waveshare A7670E Cat-1/GNSS HAT

Waveshare Cat-1/GSM/GPRS/GNSS HAT User Manual

Model: A7670E Cat-1/GNSS HAT

1. Introduction

The Waveshare Cat-1/GSM/GPRS/GNSS HAT is a versatile communication module designed for Raspberry Pi and other host devices like Arduino and STM32. Based on the A7670E module, it provides robust LTE Cat-1, 2G (GSM/GPRS/EDGE) communication, and GNSS (GPS, BeiDou, GLONASS) positioning capabilities. This HAT is ideal for a wide range of IoT applications, including remote monitoring, POS systems, industrial routers, and location-based services.

It supports various network protocols such as TCP/IP, HTTP(s), MQTT(s), FTP(s), and SSL, ensuring flexible connectivity. Additionally, it features Text-to-Speech (TTS) functionality and an onboard audio jack for voice communication and sound recording.

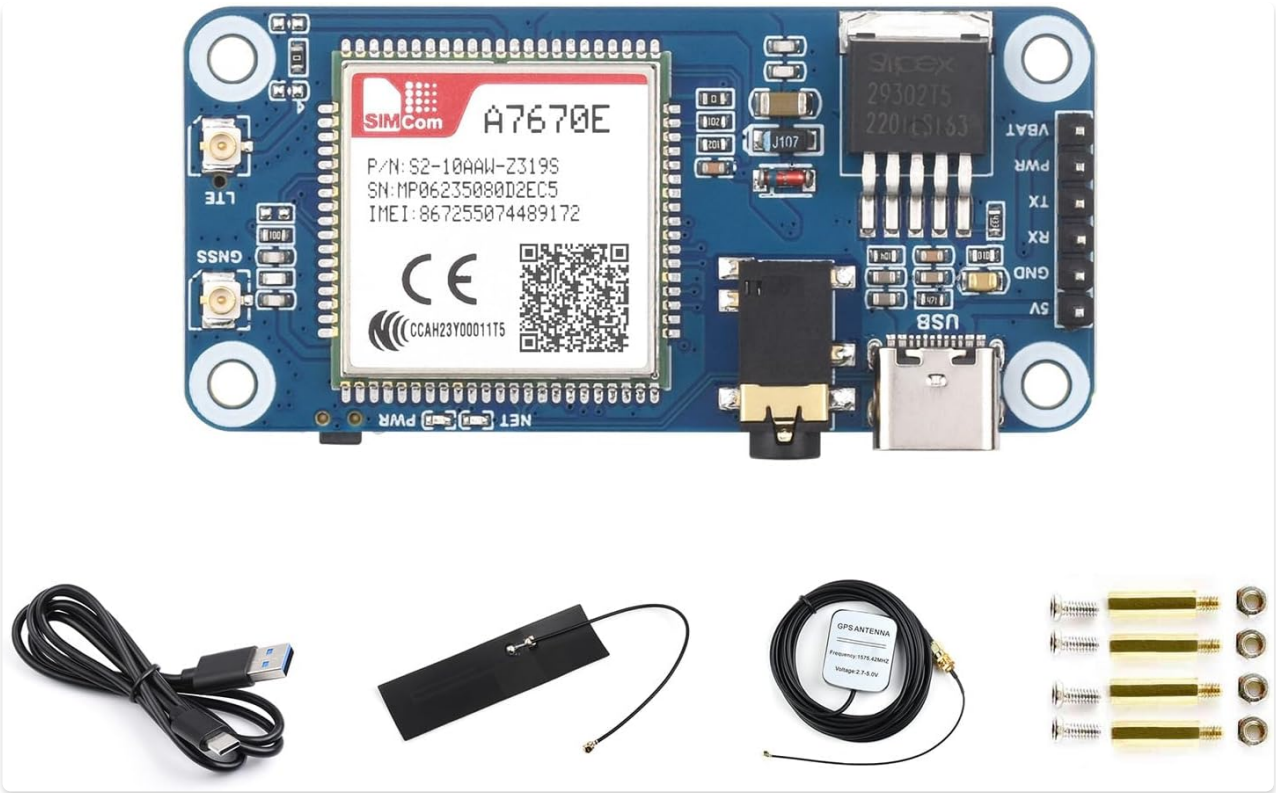


Figure 1.1: Waveshare Cat-1/GSM/GPRS/GNSS HAT with accessories.

2. Package Contents

Please verify that all items listed below are included in your package:

- A7670E Cat-1/GNSS HAT x1
- USB-A to USB-C cable x1
- FPC Antenna x1
- GPS Antenna x1
- Screws pack x1

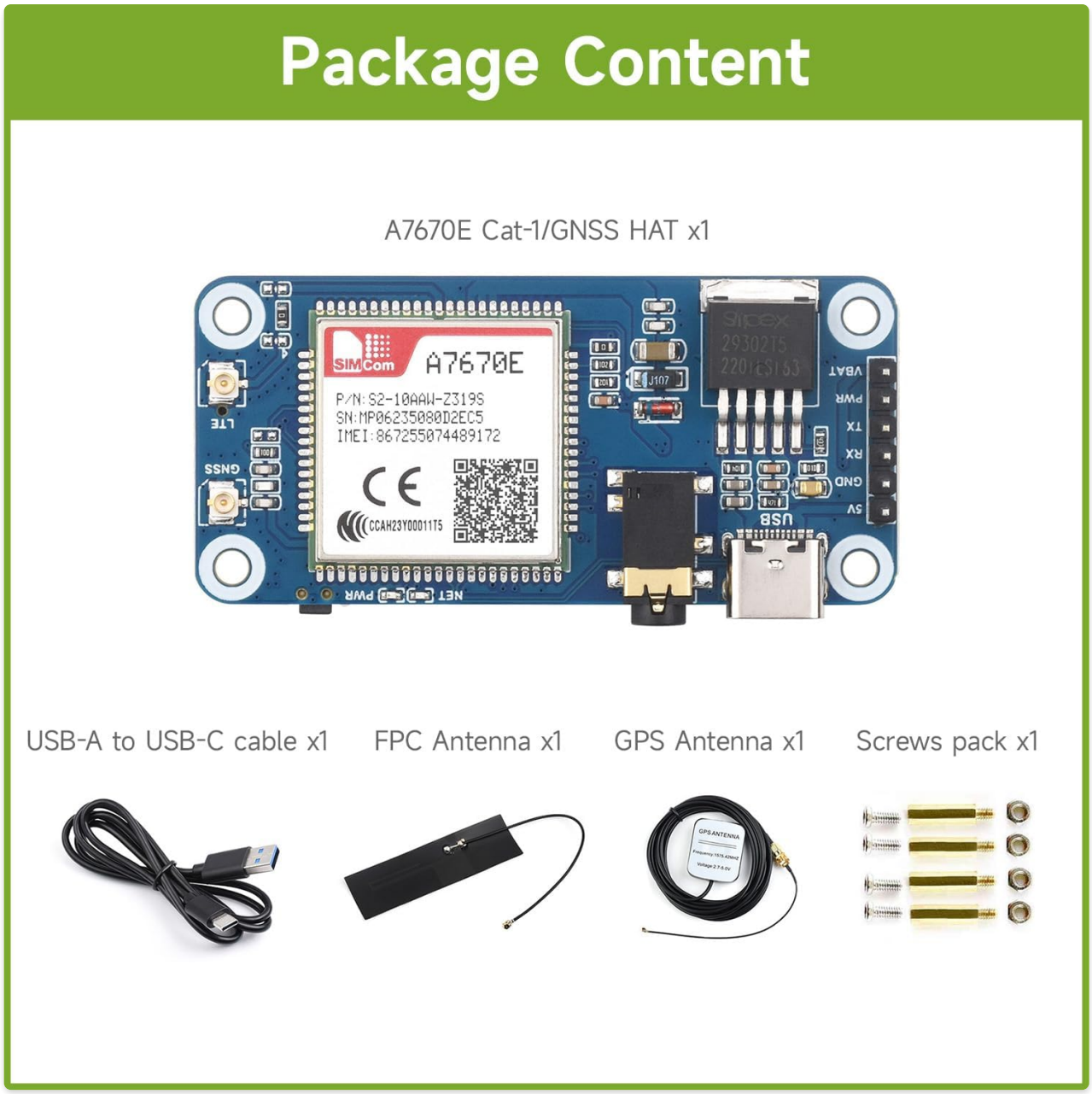


Figure 2.1: Detailed view of the package contents.

3. Product Features

The A7670E Cat-1/GNSS HAT offers a comprehensive set of features for various embedded applications:

- **Broad Network Compatibility:** Supports LTE-TDD, LTE-FDD, GSM, GPRS, and EDGE, with data rates up to 10Mbps downlink and 5Mbps uplink.
- **Operating System and Protocol Support:** Compatible with Windows and Linux. Supports TCP/IP, HTTP(s), MQTT(s), FTP(s), and SSL.
- **Positioning Capabilities:** Features GPS, BeiDou, and GLONASS support, along with LBS base station positioning for accurate geolocation.
- **Voice and Audio Functions:** Includes Text-to-Speech (TTS) for converting text to spoken words and an onboard 3.5mm audio jack for sound recording and telephone calls.
- **Easy Integration and Expansion:** Equipped with a standard Raspberry Pi 40PIN GPIO extension header for seamless integration. Also includes a SIM card slot, onboard USB interface for AT command testing,

and 2x LED indicators for status monitoring.

Figure 3.1: Overview of the A7670E Cat-1/GNSS HAT features.

4. Setup and Installation

The A7670E Cat-1/GNSS HAT is designed for easy integration with Raspberry Pi boards and other compatible host devices.

4.1 Connecting to Raspberry Pi

1. Ensure your Raspberry Pi is powered off.
2. Carefully align the 40PIN GPIO header of the HAT with the corresponding pins on your Raspberry Pi.
3. Gently press the HAT onto the GPIO header until it is securely seated.
4. Insert a compatible SIM card into the SIM card slot. The module supports both 1.8V and 3V SIM cards.
5. Connect the FPC antenna to the LTE antenna connector and the GPS antenna to the GNSS antenna connector on the HAT.

6. Optionally, connect the USB-C cable to the HAT's USB port for AT command testing and network applications.
7. Power on your Raspberry Pi.

Figure 4.1: Connecting the HAT to Raspberry Pi boards.

5. Operating Instructions

5.1 Network Communication (Dial-Up)

The module supports dial-up networking on both Windows and Linux operating systems. Data rates are up to 10Mbps (downlink) and 5Mbps (uplink) for LTE Cat-1, up to 236.8Kbps (downlink) for EDGE, and up to 85.6Kbps (downlink) for GPRS.

Figure 5.1: Example of dial-up networking setup.

5.2 GNSS Positioning

The A7670E HAT integrates GPS, BeiDou, and GLONASS for accurate global navigation satellite system positioning. It also supports LBS (Location Based Service) base station positioning, providing location information even when satellite signals are weak or unavailable.

5.3 Telephone Call & SMS Support

The module allows for making telephone calls and sending SMS messages using AT commands. The onboard 3.5mm audio jack facilitates audio input and output for calls and listening to TTS output.

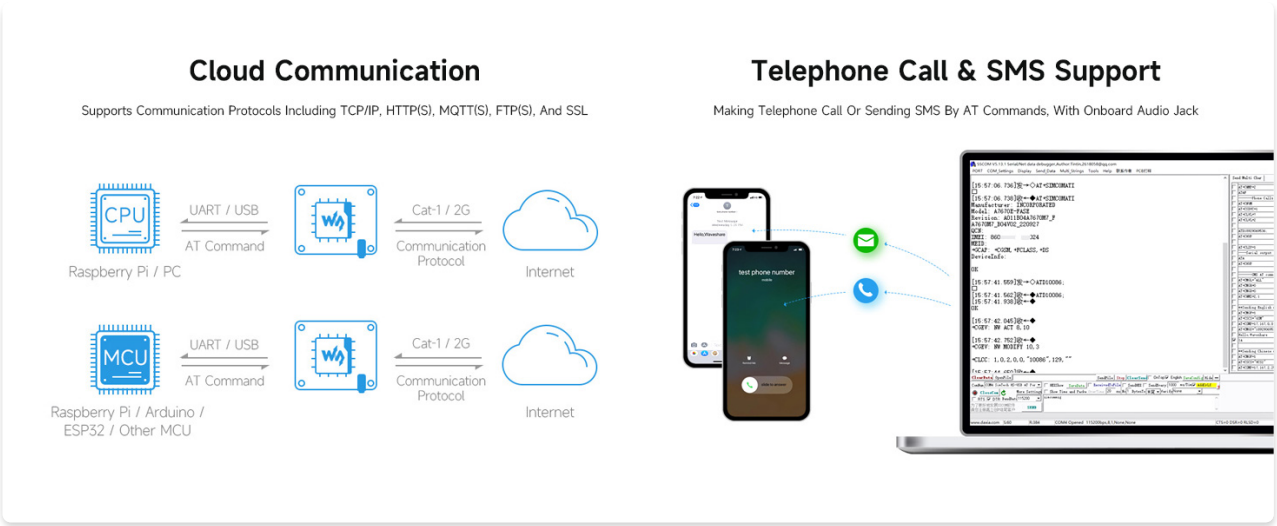


Figure 5.2: Comprehensive communication and positioning capabilities.

6. Specifications

Category	Parameter	Description
Frequency Band	LTE Cat-1	LTE-FDD: B1, B3, B5, B7, B8, B20
	2G	GSM/GPRS/EDGE: 900/1800 MHz
	GNSS	GPS, BeiDou, GLONASS
SMS and Audio	SMS	Supported types: MT, MO, CB, Text, PDU
	Audio feature	Standard 3.5mm audio jack for audio input/output
Other	Power supply	5V (via USB port or 5V pinheader), 3.7 ~ 4.2V (via VBAT pinheader, supports 3.7V Li-ion battery power input)
	Operating voltage	3.3V
	Dimensions	65 × 30.5 mm
	Weight	2.82 ounces (approx. 80g)

Specifications

FREQUENCY BAND	
LTE Cat-1	LTE-FDD: B1, B3, B5, B7, B8, B20
2G	GSM/GPRS/EDGE: 900/1800 MHz
GNSS	GPS, BeiDou, GLONASS
SMS AND AUDIO	
SMS	Supported types: MT, MO, CB, Text, PDU
	Storage: USIM card (CB couldn't be stored on SIM card)
Audio feature	Standard 3.5mm audio jack for audio input/output
OTHER	
Power supply	5V (via USB port or 5V pinheader), 3.7 ~ 4.2V (via VBAT pinheader, supports 3.7V Li-ion battery power input)
Operating voltage	3.3V
Dimensions	65 × 30.5mm

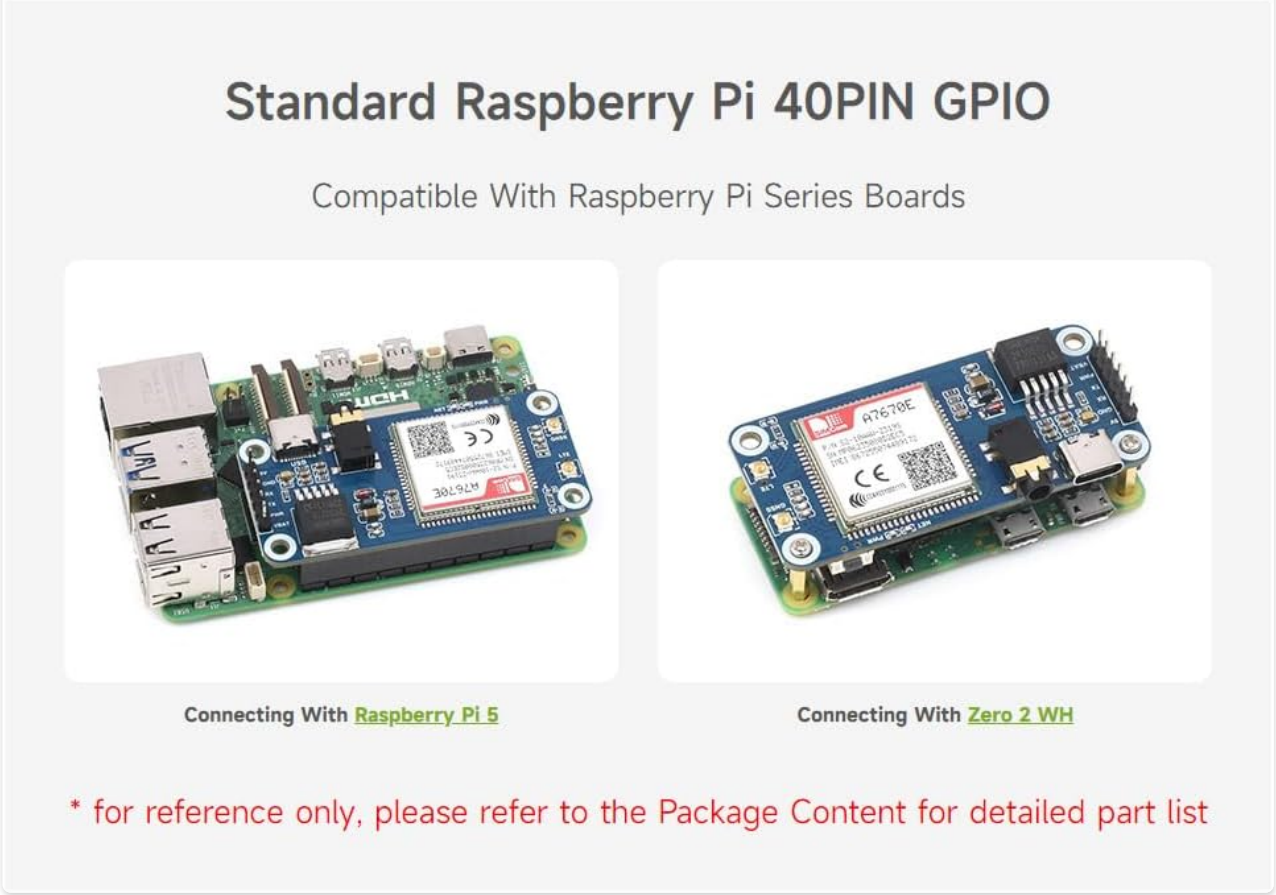


Figure 6.1: Summary of technical specifications.

7. Application Examples & Cloud Support

The A7670E Cat-1/GNSS HAT is highly suitable for various IoT and embedded applications, especially when combined with Waveshare.Cloud for data visualization and management.

7.1 General Application Example

The module can be used to upload device messages (e.g., temperature, humidity data) to a cloud server via MQTT(S) or HTTP(S). It can also receive messages for controlling devices or retrieving related data, enabling remote monitoring and control.

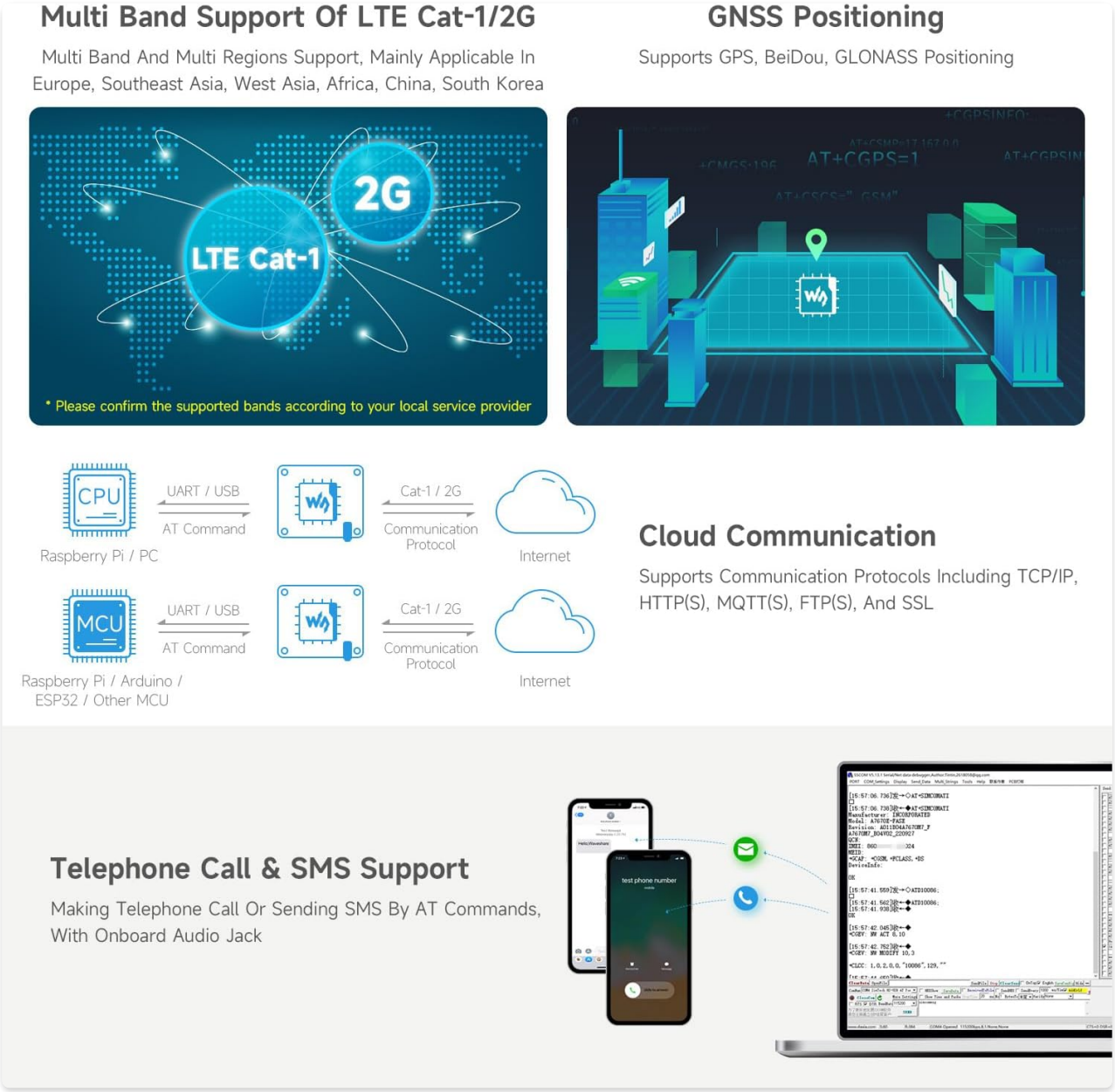


Figure 7.1: Typical application example with cloud communication.

7.2 Waveshare.Cloud Support

Waveshare.Cloud provides tutorials and demos for quick setup of smart IoT solutions. It offers large-screen data visualization displays to meet diverse application scenarios, such as smart garden solutions, server performance monitoring, and IoT GPS mobile positioning.

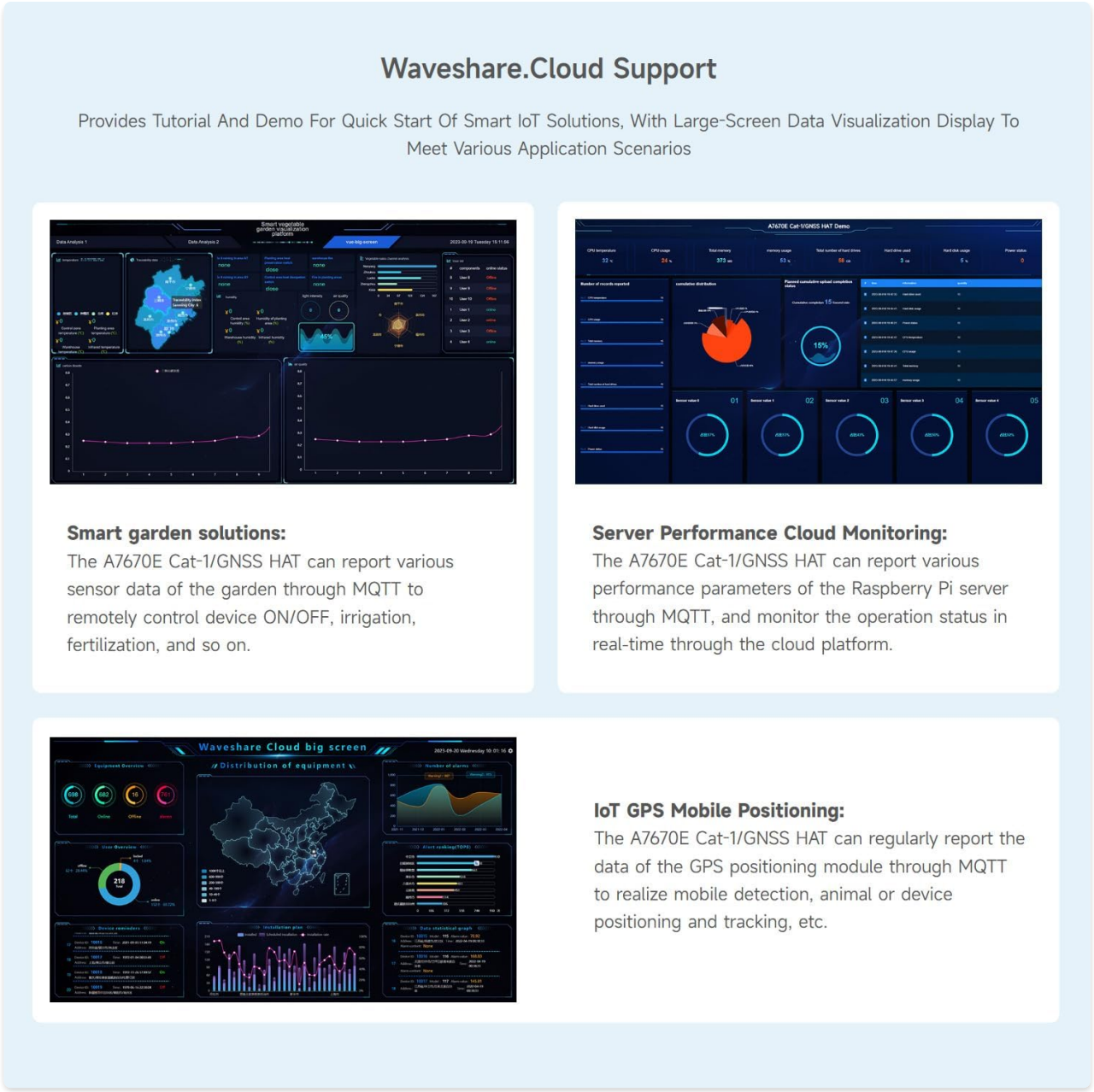


Figure 7.2: Examples of Waveshare.Cloud data visualization.

8. Pinout Definition and Dimensions

8.1 Pinout Definition

The HAT utilizes the standard Raspberry Pi 40PIN GPIO header. Key pins for module operation include:

- **5V:** 5V power input
- **GND:** Ground
- **RXD:** Module UART RX
- **TXD:** Module UART TX
- **P4:** Module ON/OFF control (keep high level for >2.5s to turn on/off)

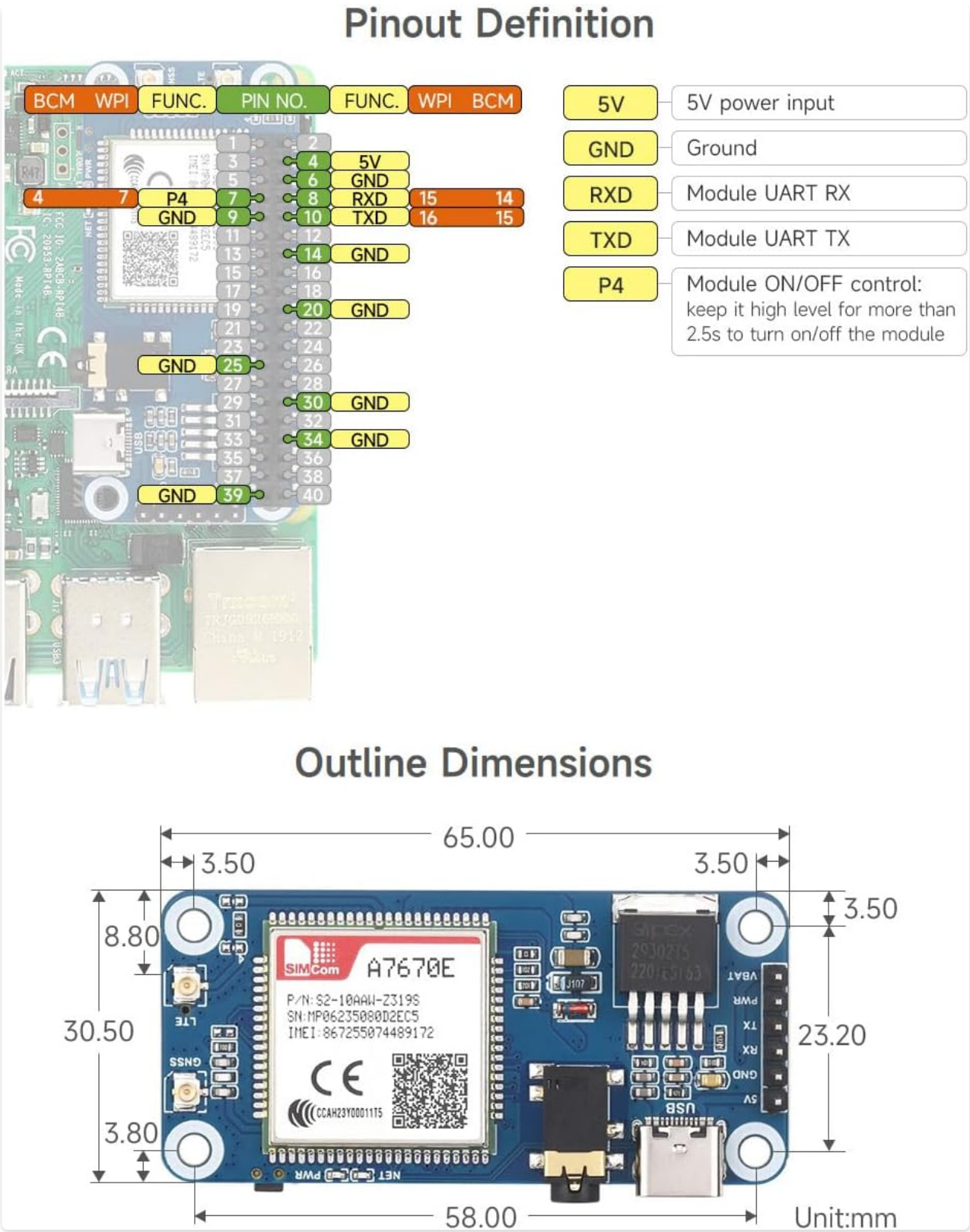


Figure 8.1: Pinout definition for the A7670E Cat-1/GNSS HAT.

8.2 Outline Dimensions

The compact design of the HAT ensures it fits well within various projects. The module dimensions are 65mm x 30.5mm.

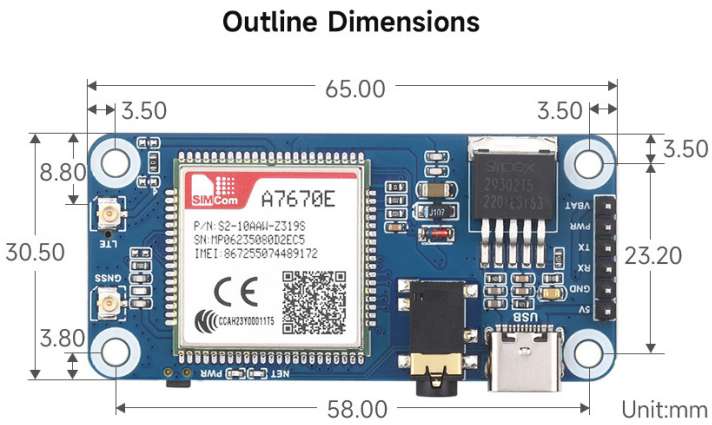


Figure 8.2: Outline dimensions of the module.

9. Troubleshooting

If you encounter issues with your Waveshare Cat-1/GSM/GPRS/GNSS HAT, consider the following common troubleshooting steps:

- **No Power/LEDs Off:** Ensure the HAT is correctly seated on the Raspberry Pi's GPIO header. Check the power supply to your Raspberry Pi. Verify the 5V power input to the HAT.
- **No Network Connection:** Confirm the SIM card is correctly inserted and activated with a valid plan. Check antenna connections (LTE and GNSS). Verify network coverage in your area. Ensure the module is powered on (P4 pin control).
- **GNSS Not Working:** Ensure the GPS antenna is connected and placed in an open area with clear sky view. Check for any obstructions.
- **AT Commands Not Responding:** Verify the USB connection or UART wiring. Ensure correct baud rate and serial port settings in your terminal software.
- **Software Issues:** Refer to the official Waveshare documentation and development resources for specific drivers, libraries, and example code for your operating system (Raspberry Pi OS, Windows, Linux).

For further assistance, please visit the Waveshare official website or contact their technical support.

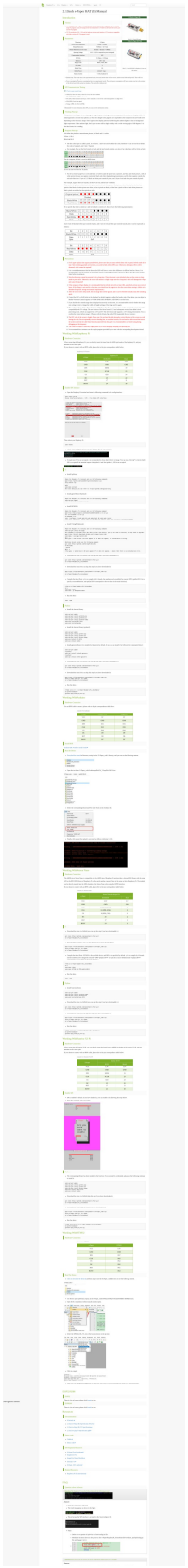
10. Maintenance

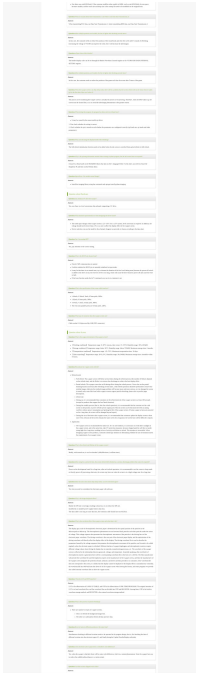
To ensure the longevity and optimal performance of your Waveshare Cat-1/GSM/GPRS/GNSS HAT, follow these maintenance guidelines:

- **Handle with Care:** Avoid dropping or subjecting the module to physical shock.
- **Keep Clean:** Use a soft, dry cloth to clean the module. Avoid using liquids or abrasive cleaners.
- **Environmental Conditions:** Operate and store the module within its specified temperature and humidity ranges. Avoid extreme temperatures, direct sunlight, and high humidity.
- **Static Electricity:** Take precautions against electrostatic discharge (ESD) when handling the module, as electronic components are sensitive to static.

- **Firmware Updates:** Periodically check the Waveshare official website for any available firmware updates for the A7670E module to ensure the best performance and compatibility.

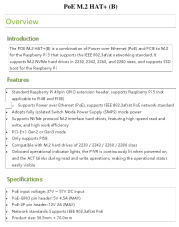
Related Documents - A7670E Cat-1/GNSS HAT

	Waveshare 2.13inch e-Paper HAT (B) User Manual and Technical Guide Comprehensive guide for the Waveshare 2.13inch e-Paper HAT (B), covering hardware connections, software setup, programming principles, and troubleshooting for Raspberry Pi, Arduino, Jetson Nano, and STM32.



[PI4-CASE-4G-5G-M.2 Assembly Tutorial: Install Raspberry Pi 5G HAT]

Comprehensive assembly tutorial for the PI4-CASE-4G-5G-M.2, designed to house a Raspberry Pi 4 with a 4G/5G M.2 module. Follow these step-by-step instructions to install your SIM card, antennas, and mounting hardware for a complete setup.



[Waveshare PoE M.2 HAT+(B) for Raspberry Pi 5: Installation and User Guide]

A comprehensive guide to installing and configuring the Waveshare PoE M.2 HAT+(B) for Raspberry Pi 5, enabling Power over Ethernet and M.2 NVMe SSD support for enhanced performance and storage.

the e-Paper is not used for a long time, you should use the program to clear the screen before storing it. (Refer to the datasheet for specific storage environment requirements.)

5. After the screen enters sleep mode, the sent image data will be ignored, and it can be refreshed not ready only after initializing again.
6. Command 0x3C of 0x03 (refer to the datasheet for details) register to specify the border. It can be set through the configuration interface of the demo. (The picture was not set originally. Test it, subject to the specific ship.)
7. If you find that the program cannot be run normally, or the screen displays abnormally, it is recommended that you check the wiring first. If there is no problem, please check whether the power supply can supply enough current, and the wiring should not be too long. If the current is insufficient or the wiring is too long, the screen may not display properly.
8. The working voltage of the e-Paper module is 3.3V. If you want to use it on a pin with a 5V logic level, you need to add a level conversion circuit for compatibility with 5V logic. The new version of the driver board (V2.1 and subsequent versions) has added a level conversion circuit, which can support both 3.3V and 5V. The old version only supports a 3.3V working environment. You can identify the version (pattern using 3). (The one with the 20-pin cap on the PCB is generally the new version.)
9. The FPC cable of the screen is relatively fragile, pay attention to bending the cable along the horizontal direction of the screen when using it, and do not bend the cable along the vertical direction of the screen.
10. The screen of e-Paper is relatively fragile, please try to avoid dropping, bumping, and pressing hard.
11. We have carefully and thoroughly test and verify the program (STM32 and Arduino) before leaving the factory to ensure that the product you receive is in good working condition. However, if you encounter any problems during use, please do not hesitate to contact us. We will provide timely assistance and resolve the issue as soon as possible to minimize any inconvenience caused.

We provide several examples for display, you can refer to them to write your own program based on your e-Paper. If you are using a different platform, you need to write the program yourself according to the corresponding pin configuration. If your display doesn't work properly, please check the pin connections carefully. Incorrect wiring may prevent the screen from displaying correctly. If you have any questions or encounter issues during development, feel free to contact us. We will provide support and help you resolve them as quickly as possible.

We provide several examples for display, you can refer to them to write your own program based on your e-Paper. If you are using a different platform, you need to write the program yourself according to the corresponding pin configuration. If your display doesn't work properly, please check the pin connections carefully. Incorrect wiring may prevent the screen from displaying correctly. If you have any questions or encounter issues during development, feel free to contact us. We will provide support and help you resolve them as quickly as possible.

Users Guides of Raspberry Pi

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 jumper cable properly.

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

Connect to Raspberry Pi:

e-Paper	Raspberry Pi	
	BCM2835	pin#
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

Take the 7.5inch HD e-Paper (B) connected to the e-paper Driver HAT as an example, just plug it directly into the Raspberry Pi:

Enable SPI interface

The Raspberry Pi series interface of e-Paper is SPI, so use it, we should firstly enable the SPI interface.

Open terminal of Raspberry Pi and open the configuration by the following command:

```
sudo raspi-config
```

Choose Interfacing Options -> SPI -> Yes

Restart Raspberry Pi

```
sudo reboot
```

Install libraries

If you are using the C program, please use "sudo" to get the root permission, otherwise, the program may fail to read the configuration file.

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

- For BCM2835 library:

```
#Open the Raspberry Pi terminal and run the following command
wget http://www.airspayce.com/mikem/bcm2835/bcm2835-1.71.tar.gz
tar zxvf bcm2835-1.71.tar.gz
cd bcm2835-1.71/
sudo ./configure && sudo make && sudo make check && sudo make install
# For more information, please refer to the official website: http://www.a
```

- For WiringPi library:

```
#Open the Raspberry Pi terminal and run the following command
sudo apt-get install wiringpi
#For Raspberry Pi models after May 2019 (earlier than 2019, you may not
#need to maintain, you may need to upgrade)
wget https://project-downloads.drogon.net/wiringpi-latest.deb
sudo dpkg -i wiringpi-latest.deb
gpio -v
#You will see version 2.52 will appear. If it does not appear, the ins
#allation is wrong
#Bullseye branch system use the following command:
git clone https://github.com/WiringPi/WiringPi
cd WiringPi
./build
gpio -v
# Run gpio -v and version 2.52 will appear. If it does not appear, it must
# then there is an installation error
```

- For Python2 library:

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

- For Python3 library:

```
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 run the following command to download the demo codes:

Method 1: Download the zip file directly:

```
sudo apt-get install p7zip-full
wget -O epaper_code.7z https://www.waveshare.com/w/upload/7/78/E-Paper_code.7z
7z x epaper_code.7z -r -o./epaper_code
cd ./epaper_code/RaspberryPi_JetsonNano/
```

Method 2: Use GitHub

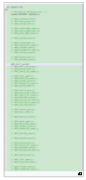
```
git clone https://github.com/waveshare/e-Paper
cd e-Paper/RaspberryPi_JetsonNano/
```

Examples

C

Open main.c to select the screen of the corresponding size.

```
cd c
sudo make examples/Test.c
```



If you are using 2.7inch e-paper, you should use 1 to select it.

After 2007 BMP3 formats, then press Ctrl+C, press the Y key and Enter to save the bmp, as shown in the figure. (The picture was not set originally. Test it, subject to the specific ship.) supporting types:

1.02inch (128 × 80) :

EPD_1in02_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 was discontinued.

EPD_1in54_V2_test(): Example for 1.54inch e-paper V2 ; (Black/White)

Purchase date later than November 2019 is the V2 version, the current shipping version has the V2 logo on the back.

EPD_1in54b_test(): Example for 1.54inch e-paper B ; (Black/White/Red). It is currently out of production.

EPD_1in54b_V2_test(): Example for 1.54inch e-Paper B V2 (Black/White/Red). The purchase date later than November 2019 is the V2 version, the current shipping version has the V2 logo on the back.

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

1.64inch (168 × 168)

EPD_1in64g_test(): Example for 1.64inch e-paper G (Black/White/Red/Yellow)

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) ;

EPD_2in7b_V2_test(): Example for 2.7inch e-paper B V2 (Black/White/Red). The purchase date is later than February 2021 is the V2 version, the current shipping version has the V2 logo on the back

2.9inch (296 × 128)

EPD_2in9_test(): Example for 2.9inch e-paper (Black/White), this version was discontinued

EPD_2in9d_V2_test(): Example for 2.9inch e-paper V2 (Black/White) ; The purchase date is later than December 2020 is the V2 version, the current shipping version has the V2 logo on the back;

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

EPD_2in9d_V3_test(): Example for 2.9inch e-paper D V3 (Black/White/Red) ; Purchase date later than July 2022 is the V3 version, the current shipping version has the V3 logo on the back;

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 was discontinued

EPD_2in13_V2_test(): Example for 2.13inch e-paper V2 ; (Black/White) ; The purchase date is later than May 2019, the V2 version is currently shipping, and the V2 logo is marked on the back;

EPD_2in13_V3_test(): Example for 2.13inch e-paper V3 ; (Black/White) ; That is the currently shipped version, with the V3 logo on the back (2020-07-29)

EPD_2in13b_test(): Example for 2.13inch e-paper B ; (Black/White/Red) and 2.13inch e-paper C ; (Black/White/Red) ;

EPD_2in13b_V3_test(): Example for 2.13inch e-paper B V3 (Black/White/Red) ; Purchase date later than April 2022 is the V3 version, the current shipping version has the V3 logo on the back;

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

2.66inch (152 × 296)

EPD_2in66_test(): Example for 2.66inch e-Paper (Black/White) ;

EPD_2in66b_test(): Example for 2.66inch e-Paper B (Black/White/Red)

3inch (400 × 168)

EPD_3in0g_test(): Example for 3inch e-paper G(Black/White/Red/Yellow)

3.52inch (360 × 240)

EPD_3in52_test(): Example for 3.52inch e-paper (Black/White)

3.7inch (280 × 480)

EPD_3in7_test(): Example for 3.7inch e-Paper(Black/White);

4.01inch (640 × 400)

EPD_4in01_test(): Example for the 4.01inch e-Paper HAT (F) (Black/White/Red/Yellow/Orange);

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) ;

EPD_4in2b_V2_test(): Example for 4.2inch e-Paper B V2 (Black/White/Red) ; The purchase date is later than September 2021, it is

the V2 version, and the current shipping version has the V2 logo on the back)

5.65inch (600 × 338)

EPD_5in65f_test(): Example for the 5.65inch e-Paper F
(/Users/Waveshare/Desktop/Example/)

5.83inch (600 × 448) :

EPD_5in83.pdf() : Example for 5.83inch e-paper (/Data/Manual) , It is currently out of production.
EPD_5in83_V2.pdf() : Example for 5.83inch e-paper V2 (/Data/Manual) , The purchase date is later than December 2020 is the V2 version, the current shipping version, with the V2 logo on the back.
EPD_5in83bc_test() : Example for 5.83inch e-paper B (/Data/Pic/Red and 5.83inch e-paper C (/Data/Pic/Yellow) ;
EPD_5in83b_V2_test() : Example for 5.83inch e-paper B V2 (/Data/Pic/Red). The purchase date is later than August 2020 for the V2 version, the current shipping version, with the V2 logo on the back.

7.3inch (800 × 480)

epd_7in3g_test.py: Example for 7inch3 e-paper G
(/Data/Pic/Red/Yellow).

7.5inch (V1 : 640 × 384 , V2 : 800 × 480) :

EPD_7in5_test() : Example for 7.5inch e-paper (/Data/Pic) , The purchase date is earlier than 2019-12-07 is the V1 version, which is currently discontinued
EPD_7in5bc_test() : For example for 7.5inch e-paper B (/Data/Pic/Red and 7.5inch e-paper C (/Data/Pic/Yellow) , The purchase date is later than 2019-12-07 is the V2 version, and the current shipping version has the V2 logo on the back
EPD_7in5_V2_test() : Example for 7.5inch e-paper V2 (/Data/Pic) , The purchase date of 7.5inch e-paper B is earlier than 2019-12-03, which is the V1 version, which is currently discontinued, and 7.5inch e-paper C is normally available.
EPD_7in5bc_V2_test() : Example for 7.5inch e-paper B V2 (/Data/Pic/Red) ; The purchase date is later than 2019-12-07 is the V2 version, with the V2 logo on the back, the purchase date is later than March 2021 is the V3 version, the current 7.5inch e-paper B shipping version, with the V3 logo on the back, the ease-use this program.

7.5inch (880 × 528)

EPD_7in5_HD.pdf() : Example for 7.5inch e-paper HD (/Data/Pic);
EPD_7in5b_HD.pdf() : Example for 7.5inch e-paper B HD
(/Data/Pic/Red);

*Note: The above two are for reference only, please refer to the screen you are specifically using

- Recompile, the compilation process may take a few seconds

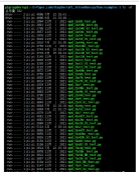
```
Return to the c directory and run the following command
make clean
make
sudo ./epd
```

Python

PS: The following are examples of the Raspberry Pi series

- Enter the root of python code.

```
cd python/examples/
ls -la
```



- Run the responding example.

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

Supporting types:

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.pdf() : Example for 1.54inch e-paper V1 (/Data/Pic) : The version in supply production when we got started 2019 11 22
epd_1in54_V2.pdf() : Example for 1.54inch e-paper V2 (/Data/Pic) : That is the currently version, which on sale since (2020-07-23). The e-paper has V2 sticker on the package.
epd_1in54b_test.py : Example for 1.54inch e-paper B (/Data/Pic/Red), It is currently out of production.
epd_1in54b_V2_test.py : Example for 1.54inch e-paper B V2
(/Data/Pic/Red). The purchase date is later than January 2021 is the V2 version, the current shipping version has the V2 logo on the back.
epd_1in54c_test.py : Example for 1.54inch e-paper C (/Data/Pic/Red)

1.64inch (128 × 160)

epd_1in64g_test.py: Example for 1.64inch e-paper G
(/Data/Pic/Red/Yellow).

2.7inch (264×176) :

epd_2in7_test.py : Example for 2.7inch e-paper (/Data/Pic) ;
epd_2in7b_test.py : Example for 2.7inch e-paper B (/Data/Pic/Red) ;
epd_2in7b_V2_test.py : Example for 2.7inch e-paper B V2
(/Data/Pic/Red). The purchase date is later than February 2021 is the V2 version, the current shipping version has the V2 logo on the back.

2.9inch (296×128) :

epd_2in9_test.py: Example for 2.9inch e-paper (/Data/Pic) ; It is currently out of production.
epd_2in9_V2_test.py: Example for 2.9inch e-paper V2 (/Data/Pic) ; The purchase date is later than December 2020 is the V2 version, the current shipping version, with the V2 logo on the back.
epd_2in9bc_test.py : Example for 2.9inch e-paper B (/Data/Pic/Red) and 2.9inch e-paper C (/Data/Pic/Yellow) ;
epd_2in9b_V3_test.py : Example for 2.9inch e-paper B V3
(/Data/Pic/Red). The purchase date later than July 2020 is the V3 version, the current shipping version has the V3 logo on the back.
epd_2in9d_test.py : Example for 2.9inch e-paper D (/Data/Pic) ;

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

epd_2in13.pdf() : Example for 2.13inch e-paper V1 (/Data/Pic) , the version in supply production only a few people before 2019 05 15
epd_2in13_V2.pdf() : Example for 2.13inch e-paper V2 (/Data/Pic) That is the current version with sticker V2 on the package (2020-07-29) ;
epd_2in13_V3.pdf() : Example for 2.13inch e-paper V3 (/Data/Pic) , The purchase date later than February 2022 is the V3 version, the current shipping version, with the V3 logo on the back.
epd_2in13bc_test.py : Example for 2.13inch e-paper B (/Data/Pic/Red) and 2.13inch e-paper C (/Data/White/Yellow) ;
epd_2in13b_V3_test.py : Example for 2.13inch e-paper B V3
(/Data/Pic/Red);
epd_2in13d_test.py : Example for 2.13inch e-paper D (/Data/Pic) ;

2.66inch (152 x 296)

epd_2in66.pdf() : Example for 2.66inch e-paper (/Data/Pic);
epd_2in66b_test.py: Example for 2.66inch e-paper (/Data/Pic/Red);

3inch (400 × 168)

EPD_3in0g_test.py: Example for 3inch e-Paper G(/Data/Pic/Red/Yellow);

3.52inch (360 × 240)

EPD_3in52_test.py: Example for 3.52inch e-Paper (/Data/Pic);

3.7inch (280 x 480)

epd_3in7.pdf(): Example for 3.7inch e-paper (/Data/Pic);

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 (/Data/Pic) ;
epd_4in2bc_test.py : Example for 4.2inch e-paper B (/Data/Pic/Red) ;
epd_4in2b_V2_test.py : Example for 4.2inch e-paper B V2
(/Data/Pic). The purchase date is later than November 2021, it is the V2 version, and the current shipping version has the V2 logo on the back;

5.65inch (600x338)

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 (/Data/Pic) , It is currently out of production.
epd_5in83_V2_test.py : Example for 5.83inch e-paper V2 (/Data/Pic) ;
The purchase date is later than December 2020 is the V2 version, the current shipping version has the V2 logo on the back.
epd_5in83bc_test.py : Example for 5.83inch e-paper B (/Data/Pic/Red and 5.83inch e-paper C (/Data/Pic/Yellow) ;
epd_5in83b_V2_test.py : Example for 5.83inch e-paper B V2
(/Data/Pic). The purchase date is later than August 2020 is the V2 version, the current shipping version has the V2 logo on the back;

7.3inch (480 × 800)

epd_7in3g_test.py: Example for 7inch3 e-paper G
(/Data/Pic/Red/Yellow).

7.5inch (V1 : 640×384 , V2 : 800×480) :

epd_7in5_test.py : Example for 7.5inch e-paper (/Data/Pic) , the version in supply production only a few people before 2019 12 07 ;
epd_7in5_V2_test.py : Example for 7.5inch e-paper V2 (/Data/Pic) , That is the current version with V2 sticker on the package (2020-07-29)
epd_7in5bc_test.py : Example for 7.5inch e-paper B (/Data/Pic/Red) and 7.5inch e-paper C (/Data/Pic/White) , 7.5inch e-paper B V1 version is supply production and ended on people before 2019 12 07
epd_7in5b_V2_test.py : Example for 7.5inch e-paper B V2
(/Data/Pic/Red) . The purchase date later than 2019-12-07 is the V2 version, with the V2 logo on the back, the purchase date later than March 2021 is the V3 version, the current 7.5inch e-paper B shipping version, with the V3 logo on the back, the ease-use this program.

7.5inch (HD 880 x 528)

epd_7in5_HD.pdf(): Example for 7.5inch e-paper HD (/Data/Pic);
epd_7in5b_HD.pdf(): Example for 7.5inch e-paper B HD
(/Data/Pic/Red);

*Note: The above two are for reference only, please refer to the screen you are specifically using

Just run the program corresponding to the screen, the program supports python2/3, take 3.54 V2 screen example.

```
# python2
sudo python2-7 epd_3in54_V2_test.py
# python3
sudo python3 epd_3in54_V2_test.py
```

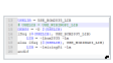
Codes Description(API)

The dynamic or deployment shared library takes care of the underlying hardware interface (GPIO, SPI, I2C), ensuring that the same upper-level API functions can be used across various hardware platforms, such as Raspberry Pi and Jetson Nano. The EPD driver and the GUI functions are implemented on top of this shared library.

C

Hardware Interface

We have carried out the underlying package. As the hardware platform is different, the internal is various. If you need to know the frames, you can go to the corresponding directory.
You can see many definitions in DEV_Config.c(.h), in the directory: RaspberryPi_JetsonNano\c\lib\Config.
C language uses two ways to drive: BCM2835 library and Wiringpi library. The BCM2835 library is used by default. If you need to use the Wiringpi library to drive, you can open RaspberryPi_JetsonNano\c\Makefile and modify lines 13-14:



- 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() functions are used to configure GPIOs. EPD enters sleep mode after Exit() function is used, and the consumption of the paper should be 0 in sleep mode if the PCB is the V2 version.

- GPIO configuration

```
void GUI_Digital_Write(UBYTE Xpt, UBYTE Ypt);
UBYTE GUI_Digital_Read(UBYTE Xpt);
```

- Set the read pin

```
void GUI_SPI_WriteByte(UBYTE Value);
```

EPD Driver

The driver file is saved under /examples/EPD_xxxx.c/h/pdf, for example:

Open .h and see the following functions

- EPD initialization: please use it when the EPD starts to work or exits the sleep mode.

```
//1.54inch e-Paper, 1.54inch e-Paper V2, 2.13inch e-Paper, 2.13inch e-Paper V2, 2.13inch e-Paper B
V3, 2.13inch e-Paper (D), 2.9inch e-Paper, 2.9inch e-Paper (D)
void EPD_xxx_Init(void);  //Only 1 GUI can be defined to initialize the screen, and the rest will not be initialized
//Clear module
void EPD_xxx_Init_Fast();
```

Where xxx represents the model. If it is 2.13D, the full-screen initialization is EPD_2IN13D_Init(), the partial refresh initialization is EPD_2IN13D_3Init(). If it is 1.54 V2, then EPD_1IN54_V2_Init(); if it is 7.5B, it is EPD_7IN5BC_Init(). Although the display colors are different, the driver code of 7.5B and 7.5C is the same.

- Clear the screen, and refresh the full-screen image.

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

Where xxx represents the model. If it is 2.13D, then it is EPD_2IN13D_Clear(); if it is 7.5B, it is EPD_7IN5_Clear(). Although the display colors are different, the driver code of 7.5B and 7.5C is the same.

- Transfer a image of picture data and open the display

```
//Black and white two-color 1bit screen
void EPD_xxx_Display(UBYTE *Image);
//Four grayscale 2bit screen, including black, white, yellow and red screen
void EPD_xxx_Display_4Gray(const UBYTE *BlackImage, const UBYTE *ryImage);
```

Sleep mode, the following are special cases:

```
//For 2.13inch e-Paper (B), 2.9inch e-Paper (B) two flexible products, partial refresh
void EPD_2IN13D_DisplayPart(UBYTE *Image);
void EPD_2IN9D_DisplayPart(UBYTE *Image);
```

After 1.54inch e-paper V2 and 2.13inch e-paper V2, due to the upgrade of the control chip, for a partial refresh, it is necessary to call EPD_xxx_DisplayPartBaseImage to display the static background image, that is, to perform parts refresh based on this image, and then call the dynamic EPD_xxx_DisplayPart()

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

- Enter sleep mode

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

Note that after entering sleep mode, there are only two ways to work again: the first is a hardware reset, and the second is to call the initialization function again.

Where xxx represents the model. If it is 2.13D, then it is EPD_2IN13D_SPRC(); if it is 7.5B, it is EPD_7IN5BC_SPRC(). Although the display colors are different, the driver code of 7.5B and 7.5C is the same.

GUI functions

GUI.exe can be found in /examples/GUI/Windows/GUI.exe, which is a Windows application

The font can be found in the /examples/Fonts/ directory

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 is a pointer of the 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 controllers, the position of the first pixel is different, here we take 1.54inch as an 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 type 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 part the 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 set window

Draw point

The function is drawing at (Xpoint, Ypoint)

```
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;
```

Draw Line

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

The function is drawing at (Xstart, Ystart) and (Xend, Yend)

- 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-axis of the rectangle
- Ystart: Start coordinate of Y-axis of the rectangle
- Xend: End coordinate of X-axis of the rectangle
- Yend: End coordinate of Y-axis of the rectangle
- Color: color of the rectangle
- Line_width: The width of 4 lines, 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: Fill the rectangle or not. Select enum:

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

Draw circle

Draw a circle: In the image cache, with (X_Center Y_Center) as the center, draw a circle with a radius of Radius. You can choose the color, the width of the line, and whether to fill the inside of the circle

```
void Paint_DrawCircle(UWORD X_Center, UWORD Y_Center, UWORD Radius, UWORD Color, DOT_PIXEL Line_width, DRAW_FILL Draw_Fill)
parameter:
  X_Center: X coordinates of the center of the circle
  Y_Center: Y coordinates of the center of the circle
  Radius: the radius of the circle
  Color: fill color
  Line_width: the width of the arc, controlling 8 default widths
  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: Fill, whether to fill the inside of the circle
    typedef enum {
        DRAW_FILL_EMPTY = 0,
        DRAW_FILL_FULL,
    } DRAW_FILL;
```

Draw character (ASCII)

Display most of the ascii characters and draw an ASCII character

```
void Paint_DrawChar(UWORD Xstart, UWORD Ystart, const char Ascii_Char, sFONT
```

```
void Paint_DrawChar(UWORD Xpoint, UWORD Ypoint, const char Acsii_Char,
    sFONT* Font, UWORD Color_Foreground, UWORD Color_Background)
```

Parameter:

- Xpoint: X coordinate of the left top pixel of character;
- Ypoint: Y coordinate of the left top pixel of character;
- Acsii_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

Show more (open) string at the left top pixel, use in turn:

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

Parameters:

- Xpoint: X coordinate of left top pixel of characters;
- Ypoint: Y coordinate of the top pixel of character;
- 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

Get Ansicon & using gt gtfew Chinese drop gamut EN GB2312 drop

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

Parameters:

- Xpoint: Coordinate of left top pixel of characters;
- Ypoint: Coordinate of left top pixel of characters;
- pString: Pointer of string;
- Font: GB2312 fonts:
  - &Font12CN: 11*12(pixel) ; 16*21 (Chinese)
  - &Font24CN: 24*41(pixel) ; 32*41 (Chinese)
- Color_Foreground: color of string;
- Color_Background: color of the background

Draw number

Draw a bit by number, (Xstart, Ystart) is the left top pixel

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

Parameter:

- Xpoint: X coordinate of the top pixel;
- Ypoint: Y coordinate of left-top pixel;
- Number: the number displayed the number saved in int format;
- Font: 5 drop gt are available
  - &Font8: 5*8
  - &Font12: 7*12
  - &Font16: 11*16
  - &Font20: 14*20
  - &Font24: 17*24
- Color_Foreground: the color of the font;
- Color_Background: the color of background;

Draw image

Show image drop of BMP file to buffer

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

Parameters:

- image_buffer: address of image drop in the buffer

Read local bmp picture and write it to buffer.

LINUX project: get_bmp.h bmp.c bmp.h bmpdebug.h support, already optimized (BGR/RGB, bmpdebug.h is bmp.c bmp.h bmpdebug.h support, already optimized (BGR/RGB, bmpdebug.h)

```
UBYTE GUI_ReadBmp(const char *path, UBYTE Xstart, UBYTE Ystart)
```

Parameters:

- Path: The path of the program
- Xstart: X coordinates of left top of picture, default is 0
- Ystart: Y coordinates of left top of picture, default is 0

Testing Code

The first one chapters introduce the classic Linux three-layer code structure, here is a little explanation of the user test code. for Raspberry Pi and Jetson Nano, in the directory: RaspberryPi_JetsonNano\c\, examples, for all test codes, multiple choices can be made in main in the directory: (note: relay and run gui_2.9inch_e-paper_test programs, you may encounter the problem of insufficient power supply, so it is recommended to use an external power supply to power the e-Paper)

```
./EPD_2in9_test.exe
```

or

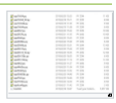
```
sudo ./EPD_2in9_test.exe
```

Then compile it again and run.

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

Python

For Jetson Nano (RaspberryPi, being run on python2.7 and python3.
For python, all codes are not encapsulated as .C
RaspberryPi and Jetson Nano:
RaspberryPi_JetsonNano\python\EPD_test.py



Bottom Interface

The epdconfig.py file
packages the underlying interface

- Initialize module and exit module：

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

Note:
1. The functions are used to set GPIO (which should be set before using e-Paper
2. If the operating system is printed with BCM 1, the module exits the interface and after Module_EXIT, the set will be the current output 0 in the end;

- GPIO initialize：

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

- SPI Write data:

```
def spi_writebyte(data)
```

Driver Interface

epd2in9.py (the name size, if it is 2.13inch e-paper, it is epd2in13.py, and so on)

- Initialize e-paper: use function epd.init() to use the beginning. It can also be used to wake up e-paper (the same wake

```
#for 2.13inch e-Paper, 2.13inch e-Paper
def init(self, update): # Choose Full_Init_Update or Set_Partial_Update
# Full_Init_Update
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 drop and display

```
#for two-color e-Paper
def display(self, image):

#Because four resolutions of 2.13inch e-Paper are supported, when partial re
fresh, they should first use displayPartBaseImage() to display one-frame
picture, then use displayPartial to dynamically display.
def displayPartBaseImage(self, image)
def displayPartial(self, image)
```

- Enter sleep mode

```
def sleep(self)
```

Testing Function

epd_xxx_test.py (the name depends on python version) are saved in the directory:
RaspberryPi and Jetson Nano：RaspberryPi\python\examples\

If the python installed in your OS is python3, you should run examples like the below:

```
sudo python epd_2in13_V2_test.py
```

If it is python3, the commandshould 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
```

The rotation effect, shall: 1.Rotate an image, in 0°, 90°, 180°, 270° in sequence

GUI

Python have powerful PIL library s, which can be used directly to drawing figures. Here we use it for drawing.

- Install the library library:

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

Import the library

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

Import library: Import image, drawing function: ImageDraw; font

- Set image buffer for drawing.

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

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 the 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, 210, 39), fill = 0)
```

The first parameter is a type of coordinates. (x, y) is the starting point, and (210, 39) is the endpoint. The fill=0 means the color is black.

- Draw line

```
draw.line((14, 53, 94, 53), fill = 0)
```

The first parameter is a type of coordinates. (x, y) is the beginning point, and (210, 39) is the endpoint. The fill=0 sets the line to black.

- Draw circle

```
draw.arc((90, 83, 155, 148), 0, 360, fill = 0)
```

The function is used to draw a circle of a square. The first parameter is a type 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 coordinates, 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, 155, 195), 0, 360, fill = 0)
```

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

- Character

You can directly import the ImageFont.ttc for drawing characters:

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

You can use the font of the system or import the font which in the picture.
To draw English characters, you can directly use the font; for Chinese characters, you need to import the font.

```
draw.text((0, 13), 'hello world', font = font, fill = 255)
draw.text((0, 53), u'微雪电子', font = font, fill = 0)
```

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

- Read bmp picture

```
image = Image.open(os.path.join(picdir, 'image.bmp'))
```

The parameter is the picture path.

- Other functions

For more information about the PIL library, you can search on the.

User Guides of Jetson Nano

Hardware connection

Jetson nano supports GPIO, I2C, SPI, UART and other communication methods. Waveshare e-Paper HAT is compatible with the Raspberry Pi GPIO interface, so it is compatible with Jetson nano GPIO interface. Waveshare e-Paper HAT can be directly connected to Jetson nano GPIO interface.
To connect the e-paper, you can refer to the following table and image.

Jetson Nano Connection

e-Paper	Jetson Nano Developer Kit	
	BOARD	Pin #
VCC	3.3V	3.3V
GND	GND	GND
DIN	19(SPI1_MOSI)	19
CLK	23(SPI1_SCK)	23
CS	24(SPI1_CS0)	24
DC	25	22
RST	17	11
BUSY	24	18

Software Setting

Install Libraries

Open the 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 /usr/local/lib/python3.6/dist-packages/Jetson.GPIO-0.1.0-py3.6.egg-info/99-gpio.rules /etc/udev/rules.d/
sudo udevadm control --reload-rules && sudo udevadm trigger
```

- Install I2C library:

```
sudo apt-get install python3-smbus
```

- Install PIL library:

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

Download Testing Program

Open a new terminal and execute:
Method 1: Using the GitHub repository

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

Run the Test Program

Please execute the following commands under /home/pi/e-Paper/RaspberryPi_JetsonNano/, otherwise, the directory will not be found.

C

- Go to the /c/ directory and run the corresponding make command:

```
cd c
sudo make clean && sudo make
```


If you are using 2.7inch e-paper, then remove the // in front of the EPD_2IN7.h#include, then remove the // in front of the EPD_2IN7_test() function, and then comment out all the functions of the other e-Paper.

Supporting Types

1.02inch (128×80)
EPD_1IN02.h#include: Header file for 1.02inch e-Paper/1.02inch e-Paper Module

1.54inch (1.54inch e-paper C : 152×152 , others : 200×200)
EPD_1IN54.h#include: Header file for 1.54inch e-Paper V1 (Black/White) ;
The version supporting this program was on sale in March 2019 and before;
EPD_1IN54_V2.h#include: Header file for 1.54inch e-Paper V2 (Black/White) ; The e-paper currently on sale (on sale since 2020-07-20). The e-paper size V2 is larger on the package.
EPD_1IN54b.h#include: Header file for 1.54inch e-Paper B (Black/White/Red). It is currently out of production.
EPD_1IN54b_V2.h#include: Header file for 1.54inch e-Paper B V2 (Black/White/Red). The purchase date is later than January 2019 is the V2 version, the current shipping version is the V2 logo on the back;
EPD_1IN54c.h#include: Header file for 1.54inch e-Paper C (Black/White/Red) ;

1.64inch (128 x 96)
EPD_1IN64g.h#include: Header file for 1.64inch e-paper G (Black/White/Red/Yellow)

2.7inch (264×176)
EPD_2IN7.h#include: Header file for 2.7inch e-paper (Black/White) ;
EPD_2IN7b.h#include: Header file for 2.7inch e-paper B (Black/White/Red) ;
EPD_2IN7b_V2.h#include: Header file for 2.7inch e-paper B V2 (Black/White/Red) ;

2.9inch (296×128)
EPD_2IN9.h#include: Header file for 2.9inch e-paper (Black/White), this version is stopped production.
EPD_2IN9_V2.h#include: Header file for 2.9inch e-paper V2 (Black/White) ; The purchase date is later than December 2019 is the V2 version, the current shipping version can be V2 logo on the back;
EPD_2IN9bc.h#include: Header file for 2.9inch e-paper B (Black/White/Red) and 2.9inch e-paper C (Black/White/Yellow) ;
EPD_2IN9d.h#include: Header file for 2.9inch e-paper D (Black/White) ;

2.13inch (2.13inch e-Paper : 250×122 , others : 212×104)
EPD_2IN13.h#include: Header file for 2.13inch e-paper V1 (Black/White) , the version is supporting program only a can be used to before 2019 05 15
EPD_2IN13_V2.h#include: Header file for 2.13inch e-paper V2 (Black/White) ; That is the currently version was listed V2 on the package (2020-07-20)
EPD_2IN13_V3.h#include: Header file for 2.13inch e-paper V3 (Black/White) ; That is the currently version was listed V2 on the package (2020-07-20)
EPD_2IN13bc.h#include: Header file for 2.13inch e-paper B (Black/White/Red) and 2.13inch e-paper C (Black/White/Yellow) ;
EPD_2IN13b_V3.h#include: Header file for 2.13inch e-paper B V3 (Black/White/Red) ; Purchase date after April 2020 is the V3 version, the current shipping version can be V3 logo on the back;
EPD_2IN13d.h#include: Header file for 2.13inch e-paper D (Black/White) ;

2.66inch (152 x 296)
EPD_2IN66.h#include: Header file for 2.66inch e-Paper (Black/White) ;
EPD_2IN66b.h#include: Header file for 2.66inch e-Paper B (Black/White/Red) ;

3inch (168 x 168)
EPD_3IN0g.h#include: Header file for 3inch e-Paper G(Black/White/Red/Yellow) ;

3.52inch (360 x 240)
EPD_3IN52.h#include: Header file for 3.52inch e-Paper (Black/White) ;

3.7inch (280 x 480)
EPD_3IN7.h#include: Header file for 3.7inch e-Paper(Black/White) ;

4.01inch (640x400)
EPD_4IN01.h#include: Header file for the 4.01inch e-Paper HAT (F) (Black/White/Green/Blue/Red/Yellow/Orange) ;

4.2inch (400×300)
EPD_4IN2.h#include: Header file for 4.2inch e-paper (Black/White) ;
EPD_4IN2b.h#include: Header file for 4.2inch e-paper B (Black/White/Red) ;
EPD_4IN2b_V2.h#include: Header file for 4.2inch e-paper B V2 (Black/White/Red) ; The purchase date is later than September 2020, is the V2 version, and the current shipping version can be V2 logo on the back;

5.65inch (600x448)
EPD_5IN65F.h#include: Header file for 5.65inch e-paper F (Black/White/Green/Blue/Red/Yellow/Orange) ;

5.83inch (600×448)
EPD_5IN83.h#include: Header file for 5.83inch e-paper (Black/White) , it is currently discontinued;
EPD_5IN83_V2.h#include: Header file for 5.83inch e-paper V2 (Black/White) , The purchase date is later than December 2020 is the V2 version, the current shipping version, with the V2 logo on the back;
EPD_5IN83bc.h#include: Header file for 5.83inch e-paper B (Black/White/Red) and 5.83inch e-paper C (Black/White/Yellow) ;
EPD_5IN83b_V2.h#include: Header file for 5.83inch e-paper B V2 (Black/White/Red) ; The purchase date is later than August 2020 for the V2 version, the current shipping version, with the V2 logo on the back;

7.5inch (V1 : 640×384 , V2 : 800×480)
EPD_7IN5.h#include: Header file for 7.5inch e-paper (Black/White) ; The purchase date is earlier than 2019-12-07 is the V1 version, which is currently discontinued;
EPD_7IN5bc.h#include: For example for 7.5inch e-paper B (Black/White/Red) and 7.5inch e-paper C (Black/White/Yellow) , The purchase date later than 2019-12-07 is the V2 version, and the current shipping version has the V2 logo on the back;
EPD_7IN5_V2.h#include: Header file for 7.5inch e-paper V2 (Black/White) ; The purchase date of 7.5inch e-paper B is earlier than 2019-12-03, which is the V1 version, which is currently discontinued and 7.5inch e-paper C is normally available.
EPD_7IN5bc_V2.h#include: Header file for 7.5inch e-paper B V2 (Black/White/Red) ; The purchase date is later than 2019-12-07 is the V2 version, with the V2 logo on the back; the purchase date is later than March 2021 is the V3 version, the current 7.5inch e-paper B shipping version, with the V3 logo on the back, the same use this program;

7.5inch(HD)(880 x 528)
EPD_7IN5_HD.h#include: Header file for 7.5inch e-paper HD (Black/White) ;
EPD_7IN5b_HD.h#include: Header file for 7.5inch e-paper B HD (Black/White/Red)

Note: The above table is for reference only, please refer to the actual object for the specific version.

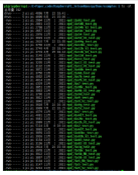
- Recompile, the compilation process may take a few seconds.

```
Return to the c directory and run the following command
make clean
make -Jetson
sudo ./epd
```

Python

Enter the python progress directory and run the corresponding *.py.

```
cd python/examples/
ls -l
```



Support Types:

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 be bought 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). It is currently out of production.

epd_1in54b_V2_test.py : Example for 1.54inch e-paper B V2 (Black/White/Red). The purchase date is after 2021 2020 in the V2 version, the current shipping version has the V2 logo on the back.

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

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) ;

epd_2in7b_V2_test.py : Example for 2.7inch e-paper B V2 (Black/White/Red). The purchase date is after then February 2021 in the V2 version, the current shipping version has the V2 logo on the back.

2.9inch (296×128)

epd_2in9_test.py: Example for 2.9inch e-paper (Black/White) ; It is currently out of production.

epd_2in9_V2_test.py: Example for 2.9inch e-paper V2 (Black/White) ; The purchase date is later than December 2020 in the V2 version, the current shipping version has the V2 logo on the back.

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

epd_2in9b_V3_test.py : Example for 2.9inch e-paper B V3 (Black/White/Red). The purchase date later than July 2020 in the V3 version, the current shipping version has the V3 logo on the back.

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) , the 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_2in13_V3_test.py : Example for 2.13inch e-paper V3 (Black/White) , The purchase date later than February 2022 is the V3 version, the current shipping version, with the V3 logo on the back.

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

epd_2in13b_V3_test.py : Example for 2.13inch e-paper B V3 (Black/White/Red)

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

2.66inch (152 x 296)

epd_2in66_test.py: Example for 2.66inch e-paper (Black/White);

epd_2in66b_test.py: Example for 2.66inch e-paper (Black/White/Red);

3inch (400 × 168)

EPD_3in0_test.py: Example for 3inch e-paper G2 (Black/White/Yellow);

3.52inch (360 x 240)

EPD_3in52_test.py: Example for 3.52inch e-paper (Black/White);

3.7inch (280 x 480)

epd_3in7_test.py: Example for 3.7inch e-Paper (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) ;

epd_4in2b_V2_test.py : Example for 4.2inch e-paper B V2 (Black/White/Red). The purchase date is later than November 2020, it is the V2 version, and the current shipping version has the V2 logo on the back.

5.65inch (600×448)

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). It is currently out of production.

epd_5in83_V2_test.py : Example for 5.83inch e-paper V2 (Black/White) ; The purchase date is later than December 2020 is the V2version, the current shipping version has the V2 logo on the back.

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

epd_5in83b_V2_test.py : Example for 5.83inch e-paper B V2 (Black/White/Red). The purchase date is later then August 2020 in the V2 version, the current shipping version has the V2 logo on the back.

7.5inch (V1 : 640×384 , V2 : 800×480)

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

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

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 V3 version is stopping production and it can be bought before 2019-12-03

epd_7in5b_V2_test.py : Example for 7.5inch e-paper B V2 (Black/White/Red) , The purchase date later than 2019-12-07 is the V2 version, with the V2 logo on the back, the purchase date later than March 2021 is the V3 version, the current 7.5inch e-paper B shipping version, with the V3 logo on the back, the issue has been progress.

7.5inch (HD 880 x 528)

epd_7in5_HD_test.py: Example for 7.5inch e-paper HD (Black/White);

epd_7in5b_HD_test.py: Example for 7.5inch e-paper B HD (Black/White/Red);

*Note: The above (just a for reference only, please refer to the screen you for the specific version

- Add run the program corresponding to the screen, the program supports Python2.7/3 and 3.54 V2 and above versions

```
# python2
sudo python2 7.sub_1in54_V2_test.py
# python3
sudo python3 epd_1in54_V2_test.py
```

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 examples, WiringPi, and BCM2835. The codes use wiringPi by default, if you want to use BCM2835, you can modify the /lib/config/dev_hardware_SPI.h file, modify lines 13 and 14. Change it as below:

```
//#define USE_BCM2835_LIB
#define USE_WIRINGPI_LIB
//#include <bcm2835.h>
#include <wiringPi.h>
#include <wiringPiSPI.h>
```

- DEV_GPIO

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

- 打印 debug 信息

```
void DEV_Delay_ms(UDOUBLE xms);
void DEV_Module_Init(void);
```

Note: The init() and Init() function are used to configure GPIOs . GPIO may sleep via gpio enter Init() function to save it, and the consumption of e-Paper Hat(s) (or it will keep as idle if the PCB is Rev2.1 version.

- GPIO 和 硬件SPI 读

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

- SPI transmit data

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

EPD

e-paper driver code file, in the directory: /lib/e-Paper/, /lib/e-Paper/

As shown below:

Open .h to see the following functions

- After the initialization, as first when the screen starts working and after exiting sleep mode.

```
//1.54inch e-Paper, 1.54inch e-Paper V2, 2.13inch e-Paper, 2.13inch e-Paper V2, 2.13inch e-Paper V3, 2.66inch e-Paper, 2.9inch e-Paper, 2.9inch e-Paper V2
//Full screen refresh initialization, the first set needs to be initialized with a global refresh initialization,
//and the second call to fast initialization can be used.
//Enter module
void EPD_xxx_Init(void);
```

Where xxx represents the model. If it is 2.13D, the full-screen initialization is EPD_2IN13D_Init(), the partial refresh initialization is EPD_2IN13D_3Init() ; If it is 1.54 V2, then EPD_1IN54_V2_Init(). If it is 7.5B, it is EPD_7IN5BC_Init(), because 7.5B and 7.5C share the driver code, but the color displayed is different.

- Transfer a piece of picture data and open the display

```
//Black and white two-color 16K screen
void EPD_xxx_Display(UBYTE *BlackImage);
//Display the specified area of the black and white two-color 16K screen
void EPD_xxx_DisplayWindows(UBYTE *BlackImage, UWORD Xstart, UWORD Ystart);
```

如果 part 显示 则Display 会 调用此函数:

```
//For 2.13inch e-paper (B), 2.9inch e-paper (B) two flexible screens, partial refresh
void EPD_2IN13D_DisplayPart(UBYTE *Image);
void EPD_2IN9D_DisplayPart(UBYTE *Image);
```

//for 1.54inch e-paper V2and 2.13inch e-paper V2, due to the upgrade of the control chip, for a partial refresh, it is necessary to call EPD_xxx_DisplayPartBaseImage to display the static background image, that is, to perform parts / refresh based on this image, and then call the dynamic EPD_xxx_DisplayPart ()

```
void EPD_1IN54_V2_DisplayPartBaseImage(UBYTE *Image);
void EPD_2IN13_V2_DisplayPart(UBYTE *Image);
void EPD_2IN13_V3_DisplayPartBaseImage(UBYTE *Image);
```

- 进入 sleep 休眠

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

Note that after entering sleep mode, there are only two ways to work again: the first is a hardware reset, and the second is to call the Init() initialization function again.

Where xxx represents the model. If it is 2.13D, then it is EPD_2IN13D_Sleep(). If it is 7.5B, it is EPD_7IN5BC_Sleep(), because 7.5B and 7.5C share the driver code, but the color displayed is different.

GUI Functions

GUI 的 API 可以 在 /lib/GUI/ 目录下找到 (GUI_Paint.h/GUI_BMPfile.h/GUI_Paint.c)

The fonts can be found in /lib/fonts/ directory

Create an image buffer

- pwidth: the image buffer
- Width: width of the image
- Height: height of the image
- Scale: display angle
- Color: clip or the image

Select Image buffer

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

Rotate Image

The function should be used after Paint_SelectImage()

- Rotate: The angle rotating. 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 block as example

Mirroring

- 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

- 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

- Color: The color of the display

Clear window

This function is used to clear a part the same

- 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 ending point
- Yend: The y-axis coordination of the ending point
- Color: The color of the window

Draw point

The function is using to draw points

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

- Dot_Style: The style of the point

Drawn Line

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 ending x-axis coordination of the line
- Yend: The ending y-axis coordination of the line
- Line_width: The width of the line

- Line_style: The style of the line

Draw rectangle

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

- Xstart: (Xstart, Ystart) are start point of a corner of the rectangle
- Ystart: (Xstart, Ystart) are start point of a corner of the rectangle
- Xend: (Xend, Yend) are end point of a diagonal of the rectangle
- Yend: (Xend, Yend) are end point of a diagonal of the rectangle
- Color: color of the rectangle
- Line_width: The width of the lines, a square are selected

- Draw_Fill: Fill the rectangle or not or empty

- Draw a circle: (X_Center, Y_Center) is the center of the circle, Radius is the radius of the circle, Color is the color of the circle, Line_width is the width of the arc, providing 0 default width
- Draw_Fill: whether to fill the inside of the circle

Draw character (ASCII)

Set(Xstart, Ystart) as left top point, draws ASCII character

Parameter:

- Xstart: X coordinate of the left top point of character;
- Ystart: Y coordinate of the left top point 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 point, draws string.

Parameter:

- Xstart: X coordinate of left top point of characters;
- Ystart: Y coordinate of the top point 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

The function is using to draw Chinese strings using GB2312 encode

Parameters:

- Xstart: Coordinate of left top point of characters;
- Ystart: Coordinate of the top point of characters;
- pString : Pointer of string;
- Font: GB2312 encode :
  - font12CN : 11*12(字宽*字高) : 16*21 (Chinese)
  - font24CN : 24*24(字宽*字高) : 32*41 (Chinese)
- Color_Foreground: color of string
- Color_Background: color of the background

Draw number

Draw a set by of numbers, (Xstart, Ystart) is the left top point

Parameter:

- Xstart: X coordinate of the top point
- Ystart: Y coordinate of left top pixel
- Number: the numbers displayed the numbers are saved in int format, the maximum is 2147483647
- font: 5 strs 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 Image

Send image data of BMP file to buffer

Parameters:

- Image_buffer: address of image store 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

[Waveshare 2.66inch e-Paper Module Manual]

Comprehensive user guide for the Waveshare 2.66-inch e-Paper module, detailing its features, specifications, SPI timing, working protocol, precautions, and integration guides for Raspberry Pi, Jetson Nano, STM32, and Arduino platforms, including API descriptions and FAQs.

Raspberry Pi&Jetson Nano\c\BIN\SIGDIGL_EPAPER.c(1.0)

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

Parameters:

- Path: The path of the program
- Xstart: X coordination of left-top of picture, default 0
- Ystart: Y coordination of left-top of picture, default 0

Testing Code

The first time chapters introduce the classic linux three-layer code structure, here is a little explanation of the user test code file Raspberry Pi and Jetson Nano, in the directory: /pi/e-Paper/RPi_and_JetsonNano/, there are many routines, we only explain the routines we usually use in the directory:

/pi/e-Paper/RPi_and_JetsonNano/c/examples/EPD_xxx_test.c

You need to run the corresponding routine in the routine directory, for example, if you want to run the 7.5inch e-paper test routine, you need to enter the following command:

```
sudo ./epd_7in5_test.c
```

change to

```
sudo ./epd_7in5_test.bin
```

Re-execute in linux command mode as follows:

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

Python

For Jetson Nano (Raspberry Pi, Jetson Nano python2.7 and python3 python is easy to use than c code

Raspberry Pi and Jetson Nano - /RaspberryPi&Jetson Nano/python/lib,

Bottom Interface

The application layer (in the underlying interface

- Initialize relevant ports in the test sample

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

Note:

1. The functions are used to set GPIO before and after driving e-Paper
2. If the board.sys have is printed with Bcm2.1, the module error (such as the mode after Module_Init(), (as we test, the current loadout 0 in this mode);

- GPIO file of the

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

- SPI Write data:

```
def spi_writebyte(data)
```

Driver Interface

Regarding (the external ink, it is a 2.13inch e-paper, it is the2.13 (it, and so on)

- Initialize 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 1in_Full_update or 1in_Partial_update
#then 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 stream

```
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 re fresh, they should first use display(WhiteImage) to display pure-white background, and then use this function to display.
def display(self, image):
def displayPartBaseImage(self, image):
def displayPartial(self, image):
```

- Enter sleep mode

```
def sleep(self):
```

Testing Routine

For the Jetson (Regarding the external ink, it is a 2.13inch e-paper, it is the 2.13inch e-paper, it is the following directory:

/pi/e-Paper/RPi_and_JetsonNano/python/examples/

Regarding Pi and Jetson Nano: /pi/e-Paper/RPi_and_JetsonNano/python/lib,

If your python version is python2, and you need to run the 7.5inch e-paper test program, re-execute it in the linux command mode as follows:

```
sudo python epd_7in5_test.py
```

If your pigpio version is python3, and you need to run the 7.5inch e-paper test program, re-execute it in the linux command mode as follows:

```
sudo python3 epd_7in5_test.py
```

Orientation

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

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

GUI

Python have powerful PIL library's, which can be used directly to draw figures. Here we use it for drawing.

- Install the library image

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

Import the library

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

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

- Get image buffer for drawing.

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

The first parameter is the color of mode 1 (where 2 parameters. The second parameter is a type of image size. The third parameter is the color of the image, 0 is black and 255 is white.

- Create an image object

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

- Draw rectangle

```
draw.rectangle((5, 10, 210, 98), fill = 0)
```

The first parameter is a type of coordination. 0, 10 is the top left point of the rectangle, 200, 240 is the right-bottom point. fills = 0 set the fill color to black.

- Draw line

```
draw.line((14, 50, 50, 50), fill = 0)
```

The first parameter is a type of coordinates, 20, 60 is the beginning point, and 200, 24 is the end point. fill = 0 is the color of the line is black.

- Draw circle

```
draw.arc((150, 15, 190, 430), 0, 360, fill = 0)
```

The function is used to draw a circle of a square. The first parameter is a type of coordination of the square. The degree of the circle is 0 to 360 °, fill = 0 is the circle is black.

If the figure is a square, according to the circle, you will get an ellipse.

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

```
draw.chord((80, 130, 250, 250), 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 directly import ImageFont.ttc 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 characters, you can directly use the fonts; for Chinese characters, you need to add a symbol:

```
draw.text((0, 11), 'hello world', font = font, fill = 255)
draw.text((0, 76), u'微雪电子', font = font, fill = 0)
```

The first parameter is a type of 2 elements, with (0, 11) as the initial line, (0) is the line color, and the third parameter /u+255 as a unique text, it will not be displayed. The second parameter should 微雪电子

- Read bmp picture

```
image = Image.open(os.path.join(picdir, 'black.bmp'))
```

The parameter is the path of picture.

- Other functions

For more information about the PIL library, you can search on the.

User Guides of STM32

Hardware connection

The driver routine we provide uses based on STM32F103ZET6, the connection pins in the library on STM32F103ZET6. If you want to use other chip, you need to port the routine and change the connection according to actual situation.

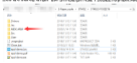
connect to STM32F103ZET

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

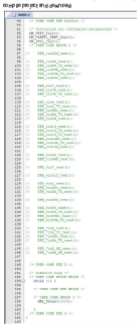
If the STM32 board you have is STM32F103RB, you can refer to the guide of it (Pepper Series)

Software 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 remove all the unnecessary annotations



Supporting types

1.02inch (128 × 80) ：

EPD_1IN02_test()：Demo for the 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 stopped 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 that can be bought(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). It is currently out of production.

EPD_1IN54b_V2_test()：Example for 1.54inch e-Paper B V2 (Black/White/Red). The purchase date later than November 2019 is the V2 version, the current shipping version has the V2 logo on the back.

EPD_1IN54c_test()：Example for 1.54inch e-paper C (Black/White/Red) ;

1.64inch (168 × 168)

EPD_1IN64g_test()：Example for 1.64inch e-paper G (Black/White/Red/Yellow)

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) ;

EPD_2IN7b_V2_test()：Example for 2.7inch e-paper B V2 (Black/White/Red) ;

2.9inch (296 × 128)

EPD_2IN9_test()：Example for 2.9inch e-paper (Black/White) ; pay version shipping price.

EPD_2IN9_V2_test()：Example for 2.9inch e-paper V2 (Black/White) ; The purchase date is later than December 2019 is the V2 version, the current shipping version has the V2 logo on the back.

EPD_2IN9bc_test()：Example for 2.9inch e-paper B (Black/White/Red) and 2.9inch e-paper C (Black/White/Yellow) ;

EPD_2IN9d_V3_test()：Example for 2.9inch e-paper D V3 (Black/White/Red) ; Purchase date later than July 2020 is the V2 version, the current shipping version has the V3 logo on the back.

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 stopped production and it can be bought before 019-05-10)

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_2IN13_V3_test()：Example for 2.13inch e-paper V3 (Black/White) that is the current version with sticker V3 on the package (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_2IN13b_V3_test()：Example for 2.13inch e-paper B V3 (Black/White/Red) ; Purchase date later than April 2020 is the V3 version, the current shipping version has the V3 logo on the back.

EPD_2IN13d_test()：Example for 2.13inch e-paper D (Black/White) ;

2.66inch (152 × 296)

EPD_2in66_test()：Example for 2.66inch e-Paper (Black/White) ;

EPD_2IN66b_test()：Example for 2.66inch e-paper B (Black/White/Red)

3inch (400 × 168)

EPD_3IN0g_test()：Example for 3inch e-paper G(Black/White/Red/Yellow)

3.52inch (360 × 240)

EPD_3IN52_test()：Example for 3.52inch e-paper (Black/White)

3.7inch (280 × 480)

EPD_3in7_test()：Example for 3.7inch e-Paper(Black/White);

4.01inch (640 × 400)

EPD_4IN01_test()：Example for the 4.01inch e-Paper HAT (F) (Black/White/Green/Blue/Red/Yellow/Orange);

4.2inch (400 × 300)

EPD_4IN2_test()：Example for 4.2inch e-paper (Black/White) ;

EPD_4IN2B_test()：Example for 4.2inch e-paper B (Black/White/Red) ;

EPD_4IN2b_V2_test()：Example for 4.2inch e-paper B V2 (Black/White/Red) ; The purchase date is later than September 2020, is the V2 version, and the current shipping version has the V2 logo on the back.

5.83inch (600 × 448)

EPD_5IN83_test()：Example for 5.83inch e-paper (Black/White) ;

5.83inch (600 × 448) ：

EPD_5IN83_test()：Example for 5.83inch e-paper (Black/White) , It is currently out of production.

EPD_5IN83_V2_test()：Example for 5.83inch e-paper V2 (Black/White) , The purchase date is later than December 2020 is the V2version, the current shipping version, with the V2 logo on the back.

EPD_5IN83b_test()：Example for 5.83inch e-paper B (Black/White/Red) and 5.83inch e-paper C (Black/White/Yellow) ;

EPD_5IN83b_V2_test()：Example for 5.83inch e-paper B V2 (Black/White/Red). The purchase date is later than August 2020 for the V2 version, the current shipping version, with the V2 logo on the back

7.3inch (V1 (640 × 384 , V2 (800 × 480) ：

EPD_7IN3_test()：Example for 7.3inch e-paper (Black/White) . The purchase date is earlier than 2019-12-07 is the V1 version, which is currently discontinued

EPD_7IN3bc_test()：For example for 7.3inch e-paper B (Black/White/Red) and 7.3inch e-paper C (Black/White/Yellow) , The purchase date later than 2019-12-07 is the V2 version, and the current shipping version has the V2 logo on the back;

EPD_7IN3_V2_test()：Example for 7.3inch e-paper V2 (Black/White) : The purchase date of 7.3inch e-paper B is earlier than 2019-12-03, which is the V1 version, which is currently discontinued and 7.3inch e-paper C is normally available.

EPD_7IN3bc_V2_test()：Example for 7.3inch e-paper B V2 (Black/White/Red) ; The purchase date is later than 2019-12-07 is the V2 version, with the V2 logo on the back; the purchase date is later than March 2021 is the V3 version, the current 7.3inch e-paper B shipping version, with the V3 logo on the back, the same use this program;

7.5inch (800 × 480)

epd_7in5_test.py：Example for 7inch(e-paper G (Black/White/Red/Yellow).

7.5inch (880 × 528)

EPD_7IN5_HD_test()：Example for 7.5inch e-paper HD (Black/White);

EPD_7IN5b_HD_test()：Example for 7.5inch e-paper B HD (Black/White/Red)

Note: The above open 2 are recommended, and please refer to the screen type for the specific version

For example, 1.54inch e-paper Module, Open the epd1in54 file and find the epd1in54.py file.

Open the program, select the corresponding pairs and start debugging

Codes description

Write in hardware interface

We package the bottom for different hardware platforms.

You can check the DEV_Config.c/.h file which is saved in \STM32\STM32-F103ZET6\User\Config

- DWR type:

```
#define DEV_RST_  port16_t
#define DEV_DC_   port17_t
#define DEV_BUSY_ port18_t
```

- Module initialization and exit:

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

Note: 1.The functions are used to set GPIO 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_Init(). Be careful, 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:

```
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 declared here

- Initialization: Initialize the communication method of the e-Paper, and wake up the 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 B, 2.9inch e-Paper D
void EPD_xxx_Init(UBYTE Mode); //Mode: 0 represents full initialization; Mode: 1 represents partial refresh
//Other type
void EPD_xxx_Init(void);
```

xxx is the type of e-paper

- Clear display: The 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 have ghosting when refreshing the screen, so you need to use EPD_xxx_DisplayPartBaseImage to display static background and then use EPD_xxx_DisplayPart() to dynamic display when partial refreshing.

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

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

- Draw and sleep mode

```
void EPD_Sleep(void);
```

Note: You must add the delay() function to your program after calling the sleep function, otherwise the display may not work properly.

Application function

Basic drawing functions are provided here. You can find them in GUI_Paint.h and GUI_Paint.c.

```
Paint_NewImage()
Paint_SelectImage()
Paint_SetRotate()
Paint_SetMirroring()
Paint_SetPixel()
Paint_Clear()
Paint_ClearWindows()
Paint_DrawPoint()
Paint_DrawLine()
Paint_DrawRectangle()
Paint_DrawCircle()
Paint_DrawChar()
Paint_DrawString_EN()
Paint_DrawString_CN()
Paint_DrawNum()
Paint_DrawTime()
Paint_DrawBitMap()
```

The API is as follows (in GUI_Paint.h):

```
UWORD Paint_NewImage(UBYTE *image, UWORD Width, UWORD Height, UWORD Rotate, UWORD Color);
Paint_SelectImage(UBYTE *image);
Paint_SetRotate(UWORD Rotate);
Paint_SetMirroring(UBYTE mirror);
Paint_SetPixel(UWORD Xpoint, UWORD Ypoint, UWORD Color);
Paint_Clear(UWORD Color);
Paint_ClearWindows(UWORD Xstart, UWORD Ystart, UWORD Xend, UWORD Yend, UWORD Color);
Paint_DrawPoint(UWORD Xpoint, UWORD Ypoint, UWORD Color, DOT_PIXEL Dot_Pixel, DOT_STYLE Dot_Style);
Paint_DrawLine(UWORD Xstart, UWORD Ystart, UWORD Xend, UWORD Yend, UWORD Color, LINE_STYLE Line_Style, LINE_WIDTH Line_Width);
Paint_DrawRectangle(UWORD Xstart, UWORD Ystart, UWORD Xend, UWORD Yend, UWORD Color, DRAW_FILL Filled, DOT_PIXEL Line_Width);
Paint_DrawCircle(UWORD X_Center, UWORD Y_Center, UWORD Radius, UWORD Color, DRAW_FILL Draw_Fill, DOT_PIXEL Line_Width);
Paint_DrawChar(UWORD Xstart, UWORD Ystart, const char Ascii_Char, sFONT* Font, UWORD Color_Foreground, UWORD Color_Background);
Paint_DrawString_EN(UWORD Xstart, UWORD Ystart, const char * pString, sFONT* Font, UWORD Color_Foreground, UWORD Color_Background);
Paint_DrawString_CN(UWORD Xstart, UWORD Ystart, const char * pString, cFONT* font, UWORD Color_Foreground, UWORD Color_Background);
Paint_DrawNum(UWORD Xpoint, UWORD Ypoint, int32_t Number, sFONT* Font, UWORD Color_Foreground, UWORD Color_Background);
Paint_DrawTime(UWORD Xstart, UWORD Ystart, PAINT_TIME *pTime, sFONT* Font, UWORD Color_Background, UWORD Color_Foreground);
Paint_DrawBitMap(const unsigned char* image_buffer);
```

- Create a new image buffer: This function is used to create a new image with width, height, image degree and fill 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 degrees
    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 usage.

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

- Set clip by 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(UWORD Rotate)
Parameter:
    Rotate: Rotate degrees, you can choose ROTATE_0, ROTATE_90, ROTATE_180, ROTATE_270 which mean to be 0, 90, 180, 270 degree respectively.
```

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

```
void Paint_SetMirroring(UBYTE mirror)
Parameter:
    mirror: You can set to be MIRROR_NONE, MIRROR_HORIZONTAL, MIRROR_VERTICAL, MIRROR_ORIGIN
```

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

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

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

```
void Paint_Clear(UWORD Color)
Parameter:
    Color:
```

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

```
void Paint_ClearWindows(UWORD Xstart, UWORD Ystart, UWORD Xend, UWORD Yend, UWORD Color)
Parameter:
    Xstart: Start coordinate of X-axis of the window
    Ystart: Start coordinate of Y-axis of the window
    Xend: End coordinate of X-axis of the window
    Yend: End coordinate of Y-axis of the window
    Color:
```

- Draw Point: Draw a point with specified position, color, size, and dot pattern.

```
void Paint_DrawPoint(UWORD Xpoint, UWORD Ypoint, UWORD 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 * 2
            DOT_PIXEL_3X3 ,     // 3 * 3
            DOT_PIXEL_4X4 ,     // 4 * 4
            DOT_PIXEL_5X5 ,     // 5 * 5
            DOT_PIXEL_6X6 ,     // 6 * 6
            DOT_PIXEL_7X7 ,     // 7 * 7
            DOT_PIXEL_8X8 ,     // 8 * 8
        } DOT_PIXEL;
    Dot_Style: style of point.
        typedef enum {
            DOT_FILL_AROUND = 1,
            DOT_STYLE_DFT,
        } DOT_STYLE;
```

- Draw line: draw a line between (Xstart, Ystart) and (Xend, Yend).

```
void Paint_DrawLine(UWORD Xstart, UWORD Ystart, UWORD Xend, UWORD Yend, UWORD Color, LINE_STYLE Line_Style = LINE_STYLE_SOLID, DOT_PIXEL Line_Width = DOT_PIXEL_1X1)
Parameter:
    Xstart: Start coordinate of X-axis of line
    Ystart: Start coordinate of Y-axis of line
    Xend: End coordinate of X-axis of line
    Yend: End coordinate of Y-axis 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 * 2
            DOT_PIXEL_3X3 ,     // 3 * 3
            DOT_PIXEL_4X4 ,     // 4 * 4
            DOT_PIXEL_5X5 ,     // 5 * 5
            DOT_PIXEL_6X6 ,     // 6 * 6
            DOT_PIXEL_7X7 ,     // 7 * 7
            DOT_PIXEL_8X8 ,     // 8 * 8
        } DOT_PIXEL;
    Line_Style: Solid or Dotted of the lines
        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, DRAW_FILL Filled, DOT_PIXEL Line_width = DRAW_FILL_EMPTY)
Parameter:
    Xstart: Start coordinate of X-axis of rectangle
    Ystart: Start coordinate of Y-axis of rectangle
    Xend: End coordinate of X-axis of rectangle
    Yend: End coordinate of Y-axis of rectangle
    Color: color of rectangle
    Line_width: the width of edges, 8 sizes are available:
        typedef enum {
            DOT_PIXEL_1X1 = 1,  // 1 x 1
            DOT_PIXEL_2X2 ,     // 2 * 2
            DOT_PIXEL_3X3 ,     // 3 * 3
            DOT_PIXEL_4X4 ,     // 4 * 4
            DOT_PIXEL_5X5 ,     // 5 * 5
            DOT_PIXEL_6X6 ,     // 6 * 6
            DOT_PIXEL_7X7 ,     // 7 * 7
            DOT_PIXEL_8X8 ,     // 8 * 8
        } DOT_PIXEL;
    Draw_Fill: fill 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 * 2
            DOT_PIXEL_3X3 ,     // 3 * 3
            DOT_PIXEL_4X4 ,     // 4 * 4
            DOT_PIXEL_5X5 ,     // 5 * 5
            DOT_PIXEL_6X6 ,     // 6 * 6
            DOT_PIXEL_7X7 ,     // 7 * 7
            DOT_PIXEL_8X8 ,     // 8 * 8
        } DOT_PIXEL;
    Draw_Fill: style of circle
        typedef enum {
            DRAW_FILL_EMPTY = 0,
            DRAW_FILL_FULL,
        } DRAW_FILL;
```

- Draw character (ASCII): Specifies where to put the 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: Font of character, such as:
        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: Distance of string
    Font: Font of string, such as:
        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 character: the function is used to draw Chinese strings (using GB2312 encoding)

```
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 point of characters
    Ystart: Coordinate of left-top point of characters
    pString: Distance of string
    Font: Font of string, such as:
        Font12CN: 11*12(Incline: 10*12) (Chinese)
        Font24CN: 23*24(Incline: 22*24) (Chinese)
    Color_Foreground: color of string
    Color_Background: color of background
```

- Draw Number: Draw a number on the specified position, with the specified number of digits and font size.

```
void Paint_DrawNum(UWORD Xpoint, UWORD Ypoint, int32_t Number, sFONT* Font, UWORD Color_Foreground, UWORD Color_Background)
Parameter:
    Xstart: X coordinate of left-top point
    Ystart: Y coordinate of left-top point
    Number: The number displayed, the numbers are saved in bit forms, the maximum is 2147483647;
    Font: Font of string, such as:
        Font8: 5*8
        Font12: 7*12
        Font16: 11*16
        Font20: 14*20
        Font24: 17*24
    Color_Foreground: color of digit
    Color_Background: color of background
```

- Display time: Display time, (Xstart, Ystart) is the left-top point. The 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 point of character
    Ystart: Y coordinate of left-top point of character
    pTime: Format of time displayed
    Font: Font of string, such as:
        Font8: 5*8
        Font12: 7*12
        Font16: 11*16
        Font20: 14*20
        Font24: 17*24
    Color_Foreground: color of digit
    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
```

User Guides of Arduino

Hardware connection

The driver board provides a corresponding routine for the Arduino UNO. If you want to use another Arduino board, you only need to change the connection according to the pin definition of your board.

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 also provide you a tool to convert pictures into .h files. You can directly call it in the Arduino UNO. The black-color e-Paper is the 0x00/0xFF with one e-Paper byte is stored into it.

Arduino corresponding pins

e-Paper	Arduino UNO	Mega2560
VCC	5V	5V
GND	GND	GND
DIN	D11	D51
CLK	D13	D52
CS	D10	D10
DC	D9	D9
RST	D8	D8
BUSY	D7	D7

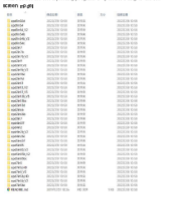
Install Arduino IDE

Arduino IDE Install Guide

Run the Program

Download the program on the product encyclopedia interface, and then unzip it. Arduino programs are located at ~\Arduino\...

Please select the corresponding program to open according to the ink screen you got



Supporting types

1.02inch (128 × 80) ：

EPD_1in02_test()：Example for 1.02inch e-Paper(), 0.2inch e-Paper

1.54inch (1.54inch e-paper v：122 × 152 , others：200 × 200) ：

epd1in54：Example for 1.54inch e-paper V1 （Black/White）; The version is stopped production, which can be bought before 2019-11-22 ;

epd1in54_V2: Example for 1.54inch e-paper V2 （Black/White）; This is the current version (can be bought since (2020-07-29). The e-Paper has V2 sticker on the backside.

epd1in54b：Example for 1.54inch e-paper B （Black/White/Red）; It is currently out of production.

epd1in54b_V2：Example for 1.54inch e-Paper B V2 (Black/White/Red). The purchase date later than November 2019 is the V2 version, the current shipping version has the V2 logo on the back;

epd1in54c：Example for 1.54inch e-paper C （Black/White/Red）;

1.64inch (128 × 160)

EPD_1in64g_test()：Example for 1.64inch e-paper G

（Black/White/Red/Yellow）

2.7inch (264 × 176)

epd2in7：Example for 2.7inch e-paper （Black/White）;

epd2in7b：Example for 2.7inch e-paper B （Black/White/Red）;

epd2in7b_V2：Example for 2.7inch e-paper B V2 （Black/White/Red）;

2.9inch (296 × 128)

epd2in9：Example for 2.9inch e-paper (Black/White); this version is stopped production ;

epd2in9_V2：Example for 2.9inch e-paper V2 （Black/White）; The purchase date is later than December 2019 is the V2 version, the current shipping version has the V2 logo on the back;

epd2in9bc：Example for 2.9inche-paper B （Black/White/Red） and 2.9inch e-paper C （Black/White/Yellow）;

epd2in9d：Example for 2.9inch e-paper D V3 （Black/White/Red）; Purchase date later than July 2020 is the V3 version, the current shipping version can see V3 logo on the back;

epd2in9d_V2：Example for 2.9inch e-paper D （Black/White）;

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

epd2in13：Example for 2.13inch e-paper V1 （Black/White）, this version is stopped production and it can be bought before 2019-09-19;

epd2in13_V2：Example for 2.13inche-paper V2 （Black/White）; This is the current version with sticker V2 on the backside (2020-07-29) ;

epd2in13_V3：Example for 2.13inche-paper V3 （Black/White）; This is the current version with sticker V3 on the backside (2020-07-29) ;

epd2in13b：Example for 2.13inche-paper B （Black/White/Red） and 2.13inch e-paper C （Black/White/Yellow）;

epd2in13b_V3：Example for 2.13inch e-paper B V3 （Black/White/Red）; Purchase date later than April 2020 is the V3 version, the current shipping version has the V3 logo on the back;

epd2in13d：Example for 2.13inch e-paper D （Black/White）;

2.66inch (152 × 296)

epd2in66：Example for 2.66inch e-Paper (Black/White);

epd2in66b：Example for 2.66inch e-Paper B (Black/White/Red);

3inch (400 × 168)

EPD_3in0g_test()：Example for 3inch e-paper G(Black/White/Red/Yellow)

3.52inch (360 × 240)

EPD_3in52_test()：Example for 3.52inch e-paper (Black/White)

3.7inch (280 × 480)

epd3in7：Example for 3.7inch e-paper (Black/White);

4.01inch (640 × 400)

epd4in01f：Example for the 4.01inch e-paper (4C) (F)

(Black/White/Green/Blue/Red/Yellow/Orange);

4.2inch (400 × 300)

epd4in2：Example for 4.2inch e-paper （Black/White）;

epd4in2b：Example for 4.2inch e-paper B （Black/White/Red）;

epd4in2b_V2：Example for 4.2inch e-paper B V2 （Black/White/Red）; The purchase date is later than November 2020, it is the V2 version, and the current shipping version has the V2 logo on the back;

5.65inch (600 × 448)

epd5in65f：Example for for 5.65inch e-Paper F

(Black/White/Green/Blue/Red/Yellow/Orange);

5.83inch (600 × 448) ：

epd5in83：Example for 5.83inch e-paper （Black/White）, is currently out of production;

epd5in83_V2：Example for 5.83inch e-paper V2 （Black/White）, The purchase date is later than December 2020 is the V2 version, the current shipping version, with the V2 logo on the back;

epd5in83bc：Example for 5.83inch e-paper B （Black/White/Red/Yellow） and 5.83inch e-paper C （Black/White/Red/Yellow）;

epd5in83b_V2：Example for 5.83inch e-paper B V2 （Black/White/Red/Yellow）

The purchase date is later than August 2020 is the V2 version, the current shipping version, with the V2 logo on the back

7.3inch (V1 (640 × 384 , V2 (800 × 480) ：

epd7in3：Example for 7.3inch e-paper （Black/White/Green/Blue/Red/Yellow/Orange）, The purchase date is ear ier than 2019-12-07 is the V1 version, which is currently discontinued

epd7in3_V2：For example for 7.3inch e-paper B （Black/White/Red/Yellow） and 7.3inch e-paper C （Black/White/Red/Yellow）, The purchase date later than 2019-12-07 is the V2 version, and the current shipping version has the V2 logo on the back;

epd7in3bc：Example for 7.3inch e-paper V2 （Black/White/Red/Yellow）, The purchase date of 7.3inch e-paper B is earlier than 2019-12-07, which is the V1 version, which is currently discontinued, and 7.3inch e-paper C is not easy available.

epd7in3b_V2：Example for 7.3inch e-paper B V2 （Black/White/Red/Yellow）; The purchase date is later than 2019-12-07 is the V2 version, with the V2 logo on the back, the purchase date is later than March 2021 is the V3 version, the current 7.3inch e-paper B shipping version, with the V3 logo on the back, the same use this program

7.5inch (800 × 168)

EPD_7in5g_test()：Example for 7.5inch e-paper G

（Black/White/Red/Yellow）

7.5inch (880 × 528)

epd7in5_HD：Example for 7.5inch e-paper HD （Black/White）;

epd7in5b_HD：Example for 7.5inch e-paper B HD （Black/White/Red）;

Notice: The above open is for reference only, please refer to the screen you for the specific version.

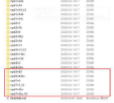
For example, 1.54inch e-paper display: Open the epd1in54.ino file in the Arduino IDE

Open the program, connect the development board to the computer, and click UNO

Select the corresponding COM port

Then click compile and download

Since the Flash of Arduino UNO is very small, several large-size screens use MEGA2560 for better results <



Code Description

Files

Take the 1.54 inch ink screen controlled by Arduino UNO as an example, open the epd1in54 directory:

It should:

Epd1in54.cpp: Open with Arduino IDE

Epd1in54.h: Header file of the driver for the screen;

Epdif.cpp: It is the hardware interface, which encapsulates the read and write of the GPIO, SPI, and some basic operations, and can be modified;

Fonts.h, Font12.cpp, Font16.cpp, Font20.cpp, Font24.cpp, these .h are the

epd.init(lut_full_update); //Fully update
epd.init(lut_partial_update); //Quick refresh
```

- 4.2inch e-Paper

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

Clear, clear the e-Paper to white

```
epd.Clear();
```

If there are pictures, you need to send the picture data to the SRAM of the e-Paper first, and then refresh to display the picture.

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

Transmit one frame of image and display

```
void DisplayFrame(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 entry 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：

The API functions are defined in epdpaint.h file:

```
class Paint {
public:
    Paint(unsigned char* image, int width, int height);
    ~Paint();
    int  GetRotate(void);
    void SetRotate(int rotate);
    unsigned char* GetImage(void);
    int  GetWidth(void);
    void SetWidth(int width);
    int  GetHeight(void);
    void SetHeight(int height);
    void Clear(int colored);
    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;
};
```

Initialize image buffer

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

The first parameter is the 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, and rotation degree.

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

Get the image data

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

Draw circle

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

Draw a circle with center coordinate (x,y) and radius.

Draw character

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

Set (x,y) as the start point, draw a character and its font and color.

Drawstring

```
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 a font, color is colored

Draw line

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

Like (x0,y0) as start point, (x1,y1) as end point

Draw cross line

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

Set (x,y) as start point, draws line, the width is width, and the color is colored

Draw a vertical line

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

Like (x,y), as a start point, draws vertical line, width is height, and the color is colored.

Draw a empty rectangle

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

Like (x0,y0) as a start point, (x1,y1) as an end point, draw a rectangle, color is hollow and colored.

Draw a full rectangle

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

Like (x0,y0) as a start point, (x1,y1) asan end point, draw a rectangle, fill it with color: colored

Draw an empty circle

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

Like (x,y) as the center, and draw an empty circle with a radius, color is colored

Draw a full circle

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

Like (x,y) as the center, and draw a solid circle with a radius, filled with color: colored.

Resources

Documentation

- [Waveshare Wiki's page for this e-Paper]
- [Detailed guide and API for e-Paper]
- [2.9inch e-Paper Specification]
- [2.9inch e-Paper Demo Description]

Demo code

- [GitHub]

Software

- [Image2Lcd]
- [Character Matrix]

FAQ

Question about Software

Question: How to download the program to the MCU, the MDK should be set to support the simulation mode?

Answer:
You should set the debug to "Use Simulator" with the MDK-ARM, otherwise the program cannot be downloaded to the MCU.

Question: Error: No such file or directory

Answer:
Please add the missing library using the command such as project folder: pio lib install "adafruit/Adafruit GFX Library"

Question: When to download the program, there is an error as "No algorithm found for: 20000000h-20000FFFh", how to solve it?

Answer:
Open the project, click "Options for Target", select the "Debug" option, choose the debugger you are using, click "Settings", check the "Reset and Run" option in the pop-up window.

Question: Can't display Chinese on the e-ink screen?

Answer:
The Chinese character library of our demos uses GB2312 encoding method, please change your .exe .jlink .flm to GB2312 encoding format, recompile and download it, and it will display normally.

Question: The e-Paper display is too dark or too light?

Answer:
You can adjust the value of vcom in the program to change the display contrast, and the screen approximate effect with both operating in perpendicularly opposite.

Question: How to set the background color?

Answer:
The background color can be changed by the parameter of the function Paint_Clear() or the VCOM_AND_DATA_INTERVAL_SETTING register.

Question: After using the program for a long time, the display is abnormal and the screen cannot
```

Answer:
In this case, the customer needs to reduce the position of the round brush and lower the screen after it bounces if bouncing (increasing the voltage of VCOM can improve the error, but it will increase the afterimage).

Question: When the e-ink screen is in deep sleep mode, the last time the screen refreshes will be retained. Should I clear it?

Answer:
The process of re-displaying the e-ink screen is equal to the process of re-powering on the power. Therefore, when the MPU wakes up, it must first clear the screen, so as to avoid the afterimage phenomenon to the greatest extent.

Question: When testing the program, the program has been stuck in e-Paper busy?

Answer:
*Busy is caused by the unsuccessful operation:
1. Check whether the wiring is correct.
2. Check whether the MPU is running and whether the parameters are configured correctly (SPI configuration, MPU configuration parameters).

Question about Hardware

Question: Does it support 3.3V GPIO port operation?

Answer:
Yes, it has come with a level conversion circuit internally, supporting 3.3V GPIO.

Question: What's the difference between the 4-wire serial port and the 3-wire serial port?

Answer:
The default communication of the screen is 3.3~5.0V GPIO and 4-wire SPI. Level conversion is required. In addition, the voltage should not be lower than 2.5V, so as not to affect the display effect of the screen.
The 3-wire SPI can save one GPIO port than the 4-wire SPI, which will reduce one GPIO port, but the refresh rate will be slower, so you can choose according to your own needs.

Question: Do I use analog SPI?

Answer:
Yes, pay attention to the correct timing.

Question: Why is the BUSY pin always busy?

Answer:
Check if SPI communication is normal.
Confirm whether the BUSY pin is normally initialized to input mode.
It may be that there is no normal read, try to shorten the duration of the low level during reset (because the power-off switch is added to the drive circuit, so there is a pin to control the switch, which will cause the power to be turned off when the MCU is in sleep mode, so the power supply will be cut off. Generally, you can try to extend the reset time).

Question about Screen

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

Answer:
* [Working conditions] : Temperature range: 0~50℃; Humidity range: 35%~65%RH
  - [Storage conditions] : Temperature range: below 30℃; Humidity range: below 60%RH; Minimum storage time: 6 months
  - [Transportation conditions] : Temperature range: -25~70℃; Maximum transportation time: 10 days
  - [After unpacking] : Temperature range: 20℃±5℃; Humidity range: 30±5%RH; Maximum storage time: Assemble within 72 hours.

Question: Precautions for e-ink screen refresh

Answer:
Refresh mode
- Full refresh: The electronic ink screen will flicker several times during the refresh process (the number of flickers depends on the refresh time), and the latter is different from the latter page to ensure the best display effect.
- Partial refresh: The electronic ink screen has no flickering effect during the refresh process. Users should be aware that partial refresh may cause ghosting after multiple consecutive uses. It is strongly recommended to perform a full refresh after multiple partial refreshes. Applications that require partial refresh will support partial refresh and full refresh, or only support partial refresh (currently only 4-level black and white e-ink screens support partial brushing, please refer to product page description).
- e-Book mode
- During use, it is recommended that users choose products that support the low-flash function).
- During the start-up process (that is, after the refresh operation), it is recommended that the customer first power the e-ink screen to keep white, or power on the system (turn power supply pins of the e-ink screen on or disconnect it with an analog switch) to reduce power consumption and prolong the life of the e-ink screen. (If some e-ink screens are powered on for a long time, the screen will be damaged (burn/ghosting).)
- During the use of the three-color e-ink screen, it is recommended the customers update the display screen at least once every 24 hours (if the screen remains the same screen for a long time, the screen burn will be difficult to repair).
- application
  - The e-ink screen is not suitable for outdoor use. It is also not suitable for environments with strong light, as the screen may turn white, and in the worst case, lose its protection measures, because charged particles will be activated by strong light for a long time, resulting in loss of activity and failure to refresh. Transmission is irreversible. When designing e-ink screen products, customers should pay attention to and minimize the exposure of the screen to ultraviolet rays.

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

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

Question: After using for a period of time, the screen refresh (full refresh) has errors & temporary display (scratches or residue)?

Answer:
Power on the device/system after a long time, after each refresh operation, it is recommended to let the screen sleep (skip if already power optimized), especially the screen with pins such as the screen is in a high voltage state for a long time.

Question: After the e-ink screen enters deep sleep mode, can it be normally used?

Answer:
Yes, but you need to re-initialize the electronic paper with software.

Question: Why is the image displayed incomplete?

Answer:
Maybe the SPI rate is too high, resulting in data loss, try to reduce the SPI rate.
Poor traces or unstable power supply result in data loss.
The data cable is too long to cause data loss, the extension cable should not exceed 20cm.

Question: What is the waveform file of the e-ink screen and what does it do?

Answer:
The digital gray scale of electrophoretic electronic paper is determined by the specific position of the particles in the combination of voltage. The electrophoretic permeation occurs between black particles and white particles under the action of voltage. The voltage required for the process of electrophoretic movement of the particles is the driving force of the electronic paper display, and the optimization of the driving waveform will directly affect the display effect of the display. The driving waveform file is used to drive the parameters for each of the voltage sequences that promotes the electrophoretic movement of the particles, and is ready to be read regularly when the electronic paper is refreshed.
Different product e-ink paper (electrophoretic ink) or environmental variations require different voltage values when driving the display (such as materials, manufacturing processes, etc. The waveform) file e-ink screen is matched to the relationship between materials, voltages and temperatures. Generally speaking, after each batch of electrophoretic inks is generated, there will be a corresponding waveform file in the form of .wbf file. The film manufacturer will provide the waveform file and electrophoretic matrix to the manufacturer of the electronic paper screen, and then the film manufacturer of electronic paper screen integrates the protection board, substrate and driver and then provides it to customers. If the waveform file does not correspond to the screen, it is likely that the display cannot be displayed or the display effect is unsatisfactory.
Generally, the waveform file has OTP built into the driver IC of the ink screen when leaving the factory, and some programs will provide a host external waveform files to drive the e-ink screen.

Question: What do LUT and OTP stand for?

Answer:
LUT is the abbreviation of LOOK UP TABLE, and OTP is the abbreviation of ONE TIME PROGRAM. The simple description of LUT is the said waveform file, and the waveform files are divided into OTP and REGISTER. Saving (only OTP if not used) waveform to register writing and REGISTER if no external waveform register writing.

Question: What is the problem with using e-paper?

Answer:
There are mainly two types of ink screens.
One is to drive the electronic paper film
The other is to electrically refresh the glass and new glass

Question: How do I link to different programs as per their port?

Answer:
See reference pinning in different programs enough to be operated in the programs, such as the pinning of the driver electronic paper and the electronic paper IC, and finally doing the Update/Turn On Display on Demo.

Question: Does the three-color e-Paper have a color/low-color difference?

Answer:
Yes, when e-Paper is batched, there will be some color differences, which is a normal phenomenon. Store the e-paper items up to reduce the reddish/yellowishness to a certain extent.

Others

Question: Are your screens shipping with a film?

Answer:
with film

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

Answer:
0.5mm/FPC 24Pin.

Question: What type of connector does the ink screen use?

Answer:
Drive socket 0.5-24pin rear flip type 2.0H (FPC connector).

Question: Does e-Paper have a built-in temperature sensor?

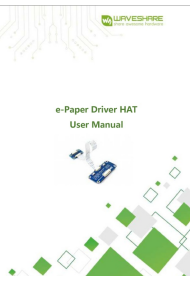
Answer:
Approved, all screens have built-in temperature sensors, and separate I2C/SPI temperature sensors can also be used with I2C/SPI.

Question: Why don't the pages of my drawing refresh and promptly?

Answer:
The full brush algorithm should be used when the ink screen is refreshed (the partial brush is not recommended).

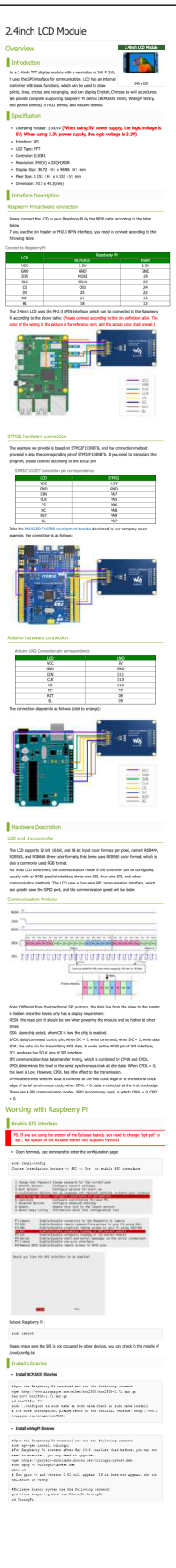
Question: What is the difference between using the program provided by the ink screen and writing the driver yourself?

Answer:
It may be a drive based on the BCM2835 library that has run the C language before. At the time, you need to import the library then run the python driver.



[Waveshare e-Paper Driver HAT User Manual: Connect SPI E-Paper Displays to Raspberry Pi, Arduino, STM32]

User manual for the Waveshare e-Paper Driver HAT, detailing its features, product parameters, interface specifications, and supported e-Paper models. Includes setup guides for Raspberry Pi, Arduino, and STM32 development boards.



```
./build
gpio -v
# If the gpio -v and version 2.81 will appear. If it does not appear, it says
# that there is an installation error
```

- **Install Python libraries**

```
#python2
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
#python3
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 Examples

Open Raspberry Pi terminal and run the following command

```
sudo apt-get install unzip -y
sudo wget https://www.waveshare.com/w/upload/8/8d/LCD_Module_RPI_code.zip
sudo unzip -o LCD_Module_RPI_code.zip
cd LCD_Module_RPI_code/RaspberryPi/
```

Run the demo codes

Please go into the RaspberryPi directory (demo codes) first and run the commands in terminal

C codes

- Re-compile the demo codes

```
cd c
sudo make clean
sudo make -j 8
```

- The test program of all screens can be called directly by entering the corresponding size

```
sudo ./main 1inch3 1inch
```

Depending on the LCD, one of the following commands should be entered:

```
#1.96inch LCD Module
sudo ./main 1.96
#1.47inch LCD Module
sudo ./main 1.47
#1.28inch LCD Module
sudo ./main 1.28
#1.14inch LCD Module
sudo ./main 1.14
#1.47inch LCD Module
sudo ./main 1.47
#1.69inch LCD Module
sudo ./main 1.69
#1.54inch LCD Module
sudo ./main 1.9
#1.9inch LCD Module
sudo ./main 1.9
#2.13inch LCD Module
sudo ./main 2.13
```

python

- Enter the python program directory and run the command to `ls`

```
cd python/examples
ls -l
```

Test programs for all screens can be viewed, sorted by size:

0in96_LCD_test.py: 0.96inch LCD test program
1in14_LCD_test.py: 1.14inch LCD test program
1in28_LCD_test.py: 1.28inch LCD test program
1in3_LCD_test.py: 1.3inch LCD test program
1in47_LCD_test.py: 1.47inch LCD test program
1in69_LCD_test.py: 1.69inch/LCD test program
1in9_LCD_test.py: 1.9inch LCD test program
2in13_LCD_test.py: 2inch LCD test program
2in9_LCD_test.py: 2inch/4 LCD test program

- Just run the program corresponding to the screen, the program supports python2/3

```
# python2
sudo python 1inch69_LCD_test.py
sudo python 1inch3_LCD_test.py
sudo python 1inch28_LCD_test.py
sudo python 1inch14_LCD_test.py
sudo python 1inch47_LCD_test.py
sudo python 1inch9_LCD_test.py
sudo python 1inch_LCD_test.py
sudo python 2inch_LCD_test.py
# python3
sudo python3 1inch96_LCD_test.py
sudo python3 1inch3_LCD_test.py
sudo python3 1inch28_LCD_test.py
sudo python3 1inch14_LCD_test.py
sudo python3 1inch47_LCD_test.py
sudo python3 1inch69_LCD_test.py
sudo python3 1inch9_LCD_test.py
sudo python3 2inch_LCD_test.py
sudo python3 2inch4_LCD_test.py
```

FBCP Porting

PS: FBCP is currently not compatible with 64-bit Raspberry Pi system, it is recommended to use 32-bit system.

FramebufferUse uses a video output device to drive a video display device from a memory buffer containing complete frame data. Simply put, a memory area is used to store the display content, and the display content can be changed by changing the data in the memory.

There is an open source project on github: flaviut/fbcp (fbcp-ili9341). Compared with other fbcp projects, this project uses partial refresh and DMA to achieve a speed of up to 60fps

Download Drivers

```
sudo apt-get install cmake -y
cd ~
wget https://www.waveshare.com/w/upload/1/1f/Waveshare_fbcp.zip
unzip Waveshare_fbcp.zip
cd Waveshare_fbcp/
sudo chmod +x -Rf install/
```

Method 1: Use a script (recommended)

Here are have written several scripts that allow users to quickly use fbcp and run corresponding commands according to their own screen.

If you use a script and do not need to modify it, you can ignore the second method below.

Note: The script will replace the corresponding /boot/config.txt and /etc/rc.local and restart, if the user needs, please back up the relevant files in advance.

```
#3.95inch LCD Module
sudo ./install/waveshare-1inch69
#1.46inch LCD Module
sudo ./install/waveshare-1inch14
#1.31inch LCD Module
sudo ./install/waveshare-1inch3
#1.99inch LCD Module
sudo ./install/waveshare-1inch99
#1.95inch LCD Module
sudo ./install/waveshare-1inch95
#1.69inch LCD Module
sudo ./install/waveshare-1inch69
#1inch LCD Module
sudo ./install/waveshare-1inch
#2.41inch LCD Module
sudo ./install/waveshare-2inch4
```

Method 2: Manual Configuration

Environment Configuration

Raspberry Pi's vc4-kms-v3d will cause fbcp to fail, so we need to close vc4-kms-v3d before installing it in fbcp.

```
sudo nano /boot/config.txt
```

Just block the statement corresponding to the picture below.

```
# Enable DRM VC4 V3D driver
#dtoverlay=vc4-kms-v3d
#max_framebuffers=2
```

A reboot is then required.

```
sudo reboot
```

Compile and run

```
cmake build
cd build/
cmake [options]
sudo make -j
sudo ./fbcp
```

Replace it by yourself according to the LCD Module you use, above cmake [options] ...

```
#3.95inch LCD Module
sudo cmake -DSPI_BUS_CLOCK_DIVISOR=1 -DGPIO_TFT_DATA_CONTROL=25 -DGPIO_TFT_RESET_PIN=27 -DSPI_BUS_CS1=ON -DSPI_BUS_CLOCK_DIVISOR_CS1=8 -DGPIO_TFT_CHIP_SELECT=8 -DSPI_BUS_CS2=ON -DSPI_BUS_CLOCK_DIVISOR_CS2=8 -DGPIO_TFT_CHIP_SELECT=7 -DUSE_SPI_DMA=ON -DSTATISTICS=0 ..
#1.46inch LCD Module
sudo cmake -DSPI_BUS_CLOCK_DIVISOR=2 -DGPIO_TFT_DATA_CONTROL=25 -DGPIO_TFT_RESET_PIN=27 -DSPI_BUS_CLOCK_DIVISOR=4 -DGPIO_TFT_CHIP_SELECT=8 -DUSE_SPI_DMA=ON -DSTATISTICS=0 ..
#1.31inch LCD Module
sudo cmake -DSPI_BUS_CLOCK_DIVISOR=2 -DGPIO_TFT_DATA_CONTROL=25 -DGPIO_TFT_RESET_PIN=27 -DSPI_BUS_CLOCK_DIVISOR=4 -DGPIO_TFT_CHIP_SELECT=8 -DUSE_SPI_DMA=ON -DSTATISTICS=0 ..
#1.99inch LCD Module
sudo cmake -DSPI_BUS_CLOCK_DIVISOR=2 -DGPIO_TFT_DATA_CONTROL=25 -DGPIO_TFT_RESET_PIN=27 -DSPI_BUS_CLOCK_DIVISOR=4 -DGPIO_TFT_CHIP_SELECT=8 -DUSE_SPI_DMA=ON -DSTATISTICS=0 ..
#1.95inch LCD Module
sudo cmake -DSPI_BUS_CLOCK_DIVISOR=2 -DGPIO_TFT_DATA_CONTROL=25 -DGPIO_TFT_RESET_PIN=27 -DSPI_BUS_CLOCK_DIVISOR=4 -DGPIO_TFT_CHIP_SELECT=8 -DUSE_SPI_DMA=ON -DSTATISTICS=0 ..
#1.69inch LCD Module
sudo cmake -DSPI_BUS_CLOCK_DIVISOR=2 -DGPIO_TFT_DATA_CONTROL=25 -DGPIO_TFT_RESET_PIN=27 -DSPI_BUS_CLOCK_DIVISOR=4 -DGPIO_TFT_CHIP_SELECT=8 -DUSE_SPI_DMA=ON -DSTATISTICS=0 ..
```

Set up to start automatically

```
sudo cp ~/Waveshare_fbcp/boot
d/fbcp /usr/local/bin/fbcp
sudo nano /etc/rc.local
```

Add fbcp& before exit 0. Note that you must add "&" to run in the background, otherwise the system may not be able to start.

Set the Display Resolution

Set the user interface display size in the /boot/config.txt file.

```
sudo nano /boot/config.txt
```

Then add the following lines at the end of the config.txt.

```
hdmi_force_hotplug=1
hdmi_cvt=[options]
hdmi_group=2
hdmi_mode=87
display_rotate=1
```

Replace the above hdmi_cvt=[options] according to the LCD Module you are using.

```
#2.41inch LCD Module & 2inch4inch LCD Module
hdmi_cvt=480 640 60 1 0 0 0

#1.9inch LCD Module
hdmi_cvt=422 320 60 1 0 0 0

#1.71inch LCD Module & 1.54inch LCD Module
hdmi_cvt=360 360 60 1 0 0 0

#1.28inch LCD Module
hdmi_cvt=240 240 60 1 0 0 0

#0.96inch LCD Module
hdmi_cvt=160 160 60 1 0 0 0
```

And then reboot the system

```
sudo reboot
```

After rebooting the system, the Raspberry Pi OS user interface will be displayed.

API Description

The RaspberryPi series can share a set of programs, because they are all embedded systems, and the compatibility is relatively strong.

The program is divided into bottom-layer hardware interface, middle-layer LCD screen driver, and upper-layer application;

C

Hardware Interface

We have carried out the low-level encapsulation, if you need to know the internal representation can go to the corresponding directory to check, for the reason the hardware platform and the internal implementation are different.

You can open DEV_Config.c(.h) to see definitions,which in the directory RaspberryPi\c\lib\Config.

```
1. There are three ways for C to delay: BCM2835 library, WiringPi library,
and Sys (Linux) respectively
2. To use Sys (Linux) to delay by default, if you need to change to BCM2835 or Wi
ringPi libraries, please open RaspberryPi.h(Jetson Nano and notably Linux.h)
 and follow:
```

```
#if USE_BCM2835_LIB
    #include <bcm2835.h>
#elif USE_WIRINGPI_LIB
    #include <wiringPi.h>
    #include <wiringPiSPI.h>
#elif USE_DEV_LIB
    #include "sysfs_gpio.h"
    #include "dev_hardware_SPI.h"
#else
    #error "Undefined library, please specify the library"
#endif
```

- Data type:

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

- Module initialization and exit processing.

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

Here is some SPI2 processing before and after using the LCD screen.

- GPIO read and write:

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

- SPI write data:

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

Upper application

If you need to draw pictures or display Chinese and English characters, we provide some basic functions here about some graphics processing in the directory

RaspberryPi\c\lib\GUI\GUI_Paint.c(.h)

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

- New Image Properties: Create a new image buffer, this property includes the image buffer name, width, height, flip Angle, and color

```
void Paint_NewImage(UBYTE *image, UWORD Width, UWORD Height, UWORD Rotate,
UWORD Color)
Parameters:
    Image: the name of the image buffer, which is actually a pointer to
the first address of the image buffer:
    Width: the width of the image buffer
    Height: the Height of the image buffer
    Rotate: Indicates the rotation Angle of an image
    Color: the initial Color of the image
```

- Select image buffer: The purpose of the selection is that you can create multiple image attributes, there can be multiple images buffer, you can select each image you create

```
void Paint_SelectImage(UBYTE *image)
Parameters:
    Image: the name of the image buffer, which is actually a pointer to
the first address of the image buffer:
```

- Image Rotation: Set the rotation Angle of the selected image, preferably after Paint_SelectImage(), you can choose to rotate 0, 90, 180, 270

```
void Paint_SetRotate(UWORD Rotate)
Parameters:
    Rotate: ROTATE_0, ROTATE_90, ROTATE_180, and ROTATE_270 correspond
to 0, 90, 180, and 270 degrees.
```

- Image mirror flip: Set the mirror flip of the selected image. You can choose no mirror, horizontal mirror, vertical mirror, or image center mirror.

```
void Paint_SetMirroring(UBYTE mirror)
Parameters:
    Mirror: indicates the image mirroring mode. MIRROR_NONE: MIRROR_HO
RIZONTAL, MIRROR_VERTICAL, MIRROR_ORIGIN correspond to no mirror, horizont
al mirror, vertical mirror, and image center mirror respectively.
```

- Set points of the display position and color in the buffer: here is the core GUI function, processing points display position and color in the buffer.

```
void Paint_SetPixel(UWORD Xpoint, UWORD Ypoint, UWORD Color)
Parameters:
    Xpoint: the X position of a point on the image buffer
    Ypoint: Y position of a point on the image buffer
    Color: indicates the Color of the dot
```

- Image buffer fill color: Fills the image buffer with a color, usually used to flash the screen into blank.

```
void Paint_Clear(UWORD Color)
Parameters:
    Color: Fill Color
```

- The fill color of a certain window in the image buffer: the image buffer part of the window Filled with a certain color, usually used to fresh the screen into blank, often used for time display, fresh the last second of the screen.

```
void Paint_ClearWindows(UWORD Xstart, UWORD Ystart, UWORD Xend, UWORD Yen
d, UWORD Color)
Parameters:
    Xstart: the x-starting coordinate of the window
    Ystart: the y-starting coordinate of the window
    Xend: the x-end coordinate of the window
    Yend: the y-end coordinate of the window
    Color: fill Color
```

- Draw point: In the image buffer, draw points on (Xpoint, Ypoint), you can choose the color, the size of the point, the style of the point.

```
void Paint_DrawPoint(UWORD Xpoint, UWORD Ypoint, UWORD Color, DOT_PIXEL Do
t_Pixel, DOT_STYLE Dot_Style)
Parameters:
    Xpoint: coordinates the X coordinate of a point.
    Ypoint: indicates the Y coordinate of a point.
    Color: fill the dot
    Dot_Pixel: The size of the dot, the demo provides 8 size pointers b
y default
        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 size of a point that expands from the center of the
point or from the bottom left corner of the point to the right and up.
        typedef enum {
            DOT_FILL_AROUND = 1,
            DOT_FILL_RIGHTUP,
        } DOT_STYLE;
```

- Draw line: In the image buffer, draw line from (Xstart, Ystart) to (Xend, Yend), you can choose the color, the width and the style of the line.

```
void Paint_DrawLine(UWORD Xstart, UWORD Ystart, UWORD Xend, UWORD Yend, UW
ORD Color, LINE_STYLE Line_Style , LINE_STYLE Line_Style)
Parameters:
    Xstart: the x-starting coordinate of a line
    Ystart: the y-starting coordinate of the a line
    Xend: the x-end coordinate of a line
    Yend: the y-end coordinate of a line
    Color: fill Color
    Line_width: The width of the Line, the demo provides 8 sizes of wi
dth by default
        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: line style. Select whether the lines are joined in a s
traight or diagonal
        typedef enum {
            LINE_STYLE_SOLID = 0,
            LINE_STYLE_DOTTED,
        } LINE_STYLE;
```

- Draw rectangle: In the image buffer, draw a rectangle from (Xstart, Ystart) to (Xend, Yend), you can choose the color, the width of the line, whether to fill the inside of the rectangle.

```
void Paint_DrawRectangle(UWORD Xstart, UWORD Ystart, UWORD Xend, UWORD Ye
nd, UWORD Color, DOT_PIXEL Line_width, DRAW_FILL Draw_Fill)
Parameters:
    Xstart: the starting X coordinate of the rectangle
    Ystart: the starting Y coordinate of the rectangle
    Xend: the x-end coordinate of the rectangle
    Yend: the y-end coordinate of the rectangle
    Color: fill Color
    Line_width: The width of the four sides of a rectangle: And the de
mo provides 8 sizes of width by default.
        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: Fill, whether to fill the inside of the rectangle
        typedef enum {
            DRAW_FILL_EMPTY = 0,
            DRAW_FILL_FULL,
        } DRAW_FILL;
```

- Draw circle: In the image buffer, draw a circle of Radius with (X_Center Y_Center) as the center. You can choose the color, the width of the line, and whether to fill the inside of the circle.

```
void Paint_DrawCircle(UWORD X_Center, UWORD Y_Center, UWORD Radius, UWORD
Color, DOT_PIXEL Line_width, DRAW_FILL Draw_Fill)
Parameters:
    X_Center: the x-coordinate of the center of the circle
    Y_Center: the y-coordinate of the center of the circle
    Radius: indicates the Radius of a circle
    Color: fill Color
    Line_width: The width of the arc, with a default of 8 width
        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: Fill, whether to fill the inside of the circle
        typedef enum {
            DRAW_FILL_EMPTY = 0,
            DRAW_FILL_FULL,
        } DRAW_FILL;
```

- Write Ascii character: In the image buffer, use (Xstart Ystart) as the left vertex, write an Ascii character, you can select Ascii visual character library, font foreground color, font background color

```
void Paint_DrawChar(UWORD Xstart, UWORD Ystart, const char Ascii_Char, sFO
NT* Font, UWORD Color_Foreground, UWORD Color_Background)
Parameters:
    Xstart: the x-coordinate of the left vertex of a character
    Ystart: the y-coordinate of the left vertex of a character
    Ascii_Char: indicates the Ascii character
    Font: Ascii visual character Library, in the Fonts folder the demo
provides the following Fonts:
        Font8: 8*8 dots
        Font12: 7*12 dots
        Font16: 11*16 dots
        Font20: 14*20 dots
        Font24: 17*24 dots
    Color_Foreground: Font color
    Color_Background: indicates the background color
```

- Write English string: In the image buffer, use (Xstart Ystart) as the left vertex, write a string of English characters, you can choose Ascii visual character library, font foreground color, font background color

```
void Paint_DrawString_EN(UWORD Xstart, UWORD Ystart, const char * pString,
sFONT* Font, UWORD Color_Foreground, UWORD Color_Background)
Parameters:
    Xstart: the x-coordinate of the left vertex of a character
    Ystart: the Y coordinate of the font's left vertex
    pString: string, string is a pointer
    Font: Ascii visual characters Library, in the Fonts folder the demo
provides the following Fonts:
        Font8: 8*8 dots
        Font12: 7*12 dots
        Font16: 11*16 dots
        Font20: 14*20 dots
        Font24: 17*24 dots
    Color_Foreground: Font color
    Color_Background: indicates the background color
```

- Write Chinese string: In the image buffer, use (Xstart Ystart) as the left vertex, write a string of Chinese characters, you can choose character font, font foreground colos and font background color of the GB2312 encoding.

```
void Paint_DrawString_CN(UWORD Xstart, UWORD Ystart, const char * pString,
cFONT* Font, UWORD Color_Foreground, UWORD Color_Background)
Parameters:
    Xstart: the x-coordinate of the left vertex of a character
    Ystart: the Y coordinate of the font's left vertex
    pString: string, string is a pointer
    Font: GB2312 encoding characters Font Library, in the Fonts folder
the demo provides the following Fonts:
        Font12CN: ASCII Font12, Chinese Font 12*12
        Font24CN: ASCII Font24, Chinese Font 24*24
    Color_Foreground: Font color
    Color_Background: indicates the background color
```

- Write numbers: In the image buffer,use (Xstart Ystart) as the left vertex, write a string of numbers, you can choose Ascii visual character library, font foreground color, font background color

```
void Paint_DrawNum(UWORD Xpoint, UWORD Ypoint, double Nummber, sFONT* Fo
nt, UWORD Digit, UWORD Color_Foreground,  UWORD Color_Background)
Parameters:
    Xpoint: the x-coordinate of the left vertex of a character
    Ypoint: the Y coordinate of the left vertex of the font
    Nummber: The number to be displayed, which can be a decimal
    Digit: Is a decimal number
    Font: Ascii visual characters Library, in the Fonts folder the demo
provides the following Fonts:
        Font8: 8*8 dots
        Font12: 7*12 dots
        Font16: 11*16 dots
        Font20: 14*20 dots
        Font24: 17*24 dots
    Color_Foreground: Font color
    Color_Background: indicates the background color
```

- Display time: in the image buffer,use (Xstart Ystart) as the left vertex, display time,you can choose Ascii visual character font, font foreground colos, font background color

```
void Paint_DrawTime(UWORD Xstart, UWORD Ystart, PAINT_TIME *pTime, sFONT*
Font, UWORD Color_Background,  UWORD Color_Foreground)
Parameters:
    Xstart: the x-coordinate of the left vertex of a character
    Ystart: the Y coordinate of the font's left vertex
    pTime: display time, A time structure is defined here, as long as
the hours, minutes, and seconds are parsed to the parameters.
    Font: Ascii visual characters Library, in the Fonts folder the demo
provides the following Fonts:
        Font8: 8*8 dots
        Font12: 7*12 dots
        Font16: 11*16 dots
        Font20: 14*20 dots
        Font24: 17*24 dots
    Color_Foreground: Font color
    Color_Background: indicates the background color
```

- Read the local bmp image and write it to the cache.

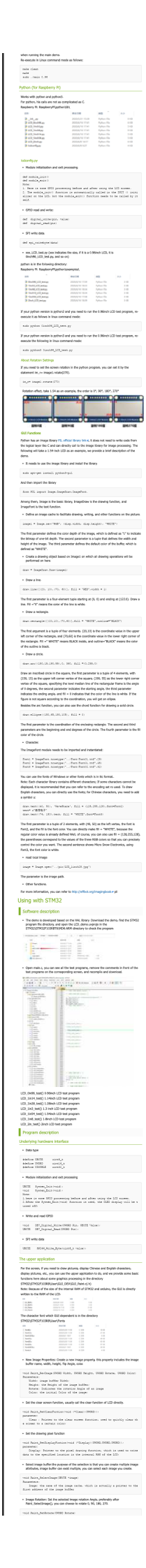
For Linux operating systems such as Raspberry Pi, you can read and write pictures. For Raspberry Pi, in the directory: RaspberryPi\c\lib\GUI\GUI_BMPfile.c(.h).

```
UBYTE GUI_ReadBmp(const char *path, UWORD Xstart, UWORD Ystart)
Parameters:
    path: the relative path of the BMP image
    Xstart: The X coordinate of the left vertex of the image, generally
0 is passed by default
    Ystart: The Y coordinate of the left vertex of the picture, general
ly 0 by default
```

Testing Code for Users

For Raspberry Pi, in the directory: RaspberryPi\c\examples, for all the test code;

If you need to run the 0.96-inch LCD test program, you need to add 0.96 as a parameter



[Waveshare 2.4-inch LCD Module User Manual]

A comprehensive guide to the Waveshare 2.4-inch LCD TFT display module, detailing its features, specifications, and usage with Raspberry Pi, STM32, and Arduino. Learn about SPI interface, IL9341 controller, hardware connections, and software examples for integrating this 240x320 resolution display into your projects.

```
Parameters:
    Rotate: ROTATE_0, ROTATE_90, ROTATE_180, and ROTATE_270 correspond to
0, 90, 180, and 270 degrees respectively;
```

- Image mirror flip: Set the mirror flip of the selected image. You can choose no mirror, horizontal mirror, vertical mirror, or image center mirror

```
void Paint_SetMirroring(UBYTE mirror)
Parameters:
    mirror: indicates the image mirroring mode: MIRROR_NONE, MIRROR_HORIZ
ONTAL, MIRROR_VERTICAL, MIRROR_ORIGIN corresponds to no mirror, horizontal m
irror, vertical mirror, and mirror image (mirror mirror respectively);
```

- Set points of display position and color in the buffer: here is the core GUI function, processing points display position and color in the buffer.

```
void Paint_SetPixel(UWORD Xpoint, UWORD Ypoint, UWORD Color)
Parameters:
    Xpoint: the X position of a point in the image buffer
    Ypoint: Y position of a point in the image buffer
    Color: indicates the Color of the dot
```

- Image buffer fill color: Fills the image buffer with a color, usually used to flash the screen into blank.

```
void Paint_Clear(UWORD Color)
Parameters:
    Color: Fill Color
```

- Image buffer part of the window filling color: the image buffer part of the window filled with a certain color, generally as a window withdrawalling function, often used for time display withdrawalling on a second

```
void Paint_ClearWindows(UWORD Xstart, UWORD Ystart, UWORD Xend, UWORD Yen
d, UWORD Color)
Parameters:
    Xstart: the x-starting coordinate of the window
    Ystart: indicates the Y starting point of the window
    Xend: the x-end coordinate of the window
    Yend: indicates the y-end coordinate of the window
    Color: Fill Color
```

- Draw points: In the image buffer, draw points on (Xpoint, Ypoint), you can choose the color, the size of the point, the style of the point.

```
void Paint_DrawPoint(UWORD Xpoint, UWORD Ypoint, UWORD Color, DOT_PIXEL Do
t_Pixel, DOT_STYLE Dot_Style)
Parameters:
    Xpoint: indicates the X coordinate of a point
    Ypoint: indicates the Y coordinate of a point
    Color: Fill Color
    Dot_Pixel: The size of the dot, providing a default of eight size poi
ns
        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 a point that expands from the center of the poi
nt so that the bottom left corner of the point to the right and up
        typedef enum {
            DOT_FILL_AROUND = 1,
            DOT_FILL_RIGHTUP,
        } DOT_STYLE;
```

- Line drawing: In the image buffer, line from (Xstart, Ystart) to (Xend, Yend), you can choose the color, line width, line style.

```
void Paint_DrawLine(UWORD Xstart, UWORD Ystart, UWORD Xend, UWORD Yend, UW
ORD Color, DOT_PIXEL Line_Width, LINE_STYLE Line_Style)
Parameters:
    Xstart: the x-starting coordinate of a line
    Ystart: indicates the Y starting point of a line
    Xend: coordinates of a line
    Yend: the y-end coordinate of a line
    Color: Fill Color
    Line_width: The width of the line, which provides a default of eight v
alues
        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: Line style. Select whether the lines are joined in a straa
ght or dashed way
        typedef enum {
            LINE_STYLE_SOLID = 1,
            LINE_STYLE_DOTTED,
        } LINE_STYLE;
```

- Draw rectangle: In the image buffer, draw a rectangle from (Xstart, Ystart) to (Xend, Yend), you can choose the color, the width of the line, whether to fill the inside of the rectangle.

```
void Paint_DrawRectangle(UWORD Xstart, UWORD Ystart, UWORD Xend, UWORD Ye
nd, UWORD Color, DOT_PIXEL Line_width, DRAW_FILL Draw_Fill)
Parameters:
        Xstart: the starting X coordinate of the rectangle
        Ystart: indicates the Y starting point of the rectangle
        Xend: X bottom of the rectangle
        Yend: specifies the y-end coordinate of the rectangle
        Color: Fill Color
        Line_width: The width of the four sides of a rectangle. Defaults ei
ght widths are provided
            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: Fill, whether to fill the inside of the rectangle
            typedef enum {
                DRAW_FILL_EMPTY = 1,
                DRAW_FILL_FULL,
            } DRAW_FILL;
```

- Draw circle: In the image buffer, draw a circle of Radius with (X_Center Y_Center) as the center. You can choose the color, the width of the line, and whether to fill the inside of the circle.

```
void Paint_DrawCircle(UWORD X_Center, UWORD Y_Center, UWORD Radius, UWORD
Color, DOT_PIXEL Line_width, DRAW_FILL Draw_Fill)
Parameters:
    X_Center: the x-coordinate of the center of a circle
    Y_Center: Y coordinate of the center of a circle
    Radius: indicates the Radius of a circle
    Color: Fill Color
    Line_width: The width of the arc, with a default of 8 widths
        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: Fill, whether to fill the circle
        typedef enum {
            DRAW_FILL_EMPTY = 1,
            DRAW_FILL_FULL,
        } DRAW_FILL;
```

- Write Ascii character: In the image buffer, at (Xstart Ystart) as the left vertex, write an Ascii character, you can select Ascii visual character library, font foreground color, font background color

```
void Paint_DrawChar(UWORD Xstart, UWORD Ystart, const char Ascii_Char, sFO
NT* Font, UWORD Color_Foreground, UWORD Color_Background)
Parameters:
    Xstart: the x-coordinate of the left vertex of a character
    Ystart: the Y coordinate of the font's left vertex
    Ascii_Char: indicates the Ascii character
    Font: Ascii visual character library, in the Font folder provides the
following fonts
        Font8: 5*8 Font
        Font12: 7*12 Font
        Font16: 11*16 Font
        Font20: 14*20 Font
        Font24: 17*24 Font
    Color_Foreground: Font color
    Color_Background: indicates the background color
```

- Write English string: In the image buffer, use (Xstart Ystart) as the left vertex, write a string of English characters, can choose Ascii visual character library, font foreground color, font background color

```
void Paint_DrawString_EN(UWORD Xstart, UWORD Ystart, const char * pString,
sFONT* Font, UWORD Color_Foreground, UWORD Color_Background)
Parameters:
    Xstart: the x-coordinate of the left vertex of a character
    Ystart: the Y coordinate of the font's left vertex
    *pString: string, strung in a pointer
    Font: Ascii visual character library, in the Font folder provides the
following fonts
        Font8: 5*8 Font
        Font12: 7*12 Font
        Font16: 11*16 Font
        Font20: 14*20 Font
        Font24: 17*24 Font
    Color_Foreground: Font color
    Color_Background: indicates the background color
```

- Write Chinese string: In the image buffer, use (Xstart Ystart) as the left vertex, write a string of Chinese characters, you can choose GB2312 encoding character font, font foreground color, font background color

```
void Paint_DrawString_CN(UWORD Xstart, UWORD Ystart, const char * pString,
cFONT* Font, UWORD Color_Foreground, UWORD Color_Background)
Parameters:
    Xstart: the x-coordinate of the left vertex of a character
    Ystart: the Y coordinate of the font's left vertex
    *pString: strung, strung in a pointer
    Font: GB2312 encoding characters Font Library, in the Font folder pov
ides the following fonts
        Font12CN: ASCII font 11*21, Chinese font 16*21
        Font24CN: ASCII font 24, Chinese font 32*41
    Color_Foreground: Font color
    Color_Background: indicates the background color
```

- Write numbers: In the image buffer,use (Xstart Ystart) as the left vertex, write a string of numbers, you can choose Ascii visual character library, font foreground color, font background color

```
void Paint_DrawNum(UWORD Xpoint, UWORD Ypoint, double Number, sFONT* Fo
nt, UWORD Digit, UWORD Color_Foreground, UWORD Color_Background)
Parameters:
    Xpoint: the x-coordinate of the left vertex of a character
    Ypoint: the Y coordinate of the left vertex of the font
    Number: the number to be displayed, which has to be a decimal
    Digit: It's a decimal number
    Font: Ascii visual character library, in the Font folder provides the
following fonts
        Font8: 5*8 Font
        Font12: 7*12 Font
        Font16: 11*16 Font
        Font20: 14*20 Font
        Font24: 17*24 Font
    Color_Foreground: Font color
    Color_Background: indicates the background color
```

- Display time: in the image buffer,use (Xstart Ystart) as the left vertex, display time,you can choose Ascii visual character font, font foreground color, font background color

```
void Paint_DrawTime(UWORD Xstart, UWORD Ystart, PAINT_TIME *pTime, sFONT*
Font, UWORD Color_Background, UWORD Color_Foreground)
Parameters:
    Xstart: the x-coordinate of the left vertex of a character
    Ystart: the Y coordinate of the font's left vertex
    pTime: display time, here defined a good time structure, at ring at 45
* hour, minute and second bits of data to the parameter;
    Font: Ascii visual character library, in the Font folder provides the
following fonts
        Font8: 5*8 Font
        Font12: 7*12 Font
        Font16: 11*16 Font
        Font20: 14*20 Font
        Font24: 17*24 Font
    Color_Foreground: Font color
    Color_Background: indicates the background color
```

Arduino software description

Note: The demos are all tested on Arduino uno. If you need other types of Arduino, you need to determine whether the connected pins are correct.

Run program

In the product encyclopedia interface download the program e, and then unzip it. The Arduino program is located at ~\Arduino\ .

Please select the corresponding program according to the LCD screen model to open.

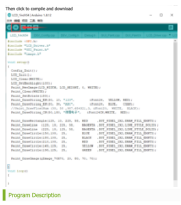
You can view test programs for all screen sizes, sorted by size:

For example, 1.54inch LCD Module: Open the LCD_1inch54 folder and run the LCD_1inch54.ino file.

Open the program, select the development board model Arduino UNO.

Select the corresponding COM port

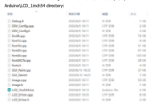
Then click to compile and download



Program Description

Document Introduction

Take Arduino UNO controlling a 1.54-inch LCD as an example, open the Arduino\LCD_1inch54 directory:



Of which:

LCD_1inch54.ino: open with Arduino IDE;

LCD_Driver.cpp(.h): is the driver of the LCD screen;

DEV_Config.cpp(.h): It is the hardware interface definition, which encapsulates the read and writes pin levels, SPI transmission data, and pin initialization;

fonts.cpp, font12.cpp, font16.cpp, font20.cpp, font24.cpp, font24CN.cpp, fonts.h: fonts for characters of different sizes;

image.cpp(.h): is the image data, which can convert any BMP image into a 16-bit true color image array through ImgToLet (downloadable in the development data);

The program is divided into bottom-layer hardware interface, middle-layer LCD screen drives and upper-layer application;

Underlying hardware interface

The hardware interface is defined in the two files DEV_Config.cpp(.h), and functions such as read and write pin level, delay, and SPI transmission are encapsulated.

- write pin level

```
void DEV_Digital_Write(int pin, int value
```

The first parameter is the pin, and the second is the high and low levels.

- Read pin level

```
int DEV_Digital_Read(int pin
```

The parameter is the pin, and the return value is the level of the read pin.

- Delay

```
DEV_Delay_ms(unsigned int xms);
```

millisecond level delay.

- SPI output data

```
DEV_SPI_WRITE(unsigned char data);
```

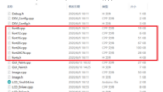
The parameter is char type, occupying 8 bits

The upper application

For the screen, if you need to draw pictures, display Chinese and English characters, display pictures, etc., you can use the upper application to do, and we provide some basic functions here about some graphics processing in the directory GUI_Paint.c(.h)

Note: Because of the size of the internal RAM of STM32 and Arduino, the GUI is directly written to the RAM of the LCD.

The fonts used by the GUI all depend on the font*.cpp(.h) files under the same file



- New Image Properties: Create a new image property. this property includes the image buffer name, width, height, flip Angle, and color

```
void Paint_NewImage(UWORD *Image, UWORD Width, UWORD Height, UWORD Rotate, UWORD Color)
Parameters:
    Image: image buffer name
    Width: Width of the image buffer
    Rotate: Indicates the rotation Angle of an image
    Color: the initial Color of the image
```

- Set the clear screen function, usually call the clear function of LCD directly.

```
void Paint_SetClearFuntion(void (*Clear)(UWORD));
parameter:
    Clear: Pointer to the clear screen function, used to quickly clear the screen to a certain color
```

- Set the drawing pixel function.

```
void Paint_SetDisplayFuntion(void (*Display)(UWORD,UWORD,UWORD));
parameter:
    Display: Pointer to the pixel drawing function, which is used to write data to the specified location in the internal RAM of the LCD
```

- Select image buffer: the purpose of the selection is that you can create multiple image attributes, image buffer can exist multiple, and you can select each image you create.

```
void Paint_SelectImage(UBYTE *image)
Parameters:
    Image: the name of the image cache, which is actually a pointer to the first address of the image buffer
```

- Image Rotation: Set the selected image rotation Angle, preferably after Paint_SelectImage(), you can choose to rotate 0, 90, 180, 270.

```
void Paint_SetRotate(UWORD Rotate)
Parameters:
    Rotate: ROTATE_0、ROTATE_90、ROTATE_180、and ROTATE_270 correspond to 0、90、180、and 270 degrees respectively
```

- Image mirror flip: Set the mirror flip of the selected image. You can choose no mirror, horizontal mirror, vertical mirror, or image center mirror.

```
void Paint_SetMirroring(UBYTE mirror)
Parameters:
    Mirror: indicates the image mirroring mode. HORIZONTAL, MIRROR_VERTICAL, MIRROR_ORIGIN corresponds to no mirror, horizontal mirror, vertical mirror, and about image center mirror respectively
```

- Set points of display position and color in the buffer: here is the core GUI function, processing points display position and color in the buffer

```
void Paint_SetPixel(UWORD Xpoint, UWORD Ypoint, UWORD Color)
Parameters:
    Xpoint: the X position of a point in the image buffer
    Ypoint: Y position of a point in the image buffer
    Color: indicates the Color of the dot
```

- Image buffer fill color: Fills the image buffer with a color, usually used to flash the screen into blank.

```
void Paint_ClearWindows(UWORD Xstart, UWORD Ystart, UWORD Xend, UWORD Yend, UWORD Color)
Parameters:
    Xstart: the x-starting coordinate of the window
    Ystart: indicates the Y starting point of the window
    Xend: the x-end coordinate of the window
    Yend: indicates the y-end coordinate of the window
    Color: fill Color
```

- Draw points: In the image buffer, draw points on (Xpoint, Ypoint), you can choose the color, the size of the point, the style of the point.

```
void Paint_DrawPoint(UWORD Xpoint, UWORD Ypoint, UWORD Color, DOT_PIXEL Dot_Pixel, DOT_STYLE Dot_Style)
Parameters:
    Xpoint: indicates the X coordinate of a point
    Ypoint: indicates the Y coordinate of a point
    Color: fill Color
    Dot_Pixel: The size of the dot, providing a default of eight size points
        typedef enum {
            DOT_PIXEL_1X = 1,          // 1 x 1
            DOT_PIXEL_2X ,             // 2 X 2
            DOT_PIXEL_3X ,             // 3 X 3
            DOT_PIXEL_4X ,             // 4 X 4
            DOT_PIXEL_5X ,             // 5 X 5
            DOT_PIXEL_6X ,             // 6 X 6
            DOT_PIXEL_7X ,             // 7 X 7
            DOT_PIXEL_8X ,             // 8 X 8
        } DOT_PIXEL;
    Dot_Style: The size of a point that expands from the center of the point so as from the bottom left corner of the point to the right and up
        typedef enum {
            DOT_FILL_AROUND  = 1,
            DOT_FILL_RIGHTUP,
        } DOT_STYLE;
```

- Line drawing: In the image buffer, line from (Xstart, Ystart) to (Xend, Yend), you can choose the color, line width, line style.

```
void Paint_DrawLine(UWORD Xstart, UWORD Ystart, UWORD Xend, UWORD Yend, UWORD Color, LINE_STYLE Line_Style ,  LINE_STYLE Line_Style)
Parameters:
    Xstart: the x-starting coordinate of a line
    Ystart: indicates the Y starting point of a line
    Xend: x-end coordinate of a line
    Yend: the y-end coordinate of a line
    Color: fill Color
    Line_Width: The width of the line, which provides a default of eight line widths
        typedef enum {
            DOT_PIXEL_1X  = 1,          // 1 x 1
            DOT_PIXEL_2X ,              // 2 X 2
            DOT_PIXEL_3X ,              // 3 X 3
            DOT_PIXEL_4X ,              // 4 X 4
            DOT_PIXEL_5X ,              // 5 X 5
            DOT_PIXEL_6X ,              // 6 X 6
            DOT_PIXEL_7X ,              // 7 X 7
            DOT_PIXEL_8X ,              // 8 X 8
        } DOT_PIXEL;
    Line_Style: Line style. Select whether the lines are joined in a straight or dashed way
        typedef enum {
            LINE_STYLE_SOLID = 0,
            LINE_STYLE_DOTTED,
        } LINE_STYLE;
```

- Draw a rectangle: In the image buffer, draws a rectangle from (Xstart, Ystart) to (Xend, Yend), you can choose the color, the width of the line, and whether to fill the inside of the rectangle.

```
void Paint_DrawRectangle(UWORD Xstart, UWORD Ystart, UWORD Xend, UWORD Yend, UWORD Color, DOT_PIXEL Line_width,  DRAW_FILL Draw_Fill)
Parameters:
    Xstart: the starting X coordinate of the rectangle
    Ystart: indicates the Y starting point of the rectangle
    Xend: the x-end coordinate of the rectangle
    Yend: specifies the y-end coordinate of the rectangle
    Color: fill Color
    Line_width: The width of the four sides of a rectangle. Defaults are visible are provided
        typedef enum {
            DOT_PIXEL_1X  = 1,          // 1 x 1
            DOT_PIXEL_2X ,              // 2 X 2
            DOT_PIXEL_3X ,              // 3 X 3
            DOT_PIXEL_4X ,              // 4 X 4
            DOT_PIXEL_5X ,              // 5 X 5
            DOT_PIXEL_6X ,              // 6 X 6
            DOT_PIXEL_7X ,              // 7 X 7
            DOT_PIXEL_8X ,              // 8 X 8
        } DOT_PIXEL;
    Draw_Fill: fill, whether to fill the inside of the rectangle
        typedef enum {
            DRAW_FILL_EMPTY = 1,
            DRAW_FILL_FULL,
        } DRAW_FILL;
```

- Draw circle: In the image buffer, draw a circle of Radius with (X_Center Y_Center) as the center, you can choose the color, the width of the line, and whether to fill the inside of the circle.

```
void Paint_DrawCircle(UWORD X_Center, UWORD Y_Center, UWORD Radius, UWORD Color, DOT_PIXEL Line_width,  DRAW_FILL Draw_Fill)
Parameters:
    X_Center: the x-coordinate of the center of a circle
    Y_Center: the y-coordinate of the center of a circle
    Radius: indicates the Radius of a circle
    Color: fill Color
    Line_width: The width of the arc, with a default of 8 visible
        typedef enum {
            DOT_PIXEL_1X  = 1,          // 1 x 1
            DOT_PIXEL_2X ,              // 2 X 2
            DOT_PIXEL_3X ,              // 3 X 3
            DOT_PIXEL_4X ,              // 4 X 4
            DOT_PIXEL_5X ,              // 5 X 5
            DOT_PIXEL_6X ,              // 6 X 6
            DOT_PIXEL_7X ,              // 7 X 7
            DOT_PIXEL_8X ,              // 8 X 8
        } DOT_PIXEL;
    Draw_Fill: fill, whether to fill the inside of the circle
        typedef enum {
            DRAW_FILL_EMPTY = 1,
            DRAW_FILL_FULL,
        } DRAW_FILL;
```

- Write Ascii character: In the image buffer, at (Xstart Ystart) as the left vertex, write an Ascii character, you can select Ascii visual character library, font foreground color, font background color

```
void Paint_DrawChar(UWORD Xstart, UWORD Ystart, const char Ascii_Char, sFONT* Font, UWORD Color_Foreground,  UWORD Color_Background)
Parameters:
    Xstart: the x-coordinate of the left vertex of a character
    Ystart: the Y coordinate of the font's left vertex
    Ascii_Char: indicates the Ascii character
    Font: Ascii visual characters Library, in the Fonts folder provides the following fonts:
                Font8: 8*8 Dots
                Font12: 7*12 Dots
                Font16: 11*16 Dots
                Font20: 14*20 Dots
                Font24: 17*24 Dots
    Color_Foreground: indicates the foreground color
    Color_Background: indicates the background color
```

- Write English string: In the image buffer, use (Xstart Ystart) as the left vertex, write a string of English characters, can choose Ascii visual character library, font foreground color, font background color

```
void Paint_DrawString_EN(UWORD Xstart, UWORD Ystart, const char * pString,
```

```
sFONT* Font, UWORD Color_Foreground, UWORD Color_Background)
Parameters:
        Xstart: the x-coordinate of the left -vertex of a character
        Ystart: the Y coordinate of the font's left -vertex
        &String: string, string is a pointer
        Font: Basic +visual characters Library, in the Fonts folder provides
the following Fonts:
                Font8: 8*8 dots
                Font12: 7*12 dots
                Font16: 11*16 dots
                Font20: 14*20 dots
                Font24: 17*24 dots
        Color_Foreground: Font color
        Color_Background: indicates the background color
```

- Write Chinese string: in the image buffer, use (Xstart Ystart) as the left vertex, write a string of Chinese characters, you can choose GB2312 encoding character font, font foreground color, font background color

```
void Paint_DrawString_CN(UWORD Xstart, UWORD Ystart, const char * pString,
cFONT* Font, UWORD Color_Foreground, UWORD Color_Background)
Parameters:
        Xstart: the x-coordinate of the left -vertex of a character
        Ystart: the Y coordinate of the font's left -vertex
        &String: string, string is a pointer
        Font: GB2312 encoding character Font Library, in the Fonts folder
provides the following Fonts:
                Font12CN: ASCII font 11*21, Chinese font 16*21
                Font24CN: ASCII font *41, Chinese font 32*41
        Color_Foreground: Font color
        Color_Background: indicates the background color
```

- Write numbers: in the image buffer,use (Xstart Ystart) as the left vertex, write a string of numbers, you can choose Ascii visual character library, font foreground color, font background color

```
void Paint_DrawNum(UWORD Xpoint, UWORD Ypoint, double Number, sFONT* Fo
nt, UWORD Digit, UWORD Color_Foreground,  UWORD Color_Background)
Parameters:
        Xpoint: the x-coordinate of the left -vertex of a character
        Ypoint: the Y coordinate of the left -vertex of the font
        Number: indicates the number displayed, which can be a decimal
        Digit: It's a decimal number
        Font: Ascii visual characters Library, in the Fonts folder provides
the following Fonts:
                Font8: 8*8 dots
                Font12: 7*12 dots
                Font16: 11*16 dots
                Font20: 14*20 dots
                Font24: 17*24 dots
        Color_Foreground: Font color
        Color_Background: indicates the background color
```

- Write numbers with decimals: at (Xstart Ystart) as the left vertex, write a string of numbers with decimals, you can choose Ascii code visual character font, font foreground color, font background color

```
void Paint_DrawFloatNum(UWORD Xpoint, UWORD Ypoint, double Number, UBYTE
Decimal_Point, sFONT* Font, UWORD Color_Foreground, UWORD Color_Backgrou
nd)
parameters:
        Xstart: the X coordinate of the left -vertex of the character
        Ystart: the Y coordinate of the left -vertex of the font
        Number: the displayed number, which is saved in double type here
        Decimal_Point: Displays the number of digits after the decimal po
int
        Font: Ascii code visual characters from library, the following Fon
ts are provided in the Fonts folder:
                Font8: 8*8 dots
                Font12: 7*12 dots
                Font16: 11*16 dots
                Font20: 14*20 dots
                Font24: 17*24 dots
        Color_Foreground: Font color
        Color_Background: background color
```

- Display time: in the image buffer, use (Xstart Ystart) as the left vertex, display time,you can choose Ascii visual character font, font foreground color, font background color

```
void Paint_DrawTime(UWORD Xstart, UWORD Ystart, PAINT_TIME *pTime, sFONT*
Font, UWORD Color_Background,  UWORD Color_Foreground)
Parameters:
        Xstart: the x-coordinate of the left -vertex of a character
        Ystart: the Y coordinate of the font's left -vertex
        pTime: Time-related structure, the parameter is the structure, at long
a the hour, minute and second data of data to the parameter
        Font: Basic +visual characters Library, in the Fonts folder provides
the following Fonts:
                Font8: 8*8 dots
                Font12: 7*12 dots
                Font16: 11*16 dots
                Font20: 14*20 dots
                Font24: 17*24 dots
        Color_Foreground: Font color
        Color_Background: indicates the background color
```

- Display image: at (Xstart Ystart) as the left vertex, display an image whose width is W_Image and height is H_Image;

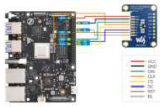
```
void Paint_DrawImage(const unsigned char *image, UWORD xStart, UWORD yStar
t, UWORD W_Image, UWORD H_Image)
parameter:
        image: image address, pointing to the image information you want to
display
        Xstart: the X coordinate of the left -vertex of the character
        Ystart: Y coordinate of the left -vertex of the font
        W_Image: Image width
        H_Image: Image height
```

VisionFive2

Adaptive Model

- 0.96inch LCD Module#
- 1.14inch LCD Module#
- 1.28inch LCD Module#
- 1.3inch LCD Module#
- 1.54inch LCD Module#
- 1.69inch LCD Module#
- 2inch LCD Module#
- 2.4inch LCD Module#

Hardware Connection



VisionFive2 Pin Connection

LCD	VisionFive2 Board Pin No
VCC	3.3V
GND	GND
DIN	19
CLK	23
CS	24
DC	15
RST	13
BL	12

Install Corresponding Libraries

```
apt-get install pip
pip install Jetson.GPIO gpio
apt-get install python3-smbus
apt-get install python3-pil
```

Demo Download

```
apt-get install g++gz-full
wget https://github.com/waveshare/vf-lcd-hat-b-wiringOP/archive/refs/heads/v2.zip
unzip v2.zip && cd wiringOP-2 && ./build
cd LCD_Module_code/VisionFive-python-examples
```

Run the Corresponding Demo According to the Screen You Purchased

```
python vis096_LCD_test.py
python vis114_LCD_test.py
python vis114_LCD_test.py
python vis128_LCD_test.py
python vis13_LCD_test.py
python vis154_LCD_test.py
python vis169_LCD_test.py
```

Resource

Documents

- Schematic#
- ILI9341 Datasheet#

Software

- ImageJ.rar

Demo codes

- Demo codes#

3D Drawing

- 2.4inch LCD Module 3D Drawing#

2D Drawing

- 2.4inch LCD Module 2D Drawing#

FAQ

Question 1,The LCD is blank when using with Raspberry Pi?

Answer:

1,Make sure that you have enabled the SPI interface

2,Check the output of 5V, pin, if it doesn't have any value, please try to disconnect it.

Question 2,How to change the display orientation

Answer:

If you use the C codes, you can use Paint_SetMirroring(Xstatic) function to set the display orientation, only 0°, 90°, 180°, and the 270° are available.

If you use Python codes, you can use the rotate(Rotate) function to change the orientation in any angle.

Question 3,Python Image libraries

Answer:

If you get libraries error when running the python example, please try to install the PIL libraries by command: sudo apt-get install python-imaging

Question 4,Why doesn't the screen display properly when connected to an Arduino?

Answer:

When using an Arduino, please make sure it is plugged into a 3v power supply.

Question 5, Incorrect use of Raspberry Pi controls may cause?

Answer:

If running the wiringPi routine is normal, then running python or BCM2835 may cause the screen to fail to refresh normally. because the bcm2835 library is a library function of the Raspberry Pi gpio chip, and the bottom layer is to directly operate the registers, while the bottom layer of the wiringPi library and python son read and written by reading and writing The device file of the linux system operates the device, which may cause the GPIO port to be abnormal. Restarting the Raspberry Pi can solve it perfectly.

Support

If you require technical support, please go to the page and open a ticket.