

1.8inch LCD Module

This is a general LCD display Module, TFT screen, 1.8inch diagonal, 128×160 resolution, with embedded controller, communicating via SPI interface.

Feature

- SPI interface requires minimum GPIO for controlling
- Comes with development resources and manual
- The basic functions have been encapsulated on the demo, which can realize picture rotation, draw dots, lines, circles, rectangles, display English characters and Chinese characters, and display pictures.

Specifications

- Operating voltage: 3.3V/5V (When using 5V power supply, the logic voltage is 5V, when using 3.3V power supply, the logic voltage is 3.3V)
- Interface: SPI
- LCD type: TFT
- Controller: ST7735S
- Resolution: 128*160 (Pixel)
- Display size: 35.04(W) * 28.03(H)(mm)
- Pixel size: 0.219(W)*0.219(H)(MM)
- Dimension: 56.5 * 34(mm)

Interface

SYMBOL	Description
VCC	Power (3.3V input)
GND	Ground
DIN	SPI data input
CLK	SPI clock input
CS	Chip selection, low active
DC	Data/Command selection (high for data, low for command)
RST	Reset, low active
BL	Backlight

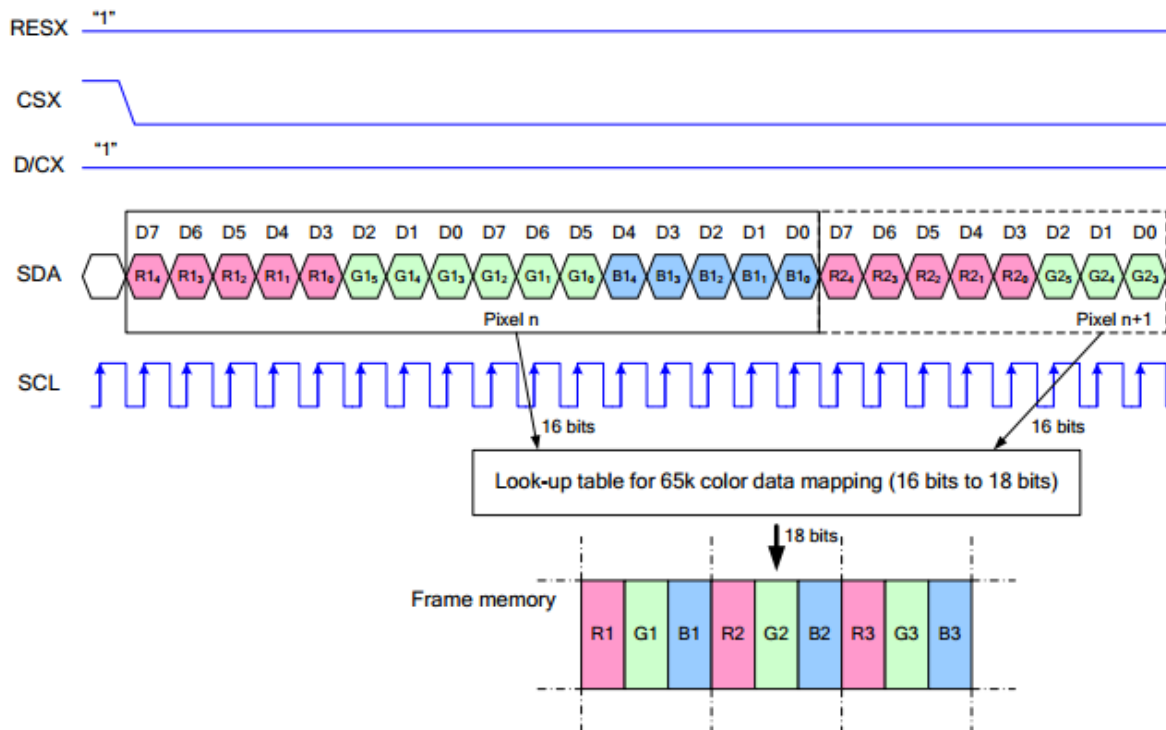
LCD and the controller

The ST7735S is a 132*162 pixel LCD controller, but the pixel of the 1.8inch LCD HAT is 128*160. So we have made some processing on the display: the horizontal direction starts from the second pixel, so that to guarantee the location of RAM in the LCD is consistent with the actual location at the same time.

This LCD accepts 8-bits/9-bits/16-bits/18-bits parallel interface, that are RGB444, RGB565, RGB666. The color format used in demo codes is RGB565.

This LCD use 4-lines SPI interface for reducing GPIO and fast speed.LCD

Communication protocol



Note: It is not like the tradition SPI protocol, it only uses MOSI to send data from master to slave for LCD display. For details please refer to Datasheet Page 105.

RESX: Reset, should be pull-down when power on, set to 1 other time.

CSX: Slave chip select. The chip is enabled only CS is set Low

D/CX: Data/Command selection; DC=0, write command; DC=1, write data

SDA: Data transmitted. (RGB data)

SCL: SPI clock

The SPI communication protocol of the data transmission uses control bits: clock phase (CPHA) and clock polarity (CPOL):

CPOL defines the level while the synchronization clock is idle. If CPOL=0, then it is LOW.

CPHA defines at which clock's tick the data transmission starts. CPHL=0 – at the first one, otherwise at the second one

This combination of two bits provides 4 modes of SPI data transmission. The commonly used is SPI0 mode, i.e. GPHL=0 and CPOL=0.

According to the figure above, data transmitting begins at the first falling edge, 8bit data are transmitted at one clock cycle. It is SPI0. MSB.

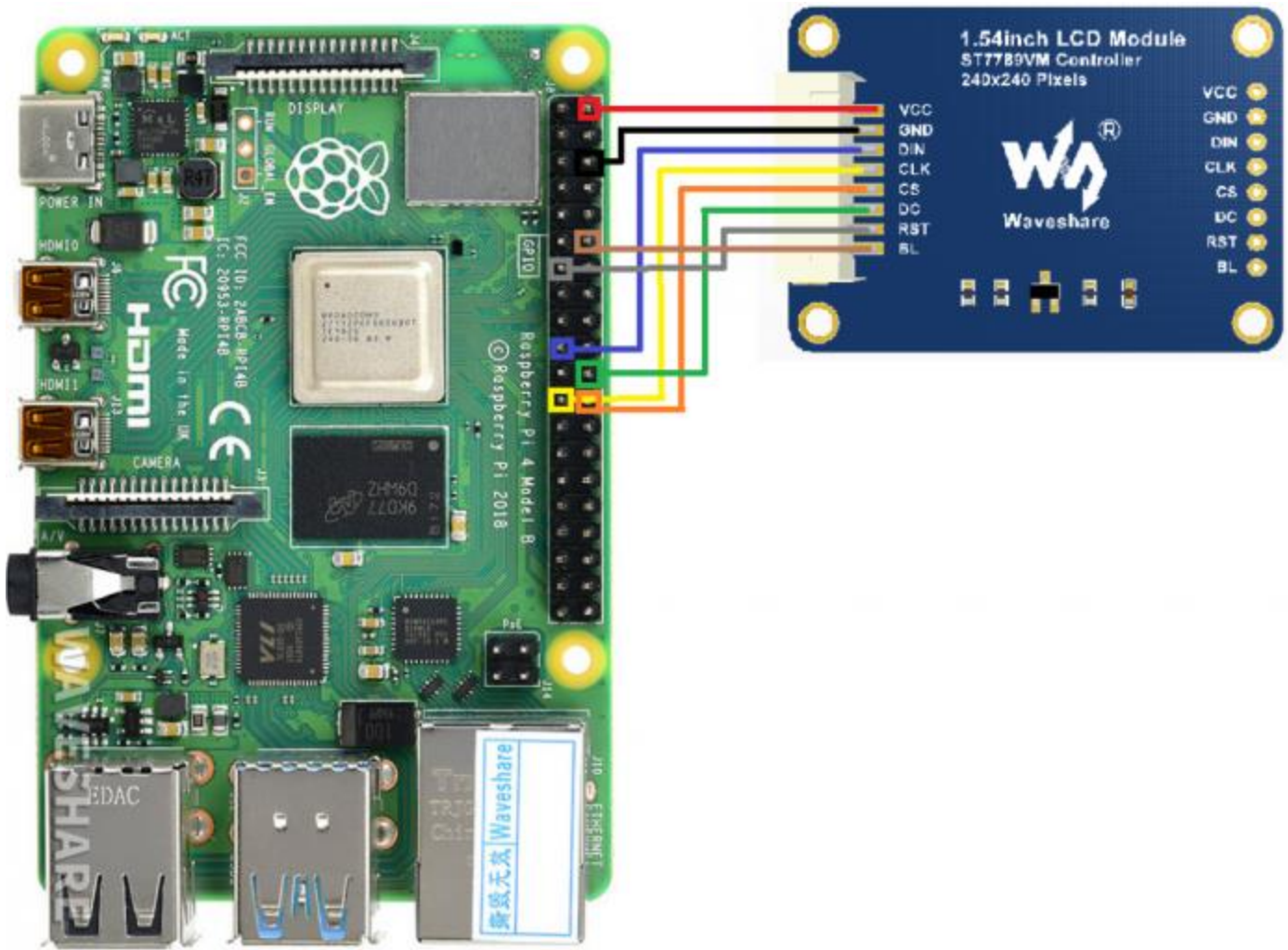
Hardware connection

Please connect the LCD to your Raspberry Pi by the 8Pin cable according to the table below

Connect to Raspberry Pi

LCD	Raspberry Pi	
	BCM2835	Board
VCC	5V	5V
GND	GND	GND
DIN	MOSI	19
CLK	SCLK	23
CS	CE0	24
DC	25	22
RST	27	13
BL	18	12

The color of actual cable may be different with the figure here, please connect them according to the pins instead of color.



Enable SPI interface

- Open terminal, use command to enter the configuration page

```
sudo raspi-config
```

```
Choose Interfacing Options -> SPI -> Yes to enable SPI interface
```

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

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

Would you like the SPI interface to be enabled?

<Yes>

<No>

Reboot Raspberry Pi :

```
sudo reboot
```

Please make sure that SPI interface was not used by other devices

Install Libraries

- Install BCM2835 libraries

```
wget http://www.airspayce.com/mikem/bcm2835/bcm2835-1.68.tar.gz
tar zxvf bcm2835-1.68.tar.gz
cd bcm2835-1.68/
sudo ./configure
sudo make
sudo make check
sudo make install
#For more details, please refer to http://www.airspayce.com/mikem/bcm2835/
```

- Install wiringPi libraries

```
sudo apt-get install wiringpi
```

For the version of the Raspberry Pi system after May 2019 (the OS version earlier than this date doesn't need to be executed), an upgrade may be required :

```
wget https://project-downloads.drogon.net/wiringpi-latest.deb
sudo dpkg -i wiringpi-latest.deb
gpio -v
#You will get 2.52 information if you install it correctly
```

- 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 p7zip-full
sudo wget https://www.waveshare.com/w/upload/a/a8/LCD_Module_RPI_code.7z
7z x LCD_Module_RPI_code.7z -O./LCD_Module_code
cd LCD_Module_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
```

This examples are made for multi-display, you can input the type of the LCD when using.

```
sudo ./main <<type of LCD>>
```

Use the command according to LCD: :

```
sudo ./main 0.96
sudo ./main 1.14
sudo ./main 1.28
sudo ./main 1.3
sudo ./main 1.54
sudo ./main 1.8
sudo ./main 2
```

python

- Enter the python directory and run ls -al

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

```
pi@eng33:~/LCD_Module_code/RaspberryPi/python/example $ ls -l
total 24
-rw-r--r-- 1 pi pi 2830 Jun 16 17:59 0inch96_LCD_test.py
-rw-r--r-- 1 pi pi 2459 Jun 16 18:34 1inch14_LCD_test.py
-rw-r--r-- 1 pi pi 2701 Jun 16 18:33 1inch28_LCD_test.py
-rw-r--r-- 1 pi pi 2665 Jun 16 17:58 1inch54_LCD_test.py
-rw-r--r-- 1 pi pi 2678 Jun 16 18:34 1inch8_LCD_test.py
-rw-r--r-- 1 pi pi 2660 Jun 16 18:39 2inch_LCD_test.py
```

You can check all the files which are listed in type:

0inch96_LCD_test.py	0.96inch LCD example
1inch14_LCD_test.py	1.14inch LCD example
1inch28_LCD_test.py	1.28inch LCD example
1inch3_LCD_test.py	1.3inch LCD example
1inch54_LCD_test.py	1.54inchLCD example
1inch8_LCD_test.py	1.8inch LCD example
2inch_LCD_test.py	2inch LCD example

- Run the example

```
# python2
sudo python 0inch96_LCD_test.py
sudo python 1inch14_LCD_test.py
sudo python 1inch28_LCD_test.py
sudo python 1inch3_LCD_test.py
sudo python 1inch54_LCD_test.py
sudo python 1inch8_LCD_test.py
sudo python 2inch_LCD_test.py

# python3
sudo python3 0inch96_LCD_test.py
sudo python3 1inch14_LCD_test.py
sudo python3 1inch28_LCD_test.py
sudo python3 1inch3_LCD_test.py
sudo python3 1inch54_LCD_test.py
sudo python3 1inch8_LCD_test.py
sudo python3 2inch_LCD_test.py
```

FBCP Transplant

The Framebuffer uses a memory area to store the display content, and changes the data in the memory to change the display content.

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

Compile and Run

```
sudo apt-get install cmake -y
cd ~
wget https://www.waveshare.com/w/upload/f/f9/Waveshare_fbcv.7z
sudo apt-get install p7zip-full
7z x Waveshare_fbcv.7z
cd waveshare_fbcv
mkdir build
cd build
cmake [options] ..
make -j
sudo ./fbcv
```

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

#0.96inch LCD Module

```
cmake -DSPI_BUS_CLOCK_DIVISOR=20 -DWAVESHARE_0INCH96_LCD=ON -DBACKLIGHT_CONTROL=ON -
DSTATISTICS=0 ..
```

#1.14inch LCD Module

```
cmake -DSPI_BUS_CLOCK_DIVISOR=20 -DWAVESHARE_1INCH14_LCD=ON -DBACKLIGHT_CONTROL=ON -
DSTATISTICS=0 ..
```

#1.3inch LCD Module

```
cmake -DSPI_BUS_CLOCK_DIVISOR=20 -DWAVESHARE_1INCH3_LCD=ON -DBACKLIGHT_CONTROL=ON -
DSTATISTICS=0 ..
```

#1.54inch LCD Module

```
cmake -DSPI_BUS_CLOCK_DIVISOR=20 -DWAVESHARE_1INCH54_LCD=ON -DBACKLIGHT_CONTROL=ON -
DSTATISTICS=0 ..
```

#1.8inch LCD Module

```
cmake -DSPI_BUS_CLOCK_DIVISOR=20 -DWAVESHARE_1INCH8_LCD=ON -DBACKLIGHT_CONTROL=ON -
DSTATISTICS=0 ..
```

#2inch LCD Module

```
cmake -DSPI_BUS_CLOCK_DIVISOR=20 -DWAVESHARE_2INCH_LCD=ON -DBACKLIGHT_CONTROL=ON -
DSTATISTICS=0 ..
```

#2.4inch LCD Module

```
cmake -DSPI_BUS_CLOCK_DIVISOR=20 -DWAVESHARE_2INCH4_LCD=ON -DBACKLIGHT_CONTROL=ON -
DSTATISTICS=0 ..
```

Auto-start when Power on

```
sudo cp ~/Fbcv-ili9341/build/fbcv-ili9341 /usr/local/bin/fbcv
sudo nano /etc/rc.local
```

And then add `fbcp&` before `exit 0`, as the picture below.

```
# Print the IP address
_IP=$(hostname -I) || true
if [ "$_IP" ]; then
    printf "My IP address is %s\n" "$_IP"
fi

fbcp&
exit 0
```

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=1
hdmi_mode=87
display_rotate=0
```

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

```
#2.4inchinch LCD Module & 2inchinch LCD Module
hdmi_cvt=640 480 60 1 0 0 0

#1.8inch LCD Module
hdmi_cvt=400 300 60 1 0 0 0

#1.3inch LCD Module & 1.54inch LCD Module
hdmi_cvt=300 300 60 1 0 0 0

#1.14inch LCD Module
hdmi_cvt=300 170 60 1 0 0 0

#0.96inch LCD Module
hdmi_cvt=300 150 60 1 0 0 0
```

[Note] If you are using Raspberry Pi 4B, you need to comment out the following lines on the `[pi4]` part. The modification is as below:

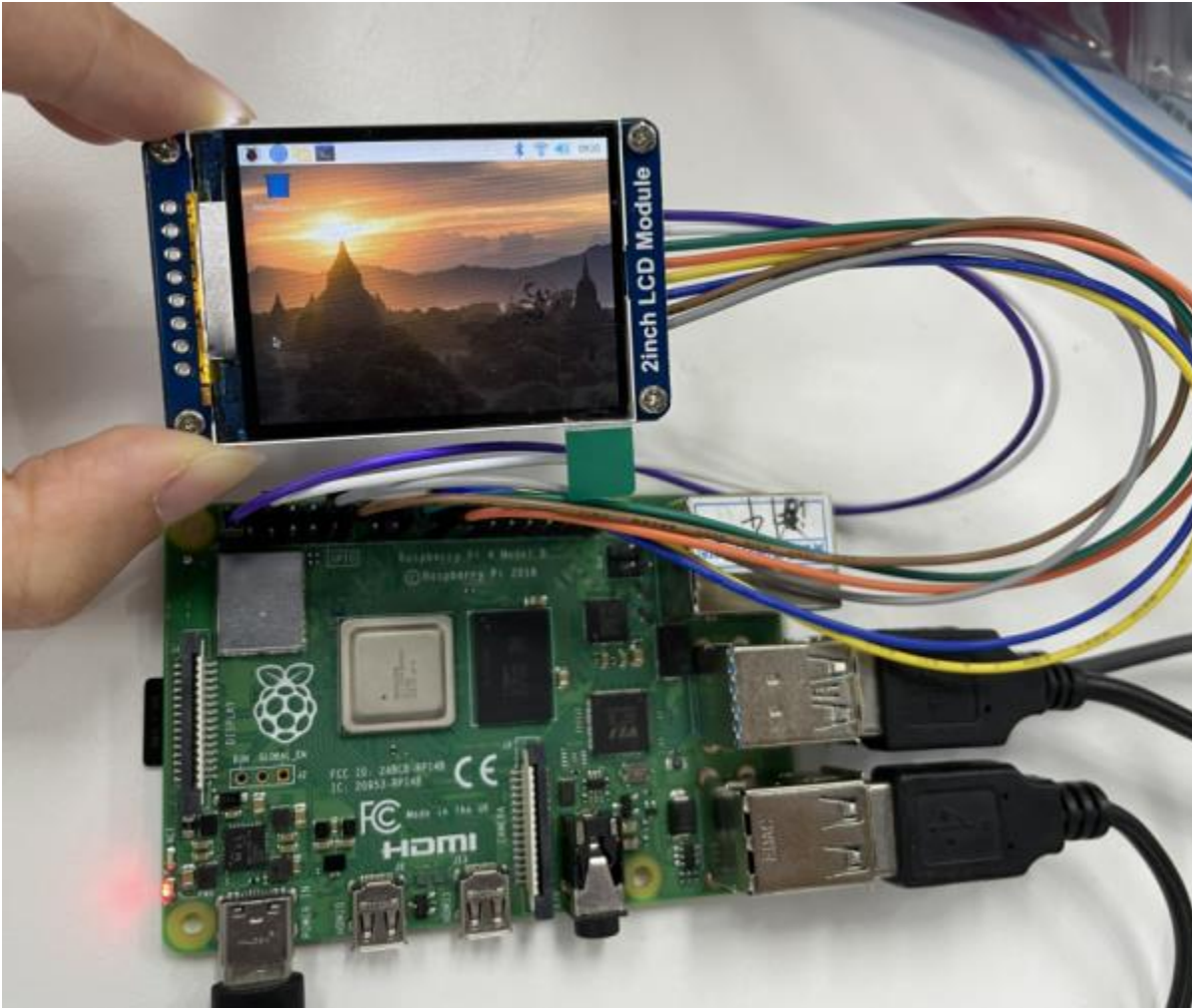
```
[pi4]
# Enable DRM VC4 V3D driver on top of the dispmanx display stack
```

```
#dtoverlay=vc4-fkms-v3d
#max_framebuffers=2
```

And then reboot the system

```
sudo reboot
```

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



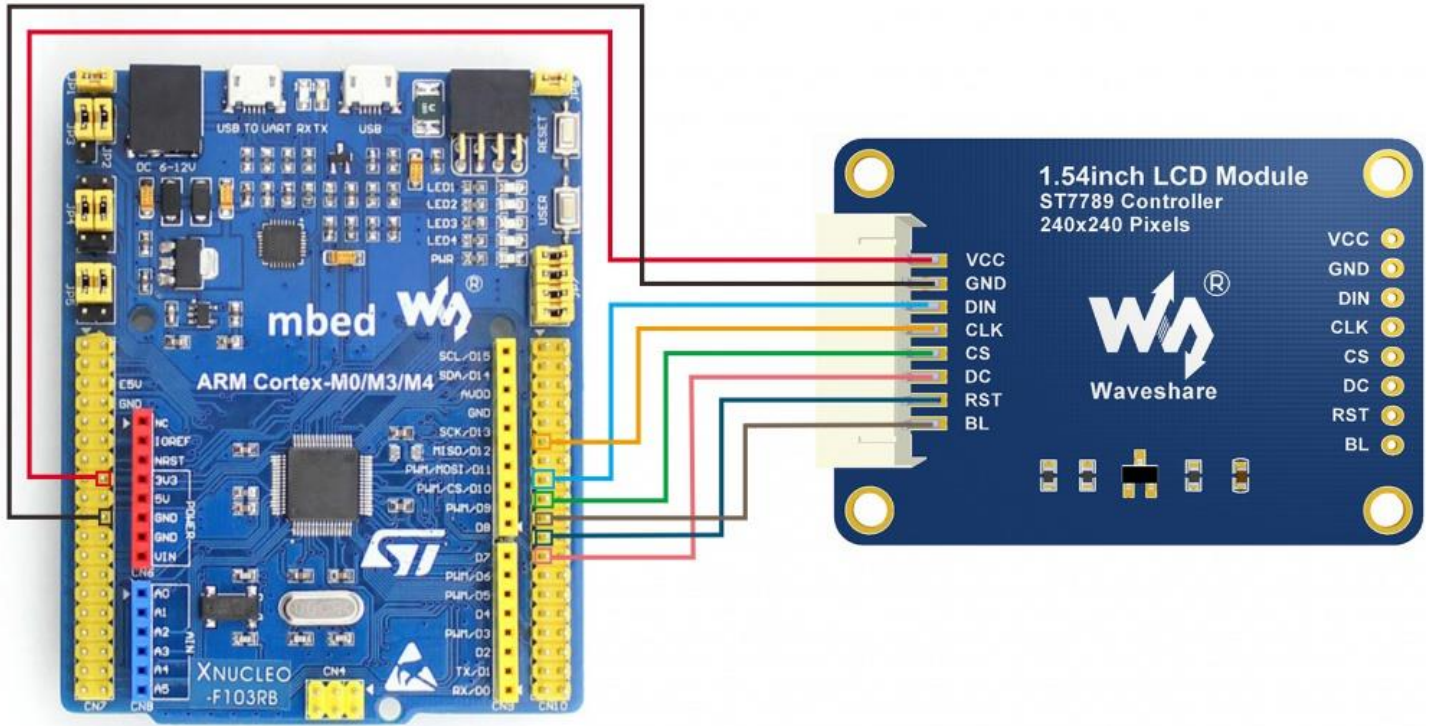
Hardware Connection

The examples are based on STM32F103RBT6 as well as the connection table. If you want to use other MCU, you need to port the project and change the connection according to the actual hardware.

Connect to STM32F103RBT6

LCD	STM32
VCC	3.3V/5V
GND	GND
DIN	PA7

CLK	PA5
CS	PB6
DC	PA8
RST	PA9
BL	PC7



About the examples

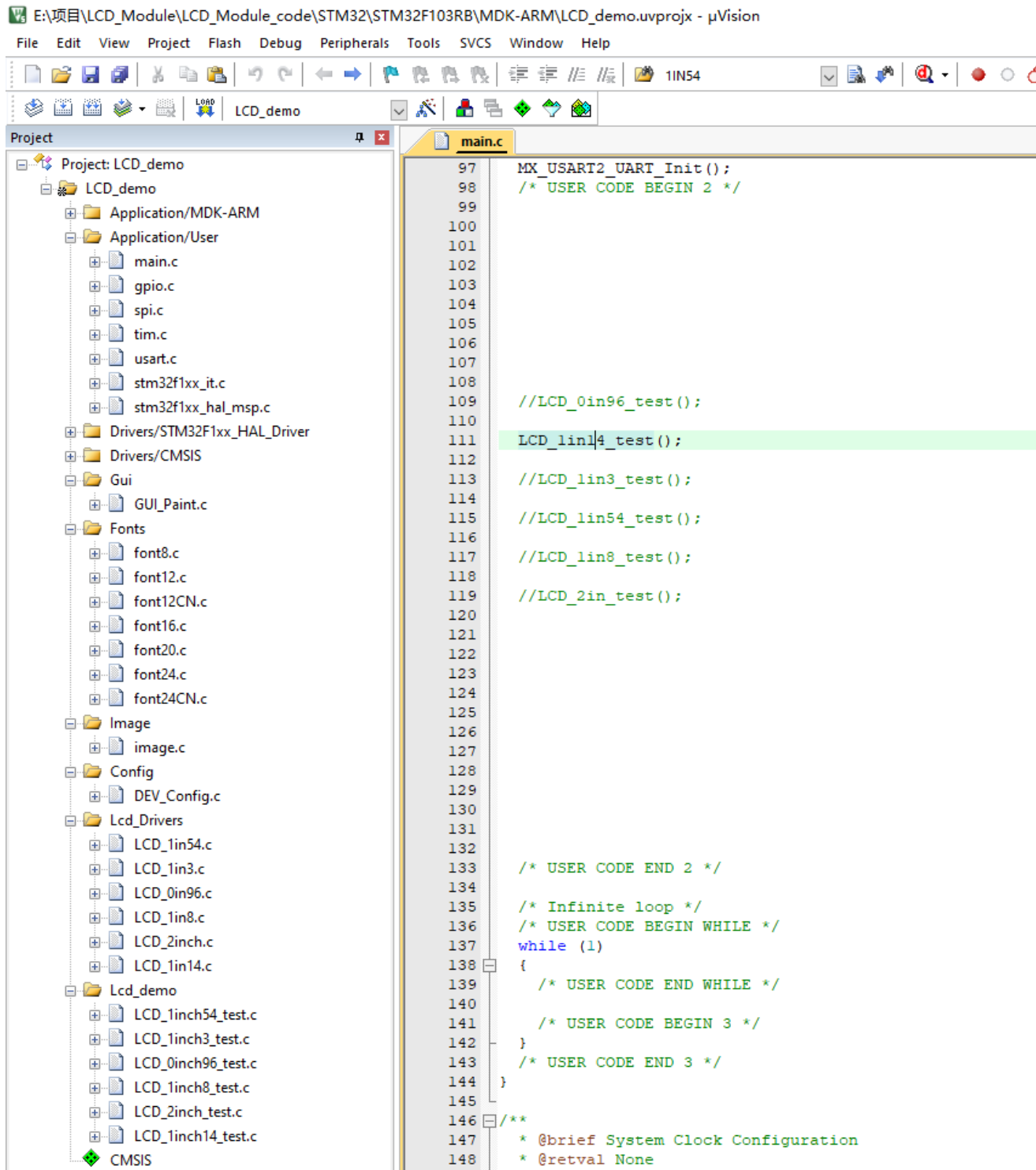
The examples use HAL libraries. Download demo codes, unzip, and find the STM32 projects. Open LCD_demo.uvprojx which is located in STM32\STM32F103RBT6\MDK-ARM directory by Keil project

> LCD_Module_code > STM32 > STM32F103RB

搜索"STM32F103RB"

名称	修改日期	类型	大小
Drivers	2020/6/17 17:59	文件夹	
Inc	2020/6/17 17:59	文件夹	
MDK-ARM	2020/6/18 16:37	文件夹	
Src	2020/6/17 17:59	文件夹	
User	2020/6/17 17:59	文件夹	
.mxproject	2020/6/8 17:22	MXPROJECT 文件	7 KB
LCD_demo.ioc	2020/6/8 17:21	STM32CubeMX	5 KB

Open main.c file, you can configure the types for actual displays, recompile the project and download it to your board.



- LCD_0in96_test() 0.96inch LCD example
- LCD_1in14_test() 1.14inch LCD example
- LCD_1in28_test() 1.28inch LCD example
- LCD_1in3_test() 1.3inch LCD example
- LCD_1in54_test() 1.54inch LCD example
- LCD_1in8_test() 1.8inch LCD example
- LCD_2in_test() 2inchLCDexample

Arduino

- Download examples from wiki. Unzip it. The path of Arduino examples is ~/Arduino UNO/...
- Copy the folders in Arduino directory to 【Installation directory】 /libraries/ (Generally the installation directory is C:\Program Files (x86)\Arduino\libraries)
- Open Arduino IDE software, and click File -> Examples to check if LCD_2inch codes are there.
- The development board used is Arduino UNO.

Hardware connection

2inch LCD	UNO PLUS
VCC	5V
GND	GND
DIN	D11
CLK	D12
CS	D10
DC	D7
RST	D8
BL	D9

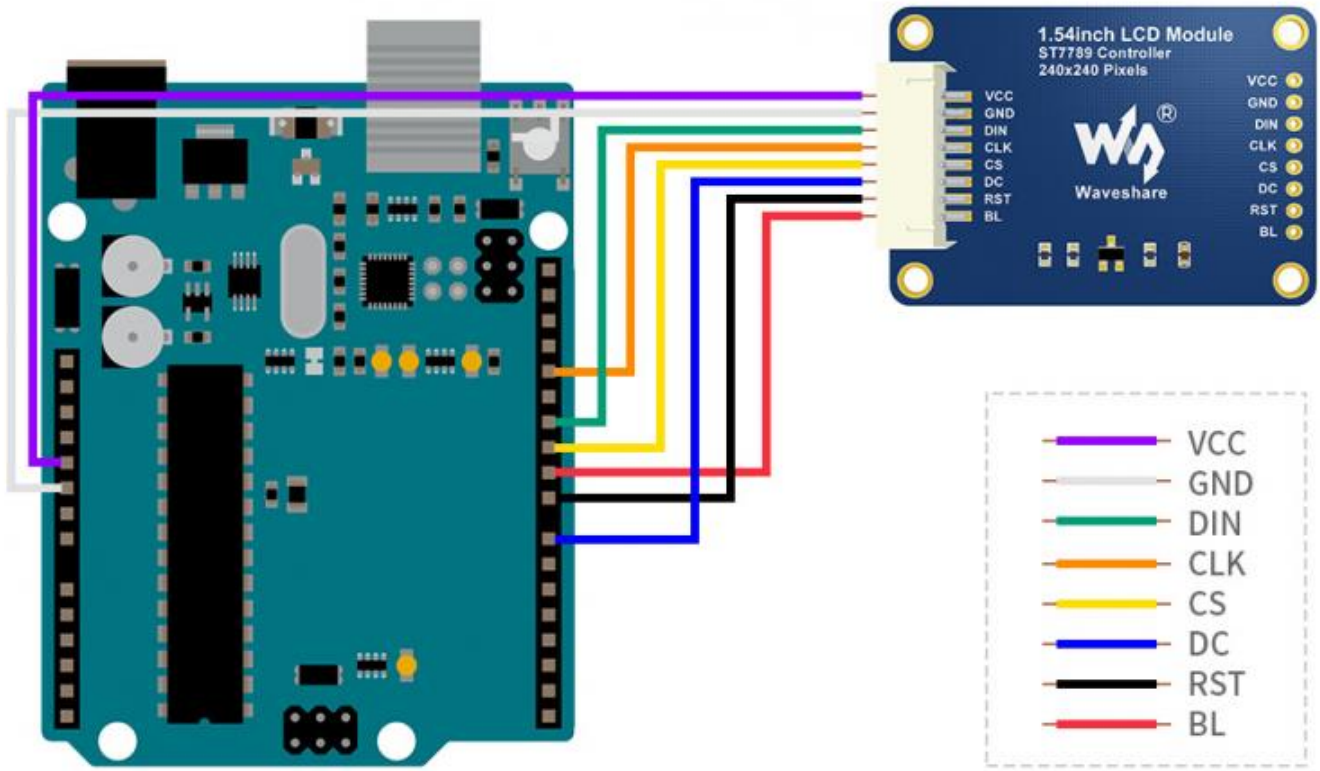
Expected result

1. The display is cleaned to white
2. Display numbers and strings
3. Draw a rectangle
4. Draw a line
5. Draw five circles
6. Display a 70x70 image
7. The examples are tested in Arduino UNO, if you want to use other versions of the Arduino, you need to change the connection according to the actual boards.

8. Hardware Connection

Arduino UNO连接引脚对应关系	
LCD	UNO
VCC	5V/3.3V
GND	GND
DIN	D11

CLK	D13
CS	D10
DC	D7
RST	D8
BL	D9



Run the example

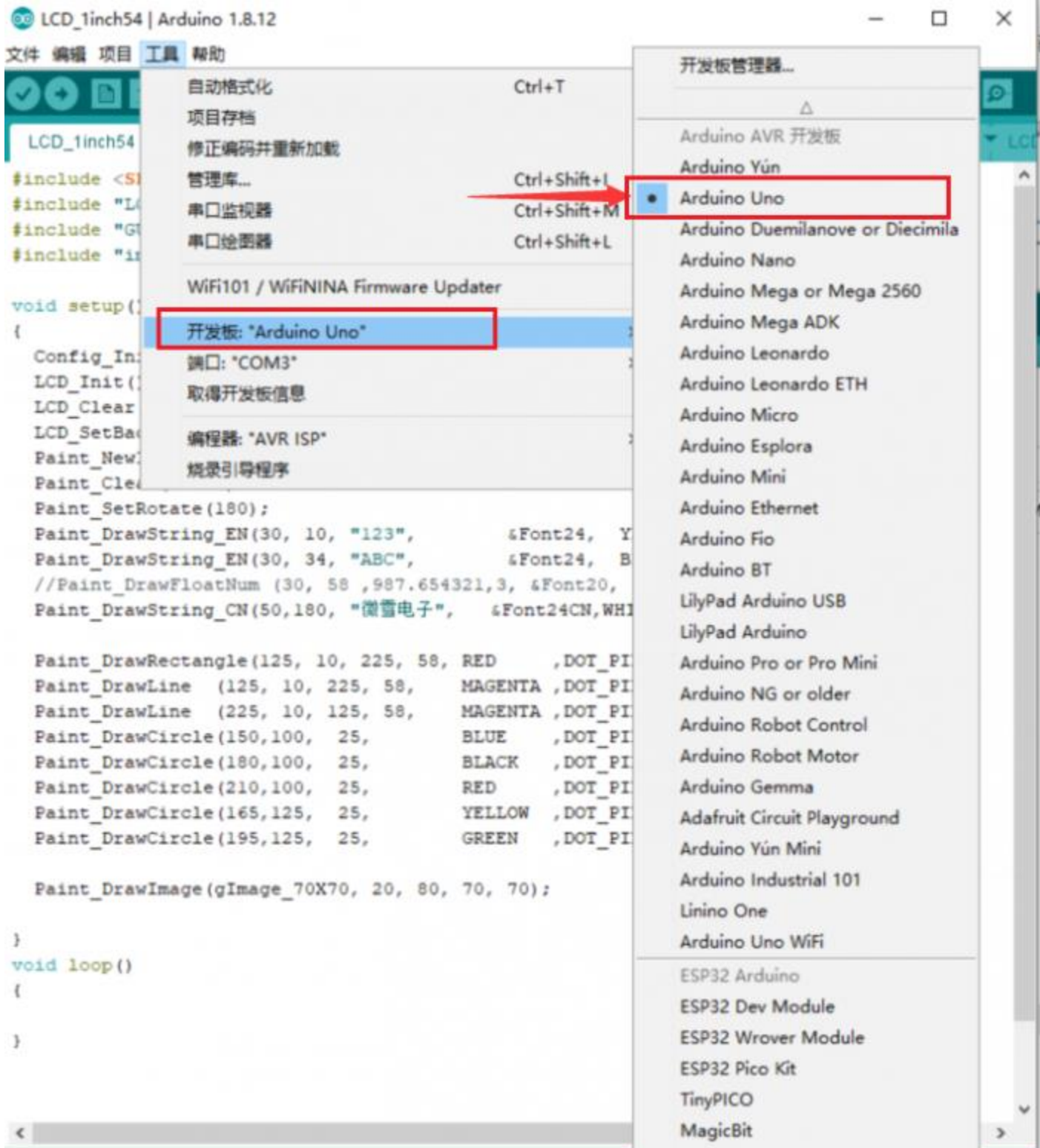
Download the demo codes and unzip it. The Arduino project is located in ~/Arduino/...

名称	修改日期	类型	大小
Arduino	2020/6/17 17:58	文件夹	
RaspberryPi	2020/6/17 17:58	文件夹	
STM32	2020/6/17 17:58	文件夹	

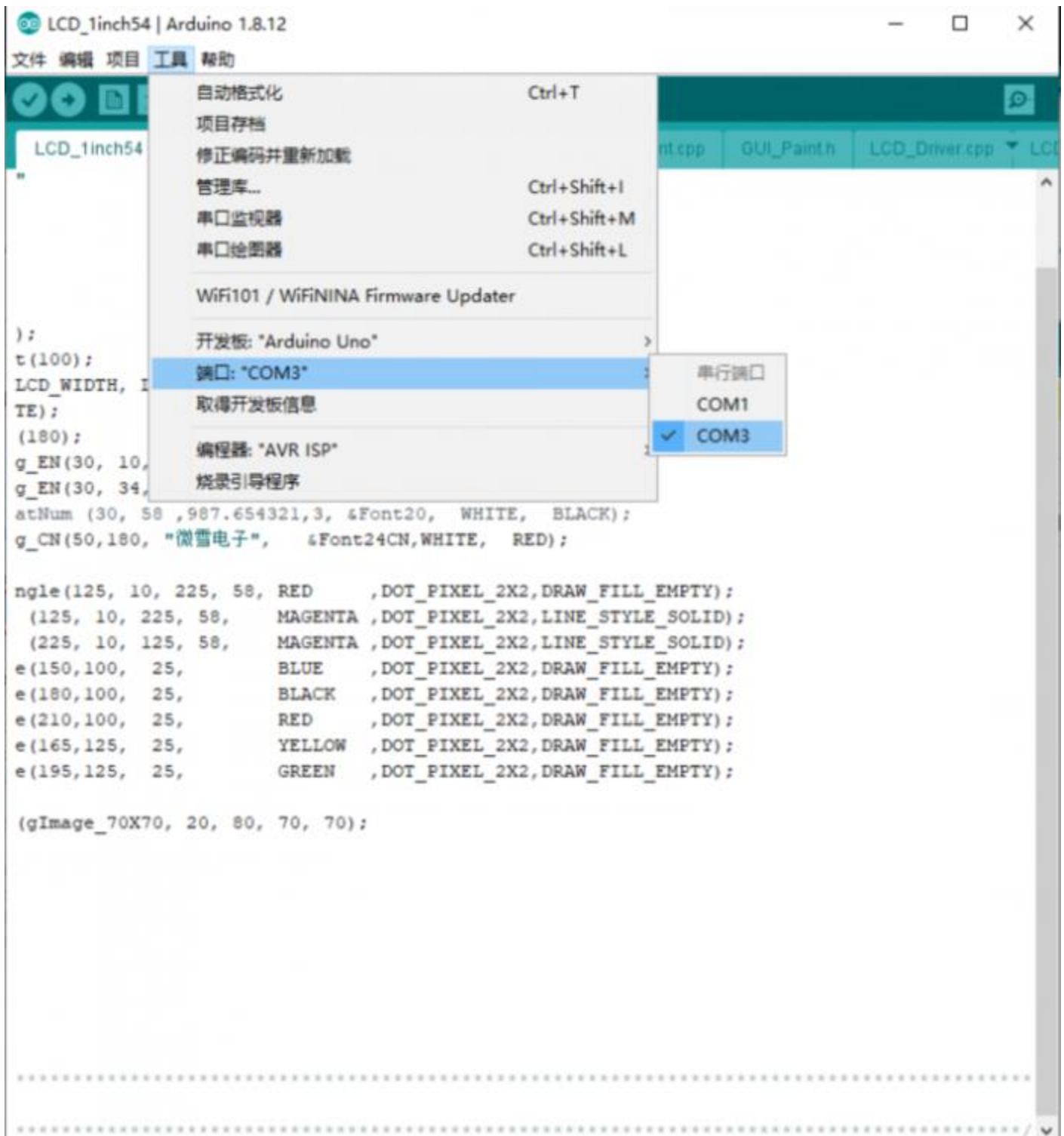
Run the project according to the actual display type

名称	修改日期	类型	大小
LCD_0inch96	2021/2/3 14:44	文件夹	
LCD_1inch3	2021/2/3 14:44	文件夹	
LCD_1inch8	2021/2/3 14:44	文件夹	
LCD_1inch14	2021/2/3 14:44	文件夹	
LCD_1inch28	2021/2/3 14:44	文件夹	
LCD_1inch54	2021/2/3 14:44	文件夹	
LCD_2inch	2021/2/3 14:44	文件夹	
LCD_2inch4	2021/2/3 14:44	文件夹	

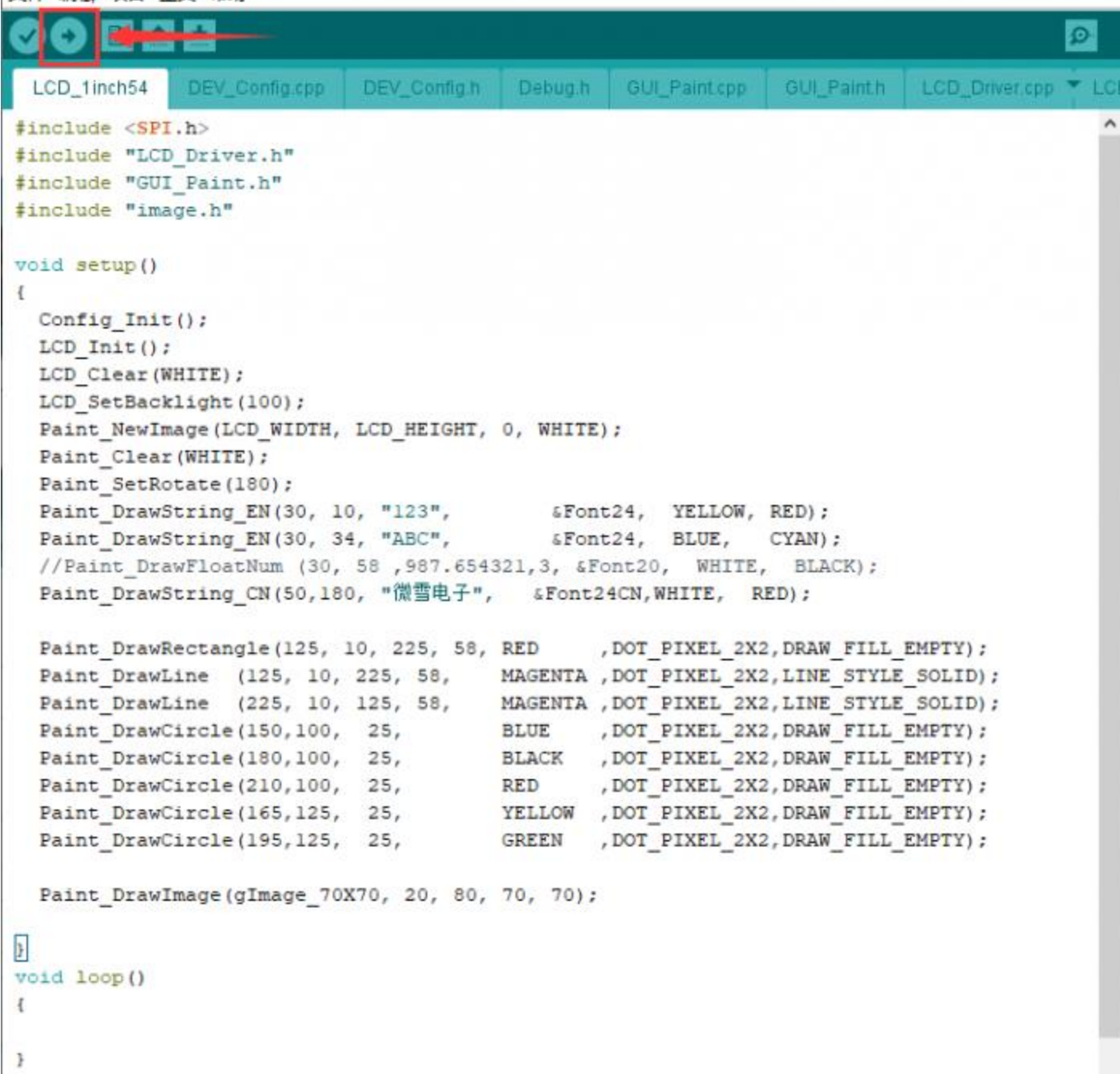
For examples: 1.54inch LCD Module. Enter the LCD_1inch54 directory and run the LCD_1inch54.ino file
Run the project and choose Arduino UNO as Board



Select the COM Port according to your Device Manager



Compile and download it to your board



```
#include <SPI.h>
#include "LCD_Driver.h"
#include "GUI_Paint.h"
#include "image.h"

void setup()
{
  Config_Init();
  LCD_Init();
  LCD_Clear(WHITE);
  LCD_SetBacklight(100);
  Paint_NewImage(LCD_WIDTH, LCD_HEIGHT, 0, WHITE);
  Paint_Clear(WHITE);
  Paint_SetRotate(180);
  Paint_DrawString_EN(30, 10, "123",      &Font24,  YELLOW,  RED);
  Paint_DrawString_EN(30, 34, "ABC",      &Font24,  BLUE,    CYAN);
  //Paint_DrawFloatNum (30, 58 ,987.654321,3, &Font20,  WHITE,  BLACK);
  Paint_DrawString_CN(50,180, "微雪电子",  &Font24CN,WHITE,  RED);

  Paint_DrawRectangle(125, 10, 225, 58, RED      ,DOT_PIXEL_2X2,DRAW_FILL_EMPTY);
  Paint_DrawLine  (125, 10, 225, 58,  MAGENTA ,DOT_PIXEL_2X2,LINE_STYLE_SOLID);
  Paint_DrawLine  (225, 10, 125, 58,  MAGENTA ,DOT_PIXEL_2X2,LINE_STYLE_SOLID);
  Paint_DrawCircle(150,100, 25,      BLUE    ,DOT_PIXEL_2X2,DRAW_FILL_EMPTY);
  Paint_DrawCircle(180,100, 25,      BLACK   ,DOT_PIXEL_2X2,DRAW_FILL_EMPTY);
  Paint_DrawCircle(210,100, 25,      RED     ,DOT_PIXEL_2X2,DRAW_FILL_EMPTY);
  Paint_DrawCircle(165,125, 25,      YELLOW  ,DOT_PIXEL_2X2,DRAW_FILL_EMPTY);
  Paint_DrawCircle(195,125, 25,      GREEN   ,DOT_PIXEL_2X2,DRAW_FILL_EMPTY);

  Paint_DrawImage(gImage_70X70, 20, 80, 70, 70);

}

void loop()
{
}
}
```

Resources

- [User Manual](#)
- [Demo code](#)
- [Schematic](#)

Datasheet

- [ST7735S](#)