Pico-ResTouch-LCD-2.8

From Waveshare Wiki
Jump to: navigation, search

Overview

Pico-ResTouch-LCD-2.8

(https://www.waveshare.com/pico-restouch-lcd-2.8.htm)

2.8inch Touch Display Module for Raspberry Pi Pico
262K Colors, 320 × 240, SPI

2.8inch Touch Display Module For Raspberry Pi Pico, 262K Colors, 320 × 240 Pixels, SPI Interface.

Specification

Parameter			
Working Voltage	5V	Resolution	320 × 240 Pixels
Communication Interface	SPI	Display Dimension	57.60 × 43.20 mm
Display Panel	IPS	Pixel Pitch	0.18 × 0.18 mm
Control Chip	ST7789	Dimension	70.20 × 50.20 mm

Hardware connection

Please take care of the direction when you connect Pico, an USB port is printed to indicate. You can also check the pin of Pico and the LCD board when connecting. You can connect the display according to the table.

LCD	Pico	Description
VCC	VSYS	Power input
GND	GND	GND
SDIO_CLK	GP5	SCK pin of SDIO interfacem clock input for slave device.
LCD_DC	GP8	Data/Command pin (High: data; Low: command)
LCD_CS	GP9	Chip select pin of LCD (Low active)

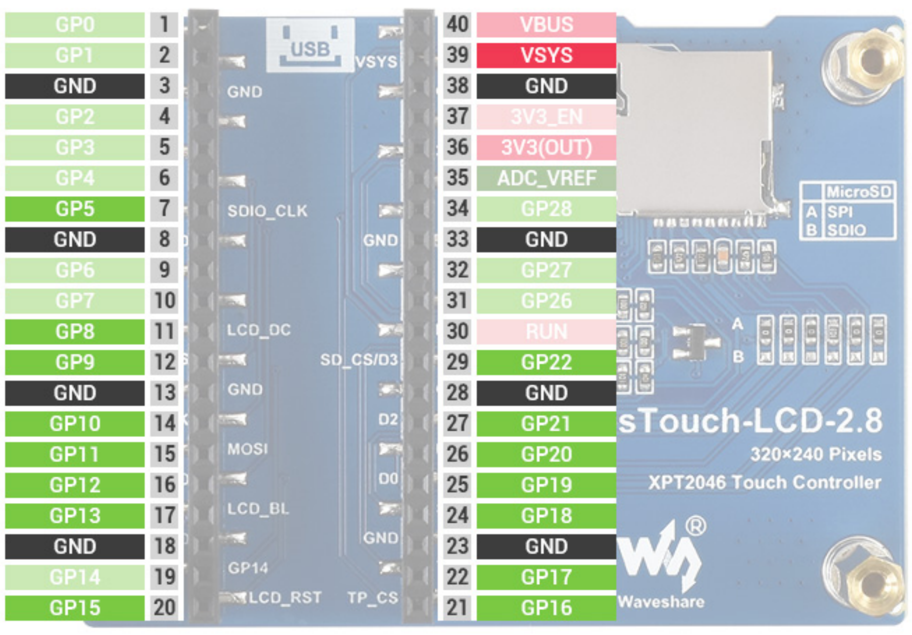
LCD_CLK	GP10	CLK pin of LCD communication, clock input for slave device
MOSI	GP11	SPI MOSI pin
MISO	GP12	SPI MISO pin
LCD_BL	GP13	LCD backlight control
LCD_RST	GP15	LCD reset pin (Low active)
TP_CS	GP16	Touch controller chip select (Low active)
TP_IRQ	GP17	Touch controller interrupt pin (Low active)
SDIO_CMD	GP18	SDIO CMD pin
D0	GP19	SDIO D0 pin
D1	GP20	SDIO D1 pin
D2	GP21	SDIO D2 pin
SD_CS/D3	GP22	SDIO CS/D3 pin

Features

- 320 × 240 resolution, IPS screen, 262K RGB, clear and colorful displaying effect.
- Dedicated touch controller, bringing more smooth touching effect than AD-controlled solutions.
- MicroSD card slot for storing images and direct displaying them easily.
- Programmable backlight control, power saving.
- Comes with development resources and manual (Raspberry Pi Pico C/C++ and MicroPython examples).

Pin

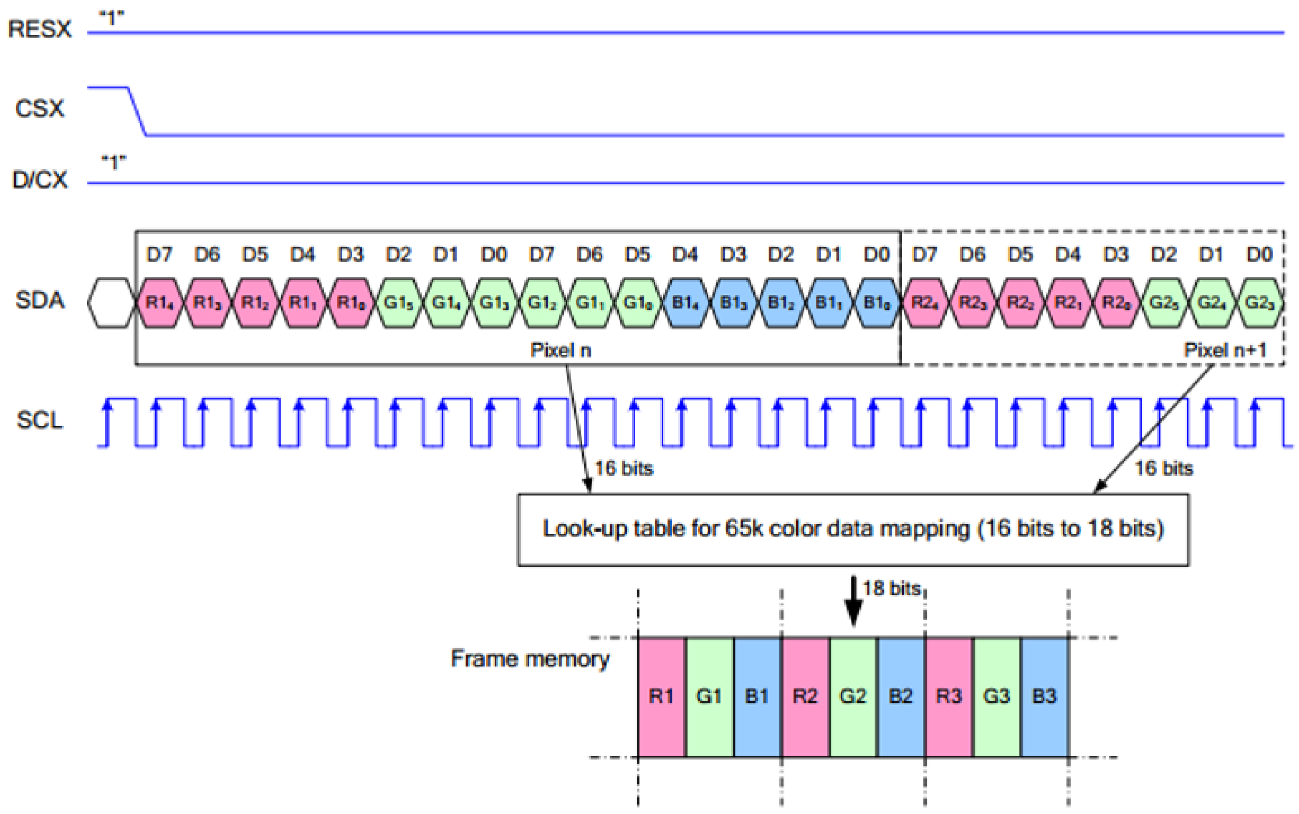
The built-in controller used by this LCD is ST7789VW, which is an LCD controller with 240 x RGB x 320 pixels, while the pixels of this LCD itself are 240 (H) RGB x 320 (V), and the internal RAM of the LCD has been fully used. The LCD supports 16-bit or 18-bit input color formats per pixel, namely RGB565, and RGB666. The test demo uses RGB565 color format, which is a commonly used RGB format. LCD uses a four-wire SPI communication interface, which can greatly save GPIO ports, At the same time, the communication speed will be faster. The maximum SPI write speed tested is 60MHz.



VSYS		5V power supply
GND		Ground
GP5	SDIO_CLK	SDIO CLK pin
GP8	LCD_DC	LCD Command/Data pin
GP9	LCD_CS	LCD chip select
GP10	CLK	SPI CLK pin
GP11	MOSI	SPI MOSI pin
GP12	MISO	SPI MISO pin
GP13	LCD_BL	LCD backlight

GP15	LCD_RST	LCD reset pin
GP16	TP_CS	Touch controller chip select
GP17	TP_IRQ	Touch controller interrupt
GP18	SDIO_CMD	SDIO CMD pin
GP19	D0	SDIO D0 pin
GP20	D1	SDIO D1 pin
GP21	D2	SDIO D2 pin
GP22	SD_CS/D3	SDIO CS/D3 pin

(/wiki/File:Pico-ResTouch-LCD-2.8-details-inter.jpg)


RESX
"1"
CSX
D/CX
"1"
SDA
D7 D6 D5 D4 D3 D2 D1 D0 D7 D6 D5 D4 D3 D2 D1 D0 D7 D6 D5 D4 D3 D2 D1 D0
Pixel n
Pixel n+1
SCL
16 bits
16 bits
Look-up table for 65k color data mapping (16 bits to 18 bits)
18 bits
Frame memory
R1 G1 B1 R2 G2 B2 R3 G3 B3

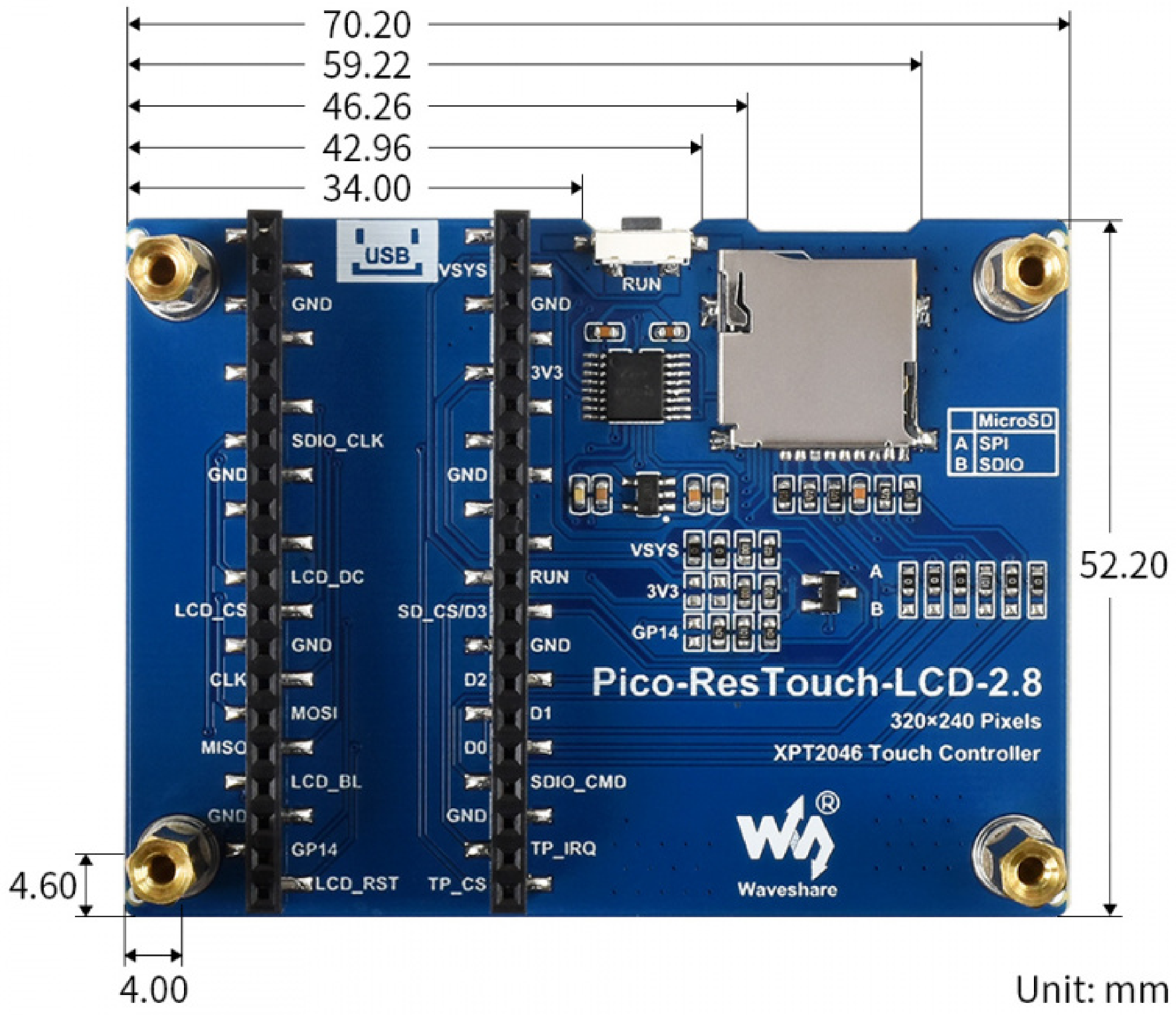
(/wiki/File:0.96inch_lcd_module_spi.png)

Note: The difference with the traditional SPI protocol is that the data cable from the slave to the host is hidden as this product is mostly used for display. Please refer to Datasheet Page 55 (8.4 Serial Interface) (https://www.waveshare.com/w/upload/a/ae/ST7789_Datasheet.pdf).

- RESX (LCD_RST) indicates reset and is set to "1" when the module is powered on.
- CSX (LCD_CS) indicates the slave chip selection and the chip is enabled when CS is low.
- D/CX (LCD_DC) indicates the chip data/command control pin. When DC=0, it is writing commands. When DC=1, it is writing data.
- SDA (MOST) is the transmitted data, that is, RGB data.
- SCL (CLK) is the SPI clock.
- As for SPI communication, the data has transmission order, that is the combination of clock phase (CPHA) and clock polarity (CPOL).
- The level of CPHA determines whether the serial synchronization clock is collected on the first clock transition edge or the second clock transition edge. When CPHA = 0, data acquisition is performed on the first transition edge.
- The level of CPOL determines the idle state level of the serial synchronous clock. CPOL = 0, which is a low level.

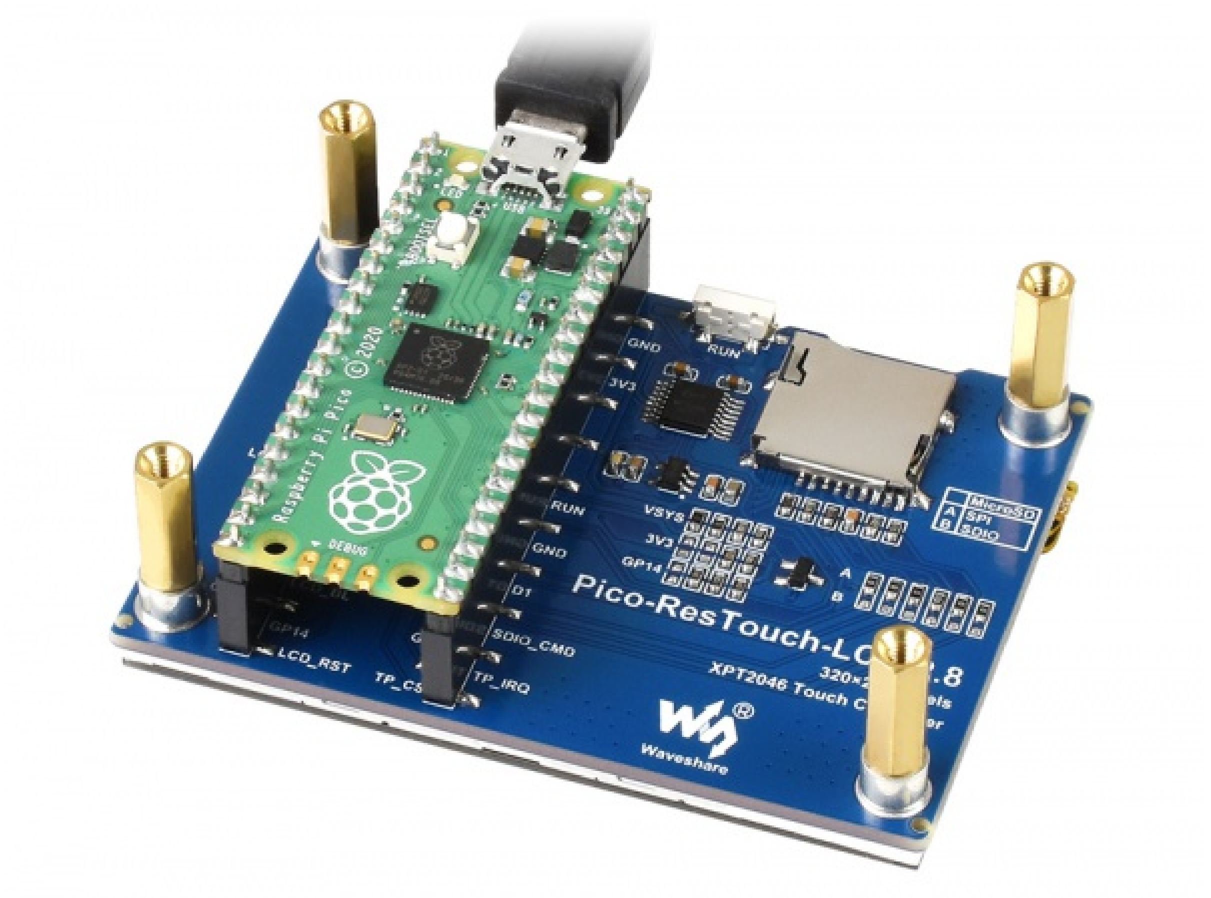
- As can be seen from the figure, when the first falling edge of SCLK starts to transmit data, 8bit data is transmitted in one clock cycle, using SPI0, bit-wise transmission, high-order first, and low-order last.

Dimension



(/wiki/File:Pico_ResTouch_LCD_2.8_Guide.jpg)

Connection



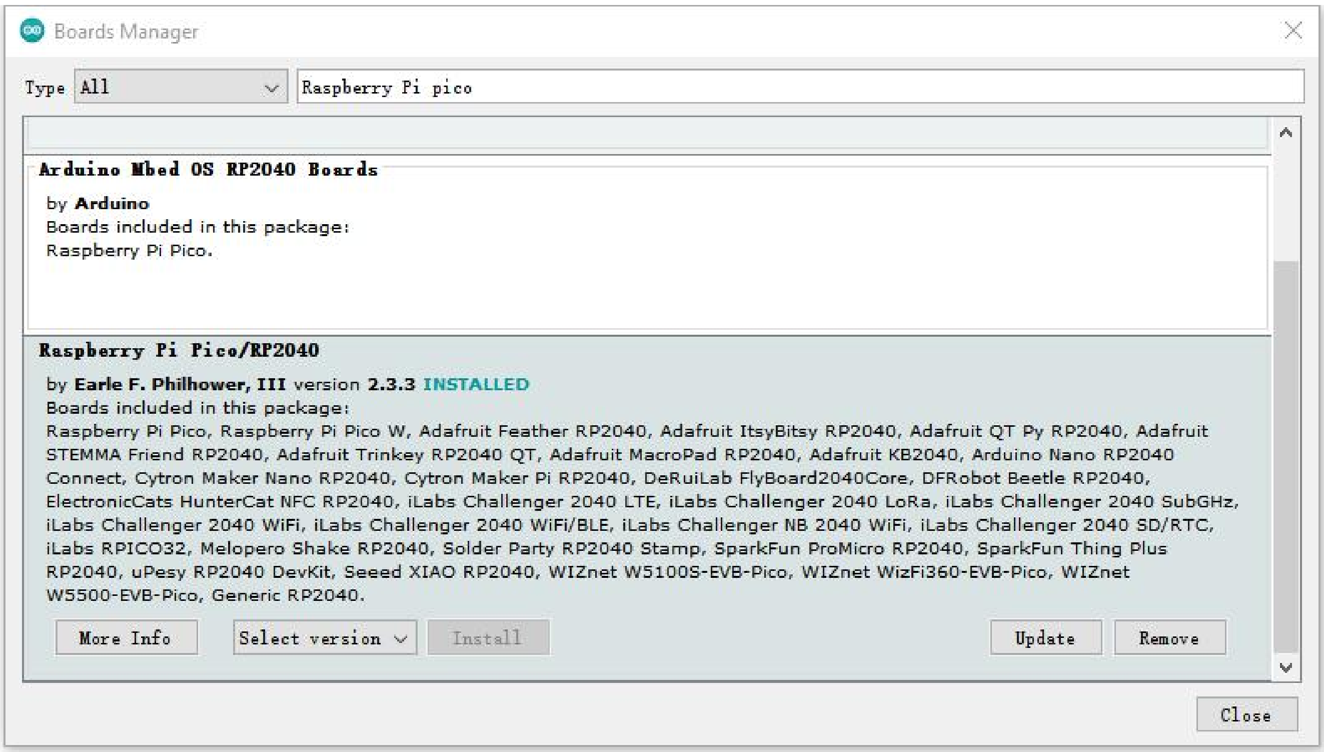
(/wiki/File:Pico-ResTouch-LCD-2.8-details-3.jpg)

Pay attention to the connection direction of the Pico, the USB of the Pico should be in the same direction as the MicroSD card.

Setup environment

- We test the demo with Arduino IDE (https://www.arduino.cc/), VScode(cmake) (https://code.visualstudio.com/download), Thonny (https://thonny.org/), click to download the related IDE, and open Arduino IDE, Thonny after installation.

1. Please install Pico SDK in Arduino IDE, click Tools->Board->Boards Manager in the menu bar and search for Raspberry Pi Pico, find the corresponding library, and click Install to install it, as shown in the figure below:

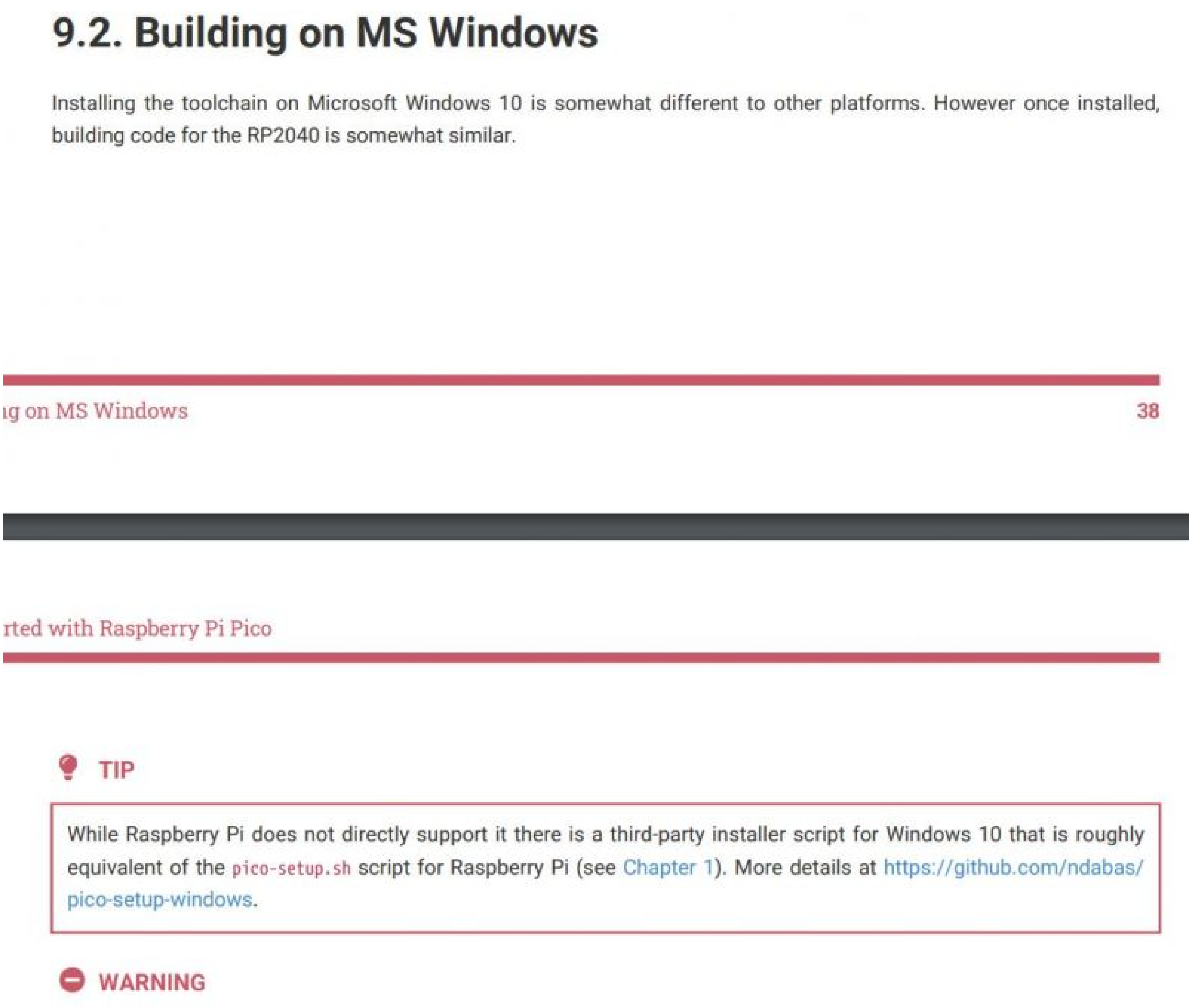


Arduino Mbed OS RP2040 Boards
by **Arduino**
Boards included in this package:
Raspberry Pi Pico.

Raspberry Pi Pico/RP2040
by **Earle F. Philhower, III** version **2.3.3** **INSTALLED**
Boards included in this package:
Raspberry Pi Pico, Raspberry Pi Pico W, Adafruit Feather RP2040, Adafruit ItsyBitsy RP2040, Adafruit QT Py RP2040, Adafruit STEMMA Friend RP2040, Adafruit Trinkey RP2040 QT, Adafruit MacroPad RP2040, Adafruit KB2040, Arduino Nano RP2040 Connect, Cytron Maker Nano RP2040, Cytron Maker Pi RP2040, DeRuiLab FlyBoard2040Core, DFRobot Beetle RP2040, ElectronicCats HunterCat NFC RP2040, iLabs Challenger 2040 LTE, iLabs Challenger 2040 LoRa, iLabs Challenger 2040 SubGHz, iLabs Challenger 2040 WiFi, iLabs Challenger 2040 WiFi/BLE, iLabs Challenger NB 2040 WiFi, iLabs Challenger 2040 SD/RTC, iLabs RPICO32, Melopero Shake RP2040, Solder Party RP2040 Stamp, SparkFun ProMicro RP2040, SparkFun Thing Plus RP2040, uPesy RP2040 DevKit, Seeed XIAO RP2040, WIZnet W5100S-EVB-Pico, WIZnet WizFi360-EVB-Pico, WIZnet W5500-EVB-Pico, Generic RP2040.

(/wiki/File:Pico_10dof_imu_spec40.jpg)

2. Please refer to The C/C++ SDK (https://www.raspberrypi.com/documentation/microcontrollers/c_sdk.html), 9.2. Building on MS Windows in GET-START document (https://www.waveshare.com/w/upload/6/65/Getting-started-with-pico.pdf) for the compilation environment of VScode(Cmake).



9.2. Building on MS Windows

Installing the toolchain on Microsoft Windows 10 is somewhat different to other platforms. However once installed, building code for the RP2040 is somewhat similar.

ıg on MS Windows 38

rted with Raspberry Pi Pico

TIP

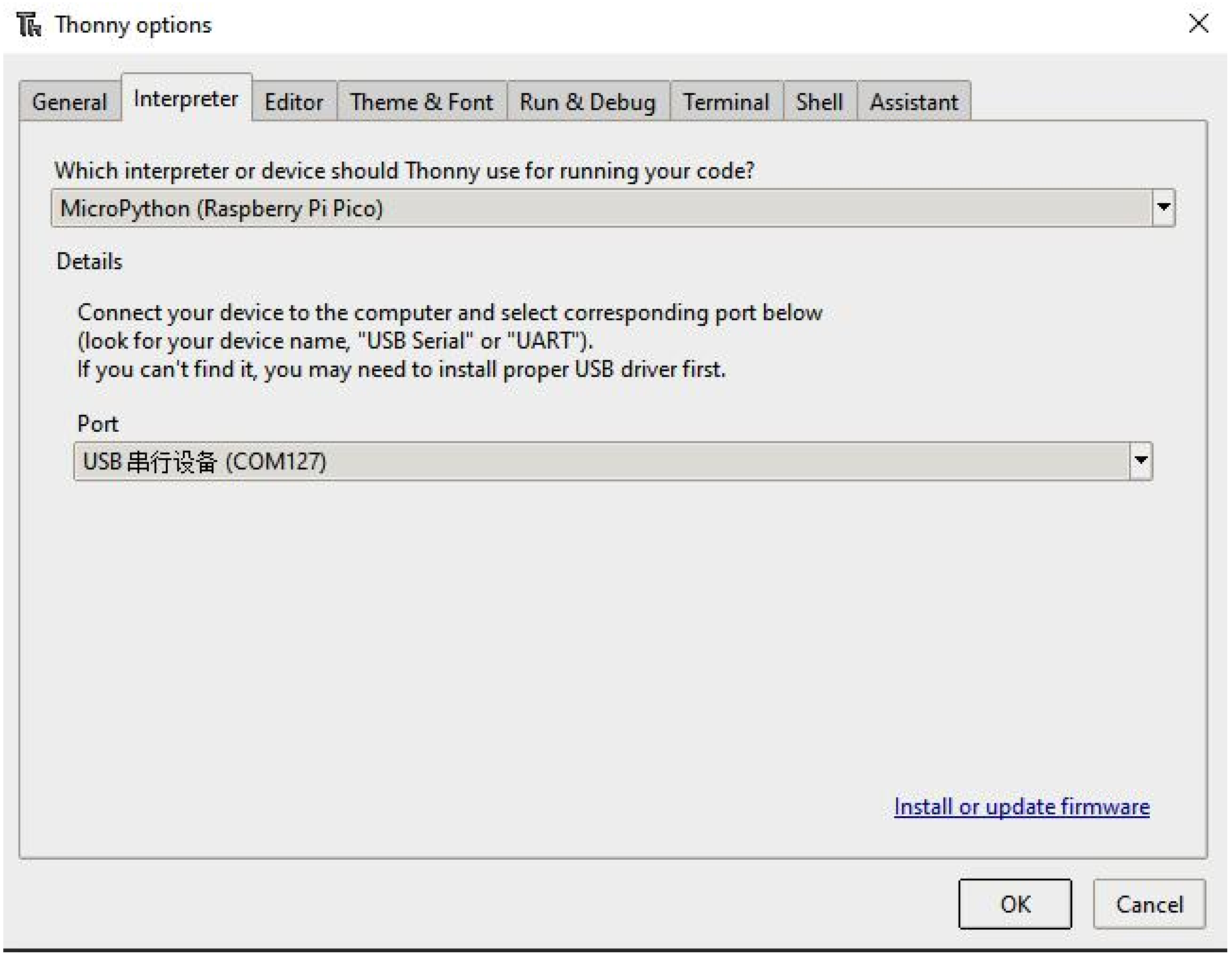
While Raspberry Pi does not directly support it there is a third-party installer script for Windows 10 that is roughly equivalent of the `pico-setup.sh` script for Raspberry Pi (see Chapter 1). More details at https://github.com/ndabas/pico-setup-windows.

WARNING

(/wiki/File:Pico_ResTouch_LCD02.jpg)

3. Please refer to the official micropython document (https://www.raspberrypi.com/documentation/microcontrollers/micropython.html) to set up the environment, and select

the Raspberry Pi Pico device in Thonny's Tools->Options->Interprete. As shown below:


Thonny options

Which interpreter or device should Thonny use for running your code?
MicroPython (Raspberry Pi Pico)

Details

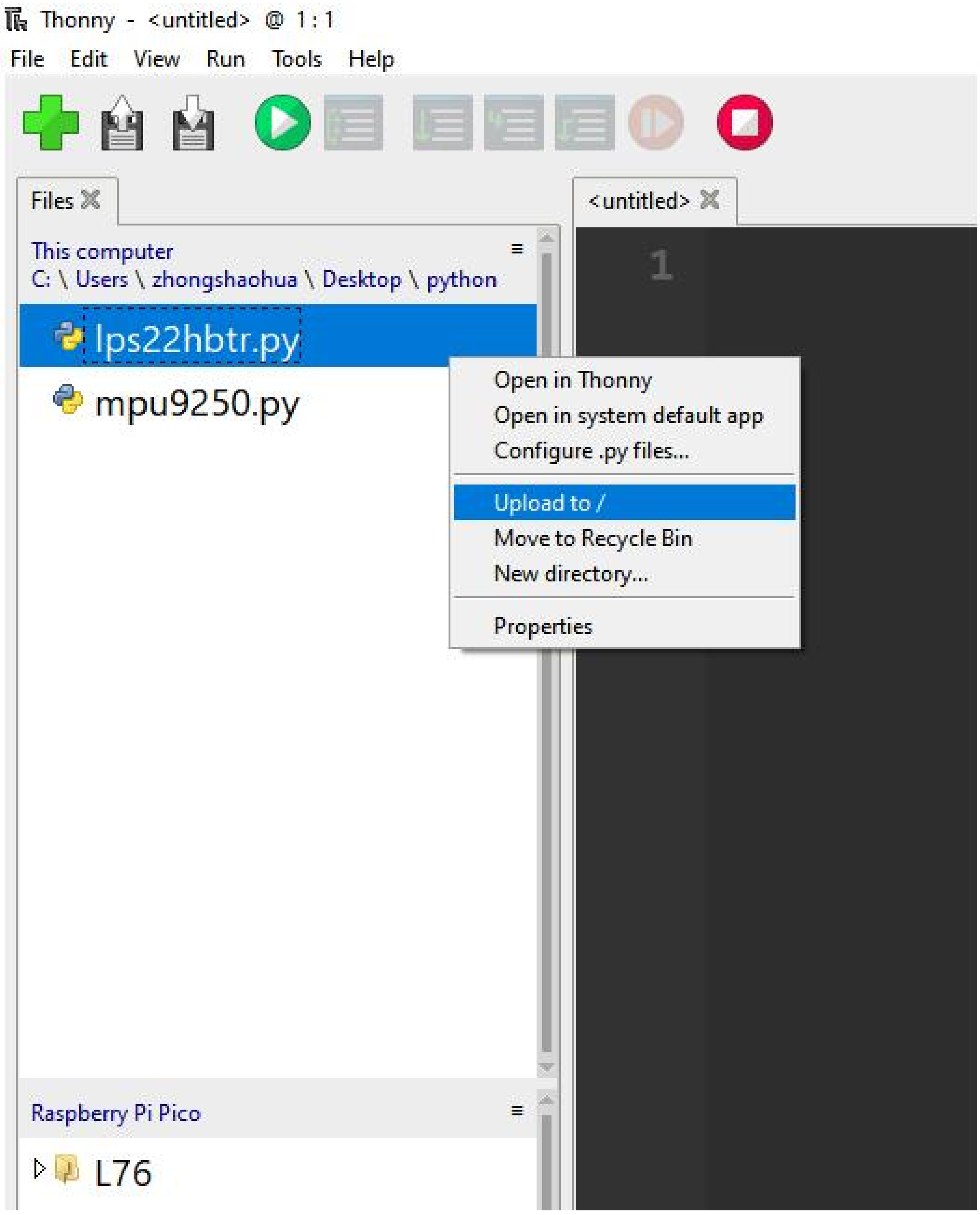
Connect your device to the computer and select corresponding port below
(look for your device name, "USB Serial" or "UART").
If you can't find it, you may need to install proper USB driver first.

Port
USB 串行设备 (COM127)

(/wiki/File:Pico-GPS-002.jpg)

Download Demo codes

1. Click to download the sample demo (https://www.waveshare.com/w/upload/8/83/Pico-ResTouch-LCD-X_X_Code_%286%29.zip).
2. Please open the VScode (Cmake) project with VScode software.
3. Unzip the sample demo and click ".ino" to open the Arduino demo. Please upload the Micropython sample demo to Pico document system. As shown below:

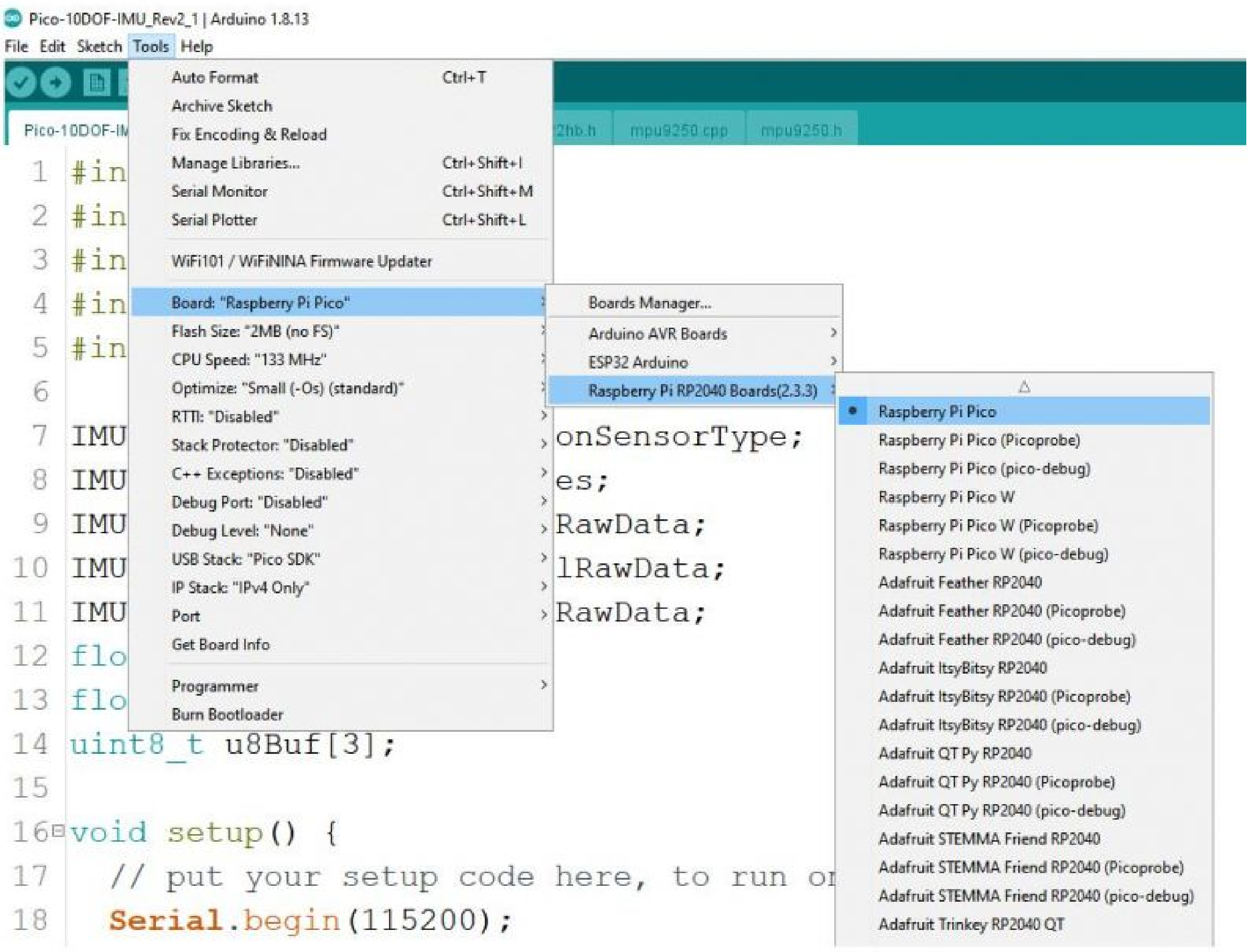


(/wiki/File:Thonny03.jpg)

Run the Demo codes

Arduino

1. Open Arduino IDE or ".ino" sample program. Click "Tools->Board->Raspberry Pi Pico" in the Tools. As shown below.


Pico-10DOF-IMU_Rev2_1 | Arduino 1.8.13
File Edit Sketch Tools Help
Pico-10DOF-IM
2hb.h
mpu9250.cpp
mpu9250.h
Auto Format Ctrl+T
Archive Sketch
Fix Encoding & Reload
Manage Libraries... Ctrl+Shift+I
Serial Monitor Ctrl+Shift+M
Serial Plotter Ctrl+Shift+L
WiFi101 / WiFiNINA Firmware Updater
Get Board Info
Programmer
Burn Bootloader
Boards Manager...
Arduino AVR Boards
ESP32 Arduino
Raspberry Pi RP2040 Boards(2.3.3)
Raspberry Pi Pico
Raspberry Pi Pico (Picoprobe)
Raspberry Pi Pico (pico-debug)
Raspberry Pi Pico W
Raspberry Pi Pico W (Picoprobe)
Raspberry Pi Pico W (pico-debug)
Adafruit Feather RP2040
Adafruit Feather RP2040 (Picoprobe)
Adafruit Feather RP2040 (pico-debug)
Adafruit ItsyBitsy RP2040
Adafruit ItsyBitsy RP2040 (Picoprobe)
Adafruit ItsyBitsy RP2040 (pico-debug)
Adafruit QT Py RP2040
Adafruit QT Py RP2040 (Picoprobe)
Adafruit QT Py RP2040 (pico-debug)
Adafruit STEMMA Friend RP2040
Adafruit STEMMA Friend RP2040 (Picoprobe)
Adafruit STEMMA Friend RP2040 (pico-debug)
Adafruit Trinkey RP2040 QT

Board: "Raspberry Pi Pico"
Flash Size: "2MB (no FS)"
CPU Speed: "133 MHz"
Optimize: "Small (-Os) (standard)"
RTTI: "Disabled"
Stack Protector: "Disabled"
C++ Exceptions: "Disabled"
Debug Port: "Disabled"
Debug Level: "None"
USB Stack: "Pico SDK"
IP Stack: "IPv4 Only"
Port

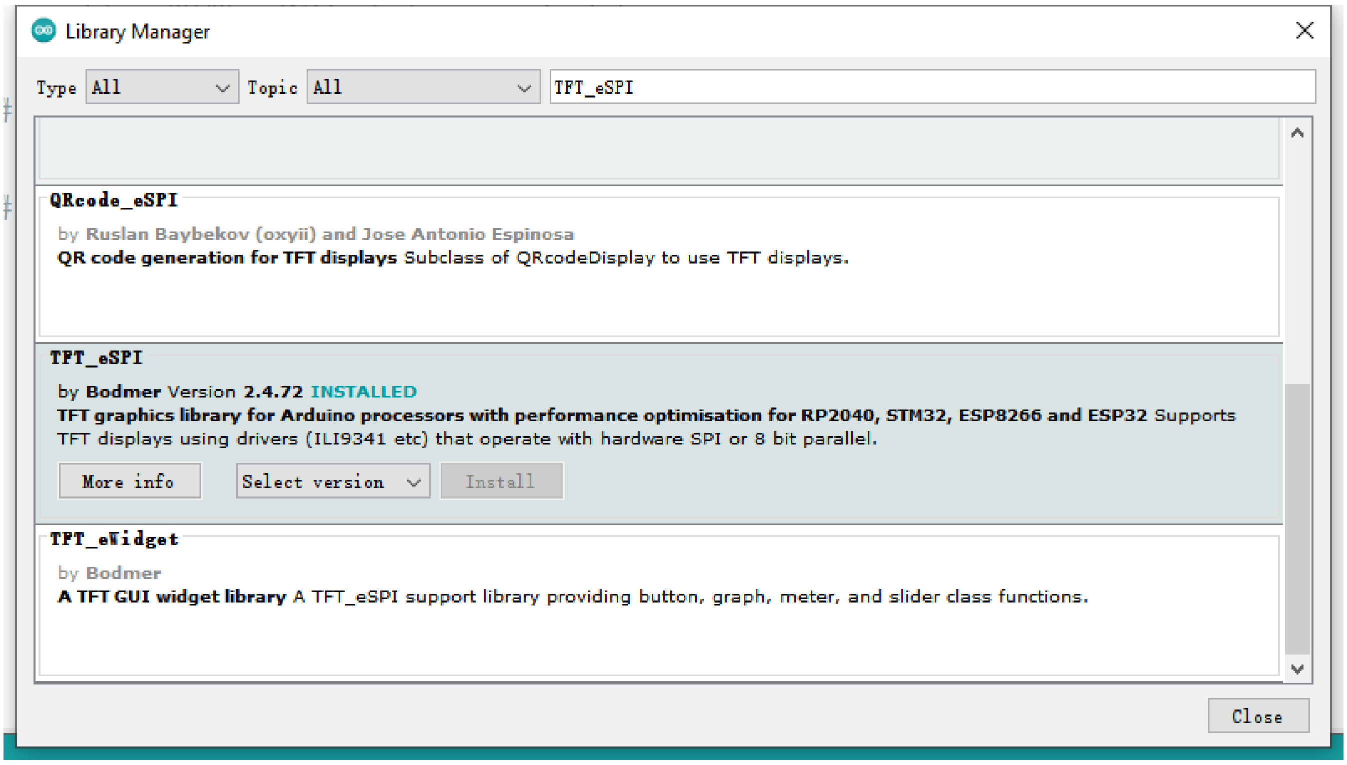
```
#in
#in
#in
#in
#in

IMU                             onSensorType;
IMU                             es;
IMU                             RawData;
IMU                             lRawData;
IMU                             RawData;
flo
flo
uint8_t u8Buf[3];

void setup() {
  // put your setup code here, to run o
  Serial.begin(115200);
```

(/wiki/File:Pico-GPS-003.jpg)

2. Install the TFT_eSPI library, click Tools->Manage Library in the menu bar, search for TFT_eSPI and click Install.



Type All Topic All TFT_eSPI

QRcode_eSPI
by Ruslan Baybekov (oxyii) and Jose Antonio Espinosa
QR code generation for TFT displays Subclass of QRcodeDisplay to use TFT displays.

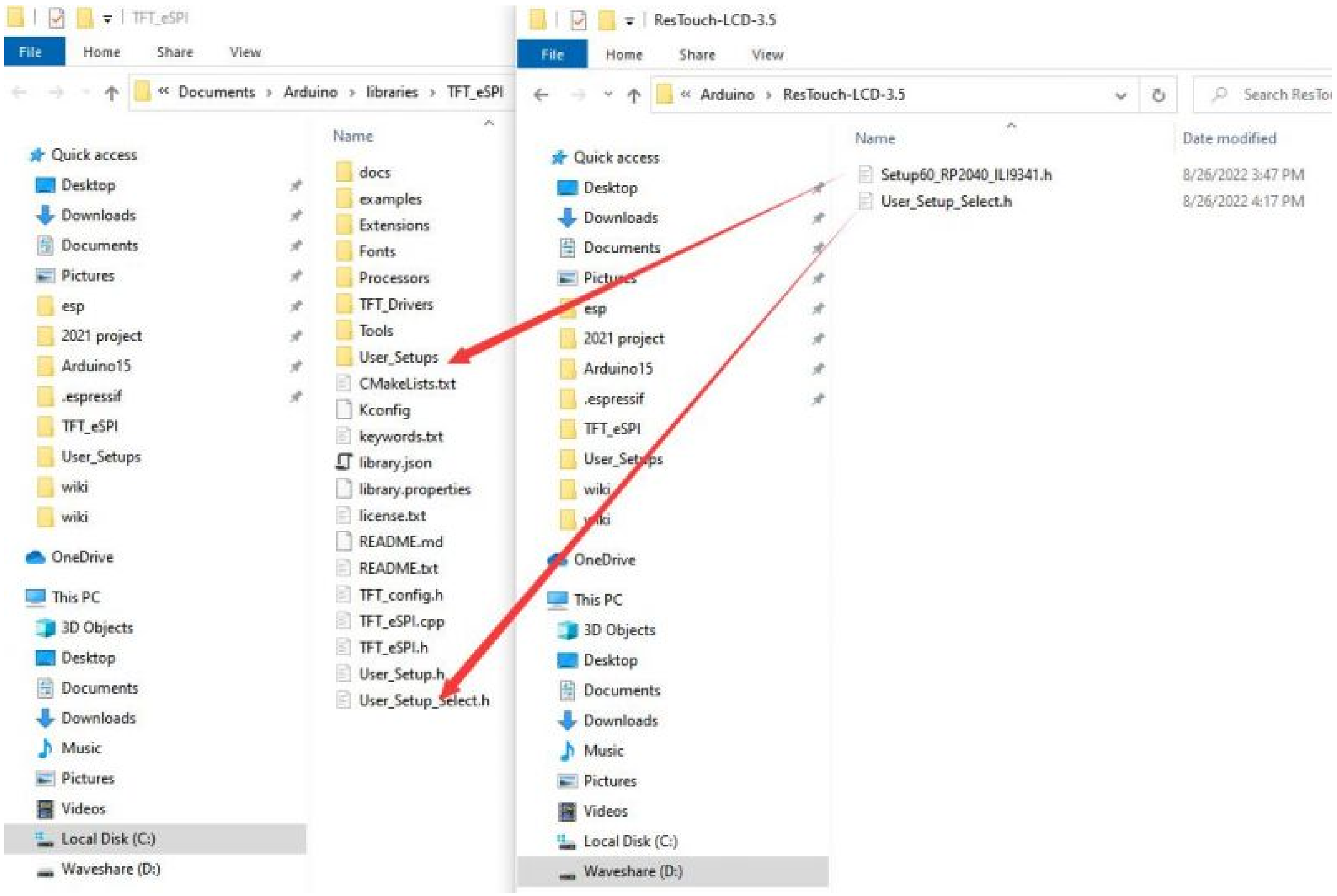
TFT_eSPI
by **Bodmer** Version **2.4.72** INSTALLED
TFT graphics library for Arduino processors with performance optimisation for RP2040, STM32, ESP8266 and ESP32 Supports TFT displays using drivers (ILI9341 etc) that operate with hardware SPI or 8 bit parallel.

TFT_eWidget
by Bodmer
A TFT GUI widget library A TFT_eSPI support library providing button, graph, meter, and slider class functions.

(/wiki/File:Pico_ResTouch_LCD_3.5_Guide09.png)

3. To configure the driver file, open the Arduino library file directory, usually in

C:\Users\xxxx\Documents\Arduino\libraries\TFT_eSPI\ , for ResTouch-LCD-3.5, put the TFT_eSPI library (User_Setups\Setup60_RP2040_ILI9341.h) (User_Setup_Select.h) with the files in the sample program folder Arduino\ResTouch-LCD-3.5, and the same for ResTouch-LCD-2.8, as shown in the figure.



(/wiki/File:Pico_ResTouch_LCD_3.5_Guide11.jpg)

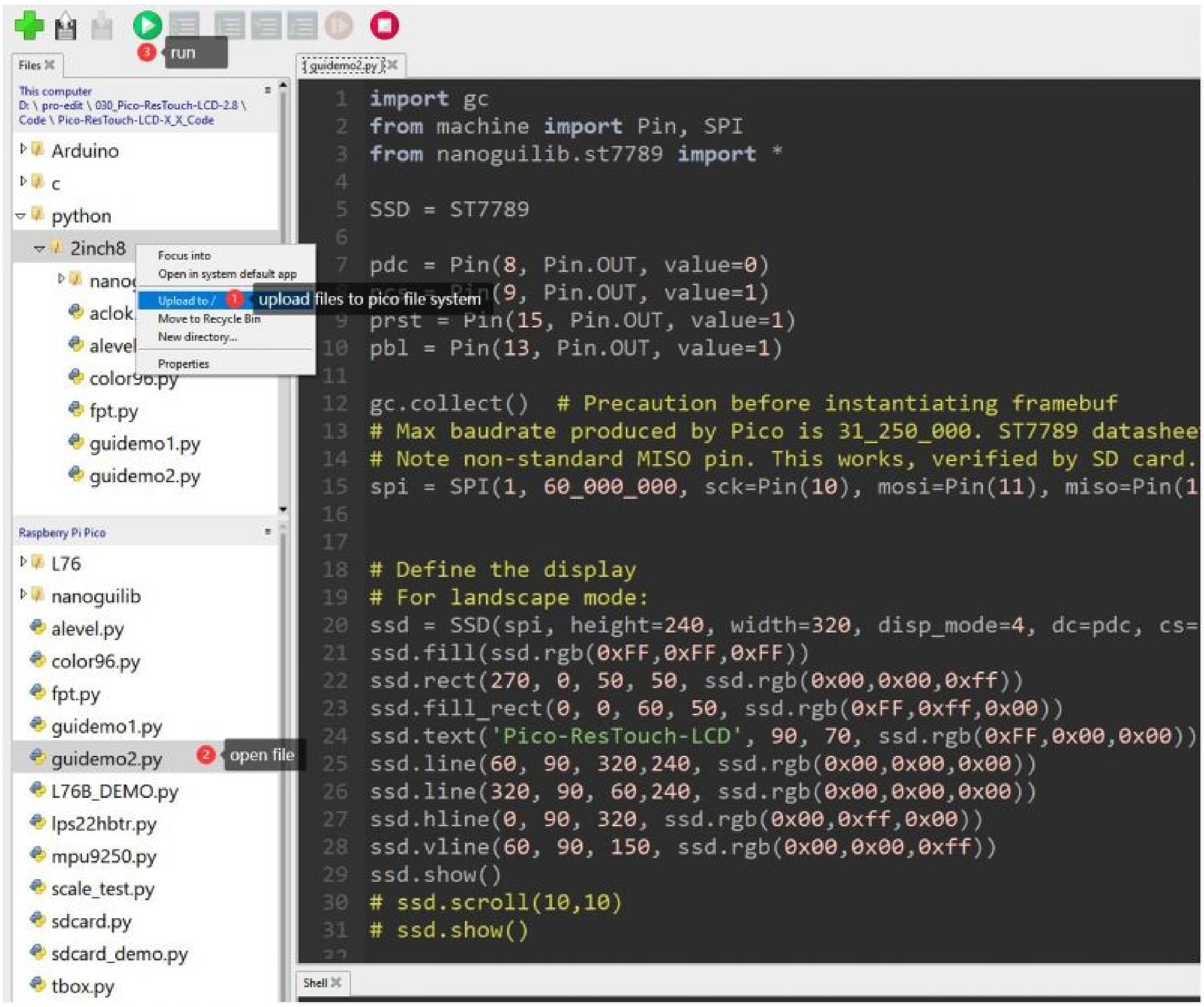
4. Select the example program under File-Examples->TFT_eSPI->480*320 in the menu bar, and then click Upload under Edit to download the code to Pico.

VScode(Cmake)

- Open the c project with VScode, and then compile to download.

Micropython

- Open Thonny IDE, and save the code under "python/2inch8/" to the Pico file system. As shown below, and the same for ResTouch-LCD-2.8.



```
import gc
from machine import Pin, SPI
from nanoguilib.st7789 import *

SSD = ST7789

pdc = Pin(8, Pin.OUT, value=0)
pcs = Pin(9, Pin.OUT, value=1)
prst = Pin(15, Pin.OUT, value=1)
pbl = Pin(13, Pin.OUT, value=1)

gc.collect()  # Precaution before instantiating framebuf
# Max baudrate produced by Pico is 31_250_000. ST7789 datashee
# Note non-standard MISO pin. This works, verified by SD card.
spi = SPI(1, 60_000_000, sck=Pin(10), mosi=Pin(11), miso=Pin(1


# Define the display
# For landscape mode:
ssd = SSD(spi, height=240, width=320, disp_mode=4, dc=pdc, cs=
ssd.fill(ssd.rgb(0xFF,0xFF,0xFF))
ssd.rect(270, 0, 50, 50, ssd.rgb(0x00,0x00,0xff))
ssd.fill_rect(0, 0, 60, 50, ssd.rgb(0xFF,0xff,0x00))
ssd.text('Pico-ResTouch-LCD', 90, 70, ssd.rgb(0xFF,0x00,0x00))
ssd.line(60, 90, 320,240, ssd.rgb(0x00,0x00,0x00))
ssd.line(320, 90, 60,240, ssd.rgb(0x00,0x00,0x00))
ssd.hline(0, 90, 320, ssd.rgb(0x00,0xff,0x00))
ssd.vline(60, 90, 150, ssd.rgb(0x00,0x00,0xff))
ssd.show()
# ssd.scroll(10,10)
# ssd.show()
```

(/wiki/File:Pico_ResTouch_LCD_3.5_Guide10.jpg)

Codes Analysis

The example will display strings, figures, image and finally the touch pad function. We use three drawing functions in main part and set the TP_DrawBoard() to loop for releasing the features.

```
GUI_Show();
LCD_Show_bmp(Bmp_ScanDir , Lcd_ScanDir);
TP_DrawBoard();
```

Note that if you want to test the LCD_ShowBMP example, you need to copy the picture from the PIC folder to the root directory of a micro SD card, insert the SD card into the slot in the backside of the LCD. Then run the examples.

- The micro SD card should in FAT format, and the resolution of pictures used should be the same as the LCD, for a 2.8inch LCD, it is 320 × 240， and 480 × 320 for a 3.5inch LCD. 24bit BMP.

The LCD controller is ST7789, we need to intialize the controller at the first, which is done in LCD_Driver.c file
And being called in lcd_test.c file,

```
 System_Init();//System intialize, configure serial port and SPI interface...<br/>
```

```
  LCD_SCAN_DIR Lcd_ScanDir = SCAN_DIR_DFT; //Set the scanmode <br/>
  LCD_Init( Lcd_ScanDir, 200);//Initialize LCD panel, confirm the scan mode and the
brightness<br/>
```

GUI functions are all saved in LCD_GUI.c file, you can call them to draw the display

- Draw point

```
void GUI_DrawPoint(POINT Xpoint, POINT Ypoint, COLOR Color,
                   DOT_PIXEL Dot_Pixel, DOT_STYLE DOT_STYLE)
```

- Draw line (dotted or solid):

```
void GUI_DrawLine(POINT Xstart, POINT Ystart, POINT Xend, POINT Yend,
                  COLOR Color, LINE_STYLE Line_Style, DOT_PIXEL Dot_Pixel)
```

- Draw a rectangle (empty or filled)

```
void GUI_DrawRectangle(POINT Xstart, POINT Ystart, POINT Xend, POINT Yend,
                       COLOR Color, DRAW_FILL Filled, DOT_PIXEL Dot_Pixel)
```

- Draw a circle (empty or filled)

```
void GUI_DrawCircle(POINT X_Center, POINT Y_Center, LENGTH Radius,
                    COLOR Color, DRAW_FILL  Draw_Fill , DOT_PIXEL Dot_Pixel)
```

- Display character

```
void GUI_DisChar(POINT Xpoint, POINT Ypoint, const char Acsii_Char,
                 sFONT* Font, COLOR Color_Background, COLOR Color_Foreground)
```

- Display string

```
void GUI_DisString_EN(POINT Xstart, POINT Ystart, const char * pString,
                      sFONT* Font, COLOR Color_Background, COLOR Color_Foreground )
```

- Display number

```
void GUI_DisNum(POINT Xpoint, POINT Ypoint, int32_t Nummber,
                sFONT* Font, COLOR Color_Background, COLOR Color_Foreground )
```

- Display time

```
void GUI_Showtime(POINT Xstart, POINT Ystart, POINT Xend, POINT Yend,
                  DEV_TIME *pTime, COLOR Color)
```

In the example, this demo shows that the BMP picture first reads the picture data in the BMP format on the SD card through the SPI protocol and displays it.
In lcd_test.c file, we use two functions for display picture:

```
SD_Init();//Initialize SD card
```

```
LCD_Show_bmp(bmp_scan_dir,lcd_scan_dir);//Display BMP picture
```

These functions are written in LCD_Bmp.c, actually read the picture data in the BMP format with a specific file name from the SD card and then call the display function written by ourselves to "express" the data as an image again

In LCD_Touch.c file:

```
  TP_Init( Lcd_ScanDir );//Initialize touch panel and set the scan mode
```

```
  TP_GetAdFac();//Calibrate the display
```

```
  TP_Dialog();//Clear
```

```
  TP_DrawBoard();//Enable the drawing board
```

There will be five colors on the right side of the screen, the default color is black, touch them to select the pen color; click the AD button, and follow the on-screen prompts to click the red + sign to calibrate the screen; click the CLEAR button in upper right corner to clear the drawing board
The touch experiment uses four sets of calibration values by default, which can meet the brush operations in four directions. There are five color choices on the right, and the default brush size is 9 pixels.
The function for touching are saved in the LCD_Touch.c file

There are five fonts available.

```
Width 5,  Height 8    font8
Width 7, Height 12    font12
Width 11, Height 16    font16
Width 14, Height 20    font20
Width 17, Height24    font24
```

- If you need characters in different sizes and fonts, you can generate the font library you want by the font extraction software provided in the Resources
- In fact, you can use the Image2Lcd (https://www.waveshare.com/w/upload/3/36/Image2Lcd.7z) software to convert a picture to arrays and display them by the functions in the example.
- Datasheet of chips are provided, you can read them for more information.

Resource

Documents

- Schematic (https://www.waveshare.com/w/upload/d/dc/Pico-ResTouch-LCD-2.8_Sch.pdf)
- ST7789 datasheet (https://www.waveshare.com/w/upload/a/ae/ST7789_Datasheet.pdf)
- XPT2046 datasheet (https://www.waveshare.com/w/upload/b/b0/XPT2046.pdf)