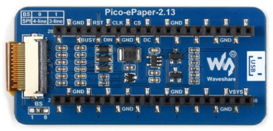
Waveshare Pico e-Paper 2.13 V4 2.13inch E-Paper E-Ink Display Module

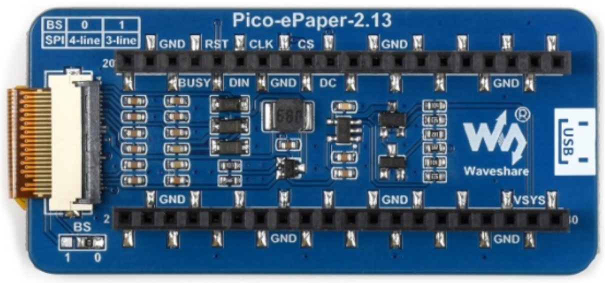


Waveshare Pico e-Paper 2.13 V4 2.13inch E-Paper E-Ink Display Module User Guide

Contents

Waveshare

Waveshare Pico e-Paper 2.13 V4 2.13inch E-Paper E-Ink Display Module



Overview

2.13inch EPD (Electronic Paper Display) Module For Raspberry Pi Pico, 250 × 122 Pixels, Black / White, SPI Interface.

Version

- V4 is completely compatible with V3 and has its own demo that supports the fast refreshing function.

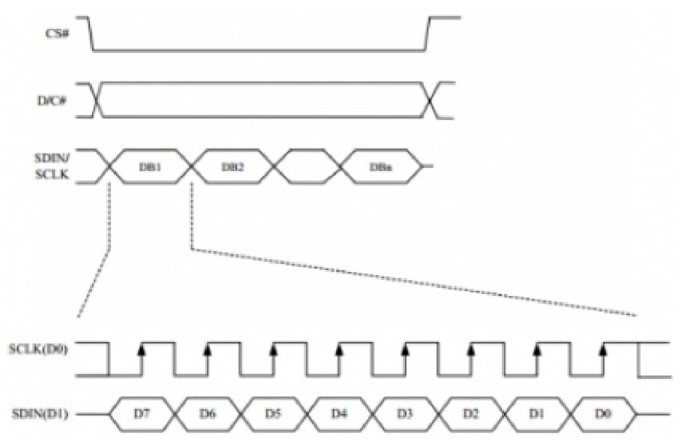
Feature

- **Dimension:**2.13inch
- **Operating voltage**: 3.3V
- **Outline dimensions (raw panel):** 59.2mm x 29.2mm x 1.05mm
- **Outline dimension**: 65.00mm x 30.50mm
- **Display size**: 48.55 x 23.71mm
- **Interface**: SPI
- **Dot pitch**: 0.194 x 0.194mm
- **Resolution**: 250 x 122 pixels
- **Display color**: Black, white
- **Greyscale**: 2
- **Partial refresh time**: 0.3s
- **full refresh time**: 2s
- **Refresh power**: 26.4mW (typ.)
- **Standby current**: <0.01uA (almost none)

Note:

- Refresh time: The refresh time is the experimental results, the actual refresh time will have errors, and the actual effect shall prevail. There will be a flickering effect during the global refresh process, this is a normal phenomenon.
- Power consumption: The power consumption data is the experimental results. The actual power consumption will have a certain error due to the existence of the driver board and the actual use situation. The actual effect shall prevail.

SPI Communication Timing



Since the ink screen only needs to be displayed, the data cable (MISO) sent from the machine and received by the host is hidden here.

- **CS:** Slave chip select, when CS is low, the chip is enabled
- **DC:** data/command control pin, write command when DC=0; write data when DC=1
- **SCLK:** SPI communication clock
- **SDIN:** SPI communication master sends, the slave receives
- **Timing:** CPHL=0, CPOL=0 (SPI0)

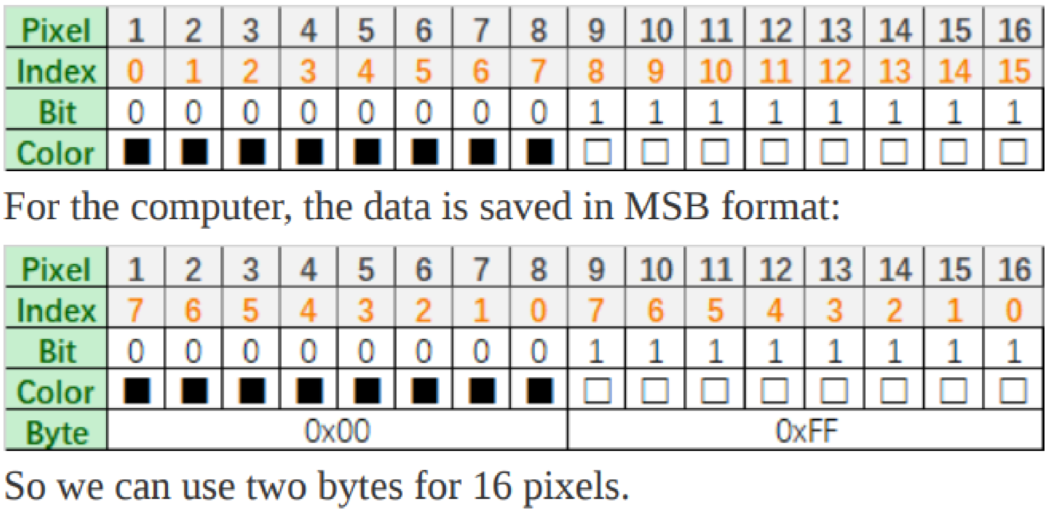
[Remarks] For specific information about SPI, you can search for information online.

Working Protocol
This product is an E-paper device adopting the image display technology of Microencapsulated Electrophoretic Display, MED. The initial approach is to create tiny spheres, in which the charged color pigments are suspended in the transparent oil and would move depending on the electronic charge. The E-paper screen displays patterns by reflecting the ambient light, so it has no background light requirement. (Note that the e- Paper cannot support updating directly under sunlight).

How to define pixels
In a monochrome picture we define the pixels, 0 is black and 1 is white.

- White □: Bit 1
- Black ■ Bit 0

The dot in the figure is called a pixel. As we know, 1 and 0 are used to define the color, therefore we can use one bit to define the color of one pixel, and 1 byte = 8pixels For example, If we set the first 8 pixels to black and the last 8 pixels to white, we show it by codes, they will be 16-bit as below:



Pixel	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
Index	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Bit	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1
Color	■	■	■	■	■	■	■	■	□	□	□	□	□	□	□	□

For the computer, the data is saved in MSB format:

Pixel	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
Index	7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0
Bit	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1
Color	■	■	■	■	■	■	■	■	□	□	□	□	□	□	□	□
Byte	0x00								0xFF							

So we can use two bytes for 16 pixels.

Precautions

1. For E-paper displays that support partial refresh, please note that you cannot refresh them with the partial refresh mode all the time. After refreshing partially several times, you need to fully refresh EPD once. Otherwise, the display effect will be abnormal, which cannot be repaired!
2. It is a normal phenomenon that the three-color EPD will have a certain color difference in different batches. Hence, It is recommended to use the program to clear all the pictures on the EPD and store it facing up. Please clear the screen several times before powering on.

3. Note that the screen cannot be powered on for a long time. When the screen is not refreshed, please set the screen to sleep mode or power off it. Otherwise, the screen will remain in a high voltage state for a long time, which will damage the e-Paper and cannot be repaired!
4. When using the e-Paper display, it is recommended that the refresh interval be at least 180s, and refresh at least once every 24 hours. If the e-Paper is not used for a long time, you should use the program to clear the screen before storing it. (Refer to the datasheet for specific storage environment requirements.)
5. After the screen enters sleep mode, the sent image data will be ignored, and it can be refreshed normally only after initializing again.
6. Control the 0x3C or 0x50 (refer to the datasheet for details) register to adjust the border color. In the demo, you can adjust the Border Waveform Control register or VCOM AND DATA INTERVAL SETTING to set the border.
7. If you find that the created image data is displayed incorrectly on the screen, it is recommended to check whether the image size setting is correct, change the width and height settings of the image and try again.
8. The working voltage of the e-Paper display is 3.3V. If you buy the raw panel and you need to add a level convert circuit for compatibility with 5V voltage. The new version of the driver board (V2.1 and subsequent versions) has added a level processing circuit, which can support both 3.3V and 5V. The old version only supports a 3.3V working environment. You can confirm the version before using it. (The one with the 20-pin chip on the PCB is generally the new version.)
9. The FPC cable of the screen is relatively fragile, pay attention to bending the cable along the horizontal direction of the screen when using it, and do not bend the cable along the vertical direction of the screen.
10. The screen of e-Paper is relatively fragile, please try to avoid dropping, bumping, and pressing hard.
11. We recommend that customers use the sample program provided by us to test with the corresponding development board.

RPi Pico

Hardware Connection

Please take care of the direction when connecting Pico. A logo of the USB port is printed to indicate the directory, you can also check the pins. If you want to connect the board by an 8-pin cable, you can refer to the table below:

e-Paper	Pico	Description
VCC	VSYS	Power input
GND	GND	Ground
DIN	GP11	MOSI pin of SPI interface, data transmitted fr om Master to Slave.
CLK	GP10	SCK pin of SPI interface, clock input
CS	GP9	Chip select pin of SPI interface, Low Active
DC	GP8	Data/Command control pin (High: Data; Low: Command)
RST	GP12	Reset pin, low active
BUSY	GP13	Busy output pin
KEY0	GP2	User key 0
KEY1	GP3	User key 1
RUN	RUN	Reset

You can just attach the board to Pico like the Pico-ePaper-7.5.

Setup Environment

You can refer to the guides for Raspberry Pi: **https://www.raspberrypi.org/documentation/pico/gettingstarted/**

Download Demo codes
Open a terminal of Pi and run the following command:

```
cd ~
sudo wget  https://www.waveshare.com/w/upload/2/27/Pico_ePaper_Code.zip
unzip Pico_ePaper_Code.zip -d Pico_ePaper_Code
cd ~/Pico_ePaper_Code
```

You can also clone the codes from Github.

```
cd ~
git clone https://github.com/waveshare/Pico_ePaper_Code.git
cd ~/Pico_ePaper_Code
```

About the examples
The guides are based on Raspberry Pi.

C codes
The example provided is compatible with several types, you need to modify the main.c file, uncomment the definition according to the actual type of display you get. For example, if you have the Pico-ePaper-2.13, please modify the main.c file, uncomment line 18 (or maybe it is line 19).

Set the project:

```
cd ~/Pico_ePaper_Code/c
```

Create a build folder and add the SDK. ../../pico-sdk is the default path of the SDK, if you save the SDK to other directories, please change it to the actual path.

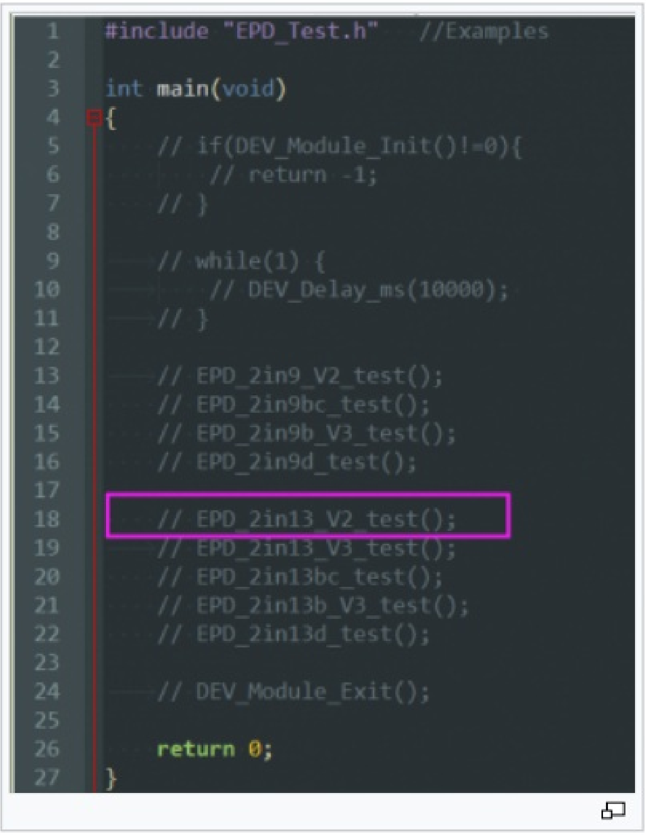
```
mkdir build
cd build
export PICO_SDK_PATH=../../pico-sdk
```

Run cmake command to generate a Makefile file.

```
cmake ..
```

Run the command make to compile the codes.

```
make -j9
```



```
#include "EPD_Test.h"   //Examples

int main(void)
{
    // if(DEV_Module_Init()!=0){
        // return -1;
    // }

    // while(1) {
        // DEV_Delay_ms(10000);
    // }

    // EPD_2in9_V2_test();
    // EPD_2in9bc_test();
    // EPD_2in9b_V3_test();
    // EPD_2in9d_test();

    // EPD_2in13_V2_test();
    // EPD_2in13_V3_test();
    // EPD_2in13bc_test();
    // EPD_2in13b_V3_test();
    // EPD_2in13d_test();

    // DEV_Module_Exit();

    return 0;
}
```

- After compiling, the epd.uf2 file is generated. Next, press and hold the BOOTSEL button on the Pico board, connect the Pico to the Raspberry Pi using the Micro USB cable, and release the button. At this point, the device will recognize a removable disk (RPI-RP2).
- Copy the epd.uf2 file just generated to the newly recognized removable disk (RPI-RP2), Pico will automatically restart the running program.

Python

- First press and hold the BOOTSEL button on the Pico board, use the Micro USB cable to connect the Pico to the Raspberry Pi, then release the button. At this point, the device will recognize a removable disk (RPIRP2).
- Copy the rp2-pico-20210418-v1.15.uf2 file in the python directory to the removable disk (RPI-RP2) just identified.
- Update Thonny IDE.

```
sudo apt upgrade thonny
```

- Open Thonny IDE (click on the Raspberry logo -> Programming -> Thonny Python IDE), and select the interpreter:
    - Select Tools -> Options... -> Interpreter.
    - Select MicroPython (Raspberry Pi Pico and ttyACM0 port).
- Open the Pico_ePaper-xxx.py file in Thonny IDE, then run the current script (click the green triangle).

C Code Analysis

Bottom Hardware Interface
We package the hardware layer for easily porting to the different hardware platforms. DEV_Config.c(.h) in the directory: Pico_ePaper_Code\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 transmits data:

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

EPD driver
The driver codes of EPD are saved in the directory: Pico_ePaper_Code\c\lib\e-Paper Open the .h header file, and you can check all the functions defined.

- Initialize e-Paper, this function is always used at the beginning and after waking up the display.

```
//2.13inch e-Paper, 2.13inch e-Paper V2, 2.13inch e-Paper (D), 2.9inch e-Paper, 2.9inch e-Paper (D)
void EPD_xxx_Init(UBYTE Mode); // Mode = 0 fully update, Mode = 1 partial update
//Other types
void EPD_xxx_Init(void);
```

xxx should be changed by the type of e-Paper, For example, if you use 2.13inch e-Paper (D), to fully update, it should be EPD_2IN13D_Init(0) and EPD_2IN13D_Init(1) for the partial update;

- **Clear:** this function is used to clear the display to white.

```
void EPD_xxx_Clear(void);
```

xxx should be changed by the type of e-Paper, For example, if you use 2.9inch e-Paper (D), it should be EPD_2IN9D_Clear();

- Send the image data (one frame) to EPD and display

```
//Bicolor version
void EPD_xxx_Display(UBYTE *Image);
//Tricolor version
void EPD_xxx_Display(const UBYTE *blackimage, const UBYTE *ryimage);
```

There are several types which are different from others

```
//Partial update for 2.13inch e-paper (D), 2.9inch e-paper (D)
void EPD_2IN13D_DisplayPart(UBYTE *Image);
void EPD_2IN9D_DisplayPart(UBYTE *Image);
```

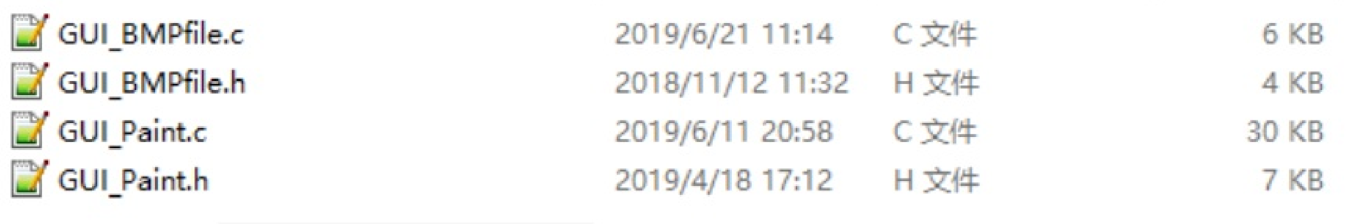
```
//For 2.13inch e-paper V2, you need to first useEPD_xxx_DisplayPartBaseImage to display a static ba
ckground and then partial update by the function EPD_xxx_DisplayPart()
void EPD_2IN13_V2_DisplayPart(UBYTE *Image);
void EPD_2IN13_V2_DisplayPartBaseImage(UBYTE *Image);
```

- Enter sleep mode

```
void EPD_xxx_Sleep(void);
```

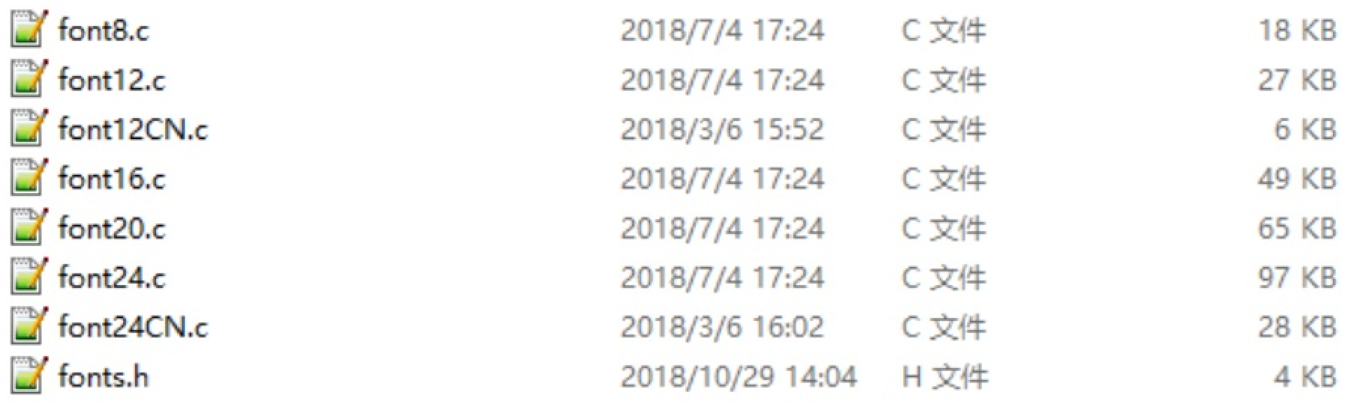
Note, You should only hardware reset or use initialize function to wake up e-Paper from sleep mode xxx is the type of e-Paper, for example, if you use 2.13inch e-Paper D, it should be EPD_2IN13D_Sleep().

Application Programming Interface
We provide basic GUI functions for testing, like draw point, line, string, and so on. The GUI function can be found in the directory: RaspberryPi_JetsonNano\c\lib\GUI\GUI_Paint.c(.h).



GUI_BMPfile.c	2019/6/21 11:14	C 文件	6 KB
GUI_BMPfile.h	2018/11/12 11:32	H 文件	4 KB
GUI_Paint.c	2019/6/11 20:58	C 文件	30 KB
GUI_Paint.h	2019/4/18 17:12	H 文件	7 KB

The fonts used can be found in the directory: RaspberryPi_JetsonNano\c\lib\Fonts.



font8.c	2018/7/4 17:24	C 文件	18 KB
font12.c	2018/7/4 17:24	C 文件	27 KB
font12CN.c	2018/3/6 15:52	C 文件	6 KB
font16.c	2018/7/4 17:24	C 文件	49 KB
font20.c	2018/7/4 17:24	C 文件	65 KB
font24.c	2018/7/4 17:24	C 文件	97 KB
font24CN.c	2018/3/6 16:02	C 文件	28 KB
fonts.h	2018/10/29 14:04	H 文件	4 KB

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

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

Select image buffer: You can create multiple image buffers at the same time and select a certain one and draw by this function.

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

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

```
void Paint_SetRotate(UWORD Rotate)
Parameters:
        Rotate: Rotate the angle of the image, the parameter can be ROTATE_0, ROTATE_90, ROTATE_18
0, ROTATE_270.
```

Note After rotating, the place of the first pixel is different, we take a 1.54-inch e-paper as an example.

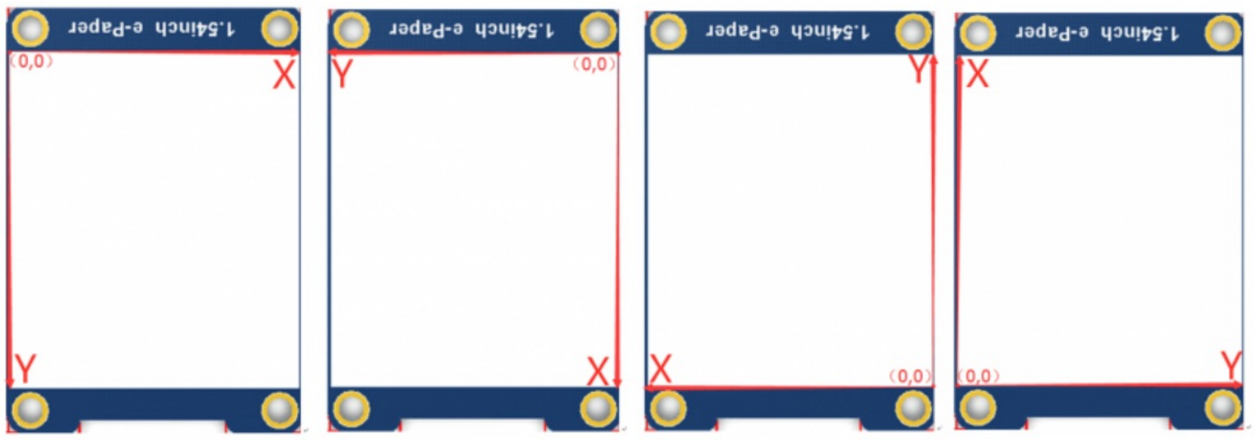


Image mirror: This function is used to set the image mirror.

```
void Paint_SetMirroring(UBYTE mirror)
Parameters:
        mirror: Mirror type if the image, the parameter can be MIRROR_NONE, MIRROR_HORIZONTAL, MIRR
OR_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)
Parameters:
        Xpoint: The X-axis value of the point in the image buffer
        Ypoint: The Y-axis value of the point in the image buffer
        Color: The color of the point
```

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

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

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

```
void Paint_ClearWindows(UWORD Xstart, UWORD Ystart, UWORD Xend, UWORD Yend, UWORD Color)
Parameters:
        Xpoint: The X-axis value of the start point in the image buffer
        Ypoint: The Y-axis value of the start point in the image buffer
        Xend: The X-axis value of the end point in the image buffer
        Yend: The Y-axis value of the end point in the image buffer
        Color: The color of the windows
```

Draw point: Draw a point at the position X point, Y point of the image buffer, you can configure the color, size, and style.

```
void Paint_DrawPoint(UWORD Xpoint, UWORD Ypoint, UWORD Color, DOT_PIXEL Dot_Pixel, DOT_STYLE Dot_St
yle)
Parameters:
        Xpoint: X-axis value of the point.
        Ypoint: Y-axis value 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, define the extended mode of the point.
                typedef enum {
                   DOT_FILL_AROUND  = 1,
                   DOT_FILL_RIGHTUP,
                } DOT_STYLE;
```

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

```
void Paint_DrawLine(UWORD Xstart, UWORD Ystart, UWORD Xend, UWORD Yend, UWORD Color, LINE_STYLE Lin
e_Style , LINE_STYLE Line_Style)
Parameters:
        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 a 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, UWORD Color, DOT_PIXEL
Line_width, DRAW_FILL Draw_Fill)
Parameters:
        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 the image buffer, use (X_Center Y_Center) as the center and Radius as the radius. You can configure the color, width of the line, and the style of the circle.

```
void Paint_DrawCircle(UWORD X_Center, UWORD Y_Center, UWORD Radius, UWORD Color, DOT_PIXEL Line_wid
th, DRAW_FILL Draw_Fill)
Parameters:
        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 character in (Xstart, Ystart) position, you can configure the font, foreground, and background.

```
void Paint_DrawChar(UWORD Xstart, UWORD Ystart, const char Ascii_Char, sFONT* Font, UWORD Color_For
eground, UWORD Color_Background)
Parameters:
        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 the string: Draw the 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)
Parameters:
        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
        Color_Foreground: foreground color
        Color_Background: background color
```

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

```
void Paint_DrawString_CN(UWORD Xstart, UWORD Ystart, const char * pString, cFONT* font, UWORD Color
_Foreground, UWORD Color_Background)
Parameters:
        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 the image buffer. You can select font, foreground, and background.

```
void Paint_DrawNum(UWORD Xpoint, UWORD Ypoint, int32_t Nummber, sFONT* Font, UWORD Color_Foregroun
d, UWORD Color_Background)
Parameters:
        Xstart: Xstart of numbers
        Ystart: Ystart of numbers
        Nummber: numbers displayed. It supports int type and 2147483647 is the maximum supported
        Font: Ascii fonts, five fonts are available:
                font8: 5*8
                font12: 7*12
                font16: 11*16
                font20: 14*20
                font24: 17*24
        Color_Foreground: foreground
        Color_Background: background
```

Display time: Display time at (Xstart Ystart) of the image buffer, you can configure fonts, foreground, and background. This function is used for partial updating. Note that some of the e-Paper don’t support partial updates and you cannot use partial updates all the time, which will have ghost problems and destroy the display.

```
void Paint_DrawTime(UWORD Xstart, UWORD Ystart, PAINT_TIME *pTime, sFONT* Font, UWORD Color_Backgro
und, UWORD Color_Foreground)
Parameters:
        Xstart: Xstart of time
        Ystart: Ystart of time
        pTime: Structure of time
        Font: Ascii font, five fonts are available
                font8: 5*8
                font12: 7*12
                font16: 11*16
                font20: 14*20
                font24: 17*24
        Color_Foreground: foreground
        Color_Background: background
```

Resource

Document

- Schematic

- 2.13inch e-Paper Specification V4 Newest
- 2.13inch e-Paper Specification V3
- 2.13inch e-Paper Specification V2

Demo codes

- Demo codes
- Github link

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

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

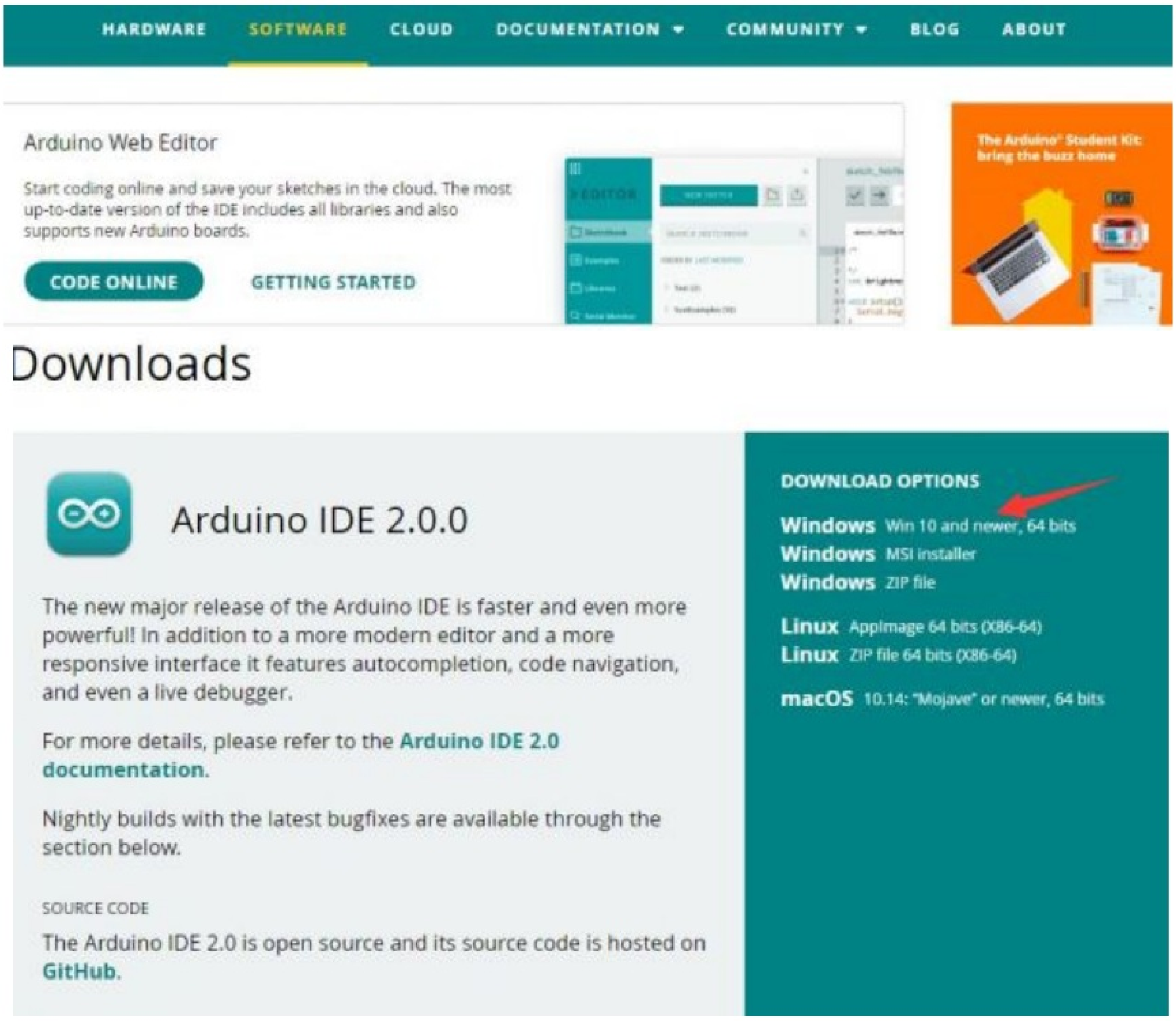
C/C++ Series

- 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 the Arduino website .



Arduino Web Editor

Start coding online and save your sketches in the cloud. The most up-to-date version of the IDE includes all libraries and also supports new Arduino boards.

Downloads

Arduino IDE 2.0.0

The new major release of the Arduino IDE is faster and even more powerful! In addition to a more modern editor and a more responsive interface it features autocompletion, code navigation, and even a live debugger.

For more details, please refer to the Arduino IDE 2.0 documentation.

Nightly builds with the latest bugfixes are available through the section below.

SOURCE CODE

The Arduino IDE 2.0 is open source and its source code is hosted on GitHub.

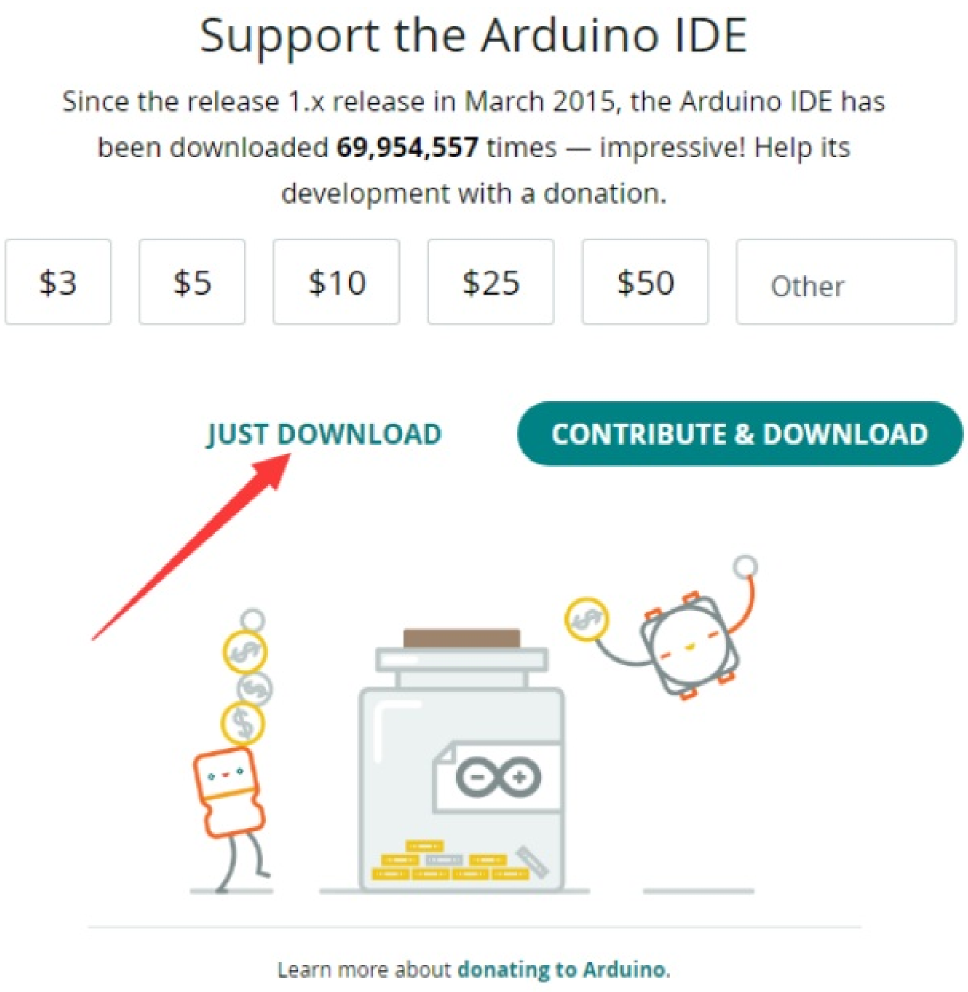
DOWNLOAD OPTIONS

Windows Win 10 and newer, 64 bits
Windows MSI installer
Windows ZIP file

Linux AppImage 64 bits (X86-64)
Linux ZIP file 64 bits (X86-64)

macOS 10.14: "Mojave" or newer, 64 bits

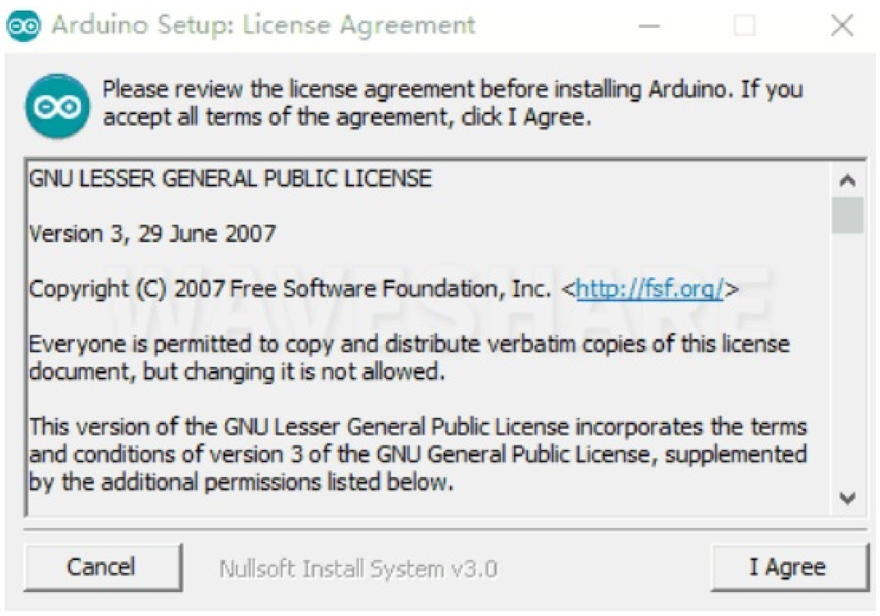
Just click on “JUST DOWNLOAD”.


Support the Arduino IDE

Since the release 1.x release in March 2015, the Arduino IDE has been downloaded **69,954,557** times — impressive! Help its development with a donation.

Learn more about **donating to Arduino**.

Click to install after downloading.



Please review the license agreement before installing Arduino. If you accept all terms of the agreement, click I Agree.

GNU LESSER GENERAL PUBLIC LICENSE

Version 3, 29 June 2007

Copyright (C) 2007 Free Software Foundation, Inc. <http://fsf.org/>

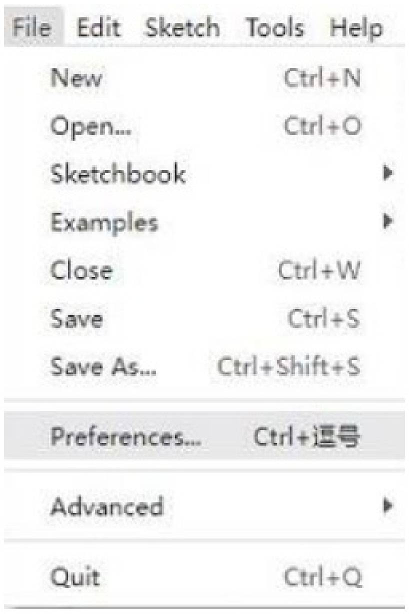
Everyone is permitted to copy and distribute verbatim copies of this license document, but changing it is not allowed.

This version of the GNU Lesser General Public License incorporates the terms and conditions of version 3 of the GNU General Public License, supplemented by the additional permissions listed below.

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

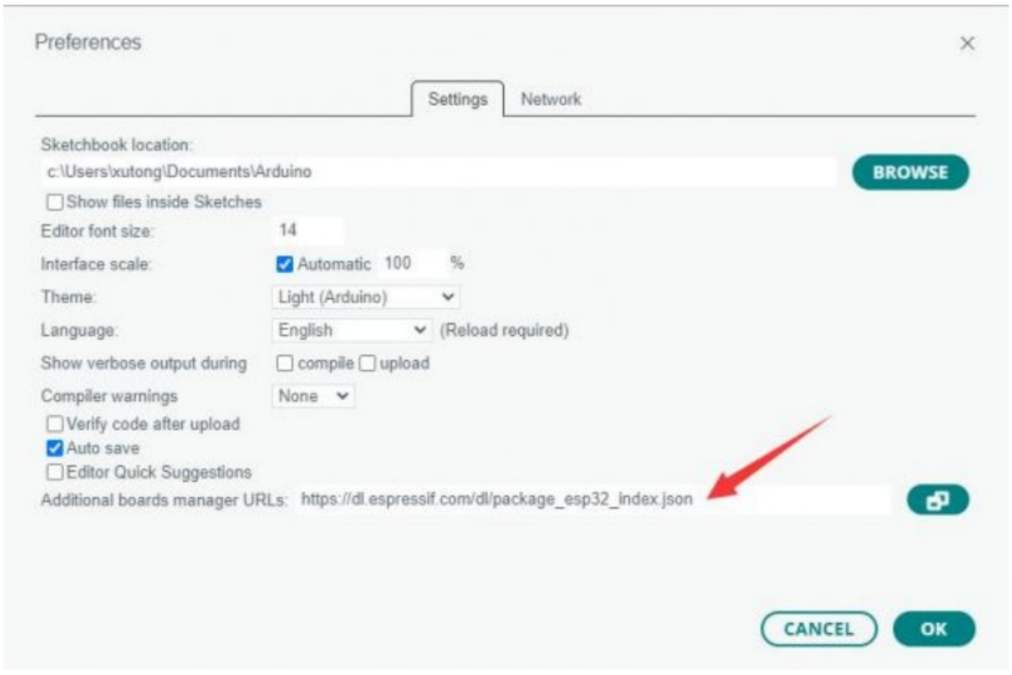
Install Arduino-Pico Core on Arduino IDE

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



Preferences

Settings | Network

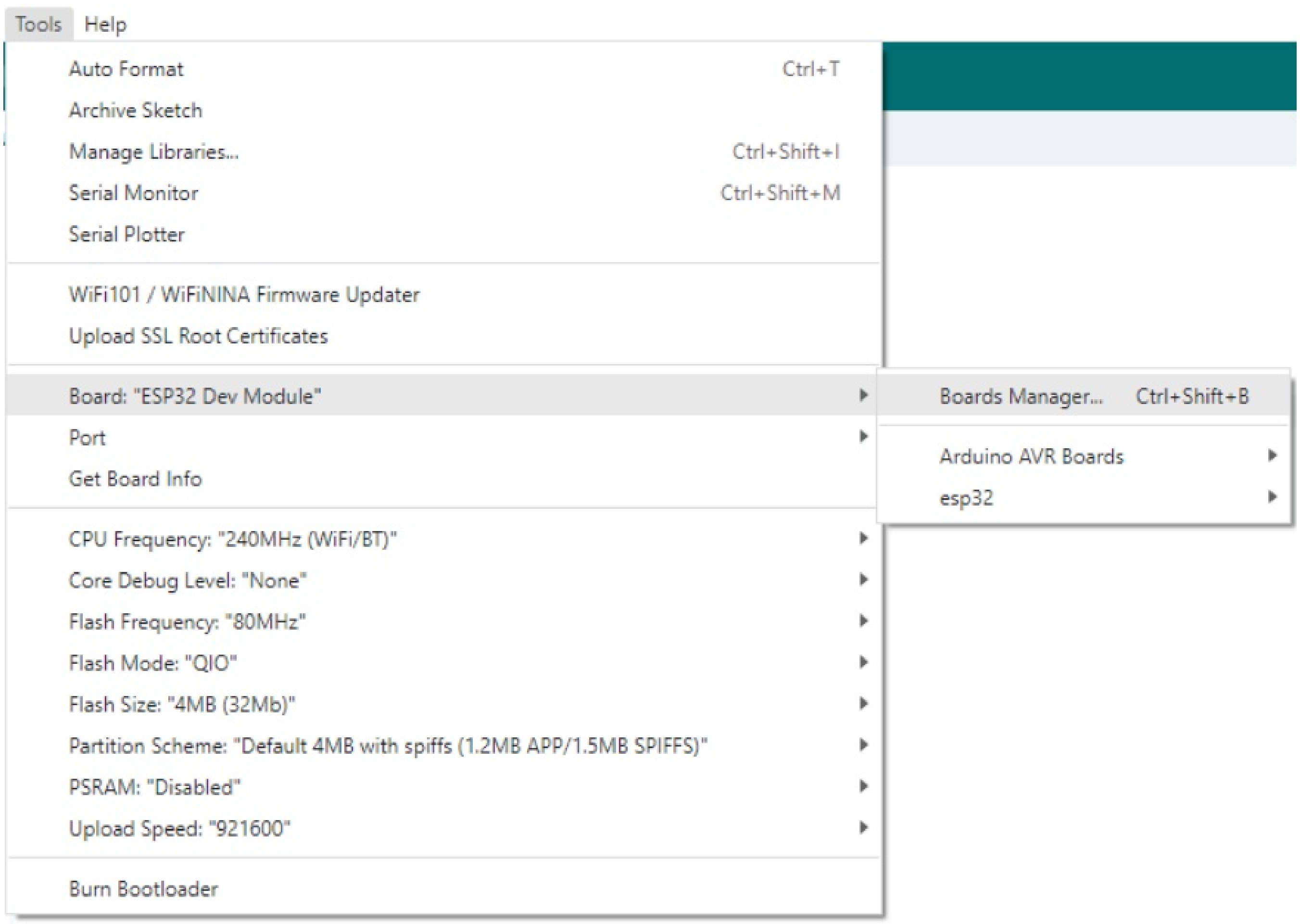
Sketchbook location:
c:\Users\xutong\Documents\Arduino BROWSE
☐ Show files inside Sketches
Editor font size: 14
Interface scale: ☑ Automatic 100 %
Theme: Light (Arduino)
Language: English (Reload required)
Show verbose output during ☐ compile ☐ upload
Compiler warnings None
☐ Verify code after upload
☑ Auto save
☐ Editor Quick Suggestions
Additional boards manager URLs: https://dl.espressif.com/dl/package_esp32_index.json

CANCEL OK

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, http://arduino.esp8266.com/stable/p
ackage_esp8266com_index.json
```

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



Tools Help

Auto Format Ctrl+T
Archive Sketch
Manage Libraries... Ctrl+Shift+I
Serial Monitor Ctrl+Shift+M
Serial Plotter

WiFi101 / WiFiNINA Firmware Updater
Upload SSL Root Certificates

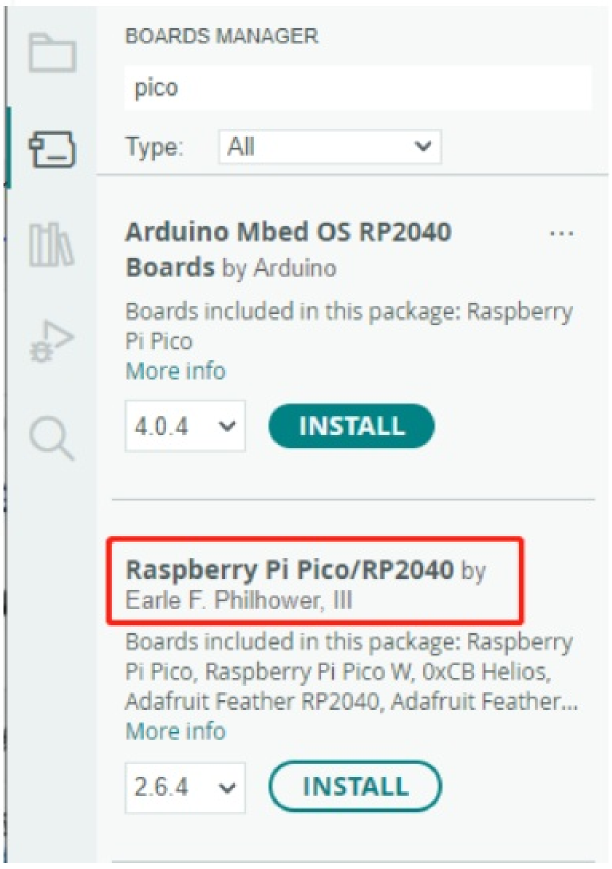
Board: "ESP32 Dev Module"
Port
Get Board Info

CPU Frequency: "240MHz (WiFi/BT)"
Core Debug Level: "None"
Flash Frequency: "80MHz"
Flash Mode: "QIO"
Flash Size: "4MB (32Mb)"
Partition Scheme: "Default 4MB with spiffs (1.2MB APP/1.5MB SPIFFS)"
PSRAM: "Disabled"
Upload Speed: "921600"

Burn Bootloader

Boards Manager... Ctrl+Shift+B

Arduino AVR Boards
esp32



BOARDS MANAGER

pico

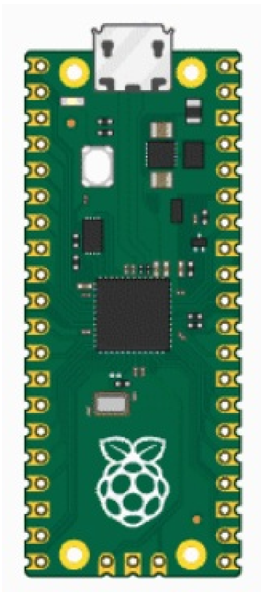
Type: All

Arduino Mbed OS RP2040 Boards by Arduino
Boards included in this package: Raspberry Pi Pico
More info
4.0.4 INSTALL

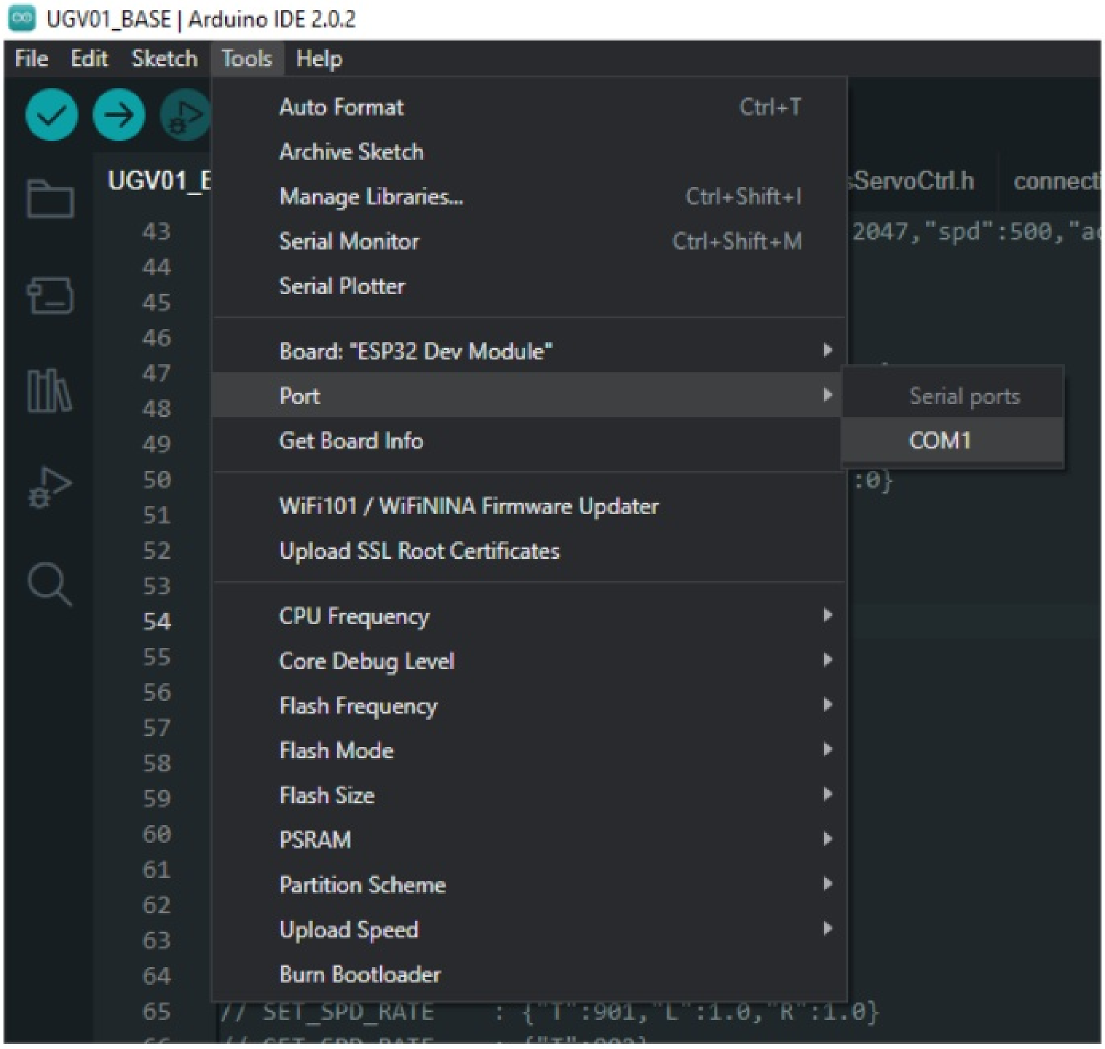
Raspberry Pi Pico/RP2040 by Earle F. Philhower, III
Boards included in this package: Raspberry Pi Pico, Raspberry Pi Pico W, 0xCB Helios, Adafruit Feather RP2040, Adafruit Feather...
More info
2.6.4 INSTALL

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).

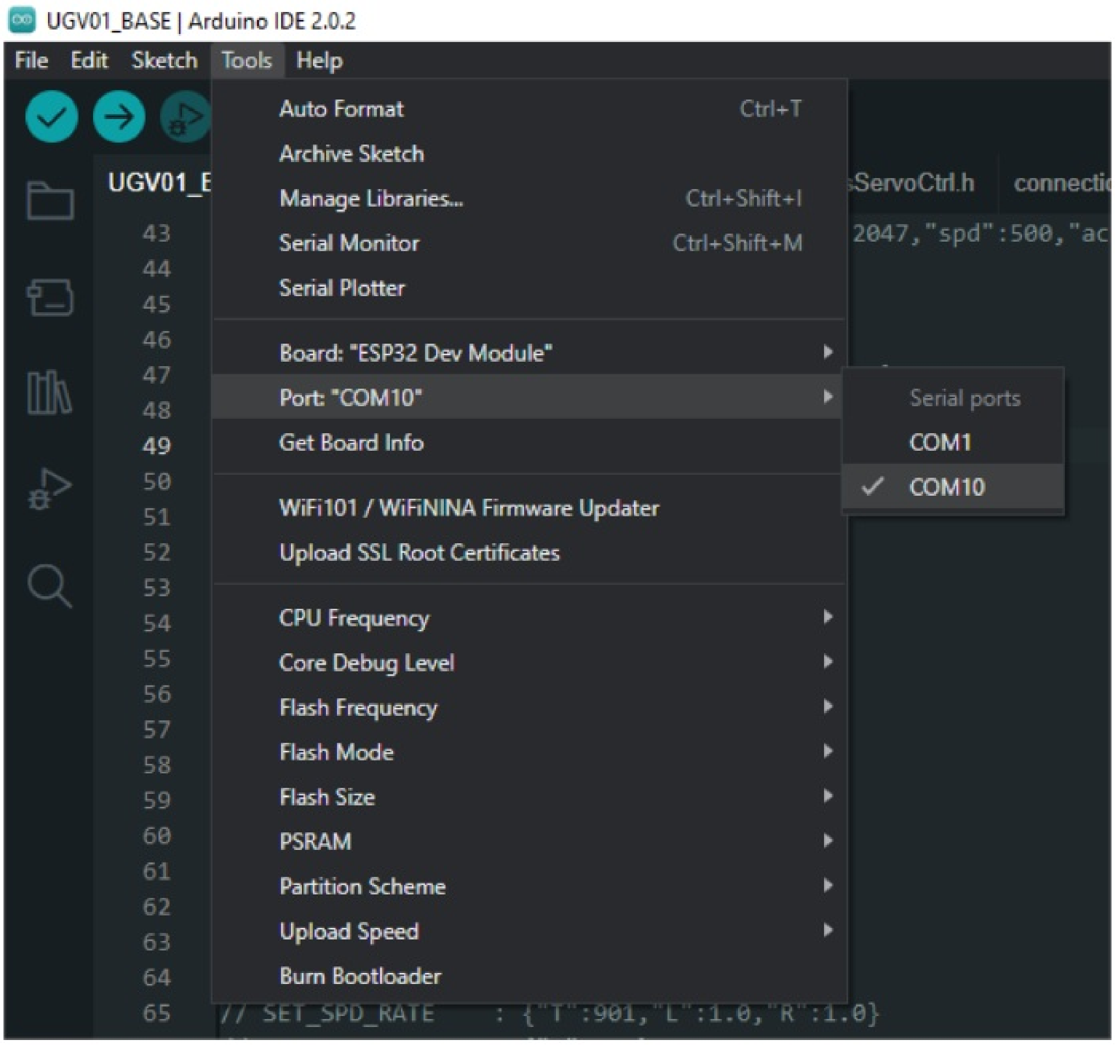


2. Download the demo, open the arduino\PWM\D1-LED path under the D1-LED.ino.
3. Click Tools -> Port, remember the existing COM, do not need to click this COM (different computers show different COM, so remember the existing COM on your computer).



```
2047,"spd":500,"a
:0}
// SET_SPD_RATE    : {"T":901,"L":1.0,"R":1.0}
```

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.

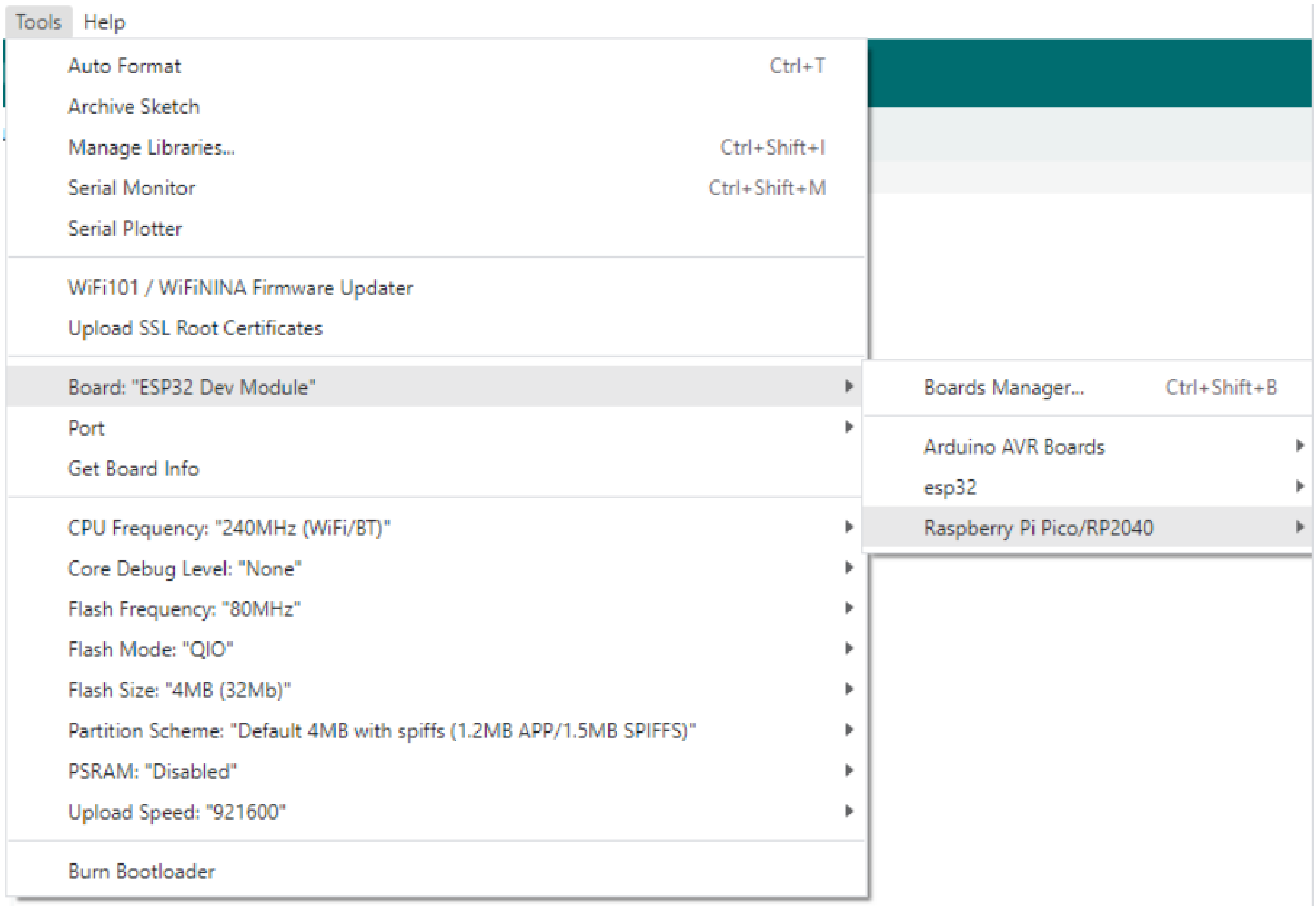


Board: "ESP32 Dev Module"

Port: "COM10"

```
2047,"spd":500,"ac
// SET_SPD_RATE     : {"T":901,"L":1.0,"R":1.0}
```

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


Tools
Help
Auto Format Ctrl+T
Archive Sketch
Manage Libraries... Ctrl+Shift+I
Serial Monitor Ctrl+Shift+M
Serial Plotter
WiFi101 / WiFiNINA Firmware Updater
Upload SSL Root Certificates
Port
Get Board Info
Burn Bootloader
Boards Manager... Ctrl+Shift+B
Arduino AVR Boards
esp32
Raspberry Pi Pico/RP2040

Board: "ESP32 Dev Module"

CPU Frequency: "240MHz (WiFi/BT)"

Core Debug Level: "None"

Flash Frequency: "80MHz"

Flash Mode: "QIO"

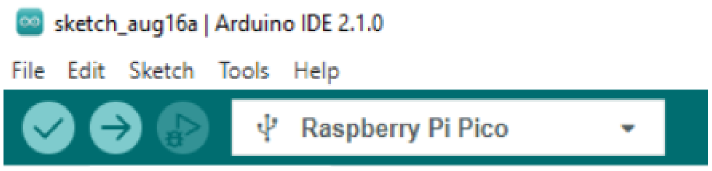
Flash Size: "4MB (32Mb)"

Partition Scheme: "Default 4MB with spiffs (1.2MB APP/1.5MB SPIFFS)"

PSRAM: "Disabled"

Upload Speed: "921600"

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


sketch_aug16a | Arduino IDE 2.1.0
File Edit Sketch Tools Help
Raspberry Pi Pico

- 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

FAQ

Question:What is the usage environment of the e-ink screen?

Answer:

- Operating conditions Temperature range: 0~50°C; Humidity range: 35%~65%RH.
- Storage conditions Temperature range: below 30°C; Humidity range: below 55%RH; Maximum storage time: 6 months.
- Transport conditions Temperature range: -25~70°C; Maximum transportation time: 10 days.
- After unpacking Temperature range: 20°C±5°C; Humidity range: 50±5%RH; Maximum storage time: Assemble within 72 hours.

Question:Precautions for e-ink screen refresh?

Answer:

Refresh mode

- Full refresh: The electronic ink screen will flicker several times during the refresh process (the number of flickers depends on the refresh time), and the flicker is to remove the afterimage to achieve the best display effect.
- Partial refresh: The electronic ink screen has no flickering effect during the refresh process. Users who use the partial brushing function note that after refreshing several times, a full brush operation should be performed to remove the residual image, otherwise the residual image problem will become more and more serious, or even damage the screen (currently only some black and white e-ink screens support partial brushing, please refer to

product page description).

Refresh rate

- During use, it is recommended that customers set the refresh interval of the e-ink screen to at least 180 seconds (except for products that support the local brush function)
- During the standby process (that is, after the refresh operation), it is recommended that the customer set the e-ink screen to sleep mode, or power off operation (the power supply part of the ink screen can be disconnected with an analog switch) to reduce power consumption and prolong the life of the e-ink screen. (If some e-ink screens are powered on for a long time, the screen will be damaged beyond repair.)
- During the use of the three-color e-ink screen, it is recommended that customers update the display screen at least once every 24 hours (if the screen remains the same screen for a long time, the screen burn will be difficult to repair).

Usage scenarios
The e-ink screen is recommended for indoor use. If you use it outdoors, you need to avoid direct sunlight on the e-ink screen and take UV protection measures at the same time. When designing e-ink screen products, customers should pay attention to determining whether the use environment meets the temperature and humidity requirements of the e-ink screen.

Question: Chinese cannot be displayed on the e-ink screen?

Answer:
The Chinese character library of our routine uses the GB2312 encoding method, please change your xxx_test.c file to GB2312 encoding format, compile and download it, and then it can be displayed normally.

Question: After using it for some time, the screen refresh (full refresh) has a serious afterimage problem that cannot be repaired.

Answer:
Power on the development board for a long time, after each refresh operation, it is recommended to set the screen to sleep mode or directly power off processing, otherwise, the screen may burn out when the screen is in a high voltage state for a long time.

Question: e-Paper show a black border?
Answer:
The border display color can be set through the Border Waveform Control register or the VCOM AND DATA INTERVAL SETTING register.

Question: What is the specification of the screen cable interface?

Answer:
0.5mm pitch, 24Pin.

- In this case, the customer needs to reduce the position of the round brush and clear the screen after 5 rounds of brushing (increasing the voltage of VCOM can improve the color, but it will increase the afterimage).

Question: After the ink screen enters deep sleep mode, can it be refreshed again?
Answer: Yes, but you need to re-initialize the electronic paper with software.

Question: When the 2.9-inch EPD is in deep sleep mode, the first time it wakes up, the screen refresh will be unclean. How can I solve it?

Answer:
The process of re-awakening the e-ink screen is the process of re-powering on, so when the EPD wakes up, the screen must be cleared first, to avoid the afterimage phenomenon to the greatest extent.

Question: Are bare-screen products shipped with a surface coating?
Answer: with film.

Question: Does the e-paper have a built-in temperature sensor?
Answer:
Yes, you can also use the IIC pin external LM75 temperature sensor.

Question: When testing the program, the program keeps stuck on an e-paper busy.

Answer:
It may be caused by the unsuccessful spi driver 1. Check whether the wiring is correct 2. Check whether the spi is turned on and whether the parameters are configured correctly (spi baud rate, spin mode, and other parameters).

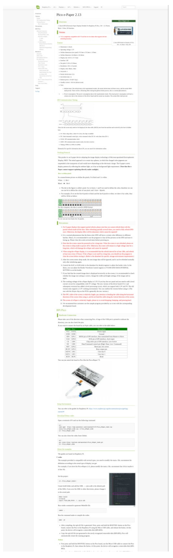
Question: What is the refresh rate/lifetime of this e-ink screen?

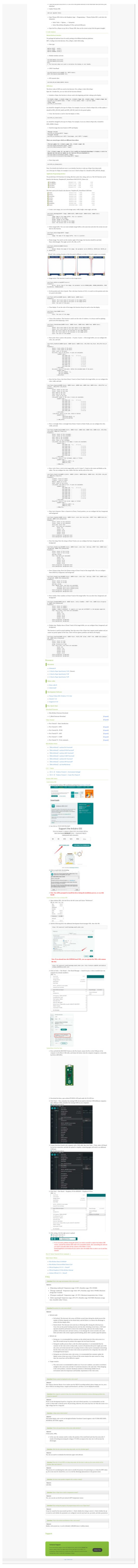
Answer:
Ideally, with normal use, it can be refreshed 1,000,000 times (1 million times).

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

Documents / Resources





Waveshare Pico e-Paper 2.13 V4 2.13inch E-Paper E-Ink Display Module [pdf] User Guide
Pico e-Paper 2.13 V4 2.13inch E-Paper E-Ink Display Module, Pico e-Paper 2.13 V4, 2.13inch E-Paper E-Ink Display Module, E-Ink Display Module, Display Module

References

- dl.espressif.com/dl/package_esp32_index.json
- GitHub - waveshareteam/Pico_ePaper_Code: Waveshrae Pico e-Paper driver code
- User Manual