**Tên thành viên:**

22146412 Võ Gia Thuận

22146424 Hà Nhật Triều

22146405 Vũ Văn Thắng

22146395 Nguyễn Quốc Tài

**DRIVER TCS34725 DOCUMENTATION**

**1. Giới thiệu chung**

- Cảm biến màu TCS34725 là một loại cảm biến đo màu sắc tiên tiến, được sản xuất bởi AMS (trước đây là TAOS).

- Nó có khả năng đo màu sắc môi trường xung quanh bằng cách tách ánh sáng thành các thành phần Red (Đỏ), Green (Xanh lá), Blue (Xanh dương) và Clear (ánh sáng trắng tổng hợp).

- Nhận diện màu trong robot và thiết bị tự động.

- Điều chỉnh cân bằng trắng cho máy ảnh.

**2. Phần cứng yêu cầu**

- Cảm biến màu TCS34725

- Chuẩn giao tiếp I2C: Module yêu cầu hệ thống hỗ trợ giao tiếp I2C, với khả năng truy cập tới địa chỉ thiết

bị của TCS34725 (thường là 0x29). Các đường tín hiệu cần thiết gồm:

+ SDA (Serial Data Line)

+ SCL (Serial Clock Line)

- Vi xử lý / máy chủ nhúng hỗ trợ I2C: Raspberry Pi (tất cả các phiên bản đều có giao tiếp I2C tích hợp)

- Dây nối và nguồn điện TCS34725 thường hoạt động ở mức điện áp 3.3V hoặc 5V, cần đảm bảo nguồn cung ổn định để cảm biến hoạt động chính xác.

**3. Phần mềm**

Để phát triển, biên dịch và vận hành driver kernel cho TCS34725, hệ thống cần đáp ứng các yêu cầu phần mềm sau:

- Hệ điều hành:

Linux Kernel 5.x hoặc mới hơn (có hỗ trợ cơ chế driver module và giao tiếp I2C).

Đã cấu hình và bật hỗ trợ cho I2C(CONFIG\_I2C,CONFIG\_I2C\_CHARDEV yrong kernel config).

Hệ điều hành có thể là:

Raspberry Pi OS / Raspbian

Debian/Ubuntu x86

**4. Cấu trúc file và tổ chức driver**

**tcs34725\_driver.c**

Cài đặt toàn bộ logic điều khiển cảm biến màu TCS34725.

Định nghĩa các hàm như: probe, remove, open, release, ioctl, init\_sensor, read\_color, set\_gain, read\_status.

Đăng ký thiết bị ký tự (miscdevice) để tạo node /dev/tcs34725.

Tương tác trực tiếp với phần cứng qua giao tiếp I2C.

**test\_tcs34725.c**

Chức năng: Là chương trình ví dụ mẫu minh họa cách giao tiếp với driver qua file /dev/tcs34725.

Sử dụng hàm open(), ioctl(), và close() để đọc dữ liệu màu (R, G, B, C), kiểm tra trạng thái dữ liệu và thiết lập gain.

Xử lý logic ứng dụng như xác định màu chủ đạo và hiển thị thông tin ra terminal.

Dùng các lệnh IOCTL đã định nghĩa trong driver để tương tác.

**Makefile**

Chức năng: Biên dịch file mã nguồn tcs34725\_driver.c thành module kernel .ko.

Sử dụng hạ tầng build sẵn có của kernel tại thư mục /lib/modules/$(uname -r)/build.

Hỗ trợ lệnh make để build và make clean để dọn dẹp.

**5. Danh sách hàm và chức năng trong driver**

\*.static void tcs34725\_init\_sensor(struct i2c\_client \*client);

**Thông số:** struct i2c\_client \*client: Con trỏ tới cấu trúc i2c\_client đại diện cho thiết bị TCS34725.

**Chức năng:** Khởi tạo cảm biến TCS34725. Thiết lập thời gian tích hợp (ATIME), bật nguồn và bật ADC.

**Trả về:** Không trả về giá trị.

\*. static int tcs34725\_read\_color(struct i2c\_client \*client, u8 reg\_low);

**Thông số**:

struct i2c\_client \*client: Con trỏ tới cấu trúc i2c\_client.

u8 reg\_low: Địa chỉ thanh ghi byte thấp của màu cần đọc (RDATAL, GDATAL, BDATAL hoặc CDATAL).

**Chức năng:** Đọc giá trị màu 16-bit (low và high byte) từ cảm biến (R, G, B hoặc Clear).

**Trả về:** Giá trị màu 16-bit nếu thành công, giá trị âm nếu có lỗi.

\*. static int tcs34725\_set\_gain(struct i2c\_client \*client, u8 gain);

**Thông số:**

struct i2c\_client \*client: Con trỏ tới cấu trúc i2c\_client.

u8 gain: Giá trị hệ số khuếch đại cần thiết lập (0–3).

**Chức năng:** Thiết lập hệ số khuếch đại (gain) cho cảm biến TCS34725.

**Trả về:** 0 nếu thành công, giá trị âm nếu lỗi hoặc gain không hợp lệ.

\*. static bool tcs34725\_read\_status(struct i2c\_client \*client);

**Thông số:**

struct i2c\_client \*client: Con trỏ tới cấu trúc i2c\_client.

**Chức năng:** Đọc thanh ghi trạng thái để kiểm tra xem dữ liệu màu đã sẵn sàng chưa (bit AVALID).

**Trả về:** true nếu dữ liệu hợp lệ, false nếu chưa sẵn sàng hoặc lỗi đọc thanh ghi.

\*. static long tcs34725\_ioctl(struct file \*file, unsigned int cmd, unsigned long arg);

**Thông số:**

struct file \*file: Con trỏ tới file cấu trúc của thiết bị.

unsigned int cmd: Mã lệnh IOCTL.

unsigned long arg: Tham số từ người dùng (user space), truyền qua IOCTL.

**Chức năng:** Xử lý các lệnh IOCTL để đọc giá trị R, G, B, C, trạng thái hoặc thiết lập gain cho cảm biến.

**Trả về:** 0 nếu thành công, giá trị âm nếu có lỗi.

\*. static int tcs34725\_open(struct inode \*inodep, struct file \*filep);

**Thông số:**

struct inode \*inodep: Con trỏ tới inode của thiết bị.

struct file \*filep: Con trỏ tới file được mở.

**Chức năng:** Ghi nhận khi thiết bị được mở.

**Trả về**: 0 luôn (không lỗi).

\*. static int tcs34725\_release(struct inode \*inodep, struct file \*filep);

**Thông số:**

struct inode \*inodep: Con trỏ tới inode của thiết bị.

struct file \*filep: Con trỏ tới file được đóng.

**Chức năng:** Ghi nhận khi thiết bị bị đóng.

**Trả về:** 0 luôn (không lỗi).

\*. static int tcs34725\_probe(struct i2c\_client \*client);

**Thông số:**

struct i2c\_client \*client: Con trỏ tới cấu trúc i2c\_client.

**Chức năng:** Hàm khởi tạo driver khi kernel phát hiện thiết bị phù hợp. Gọi hàm tcs34725\_init\_sensor, đăng ký thiết bị ký tự (char device) và tạo node trong /dev.

**Trả về:** 0 nếu thành công, giá trị âm nếu lỗi tạo thiết bị hoặc đăng ký thiết bị ký tự.

\*. static void tcs34725\_remove(struct i2c\_client \*client);

**Thông số:**

struct i2c\_client \*client: Con trỏ tới cấu trúc i2c\_client.

**Chức năng:** Giải phóng tài nguyên khi thiết bị bị ngắt kết nối: xóa device, class, và giải phóng major number.

**Trả về:** Không trả về giá trị.

\*. static int tcs34725\_init(void);

**Chức năng:** Hàm khởi tạo module kernel. Thêm driver TCS34725 vào hệ thống thông qua i2c\_add\_driver.

**Trả về:** 0 nếu thành công, giá trị âm nếu không thêm được driver.

\*. static void tcs34725\_exit(void);

**Chức năng:** Hàm cleanup khi module bị gỡ bỏ. Gọi i2c\_del\_driver để gỡ driver khỏi kernel.

**Trả về:** Không trả về giá trị.

**6. Ví dụ sử dụng driver tcs34725**

Có trong file **test\_tcs34725.c**

**7. Hướng dẫn nạp driver tcs34725 và chạy chương trình**

**Bước 1:** Tải 3 file **tcs34725\_driver.c**, **test\_tcs34725.c**, **Makefile** về máy chủ Raspberry Pi.

**Bước 2:** Mở terminal trong Raspberry Pi và thực hiện lệnh **touch tcs34725-overlay.dts**

**Bước 3:** Chỉnh sửa file dts bằng lệnh **nano** **tcs34725-overlay.dts** và save lại với nội dung như sau:

/dts-v1/;

/plugin/;

/ {

compatible = "brcm,bcm2835";

fragment@0 {

target = <&i2c1>;

\_\_overlay\_\_ {

tcs34725@29 {

compatible = "taos,tcs34725";

reg = <0x29>; // I2C address of TCS34725

status = "okay";

};

};

};

};

**Bước 4**: Chuyển file .dts thành .dtbo với lệnh **sudo dtc -@ -I dts -O dtb tcs34725-overlay.dts -o tcs34725-overlay.dtbo**

**Bước 5:** Copy file .dtbo vào /boot/overlays/ bằng lệnh **sudo cp tcs34725-overlay.dtbo /boot/overlays/**

**Bước 6**: Vào thư mục /boot và chỉnh sửa file config.txt bằng lệnh **sudo nano config.txt**

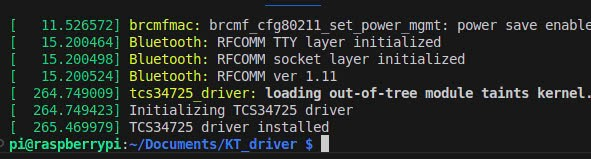
Trong file config.txt, lướt xuống cuối cùng và thêm **dtoverlay = tcs34725-overlay** và save.

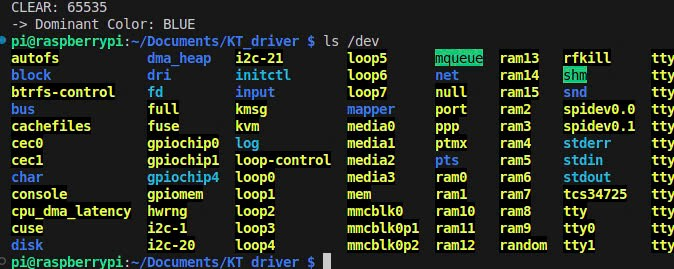
**Bước 7:** Reboot lại raspberry pi bằng lệnh **sudo reboot**

**Bước 8:** Vào thư mục chứa 3 file ở bước 1 sau đó bấm lệnh **make** để chạy file Makefile

**Bước 9:** Cài đặt driver bằng lệnh **sudo insmod tcs34725\_driver.ko,** nếu muốn xóa driver thì thực hiện lệnh **sudo rmmod tcs34725\_driver**

**Bước 10:** Kiểm tra xem driver có cài đặt thành công chưa bằng lệnh **sudo dmesg** nếu hiện trên terminal dòng text “TCS34725 driver installed” là cài đặt driver thành công, để xem thiết bị đã được đăng kí bằng cách thực hiện lệnh **ls /dev** và tìm tên device đã đăng kí có tên là tcs34725, nếu có thì đã đăng kí thiết bị thành công.

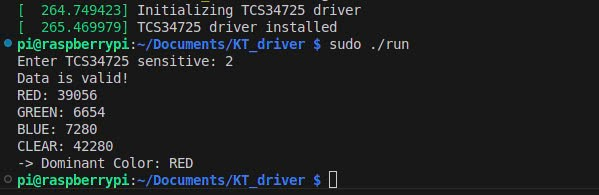




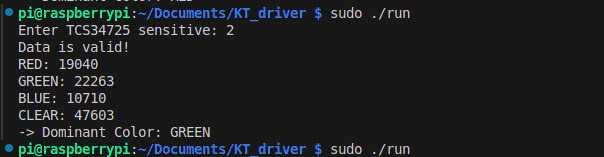
**Bước 11:** Biên dịch chạy file test\_tcs34725.c bằng lệnh **gcc test\_tcs34725.c -o run** và chạy chương trình bằng lệnh **sudo ./run**

**Một số kết quả sau khi chạy chương trình:**

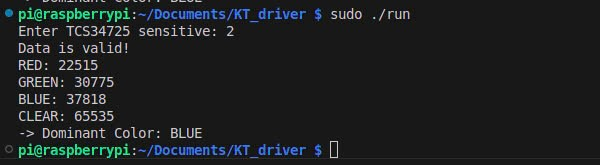
+ Cảm biến khi chiếu vào tấm giấy red:



+ Cảm biến khi chiếu vào tấm giấy green:



+ Cảm biến khi chiếu vào tấm giấy blue:



**Chúc bạn nạp driver ts34725 thành công tốt đẹp!**