Overview

Provide Pico C demo.

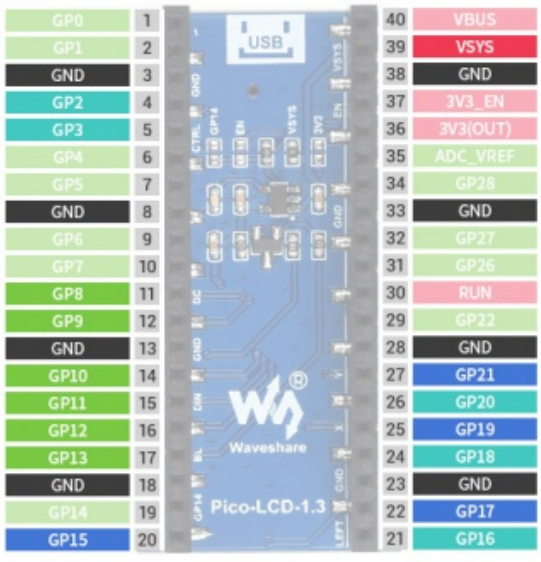
Specifications

Pico LCD 1.3

1.3inch LCD Display Module for Raspberry Pi Pico, 65K Colors, 240 × 240, SPI

Item	Parameter
Supply Voltage	2.6V ~ 5.5V
Operating Current	40mA
Screen Type	IPS
Controller	ST7789VW
Communication Interface	4-wire SPI
Resolution	240(H)RGB x 240(V) Pixels
Pixel Size	0.0975 (H) x 0.0975 (V) mm
Display Size	23.4 (H) x 23.4 (V) mm
Dimensions	26.5 (H) x52.00 (V) mm

Pinout



VSYS		Power supply 1.8V~5.5V
GND		Ground
GP8	LCD_DC	Data/Command, High for Data, Low for Command
GP9	LCD_CS	Chip select, low active
GP10	LCD_CLK	SPI clock input
GP11	LCD_DIN	SPI data input
GP12	LCD_RST	Reset, low active
GP13	LCD_BL	Backlight

GP15	A	User key A
GP2	UP	Joystick up
GP18	DOWM	Joystick down

GP19	X	User key X	GP20	RIGHT	Joystick right
GP21	Y	User key Y	GP3	CTRL	Joystick press center

Dimension

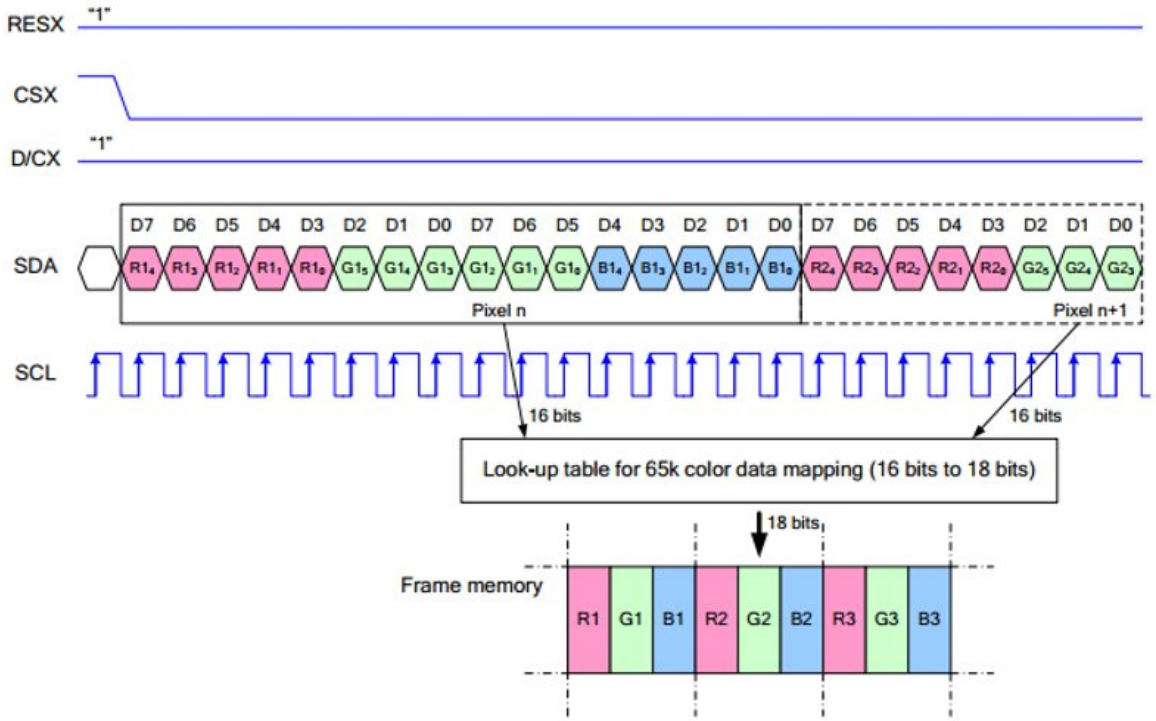
LCD and the controller

The ST7789VW is a single-chip controller/driver for 262K-color, graphic type TFT-LCD. It consists of 240 source line and 320 gate line driving circuits. The resolution of this LCD is 240 (H) RGB x 240 (V), it supports horizontal mode and vertical mode, and it doesn't use all the RAM of the controller.

This LCD accepts 8-bits/9-bits/16-bits/18-bits parallel interface, that are RGB444, RGB565, RGB666. The color format used in demo codes is RGB565.

This LCD uses a 4-line SPI interface for reducing GPIO and fast speed.

Working Protocol



Note: Different from the traditional SPI protocol, the data line from the slave to the master is hidden since the device only has a display requirement.

RESX Is the reset pin, it should be low when powering the module and be higher at other times;

CSX is slave chip select, when CS is low, the chip is enabled.

D/CX is data/command control pin, when DC = 0, write command, when DC = 1, write data

SDA is the data pin for transmitting RGB data, it works as the MOSI pin of SPI interface;
SCL work s the SCLK pins of SPI interface.
SPI communication has data transfer timing, which is combined by CPHA and CPOL.
CPOL determines the level of the serial synchronous clock at an idle state. When CPOL = 0, the level is Low. However, CPOL has little effect on the transmission.
CPHA determines whether data is collected at the first clock edge or at the second clock edge of the serial synchronous clock; when CPHL = 0, data is collected at the first clock edge.
There are 4 SPI communication modes. SPI0 is commonly used, in which CPHL = 0, CPOL = 0.

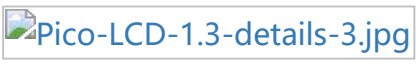
Pico LCD 1.3 Connection & Demo

Hardware Connection

Please take care of the direction when you connect Pico, a USB port is printed to indicate. You can also check the pin of Pico and the LCD board when connecting.
You can connect the display according to the table.

e-Paper	Pico	Description
VCC	VSYS	Power Input
GND	GND	GND
DIN	GP11	MOSI pin of SPI, slave device data input
CLK	GP10	SCK pin of SPI, clock pin
CS	GP9	Chip selection of SPI, low active
DC	GP8	Data/Command control pin (High for data; Low for command)
RST	GP12	Reset pin, low active
BL	GP13	Backlight control
A	GP15	User button A
B	GP17	User button B
X	GP19	User button X
Y	GP21	User buttonY
UP	GP2	Joystick-up
DOWM	GP18	Joystick-down
LEFT	GP16	Joystick-left
RIGHT	GP20	Joystick-right
CTRL	GP3	Joystick-center

Connection (Directly)



Connection (with adapter board)

Setup environment

Please refer to Raspberry Pi's guide:
https://www.raspberrypi.org/documentation/pico/getting-started/

Download Demo codes

Open terminal and run the following command:

```
sudo apt-get install p7zip-full
cd ~
sudo wget  https://files.waveshare.com/upload/2/28/Pico_code.7z
7z x Pico_code.7z -o./Pico_code
cd ~/Pico_code
cd c/build/
```

Run the Demo codes

This guides is based on Raspberry Pi.

C Examples

Open a terminal and enter the directory of C codes:

```
cd ~/Pico_LCD_code/c/
```

Create a build folder and add SDK:
For example, if the path of SDK is ../../pico-sdk
Then you should create build and add the path like these:

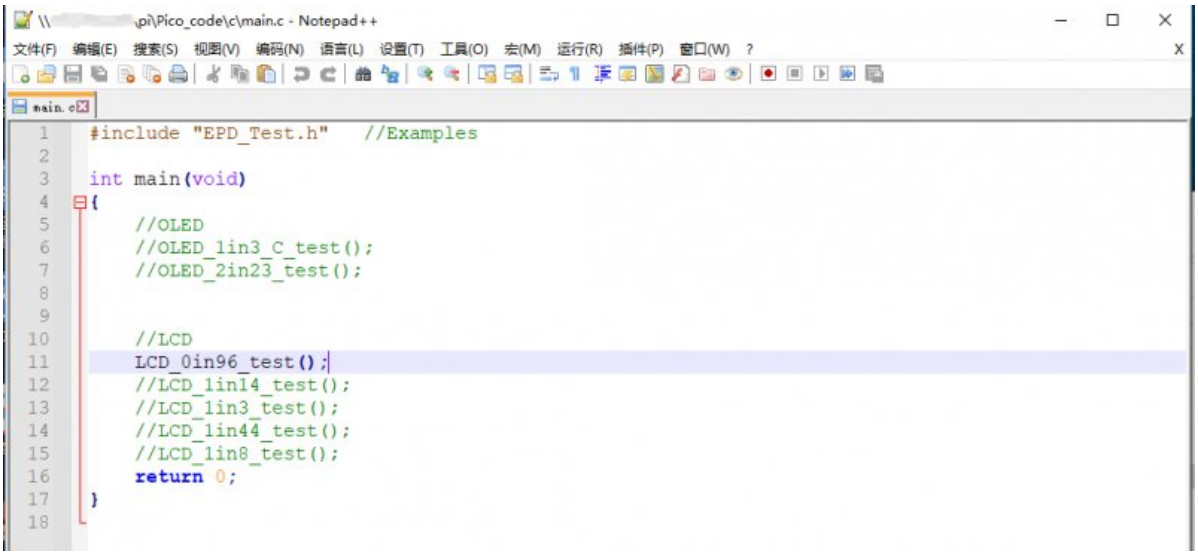
```
cd build
export PICO_SDK_PATH=... /... /.../pico-sdk
(Note: Be sure to write the right path for your own SDK)
```

Run cmake.. command to to generate Makefile file

```
cmake ..
```

Open main.c under the c folder, you can change the routine you need. This routine can drive the display of our company's Pico series and the source code will be

updated all the time. Please select the corresponding LCD or OLED test function and comment out the irrelevant functions.



Run make command to build.

```
make -j9
```

After the compilation is complete, the uf2 file will be generated. Press and hold the button on the Pico board, connect the pico to the USB port of the Raspberry Pi through the Micro USB cable, and release the button. After connecting, the Raspberry Pi will automatically recognize a removable disk (RPI-RP2), and copy the main.uf2 file in the build folder to the recognized removable disk (RPI-RP2).

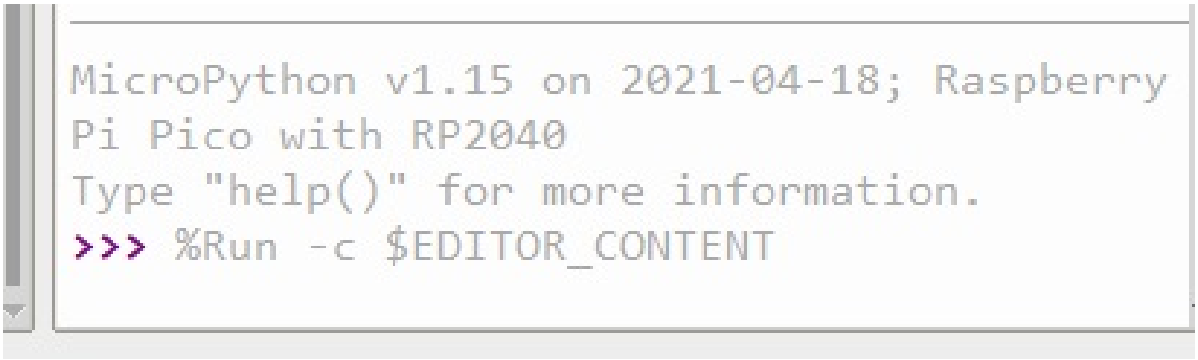
```
cp main.uf2 /media/pi/RPI-RP2/
```

Python codes

Use in Windows

- 1. Press and hold the BOOTSET button on the Pico board, connect the pico to the USB port of the computer through the Micro USB cable, and release the button after the computer recognizes a removable hard disk (RPI-RP2).
- 2. Copy the rp2-pico-20210418-v1.15.uf2 file in the python directory to the recognized removable disk (RPI-RP2).
- 3. Open Thonny IDE (Note: Use the latest version of Thonny, otherwise there is no Pico support package, the latest version under Windows is v3.3.3).
- 4. Click Tools->Settings->Interpreter, select Pico and the corresponding port as shown in the figure.

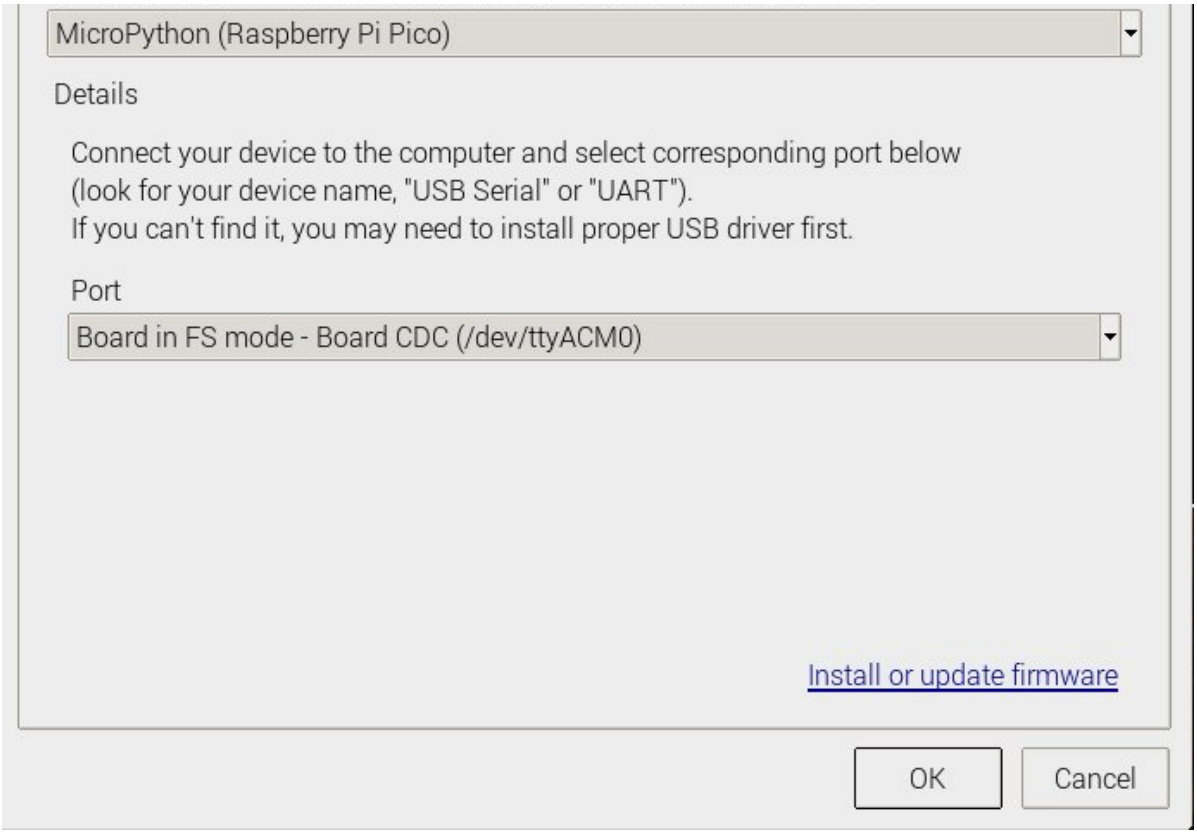
- 5. File -> Open -> the corresponding .py file, click to run, as shown in the following figure:



This demo provides a simple program...

Run in Raspberry Pi

- Hold the BOOTSET key of the Pico board, then connect the Pico to Raspberry Pi by USB cable, then release the key.
- Once the removable disk (RPI-RPI2) is recognized, copy the rp2-pico-20210418-v1.15.uf2 file to pico.
- Open the Thonny IDE in Raspberry Pi, update it if it doesn't support Pico
- Configure the port by choosing MicroPython (Raspberry Pi and ttyACM0 port) in Tools -> Options... -> Interpreter



If your Thonny doesn't support Pico, you can update it with the following command:

```
sudo apt upgrade thonny
```

- Choose File->Open...->python/ and select the corresponding .py file to run the codes

Codes Analysis

C

Bottom hardware interface

We package the hardware layer for easily porting to the different hardware platforms.
DEV_Config.c(.h) in the directory:...\c\lib\Config

- Data type:

```
#define UBYTE   uint8_t
#define UWORD   uint16_t
#define UDOUBLE uint32_t
```

- Module initialize and exit:

```
void DEV_Module_Init(void);
void DEV_Module_Exit(void);
Note:
1.The functions above are used to initialize the display or exit handle.
```

- GPIO write/read:

```
void    DEV_Digital_Write(UWORD Pin, UBYTE Value);
UBYTE   DEV_Digital_Read(UWORD Pin);
```

- SPI transmit data

```
void DEV_SPI_WriteByte(UBYTE Value);
```

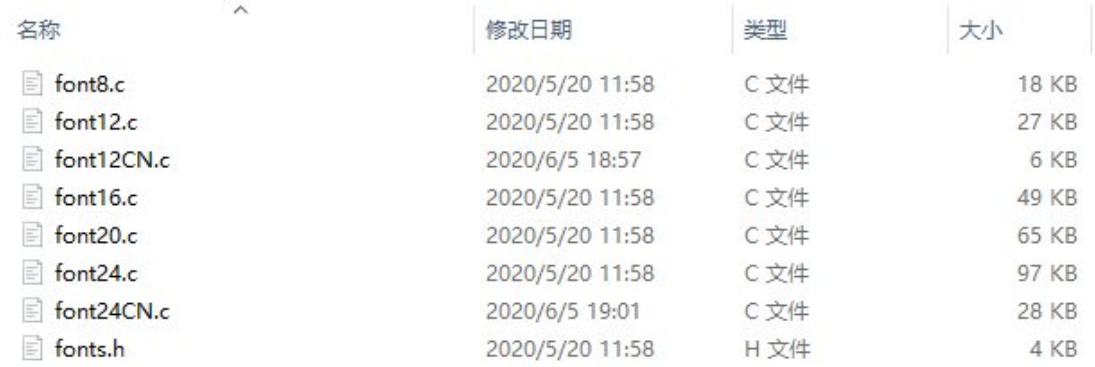
Application functions

We provide basic GUI functions for testing, like draw point, line, string, and so on.

The GUI function can be found in directory:..\c\lib\GUI\GUI_Paint.c(.h)

GUI_Paint.c	2021/2/1 11:18	C 文件	32 KB
GUI_Paint.h	2021/2/1 11:17	H 文件	6 KB

The fonts used can be found in directory: RaspberryPi\c\lib\Fonts



名称	修改日期	类型	大小
font8.c	2020/5/20 11:58	C 文件	18 KB
font12.c	2020/5/20 11:58	C 文件	27 KB
font12CN.c	2020/6/5 18:57	C 文件	6 KB
font16.c	2020/5/20 11:58	C 文件	49 KB
font20.c	2020/5/20 11:58	C 文件	65 KB
font24.c	2020/5/20 11:58	C 文件	97 KB
font24CN.c	2020/6/5 19:01	C 文件	28 KB
fonts.h	2020/5/20 11:58	H 文件	4 KB

- Create a new image, you can set the image name, width, height, rotate angle and color.

```
void Paint_NewImage(UWORD *image, UWORD Width, UWORD Height, UWORD Rotate, UWORD
Color, UWORD Depth)
Parameter:
        image : Name of the image buffer, this is a pointer;
        Width : Width of the image;
        Height: Height of the image;
        Rotate: Rotate angle of the Image;
        Color : The initial color of the image;
        Depth : Depth of the color
```

- Select image buffer: You can create multiple image buffers at the same time and

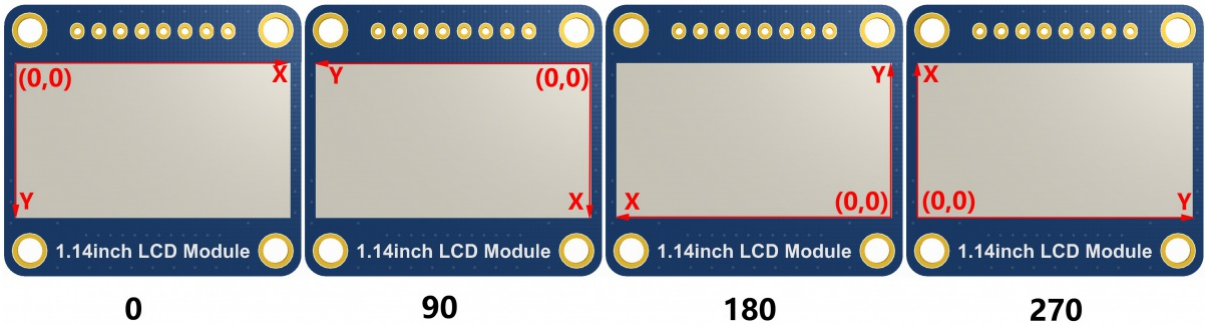
select the certain one and drawing by this function.

```
void Paint_SelectImage(UBYTE *image)
Parameter:
        image: The name of the image buffer, this is a pointer;
```

- Rotate image: You need to set the rotate angle of the image, this function should be used after Paint_SelectImage(). The angle can 0, 90, 180, 270

```
void Paint_SetRotate(UWORD Rotate)
Parameter:
        Rotate: Rotate angle of the image, the parameter can be ROTATE_0, ROTATE
_90, ROTATE_180, ROTATE_270.
```

【Note】Afer rotating, the place of the first pixel is different as below



- Image mirror: This function is used to set the image mirror.

```
void Paint_SetMirroring(UBYTE mirror)
Parameter:
        mirror: Mirror type if the image, the parameter can be MIRROR_NONE、MIRR
OR_HORIZONTAL、MIRROR_VERTICAL、MIRROR_ORIGIN.
```

- Set the position and color of pixels: This is the basic function of GUI, it is used to set the position and color of a pixel in the buffer.

```
void Paint_SetPixel(UWORD Xpoint, UWORD Ypoint, UWORD Color)
Parameter:
        Xpoint: The X-axis position of the point in the image buffer
        Ypoint: The Y-axis position of the point in the image buffer
        Color : The color of the point
```

- Color of the image: To set the color of the image, this function always be used to clear the display.

```
void Paint_Clear(UWORD Color)
Parameter:
        Color: The color of the image
```

- Color of the windows: This function is used to set the color of windows, it always used for updating partial areas like displaying a clock.

```
void Paint_ClearWindows(UWORD Xstart, UWORD Ystart, UWORD Xend, UWORD Yend, UWOR
D Color)
Parameter:
        Xstart: X-axis position of the start point.
        Ystart: Y-axis position of the start point.
        Xend: X-axis position of the end point.
        Yend: Y-axis position of the end point
        Color: Color of the windows.
```

- Draw point: Draw a point at the position （Xpoint, Ypoint） of image buffer, you can configure the color, size, and the style.

```
void Paint_DrawPoint(UWORD Xpoint, UWORD Ypoint, UWORD Color, DOT_PIXEL Dot_Pixe
l, DOT_STYLE Dot_Style)
Parameter:
        Xpoint: X-axis position of the point.
        Ypoint: Y-axis position of the point
        Color: Color of the point
        Dot_Pixel: Size of the point, 8 sizes are available.
                 typedef enum {
                        DOT_PIXEL_1X1  = 1,    // 1 x 1
                        DOT_PIXEL_2X2  ,               // 2 X 2
                        DOT_PIXEL_3X3  ,               // 3 X 3
                        DOT_PIXEL_4X4  ,               // 4 X 4
                        DOT_PIXEL_5X5  ,               // 5 X 5
                        DOT_PIXEL_6X6  ,               // 6 X 6
                        DOT_PIXEL_7X7  ,               // 7 X 7
                        DOT_PIXEL_8X8  ,               // 8 X 8
                } DOT_PIXEL;
        Dot_Style: Style of the point, it define the extednded mode of the poin
t.
                typedef enum {
                   DOT_FILL_AROUND  = 1,
                   DOT_FILL_RIGHTUP,
                } DOT_STYLE;
```

- Draw line: Draw a lin from (Xstart, Ystart) to (Xend, Yend) in image buffer, you can configure the color, width and the style.

```
void Paint_DrawLine(UWORD Xstart, UWORD Ystart, UWORD Xend, UWORD Yend, UWORD Co
lor, LINE_STYLE Line_Style , LINE_STYLE Line_Style)
Parameter:
        Xstart: Xstart of the line
        Ystart: Ystart of the line
        Xend: Xend of the line
```

```
        Yend: Yend of the line
        Color: Color of the line
        Line_width: Width of the line, 8 sizes are available.
                typedef enum {
                        DOT_PIXEL_1X1  = 1,    // 1 x 1
                        DOT_PIXEL_2X2  ,               // 2 X 2
                        DOT_PIXEL_3X3  ,               // 3 X 3
                        DOT_PIXEL_4X4  ,               // 4 X 4
                        DOT_PIXEL_5X5  ,               // 5 X 5
                        DOT_PIXEL_6X6  ,               // 6 X 6
                        DOT_PIXEL_7X7  ,               // 7 X 7
                        DOT_PIXEL_8X8  ,               // 8 X 8
                } DOT_PIXEL;
         Line_Style: Style of the line, Solid or Dotted.
                typedef enum {
                        LINE_STYLE_SOLID = 0,
                        LINE_STYLE_DOTTED,
                } LINE_STYLE;
```

- Draw rectangle: Draw a rectangle from (Xstart, Ystart) to (Xend, Yend) , you can configure the color, width, and style.

```
void Paint_DrawRectangle(UWORD Xstart, UWORD Ystart, UWORD Xend, UWORD Yend, UWO
RD Color, DOT_PIXEL Line_width, DRAW_FILL Draw_Fill)
Parameter:
        Xstart: Xstart of the rectangle.
        Ystart: Ystart of the rectangle.
        Xend: Xend of the rectangle.
        Yend: Yend of the rectangle.
        Color: Color of the rectangle
        Line_width: The width of the edges. 8 sizes are available.
                typedef enum {
                        DOT_PIXEL_1X1  = 1,    // 1 x 1
                        DOT_PIXEL_2X2  ,               // 2 X 2
                        DOT_PIXEL_3X3  ,               // 3 X 3
                        DOT_PIXEL_4X4  ,               // 4 X 4
                        DOT_PIXEL_5X5  ,               // 5 X 5
                        DOT_PIXEL_6X6  ,               // 6 X 6
                        DOT_PIXEL_7X7  ,               // 7 X 7
                        DOT_PIXEL_8X8  ,               // 8 X 8
                } DOT_PIXEL;
        Draw_Fill: Style of the rectangle, empty or filled.
                typedef enum {
                        DRAW_FILL_EMPTY = 0,
                        DRAW_FILL_FULL,
                } DRAW_FILL;
```

- Draw circle: Draw a circle in image buffer, use (X_Center Y_Center) as center and Radius as radius. You can configure the color, width of line and the style of circle.

```
void Paint_DrawCircle(UWORD X_Center, UWORD Y_Center, UWORD Radius, UWORD Color,
DOT_PIXEL Line_width, DRAW_FILL Draw_Fill)
Parameter:
        X_Center: X-axis of center
        Y_Center: Y-axis of center
        Radius：radius of circle
        Color:  Color of the circle
        Line_width: The width of arc, 8 sizes are available.
                typedef enum {
                        DOT_PIXEL_1X1  = 1,    // 1 x 1
                        DOT_PIXEL_2X2  ,               // 2 X 2
                        DOT_PIXEL_3X3  ,               // 3 X 3
                        DOT_PIXEL_4X4  ,               // 4 X 4
                        DOT_PIXEL_5X5  ,               // 5 X 5
                        DOT_PIXEL_6X6  ,               // 6 X 6
                        DOT_PIXEL_7X7  ,               // 7 X 7
                        DOT_PIXEL_8X8  ,               // 8 X 8
                } DOT_PIXEL;
        Draw_Fill: Style of the circle: empty or filled.
                typedef enum {
                        DRAW_FILL_EMPTY = 0,
                        DRAW_FILL_FULL,
                } DRAW_FILL;
```

- Show Ascii character: Show a characeter in (Xstart, Ystart) position, you can configure the font, foreground and the background.

```
void Paint_DrawChar(UWORD Xstart, UWORD Ystart, const char Ascii_Char, sFONT* Fo
nt, UWORD Color_Foreground, UWORD Color_Background)
Parameter:
        Xstart: Xstart of the character
        Ystart: Ystart of the character
        Ascii_Char：Ascii char
        Font: five fonts are avaialble：
                font8：5*8
                font12：7*12
                font16：11*16
                font20：14*20
                font24：17*24
        Color_Foreground: foreground color
        Color_Background: background color
```

- Draw string: Draw string at (Xstart Ystart) , you can configure the fonts, foreground and the background

```
void Paint_DrawString_EN(UWORD Xstart, UWORD Ystart, const char * pString, sFONT
* Font, UWORD Color_Foreground, UWORD Color_Background)
Parameter:
        Xstart: Xstart of the string
        Ystart: Ystart of the string
```

```
        pString: String
        Font: five fonts are available:
                font8: 5*8
                font12: 7*12
                font16: 11*16
                font20: 14*20
                font24: 17*24的
        Color_Foreground: foreground color
        Color_Background: background color
```

- Draw Chiness string: Draw Chinese string at (Xstart Ystart) of image buffer. You can configure fonts (GB2312), foreground and the background.

```
void Paint_DrawString_CN(UWORD Xstart, UWORD Ystart, const char * pString, cFONT
* font, UWORD Color_Foreground, UWORD Color_Background)
Parameter:
        Xstart: Xstart of string
        Ystart: Ystart of string
        pString: string
        Font: GB2312 fonts, two fonts are available
:
                font12CN: ascii 11*21，Chinese 16*21
                font24CN: ascii 24*41，Chinese 32*41
        Color_Foreground: Foreground color
        Color_Background: Background color
```

- Draw number: Draw numbers at (Xstart Ystart) of image buffer. You can select font, foreground and the background.

```
void Paint_DrawNum(UWORD Xpoint, UWORD Ypoint, double Nummber, sFONT* Font, UWOR
D Digit,UWORD Color_Foreground, UWORD Color_Background);
Parameter:
        Xstart: Left vertex X coordinate
        Ystart: Left vertex Y coordinate
        Nummber:  The displayed number is stored in a 32-bit int type, which can
be displayed up to 2147483647.
                font8: 5*8
                font12: 7*12
                font16: 11*16
                font20: 14*20
                font24: 17*24
        Digit: Display decimal places
        Color_Foreground: Foreground color
        Color_Background: Background color
```

- Display time: Display time at (Xstart Ystart) of image buffer, you can configure fonts, foreground and the background.

```
void Paint_DrawTime(UWORD Xstart, UWORD Ystart, PAINT_TIME *pTime, sFONT* Font,
```

```
UWORD Color_Background, UWORD Color_Foreground)
Parameter:
        Xstart: Xstart of time
        Ystart: Ystart of time
        pTime: Structure of time
        Font: Ascii font, five fonts are avaialble
                font8: 5*8
                font12: 7*12
                font16: 11*16
                font20: 14*20
                font24: 17*24
        Color_Foreground: Foreground
        Color_Background: Background
```

Resource

Document

- Schematic
- ST7789VW Datasheet

Examples

- Example demo
- ESP_S2_MicroPython Demo

Development Software

- Thonny Python IDE (Windows V3.3.3)
- Zimo221.7z
- Image2Lcd.7z

Pico Quick Start

Download Firmware

- MicroPython Firmware Download
- C_Blink Firmware Download [Expand]

Video Tutorial [Expand]

- Pico Tutorial I - Basic Introduction
- Pico Tutorial II - GPIO [Expand]
- Pico Tutorial III - PWM [Expand]
- Pico Tutorial IV - ADC [Expand]
- Pico Tutorial V - UART [Expand]
- Pico Tutorial VI - To be continued... [Expand]

MicroPython Series

- 【MicroPython】 machine.Pin Function
- 【MicroPython】 machine.PWM Function
- 【MicroPython】 machine.ADC Function
- 【MicroPython】 machine.UART Function
- 【MicroPython】 machine.I2C Function
- 【MicroPython】 machine.SPI Function
- 【MicroPython】 rp2.StateMachine

C/C++ Series

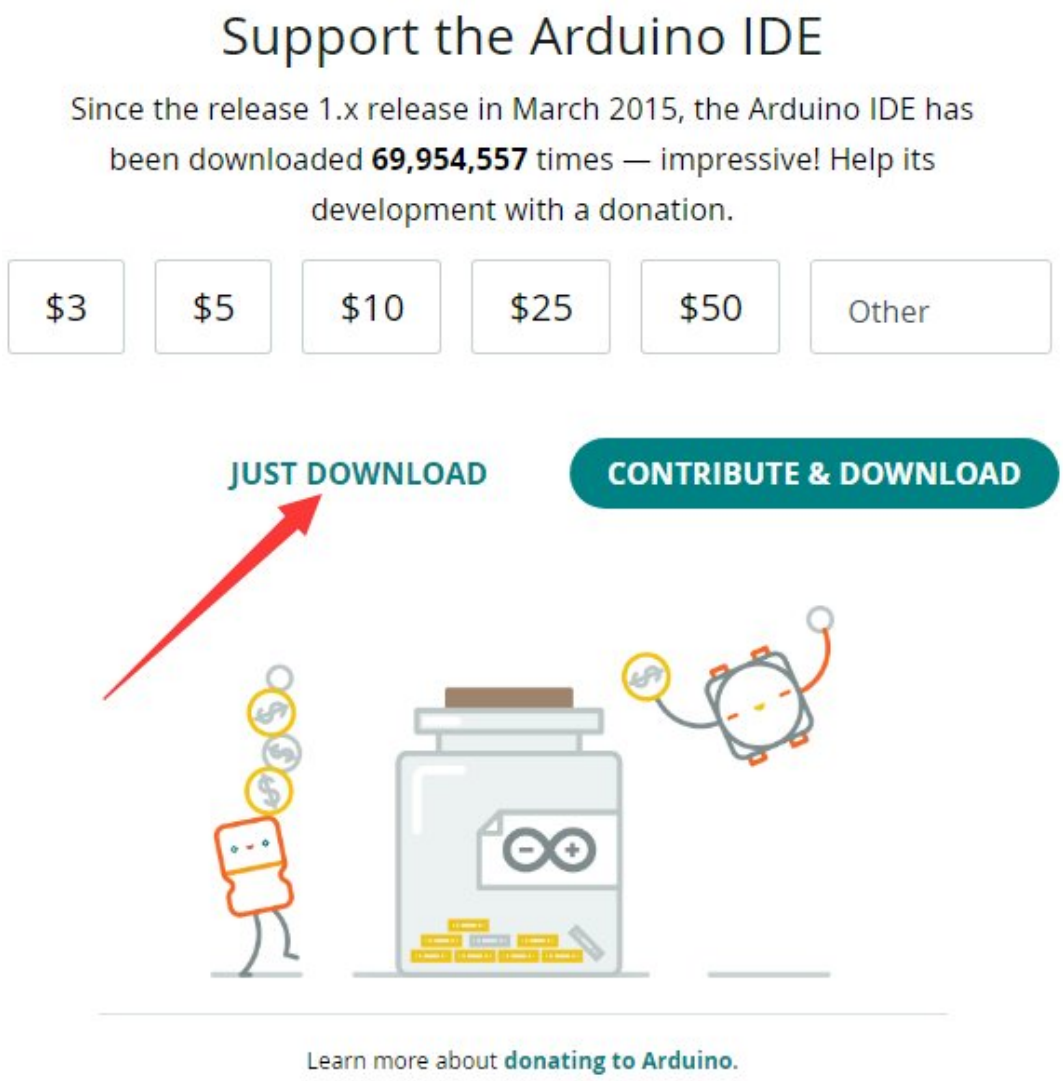
- 【C/C++】 Windows Tutorial 1 - Environment Setting
- 【C/C++】 Windows Tutorial 1 - Create New Project

Arduino IDE Series

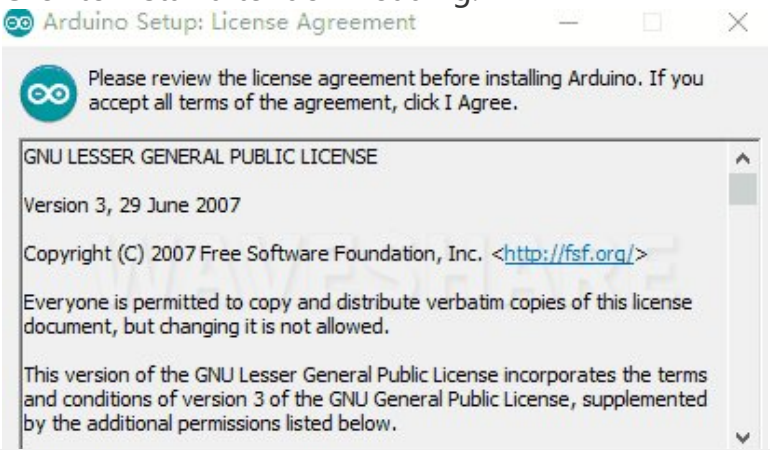
Install Arduino IDE

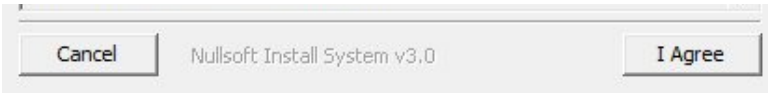
1. Download the Arduino IDE installation package from Arduino website.

2. Just click on "JUST DOWNLOAD".



3. Click to install after downloading.





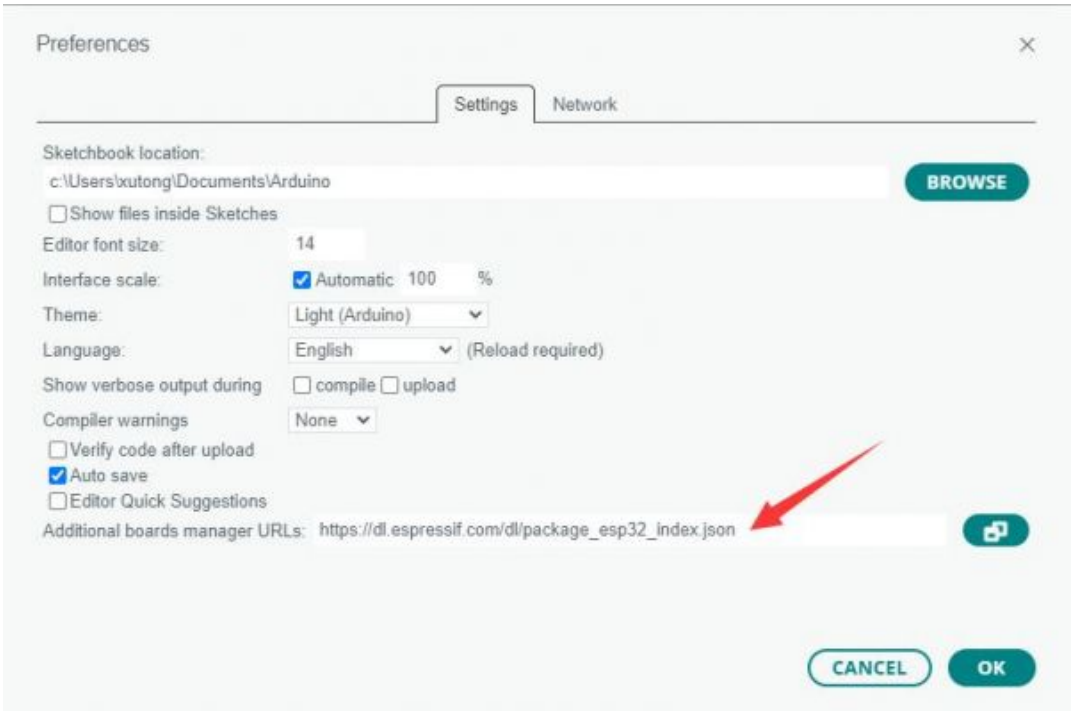
4. **Note: You will be prompted to install the driver during the installation process, we can click Install.**

Install Arduino-Pico Core on Arduino IDE

1. Open Arduino IDE, click the File on the left corner and choose "Preferences".

2. Add the following link in the additional development board manager URL, then click OK.

```
https://github.com/earlephilhower/arduino-pico/releases/download/global/package_rp2040_index.json
```

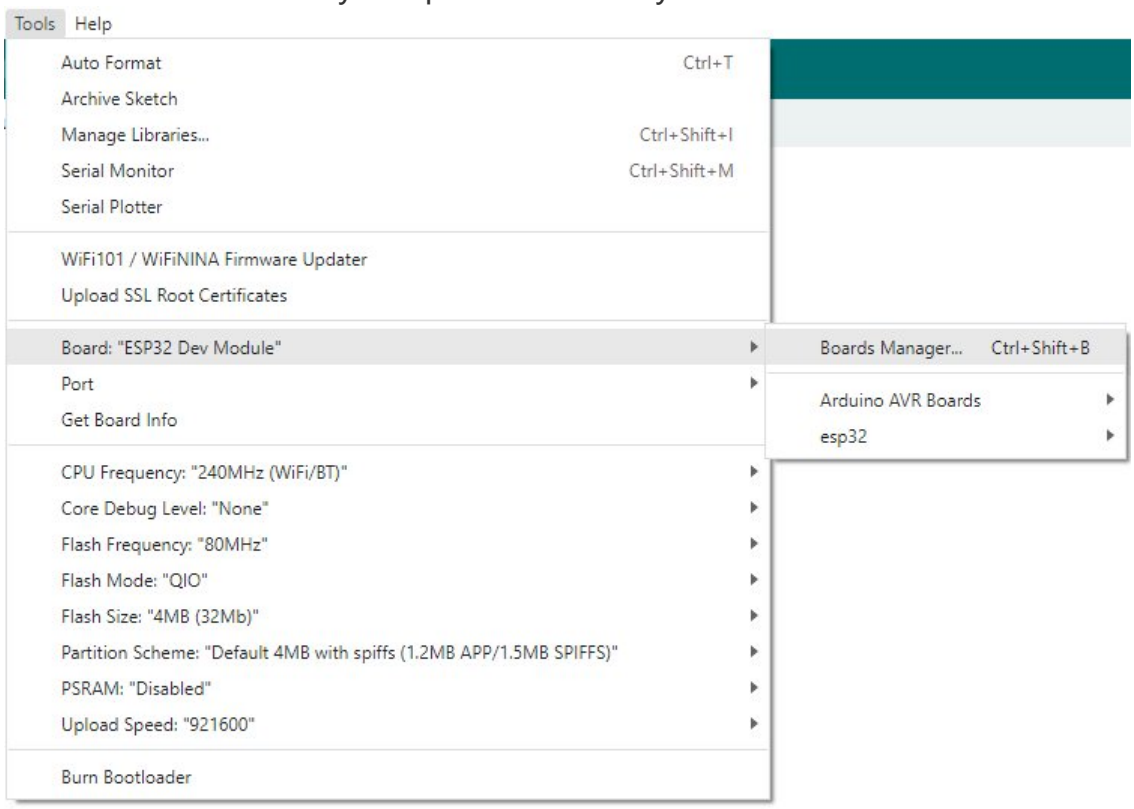


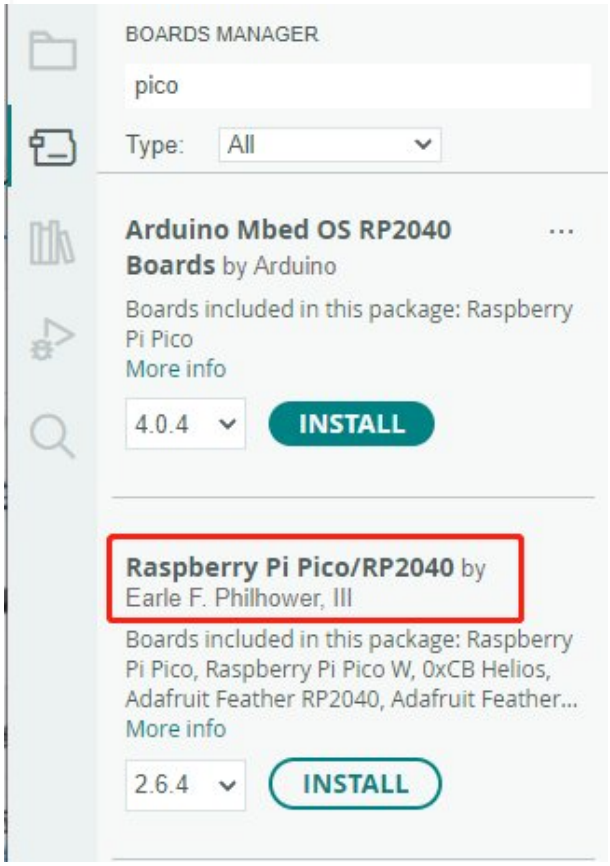
Note: If you already have the ESP8266 board URL, you can separate the URLs with commas like this:

```
https://dl.espressif.com/dl/package_esp32_index.json,https://github.co
```

```
m/earlephilhower/arduino-pico/releases/download/global/package_rp2040_i
ndex.json
```

3. Click on Tools -> Dev Board -> Dev Board Manager -> Search for pico, it shows installed since my computer has already installed it.

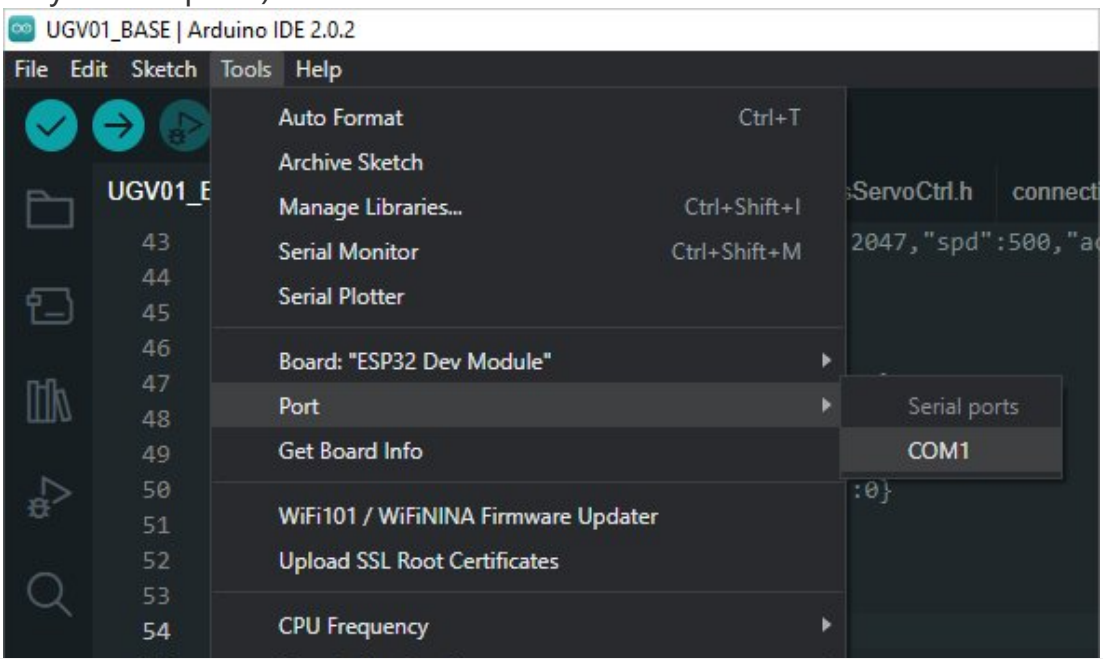


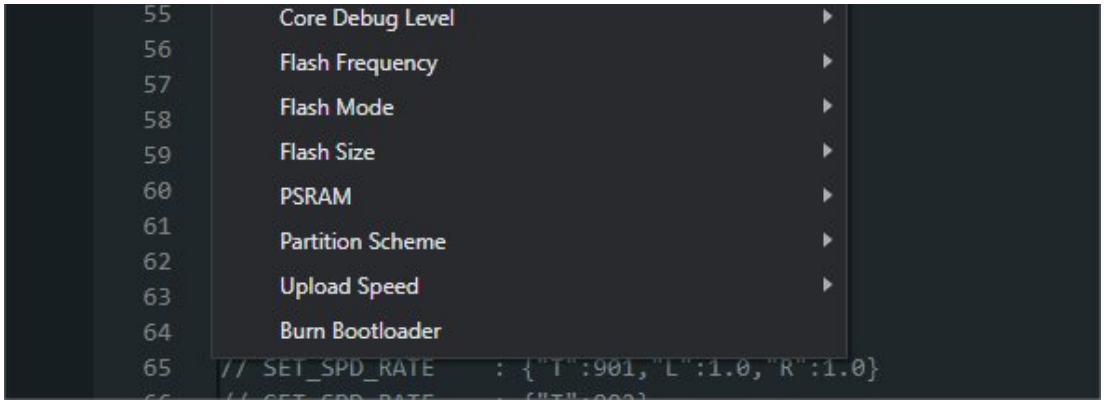


Upload Demo At the First Time

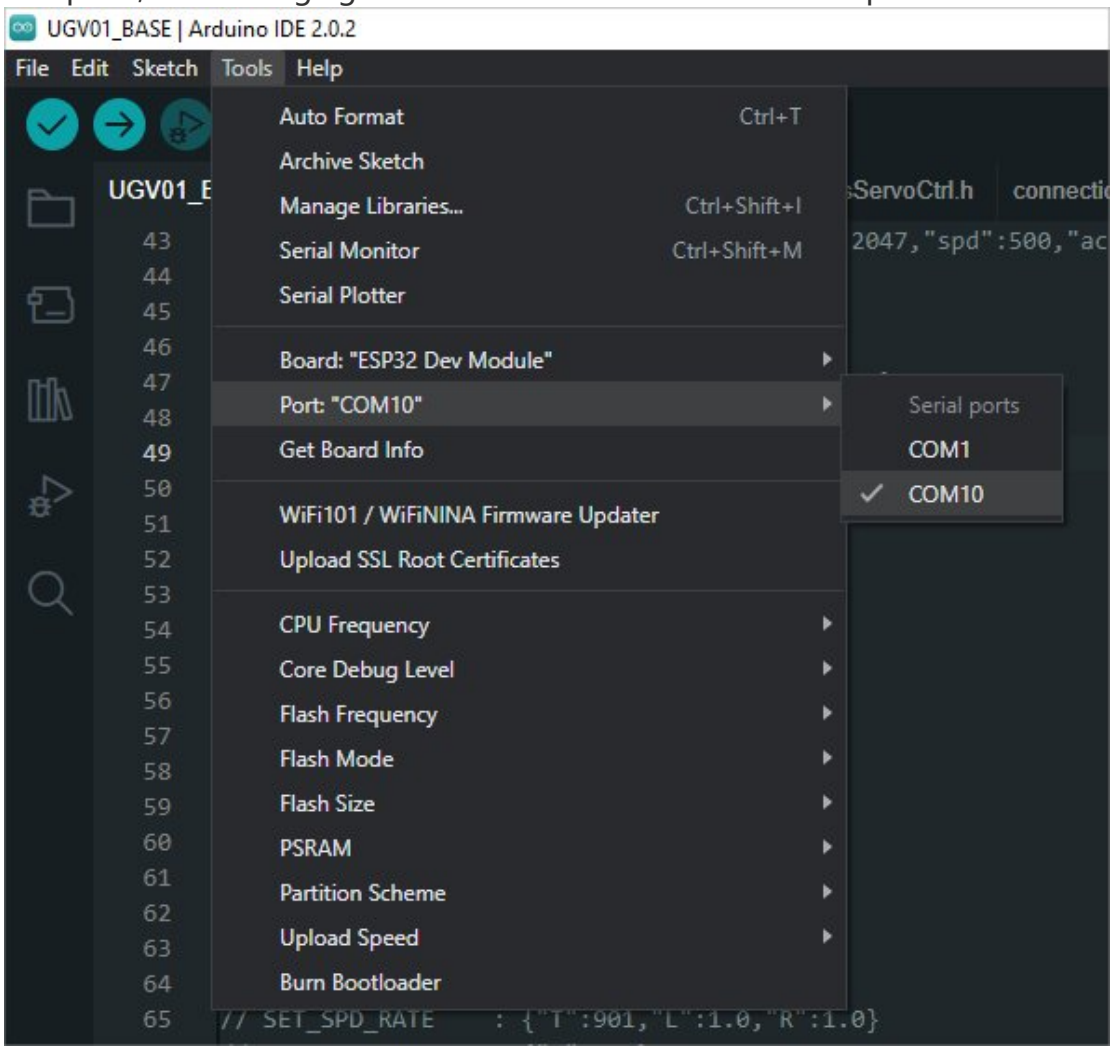
1. Press and hold the BOOTSET button on the Pico board, connect the Pico to the USB port of the computer via the Micro USB cable, and release the button when the computer recognizes a removable hard drive (RPI-RP2).

2. Download the demo, open arduino\PWM\D1-LED path under the D1-LED.ino.
3. Click Tools -> Port, remember the existing COM, do not need to click this COM (different computers show different COM, remember the existing COM on your computer).

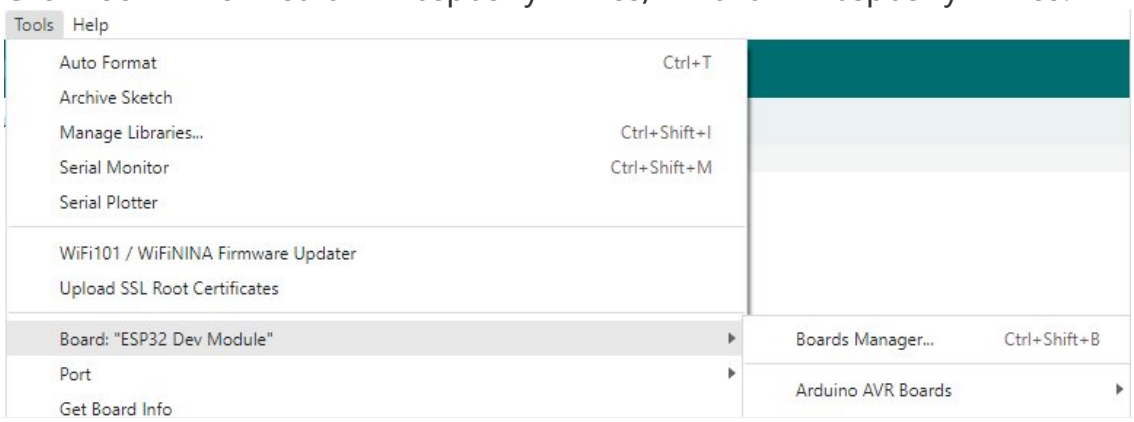




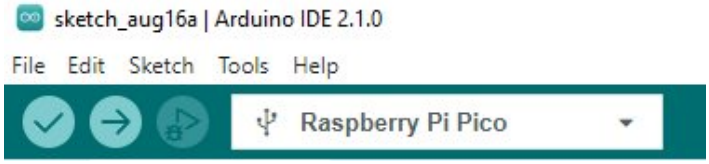
4. Connect the driver board to the computer with a USB cable, then click Tools -> Ports, select uf2 Board for the first connection, and after the upload is complete, connecting again will result in an additional COM port.



5. Click Tool -> Dev Board -> Raspberry Pi Pico/RP2040 -> Raspberry Pi Pico.



6. After setting, click the right arrow to upload.



   - If you encounter problems during the period, you need to reinstall or replace the Arduino IDE version, uninstall the Arduino IDE needs to be uninstalled cleanly, after uninstalling the software you need to manually delete all the contents of the folder C:\Users\[name]\AppData\Local\Arduino15 (you need to show the hidden files in order to see it) and then reinstall.

Pico-W Series Tutorial (To be continued...)

Open Source Demo

- MicroPython Demo (GitHub)
- MicroPython Firmware/Blink Demo (C)
- Official Raspberry Pi C/C++ Demo
- Official Raspberry Pi MicroPython Demo
- Arduino Official C/C++ Demo

Support

Technical Support

If you need technical support or have any feedback/review, please click the **Submit Now** button to submit a ticket, Our support team will check and reply to you within 1 to 2 working days. Please be patient as we make every effort to help you to resolve the issue.

Submit Now

Working Time: 9 AM - 6 AM GMT+8 (Monday to Friday)