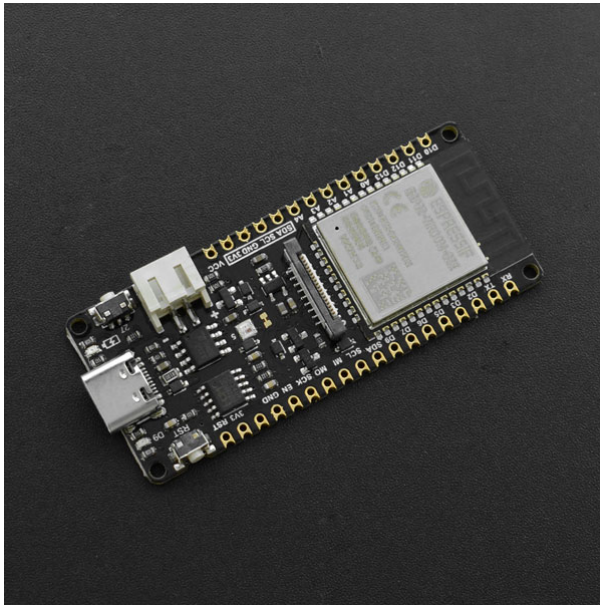


SKU:DFR0654 (<https://www.dfrobot.com/product-2195.html>)



(<https://www.dfrobot.com/product-2195.html>)

1. Introduction

FireBeetle 2 ESP32-E, specially designed for IoT, is an ESP-WROOM-32E-based main controller board with dual-core chips. It supports WiFi and Bluetooth dual-mode communication, and features small size, ultra-low power consumption, on-board charging circuit and easy-to-use interface, which can be conveniently used for smart home IoT, industrial IOT applications, wearable devices and so on. You can easily create your own IoT smart home system when connecting it with an IoT platform like IFTTT. FireBeetle 2 ESP32-E supports Arduino programming, and will support Scratch graphical programming and MicroPython programming very soon. We provide you with detailed online tutorials and application cases, and there are thousands of sensors with welding-free Gravity interface and actuators to help you get started easily.

Besides, the stamp hole design makes it able to be easily embedded in your PCB, greatly saving your costs and time to build and test prototype.

2. What is FireBeetle Series?

FireBeetle was originally designed to be a high-performance and more Mini Arduino open-source development board series. Now it is not only fully compatible with Arduino development environment, but also comes with abundant hardware and software resources. FireBeetle will support the various development environment like MakeCode, Mind+, Pingpong and MicroPython (to be improved soon), which allows you to program your hardware by graphical programming, C language, Python or JS.

This open source board of high-flexibility could bring you infinite possibilities! There are a large number of detailed tutorials and thousands of easy-to-use Gravity peripherals that provide you with the simplest way to program. No matter you are a student, an electronic enthusiast, an artist or a designer, this would be your best partner to open up the world of electronic without dealing with complicated circuits, brain-buring codings, and all complex communication protocols. Turn your worthy ideas into fantastic reality with this FireBeetle series board!

3. Features

- Compatible with DFRobot FireBeetle V2 Series
- Small Size of 25.4×60 mm
- ESP32 Dual-core low power maincontroller, WiFi+BT4.0
- GDI Display Port, esay to connect
- Onboard Charging Circuit and PH2.0 li-ion Battery Port

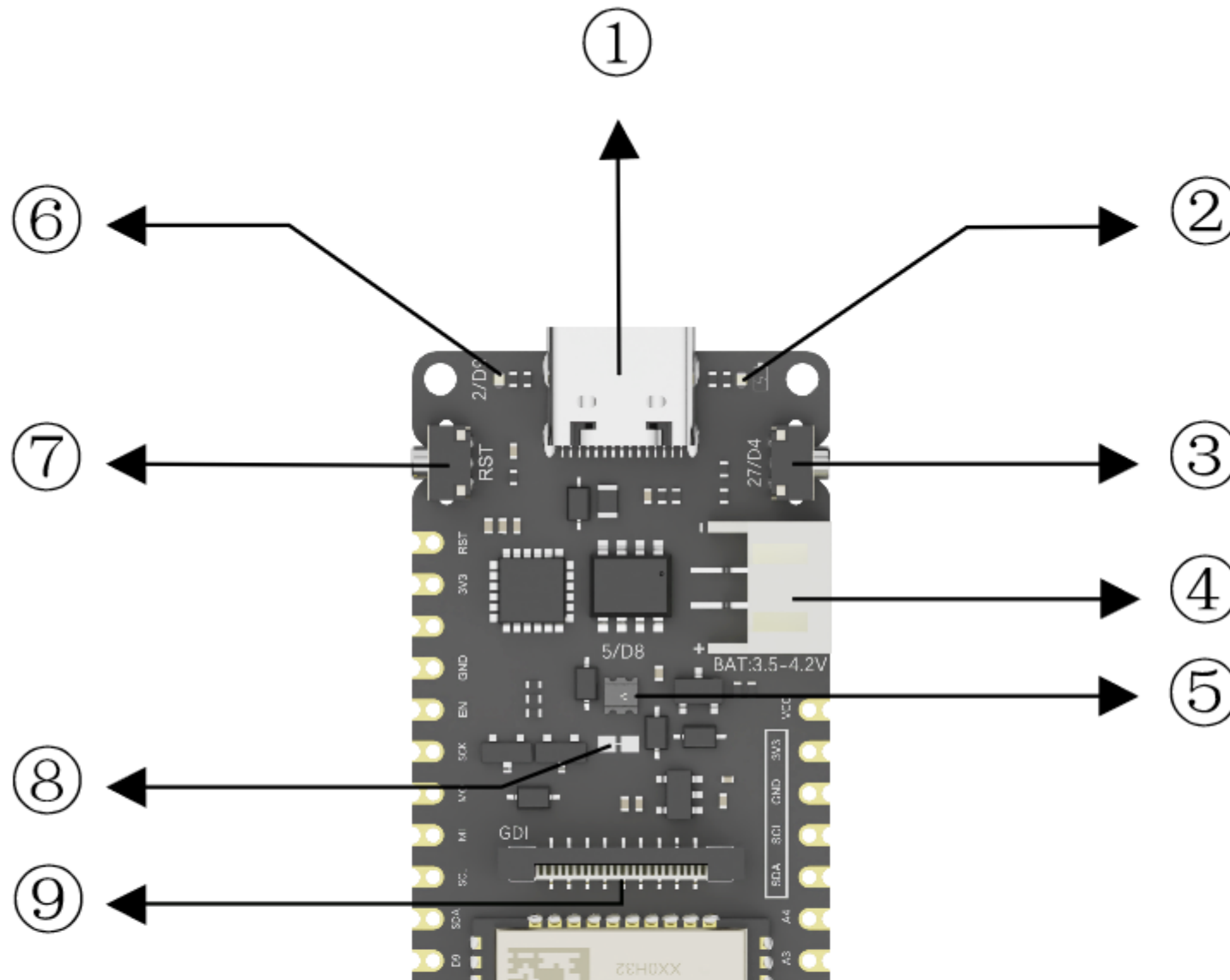
4. Specification

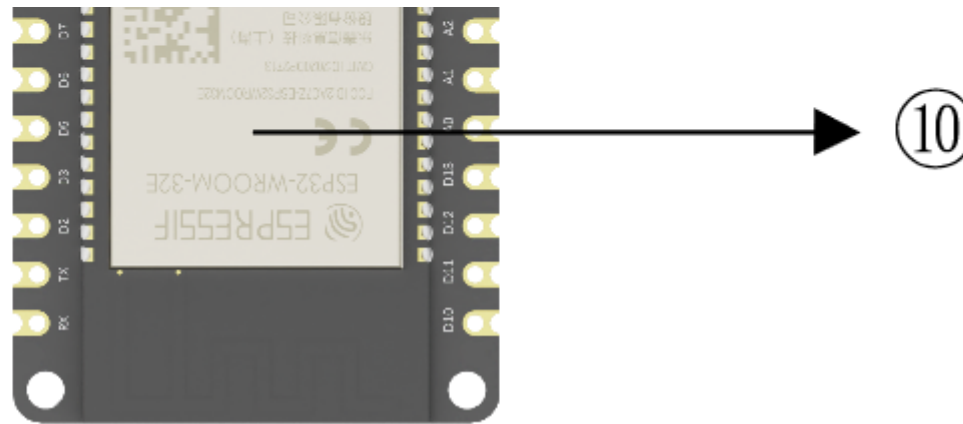
- Operating Voltage: 3.3V

- Input Voltage: 3.3V-5.5V (Support USB Charging)
- Processor: Tensilica LX6 dual-core processor (One for high-speed connection; one for independent application development)
- Main Frequency: 240MHz

- SRAM: 520KB
- Flash: 32Mbit
- Wi-Fi Standard: FCC/CE/TELEC/KCC
- Wi-Fi Protocol: 802.11 b/g/n/d/e/i/k/r (802.11n, speed up to 150 Mbps), A-MPDU and A-MSDU Aggregation, support 0.4us guard interval)
- Frequency Range: 2.4-2.5 GHz
- Bluetooth Protocol: Bluetooth v4.2 BR/EDR and BLE standard compliant
- Bluetooth Audio: CVSD and SBC audio
- On-chip Clock: 40MHz crystal, 32.768KHz crystal
- Digital I/O x18
- Analog Input x11
- SPI x1
- IIC x1
- I2S x1
- RGB_LED: WS2812
- Connector: FireBeetle V2 series compatible
- Operating Temperature: -20°C to +85°C
- Module Size: 25.4 × 60(mm)
- Mount Hole Size: M2, diameter 2.0mm
- Weight: 13g

5. Board Overview

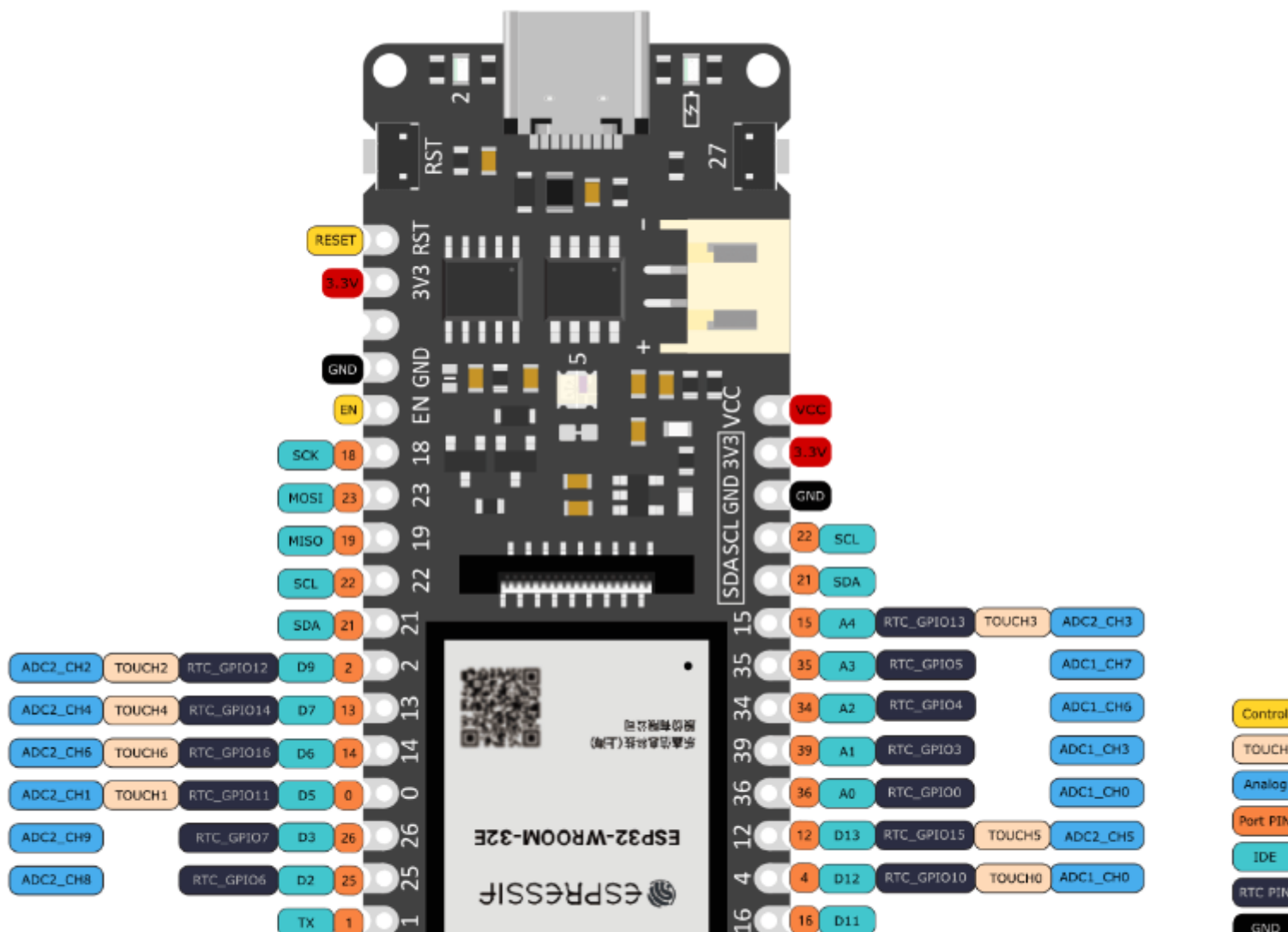




No.	Function	Description
①	USB Interface	Program download and power supply, 4.75V-5.5V compatible
②	Charging Indicator	Red LED for indicating charging status: 1. Off when fully charged or not charged; 2. On when charging; 3. Quick flash when powered by USB, and no battery connected.
③	User Button	Controlled by pin IO27/D4
④	Li-ion Battery Port	Support 3.5V-4.2V
⑤	Onboard RGB Light	WS2812 RGB LED, controlled by pin IO5/D8
⑥	Onboard LED	LED controlled by pin IO2/D9
⑦	Reset Button	Press to reset program
⑧	Low-power Solder Jumper Pad	<p>Designed for low power mode and default to be connected. Slightly cut off the thin wire with a knife to disconnect it. When disconnected, static power consumption can be reduced by 500 μA. The power consumption can be reduced to 13 μA after controlling the maincontroller to enter the sleep mode through the program.</p> <p>Note: when the pad is disconnected, you can only drive RGB LED light via the USB Power supply.</p>

⑨	GDI	DFRobot dedicated Display interface. Refer to the GDI part of this page.
⑩	ESP32-E Chip	ESP32-WROOM-32E

6. Pinout





Category	Description
Control	FireBeetle enable/reset pins
Touch	Pin with capacitive touch function
Analog	Analog pin
Port PIN	Default physical pin number of the chip, which can be used to directly control the corresponding pin
Arduino IDE	In Arduino IDE, the pin numbers have been remapped by FireBeetle, and you can directly use this symbol to control the corresponding pin
RTC PIN	FireBeetle 2 ESP32-E supports low power function, and in Deep-sleep mode, only RTC pin keeps working and can be used as a wake-up source. When RTC pin is used as an output pin, it keeps outputting level value when the chip is in Deep-sleep mode, while as an input pin, it can wake up the chip from Deep-sleep.
GND	Common ground for all power supplies and logics
Power	When powered by 5V-USB, VCC outputs about 4.7V and 3V3 outputs 3.3V; When powered by 4V li-ion battery, VCC outputs about 4V and 3V3 outputs 3.3V(Actual measurement)

FireBeetle 2 ESP32-E has up to 24 physical GPIOs, which are mainly for connecting peripherals like sensors, actuators, etc. Meanwhile, these IO pins can be multiplexed for other functions such as UART, SPI, I2C and so on. The table below provides users with a detailed description of FireBeetle 2 ESP32-E GPIO.

Pin Number	Name	Function	ADC	Communication	Remark

GPIO 0	0/D5	Used as input or output	ADC2_CH1		Occupied when using USB transmission
--------	------	-------------------------	----------	--	--------------------------------------

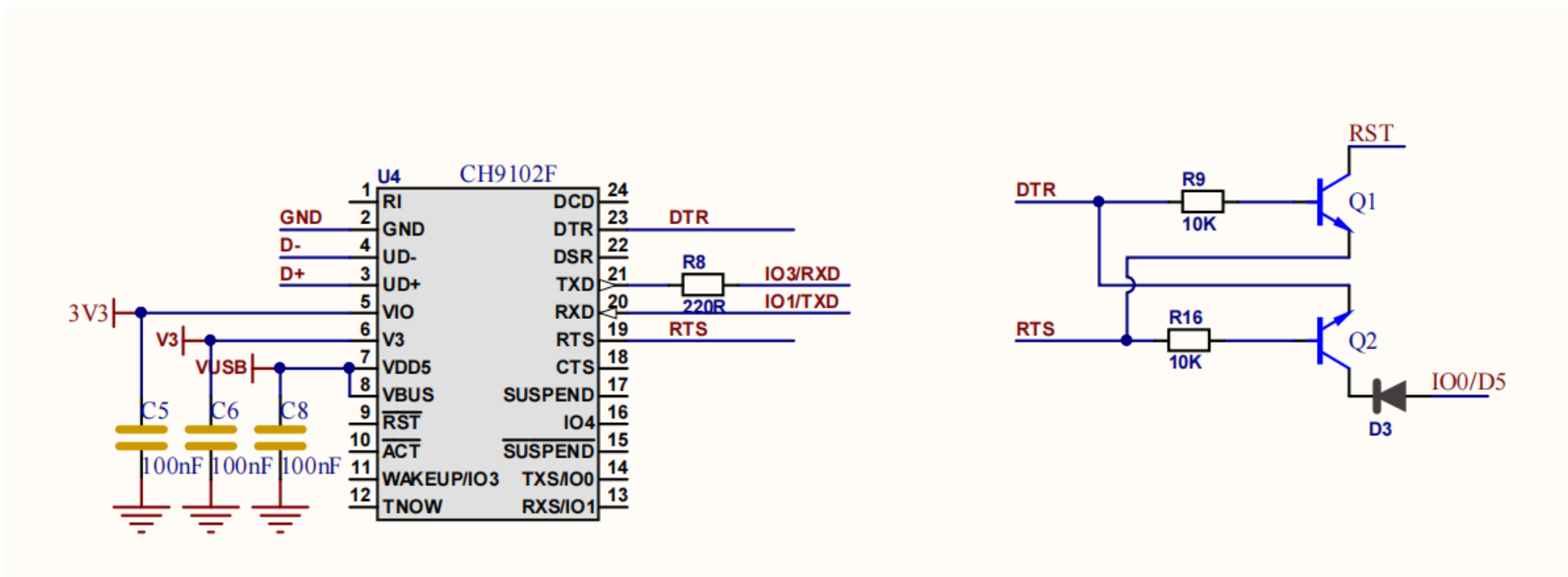
GPIO 1	1/TXD	Used as input or output		UART0_TX	Occupied when using USB power supply and serial printing
GPIO 2	2/D9	Used as input or output	ADC2_CH2		For controlling onboard LED by outputting digital signal
GPIO 3	3/RXD	Used as input or output		UART0_RX	Occupied when using USB power supply and serial printing
GPIO 4	4/D12	Used as input or output	ADC2_CH0		
GPIO 12	12/D13	Used as input or output	ADC2_CH5		
GPIO 13	13/D7	Used as input or output	ADC2_CH4		
GPIO 14	14/D6	Used as input or output	ADC2_CH6		
GPIO 15	15/A4	Used as input or output	ADC2_CH3		
GPIO 16	16/D11	Used as input or output			
GPIO 17	17/D10	Used as input or output			

GPIO 18	18/SCK	Used as input or output		SPI_SCK	
GPIO 19	19/MISO	Used as input or output		SPI_MISO	
GPIO 21	21/SDA	Used as input or output		I2C_SDA	
GPIO 22	22/SCL	Used as input or output		I2C_SCL	
GPIO 23	23/MOSI	Used as input or output		SPI_MOSI	
GPIO 25	25/D2	Used as input or output	ADC2_CH8		DAC_1 (Range: 0-255; Output Voltage: 0-3.3V)
GPIO 26	26/D3	Used as input or output	ADC2_CH9		DAC_2 (Range: 0-255; Output Voltage: 0-3.3V)
GPIO 34	34/A2	Used as input only	ADC1_CH6		
GPIO 35	35/A3	Used as input only	ADC1_CH7		
GPIO 36	36/A0	Used as input only			
GPIO 39	39/A1	Used as input only			
Note:					

- It is recommended not to multiplex the pin IO0/D5, IO1/TXD and IO3/RX since they will be occupied when using USB-related function.

The USB related circuit design is shown below:

THE USB-RELATED CIRCUIT DESIGN IS SHOWN BELOW:

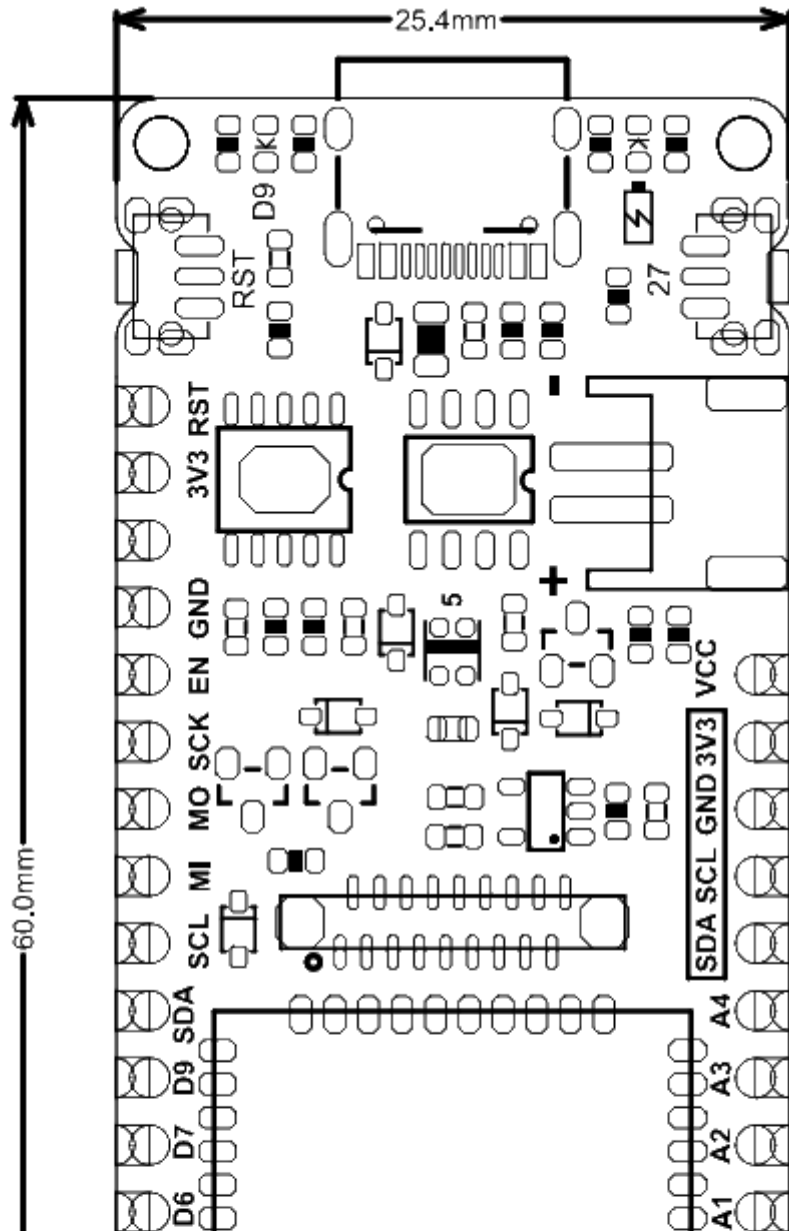


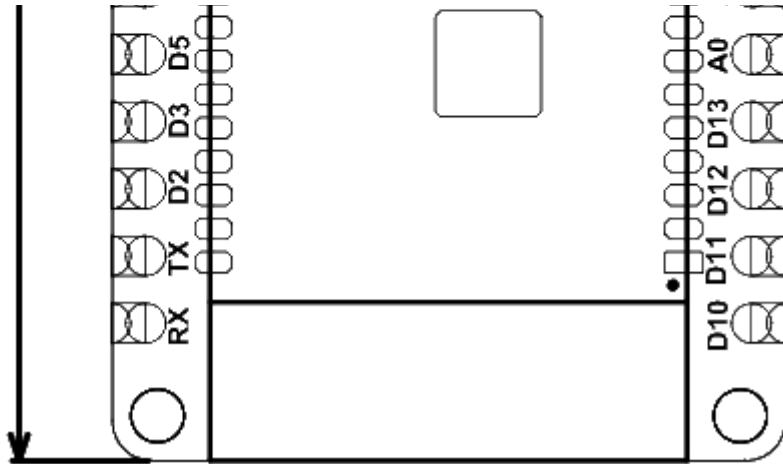
- IO34-39 are for input only.
- FireBeetle 2 ESP32-E has two 8-bit DAC channel that converts 2-way digital signal to 2 analog voltage outputs. Two channels can work independently. DAC circuit is composed of built-in resistors in series and a buffer. The two DACs can be used as reference voltage.

7. Dimension

- Pin Pitch: 2.54mm
- Mounting Hole Pitch: 22mm or 56.6mm

- Mounting Hole Size: 2mm
- Board Size: 25.4×60mm/1×2.36"
- Thickness: 1.6mm






8. Getting Started (Use for first time)

8.1 Download Arduino IDE

- Click to enter the official Arduino website (<https://www.arduino.cc/en/Main/Software>)
- Select and download the Arduino IDE package that is appropriate for your computer OS.

Downloads



Arduino IDE 1.8.19

The open-source Arduino Software (IDE) makes it easy to write code and upload it to the board. This software can be used with any Arduino board.


Refer to the [Getting Started](#) page for Installation instructions.

SOURCE CODE

Active development of the Arduino software is [hosted by GitHub](#). See the instructions for [building the code](#). Latest release source code archives are available [here](#). The archives are PGP-signed so they can be verified using [this](#) gpg key.

DOWNLOAD OPTIONS

Windows Win 7 and newer
Windows ZIP file

Windows app Win 8.1 or 10 

Linux 32 bits
Linux 64 bits
Linux ARM 32 bits
Linux ARM 64 bits

Mac OS X 10.10 or newer

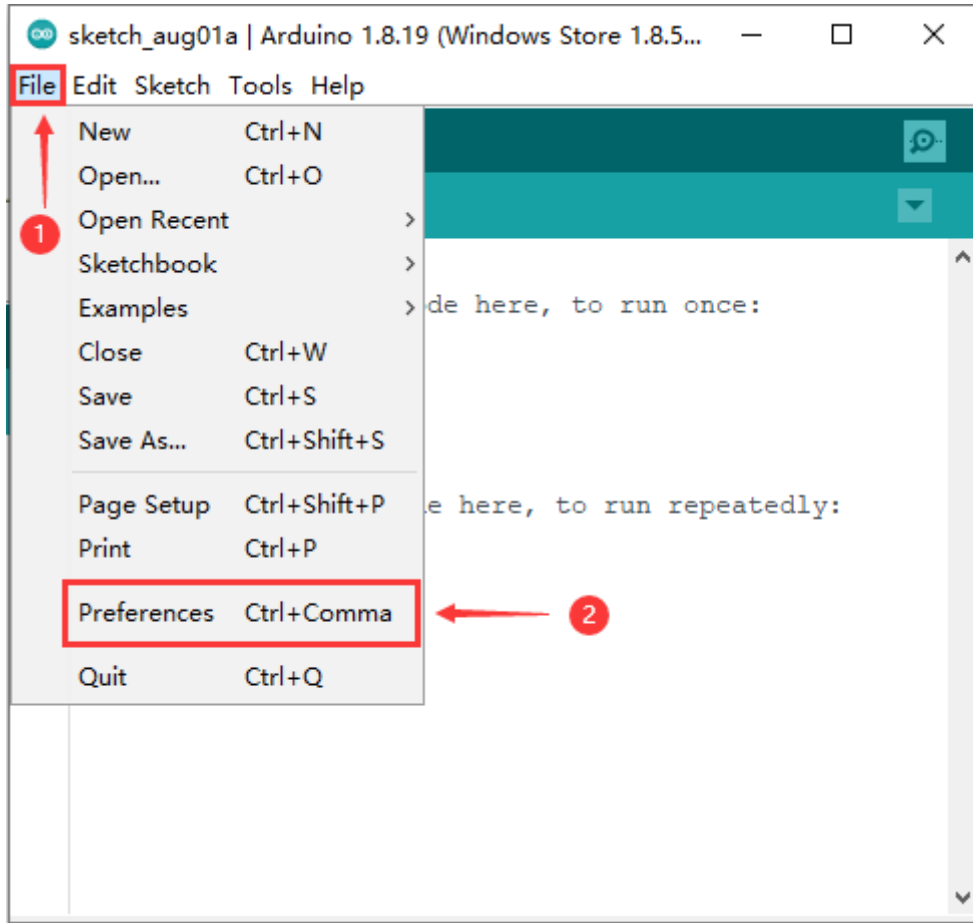
[Release Notes](#)

[Checksums \(sha512\)](#)

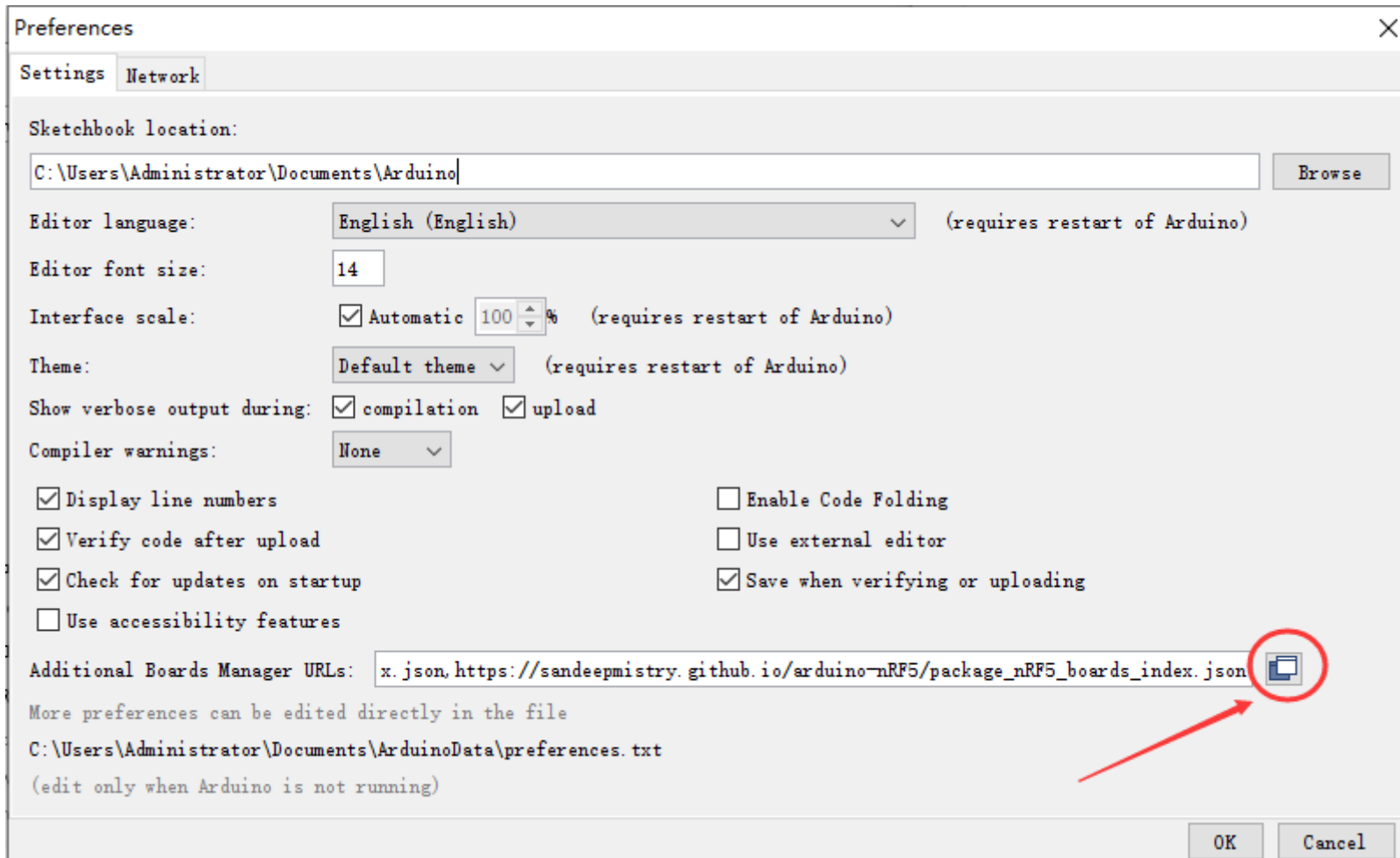
8.2 Configure Arduino IDE

It is required to install ESP32 board in Arduino IDE for using FireBeetle 2 ESP32-E for the first time.

- Open Arduino IDE, click File-Preferences, as shown below



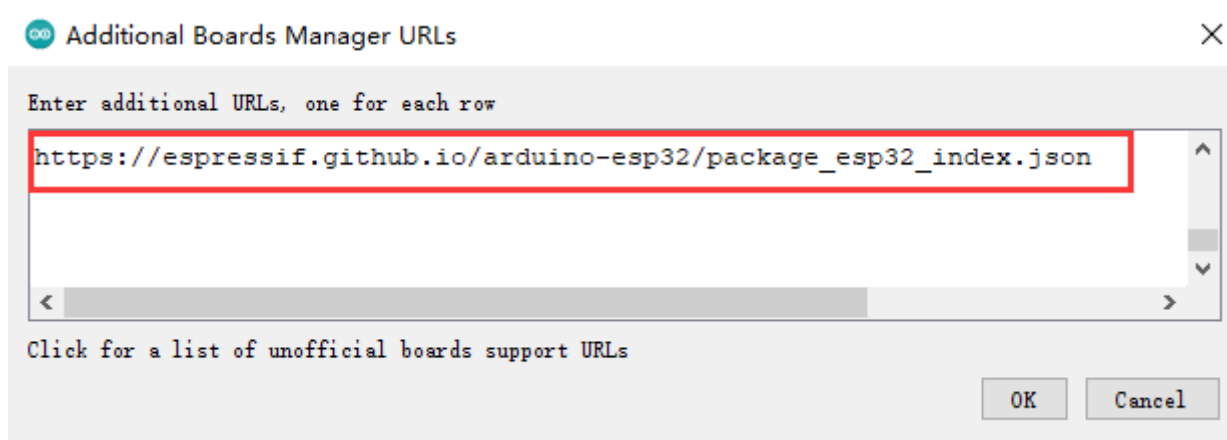
- Click the icon circled in red in the **Preferences** window.



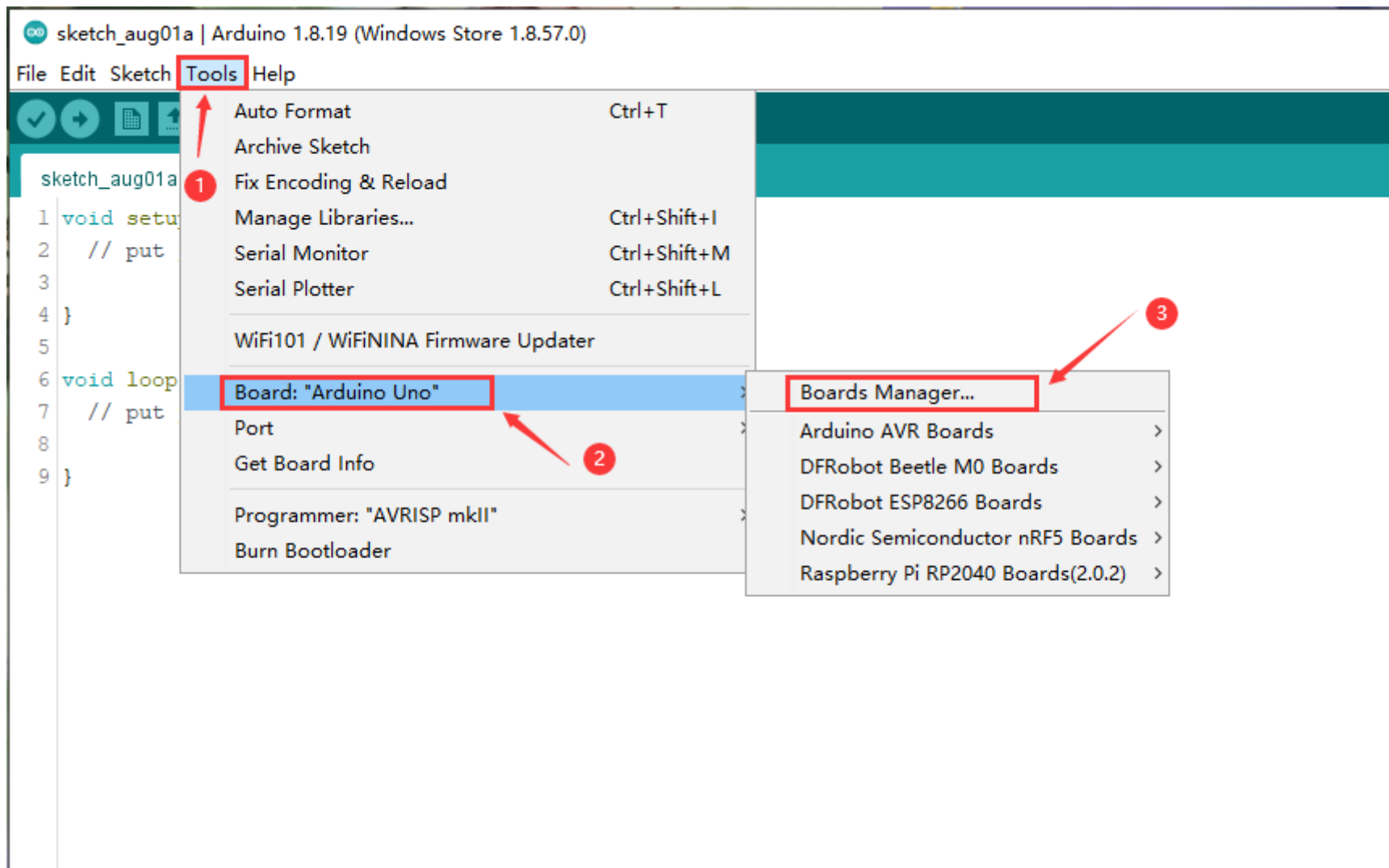
- Add the json URL below to the newly-popped up window. There should be one URL for each line.

https://espressif.github.io/arduino-esp32/package_esp32_index.json (https://espressif.github.io/arduino-esp32/package_esp32_index.json)

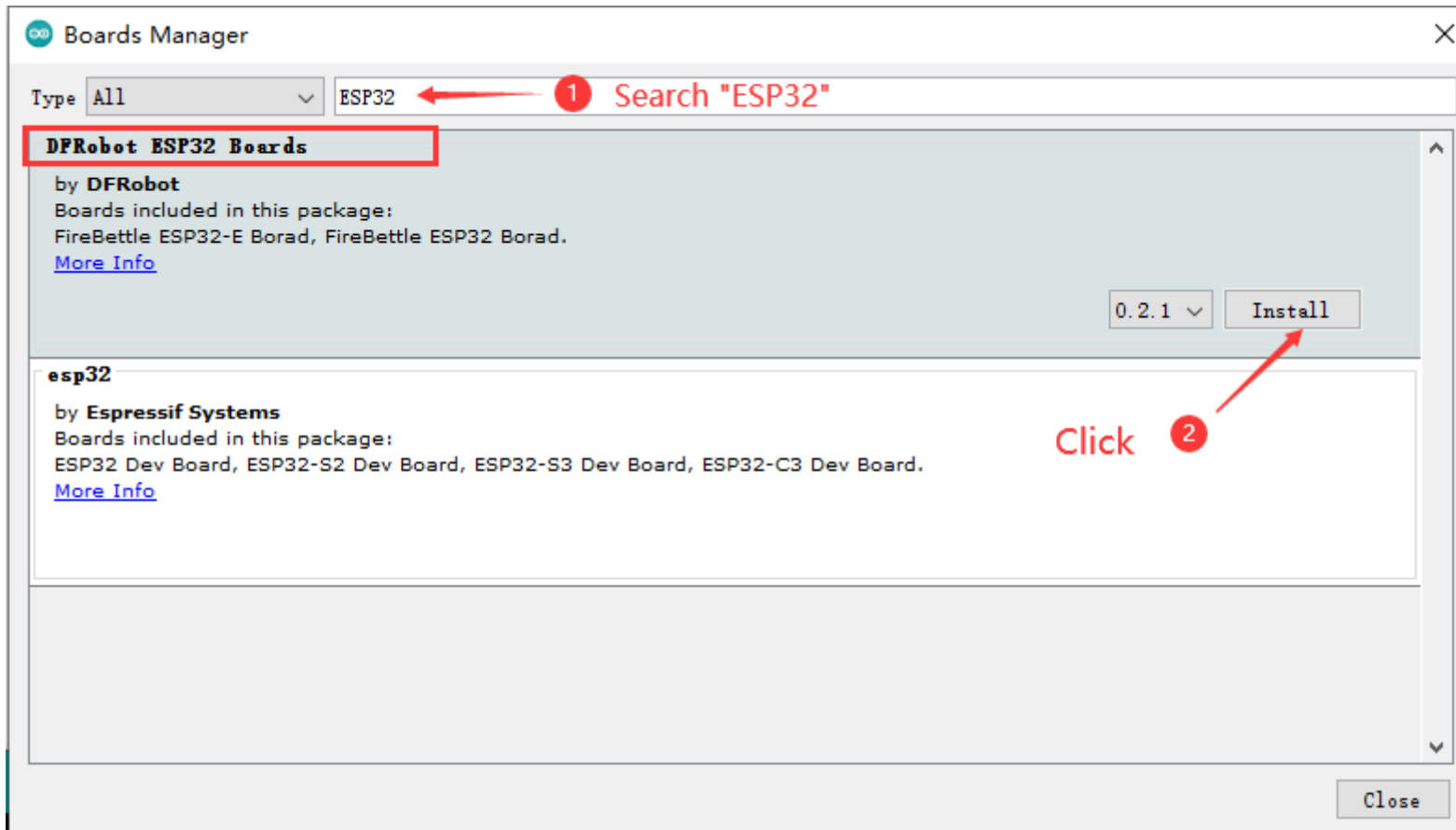
Click OK then.



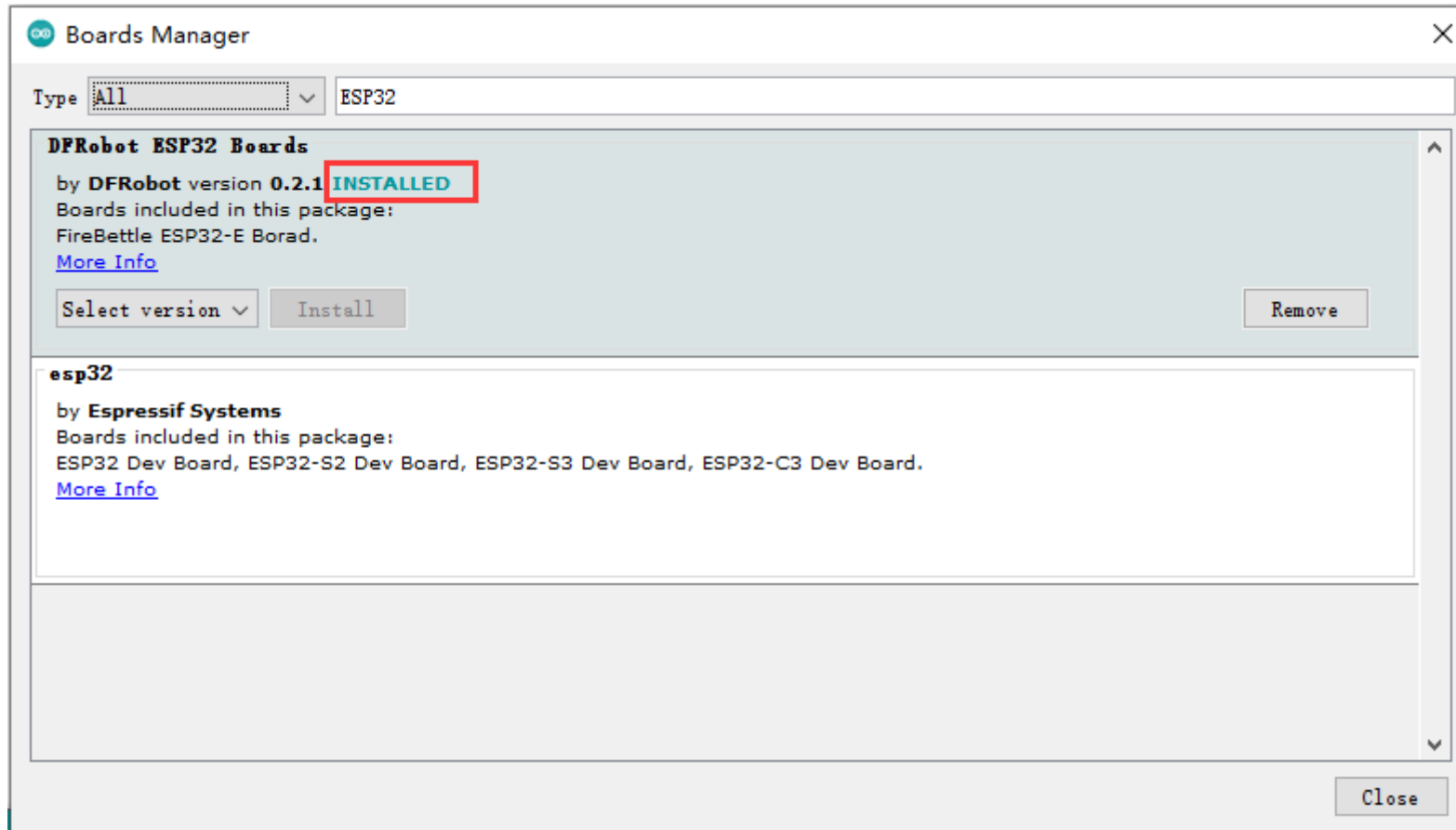
- Download SDK. Click **Tools->Board>Boards Manager**.



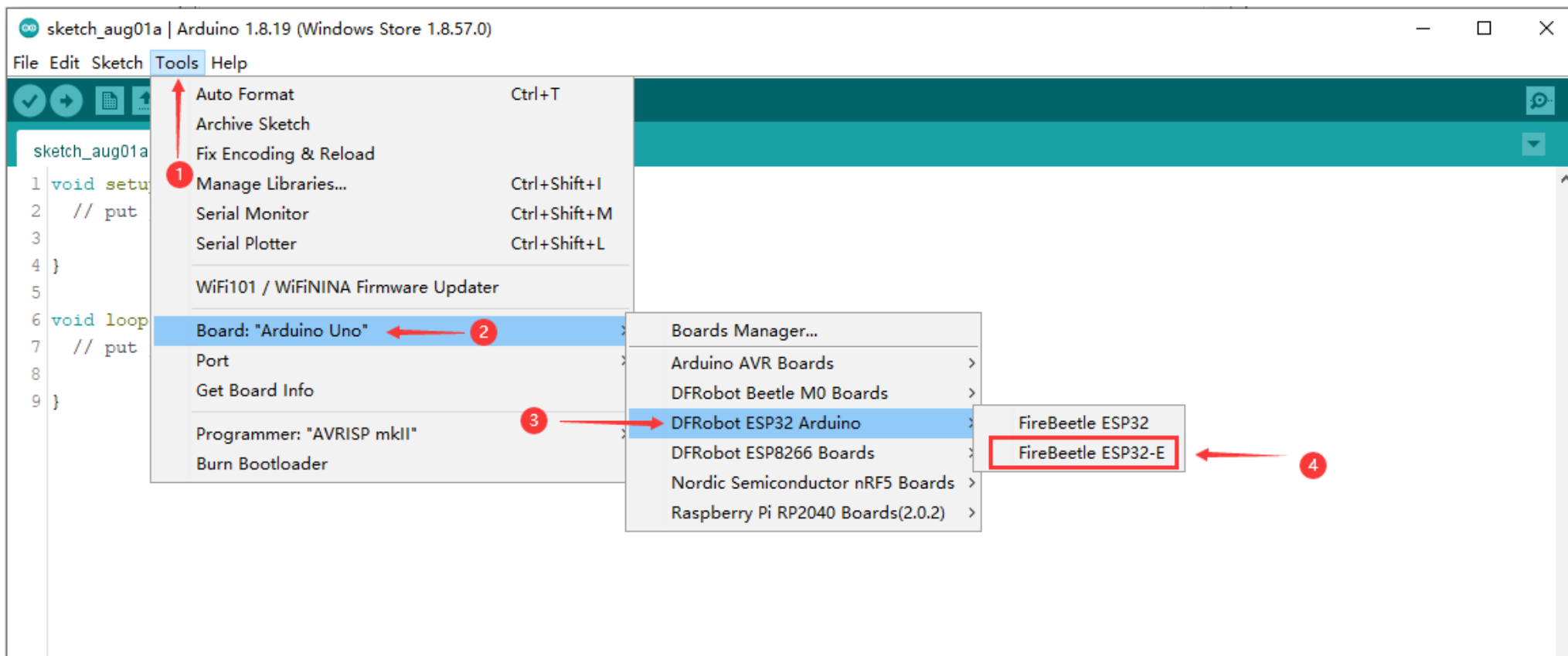
- Enter "ESP32" in the search bar, then the SDK for **DFRobot ESP32 Boards** will appear automatically. Click install now.



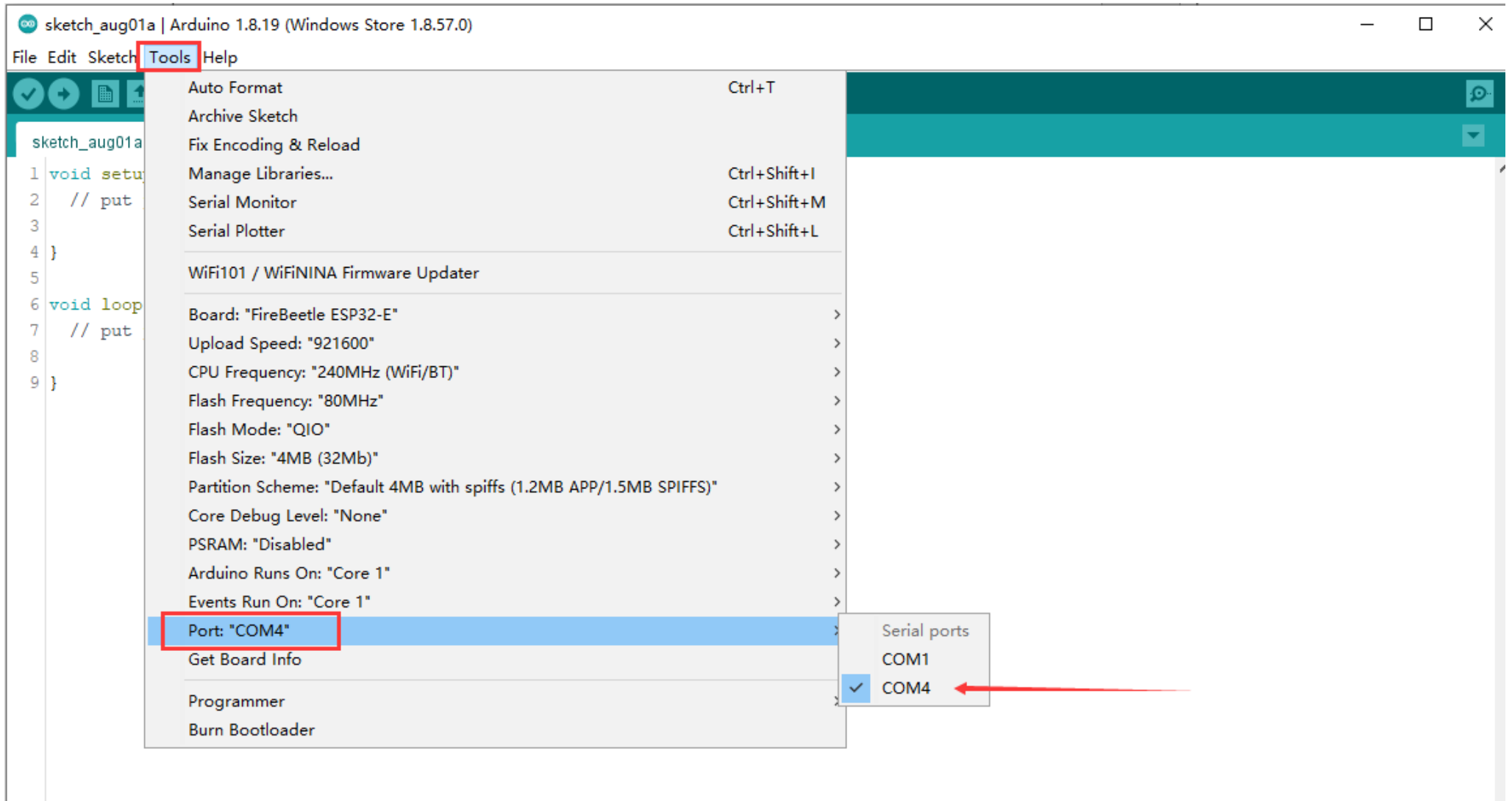
- When the SDK installation completes, the word "installed" will appear at the position marked in red below. Close the window.



- Select FireBeetle 2 ESP32-E development board. Click **Tools-->Board-->DFRobot ESP32 Arduino-->FireBeetle ESP32-E**, as shown below.



- After that, connect your FireBeetle 2 ESP32-E to your computer. Here, a port COM4 comes out after the connection, indicating that it is the port for FireBeetle 2 ESP32-E. Select it as the way shown below.



- Now, the FireBeetle 2 ESP32-E is configured in Arduino IDE and it's ready to go.

9. How to Use FireBeetle 2 ESP32-E on Arduino - Basics

9.1 Blink an LED

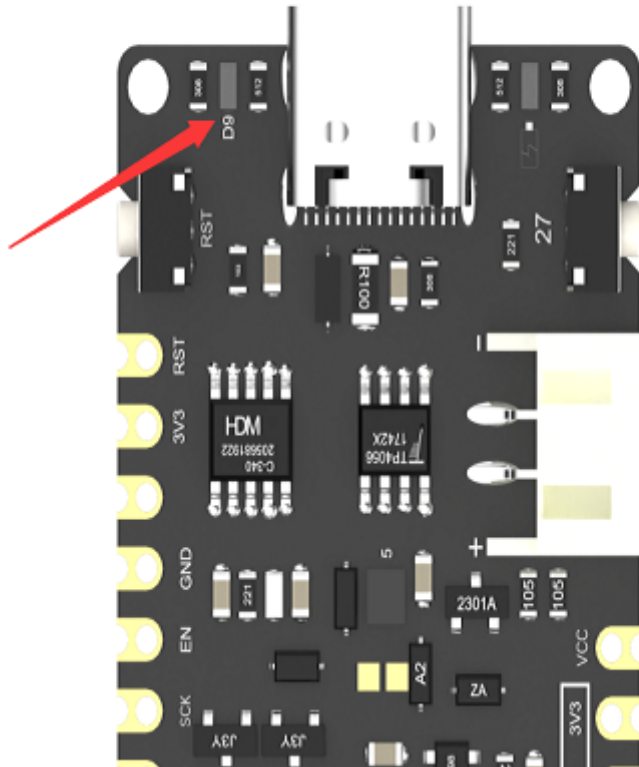
This chapter starts with blinking an LED to demonstrate the usage of FireBeetle 2 ESP32-E.

Preparation:

- FireBeetle 2 ESP32-E (<https://www.dfrobot.com/product-2195.html>) (SKU: DFR01654) ×1

Function Description:

The LED(circled in red below) on FireBeetle 2 ESP32-E is default to be connected to pin 2/D9. Now make it blink by programming.



Sample Code:

The on-board LED blinks at an interval of 1 second.

```
int ledPin = D9;    //Define LED pin
void setup(){
  pinMode(ledPin, OUTPUT); // Set ledPin as output mode
}

void loop(){
  digitalWrite(ledPin, HIGH); // Outputting high, the LED turns on
  delay(1000); //Delay 1 second
  digitalWrite(ledPin, LOW); // Outputting low, the LED turns off
  delay(1000);
}
```

Result:

When the program is uploaded, the on-board LED blinks at a one-second interval repeatedly.

9.2 GPIO

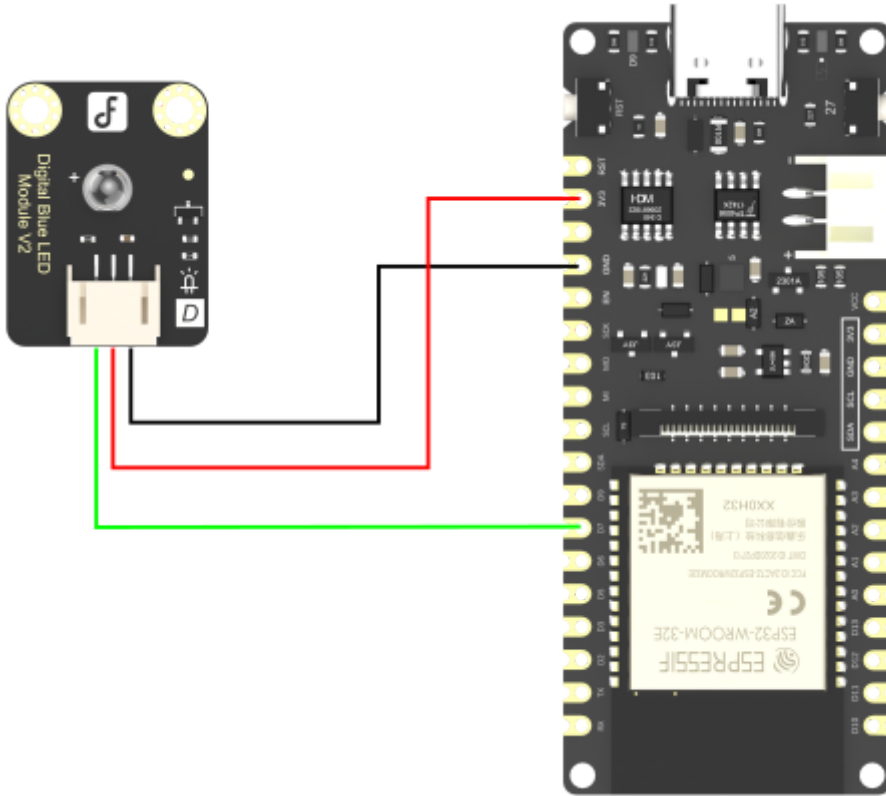
9.2.1 Digital Pin

9.2.1.1 Digital Write Pin

Preparation:

- FireBeetle 2 ESP32-E (<https://www.dfrobot.com/product-2195.html>) (SKU: DFR0654) ×1
- Gravity: Digital Blue LED Light Module (<https://www.dfrobot.com/product-492.html>) (SKU: DFR0021-B) ×1

Connection:



Sample Code:

When the digital pin outputs high, the LED turns on.

```
int ledPin = D7; //Define LED pin
void setup(){
  pinMode(ledPin, OUTPUT);
  digitalWrite(ledPin, HIGH);
}
```

```
void loop(){  
}
```

Result:

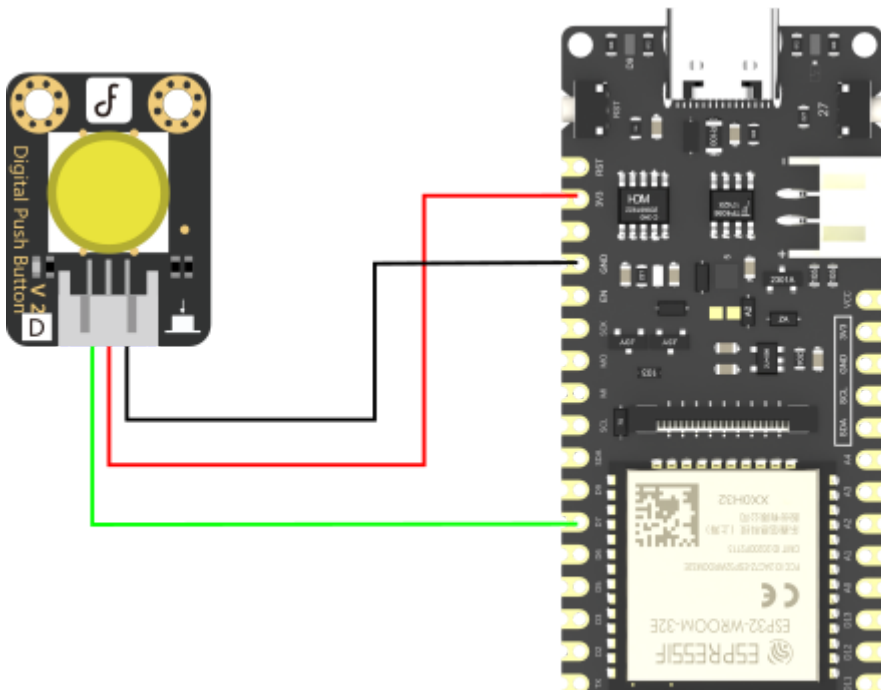
When the program is uploaded, the blue LED lights up.

9.2.1.2 Digital Read Pin

Preparation:

- FireBeetle 2 ESP32-E (<https://www.dfrobot.com/product-2195.html>) (SKU: DFR0654) ×1
- Gravity: Digital Yellow Push Button (<https://www.dfrobot.com/product-73.html>) (SKU: DFR0029-Y) ×1

Connection:





Sample Code:

The on-board LED lights up when the button is pressed, and turns off when released.

```
int buttonPin = D7; //Define button pin
int ledPin = D9; //Define LED pin
int buttonState = 0; //Variable for reading button statu

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

void loop(){
  buttonState = digitalRead(buttonPin); //Read the status of button value

  if(buttonState == HIGH){ //If the button is pressed
    digitalWrite(ledPin,HIGH);
  } else{
    digitalWrite(ledPin,LOW);
  }
}
```

Results

When the program is uploaded, press down the button connected to pin D7, the on-board LED turns on; Release, it goes off.

9.2.2 Analog Pin

9.2.2.1 Analog Read Pin

Preparation:

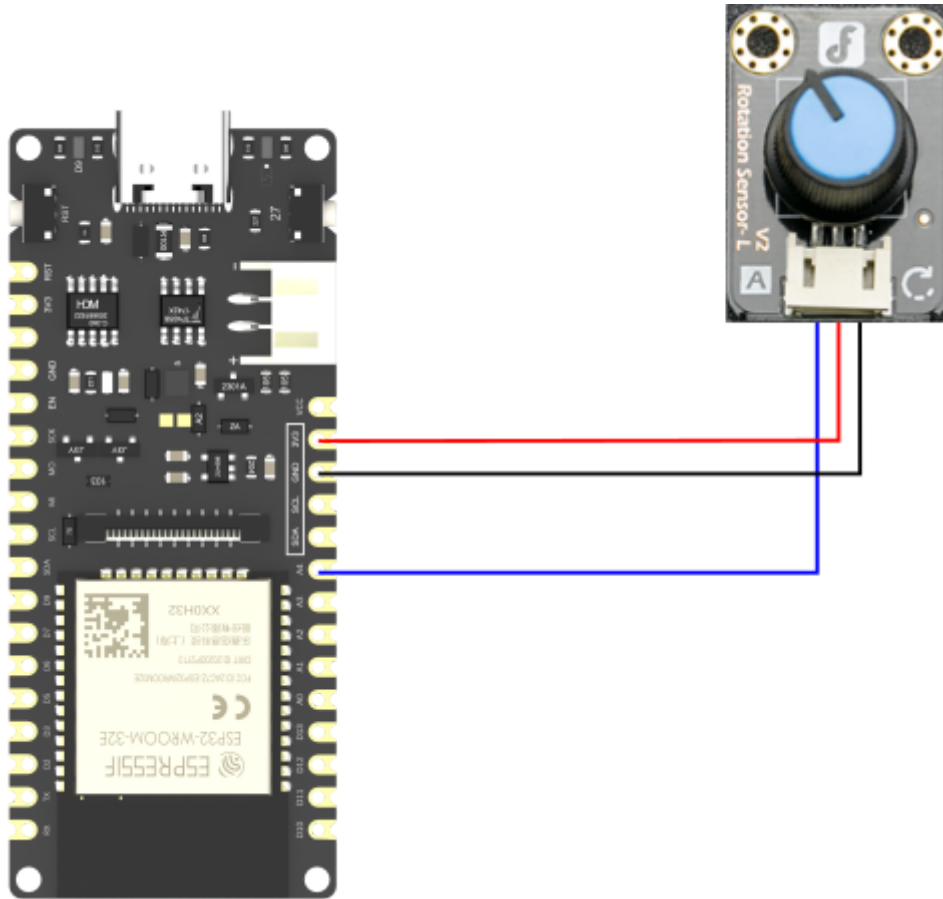
- FireBeetle 2 ESP32-E (<https://www.dfrobot.com/product-2195.html>) (SKU: DFR0654) ×1
- Gravity: Analog Rotation Potentiometer Sensor for Arduino - Rotation 300° (<https://www.dfrobot.com/product-87.html>) (SKU: DFR0054) ×1

What is ADC?

Analog-to-Digital converters (ADC) (<https://wiki.analog.com/university/courses/electronics/text/chapter-20>) translate analog signals, real world signals like temperature, pressure, voltage, current, distance, or light intensity, into a digital representation of that signal. This digital representation can then be processed, manipulated, computed, transmitted or stored. FireBeetle 2 ESP32-E has a 12-bit ADC with a max output of 4095.

Note: The ADC on ESP32 can only measure around 2.5V instead of 3.3V, as shown in the example below.

Connection:



Sample Code:

The program below can be used to read the sensor module connected to the pin IO15/A4, and then print out the real-time analog angle sensor readings and the detected voltage on serial monitor.

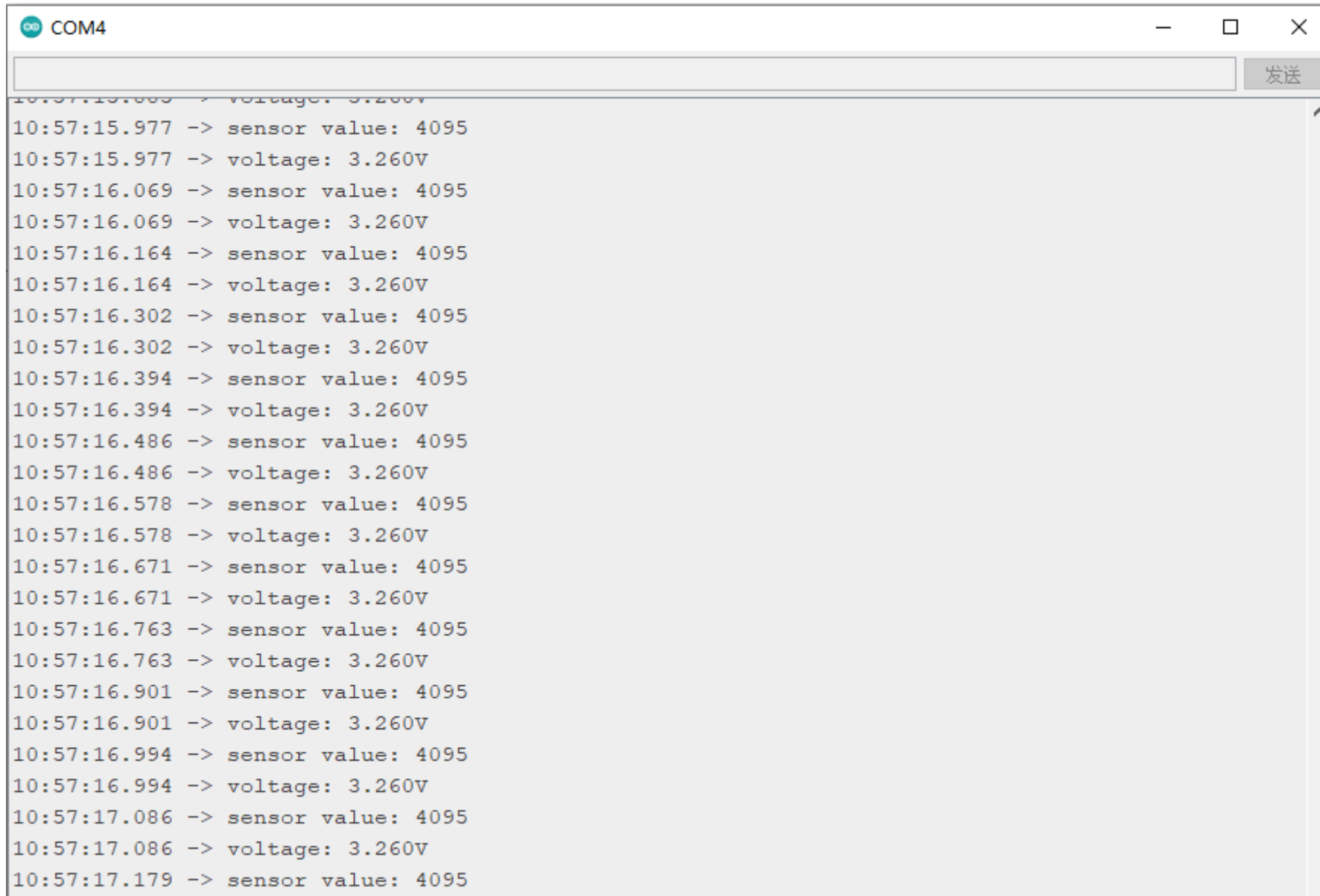
```
int sensorPin = A4; // Define analog angle sensor pin
int sensorValue = 0;

void setup(){
  pinMode(sensorPin, INPUT);
  Serial.begin(9600); //Initialize serial port
}

void loop(){
  sensorValue = analogRead(sensorPin); //Read the sensor value on analog pin A4
  Serial.printf("sensor value: %d\n", sensorValue); //Print the read sensor value
  Serial.printf("voltage: %.3fV\n", sensorValue * 3.26 / 4095); //Print the detected voltage
  delay(100);
}
```

Result:

Open the serial monitor and you can see the current value from the angle sensor printed in it. When the angle sensor is rotated, the printed value changes accordingly. It can be observed that when the sensor value is 4095, the detected voltage is about 3.3V.



```
COM4
10:57:15.977 -> sensor value: 4095
10:57:15.977 -> voltage: 3.260V
10:57:16.069 -> sensor value: 4095
10:57:16.069 -> voltage: 3.260V
10:57:16.164 -> sensor value: 4095
10:57:16.164 -> voltage: 3.260V
10:57:16.302 -> sensor value: 4095
10:57:16.302 -> voltage: 3.260V
10:57:16.394 -> sensor value: 4095
10:57:16.394 -> voltage: 3.260V
10:57:16.486 -> sensor value: 4095
10:57:16.486 -> voltage: 3.260V
10:57:16.578 -> sensor value: 4095
10:57:16.578 -> voltage: 3.260V
10:57:16.671 -> sensor value: 4095
10:57:16.671 -> voltage: 3.260V
10:57:16.763 -> sensor value: 4095
10:57:16.763 -> voltage: 3.260V
10:57:16.901 -> sensor value: 4095
10:57:16.901 -> voltage: 3.260V
10:57:16.994 -> sensor value: 4095
10:57:16.994 -> voltage: 3.260V
10:57:17.086 -> sensor value: 4095
10:57:17.086 -> voltage: 3.260V
10:57:17.179 -> sensor value: 4095
```

9.2.2.2 PWM Output (Analog Write)

Preparation:

- FireBeetle 2 ESP32-E (<https://www.dfrobot.com/product-2195.html>) (SKU: DFR0654) ×1

What is PWM?

PWM (Pulse Width Modulation) is a very effective technique to control analog circuits using the digital output of MCU. It is widely used in many fields such as lighting control, motor speed control, measurement, communication, power control and conversion.

The PWM controller of ESP32 has 16 independent channels, which can be configured to generate PWM signals with different properties. All pins that can be used as outputs can serve as PWM pins (GPIO 34 to 39 cannot generate PWM).

In the Arduino IDE, PWM output is defined as analog output. When using Arduino for PWM LED dimming, you should follow the steps below:

1. First, select a PWM channel. There are 16 channels from 0 to 15.
2. Then, set the PWM signal frequency. For LEDs, a frequency of 5000 Hz is suitable.
3. Set the duty cycle resolution of the signal, ranging from 1 to 16 bits. Here an 8-bit resolution is used. And we can use the value between 0 and 255 (2 to the power of 8) to control the LED brightness.

The sample code below will demonstrate how to drive the onboard LED using PWM to display a breathing light effect. No external hardware connection is required for this example.

Sample Code:

Function: Drive the onboard LED using PWM to display a breathing light effect.


```
const int ledPin = D9; //Define LED Pin
const int freq = 5000; //Set PWM signal frequency
const int ledChannel = 0; //There are 16 channels from 0 to 15, set to PWM channel 0
const int resolution = 8; //Set the duty cycle resolution of the signal, from 1 to 16 bits. Select 8-bit resolution here

void setup(){
  ledcSetup(ledChannel,freq,resolution);
  ledcAttachPin(ledPin,ledChannel); //Set the pin for outputting PWM signals and the channel for generating PWM signals
}

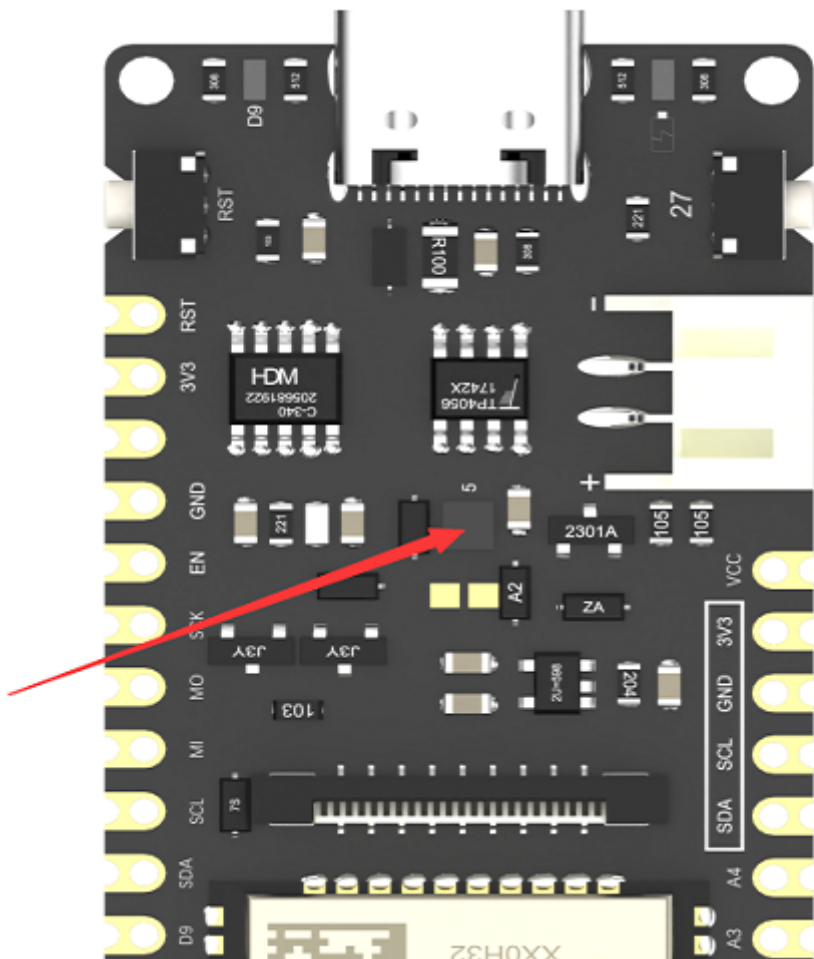
void loop(){
  for(int dutyCycle = 0;dutyCycle <= 255;dutyCycle++){
    ledcWrite(ledChannel,dutyCycle);
    delay(15);
  }
  for(int dutyCycle = 255;dutyCycle >= 0;dutyCycle--){
    ledcWrite(ledChannel,dutyCycle);
    delay(15);
  }
}
```

Result:

After the program is uploaded successfully, the light intensity of the on-board green LED appears to rise and fall in a manner that resembles human breathing.

9.3 RGB LED

Firebeetle 2 ESP32-E has an onboard WS2812 RGB LED connected to the pin IO5/D8, which is not broken out from the board, so IO5/D8 is exclusively reserved for the RGB LED, located as shown in the figure below:



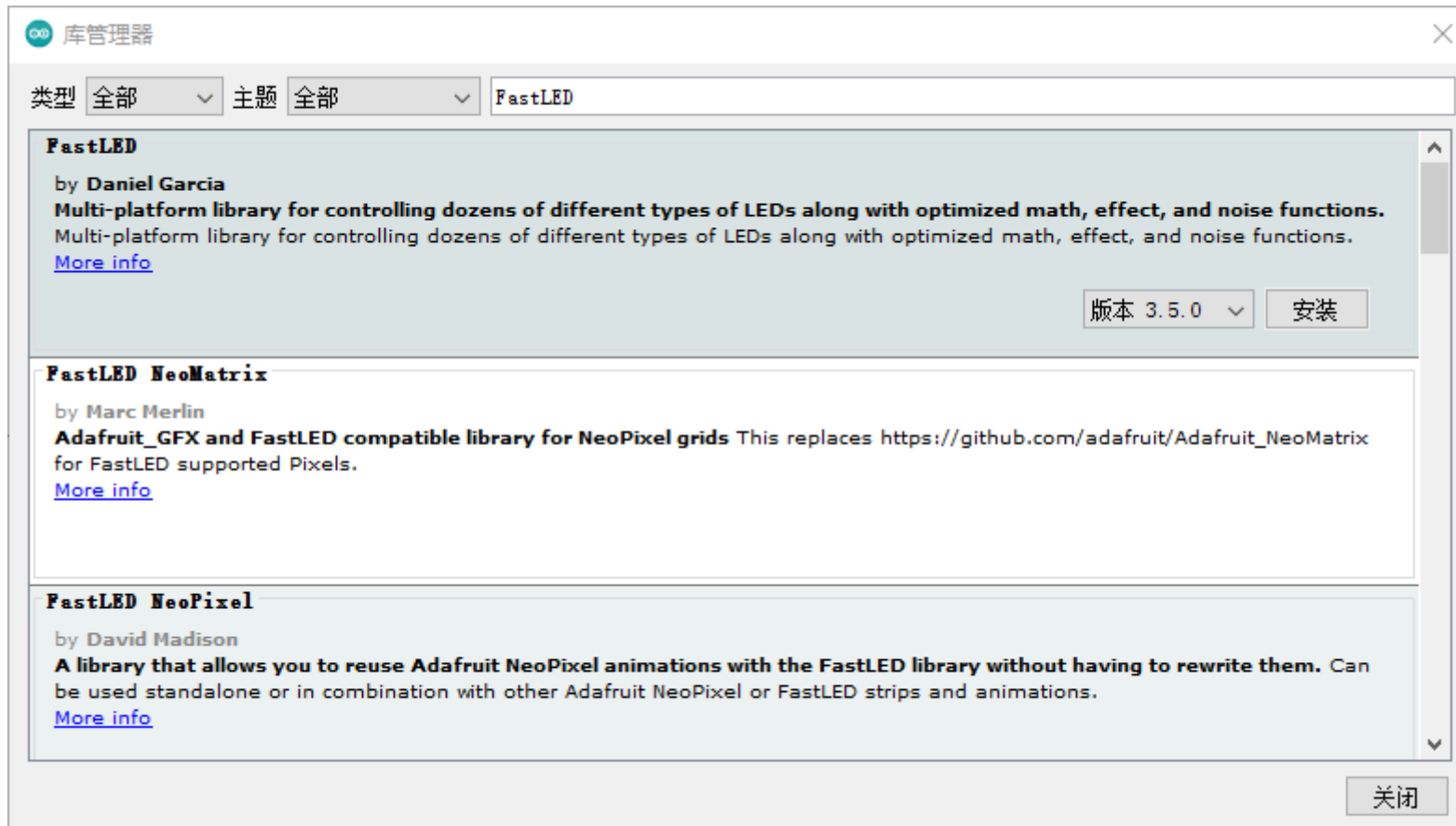
Preparation:

- FireBeetle 2 ESP32-E (<https://www.dfrobot.com/product-2195.html>) (SKU: DFR0654) ×1

Function Description:

https://wiki.dfrobot.com/FireBeetle_Board_ESP32_E_SKU_DFR0654

FastLED is a powerful and easy-to-use Arduino library for controlling LED strips such as WS2810 and LPD8806. Currently, it is widely recognized as one of the most popular LED control libraries for Arduino developers. You can install FastLED by referring to the method for installing the SDK of FireBeetle 2 ESP32-E before, as shown below, click "Install".



Next, we'll learn how to light up the RGB LED without external hardware connections.

Sample Code:

Program function: Light up the on-board RGB LED and make it show red, green, blue and a randomly-mixed color in sequence repeatedly.

```
#include <FastLED.h>
#define NUM_LEDS 1    //Number of RGB LED beads
#define DATA_PIN D8  //The pin for controlling RGB LED
#define LED_TYPE NEOPIXEL //RGB LED strip type
CRGB leds[NUM_LEDS]; //Instantiate RGB LED

void setup() {
    FastLED.addLeds<LED_TYPE, DATA_PIN>(leds, NUM_LEDS); //Initialize RGB LED
}

void loop() {
    leds[0] = CRGB::Red; //LED shows red light
    FastLED.show();
    delay(1000);
    leds[0] = CRGB::Green; //LED shows green light
    FastLED.show();
    delay(1000);
    leds[0] = CRGB::Blue; // LED shows blue light
    FastLED.show();
    delay(1000);
    leds[0] = CRGB(random(0,255),random(0,255),random(0,255)); // LED shows a randomly mixed color
    FastLED.show();
    delay(1000);
}
```

Result:

After the program is successfully uploaded, the on-board RGB LED switches between red, green, blue and randomly mixed colors every second.

9.4 Serial Port

Preparation:

- FireBeetle 2 ESP32-E (<https://www.dfrobot.com/product-2195.html>) (SKU: DFR0654) ×1

Function Description:

We have learned how to light up the LED through the IO port before. In this lesson, we'll learn about the principle of serial communication. FireBeetle 2 ESP32-E development board has two hardware serial ports, and both of them are remappable. UART0, also Serial in Arduino, is used for USB.

Serial Name	Arduino	TX	RX
UART0	Serial	pin1	pin3
UART2	Serial2	pin17	pin16

Sample Code:

Program function: the UART serial prints timing data once per second.

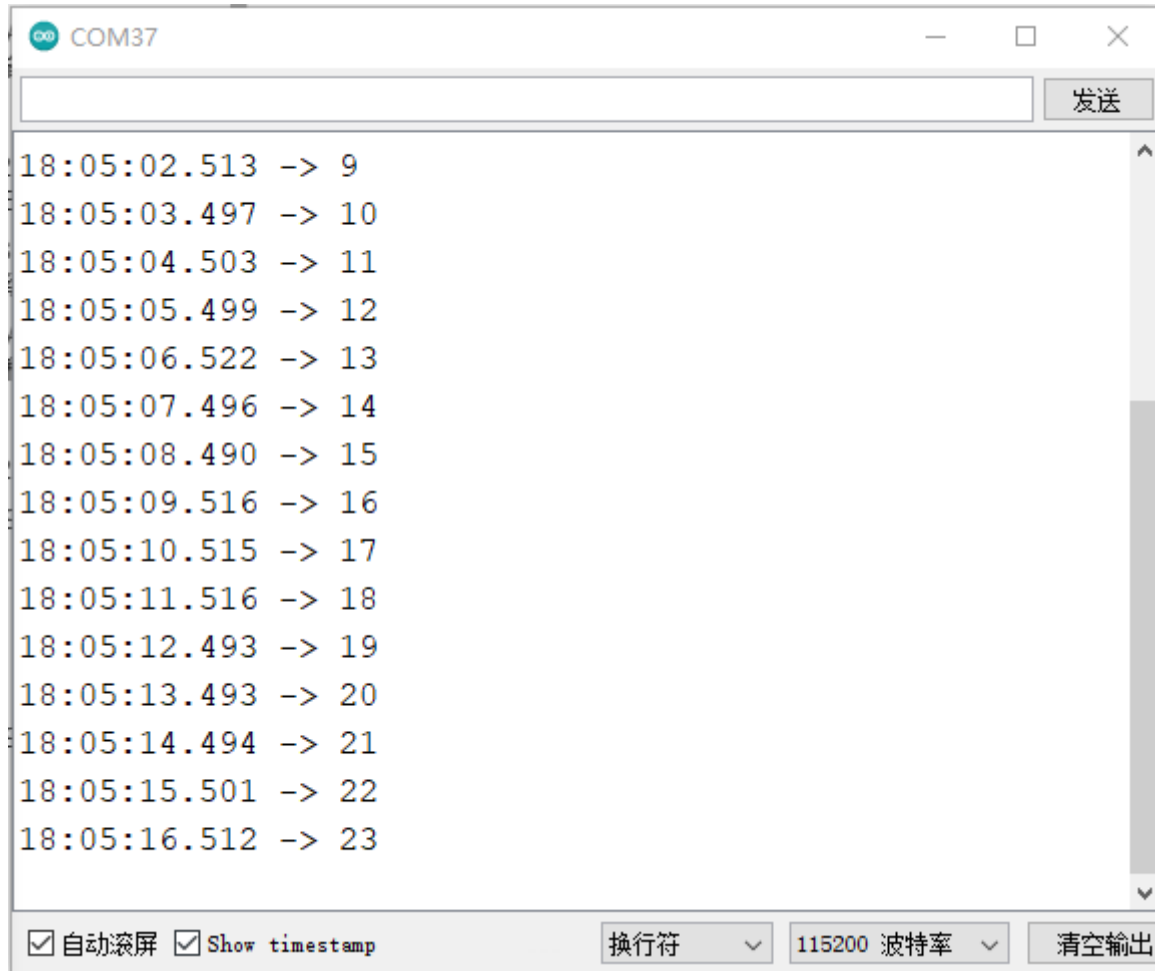
```
void setup() {
  Serial.begin(115200); //Initialize the serial port and set the baud rate to 115200
}

void loop() {
  static unsigned long i = 0; //Static variables(local variables), initialized only once
  Serial.println(i++); //serial prints i++
  delay(1000);
}
```

Result:

Result:

Open the serial monitor, set the baud rate to 115200, and you can see the printed value increasing every second.



The screenshot shows a serial monitor window titled "COM37". The window contains a list of 16 lines of text, each representing a timestamp followed by a value in parentheses, separated by a right-pointing arrow. The values increase by 1 in each successive line. At the bottom of the window, there are several controls: two checked checkboxes labeled "自动滚屏" and "Show timestamp", a dropdown menu for "换行符" (set to "\n"), a dropdown menu for "波特率" (set to "115200"), and a button labeled "清空输出".

```
18:05:02.513 -> 9
18:05:03.497 -> 10
18:05:04.503 -> 11
18:05:05.499 -> 12
18:05:06.522 -> 13
18:05:07.496 -> 14
18:05:08.490 -> 15
18:05:09.516 -> 16
18:05:10.515 -> 17
18:05:11.516 -> 18
18:05:12.493 -> 19
18:05:13.493 -> 20
18:05:14.494 -> 21
18:05:15.501 -> 22
18:05:16.512 -> 23
```

9.5 Capacitive Touch Function

This section introduces how to get and print the status of the touch sensor on FireBeetle 2 ESP32-E by Arduino code.

Preparation:

https://wiki.dfrobot.com/FireBeetle_Board_ESP32_E_SKU_DFR0654

Preparation.

- FireBeetle 2 ESP32-E (<https://www.dfrobot.com/product-2195.html>) (SKU: DFR0654) ×1

Function Description:

FireBeetle 2 ESP32-E provides the function of capacitive touch sensing with a total of 7 touch pins available, labeled as T0 to T6. The corresponding pin numbers are shown in the table below:

Touch Sensor	Pin Number
T0	IO4/D12
T1	IO0/D5
T2	IO2/D9 (Connected to the on-board LED, and can't be used to test the touch sensing function)
T3	IO15/A4
T4	IO13/D7
T5	IO12/D13
T6	IO14/D6

PinMode setting is not required. The touchRead() returns a value within 0-255. The larger the touch intensity, the smaller the returned value. To obtain the GPIO status of the touch sensor, just need to call the touchRead function, whose function prototype is: uint16_t touchRead(uint8_t pin).

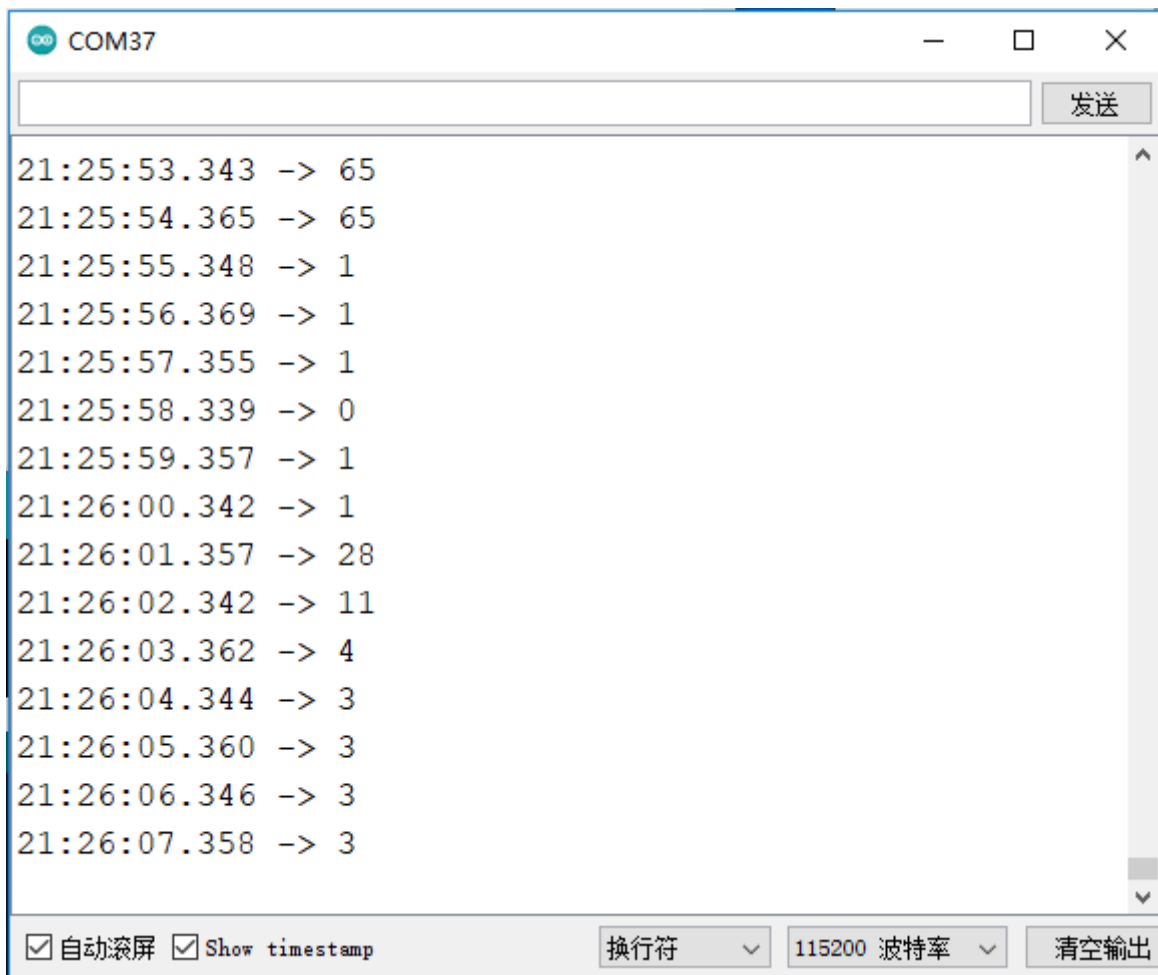
Sample Code:

Program function: Burn the routine, use the pin IO4/D12 as a touch button, and return the touch value through the serial monitor.

```
void setup() {  
  Serial.begin(115200);  
  Serial.println("FireBeetle Board-ESP32 Touch Test");  
}  
  
void loop() {  
  Serial.println(touchRead(T0));  
  delay(100);  
}
```

Result:

The value returned decreases as the intensity of touch on pin IO4/D12 increases, with a value of 65 indicating no touch.



```
COM37
21:25:53.343 -> 65
21:25:54.365 -> 65
21:25:55.348 -> 1
21:25:56.369 -> 1
21:25:57.355 -> 1
21:25:58.339 -> 0
21:25:59.357 -> 1
21:26:00.342 -> 1
21:26:01.357 -> 28
21:26:02.342 -> 11
21:26:03.362 -> 4
21:26:04.344 -> 3
21:26:05.360 -> 3
21:26:06.346 -> 3
21:26:07.358 -> 3
```

自动滚屏 Show timestamp 换行符 115200 波特率 清空输出

9.6 Interrupt

Function Description:

All GPIO pins broken out from the FireBeetle 2 ESP32-E development board, except those that are already occupied or have specific functions, are available for users to use. For more information, please refer to the https://wiki.dfrobot.com/FireBeetle_Board_ESP32_E_SKU_DFR0654.

All GPIO pins broken out from the FireBeetle 2 ESP32-E development board, except those that are already occupied or have specific functions, can be used as external interrupt pins.

- Enable external interrupt(`attachInterrupt(pin, function, mode)`), the parameters are as follows:

pin: the external interrupt pin;

function: the callback function for the external interrupt;

mode: 5 external interrupt modes.

As shown in the table below:

Interrupt Trigger Mode	Description
RISING	Trigger on rising edge
FALLING	Trigger on falling edge
CHANGE	Trigger on any level change
ONLOW	Trigger on low
ONHIGH	Trigger on high

Disable pin interrupt `detchInterrupt(pin)`, no value is returned.

Preparation:

- FireBeetle 2 ESP32-E (<https://www.dfrobot.com/product-2195.html>) (SKU: DFR0654) ×1

Sample Code:

Program function: when the button is pressed, the interrupt is triggered and the serial prints the number of times the button has been pressed.

```
#include <Arduino.h>

struct Button {          //Define the button struct
    const uint8_t PIN;   //Define button pin
    uint32_t numberKeyPresses; //Define the number of times the button is pressed.
    bool pressed;       //Determine if the button is pressed, return true if it's pressed
};
Button button = {27, 0, false}; //Instantiated a button, and use the on-board button.

void ARDUINO_ISR_ATTR isr() { //Interrupt processing function

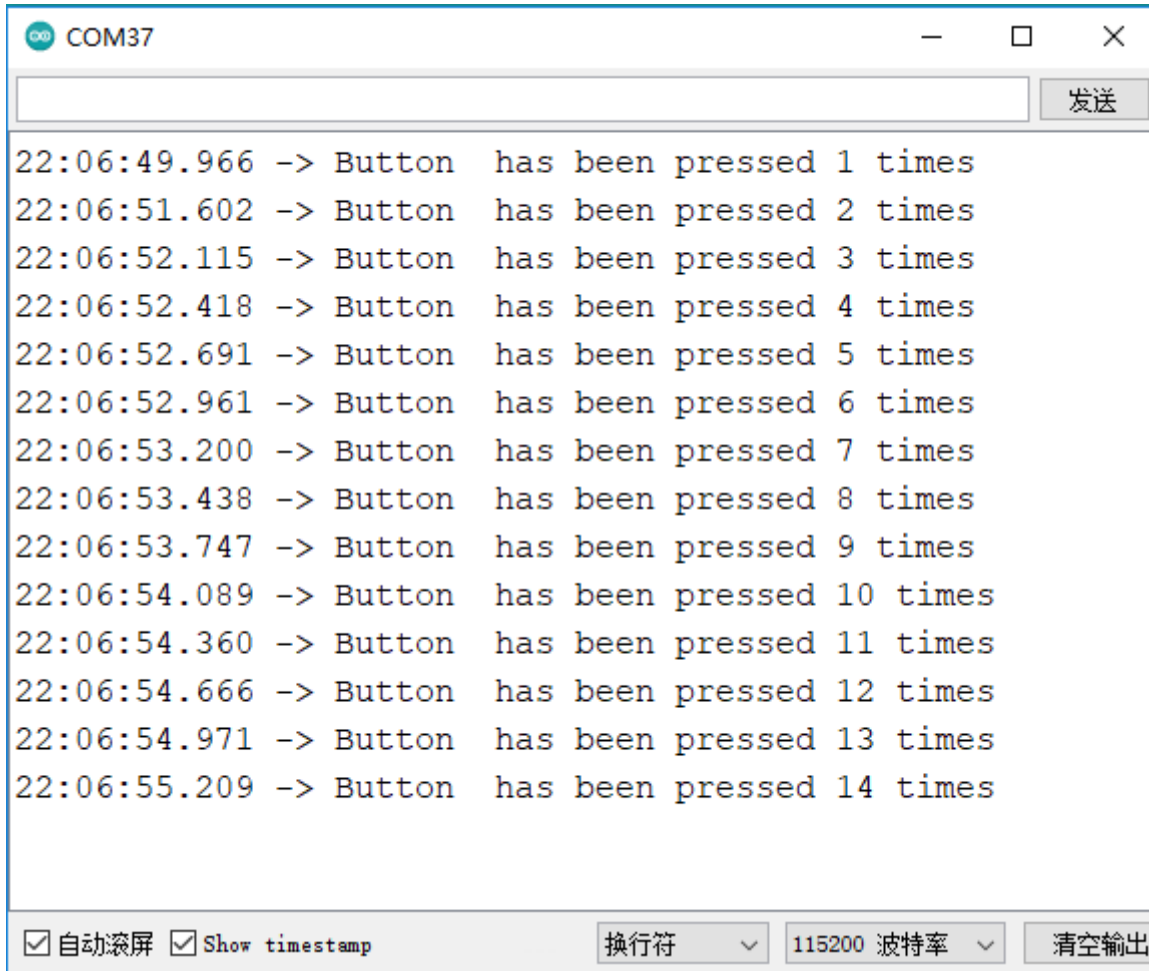
    button.pressed = true;
}

void setup() {
    Serial.begin(115200);
    pinMode(button.PIN, INPUT_PULLUP);
    attachInterrupt(button.PIN, isr, FALLING); //Register the interrupt function, and set the trigger mode to FALLING
}

void loop() {
    if (button.pressed) {
        button.pressed = false;
        delay(50);
        if(digitalRead(button.PIN)==LOW){
            button.numberKeyPresses += 1;
            Serial.printf("Button has been pressed %u times\n", button.numberKeyPresses); //Press the button, and print the
        }
    }
}
```

Result:

When the program is successfully uploaded, press the on-board button once, the serial port will print a message. The number accumulates every time you press the button, as shown in the figure below.



The screenshot shows a serial terminal window titled 'COM37'. The window contains a list of 14 lines of text, each representing a button press event. The text is as follows:

```
22:06:49.966 -> Button has been pressed 1 times
22:06:51.602 -> Button has been pressed 2 times
22:06:52.115 -> Button has been pressed 3 times
22:06:52.418 -> Button has been pressed 4 times
22:06:52.691 -> Button has been pressed 5 times
22:06:52.961 -> Button has been pressed 6 times
22:06:53.200 -> Button has been pressed 7 times
22:06:53.438 -> Button has been pressed 8 times
22:06:53.747 -> Button has been pressed 9 times
22:06:54.089 -> Button has been pressed 10 times
22:06:54.360 -> Button has been pressed 11 times
22:06:54.666 -> Button has been pressed 12 times
22:06:54.971 -> Button has been pressed 13 times
22:06:55.209 -> Button has been pressed 14 times
```

At the bottom of the window, there are several controls: a checked checkbox for '自动滚屏' (Auto scroll), a checked checkbox for 'Show timestamp', a dropdown menu for '换行符' (Line feed) set to '\n', a dropdown menu for '波特率' (Baud rate) set to '115200', and a '清空输出' (Clear output) button.

9.7 I2C Communication

Function Description:

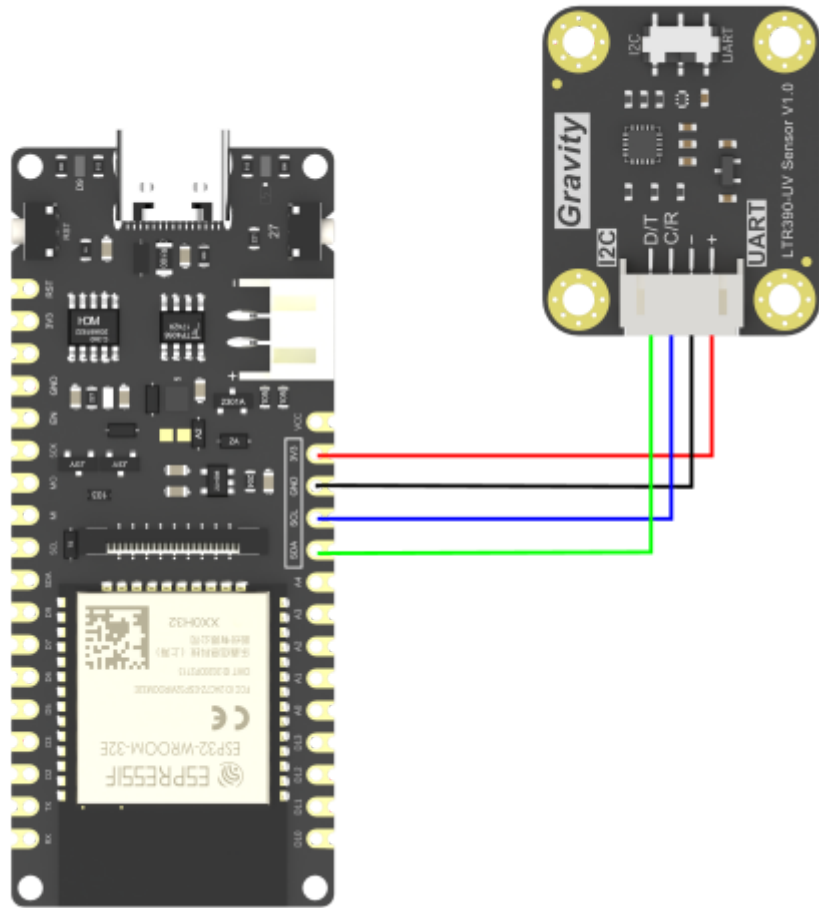
The FireBeetle 2 ESP32-E's I2C can be configured to any I/O port by passing relevant parameters. For ease of use, we have already provided default configuration for I2C which is fully compatible with Arduino. The default configured pins can be seen in the pinout diagram.

In this section, we will obtain the ambient light intensity by driving LTR390-UV light sensor based on the default I2C configuration. About how to use the LTR390-UV light sensor, please refer to the LTR390-UV Ultraviolet Light Sensor Wiki (https://wiki.dfrobot.com/SKU_SEN0540_Gravity_LTR390_UV_Sensor).

Preparation:

- FireBeetle 2 ESP32-E (<https://www.dfrobot.com/product-2195.html>) (SKU: DFR0654) ×1
- LTR390 UV Module (<https://www.dfrobot.com/product-2650.html>) (SKU: SEN0540) ×1

Connection Diagram:



Sample Code:

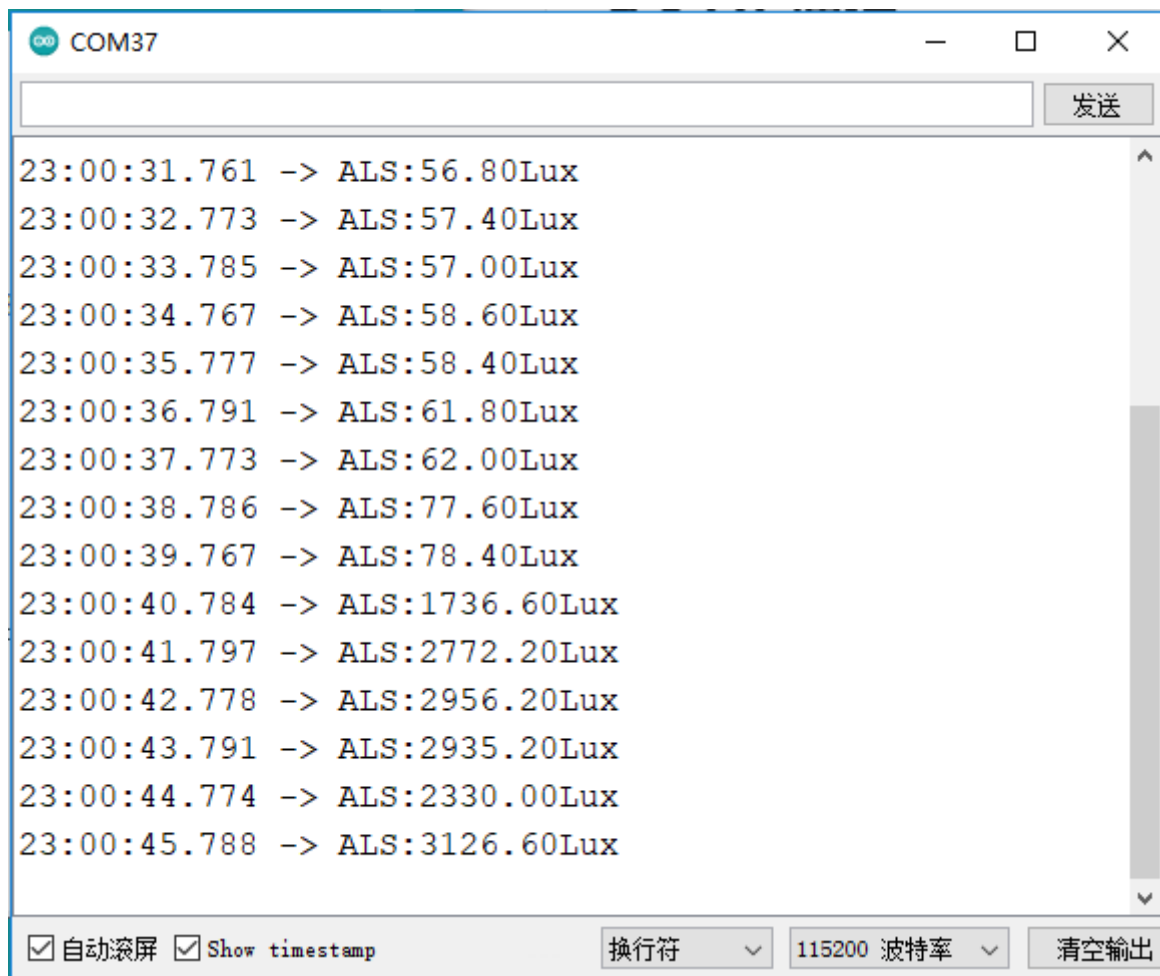
Program function: Read the light intensity in the current environment.

```
#include "DFRobot_LTR390UV.h"
DFRobot_LTR390UV ltr390(/*addr = */LTR390UV_DEVICE_ADDR, /*pWire = */&Wire);

void setup()
{
  Serial.begin(115200);
  while(ltr390.begin() != 0){
    Serial.println(" Sensor initialize failed!!");
    delay(1000);
  }
  Serial.println(" Sensor initialize success!!");
  ltr390.setALSOrUVSMeasRate(ltr390.e18bit,ltr390.e100ms);//18-bit data, sampling rate 100ms
  ltr390.setALSOrUVSGain(ltr390.eGain3);//3x Gain
  ltr390.setMode(ltr390.eALSMode);//Set ambient light mode
}
void loop()
{
  float als = 0;
  als = ltr390.readALSTransformData();//Read the converted data of ambient light, can only be used in ambient light mode
  Serial.print("ALS:");
  Serial.print(als);
  Serial.println("Lux");
  delay(1000);
}
```

Result:

When the program is uploaded, open serial monitor and you can see that the UV sensor is constantly collecting the current light intensity value.



```
COM37  
发送  
23:00:31.761 -> ALS:56.80Lux  
23:00:32.773 -> ALS:57.40Lux  
23:00:33.785 -> ALS:57.00Lux  
23:00:34.767 -> ALS:58.60Lux  
23:00:35.777 -> ALS:58.40Lux  
23:00:36.791 -> ALS:61.80Lux  
23:00:37.773 -> ALS:62.00Lux  
23:00:38.786 -> ALS:77.60Lux  
23:00:39.767 -> ALS:78.40Lux  
23:00:40.784 -> ALS:1736.60Lux  
23:00:41.797 -> ALS:2772.20Lux  
23:00:42.778 -> ALS:2956.20Lux  
23:00:43.791 -> ALS:2935.20Lux  
23:00:44.774 -> ALS:2330.00Lux  
23:00:45.788 -> ALS:3126.60Lux  
自动滚屏 Show timestamp 换行符 115200 波特率 清空输出
```

9.8 SPI Communication

Function Description:

SPI communication is used in many sensors because it offers a faster speed compared to I2C and doesn't have the issue of address

SPI communication is used in many sensors because it offers a faster speed compared to I2C and doesn't have the issue of address conflicts. SPI is a high-speed, full-duplex, synchronous communication bus. The SPI of FireBeetle 2 ESP32-E can be configured to all I/O ports, and you can refer to the underlying code for usage (not recommended for beginners). To provide a better user experience, FireBeetle 2 ESP32-E is configured by default with IO18 (SCK), IO19 (MISO), and IO23 (MOSI) as the SPI interface, which works well with Arduino.

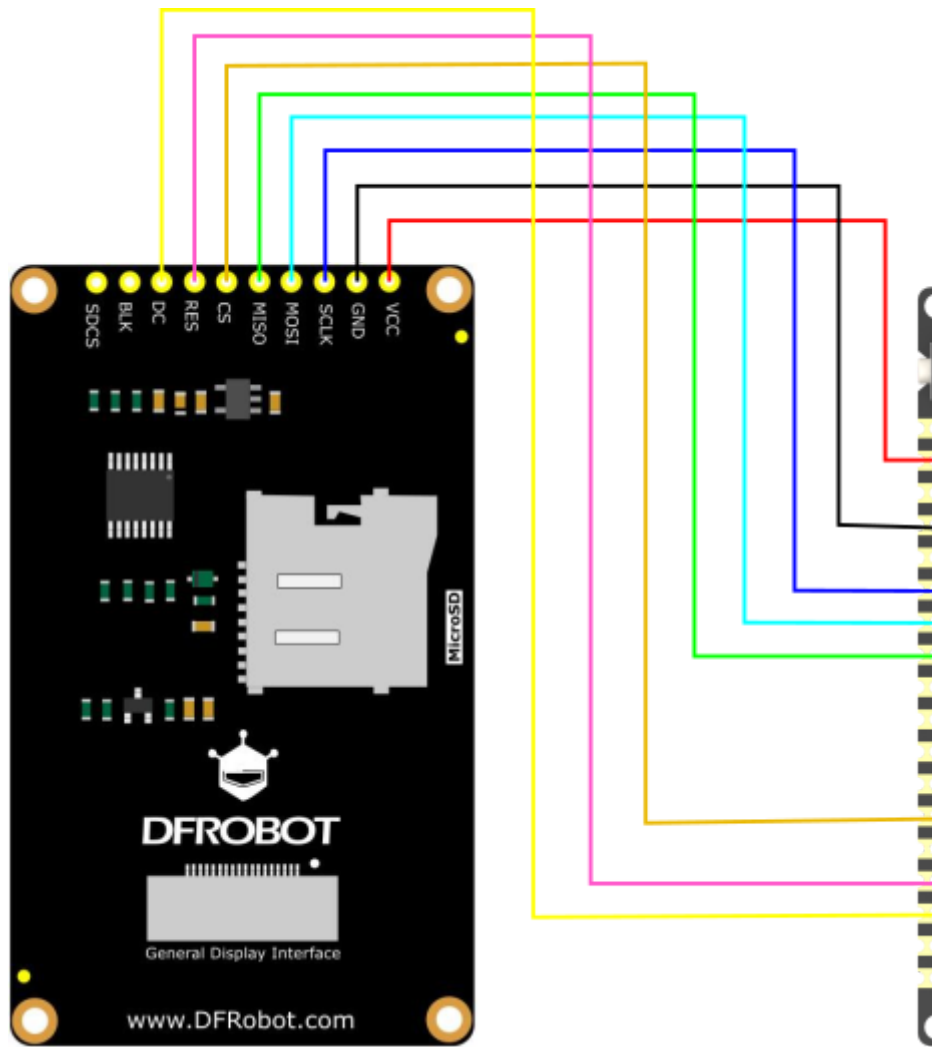
Hardware Preparation:

- FireBeetle 2 ESP32-E (<https://www.dfrobot.com/product-2195.html>) (SKU: DFR0654) ×1
- Gravity: I2C OLED-128x64 Display (<https://www.dfrobot.com/product-1576.html>) (SKU: DFR0486) ×1

Software Preparation:

Download GDL library: https://github.com/cdjg/DFRobot_GDL (https://github.com/cdjg/DFRobot_GDL)

Connection Diagram:



Pin Connections:

FireBeetle 2 ESP32-E	Display
----------------------	---------

3V3	VCC
GND	GND

FireBeetle 2 ESP32-E	Display
SCK	SCLK
MO	MOSI
MI	MISO
D6	CS
D3	RES
D2	DC

Sample Code:

Function program: Drive the display via SPI to show text and numbers.

```
#include "DFRobot_GDL.h"
#define TFT_DC  D2
#define TFT_CS  D6
#define TFT_RST D3
#define TFT_BL  D13

DFRobot_ST7789_240x320_HW_SPI screen(/*dc=*/TFT_DC,/*cs=*/TFT_CS,/*rst=*/TFT_RST);

void setup() {
  Serial.begin(115200);    //Init
  screen.begin();
  screen.setTextSize(2);  //Text size is 4, range is 1-4
  screen.fillScreen(COLOR_RGB565_BLACK);  //Screen background color
  screen.setFont(&FreeMono12pt7b);        //Font format
  screen.setCursor(/*x=*/10,/*y=*/120);    //Starting point of text
  screen.setTextColor(COLOR_RGB565_LGRAY); //Color of text
  screen.print("DFRobot");    //Output text content
  screen.setCursor(10,200);
  screen.setTextColor(COLOR_RGB565_GREEN);
  screen.setTextWrap(true);
  screen.print("20220828");
}

void loop() {
}
```

Result:

The screen displays the text "DFRobot" in white and the number "20220828" in green.

FPC PINS	Beetle ESP32 C3 Pins	Description
VCC	3V3	3.3V
BLK (PWM dimming)	10	Backlight

FPC PINS	Beetle ESP32 C3 Pins	Description
GND	GND	GND
SCLK	4/SCK	SPI clock
MOSI	6/MOSI	Host output, slave input
MISO	5/MISO	Host input, slave output
DC	1	Data/command
RES	2	Reset
CS	7	TFT Chip Select
SDCS	0	SD card chip select
FCS	NC	Font library
TCS	3	Touch
SCL	22/SCL	I2C clock
SDA	21/SDA	I2C data
INT	NC	INT
BUSY-TE	NC	Anti-tear pins
X1	NC	custom pin 1
X2	NC	custom pin 2

When using FPC to connect the screen, please configure the corresponding pin numbers according to the GDL demo. Normally, only three pins need to be configured on different main controllers. The screen configuration is shown below:

```
/*ESP32 and ESP8266*/  
#elif defined(ESP32) || defined(ESP8266)  
#define TFT_DC D2  
#define TFT_CS D6  
#define TFT_RST D3  
#define TFT_BL D13
```

Please refer to the GDL display screen wiki

(https://wiki.dfrobot.com/2.0_Inches_320_240_IPS_TFT_LCD_Display_with_MicroSD_Card_Breakout_SKU_DFR0664) for specific usage instructions.

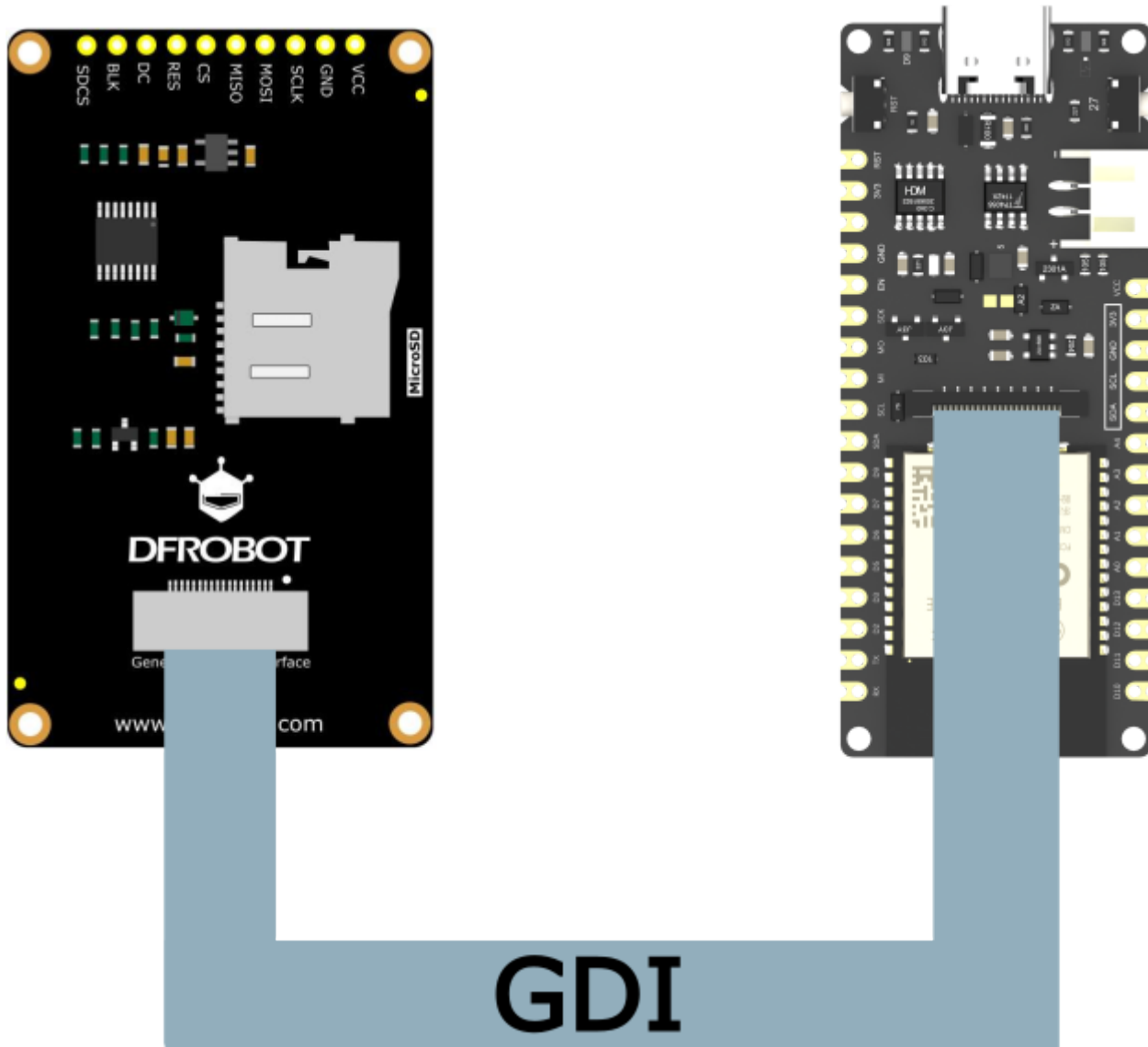
Displays that support GDI:

- 1.54" 240x240 IPS wide viewing angle TFT display (<https://www.dfrobot.com/product-2072.html>)
- 1.8" 128x160 IPS TFT LCD Display (<https://www.dfrobot.com/product-2580.html>)
- 2.0" 320x240 IPS wide viewing angle TFT display (<https://www.dfrobot.com/product-2071.html>)
- 2.8" 320x240 IPS TFT resistive touch display (<https://www.dfrobot.com/product-2106.html>)
- 3.5" 480x320 IPS TFT capacitive touch display (<https://www.dfrobot.com/product-2107.html>)
- 1.51" OLED Transparent Display with Converter (<https://www.dfrobot.com/product-2521.html>)

Hardware Preparation:

- FireBeetle 2 ESP32-E (<https://www.dfrobot.com/product-2195.html>) (SKU: DFR0654) ×1
- Gravity: I2C OLED-128x64 Display (<https://www.dfrobot.com/product-1576.html>) (SKU: DFR0486) ×1

Connection Diagram:



Sample Code:

https://wiki.dfrobot.com/FireBeetle_Board_ESP32_E_SKU_DFR0654

Function program: Show "DFRobot" and "20220828" on the display.

```
#include "DFRobot_GDL.h"
#define TFT_DC  D2
#define TFT_CS  D6
#define TFT_RST D3
#define TFT_BL  D13

DFRobot_ST7789_240x320_HW_SPI screen(/*dc=*/TFT_DC,/*cs=*/TFT_CS,/*rst=*/TFT_RST);

void setup() {
  Serial.begin(115200);    //Init
  screen.begin();
  screen.setTextSize(2); //Text size 4, range 1 to 4
  screen.fillScreen(COLOR_RGB565_BLACK); //Screen background color
  screen.setFont(&FreeMono12pt7b); // Font
  screen.setCursor(/*x=*/10,/*y=*/120); // Starting point of the text
  screen.setTextColor(COLOR_RGB565_LGRAY); //Text color
  screen.print("DFRobot"); //Output text content
  screen.setCursor(10,200);
  screen.setTextColor(COLOR_RGB565_GREEN);
  screen.setTextWrap(true);
  screen.print("20220828");
}

void loop() {
}
```

Result

The display shows "DFRobot" and "20220828" when all ready.

10. Advanced Tutorial for Arduino

10.1 Deep-sleep Mode

Function Description:

This section describes how to put the FireBeetle 2 ESP32-E into low power Deep_sleep mode through Arduino code.

The ESP32 deep_sleep mode wake-up methods are as follows:

- Timer wake-up
- Two-pin wake-up method
- Touch button wake-up
- ULP wake-up
- Return: Wake-up reason code

Reason Code	Reason	Description
2	ESP_SLEEP_WAKEUP_EXT0	Wake-up by RTC_GPIO
3	ESP_SLEEP_WAKEUP_EXT1	Wake-up by change in RTC_CNTL pin set
4	ESP_SLEEP_WAKEUP_TIMER	Wake-up by ESP timer
5	ESP_SLEEP_WAKEUP_TOUCHPAD	Wake-up by touch
6	ESP_SLEEP_WAKEUP_ULP	Wake-up by ULP (Ultra Low Power)

Hardware Preparation:

- FireBeetle 2 ESP32-E (<https://www.dfrobot.com/product-2195.html>) (SKU: DFR0654) x 1

Sample Code

Function: Set FireBeetle 2 ESP32-E to enter deep sleep mode, with timer as wake up source, and wake up every 5 seconds.

```
#define uS_TO_S_FACTOR 1000000ULL // Conversion factor from microseconds to seconds
#define TIME_TO_SLEEP 5 // Time for ESP32-E to enter deep sleep
RTC_DATA_ATTR int bootCount = 0;

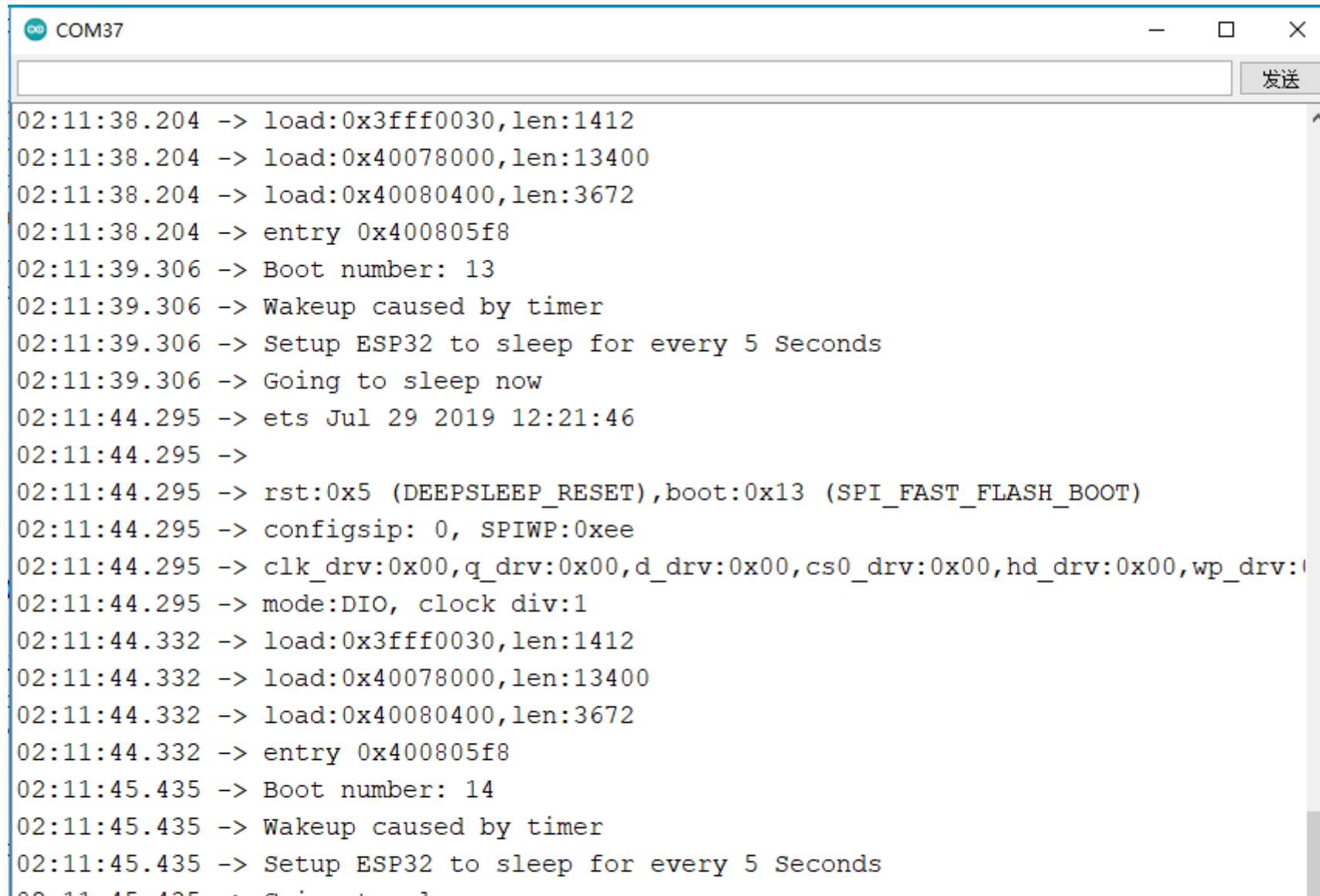
void print_wakeup_reason(){
    esp_sleep_wakeup_cause_t wakeup_reason;
    wakeup_reason = esp_sleep_get_wakeup_cause();

    switch(wakeup_reason) { // Check the wake-up reason and print the corresponding message
        case ESP_SLEEP_WAKEUP_EXT0 : Serial.println("Wakeup caused by external signal using RTC_IO"); break;
        case ESP_SLEEP_WAKEUP_EXT1 : Serial.println("Wakeup caused by external signal using RTC_CNTL"); break;
        case ESP_SLEEP_WAKEUP_TIMER : Serial.println("Wakeup caused by timer"); break;
        case ESP_SLEEP_WAKEUP_TOUCHPAD : Serial.println("Wakeup caused by touchpad"); break;
        case ESP_SLEEP_WAKEUP_ULP : Serial.println("Wakeup caused by ULP program"); break;
        default : Serial.printf("Wakeup was not caused by deep sleep: %d\n",wakeup_reason); break;
    }
}

void setup(){
    Serial.begin(115200);
    delay(1000);
    ++bootCount;
    Serial.println("Boot number: " + String(bootCount));
    print_wakeup_reason(); // Print the wake-up reason
    esp_sleep_enable_timer_wakeup(TIME_TO_SLEEP * uS_TO_S_FACTOR);
    Serial.println("Setup ESP32 to sleep for every " + String(TIME_TO_SLEEP) + " Seconds");
    Serial.println("Going to sleep now"); // We have set the wake up reason. Now we can start go to sleep of the peripheral.
    Serial.flush();
    esp_deep_sleep_start();
    Serial.println("This will never be printed");
}
```

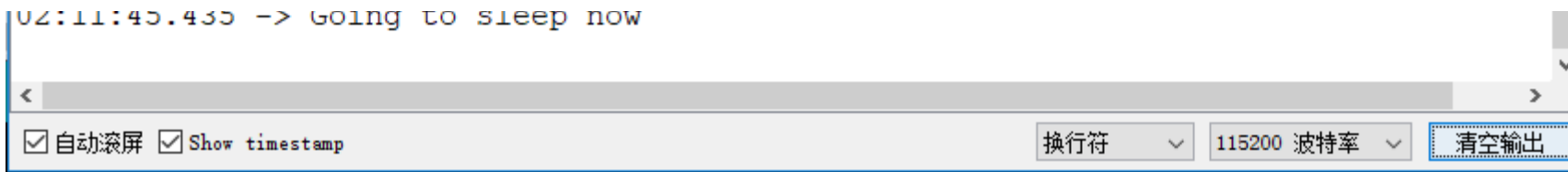
```
r  
void loop(){  
}
```

Result



```
COM37  
02:11:38.204 -> load:0x3fff0030,len:1412  
02:11:38.204 -> load:0x40078000,len:13400  
02:11:38.204 -> load:0x40080400,len:3672  
02:11:38.204 -> entry 0x400805f8  
02:11:39.306 -> Boot number: 13  
02:11:39.306 -> Wakeup caused by timer  
02:11:39.306 -> Setup ESP32 to sleep for every 5 Seconds  
02:11:39.306 -> Going to sleep now  
02:11:44.295 -> ets Jul 29 2019 12:21:46  
02:11:44.295 ->  
02:11:44.295 -> rst:0x5 (DEEPSLEEP_RESET),boot:0x13 (SPI_FAST_FLASH_BOOT)  
02:11:44.295 -> configsip: 0, SPIWP:0xee  
02:11:44.295 -> clk_drv:0x00,q_drv:0x00,d_drv:0x00,cs0_drv:0x00,hd_drv:0x00,wp_drv:  
02:11:44.295 -> mode:DIO, clock div:1  
02:11:44.332 -> load:0x3fff0030,len:1412  
02:11:44.332 -> load:0x40078000,len:13400  
02:11:44.332 -> load:0x40080400,len:3672  
02:11:44.332 -> entry 0x400805f8  
02:11:45.435 -> Boot number: 14  
02:11:45.435 -> Wakeup caused by timer  
02:11:45.435 -> Setup ESP32 to sleep for every 5 Seconds  
02:11:45.435 -> Going to sleep now
```

```
02:11:45.435 -> Going to sleep now
```



10.2 How to Use the SD Library

10.2.1 SD Class

The SD class provides the functionality to access the SD card and manipulate files and folders. Its member functions are as follows:

- **begin()**

Description: Initializes the SD card library and SD card.

Syntax: SD.begin() and begin(cspin)

When using SD.begin(), the default is to connect the SS pin of the Arduino SPI to the CS enable selection pin of the SD card. You can also use begin(cspin) to specify a pin to connect to the CS enable selection pin of the SD card, but you need to ensure that the SS pin of the SPI is in output mode, otherwise the SD card library will not work.

Parameters:

cspin, the Arduino pin connected to the SD card CS pin.

Return value: a boolean value, true if initialization is successful, false if initialization fails.

- **exists()**

Description: Checks if a file or folder exists on the SD card.

Syntax: SD.exists(filename)

Parameters:

filename, the name of the file to check. It can include a path, separated by "/".

Return value: a boolean value, true if the file or folder exists, false if it does not exist.

- **open()**

Description: Opens a file on the SD card. If the file does not exist and is opened for writing, the Arduino will create a file with the specified filename. (The path must exist in advance)

Syntax: SD.open(filename) and SD.open(filename, mode)

Parameters:

filename, the name of the file to open. It can include a path, separated by "/".

mode (optional), the mode to open the file. The default is to open in read-only mode. You can also open the file in the following two modes:

- FILE_READ: open the file in read-only mode;
- FILE_WRITE: open the file in write mode.

Return value: returns an object corresponding to the opened file. If the file cannot be opened, false is returned.

- **remove()**

Description: Removes a file from the SD card. If the file does not exist, the function returns an uncertain value. Therefore, it is best to use SD.exists(filename) to check if the file exists before removing the file.

Syntax: SD.remove(filename)

Parameters:

filename, the name of the file to remove. It can include a path, separated by "/".

Return value: a boolean value, true if the file is successfully removed, false if the file removal fails.

- **mkdir()**

Description: Creates a folder.

Parameters:

filename, the name of the folder to create. It can include a path, separated by "/".

Return value: a boolean value, true if the creation is successful, false if the creation fails.

- **rmdir()**

Description: Removes a folder. The folder to be removed must be empty.

Syntax: SD.rmdir(filename)

Parameters:

filename, the name of the folder to remove. It can include a path, separated by "/".

Return value: a boolean value, true if the removal is successful, false if the removal fails.

10.2.2 File Class

The File class provides the functionality to read/write files, and its member functions are similar to the serial-related functions previously used. The member functions are as follows:

- **available()**

Description: Checks the number of bytes of readable data in the current file.

Syntax: file.available()

Parameters:

file, an object of the File type.

Return value: the number of available bytes.

- **close()**

Description: Closes the file and ensures that the data has been completely written to the SD card.

Syntax: file.close()

Parameters:

file, an object of the File type.

Return value: none.

- **flush()**

Description: Ensure that data has been written to the SD card. When the file is closed, flush() will run automatically.

Syntax: file.flush()

Parameters:

file, an object of type File. Return value: None.

- **peek()**

Description: Read the current byte but does not move to the next byte.

Parameters:

file, an object of type File.

Return value: The next byte or character. Returns -1 if there is no readable data.

- **position()**

Description: Get the position in the current file (i.e. the position of the next byte to be read/written).

Syntax: file.position()

Parameters:

file, an object of type File.

Return value: The position in the current file.

- **print()**

Description: Output data to a file. The file to be written to should already be open and waiting to be written to.

Syntax: file.print(data) or file.print(data, BASE)

Parameters:

- file, an object of type File. data, the data to be written (can be of type char, byte, int, long, or String).
- BASE (optional), specifies the output format of the data: BIN (binary); OCT (octal); DEC (decimal); HEX (hexadecimal).
- Return value: The number of bytes sent.

- **println()**

Description: Output data to a file and add a newline.

Syntax: file.println(data) or file.println(data, BASE)

Parameters:

- file. an object of type File.

- data, the data to be written (can be of type char, byte, int, long, or String). BASE (optional), specifies the output format of the data: BIN (binary); OCT (octal); DEC (decimal); HEX (hexadecimal).

- Return value: The number of bytes sent.

- **seek()**

Description: Seek to a specified position. The position must be between 0 and the size of the file.

Syntax: file.seek(pos)

Parameters:

- file, an object of type File.

- pos, the position to seek to.

- Return value: A boolean value. True if the seek was successful, false if it failed.

- **size()**

Description: Get the size of the file.

Syntax: file.size()

Parameters:

- file, an object of type File.

- Return value: The size of the file in bytes.

- **read()**

Description: Read 1 byte of data.

Syntax: file.read()

Parameters:

- file, an object of type File.
- Return value: The next byte or character. Returns -1 if there is no readable data.

- **write()**

Description: Write data to a file.

Syntax: file.write(data) and file.write(buf, len)

Parameters:

- file, an object of type File.
- data: the data to be written, can be of type byte, char, or string (char*). buf: a character array or byte data.
- len: the number of elements in the buf array.
- Return value: The number of bytes sent.

- **isDirectory()**

Description: Determine whether the current file is a directory.

Syntax: file.isDirectory()

Parameters:

- file, an object of type File.
- Return value: A boolean value. True if the file is a directory. false if it is not.

- **openNextFile()**

Description: Open the next file.

Syntax: file.openNextFile()

Parameters:

- file, an object of type File.
- Return value: The object corresponding to the next file.

- **rewindDirectory()**

Description: Return to the first file in the current directory.

Syntax: file.rewindDirectory()

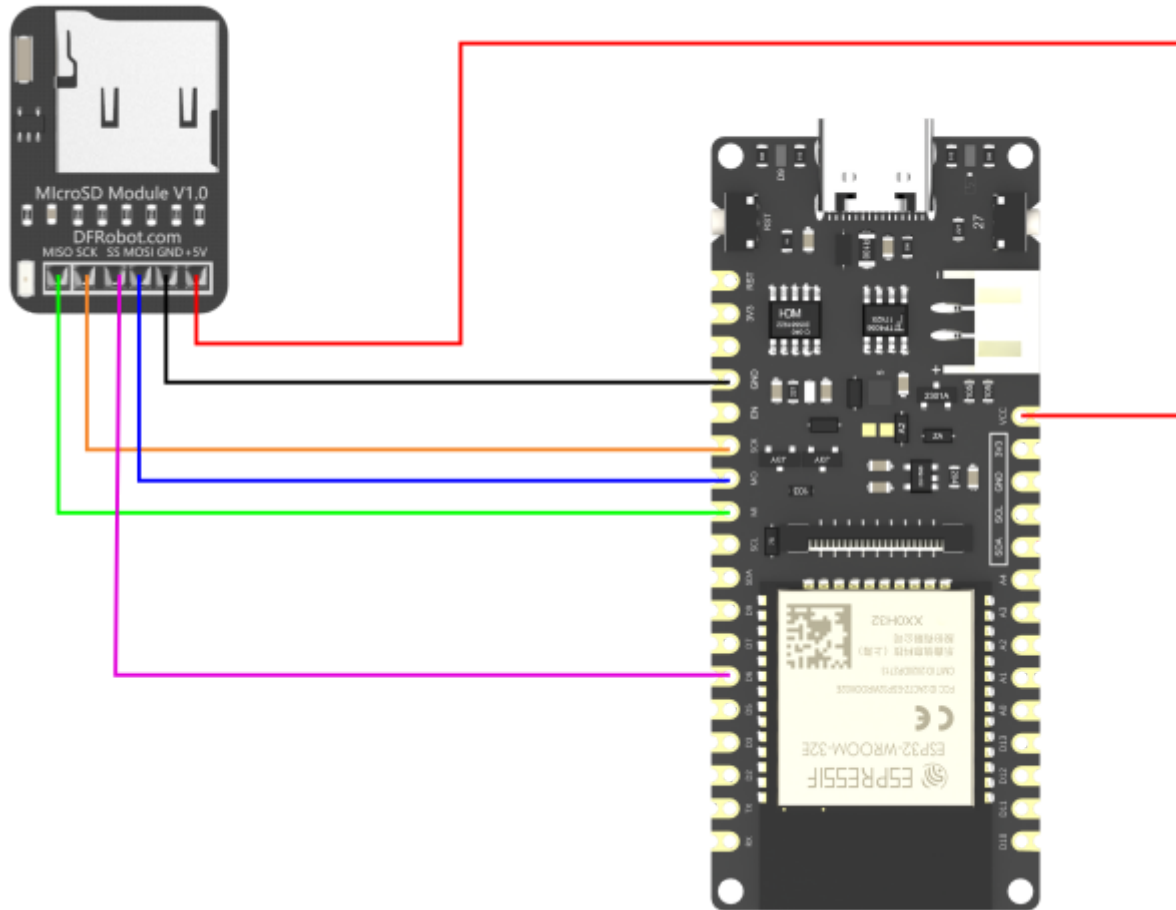
Parameters:

- file, an object of type File.
- Return value: None.

10.2.3 Preparation

- FireBeetle 2 ESP32-E (<https://www.dfrobot.com/product-2195.html>) (SKU: DFR0654) ×1
- MicroSD Card Module (<https://www.dfrobot.com/product-875.html>) (SKU: DFR0229) ×1

10.2.4 Connection Diagram



Pin Connections:

FireBeetle 2 ESP32-E	MicroSD Card Module
----------------------	---------------------

VCC	+5V
GND	GND

FireBeetle 2 ESP32-E	MicroSD Card Module
SCK	SCK
MO	MOSI
MI	MISO
D6	SS

10.2.5 Sample Code

The following program implements functions such as accessing SD card, manipulating files and folders, including reading and writing files.

```
/* Connect the SD card to the following pins:
 * SD Card | ESP32
 *   MISO   MISO
 *   SCK    SCK
 *   ss     D6
 *   MOSI   MOSI
 *   GND    GND
 *   +5V    VCC
 */
#include "FS.h"
#include "SD.h"
#include "SPI.h"

void listDir(fs::FS &fs, const char * dirname, uint8_t levels){
    Serial.printf("Listing directory: %s\n", dirname);

    File root = fs.open(dirname);
    if(!root){
        Serial.println("Failed to open directory");
        return;
    }
    if(!root.isDirectory()){
        Serial.println("Not a directory");
        return;
    }

    File file = root.openNextFile();
    while(file){
        if(file.isDirectory()){
            Serial.print("  DIR : ");
            Serial.println(file.name());
        }
    }
}
```

```
        Serial.println(file.name());
        if(levels){
            listDir(fs, file.path(), levels -1);
        }
    } else {

        Serial.print(" FILE: ");
        Serial.print(file.name());
        Serial.print(" SIZE: ");
        Serial.println(file.size());
    }
    file = root.openNextFile();
}
}

void createDir(fs::FS &fs, const char * path){
    Serial.printf("Creating Dir: %s\n", path);
    if(fs.mkdir(path)){
        Serial.println("Dir created");
    } else {
        Serial.println("mkdir failed");
    }
}

void removeDir(fs::FS &fs, const char * path){
    Serial.printf("Removing Dir: %s\n", path);
    if(fs.rmdir(path)){
        Serial.println("Dir removed");
    } else {
        Serial.println("rmdir failed");
    }
}

void readFile(fs::FS &fs, const char * path){
    Serial.printf("Reading file: %s\n", path);

    File file = fs.open(path);
    if(!file){
        Serial.println("File not found");
    }
    while(file.available()){
        Serial.write(file.read());
    }
}
```



```
    if(!file){
        Serial.println("Failed to open file for reading");
        return;
    }

    Serial.print("Read from file: ");
    while(file.available()){
        Serial.write(file.read());
    }
    file.close();
}

void writeFile(fs::FS &fs, const char * path, const char * message){
    Serial.printf("Writing file: %s\n", path);

    File file = fs.open(path, FILE_WRITE);
    if(!file){
        Serial.println("Failed to open file for writing");
        return;
    }
    if(file.print(message)){
        Serial.println("File written");
    } else {
        Serial.println("Write failed");
    }
    file.close();
}

void appendFile(fs::FS &fs, const char * path, const char * message){
    Serial.printf("Appending to file: %s\n", path);

    File file = fs.open(path, FILE_APPEND);
    if(!file){
        Serial.println("Failed to open file for appending");
        return;
    }
}
```

```
    if(!file.print(message)){
        Serial.println("Message appended");
    } else {
        Serial.println("Append failed");
    }
    file.close();
}

void renameFile(fs::FS &fs, const char * path1, const char * path2){
    Serial.printf("Renaming file %s to %s\n", path1, path2);
    if (fs.rename(path1, path2)) {
        Serial.println("File renamed");
    } else {
        Serial.println("Rename failed");
    }
}

void deleteFile(fs::FS &fs, const char * path){
    Serial.printf("Deleting file: %s\n", path);
    if(fs.remove(path)){
        Serial.println("File deleted");
    } else {
        Serial.println("Delete failed");
    }
}

void testFileIO(fs::FS &fs, const char * path){
    File file = fs.open(path);
    static uint8_t buf[512];
    size_t len = 0;
    uint32_t start = millis();
    uint32_t end = start;
    if(file){
        len = file.size();
        size_t flen = len;
        start = millis();
        while (flen > 0) {
            if (!file.read(buf, len))
                break;
            len = sizeof(buf);
            while (len > 0)
                len -= sizeof(buf);
            end = millis();
            flen -= len;
        }
    }
}
```

```
while(len){
    size_t toRead = len;
    if(toRead > 512){
        toRead = 512;
    }

    file.read(buf, toRead);
    len -= toRead;
}
end = millis() - start;
Serial.printf("%u bytes read for %u ms\n", flen, end);
file.close();
} else {
    Serial.println("Failed to open file for reading");
}

file = fs.open(path, FILE_WRITE);
if(!file){
    Serial.println("Failed to open file for writing");
    return;
}

size_t i;
start = millis();
for(i=0; i<2048; i++){
    file.write(buf, 512);
}
end = millis() - start;
Serial.printf("%u bytes written for %u ms\n", 2048 * 512, end);
file.close();
}

void setup(){
    Serial.begin(115200);
    if(!SD.begin()){
        Serial.println("Card Mount Failed");
    }
}
```

```
        return;
    }
    uint8_t cardType = SD.cardType();

    if(cardType == CARD_NONE){
        Serial.println("No SD card attached");
        return;
    }

    Serial.print("SD Card Type: ");
    if(cardType == CARD_MMC){
        Serial.println("MMC");
    } else if(cardType == CARD_SD){
        Serial.println("SDSC");
    } else if(cardType == CARD_SDHC){
        Serial.println("SDHC");
    } else {
        Serial.println("UNKNOWN");
    }

    uint64_t cardSize = SD.cardSize() / (1024 * 1024);
    Serial.printf("SD Card Size: %lluMB\n", cardSize);

    listDir(SD, "/", 0);
    createDir(SD, "/mydir");
    listDir(SD, "/", 0);
    removeDir(SD, "/mydir");
    listDir(SD, "/", 2);
    writeFile(SD, "/hello.txt", "Hello ");
    appendFile(SD, "/hello.txt", "World!\n");
    readFile(SD, "/hello.txt");
    deleteFile(SD, "/foo.txt");
    renameFile(SD, "/hello.txt", "/foo.txt");
    readFile(SD, "/foo.txt");
    testFileIO(SD, "/test.txt");
    Serial.printf("Total space: %lluMB\n", SD.totalBytes() / (1024 * 1024));
```

```
Serial.printf("Used space: %LuMB\n", SD.usedBytes() / (1024 * 1024));  
}  
  
void loop(){  
  
}
```

10.2.6 Result

When the program is successfully uploaded, open the serial monitor, you will a series of operations on the SD card, as well as reading its memory.

COM31

```
17:20:47.692 -> SD Card Type: SDHC
17:20:47.692 -> SD Card Size: 15193MB
17:20:47.692 -> Listing directory: /
17:20:47.692 ->   DIR : System Volume Information
17:20:47.692 ->   FILE: test.txt  SIZE: 1048576
17:20:47.692 ->   FILE: foo.txt  SIZE: 13
17:20:47.692 -> Creating Dir: /mydir
17:20:47.692 -> Dir created
17:20:47.692 -> Listing directory: /
17:20:47.692 ->   DIR : System Volume Information
17:20:47.740 ->   FILE: test.txt  SIZE: 1048576
17:20:47.740 ->   FILE: foo.txt  SIZE: 13
17:20:47.740 ->   DIR : mydir
17:20:47.740 -> Removing Dir: /mydir
17:20:47.740 -> Dir removed
17:20:47.740 -> Listing directory: /
17:20:47.740 ->   DIR : System Volume Information
17:20:47.740 -> Listing directory: /System Volume Information
17:20:47.740 ->   FILE: IndexerVolumeGuid  SIZE: 76
17:20:47.740 ->   FILE: WPSettings.dat  SIZE: 12
17:20:47.740 ->   FILE: test.txt  SIZE: 1048576
17:20:47.740 ->   FILE: foo.txt  SIZE: 13
17:20:47.740 -> Writing file: /hello.txt
17:20:47.740 -> File written
17:20:47.787 -> Appending to file: /hello.txt
17:20:47.787 -> Message appended
17:20:47.787 -> Reading file: /hello.txt
17:20:47.787 -> Read from file: Hello World!
17:20:47.787 -> Deleting file: /foo.txt
17:20:47.787 -> File deleted
17:20:47.787 -> Renaming file /hello.txt to /foo.txt
```

```
17:20:47.787 -> File renamed
17:20:47.787 -> Reading file: /foo.txt
17:20:47.787 -> Read from file: Hello World!
17:20:50.658 -> 1048576 bytes read for 2876 ms
17:20:55.684 -> 1048576 bytes written for 5005 ms
17:20:55.732 -> Total space: 15160MB
17:20:55.732 -> Used space: 1MB
```

10.3 WiFi

10.3.1 WiFi Basic Tutorial

Function:

The ESP32 has WiFi capability. The following example uses ESP32 to create a WiFi server. The client can connect to the server and remotely control the on/off state of the LED.

Hardware Preparation:

- FireBeetle 2 ESP32-E (<https://www.dfrobot.com/product-2195.html>) (SKU: DFR0654) ×1
- Mobile Phone ×1

Sample Code

Program Function: Connect to the WiFi server created by FireBeetle 2 ESP32-E using a mobile phone, access 192.168.4.1, and remotely control the on/off state of the onboard LED.

```
/*WiFiAccessPoint.ino creates a wifi hotspot and provides a web service
Steps:
1. Connect to the wifi "yourAp"
2. Access https://192.168.4.1/H to turn on the LED, or access https://192.168.4.1/L to turn off the LED*/

#include <WiFi.h>
#include <WiFiClient.h>
#include <WiFiAP.h>

// Set your wifi name and password
const char *ssid = "esp32";
const char *password = "123456789";

WiFiServer server(80);

void setup() {
  pinMode(LED_BUILTIN, OUTPUT); //Set the LED pin as output
  Serial.begin(115200);
  Serial.begin(115200);
  Serial.println();
  Serial.println("Configuring access point...");

  // Configure wifi and get IP address
  WiFi.softAP(ssid, password);
  IPAddress myIP = WiFi.softAPIP();
  Serial.print("AP IP address: ");
  Serial.println(myIP);
  server.begin();
  Serial.println("Server started");
}
```



```
void loop() {
  WiFiClient client = server.available(); // Listen to the server

  if (client) { // If there is message from the server

    Serial.println("New Client."); // Print the message on the serial port
    String currentLine = ""; // Create a String to save the incoming data from the client
    while (client.connected()) {
      char c = client.read();
      Serial.write(c);
      if (c == '\n') {
        if (currentLine.length() == 0) {
          client.println("HTTP/1.1 200 OK");
          client.println("Content-type:text/html");
          client.println();
          client.print("Click <a href=\"/H\">here</a> to turn ON the LED.<br>");
          client.print("Click <a href=\"/L\">here</a> to turn OFF the LED.<br>");
          client.println();
          break;
        } else {
          currentLine = "";
        }
      } else if (c != '\r') {
        currentLine += c;
      }
      if (currentLine.endsWith("GET /H")) {
        digitalWrite(LED_BUILTIN, HIGH); // GET /H turns on the LED
      }
      if (currentLine.endsWith("GET /L")) {
        digitalWrite(LED_BUILTIN, LOW); // GET /L turns off the LED
      }
    }
    client.stop();
    Serial.println("Client Disconnected.");
  }
}
```

Result

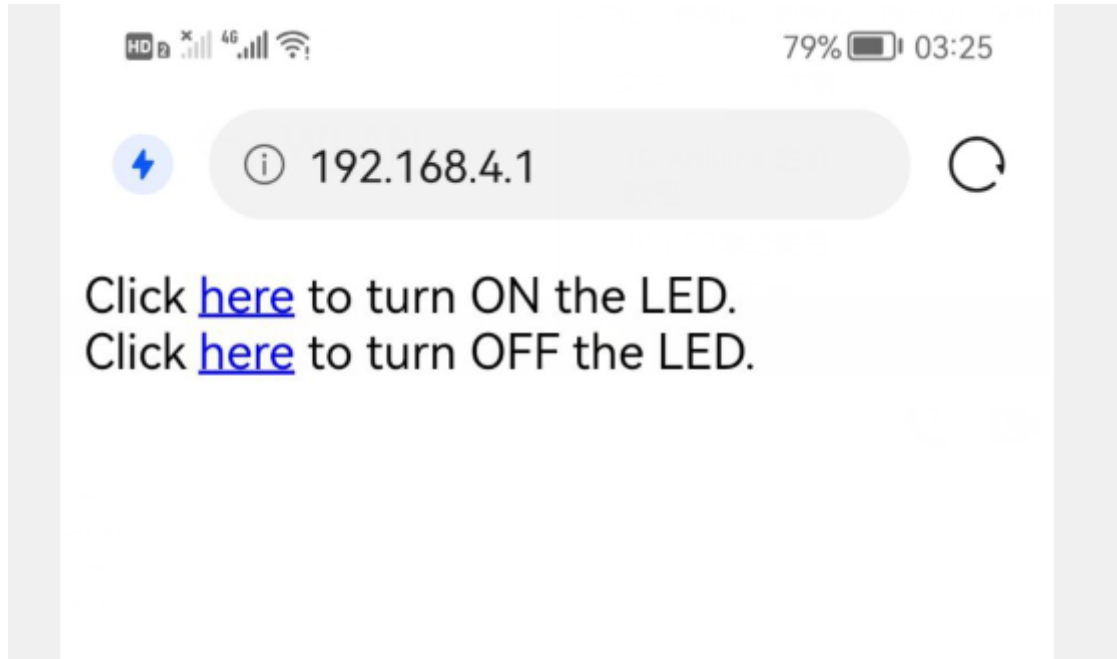
After connecting to the ESP32, accessing the given URL, and clicking **turn on the light**, the onboard LED light will turn on; clicking **turn off the light** will turn off the onboard LED light.

- Connect to ESP32



加密

- 192.168.4.1 Website interface



10.3.2 Getting Network Time via WiFi

Function

The ESP32 supports both STA and AP mode for WiFi connection.

- STA mode: The ESP32 module connects to the Internet through a router, and remote control of the device is achieved through the Internet via a mobile phone or computer.
- AP mode: The ESP32 module serves as a hotspot, allowing direct communication between a mobile phone or computer and the module, enabling wireless control within a local area network.
- STA+AP mode: The coexistence mode of the two modes, which can realize seamless switching between Internet control and local area network control

NETWORK CONTROL

The following example code defaults to STA mode.

Hardware Preparation:

- FireBeetle 2 ESP32-E (<https://www.dfrobot.com/product-2195.html>) (SKU: DFR0654) ×1

Sample Code

Function: Get and set time from Network Time Server.

```
#include <WiFi.h>

const char* ssid="WIFI_SSID"; //Fill in the WIFI name
const char* password="WIFI_PASSWORD"; //Fill in the WIFI password
const char* ntpServer = "pool.ntp.org"; //Get the time from the network time server
const long gmtOffset_sec = 28800; //UTC time is used here, China is in the UTC+8 time zone, which is 8*60*60
const int daylightOffset_sec = 0; //Use daylight saving time daylightOffset_sec = 3600, otherwise it is equal to 0

void printLocalTime(){
    struct tm timeinfo;
    if(!getLocalTime(&timeinfo)){ //If the local time is obtained, put it into the timeinfo structure
        Serial.println("Failed to obtain time");
        return ;
    }
    Serial.println(&timeinfo,"%A, %B %d %Y %H:%M:%S"); //Output the obtained time in this format
}

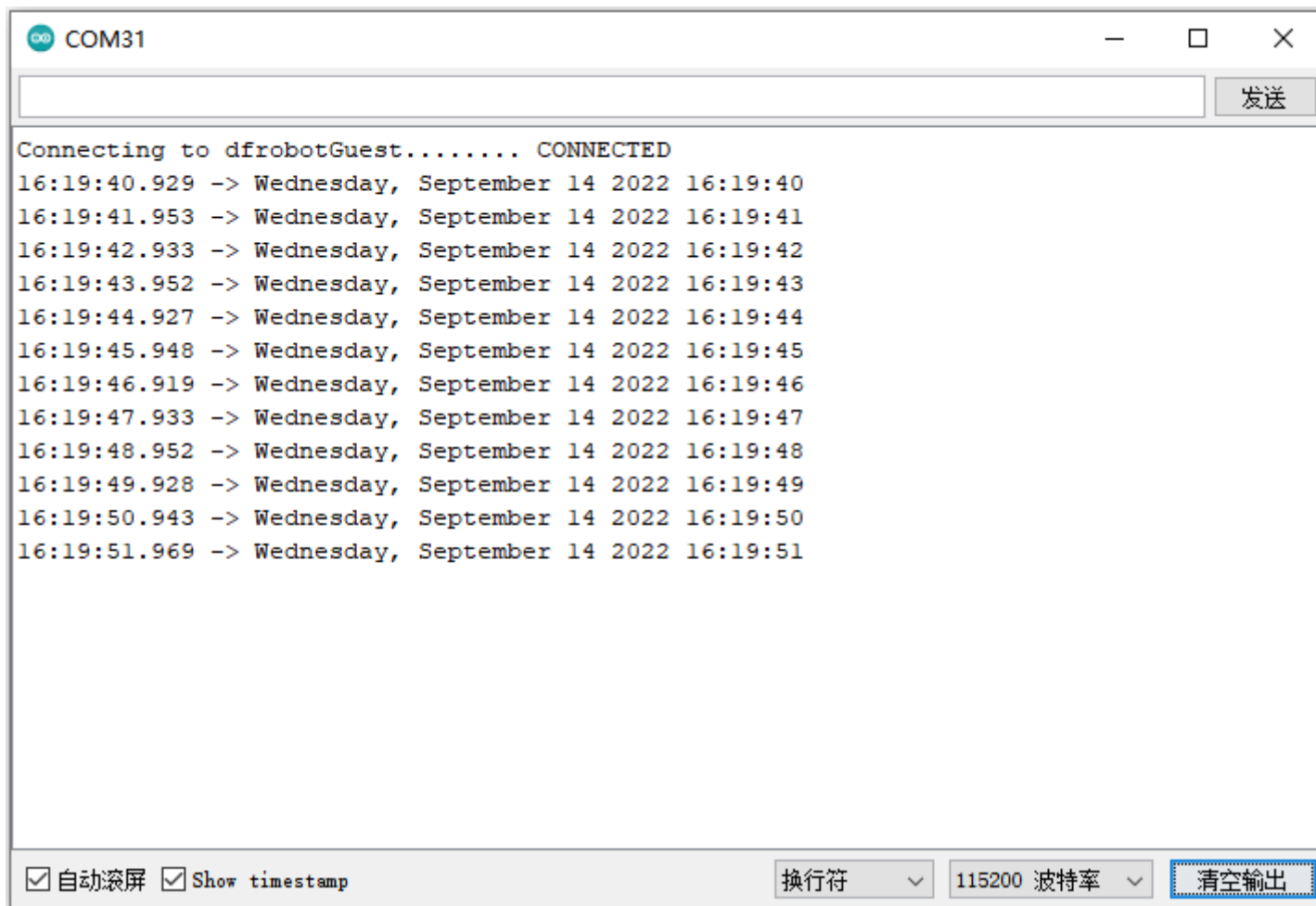
void setup() {
    Serial.begin(115200);
    Serial.printf("Connecting to %s",ssid);
    WiFi.begin(ssid,password); //Connect to WIFI
    while(WiFi.status()!=WL_CONNECTED){ //Wait for the connection to be successful
        delay(500);
        Serial.print(".");
    }
    Serial.println(" CONNECTED");
    configTime(gmtOffset_sec, daylightOffset_sec, ntpServer); //Configure network time as the system time of the ESP32-E main
}

void loop()
}
```

```
  \n    printLocalTime(); //Print local time\n    delay(1000);\n  }
```

Result

When the program is uploaded, you can see the obtained time as shown in the figure below.



10.4 Bluetooth

10.4.1 Bluetooth Basic Tutorial

Function

ESP32 has Bluetooth functionality, and this example will demonstrate the use of two FireBeetle 2 ESP32-E boards for Bluetooth communication. One ESP32 board will send data to the other ESP32 board, which is the most basic model for using Bluetooth wireless communication.

Hardware Preparation:

- FireBeetle 2 ESP32-E (<https://www.dfrobot.com/product-2195.html>) (SKU: DFR0654) ×1

Sample Code

Program: Use one FireBeetle 2 ESP32-E as the host and the other as the slave to establish a Bluetooth wireless communication, where the host sends data to the slave.

Note: The programs for the host and slave are created in separate windows, and should be compiled and uploaded separately. Otherwise, it will not be possible to open two serial ports.

Program for Slave:

```
#include "BluetoothSerial.h"

#if !defined(CONFIG_BT_ENABLED) || !defined(CONFIG_BLUEDROID_ENABLED)
#error Bluetooth is not enabled! Please run `make menuconfig` to and enable it
#endif

BluetoothSerial SerialBT;

void setup() {
  Serial.begin(115200);
  SerialBT.begin("ESP32_one");      //Bluetooth device name
  Serial.println("The device started, now you can pair it with bluetooth!");
}

void loop() {
  if (Serial.available()) {
    SerialBT.write(Serial.read());
  }
  if (SerialBT.available()) {
    Serial.write(SerialBT.read());
  }
  delay(20);
}
```

Program for host:


```
#include "BluetoothSerial.h"

BluetoothSerial SerialBT;

String MACadd = "AA:BB:CC:11:22:33";
uint8_t address[6] = {0xAA, 0xBB, 0xCC, 0x11, 0x22, 0x33};
String name = "ESP32_one";
const char *pin = "1234"; //<- Standard pin will be provided by default
bool connected;

void setup() {
  Serial.begin(115200);
  SerialBT.begin("ESP32test", true);
  Serial.println("The device started in master mode, make sure remote BT device is on!");

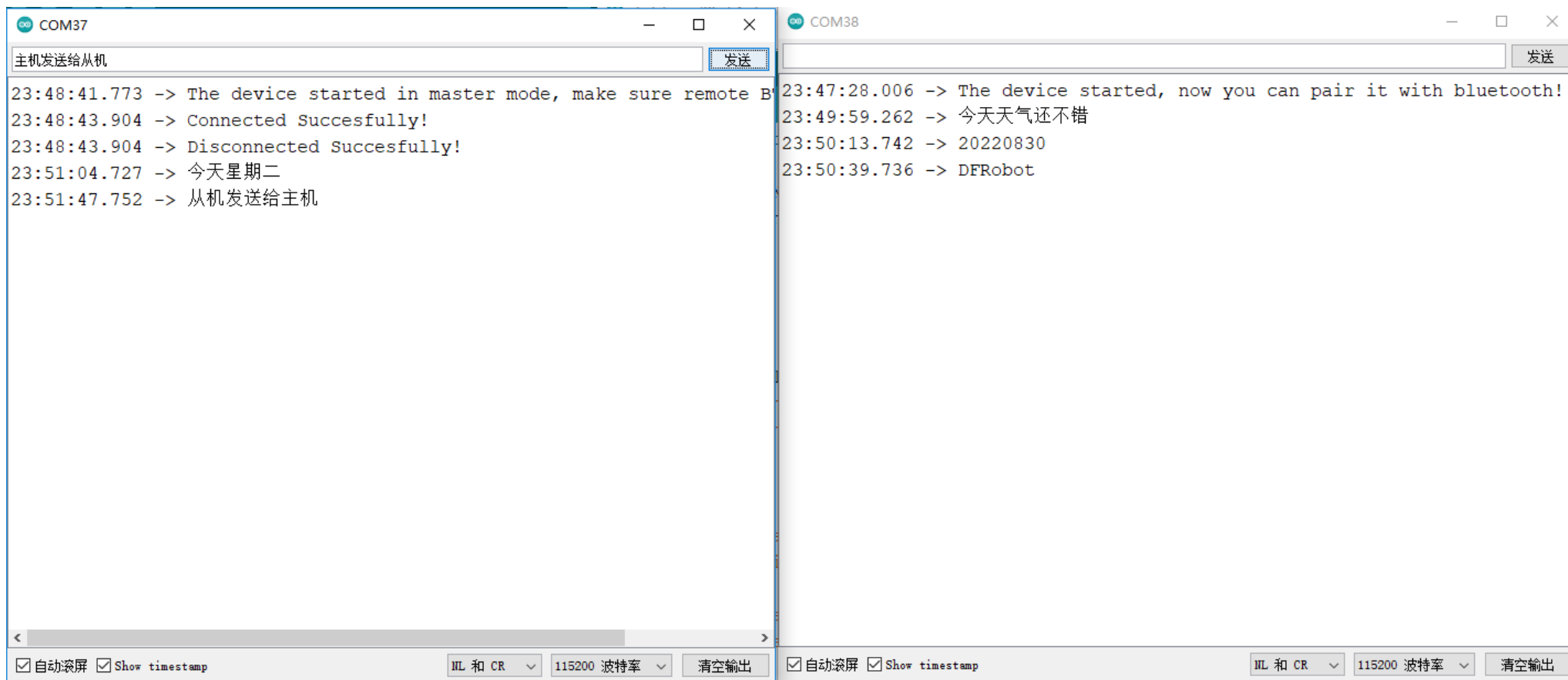
  // Connecting using the address provides a faster connection (up to 10 seconds), while connecting using the name can take
  // It first parses the name to an address, but it allows connecting to different devices with the same name.
  // Set CoreDebugLevel to Info to see the device Bluetooth address and device name
  connected = SerialBT.connect(name);
  if(connected) {
    Serial.println("Connected Successfully!");
  } else {
    while(!SerialBT.connected(10000)) {
      Serial.println("Failed to connect. Make sure remote device is available and in range, then restart app.");
    }
  }
  // disconnect() may take up to 10 seconds
  if (SerialBT.disconnect()) {
    Serial.println("Disconnected Successfully!");
  }
  // This will reconnect to the name (if parsed it will use the address) or the address (name/address) used with connect
```

```
// THIS WILL RECONNECT TO THE NAME (IF PARSED, IT WILL USE THE ADDRESS) OF THE ADDRESS (NAME/ADDRESS) USED WITH CONNECT.
SerialBT.connect();
}

void loop() {
  if (Serial.available()) {
    SerialBT.write(Serial.read());
  }
  if (SerialBT.available()) {
    Serial.write(SerialBT.read());
  }
  delay(20);
}
```

Result:

Open two serial monitor windows simultaneously, edit the message to be sent in the "Send" column of one of the serial monitor windows, and the message should be displayed in the other serial monitor window, which proves that the Bluetooth communication between the two FireBeetle 2 ESP32-E development boards is successful.



10.4.2 Bluetooth Server

BLEDevice

- **init()** Description: Create a BLE device
- **createServer()** Description: Create a BLE server

BLEServer

- **createService()** **Description:** Create a BLE service
- **setCallbacks()** **Description:** Create server callbacks
- **start()** **Description:** Start the server
- **getAdvertising()** **Description:** Configure advertising function

BLEService

- **createCharacteristic()** **Description:** Create characteristic of the service

BLECharacteristic

- **setCallbacks()** **Description:** Set callbacks for characteristic
- **addDescriptor()** **Description:**
- **setValue()** **Description:** Set the value of the characteristic
- **getValue()** **Description:** Get the value of the characteristic
- **notify()** **Description:** Send notification

BLEAdvertising

- **start()** **Description:** Start advertising

Hardware Preparation:

- FireBeetle 2 ESP32-E (<https://www.dfrobot.com/product-2195.html>) (SKU: DFR0654) ×1

Sample Code

Function: Establish a BLE server that can provide data and send notifications to clients. When the server receives data from a client, it sends the received data to the client as a notification.

```
#include <BLEDevice.h>
#include <BLEServer.h>
#include <BLEUtils.h>
#include <BLE2902.h>
#define SERVICE_UUID          "DFCD0001-36E1-4688-B7F5-EA07361B26A8"
#define CHARACTERISTIC1_UUID  "DFCD000A-36E1-4688-B7F5-EA07361B26A8"
bool deviceConnected = false;
BLEServer *pServer;
BLEService *pService;
BLECharacteristic* pCharacteristic;

class MyServerCallbacks: public BLEServerCallbacks {
    void onConnect(BLEServer* pServer) {
        deviceConnected = true;
    };

    void onDisconnect(BLEServer* pServer) {
        deviceConnected = false;
    }
};

class MyCallbacks: public BLECharacteristicCallbacks {
    void onWrite(BLECharacteristic *pCharacteristic) {
        std::string value = pCharacteristic->getValue();

        if (value.length() > 0) {
            Serial.println("*****");
            Serial.print("New value: ");
            for (int i = 0; i < value.length(); i++){
                Serial.print(value[i]);
            }
            Serial.println();
        }
    }
};
```

```
        Serial.println();
        Serial.println("*****");
        pCharacteristic->notify();
    }
}

};

void setupBLE()
{
    BLEDevice::init("DFRobot_ESP32"); //Create BLE device
    pServer = BLEDevice::createServer(); //Create BLE server
    pServer->setCallbacks(new MyServerCallbacks()); //Set the server's callback function
    pService = pServer->createService(SERVICE_UUID); //Create BLE service
    pCharacteristic = pService->createCharacteristic(
        CHARACTERISTIC1_UUID,
        BLECharacteristic::PROPERTY_READ|
        BLECharacteristic::PROPERTY_NOTIFY|
        BLECharacteristic::PROPERTY_WRITE); //Create characteristic for the server
    pCharacteristic->setCallbacks(new MyCallbacks()); //Set the callback function for the characteristic
    pCharacteristic->addDescriptor(new BLE2902());
    pCharacteristic->setValue("Hello DFRobot");
    pService->start();
    BLEAdvertising *pAdvertising = pServer->getAdvertising();
    pAdvertising->start();
}

void setup() {
    Serial.begin(115200);
    setupBLE();
}

void loop() {
    delay(3000);
}
```

Result:

Using FireBeetle 2 ESP32-E as a BLE server, the client can be a mobile phone. Install the Bluetooth helper LightBlue from the app store on

the phone and establish a BLE connection with the ESP32-E module. Here, we will demonstrate the operations provided by Light Blue on an iPhone. Similar Bluetooth software helpers are also available for Android phones.








Operation on Mobile client are as follows:


Step 1. Connect to DFRobot_ESP32.

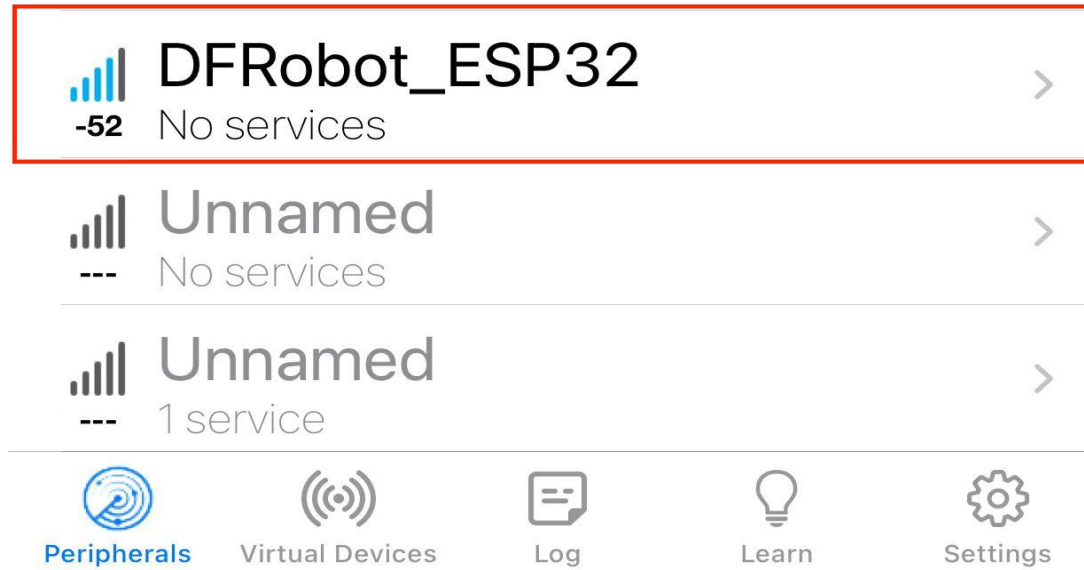
无SIM卡 上午 12:17 20%

Sort **LightBlue** Filter

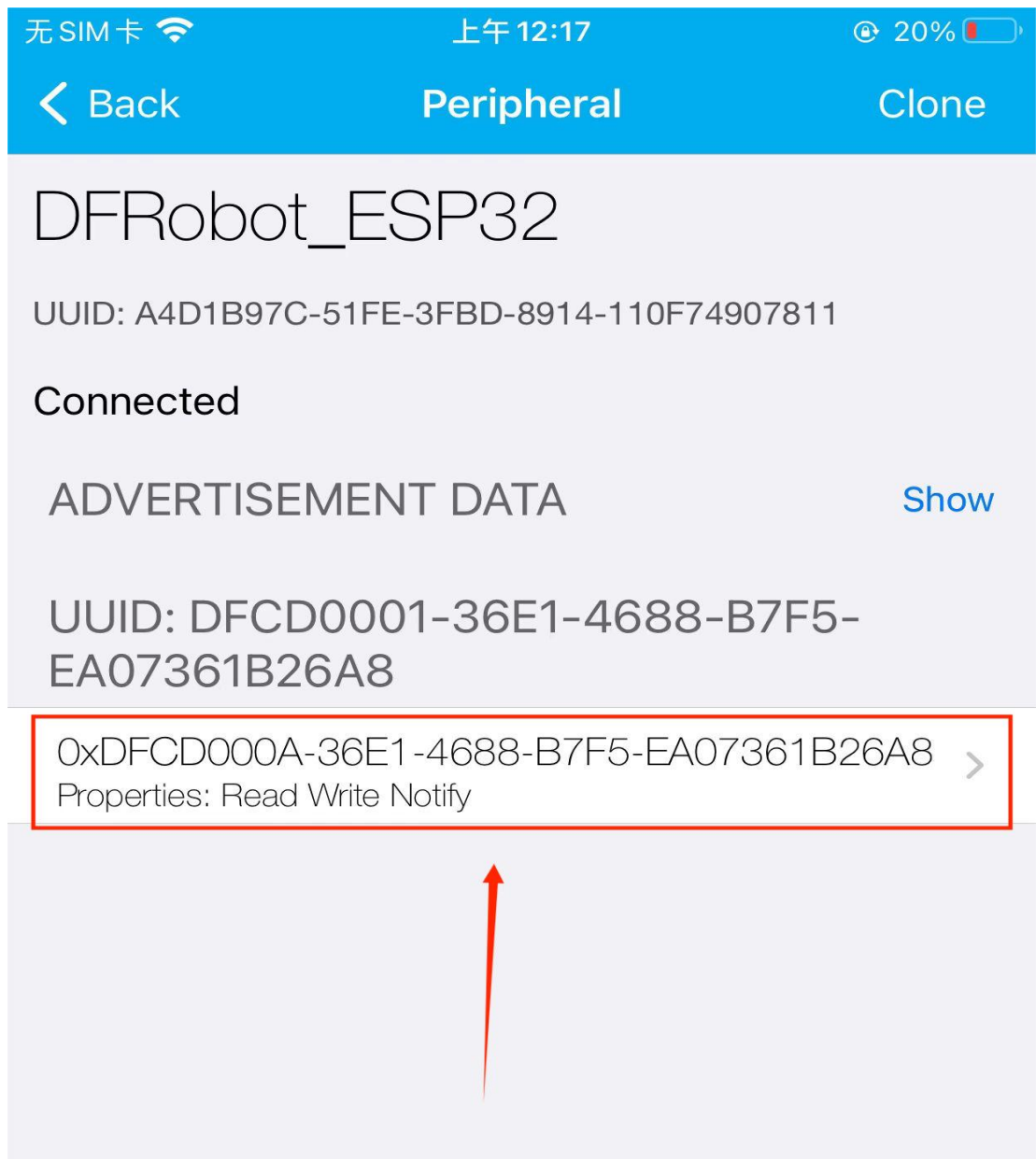
Peripherals Nearby

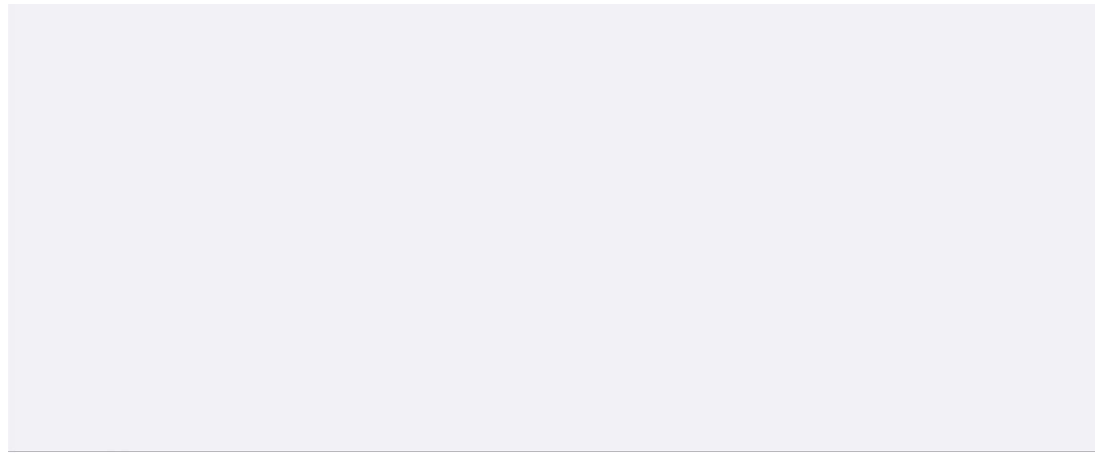
-  **GWM6A52MH264451** >
-97 1 service
-  **Unnamed** >
-100 No services
-  **HOZ0A223201A39E** >
--- 1 service
-  **HB7780352239** >
-102 No services
-  **Unnamed** >
--- No services
-  **Unnamed** >
-40 No services
-  **Unnamed** >
--- 1 service





Step 2. Click on the icon as shown in the figure.





Peripherals



Virtual Devices



Log

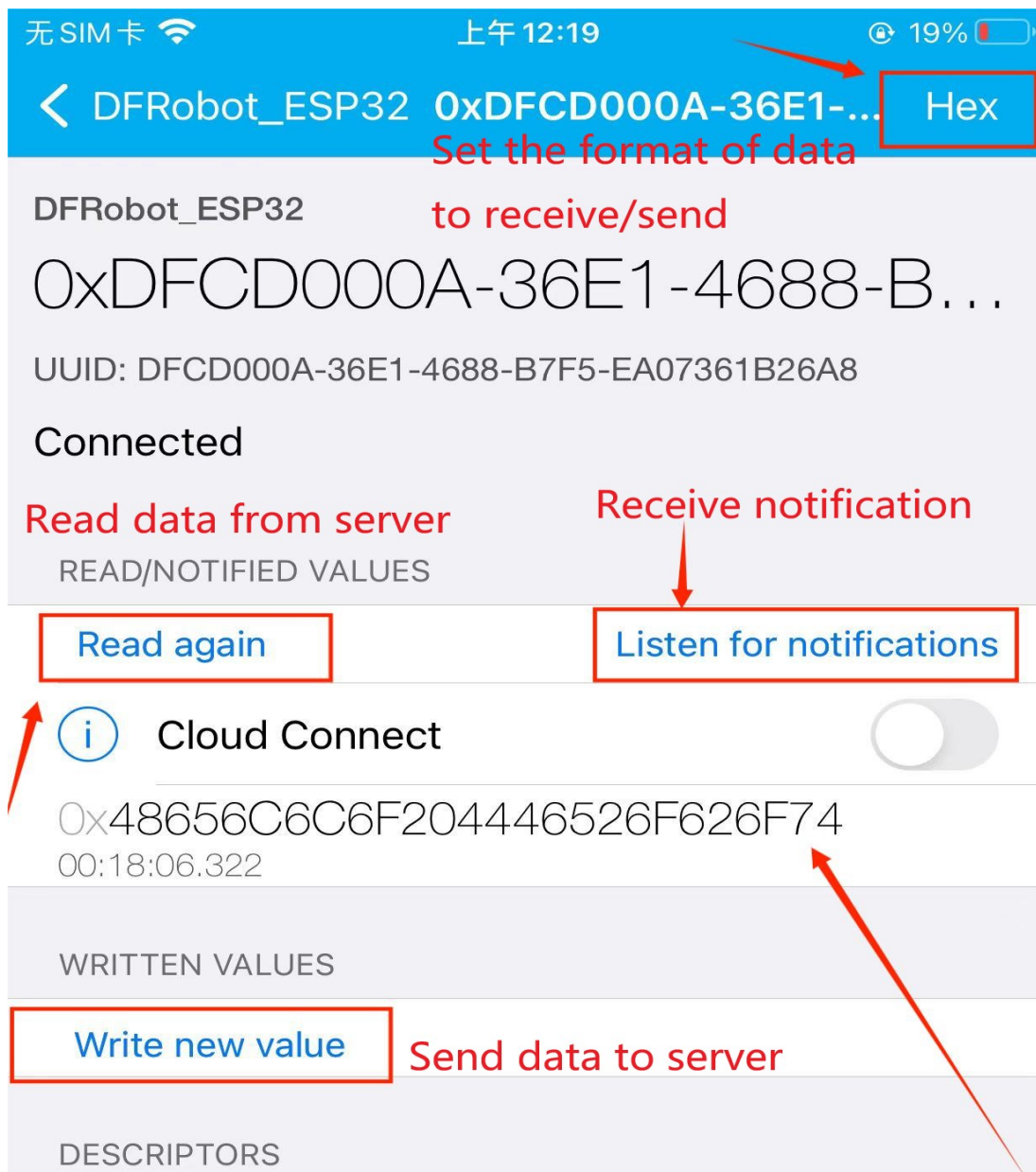


Learn



Settings

Step 3. Send data to the server.








0 **Send 48656C6C6F204446526F626F74**
Client Characteristic Configuration

PROPERTIES

Read

Write

 **Peripherals**  Virtual Devices  Log  Learn  Settings

Step 4. The transmitted data can be viewed.

无SIM卡 上午 12:27 17%

< DFRobot_ESP32 0xDFCD000A-36E1-... Hex

DFRobot_ESP32

0xDFCD000A-36E1-4688-B...

UUID: DFCD000A-36E1-4688-B7F5-EA07361B26A8

Connected

READ/NOTIFIED VALUES

Read again Listen for notifications

i Cloud Connect

0x48656C6C6F204446526F626F74
00:27:27.172

0x48656C6C6F204446526F626F74
00:26:06.162

WRITTEN VALUES **Transmitted data**

Write new value

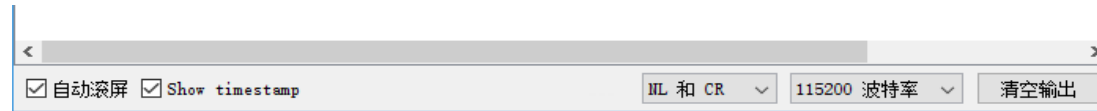
The screenshot shows a configuration window for a device. At the top, the MAC address `0x48656C6C6F204446526F626F74` and its corresponding UUID `00:27:27.036` are displayed in a red-bordered box. Below this, the 'DESCRIPTORS' section shows a single entry: `0` with the label 'Client Characteristic Configuration'. The 'PROPERTIES' section is currently empty. At the bottom, there are five navigation icons: 'Peripherals' (highlighted in blue), 'Virtual Devices', 'Log', 'Learn', and 'Settings'.

Step 5. The received data will be viewed on serial port.

The screenshot shows a serial port terminal window titled 'COM37'. The terminal displays the following output:

```
00:23:44.141 -> ets Jul 29 2019 12:21:46
00:23:44.141 ->
00:23:44.141 -> rst:0x1 (POWERON_RESET),boot:0x13 (SPI_FAST_FLASH_BOOT)
00:23:44.141 -> configspi: 0, SPIWP:0xee
00:23:44.141 -> clk_drv:0x00,q_drv:0x00,d_drv:0x00,cs0_drv:0x00,hd_drv:0x00,wp_drv:0x00
00:23:44.141 -> mode:DIO, clock div:1
00:23:44.141 -> load:0x3fff0030,len:1412
00:23:44.141 -> load:0x40078000,len:13400
00:23:44.175 -> load:0x40080400,len:3672
00:23:44.175 -> entry 0x400805f8
00:27:26.880 -> *****
00:27:26.880 -> New value: Hello DFRobot
00:27:26.880 -> *****
```

The line `New value: Hello DFRobot` is highlighted with an orange box. Below the terminal output, the text **Received data from the client** is written in red.



10.5 IFTTT

10.5.1 What is IFTTT?

If This Then That (commonly known as IFTTT, /ift/), is a web-based service that allows users to create chains of conditional statements triggered by changes that occur within other web services. It is both a website and a mobile app of free service with the following slogan: "Put the Internet to work for you". IFTTT aims to help people use the open API of various websites to monitor the triggers set by users. If triggers are triggered, actions set by users will be executed. Usually, we can create n applets to meet our various automation needs.

IFTTT

Recipe

If This Then That

Trigger

Action

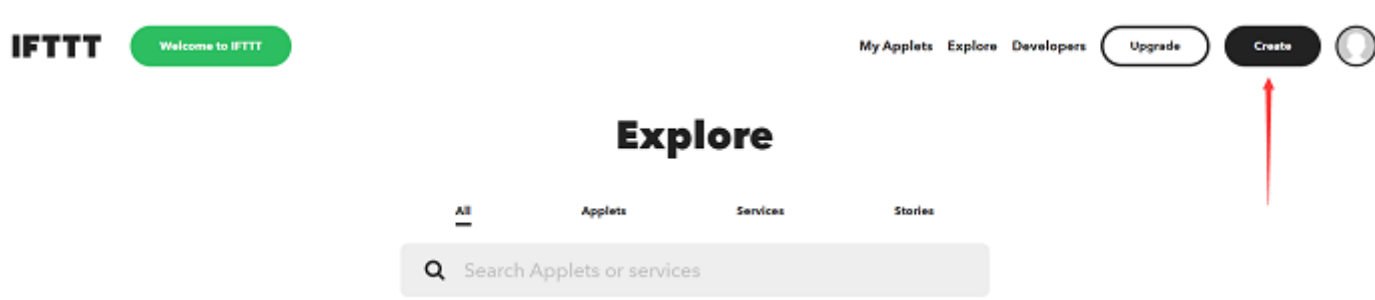
10.5.2 Sending Email

This application utilizes two small programs, Webhooks and Email, to achieve sending an HTTP POST request every 10 seconds. Once the Trigger is activated, the Action is executed to send a data email.

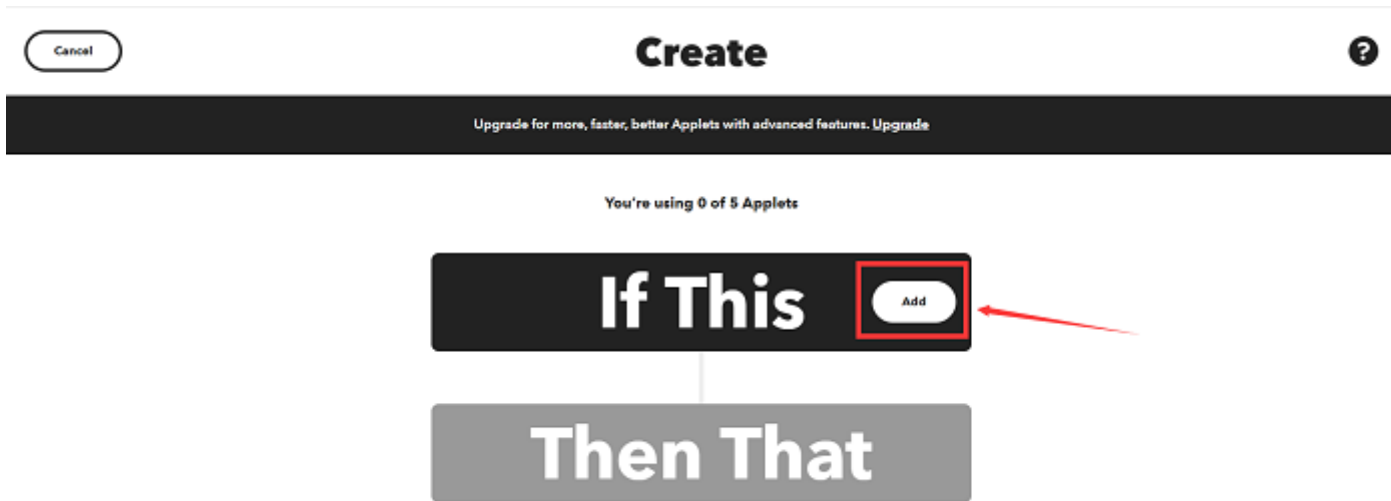
10.5.2.1 Operation Steps

- Access the IFTTT website: <https://ifttt.com/> (<https://ifttt.com/>)
- Register an account if you don't have one.
- Configure IFTTT
- Step 1. Create Trigger

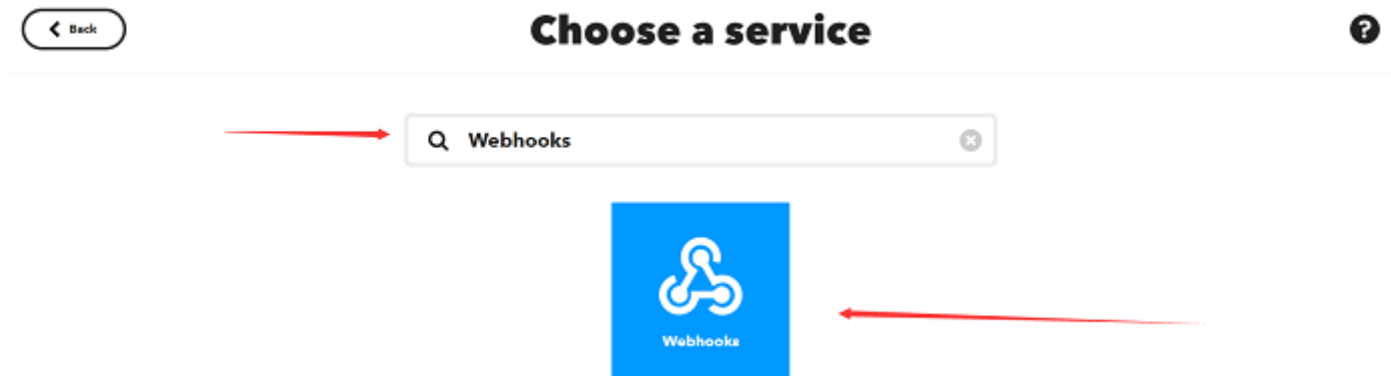
1. Sign in the IFTTT, click "create" to create your APP.



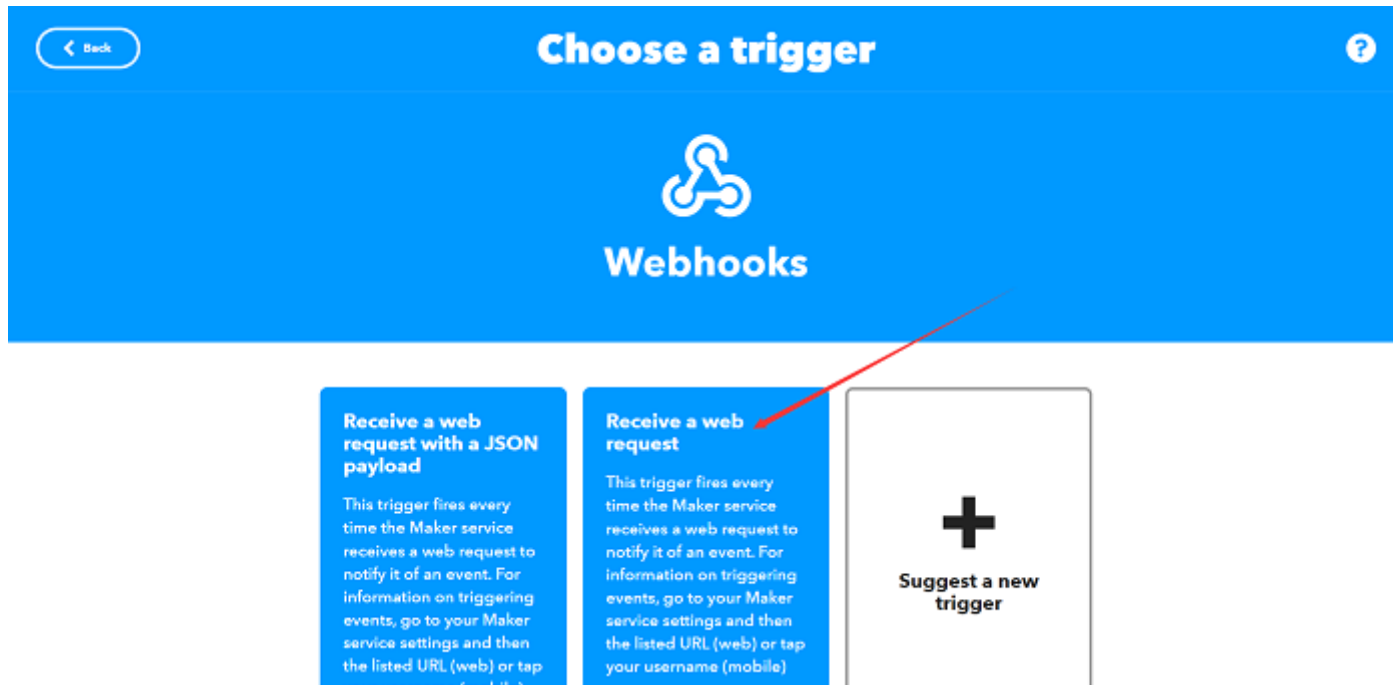
2. Click "Add" in "If This".



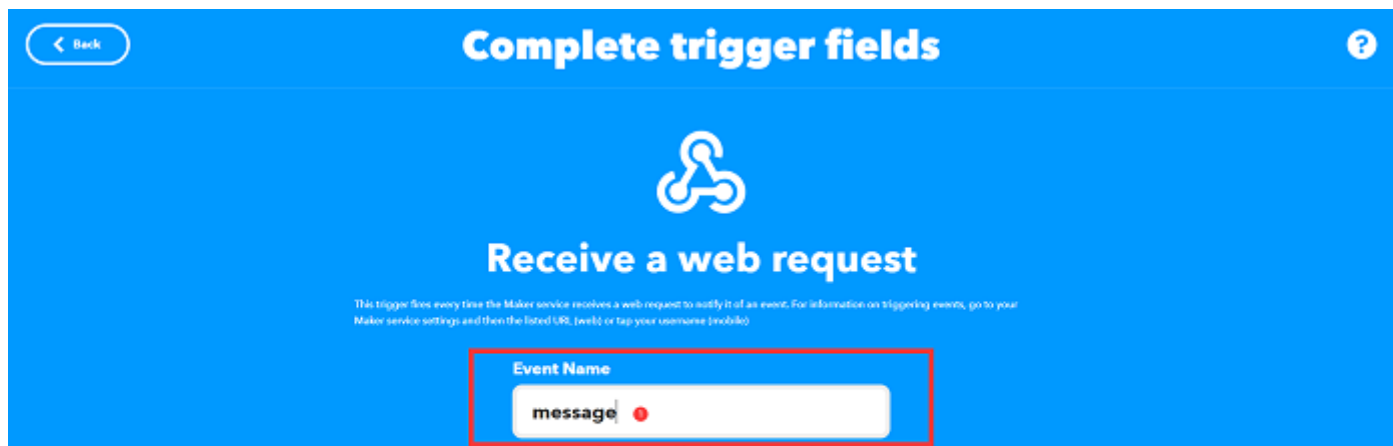
3. Input "Webhooks".



4. Click to enter "Webhooks", and select "Receive a web request".



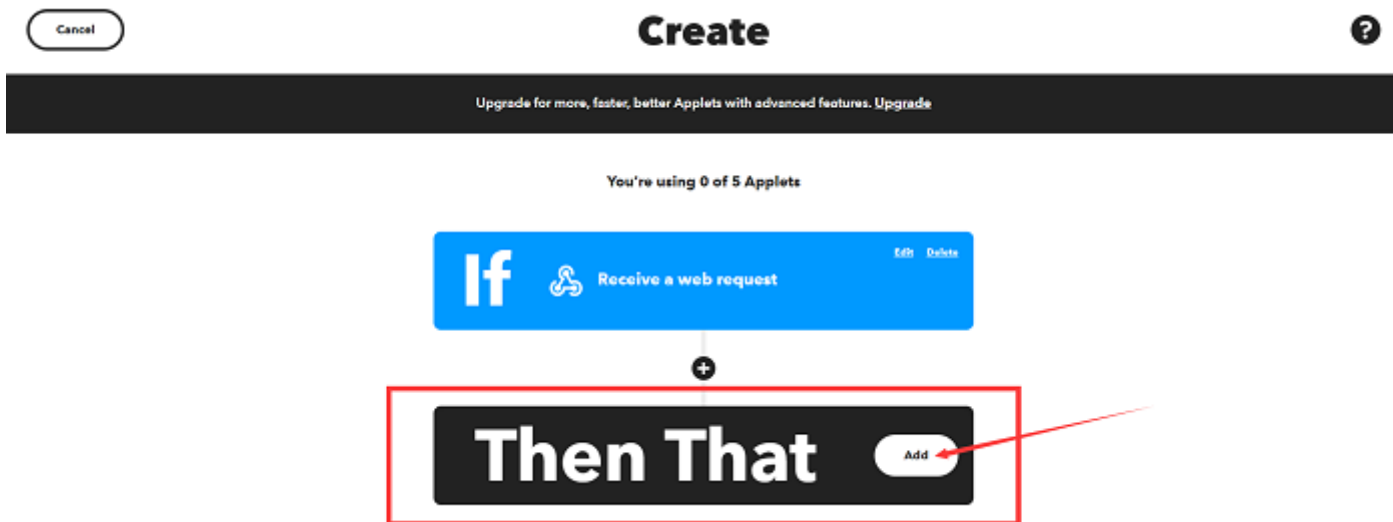
5. Next, create our Event Name "message" and click "Create trigger" to complete the setup for "this".



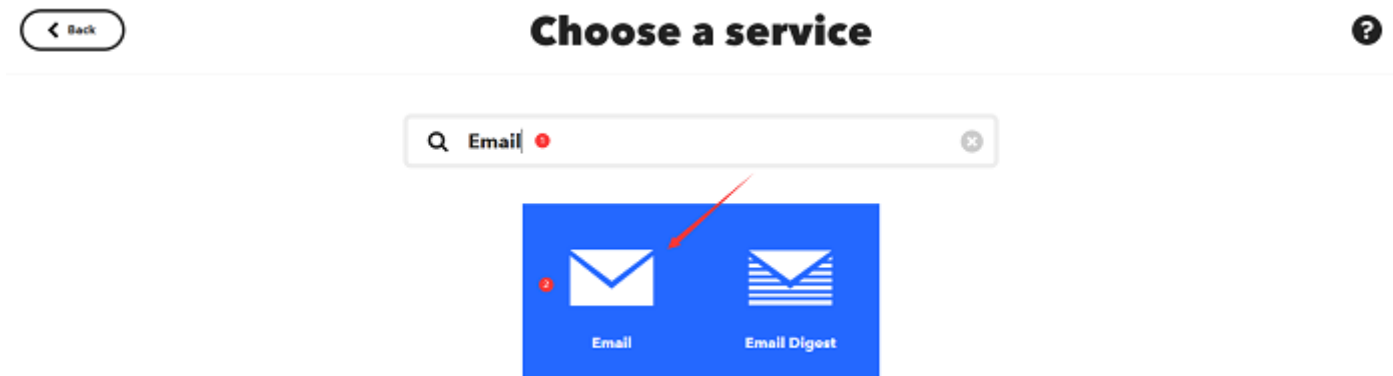


- **Step 2. Create Action**

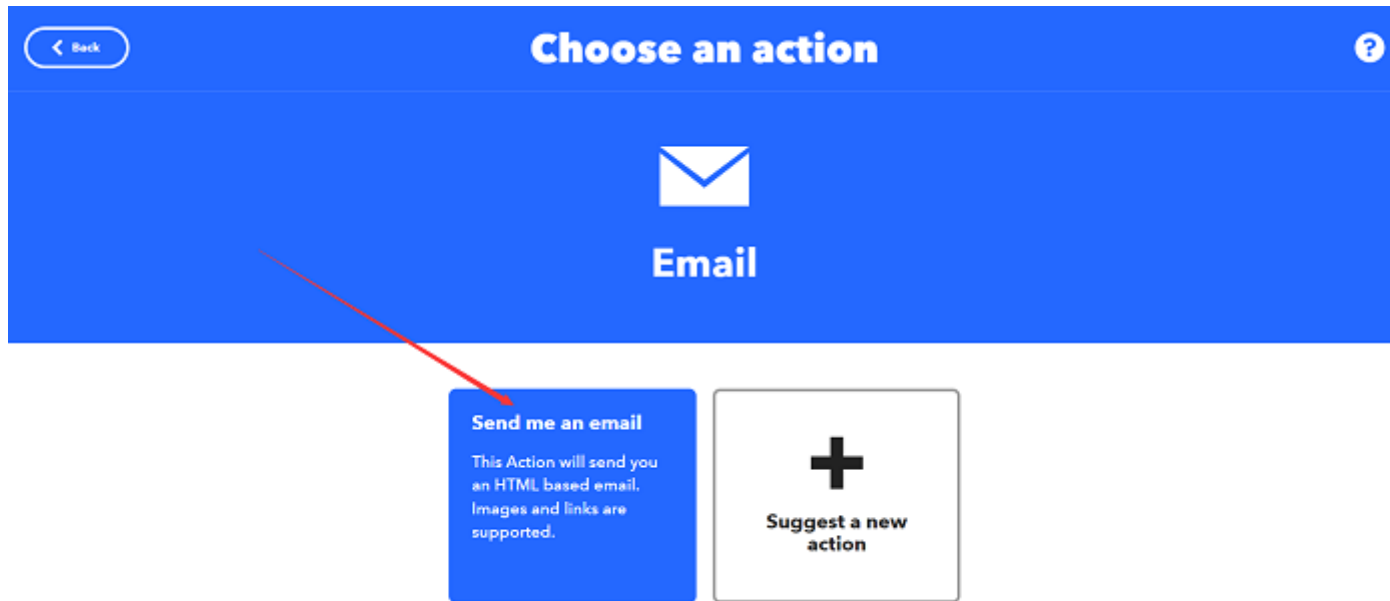
1. After setting up the trigger for "this", proceed to add the action instruction for "that" by clicking "Add" in the "Then That" section.



2. Search and click on "Email".



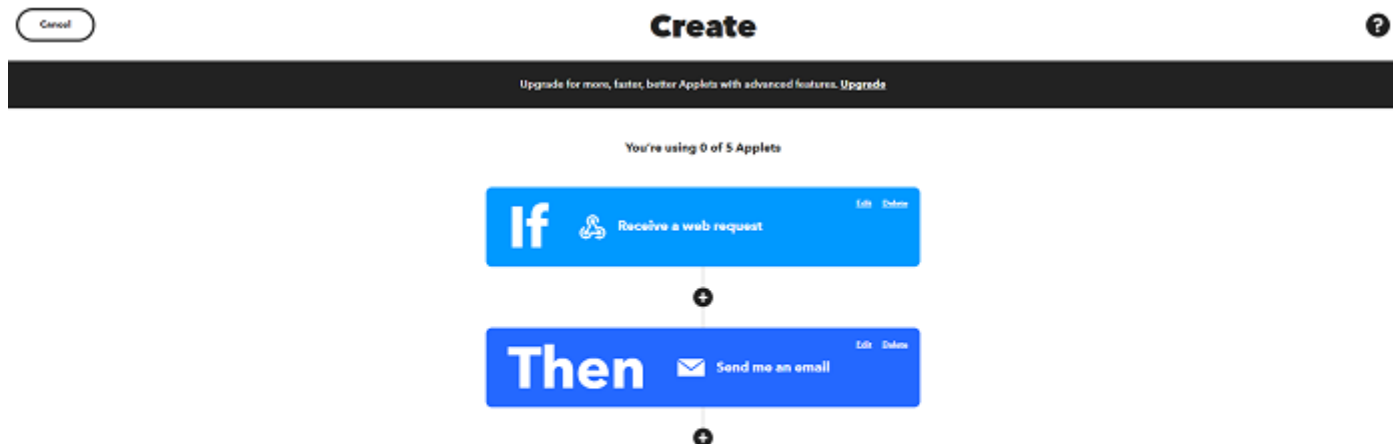
3. Select "Send me an email".



4. Once selected, you can edit the email content by entering the email subject in the "subject" field and the email body in the "Body" field. Then click "Create action" to complete the creation.

The screenshot shows a blue interface for configuring an email action. At the top left is a 'Back' button and at the top right is a help icon. The main heading is 'Complete action fields'. Below it is an envelope icon and the title 'Send me an email'. A small note says 'This Action will send you an HTML based email. Images and links are supported.' There are two sections: 'Subject' and 'Body'. The 'Subject' section contains a text box with the text 'The event named " EventName ." occurred on the Maker Webhooks service' and an 'Add ingredient' button. The 'Body' section contains a text box with the text 'What: EventName
 When: OccurredAt
 Extra Data: Value1 , Value2 , Value3 ,' and an 'Add ingredient' button. At the bottom center is a large white 'Create action' button with a red arrow pointing to it.



5. Click "Continue".





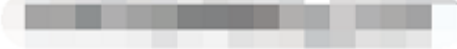
6. Review and then click "Finish".

Review and finish

Applet Title

If Maker Event "message", then Send me an email at

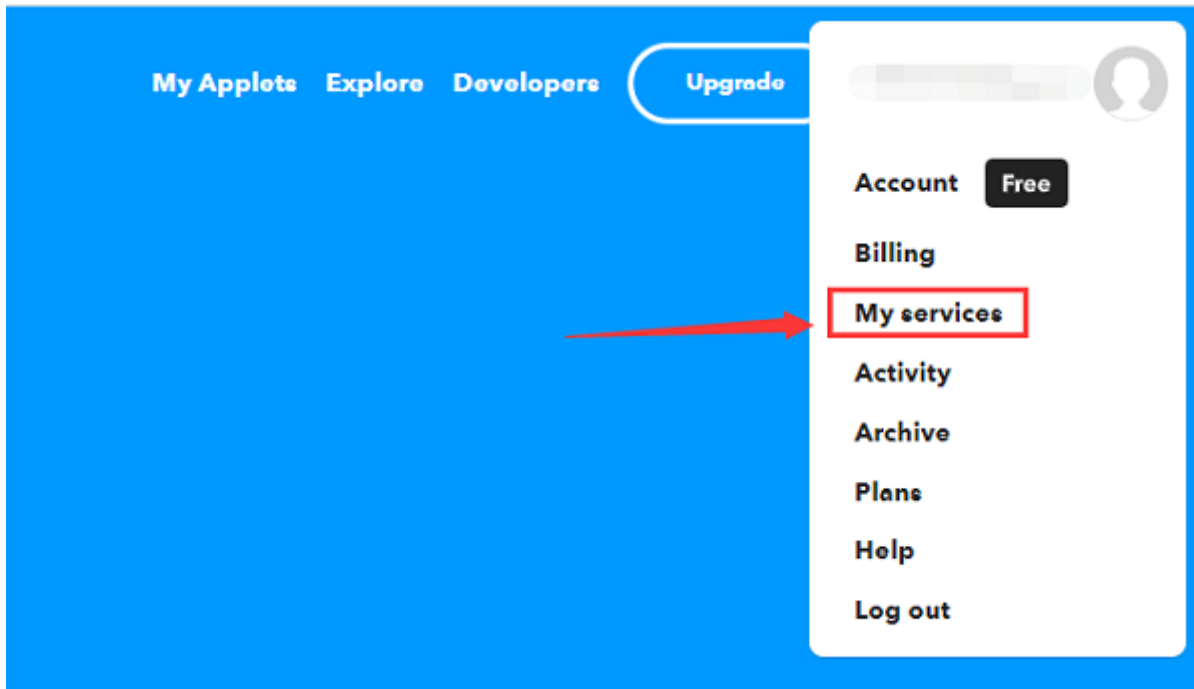


by g714173948 677/140











- **Step 3. Find IFTTT_Key**



1. Click "My services"



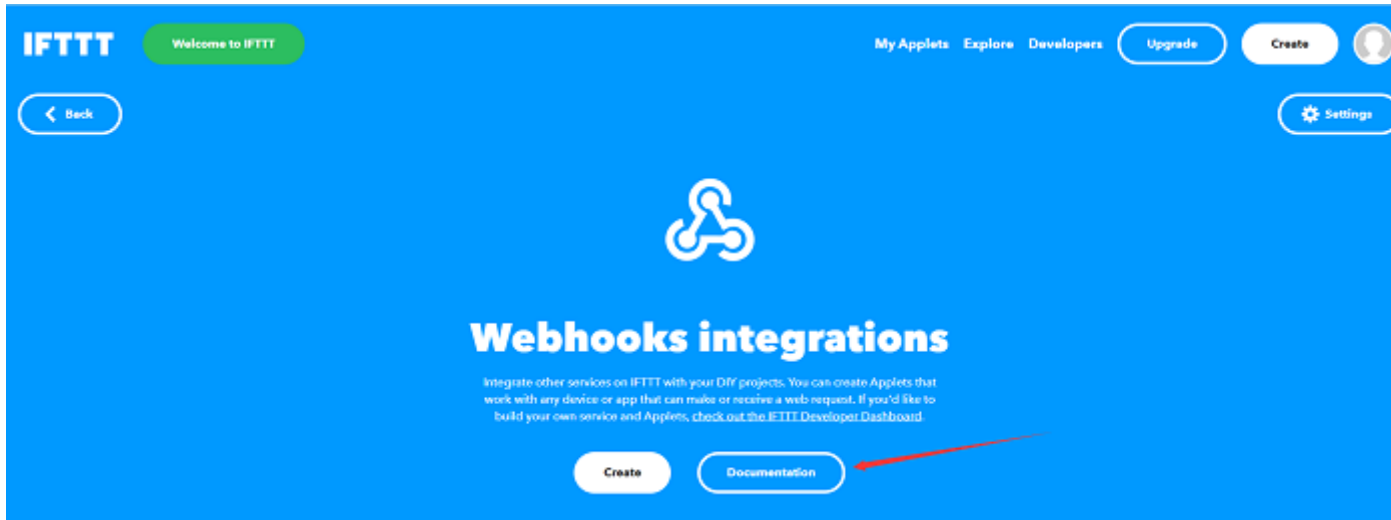
2. Click "Webhooks"

1 My services

-  **Date & Time** >
-  **Email** >
-  **Email Digest** >
-  **IFTTT** >
-  **RSS Feed** >
-  **Space** >
-  **Weather Underground** >
-  **Webhooks** >

2  

3. Click "Documentation"

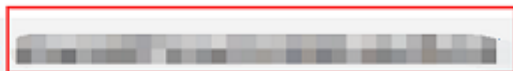


3. Copy the key at the position shown below in the diagram.



Your key is:

◀ Back to service



To trigger an Event with an arbitrary JSON payload

Make a POST or GET web request to:

```
https://maker.ifttt.com/trigger/[event]/json/with/key/45e0WPziszV490nNdfH5X
```

** Note the extra /json path element in this trigger.*

With any JSON body. For example:

```
{ "this" : [ [ "is": [ "some": [ "test", "data" ] ] ] ] }
```

You can also try it with `curl` from a command line.

```
curl -X POST -H "Content-Type: application/json" -d '{"this":[{"is":["some":["test","data"]]}]}'  
https://maker.ifttt.com/trigger/[event]/json/with/key/45e0WPziszV490nNdfH5X
```

Please read [our FAQ](#) on using Webhooks for more info.

Test It

To trigger an Event with 3 JSON values

Make a POST or GET web request to:

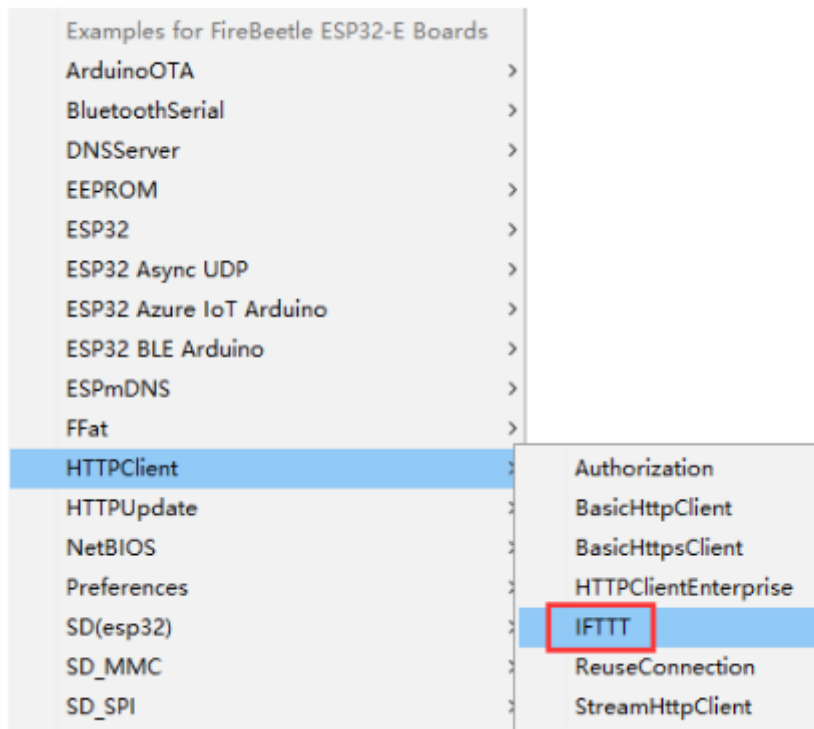
```
https://maker.ifttt.com/trigger/[event]/with/key/45e0WPziszV490nNdfH5X
```

With an optional JSON body of:

10.5.2.2 Sample Code

- Burning Arduino sketch

– Open the built-in sample sketch



- Function: If a "message" is received, FireBeetle 2 ESP32-E send email to the preset email address.

```
#include <WiFi.h>
#include <HTTPClient.h>
//Configure WiFi name and password
char *WIFI_SSID          = "WIFI_SSID";
char *WIFI_PASSWORD     = "WIFI_PASSWORD";
//Configure IFTTT
char *IFTTT_ENVENT      = "Your_Event";
char *IFTTT_KEY         = "Your_Key";
//IFTTT Send Message
char *IFTTT_VALUE_1     = "Value1";
char *IFTTT_VALUE_2     = "Value2";
char *IFTTT_VALUE_3     = "Value3";
HTTPClient ifttt;
unsigned long lastTime = 0;
unsigned long timerDelay = 10000;
void setup() {
  Serial.begin(115200);
  WiFi.begin(WIFI_SSID, WIFI_PASSWORD);
  Serial.println("Connecting");
  while(WiFi.status() != WL_CONNECTED) {
    delay(500);
    Serial.print(".");
  }
  Serial.println("");
  Serial.print("Wifi Connect Success");
}
void loop() {
  //Send an HTTP POST request every 10 seconds
  if ((millis() - lastTime) > timerDelay) {
    //Check WiFi connection status
    if(WiFi.status() == WL_CONNECTED){
```

```
if(WiFi.status() == WL_CONNECTED){
    ifttt.IFTTTBegin(IFTTT_ENVENT,IFTTT_KEY);
    int dataSendState = ifttt.IFTTTSend(IFTTT_VALUE_1,IFTTT_VALUE_2,IFTTT_VALUE_3);
    Serial.println(dataSendState);//Whether the printing data is sent successfully
}else {

    Serial.println("WiFi Disconnected");
}
lastTime = millis();
}
}
```

- Configure Parameters in Arduino Code

```
//Configure WiFi name and password
char *WIFI_SSID          = "WIFI_SSID";//Input WiFi name
char *WIFI_PASSWORD     = "WIFI_PASSWORD";//Input WiFi Password
//Configure IFTTT
char *IFTTT_ENVENT      = "Your_Event";//Input Event Name
char *IFTTT_KEY         = "Your_Key";//Input the key you found in IFTTT
//IFTTT Send Message
char *IFTTT_VALUE_1     = "Value1";
char *IFTTT_VALUE_2     = "Value2";
char *IFTTT_VALUE_3     = "Value3";//Configure the three values in email information
```

10.5.2.3 Result

Receive the data from FireBeele-ESP32-E in the Email box.

What: message

When: December 8, 2020 at 02:26PM

Extra Data: Value1, Value2, Value3,



Manage



[Unsubscribe](#) from these notifications or sign in to manage your [Email service](#).

FAQ

1. Cannot add board link to Arduino IDE?

A: First off all, make sure the Network connection is in good condition. Hope this post (<https://forum.arduino.cc/t/arduino-ide-cant-download-libraries-and-boards/650155/5>) from Arduino helps. If that doesn't work, try replacing the board link from

https://download.dfrobot.com/FireBeetle/package_DFRobot_index.json

(https://download.dfrobot.com/FireBeetle/package_DFRobot_index.json) to

http://download.dfrobot.com/FireBeetle/package_DFRobot_index.json

(**http://download.dfrobot.com/FireBeetle/package_DFRobot_index.json**) and reload it.

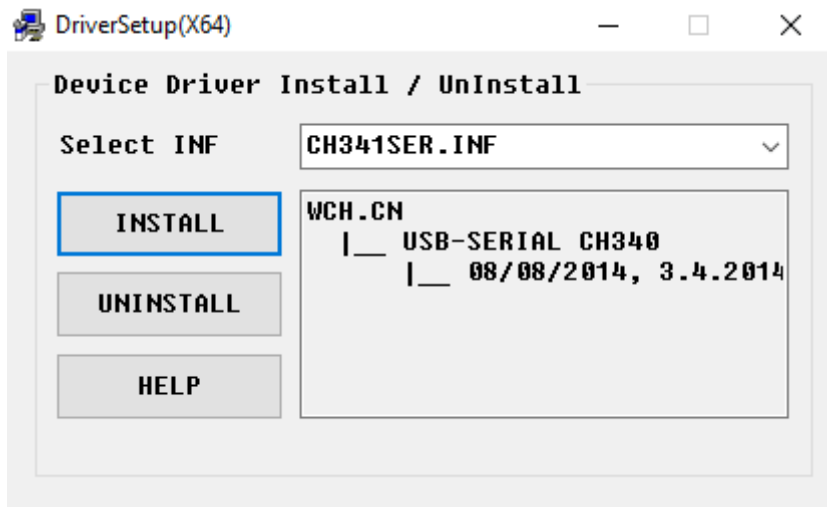
2. When using the SD library, the serial monitor does not show any print output or displays a connection failure after uploading the program

A: Press RST button to reset and try again.

3. The driver is not installed automatically after plugging into the device?

A: FireBeetle 2 ESP32-E uses the CH340 serial chip, which can be used without installing a driver in most devices. If you find that the driver is not automatically installed when you plug in the device, you can manually install it yourself:


- CH340 Driver for Windows (<https://dfimg.dfrobot.com/nobody/wiki/0e0d6b3864f7163833ec5d7ad4af7632.EXE>)
- CH340 Driver for MAC (<https://dfimg.dfrobot.com/nobody/wiki/c195a13df2a1989d5dc04e76e6bcb701.ZIP>)



For any other questions, advice or cool ideas to share, please visit the **DFRobot Forum** (<https://www.dfrobot.com/forum/>).

More Documents

- FireBeetle 2 2Schematic (<https://dfimg.dfrobot.com/nobody/wiki/fd28d987619c16281bdc4f40990e5a1c.PDF>)

 Get **FireBeetle_Board_ESP32_E** (<https://www.dfrobot.com/product-2195.html>) from DFRobot Store or **DFRobot Distributor**.
(<https://www.dfrobot.com/distributor>)

Turn to the Top