

# Table of contents

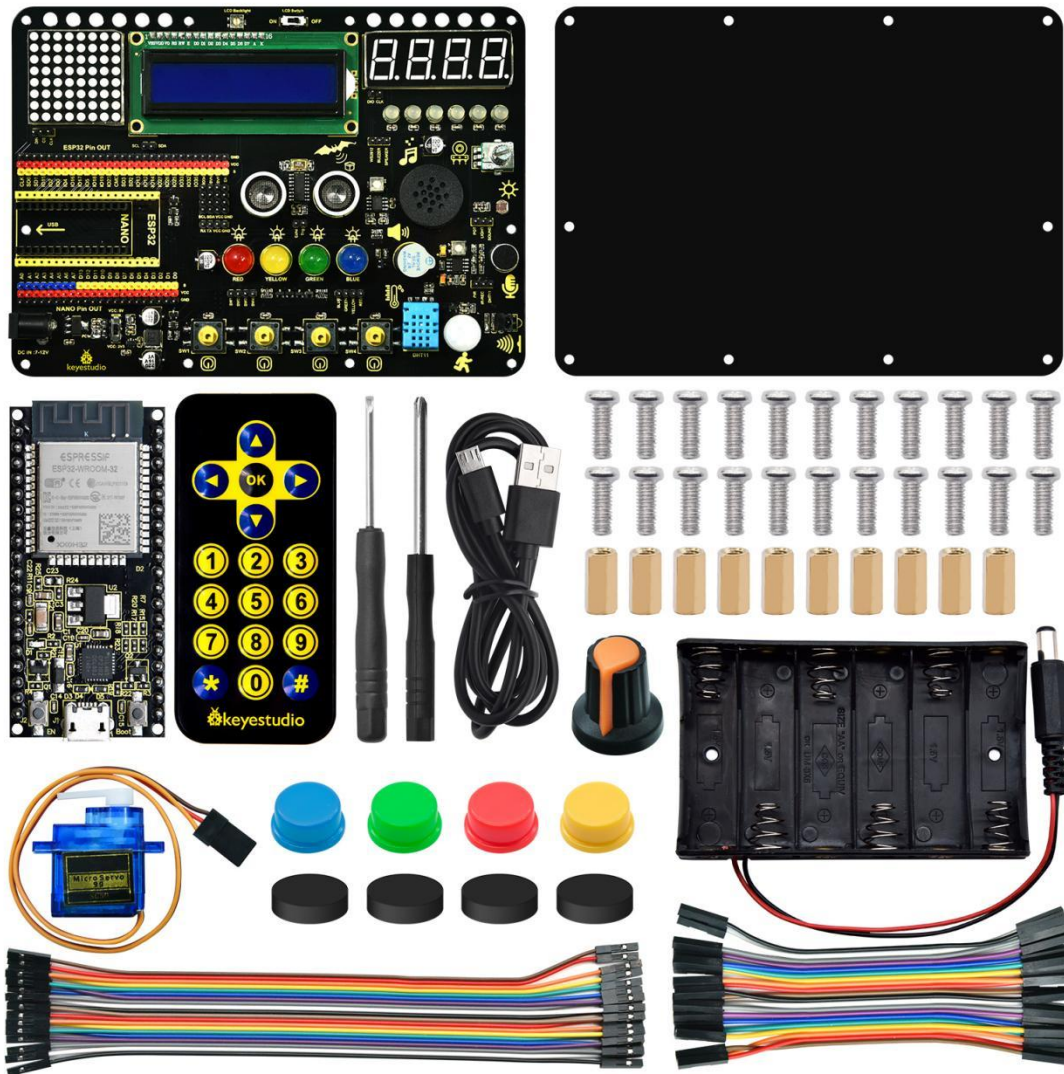
1. Product Introduction .....	3
3.ESP32 Control Board .....	7
4.ESP32 Integrated Board .....	11
5.Assembling the kit.....	13
6. Configuring Arduino .....	16
6.1 Install Arduino IDE and Driver .....	16
6.2 Install the Driver .....	23
6.3 Install the ESP32 environment in Arduino .....	27
6.4 Install Libraries .....	33
7. Arduino Project.....	36
Project 1: LED Blinking .....	36
Project 2: Breathing LED .....	39
Project 3: SOS Distress Device .....	42
Project 4: Traffic Light .....	44
Project 5: Rainbow Ambient Light .....	46
Project 6: Water Flow Light .....	51
Project 7: Active Buzzer .....	54
Project 8: Music Performer .....	56
Project 9: Digital Tube Display .....	62
Project 10: Dot Matrix Display .....	68
Project 11: LCD .....	74
Project 13: Mini Lamp .....	81
Project 14: Counter .....	87
Project 15: Responder .....	90
Project 16: Timebomb .....	93

Project 17: Invasion Alarm .....	96
Project 18: Beating Heart .....	101
Project 19 : Dimming Lamp .....	103
Project 20: Light Pillar .....	108
Project 21: Sound Controlled LED .....	113
Project 22: Noise Meter .....	119
Project 23: Smart Cup .....	121
Project 24: Weather Station .....	126
Project 25: Ultrasonic Rangefinder .....	128
Project 26: Human Body Piano .....	133
Project 27: Intelligent Parking .....	136
Project 28: Intelligent Gate .....	139
Project 29: IR Remote Data .....	141
Project 30: IR Remote Control .....	146
Project 31: Connect the ESP32 board to WiFi .....	150
Project 32: ESP32 WiFi Control LED .....	154
Project 33: ESP32 Reads Data .....	157
Project 34: Smart Home .....	160

## Read Me First

1. Please download all the files needed to run the inventor starter kit, including the driver, codes, libraries, etc:  
<https://fs.keyestudio.com/FKS0001>
2. Technical Support: [service@keyestudio.com](mailto:service@keyestudio.com)

# 1. Product Introduction



## 1)Description

This learning kit is a programmable tool specialized for kids above 6, which boasts 15 modules and sensors such as LEDs, buttons, a LCD, a photosensor, a sound sensor, an IR receiver, a temperature and humidity sensor as well as 30+ interesting projects.

Arduino C language is provided, which empower to cultivate programming thinking.

## 2) Features

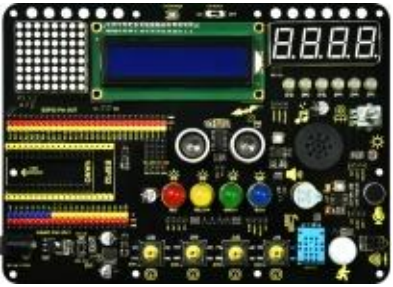









1. **Easy wiring:** The wiring of VCC and GND are hidden
2. **Multiple functions:** Nano or ESP32 development board are available as the control board and 15 sensors are integrated. When the external power supply is connected, the DIP switch can be used to control the VCC voltage to 3.3V or 5V.
3. **Simple structure:** Use 6mm dual-pass copper pillars to connect the acrylic board.
4. **High expansibility:** VCC and GNG pins are provided
5. **Programming learning:** Arduino C language is available.

## 3) Parameters

- **Working voltage:** 5V or 3.3V
- **DC power:** 7-12V
- **USB power:** 5V
- **Working current:**  $\geq 35\text{mA}$
- **Working temperature:**  $-10^{\circ}\text{C} \sim +65^{\circ}\text{C}$

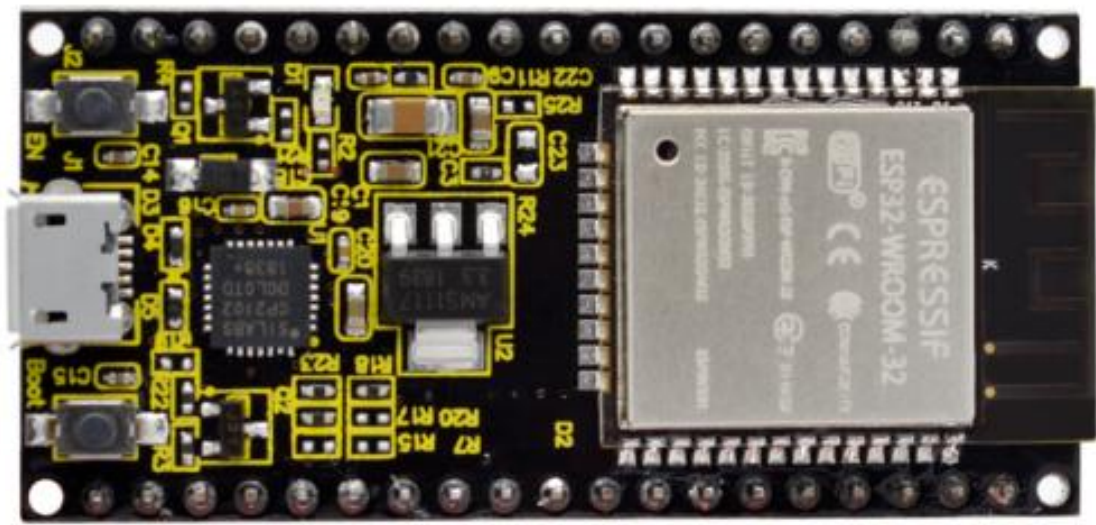


## 2.Kit List

#	Picture	Components	QTY
1		Keyestudio Circuit Integrated board	1
2		Keyestudio ESP32 Board	1
3		3XAA Battery Holder(batteries not included)	1
4		Servo	1
5		Slotted Screwdriver	1
6		Phillips Screwdriver	1
7		IR Remote Control	1
8		USB Cable	1
9		M2.5*6MM Round Head Screw	22
10		M2.5*9MM Dual-pass Copper Pillar	10

#	Picture	Components	QTY
11		20CM F-F DuPont Wires	20
12		10CM F-F DuPont Wires	20
13		Acrylic Board	1
14		Rubber Pad	4
15		Potentiometer Cap	1
16		Red Button Cap	1
17		Green Button Cap	1
18		Yellow Button Cap	1
19		Blue Button Cap	1

## 3.ESP32 Control Board



### 3.1 Introduction

Keystudio ESP32 Core board is a Mini development board based on the ESP-WROOM-32 module. The board has brought out most I/O ports to pin headers of 2.54mm pitch. These provide an easy way of connecting peripherals according to your own needs.

When it comes to developing and debugging with the development board, the both side standard pin headers can make your operation more simple and handy.

The ESP-WROOM-32 module is the industry's leading integrated WiFi + Bluetooth solution with less than 10 external components. It integrates antenna switches, RF balun, power amplifiers, low noise amplifiers, filters as well as power management modules. At the same time, it also integrates TSMC's low-power 40nm technology, power performance and RF performance, making it safe, reliable and easy to expand to a variety of applications.

### 3.2 Specifications

Microcontroller: ESP-WROOM-32 Module

USB-serial port chip: CP2102-GMR

Working voltage: DC 5V

Working current: 80mA (Average)

Current supply: 500mA (Minimum)

Working temperature range : -40°C ~ +85°C

WiFi mode: Station/SoftAP/SoftAP+Station/P2P

WiFi protocol : 802.11 b/g/n/e/i (802.11n, speed up to 150 Mbps)

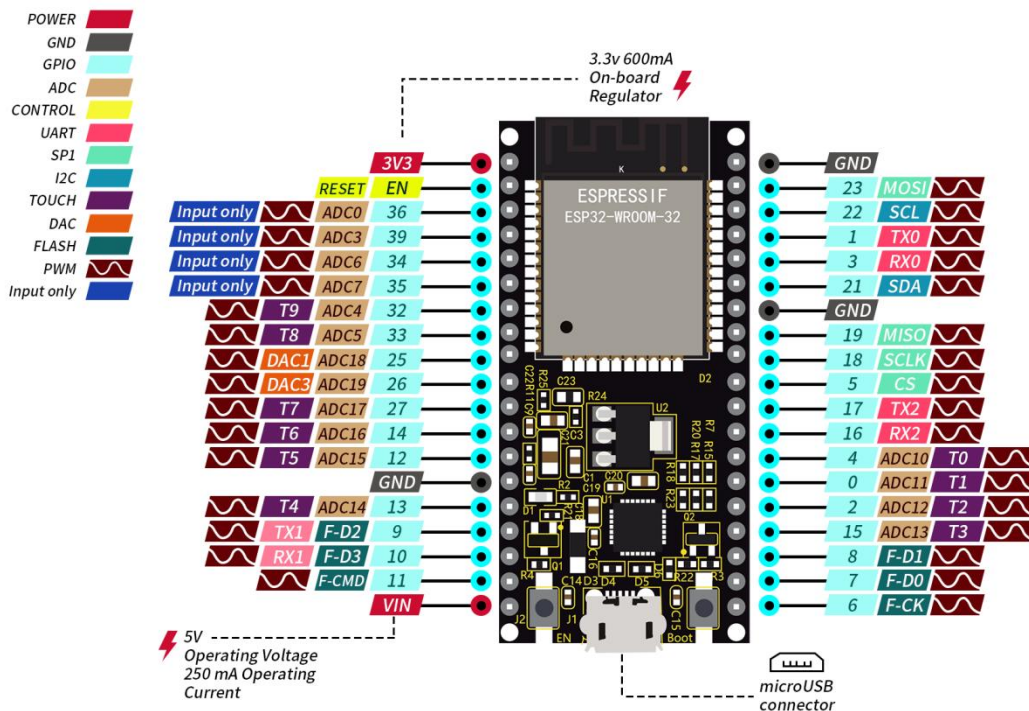
WiFi frequency range: 2.4 GHz ~ 2.5 GHz

Bluetooth protocol : conform to Bluetooth v4.2 BR/EDR and BLE Standard

Dimensions: 55x26x13mm

Weight: 9.3g

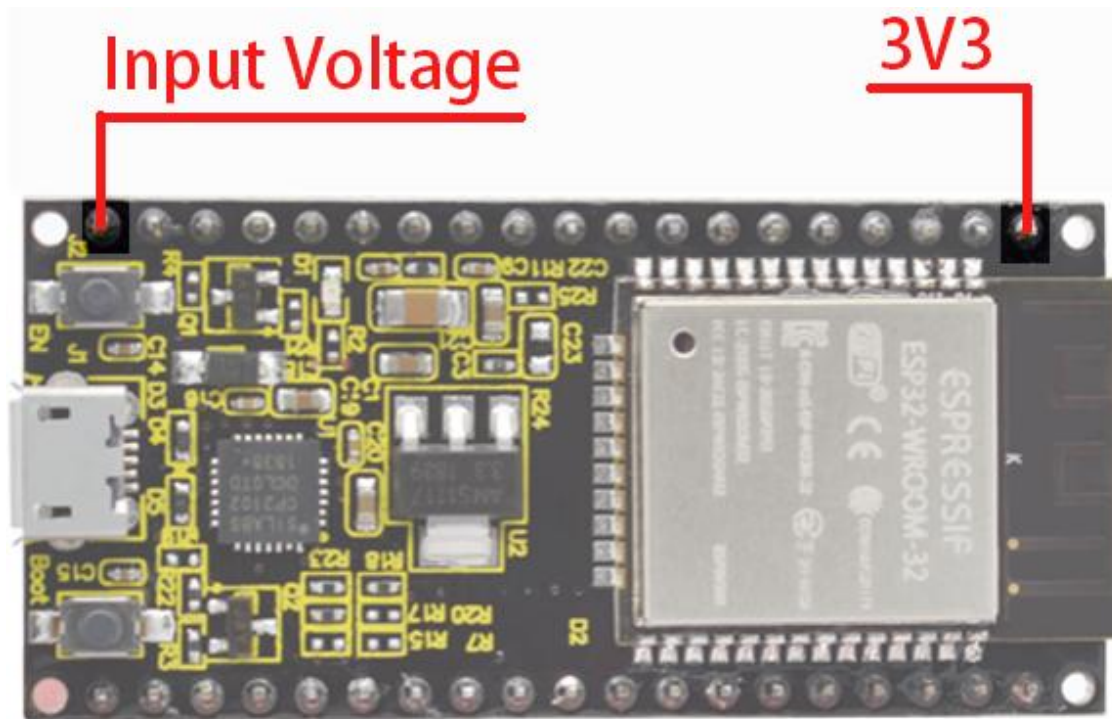
### 3.3 Pin out



ESP32 has fewer pins than commonly used processors, but it doesn't have any problems reusing multiple functions on pins.

**Warning:** The pin voltage level of the ESP32 is 3.3V. If you want to connect the ESP32 to another device with an operating voltage of 5V, you should use a level converter to convert the voltage level.

● **Power Pins:** The module has two power pins +5V and 3.3V. You can use these two pins to power other devices and modules.



● **GND Pins:** The module has three grounded pins.

● **Enable pin (EN) :** This pin is used to enable and disable modules. The pin enables module at high level and disables module at low level.

● **Input/Output pins (GPIO) :** You can use 32 GPIO pins to communicate with LEDs, switches and other input/output devices. You can also pull these pins up or down internally.

**Note:** Though GPIO6 to GPIO11 pins (SCK/CLK, SDO/SD0, SDI/SD1, SHD/SD2, SWP/SD3 and SCS/CMD pins) are used for SPI communication for the internal module, which are not recommended.

● **ADC:** You can use the 16 ADC pins on this module to convert analog voltages (the output of some sensors) into digital voltages. Some of these converters are connected to internal amplifiers and which are capable of measuring small voltages with high accuracy.

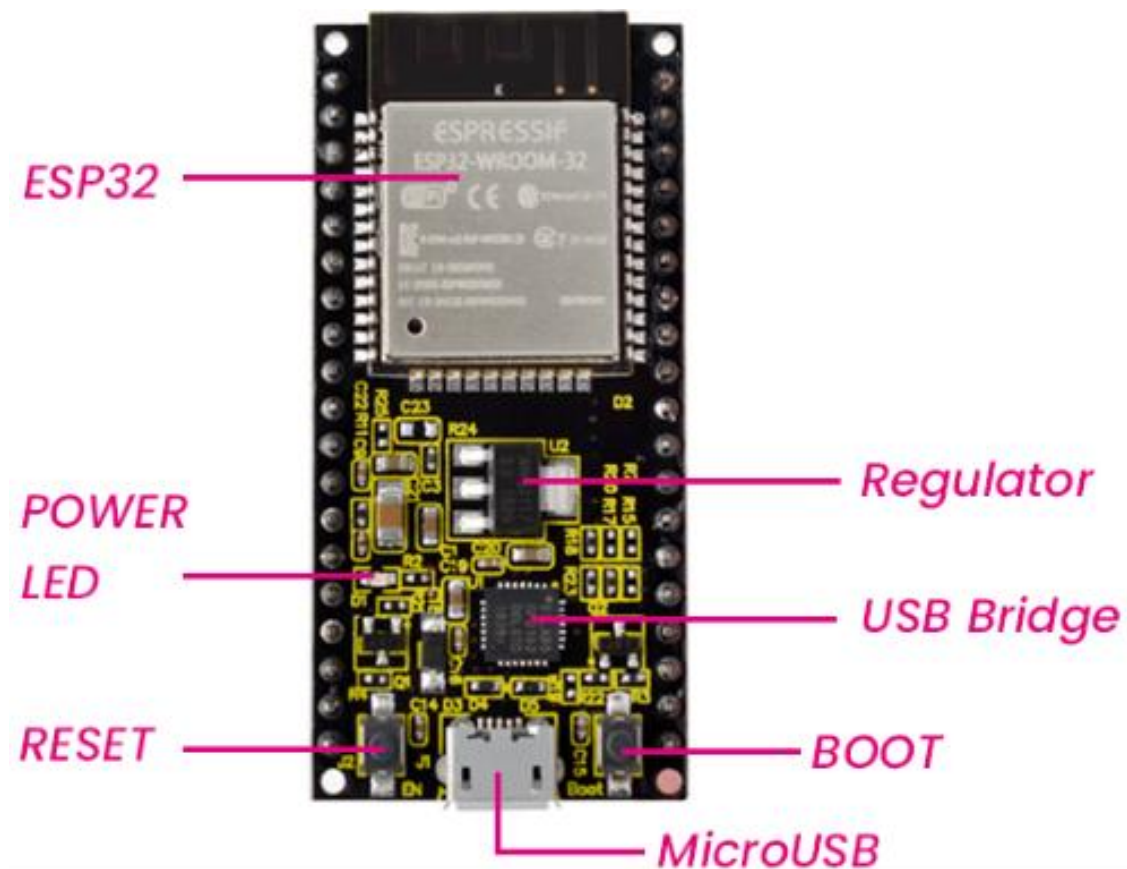
● **DAC:** ESP32 module has two A/D converters with 8-bit precision.

● **Touch pad:** There are 10 pins on the ESP32 module that are sensitive to capacitance changes. You can attach these pins to certain PCB's pads and use them as touch switches.

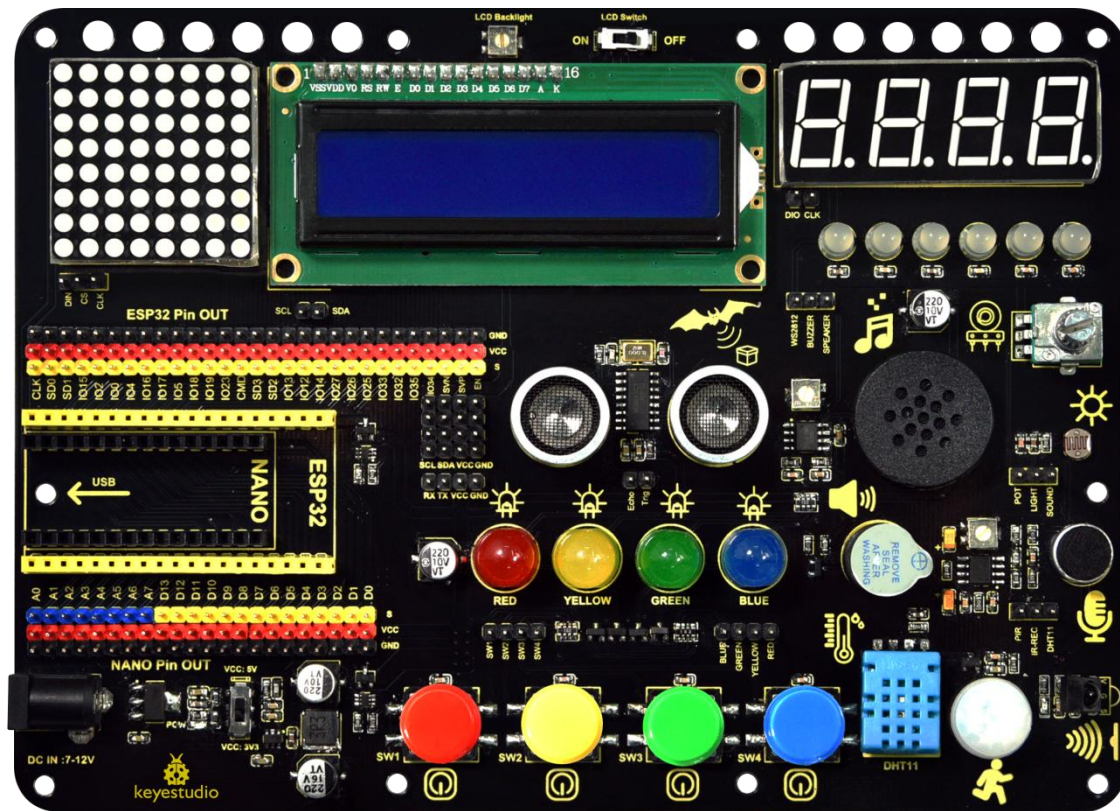


- **SPI:** There are two SPI interfaces on the module, which can be used to connect the display screen, SD/microSD memory card module as well as external flash memory.
- **I2C:** SDA and SCL pins are used for I2C communication.
- **Serial Communication (UART) :** There are two UART serial interfaces on this module, which can be used to transfer up to 5Mbps of information between two devices . The UART0 also has CTS and RTS control functions.
- **PWM:** Almost all ESP32 input/output pins can be used for PWM (pulse-width modulation). Using these pins can control motors, LED lights and colors.

### 3.4 Components



## 4.ESP32 Integrated Board

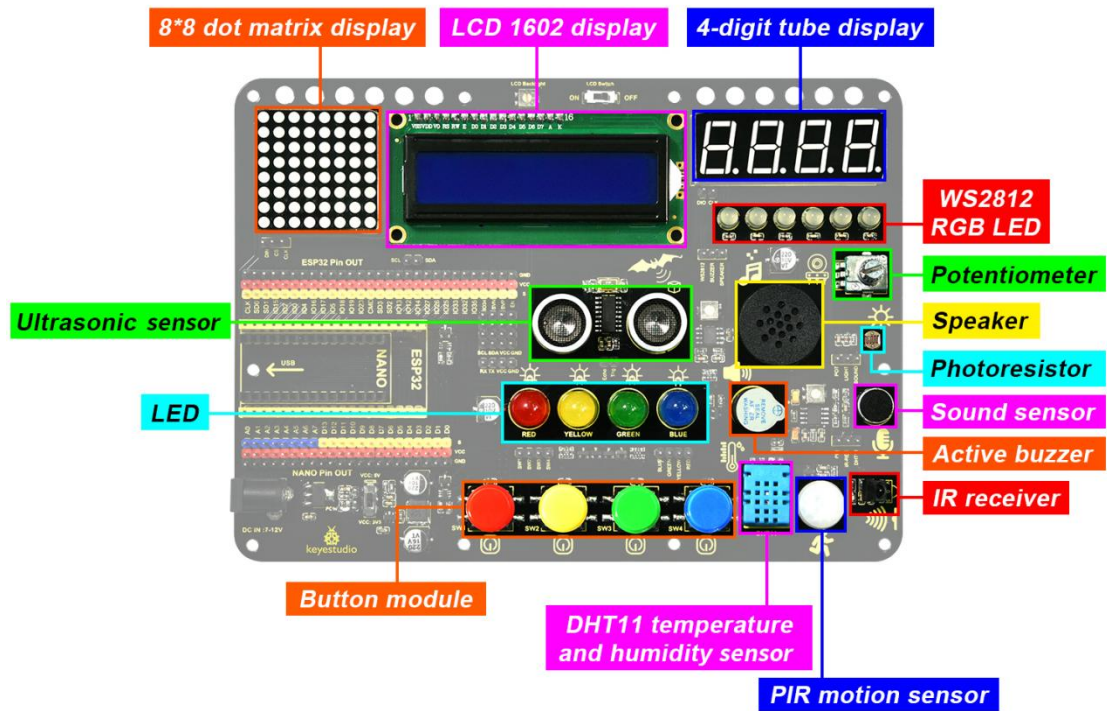


This expansion board is compatible with the Keyestudio Nano Plus development board and Keyestudio ESP32 development board. The voltage on VCC can be set to 3.3V (ESP32) or 5V(Nano) via a DIP switch.

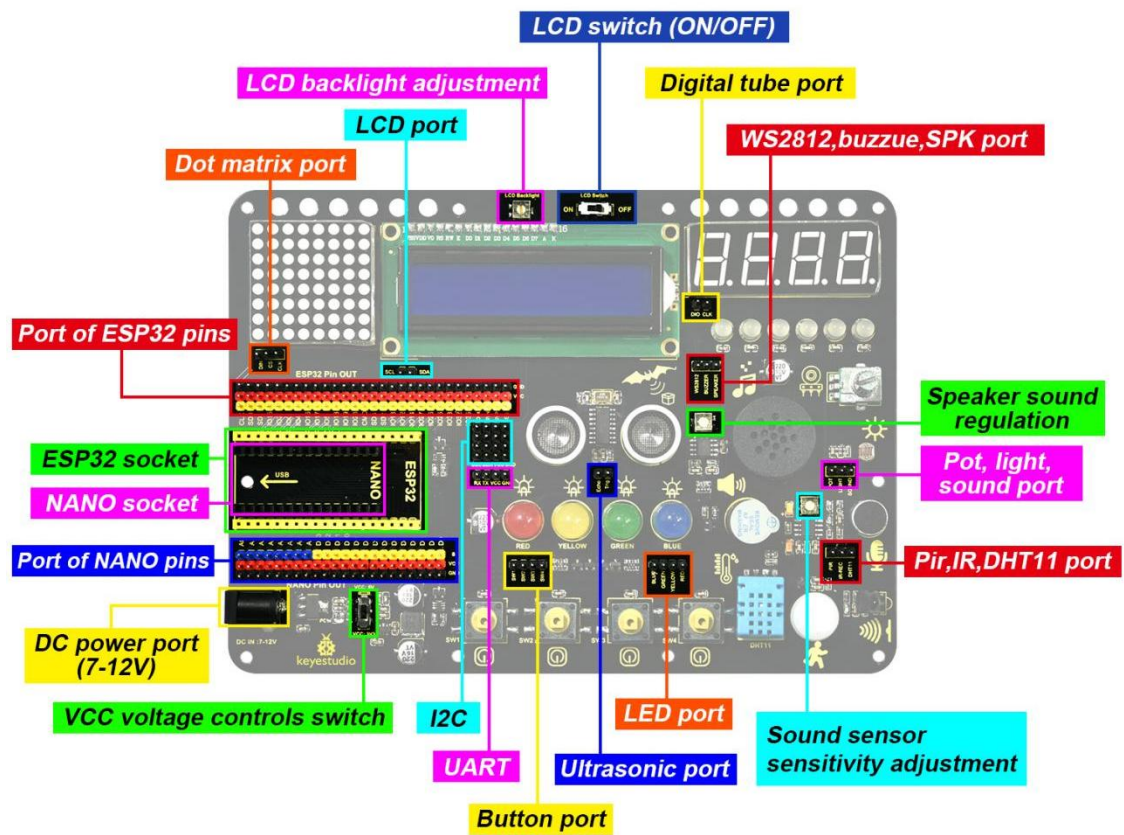
It integrates four buttons, four LEDs(red, yellow, green and blue), six WS2812 RGB LEDs, a buzzer, a PIR motion sensor, an IR receiver, a sound sensor, a photoresistor, a 8002 amplifier, a potentiometer, an ultrasonic sensor, a 4-bit digital tube display, a 8x8 dot matrix display and a LCD 1602 display.

What's more, each pin possesses its own VCC and GND, which brings a higher compatibility to the board and provides more opportunities for extended learning of other modules.

### Modules Introduction



## Pin out

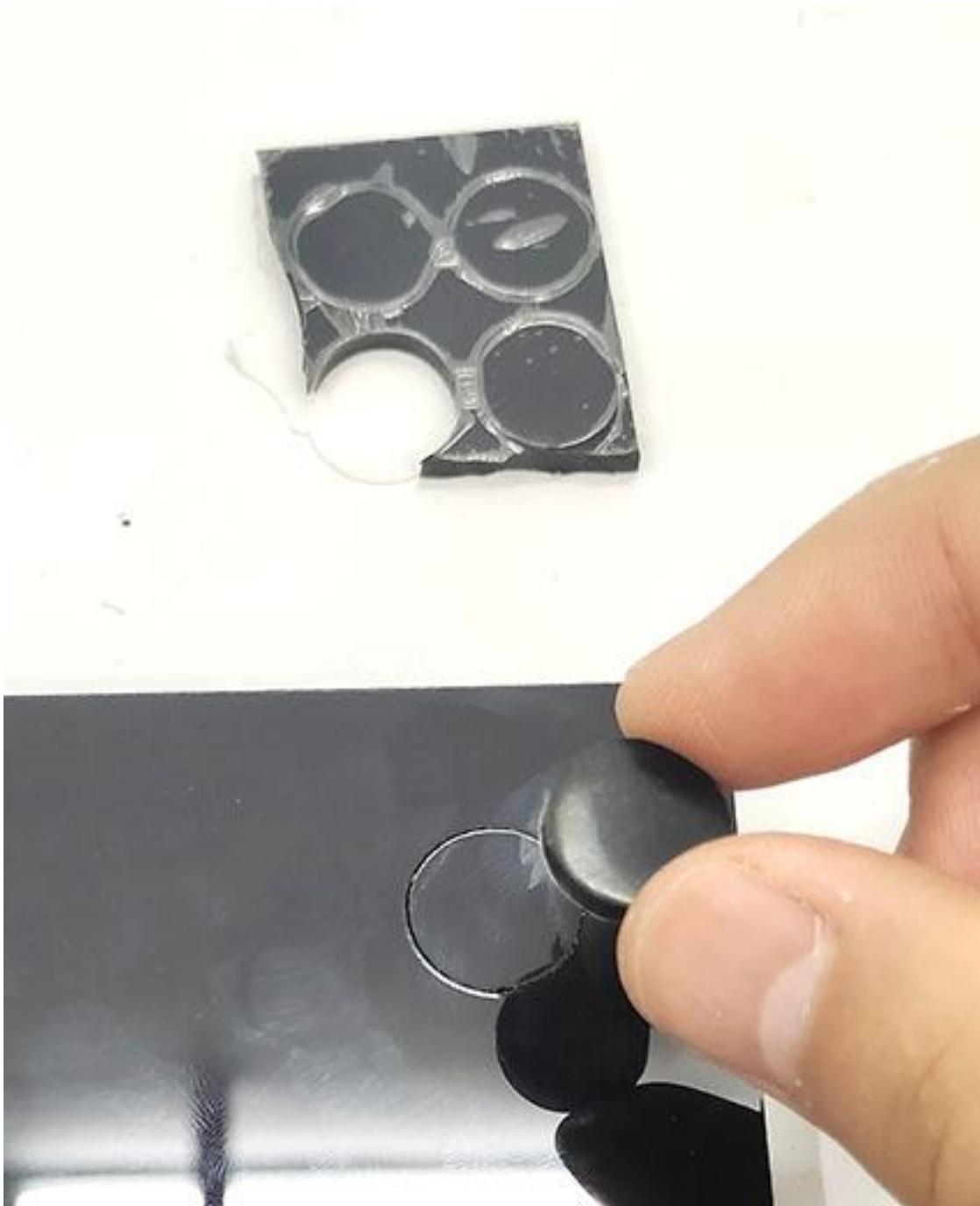




## 5.Assembling the kit

Step 1: Remove the protective film from the acrylic board.

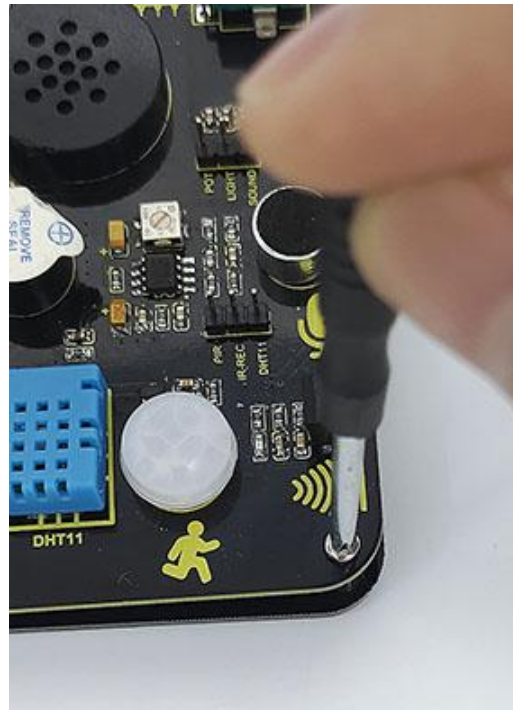
Step 2: Stick the four rubber pads in the positioning circle under the acrylic board.



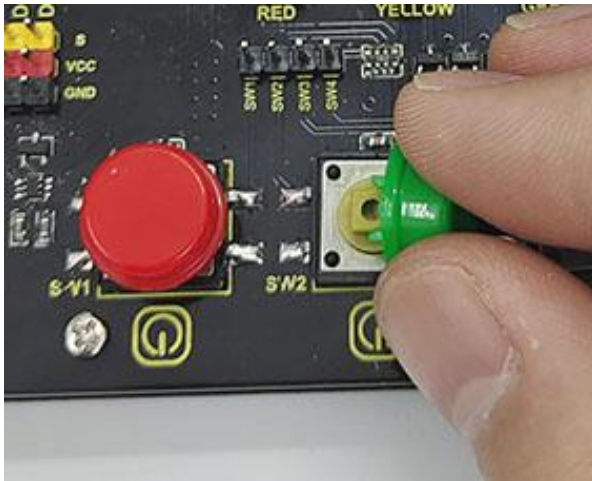
Step 3: Push the screws one by one through the holes from the bottom of the acrylic board (the side with rubber pad), and then screw copper pillars onto them, as shown below.



Step 4: Align the holes on the expansion board with copper pillars. Do note whether the middle two holes are aligned. If not, rotate the board to 180°. After adjustment, tighten the board with screws in place.



Step 5: Install the button caps and potentiometer caps.

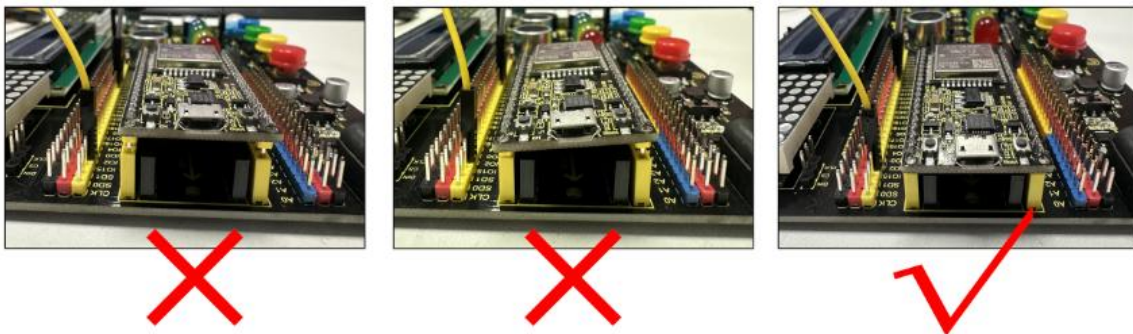


Step 6: Install the ESP32 board.

Pay attention:

1.The USB port of the board faces outwards. Installing it in the opposite direction may burn the board.

2.The EPS32 board needs to be fully inserted into the EPS32 port of the integrated board to ensure a stable connection between the EPS32 interface board and the integrated board, otherwise the kit may not work properly.



3.This kit consumes more power when driving the servo and connecting to WiFi. Please provide an external power supply for this kit.



## 6. Configuring Arduino

### 6.1 Install Arduino IDE and Driver

#### 1) Download and install Arduino IDE (Windows)


You could download the latest Arduino IDE from the official website:

<https://www.arduino.cc/en/software>

There are versions for Windows, Mac, and Linux systems.

Here we will choose the Windows version to show you how to download, install and use it. 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. Select **Win 10 and newer, 64 bits** in DOWNLOAD OPTIONS.



The screenshot shows the Arduino IDE 2.2.1 download page. The page has a teal header with navigation links: ACTION, STORE, HARDWARE, SOFTWARE, CLOUD, DOCUMENTATION, COMMUNITY, BLOG, and ABOUT. Below the header, there's a section for the Arduino Web Editor and a 'Downloads' section. The 'Downloads' section features the Arduino IDE 2.2.1 logo and a description of the new release. To the right, there's a 'DOWNLOAD OPTIONS' section with a red box highlighting 'Windows Win 10 and newer, 64 bits' and a red arrow pointing to it. Other options include Windows MSI installer, Windows ZIP file, Linux AppImage 64 bits (X86-64), Linux ZIP file 64 bits (X86-64), macOS Intel, 10.14: "Mojave" or newer, 64 bits, and macOS Apple Silicon, 11: "Big Sur" or newer, 64 bits. A 'Release Notes' link is also present.

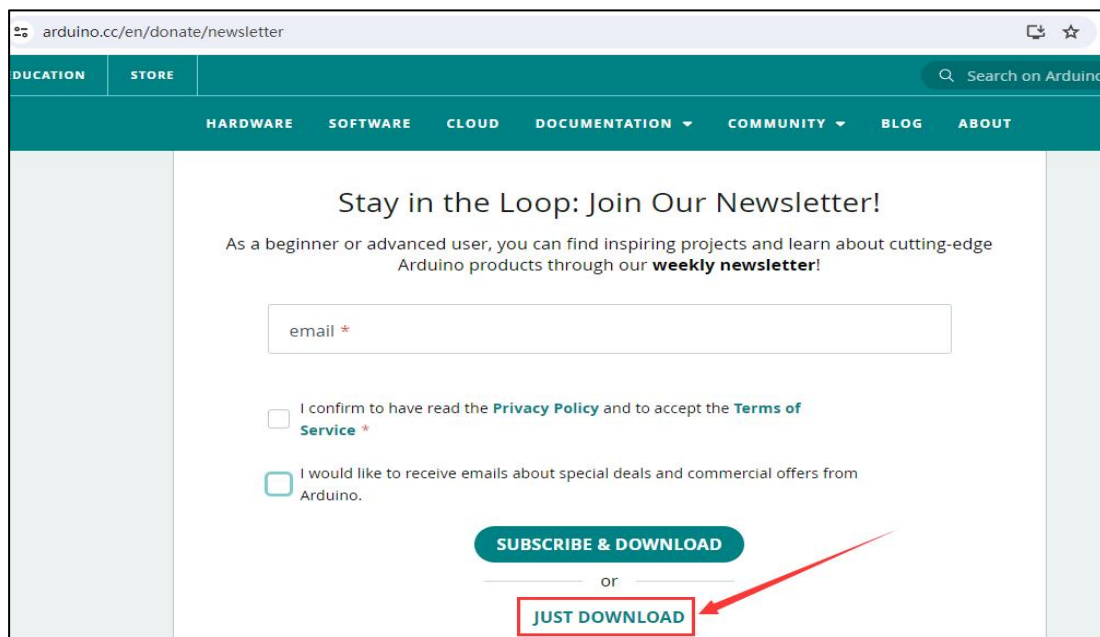
Operating System	Version/Architecture
Windows	Win 10 and newer, 64 bits
Windows	MSI installer
Windows	ZIP file
Linux	AppImage 64 bits (X86-64)
Linux	ZIP file 64 bits (X86-64)
macOS	Intel, 10.14: "Mojave" or newer, 64 bits
macOS	Apple Silicon, 11: "Big Sur" or newer, 64 bits

- 2. Click **JUST DOWNLOAD**



The screenshot shows the Arduino IDE download page at [arduino.cc/en/donate/](https://arduino.cc/en/donate/). The page has a teal header with navigation links: EDUCATION, STORE, HARDWARE, SOFTWARE, CLOUD, DOCUMENTATION, COMMUNITY, BLOG, and ABOUT. A search bar is on the right. The main content area has the heading "Download Arduino IDE & support its progress" and text stating that the IDE has been downloaded 77,971,368 times since its 1.x release in March 2015. Below this are donation buttons for \$3, \$5, \$10, \$25, \$50, and "Other". A teal button labeled "CONTRIBUTE AND DOWNLOAD" is followed by "or" and a red-outlined button labeled "JUST DOWNLOAD". A red arrow points to the "JUST DOWNLOAD" button.

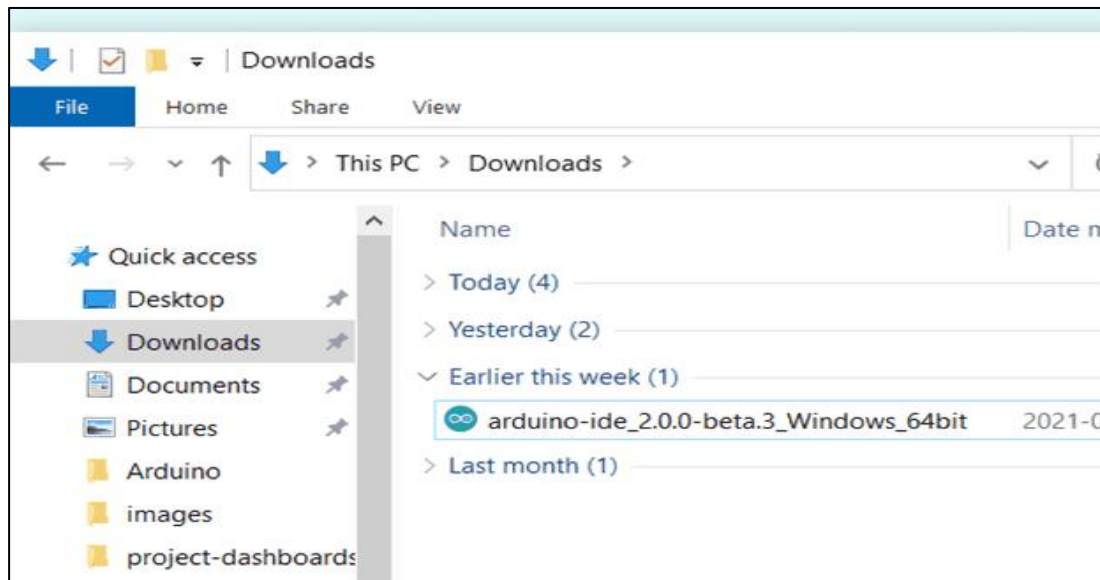
- 3. Join Newsletter or you can just Click **JUST DOWNLOAD**



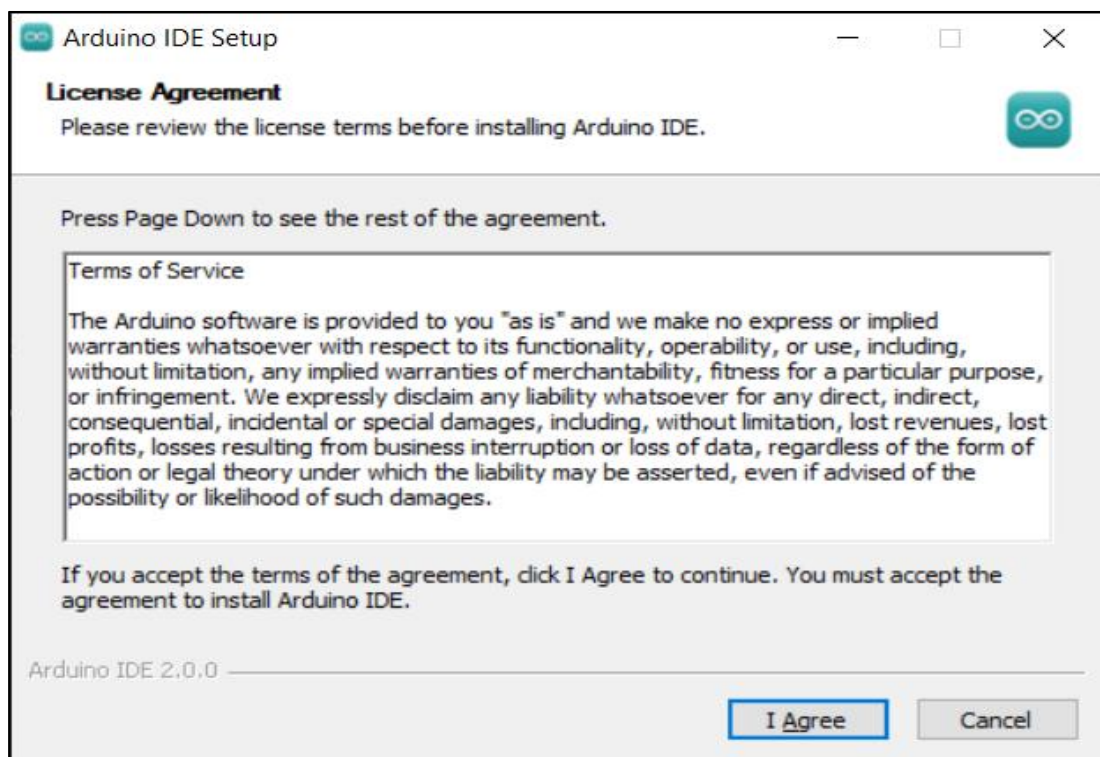
The screenshot shows the Arduino newsletter sign-up page at [arduino.cc/en/donate/newsletter](https://arduino.cc/en/donate/newsletter). The page has the same teal header as the previous screenshot. The main content area has the heading "Stay in the Loop: Join Our Newsletter!" and text stating that users can find inspiring projects and learn about cutting-edge Arduino products through their weekly newsletter. Below this is an email input field with a red asterisk. There are two checkboxes: "I confirm to have read the Privacy Policy and to accept the Terms of Service \*" and "I would like to receive emails about special deals and commercial offers from Arduino.". A teal button labeled "SUBSCRIBE & DOWNLOAD" is followed by "or" and a red-outlined button labeled "JUST DOWNLOAD". A red arrow points to the "JUST DOWNLOAD" button.



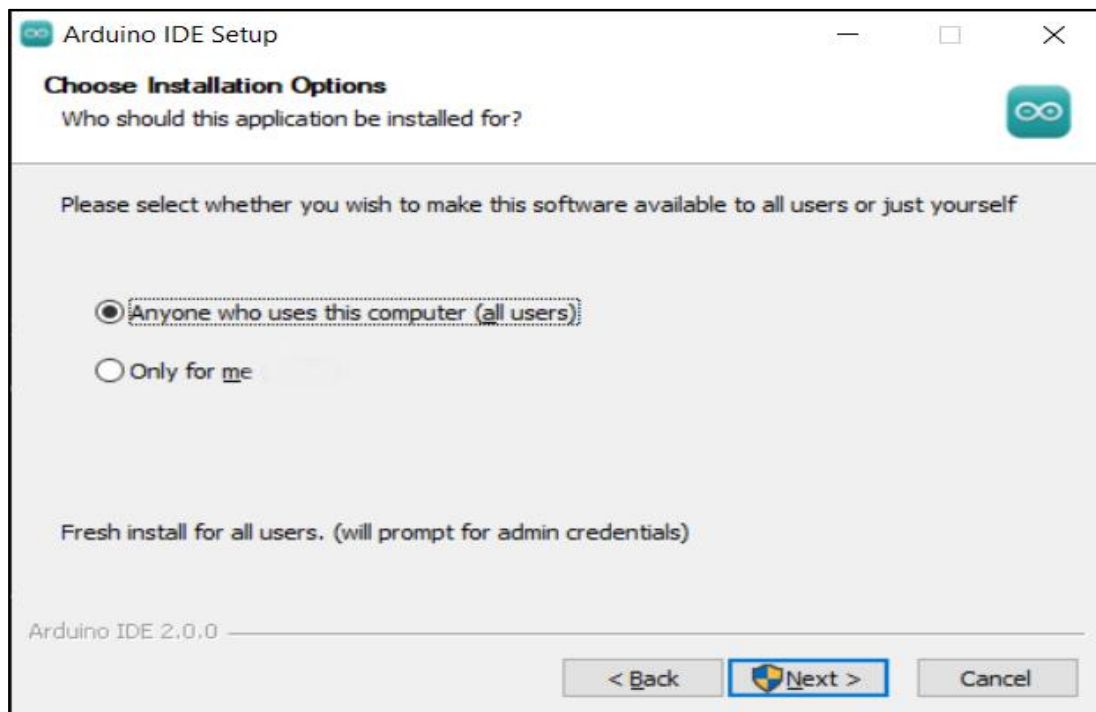
- 4. Save the .exe file downloaded from the software page to your hard drive and simply run the file .



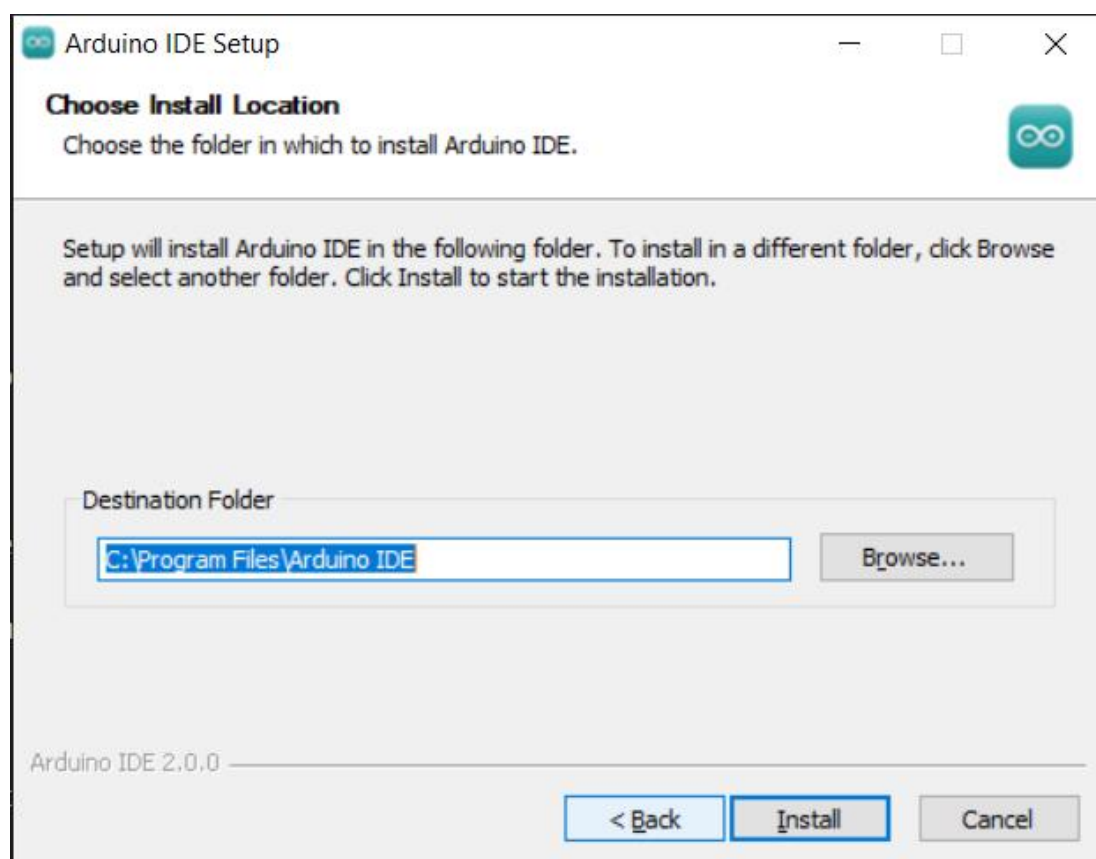
- 5. Read the License Agreement and agree it.



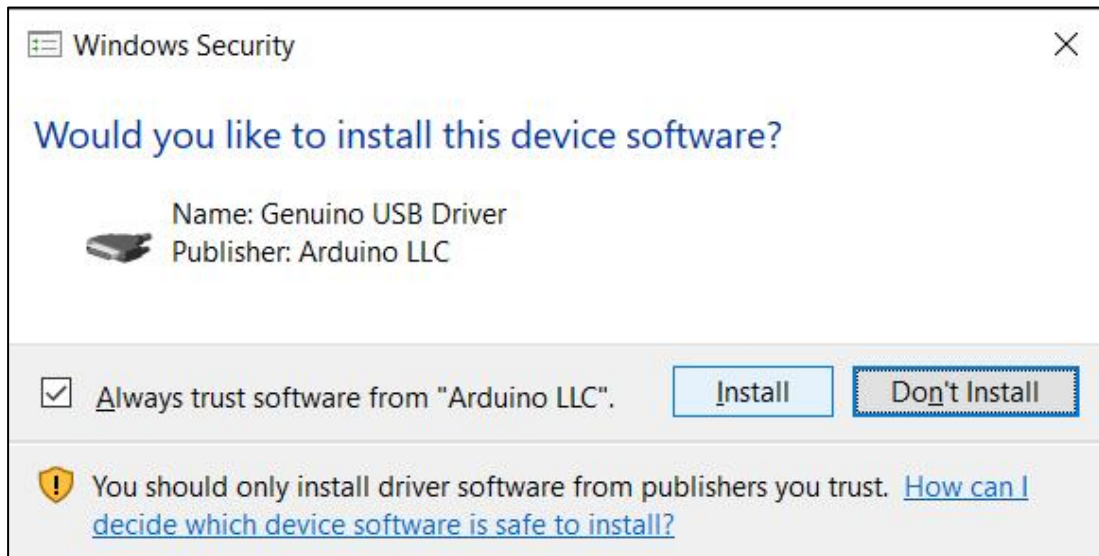
- 6. Choose the installation options.



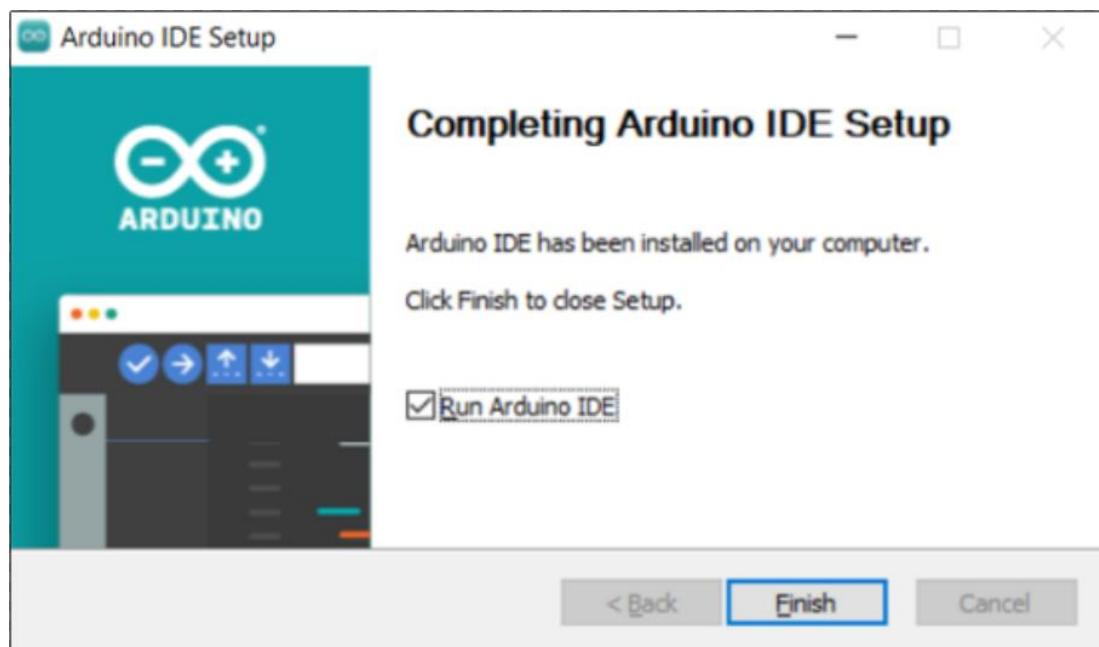
- 7. Choose the install location.



8. In addition, the security center may pop up a few times asking you if you want to install some device driver. Please install all of them.

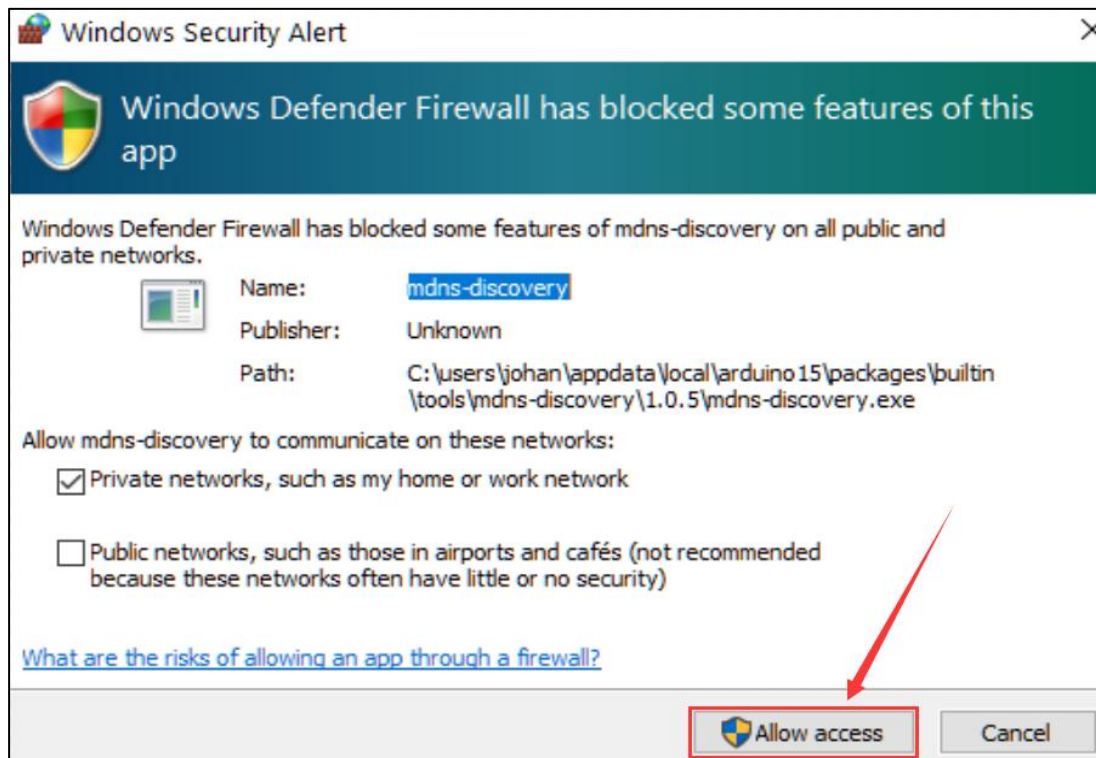


9. Click finish and run Arduino IDE





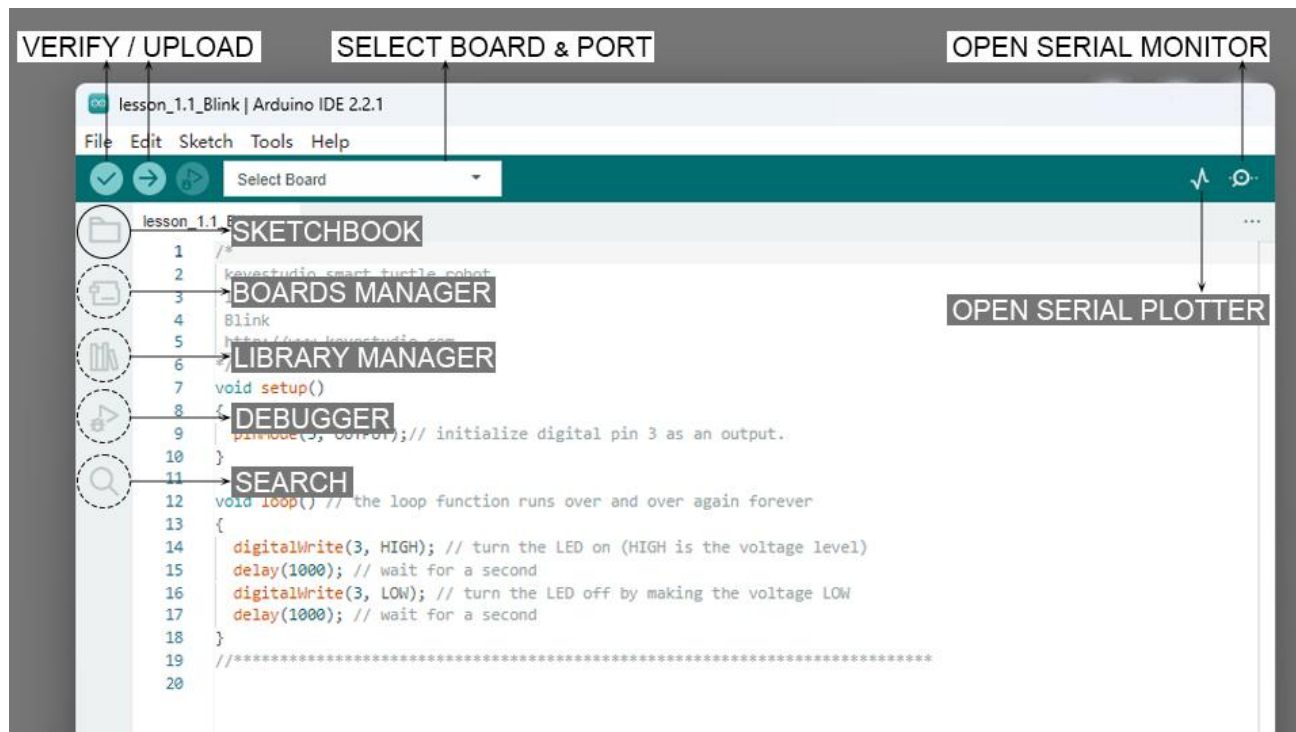
10. Firewall will ask whether we'd like to give allow access, just simply click on Allow access.



11. Wait for some time to allow arduino IDE to automatically install the Arduino AVR Boards, built-in libraries, and other required files.

```
Output
Downloading packages
arduino:avr-gcc@7.3.0-atmel3.6.1-arduino7
arduino:avrdude@6.3.0-arduino17
arduino:arduinoOTA@1.3.0
arduino:avr@1.8.6
Installing arduino:avr-gcc@7.3.0-atmel3.6.1-arduino7
arduino:avr-gcc@7.3.0-atmel3.6.1-arduino7 installed
Installing arduino:avrdude@6.3.0-arduino17
arduino:avrdude@6.3.0-arduino17 installed
Installing arduino:arduinoOTA@1.3.0
```

## 2) 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](#)

## 6.2 Install the Driver

### 1).Windows System

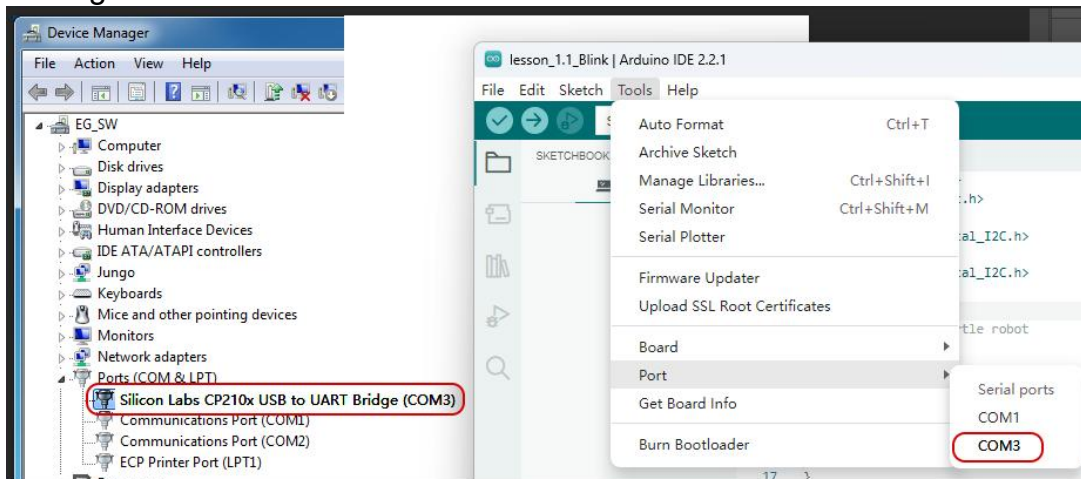
The USB-to-serial chip of the ESP32 board is CP2102.

Connect the ESP32 board to the computer with the usb cable and wait for Windows to begin its driver installation process. Often CP2102 drivers 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.

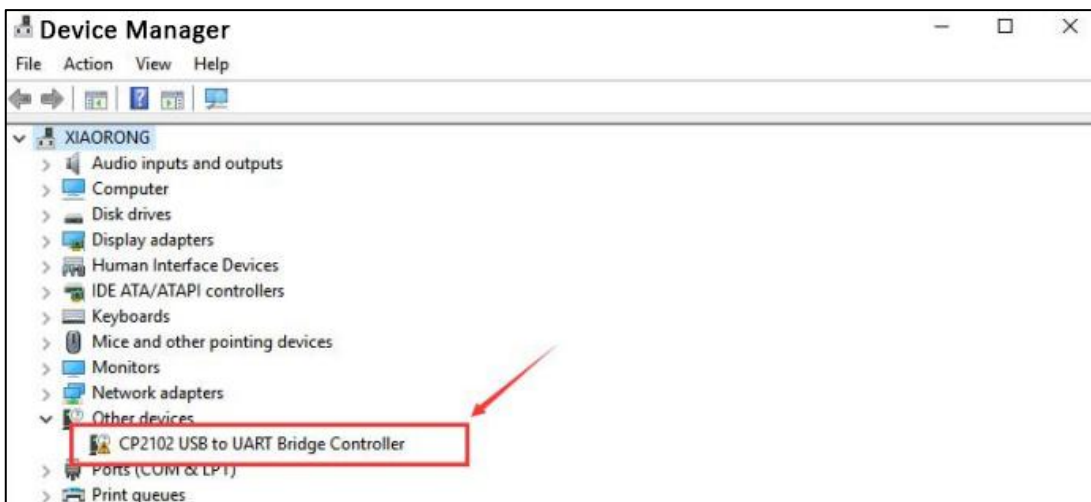
Open the **Device Manager** by right clicking **"My computer"** and selecting **control panel**.

Look under **Ports (COM & LPT)**. You should see an open port named **Silicon Labs CP210x USB to UART Bridge (COM-X)**

Click **Tools>Port** at Arduino IDE, you can find the com port displayed by device manager



If the installation process fail, you should see a device with a tiny yellow triangle and exclamation mark next to it.



**Now let's install CP210x Chip driver manually if the installation process fail.**

1. In the tutorial package we downloaded(<https://fs.keyestudio.com/FKS0001>), you can find the CP210x\_6.7.4 driver file.



CP210x\_6.7.4

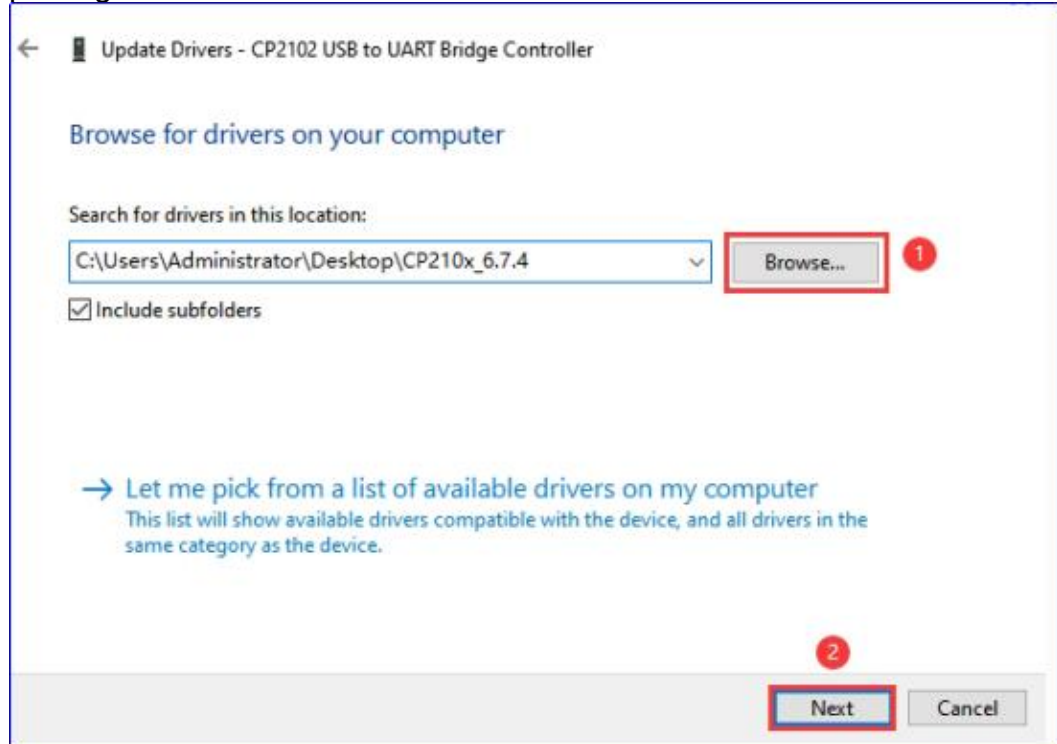
2. Right click on the **"CP210x USB to UART Bridge Controller"** and choose the **"Update Driver Software"** option.



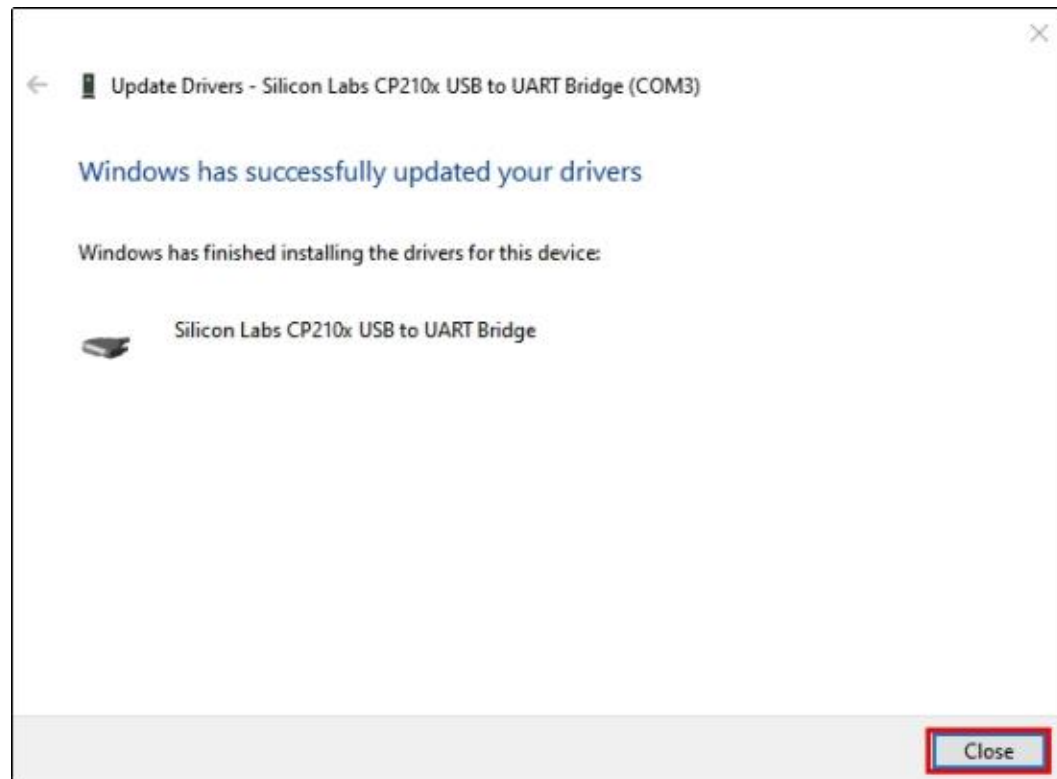
3. Choose the **"Browse my computer for Driver software"** option.



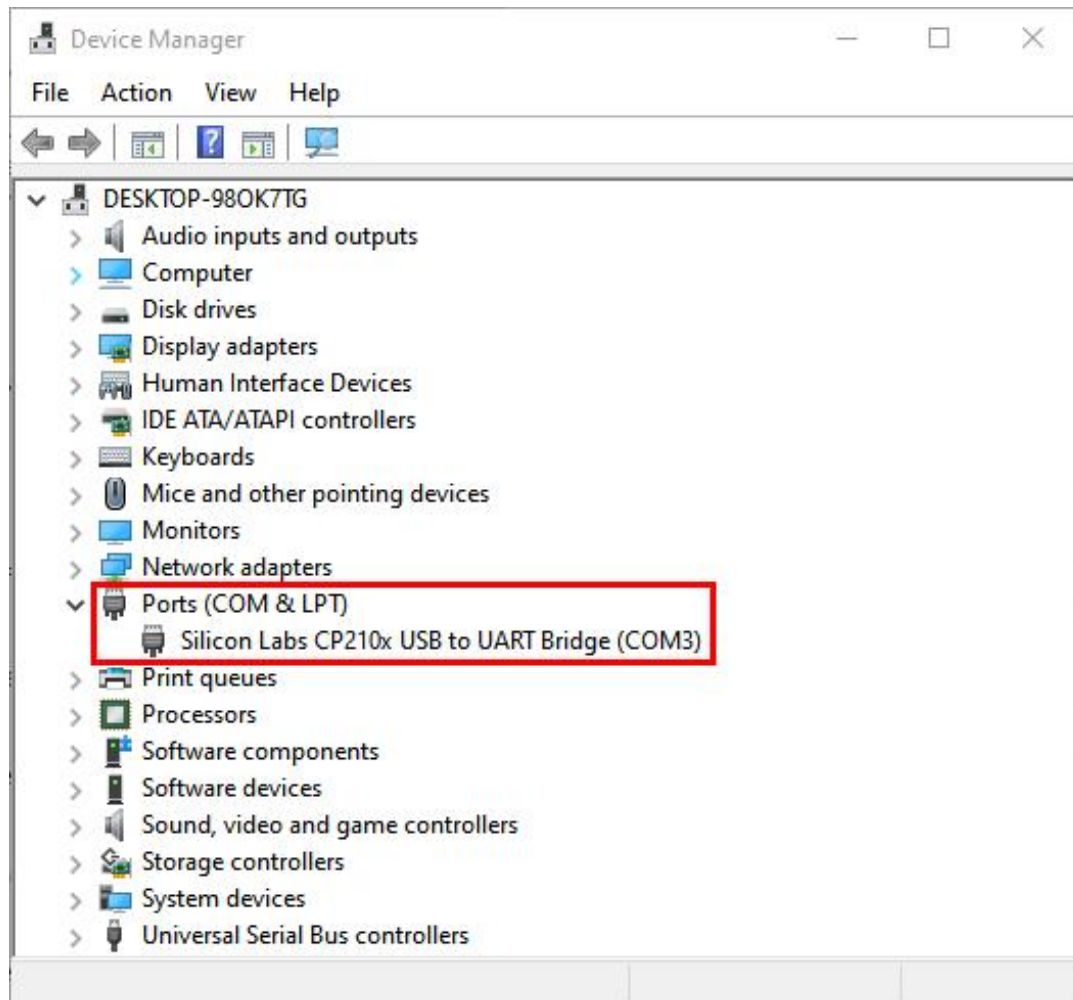
4. Select the driver file named **"CP210x\_6.7.4"**, located in the tutorial package we downloaded.



5. After a while, the driver is installed successfully.



6. Finally, click Computer--Attributes--Device Manager:



## 2).MAC System

Link:

[https://wiki.keyestudio.com/How\\_to\\_Install\\_the\\_Driver\\_of\\_CP2102\\_on\\_MAC\\_System](https://wiki.keyestudio.com/How_to_Install_the_Driver_of_CP2102_on_MAC_System)



## 6.3 Install the ESP32 environment in Arduino

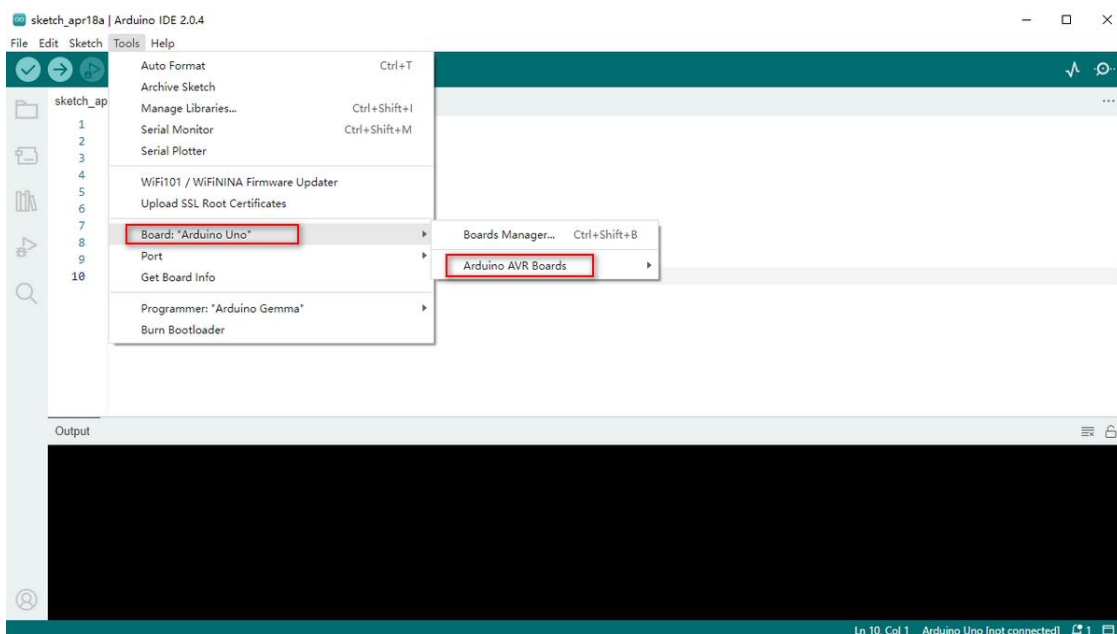
More information for the ESP32 Development Board:

[https://docs.espressif.com/projects/arduino-esp32/en/latest/getting\\_started.html#about-arduino-esp32](https://docs.espressif.com/projects/arduino-esp32/en/latest/getting_started.html#about-arduino-esp32)

### 1). Windows System

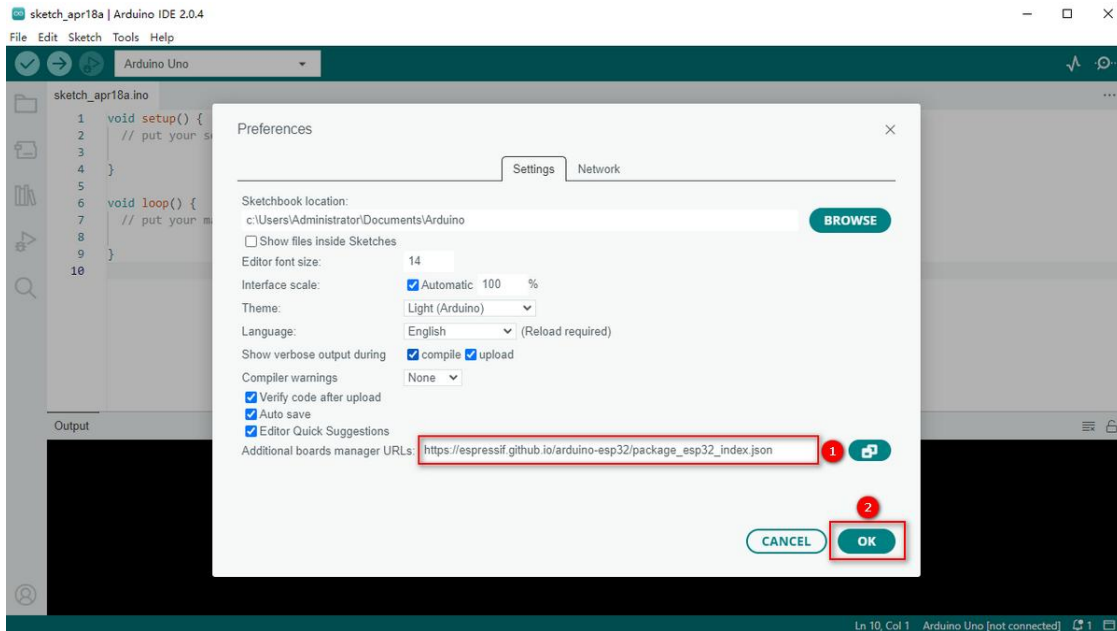
Before using Arduino IDE to program the starter kit, you need to configure the Arduino IDE, select the correct board type (**ESP32 Dev Module**) for the ESP32 board, and select the **COM port** that is assigned in the device manager.

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

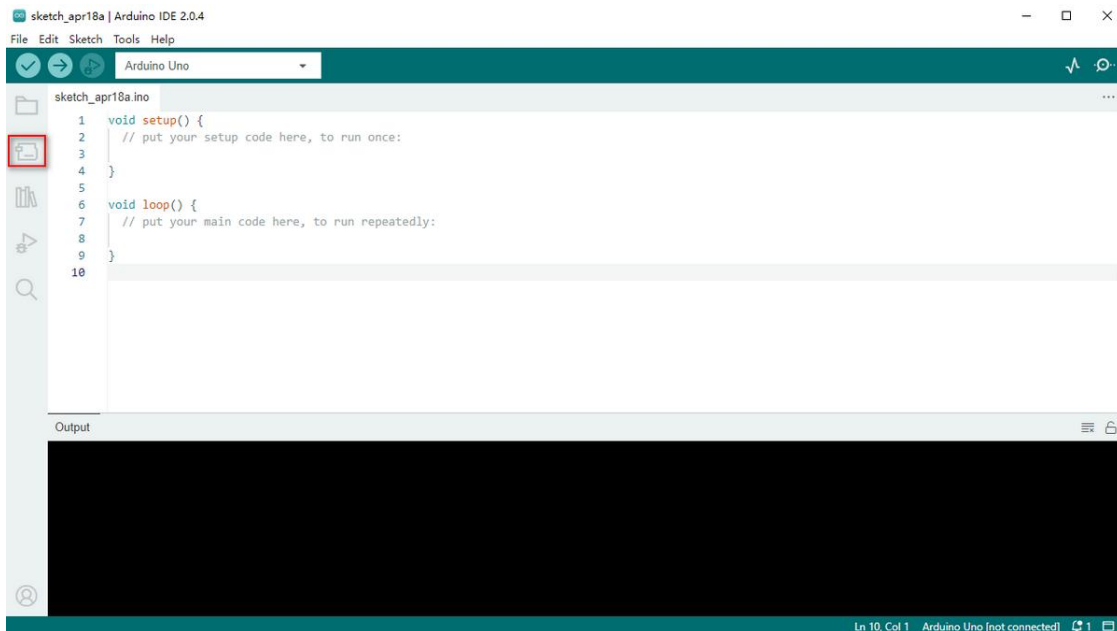


## Install the development environment for ESP32:

Click **"File» Preferences"**, copy and paste the link "[https://espressif.github.io/arduino-esp32/package\\_esp32\\_index.json](https://espressif.github.io/arduino-esp32/package_esp32_index.json)" in **Additional boards manager URLs** and click **OK**.



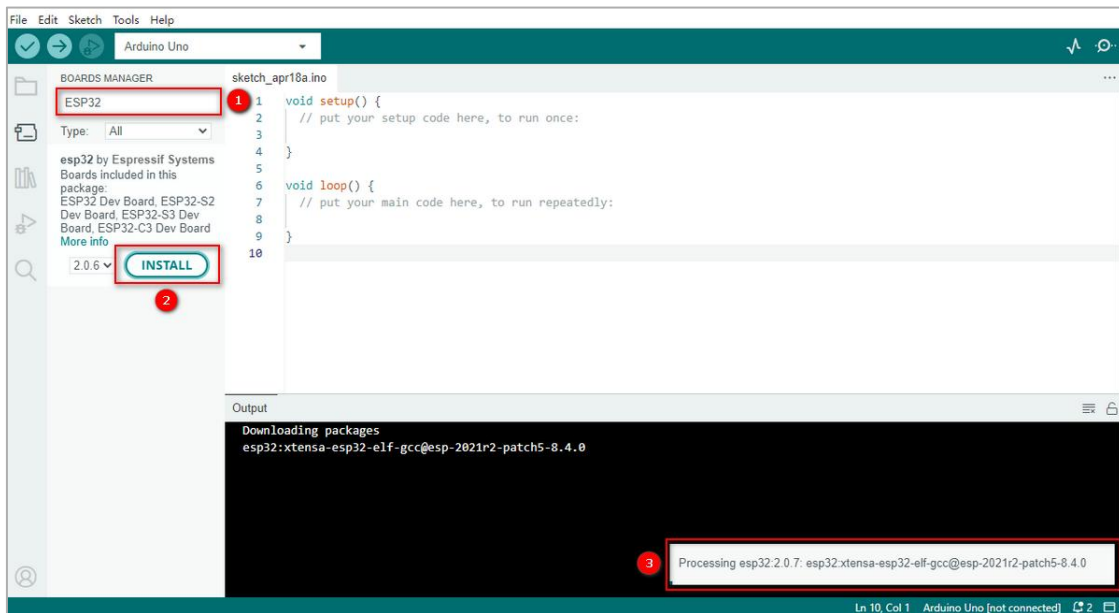
Click the icon of **"Board Manager"**  in the upper left corner.



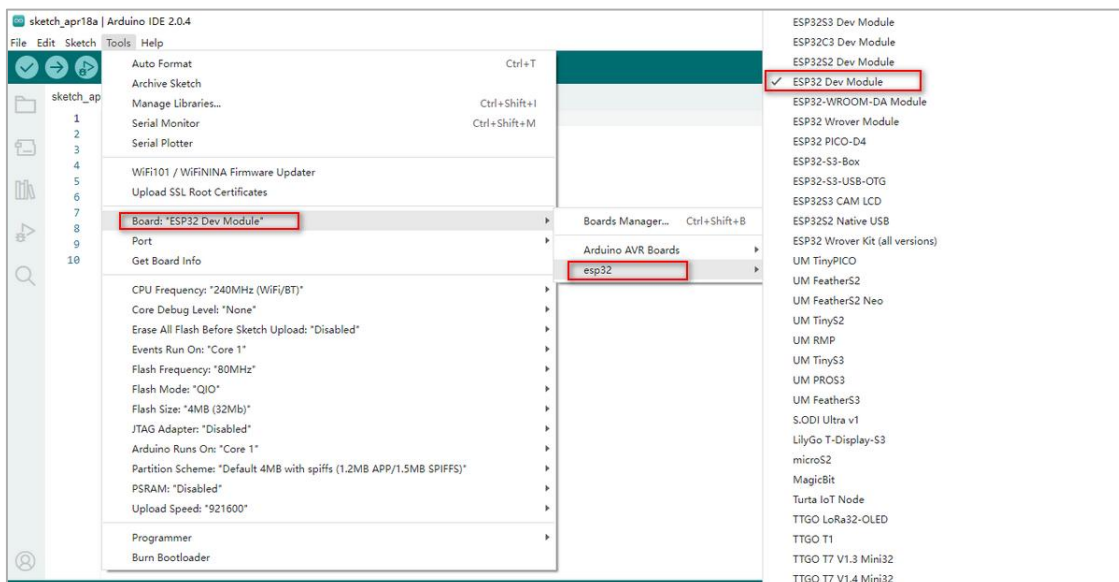


Enter "**esp32**" in the search box. Install the ESP32 environment in Arduino : **2.0.6** (ESP32 by Espressif Systems).

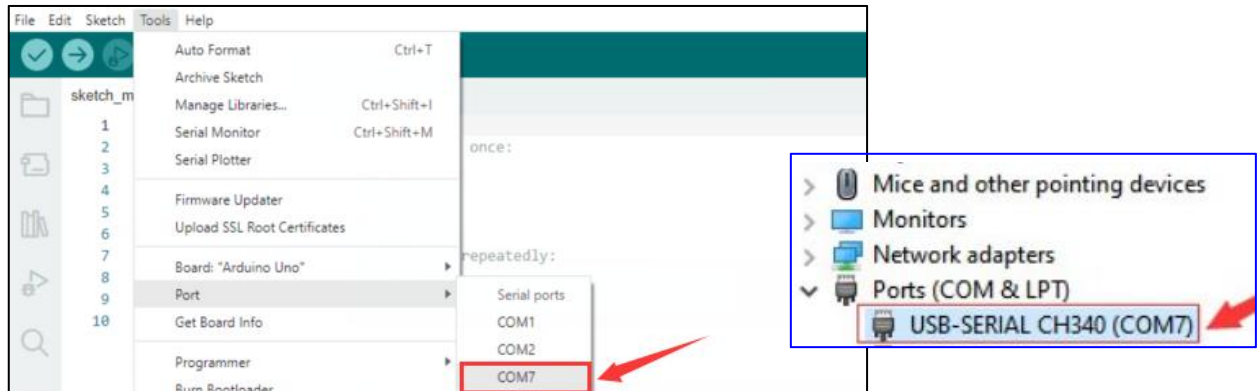
**!** Note: You must select version **2.0.6** of the ESP32 environment, otherwise it may be incompatible with the library files we provide.



After the **2.0.6** version development environment is installed, Click **Tools> Board > esp32**, you can choose the **EPS32 Dev Module**

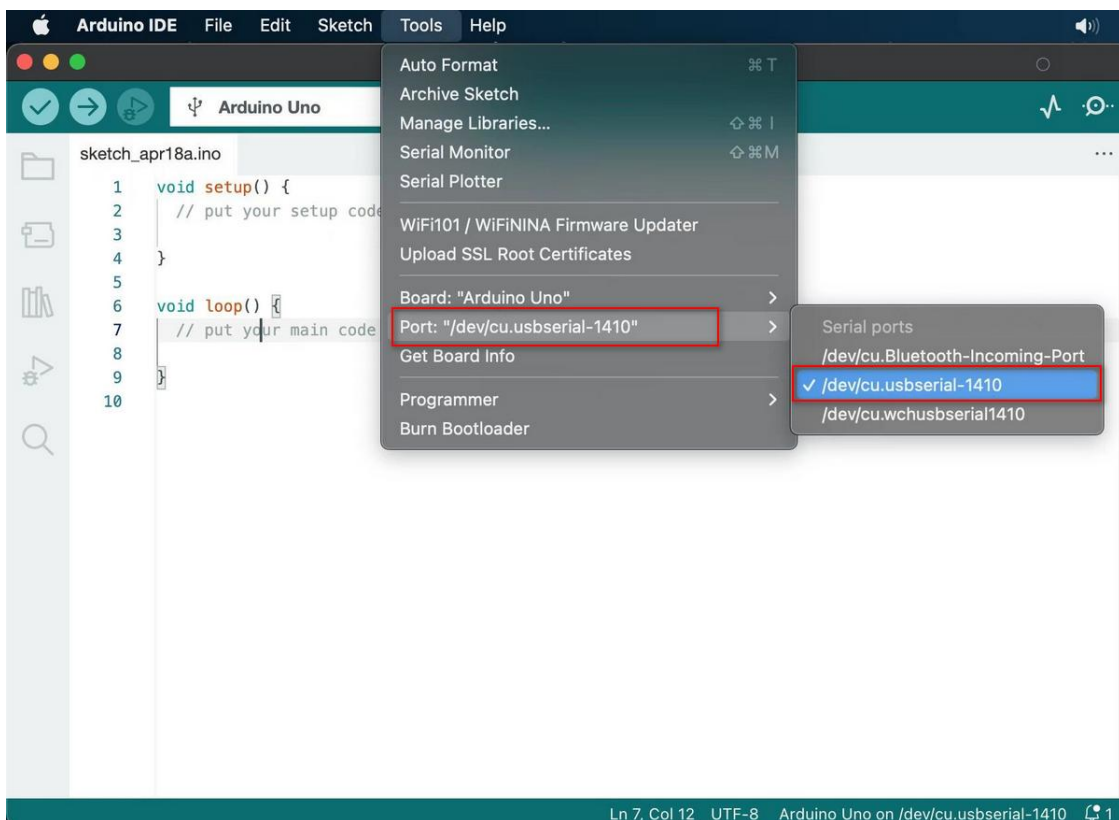


Select the **COM port** assigned to the ESP32 board in the device manager and you can start programming.(The COM number depends on your computer.)

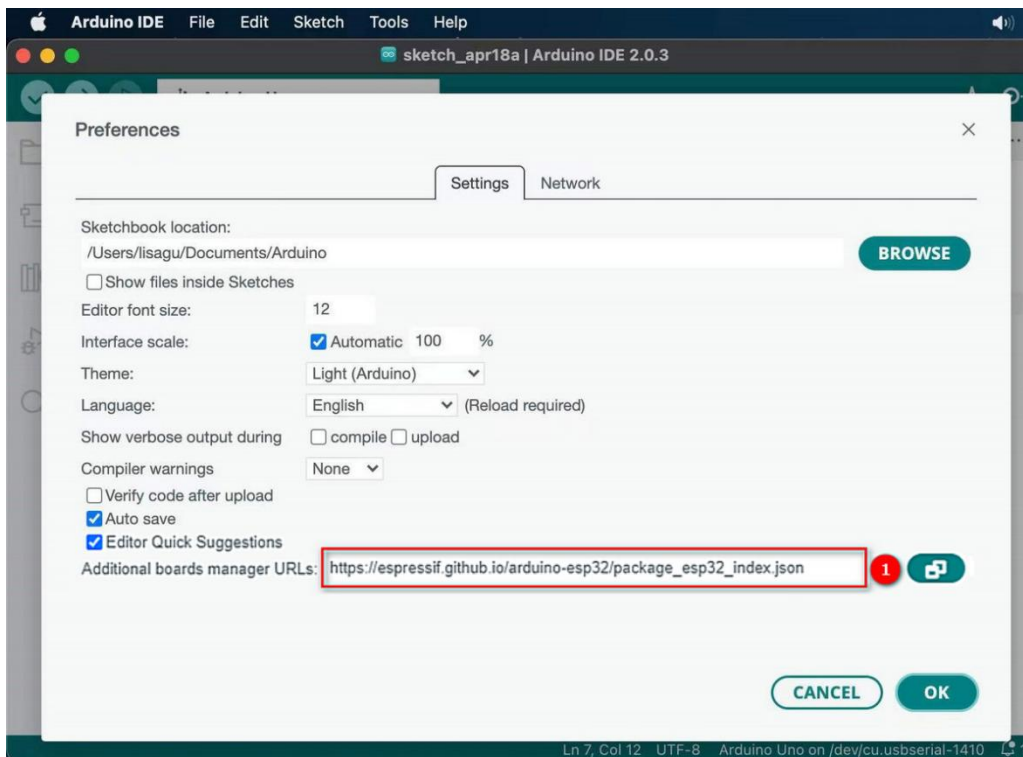


## 2).MAC System

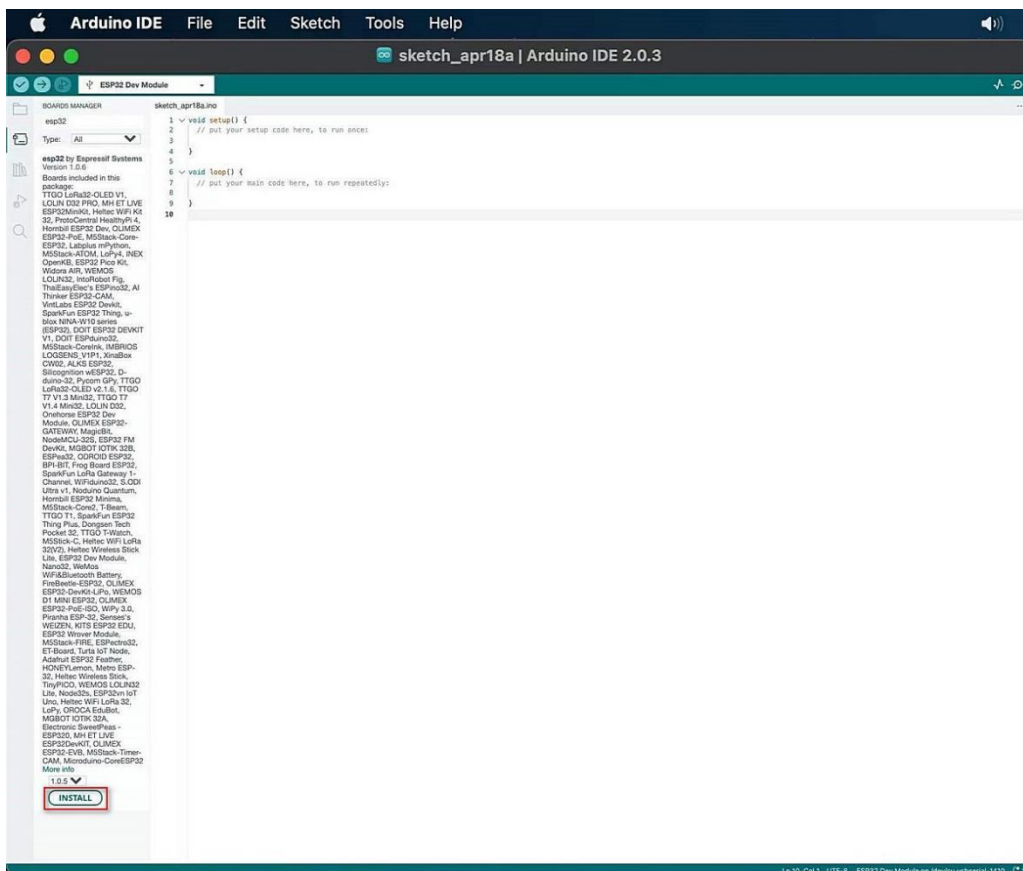
The setting method of Arduino IDE resembles that of Windows. The only difference is COM port:



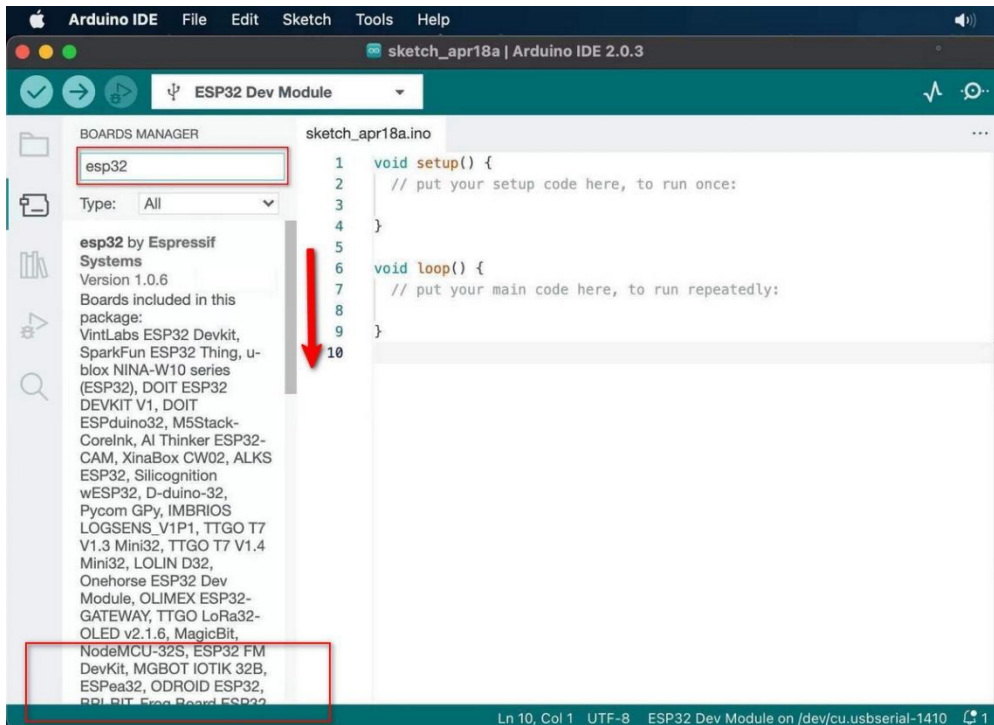
Tap Arduino IDE » Preferences, add the link“  
[https://espressif.github.io/arduino-esp32/package\\_esp32\\_index.json](https://espressif.github.io/arduino-esp32/package_esp32_index.json)” in  
Additional boards manager URLs and click OK.



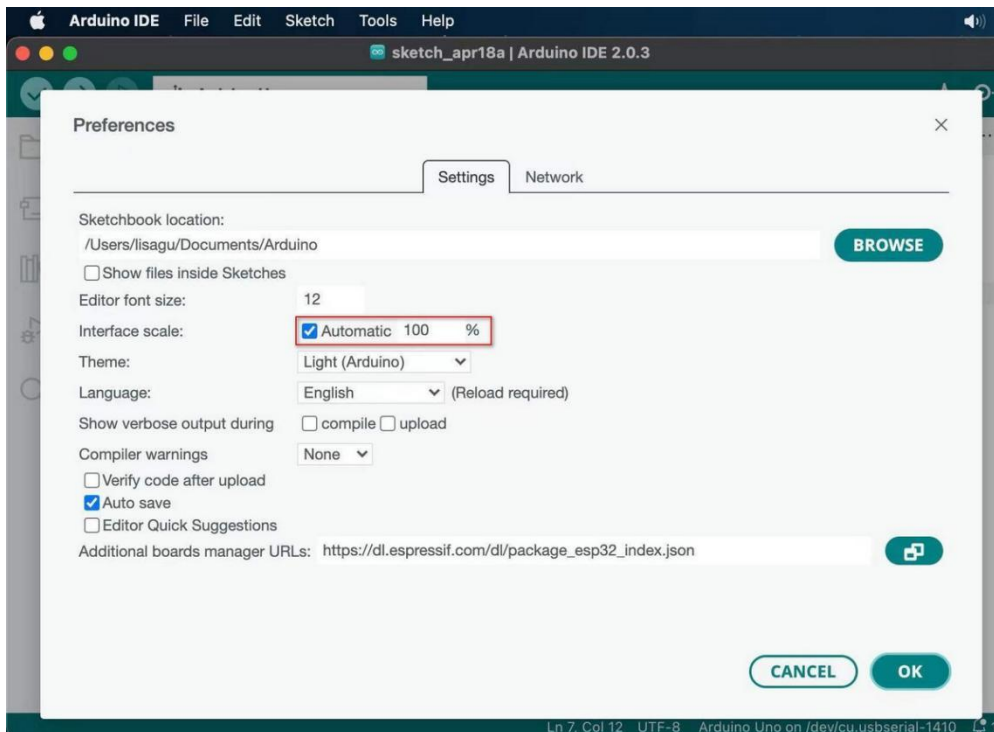
Search for ESP32 In the search box and click Install



If you use an Apple laptop, slid the scroll bar on the right to the bottom, and the **"INSTALL"** installation button can't be displayed, you need to adjust the **"Interface scale"** of the Arduino IDE.



Click **"Arduino IDE » Preferences » Interface scale"** to select the appropriate size until you see the **"INSTALL"** button.



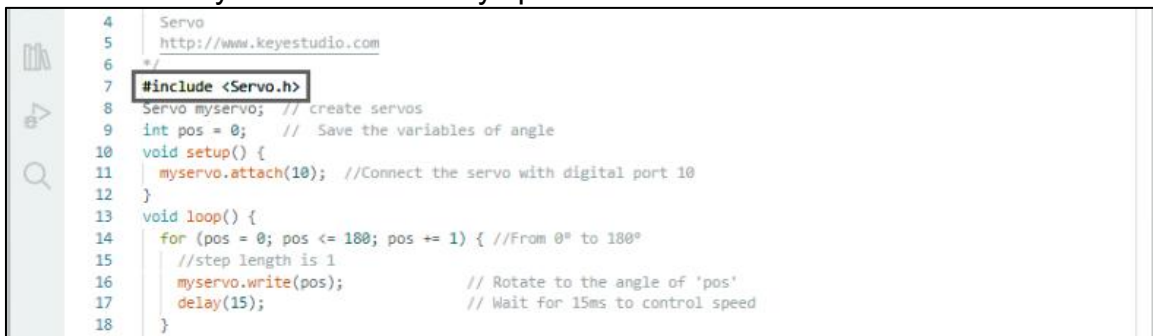
## 6.4 Install Libraries

### 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.

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.



```
4  Servo
5  http://www.keyestudio.com
6  */
7  #include <Servo.h>
8  Servo myservo; // create servos
9  int pos = 0;    // Save the variables of angle
10 void setup() {
11   myservo.attach(10); //Connect the servo with digital port 10
12 }
13 void loop() {
14   for (pos = 0; pos <= 180; pos += 1) { //From 0° to 180°
15     //step length is 1
16     myservo.write(pos);           // Rotate to the angle of 'pos'
17     delay(15);                   // Wait for 15ms to control speed
18   }
```

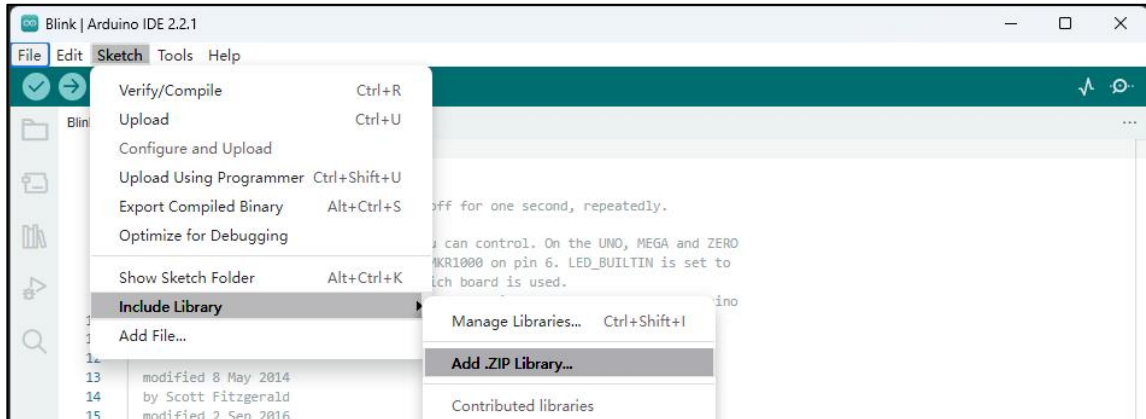
To make the starter kit work, we will need to **add these library files to the Arduino IDE**. You can find them in the tutorial package.



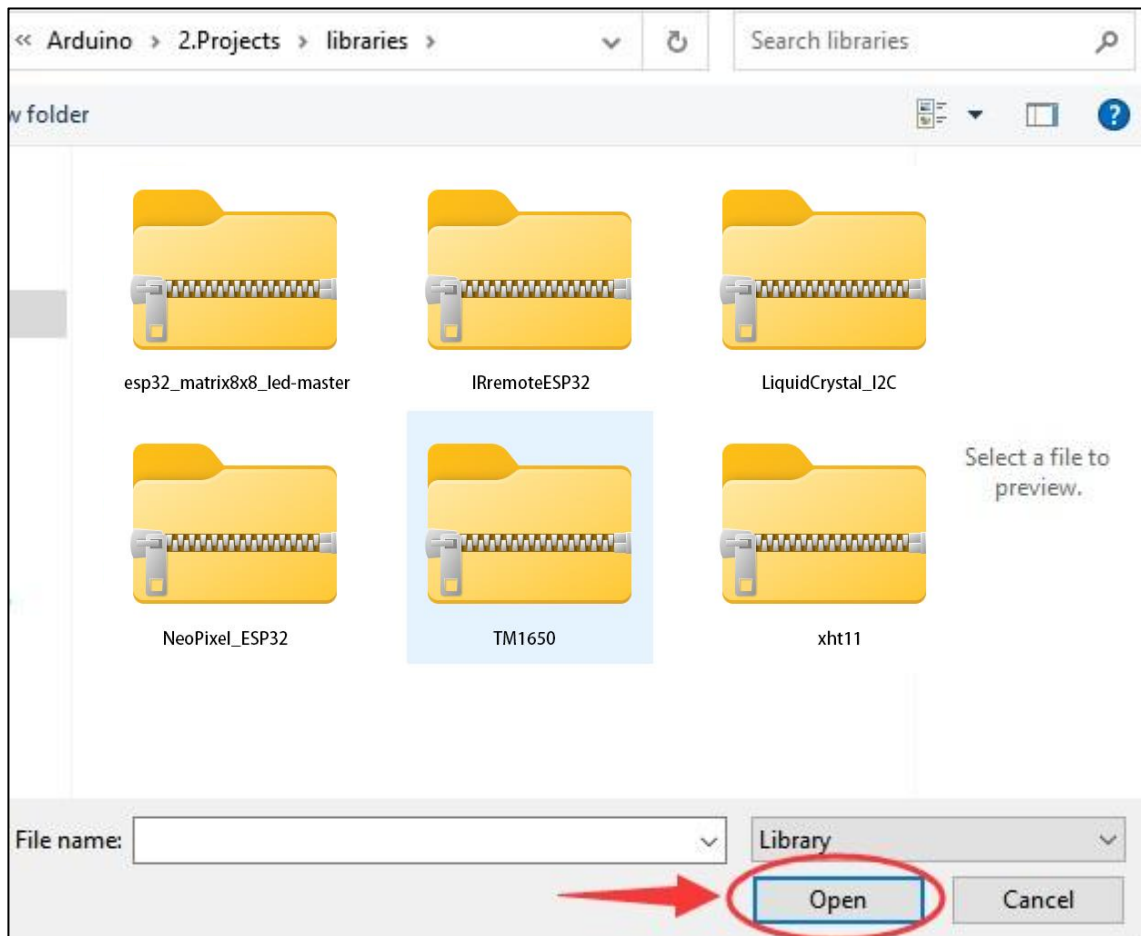
## Importing a .zip Library

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

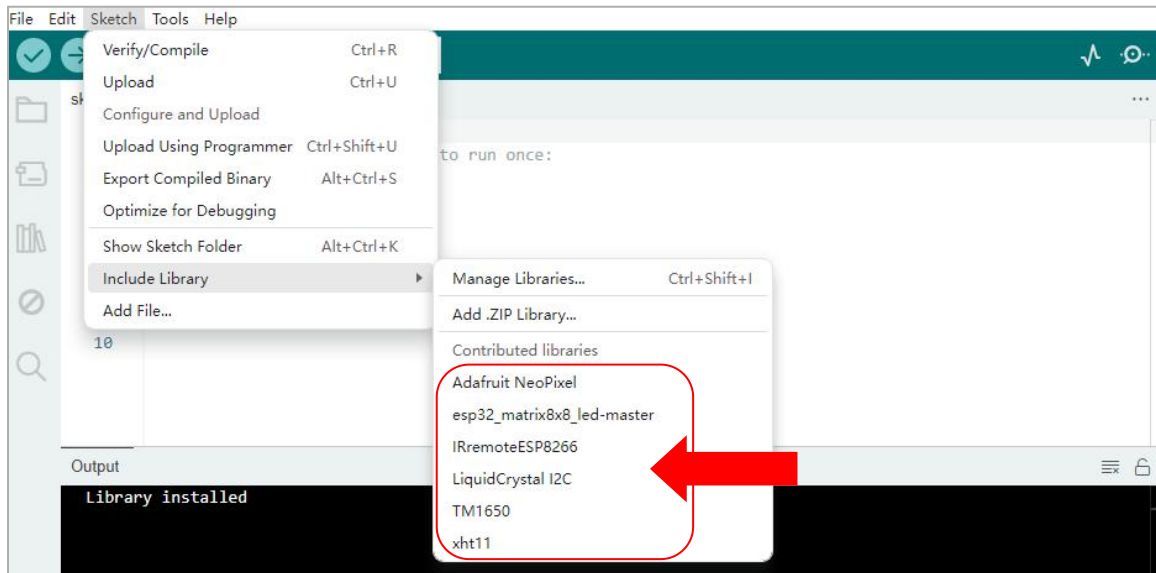
You will be prompted to select the library you want to add.



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.





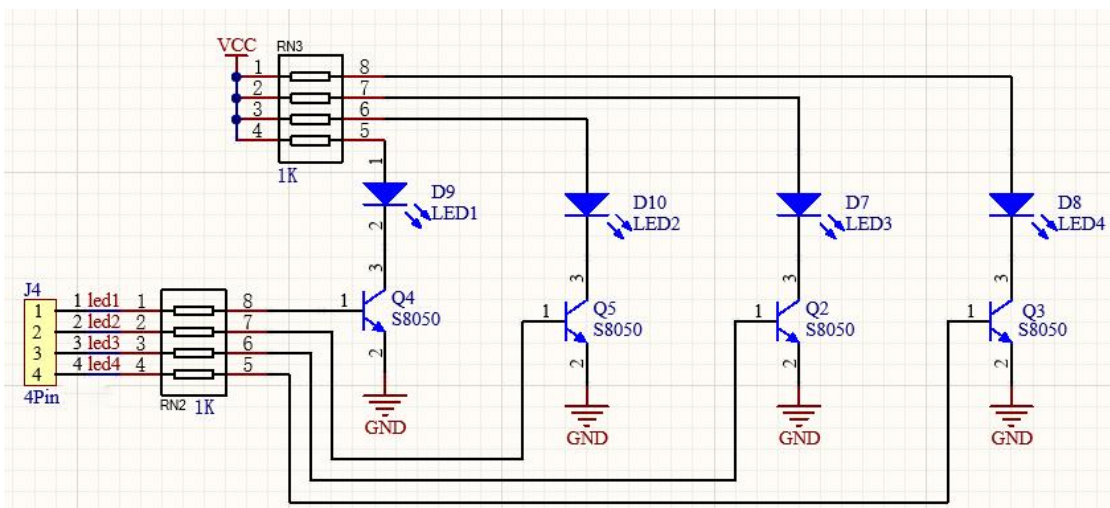
# 7. Arduino Project

## Project 1: LED Blinking

### 1. Description

LED blinking is a simple project designed for starters. You only need to install an LED on Arduino board and upload the code on Arduino IDE. This project reinforces the learning of Arduino conceptual framework and using methods for starters.

### 2. Working Principle

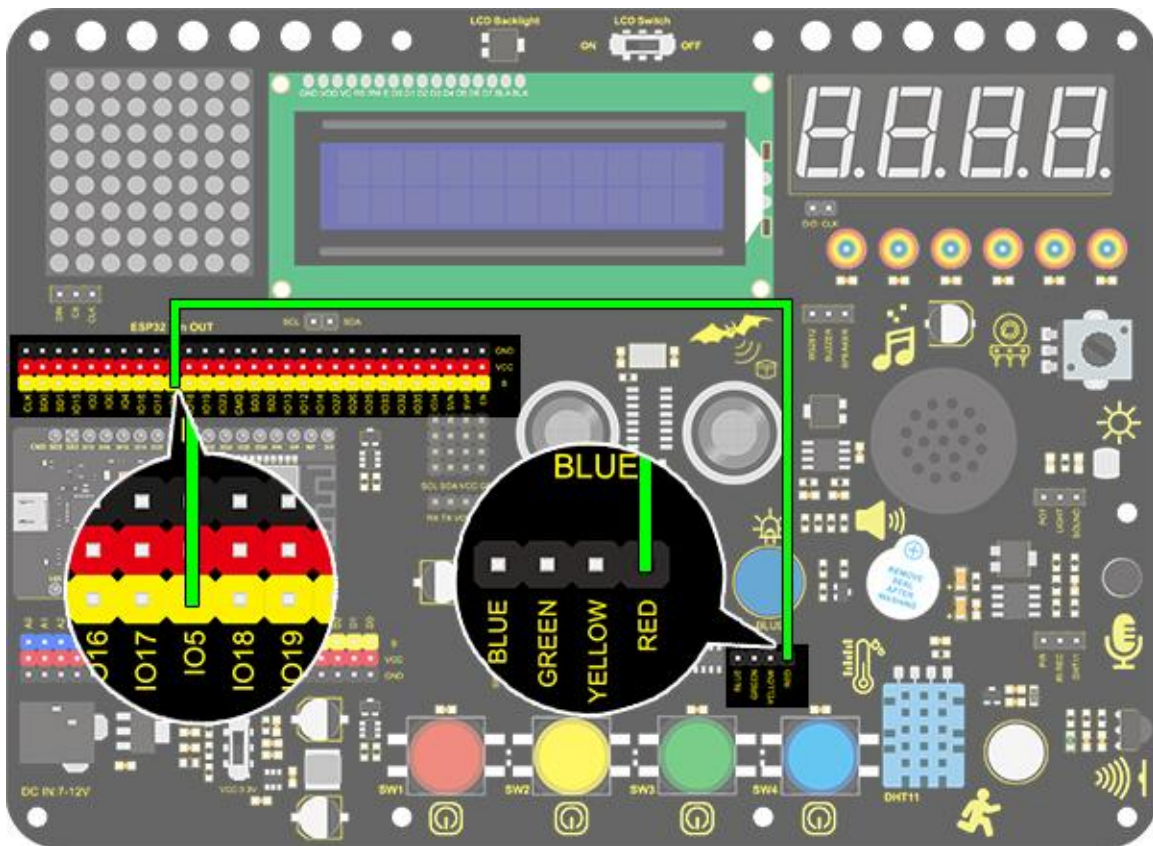


**LED:** The above is the circuit diagram of LED. Generally speaking, limited IO ports of output current may cause low brightness of LED, so a NPN triode (Q2) is applied in circuit as a switch. In this case, the LED will light up if the base(pin 1) of triode is at a high level. On the contrary, LED goes off when the base is at low.

**Triode switch:** To have a clear idea of its principle, certain knowledge of electronic circuit is required. For details, please consult materials by yourself. Briefly, LED on and off rely on the high and low levels of triode base, which are decided by the pin on the development board. LED lights up when the base(pin 1) is at a high level, and it goes off when the base is at low.



### 3. Wiring Diagram



### 4. Upload Code

Use the Arduino IDE to open the .ino format code [project\\_1](#) directly from the tutorial package.

Connect the ESP32 board to the computer with the USB cable.

Select board type "**ESP32 Dev Module**" and select port **COM-XX** (This depends on the number your computer assigns to the ESP32 board, which you can check it in the device manager).

```
/*
  keyestudio ESP32 Inventor Learning Kit
  Project 1: LED Blinking
  http://www.keyestudio.com
*/
int ledPin = 5; //Define LED to connect with pin IO5
void setup() {
  pinMode(ledPin, OUTPUT); //Set the mode to output
}

void loop() {
  digitalWrite(ledPin, HIGH); //Output a high level, LED lights up
  delay(1000); //Delay 1000ms
  digitalWrite(ledPin, LOW); //Output a low level, LED goes off
  delay(1000);
}
```

## 5. Test Result

After uploading the code and powering on, LED will light up for 1s and off for 1 s.

## 6. Code Explanation

**setup()Function:** It is used to initialize variables and pin modes and to enable the library. It runs once only after each time the board powering on or being reset.

**loop()Function:** Followed by setup(), loop()function perpetually executes its code, such as read the pin or output the pin.

**int ledPin = 3:** “int” is a variable within range of -32768 ~ 32767. This example code means we define an variable **ledpin** with an assignment of 5. Therefore, we adopt **ledpin** rather than “5” in later steps, which largely simplifies experimental recordings when considerable sensors and pins are included.

**pinMode(pin,mode):** “pin” is the pin number of mode setting. And the “mode” is optional for INPUT, OUTPUT, and INPUT\_PULLUP. Here we set pin 3 to output mode.

**digitalWrite(pin, value):** “pin” is the digital tube pin of MCU, and here we define as pin 5. “value” is the digital output level (HIGH/LOW).

If we apply pinMode() to set pin to OUTPUT, its voltage should be modified correspondingly. For instance, 5V (it is 3.3V if on a 3.3V-board) corresponds to HIGH, while 0V (GND) is for LOW.

However, if LED links with the pin rather than setting pinMode() to OUTPUT, LED may become dim when recalling digitalWrite(HIGH). This is because digitalWrite() enables the inner pull-up resistor, whose function is similar to a great current-limiting resistor.

**delay(ms) :**It is a delay function and “ms” is the delay time in micro seconds.

For more Arduino grammar explanations, please refer to:  
<https://www.arduino.cc/reference/en/>

## Project 2: Breathing LED

### 1. Description

Arduino breathing led utilizes on-board programmable PWM to output analog waveform. After powering on, LED brightness can be adjusted through duty cycle of the waveform to eventually realize the effect of breathing led.

In this way, ambient light can be simulated by changing LED brightness over time. Also, breathing led can form a colorful mini light to construct a tranquil and warm environment.

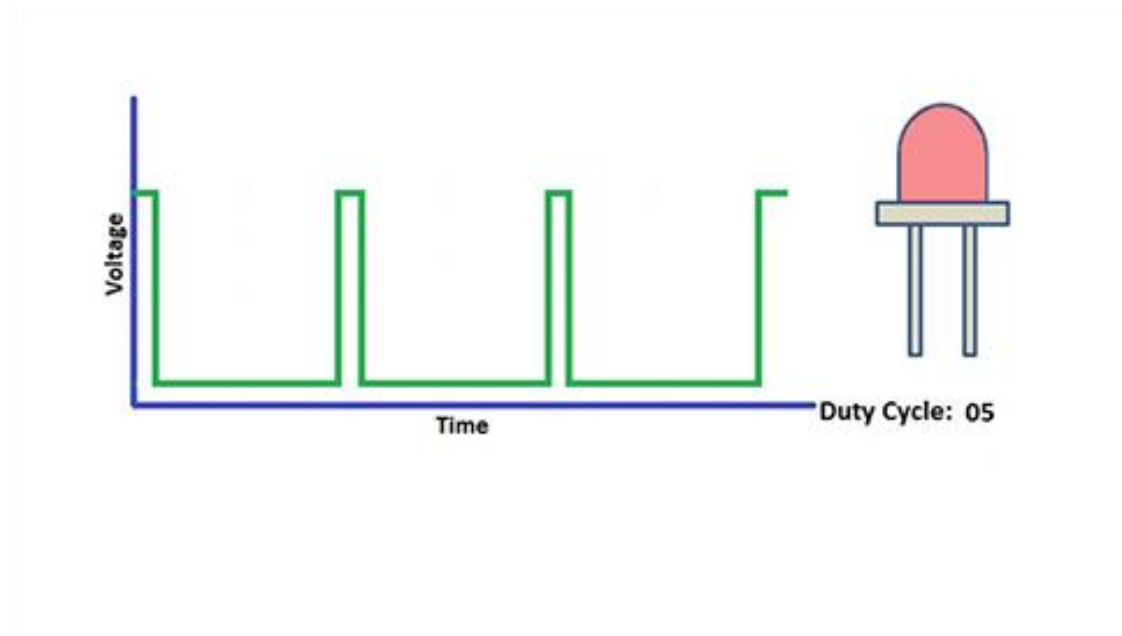
### 2. What is PWM?

PWM controls analog output via digital means, which is able to adjust duty cycle of the wave (a signal circularly shifting between high level and low level).

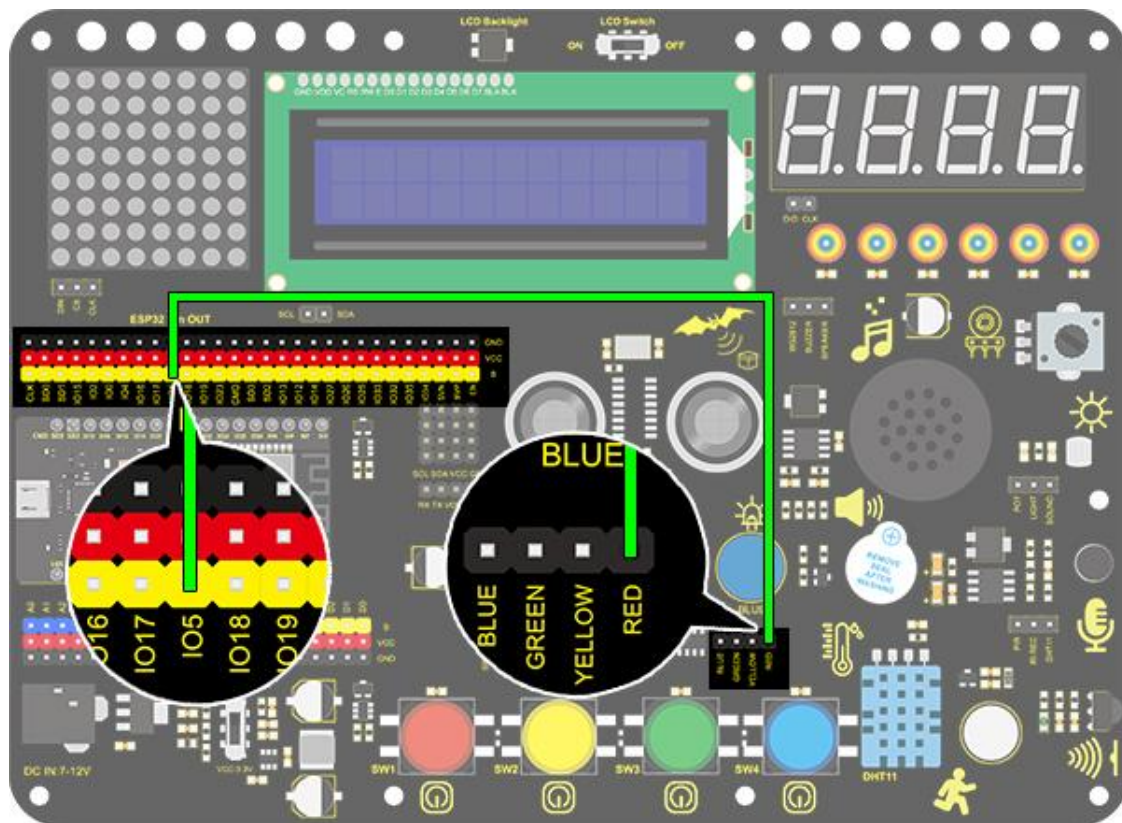
For Arduino, digital ports of voltage output are LOW and HIGH, which respectively correspond to 0V and 5V. Generally, we define LOW as 0 and HIGH as 1. Arduino will output 500 signals of 0 or 1 within 1s. If they are "1", 5V will be output. Oppositely, if they are all 0, the output will be 0V. Or if they are 0101010101..., the average output will be 2.5V.

In other words, output ratio of 0 and 1 affects the voltage value, the more 0 and 1 signals are output per unit time, the more accurate the control will be.

The GPIO34, 35, 36, and 39 of ESP32 cannot use PWM.



### 3. Wiring Diagram



### 4. Upload Code

Use the Arduino IDE to open the .ino format code [project\\_2](#) directly from the tutorial package.

Connect the ESP32 board to the computer with the USB cable.

Select board type "**ESP32 Dev Module**" and select port **COM-XX** (This depends on the number your computer assigns to the ESP32 board, which you can check it in the device manager).

```

/*
  keyestudio ESP32 Inventor Learning Kit
  Project 2: Breathing LED
  http://www.keyestudio.com
*/
#define PIN_LED 5 //define the led pin
#define CHN 0 //define the pwm channel
#define FRQ 1000 //define the pwm frequency
#define PWM_BIT 8 //define the pwm precision
void setup() {
  ledcSetup(CHN, FRQ, PWM_BIT); //setup pwm channel
  ledcAttachPin(PIN_LED, CHN); //attach the led pin to pwm channel
}

void loop() {
  for (int i = 0; i < 255; i++) { //make light fade in
    ledcWrite(CHN, i);
    delay(10);
  }
  for (int i = 255; i > -1; i--) { //make light fade out
    ledcWrite(CHN, i);
    delay(10);
  }
}

```

## 5. Test Result

After uploading the code, we will see the LED slowly brighten and dim, just like the rhythm of breathing.

## 6. Code Explanation

**#define:** It is used to define constants ( unchanged)

**ledcSetup():** It is used to set the frequency and count bits corresponding to the LEDC channel (duty ratio resolution)

The first parameter chan represents the channel number, which ranges from 0 to 15, and can set 16 channels. The high-speed channel (0 ~ 7) is driven by 80MHz clock, and the low-speed channel (8~ 15) is driven by 1MHz clock. The second parameter freq is the desired frequency. The third parameter is the count number of duty ratio resolution, which ranges from 0 to 20. (This value determines the writable value of duty ratio in the ledcWrite method. For example, if the value is 10, the maximum value of duty ratio is 1023, that is,  $(1 \ll \text{bit\_num}) - 1$ ).

**double ledcSetup(uint8\_t chan, double freq, uint8\_t bit\_num)**

**ledcAttachPin():** Its function is to bind the specified LEDC channel to the specified IO port to achieve PWM output.



The pin represents the IO port we want to output, and the channel is the LEDC channel we specify.

```
void ledcAttachPin(uint8_t pin, uint8_t channel);
```

**ledcWrite():** Its function is the output duty cycle of the specified LEDC channel.

The chan is the LEDC channel we specify. The duty means duty cycle, whose range depends on the bit\_num of the ledcSetup() function.

```
void ledcWrite(uint8_t chan, uint32_t duty)
```

**for (int i = 0; i <= 255; i++){ ... }:** It indicates that variable i ranges from 0 to 255, i++ means i increments by 1 each time until the judgment expression i <= 255 is not satisfied. Otherwise, the code in braces is executed for 256 times in total.

**for (int i = 255; i >= 0; i--){ ... },** i-- indicates that i is reduced by 1 each time. If i >= 0 is not satisfied, the for() loop is jumped out, and 256 times are executed in total.

**i++ :** The variable will add 1 per loop

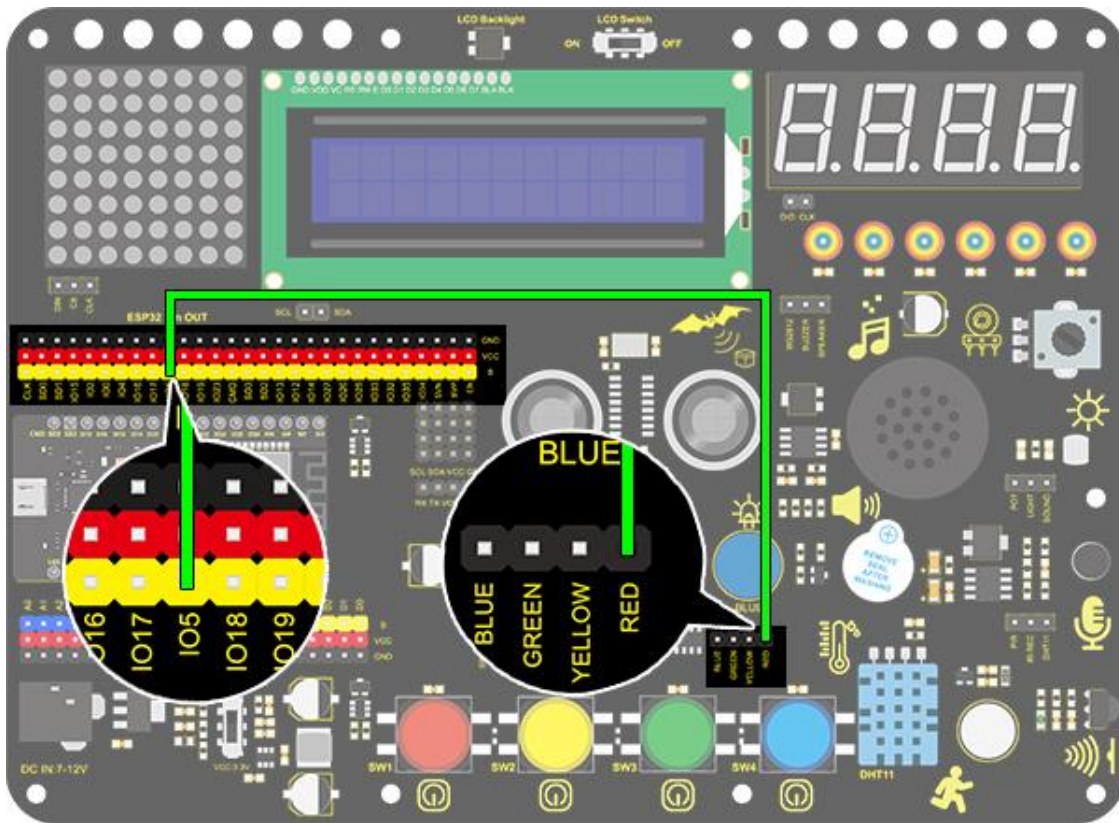
For more details, please refer to the link:<https://www.arduino.cc/reference/en/>

## Project 3: SOS Distress Device

### 1. Description

Arduino SOS device is able to emit distress signals, which coincides with the principle of Morse code. It is convenient for emergencies.

## 2. Wiring Diagram



## 3. Upload Code

Use the Arduino IDE to open the .ino format code [project\\_3](#) directly from the tutorial package.

Connect the ESP32 board to the computer with the USB cable.

Select board type "**ESP32 Dev Module**" and select port **COM-XX** (This depends on the number your computer assigns to the ESP32 board, which you can check it in the device manager).

```

/*
  keyestudio ESP32 Inventor Learning Kit
  Project 3: SOS Distress Device
  http://www.keyestudio.com
*/
int ledPin = 5; //Define pin as I05

void setup() {
  pinMode(ledPin, OUTPUT);
}

void loop() {
  //Three quickly blinks mean "S"
  for(int x=0;x<3;x++){
    digitalWrite(ledPin,HIGH);      //Set LED to light up
    delay(150);                     //Delay 150ms
    digitalWrite(ledPin,LOW);       //Set LED to turn off
    delay(100);                     //Delay 100ms
  }
}

```

## 4. Test Result

After the code is successfully uploaded, we can see that the LED flashes 3 times quickly, then flashes 3 times slowly and then flashes 3 times quickly, alternating between fast and slow.

# Project 4: Traffic Light

## 1. Description

The traffic light module is a device used to control the route of pedestrians and vehicles. It includes a red, a yellow and a green light, which implies different instructions.

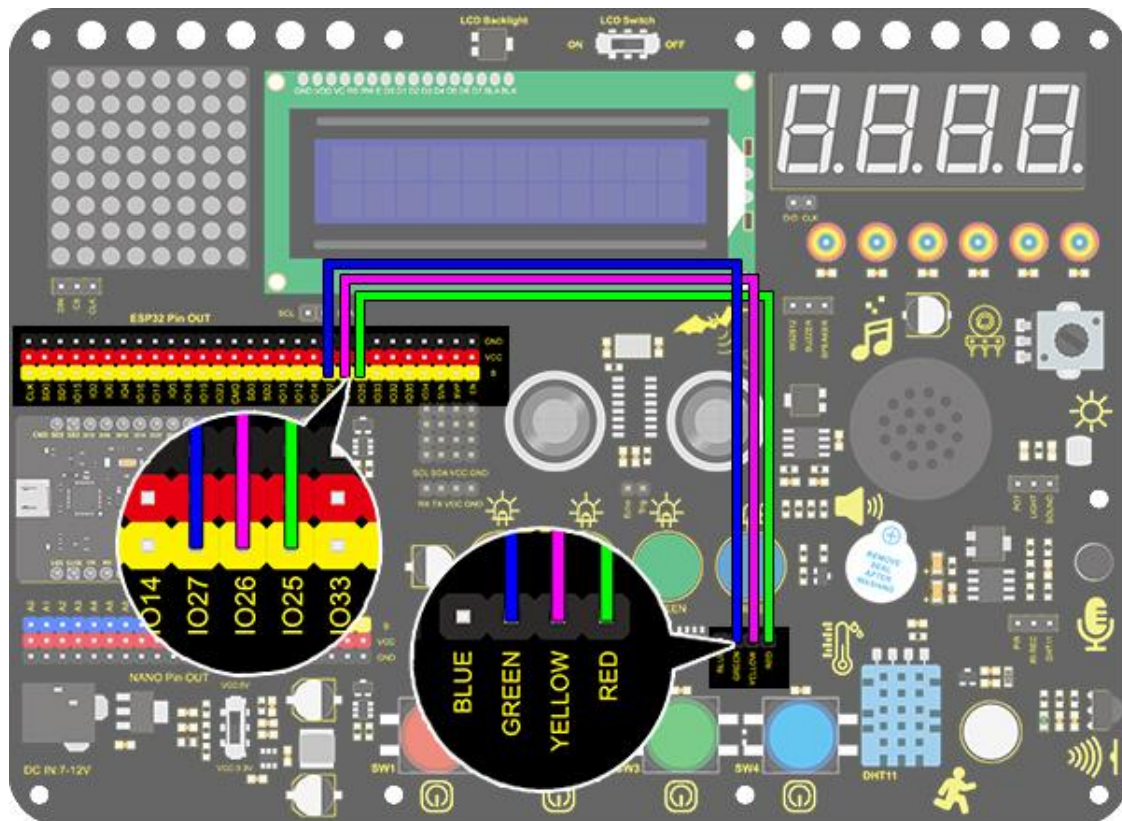
**Red for Stop:** Pedestrians and vehicles stop proceeding.

**Yellow for Caution:** Pedestrians and vehicles are ready for stopping. If the drive is already in process, the speed should be slow.

**Green for Proceed:** Pedestrians and vehicles keep going with the abidance of traffic regulations.

In this project, you can use Arduino to write code to control traffic lights. For instance, set the duration of each lights and the interval time among them. Besides, you may also add a timer to alter light colors to schedule.

## 2. Wiring Diagram



## 3. Upload Code

Use the Arduino IDE to open the .ino format code [project\\_4](#) directly from the tutorial package.

Connect the ESP32 board to the computer with the USB cable.

Select board type "**ESP32 Dev Module**" and select port **COM-XX** (This depends on the number your computer assigns to the ESP32 board, which you can check it in the device manager).

```

/*
  keyestudio ESP32 Inventor Learning Kit
  Project 4 Traffic Light
  http://www.keyestudio.com
*/
int greenPin = 27; //Green LED connects to I027
int yellowPin = 26; //Yellow LED connects to I026
int redPin = 25; //Red LED connects to I025
void setup() {
  //Set all LED interfaces to output mode
  pinMode(greenPin, OUTPUT);
  pinMode(yellowPin, OUTPUT);
  pinMode(redPin, OUTPUT);
}

void loop() {
  digitalWrite(greenPin, HIGH); //Light green LED up
  delay(5000); //Delay 5s
}

```

#### 4. Test Result

After uploading the code, green LED will light up for 5s, yellow LED will blink for 3 times, and red LED will light up for 5s, in circulation.

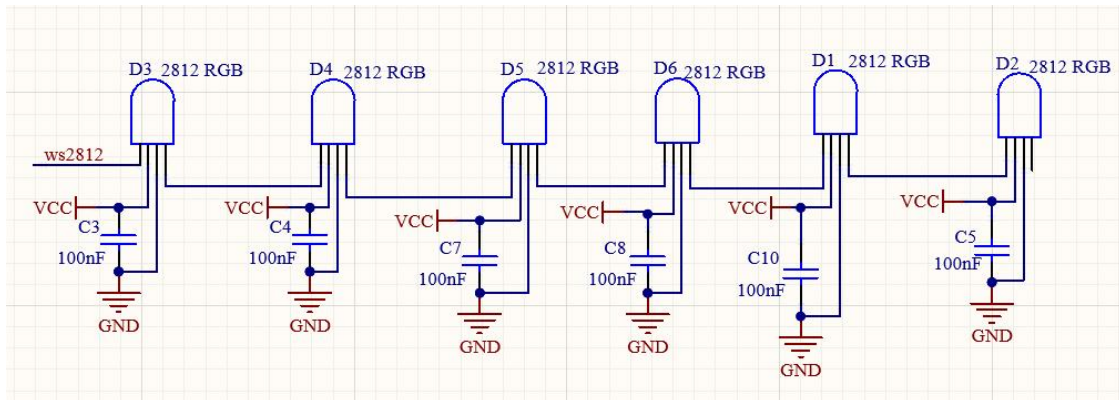
## Project 5: Rainbow Ambient Light

### 1. Description

Arduino 2812RGB LED is a programable colorful dreamy light, whose color, brightness and rhythm are adjustable. This rainbow ambient light can be used as a dynamic decoration at will. Or you may control it to "dance with music". Importantly, it can be improved as an alarm. Its built-in sensor detects the ambient surroundings to warn users by changing its color, brightness and rhythm.



## 2. Working Principle

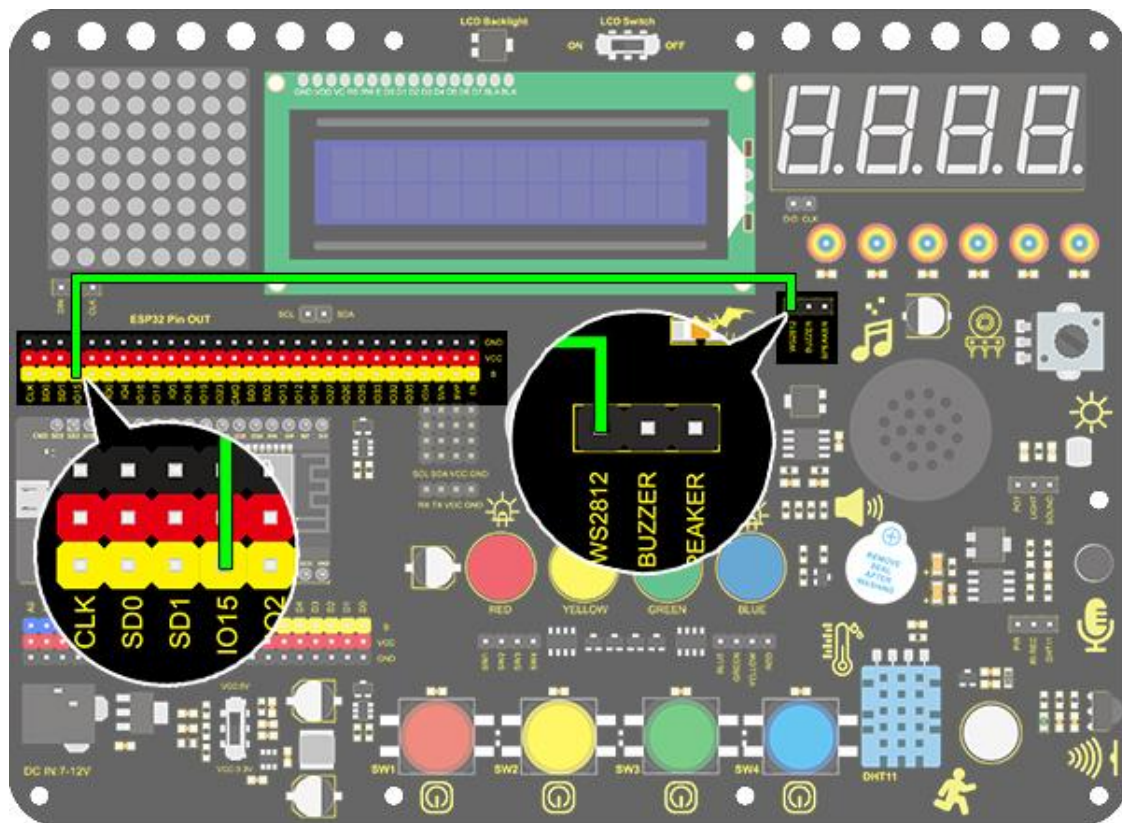


The data protocol adopts communication mode of single-line return-to-zero code. After the pixel is reset on power, DIN terminal receives data from the controller. The firstly arriving 24bit data will be extracted by the first pixel and be sent to the inner data register.

Remaining data will be amplified by an amplification circuit and be transmitted through DOUT port to the next cascaded pixel. Being transmitted through pixels, the signal decreases 24bit each time.

Besides, The pixel adopts automatic shaping and forwarding technology, insomuch that the cascade number of the pixel is only limited by the signal transmission speed.

### 3. Wiring Diagram



### 4. Upload Code

Before uploading the code, please ensure the `<NeoPixel_ESP32.h>` library file is loaded to arduino IDE.

Use the Arduino IDE to open the .ino format code [project\\_5.1](#) directly from the tutorial package.

Connect the ESP32 board to the computer with the USB cable.

Select board type "**ESP32 Dev Module**" and select port **COM-XX** (This depends on the number your computer assigns to the ESP32 board, which you can check it in the device manager).

```

/*
  keyestudio ESP32 Inventor Learning Kit
  Project 5.1 Rainbow Ambient Light
  http://www.keyestudio.com
*/
//Add 2812RGB library file
#include <NeoPixel_ESP32.h>

#define PIN 15

Adafruit_NeoPixel strip = Adafruit_NeoPixel(6, PIN); //Defines the instance strip and assigns the RGB

void setup() {
  strip.begin();          //Activate RGB LED
  strip.show(); // Refresh the display
}

void loop() {
  strip.setPixelColor(0, strip.Color(255, 0, 0)); //The frist RGB LED is red
  strip.setPixelColor(1, strip.Color(0, 255, 0)); //The second RGB LED is green

```

## 5. Test Result

After uploading the code and powering on, the LED will light up in different colors.

From left to right:

The first RGB LED is red

The second RGB LED is green

The third RGB LED is blue

The fourth RGB LED is yellow

The fifth RGB LED is purple

The sixth RGB LED is white



## 6. Extended Code

Specifically, we replace RGB value with variables. And then we control these variables to form an expected light show.

The wirings remain unchanged.

Use the Arduino IDE to open the .ino format code [project\\_5.2](#) directly from the tutorial package.

Connect the ESP32 board to the computer with the USB cable.

Select board type "**ESP32 Dev Module**" and select port **COM-XX** (This depends on the number your computer assigns to the ESP32 board, which you can check it in the device manager).

```
/*
  keyestudio ESP32 Inventor Learning Kit
  Project 5.2 Rainbow Ambient Light
  http://www.keyestudio.com
*/
//Add 2812RGB library file
#include <NeoPixel_ESP32.h>

#define PIN 15

Adafruit_NeoPixel strip = Adafruit_NeoPixel(6, PIN, NEO_GRB + NEO_KHZ800);

void setup() {
  strip.begin();
  strip.show(); // Initialize all pixels to 'off'
}

void loop() {
  // Some example procedures showing how to display to the pixels:
  colorWipe(strip.Color(255, 0, 0), 50); // Red
  colorWipe(strip.Color(0, 255, 0), 50); // Green
  colorWipe(strip.Color(0, 0, 255), 50); // Blue
  // Send a theater pixel chase in...
  theaterChase(strip.Color(127, 127, 127), 50); // White
  theaterChase(strip.Color(127, 0, 0), 50); // Red
  theaterChase(strip.Color(0, 0, 127), 50); // Blue
}
```

## 5. Test Result

After uploading the code and powering on, the LED will light up in different colors and make a light show.

## 7. Code Explanation

**#include <NeoPixel\_ESP32.h>** : Libraries are included, so that codes in library can be directly recalled.

**Adafruit\_NeoPixel strip = Adafruit\_NeoPixel(6, PIN);** Define an instance strip and set the number of RGB. Here we input 6.

PIN = 15

**strip.begin();** Initialize 2812RGB

**strip.setPixelColor(uint16\_t n, uint8\_t color);** The uint16\_t n is used to set the number of 2812RGB and the second parameter is the value of the displayed color.

**strip.Color(uint8\_t red , uint8\_t green , uint8\_t blue);**Set the color function. The value range of the three parameters( red, green and blue) is (0-255). We can synthesize various colors by setting the values of the three colors.

**strip.show();** Display 2812RGB

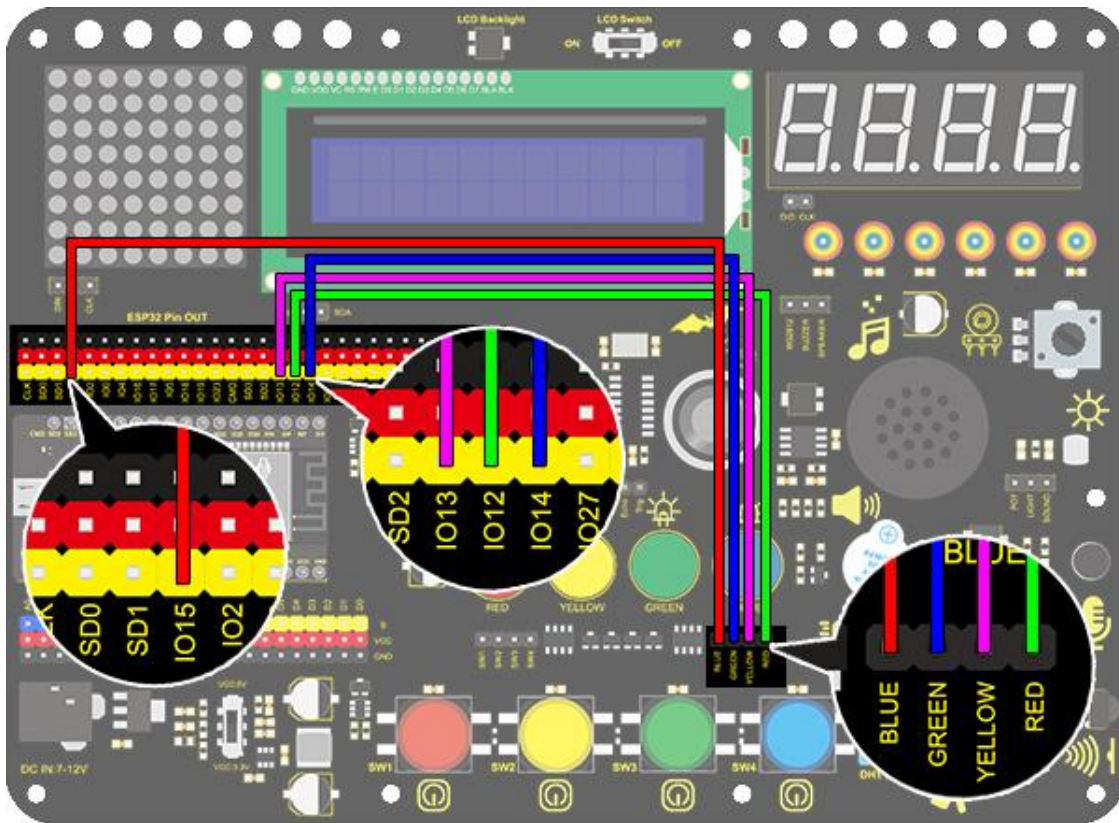
## Project 6: Water Flow Light

### 1. Description

This simple water flow light project enables to help you learn electronic packaging. In this project, we will control LEDs to change the color in a specified speed via a Arduino board.



## 2. Wiring Diagram



## 3. Upload Code

Use the Arduino IDE to open the .ino format code [project\\_6](#) directly from the tutorial package.

Connect the ESP32 board to the computer with the USB cable.

Select board type "**ESP32 Dev Module**" and select port **COM-XX** (This depends on the number your computer assigns to the ESP32 board, which you can check it in the device manager).

```

/*
  keyestudio ESP32 Inventor Learning Kit
  Project 6 Water Flow Light
  http://www.keyestudio.com
*/
void setup() {
  for(int i = 12; i <= 15 ;i++){    //Use "for" loop statement to set IO12-IO15 pin to output mode
    pinMode(i,OUTPUT);
  }
}

void loop() {
  for(int i = 12; i <= 15; i++){    //Use "for" loop statement to light up LED on IO12-IO15
    digitalWrite(i,HIGH);
    delay(200);
    digitalWrite(i,LOW);
  }
  for(int i = 15; i >= 12; i--){    //Use "for" loop statement to light up LED on IO15-IO12
    digitalWrite(i,HIGH);
    delay(200);
    digitalWrite(i,LOW);
  }
}

```

#### 4. Test Result

After uploading code and powering on, the LEDs go from left to right and then from right to left

#### 6. Code Explanation

**for(int i = 12;i <= 15 ;i++){ pinMode(i,OUTPUT); }** : We can use "for" statement to define continuous pins. Yet it features a disadvantage of non-replacement ability of pins, which deteriorates the code portability.

```

for(int i = 12; i <= 15; i++){
  digitalWrite(i,HIGH);
  delay(200);
  digitalWrite(i,LOW);
}

```

In the first loop, LED on IO12 pin will light up and off after a 200ms delay. At the second time, LED on IO13 pin will turn on and off also after a 200ms delay. Until the IO15 pin is extinguished and the for loop is popped out, and the second for loop is the same except from IO15 pin to IO12 pin.

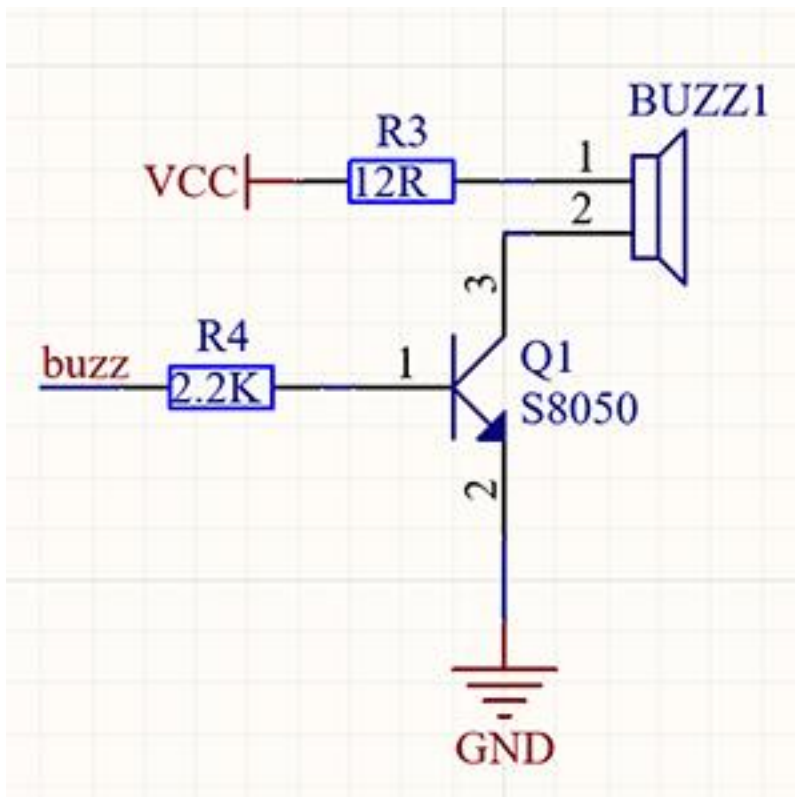
## Project 7: Active Buzzer

### 1. Description

An active buzzer is a component used as an alarm, a reminder or an entertaining device, which boasts a reliable sound.

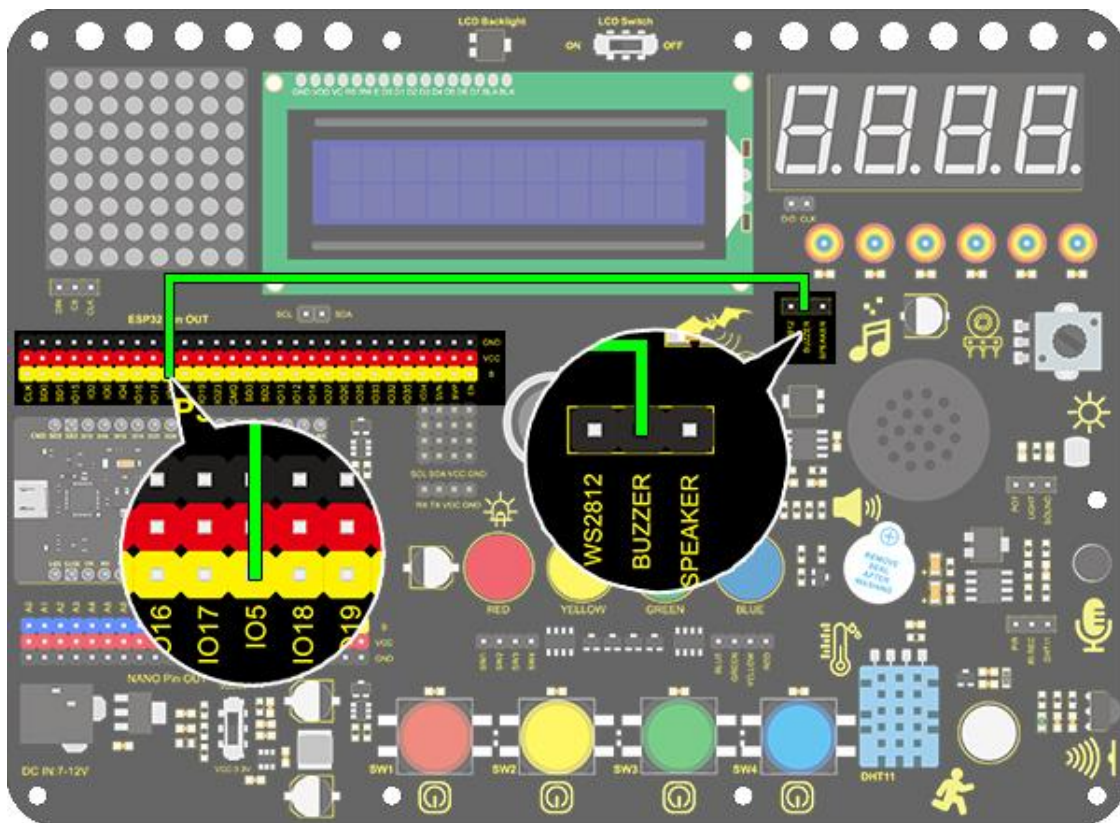
What's more, it empowers to stimulate highly controllable sounds, making our projects more interesting.

### 2. Working Principle



An active buzzer integrates a multi-vibrator, so it makes sound only via DC voltage. Pin 1 of the buzzer connects to VCC and pin 2 is controlled by a triode. When a high level is provided for the base (pin 1) of the triode, its collector (pin 3) and emitter (pin 2) link to GND, and then the buzzer emits sound. Oppositely, if we offer a low level to the base, the rest of pins will be disconnected, so the buzzer will remain quiet.

### 3. Wiring Diagram



### 4. Upload Code

If the development board outputs a high level, the buzzer will emit sound. If it outputs a low level, the buzzer will stop ringing.

Use the Arduino IDE to open the .ino format code [project\\_7](#) directly from the tutorial package.

Connect the ESP32 board to the computer with the USB cable.

Select board type "**ESP32 Dev Module**" and select port **COM-XX** (This depends on the number your computer assigns to the ESP32 board, which you can check it in the device manager).

```
/*
  keyestudio ESP32 Inventor Learning Kit
  Project 7 Active Buzzer
  http://www.keyestudio.com
*/
int buzzer = 5; //Define buzzer connected to I05 pin
void setup() {
  pinMode(buzzer, OUTPUT); //Set the output mode
}

void loop() {
  digitalWrite(buzzer, HIGH); //I05 pin outputs a high level to cause the buzzer to emit sound
  delay(1000); //Delay 1000ms
  digitalWrite(buzzer, LOW); //I05 outputs a low level to prevent the buzzer to emit sound
  delay(1000);
}
```

## 5. Test Result

After uploading code and powering on, the buzzer emits sound for 1s and stays quiet for 1s.

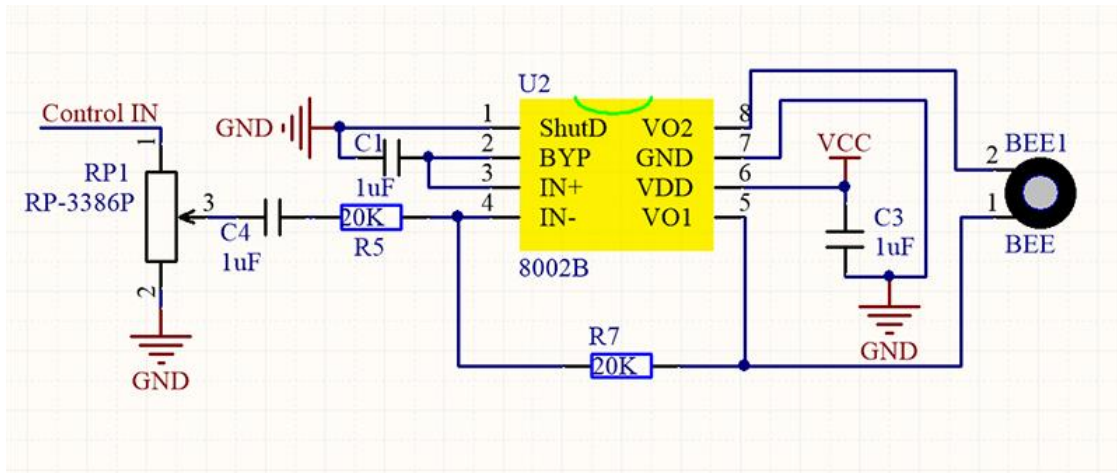
# Project 8: Music Performer

## 1. Description

In this project, we will use a power amplifier speaker to play music. This speaker can not only play simple songs, but also perform what you desire. Thus, you can program other interesting codes in the project to accomplish splendid learning outcomes.



## 2. Working Principle



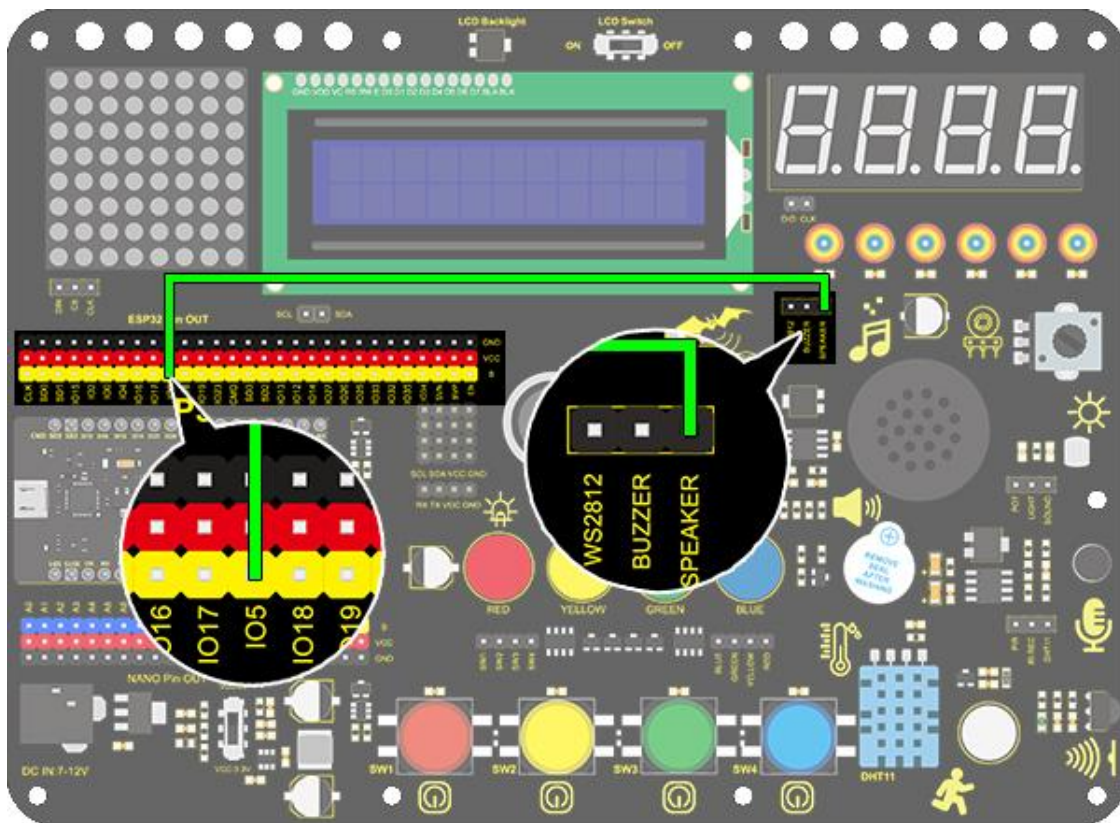
The electrical signal is input from pin 1 of RP1 (adjusts signal intensity, which is also the sound volume).

After coupling in C4 and passing R5, the signal reaches IN- pin of 8002B, in which it is operationally amplified and output to BEE1 speaker.

### Frequency Comparison Table in C

Note	Frequency(Hz)	Note	Frequency(Hz)	Note	Frequency(Hz)
Flat 1	262	Natural 1	523	Sharp 1	1047
Do		Do		Do	
Flat 2	294	Natural 2	587	Sharp 2	1175
Re		Re		Re	
Flat 3	330	Natural 3	659	Sharp 3	1319
Mi		Mi		Mi	
Flat 4	349	Natural 4	698	Sharp 4	1397
Fa		Fa		Fa	
Flat 5	392	Natural 5	784	Sharp 5	1568
So		So		So	
Flat 6	440	Natural 6	880	Sharp 6	1760
La		La		La	
Flat 7	494	Natural 7	988	Sharp 7	1967
Si		Si		Si	

### 3. Wiring Diagram



### 4. Upload Code

According to the comparison table, we set a pin to output mode. And we use function "tong(Pin , frequency);" to generate square waves in certain frequency to emit corresponding sound. Finally, the notes will be output after adding a delay time.

Use the Arduino IDE to open the .ino format code [project\\_8.1](#) directly from the tutorial package.

Connect the ESP32 board to the computer with the USB cable.

Select board type "**ESP32 Dev Module**" and select port **COM-XX** (This depends on the number your computer assigns to the ESP32 board, which you can check it in the device manager).

```

/*
  keyestudio ESP32 Inventor Learning Kit
  Project 8.1 Music Performer
  http://www.keyestudio.com
*/
int beepin = 5; //Define the speaker pin to I05
void setup() {
  pinMode(beepin, OUTPUT); //Define the I05 port to output mode
}

void loop() {
  tone(beepin, 262); //Flat DO plays 500ms
  delay(500);
  tone(beepin, 294); //Flat Re plays 500ms
  delay(500);
  tone(beepin, 330); //Flat Mi plays 500ms
  delay(500);
  tone(beepin, 349); //Flat Fa plays 500ms
  delay(500);
  tone(beepin, 392); //Flat So plays 500ms
  delay(500);
  tone(beepin, 440); //Flat La plays 500ms
  delay(500);
  tone(beepin, 494); //Flat Si plays 500ms
  delay(500);
  noTone(beepin); //Stop for 1s
  delay(1000);
}

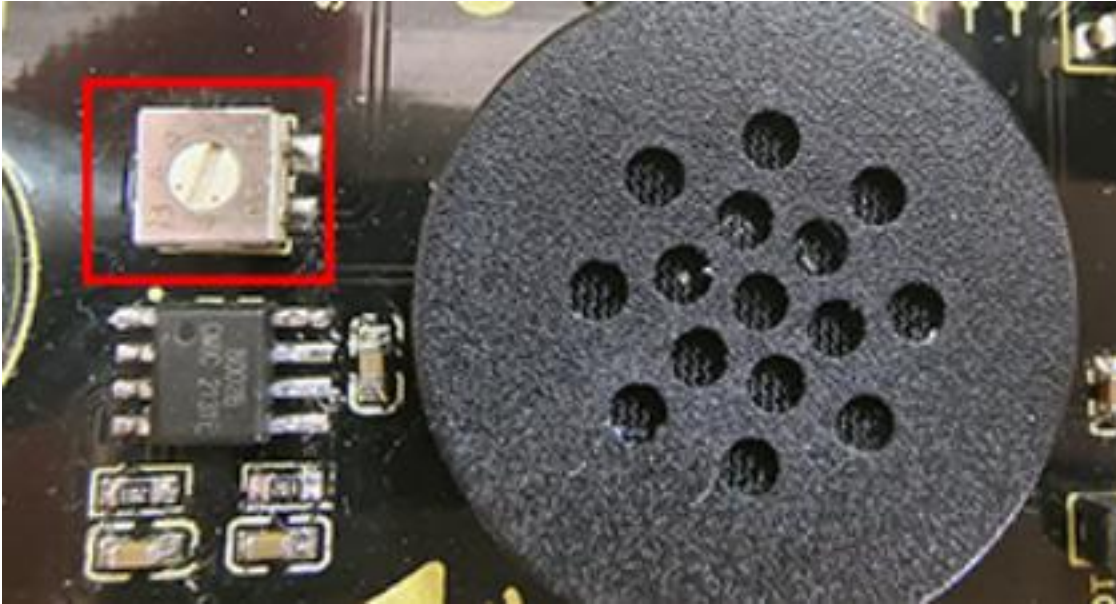
```

## 5. Test Result

After uploading code and powering on, the amplifier circularly plays music tones with corresponding frequency: DO, Re, Mi, Fa, So, La, Si.

### Power amplifier sound adjustment:

**There is a potentiometer next to the speaker. We can adjust the sound of the speaker by twisting it.** (Note: Please use appropriate strength to adjust it, so as not to break the potentiometer)



## Knowledge Expansion

Let's play a birthday song. The wirings remain unchanged.

### Numbered musical notation:

1=G  $\frac{3}{4}$  Happy Birthday to You

5	5		6	5	i		7	0	5	5		6	5	2		i	-	5	5		5	3	i	
Happy	birth-	day	to	you!	Happy	birth-day	to	you!	Happy	birth-	th	day												

7	6	-		6	0	4	4		3	i	2	i	
to	my	dear		Happy	birth	day	to	you					

## Comparison Diagram of Flat, Natural and Sharp

Double flat	Flat	Natural	Sharp	Double sharp
1 2 3 4 5 6 7	1 2 3 4 5 6 7	1 2 3 4 5 6 7	1 2 3 4 5 6 7	1 2 3 4 5 6 7

Use the Arduino IDE to open the .ino format code [project\\_8.2](#) directly from the tutorial package.

Connect the ESP32 board to the computer with the USB cable.

Select board type "**ESP32 Dev Module**" and select port **COM-XX** (This depends on the number your computer assigns to the ESP32 board, which you can check it in the device manager).

```
/*
  keyestudio ESP32 Inventor Learning Kit
  Project 8.2 Music Performer
  http://www.keyestudio.com
*/
int beepin = 5; //Define the speaker pin to I05
// do, re, mi, fa, so, la, si
int doremi[] = {262, 294, 330, 370, 392, 440, 494,      //Falt 0-6
               523, 587, 659, 698, 784, 880, 988,      //Natural 7-13
               1047,1175,1319,1397,1568,1760,1967};    //Sharp 14-20
int happybirthday[] = {5,5,6,5,8,7,5,5,6,5,9,8,5,5,12,10,8,7,6,11,11,10,8,9,8}; //Find the number in
int meter[] = {1,1,2,2,2,4, 1,1,2,2,2,4, 1,1,2,2,2,2,2, 1,1,2,2,2,4}; // Beats

void setup() {
  pinMode(beepin, OUTPUT); //Set I05 pin to output mode
}

void loop() {
  for( int i = 0 ; i <= 24 ;i++){ //i<=24, because there are only 24 tones in this song
    //Use tone()function to generate a waveform in "frequency"
    tone(beepin, doremi[happybirthday[i] - 1]);
    delay(meter[i] * 200); //Wait for 1000ms
    noTone(beepin); //Stop singing
  }
}
```

## 7. Code Explanation

### **doremi[] { ... };**

Linear array is used to store data, which generally are considered as a series of variables of the same type.

Analogically, data are neatly put in ordered boxes, so that we can take the sequenced numbers to use corresponding data.

### **tone(pin, frequency);**

"pin" is the arduino pin generating tones in a total of 6 pins. "frequency" is the note frequency in the unit of Hz.

**unsigned int** is the data type within range of 0 ~ 65, 535 ((2<sup>16</sup>) - 1).

7. "tone" function controls the module to generate square waves in certain frequency(duty cycle of 50%). It sings until "noTone()"(Stop to sing) is activated.
8. Tones can be emitted by connecting the pin to a piezoelectric buzzer or other speakers.

9. For each time, tone() generates only one type of tone. Thus, if a tone is played on some pins, this function will be invalid.
10. tone() function disturbs the PWM output on pin 3 and pin 11 (on any board excluding Mega).
11. The sound frequency generated by tone() must be more than 31Hz. So when you play tones in different frequency on numerous pins, noTone() is necessary on one pin and followed by tone() on next pin.

**noTone(beeppin);**

It stops the tone generation(stops singing). You can directly add the pin number.

## **Project 9: Digital Tube Display**

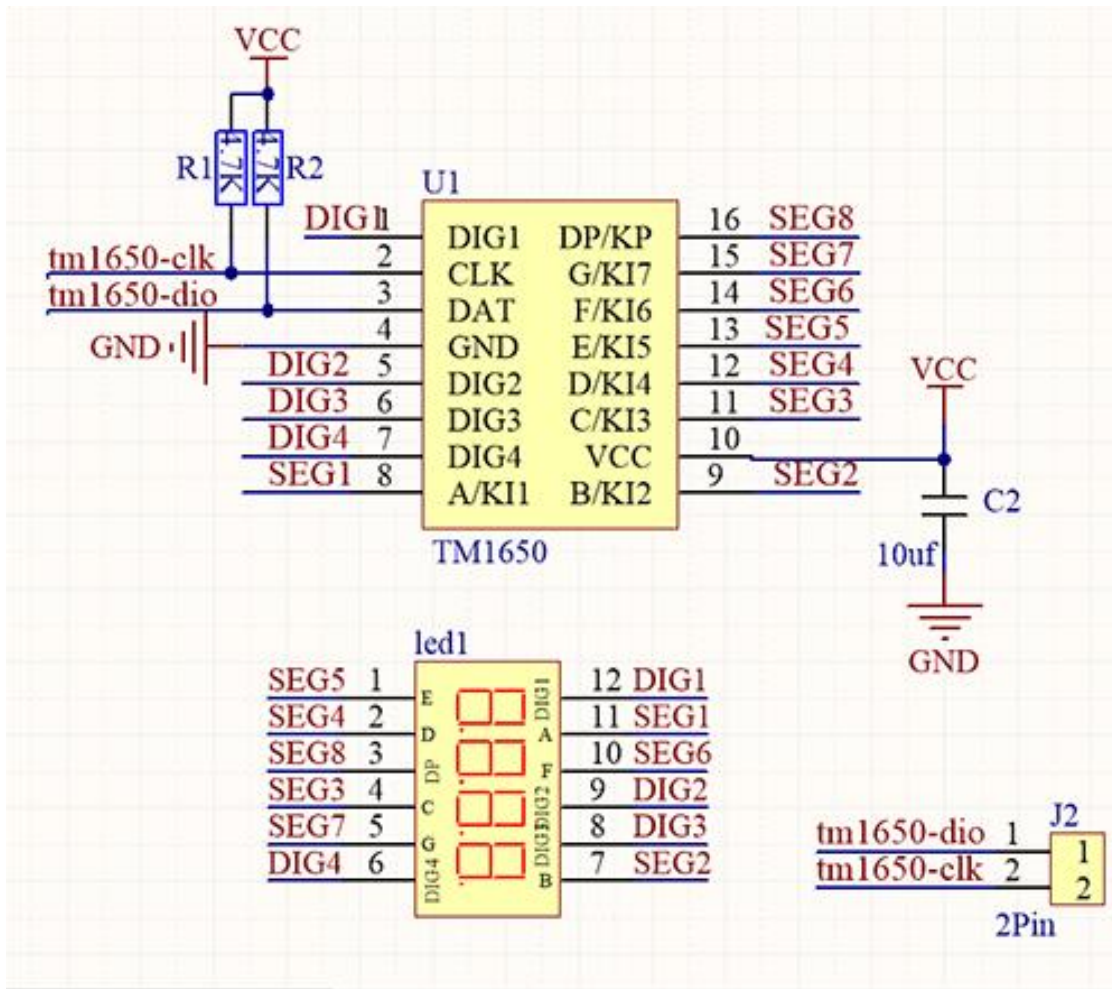
### **1. Description**

This 4-Digit tube display is a device used to display counting or time, which is able to display numbers from 0 ~ 9 and simple letters. It consists of four digital tubes, each of which has seven light-emitting diodes (LED).

Moreover, multiple functions can be realized by connecting their pins to the Arduino development board, such as timekeeping and some game storing.

### **2. Working Principle**





TM1650 utilizes IIC protocol and adopts two bus lines (SDA and SCL).

**Data Command:** 0x48.

This command tells TM1650 to light up the digital tubes rather than key scanning.

**Display Command:**

B7	B6	B5	B4	B3	B2	B1	B0	Function	Description
×	0	0	0		×	×		Luminance Settings	Level 8 brightness
×	0	0	1		×	×			Level 1 brightness
×	0	1	0		×	×			Level 2 brightness
×	0	1	1		×	×			Level 3 brightness
×	1	0	0		×	×			Level 4 brightness
×	1	0	1		×	×			Level 5 brightness
×	1	1	0		×	×			Level 6 brightness
×	1	1	1		×	×			Level 7 brightness
×				0	×	×		Segment 8/7 display mode	Segment 8 display mode
×				1	×	×			Segment 7 display mode
×					×	×	0	On/off display bit	Display OFF
×					×	×	1		Display ON

Actually, it is one byte of data with different bits representing different functions.

**bit[6:4]:** Set the brightness of LED. Note that 000 indicates the brightest.

**bit[3]:** Determine whether there is a decimal dot.

**bit[0]:** Determine whether to turn on the display.

### Digital Tube Turns on

Take an example: Level 8 brightness without a dot signifies 0x05.

Steps: Starting signal — Send 0x48 — Slave-device receives — Send 0x05 — Slave-device receives — Ending signal

After turning on, there is no need to repeatedly send 0x48, as the function of digital tube has confirmed.

Besides, the brightness and display methods can be enumerated with multiple data in one place, so that it is clear and space-saving.

### Digital Tube Turns off

Steps: Starting signal — Send 0x48 — Slave-device receives — Send 0x00 — Slave-device receives — Ending signal

### Digital Tube Displays Numbers

We firstly tell TM1650 to display numbers on the predetermined tube. And then the number will be displayed. Its eight bit corresponds to eight segment, with 1 for lighting up and 0 for lighting off. If there is a doubt of the corresponding relation, you may light up bit by bit in loop.

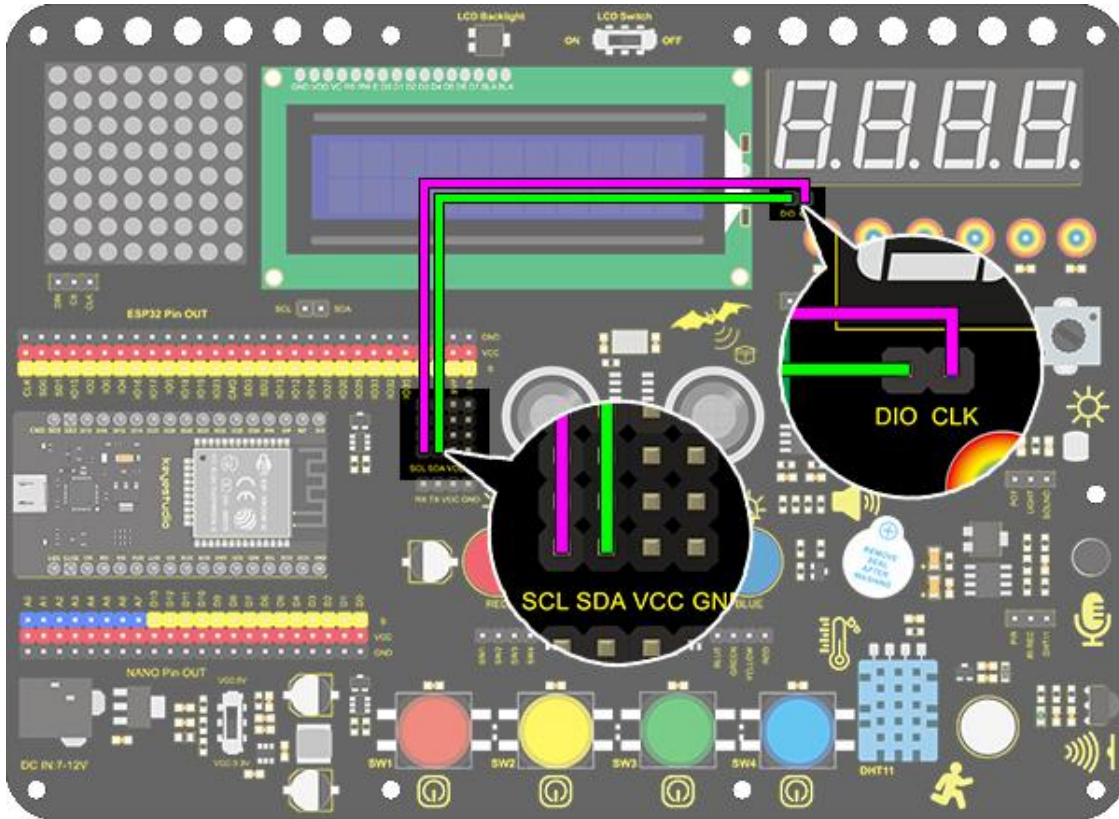
For example, when bit 1 is turned on and displays 8, the data is 0x68. If there is a dot, 8 will also be displayed when sending 0x7f.

Steps: Starting signal — Send 0x68 — Slave-device receives — Send 0x7f — Slave-device receives — Ending signal

Result: 8 is displayed on Bit 1.

For convenience, an array of corresponding value to 0~9 can be made. After further improvement, it is able to display numbers, adjust brightness, shift the decimal dot and tubes.

### 3. Wiring Diagram



### 4. Upload Code

Use the Arduino IDE to open the .ino format code [project\\_9.1](#) directly from the tutorial package.

Connect the ESP32 board to the computer with the USB cable.

Select board type "**ESP32 Dev Module**" and select port **COM-XX** (This depends on the number your computer assigns to the ESP32 board, which you can check it in the device manager).

```

/*
  keyestudio ESP32 Inventor Learning Kit
  Project 9.1 Digital Tube Display
  http://www.keyestudio.com
*/
#include "TM1650.h"
#define CLK 22    //pins definitions for TM1650 and can be changed to other ports
#define DIO 21
TM1650 DigitalTube(CLK,DIO);

void setup(){
  for(char b=0;b<4;b++){
    DigitalTube.clearBit(b);    //DigitalTube.clearBit(0 to 3); Clear bit display.
  }
}

void loop(){
  DigitalTube.displayFloatNum(9999);    //Values or variables added to the parentheses can be displayed
}

```

## 5. Test Result

After connecting the wiring and uploading code, the digital tube display shows "9999", as shown below.



## 6. Extended Code

Let's have some difficult operations. Rather than static numbers, we handle it to show some dynamic ones.

The following code manipulates the tubes to display 1~9999 by "for" loop.

**The wiring remains unchanged.**

Use the Arduino IDE to open the .ino format code [project\\_9.2](#) directly from the tutorial package.

Connect the ESP32 board to the computer with the USB cable.

Select board type "**ESP32 Dev Module**" and select port **COM-XX** (This depends on the number your computer assigns to the ESP32 board, which you can check it in the device manager).

```
/*
  keyestudio ESP32 Inventor Learning Kit
  Project 9.2 Digital Tube Display
  http://www.keyestudio.com
*/
#include "TM1650.h"
#define CLK 22    //pins definitions for TM1650 and can be changed to other ports
#define DIO 21
TM1650 DigitalTube(CLK,DIO);

void setup(){
  for(char b=0;b<4;b++){
    DigitalTube.clearBit(b);    //DigitalTube.clearBit(0 to 3); Clear bit display.
  }
}

void loop(){
  for(int num=0; num<10000; num++){ //If num is less than 10000, num will increase by 1 for each cycle
    DigitalTube.displayFloatNum(num); //Values or variables in the parentheses can be displayed through
    delay(100);
  }
}
```

## Test Result

After uploading code, the digital tube displays 1~9999 by "for" loop

## 7. Code Explanation

**TM1650 DigitalTube(CLK,DIO);** Create an example for DigitalTube, and import the pin number connecting CLK to DIO into the code.

**DigitalTube.clear();** Clear the display

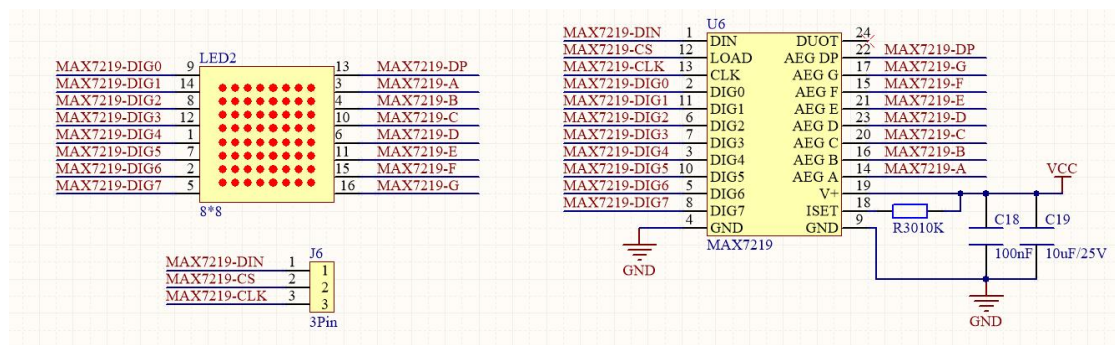
**DigitalTube.displayFloatNum(num);** This is the digital tube display function, the maximum number is 9999.

# Project 10: Dot Matrix Display

## 1. Description

This module consists of a 8x8 LED dot matrix with one control pin for each row as well as each column to adjust the brightness of LED. Connecting with Arduino board, the brightness of LED is controlled to display characters and figures via Arduino programming. In this way, simple characters, numbers and figures are able to be displayed. It also can be applied in game machines or screens.

## 2. Working Principle



MAX7219 is an IC with SPI communication and can be used to control the 8x8 dot matrix. The MAX7219 SPI communication has integrated in our libraries and you can recall directly.

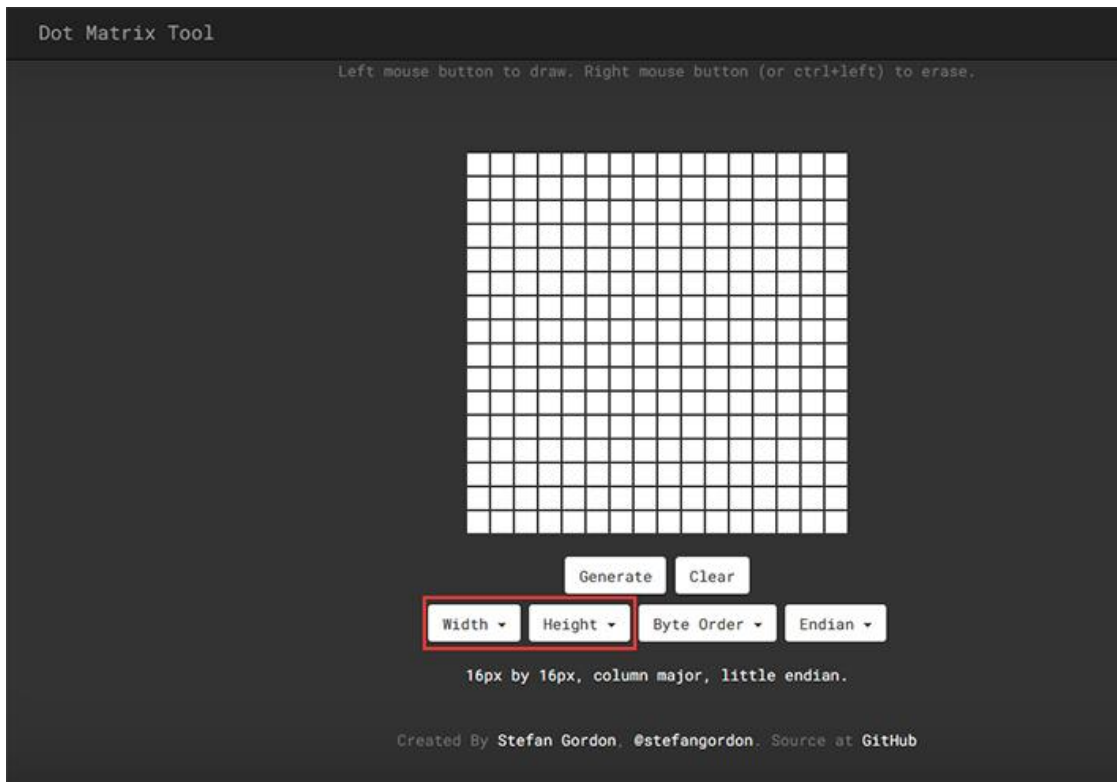
### Dot Matrix Modulo Operation

Click the link for Modulo : <http://dotmatrixtool.com/#>

### Steps:

1. Click the link and set the height and width of the dot matrix. Here we set both to 8.

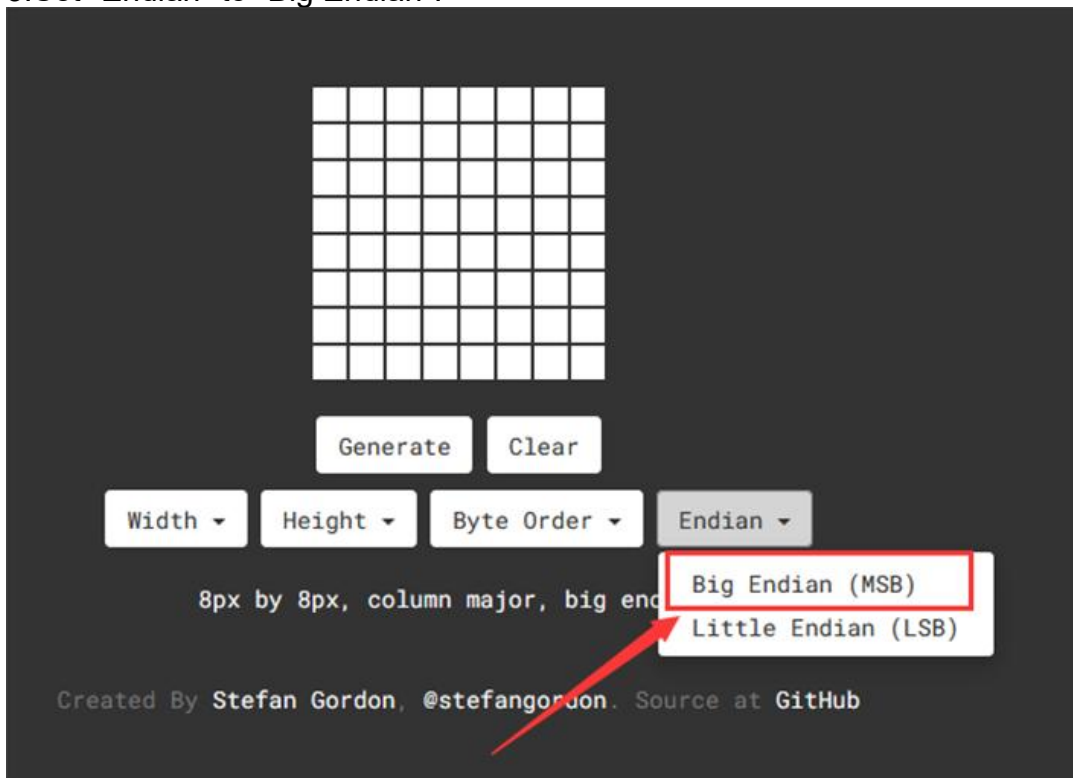




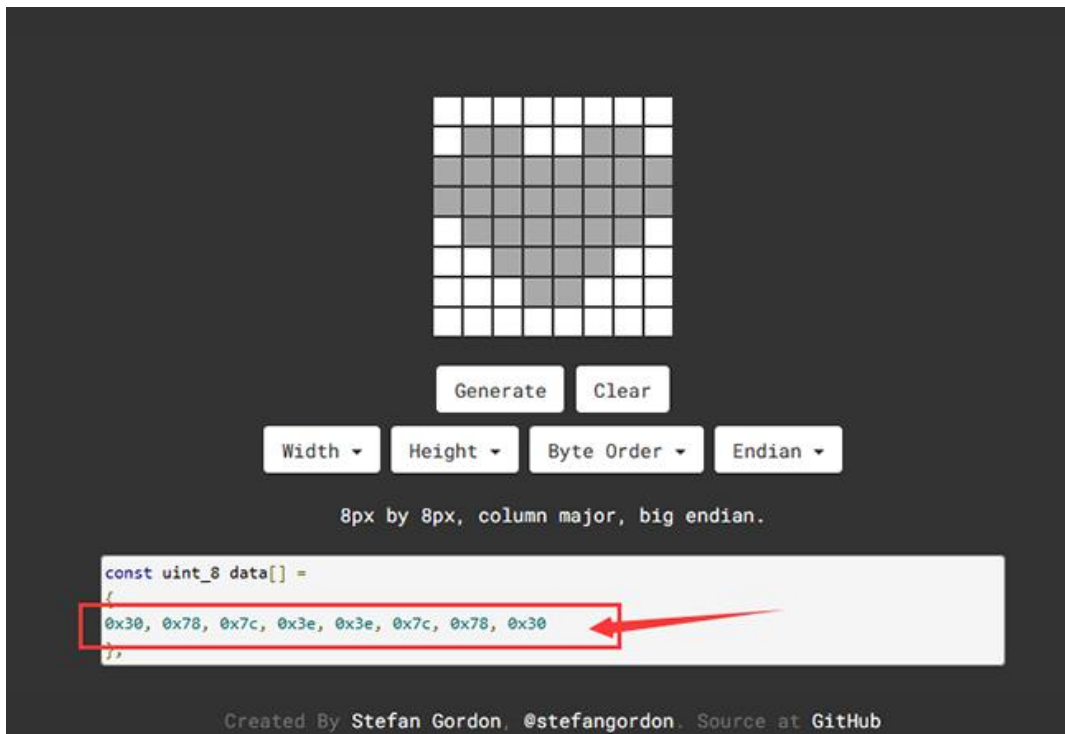
2. Set "Byte Order" to "Column Major".



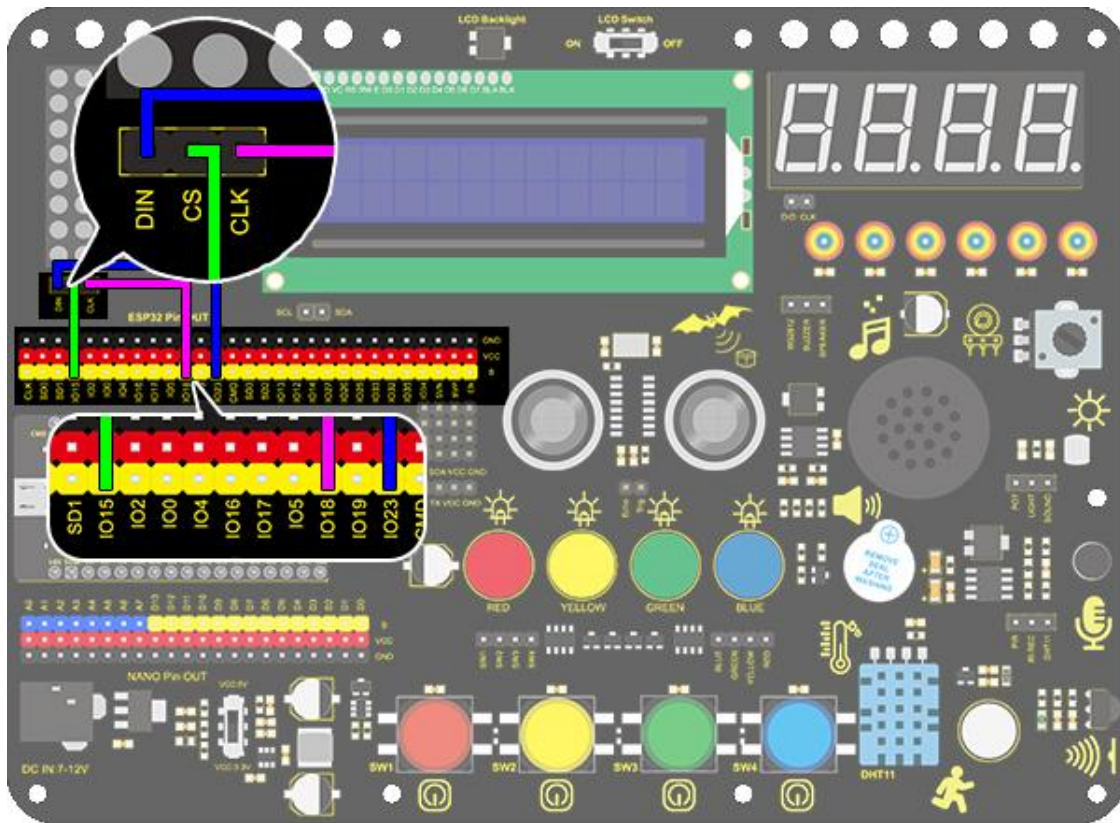
3. Set "Endian" to "Big Endian".



4. Click the white tiles to form a pattern you want (click again for deselecting), and then click "Generate" to generate an array for this icon. Copy this array and paste it in code, and then the pattern will be displayed on the dot matrix.



### 3. Wiring Diagram



### 4. Upload Code

Use the Arduino IDE to open the .ino format code [project\\_10](#) directly from the tutorial package.

Connect the ESP32 board to the computer with the USB cable.

Select board type "**ESP32 Dev Module**" and select port **COM-XX** (This depends on the number your computer assigns to the ESP32 board, which you can check it in the device manager).

```

/*
  keyestudio ESP32 Inventor Learning Kit
  Project 10 Dot Matrix Display
  http://www.keyestudio.com
*/

#include "LedControl.h"
int DIN = 23;
int CLK = 18;
int CS = 15;
LedControl lc=LedControl(DIN,CLK,CS,1);
const byte IMAGES[8] = {0x30, 0x78, 0x7c, 0x3e, 0x3e, 0x7c, 0x78, 0x30};

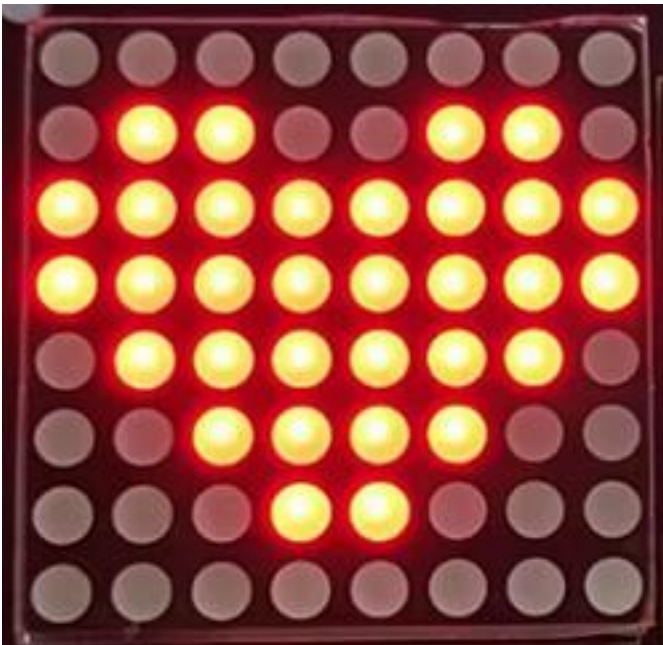
void setup() {
  lc.shutdown(0,false);
  // Set brightness to a medium value
  lc.setIntensity(0,8);
  // Clear the display
  lc.clearDisplay(0);
}

void loop(){
  for(int i=0; i < 8; i++){
    lc.setRow(0,i,IMAGES[i]);
  }
}

```

## 5. Test Result

After connecting the wiring and uploading code, a heart will be displayed on the dot matrix, as shown below.



## 6. Code Explanation

**lc.shutdown(0,false);** Select the state of power saving mode, with false for exiting and true for entering. It will not display anything if entering this mode.

**lc.setIntensity(0,8);** Set the range of brightness intensity to level 0-8, among which 8 is the brightest.

**lc.clearDisplay(0);** Clear the pattern displayed on the dot matrix.

**lc.setRow(0,i,IMAGES[i]);** It is a dot matrix display function, the first parameter is the address of the display we set to 0, the second parameter is the display line, we use for loop variable (0-7), the third parameter is to set the value of the dot call array displayed in the dot matrix row.

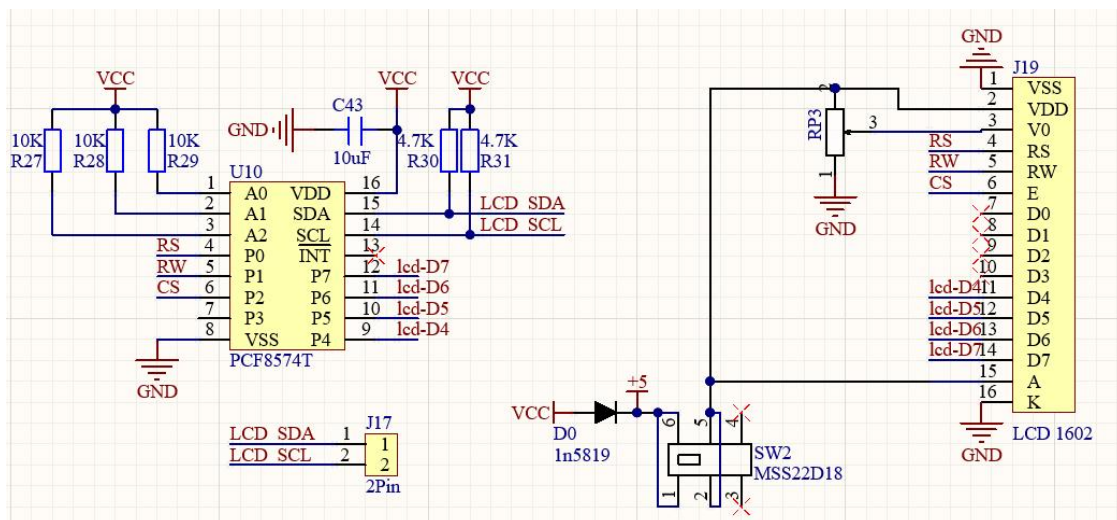
## Project 11: LCD

## 1. Description

Arduino I2C 1602 LCD is a commonly-used auxiliary device for MCU development board to connect with external sensors and modules. It features a 16-bit wide character, 2-line LCD screen and adjustable brightness. This programmable module is convenient for data editing, display and management . Besides, it can display not only characters and figures but sensors value, like temperature, humidity or pressure value.

As a result of its usability, the display is widely applied in many fields, including smart home products, industrial monitoring systems, robot control systems and automotive electronics systems.

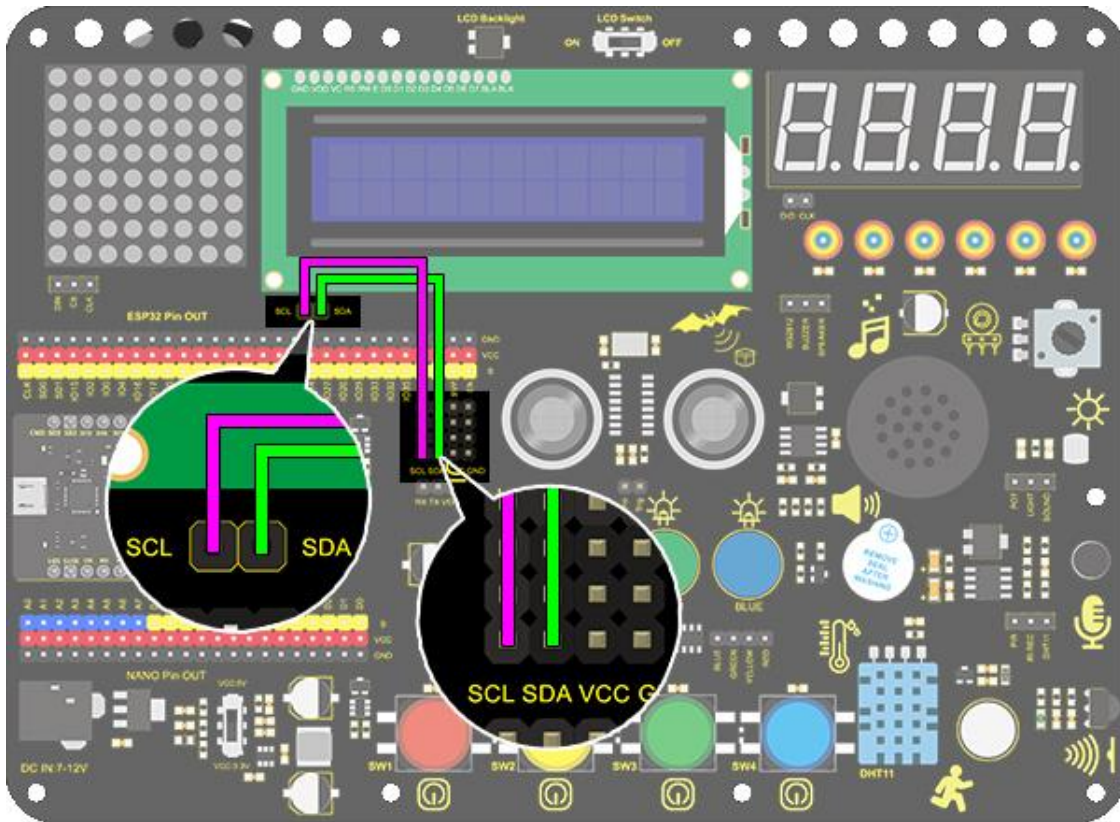
## 2. Working Principle



It is the same as IIC communication principle. Underlying functions have been packaged in libraries so that you can recall them directly. If you are interested in these, you may have a further look of underlying driving principles.

### 3. Wiring Diagram





#### 4. Upload Code

Use the Arduino IDE to open the .ino format code [project\\_11](#) directly from the tutorial package.

Connect the ESP32 board to the computer with the USB cable.

Select board type "**ESP32 Dev Module**" and select port **COM-XX** (This depends on the number your computer assigns to the ESP32 board, which you can check it in the device manager).

```

/*
  keyestudio ESP32 Inventor Learning Kit
  Project 11 LCD
  http://www.keyestudio.com
*/
#include <Wire.h>
#include <LiquidCrystal_I2C.h>
LiquidCrystal_I2C lcd(0x27,16,2); // set the LCD address to 0x27 for a 16 chars and 2 line display
void setup()
{
  lcd.init(); // initialize the lcd
  // Print a message to the LCD.
  lcd.backlight(); //Turn on the LCD backlight
  lcd.setCursor(2,0); //Set the display position
  lcd.print("Hello,world!"); //LCD displays "Hello, world!"
  lcd.setCursor(2,1);
  lcd.print("keyestudio!"); //LCD displays "keyestudio!"
}
void loop()
{
}

```

## 5. Test Result

After connecting the wiring and uploading code, turn on the LCD, "Hello, world!" and "keyestudio!" will be displayed on the LCD.



What if the LCD module does not display anything?

1. Use the flat screwdriver included in the package to rotate the potentiometer next to the LCD module to adjust the backlight of the LCD.
2. Check whether the external battery has sufficient power.



## 6. Code Explanation

**#include <LiquidCrystal\_I2C.h>** #include is a "include" command of libraries, so we can recall functions in file.h.

**LiquidCrystal\_I2C lcd(0x27,16,2);** Define an LCD. 0x27 is its IIC address, and 16 means the number of columns(display 16 characters in total), and 2 is the number of rows.

**lcd.init();** Initialize LCD

**lcd.backlight();** Turn on LCD backlight, which clarifies the displayed characters.

**lcd.setCursor(3,0);** Set the display position. (3,0) indicates the the beginning of column 3, row 0.

**lcd.print("Hello, world!");** Define the displayed characters. Enclose the strings in quotation marks, for instance, lcd.print("Hello, world!"). The marks can be omitted if displaying one value, for example, lcd.print(value).

## Project 12: Servo

### 1. Description

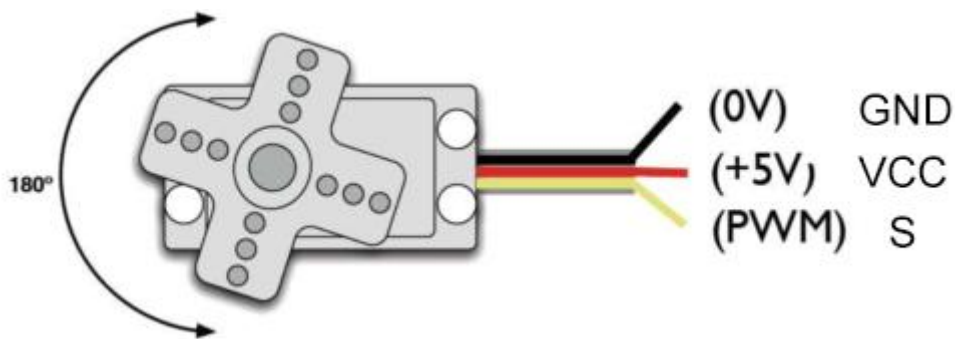
This servo features high performance and high precision with a maximum rotation angle of  $180^\circ$ . Weighing only 9g, it is perfectly suitable for any mini device in multiple occasions. What's more, it enjoys short startup time, low noise and strong stability.

### 2. Working Principle

**Angle range:**  $180^\circ$  ( $360^\circ$ ,  $180^\circ$  and  $90^\circ$ )

**Drive voltage:** 3.3V or 5V

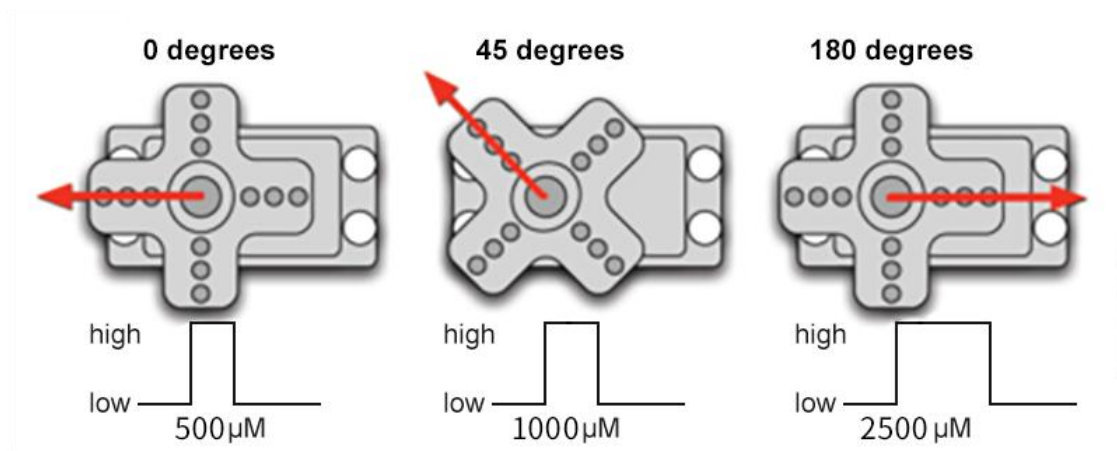
**Pin:** Three wires



**GND:** Grounded(brown)

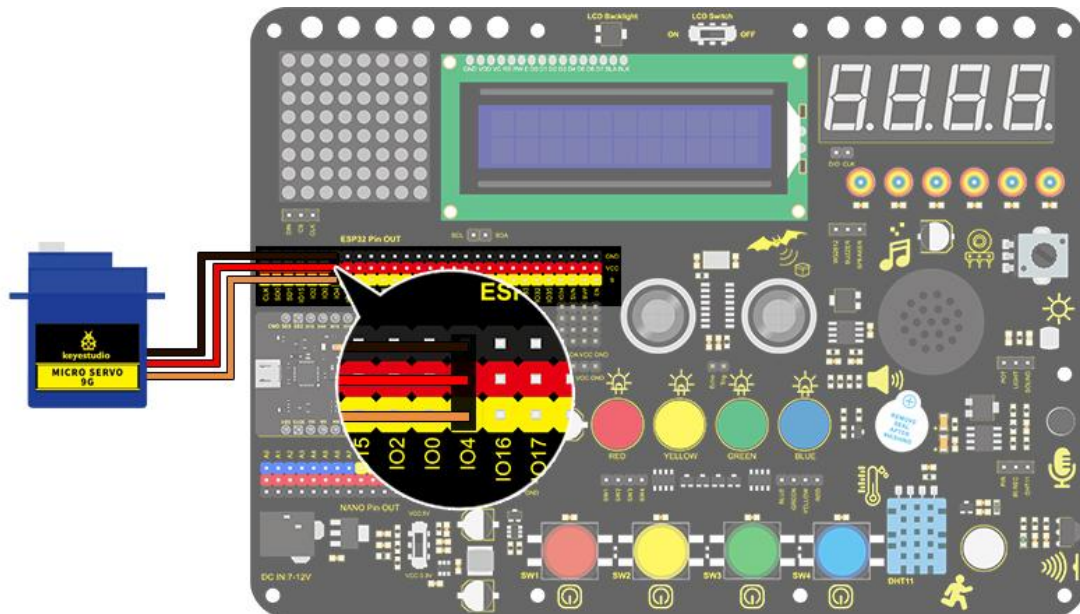
**VCC:** A red pin that connects to a +5v (3.3V) power

**S:** A orange signal pin that controlled via PWM signal



**Control Principle:** The rotation angle is controlled via duty cycle of PWM. Theoretically, standard PWM cycle is 20ms(50Hz), so pulse width should distribute within 1ms~2ms. However, the actual pulse width reaches 0.5ms~2.5ms, which corresponds to 0°~180°. Pay attention that, for the same signal, the rotation angle may vary from servo brands.

### 3. Wiring Diagram



Add an external power source instead of just using USB for power.





## 4. Upload Code

Use the Arduino IDE to open the .ino format code [project\\_12](#) directly from the tutorial package.

Connect the ESP32 board to the computer with the USB cable.

Select board type "**ESP32 Dev Module**" and select port **COM-XX** (This depends on the number your computer assigns to the ESP32 board, which you can check it in the device manager).

```
/*
  keyestudio ESP32 Inventor Learning Kit
  Project 12 Servo
  http://www.keyestudio.com
*/
int servoPin = 4; //servo PIN

void setup() {
  pinMode(servoPin, OUTPUT); //servo pin is set to output
}

void loop() {
  for(int i = 0 ; i <= 180 ; i++) {
    servopulse(servoPin, i); //Set the servo to rotate from 0° to 180°
    delay(10); //delay 10ms
  }
  for(int i = 180 ; i >= 0 ; i--) {
    servopulse(servoPin, i); //Set the servo to rotate from 180° to 0°
    delay(10); //delay 10ms
  }
}

void servopulse(int pin, int myangle) { //Impulse function
  int pulsewidth = map(myangle, 0, 180, 500, 2500); //Map Angle to pulse width
  for (int i = 0; i < 10; i++) { //Output a few more pulses
    digitalWrite(pin, HIGH); //Set the servo interface level to high
    delayMicroseconds(pulsewidth); //The number of microseconds of delayed pulse width value
    digitalWrite(pin, LOW); //Lower the level of servo interface
  }
}
```

## 5. Test Result

After connecting the wiring and uploading code, the servo starts to rotate from 0° to 180° and then reverse.

## 6. Code Explanation



**void servopulse(int pin, int myangle)** : To integrate the code together for easy use and management, the first parameter is the pin number, the second parameter is the Angle of the servo.

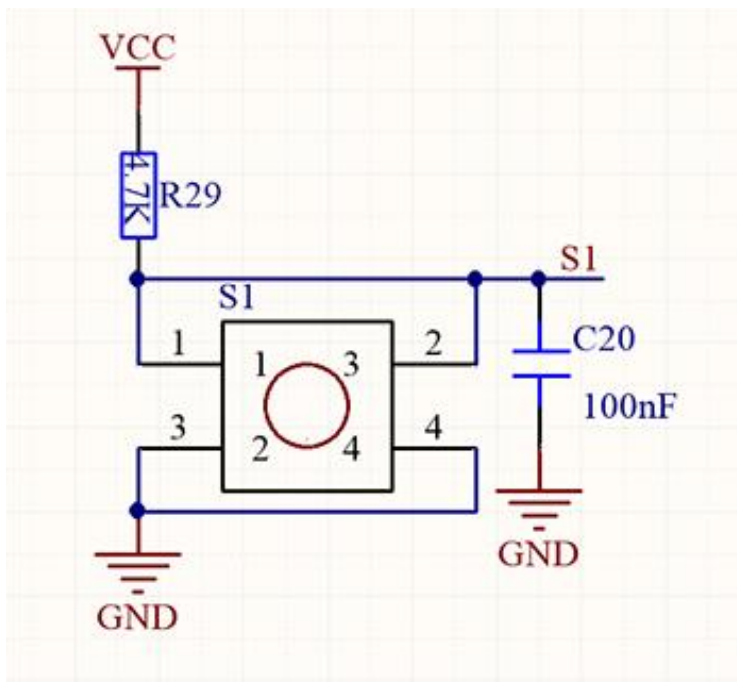
**map(myangle, 0, 180, 500, 2500);** This is a mapping variable range function used to map the range of myangle variable from 0-180 to 500-2500, so that we can get a value of 2500 when the servo is set to 180°, 500-2500 is the time that the servo high level is maintained.

## Project 13: Mini Lamp

### 1. Description

In this project, we are going to control a lamp via Arduino UNO and a button. When we press the button, the state of the lamp will shift(ON or OFF).

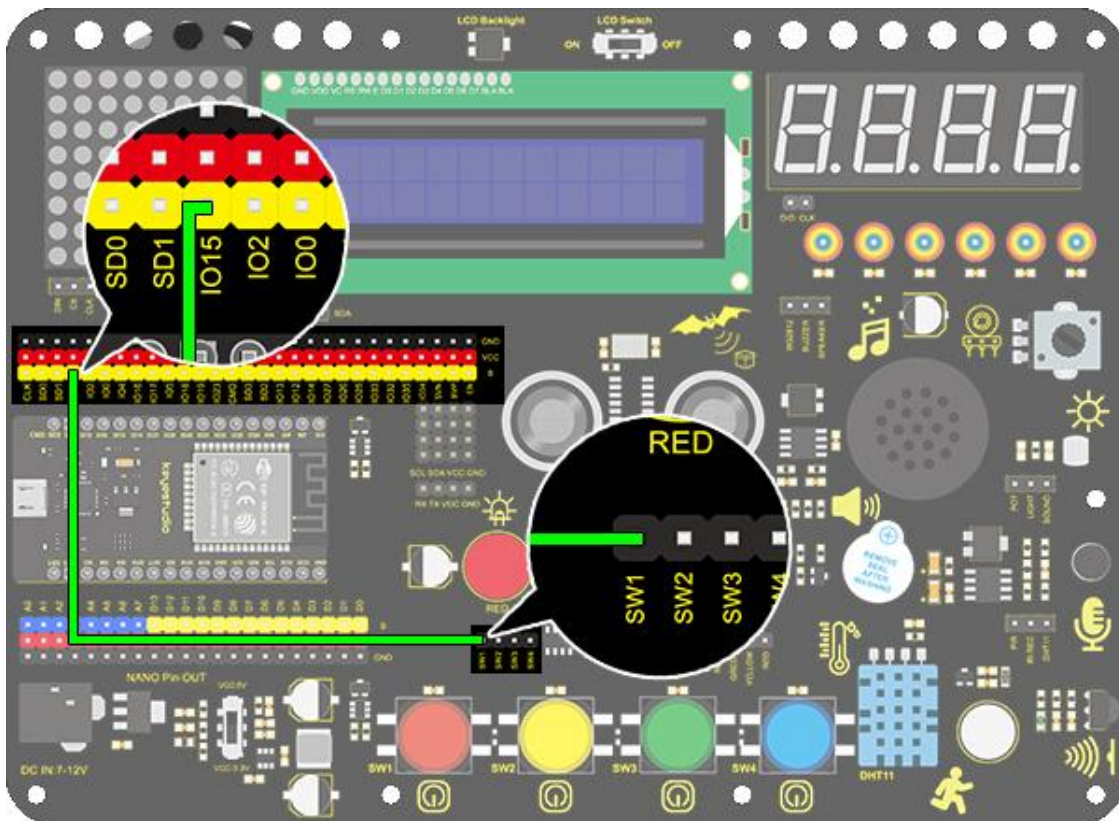
### 2. Working Principle



When the button is released, a voltage VCC passing through R29 provides a high level for S terminal.

When pressed, pin 1 and 3, pin 2 and 4 are connected and voltage on S1 arrives GND as a low level. At this moment, R29 avoids a short circuit between VCC and GND.

### 3. Wiring Diagram



### 4. Upload Code

Use the Arduino IDE to open the .ino format code [project\\_13.1](#) directly from the tutorial package.

Connect the ESP32 board to the computer with the USB cable.

Select board type "**ESP32 Dev Module**" and select port **COM-XX** (This depends on the number your computer assigns to the ESP32 board, which you can check it in the device manager).

```

/*
  keyestudio ESP32 Inventor Learning Kit
  Project 13.1 Mini Lamp
  http://www.keyestudio.com
*/
int button = 15;
int value = 0;
void setup() {
  Serial.begin(9600); //Set the serial baud rate to 9600
  pinMode(button, INPUT); //Connect the button pin to digital port 8 and set it to input mode.
}

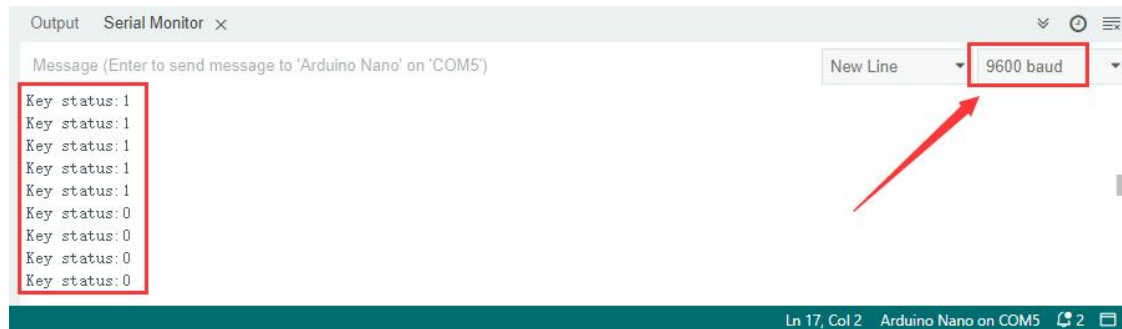
void loop() {
  value = digitalRead(button); //Read the button value
  Serial.print("Key status:"); //Print "Key status:" on serial port
  Serial.println(value); //Print the button variable on the serial port and wrapping lines
}

```

## 5. Test Result

After connecting the wiring and uploading code, open the serial monitor and set the baud rate to 9600.

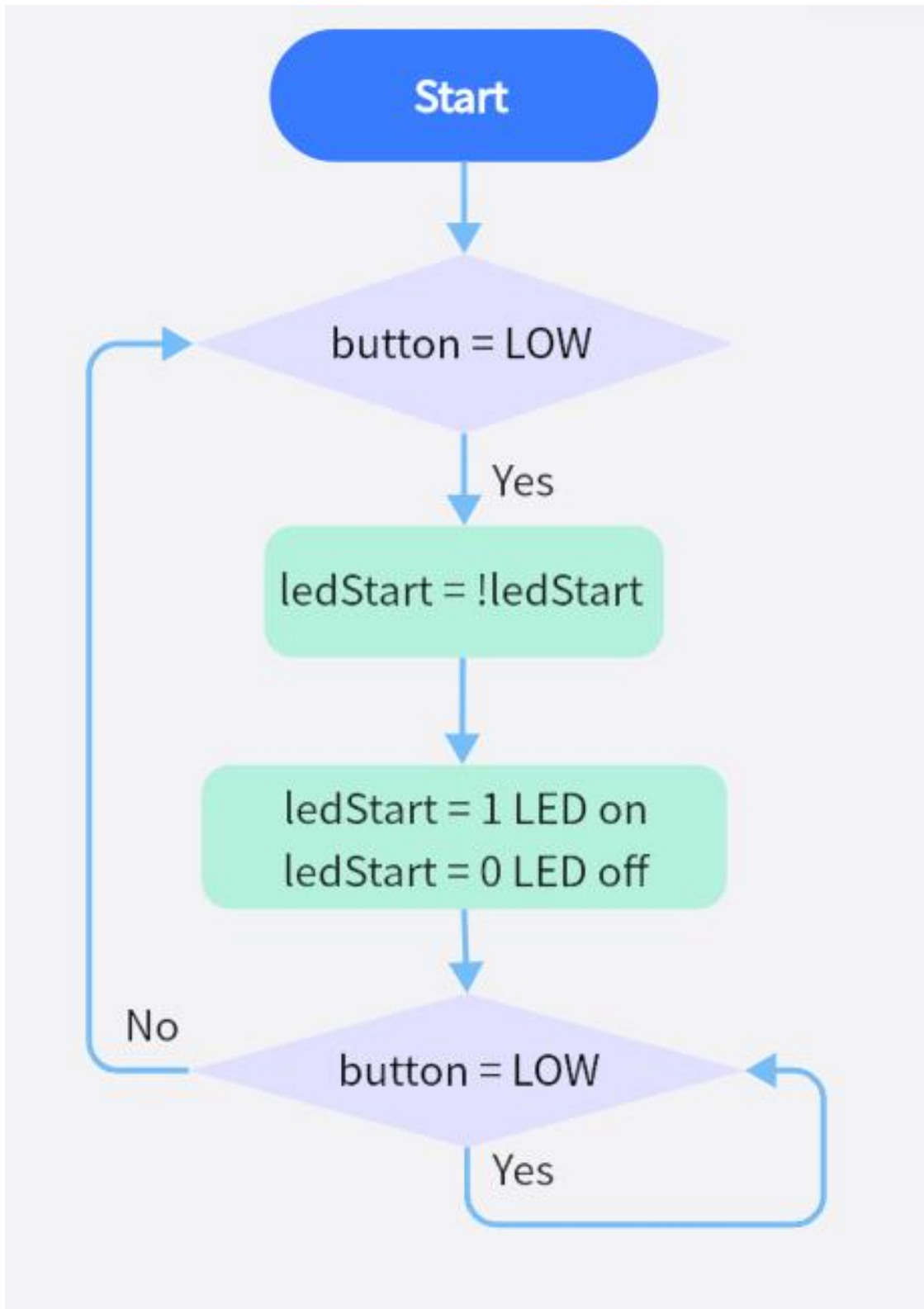
When we press the button, serial port prints "Key status: 0"; When we release it, serial port prints "Key status: 1".



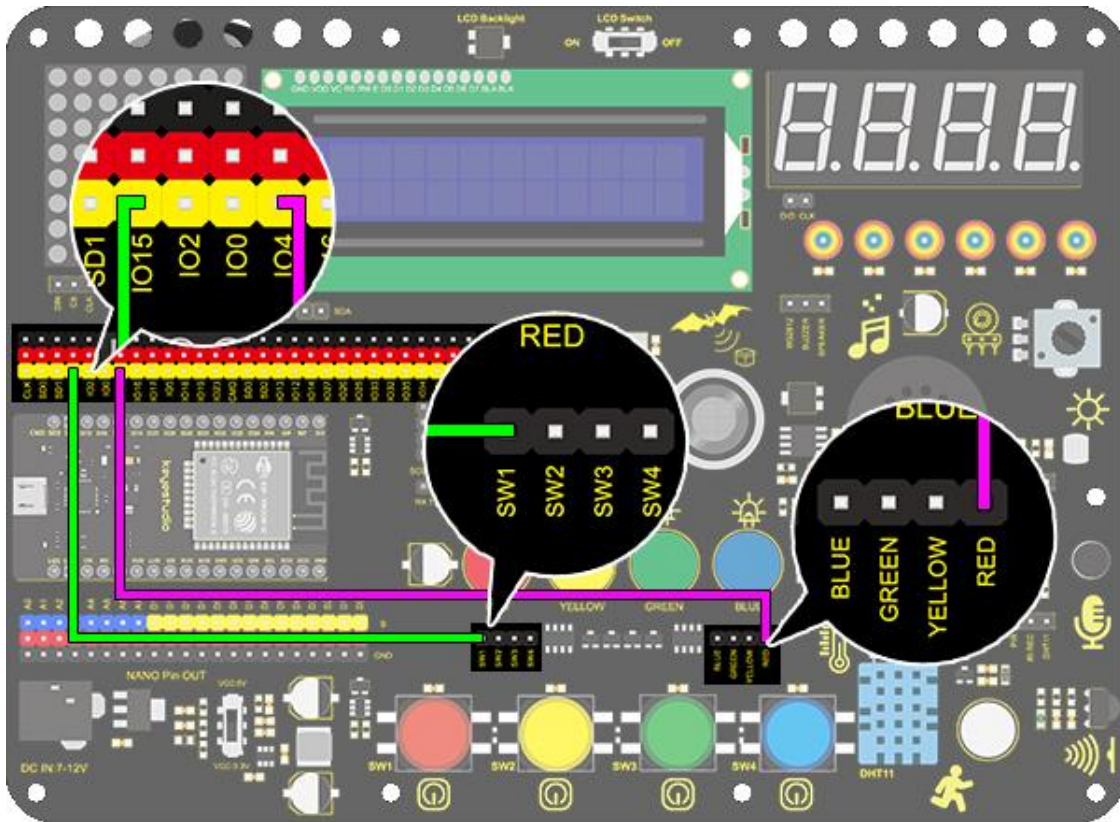
## 6. Knowledge Expansion

Next, we will control the LED through the state of buttons.

**Flow Chart:**



**Wiring Diagram:**



### Code:

Make a mini lamp with a button and a LED.

Use the Arduino IDE to open the .ino format code [project\\_13.2](#) directly from the tutorial package.

Connect the ESP32 board to the computer with the USB cable.

Select board type "**ESP32 Dev Module**" and select port **COM-XX** (This depends on the number your computer assigns to the ESP32 board, which you can check it in the device manager).

```

/*
  keyestudio ESP32 Inventor Learning Kit
  Project 13.2 Mini Lamp
  http://www.keyestudio.com
*/
#define led      4
#define button 15
bool ledState = false;

void setup() {
  // initialize digital pin PIN_LED as an output.
  pinMode(led, OUTPUT);
  pinMode(button, INPUT);
}

// the loop function runs over and over again forever
void loop() {
  if (digitalRead(button) == LOW) {    //When the button value is 0 for the first time, button jitter is
    delay(20);                          //Delay 20ms
    if (digitalRead(button) == LOW) {    //judge whether the button value is 0

```

## Test Result:

You can control the red LED light on and off by the red button.

## 7. Code Explanation

**pinMode(button, INPUT);** Set pin IO15 on the development board to input, so that the state of button can be identified.

When we press the button, IO15 is at a low level(0). If we release it, IO15 will be at high(1).

**value = digitalRead(button);** digitalRead(button) reads the high and low level (1 or 0) of the digital IO15 pin and assigns the button value to value.

**Serial.begin(9600);** Set the serial baud rate. It is necessary to print value on serial port.

**Serial.print("Key status:");** Serial port prints value. Contents in print() will be printed. If it is character string, quotation marks are needed, for instance, "Key status:".

**Serial.println(button);** Serial port prints contents in println() in a new line. Here we print the button value.

**if (digitalRead(button) == LOW) { ... }:** Determine if the button pin is equal to the low level. If so, execute the if statement {... }, otherwise not executed.

**!** takes the inverse operator, so if this value is 1, it's 0, and if it's 0, it's 1, so we can turn a light on and off by pressing a button.



**==** is used to determine whether the value of the variable on the left is equal to the value on the right. Please refer to the official website for details:  
<https://www.arduino.cc/reference/en/>.

The screenshot shows the Arduino Reference website with a teal header. The main content area is divided into several columns. On the left, there's a sidebar with links like VARIABLES, STRUCTURE, LIBRARIES, IOT CLOUD API, and GLOSSARY. The main content area has a red box highlighting the 'Operators' section, which includes:

- Arithmetic Operators**
  - % (remainder)
  - \* (multiplication)
  - + (addition)
  - (subtraction)
  - / (division)
  - = (assignment operator)
- Comparison Operators**
  - != (not equal to)
  - < (less than)
  - <= (less than or equal to)
  - == (equal to)
  - > (greater than)
  - >= (greater than or equal to)
- Boolean Operators**
  - ! (logical not)
  - && (logical and)
  - || (logical or)
- Pointer Access Operators**
  - & (reference operator)
  - \* (dereference operator)
- Bitwise Operators**
  - & (bitwise and)
  - << (bitshift left)
  - >> (bitshift right)
  - ^ (bitwise xor)
  - | (bitwise or)
  - ~ (bitwise not)
- Compound Operators**
  - %= (compound remainder)
  - &= (compound bitwise and)
  - \*= (compound multiplication)
  - ++ (increment)
  - += (compound addition)
  - (decrement)
  - = (compound subtraction)
  - /= (compound division)
  - ^= (compound bitwise xor)
  - |= (compound bitwise or)

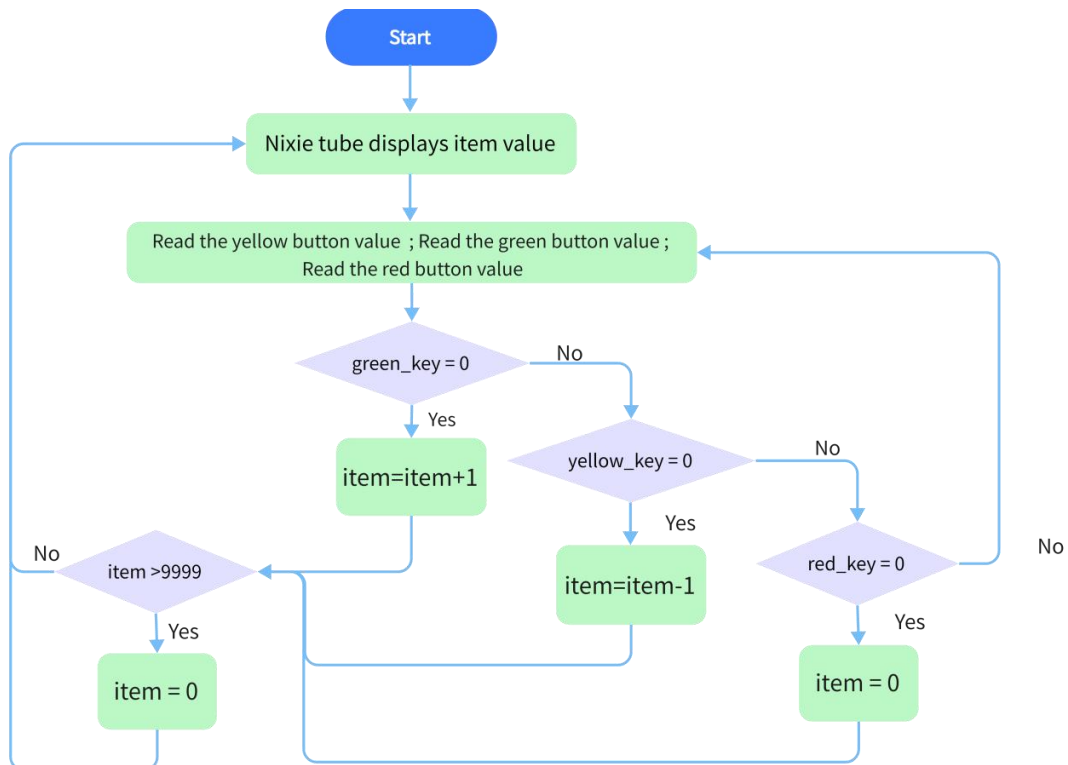
Other sections visible include 'Sketch' (loop(), setup()), 'Control Structure' (break, continue, do...while, else, for, goto, if, return, switch...case, while), and 'Further Syntax' (#define, #include, /\* \*/, //, ;, {}). A 'Help' button is visible in the bottom right corner.

## Project 14: Counter

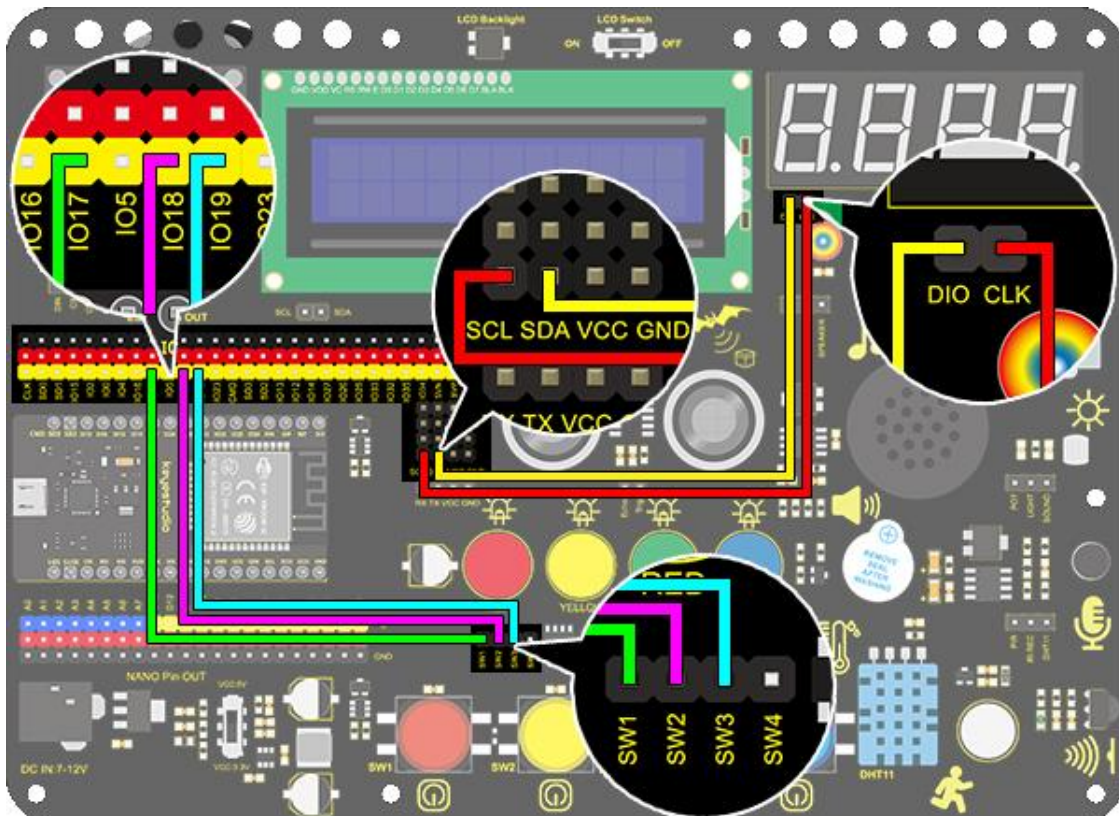
### 1. Description

Arduino 4-bit digital tube counter can record numbers within 0~9999. It features display speed, count mode adjustment as well as reset function. This module is widely applied in real-time counter (such as button-press and DC motor rotation count), gaming and experiment equipment.

## 2. Flow Chart



## 3. Wiring Diagram



### 3. Upload Code

A counter includes three buttons: plus, minus, and reset(return to zero). We program "if" to determine the state of button, "pressed" for execution. For better results, we need to add a 200ms delay.

Use the Arduino IDE to open the .ino format code [project\\_14](#) directly from the tutorial package.

Connect the ESP32 board to the computer with the USB cable.

Select board type "**ESP32 Dev Module**" and select port **COM-XX** (This depends on the number your computer assigns to the ESP32 board, which you can check it in the device manager).

```
/*
  keyestudio ESP32 Inventor Learning Kit
  Project 14 Counter
  http://www.keyestudio.com
*/
#include "TM1650.h" //Upload TM1650 library file
int item = 0; //Displayed value
#define CLK 22 //pins definitions for TM1650 and can be changed to other ports
#define DIO 21
TM1650 DigitalTube(CLK,DIO);

int res = 17; //Reset button
int subtract = 18; //minus button
int add = 19; //plus button

void setup(){
  //set the pin connecting with button to input
  pinMode(res,INPUT);
  pinMode(add,INPUT);
  pinMode(subtract,INPUT);
  for(char b=0;b<4;b++){
    DigitalTube.clearBit(b); //DigitalTube.clearBit(0 to 3); Clear bit display.
  }
}
```

### 4. Test Result

After connecting the wiring and uploading code, press green button to add 1, yellow to minus 1, and red to reset. Press the button and hold it, and the displayed value will keep adding or reducing.

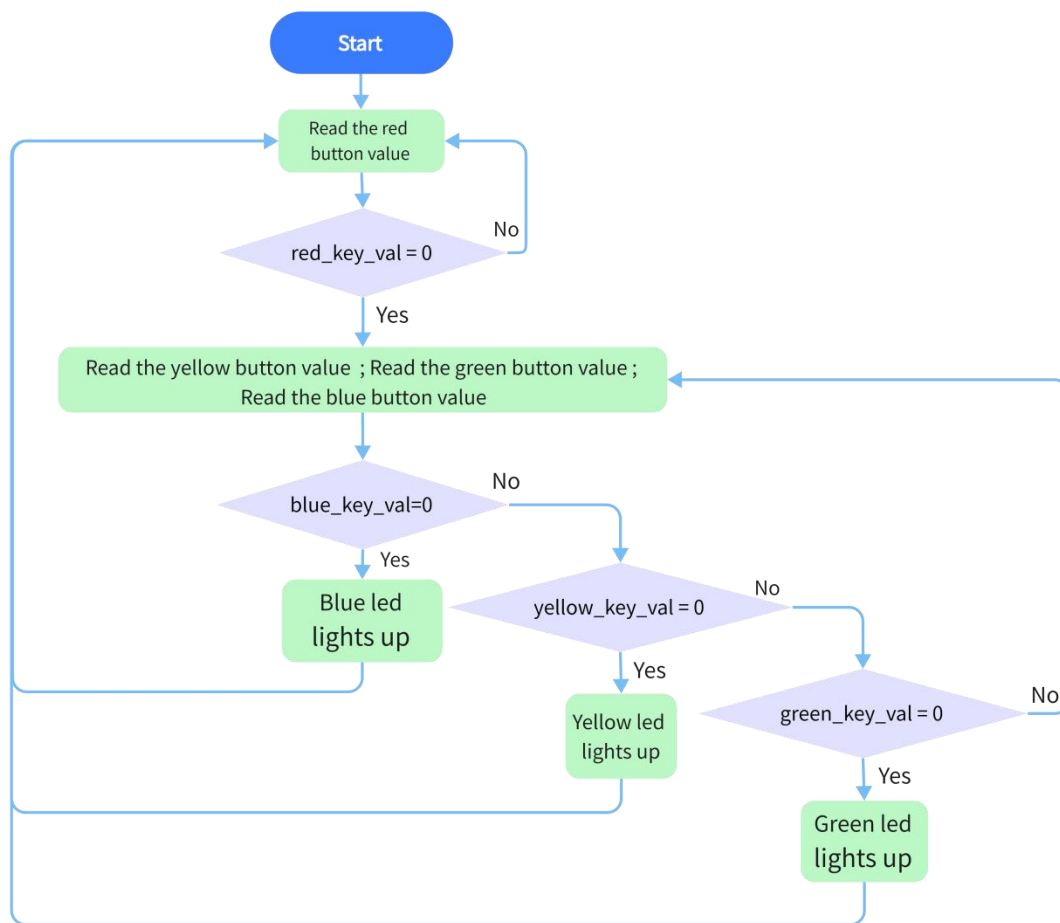
# Project 15: Responder

## 1. Description

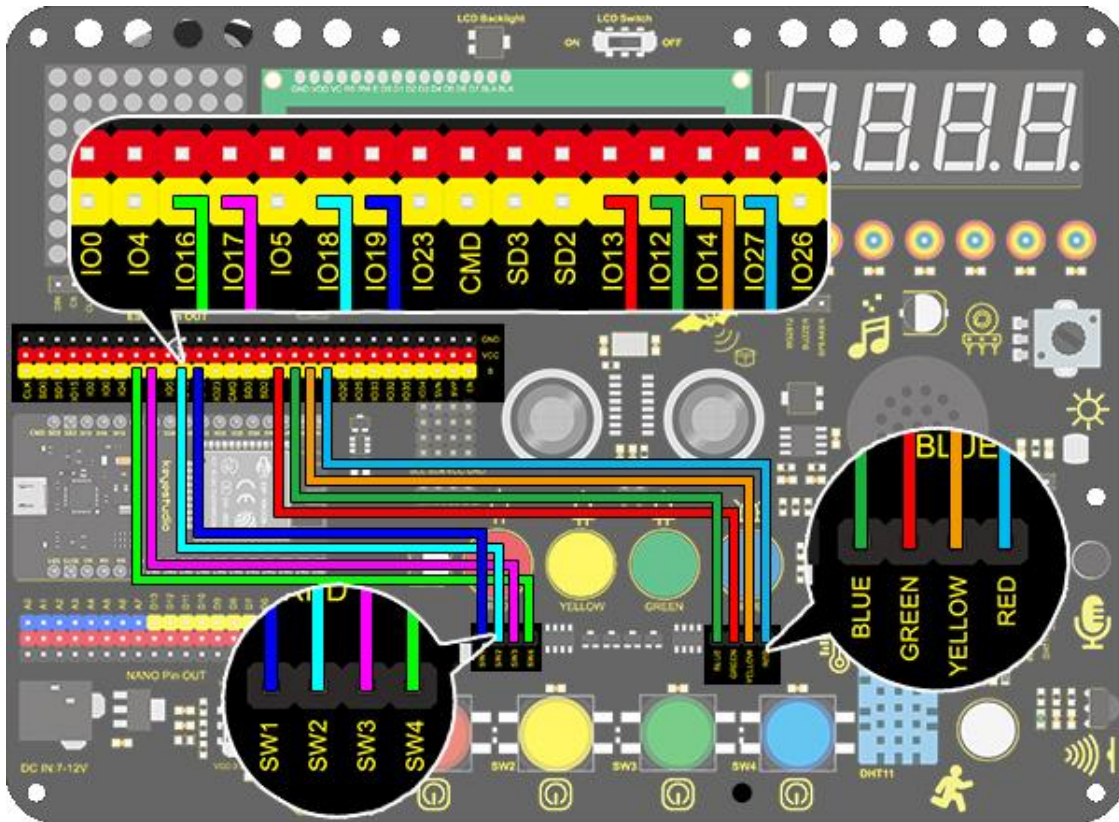
This programmable responder inputs and receives signals through Arduino development board and a group of buttons, and it judges the correctness of answers via a LED. It is a good object to exercise students' reaction ability and draw their attention to questions. If the answer is correct, the respondent obtains a lot scores.

Moreover, it simplifies teachers' manipulation of question-grabbers and cuts answer clutters. It may even stimulate students' interests in learning.

## 2. Flow Chart



## 3. Wiring Diagram



#### 4. Upload Code

Imagine a question-master and three respondents.

Respondents are allowed to grab questions only when the master presses the red button. Otherwise, their replies are invalid and lights are all off. Plus, if one of the three presses his/her button, the remaining two buttons are also invalid.

Use the Arduino IDE to open the .ino format code [project\\_15](#) directly from the tutorial package.

Connect the ESP32 board to the computer with the USB cable.

Select board type "**ESP32 Dev Module**" and select port **COM-XX** (This depends on the number your computer assigns to the ESP32 board, which you can check it in the device manager).



```

/*
  keyestudio ESP32 Inventor Learning Kit
  Project 15 Responder
  http://www.keyestudio.com
*/
int blue_key = 16;      //Set blue button to connect pin D3
int green_key= 17;      //Set green button to connect pin D4
int yellow_key = 18;    //Set yellow button to connect pin D5
int red_key = 19;       //Set red button to connect pin D6

int blue_led = 12;      //Set blue LED to connect pin D7
int green_led = 13;     //Set green LED to connect pin D8
int yellow_led = 14;    //Set yellow LED to connect pin D9
int red_led = 27;       //Set red LED to connect pin D10

void setup(){
  //Set the pin connecting with button to input
  pinMode(blue_key,INPUT);
  pinMode(green_key,INPUT);
  pinMode(yellow_key,INPUT);
  pinMode(red_key,INPUT);
  //Set the pin connecting with LED to output

```

## 5. Test Result

Let's simulate a quick-answer game.

Press the red button to turn off all LED lights. Then we can select the yellow, green and blue buttons to turn on the corresponding LED lights. The person whose LED light turns on first can answer first.

## 6. Code Explanation

**while(1) { ... }** Unlimited loop function.

When the expression or value in while() is True, the execution circulates in while{}. On the contrary, the loop quits when it is False.

In this example, "1" in while(1) represents True, so code is on a loop when entering "while", which is endless.

For how to exit, we need a "break" statement.

**break;** It is used to exit a loop.



# Project 16: Timebomb

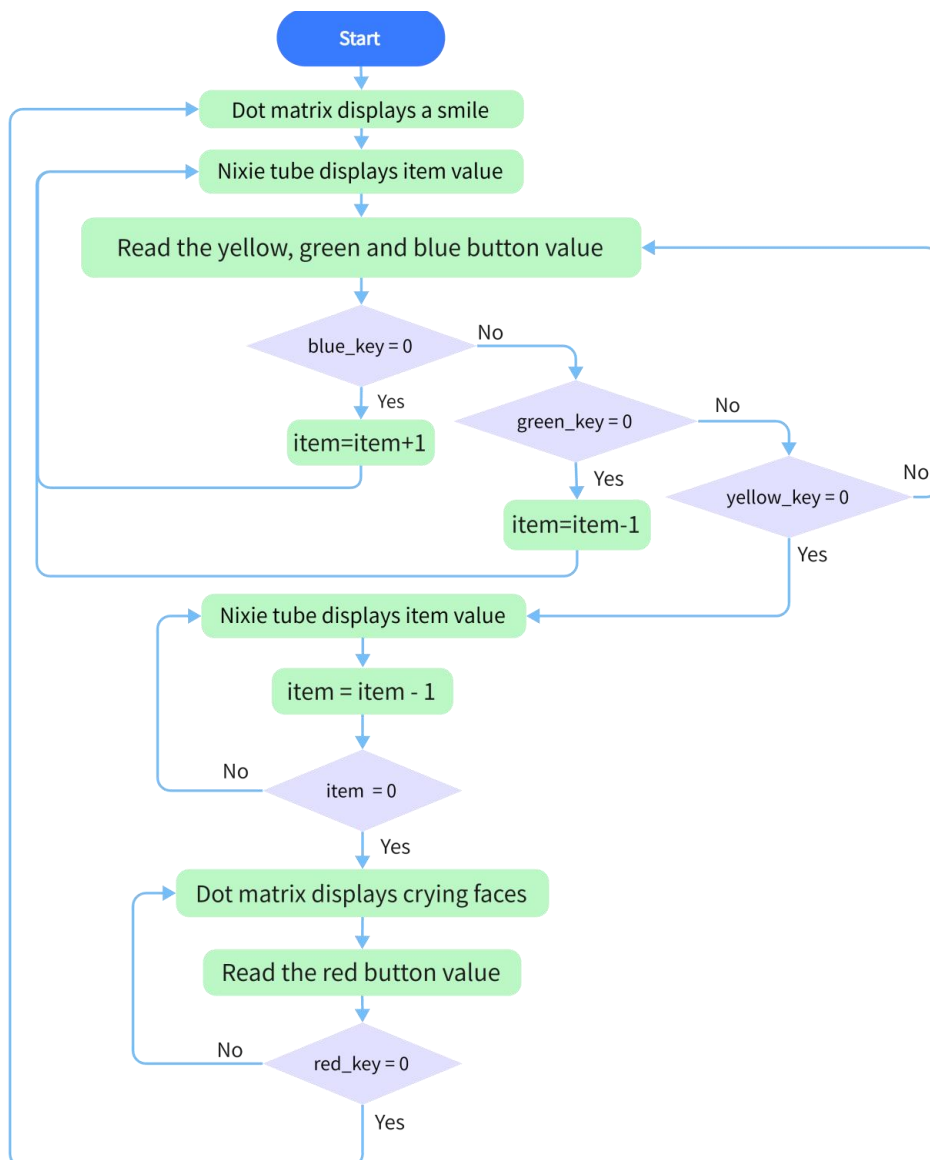
## 1. Description

This project will give you an opportunity experience an interesting timebomb game.

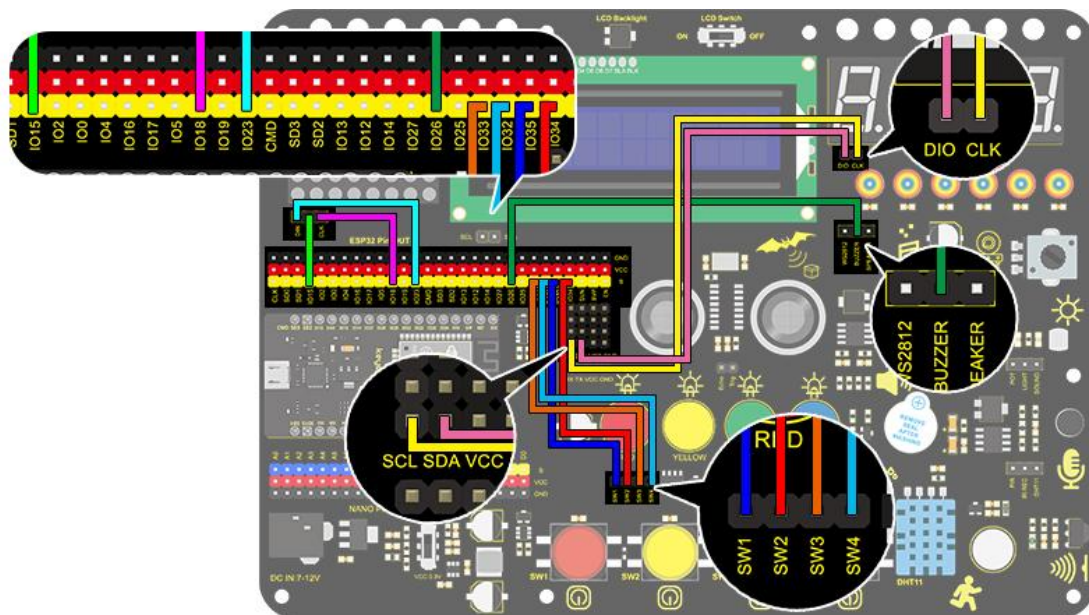
In this project, the dot matrix represents your timebomb, while the digital tube displays remaining time. Buttons can not only control the bomb but also set its time. You may set a countdown to control this bomb, and it explodes when the countdown is over. Beyond that, a buzzer is adopted to alarm.

Anyhow, by programming on multiple sensors, your comprehensive capability of logic thinking can be enhanced.

## 2. Flow Chart



### 3. Wiring Diagram



## 4. Upload Code

When mentioning a timebomb, we think of a timer and an activate button. In this project, however, it is an analog bomb, so we also need a reset button. We set blue for plus, green for minus, yellow for counting down and red for resetting. The time (unit: s) is displayed on digital tube and the 8x8 dot matrix shows the state of bomb(smile for safe and cry for explosion).

Use the Arduino IDE to open the .ino format code **project\_16** directly from the tutorial package.

Connect the ESP32 board to the computer with the USB cable.

Select board type "**ESP32 Dev Module**" and select port **COM-XX** (This depends on the number your computer assigns to the ESP32 board, which you can check it in the device manager).

```

/*
  keyestudio ESP32 Inventor Learning Kit
  Project 16 Timebomb
  http://www.keyestudio.com
*/
#include "TM1650.h" //Upload TM1650 Libraries
#include "LedControl.h"
//Dot matrix
int DIN = 23;
int CLK = 18;
int CS = 15;
LedControl lc=LedControl(DIN,CLK,CS,1);

byte smile[8]= {0x20,0x44,0x22,0x02,0x02,0x22,0x44,0x20}; //Smile face
byte weep[8]= {0x20,0x42,0x24,0x04,0x04,0x24,0x42,0x20}; //Crying face

// Button, buzzer and digital tube
int item = 0; //displayed value
TM1650 DigitalTube(22,21); //Set the SCL pin of the digital tube to 22 and the DIO pin to 21

int addition = 32; //Set the plus button to I032
int subtraction = 33; //Set the minus button to I033
int start = 34; //Set the start button to I034
int reset = 35; //Set the reset button to I035
int buzz = 26; //Set the buzzer to I026

int buzz_val = 1; //The variable of buzzer

```

## 5. Test Result

After connecting the wires and uploading the code, the blue button (increase) and green button (decrease) can be used to set the time, and the yellow button can be used to count down. When the countdown ends, the smiley face on the Dot Matrix Display turns into a crying face, simulating the explosion of the bomb. Pressing the red button can reset the program.

# Project 17: Invasion Alarm

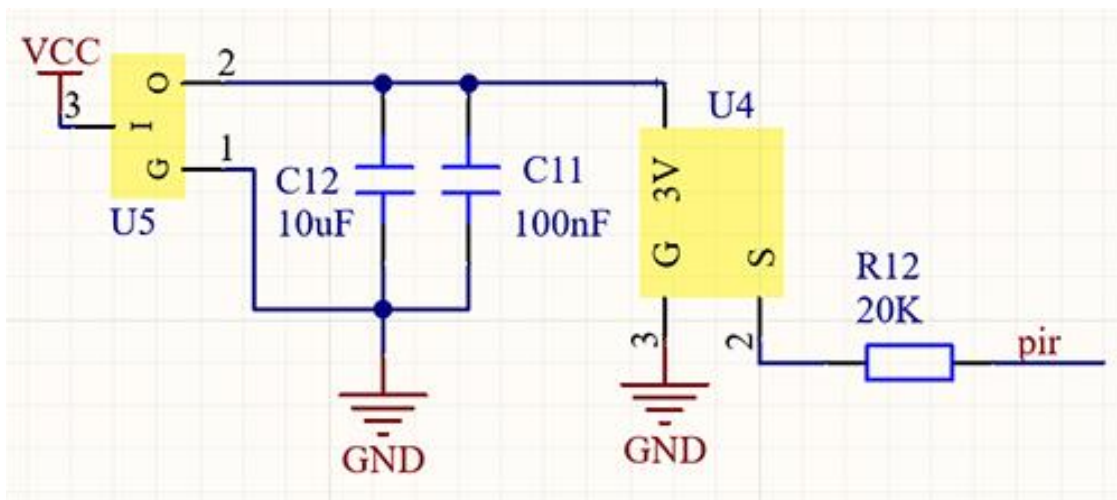
## 1. Description

This invasion alarm system is able to detect invaders in houses or small offices and warn the host to take measures in time.

In this project, the sensor monitors a certain area. Some device on Arduino board will trigger LED to light up and buzzer to beep for caution if a movement is detected in that zone.

Virtually, this module features practicability, easy installation and low costs. With the exception of home and office, it also applies to factories, warehouses and markets, which, to a large extent, protects property security.

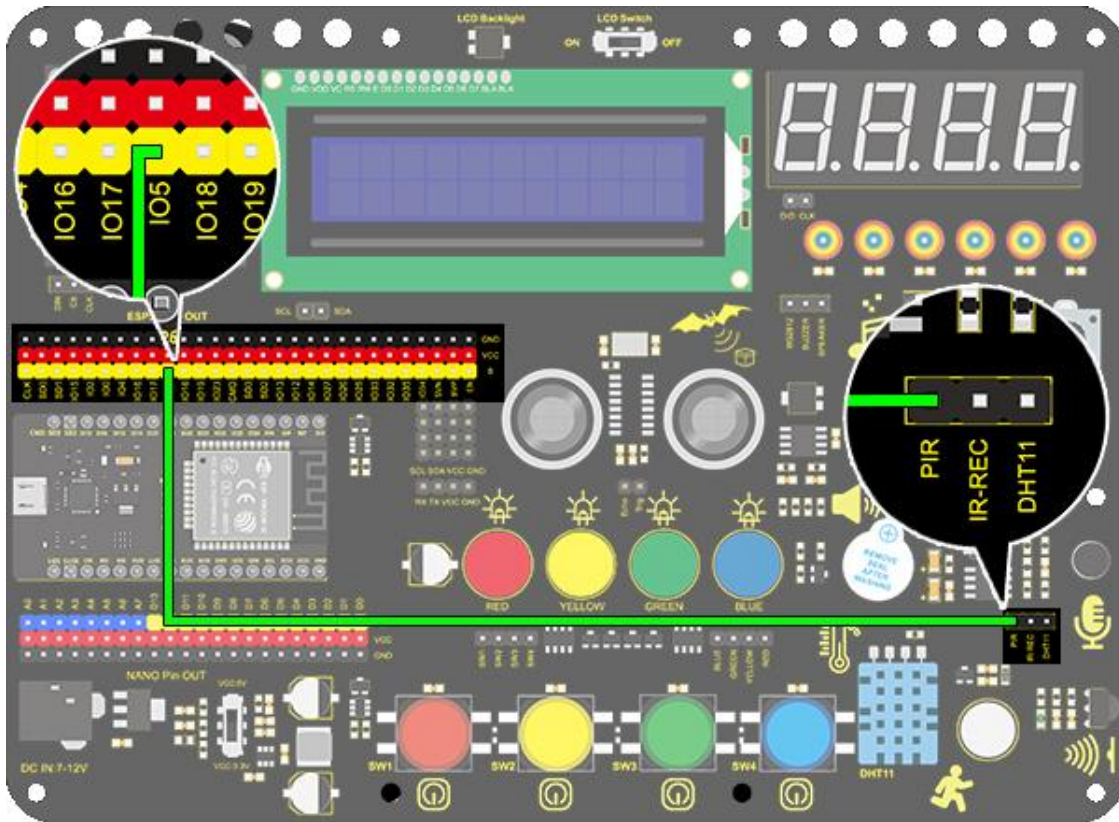
## 2. Working Principle



Human body(37°C) always emits infrared ray with a wavelength of 10μm, which approximates to that of the sensor detected.

On this account, this module is able to detect human beings movement. If there is, PIR sensor outputs a high level about 3s. If not, it outputs a low level .

## 3. Wiring Diagram



#### 4. Upload Code

From the working principle, we can read the level of the sensor pin to judge whether there are people nearby.

Use the Arduino IDE to open the .ino format code [project\\_17.1](#) directly from the tutorial package.

Connect the ESP32 board to the computer with the USB cable.

Select board type "**ESP32 Dev Module**" and select port **COM-XX** (This depends on the number your computer assigns to the ESP32 board, which you can check it in the device manager).

```

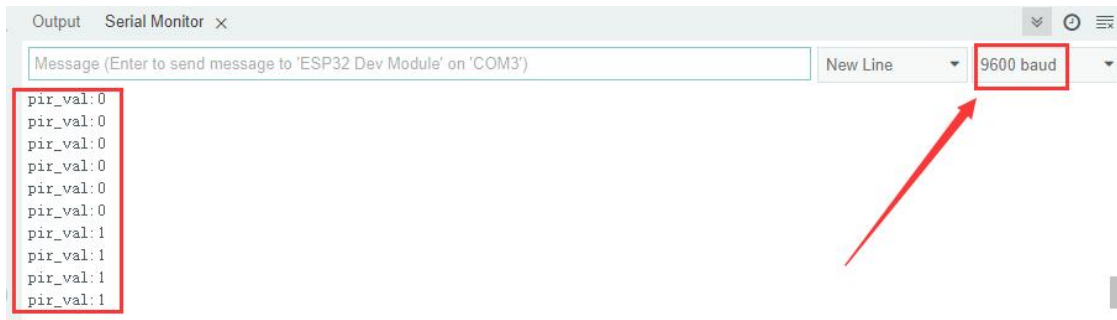
/*
  keyestudio ESP32 Inventor Learning Kit
  Project 17.1 Invasion Alarm
  http://www.keyestudio.com
*/
int pir = 5;    //Define I05 as PIR sensor pin
void setup() {
  pinMode(pir,INPUT);    //Set I05 pin to input
  Serial.begin(9600);
}

void loop() {
  int pir_val = digitalRead(pir);    //Read the PIR result and assign it to pir_val
  Serial.print("pir_val:"); //Print "pir_val"
  Serial.println(pir_val);
  delay(500);
}

```

## 5. Test Result

After connecting the wiring and uploading the code, open serial monitor to set baud rate to 9600, and the serial port prints the PIR value. If the PIR sensor detects a person, it will display 1.

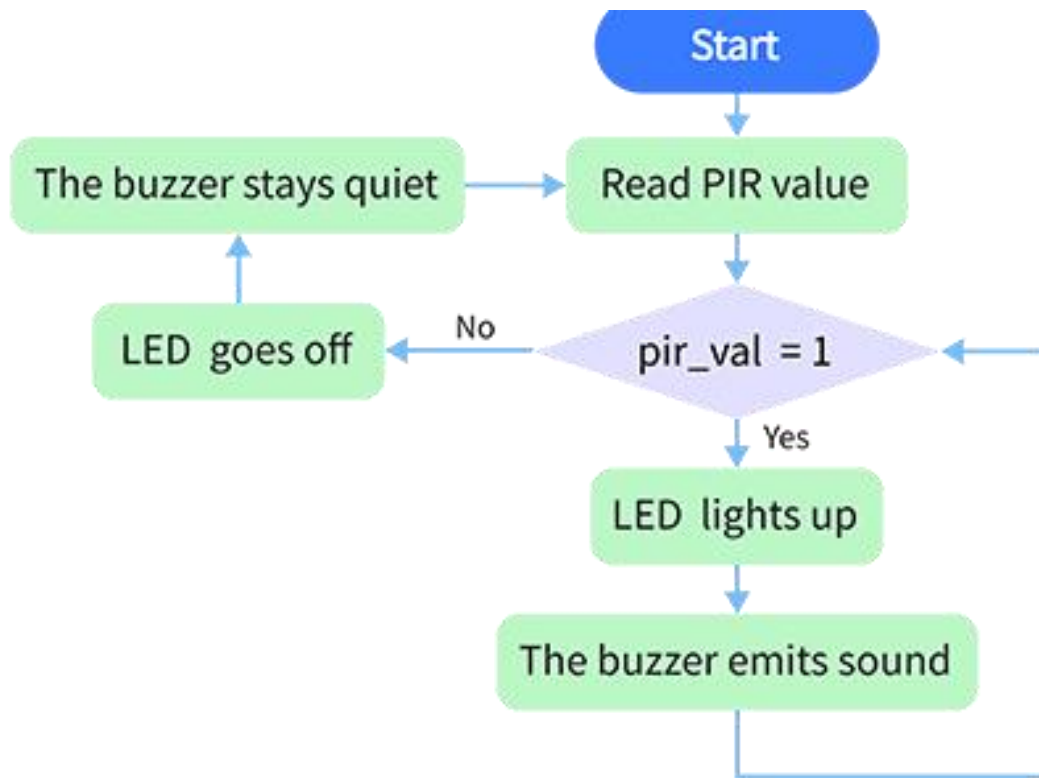


## 6. Knowledge Expansion

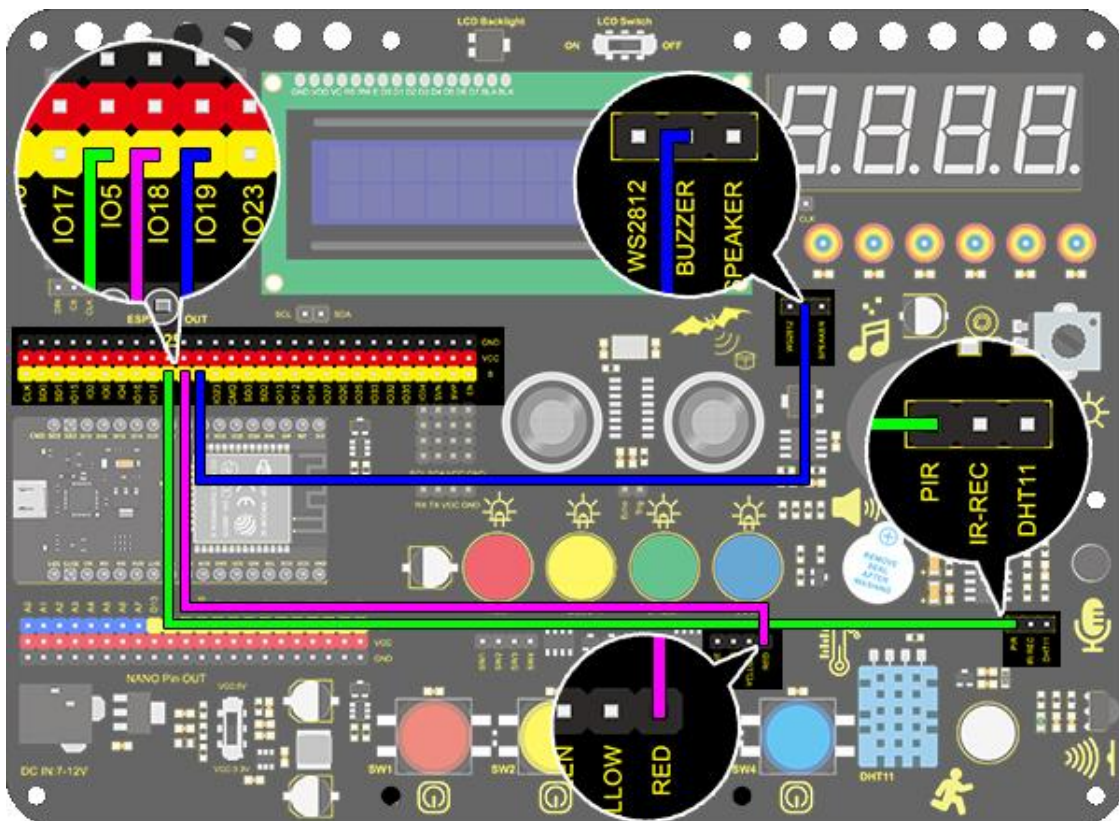
Let's make an invasion alarm. When the PIR sensor detects human, LED lights up and the buzzer emits sound. In contrast, LED goes off and the buzzer stays quiet.

**Flow Chart:**





**Wiring Diagram:**



## Code:

To fulfil an invasion alarm, an "if() else" statement is necessary.

Use the Arduino IDE to open the .ino format code [project\\_17.2](#) directly from the tutorial package.

Connect the ESP32 board to the computer with the USB cable.

Select board type "**ESP32 Dev Module**" and select port **COM-XX** (This depends on the number your computer assigns to the ESP32 board, which you can check it in the device manager).

```
/*
  keyestudio ESP32 Inventor Learning Kit
  Project 17.2 Invasion Alarm
  http://www.keyestudio.com
*/
int pir = 5;           //Set PIR sensor pin to IO5
int red_led = 18;      //Set red LED to pin IO18
int buzz = 19;         //Set buzzer to pin IO19

void setup() {
  // put your setup code here, to run once:
  pinMode(pir,INPUT);   //Set PIR pin to input mode
  pinMode(red_led,OUTPUT); //Set LED pin to output mode
  pinMode(buzz,OUTPUT); //Set buzzer pin to output mode
}

void loop() {
  // put your main code here, to run repeatedly:
  int pir_val = digitalRead(pir);
  if(pir_val == 1){
```

## Test Result

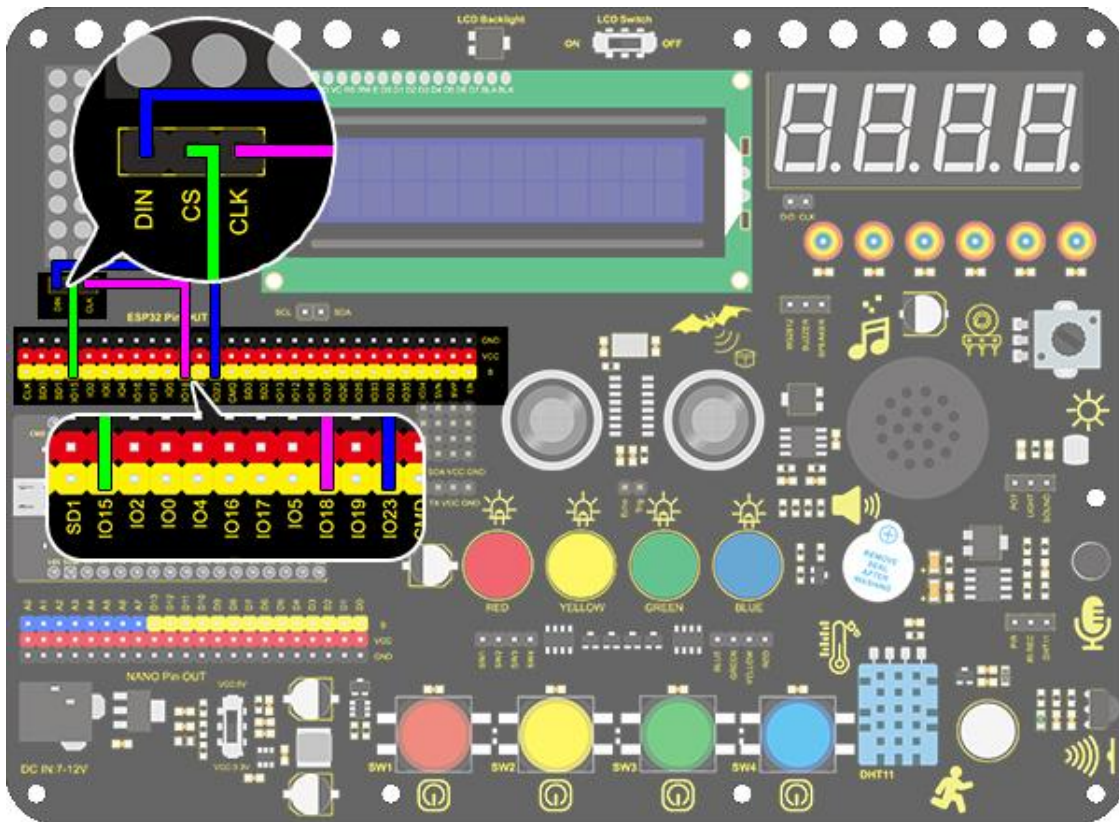
If the PIR sensor detects a person nearby, the red LED will light up and the buzzer will sound.

# Project 18: Beating Heart

## 1. Description

In this project, a beating heart will be presented via an Arduino board, a 8X8 dot matrix display, a circuit board and some electronic components. By programming, you can control the beating frequency, heart dimension and its brightness.

## 2. Wiring Diagram



## 3. Upload Code

When a large heart and a small heart switch display according to a certain time will give people a feeling of beating.

Use the Arduino IDE to open the .ino format code [project\\_18](#) directly from the tutorial package.

Connect the ESP32 board to the computer with the USB cable.

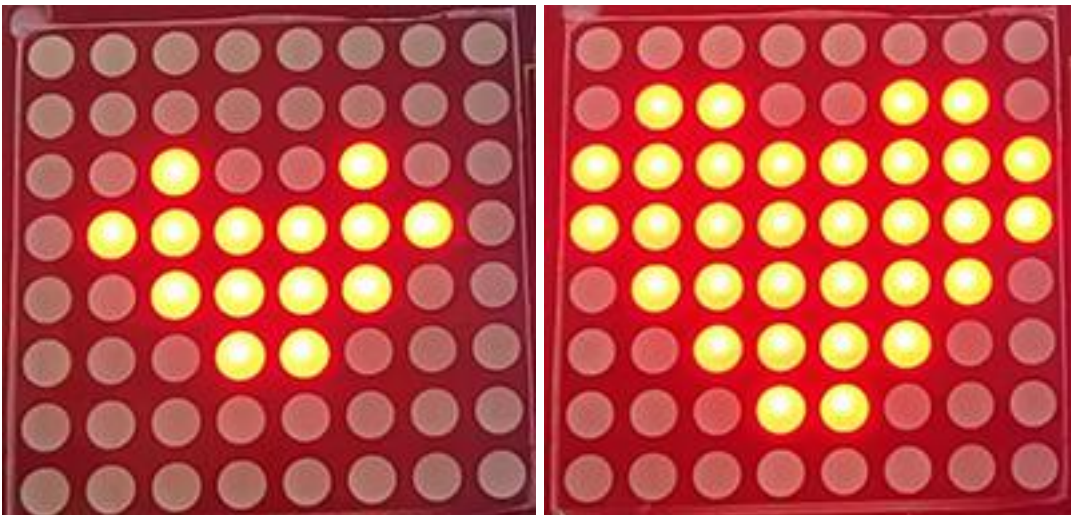
Select board type "**ESP32 Dev Module**" and select port **COM-XX** (This depends on the number your computer assigns to the ESP32 board, which you can check it in the device manager).

```
/*
  keyestudio ESP32 Inventor Learning Kit
  Project 18 Beating Heart
  http://www.keyestudio.com
*/

#include "LedControl.h"
int DIN = 23;
int CLK = 18;
int CS = 15;
LedControl lc=LedControl(DIN,CLK,CS,1);
const byte IMAGES1[] = {0x30, 0x78, 0x7c, 0x3e, 0x3e, 0x7c, 0x78, 0x30}; // a big heart
const byte IMAGES2[] = {0x00, 0x10, 0x38, 0x1c, 0x1c, 0x38, 0x10, 0x00}; //a small heart
void setup() {
  lc.shutdown(0,false);
  // Set brightness to a medium value
  lc.setIntensity(0,8);
  // Clear the display
  lc.clearDisplay(0);
```

#### 4. Test Result

After connecting the wiring and uploading code, the two sizes of hearts are displayed alternately.





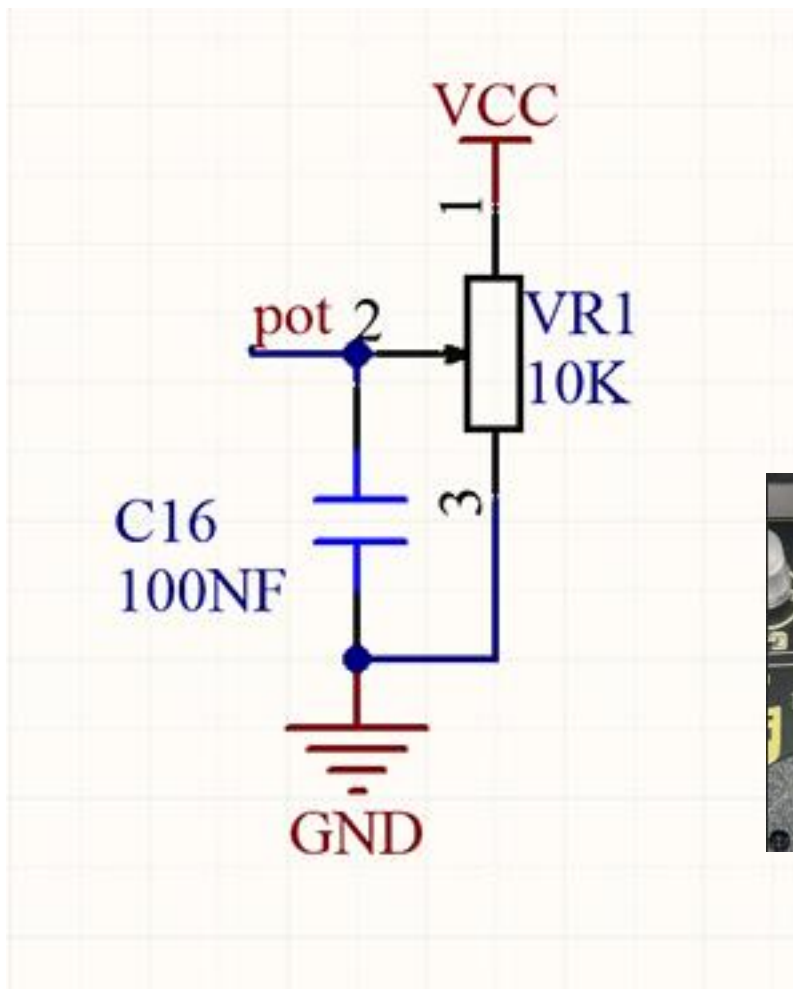
## Project 19 : Dimming Lamp

### 1. Description

The dimming lamp adjusts the brightness of LED via a potentiometer and an Arduino controller. The brightness is subject to resistance value, which can be read and adjusted by connecting the ends of the potentiometer to digital or analog pins on board.

What's more, this system is applied to control voltage or current of other devices such as fans, bulbs and heaters.

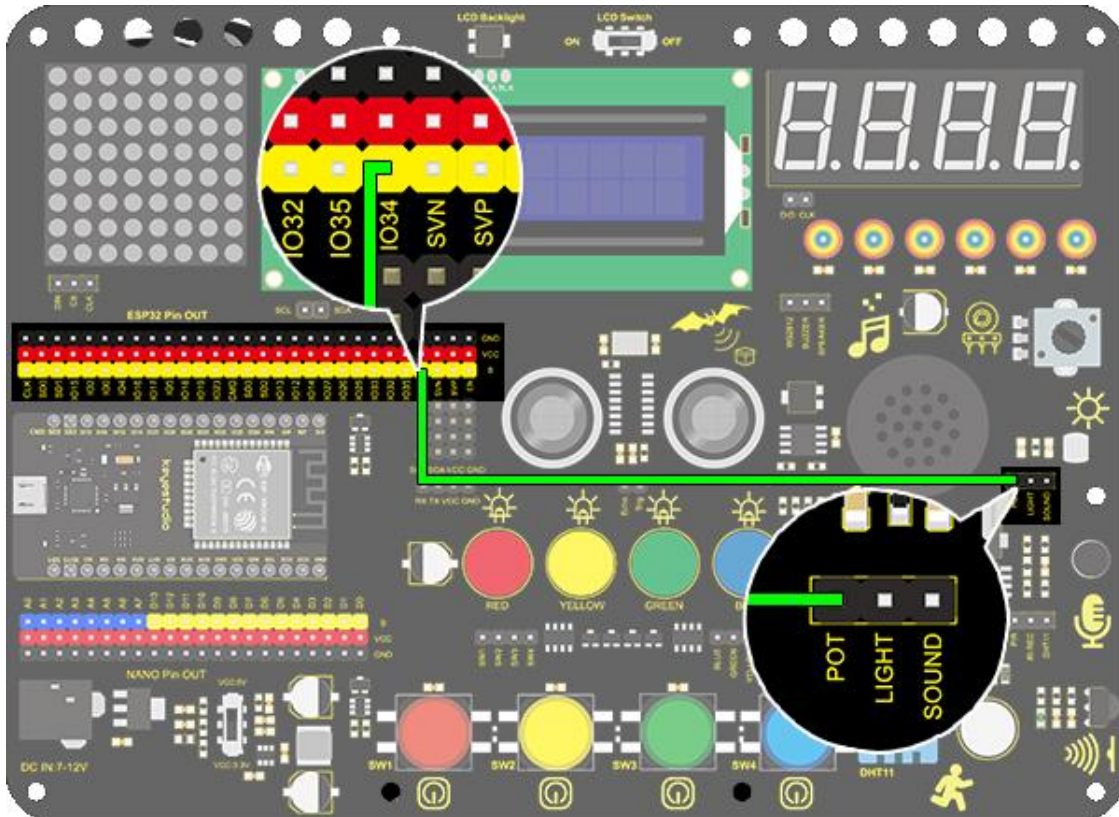
### 2. Working Principle of potentiometer



Essentially, potentiometer is an element that can change the value of resistance. According to Ohm's law ( $U=I \cdot R$ ), the resistance affects the voltage. Our potentiometer is 10K.

In this project, the maximum resistance is 10K. The ESP32 board will equally divide the voltage of 3V into 4095 parts ( $3/4095=0.0007326007326$ ). The analog voltage is obtained by multiplying the read value and 0.0007326007326.

### 3. Wiring Diagram



### 4. Upload Code

Here we adopt `analogRead(Pin)` to read the analog value of the potentiometer. Input the analog pin number connecting with the sensor into this function, and the analog value can be read.

Use the Arduino IDE to open the .ino format code [project\\_19.1](#) directly from the tutorial package.

Connect the ESP32 board to the computer with the USB cable.

Select board type "**ESP32 Dev Module**" and select port **COM-XX** (This depends on the number your computer assigns to the ESP32 board, which you can check it in the device manager).



```

/*
  keyestudio ESP32 Inventor Learning Kit
  Project 19.1 Dimming Lamp
  http://www.keyestudio.com
*/
int pot = 34;      //Define variable pot to IO34
void setup() {
  // put your setup code here, to run once:
  Serial.begin(9600);      //Set baud rate to 9600
}

void loop() {
  // put your main code here, to run repeatedly:
  int value = analogRead(pot); //Read io34 and assign it to the variable value
  Serial.println(value);        //Print the variable value and wrap it around
  delay(200);
}

```

## 5. Test Result

After connecting the wiring and uploading code, open serial monitor to set baud rate to 9600, and serial monitor will display the analog value of the potentiometer will be displayed, within the range of 0-4095. Rotating the potentiometer can change the size of the analog value.

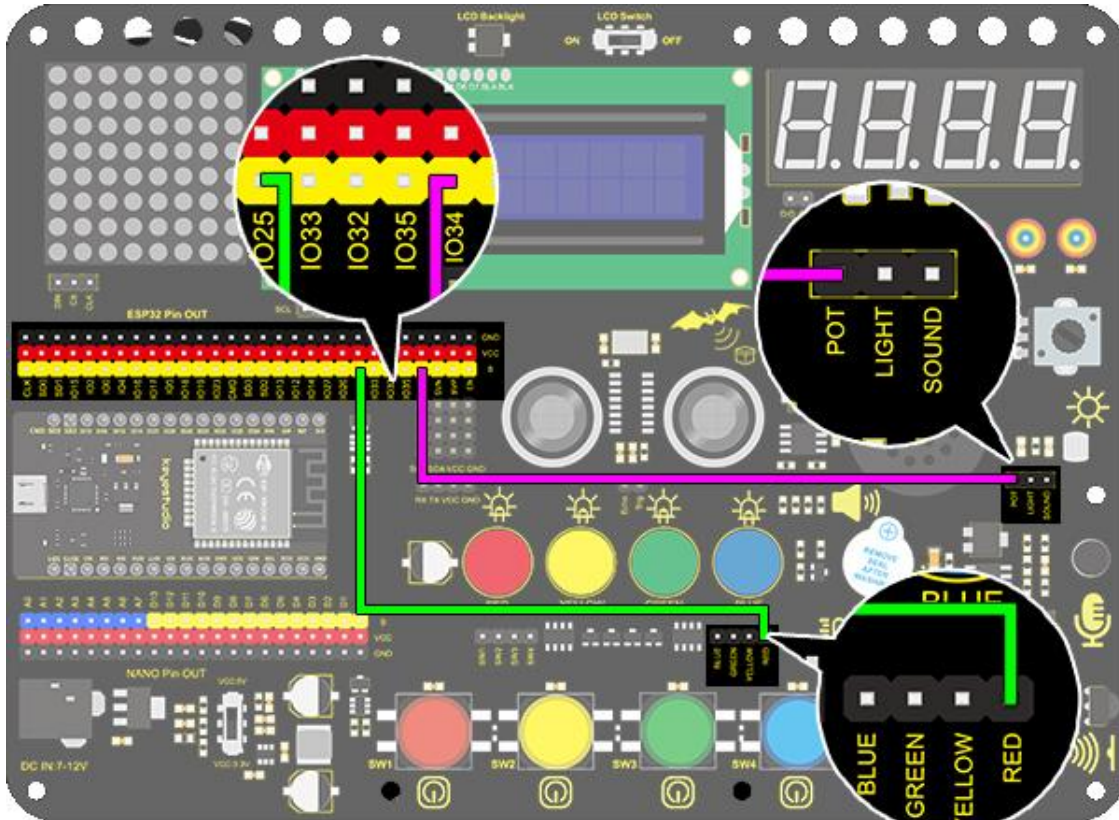


## 6. Knowledge Expansion

We will control the brightness of LED via a potentiometer.

As we know, it is influenced by PWM. However, the range of analog value is 0-4095 while that of PWM is 0-255. Thus, a "map(value, fromLow, fromHigh, toLow, toHigh)" function is needed.

## Wiring Diagram:



## Code:

Use the Arduino IDE to open the .ino format code [project\\_19.2](#) directly from the tutorial package.

Connect the ESP32 board to the computer with the USB cable.

Select board type "**ESP32 Dev Module**" and select port **COM-XX** (This depends on the number your computer assigns to the ESP32 board, which you can check it in the device manager).

```

/*
  keyestudio ESP32 Inventor Learning Kit
  Project 19.2 Dimming Lamp
  http://www.keyestudio.com
*/
int led = 25;          //Define LED to I025
int pot = 34;          //Define pot to I034
void setup() {
  // put your setup code here, to run once:
  pinMode(led,OUTPUT); //Set LED pin to output
}

void loop() {
  // put your main code here, to run repeatedly:
  int value = analogRead(pot);
  int led_val = map(value,0,4095,0,255); //Convert the range of potentiometer analog value to the range
  analogWrite(led,led_val);
}

```

## Test Result

After the code is uploaded successfully, rotating the potentiometer will change the brightness of the red LED.

## 7. Code Explanation

**analogRead(pot);** Read the analog value. Put the input pin of analog value in brackets.

**map(value, fromLow, fromHigh, toLow, toHigh)** map(value,0,4095,0,255);  
Convert the range of value from 0-4095 to 0-255. Because the range of value does not conform to that of PWM, a conversion is necessary.

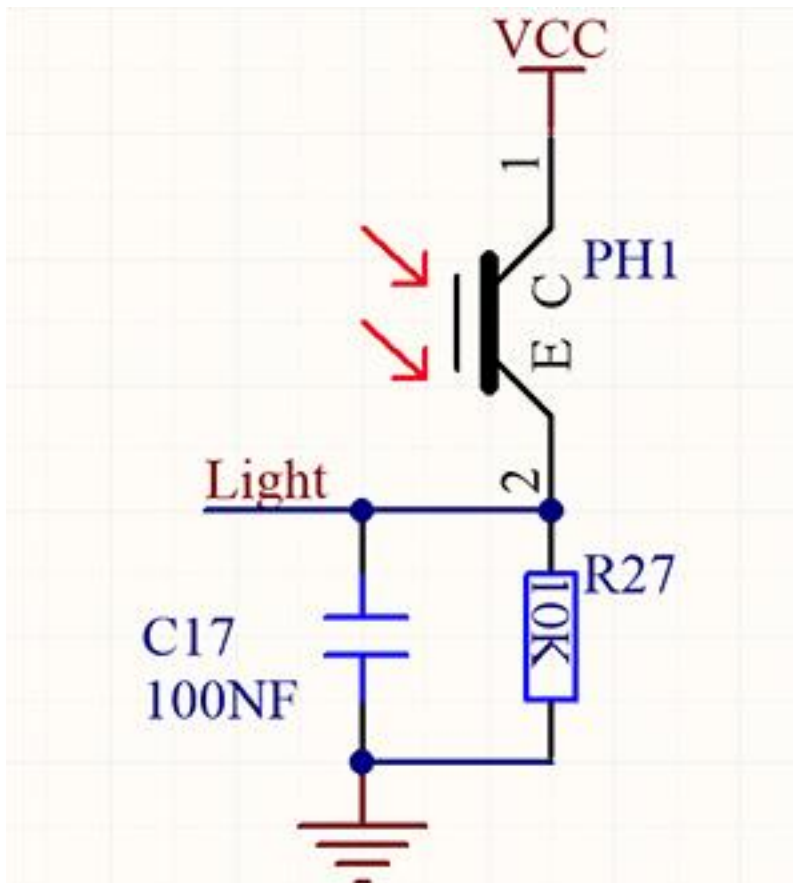
# Project 20: Light Pillar

## 1. Description

The resistance (less than  $1\text{K}\Omega$ ) of the photoresistor varies from the light, thus it can control the brightness of the dot matrix. When controlling, we connect this resistor to an analog pin on the board to monitor the change of resistance. In this way, the light automatically controls the brightness of the display.

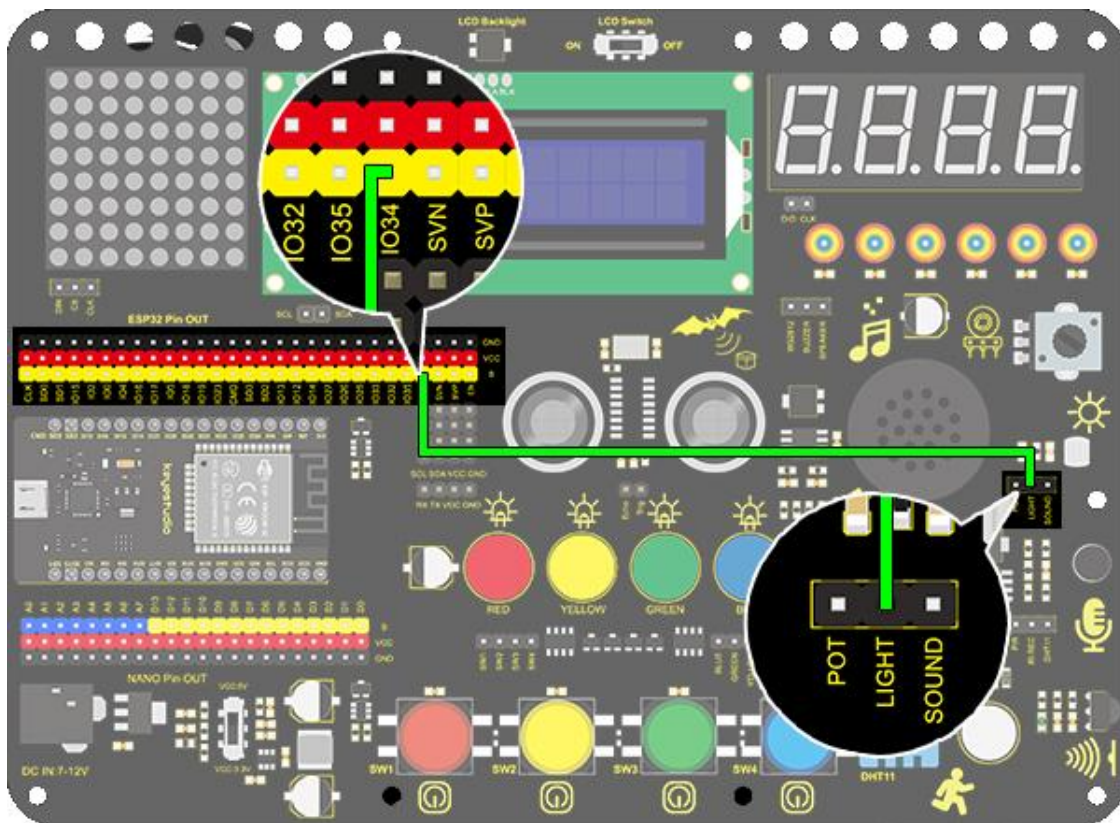
Besides, the photoresistor is widely applied to our daily life. For instance, a curtain automatically opens or closes according to the outer light intensity.

## 2. Working Principle



When it is totally in dark, the resistance equals  $0.2\text{M}\Omega$ , and the voltage at signal terminal (point 2) approaches to 0V. The stronger the light is, the smaller the resistance and voltage will be.

### 3. Wiring Diagram



### 4. Upload Code

We adopt the `analogRead(Pin)` function to read the analog value. Connect the sensor to IO34 pin, and the value will be printed on the serial monitor.

Use the Arduino IDE to open the .ino format code [project\\_20.1](#) directly from the tutorial package.

Connect the ESP32 board to the computer with the USB cable.

Select board type "**ESP32 Dev Module**" and select port **COM-XX** (This depends on the number your computer assigns to the ESP32 board, which you can check in the device manager).



```

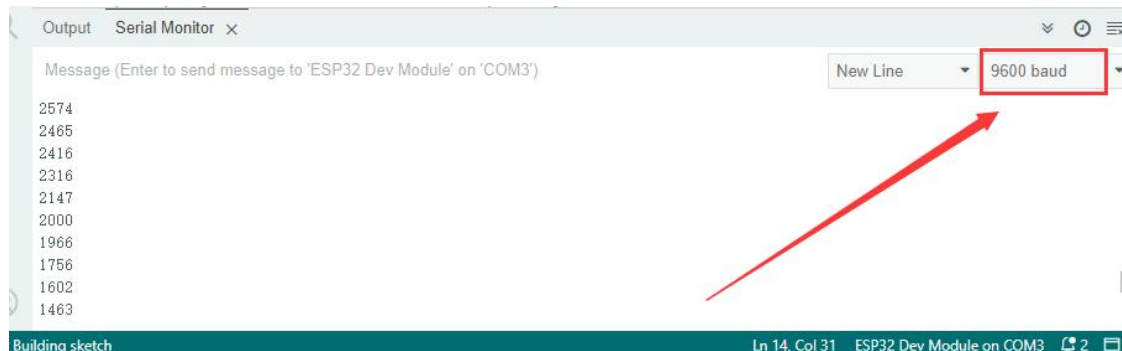
/*
  keyestudio ESP32 Inventor Learning Kit
  Project 20.1 Light Pillar
  http://www.keyestudio.com
*/
int light = 34;      //Define light to I034
void setup() {
  // put your setup code here, to run once:
  Serial.begin(9600);      //Set baud rate to 9600
}

void loop() {
  // put your main code here, to run repeatedly:
  int value = analogRead(light);      //Read I034 and assign it to the variable value
  Serial.println(value);      //Print the variable value and wrap it around
  delay(200);
}

```

## 5. Test Result

After connecting the wiring and uploading code, open serial monitor to set baud rate to 9600, the analog value of the photoresistor will be displayed, withing the range of 0-4095. Changing the light intensity around it can change its value.

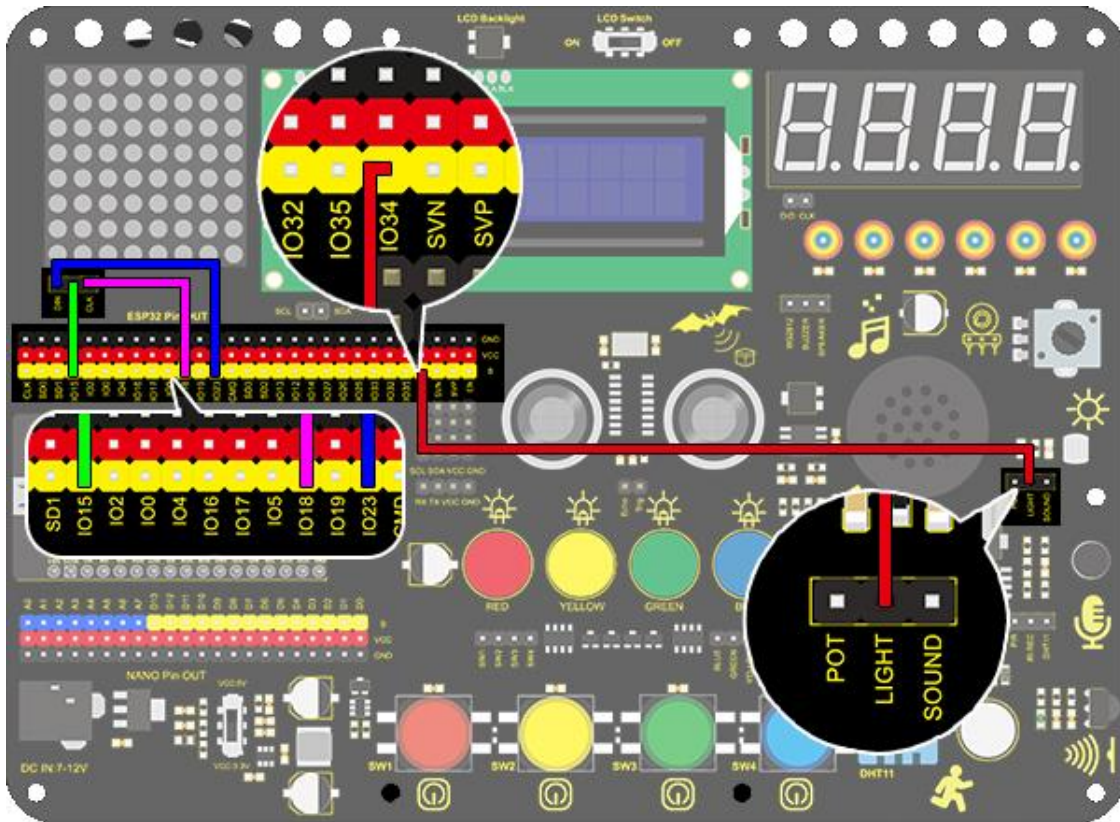


## 6. Knowledge Expansion

We will use this photoresistor to sense the ambient light intensity. The two columns of middle are included in this experiment to represent light intensity. The stronger it is, the more lighted LEDs will be. This forms a "light pillar".

### Wiring Diagram:





### Code:

Use the Arduino IDE to open the .ino format code [project\\_20.2](#) directly from the tutorial package.

Connect the ESP32 board to the computer with the USB cable.

Select board type "**ESP32 Dev Module**" and select port **COM-XX** (This depends on the number your computer assigns to the ESP32 board, which you can check it in the device manager).

```

/*
  keyestudio ESP32 Inventor Learning Kit
  Project 20.2 Light Pillar
  http://www.keyestudio.com
*/

#include "LedControl.h"
int DIN = 23;
int CLK = 18;
int CS = 15;
LedControl lc=LedControl(DIN,CLK,CS,1);
const byte IMAGES[8] = {0x01,0x03,0x07,0x0F,0x1F,0x3F,0x7F,0xFF}; //Data of light pillar

int light = 34;

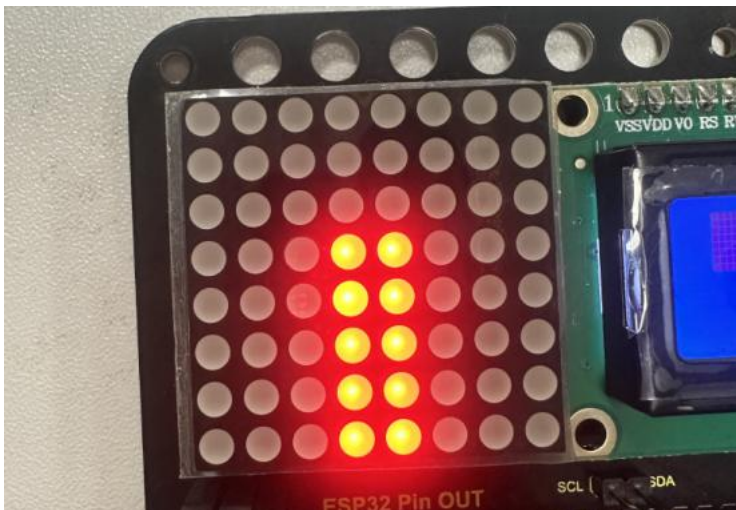
void setup() {
  lc.shutdown(0,false);
  // Set brightness to a medium value
  lc.setIntensity(0,8);
  // Clear the display
  lc.clearDisplay(0);
  pinMode(light,INPUT);
}

void loop(){
  int value = analogRead(light);

```

## Test Result:

The stronger the light near the photoresistor, the higher the light column of the LED matrix.

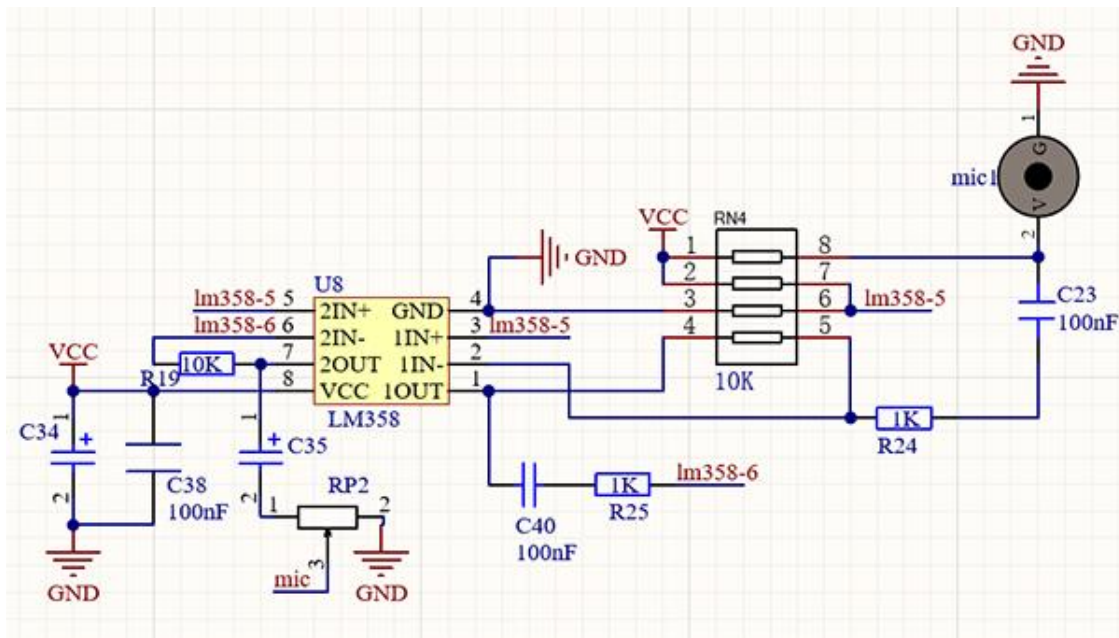


# Project 21: Sound Controlled LED

## 1. Description

Sound controlled LED is a device used to detect sound in a way that controls the brightness of LED, which is composed of a Arduino board and some components. It can connect to multiple sensors such as microphones. It converts sound to changing voltage signal to be received by Arduino to control the LED on and off.

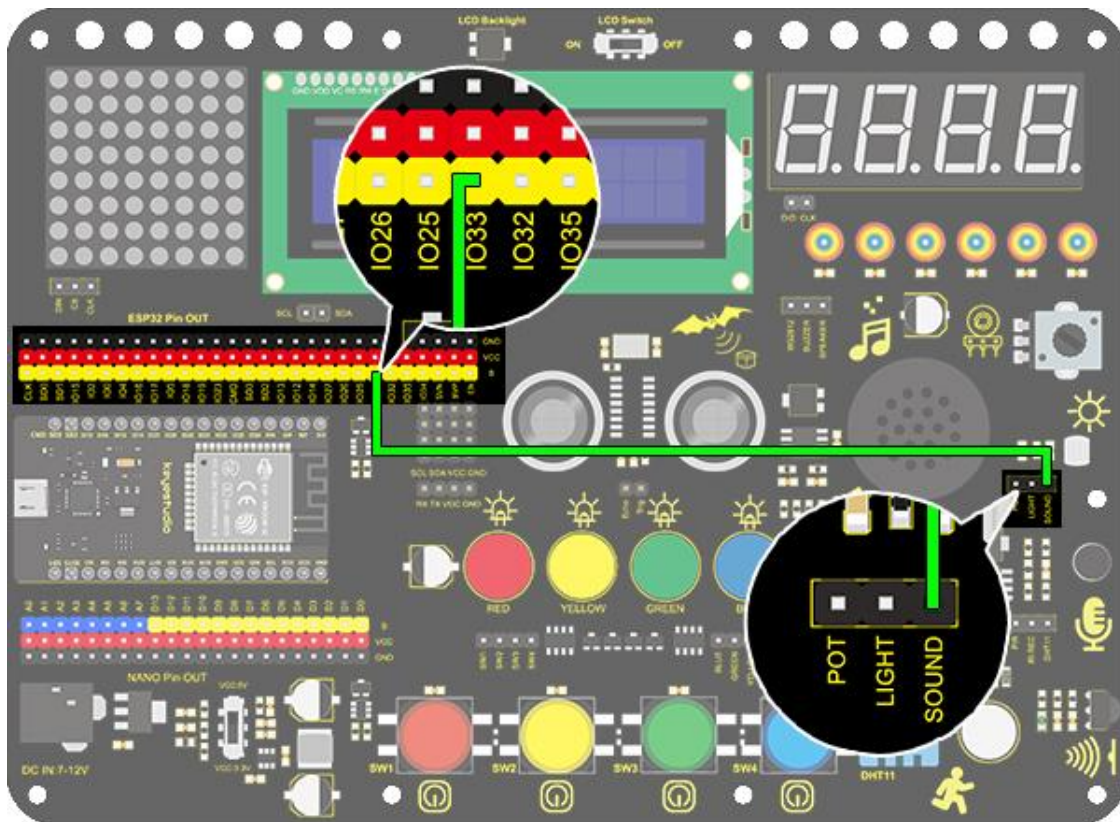
## 2. Working Principle



When detecting a sound, the electret film in microphone vibrates, which changes the capacitance and generates a subtle change of voltage.

Next, we make use of LM3 chip to build a proper circuit to amplify the detected sound up, which can be adjusted by a potentiometer. Rotate it clockwise to enlarge the times.

## 3. Wiring Diagram



#### 4. Upload Code

Connect the sensor to pin IO33. Read the sound analog value through `analogRead(Pin)` function and print it on serial monitor.

Use the Arduino IDE to open the .ino format code [project\\_21.1](#) directly from the tutorial package.

Connect the ESP32 board to the computer with the USB cable.

Select board type "**ESP32 Dev Module**" and select port **COM-XX** (This depends on the number your computer assigns to the ESP32 board, which you can check it in the device manager).

```

/*
  keyestudio ESP32 Inventor Learning Kit
  Project 21.1: Sound Controlled LED
  http://www.keyestudio.com
*/
int sound = 33; //Define sound as I033
void setup(){
  Serial.begin(9600);
  pinMode(sound, INPUT);
}

void loop(){
  int value = analogRead(sound);
  Serial.println(value);
}

```

## 5. Test Result

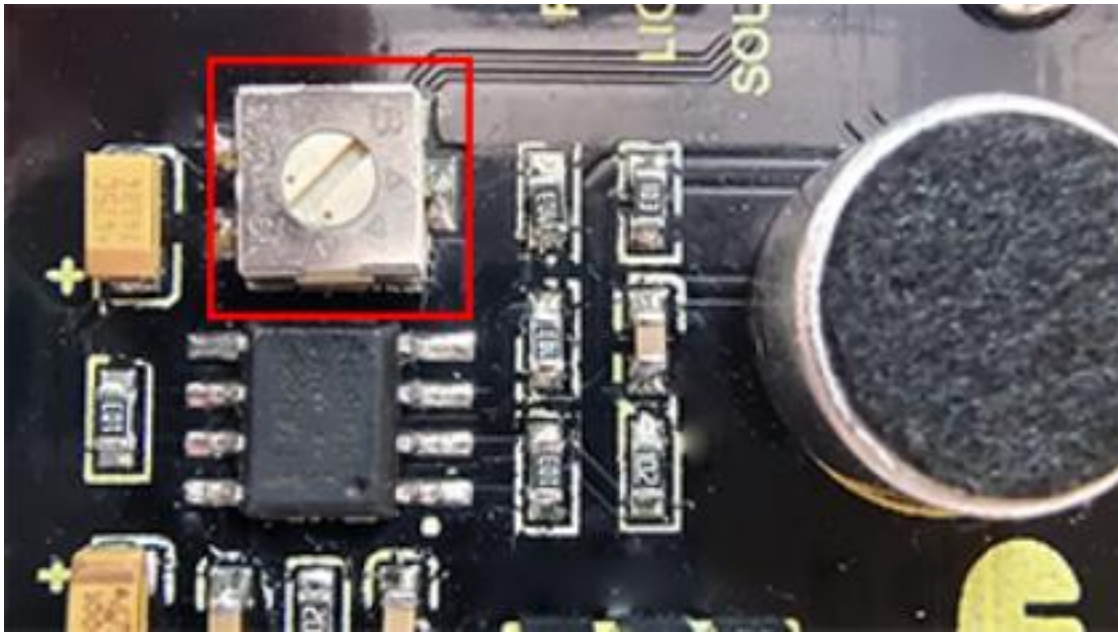
After connecting the wiring and uploading code, open serial monitor to set baud rate to 9600, the analog value of the sound sensor will be displayed.



### Sensitivity adjustment:

If you think the sound sensor is not sensitive enough, you can use the provided flat-blade screwdriver to rotate the potentiometer (right is the highest, left is the lowest) to adjust its sensitivity.



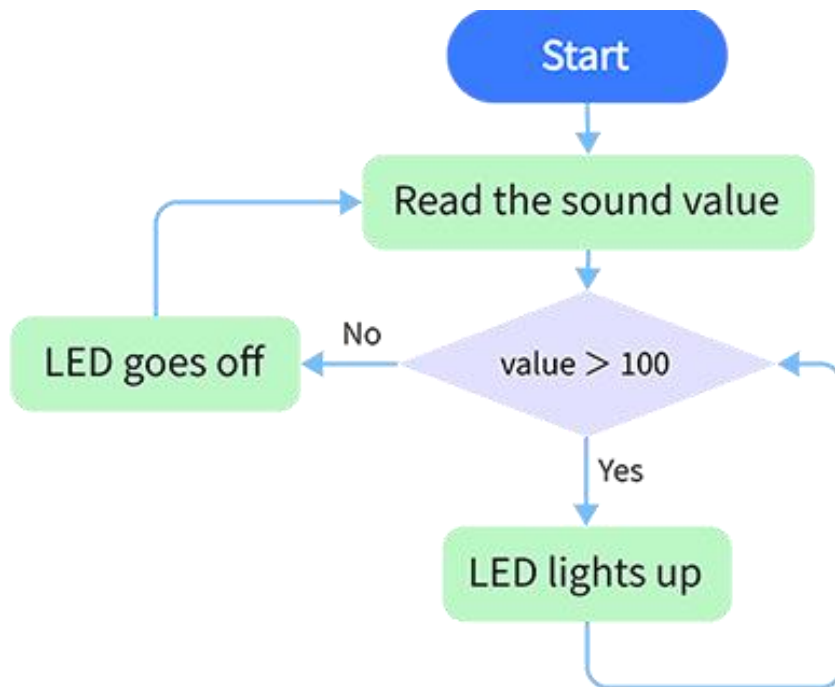


## 6. Knowledge Expansion

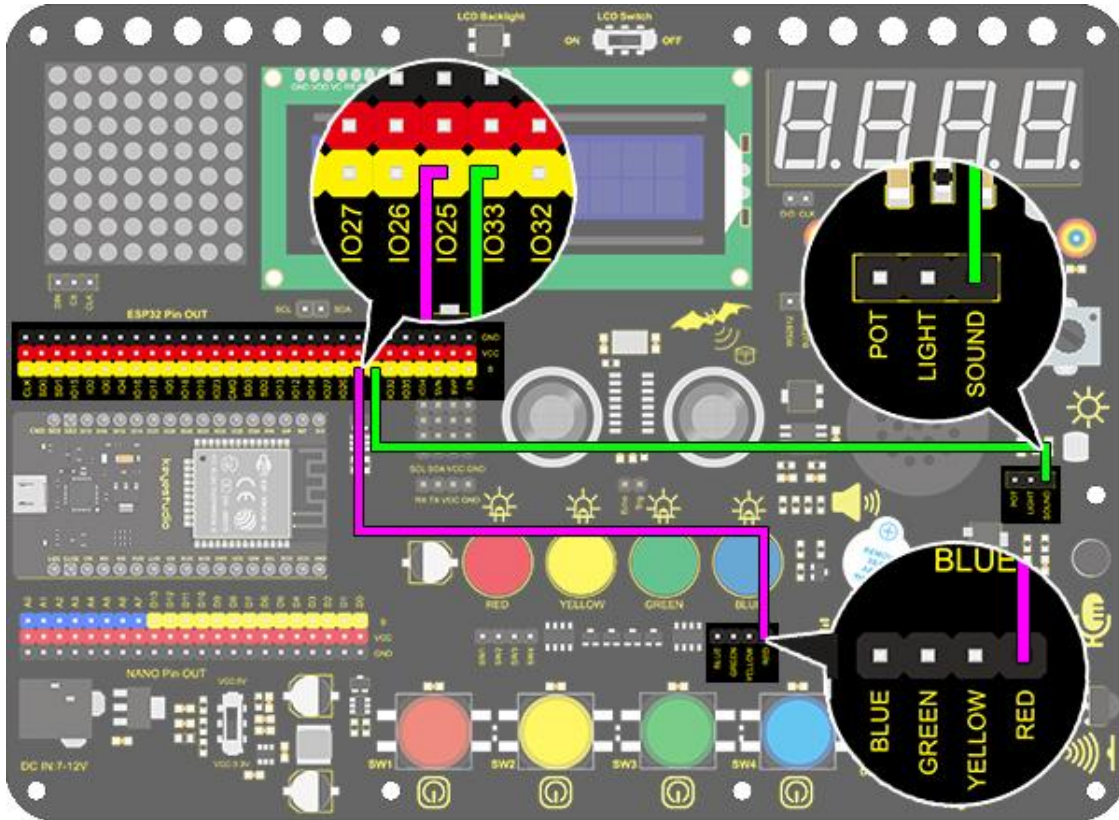
The commonly seen corridor light is a kind of sound controlled light. Meanwhile, it also includes a photoresistor.

Differed from that, here we establish a model that an LED only is affected by sound. When the analog volume exceeds 100, LED lights up for 2S and then goes off.

### Flow Chart:



## Wiring Diagram:



## Code:

Use the Arduino IDE to open the .ino format code [project\\_21.2](#) directly from the tutorial package.

Connect the ESP32 board to the computer with the USB cable.

Select board type "**ESP32 Dev Module**" and select port **COM-XX** (This depends on the number your computer assigns to the ESP32 board, which you can check it in the device manager).

```

/*
  keyestudio ESP32 Inventor Learning Kit
  Project 21.2: Sound Controlled LED
  http://www.keyestudio.com
*/
int sound = 33; //Define sound to I033
int led = 25;   //Define led to I025
void setup(){
  pinMode(led,OUTPUT); //Set I025 to output
}

void loop(){
  int value = analogRead(sound); //Read analog value of I033 and assign it to value
  if(value > 100){                //Judge whether value is greater than 100
    digitalWrite(led,HIGH);       //If I025 pin outputs high level, LED lights up
    delay(2000);
  }
  else{
    digitalWrite(led,LOW);        //If I025 pin outputs low level, LED lights off
  }
}

```

### Test Result:

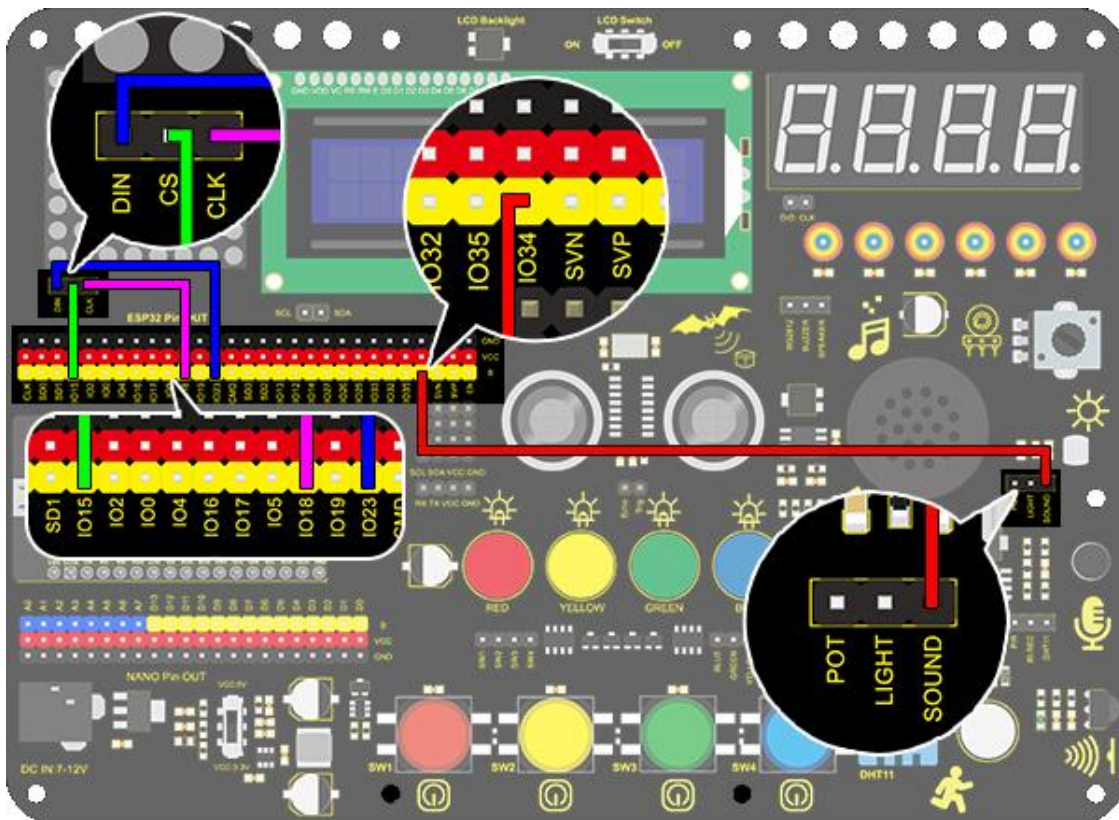
When the value detected by the sound sensor is greater than 100, the red LED will light up.

# Project 22: Noise Meter

## 1. Description

The noise meter will be able to use the number of dots on the LED matrix to reflect the size of the noise.

## 2. Wiring Diagram



## 3. Upload Code

The noise meter is able to detect the ambient noise.

Use the Arduino IDE to open the .ino format code [project\\_22](#) directly from the tutorial package.

Connect the ESP32 board to the computer with the USB cable.



Select board type "**ESP32 Dev Module**" and select port **COM-XX** (This depends on the number your computer assigns to the ESP32 board, which you can check it in the device manager).

```
/*
  keyestudio ESP32 Inventor Learning Kit
  Project 22 Noisemeter
  http://www.keyestudio.com
*/
#include <LedControl.h>

int DIN = 23;
int CLK = 18;
int CS = 15;
int sensor = 34;

LedControl lc=LedControl(DIN,CLK,CS,1);
byte data_val[8][8]= {
  {0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x01},
  {0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x03, 0x01},
  {0x00, 0x00, 0x00, 0x00, 0x00, 0x07, 0x03, 0x01},
  {0x00, 0x00, 0x00, 0x00, 0x0f, 0x07, 0x03, 0x01},
  {0x00, 0x00, 0x00, 0x1f, 0x0f, 0x07, 0x03, 0x01},
  {0x00, 0x00, 0x3f, 0x1f, 0x0f, 0x07, 0x03, 0x01},
  {0x00, 0x7f, 0x3f, 0x1f, 0x0f, 0x07, 0x03, 0x01},
  {0xff, 0x7f, 0x3f, 0x1f, 0x0f, 0x07, 0x03, 0x01}
};
```

#### 4. Upload Code

The greater the sound value detected by the sound sensor, the more dots light up on the LED matrix.



#### 5. Code Explanation

**data\_val[ ] [ ] { ... };** Two-dimensional array. If we use an axis X metaphor for linear array, two-dimensional array is axis X and Y. In this code, the value in the first square brackets is on axis X, and the second is on axis Y. For instance, column 3 and row 4, that is **data\_val[ 3] [4 ]**.



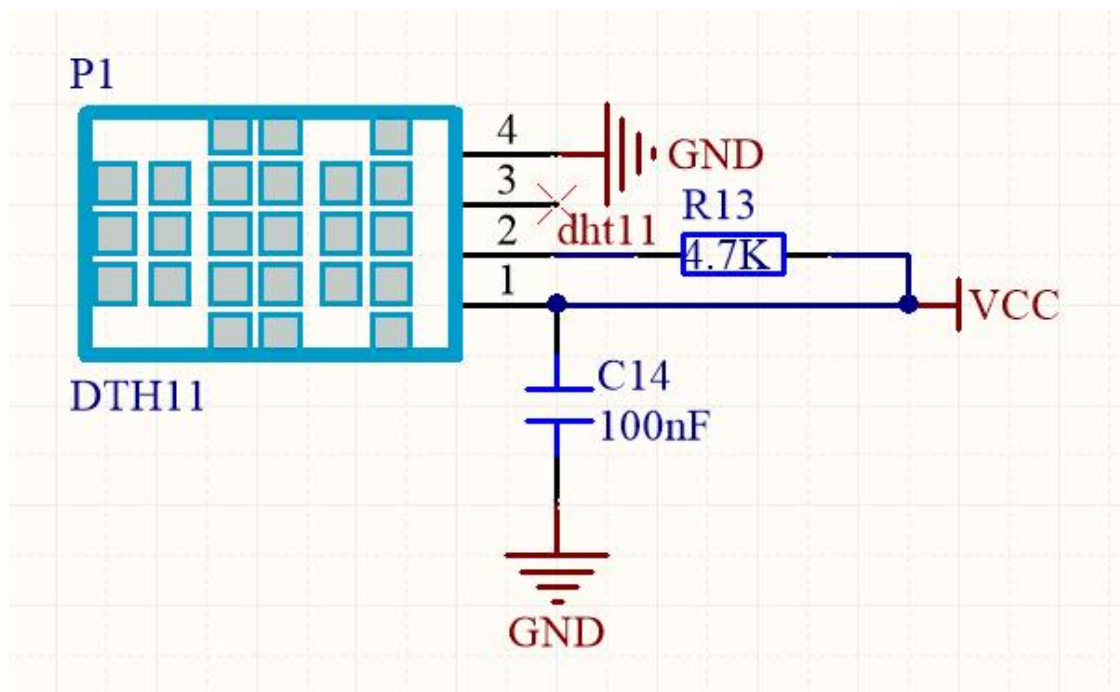
## Project 23: Smart Cup

### 1. Description

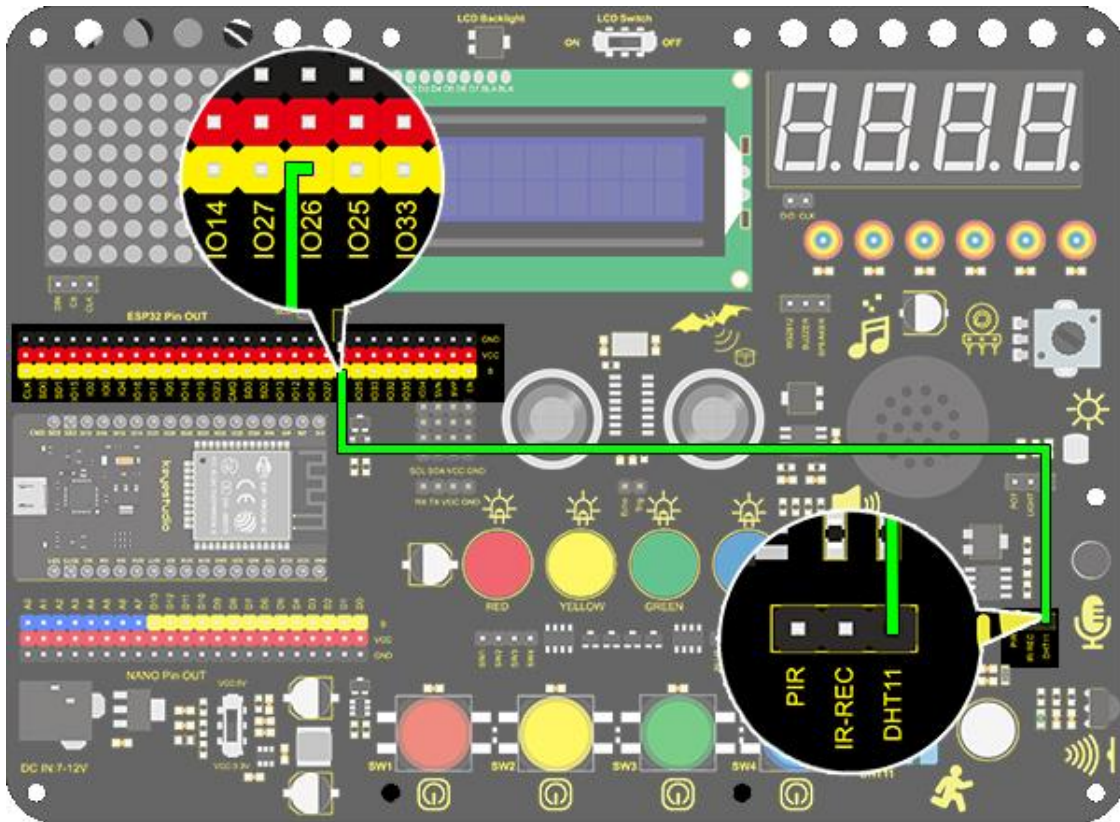
In this project, we mainly adopt the Arduino development board to create a programmable smart cup, which reveals the temperature of inner liquid through a RGB indicator. It can control the brightness of the light by setting a temperature threshold. If the threshold is exceeded, it will get brighter. Otherwise, it gets darker.

The smart cup enables to help users better control the temperature of their drinking water and effectively prevent overheating or freezing.

### 2. Working Principle



### 3. Wiring Diagram



#### 4. Upload Code

Use the Arduino IDE to open the .ino format code [project\\_23.1](#) directly from the tutorial package.

Connect the ESP32 board to the computer with the USB cable.

Select board type "**ESP32 Dev Module**" and select port **COM-XX** (This depends on the number your computer assigns to the ESP32 board, which you can check it in the device manager).

```

/*
  keyestudio ESP32 Inventor Learning Kit
  Project 23.1 Smart Cup
  http://www.keyestudio.com
*/
#include <xht11.h>
xht11 xht(26); //The DHT11 sensor connects to I026
unsigned char dat[] = {0,0,0,0}; //Define an array to store temperature and humidity data

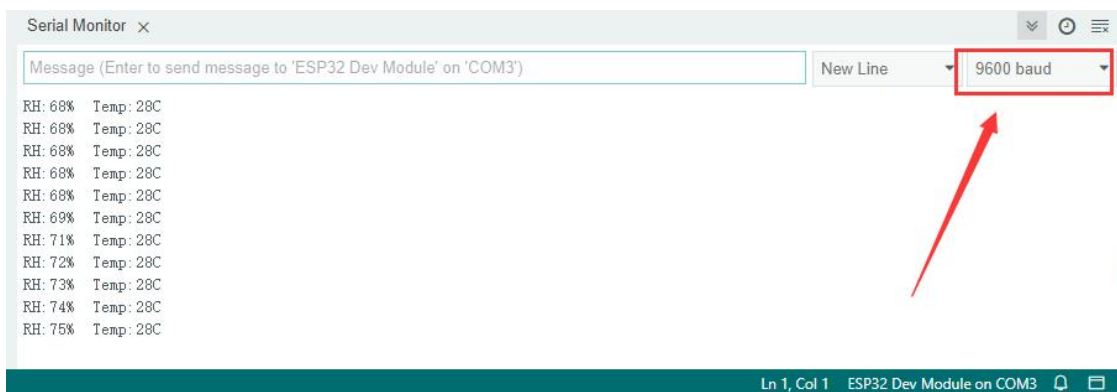
void setup() {
  // put your setup code here, to run once:
  Serial.begin(9600);
}

void loop() {
  // put your main code here, to run repeatedly:
  if (xht.receive(dat)) { //Check correct return to true
    Serial.print("RH:");
    Serial.print(dat[0]); //The integral part of humidity,dht[1] is the decimal part
    Serial.print("% ");
    Serial.print("Temp:");
    Serial.print(dat[2]); //The integer part of the temperature,dht[3] is the decimal part
    Serial.println("C");
  } else { //Read error
    Serial.print("Error");
  }
}

```

## 5. Test Result

After connecting the wiring and uploading code, open serial monitor to set baud rate to 9600, and the temperature and humidity value will be displayed.



## 6. Knowledge Expansion

Now, we will make a smart cup which can show liquid temperature. We use 4 LEDs to divide  $100^{\circ}$  into four levels, as shown below:

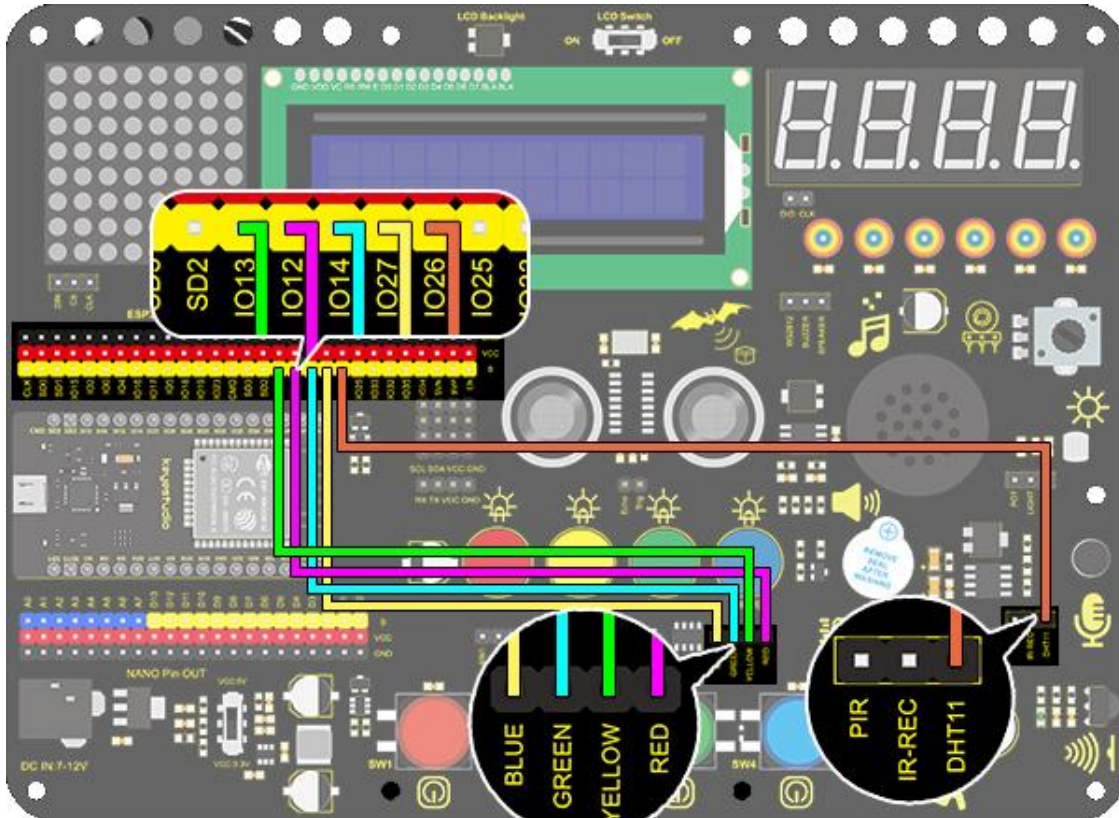
**Red LED:** 100-75°C

**Yellow LED: 75-50°C**

**Green LED: 50-25°C**

### Blue LED: 25-0°C

### Wiring Diagram:



**Code:**

Use the Arduino IDE to open the .ino format code [project\\_23.2](#) directly from the tutorial package.

Connect the ESP32 board to the computer with the USB cable.

Select board type "**ESP32 Dev Module**" and select port **COM-XX** (This depends on the number your computer assigns to the ESP32 board, which you can check it in the device manager).

```

/*
  keyestudio ESP32 ESP32 Inventor Learning Kit
  Project 23.2 Smart Cup
  http://www.keyestudio.com
*/
#include <xht11.h>
xht11 xht(26); //Define DHT11 to pin IO26
unsigned char dat[4] = { 0, 0, 0, 0 }; //Define an array to store temperature and humidity data

int red_led = 12;
int yellow_led = 13; //Define yellow_led to io13
int green_led = 14; //Define green_led to io14
int blue_led = 27; //Define blue_led to io27
int temperature = 0; //Set an variable to save the temperature value
void setup() {
  // put your setup code here, to run once:
  pinMode(red_led, OUTPUT); //Set io12 to ouput
  pinMode(green_led, OUTPUT); //Set io13 to ouput
  pinMode(blue_led, OUTPUT); //Set io14 to ouput
  pinMode(yellow_led, OUTPUT); //Set io27 to ouput
  Serial.begin(9600);
}

```

Test Result:

**Red LED:** 100-75°C

**Yellow LED:** 75-50°C

**Green LED:** 50-25°C

**Blue LED:** 25-0°C

If the blue LED is on, it means the temperature detected by the DHT11 sensor is in the range of 0-25°.

## 7. Code Explanation

**xht11 xht(Pin);** Set the instance named xht and add the pins

**unsigned char dat[4] = { 0, 0, 0, 0 };** dat[0] is the integer part of the humidity value. dat[1] is the decimal part of the humidity value. dat[2] is the integer part of the temperature value, and dat[3] is the decimal part of the temperature value

**&&** (value < 100 && value > 75) means that, it is true only both expressions satisfying the condition, or else it is false.

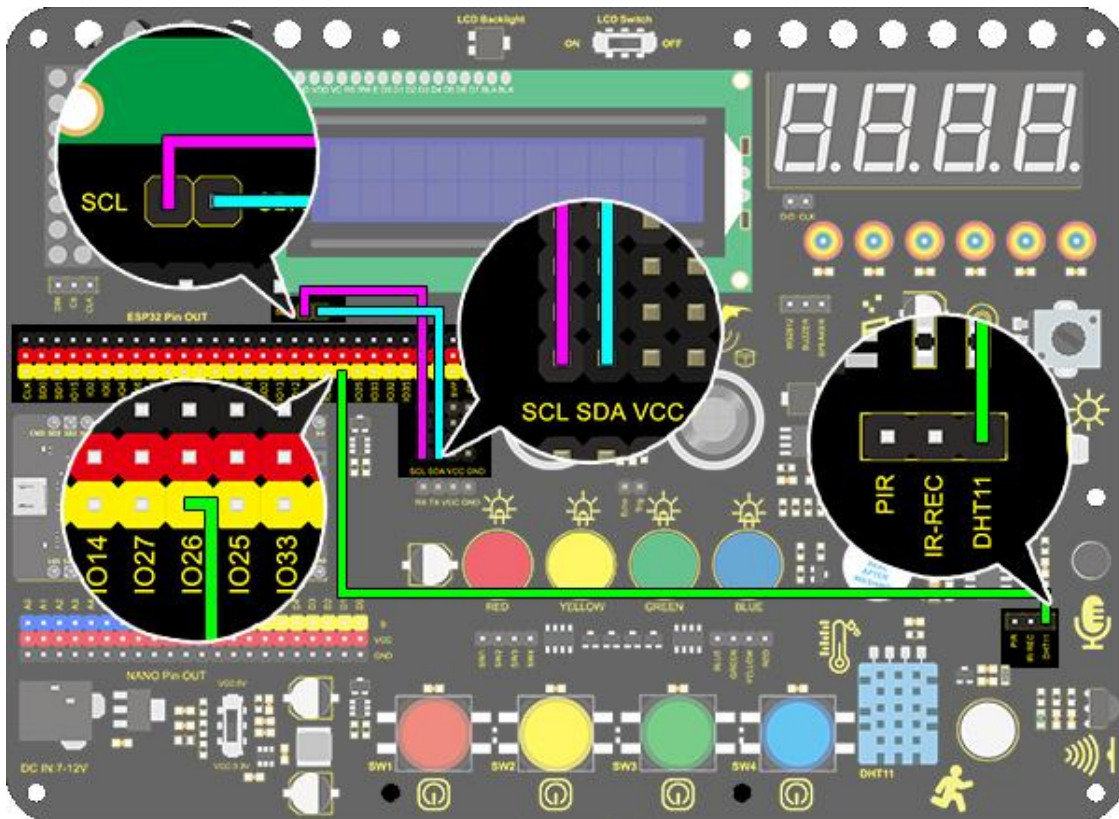


# Project 24: Weather Station

## 1. Description

This weather station records the ambient temperature and humidity value via Arduino board and a temperature and humidity sensor.

## 2. Wiring Diagram



## 3. Upload Code

It is a simple weather device that responds to ambient humidity and temperature

Use the Arduino IDE to open the .ino format code [project\\_24](#) directly from the tutorial package.

Connect the ESP32 board to the computer with the USB cable.

Select board type "**ESP32 Dev Module**" and select port **COM-XX** (This depends on the number your computer assigns to the ESP32 board, which you can check it in the device manager).

```

/*
  keyestudio ESP32 Inventor Learning Kit
  Project 24: Weather Station
  http://www.keyestudio.com
*/
#include <LiquidCrystal_I2C.h>
#include <xht11.h>
LiquidCrystal_I2C lcd(0x27, 16,2); // set the LCD address to 0x27 for a 16 chars and 2 line display
xht11 xht(26); //The DHT11 sensor connects to IO26
unsigned char dat[] = { 0, 0, 0, 0 }; //Define an array to store temperature and humidity data

void setup() {
  lcd.init(); // initialize the lcd
  lcd.backlight();
}

void loop() {
  if (xht.receive(dat)) { //Check correct return to true
    lcd.setCursor(0, 0);
    lcd.print("humidity:");
    lcd.setCursor(9, 0);
    lcd.print(dat[0]);
    lcd.setCursor(0, 1);
    lcd.print("temperature:");
    lcd.setCursor(12, 1);
    lcd.print(dat[2]);
  }
}

```

#### 4. Test Result

After connecting the wiring and uploading code, the LCD display will directly display the ambient humidity and temperature value.



# Project 25: Ultrasonic Rangefinder

## 1. Description

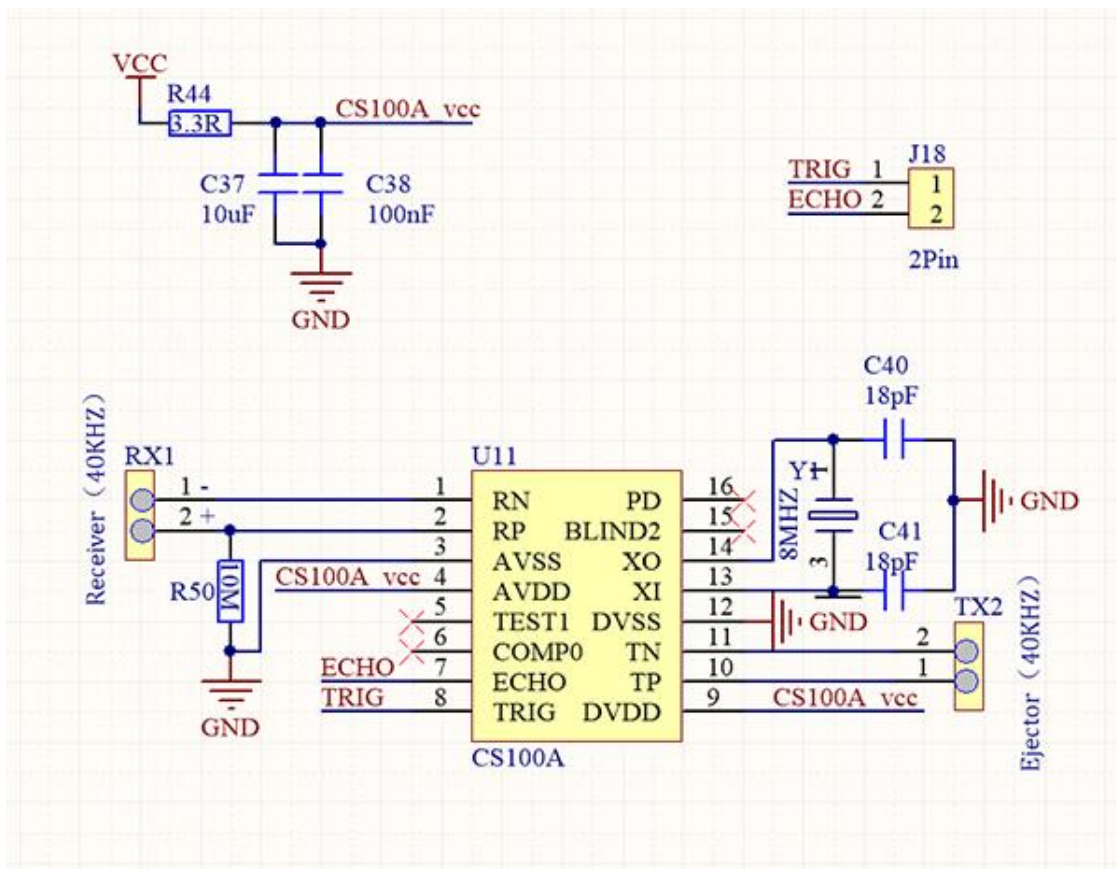
This ultrasonic rangefinder measures distance of obstacles by emitting sound waves and then receiving the echo. That is to say, the distance is not an immediate value, but an observed one by a theoretical calculation of time difference between emitter and receiver.

Ultrasonic is able to detect the shape of objects, set up automatic doors and estimate flow velocity and pressure.

What's more, it supports cooperative works with computers. As a result, the measured value can be transmitted to computers via Arduino board.

In daily life, it is widely used for motors, servos and LEDs as well as systems(automatic navigation, control and security monitoring systems).

## 2. Working Principle



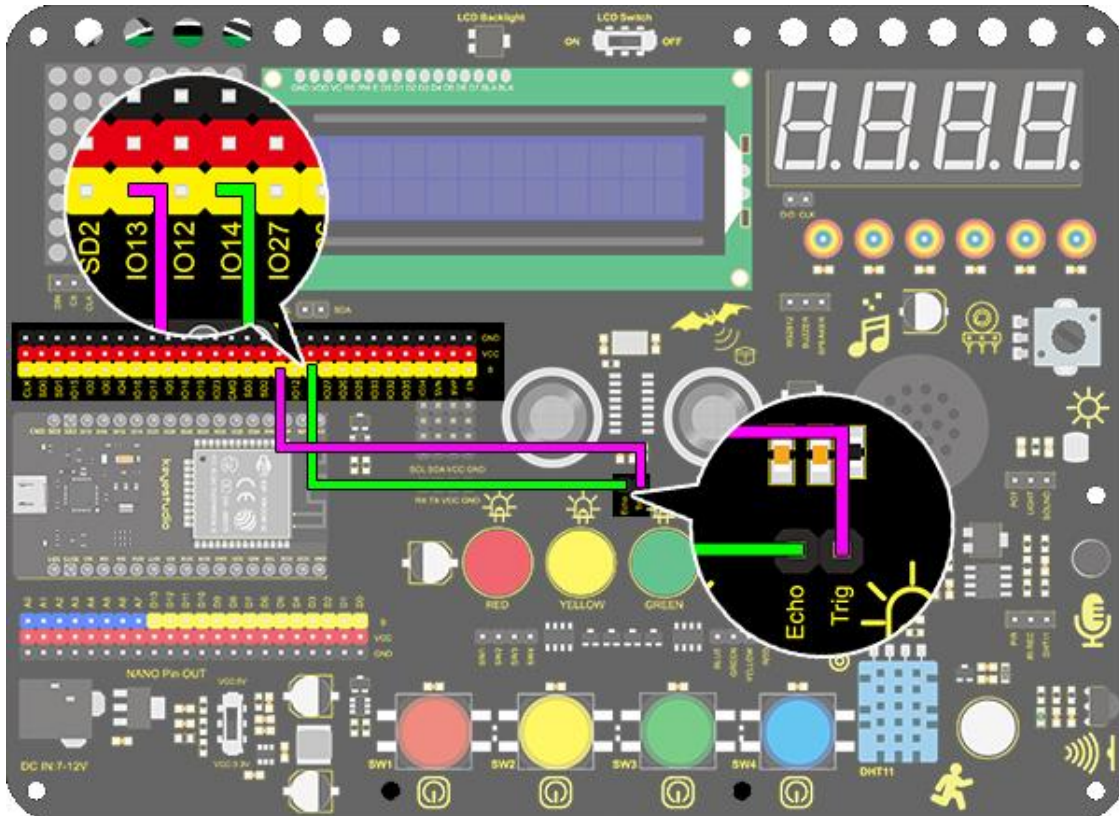
As we all know, ultrasonic is a kind of inaudible sound wave signal with high frequency. Similar to a bat, this module measures distance of obstacles by calculating the time difference between wave-emitting and echo-receiving.

**Maximum distance:** 3M

**Minimum distance:** 5cm

**Detection angle:**  $\leq 15^\circ$

### 3. Wiring Diagram



### 4. Upload Code

As its principle, we need to use a `pulseIn(pin, value)` function.

Use the Arduino IDE to open the .ino format code [project\\_25.1](#) directly from the tutorial package.

Connect the ESP32 board to the computer with the USB cable.

Select board type "**ESP32 Dev Module**" and select port **COM-XX** (This depends on the number your computer assigns to the ESP32 board, which you can check it in the device manager).



```

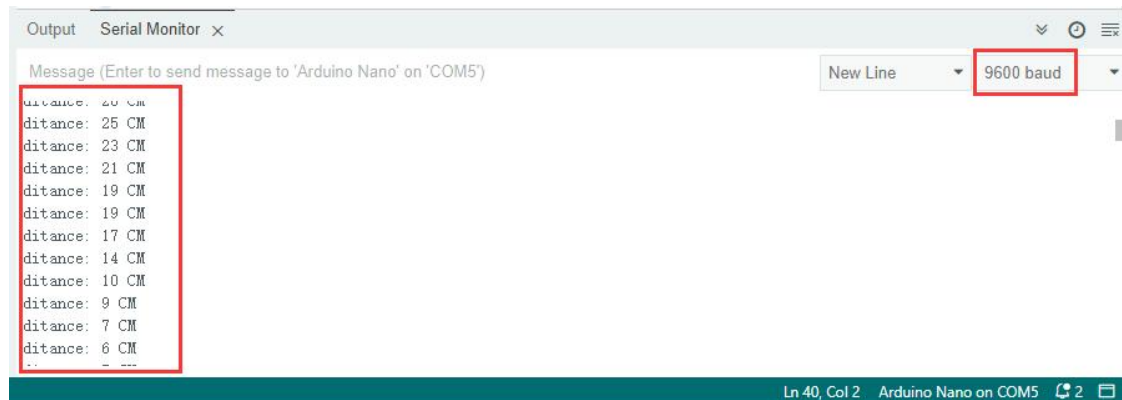
/*
  keyestudio ESP32 Inventor Learning Kit
  Project 25.1: Ultrasonic Rangefinder
  http://www.keyestudio.com
*/
int distance = 0; //Define a variable to receive the distance value
int EchoPin = 14; //Connect Echo pin to io14
int TrigPin = 13; //Connect Trig pin to io13

float checkdistance() { //Acquire the distance
  // preserve a short low level to ensure a clear high pulse:
  digitalWrite(TrigPin, LOW);
  delayMicroseconds(2); //Delay 2um
  //Trigger the sensor by a high pulse of 10um or longer
  digitalWrite(TrigPin, HIGH);
  delayMicroseconds(10); //Delay 10um
  digitalWrite(TrigPin, LOW);
  //Read the signal from the sensor: a high level pulse
  //Duration is detected from the point sending "ping" command to the time receiving echo signal (unit:
  float distance = pulseIn(EchoPin, HIGH) / 58.00; //Convert into distance
  delay(10);
  return distance; //Return the distance value
}

```

## 5. Test Result

After connecting the wiring and uploading code, open serial monitor to set baud rate to 9600, the serial port prints the distance value.





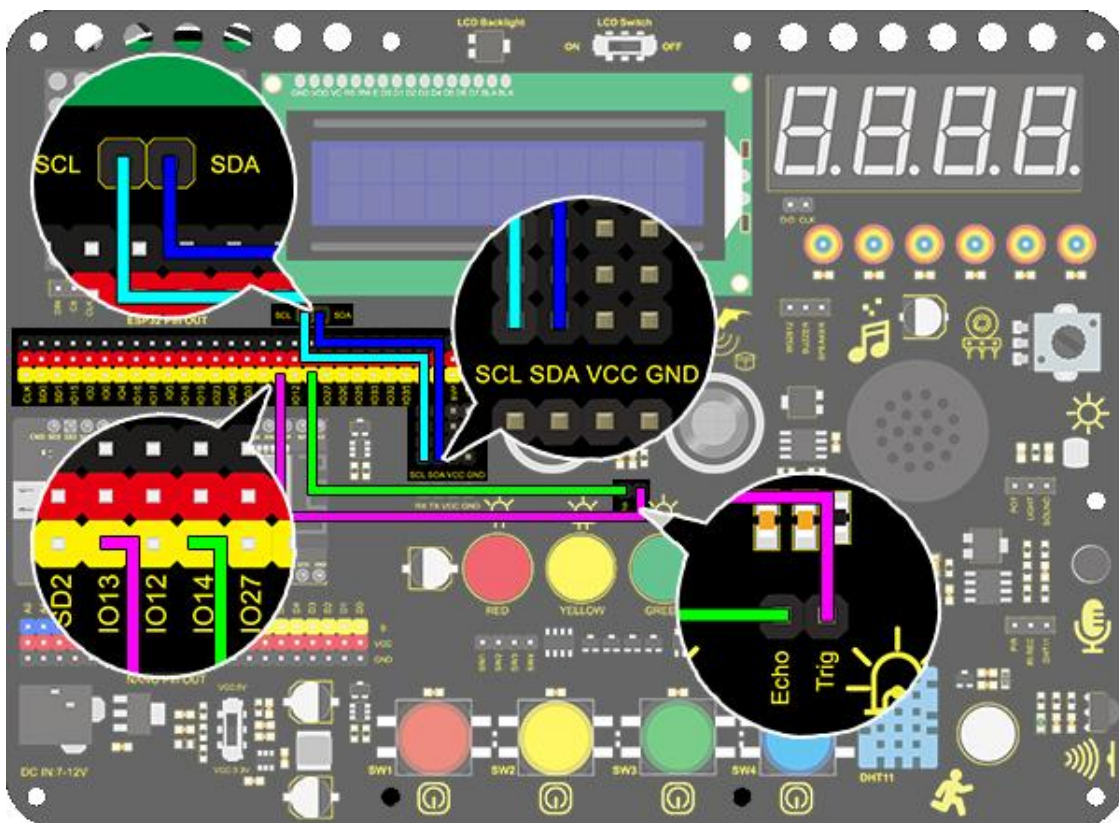
## 6. Knowledge Expansion

Let's make a rangefinder.

We display characters on LCD 1602. Program to show "Keyestudio" at (3,0) and "distance:" at (0,1) followed by the distance value at (9,1).

When the value is smaller than 100(or 10), a residue of the third(or the second) bit still exists. Therefore, an "if" judgement is necessary to determine a certain condition.

### Wiring Diagram:



**Code:**

Use the Arduino IDE to open the .ino format code [project\\_25.2](#) directly from the tutorial package.

Connect the ESP32 board to the computer with the USB cable.

Select board type "**ESP32 Dev Module**" and select port **COM-XX** (This depends on the number your computer assigns to the ESP32 board, which you can check it in the device manager).

```

/*
  keyestudio ESP32 Inventor Learning Kit
  Project 25.2: Ultrasonic Rangefinder
  http://www.keyestudio.com
*/
#include <Wire.h>
#include <LiquidCrystal_I2C.h>
LiquidCrystal_I2C lcd(0x27,16,2); //set the LCD address to 0x27 for a 16 chars and 2 line display

int distance = 0; //Define a variable to receive the diatance value
int EchoPin = 14; //Connect Echo pin to io14
int TrigPin = 13; //Connect Trig pin to io13
float checkdistance() { //Acquire the distance
  // preserve a short low level to ensure a clear high pulse:
  digitalWrite(TrigPin, LOW);
  delayMicroseconds(2);
  //Trigger the sensor by a high pulse of 10um or longer
  digitalWrite(TrigPin, HIGH);
  delayMicroseconds(10);
  digitalWrite(TrigPin, LOW);
  // Read the signal from the sensor: a high level pulse
  //Duration is detected from the point sending "ping" command to the time receiving echo signal (unit:
  float distance = pulseIn(EchoPin, HIGH) / 58.00; //Convert into distance
  delay(10);
  return distance;
}

```

## 7. Code Explanation

**float checkdistance()** :Self-defining function. It greatly reduces loop() by collecting some specific codes which can be directly recalled.

**delayMicroseconds();** Delay function. delay()is in ms while delayMicroseconds() is in um for some precise delays.

**pulseIn(pin, value)** :Pulse-grabbing function.

**pin:** the number of the Arduino pin on which you want to read the pulse. Allowed data types: int.

**value:** type of pulse to read: either HIGH or LOW. Allowed data types: int.

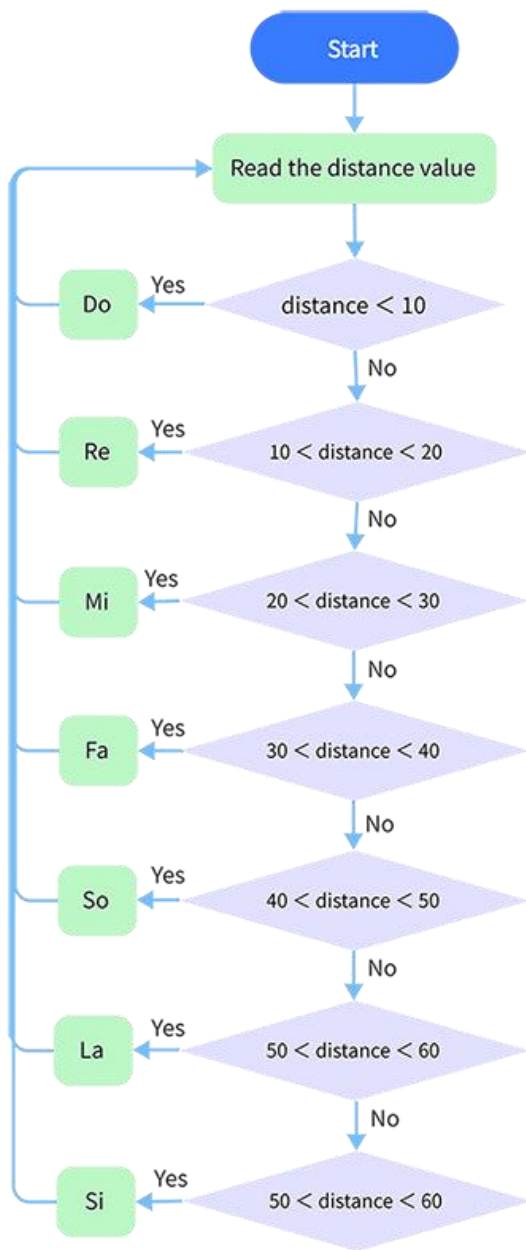
Please refer to the website for more details: [[pulseIn\(\) - Arduino Reference](#)]:

# Project 26: Human Body Piano

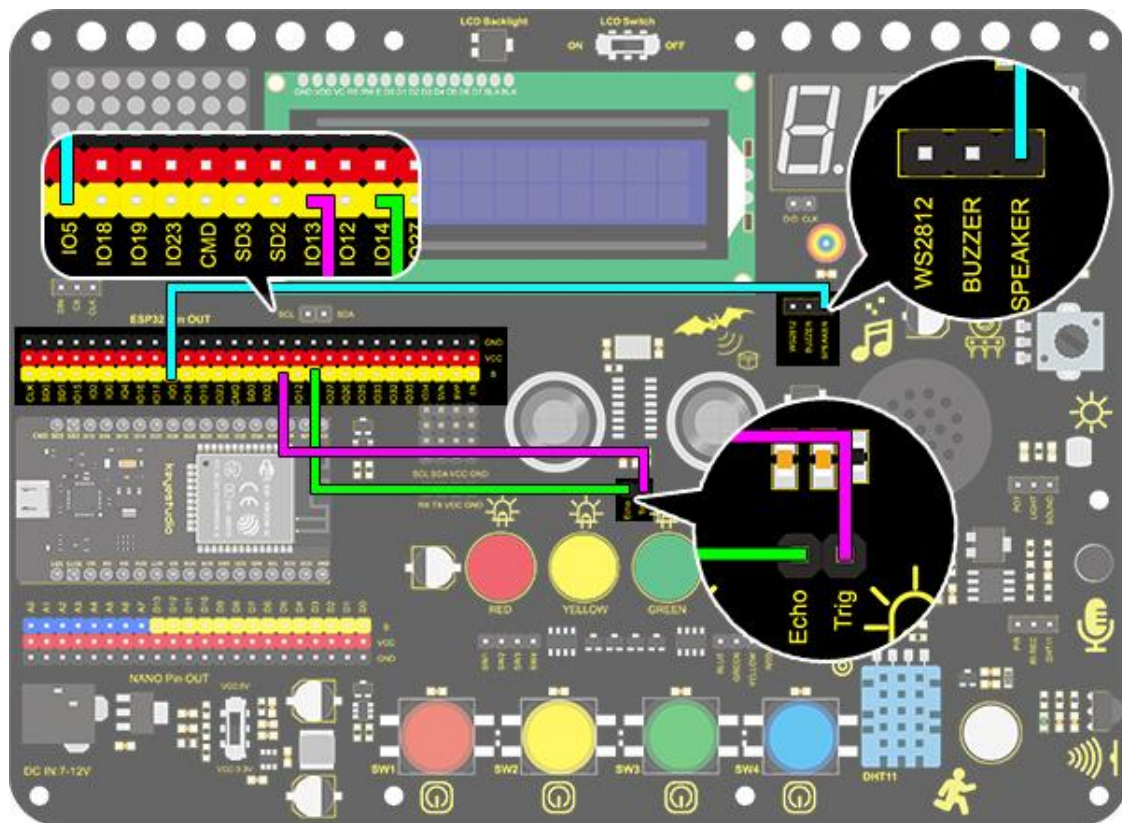
## 1. Description

The analog piano includes a development board and an ultrasonic sensor. It enables to play different tones by detecting the position of your fingers. Thus, this module is able to stimulate a piano to perform music and songs.

## 2. Flow Chart



### 3. Wiring Diagram



### 4. Upload Code

It is interesting that the played tones vary from distance of human's body.

Use the Arduino IDE to open the .ino format code [project\\_26](#) directly from the tutorial package.

Connect the ESP32 board to the computer with the USB cable.

Select board type "**ESP32 Dev Module**" and select port **COM-XX** (This depends on the number your computer assigns to the ESP32 board, which you can check it in the device manager).

```

/*
  keyestudio ESP32 Inventor Learning Kit
  Project 26 Human Body Piano
  http://www.keyestudio.com
*/
int distance = 0; //Define a variable to receive the distance
int EchoPin = 14; //Connect Echo pin to io14
int TrigPin = 13; //Connect Trig pin to io13

int beepPin = 5;

float checkdistance() { //Acquire distance
  // preserve a short low level to ensure a clear high pulse:
  digitalWrite(TrigPin, LOW);
  delayMicroseconds(2);
  // Trigger the sensor by a high pulse of 10um or longer
  digitalWrite(TrigPin, HIGH);
  delayMicroseconds(10);
  digitalWrite(TrigPin, LOW);
  // Read the signal from the sensor: a high level pulse
  //Duration is detected from the point sending "ping" command to the time receiving echo signal (unit:
  float distance = pulseIn(EchoPin, HIGH) / 58.00; //Convert into distance
  delay(10);
  return distance;
}

```

## 5. Test Result

Connect the wirings and upload the code.

Play Do when the distance is less than 10.

Play Re when the distance is within 10~20.

Play Mi when the distance is within 20~30.

Play Fa when the distance is within 30~40.

Play So when the distance is within 40~50.

Play La when the distance is within 50~60.

Play Si when the distance is within 60~70.

## 6. Code Explanation

|| : logical and operational signs. if (distance < 2 || distance >= 400): If one of the expressions satisfies the condition, it is true, otherwise it is false.

Please refer to official website for more details:[[|| - Arduino Reference](#)]:



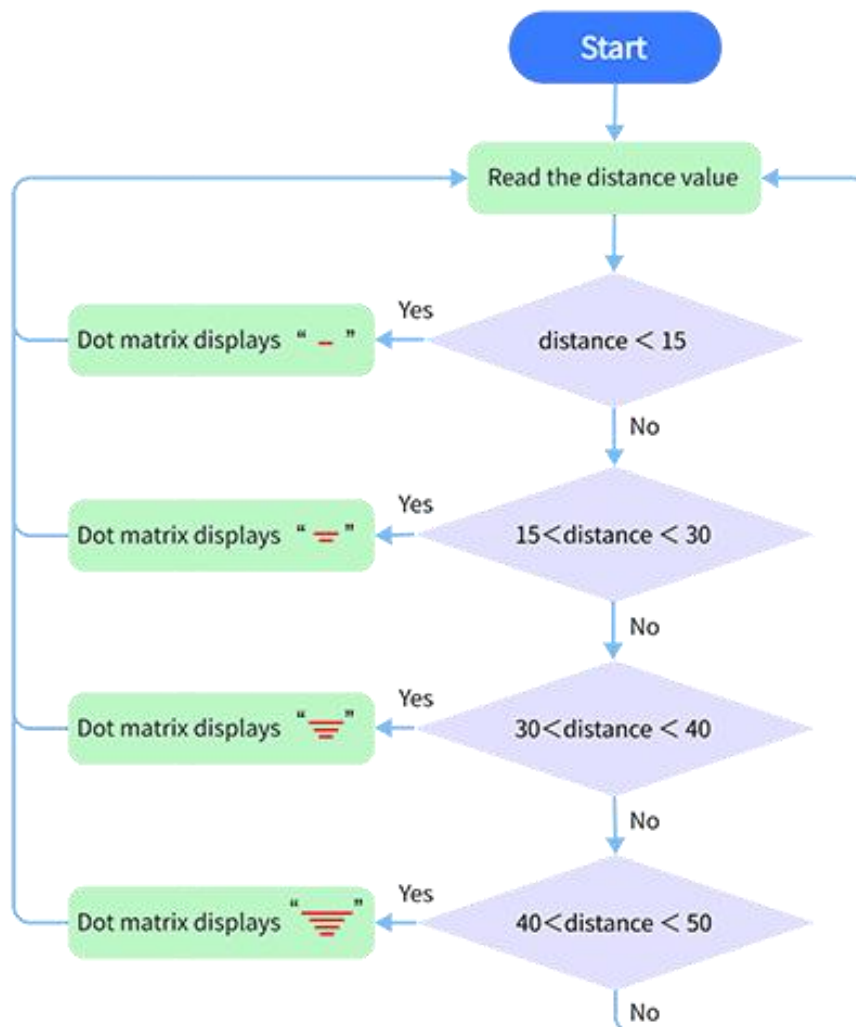
# Project 27: Intelligent Parking

## 1. Description

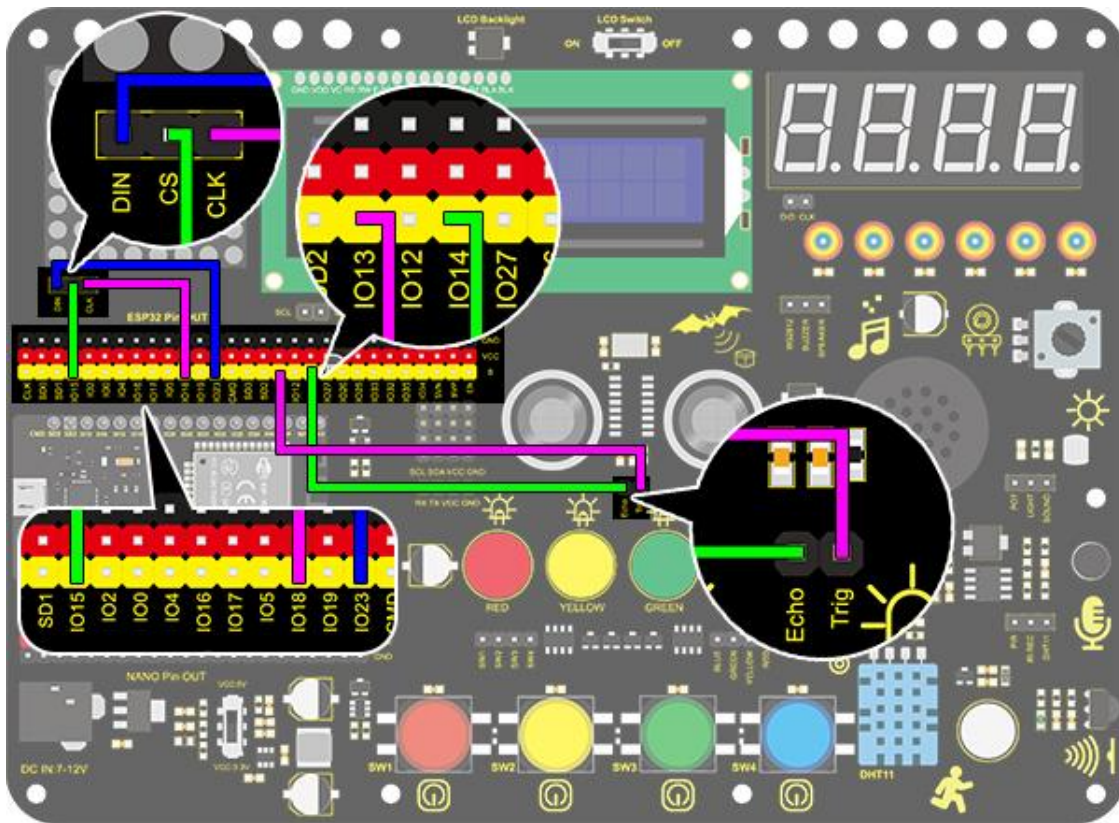
This intelligent parking system detects and optimizes parking position via an ultrasonic sensor. With this system, wrong parking is avoided to a large extent.

Firstly, you need to install the sensor around the carpark. And then it will detect the distance between the car and its edges and send the information to the development board so as to control the car to automatically adjust to the optimal parking position.

## 2. Flow Chart



### 3. Wiring Diagram



### 4. Upload Code

When parking, we can use the ultrasonic to know the situation of blind spots. In this project, lines displayed on the dot matrix indicates the distance of the car.

Use the Arduino IDE to open the .ino format code [project\\_27](#) directly from the tutorial package.

Connect the ESP32 board to the computer with the USB cable.

Select board type "**ESP32 Dev Module**" and select port **COM-XX** (This depends on the number your computer assigns to the ESP32 board, which you can check it in the device manager).

```

/*
  keyestudio ESP32 Inventor Learning Kit
  Project 27 Intelligent Parking
  http://www.keyestudio.com
*/
#include <LedControl.h>

int DIN = 23; //Define DIN pin to I023
int CS = 15;  //Define CS pin to I015
int CLK = 18; //Define CLK pin to I018

int temp = 0;

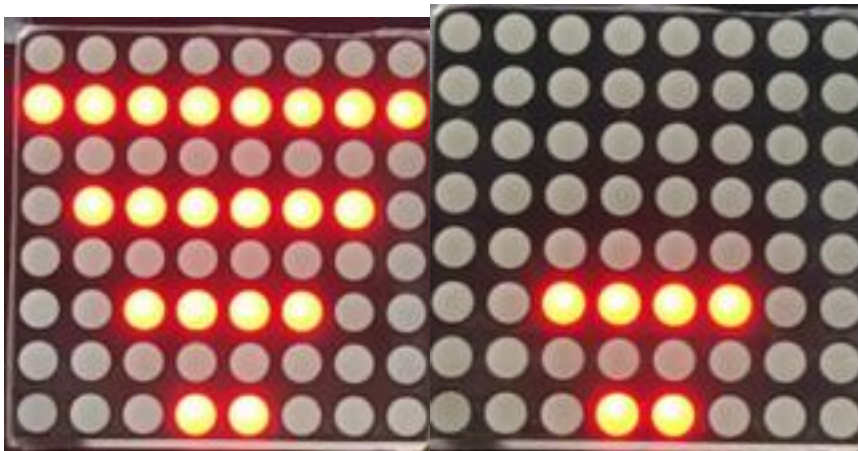
int distance = 0; //Define a variable to receive the distance
int EchoPin = 14; //Connect Echo pin to I014
int TrigPin = 13; //Connect Trig pin to I013

float checkdistance() { //Acquire distance
  // preserve a short low level to ensure a clear high pulse:
  digitalWrite(TrigPin, LOW);
  delayMicroseconds(2);
  // Trigger the sensor by a high pulse of 10um or longer
  digitalWrite(TrigPin, HIGH);
  delayMicroseconds(10);
  digitalWrite(TrigPin, LOW);
  // Read the signal from the sensor: a high level pulse
  //Duration is detected from the point sending "ping" command to the time receiving echo signal (unit:
  float distance = pulseIn(EchoPin, HIGH) / 58.00; //Convert into distance
}

```

## 5. Test Result

After connecting the wiring and uploading code, lines will be displayed on the dot matrix. If the detected distance is less than 50cm, there will be fewer lines.



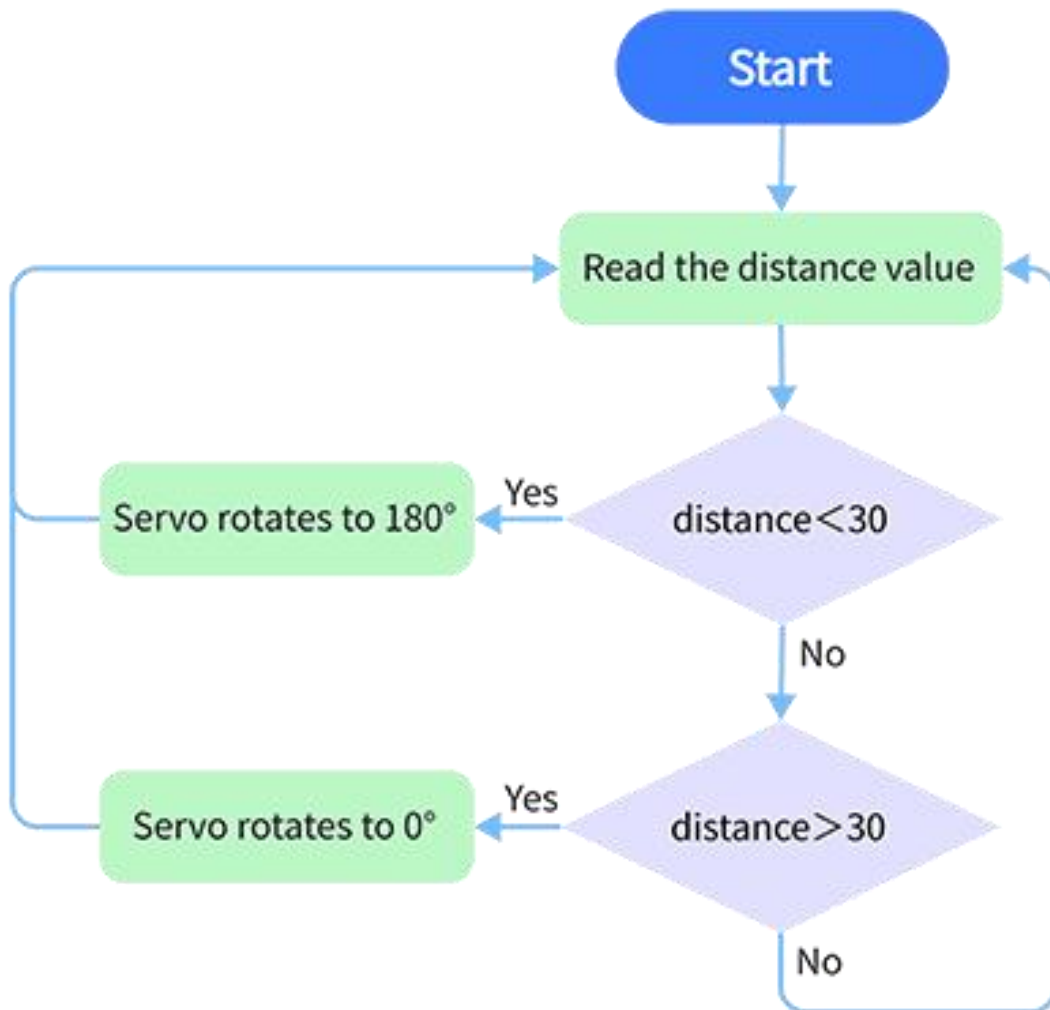
## Project 28: Intelligent Gate

### 1. Description

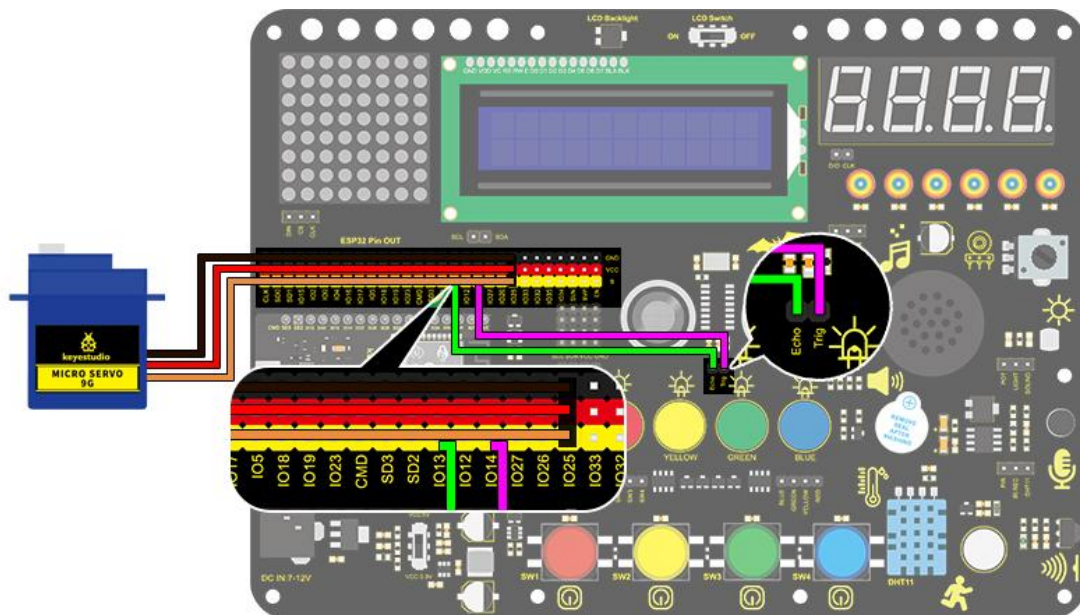
The intelligent gate is an intelligent parking lot system that integrates MCU and ultrasonic sensor, which automatically controls the gate according to the distance of cars, so as to better control the car access.

When a certain distance is reached, MCU receives the signal from the sensor and estimates the distance via the signal intensity. If the car is approaching or leaving, MCU will open or close the gate via a servo.

### 2. Flow Chart



### 3. Wiring Diagram



### 4. Upload Code

Use the Arduino IDE to open the .ino format code [project\\_28](#) directly from the tutorial package.

Connect the ESP32 board to the computer with the USB cable.

Select board type "**ESP32 Dev Module**" and select port **COM-XX** (This depends on the number your computer assigns to the ESP32 board, which you can check it in the device manager).

```
/*
  keyestudio ESP32 Inventor Learning Kit
  Project 28 Intelligent Gate
  http://www.keyestudio.com
*/
#define servo_pin 25
int distance = 0; //Define a variable to receive the distance
int EchoPin = 14; //Connect Echo pin to IO14
int TrigPin = 13; //Connect Trig pin to IO13

//Ultrasonic ranging program
float checkdistance() { //Acquire distance
  //preserve a short low level to ensure a clear high pulse:
  digitalWrite(TrigPin, LOW);
  delayMicroseconds(2);
```

### 5. Test Result

After connecting the wiring and uploading code, the servo will rotate to 180° for 5s if the detected distance is less than 30cm. On the contrary, the servo will rotate to 0°.

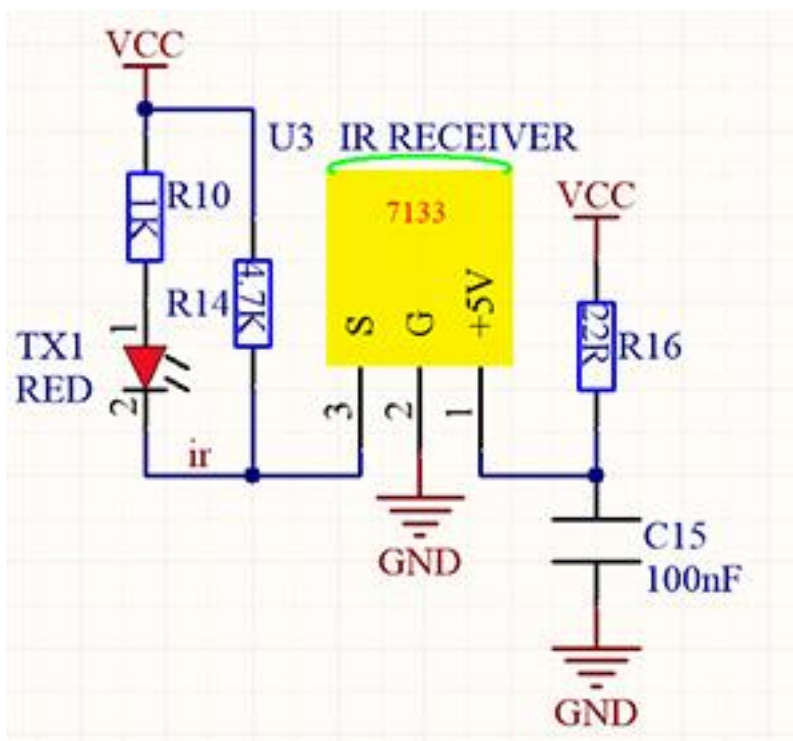


## Project 29: IR Remote Data

### 1. Description

The IR remote control uses IR signal to control LED, which greatly simplifies the process of controlling LED.

### 2. Working Principle



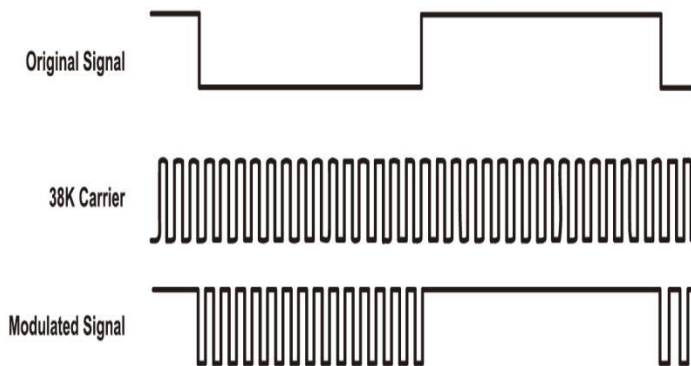
In this project, we often use a carrier of about 38K for modulation.

IR remote control system includes modulation, emitting and receiving. It sends data through modulating, which improves the transmission efficiency and reduces the power consumption.

Generally, the frequency of carrier modulation is within 30kHz~60kHz(usually 38kHz). The duty cycle of the square wave is 1/3, as shown below, which is decided by the 455kHz crystal oscillator on the emitting end.

An Integer frequency division is essential for crystal oscillator at this end, and the frequency coefficient usually evaluates 12. Therefore,  $455\text{kHz} \div 12 \approx 37.9\text{kHz} \approx 38\text{kHz}$ .

### 38KH carrier (complete) emitting diagram:



**Carrier frequency:** 38KHz

**Wave length:** 940nm

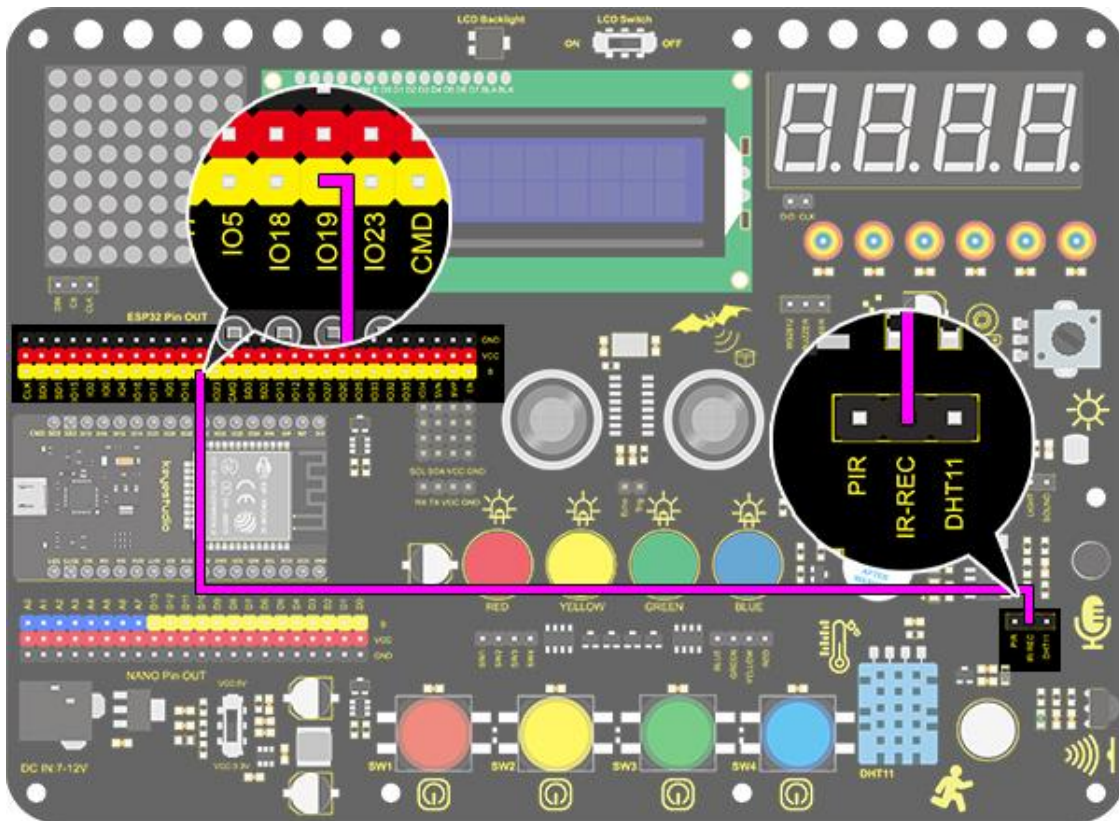
**Receiving angle:** 90°

**Control distance:** 6M

### Schematic diagram of remote control buttons:



### 3. Wiring Diagram



### 4. Upload Code

Use the Arduino IDE to open the .ino format code [project\\_29.1](#) directly from the tutorial package.

Connect the ESP32 board to the computer with the USB cable.

Select board type "**ESP32 Dev Module**" and select port **COM-XX** (This depends on the number your computer assigns to the ESP32 board, which you can check it in the device manager).

```

/*
  keyestudio ESP32 Inventor Learning Kit
  Project 29.1 IR Remote Control
  http://www.keyestudio.com
*/
#include <Arduino.h>
#include <IRremoteESP8266.h>
#include <IRrecv.h>
#include <IRutils.h>

const uint16_t recvpin = 19; // Infrared receiving pin
IRrecv irrecv(recvpin); // Create a class object used to receive class
decode_results results; // Create a decoding results class object
long ir_rec;

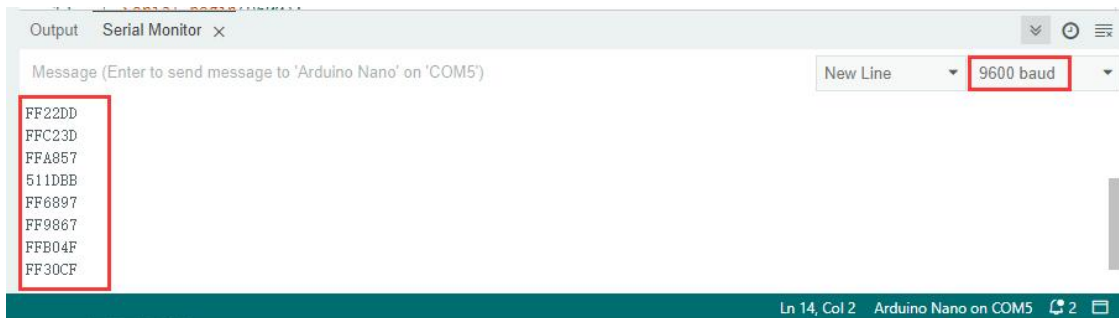
void setup()
{
  Serial.begin(9600); // Initialize the serial port and set the baud rate to 9600
  irrecv.enableIRIn(); // start receiving signals
}

void loop() {
  if (irrecv.decode(&results)) {
    ir_rec = results.value; //assign the signal to the variable ir_rec
    if(ir_rec != 0){          //Prevent the code from repeating execute when the button is pressed

```

## 5. Test Result

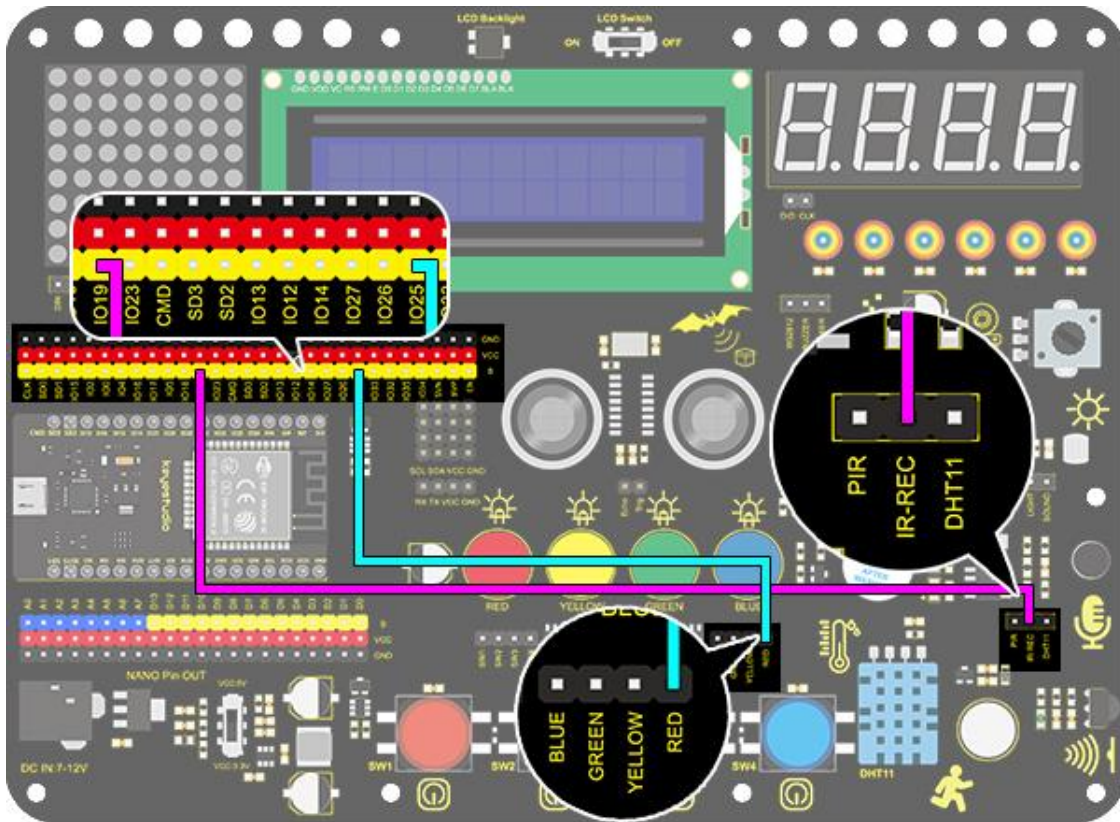
After connecting the wiring and uploading code, open the serial monitor and set the baud rate to 9600. Press the button on the remote control, and you will see the value in hexadecimal.



## 6. Knowledge Expansion

Next, we will use an IR remote control to control the LED. Press OK to light up the LED and press again to turn it off.

### Wiring Diagram:



### Code:

Use the Arduino IDE to open the .ino format code [project\\_29.2](#) directly from the tutorial package.

Connect the ESP32 board to the computer with the USB cable.

Select board type "[ESP32 Dev Module](#)" and select port [COM-XX](#) (This depends on the number your computer assigns to the ESP32 board, which you can check it in the device manager).



```

/*
  keyestudio ESP32 Inventor Learning Kit
  Project 29.2 IR Remote Control
  http://www.keyestudio.com
*/
#include <Arduino.h>
#include <IRremoteESP8266.h>
#include <IRrecv.h>
#include <IRutils.h>

int led = 25;
int led_val = 0;
const uint16_t recvpin = 19; // Infrared receiving pin
IRrecv irrecv(recvpin);      // Create a class object used to receive class
decode_results results;      // Create a decoding results class object
long ir_rec;

void setup() {
  Serial.begin(9600); // Initialize the serial port and set the baud rate to 9600
  irrecv.enableIRIn(); // start receiving signals
  pinMode(led, OUTPUT);
}

void loop() {
  if (irrecv.decode(&results)) {

```

### Test Result:

Press OK to light up the LED and press again to turn it off.

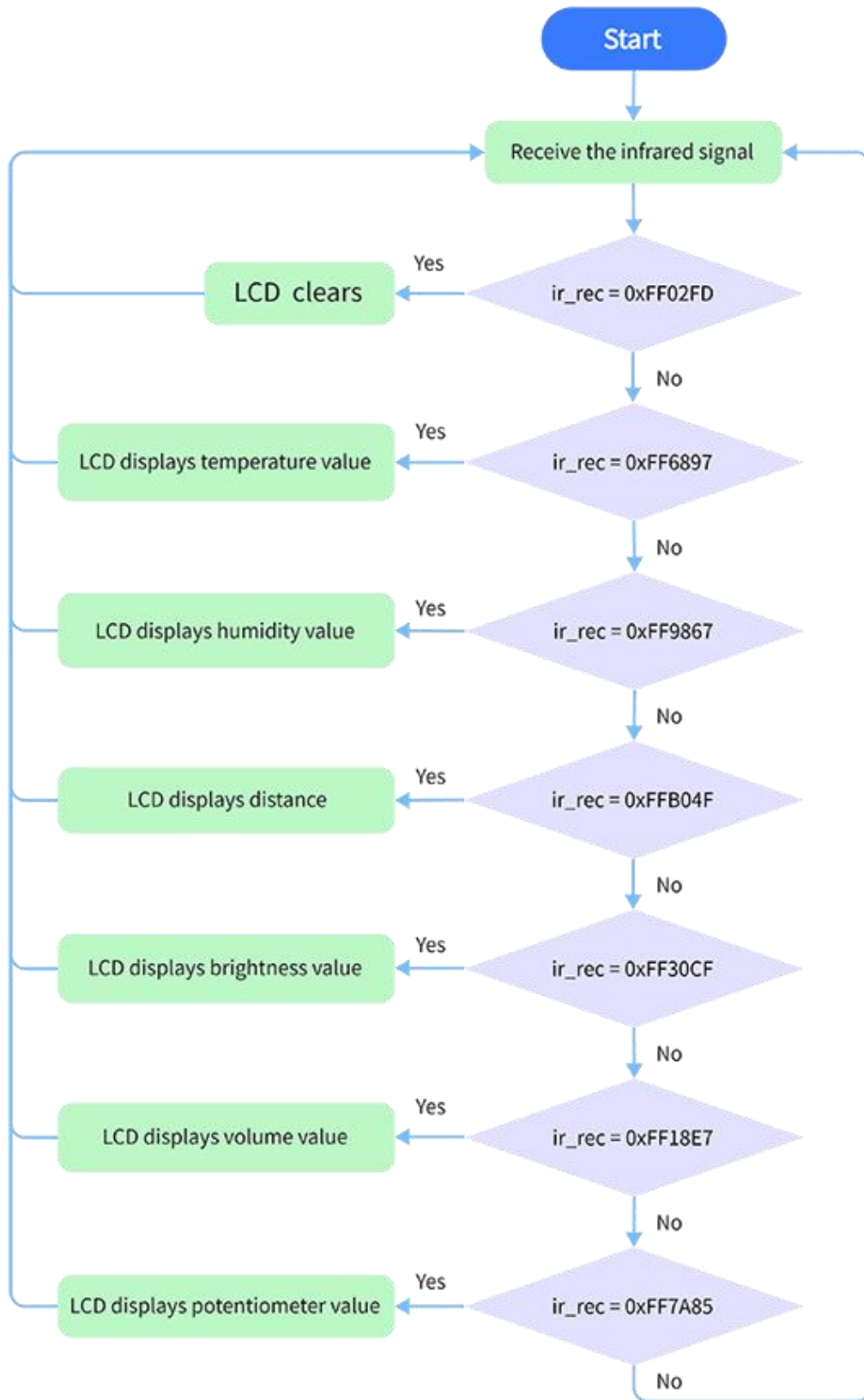
## Project 30: IR Remote Control

### 1. Description

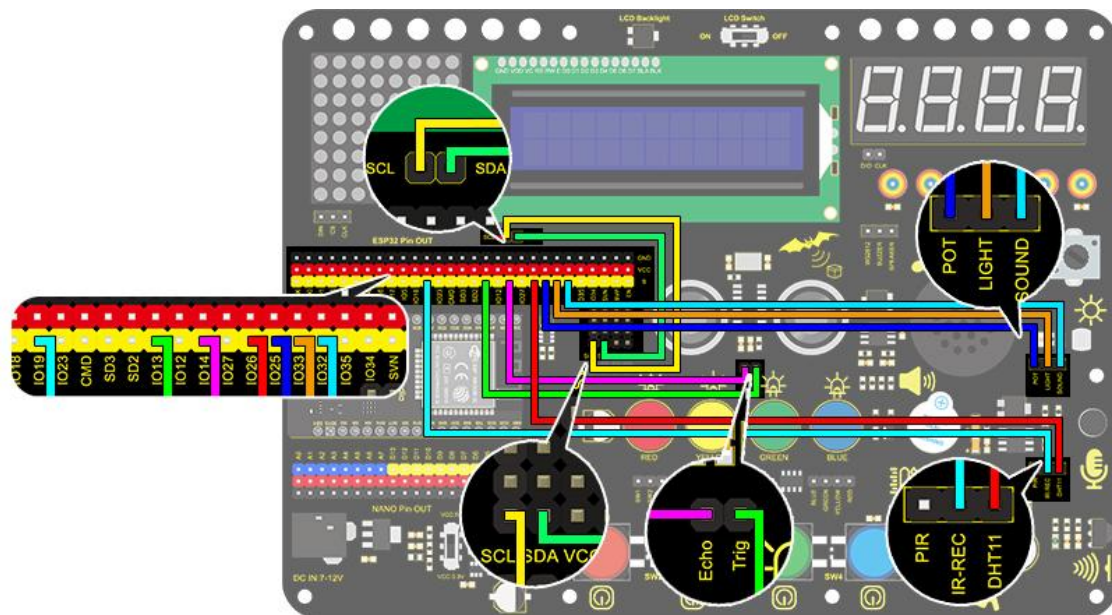
In this technology era, we are all familiar with smart home. It is a system that can control electric appliance via buttons.

In this project, we seek to stimulate a smart home via an IR remote control. With Arduino MCU as its core, it can be used to control light, air conditioners, TV and security monitors.

## 2. Flow Chart



### 3. Wiring Diagram



### 4. Upload Code

With the IR remote control, this smart home reveals various sensor values on LCD, including a temperature and humidity sensor, a sound sensor, a photoresistor, a potentiometer and an ultrasonic sensor.

Use the Arduino IDE to open the .ino format code [project\\_30](#) directly from the tutorial package.

Connect the ESP32 board to the computer with the USB cable.

Select board type "**ESP32 Dev Module**" and select port **COM-XX** (This depends on the number your computer assigns to the ESP32 board, which you can check it in the device manager).

```

/*
  keyestudio ESP32 Inventor Learning Kit
  Project 30 IR Remote Control
  http://www.keyestudio.com
*/
#include <LiquidCrystal_I2C.h>
#include <IRremoteESP8266.h>
#include <IRrecv.h>
#include <IRutils.h>
#include <xht11.h>

LiquidCrystal_I2C lcd(0x27, 16, 2); // set the LCD address to 0x27 for a 16 chars and 2 line display

const uint16_t recvPin = 19; // Infrared receiving pin
IRrecv irrecv(recvPin); // Create a class object used to receive class
decode_results results; // Create a decoding results class object
long ir_rec;

xht11 xht(26); //The DHT11 connects to IO26
unsigned char dat[] = { 0, 0, 0, 0 }; //Define an array to store temperature and humidity data

int distance = 0; //Define a variable to receive the distance
int EchoPin = 14; //Connect Echo pin to IO14
int TrigPin = 13; //Connect Trig pin to IO13

int lighth_sensor = 33; //Define the photoresistor pin
int sound_sensor = 32; //efine the sound sensor pin
int pot_sensor = 25; //Define the potentiometer pin

```

## 5. Test Result

After connecting the wiring and uploading code, we can see the corresponding contents on LCD by pressing buttons. OK button clears the sensor display.



# Project 31: Connect the ESP32 board to WiFi

## 1. Description

ESP32 boasts a built-in Wi-Fi and Bluetooth module that is widely used in Internet of Things (IoT). With this function, it can remotely control the data transmission through the wireless network.

In applications, ESP32 can be used as a client to connect to a Wi-Fi network, or as a hotspot to create its own network. Through these connections, ESP32 receives commands to control external devices, such as turning on/off lights and adjusting temperature. In the code, protocols like HTTP and MQTT are used to communicate with the server to achieve data sending and receiving, so as to remotely control and monitoring.

## 2. ESP32 wifi

ESP32 development board comes with built-in Wi-Fi (2.4G) and Bluetooth (4.2), which enable it to easily connect to Wi-Fi network and communicate with other devices in the network. You can display web pages in your browser via ESP32.

- Base station mode (STA / Wi-Fi Client mode): ESP32 is connected to Wi-Fi hotspot (AP).
- AP mode (Soft-AP / Wi-Fi hotspot mode): Wi-Fi device(s) is(are) connected to ESP32.
- AP-STA mode: ESP32 is both Wi-Fi hotspot and a Wi-Fi device connected to another Wi-Fi.
- These modes supports multiple security modes, including WPA, WPA2 and WEP.
- It is able to scan Wi-Fi hotspot (active or passive)
- It support promiscuous mode monitoring IEEE802.11 Wi-Fi packets.

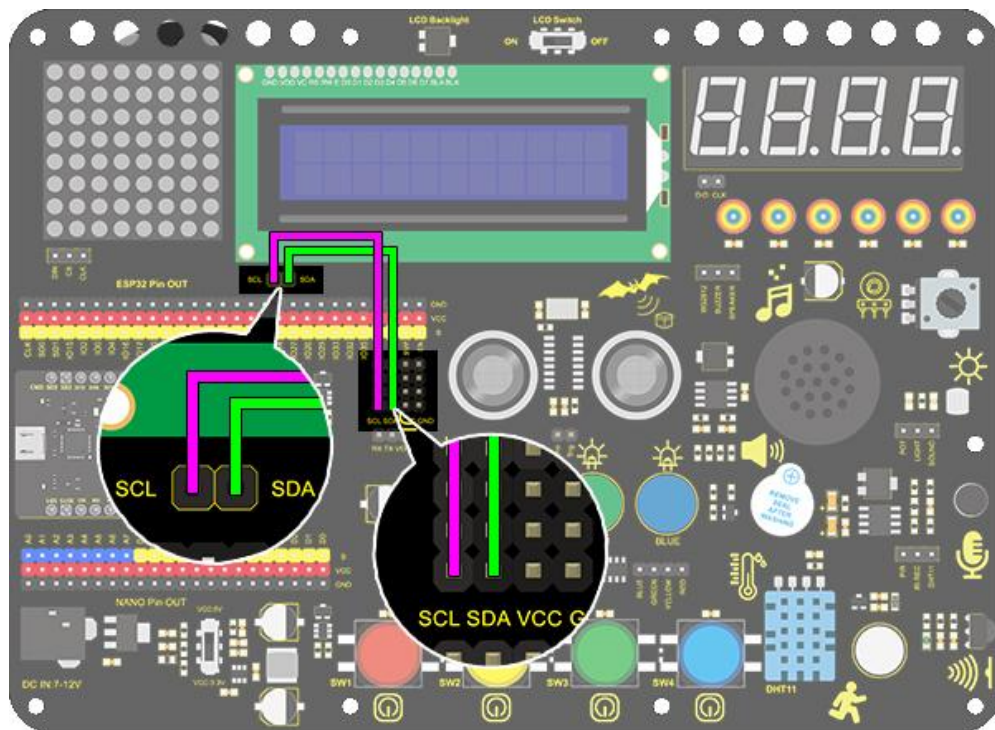
---

For more wifi reference, please visit: [https://docs.espressif.com/projects/esp-idf/en/latest/esp32/api-reference/network/esp\\_wifi.html](https://docs.espressif.com/projects/esp-idf/en/latest/esp32/api-reference/network/esp_wifi.html)

espressif official: <https://www.espressif.com.cn/en/home>



### 3. Wiring Diagram



#### Notes:

1. You need to prepare a 2.4GHz frequency WIFI(not 5GHz). It can be a mobile hotspot or a router.
2. The ESP32 board consumes more power when connected to the network, so you need to connect an external power supply to this kit. We provide you with a 6XAA Battery Holder (battery not included), which you can connect to the DC port of the ESP32 integrated board.



3. Remember your wifi network name and password and fill it into the code before uploading it.

```
const char* ssid = "your_SSID"; // Fill in WiFi name, for example,= "KEYES"  
const char* password = "your_password"; // Fill in WiFi password, for example,=  
"123456"
```

## 4. Upload Code

Use the Arduino IDE to open the .ino format code [project\\_31.1](#) directly from the tutorial package.

Connect the ESP32 board to the computer with the USB cable.

Select board type "**ESP32 Dev Module**" and select port **COM-XX** (This depends on the number your computer assigns to the ESP32 board, which you can check it in the device manager).

```
/*
  keyestudio ESP32 Inventor Learning Kit
  Project 31 ESP32 WiFi
  http://www.keyestudio.com
*/
#include <WiFi.h>
#include <LiquidCrystal_I2C.h>

LiquidCrystal_I2C lcd(0x27, 16, 2);

const char* ssid = "your_SSID"; // set to your WiFi name
const char* password = "your_password"; // set your WiFi password

WiFiServer server(80);

int i = 0;

void setup() {
  lcd.init(); // initialize the lcd
  // We start by connecting to a WiFi network
  lcd.backlight();

  lcd.setCursor(0, 0);
```

## 5. Test Result

After uploading the code, LCD1602 shows the IP address of the wifi that you connected to ESP32.



---

## 6. Knowledge Expansion

IP address displays “Holly World!”.

### Upload Code:

Use the Arduino IDE to open the .ino format code [project\\_31.2](#) directly from the tutorial package.

Connect the ESP32 board to the computer with the USB cable.

Select board type "**ESP32 Dev Module**" and select port **COM-XX** (This depends on the number your computer assigns to the ESP32 board, which you can check it in the device manager).

```
project_31_2.ino
1  #include <WiFi.h>
2  #include <ESPAsyncWebServer.h>
3  #include <LiquidCrystal_I2C.h>
4  LiquidCrystal_I2C lcd(0x27, 16, 2);
5
6  // WiFi configuration
7
8  const char* ssid = "your-SSID"; // your WiFi name
9  const char* password = "your-PASSWORD"; // your WiFi password
10 int i = 0;
11 // Create a Web Server
12 AsyncWebServer server(80);
13
14 void setup() {
15     lcd.init(); // initialize the lcd
16     lcd.backlight();
17     lcd.setCursor(0, 0);
18     lcd.print("IP:");
```

### Test result:

Use a computer or mobile phone that is connected to the same network as the ESP32 board, and access the IP address shown on the LCD1602 and you will see “Hello world”.

19:23

243 KB/s 5G HD 5G HD 100



192.168.0.74



# Hello, World!

## Project 32: ESP32 WiFi Control LED

### 1. Description

Next we will learn to control the LED through wifi on a mobile phone or a computer.

#### Notes:

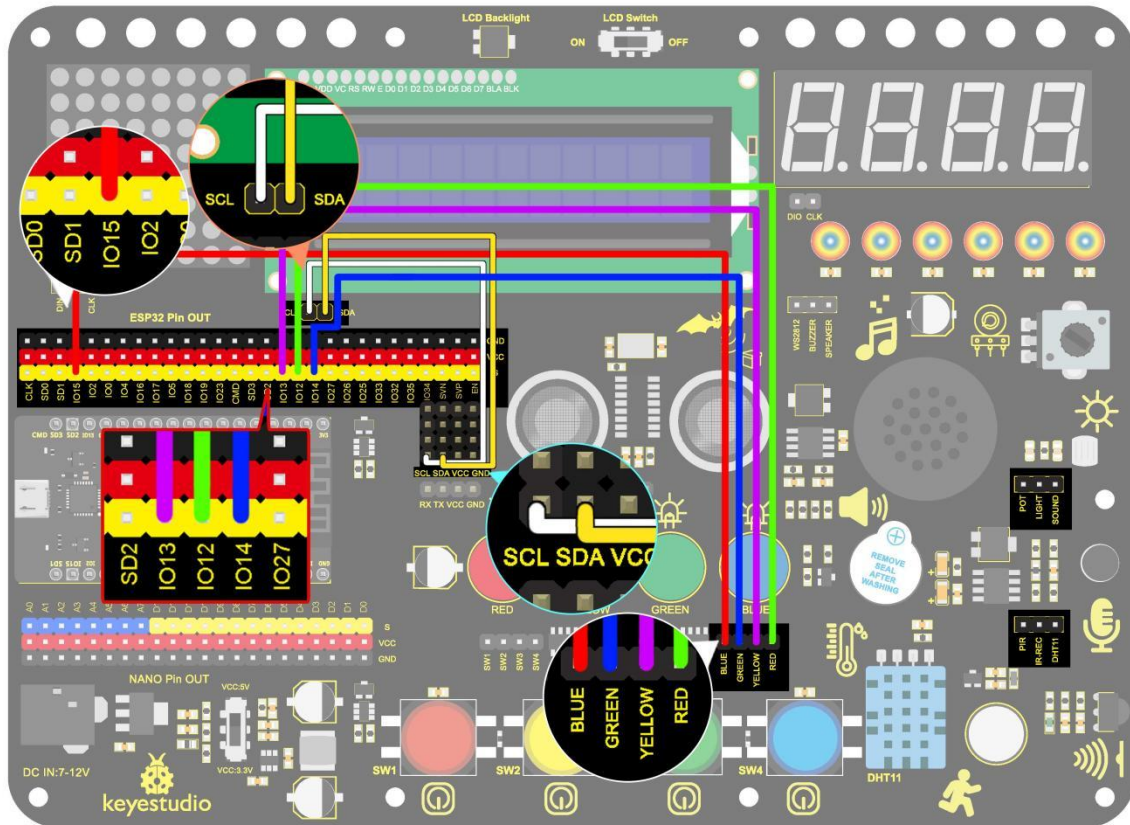
1. You need to prepare your a 2.4GHz frequency WIFI, not 5GHz frequency. It can be a mobile hotspot or a router.
2. The ESP32 board consumes more power when connected to the network, so you need to connect an external power supply to this kit. We provide you with a 6XAA Battery Holder (battery not included), which you can connect to the DC port of the ESP32 integrated board.



3. When using other devices to control this kit, the ESP32 board needs to be connected to the same network as your control device.
4. Remember your wifi network name and password and fill it into the code before uploading it.

```
const char* ssid = "your_SSID"; // Fill in WiFi name, for example,= "KEYES"  
const char* password = "your_password"; // Fill in WiFi password, for example,=  
"123456"
```

## 2. Wiring Diagram



## 3. Upload Code

Use the Arduino IDE to open the .ino format code [project\\_32](#) directly from the tutorial package.

Connect the ESP32 board to the computer with the USB cable.

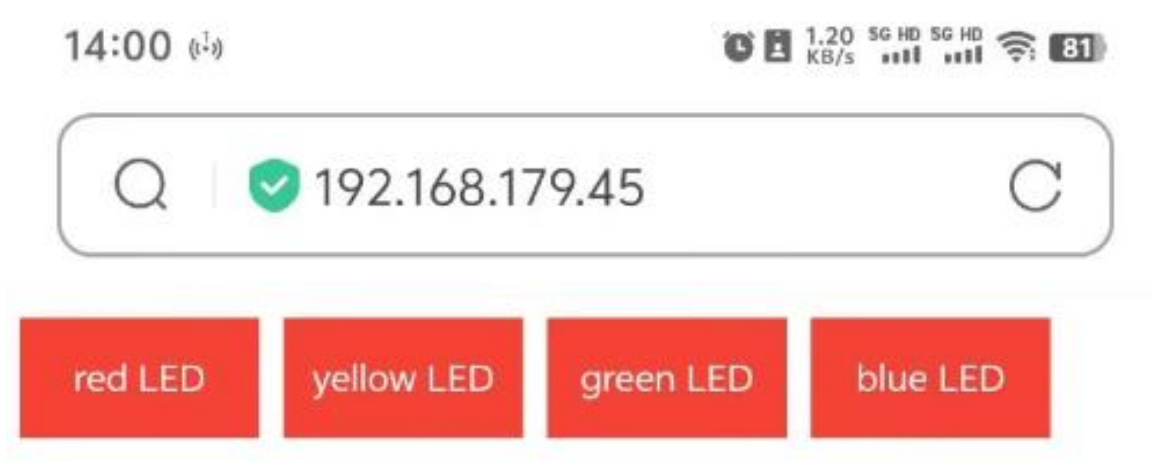
Select board type "**ESP32 Dev Module**" and select port **COM-XX** (This depends on the number your computer assigns to the ESP32 board, which you can check it in the device manager).



```
project_32.ino  debug_custom.json  ...
1  #include <WiFi.h>
2  #include <ESPAsyncWebServer.h>
3  #include <LiquidCrystal_I2C.h>
4
5  LiquidCrystal_I2C lcd(0x27, 16, 2);
6
7  // WiFi configuration
8  const char* ssid = "your-SSID"; // your WiFi name
9  const char* password = "your-PASSWORD"; // your WiFi password
10
11 // Create a Web Server
12 AsyncWebServer server(80);
13
14 // LED pin configuration
15 #define redLED 12
16 #define yellowLED 13
17 #define greenLED 14
18 #define blueLED 15
19
20 // LED state
21 bool redLEDState = false;
22 bool yellowLEDState = false;
23 bool greenLEDState = false;
24 bool blueLEDState = false;
25 int i = 0;
```

#### 4. Test Result

After uploading the code, LCD1602 shows the IP address of the wifi. Use a computer or mobile phone that is connected to the same network as the ESP32 board, open the browser and enter the IP address, you will see the control page.



# Project 33: ESP32 Reads Data

## 1. Description

We learned how to control the led light through ESP32 wifi and display the IP address on the LCD1602. Next, we will use the esp32 board read the sensor data and transmit it to the web page.

### Notes:

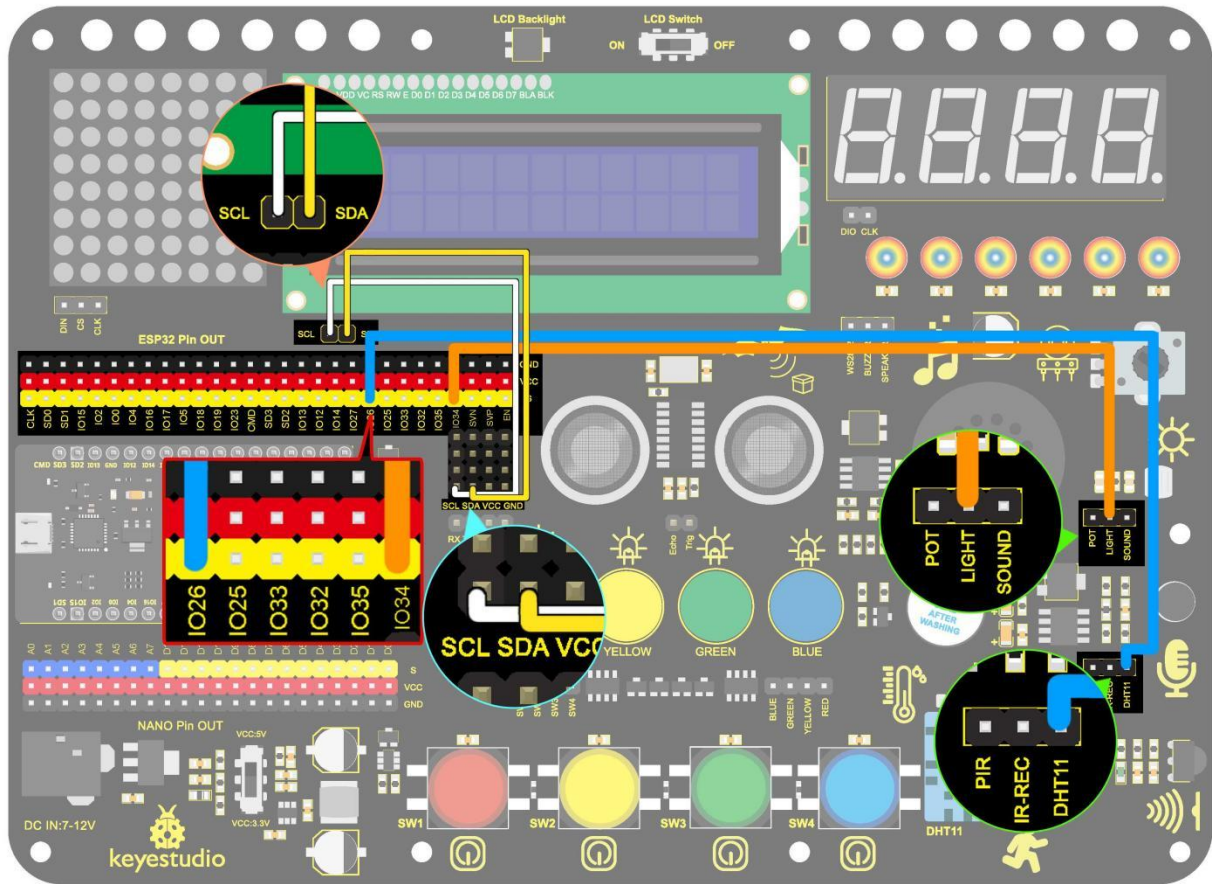
1. You need to prepare your a 2.4GHz frequency WIFI, not 5GHz frequency. It can be a mobile hotspot or a router.
2. The ESP32 board consumes more power when connected to the network, so you need to connect an external power supply to this kit. We provide you with a 6XAA Battery Holder (battery not included), which you can connect to the DC port of the the ESP32 integrated board.



3. When using other devices to control this kit, the ESP32 board needs to be connected to the same network as your control device.
4. Remember your wifi network name and password and fill it into the code before uploading it.

```
const char* ssid = "your_SSID"; // Fill in WiFi name, for example,= "KEYES"  
const char* password = "your_password"; // Fill in WiFi password, for example,=  
"123456"
```

## 2. Wiring Diagram



### 3. Upload Code

Use the Arduino IDE to open the .ino format code [project\\_33](#) directly from the tutorial package.

Connect the ESP32 board to the computer with the USB cable.

Select board type "**ESP32 Dev Module**" and select port **COM-XX** (This depends on the number your computer assigns to the ESP32 board, which you can check it in the device manager).

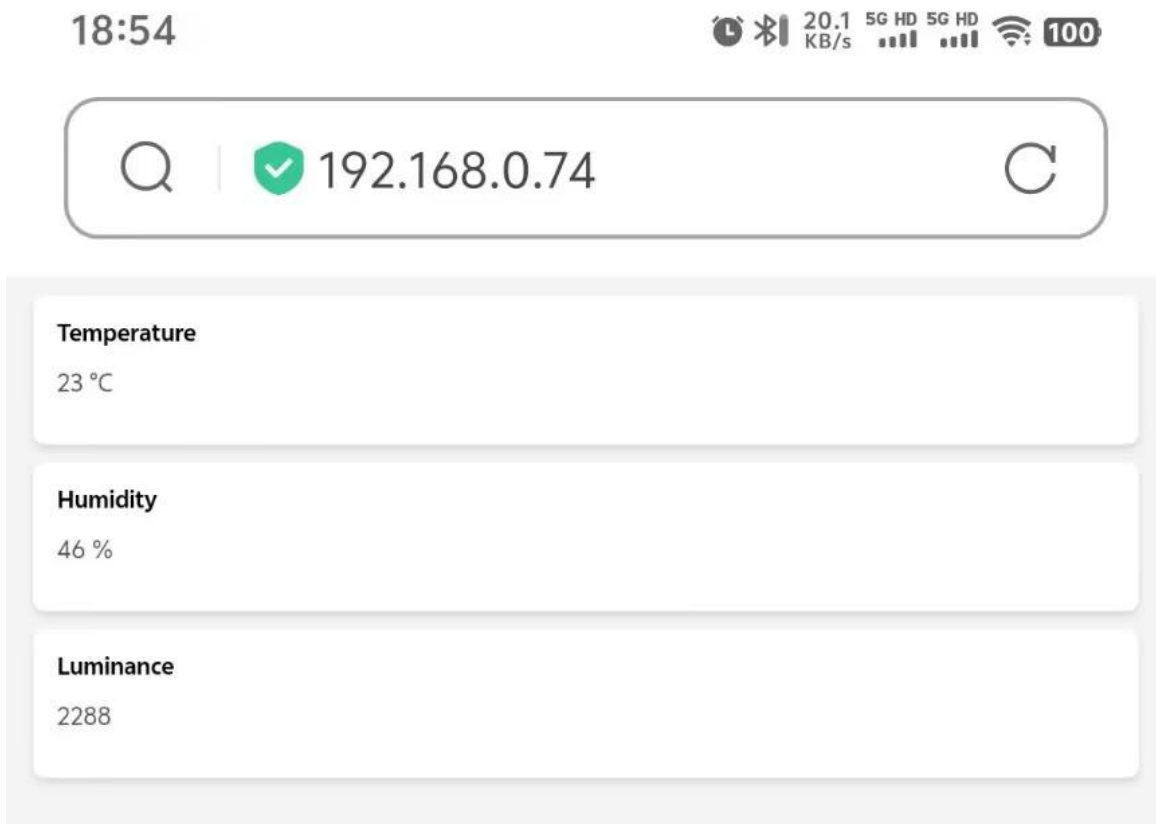
```

project_33.ino  debug_custom.json
1  #include <WiFi.h>
2  #include <ESPAsyncWebServer.h>
3  #include <xht11.h>
4  #include <LiquidCrystal_I2C.h>
5  LiquidCrystal_I2C lcd(0x27, 16, 2);
6
7  // WiFi configuration
8  const char* ssid = "your-SSID"; // your WiFi name
9  const char* password = "your-PASSWORD"; // your WiFi password
10
11 // Create a Web Server
12 AsyncWebServer server(80);
13
14 // DHT11 configuration
15 xht11 xht(26); //set DHT11 sensor pin to IO26
16 unsigned char dat[] = { 0, 0, 0, 0 }; //Define an array to store temperature and humidity values
17 int i = 0;

```

#### 4. Test Result

After uploading the code, LCD1602 shows the IP address. Use a computer or mobile phone that is connected to the same network as the ESP32 board, open the browser and enter the IP address, you can see the sensor values on the control page which refresh every 5 seconds.



# Project 34: Smart Home

## 1. Description

In this project, we simulate the smart home with the inventor kit.

### Notes:

1. You need to prepare your a 2.4GHz frequency WIFI, not 5GHz frequency. It can be a mobile hotspot or a router.
2. The ESP32 board consumes more power when connected to the network, so you need to connect an external power supply to this kit. We provide you with a 6XAA Battery Holder (battery not included), which you can connect to the DC port of the ESP32 integrated board.



3. When using other devices to control this kit, the ESP32 board needs to be connected to the same network as your control device.
4. Remember your wifi network name and password and fill it into the code before uploading it.

```
const char* ssid = "your_SSID"; // Fill in WiFi name, for example,= "KEYES"  
const char* password = "your_password"; // Fill in WiFi password, for example,=  
"123456"
```

## 2. Wiring Diagram





```
project_34.ino
1  #include <WiFi.h>
2  #include <ESPAsyncWebServer.h>
3  #include <xht11.h>
4  #include <LiquidCrystal_I2C.h>
5
6  LiquidCrystal_I2C lcd(0x27, 16, 2);
7
8  // WiFi configuration
9  const char* ssid = "your-SSID"; // your WiFi name
10 const char* password = "your-PASSWORD"; // your WiFi password
11
12 // DHT11 configuration
13 xht11 xht(26); //set DHT11 sensor pin to I026
14 unsigned char dat[] = { 0, 0, 0, 0 }; //Define an array to store temperature and humidity values
15 int i = 0;
16
17 // photoresistor analog pin
18 #define LDR_PIN 34 // connect the photoresistor to GPIO 34
19
20 // LED pins
21 #define redLED_PIN 12
22 #define yellowLED_PIN 13
23 #define greenLED_PIN 14
24 #define blueLED_PIN 15
25 // LED state
26 bool redLEDState = false;
```

## 4. Test Result

After uploading the code, LCD1602 shows the IP address. Open the browser, enter the IP address and you will see the control page.

At this time, you can use the control device to read the value read by the sensor, and you can also control the on and off of the LED.

