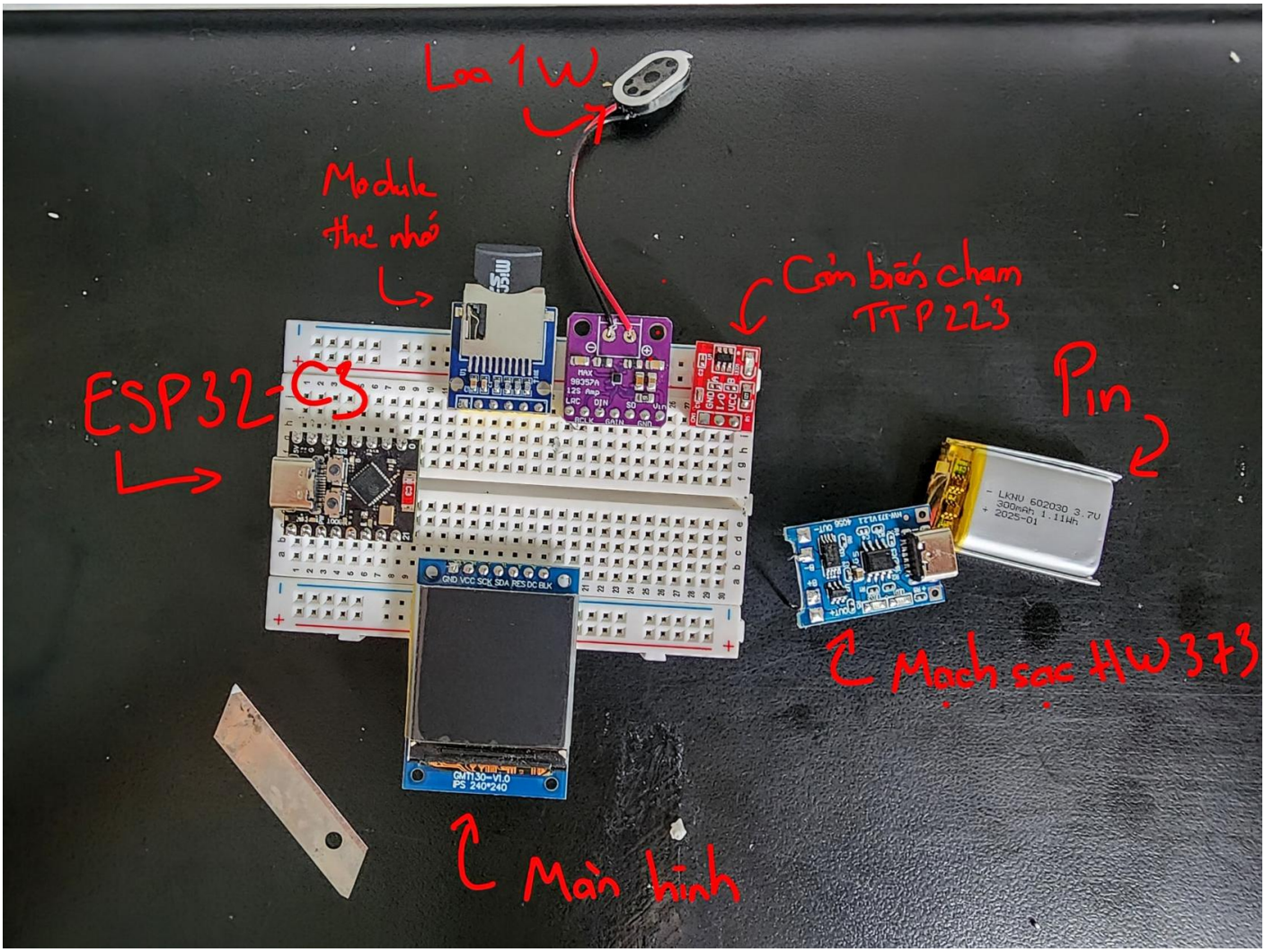
ESP32 - The Cube

- Một dự án "out of the box" - dự án đầu tiên trong hành trình tìm hiểu về **lập trình nhúng (embedded programming)**
- Dự án đầu tiên này sẽ phát triển một pet robot để bàn tương tự như Dasai Mochi Robot
- Tính năng chính:
  - Hiển thị biểu cảm ngẫu nhiên
  - Hiển thị biểu cảm khi tương tác chạm
  - Phát âm thanh, nhạc
  - Tương tác với GenAI bằng giọng nói (cập nhật sau)

- Chuẩn bị
- Chương 1: Phần cứng
  - Phần 1: Những mối hàn đầu tiên
  - Phần 2: "Hello World!!!"
  - Phần 3: Màn hình
  - Phần 4: Thẻ nhớ

Chuẩn bị



- Linh kiện cần thiết
- **ESP32-C3 SuperMini** (ESP32-C3 SuperMini datasheet.pdf). Ban đầu linh kiện này được sử dụng do **nhỏ gọn** và **đủ mạnh** để có thể nhét vào trong khung in 3D.
- Module thẻ nhớ **TF V474** và thẻ nhớ **MicroSD 2GB**. Dùng để lưu trữ dữ liệu vì bộ nhớ trong của ESP32 C3 là rất nhỏ (4MB).
- Module khuếch đại **MAX98357A** và **loa 4ohm 3W**. Chắc chắn là dùng để phát âm thanh rồi. Nhưng cần dùng **MAX98357A** vì tín hiệu điện đầu ra của ESP32 là rất nhỏ không đủ để phát âm thanh qua loa.
- Pin 3.7v và mạch sạc **HW-373.** Chắc chắn là để cấp nguồn chứ còn làm gì nữa.
- Cảm biến chạm **TTP223**. Để người dùng có thể tương tác với robot (pet mà :)))))
- Màn hình **TFT 1.3in 240*240 IC ST7789VW.** Để hiển thị biểu cảm của robot. Ban đầu sử dụng màn hình OLED đơn sắc 0.96in nhưng gặp 2 vấn đề:
  - Giao tiếp I2C tốc độ chậm hiển thị hình ảnh không được mượt.
  - Màn hình đơn sắc hiển thị hình ảnh không có chi tiết, xấu.

Phần mềm cần thiết

- **VSCode** và plugin **PlatformIO.** Chắc chắn là dùng để lập trình rồi.
  - VSCode được sử dụng do mình đã quen dùng từ trước
  - Mạnh hơn Ardruino IDE. Có gợi ý code...
  - Nhẹ hơn Clion nhưng vẫn đầy đủ tính năng cần thiết

Một vài kiến thức khác với ESP32

Giao ti?p v?i ESP32

Tiêu chí	I2C (Inter-Integrated Circuit)	I2S (Inter-IC Sound)	SPI (Serial Peripheral Interface)	UART (Universal Asynchronous Receiver/Transmitter)
Số dây cần	2 (SDA, SCL)	3 (BCLK, LRCLK/WS, DIN/DOUT) có thể có thêm MCLK	4+ (SCK, MOSI, MISO, CS/SS)	2 (TX/RX)
Kiểu giao tiếp	Đồng bộ (có clock)	Đồng bộ (có clock)	Đồng bộ (có clock)	Không đồng bộ (không cần clock)
Số thiết bị	+++ (thiết bị có địa chỉ riêng)	1 master vs 1 slave hoặc 1 master nhiều slave với thiết bị chia	+++ (mỗi thiết bị cần một dây CS riêng)	1 master vs 1 slave
Tốc độ	Thấp (100kHz - vài MHz)	Cao (vài MHz tùy cấu hình)	Rất cao (vài MHz đến vài chục MHz)	Thấp (9.6kbps - vài Mbps)
Độ phức tạp	Ít dây => dễ nối	Phức tạp, cần cấu hình và định dạng dữ liệu đúng	Phức tạp, tốn chân GPIO	Ít dây => dễ nối
Ưu điểm	Tiết kiệm chân Dễ thêm thiết bị	Chuẩn hóa cho audio Chất lượng cao Hỗ trợ DMA	Tốc độ cao Truyền được dữ liệu lớn	Đơn giản
Nhược điểm	Tốc độ hạn chế Dễ xung đột	Chỉ dùng cho Audio Khó quản lý với buffer audio lớn Dễ nhiễu trên dây dài	Thiết kế phức tạp Tốn chân	Chỉ có thể kết nối 1 - 1
Ứng dụng	Cảm biến, màn hình OLED nhỏ....	Xử lý âm thanh (thu, phát, truyền...)	Màn hình TFT, thẻ nhớ,...	Giao tiếp với PC, GPS, GSM...

Các chân trên ESP32

Trên ESP32 có nhiều chân sẽ được dùng chung với nhiều thiết bị khác nhau. Nên cân nhắc để hàn chân sao cho hợp lý.

Dùng chung cho t?t c? các module

Tất cả các module đều cần sử dụng đến nguồn (có thể trên mạch ESP32 hoặc nguồn cấp ngoài). Cần cân nhắc để nối chung hoặc chọn dây nguồn phù hợp.

- 5V
- 3.3V
- GND

Giao ti?p

Các chân giao tiếp này nằm chung với các chân GPIO có đánh số. Cần check datasheet để biết chính xác nó nằm ở đâu

- I2C (SDA, SCL)
- SPI (SCK. MOSI, MISO, CS/SS)
- UART (RX/TX)

Chương 1: Phần cứng

Phần 1: Những mối hàn đầu tiên

Bắt đầu luôn với chút kinh nghiệm được rút ra từ đau thương.

- Nên đọc kỹ **thông tin**, tìm **datasheet** với mỗi linh kiện để đảm bảo có thể phục vụ nhu cầu và tính tương thích với ESP32 cũng như những linh kiện khác.
- ESP32 sẽ có nhiều chân được dùng chung với nhau 3.3V, 5V, GND, I2C, SPI... chần chú ý những chân này để hàm dây cho phù hợp.
- Nếu có thể nên sử dụng **pin header** trên bo mạch kết hợp với **breadboard** và các **jumper wire** để thuận tiện linh hoạt trong quá trình phát triển
- Trên mạch ESP32 có những linh kiện rất nhỏ rất nhạy cảm, nếu quá nhiệt rất dễ bong ra (đã chết một chiếc ESP32-C3-SuperMini vì lý do này T.T) vậy nên **hết sức cẩn thận khi hàn.**

Phần 2: "Hello World!!!"

Giống như lập trình phần mềm. Lập trình nhúng cũng có những bài "Hello world!!!".

Trong phần này ngoài thực hiện log lên màn hình dòng "Hello world!!!" chúng ta sẽ thêm một chút liên quan đến phần cứng là điều khiển đèn led trên ESP32

Cấu hình Dự án

Để cấu hình chúng ta sử dụng file `platformio.ini`

```
[env:esp32-c3-devkitm-1]
platform = espressif32                    ; Khai báo n?n t?ng s? d?ng
board = esp32-c3-devkitm-1                ; Khai báo board, thay ??i v?i nh?ng board khác nhau
framework = arduino                       ; Khai báo framework
lib_deps =                                ; Khai báo th? vi?n
 SPI                                    ;  ?i "hello world" thì ch?a c?n ??n th? vi?n nào
 SD                                     ;  có th? xóa b? lib_deps n?u ch?a dùng ??n
monitor_speed = 115200                    ; Khai báo t?c ?? giao ti?p Serial Monitor
build_flags =                             ; T??ng t? nh? Scripting Define Symbols trong Unity
 -D ARDUINO_USB_MODE=1                  ;  ?n 2 báo 2 tr??ng t??ng t? nh? trong ví d?
 -D ARDUINO_USB_CDC_ON_BOOT=1           ; ?? có th? xem ???c log
```

Các hàm cơ bản

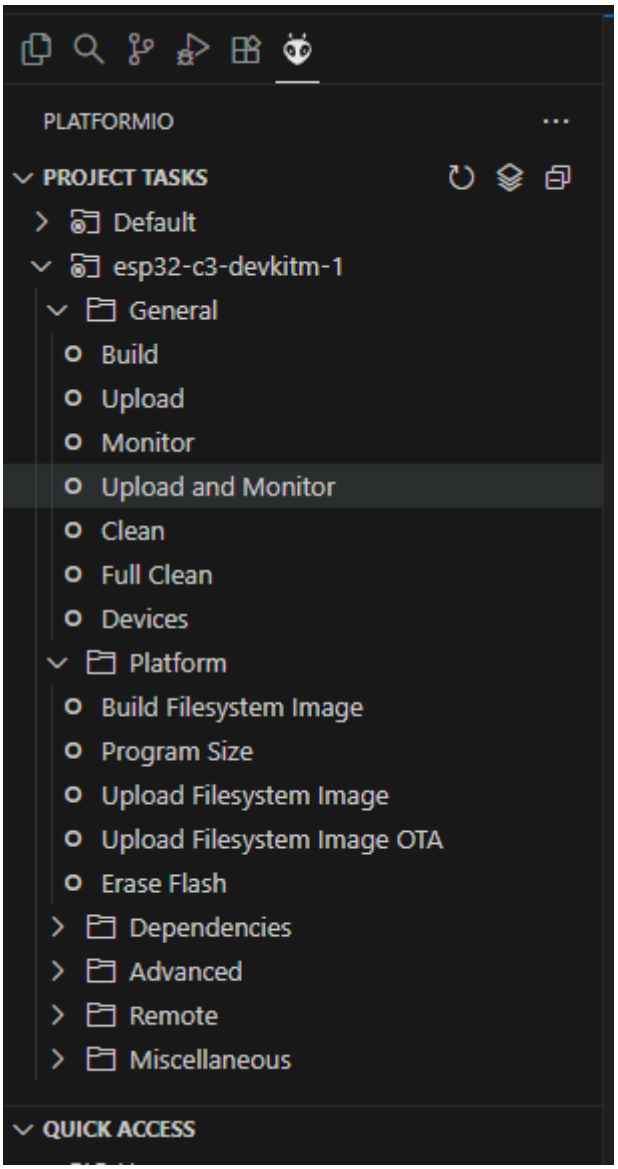
```
// M?t file main.cpp c? b?n khi làm vi?c v?i ESP32

#include <Arduino.h> // Khai báo th? vi?n

// T??ng t? nh? Awake trong Unity.
// Hàm này ???c g?i 1 l?n khi ESP32 kh?i ??ng.
// Dùng ?? kh?i t?o, setup...
void setup()
{
}

// T??ng t? nh? hàm Update trong Unity
// L?p l?i m?i khi các l?nh bên trong loop ???c th?c hi?n h?t
void loop()
{
}
```

"Hello World!!!"

```
#include <Arduino.h>

// Khai báo chân giao ti?p v?i ?èn led
const int LED_PIN = 8;

void setup()
{
  // Setup chân led là output
  pinMode(LED_PIN, OUTPUT);

  // Kh?i t?o serial có t?c ?? trùng v?i t?c ?? ?ã khai báo ? platformio.ini
  Serial.begin(115200);
  // Ch? ??n khi serial kh?i t?o xong ho?c h?t th?i gian ch?
  unsigned long start = millis();
  while (!Serial && millis() - start < 2000)
  {
    delay(1);
  }

  // Delay 1s ?? c?a s? Serial Monitor k?p b?t lên và nhìn th?y log
  // N?u không có thì code v?n ch?y ???c nh?ng c?a s? Serial Monitor b?t lên sau s? không nhìn th?y log
  delay(1000);

  // Hello world ?ây r?i =)))
  Serial.println("Hello world!!!");
}

void loop()
{
  digitalWrite(LED_PIN, LOW);   // B?t ?èn LED
  delay(100);                   // Ch? 0.1 giây
  digitalWrite(LED_PIN, HIGH);  // T?t ?èn LED
  delay(100);                   // Ch? 0.1 giây
}
```

Sau khi code xong thì thực hiện

- Kết nối ESP32 với PC bằng cáp USB
- Upload and Monitor (trong VSCode > tab PlatformIO trên Activity Bar > Upload and Monitor)



Chờ đến khi code được nạp xong. Nhìn cửa sổ terminal và mong chờ kết quả:

- Dòng text "Hello world!!!" được in ra 1 lần
- Trên board ESP32 led xanh nháy liên tục

Hành trình fix bug đầu tiên

V?n ??

Mặc dù đã nạp code thành công. Đèn nguồn có sáng nhưng tôi không nhận được bất kỳ dòng log nào dù có viết `Serial.println("Hello world!!!");` ở `setup()` hay `loop()`

Quá trình debug

Ki?m tra chip

Để đảm bảo ESP32 này không chết trong lúc mình hàn. Nên mình đã dùng code điều khiển led để kiểm tra xem nó có còn hoạt động bình thường không.

Rất mừng là nó còn sống. Trước đấy vừa chết một con xong (T.T)

=> Vấn đề có thể do giao tiếp Serial qua USB

Ki?m tra các l?i c?u hình

Đảm bảo tốc độ `monitor_speed = 115200` là trùng nhau trong `platformio.ini` và `main.cpp`

Thử delay trước khi khởi tạo `Serial.begin(115200);`

Nhưng vấn đề vẫn không được giải quyết.

=> Có thể vấn đề do driver

Ki?m tra các môi tr??ng khác

Chuyển sang macOS và sử dụng CLion với PlatformIO => Vẫn không được!!!

Dùng Ardruino IDE để code và nạp code => Code đã chạy và monitor đã hiện log.

=> Vấn đề có thể do cấu hình

M?u ch?t

So sánh 2 cách cấu hình thì trên Ardruino IDE có `USB CDC On Boot: "Enabled"`.

Tương tự thêm `build_flags -D ARDUINO_USB_CDC_ON_BOOT=1` vào `platformio.ini` thì sẽ gặp lỗi biên dịch của Serial. Cần phải thêm cả cờ `-D ARDUINO_USB_MODE=1`.

Lúc này vấn đề đã được giải quyết.

Gi?i thích

Mạch ESP32 - C3 SuperMini đang dùng sử dụng Native USB. Nghĩa là con chip sẽ tự mình xử lú giao tiếp USB. Để tính năng này hoạt động đúng thì trình biên dịch cần thực hiện 2 việc:

- `-D ARDUINO_USB_MODE=1`: Chuyển chế độ hoạt động của cổng USB sang chế độ CDC
- `-D ARDUINO_USB_CDC_ON_BOOT=1`: Kích hoạt chế độ CDC ngay khi khởi động.

Phần 3: Màn hình

Ok, đầu tiên mình muốn chọn lắp màn hình trước.

Màn hình là phần giúp hiển thị trực quan nhất. Mình có thể thấy được thành quả của mình hiển thị ngay trên màn hình chứ không phải chỉ ở mỗi `Serial.print`

Màn hình cũng giúp mình kiểm tra được những lỗi phần cứng mà chỉ dùng `Serial.print` thì không thể tìm ra hết được.

Lắp đặt

Màn hình	ESP32-C3	Chắc năng	Lý do	Ghi Chú
VCC	**3.3V** hoặc 5V	Cấp nguồn cho màn hình.	Nguồn vào từ USB qua ESP32 có thể đẩy ra thành 2 nguồn 3.3V hoặc 5V.	Với những thiết bị nhẹ điện. Có thể sử dụng nguồn từ ESP32. Chọn nguồn phù hợp với thiết bị
GND	GND	Nối đất	Tạo một điểm tham chiếu chung cho toàn bộ thiết bị trên hệ thống	Toàn bộ thiết bị đều cần kết nối GND
SCK	GPIO4	Clock (SPI)	Màn hình sử dụng giao thức SPI nên cần sử dụng chân clock GPIO4 là chân clock của mạch ESP32-C3 này	Chân **clock** có thể thay đổi với những board mạch khác nhau. Kiểm tra datasheet để biết thêm chi tiết.
SDA	GPIO6	Data out (MOSI)	Màn hình nhận dữ liệu từ cổng ra của ESP32	MOSI cũng là một chân đặc biệt trên ESP32. Chú ý kiểm tra datasheet
RES	GPIO1	Reset	Là chân điều khiển không yêu cầu tốc độ cao. Chọn bất kỳ chân GPIO còn trống thuận tiện cho đi dây	Tránh chọn các chân đặc biệt Cần chú ý khai báo chính xác trong code
DC	GPIO2	Data/Command	Tương tự như RES. Có thể chọn chân GPIO thuận tiện	

BLK	GPIO3	Điều khiển nền của màn hình	Tương tự như RES và DC. Có thể chọn chân thuận tiện cho việc đi dây	

Kiểm tra

Cài ??t th? vi?n TFT_eSPI

> Arduino and PlatformIO IDE compatible TFT library optimised for the Raspberry Pi Pico (RP2040), STM32, ESP8266 and ESP32 that supports different driver chips

Để thêm thư viện chúng ta có thể thực hiện 1 trong 2 cách sau:

- Truy cập PIO Home → Library → Search TFT_eSPI và cài đặt library TFT_eSPI của Bodmer
- Hoặc thêm `bodmer/TFT_eSPI` vào phần `lib_deps` trong file `platformio.ini`

Khi đó file `platformio.ini` sẽ trông như thế này

```
[env:esp32-c3-devkitm-1]
platform = platformio/espressif32 @ 6.6.0
board = esp32-c3-devkitm-1
framework = arduino
platform_packages =
  framework-arduinoespressif32 @ ~3.20014.0
lib_deps =
 SPI
 bodmer/TFT_eSPI@^2.5.43
monitor_speed = 115200
build_flags =
 -D ARDUINO_USB_MODE=1
 -D ARDUINO_USB_CDC_ON_BOOT=1
```

Tiếp theo cần cấu hình thư viện để nó tương thích với đúng phần cứng.

Truy cập `.pio ? libdeps ? esp32-c3-devkitm-1 ? TFT_eSPI ? User_Setup.h` và thực hiện config tương ứng với cấu hình phần cứng.

```
// Driver c?a màn hình
#define ST7789_DRIVER
#define ESP32_C3_DMA_ERROR_WORKAROUND
#define TFT_RGB_ORDER TFT_RGB // N?u màu in ra b? sai Blue và Red thì ??i l?i thành TFT_BGR

// Kích th??c c?a màn hình
#define TFT_WIDTH 240
#define TFT_HEIGHT 240

// Các chân k?t n?i ?ang ???c s? d?ng
#define TFT_MOSI 6
#define TFT_SCLK 4
#define TFT_CS   -1
```

```
#define TFT_DC   2
#define TFT_RST  1
#define TFT_BL   3

// Tín hi?u b?t/t?t ?èn n?n
#define TFT_BACKLIGHT_ON HIGH

// Các font s? ???c compile
#define LOAD_GLCD    // Font 1 (built-in), required for basic text
#define LOAD_FONT2   // Small 16 px font
#define LOAD_FONT4   // Medium 26 px font
#define LOAD_FONT6
#define LOAD_FONT7
#define LOAD_FONT8
#define LOAD_GFXFF  // FreeFonts (optional)
#define SMOOTH_FONT // Anti-aliased fonts (needs FS)

// T?c ?? SPI
#define SPI_FREQUENCY       40000000
#define SPI_READ_FREQUENCY  20000000
```

> Cách này có điểm yếu là sẽ bị **reset** khi `Full Clean` hoặc `Update/Re-install` thư viện. Nhưng trong quá trình test các các setup khác đều gặp lỗi bootloop do config sai. Nên hiện tại vẫn sử dụng cách này để setup.

"Hello World!"

Tiếp theo để test xem màn hình đã được setup đúng và phần cứng còn hoạt động tốt hay không. Chúng ta sẽ thử với đoạn code hiển thị "Hello world!" trên màn hình.

Truy cập `src ? main.cpp` và sử dụng đoạn code dưới đây:

```
#include <Arduino.h>
#include <TFT_eSPI.h>

// T?o ??i t??ng tft t? th? vi?n
TFT_eSPI tft = TFT_eSPI();

void setup() {
  tft.init(); // Kh?i t?i tft controller

  tft.setRotation(0); // Set h??ng xoay màn hình // S? d?ng các tham s? 0, 1, 2, 3
  tft.fillScreen(TFT_BLACK); // Fill toàn b? màn hình b?ng màu ?en
  tft.setTextColor(TFT_GREEN); // Set màu ch?
  tft.setTextSize(2); // Set c? ch?
  tft.setCursor(0, 0); // Set t?a ?? ?i?m b?t ??u vi?t ch?
  tft.println("Hello World!"); // Hi?n th? ch?
}

void loop() {}
```

Sau đó thực hiện Upload. Quá trình Upload có thể có warning (console màu vàng) nhưng không ảnh hưởng gì.

Sau khi upload xong kiểm tra màn hình nếu hiển thị như hình dưới là đã chạy **thành công** sẵn sàng cho bước tiếp theo thôi.

Trong trường hợp chưa hiển thị được "Hello World!" trên màn hình hãy kiểm tra cửa sổ terminal trên VSCode:

- Nếu **có** log chạy liên tục thì rất có thể bạn bị bootloop.
- Nếu **không** có hãy nhấn nút **Reset** trên ESP32 để ép ESP32 khởi động lại.

Nếu vẫn không được thực hiện check list sau:

- Kiểm tra các chân kết nối đảm bảo không chân nào bị lỏng.
- Kiểm tra lại file `User_Setup.h` đảm bảo đã config đúng file, đúng tham số.
- Đảm bảo đã khai báo đúng và đủ file `platformio.ini`
- Thực hiện Clean và Upload lại.

Nếu vẫn không được nữa. Thì nên thử với một cái màn hình khác để đảm bảo phần cứng không bị hỏng.

Phần 4: Thẻ nhớ

Ok tiếp theo là lắp đặt thẻ nhớ.

Trong dự án này thẻ nhớ sẽ là nơi lưu trữ dữ liệu chính như hình gif, audio...

Trước khi bắt đầu thì có một điều cần chú ý

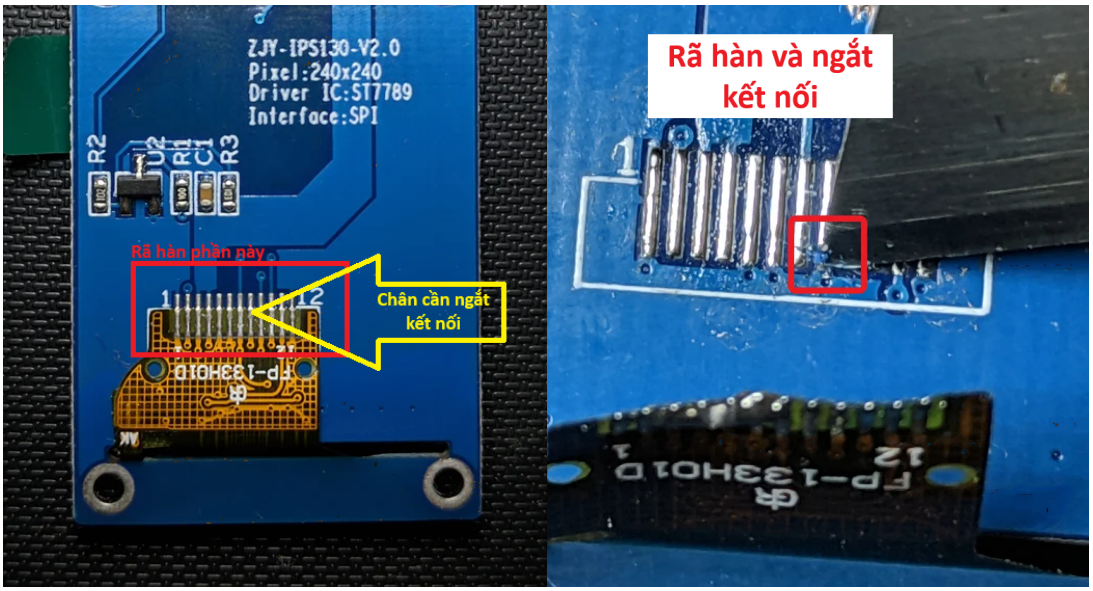
Xử lý conflict phần cứng

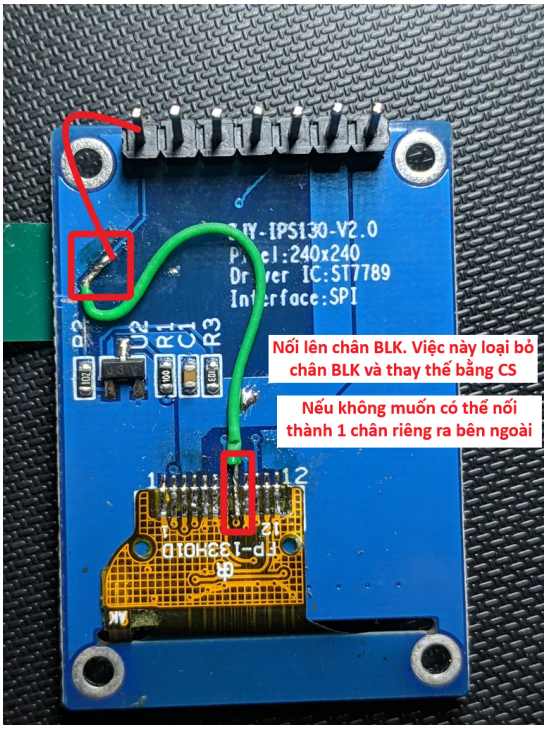
Thẻ nhớ sử dụng giao tiếp SPI. Với ESP32 khi sử dụng nhiều thiết bị giao tiếp chung SPI thì cần có chân CS để điều khiển.

Trong trường hợp hiện tại chúng ta có 2 thiết bị giao tiếp SPI là thẻ nhớ và màn hình. Khi cần giao tiếp với thẻ nhớ thì tín hiệu CS trên thẻ nhớ được bật và tắt đối với màn hình.

Thế nhưng trong phần trước. Màn hình chúng ta sử dụng có 7 pin và không có chân CS. Điều này là do chân CS được nhà sản xuất nối trực tiếp với GND. Tín hiệu CS trên màn hình sẽ luôn ở trạng thái bật. Vì vậy chúng ta có 2 hướng giải quyết:

- Kiểu người có tiền: Thay màn mới cho đẹp
- Kiểu người hết tiền chúng ta có những bước sau:
  - Xác định chân CS đã bị hàn với GND trên màn hình. Là chân thứ 8 trên dây nối màn hình với PCB
  - Rã hàn để thấy được phần kết nối CS với GND
  - Ngắt kết nối của chân CS với GND bằng cách cắt qua lớp đồng trên PCB
  - Hàn lại dây nối màn hình với PCB
  - Hàn chân CS ra một chân mới và kết nối với ESP32.
  - Trong phần khai báo thư viện TFT_eSPI phần `#define TFT_CS -1` sửa thành `#define TFT_CS  <pin ?ã k?t n?i trên ESP32>`





Ok vậy là xong. Đừng làm chết màn hình là được :)))

Lắp đặt

Màn hình	ESP32-C3	Chắc năng	Ghi chú
VCC (3.3)	**3.3V** hoặc 5V	Cấp nguồn cho màn hình.	Nguồn vào từ USB qua ESP32 có thể đẩy ra thành 2 nguồn 3.3V hoặc 5V.

GND	GND	Nối đất	Tạo một điểm tham chiếu chung cho toàn bộ thiết bị trên hệ thống
SCK	GPIO4	Clock (SPI)	Dùng chung với Màn hình
MOSI	GPIO6	Data out	Dùng chung với Màn hình
MISO	GPIO5	Data in	Dùng chân chuyên dụng của ESP32
CS	GPIO7	Điều khiển khi giao tiếp SPI	Chọn bất kỳ chân GPIO còn trống thuận tiện cho đi dây

Kiểm tra

`platformio.ini`

Trong phần thiết lập này chúng ta cần khai báo thêm thư viện SD trong `lib_deps`

```
[env:esp32-c3-devkitm-1]
platform = platformio/espressif32 @ 6.6.0
board = esp32-c3-devkitm-1
framework = arduino
platform_packages =
  framework-arduinoespressif32 @ ~3.20014.0
lib_deps =
 SD // Khai báo thêm th? vi?n này
 SPI
 bodmer/TFT_eSPI@^2.5.43
monitor_speed = 115200
build_flags =
 -D ARDUINO_USB_MODE=1
 -D ARDUINO_USB_CDC_ON_BOOT=1
```

Code and Test

```
#include <Arduino.h>
#include <TFT_eSPI.h>
#include <SD.h> // Thêm khai báo th? vi?n

// T?o ??i t??ng tft t? th? vi?n // ?? hi?n th? k?t qu? test
TFT_eSPI tft = TFT_eSPI();

// Khai báo ph?n c?ng cho SDCard
const int SDCARD_PIN_CS = 7;
const int SDCARD_PIN_SCK = 4;
const int SDCARD_PIN_MISO = 5;
const int SDCARD_PIN_MOSI = 6;

// Khai báo hàm
void initTFT(); // Kh?i t?o màn hình
void initSD(); // Kh?i t?o SDCard
void listDir(fs::FS &fs, const char * dirname, uint8_t levels); // Hàm hi?n th? các t?p trong SD // Dùng
?? test ch?a c?n quan tâm nhi?u

void setup() {
  initTFT();
```

```
  initSD();
}

int count = 0;
void loop() {}

void initTFT()
{
  tft.init();
  tft.setRotation(0);
  tft.fillScreen(TFT_BLACK);
  tft.setTextColor(TFT_GREEN);
  tft.setTextSize(1);
  tft.setCursor(0, 0);
  tft.println("Init TFT success!!!");
}

void initSD()
{
  SPI.end();
  SPI.begin(SDCARD_PIN_SCK, SDCARD_PIN_MISO, SDCARD_PIN_MOSI, SDCARD_PIN_CS); // Kh?i t?o giao ti?p SPI

  tft.print("Initializing SD card using CS pin: ");
  tft.println(SDCARD_PIN_CS);

  if (!SD.begin(SDCARD_PIN_CS, SPI)) {
    tft.setTextColor(TFT_RED);
    tft.println("SD mount failed (wiring/freq).");
    return;
  }

  tft.setTextColor(TFT_GREEN);
  tft.println("SD card initialized.");

  uint8_t type = SD.cardType();
  if (type == CARD_NONE) {
    tft.setTextColor(TFT_RED);
    tft.println("No SD card detected.");
    return;
  }
  tft.printf("Type: %s\n", type == CARD_MMC ? "MMC" :
                           type == CARD_SD  ? "SDSC" :
                           type == CARD_SDHC? "SDHC/SDXC" : "UNKNOWN");
  tft.printf("Size: %llu MB\n", SD.cardSize() / (1024ULL * 1024ULL));

  tft.println("Filesystem Content:");
  listDir(SD, "/", 0);
  tft.println("\n--- Test Complete ---");
}

// Hàm li?t kê n?i dung th? m?c (l?y t? ví d? c?a th? vi?n SD)
void listDir(fs::FS &fs, const char * dirname, uint8_t levels) {
  tft.printf("Listing directory: %s\n", dirname);

  File root = fs.open(dirname);
  if (!root) {
    tft.println("Failed to open directory");
    return;
  }
  if (!root.isDirectory()) {
    tft.println("Not a directory");
    return;
  }

  File file = root.openNextFile();
  while (file) {
```

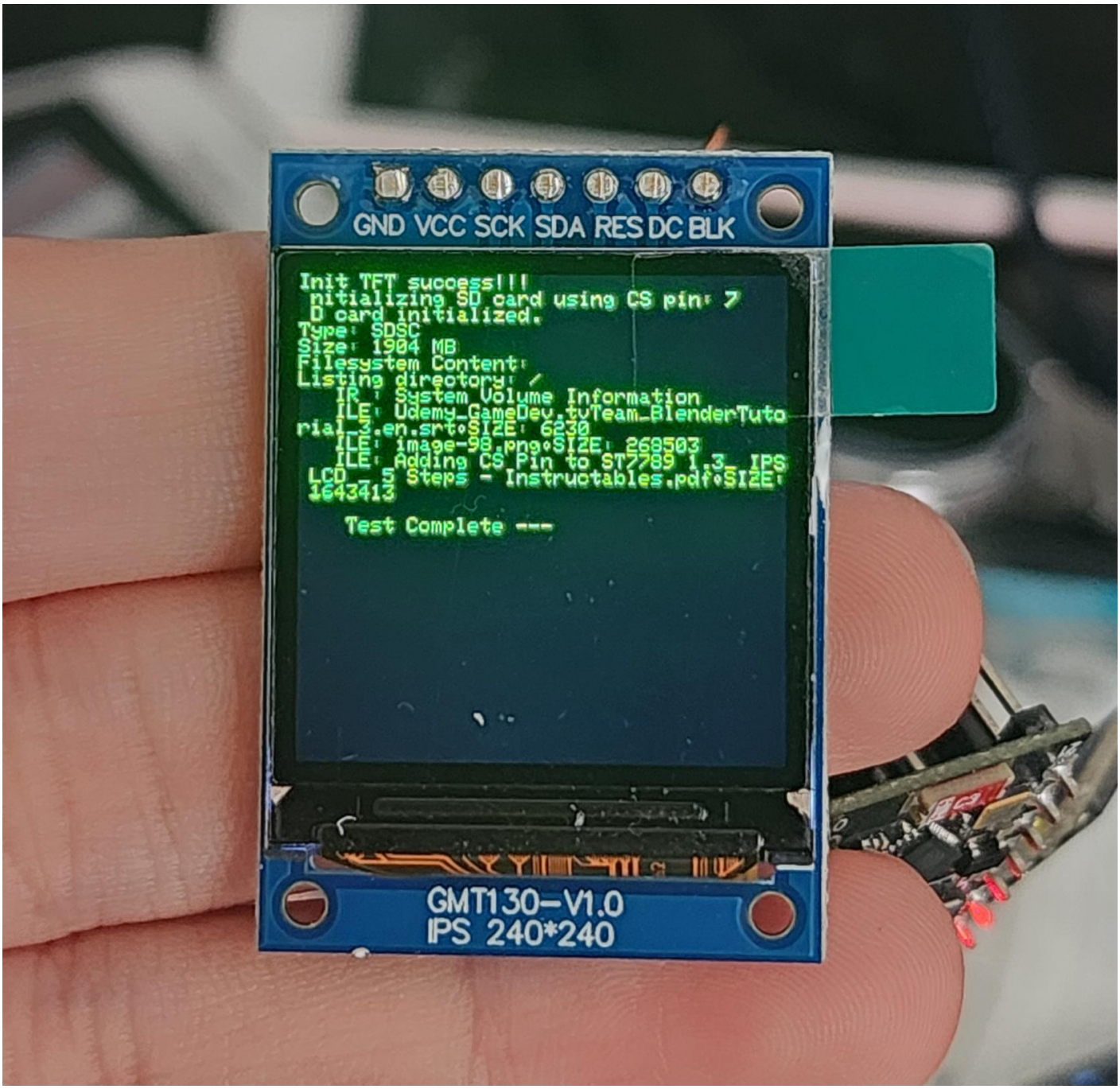
```
    if (file.isDirectory()) {
      tft.print("  DIR : ");
      tft.println(file.name());
      if (levels) {
        listDir(fs, file.path(), levels - 1);
      }
    } else {
      tft.print("  FILE: ");
      tft.print(file.name());
      tft.print("\tSIZE: ");
      tft.println(file.size());
    }
    file = root.openNextFile();
  }
}
```

Upload and Monitor

Kết quả hiển thị như hình dưới (hoặc tương tự toàn màu green) là OK

Nếu chỉ hiển thị đến "Initializing SD card using CS pin: " thì cần check thêm terminal để kiểm tra lỗi

- Có thể conflict phần cứng nhắc tới ở trên chưa được xử lý
- Thẻ nhớ chưa được format đúng (ESP32)
- Lỗi thẻ nhớ hoặc module thẻ nhớ


GND VCC SCK SDA RES DC BLK
Init TFT success!!!
nitializing SD card using CS pin: 7
D card initialized.
Type: SDSC
Size: 1904 MB
Filesystem Content:
Listing directory: /
IR : System Volume Information
ILE: Udemy_GameDev.tvTeam_BlenderTuto
rial_3.en.srt SIZE: 6230
ILE: image-98.png SIZE: 268503
ILE: Adding CS Pin to ST7789 1.3_ IPS
LCD_ 5 Steps - Instructables.pdf SIZE:
1643413
Test Complete ---
GMT130-V1.0
IPS 240*240