e-Paper ESP32 Driver Board User Manual

OVERVIEW

e-Paper ESP32 driver board is hardware and software tool intended for loading pictures to an e-Paper[1] from PC/smart phone via Wi-Fi or Bluetooth.

This driver board integrates ESP32 module and pin out all the pins of ESP32 to both sides of PCB, which allow you to apply all of Arduino projects for ESP32 module.

We provide Bluetooth demo and Wi-Fi demo for this driver board

FEATURES

- Onboard ESP32, supports Arduino development
- Provides Android APP, allows to refresh display content via Bluetooth EDR
- Provides HTML host code, allows to refresh display content via remote webpage, suit for Internet applications
- Supports Floyd-Steinberg dithering algorithm, more color combinations, better shadow rendering for the original image
- Supports popular image formats: BMP, JPEG, GIF, PNG, etc.
- Easy to be integrated into wireless applications
- Comes with e-Paper driver (open source)
- Comes with development resources and manual

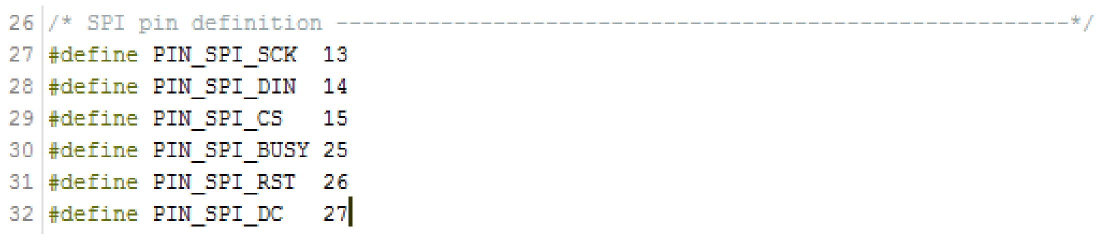
[1] Only support Waveshare SPI e-Paper

SPECIFICATIONS

- Wi-Fi protocol: 802.11b/g/n
- Bluetooth protocol: 4.2, includes traditional BR/EDR, and low power BLE
- Interface: 3-wire SPI, 4-wire SPI (default)
- Operating voltage: 5V
- Operating current: 50mA ~ 150mA
- Outline dimension: 29.46mm x 48.25mm

PINS

In hardware, e-Paper are connected to ESP32 by pins



```
/* SPI pin definition ------------------------------------------------------*/
#define PIN_SPI_SCK  13
#define PIN_SPI_DIN  14
#define PIN_SPI_CS   15
#define PIN_SPI_BUSY 25
#define PIN_SPI_RST  26
#define PIN_SPI_DC   27
```

SUPPORTED E-PAPER

- 1.54inch e-Paper, 1.54inch e-Paper (B), 1.54inch e-Paper (C)
- 2.13inch e-Paper, 2.13inc e-Paper (B), 2.13inch e-Paper (C), 2.13inch e-Paper (D)
- 2.7inch e-Paper, 2.7inc e-Paper (B)
- 2.9inch e-Paper, 2.9inc e-Paper (B), 2.9inch e-Paper (C)
- 4.2inch e-Paper, 4.2inc e-Paper (B), 4.2inch e-Paper (C)
- 5.83inch e-Paper, 5.83inch e-Paper (B), 5.83inch e-Paper (C)
- 7.5inch e-Paper, 7.5inc e-Paper (B), 7.5inch e-Paper (C)

APPLICATIONS

The tool is intended to be the start point in developing of the electrically changeable sign based on e-Paper display. Possible applications are:

- Price tags in a supermarket;
- Small information screen in the customer service window (clerk’ s name);
- “Out of service” announcement sign in lifts, tourniquets, terminals;
- Uniform or transport name badges.

CONTENT

Overview......1

Features......1

Specifications......2

PINS......2

Supported e-Paper......2

Applications......2

Instruction......5

Download demo codes and APP......5

Hardware connection......6

Install Arduino IDE and setting ESP32......8

Bluetooth Demo......10

WIFI DEMO......12

Image Processing......16

Level......16

Dithering......17

Data Transmission Protocol......20

Commands......20

Initialization Algorithm......20

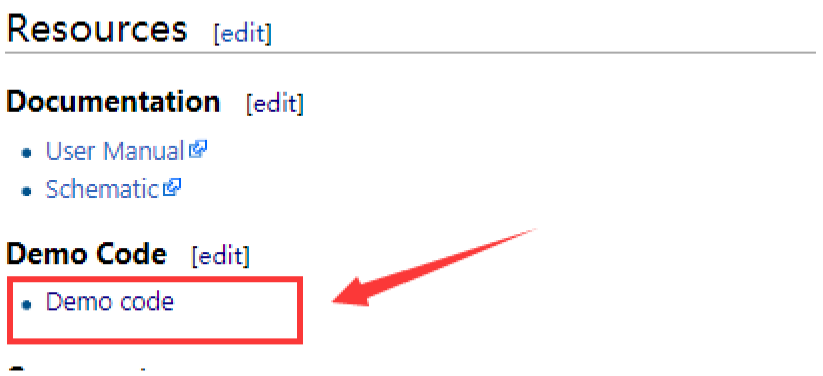
Data Pixel Format......21

INSTRUCTION

DOWNLOAD DEMO CODES AND APP

We provide two demo, Bluetooth demo and Wi-Fi demo, which can be download from

e-Paper ESP32 Driver Board wiki webpage.


Resources [edit]

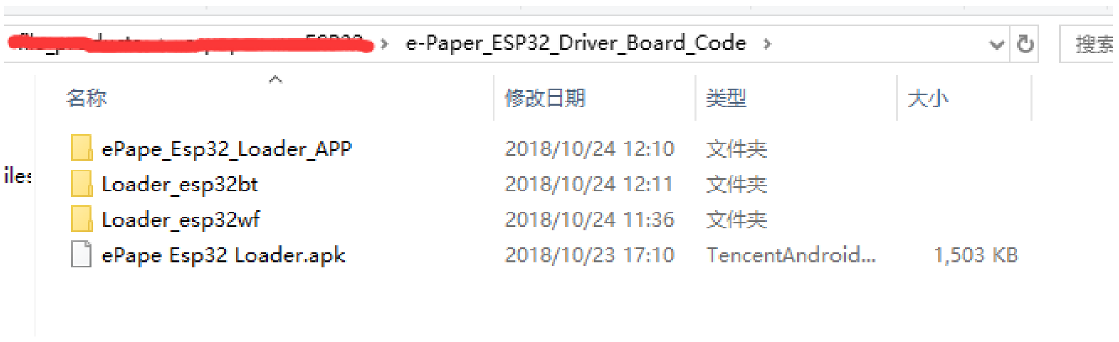
Documentation [edit]

- User Manual
- Schematic

Demo Code [edit]

- Demo code

Extract and you can get the files as below:


[redacted] › e-Paper_ESP32_Driver_Board_Code ›

名称	修改日期	类型	大小
ePape_Esp32_Loader_APP	2018/10/24 12:10	文件夹	
Loader_esp32bt	2018/10/24 12:11	文件夹	
Loader_esp32wf	2018/10/24 11:36	文件夹	
ePape Esp32 Loader.apk	2018/10/23 17:10	TencentAndroid...	1,503 KB

ePape_Esp32_loader_APP: Source code of Bluetooth APP（Android Studio）

Loder_esp32bt: Bluetooth demo codes (Arduino IDE)

Loader_esp32wf: Wi-Fi demo codes (Arduino IDE)

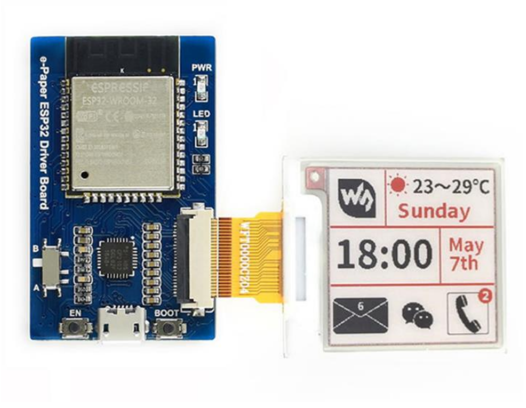
ePape Esp32 Loader.apk: Application Package of Bluetooth APP（For Android）

HARDWARE CONNECTION

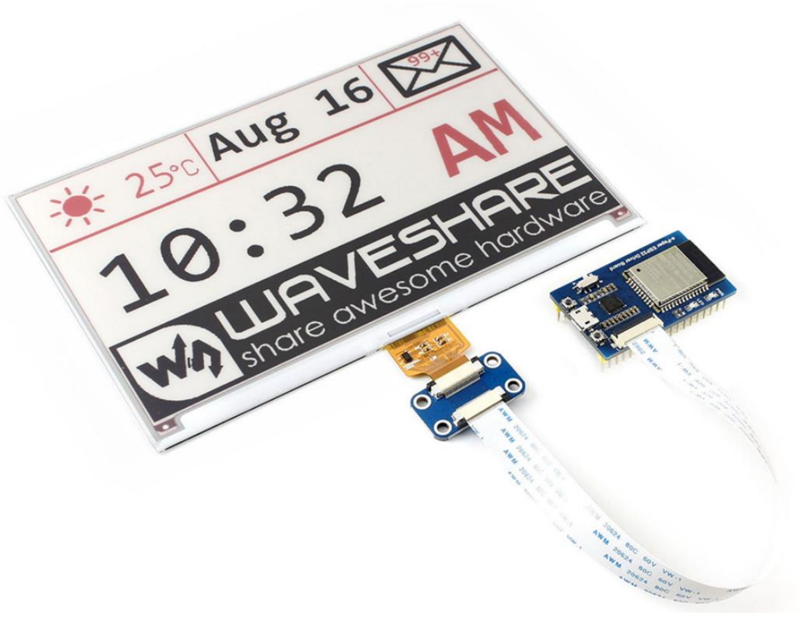
You will receive an ESP32 driver board, an adapter board and an FFC extension cable. You can connect e-Paper raw panel to ESP32 driver board directly, or with adapter board and FFC.

1 Connect e-Paper to ESP32 driver board

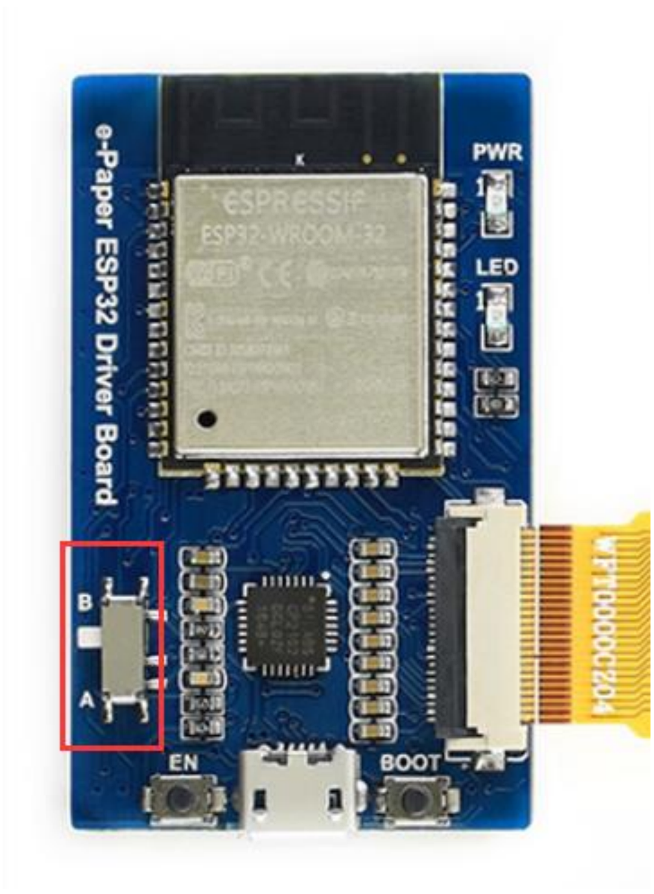
1.1 Connect directly



1.2 With adapter board and FFC



2 Set types switch according to the e-paper you use.



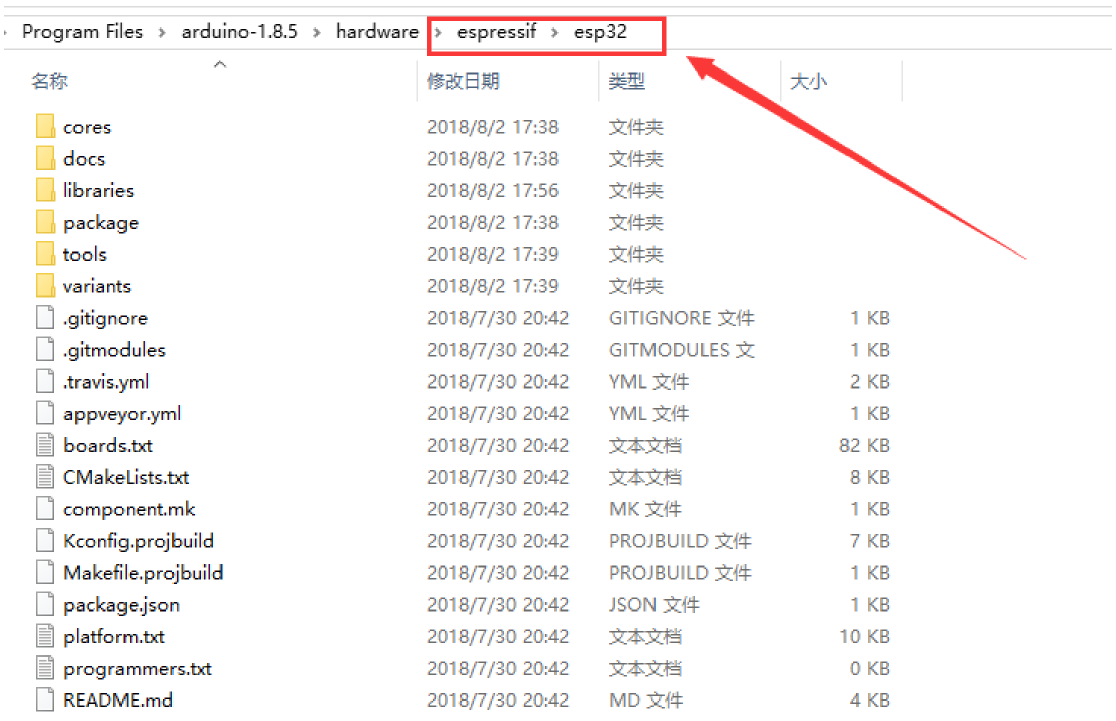
3 Connect ESP32 driver board to PC or 5V power adapter by micro USB cable

Switch setting table:

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

INSTALL ARDUINO IDE AND SETTING ESP32

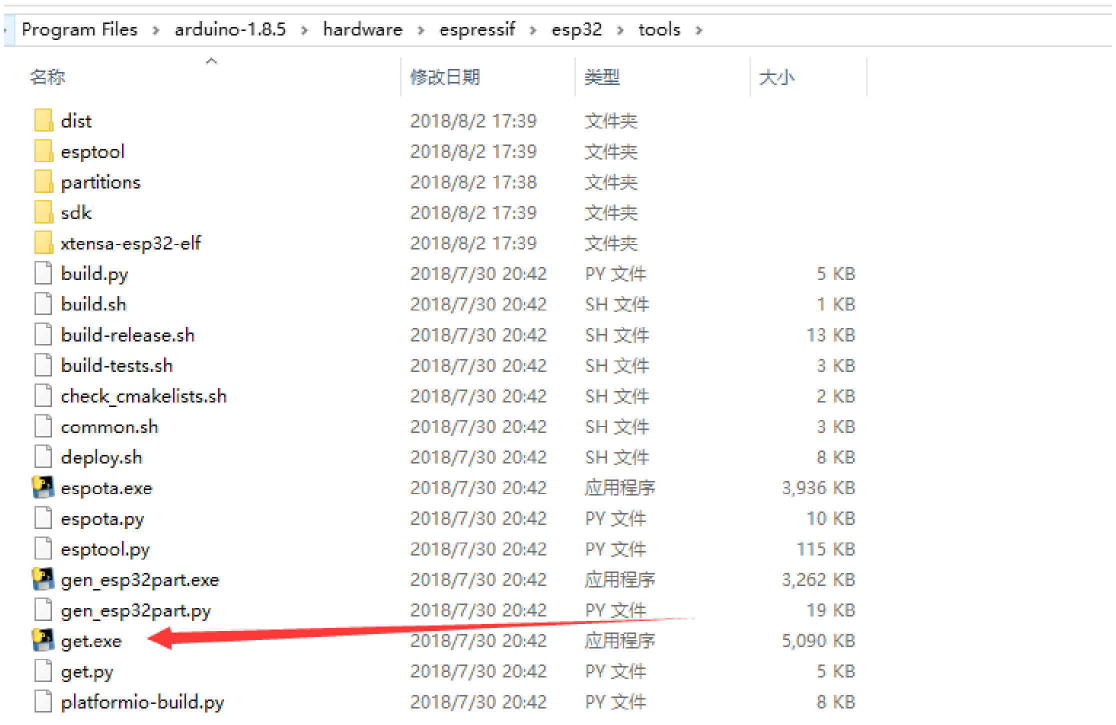
1. If you didn’ t install Arduino IDE before or the IDE version you installed is much old. We recommend you download the newest Arduino IDE from Arduino Website and install.
2. Download Arduino-ESP32 zip from github. unzip to Arduino IDE installation directory-> Hardware->espressif->esp32. If the folders espressif and esp32 doesn’ t exist, you need to create them.



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

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

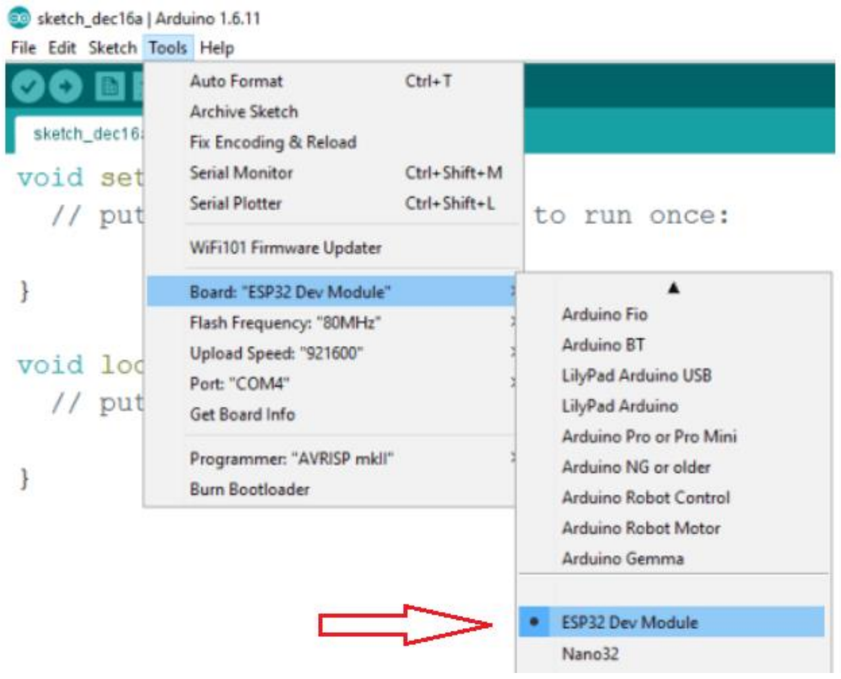
3. Enter tools directory, and run get.exe file as administrator



Program Files › arduino-1.8.5 › hardware › espressif › esp32 › tools ›

名称	修改日期	类型	大小
dist	2018/8/2 17:39	文件夹	
esptool	2018/8/2 17:39	文件夹	
partitions	2018/8/2 17:38	文件夹	
sdk	2018/8/2 17:39	文件夹	
xtensa-esp32-elf	2018/8/2 17:39	文件夹	
build.py	2018/7/30 20:42	PY 文件	5 KB
build.sh	2018/7/30 20:42	SH 文件	1 KB
build-release.sh	2018/7/30 20:42	SH 文件	13 KB
build-tests.sh	2018/7/30 20:42	SH 文件	3 KB
check_cmakelists.sh	2018/7/30 20:42	SH 文件	2 KB
common.sh	2018/7/30 20:42	SH 文件	3 KB
deploy.sh	2018/7/30 20:42	SH 文件	8 KB
espota.exe	2018/7/30 20:42	应用程序	3,936 KB
espota.py	2018/7/30 20:42	PY 文件	10 KB
esptool.py	2018/7/30 20:42	PY 文件	115 KB
gen_esp32part.exe	2018/7/30 20:42	应用程序	3,262 KB
gen_esp32part.py	2018/7/30 20:42	PY 文件	19 KB
get.exe	2018/7/30 20:42	应用程序	5,090 KB
get.py	2018/7/30 20:42	PY 文件	5 KB
platformio-build.py	2018/7/30 20:42	PY 文件	8 KB

4. After installing, ESP32 Dev Module is selectable in IDE



sketch_dec16a | Arduino 1.6.11

File Edit Sketch Tools Help

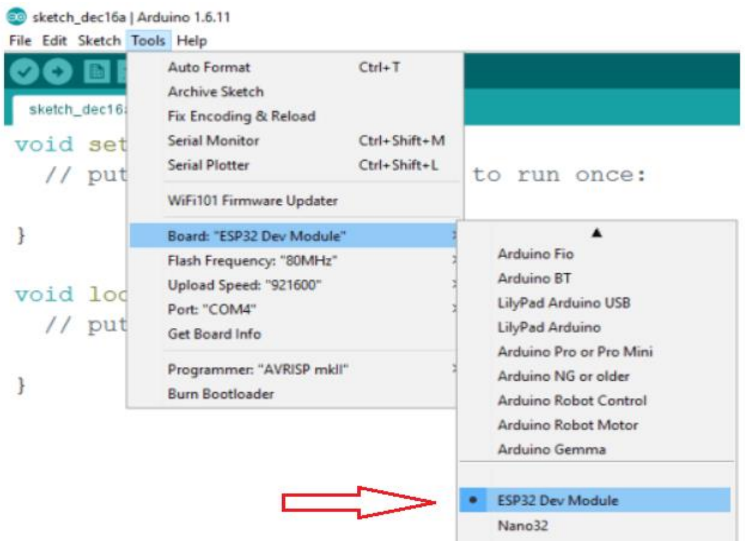
```
void set
  // put                      to run once:

}

void loo
  // put

}
```

Auto Format Ctrl+T
Archive Sketch
Fix Encoding & Reload
Serial Monitor Ctrl+Shift+M
Serial Plotter Ctrl+Shift+L
WiFi101 Firmware Updater
Board: "ESP32 Dev Module"
Flash Frequency: "80MHz"
Upload Speed: "921600"
Port: "COM4"
Get Board Info
Programmer: "AVRISP mkII"
Burn Bootloader

Arduino Fio
Arduino BT
LilyPad Arduino USB
LilyPad Arduino
Arduino Pro or Pro Mini
Arduino NG or older
Arduino Robot Control
Arduino Robot Motor
Arduino Gemma
● ESP32 Dev Module
Nano32

BLUETOOTH DEMO

1. Open Loader_esp32bt folder, double-click Loader_esp32bt.ino file to open Arduino project.
2. Select Tools->Boards->ESP32 Dev Module, and Port



3. Click uploading. Codes are compiled and programmed to ESP32 board
4. Install APP to phone

4.1. Copy the .apk file downloaded to Android phone and install

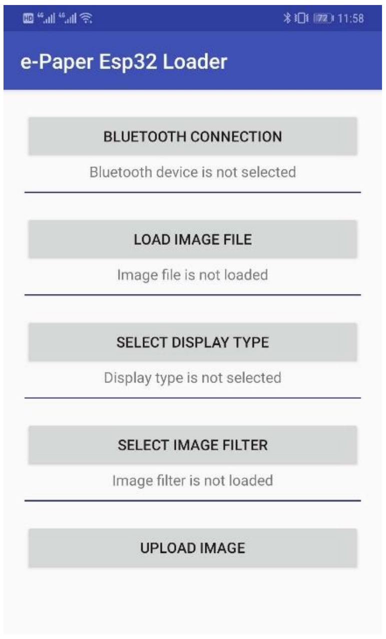
4.2. There are five options in the main page of the app

BLUETOOTH CONNECTION: Connect ESP32 device via Bluetooth

LOAD IMAGE FILE: Load an image from phone

SELECT DISPLAY TYPE: select the e-paper type connected

SELECT IMAGE FILTER: select filter types



UPLOAD IMAGE: uploaded image filtered to e-Paper and refresh

5. Make sure Bluetooth of your phone has been open. Click **BLUETOOTH CONNECTION**-> SCAN to scan Bluetooth device
6. Click ESP32 devices to connect. If your phone is the first time to connect the ESP32 device, it will prompt to pair, confirm it.
7. Click **LOAD IMAGE FILE** to select one image to refresh
8. Click **SELECT DISPLAY TYPE** to select e-Paper type
9. Click **SLECT IMAGE FILTER** to choose filter type

   - **LEVEL:MONO** (Process image, convert it to monochrome image)

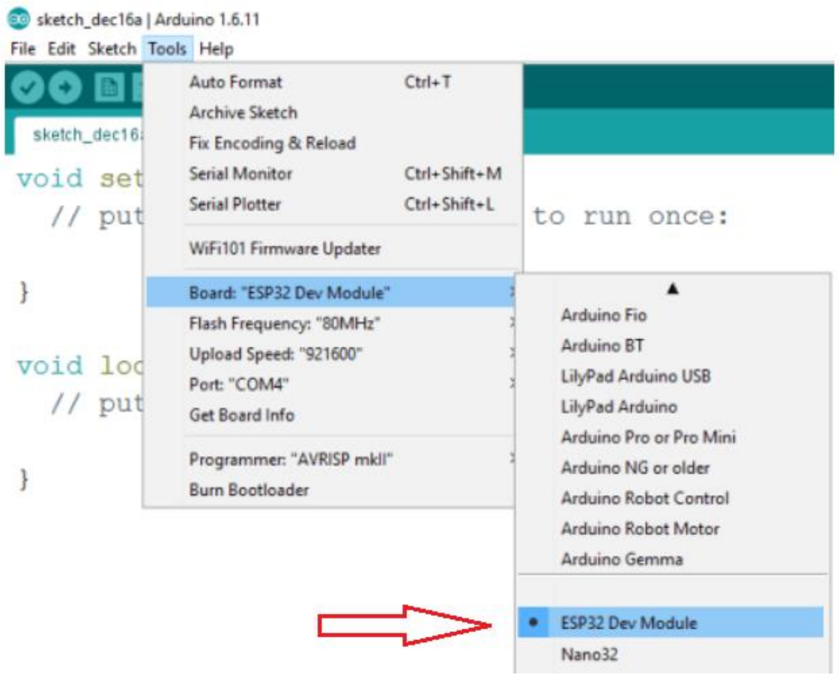
   - **LEVEL:COLOR** (Process image, convert it to three-color image)

   - **DITHERING:MONO** (Process image, convert it to monochrome image)

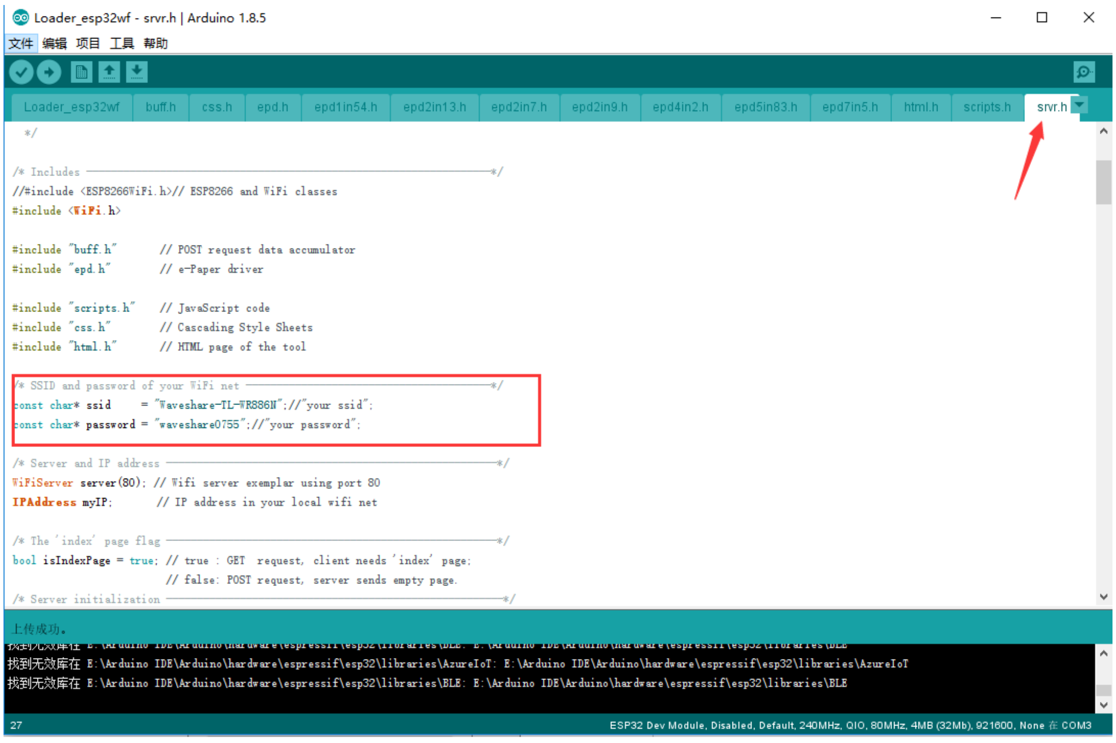
   - **DITHERING:COLOR** (Process image, convert it to three-color image)
10. Click **UPLOAD IMAGE** to upload image to e-Paper

WIFI DEMO

1. Open Loader_esp32wf folder. Double-click Loader_esp32wf.ino to open Arduino project
2. Choose Tools->Boards->ESP32 Dev Module and select Port.



3. Open srvr.h file, modify the ssid and password to the one you use



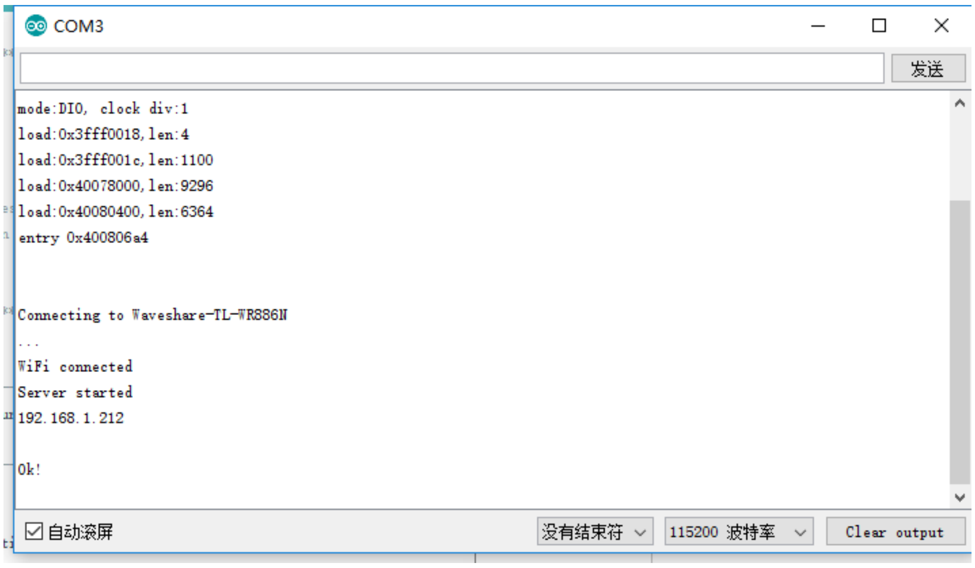
```
 */

/* Includes ------------------------------------------------------------------*/
//#include <ESP8266WiFi.h>// ESP8266 and WiFi classes
#include <WiFi.h>

#include "buff.h"       // POST request data accumulator
#include "epd.h"        // e-Paper driver

#include "scripts.h"    // JavaScript code
#include "css.h"        // Cascading Style Sheets
#include "html.h"       // HTML page of the tool

/* SSID and password of your WiFi net ----------------------------------------*/
const char* ssid     = "Waveshare-TL-WR886N";//"your ssid";
const char* password = "waveshare0755";//"your password";

/* Server and IP address ------------------------------------------------------*/
WiFiServer server(80); // Wifi server exemplar using port 80
IPAddress myIP;        // IP address in your local wifi net

/* The 'index' page flag ------------------------------------------------------*/
bool isIndexPage = true; // true : GET  request, client needs 'index' page;
                         // false: POST request, server sends empty page.
/* Server initialization -------------------------------------------------------*/
```

```
上传成功。
找到无效库在 E:\Arduino IDE\Arduino\hardware\espressif\esp32\libraries\AzureIoT: E:\Arduino IDE\Arduino\hardware\espressif\esp32\libraries\AzureIoT
找到无效库在 E:\Arduino IDE\Arduino\hardware\espressif\esp32\libraries\BLE: E:\Arduino IDE\Arduino\hardware\espressif\esp32\libraries\BLE
```

27 ESP32 Dev Module, Disabled, Default, 240MHz, QIO, 80MHz, 4MB (32Mb), 921600, None 在 COM3

4. Save and Click Upload to program demo to ESP32 driver board

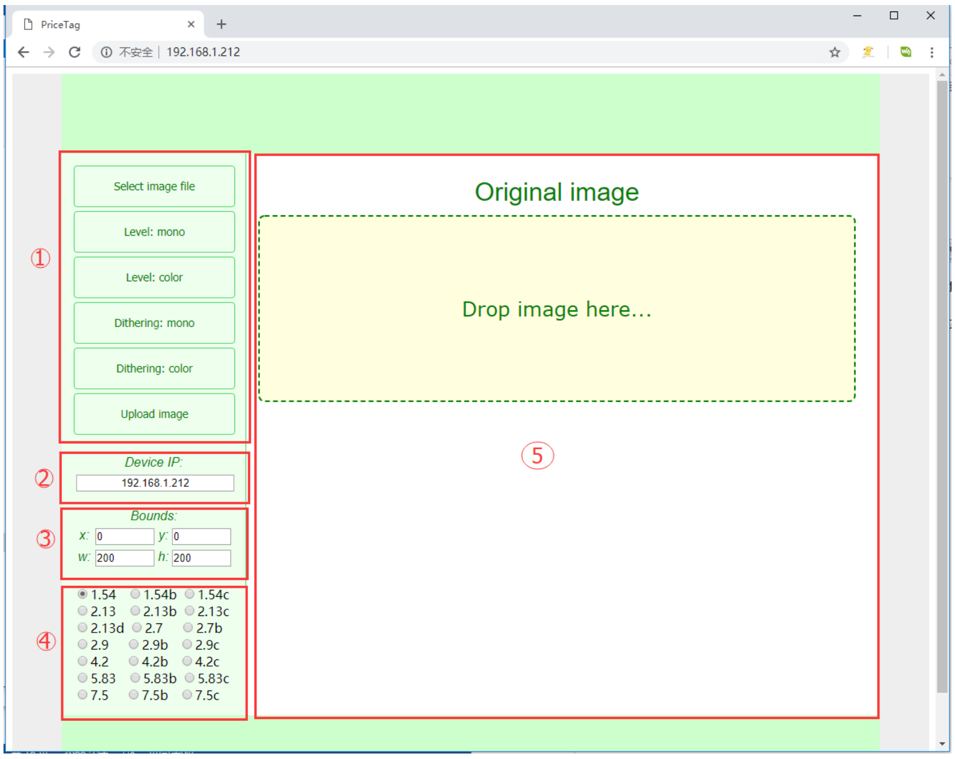
5. Open Serial Monitor, change the baud rate to 115200. Press EN button of ESP32 driver board, ESP32 will restart and print its IP address to serial as below:



```
mode:DIO, clock div:1
load:0x3fff0018,len:4
load:0x3fff001c,len:1100
load:0x40078000,len:9296
load:0x40080400,len:6364
entry 0x400806a4


Connecting to Waveshare-TL-WR886N
...
WiFi connected
Server started
192.168.1.212

Ok!
```

6. Open browser of PC or smart phone (PC and smart phone must be connected to the same LAN as ESP32), input the IP address of ESP32 to enter the control

webpage



Device IP: 192.168.1.212

Bounds:
x: 0 y: 0
w: 200 h: 200

[✓ 1.54] 1.54b 1.54c
2.13 2.13b 2.13c
2.13d 2.7 2.7b
2.9 2.9b 2.9c
4.2 4.2b 4.2c
5.83 5.83b 5.83c
7.5 7.5b 7.5c

6.1. Image handle area:

- Select Image file: Click to select one local picture
- Level:mono (Process image, convert it to monochrome image)
- Level:color (Process image, convert it to three-color image)
- Dithering:mono (Process image, convert it to monochrome image)
- Dithering:color (Process image, convert it to three-color image)
- Update image: Upload image to e-Paper and refresh

6.2. IP address : Display the IP of device connected

6.3. Bounds: w and h are the width and height of e-Paper you selected, which cannot modify. x and y are the begin position of image which you want to display. x and y are adjustable, and you need to choose Process method again if you modify x and y.

6.4. Display selection: Here you can select the e-Paper type connected

6.5. Image display are: The image you load and processed will be display here

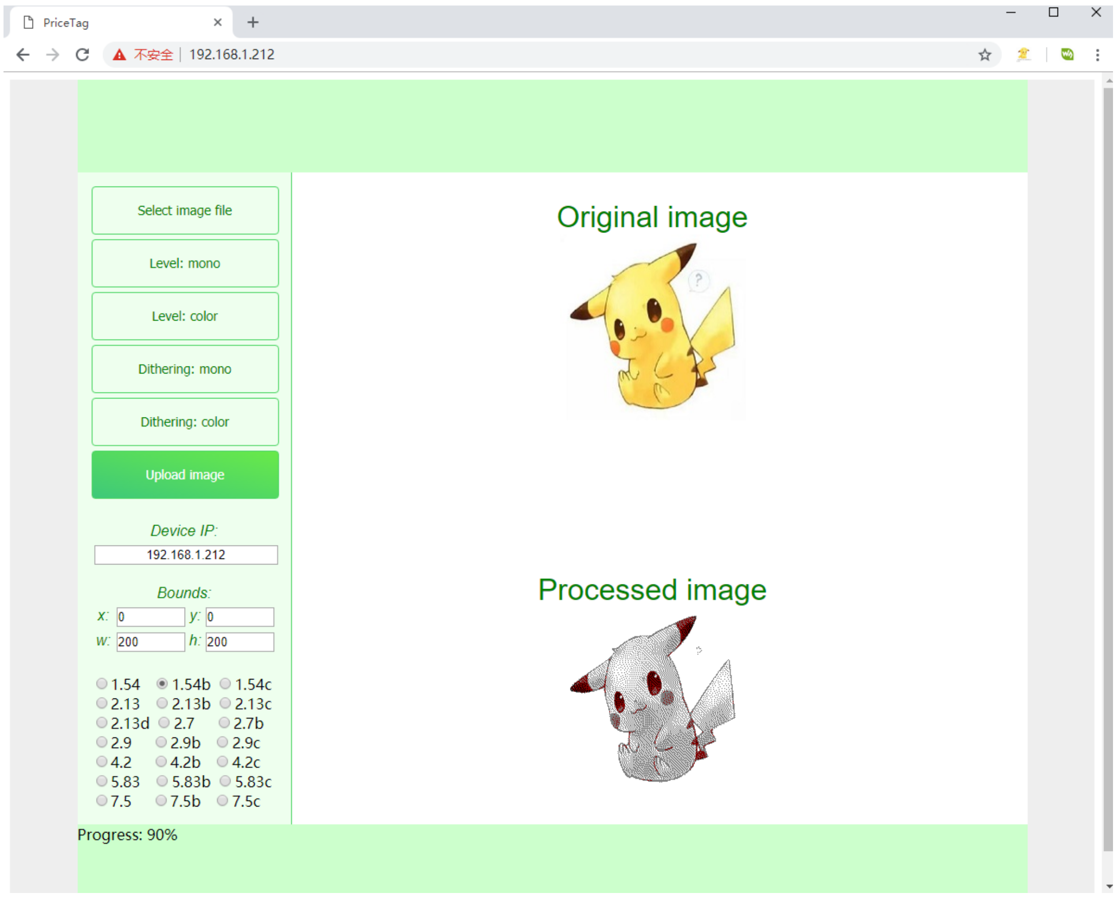
6.6. The uploading progress will be displayed on the bottom

7. ①Click Select image file to load one local image, you can also drag the picture to Original image area directly

8. ④Select display type, for example: 1.54b

9. ①Choose one image filter, for example: Dithering: color

10. ①Click Upload image to upload image to e-Paper and display



Device IP: 192.168.1.212

Bounds: x: 0 y: 0 w: 200 h: 200

○ 1.54	[✓ 1.54b]	○ 1.54c
○ 2.13	○ 2.13b	○ 2.13c
○ 2.13d	○ 2.7	○ 2.7b
○ 2.9	○ 2.9b	○ 2.9c
○ 4.2	○ 4.2b	○ 4.2c
○ 5.83	○ 5.83b	○ 5.83c
○ 7.5	○ 7.5b	○ 7.5c

Progress: 90%

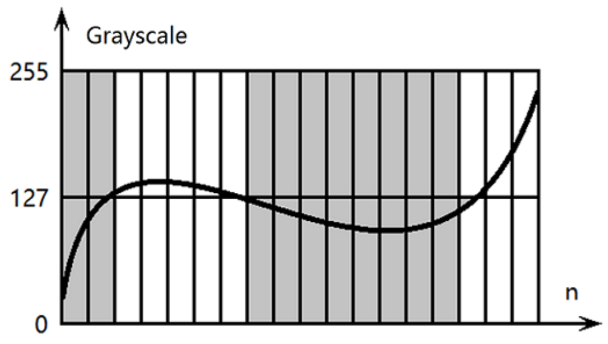
IMAGE PROCESSING

In the demo we provide, we use two kinds of image processing: Level and Dithering

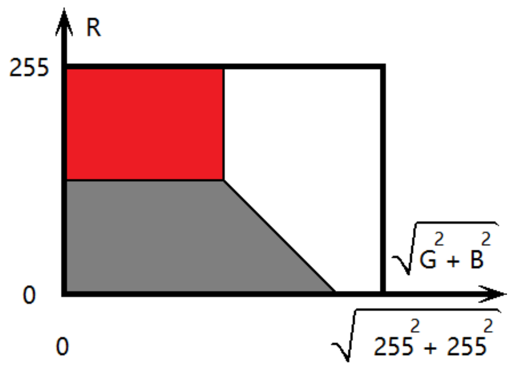
LEVEL

The Level supposes that that image can be divided on a few large regions, in which all of pixels have color ‘close’ to one of available colors: black, white or red and ‘far’ to others available colors. This kind of processing is suitable for 2- or 3-color schematics or texts.

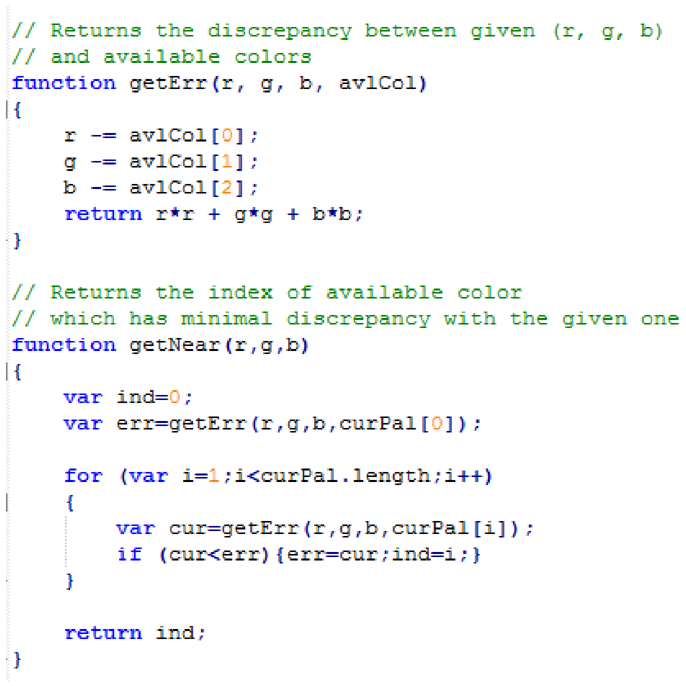
For example, if pixel’ s color of a grayscale image is equals and less than 127, the assigned available color is black, otherwise is white.



In case of colored image, the green and blue channel are combined to green-blue or not-red channel which is ‘orthogonal’ to the red one. On the color diagram it is shown that pixels with high value of red channel and low value of green-blue channel is takes red color, otherwise black or white as in previous example.



Mathematically the definition of color is based on the discrepancy calculation – the sum of squares of channel differences between given and available colors. Pixels take available color which has minimal discrepancy with their color. In the code snipped below the available colors are stored in curPal array:



```
// Returns the discrepancy between given (r, g, b)
// and available colors
function getErr(r, g, b, avlCol)
{
    r -= avlCol[0];
    g -= avlCol[1];
    b -= avlCol[2];
    return r*r + g*g + b*b;
}

// Returns the index of available color
// which has minimal discrepancy with the given one
function getNear(r,g,b)
{
    var ind=0;
    var err=getErr(r,g,b,curPal[0]);

    for (var i=1;i<curPal.length;i++)
    {
        var cur=getErr(r,g,b,curPal[i]);
        if (cur<err){err=cur;ind=i;}
    }

    return ind;
}
```

DITHERING

In case of smooth colored pictures during the Level processing an image lost a lot of thin details expressed by smooth gradient of colors situated close to each other in the color diagram. The most of gradients cover large areas of pictures taken by camera, thus it is possible to express some shades by mixing closest available colors on those areas.

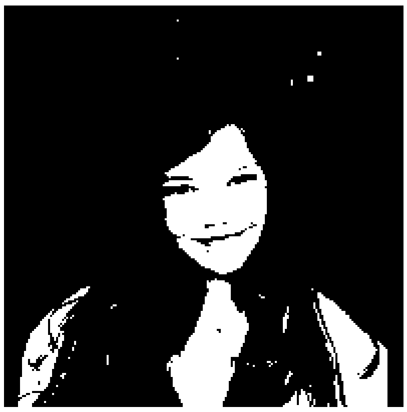
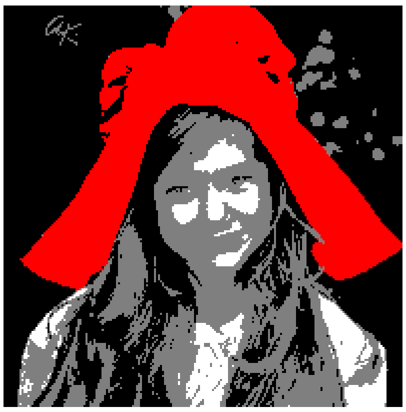
Eyes feel the average color of pixels in a small area. It means there are more seeming colors, but in other hand the picture seems noisy we well or as if it has low resolution. Good algorithms of color the mixing can prevent pixilation (creating clearly seen small grain-like areas in a picture). One of them is the Dithering.

The application uses the Floyd-Steinberg Dithering - most famous 2D error diffusion formula (was published by Robert Floyd and Louis Steinberg in 1976). It diffuses errors according the pattern:

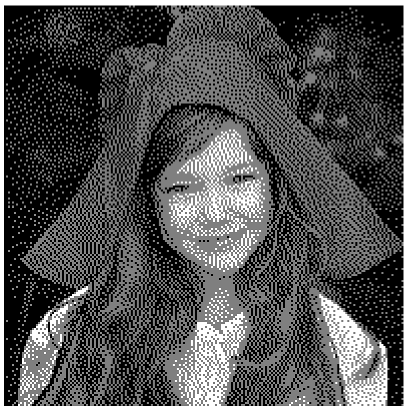
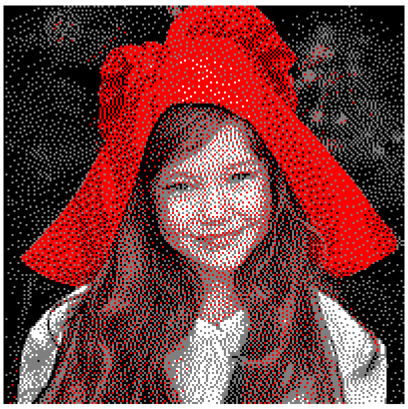
```
      X    7
3     5    1

   (1/16)
```

Here X - is an error (scalar/vector difference between original and available grayscale/colored value of pixel). This error is distributed between right, right-bottom, bottom and left-bottom pixels, is just added to their values with factors 7/16, 1/16, 5/16 and 3/16 respectively. Thus, the average original color stays within this small group of pixels. The algorithm doesn' t change left, left-top, top and right-top pixels because they are already corrected at previous iteration of the algorithm.

Original Image

“Level: mono” 和 “Level: color”.

“Dithering: mono” and “Dithering: color”

DATA TRANSMISSION PROTOCOL

The offered protocol allows to divide image data and send it part by part to the esp32 module. If you don’ t know well about Wi-Fi functionality or can’ t use it by some reason but need to develop a tool for the file transmission via HTML page and Wi-Fi net in haste, then you can use the solution described in this document. It is based on POST request data transmission. Consider the data transmission mechanism of this solution, you can modify it or add your own functions/commands if you need.

COMMANDS

The protocol of communication between image dividing code snippet on the client-side and data receiving code at the server-side includes 4 commands:

- EPDn – the initialization of n-type display (n is a character in range ‘a’ ..’ l’);
- LOAD – the image data loading (black and/or red channel);
- NEXT – the switching from the black channel to the red one;
- DONE – the refreshing of display and turning into the deep sleep mode.

INITIALIZATION ALGORITHM

As it is shown in the diagram (figure 12), the event handler of “Upload image” button creates an object for sending commands and listening responses from server, sends EPDn-command. The server-side application receives the EPDn-command, initializes the e-Paper display and open monochrome or black channel for writing. The memory writing commands (EPDn and NEXT) for white-black display is

EPD_SendCommand(0x24);//WRITE_RAM

and for white-black-red display is (black and red channels correspondently):

EPD_SendCommand(0x10); //DATA_START_TRANSMISSION_1,

EPD_SendCommand(0x13); //DATA_START_TRANSMISSION_2,

but there are exceptions:

- White-black displays 2.7 and 4.2 use the red channel (code 0x13) instead of monochrome channel (code 0x24);
- The display 7.5 loads the red and black data simultaneously.

Thus, when you copy code snippets from initialization functions pay attention how they write image data into display’ s memory!

DATA PIXEL FORMAT

When your browser opens an image file, the pixel format is 24 bpp by default. The image processing reduces it to 1 bpp (white-black display):

0 – white;

1 – black;

or 2 bpp (white-black-red display):

0 – white;

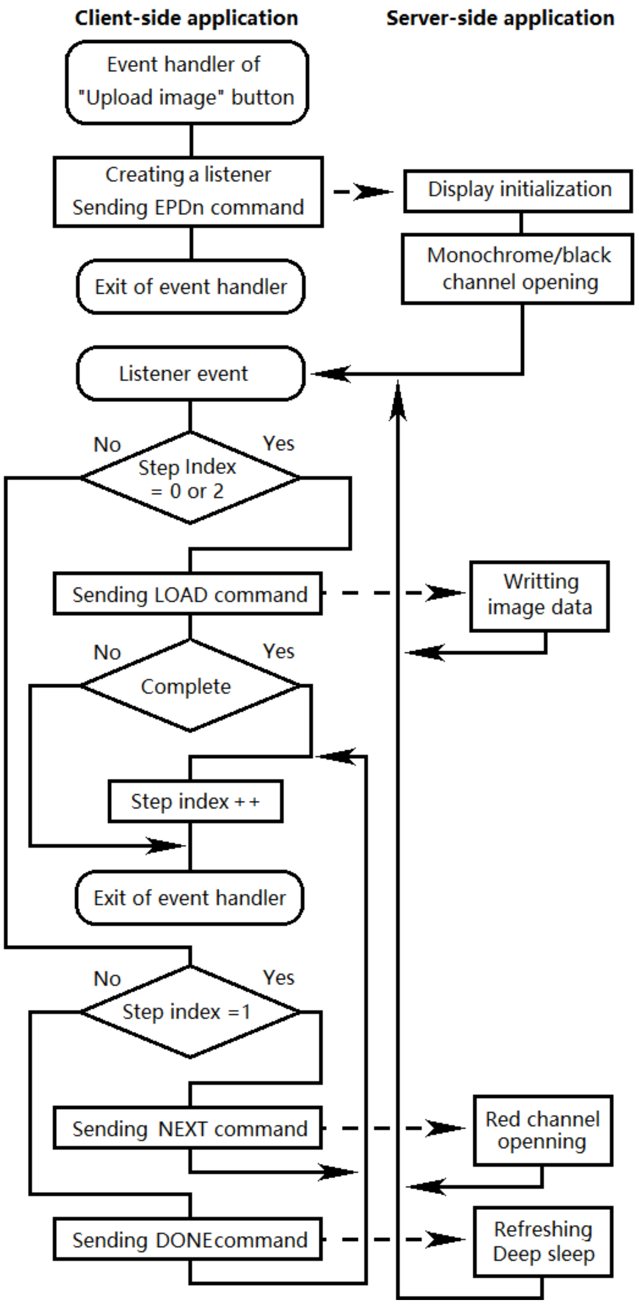
1 – black;

2 – gray (1.54inch b-type only);

3 – red.

Before the xhReq object sends a POST-request to the server, the client-side application packs image data into bytes or words. The bits order in these bytes/words is native display memory order or is suitable for fast conversion into the native order.

The table 2 shows these changes from the start of conversion to the writing of data into display’ s memory:



Important: before the first writing of the image data into a display, one of commands: WRITE_RAM, DATA_START_TRANSMISSION_1 or DATA_START_TRANSMISSION_2 is required, but 2.13-display can load data line by line only. It means before the image data writing of each line, the command WRITE_RAM must be executed.

Display type (channel)	Transmission format (p – pixel, b - bits)	Native format
1.54 (black) 1.54b (red) 2.13 (black) 2.13b (both) 2.7b (both) 2.9 (black) 2.9b (both) 4.2 (black) 4.2b (both)	p:01234567 - b:76543210 0 – black or red; 1 – white.	p:01234567 - b:76543210 0 – black or red; 1 – white. ***Comment***: *there are no any changes*
2.7	p:01234567 - b:76543210 0 – black; 1 – white.	p:01234567 - b:76543210 0 – white; 1 – black (). ***Comment:*** *bitwise inversion*

7.5	p:01234567 - b:76543210
	0 – black or red;
	1 – white.
		p:0 - b:7,6,5,4 (7 is high, 4 is low);
		p:1 – b:3,2,1,0;
		p:2 – b:15,14,13,12;
		p:3 – b:11,10,9,8;

		0000 (0) – black; 0011 (3) – white.

1.54b (black)	p:0 – b:1,0;
	
	p:1 – b:3,2;
	
	p:2 – b:5,4;
	
	p:3 – b:7,6;
	
	00 (0) – black;
	
	01 (1) – white;
	
	10 (2) – gray;
	
	11 (3) – red, is read as 10 (2).
		p:0 – b:7,6;
		
		p:1 – b:5,4;
		
		p:2 – b:3,2;
		
		p:3 – b:1,0;
		
		00 (0) – black;
		
		01 (1) – gray;
		
		11 (3) – white;

7.5b	p:0 – b:1,0;
	
	p:1 – b:3,2;
	
	p:2 – b:5,4;
	
	p:3 – b:7,6;
	
	00 (0) – black;
	
	01 (1) – white;
	
	10 (2) – gray;
	
	11 (3) – red.
		p:0 - b:7,6,5,4 (7 is high, 4 is low);
		
		p:1 – b:3,2,1,0;
		
		p:2 – b:15,14,13,12;
		
		p:3 – b:11,10,9,8;
		
		0000 (0) – black;
		
		0011 (3) – white;
		
		0100 (4) – red.