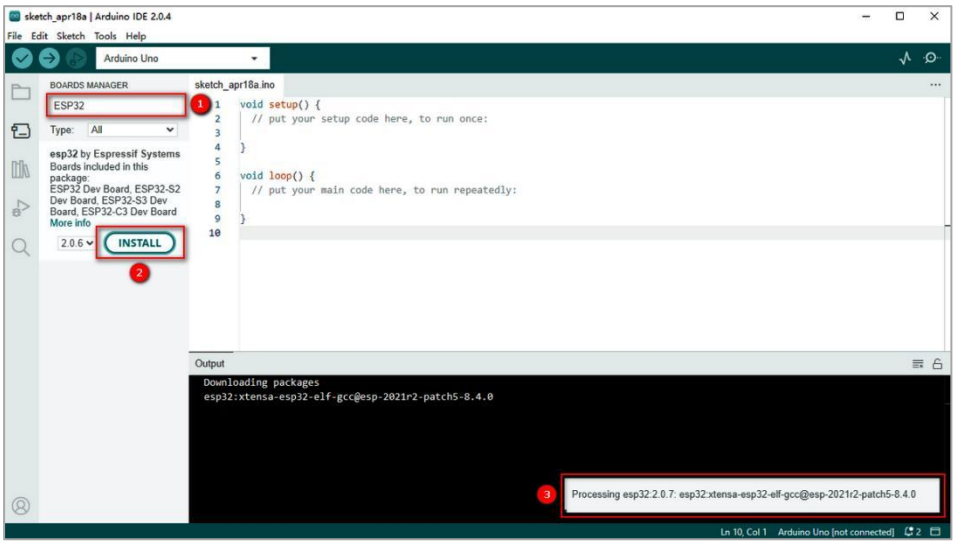
Copy the link of ESP32 board (https://espressif.github.io/arduino-esp32/package_esp32_index.json) into the **Additional boards manager URLs**, and click **OK**.



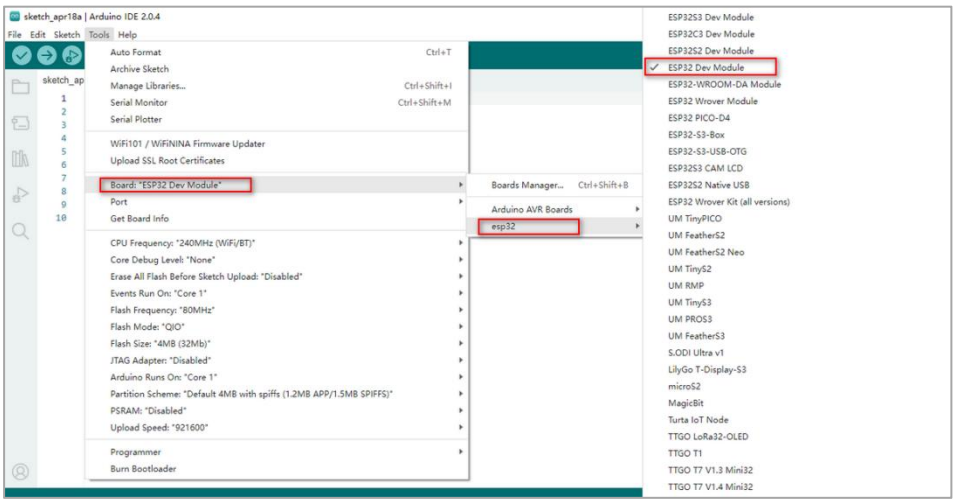
Click the icon of "**Board Manager**" in the upper left corner.

Enter "**esp32**" in the search box. Install the ESP32 environment in Arduino : **2.0.6** (ESP32 by Espressif Systems).

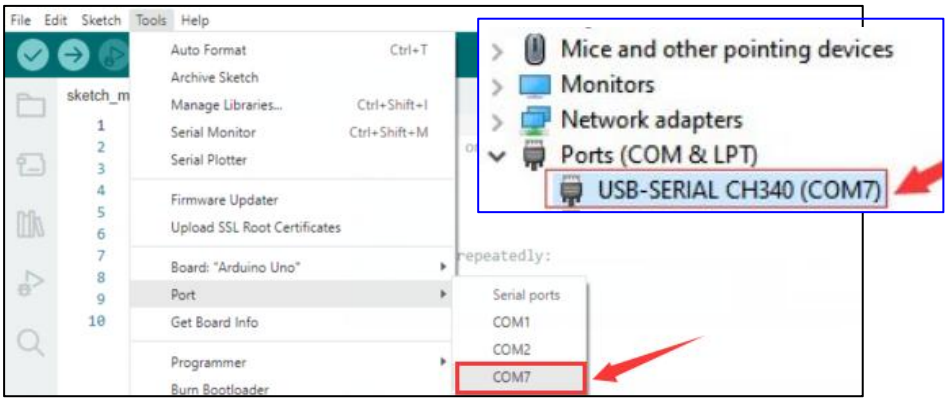
Note: You must select version **2.0.6**, otherwise it may be incompatible with the library files we provide.



Click **Tools> Board > esp32 ,**and choose the **EPS32 Dev Module.**



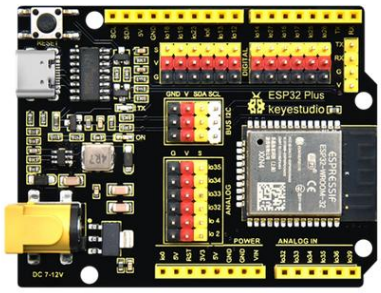
Click **Tools > Port,** then select port "COM-XX" which is shown in Device Manager.



2.Set the Angle of the Servo.

In the next lesson, we will assemble this smart farm kit. Before assembling the servo to the kit, we need to **set its angle to 180°** so that it will work as expected.

You need to prepare:



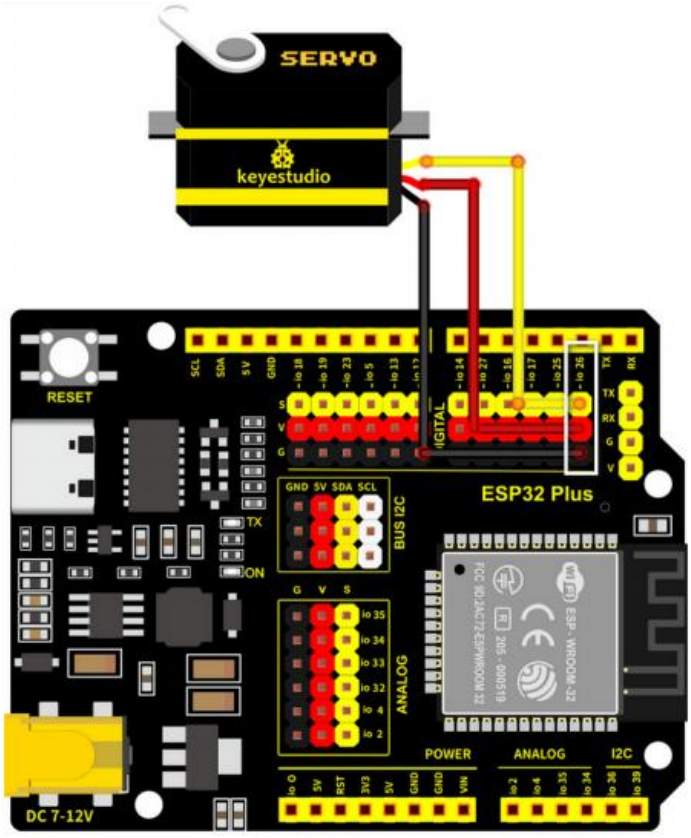
Esp32 plus board*1

USB Cable*1

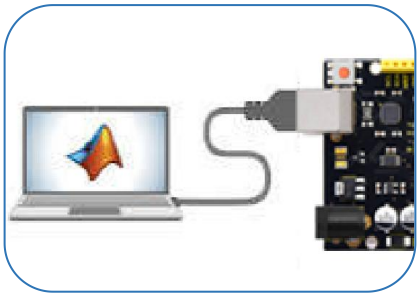
9G 180° Servo*1

1.Connect the servo to the **pin io26** of the Esp32 plus board

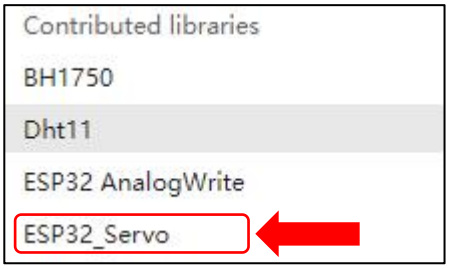
Note: The brown, red and orange wire of the servo are respectively attached to Gnd(G), 5v(V) and **Pin io26.**



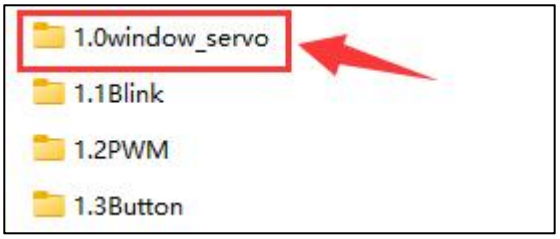
2.Connect the Esp32 plus board to the computer



3.Make sure you have installed the **ESP32_Servo.h** library for the Arduino IDE. If not, please refer to the previous section to install it.

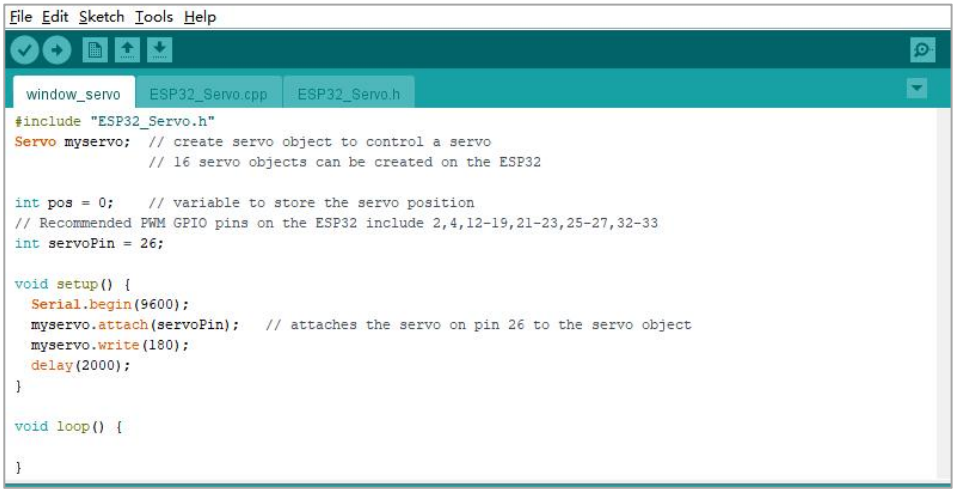


4.Open the **window_servo** code provided in our tutorial package with Arduino IDE.



5.Click on **Tools**, select "**EPS32 Dev Module**" for the board type, and select **COM-XX** for Port as shown in the Device Manager.

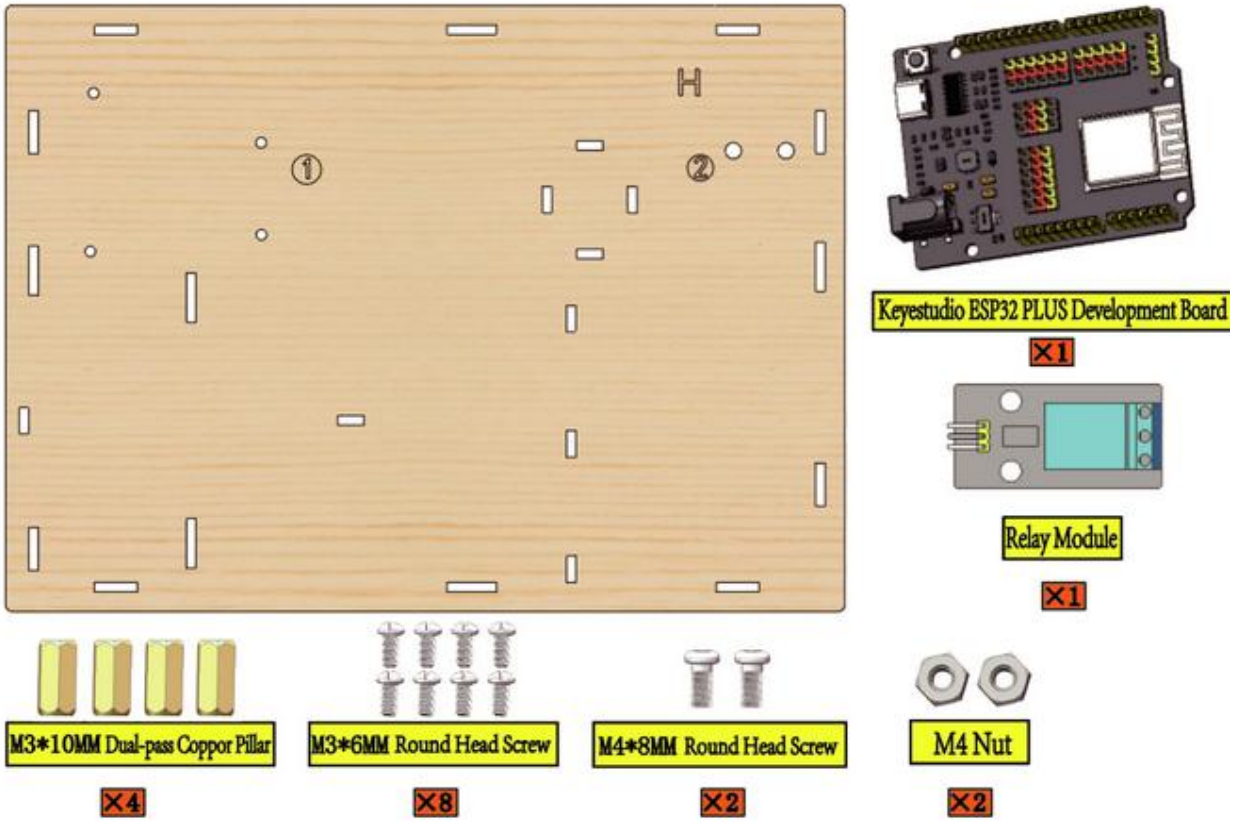
6.Click on upload >>>done uploading, the servo will be set to 180°.



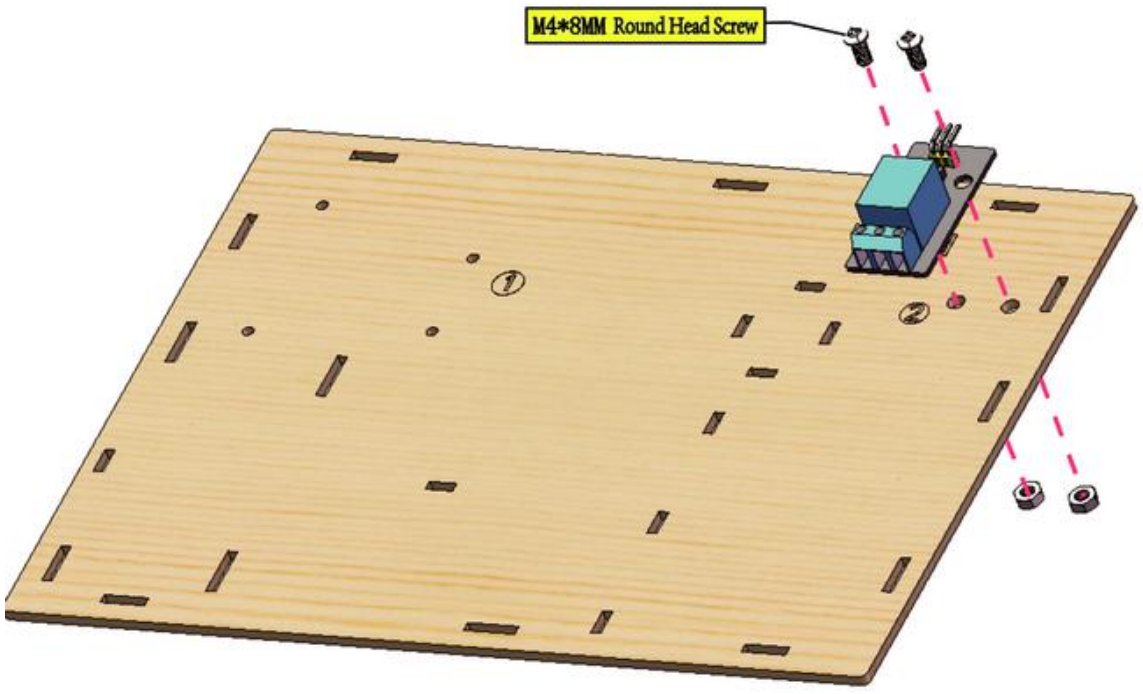
3.Assemble the Smart Farm Kit

Step 1 Install the ESP32 Board and the Relay Module

1.1 Required components



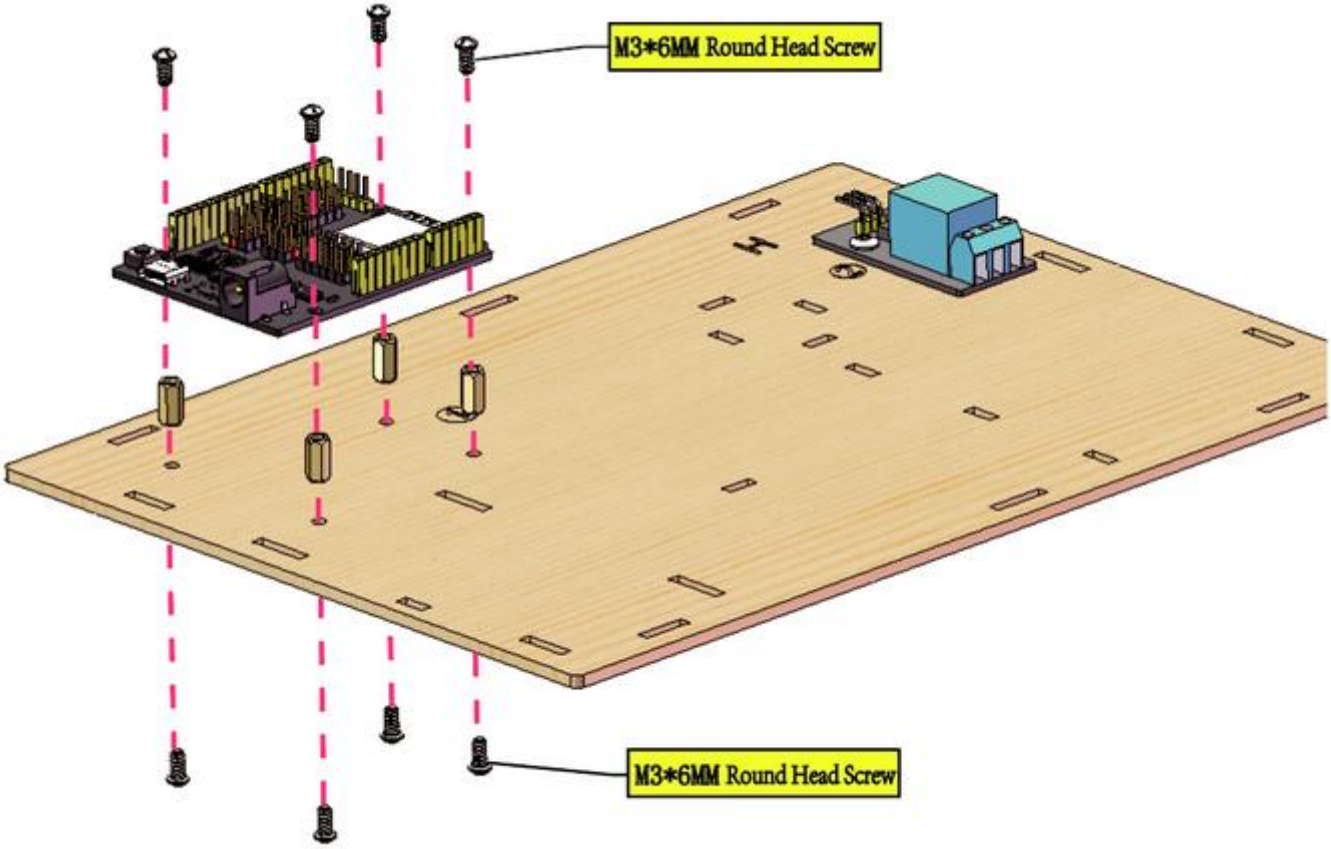
1.2



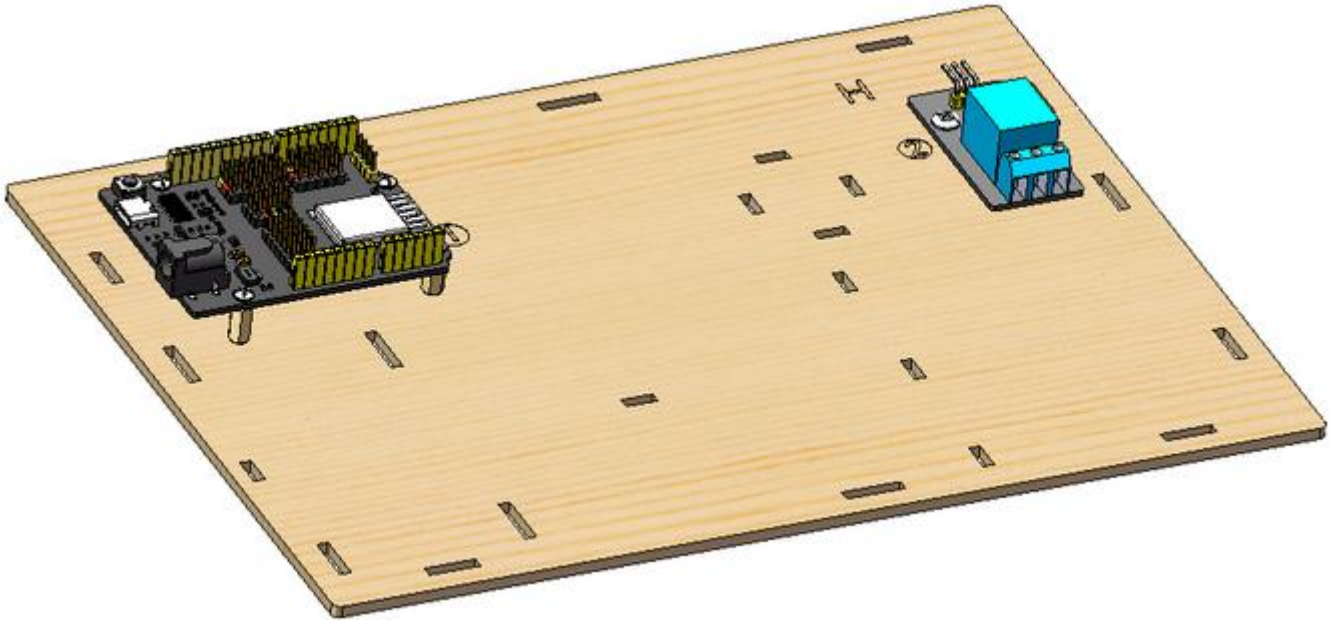
1. 3

---

1. 4

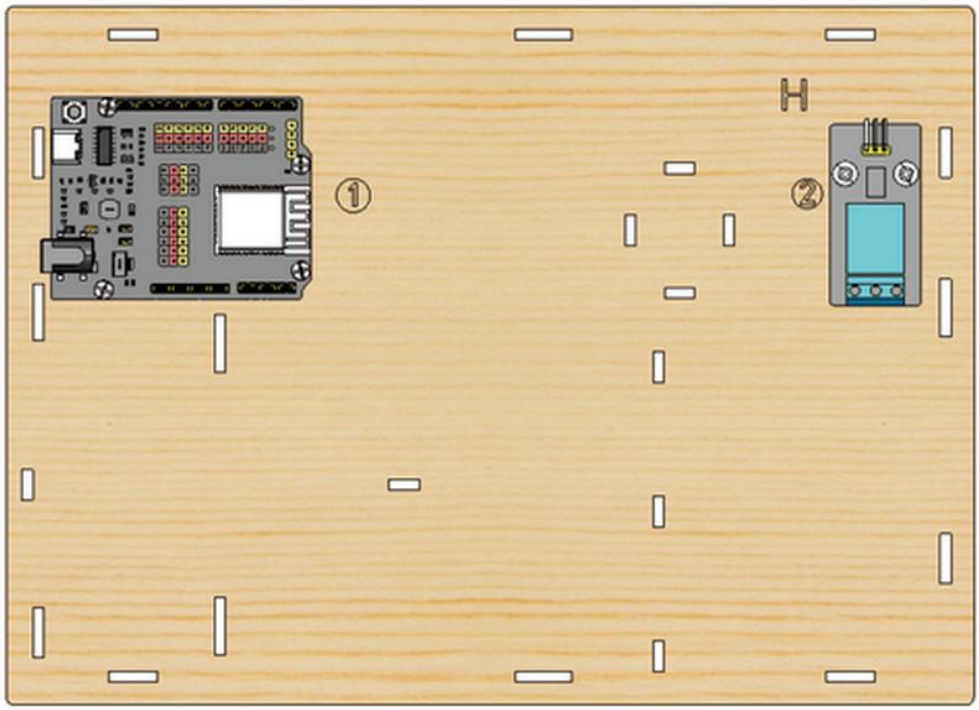


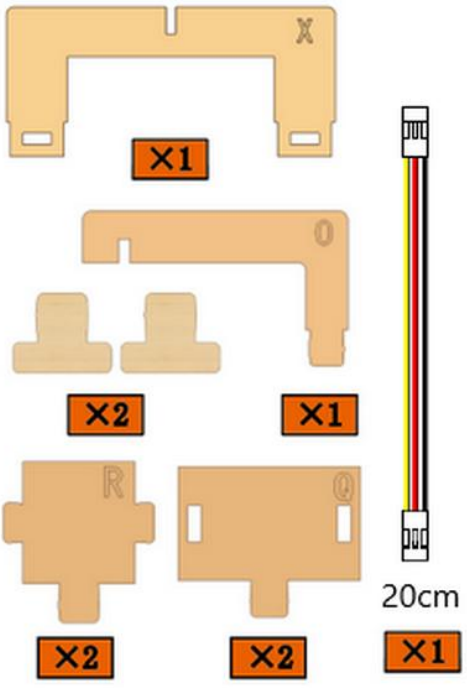
1.5



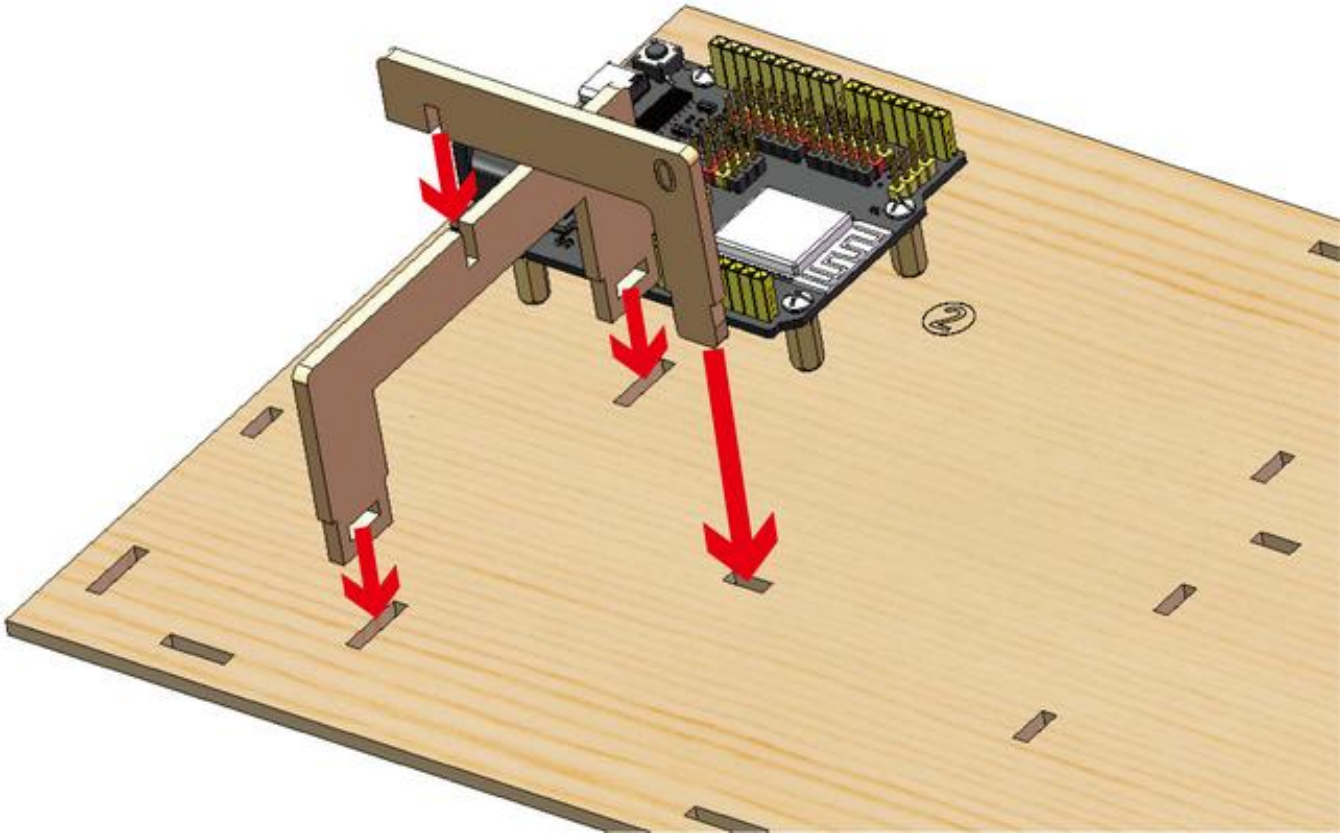
Step 2 Install the Fixing Frame for Battery Case and install the Feeding Cabin, connect the ESP32 board and the Relay Module

2.1 Required components

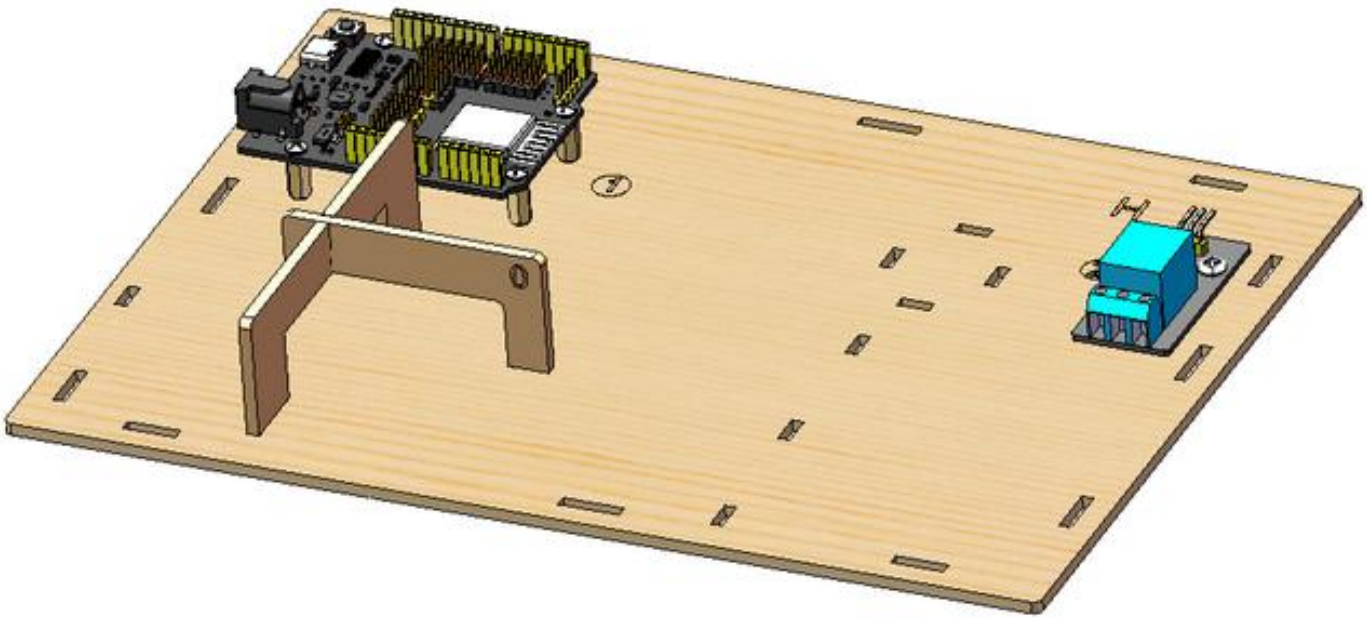




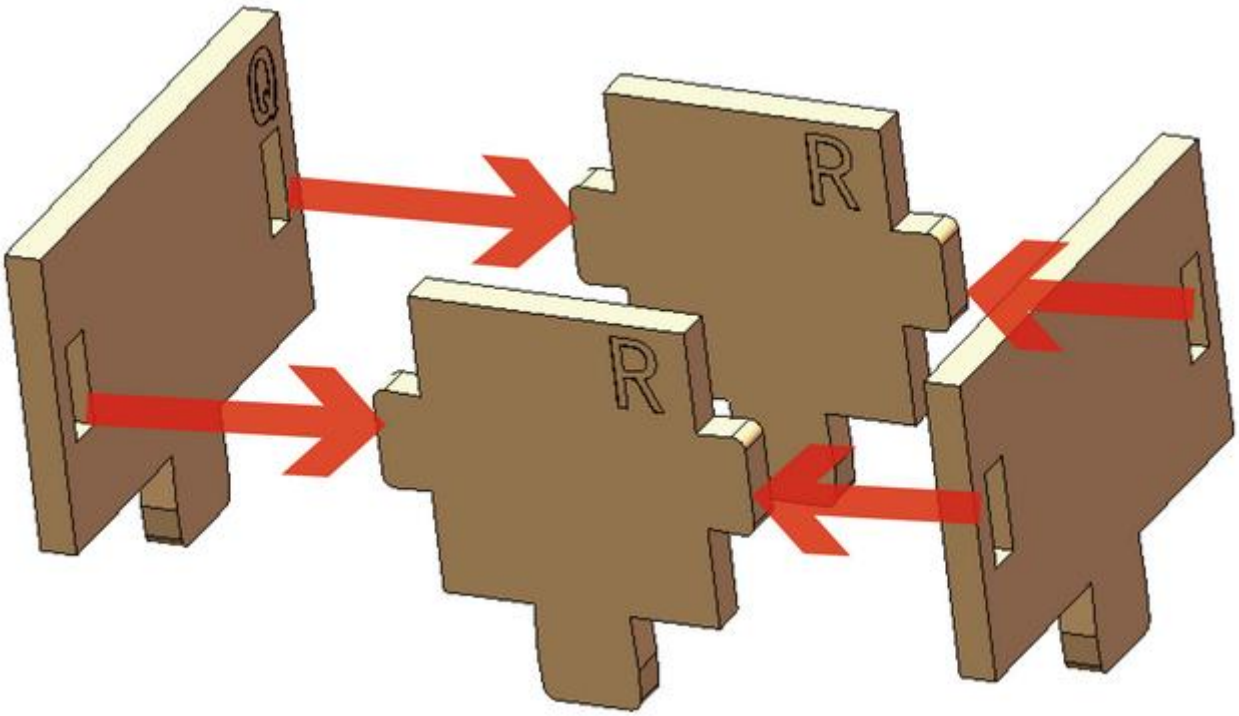
2.2 Assemble the Battery Case Holder



2.3

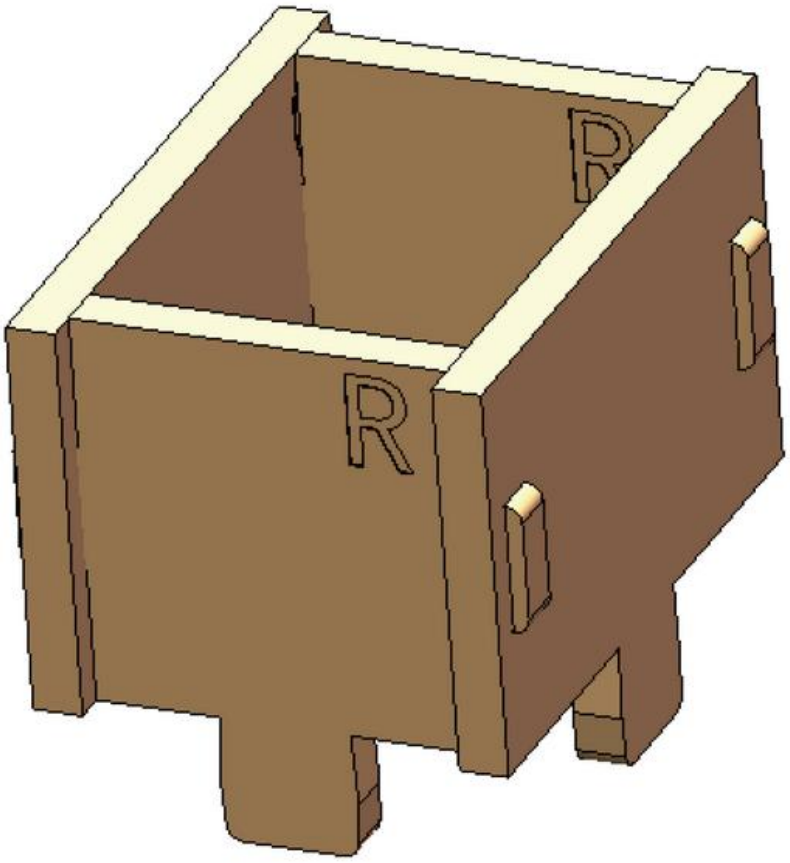


2. 4

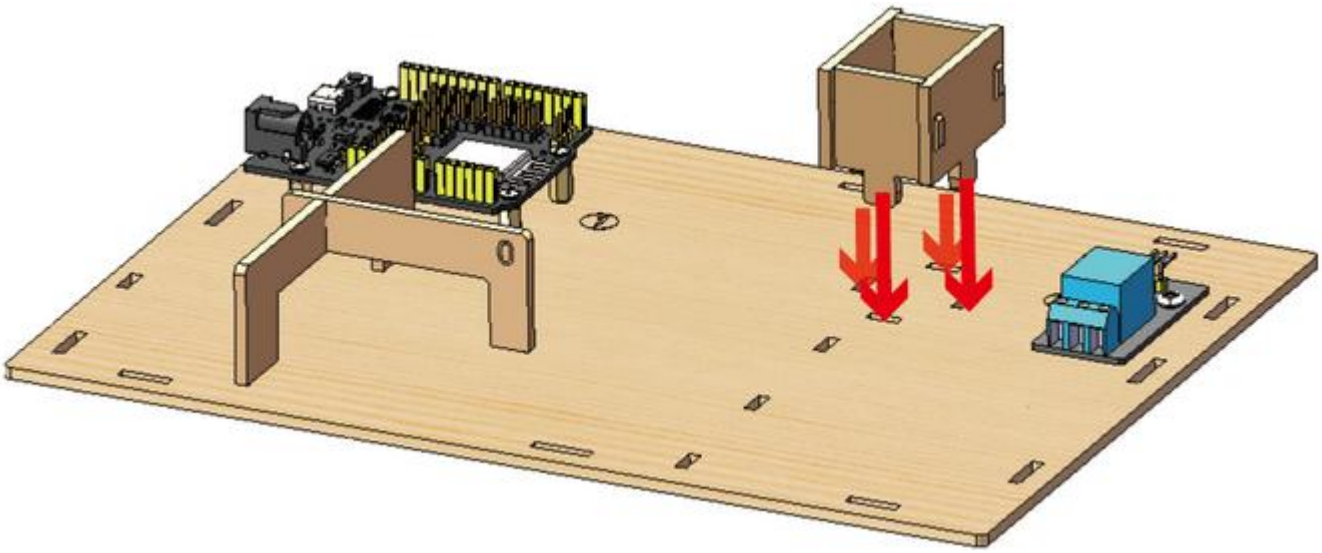


---

2. 5

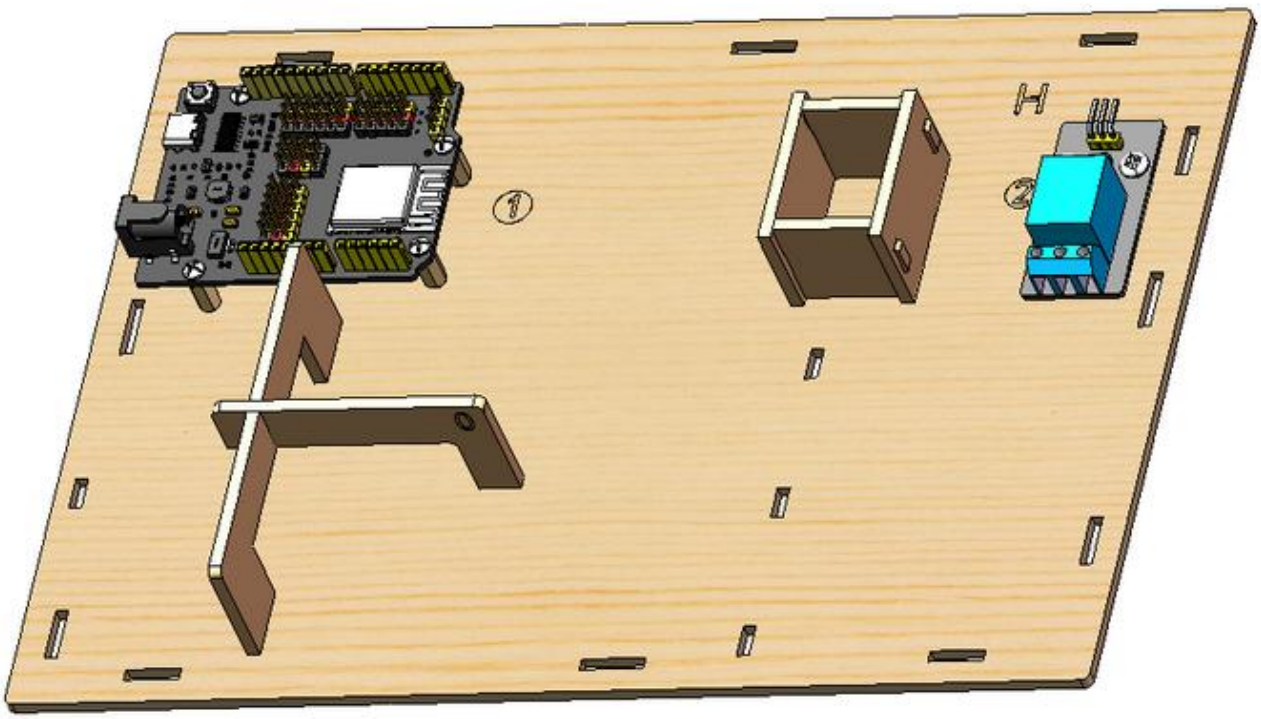


2. 6



---

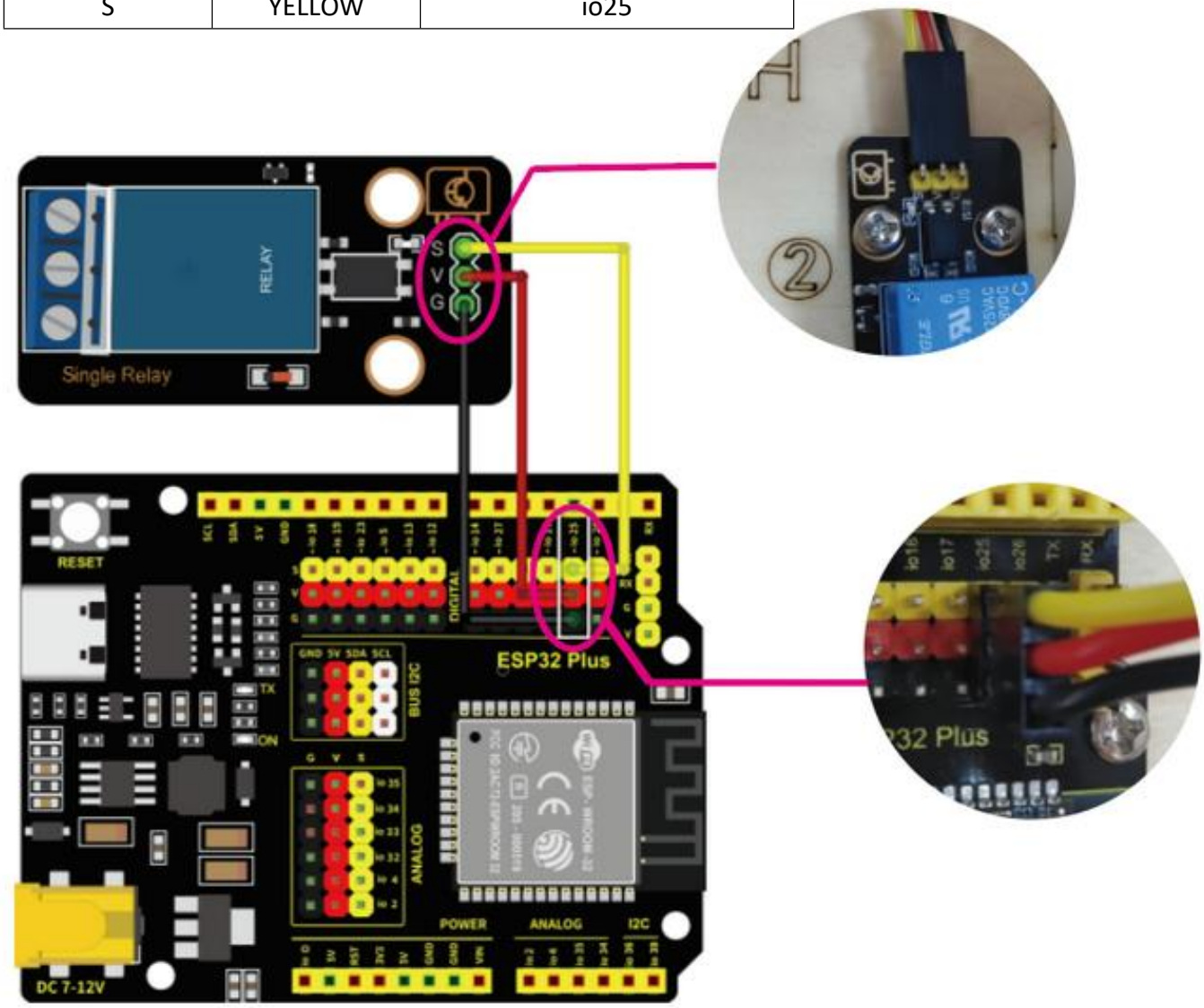
2. 7

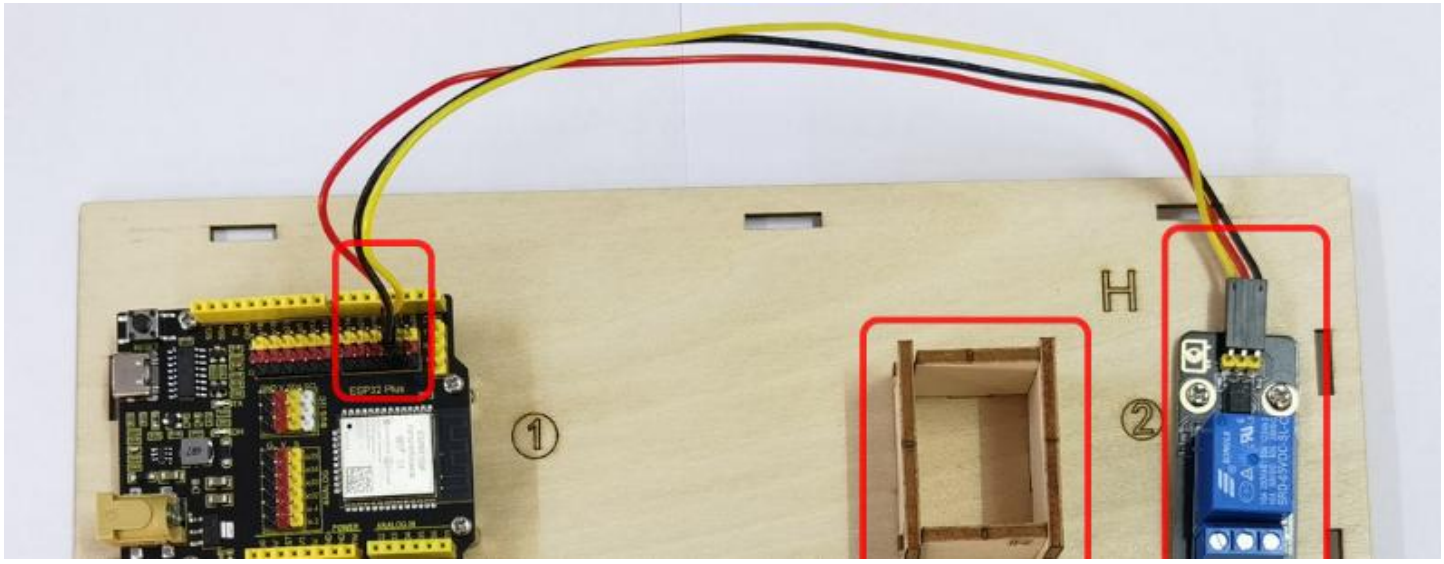


2.8 Connect the ESP32 board and the Relay Module

Module	Wire	Pin of the esp32 board
Relay Module	3PIN 20cm	IO25

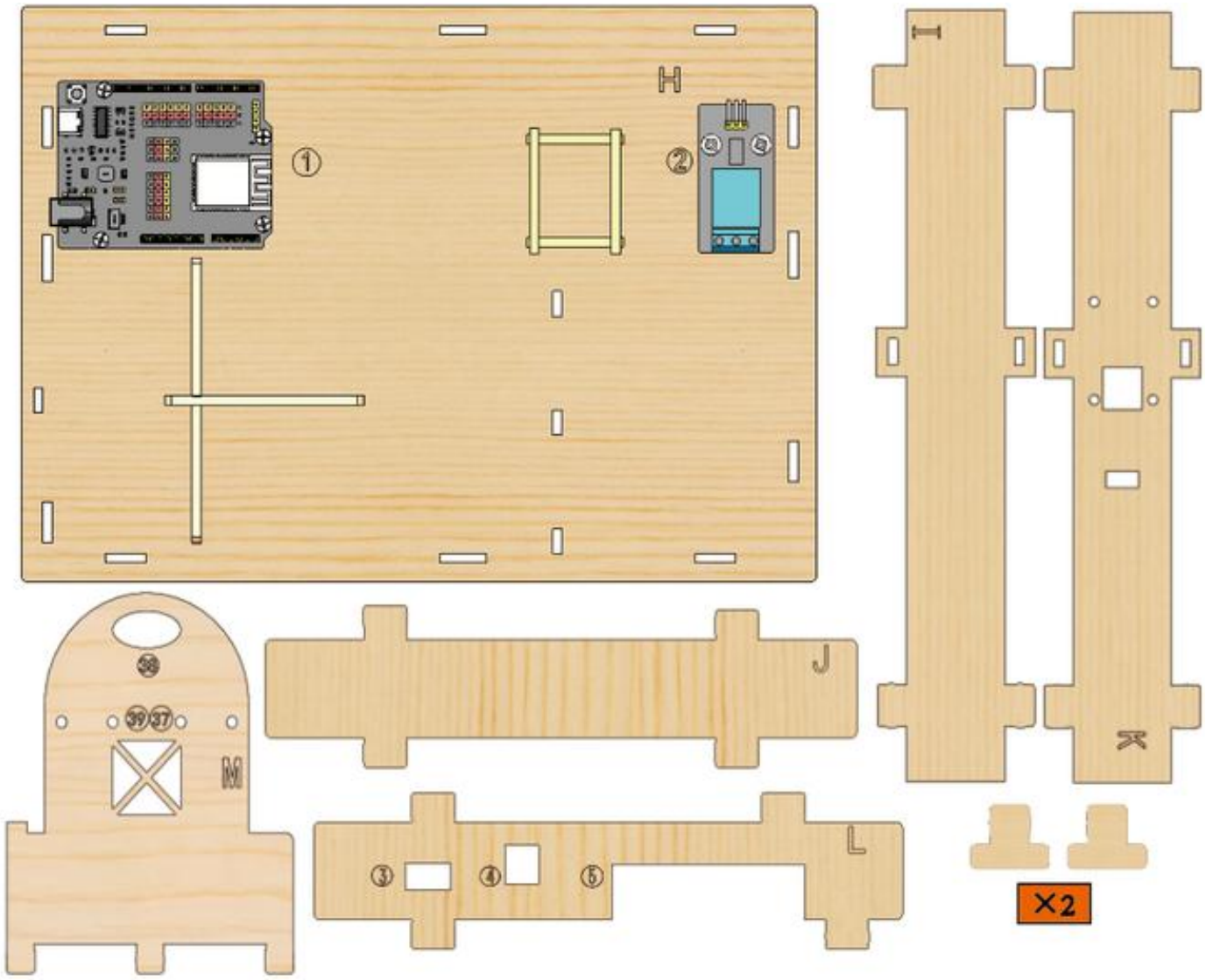
Module Pin	Wire Color	ESP32 Board Pin
V	RED	V
G	BLACK	G
S	YELLOW	io25



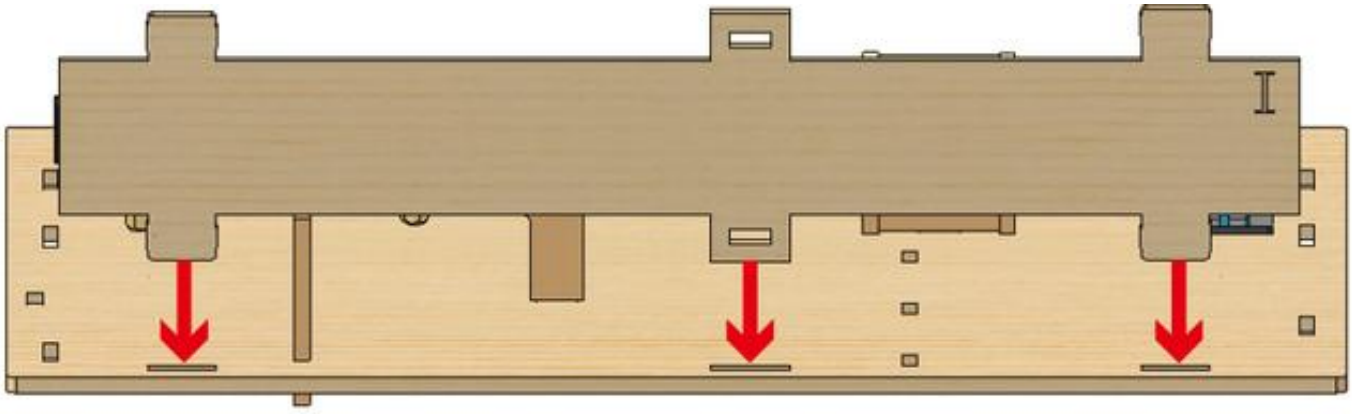


Step 3 Install the Substructure of the house

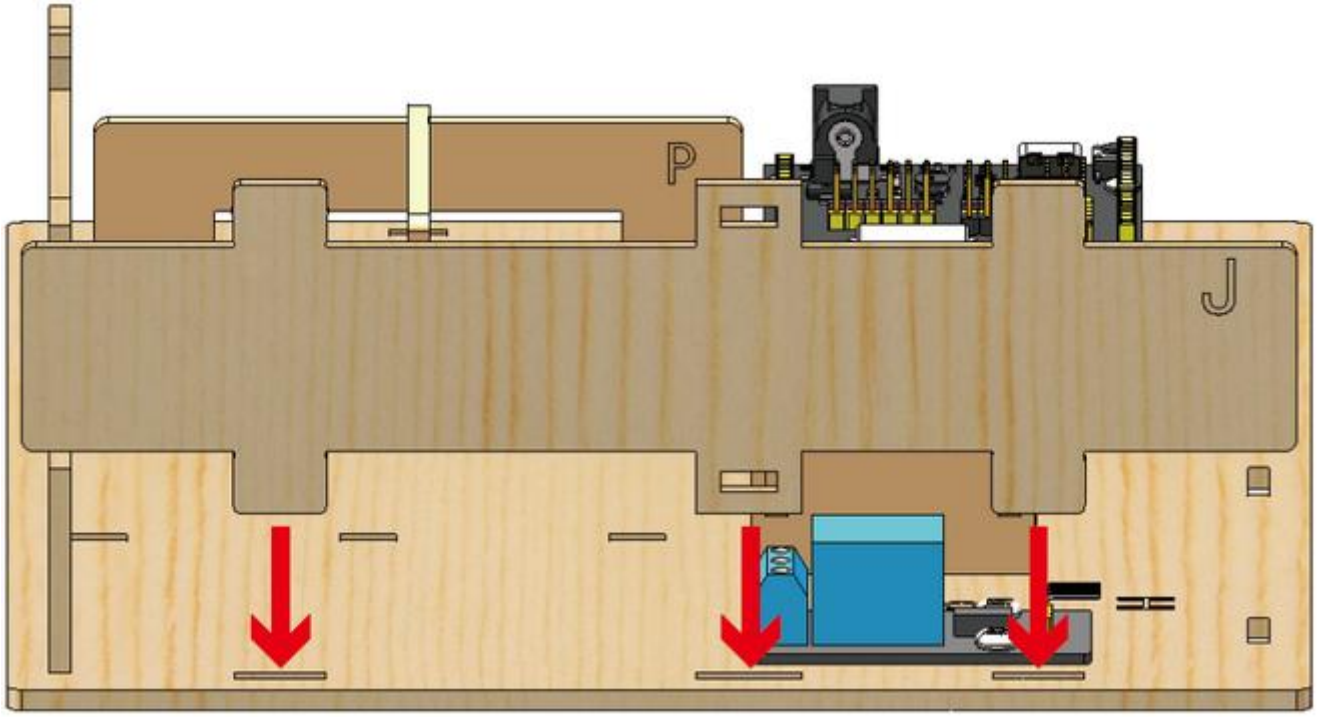
3.1 Required components



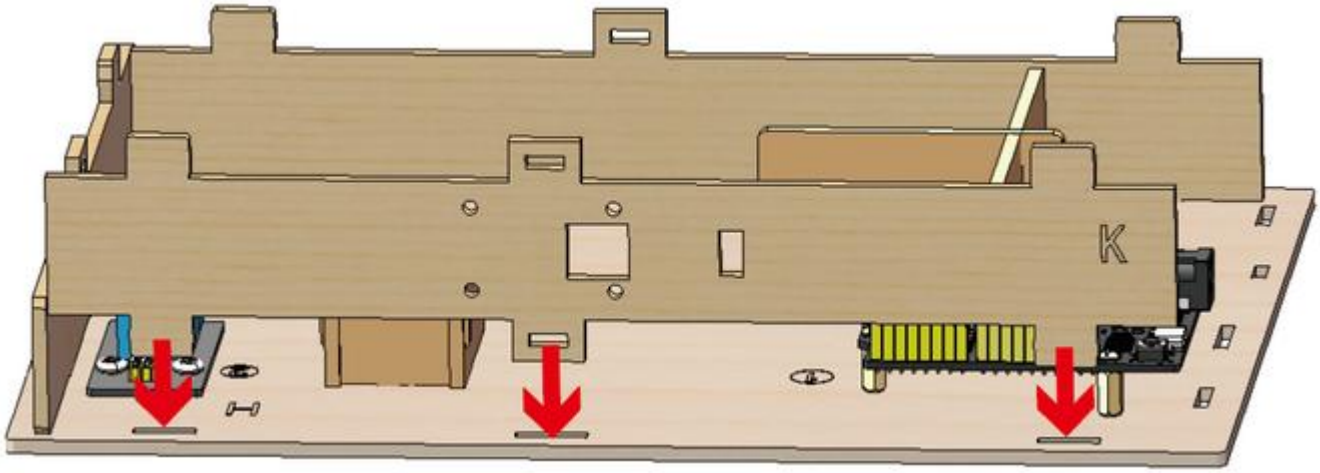
3.2



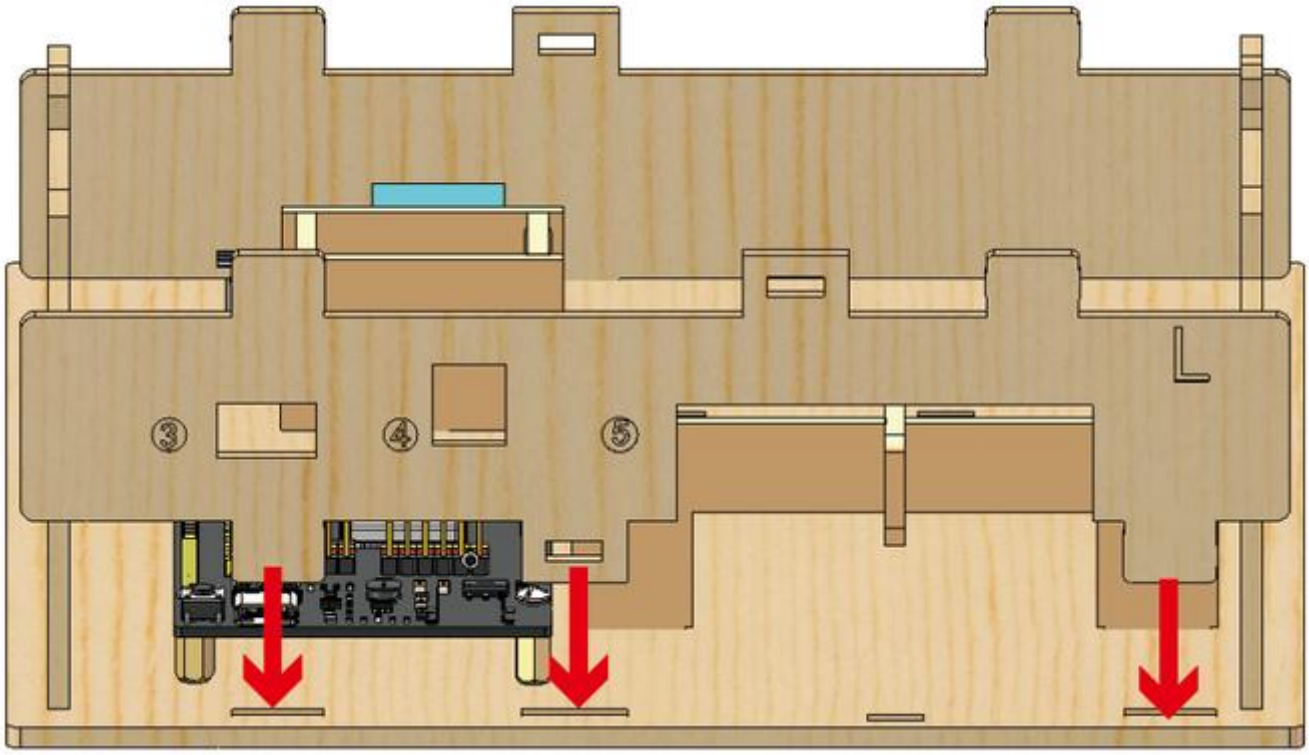
3.3



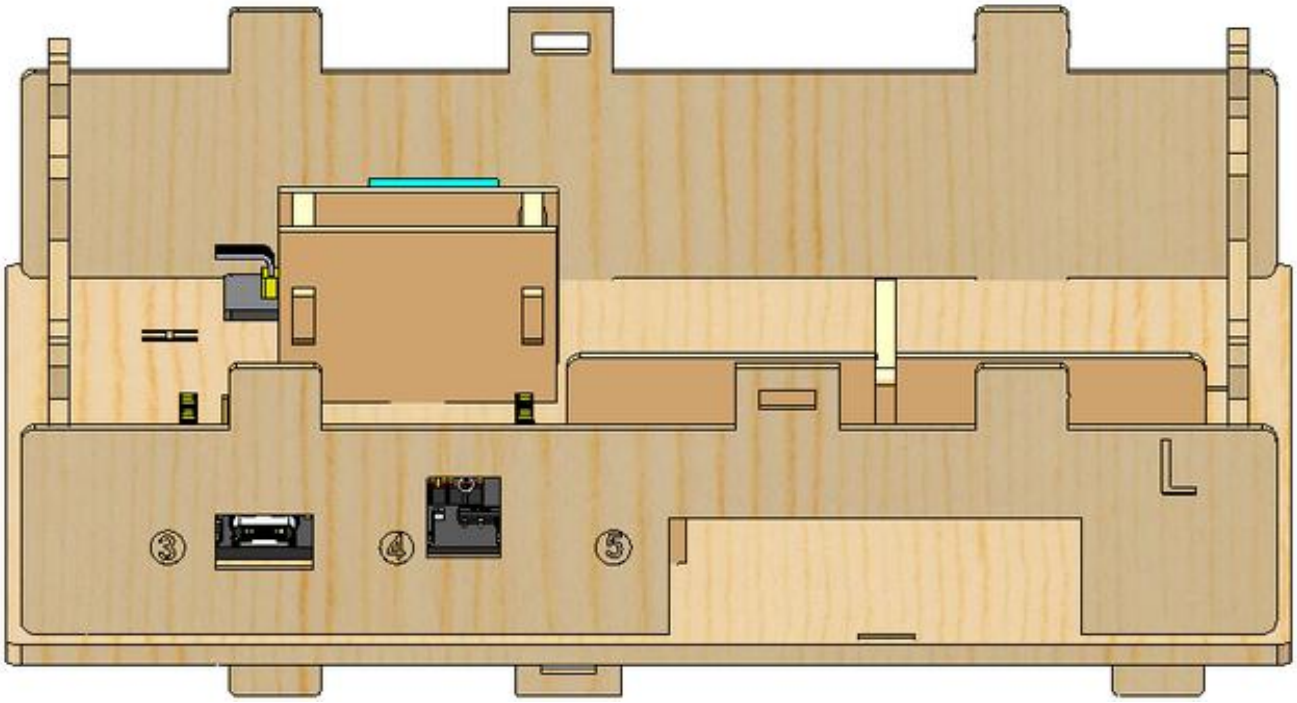
3.4



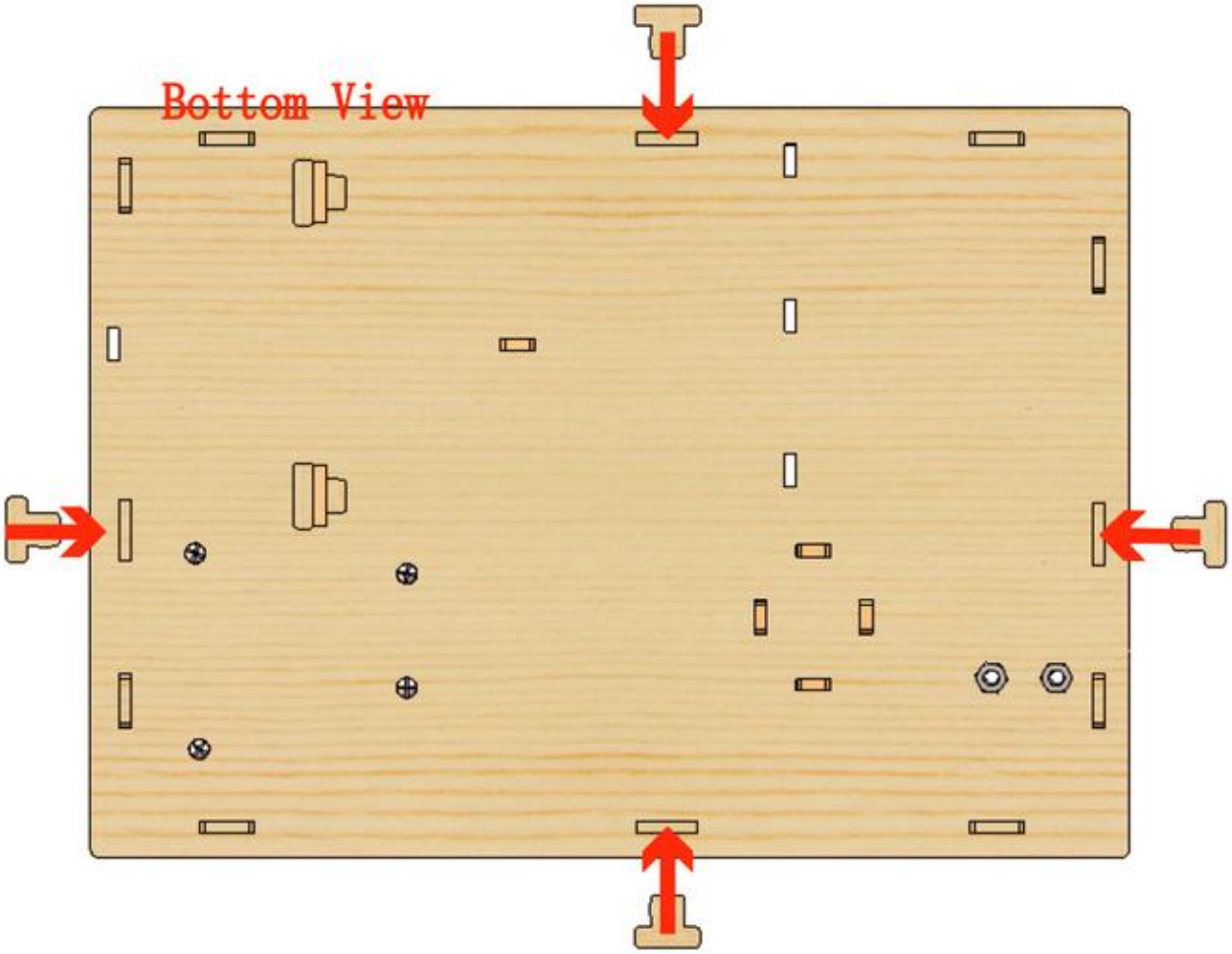
3.5



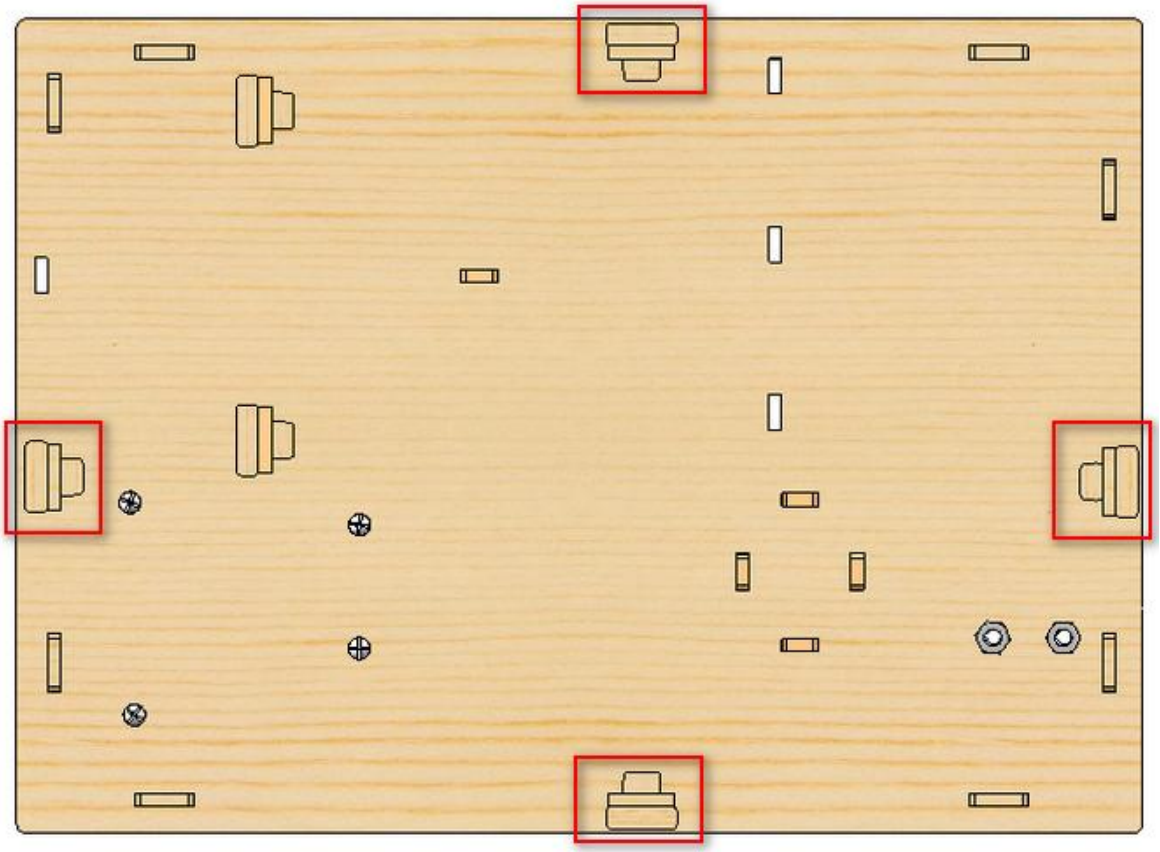
3.6



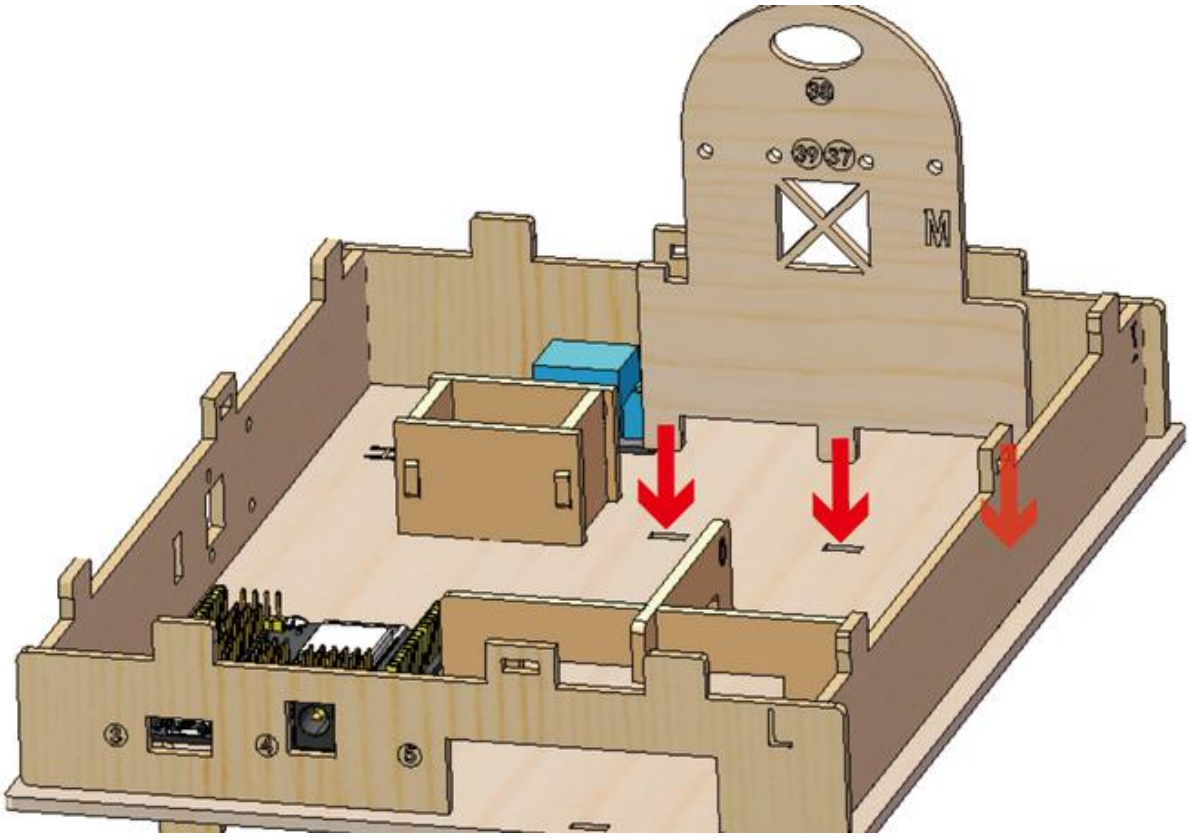
3.7



3.8

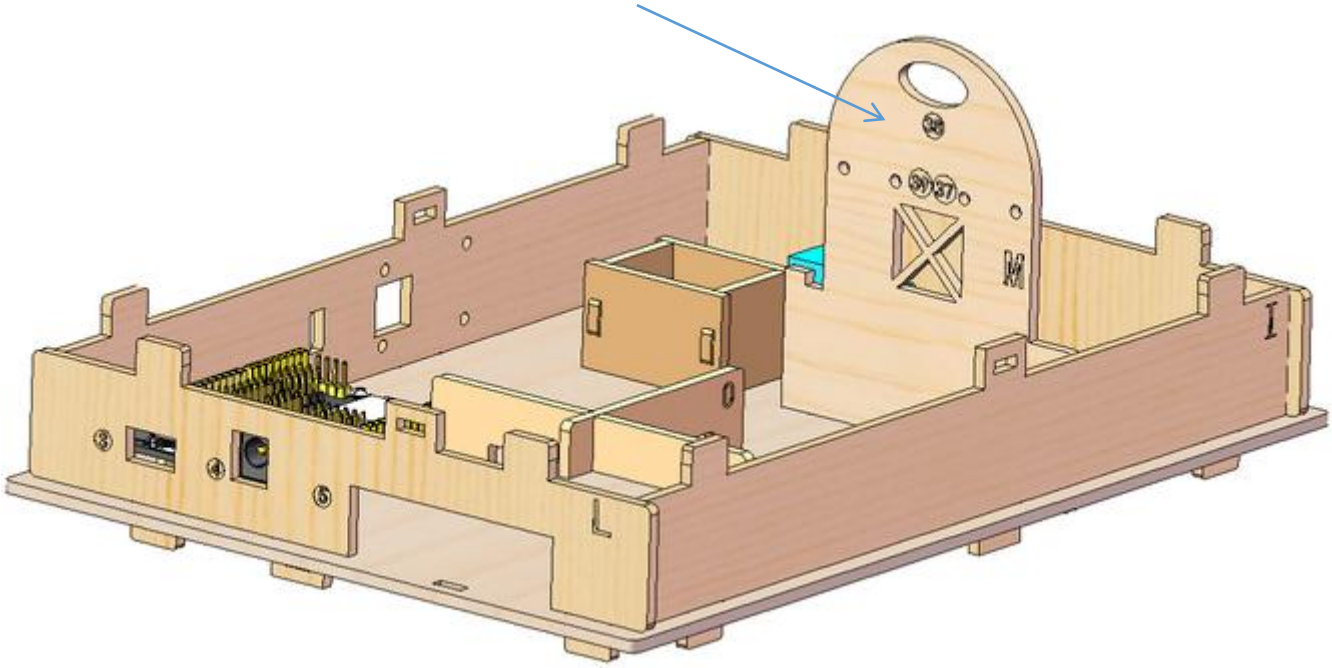


3.9



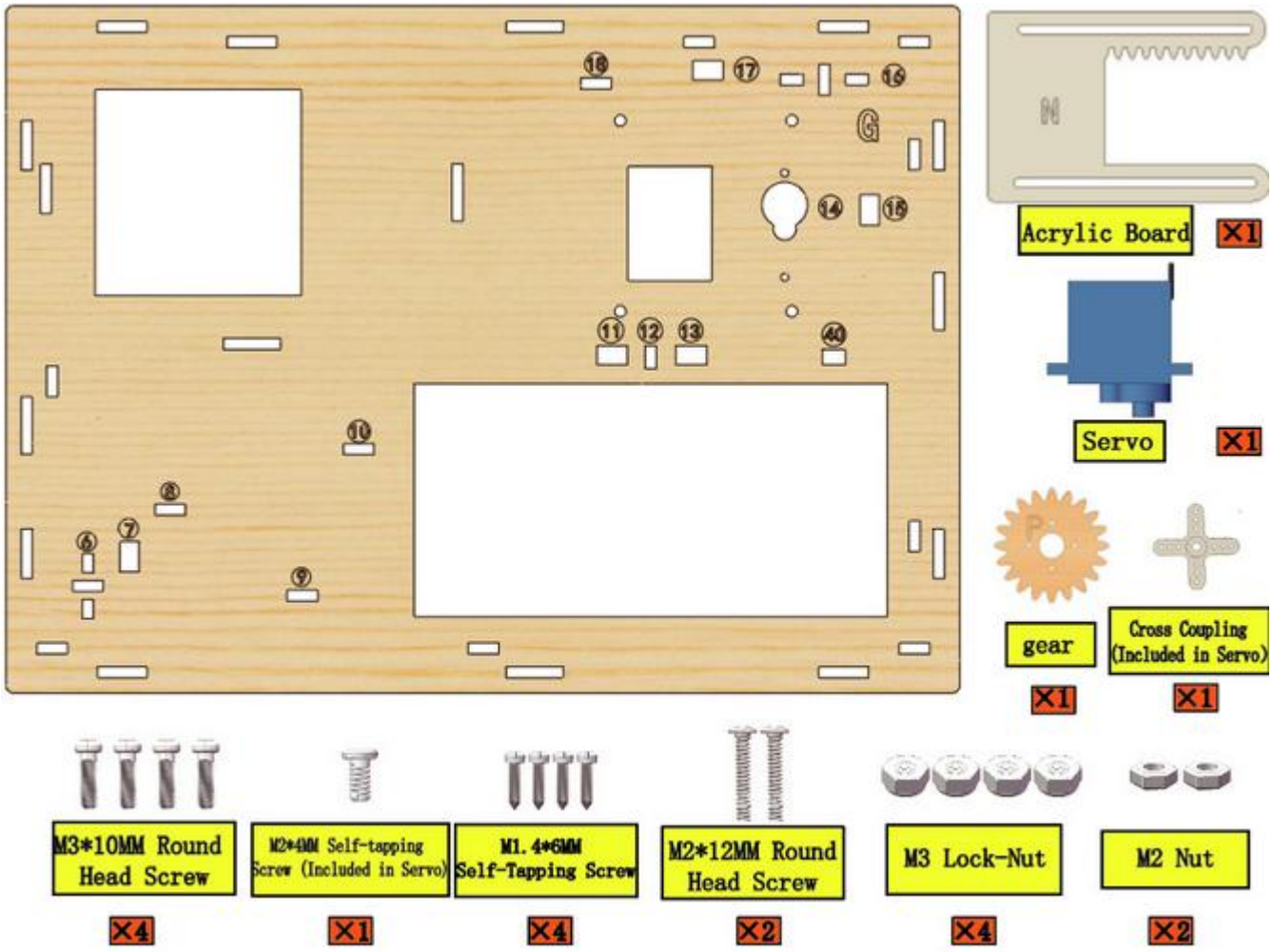
3.10

Note the installation direction of the board, with M facing us.

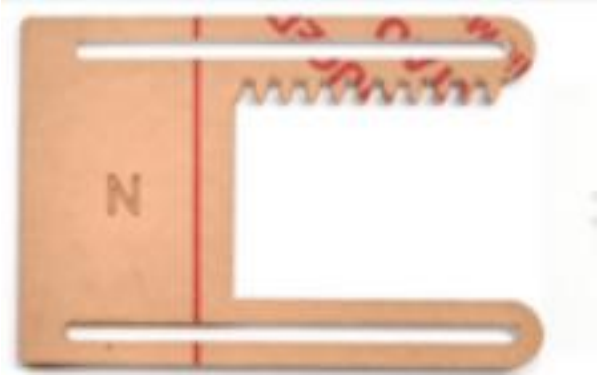


Step 4 Install the Door of the Feeding Cabin

4.1 Required components

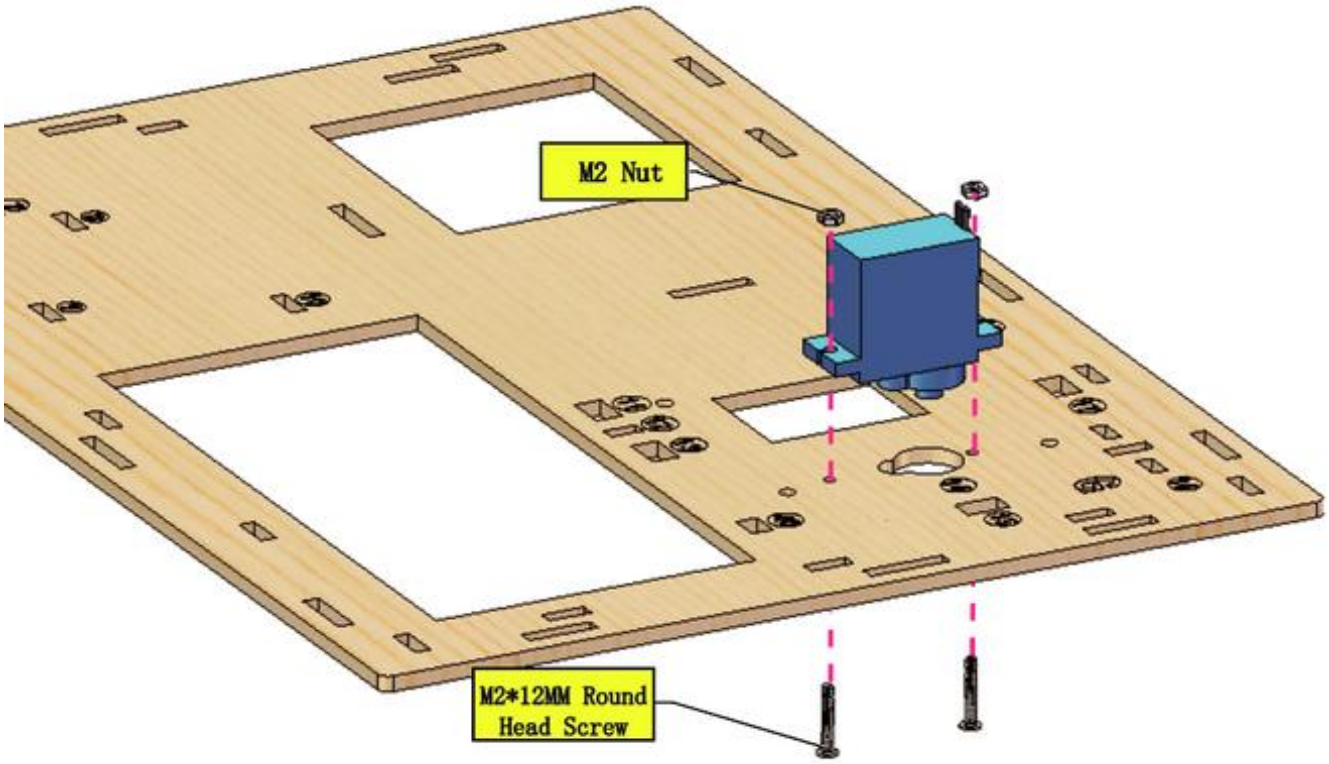


4.2



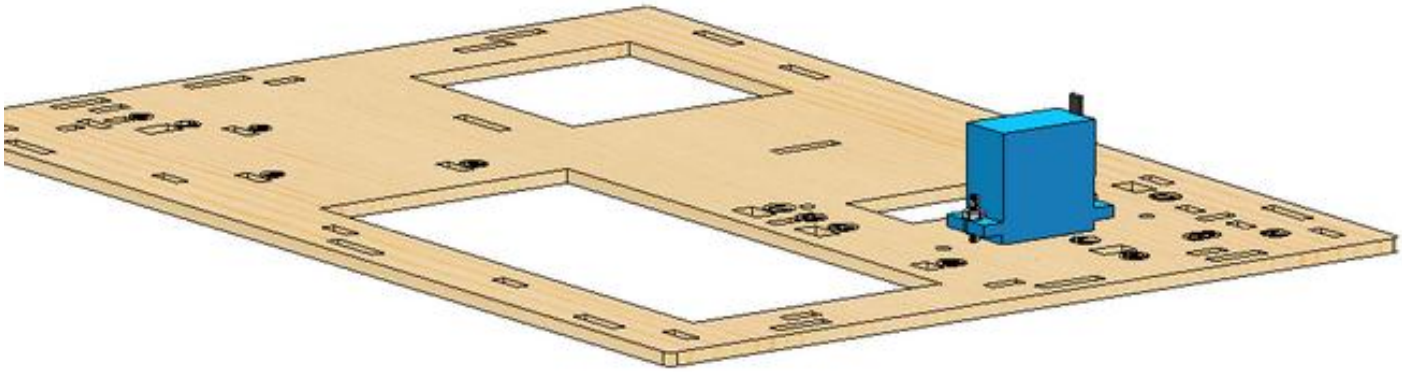
The acrylic sheet is packed separately, and it is recommended that you tear off its protective film to reduce the friction when it moves as a door.

4. 3

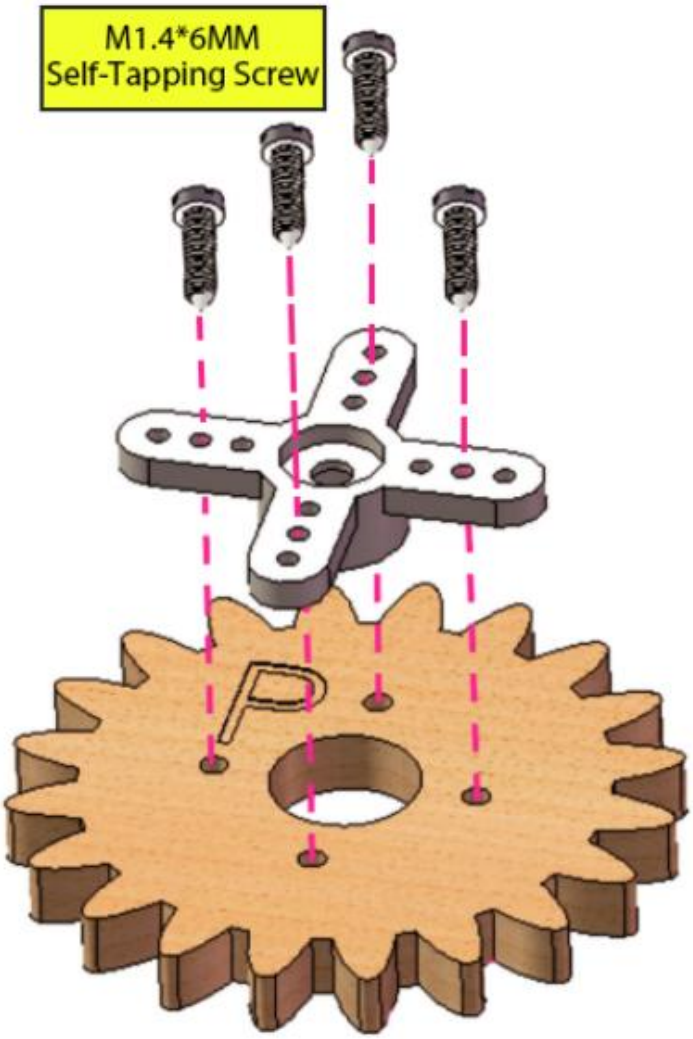


---

4. 4 **Note: The screws need to be tightened to keep the servo stable, otherwise the door may get stuck**

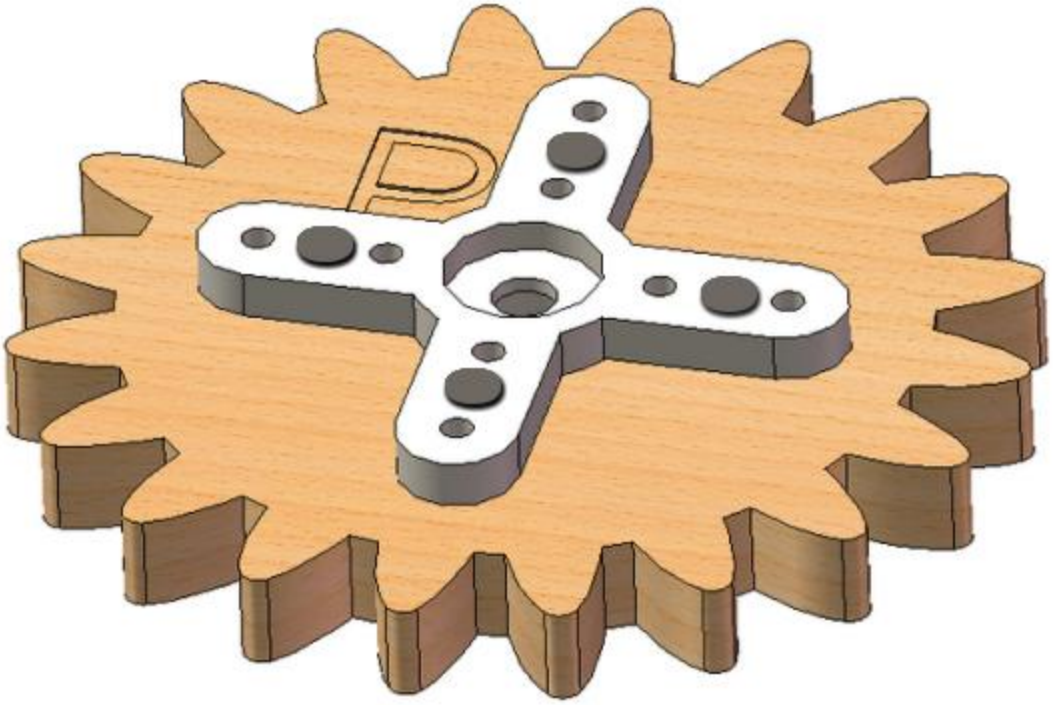


4. 5

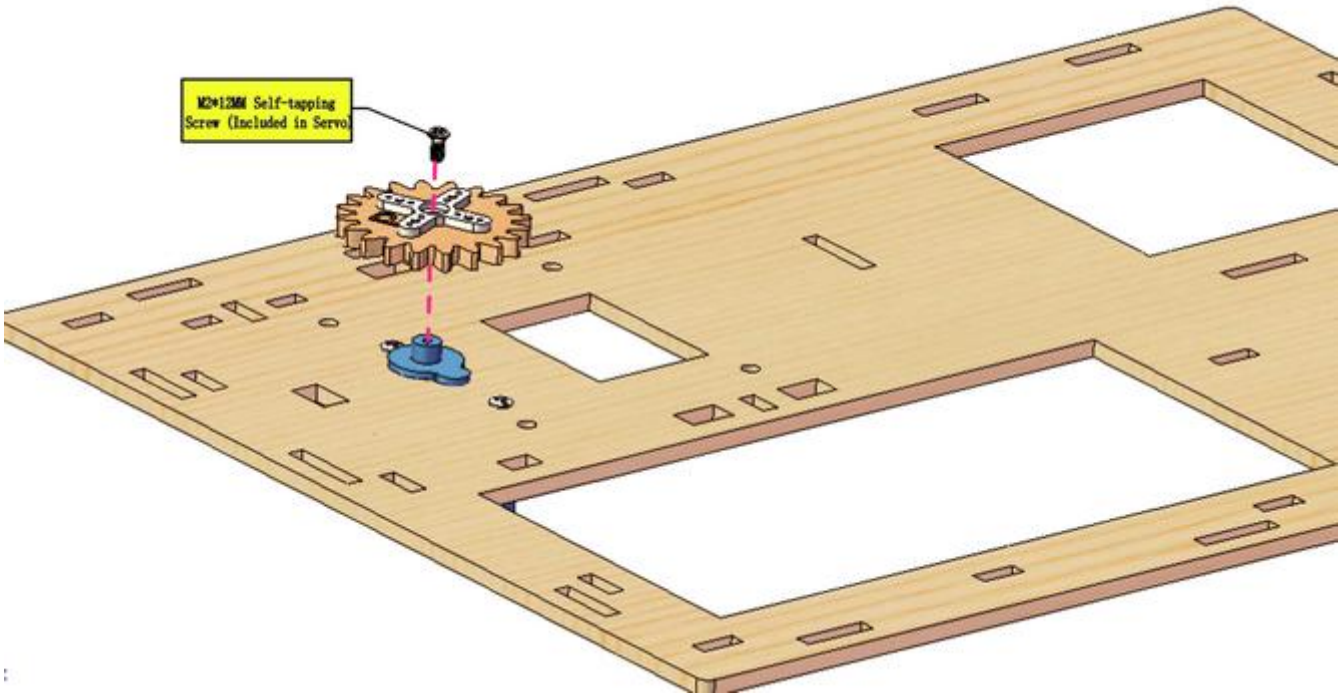


---

4. 6

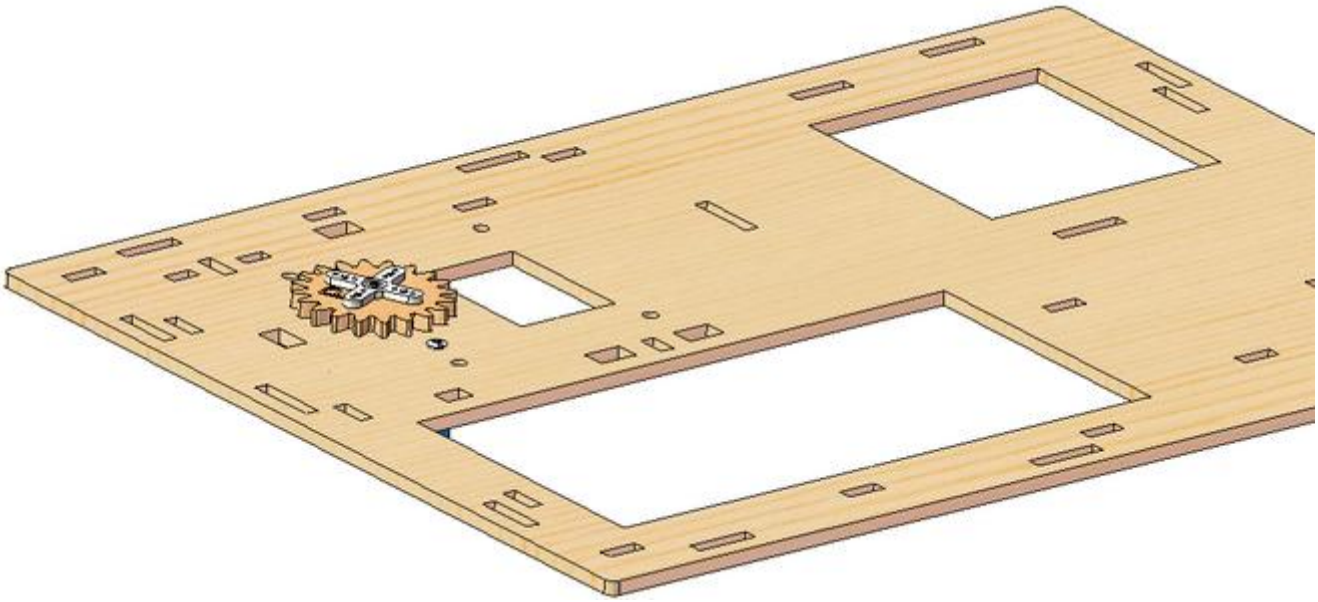


4. 7



---

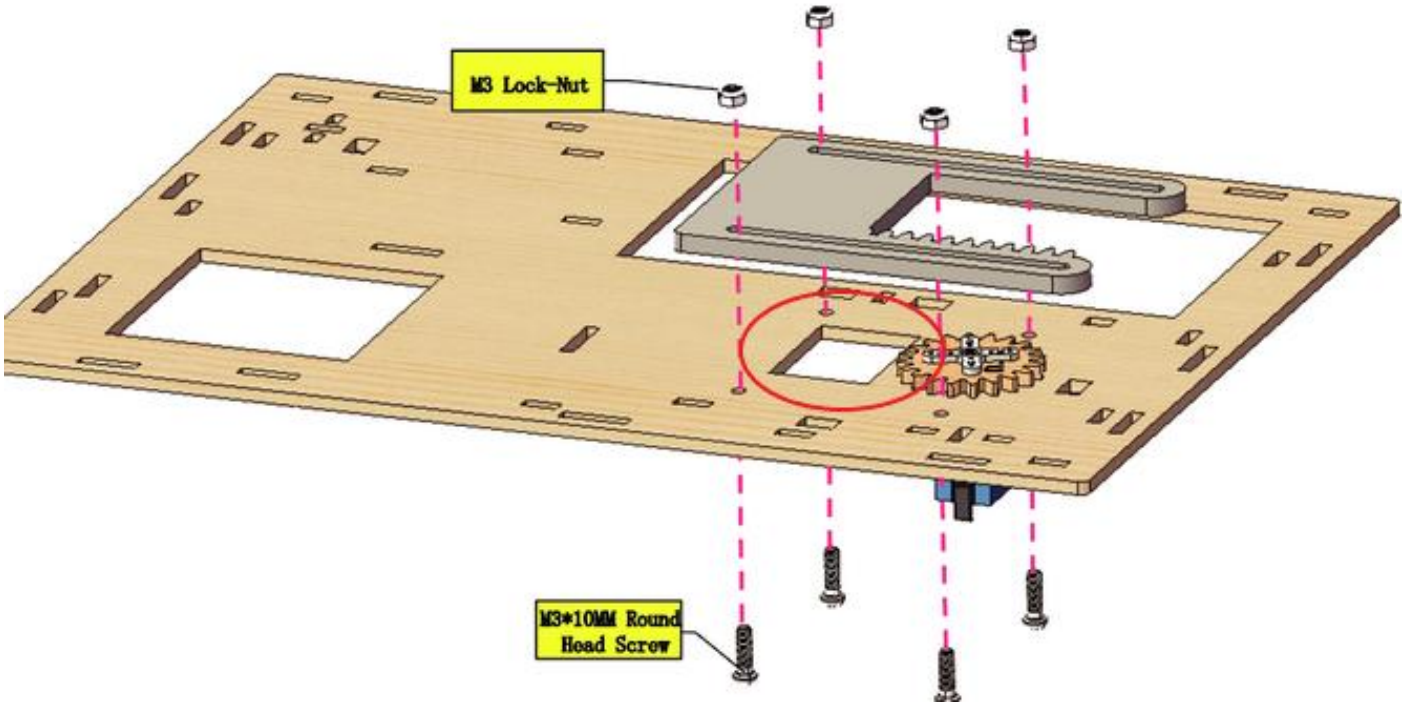
4. 8 Do not turn the gear after it is installed on the servo. If you have already turned the gear you will need to readjust the servo angle to **180**°.



4. 9

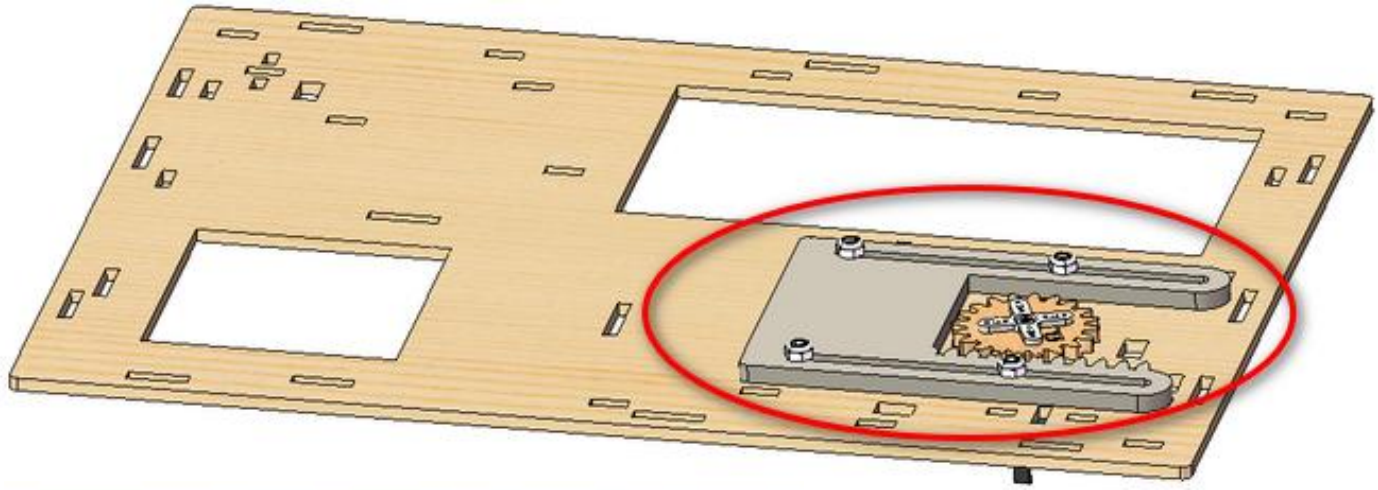
Note:

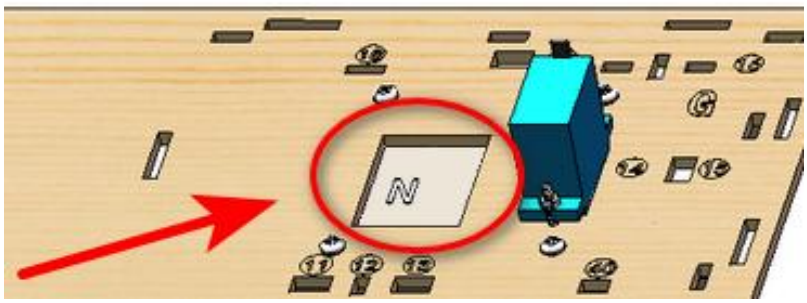
Cover the hole below with the door. Keep the door closed in its initial state.

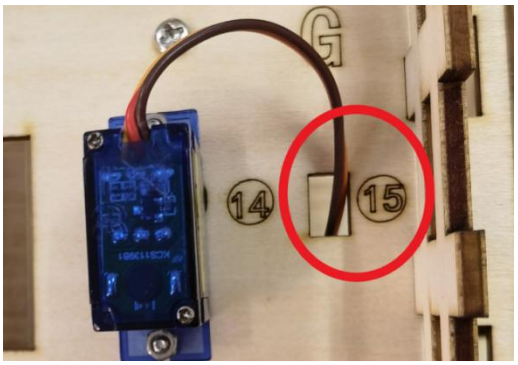


---

4. 10

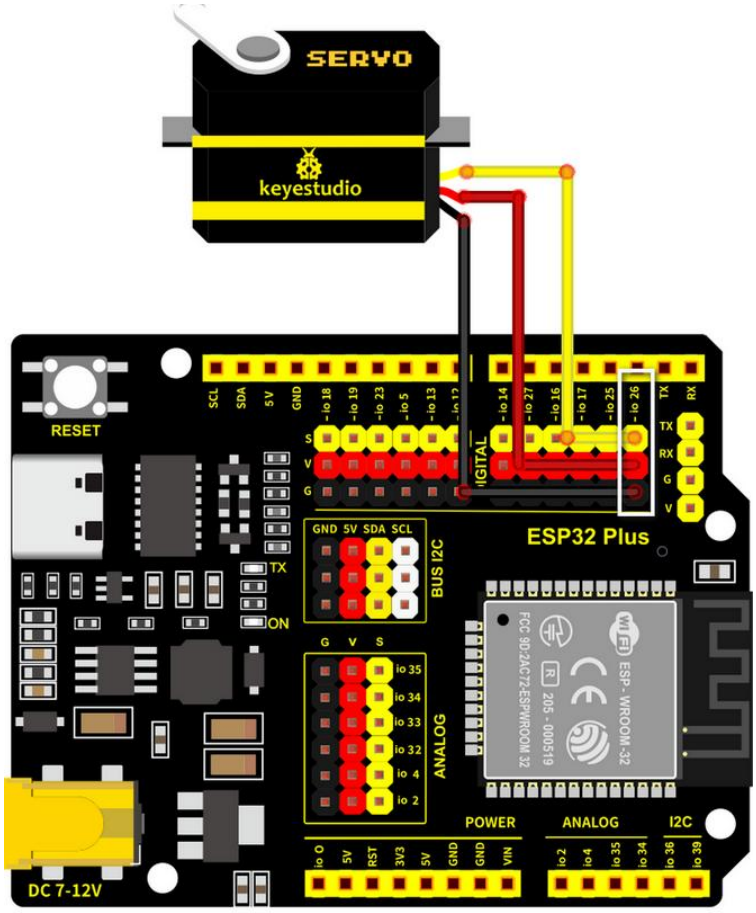






4. 11 Test the door !

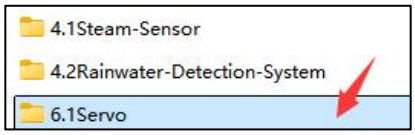
1) Connect Servo to pin IO26 of the ESP32 board. Connect yellow to S, red to V, black to G.

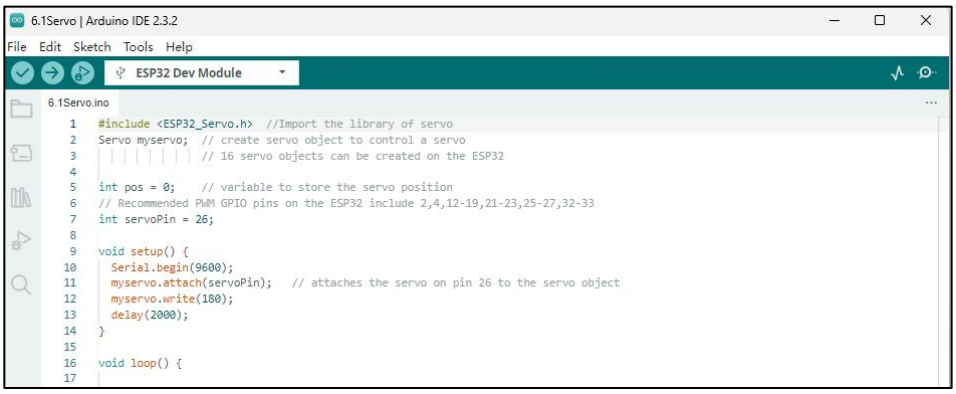


2) Connect 6 AA batteries to the DC 7-12V port of ESP32 board. (Batteries not included in the kit)

3) Upload the Test code

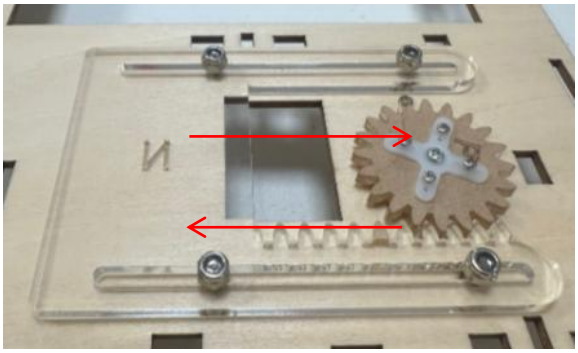
1. Connect the ESP32 board to the computer with the usb cable. Open the INO file inside the **6.1Servo** folder with Arduino IDE.





2. Click on Tools, select "ESP32 Dev Module" for the board type in the drop-down menu bar, and select COM-XX for Port (According to the port assigned by your computer in the device manager)

3. Please make sure you have uploaded the **ESP32_Servo** library and then upload the code. The door of the feeding cabin will open and close slowly.



NOTE: After uploading the code, if the door cannot be opened and closed and the servo is hot, please turn off the power immediately.

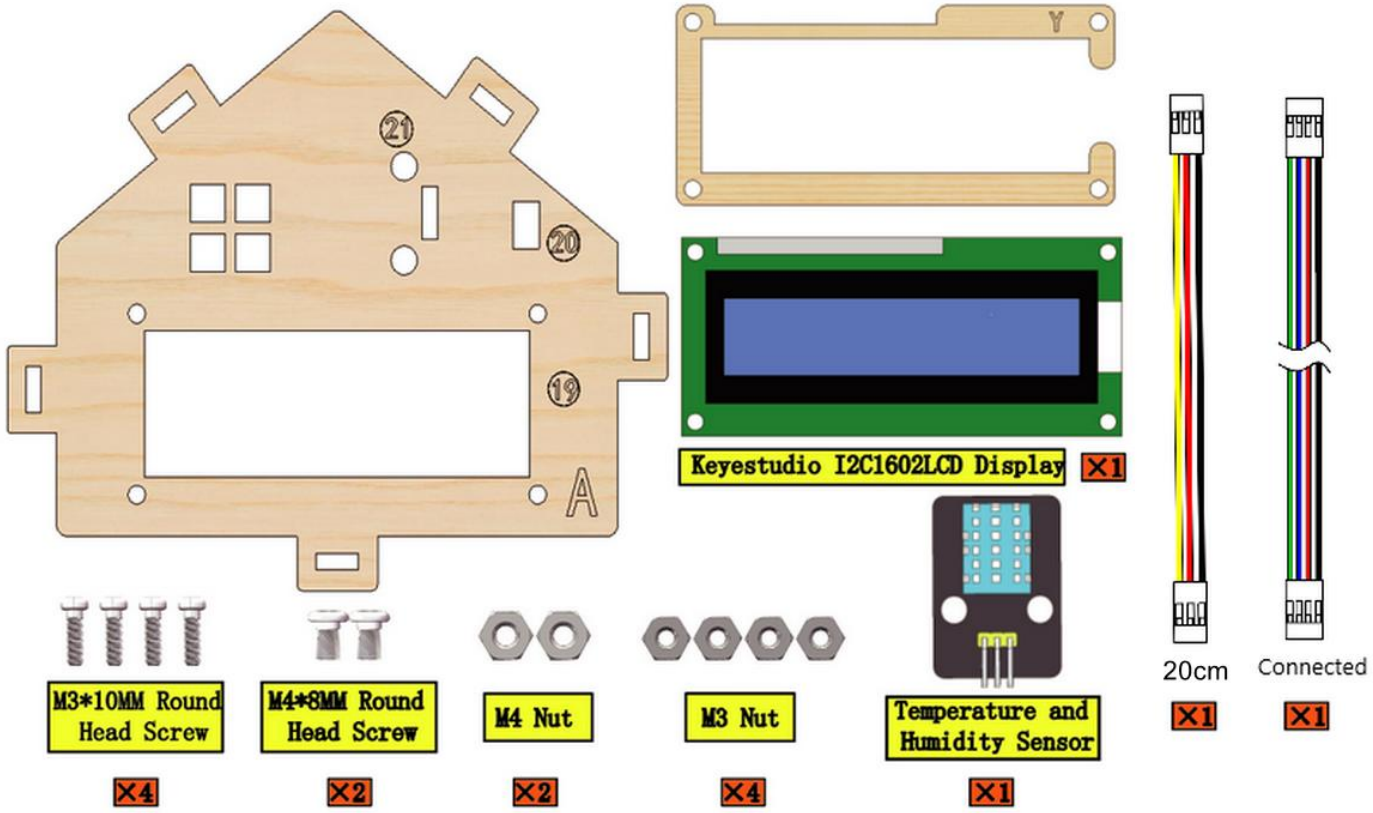
And check:

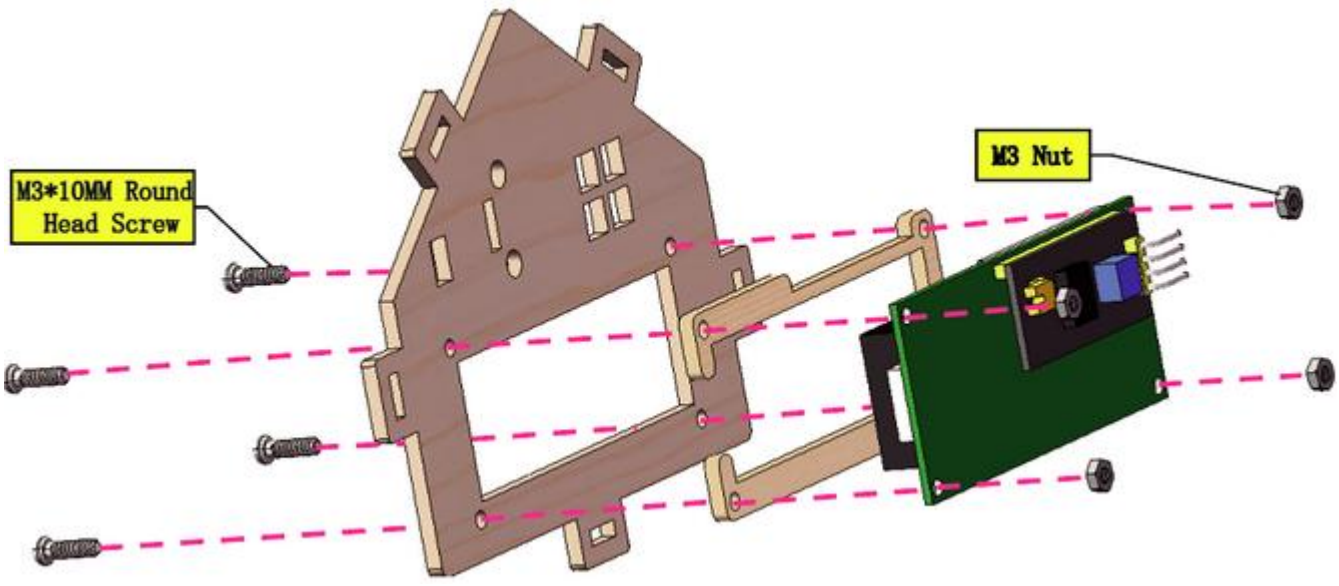
1. Whether the plastic door has good contact and force points with the gear structure of the servo.

2. Whether the tip of the fixing screw on the gear structure of the servo is stuck with the plastic shell of the servo. If so, please loosen the fixing screw a little to prevent its tip from contacting the servo.

Step 5 Install the LCD display and the DHT11 Sensor

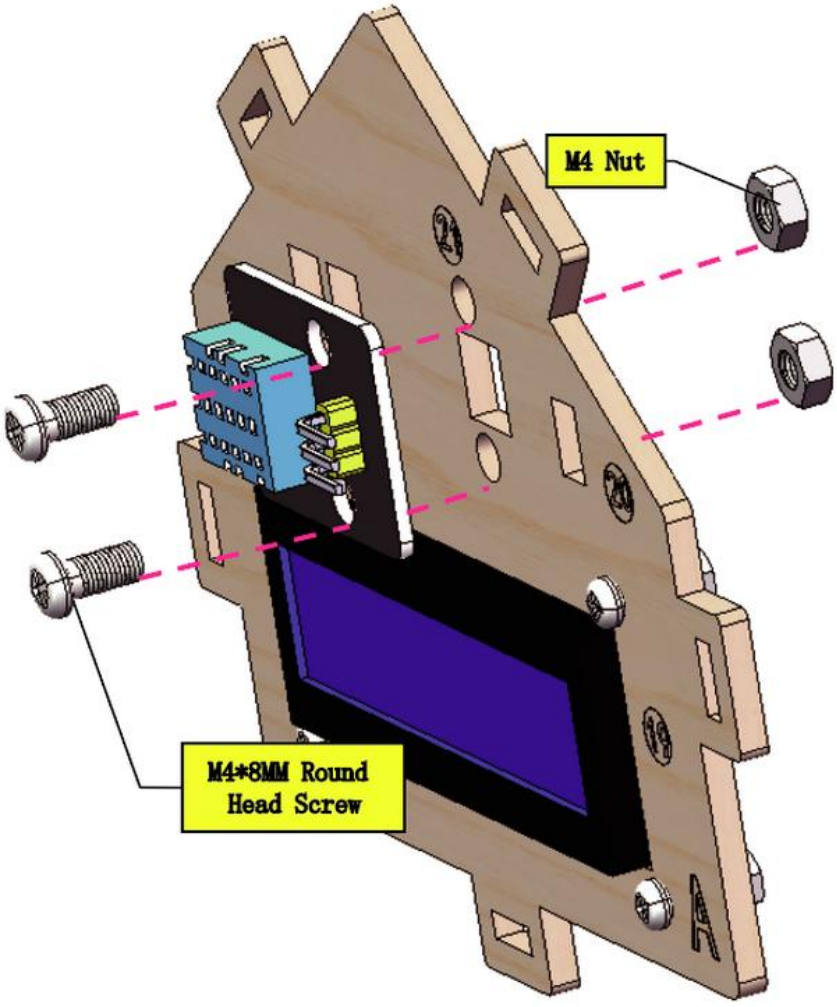
5.1 Required components



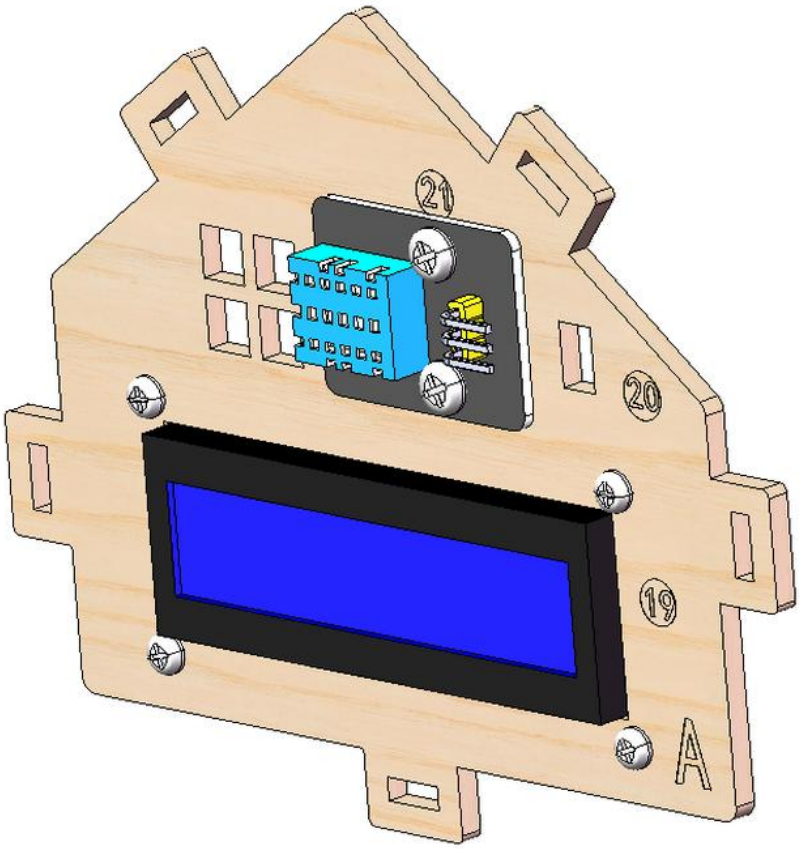


5.3

5.4



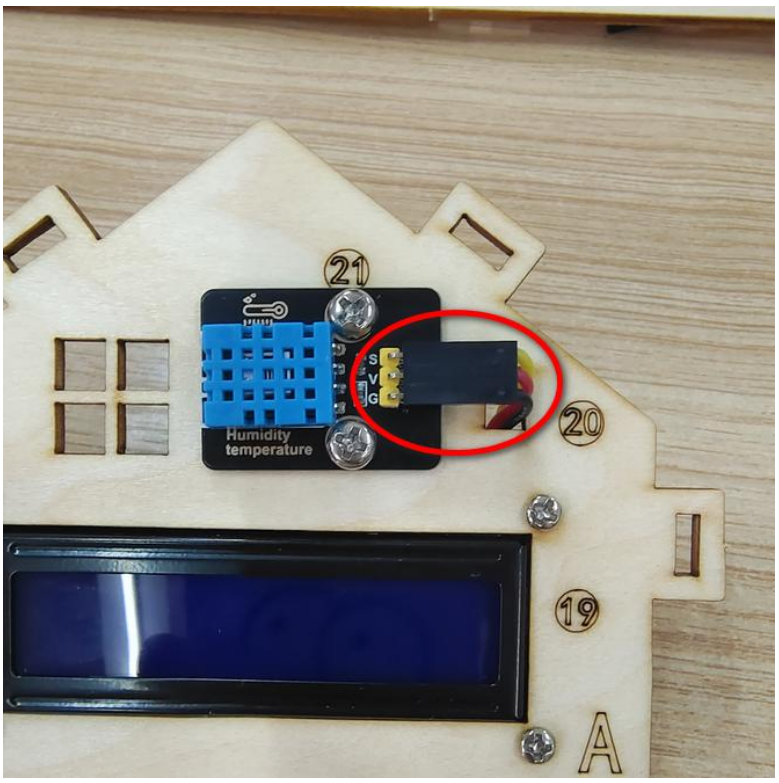
5.5



5.6 **Connect modules via Dupont wires.**

Module	Wire
Temperature and Humidity Sensor	3PIN **20cm**

Connect yellow wire to S, red wire to V, black wire to G.

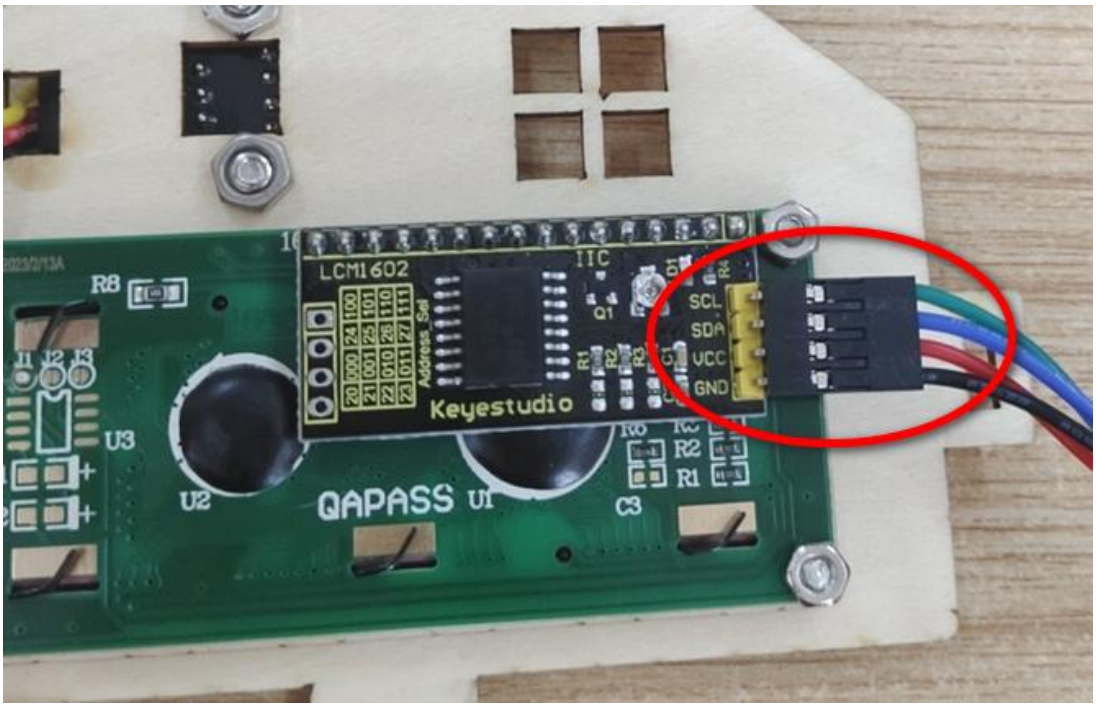


5.7

Module Wire

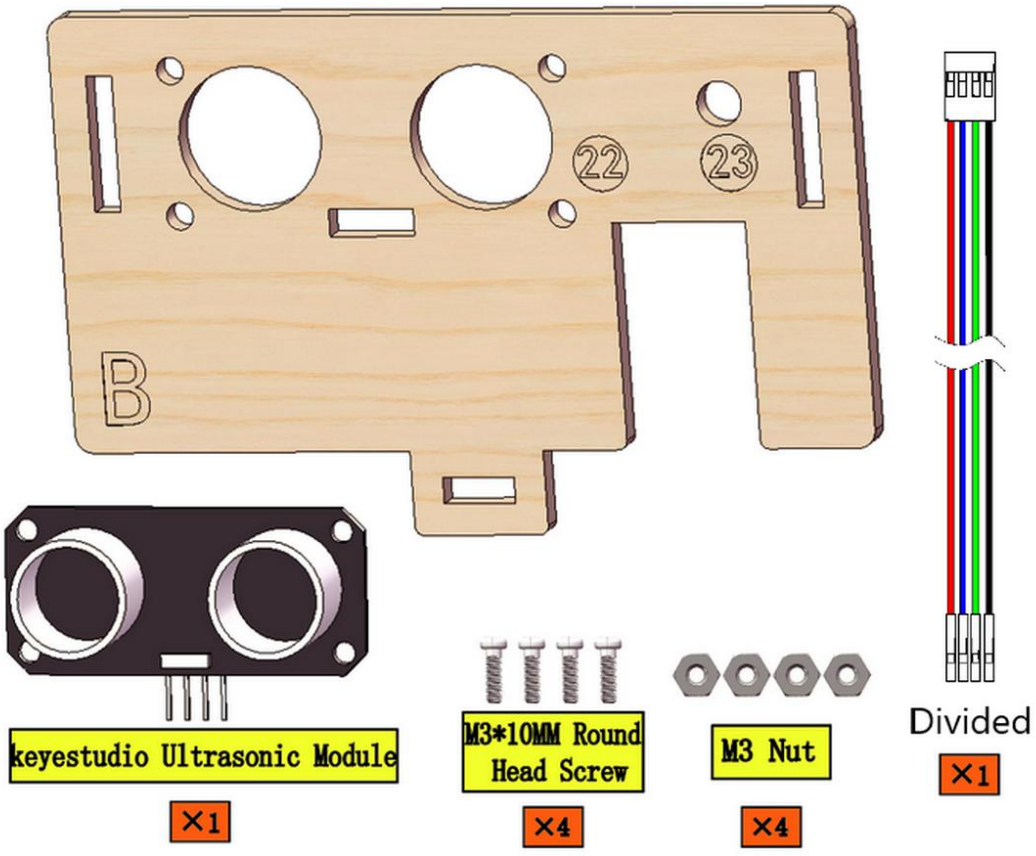
LCD 1602 4PIN **(Black-Red-Blue-Green)**

Connect green to SCL, blue to SDA, red to VCC, black to GND.

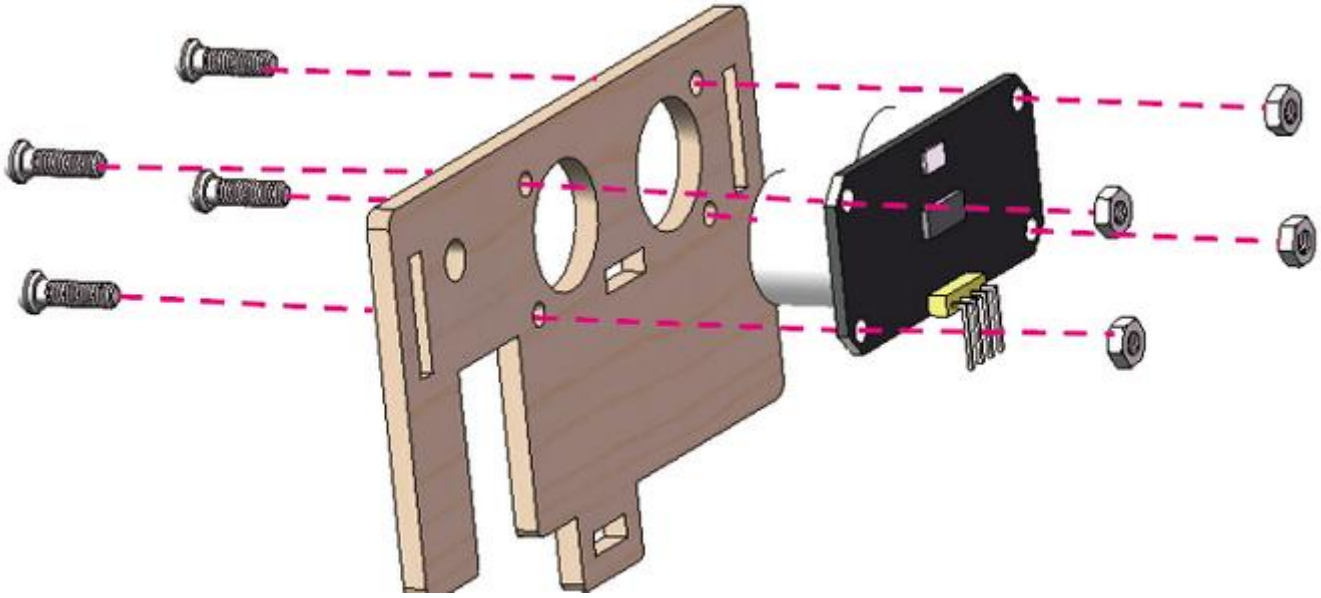


Step 6 Install the Ultrasonic Module

6.1 Required components

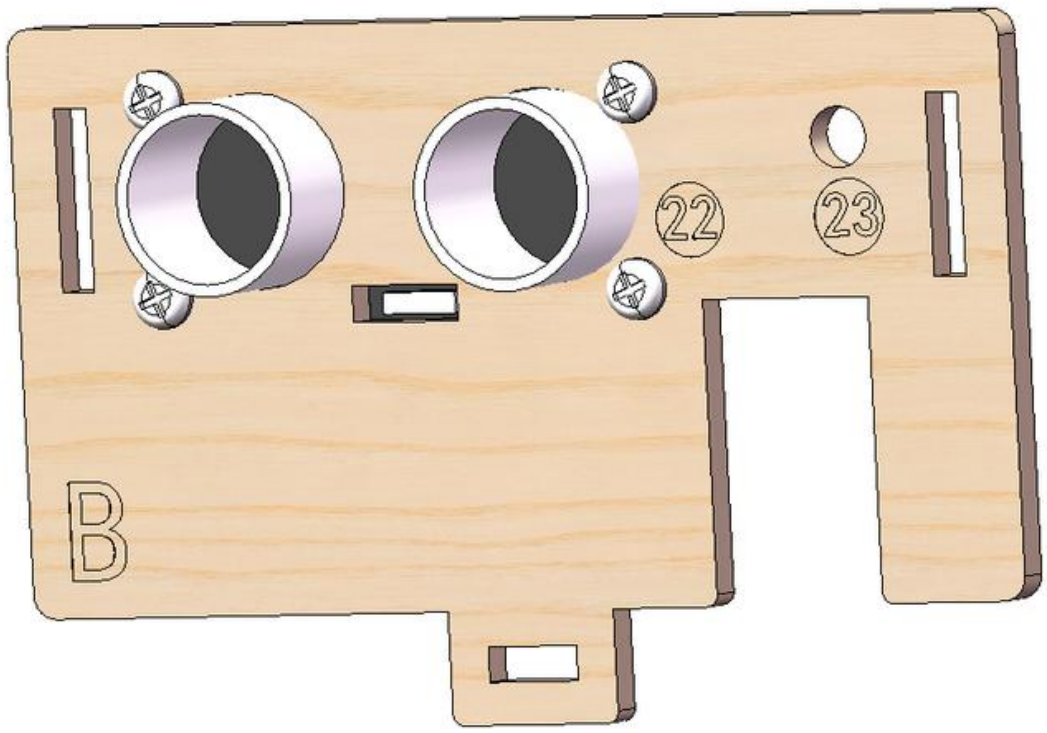


6. 2



---

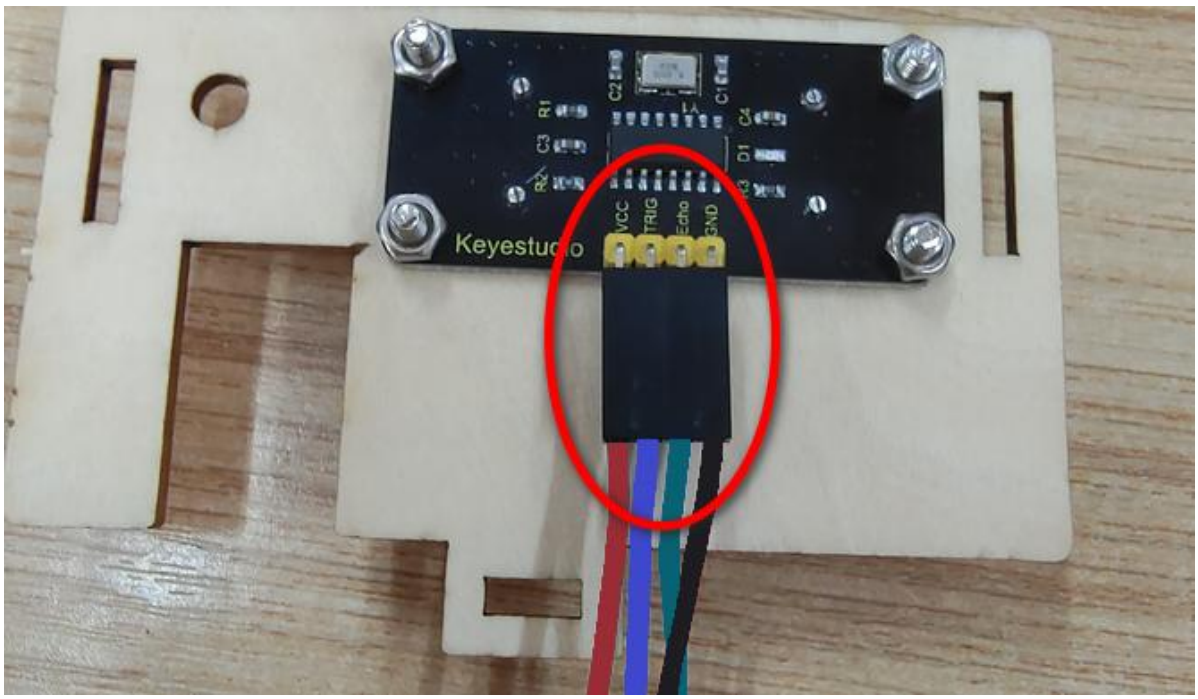
6. 3



6.4 Wiring

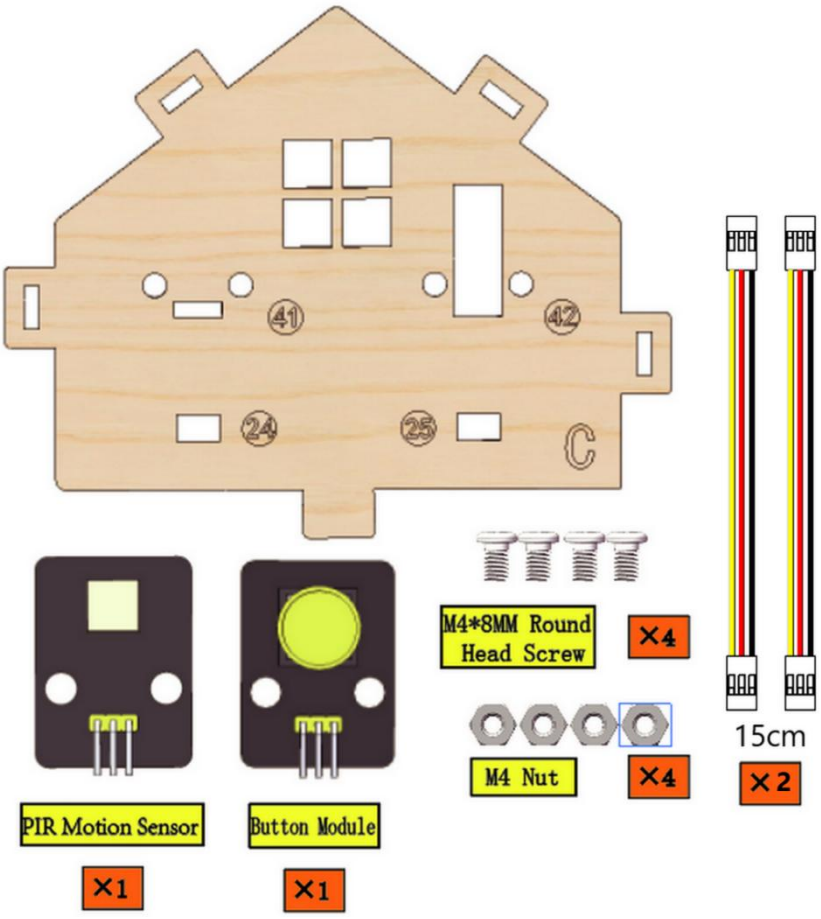
Module	Wire
Ultrasonic module	4PIN **(Black-Green-Blue-Red)**

Pay attention to the color of the Dupont wire: For the ultrasonic module, connect blue to TRIG,green to ECHO, red to VCC, black to GND.

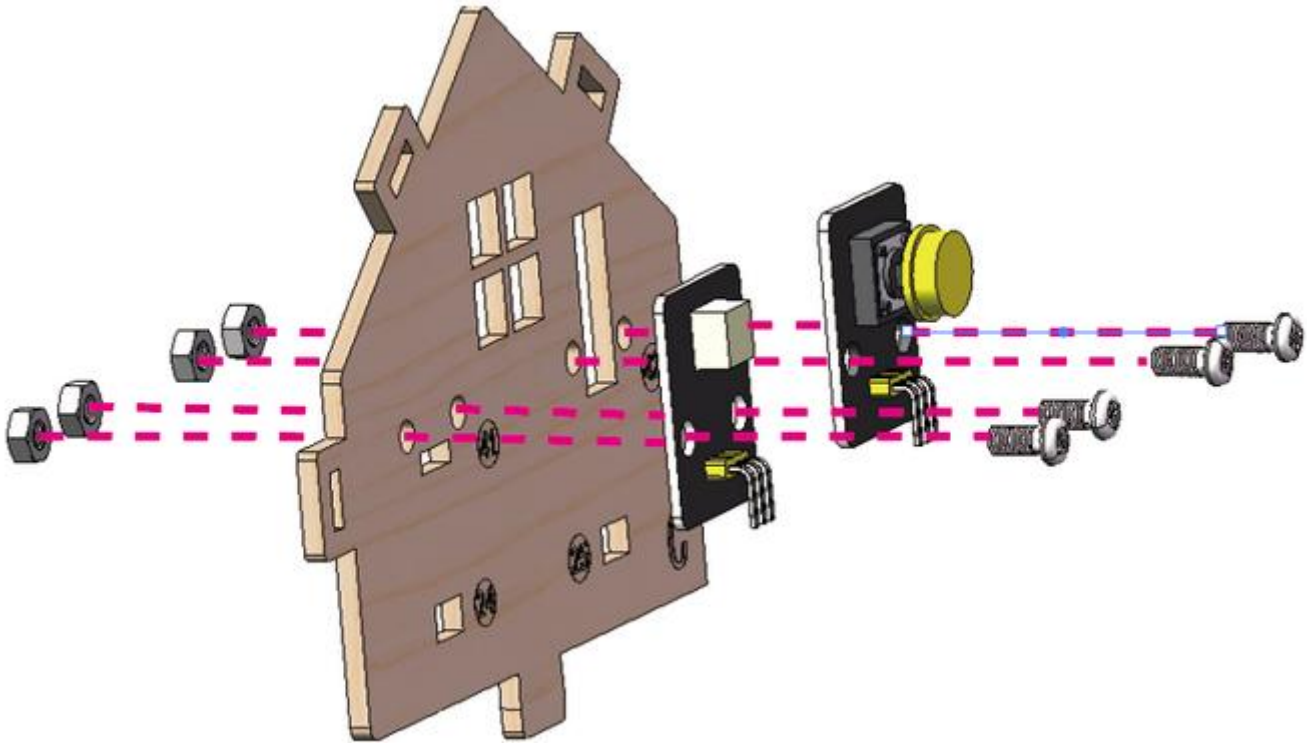


Step 7 Install the PIR Motion Sensor and Button Module

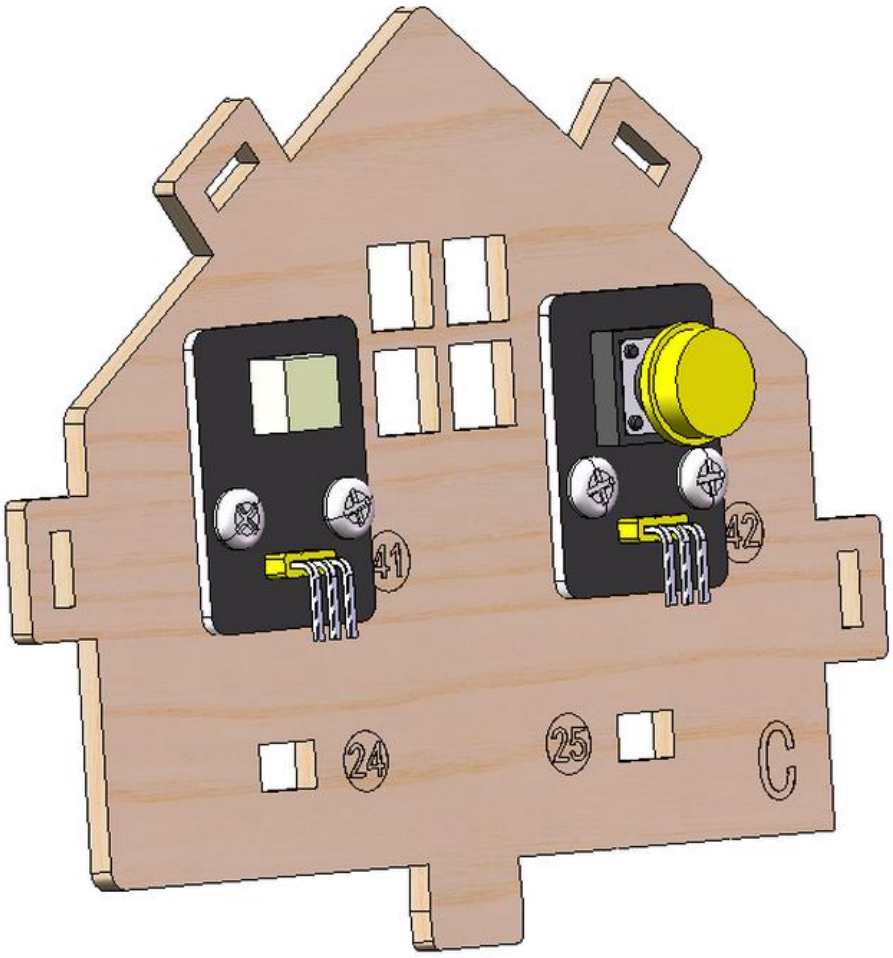
7.1 Required components



7.2



7.3

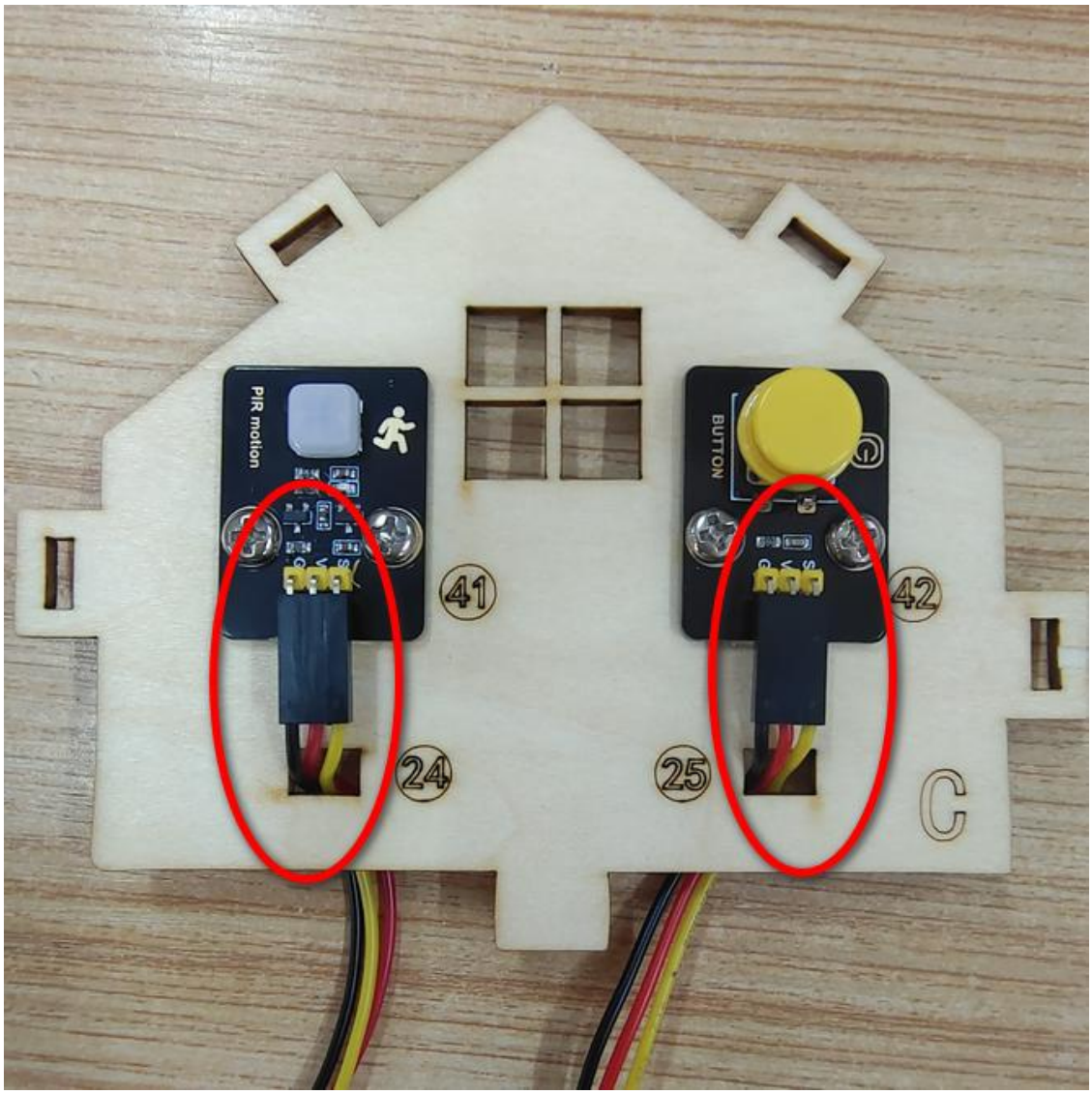


7.4 Wiring

Connect modules via Dupont wires.

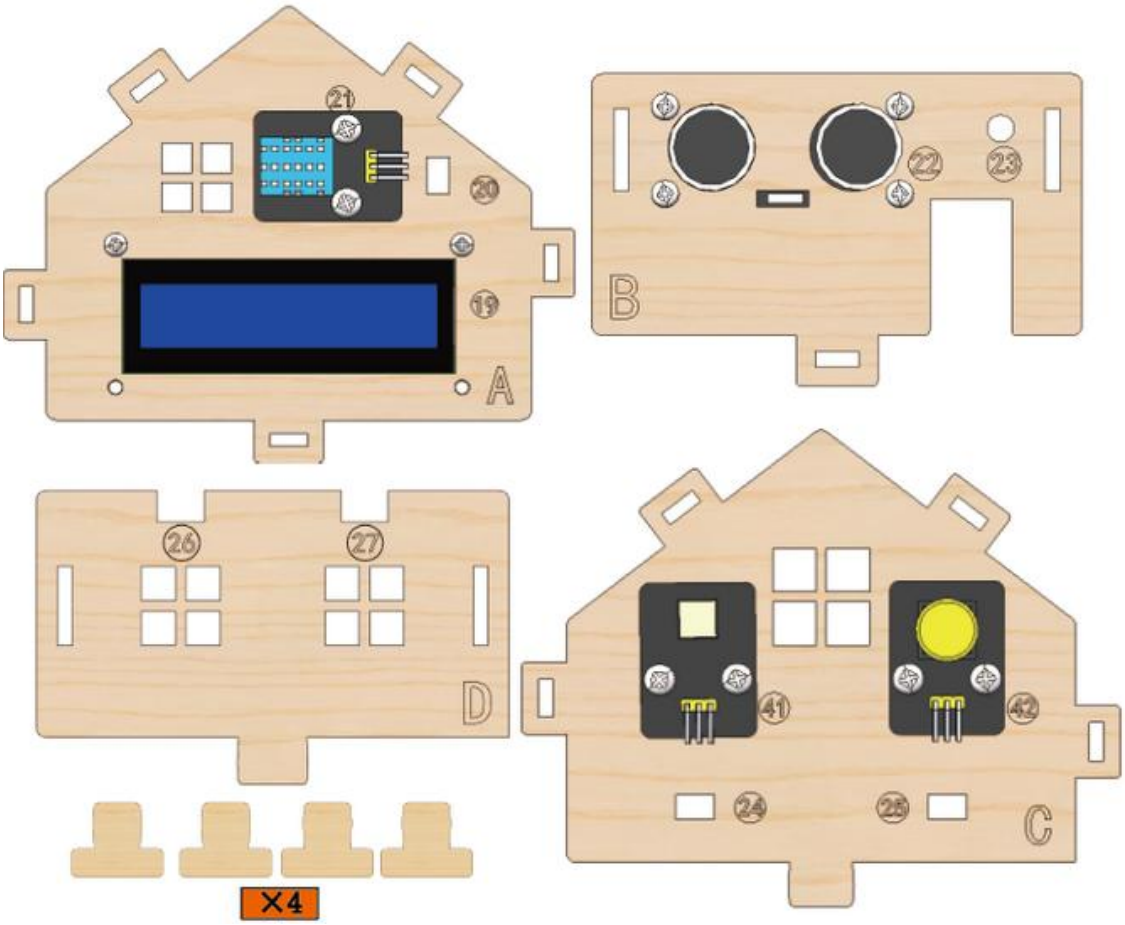
Module	Wire
PIR Motion Sensor	3PIN **15cm**
Button Module	3PIN **15cm**

Pay attention to the color of the Dupont wire: Connect yellow to S, red to V, black to G.

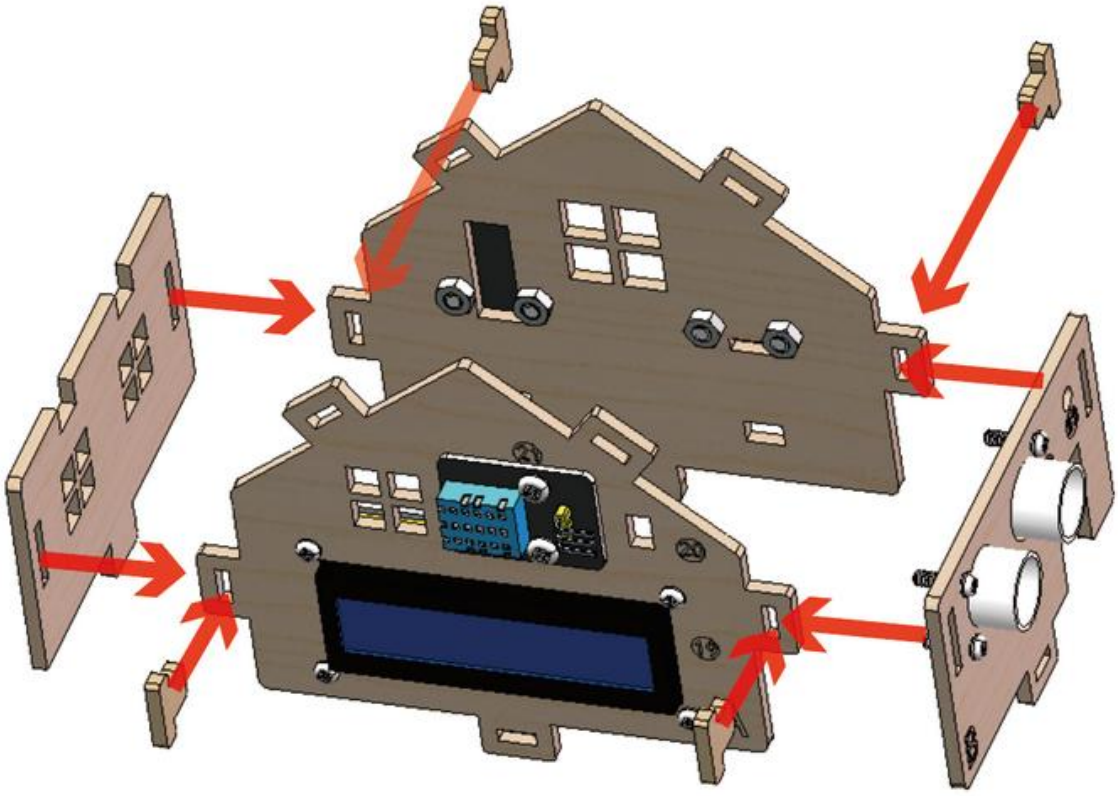


Step 8 Install the Walls of the House

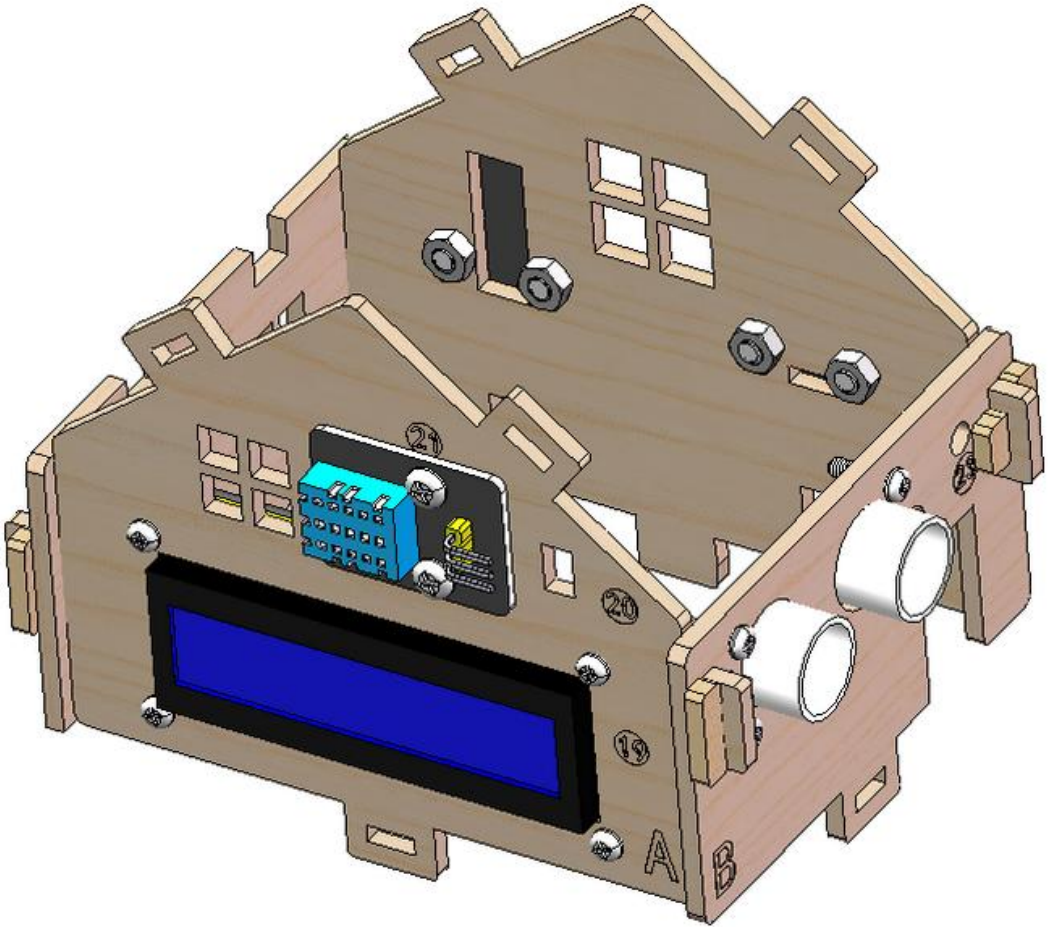
8.1 Required components



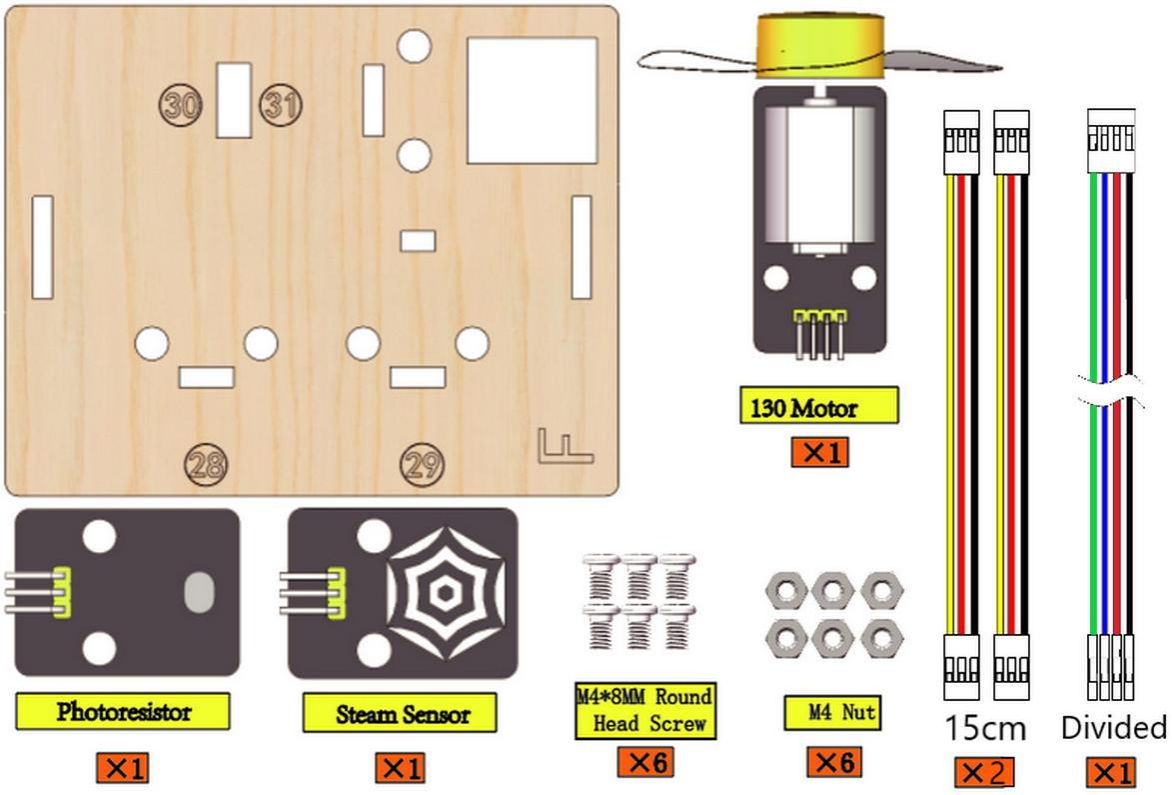
8.2



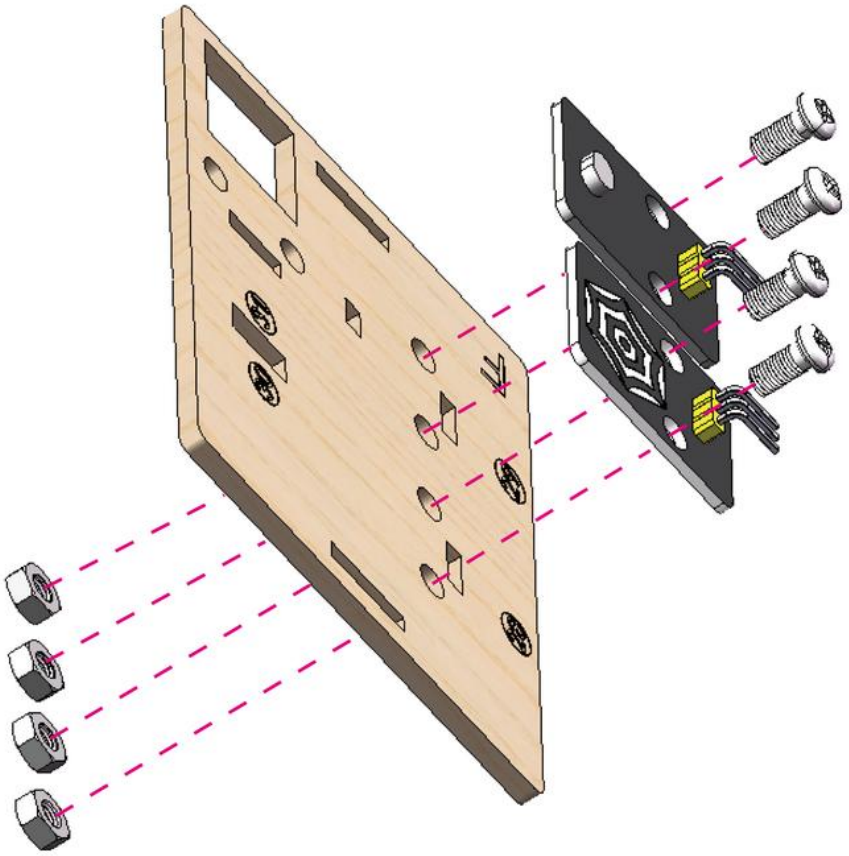
8.3


21
20
19
A
B

8.4

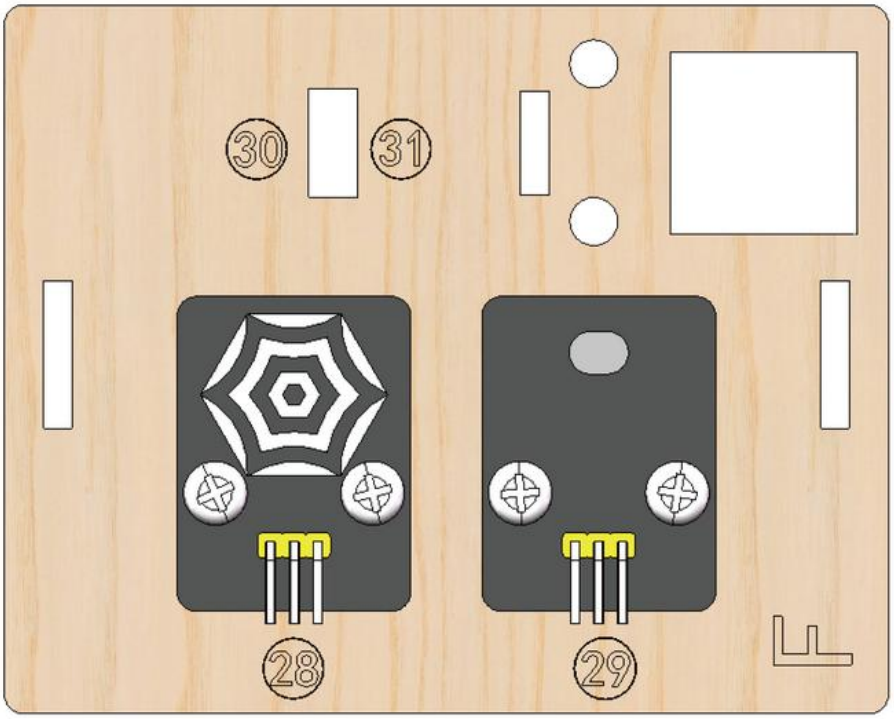

30
31
28
29
130 Motor
×1
Photoresistor
×1
Steam Sensor
×1
M4*8MM Round Head Screw
×6
M4 Nut
×6
15cm
×2
Divided
×1

8. 5

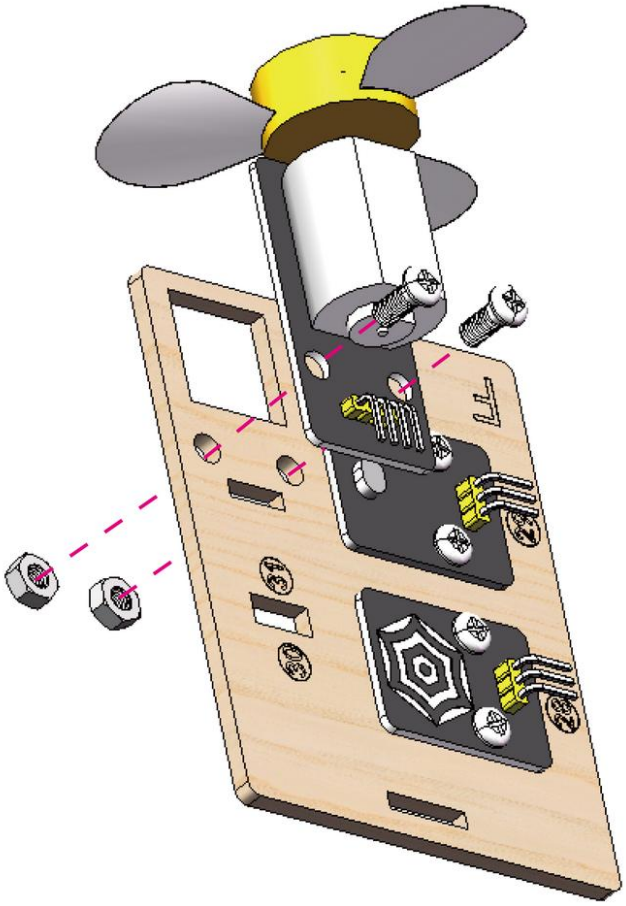


---

8. 6

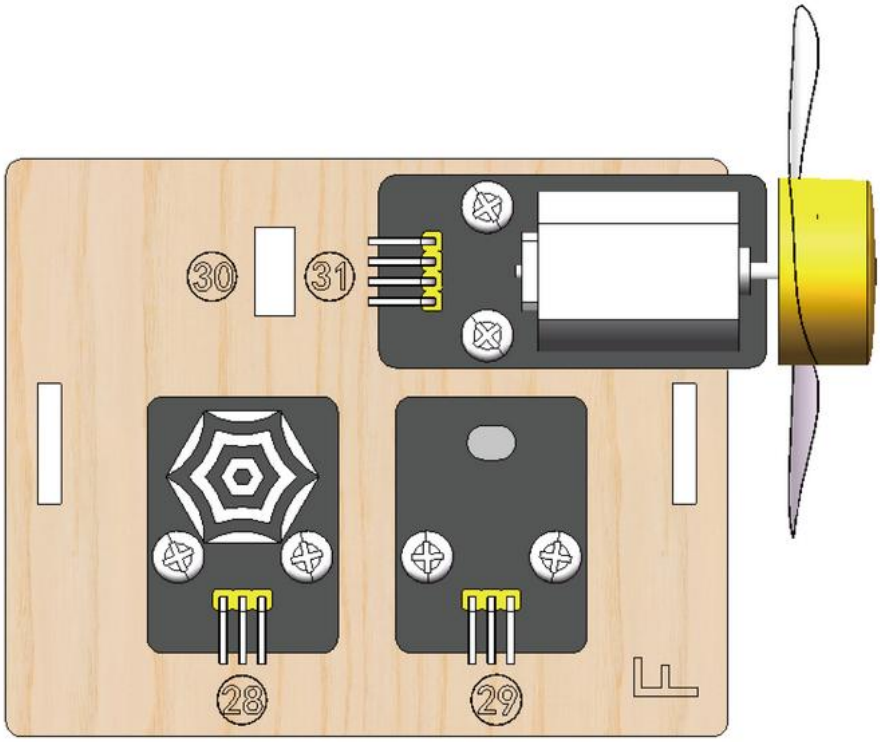


8. 7



---

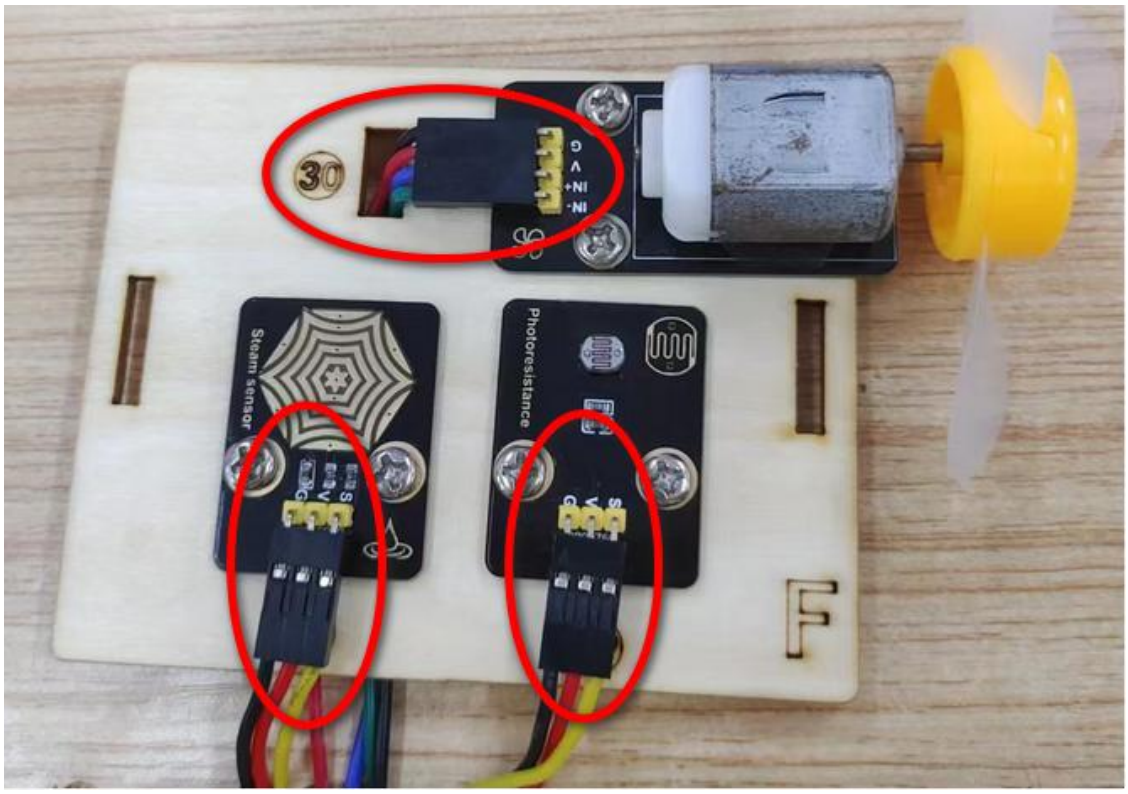
8. 8



8.9 Wiring

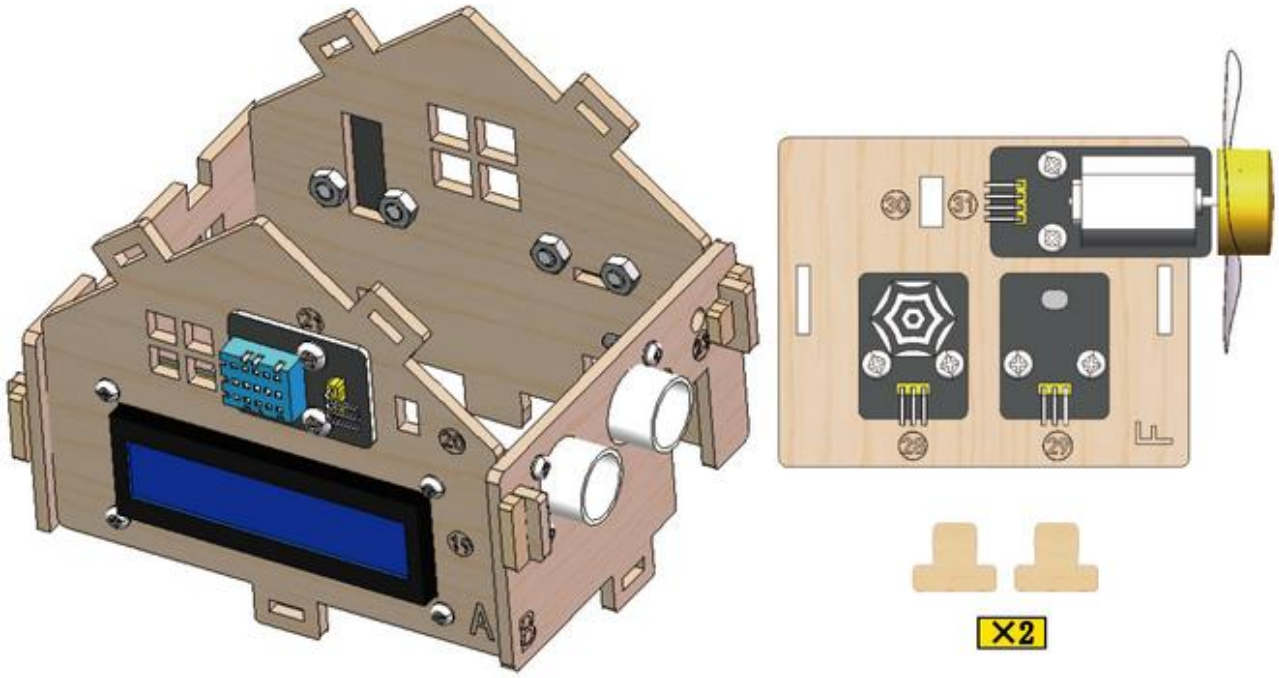
Module	Wire
Fan	4PIN **(Black-Red-Blue-Green)**
Steam Sensor	3PIN **15cm**
Photoresistor	3PIN **15cm**

Pay attention to the color of the Dupont wire: Connect yellow to S, red to V, black to G, blue to IN+, green to IN-.

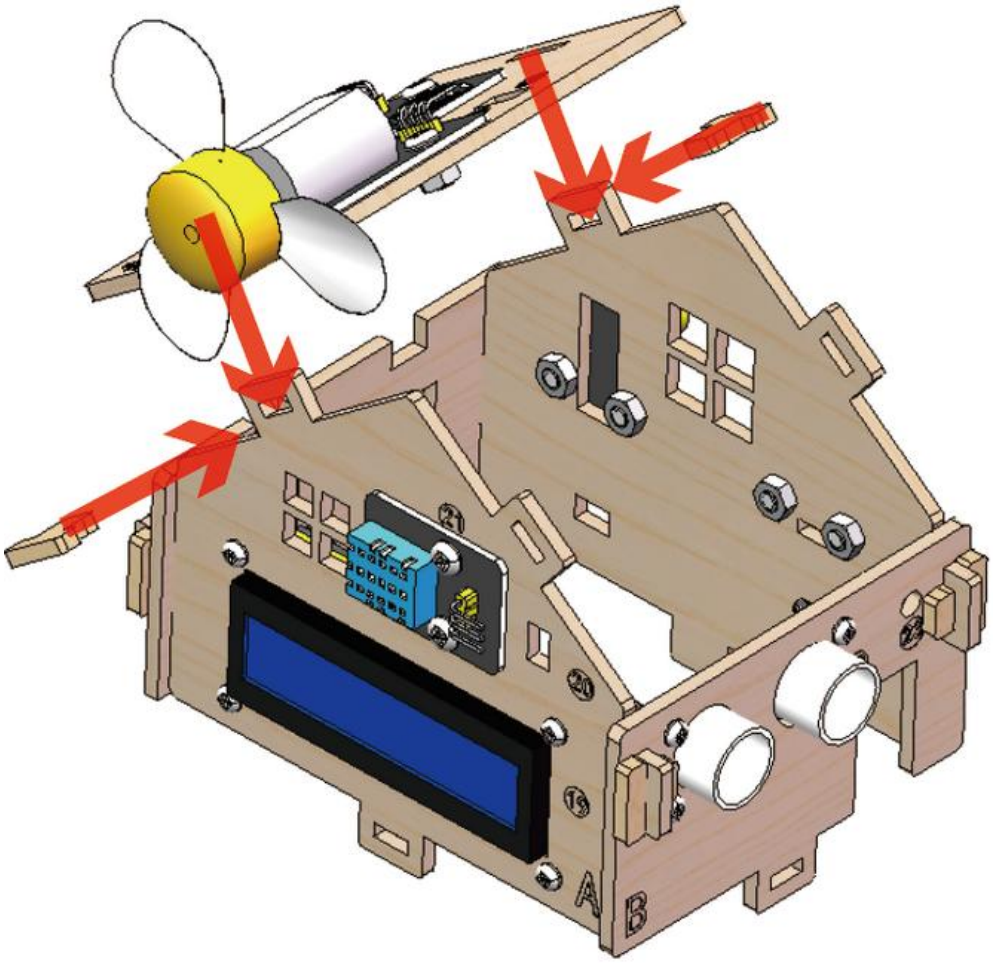


Step 9 Install the Roof of the house

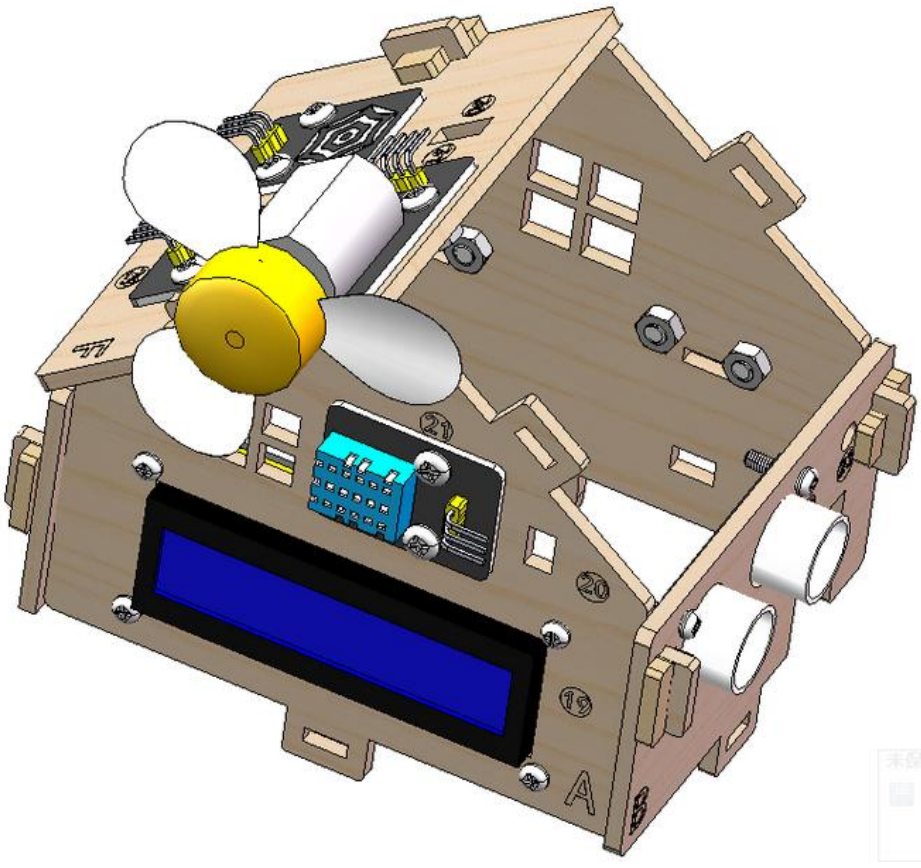
9.1 Required components



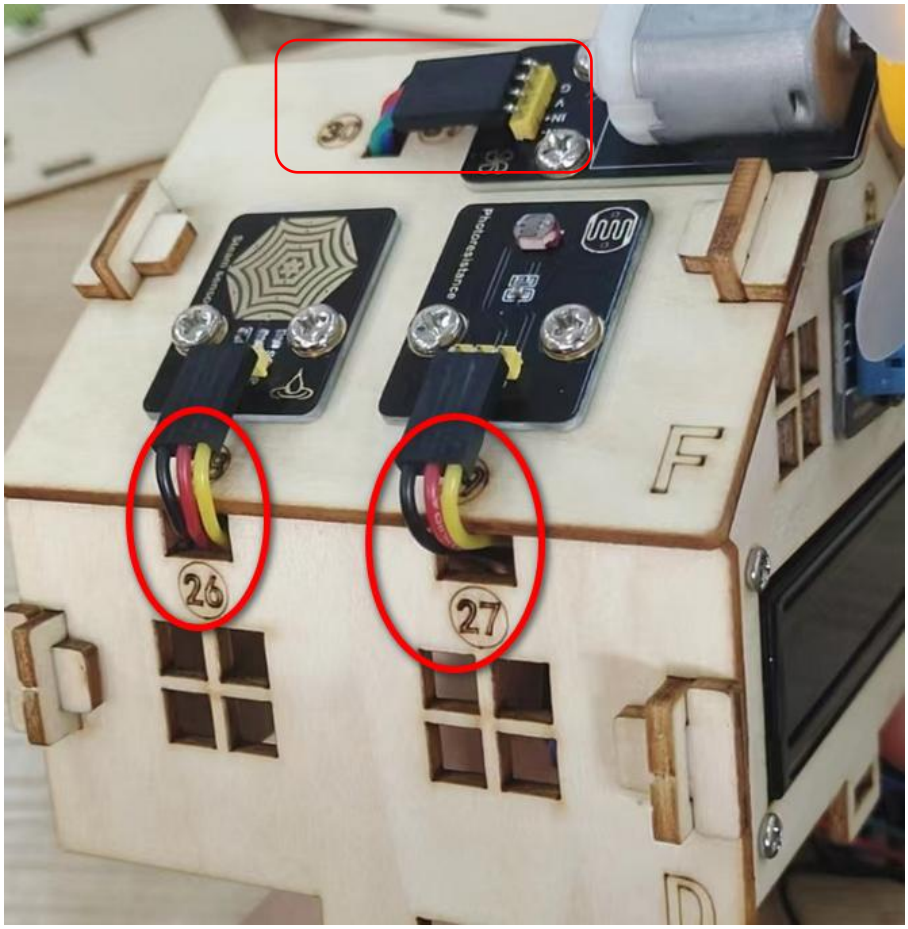
9.2



9.3

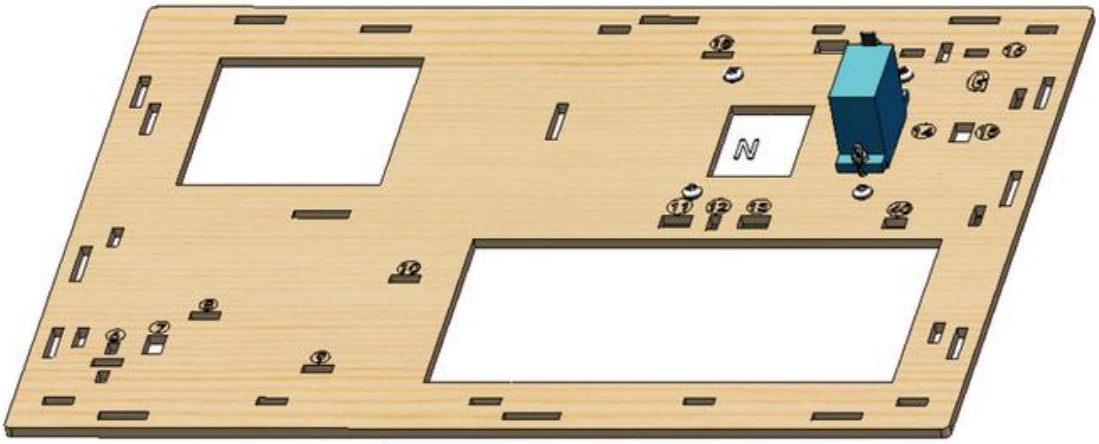


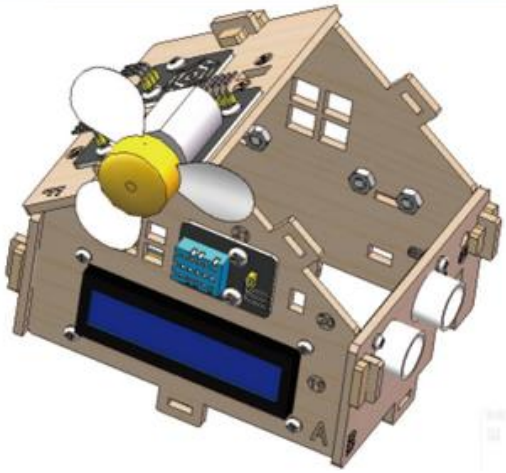
9.4 Keep the wires organized

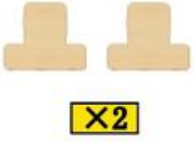


Step 10 Install the House and Ground

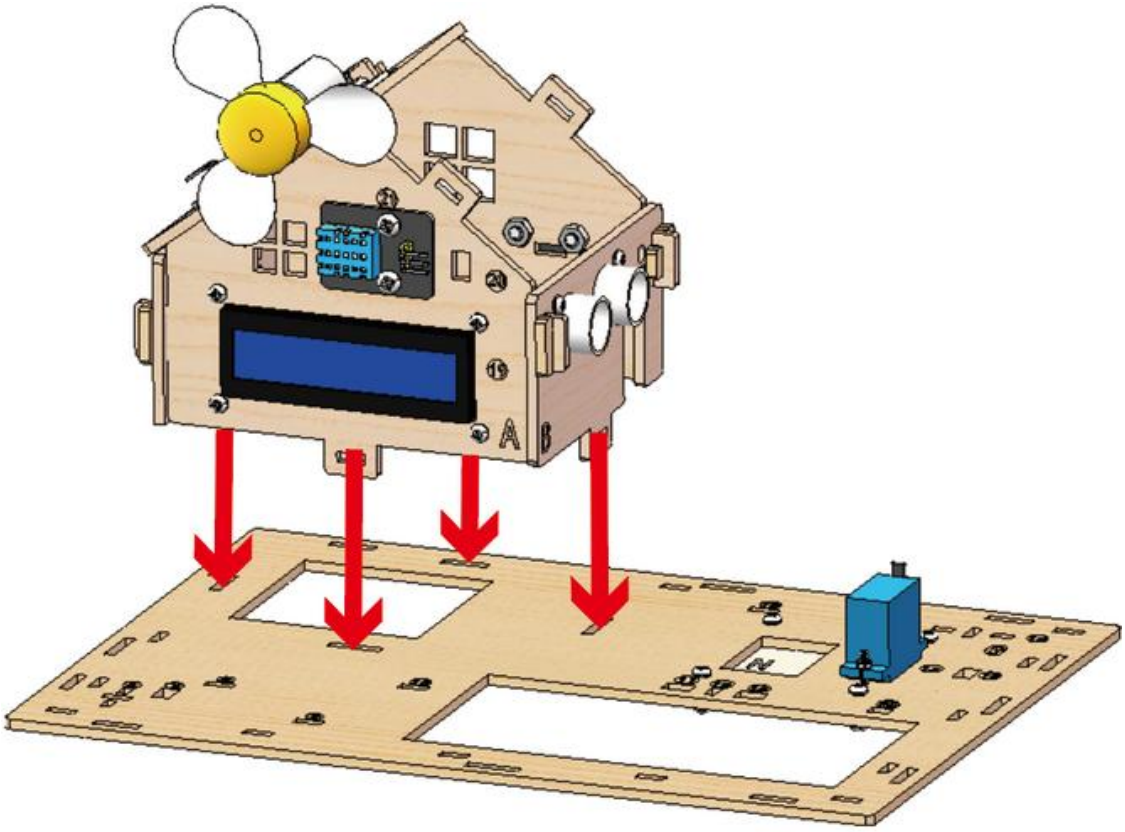
9.1 Required components



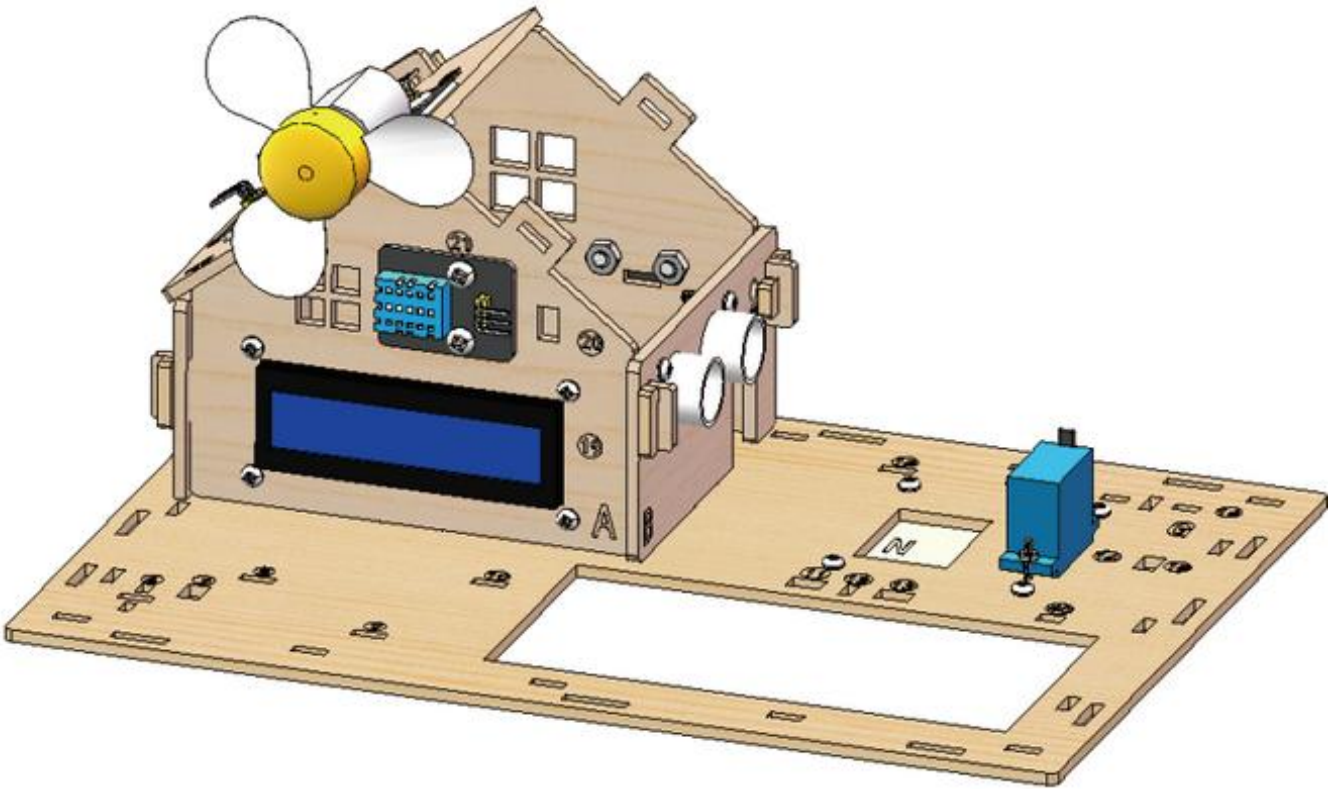




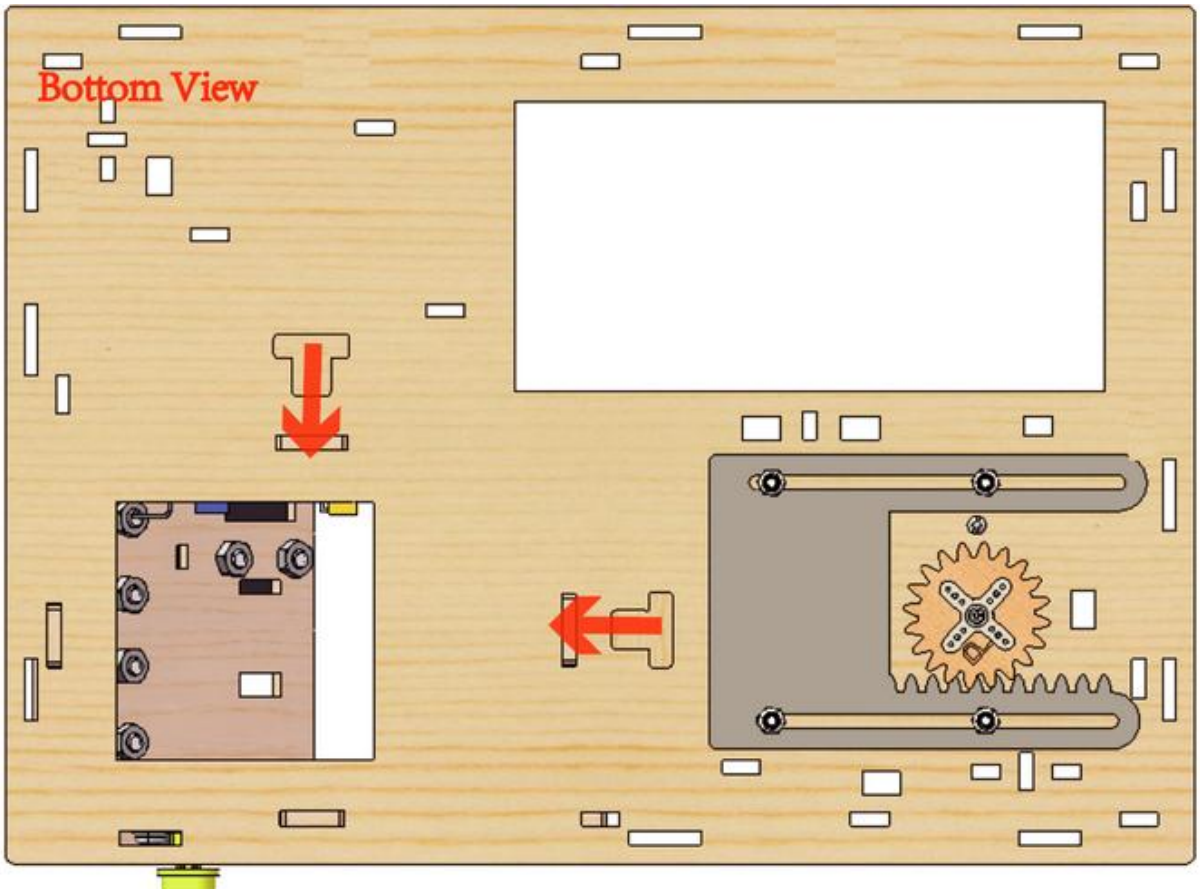
10. 2



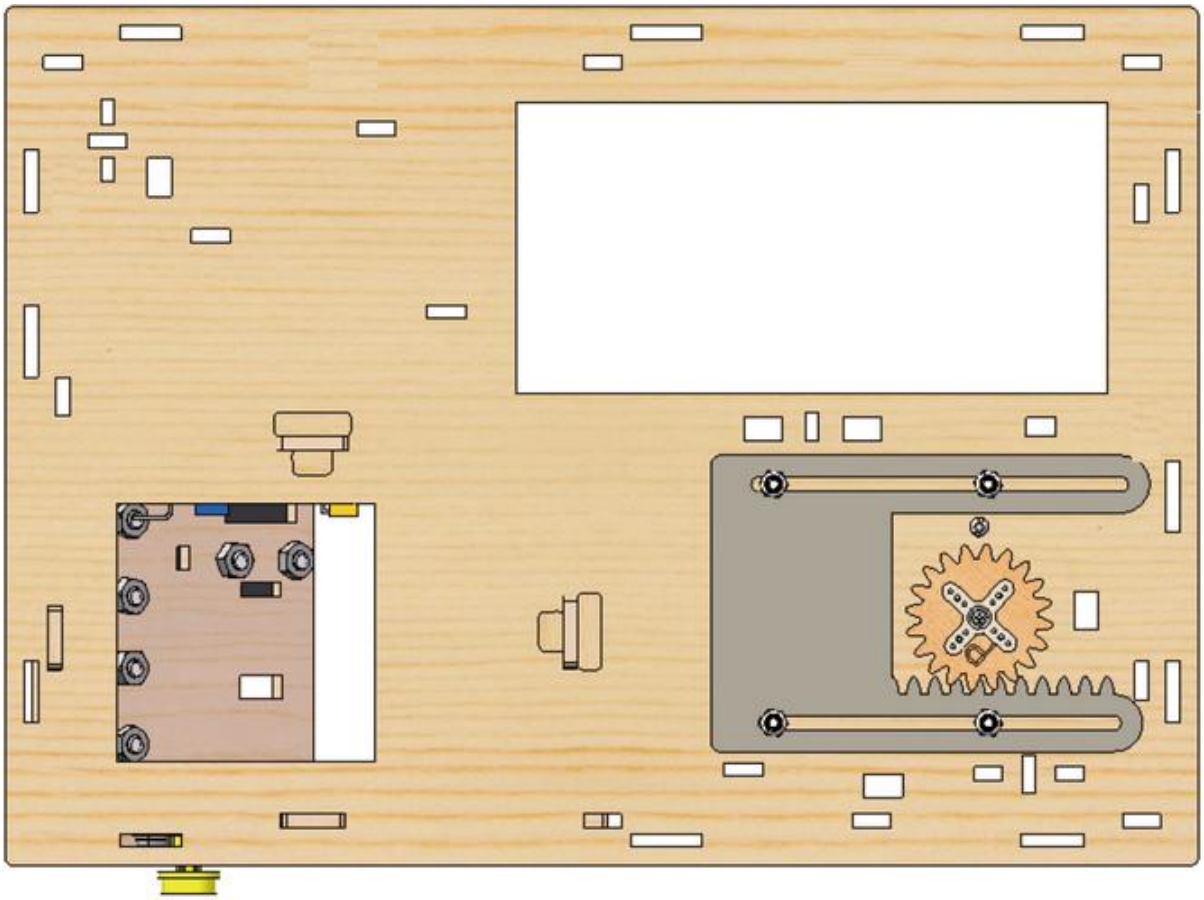
10. 3



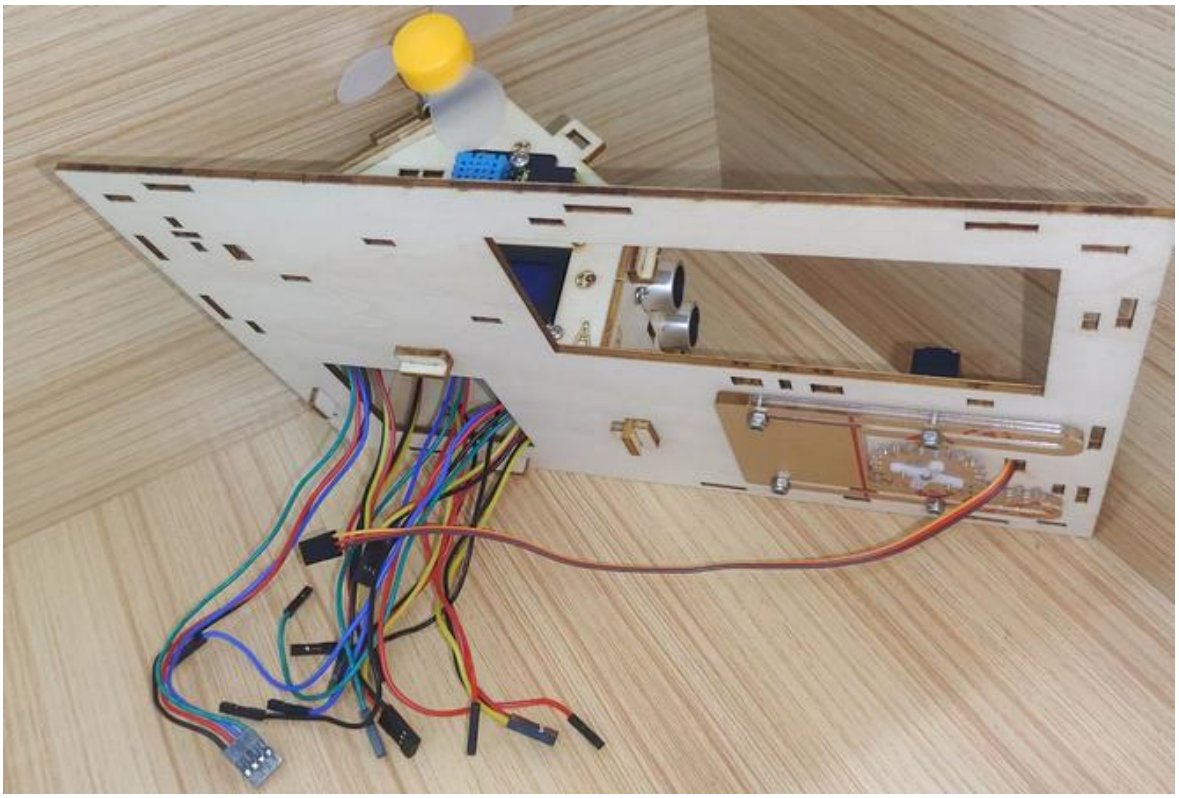
10. 4



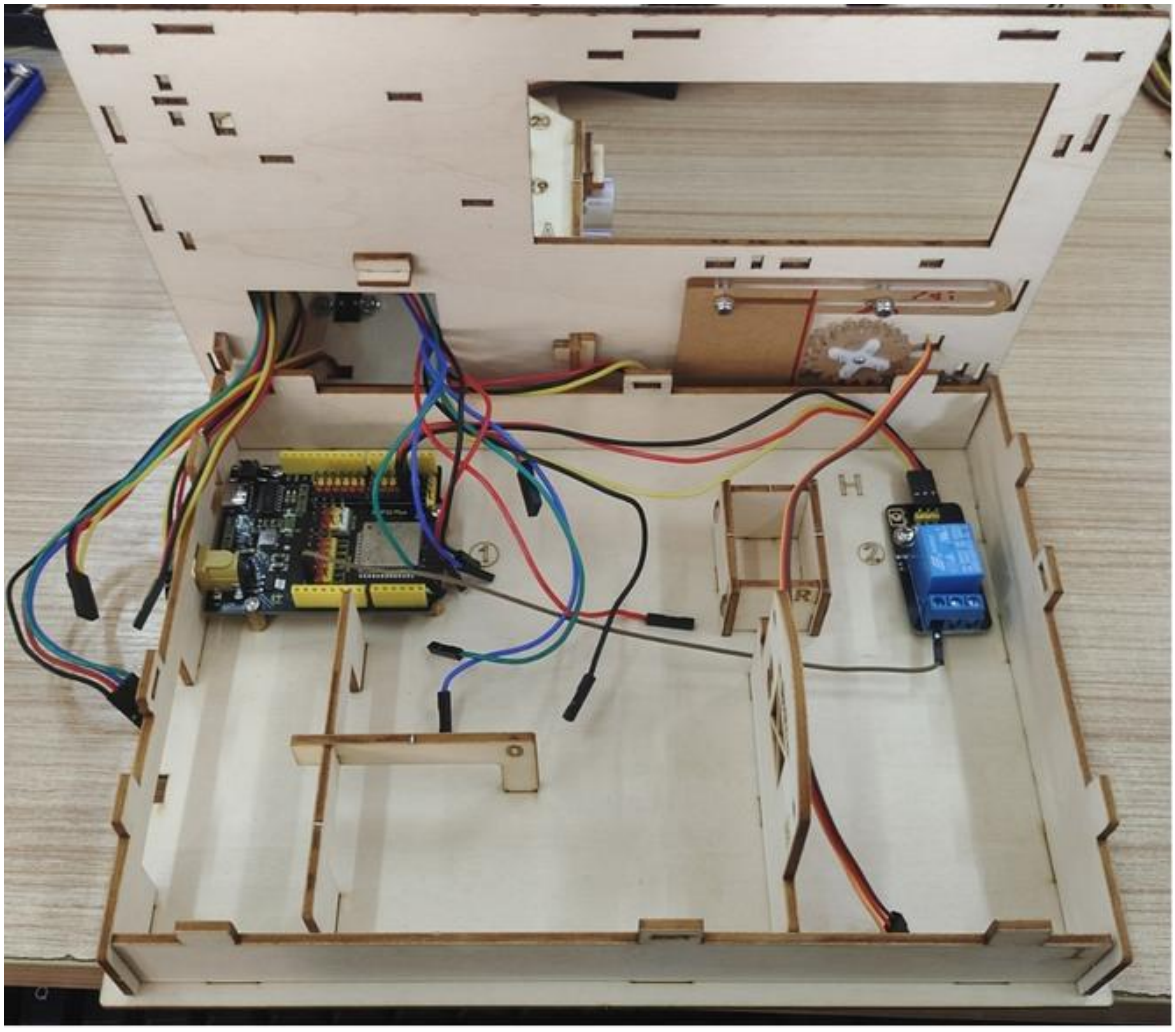
10. 5



10.6 Arrange the wires



10.7



Step 11 Wiring the House

11.1 Wiring List

Pay attention to the color of the Dupont wire: Connect yellow to S, red to V, black to G.

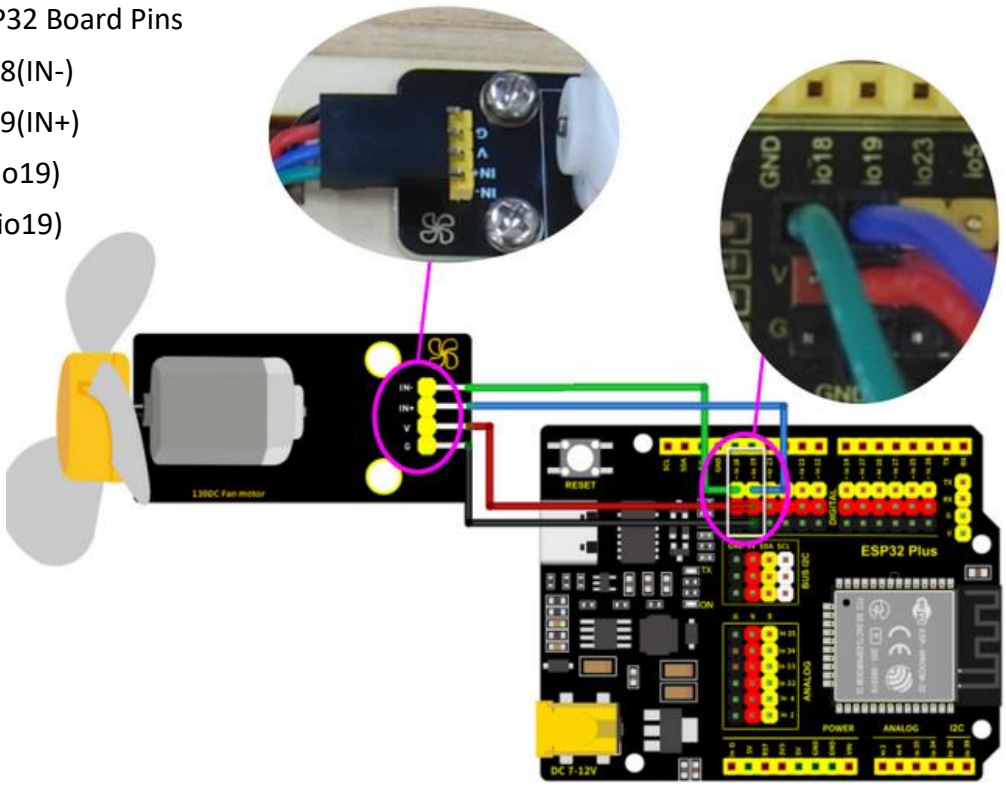
NO.	Components	Wires	ESP32 Board Pins
1	Fan	4pin, **Divided** Black-Red-Blue-Green	io18(IN-) | io19(IN+)
2	PIR Motion Sensor	3pin 15cm	io23
3	Button	3pin 15cm	io5
4	Ultrasonic Module	4pin,**Divided** Black-Green-Blue-Red	D12(TRIG) D13(ECHO)
5	LCD 1602	4pin, **Connected**	I2C
6	Temperature and Humidity Sensor	3pin 20cm	io17
7	Steam Sensor	3pin 15cm	io35
8	Photoresistor	3pin 15cm	io34
9	Servo	--	io26
10	Buzzer	3pin 20cm	io16
11	LED	3pin 20cm	io27
12	Water Level Sensor	3pin 25cm	io33
13	Soil Humidity Sensor	3pin 20cm	io32
14	Water Pump	3pin 20cm	io25

11.2 Fan

Pass the Dupont wire connected to the fan through the hole **marked 30** on the wooden board.

Components	Wire	ESP32 Board Pins
Fan	4PIN **Divided (Black-Red-Blue-Green)**	io18(IN-), io19(IN+)

Module Pin	Wire Color	ESP32 Board Pins
IN-	GREEN	io18(IN-)
IN+	BLUE	io19(IN+)
V	RED	V (io19)
G	BLACK	G (io19)



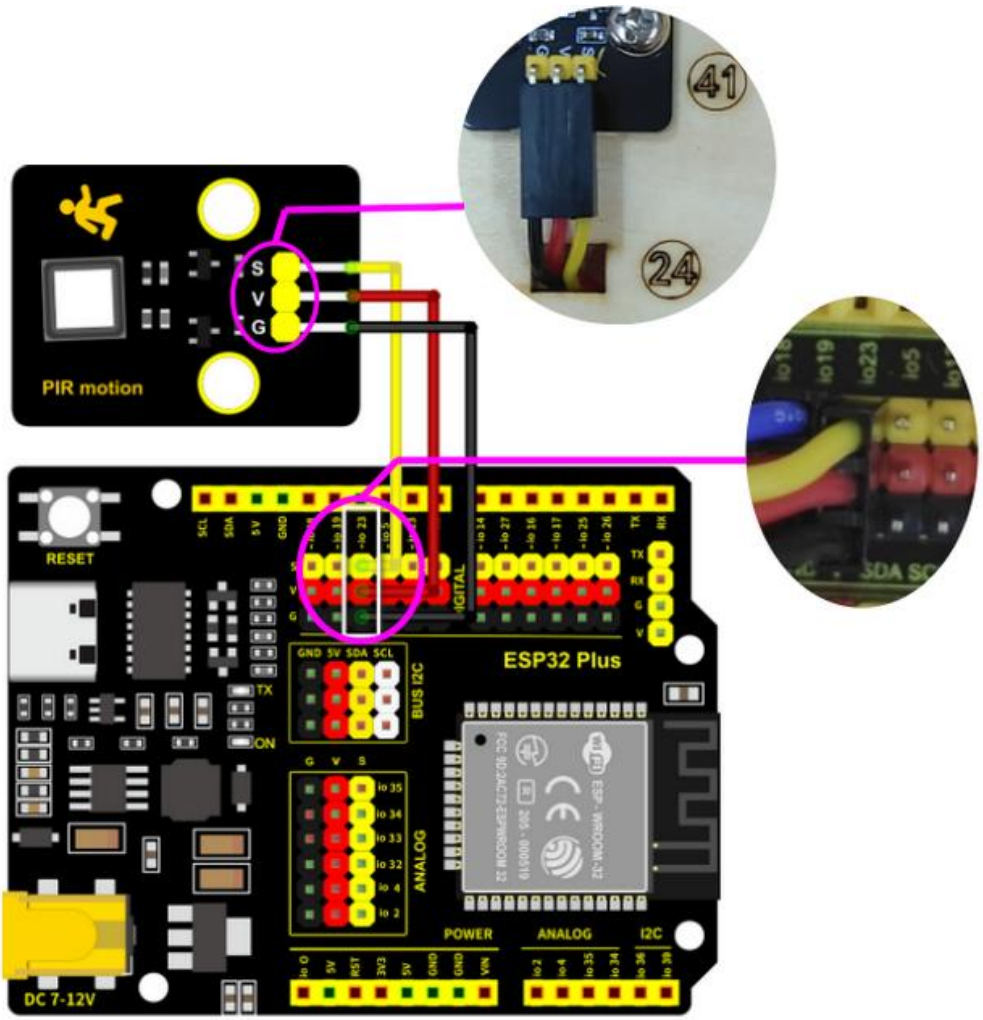
11.3 PIR Motion Sensor

Pass the Dupont wire connected to the PIR motion sensor through the hole **marked 24** on the wooden board.

Component	Wire	ESP32 Board Pin
PIR Motion Sensor	3PIN 15cm	io23

Connect red to V, black to G, yellow to S.

Module Pin	Wire Color	ESP32 Board Pin
V	RED	V
G	BLACK	G
S	YELLOW	io23



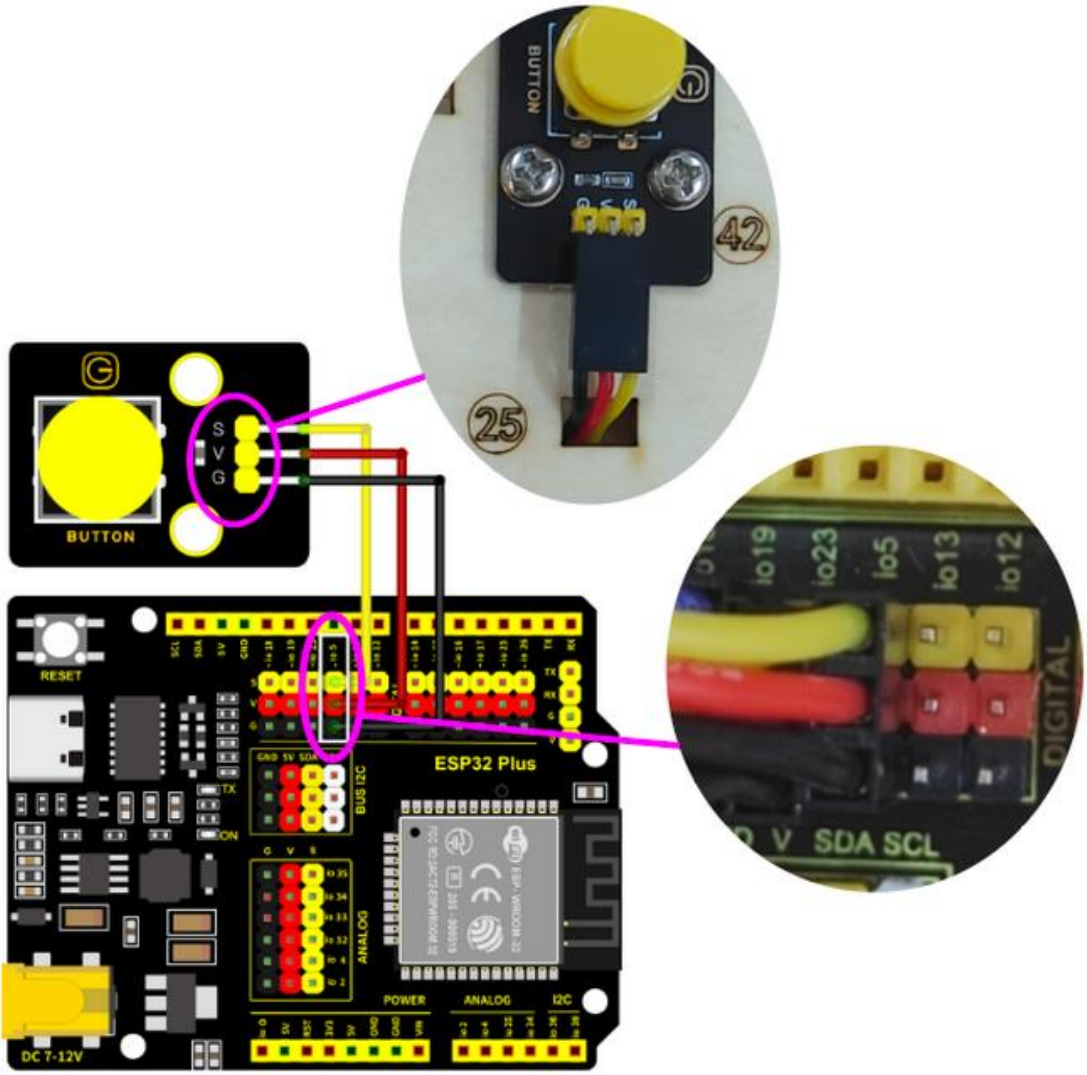
11.4 Button Module

Pass the Dupont wire connected to the button module through the hole **marked 25** on the wooden board.

Component	Wire	ESP32 Board Pin
Button	3PIN 15cm	io5

Connect red to V, black to G, yellow to S.

Module Pin	Wire Color	ESP32 Board Pin
V	RED	V
G	BLACK	G
S	YELLOW	io5



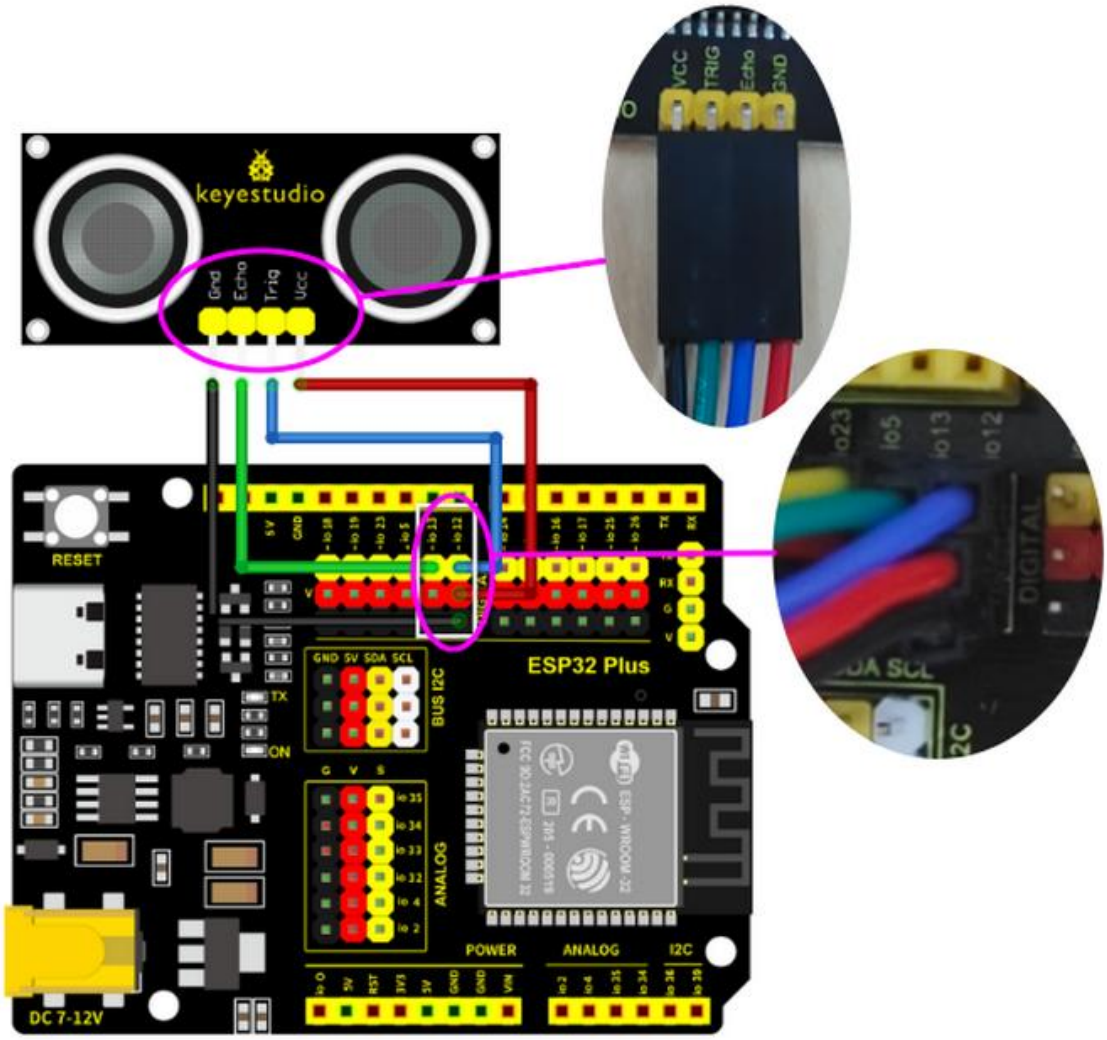
11.5 Ultrasonic Module

Pass the Dupont wire connected to the ultrasonic module through the hole **marked 25** on the wooden board.

Component	Wire	ESP32 Board Pins
Ultrasonic Module	4PIN **Divided (Black-Green-Blue-Red)**	io13(ECHO), io12(TRIG)

Connect red to V, black to G, blue to io12, green to io13.

Module Pin	Wire Color	ESP32 Board Pins
V	RED	V (io12)
G	BLACK	G (io12)
ECHO	GREEN	io13
TRIG	BLUE	io12

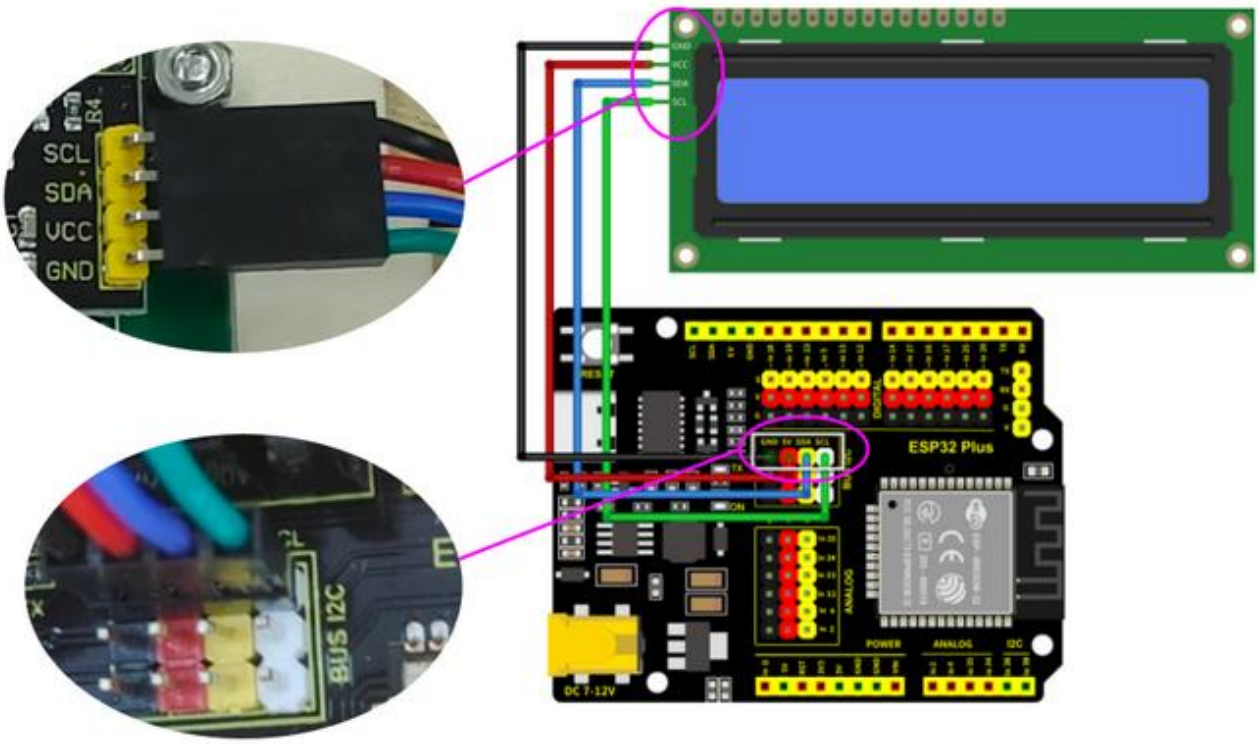


11.6 1602 LCD Display

Component	Wire	ESP32 Board Pins
LCD1602	4PIN **Connected (Black-Red-Blue-Green)**	I2C

Connect red to V, black to G, blue to SDA, green to SCL.

Module Pin	Wire Color	ESP32 Board Pins
V	RED	V (I2C)
G	BLACK	GND (I2C)
SCL	GREEN	SCL (I2C)
SDA	BLUE	SDA (I2C)



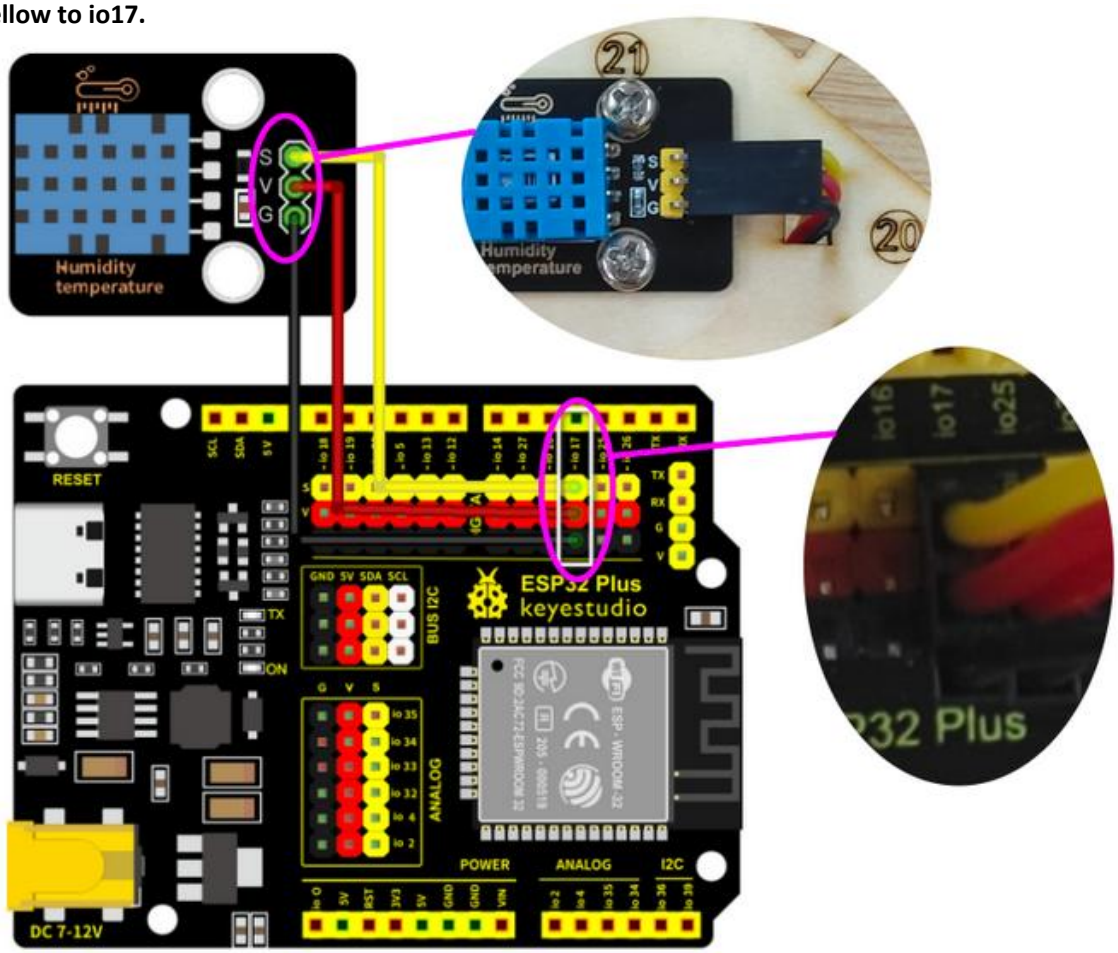
11.7 Temperature and Humidity Sensor

Pass the Dupont wire connected to the button module through the hole **marked 20** on the wooden board.

Component	Wire	ESP32 Board Pins
Temperature and Humidity Sensor	3PIN 20cm	io17

Connect red to V, black to G, yellow to io17.

Module Pin	Wire Color	ESP32 Board Pin
V	RED	V
G	BLACK	G
S	YELLOW	io17

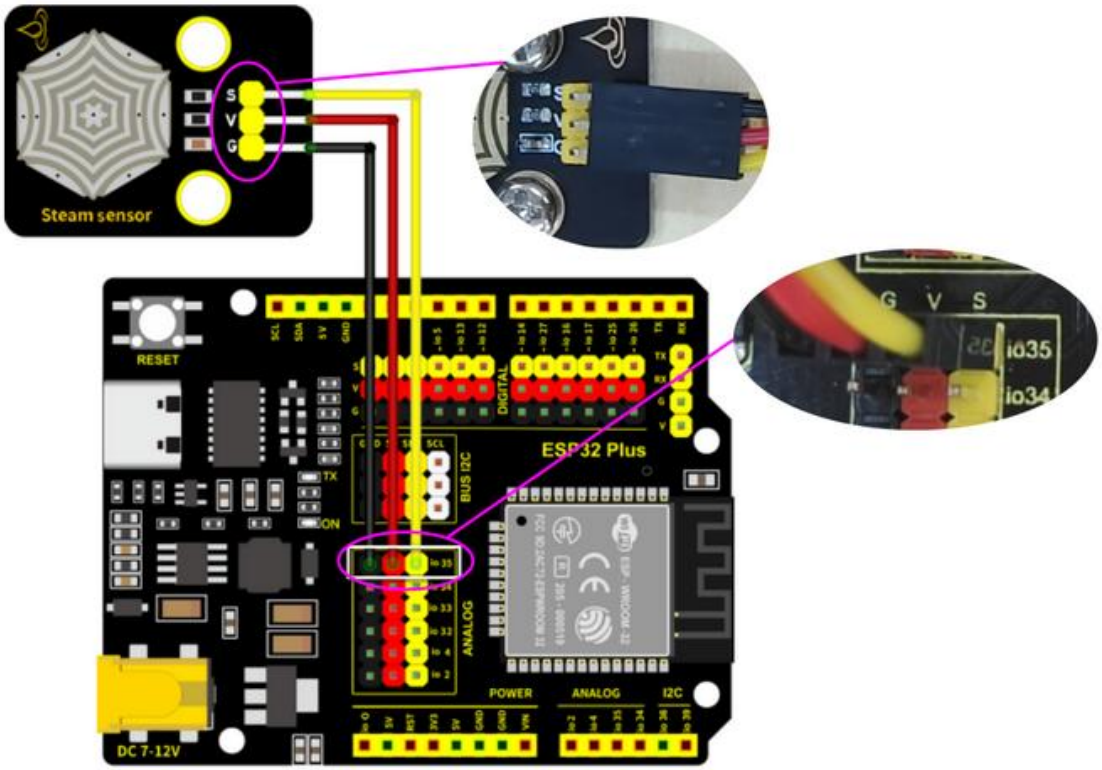


11.8 Steam Sensor

Component	Wire	ESP32 Board Pin
Steam Sensor	3PIN 15cm	io35

Connect red to V, black to G, yellow to io35.

Module Pin	Wire Color	ESP32 Board Pin
V	RED	V
G	BLACK	G
S	YELLOW	io35

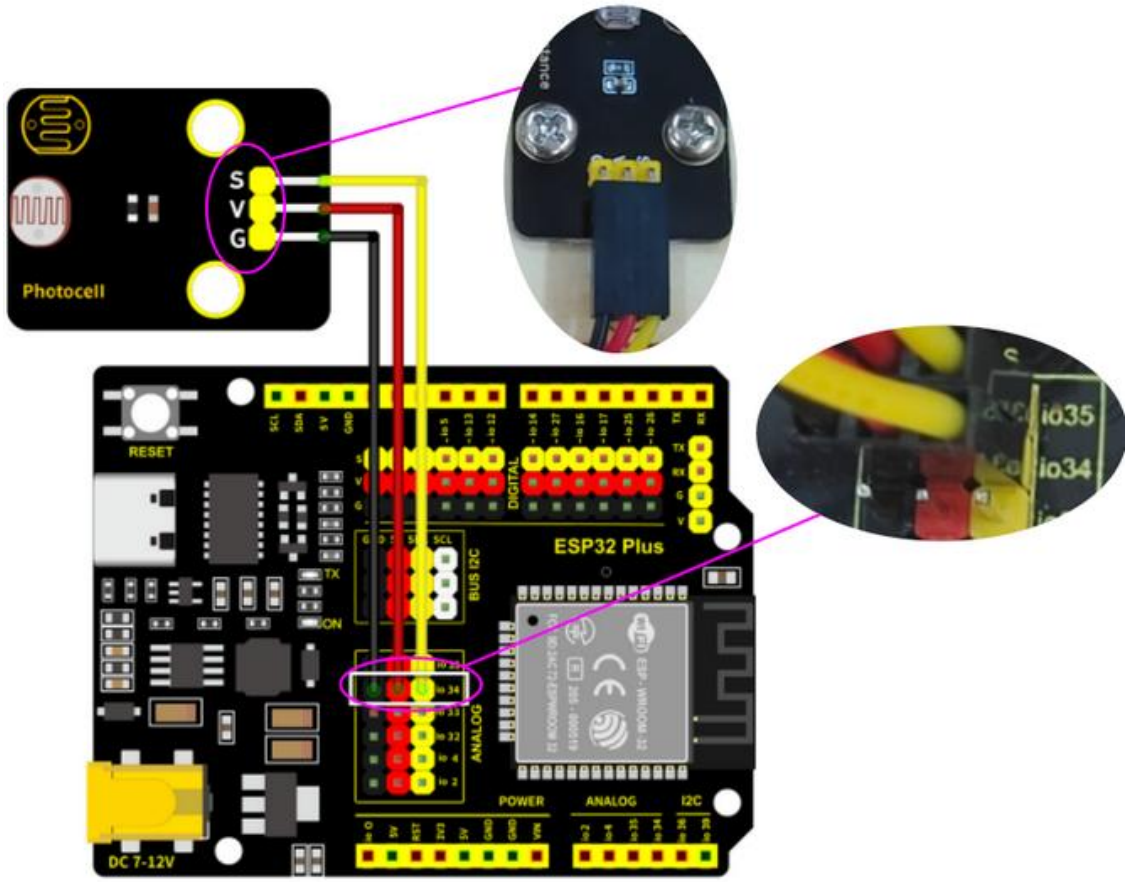


11.9 Photoresistor

Component	Wire	ESP32 Board Pin
Photoresistor	3PIN 15cm	io34

Connect red to V, black to G, yellow to io34.

Module Pin	Wire Color	ESP32 Board Pin
V	RED	V
G	BLACK	G
S	YELLOW	io34



---

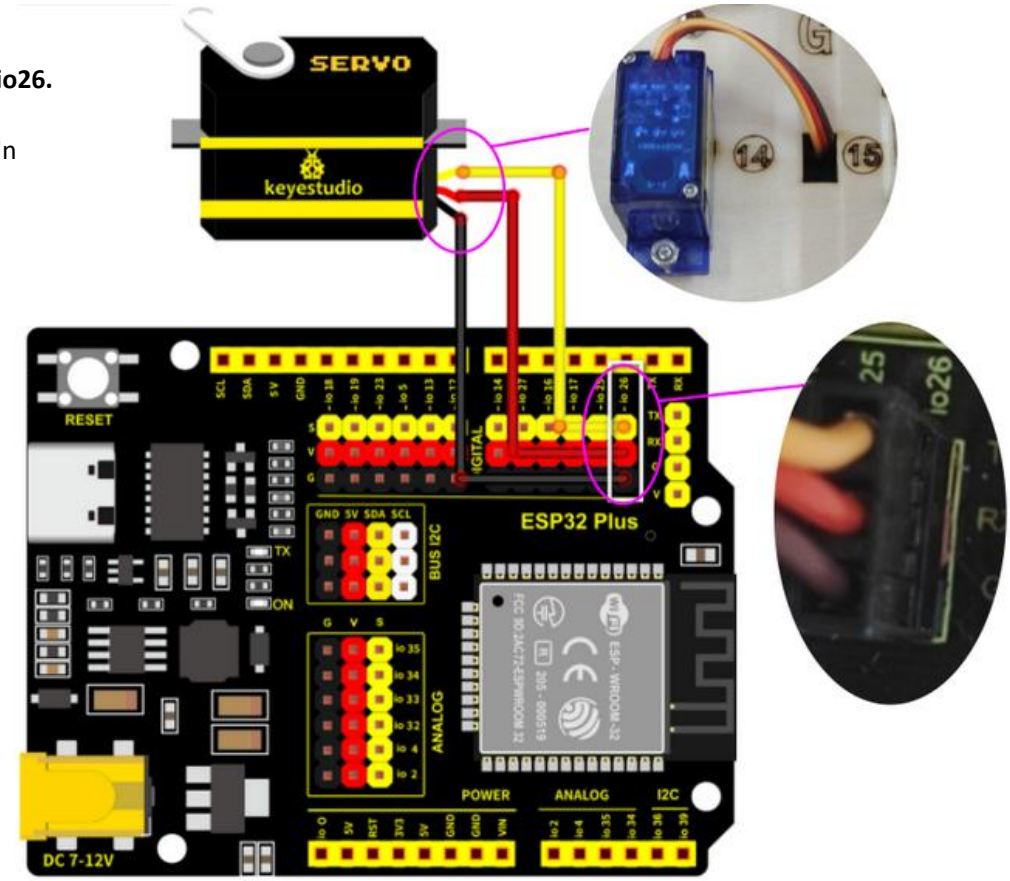
11. 10 Servo

Pass the wire of Servo through the Hole 15, and then connect it to ESP32 board.

Component	Wire	ESP32 Board Pin
Servo	3PIN	io26

Connect red to V, black to G, yellow to io26.

Servo Pin	Wire Color	ESP32 Board Pin
V	RED	V
G	BLACK	G
IO26	YELLOW	io26



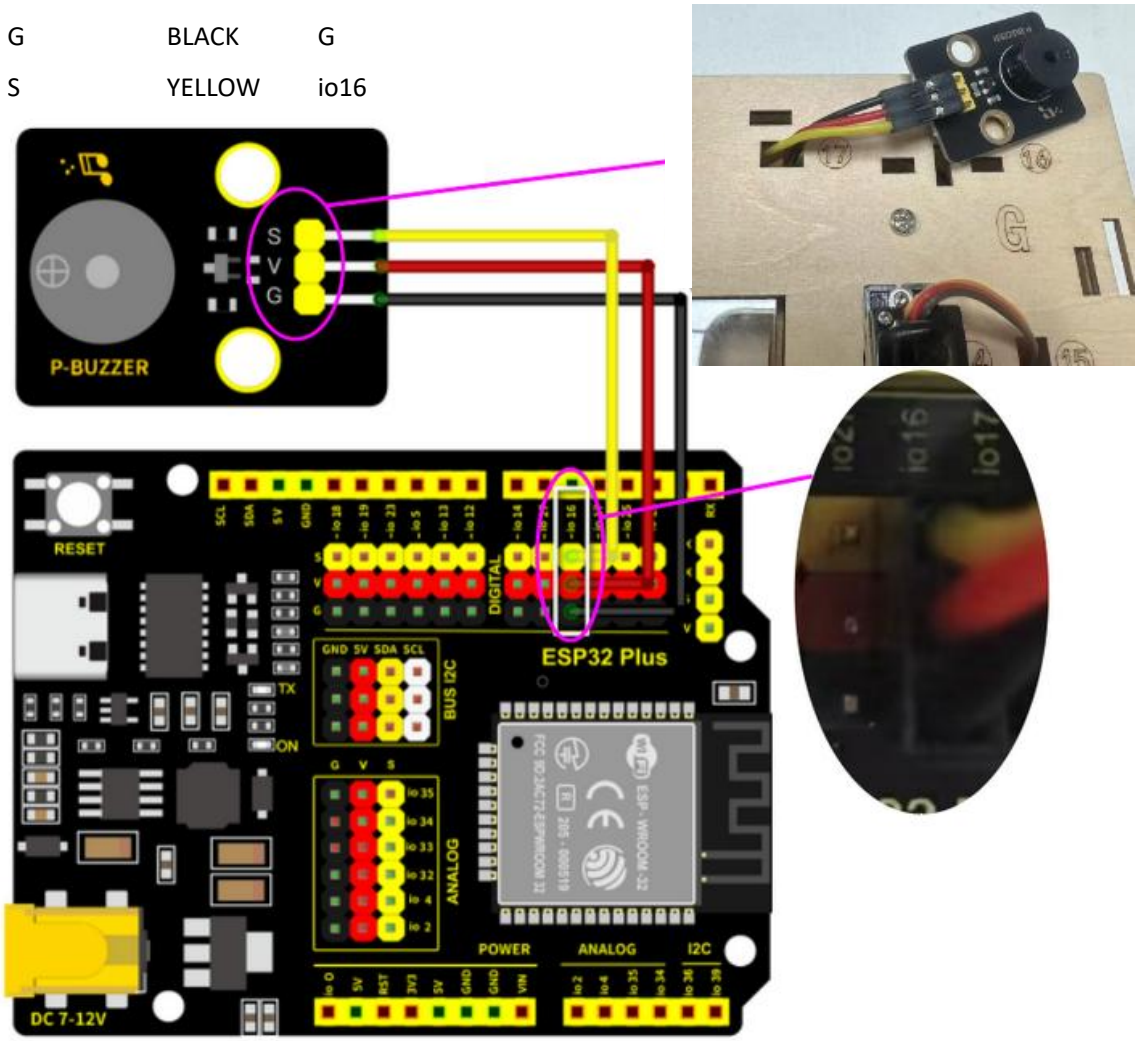
11.11 Buzzer

Pass the wire of Buzzer through the Hole 17, and then connect it to ESP32 board.

Component	Wire	ESP32 Board Pin
Buzzer	3PIN 20cm	

Connect red to V, black to G, yellow to io16.

Module Pin	Wire Color	ESP32 Board Pin
V	RED	V
G	BLACK	G
S	YELLOW	io16



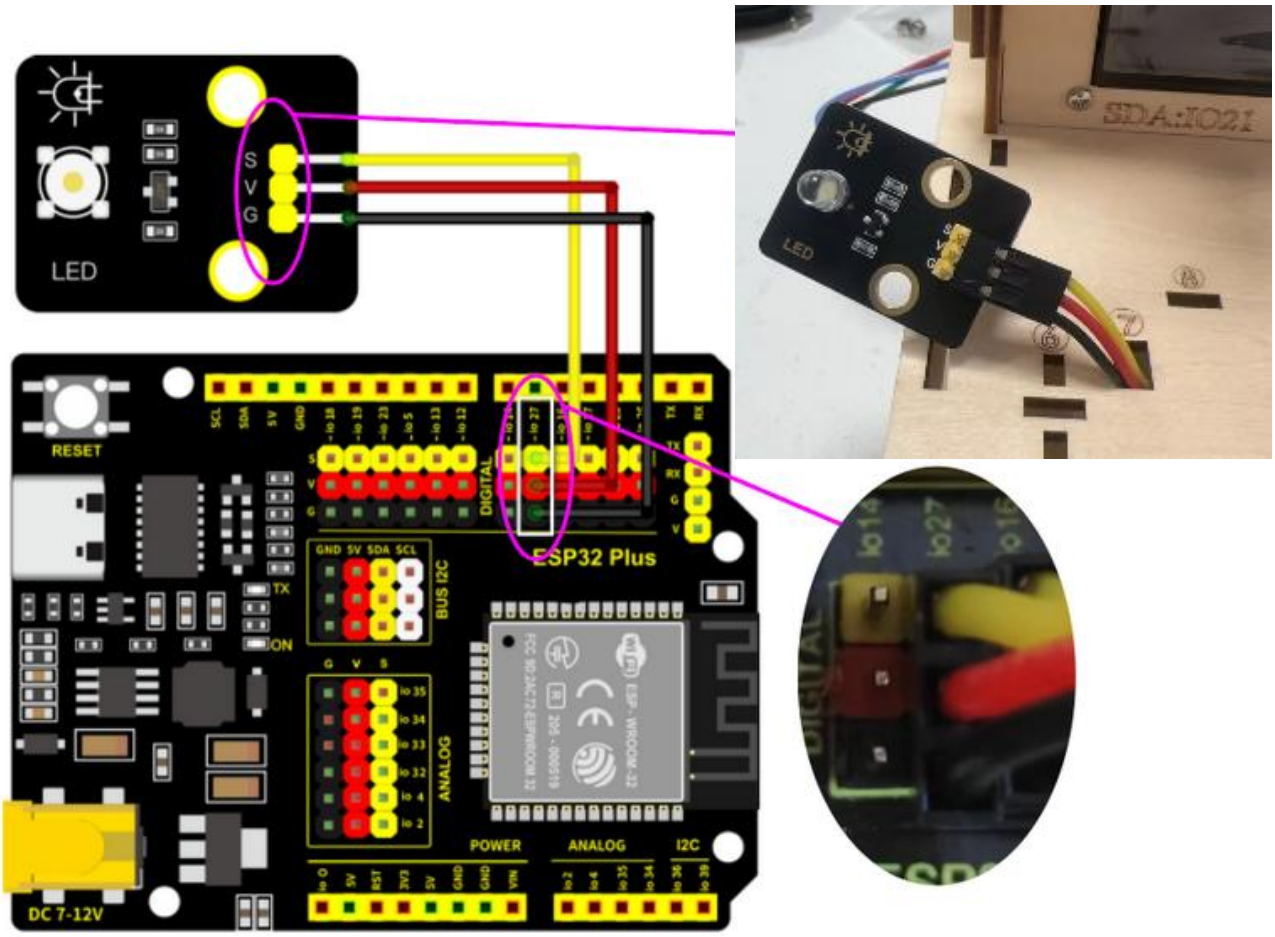
11.12 LED Module

Pass the wire of LED through the Hole 7, and then connect it to ESP32 board.

Component	Wire	ESP32 Board Pin
LED	3PIN 20cm	io27

Connect red to V, black to G, yellow to io27.

Module Pin	Wire Color	ESP32 Board Pin
V	RED	V
G	BLACK	G
S	YELLOW	io27



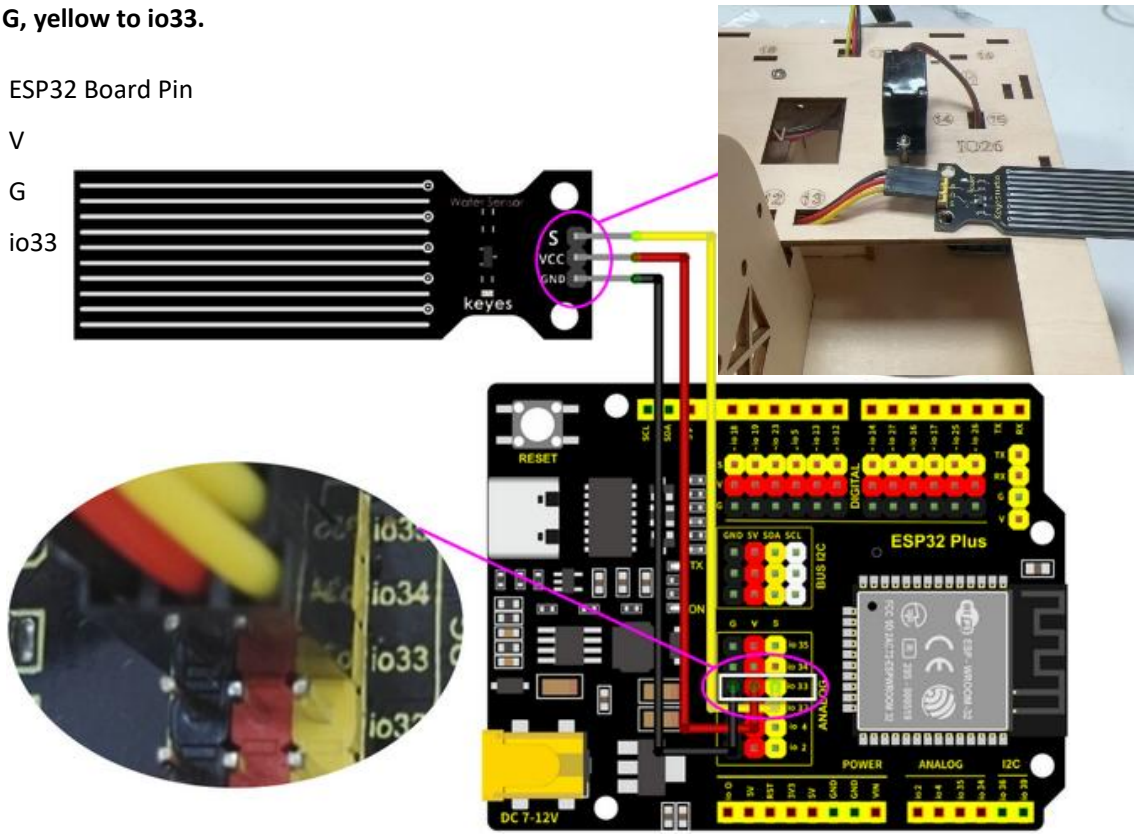
11.13 Water Lever Sensor

Pass the wire of water level sensor through the Hole 13, and then connect it to ESP32 board.

Component	Wire	ESP32 Board Pin
Water Lever Sensor	**3PIN 25cm**	io33

Connect red to V, black to G, yellow to io33.

Module Pin	Wire Color	ESP32 Board Pin
V	RED	V
G	BLACK	G
S	YELLOW	io33



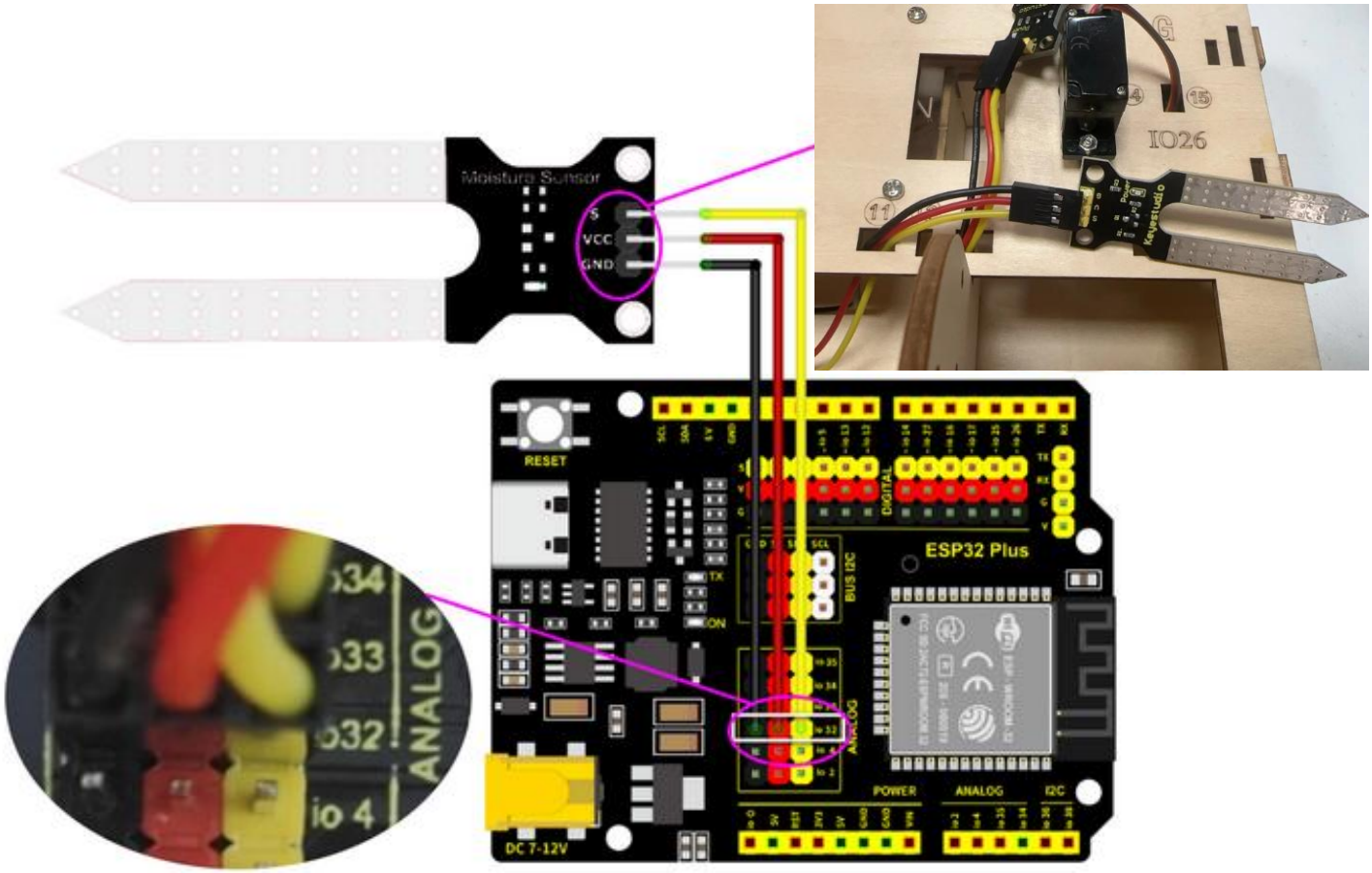
11.14 Soil Humidity Sensor

Pass the wire of soil humidity sensor through the Hole 11, and then connect it to ESP32 board.

Component	Wire	ESP32 Board Pin
Soil Humidity Sensor	3PIN 20cm	io32

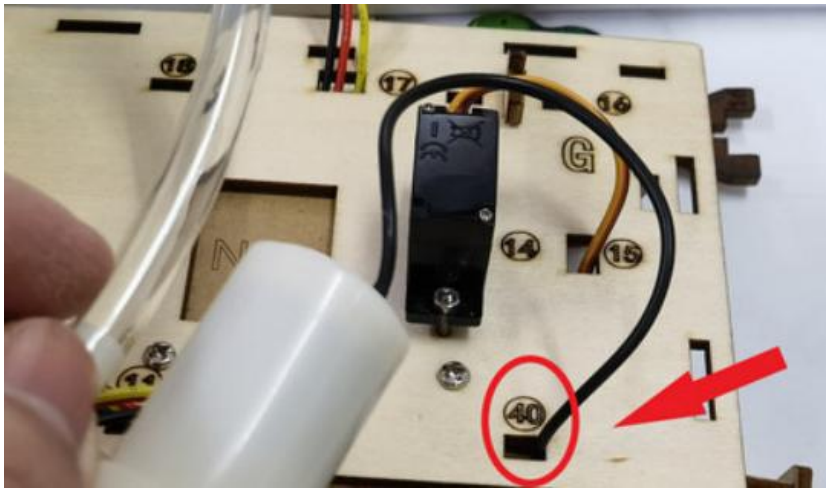
Connect red to V, black to G, yellow to io32.

Module Pin	Wire Color	ESP32 Board Pin
V	RED	V
G	BLACK	G
S	YELLOW	io32



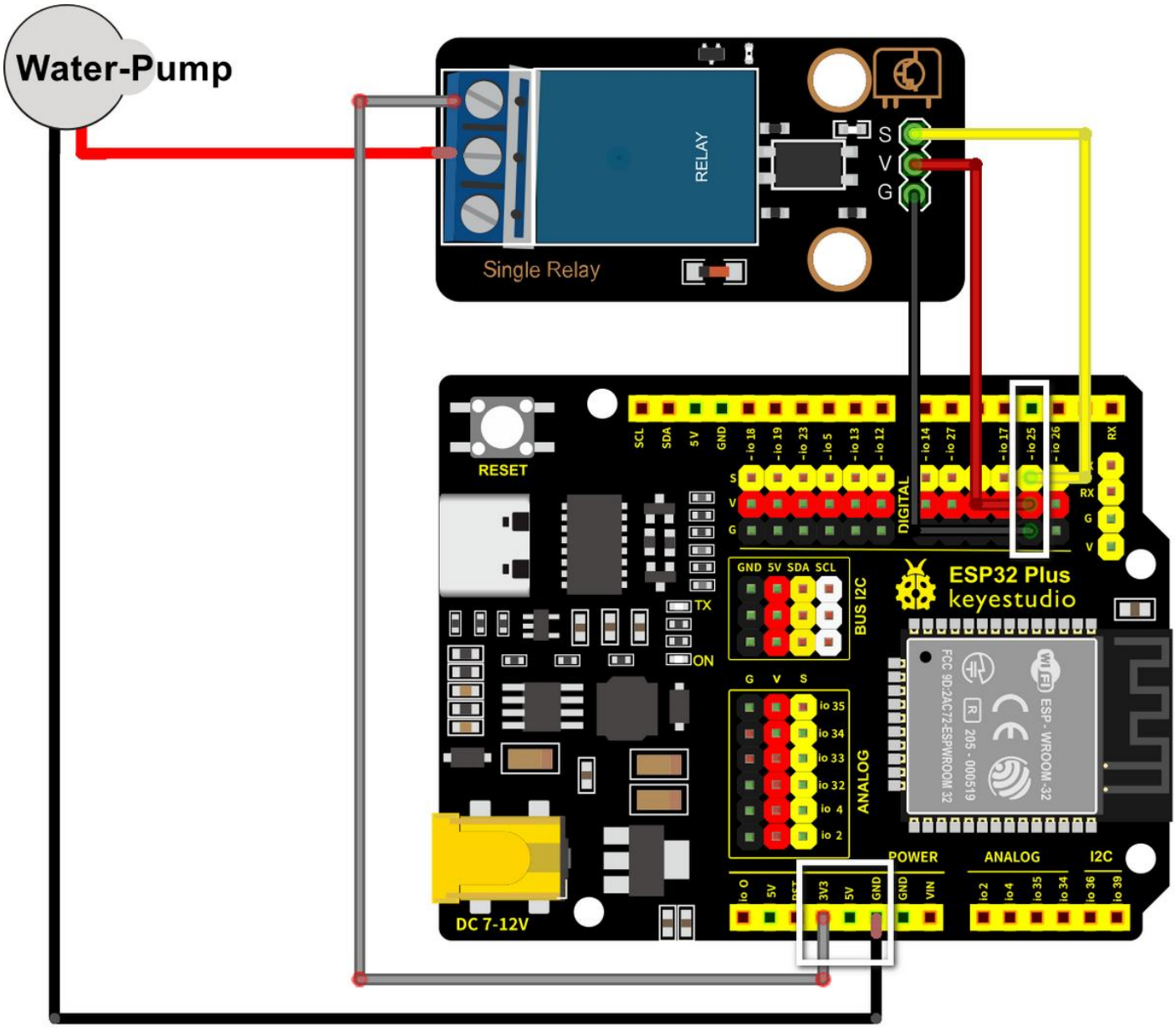
11.14 Relay Module and Water Pump

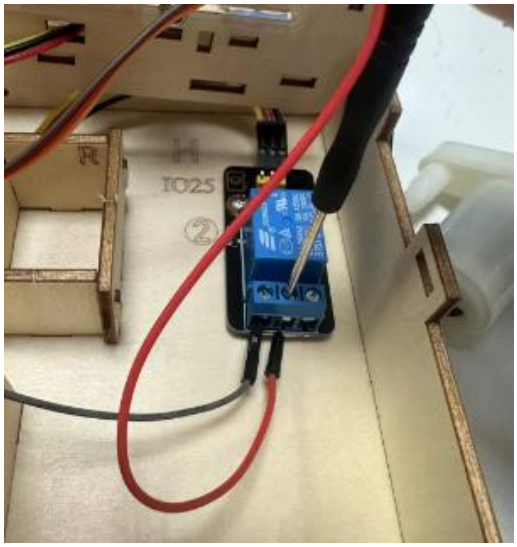
Pass the wire of Water Pump through the Hole 40 in the way as shown below:



The red wire of the water pump is connected to the middle terminal of the relay module, and the black wire is connected to the GND of the ESP32 board.

In addition, you need to use a Dupont wire to connect the left terminal of the relay module to the 3.3V of the ESP32.

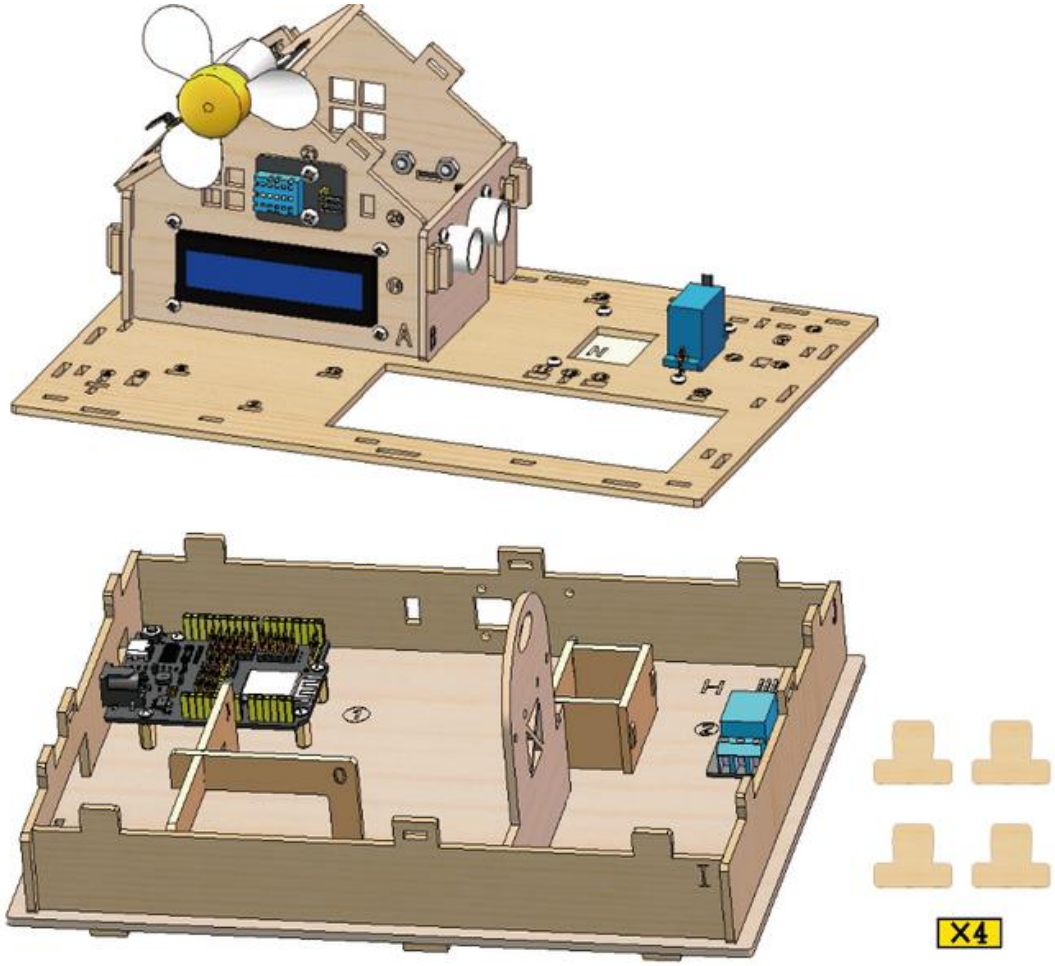




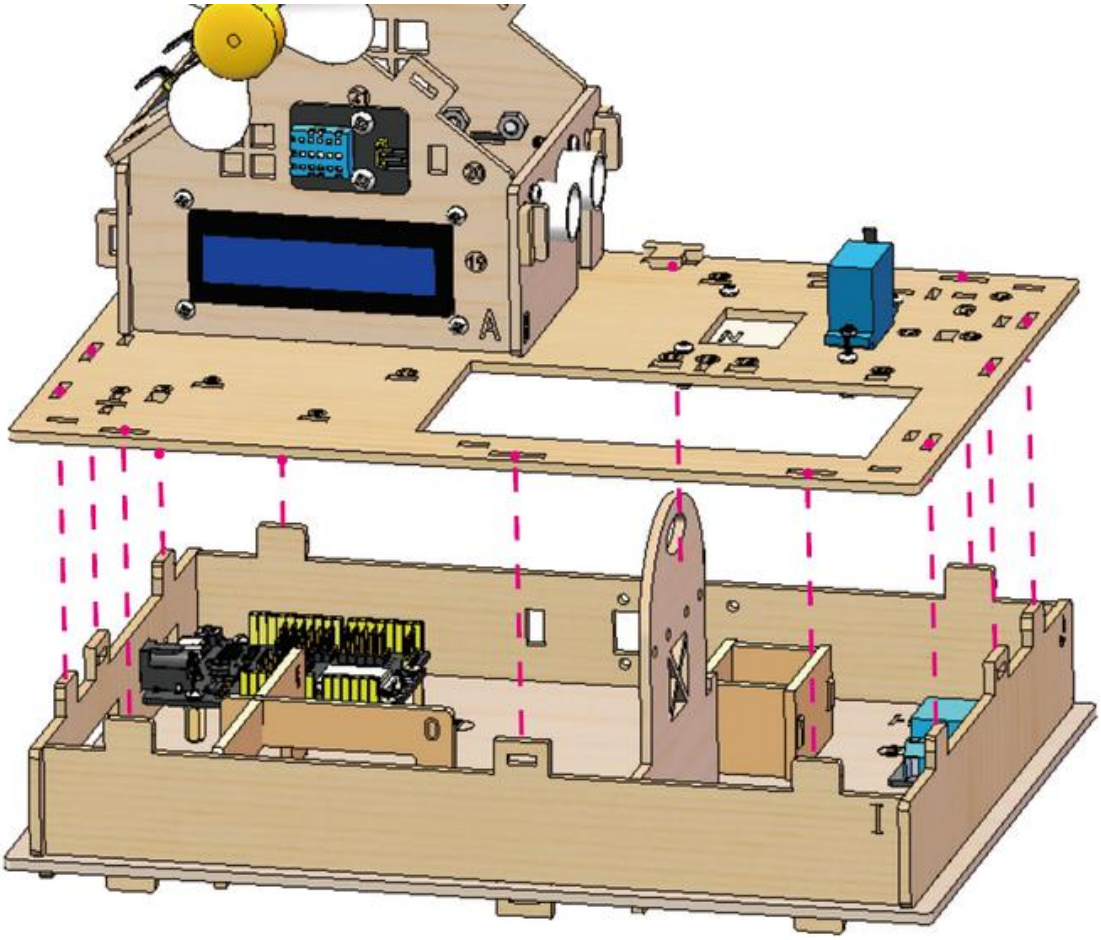
Insert the male terminal of the Dupont wire into the female terminal of the relay module and tighten it with a screwdriver.

Step 12 Install the house and foundation

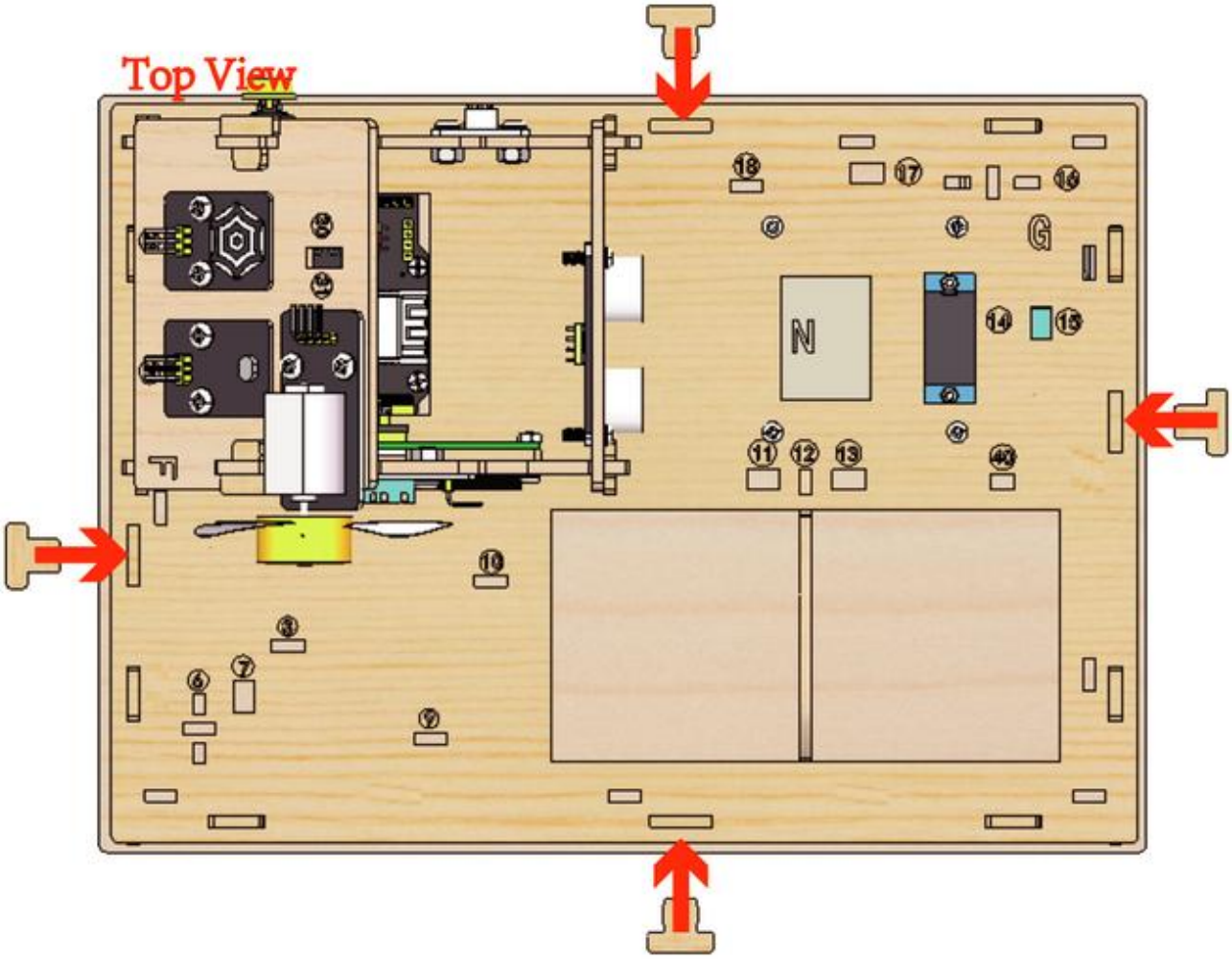
12.1 Required component



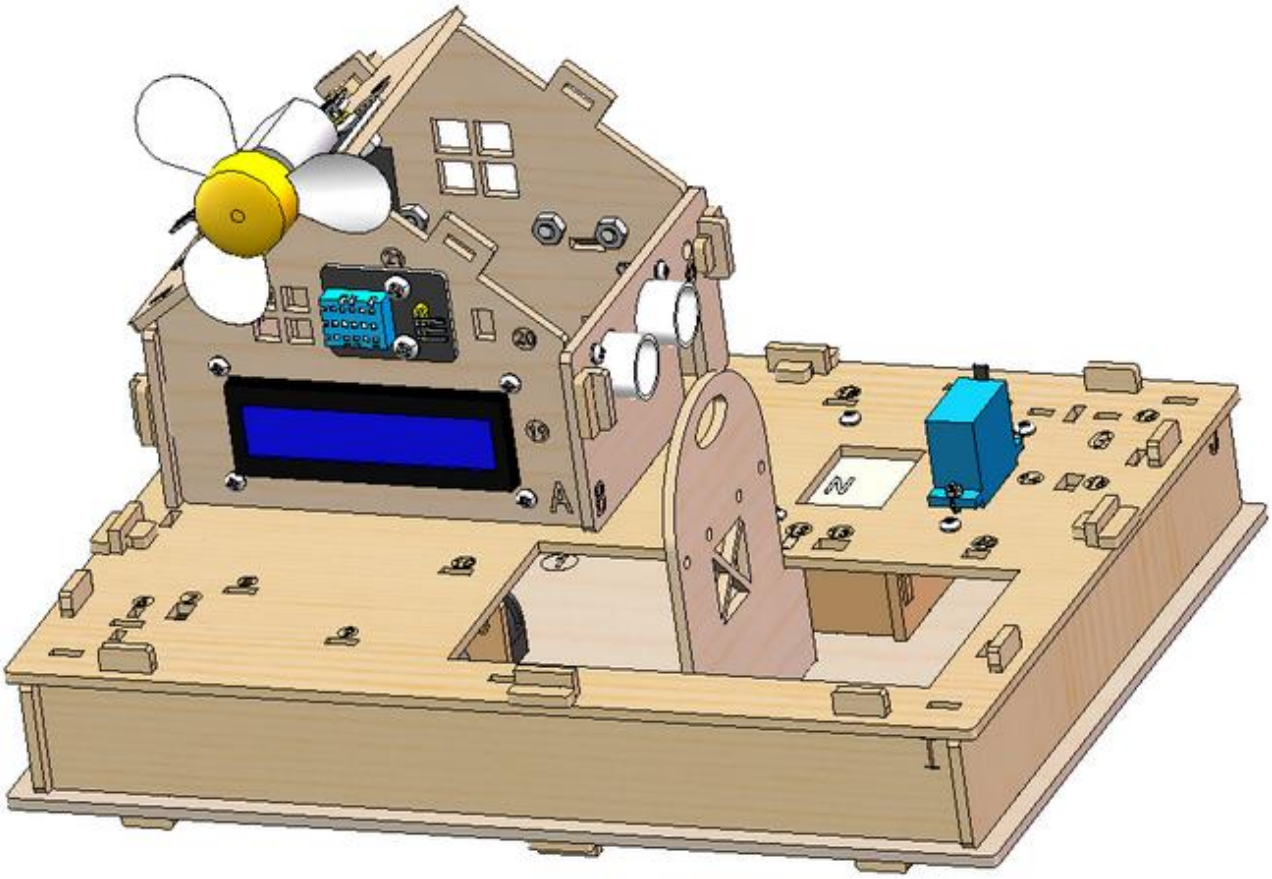
12.2



12.3

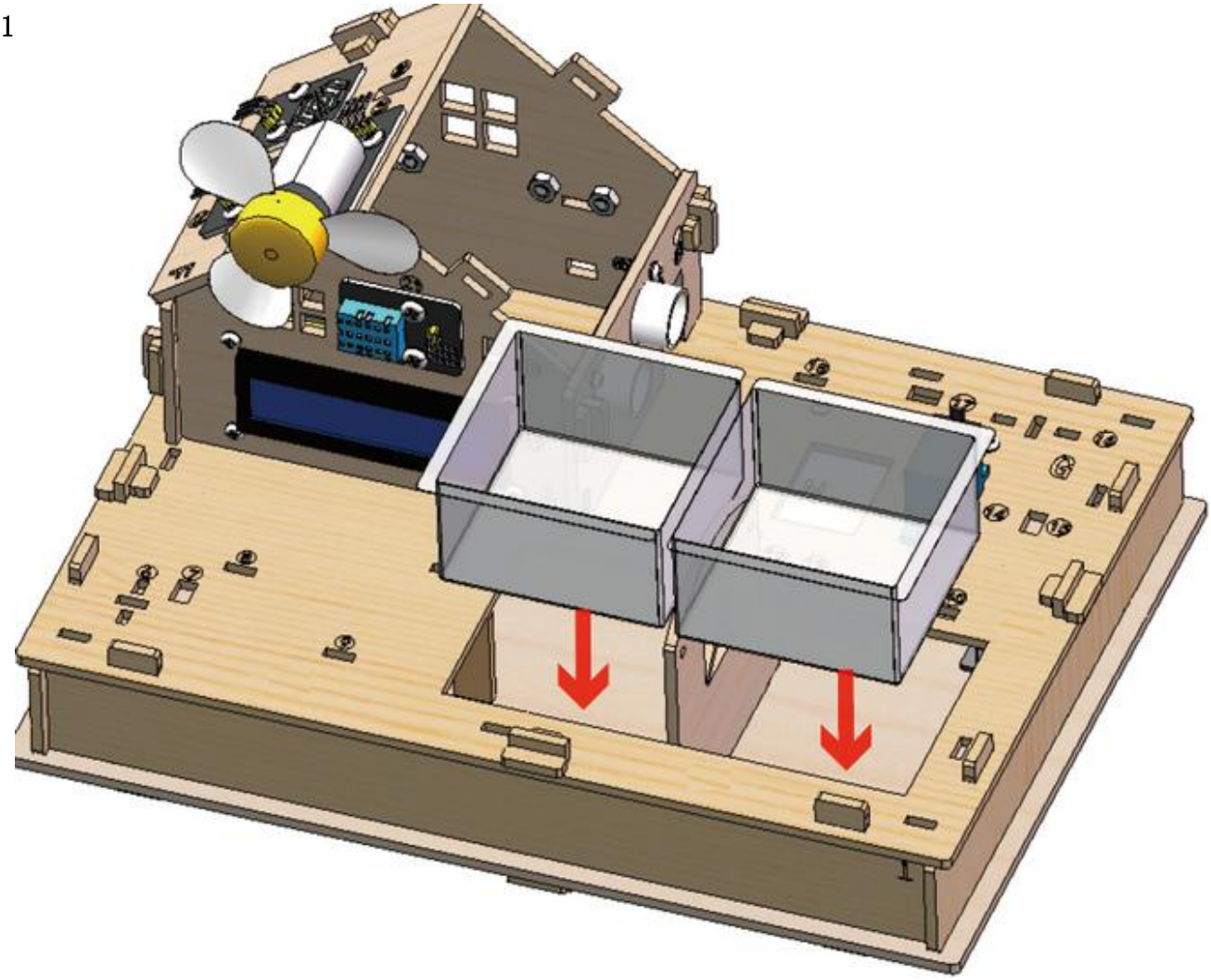


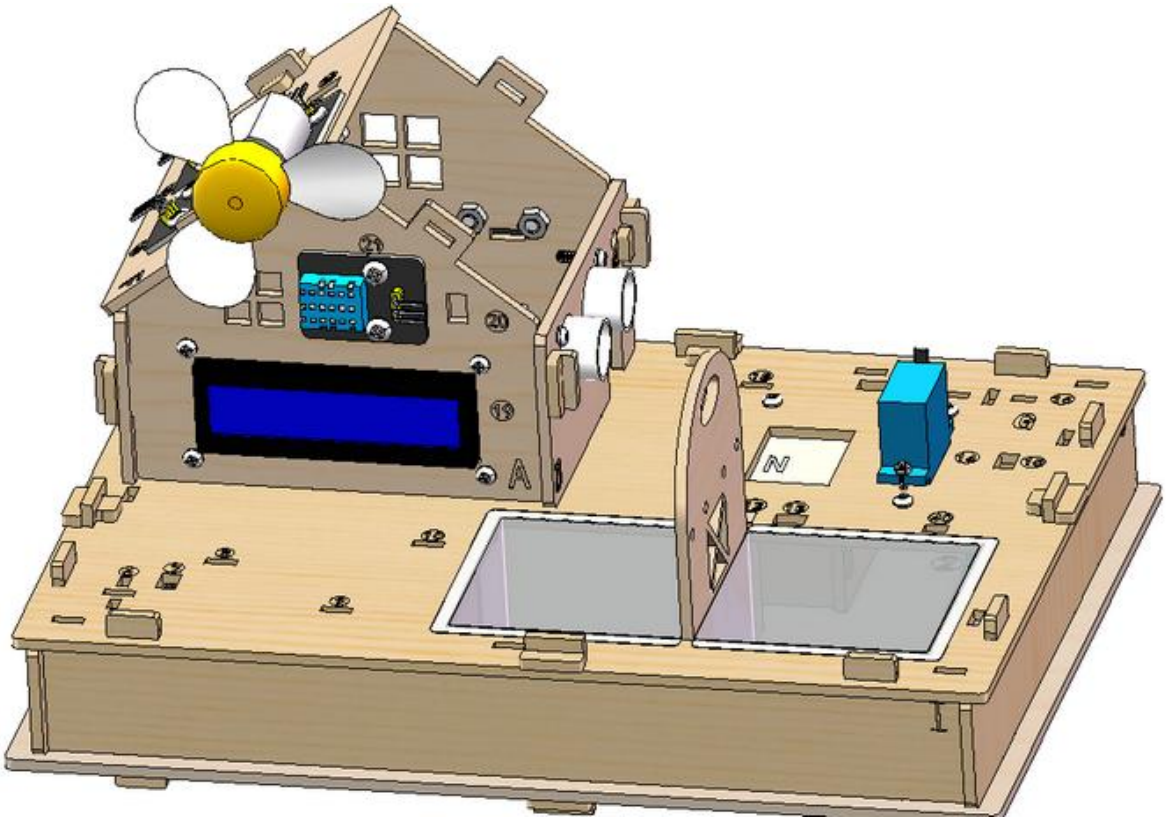
12.4



Step 13 Install the Plastic Sinks

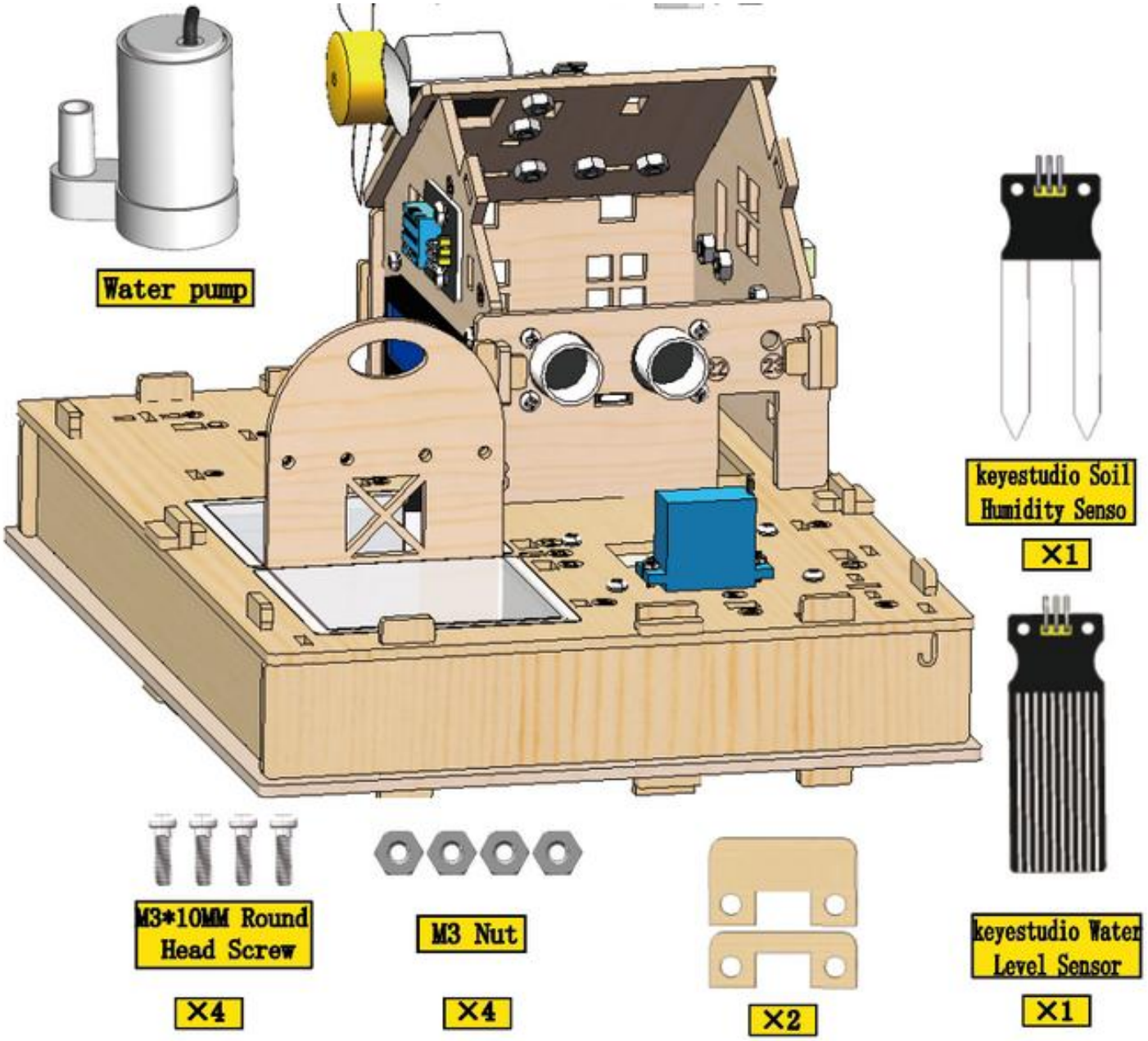
13. 1



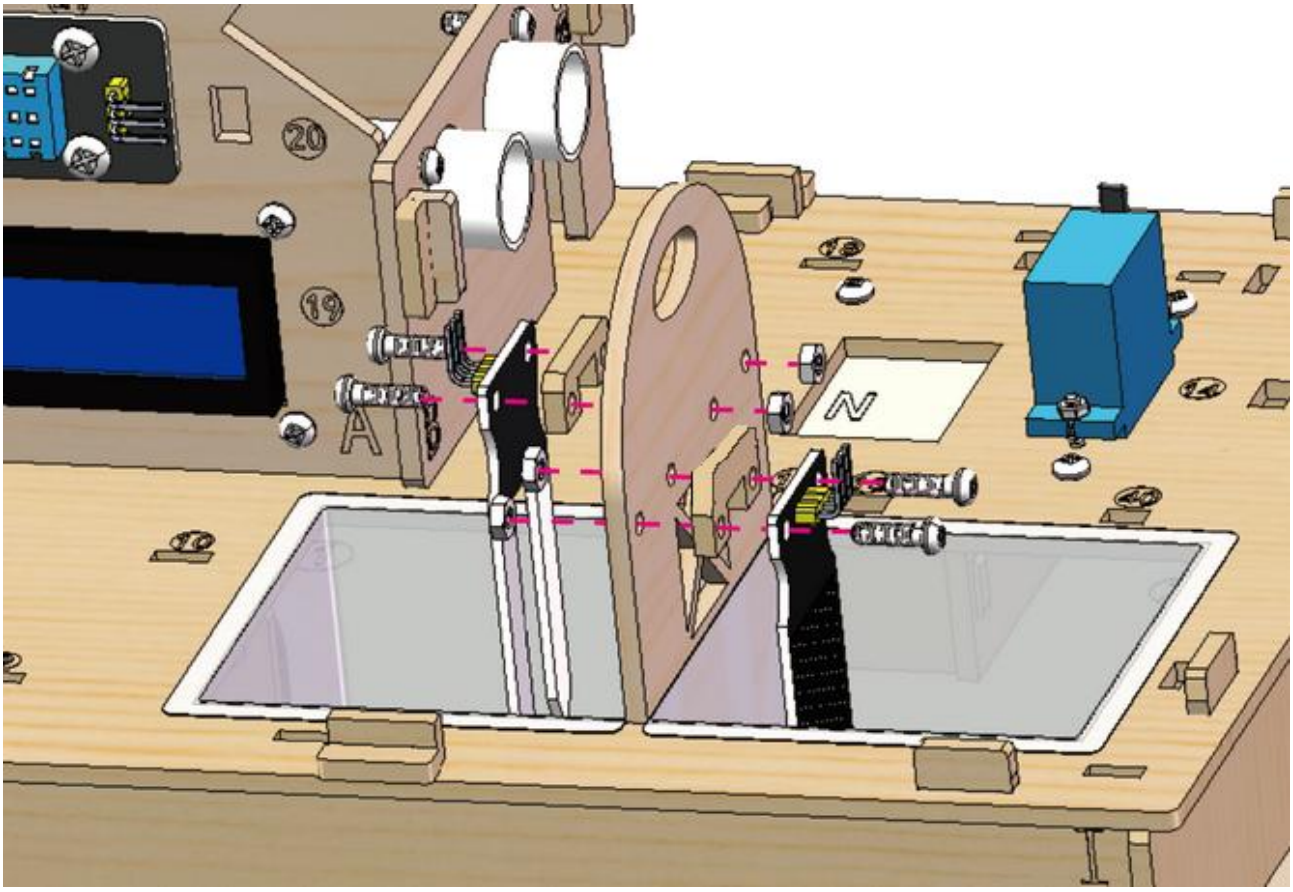


Step 14 Install the soil module and water level module

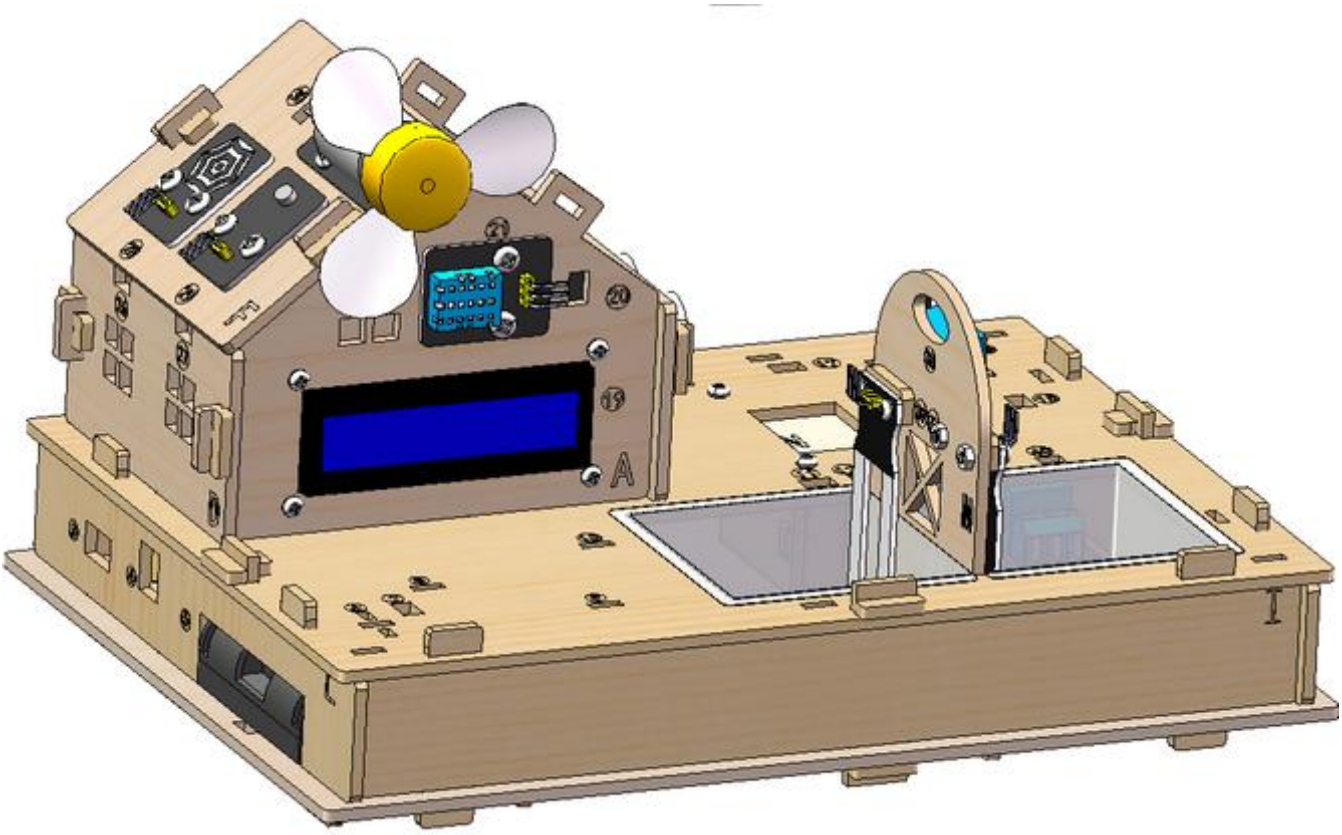
14.1 Required components



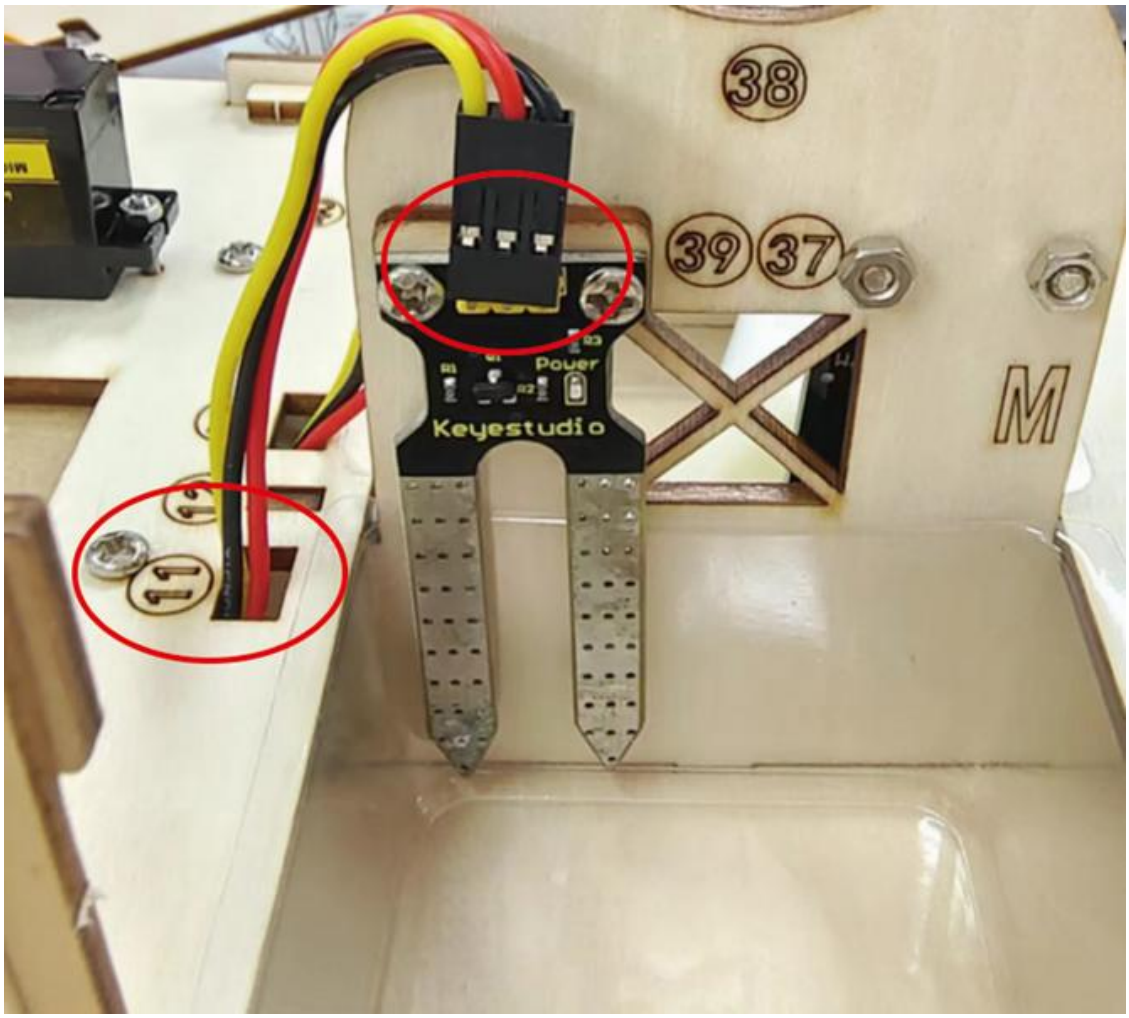
14.2



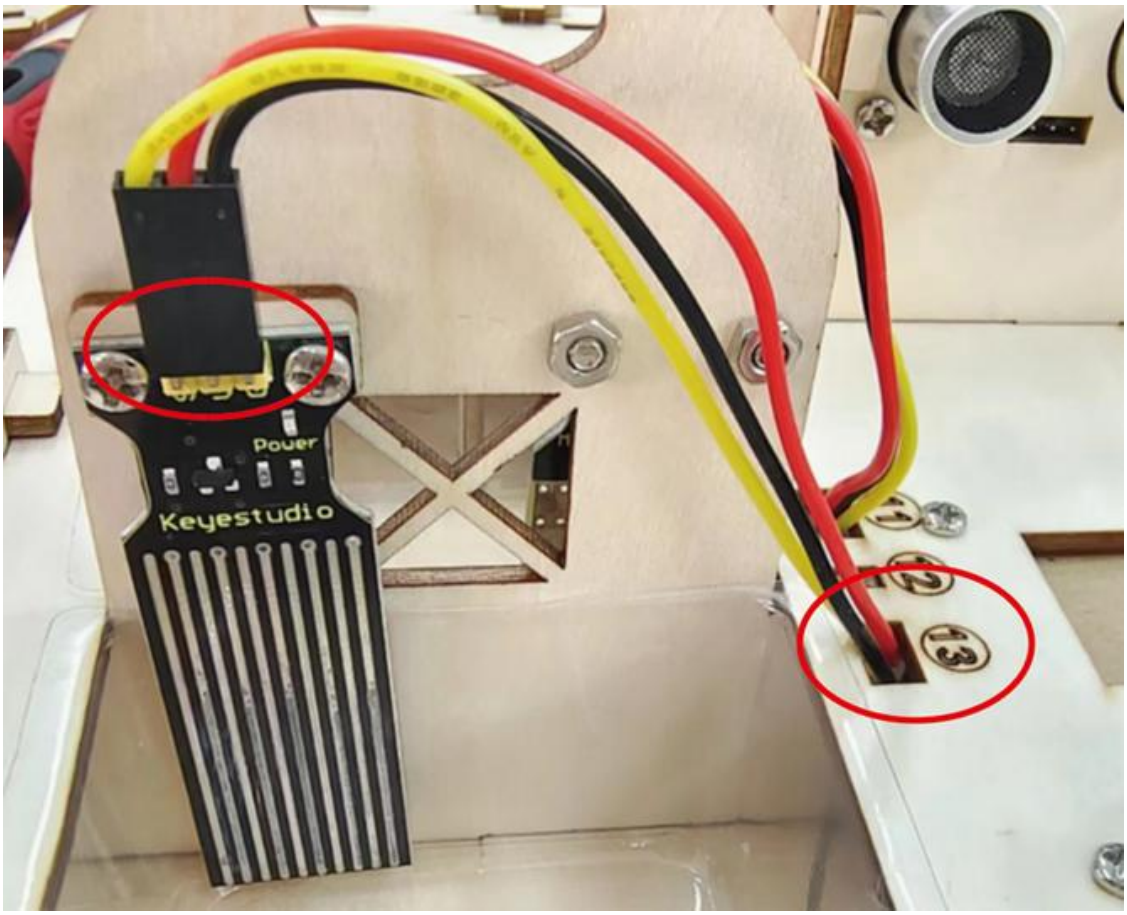
14.3



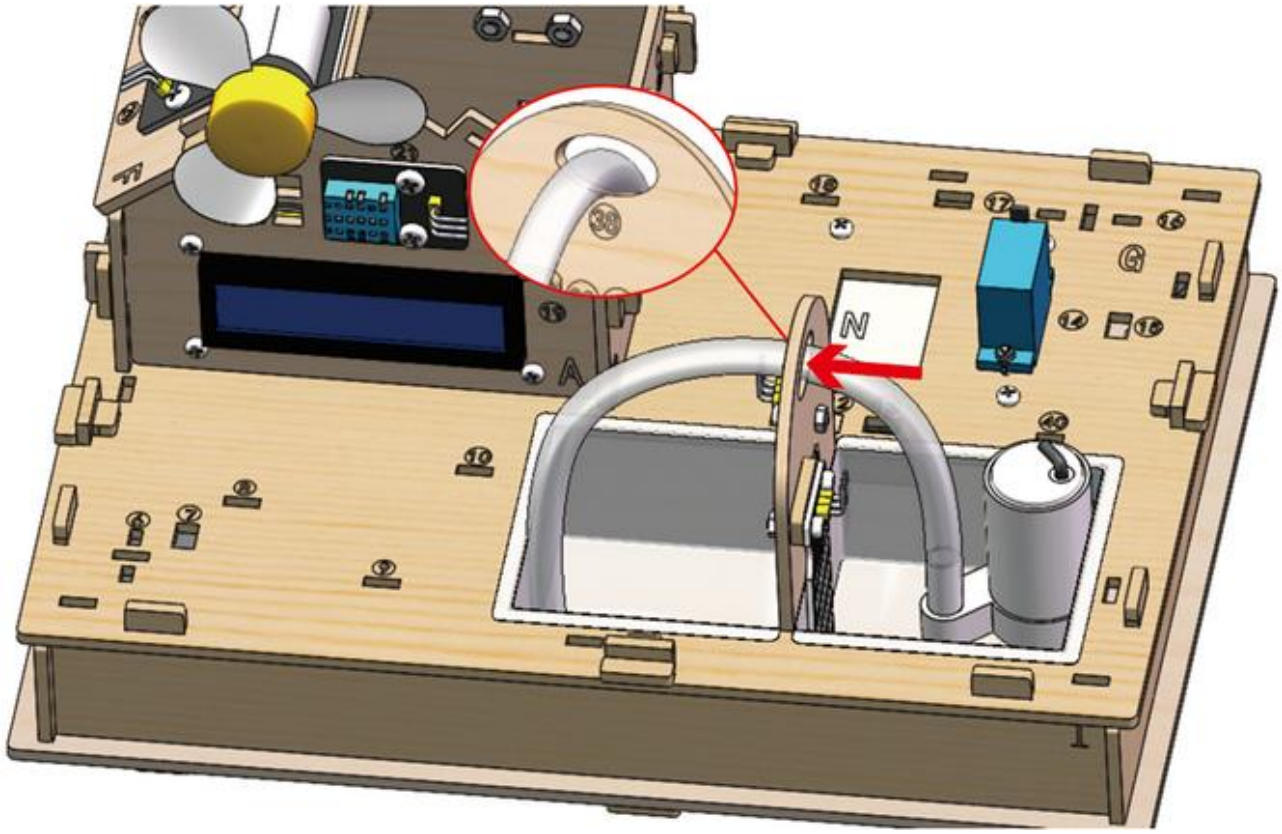
14. 4



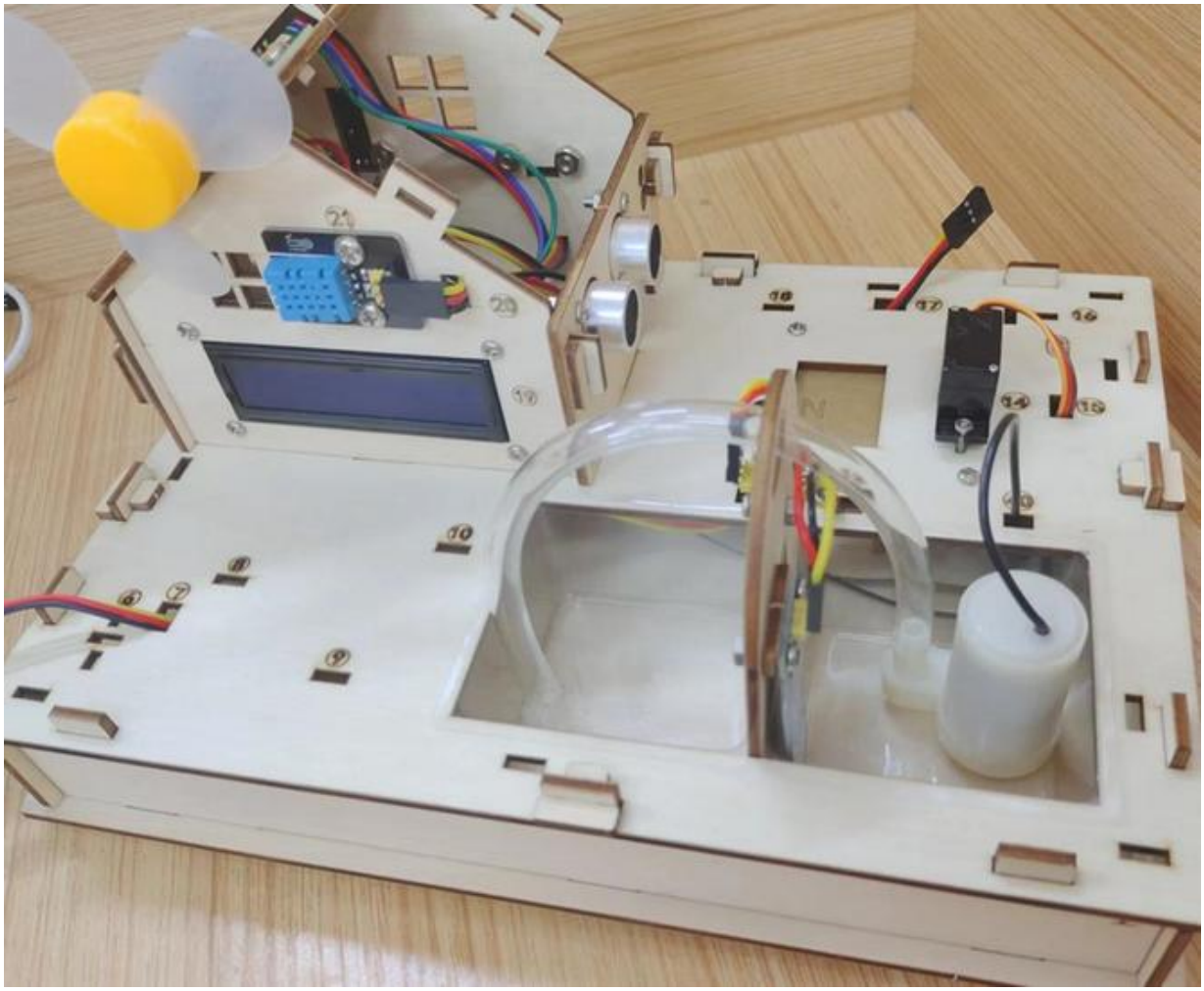
14. 5



14. 6

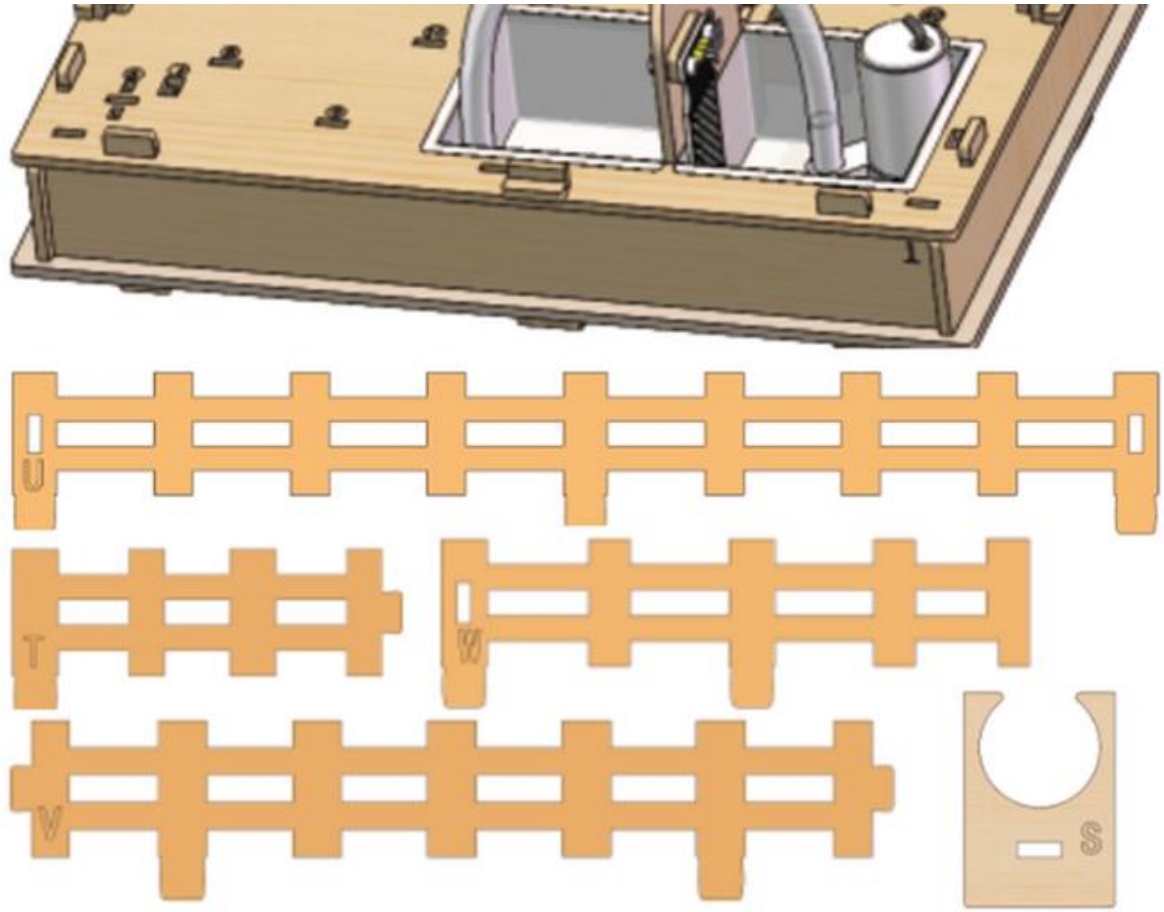


14. 7



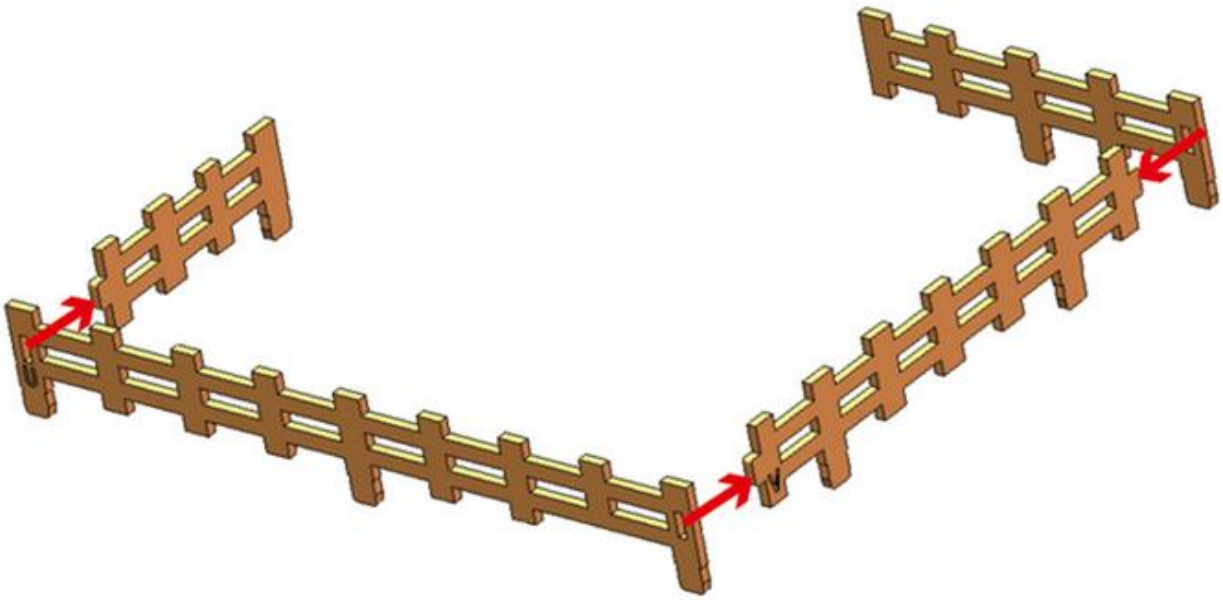
Step 15 Install fence

15. 1

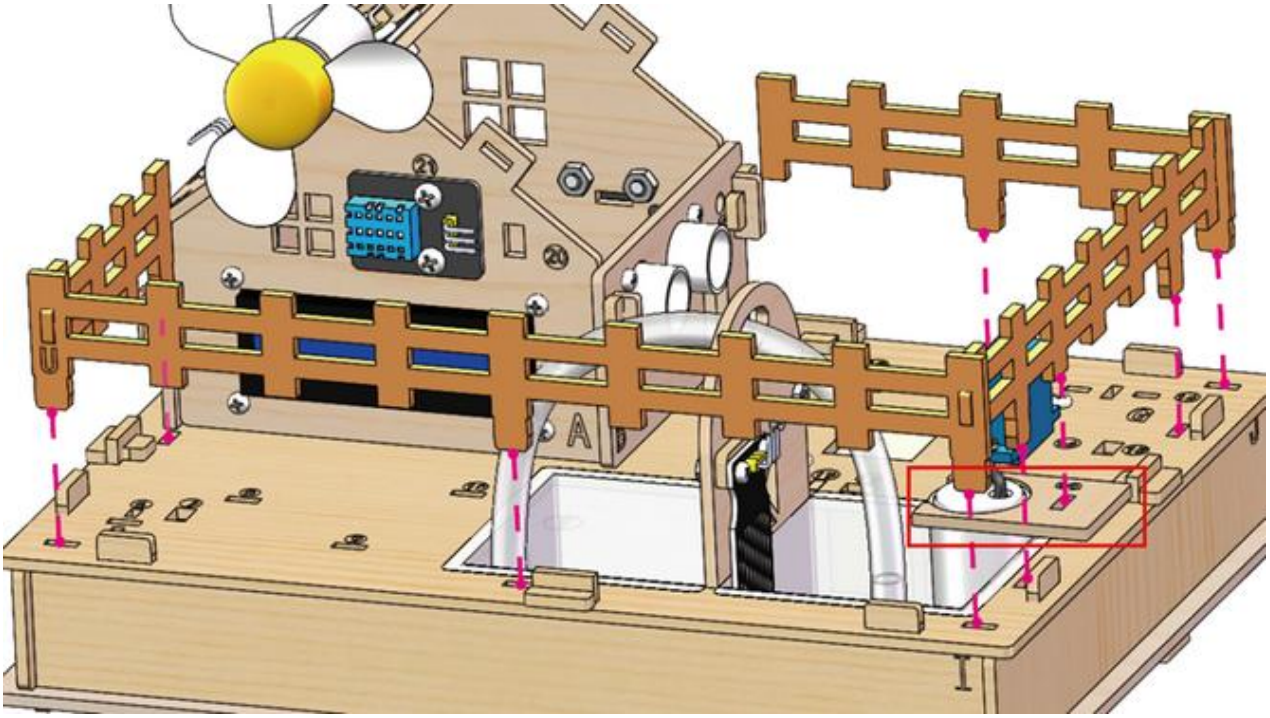


---

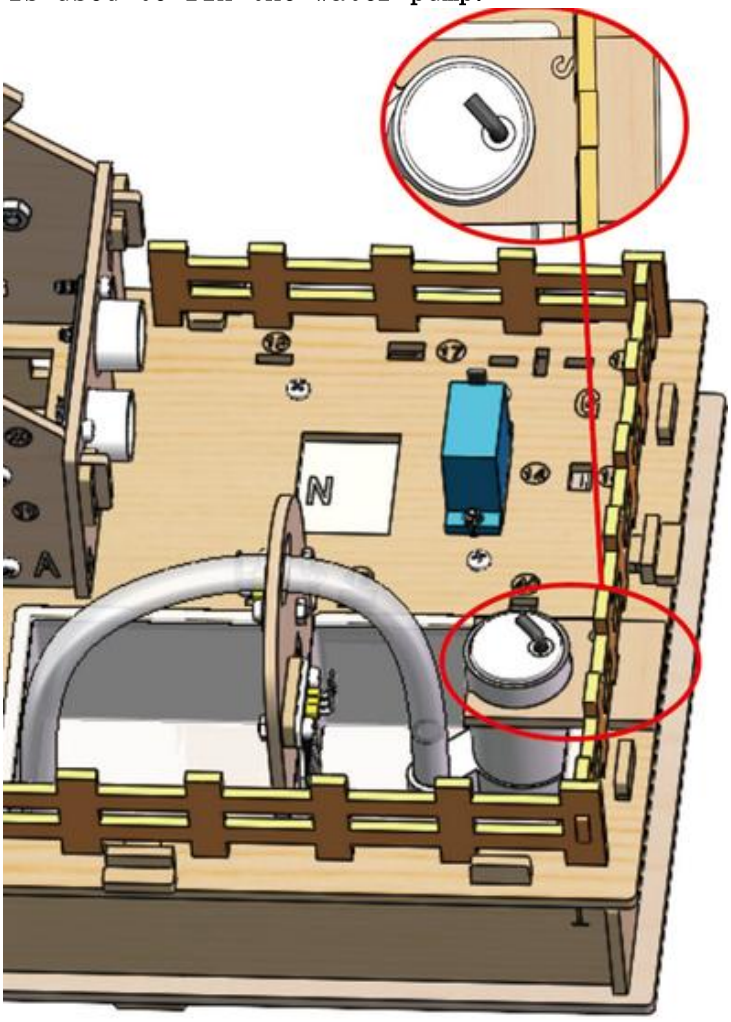
15. 2



15.3

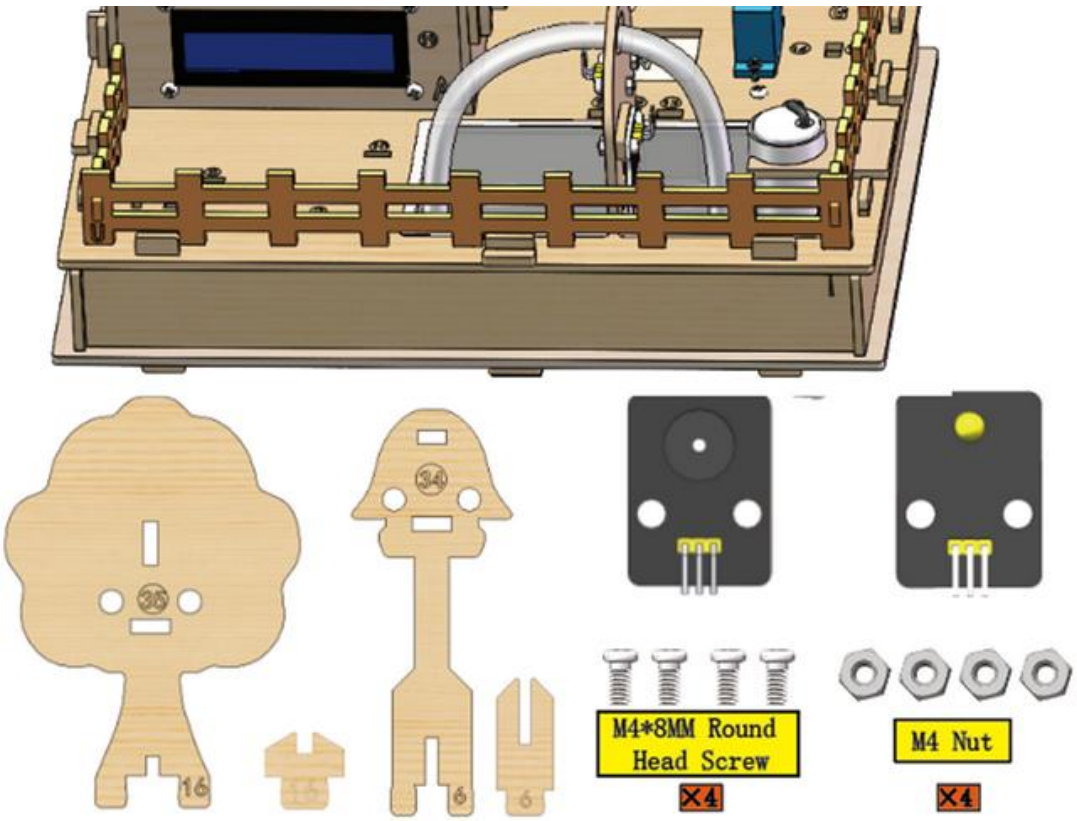


15.4 This wooden board is used to fix the water pump.

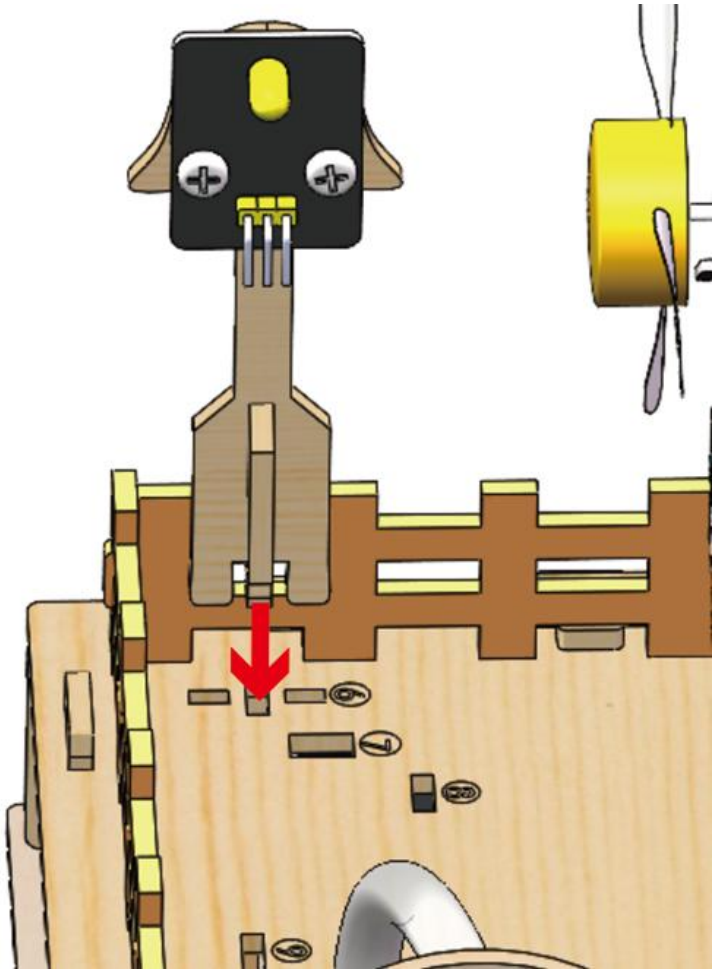


Step 16 Install the Buzzer and the Led Module

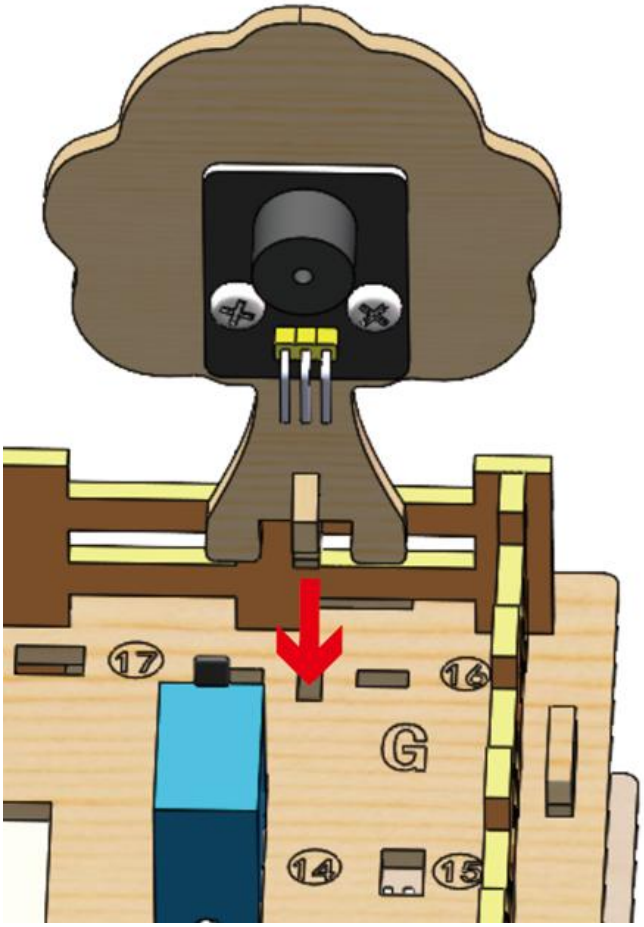
16.1 Required components



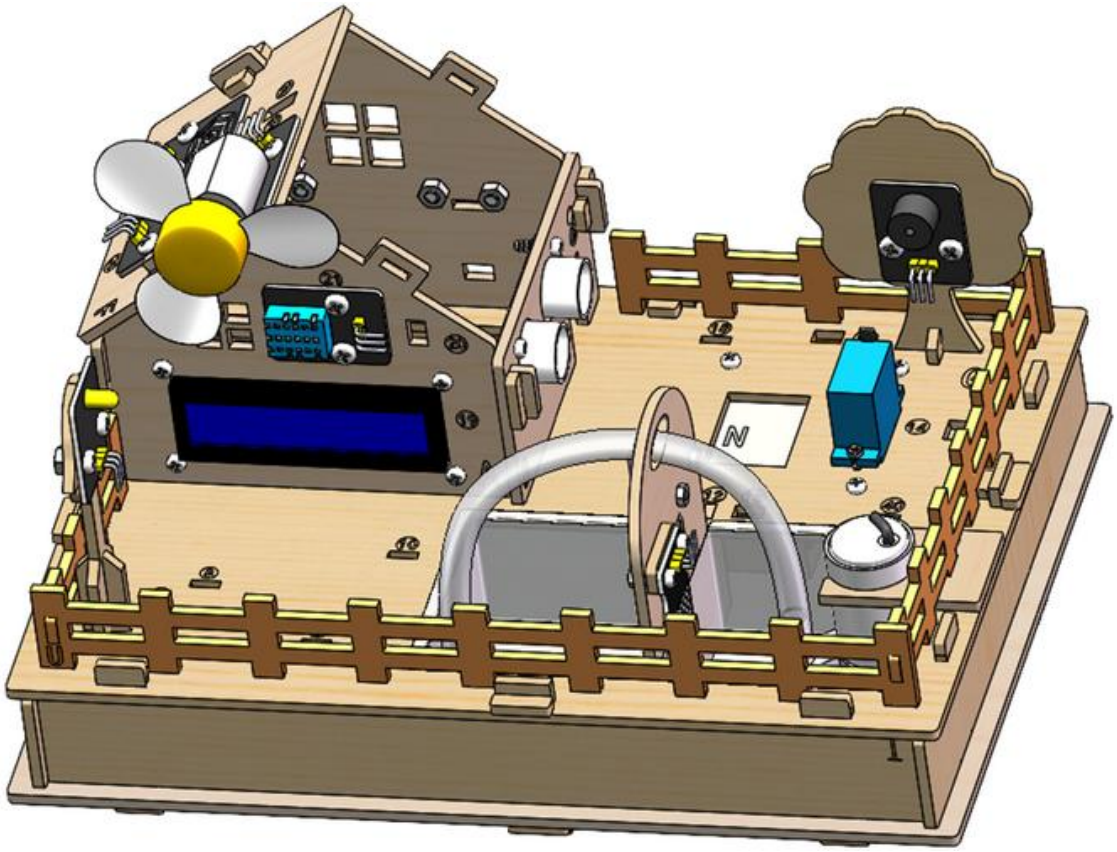
16.2



16. 3

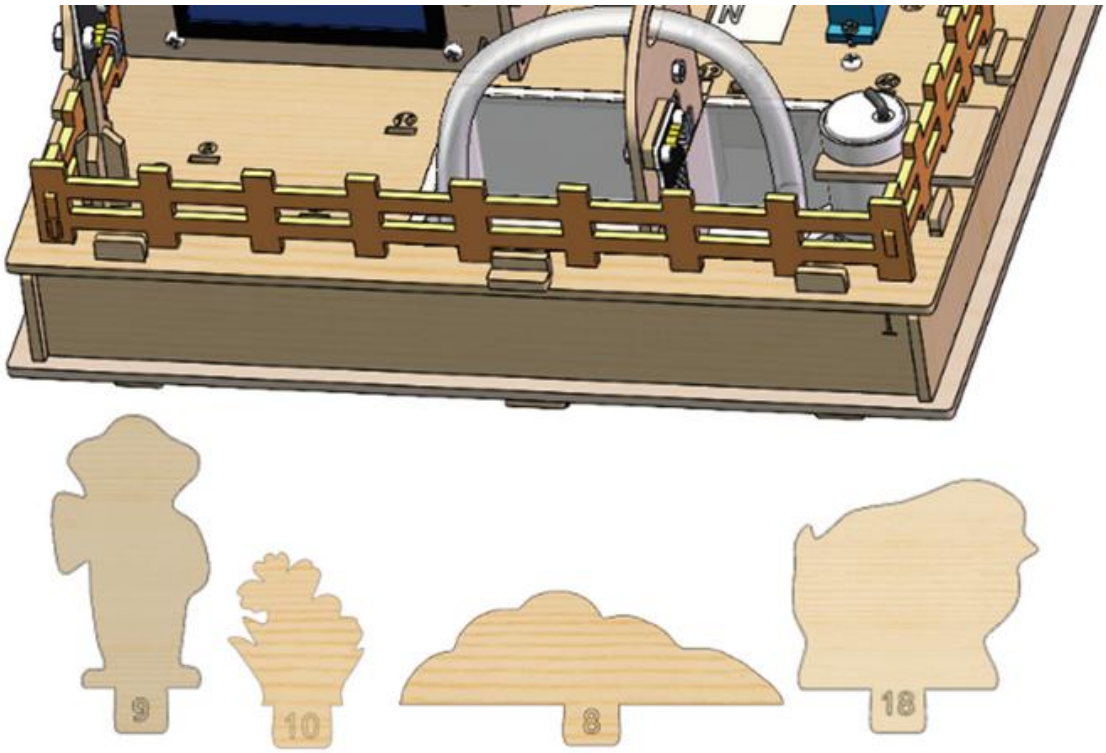


16. 4

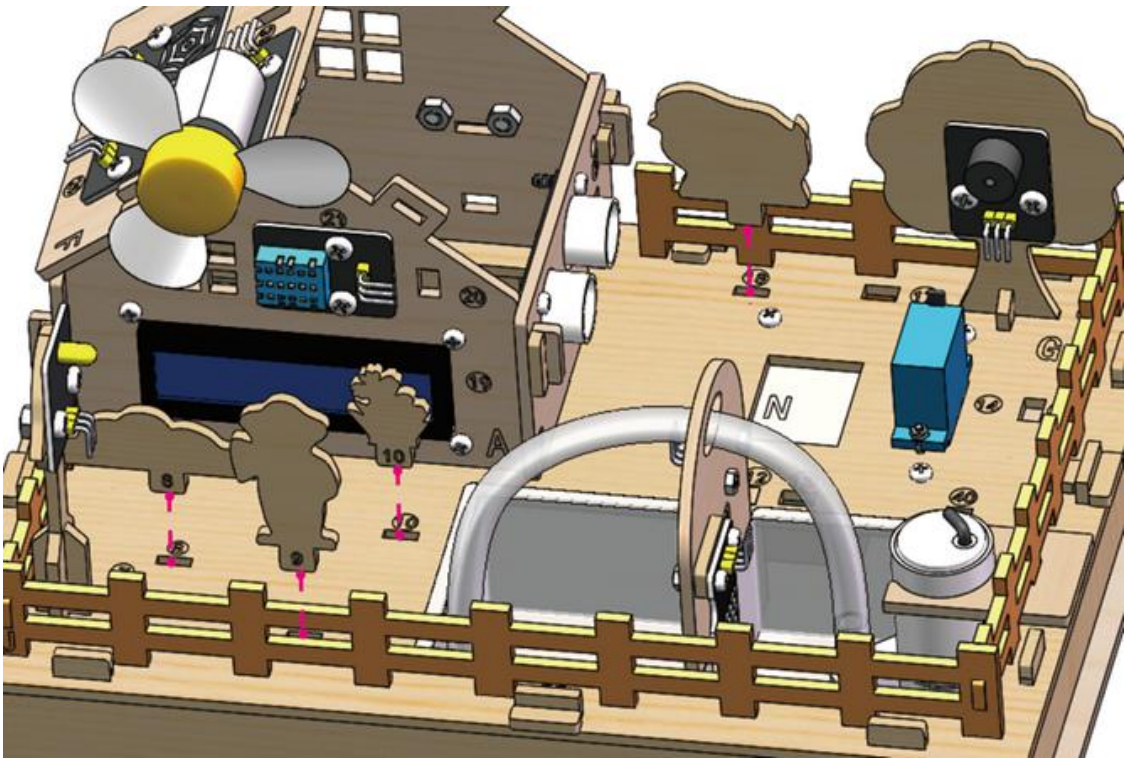


Step 17 Decorate the House

17.1 Required components

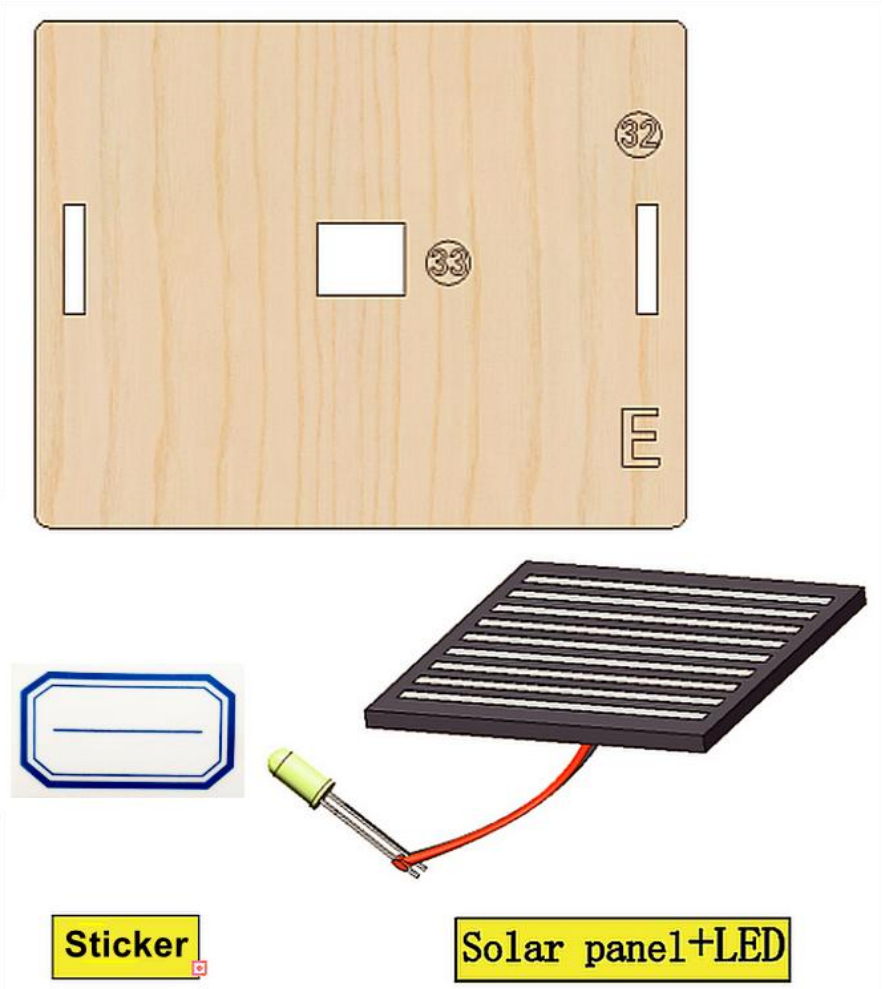


17.2



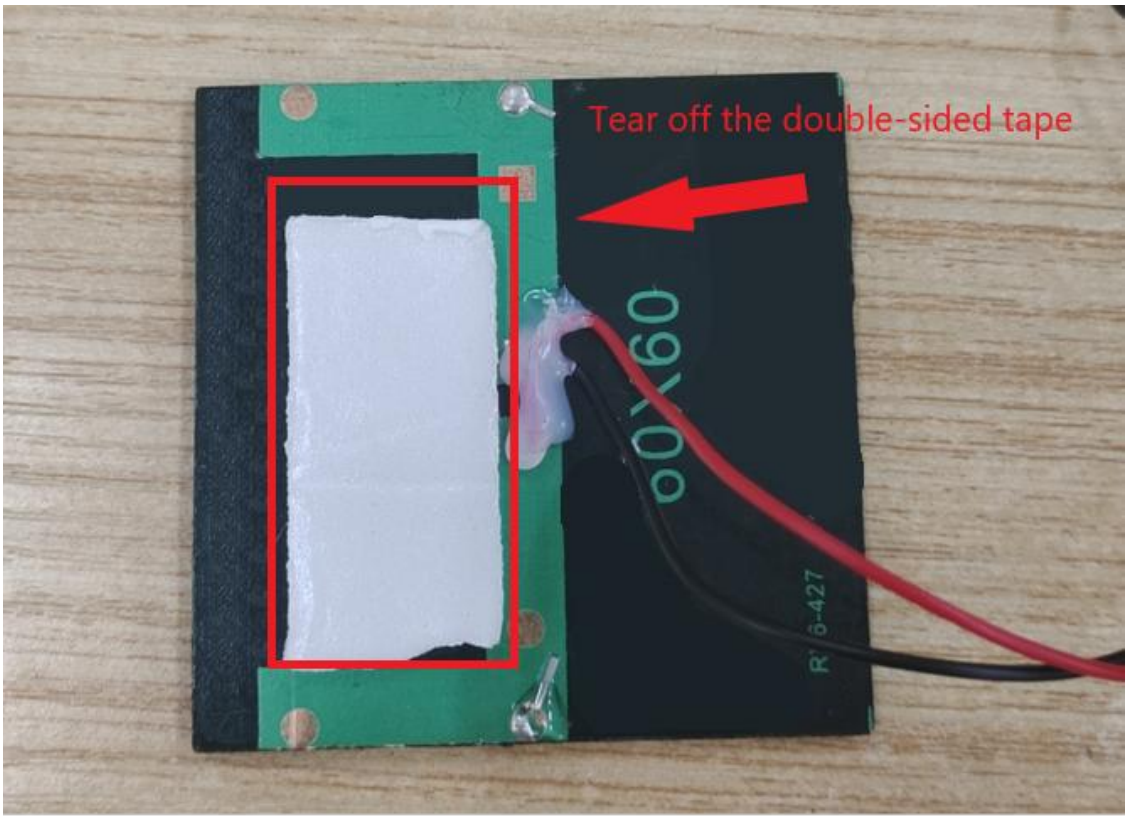
Step 18 Install Solar Panel

18. 1

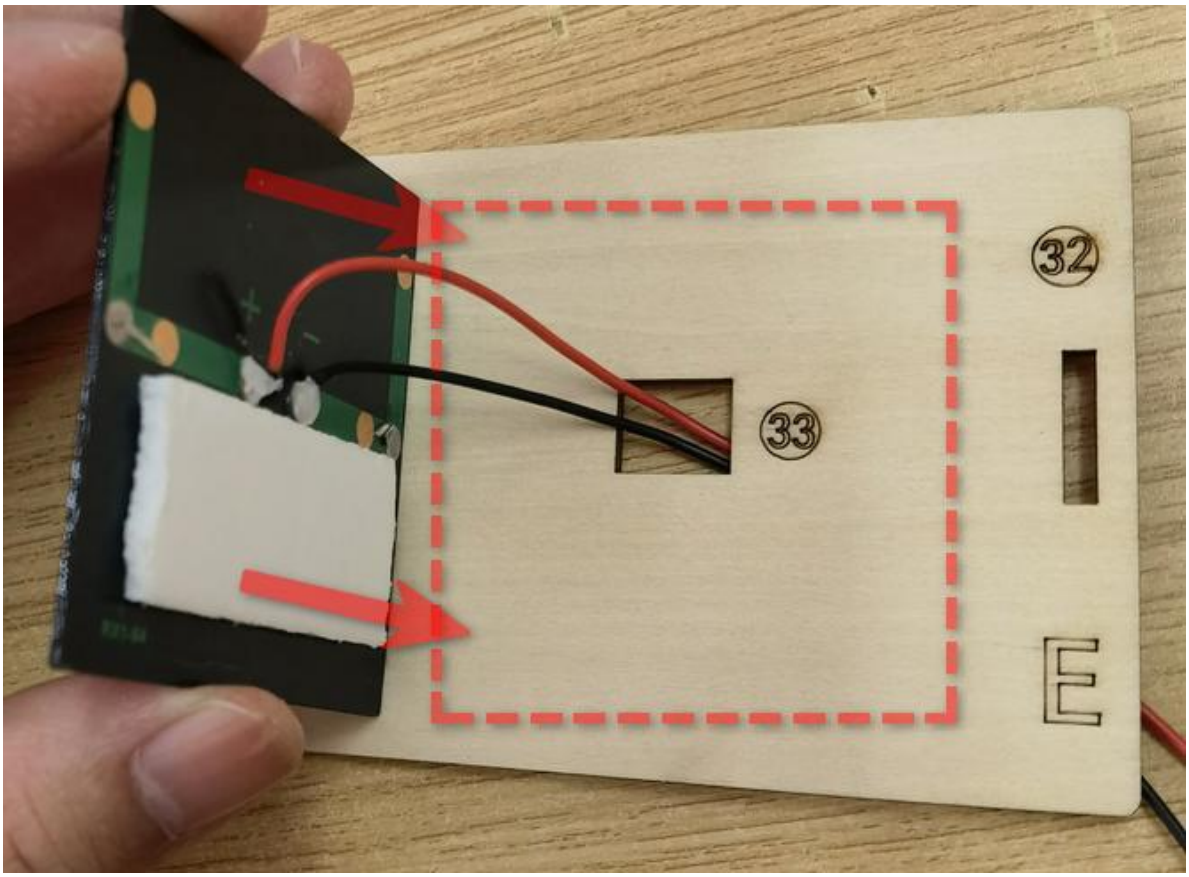


---

18. 2

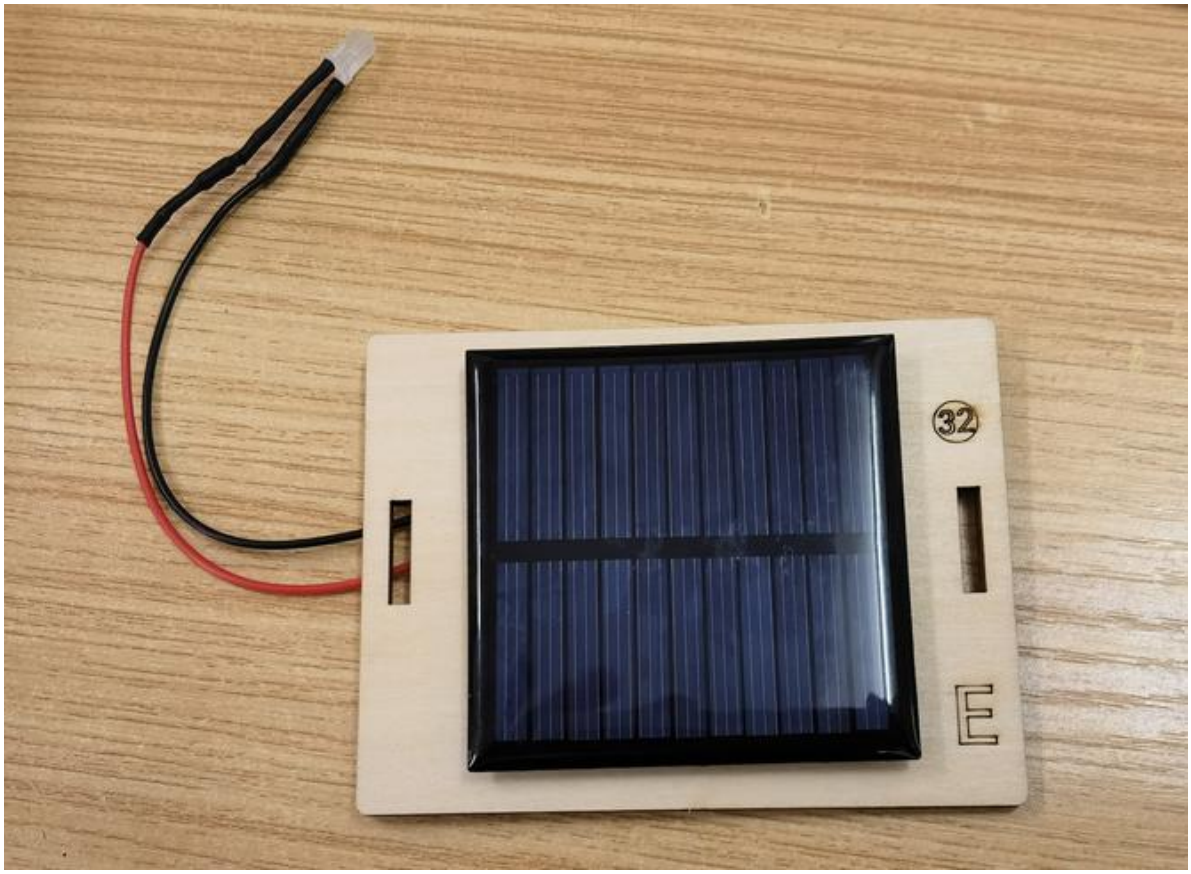


18. 3

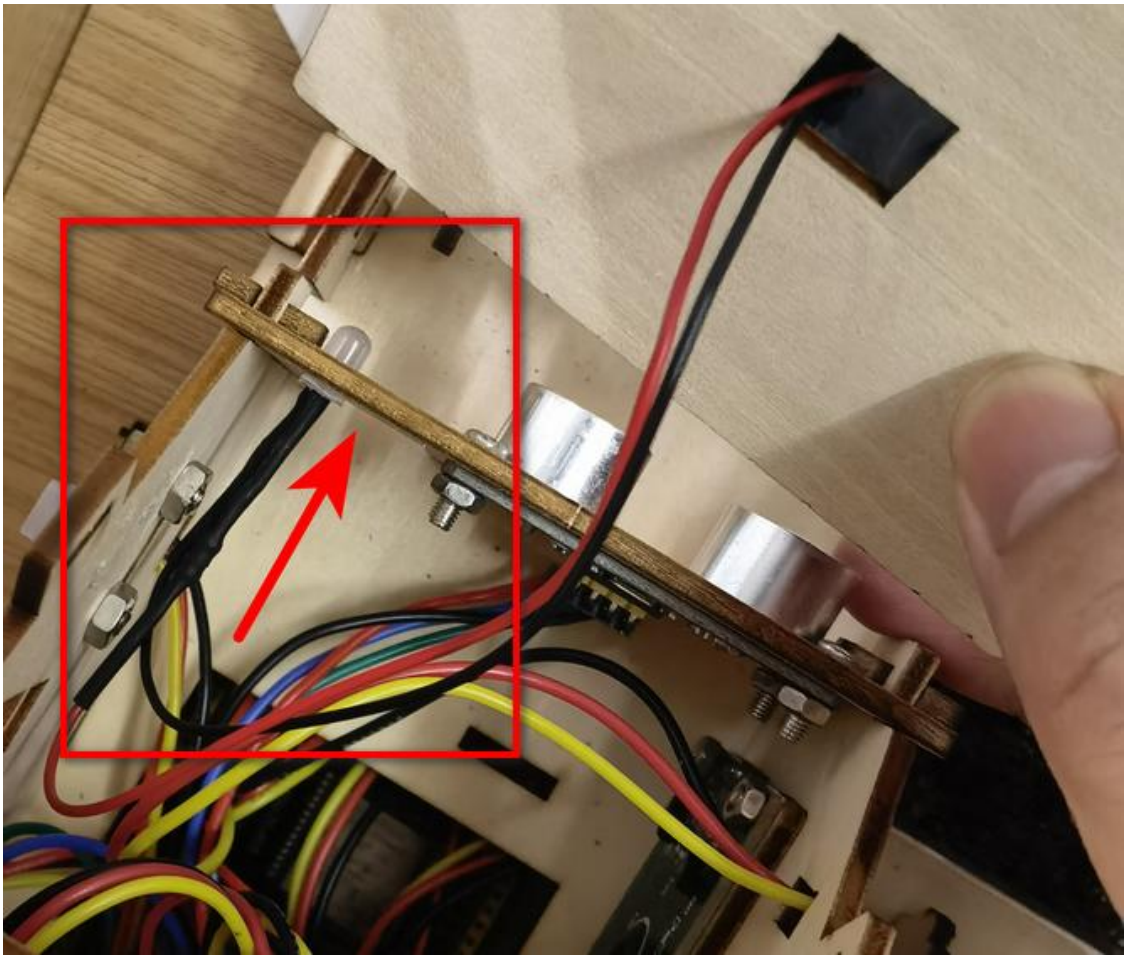


---

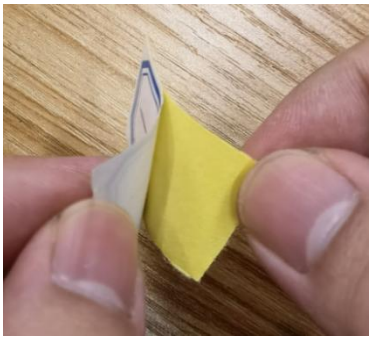
18. 4

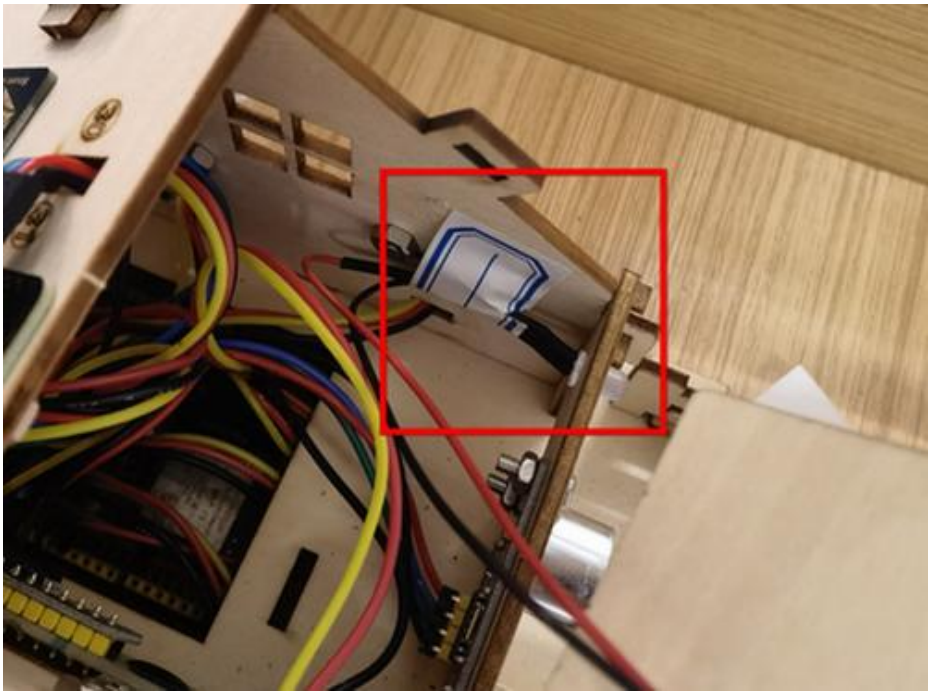


18.5 Install the LED light of the solar panel into this hole.

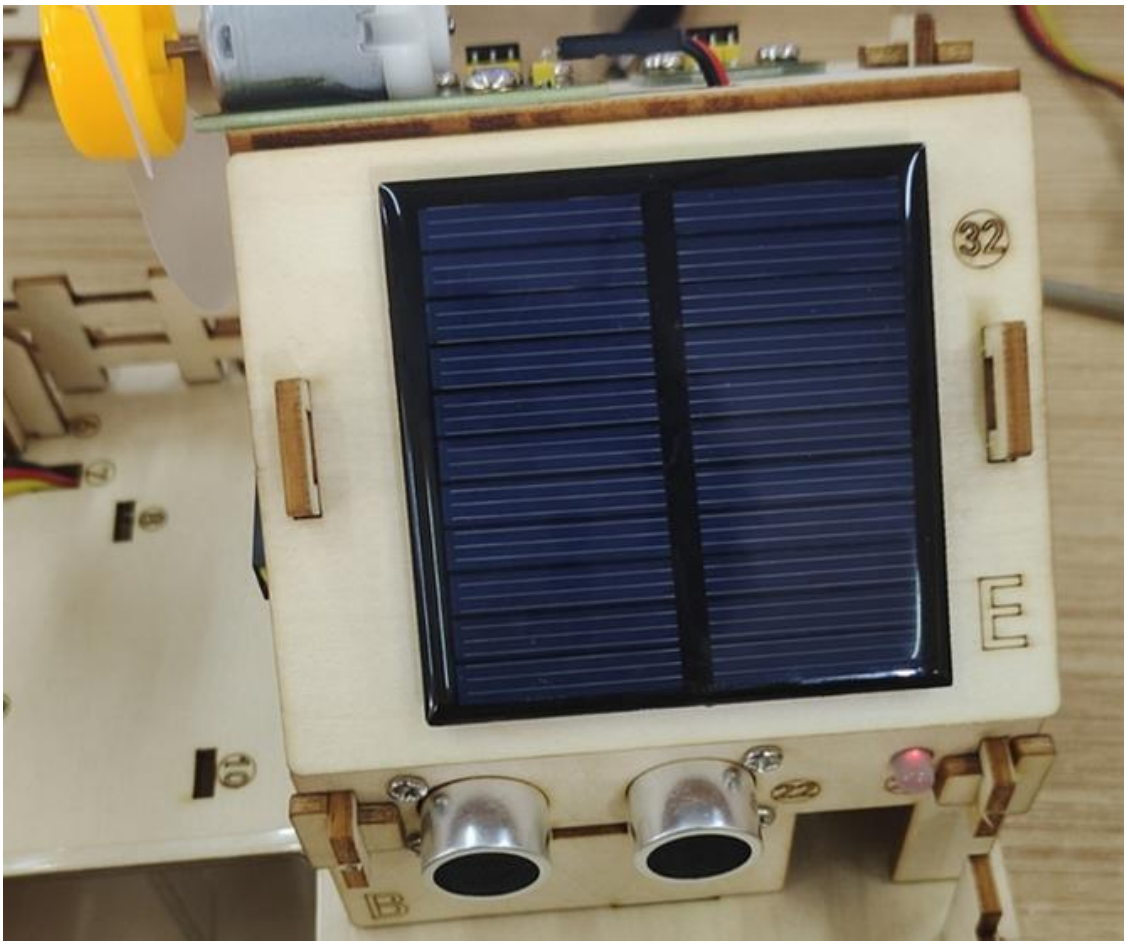


18.6 Use a sticker to secure its wires to the wall



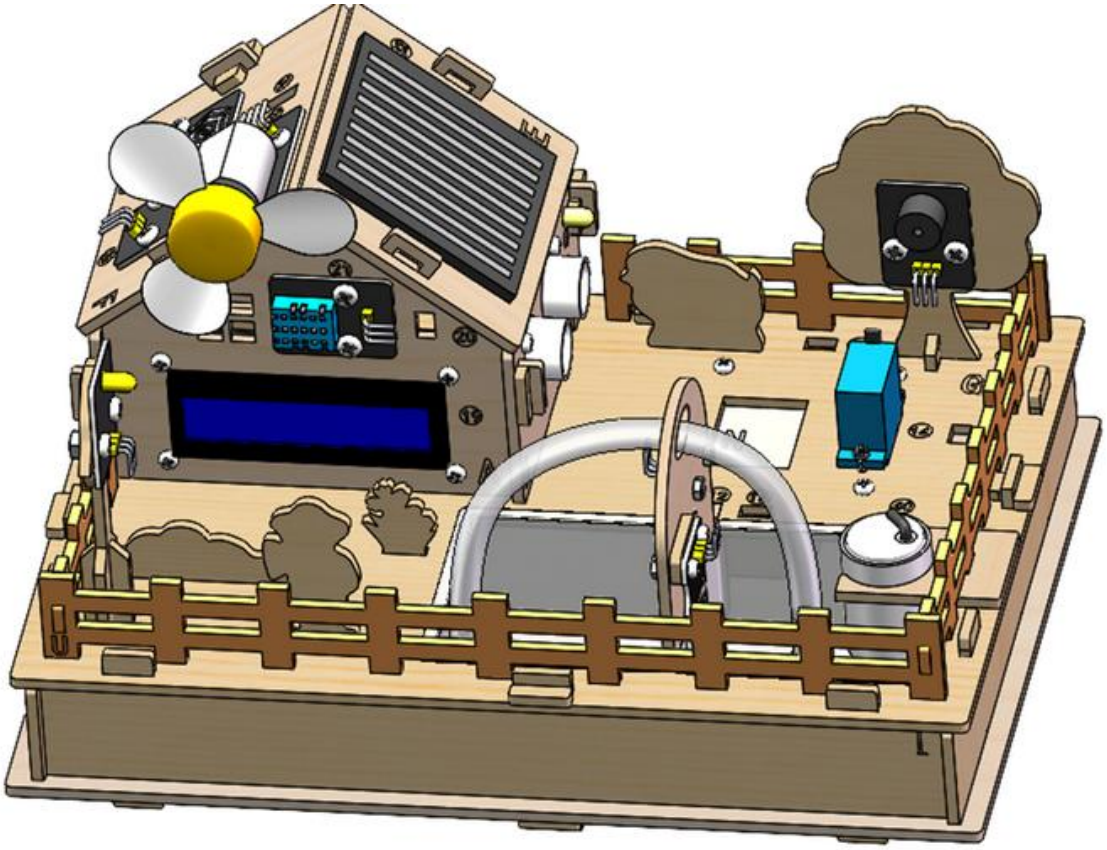


18. 7



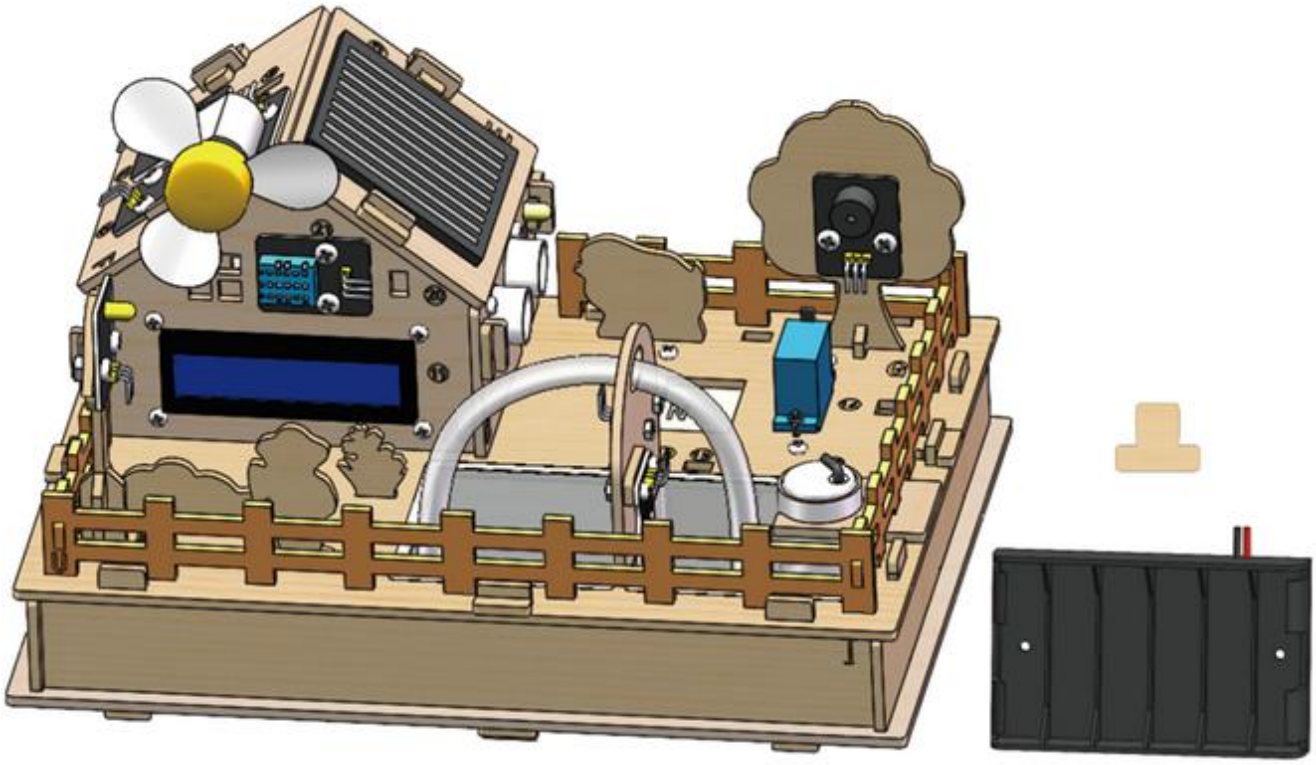
---

18. 8



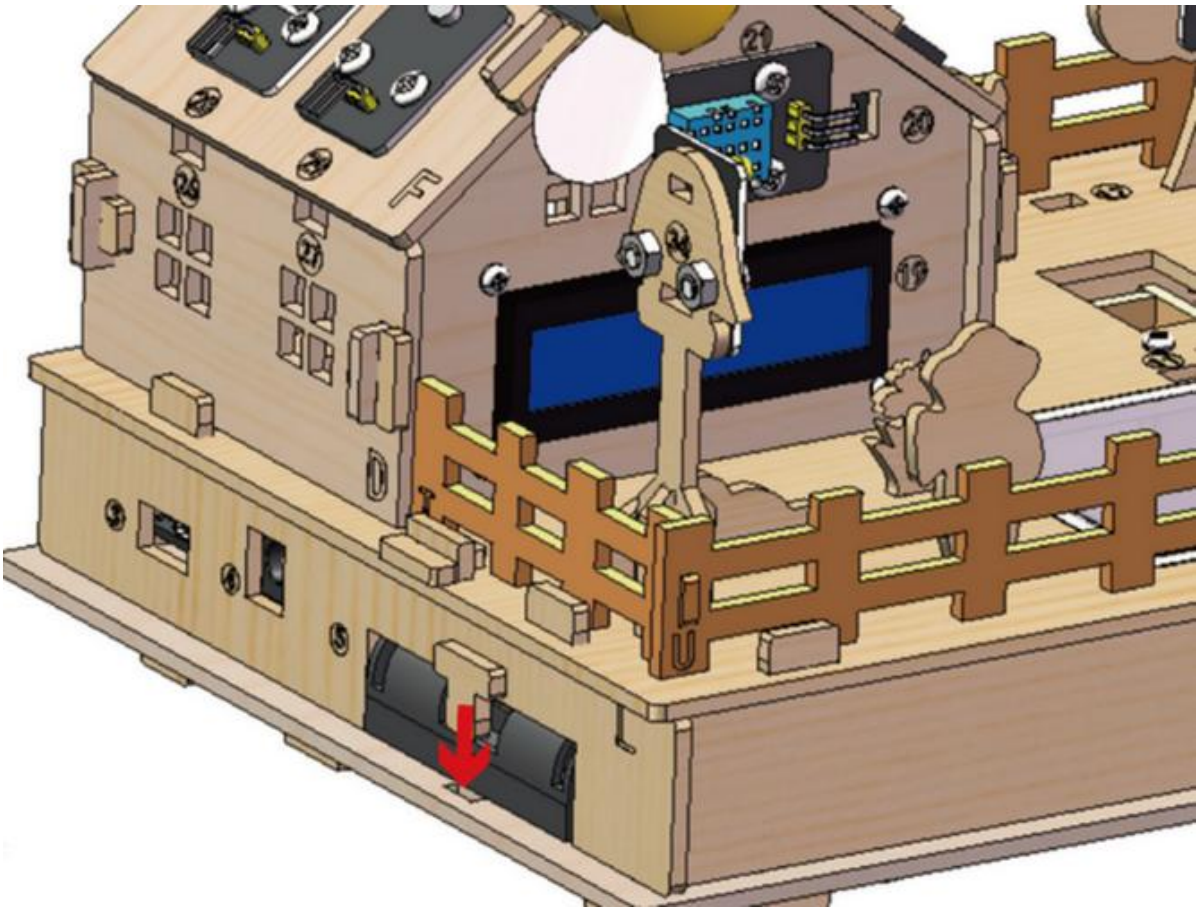
Step 19 Install Battery Case

19.1

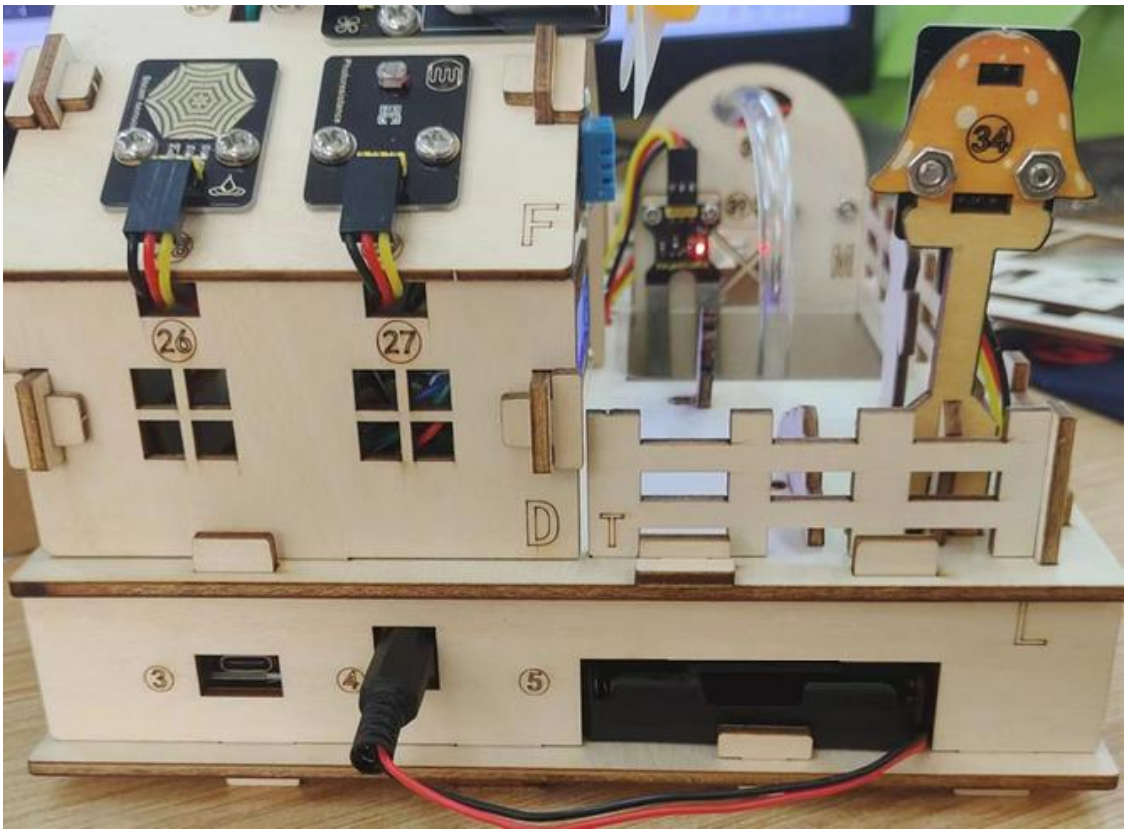


19.2 Install 6 AA batteries (Not included in the kit)

19. 3



19. 4



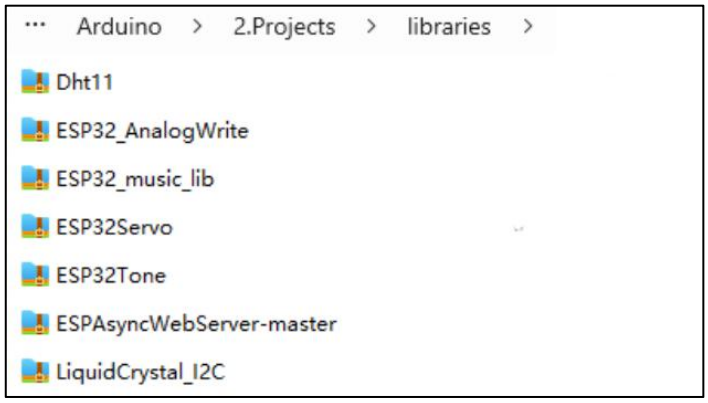
4.Projects

Things to note before starting the projects

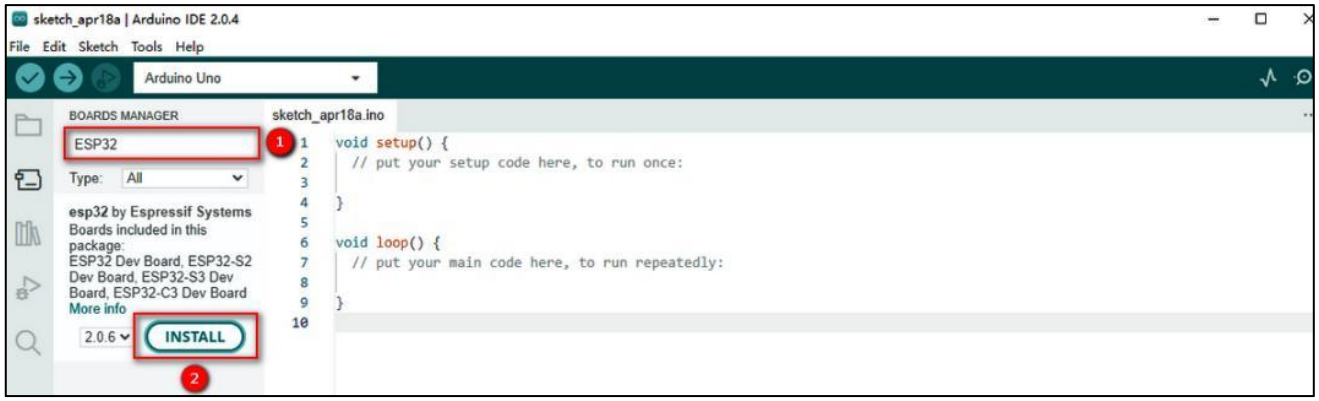
Please make sure you have configured the Arduino according to section 1.

1. Driver for KEYESTUDIO ESP32 PLUS Board

2. Add Libraries to Arduino IDE

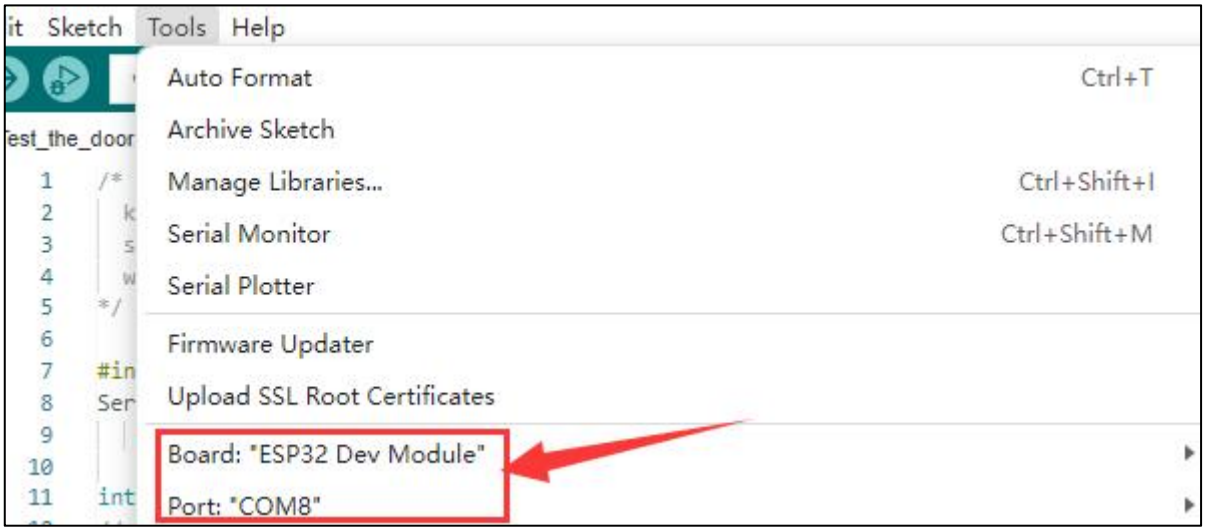


3. Install ESP32 environment in Arduino : **2.0.6** (ESP32 by Espressif Systems).



Note: You must select **version 2.0.6**, otherwise it may be incompatible with the library files we provide.

Click on **Tools**, select "**EPS32 Dev Module**" for the board type, and select **COM-XX** Port that the computer assigns to your Arduino device.



4. You can download all the files needed to run the robot arm, including the driver, codes, libraries, etc:
https://fs.keyestudio.com/KS0567

5. You need to prepare a 2.4 GHz WiFi(It can be a mobile hotspot or a router), another phone that can connect to the same WiFi and install the app we provided.

Arduino's memory can store a project at a time, updating the code erases the previous code.
Let's learn how this smart farm works step by step through projects of different difficulty levels.

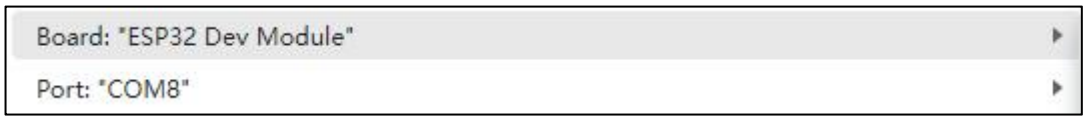
4.1 Lighting System

1.1 Light up an LED

Open the **1.1Blink** code with Arduino IDE.

```
#define LED_BUILTIN 27  //LED pins

void setup() {
  // initialize digital pin LED_BUILTIN as an output.
  pinMode(LED_BUILTIN, OUTPUT);
}

// the loop function runs over and over again forever
void loop() {
  digitalWrite(LED_BUILTIN, HIGH);  // turn the LED on (HIGH is the voltage level)
  delay(1000);                      // wait for a second
  digitalWrite(LED_BUILTIN, LOW);   // turn the LED off by making the voltage LOW
  delay(1000);                      // wait for a second
}
```

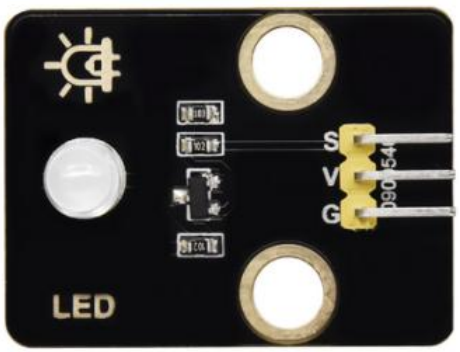
Choose the **ESP32 Dev Module** board and **COM** port, and upload the code.



Test Result:

LED blinks per second, because io27 on ESP32 board outputs high and low level alternatively every second.

Power Level	Result
HIGH	LED on
LOW	LED off

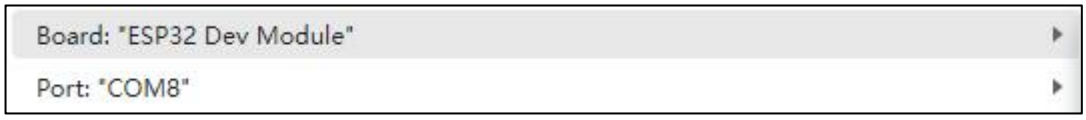


1.2 Control the LED with PWM

Open the **1.2PWM** code with Arduino IDE.

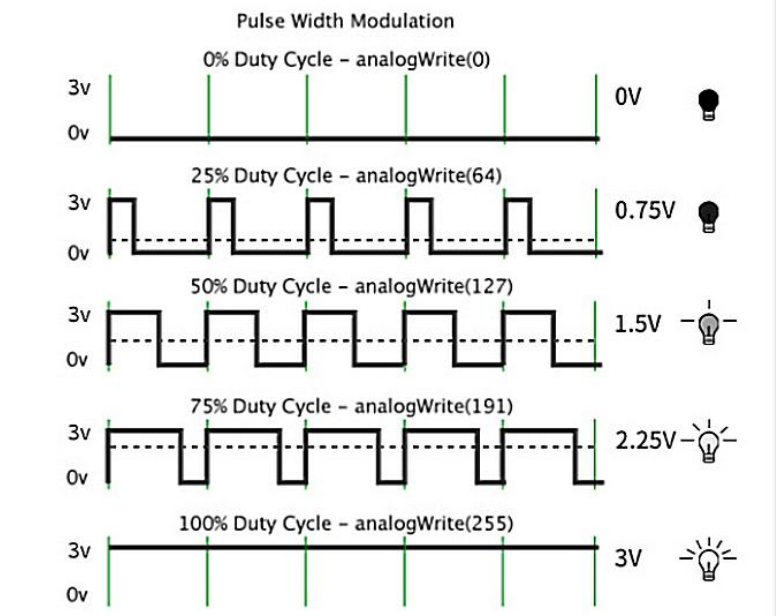
```
#include <analogWrite.h>  //Import PWM output library
#define led 27    //Define LED pin

void setup(){
  pinMode(led, OUTPUT);  //Set pin to output mode
}

void loop(){
  for(int i=0; i<255; i++)  //for loop statement. Constantly increase variable i till 255, exit the loo
  {
    analogWrite(led, i);  //PWM output, used to control the brightness of LED
    delay(3);
  }
  for(int i=255; i>0; i--)  //for loop statement. Constantly decrease variable i till 0, exit the loop
  {
    analogWrite(led, i);
    delay(3);
  }
}
```

Choose the **ESP32 Dev Module** board and **COM** port, and upload the code.



Test Result:

At an appropriate signal frequency, PWM changes effective output voltage by changing the duty cycle in one period. In plain English, within a specified time, the more high level the IO port outputs, the greater PWM value is, and the lighter LED will be.



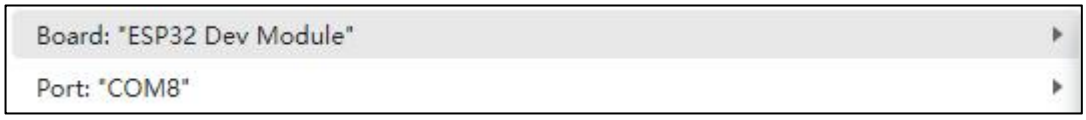
The LED module will slowly light up from dark to bright, and then from dark to bright.

1.3 Read the digital value of Button

Open the **1.3Button** code with Arduino IDE.

```
#define ButtonPin 5 //Define the button pin to 5

void setup() {
  //initialize serial port and set baud rate to 9600
  Serial.begin(9600);
  //Set pin to input mode
  pinMode(ButtonPin,INPUT);
}

void loop() {
  //Define a value as the read button value
  int ReadValue = digitalRead(ButtonPin);
  //Serial port prints the defined value
  Serial.print("The current status of the button is : ");
  Serial.println(ReadValue);
  delay(500);
}
```

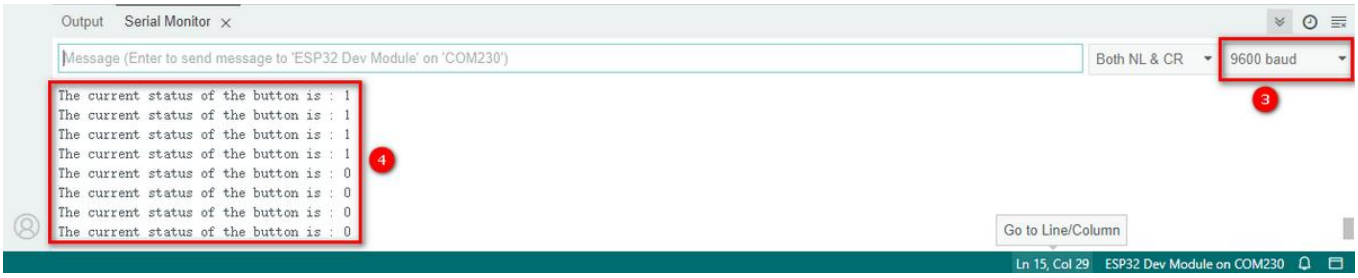
Choose the **ESP32 Dev Module** board and **COM** port, and upload the code.



Test Result:

Open serial monitor and set the baud rate to 9600.

When the button is released, the value is 1; if you press the button, it becomes 0.



The principle of the button module is a circuit controlled by this button.

When the button is pressed, the circuit is closed so that current passes through the button to GND, which causes the digital input pin to detect a low level.

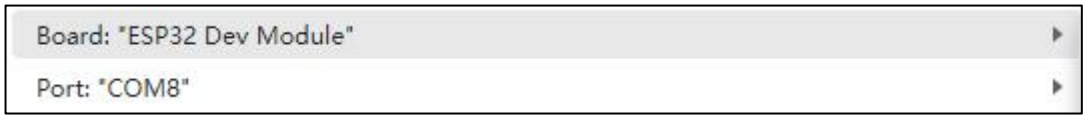
When the button is released, the circuit is cut and pin level increases due to a pull-up resistor, which makes the digital pin to detect a high level.

1.4 Auto-locking Button

Open the **1.4Self-Locking_Button** code with Arduino IDE.

```
#define ButtonPin 5 //Define the button pin
int value = 0;      //Define a value to determine the status of button

void setup() {
  //Initialize the serial port and set baud rate to 9600
  Serial.begin(9600);
  //Set the pin to inpu tmode
  pinMode(ButtonPin,INPUT);
}

void loop() {
  //Define a value as the read button value
  int ReadValue = digitalRead(ButtonPin);
  //Detect whether the button is pressed
  if (ReadValue == 0) {
    //Eliminate the button shake
    delay(10);
    if (ReadValue == 0) {
      value = !value;
      Serial.print("The current status of the button is : ");
      Serial.println(value);
    }
    //Detect again whether the button is still pressed
    //Pressed: execute the loop; Released: exit the loop to next step
    while (digitalRead(ButtonPin) == 0);
  }
}
```

Choose the **ESP32 Dev Module** board and **COM** port, and upload the code.



Test Result:

Open serial monitor and set the baud rate to 9600.

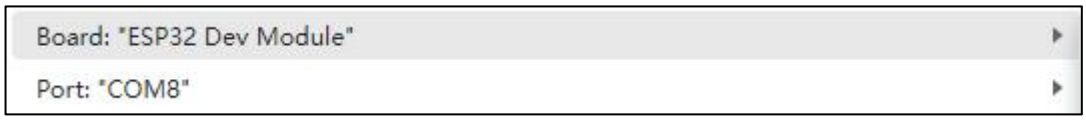
When you press the button once, 1 will be displayed. If you press button for the second time, the value becomes 0. Now, a common button boasts the function of an auto-locking one.

1.5 Use the button to control LED module

Open the **1.5Lighting-System** code with Arduino IDE.

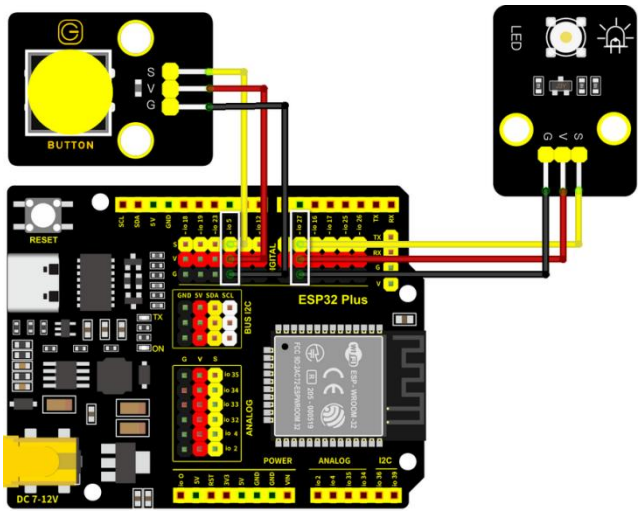
```
#define ButtonPin 5   //Define a button pin
#define LED       27  //Define LED pin
int value = 0;        //Define a value to detect button status

void setup() {
  //initialize serial port and set baud rate to 9600
  Serial.begin(9600);
  //Set pin to input mode
  pinMode(ButtonPin,INPUT);
  //Set pin to output mode
  pinMode(LED,OUTPUT);
}

void loop() {
  //Define a value as the read button value
  int ReadValue = digitalRead(ButtonPin);
  //Detect whether the button is pressed
  if (ReadValue == 0) {
    //Eliminate the button shake
    delay(10);
    if (ReadValue == 0) {
      value = !value;
      //Detect the button status, press once to light up LED, press again to turn off LED, in a loop
      if(value) {
```

Choose the **ESP32 Dev Module** board and **COM** port, and upload the code.



Test Result:

When you press the button once, LED lights up; if you press again, LED turns off. This operation is a loop, which is consistent with the lighting principle in reality.



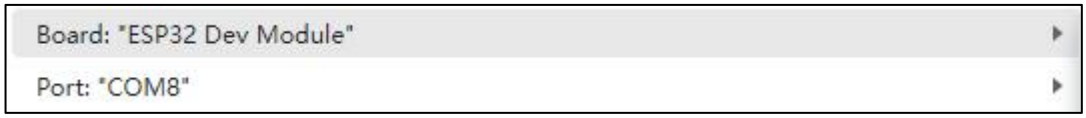
4.2 Light Control System

2.1 Photocell-sensor

Open the **2.1Photocell-sensor** code with Arduino IDE.

```
#define PhotocellPin 34  //Define the photoresistor pin

void setup() {
  //Initialize the serial port
  Serial.begin(9600);
  //Set the pin to input mode
  pinMode(PhotocellPin,INPUT);
}

void loop() {
  //Read the value of photoresistor
  int ReadValue = analogRead(PhotocellPin);
  //Print the value. NOTE: ESP32 board is 12-bit ADC, whose detection value range is within 0~4095.
  Serial.print("Photocell value: ");
  Serial.println(ReadValue);
  delay(500);
}
```

Choose the **ESP32 Dev Module** board and **COM** port, and upload the code.

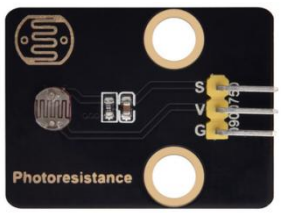


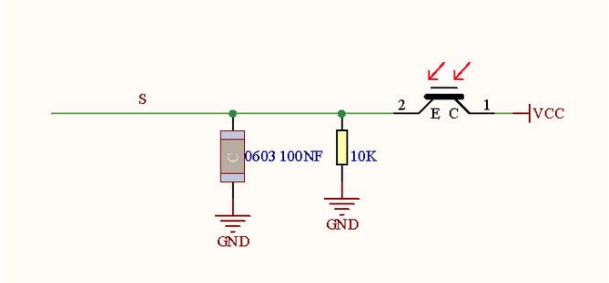
Test Result:

Open the serial monitor.

The brighter the light detected by the photoresistor is, the greater the value will be.

A photoresistor module converts light signal into electric signal (voltage, current, and resistor). When light hits the photoresistor, the stronger the light is, the smaller the resistance will be, so the greater the VCC voltage will pass through the photoresistor.



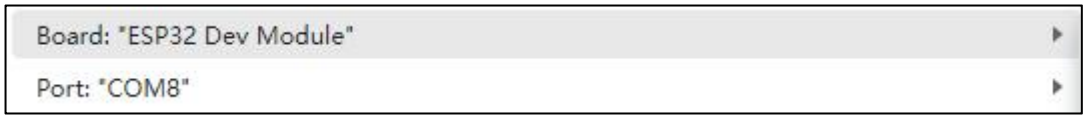


2.2 Light Control System

Open the **2.2Light-Control-System** code with Arduino IDE.

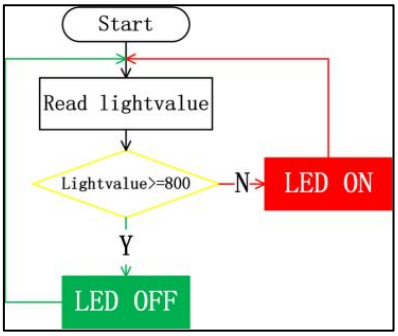
```
#define PhotocellPin 34  //Define the photoresistor pin
#define LED          27  //Define LED pin

void setup() {
  //Initialize serial port
  Serial.begin(9600);
  //Set the photoresistor pin to input mode
  pinMode(PhotocellPin,INPUT);
  //Set the LED pin to output mode
  pinMode(LED,OUTPUT);
}

void loop() {
  //Read the value of the photoresistor
  int ReadValue = analogRead(PhotocellPin);
```

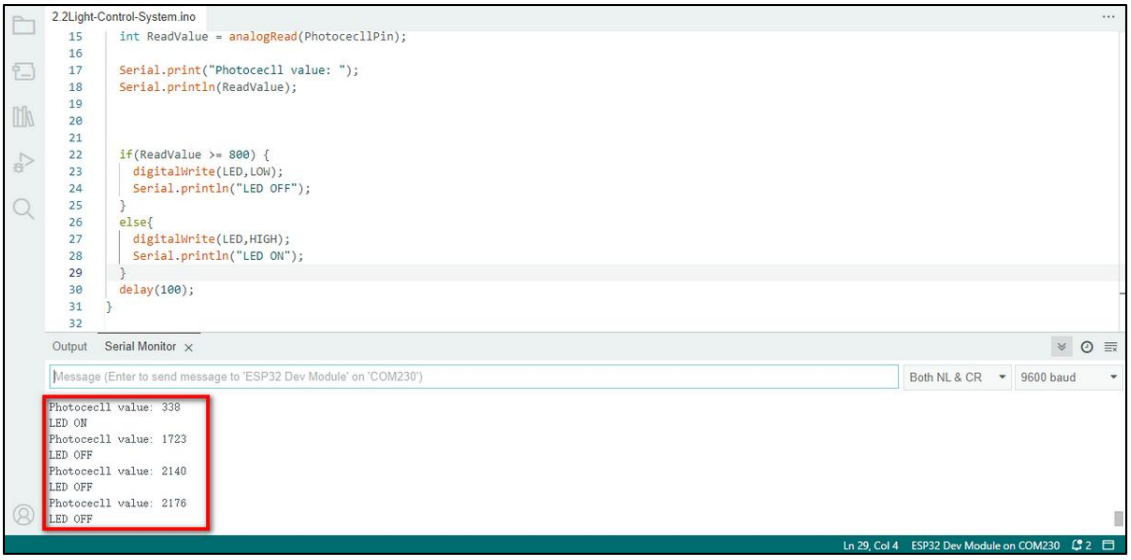
Choose the **ESP32 Dev Module** board and **COM** port, and upload the code.



Test Result:

When the value of the photoresistor is greater than 800 (in daytime), LED goes off. However, if the value is less than 800, LED will automatically light on.





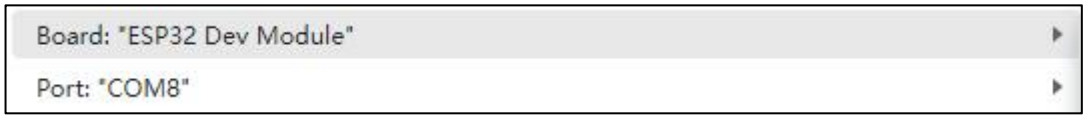
4.3 Alarm System

3.1 PIR Motion Sensor

Open the **3.1PIR-Motion-Sensor** code with Arduino IDE.

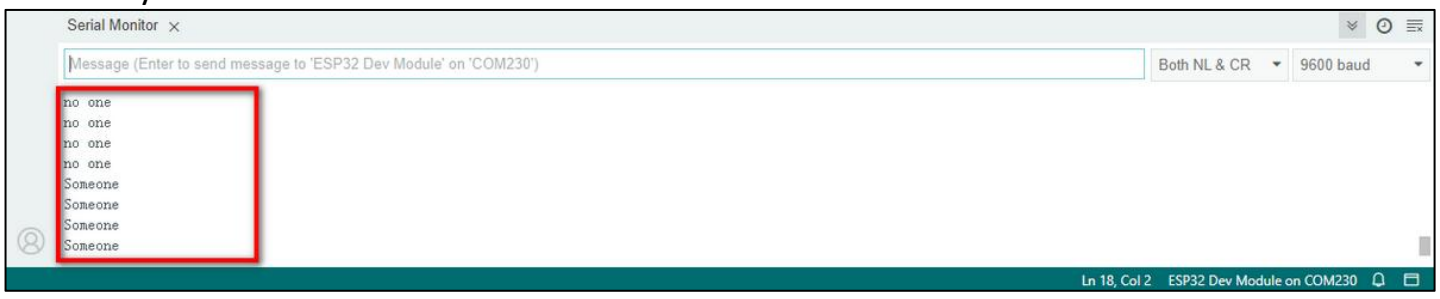
```
#define PyroelectricPIN 23

void setup() {
  Serial.begin(9600);
  pinMode(PyroelectricPIN,INPUT);
}

void loop() {
  //Read the value of PIR motion sensor
  int ReadValue = digitalRead(PyroelectricPIN);
  if(ReadValue){
    Serial.println("Someone");
  }
  else{
    Serial.println("No one");
  }
  delay(100);
}
```

Choose the **ESP32 Dev Module** board and **COM** port, and upload the code.



Test Result:

When someone is in the area, **Someone** is displayed on the monitor, and the red LED on the sensor goes off. However, if there is no one, **No one** will be printed and the LED on the sensor will always be on.



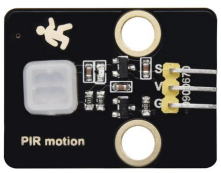
Voltage: 3~5V

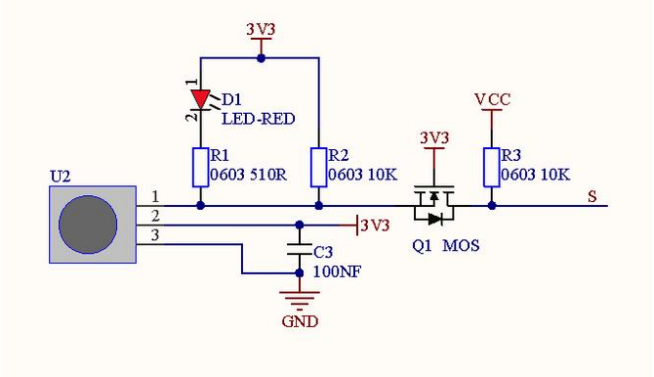
Current: 3.6mA

Power: 18mW

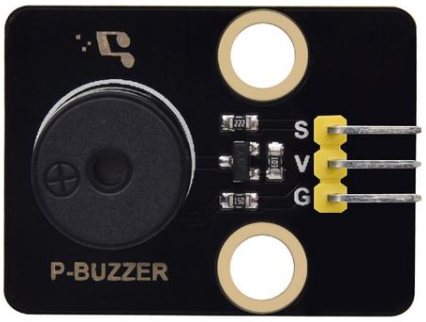
Angle of View: Y = 90°, X = 110° (Theoretical value)

Detection Distance: ≤5m

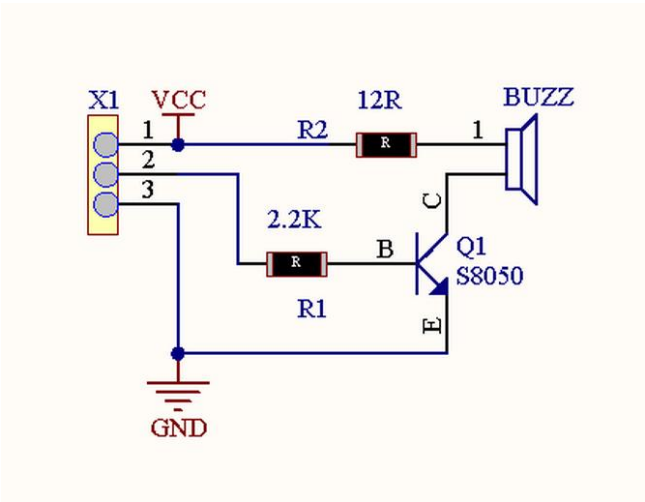




3.2 Passive Buzzer



- **Passive Buzzer** cannot vibrate to emit sound itself, unless putting a square wave signal with a certain frequency. Moreover, the emitting sound varies due to the different frequency of square wave, so a passive buzzer can simulate tunes.
  - An analog squire wave can be generated by changing the power level at pins. For example, after the high level lasting for 500ms, it shifts to a low level for another 500ms then to a high level again...
  - We drive the buzzer via a squire wave within 200~5000Hz, and we can compute the frequency(f): **f=1/T**, T is the period (the total time of high and low level).



Parameters:

Voltage: 3~5V

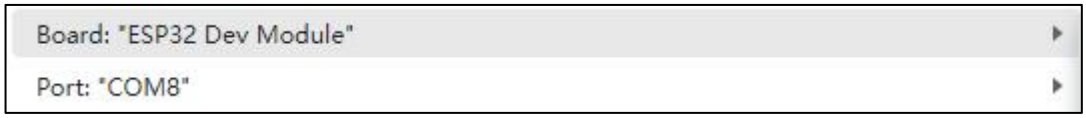
Current: ≤5mA

Power: ≤25mW

Open the **3.2Passive-Buzzer** code with Arduino IDE.

```
#define BuzzerPin 16  //Define the buzzer pin

void setup() {
  //Set the pin to output mode
  pinMode(BuzzerPin,OUTPUT);
}

void loop() {
  digitalWrite(BuzzerPin,HIGH);
  delayMicroseconds(500);//Delay 500us
  digitalWrite(BuzzerPin,LOW);
  delayMicroseconds(500);//Delay 500us
}
```

Choose the **ESP32 Dev Module** board and **COM** port, and upload the code.



Test Result:

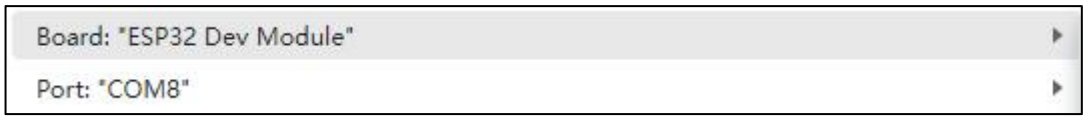
Passive Buzzer keeps emitting sound.

3.3 Buzzer-Tone

Open the **3.3Buzzer-Tone** code with Arduino IDE.

```
const int buzzerPin = 16;   //Set buzzer pin to 16
void setup() {
  pinMode(buzzerPin,OUTPUT);
}
void loop() {
//tone(buzzerPin,294,250,0);  //4 parameters are: pin, frequency, delay, path
    tone(buzzerPin,532);           //duo --C2
    delay(100);
    tone(buzzerPin,587);           //re --D3
    delay(100);
    tone(buzzerPin,659);           //mi --E3
    delay(100);
   //Alarm
   for(int i = 200; i<=1000; i+=10){
    tone(buzzerPin,i);
    delay(10);
   }
    //Alarm
   for(int i = 1000; i>=200; i-=10){
    tone(buzzerPin,i);
    delay(10);
```

Choose the **ESP32 Dev Module** board and **COM** port, and upload the code.



Test Result:

Buzzer alarms via tone() function.

tone() generates PWM signal with a certain frequency to drive the buzzer to vibrate, and the duration and tone is controlled by related parameters.

There are two ways to define the duration. One is to adjust the parameters of the tone() function to set a duration, and the other is to adopt a noTone() function to directly stop the sound. If you do not define a duration in tone(), the sound signal will always be generated unless a noTone() stops it.

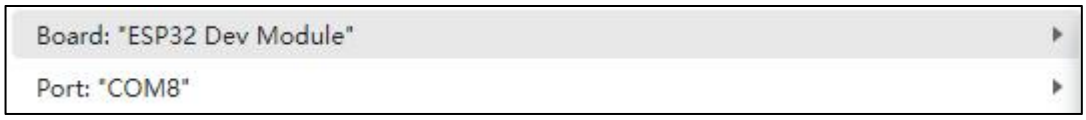
In Arduino, one sound can only be produced at a time. If one pin of ESP32 is generating a sound signal via tone(), it is not acceptable to emit sound by this function on another pin.

3.4 Buzzer-Music

Open the **3.4Buzzer-Music** code with Arduino IDE.

```
#define NOTE_B0  31
#define NOTE_C1  33
#define NOTE_CS1 35
#define NOTE_D1  37
#define NOTE_DS1 39
#define NOTE_E1  41
#define NOTE_F1  44
#define NOTE_FS1 46
#define NOTE_G1  49
#define NOTE_GS1 52
#define NOTE_A1  55
#define NOTE_AS1 58
#define NOTE_B1  62
#define NOTE_C2  65
#define NOTE_CS2 69
#define NOTE_D2  73
#define NOTE_DS2 78
#define NOTE_E2  82
#define NOTE_F2  87
#define NOTE_FS2 93
#define NOTE_G2  98
#define NOTE_GS2 104
#define NOTE_A2  110
#define NOTE_AS2 117
#define NOTE_B2  123
#define NOTE_C3  131
#define NOTE_CS3 139
#define NOTE_D3  147
#define NOTE_DS3 156
#define NOTE_E3  165
#define NOTE_F3  175
#define NOTE_FS3 185
#define NOTE_G3  196
#define NOTE_GS3 208
#define NOTE_A3  220
#define NOTE_AS3 233
#define NOTE_B3  247
#define NOTE_C4  262
#define NOTE_CS4 277
#define NOTE_D4  294
```

Choose the **ESP32 Dev Module** board and **COM** port, and upload the code.



Test Result:

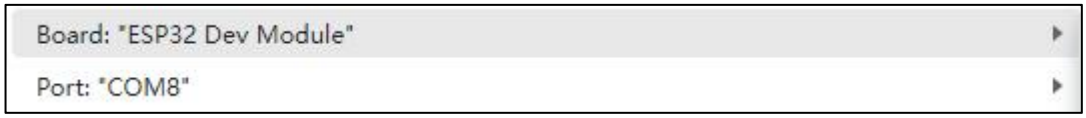
The buzzer will play music.

3.5 Alarm System

Open the **3.5Alarm-System** code with Arduino IDE.

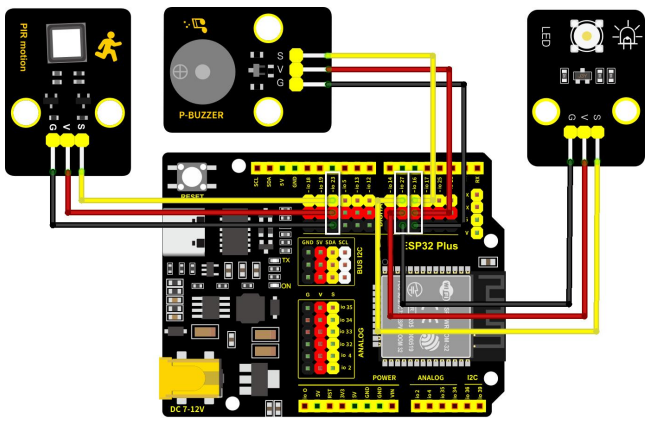
```
#define BuzzerPin 16        //Set buzzer pin to 16
#define PyroelectricPIN 23  //Set PIR mition sensor to 23
#define Led 27              //Set Led pin to 27

void setup() {
  Serial.begin(9600);
  //Set the pins' modes
  pinMode(BuzzerPin,OUTPUT);
  pinMode(PyroelectricPIN,INPUT);
  pinMode(Led,OUTPUT);
}
void loop() {
  //Read the value of PIR motion sensor
  int ReadValue = digitalRead(PyroelectricPIN);
  if(ReadValue){
    Serial.println("Someone");
    digitalWrite(Led,HIGH);
    //Alarm
    for(int i = 200; i<=1000; i+=10){
      tone(BuzzerPin,i);
      delay(10);
    }
    digitalWrite(Led,LOW);
    //Alarm
    for(int i = 1000; i>=200; i-=10){
      tone(BuzzerPin,i);
      delay(10);
    }
  }
  //Stop alarming
  noTone(BuzzerPin);
  Serial.println("No one");
}
```

Choose the **ESP32 Dev Module** board and **COM** port, and upload the code.



Test Result:

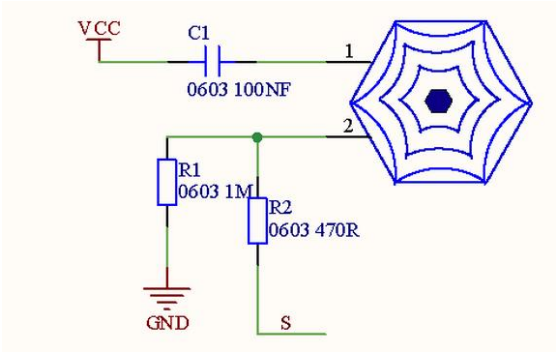
When the sensor detects a motion, buzzer emits sound and LED blinks to remind of an invasion.



4.4 Rain Detection System

4.1 Steam Sensor

Steam sensor detects the presence of water, so it is usually used in rain detection. If the rain hits the conductive pads on the sensor, it will send a signal to the Arduino board.



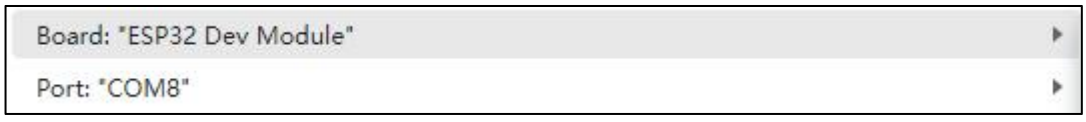
Parameters:

- Voltage: 3~5V
- Current: 1.5mA
- Power: 7.5mW

Open the **3.5Alarm-System** code with Arduino IDE.

```
#define SteamPin 35   //Define the steam sensor pin to 35

void setup() {
  Serial.begin(9600);
  pinMode(SteamPin,INPUT);
}

void loop() {
  //Read the value of steam sensor
  int ReadValue = analogRead(SteamPin);
  Serial.print("Steam Value: ");
  Serial.println(ReadValue);
  delay(500);
}
```

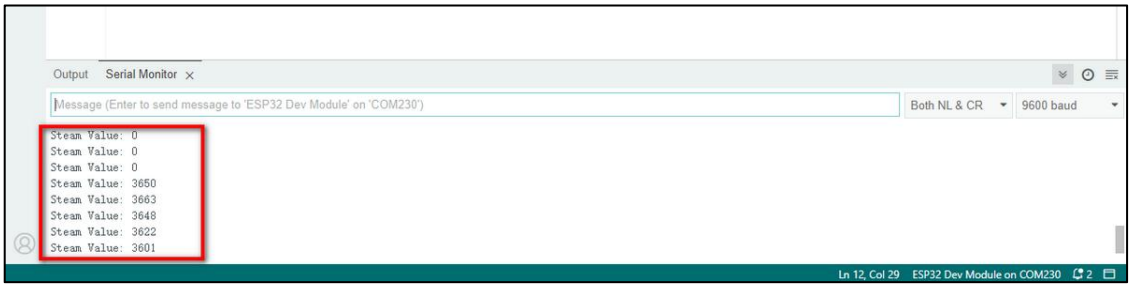
Choose the **ESP32 Dev Module** board and **COM** port, and upload the code.



Test Result:

Touch the detection area with your finger. The larger the area you touched is, the larger the value will be.

You can open the serial monitor to observe the currently detected value (range: 0~4095).

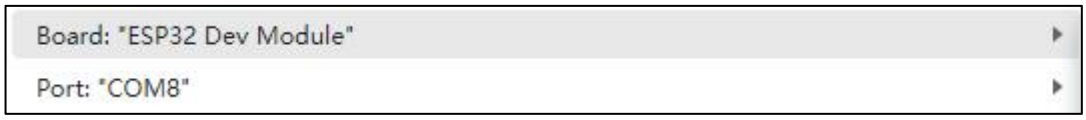


4.2 Rainwater Detection System

Open the **4.2Rainwater-Detection-System** code with Arduino IDE.

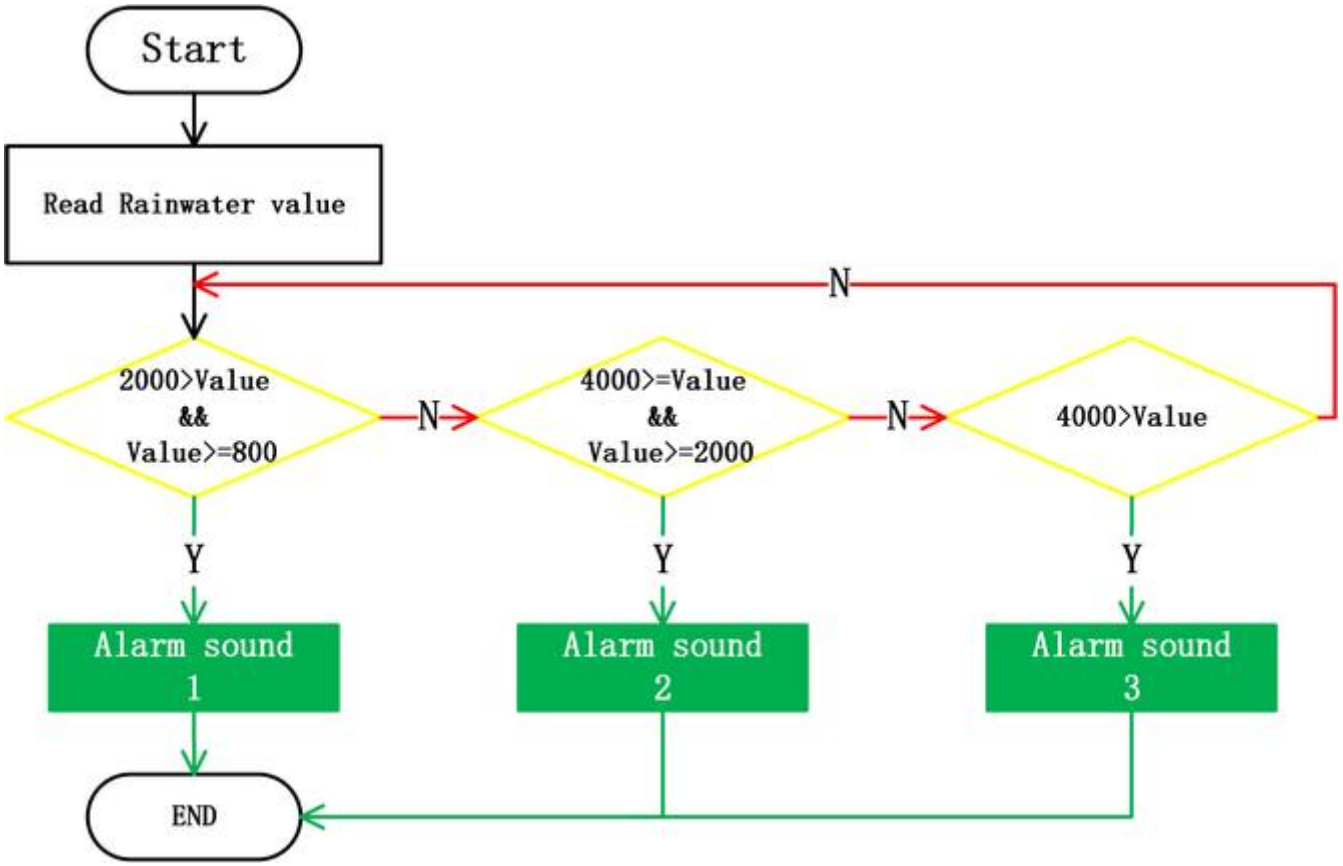
```
#define SteamPin 35   //Define pins
#define BuzzerPin 16

void setup() {
  Serial.begin(9600);
  pinMode(SteamPin,INPUT);
  pinMode(BuzzerPin,OUTPUT);
}

void loop() {
  //Read the value of steam sensor
  int ReadValue = analogRead(SteamPin);
  Serial.print("Steam Value: ");
  Serial.println(ReadValue);
  //Determine whether the detected value is within 800~2000
```

Choose the **ESP32 Dev Module** board and **COM** port, and upload the code.



Test Result:

The greater the detected value of the steam sensor is, the loader the sound emitted by the buzzer will be.



4.5 Solar Power System

Parameters

- Voltage: 5V
- Current: 80mA
- Power: 400mW
- Dimensions: 60*60mm

Codes are not required in this project. Importantly, we learn about the new environmental energy --- solar power.

When good illumination is provided, LED will light up in yellow. The brighter the light is, the brighter the LED will be.

The solar panel absorbs light and directly or indirectly converts solar radiation into electricity. Compared with ordinary coal power generation, solar, wind and water power are more energy-saving and environment-friendly.

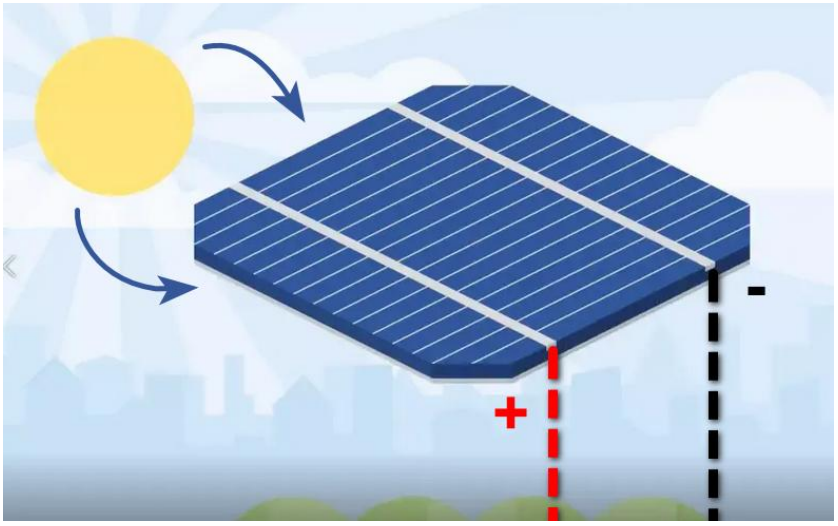
The Sun emits energy in waves with a wide range of wavelengths, from ultraviolet to visible and infrared light.

Wavelength of Ultraviolet: 150~400nm;

Wavelength of Visible Light: 400~760nm;

Wavelength of Infrared Light: 760~4000nm;

The panel absorbs one of these ranges of wavelength and converts them into electricity.



FAQ

Q: Why does solar panel still work without sunlight?

A: It works with not only sunlight but also ambient light. The brighter the light is, the greater the voltage will be, and the lighter the LED will be.

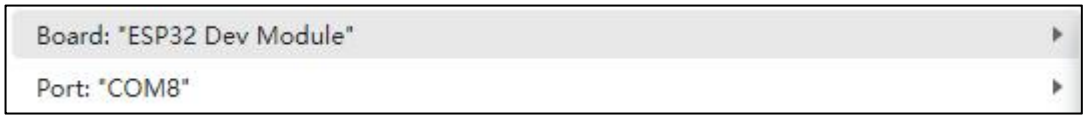
4.6 Smart Feeding System

6.1 Door of feeding cabin

Open the **6.1Servo** code with Arduino IDE.

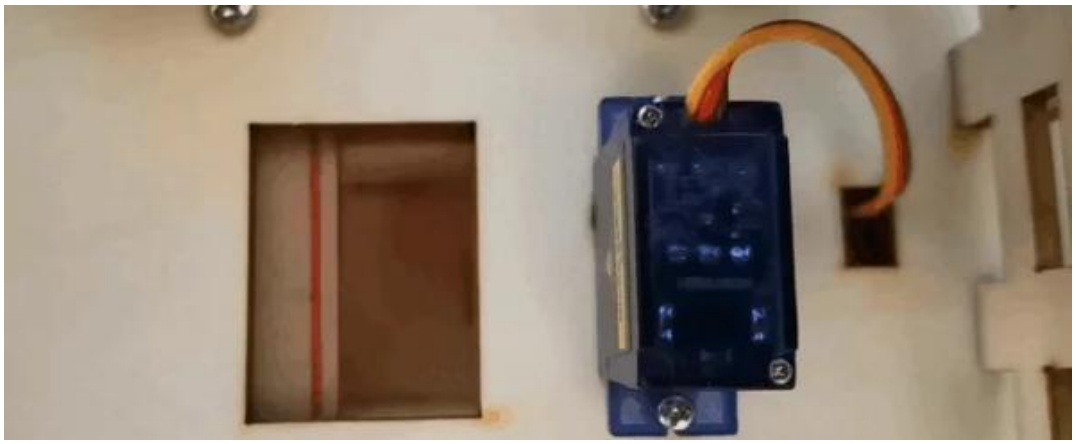
```
#include <ESP32_Servo.h>  //Import the library of servo
Servo myservo;  // create servo object to control a servo
                // 16 servo objects can be created on the ESP32

int pos = 0; // variable to store the servo position
// Recommended PWM GPIO pins on the ESP32 include 2,4,12-19,21-23,25-27,32-33
int servoPin = 26;

void setup() {
  Serial.begin(9600);
  myservo.attach(servoPin);   // attaches the servo on pin 26 to the servo object
  myservo.write(180);
  delay(2000);
}

void loop() {

  for (pos = 80; pos <= 179; pos += 1) { // goes from 0 to 80 degrees
    // in steps of 1 degree
```

Choose the **ESP32 Dev Module** board and **COM** port, and upload the code.



Test Result:

The door of feeding cabin is slowly opened and then closed.



NOTE: SG90 servo can rotate 180° . As the feeding box is small, 100° of rotation is enough to completely close the box.

80° : fully open

120° : half open

180° : close

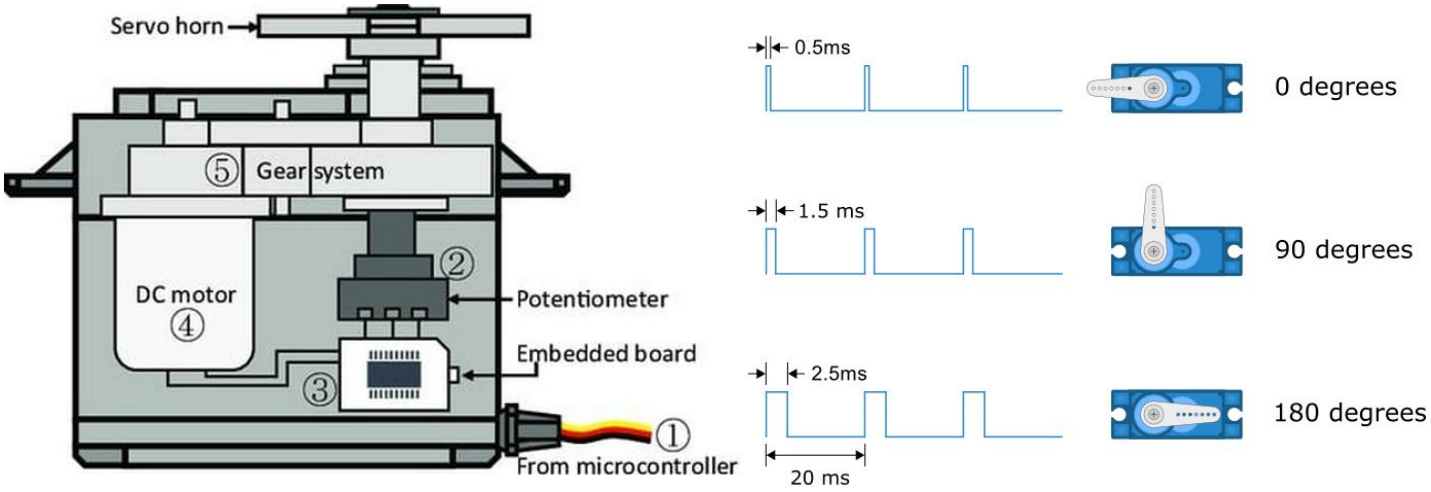
ATTENTION

Do not put your fingers into the box to avoid nipping!

Do not block the door with something to avoid damaging servo!

The door is controlled by a servo.

Internal Structure:



① Signal(S): It receives the control signal from microcontroller.

② Potentiometer: the feedback part of the Servo. It measures the position of output shaft.

③ Embedded board (Internal controller): the core of the Servo. It processes external control signal and the feedback signal of position and drives the Servo.

④ DC motor: the execution part. It outputs speed, torque and position.

⑤ Gear system: It scales the outputs from motor to the final output Angle ccording to a certain transmission ratio.

Drive the Servo:

Signal(S) receives PWM to control the output of Servo, and **the position of output shaft directly relies on the duty cycle of PWM**.

For instance:

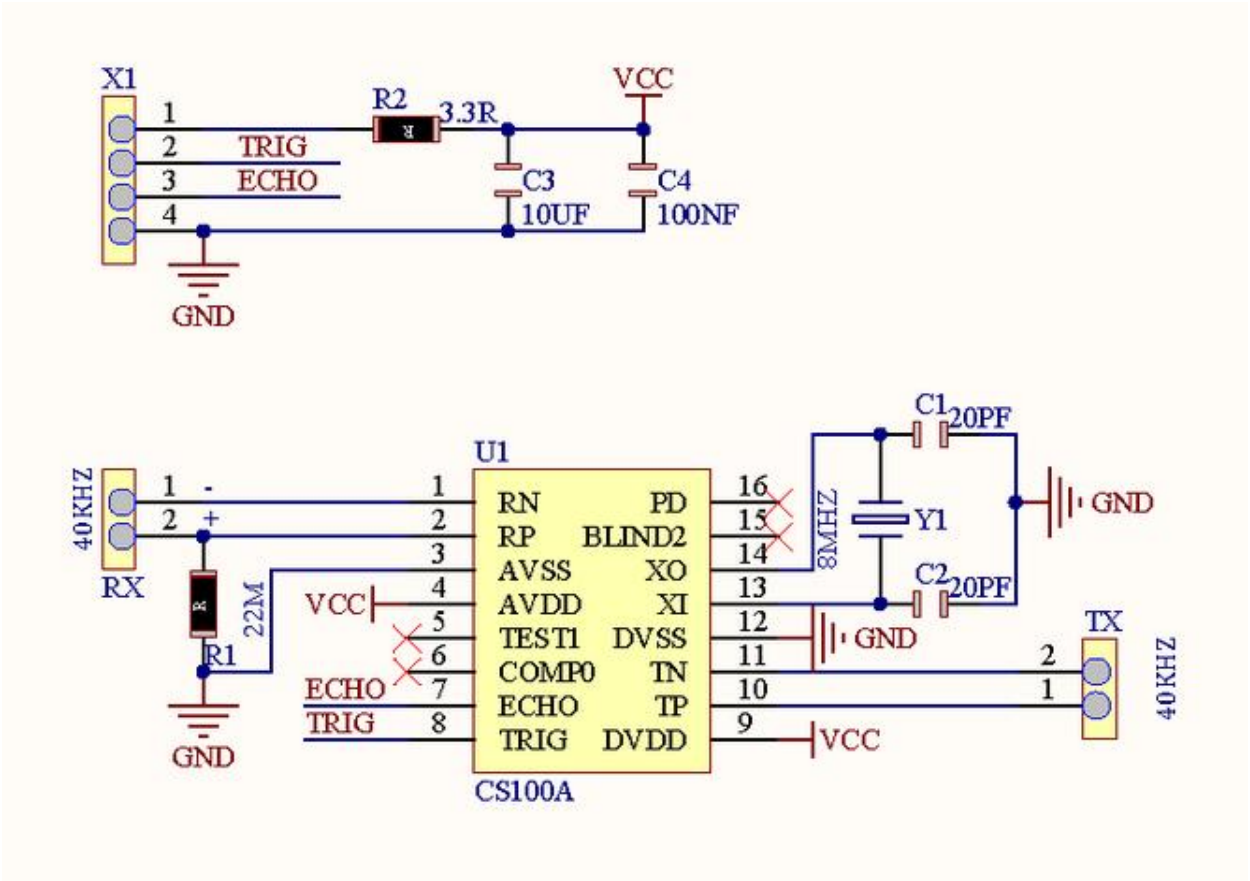
- If we send a signal with pulse width of 1.5ms to Servo, its shaft(horn) will revolves to the middle position(90°);
- If pulse width = 0.5ms, the shaft turns to its minimum(0°);
- If pulse width = 2.5ms, the shaft turns to its maximum(180°).

NOTE: The maximum angle varies from the types of Servos. Some are 170° while some are only 90°. In spite of this, Servos usually will move a half (of the maximum) if they receive a signal with pulse width of 1.5ms.

6.2 Ultrasonic-Sensor

Specification:

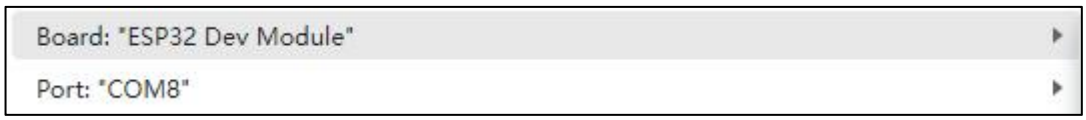
Working Voltage: DC 3.3V-5V
Working Current: 50mA - 100mA
Working Frequency: 40KHz
Max Power: 0.5WMax Range: 3m
Min Range: less than 4cm
Measuring Angle: less than 15 degree
Trigger Input Signal: 10μS TTL pulse
Interface: 4 pin port with 2.54mm space
Position Holes: the diameter is 3mm
Working Temperature: -10℃ - +60℃
Size: 49mm*22mm*19mm
Environmental attributes: ROHS



Open the **6.2Ultrasonic-Sensor** code with Arduino IDE.

```
#define Trigpin 12 //connect trig to io12
#define Echopin 13 //connect echo to io13
int duration,distance;

void setup(){
  Serial.begin(9600); //Set the baud rate to 9600
  pinMode(Trigpin,OUTPUT);  //set trig pin to output mode
  pinMode(Echopin,INPUT);   //set echo pin to input mode
}
void loop(){
  digitalWrite(Trigpin,LOW);
  delayMicroseconds(2);
  digitalWrite(Trigpin,HIGH);
  delayMicroseconds(10);        //Trigger the trig pin via a high level lasting at least 10us
  digitalWrite(Trigpin,LOW);
  duration = pulseIn(Echopin,HIGH);     //the time of high level at echo pin
  distance = duration/58;               //convert into distance(cm)
  delay(50);
  Serial.print("distance:");    //Serial monitor prints the value
  Serial.print(distance);
  Serial.println("cm");
}
```

Choose the **ESP32 Dev Module** board and **COM** port, and upload the code.



Test Result:

In this kit, the detection range is within 3~8cm.

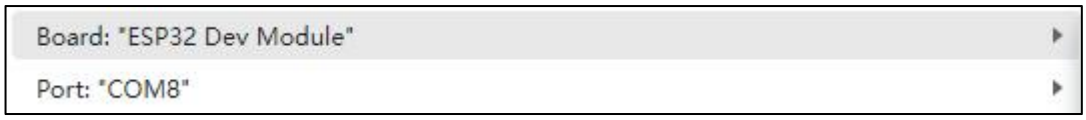
Open the serial monitor and set the baud rate to 9600, the serial monitor will display the distance between the ultrasonic module and the obstacle in front.

6.3 Intelligent Feeding System

Open the **6.3Intelligent-Feeding-System** code with Arduino IDE.

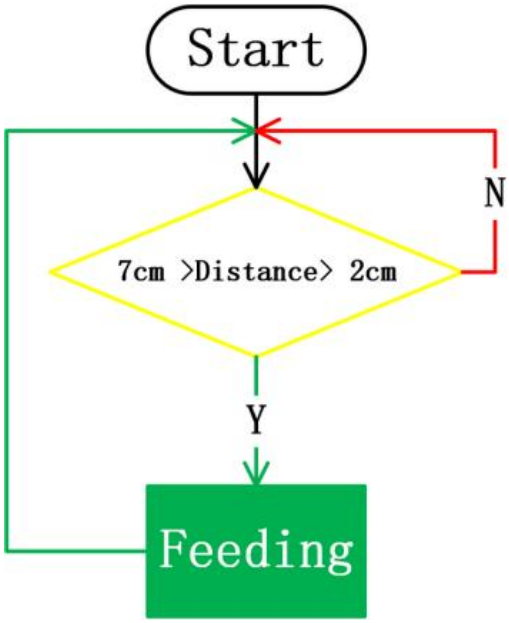
```
#include <ESP32_Servo.h>  //Import the library of servo on ESP32 board
Servo myservo;  // create an object to control servo
                // 16 servo objects can be created in total on the ESP32

#define TrigPin 12 //connect trig to D12
#define EchoPin 13 //connect echo to D13
#define ServoPin 26
int duration,distance;

void setup(){

  Serial.begin(9600); //Set the baud rate to 9600
  pinMode(TrigPin,OUTPUT);  //set trig pin to output mode
  pinMode(EchoPin,INPUT);   //Set echo pin to input mode
  myservo.attach(ServoPin);   // attaches the servo on pin 26 to the servo object
}
void loop(){
  Serial.println(getDistance());
  //When the distance is detected within 2~7cm, open the feeding box. Or else, close.
  if (getDistance() >= 2 && 7 >= getDistance()) {
```

Choose the **ESP32 Dev Module** board and **COM** port, and upload the code.



Test Result:

The smart feeding system intelligently feeds domestic fowls via an ultrasonic module and a servo. The former detects the distance to animals while the later controls to open or close the feeding box. When a pet is detected close to the box, servo opens it to feed.



ATTENTION

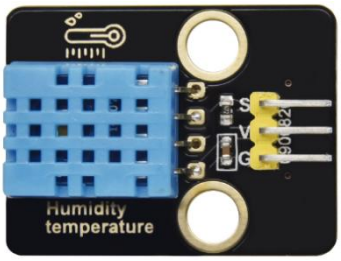
Do not put your fingers into the box to avoid nipping!

Do not block the door with something to avoid damaging servo!

4.7 Temperature Control System

7.1 DHT11 temperature and humidity sensor

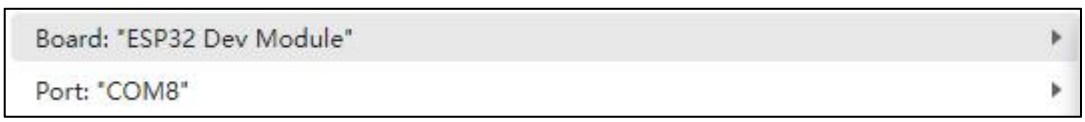
DHT11 temperature and humidity sensor outputs digital signals. It applies principles of analog signal acquisition and conversion as well as temperature and humidity sensing technology, so that it features long-term stability and high reliability. Besides, the sensor integrates a high-precision resistive humidity sensor and a resistive thermosensitive temperature sensor, and is connected with an 8-bit high-performance MCU.



Open the **6.3Intelligent-Feeding-System** code with Arduino IDE.

```
#include <dht11.h>
#define DHT11PIN 17

dht11 DHT11;

void setup()
{
  Serial.begin(9600);
  Serial.println("DHT11 TEST PROGRAM ");
  Serial.print("LIBRARY VERSION: ");
  Serial.println(DHT11LIB_VERSION);
  Serial.println();
}

void loop()
{
  Serial.println("\n");
```

Choose the **ESP32 Dev Module** board and **COM** port, and upload the code.

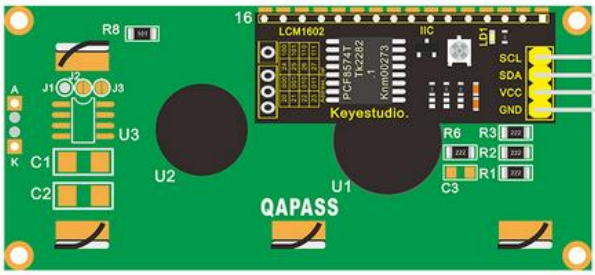


Test Result:

Open the serial monitor and set the baud rate to 9600, the serial monitor will display the the current temperature and humidity value.

7.2 LCD 1602 Module

LCD 1602 possesses a standard 14-pin (without backlight) or 16-pin (with backlight) interface, saving the pins of MCU. Its display drives IC to realize I2C control.



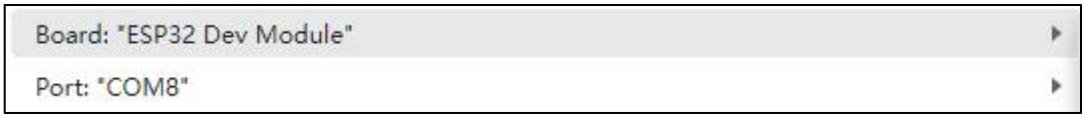
Specifications

Power supply range: 5V
Working current: <130mA
Recommended ambient temperature: -10°C ~ 50°C
Product size: 80mm * 36mm
IIC address: 0x27

Open the **7.2LCD1602** code with Arduino IDE.

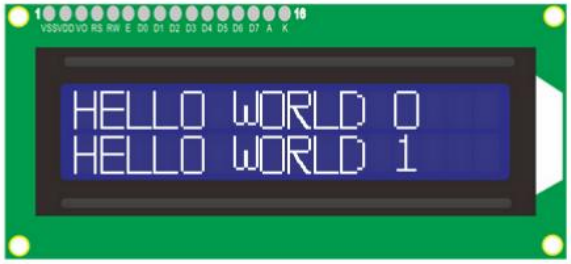
```
#include <LiquidCrystal_I2C.h>

//Initialize LCD 1602, 0x27 is I2C address
LiquidCrystal_I2C lcd(0x27,16,2);

void setup() {
  //Initialize LCD
  lcd.init();
  // Turn the (optional) backlight off/on
  lcd.backlight();
  //lcd.noBacklight();

  //Set the position o dcursor
  lcd.setCursor(0, 0);
  //LCD prints
  lcd.print("HELLO WORLD 0");
  lcd.setCursor(0, 1);
```

Choose the **ESP32 Dev Module** board and **COM** port, and upload the code.



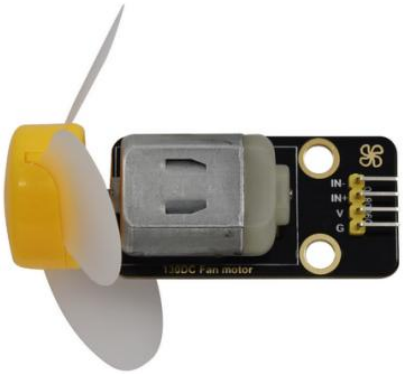
Test Result:

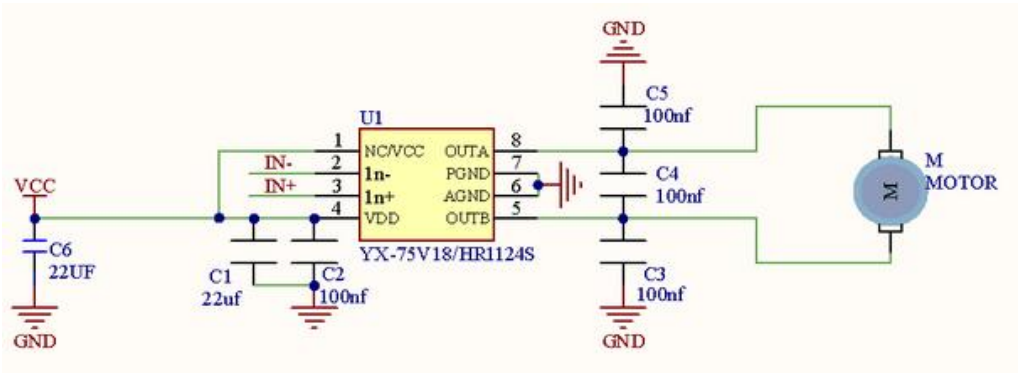
LCD1602 opens its backlight and displays ”HELLO WORLD 0“ and ”HELLO WORLD 1“.



7.3 Motor and Fan

130 Motor is able to adjust speed via PWM. In the process, two pins are needed to be connected for controlling.

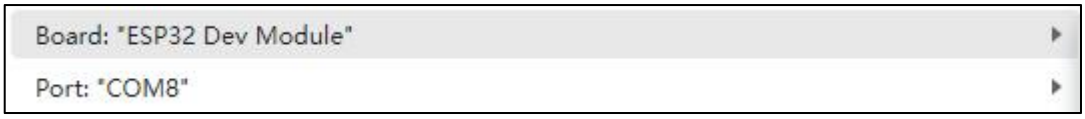




Open the **7.3Motor** code with Arduino IDE.

```
#define MotorPin1 19//(IN+)
#define MotorPin2 18//(IN-)

void setup() {
  pinMode(MotorPin1,OUTPUT);
  pinMode(MotorPin2,OUTPUT);

  // Set PWM output to adjust the speed of motor
  ledcSetup(1, 1200, 8); // Set frequency of LEDC Channel 1 to 1200, PWM resolution to 8, so duty cycle
  ledcAttachPin(MotorPin1, 1); // Bound LEDC Channel 1 to the specified left motor pin gpio19 to output
  ledcSetup(3, 1200, 8); // Set frequency of LEDC Channel 3 to 1200, PWM resolution to 8, so duty cycle
  ledcAttachPin(MotorPin2, 3); // Bound LEDC Channel 3 to the specified left motor pin gpio18 to output
}

void loop() {
  //Turn left
  ledcWrite(1, 70);
```

Choose the **ESP32 Dev Module** board and **COM** port, and upload the code.



Test Result:

130 motor alternatively rotates left and right every 2 seconds.

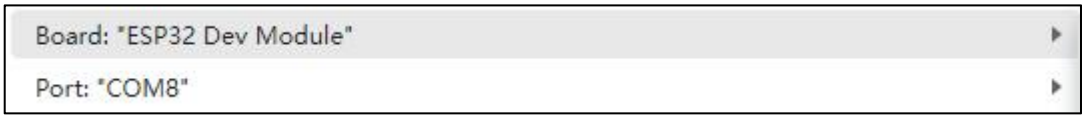
NOTE: Since the fan is a high-power electronic device, please remember to use batteries to power it.

7.4 Temperature Control System

Open the **7.4Temperature-Control-System** code with Arduino IDE.

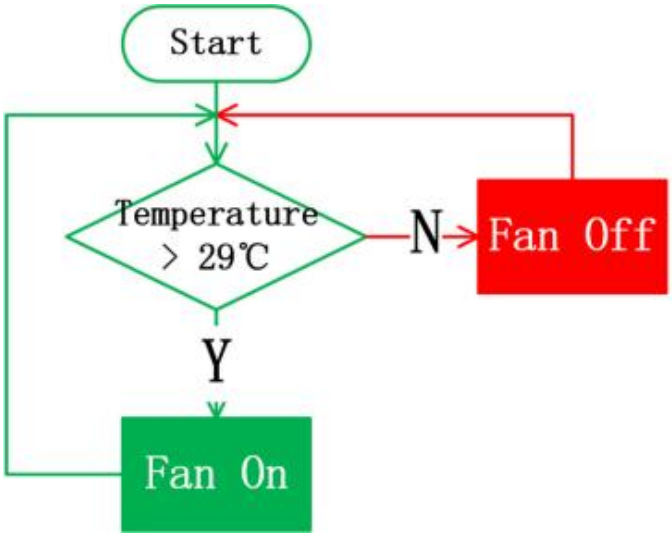
```
#include <LiquidCrystal_I2C.h>
#include <dht11.h>

#define DHT11PIN 17
#define MotorPin1 19//(IN+)
#define MotorPin2 18//(IN-)

dht11 DHT11;

LiquidCrystal_I2C lcd(0x27,16,2);

void setup() {
  lcd.init();
  lcd.backlight();

  pinMode(MotorPin1,OUTPUT);
  pinMode(MotorPin2,OUTPUT);

  //Set PWM output to adjust the speed of motor
  ledcSetup(1, 1200, 8);//Set frequency of LEDC Channel 1 to 1200, PWM resolution to 8, so duty cycle =
  ledcAttachPin(MotorPin1, 1);  //Bound LEDC Channel 1 to the specified left motor pin gpio19 to output
```

Choose the **ESP32 Dev Module** board and **COM** port, and upload the code.



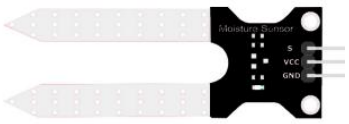
Test Result:

When the temperature reaches 29°C, the fan will turn on to dissipate heat. When it is lower than 29°C, the fan will turn off (the fan just simulates heat dissipation, so the effect is not good), which saves energy for the farm.



4.8 Soil Humidity Monitoring System

8.1 Soil Humidity Sensor



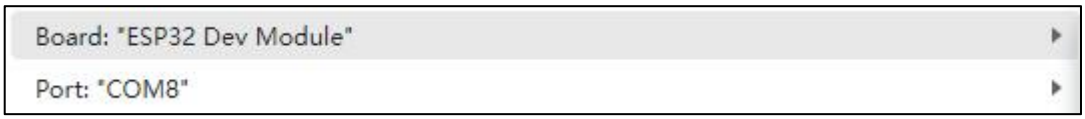
Pay attention!

Do not overflow water from plastic pools in experiments. Spilling water on other sensors may cause not only a short circuit or modules to be out of work but also heat generation and even explosion. Do be extra careful! Especially for younger users, please operate with your parents.

Open the **8.1Soil-Humidity-Sensor** code with Arduino IDE.

```
#define SoilHumidityPin 32

void setup() {
  Serial.begin(9600);
  pinMode(SoilHumidityPin,INPUT);
}

void loop() {
  //Define a variab as the value of soil humidity sensor
  int ReadValue = analogRead(SoilHumidityPin);
  Serial.println(ReadValue);
  delay(500);
}
```

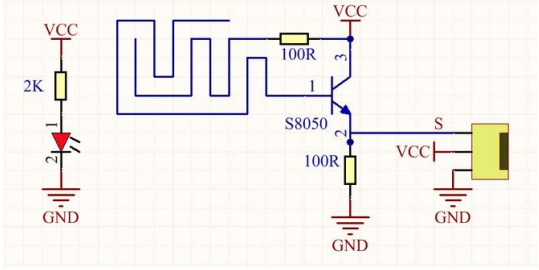
Choose the **ESP32 Dev Module** board and **COM** port, and upload the code.



Test Result:

Open the serial monitor. Touch the detection area of the sensor with a wet finger and the currently detected humidity value will be printed on the monitor (range: 0~4095).

Soil humidity sensors are mainly used to measure water content in volumetric soil, monitor soil moisture, irrigate crops and protect forests.



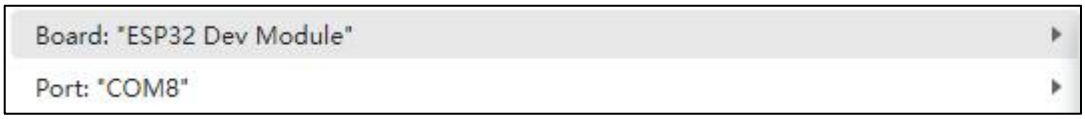
8.2 Soil Humidity Monitoring System

We adopt LCD1602 to reveal the real-time value of soil humidity value. When the value is lower than the set minimum humidity, the buzzer will emit sound to prompt farmers of irrigation.

Open the **8.2Soil-Humidity-Testing-System** code with Arduino IDE.

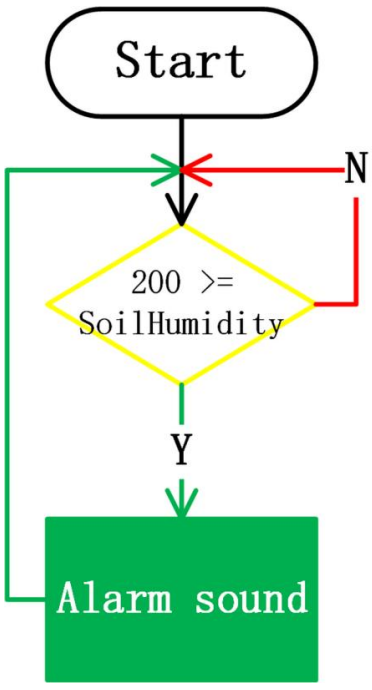
```
#include <LiquidCrystal_I2C.h>

#define BuzzerPin 16
#define SoilHumidityPin 32

LiquidCrystal_I2C lcd(0x27,16,2);

void setup() {

  pinMode(BuzzerPin,OUTPUT);
  pinMode(SoilHumidityPin,INPUT);

  lcd.init();
  lcd.backlight();
  lcd.clear();
```

Choose the **ESP32 Dev Module** board and **COM** port, and upload the code.



Test Result:

The LCD1602 displays the reveal the real-time value of soil humidity value. When the value detected by the soil humidity sensor is lower than 200, the buzzer emits sound to alarm.



Pay attention!

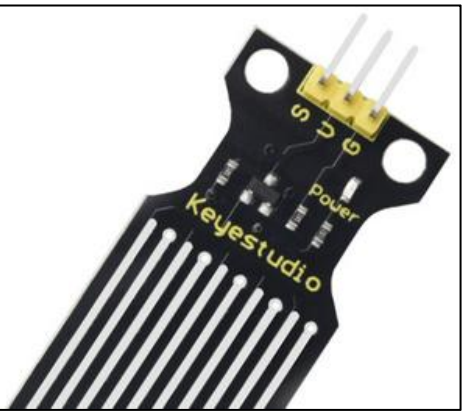
Do not overflow water from plastic pools in experiments. Spilling water on other sensors may cause not only a short circuit or modules to be out of work but also heat generation and even explosion. Do be extra careful! Especially for younger users, please operate with your parents.

4.9 Water Level Monitoring System

9.1 Water Level Sensor

The water level sensor integrates a series of exposed parallel lines to measure the volume of water and droplets.

- Size: 65mm x 20mm x 8mm
- Easy- to-use
- Portable
- Voltage: DC5V
- Current: < 20mA
- Working Humidity: 10%-90% without condensation
- N.W.:3.8g



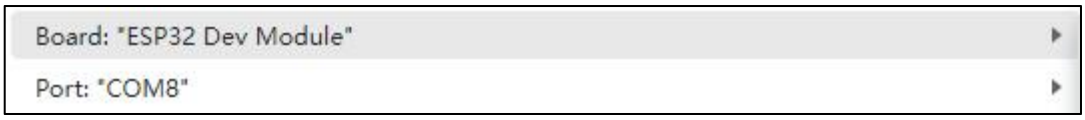
Pay attention!

With the exception of the detection area, the sensor is not waterproof. Spilling water on other area may result in a short circuit.

Open the **9.1Water-Level-Sensor** code with Arduino IDE.

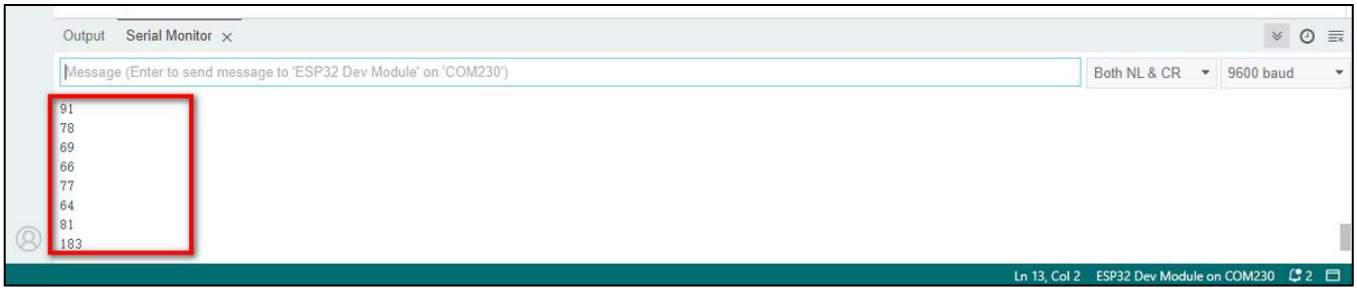
```
#define WaterLevelPin 33

void setup() {

  Serial.begin(9600);
  pinMode(WaterLevelPin,INPUT);
}

void loop() {
  int ReadValue = analogRead(WaterLevelPin);
  Serial.println(ReadValue);
  delay(500);
}
```

Choose the **ESP32 Dev Module** board and **COM** port, and upload the code.



Test Result:

Open the serial monitor. Touch the detection area of the sensor with a wet finger and the currently detected value will be printed on the monitor (range: 0~4095).

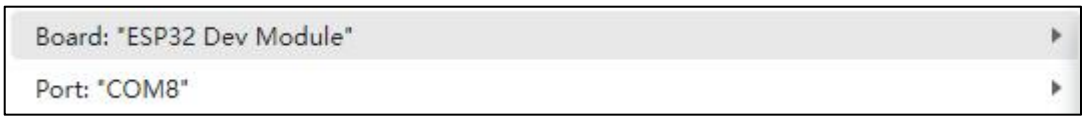


9.2 Water Level Monitoring System

Open the **9.2Water-Level-Testing-System** code with Arduino IDE.

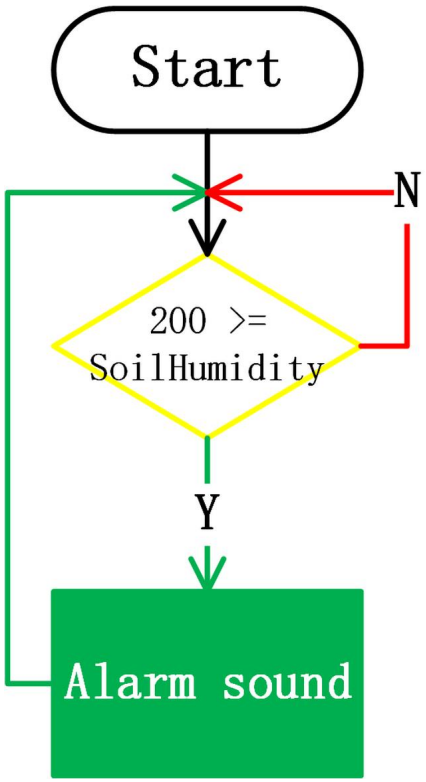
```
#include <LiquidCrystal_I2C.h>

#define BuzzerPin 16
#define WaterLevelPin 33

LiquidCrystal_I2C lcd(0x27,16,2);

void setup() {

  //Initialize the serial port
  Serial.begin(9600);
  pinMode(WaterLevelPin,INPUT); //Set the water level pin to input mode

  lcd.init(); //Initialize LCD
  lcd.backlight();  //turn on the LCD backlight
  lcd.clear();  //clear displays on LCD
```

Choose the **ESP32 Dev Module** board and **COM** port, and upload the code.



Test Result:

LCD displays the real-time value of water level. When the water level sensor detects that the water level is lower than 200, the buzzer starts to alarm.



4.10 Auto-Irrigation System

10.1 Water Pumping System

In this experiment, we use ESP32 development board to turn on/off the water pump by a relay module. A pump lifts water and transports liquids, and usually is combined with a relay module in usage.

Relay Module:

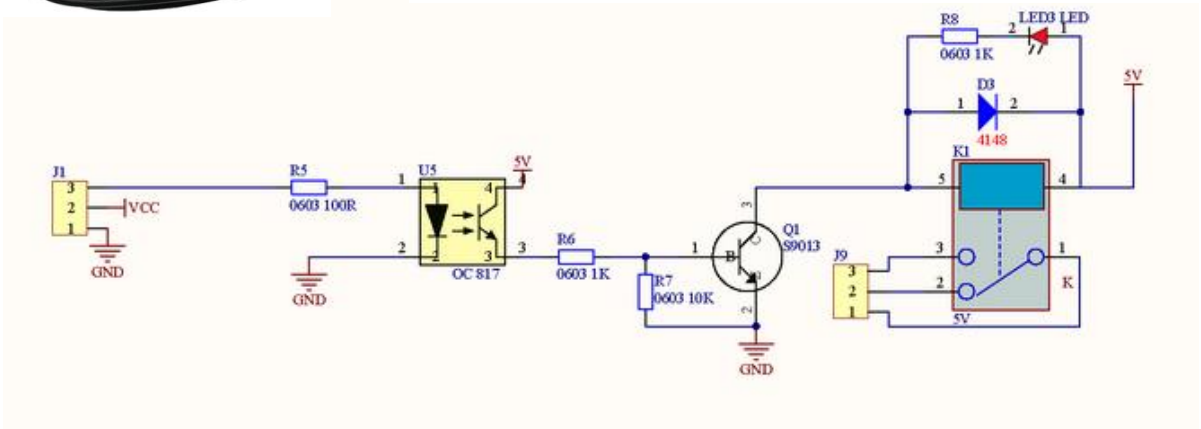
In usage, it is often used in the management of high voltage and load current, say, motors, high-current sensors and high-power light.

- **Normally Open (NO):** This pin is normally open, unless a signal is received by the signal pin of the relay. Therefore, common pins are disconnected via NC pin and connected through NO pin.
- **Common Contact (COM):** This pin connects to other modules, for example, water pump.
- **Normally Closed (NC):** NC pin is linked with COM pin to form a closed circuit. It uses ESP32 board to control the closure and the disconnection of the relay module.

Water Pump:

Parameters:

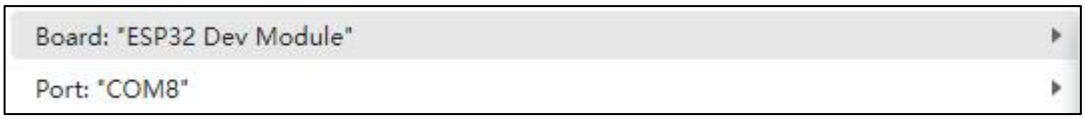
- Power voltage: 5V
- Static current: 2mA
- Maximum contact voltage: 250VAC/30VDC
- Maximum current: 10A



Open the **10.1Water-Pump** code with Arduino IDE.

```
#define RelayPin 25

void setup() {
  Serial.begin(9600);
  pinMode(RelayPin,OUTPUT);
}

void loop() {
  //Serial.read() receives one byte once. For example, when input "aaa", it receives one "a" at a time
  if(Serial.available() > 0) {
    if (Serial.read() == 'a') //When the input value equals to "a", irrigation begins.
    {
      digitalWrite(RelayPin,HIGH);
      delay(400);//irrigation delay
      digitalWrite(RelayPin,LOW);
      delay(700);
```

Choose the **ESP32 Dev Module** board and **COM** port, and upload the code.



Test Result:

Open the serial monitor and **input "a", pump water once**.

Pay attention!

Do not overflow water from plastic pools in experiments. Spilling water on other sensors may cause not only a short circuit or modules to be out of work but also heat generation and even explosion. Do be extra careful! Especially for younger users, please operate with your parents.

10.2 Auto-Irrigation System

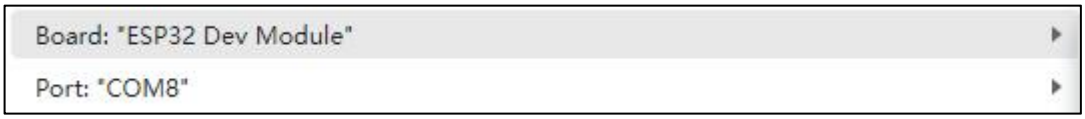
In this experiment, we connect the two sensors on ESP32 development board and program to read their output values to control the relay and water pump.

If the soil is very dry, the relay will turn on to control the water pump to irrigate plants; And if the water level is too low, the water pump will not be able to work, and the buzzer will alarm

Open the **10.2Auto-irrigation** code with Arduino IDE.

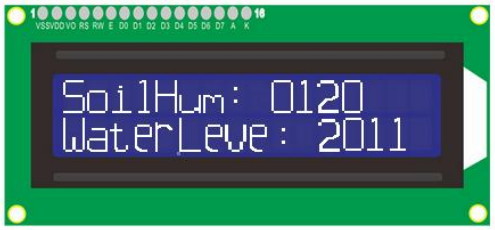
```
#include <LiquidCrystal_I2C.h>

#define BuzzerPin 16
#define SoilHumidityPin 32
#define WaterLevelPin 33
#define RelayPin 25
#define ButtonPin 5 //Define a button pin
int value = 0;      //Set an initial button value

LiquidCrystal_I2C lcd(0x27,16,2);

void setup() {
  //Set the pins mode
  pinMode(BuzzerPin,OUTPUT);
  pinMode(SoilHumidityPin,INPUT);
```

Choose the **ESP32 Dev Module** board and **COM** port, and upload the code.



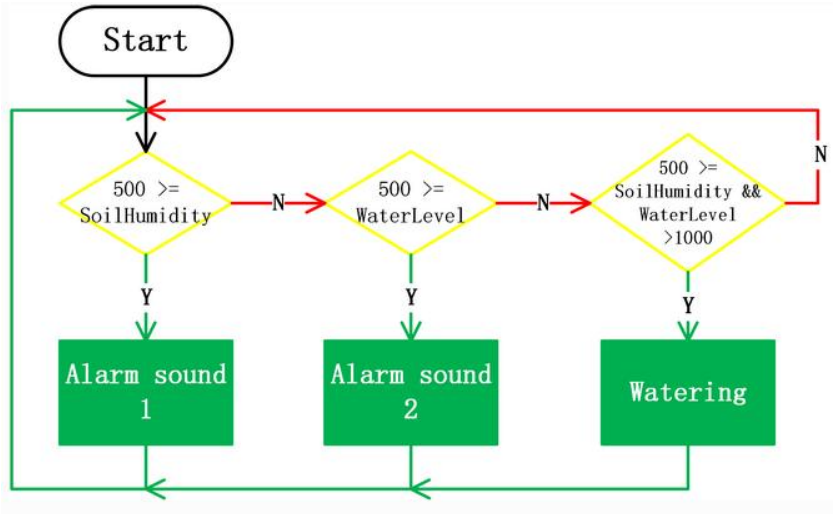
Test Result:

LCD 1602 will display the current value of soil humidity and water level.



When the detected soil humidity value is lower than 500, the buzzer alarms to notify that the soil is being arid. If the water level value is greater than 1000, the irrigation starts automatically.

When the detected water level is lower than 500, the water pumping system doesn't work, and the buzzer alarms to notify that water is insufficient. Press the button to stop alarming.



4.11 Web-controlled Smart Farm

11.0 Connect the ESP32 Board to the Network

ESP32 board is equipped with Wi-Fi(**2.4G**) and Bluetooth(4.2), which enable it to easily connect to WiFi and communicate with other devices on the network.

What do you need to prepare:

-A **2.4 GHz** WiFi(It can be a mobile hotspot or a router)
-The WIFI name and password
-A phone/IPAD/computer that can connect to the same WiFi.

Arduino IDE provides you wih library file <WiFi.h>, which support Wi-Fi configurations and ESP32 Wi-Fi networking monitoring.

- **Base station mode** (STA or Wi-Fi client-side mode): In this mode, ESP32 connects to the Wi-Fi hotspot (AP).
- **AP mode** (Soft-AP or Wi-Fi hotspot mode): In this mode, other Wi-Fi devices connect to ESP32.
- **AP-STA mode**: In this mode, ESP32 is a Wi-Fi hotspot as well as a Wi-Fi device connecting to another Wi-Fi hotspot.
- These modes is compatible with multiple safe modes, like WPA, WPA2 and WEP.
- It is able to scan for Wi-Fi hotspot, including active and passive scan.
- It supports promiscuous mode to monitor IEEE802.11 Wi-Fi Packets.

For wifi details, please refer to:

https://docs.espressif.com/projects/esp-idf/en/latest/esp32/api-reference/network/esp_wifi.html

ESPRESSIF official website: https://www.espressif.com.cn/en/home

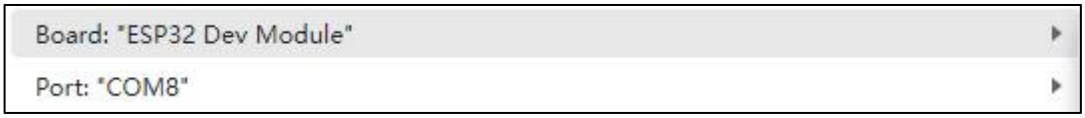
Open the **11.0Connect-the-ESP32-to-the-Network** code with Arduino IDE.

```
#include <WiFi.h>

const char* ssid = "your_SSID";
const char* password = "your_PASSWORD";

void setup() {
  Serial.begin(9600);
  //Initialize Wifi
  WiFi.begin(ssid, password);
  //Scan for wifi. If connection fails, stay in connecting, and execute "while" loop
  while (WiFi.status() != WL_CONNECTED) {
    delay(1000);
    Serial.println("Connecting to WiFi...");
  }
  //Connected. Print the IP address
  Serial.println("Connected to WiFi");
  Serial.println(WiFi.localIP());
}

void loop() {
}
```

Change your_SSID in the code to the name of your wifi, and your_PASSWORD to the wifi password

```
#include <WiFi.h>

const char* ssid = "your_SSID";
const char* password = "your_PASSWORD";
```

Choose the **ESP32 Dev Module** board and **COM** port, and upload the code.



Test Result:

Upload the code, and the board will connect to Wi-Fi network and print the IP Address on the serial monitor.

11.1 Set Up a Website-HELLOWORLD

As long as connecting to Wi-Fi, Web server library of ESP32 is able to provide web pages. In the following example code, we set up a simple website to show “Hello, World!”.

Open the **11.1WiFi-HTML-HELLOWORLD** code with Arduino IDE.

```
#include <WiFi.h>
#include <WebServer.h>

const char* ssid = "your_SSID";
const char* password = "your_PASSWORD";

WebServer server(80); //Set the server port to 80. Enter the website by IP address rather than the port

//Initialize the website
void handleRoot() {
  //Used to send HTTP to the client-side for response, sending 200 means success.
  server.send(200, "text/html", "<h1>Hello, World!</h1>");
}

void setup() {
  Serial.begin(9600);
  //Initialize wifi
  WiFi.begin(ssid, password);
```

Change your_SSID in the code to the name of your wifi, and your_PASSWORD to the wifi password. Then upload the code.

```
#include <WiFi.h>

const char* ssid = "your_SSID";
const char* password = "your_PASSWORD";
```

Test Result:

In this example code, we establish a Web server by WebServer library on ESP32. The function handleRoot() asks for processing in root path and sends HTML response of “Hello, World!” to client-side. Then, setup() sets the root route, and server.begin() starts the Web server.

Click on the serial monitor to view the IP address：

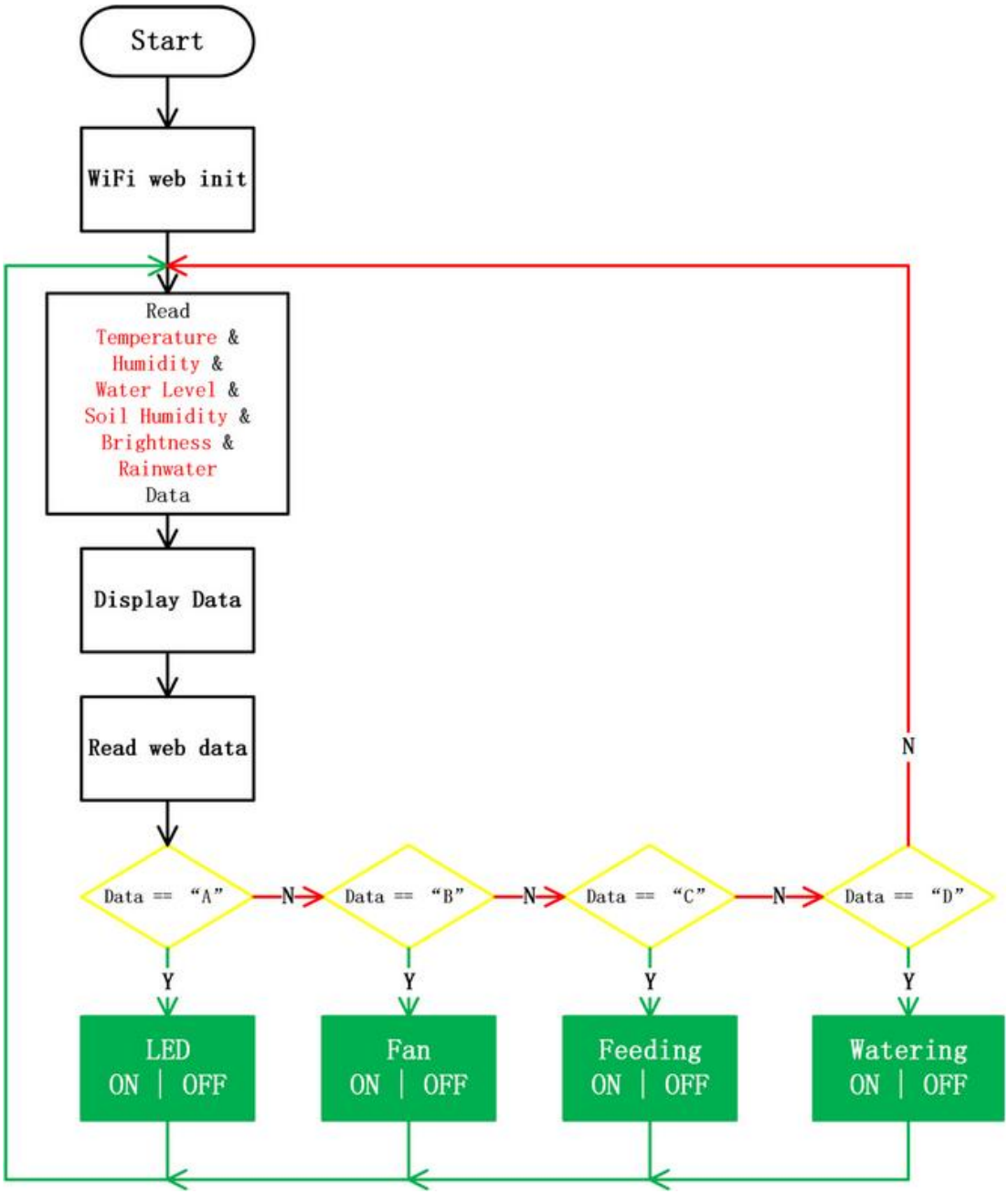
NOTE: When PC, mobile phones and ESP32 board are connected to one network, you can visit this website at PC and phones at the same time.

Access the IP in the PC browser or phone browser:

NOTE:

Note: Requires **2.4 GHz** WIFI, not 5G. The PC or mobile phone accessing the IP address needs to be connected to the same WIFI as the ESP32 board

11.2 Web-controlled smart farm



Open the **11.2WiFi-HTML-Smart-Farm** code with Arduino IDE.

```
#include <Arduino.h>
/* Determine which development board it is (ESP32 or 8266).
The library files of these two boards are separated, so the corresponding library should be imported to

#ifdef ESP32
  #include <WiFi.h>
  #include <AsyncTCP.h>
#elif defined(ESP8266)
  #include <ESP8266WiFi.h>
  #include <ESPAsyncTCP.h>
#endif
// Import library
#include <ESPAsyncWebServer.h>
#include <LiquidCrystal_I2C.h>
#include <dht11.h>
#include <analogWrite.h>
#include <ESP32_Servo.h>

#define DHT11PIN        17  //Temperature and humiddity sensor pin
```

Change your_SSID in the code to the name of your wifi, and your_PASSWORD to the wifi password. Then upload the code.

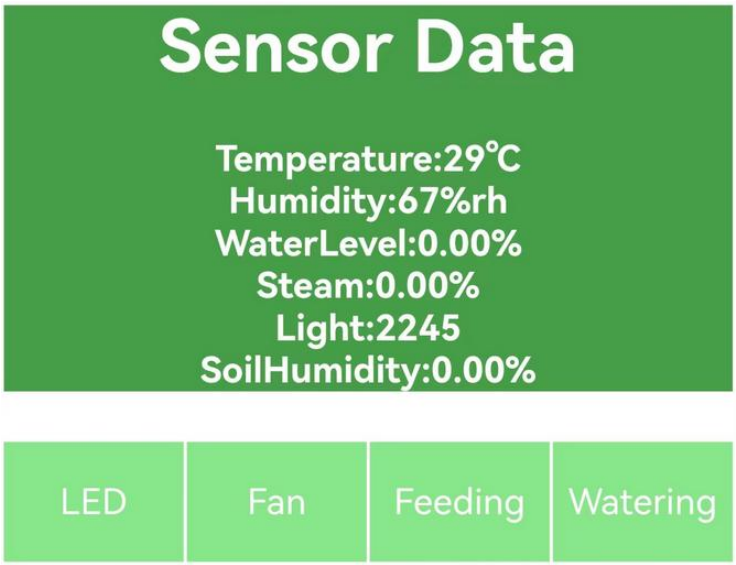
```
#include <WiFi.h>

const char* ssid = "your_SSID";
const char* password = "your_PASSWORD";
```

Test Result:

NOTE: When PC, mobile phones and ESP32 board are connected to one network, you can visit this website at PC and phones at the same time.

Click on the serial monitor to view the IP address：

Input the IP address in browsers at mobile phones or PC, you can see the sensor values at the top of the page and control the LED, fan, feeding cabin and water pump with the buttons below.



Note: Requires **2.4 GHz** WIFI, not 5G. The PC or mobile phone accessing the IP address needs to be connected to the same WIFI as the ESP32 board

4.12 APP Control Smart Farm

Pay attention: ***Do not overflow water from plastic pools in experiments. Spilling water on other sensors may cause a short circuit or modules to be out of work. If batteries get wet, even explosion may occur. Do be extra careful! For younger users, please operate with your parents.*** ***Use batteries for power instead of just USB.***

Open the **12.1APP-Smart-Farm** code with Arduino IDE.

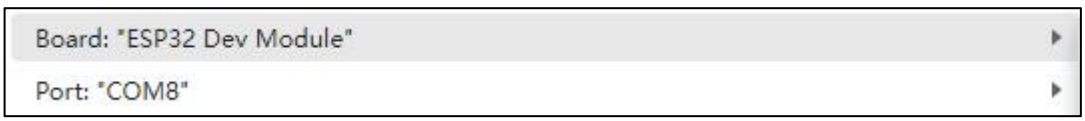
Note: "BuzzerMusic.h" and "12.1APP-Smart-Farm.ino" need to be placed in the same folder, do not separate them.

```
#include <Arduino.h>
#ifdef ESP32
  #include <WiFi.h>
#elif defined(ESP8266)
  #include <ESP8266WiFi.h>
#endif

#include <dht11.h>
#include <analogWrite.h>
#include <ESP32_Servo.h>
#include <LiquidCrystal_I2C.h>
#include "BuzzerMusic.h"

//To be displayed
#define DHT11PIN        17  //Temperature and humidity sensor pin
#define RAINWATERPIN    35  //Steam sensor pin
#define LIGHTPIN        34  //Photoresistor pin
#define WATERLEVELPIN   33  //Water level sensor pin
#define SOILHUMIDITYPIN 32  //Soil humidity sensor pin
//To be controlled
#define LEDPIN          27  //LED pin
#define RELAYPIN        25  //Relay pin (to control water pump)
#define SERVOPIN        26  //Servo pin
```

Change your_SSID in the code to the name of your wifi, and your_PASSWORD to the wifi password

```
#include <WiFi.h>

const char* ssid = "your_SSID";
const char* password = "your_PASSWORD";
```

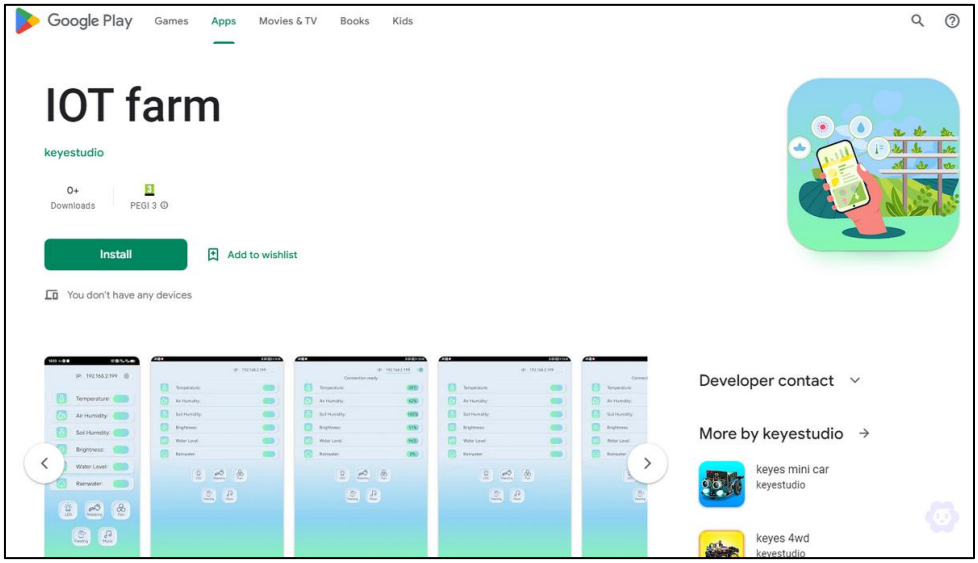
Choose the **ESP32 Dev Module** board and **COM** port, and upload the code.



Download APP

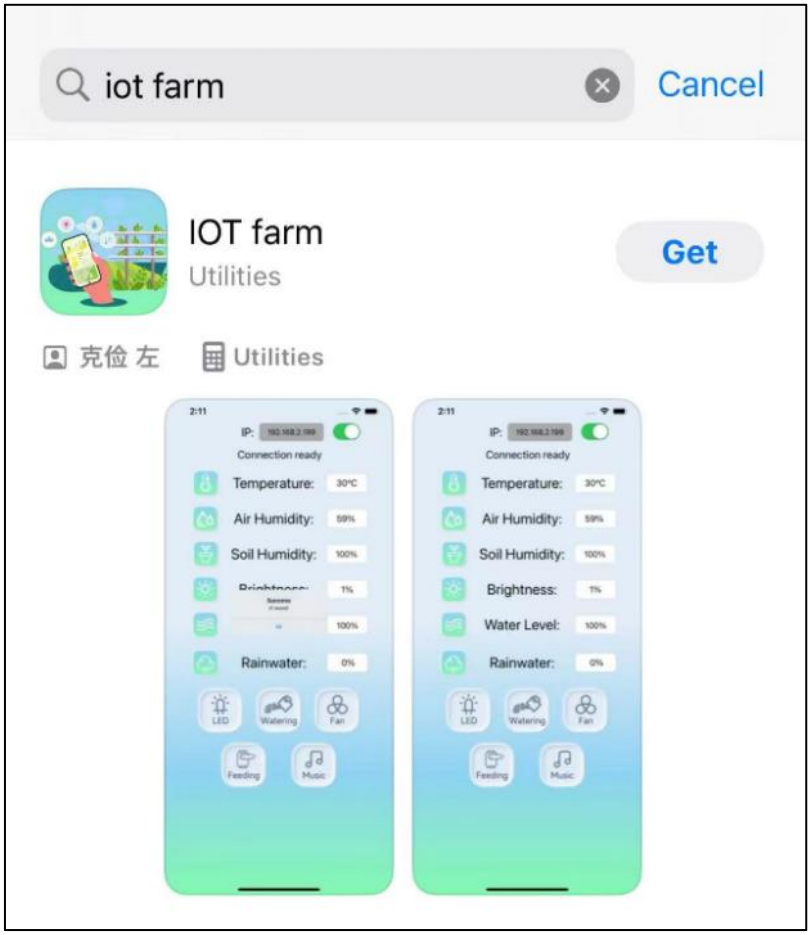
For Android：

Method 1: Search "IOT Farm" in Google Play and download it.

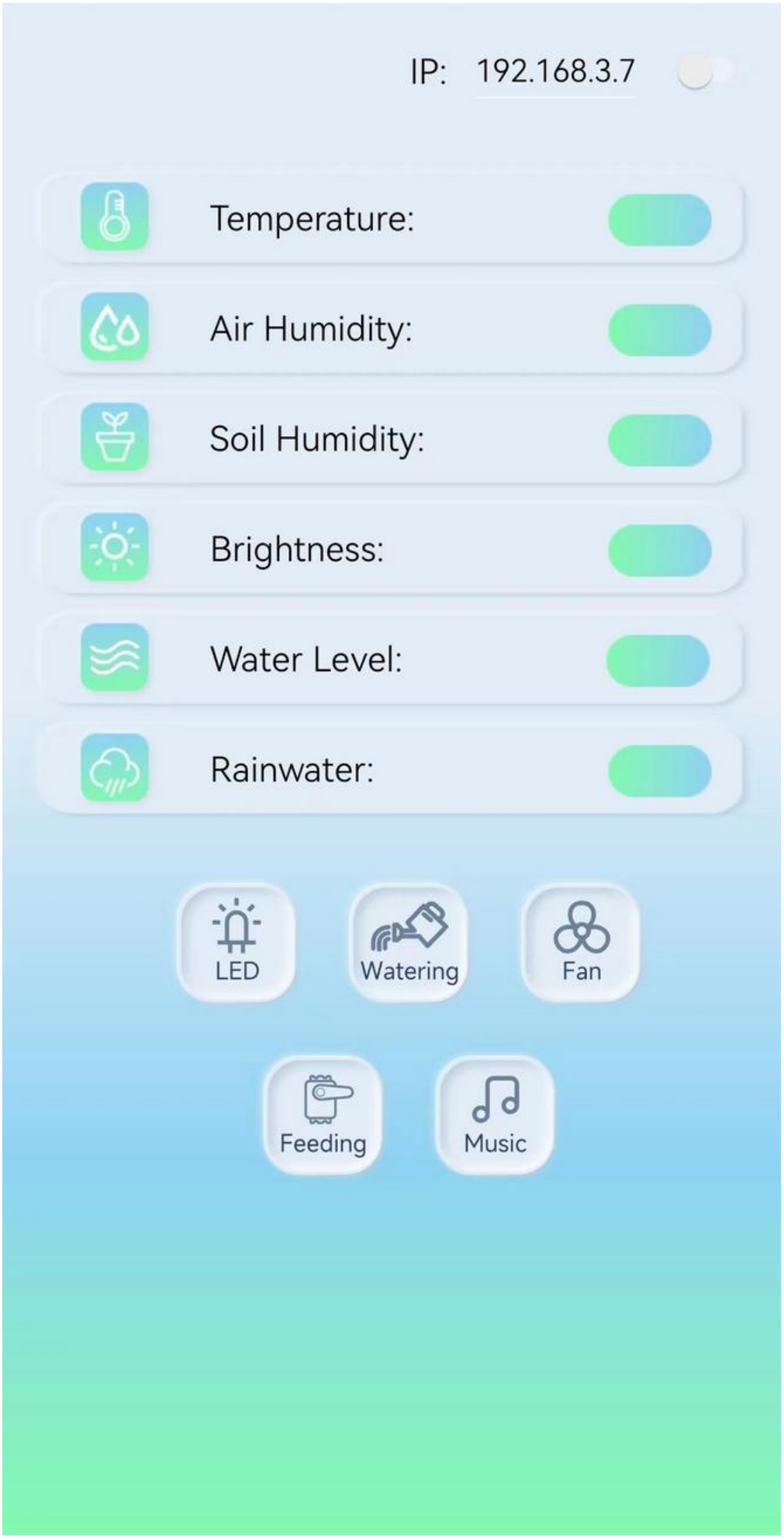


For iOS：

Search IOT farm in APP Store and tap to download.

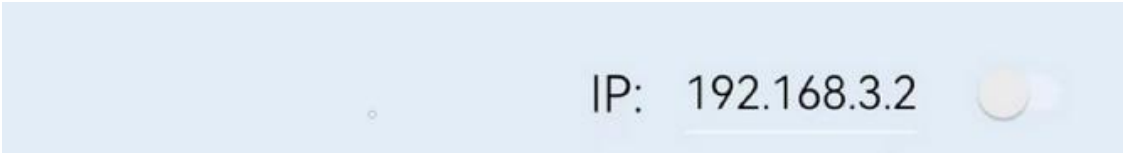


The home page of the APP



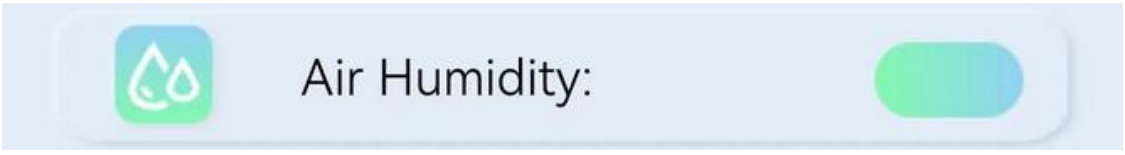
APP Function Description

1. After upload the code, connect the phone to the same WIFI as the ESP32, you only need to input IP address at upper-right conner to connect. **Note:** Requires **2.4 GHz** WIFI, not 5G.

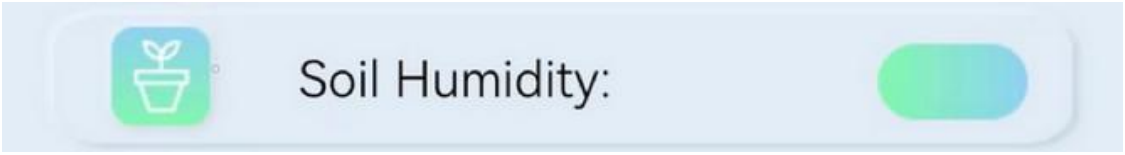


2. Displays the temperature value of the farm in real time.

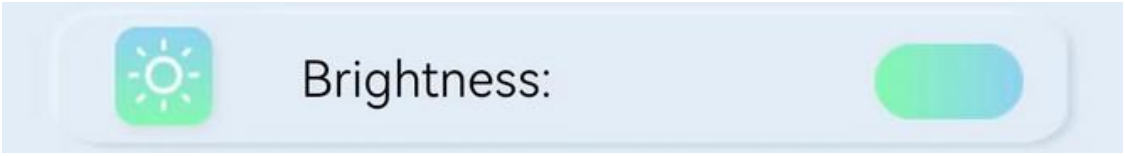
3. Displays the air humidity value of the farm in real time.



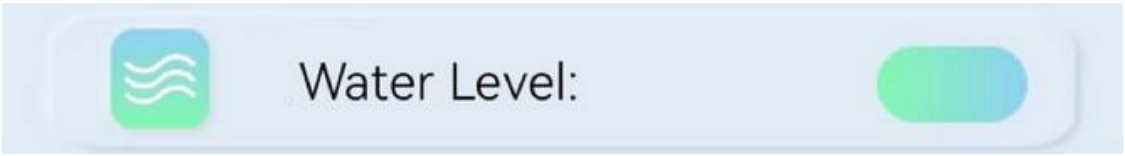
4. Displays the soil humidity value of the farm in real time.



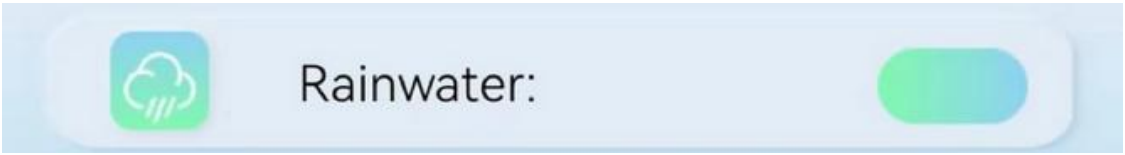
5. Displays the sun brightness value of the farm in real time.



6. Displays the water level of the farm in real time.



7. Displays the analog rainfall value of the farm in real time.



8. Control LED.

9. Control irrigation via water pump.

10. Control the fan to adjust temperature.

11. Control servo to open or close feeding box.

12. Control buzzer to play music.