

M5STACK

DEMO BOARD



M5STACK



=

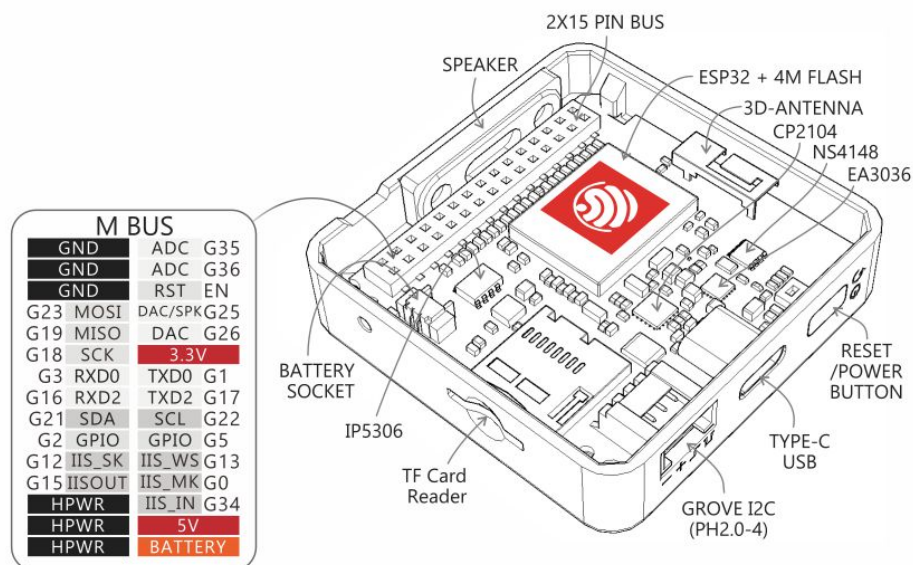
CONTENTS

Contents.....	1
Introduction.....	2
Description	2
M5Core	3
Hardware and Sensor	4
Accessories	5
ARDUINO IDE.....	6
Environment Setup.....	6
API.....	17
LCD.....	17
Button.....	20
Speaker.....	22
Module.....	24
Relay.....	24
.....	25
Microphone.....	27
Light Sensor.....	29
Joystick.....	31
Step Motor.....	34
DC-Motor.....	39
Servo.....	42
Keyboard.....	45
Encoder.....	48
DHT12 Temperature and humidity detection.....	51
BMP280 Air pressure detection.....	54
LED MATRIX.....	57
RFID.....	60
DAC.....	63
ADC.....	66
RS-232.....	72
appendix.....	73
Example Github	74
Arduino API	75
Document & Datasheet	75

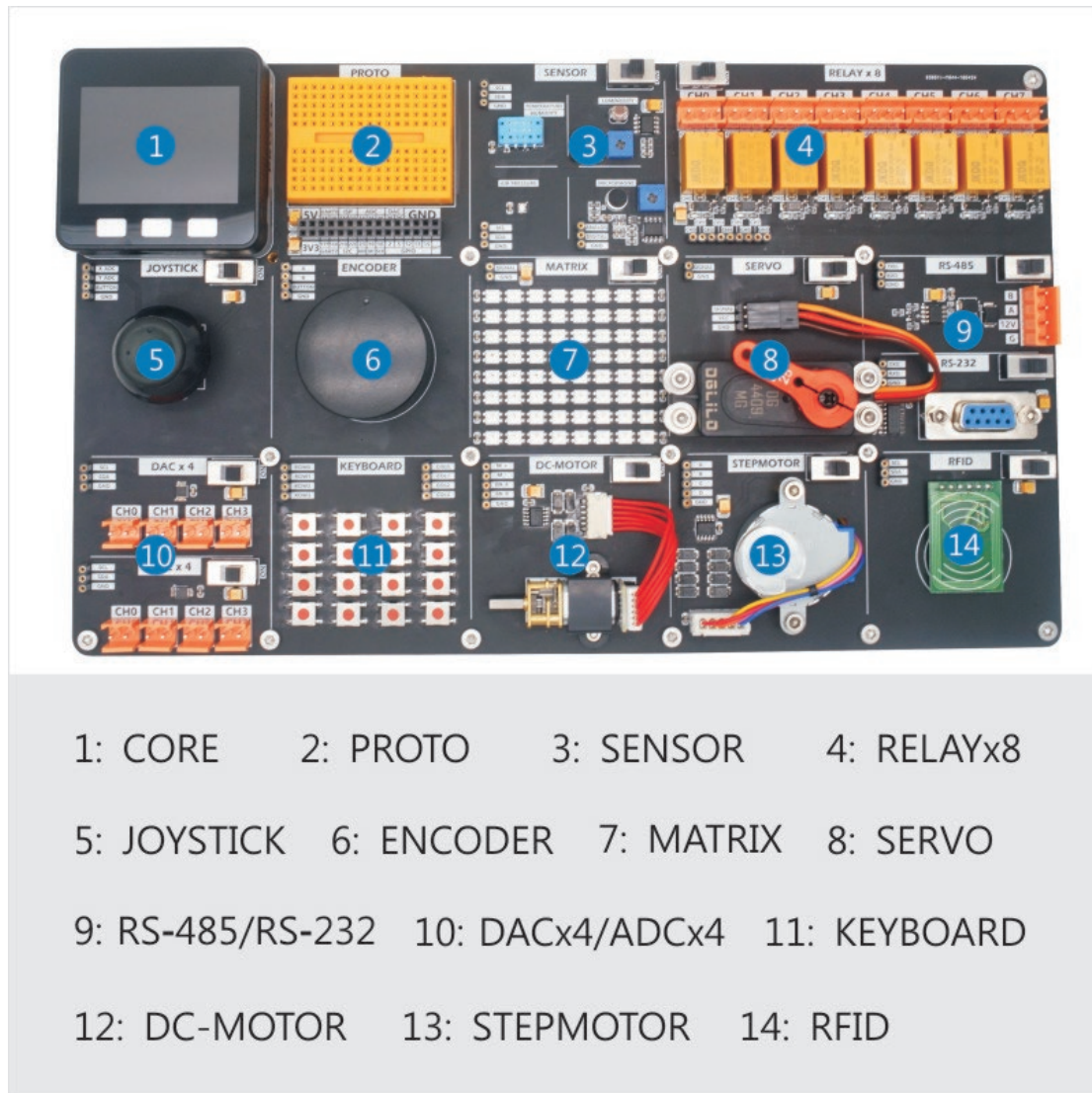
Equipped with a wealth of peripheral modules, including environmental detection related sensors, rocker control, rotary encoder, matrix buttons, radio frequency identification, mechanical motion control (including three-motor drive methods), three-color LED light board, relay Control, integrate multiple groups of ADC and DAC conversion circuits, and support RS485 and RS232 bus communication. Each module has an independent power switch. Combining the M5 main control with its Internet of Things properties as the control core, the DemoBoard development board covering "sound, light, electricity, force" and other aspects will be your learning hardware, A great tool for programming.

M5Core

The front of the BASIC controller is a 2-inch color TFT LCD screen with a resolution of 320x240. Built-in IoT chip ESP32, integrated Bluetooth, and WiFi modules in one. ESP32 is equipped with a dual-core 32-bit MCU with a main frequency of up to 240 MHz and a computing capacity of up to 650 DMIPS. The chip has abundant pin resources and integrates a wealth of hardware peripherals, including capacitive touch sensors, Hall sensors, low-noise sensor amplifiers, SD card interfaces, Ethernet interfaces, high-speed SDIO/SPI, UART, I2S, and I2C, etc.



Hardware and Sensor



1 - Use M5Core as the control core, compatible with Module stacking and Unit expansion system.

2 - Protoboard, M5-BUS bus expansion

3 - Environmental sensor series (temperature, humidity, pressure, light, microphone)

4 - 8 relay outputs

5 - Joystick input

6 - Rotary encoder

7 - 8x8 matrix RGB LED

8 - 10KG Serve

9 - RS-485, RS232 Communication function

10 - 4 channel DAC, 4 channel ADC

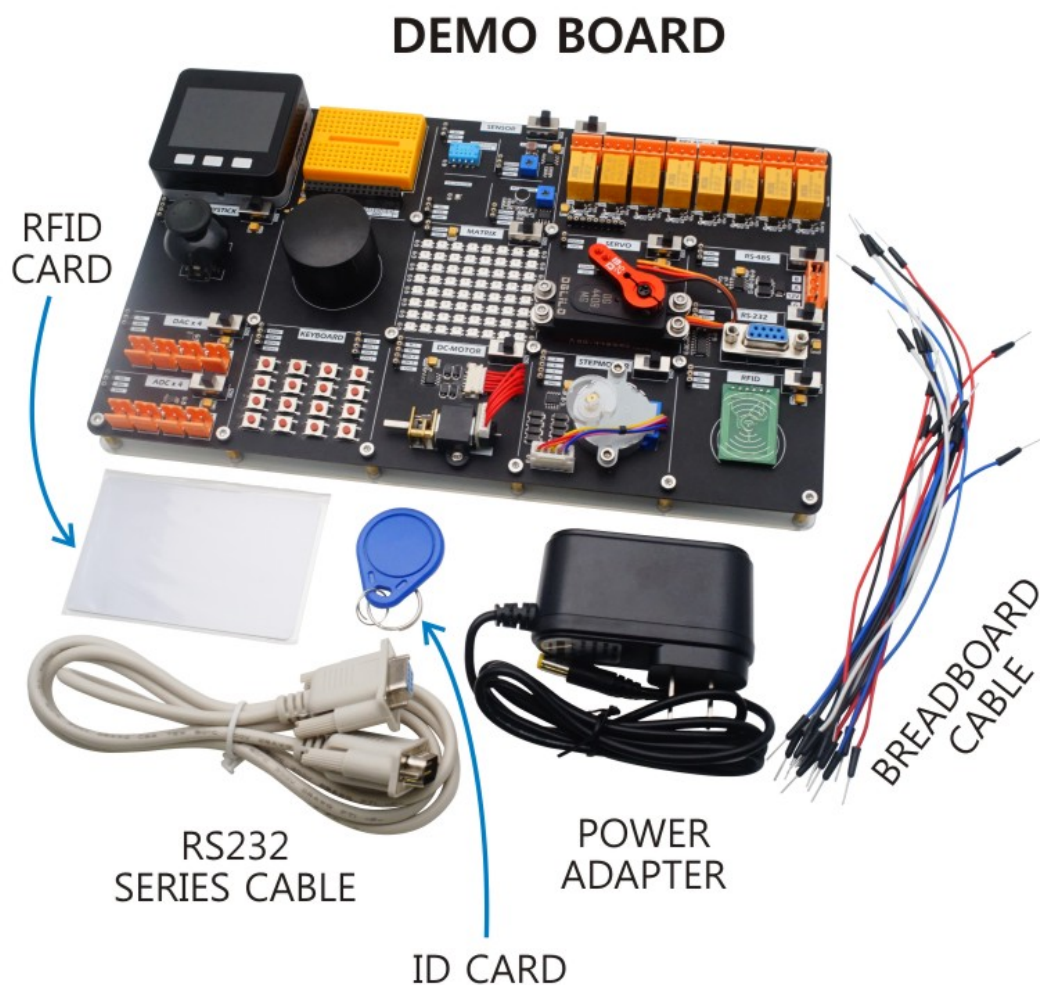
11 - 4x4 button matrix

12 - DC motor (with feedback)

- 13 - Four-phase five-wire stepping motor
- 14- Radio Frequency Identification Reader (RFID)

Accessories

The development board provides a series of supporting hardware, which can help users test the functions of each module on the board.



- 1x 12V power adapter (before using the development board, please connect the 12V power adapter to power the development board)
- 1x RS232 connection line (used as RS232 port connection)
- 1x RFID CARD (used to test radio frequency modules)
- 1x ID CARD (used to test radio frequency modules)
- 16x Bread wire (used for onboard jumpers to connect the interfaces of various hardware modules)

ARDUINO IDE

Environment Setup

Software Install

The development environment used in this tutorial is the Arduino IDE. This chapter will show you how to download related software, resource libraries, and some basic configurations

Arduino IDE is open-source software, with cross-platform capabilities, it can run on Windows, Mac OS X, and Linux. (The software installation operation demonstrated in this tutorial is based on the Windows 10 operating system)

First visit (<https://www.arduino.cc/en/Main/Software>), enter the download page and select the installation package corresponding to your operating system to download.

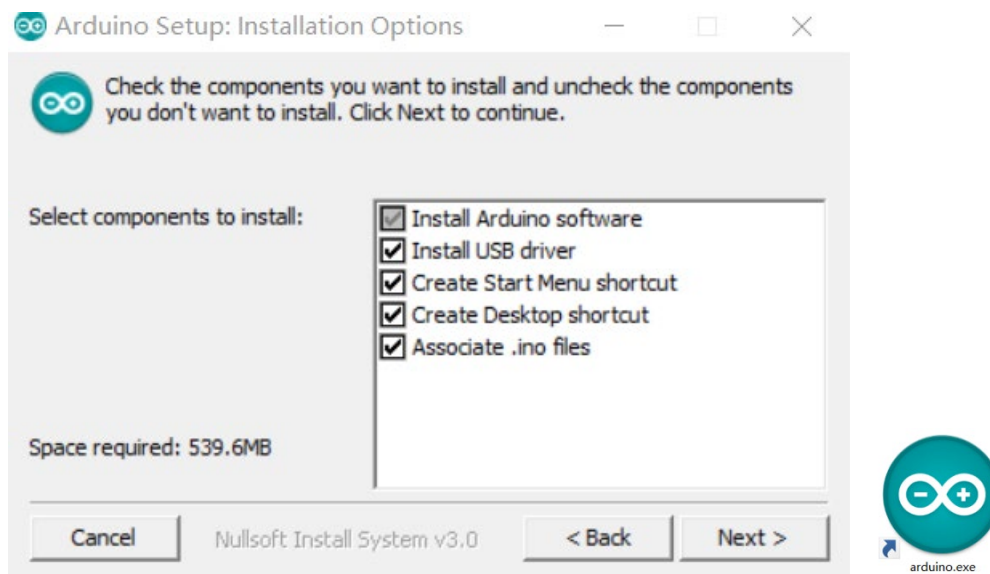


The screenshot shows the Arduino IDE download page. At the top is a teal navigation bar with the Arduino logo and links for HOME, STORE, SOFTWARE, EDU, RESOURCES, COMMUNITY, and HELP. There are also search, cart, and sign-in icons. The main heading is "Download the Arduino IDE". Below this is a large teal box containing the Arduino logo and the text "ARDUINO 1.8.10". The text describes the IDE as open-source software that runs on Windows, Mac OS X, and Linux, and is based on Processing. It also mentions that the software can be used with any Arduino board and refers to the "Getting Started" page for installation instructions. To the right of this box is a teal sidebar with several download options: "Windows Installer, for Windows XP and up" and "Windows ZIP file for non admin install", "Windows app" (Requires Win 8.1 or 10) with a "Get" button, "Mac OS X 10.8 Mountain Lion or newer", "Linux 32 bits", "Linux 64 bits", "Linux ARM 32 bits", and "Linux ARM 64 bits". At the bottom of the sidebar are links for "Release Notes", "Source Code", and "Checksums (sha512)".

Windows 10 users can click the "Windows Installer, for Windows XP and up" option, and the page will switch to the donation and download page. If you need to donate, you can click "CONTRIBUTE & DOWNLOAD",

and only download the installation package file and click "JUST DOWNLOAD".

After the download is complete, double-click to open the installation package. According to the installation prompt, choose to agree to the authorization agreement. For the configuration options and installation path of the installation, you can follow the default configuration if there is no special requirement. Click Next according to the prompts until the installation is complete. According to the default configuration, after the installation is complete, an Arduino shortcut icon will be created on the desktop. Double-click it to start the Arduino IDE.



Basic Introduction

At the top of the program are the tabs and the function menu bar, which provide a series of operation and configuration options, including compile, upload, new, open, save, and serial monitor functions.



Located in the middle is the programming area for code editing. Below the programming area is the log output area. When enabled, it will show you the current working status of the Arduino IDE, such as compiling or uploading the program and output the log information during the compilation and upload process, which is convenient for program debugging and Troubleshooting.


```
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

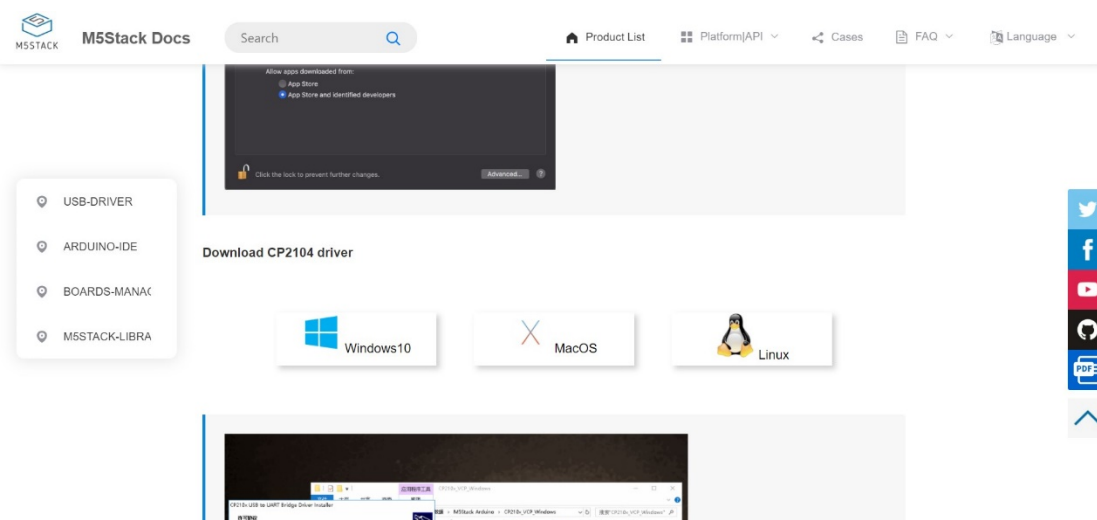
Install CP2104 driver

The main controller used by DemoBoard is BASIC. Before use, we need to install the matching USB driver (CP2104),

Visit the Arduino environment configuration tutorial page in the official documentation of M5Stack.

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

Select the "Install USB Driver" option in the catalog, jump to the bottom of the page, and select the CP210X driver compressed package of the corresponding operating system to download.



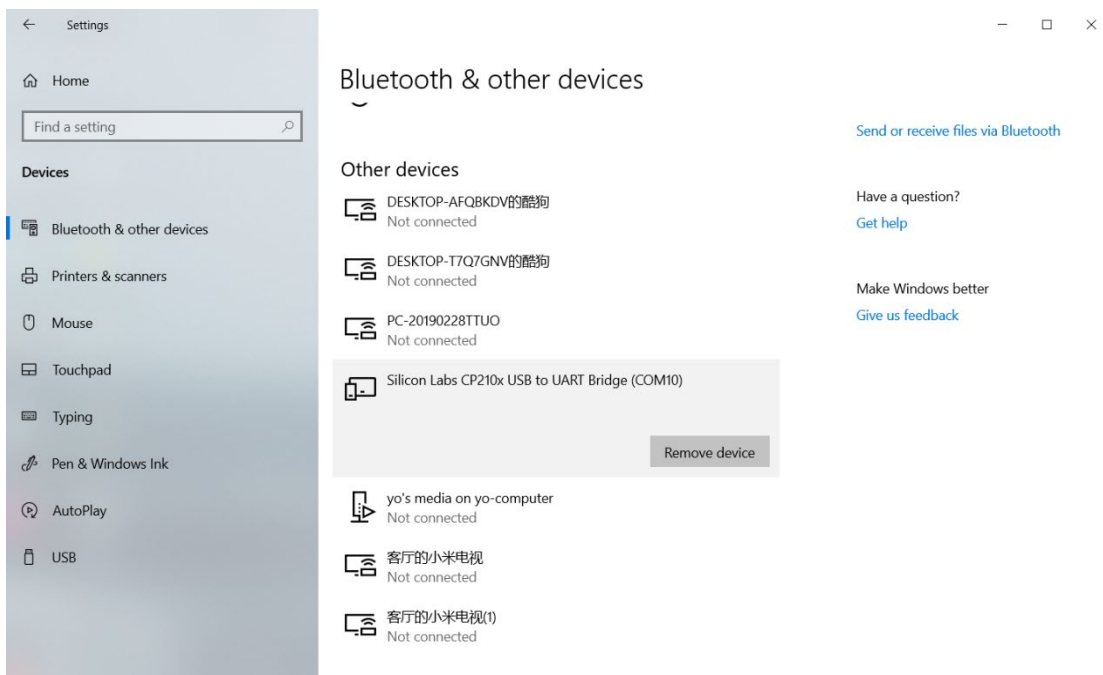
Unzip the downloaded compressed package, open the corresponding installation program (64-bit operating system selection _x64, 32-bit operating system selection _x86) according to the number of operating

systems, and click Next in turn according to the prompts until the installation is complete.

CP210x USB to UART Bridge Driver Installer



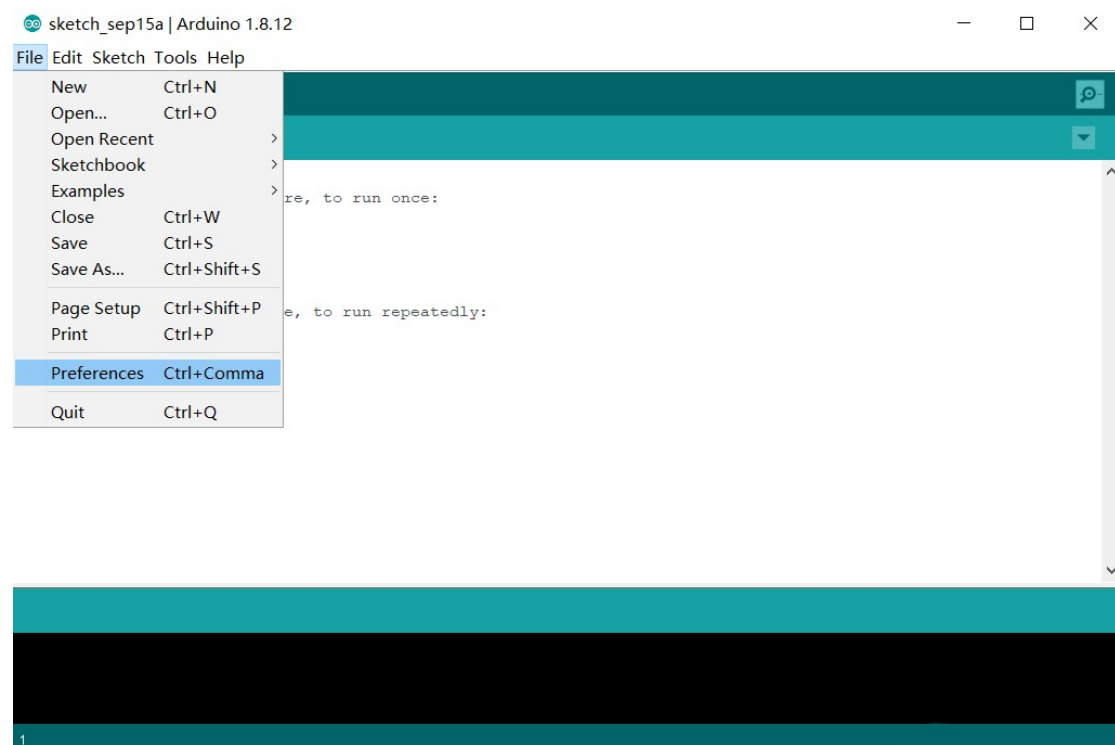
After completing the installation, use the Type-C data cable to connect the M5Stack device to the personal computer. Select the "Device" option from the Windows settings page to view the information of the currently connected device. As shown in Figure 2-11, the current M5Stack device has been successfully connected, and the currently used port is COM10. It means that the USB CP210X driver has been installed successfully and the device is operating normally.



ESP32 Board

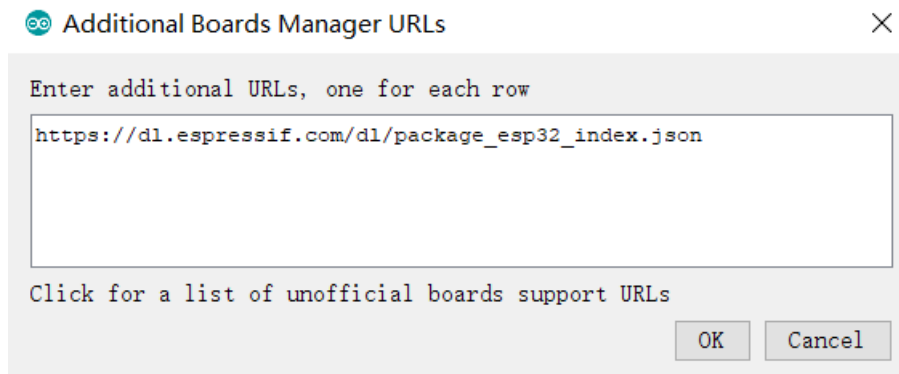
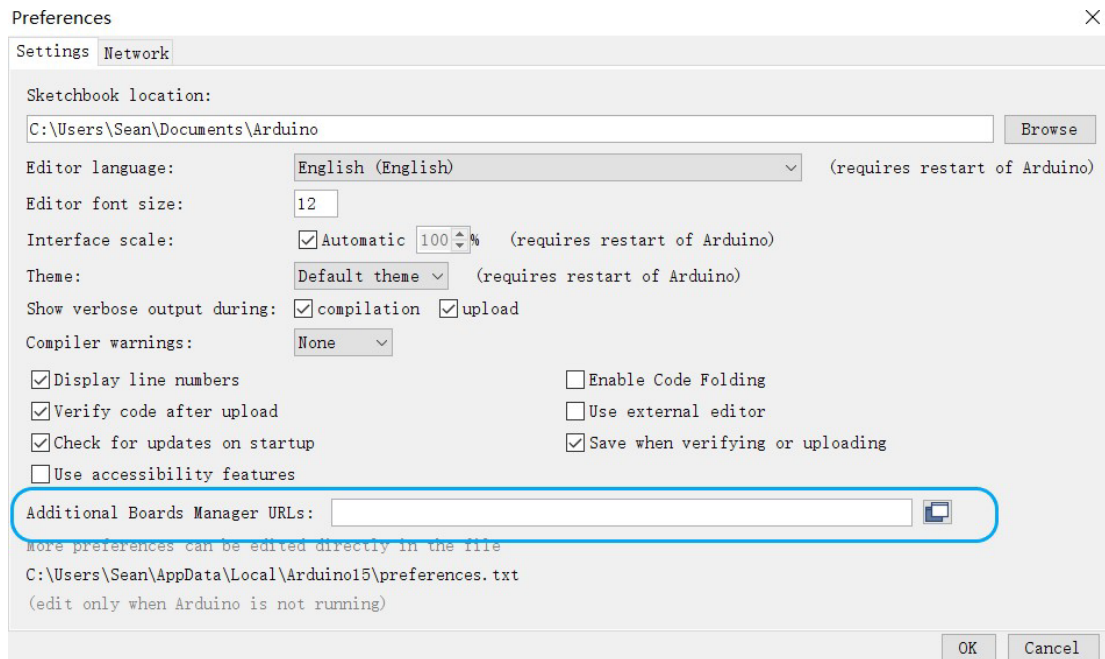
In addition to supporting Arduino's official development boards (such as Arduino Uno), Arduino IDE supports many controller chips on the market. When developing different devices, we need to specify the current development board information in the configuration options,

This section will show you how to configure the ESP32 board management information used by M5Stack in the Arduino IDE. Click the "File" tab at the top of the Arduino IDE page and open the "Preferences" setting.

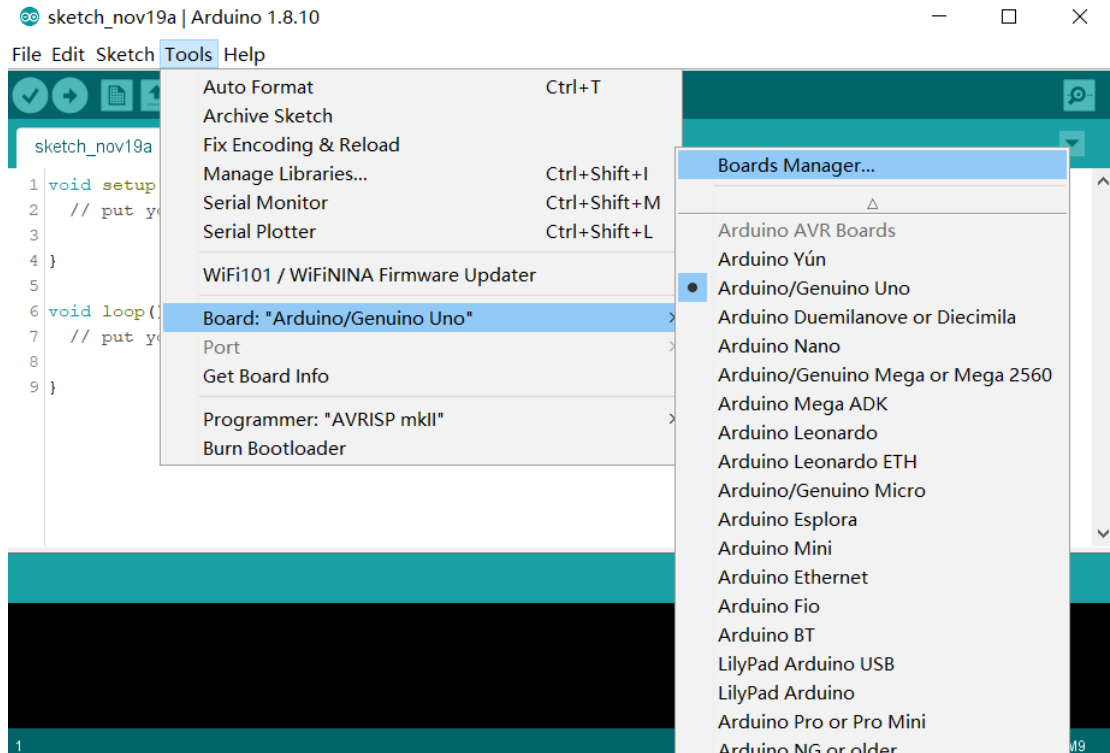


Click the Add button on the right side of the "Add Board Management Address" column, enter the URL address below in the pop-up window, and click OK to save.

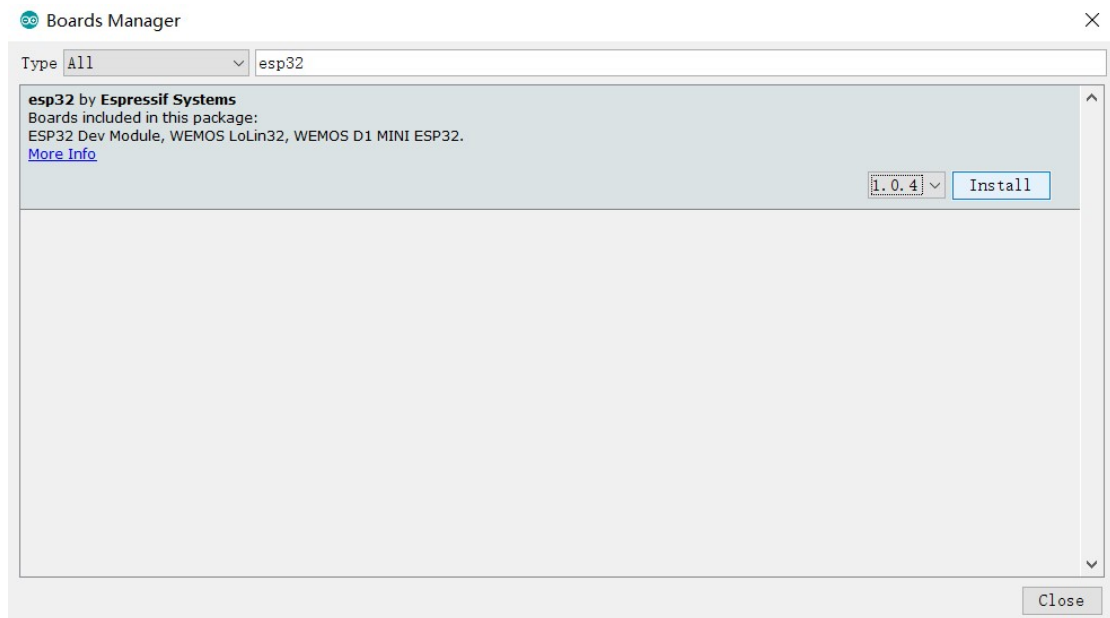
https://dl.espressif.com/dl/package_esp32_index.json



Also, in the "Preferences" setting page, you can set some display parameters, such as the editor text size, adding line number display, compiling/burning log prompts, and other auxiliary content. After saving the settings, select the "Tools" tab, point the mouse to the "Development Board" option, find the "Development Board Manager" in its sub-directory, and click Open.

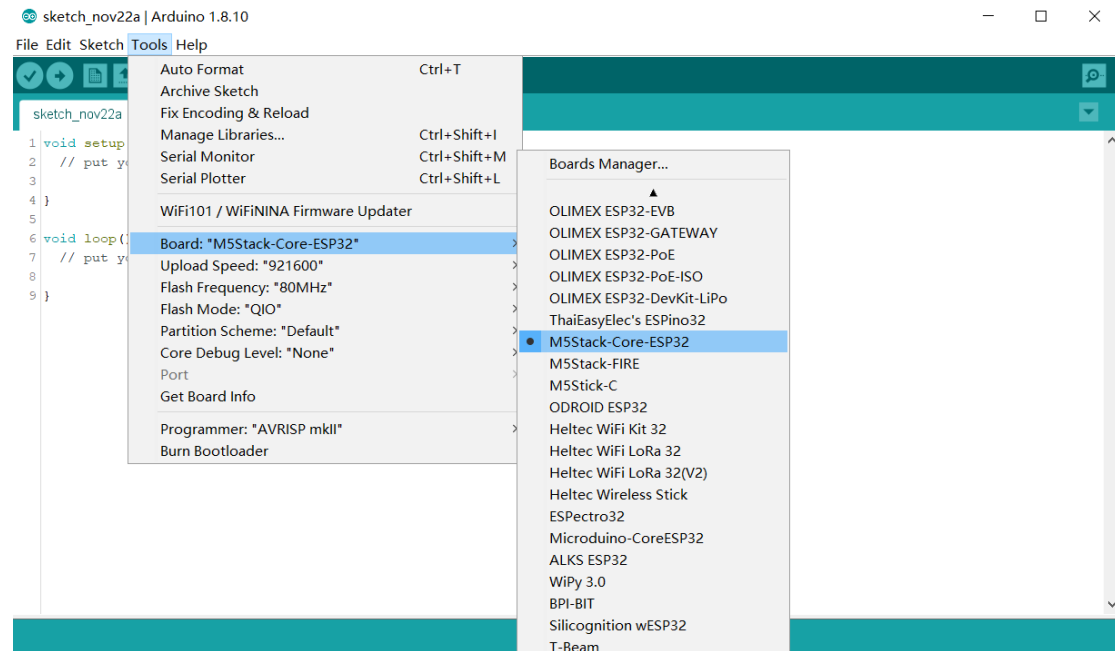


In the pop-up board management window, search for "esp32" and the content of the "esp32 by Espressif Systems" development board will appear.



Click the version number option to switch to install different versions. For better support, it is recommended to download the latest version. After the installation is complete, close the board manager and go back to the -->"Tools"-->"Development Board" option. Below the

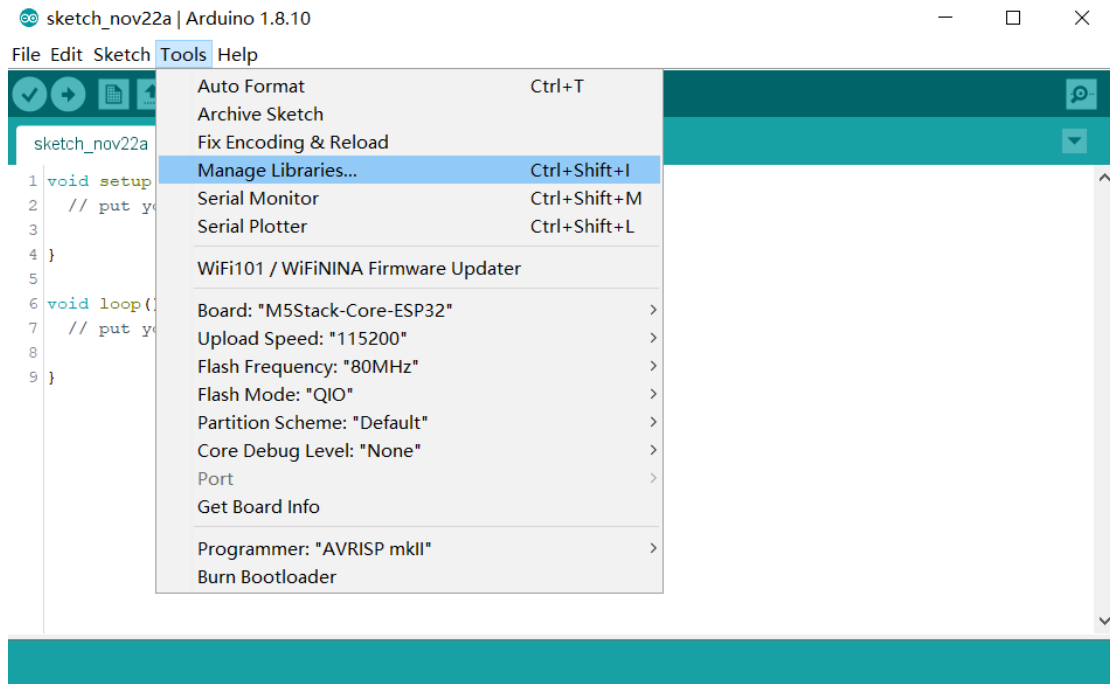
directory, you will find "M5Stack-Core-ESP32" and other M5Stack, ESP32 related board resources Options.



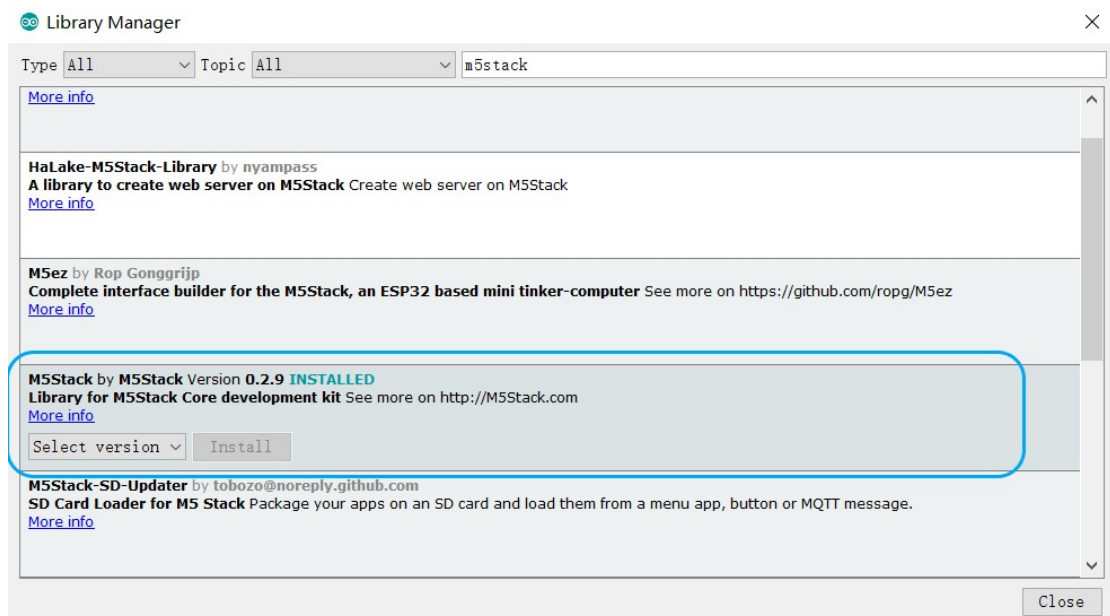
Select "M5Stack-Core-ESP32", connect the device to set the port and other parameters. The "Upload Speed" parameter can set the communication speed (bps) at which the Arduino IDE transmits executable files to the device when uploading programs. In the code debugging stage, it is often necessary to repeatedly burn the program. To speed up the development progress, you can choose a higher speed to reduce the burning time.

M5Stack Library

M5Stack officially provides an Arduino program library adapted to M5Stack devices. The program library encapsulates and integrates a series of M5Stack device hardware (such as sensors, LCD screens, etc.) function support and drivers. Through this library, users can develop M5Stack devices very conveniently and quickly. Select the "Tools" tab, open the "Manage Library" option, and click to open the Library Manager.



Search for "M5Stack" in the library manager, and you will find a series of libraries related to M5Stack. Here we choose the library maintained by the official and with the "by m5stack" field. Select the latest version and install it.



After completing the above steps, the development environment of M5Stack & Arduino IDE has been set up.

Hello World

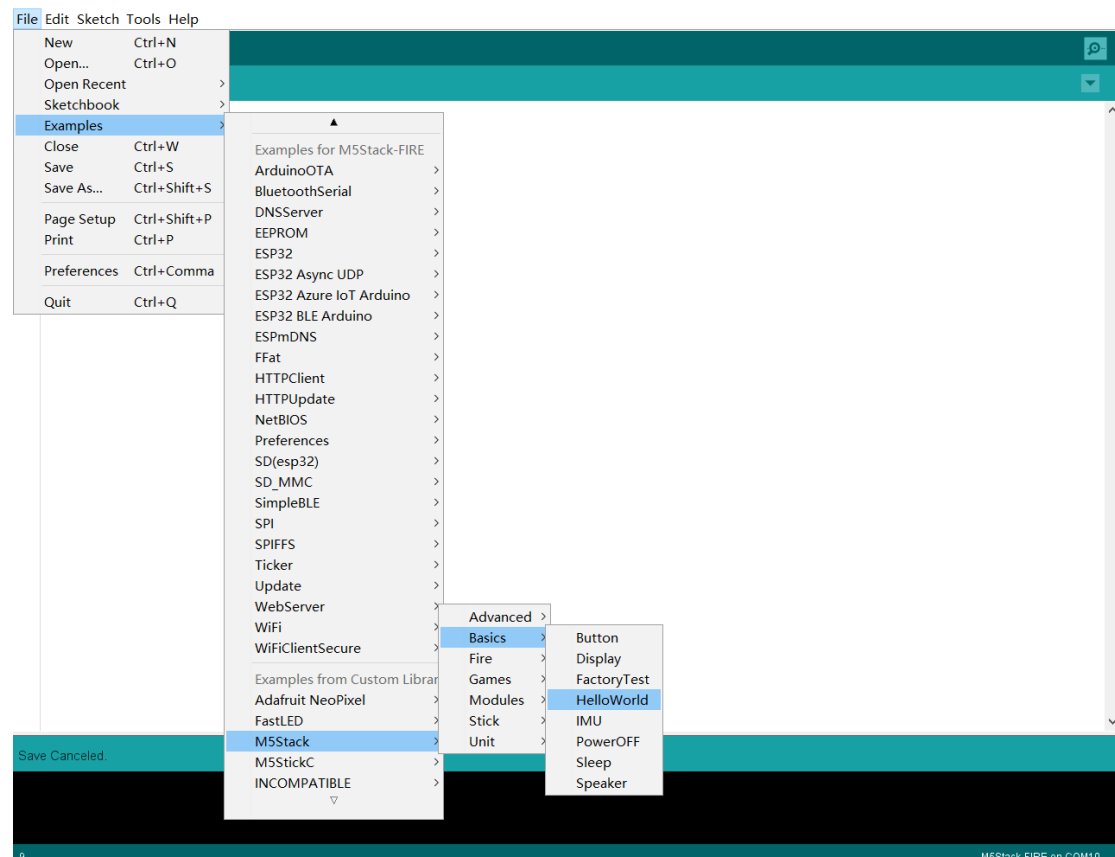
At the top of the program are tabs and function menu bars, which provide a series of operation and configuration options, and in the middle is the programming area for code editing.

Below the programming area is the log output area. When enabled, it will show you the current working status of the Arduino IDE, such as compiling or uploading the program and output the log information during the compilation and upload process, which is convenient for program debugging and Troubleshooting.

In addition to providing a series of functions, the M5Stack program also provides many programming cases for users' reference.

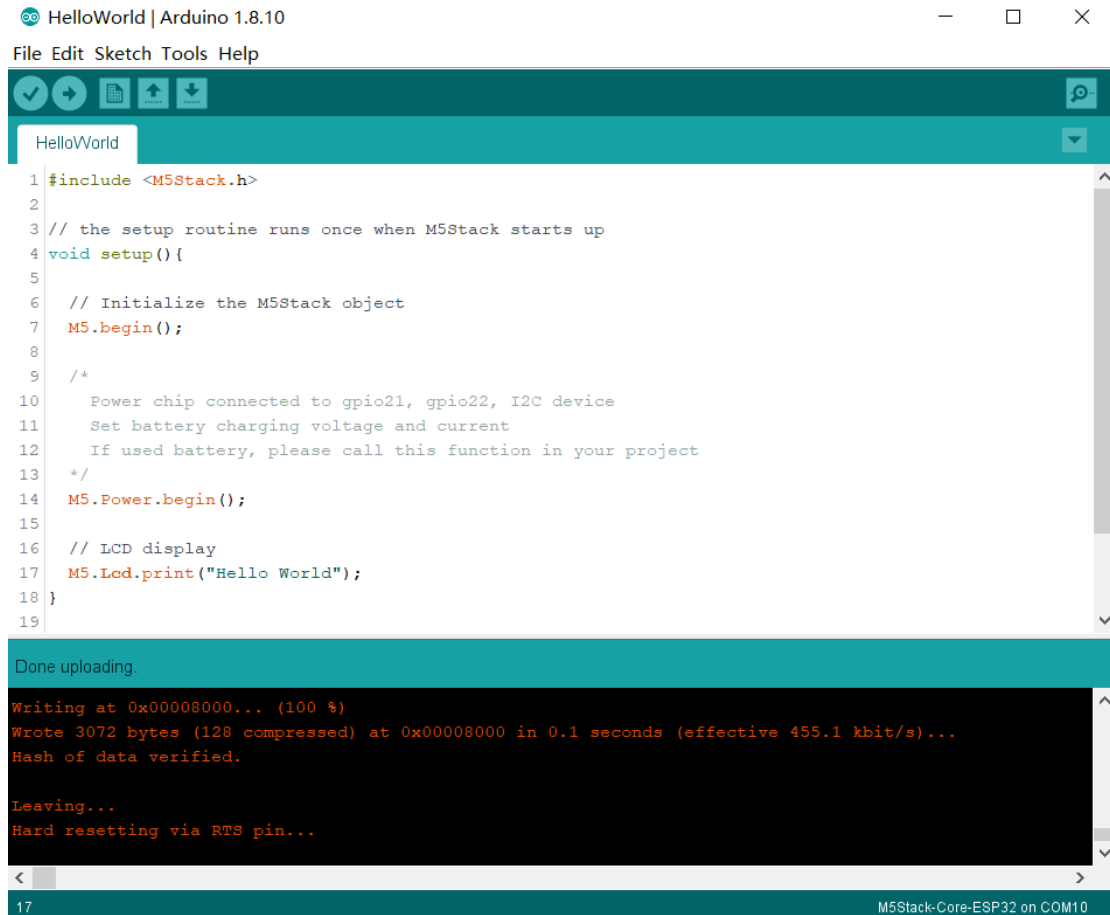
After completing the deployment of the development environment, we can try to run a simple case program to familiarize ourselves with the use of software and hardware.

Click the "File" tab at the top of the Arduino IDE page-->"Example Program"-->"M5Stack"-->"Basics"-->"HelloWorld" to open the example program.



Connect the M5Stack device to the computer through the Type-C data cable, select the corresponding port in the "Port" option in the

"Tools" tab, and confirm that the development board and other configuration information are correct.



The screenshot shows the Arduino IDE interface. At the top, the window title is "HelloWorld | Arduino 1.8.10". Below the title bar is a menu bar with "File", "Edit", "Sketch", "Tools", and "Help". A toolbar contains icons for checkmark, back, forward, save, and upload. The main editor area shows the following code:

```
1 #include <M5Stack.h>
2
3 // the setup routine runs once when M5Stack starts up
4 void setup(){
5
6   // Initialize the M5Stack object
7   M5.begin();
8
9   /*
10    Power chip connected to gpio21, gpio22, I2C device
11    Set battery charging voltage and current
12    If used battery, please call this function in your project
13   */
14   M5.Power.begin();
15
16   // LCD display
17   M5.Lcd.print("Hello World");
18 }
19
```

Below the code editor is a status bar that says "Done uploading." and a log window showing the following output:

```
Writing at 0x00008000... (100 %)
Wrote 3072 bytes (128 compressed) at 0x00008000 in 0.1 seconds (effective 455.1 kbit/s)...
Hash of data verified.

Leaving...
Hard resetting via RTS pin...
```

At the bottom of the IDE, the status bar shows "17" on the left and "M5Stack-Core-ESP32 on COM10" on the right.

Click the "Upload" button on the function menu bar, and the program will automatically start to compile. After the compilation is passed, the program will be uploaded automatically, and the log will output the percentage of the progress of the program upload at this time. At this time, please keep the computer and the M5Stack device connected until the program upload is completed. When the status bar prompts "Done uploading.", it means the program has been uploaded. Check the M5Stack device at this time, and you will find that the upper left corner of the LCD screen displays the words "Hello World". The screen is as expected by our program, and it is successfully displayed.



API

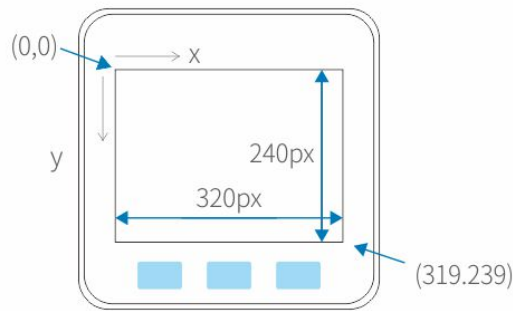
LCD

Description

When learning environment configuration, we ran the "HelloWorld" case program and successfully displayed text on the LCD. This color LCD screen can do much more than that.

The M5Stack library provides a series of commonly used program APIs for LCD drivers. Next, we will learn how to use these APIs to freely print content and draw patterns on the LCD screen of the M5Stack device.

This "320x240" pixel color LCD screen. In actual use, you can think of the screen as a plane coordinate system, with the horizontal axis is the x-axis and the vertical axis is the y-axis. Starting from the origin of coordinates (0,0) in the upper left corner to the diagonal (319,239) in the lower right corner, a rectangle formed by the diagonal is the display area of the screen. When the coordinates of the content leave this range, it will not be displayed normally.



Text display

In the program, we can not only modify the displayed text content but also set the display position of the content and the font size. By setting the text cursor, you can specify the position of a character in the text content. By setting the font-size value, the size of the font display can be adjusted.

Set cursor position:

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

Description: Set the text cursor to the coordinates (x,y)

parameter:

`uint16_t` x: x-axis coordinate
`uint16_t` y: y-axis coordinate

return: none

```
eg: M5.lcd.setCursor(40, 60);
```

Set font size:

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

Description: Set font size

parameter:

`uint8_t` size: Font size (allowable input range 1~7)

return: none

```
eg: M5.lcd.setTextSize(3);
```

Print Text:

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

Description: Print the specified information on the LCD screen.

parameter:

val: The output value can be integers, floating-point numbers, characters, and strings.

format: Specify the output format, BIN (binary), OCT (octal), DEC (decimal), HEX (hexadecimal). When outputting floating-point numbers, you can specify the number of digits left after the decimal point.

return: The number of bytes occupied by the content

eg:

```
M5.lcd.print(78); // Print result 78
```

```
M5.lcd.print(1.23456); // Print result is 1.23 (by default, two decimal places are reserved)
```

```
M5.lcd.print("M"); // Print M
```

```
M5.lcd.print("Hello M5Stack"); // Print Hello M5Stack
```

```
M5.lcd.print(78,HEX); // Print 4E
```

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

Example

Control LCD to display text dynamically, and display "Hello World" and "Hello M5Stack" at intervals

```
#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);
}
```

Button

Description

Three programmable physical buttons are provided on the control panel. With the existing API, interactive operations such as short press and long press can be realized.

Button click:

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

Description: When the button is pressed, the function returns the value "1" once, and then set to "0". When the button is released, the function return value is always "0".

parameter: none

return:none

eg:

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

Long press:

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

Description: Specify the duration of pressing the button. Actually, when the pressing duration is greater than the set duration, it returns "1", otherwise it returns "0".

parameter:

`int32_t` ms: Set long press time

return:none

eg:

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

Status update:

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

Description: Update the status of buttons A, B, C, Yang, and speakers. Note: When using a key program, you need to add "M5.update();" to the cycle of the program, so that the key value is constantly refreshed as the program runs, otherwise the program key is only valid once.

parameter: none

return:none

eg:

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

Speaker

Description

The speaker integrated on the controller emits various tones for audio playback or function prompts.

Play the specified tone:

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

Description: Play audio at a specified frequency and support setting the playing time.

parameter:

uint16_t freq: Set the frequency of playing sound
uint32_t duration:

return:none

eg:

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

Note: The hearing range of ordinary people is 20Hz~20KHz. Sounds larger or smaller than this range may not be recognized by people. Excessive use of high frequency or long-term high-frequency playback is likely to cause damage to the speaker hardware. Therefore, please pay attention to the appropriate range when filling in the parameters.

Example

combines the key function to control the speaker to emit different sounds.

```
#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();  
}
```


Warning:

When using M5Core for programming and development, you need to pay attention to the pin occupancy of the built-in hardware and avoid using the same pin to drive different hardware, resulting in failure to work properly. The following pins represent their application in M5Core, (For example, GPIO32 is the pin occupied by the LCD screen)

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

more complete pin mapping information, please visit

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

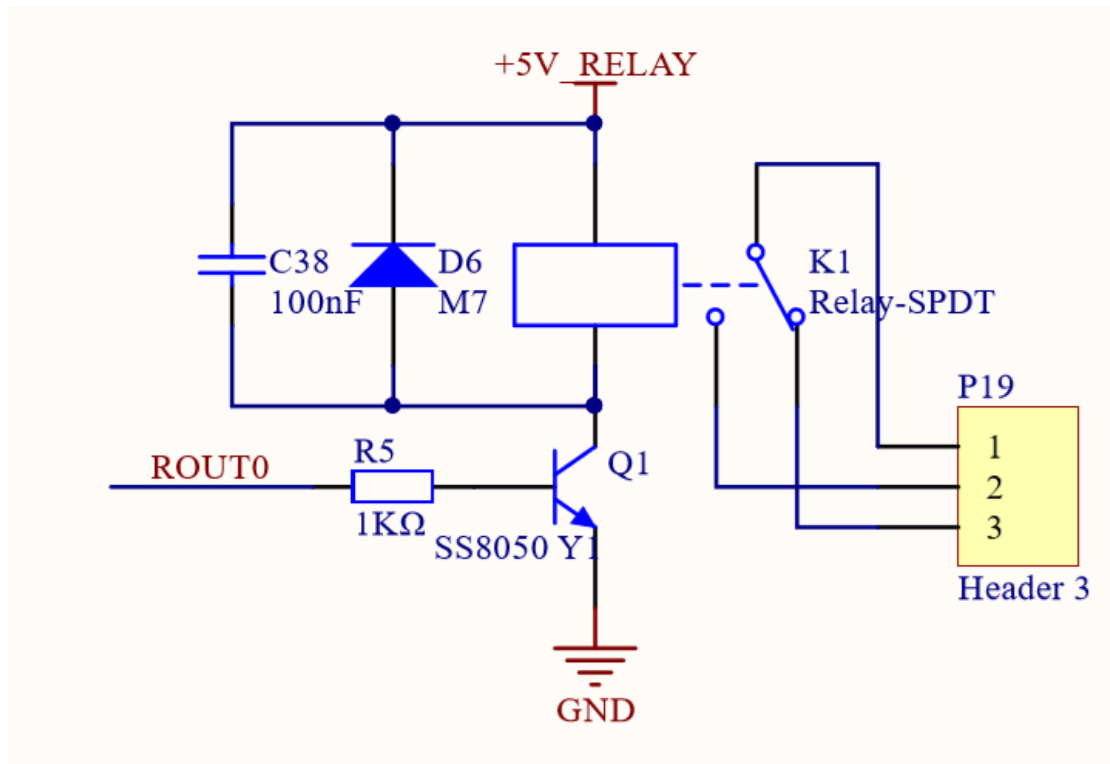
MODULE

Relay

Description

Relay is a relay control module. When the coil inside the relay is energized, it will generate a magnetic force to absorb the switch action, and then realize the switch control. Such a small current control high current electrical circuit scheme can control DC/3A-30V or AC/3A -220V level line is on and off.

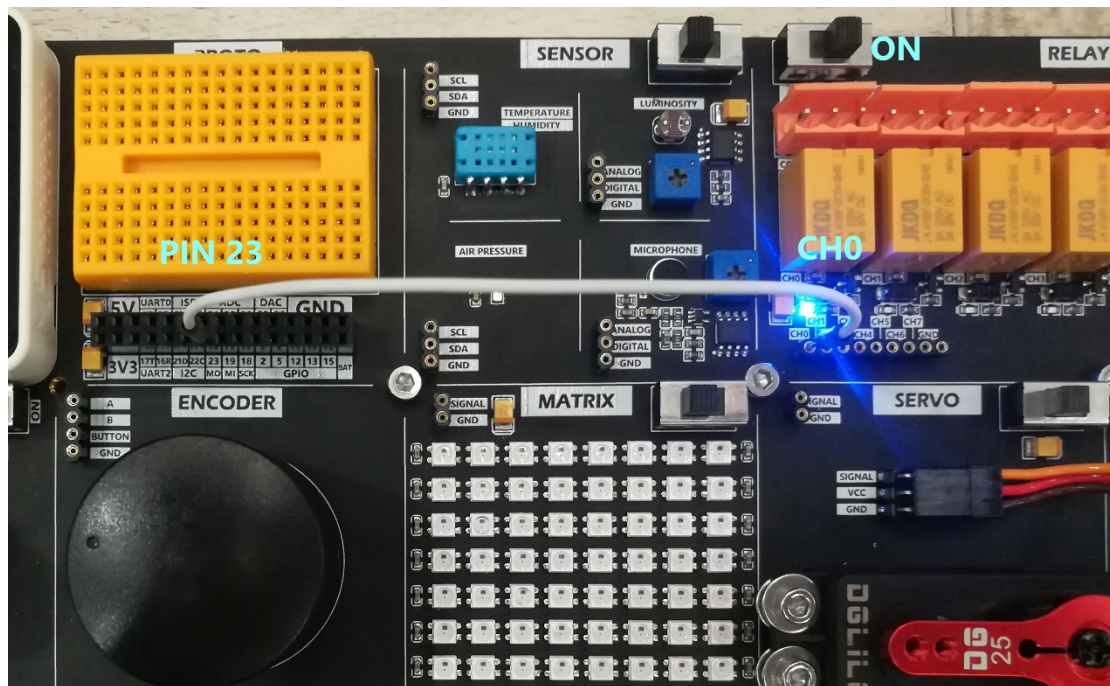
Each relay provides three control circuit pins, which are the common terminal COM, the normally closed terminal NC, and the normally open terminal NO. By default, NC and COM are connected, and NO and COM are disconnected by default. When the relay coil is energized, The switch relationship between the two and the common terminal COM will be reversed.



Hardware connection

When using the Relay relay control module, you only need to connect the control pin of the controller to the relay coil and turn on the independent power switch of the module to power it. In this way, when the controller pin outputs a high level, the relay coil will be energized and generated The magnetic field force absorbs the action

switch and makes the corresponding action.



Example

```
#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);

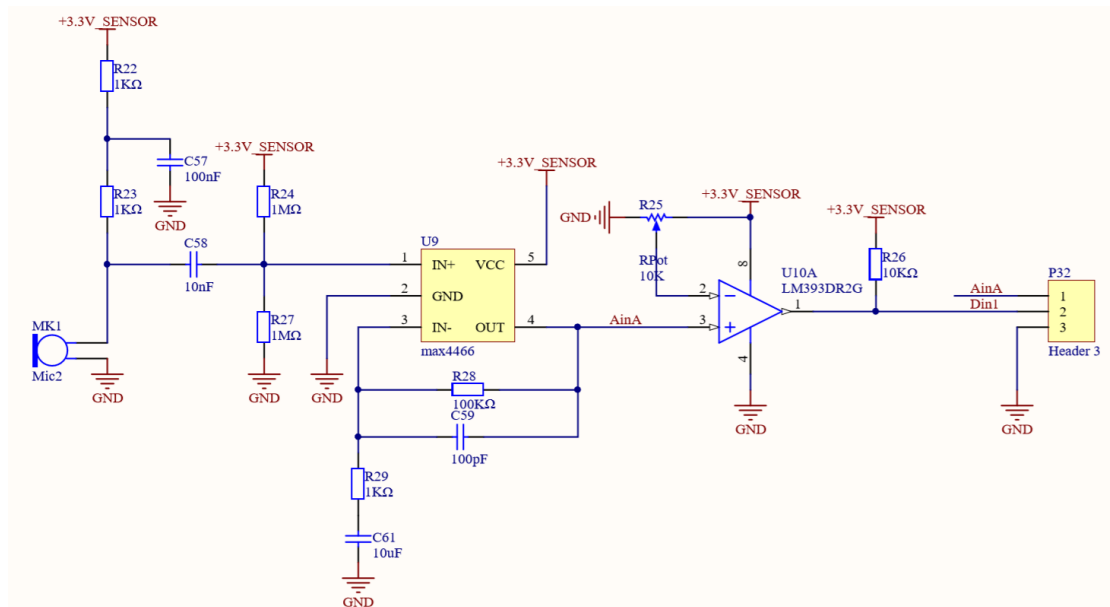
  digitalWrite(In_0, HIGH);

  delay(1000);
}
```

Microphone

Description

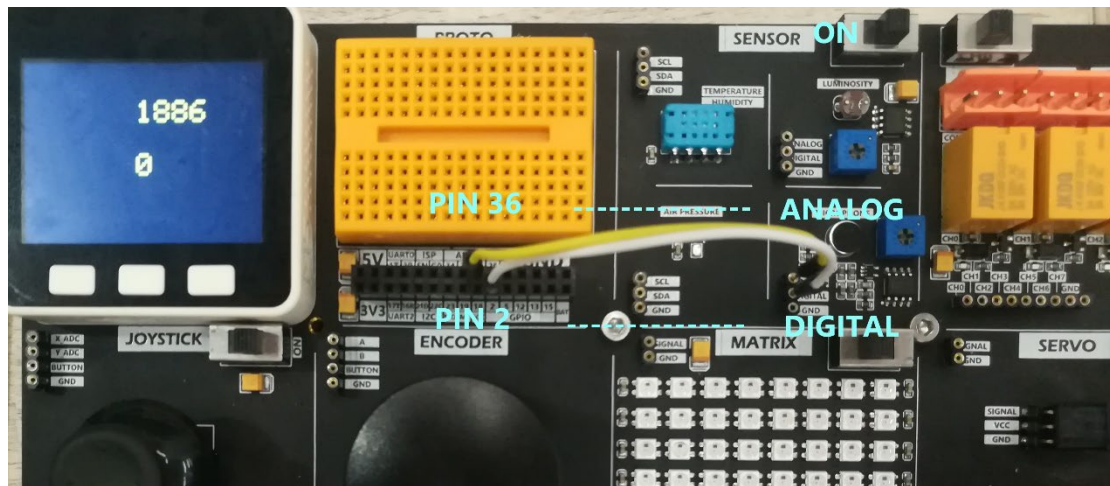
The Microphone module integrates the MAX4466 microphone preamplifier, which can effectively collect the analog and digital information input through the microphone. It also provides an adjustable resistor for the user to adjust the threshold value of identification.



Hardware connection

The Microphone module supports the reading of two signal values, digital and analog. It should be noted that not all GPIO interfaces of M5Core support AD conversion when making hardware connections. Therefore, the interface needs to be connected when using the analog reading function. Connect to pins that support ADC (such as

34/35/36), and use general GPIO for digital reading.



Example

```
#include <M5Stack.h>

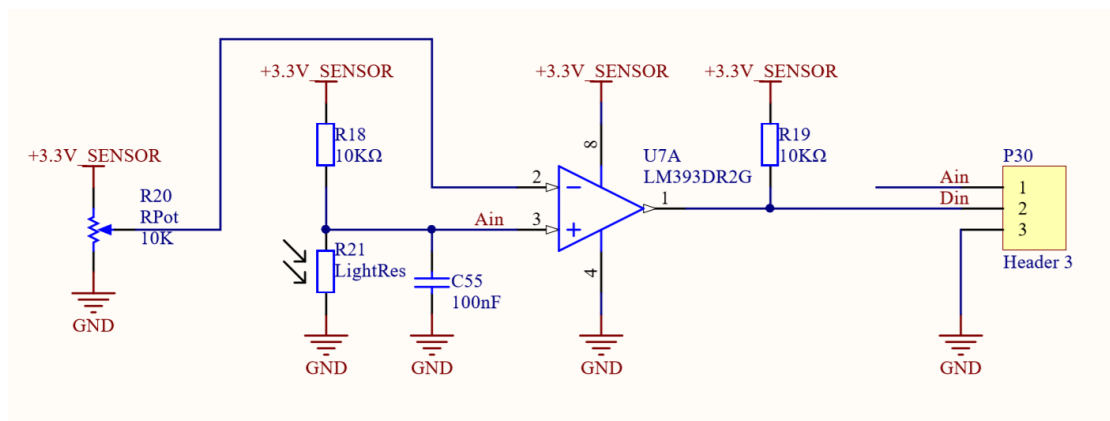
const int Analog = 36;
const int Digital = 2;
void setup() {
    // put your setup code here, to run once:
    M5.begin();
    pinMode(Digital, INPUT_PULLUP);
    dacWrite(25, 0);
    M5.Lcd.setCursor(100, 0, 4);
    M5.Lcd.print("MICROPHONE");
}
uint16_t a_data;
uint16_t d_data;
void loop() {
    // 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 Sensor

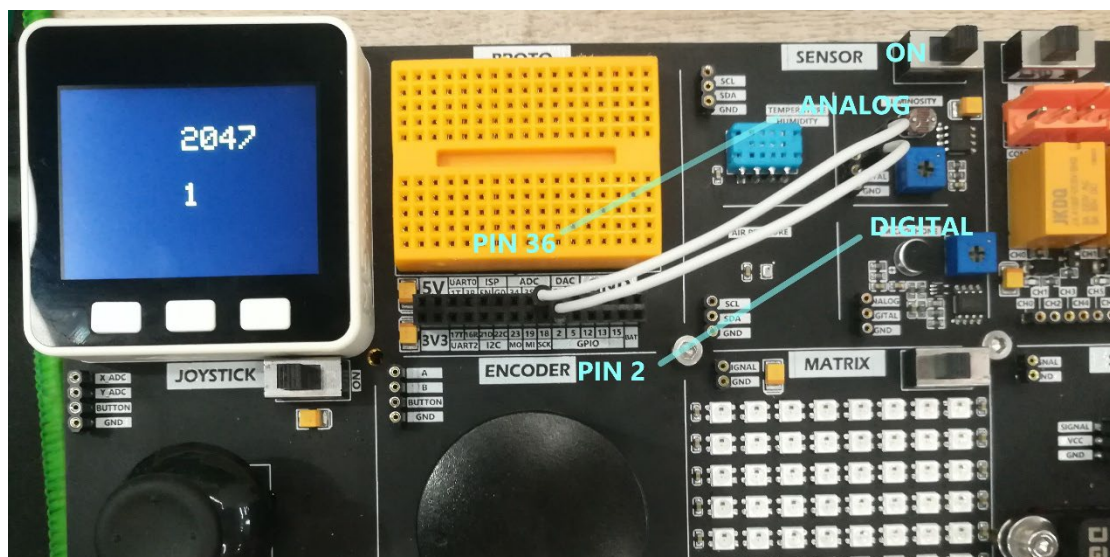
Description

Light Sensor is a light sensor with an integrated photoresistor, LM393DR2G voltage comparator, which can effectively collect light input analog and digital information. It also provides adjustment resistance for users to adjust the threshold value of recognition.



Hardware connection

The Light module supports the reading of both digital and analog signal values. When making hardware connections, it should be noted that not all GPIO interfaces of M5Core support AD conversion, so when using the analog reading function, you need to connect the interface Connect to pins that support ADC (such as 34/35/36) and use general GPIO for digital reading.



Example

```
#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("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);

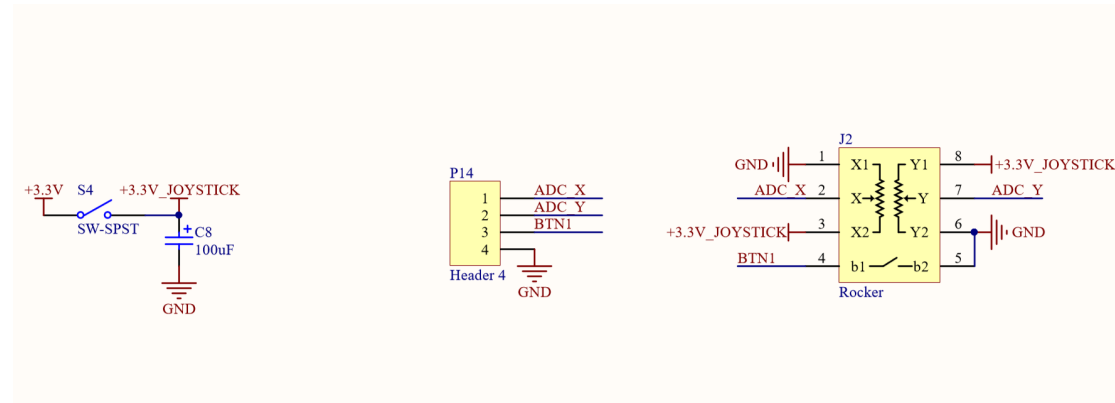
  delay(200);
}
```

Joystick

Description

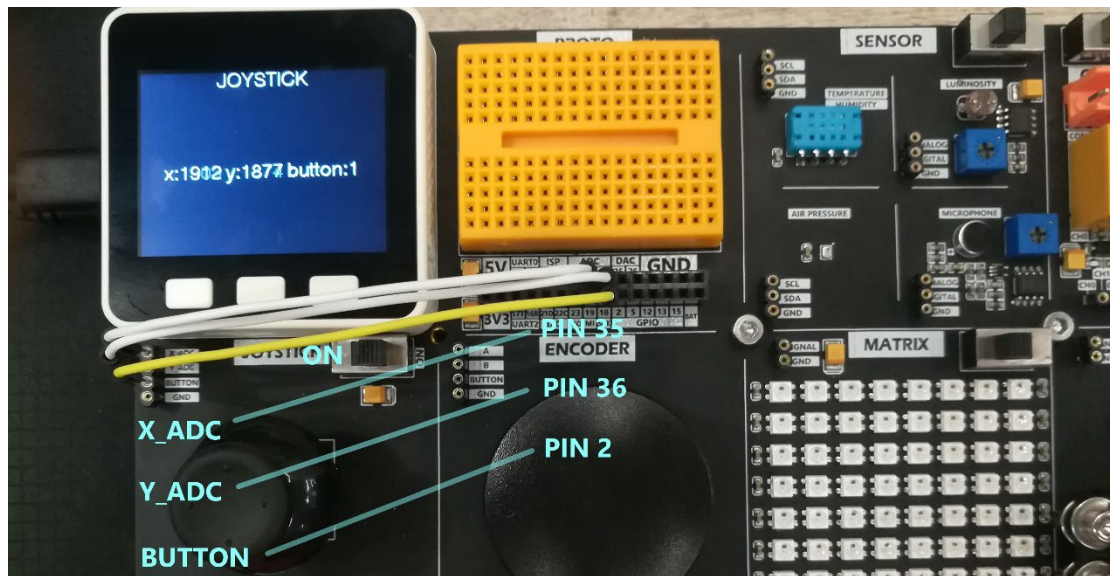
The joystick is a joystick input module. The module has built-in two sets of sliding rheostats and a key switch. When the joystick is operated, its internal resistance changes accordingly. Connect the X_ADC and Y_ADC interfaces of the module to the M5Core support. On the ADC converted pin, read the digital information of the joystick offset through the program.

When the BUTTON is pressed down on the joystick, its internal key switch will act accordingly to change the output signal to a low level.



Hardware connection

Connect the X_ADC and Y_ADC of the Joystick module to the pins that support the ADC function on the M5Core (in this example, PIN 35/36 will be used), and connect the BUTTON to PIN 2.



Example

```
#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);

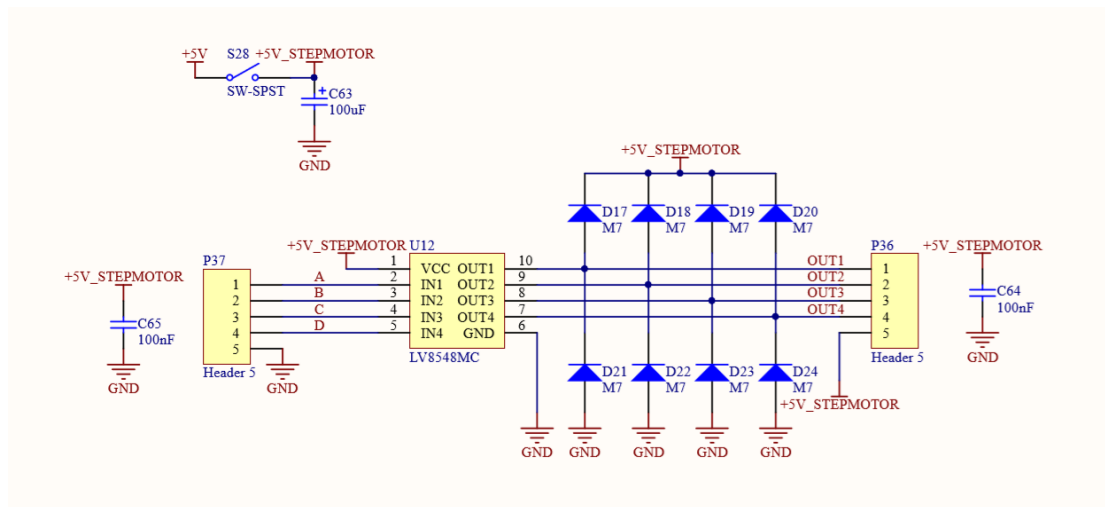
M5.Lcd.setCursor(30, 120, 4);
M5.Lcd.printf("x:%04d y:%04d button:%d\n", x_data, y_data, button_data);

delay(200);
}
```

Step Motor

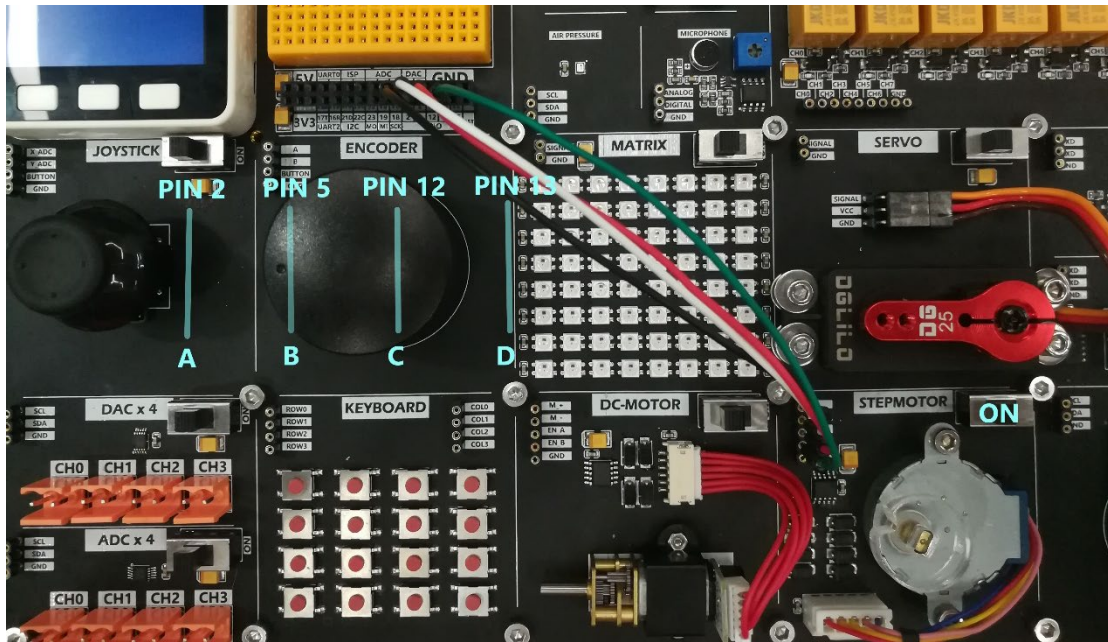
Description

Step Motor is a stepping motor module with integrated motor drive chip LV8548MC, using a five-wire four-phase connection. The A, B, C, D interfaces on the board can control the four magnetic pole coils in the stepping motor through the drive chip. The power is turned on and off to control the rotation of the motor.



Hardware connection

Connect the control pins PIN 2, 5, 12, and 13 to the A, B, C, and D ports of the stepper motor module, and turn on the independent power supply.



Different coil energization sequence will affect the rotation speed of the stepper motor, the number of beats for a complete revolution, the step value, and the stability during operation.

Example-1

Coil	Step1	Step2	Step3	Step4
MotorA	High			
MotorB		High		
MotorC			High	
MotorD				High

```
#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);
}

```

Example-2

Coil	Step1	Step2	Step3	Step4
MotorA	High	High		
MotorB		High	High	
MotorC			High	High
MotorD	High			High

```

#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]);
}
```

Example-3

Coil	Step1	Step2	Step3	Step4	Step5	Step6	Step7	Step8
MotorA	High						High	High
MotorB	High	High	High					
MotorC			High	High	High			
MotorD					High	High	High	

We can further subdivide the step, only need to modify the array of power-on logic, and the drive function.

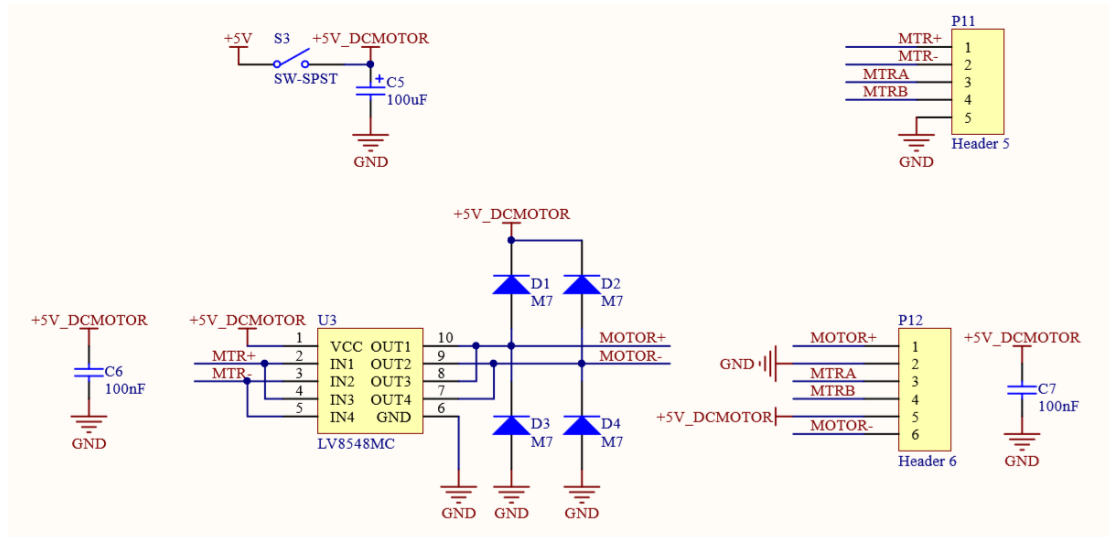
```
const int logic[8][4] = {  
    {1,1,0,0},  
    {0,1,0,0},  
    {0,1,1,0},  
    {0,0,1,0},  
    {0,0,1,1},  
    {0,0,0,1},  
    {1,0,0,1},  
    {1,0,0,0}  
};
```

```
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

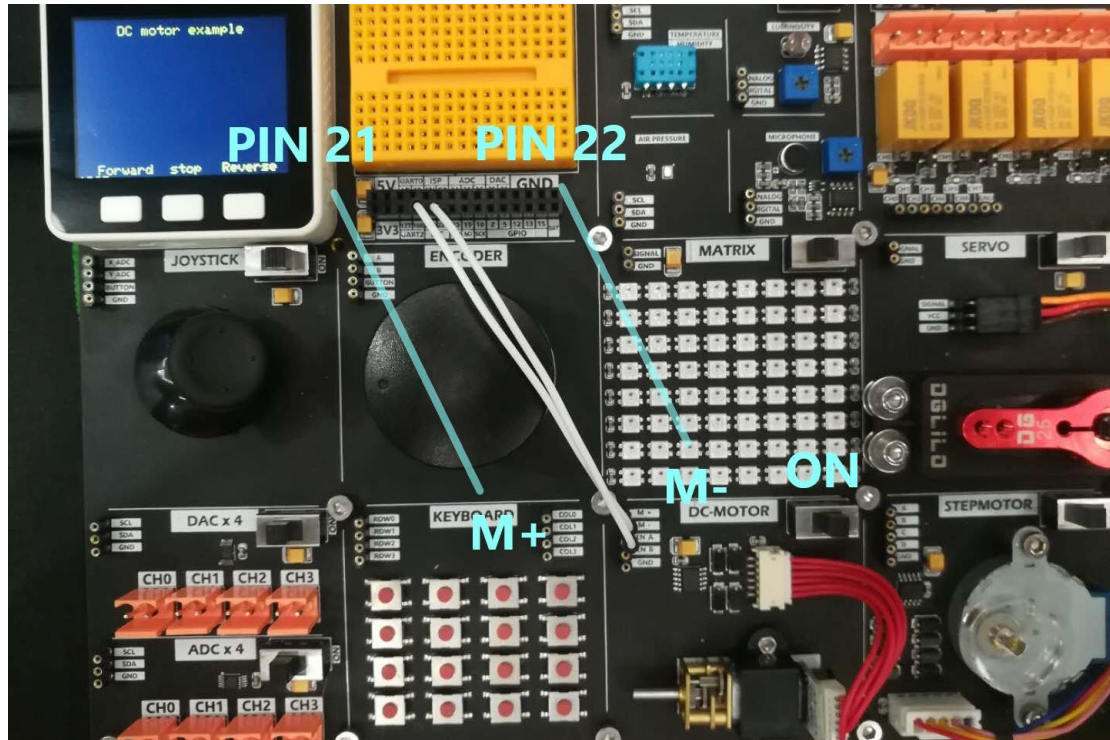
Description

DC-Motor is a DC motor module with feedback. The module has a built-in LV8548MC motor driver chip. It provides M+ and M- two interfaces for controlling forward and reverse rotation. When M+ is high and M- is low, The motor rotates forward. When M+ is low level and M- is high level, the motor is reversed. When M+ and M- are both low and low, the motor brakes.



Hardware connection

Connect the control pins PIN 21, 22 to the M+ and M- ports of the DC motor module respectively, and turn on the independent power supply.



Example

```
#include <M5Stack.h>

void setup() {

  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);
}

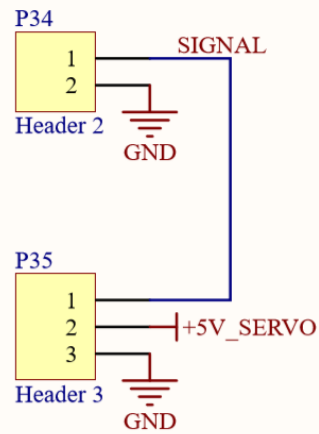
void loop() {
  M5.update();

  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

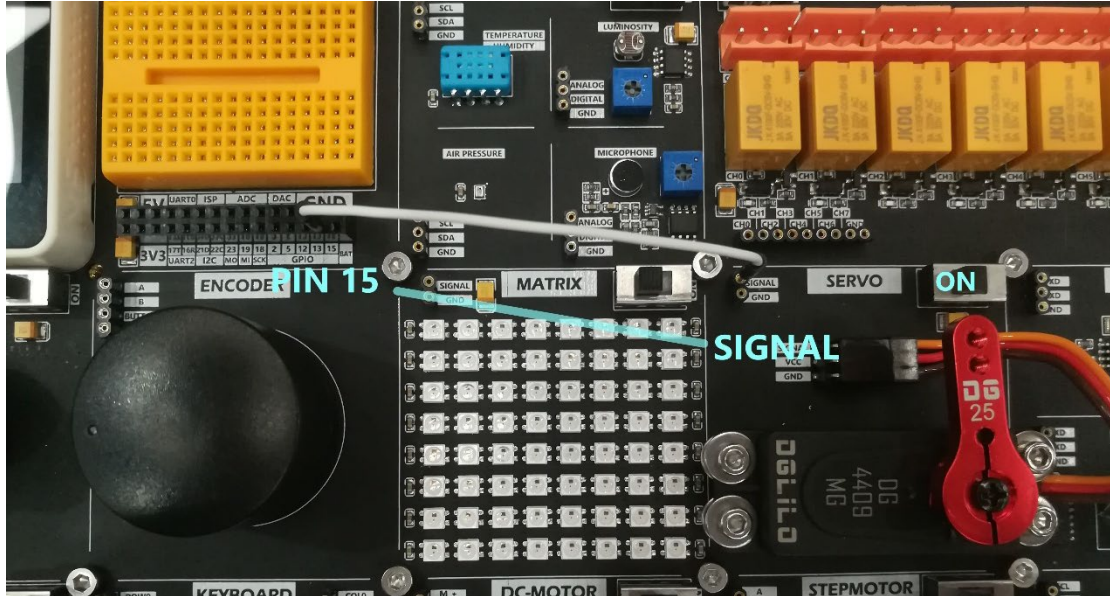
Description

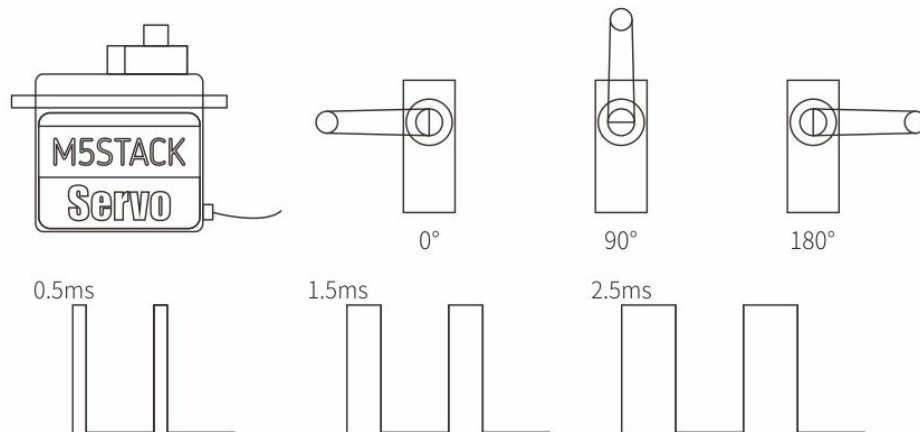
The development board is equipped with a steering gear ($0^{\circ}\sim 180^{\circ}$) with a torque of up to 10KG. The default 5V power supply is used for control by a single-signal bus. The driving signal is 50Hz, which is controlled to rotate to different angles according to the PWM duty cycle. PWM waveform is medium and high The level time is ($0.5\mu s\sim 2.5\mu s$) respectively to control the rotation of the steering gear ($0^{\circ}\sim 180^{\circ}$)



Hardware connection

Connect the control pin PIN 15 to the SIGNAL interface of the servo and turn on the independent power supply,





Example

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

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

void setup() {

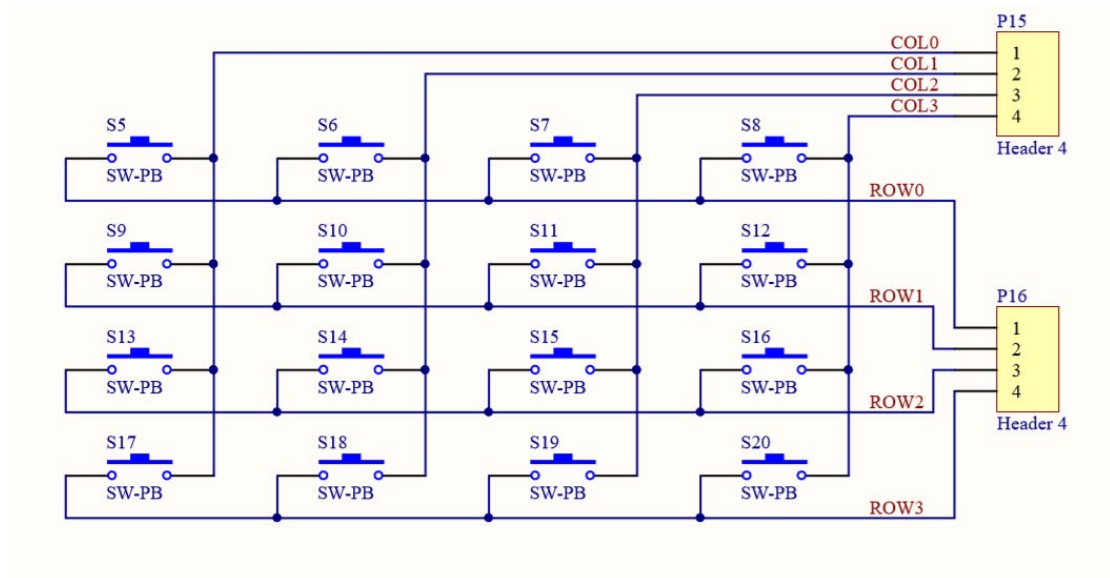
  M5.begin();
  M5.Lcd.setCursor(120, 110, 4);
  M5.Lcd.println("SERVO");
  ledcSetup(ledChannel, freq, resolution);
  ledcAttachPin(servo_pin, ledChannel);
}

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

Keyboard

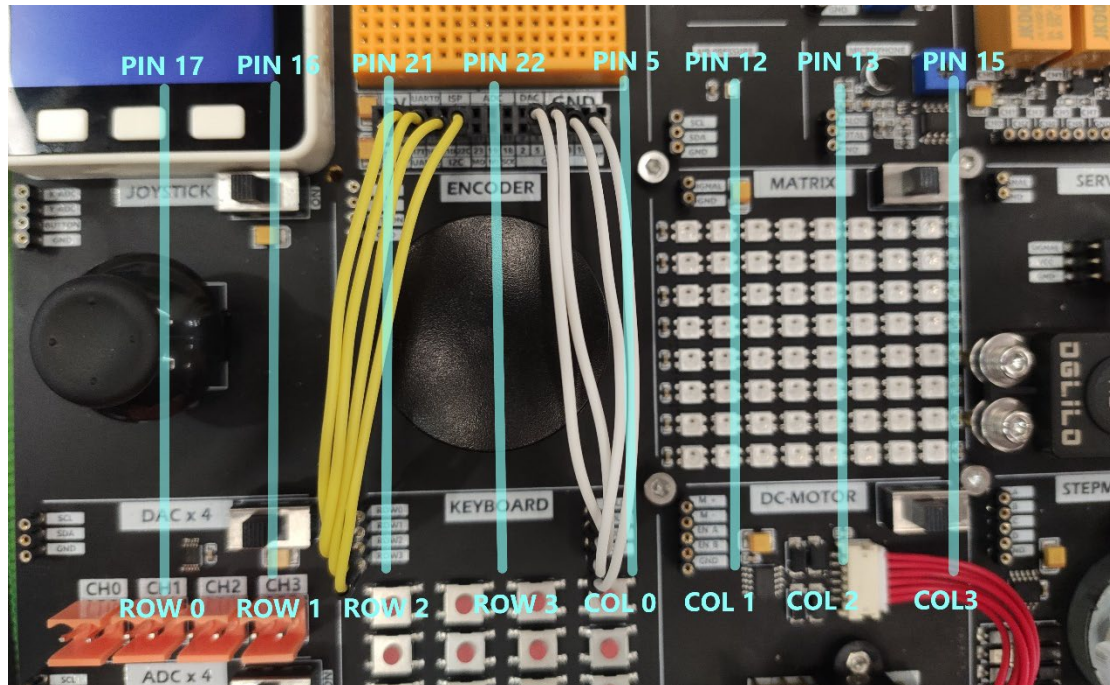
Description

A 4 x 4 matrix key module is provided on the development board. The specific key pressed can be detected by program scanning, which can provide you with richer key-value input and diversified control functions.



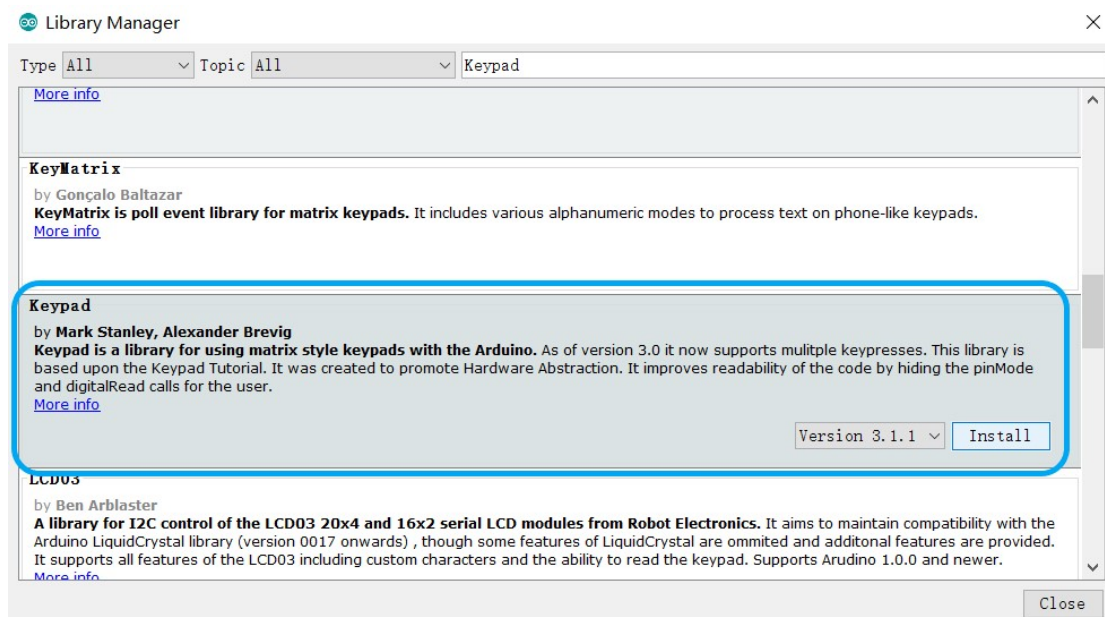
Hardware connection

Connect the control pins to each pin of the row and column of the button matrix, so that the program can scan the action of the button.



Example

In this case, the library `<Keypad.h>` is used, you can search and install it in the library management of Arduino.



```
#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");
}

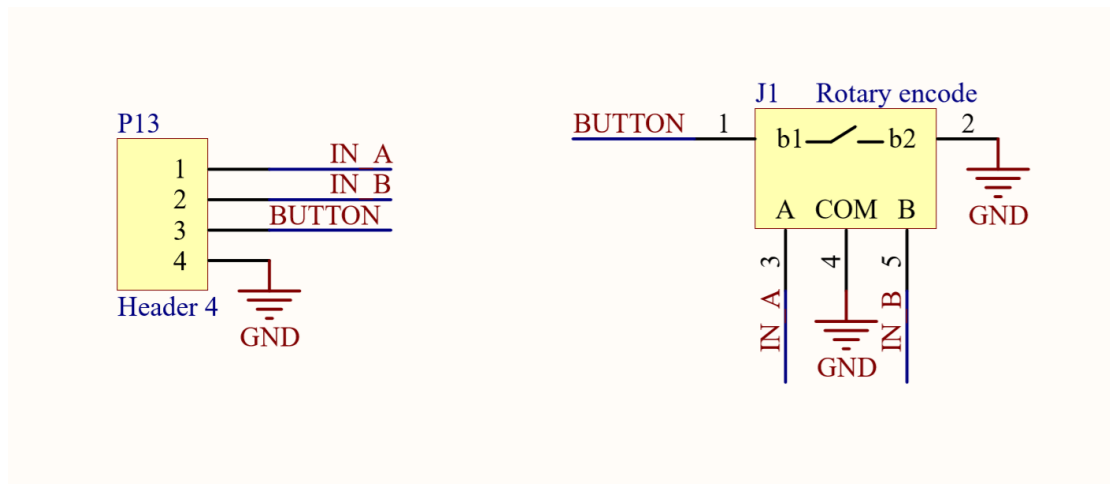
void loop() {
  //put your main code here, to run repeatedly:
  char key = keypad.getKey();
  if(key){
    Serial.println(key);
    M5.Lcd.fillRect(150, 150, 80, 50, BLACK);
    M5.Lcd.setCursor(150, 140, 4);
    M5.Lcd.printf("%c", 'A');
  }
}

```


Encoder

Description

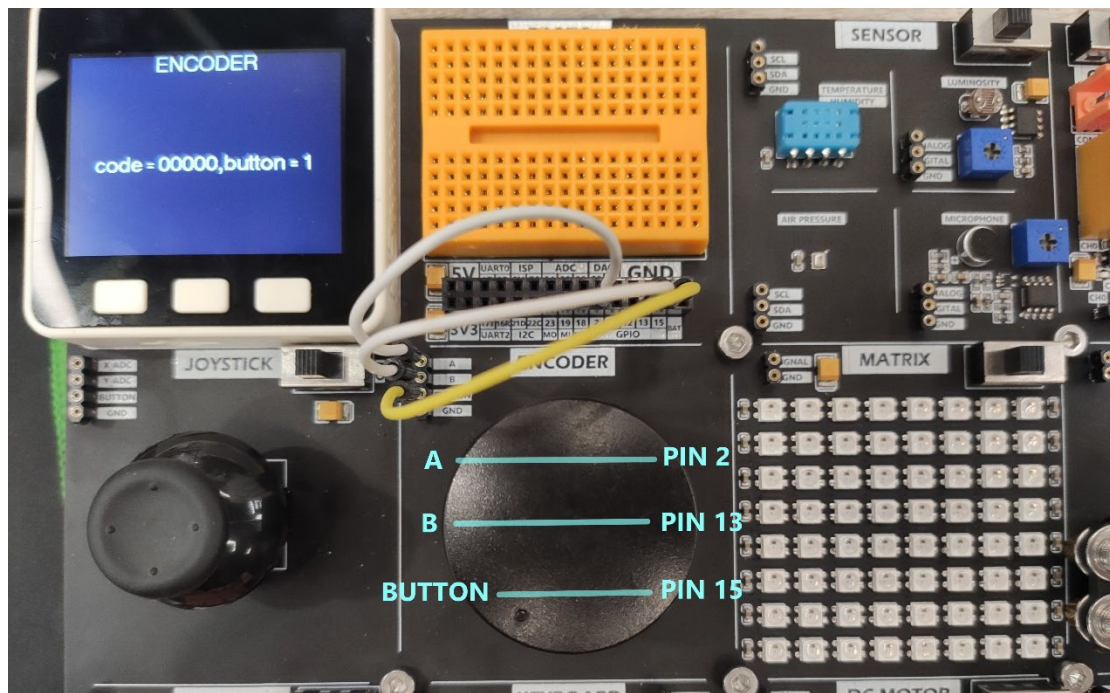
When the knob is rotated, port A and port B will generate a correspondingly high level according to the selected rotation direction. When the knob is pressed, its internal key switch will act accordingly to change the output signal to a low level.



Hardware connection

Connect the control spindle to an ordinary GPIO, and re-insert the replacement change in the program to determine the action of the

knob.



Example

```
#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);
    dacWrite(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);

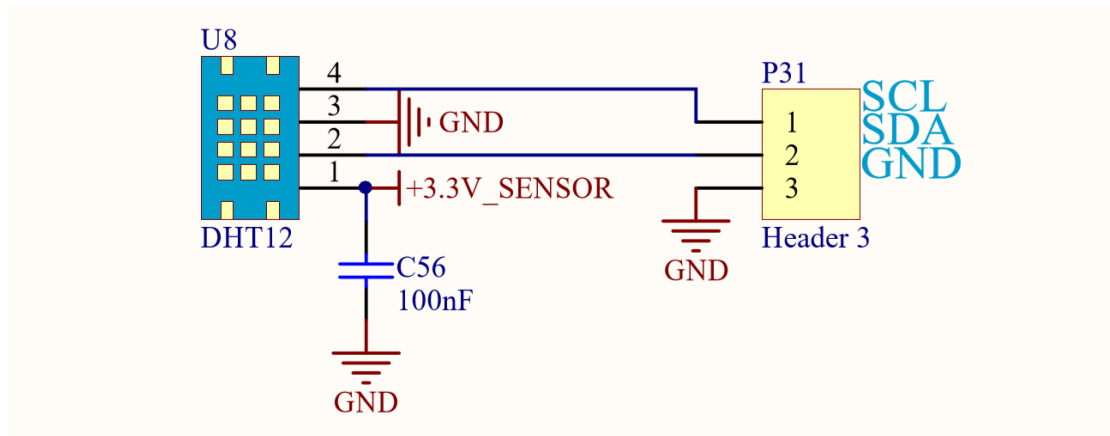
    M5.Lcd.setCursor(30, 120, 4);
    M5.Lcd.printf("code = %05d,button = %d\r\n ",count_change,value);
    delay(1);
}

```

DHT12 Temperature and humidity detection

Description

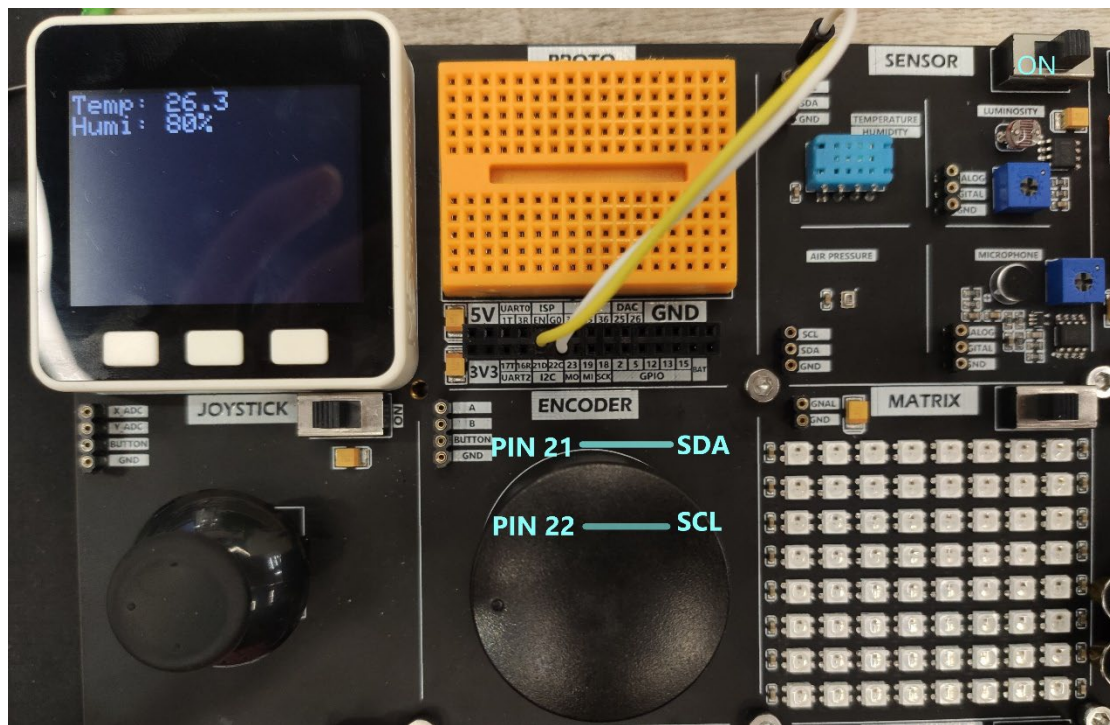
The DHT12 temperature and humidity sensor located in the sensor module group can collect temperature and humidity data in the environment, and transmit data through the I2C protocol. The I2C address is (0x5c)



Hardware connection

Sensor devices that use I2C protocol for communication can be connected to M5Core's default I2C protocol pins PIN21 (SDA), PIN22

(SCL) when in use



Example

The following code is only the main program and does not include its dependent library files such as `<DHT12.h>`. For the complete code, please visit the Github address below to get it.

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

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 / SENSOR / TEMP_HUM / Go to file Add file

zhouyangyale secd update 4071aa9 on 3 Jun 2019

..		
DHT12.cpp	secd update	16 months ago
DHT12.h	secd update	16 months ago
TEMP_HUM.ino	secd update	16 months ago

Read the temperature and humidity values measured by the DHT12 sensor and display them on the screen.

```
#include <M5Stack.h>
```

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

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() {

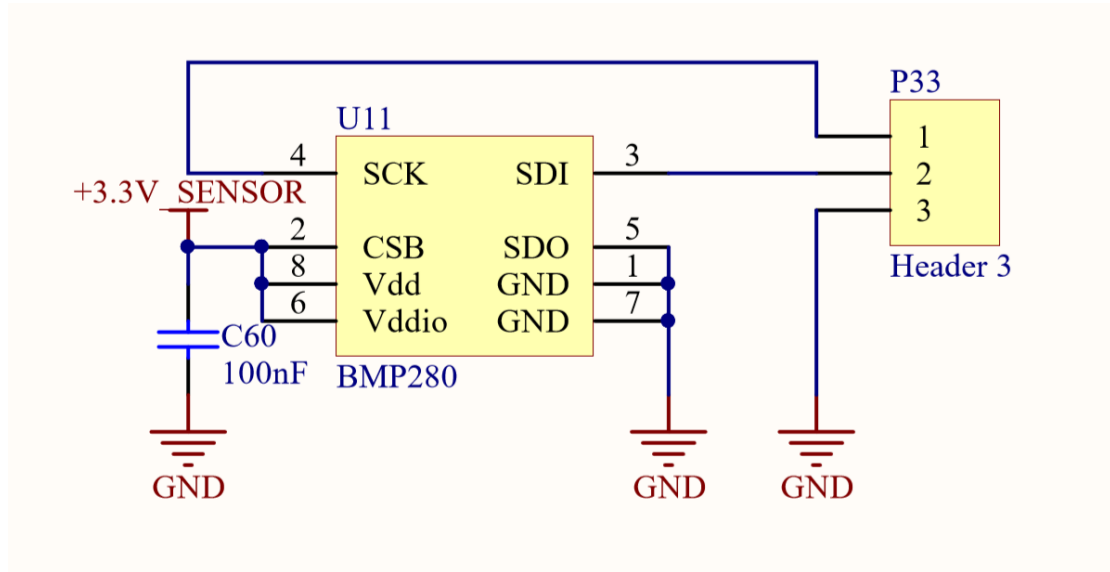
  float tmp = dht12.readTemperature();
  float hum = dht12.readHumidity();
  M5.Lcd.setCursor(30, 100, 4);
  M5.Lcd.printf("Temp: %2.1f Humi: %2.0f%%", tmp, hum);

  delay(100);
}
```

BMP280 Air pressure detection

Description

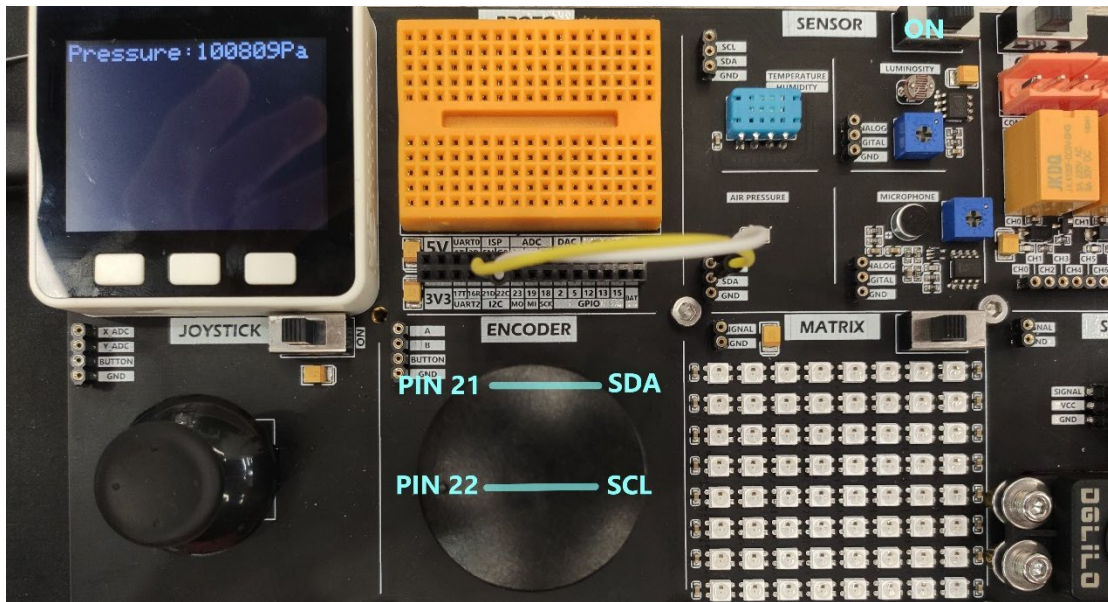
The BMP280 barometric pressure sensor located in the sensor module group can collect the barometric data of the current position and transmit data through the I2C protocol. The I2C address is (0x76)



Hardware connection

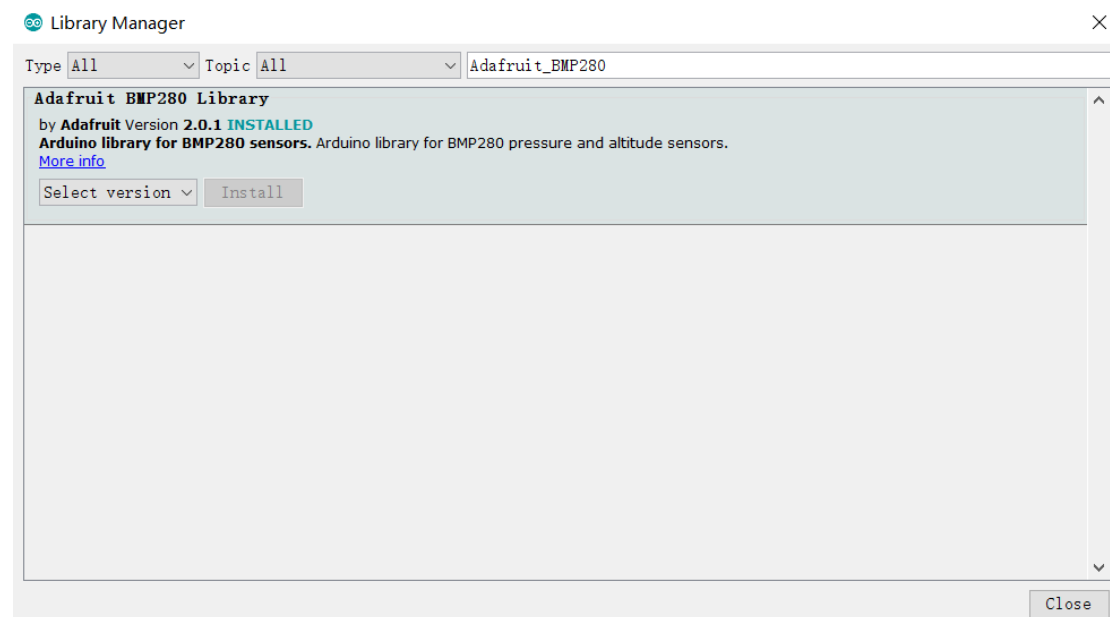
Sensor devices that use I2C protocol for communication can be connected to M5Core's default I2C protocol pins PIN21 (SDA), PIN22

(SCL) when in use



Example

In this case, the library `<Adafruit_BMP280.h>` is used, you can search and install it in the library management of Arduino.



Read the atmospheric pressure value measured by the BMP280 sensor and display it on the screen.

```
#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");

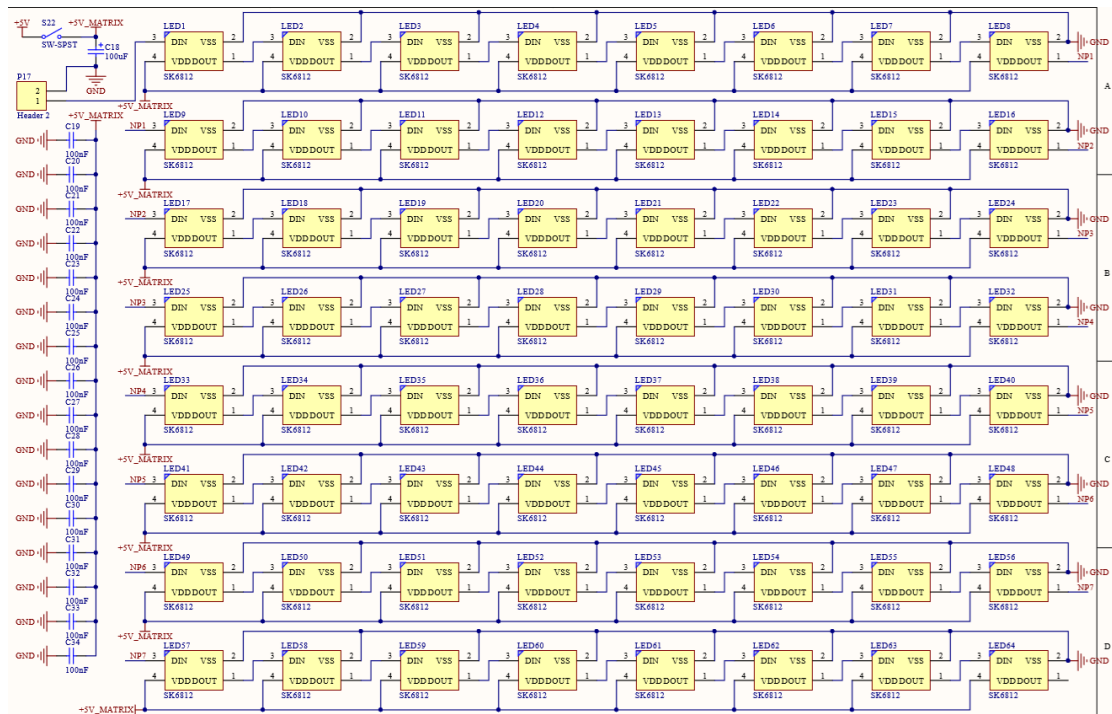
  if (!bme.begin(0x76)){
    Serial.println("Could not find a valid BMP280 sensor, check wiring!");
    while (1);
  }
}

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

LED MATRIX

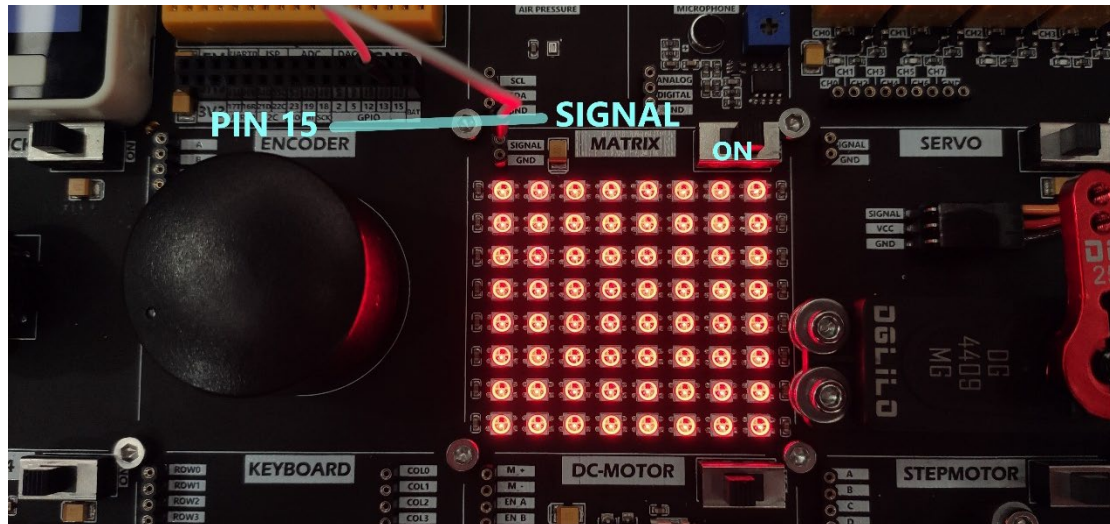
Description

MATRIX is an 8x8 LED matrix. Through programming, it can control any LED light on the matrix to emit light, and adjust the color, brightness, and other attributes.



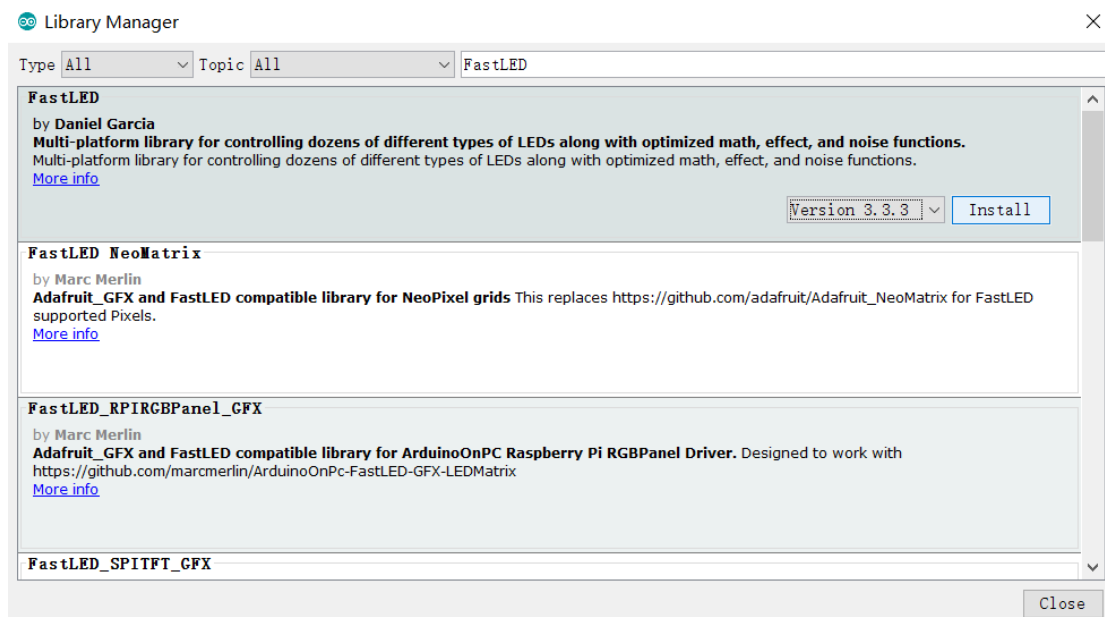
Hardware connection

Connect the control pin to the SIGNAL interface of the matrix and turn on the independent power switch.



Example

The library `<FastLED.h>` is used in this case, you can search and install it in the library management of Arduino.



Drive the LED matrix to perform the water lamp effect

```

#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_LEDS).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);
}

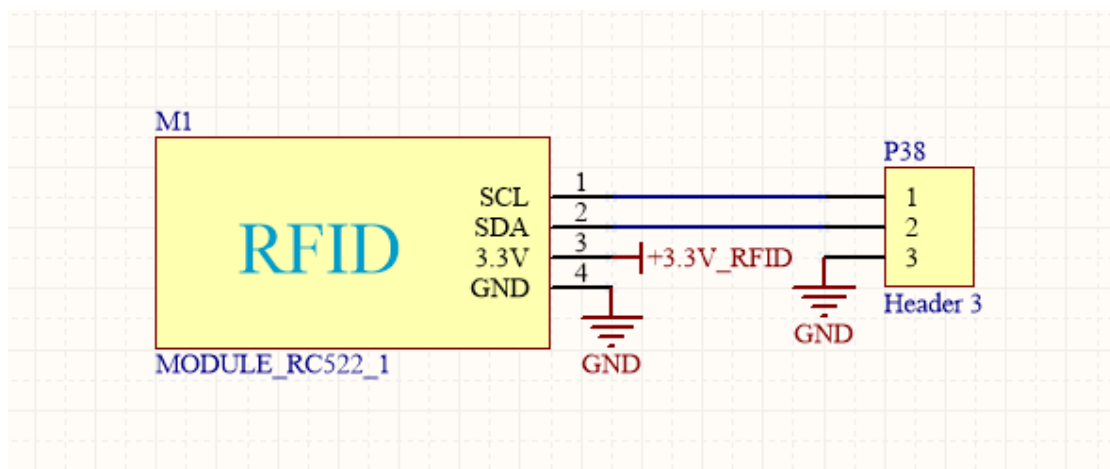
```

RFID

Description

The RFID module integrates an RC522 radio frequency identification chip and uses the I2C protocol for data transmission. The I2C address is (0x28)

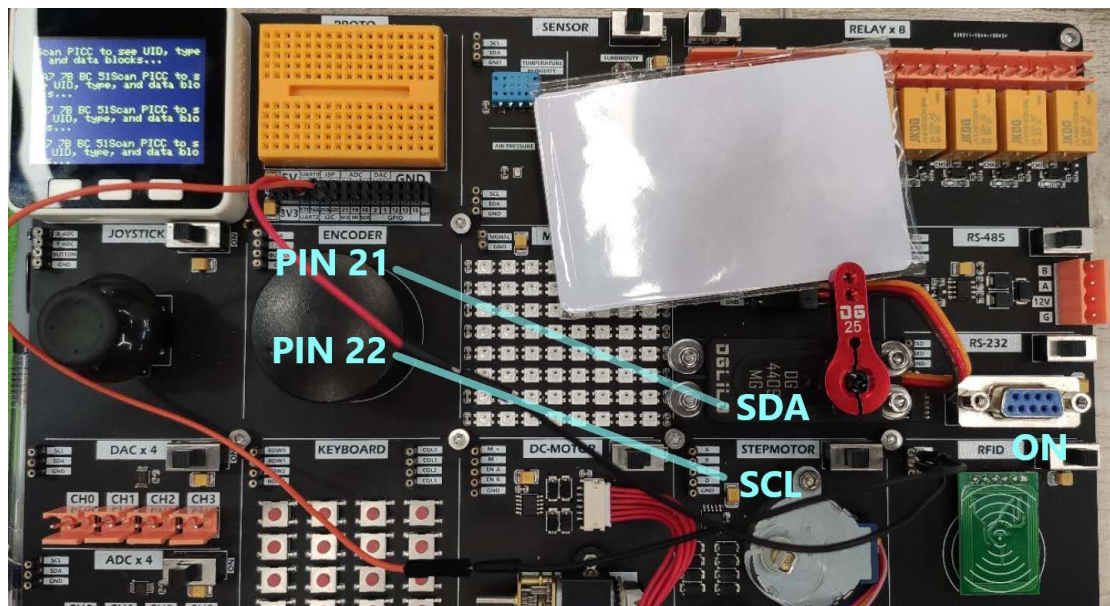
The working frequency is 13.56MHz. It supports multiple functions such as card reading, card writing, identification, recording, encoding, and authorization of RF cards.



Hardware connection

Sensor devices that use I2C protocol for communication can be connected to M5Core's default I2C protocol pins PIN21 (SDA), PIN22

(SCL).



Example

The following code is only the main program and does not include its dependent library files such as `<MFRC522_I2C.h>`. For the complete code, please visit the Github address below to obtain it., <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 second update 4071aa9 on 3 Jun 2019

MFRC522_I2C.cpp	update	16 months a
MFRC522_I2C.h	update	16 months a
RFID.ino	second update	16 months a

Read the ID and software version of the IC card

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

```

// 0x28 is i2c address on SDA. Check your address with i2cscanner 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();
}

```

```

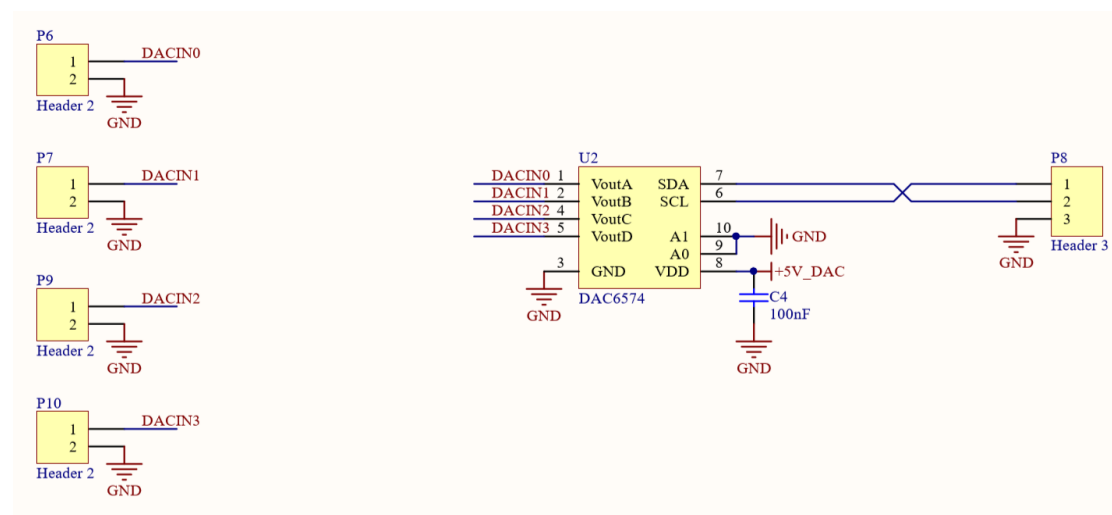
void ShowReaderDetails() {
  // Get the MFRC522 software version
  byte v = mfrc522.PCD_ReadRegister(mfrc522.VersionReg);
  Serial.print(F("MFRC522 Software Version: 0x"));
  Serial.print(v, HEX);
  if (v == 0x91)
    Serial.print(F(" = v1.0"));
  else if (v == 0x92)
    Serial.print(F(" = v2.0"));
  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

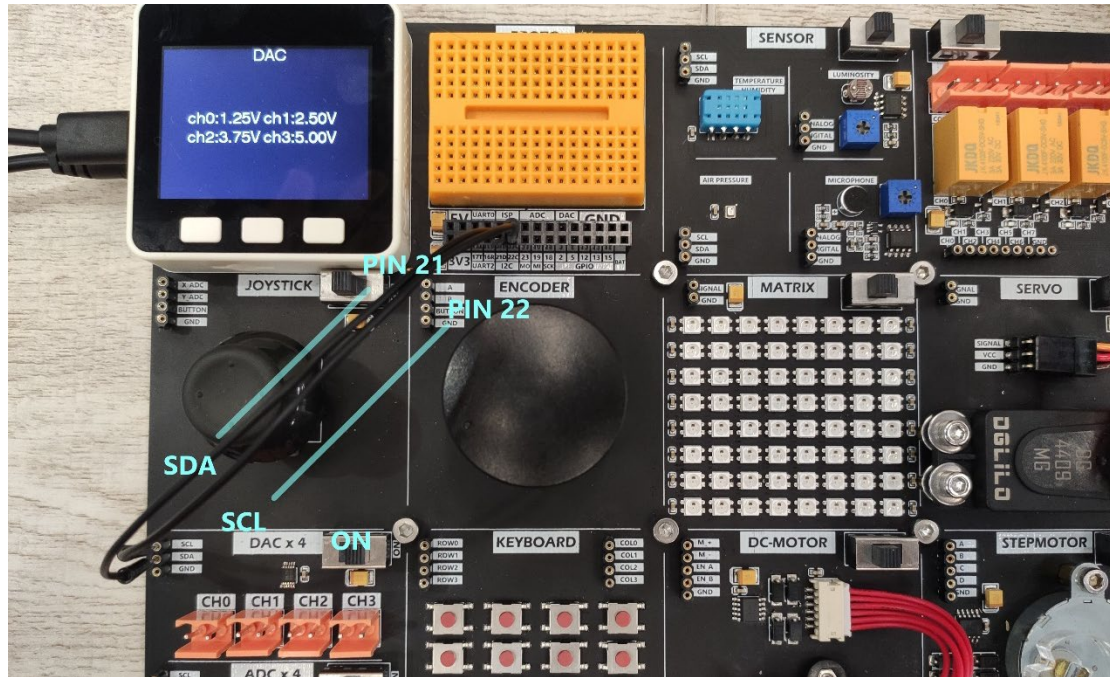
Description

The development board provides 4 DAC conversion interfaces, which means that you can program the control channel to output different voltages. Use I2C protocol for control, and the communication address is (0x28)



Hardware connection

Sensor devices that use the I2C protocol for communication can be connected to M5Core's default I2C protocol pins PIN21 (SDA), PIN22 (SCL).



Example

Drive 4 DAC channels to output voltage 1.25V, 2.50V, 3.75V, 5.00V respectively

```
#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

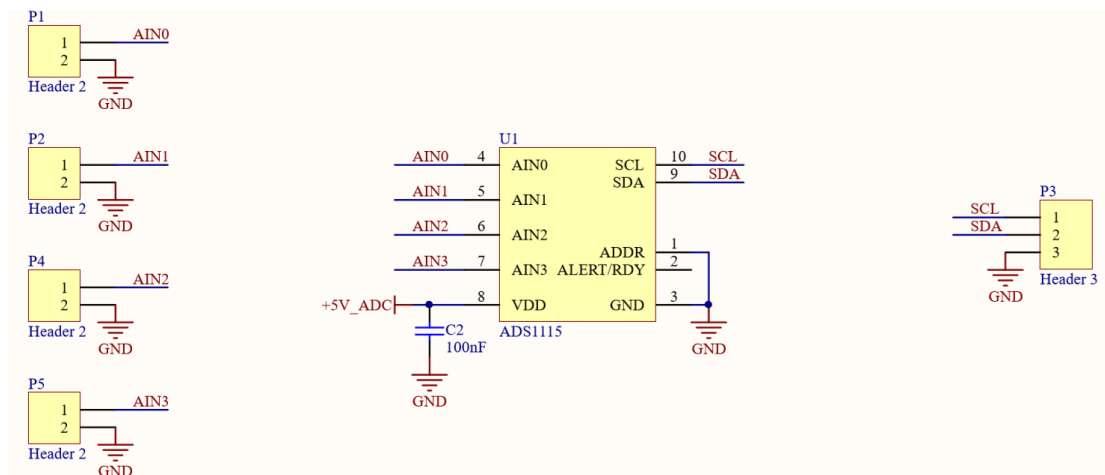
M5.Lcd.setCursor(40, 100, 4);
M5.Lcd.println("ch0:1.25V ch1:2.50V");
M5.Lcd.setCursor(40, 130, 4);
M5.Lcd.println("ch2:3.75V ch3:5.00V");
}

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

ADC

Description

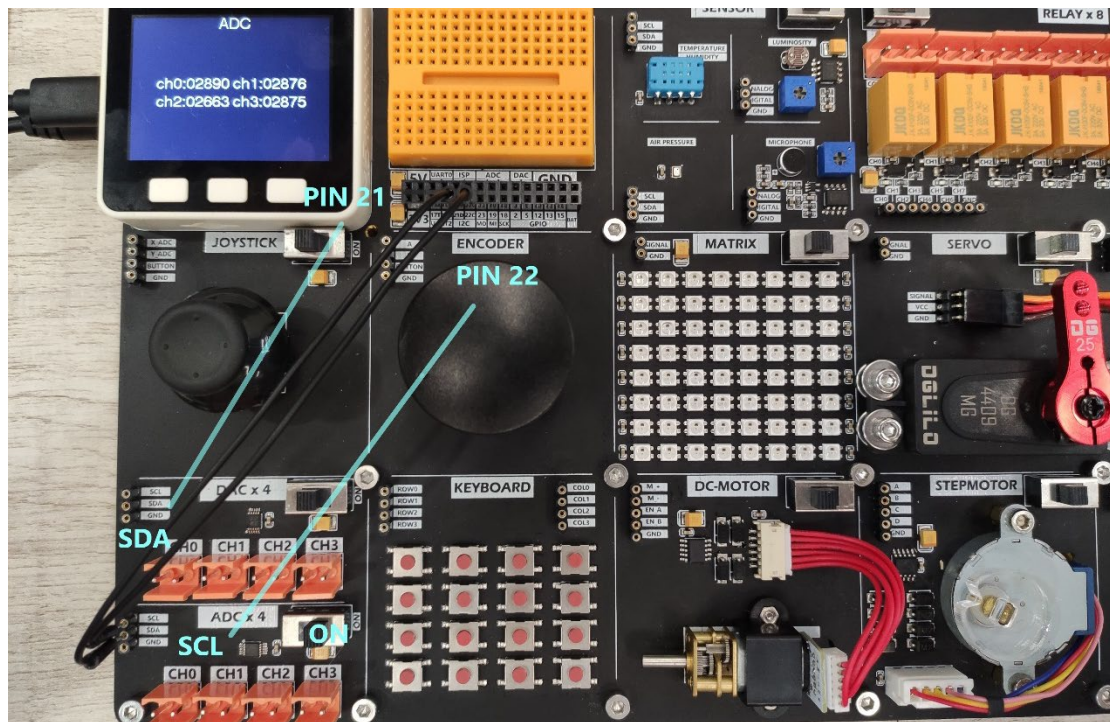
The development board provides 4 ADC conversion interfaces, which means that you can input some analog signals and convert them into digital signals for analysis and calculation. Use I2C protocol for control, and the communication address is (0x48)



Hardware connection

Sensor devices that use the I2C protocol for communication can be connected to M5Core's default I2C protocol pins PIN21 (SDA), PIN22

(SCL)



Example

Read the analog signals input by the 4 ADC conversion interfaces, and convert them into digital signals and display them on the screen.

```
#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);

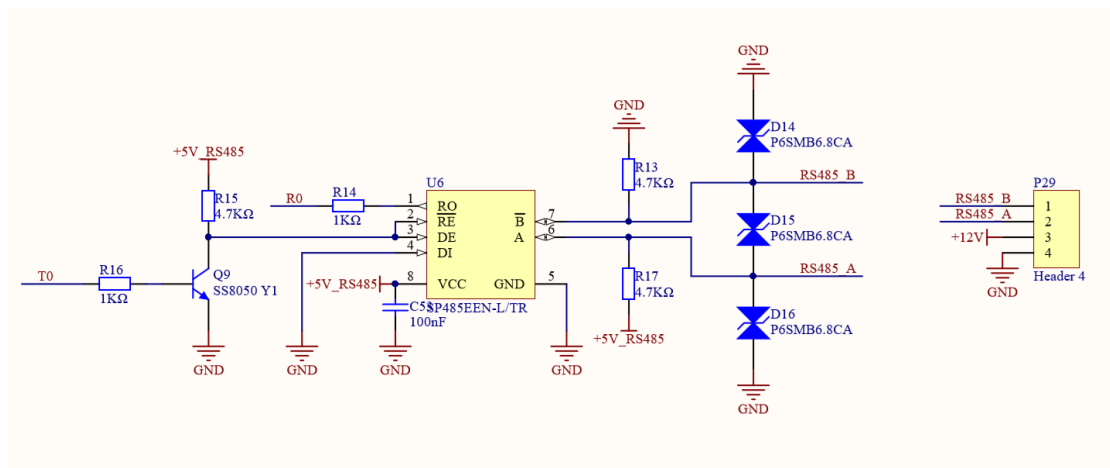
    M5.Lcd.setCursor(40, 100, 4);
    M5.Lcd.printf("ch0:%05d ch1:%05d\n", adc_ch0, adc_ch1);
    M5.Lcd.setCursor(40, 130, 4);
    M5.Lcd.printf("ch2:%05d ch3:%05d\n", adc_ch2, adc_ch3);
    delay(500);
}

```

RS-485

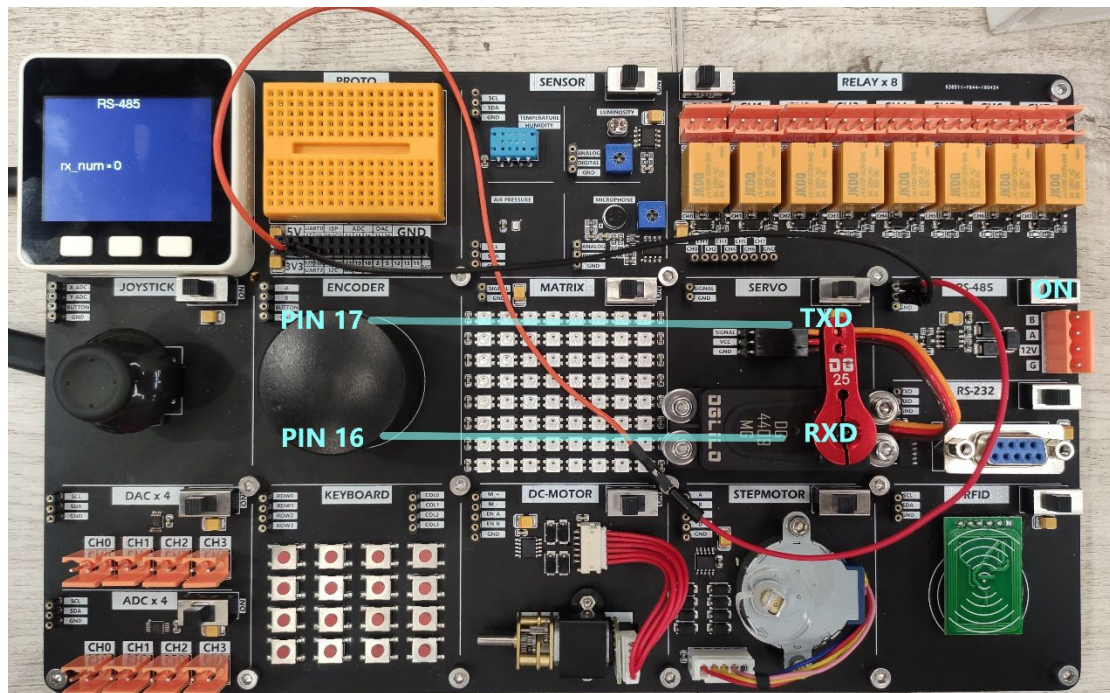
Description

RS-485 is a very common electrical characteristic standard in industrial control scenarios. The differential signal used in its communication can effectively resist electronic noise interference. The RS-485 conversion module on the Demoboard can convert ordinary TTL level signals into RS485 level signals, realize the conversion of the protocol, and then control the corresponding type of equipment



Hardware connection

Devices that use serial ports for communication can be connected to M5Core's default serial pins PIN17 (TXD), PIN16 (RXD),|



Example

Receive content from Serial2 and forward to Serial0 (USB), receive content from Serial0 (USB) and forward to 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() {
```

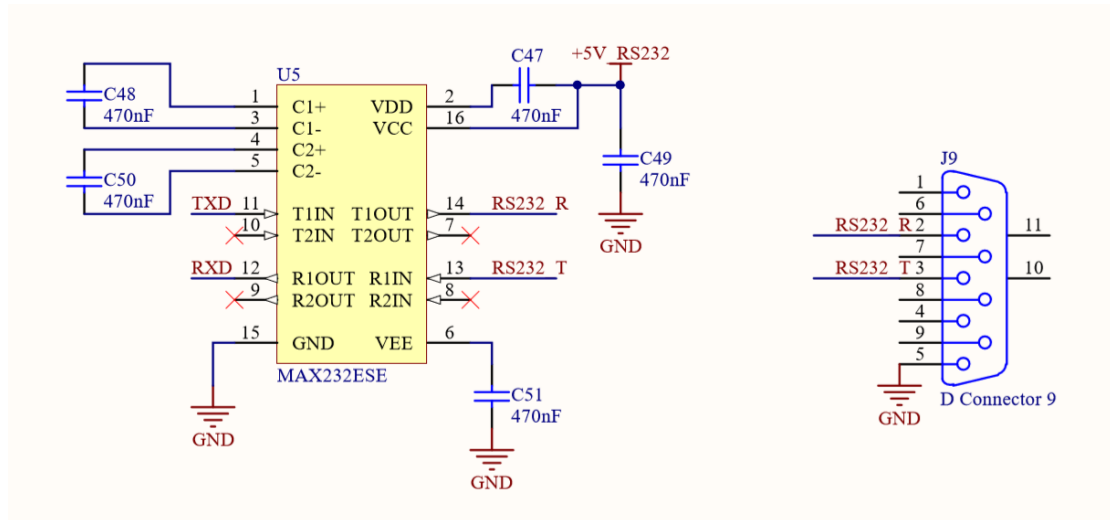
```
    if(Serial.available()) {
        int ch = Serial.read();
        Serial2.write(ch);
    }
```

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


RS-232

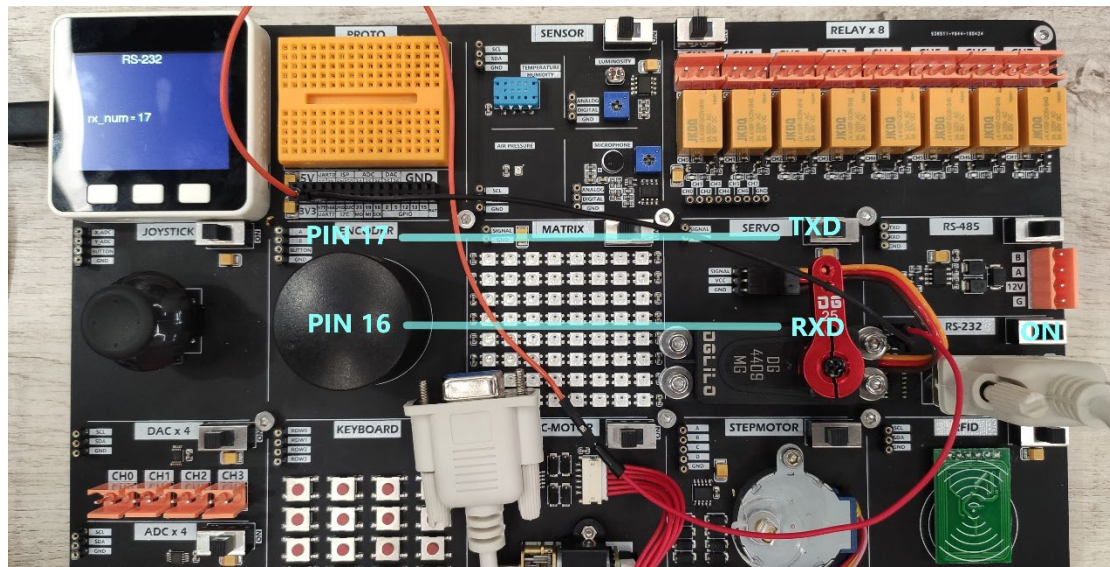
Description

The RS-232 conversion module can convert TTL level signals to RS232 level signals, realize protocol conversion, and then control corresponding types of equipment.



Hardware connection

Devices that use serial ports for communication can be connected to M5Core's default serial pins PIN17 (TXD), PIN16 (RXD).



Example

Receive content from Serial2 and forward to Serial0 (USB), receive content from Serial0 (USB) and forward to 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);
  }
}
```

APPENDIX

Example Github

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

The screenshot displays the GitHub interface for the repository `m5stack/DEMO-BOARD`. At the top, there is a search bar and navigation links for Pulls, Issues, Marketplace, and Explore. Below the repository name, there are buttons for Watch (2), Star (0), and Fork. The main navigation bar includes Code, Issues, Pull requests, Actions, Projects, Wiki, Security, and Insights. The file browser shows the `master` branch with a list of files and folders, including `.vscode`, `ADC`, `DAC`, `DC-Motor`, `ENCODER`, and `FactoryTest`. A `Code` dropdown menu is open, providing options to clone the repository with HTTPS or SSH, open it with GitHub Desktop, or download it as a ZIP file. The right sidebar contains sections for 'About' (with a note that no description or website is provided), 'Releases' (with a note that no releases are published), and 'Packages'.

Arduino API

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

Product List Platform|API Cases FAQ Language Search

M5Core API

System Speaker LCD Button IMU Sensor(MPU9250) TF Card

Power I/O I2C WIFI Timer

M5StickC API

System AXP192 TFT-SCREEN IMU RTC PWM

M5Core2 API

Document & Datasheet

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

M5Stack Docs Search Product List Platform|API Cases FAQ Language

Module parameters

Module Name	working Voltage	Parameter
ADC	5V	4x ADC port/ADS1115
DAC	5V	4x DAC port/DAC6574
Joystick	3.3V	axis-X/Y potentiometer input, axis-Z button input
DHT12	3.3V	I2C address 0x5C
BMP280	3.3V	I2C address 0x76
Light	3.3V	A/D sampling supported, adjustable threshold
Microphone	3.3V	A/D sampling supported, adjustable threshold
Relay	5V	8 channels /3A-220V-AC/3A-30V-DC
RGB LED	5V	8x8 LED matrix
Servo	5V	10KG torsion