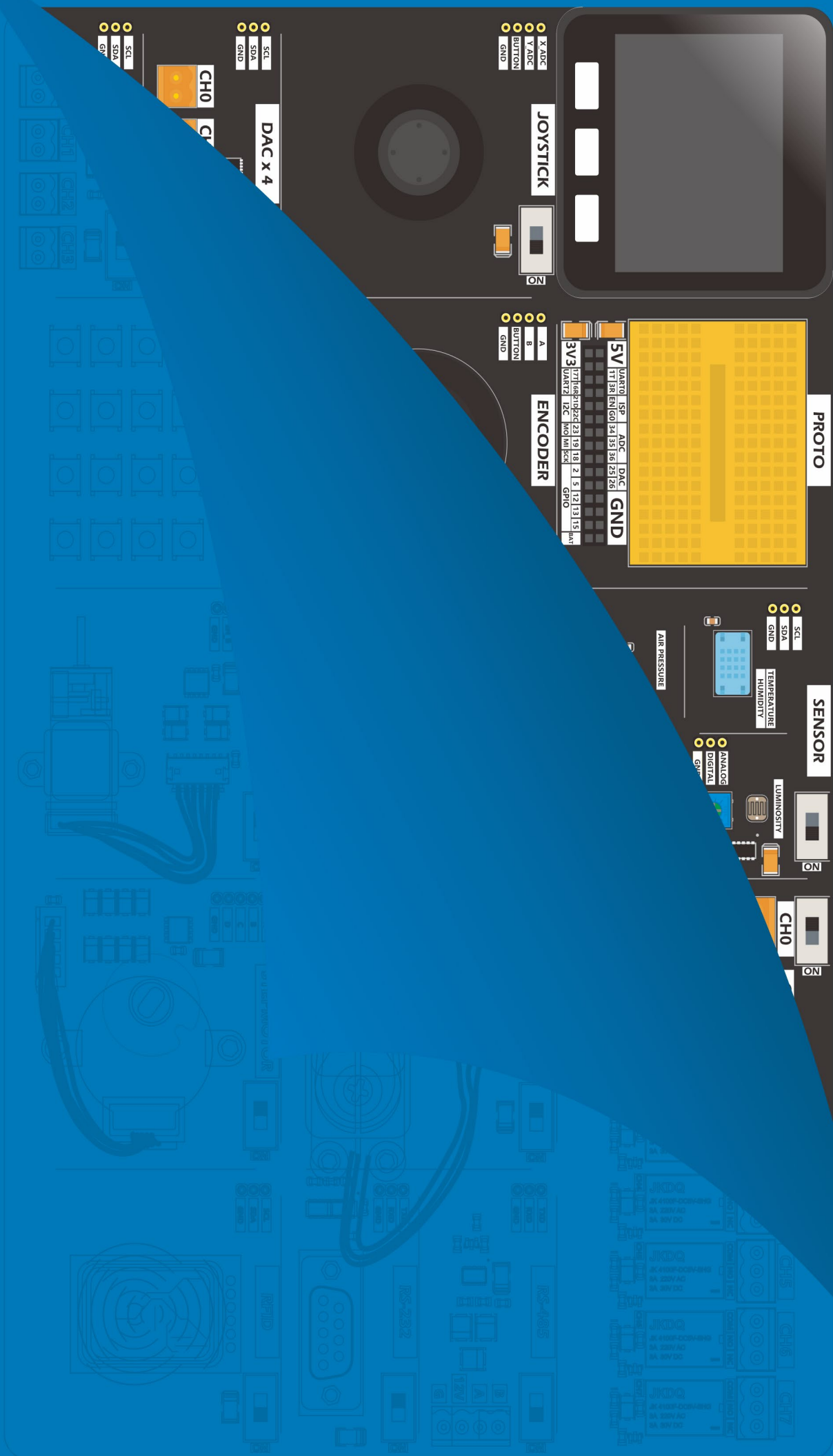


M5STACK

DEMO BOARD



M5STACK



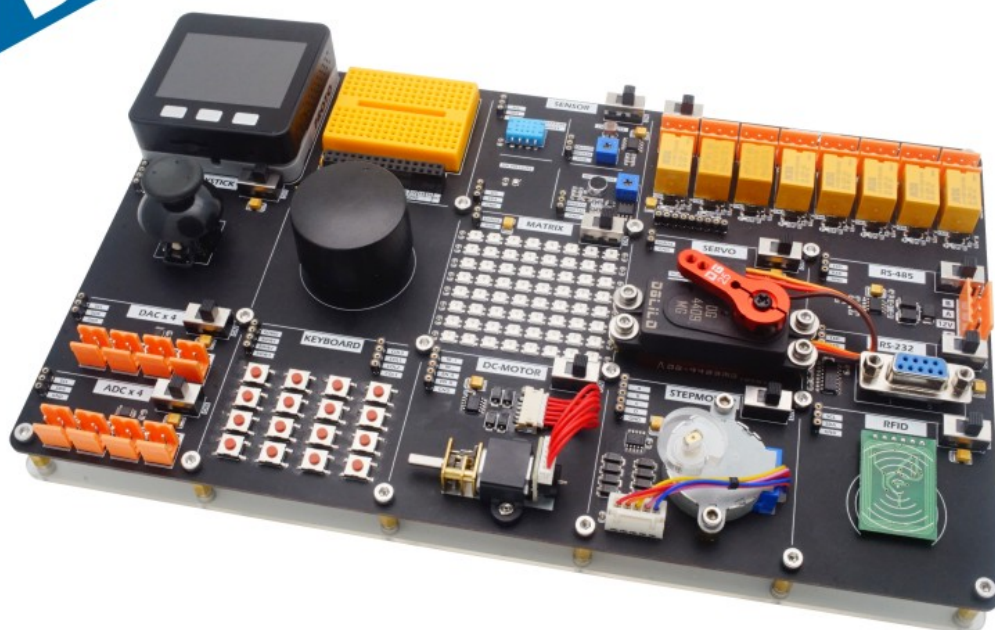
目录

目录.....	1
产品介绍.....	2
描述	2
核心控制器	3
硬件布局	4
配件说明	5
ARDUINO IDE.....	6
环境搭建.....	6
常用 API	17
LCD.....	17
按键.....	20
扬声器.....	21
模块驱动.....	23
Relay 模块	23
Microphone 模块	25
Light 模块	28
Joystick 模块	30
Step Motor 模块	33
DC-Motor 模块	38
Servo 模块	41
Keyboard 矩阵按键	43
Encoder 旋转编码器	45
DHT12 温湿度检测	49
BMP280 气压检测	52
MATRIX /LED 矩阵模块	55
RFID 无线射频识别	57
DAC 数字-模拟转换模块	61
ADC 模拟-数字转换模块	63
RS-232 转换模块	69
附录.....	70
Example Github	71
Arduino API	72
Document & Datasheet	72

NEW
DEMO BOARD



M5STACK



产品介绍

描述

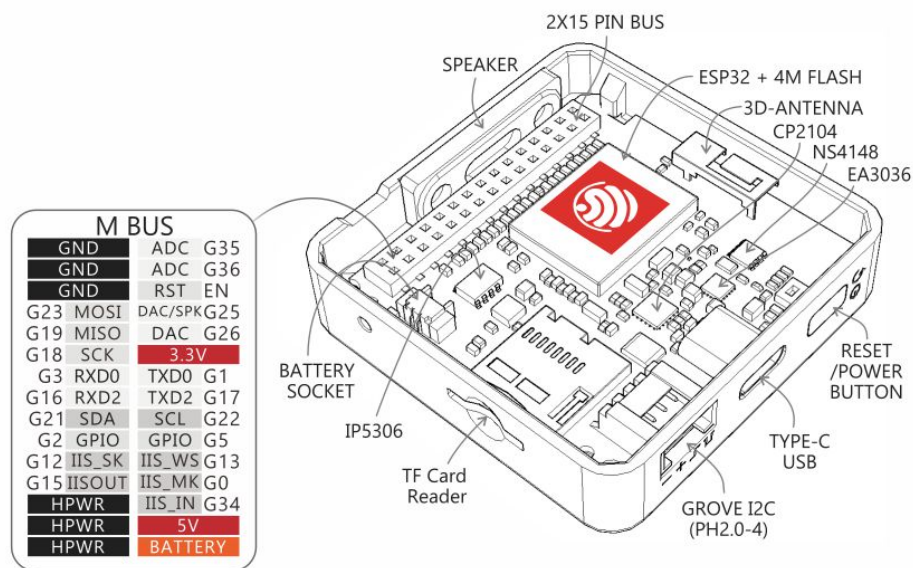
DemoBoard 是一款多功能 IoT 学习开发板. 采用 M5Stack 控制器系列中的 BASIC 作为控制核心, 内置 ESP32 主控芯片, 完全兼容 M5Stack 模块堆叠与硬件拓展体系.

配备丰富的外设模块, 其中有, 环境检测相关传感器, 摇杆控制、旋转编码器、矩阵按键、无线射频识别, 机械运动控制 (包含三种电机驱动方式), 三色灯 LED 灯板、继电器控制, 集成多组 ADC、DAC 转换电路, 支

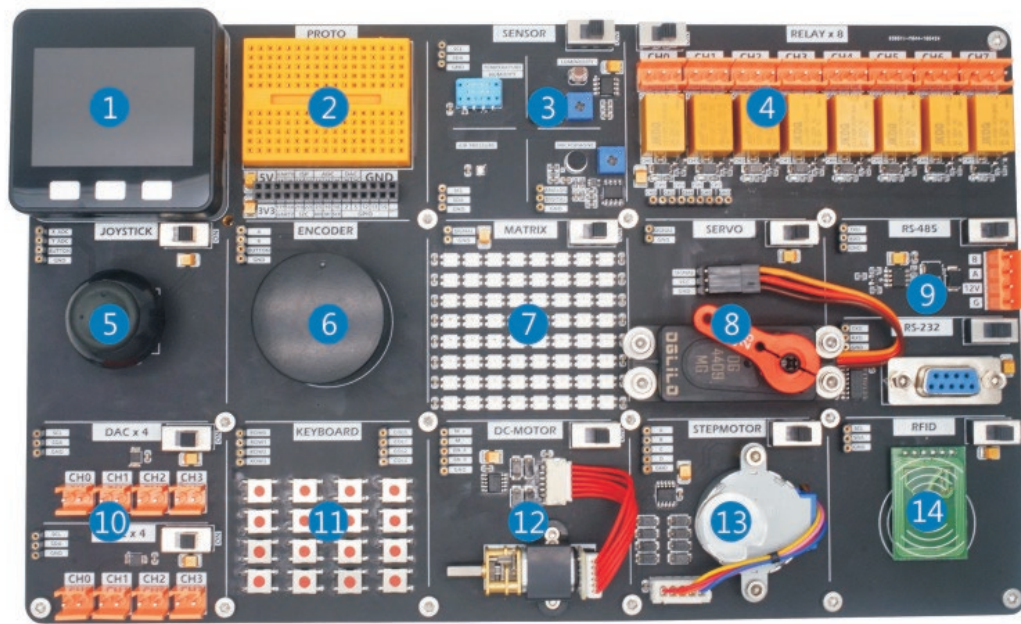
持 RS485、RS232 总线通信。 每一个模块拥有独立电源开关。 结合自带物联网属性的 M5 主控用作为控制核心，涵盖了“声、光、电、力”学等多个方面内容的 DemoBoard 开发板会是你学习硬件、编程的一大利器。

核心控制器

BASIC 控制器的正面为一块 2 英寸的彩色 TFT LCD 屏幕,分辨率为 320x240。内置物联网芯片 ESP32，集成蓝牙和 WiFi 模块于一身。ESP32 搭载了双核 32-bit MCU，主频高达 240 MHz，计算能力高达 650 DMIPS。芯片拥有丰富管脚资源同时，集成了丰富的硬件外设，包括电容式触摸传感器、霍尔传感器、低噪声传感放大器，SD 卡接口、以太网接口、高速 SDIO / SPI、UART、I2S 和 I2C 等。



硬件布局

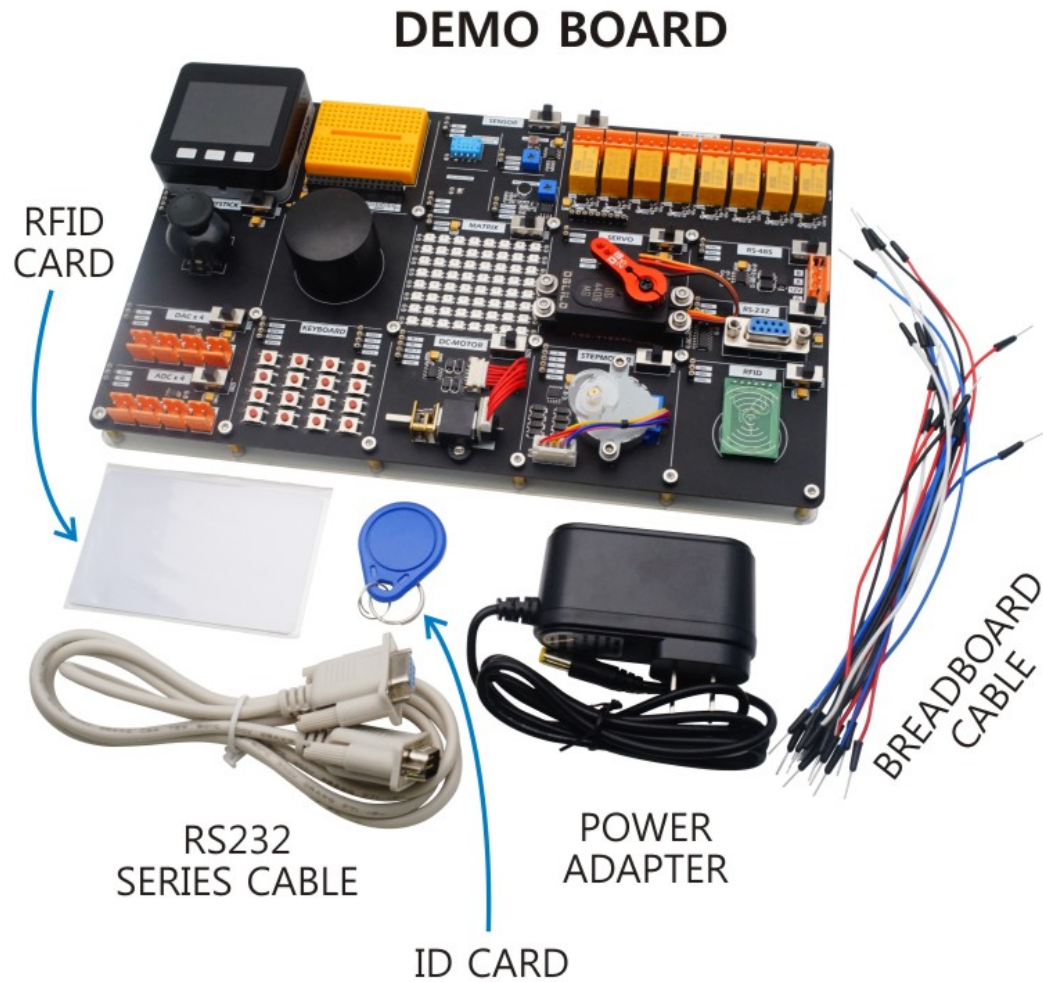


- 1: CORE 2: PROTO 3: SENSOR 4: RELAYx8
- 5: JOYSTICK 6: ENCODER 7: MATRIX 8: SERVO
- 9: RS-485/RS-232 10: DACx4/ADCx4 11: KEYBOARD
- 12: DC-MOTOR 13: STEPMOTOR 14: RFID

- 1 - 采用 M5Core 作为控制核心，兼容 Module 堆叠、Unit 拓展体系。
- 2 - Proto 板、M5-BUS 总线拓展
- 3 - 环境传感器系列（温度、湿度、气压、光线、麦克风）
- 4 - 8 路继电器输出
- 5 - 摇杆输入
- 6 - 旋转编码器
- 7 - 8x8 矩阵 RGB LED
- 8 - 10KG 舵机
- 9 - RS-458, RS232 通信功能
- 10 - 4 路 DAC, 4 路 ADC
- 11 - 4x4 按键矩阵
- 12 - 直流电机（带反馈）
- 13 - 四相五线制步进电机
- 14- 无线射频识别读卡器 (RFID)

配件说明

开发板提供一系列的配套硬件，能够帮助用户测试板上各个模块的功能。



- 1x 12V 电源适配器（在使用开发板前，请连接 12V 电源适配器为开发板供电）
- 1x RS232 连接线（用作 RS232 端口连接）
- 1x RFID CARD（用于测试无线射频模块）
- 1x ID CARD（用于测试无线射频模块）
- 16x 面包线（用于板上跳线，连接各个硬件模块的接口）

ARDUINO IDE

环境搭建

软件安装

本教程中所使用的开发环境为 Arduino IDE，本章节将向你介绍如何下载相关的软件，资源库，以及一些基本的配置

Arduino IDE 是一款开源软件，具备跨平台能力的它可以在 Windows，Mac OS X 和 Linux 上运行。（本教程所演示的软件安装操作基于 Windows10 操作系统）

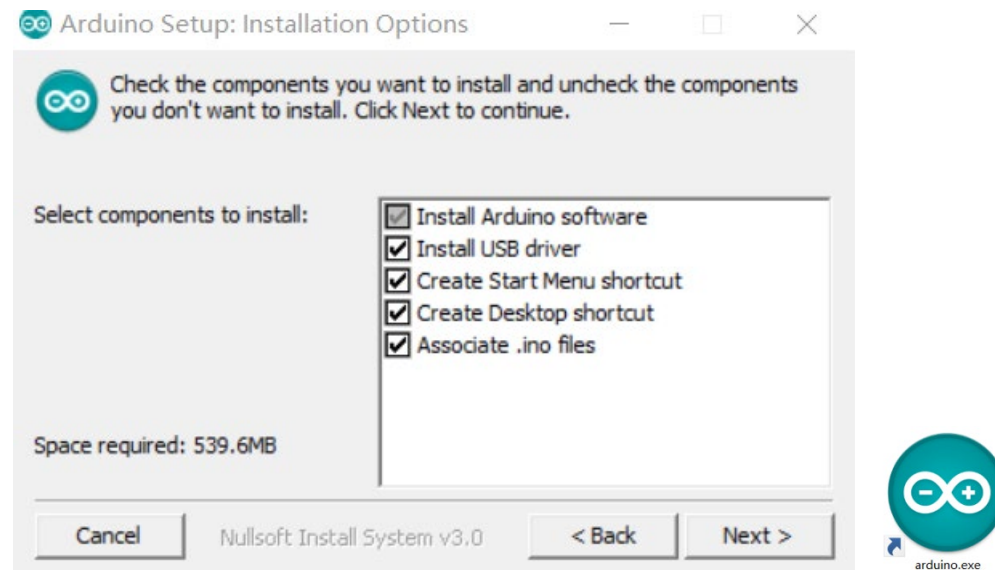
首先访问(<https://www.arduino.cc/en/Main/Software>)，进入下载页面，选择对应自己操作系统的安装包进行下载。



Windows10 的用户可以点击，“Windows Installer, for Windows XP and up” 选项，页面将会切换至捐赠与下载页面，如需要捐赠可以点击“CONTRIBUTE & DOWNLOAD”，仅下载安装包文件则点击“JUST DOWNLOAD”。

下载完成后，双击打开安装包，根据安装提示，选择同意授权协议，关于安装的配置选项与安装路径，若无特殊需求可以按照默认配置。根据提示依次

点击下一步，直到安装完成。按照默认配置，安装完成后将会在桌面上创建 Arduino 的快捷方式图标，双击它，即可启动 Arduino IDE。



基本介绍

位于程序顶部的是选项卡以及功能菜单栏，提供了一系列的操作和配置选项，分别是编译、上传、新建、打开、保存、串口监视器功能



位于中间的是编程区域，用于代码的编辑。位于编程区域下方的是日志输出区域，启用后，它将向你展示当前 Arduino IDE 的工作状态，如正在编译或是正在上传程序，并输出编译与上传过程中的日志信息，方便进行程序调试与错误排查。


```
sketch_nov19a
1 void setup() {
2   // put your setup code here, to run once:
3
4 }
5
6 void loop() {
7   // put your main code here, to run repeatedly:
8
9 }
```

1 M5Stack-Core-ESP32, QIO, 80MHz, Default, 921600, None on COM9

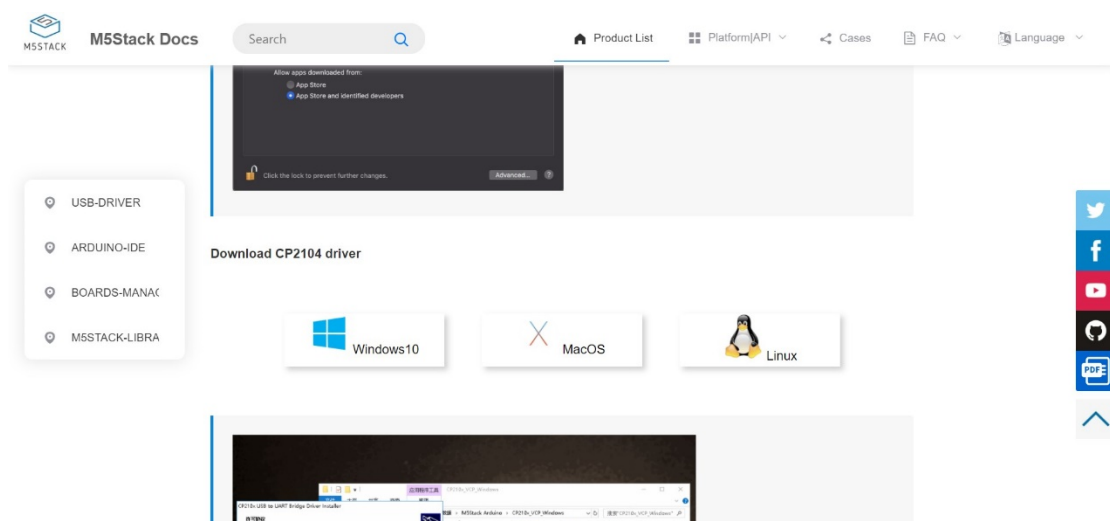
安装 CP2104 驱动

DemoBoard 所使用的主控器为 BASIC，在使用前，我们需要安装与其匹配的 USB 驱动 (CP2014)，

访问 M5Stack 的官方文档中的 Arduino 环境配置教程页。

(https://docs.m5stack.com/#/en/arduino/arduino_development)

选择目录中的“Install USB Driver”选项，跳转至页面底部，选择相应操作系统的 CP210X 驱动程序压缩包 进行下载。

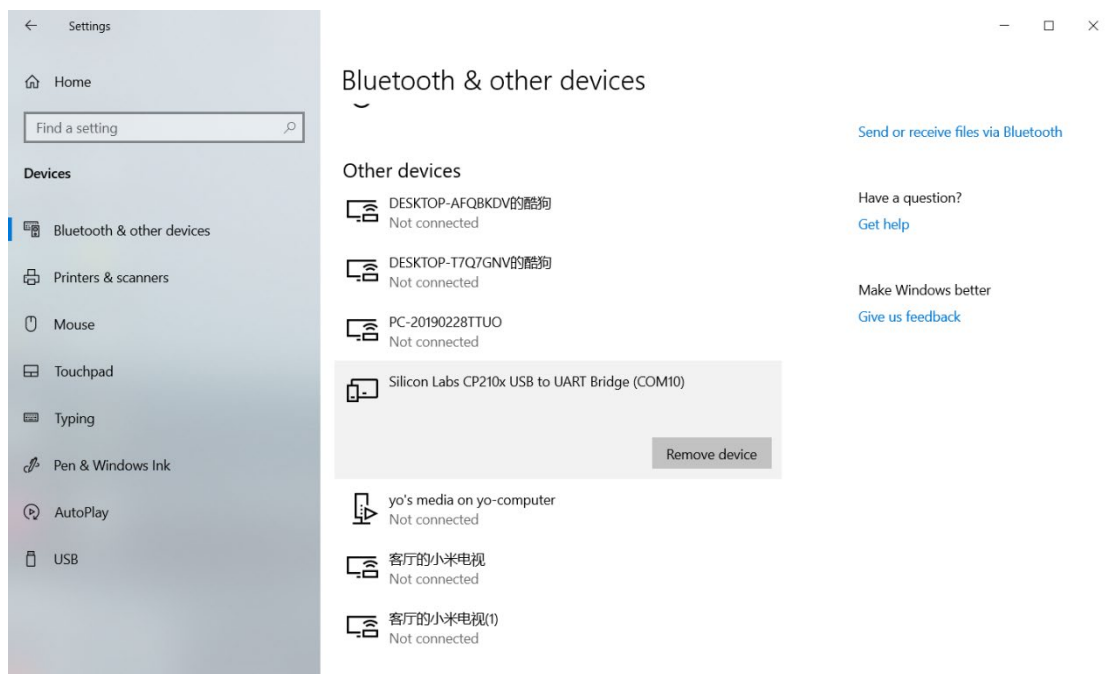


将下载完成的压缩包解压，根据操作系统位数，打开相应的安装程序（64 位操作系统选择_x64，32 位操作系统选择_x86），并按照提示依次点击下一步，直到安装完成。

CP210x USB to UART Bridge Driver Installer



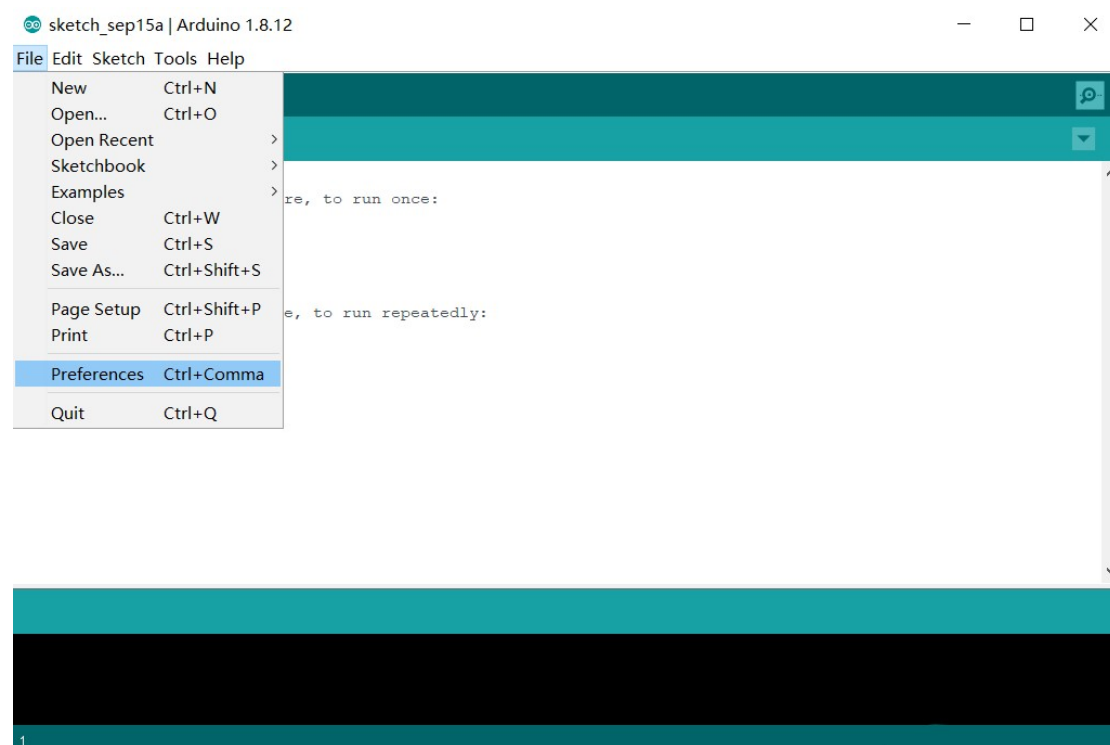
完成安装后，使用 Type-C 数据线将 M5Stack 设备连接至个人电脑。从 Windows 设置页面中选择“设备”选项，查看当前已连接的设备信息。如图 2-11 所示，当前 M5Stack 设备已经成功连接，当前所使用的端口为 COM10。表示 USB CP210X 驱动已经安装成功，且设备运行正常。



ESP32 板管理

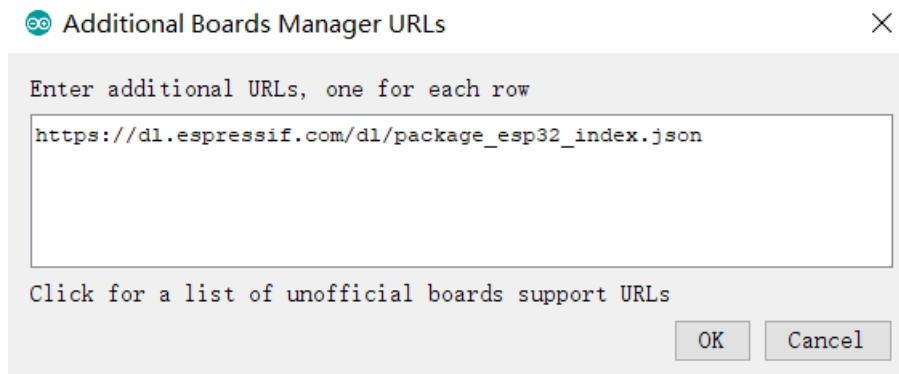
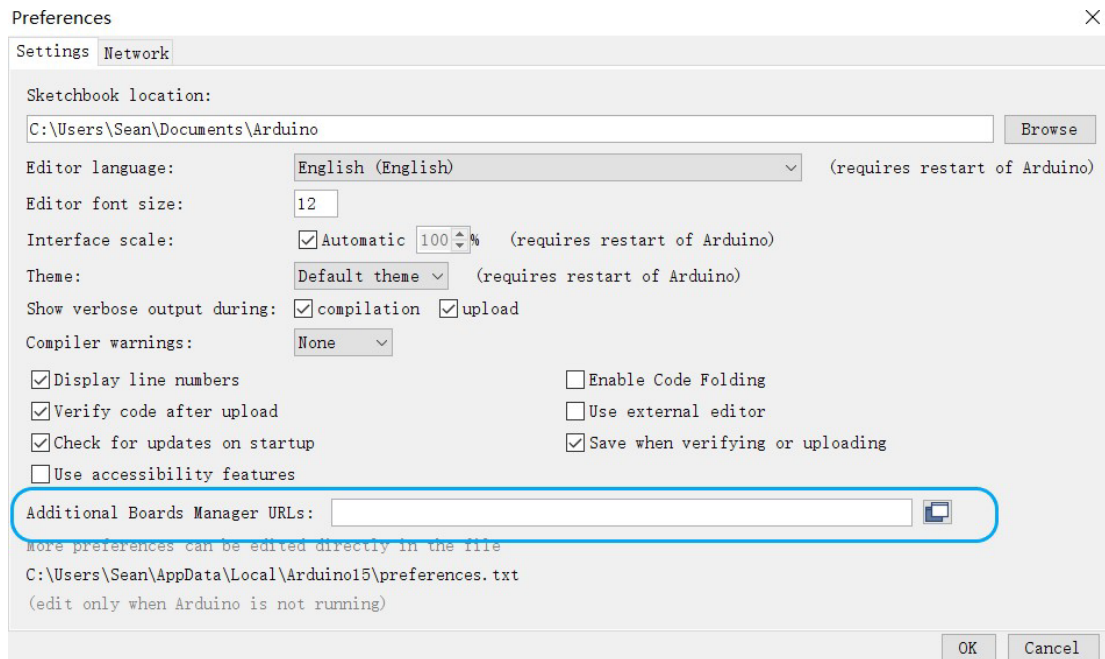
Arduino IDE 除了支持 Arduino 官方推出的开发板(如 Arduino Uno)以外，对市面上的很多款控制器芯片均有支持。在进行不同的设备开发时，我们需要在配置选项指定当前使用的开发板信息，

本小节将向你说明，如何在 Arduino IDE 中配置 M5Stack 所使用的 ESP32 板管理信息。点击 Arduino IDE 页面上方的“文件”选项卡，并打开“首选项”设置。



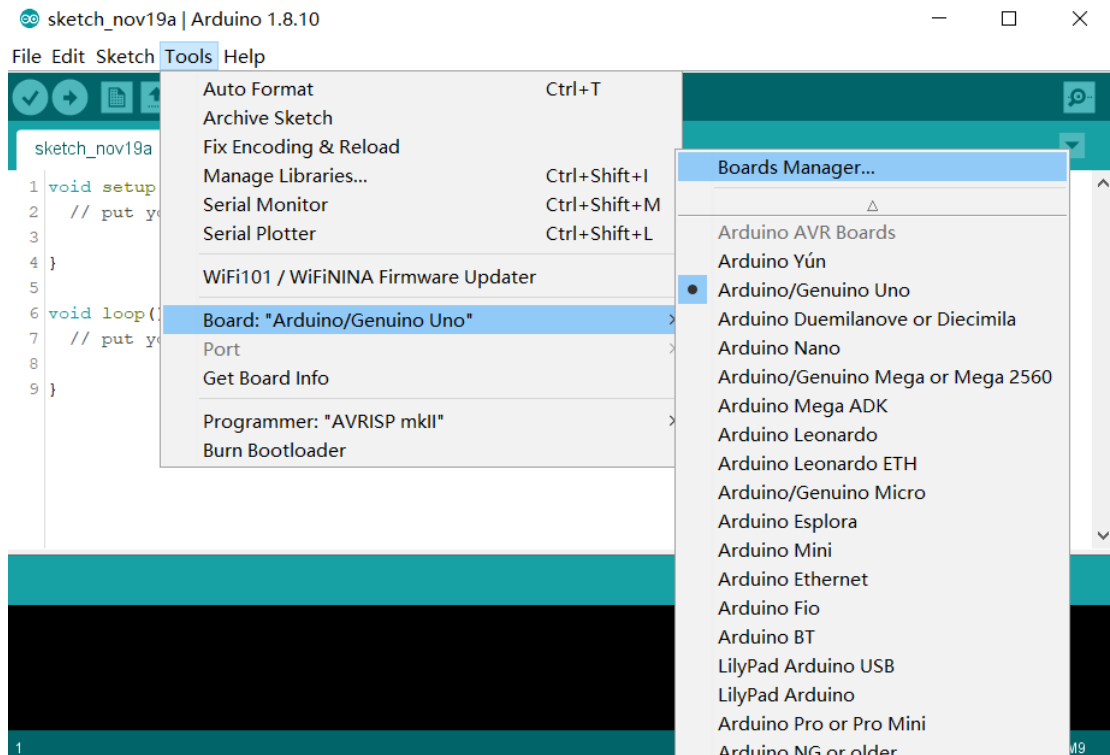
点击“添加板管理地址”栏右侧的添加按钮，在弹出的窗口中输入下方的 URL 地址，并点击 OK 保存。

https://dl.espressif.com/dl/package_esp32_index.json

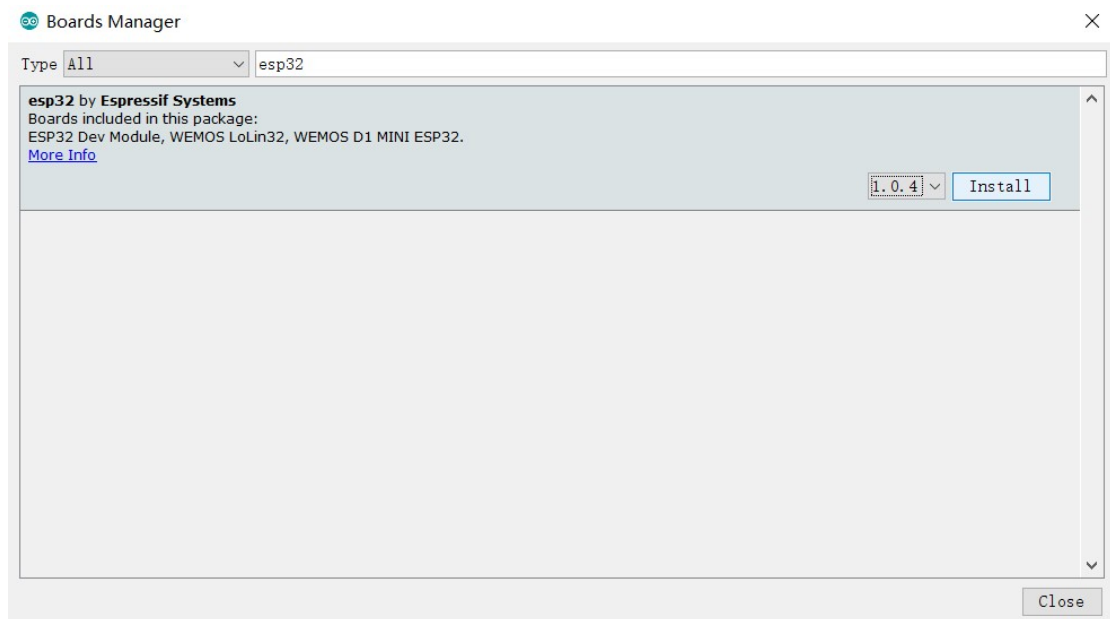


除此之外，在“首选项”设置页面中，你还可以设置一些显示参数，例如编辑器文字大小、以及添加行号显示，编译/烧录日志提示等辅助内容。

保存设置后，选择“工具”选项卡，鼠标指向“开发板”选项，在其下级目录中找到“开发板管理器”，单击打开。

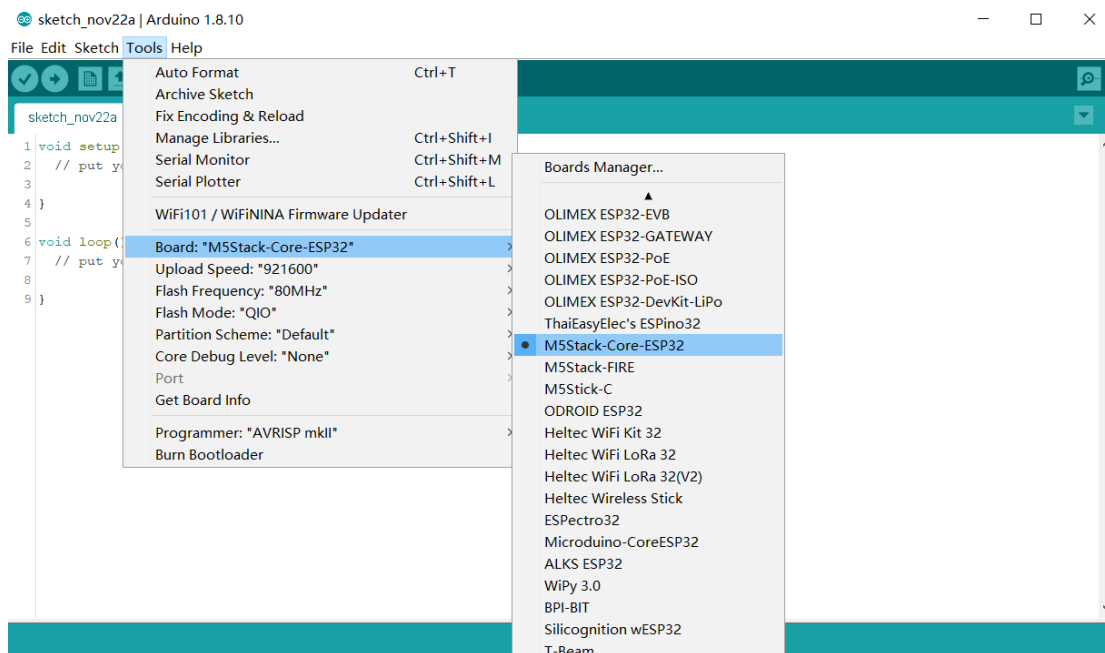


在弹出的板管理窗口中，搜索“esp32”，则会出现“esp32 by Espressif Systems”开发板内容。



点击版本号选项能够切换安装不同的版本，为了获得更好的支持，建议下载最新的版本。

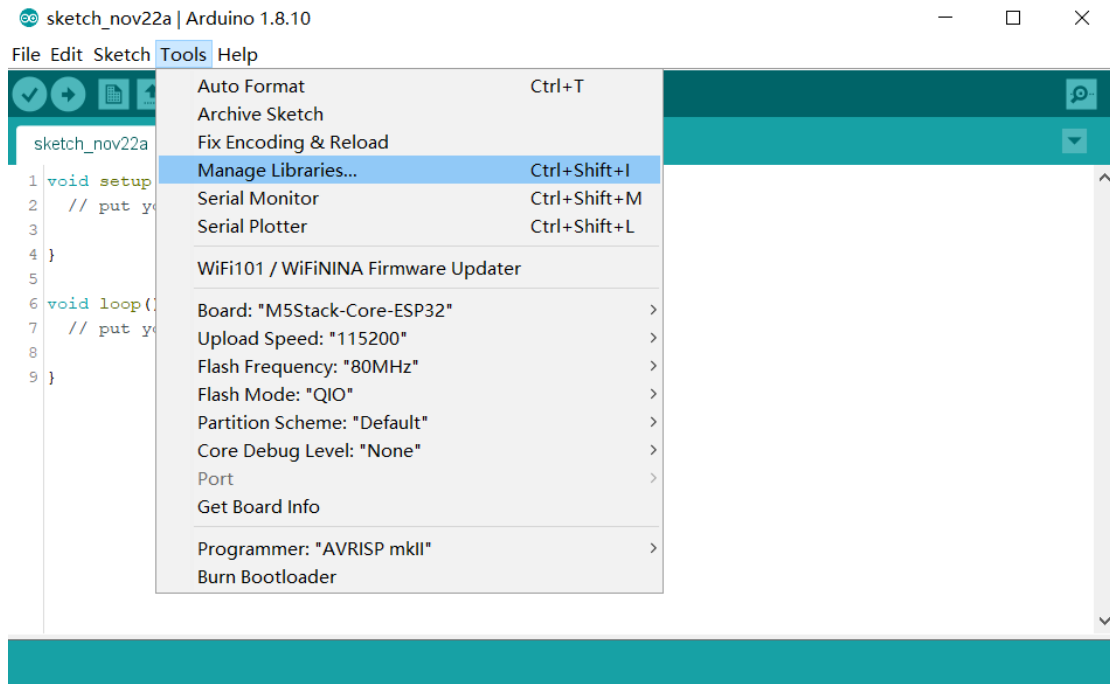
安装完成后，关闭板管理器，回到-->“工具”-->“开发板”选项，在目录的下方，你会发现出现了“M5Stack-Core-ESP32”等 M5Stack、ESP32 相关板资源选项。



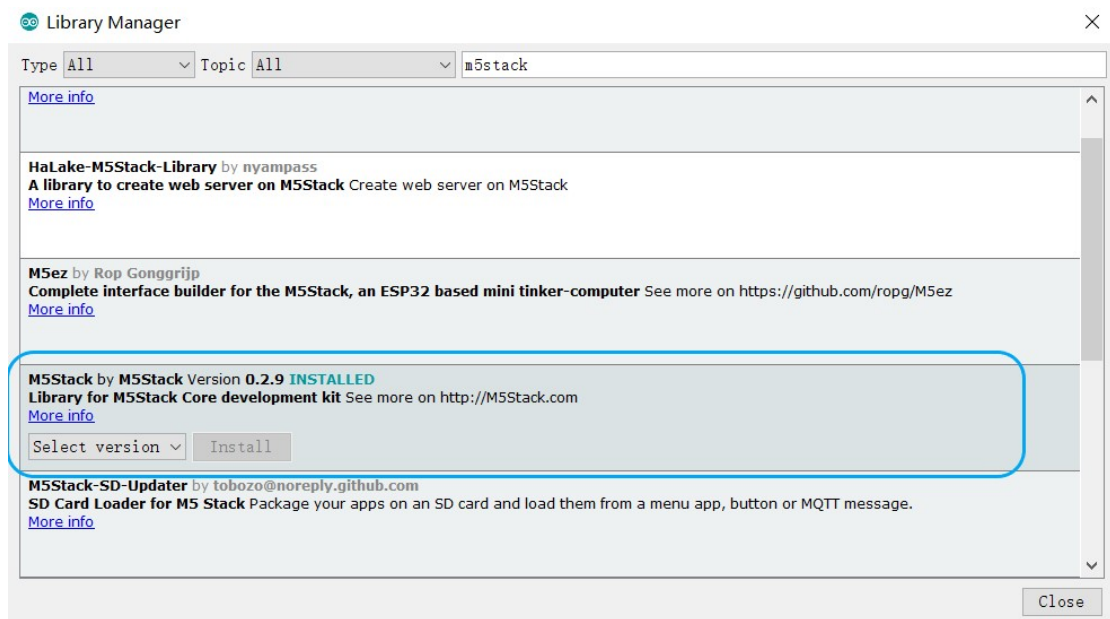
选择“M5Stack-Core-ESP32”，连接设备设置端口等参数。“Upload Speed”参数能够设置上传程序时，Arduino IDE 传输可执行文件到设备的通信速度 (bps)。当在代码调试阶段，往往需要反复的进行程序烧录，为了加快开发进度，可以选择较高的速度来减少烧录的时间。

安装 M5Stack 库

M5Stack 官方提供了适配 M5Stack 设备的 Arduino 程序库，程序库内封装集成了一系列 M5Stack 设备硬件 (如传感器、LCD 屏幕等) 的功能支持与驱动。通过该程序库用户能够非常方便快捷的进行 M5Stack 设备开发。选择“工具”选项卡，打开“管理库”选项，单击打开库管理器。



在库管理器中搜索“M5Stack”，就会检索到一系列有关 M5Stack 的程序库，在这里我们选择由官方维护，带有的“by m5stack”字段的程序库。选择最新版本，进行安装。



完成以上步骤，M5Stack & Arduino IDE 的开发环境就已经搭建完成了。

Hello World

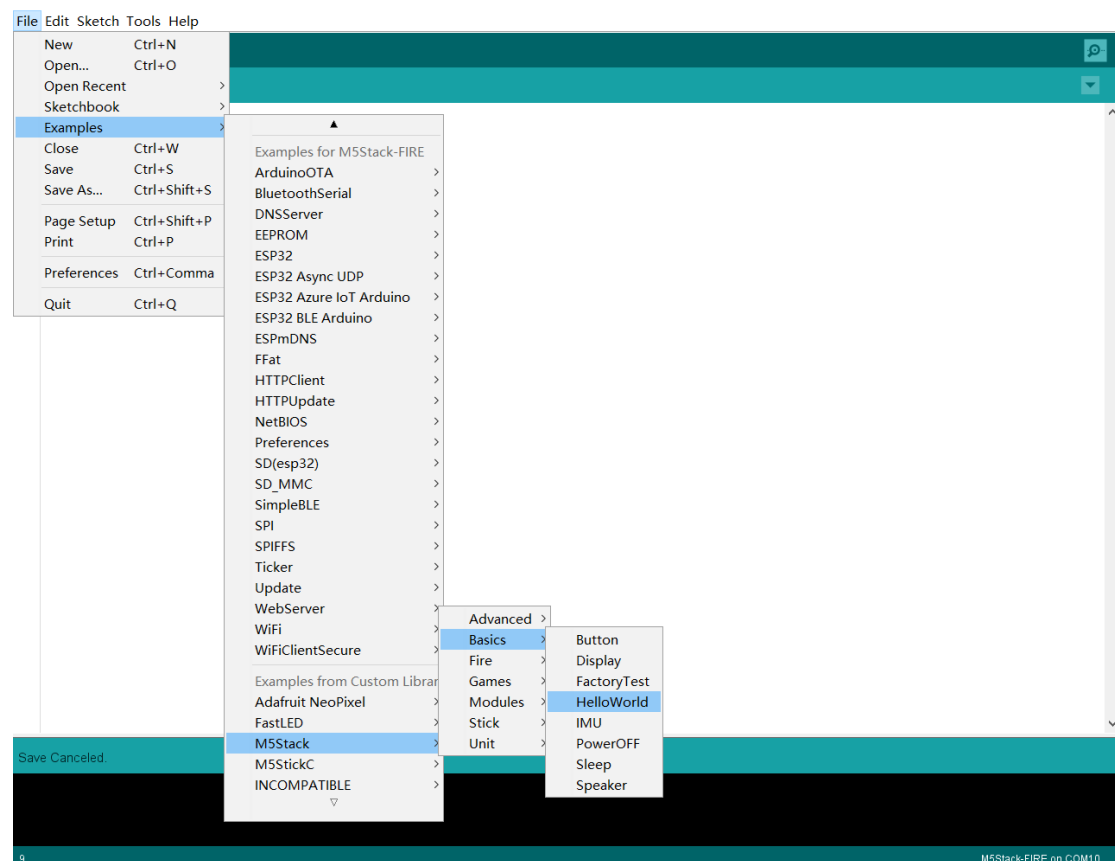
位于程序顶部的是选项卡以及功能菜单栏，提供了一系列的操作和配置选项，位于中间的是编程区域，用于代码的编辑。

位于编程区域下方的是日志输出区域，启用后，它将向你展示当前 Arduino IDE 的工作状态，如正在编译或是正在上传程序，并输出编译与上传过程中的日志信息，方便进行程序调试与错误排查。

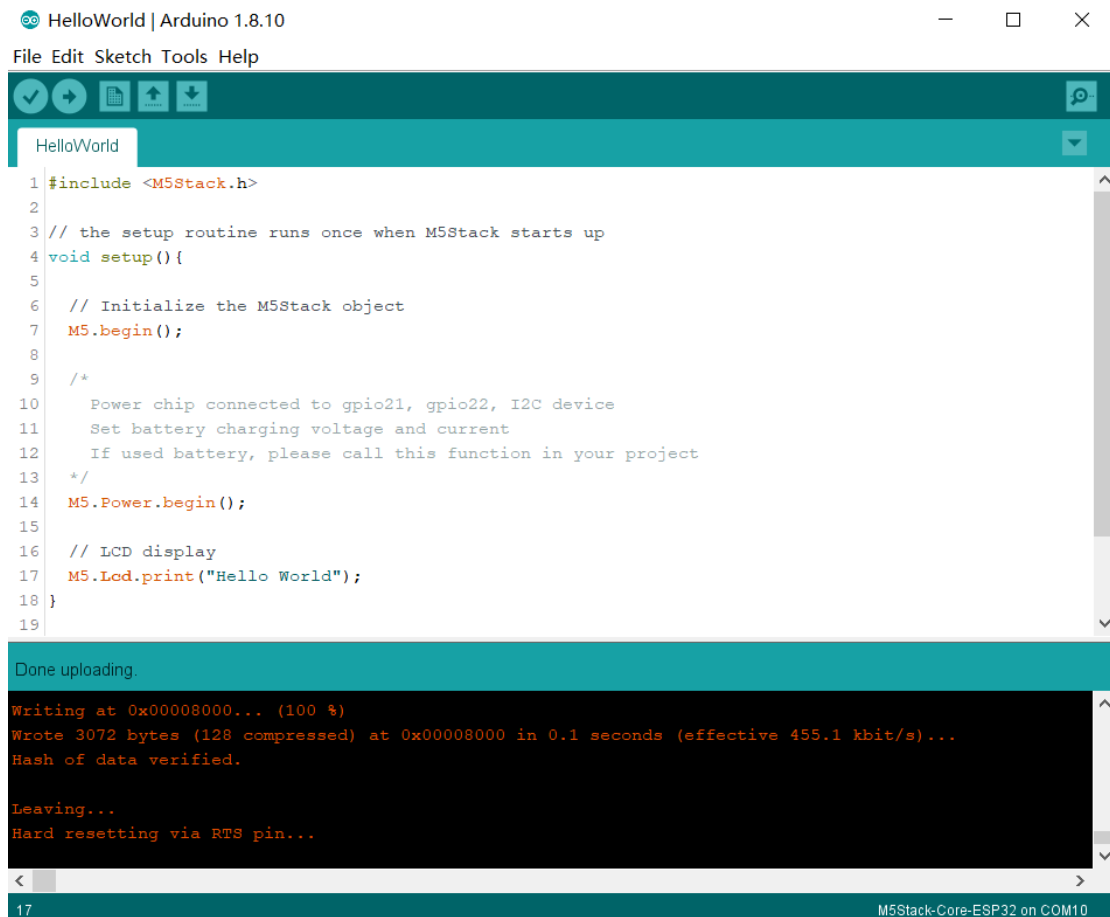
M5Stack 程序除了提供一系列功能函数以外，还提供了许多的编程案例供用户参考使用。

在完成开发环境部署后，我们可以尝试着运行一个简单的案例程序，来熟悉硬件的使用流程。

点击 Arduino IDE 页面上方的“文件”选项卡-->“案例程序”-->“M5Stack”-->“Basics”-->“HelloWorld”打开案例程序。



通过 Type-C 数据线将 M5Stack 设备连接至电脑，在“工具”选项卡中的“端口”选项选择相应的端口，并确认开发板等配置信息正确。



点击功能菜单栏上的“上传”按钮，程序将自动开始编译。当编译通过后，程序将会自动进行上传，日志将输出此时程序上传的进度百分比，此时请保持电脑与 M5Stack 设备的连接，直到程序上传完成。当状态栏提示“Done uploading.”则表示程序已经上传完成。

此时查看 M5Stack 设备，你会发现其 LCD 屏幕的左上角显示出“Hello World”字样屏幕和我们程序内容预期的一样，成功的进行了显示。



常用 API

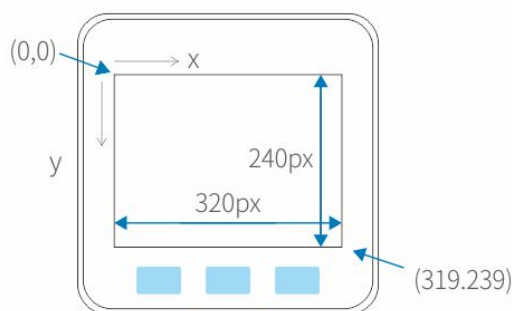
LCD

描述

在学习环境配置时，我们运行了“HelloWorld”案例程序，成功的在 LCD 上显示文字，这块彩色 LCD 屏幕能做的远远不止这些。

M5Stack 库为 LCD 的驱动提供了一系列常用的程序 API。接下来我们将学习如何使用这些 API，在 M5Stack 设备的 LCD 屏幕上自由的打印内容、绘制图案。

这块“320x240”像素的彩色 LCD 屏幕。在实际使用时，你可以将屏幕看作一个平面坐标系，横向为 x 轴，纵向为 y 轴。从左上角的坐标原点 (0, 0) 开始，到右下角的 (319, 239) 对角线所形成的一个矩形是屏幕的显示区域，当内容的坐标离开这个范围时，将无法正常显示。



文本显示

在程序上，我们除了能够修改显示的文本内容，还能够设置内容显示的位置，以及字体的字号大小。通过设置文本光标，能够指定文本内容的一个字符所处的位置。通过设置字号数值，能够调整字体显示的大小。

设置光标位置：

```
void M5.lcd.setCursor(uint16_t x, uint16_t y);
```

说明：设置文本光标至坐标(x,y)

参数：

`uint16_t x`: x 轴坐标
`uint16_t y`: y 轴坐标

返回值:无

使用案例: `M5.lcd.setCursor(40, 60);`

设置字号大小:

`void M5.lcd.setTextSize(uint8_t size);`

说明：设置文本字号大小

参数：

`uint8_t size`: 字号大小(允许输入范围 1~7)

返回值:无

使用案例: `M5.lcd.setTextSize(3);`

打印文本信息:

`int M5.lcd.print(val);`

`int M5.lcd.print(val,format);`

`int M5.lcd.println(val);`

`int M5.lcd.println(val,format);`

说明：将指定信息打印到 LCD 屏幕上。

参数：

`val`: 输出值，允许是整数、浮点数、字符、字符串。

`format`: 指定输出格式、BIN（2 进制）、OCT（8 进制）、DEC（10 进制）、HEX（16 进制）。输出浮点数时，可以指定小数点后保留位数。

返回值:内容的所占的字节数

使用案例:


```
M5.lcd.print(78); //输出结果 78

M5.lcd.print(1.23456); //输出结果 1.23 (默认情况下, 保留小数点后两位)

M5.lcd.print("M"); //输出 M

M5.lcd.print("Hello M5Stack"); //输出 Hello M5Stack

M5.lcd.print(78, HEX); //输出 4E

M5.lcd.print(1.23456, 4); //输出 1.2346
```

案例程序

控制 LCD 动态显示文本, 间隔变化显示“Hello World”和“Hello M5Stack”。

```
#include <M5Stack.h>

void setup(){

    M5.begin();
    M5.Lcd.setTextSize(3);
}

void loop() {
    M5.Lcd.setCursor(40, 60);
    M5.Lcd.print("Hello World");
    delay(1000);
    M5.Lcd.clear(BLACK);
    M5.lcd.setCursor(40, 100);
    M5.Lcd.print("Hello M5Stack");
    delay(1000);
    M5.Lcd.clear(BLACK);
}
```

按键

描述

控制的面板上提供了三个可编程物理按键，配合现有的 API 可以实现短按，长按等交互操作。

按键单击判断：

```
uint8_t M5.BtnA.wasPressed();
```

说明：当按下按键时，函数返回一次"1"值，之后置位为"0"。松开按键时，函数返回值一直为"0"。

参数：无

返回值：无

使用案例：

```
if (M5.BtnA.wasPressed()) {  
    M5.Lcd.printf("Button A was pressed.");  
    delay(1000);  
}
```

按键长按判断：

```
uint8_t M5.BtnA.pressedFor(int32_t ms);
```

说明：指定按下按钮时长，实际当按下时长大于设置时长则返回"1"，否则返回"0"。

参数：

`int32_t ms`: 设定长按时间

返回值：无

使用案例：

```
if (M5.BtnA.pressedFor(2000)) {  
    M5.Lcd.printf("Button A was pressed for more than 2 second  
s.");  
    delay(1000);  
}
```

状态刷新：

```
void M5.update();
```

说明：更新按键 A、B、C、扬以及扬声器的状态。注意：在使用按键程序时，你需要将"M5.update();"添加到程序的循环中，让按键值随着程序的运行不断刷新，否则会出现程序按键仅一次有效的现象。

参数：无

返回值：无

使用案例：

```
M5.update();
```

扬声器

描述

控制器上集成的扬声器发出各式各样的音调，用于音频播放，或是功能提示音。

播放指定音调：

```
void M5.Speaker.tone(uint16_t freq, uint32_t duration);
```

说明：播放指定频率音频，并支持设定播放时长。

参数：

uint16_t freq：设置播放声音频率

uint32_t duration：

返回值：无

使用案例：

```
M5.Speaker.tone(440, 1000);
```

注意：普通的人的听力范围在 20Hz~20KHz. 大于或小于该范围的声音，可能无法被人们识别。过高频率使用或长时间的高频播放容易对扬声器硬件造成损害，因此，在填写参数时请注意合适的范围。

案例程序

结合按键功能，控制扬声器发出不同的声音。

```
#include <M5Stack.h>

void setup() {

  M5.begin();
  M5.Lcd.setTextSize(3);
  M5.lcd.setCursor(40, 100);

  M5.Lcd.print("M5Stack Speaker test");
}

void loop() {
  if(M5.BtnA.wasPressed()) {
    M5.Speaker.tone(262, 1000);
  }
  if(M5.BtnB.wasPressed())
  {
    M5.Speaker.tone(294, 1000);
  }
  if(M5.BtnB.wasPressed())
  {
    M5.Speaker.tone(330, 1000);
  }
  M5.update();
}
```

注意事项

在使用 M5Core 进行编程开发的时候，你需要留意内置硬件的引脚占用情况，避免使用同一引脚驱动不同的硬件，导致无法正常工作。以下引脚分别代表了其在 M5Core 内的应用情况，（例如其中 GPIO32 为 LCD 屏幕占用引脚）

ESP32 Chip	GPIO23	GPIO19	GPIO18	GPIO14	GPIO27	GPIO33	GPIO32	GPIO4
ILI9341	MOSI	/	CLK	CS	DC	RST	BL	
TF Card	MOSI	MISO	CLK					CS

查看完整引脚映射资料请访问

<https://docs.m5stack.com/#/en/core/basic>

模块驱动

Relay 模块

描述

Relay 是一个继电器控制模块，当继电器内部的线圈通电，将产生磁场力吸附开关动作，进而实现开关控制。这样一个小电流控制大电流的电气线路方案，能够控制 DC/3A-30V 或 AC/3A-220V 级别的线路通断。

每一个单独的继电器提供了三个控制线路引脚，分别是公共端 COM，常闭端 NC，常开端 NO。默认状况下 NC 与 COM 默认连接，NO 与 COM 默认断开，当继电器线圈通电，二者与公共端 COM 的开关关系将反转。

程序案例

```
#include <M5Stack.h>

const int In_0 = 23;
void setup() {
    // put your setup code here, to run once:
    pinMode(In_0,OUTPUT);
}

void loop() {
    // put your main code here, to run repeatedly:
    digitalWrite(In_0, LOW);

    delay(1000);

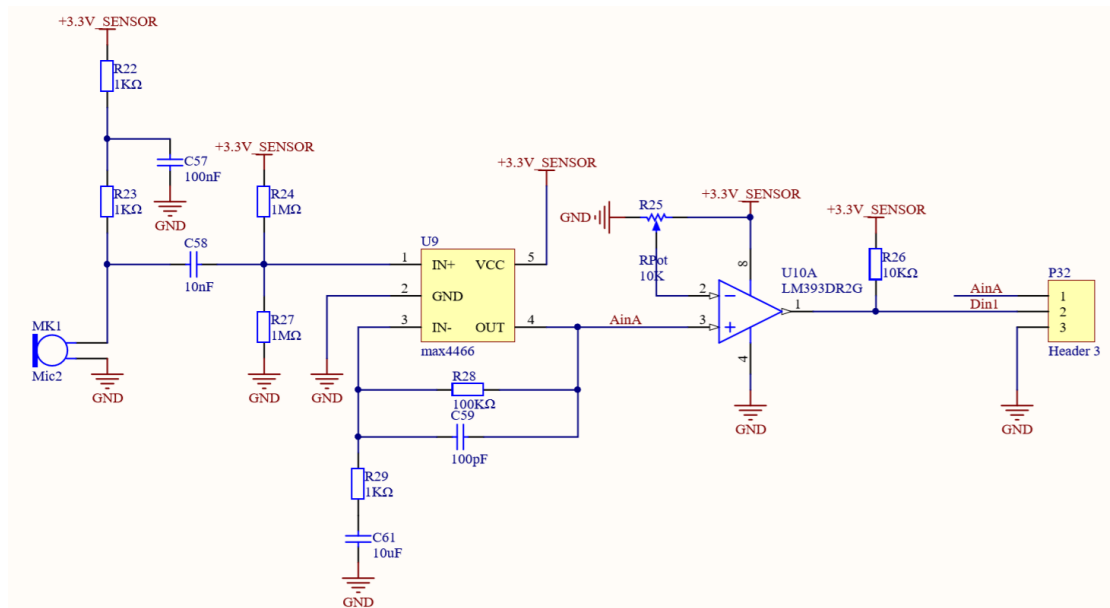
    digitalWrite(In_0, HIGH);

    delay(1000);
}
```

Microphone 模块

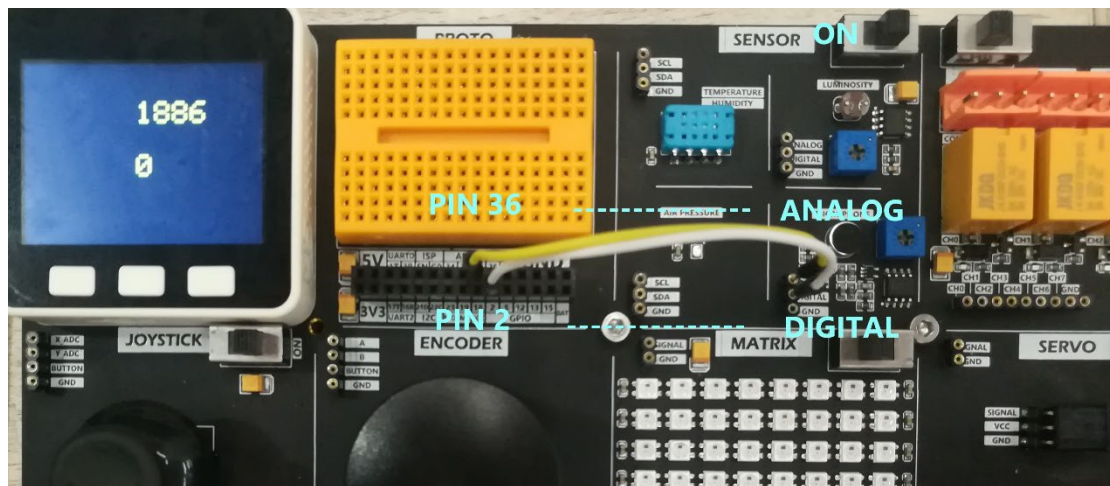
描述

Microphone 模块内部集成 MAX4466 麦克风前置放大器，能够有效采集通过麦克风输入模拟量、数字量信息. 并提供调节电阻供用户调节识别的阈值.



硬件连接

Microphone 模块支持读取数字量和模拟量两种信号值，在进行硬件连接的时候需要注意的是，M5Core 并不是所有的 GPIO 接口都支持 AD 转换，因此使用模拟量读取功能的时候需要将接口连接至支持 ADC 的引脚（如 34/35/36），数字量读取则使用一般的 GPIO 即可。



案例

```
#include <M5Stack.h>

const int Analog = 36;
const int Digital = 2;
void setup() {
    // put your setup code here, to run once:
```

```
M5.begin();
pinMode(Digital, INPUT_PULLUP);
dacWrite(25, 0);

M5.Lcd.setCursor(100, 0, 4);
M5.Lcd.print("MICROPHONE");

}
uint16_t a_data;
uint16_t d_data;

void loop() {
    // put your main code here, to run repeatedly:
    a_data = analogRead(Analog);
    d_data = digitalRead(Digital);

    Serial.printf("Analog:%0d Digital:%0d\n", a_data, d_data);

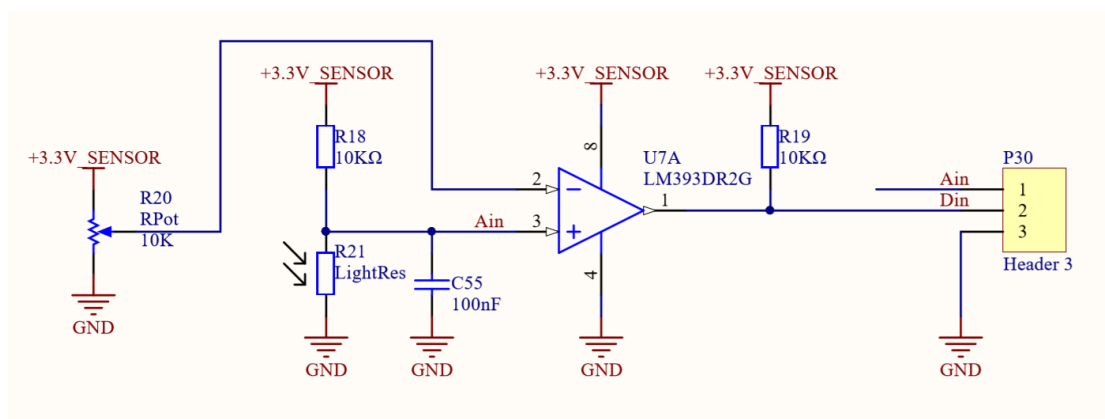
    M5.Lcd.setCursor(30, 120, 4);
    M5.Lcd.printf("Analog:%0d Digital:%0d\n", a_data, d_data);

    delay(200);
}
```

Light 模块

描述

Light 模块是一个光线感应传感器，其内部集成光敏电阻，LM393DR2G 电压比较器，能够有效采集光线输入模拟量、数字量信息. 并提供调节电阻供用户调节识别的阈值。



硬件连接

Light 模块支持读取数字量和模拟量两种信号值，在进行硬件连接的时候需要注意的是，M5Core 并不是所有的 GPIO 接口都支持 AD 转换，因此使用模拟量读取功能的时候需要将接口连接至支持 ADC 的引脚（如 34/35/36），数字量读取则使用一般的 GPIO 即可。



案例

```
#include <M5Stack.h>

const int Analog = 36;
const int Digital = 2;
void setup() {
    // put your setup code here, to run once:
    M5.begin();
    pinMode(Digital, INPUT_PULLUP);
    digitalWrite(25, 0);

    M5.Lcd.setCursor(100, 0, 4);
    M5.Lcd.print("LUMINOSITY");
}
uint16_t a_data;
uint16_t d_data;

void loop() {
    // put your main code here, to run repeatedly:
    a_data = analogRead(Analog);
    d_data = digitalRead(Digital);

    Serial.printf("Analog:%0d Digital:%0d\n", a_data, d_data);

    M5.Lcd.setCursor(30, 120, 4);
    M5.Lcd.printf("Analog:%0d Digital:%0d\n", a_data, d_data);

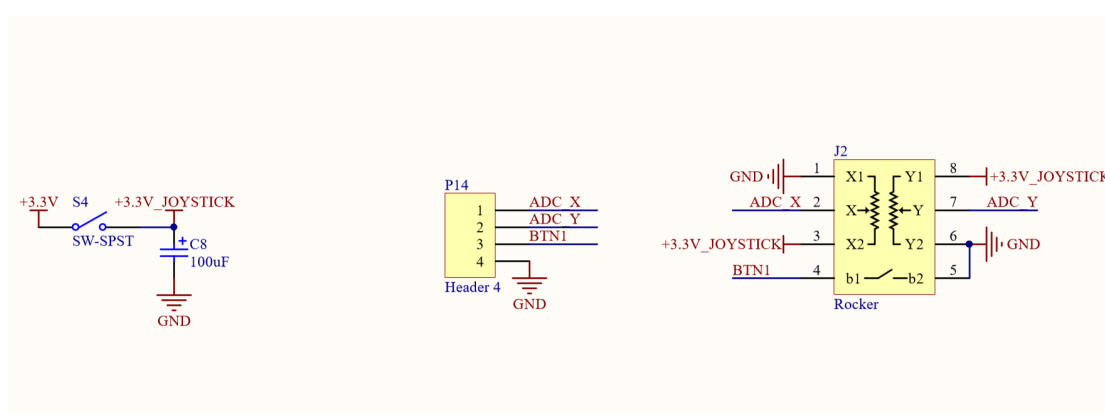
    delay(200);
}
```

Joystick 模块

描述

Joystick 是一个摇杆输入模块，该模块内置两组滑动变阻器与一个按键开关。当操作摇杆时，其内部的电阻值也相应的发生变化，将模块的 X_ADC 与 Y_ADC 接口连接到 M5Core 上支持的 ADC 转换的引脚上，通过程序读取摇杆偏移的数字量信息。

将 BUTTON 当按下摇杆时，其内部的按键开关也相应的做出动作，将输出信号变为低电平。



硬件连接

将 Joystick 模块的 X_ADC 和 Y_ADC 分别连接到 M5Core 上支持 ADC 功能的引脚上(在该案例程序中，将使用 PIN 35/36)，将 BUTTON 连接至 PIN 2.



案例

```
#include <M5Stack.h>

/*
  note:Reading the adc value requires writing the pin 25 of the
  adc to 0.
  dacWrite(25, 0);
*/

void setup() {
  // put your setup code here, to run once:
  M5.begin();
  pinMode(2, INPUT_PULLUP);
  dacWrite(25, 0);

  M5.Lcd.setCursor(100, 0, 4);
  M5.Lcd.print("JOYSTICK");
}

uint16_t x_data;
uint16_t y_data;
uint8_t button_data;
void loop() {
  // put your main code here, to run repeatedly:
  x_data = analogRead(35);
  y_data = analogRead(36);
  button_data = digitalRead(2);

  Serial.printf("x:%0d y:%0d button:%d\n", x_data, y_data, button_data);

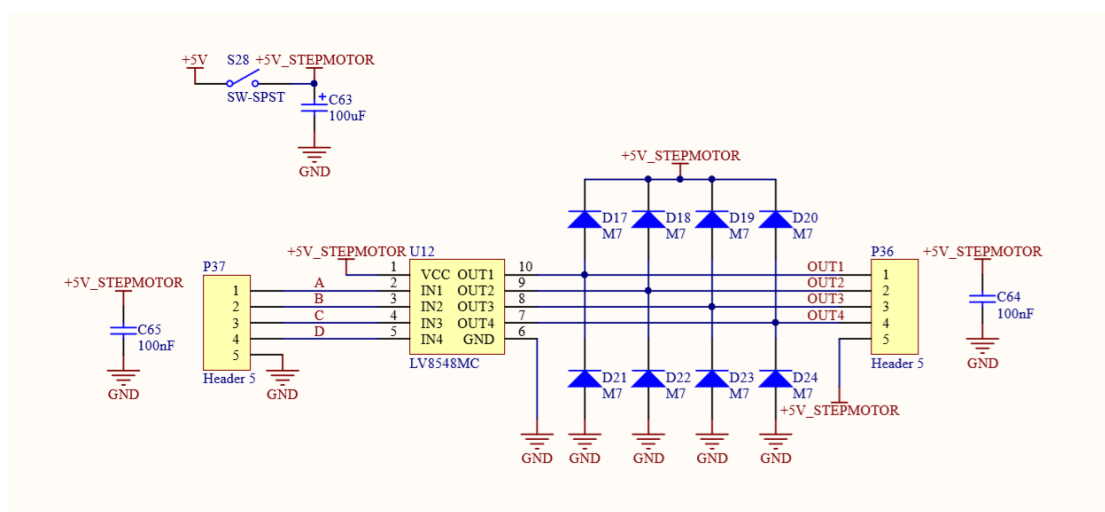
  M5.Lcd.setCursor(30, 120, 4);
  M5.Lcd.printf("x:%04d y:%04d button:%d\n", x_data, y_data, button_data);

  delay(200);
}
```


Step Motor 模块

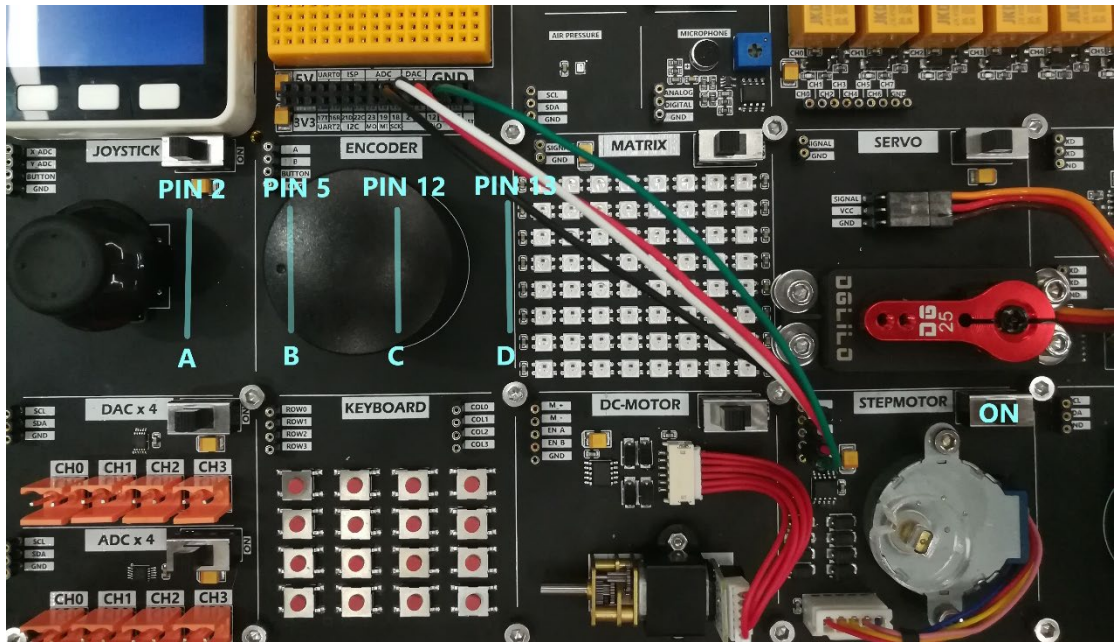
描述

Step Motor 是一个步进电机模块，内部集成电机驱动芯片 LV8548MC，使用五线四相制的连接方式。板上的 A, B, C, D 接口通过驱动芯片能够控制步进电机内四个磁极线圈的通断电，进而控制电机转动。



硬件连接

将控制引脚 PIN 2, 5, 12, 13 分别连接至步进电机模块的 A, B, C, D 接口，并打开独立电源，



不同的线圈通电顺序，会影响步进电机的旋转速度，完整旋转一周的拍数，以及步进值，运行时的稳定性。接下来的三个案例，将举例三种细分情况下的步进电机控制。

案例一

线圈	Step1	Step2	Step3	Step4
MotorA	高电平			
MotorB		高电平		
MotorC			高电平	
MotorD				高电平

```
#include <M5Stack.h>

const int MOTOR_A_A = 2;
const int MOTOR_A_B = 5;
const int MOTOR_A_C = 12;
const int MOTOR_A_D = 13;
void setup() {
    M5.begin();
    pinMode(MOTOR_A_A, OUTPUT);
    pinMode(MOTOR_A_B, OUTPUT);
    pinMode(MOTOR_A_C, OUTPUT);
    pinMode(MOTOR_A_D, OUTPUT);
}
void loop() {
```

```

digitalWrite(MOTOR_A_A, HIGH);
digitalWrite(MOTOR_A_B, HIGH);
digitalWrite(MOTOR_A_C, LOW);
digitalWrite(MOTOR_A_D, LOW);
delay(2);
digitalWrite(MOTOR_A_A, LOW);
digitalWrite(MOTOR_A_B, HIGH);
digitalWrite(MOTOR_A_C, HIGH);
digitalWrite(MOTOR_A_D, LOW);
delay(2);
digitalWrite(MOTOR_A_A, LOW);
digitalWrite(MOTOR_A_B, LOW);
digitalWrite(MOTOR_A_C, HIGH);
digitalWrite(MOTOR_A_D, HIGH);
delay(2);
digitalWrite(MOTOR_A_A, HIGH);
digitalWrite(MOTOR_A_B, LOW);
digitalWrite(MOTOR_A_C, LOW);
digitalWrite(MOTOR_A_D, HIGH);
delay(2);
}

```

案例二

线圈	Step1	Step2	Step3	Step4
MotorA	高电平	高电平		
MotorB		高电平	高电平	
MotorC			高电平	高电平
MotorD	高电平			高电平

```

#include <M5Stack.h>

const int MotorA = 2;
const int MotorB = 5;
const int MotorC = 12;
const int MotorD = 13;

const int pinMotor[4] = {MotorA, MotorB, MotorC, MotorD};

const int logic[4][4] = {

```

```

    {1,1,0,0},
    {0,1,1,0},
    {0,0,1,1},
    {1,0,0,1}
};

int count;

void setup() {
    M5.begin();
    pinMode(pinMotor[0], OUTPUT);
    pinMode(pinMotor[1], OUTPUT);
    pinMode(pinMotor[2], OUTPUT);
    pinMode(pinMotor[3], OUTPUT);
}

void loop() {
    for (int i = 0; i < 2048 ; i++) {
        driveMotor();
        delay(2);
    }
}

void driveMotor() {
    count++;
    int step = count % 4;
    digitalWrite(pinMotor[0] , logic[step][0]);
    digitalWrite(pinMotor[1] , logic[step][1]);
    digitalWrite(pinMotor[2] , logic[step][2]);
    digitalWrite(pinMotor[3] , logic[step][3]);
}

```

案例三

线圈	Step1	Step2	Step3	Step4	Step5	Step6	Step7	Step8
MotorA	高电平						高电平	高电平
MotorB	高电平	高电平	高电平					
MotorC			高电平	高电平	高电平			
MotorD					高电平	高电平	高电平	

我们还可以进一步的将 step 细分, 只需要修改通电逻辑的数组, 以及驱动函数。

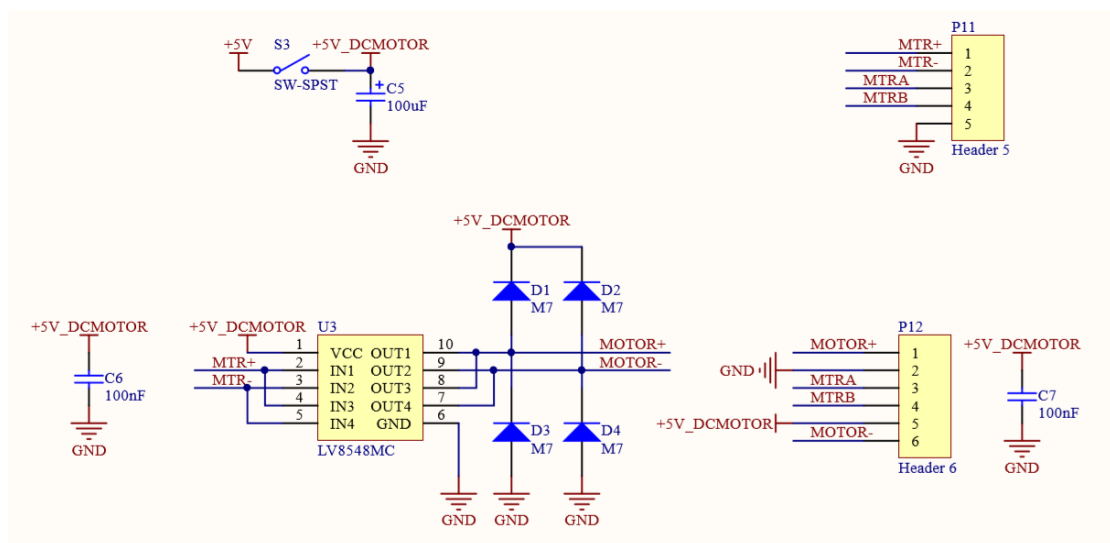
```
const int logic[8][4] = {  
    {1,1,0,0},  
    {0,1,0,0},  
    {0,1,1,0},  
    {0,0,1,0},  
    {0,0,1,1},  
    {0,0,0,1},  
    {1,0,0,1},  
    {1,0,0,0}  
};
```

```
void driveMotor() {  
    count++;  
    int step = count % 8;  
    digitalWrite(pinMotor[0] , logic[step][0]);  
    digitalWrite(pinMotor[1] , logic[step][1]);  
    digitalWrite(pinMotor[2] , logic[step][2]);  
    digitalWrite(pinMotor[3] , logic[step][3]);  
}
```

DC-Motor 模块

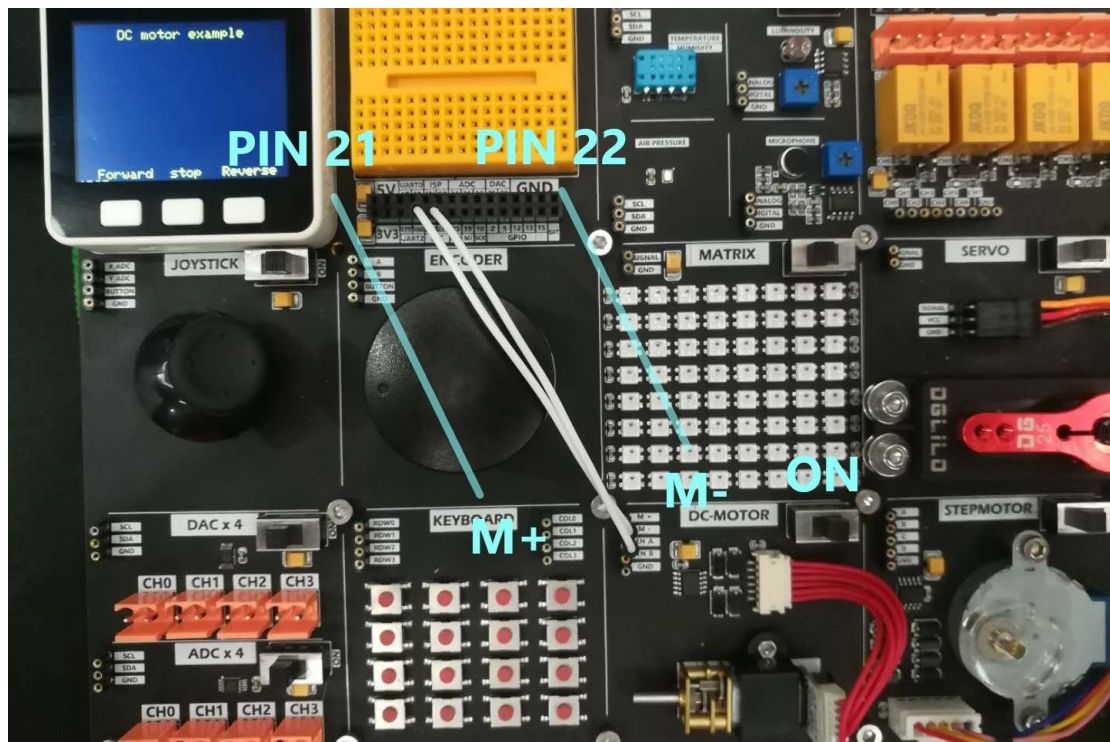
描述

DC-Motor 是一个带反馈的直流电机模块，该模块内置 LV8548MC 电机驱动芯片，提供了 M+，M-两个接口用于控制正反转，当 M+为高电平，M-为低电平时，电机为正转。当 M+为低电平，M-为高电平时，电机为反转。当 M+，M-同为低电平为低电平时，电机制动。



硬件连接

将控制引脚 PIN 21，22 分别连接至直流电机模块的 M+，M-接口，并打开独立电源.



案例

```
#include <M5Stack.h>
// The setup() function runs once each time the micro-
controller starts
void setup() {
    // init lcd, serial, but don't init sd card
    M5.begin(true, false, true);
    M5.Lcd.clear(BLACK);
    M5.Lcd.setTextColor(YELLOW);
    M5.Lcd.setTextSize(2);
    M5.Lcd.setCursor(65, 10);
    M5.Lcd.println("DC motor example");
    M5.Lcd.setCursor(30, 220);
    M5.Lcd.println("Forward");
    M5.Lcd.setCursor(140, 220);
    M5.Lcd.println("stop");
    M5.Lcd.setCursor(220, 220);
    M5.Lcd.println("Reverse");

    pinMode(21, OUTPUT);
    pinMode(22, OUTPUT);
}
```

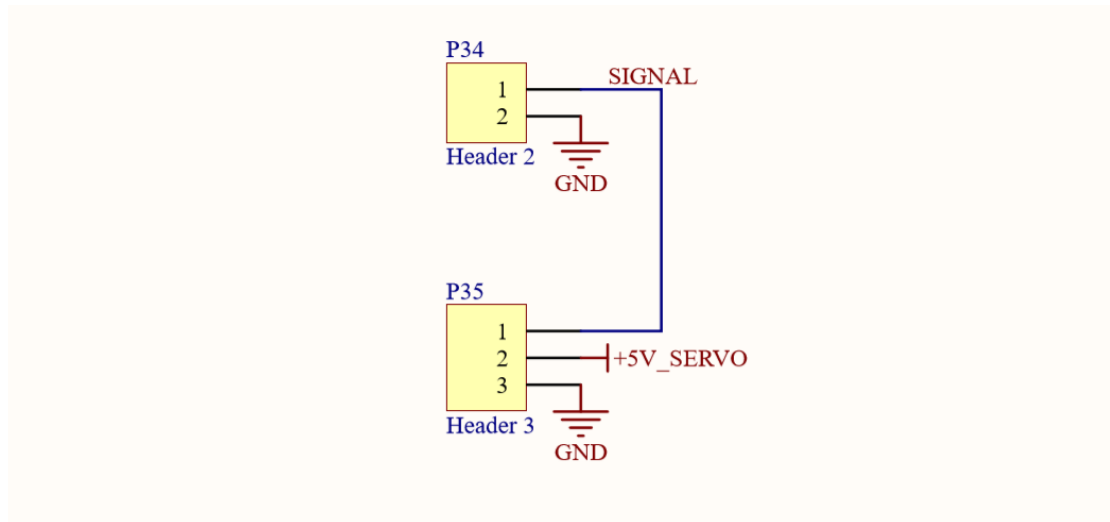
```
// Add the main program code into the continuous loop() function
void loop() {
    M5.update();

    if (M5.BtnA.wasReleased()) {
        M5.Lcd.print('A');
        digitalWrite(22, LOW);
        digitalWrite(21, HIGH);
    } else if (M5.BtnB.wasReleased()) {
        M5.Lcd.print('B');
        digitalWrite(21, LOW);
        digitalWrite(22, LOW);
    } else if (M5.BtnC.wasReleased()) {
        M5.Lcd.print('C');
        digitalWrite(21, LOW);
        digitalWrite(22, HIGH);
    }
}
```


Servo 模块

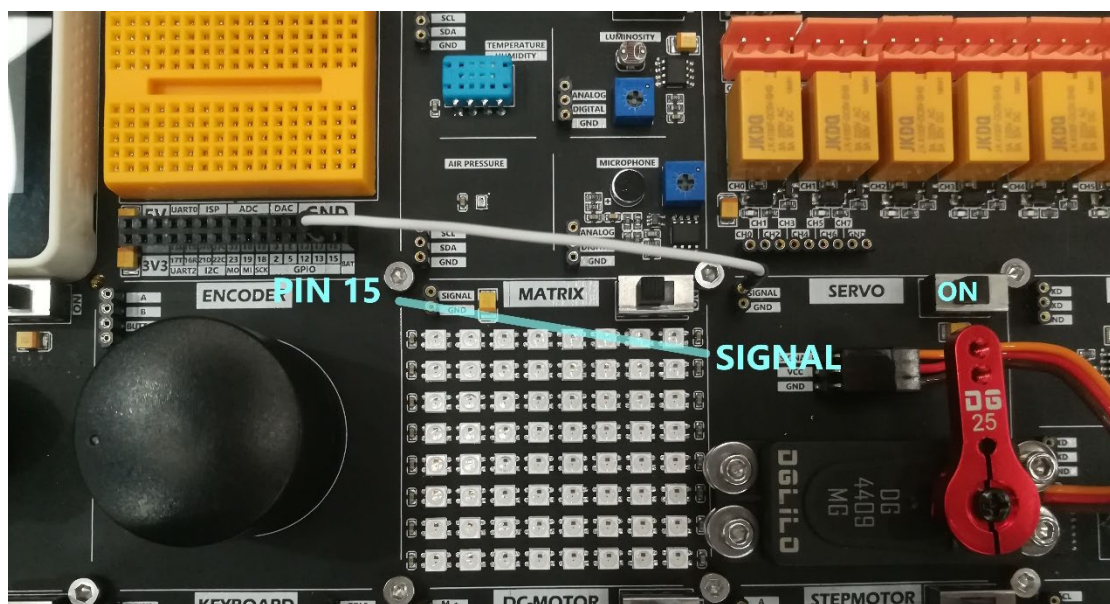
描述

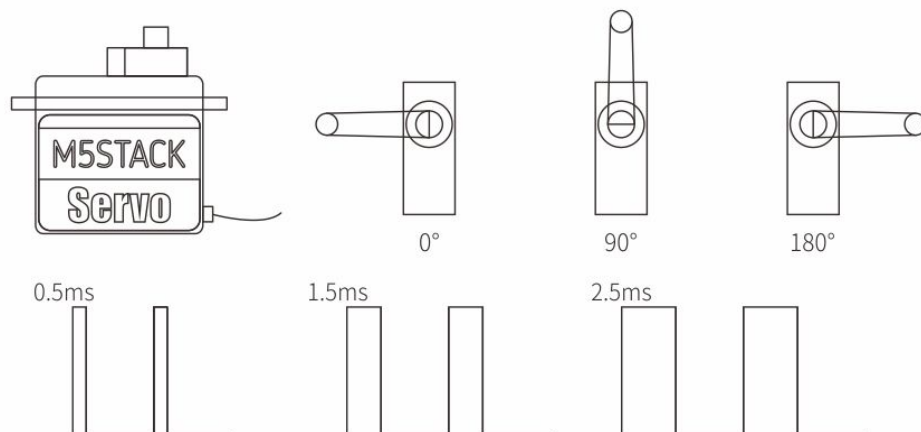
开发板上配备了一个扭力高达 10KG 的舵机（ $0^{\circ}\sim 180^{\circ}$ ），默认 5V 电源供电，使用单信号总线进行控制。驱动信号为 50Hz，根据 PWM 占空比控制旋转至不同角度。PWM 波形中高电平时间为（ $0.5\mu s\sim 2.5\mu s$ ）分别控制舵机旋转（ $0^{\circ}\sim 180^{\circ}$ ）



硬件连接

将控制引脚 PIN 15 连接至舵机的 SIGNAL 接口，并打开独立电源，





案例

```
#include <M5Stack.h>
#include "driver/ledc.h"

const int servo_pin = 15;
int freq = 50;
int ledChannel = 0;
int resolution = 8;

void setup() {

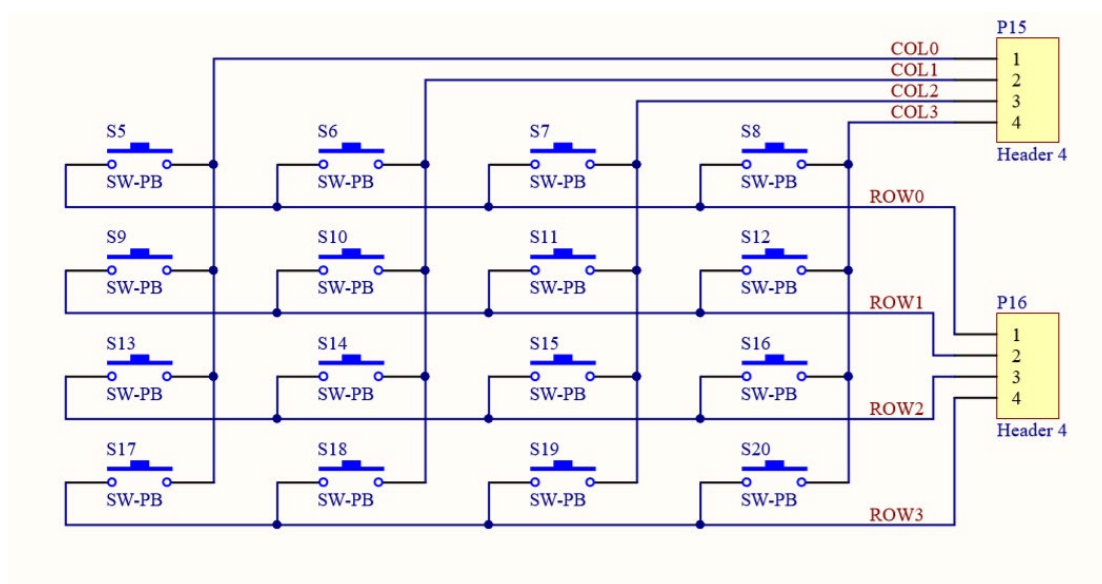
    M5.begin();
    M5.Lcd.setCursor(120, 110, 4);
    M5.Lcd.println("SERVO");
    ledcSetup(ledChannel, freq, resolution);
    ledcAttachPin(servo_pin, ledChannel);
}

void loop() {
    ledcWrite(ledChannel, 6); // 0°
    delay(1000);
    ledcWrite(ledChannel, 18); // 90°
    delay(1000);
    ledcWrite(ledChannel, 30); // 180°
    delay(1000);
}
```

Keyboard 矩阵按键

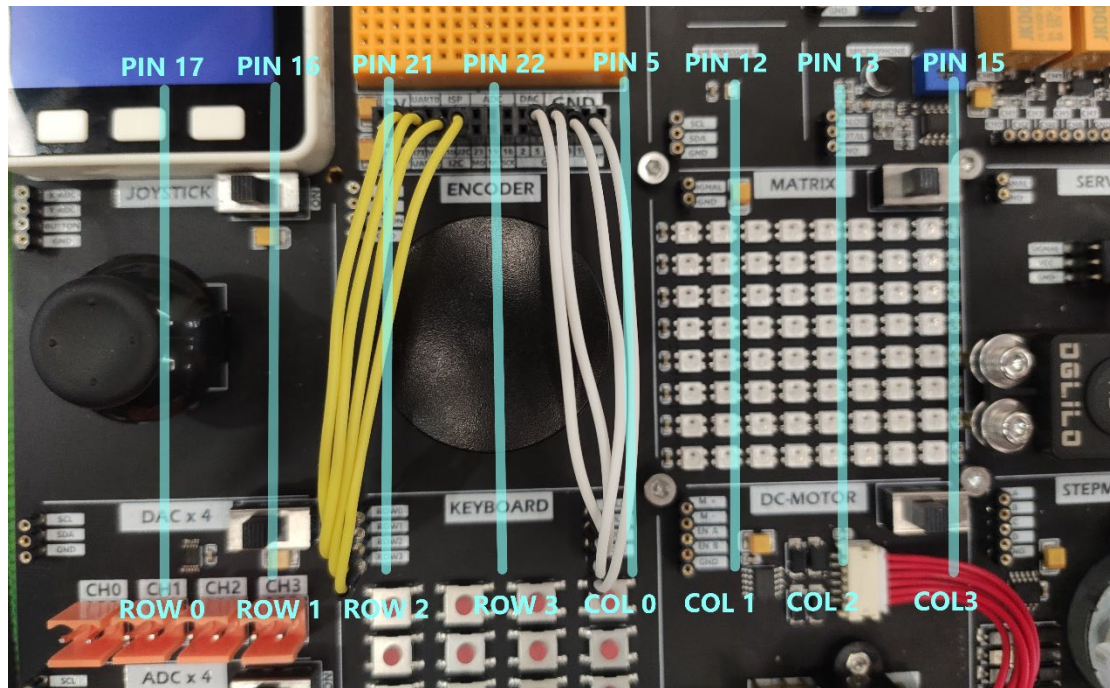
描述

开发板上提供了一个 4 x 4 的矩阵按键模块，通过编程扫描能够检测出具体被按下的按键，能够提供你更加丰富的键值输入与多样化的控制功能。



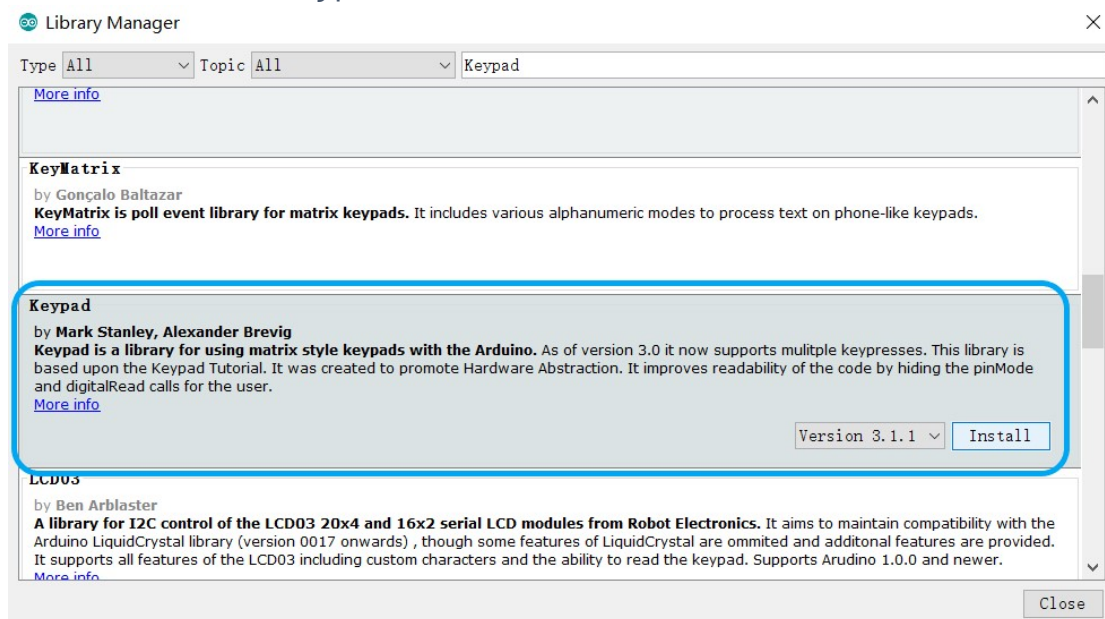
硬件连接

将控制引脚分别连接至按键矩阵行和列的各个引脚，使程序能够扫描按键的动作。



案例

该案例中使用了库<Keypad.h>，你可以在 Arduino 的库管理中搜索并安装。



```
#include<M5Stack.h>
#include <Keypad.h>
/**
 * note:You should first install the keyboard library.
 * https://github.com/Chris--A/Keypad
 */
const byte ROWS = 4; //four rows
```

```

const byte COLS = 4; //three columns
byte rowPins[ROWS] = {17,16,21,22};
byte colPins[COLS] = {5, 26, 13, 15};

char keys[ROWS][COLS] = {
  {'a','b','c','d'},
  {'e','f','g','h'},
  {'i','j','k','l'},
  {'m','n','o','p'}
};

Keypad keypad = Keypad(makeKeymap(keys), rowPins, colPins, ROWS, COLS);

void setup() {
  // put your setup code here, to run once:
  M5.begin();

  M5.Lcd.setCursor(100, 0, 4);
  M5.Lcd.println("KEYBOARD");
}

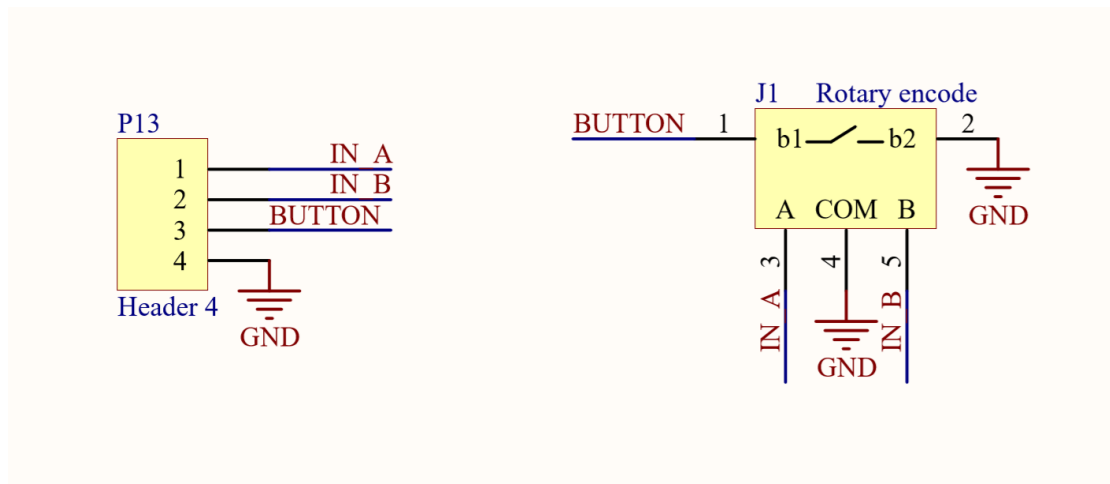
void loop() {
  //put your main code here, to run repeatedly:
  char key = keypad.getKey();
  if(key){
    Serial.println(key);
    M5.Lcd.fillRect(150, 150, 80, 50, BLACK);
    M5.Lcd.setCursor(150, 140, 4);
    M5.Lcd.printf("%c", 'A');
  }
}

```

Encoder 旋转编码器

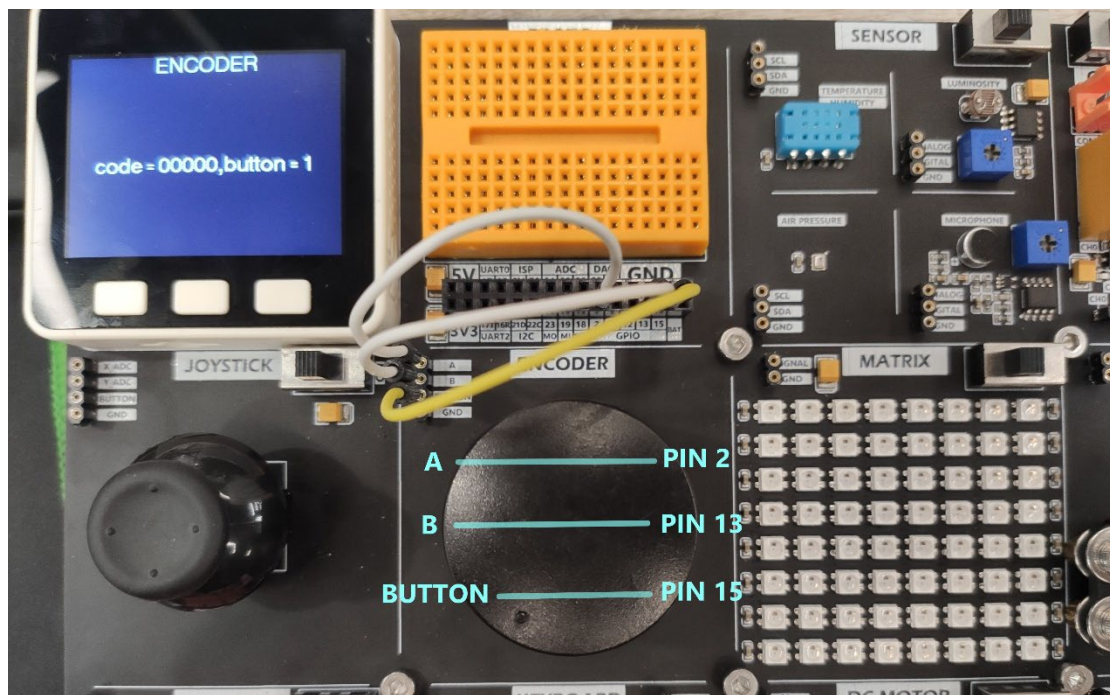
描述

当旋转旋钮时，端口 A 与端口 B. 后根据选择旋转方向产生相应的高电平. 按下旋钮时，其内部的按键开关也相应的做出动作，将输出信号变为低电平.



硬件连接

将控制引脚连接至普通的 GPIO，在程序中读取引脚的电平变化即可判断出旋钮的动作。



案例

```
#include <M5Stack.h>

const int phaseA = 2;
const int phaseB = 13;
const int Button = 15;
```

```
#define GET_CODE() uint8_t(digitalRead(phaseA) << 4 | digitalRead(phaseB))
```

```
int32_t count = 65536;  
int32_t count_last = 65536;  
int32_t count_change = 0;  
uint8_t code = 0;  
uint8_t code_old = 0;
```

```
void encoderEvent() {  
    code = GET_CODE();  
    if(code != code_old) {  
        if(code == 0x00) {  
            count_last = count;  
            if(code_old == 0x10) {  
                count--;  
                count_change == -65536 ? count_change : count_change--;  
            } else {  
                count_change == 65536 ? count_change : count_change++;  
            }  
        }  
        code_old = code;  
    }  
}
```

```
void setup() {  
    // put your setup code here, to run once:  
    M5.begin();  
    pinMode(phaseA, INPUT_PULLUP);  
    pinMode(phaseB, INPUT_PULLUP);  
    pinMode(Button, INPUT_PULLUP);  
    digitalWrite(25, 0);  
  
    M5.Lcd.setCursor(100, 0, 4);  
    M5.Lcd.print("ENCODER");  
  
    code = GET_CODE();  
    code_old = code;  
}
```

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

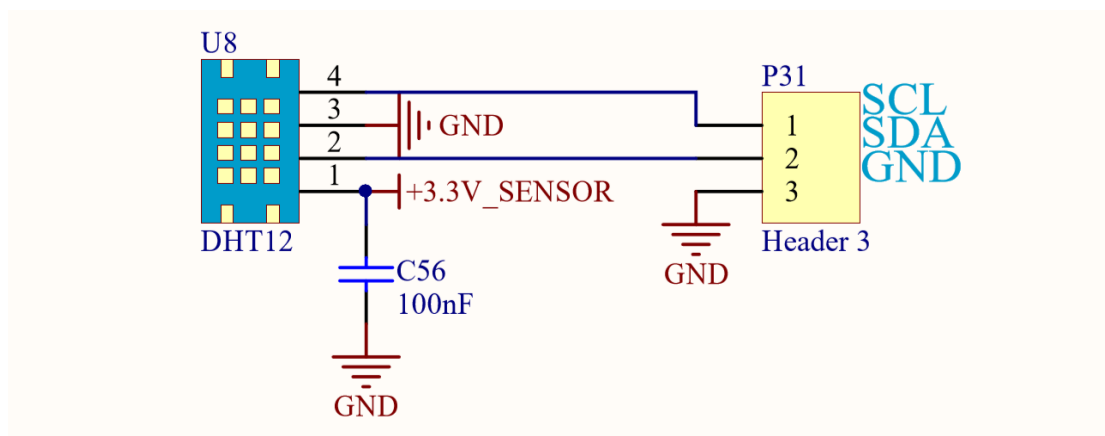
```
uint8_t value = digitalRead(Button);
encoderEvent();
Serial.printf("code = %d,button = %d\r\n ",count_change,value);

M5.Lcd.setCursor(30, 120, 4);
M5.Lcd.printf("code = %05d,button = %d\r\n ",count_change,value);
delay(1);
}
```


DHT12 温湿度检测

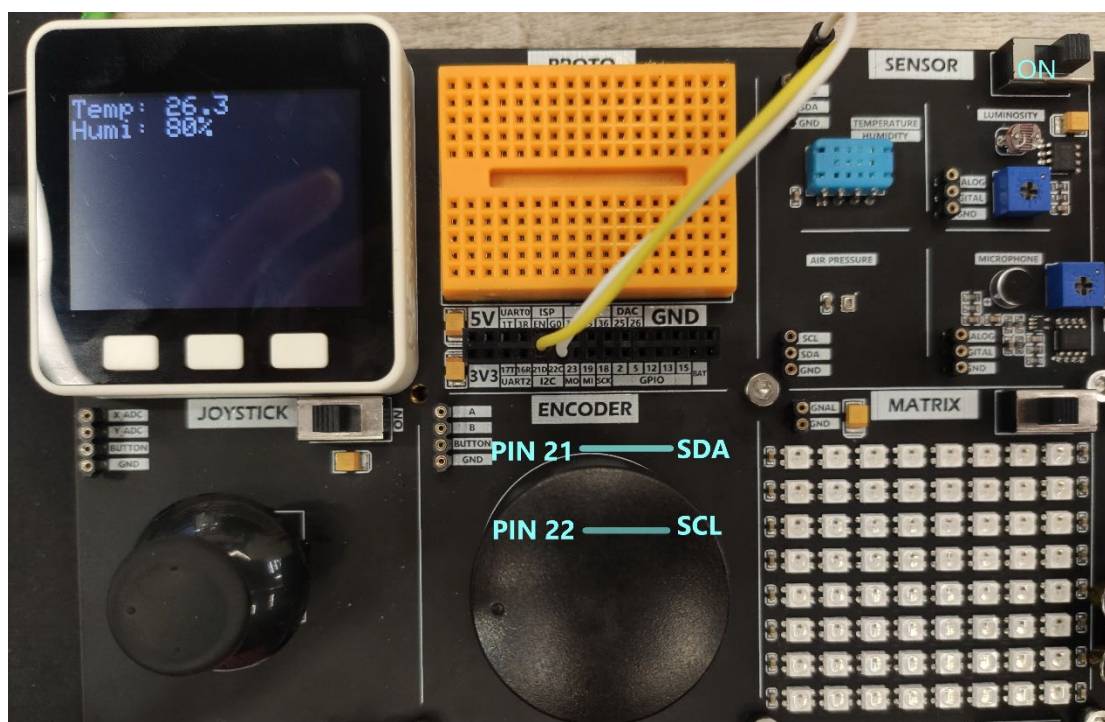
描述

位于传感器模块组里的 DHT12 温湿度传感器能够采集环境中的温度、湿度数据，并通过 I2C 协议进行数据传输。I2C 地址为 (0x5c)



硬件连接

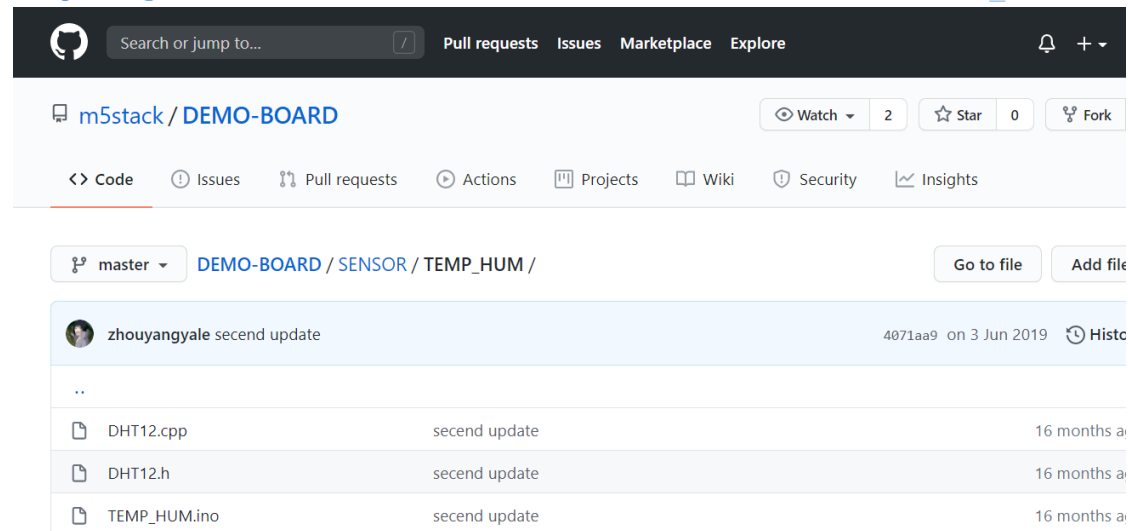
使用 I2C 协议进行通信的传感器设备，在使用时可以将其连接到 M5Core 的默认 I2C 协议引脚 PIN21 (SDA)、PIN22 (SCL)，



案例

以下代码仅为主程序，并未包含其依赖库文件如<DHT12.h>，完整代码请访问下方 Github 地址获取。，

https://github.com/m5stack/DEMO-BOARD/tree/master/SENSOR/TEMP_HUM



读取 DHT12 传感器所测量的温度，湿度值，并显示到屏幕上。

```
#include <M5Stack.h>
#include "DHT12.h"
#include <Wire.h> //The DHT12 uses I2C comunication.

DHT12 dht12; //Preset scale CELSIUS and ID 0x5c.

void setup() {
    M5.begin();
    Wire.begin();
    M5.Lcd.setCursor(80, 0, 4);
    M5.Lcd.print("TEMPERATURE");
}

void loop() {

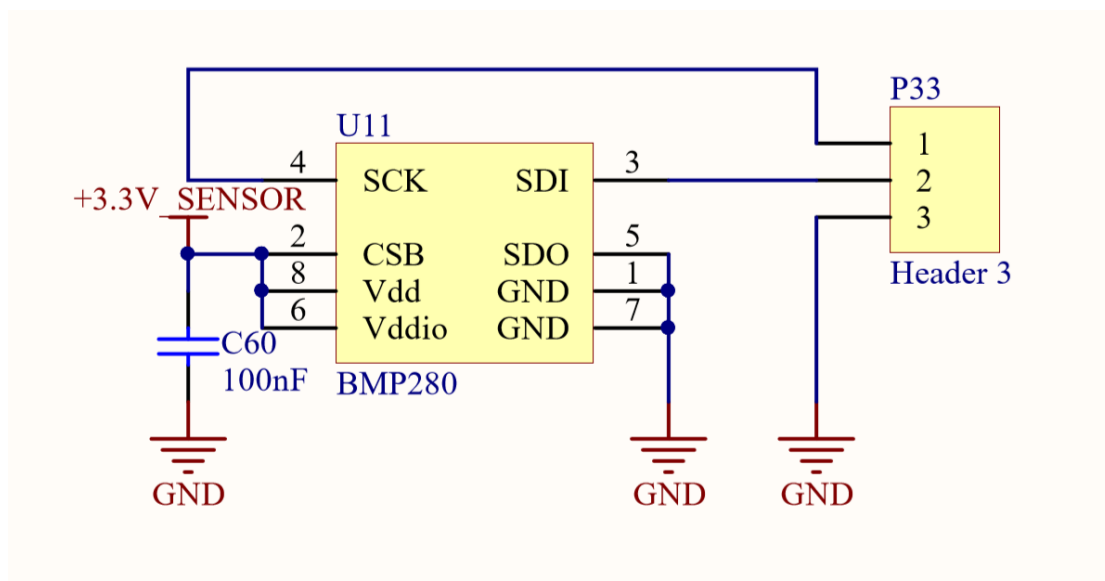
    float tmp = dht12.readTemperature();
    float hum = dht12.readHumidity();
    M5.Lcd.setCursor(30, 100, 4);
    M5.Lcd.printf("Temp: %2.1f  Humi: %2.0f%%", tmp, hum);

    delay(100);
}
```


BMP280 气压检测

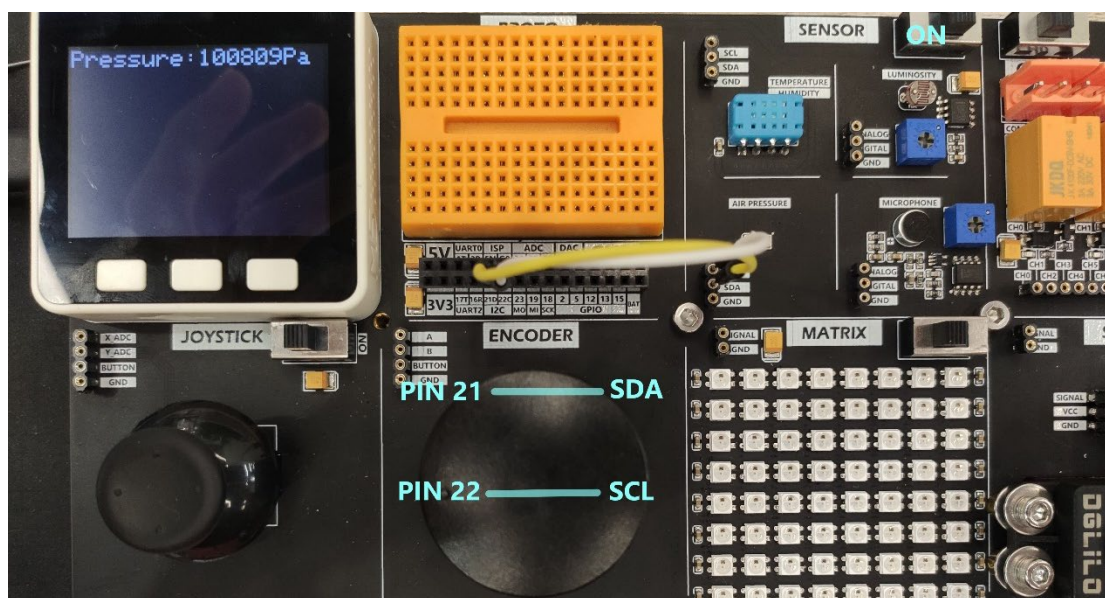
描述

位于传感器模块组里的 BMP280 气压传感器能够采集当前位置的气压值数据，并通过 I2C 协议进行数据传输。I2C 地址为 (0x76)



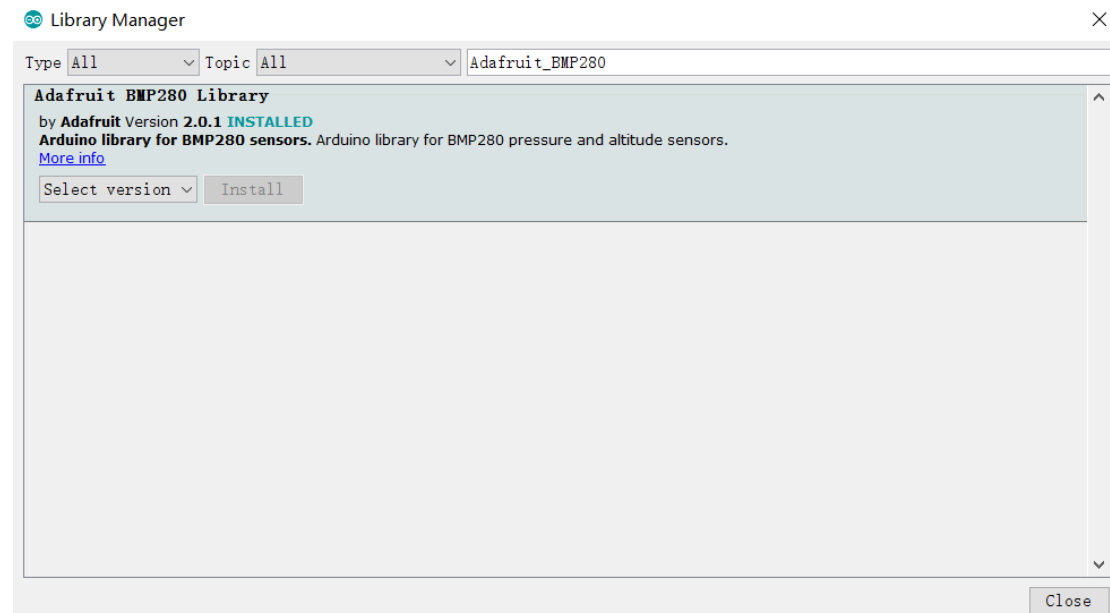
硬件连接

使用 I2C 协议进行通信的传感器设备，在使用时可以将其连接到 M5Core 的默认 I2C 协议引脚 PIN21 (SDA)、PIN22 (SCL)，



案例

该案例中使用了库<Adafruit_BMP280.h>你可以在 Arduino 的库管理中搜索并安装。



读取 BMP280 传感器所测量的大气压值并显示到屏幕上。

```
#include <M5Stack.h>
#include <Wire.h>
#include "Adafruit_Sensor.h"
#include <Adafruit_BMP280.h>

/*
   note: need add Library Adafruit_BMP280 from Library manage
*/

Adafruit_BMP280 bme;
void setup() {
  M5.begin();
  Wire.begin();
  M5.Lcd.setCursor(70, 0, 4);
  M5.Lcd.print("AIR_PRESSURE");

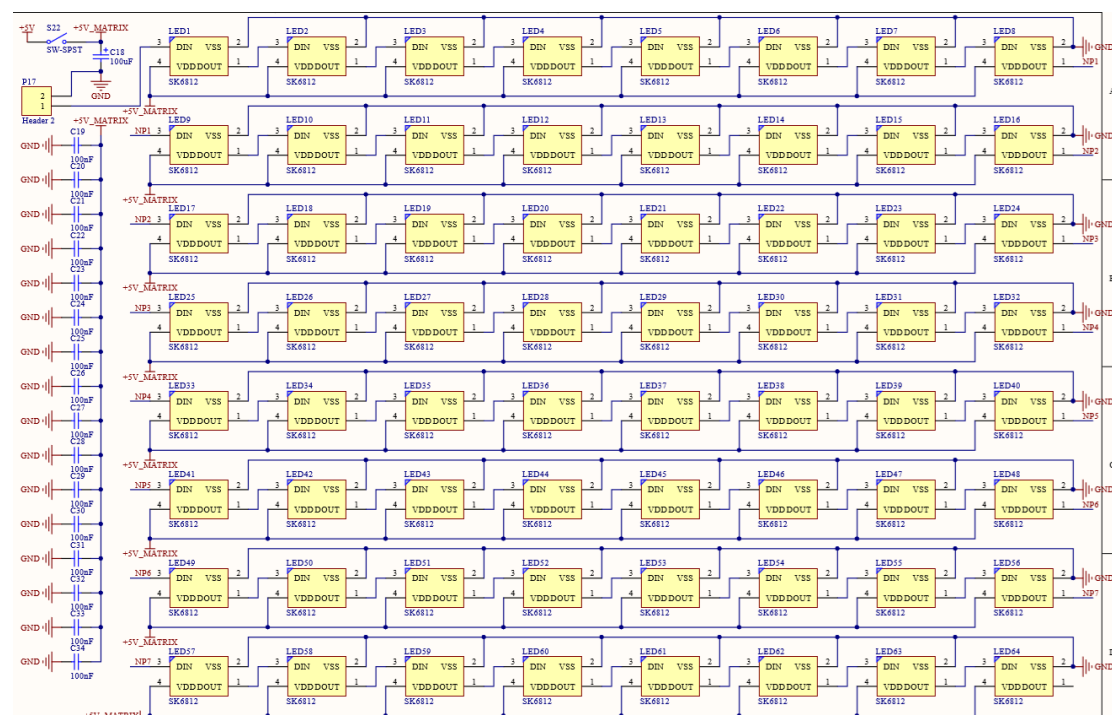
  if (!bme.begin(0x76)){
    Serial.println("Could not find a valid BMP280 sensor, check wiring!");
    while (1);
  }
}
```

```
void loop() {  
    float pressure = bme.readPressure();  
    M5.Lcd.setCursor(50, 100, 4);  
    M5.Lcd.printf("Pressure:%2.0fPa\r\n",pressure);  
    delay(100);  
}
```

MATRIX /LED 矩阵模块

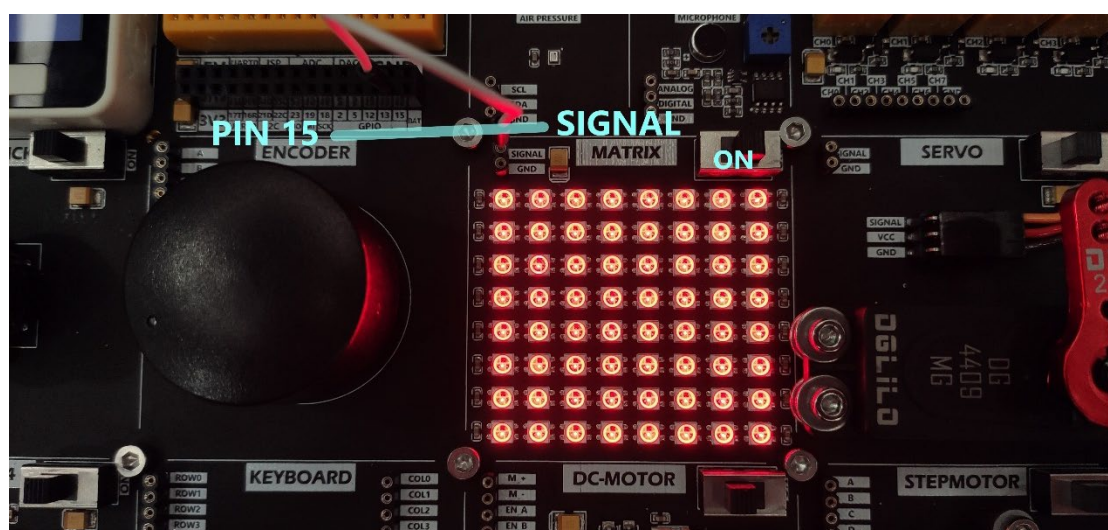
描述

MATRIX 是一块 8x8 的 LED 矩阵，通过编程能够控制矩阵上的任意一颗 LED 灯发光，以及调整颜色，亮度等属性。



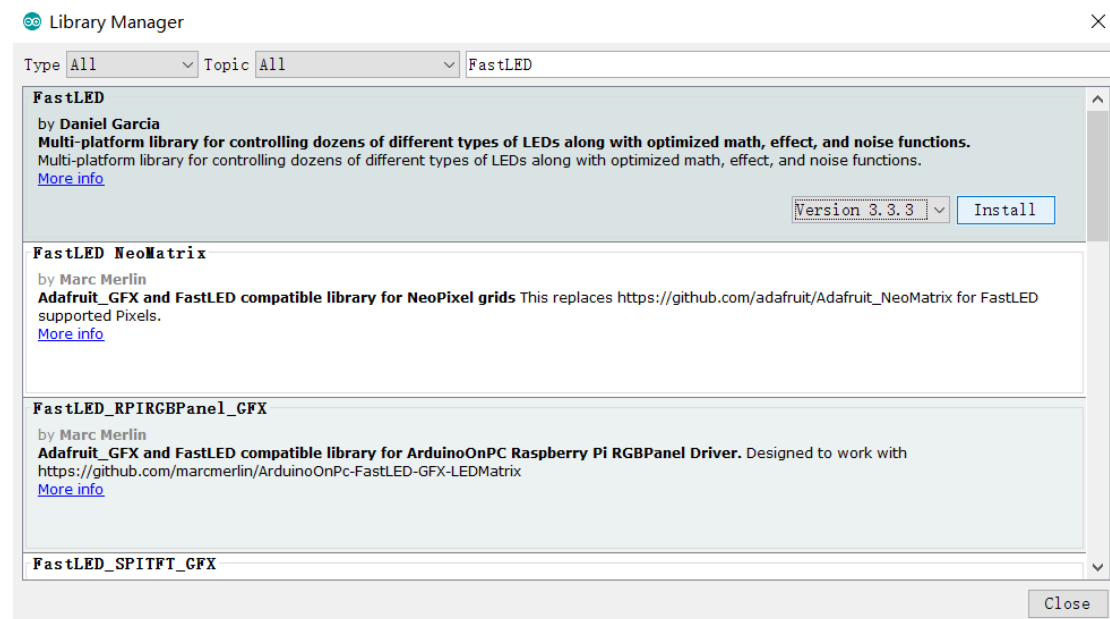
硬件连接

将控制引脚连接至矩阵的 SIGNAL 接口，并打开独立电源开关。



案例

该案例中使用了库<FastLED.h>，你可以在 Arduino 的库管理中搜索并安装。



驱动 LED 矩阵执行流水灯效果

```
#include <M5Stack.h>
#include <FastLED.h>
/****
 *   note:You should first install the Fastled Library.
 *
 ****/
#define DATA_PIN    15
#define LED_TYPE     WS2811
#define COLOR_ORDER  GRB
#define NUM_LEDS     64
CRGB leds[NUM_LEDS];
#define BRIGHTNESS   5
void setup() {
    // put your setup code here, to run once:
    M5.begin();
    M5.Lcd.setCursor(120, 110, 4);
    M5.Lcd.println("MATRIX");
}
```



```
FastLED.addLeds<LED_TYPE,DATA_PIN,COLOR_ORDER>(leds, NUM_LED
S).setCorrection(TypicalLEDStrip);
FastLED.setBrightness(BRIGHTNESS);

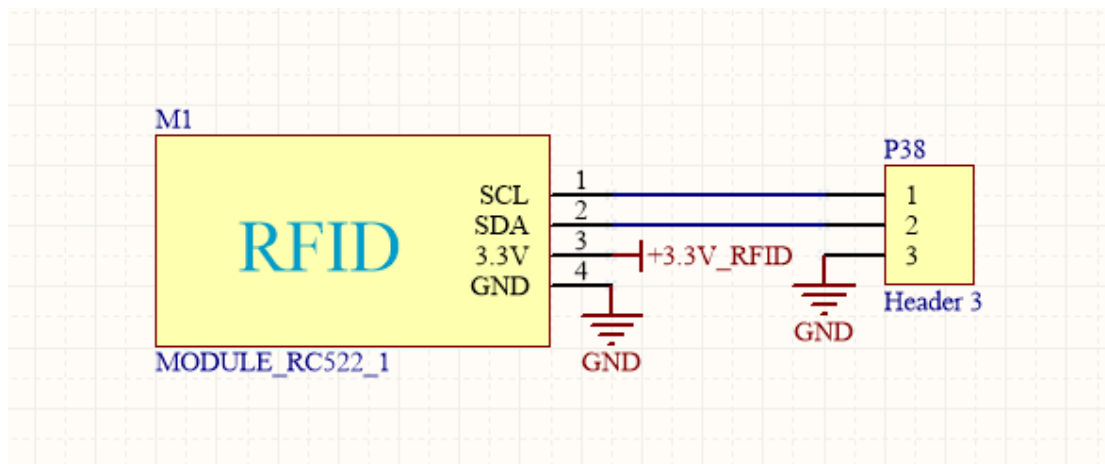
}
void loop() {
  for(int i = 0; i < 64; i++){
    leds[i] = CRGB::White;
    FastLED.show();
  }
  delay(500);
  // Now turn the LED off, then pause
  for(int i = 0; i < 64; i++){
    leds[i] = CRGB::Black;
    FastLED.show();
  }
  delay(500);
}
```

RFID 无线射频识别

描述

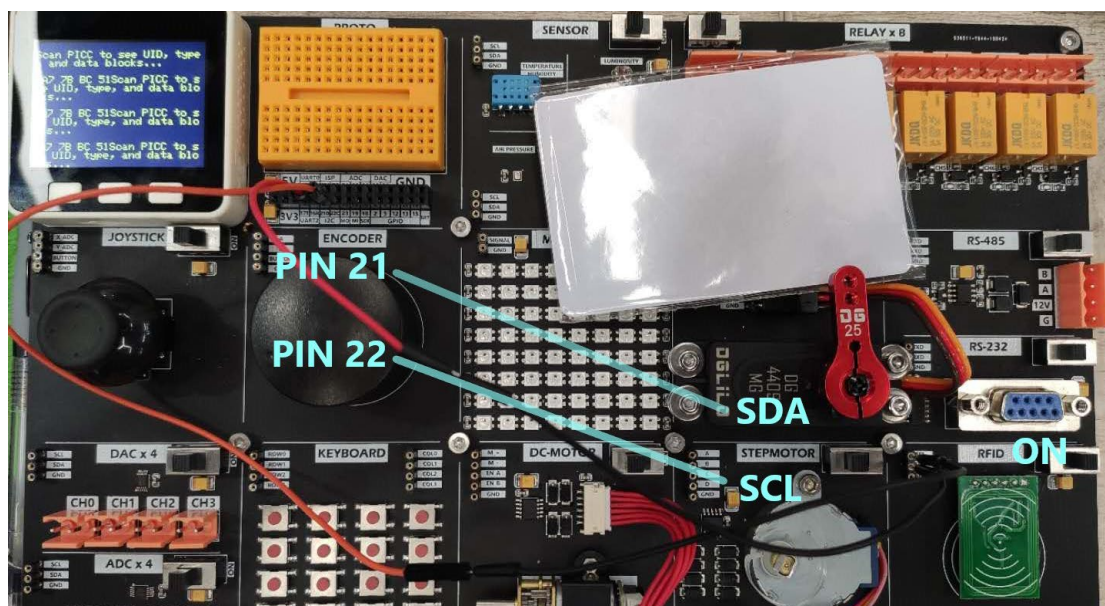
RFID 模块上集成了 RC522 射频识别芯片，并通过 I2C 协议进行数据传输。I2C 地址为（0x28）

工作频率为 13.56MHz. 支持读卡、写卡、识别、记录、对 RF 卡进行编码和授权等多个功能。



硬件连接

使用 I2C 协议进行通信的传感器设备，在使用时可以将其连接到 M5Core 的默认 I2C 协议引脚 PIN21 (SDA)、PIN22 (SCL)，



案例

以下代码仅为主程序，并未包含其依赖库文件如<MFRC522_I2C.h>，完整代码请访问下方 Github 地址获取。 ，

<https://github.com/m5stack/DEMO-BOARD/tree/master/RFID>

Search or jump to...

Pull requests Issues Marketplace Explore

m5stack / DEMO-BOARD

Watch 2 Star 0 Fork

Code Issues Pull requests Actions Projects Wiki Security Insights

master DEMO-BOARD / RFID /

Go to file Add file

zhouyangyale secd update 4071aa9 on 3 Jun 2019

File	Update	Time
MFRC522_I2C.cpp	update	16 months a
MFRC522_I2C.h	update	16 months a
RFID.ino	secd update	16 months a

读取 IC 卡的 ID 以及软件版本

```
#include <Wire.h>
#include "MFRC522_I2C.h"
#include <M5Stack.h>

// 0x28 is i2c address on SDA. Check your address with i2c scanner if not match.
MFRC522 mfrc522(0x28); // Create MFRC522 instance.

void setup() {

  M5.begin();
  Wire.begin();
  M5.Lcd.setCursor(140, 0, 4);
  M5.Lcd.println("RFID");
  mfrc522.PCD_Init(); // Init MFRC522
  ShowReaderDetails(); // Show details of PCD - MFR
C522 Card Reader details
  Serial.println(F("Scan PICC to see UID, type, and data block
s..."));
  M5.Lcd.setCursor(0,30,2);
  M5.Lcd.println("Scan PICC to see UID, type, and data blocks.
..");
}

void loop() {
  // Look for new cards, and select one if present
  if (!mfrc522.PICC_IsNewCardPresent() || ! mfrc522.PICC_ReadC
ardSerial()) {
```

```

    delay(50);
    return;
}

// Now a card is selected. The UID and SAK is in mfrc522.uid
.

// Dump UID
Serial.print(F("Card UID:"));
M5.Lcd.println(" ");
for (byte i = 0; i < mfrc522.uid.size; i++) {
    Serial.print(mfrc522.uid.uidByte[i] < 0x10 ? " 0" : " ");
    Serial.print(mfrc522.uid.uidByte[i], HEX);
    M5.Lcd.print(mfrc522.uid.uidByte[i] < 0x10 ? " 0" : " ");
    M5.Lcd.print(mfrc522.uid.uidByte[i], HEX);
}
Serial.println();
}

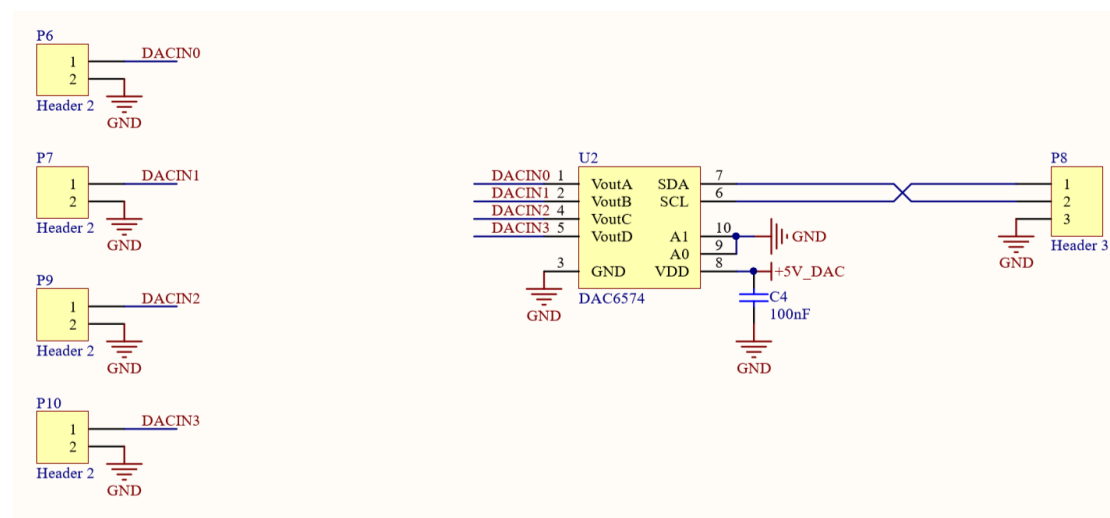
void ShowReaderDetails() {
    // Get the MFRC522 software version
    byte v = mfrc522.PCD_ReadRegister(mfrc522.VersionReg);
    Serial.print(F("MFRC522 Software Version: 0x"));
    Serial.print(v, HEX);
    if (v == 0x91)
        Serial.print(F(" = v1.0"));
    else if (v == 0x92)
        Serial.print(F(" = v2.0"));
    else
        Serial.print(F(" (unknown)"));
    Serial.println("");
    // When 0x00 or 0xFF is returned, communication probably failed
    if((v == 0x00) || (v == 0xFF)) {
        Serial.println(F("WARNING: Communication failure, is the MFRC522 properly connected?"));
    }
}
}

```

DAC 数字-模拟转换模块

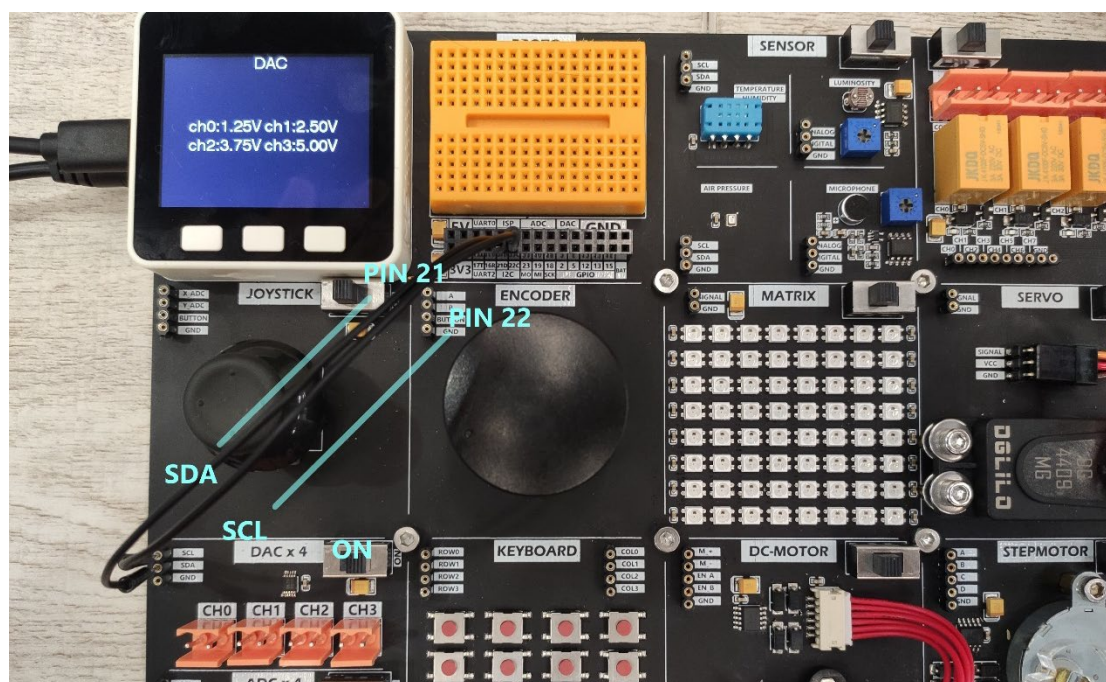
描述

开发板上提供了 4 个 DAC 转换接口，这意味着你可以编程控制通道输出不同电压。使用 I2C 协议进行控制，通信地址为 (0x28)



硬件连接

使用 I2C 协议进行通信的传感器设备，在使用时可以将其连接到 M5Core 默认的 I2C 协议引脚 PIN21 (SDA)、PIN22 (SCL)，



案例

驱动 4 个 DAC 通道分别输出电压 1.25V , 2.50V , 3.75V , 5.00V

```
#include <M5Stack.h>

#define DAC_ADDR 0x4C

void outVoltage(uint8_t ch,uint16_t v){

    Wire.beginTransmission(DAC_ADDR);
    Wire.write(0x10|(ch<<1));

    Wire.write((v >> 2) & 0xff);
    Wire.write((v << 6) & 0xff);
    Wire.endTransmission();
}

void setup() {
    // put your setup code here, to run once:
    M5.begin();
    Wire.begin(21, 22);
    dacWrite(25, 0);

    M5.Lcd.setCursor(140, 0, 4);
    M5.Lcd.print("DAC");

    outVoltage(0,256);    //1.25v
    outVoltage(1,512);    //2.50v
    outVoltage(2,768);    //3.75v
    outVoltage(3,1023);   //5.00v

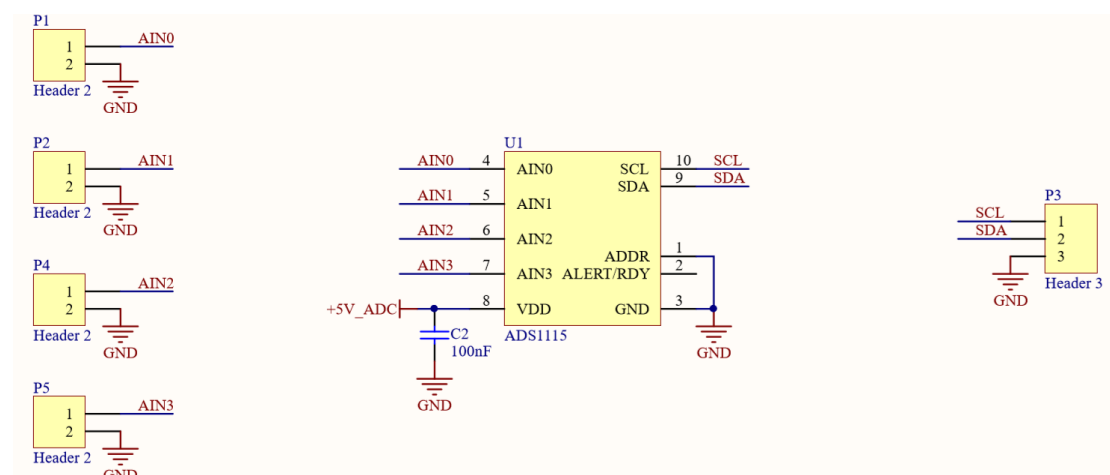
    M5.Lcd.setCursor(40, 100, 4);
    M5.Lcd.println("ch0:1.25V ch1:2.50V");
    M5.Lcd.setCursor(40, 130, 4);
    M5.Lcd.println("ch2:3.75V ch3:5.00V");
}

void loop() {
    delay(200);
}
```


ADC 模拟-数字转换模块

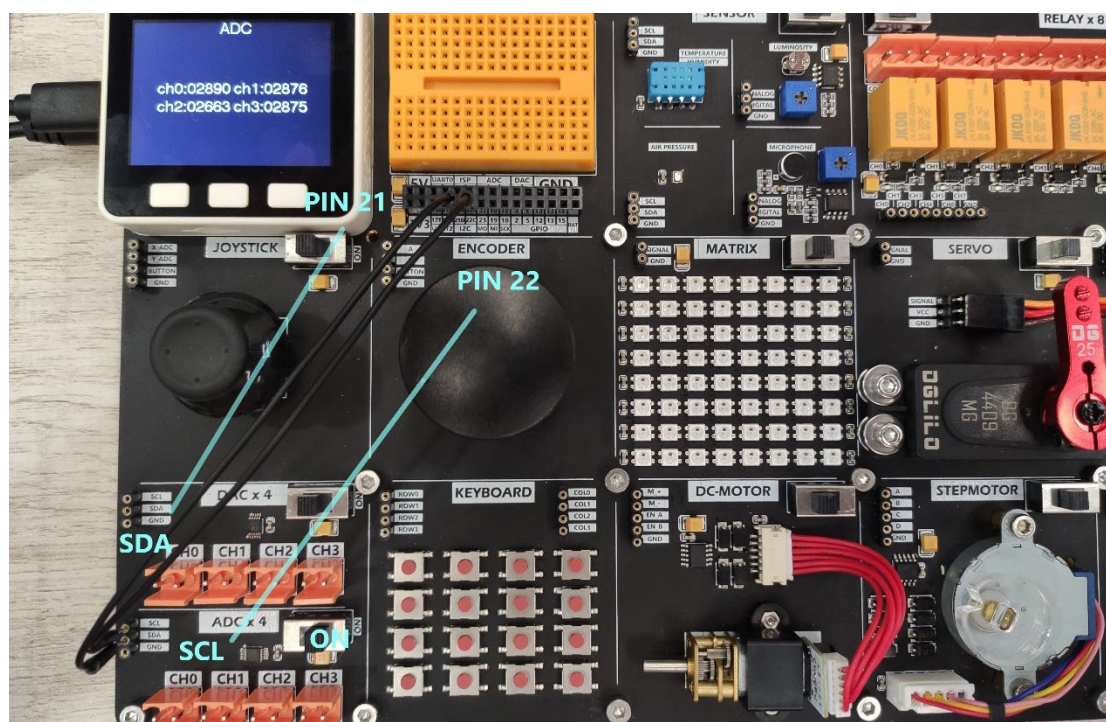
描述

开发板上提供了 4 个 ADC 转换接口，这意味着你可以输入一些模拟量信号，并转换成数字量进行分析计算。使用 I2C 协议进行控制，通信地址为（0x48）



硬件连接

使用 I2C 协议进行通信的传感器设备，在使用时可以将其连接到 M5Core 默认的 I2C 协议引脚 PIN21 (SDA)、PIN22 (SCL)，



案例

读取 4 个 ADC 转换接口输入的模拟量信号，并转换成数字量显示在屏幕上。

```
#include <M5Stack.h>
#define ADC_ADDR 0x48
uint16_t InVoltage(uint8_t ch){

    uint8_t data_L = 0;
    uint8_t data_H = 0;
    uint16_t data_adc = 0;

    Wire.beginTransmission(ADC_ADDR);
    Wire.write(0x01);
    Wire.write(0xC0 | (ch << 4));
    Wire.write(0x83);
    Wire.endTransmission();

    Wire.beginTransmission(ADC_ADDR);
    Wire.write(0x00);
    Wire.endTransmission();

    delay(50);

    Wire.requestFrom(ADC_ADDR, 2);
    while(Wire.available()){
        data_H = Wire.read();
        data_L = Wire.read();
    }

    data_adc = (data_H << 8) | data_L;
    return data_adc;
}

void setup() {
    M5.begin();
    Wire.begin();
    dacWrite(25, 0);
    M5.Lcd.setCursor(140, 0, 4);
    M5.Lcd.print("ADC");
}

uint16_t adc_ch0 = 0;
uint16_t adc_ch1 = 0;
uint16_t adc_ch2 = 0;
```



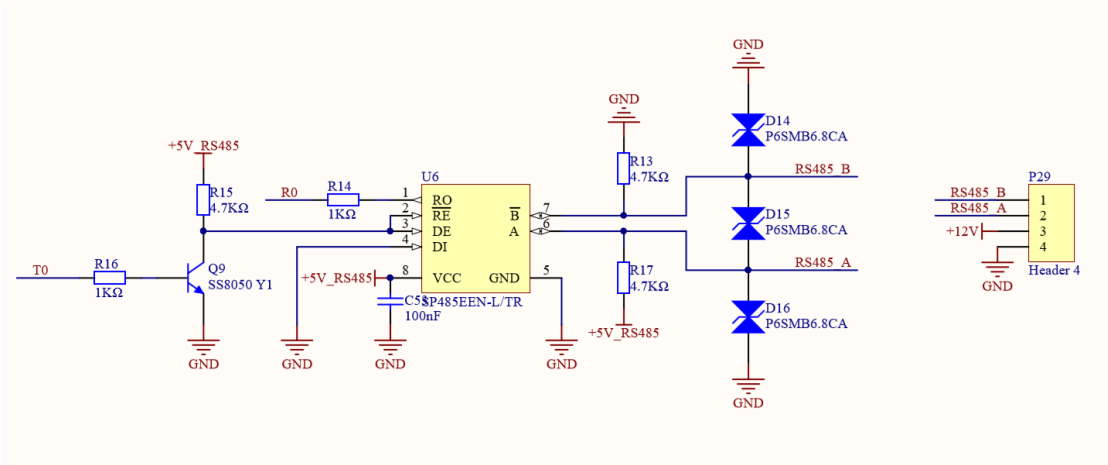
```
uint16_t adc_ch3 = 0;
void loop() {
    adc_ch0 = InVoltage(0);
    adc_ch1 = InVoltage(1);
    adc_ch2 = InVoltage(2);
    adc_ch3 = InVoltage(3);
    Serial.printf("ch0:%d ch1:%d ch2:%d ch3:%d\n", adc_ch0, adc_ch1, adc_ch2, adc_ch3);

    M5.Lcd.setCursor(40, 100, 4);
    M5.Lcd.printf("ch0:%05d ch1:%05d\n", adc_ch0, adc_ch1);
    M5.Lcd.setCursor(40, 130, 4);
    M5.Lcd.printf("ch2:%05d ch3:%05d\n", adc_ch2, adc_ch3);
    delay(500);
}
```

RS-485 转换模块

描述

RS-485 是工业控制场景中一种非常常见的电气特性标准，其通信采用的差分信号能够有效的抵抗电子噪声干扰。 Demoboard 上的 RS-485 转换模块能够将普通的 TTL 电平信号转换为 RS485 电平信号，实现协议的转换，进而控制相应类型的设备



硬件连接

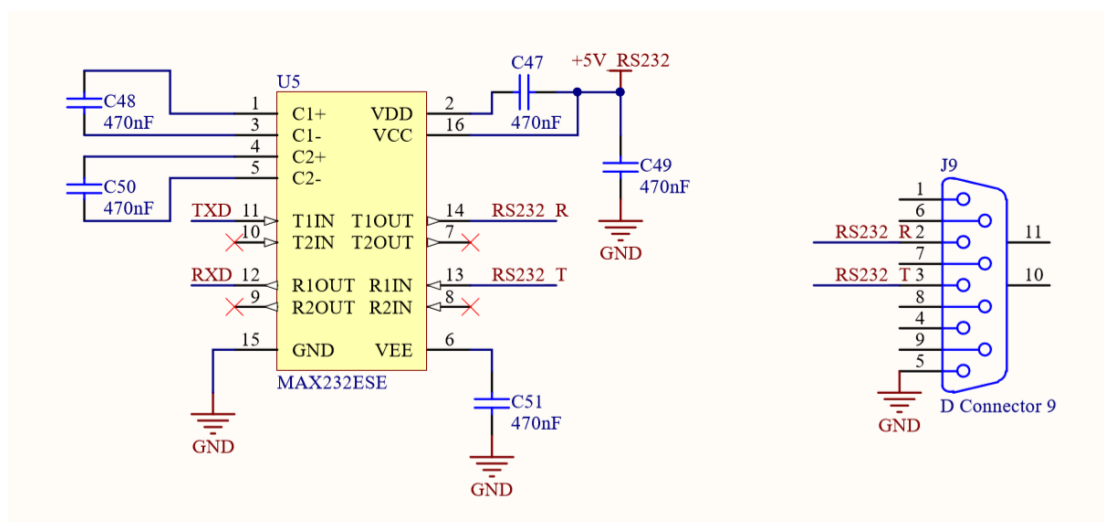
使用串口进行通信的设备，在使用时可以将其连接到 M5Core 默认的串口引脚 PIN17(TXD)、PIN16(RXD)，


```
}  
  
if(Serial2.available()) {  
    int ch = Serial2.read();  
    Serial.write(ch);  
}  
}
```

RS-232 转换模块

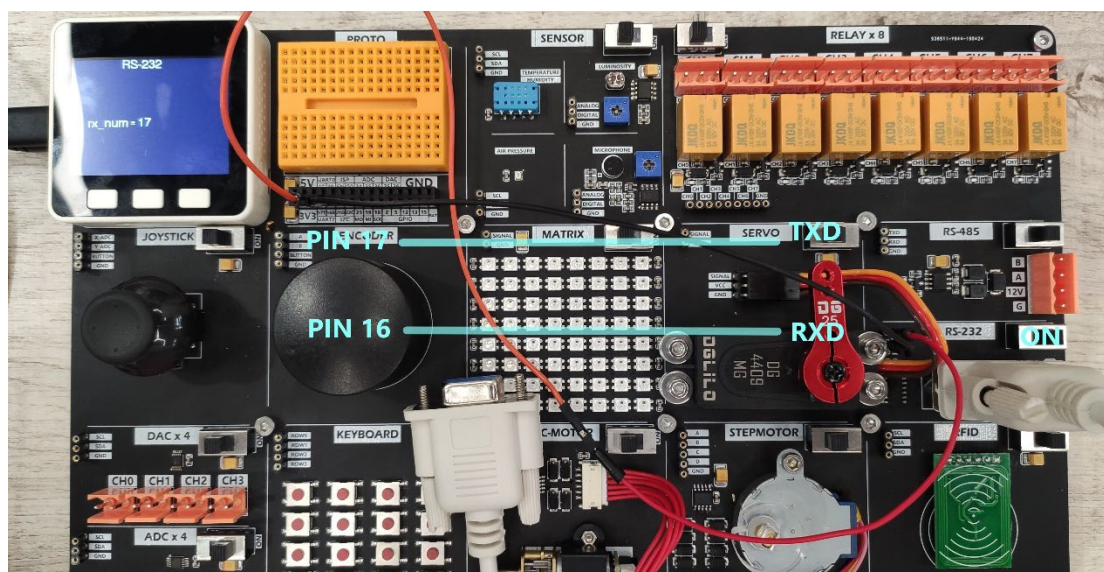
描述

RS-232 转换模块能够将 TTL 电平信号转换为 RS232 电平信号，实现协议的转换，进而控制相应类型的设备



硬件连接

使用串口进行通信的设备，在使用时可以将其连接到 M5Core 默认的串口引脚 PIN17 (TXD)、PIN16 (RXD)，



案例

接收来自 Serial2 的内容并转发至 Serial0(USB)，接收来自 Serial0(USB) 的内容并转发至 Serial2。

```
#include <M5Stack.h>

void setup() {

    M5.begin();
    M5.Power.begin();
    Serial.begin(115200);

    // Serial2.begin(unsigned long baud, uint32_t config, int8_t rxPin, i
nt8_t txPin, bool invert)
    Serial2.begin(115200, SERIAL_8N1, 16, 17);
    pinMode(5, OUTPUT);
    digitalWrite(5, 1);
}

void loop() {

    if(Serial.available()) {
        int ch = Serial.read();
        Serial2.write(ch);
    }

    if(Serial2.available()) {
        int ch = Serial2.read();
        Serial.write(ch);
    }
}
```

附录

Example Github

<https://github.com/m5stack/DEMO-BOARD>



Pulls

Issues

Marketplace

Explore



 **m5stack** / **DEMO-BOARD**

 Watch

2

 Star

0

 Fork

<> Code

 Issues

 Pull requests

 Actions

 Projects

 Wiki

 Security

 Insights

 master

Go to file

Add file

 Code

 **vany5921** Update Readme.txt

 .vscode	secend update
 ADC	secend update
 DAC	secend update
 DC-Motor	secend update
 ENCODER	secend update
 FactoryTest	Update Readme.txt

Clone with HTTPS

Use Git or checkout with SVN using the web URL.



 Open with GitHub Desktop

 Download ZIP

About

No description, website, or topics provided.

 Readme

Releases

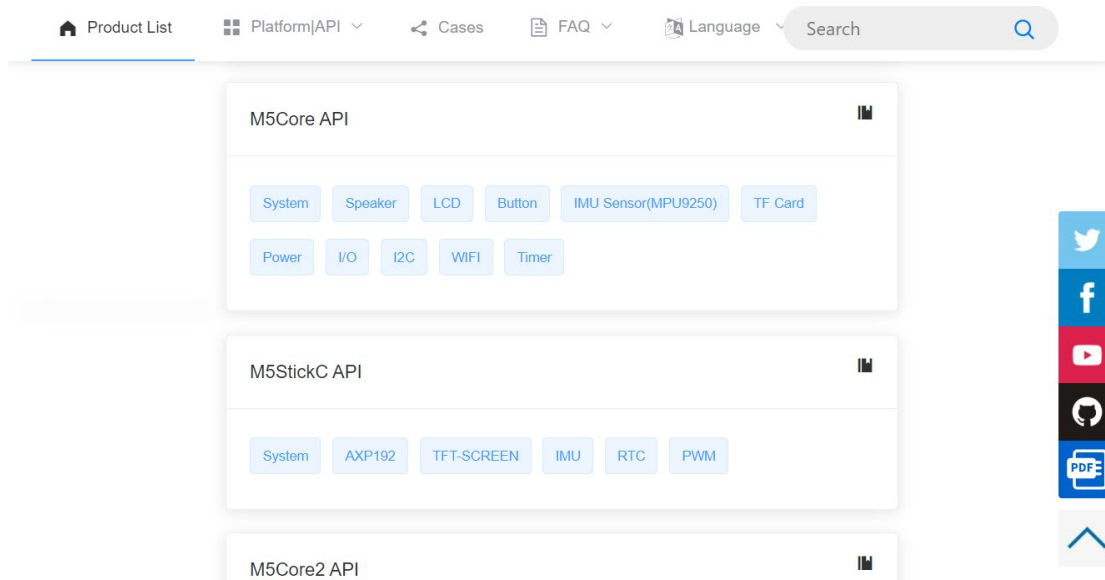
No releases published

[Create a new release](#)

Packages

Arduino API

https://docs.m5stack.com/#/en/arduino/arduino_home_page



Document & Datasheet

<https://docs.m5stack.com/#/en/app/demo-board>

