
FIREBEETLE BOARD-ESP32 USER MANUAL V0.1

CONTENTS

CONTENTS	2
Chapter1 Introduction.....	3
Introduction of FireBeetle	3
Introduction of FireBeetle Board-ESP32.....	4
Chapter2: Quick-start	7
FireBeetle Board-ESP32 Hardware	7
Characteristics.....	7
Specification	8
Arduino IDE for FireBeetle Board-ESP32.....	9
STEP1: The way to download Arduino IDE software.	9
STEP2: Language setting of Arduino IDE	11
STEP3: Installation of the core center of the FireBeetle Board-ESP32 development board.	11
STEP4: Connect FireBeetle Board-ESP32 to computer.	13
STEP5: Processing in the Arduino IDE.....	15
STEP6: Upload codes to FireBeetle-ESP32 motherboard.	16
Chapter 3: The Basics of Using Peripherals in the FireBeetle Board-ESP32	20
Project1: The experiment of Serial Port	20
Project2: PWM (Breathing light)	22
Project3 ADC	25
Project4 I2C.....	28
Project5 SPI.....	35
Project6: Hall Sensor	39
Project7 DAC	41
Project8 Touch Sensor	43
Project9 Deep_Sleep (Low power Management).....	45

*Click the top left corner, the community of DFRobot, a pleasant surprise!

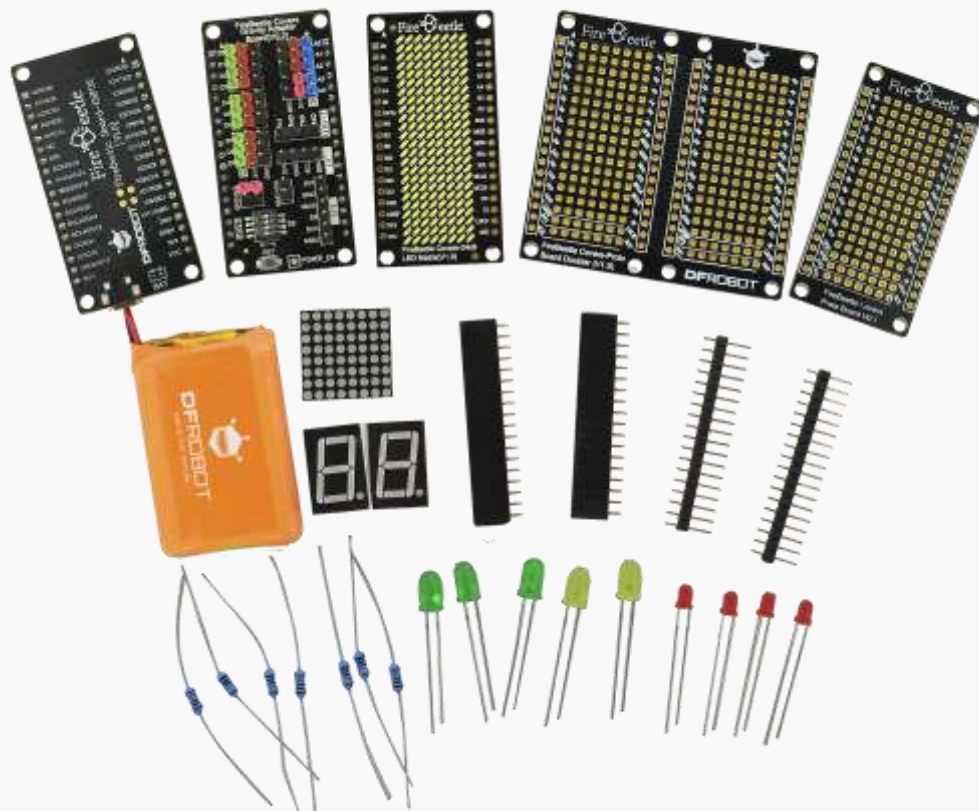
Chapter1 Introduction

Introduction of FireBeetle

FireBeetle has been translated to firefly in Chinese. It is a new product range that developed by DFRobot. The ultralow-power design of external hardware and small compatibility interface can fast build Internet of Things (IOT) conveniently.

In simple terms, FireBeetle is an easy to use Internet of Things development platform. It is based on the theme of series of low-power hardware development and designed to work simple for intelligent hardware enthusiasts, interaction designers and electronic software engineers.

For example, a simple Internet of Things project — intelligent watering device of real-time soil moisture monitoring. When the humidity is less than 300(a parameter), it will control the pump watering and upload data in every 5 mins. Therefore, you can check the real-time data on apps of mobile phones, such as Blynk. You can also monitor the ambient temperature by the installation of temperature and humidity sensors on the watering device. See, it's so easy to build an intelligent watering device!



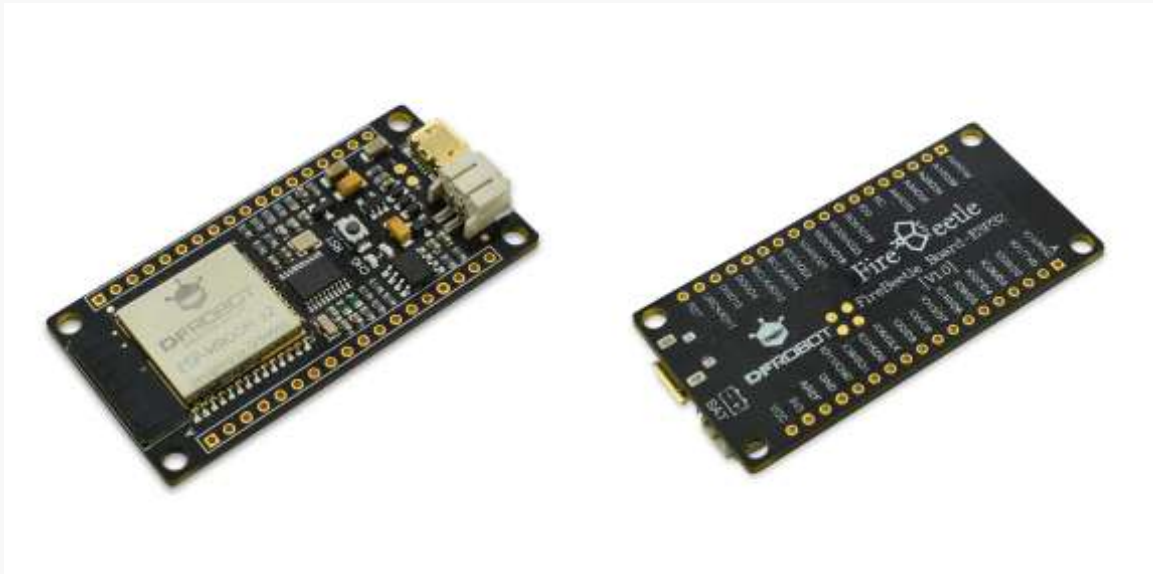
There are three categories of FireBeetle, containing Boards (main control), Covers (expansion boards) and related Accessories. The core controller chips of Boards mainly are MCUs such as ESP32,

Bluno etc. These chips are of high performance, low power consumption and has the function of wireless transmission. Covers contain LED dot matrixs, I2S audio drivers, GPS/GPRS/SIM etc. They can provide rich peripherals for the motherboard. Accessories contain 2.54mm pin headers, female headers, LED lights, buttons and other extensible peripheral hardware.

Caution: A lurry of FireBeetle products are still in development.

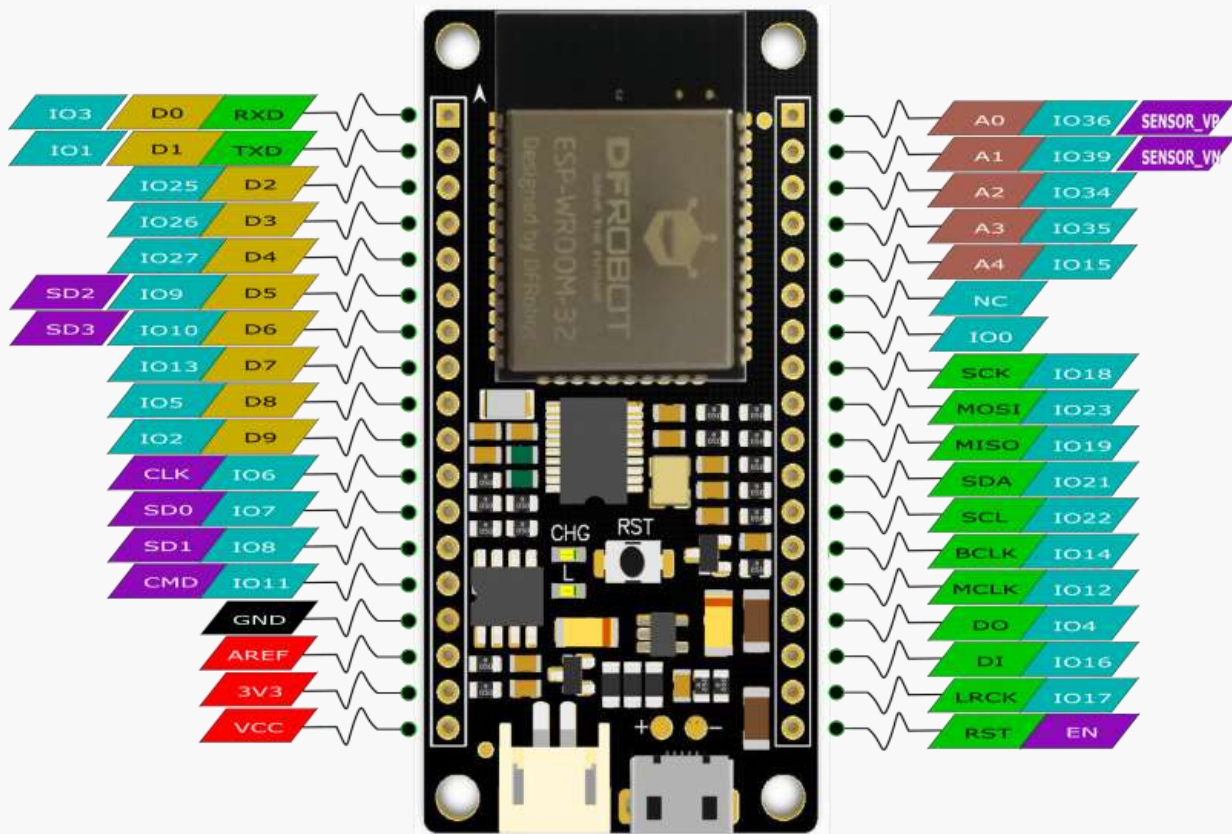
Introduction of FireBeetle Board-ESP32

FireBeetle Board-ESP32 is one of the FireBeetle series and the chip of the master control board is ESP32. Aimed at Internet of Things, the SOC combines Wi-Fi, Bluetooth and MCU. The master control board uses ultra-low power consumption external hardware design (an actual measurement of the electricity current under low power consumption mode is 10 μ A), supporting onboard lithium battery charging. The best choice for mobile devices, wearable electronics and IOT applications. It can directly use to low power consumption projects.



Additionally, FireBeetle Board-ESP32 has abundant peripheral equipment, e.g. ADC, I2C, I2S, SPI, UART etc. And the programming is completely compatible with Arduino IDE.

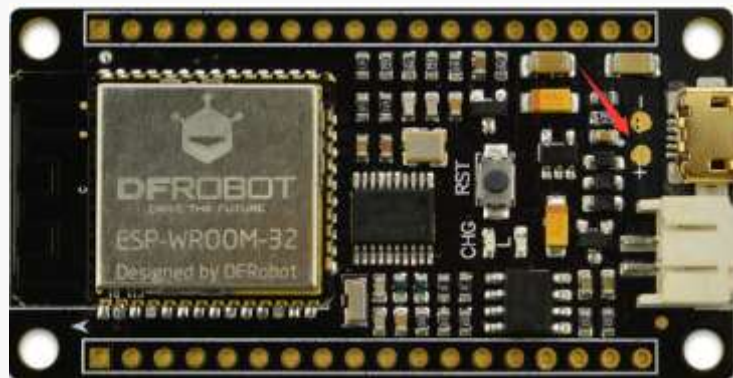
Caution: All peripheral equipment of FireBeetle Board-ESP32 control board can be configured to any pin. When programed with Arduino IDE, all peripheral equipments adopt default profile except special one. The following image shows default profile and corresponding hardware IO.



- IO3/RXD: D0(Arduino), the RX of Serial, connects to IO3 of ESP32
- IO1/TXD: D1 (Arduino), the TX of Serial, connects to IO1 of ESP32
- IO25/D2: D2 (Arduino) of GPIO digital input and output port and PWM output pin, connect to IO25 of ESP32.
- IO26/D3: D3 (Arduino) of GPIO digital input and output port and PWM output pin, connect to IO26 of ESP32.
- IO27/D4: D4 (Arduino) of GPIO digital input and output port and PWM output pin, connect to IO27 of ESP32.
- IO9/ D5: D5 (Arduino) of GPIO digital input and output port and PWM output pin, connect to IO9 of ESP32.
- IO10/D6: D6 (Arduino) of GPIO digital input and output port and PWM output pin, connect to IO9 of ESP32.
- IO13/D7: D7 (Arduino) of GPIO digital input and output port and PWM output pin, connect to IO13 of ESP32.
- IO5/D8: D8 (Arduino) of GPIO digital input and output port and PWM output pin, connect to IO5 of ESP32.
- IO2/D9: D9 (Arduino) of GPIO digital input and output port and PWM output pin, connect to IO2 of ESP32.
- CLK: Clock pin to IO6 of ESP32
- SD0: SD0 port to IO7 of ESP32
- SD1 : SD1 port to IO7 of ESP32
- CMD: CMD port to IO11 of ESP32
- GND: Power line ground
- AREF: input voltage for reference, here connect to NC.
- 3V3: 3.3V Vo, can provide 600mA current out at most.
- VCC : The Vi/Vo is 5V charged by USB and 3.7V charged by lithium battery.
- IO36/A0: A0 (Arduino), analog input, connect to IO36 of ESP32.
- IO39/A1: A1 (Arduino), analog input, connect to IO39 of ESP32.
- IO34/A2: A2 (Arduino), analog input, connect to IO34 of ESP32.

- IO35/A3: A3 (Arduino), analog input, connect to IO35 of ESP32.
- IO15/A4: A4 (Arduino), analog input, connect to IO15 of ESP32.
- NC: Not connected
- IO0: Digital interface, connect to IO0 of ESP32.
- SCK: Clock pin of SPI, connect to IO18 of ESP32.
- MOSI: SPI's MOSI data-wire, connect to IO23 of ESP32.
- MISO: SPI's MOSO data-wire, connect to IO19 of ESP32.
- SDA: I2C's data-wire, connect to IO21 of ESP32.
- SCL: Clock pin of I2C, connect to IO22 of ESP32.
- BCLK: Clock pin of I2S, connect to IO14 of ESP32.
- MCLK: Clock pin of I2S, connect to IO12 of ESP32.
- DO: DO data-wire of I2S, connect to IO4 of ESP32.
- DI: DI data-wire of I2S, connect to IO16 of ESP32.
- LRCK: LRCK data-wire of I2S, connect to IO17 of ESP32.
- RST: Low level reset port.

For master control board of the FireBeetle, we have reserved external DC ports for more complex projects, supporting other peripheral chargers such as wireless charging module and solar charging module.



**Caution: + should be connected to the positive pole of the external charger;
- should be connected to the negative pole of the external charger.
The input voltage is from 4.7V to 6V.**

Chapter2: Quick-start

FireBeetle Board-ESP32 Hardware

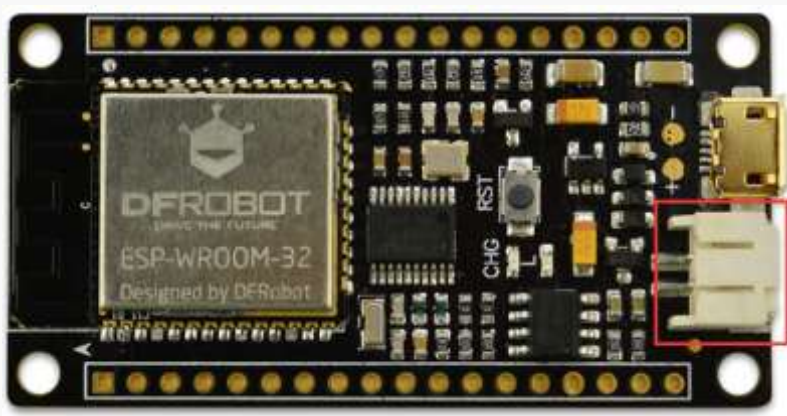
In the Chapter1, we have introduced pins and corresponding peripheral default profiles of Arduino IDE. Please refer to the Chapter1 if you have any enquiry related to the special provision of pins.

Characteristics

- Working voltage: 3.3V
- Input voltage: 3.3V~5V
- Support electric current of low power consumption: 10 μ A
- Support maximum discharge current: 600mA
- Support maximum charge current: 500mA
- Support USB charging.
- Processer: Tensilica LX6 dual core processer (One for high speed connection; one for independent programing).
- Frequency: 240MHz
- SRAM: 520KB
- Flash: 16Mbit
- Wi-Fi standard: FCC/CE/TELEC/KCC
- Wi-Fi protocol: 802.11 b/g/n/d/e/l/k/r (802.11n, high speed can reach to 150 Mbps), converge A-MPDU and A-MSDU, supporting 0.4us protecting interval.
- Frequency range: 2.4~2.5 GHz
- Bluetooth protocol: Comply with BR/EDR/BLE standard of Bluetooth v4.2.
- Bluetooth audio: the current under low power consumption of CVSD and SBC is 10 μ A
- Working current: 80mA in average
- Frequency range: 2.4~2.5GHz
- Support one-key downloading.
- Support micropython.
- On-chip clock: 40MHz crystal and 32.768 KHz crystal.
- Digital I/O: 10 (default setting of arduino)
- Simulative input: 5(default setting of arduino)
- SPI: 1 (default setting of arduino)
- I2C: 1 (default setting of arduino)
- I2S: 1 (default setting of arduino)
- LED_BUILTIN: D9
- Interface: FireBeetle series compatible
- Working temperature: -40 $^{\circ}$ C ~+85 $^{\circ}$ C

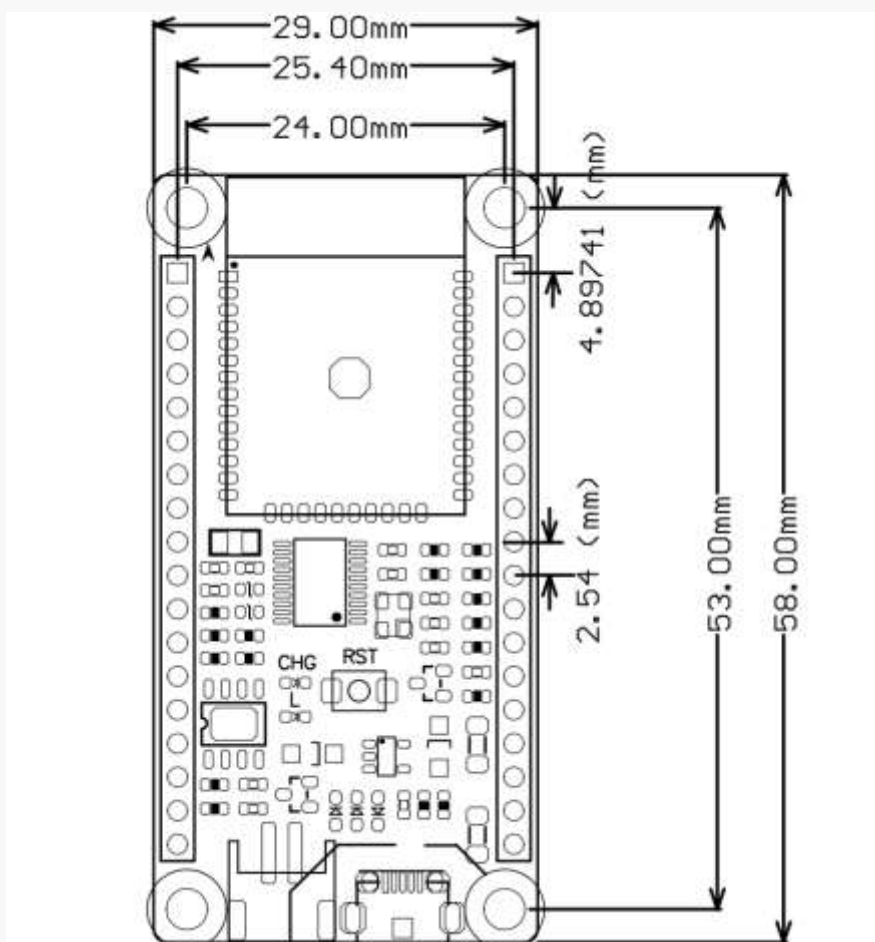
- Dimension: 24 × 53(mm)/0.94 x 2.09(inches)
- The dimension of mounting hole: inner diameter 3.1 mm; outside diameter 6mm.
- On-board reset bottom
- Hardware vision: V1.0

FireBeetle Board-ESP32 can be charged by lithium battery. The port as shown below.



Specification

FireBeetle Board-ESP32 mother board is fully compatible with FireBeetle range, the dimension parameters of it shown as below.



- Distance from pin to pin: 2.54mm.
- Distance between two mounting holes: 24mm/53mm.
- Dimension of mounting hole: 3.1mm.
- Dimension of motherboard: 29.00×58.00mm
- Thickness of the board: 1.6mm

Arduino IDE for FireBeetle Board-ESP32

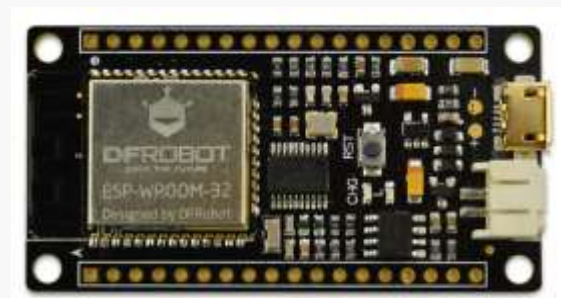
FireBeetle Board-ESP32 is easy for and on its users, even for a novice. The tutorial below present the way to download and install Arduino IDE software, processing in Arduino IDE.

3 Basic necessities: a FireBeetle Board-ESP32, USB data wire, and a computer that runs with Windows/Mac OS/Linux system and has network.

The FireBeetle Board-ESP32 and USB data wire that needed, as shown below.



USB data wire



FireBeetle Board-ESP32

Arduino IDE for FireBeetle Board-ESP32 let your Arduino IDE support FireBeetle Board-ESP32 control board. You can set up the development environment quickly by the instruction below.

STEP1: the way to download Arduino IDE software.

The instructions below are based on Windows operating system. It works as a reference to other operating systems.

Firstly, please download the latest version of Arduino IDE software from the official website.

Our official website: <http://arduino.cc/en/Main/Software>

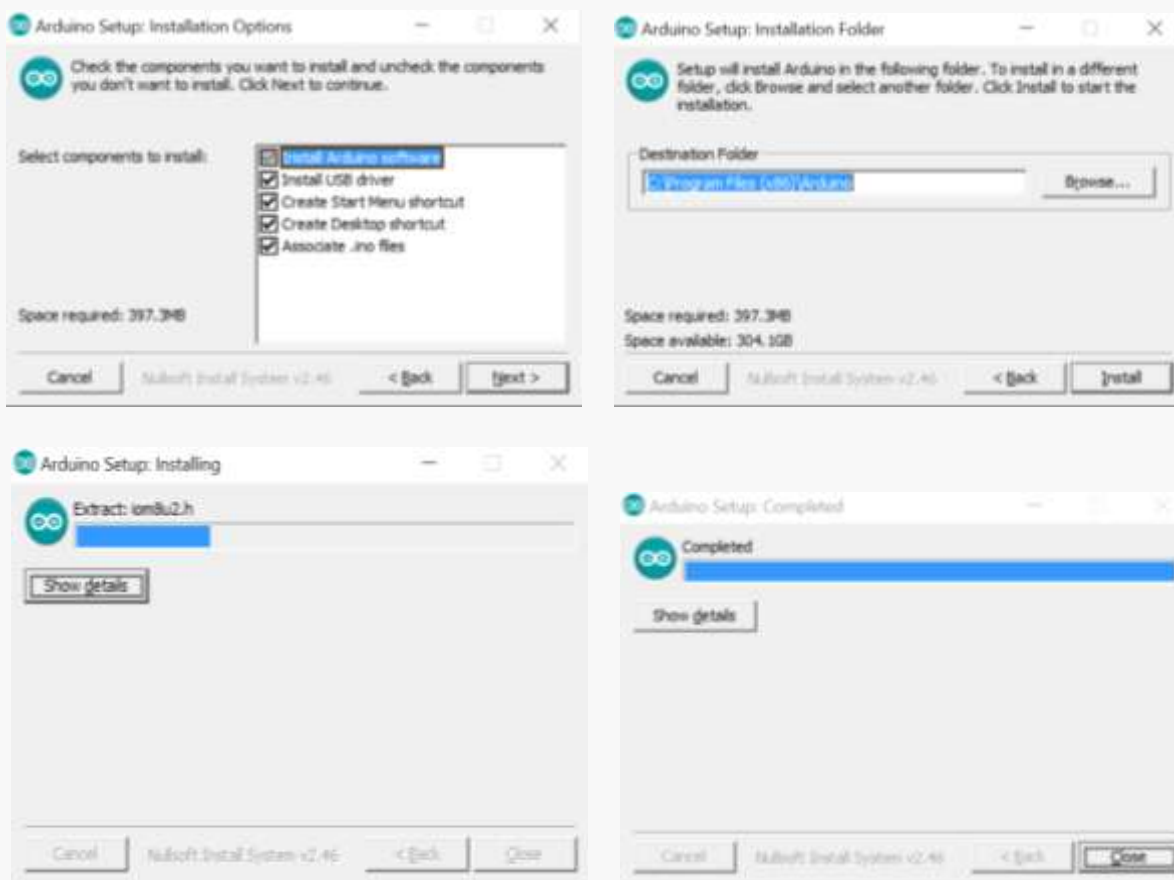
Download link: <http://arduino.cc/en/Main/Software>

Download the Arduino IDE



Caution: as the tutorial used Arduino IDE 1.8.0, please use Arduino 1.8.0 or later ones for the best experience.

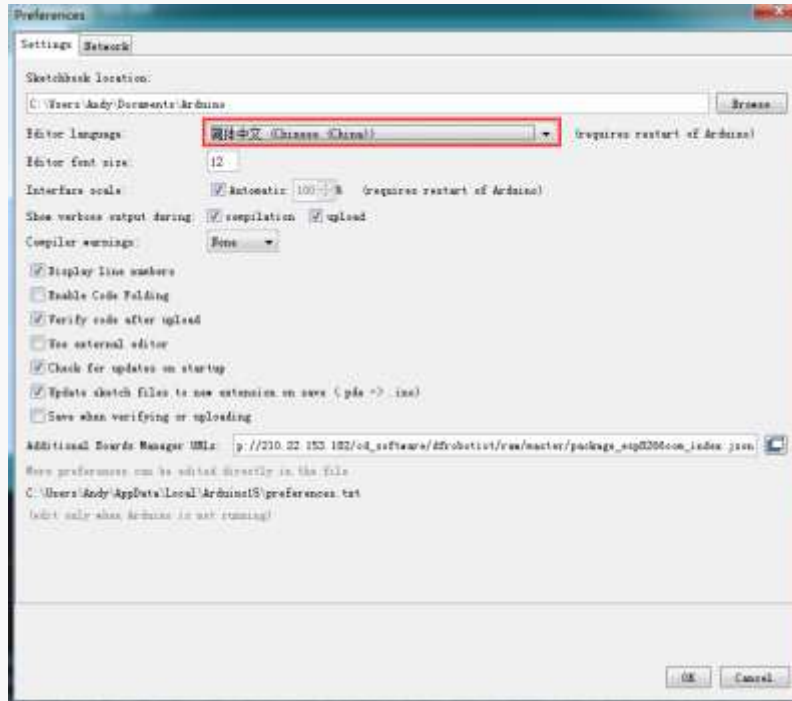
Please choose the corresponding installation package from the list to the right of the download page. Windows system users can choose either Windows installer or Windows ZIP to download. Nonetheless **Windows installer** is better for freshman. Installing driver by hand is needed for Windows ZIP. For Windows installer, there is no need for extra manual install drive. You can execute installation directly and follow the Setup Wizard to accomplish the configuration. The drive will be installed automatically after the installation.



STEP2: language setting of Arduino IDE

Arduino IDE supports multiple languages itself. We only require choosing the desired language and restart IDE. E.g. Chinese (China)

Open **File->Preferences->Editor Language**



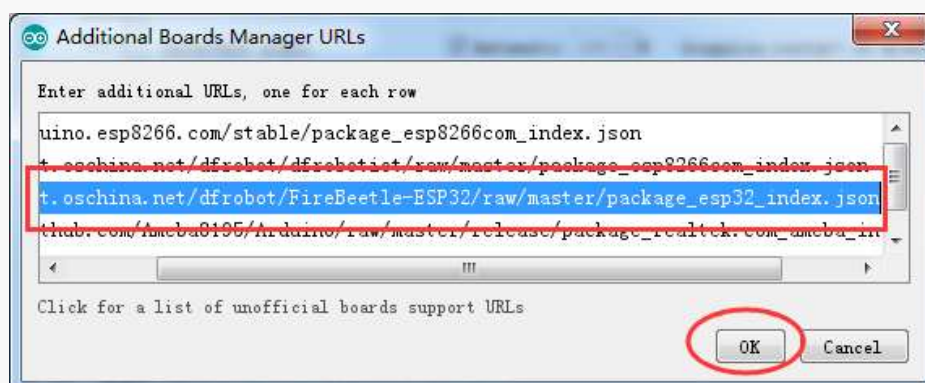
STEP3: install the core center of the FireBeetle Board-ESP32.

Arduino IDE installation package does not contain the core center of the FireBeetle Board-ESP32. Therefore, we need to add it manually. To add FireBeetle Board-ESP32 we should add the core center of the FireBeetle Board-ESP32 firstly by hand in the Arduino development manager.

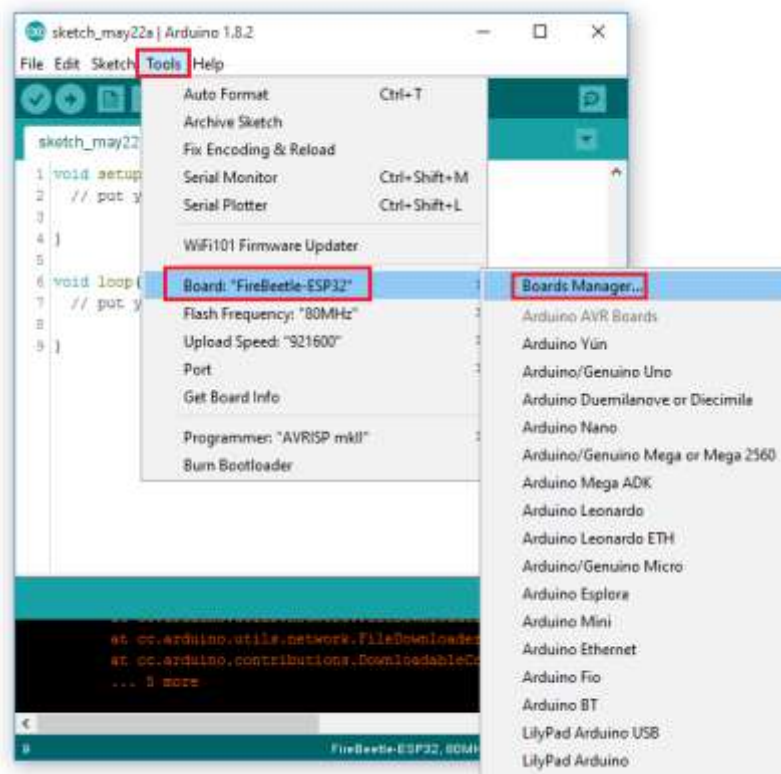
A. Open **file-> Preferences**, Copy the URL below to the **Additional Boards Manager URLs** and click OK to complete the setup.

https://git.oschina.net/dfrobot/FireBeetle-ESP32/raw/master/package_esp32_index.json

Caution: If other URL has been put, please click the bottom  and add the URL, seen as below.



B. Open **tools-> Boards-> Boards Manager**



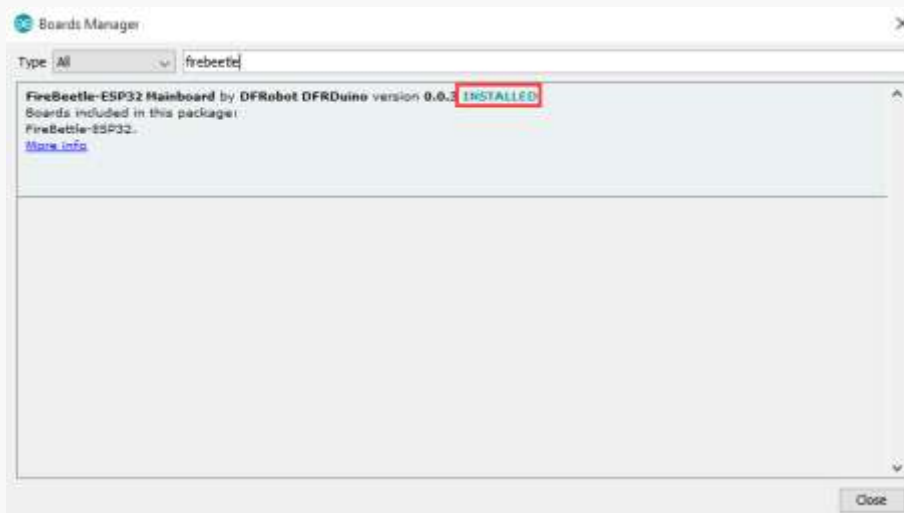
Input **FireBeetle** and wait for the loading in **the board manager**.

FireBeetle 0.03 is the right one. You can also choose the latest version and click Install.

It will cost 5 to 10 minutes in average.

The FireBeetle-ESP32 Mainboard will be marked with **INSTALLED** after the installation as shown below.

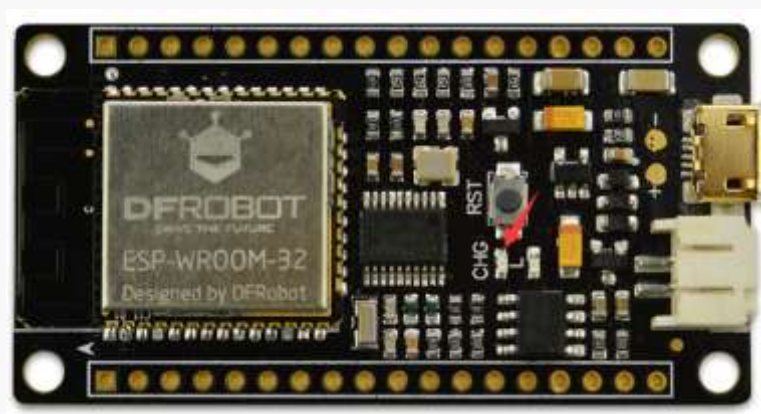




Caution: The network barrier will lead to installation blocking. Then you should restart Addino IDE forcefully and redo the steps before. Or you should speed up the network with proxy software to accomplish the installation. Moreover, some key processes may be blocked by the firewall and antivirus software. That's why you should authorize it to change and add it to the write page.

STEP4: connect FireBeetle Board-ESP32 to your computer.

When we installed the Arduino IDE and the core center of the FireBeetle Board-ESP32 successfully, FireBeetle Board-ESP32 can be connected to computer by USB data wire. If the operation is right, CHG power light of the FireBeetle Board-ESP32 will flash to check whether the lithium battery has been applied.

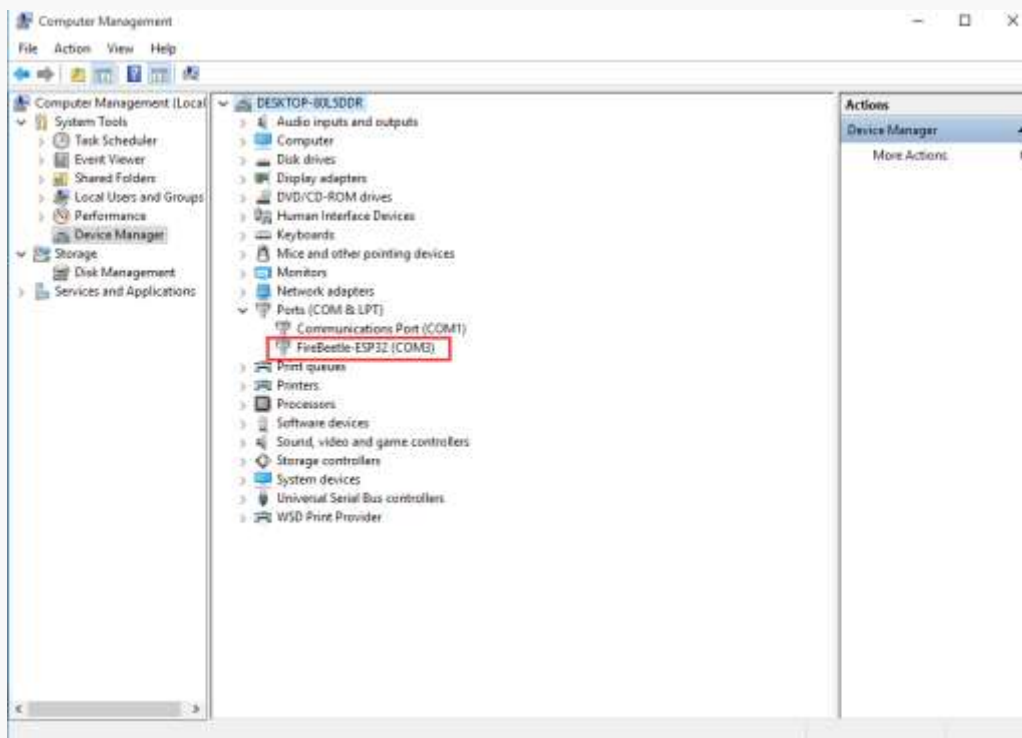


You require confirming the developing board has been recognized by the computer and which COM has been connected (for COM to COM interaction).

The following steps will help you to confirm.

Open **Control panel-> Device Manager->Ports (COM&LPT)**.

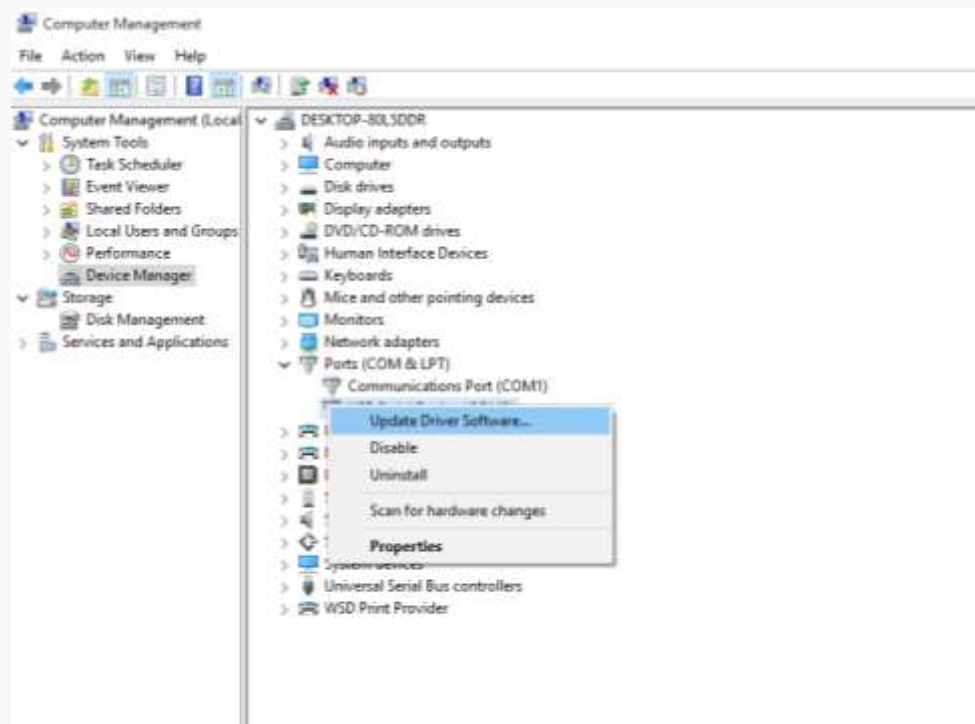
The COM connects to FireBeetle Board-ESP32 shown in the list as below (Here is COM3)



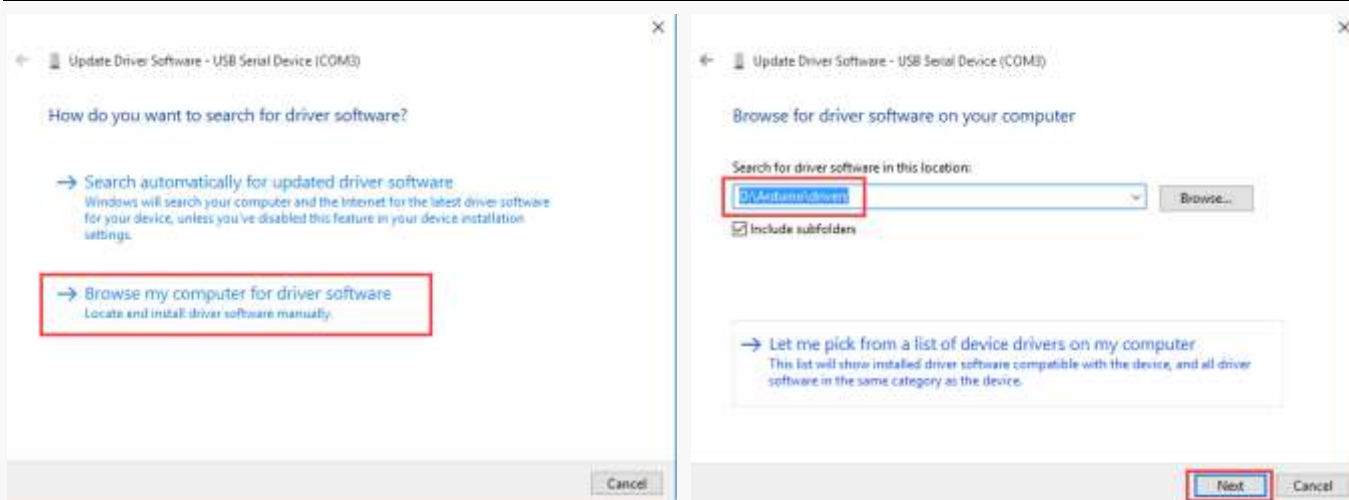
If the device cannot be recognized, you should download FireBeetle Board-ESP32 driver to the computer.

Download link: <https://github.com/Chocho2017/FireBeetle-Board-ESP32.git>

Save the drive file FireBeetle-ESP32.inf to anywhere in your computer, right click on **FireBeetle-ESP32** and choose **update driver** as below.



Browse the computer to find out the driver software and input file catalog of FireBeetle-ESP32.inf saved before, click Next to continue, shown as below.



Caution: FireBeetle-ESP32.inf file should be saved into the drivers' file of Arduino IDE.

And follow the prompts to fulfill the installation of drive file.

STEP5: program in the Arduino IDE.

After the installation of Arduino IDE software, operating it and open the processing window. In this window, you can edit and upload codes, and use the built-in serial monitor to the communication between serial and development board. Now let's check the Arduino IDE interface carefully.



Different from common C language programming, a Arduino programming consists of `void setup()` and `void loop()`.

'void setup()', provided for initialization programs. It only runs once when the development board power-on.

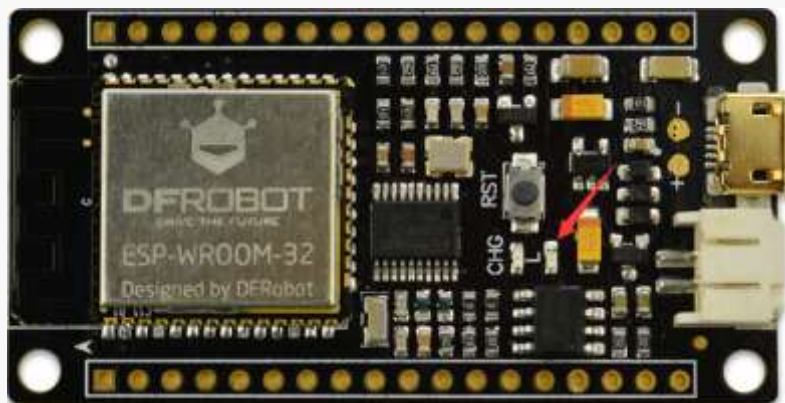
'void loop ()', provided for repeatable programs. These repeatable programs of the board contribute to real-time interaction with the externals.

The write area in the image is for programming. The black area is information window, shows the information of uploading/ verifying /compiling.

You can set information related by click [File->Page Setup](#).

STEP6: upload codes to FireBeetle-ESP32.

In this step, an example process **Blink** of FireBeetle Board-ESP32 will be represented to you. **Blink** controls the LED light of D9 flash in every one second. Same as many Arduino boards, FireBeetle Board-ESP32 has a LED light, D9. Therefore, no other peripheral object is needed. LED's status indicator can be found in the mother board of FireBeetle Board-ESP32.



Please confirm the accuracy of the codes in priority before uploading. Please click [verify/compile](#) to confirm.

```
Arduino IDE - Blink | Arduino 1.8.2
File Edit Sketch Tools Help

Blink.g
20 modified 3 Sep 2016
21 by Chinyee Huang
22
23
24 // the setup function runs once when you press reset or power the board
25 void setup() {
26   // initialize digital pin LED_BUILTIN as an output.
27   pinMode(LED_BUILTIN, OUTPUT);
28 }
29
30 // the loop function runs over and over again forever
31 void loop() {
32   digitalWrite(LED_BUILTIN, HIGH); // turn the LED on (HIGH is the voltage level)
33   delay(1000); // wait for a second
34   digitalWrite(LED_BUILTIN, LOW); // turn the LED off by making the voltage LOW
35   delay(1000); // wait for a second
36 }

'OUTPUT1' was not declared in this scope
Copy error messages
exit status 1
'OUTPUT1' was not declared in this scope
FireBeetle-ESP32: 80MHz, 0.16MB on 0.0s
```

Caution: the example code of Blink is in File>Examples>Basics>Blink.

A message of **Done compiling** will be displayed in the information window after few seconds if your operation is right. The message showed a successful compilation. Please backup to check whether the processing is completed if any error.

```

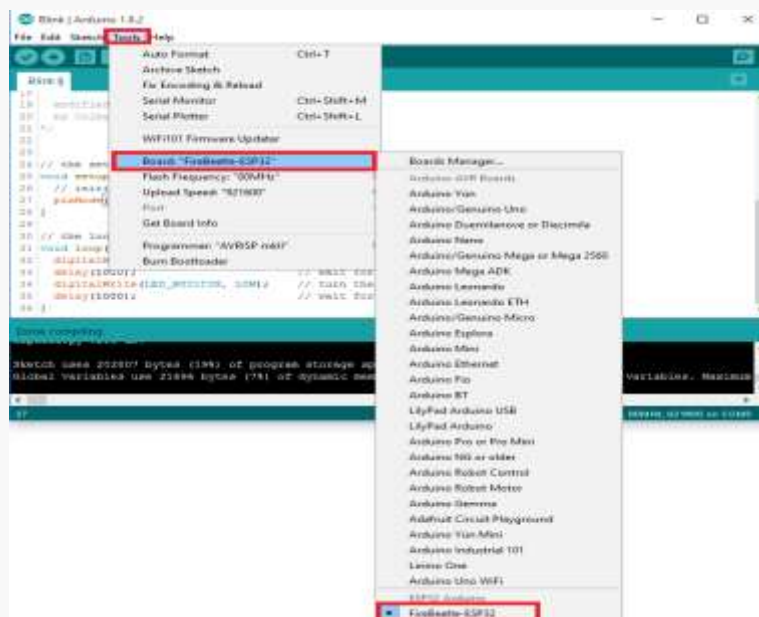
18 // verified 8 Sep 2018
19 // by Gung Deenan
20 //
21 //
22 //
23 //
24 // the setup function runs once when you press reset or power the board
25 void setup() {
26   // initialize digital pin LED_BUILTIN as an output.
27   pinMode(LED_BUILTIN, OUTPUT);
28 }
29 //
30 // the loop function runs over and over again forever
31 void loop() {
32   digitalWrite(LED_BUILTIN, HIGH); // turn the LED on (HIGH is the voltage level)
33   delay(1000); // wait for a second
34   digitalWrite(LED_BUILTIN, LOW); // turn the LED off by making the voltage LOW
35   delay(1000); // wait for a second
36 }

```

Done compiling

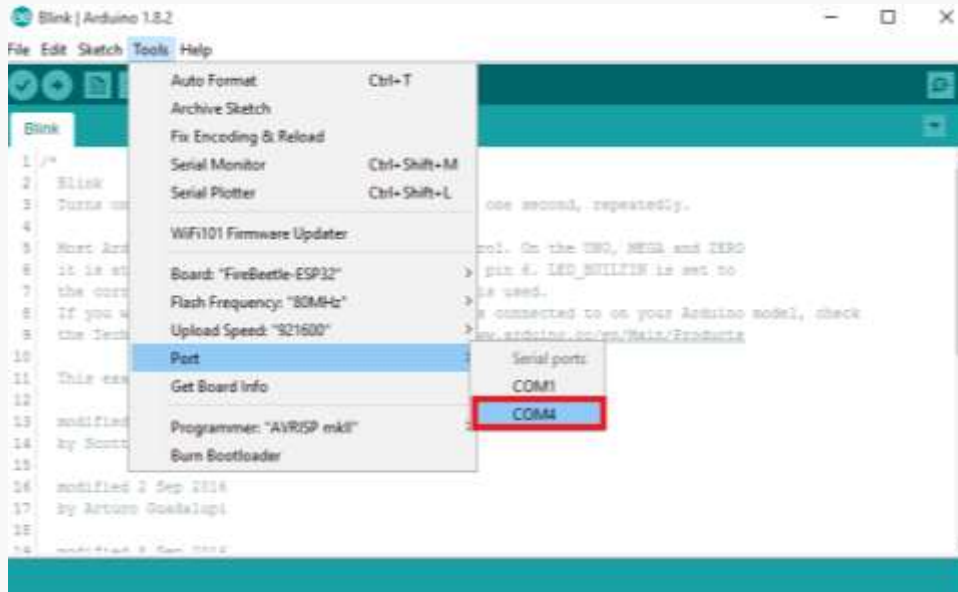
Sketch uses 302107 bytes (19%) of program storage space. Maximum is 1544464 bytes.
Global variables use 21896 bytes (7%) of dynamic memory, leaving 273216 bytes for local variables. Maximum

After the programming, please choose **Tools->Board-> FireBeetle-ESP32**. In general, this step performs only once on first use if you would not change the board.



Switch to **Tools->Port**, according to the number of serial port shown in the FireBeetle Board-ESP32, we should choose **COM3** as the communication port. The communication port appears only when the development board has connected to a computer and been successfully recognized. We should confirm the communication port carefully before every uploading, for a development board may occupy different ports after insertion and extraction.

After the choice of COM, the information of board and port shows in **the lower right of the window**.



Finally, clicking upload to burn the program onto the FireBeetle Board-ESP32.



If you upload it successfully, message **Done uploading** would be presented in the information window. Meanwhile, D9 starts to flash.



In short, there are three steps to upload codes for Arduino.

- Compile the code.
- Choose the version of development board and port.
- Upload

Above are primary methods of how to use FireBeetle Board-ESP32 in the Arduino IDE. Please feel free to our forum to contact us if you have any enquiry or suggestion.

Caution: developers are the most common users of FireBeetle Board-ESP32 and not all peripherals have examples for reference, for there is some bugs to be discovered and repaired. IO/I2C/SPI can be used directly in the Arduino IDE now and we are making efforts to explore more.

Link to the forum: <http://www.dfrobot.com.cn/community/forum.php>

Welcome to our community! Many detailed tutors and fantastic projects are waiting for you! You can share your projects and idea with other forum members. Welcome to DFRobot!

Link to the DFRobot: <http://www.dfrobot.com.cn>

Chapter 3: The Basics of Using Peripherals in the FireBeetle Board-ESP32

This chapter covers the basics of how to use peripherals in the FireBeetle Board-ESP32 by some example projects. You can also modify the example projects below to finish your own projects. The FireBeetle Board-ESP32 mainly includes UART; I2C; SPI; ADC; PWM; DAC; and integrated hall sensor etc.

Project1: The experiment of Serial Port

In the first chapter, we have uploaded Blink, a flash program to test the LED light in the board. Now, let's print the timing data in every second with UART pin.

Components in need:

1 x FireBeetle Board-ESP32



Hardware linking method:

No need for other sensor. Please connect FireBeetle Board-ESP32 directly to your computer by USB data wire.

Enter code

You would better enter the code manually than open **Course->Item-1** to be familiar with it.

Sample code:

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

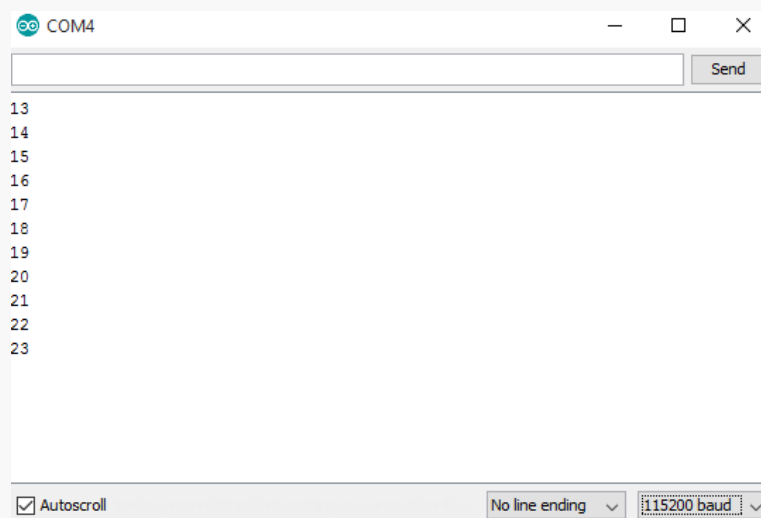
```
}  
void loop() {  
    static unsigned long i = 0;  
    Serial.println(i++);  
    delay(1000);  
}
```

Please click [verify/compile](#) to review and please upload it after the confirmation. Once you click [Upload](#), IDE will send codes to FireBeetle Board-ESP32.

Caution: please referring the previous tutorial if you forget the way to verify/compile/upload. The example is Ttem-1 of the file: Course.

The phenomena

Once the upload finishes, please open the built-in serial monitor of Arduino IDE, print message shown as below.



Project2: PWM (Breathing light)

PWM, let FireBeetle Board-ESP32 drive LED light by PWM, realizing light gradient overlay which is similar to the action of breathing. Please refer to knowledge expand for the explanation of PWM.

Components in need

1 x FireBeetle Board-ESP32



Hardware linking method

No need for other sensor, L LED light in the FireBeetle Board-ESP32 is available. Please connect FireBeetle Board-ESP32 directly to your computer by USB data wire.

Enter code

Open Arduino IDE. You would better enter the code manually than open **Course->Item-2** to be familiar with it.

Sample code:

```
//use first channel of 16 channels (started from zero)
#define LEDC_CHANNEL_0    0

// use 13 bit precision for LEDC timer
#define LEDC_TIMER_13_BIT 13

// use 5000 Hz as a LEDC base frequency
#define LEDC_BASE_FREQ    5000

//Set LED light
```

```
#define LED_PIN          D9

int brightness = 0;    // how bright the LED is
int fadeAmount = 5;   // how many points to fade the LED by

// value has to be between 0 and valueMax
void ledcAnalogWrite
(uint32_t value, uint32_t valueMax = 255) {
    // calculate duty
    uint32_t duty = (LEDC_BASE_FREQ / valueMax) * min(value, valueMax);
    // write duty to LEDC
    ledcWrite(LEDC_CHANNEL_0, duty);
}

void setup() {
    ledcSetup(LEDC_CHANNEL_0, LEDC_BASE_FREQ, LEDC_TIMER_13_BIT);
    ledcAttachPin(LED_PIN, LEDC_CHANNEL_0);
}

void loop() {
    ledcAnalogWrite(brightness);
    brightness += fadeAmount;

    if (brightness <= 0 || brightness >= 255) {
        fadeAmount = -fadeAmount;
    }
    delay(30);
}
```

Please click [verify/compile](#) to review and please upload it after the confirmation.

Once you click [Upload](#), IDE will send code to FireBeetle Board-ESP32.

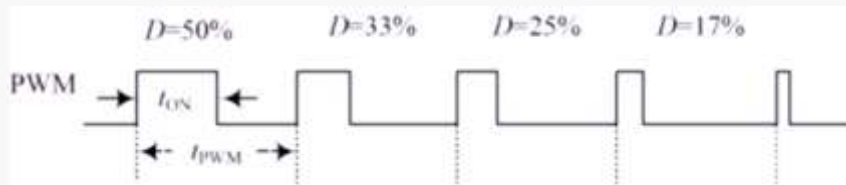
You can see the breath of **L** LED light when you uploaded the codes.

Let's review the codes and hardware and check the way they work.

Knowledge learning

What is PWM (pulse-width modulation) ?

Pulse-width modulation (PWM), or pulse-duration modulation (PDM) is a modulation technique used to encode messages into a pulsing signal. MCU (Microcontroller) controls the on-off of switching devices and get a series of pulses has equally amplitude. And use these pulses to replace sinusoid or other waveforms in need, shown as below.



t_{ON} is the duration of high-level voltage.

t_{PWM} is the period of PWM wave.

$t_{PWM}-t_{ON}$ is the duration of low-level voltage.

Duty cycle is the ratio of the duration of high-level voltage in the whole period of PWM wave:

$$D = t_{on} / t_{PWM} \circ$$

Code analysis

PWM of FireBeetle Board-ESP32 is advanced than ordinary Arduino UNO. Simple analogWrite function cannot drive PWM in setting while it can be drove by the setting of timer function and corresponding frequency parameters etc.

```
#define LEDC_CHANNEL_0 0
```

Defining the channel used for timer. There are 16 channels in the FireBeetle Board-ESP32. Here we choose channel 0.

```
#define LEDC_TIMER_13_BIT 13
```

Defining that the timer is 13 bit, the crest value is the thirteenth of 2.

```
#define LEDC_BASE_FREQ 5000
```

Set the frequency of timer, the unit is Hz. Brightness parameter shows the duty cycle and fade-Amount parameter shows the number in every change.

```
void ledcAnalogWrite (uint32_t value, uint32_t valueMax = 255)
```

The function counts and set PWM duty cycle, similar to analogWhite of Arduino. You can see the crest value of passing parameter is 255 to make it compatible with analogWhite.

```
ledcSetup (LEDC_CHANNEL_0, LEDC_BASE_FREQ, LEDC_TIMER_13_BIT);
ledcAttachPin (LED_PIN, LEDC_CHANNEL_0);
```

The two codes are the setting functions of FireBeetle Board-ESP32 timer. We need to know how to use functions mentioned before to set timer. Please check base code if you are interested in the function prototypes ([the link of base code: C:\Users\your-PC\AppData\Local\Arduino15\packages\esp32\hardware\DFRobot_FireBeetle-ESP32\0.0.3\libraries\ESP32\](C:\Users\your-PC\AppData\Local\Arduino15\packages\esp32\hardware\DFRobot_FireBeetle-ESP32\0.0.3\libraries\ESP32\)).

About what is PWM, please refer to the previous Knowledge study for PWM.

Caution: Every pin in FireBeetle Board-ESP32 can be configured to PWM output end. You can try changing codes to finish your own projects.

Project3 ADC

ADC, an analog-to-digital converter, shorthand as A/D converter, can convert analog signal to digital signal. There are 12 bits of FireBeetle Board-ESP32 and its crest output value is 4095. There are 10 bits of Arduino UNO and its crest output value is 1023. In comparison, FireBeetle Board-ESP32 has higher precision, faster conversion rate than Arduino UNO. Moreover, FireBeetle Board-ESP32 is fully compatible with Arduino analogRead function, just read it.

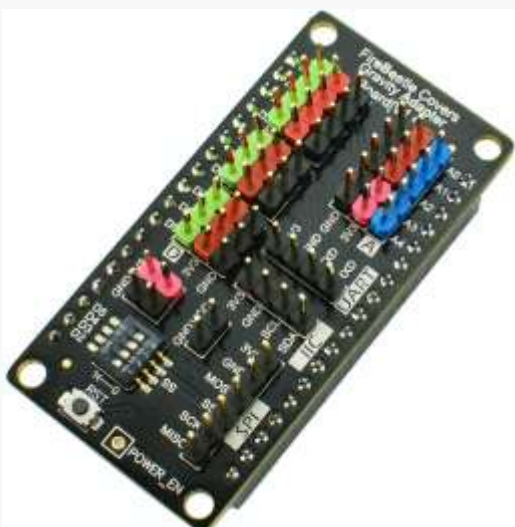
Components in need

1 x simulative angle sensor



Caution: Please use Gravity port of the sensor to get more convenient experience, plug and play. Please [click the link](#) to buy on line.

1 x FireBeetle Covers-Gravity Adapter Board



Caution: FireBeetle Covers-Gravity Adapter Board is better for freshman to do ADC, plug and play.

1 x FireBeetle Board-ESP32



Hardware linking method

Plug FireBeetle Covers-Gravity Adapter Board to FireBeetle Board-ESP32 and plug simulative angle sensor into A0 here in the ADC experiment.

And connect FireBeetle Board-ESP32 directly to your computer by USB data wire.

With codes as follows

Open Arduino IDE. You would better enter the code manually than open **Course-> Item-3** to be familiar with it.

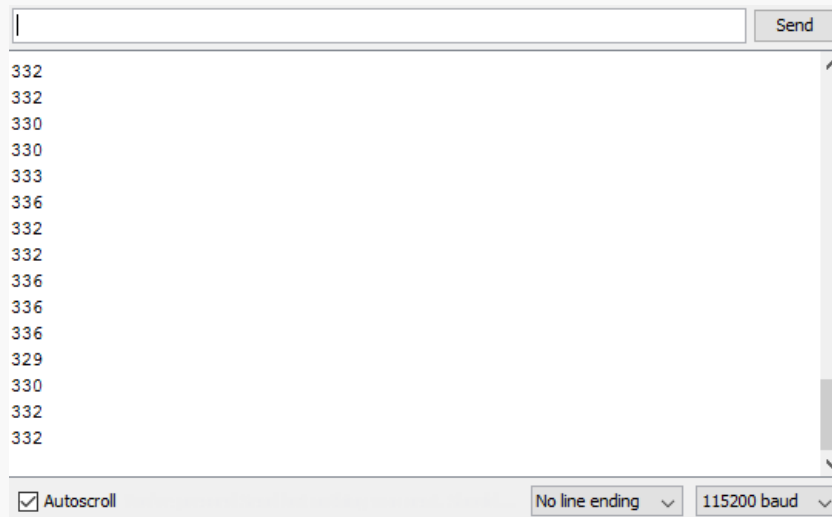
With codes as follows

```
void setup() {  
  // put your setup code here, to run once:  
  Serial.begin(115200);  
}  
  
void loop() {  
  // put your main code here, to run repeatedly:  
  Serial.println(analogRead(A0));  
  delay(100);  
}
```

Please click **verify/compile** to review and please upload it after the confirmation.

Once you click **Upload**, IDE will send codes to FireBeetle Board-ESP32.

Once the upload finishes, open the built-in serial monitor of Arduino IDE, rotate simulative angle sensor, you can see the movement of the numbers, shown as below.



Caution: Please referring the previous tutorial if you forget the way to verify/compile/upload.

Code analysis

Because ADC of FireBeetle Board-ESP32 is fully compatible with Arduino analogRead function, no more explanation of analogRead function.

Caution: If you are not familiar to the basics of Arduino function, please [click the link](#) to learn more.

Project4 I2C

I2C of FireBeetle Board-ESP32 can be set to all I/O and allocated by uploading relative data. We have already set the default configuration to make it more convenient to use. It is fully compatible with Arduino. Look up the Chapter1, the introduction, the default pins. The project based on the default configuration of I2C to drive OLED display screen.

Components in need

1 x Gravity I2C OLED-2864 Display Screen



Caution: Please use Gravity port of the sensor to get more convenient experience, plug and play.

1 x FireBeetle Covers-Gravity Adapter Board



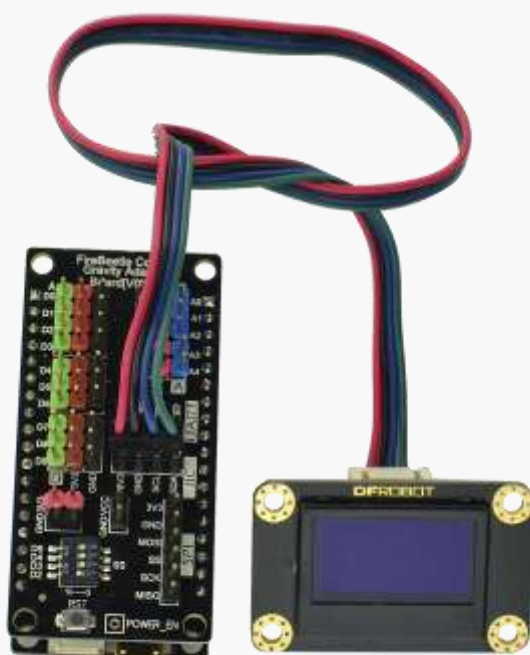
Caution: FireBeetle Covers-Gravity Adapter Board is better for freshman to do ADC, plug and play.

1 x FireBeetle Board-ESP32



Hardware connection

Plug FireBeetle Covers-Gravity Adapter Board to FireBeetle Board-ESP32 and plug Gravity I2C OLED-2864 Display Screen into I2C here in the I2C experiment, shown as below.



Then connect FireBeetle Board-ESP32 directly to your computer by USB data wire.

Enter code

Above all, we should know the output value range of simulative angle sensor for further ease of use the data mapping. Open Arduino IDE. You would better enter the code manually than open **Course->Item-4** to be familiar with it.

With codes as follows

```
#include <Wire.h>

int UG2864Address = 0x3C; //OLED UG2864 7 bit address of the device

unsigned char show2[] = {
    0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00, //*page0
    0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00, //
    0x08,0xF8,0x08,0x08,0x08,0x10,0xE0,0x00,0x08,0xF8,0x88,0x88,0xE8,0x08,0x00,0x00, //DF
    0x08,0xF8,0x88,0x88,0x88,0x88,0x70,0x00,0x00,0x00,0x80,0x80,0x80,0x80,0x00,0x00, //Ro
    0x08,0xF8,0x00,0x80,0x80,0x00,0x00,0x00,0x00,0x00,0x80,0x80,0x80,0x80,0x00,0x00, //bo
    0x00,0x80,0x80,0xE0,0x80,0x80,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00, //t
    0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00, //
    0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00, //
    0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00, //*page1
    0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00, //
    0x20,0x3F,0x20,0x20,0x20,0x10,0x0F,0x00,0x20,0x3F,0x20,0x00,0x03,0x00,0x00,0x00, //DF
    0x20,0x3F,0x20,0x00,0x03,0x0C,0x30,0x20,0x00,0x1F,0x20,0x20,0x20,0x20,0x1F,0x00, //Ro
    0x00,0x3F,0x11,0x20,0x20,0x11,0x0E,0x00,0x00,0x1F,0x20,0x20,0x20,0x20,0x1F,0x00, //bo
    0x00,0x00,0x00,0x1F,0x20,0x20,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00, //t
    0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00, //
    0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00, //
    0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00, //*page2
    0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00, //
    0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00, //
    0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00, //
    0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00, //
    0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00, //
    0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00, //
    0x80,0x80,0x00,0x80,0x00,0x80,0x80,0x80,0x80,0x80,0x00,0x80,0x00,0x80,0x80,0x80, //*page3
    0x80,0x80,0x00,0x80,0x00,0x80,0x80,0x80,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x80, //
    0x80,0x88,0xF8,0x00,0x00,0x80,0x80,0xF0,0x88,0x88,0x88,0x18,0x80,0x80,0x80,0x00, //
    0x80,0x80,0x80,0x00,0x00,0x00,0x80,0x80,0x80,0x80,0x00,0x00,0x08,0xF8,0x00,0x80, //ro
    0x80,0x00,0x00,0x00,0x00,0x00,0x80,0x80,0x80,0x80,0x00,0x00,0x00,0x80,0x80,0xE0, //bo
    0x80,0x80,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x80,0x80,0x80,0x00,0x00, //t.c
    0x00,0x00,0x80,0x80,0x80,0x80,0x00,0x00,0x80,0x80,0x80,0x80,0x80,0x80,0x80,0x00, //om
    0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00, //
    0x0F,0x30,0x0C,0x03,0x0C,0x30,0x0F,0x00,0x0F,0x30,0x0C,0x03,0x0C,0x30,0x0F,0x00, //*page4
    0x0F,0x30,0x0C,0x03,0x0C,0x30,0x0F,0x00,0x00,0xc0,0xc0,0x00,0x00,0x0E,0x11,0x20, //
    0x20,0x10,0x3F,0x20,0x00,0x20,0x20,0x3F,0x20,0x20,0x00,0x00,0x20,0x20,0x3F,0x21, //
    0x20,0x00,0x01,0x00,0x00,0x1F,0x20,0x20,0x20,0x20,0x1F,0x00,0x00,0x3F,0x11,0x20, //ro
    0x20,0x11,0x0E,0x00,0x00,0x1F,0x20,0x20,0x20,0x20,0x1F,0x00,0x00,0x00,0x00,0x1F, //bo
    0x20,0x20,0x00,0x00,0x00,0xc0,0xc0,0x00,0x00,0x0E,0x11,0x20,0x20,0x20,0x11,0x00, //t.c
    0x00,0x1F,0x20,0x20,0x20,0x20,0x1F,0x00,0x20,0x3F,0x20,0x00,0x3F,0x20,0x00,0x3F, //om
    0x00,0x00,0xc0,0xc0,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00, //
}
```



```
Writec(0XA1); //set segment remap column address 127 is mapped to SEG0
Writec(0XA6); //normal / reverse normal display
Writec(0XA8); //multiplex ratio
Writec(0X3F); //1/64
Writec(0XC8); //Com scan direction remapped mode. Scan from COM[N-1] to COM0
Writec(0XD3); //set display offset
Writec(0X00);
Writec(0XD5); //set osc division
Writec(0X80);
Writec(0XD9); //set pre-charge period
Writec(0X11);
Writec(0XDa); //set COM pins
Writec(0X12);
Writec(0X8d); /*set charge pump enable*/
Writec(0X14);
Writec(0Xdb); //Set VcomH
Writec(0X20);
Writec(0XAF); //display ON
}
```

```
void fill(unsigned char dat){
    unsigned char i,j;
    Writec(0x00);
    Writec(0x10);
    Writec(0xB0);
    for(j=0;j<8;j++){
        Writec(0xB0+j);
        Writec(0x00);
        Writec(0x10);
        for(i=0;i<128;i++){
            Writed(dat);
        }
    }
}
```

```
void show(){
    unsigned char x,y;
    unsigned int j=0;

    Writec(0x00); //set lower column address
    Writec(0x10); //set higher column address

    for(y=0;y<8;y++){
        Writec(0xB0+y);
        Writec(0x00);
        Writec(0x10);
        for(x=0;x<128;x++){
            Writed(show2[j++]);
        }
    }
}
```



```
    }  
}  
  
void setup() {  
  Wire.begin();  
  SSD1306();  
  fill(0xff); //light up the screen  
  delay(100);  
  fill(0x00); //clean up the screen  
  delay(100);  
}  
  
void loop() {  
  show();  
}
```

Please click [verify/compile](#) to review and please upload it after the confirmation.

Once you click [Upload](#), IDE will send codes to FireBeetle Board-ESP32.

Once the upload finishes, the Gravity I2C OLED-2864 Display Screen will show 'DFRobot [www.dfrobot.com.cn](#)', shown as below.



Code analysis

The reason why more code in this project is that the direct drive to the register of the Gravity I2C OLED-2864 Display Screen with I2C communication. Please [click the link](#) or visit DFRobot Wiki on line for advanced uses of the Gravity I2C OLED-2864 Display Screen.

```
void Writec(unsigned char COM)
```

Set the function of register by setting the Gravity I2C OLED-2864 Display Screen of I2C communication. And the usage method of I2C is fully compliant with Arduino.

```
void Writed(unsigned char DATA)
```

Write data function. The usage method of I2C is fully compliant with Arduino.

Caution: I2C of FireBeetle Board-ESP32 is fully compliant with Arduino, mainly by calling Wire files.

Project5 SPI

SPI communication is widely used by sensors, for the higher rate than I2C and no address conflict. SPI, a communication bus of high speed, full-duplex, synchrony. SPI of FireBeetle Board-ESP32 can be allocated to all I/O. You can read underlying code to practice (not recommend to freshmen). For better user experience, SPI of FireBeetle Board-ESP32 allocated IO18/IO19/IO23 by default, fully compatible with usage method of Arduino.

The project uses FireBeetle Board-ESP32 to read data of the temperature and humidity sensor BME280. The example uses library file of BME280.

Please read the BME280 library file for SPI driver knowledge.

Please [click here](#) to download the library file of BME280.

Components in need

1 × BME280 temperature and humidity sensor



Caution: BME280 support communication between I2C and SPI itself. We use SPI here.

1 × FireBeetle Covers-Gravity Adapter Board



Caution: FireBeetle Covers-Gravity Adapter Board is better for freshman to do ADC, plug and play.

1 x FireBeetle Board-ESP32



Enter code

Open Arduino IDE. You would better enter the code manually than open **Course-> Item-5** to be familiar with it.

Sample code:

```
#include <DFRobot_BME280.h>

#define SEA_LEVEL_PRESSURE 1013.25f
#define BME_CS D2

DFRobot_BME280 bme(BME_CS); //SPI

float temp, pa, hum, alt;

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

  // I2c default address is 0x77, if the need to change please modify bme.begin(Addr)
  if (!bme.begin()) {
    Serial.println("No sensor device found, check line or address!");
    while (1);
  }

  Serial.println("-- BME280 DEMO --");
}

void loop() {
  temp = bme.temperatureValue();
  pa = bme.pressureValue();
```

```
hum = bme.altitudeValue(SEA_LEVEL_PRESSURE);
alt = bme.humidityValue();

Serial.print("Temp:");
Serial.print(temp);
Serial.println(" °C");

Serial.print("Pa:");
Serial.print(pa);
Serial.println(" Pa");

Serial.print("Hum:");
Serial.print(hum);
Serial.println(" m");

Serial.print("Alt:");
Serial.print(alt);
Serial.println(" %");


Serial.println("-----END-----");

delay(1000);
}
```

IDE Please click [verify/compile](#) to review and please upload it after the confirmation.

Once you click [Upload](#), IDE will send codes to FireBeetle Board-ESP32.

Once the upload finishes, open the built-in serial monitor of Arduino IDE, print message shown as below.

A screenshot of the Arduino IDE serial monitor window. The window displays three lines of sensor data, each followed by a separator line. The data points are: 1) Temp: 33.00 °C, Pa: 94922.47 Pa, Hum: 547.23 m, Alt: 72.19 %; 2) Temp: 33.13 °C, Pa: 94915.52 Pa, Hum: 547.85 m, Alt: 73.14 %; 3) Temp: 33.25 °C, Pa: 94920.27 Pa, Hum: 547.43 m, Alt: 74.04 %. The separator lines are labeled "-----EIND-----". The serial monitor has a vertical scrollbar on the right side.

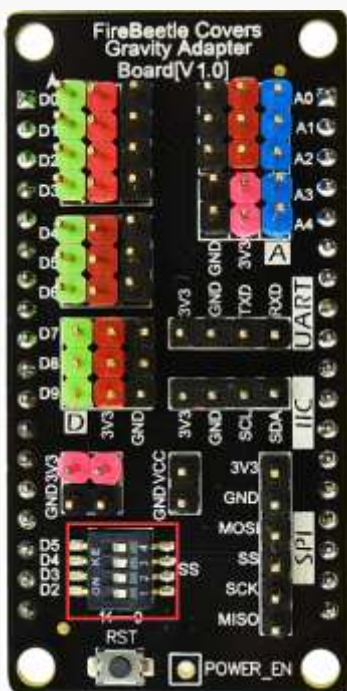
```
Temp:33.00 °C
Pa:94922.47 Pa
Hum:547.23 m
Alt:72.19 %
-----EIND-----
Temp:33.13 °C
Pa:94915.52 Pa
Hum:547.85 m
Alt:73.14 %
-----EIND-----
Temp:33.25 °C
Pa:94920.27 Pa
Hum:547.43 m
Alt:74.04 %
-----EIND-----
```

Code analysis

The project mainly uses library file of BME280. Although there is no operation to the underlying of SPI, SPI of FireBeetle Board-ESP32 is fully compliant with Arduino.

```
#define BME_CS D2
```

Different from Arduino, the setting of pin to SPI should transmit D2 not 2. What is more, D2, the SS dial switch of FireBeetle Covers-Gravity Adapter Board should be gated. And BME280 should be connected to SPI interface of FireBeetle Covers-Gravity Adapter Board by Dupont line, shown as below.



Project6: Hall Sensor

The hall sensor integrated by FireBeetle Board-ESP32 is based on N-carrier. Putting the chip into the electromagnetic field, a low voltage came into the resistance. The voltage not just can be directly collected and measured by ADC, but also can be measured by ADC after the amplification of the pre-analogue amplifier with super-low noise.

The experiment is based on the hall sensor of FireBeetle Board-ESP32. When the magnet move close to the front board (with a logo), the negative number been printed, closer to the magnet, lower the number. When the magnet moves close to the opposite one, printed the positive number, closer to the magnet, higher the number.

Components in need

1 x FireBeetle Board-ESP32



Caution: No need for other sensor.

Enter code

Open Arduino IDE. You would better enter the code manually than open **Course-> Item-6** to be familiar with it.

Sample code:

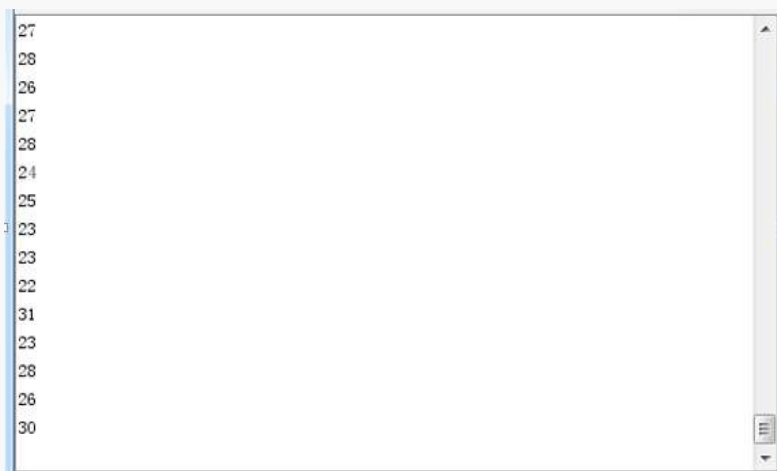
```
void setup() {  
  // put your setup code here, to run once:  
  Serial.begin(115200);  
}
```

```
void loop() {  
  // put your main code here, to run repeatedly:  
  Serial.println(hallRead());  
  delay(100);  
}
```

Please click [verify/compile](#) to review and please upload it after the confirmation.

Once you click [Upload](#), IDE will send codes to FireBeetle Board-ESP32.

Once the upload finishes, open the built-in serial monitor of Arduino IDE, print message shown as below.



Caution: Change the distance of the magnet to FireBeetle Board-ESP32, you can see the variety of the printed numbers.

Code analysis

The hall sensor used in this project is integrated by FireBeetle Board-ESP32. The bottom uses IO36 and IO39 to read the voltage of the hall sensor (Please refer to source if you are interested in the bottom). We only need to call the function *hallRead*.

```
hallRead()
```

The return type of digital signal value got from the hall sensor is int.

You can try modifying the code to do your own project.

Project7 DAC

Opposite to ADC of Project3, DAC converts digital signal to analog signal. FireBeetle Board-ESP32 integrated two 8-bit DAC passing ways to convert two signals respectively. DAC circuit is composed of built-in series resistor and a buffer. Moreover, the two independent DAC can be used as both reference voltage and power supply for other circuits. The project will explain the way to input trapezium wave with DAC.

Components in need

1 x FireBeetle Board-ESP32



Caution: No need for other sensor.

Enter code

Open Arduino IDE. You would better enter the code manually than open **Course-> Item-7** to be familiar with it.

Sample code:

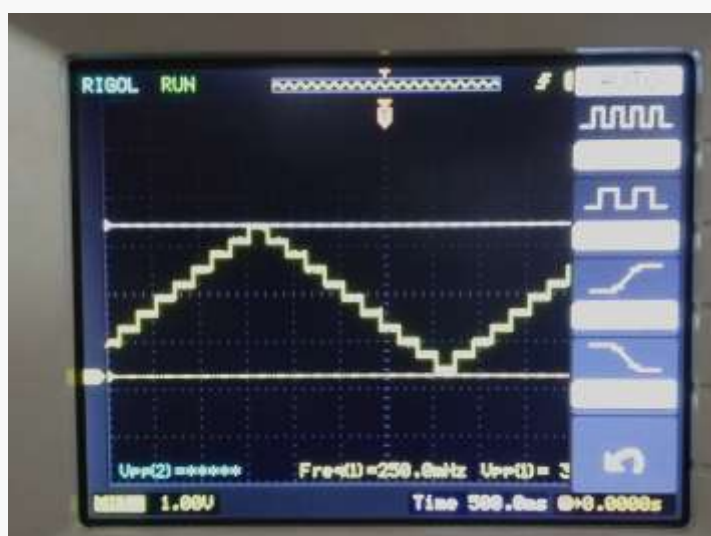
```
void setup() {  
  }  
  
void loop() {  
  // put your main code here, to run repeatedly:  
  for(int i=0;i<10;i++){  
    dacWrite(D2,i*25);  
    delay(200);  
  }  
}
```

```
}  
for(int j=10;j>0;j--){  
  dacWrite(D2,j*25);  
  delay(200);  
}  
}
```

Please click [verify/compile](#) to review and please upload it after the confirmation.

Once you click [Upload](#), IDE will send codes to FireBeetle Board-ESP32.

Once the upload finishes, use an oscilloscope to test the voltage of D2, trapezium wave shown as below.



Code analysis:

As we see, the operation of DAC in Arduino is convenient, just call `dacWrite`. The antitype of `dacWrite` is as below:

```
void dacWrite(uint8_t pin, uint8_t value)
```

Pin is the digital output interface of DAC, D2/D3; Value is the output number, from 0 to 255, the corresponding voltage value is 0 to V_{cc} .

Project8 Touch Sensor

FireBeetle Board-ESP32 offers as many as 10 GPIO capacitance sensors. It can probe the difference of capacitance caused by the direct access or approach with hands or others. The characteristic of the low noise and high sensitivity works for relatively smaller touch board. It can be directly used to touch switch items.

The project will explain the way to obtain and print the state of FireBeetle Board-ESP32s' capacitance sensors with Arduino code.

Components in need

1 x FireBeetle Board-ESP32



Caution: No need for other sensor.

Enter code

Open Arduino IDE. You would better enter the code manually than open **Course-> Item-8** to be familiar with it.

Sample code:

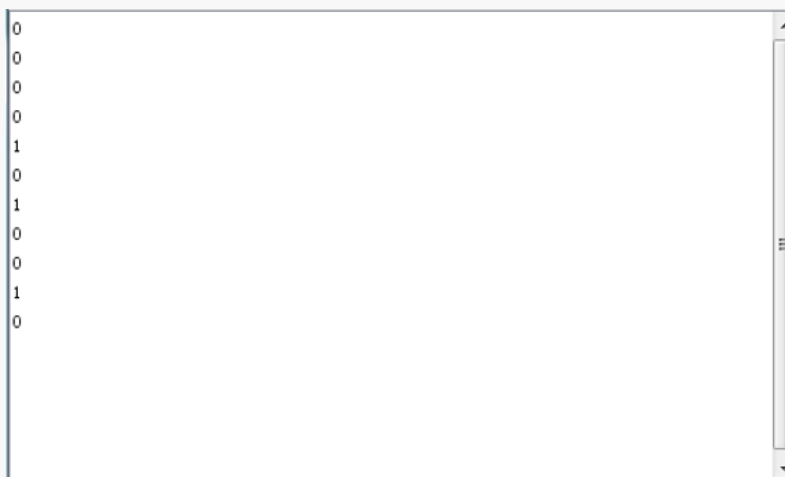
```
void setup()
{
  Serial.begin(115200);
  delay(1000); // give me time to bring up serial monitor
  Serial.println("FireBeetle Board-ESP32 Touch Test");
}

void loop(){
  Serial.println(touchRead(T2)); // get value using T0->D9
```

```

delay(1000);
}
    
```

Please click [verify/compile](#) to review and please upload it after the confirmation. Once you click [Upload](#), IDE will send codes to FireBeetle Board-ESP32. Once the upload finishes, open the built-in serial monitor of Arduino IDE, touch D9 with your hand (T2 corresponds to D9) print 1 shown as below.



Code analysis:

Obtaining GPIO state of the capacitance sensors only requiring calling touchRead. The antitype of touchRead is as below:

```
uint16_t touchRead(uint8_t pin)
```

0 means no touch, 1 means touch. From T0 to T9 are pins, the corresponding pins of Firebeetle shown in the list as below.

Sequence No.	Corresponding Hardware	FireBeetle Board-ESP32
T0	GPIO4	DO/IO4
T1	GPIO0	IO0
T2	GPIO2	IO2/D9
T3	MTDO	A4/IO15
T4	MTCK	IO13/D7
T5	MTD1	MCLK/IO12
T6	MTMS	BCLK/EO14
T7	GPIO27	IO27/D4
T8	32K_XN	External crystal
T9	32K_XP	External crystal

Caution: Please do not use T8/T9 for touch input. They have already been connect to outer crystals of 32.768 KHz.

Project9 Deep_Sleep (Low power Management)

FireBeetle Board-ESP32 adopts advanced power management technique, offering different power-down modes for different environment. If you want to recall the FireBeetle Board-ESP32 from power-down mode, external interrupt and RTC timer both take effects.

The project explains the way to turn FireBeetle Board-ESP32 to power-down mode with Arduino code.

Components in need

1 x FireBeetle Board-ESP32



Caution: No need for other sensor.

Power-down modes of FireBeetle Board-ESP32:

- Active: The video chip is working, signal can be receipt, sent and monitored.
- Modem-sleep: Operational CPU, clock can be configured, Wi-Fi/Bluetooth radio is turned off.
- Light-sleep: Suspend operation of CPU, coprocessors RTC and ULP is running.
- Deep-sleep: Solely runs RTC, Wi-Fi and Bluetooth's connection data stored in RTC, coprocessors ULP works.
- Hibernate: Prohibit running of built-in 8MHz crystal and coprocessors ULP, the electricity of RTC memory recall is cut off.

Power mode	Comment	Power consumption
Active mode (RF working)	Wi-Fi Tx packet 13 dBm - 21 dBm	160 - 260 mA
	Wi-Fi / BT Tx packet 0 dBm	120 mA
	Wi-Fi / BT Rx and listening	80 - 90 mA
	Association sleep pattern (by light sleep)	0.9 mA@DTIM3 1.2 mA@DTIM1
Modem sleep	The CPU is powered on	Max speed: 20 mA Normal: 5 - 10 mA Slow speed: 3 mA
Light sleep		0.8 mA
Deep sleep	The ULP-coprocessor is powered on	0.5 mA
	Sensor-monitored deep sleep pattern	25 uA @1% duty
	RTC timer + recovery-memory	20 uA
	RTC timer only	5 uA
Shutdown (Turn-off)		2 uA

Caution: We could only support Deep_sleep mode in the example of power-down modes, for the improvement of Arduino' Support-Confidence.

Enter code (1)

Open Arduino IDE. You would better enter the code manually than open **Course->Item-9-1** to be familiar with it.

Sample code:

```
void setup() {
  // put your setup code here, to run once:
  pinMode(D9,OUTPUT);
  digitalWrite(D9,HIGH);
  delay(1000);
  digitalWrite(D9,LOW);
  ESP.deepSleep(5000000);
}

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

Please click [verify/compile](#) to review and please upload it after the confirmation. Once you click [Upload](#), IDE will send codes to FireBeetle Board-ESP32.

Code analysis

The example used `ESP.deepSleep` which has been encapsulated by the Arduino libraries. So there is no need to allocate extra sleep mode. The antitype function is as below:

```
void deepSleep(uint32_t time_us)
```

The parameter been passed is `uint32_t` type of data, value of time setting. The unit of it is μs (microsecond).

In the sample program, let FireBeetle Board-ESP32 sleep and restart in 5 seconds, you can see [L](#) LED light of the FireBeetle Board-ESP32 flash in every 5 seconds.

Enter code (2)

Open Arduino IDE. You would better enter the code manually than open [Course-> Item-9-2](#) to be familiar with it.

Sample code:

```
void setup() {
  // put your setup code here, to run once:
  pinMode(D9,OUTPUT);
  digitalWrite(D9,HIGH);
  delay(1000);
  digitalWrite(D9,LOW);
  esp_deep_sleep_enable_ext0_wakeup(GPIO_NUM_25,LOW);
  esp_deep_sleep_start();
}

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

Please click [verify/compile](#) to review and please upload it after the confirmation. Once you click [Upload](#), IDE will send codes to FireBeetle Board-ESP32.

Code analysis

Different from the way realized in Item-9-1, here the FireBeetle Board-ESP32 keep in sleep, you should take external interrupt to awake the motherboard and set GPIO25 into awaking interface of low lever interruption.

```
esp_deep_sleep_enable_ext0_wakeup(GPIO_NUM_25,LOW);
```

esp_deep_sleep_enable_ext0_wakeup is an bottom level setting function. Please do not worry about its principle and underlying settings related, you just need to know how to use it. Allocable interruption pin and corresponding pins in the FireBeetle Board-ESP32 shown as below.

The interrupt Parameter	Corresponidng Hardware	FireBeetle Board-ESP32
GPIO_NUM_0	GPIO0	IO0
GPIO_NUM_1	GPIO1	IO1/TXD
GPIO_NUM_2	GPIO2	IO2/D9
GPIO_NUM_3	GPIO3	IO3/RXD
GPIO_NUM_4	GPIO4	DO/IO4
GPIO_NUM_5	GPIO5	IO5/D8
GPIO_NUM_6	GPIO6	IO6/CLK
GPIO_NUM_7	GPIO7	IO7/SD0
GPIO_NUM_8	GPIO8	IO8/SD1
GPIO_NUM_9	GPIO9	IO9/D5
GPIO_NUM_10	GPIO10	IO10/D6
GPIO_NUM_11	GPIO11	IO11/CMD
GPIO_NUM_12	GPIO12	MCLK/IO12
GPIO_NUM_13	GPIO13	IO13/D7
GPIO_NUM_14	GPIO14	BCLK/IO14
GPIO_NUM_15	GPIO15	A4/IO15
GPIO_NUM_16	GPIO16	DI/IO16
GPIO_NUM_17	GPIO17	LRCK/IO17
GPIO_NUM_18	GPIO18	SCK/IO18
GPIO_NUM_19	GPIO19	MISO/IO19
GPIO_NUM_21	GPIO21	SDA/IO21
GPIO_NUM_22	GPIO22	SCL/IO22
GPIO_NUM_23	GPIO23	MOSI/IO23
GPIO_NUM_25	GPIO25	IO25/D2
GPIO_NUM_26	GPIO26	IO26/D3
GPIO_NUM_27	GPIO27	IO27/D4
GPIO_NUM_34	GPIO34	A2/IO34
GPIO_NUM_35	GPIO35	A3/IO35
GPIO_NUM_36	GPIO36	A0/IO36
GPIO_NUM_39	GPIO39	A1/IO39

Caution: Please only use GPIO_NUM_X to pass a parameter of pin when allocate the interrupt pin to awake. Due to the underlying function called cannot pass Dx directly in Arduino ino file.

