

# 基础入门

## 1. 硬件与开发环境

### 1.1 Arduino IDE

#### 目标平台




- 专为 **Arduino** 开发板以及兼容Arduino的开发板设计，支持通过简单的 C/C++ 代码控制硬件。
- 主要用于编写、编译和上传程序到 Arduino 微控制器。
- 适合嵌入式开发和物联网 (IoT) 项目。

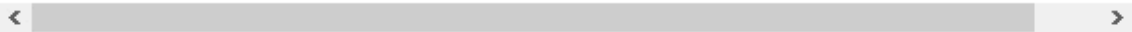
#### 支持的编程语言

- 主要支持 **C/C++** 语言，但由于 Arduino 本身简化了很多复杂的部分，实际上可以认为 Arduino 编程是一种简化的 C/C++，包含了很多特定的库和框架来操作硬件。
- 提供了大量 Arduino 库，帮助开发者与传感器、马达、屏幕等硬件交互。

#### Arduino IDE的安装

- 点击安装文件中的 `start.bat` 即可

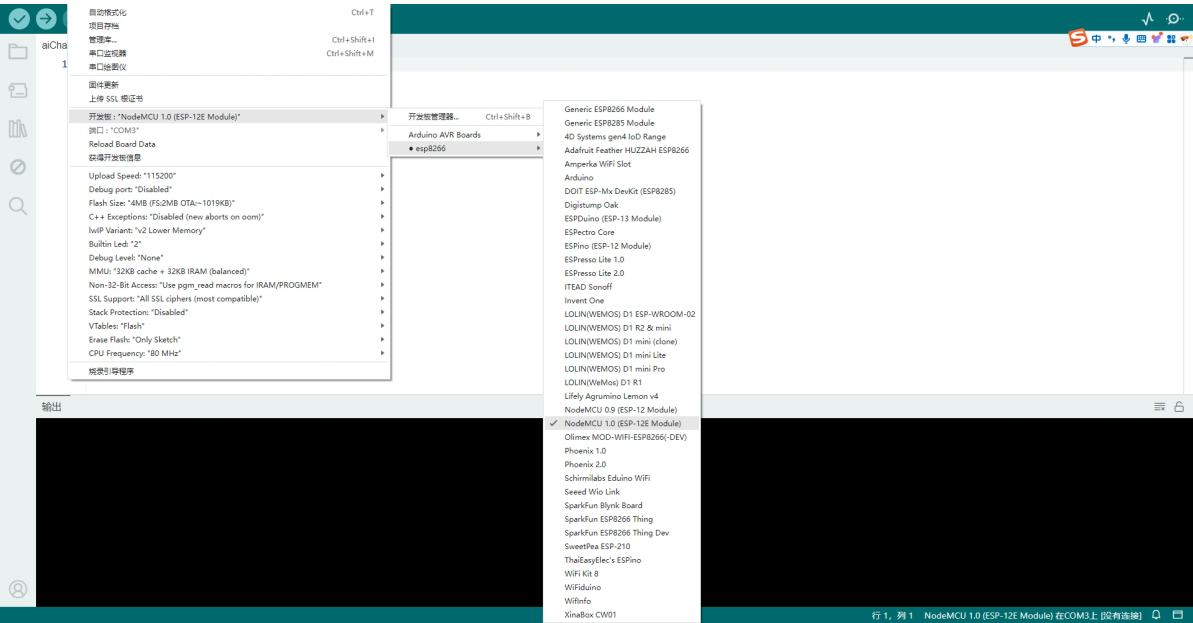
名称	修改日期	类型	大小
 .arduinoIDE.zip	2024/12/25 10:18	压缩(zipped)文件...	34
 Arduino IDE.zip	2024/12/25 10:22	压缩(zipped)文件...	1,691,10
 re-start.bat	2024/12/25 10:35	Windows 批处理...	1
 start.bat	2024/12/25 10:26	Windows 批处理...	1



#### 总结

- **Arduino IDE** 专注于硬件开发，支持 Arduino 板上的编程，非常适合嵌入式和物联网开发。

## 1.2 配置 ESP8266 开发板



## 1.3 OLED 屏幕接线与基本介绍（SCL、SDA）

### OLED 屏幕简介

OLED（**有机发光二极管**）屏幕是一种显示技术，因其低功耗、高对比度和灵活性被广泛应用于各种设备。OLED 屏幕不同于传统的液晶屏（LCD），它没有背光，而是每个像素点都能自发光，因此能够显示更加深邃的黑色，并且能在较低的功耗下提供更高的显示效果。

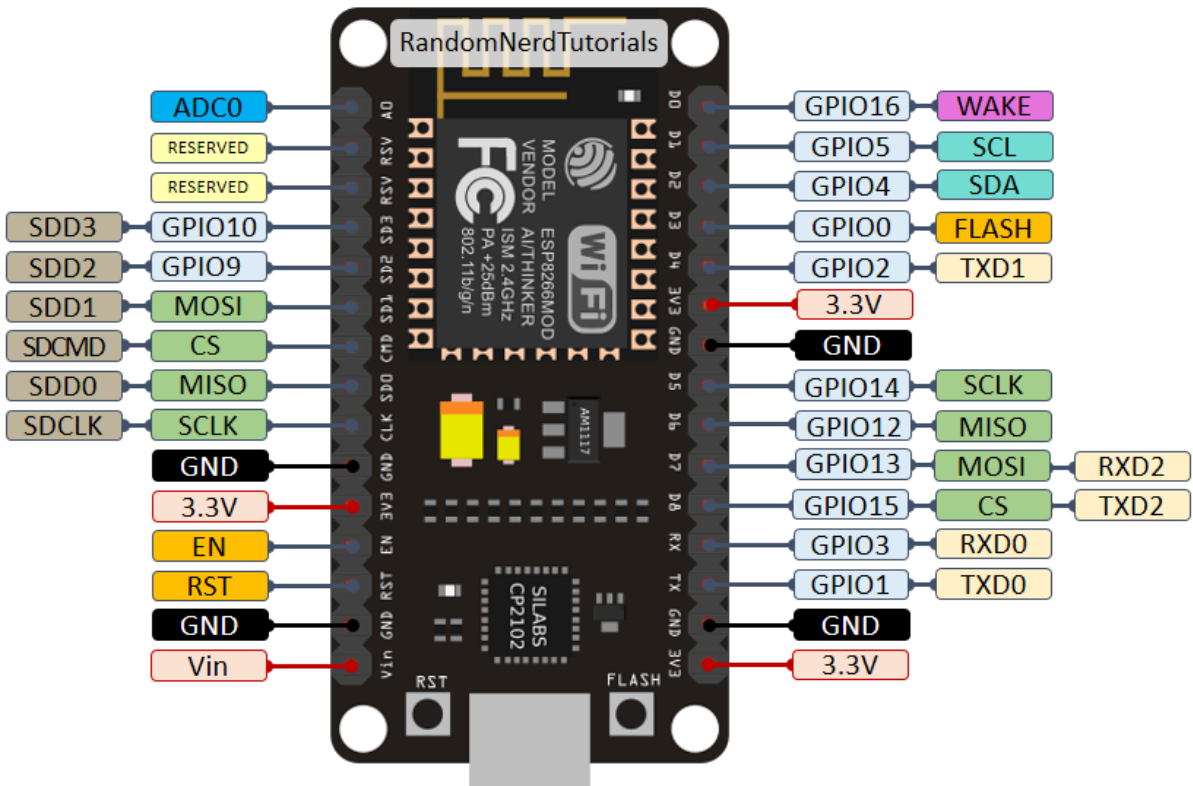
常见的 OLED 屏幕通常采用 I2C 或 SPI 接口与微控制器（如 Arduino、ESP32、ESP8266 等）进行通信。在本示例中，我们将讨论常见的 I2C 接口的 OLED 屏幕。

### OLED 屏幕的常见引脚（I2C 接口）

I2C（Inter-Integrated Circuit）是一种串行通信协议，使用两根信号线进行数据传输。I2C 总线上的每个设备都有一个唯一的地址。对于 OLED 屏幕，常见的引脚包括：

- VCC**（电源引脚）：
  - 连接至 3.3V 或 5V（根据 OLED 屏幕的规格要求，通常为 3.3V 或 5V）。
- GND**（接地引脚）：
  - 连接至微控制器的地（GND）。
- SCL**（时钟信号线）：
  - SCL（Serial Clock Line）是 I2C 总线中的时钟信号线，用于同步数据传输。它由主设备（如 Arduino）产生，控制 I2C 总线上的数据传输速度。
  - 通常连接到微控制器的 SCL 引脚（在 Arduino 上是 A5 引脚，在 ESP32/ESP8266 上是专门的 SCL 引脚）。
- SDA**（数据线）：
  - SDA（Serial Data Line）是 I2C 总线中的数据传输线。它用于在主设备和从设备（如 OLED 屏幕）之间传输数据。
  - 通常连接到微控制器的 SDA 引脚（在 Arduino 上是 A4 引脚，在 ESP32/ESP8266 上是专门的 SDA 引脚）。

ESP8266(ESP12F)引脚示意图



OLED 屏幕接线示意图

以下是一个典型的 OLED 屏幕与 Arduino 连接的接线图，假设你的 OLED 屏幕使用 I2C 接口：

OLED 引脚	Arduino 引脚
VCC	3.3V 或 5V
GND	GND
SCL	A5 (Arduino Uno) 或 SCL (ESP32/ESP8266)
SDA	A4 (Arduino Uno) 或 SDA (ESP32/ESP8266)

OLED 屏幕基本工作原理

1. I2C 通信协议：
- I2C 协议通过两根线（SCL 和 SDA）进行数据传输，支持多个设备共享同一条总线。
  - 在 I2C 协议中，主设备（如 Arduino）负责生成时钟信号（SCL），并且发送数据到 OLED 屏幕（SDA）。
  - OLED 屏幕通过其地址响应主设备的命令和数据。
2. 显示原理：
- OLED 屏幕由有机材料构成，每个像素点由发光二极管组成，能够通过控制电流亮灭。
  - OLED 屏幕可以显示非常清晰的文字和图像，并且具有较高的对比度。

## 其他注意事项

### 1. 电压兼容性:

- 许多 OLED 屏幕支持 3.3V 和 5V 电压，但请查看屏幕的规格说明书以确定所需电压。
- 如果你的开发板是 3.3V，而 OLED 屏幕是 5V 兼容型，则可以直接连接；但如果不兼容，请确保电压匹配以避免损坏设备。

### 2. I2C 地址:

- OLED 屏幕通常有多个 I2C 地址。例如，常见的 SSD1306 屏幕的 I2C 地址是 `0x3C` 或 `0x3D`，根据你的硬件，可能需要调整代码中的地址。

### 3. 显示内容的清晰度:

- OLED 屏幕通常有很高的分辨率，能够显示较小的字体和细节。在 Arduino 中，可以通过设置 `setTextSize()` 调整字体大小，使其适应显示内容。

## 总结

- **SCL** 和 **SDA** 是 I2C 总线的时钟信号和数据线，它们负责在主设备（如 Arduino）和 OLED 屏幕之间进行通信。
- **OLED 屏幕** 因其低功耗、高对比度和清晰度，常用于嵌入式系统中，尤其在小型显示屏上非常受欢迎。
- 使用 I2C 接口连接 OLED 屏幕可以减少引脚占用，并简化连接线。

## 1.4 使用 ESP8266WiFi.h 库连接 WiFi

### 使用 ESP8266WiFi.h 库连接 WiFi

`ESP8266WiFi.h` 是 ESP8266 WiFi 模块的官方库，用于在 ESP8266 开发板上实现与 Wi-Fi 网络的连接。这个库提供了多种方法来管理 Wi-Fi 连接，包括连接到指定的 Wi-Fi 网络、检查连接状态、断开连接等功能。

#### 步骤概述

##### 1. 导入必要的库:

- 使用 `ESP8266WiFi.h` 库来管理 Wi-Fi 连接。

##### 2. 设置 Wi-Fi 网络的 SSID 和密码:

- 在代码中指定连接的 Wi-Fi 网络名称 (SSID) 和密码。

##### 3. 连接到 Wi-Fi 网络:

- 使用 `WiFi.begin()` 方法连接到指定的 Wi-Fi 网络。

##### 4. 检查连接状态:

- 使用 `WiFi.status()` 来确认 ESP8266 是否已成功连接到 Wi-Fi 网络。

##### 5. 获取连接的 IP 地址:

- 连接成功后，可以使用 `WiFi.localIP()` 获取 ESP8266 在局域网中的 IP 地址。

## 示例代码：使用 `ESP8266WiFi.h` 库连接 Wi-Fi

```
#include <ESP8266WiFi.h> // 引入 ESP8266 WiFi 库

// Wi-Fi 配置
const char* ssid = "Your_SSID"; // Wi-Fi 名称
const char* password = "Your_PASSWORD"; // Wi-Fi 密码

void setup() {
    // 启动串口通信
    Serial.begin(115200);
    delay(1000);

    // 输出启动信息
    Serial.println("正在连接WiFi...");

    // 连接 Wi-Fi
    WiFi.begin(ssid, password);

    // 等待连接，直到 Wi-Fi 连接成功
    int attemptCount = 0;
    while (WiFi.status() != WL_CONNECTED) {
        delay(500);
        Serial.print(".");

        // 最大重试次数（例如，最多重试10次）
        attemptCount++;
        if (attemptCount >= 10) {
            Serial.println("\nWi-Fi连接失败，退出程序");
            return; // 如果超过最大重试次数，则退出
        }
    }

    // 输出连接成功的 IP 地址
    Serial.println("\nWi-Fi已连接");
    Serial.print("IP 地址: ");
    Serial.println(WiFi.localIP()); // 打印ESP8266的IP地址
}

void loop() {
    // 在 loop 中可以添加需要执行的其他任务
}
```

## 详细说明：

### 1. 引入库：

- `#include <ESP8266WiFi.h>` 用于引入 ESP8266 WiFi 库，允许你使用该库的函数来连接 Wi-Fi 网络。

### 2. 设置 Wi-Fi 配置：

- `const char* ssid = "Your_SSID";` 用于指定 Wi-Fi 网络名称。
- `const char* password = "Your_PASSWORD";` 用于指定连接的 Wi-Fi 密码。

### 3. 连接 Wi-Fi：

- `WiFi.begin(ssid, password);` 调用 `begin` 函数来尝试连接到指定的 Wi-Fi 网络。
- 你可以使用 `WiFi.status()` 来检查连接状态。这个函数返回一个连接状态，`WL_CONNECTED` 表示已经连接。

#### 4. 重试连接：

- 在 `while` 循环中，ESP8266 会不断尝试连接到 Wi-Fi 网络，直到成功为止。如果连接失败超过预定的重试次数（此示例中为 10 次），会退出连接过程。

#### 5. 获取 IP 地址：

- 一旦成功连接到 Wi-Fi 网络，你可以使用 `WiFi.localIP()` 获取 ESP8266 的 IP 地址。

## 其他常用的 `ESP8266WiFi.h` 函数

- `WiFi.status()`：返回当前连接状态，常见状态包括：
  - `WL_CONNECTED`：已连接 Wi-Fi 网络。
  - `WL_DISCONNECTED`：未连接。
  - `WL_IDLE_STATUS`：空闲状态。
  - `WL_CONNECT_FAILED`：连接失败。
- `WiFi.begin()`：启动连接 Wi-Fi 网络，需要提供 SSID 和密码。
- `WiFi.localIP()`：返回 ESP8266 当前连接的本地 IP 地址。
- `WiFi.disconnect()`：断开 Wi-Fi 连接。

## 其他注意事项

- **重启 Wi-Fi 连接**：如果需要重启 Wi-Fi 连接，可以使用 `WiFi.disconnect()` 函数，然后再调用 `WiFi.begin()` 来重新连接。
- **Wi-Fi 延迟**：在某些情况下，Wi-Fi 连接可能需要一些时间。在连接期间，建议增加一定的延迟，确保 Wi-Fi 连接稳定。
- **Wi-Fi 状态检测**：你可以定期使用 `WiFi.status()` 检查 Wi-Fi 连接是否断开，若断开，则可以尝试重新连接。

## 1.5 使用 `Adafruit_SSD1306` 库初始化 OLED

### 使用 `Adafruit_SSD1306` 库初始化 OLED 并显示简单文字

`Adafruit_SSD1306` 是一款常用的库，用于控制基于 SSD1306 驱动芯片的 OLED 屏幕。这个库支持多种尺寸的 OLED 屏幕，常见的有 128x64 和 128x32 两种分辨率的显示屏。下面将展示如何在 Arduino 上使用 `Adafruit_SSD1306` 库初始化 OLED 屏幕并显示简单的文字。

## 步骤

### 1. 安装库

- 打开 Arduino IDE，点击 **工具** -> **库管理器**。
- 在库管理器中，搜索 `Adafruit SSD1306` 并安装它。
- 同时需要安装 `Adafruit GFX` 库，它是 `Adafruit_SSD1306` 库的依赖库。

### 2. 接线

- 连接 OLED 屏幕与开发板。假设使用 I2C 接口连接，接线方式如下：
  - **VCC**：连接到开发板的 3.3V 或 5V（根据 OLED 屏幕要求）。

- **GND**: 连接到 GND。
- **SCL**: 连接到开发板的 SCL (对于 Arduino Uno 是 A5) 。
- **SDA**: 连接到开发板的 SDA (对于 Arduino Uno 是 A4) 。

### 3. 编写代码

- 使用 `Adafruit_SSD1306` 库初始化 OLED 屏幕并显示简单文字。

## 示例代码

```
#include <Wire.h>           // I2C 库
#include <Adafruit_GFX.h>    // 图形库
#include <Adafruit_SSD1306.h> // OLED 屏幕库

#define SCREEN_WIDTH 128    // 屏幕宽度
#define SCREEN_HEIGHT 32    // 屏幕高度

#define OLED_RESET -1       // OLED 屏幕的重置引脚 (-1 表示不使用硬件重置)
Adafruit_SSD1306 display(SCREEN_WIDTH, SCREEN_HEIGHT, &Wire, OLED_RESET); // 创建 OLED 对象

void setup() {
  // 启动串口通信
  Serial.begin(115200);

  // 初始化 OLED 屏幕
  if (!display.begin(SSD1306_SWITCHCAPVCC, 0x3C)) {
    Serial.println(F("SSD1306 初始化失败"));
    while (true); // 如果初始化失败, 停在这里
  }

  // 清空显示
  display.clearDisplay();

  // 设置文本颜色为白色
  display.setTextColor(SSD1306_WHITE);

  // 设置文本大小
  display.setTextSize(1);

  // 设置光标位置
  display.setCursor(0, 0);

  // 显示简单的文字
  display.println(F("Hello, OLED!"));

  // 刷新显示
  display.display();
}

void loop() {
  // 在循环中可以添加更多操作
}
```

## 代码说明

### 1. 导入必要的库：

- `wire.h`：I2C 通信库，用于与 OLED 屏幕进行通信。
- `Adafruit_GFX.h`：Adafruit 图形库，提供绘图和显示文本等功能。
- `Adafruit_SSD1306.h`：用于控制 SSD1306 驱动的 OLED 屏幕的库。

### 2. 定义 OLED 屏幕的尺寸和 I2C 地址：

- `SCREEN_WIDTH` 和 `SCREEN_HEIGHT` 定义了 OLED 屏幕的分辨率（128x32）。
- `OLED_RESET` 是重置引脚（通常使用 `-1` 表示不使用硬件重置引脚）。

### 3. 初始化 OLED 屏幕：

- `display.begin(SSD1306_SWITCHCAPVCC, 0x3C)` 初始化 OLED 屏幕。如果成功，返回 `true`；如果失败，则返回 `false`，并输出错误信息。

### 4. 设置显示内容：

- `display.setTextColor(SSD1306_WHITE)`：设置文本颜色为白色。
- `display.setTextSize(1)`：设置文本大小为 1（可以调整为更大的值以放大文本）。
- `display.setCursor(0, 0)`：设置文本显示的起始位置为屏幕的左上角（坐标为 (0, 0)）。
- `display.println("Hello, OLED!")`：显示文本 "Hello, OLED!"，并换行。

### 5. 刷新显示：

- `display.display()` 刷新屏幕，将之前设置的内容显示出来。

## 常用方法：

- `display.clearDisplay()`：清空屏幕，通常在显示新内容之前调用。
- `display.setTextSize(size)`：设置显示文本的大小，`size` 是文本的倍数，常见值为 1、2、3 等。
- `display.setTextColor(color)`：设置文本的颜色，常见颜色是 `SSD1306_WHITE` 或 `SSD1306_BLACK`。
- `display.setCursor(x, y)`：设置光标的位置，`x` 是横坐标，`y` 是纵坐标，通常以像素为单位。
- `display.println(text)`：显示文本并换行。
- `display.display()`：刷新屏幕，将内容显示出来。

## 总结

使用 `Adafruit_SSD1306` 库可以方便地在 OLED 屏幕上显示文字、图形等内容。在初始化时，你需要指定 OLED 屏幕的尺寸和 I2C 地址，并使用适当的函数来显示文本或图形。