Lab 4 Prep (Learning Platformio)

Wednesday, 20 August 2025 3:45 PM

Study Night

- Go to the study night on Monday!

How to install Platformio and VSCode

- VSCode, you can just Google it and get it from the web
- Platformio, go to the extensions tab (left side) and then search platformio and install it.

How to permanently install a library

Go to your home directory, and navigate to .platformio directory. If you don't have a lib directory yet, and create one. Download the TFT_eSPI library from Github, and add it in here. Make sure you still modify the User Select file to update the pin mappings.

Close and reopen VSCode to force everything to update. Now the library will always be found in every project, you don't need to add it manually to each project.

How to get it to program the Arduino

1. Make a new project
2. Go to the PIO homepage, and then on the left, hit libraries. Make sure you are under the registry tab (at the top)
3. Search the TFT_eSPI library, click on, click add to project and then select your project.
4. Add these build flags to your platform.ini file, under the lib deps:
   build_flags =
   -D USER_SETUP_LOADED=1
   -include $PROJECT_LIBDEPS_DIR/$PIOENV/TFT_eSPI/User_Setups/Setup206_LilyGo_T_Display_S3.h
5. The ESP should now be using the right pins, so you won't get an infinite restart error.
6. Plug the ESP your laptop, make sure port is correct and hit upload. Sometimes you might have to clean before uploading.

Issues when uploading?

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

1. Hold the boot button on the front, the one closest to the side button.
2. Push the side button (reset button).
3. Release the side button and then release the front button.
4. The ESP32 is now in program mode, and you should now be able to program.
5. After programming, push the reset button (side button) to exit program mode.

Close Serial Monitor before uploading (can't upload if communicating over Serial).

Close and reopen the terminal you have been using. Sometimes when you open

Core Crashing

If your code isn't working and you don't know why, check the serial monitor. You may be causing a large issue that causes the ESP32 to crash, forcing it to reboot all the time. If you check the serial monitor, it will print the error and what happened if this is the case.

Documentation

- Below is a link to the official Github:

https://github.com/Xinyuan-LilyGO/T-Display-S3

- This has everything you need, scroll down the readme and FAQ to see some troubleshooting steps that can be useful.
- Navigate to image and then select T-DISPLAY-S3.jpg
  This will show you the pinouts for the board. The number written closest to the board is the number you will use to reference that pin in code.

Blocking and Non Blocking Waits

- Delay() is a blocking wait. Sometimes useful, but mostly not. Blocking means that during the entire duration of the delay, the CPU halts, and does nothing. This is a waste of resource (but sometimes it is what you want).
- Millis(). This function returns the time since the Arduino booted in milliseconds. You record the start time and then you can do other things during your wait. All you do is periodically run an if statement to check if the delay time has elapsed. If it has, you can now handle whatever you need to. If it hasn't, keep doing other stuff. This is non-blocking, so that means the CPU continues to run during the wait, allowing you to do other things in the meantime (which is often very useful).

TFT_eSPI Notes

- When writing new text, the old text is not cleared so it just writes on top
- Use either fill screen or fill rect to clear the screen before writing again. Fill rect 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.

Debugging

- 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
- On the extensions toolbar on the left, select the debugger (it has a right arrow with a bug on the left)
  
- 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 some breakpoints by clicking on the left of the line number.
- Run the debugger by clicking on the green arrow next to the dropdown at the top left.
- The ESP32 may halt right at the start. If you don't understand what you are looking at/this isn't a breakpoint you set, press continue to carry on to the next breakpoint.
- 
  - 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.
  - The first button on the left is continue. This unpauses, and lets the Arduino continue executing as it normally would. It will stop when it hits the next breakpoint.
  - The second button is 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, regardless if there is a breakpoint).
    - The third button is 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.
    - The fourth button is 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.
    - The fifth button is restart. This will reset the Arduino and run the code again from the top.
    - The sixth button is 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.**
- This is the preferred method for debugging. It is significantly better than using print lines, as it is faster to use, provides more information than printing does and is