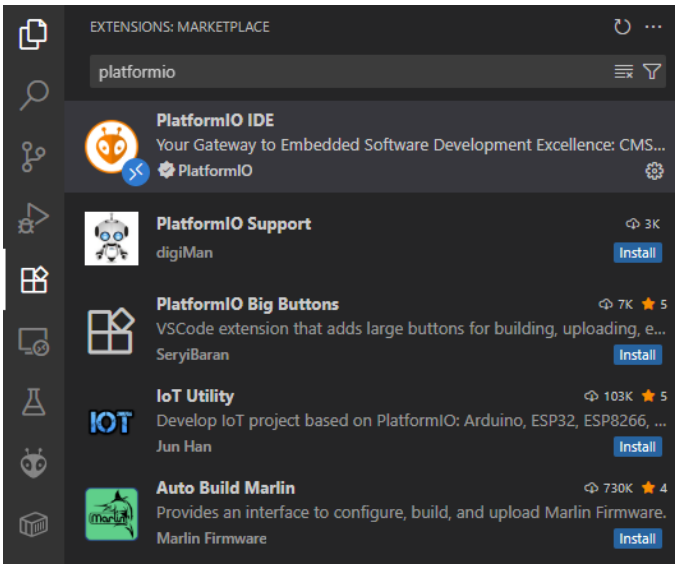
Cheat Sheet for Using Platformio and TFT_eSPI

Installing Platformio

Install VSCode: https://code.visualstudio.com/docs/setup/setup-overview?originUrl=%2Fdocs%2Fsetup%2Fwindows

Add Platformio, using Extension tab on VSCode (ctrl+shift+x):



Options for Adding TFT_eSPI library

The TFT_eSPI library allows you to use the screen on the TTGO (*Note:* These methods should work for most libraries you want to install).

Option 1: Adding TFT_eSPI library to config folder (visible to all projects)

- To add the library, go to the .platformio folder in your home directory (for example, C:\Users\Tiz\.platformio), and create a folder called "lib".
- Download the TFT_eSPI library: https://github.com/Bodmer/TFT_eSPI (Click <> Code -> Download ZIP)
- Restart VSCode (close and open it).
- Now you need to change the board you are using in the header files to set the pin configurations. Go to User_Setup_Select.h
- Comment out the default setup and uncomment: #include <User_Setups/Setup206_LilyGo_T_Display_S3.h>

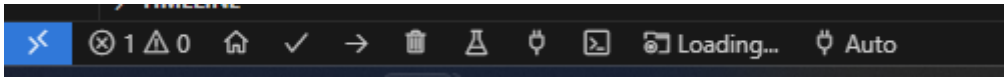
Option 2: Adding TFT_eSPI library using Platformio extension (project specific)

- Go to PIO homepage.
- Go to Libraries tab and search for the TFT_eSPI library.
- Add it to your project.
- Add something like this to your .ini file:

```
[env:lilygo-t-display-s3]
platform = espressif32
board = lilygo-t-display-s3
framework = arduino
board_build.mcu = esp32s3
upload_protocol = esptool
lib_deps = bodmer/TFT_eSPI@^2.5.43
build_flags =
  -D USER_SETUP_LOADED=1
  -include
$PROJECT_LIBDEPS_DIR/$PIOENV/TFT_eSPI/User_Setups/Setup206_LilyGo_T_Display_S3.h
```

Uploading code:

- Use the toolbar at the bottom:
- Build: compile your code and check there are no errors
- Upload: upload the code to the TTGO
- Monitor: if you have any serial outputs you can look at them using the monitor.
- Ensure you have the right COM port selected



Documentation

Official documentation for the board: https://github.com/Xinyuan-LilyGO/T-Display-S3/

You can check the pinout: image -> T-DISPLAY-S3.jpg

Issues when uploading?

Read the FAQ in the T-Display-S3 Github: https://github.com/Xinyuan-LilyGO/T-Display-S3/

- Uploading in boot mode:
  - Hold the boot button (closest to the reset button which is on the side)
  - Push the reset button and let go
  - Release the boot button
  - The ESP32 is now in boot mode, so you should be able to upload code.
  - Press the reset button to exit boot mode after code is uploaded.
- Other options:
  - Close the serial monitor before uploading
  - Close and reopen terminal you have been using

If your code isn't working after it has uploaded and you don't know why

Open the serial monitor and check if the core is crashing and resetting and what error it is printing.

Blocking and Non-Blocking Waits

delay() - the CPU stops and does nothing during the delay which may or may not be useful.

millis() – this function gets the current time since the program started in milliseconds. You can record the current time and check later how much time has elapsed, which doesn't block the CPU and is more similar to using a stopwatch which runs in the background.

TFT_eSPI Printing Strings

When writing new text, the old text is not cleared so it just writes on top

Use either fillScreen or fillRect to clear the screen before writing again. fillRect is better as this way you only "wipe" the part of the screen that changes which helps reduce flickering.

You should also clear and write to the screen *only when necessary*. This will help to avoid flickering as well and is also just good practice anyway.

The absolute best way to prevent flickering is to clear your screen by only setting what you wrote to black. So, if you wrote an integer value, clear it by writing the same integer value in black, and then writing the new integer value in white. More complicated to program but the only way to reliably eliminate all flicker.

There are multiple options for printing text: drawString, print and println.

Debugging

If you want more information here is a detailed guide:
https://piolabs.com/blog/insights/debugging-introduction.html

Add the following lines to the platform.ini file:

```
board_build.mcu = esp32s3
board_build.f_cpu = 240000000L
debug_tool = esp-builtin
upload_protocol = esp-builtin
```

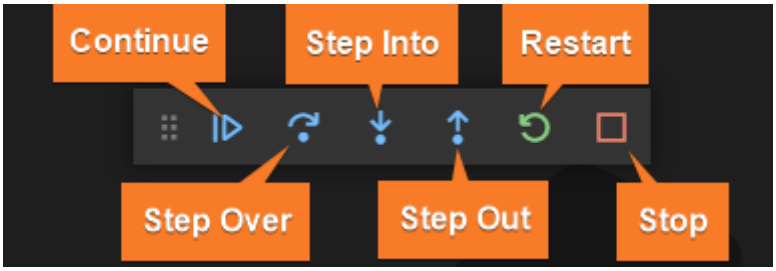
Run using debugger:

At the top, next to run and debug, click on the dropdown. Make sure your project is selected, with PIO Debug written on the left (with no extra options).

Add breakpoints by clicking to the left of the line number.

Run the debugger by clicking the green around next to the drop down at the top left.

These are your buttons to control what the program is doing while debugging:



The program will run as normal until it hits a breakpoint. When it hits a breakpoint, it stops until you decide what happens next by clicking one of the buttons

- Continue. This unpauses, and lets the TTGO continue executing as it normally would. It will stop when it hits the next breakpoint
- Step Over. If the line you have stopped on involves a function call, step over will just run the function and move to the next line (stopping at the next line, over will just run the function and move to the next line (stopping at the next line, regardless if there is a breakpoint).
- Step Into. If the current line involves a function called, this will step into the function and stop on the first line of the function.
- Step Out. If you are in a function, this steps out of it, by executing the remaining code in the function and then stopping on the line that called the function
- Restart. This will reset the Arduino and run the code again from the top.
- Stop. This will stop debugging and the Arduino will carry on executing as it normal does.

Don't forget to comment the lines in the platformio.ini file when you have finished debugging.

You can use print lines for debugging, but the preferred method is using the debugger.