ESP32 Note

# Useful linkage

ESP官网:

[Welcome to ESP32 Arduino Core’s documentation — Arduino-ESP32 2.0.6 documentation (espressif.com)](https://docs.espressif.com/projects/arduino-esp32/en/latest/index.html)

Arduino官网:

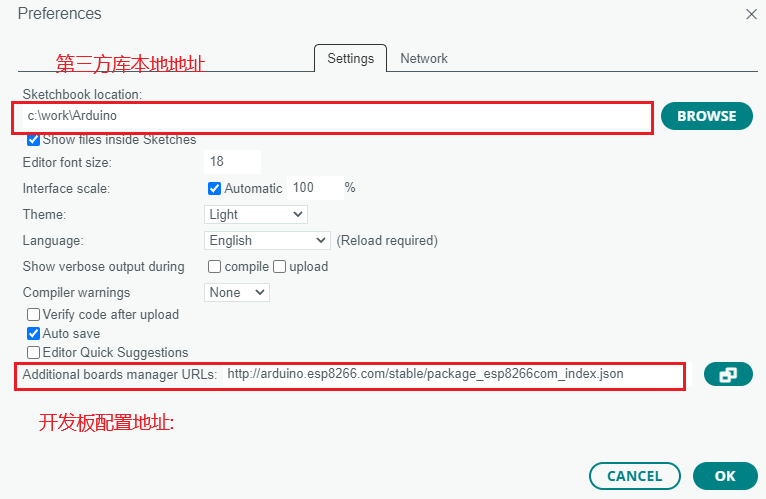
[Arduino Reference - Arduino Reference](https://www.arduino.cc/reference/en/)

Simulation: [esp32-arduino.ino - Wokwi ESP32, STM32, Arduino Simulator](https://wokwi.com/projects/305566932847821378)

Arduino内置库地址:

C:\Users\osun\AppData\Local\Arduino15

Arduino第三方库地址:

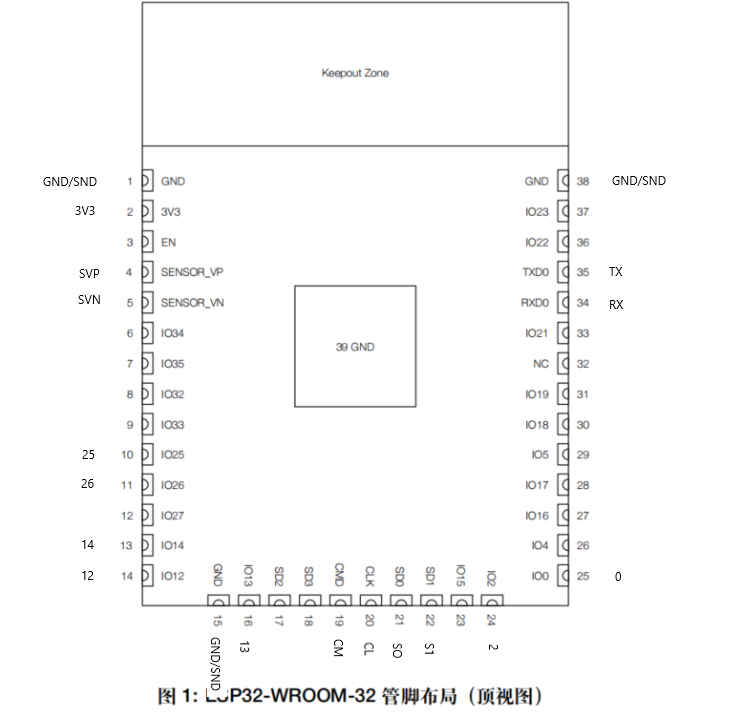


Useful linkage:

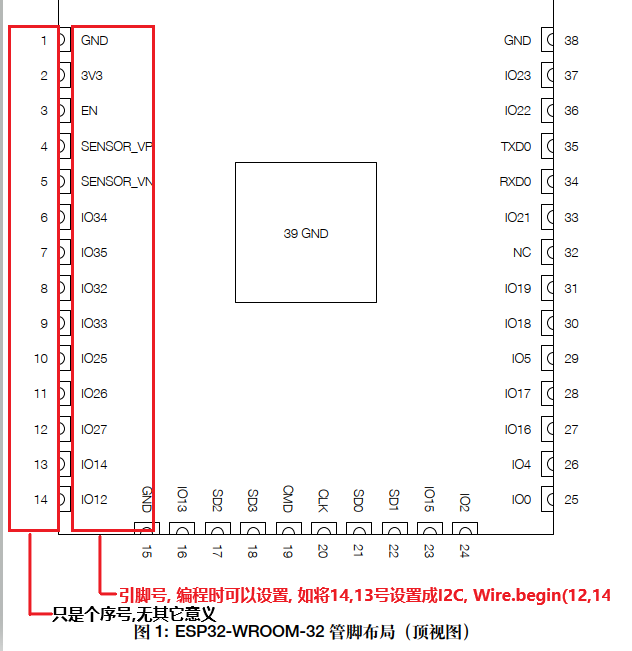
1. [通过Arduino使用ESP-WROOM-32（环境搭建和配置-LED闪烁篇）-电子发烧友网 (elecfans.com)](https://www.elecfans.com/d/2012628.html)

# ESP32硬件

## 引脚定义:



## 引脚的认识



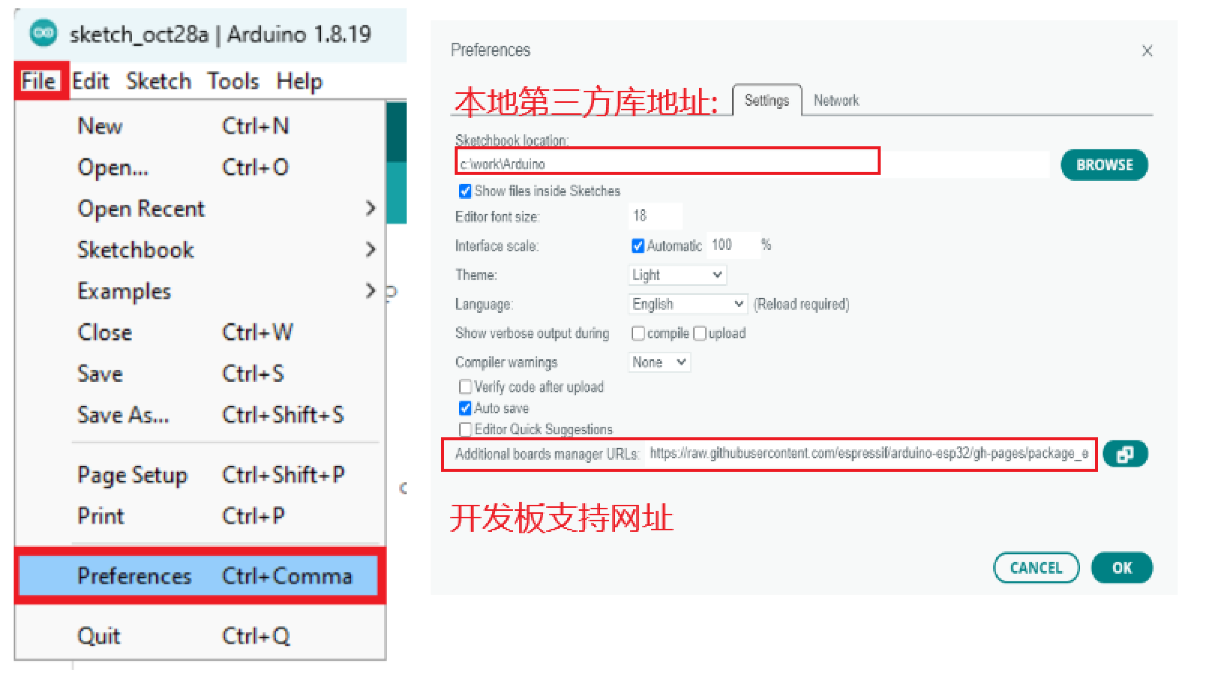
# 环境配置

## Arduino IDE下载安装:

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



## 在Arduino IDE中安装ESP32, 或其它开发板,如ESP8266:



**ESP32 URL:** <https://raw.githubusercontent.com/espressif/arduino-esp32/gh-pages/package_esp32_index.json>

**ESP8266 URL:** http://arduino.esp8266.com/stable/package\_esp8266com\_index.json

**Arduino内置库地址:** C:\Users\osun\AppData\Local\Arduino15

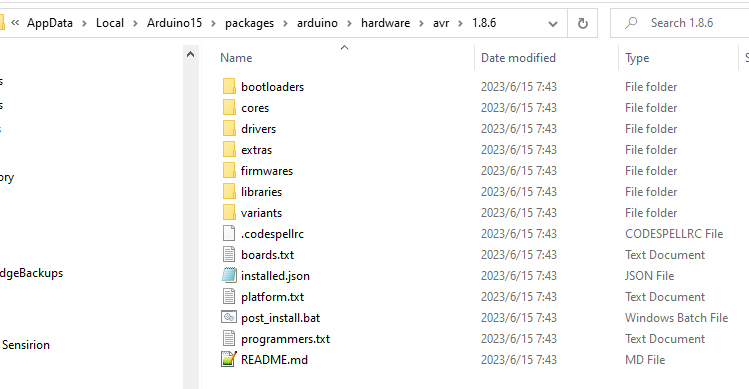
# Arduino配置文件

在单片机代码中，需要对不同的引脚、定时器、接口进行操作，由于不同单片机引脚和功能不一样，因此代码无法共用。那么，arduino是如何做到用一份代码就可以分别在不同类型的板子上执行呢？

其实arduino实现了一个框架，针对不同的芯片，把相同的部分提供一套通用机制，不同的部分抽象出来，制定一套规则，只要所有的芯片都遵守这个规则，就可以正常的工作起来。

这里主要讲讲arduino各种板子配置文件的规则。

**以ESP8266开发板的配置文件为例:**



**Bootloaders:** 不用芯片需要用的bootloader文件;

**cores:** 单片机程序核心代码所在的位置, 这里定义了所有芯片的通用功能,比如: 延时, 串口通讯, 按键, 输入输出等基本功能.

**Firmwares:** 一些固件程序, 有的没有.

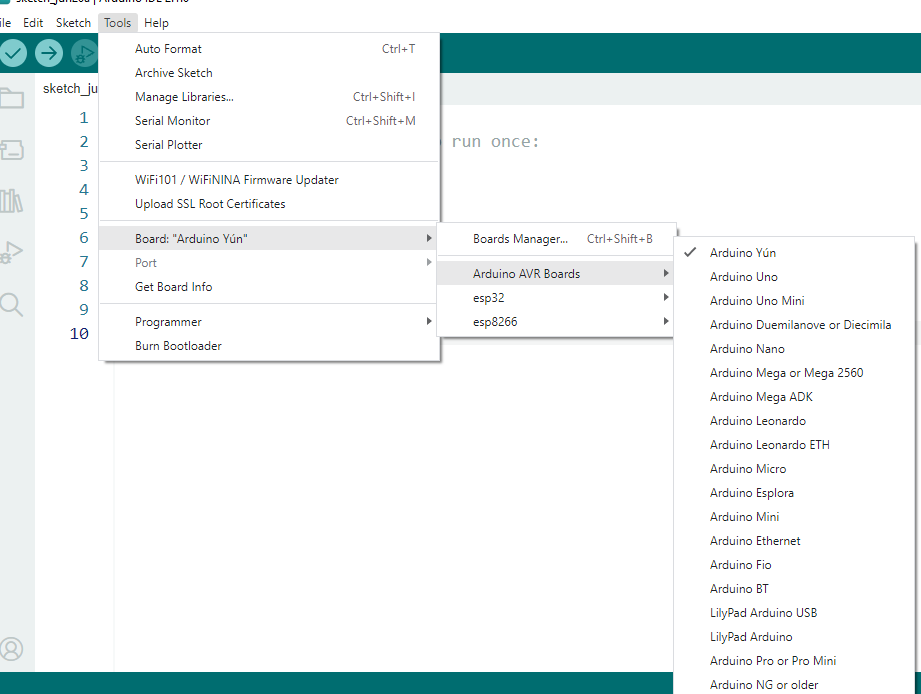
**Libraries:** 开源库文件, 比如EPROM,wifi,wire等库文件.

**Libraries\SPI\examples:**示例程序, (通过IDE的file/examples菜单可以浏览得到)..

**variants:** 针对不同芯片定义的引脚文件.

**boards.txt:** 各种型号arduino板子参数描述文件, 烧写固件时要用到.

通过如下Tools/board下面的选项就在这里配置.

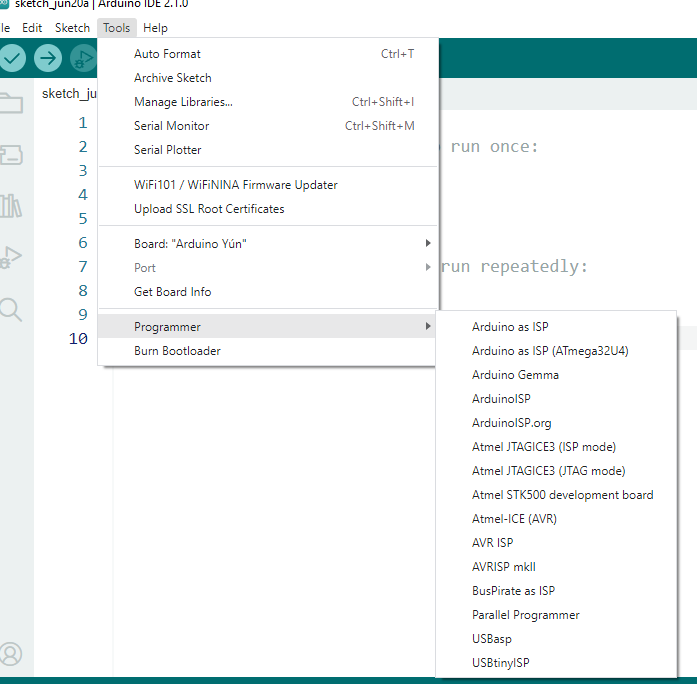


**platform.txt:**

这个文件里定义了编译步骤，以及每个文件编译使用的编译选项。可以在文件à参数选项 里把显示编译详细信息勾选上，与该txt文件中的内容对比起来看。



**programmers.txt:** 打开Arduino依次选择tools/programmer, 可以看到很多编程器选项.



其实这些编程器是通过programmer.txt定义的,在这个文件里主要定义了编程器需要遵守的规则:

# ESP开发模式

arduino里面的esp开发模式常用的有两种：

* 1. ESP32 DEV Module（对应ESP WROOM 32 芯片）

1. ESP32 WROVER Module（对应ESP WROVER 32 芯片）

更多帮助见ESPRESSIF官网: [Getting Started — Arduino-ESP32 2.0.6 documentation (espressif.com)](https://docs.espressif.com/projects/arduino-esp32/en/latest/getting_started.html)

# WiFi

## 将ESP32设置位STA

#include<WiFi.h>

const char\* ssid="nova 10";   //定义两个字符串指针常量

const char\* password="bydzv72m8smkuth";

void setup() {

    Serial.begin(115200);

    delay(10);

    // We start by connecting to a WiFi network

    Serial.println();

    Serial.println();

    Serial.print("Connecting to ");

    Serial.println(ssid);

    WiFi.begin(ssid, password);

    while (WiFi.status() != WL\_CONNECTED) {

        delay(500);

        Serial.print(".");

    }

    Serial.println("");

    Serial.println("WiFi connected.");

    Serial.println("IP address: ");

    Serial.println(WiFi.localIP());

}

void loop() {

  // put your main code here, to run repeatedly:

}

## 将ESP32设置为AP

#include<WiFi.h>

IPAddress local\_IP(192,168,4,22);

IPAddress gateway(192,168,4,22);

IPAddress subnet(255,255,255,0);

const char \*ssid = "ESP32\_AP\_OSUN";

const char \*password = "12345678";

void setup()

{

  Serial.begin(115200);

  Serial.println();

  WiFi.mode(WIFI\_AP); //设置工作在AP模式

  WiFi.softAPConfig(local\_IP, gateway, subnet); //设置AP地址

  while(!WiFi.softAP(ssid, password)){}; //启动AP

  Serial.println("AP启动成功");

  Serial.print("IP address: ");

  Serial.println(WiFi.softAPIP()); // 打印IP地址

  WiFi.softAPsetHostname("myHostName"); //设置主机名

  Serial.print("HostName: ");

  Serial.println(WiFi.softAPgetHostname()); //打印主机名

  Serial.print("mac Address: ");

  Serial.println(WiFi.softAPmacAddress()); //打印mac地址

  Serial.println(WiFi.getMode());

}

void loop() {

  //delay(1000);

  //Serial.println(WiFi.softAPgetStationNum()); //打印客户端连接数

}

# 通过WiFi和PC传输数据

# BlueTooth

# 通过BlueTooth和其它设备通讯

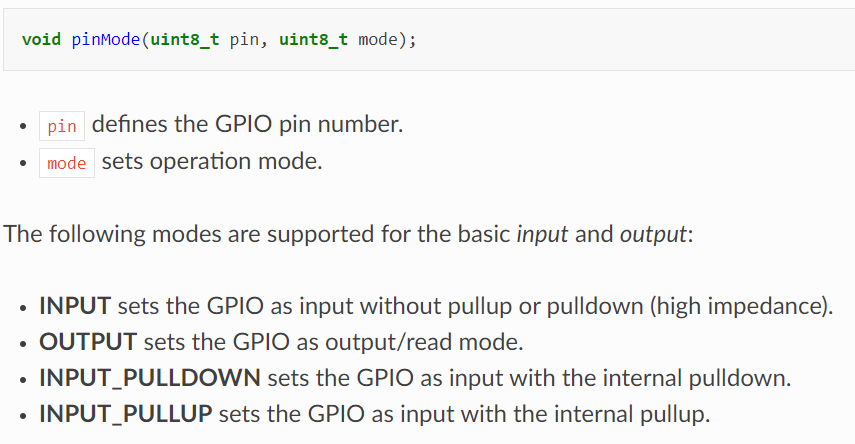
# Communication:

# Digital I/O

**gpio.h地址:** C:\Users\osun\AppData\Local\Arduino15\packages\esp32\hardware\esp32\2.0.9\tools\sdk\esp32\include\esp\_rom\include\esp32\rom

**导入方式:** #include "esp32/rom/gpio.h" (可能会变化)

## pinMode:



A screenshot of a computer

Description automatically generated with medium confidence

## digitalWrite:

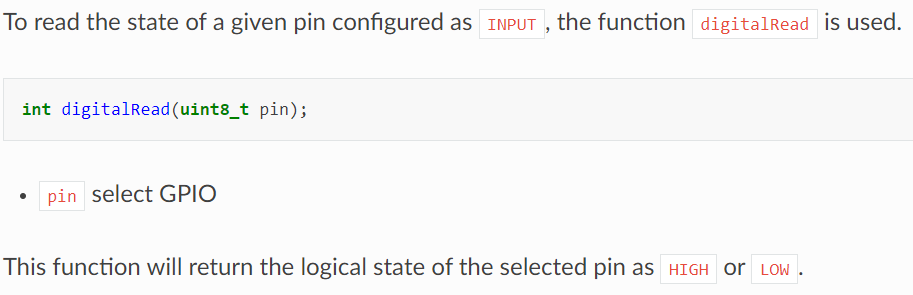
对于输出(OUTPUT)引脚, 设置是HIGH or LOW;

A screenshot of a computer

Description automatically generated with medium confidence

## digitalRead:

读取输入引脚的电平; 返回:HIGH or LOW;



[(89条消息) 一灯大师，点亮ESP32的LED\_esp32 led\_Wireless\_Link的博客-CSDN博客](https://wlink.blog.csdn.net/article/details/124017853)

# Analog I/O

## analogRead():

读出模拟引脚上的电压，并以二进制数值返回(0-1023)对应为0~5v/0~3.3v, 请参考官网;

int analogPin = A3; // A3是变量,已经在头文件pins\_arduino.h里定义; 或者给引脚号也可以, 如39;

int val = 0;  // variable to store the value read

void setup() {

  Serial.begin(9600);           //  setup serial

}

void loop() {

  val = analogRead(analogPin);  // read the input pin

  Serial.println(val);          // debug value

}

## analogWrite(): 给引脚输出PWM电压

analogWrite(pin,value):

pin: int

value: 具体的pwm值，(0~255对应表示一个周期的高电平占空比)

int ledPin = 9;      // LED connected to digital pin 9

int analogPin = 3;   // potentiometer connected to analog pin 3

int val = 0;         // variable to store the read value

void setup() {

  pinMode(ledPin, OUTPUT);  // sets the pin as output

}

void loop() {

  val = analogRead(analogPin);  // read the input pin

  analogWrite(ledPin, val / 4); // analogRead values go from 0 to 1023, analogWrite values from 0 to 255

}

# Advance I/O

# Serial:

Arduino, Serial: [Serial - Arduino Reference](https://www.arduino.cc/reference/en/language/functions/communication/serial/)

## if(Serial): 判断端口是否可用:

Serial返回: bool

void setup() {

  //Initialize serial and wait for port to open:

  Serial.begin(9600);

  while (!Serial) {

    ; // wait for serial port to connect. Needed for native USB

  }

}

void loop() {

  //proceed normally

}

## available(): 判断串口缓存里是否有数据;

**Note:** 如果串口缓存里是空的,去读的话返回的就是空; 注意使用时通常用delay(100)以保证串口字符接收完毕，即保证Serial.available()返回的是缓冲区准确的可读字节数。

**Return:** 可读的字节长度;

int incomingByte = 0; // for incoming serial data

void setup() {

  Serial.begin(9600); // opens serial port, sets data rate to 9600 bps

}

void loop() {

  // reply only when you receive data:

if (Serial.available() > 0) {

delay(100); // 等待数据传完

    // read the incoming byte:

    incomingByte = Serial.read();

    // say what you got:

    Serial.print("I received: ");

    Serial.println(incomingByte, DEC);

  }

}

## availableForWrite(): 缓存区可以写入字节的长度

Return: 字节长度

## begin():设置波特率

void setup() {

    Serial.begin(9600); // opens serial port, sets data rate to 9600 bps

}

void loop() {}

## end(): 终止通讯

**Note:** Disables serial communication, allowing the RX and TX pins to be used for general input and output. To re-enable serial communication, call Serial.begin().

## find(target, length): 串口查找指定字符串

**Parameter:**

target: the string to search for. Allowed data types: char.

length: length of the target. Allowed data types: size\_t. 可以不传;

**Note:**

从串口缓存区读取数据，寻找目标字符串target(char型)

**Return:** Bool

char target[] ="test";

void setup() {

  Serial.begin(9600);

  while(Serial.read()>= 0){}//clear serialbuffer

}

void loop() {

  // read data from serial port

   if(Serial.available()>0){

      delay(100);

      if( Serial.find(target)){

         Serial.print("find traget:");

         Serial.println(target);

        }

     }

    // clear serial buffer

    while(Serial.read() >= 0){}

}

## findUntil(): 等待串口缓存中出现目标数据

**Parameter:**

target: 要搜索的字符串: char

terminal: the terminal string in the search. Allowed data types: char.

**Return:** Bool

## flush(): 等待输出数据传送完毕

Note: 如果要清空串口缓存的话，可以使用：\*\*\*\*while(Serial.read() >= 0)\*\*\*\*来代替。). 因Serial.read()函数读取串口缓存中的一个字符，并删除已读字符，如果清空了返回值为-1。因此可以用这句代码来清空串口缓存。

## parseFloat(): 读串口缓存区第一个有效的浮点型数据，

Note: 数字将被跳过。当读到第一个非浮点数时函数结束。

用法:

*Serial*.parseFloat()  
*Serial*.parseFloat(lookahead)  
*Serial*.parseFloat(lookahead, ignore)

**Parameter:**

**lookahead:**

Allowed lookahead values:

* SKIP\_ALL: all characters other than a minus sign, decimal point, or digits are ignored when scanning the stream for a floating point number. This is the default mode.
* SKIP\_NONE: Nothing is skipped, and the stream is not touched unless the first waiting character is valid.
* SKIP\_WHITESPACE: Only tabs, spaces, line feeds, and carriage returns are skipped.

**ignore:** used to skip the indicated char in the search. Used for example to skip thousands divider. Allowed data types: char

## parseInt(): **从串口接收数据流中读取第一个有效整数（包括负数）**

**Note:**

非数字的首字符或者负号将被跳过

当可配置的超时值没有读到有效字符时，或者读不到有效整数时，分析停止

如果超时且读不到有效整数时，返回0;

**用法:**

*Serial*.parseInt()  
*Serial*.parseInt(lookahead)  
*Serial*.parseInt(lookahead, ignore)

**Parameter:**

同parseFloat()

int comInt;

void loop() {

  // read data from serial port

  if(Serial.available()>0){

      delay(100);

      comInt = Serial.parseInt();

      Serial.print("Serial.parseInt:");

      Serial.println(comInt);

    }

    // clear serial buffer

    while(Serial.read() >= 0){}

## peek(): 读串口缓存中下一字节的数据（字符型），但不删除该数据。

**Note:** 也就是说，连续的调用peek()将返回同一个字符。而调用read()则会返回下一个字符。

**Return:** The first byte of incoming serial data available (or -1 if no data is available). Data type: int.

## print(): 打印

**Note:** Serial.print("\n") 和 Serial.println() 等效，相当于多了一个换行符;

**用法:**

1. Serial.print(78, BIN) 得到 "1001110"
2. Serial.print(78, OCT) 得到 "116"
3. Serial.print(78, DEC) 得到 "78"
4. Serial.print(78, HEX) 得到 "4E"
5. Serial.print(1.23456, 0) 得到 "1"
6. Serial.print(1.23456, 2) 得到 "1.23"
7. Serial.print(1.23456, 4) 得到 "1.2346"
8. Serial.print('N') 得到 "N"
9. Serial.print("Hello world.") 得到 "Hello world."

## read(): Reads incoming serial data.

**Note:** 读取串口数据，一次读一个字符，读完后删除已读数据。

int incomingByte = 0; // for incoming serial data

void setup() {

  Serial.begin(9600); // opens serial port, sets data rate to 9600 bps

}

void loop() {

  // send data only when you receive data:

  if (Serial.available() > 0) {

    // read the incoming byte:

    incomingByte = Serial.read();

    // say what you got:

    Serial.print("I received: ");

    Serial.println(incomingByte, DEC);

  }

}

## readBytes():从串口读取指定长度length的字符到缓存数组buffer

Serial.readBytes(buffer,length);

**参数：** buffer: 缓存变量

length:设定的读取长度

**返回：**返回存入缓存的字符数，0表示没有有效数据。

char buffer[18];

int numdata=0;

void setup() {

  Serial.begin(9600);

  while(Serial.read()>= 0){}//clear serial port

}

void loop() {

  // read data from serial port

  if(Serial.available()>0){

      delay(100);

      numdata = Serial.readBytes(buffer,3);

      Serial.print("Serial.readBytes:");

      Serial.println(buffer);

    }

    // clear serial buffer

  while(Serial.read() >= 0){}

  for(int i=0; i<18; i++){

        buffer[i]='\0';

    }

}

## readBytesUntil():从串口缓存读取指定长度的字符到数组buffer，遇到终止字符character后停止。

Serial.readBytesUntil(character ,buffer,length);

参数： character : 查找的字符 (char)

buffer: 存储读取数据的缓存(char[] 或byte[])

length:设定的读取长度

返回：返回存入缓存的字符数，0表示没有有效数据。

char buffer[18];

char character = ','; //终止字符

int numdata=0;

void setup() {

  Serial.begin(9600);

  while(Serial.read()>= 0){}//clear serialport

}

void loop() {

  // read data from serial port

  if(Serial.available()>0){

      delay(100);

      numdata =Serial.readBytesUntil(character,buffer,3);

      Serial.print("Serial.readBytes:");

      Serial.println(buffer);

    }

    // clear serial buffer

  while(Serial.read() >= 0){}

  for(int i=0; i<18; i++){

        buffer[i]='\0';

    }

}

## readString():从串口缓存区读取全部数据到一个字符串型变量

String comdata = "";

void setup() {

  Serial.begin(9600);

  while(Serial.read()>= 0){} //clear serialbuffer

}

void loop() {

  // read data from serial port

  if(Serial.available()>0){

      delay(100);

      comdata = Serial.readString();

      Serial.print("Serial.readString:");

      Serial.println(comdata);

    }

    comdata = "";

}

## setTimeout(): 设置串口等待时间, 默认1000 millseconds

## write(): 向串口写数据

Serial.write(val)

Serial.write(str)

Serial.write(buf, len)

参数 ：val: 字节

str: 一串字节

buf: 字节数组

len: buf的长度

void setup(){

Serial.begin(9600);

}

void loop(){

  Serial.write(45); // send a byte with the value 45

  int bytesSent = Serial.write(“hello”); //send the string “hello” and return the length of the string.

}

## serialEvent(): 串口准备好时触发的事件函数

Serial.serialEvent{"statements"}

参数

statements: 执行任何有效的语句。

Note: Called at the end of loop() when data is available. Use Serial.read() to capture this data.

串口数据准备好时触发的事件函数，即串口数据准备好调用该函数。

[(89条消息) Arduino串口函数详解\_LoveArduinos的博客-CSDN博客](https://blog.csdn.net/u014421313/article/details/125421394?ops_request_misc=%257B%2522request%255Fid%2522%253A%2522168834495116800182199134%2522%252C%2522scm%2522%253A%252220140713.130102334..%2522%257D&request_id=168834495116800182199134&biz_id=0&utm_medium=distribute.pc_search_result.none-task-blog-2~all~sobaiduend~default-1-125421394-null-null.142%5ev88%5econtrol_2,239%5ev2%5einsert_chatgpt&utm_term=Serial.find&spm=1018.2226.3001.4187)

# Time

## delay(): 暂停, 毫秒

int ledPin = 13;              // LED connected to digital pin 13

void setup() {

  pinMode(ledPin, OUTPUT);    // sets the digital pin as output

}

void loop() {

  digitalWrite(ledPin, HIGH); // sets the LED on

  delay(1000);                // waits for a second

  digitalWrite(ledPin, LOW);  // sets the LED off

  delay(1000);                // waits for a second

}

## delayMicroseconds(): 暂停, 微秒

int outPin = 8;               // digital pin 8

void setup() {

  pinMode(outPin, OUTPUT);    // sets the digital pin as output

}

void loop() {

  digitalWrite(outPin, HIGH); // sets the pin on

  delayMicroseconds(50);      // pauses for 50 microseconds

  digitalWrite(outPin, LOW);  // sets the pin off

  delayMicroseconds(50);      // pauses for 50 microseconds

}

## micros()/millis(): 从开始执行到现在目前的时间(单位：ms)

unsigned long time;

void setup() {

  Serial.begin(9600);

}

void loop() {

  Serial.print("Time: ");

time\_mc = micros(); //最长70分钟, 然会归零;

time\_ml = millis(); //最长50天, 然后归零;

  Serial.println(time); //prints time since program started

  delay(1000);          // wait a second so as not to send massive amounts of data

}

# Math

## abs(): 绝对值

## constrain(): 把一个数(任何类型)约束在某一范围

constrain(x, a, b)

**Parameters:**

x: the number to constrain Allowed data types: all data types.

a: the lower end of the range. Allowed data types: all data types.

b: the upper end of the range. Allowed data types: all data types.

**Return:**

x: if x is between a and b.

a: if x is less than a.

b: if x is greater than b.

## map(): 将一个数字从一个范围映射到另一个范围

map(value, fromLow, fromHigh, toLow, toHigh)

**Parameters:**

value：要映射的数

fromLow：当前值范围的下限

fromHigh：当前值范围的下限

toLow：目标值范围的下限

toHigh：目标值范围的上限

**返回值：映射后的值**

**eg: (读取IO0的模拟电压(01023)，并将其设置到IO9(0255)(其实也就是PWM))**

int val = analogRead(0);

val = map(val, 0, 1023, 0, 255);

analogWrite(9, val);

## max(x,y): 返回最大值

## min(x,y): 返回最小值

## pow(base, exponet): 计算一个数的幂

## sq(): 平方

## sqrt: 平方根

# Characters

## isAplha(thisChar): 判断字符串中是否有字母

## isAlphaNumeric(thisChar): 判断输入的字符是否全是字母或者数字

Note: 为真时返回true。

## isAscii(thisChar): 用于检查参数是否为ASCII 码字符

Note: 也就是判断c 的范围是否在0 到127 之间。为真时返回true。

## isControl(thisChar): 判断字符是否属于控制字符类别

## isDigit(thisChar): 用来检测一个字符是否是十进制数字

## isGrap(thisChar): 检查所传的字符是否有图形表示法

Note: 带有图形表示法的字符是除了空白字符（比如 ’ '）以外的所有可打印的字符。

## isHexadecimalDigit(thisChar): 检查所传字符是否是十六进制位数

## isLowerCase(thisChar): 检查所传字符是否是小写字母

## isPrintable(thisChar): 是否为可打印字符

## isSpace(thisChar): 是不是空格

## isUpperCase(thisChar): 是否为大写字符

## isWhiteSpace(thisChar): 是否是空白字符，包含TAB，回车等。

# I2C

## Read SCD30

// Basic demo for readings from Adafruit SCD30

#include <Adafruit\_SCD30.h>

#include <Wire.h>

Adafruit\_SCD30  scd30;

void setup(void) {

  Serial.begin(115200);

  while (!Serial) delay(10);     // will pause Zero, Leonardo, etc until serial console opens

  Wire.begin(12, 14); //Redesign the I2C pin.

  Serial.println("Adafruit SCD30 test!");

  // Try to initialize!

  if (!scd30.begin()) {

    Serial.println("Failed to find SCD30 chip");

    while (1) { delay(10); }

  }

  Serial.println("SCD30 Found!");

  // if (!scd30.setMeasurementInterval(10)){

  //   Serial.println("Failed to set measurement interval");

  //   while(1){ delay(10);}

  // }

  Serial.print("Measurement Interval: ");

  Serial.print(scd30.getMeasurementInterval());

  Serial.println(" seconds");

}

void loop() {

  if (scd30.dataReady()){

    Serial.println("Data available!");

    if (!scd30.read()){ Serial.println("Error reading sensor data"); return; }

    Serial.print("Temperature: ");

    Serial.print(scd30.temperature);

    Serial.println(" degrees C");

    Serial.print("Relative Humidity: ");

    Serial.print(scd30.relative\_humidity);

    Serial.println(" %");

    Serial.print("CO2: ");

    Serial.print(scd30.CO2, 3);

    Serial.println(" ppm");

    Serial.println("");

  } else {

    //Serial.println("No data");

  }

  delay(100);

}

# UART

# SPI

ESP PSI设置:

* + 1. [Esp32 spi slave配置](https://blog.csdn.net/smile_5me/article/details/115377584?ops_request_misc=&request_id=&biz_id=102&utm_term=ESP32%20SPI%E9%85%8D%E7%BD%AE&utm_medium=distribute.pc_search_result.none-task-blog-2~all~sobaiduweb~default-6-115377584.142%5ev88%5econtrol_2,239%5ev2%5einsert_chatgpt&spm=1018.2226.3001.4187);

# PWM

# gpio.h:

# Wire

## Wire.begin()

Wire.begin() 和 Wire.begin(address)

初始化wire库, 并且加入到I2C网络,前者作为Master,后者作为Slaver,并且只能调用一次.

参数:

address: 7位的器件地址(可选), 如果没有输入,则以Master的形式加入到I2C网络.

例:

**官方的建议:**

int sda\_pin = 16; // GPIO16 as I2C SDA

int scl\_pin = 17; // GPIO17 as I2C SCL

void setup()

{

    Wire.setPins(sda\_pin, scl\_pin); // Set the I2C pins before begin

    Wire.begin(); // join i2c bus (address optional for master)

}

**下面的写法也可以:**

void setup()

{

    Wire.begin(16, 17); // join i2c bus (address optional for master)

}

## Wire.requestFrom()

主设备请求从设备一个字节, 这个字节可以被主设备用read()或availabel()接受.

Wire.requestFrom(address, uantity)

Wire.requestFrom(address, quantity, stop)

参数:

address: 7位的器件地址;

quantity: 请求得到的数量;

stop: bool, ‘1’ 则在请求结束后发送一个停止命令, 并释放总线, ‘0’则继续发送请求保持连接

返回: 字节形式, 从从机接受到的字节数目.

## Wire.beginTransmission()

Wire.beginTransmission(address)

开始一次传输数据, 发送一个I2C开始字符.

参数:

address: 器件7位地址

返回: none

## Wire.endTransmission()

Wire.endTransmission()或Wire.endTransmission(stop)

结束一个由beginTransmission（）开始的并且由write（）排列的从机的传输。在Arduino中 endTransmission（）接受到

一个布尔形变量，如果为1 则endTransmission（）发送一个停止信息；如果为0 则发送开始信息

返回:

0 成功 1 数据溢出 2 发送addtess时从机接受到NACK 3 发送数据时接受到 NACK 4 其他错误

## Wire.write()

向从机发送数据

形式:

Wire.write(value)

Wire.write(string)

Wire.write(data, length)

参数:

value: 要发送的数值;

string: 字符组的指针

length: 传输的数量

## Wire.available()

Wire.requestFrom()请求从机数据后,可以用available()接收

## Wire.read()

Wire.requestFrom() 请求从及数据后,可以使用read接收.

## Wire.onReceive()

从机接收主机发来的数据.

## Wire.onRequest()

从机请求主机发送数据.