**Keyestudio IoT Smart Home Kit for ESP32**

Contents

[1. Description 3](#_Toc2619)

[2. Features 3](#_Toc23233)

[3. Parameters 4](#_Toc7100)

[4. Installation of arduino IDE 4](#_Toc1558)

[5. Installation of Structure 5](#_Toc12376)

[6. Projects 5](#_Toc25104)

[Project 1: Control LED 6](#_Toc13852)

[Project 1.1 LED Flashing 7](#_Toc20047)

[Project 1.2 Breathing LED 8](#_Toc6370)

[Project 2: Table Lamp 10](#_Toc20241)

[Project 2.1 Read the Button 10](#_Toc15482)

[Project 2.2. Table Lamp 12](#_Toc3919)

[Project 3: PIR Motion Sensor 14](#_Toc9430)

[Project 3.1 Read the PIR Motion Sensor 15](#_Toc23709)

[Project 3.2 PIR Motion Sensor 16](#_Toc9437)

[Project 4: Play Music 17](#_Toc13136)

[Project 4.1 Play Happy Birthday 18](#_Toc28818)

[Project 4.2 Music Box 19](#_Toc25708)

[Project 5: Automatic Doors and Windows 21](#_Toc5550)

[Project 5.1 Control the Door 24](#_Toc27752)

[Project 5.2 Close the Window 25](#_Toc13799)

[Project 6: Atmosphere Lamp 26](#_Toc2263)

[Project 6.1 Control SK6812 28](#_Toc30829)

[Project 6.2 Button 29](#_Toc19560)

[Project 7: Fan 30](#_Toc5231)

[Project 7.1 Control the Fan 31](#_Toc11234)

[Project 7.2 Switch On or Off the Fan 32](#_Toc9379)

[Project 8: LCD1602 Display 35](#_Toc5125)

[Project 8.1 Display Characters 36](#_Toc13345)

[Project 8.2 Dangerous Gas Alarm 36](#_Toc22393)

[Project 9: Temperature and Humidity Sensor 39](#_Toc14104)

[Project 9.1 Temperature and Humidity Tester 39](#_Toc32053)

[Project 10: RFID RC522 Module 41](#_Toc30242)

[Project 10.1 Open the Door 42](#_Toc31482)

[Project 11: Morse Code  45](#_Toc26949)

[Project 11.1 Morse Code Open the Door 46](#_Toc1244)

[Project 12: WiFi 49](#_Toc10206)

[Project 12.1 Smart Home 50](#_Toc8692)

[Project 12.2 Control Smart Home 54](#_Toc32012)

[Project 13: Mobile Phone APP 58](#_Toc30632)

[Project 13.1 Test APP 59](#_Toc27736)

[Project 13.2 IoT Smart Home 65](#_Toc2095)

[7. Resources 66](#_Toc4263)

# Description

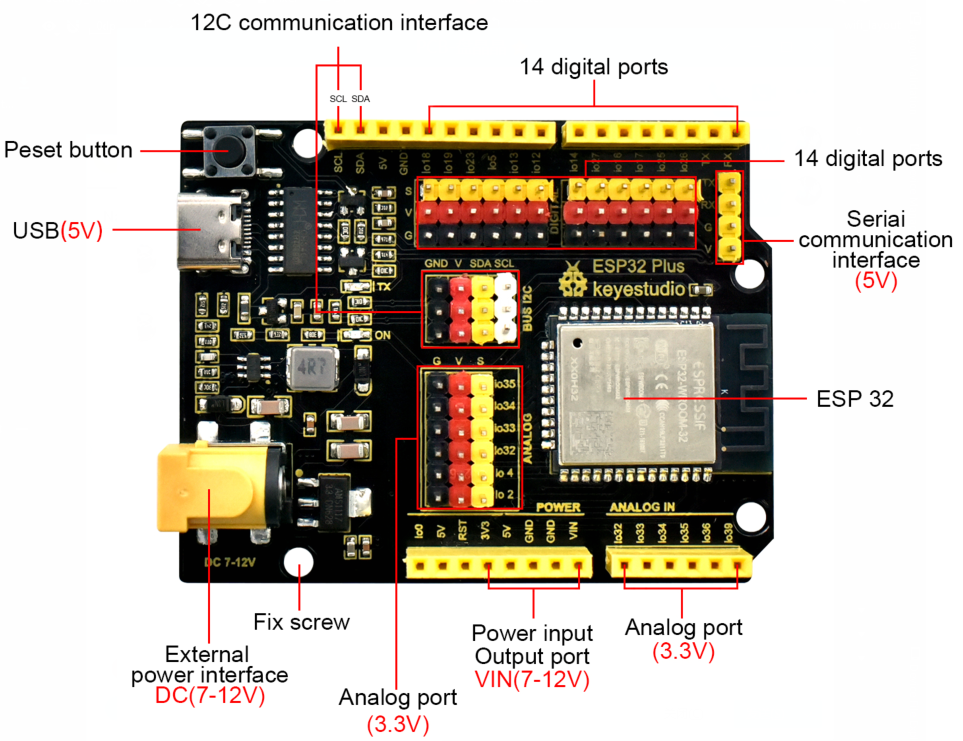
As the rapid development of the Internet grows, various intelligent devices are gradually integrated into our daily life. For example, we can use RFID to open the door. In addition, the kitchen is equipped with a gas detection alarm, which alerts people to the danger when dangerous gas and large smoke are detected. When it detects rain, it can automatically collect clothes and close windows. All kinds of electrical equipment can be controlled by mobile phone, control lights, fans, air conditioning and so on.

In this connection, we seek to launch this smart home product with ESP32 control, which has a host of sensors and modules as well as networking function, making the relevant knowledge of the Internet more accessible to you.

# Features

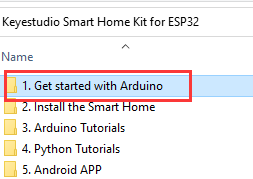
1. Elegant appearance
2. A host of sensor modules
3. Mobile phone APP network control
4. Morse password door
5. It can automatically close windows
6. RFID function
7. C language and MicroPython

# Parameters



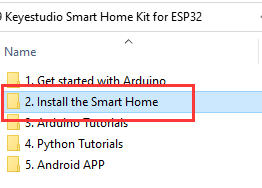
# Installation of arduino IDE

Please open the files below for arduino IDE installation, which contain arduino IDE installation, ESP32 environment installation, add library files as well as CH340 driver installation.



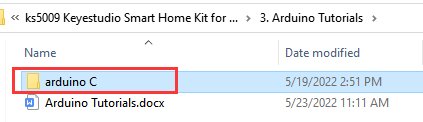
# Installation of Structure

Please open the files below for structural installation



# Projects

Sample code for all projects is in the folder, as shown below:



## Project 1: Control LED

We will first learn how to control LED.



1. **Working Principle**

LED is also the light-emitting diode, which can be made into an electronic module. It will shine if we control pins to output high level, otherwise it will be off.

1. **[Parameter](C:/Users/NINGMEI/AppData/Local/youdao/dict/Application/8.10.7.0/resultui/html/index.html" \l "/javascript:;)s**

|  |  |
| --- | --- |
| Working voltage | DC 3~5V |
| Working current | <20mA |
| Power | 0.1W |

1. **Control Pin**

|  |  |
| --- | --- |
| Yellow LED | 12 |

### Project 1.1 **LED Flashing**

1. **Description**

We can make the LED pin output high level and low level to make the LED flash.

1. **Test Code**

|  |
| --- |
| #define led\_y 12 //Define the yellow led pin to 12  void setup() { //The code inside the setup function runs only once  pinMode(led\_y, OUTPUT); //Set pin to output mode  }  void loop() { //The code inside the loop function will always run in a loop  digitalWrite(led\_y, HIGH); //Light up the LED  delay(200); //Delay statement, in ms  digitalWrite(led\_y, LOW); //Close the LED  delay(200);  } |

1. **Test Result**

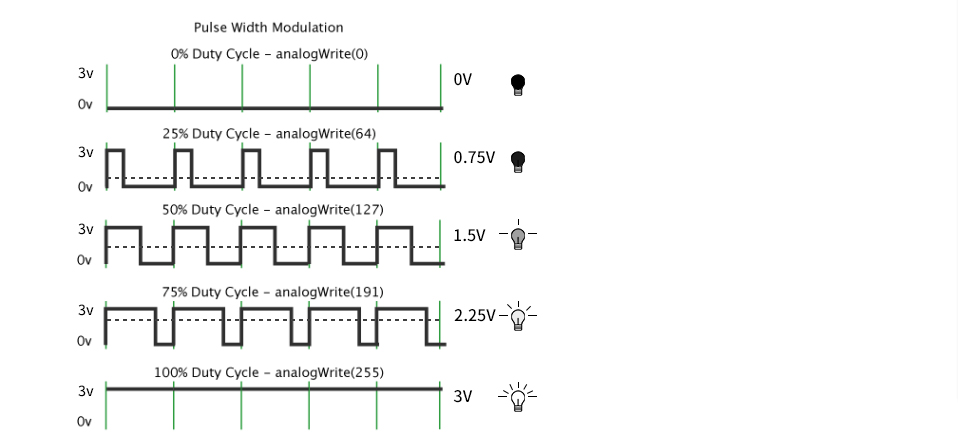
After uploading the code , you can see white and yellow LEDs flashing together.

### Project 1.2 Breathing LED

1. **Description**

A“breathing LED”is a phenomenon where an LED's brightness smoothly changes from dark to bright and back to dark, continuing to do so and giving the illusion of an LED“breathing. However, how to control LED’s brightness?

It makes sense to take advantage of PWM. Output the number of high level and low level in unit time, the more time the high level occupies, the larger the PWM value, the brighter the LED.



We provide the PWM output library file < analogwrite.h > for ESP32, therefore solely a simple statement analogWrite(); can control the PWM output.

1. **Test Code**

|  |
| --- |
| #include <analogWrite.h> //Import PWM output library files  #define led\_y 12 //Define LED pins  void setup(){  pinMode(led\_y, OUTPUT); //Set pin to output mode  }  void loop(){  for(int i=0; i<255; i++) //The for loop statement increments the value of variable i until it exits the loop at 255  {  analogWrite(led\_y, i); //PWM output, control LED brightness  delay(3);  }  for(int i=255; i>0; i--) //The for loop statement continues to decrease the value of variable i until it exits the loop at 0  analogWrite(led\_y, i);  delay(3);  }  } |

1. **Test Result**

The LED gradually gets dimmer then brighter, cyclically, like human breathe.

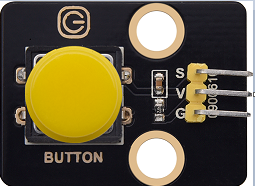
## Project 2: [Table](C:/Users/NINGMEI/AppData/Local/youdao/dict/Application/8.10.7.0/resultui/html/index.html" \l "/javascript:;) [Lamp](C:/Users/NINGMEI/AppData/Local/youdao/dict/Application/8.10.7.0/resultui/html/index.html" \l "/javascript:;)

1. **Description**

The common table lamp uses LED lights and buttons, which can control the light on and off pressing the button.

1. **Button Principle**

The button module is a digital sensor, which can only read 0 or 1. When the module is not pressed, it is in a high level state, that is, 1, when pressed, it is a low level 0.



1. **Pins of the Button**

|  |  |
| --- | --- |
| Button 1 | 16 |
| Button 2 | 27 |

### Project 2.1 Read the Button

**1. Description**

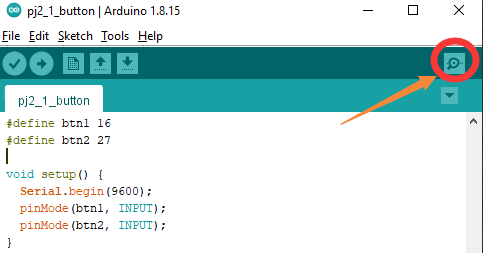
We will work to read the status value of the button and display it on the serial monitor, so as to see it intuitively.

1. **Test Code**

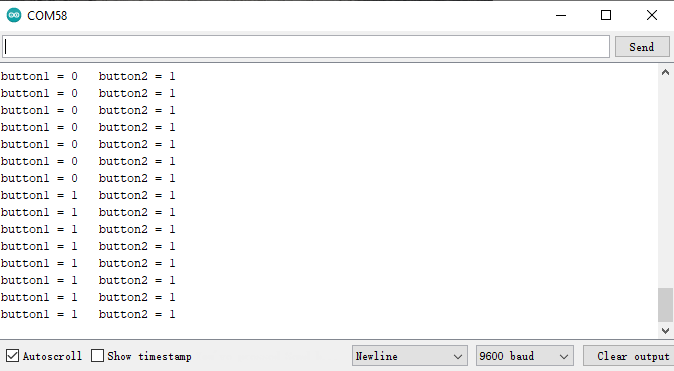
|  |
| --- |
| #define btn1 16  #define btn2 27  void setup() {  Serial.begin(9600);  pinMode(btn1, INPUT);  pinMode(btn2, INPUT);  }  void loop() {  boolean btn1\_val = digitalRead(btn1);  boolean btn2\_val = digitalRead(btn2);  Serial.print("button1 = ");  Serial.print(btn1\_val);  Serial.print(" ");  Serial.print("button2 = ");  Serial.println(btn2\_val);  delay(100);  } |

1. **Test Result**

Open the serial monitor of the arduino IDE



Press the button again to see the change of the button state value, as shown below:



### Project 2.2. Table Lamp

**1. Description**

For common simple table lamp, click the button it will be opened, click it again, the lamp will be closed.

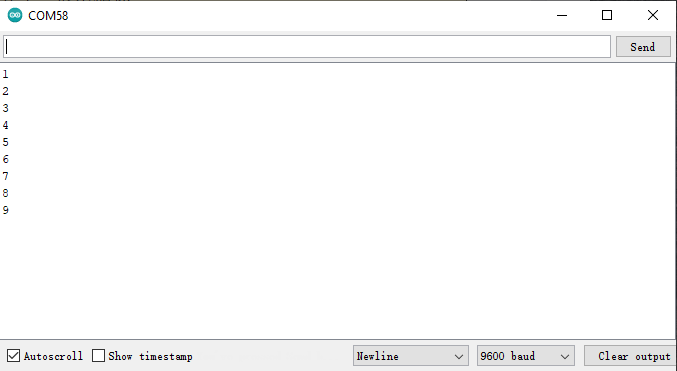
1. **Test Code**

Calculate the clicked button times and take the remainder of 2, you can get 0 or 1 two state values.

|  |
| --- |
| #define btn1 16  #define led\_y 12  int btn\_count = 0; //Used to count the clicked button times  void setup() {  Serial.begin(9600);  pinMode(btn1, INPUT);  pinMode(led\_y, OUTPUT);  }  void loop() {  boolean btn1\_val = digitalRead(btn1);  if(btn1\_val == 0) //If the button is pressed  {  delay(10); //Delay 10ms to eliminate button jitter  if(btn1\_val == 0) //Make sure the button is pressed again  {  boolean btn\_state = 1;  while(btn\_state == 1) //Loop indefinitely until the button is released  {  boolean btn\_val = digitalRead(btn1);  if(btn\_val == 1) //If the button is released  {  btn\_count++; //Automatically increments by 1, account the clicked button times  Serial.println(btn\_count);  btn\_state = 0; //The button is released and exits the loop  }  }  }  boolean value = btn\_count % 2; //Take the remainder of the value, you will get 0 or 1  if(value == 1)  {  digitalWrite(led\_y, HIGH);  }  else{  digitalWrite(led\_y, LOW);  }  }  } |

1. **Test Result**

Open the serial monitor and print out the clicked button times, then click the button once, the LED will be on, click it again, it will be off.



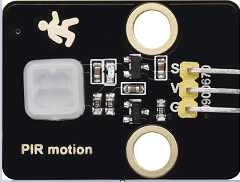
## Project 3: PIR Motion Sensor

**1. Description**

The PIR motion sensor has many application scenarios in daily life, such as automatic induction lamp of stairs, automatic induction faucet of washbasin, etc.

It is also a digital sensor like buttons, which has two state

values 0 or 1.  And it will be sensed when people are moving.



1. **Control Pin**

|  |  |
| --- | --- |
| PIR motion sensor | 14 |

### Project 3.1 Read the PIR Motion Sensor

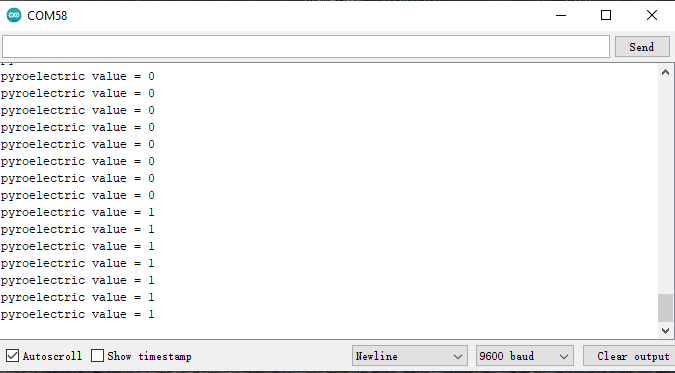
We will print out the value of the PIR motion sensor through the serial monitor.

1. **Test Code**

|  |
| --- |
| #define pyroelectric 14  void setup() {  Serial.begin(9600);  pinMode(pyroelectric, INPUT);  }  void loop() {  boolean pyroelectric\_val = digitalRead(pyroelectric);  Serial.print("pyroelectric value = ");  Serial.println(pyroelectric\_val);  delay(200);  } |

1. **Test Result**

When you stand still in front of the sensor, the reading value is 0, move a little, it will change to 1.



### Project 3.2 PIR Motion Sensor

If someone moves in front of the sensor, the LED will light up.

1. **Test Code**

|  |
| --- |
| #define pyroelectric 14  #define led\_y 12 //Define the yellow led pin to 12  void setup() {  Serial.begin(9600);  pinMode(pyroelectric, INPUT);  pinMode(led\_y, OUTPUT); //Set pin to output mode  }  void loop() {  boolean pyroelectric\_val = digitalRead(pyroelectric);  Serial.print("pyroelectric value = ");  Serial.println(pyroelectric\_val);  delay(200);  if(pyroelectric\_val == 1)  {  digitalWrite(led\_y, HIGH);  }else{  digitalWrite(led\_y, LOW);  }  } |

1. **Test Result**

Move your hand in front of the sensor,  the LED will turn on. After 5s of immobility, the LED lights will turn off.

## Project 4: Play Music

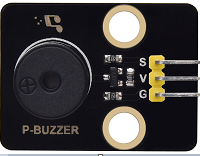
**1. Description**

There is a audio power amplifier element in the car expansion board, which is as an external amplification equipment to play music.

In this project, we will work to play a  piece of music by using it.

**2. Component Knowledge**

**Passive Buzzer:** The audio power amplifier (like the passive buzzer) does not have internal oscillation. When controlling, we need to input square waves of different frequencies to the positive pole of the component and ground the negative pole to control the power amplifier to chime sounds of different frequencies.



1. **Control Pin**

|  |  |
| --- | --- |
| Passive Buzzer | 25 |

### Project 4.1 Play Happy Birthday

**1. Test Code**

|  |
| --- |
| #include <ESP32Tone.h>  #define buzzer\_pin 25  void setup() {  pinMode(buzzer\_pin, OUTPUT);  birthday();  }  void loop() {    }  void birthday()  {  tone(buzzer\_pin,294,250,0); //The four parameters are pin, frequency, delay and channel  tone(buzzer\_pin,440,250,0);  tone(buzzer\_pin,392,250,0);  tone(buzzer\_pin,532,250,0);  tone(buzzer\_pin,494,250,0);  tone(buzzer\_pin,392,250,0);  tone(buzzer\_pin,440,250,0);  tone(buzzer\_pin,392,250,0);  tone(buzzer\_pin,587,250,0);  tone(buzzer\_pin,532,250,0);  tone(buzzer\_pin,392,250,0);  tone(buzzer\_pin,784,250,0);  tone(buzzer\_pin,659,250,0);  tone(buzzer\_pin,532,250,0);  tone(buzzer\_pin,494,250,0);  tone(buzzer\_pin,440,250,0);  tone(buzzer\_pin,698,250,0);  tone(buzzer\_pin,659,250,0);  tone(buzzer\_pin,532,250,0);  tone(buzzer\_pin,587,250,0);  tone(buzzer\_pin,532,500,0);  noTone(buzzer\_pin,0); //Close  } |

**2. Test Result**

The passive buzzer will play happy Birthday.

### Project 4.2 Music Box

we will make a music box and switch tunes by pressing buttons.

**1. Test Code**

|  |
| --- |
| #include <ESP32Tone.h>  #include <musicESP32\_home.h>  music Music(25);  #define btn1 16  int btn\_count = 0; //Used to count the clicked button times  boolean music\_flag = 0;  void setup() {  Serial.begin(9600);  pinMode(btn1, INPUT);  pinMode(25, OUTPUT);  // Music.tetris();  // Music.birthday();  // Music.Ode\_to\_Joy();  // Music.christmas();  // Music.super\_mario();  // Music.star\_war\_tone();  }  void loop() {  boolean btn1\_val = digitalRead(btn1);  if(btn1\_val == 0) //If the button is pressed  {  delay(10); //Delay 10ms to eliminate button jitter  if(btn1\_val == 0) //Make sure the button is pressed again  {  boolean btn\_state = 1;  while(btn\_state == 1) //Loop indefinitely until the button is released  {  boolean btn\_val = digitalRead(btn1);  if(btn\_val == 1) //If the button is released  {  music\_flag = 1;  btn\_count++; //Automatically increments by 1 to count the number of times the button is clicked  Serial.println(btn\_count);  if(btn\_count == 4)  {  btn\_count = 1;  }  switch(btn\_count)  {  case 1: if(music\_flag == 1){Music.Ode\_to\_Joy();music\_flag=0;} break;  case 2: if(music\_flag == 1){Music.christmas();music\_flag=0;} break;  case 3: if(music\_flag == 1){Music.tetris();music\_flag=0;} break;  }  btn\_state = 0; //The button is released and exits the loop  }  }  }  }  } |

**2. Test Result**

Click button 1 once, it will play a tetris, then click it again, it will play Ode\_to\_Joy, after playing, click the button 1 for the third time, it will play Christmas.

## Project 5: Automatic Doors and Windows

1. **Description**

Automatic doors and windows need power device, which will become more automatic with a 180 degree servo and some sensors. Adding a raindrop sensor, you can achieve the effect of closing windows automatically when raining. If adding a RFID, we can realize the effect of swiping to open the door and so on.

1. **Component Knowledge**

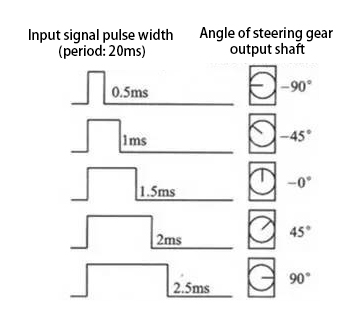
**Servo:** Servo is a position servo [driver](C:/Users/NINGMEI/AppData/Local/youdao/dict/Application/8.10.7.0/resultui/html/index.html" \l "/javascript:;) device consists of a housing, a circuit board, a coreless motor, a gear and a position detector.

Its working principle is that the servo receives the signal sent by MCU or receiver and produces a reference signal with a period of 20ms and width of 1.5ms, then compares the acquired DC bias voltage to the voltage of the potentiometer and obtain the voltage difference output.

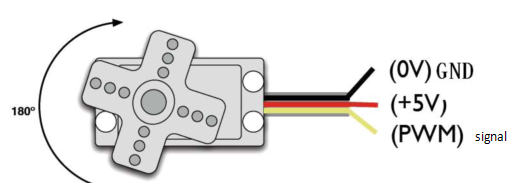
The IC on the circuit board judges the direction of rotation, and then drives the coreless motor to start rotation. The power is transmitted to the swing arm through the reduction gear, and the signal is sent back by the position detector to judge whether the positioning has been reached, which is suitable for those control systems that require constant angle change and can be maintained.

When the motor speed is constant, the potentiometer is driven to rotate through the cascade reduction gear, which leads that the voltage difference is 0, and the motor stops rotating. Generally, the angle range of servo rotation is 0° --180 °.

The pulse period of the control servo is 20ms, the pulse width is 0.5ms ~ 2.5ms, and the corresponding position is -90°~ +90°.  Here is an example of a 180° servo:



In general, servo has three lines in brown, red and orange. The brown wire is grounded, the red one is a positive pole line and the orange one is a signal line.





1. **Pin**

|  |  |
| --- | --- |
| The servo of the window | 5 |
| The servo of the door | 13 |

### Project 5.1 Control the Door

**1. Test Code**

|  |
| --- |
| #include <ESP32\_Servo.h>  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 = 13;    void setup() {  myservo.attach(servoPin); // attaches the servo on pin 18 to the servo object  }  void loop() {  for (pos = 0; pos <= 180; pos += 1) { // goes from 0 degrees to 180 degrees  // in steps of 1 degree  myservo.write(pos); // tell servo to go to position in variable 'pos'  delay(15); // waits 15ms for the servo to reach the position  }  for (pos = 180; pos >= 0; pos -= 1) { // goes from 180 degrees to 0 degrees  myservo.write(pos); // tell servo to go to position in variable 'pos'  delay(15); // waits 15ms for the servo to reach the position  }  } |

**2. Test Result**

The servo of the door turns with the door, back and forth

### Project 5.2 Close the Window

1. **Description**

We will work to use a servo and a raindrop sensor to make an device closing windows automatically when raining.

1. **Component Knowledge**

**Raindrop Sensor:** This is an analog input module, the greater the area covered by water on the detection surface, the greater the value returned (range 0~4096).

1. **Test Code**

|  |
| --- |
| #include <ESP32\_Servo.h>  Servo myservo;  #define servoPin 5  #define waterPin 34  void setup() {  Serial.begin(9600);  pinMode(waterPin, INPUT);  myservo.attach(servoPin);  myservo.write(176);  delay(200);  }  void loop() {  int water\_val = analogRead(waterPin);  Serial.println(water\_val);  if(water\_val > 1500) {  myservo.write(0);  delay(200);  }  else {  myservo.write(176);  delay(200);  }  } |

1. **Test Result**

At first, the window opens automatically, and when you touch the raindrop sensor with your hand (which has water on the skin), the window will close.

## Project 6: Atmosphere Lamp

1. **Description**

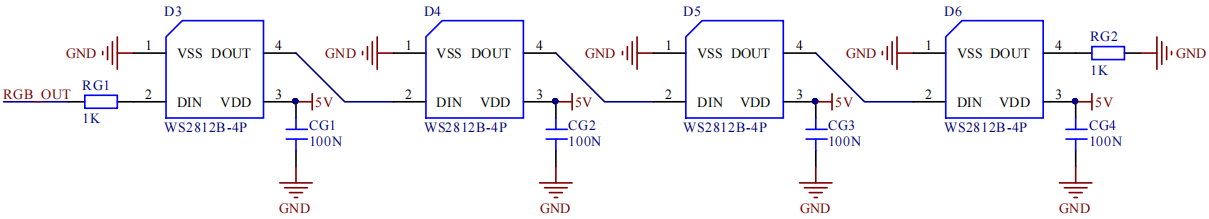
The atmosphere lamp of smart home is 4 SK6812RGB LEDs. RGB LED belongs to a simple luminous module, which can adjust the color to bring out the lamp effect of different colors. Furthermore, it can be widely used in buildings, bridges, roads, gardens, courtyards, floors and other fields of decorative lighting and venue layout, Christmas, Halloween, Valentine's Day, Easter, National Day as well as other festivals during the atmosphere and other scenes.

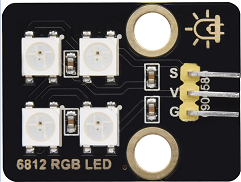
In this experiment, we will make various lighting effects.

1. **Component Knowledge**

From the schematic diagram, we can see that these four RGB LEDs are all connected in series. In fact, no matter how many they are, we can use a pin to control a RGB LED and let it display any color. Each RGBLED is an independent pixel, composed of R, G and B colors, which can achieve 256 levels of brightness display and complete the full true color display of 16777216 colors.

What’s more, the pixel point contains a data latch signal shaping amplifier drive circuit and a signal shaping circuit, which effectively ensures the color of the pixel point light is highly consistent.





1. **Pin**

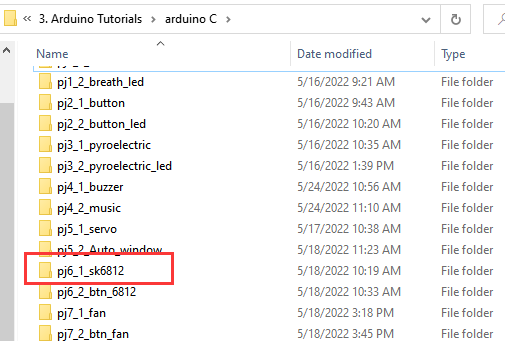
|  |  |
| --- | --- |
| SK6812 | 26 |

### Project 6.1 Control SK6812

We will control SK6812 to display various lighting effects.

**1. Test Code**

Please open the provided test code pj6\_1\_SK6812, as shown in the image below:



**2. Test Result**

The atmosphere lamps of the smart home will display a variety of colors and light effects.

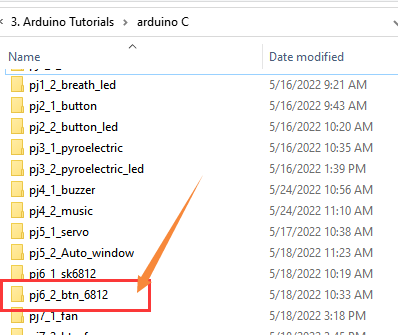
### Project 6.2 Button

1. **Description**

There are two buttons to switch the color of the atmosphere lamp.

1. **Test Code**

Please open the provided test code pj6\_2\_btn\_6812, as shown below:



1. **Test Result**

We can switch the color of the atmosphere lamp by clicking buttons 1 and 2.

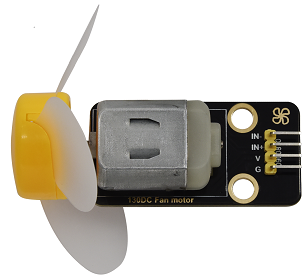
## Project 7: Fan

1. **Description**

In this project, we will learn how to make a small fan.

1. **Component Knowledge**

The small fan uses a 130 DC motor and safe fan blades.  You can use PWM output to control the fan speed.



1. **Control Method**

Two pins are required to control the motor of the fan, one for INA and two for INB.  The PWM value range is 0~255. When the PWM output of the two pins is different, the fan can rotate.

|  |  |
| --- | --- |
| INA - INB <= -45 | Rotate clockwise |
| INA - INB >= 45 | Rotate [anticlockwise](C:/Users/NINGMEI/AppData/Local/youdao/dict/Application/8.10.7.0/resultui/html/index.html" \l "/javascript:;) |
| INA == 0, INB == 0 | Stop |

1. **Control Pins**

|  |  |
| --- | --- |
| INA | 19 |
| INB | 18 |

### Project 7.1 Control the Fan

We can control the [anticlockwise](C:/Users/NINGMEI/AppData/Local/youdao/dict/Application/8.10.7.0/resultui/html/index.html" \l "/javascript:;) and clockwise rotation speed of the fan.

**1. Test Code**

|  |
| --- |
| #include <analogWrite.h>  #define fanPin1 19  #define fanPin2 18  void setup() {  pinMode(fanPin1, OUTPUT);  pinMode(fanPin2, OUTPUT);  }  void loop() {  digitalWrite(fanPin1, LOW); //pwm = 0  analogWrite(fanPin2, 180);  delay(3000);  digitalWrite(fanPin1, LOW);  digitalWrite(fanPin2, LOW);  delay(1000);  digitalWrite(fanPin1, HIGH); //pwm = 255  analogWrite(fanPin2, 210);  delay(3000);  digitalWrite(fanPin1, LOW);  digitalWrite(fanPin2, LOW);  delay(1000);  } |

**2. Test Result**

The fan will rotate clockwise and [anticlockwise](C:/Users/NINGMEI/AppData/Local/youdao/dict/Application/8.10.7.0/resultui/html/index.html" \l "/javascript:;) at different

speeds.

### Project 7.2 Switch On or Off the Fan

One button switches the fan on and the other button controls the speed of the fan.

**1. Test Code**

|  |
| --- |
| #include <analogWrite.h>  #define fanPin1 19  #define fanPin2 18  #define btn1 16  int btn\_count = 0; //Used to count the clicked button times  #define btn2 27  int btn\_count2 = 0;  int speed\_val = 130; //Define the speed variables  void setup() {  Serial.begin(9600);  pinMode(btn1, INPUT);  pinMode(btn2, INPUT);  pinMode(fanPin1, OUTPUT);  pinMode(fanPin2, OUTPUT);  }  void loop() {  boolean btn1\_val = digitalRead(btn1);  if(btn1\_val == 0) //If the button is pressed  {  delay(10); //Delay 10ms to eliminate button jitter  if(btn1\_val == 0) //Make sure the button is pressed again  {  boolean btn\_state = 1;  while(btn\_state == 1) //Loop indefinitely until the button is released  {  boolean btn\_val = digitalRead(btn1);  if(btn\_val == 1) //If the button is released  btn\_count++; //Automatically increments by 1 to count the clicked button times  Serial.println(btn\_count);  btn\_state = 0; //The button is released and exits the loop  }  }  }  boolean value = btn\_count % 2; //Take the remainder of the value, you will get 0 or 1  while(value == 1)  {  //Serial.println("on");  digitalWrite(fanPin1, LOW); //pwm = 0  analogWrite(fanPin2, speed\_val);    boolean btn2\_val = digitalRead(btn2);  if(btn2\_val == 0)  {  delay(10);  if(btn2\_val == 0)  {  boolean btn\_state2 = 1;  while(btn\_state2 == 1)  {  boolean btn2\_val = digitalRead(btn2);  if(btn2\_val == 1)  {  btn\_count2++;  if(btn\_count2 > 3)  {  btn\_count2 = 1;  }  switch(btn\_count2)  {  case 1: speed\_val = 130; Serial.println(speed\_val);break; //Adjust the speed  case 2: speed\_val = 180; Serial.println(speed\_val);break;  case 3: speed\_val = 230; Serial.println(speed\_val);break;  }  btn\_state2 = 0;  }  }  }  }  boolean btn1\_val = digitalRead(btn1);  if(btn1\_val == 0) //If the button is pressed  {  digitalWrite(fanPin1, LOW); //pwm = 0  analogWrite(fanPin2, 0);  value = 0; //Exit the loop  }    }  }  } |

**2. Test Result**

Click button 1, the fan starts to rotate, click button 2, the

speed can be adjusted(there are three different speeds), press the button 1 again, the fan stops.

## Project 8: LCD1602 Display

1. **Description**

As we all know, screen is one of the best ways for people to interact with electronic devices.

1. **Component Knowledge**

1602 is a line that can display 16 characters. There are two lines, which use IIC communication protocol.



1. **Control Pins**

|  |  |
| --- | --- |
| SDA | SDA |
| SCL | SCL |

### Project 8.1 Display [Character](C:/Users/NINGMEI/AppData/Local/youdao/dict/Application/8.10.7.0/resultui/html/index.html" \l "/javascript:;)s

**1. Test Code**

|  |
| --- |
| #include <Wire.h>  #include <LiquidCrystal\_I2C.h>  LiquidCrystal\_I2C mylcd(0x27,16,2);  void setup(){  mylcd.init();  mylcd.backlight();  }  void loop(){  mylcd.setCursor(0, 0);  mylcd.print("hello");  mylcd.setCursor(0, 1);  mylcd.print("keyestudio");  //mylcd.clear();  } |

**2. Test Result**

The first line of the LCD1602 shows hello and the second line shows keyestudio.

### Project 8.2 Dangerous Gas Alarm

**1. Description**

When a gas sensor detects a high concentration of dangerous gas, the buzzer will sound an alarm and the display will show dangerous.

**2. Component Knowledge**

**MQ2 Smoke Sensor**: It is a gas leak monitoring device for homes and factories, which is suitable for liquefied gas, benzene, alkyl, alcohol, hydrogen as well as smoke detection.  Our sensor leads to digital pin D and analog output pin A, which is connected to D as a digital sensor in this project .



**3. Test Code**

|  |
| --- |
| #include <Wire.h>  #include <LiquidCrystal\_I2C.h>  LiquidCrystal\_I2C mylcd(0x27,16,2);  #define gasPin 23  #define buzPin 25  boolean i = 1;  boolean j = 1;  void setup(){  Serial.begin(9600);  mylcd.init();  mylcd.backlight();  pinMode(buzPin, OUTPUT);  pinMode(gasPin, INPUT);  mylcd.setCursor(0, 0);  mylcd.print("safety");  }  void loop(){  boolean gasVal = digitalRead(gasPin); //Reads the value detected by the gas sensor  Serial.println(gasVal);  if(gasVal == 0) //If the hazardous gas is detected，LCD displays dangerous，the buzzer makes an alarm  {  while(i == 1)  {  mylcd.clear();  mylcd.setCursor(0, 0);  mylcd.print("dangerous");  i = 0;  j = 1;  }  digitalWrite(buzPin,HIGH);  delay(1);  digitalWrite(buzPin,LOW);  delay(1);  }  else{  digitalWrite(buzPin,LOW);  while(j == 1)  {  mylcd.clear();  mylcd.setCursor(0, 0);  mylcd.print("safety");  i = 1;  j = 0;  }  }  } |

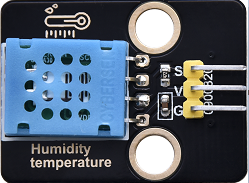
1. **Test Result**

The screen displays "safety" in normal state. However, when the gas sensor detects some dangerous gases, such as carbon monoxide, at a certain concentration, the buzzer will sound an alarm and the screen displays "dangerous".

## Project 9: Temperature and Humidity Sensor

1. **Component Knowledge**

Its communication mode is serial data and single bus. The temperature measurement range is -20 ~ +60℃, accuracy is ±2℃. However, the humidity range is 5 ~ 95%RH, the accuracy is ±5%RH.



1. **Control Pin**

|  |  |
| --- | --- |
| Temperature and Humidity Sensor | 17 |

### Project 9.1 Temperature and Humidity Tester

**1. Test Code**

|  |
| --- |
| //\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  /\*  \* Filename : xht11  \* Description : Read the temperature and humidity values of XHT11.  \* Auther : http//www.keyestudio.coml  \*/  #include <Wire.h>  #include <LiquidCrystal\_I2C.h>  LiquidCrystal\_I2C mylcd(0x27,16,2);  #include "xht11.h"  xht11 xht(17);  unsigned char dht[4] = {0, 0, 0, 0};//Only the first 32 bits of data are received, not the parity bits  void setup() {  Serial.begin(9600);//Start the serial port monitor and set baud rate to 9600  mylcd.init();  mylcd.backlight();  }  void loop() {  if (xht.receive(dht)) { //Returns true when checked correctly  Serial.print("RH:");  Serial.print(dht[0]); //The integral part of humidity, DHT [1] is the fractional part  Serial.print("% ");  Serial.print("Temp:");  Serial.print(dht[2]); //The integral part of temperature, DHT [3] is the fractional part  Serial.println("C");    mylcd.setCursor(0, 0);  mylcd.print("T = ");  mylcd.print(dht[2]);  mylcd.setCursor(0, 1);  mylcd.print("H = ");  mylcd.print(dht[0]);  //mylcd.clear();  delay(200);  } else { //Read error  Serial.println("sensor error");  }  delay(1000); //It takes 1000ms to wait for the device to read  }  //\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* |

**2. Test Result**

The LCD1602 displays the temperature (T = \*\* ° C) and humidity (H = \*\* %RH). When you breathe into the T/H sensor, you can see that the humidity rises.

## Project 10: RFID RC522 Module

1. **Component Knowledge**

Radio frequency identification, the card reader is composed of a radio frequency module and a high-level magnetic field. The Tag transponder is a sensing device, which doesn’t contain a battery. It only contains tiny integrated circuit chips and media for storing data and antennas for receiving and transmitting signals.

To read the data in the tag, first put it into the reading range of the card reader. The reader will generate a magnetic field, which can produce electricity according to Lenz's law, then the RFID tag will supply power, thereby activating the device.



1. **Control Pins**

Use IIC communication

|  |  |
| --- | --- |
| SDA | SDA |
| SCL | SCL |

### Project 10.1 Open the Door

**1. Test Code**

|  |
| --- |
| //\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  /\*  \* Filename : RFID  \* Description : RFID reader UID  \* Auther : http//www.keyestudio.com  \*/  #include <Wire.h>  #include <LiquidCrystal\_I2C.h>  LiquidCrystal\_I2C mylcd(0x27,16,2);  #include <ESP32\_Servo.h>  Servo myservo;  #include <Wire.h>  #include "MFRC522\_I2C.h"  // IIC pins default to GPIO21 and GPIO22 of ESP32  // 0x28 is the i2c address of SDA, if doesn't match，please check your address with i2c.  MFRC522 mfrc522(0x28); // create MFRC522.  #define servoPin 13  #define btnPin 16  boolean btnFlag = 0;  String password = "";  void setup() {  Serial.begin(115200); // initialize and PC's serial communication  mylcd.init();  mylcd.backlight();  Wire.begin(); // initialize I2C  mfrc522.PCD\_Init(); // initialize MFRC522  ShowReaderDetails(); // display PCD - MFRC522 read carder  Serial.println(F("Scan PICC to see UID, type, and data blocks..."));  myservo.attach(servoPin);  pinMode(btnPin, INPUT);  mylcd.setCursor(0, 0);  mylcd.print("Card");  }  void loop() {  //  if ( ! mfrc522.PICC\_IsNewCardPresent() || ! mfrc522.PICC\_ReadCardSerial() ) {  delay(50);  password = "";  if(btnFlag == 1)  {  boolean btnVal = digitalRead(btnPin);  if(btnVal == 0) //Swipe the card to open the door and click button 1 to close the door  {  Serial.println("close");  mylcd.setCursor(0, 0);  mylcd.print("close");  myservo.write(0);  btnFlag = 0;  }  }  return;  }    // select one of door cards. UID and SAK are mfrc522.uid.    // save UID  Serial.print(F("Card UID:"));  for (byte i = 0; i < mfrc522.uid.size; i++) {  Serial.print(mfrc522.uid.uidByte[i] < 0x10 ? " 0" : " ");  //Serial.print(mfrc522.uid.uidByte[i], HEX);  Serial.print(mfrc522.uid.uidByte[i]);  password = password + String(mfrc522.uid.uidByte[i]);  }  if(password == "17121741227") //The card number is correct, open the door  {  Serial.println("open");  mylcd.setCursor(0, 0);  mylcd.clear();  mylcd.print("open");  myservo.write(180);  password = "";  btnFlag = 1;  }  else //The card number is wrong，LCD displays error  {  password = "";  mylcd.setCursor(0, 0);  mylcd.print("error");  }  //Serial.println(password);  }  void ShowReaderDetails() {  // attain the MFRC522 software  byte v = mfrc522.PCD\_ReadRegister(mfrc522.VersionReg);  Serial.print(F("MFRC522 Software Version: 0x"));  Serial.print(v, HEX);  if (v == 0x91)  Serial.print(F(" = v1.0"));  else if (v == 0x92)  Serial.print(F(" = v2.0"));  else  Serial.print(F(" (unknown)"));  Serial.println("");  // when returning to 0x00 or 0xFF, may fail to transmit communication signals  if ((v == 0x00) || (v == 0xFF)) {  Serial.println(F("WARNING: Communication failure, is the MFRC522 properly connected?"));  }  }  //\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* |

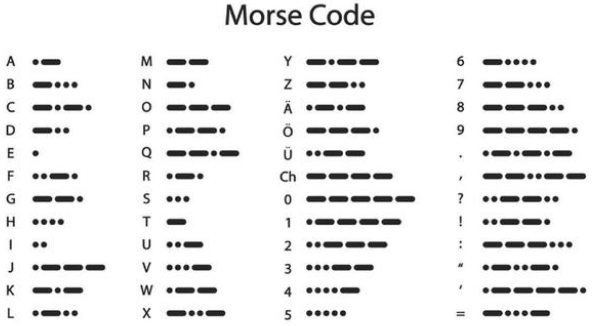
**2. Test Result**

Close the provided card to the RFID induction area, the door will turn and open, and LCD1602 shows "The door is open". Click button 1 and the door turns and closes. However, when swiping another blue induction block, the LCD1602 shows "Error".

## Project 11: [Morse](C:/Users/NINGMEI/AppData/Local/youdao/dict/Application/8.10.7.0/resultui/html/index.html" \l "/javascript:;) [Code](C:/Users/NINGMEI/AppData/Local/youdao/dict/Application/8.10.7.0/resultui/html/index.html" \l "/javascript:;)

Morse code, also known as Morse password, is an on-again, off-again signal code that expresses different letters, numbers, and punctuation marks in different sequences. Now we use it as our password gate.

The Morse code corresponds to the following characters:



### Project 11.1 Morse Code Open the Door

**1. Description**

We use as the correct password. What’s more, there is a button library file OneButton, which is very simple to click, double click, long press and other functions. For Morse password, click is“.”, long press and release is “-”.

**2. Test Code**

|  |
| --- |
| #include <Wire.h>  #include <LiquidCrystal\_I2C.h>  LiquidCrystal\_I2C mylcd(0x27,16,2);  #include "OneButton.h"  // Setup a new OneButton on pin 16.  OneButton button1(16, true);  // Setup a new OneButton on pin 27.  OneButton button2(27, true);  #include <ESP32\_Servo.h>  Servo myservo;  int servoPin = 13;  String password = "";  String correct\_p = "-.-"; //The correct password for the password door  // setup code here, to run once:  void setup() {  Serial.begin(115200);  mylcd.init();  mylcd.backlight();  // link the button 1 functions.  button1.attachClick(click1);  button1.attachLongPressStop(longPressStop1);  // link the button 2 functions.  button2.attachClick(click2);  button2.attachLongPressStop(longPressStop2);  myservo.attach(servoPin);  mylcd.setCursor(0, 0);  mylcd.print("Enter password");  }  void loop() {  // keep watching the push buttons:  button1.tick();  button2.tick();  delay(10);  }  // ----- button 1 callback functions  // This function will be called when the button1 was pressed 1 time (and no 2. button press followed).  void click1() {  Serial.print(".");  password = password + '.';  mylcd.setCursor(0, 1);  mylcd.print(password);  } // click1  // This function will be called once, when the button1 is released after being pressed for a long time.  void longPressStop1() {  Serial.print("-");  password = password + '-';  mylcd.setCursor(0, 1);  mylcd.print(password);  } // longPressStop1  // ... and the same for button 2:  void click2() {  Serial.println(password);  if(password == correct\_p)  {  myservo.write(180); //Open the door if the password is correct  mylcd.clear();  mylcd.setCursor(0, 0);  mylcd.print("open");  }  else  {  mylcd.clear();  mylcd.setCursor(0, 0);  mylcd.print("error");  delay(2000);  mylcd.clear();  mylcd.setCursor(0, 0);  mylcd.print("input again");  }  password = "";  } // click2  void longPressStop2() {  //Serial.println("Button 2 longPress stop");  myservo.write(0); //Close the door  mylcd.clear();  mylcd.setCursor(0, 0);  mylcd.print("close");  } // longPressStop2 |

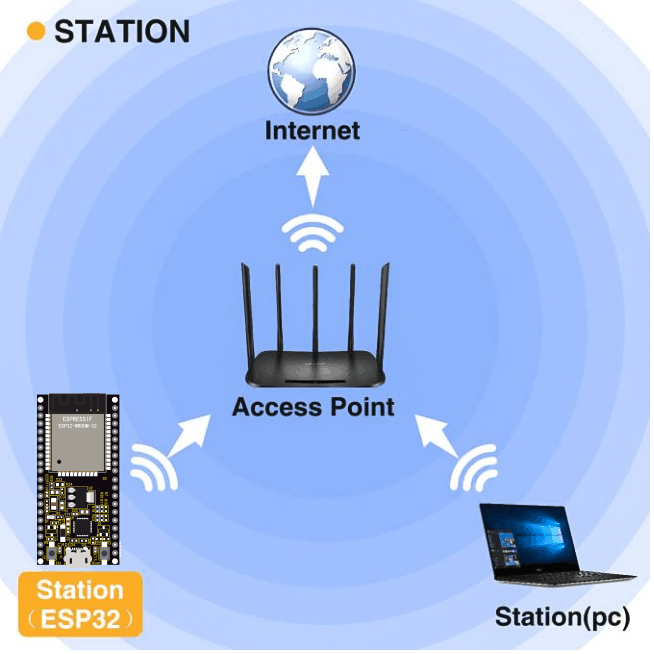
1. **Test Result**

At first, the LCD1602 displays "Enter password", then click or long press button 1 to tap the password. If we input the correct password "-.-", then click button 2, the door will open, and the LCD1602 will display "open".

If other incorrect passwords are entered, the door will not move, the LCD1602 will display “error” and then “enter again” 2s later. Furthermore, long press button 2 can close the door.

## Project 12: WiFi

The easiest way to access the Internet is to use a WiFi to connect. The ESP32 main control board comes with a WiFi module, making our smart home accessible to the Internet easily.



### Project 12.1 Smart Home

**1. Description**

We connect the smart home to a LAN, which is the WiFi in your home or the hot spot of your phone. After the connection is successful, an address will be assigned, which can be used for communication. We will print the assigned address in the serial monitor.

**2. Test Code**

Note: ssiD and password in the code should be filled with your own WiFi name and password.



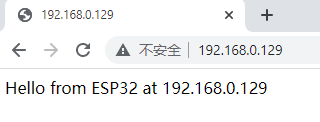
|  |
| --- |
| #include <Arduino.h>  #include <WiFi.h>  #include <ESPmDNS.h>  #include <WiFiClient.h>  String item = "0";  const char\* ssid = "ChinaNet-2.4G-0DF0";  const char\* password = "ChinaNet@233";  WiFiServer server(80);  void setup() {  Serial.begin(115200);  WiFi.begin(ssid, password);  while (WiFi.status() != WL\_CONNECTED) {  delay(500);  Serial.print(".");  }  Serial.println("");  Serial.print("Connected to ");  Serial.println(ssid);  Serial.print("IP address: ");  Serial.println(WiFi.localIP());  server.begin();  Serial.println("TCP server started");  MDNS.addService("http", "tcp", 80);  }  void loop() {  Serial.print("Connected to ");  Serial.println(ssid);  Serial.print("IP address: ");  Serial.println(WiFi.localIP()); //The assigned IP address is printed on the serial monitor  delay(200);  WiFiClient client = server.available();  if (!client) {  return;  }  while(client.connected() && !client.available()){  delay(1);  }  String req = client.readStringUntil('\r');  int addr\_start = req.indexOf(' ');  int addr\_end = req.indexOf(' ', addr\_start + 1);  if (addr\_start == -1 || addr\_end == -1) {  Serial.print("Invalid request: ");  Serial.println(req);  return;  }  req = req.substring(addr\_start + 1, addr\_end);  item=req;  Serial.println(item);  String s;  if (req == "/") //Browser accesses address can read the information sent by the client.println(s);  {  IPAddress ip = WiFi.localIP();  String ipStr = String(ip[0]) + '.' + String(ip[1]) + '.' + String(ip[2]) + '.' + String(ip[3]);  s = "HTTP/1.1 200 OK\r\nContent-Type: text/html\r\n\r\n<!DOCTYPE HTML>\r\n<html>Hello from ESP32 at ";  s += ipStr;  s += "</html>\r\n\r\n";  Serial.println("Sending 200");  client.println(s); //Send the string S, you can read the information when visiting the address of E smart home using a browser.  //client.print(s);  client.stop();  } |

**3. Test Result**

If the WiFi is connected successfully, the serial monitor will print out the assigned IP address.



Open a browser to access the IP address, then we will read the contents of the string S sent out by the client.println(s); in the code.



### Project 12.2 Control Smart Home

**1. Description**

In this project, we will learn how to realize different functions of the smart home through accessing different strings under the address. There is a LCD screen that can print out the IP address, which is much more convenient.

**2. Test Code**

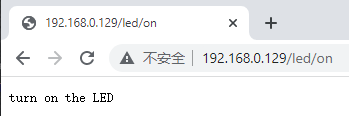
|  |
| --- |
| #include <Arduino.h>  #include <WiFi.h>  #include <ESPmDNS.h>  #include <WiFiClient.h>  String item = "0";  const char\* ssid = "ChinaNet-2.4G-0DF0";  const char\* password = "ChinaNet@233";  WiFiServer server(80);  #include <Wire.h>  #include <LiquidCrystal\_I2C.h>  LiquidCrystal\_I2C mylcd(0x27,16,2);  #include <analogWrite.h>  #define fanPin1 19  #define fanPin2 18  #define led\_y 12 //Define the yellow led pin to 12  void setup() {  Serial.begin(115200);  mylcd.init();  mylcd.backlight();  pinMode(led\_y, OUTPUT);  pinMode(fanPin1, OUTPUT);  pinMode(fanPin2, OUTPUT);    WiFi.begin(ssid, password);  while (WiFi.status() != WL\_CONNECTED) {  delay(500);  Serial.print(".");  }  Serial.println("");  Serial.print("Connected to ");  Serial.println(ssid);  Serial.print("IP address: ");  Serial.println(WiFi.localIP());  server.begin();  Serial.println("TCP server started");  MDNS.addService("http", "tcp", 80);  mylcd.setCursor(0, 0);  mylcd.print("ip:");  mylcd.setCursor(0, 1);  mylcd.print(WiFi.localIP()); //LCD displays the ip adress  }  void loop() {  WiFiClient client = server.available();  if (!client) {  return;  }  while(client.connected() && !client.available()){  delay(1);  }  String req = client.readStringUntil('\r');  int addr\_start = req.indexOf(' ');  int addr\_end = req.indexOf(' ', addr\_start + 1);  if (addr\_start == -1 || addr\_end == -1) {  Serial.print("Invalid request: ");  Serial.println(req);  return;  }  req = req.substring(addr\_start + 1, addr\_end);  item=req;  Serial.println(item);  String s;  if (req == "/") //Browser accesses address can read the information sent by the client.println(s);  {  IPAddress ip = WiFi.localIP();  String ipStr = String(ip[0]) + '.' + String(ip[1]) + '.' + String(ip[2]) + '.' + String(ip[3]);  s = "HTTP/1.1 200 OK\r\nContent-Type: text/html\r\n\r\n<!DOCTYPE HTML>\r\n<html>Hello from ESP32 at ";  s += ipStr;  s += "</html>\r\n\r\n";  Serial.println("Sending 200");  client.println(s); //Send the string S, you can read the information when visiting the address of E smart home using a browser.  //client.print(s);  }  if(req == "/led/on") //Browser accesses the ip address/led/on  {  client.println("turn on the LED");  digitalWrite(led\_y, HIGH);  }  if(req == "/led/off") //Browser accesses the ip address/led/off  {  client.println("turn off the LED");  digitalWrite(led\_y, LOW);  }  if(req == "/fan/on") //Browser accesses the ip address/fan/on  {  client.println("turn on the fan");  digitalWrite(fanPin1, LOW); //pwm = 0  analogWrite(fanPin2, 180);  }  if(req == "/fan/off") //Browser accesses the ip address/fan/off  {  client.println("turn off the fan");  digitalWrite(fanPin1, LOW); //pwm = 0  analogWrite(fanPin2, 0);  }  //client.print(s);  client.stop();  } |

**3. Test Result**

If the smart home is successfully connected to WiFi, the LCD screen will display the assigned address.

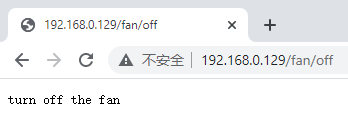


Accessing address must add / led/on when using the browser, such as my address is 192.168.0.129/ led/on. Then the smart home LED lights will be turned on, if accessing 192.168.0.129/ led /off, then the LED lights will be off.



When the browser accesses 192.168.0.129/fan/ on, the fan of

the smart home will be turned on and at 192.168.0.129/fan/ off will be turned off.

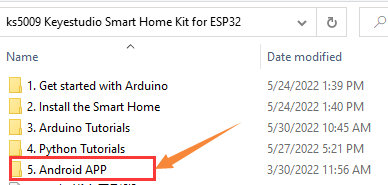


## Project 13: Mobile Phone APP

**Download APP**

**Android APP：**

The Android apk installation package is available in our resource pack, as shown below:



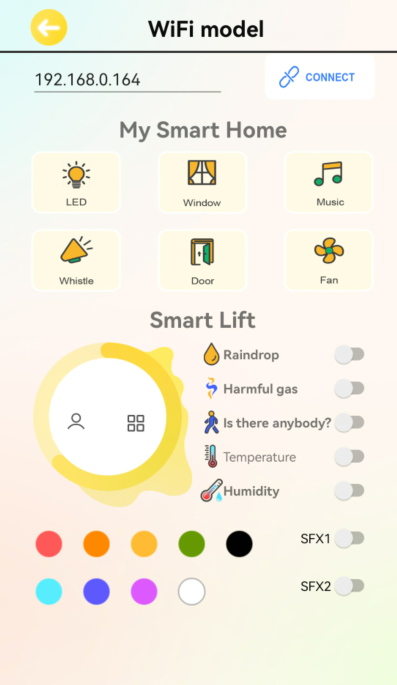
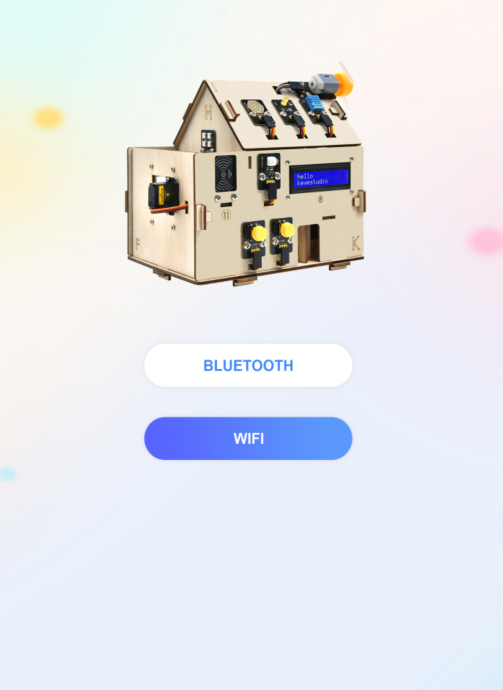
Download from Google play:

Please search for keyes IoT home on Google play to download it.

**Icon:**



APP [Interface](C:/Users/NINGMEI/AppData/Local/youdao/dict/Application/8.10.7.0/resultui/html/index.html" \l "/javascript:;)



**Download iOS APP**

Please search for keyes IoT home on APP Store to download it.

### Project 13.1 Test APP

**1. Description**

We will use APP to control the smart home LED lights and fan switches.

**2. Test Code**

|  |
| --- |
| #include <Arduino.h>  #include <WiFi.h>  #include <ESPmDNS.h>  #include <WiFiClient.h>  String item = "0";  const char\* ssid = "ChinaNet-2.4G-0DF0";  const char\* password = "ChinaNet@233";  WiFiServer server(80);  #include <Wire.h>  #include <LiquidCrystal\_I2C.h>  LiquidCrystal\_I2C mylcd(0x27,16,2);  #include <analogWrite.h>  #define fanPin1 19  #define fanPin2 18  #define led\_y 12 //Define the yellow led pin to 12  void setup() {  Serial.begin(115200);  mylcd.init();  mylcd.backlight();  pinMode(led\_y, OUTPUT);  pinMode(fanPin1, OUTPUT);  pinMode(fanPin2, OUTPUT);    WiFi.begin(ssid, password);  while (WiFi.status() != WL\_CONNECTED) {  delay(500);  Serial.print(".");  }  Serial.println("");  Serial.print("Connected to ");  Serial.println(ssid);  Serial.print("IP address: ");  Serial.println(WiFi.localIP());  server.begin();  Serial.println("TCP server started");  MDNS.addService("http", "tcp", 80);  mylcd.setCursor(0, 0);  mylcd.print("ip:");  mylcd.setCursor(0, 1);  mylcd.print(WiFi.localIP()); //LCD displays ip adress  }  void loop() {  WiFiClient client = server.available();  if (!client) {  return;  }  while(client.connected() && !client.available()){  delay(1);  }  String req = client.readStringUntil('\r');  int addr\_start = req.indexOf(' ');  int addr\_end = req.indexOf(' ', addr\_start + 1);  if (addr\_start == -1 || addr\_end == -1) {  Serial.print("Invalid request: ");  Serial.println(req);  return;  }  req = req.substring(addr\_start + 1, addr\_end);  item=req;  Serial.println(item);  String s;  if (req == "/") //Browser accesses address can read the information sent by the client.println(s);  {  IPAddress ip = WiFi.localIP();  String ipStr = String(ip[0]) + '.' + String(ip[1]) + '.' + String(ip[2]) + '.' + String(ip[3]);  s = "HTTP/1.1 200 OK\r\nContent-Type: text/html\r\n\r\n<!DOCTYPE HTML>\r\n<html>ESP32 ip:";  s += ipStr;  s += "</html>\r\n\r\n";  Serial.println("Sending 200");  client.println(s); //Send the string S, then you can read the information when visiting the address of E smart home using a browser.  }  if(req == "/led/on") //Browser accesses address ip address/led/on  {  client.println("turn on the LED");  digitalWrite(led\_y, HIGH);  }  if(req == "/led/off") //Browser accesses address ip address/led/off  {  client.println("turn off the LED");  digitalWrite(led\_y, LOW);  }  if(req == "/fan/on") //Browser accesses address ip address/fan/on  {  client.println("turn on the fan");  digitalWrite(fanPin1, LOW); //pwm = 0  analogWrite(fanPin2, 180);  }  if(req == "/fan/off") //Browser accesses address ip address/fan/off  {  client.println("turn off the fan");  digitalWrite(fanPin1, LOW); //pwm = 0  analogWrite(fanPin2, 0);  }  //client.print(s);  client.stop();  } |

**3. Test Result**

1. Open the APP and select WIFI



2. APP controls LED and the fan

The mobile phone and the smart home must share the same WiFi, or the smart home connects to the hotspot of the mobile phone.

APP input IP address (LCD1602 displays the assigned IP address), then click connect, the connection is successful if ESP32 IP: 192.168...... is displayed.

Next, you can click the LED, then the smart home LED will be turned on. Click the fan button and the fan will be turned on, as shown below:



### Project 13.2 IoT Smart Home

**1. Description**

The IOT smart home connects to the family WiFi through

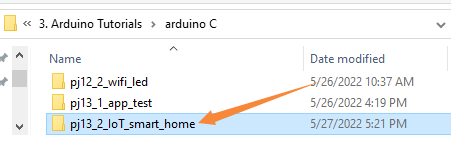
WiFi, and the mobile phone used for operation should also be connected to the same WiFi.

What’s more, the smart home also can connect to the hotspot of the mobile phone. If the connection is successful, the LCD1602 will display the IP address. Using the phone APP to input the corresponding IP for communication is

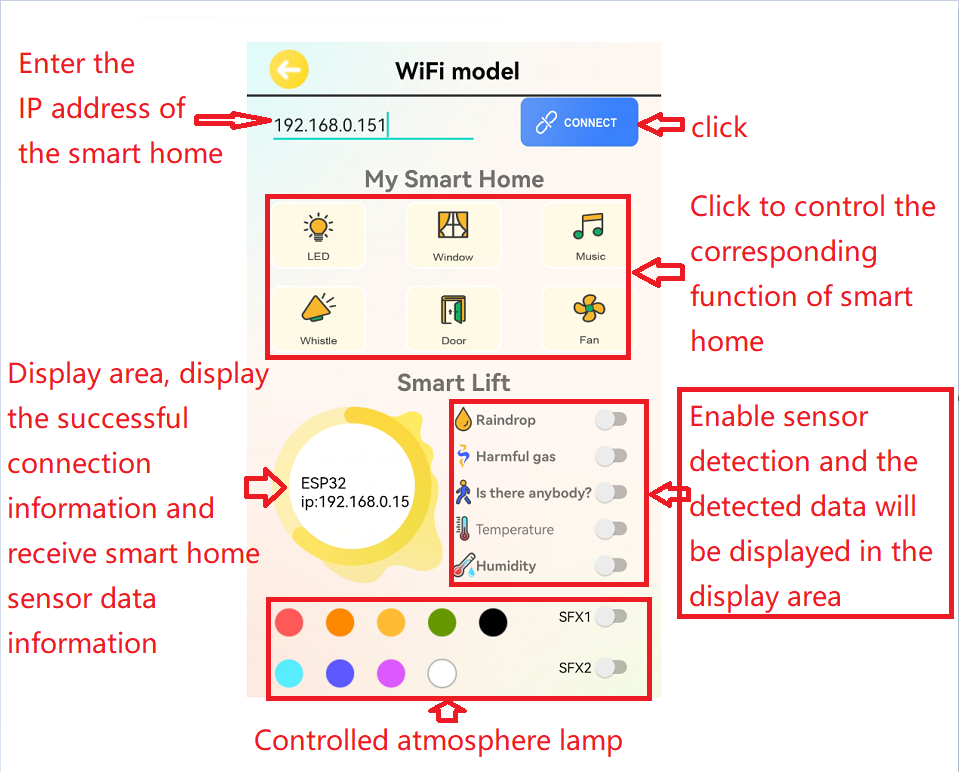
enable to realize the APP control of various functions of the smart home.

**2. Test Code**

Please refer to the sample code, as shown below:



**3. Test Result**



**7. Resources**

Download code, libraries and more details, please refer to the following link:

https://fs.keyestudio.com/KS5009