e-Paper Shield User Manual

OVERVIEW

This is a universal driver board for connecting Arduino/NUCLEO and e-Paper raw panels, can be used to drive various SPI interface e-Paper from Waveshare. This module has external SPI RAM chip onboard, allows for SD card reading/writing.

FEATURES

- Standard Arduino connectivity, compatible with host boards like UNO, Leonardo, NUCLEO, XNUCLEO. Universal driver board for all types of Waveshare SPI raw panel e-Paper.
- Onboard voltage translator, compatible with 3.3V/5V MCUs
- Onboard external 1M RAM 24LC024, more available space
- Micro SD slot, images can be stored on TF card for displaying
- Reserved control interface, for connecting with other control boards
- Comes with development resources and manual (examples for Arduino/NUCLEO)

SPECIFICATION

Operating voltage: 5V/3.3V

Interface: SPI

Regulator: RT9193-33

Level convertor: TXS0108

External RAM: 23LC1024

SD card: support

Dimension: 53.34mm*53.34mm

PINOUTS

PIN	Arduino / NUCLEO	Description
VCC	5V	Power
GND	GND	Ground
SCK	SCK	SPI clock input
MOSI	MOSI	SPI data input
MISO	MISO	SPI data output
EPD_CS	D10	e-Paper chip selection
EPD_DC	D9	e-Paper data/command
EPD_RST	D8	e-Paper reset

EPD_BUSY	D7	e-Paper busy
SD_CS	D6	SD card chip selection
SPIRAM_CS	D5	External SPIRAM chip selection

SUPPORTS E-PAPER

Type	Display Color	Grey Scale	Resolution	Refresh time (s)	Partial Refresh
1.54inch e-Paper	Black, White	2	200×200	2	√
1.54inch e-Paper (B)	Red, Black, White	2	200×200	8	×
1.54inch e-Paper (C)	Yellow, Black, White	2	152×152	27	×
2.13inch e-Paper	Black, White	2	250×122	2	√
2.13inch e-Paper (B)	Red, Black, White	2	212×104	15	×
2.13inch e-Paper (C)	Yellow, Black, White	2	212×104	15	×
2.7inch e-Paper	Black, White	2	264×176	6	×
2.7inch e-Paper (B)	Red, Black, White	2	264×176	15	×

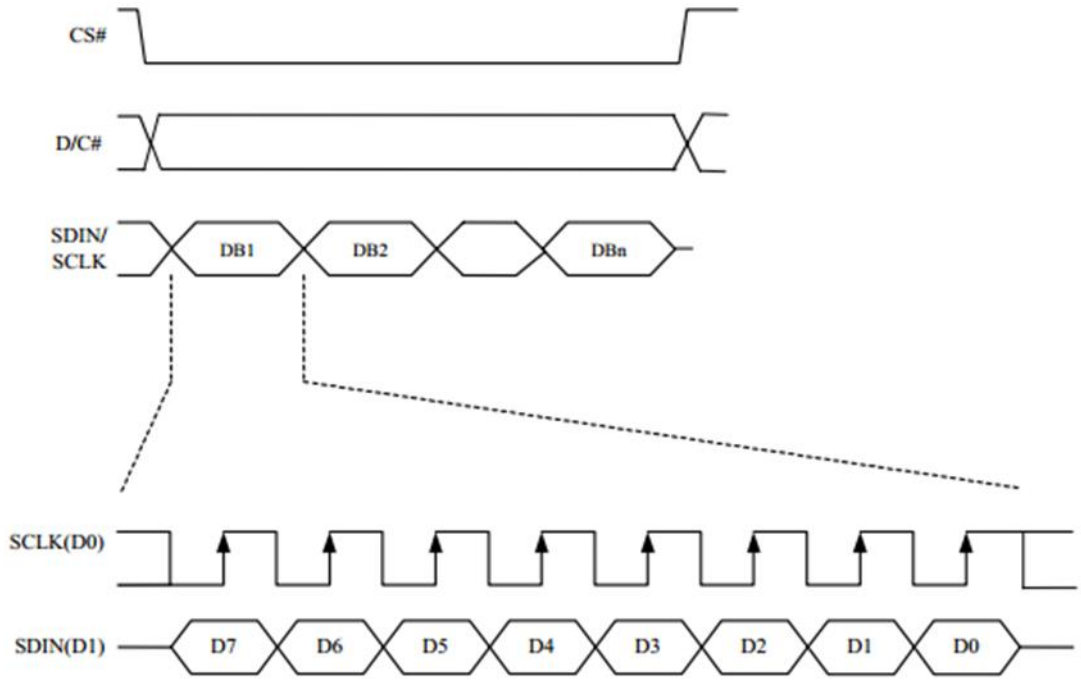
2.9inch e-Paper	Black, White	2	296×128	2	√
2.9inch e-Paper (B)	Red, Black, White	2	296×128	15	×
2.9inch e-Paper (C)	Yellow, Black, White	2	296×128	15	×
4.2inch e-Paper	Black, White	2	400×300	4	×
4.2inch e-Paper (B)	Red, Black, White	2	400×300	15	×
4.2inch e-Paper (C)	Yellow, Black, White	2	400×300	15	×
5.83inch e-Paper	Black, White	2	200×200	3.5	√
5.83inch e-Paper (B)	Red, Black, White	2	200×200	14	×
5.83inch e-Paper (C)	Yellow, Black, White	2	152×152	26	×
7.5inch e-Paper	Black, White	2	250×122	6	√
7.5inch e-Paper (B)	Red, Black, White	2	212×104	16	×
7.5inch e-Paper (C)	Yellow, Black, White	2	212×104	16	×

Refresh time: This is the time of full refresh. If the e-paper support partial refresh, partial refresh time is about 0.3s

HARDWARE DESCRIPTION

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 sunshine, the E-paper screen still has high visibility with a wide viewing angle of 180 degree. It is the ideal choice for E-reading.

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 display requirement.

- CS is 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 by CPHA and CPOL.

- 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.
- CPHA determines whether data is collected at the first clock edge or at the second clock edge of 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. In here, SPI0 is in used, and data is transferred by bits, MSB first.

HOW TO USE

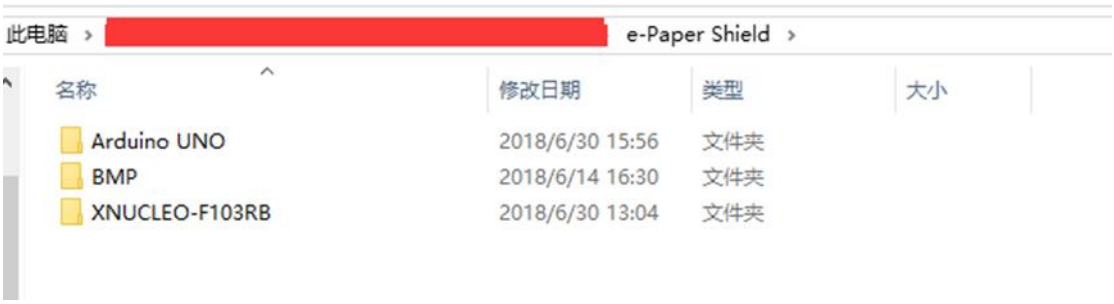
HARDWARE CONFIGURATION

- If your Arduino board has ICSP interface, set the SPI Config of e-Paper shield into ICSP (by default)
- If your Arduino board has no ICSP interface, set the SPI Config of e-Paper shield into SCLK/D13, MISO/D12, MOSI/D11 separately

FILES INCLUED IN DEMO CODE

Open the product page on our website, find the wiki link on Development Resources, click and open it. Demo code are provided on Wiki.

Download and unzip, you can get three demo codes as below:



Arduino UNO： This code is used for Arduino UNO board;

XNUCLEO-F103RB: These are used for XNUCLEO_F103RB board. Two demo code include, one use internal RAM and another use external RAM;

BMP: Pictures for display, you can copy to SD card if require.

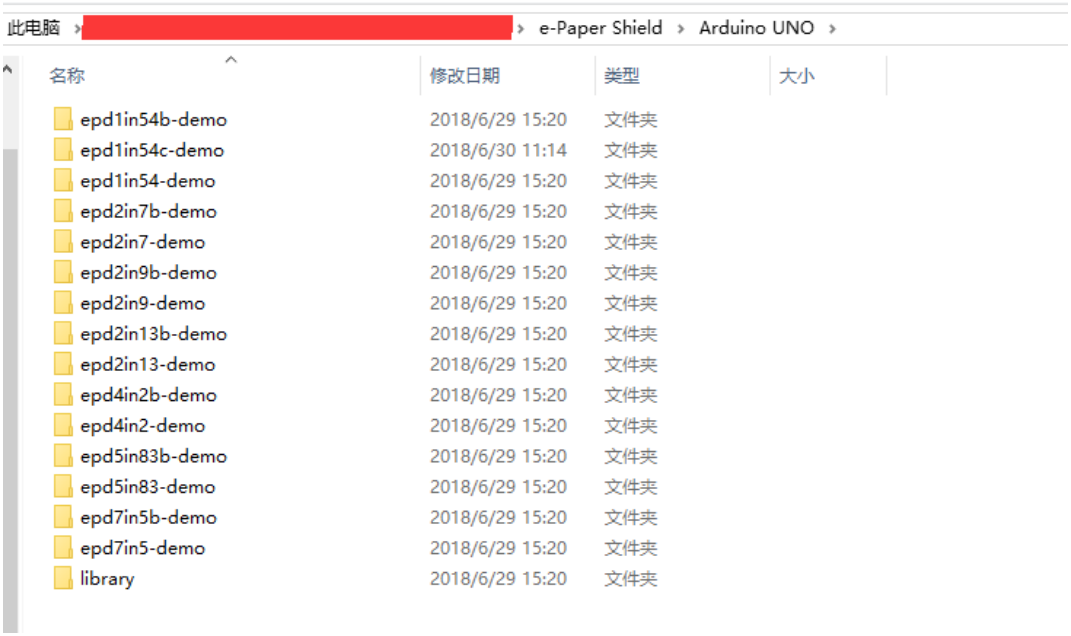
- Format of picture: monochrome BMP
- SD card format: FAT

DEMO CODES

We provide demo codes which are based on Arduino UNO and XNUCLEO-F103RB.

ARDUINO UNO

Open Arduino directory, you can get the demo codes as:



Choose one according to the type of e-Paper you have.

Before use, you need to copy the Waveshare_ePaper folder which is in library to libraries directory of IDE, the libraries directory is under the installation directory of Arduino IDE, it generally is in: C:\Program Files (x86)\Arduino\libraries

DISPLAY PICTURES FROM SD CARD

Use 1.5inch e-Paper as example:

1. Initialize e-Paper and clear its buffer

```
GUIPAINT paint;
EPD1IN54 epd;
EPD_SDCARD SD;

if(epd.EPD_Init(lut_full_update) != 0) {
    Serial.print("e-Paper init failed");
    return;
}
epd.EPD_Clear();
Serial.print("1.54inch e-Paper demo\n");
```

2. Initialize SD card, make sure you have insert the SD card, or it will cause error

```
//1.Initialize the SD card
SD.SDCard_Init();
```

3. Create a framebuffer for image, and set it to monochrome, set the width and height, rotate and color parameters. Clear buffer.

```
//2.Create a new image cache named IMAGE_BW and fill it with white
paint.Paint_NewImage(IMAGE_BW, EPD_WIDTH, EPD_HEIGHT, IMAGE_ROTATE_0, IMAGE_COLOR_POSITIVE);
paint.Paint_Clear(WHITE);
```

4. Read picture and save it to external RAM

```
//3.Read BMP pictures into RAM
SD.SDCard_ReadBMP("1in54.bmp", 0, 0);
```

5. Transmitting data from external RAM to buffer of e-Paper and refresh

```
//4.Refresh the picture in RAM to e-Paper
epd.EPD_Display();
Dev->Dev_Delay_ms(2000);
```

DISPLAY PICTURES FROM ARRAY

1. Create a buffer for image and set it to monochrome. Define height and width, rotate and invert parameters. Clear buffer

```
//2.show image for array, IMAGE_ROTATE_0 and IMAGE_COLOR_POSITIVE will not affect reading
paint.Paint_DrawBitMap(IMAGE_DATA);
```

2. Save data from array to external RAM

3. Transmitting data from external RAM to buffer of e-Paper and refresh

DRAWING

1. Initialize e-paper, and clear the buffer

2. create an image buffer, set it to monochrome. Define the width and length. Set the rotate and invert parameters. Clear buffer

3. Draw point, set its position, color, size and full/empty

```
//2.Drawing on the image
    paint.Paint_DrawPoint(5, 10, BLACK, DOT_PIXEL_1X1, DOT_STYLE_DFT);
    paint.Paint_DrawPoint(5, 25, BLACK, DOT_PIXEL_2X2, DOT_STYLE_DFT);
    paint.Paint_DrawPoint(5, 40, BLACK, DOT_PIXEL_3X3, DOT_STYLE_DFT);
    paint.Paint_DrawPoint(5, 55, BLACK, DOT_PIXEL_4X4, DOT_STYLE_DFT);
```

4. Draw line, set its begin position, color, solid/dotted and width

```
paint.Paint_DrawLine(20, 10, 70, 60, BLACK, LINE_STYLE_SOLID, DOT_PIXEL_1X1);
paint.Paint_DrawLine(70, 10, 20, 60, BLACK, LINE_STYLE_SOLID, DOT_PIXEL_1X1);
paint.Paint_DrawLine(170, 15, 170, 55, BLACK, LINE_STYLE_DOTTED, DOT_PIXEL_1X1);
paint.Paint_DrawLine(150, 35, 190, 35, BLACK, LINE_STYLE_DOTTED, DOT_PIXEL_1X1);
```

5. Draw rectangle, set the begin position of diagonal line, color, full/empty and width of line

```
paint.Paint_DrawRectangle(20, 10, 70, 60, BLACK, DRAW_FILL_EMPTY, DOT_PIXEL_1X1);
paint.Paint_DrawRectangle(85, 10, 130, 60, BLACK, DRAW_FILL_FULL, DOT_PIXEL_1X1);
```

6. Draw circle, set center position, radius, color, full/empty and width of line

```
paint.Paint_DrawCircle(170, 35, 20, BLACK, DRAW_FILL_EMPTY, DOT_PIXEL_1X1);
paint.Paint_DrawCircle(170, 85, 20, BLACK, DRAW_FILL_FULL, DOT_PIXEL_1X1);
```

7. Display string, set the begin position, contents, font size, background color and the color of font

```
paint.Paint_DrawString_EN(5, 70, "hello world", &Font16, WHITE, BLACK);
paint.Paint_DrawString_EN(5, 90, "waveshare", &Font20, BLACK, WHITE);
```

8. Display number, set the begin position, number, font size, background color and the color of font

```
paint.Paint_DrawNum(5, 120, 123456789, &Font20, BLACK, WHITE);
```

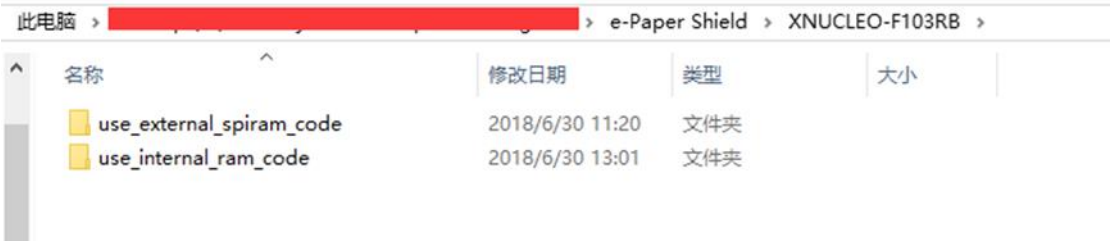
9. Display time, set the begin position, structure of time, font size, background color and the color of font (partial refresh)

```
PAINT_TIME sPaint_time;
sPaint_time.Hour = 12;
sPaint_time.Min = 34;
sPaint_time.Sec = 56;
for (;;) {
    sPaint_time.Sec = sPaint_time.Sec + 1;
    if (sPaint_time.Sec == 60) {
        sPaint_time.Min = sPaint_time.Min + 1;
        sPaint_time.Sec = 0;
        if (sPaint_time.Min == 60) {
            sPaint_time.Hour =  sPaint_time.Hour + 1;
            sPaint_time.Min = 0;
            if (sPaint_time.Hour == 24) {
                sPaint_time.Hour = 0;
                sPaint_time.Min = 0;
                sPaint_time.Sec = 0;
            }
        }
    }
    paint.Paint_ClearWindows(15, 150, 15 + Font24.Width * 7, 150 + Font24.Height, WHITE);
    paint.Paint_DrawTime(15, 150, &sPaint_time, &Font24, WHITE, BLACK);
```

NUCLEO-F103RB

- The development board used is XNUCLEO-F103RB
- The examples are based on HAL libraries. You can port it to other STM chip with STM32CubeMX
- The examples are compiled in Keil v5

For STM32, we also provide two kinds of demos, one use internal RAM and another use external RAM



DISPLAY PICTURES FROM SD CARD

Use 1.54inch e-Paper as example

1. Initialize e-Paper and clear its buffer
2. Initialize SD card, make sure you have inserted the SD card, or it will cause error
3. Create a framebuffer for image, and set it to monochrome, set the width and height, rotate and color parameters. Clear buffer.
4. Read picture and save it to external RAM
5. Transmitting data from external RAM to buffer of e-Paper and refresh

```
printf("1.54inch e-Paper demo\r\n");
System_Init();

if(EPD_Init(lut_full_update) != 0) {
    printf("e-Paper init failed\r\n");
}
EPD_Clear();
Dev_Delay_ms(500);
#if 1
    /*show sd card pic*/
  //1.Initialize the SD card
  SDCard_Init();

  //2.Create a new image cache named IMAGE_BW and fill it with white
  Paint_NewImage(IMAGE_BW, EPD_WIDTH, EPD_HEIGHT, IMAGE_ROTATE_0, IMAGE_COLOR_POSITIVE);
  Paint_Clear(WHITE);

  //3.Read BMP pictures into RAM
  SDCard_ReadBMP("lin54.bmp", 0, 0);

  //4.Refresh the picture in RAM to e-Paper
  EPD_Display();
  Dev_Delay_ms(2000);
```

DISPLAY PICTURES FROM ARRAY

1. Initialize e-Paper and clear its buffer.

2. Create a buffer for image and set it to monochrome. Define height and width, rotate and invert parameters. Clear buffer

3. Save data from array to external RAM

4. Transmitting data from external RAM to buffer of e-Paper and refresh

```
/*show image for array*/
printf("show image for array\r\n");
//1.Create a new image cache named IMAGE_BW and fill it with white
Paint_NewImage(IMAGE_BW, EPD_WIDTH, EPD_HEIGHT, IMAGE_ROTATE_0, IMAGE_COLOR_POSITIVE);
Paint_Clear(WHITE);

printf("Paint_DrawBitMap\r\n");
//2.show image for array, IMAGE_ROTATE_0 and IMAGE_COLOR_POSITIVE will not affect reading
Paint_DrawBitMap(IMAGE_DATA);

printf("EPD_Display\r\n");
//3.Refresh the picture in RAM to e-Paper
EPD_Display();
Dev_Delay_ms(2000);
```

DRAWING

1. Initialize e-paper, and clear the buffer

2. create an image buffer, set it to monochrome. Define the width and length. Set the rotate and invert parameters. Clear buffer

3. Draw point, set its position, color, size and full/empty

4. Draw line, set its begin position, color, solid/dotted and width

5. Draw rectangle, set the begin position of diagonal line, color, full/empty and width of line

6. Draw circle, set center position, radius, color, full/empty and width of line

7. Display string, set the begin position, contents, font size, background color and the color of font

8. Display number, set the begin position, number, font size, background color and the color of font

9. Transmit data from external RAM to buffer of e-Paper and refresh

```
//1.Create a new image cache named IMAGE_BW and fill it with white
Paint_NewImage(IMAGE_BW, EPD_WIDTH, EPD_HEIGHT, IMAGE_ROTATE_90, IMAGE_COLOR_POSITIVE);
Paint_Clear(WHITE);

//2.Drawing on the image
Paint_DrawPoint(5, 10, BLACK, DOT_PIXEL_1X1, DOT_STYLE_DFT);
Paint_DrawPoint(5, 25, BLACK, DOT_PIXEL_2X2, DOT_STYLE_DFT);
Paint_DrawPoint(5, 40, BLACK, DOT_PIXEL_3X3, DOT_STYLE_DFT);
Paint_DrawPoint(5, 55, BLACK, DOT_PIXEL_4X4, DOT_STYLE_DFT);

Paint_DrawLine(20, 10, 70, 60, BLACK, LINE_STYLE_SOLID, DOT_PIXEL_1X1);
Paint_DrawLine(70, 10, 20, 60, BLACK, LINE_STYLE_SOLID, DOT_PIXEL_1X1);
Paint_DrawLine(170, 15, 170, 55, BLACK, LINE_STYLE_DOTTED, DOT_PIXEL_1X1);
Paint_DrawLine(150, 35, 190, 35, BLACK, LINE_STYLE_DOTTED, DOT_PIXEL_1X1);

Paint_DrawRectangle(20, 10, 70, 60, BLACK, DRAW_FILL_EMPTY, DOT_PIXEL_1X1);
Paint_DrawRectangle(85, 10, 130, 60, BLACK, DRAW_FILL_FULL, DOT_PIXEL_1X1);

Paint_DrawCircle(170, 35, 20, BLACK, DRAW_FILL_EMPTY, DOT_PIXEL_1X1);
Paint_DrawCircle(170, 85, 20, BLACK, DRAW_FILL_FULL, DOT_PIXEL_1X1);

Paint_DrawString_EN(5, 70, "hello world", &Font16, WHITE, BLACK);
Paint_DrawString_EN(5, 90, "waveshare", &Font20, BLACK, WHITE);

Paint_DrawNum(5, 120, 123456789, &Font20, BLACK, WHITE);

//3.Refresh the picture in RAM to e-Paper
EPD_Display();
Dev_Delay_ms(2000);
```

PARTIAL REFRESH

Only the two color (black and white) version of 1.54inch, 2.13inch and 2.9inch e-Paper could support partial refresh.

1. Initial e-Paper to partial refresh mode
2. Define a time structure, and set the time (hour, minute, second)
3. Clear data of a window
4. Display the time, set its begin coordinate, font size, background color and the font color.
5. Transmit data from external RAM to buffer of e-Paper and refresh

```
//4.Partial refresh, example shows time
if(EPD_Init(lut_partial_update) != 0) {
    printf("e-Paper init failed\r\n");
}

PAINT_TIME sPaint_time;
sPaint_time.Hour = 12;
sPaint_time.Min = 34;
sPaint_time.Sec = 56;
for (;;) {
    sPaint_time.Sec = sPaint_time.Sec + 1;
    if (sPaint_time.Sec == 60) {
        sPaint_time.Min = sPaint_time.Min + 1;
        sPaint_time.Sec = 0;
        if (sPaint_time.Min == 60) {
            sPaint_time.Hour =  sPaint_time.Hour + 1;
            sPaint_time.Min = 0;
            if (sPaint_time.Hour == 24) {
                sPaint_time.Hour = 0;
                sPaint_time.Min = 0;
                sPaint_time.Sec = 0;
            }
        }
    }
    Paint_ClearWindows(15, 150, 15 + Font24.Width * 7, 150 + Font24.Height, WHITE);
    Paint_DrawTime(15, 150, &sPaint_time, &Font24, WHITE, BLACK);

    EPD_Display();
    Dev_Delay_ms(500);//Analog clock 1s
}
```

DISPLAY PICTURES

There are two ways to display pictures. One is display directly and other is indirectly.

Display directly: Read the data of pictures with library functions and decode. Then convert it to arrays and send to module. The python demo code realize this function

Display indirectly: Converting pictures to relative arrays on PC and save as c file. Then you can use the c file on your project. This chapter we will talk about how to convert a picture to array.

1. Open a picture with drawing tool comes with Windows system, create a new image, and set the pixel to 104x212 (refer to resolution of the e-Paper you use)
2. Because this module can only display two gray level (Only black and white), we need to convert picture to monochrome bitmap before converting it to array. That is, File --> BMP picture --> Monochrome Bitmap.

   There is a monochrome bitmap on examples pack for demonstration (raspberrypi/python/monocolor.bmp).
3. Use Image2Lcd.exe software to generate corresponding array for picture (.c file).

   Open picture with this software, set the parameters:

   - Output data type: C language array
   - Scanning mode: vertical scanning
   - Output gray: single color (two gray level)

- Maximum width and height: 104 and 212 (refer to e-Paper’ s resolution)
- Include the data of Image Header: uncheck
- Inverse color: Check (Check: the white on image will be converted to 1, and black is converted to 0)

4. Click Save, to generate .c file. Copy the corresponding array into your project, and you can display picture by calling this array.

【Note】Image data are analyzed in different way for different size of e-Paper. You should refer to corresponding datasheet or user manual of the e-paper you use.