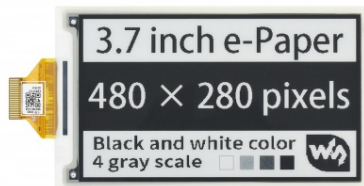


3.7inch e-Paper HAT

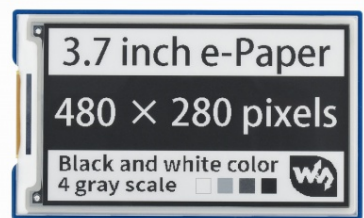
From Waveshare Wiki

3.7inch e-Paper



480x280, 3.7inch E-Ink raw panel without driver board

3.7inch e-paper HAT



480x280, 3.7inch E-Ink display HAT for Raspberry Pi, SPI interface

Primary Attribute

Category: OLEDs / LCDs, LCD e-Paper

Brand: Waveshare

Website

International: Waveshare website (<http://www.waveshare.com/3.7inch-e-paper-hat.htm>)

Chinese: 官方中文站点 (<http://www.waveshare.net/shop/3.7inch-e-paper-hat.htm>)

Onboard Interfaces

SPI

Related Products

- 4.3inch e-Paper UART Module
- 13.3inch e-Paper HAT
- 13.3inch HDMI e-Paper
- 12.48inch e-Paper Module
- 12.48inch e-Paper Module (B)
- 10.3inch HDMI e-Paper
- 10.3inch e-Paper HAT (D)
- 10.3inch e-Paper HAT
- 9.7inch HDMI e-Paper
- 9.7inch e-Paper HAT
- 7.8inch e-Paper HAT
- 7.8inch HDMI e-Paper
- 7.5inch HD e-Paper HAT
- 7.5inch HD e-Paper HAT (B)
- 7.5inch e-Paper HAT
- 7.5inch e-Paper HAT (B)
- 7.5inch e-Paper HAT (C)
- 7.5inch NFC-Powered e-Paper
- 7.5inch NFC-Powered HD e-Paper
- 6inch e-Paper HAT

- 6inch HD e-Paper HAT
- 5.83inch e-Paper HAT
- 5.83inch e-Paper HAT (B)
- 5.83inch e-Paper HAT (C)
- 5.65inch e-Paper Module (F)
- 4.2inch e-Paper Cloud Module
- 4.2inch e-Paper Module
- 4.2inch e-Paper Module (B)
- 4.2inch e-Paper Module (C)
- 4.2inch NFC-Powered e-Paper
- 4.01inch e-Paper (F)
- 4.01inch e-Paper HAT (F)
- **3.7inch e-Paper**
- **3.7inch e-Paper HAT**
- 2.9inch e-Paper Module
- 2.9inch e-Paper Module (B)
- 2.9inch e-Paper Module (C)
- 2.9inch e-Paper HAT (D)
- 2.9inch Touch e-Paper HAT
- 2.9inch NFC-Powered e-Paper
- 2.7inch NFC-Powered e-Paper Module
- 2.7inch e-Paper HAT
- 2.7inch e-Paper HAT (B)
- 2.66inch e-Paper
- 2.66inch e-Paper Module
- 2.66inch e-Paper (B)
- 2.66inch e-Paper Module (B)
- 2.13inch e-Paper Cloud Module
- 2.13inch e-Paper HAT
- 2.13inch e-Paper HAT (B)
- 2.13inch e-Paper HAT (C)
- 2.13inch e-Paper HAT (D)
- 2.13inch Touch e-Paper HAT
- 2.13inch NFC-Powered e-Paper
- 1.54inch NFC-Powered e-Paper (BB)
- 1.54inch NFC-Powered e-Paper (BW)
- 1.54inch e-Paper Module
- 1.54inch e-Paper Module (B)
- 1.54inch e-Paper Module (C)
- 1.02inch e-paper Module
- e-Paper Driver HAT
- EINK-DISP-103
- E-Paper Shield
- e-Paper ESP8266 Driver Board
- E-Paper ESP32 Driver Board
- E-Paper NB-IoT GPRS HAT

Contents

- 1 Introduction
- 2 Interfaces
 - 2.1 SPI timing
 - 2.2 Working protocol
 - 2.2.1 Pixel & Byte
 - 2.3 Hardware connection
 - 2.4 Enable SPI interface
 - 2.5 Install libraries
 - 2.6 Download demo codes
 - 2.7 Examples
 - 2.7.1 C
 - 2.7.2 Supports type
 - 2.7.3 Python
 - 2.7.4 Supports type
 - 2.8 Description of codes (API)
 - 2.8.1 C
 - 2.8.2 GUI functions
 - 2.8.3 Testing Code
 - 2.9 Python (Can be used for Jetson nano and Raspberry Pi)
 - 2.9.1 epdconfig.py

- 2.9.2 epdxxx.py(xxx is the type of e-Paper)
 - 2.9.3 epd_XXX_test.py(xxx is type of e-paper)
 - 2.9.4 Orientation
 - 2.9.5 GUI
- 2.10 Hardware connection
- 2.11 Install libraries
- 2.12 Download demo codes
- 2.13 Examples
 - 2.13.1 C
 - 2.13.2 Supports type
 - 2.13.3 Python
 - 2.13.4 Supports type
- 2.14 Description of codes (API)
 - 2.14.1 C
 - 2.14.2 GUI functions
 - 2.14.3 Testing Code
- 2.15 Python (Can be used for Jetson nano and Raspberry Pi)
 - 2.15.1 epdconfig.py
 - 2.15.2 epdxxx.py(xxx is the type of e-Paper)
 - 2.15.3 epd_XXX_test.py(xxx is type of e-paper)
 - 2.15.4 Orientation
 - 2.15.5 GUI
- 2.16 Hardware connection
- 2.17 Software settings
 - 2.17.1 Supports type
- 2.18 Codes description
 - 2.18.1 Bottom hardware interface
 - 2.18.2 Middle EPD driver
 - 2.18.3 Application function
- 2.19 Hardware connection
- 2.20 Install Arduino IDE
- 2.21 Run examples
 - 2.21.1 Supports type
- 2.22 Code Description
 - 2.22.1 Files
 - 2.22.2 Bottom hardware interface
 - 2.22.3 Middle EPD driver
 - 2.22.4 Application functions
- 2.23 Setup Arduino IDE (Windows)
- 2.24 Download Demo Codes
- 2.25 Hardware connection
- 2.26 Run the Demo Codes
- 2.27 Setup Arduino IDE (Windows)
- 2.28 Download Demo Codes
- 2.29 Hardware connection
- 2.30 Run the Demo Codes
- 2.31 Resources
 - 2.31.1 Documentation
 - 2.31.2 Demo code
 - 2.31.3 Related Resources
- 2.32 FAQ
- 2.33 Support

Introduction

Note: The raw panel requires a driver board, If you are the first time use this e-Paper, we recommend you to buy the HAT version or buy more one driver hat for easy use, otherwise you need to make the driver board yourself. And this instruction is based on the HAT version.

480x280, 3.7inch E-Ink display HAT for Raspberry Pi, SPI interface

More (<http://www.waveshare.com/3.7inch-e-paper-hat.htm>)

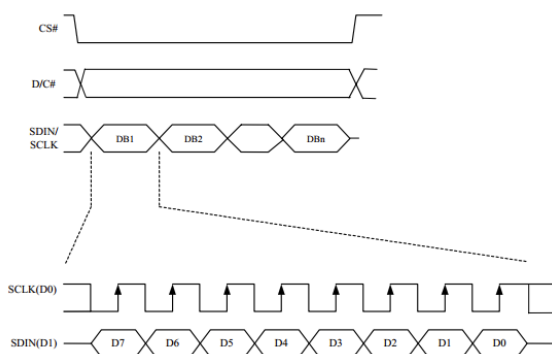
Interfaces

VCC	3.3V
GND	GND
DIN	SPI MOSI
CLK	SPI SCK
CS	SPI chip select (Low active)
DC	Data/Command control pin (High for data, and low for command)
RST	External reset pin (Low for reset)
BUSY	Busy state output pin (Low for busy)

SPI timing

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 suspending in the transparent oil and would move depending on the electronic charge. The E-paper screen display patterns by reflecting the ambient light, so it has no background light requirement. Under ambient light, the E-paper screen still has high visibility with a wide viewing angle of 180 degrees. It is the ideal choice for E-reading. (**Note that the e-Paper cannot support updating directly under sunlight**)

Communication 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.

- CS is the slave chip select. When CS is low, the chip is enabled.
- DC is data/command control pin, when DC = 0, write command, when DC = 1, write data.
- SCLK is the SPI communication clock.
- SDIN is the data line from the master to the slave in SPI communication.

SPI communication has data transfer timing, which is combined with CPHA and CPOL.

1. CPOL determines the level of the serial synchronous clock at idle state. When CPOL = 0, the level is Low. However, CPOL has little effect to the transmission.
2. CPHA determines 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.

As you can see from the figure above, data transmission starts at the first falling edge of SCLK, and 8 bits of data are transferred in one clock cycle. Here, SPI0 is in used, and data is transferred by bits, MSB first.

Working protocol

Pixel & Byte

We define the pixels in a monochrome picture, 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 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 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.

- In addition to bicolor displaying, the 3.7inch e-Paper also supports four grayscale.

To display grey pixels, we need to define data for gray

- Black: 00b
- Dark Grey: 10b
- Light Grey: 01b
- White: 11b

Pixel	1		2		3		4		5		6		7		8	
Bit	7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0
Data	0	0	0	0	1	0	1	0	0	1	0	1	1	1	1	1
Color	Black		Black		Dark Grey		Dark Grey		Light Grey		Light Grey		White		White	
Byte	0x0A								0x5F							

The display divides a four-grayscale picture into two pictures. The pixels in the same position of pictures are combined into one pixel.

Register	White	Light Grey	Dark Grey	Black
0x24	0x01	0x00	0x01	0x00
0x26	0x01	0x01	0x00	0x00

With the tables above, you can define the data which can be used to display grayscale pixels in the 3.7inch e-Paper

Pixel	1		2		3		4		5		6		7		8	
Bit	7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0
Data	0	0	0	0	1	0	1	0	0	1	0	1	1	1	1	1
Color	Black		Black		Dark Grey		Dark Grey		Light Grey		Light Grey		White		White	
Byte	0x0A								0x5F							
Bit in 0x24 register	0		0		1		1		0		0		1		1	
Bit in 0x26 register	0		0		0		0		1		1		1		1	
Data sent to 0x24 register	0x33															
Data sent to 0x26 register	0x0F															

We provide several examples for testing, you can test the e-Paper by following the guides. If your developing board is different, you need to write your codes yourself by following the resources.

Hardware connection

If the e-Paper you have is the HAT version which has 40pin GPIO, you can directly attach the e-Paper HAT on Raspberry Pi, otherwise, you can connect your e-Paper to Raspberry Pi by 8pins cable provided.

To connect the e-Paper, you can following the table below

Connect to Raspberry Pi		
e-Paper	Raspberry Pi	
	BCM2835	Board
VCC	3.3V	3.3V
GND	GND	GND
DIN	MOSI	19
CLK	SCLK	23
CS	CE0	24
DC	25	22
RST	17	11
BUSY	24	18

Enable SPI interface

The communication interface of e-Paper is SPI, to use it, we should firstly enable SPI interface of SPI. Open terminal of Raspberry Pi, and open configuration by the following command

```
sudo raspi-config
```

Choose Interfacing Options -> SPI -> Yes

```

1 Change User Password  Change password for the current user
2 Network Options      Configure network settings
3 Boot Options         Configure options for start-up
4 Localisation Options Set up language and regional settings to match your location
5 Interfacing Options  Configure connections to peripherals
6 Overclock            Configure overclocking for your Pi
7 Advanced Options     Configure advanced settings
8 Update              Update this tool to the latest version
9 About raspi-config   Information about this configuration tool

```

```

P1 Camera      Enable/Disable connection to the Raspberry Pi Camera
P2 SSH         Enable/Disable remote command line access to your Pi using SSH
P3 VNC         Enable/Disable graphical remote access to your Pi using RealVNC
P4 SPI         Enable/Disable automatic loading of SPI kernel module
P5 I2C         Enable/Disable automatic loading of I2C kernel module
P6 Serial      Enable/Disable shell and kernel messages on the serial connection
P7 1-Wire      Enable/Disable one-wire interface
P8 Remote GPIO Enable/Disable remote access to GPIO pins

```

Would you like the SPI interface to be enabled?

<Yes>

<No>

Restart Raspberry Pi

```
sudo reboot
```

Install libraries

Open terminal of Raspberry Pi and run the following commands to install corresponding libraries:

■ Install BCM2835 libraries

```

wget http://www.airspayce.com/mikem/bcm2835/bcm2835-1.60.tar.gz
tar zxvf bcm2835-1.60.tar.gz
cd bcm2835-1.60/
sudo ./configure
sudo make
sudo make check
sudo make install

```

■ Install WiringPi libraries

```

sudo apt-get install wiringpi
wget https://project-downloads.drogon.net/wiringpi-latest.deb
sudo dpkg -i wiringpi-latest.deb
gpio -v

```

■ Install Python2 libraries

```

sudo apt-get update
sudo apt-get install python-pip
sudo apt-get install python-pil
sudo apt-get install python-numpy
sudo pip install RPi.GPIO
sudo pip install spidev

```

■ Install Python3 libraries

```

sudo apt-get update
sudo apt-get install python3-pip
sudo apt-get install python3-pil
sudo apt-get install python3-numpy
sudo pip3 install RPi.GPIO
sudo pip3 install spidev

```

Download demo codes

Open the terminal of Raspberry Pi and clone demo codes by the following commands:

```
sudo git clone https://github.com/waveshare/e-Paper
```

Examples

C

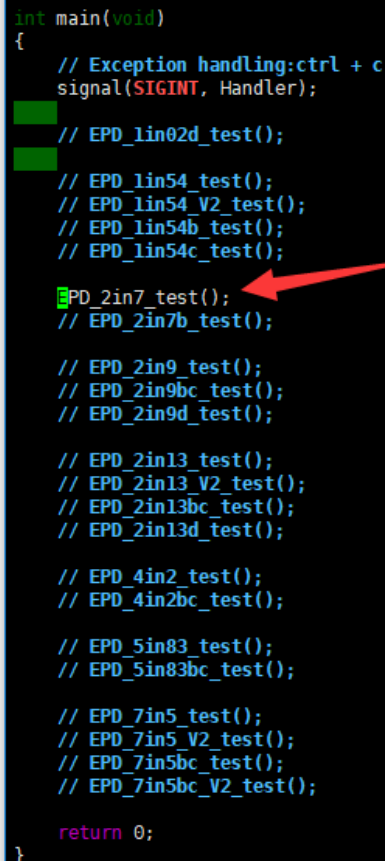
- Enter the folder of C examples

```
cd ~/e-Paper/RaspberryPi&JetsonNano/  
cd c
```

- Modify the main.c file for corresponding e-Paper

```
sudo nano examples/main.c
```

For example, if you want to update a 2.7inch e-Paper/2.7inch e-Paper HAT, you should modify the main.c file, uncomment the line `EPD_2in7_test()`, comment others, and save.



```
int main(void)  
{  
    // Exception handling:ctrl + c  
    signal(SIGINT, Handler);  
  
    // EPD_1in02d_test();  
  
    // EPD_1in54_test();  
    // EPD_1in54_V2_test();  
    // EPD_1in54b_test();  
    // EPD_1in54c_test();  
  
    EPD_2in7_test();  
    // EPD_2in7b_test();  
  
    // EPD_2in9_test();  
    // EPD_2in9bc_test();  
    // EPD_2in9d_test();  
  
    // EPD_2in13_test();  
    // EPD_2in13_V2_test();  
    // EPD_2in13bc_test();  
    // EPD_2in13d_test();  
  
    // EPD_4in2_test();  
    // EPD_4in2bc_test();  
  
    // EPD_5in83_test();  
    // EPD_5in83bc_test();  
  
    // EPD_7in5_test();  
    // EPD_7in5_V2_test();  
    // EPD_7in5bc_test();  
    // EPD_7in5bc_V2_test();  
  
    return 0;  
}
```

- Compile codes

```
sudo make clean  
sudo make
```

- Try to run the example

```
sudo ./epd
```

Supports type

1.02inch (128×80) :

`EPD_1in02d_test()`: Example for 1.02inch e-Paper/1.02inch e-Paper Module

1.54inch (1.54inch e-paper c: 152×152, others: 200×200) :

EPD_1in54_test(): Example for 1.54inch e-paper V1 (Black/White) : This version is stop production which can be bought before 2019-11-22;

EPD_1in54_V2_test(): Example for 1.54inch e-paper V2 (Black/White): This is the current version which can be buy now (2020-07-29). The e-Paper has V2 sticker on the backside.

EPD_1in54b_test(): Example for 1.54inch e-paper B (Black/White/Red) ;

EPD_1in54c_test(): Example for 1.54inch e-paper C (Black/White/Red) ;br />

2.7inch (264×176) :

EPD_2in7_test(): Example for 2.7inch e-paper (Black/White) ;

EPD_2in7b_test(): Example for 2.7inch e-paper B (Black.White/Red) ;<be />

2.9inch (296×128) :

EPD_2in9_test(): Example for 2.9inch e-paper (Black/White) ;

EPD_2in9bc_test(): Example for 2.9inch e-paper B (Black/White/Red) and 2.9inch e-paper C (Black/White/Yellow) ;

EPD_2in9d_test(): Example for 2.9inch e-paper D (Black/White) ;

2.13inch (2.13inch e-Paper: 250×122, others: 212×104) :

EPD_2in13_test(): Example for 2.13inch e-paper V1 (Black/White) , this version is stop production and it can be bought before 019-05-15;

EPD_2in13_V2_test(): Example for 2.13inch e-paper V2 (Black/White) This is the current version with sticker V2 on the backside (2020-07-29);

EPD_2in13bc_test(): Example for 2.13inch e-paper B (Black/White/Red) and 2.13inch e-paper C (Blackj/White/Yellow) ;

EPD_2in13d_test(): Example for 2.13inch e-paper D (Black/White) ;

4.01inch (640x400)

EPD_4in01_test(): Example for the 4.01inch e-Paper HAT (F);

4.2inch (400×300)

EPD_4in2_test(): Example for 4.2inch e-paper (Black/White) ;

EPD_4in2bc_test(): Example for 4.2inch e-paper B (Black/White/Red) ;

5.65inch (600x448)

EPD_5in65f_test(): Example for for 5.65inch e-Paper F (Seven-color);

5.83inch (600×448) :

EPD_5in83_test(): Example for 5.83inch e-paper (Black/White) ;

EPD_5in83bc_test(): Example for 5.83inch e-paper B (Black/White/Red) and 5.83inch e-paper C (Black/White/Yellow) ;

7.5inch (V1: 640×384, V2: 800×480) :

EPD_7in5_test(): Example for 7.5inch e-paper (Black/White) , this version is stop production and it can be bought before 2019-12-07;

EPD_7in5bc_test(): Example for 7.5inch e-paper B (Black/White/Red) and 7.5inch e-paper C (Black/White/Yellow) , 7.5inch e-paper B V1 version is stop production and it can be bought before 2019-12-07;

EPD_7in5_V2_test(): Example for 7.5inch e-paper V2 (Black/White) , This is the current version with V2 sticker on the backside (2020-07-29)

EPD_7in5bc_V2_test(): Example for 7.5inch e-paper B V2 (Black/White/Red) ; This is the current version with V2 sticker on the backside. (2020-07-29);

Python

- Enter the folder of python code

```
cd ~/e-Paper/RaspberryPi\&JetsonNano/
cd python/examples
```

- Check the folder, you can see that there are .py files for different e-Paper.

```
ls -al
```



```

pi@raspberrypi:~/e-paper/RaspberryPi&JetsonNano/python/examples $ ls -al
total 100
drwx----- 2 pi pi 4096 Oct 12 16:20 .
drwx----- 5 pi pi 4096 Oct 12 16:20 ..
-rwx----- 1 pi pi 2994 Oct  8 15:00 epd_lin02_test.py
-rwx----- 1 pi pi 2801 Jul 24 19:14 epd_lin54b_test.py
-rwx----- 1 pi pi 2657 Jul 24 19:14 epd_lin54c_test.py
-rwx----- 1 pi pi 2976 Jul 24 19:14 epd_lin54_test.py
-rwx----- 1 pi pi 2969 Jul 24 19:14 epd_lin54_V2_test.py
-rwx----- 1 pi pi 3756 Jul 24 19:14 epd_2in13bc_test.py
-rwx----- 1 pi pi 3020 Jul 25 15:50 epd_2in13d_test.py
-rwx----- 1 pi pi 3026 Jul 24 19:14 epd_2in13_test.py
-rwx----- 1 pi pi 3097 Jul 24 19:14 epd_2in13_V2_test.py
-rwx----- 1 pi pi 4006 Jul 24 19:14 epd_2in7b_test.py
-rwx----- 1 pi pi 4331 Oct 10 18:35 epd_2in7_test.py
-rwx----- 1 pi pi 3851 Jul 24 19:14 epd_2in9bc_test.py
-rwx----- 1 pi pi 3760 Jul 25 15:53 epd_2in9d_test.py
-rwx----- 1 pi pi 3848 Dec 30 11:07 epd_2in9_test.py
-rwx----- 1 pi pi 3981 Jul 24 19:14 epd_4in2bc_test.py
-rwx----- 1 pi pi 3147 Jul 24 19:14 epd_4in2_test.py
-rwx----- 1 pi pi 3859 Jul 24 19:14 epd_5in83bc_test.py
-rwx----- 1 pi pi 3154 Jul 24 19:14 epd_5in83_test.py
-rwx----- 1 pi pi 3981 Jul 24 19:14 epd_7in5bc_test.py
-rwx----- 1 pi pi 3899 Sep 30 18:51 epd_7in5b_V2_test.py
-rwx----- 1 pi pi 3147 Jul 24 19:14 epd_7in5_test.py
-rwx----- 1 pi pi 3131 Sep 30 11:05 epd_7in5_V2_test.py

```

- Run the responding example

```

sudo python epd_xxx_test.py
sudo python3 epd_xxx_test.py

```

Supports type

1.02inch (128×80) :

epd_1in02_test.py: Example for 1.02inch e-Paper/1.02inch e-Paper Module

1.54inch (1.54inch e-paper c: 152×152, others: 200×200) :

epd_1in54_test.py: Example for 1.54inch e-paper V1 (Balck/White) : This version is stop production which can be bought before 2019-11-22;

epd_1in54_V2_test.py: Example for 1.54inch e-paper V2 (Balck/White): This is the current version which can be buy now (2020-07-29). The e-Paper has V2 sticker on the backside.

epd_1in54b_test.py: Example for 1.54inch e-paper B (Black/White/Red) ;

epd_1in54c_test.py: Example for 1.54inch e-paper C (Black/White/Red) ;br />

2.7inch (264×176) :

epd_2in7_test.py: Example for 2.7inch e-paper (Black/White) ;

epd_2in7b_test.py: Example for 2.7inch e-paper B (Black.White/Red) ;<be />

2.9inch (296×128) :

epd_2in9_test.py: Example for 2.9inch e-paper (Black/White) ;

epd_2in9bc_test.py: Example for 2.9inch e-paper B (Balck/White/Red) and 2.9inch e-paper C (Black/White/Yellow) ;

epd_2in9d_test.py: Example for 2.9inch e-paper D (Black/White) ;

2.13inch (2.13inch e-Paper: 250×122, others: 212×104) :

epd_2in13_test.py: Example for 2.13inch e-paper V1 (Black/White) , this version is stop production and it can be bought before 019-05-15;

epd_2in13_V2_test.py: Example for 2.13inch e-paper V2 (Black/White) This is the current version with sticker V2 on the backside (2020-07-29);

epd_2in13bc_test.py: Example for 2.13inch e-paper B (Black/White/Red) and 2.13inch e-paper C (Blackj/White /Yellow) ;

epd_2in13d_test.py: Example for 2.13inch e-paper D (Black/White) ;

4.01inch (640x400)

epd_4in01f_test.py: Example for 4.01inch e-Paper (Seven-color);

4.2inch (400×300)

epd_4in2_test.py: Example for 4.2inch e-paper (Black/White) ;

epd_4in2bc_test.py: Example for 4.2inch e-paper B (Black/White/Red) ;

5.65inch (600x448)

epd_5in65f_test.py: Example for 5.65inch e-Paper F (Seven-color);

5.83inch (600×448) :

epd_5in83_test.py: Example for 5.83inch e-paper (Black/White) ;

epd_5in83bc_test.py: Example for 5.83inch e-paper B (Black/White/Red) and 5.83inch e-paper C (Black/White/Yellow) ;

7.5inch (V1: 640×384, V2: 800×480) :

epd_7in5_test.py: Example for 7.5inch e-paper (Black/White) , this version is stop production and it can be bought before 2019-12-07;

epd_7in5bc_test.py: Example for 7.5inch e-paper B (Black/White/Red) and 7.5inch e-paper C (Black/White/Yellow) , 7.5inch e-paper B V1 version is stop production and it can be bought before 2019-12-07;

epd_7in5_V2_test.py: Example for 7.5inch e-paper V2 (Black/White) , This is the current version with V2 sticker on the backside (2020-07-29)

epd_7in5bc_V2_test.py: Example for 7.5inch e-paper B V2 (Black/White/Red) ; This is the current version with V2 sticker on the backside. (2020-07-29);

Description of codes (API)

The libraries for Raspberry Pi and Jetson Nano are same. Examples contain three parts, hardware interface, EPD driver and the GUI functions.

C

Hardware interface

Two libraries are used by C example, WiringPi and BCM2835. The codes use wiringPi by default, if you want to use BCM2835, you can modify RaspberryPi&JetsonNano\c\Makefile file, modify lines 13 and 14. Change it as below:

```
13 USELIB = USE_BCM2835_LIB
14 # USELIB = USE_WIRINGPI_LIB
15 DEBUG = -D $(USELIB)
16 ifeq ($(USELIB), USE_BCM2835_LIB)
17     LIB = -lbcm2835 -lm
18 else ifeq ($(USELIB), USE_WIRINGPI_LIB)
19     LIB = -lwiringPi -lm
20 endif
```

▪ Data type

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

▪ Init and Exit

```
void DEV_Module_Init(void);
void DEV_Module_Exit(void);
```

Note: The Init() and Exit() function are used to configure GPIOs . EPD enter sleep mode after Exit() function is used, and the consumption of e-Paper should be 0 in sleep mode if the PCB is Rev2.1 version.

▪ GPIO Read/Write

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

▪ SPI transmit data

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

EPD driver

The driver file are saved under RaspberryPi&JetsonNano\c\lib\e-Paper



▪ Initial EPD

For 1.54inch e-Paper, 1.54inch e-Paper V2, 2.13inch e-Paper, 2.13inch e-Paper V2, 2.13inch e-Paper (D), 2.9inch e-Paper and 2.9inch e-Paper (D), partial refresh is supported. You can set Mode = 0 for full refresh and Mode =1 for partial refresh

```
void EPD_xxx_Init(UBYTE Mode);
```

For other e-Paper

```
void EPD_xxx_Init(void);
```

- Transmite one frame of e-Paper and display

For black/white e-Paper

```
void EPD_xxx_Display(UBYTE *Image);
```

For three-color e-Paper

```
void EPD_xxx_Display(const UBYTE *blackimage, const UBYTE *ryimage);
```

There are exception:

For 2.13inch e-Paper (D) and 2.13inch e-Paper (D), if you want to do partial refresh, you should use function

```
void EPD_2IN13D_DisplayPart(UBYTE *Image);
void EPD_2IN9D_DisplayPart(UBYTE *Image);
```

For 1.54inch e-Paper V2 and 2.13inch e-Paper V2. you should first display static background (base image) and then dynamicly display (display Part) when partial refresh.

```
void EPD_1IN54_V2_DisplayPartBaseImage(UBYTE *Image);
void EPD_1IN54_V2_DisplayPart(UBYTE *Image);
void EPD_2IN13_V2_DisplayPart(UBYTE *Image);
void EPD_2IN13_V2_DisplayPartBaseImage(UBYTE *Image);
```





- Sleep mode

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









To wake up module, you should set hardware reset (re power on it) or call the init function.

GUI functions

GUI files can be found in RaspberryPi&JetsonNano\c\lib\GUI\GUI_Paint.c(h) directory

 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 can be found in RaspberryPi&JetsonNano\c\lib\Fonts directory

 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 an image buffer

```
void Paint_NewImage(UBYTE *image, UWORD Width, UWORD Height, UWORD Rotate, UWORD Color)
```

- Image: the Image buffer
- Width: width of the image
- Height: Height of the image
- Rotate: Rotate angle
- Color: Color of the image

Select image buffer

```
void Paint_SelectImage(UBYTE *image)
```

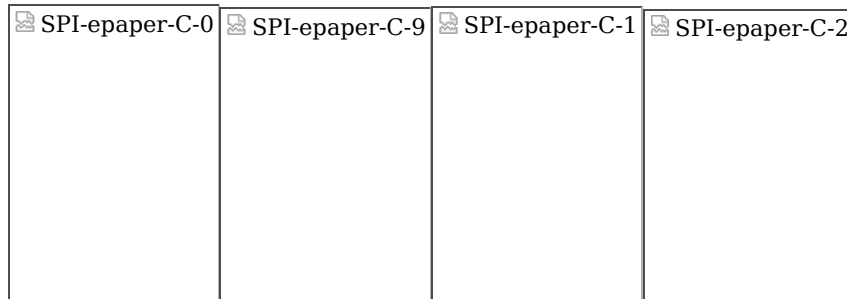
- The image buffer, it is a pointer of image buffer's first address

Rotate image

This function should be used after Paint_SelectImage()

```
void Paint_SetRotate(WORD Rotate)
```

- Rotate: The angle rotated. It should be ROTATE_0, ROTATE_90, ROTATE_180, ROTATE_270
- Note: For different orientation, the position of the first pixel is different, here we take 1.54inch as example



Mirroring

```
void Paint_SetMirroring(UBYTE mirror)
```

- mirror: The type of mirroring. (MIRROR_NONE, MIRROR_HORIZONTAL, MIRROR_VERTICAL, MIRROR_ORIGIN)

Set Pixel

This function is used to set the position and types of the pixel

```
void Paint_SetPixel(WORD Xpoint, WORD Ypoint, WORD Color)
```

- Xpoint: The x-axis coordination of pixel
- Ypoint: The y-axis coordination of pixel
- Color: The color of the pixel

Clear

This function is used to clear the e-Paper

```
void Paint_Clear(WORD Color)
```

- Color: The color of the display

Clear window

This function is used to clear a partial area

```
void Paint_ClearWindows(WORD Xstart, WORD Ystart, WORD Xend, WORD Yend, WORD Color)
```

- Xstart: The x-axis coordination of the start point
- Ystart: The y-axis coordination of the start point
- Xend: The x-axis coordination of the end point
- Yend: The y-axis coordination of the end point
- Color: The color of the windows

Draw point

This function is used to draw point.

```
void Paint_DrawPoint(WORD Xpoint, WORD Ypoint, WORD Color, DOT_PIXEL Dot_Pixel, DOT_STYLE Dot_Style)
```

- Xpoint: The x-axis coordination of point
- Ypoint: The y-axis coordination of point
- Dot_Pixel: The size of the point

```
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: The style of the point

```

typedef enum {
    DOT_FILL_AROUND = 1,
    DOT_FILL_RIGHTUP,
} DOT_STYLE;

```

Drawn Line

```
void Paint_DrawLine(UWORD Xstart, UWORD Ystart, UWORD Xend, UWORD Yend, UWORD Color, LINE_STYLE Line_Style , LINE_STYLE Line_Style)
```

This function is used to draw a line

- Xstart: The start x-axis coordination of the line
- Ystart: The start y-axis coordination of the line
- Xend: The end x-axis coordination of the line
- Yend: The end y-axis coordination of the line
- Line_width: The width of the line

```

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: The style of the line

```

typedef enum {
    LINE_STYLE_SOLID = 0,
    LINE_STYLE_DOTTED,
} LINE_STYLE;

```

Draw rectangle

Draw a rectangle from (Xstart, Ystart) to (Xend, Yend).

```
void Paint_DrawRectangle(UWORD Xstart, UWORD Ystart, UWORD Xend, UWORD Yend, UWORD Color, DOT_PIXEL Line_width, DRAW_FILL Draw_Fill)
```

- Xstart: Start coordinate of X-axes of the rectangle
- Ystart: Start coordinate of Y-axes of the rectangle
- Xend: End coordinate of X-end of the rectangle
- Yend: End coordinate of Y-end of the rectangle
- Color: color of the rectangle
- Line_width: The width of edges, 8 sides 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: set the rectangle full or empty.

```

typedef enum {
    DRAW_FILL_EMPTY = 0,
    DRAW_FILL_FULL,
} DRAW_FILL;

```

Draw character (ASCII)

Set(Xstart Ystart) as left-top point, draw a ASCII character.

```
void Paint_DrawChar(UWORD Xstart, UWORD Ystart, const char Ascii_Char, sFONT* Font, UWORD Color_Foreground, UWORD Color_Background)
```

Parameter:

- Xstart: X coordinate of the left-top pixel of character;
- Ystart: Y coordinate of the left-top pixel of character;
- Ascii_Char: Ascii character;
- Font: 5 fonts are available;
 - font12: 7*12
 - font16: 11*16
 - font20: 14*20
 - font24: 17*24
- Color_Foreground: color of character;
- Color_Background: color of background;

Draw String

Set point (Xstart Ystart) as the left-top pixel, draw a string.

```
void Paint_DrawString_EN(UWORD Xstart, UWORD Ystart, const char * pString, sFONT* Font, UWORD Color_Foreground, UWORD Color_Background)
```

Parameters:

- Xstart: X coordinate of left-top pixel of characters;
- Ystart: Y coordinate of left-top pixel of characters;
- pString: Pointer of string
- Font: 5 fonts are available:
 - font8: 5*8
 - font12: 7*12
 - font16: 11*16
 - font20: 14*20
 - font24: 17*24
- Color_Foreground: color of string
- Color_Background: color of the background

Draw Chinese characters

this function is used to draw Chinese fonts based ON GB2312 fonts.

```
void Paint_DrawString_CN(UWORD Xstart, UWORD Ystart, const char * pString, cFONT* font, UWORD Color_Foreground, UWORD Color_Background)
```

Parameters:

- Xstart: Coordinate of left-top pixel of characters;
- Ystart: Coordinate of left-top pixel of characters;
- pString: Pointer of string;
- Font: GB2312 fonts:
 - font12CN: 11*21(ascii), 16*21 (Chinese)
 - font24CN: 24*41(ascii), 32*41 (Chinese)
- Color_Foreground: color of string
- Color_Background: color of the background

Draw number

Draw a string of numbers, (Xstart, Ystart) is the left-top pixel.

```
void Paint_DrawNum(UWORD Xpoint, UWORD Ypoint, int32_t Nummber, sFONT* Font, UWORD Color_Foreground, UWORD Color_Background)
```

Parameter:

- Xstart: X coordinate of left-top pixel;
- Ystart: Y coordinate of left-top pixel;
- Nummber: the numbers displayed. the numbers are saved in int format, the maximum is 2147483647;
- Font: 5 fonts are available:
 - font8: 5*8
 - font12: 7*12
 - font16: 11*16
 - font20: 14*20
 - font24: 17*24
- Color_Foreground: color of font;
- Color_Background: color of background;

Draw image

Send image data of BMP file to buffer

```
void Paint_DrawBitMap(const unsigned char* image_buffer)
```

Parameters:

- image_buffer: address of image data in buffer

Read local bmp picture and write it to buffer

Linux platform like Jetson Nano and Raspberry Pi support to directly operate bmp pictures Raspberry Pi & Jetson Nano: RaspberryPi&JetsonNano\c\lib\GUI\GUI_BMPfile.c(.h)

```
JUBYTE GUI_ReadBmp(const char *path, UWORD Xstart, UWORD Ystart)
```

Parameters:

- path: The path of BMP pictures
- Xstart: X coordination of left-top of picture, default 0;
- Ystart: Y coordination of left-top of picture, default 0;

Testing Code

In the above part, we describe the tree structures of Linux codes, here we talk about the testing code for user. Raspberry Pi & Jetson Nano: RaspberryPi&JetsonNano\examples. The codes in examples are testing code, you can modify the definition in main.c file for different types of e-Paper. For example, if you want to test 2.13inch e-paper, you need to delete the "//" symbol on line 32. Use 5.65inch e-Paper as example, you need to change the line:

```
//EPD_5in65f_test();
```

to

```
EPD_5in65f_test();
```

Then compile it again and run

```
make clean
make
sudo ./epd
```

Python (Can be used for Jetson nano and Raspberry Pi)

It is compatible with python2.7 and python3

python is easy to use than c codes

Raspberry Pi and Jetson Nano: RaspberryPi&JetsonNano\python\lib\

epdconfigu.py

- Initialize module and exit handle:

```
def module_init()
def module_exit()
```

Note:

1. The functions are used to set GPIP before and after driving e-Paper

2. If the board you have is printed with Rev2.1, module enter low-ultra mode after Module_Exit(). (as we test, the current is about 0 in this mode);

- GPIO Read/Write:

```
def digital_write(pin, value)
def digital_read(pin)
```

- SPI Write data:

```
def spi_writebyte(data)
```

epdxxx.py(xxx is the type of e-Paper)

- Initailize e-paper: this function should be used at the begining. It can also be used to wake up e-Paper from Sleep mode.

```
#For 2.13inch e-Paper, 2.9inch e-Paper
def init(self, update) # Choose lut_full_update or lut_partial_update
#Other type
def init(self)
```

- Clear e-paper: This function is used to clear e-Paper to white;

```
def Clear(self)
def Clear(self, color) # Some types of e-Paper should use this function to clear screen
```

- Convert image to arrays

```
def getbuffer(self, image)
```

- Transmit one frame of image data and display

```
#For two-color e-paper
def display(self, image)

#Because that controllers of 2.13inch e-paper are updated, when partial refresh, they should first use displayPartBaseImage() to display static background
def displayPartBaseImage(self, image)
def displayPart(self, image)
```

- Enter sleep mode

```
def sleep(self)
```

epd_xxx_test.py(xxx is type of e-paper)

python examples are saved in directory:

Raspberry Pi和Jetson Nano: RaspberryPi&JetsonNano\python\examples\

If the python installed in your OS is python2, you should run examples like below:

```
sudo python epd_2in13_V2_test.py
```

If it is python3, the commands should be:

```
sudo python3 epd_2in13-V2_test.py
```

Orientation

To rotate the display, you can use transpose function like `blackimage = blackimage.transpose(Image.ROTATE_270)`:

```
blackimage = blackimage.transpose(Image.ROTATE_270)
redimage = redimage.transpose(Image.ROTATE_270)
#Support ROTATE_90, ROTATE_180, ROTATE_270
```

GUI

Python has a powerful PIL library (<http://effbot.org/imagingbook>), which can be used directly to drawing figures. Here we use it for drawing

- Install the library firstly

```
sudo apt-get install python3-pil
```

Import the library

```
from PIL import Image, ImageDraw, ImageFont
```

Image: library; ImageDraw: drawing function; ImageFont: fonts

- Set image buffer for drawing.

```
image = Image.new('1', (epd.width, epd.height), 255) # 255: clear the frame
```

The first parameter is the depth of color, 1 means 2 grayscale. The second parameter is a tuple of image size. The third parameter is color of the image, 0 is black and 255 is white.

- Create an image object.


```
draw = ImageDraw.Draw(image)
```

- Draw rectangle

```
draw.rectangle((0, 10, 200, 34), fill = 0)
```

The first parameter is a tuple of coordination. 0, 10 is the top-left point of rectangle, 200, 34) is the right-bottom point. fille = 0 set the filled color to black.

- Draw line

```
draw.line((16, 60, 56, 60), fill = 0)
```

The first parameter is a type of coordination, 16, 60 is the begin point, 200, 34 is the end point. fill=0 set the line to black

- Draw circle

```
draw.arc((90, 60, 150, 120), 0, 360, fill = 0)
```

This function is used to draw a encircle of a square. The first parameter is a tuple of coordination of the square. the degree of the circle is 0 to 360 °, fille=0 set the circle to black.

If the figure is not square according to the coordination, you will get an ellipse.。

Besides the arc function, you can also use the chord function for drawing solid circle.

```
draw.chord((90, 130, 150, 190), 0, 360, fill = 0)
```

The first parameter is the coordination of the enclosing rectangle. The second and third parameters are the beginning and end degrees of the circle. The fourth parameter is the fill color of the circle.

- Character

You can directkt import ImageFont model for drawing characters:

```
font = ImageFont.truetype(os.path.join(picdir, 'Font.ttc'), 24)
```

You can use the fonts of Windows or other fonts which is in ttc format.

To draw English character, you can directly use the fonts; for Chinese character, you need to add a symbol u:

```
draw.text((8, 12), 'hello world', font = font, fill = 255)
draw.text((8, 36), u'电子墨水屏', font = font, fill = 0)
```

The first parameter is a tuple of coordination of character, the second parameter is the font and las one is set the color.

- Read local picture

```
image = Image.open(os.path.join(picdir, '2in13-v2.bmp'))
```

The parameter is the path of picture.

- Other functions.

For more information about the PIL library, please refer to <http://effbot.org/imagingbook>.

Hardware connection

Because the 40PIN GPIO of Jetson Nano are compatible with Raspberry Pi's and the Jetson.GPIO libraries are compatible with RPi.GPIO, therefore we use the same connection as Raspberry Pi
To connect the e-Paper, you can following the table below

Jetson Nano Connection

e-Paper	Jetson Nano Developer Kit	
	BCM2835	Board
VCC	3.3V	3.3V
GND	GND	GND
DIN	10(SPI0_MOSI)	19
CLK	11(SPI0_SCK	23
CS	8(SPI0_CS0)	24

DC	25	22
RST	17	11
BUSY	24	18

Install libraries

Open terminal of Jetson Nano and run the following commands to install corresponding libraries:

```
sudo apt-get update
sudo apt-get install python3-pip
sudo pip3 install Jetson.GPIO
sudo groupadd -f -r gpio
sudo usermod -a -G gpio your_user_name
sudo cp /opt/nvidia/jetson-gpio/etc/99-gpio.rules /etc/udev/rules.d/
sudo udevadm control --reload-rules && sudo udevadm trigger
```

- Install I2C libraries

```
sudo apt-get install python-smbus
```

- Install PIL libraries

```
sudo apt-get install python3-pil
sudo apt-get install python3-numpy
```

Download demo codes

Open the terminal of Jetson Nano and clone demo codes by the following commands:

```
sudo git clone https://github.com/waveshare/e-Paper
```

Examples

C

- Enter the folder of C examples

```
cd ~/e-Paper/RaspberryPi\&JetsonNano/
cd c
```

- Modify the main.c file for corresponding e-Paper

```
sudo nano examples/main.c
```

For example, if you want to update a 2.7inch e-Paper/2.7inch e-Paper HAT, you should modify the main.c file, uncomment the line `EPD_2in7_test()`, comment others, and save.

```

int main(void)
{
    // Exception handling:ctrl + c
    signal(SIGINT, Handler);

    // EPD_1in02d_test();

    // EPD_1in54_test();
    // EPD_1in54_V2_test();
    // EPD_1in54b_test();
    // EPD_1in54c_test();

    EPD_2in7_test();
    // EPD_2in7b_test();

    // EPD_2in9_test();
    // EPD_2in9bc_test();
    // EPD_2in9d_test();

    // EPD_2in13_test();
    // EPD_2in13_V2_test();
    // EPD_2in13bc_test();
    // EPD_2in13d_test();

    // EPD_4in2_test();
    // EPD_4in2bc_test();

    // EPD_5in83_test();
    // EPD_5in83bc_test();

    // EPD_7in5_test();
    // EPD_7in5_V2_test();
    // EPD_7in5bc_test();
    // EPD_7in5bc_V2_test();

    return 0;
}

```

- Compile codes

```

sudo make clean
sudo make

```

- Try to run the example

```

sudo ./epd

```

Supports type

1.02inch (128×80) :

EPD_1in02d_test(): Example for 1.02inch e-Paper/1.02inch e-Paper Module

1.54inch (1.54inch e-paper c: 152×152, others: 200×200) :

EPD_1in54_test(): Example for 1.54inch e-paper V1 (Black/White) : This version is stop production which can be bought before 2019-11-22;

EPD_1in54_V2_test(): Example for 1.54inch e-paper V2 (Black/White): This is the current version which can be buy now (2020-07-29). The e-Paper has V2 sticker on the backside.

EPD_1in54b_test(): Example for 1.54inch e-paper B (Black/White/Red) ;

EPD_1in54c_test(): Example for 1.54inch e-paper C (Black/White/Red) ;br />

2.7inch (264×176) :

EPD_2in7_test(): Example for 2.7inch e-paper (Black/White) ;

EPD_2in7b_test(): Example for 2.7inch e-paper B (Black.White/Red) ;<be />

2.9inch (296×128) :

EPD_2in9_test(): Example for 2.9inch e-paper (Black/White) ;

EPD_2in9bc_test(): Example for 2.9inch e-paper B (Black/White/Red) and 2.9inch e-paper C (Black/White/Yellow) ;

EPD_2in9d_test(): Example for 2.9inch e-paper D (Black/White) ;

2.13inch (2.13inch e-Paper: 250×122, others: 212×104) :

EPD_2in13_test(): Example for 2.13inch e-paper V1 (Black/White) , this version is stop production and it can be bought before 019-05-15;

EPD_2in13_V2_test(): Example for 2.13inch e-paper V2 (Black/White) This is the current version with sticker V2 on the backside (2020-07-29);

EPD_2in13bc_test(): Example for 2.13inch e-paper B (Black/White/Red) and 2.13inch e-paper C (Black/White/Yellow) ;

EPD_2in13d_test(): Example for 2.13inch e-paper D (Black/White) ;

4.01inch (640×400)

EPD_4in01f_test(): Example for 4.01inch e-paper HAT (F) (Seven-color) ;

4.2inch (400×300)

EPD_4in2_test(): Example for 4.2inch e-paper (Black/White) ;

EPD_4in2bc_test(): Example for 4.2inch e-paper B (Black/White/Red) ;

5.65inch (600×448)

EPD_5in65f_test(): Example for for 5.65inch e-Paper F (Seven-color);

5.83inch (600×448) :

EPD_5in83_test(): Example for 5.83inch e-paper (Black/White) ;

EPD_5in83bc_test(): Example for 5.83inch e-paper B (Black/White/Red) and 5.83inch e-paper C (Black/White/Yellow) ;

7.5inch (V1: 640×384, V2: 800×480) :

EPD_7in5_test(): Example for 7.5inch e-paper (Black/White) , this version is stop production and it can be bought before 2019-12-07;

EPD_7in5bc_test(): Example for 7.5inch e-paper B (Black/White/Red) and 7.5inch e-paper C (Black/White/Yellow) , 7.5inch e-paper B V1 version is stop production and it can be bought before 2019-12-07;

EPD_7in5_V2_test(): Example for 7.5inch e-paper V2 (Black/White) , This is the current version with V2 sticker on the backside (2020-07-29)

EPD_7in5bc_V2_test(): Example for 7.5inch e-paper B V2 (Black/White/Red) ; This is the current version with V2 sticker on the backside. (2020-07-29);

Python

- Enter the folder of python code

```
cd ~/e-Paper/RaspberryPi&JetsonNano/
cd python/examples
```

- Check the folder, you can see that there are .py files for different e-Paper.

```
ls -al
```

```
pi@raspberrypi:~/e-paper/RaspberryPi&JetsonNano/python/examples $ ls -al
total 100
drwx----- 2 pi pi 4096 Oct 12 16:20 .
drwx----- 5 pi pi 4096 Oct 12 16:20 ..
-rwx----- 1 pi pi 2994 Oct  8 15:00 epd_1in02_test.py
-rwx----- 1 pi pi 2801 Jul 24 19:14 epd_1in54b_test.py
-rwx----- 1 pi pi 2657 Jul 24 19:14 epd_1in54c_test.py
-rwx----- 1 pi pi 2976 Jul 24 19:14 epd_1in54_test.py
-rwx----- 1 pi pi 2969 Jul 24 19:14 epd_1in54_V2_test.py
-rwx----- 1 pi pi 3756 Jul 24 19:14 epd_2in13bc_test.py
-rwx----- 1 pi pi 3020 Jul 25 15:50 epd_2in13d_test.py
-rwx----- 1 pi pi 3026 Jul 24 19:14 epd_2in13_test.py
-rwx----- 1 pi pi 3097 Jul 24 19:14 epd_2in13_V2_test.py
-rwx----- 1 pi pi 4006 Jul 24 19:14 epd_2in7b_test.py
-rwx----- 1 pi pi 4331 Oct 10 18:35 epd_2in7_test.py
-rwx----- 1 pi pi 3851 Jul 24 19:14 epd_2in9bc_test.py
-rwx----- 1 pi pi 3760 Jul 25 15:53 epd_2in9d_test.py
-rwx----- 1 pi pi 3848 Dec 30 11:07 epd_2in9_test.py
-rwx----- 1 pi pi 3981 Jul 24 19:14 epd_4in2bc_test.py
-rwx----- 1 pi pi 3147 Jul 24 19:14 epd_4in2_test.py
-rwx----- 1 pi pi 3859 Jul 24 19:14 epd_5in83bc_test.py
-rwx----- 1 pi pi 3154 Jul 24 19:14 epd_5in83_test.py
-rwx----- 1 pi pi 3981 Jul 24 19:14 epd_7in5bc_test.py
-rwx----- 1 pi pi 3899 Sep 30 18:51 epd_7in5b_V2_test.py
-rwx----- 1 pi pi 3147 Jul 24 19:14 epd_7in5_test.py
-rwx----- 1 pi pi 3131 Sep 30 11:05 epd_7in5_V2_test.py
```

- Run the responding example, for example.

```
sudo python epd_xxx_test.py
sudo python3 epd_xxx_test.py
```

Supports type

1.02inch (128×80) :

epd_1in02_test.py: Example for 1.02inch e-Paper/1.02inch e-Paper Module

1.54inch (1.54inch e-paper c: 152×152, others: 200×200) :

epd_1in54_test.py: Example for 1.54inch e-paper V1 (Black/White) : This version is stopped production which can be bought before 2019-11-22;

epd_1in54_V2_test.py: Example for 1.54inch e-paper V2 (Black/White): This is the current version which can buy now (2020-07-29). The e-Paper has a V2 sticker on the backside.

epd_1in54b_test.py: Example for 1.54inch e-paper B (Black/White/Red) ;

epd_1in54c_test.py: Example for 1.54inch e-paper C (Black/White/Red) ;

2.7inch (264×176) :

epd_2in7_test.py: Example for 2.7inch e-paper (Black/White) ;

epd_2in7b_test.py: Example for 2.7inch e-paper B (Black/White/Red) ;

2.9inch (296×128) :

epd_2in9_test.py: Example for 2.9inch e-paper (Black/White) ;

epd_2in9bc_test.py: Example for 2.9inch e-paper B (Black/White/Red) and 2.9inch e-paper C (Black/White/Yellow) ;

epd_2in9d_test.py: Example for 2.9inch e-paper D (Black/White) ;

2.13inch (2.13inch e-Paper: 250×122, others: 212×104) :

epd_2in13_test.py: Example for 2.13inch e-paper V1 (Black/White) , this version is stopped production and it can be bought before 019-05-15;

epd_2in13_V2_test.py: Example for 2.13inch e-paper V2 (Black/White) This is the current version with sticker V2 on the backside (2020-07-29);

epd_2in13bc_test.py: Example for 2.13inch e-paper B (Black/White/Red) and 2.13inch e-paper C (Black/White/Yellow) ;

epd_2in13d_test.py: Example for 2.13inch e-paper D (Black/White) ;

4.01inch (640x400):

epd_4in01f_test.py: Example for 4.01inch e-paper HAT (F) (Seven-color) ;

4.2inch (400×300)

epd_4in2_test.py: Example for 4.2inch e-paper (Black/White) ;

epd_4in2bc_test.py: Example for 4.2inch e-paper B (Black/White/Red) ;

5.65inch (600x448)

epd_5in65f_test.py: Example for 5.65inch e-Paper F (Seven-color);

5.83inch (600×448) :

epd_5in83_test.py: Example for 5.83inch e-paper (Black/White) ;

epd_5in83bc_test.py: Example for 5.83inch e-paper B (Black/White/Red) and 5.83inch e-paper C (Black/White/Yellow) ;

7.5inch (V1: 640×384, V2: 800×480) :

epd_7in5_test.py: Example for 7.5inch e-paper (Black/White) , this version is stopped production and it can be bought before 2019-12-07;

epd_7in5bc_test.py: Example for 7.5inch e-paper B (Black/White/Red) and 7.5inch e-paper C (Black/White/Yellow) , 7.5inch e-paper B V1 version is stopped production and it can be bought before 2019-12-07;

epd_7in5_V2_test.py: Example for 7.5inch e-paper V2 (Black/White) , This is the current version with V2 sticker on the backside (2020-07-29)

epd_7in5bc_V2_test.py: Example for 7.5inch e-paper B V2 (Black/White/Red) ; This is the current version with V2 sticker on the backside. (2020-07-29);

Description of codes (API)

The libraries for Raspberry Pi and Jetson Nano are the same. Examples contain three parts, hardware interface, EPD driver, and the GUI functions.

C

Hardware interface

Two libraries are used by C example, WiringPi, and BCM2835. The codes use wiringPi by default, if you want to use BCM2835, you can modify the RaspberryPi&JetsonNano\c\Makefile file, modify lines 13 and 14. Change it as below:

```

13 USELIB = USE_BCM2835_LIB
14 # USELIB = USE_WIRINGPI_LIB
15 DEBUG = -D $(USELIB)
16 ifeq ($(USELIB), USE_BCM2835_LIB)
17     LIB = -lbcm2835 -lm
18 else ifeq ($(USELIB), USE_WIRINGPI_LIB)
19     LIB = -lwiringPi -lm
20 endif

```

▪ Data type

```

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

```

▪ Init and Exit

```

void DEV_Module_Init(void);
void DEV_Module_Exit(void);

```

Note: The Init() and Exit() function are used to configure GPIOs . EPD enter sleep mode after Exit() function is used, and the consumption of e-Paper should be 0 in sleep mode if the PCB is Rev2.1 version.

▪ GPIO Read/Write

```

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

```

▪ SPI transmit data

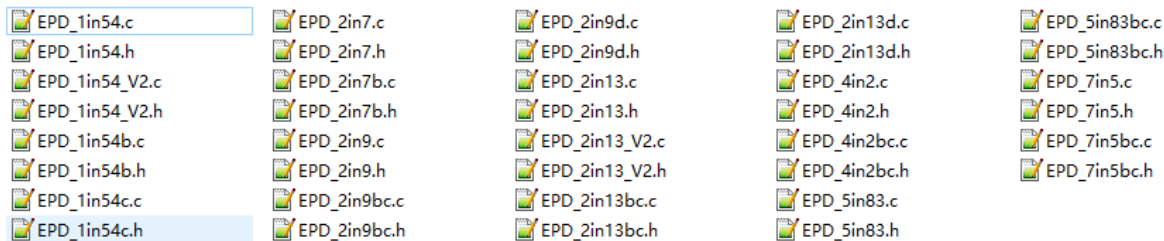
```

void DEV_SPI_WriteByte(UBYTE Value);

```

EPD driver

The driver file are saved under RaspberryPi&JetsonNano\c\lib\e-Paper



▪ Initial EPD

For 1.54inch e-Paper, 1.54inch e-Paper V2, 2.13inch e-Paper, 2.13inch e-Paper V2, 2.13inch e-Paper (D), 2.9inch e-Paper and 2.9inch e-Paper (D), partial refresh is supported. You can set Mode = 0 for full refresh and Mode =1 for partial refresh

```

void EPD_xxx_Init(UBYTE Mode);

```

For other e-Paper

```

void EPD_xxx_Init(void);

```

▪ Trasmite one frame of e-Paper and display

For black/white e-Paper

```

void EPD_xxx_Display(UBYTE *Image);

```

For three-color e-Paper

```

void EPD_xxx_Display(const UBYTE *blackimage, const UBYTE *ryimage);

```

There are exception:

For 2.13inch e-Paper (D) and 2.13inch e-Paper (D), if you want to do partial refresh, you should use function

```

void EPD_2IN13D_DisplayPart(UBYTE *Image);
void EPD_2IN9D_DisplayPart(UBYTE *Image);

```

For 1.54inch e-Paper V2 and 2.13inch e-Paper V2. you should first display static background (base image) and then dynamically display (display Part) when partial refresh.

```
void EPD_1IN54_V2_DisplayPartBaseImage(UBYTE *Image);
void EPD_1IN54_V2_DisplayPart(UBYTE *Image);
void EPD_2IN13_V2_DisplayPart(UBYTE *Image);
void EPD_2IN13_V2_DisplayPartBaseImage(UBYTE *Image);
```





- Sleep mode

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









To wake up module, you should set hardware reset (re power on it) or call the init function.

GUI functions

GUI files can be found in RaspberryPi&JetsonNano\c\lib\GUI\GUI_Paint.c(.h) directory

	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 can be found in RaspberryPi&JetsonNano\c\lib\Fonts directory

	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 an image buffer

```
void Paint_NewImage(UBYTE *image, UWORD Width, UWORD Height, UWORD Rotate, UWORD Color)
```

- Image: the Image buffer
- Width: width of the image
- Height: Height of the image
- Rotate: Rotate angle
- Color: Color of the image

Select image buffer

```
void Paint_SelectImage(UBYTE *image)
```

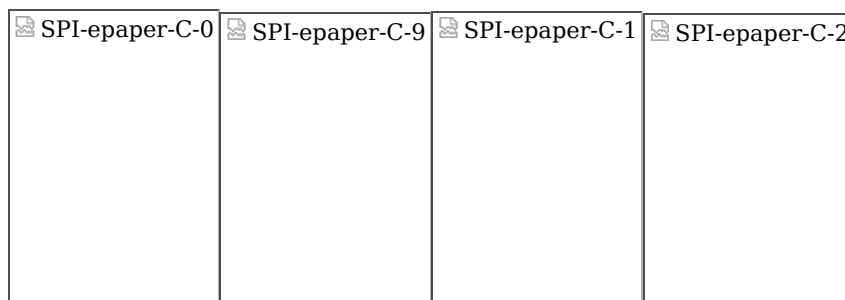
- The image buffer, it is a pointer of image buffer's first address

Rotate image

This function should be used after Paint_SelectImage()

```
void Paint_SetRotate(UWORD Rotate)
```

- Rotate: The angle rotated. It should be ROTATE_0, ROTATE_90, ROTATE_180, ROTATE_270
- Note: For different orientation, the position of the first pixel is different, here we take 1.54inch as example



Mirroring

```
void Paint_SetMirroring(UBYTE mirror)
```

- mirror: The type of mirroring. (MIRROR_NONE, MIRROR_HORIZONTAL, MIRROR_VERTICAL, MIRROR_ORIGIN)

Set Pixel

This function is used to set the position and types of the pixel

```
void Paint_SetPixel(UWORD Xpoint, UWORD Ypoint, UWORD Color)
```

- Xpoint: The x-axis coordination of pixel
- Ypoint: The y-axis coordination of pixel
- Color: The color of the pixel

Clear

This function is used to clear the e-Paper

```
void Paint_Clear(UWORD Color)
```

- Color: The color of the display

Clear window

This function is used to clear a partial area

```
void Paint_ClearWindows(UWORD Xstart, UWORD Ystart, UWORD Xend, UWORD Yend, UWORD Color)
```

- Xstart: The x-axis coordination of the start point
- Ystart: The y-axis coordination of the start point
- Xend: The x-axis coordination of the end point
- Yend: The y-axis coordination of the end point
- Color: The color of the windows

Draw point

This function is used to draw points.

```
void Paint_DrawPoint(UWORD Xpoint, UWORD Ypoint, UWORD Color, DOT_PIXEL Dot_Pixel, DOT_STYLE Dot_Style)
```

- Xpoint: The x-axis coordination of point
- Ypoint: The y-axis coordination of point
- Dot_Pixel: The size of the point

```
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: The style of the point

```
typedef enum {
    DOT_FILL_AROUND = 1,
    DOT_FILL_RIGHTUP,
} DOT_STYLE;
```

Drawn Line

```
void Paint_DrawLine(UWORD Xstart, UWORD Ystart, UWORD Xend, UWORD Yend, UWORD Color, LINE_STYLE Line_Style , LINE_STYLE Line_Style)
```

This function is used to draw a line

- Xstart: The start x-axis coordination of the line
- Ystart: The start y-axis coordination of the line
- Xend: The end x-axis coordination of the line
- Yend: The end y-axis coordination of the line
- Line_width: The width of the line

```
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:** The style of the line

```

typedef enum {
    LINE_STYLE_SOLID = 0,
    LINE_STYLE_DOTTED,
} LINE_STYLE;

```

Draw rectangle

Draw a rectangle from (Xstart, Ystart) to (Xend, Yend).

```
void Paint_DrawRectangle(UWORD Xstart, UWORD Ystart, UWORD Xend, UWORD Yend, UWORD Color, DOT_PIXEL Line_width, DRAW_FILL Draw_Fill)
```

- **Xstart:** Start coordinate of X-axes of the rectangle
- **Ystart:** Start coordinate of Y-axes of the rectangle
- **Xend:** End coordinate of X-end of the rectangle
- **Yend:** End coordinate of Y-end of the rectangle
- **Color:** color of the rectangle
- **Line_width:** The width of edges, 8 sides 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:** set the rectangle full or empty.

```

typedef enum {
    DRAW_FILL_EMPTY = 0,
    DRAW_FILL_FULL,
} DRAW_FILL;

```

Draw character (ASCII)

Set(Xstart Ystart) as left-top point, draw a ASCII character.

```
void Paint_DrawChar(UWORD Xstart, UWORD Ystart, const char Ascii_Char, sFONT* Font, UWORD Color_Foreground, UWORD Color_Background)
```

Parameter:

- **Xstart:** X coordinate of the left-top pixel of character;
- **Ystart:** Y coordinate of the left-top pixel of character;
- **Ascii_Char:** Ascii character;
- **Font:** 5 fonts are available;
 - font12: 7*12
 - font16: 11*16
 - font20: 14*20
 - font24: 17*24
- **Color_Foreground:** color of character;
- **Color_Background:** color of background;

Draw String

Set point (Xstart Ystart) as the left-top pixel, draw a string.

```
void Paint_DrawString_EN(UWORD Xstart, UWORD Ystart, const char * pString, sFONT* Font, UWORD Color_Foreground, UWORD Color_Background)
```

Parameters:

- **Xstart:** X coordinate of left-top pixel of characters;
- **Ystart:** Y coordinate of left-top pixel of characters;
- **pString:** Pointer of string
- **Font:** 5 fonts are available;
 - font8: 5*8
 - font12: 7*12

```
font16: 11*16
```

```
font20: 14*20
```

```
font24: 17*24
```

- Color_Foreground: color of string
- Color_Background: color of the background

Draw Chinese characters

this function is used to draw Chinese fonts based ON GB2312 fonts.

```
void Paint_DrawString_CN(UWORD Xstart, UWORD Ystart, const char * pString, cFONT* font, UWORD Color_Foreground, UWORD Color_Background)
```

Parameters:

- Xstart: Coordinate of left-top pixel of characters;
- Ystart: Coordinate of left-top pixel of characters;
- pString: Pointer of string;
- Font: GB2312 fonts:
 - font12CN: 11*21(ascii), 16*21 (Chinese)
 - font24CN: 24*41(ascii), 32*41 (Chinese)
- Color_Foreground: color of string
- Color_Background: color of the background

Draw number

Draw a string of numbers, (Xstart, Ystart) is the left-top pixel.

```
void Paint_DrawNum(UWORD Xpoint, UWORD Ypoint, int32_t Nummber, sFONT* Font, UWORD Color_Foreground, UWORD Color_Background)
```

Parameter:

- Xstart: X coordinate of left-top pixel;
- Ystart: Y coordicate of left-to pixel;
- Nummber: the numbers displayed. the numbers are saved in int format, the maximum is 2147483647;
- Font: 5 fonts are available:
 - font8: 5*8
 - font12: 7*12
 - font16: 11*16
 - font20: 14*20
 - font24: 17*24
- Color_Foreground: color of font;
- Color_Background: color of background;

Draw image

Send image data of BMP file to buffer

```
void Paint_DrawBitMap(const unsigned char* image_buffer)
```

Parameters:

- image_buffer: adrress of image data in buffer

Read local bmp picture and write it to buffer

Linux platform like Jetson Nano and Raspberry Pi support to directly operate bmp pictures Raspberry Pi & Jetson

Nano: RaspberryPi&JetsonNano\c\lib\GUI\GUI_BMPfile.c(.h)

```
U8BYTE GUI_ReadBmp(const char *path, UWORD Xstart, UWORD Ystart)
```

Parameters:

- path: The path of BMP pictures
- Xstart: X coordination of left-top of picture, default 0;
- Ystart: Y coordination of left-top of picture, default 0;

Testing Code

In the above part, we describe the tree structures of Linux codes, here we talk about the testing code for user. Raspberry Pi & Jetson Nano: RaspberryPi&JetsonNano\c\examples. The codes in examples are testing code, you can modify the definition in main.c file for different types of e-paper. For example, if you want to test the 2.13inch e-paper, you need to delete the "//" symbol on line 32. Use 5.65inch e-Paper as an example, you need to change the line:

```
//EPD_5in65f_test();
```

to

```
EPD_5in65f_test();
```

Then compile it again and run

```
make clean
make
sudo ./epd
```

Python (Can be used for Jetson nano and Raspberry Pi)

It is compatible with python2.7 and python3

python is easy to use than c codes

Raspberry Pi and Jetson Nano: RaspberryPi&JetsonNano\python\lib\

epdconfig.py

- Initialize module and exit handle:

```
def module_init()
def module_exit()
```

Note:

1. The functions are used to set GPIIP before and after driving e-Paper

2. If the board you have is printed with Rev2.1, the module enter low-ultra mode after Module_Exit(). (as we test, the current is about 0 in this mode);

- GPIO Read/Write:

```
def digital_write(pin, value)
def digital_read(pin)
```

- SPI Write data:

```
def spi_writebyte(data)
```

epdxxx.py(xxx is the type of e-Paper)

- Initailize e-paper: this function should be used at the beginning. It can also be used to wake up e-Paper from Sleep mode.

```
#For 2.13inch e-Paper、2.9inch e-Paper
def init(self, update) # Choose lut_full_update or lut_partial_update
#Other type
def init(self)
```

- Clear e-paper: This function is used to clear e-Paper to white;

```
def Clear(self)
def Clear(self, color) # Some types of e-Paper should use this function to clear screen
```

- Convert image to arrays

```
def getbuffer(self, image)
```

- Transmit one frame of image data and display

```
#For two-color e-paper
def display(self, image)
#Because that controllers of 2.13inch e-paper are updated, when partial refresh, they should first use displayPartBaseImage() to display static background
def displayPartBaseImage(self, image)
def displayPart(self, image)
```

- Enter sleep mode

```
def sleep(self)
```

epd_xxx_test.py(XXX is type of e-paper)

python examples are saved in directory:

Raspberry Pi and Jetson Nano: RaspberryPi&JetsonNano\python\examples\

If the python installed in your OS is python2, you should run examples like below:

```
sudo python epd_2in13_V2_test.py
```

If it is python3, the commands should be:

```
sudo python3 epd_2in13_V2_test.py
```

Orientation

To rotate the display, you can use transpose function like `blackimage = blackimage.transpose(Image.ROTATE_270)`:

```
blackimage = blackimage.transpose(Image.ROTATE_270)
redimage = redimage.transpose(Image.ROTATE_270)
#Support ROTATE_90, ROTATE_180, ROTATE_270
```

GUI

Python has a powerful PIL library (<http://effbot.org/imagingbook>), which can be used directly to drawing figures. Here we use it for drawing

- Install the library firstly

```
sudo apt-get install python3-pil
```

Import the library

```
from PIL import Image, ImageDraw, ImageFont
```

Image: library; ImageDraw: drawing function; ImageFont: fonts

- Set image buffer for drawing.

```
image = Image.new('1', (epd.width, epd.height), 255) # 255: clear the frame
```

The first parameter is the depth of color, 1 means 2 grayscale. The second parameter is a tuple of image size. The third parameter is color of the image, 0 is black and 255 is white.

- Create an image object.

```
draw = ImageDraw.Draw(image)
```

- Draw rectangle

```
draw.rectangle((0, 10, 200, 34), fill = 0)
```

The first parameter is a tuple of coordination. 0, 10 is the top-left point of rectangle, 200, 34) is the right-bottom point. `fill = 0` set the filled color to black.

- Draw line

```
draw.line((16, 60, 56, 60), fill = 0)
```

The first parameter is a type of coordination, 16, 60 is the beginning point, 200, 34 is the endpoint. `fill=0` set the line to black

- Draw circle

```
draw.arc((90, 60, 150, 120), 0, 360, fill = 0)
```

This function is used to draw a encircle of a square. The first parameter is a tuple of coordination of the square. the degree of the circle is 0 to 360 °, `fill=0` set the circle to black.

If the figure is not square according to the coordination, you will get an ellipse..

Besides the arc function, you can also use the chord function for drawing a solid circle.

```
draw.chord((90, 130, 150, 190), 0, 360, fill = 0)
```

The first parameter is the coordination of the enclosing rectangle. The second and third parameters are the beginning and end degrees of the circle. The fourth parameter is the fill color of the circle.

- Character

You can directkt import ImageFont model for drawing characters:

```
font = ImageFont.truetype(os.path.join(picdir, 'Font.ttc'), 24)
```

You can use the fonts of Windows or other fonts which is in ttc format.

To draw English character, you can directly use the fonts; for Chinese character, you need to add a symbol u:

```
draw.text((8, 12), 'hello world', font = font, fill = 255)
draw.text((8, 36), u'电子墨水屏', font = font, fill = 0)
```

The first parameter is a tuple of coordination of character, the second parameter is the font and las one is set the color.

- Read local picture

```
image = Image.open(os.path.join(picdir, '2in13-v2.bmp'))
```

The parameter is the path of picture.

- Other functions.


For more information about the PIL library, please refer to <http://effbot.org/imagingbook>.

Hardware connection

The demo codes we provide are based on STM32F103ZET6, the connection table is also based on STM32F103ZET6, if you want to use other chip, you need to port the codes yourself and change the connection according to actual situation.

Connect to STM32F103ZET

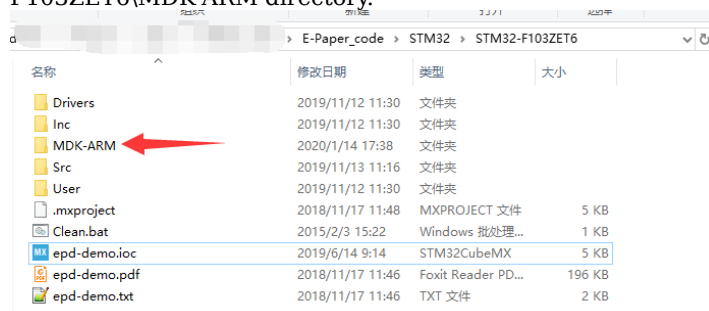
e-Paper	STM32
Vcc	3.3V
GND	GND
DIN	PA7
CLK	PA5
CS	PA4
DC	PA2
RST	PA1
BUSY	PA3

 E-paper driver hat to stm32 connet.png

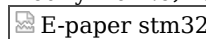
If the STM32 board you have is STM32F103RB, you can refer to the guides of E-Paper Shield

Softawre settings

The codes are based on HAL library. Download the codes and the project files are saved under the STM32\STM32-F103ZET6\MDK-ARM directory.



Modify main.c, define the line according to the e-paper type and re-compile project and download.

 E-paper stm32 code2.png

Supports type

1.02inch (128×80) :

EPD_1in02d_test(): Example for 1.02inch e-Paper/1.02inch e-Paper Module

1.54inch (1.54inch e-paper c: 152×152, others: 200×200) :

EPD_1in54_test(): Example for 1.54inch e-paper V1 (Black/White) : This version is stop production which can be bought before 2019-11-22;

EPD_1in54_V2_test(): Example for 1.54inch e-paper V2 (Black/White): This is the current version which can be buy now (2020-07-29). The e-Paper has V2 sticker on the backside.

EPD_1in54b_test(): Example for 1.54inch e-paper B (Black/White/Red) ;

EPD_1in54c_test(): Example for 1.54inch e-paper C (Black/White/Red) ;br />

2.7inch (264×176) :

EPD_2in7_test(): Example for 2.7inch e-paper (Black/White) ;

EPD_2in7b_test(): Example for 2.7inch e-paper B (Black.White/Red) ;

2.9inch (296×128) :

EPD_2in9_test(): Example for 2.9inch e-paper (Black/White) ;

EPD_2in9bc_test(): Example for 2.9inch e-paper B (Black/White/Red) and 2.9inch e-paper C (Black/White/Yellow) ;

EPD_2in9d_test(): Example for 2.9inch e-paper D (Black/White) ;

2.13inch (2.13inch e-Paper: 250×122, others: 212×104)

EPD_2in13_test(): Example for 2.13inch e-paper V1 (Black/White) , this version is stop production and it can be bought before 019-05-15;

EPD_2in13_V2_test(): Example for 2.13inch e-paper V2 (Black/White) This is the current version with sticker V2 on the backside (2020-07-29);

EPD_2in13bc_test(): Example for 2.13inch e-paper B (Black/White/Red) and 2.13inch e-paper C (Blackj/White/Yellow) ;

EPD_2in13d_test(): Example for 2.13inch e-paper D (Black/White) ;

4.01inch (640×400)

EPD_4in01f_test(): Example for 4.01inch e-paper HAT (F) (Seven-color) ;

4.2inch (400×300)

EPD_4in2_test(): Example for 4.2inch e-paper (Black/White) ;

EPD_4in2bc_test(): Example for 4.2inch e-paper B (Black/White/Red) ;

5.65inch (600x448)

EPD_5in65f_test(): Example for for 5.65inch e-Paper F (Seven-color);

5.83inch (600×448) :

EPD_5in83_test(): Example for 5.83inch e-paper (Black/White) ;

EPD_5in83bc_test(): Example for 5.83inch e-paper B (Black/White/Red) and 5.83inch e-paper C (Black/White/Yellow) ;

7.5inch (V1: 640×384, V2: 800×480) :

EPD_7in5_test(): Example for 7.5inch e-paper (Black/White) , this version is stop production and it can be bought before 2019-12-07;

EPD_7in5bc_test(): Example for 7.5inch e-paper B (Black/White/Red) and 7.5inch e-paper C (Black/White/Yellow) , 7.5inch e-paper B V1 version is stop production and it can be bought before 2019-12-07;

EPD_7in5_V2_test(): Example for 7.5inch e-paper V2 (Black/White) , This is the current version with V2 sticker on the backside (2020-07-29)

EPD_7in5bc_V2_test(): Example for 7.5inch e-paper B V2 (Black/White/Red) ; This is the current version with V2 sticker on the backside. (2020-07-29);

Codes description

Bottom hardware interface

We package the bottom for different hardware platforms.

You can check the DEV_Config.c(.h) file which is located in \STM32\STM32-F103ZET6\User\Config

- Data type:

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

- Module initialized and exit:

```
void DEV_Module_Init(void);
void DEV_Module_Exit(void);
```

Note: 1.The functions are used to set GPIIP before and after driving e-Paper.

2..If the board you have is printed with Rev2.1, the module enters low-ultra mode after DEV_Module_Exit(). (as we test, the current is about 0 in this mode);

- GPIO Read/Write:

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

- SPI Write data

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

Middle EPD driver

The epd driver are saved in: STM32\STM32-F103ZET6\User\e-Paper

Open .h file, functions are declarated here

- Initialization: It should be used to initialize e-Paper or wakeup e-Paper from sleep mode.

```
//1.54inch e-Paper, 1.54inch e-Paper V2, 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); //ode = 0 Initialize full refresh; Mode = 1 Initilize partial refresh
//Other type
void EPD_xxx_Init(void);
```

xxx is the type of e-paper

- Clear display: This function is used to clear the e-paper to white

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

xxx is the type of e-Paper.

- Transmit a frame of image and display

```
//Black/White e-Paper
void EPD_xxx_Display(UBYTE *Image);
```

```
Because controllers of 1.54inch e-paper V2 and 2.13inch e-paper V2 were updated, you need to use EPD_xxx_DisplayPartBaseImage to display static image and
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 hardware reset or use initialize function to wake up e-Paper from sleep mode

xxx is the type of e-Paper

Application function

Basic drawing functions are provided here. You can find they in:\STM32\STM32-F103ZET6\User\GUI\GUI_Paint.c(.h)

The fonts are saved in the directory:\STM32\STM32-F103ZET6\User\Fonts

- Create a new image buffer: This function is used to create a new image with width, height, Rotate degree and its color.

```
void Paint_NewImage(UBYTE *image, UWORD Width, UWORD Height, UWORD Rotate, UWORD Color)
Parameter:
    image: The buffer of the image, this is a pointer of buffer address;
    Width: width of the image;
    Height: Height of the image;
    Rotate: Rotate degree;
    Color: Initial color of the image;
```

- Select image buffer: this function is used to select the image buffer. You can create multiple image buffers with the last function, then select the buffer for every image.

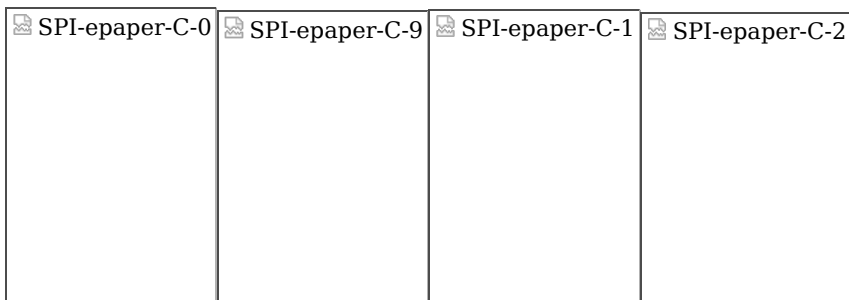
```
void Paint_SelectImage(UBYTE *image)
Parameter:
    image: The name of image buffer, it is a pointer of buffer address;
```

- Set display orientation: This function is used to set the rotate degree, it is generally be used after Paint_SelectImage(). You can set the rotate degree to 0、90、180、270 degree.


```
void Paint_SetRotate(WORD Rotate)
```

Parameter:

Rotate: Rotate degree, you can choose ROTATE_0, ROTATE_90, ROTATE_180, ROTATE_270 which stands for 0, 90, 180, 270 degree repetitively.



- Image mirroring: This function is used to mirror image.

```
void Paint_SetMirroring(UBYTE mirror)
```

Parameter:

mirror: You can set it to MIRROR_NONE, MIRROR_HORIZONTAL, MIRROR_VERTICAL, MIRROR_ORIGIN

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

```
void Paint_SetPixel(WORD Xpoint, WORD Ypoint, WORD Color)
```

Parameter:

Xpoint: X-axes in the buffer;

Ypoint: Y-axes in buffer;

Color: color

- Clear: This function is used to clear the screen to certain color.

```
void Paint_Clear(WORD Color)
```

Parameter:

Color:

- Clear windows: this function is used to clear a window. It is generally used for time display.

```
void Paint_ClearWindows(WORD Xstart, WORD Ystart, WORD Xend, WORD Yend, WORD Color)
```

Parameter:

Xstart: Start coordinate of X-axes of the window;

Ystart: Start coordinate of Y-axes of the window;

Xend: End coordinate of X-axes of the window;

Yend: End coordinate of Y-axes of the window;

Color:

- Draw point: Draw a point on the position (Xpoint, Ypoint) in buffer

```
void Paint_DrawPoint(WORD Xpoint, WORD Ypoint, WORD Color, DOT_PIXEL Dot_Pixel, DOT_STYLE Dot_Style)
```

Parameter:

Xpoint: X coordinate of point;

Ypoint: Y coordinate of point;

Color: color of point;

Dot_Pixel: the size of point, there are 8 sizes 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_Style: style of point.

```
typedef enum {
    DOT_FILL_AROUND = 1,
    DOT_FILL_RIGHTUP,
}
```

Dot_Style;

- Draw line: draw a line for (Xstart, Ystart) to (Xend, Yend)

```
void Paint_DrawLine(WORD Xstart, WORD Ystart, WORD Xend, WORD Yend, WORD Color, LINE_STYLE Line_Style , LINE_STYLE Line_Style)
```

Parameter:

Xstart: Start coordinate of X-axes of line;

Ystart: Start coordinate of Y-axes of line;

Xend: End coordinate of X-axes of line;

Yend: End coordinate of Y-axes of line

Color:

Line_width: the width of 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;
    typedef enum {
        LINE_STYLE_SOLID = 0,
        LINE_STYLE_DOTTED,
    } LINE_STYLE;

```

- Draw rectangle: Draw a rectangle from (Xstart, Ystart) to (Xend, Yend).

```

void Paint_DrawRectangle(UWORD Xstart, UWORD Ystart, UWORD Xend, UWORD Yend, UWORD Color, DOT_PIXEL Line_width, DRAW_FILL Draw_Fill)
Parameter:
    Xstart: Start coordinate of X-axes of rectangle
    Ystart: Start coordinate of Y-axes of rectangle
    Xend: End coordinate of X-end of rectangle
    Yend: End coordinate of Y-end of rectangle
    Color: color of rectangle
    Line_width: The width of edges, 8 sides 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: set the rectangle full or empty.
    typedef enum {
        DRAW_FILL_EMPTY = 0,
        DRAW_FILL_FULL,
    } DRAW_FILL;

```

- Draw circle: Draw a circle, use (X_Center Y_Center) as center;

```

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 coordinate of center
    Y_Center: Y coordinate of center
    Radius: Radius of circle
    Color: color of circle
    Line_width: width of circle, 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 circle
    typedef enum {
        DRAW_FILL_EMPTY = 0,
        DRAW_FILL_FULL,
    } DRAW_FILL;

```

- Draw character (ASCII): Set(Xstart Ystart) as left-top point, draw a ASCII character.

```

void Paint_DrawChar(UWORD Xstart, UWORD Ystart, const char Ascii_Char, sFONT* Font, UWORD Color_Foreground, UWORD Color_Background)
Parameter:
    Xstart: X coordinate of left-top pixel of character;
    Ystart: Y coordinate of left-top pixel of character;
    Ascii_Char: Ascii character;
    Font: 5 fonts are available;
        font8: 5*8
        font12: 7*12
        font16: 11*16
        font20: 14*20
        font24: 17*24
    Color_Foreground: color of character;
    Color_Background: color of background;

```

- Draw String: Set point (Xstart Ystart) as the left-top pixel, draw a string.

```

void Paint_DrawString_EN(UWORD Xstart, UWORD Ystart, const char * pString, sFONT* Font, UWORD Color_Foreground, UWORD Color_Background)
Parameter:
    Xstart: X coordinate of left-top pixel of characters;
    Ystart: Y coordinate of left-top pixel of characters;
    pString: Pointer of string
    Font: 5 fonts are available;
        font8: 5*8
        font12: 7*12
        font16: 11*16
        font20: 14*20
        font24: 17*24
    Color_Foreground: color of string
    Color_Background: color of background

```

- Draw Chinese characters: this function is used to draw Chinese fonts based ON GB2312 fonts.

```

void Paint_DrawString_CN(UWORD Xstart, UWORD Ystart, const char * pString, cFONT* font, UWORD Color_Foreground, UWORD Color_Background)
Parameter:
    Xstart: Coordinate of left-top pixel of characters;
    Ystart: Coordinate of left-top pixel of characters;

```

```
pString: Pointer of string
Font: GB2312 fonts
      font12CN: 11*21(ascii), 16*21 (Chinese)
      font24CN: 24*41(ascii), 32*41 (Chinese)
Color_Foreground: color of string
Color_Background: color of background
```

- Draw number: Draw a string of numbers, (Xstart, Ystart) is the left-top pixel.

```
void Paint_DrawNum(UWORD Xpoint, UWORD Ypoint, int32_t Nummer, sFONT* Font, UWORD Color_Foreground, UWORD Color_Background)
Parameter:
  Xstart: X coordinate of left-top pixel;
  Ystart: Y coordicate of left-to pixel;
  Nummer: the numbers displayed. the numbers are saved in int format, the maximum is 2147483647;
  Font: 5 fonts are available:
        font8: 5*8
        font12: 7*12
        font16: 11*16
        font20: 14*20
        font24: 17*24
  Color_Foreground: color of font;
  Color_Background: color of background;
```

- Display time: Display time, (Xstart, Ystart) is the left-top pixel. This function is used for e-Paper which supports partial refresh

```
void Paint_DrawTime(UWORD Xstart, UWORD Ystart, PAINT_TIME *pTime, sFONT* Font, UWORD Color_Background, UWORD Color_Foreground)
Parameter:
  Xstart: X coordinate of left-top pixel of character;
  Ystart: Y coordinate of left-top pixel of character;
  pTime: Pointer of time displayed;
  Font: 5 fonts are available;
        font8: 5*8
        font12: 7*12
        font16: 11*16
        font20: 14*20
        font24: 17*24
  Color_Foreground: color of fonts
  Color_Background: color of background
```

- Draw image: send image data of bmp file to buffer

```
void Paint_DrawBitMap(const unsigned char* image_buffer)
Parameter:
  image_buffer: address of image data in buffer
```

Hardware connection

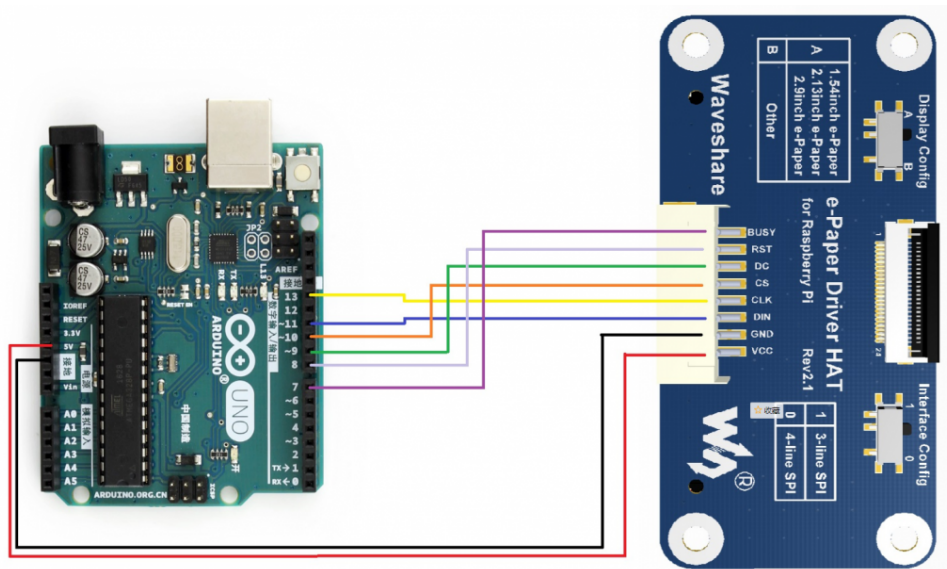
The demo code provide are based on Arduino UNO. If you want to use another Arduino board, you may need to change the connection even porting the codes yourself.

The RAM of Arduino UNO is too small to realize drawing function of e-Paper. In this case, for most of the e-Paper, we only make the image display function and save the image to flash.

We recommend you to use E-Paper Shield if you use Arduino UNO. The seven-color e-Paper is not compatible with the e-Paper shield, please note it.

Connect to Arduino UNO

e-Paper	Arduino
Vcc	5V
GND	GND
DIN	D11
CLK	D13
CS	D10
DC	D9
RST	D8
BUSY	D7



Install Arduino IDE

- Go to Arduino IDE (<https://www.arduino.cc/en/Main/Software>), and download the IDE
- Install it according to the guide of Arduino website.
- Open the IDE software after installing.

Run examples

- Download the demo codes and unzip it. Open the Arduino examples (~/.Arduino/...)
- Open a project, for example, open the opd1in54.ino project
- Open the project and set the board to Arduino UNO
- Select the correct COM port
- Build and upload the codes to board

【Note】 Because the flash of Arduino UNO is small, we recommend you use MEGA2560 if you use the big size e-Paper

Supports type

1.02inch (128×80) :

epd1in02d: Example for 1.02inch e-Paper/1.02inch e-Paper Module

1.54inch (1.54inch e-paper c: 152×152, others: 200×200) :

epd_1in54: Example for 1.54inch e-paper V1 (Black/White) : This version is stopped production which can be bought before 2019-11-22;

epd_1in54_V2: Example for 1.54inch e-paper V2 (Black/White): This is the current version which can buy now (2020-07-29). The e-Paper has a V2 sticker on the backside.

epd_1in54b: Example for 1.54inch e-paper B (Black/White/Red) ;

epd_1in54c: Example for 1.54inch e-paper C (Black/White/Red) ;

2.7inch (264×176) :

epd_2in7: Example for 2.7inch e-paper (Black/White) ;

epd_2in7b: Example for 2.7inch e-paper B (Black/White/Red) ;

2.9inch (296×128) :

epd_2in9: Example for 2.9inch e-paper (Black/White) ;

epd_2in9: Example for 2.9inch e-paper B (Black/White/Red) and 2.9inch e-paper C (Black/White/Yellow) ;

epd_2in9: Example for 2.9inch e-paper D (Black/White) ;

2.13inch (2.13inch e-Paper: 250×122, others: 212×104) :

epd_2in13: Example for 2.13inch e-paper V1 (Black/White) , this version is stopped production and it can be bought before 2019-05-15;

epd_2in13_V2: Example for 2.13inch e-paper V2 (Black/White) This is the current version with sticker V2 on the backside (2020-07-29);

epd_2in13bc: Example for 2.13inch e-paper B (Black/White/Red) and 2.13inch e-paper C (Black/White/Yellow) ;

epd_2in13d: Example for 2.13inch e-paper D (Black/White) ;

4.01inch (640×400)

epd_4in01f: Example for 4.01inch e-paper (Seven-color) ;

4.2inch (400×300)

epd_4in2: Example for 4.2inch e-paper (Black/White) ;

epd_4in2bc: Example for 4.2inch e-paper B (Black/White/Red) ;

5.65inch (600×448)

epd_5in65f: Example for for 5.65inch e-Paper F (Seven-color);

5.83inch (600×448) :

epd_5in83: Example for 5.83inch e-paper (Black/White) ;

epd_5in83bc: Example for 5.83inch e-paper B (Black/White/Red) and 5.83inch e-paper C (Black/White/Yellow) ;

7.5inch (V1: 640×384, V2: 800×480) :

epd_7in5: Example for 7.5inch e-paper (Black/White) , this version is stopped production and it can be bought before 2019-12-07;

epd_7in5bc: Example for 7.5inch e-paper B (Black/White/Red) and 7.5inch e-paper C (Black/White/Yellow) , 7.5inch e-paper B V1 version is stopped production and it can be bought before 2019-12-07;
















epd_7in5_V2: Example for 7.5inch e-paper V2 (Black/White) , This is the current version with V2 sticker on the backside (2020-07-29)

epd_7in5bc: Example for 7.5inch e-paper B V2 (Black/White/Red) ; This is the current version with a V2 sticker on the backside. (2020-07-29);

Code Description

Files

Use 2.13inch e-Paper as example, open the folder of epd2in13 v2

Name	Date modified	Type	Size
 epd2in13_V2	12/27/2019 5:21 PM	C++ Source File	10 KB
 epd2in13_V2.h	12/27/2019 5:22 PM	C/C++ Header	3 KB
 epd2in13_V2	12/27/2019 5:23 PM	Arduino file	1 KB
 epdif	3/27/2019 4:40 PM	C++ Source File	2 KB
 epdif.h	8/11/2017 9:23 AM	C/C++ Header	2 KB
 epdpaint	9/12/2017 9:02 AM	C++ Source File	9 KB
 epdpaint.h	6/24/2019 3:18 PM	C/C++ Header	3 KB
 font8	7/17/2017 10:20 AM	C Source File	18 KB
 font12	7/17/2017 10:27 AM	C Source File	26 KB
 font16	7/17/2017 10:27 AM	C Source File	47 KB
 font20	7/17/2017 10:27 AM	C Source File	63 KB
 font24	7/17/2017 10:21 AM	C Source File	95 KB
 fonts.h	7/13/2017 10:25 AM	C/C++ Header	3 KB
 imagedata	12/27/2019 5:10 PM	C++ Source File	22 KB
 imagedata.h	12/27/2019 5:11 PM	C/C++ Header	2 KB

epd2in13.ino: The project file

epd2in13.cpp(.h):EPD driver of e-Paper

epdif.cpp(.h): Hardware interface, functions for GPIO and SPI ;

font8.cpp、font12.cpp、font16.cpp、font20.cpp、font24.cpp、fonts.h: fonts which has

imagedata.cpp(.h): Image data, which is pre-converted data for display directly.

The codes are include three parts: Bottom hardware interface, middle EPD driver and application functions.程;

Bottom hardware interface

hardware interfaces are defined in epdif.cpp(.h) file

Write GPIO

```
void DigitalWrite(int pin, int value)
```

The first parameter is GPIO, and second parameter is level

Read GPIO

```
int DigitalRead(int pin)
```

The parameter is GPIO, and return value is level

Delay

```
DelayMs(unsigned int delaytime)
```

Delay time, unit is ms

SPI transmit data

```
SpiTransfer(unsigned char data)
```

Type of parameter is char

Hardware initailze

```
int IfInit(void)
```

The initialize function of SPI, input/ouptu are packaged here.

Middle EPD driver

Instantiate e-Paper class

The Arduino codes are based on C++, should instantiate e-Paper class is necessary.

```
Epd epd;
```

Initialize e-Paper, it should be used to initialize e-Paper or wakeup e-Paper from sleep mode.

- 2.13inch e-Paper, 2.9inch e-Paper

```
epd.Init(lut_full_update); //Fully update  
epd.Init(lut_partial_update); //Partial update
```

- 4.2inch e-Paper

```
epd.Init();
```

Clear, clear the e-Paper to white

```
epd.clear();
```

In some of project, the operation is divided to two part, they work in the same way

```
epd.ClearFrameMemory(0xFF);  
epd.DisplayFrame();//Display it
```

Transmit one frame of image and display

```
void Display(const unsigned char* frame_buffer);  
void DisplayFrame(const unsigned char* frame_buffer_black, const unsigned char* frame_buffer_red); //Three color e-Paper
```


Sleep

```
epd.Sleep();
```

Set the e-Paper enter sleep mode. The consumption of the e-Paper will be reduced. However, you still need to update the display periodically to avoid a ghost problem.

Application functions

The drawing functions are defined in this part.
The coordination of the image buffer:

 E-paper arduino pic1.png

The functions are defined in epdpaint.h file

```
class Paint {
public:
    Paint(unsigned char* image, int width, int height);
    ~Paint();
    void Clear(int colored);
    int GetWidth(void);
    void SetWidth(int width);
    int GetHeight(void);
    void SetHeight(int height);
    int GetRotate(void);
    void SetRotate(int rotate);
    unsigned char* GetImage(void);
    void DrawAbsolutePixel(int x, int y, int colored);
    void DrawPixel(int x, int y, int colored);
    void DrawCharAt(int x, int y, char ascii_char, sFONT* font, int colored);
    void DrawStringAt(int x, int y, const char* text, sFONT* font, int colored);
    void DrawLine(int x0, int y0, int x1, int y1, int colored);
    void DrawHorizontalLine(int x, int y, int width, int colored);
    void DrawVerticalLine(int x, int y, int height, int colored);
    void DrawRectangle(int x0, int y0, int x1, int y1, int colored);
    void DrawFilledRectangle(int x0, int y0, int x1, int y1, int colored);
    void DrawCircle(int x, int y, int radius, int colored);
    void DrawFilledCircle(int x, int y, int radius, int colored);

private:
    unsigned char* image;
    int width;
    int height;
    int rotate;
};
```

Initailze image buffer

```
Paint(unsigned char* image, int width, int height);
```

The first parameter is image buffer, the second one is the width of the picture, and the third one is the height.

```
Paint paint(image, 0, 0); // width should be the multiple of 8
```

The second and third parameters are set to 0, you can re-configure them with functions below:

Set the width, height, rotate degree.

```
int GetWidth(void); //Get the width
void SetWidth(int width); //Set the width
int GetHeight(void); //Get the height
void SetHeight(int height); //Set the height
int GetRotate(void); //Get the degree
void SetRotate(int rotate); //Set the rotate degree
```

Get the image data

```
unsigned char* GetImage(void);
```

Draw circle

```
void DrawPixel(int x, int y, int colored);
```

Coordination (x,y)

Draw characater

```
void DrawCharAt(int x, int y, char ascii_char, sFONT* font, int colored);
```

Set (x,y) as the start point, draw characters ascii_char, set the fonts as font, color is colored.

Draw string

```
void DrawStringAt(int x, int y, const char* text, sFONT* font, int colored);
```

Set (x,y) as the start point, draw the string text, font is font, color is colored

Draw line

```
void DrawLine(int x0, int y0, int x1, int y1, int colored);
```

Use (x0,y0) as start point, (x1,y1) as end point;

Draw cross line

```
void DrawHorizontalLine(int x, int y, int width, int colored);
```

Set (x0,y0) as start points, draw a line, the width is width, and color is colored

Draw a vertical line

```
void DrawVerticalLine(int x, int y, int height, int colored);
```

Use (x0,y0) as start point, draw a vertical line, width is height and color is colored.

Draw a empty rectangle

```
void DrawRectangle(int x0, int y0, int x1, int y1, int colored);
```

User (x0,y0) as start point, (x1,y1) is end point, draw a rectangle, color of edges are colored.

Draw a full rectangle

```
void DrawFilledRectangle(int x0, int y0, int x1, int y1, int colored);
```

Use (x0,y0) as start point, (x1,y1) is end point, draw a rectangle, filled it with color: colored

Draw an empty circle

```
void DrawCircle(int x, int y, int radius, int colored);
```

Use (x,y) as center, draw a empty circle with radius, color is colored

Draw a full circle

```
void DrawFilledCircle(int x, int y, int radius, int colored);
```

Use (x,y) as center, draw a circle, radius is radius, filled with color: colored

Setup Arduino IDE (Windows)

To use the ESP32 board with Arduino IDE, you need to first add the support of the board package to the IDE. We use Windows PC as an example.

The ESP32 used here is the Waveshare ESP32 e-Paper Driver Board (<https://www.waveshare.com/e-paper-esp32-driver-board.htm>)

- Download the Arduino IDE from Arduino website (<https://www.arduino.cc/en/software/>)
- Install the Arduino-ESP32 Support Page: Click to download (<https://code.load.github.com/espressif/arduino-esp32/zip/master>)
- Unzip the support package to the hardware/espressif/esp32 folder, which is under the installation directory of Arduino IDE. You may need to create them if you didn't find the directory on your PC.

Program Files > arduino-1.8.5 > hardware > espressif > esp32

名称	修改日期	类型	大小
cores	2018/8/2 17:38	文件夹	
docs	2018/8/2 17:38	文件夹	
libraries	2018/8/2 17:56	文件夹	
package	2018/8/2 17:38	文件夹	
tools	2018/8/2 17:39	文件夹	
variants	2018/8/2 17:39	文件夹	
.gitignore	2018/7/30 20:42	GITIGNORE 文件	1 KB
.gitmodules	2018/7/30 20:42	GITMODULES 文	1 KB
.travis.yml	2018/7/30 20:42	YML 文件	2 KB
appveyor.yml	2018/7/30 20:42	YML 文件	1 KB
boards.txt	2018/7/30 20:42	文本文档	82 KB
CMakeLists.txt	2018/7/30 20:42	文本文档	8 KB
component.mk	2018/7/30 20:42	MK 文件	1 KB
Kconfig.projbuild	2018/7/30 20:42	PROJBUILD 文件	7 KB
Makefile.projbuild	2018/7/30 20:42	PROJBUILD 文件	1 KB
package.json	2018/7/30 20:42	JSON 文件	1 KB
platform.txt	2018/7/30 20:42	文本文档	10 KB
programmers.txt	2018/7/30 20:42	文本文档	0 KB
README.md	2018/7/30 20:42	MD 文件	4 KB

- Enter the tools folder, run the get.exe file as administrator.
- After installing, restart the Arduino IDE, and you can find the ESP DEV Module options in the Tools-Boards menu of IDE

Download Demo Codes

- Download the ESP32 Demo codes e-Paper ESP32 Driver Codes (https://www.waveshare.com/w/upload/5/50/E-Paper_ESP32_Driver_Board_Code.7z) to your PC.
- Unzip the demo codes

The demo codes are included in the examples folder.

E-Paper_ESP32_Driver_Board_Code >

名称	修改日期	类型	大小
ePape_Esp32_Loader_APP	2020/8/12 18:42	文件夹	
examples	2020/7/22 17:33	文件夹	
Loader_esp32bt	2020/8/10 18:55	文件夹	
Loader_esp32wrf	2020/8/29 14:01	文件夹	
app-release.apk	2020/8/12 18:42	APK 文件	1,433 KB

- You should copy the esp32-waveshare-epd folder to the \hardware\espressif\esp32\libraries folder which is under the installation directory of IDE. The esp32-waveshare-epd is saved under the examples directory.


Arduino > hardware > espressif > esp32 > libraries >


名称	修改日期	类型	大小
ArduinoOTA	2020/7/22 17:32	文件夹	
AsyncUDP	2020/7/22 17:32	文件夹	
AzureIoT	2020/7/19 7:21	文件夹	
BLE	2020/7/22 17:32	文件夹	
BluetoothSerial	2020/7/22 17:32	文件夹	
DNSServer	2020/7/22 17:32	文件夹	
EEPROM	2020/7/22 17:32	文件夹	
ESP32	2020/7/22 17:32	文件夹	
esp32-waveshare-epd	2020/9/1 15:14	文件夹	
ESPmDNS	2020/7/22 17:32	文件夹	
FFat	2020/7/22 17:32	文件夹	
FS	2020/7/22 17:32	文件夹	
HTTPClient	2020/7/22 17:32	文件夹	
HTTPUpdate	2020/7/22 17:32	文件夹	
NetBIOS	2020/7/22 17:32	文件夹	
Preferences	2020/7/22 17:32	文件夹	
SD	2020/7/22 17:32	文件夹	
SD_MMC	2020/7/22 17:32	文件夹	
SimpleBLE	2020/7/22 17:32	文件夹	
SPI	2020/7/22 17:32	文件夹	
SPIFFS	2020/7/22 17:32	文件夹	
Ticker	2020/7/22 17:32	文件夹	
Update	2020/7/22 17:32	文件夹	
WebServer	2020/7/22 17:32	文件夹	
WiFi	2020/7/22 17:32	文件夹	
WiFiClientSecure	2020/7/22 17:32	文件夹	
Wire	2020/7/22 17:32	文件夹	
README.md	2020/7/19 7:21	MD 文件	3 KB

Hardware connection

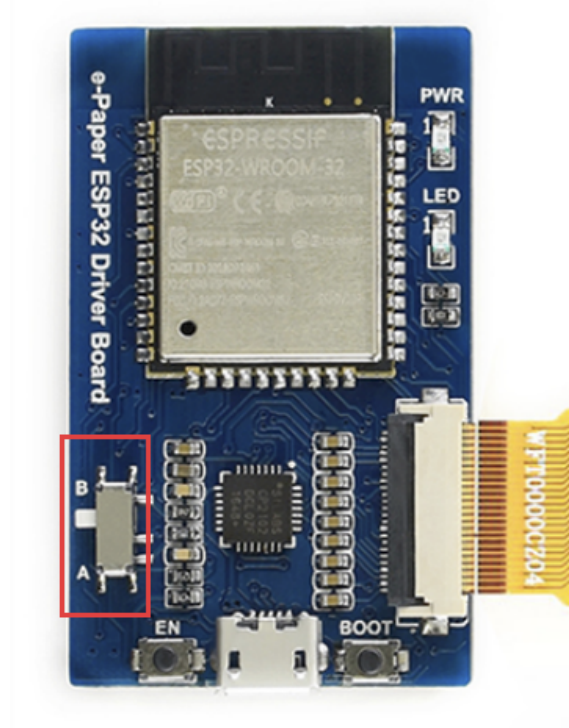
- Raw panel + ESP32 Driver Board

If you have the ESP32 e-Paper Driver board as well as an e-Paper raw panel. You can just connect the e-paper panel to the ESP32 board as below:

 Epd esp32 hard 1.png

 Epd esp32 hard 2.png

Configure the switch according to the type of display



Trigger state	e-Paper
A	1.54inch e-Paper, 2.13inch e-Paper, 2.13inch e-Paper (D), 2.9inch e-Paper
B	1.54inch e-Paper (B), 1.54inch e-Paper (C) 2.13inch e-Paper (B), 2.13inch e-Paper (C) 2.7inch e-Paper (B) 2.9inch e-Paper (C), 2.9inch e-Paper (B) 4.2inch e-Paper (B), 4.2inch e-Paper (C) 5.83inch e-Paper (B), 5.83inch e-Paper (C) 7.5inch e-Paper (B), 7.5inch e-Paper (C), 7.5inch HD e-Paper, 7.5inch HD e-Paper (B)

Note: The ACeP e-Paper doesn't support this connecting method.

- e-Paper Module/HAT + ESP32 Board

If you have the ESP32 Board as well as the e-Paper module/HAT which has PCB already, you can wire the displays by 8-pin cable pin


Connect to ESP32 Driver Board

e-Paper	ESP32
Vcc	3.3V
GND	GND
DIN	GPIO14

CLK	GPIO13
CS	GPIO15
DC	GPIO27
RST	GPIO26
BUSY	GPIO25

Run the Demo Codes

- Open the Arduino IDE
- Choose File->Examples-> waveshare-e-Paper -> And the project according to the type of display

 Epd esp32 example 2.png

- Build and program the ESP32 board
- Open the serial monitor, you can check the debug information when running the demo codes

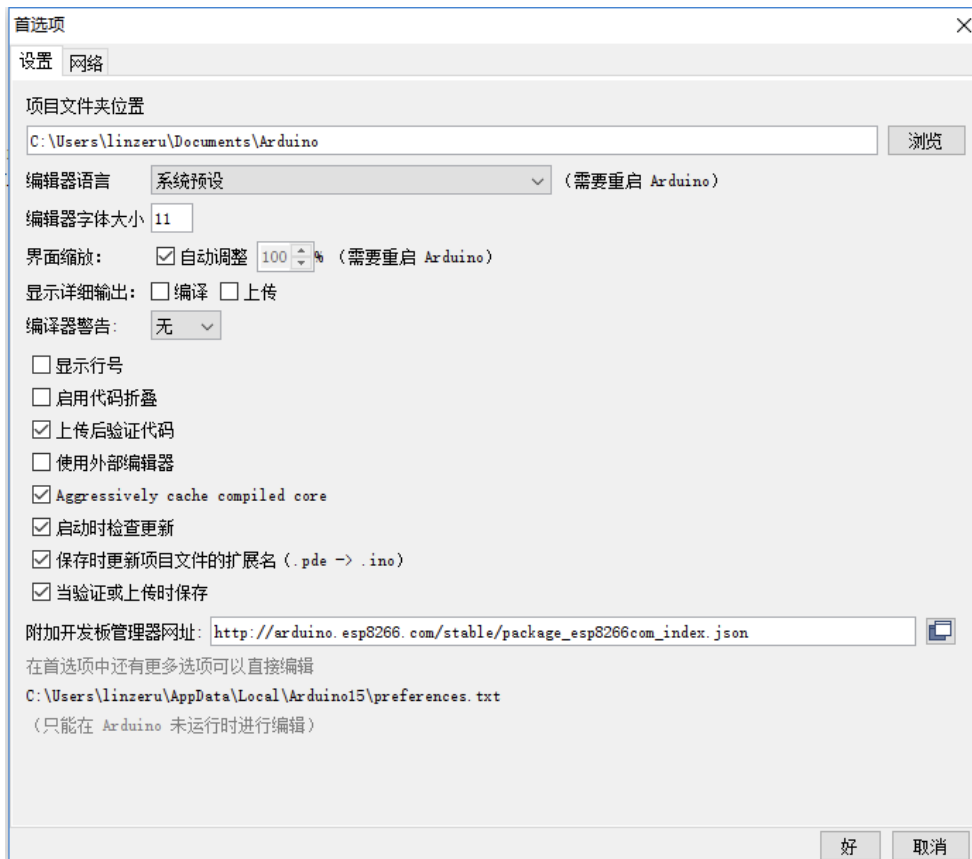
Setup Arduino IDE (Windows)

To use the ESP8266 board with Arduino IDE, you need to first add the support of the board package to the IDE. We use Windows PC as an example.

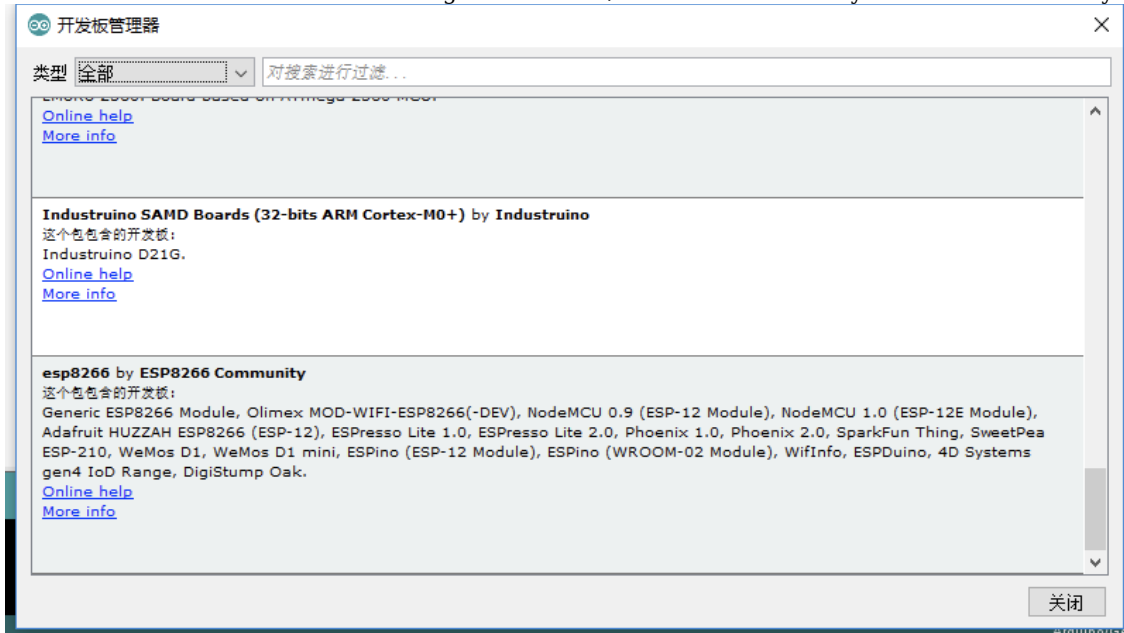
The ESP8266 used here is the Waveshare ESP8266 e-Paper Driver Board (<https://www.waveshare.com/product/displays/e-paper-esp8266-driver-board.htm>)

- Download the Arduino IDE from Arduino website (<https://www.arduino.cc/en/software/>)
- Install the Arduino-ESP8266 Support Page:

1. Open the Arduino IDE
2. Go to File->Preferences Menu, open the Settings tab
3. Text http://arduino.esp8266.com/stable/package_esp8266com_index.json into the text box **Additional Boards Manager URLs** and click OK



4. Go to Tools->Board-> Boards Manager.... find and/or install ESP8266 by ESP8266 Community.



- After installing, restart the Arduino IDE, and you can find the NodeMCU1.0(ESP-12E Module) options in the Tools->Boards menu of IDE

Download Demo Codes

- Download the ESP8266 Demo codes e-Paper ESP8266 Driver Codes (https://www.waveshare.net/w/upload/d/d5/E-Paper_ESP8266_Driver_Board_Code.7z) to your PC.
- Unzip the demo codes

The demo codes are included in the examples folder.

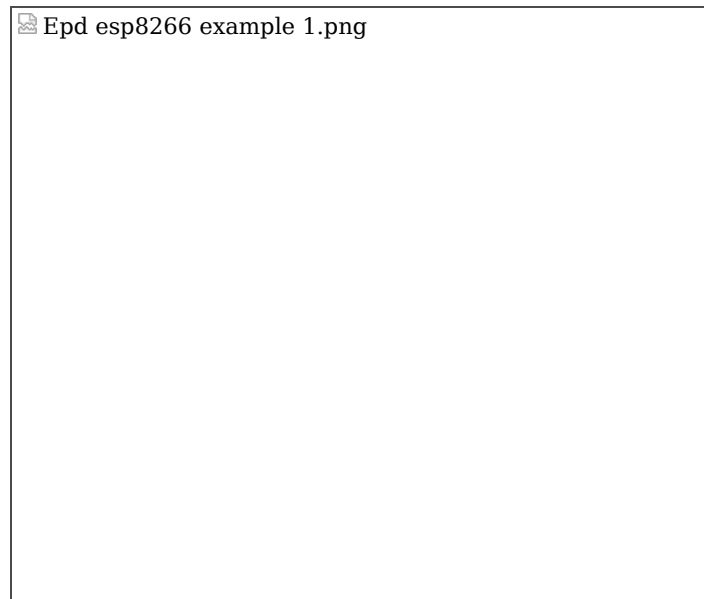
E-Paper_ESP8266_Driver_Board_Code			
名称	修改日期	类型	大小
examples	2020/2/27 15:27	文件夹	
Loader	2020/8/12 12:02	文件夹	

- You should copy the esp8266-waveshare-epd folder to the %LOCALAPPDATA%\Arduino15\packages\esp8266

\\hardware\\esp8266\\2.7.1\\libraries folder which is under the installation directory of IDE.

The path may be different according to the version of Arduino IDE and the ESP8266 libraries, you need to modify it if it is different.

The esp8266-waveshare-epd is saved under the examples directory.

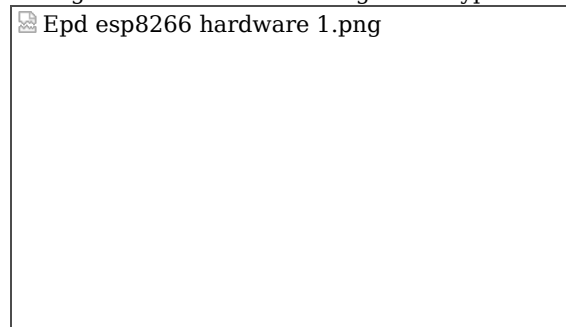


Hardware connection

- Raw panel + ESP8266 Driver Board

If you have the ESP8266 e-Paper Driver board as well as an e-Paper raw panel. You can just connect the e-paper panel to the ESP8266 board

Configure the switch according to the type of display



Trigger state	e-Paper
A	1.54inch e-Paper, 2.13inch e-Paper, 2.13inch e-Paper (D), 2.9inch e-Paper
B	1.54inch e-Paper (B), 1.54inch e-Paper (C) 2.13inch e-Paper (B), 2.13inch e-Paper (C) 2.7inch e-Paper (B) 2.9inch e-Paper (C), 2.9inch e-Paper (B) 4.2inch e-Paper (B), 4.2inch e-Paper (C) 5.83inch e-Paper (B), 5.83inch e-Paper (C) 7.5inch e-Paper (B), 7.5inch e-Paper (C), 7.5inch HD e-Paper, 7.5inch HD e-Paper (B)

Note: The ACeP e-Paper doesn't support this connecting method.

- e-Paper Module/HAT + ESP8266 Board

If you have the ESP8266 Board as well as the e-Paper module/HAT which has PCB already, you can wire the displays by 8-pin cable pin

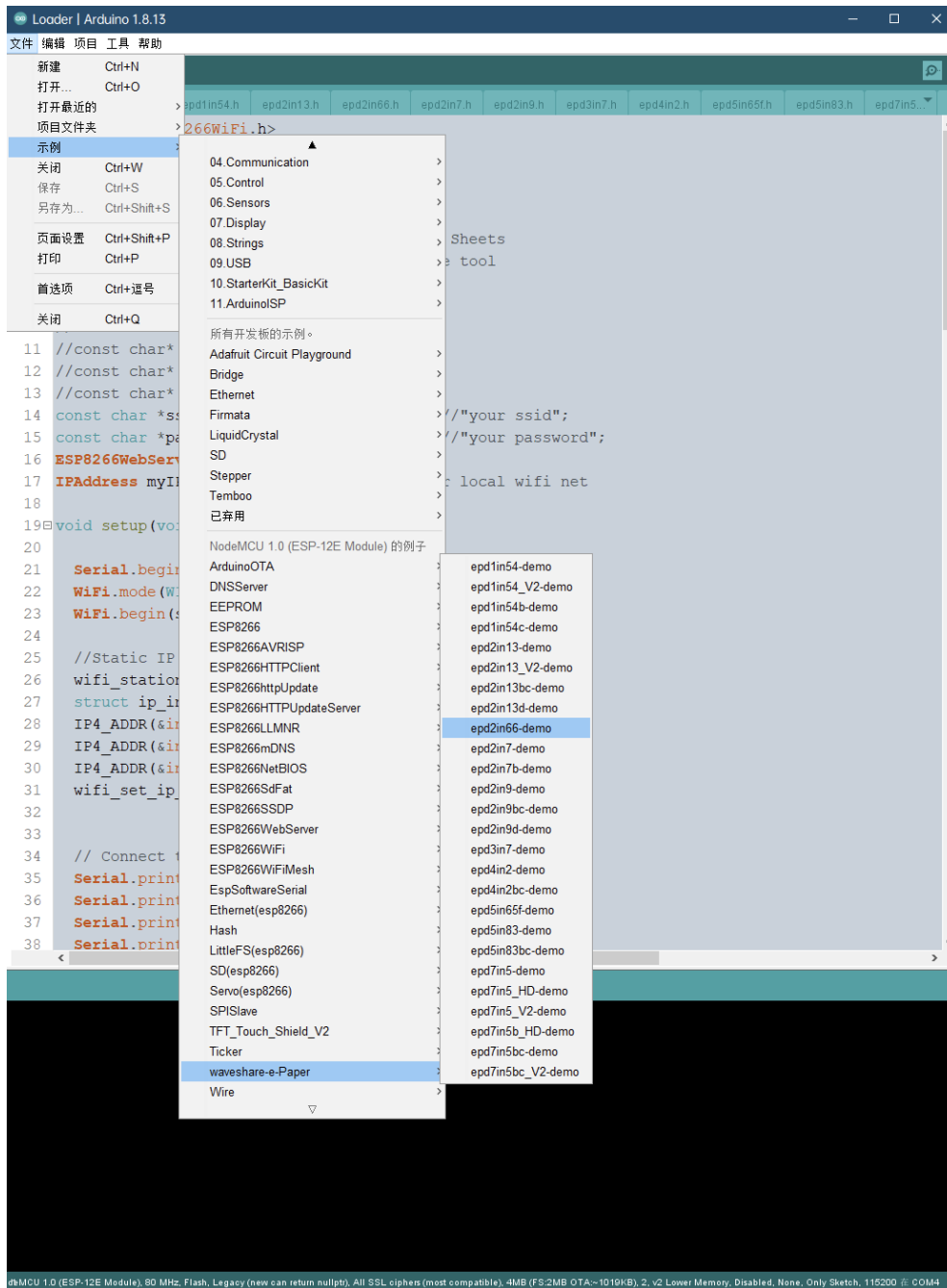
Connect to ESP8266 Driver Board

e-Paper	ESP8266
Vcc	3.3V
GND	GND
DIN	GPIO13

CLK	GPIO14
CS	GPIO15
DC	GPIO4
RST	GPIO2
BUSY	GPIO5

Run the Demo Codes

- Open the Arduino IDE
- Choose File->Examples-> waveshare-e-Paper -> And the project according to the type of display



- Build and program the ESP8266 board
- Open the serial monitor, you can check the debug information when running the demo codes

Resources

Documentation

- Instruction about make new font (<https://wavesharejfs.blogspot.com/2018/08/make-new-larger-font-for-waveshare-spi.html>)
- Make BMP file for e-Paper (https://www.waveshare.com/wiki/E-Paper_Floyd-Steinberg)
- 3.7inch e-Paper HAT Schematic (https://www.waveshare.com/w/upload/8/8e/3.7inch_e-Paper_Schematic.pdf)

- 3.7inch e-Paper Specification (https://www.waveshare.com/w/upload/7/71/3.7inch_e-Paper_Specification.pdf)
- SSD1677_1.0 Datasheet (https://www.waveshare.com/w/upload/2/2a/SSD1677_1.0.pdf)

Demo code

- Github (<https://github.com/waveshare/e-Paper>)

Related Resources

This is a post in Arduino Form about our SPI e-Paper thanks to ZinggJM, maybe you want to refer to it.

- Waveshare e-Paper display with SPI (<https://forum.arduino.cc/index.php?topic=487007.0>)

FAQ

Question:

Working requirements of e-Paper?

Answer:

- Two-color B/W e-paper
 - 【Working】 Temperature: 0~50°C; Humidity: 35%~65%RH
 - 【Storage】 Temperature: ≤30°C; Humidity: ≤55%RH; Max storage time: 6 months
 - 【Transport】 Temperature: -25~70°C; Max transport time: 10 days
 - 【Unpack】 Temperature: 20°C±5°C; Humidity: 50%RH±5%RH; Max storage time: Should be assembled in 72h
- Three-Color e-Paper
 - 【Working】 Temperature: 0~40°C; Humidity: 35%~65%RH
 - 【Storage】 Temperature: ≤30°C; Humidity: ≤55%RH; Max storage time: 3 months
 - 【Transport】 Temperature: -25~60°C; Max transport time: 10 days
 - 【Unpack】 Temperature: 20°C±5°C; Humidity: 50%RH±5%RH; Max storage time: Should be assembled in 72h

When store three-color e-Paper, please refresh it to white, and keep the screen upward. Note that you need to update it at least every three months.

Question:

What do you need to note about e-Paper refreshing?

Answer:

- Refresh mode
 - Full refresh: e-Paper flicker when full refreshing to remove ghost image for best display.
 - Partial refresh: It don't flicker if you use partial refresh (only some of the two-color e-paper support partial refresh). Note that you cannot use Partial refresh all the time, you should full refresh e-paper regularly, otherwise, ghost problem will get worse even damage.
- Refresh rate
 - When using, you should set the update interval at least 180s.(except Partial supportable types)
 - Please set the e-Paper to sleep mode or power off it directly, otherwise, the e-Paper is damaged because of working in high voltage for long time.
 - You need to update the content of three-color e-Paper at least one time every 24h to avoid from burn-in problem.
- Working place
- We recommend you to use the e-Paper indoor. If you need to set the e-paper outdoor, Please place the e-paper in shadow and protect it from UV. When you designed you e-paper product, you should take care about the working situation like temperature, humidity, etc.

Question:

How much could the flexible e-paper be bended

Answer:

- The IC part of e-Paper cannot be bended, you can bend the display area in degree larger than 60°C

Question:

Why the e-Paper cant work with Arduino?

Answer:

The I/O level of Arduino is 5V, and the e-Paper should be driven with 3V3. If your Arduino cant drive the e-Paper successfully, please try to convert the level to 3.3V

You can also try to connect the Vcc pin to the 5V of Arduino to see whether the e-Paper works, but we recommend you not to use 5V for a long time.

Question:

Why does the color of e-Paper look a little black or grey?

Answer:

You can try to change the value of Vcom on demo codes.

Question:

Three-color e-paper looks more red/yellow than the picture on website?

Answer:

Because of different batch, some of them have aberration. Store the e-Paper right side up will reduce it. And if the e-Paper didn't be refreshed for long time, it will become more and more red/yellow. Please use the demo code to refresh the e-paper for several times in this case.

Question:

Why my e-paper has ghosting problem after working for some days

Answer:

Please set the e-paper to sleep mode or disconnect it if you needn't refresh the e-paper but need to power on your development board or Raspberry Pi for long time. Otherwise, the voltage of panel keeps high and it will damage the panel

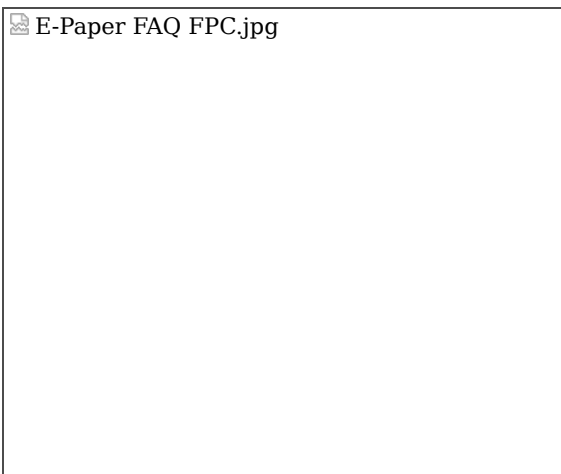
Question:

Why the FPC of the e-Paper is broken after using for some times?

Answer:

Please make sure you have used it in correct way.

 E-Paper FAQ FPC.jpg



Support

If you require technical support, please go to the Support (<https://support.waveshare.com>)

[/hc/en-us/requests/new](#)) page and open a tickets.

Retrieved from "https://www.waveshare.com/w/index.php?title=3.7inch_e-Paper_HAT&oldid=21451"

- This page was last modified on 23 February 2021, at 09:24.
- This page has been accessed 6,765 times.

If you require technical support, please go to the Support (<https://support.waveshare.com>