

**ĐẠI HỌC QUỐC GIA THÀNH PHỐ HỒ CHÍ MINH  
TRƯỜNG ĐẠI HỌC BÁCH KHOA  
KHOA KHOA HỌC VÀ KỸ THUẬT MÁY TÍNH**



**BÁO CÁO  
ĐỒ ÁN MÔN HỌC KỸ THUẬT MÁY TÍNH  
PHÁT TRIỂN THƯ VIỆN GIAO DIỆN  
CHO MẠCH NHÚNG ESP32  
SỬ DỤNG THƯ VIỆN ĐỒ HỌA LVGL**

Ngành: Kỹ thuật máy tính

**HỘI ĐỒNG: HỘI ĐỒNG X KỸ THUẬT MÁY TÍNH  
GVHD: TS. LÊ TRỌNG NHÂN  
TKHD: TS. LÊ TRỌNG NHÂN**

—o0o—

**SVTH 1: Nguyễn Trọng Vinh (2015070)  
SVTH 2: Lê Thanh Dương (MSSV)  
SVTH 3: Lê Đỗ Minh Thông (MSSV)**

TP. Hồ Chí Minh, 05/2025

# Lời cam đoan

Chúng em xin cam đoan rằng những nội dung trình bày trong báo cáo luận văn này là công trình nghiên cứu của nhóm dưới sự hướng dẫn của thầy Trần Thanh Bình và thầy Lê Trọng Nhân. Nội dung và số liệu trong báo cáo không phải là bản sao chép từ bất kỳ báo cáo, tiểu luận nào có trước. Tất cả những sự giúp đỡ cho việc xây dựng bài báo cáo đều được trích dẫn và ghi nguồn đầy đủ, rõ ràng. Nếu không đúng sự thật, chúng em xin chịu mọi trách nhiệm trước các thầy, cô, và nhà trường.

Thành phố Hồ Chí Minh, tháng 12 năm 2024.

# Lời cảm ơn

Chúng em xin được gửi lời cảm ơn chân thành nhất đến thầy Trần Thanh Bình và thầy Lê Trọng Nhân, giảng viên hướng dẫn trực tiếp đề tài. Hai thầy là người đã theo dõi, cũng như góp ý, sửa chữa những sai sót cho chúng em và đã tạo điều kiện làm việc tốt cũng như đã không ngừng hỗ trợ cho nhóm chúng em. Sau một học kỳ thực hiện đề tài, bên cạnh sự nỗ lực của các cá nhân, sự hỗ trợ nhiệt tình từ các thầy và anh đã giúp nhóm em rất nhiều trong việc bắt kịp tiến độ đã đề ra và hoàn thiện hơn đề tài của mình.

Chúng em muốn bày tỏ lòng biết ơn sâu sắc đối với những nhận xét và góp ý chân thành của các thầy. Những góp ý của các thầy không chỉ giúp chúng em cải thiện nội dung của luận văn mà còn là nguồn động viên lớn giúp em trở nên tự tin hơn trong việc hoàn thành môn học và bảo vệ đồ án. Em xin kính chúc Trần Thanh Bình và thầy Lê Trọng Nhân sức khỏe dồi dào và thành công trong công tác giảng dạy. Mong hai thầy hãy tiếp tục mang lại sự tích cực cho các sinh viên mai sau.

Cuối cùng, mặc dù đã cố gắng hoàn thành đồ án trong phạm vi và khả năng cho phép, nhưng chúng em chắc chắn không thể tránh khỏi thiếu sót, rất mong nhận được sự góp ý và chỉ bảo của quý thầy cô và các bạn.

Chúng em xin chân thành cảm ơn.

Thành phố Hồ Chí Minh, tháng 12 năm 2024.

# Tóm tắt đồ án

Đồ án này tập trung vào việc hiện thực một hệ thống cân bằng một loại máy bay không người lái (UAV) được sử dụng rộng rãi trong nhiều ứng dụng từ giám sát đến giao hàng. Mục tiêu của đồ án là phát triển một hệ thống cân bằng cho quadcopter. Trong quá trình nghiên cứu, các thành phần chính của hệ thống đã được xác định gồm:

- **Bộ điều khiển (Controller):** Một thuật toán điều khiển PID (Proportional-Integral-Derivative) đã được áp dụng để điều khiển quadcopter và giữ cho nó ổn định trong không gian.
- **Các cảm biến (Sensors):** Các cảm biến bao gồm mpu6050 và cảm biến áp suất được sử dụng để thu thập dữ liệu về độ cao, các góc xoay và tốc độ của quadcopter.

Sau khi hiện thực, hệ thống đã được kiểm tra và đánh giá thông qua một loạt các thử nghiệm mô phỏng và thực tế. Kết quả cho thấy rằng hệ thống đề xuất có khả năng duy trì cân bằng ở một mức độ nhất định, đồng thời cung cấp khả năng điều khiển linh hoạt và ổn định cho quadcopter. Những kết quả này không chỉ mang lại đóng góp cho lĩnh vực nghiên cứu về UAV mà còn có thể được áp dụng trong nhiều ứng dụng thực tế khác, bao gồm giám sát môi trường, giao hàng, và giải cứu khẩn cấp.

Thành phố Hồ Chí Minh, tháng 12 năm 2024.

# Mục lục

Lời cam đoan	I
Lời cảm ơn	II
Tóm tắt đề án	III
Danh sách bảng	V
Danh sách hình vẽ	VI
<b>1 Tổng quan về đề tài</b>	<b>1</b>
1.1 Giới thiệu . . . . .	1
1.2 Mục tiêu . . . . .	1
1.3 Thực trạng . . . . .	2
1.4 Phạm vi dự án . . . . .	3
1.5 Ý nghĩa thực tiễn . . . . .	3
<b>2 CƠ SỞ LÝ THUYẾT</b>	<b>5</b>
2.1 Tổng quan về vi điều khiển ESP32-S3 . . . . .	5
2.1.1 Kiến trúc phần cứng của ESP32-S3 . . . . .	6
2.1.2 Các tính năng kết nối không dây . . . . .	6
2.1.3 Các giao tiếp ngoại vi . . . . .	7
2.1.4 Hệ điều hành và môi trường phát triển . . . . .	7
2.2 Cơ sở lý thuyết về module RS485 . . . . .	8
2.2.1 Giới thiệu chung về module RS485 . . . . .	8
2.2.2 Nguyên lý hoạt động của module RS485 . . . . .	8
2.2.3 Kết nối với vi điều khiển (ví dụ ESP32-S3) . . . . .	9
2.2.4 Ứng dụng thực tế . . . . .	9
2.3 MQTT trong IoT . . . . .	9
2.3.1 Tổng quan về MQTT . . . . .	9

2.3.2	Kiến trúc MQTT . . . . .	10
2.3.3	Nguyên lý hoạt động . . . . .	11
2.3.4	Cấu trúc topic . . . . .	12
2.3.5	Chất lượng dịch vụ (QoS) . . . . .	12
2.3.6	Ứng dụng của MQTT trong IoT . . . . .	12
2.3.7	Ưu và nhược điểm của MQTT . . . . .	13
2.4	Thư viện đồ họa LVGL . . . . .	13
2.4.1	Tổng quan về LVGL . . . . .	13
2.4.2	Kiến trúc LVGL . . . . .	14
2.4.3	Các thành phần chính giao diện . . . . .	15
2.5	Công nghệ hiển thị cho hệ thống nhúng . . . . .	16
2.5.1	Màn hình LCD và TFT . . . . .	16
2.5.2	Màn hình OLED . . . . .	17
2.5.3	Giao tiếp với màn hình . . . . .	17
2.6	Phân tích các giải pháp GUI cho ESP32 . . . . .	18
2.6.1	Các thư viện đồ họa hiện có . . . . .	18
2.6.1.1	TFT_eSPI (Bodmer) . . . . .	18
2.6.1.2	Adafruit GFX . . . . .	18
2.6.1.3	U8g2 . . . . .	19
2.6.1.4	LVGL . . . . .	19
2.6.2	So sánh và lựa chọn thư viện phù hợp . . . . .	20
2.6.2.1	Hiệu suất và tài nguyên . . . . .	20
2.6.2.2	Tính năng và khả năng mở rộng . . . . .	20
2.6.2.3	Độ phức tạp và đường cong học tập . . . . .	20
2.6.2.4	Hỗ trợ cộng đồng và tài liệu . . . . .	21
2.6.2.5	Trường hợp sử dụng phù hợp . . . . .	21
2.6.3	Tại sao chọn LVGL cho dự án phát triển thư viện giao diện . . . . .	21

# Danh sách bảng

# Danh sách hình vẽ

2.1	ESP32-S3 . . . . .	5
2.2	Kiến trúc phần cứng ESP32-S3 . . . . .	6
2.3	RS485 module . . . . .	8
2.4	MQTT trong IoT . . . . .	10
2.5	Mô hình Publish/Subscriber trong giao thức MQTT . . . . .	11
2.6	Cơ chế hoạt động của giao thức MQTT . . . . .	11
2.7	Ứng dụng của MQTT trong IoT . . . . .	13
2.8	Tổng quan về luồng dữ liệu của LVGL . . . . .	15



# Chương 1

## Tổng quan về đề tài

### 1.1 Giới thiệu

Trong thời đại hiện nay, drone (thiết bị bay không người lái) đang trở thành một công nghệ ngày càng phổ biến và quan trọng trong nhiều lĩnh vực, từ giám sát đến giao hàng và cứu hộ. Sức mạnh của chúng không chỉ là khả năng tiếp cận các khu vực khó tiếp cận mà còn là sự linh hoạt và hiệu quả trong thực hiện các nhiệm vụ.

Tuy nhiên, việc điều khiển drone trong môi trường đa dạng và không gian hạn chế đặt ra nhiều thách thức. Một trong những thách thức chính là việc duy trì cân bằng và tránh vật cản trong quá trình di chuyển.

Hệ thống tự cân bằng không chỉ tăng cường an toàn mà còn mở ra nhiều cơ hội ứng dụng mới trong các ngành công nghiệp khác nhau. Từ giao hàng hàng hóa đến giám sát môi trường và nông nghiệp, drone có thể phục vụ một loạt các mục đích với sự hỗ trợ của hệ thống cân bằng.

Chính vì vậy, nhóm chúng em đã chọn đề tài **Hệ thống cân bằng cho thiết bị bay không người lái**, nhằm nghiên cứu và phát triển các giải pháp để giải quyết những thách thức này. Đồng thời, chúng em hy vọng rằng việc thành công trong dự án này sẽ mang lại những đóng góp quan trọng cho sự phát triển của công nghệ drone trong tương lai.

### 1.2 Mục tiêu

Mục tiêu của nhóm chúng em trong dự án này là: Phát triển một hệ thống cân bằng tự động cho drone, nhằm đảm bảo khả năng hoạt động ổn định và chính xác trong quá trình bay để giúp người điều khiển có thể dễ dàng điều khiển drone. Hệ thống này sẽ sử dụng các thuật toán điều khiển tiên tiến, kết hợp với các cảm biến hiện đại để đọc và xử

lý dữ liệu về trạng thái của drone, từ đó đưa ra các tín hiệu điều khiển động cơ phù hợp, giúp duy trì trạng thái cân bằng theo thời gian thực. Cụ thể, mục tiêu của dự án bao gồm:

- **Xây dựng hệ thống phần cứng:** Thiết kế và tích hợp các thành phần chính của drone, bao gồm vi điều khiển (Teensy), mô-đun kết nối không dây (ESP8266), cảm biến (MPU6050 và barometer), và các bộ điều khiển động cơ (ESC). Đảm bảo sự phối hợp mượt mà giữa các thành phần để tạo thành một hệ thống hoàn chỉnh.
- **Phát triển thuật toán điều khiển cân bằng:** Ứng dụng thuật toán PID (Proportional-Integral-Derivative) để tính toán các tín hiệu điều khiển dựa trên dữ liệu từ cảm biến. Thuật toán sẽ liên tục điều chỉnh tốc độ quay của động cơ, nhằm giữ drone cân bằng và thực hiện các thao tác bay chính xác theo mong muốn của người điều khiển.
- **Tăng cường khả năng ổn định và tin cậy:** Sử dụng bộ lọc Kalman để loại bỏ nhiễu trong dữ liệu cảm biến, cải thiện độ chính xác của hệ thống. Đồng thời, tích hợp cơ chế kiểm tra và giám sát tính toàn vẹn của dữ liệu, đảm bảo sự ổn định và an toàn khi vận hành.
- **Kết nối và điều khiển từ xa:** Phát triển giao diện kết nối giữa drone và ứng dụng desktop thông qua Wi-Fi, giúp người dùng dễ dàng cấu hình thông số và theo dõi trạng thái của drone theo thời gian thực.

Thông qua việc đạt được các mục tiêu trên, đề tài không chỉ tập trung vào việc tạo ra một hệ thống cân bằng tự động cho drone, mà còn hướng tới việc xây dựng nền tảng công nghệ có thể được mở rộng cho các ứng dụng khác trong tương lai, như quay phim, khảo sát địa hình, và vận chuyển hàng hóa.

### 1.3 Thực trạng

Bây giờ nền công nghiệp về drone đã phát triển vượt bậc, vẫn còn tồn tại một số hạn chế trong các hệ thống điều khiển hiện tại. Đặc biệt, khả năng cân bằng của drone trong môi trường đa dạng vẫn còn hạn chế, đặc biệt là khi di chuyển theo quỹ đạo cụ thể hoặc trong điều kiện thời tiết không lý tưởng.

Mặc dù đã có sự tiến bộ đáng kể trong việc phát triển các hệ thống cân bằng cho drone, nhưng vấn đề vẫn còn tồn tại và đòi hỏi sự nghiên cứu và phát triển tiếp tục để tối ưu hóa hiệu suất và độ chính xác của chúng trong các ứng dụng thực tế. Có thể kể đến một số thách thức như:

- Độ chính xác của cảm biến: Các hệ thống cân bằng phụ thuộc nhiều vào độ chính xác của cảm biến, như cảm biến gia tốc, cảm biến áp suất, và cảm biến khoảng cách. Sự không chính xác hoặc nhiễu từ các cảm biến này có thể dẫn đến sai lệch trong việc đo lường và điều khiển.
- Xử lý dữ liệu và thời gian phản ứng: Trong khi cân bằng đòi hỏi phản ứng nhanh chóng từ drone, việc xử lý dữ liệu từ các cảm biến và áp dụng các thuật toán phức tạp có thể gây trễ trong quá trình điều khiển.
- Điều kiện môi trường biến đổi: Drone thường phải hoạt động trong các điều kiện môi trường biến đổi, từ thời tiết không ổn định đến ánh sáng yếu và môi trường địa hình khác nghiệt. Điều này làm tăng thêm độ khó trong việc cân bằng.

### 1.4 Phạm vi dự án

Trước những khó khăn khi thực hiện dự án, chúng em đã giới hạn những tính năng của hệ thống nhằm đảm bảo dự án nằm trong khả năng kiến thức cũng như đảm bảo dự án được hoàn thành đúng tiến độ. Loại thiết bị bay không người lái nhóm em chọn là loại có 4 cánh (quadcopter) hình chữ X. Hệ thống cân bằng sẽ được triển khai trên bốn bậc tự do là độ cao và ba góc xoay pitch, roll yaw trong đó quan trọng nhất là tập trung vào hai góc pitch và roll để giữ cho drone nằm yên trên mặt phẳng. khi hoạt động drone có thể duy trì cân bằng ổn định, các góc nghiêng không lệch quá lớn, độ cao được giữ ổn định.

Vì thời gian thực hiện hạn chế và tránh các tác động ngẫu nhiên của thời tiết để nghiên cứu và thực hiện đề tài này, trước tiên nhóm em sẽ tiến hành trong môi trường phòng thí nghiệm trước, từ đó sẽ mở rộng, thêm các yếu tố ngoại vi gắn liền hơn với thực tế nếu còn đủ thời gian.

### 1.5 Ý nghĩa thực tiễn

Hệ thống cân bằng góp phần là cơ sở nền tảng để phát triển những ứng dụng của drone vào trong thực tiễn. Trong thời đại công nghệ 4.0, drone ngày càng xuất hiện phổ biến và rộng rãi. Hiện tại có rất nhiều thứ để con người tận dụng drone trong công việc và cả giải trí, ví dụ như:

1. Chụp ảnh và quay phim: Drone được sử dụng rộng rãi trong lĩnh vực chụp ảnh và quay phim, giúp ghi lại những hình ảnh và thước phim độc đáo từ trên cao.

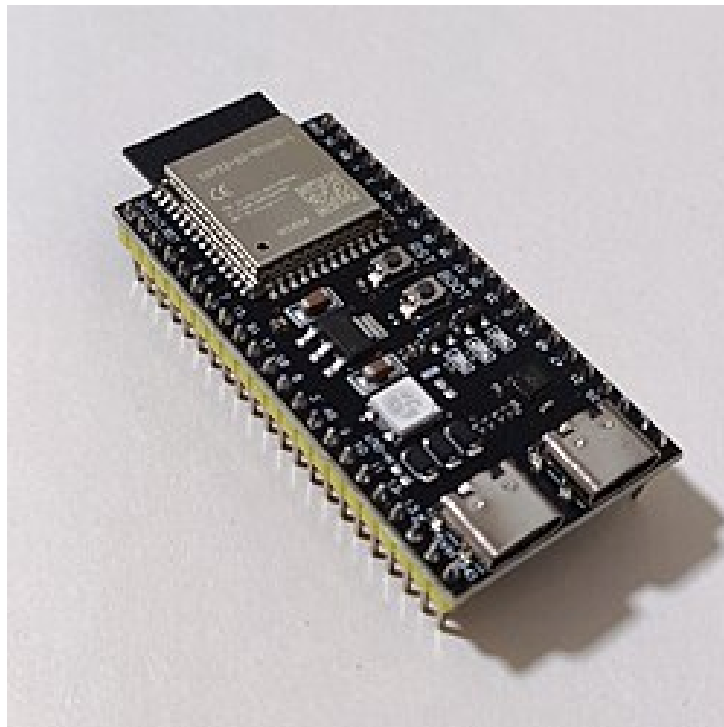
2. Kiểm tra và giám sát: Drone được sử dụng để kiểm tra và giám sát các công trình xây dựng, đường dây điện, đường ống, cánh đồng, v.v., giúp phát hiện sớm các vấn đề và đưa ra giải pháp kịp thời.
3. Giao hàng: Drone được sử dụng để giao hàng ở những khu vực khó tiếp cận, giúp rút ngắn thời gian giao hàng và tiết kiệm chi phí vận chuyển.
4. Tìm kiếm cứu nạn: Drone được sử dụng để tìm kiếm người mất tích, nạn nhân trong các thảm họa thiên tai, giúp cứu trợ kịp thời và hiệu quả.
5. Nông nghiệp: Drone được sử dụng để phun thuốc trừ sâu, bón phân, tưới nước cho cây trồng, giúp nâng cao năng suất và hiệu quả sản xuất nông nghiệp.
6. Khảo sát địa hình: Drone được sử dụng để khảo sát địa hình ở những nơi con người khó tiếp cận, lập bản đồ, cung cấp dữ liệu cho các dự án xây dựng, quy hoạch đô thị, v.v.
7. Giải trí: Drone được sử dụng cho các hoạt động giải trí như đua drone, biểu diễn drone và nhiều hoạt động giải trí khác.

## Chương 2

# CƠ SỞ LÝ THUYẾT

### 2.1 Tổng quan về vi điều khiển ESP32-S3

ESP32-S3 là một vi điều khiển thuộc dòng ESP32 do Espressif Systems phát triển, được thiết kế đặc biệt cho các ứng dụng AIoT (Artificial Intelligence of Things), tích hợp khả năng xử lý tín hiệu số, hỗ trợ tăng tốc AI và xử lý đồ họa hiệu quả. Với kiến trúc mạnh mẽ, khả năng kết nối không dây linh hoạt và hỗ trợ phần cứng cho các tác vụ học máy, ESP32-S3 trở thành lựa chọn lý tưởng cho các ứng dụng IoT hiện đại như điều khiển thông minh, nhận diện hình ảnh, và giao diện người dùng đồ họa.



Hình 2.1: ESP32-S3

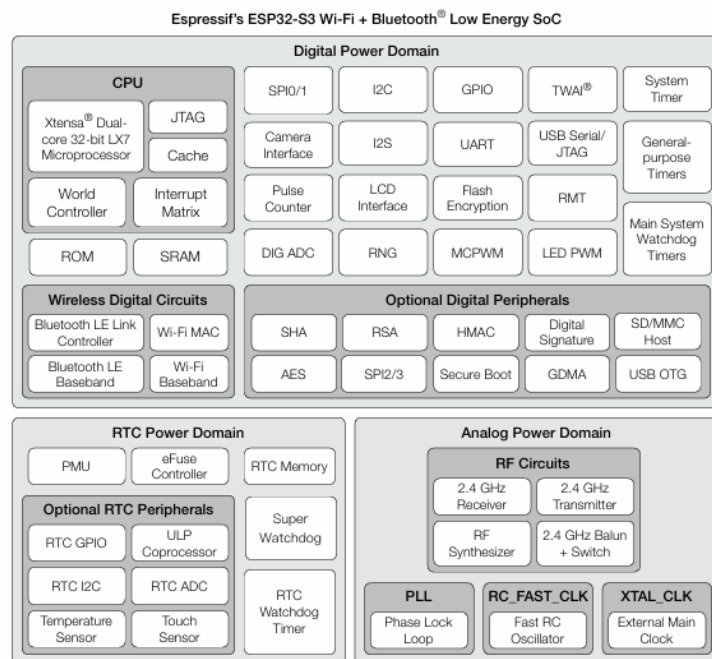
### 2.1.1 Kiến trúc phần cứng của ESP32-S3

ESP32-S3 sử dụng vi xử lý dual-core Tensilica Xtensa LX7, chạy ở tốc độ lên đến 240 MHz. So với LX6 trên ESP32 tiêu chuẩn, LX7 cung cấp hiệu suất cao hơn và hiệu quả năng lượng tốt hơn, đặc biệt trong các tác vụ xử lý tín hiệu và AI.

Bộ vi xử lý hỗ trợ bộ chỉ dẫn vector SIMD (Single Instruction, Multiple Data) và có tích hợp tăng tốc phần cứng cho các tác vụ AI như nhận diện giọng nói, phân loại hình ảnh, hoặc phát hiện đối tượng. Điều này cho phép ESP32-S3 thực hiện các ứng dụng machine learning ở mức edge mà không cần đến bộ xử lý trung tâm.

Bộ nhớ tích hợp gồm 512KB SRAM, 384KB ROM, và hỗ trợ bộ nhớ flash ngoài lên đến 16MB. Ngoài ra, bộ nhớ PSRAM ngoài có thể được tích hợp để hỗ trợ các ứng dụng yêu cầu dung lượng lớn như LVGL hoặc TensorFlow Lite Micro.

ESP32-S3 có tổng cộng 45 chân GPIO, hỗ trợ chức năng đa dạng như PWM, ADC, DAC, SPI, I2C, UART, I2S, SDIO và USB-OTG. Đặc biệt, chip hỗ trợ giao tiếp USB đầy đủ (USB 1.1 OTG), giúp kết nối trực tiếp với máy tính hoặc thiết bị ngoại vi mà không cần chip chuyển đổi USB-UART.



Hình 2.2: Kiến trúc phần cứng ESP32-S3

### 2.1.2 Các tính năng kết nối không dây

ESP32-S3 hỗ trợ Wi-Fi IEEE 802.11 b/g/n (2.4GHz) và Bluetooth 5.0 LE với các tính năng mở rộng như BLE Mesh và Advertising Extensions, giúp xây dựng các mạng thiết bị IoT phức tạp và tiết kiệm năng lượng hơn.

Mặc dù ESP32-S3 không hỗ trợ Bluetooth cổ điển (BR/EDR), nhưng BLE 5.0 đủ mạnh để đáp ứng phần lớn các ứng dụng IoT hiện đại, đặc biệt là thiết bị wearable, cảm biến không dây, và các thiết bị cần truyền dữ liệu định kỳ.

ESP32-S3 tích hợp nhiều cơ chế bảo mật phần cứng: Secure Boot, Flash Encryption, hỗ trợ các thuật toán mã hóa AES, SHA, RSA, ECC thông qua bộ tăng tốc phần cứng, đảm bảo an toàn cho dữ liệu và firmware.

### 2.1.3 Các giao tiếp ngoại vi

ESP32-S3 cung cấp đầy đủ các giao tiếp ngoại vi tương tự như ESP32 tiêu chuẩn nhưng được mở rộng và cải tiến hơn:

**GPIO:** 45 chân có thể lập trình, hỗ trợ ngắt, và nhiều chức năng đặc biệt.

**SPI, I2C, UART:** Hỗ trợ đa kênh, tốc độ cao và cấu hình linh hoạt.

**USB-OTG:** Cho phép thiết bị hoạt động như USB Host hoặc Device.

**ADC/DAC:** 20 kênh ADC độ phân giải 12-bit, cùng với 2 kênh DAC.

**PWM:** 8 bộ định thời PWM, mỗi bộ có 4 kênh đầu ra, đủ cho điều khiển động cơ, đèn LED RGB, v.v.

**SD/MMC/SDIO:** Giao tiếp với thẻ nhớ hoặc thiết bị lưu trữ.

**Camera Interface:** ESP32-S3 hỗ trợ giao tiếp camera (DVP), rất thích hợp cho ứng dụng nhận diện hình ảnh.

### 2.1.4 Hệ điều hành và môi trường phát triển

ESP32-S3 được hỗ trợ đầy đủ bởi ESP-IDF – bộ SDK chính thức của Espressif, đi kèm FreeRTOS, tích hợp nhiều thư viện cho AI, LVGL, và các giao thức mạng (MQTT, HTTP, mDNS, v.v.).

Bên cạnh ESP-IDF, ESP32-S3 cũng có thể được lập trình bằng Arduino IDE với lõi hỗ trợ riêng, thích hợp cho lập trình viên mới bắt đầu.

Ngoài ra, ESP32-S3 hỗ trợ MicroPython, TensorFlow Lite Micro, và các công cụ debug nâng cao như JTAG thông qua USB.

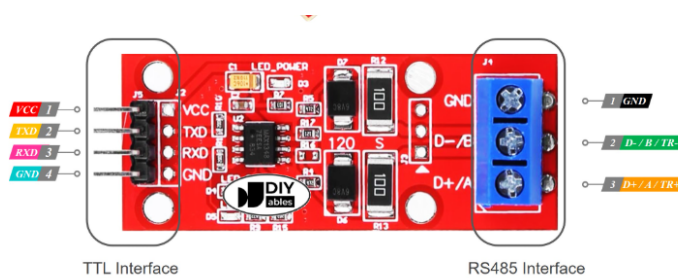
Với khả năng xử lý mạnh mẽ, tích hợp tăng tốc AI, hỗ trợ đồ họa, và nhiều phương thức kết nối, ESP32-S3 là nền tảng lý tưởng cho các hệ thống IoT có giao diện phức tạp, thiết bị thông minh, và các ứng dụng học máy nhúng.

## 2.2 Cơ sở lý thuyết về module RS485

### 2.2.1 Giới thiệu chung về module RS485

Module RS485 là một thiết bị trung gian giúp chuyển đổi tín hiệu UART (TTL) từ vi điều khiển sang chuẩn truyền thông RS485, cho phép giao tiếp trong các hệ thống truyền dữ liệu công nghiệp hoặc khoảng cách xa. Các module này thường được tích hợp sẵn mạch điện trở kéo, bảo vệ điện áp và mạch truyền nhận để đảm bảo tín hiệu ổn định và an toàn trong quá trình truyền thông.

RS485 hỗ trợ giao tiếp đa điểm (multi-point), cho phép kết nối nhiều thiết bị với nhau trên cùng một đường truyền dữ liệu (bus), giúp đơn giản hóa hệ thống dây và giảm chi phí triển khai trong các ứng dụng như hệ thống giám sát, tự động hóa tòa nhà, mạng cảm biến môi trường, v.v.



Hình 2.3: RS485 module

### 2.2.2 Nguyên lý hoạt động của module RS485

Module RS485 hoạt động dựa trên nguyên lý truyền vi sai (differential signaling), sử dụng hai dây tín hiệu chính là A và B. Tín hiệu được truyền dưới dạng chênh lệch điện áp giữa hai dây này thay vì so với mass như các chuẩn UART hay RS232 thông thường. Khi mức điện áp trên chân A cao hơn B ( $A > B$ ), tín hiệu được xem là logic "1", ngược lại nếu  $A < B$  thì là logic "0". Điều này giúp loại bỏ nhiễu điện từ chung trên đường truyền và duy trì tính toàn vẹn dữ liệu.

Các module RS485 phổ biến hiện nay sử dụng IC chuyển đổi như MAX485, SP485, hoặc SN75176 để thực hiện quá trình mã hóa và giải mã tín hiệu vi sai. Các IC này thường hỗ trợ cả chế độ truyền và nhận (half-duplex), và cần thêm một chân điều khiển (thường là DE/RE) để chuyển đổi giữa hai chế độ.



### 2.2.3 Kết nối với vi điều khiển (ví dụ ESP32-S3)

Việc kết nối module RS485 với vi điều khiển như ESP32-S3 thường thông qua giao tiếp UART. Các chân TX và RX của vi điều khiển được nối với module RS485 thông qua mạch chuyển đổi mức tín hiệu. Ngoài ra, chân DE (Driver Enable) và RE (Receiver Enable) thường được điều khiển bằng phần mềm để bật chế độ truyền hoặc nhận dữ liệu. Trong nhiều module, DE và RE được nối với nhau và điều khiển chung qua một chân GPIO của vi điều khiển để đơn giản hóa việc lập trình.

Ví dụ, để gửi dữ liệu, chân DE/RE được đặt ở mức HIGH để bật chế độ truyền. Sau khi gửi xong, chân này sẽ được kéo xuống mức LOW để chuyển sang chế độ nhận. Thời điểm chuyển trạng thái cần được lập trình chính xác để tránh mất dữ liệu hoặc xung đột đường truyền.

### 2.2.4 Ứng dụng thực tế

Module RS485 được sử dụng rộng rãi trong nhiều ứng dụng yêu cầu truyền thông ổn định ở khoảng cách xa như:

- Giao tiếp giữa các bộ điều khiển khả trình (PLC) và cảm biến/thiết bị chấp hành.
- Mạng Modbus RTU trong hệ thống SCADA.
- Mạng cảm biến trong nhà máy, nông nghiệp thông minh, hoặc giám sát năng lượng.
- Hệ thống đo đạc từ xa như đồng hồ điện, nước, và hệ thống kiểm soát truy cập.

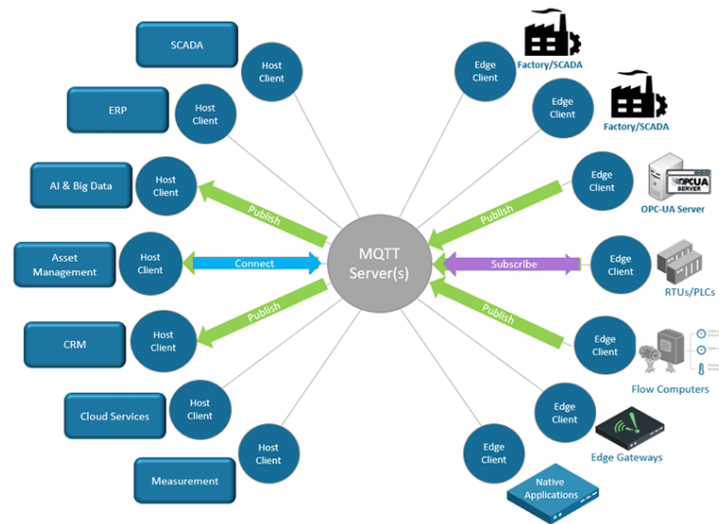
Với khả năng chống nhiễu tốt, chi phí thấp và triển khai dễ dàng, module RS485 là lựa chọn lý tưởng cho các hệ thống truyền thông công nghiệp và IoT.

## 2.3 MQTT trong IoT

### 2.3.1 Tổng quan về MQTT

MQTT (Message Queuing Telemetry Transport) là một giao thức truyền thông nhẹ, được thiết kế tối ưu cho các ứng dụng có băng thông thấp, độ trễ cao hoặc yêu cầu tiêu thụ năng lượng thấp — đặc điểm điển hình trong các hệ thống Internet of Things (IoT). Giao thức này hoạt động theo mô hình **publish/subscribe** (xuất bản/đăng ký), cho phép các thiết bị trao đổi dữ liệu thông qua một máy chủ trung gian gọi là **broker**.

MQTT ban đầu được phát triển bởi IBM vào năm 1999 và hiện nay là một chuẩn mở, được quản lý bởi tổ chức OASIS (Organization for the Advancement of Structured Information Standards).

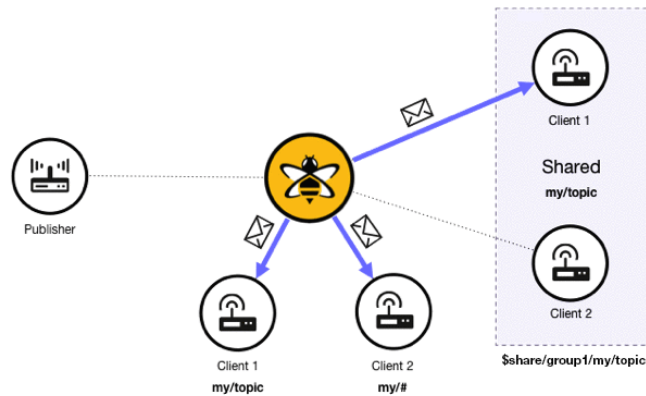


Hình 2.4: MQTT trong IoT

### 2.3.2 Kiến trúc MQTT

Kiến trúc MQTT bao gồm ba thành phần chính:

- **Publisher (Thiết bị xuất bản):** Gửi thông điệp đến một chủ đề (topic) cụ thể.
- **Subscriber (Thiết bị đăng ký):** Nhận thông điệp bằng cách đăng ký các chủ đề quan tâm.
- **Broker (Máy chủ trung gian):** Tiếp nhận các thông điệp từ publisher và phân phối chúng đến các subscriber phù hợp. Ví dụ broker phổ biến: Mosquitto, HiveMQ, EMQX.



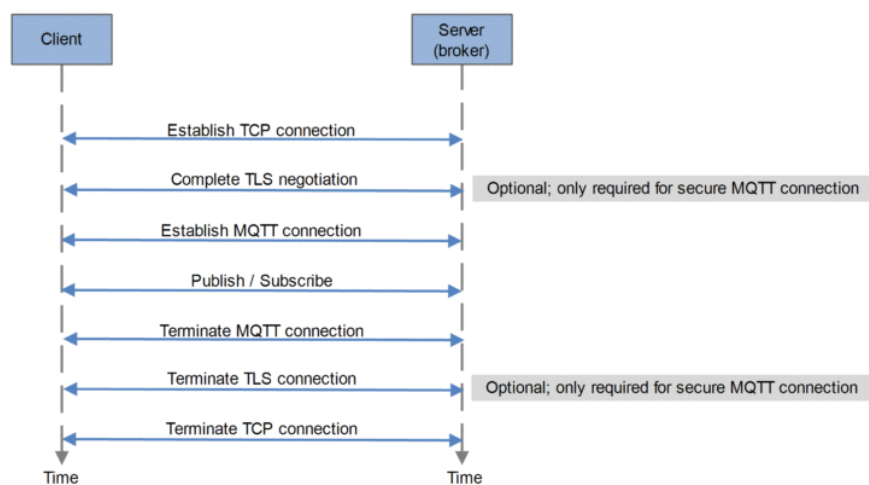
Hình 2.5: Mô hình Publish/Subscriber trong giao thức MQTT

### 2.3.3 Nguyên lý hoạt động

Khi một thiết bị **publisher** gửi một thông điệp (message) đến một **topic**, **broker** sẽ tiếp nhận thông điệp và phân phối nó đến tất cả các thiết bị **subscriber** đã đăng ký với topic đó. Giao thức MQTT cho phép hệ thống mở rộng dễ dàng và giảm đáng kể độ phụ thuộc giữa các thiết bị đầu cuối.

Ví dụ:

- Publisher gửi dữ liệu cảm biến đến topic “/home/temperature”.
- Tất cả các subscriber quan tâm đến chủ đề này sẽ nhận được dữ liệu mà không cần biết thông tin về publisher.



Hình 2.6: Cơ chế hoạt động của giao thức MQTT

### 2.3.4 Cấu trúc topic

Topic trong MQTT được tổ chức theo dạng cây, sử dụng dấu gạch chéo / để phân tách các cấp.

Ví dụ:

- /home/livingroom/temperature
- /device/esp32/status

MQTT hỗ trợ ký tự đại diện:

- + đại diện cho một cấp (level).
- # đại diện cho nhiều cấp (từ vị trí hiện tại trở đi).

### 2.3.5 Chất lượng dịch vụ (QoS)

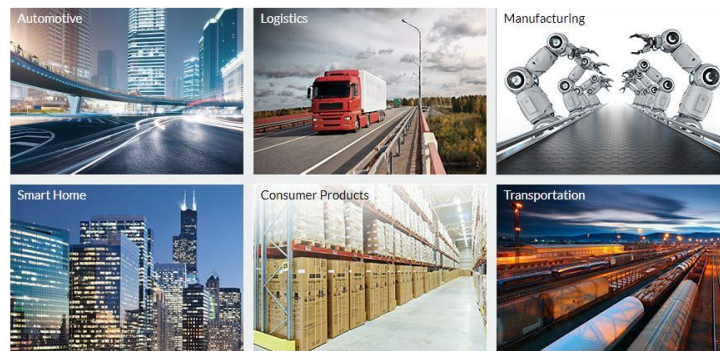
MQTT hỗ trợ ba mức độ đảm bảo chất lượng dịch vụ (Quality of Service - QoS):

- **QoS 0 - At most once:** Gửi một lần và không đảm bảo nhận được.
- **QoS 1 - At least once:** Gửi ít nhất một lần, có thể trùng lặp.
- **QoS 2 - Exactly once:** Gửi đúng một lần (đảm bảo không trùng và không mất).

### 2.3.6 Ứng dụng của MQTT trong IoT

MQTT được sử dụng rộng rãi trong nhiều hệ thống IoT nhờ vào tính nhẹ, đơn giản và hiệu quả:

- **Nhà thông minh (Smart home):** Kết nối thiết bị đèn, cảm biến, điều hòa.
- **Nông nghiệp thông minh:** Gửi dữ liệu độ ẩm, nhiệt độ, mực nước về máy chủ.
- **Giám sát công nghiệp:** Truyền dữ liệu cảm biến từ xa về trung tâm điều khiển.
- **Thiết bị đeo và sức khỏe:** Đồng bộ hóa dữ liệu sức khỏe với cloud server.



Hình 2.7: Ứng dụng của MQTT trong IoT

### 2.3.7 Ưu và nhược điểm của MQTT

**Ưu điểm:**

- Giao thức nhẹ, phù hợp với thiết bị tài nguyên hạn chế.
- Giao tiếp không đồng bộ và tách biệt giữa publisher và subscriber.
- Hỗ trợ nhiều mức QoS và cơ chế giữ kết nối (keep-alive).

**Nhược điểm:**

- Phụ thuộc vào broker trung gian.
- Không mã hóa mặc định, cần kết hợp với SSL/TLS để đảm bảo an toàn.
- Không phù hợp với các ứng dụng yêu cầu thời gian thực chính xác tuyệt đối.

## 2.4 Thư viện đồ họa LVGL

LVGL (Light and Versatile Graphics Library) là một thư viện đồ họa mã nguồn mở, nhẹ nhàng và linh hoạt, được thiết kế đặc biệt cho các hệ thống nhúng với tài nguyên hạn chế. Thư viện này cung cấp tất cả các công cụ cần thiết để tạo ra giao diện người dùng đồ họa (GUI) hấp dẫn và chuyên nghiệp trên các vi điều khiển như ESP32, với hiệu suất tối ưu và yêu cầu bộ nhớ thấp.

### 2.4.1 Tổng quan về LVGL

LVGL được phát triển từ năm 2016 bởi Gábor Kiss-Vámosi, ban đầu với tên gọi LittlevGL. Qua nhiều năm phát triển, LVGL đã trở thành một trong những thư viện GUI phổ biến nhất cho các hệ thống nhúng, với các phiên bản chính từ v5 đến v9 hiện nay.

Mỗi phiên bản đều mang đến những cải tiến đáng kể về hiệu suất, tính năng và khả năng tương thích.

LVGL có nhiều đặc điểm và ưu điểm nổi bật so với các thư viện đồ họa khác cho hệ thống nhúng. Thư viện này được thiết kế với kiến trúc module hóa, cho phép chỉ biên dịch các thành phần cần thiết, giảm thiểu kích thước mã nguồn. LVGL hỗ trợ đa nền tảng, có thể chạy trên hầu hết các vi điều khiển 16, 32 hoặc 64 bit, từ các chip đơn giản như STM32 đến các nền tảng mạnh mẽ hơn như ESP32 hoặc Raspberry Pi.

Một trong những ưu điểm lớn nhất của LVGL là khả năng tạo ra giao diện người dùng hiện đại và hấp dẫn với yêu cầu tài nguyên tối thiểu. Thư viện có thể hoạt động với chỉ 64 KB flash và 16 KB RAM, mặc dù cấu hình được khuyến nghị là khoảng 180 KB flash và 48 KB RAM để sử dụng đầy đủ tính năng. LVGL cũng hỗ trợ hoạt động với chỉ một frame buffer, giảm thiểu yêu cầu bộ nhớ đồng thời vẫn duy trì hiệu ứng đồ họa mượt mà.

Về yêu cầu hệ thống và tương thích, LVGL có thể hoạt động trên hầu hết các vi điều khiển với tốc độ xung nhịp từ 16 MHz trở lên, mặc dù khuyến nghị tối thiểu 32 MHz cho hiệu suất tốt. Thư viện hỗ trợ nhiều loại màn hình với độ sâu màu khác nhau, từ màn hình đơn sắc đến màn hình TFT màu 16 hoặc 24 bit. LVGL cũng tương thích với nhiều loại thiết bị đầu vào như màn hình cảm ứng, nút bấm, encoder, và bàn phím.

## **2.4.2 Kiến trúc LVGL**

LVGL được xây dựng với kiến trúc module hóa, bao gồm các thành phần cốt lõi và các module chức năng. Cấu trúc này cho phép tùy chỉnh và mở rộng linh hoạt, đồng thời duy trì hiệu suất tối ưu cho các hệ thống nhúng.

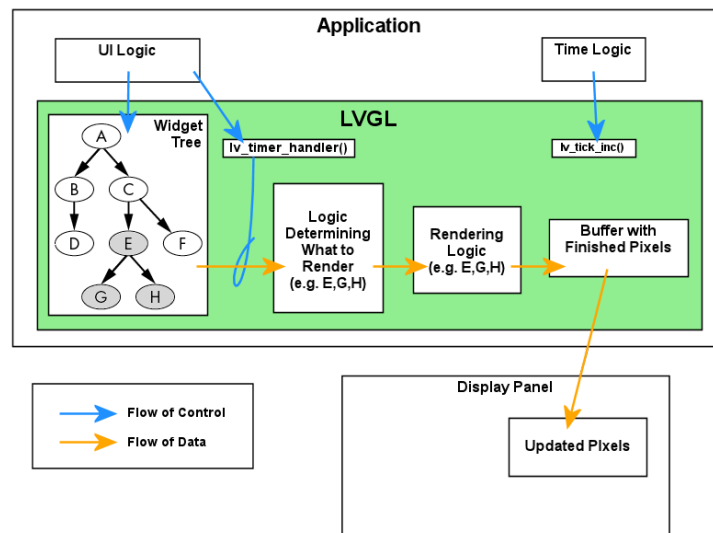
Cấu trúc lõi của LVGL bao gồm các module cơ bản như hệ thống đối tượng, quản lý bộ nhớ, hệ thống sự kiện, và engine vẽ. Các module này tạo nền tảng cho toàn bộ thư viện và cung cấp các chức năng cơ bản như quản lý đối tượng, xử lý sự kiện, và render đồ họa. LVGL sử dụng kiến trúc hướng đối tượng mặc dù được viết bằng C, với các cấu trúc dữ liệu và hàm được tổ chức theo cách mô phỏng lập trình hướng đối tượng.

Hệ thống đối tượng và thừa kế trong LVGL cho phép tạo ra các widget phức tạp từ các thành phần cơ bản. Mỗi đối tượng trong LVGL đều kế thừa từ lớp cơ sở `lv_obj`, thừa hưởng các thuộc tính và phương thức của lớp cha, đồng thời có thể mở rộng với các chức năng riêng. Cơ chế thừa kế này giúp giảm thiểu mã lặp lại và tạo ra hệ thống widget nhất quán và dễ mở rộng.

Cơ chế quản lý bộ nhớ trong LVGL được thiết kế đặc biệt cho các hệ thống nhúng với bộ nhớ hạn chế. LVGL sử dụng bộ cấp phát bộ nhớ động riêng, cho phép kiểm soát chặt chẽ việc sử dụng bộ nhớ và tránh phân mảnh. Thư viện cũng hỗ trợ các cơ chế tái

sử dụng bộ nhớ và giải phóng tự động, giảm thiểu rò rỉ bộ nhớ và tối ưu hóa hiệu suất.

Hệ thống sự kiện và callback trong LVGL cho phép xử lý tương tác người dùng một cách linh hoạt và hiệu quả. Mỗi đối tượng có thể đăng ký các hàm callback để phản hồi với các sự kiện như nhấn, kéo, hoặc thay đổi giá trị. Hệ thống sự kiện hỗ trợ lan truyền sự kiện qua cây đối tượng, cho phép xử lý sự kiện ở nhiều cấp độ khác nhau.



Hình 2.8: Tổng quan về luồng dữ liệu của LVGL

### 2.4.3 Các thành phần chính giao diện

LVGL cung cấp một bộ widget phong phú để xây dựng giao diện người dùng, từ các thành phần cơ bản đến các container phức tạp và hệ thống bố cục linh hoạt.

Hệ thống widget cơ bản của LVGL bao gồm các thành phần như Button (nút bấm), Label (nhãn), và Image (hình ảnh). Button cung cấp chức năng tương tác cơ bản, với hỗ trợ cho các trạng thái khác nhau như nhấn, thả, và vô hiệu hóa. Label cho phép hiển thị văn bản với nhiều tùy chọn về font, căn chỉnh, và cuộn. Image hỗ trợ nhiều định dạng hình ảnh, bao gồm cả hình ảnh nén và hình ảnh với kênh alpha.

Ngoài các widget cơ bản, LVGL còn cung cấp các thành phần phức tạp hơn như Chart (biểu đồ), Table (bảng), Dropdown (danh sách thả xuống), và Keyboard (bàn phím ảo). Các widget này cho phép xây dựng giao diện người dùng phức tạp với ít mã nguồn hơn, đồng thời duy trì hiệu suất tốt trên các hệ thống nhúng.

Hệ thống styles và themes trong LVGL cho phép tùy chỉnh giao diện một cách linh hoạt và nhất quán. Styles định nghĩa các thuộc tính như màu sắc, font, padding, và border cho các widget. LVGL sử dụng cơ chế CSS-like để áp dụng styles, cho phép kế thừa và ghi đè các thuộc tính. Themes là tập hợp các styles được định nghĩa sẵn, cung cấp giao diện nhất quán cho toàn bộ ứng dụng.

Bố cục và container trong LVGL giúp tổ chức các widget một cách linh hoạt và phản ứng. LVGL hỗ trợ các hệ thống bố cục hiện đại như Flex (dựa trên CSS Flexbox) và Grid (dựa trên CSS Grid), cho phép tạo ra giao diện phản ứng tự động điều chỉnh theo kích thước màn hình. Các container như Panel, Tabview, và Window giúp tổ chức nội dung thành các phần logic và cung cấp các chức năng như cuộn và chuyển tab.

Hiệu ứng đồ họa và animation trong LVGL mang lại trải nghiệm người dùng động và hấp dẫn. Thư viện hỗ trợ các hiệu ứng như fade, move, và scale, với khả năng tùy chỉnh thời gian, đường cong chuyển động, và callback. Animations có thể được áp dụng cho hầu hết các thuộc tính của widget, từ vị trí và kích thước đến màu sắc và độ trong suốt.

## **2.5 Công nghệ hiển thị cho hệ thống nhúng**

Công nghệ hiển thị đóng vai trò quan trọng trong việc phát triển giao diện người dùng cho các hệ thống nhúng như ESP32. Các loại màn hình khác nhau cung cấp nhiều tùy chọn về kích thước, độ phân giải, chất lượng hiển thị và phương thức giao tiếp, cho phép nhà phát triển lựa chọn giải pháp phù hợp nhất với yêu cầu của dự án.

### **2.5.1 Màn hình LCD và TFT**

Màn hình LCD (Liquid Crystal Display) và đặc biệt là công nghệ TFT (Thin Film Transistor) LCD đã trở thành lựa chọn phổ biến cho các ứng dụng nhúng nhờ khả năng hiển thị màu sắc sống động, độ phân giải cao và giá thành hợp lý.

Nguyên lý hoạt động của màn hình LCD dựa trên việc điều khiển các tinh thể lỏng để thay đổi cách ánh sáng đi qua chúng. Mỗi điểm ảnh (pixel) trên màn hình LCD bao gồm các tinh thể lỏng được đặt giữa hai tấm phân cực. Khi áp dụng điện áp, các tinh thể lỏng xoay và thay đổi cách ánh sáng đi qua, tạo ra các mức độ sáng tối khác nhau. Trong màn hình TFT LCD, mỗi pixel được điều khiển bởi một hoặc nhiều transistor màng mỏng, cho phép điều khiển độc lập và cải thiện đáng kể chất lượng hiển thị so với LCD thông thường.

Các loại màn hình phổ biến cho ESP32 bao gồm ILI9341 và ST7789. Màn hình ILI9341 thường có kích thước 2.8 inch với độ phân giải 240x320 pixel, trong khi ST7789 thường được sử dụng trong các màn hình có kích thước từ 1.3 đến 2.0 inch với độ phân giải tương tự. Cả hai loại controller này đều hỗ trợ giao tiếp SPI, cho phép kết nối dễ dàng với ESP32 và tiêu thụ ít chân GPIO.

Các đặc tính kỹ thuật quan trọng của màn hình TFT LCD bao gồm độ phân giải,



độ sâu màu, góc nhìn và tốc độ làm mới. Độ phân giải thông thường cho các màn hình nhỏ dao động từ 240x240 đến 480x320 pixel. Độ sâu màu thường là 16-bit (65,536 màu) hoặc 18-bit (262,144 màu). Các yếu tố như góc nhìn và độ sáng cũng đóng vai trò quan trọng đối với trải nghiệm người dùng.

### **2.5.2 Màn hình OLED**

Màn hình OLED (Organic Light Emitting Diode) là công nghệ hiển thị tiên tiến với nhiều ưu điểm so với LCD, đặc biệt phù hợp cho các ứng dụng nhúng yêu cầu tiết kiệm năng lượng và chất lượng hiển thị cao.

Nguyên lý hoạt động của màn hình OLED dựa trên các diode phát quang hữu cơ có khả năng tự phát sáng khi có dòng điện đi qua. Mỗi pixel là một diode phát quang độc lập, không cần đèn nền. Điều này cho phép tỷ lệ tương phản cao, thời gian đáp ứng nhanh, góc nhìn rộng và tiêu thụ năng lượng thấp khi hiển thị nội dung tối.

So với LCD/TFT, OLED có tỷ lệ tương phản và độ sống động màu sắc cao hơn, thời gian đáp ứng nhanh hơn (dưới 1ms), và tiêu thụ năng lượng thấp hơn. Tuy nhiên, OLED có tuổi thọ ngắn hơn, dễ bị hiện tượng burn-in và chi phí cao hơn.

Trong hệ thống nhúng, OLED phù hợp cho thiết bị đeo, thiết bị y tế cầm tay và các thiết bị IoT chạy pin. Nó cũng là lựa chọn lý tưởng trong môi trường ánh sáng yếu.

### **2.5.3 Giao tiếp với màn hình**

Giao tiếp giữa ESP32 và màn hình là yếu tố ảnh hưởng đến hiệu suất và độ phức tạp của hệ thống. Các giao thức phổ biến gồm SPI, I2C và giao tiếp song song.

- **SPI (Serial Peripheral Interface):** phổ biến cho màn hình TFT và OLED. Giao thức này sử dụng các đường MOSI, MISO, SCK và CS, có tốc độ cao (lên đến 80MHz). ESP32 có thể dùng DMA để cải thiện hiệu suất truyền dữ liệu.
- **I2C (Inter-Integrated Circuit):** phù hợp với màn hình OLED nhỏ và controller cảm ứng. Chỉ sử dụng hai dây SDA và SCL, giúp tiết kiệm GPIO. Nhược điểm là tốc độ thấp hơn (100kHz đến 1MHz).
- **Giao tiếp song song (Parallel):** cung cấp băng thông cao nhất nhưng cần nhiều GPIO. Giao tiếp 8-bit hoặc 16-bit được hỗ trợ qua giao diện I8080/6800.

Về điều khiển cảm ứng, các controller như XPT2046 (cảm ứng điện trở) và FT6X36 (cảm ứng điện dung) sử dụng SPI hoặc I2C. Cảm ứng điện dung hỗ trợ multi-touch và có trải nghiệm tốt hơn, nhưng phức tạp và đắt hơn cảm ứng điện trở.

Việc lựa chọn công nghệ hiển thị và phương thức giao tiếp phù hợp là yếu tố then chốt để tạo ra giao diện người dùng hiệu quả trên nền tảng ESP32. Kết hợp với thư viện như LVGL sẽ nâng cao trải nghiệm người dùng.

## **2.6 Phân tích các giải pháp GUI cho ESP32**

Khi phát triển ứng dụng với giao diện đồ họa người dùng (GUI) cho ESP32, nhà phát triển có nhiều lựa chọn về thư viện đồ họa. Mỗi thư viện có những ưu điểm, nhược điểm và trường hợp sử dụng riêng. Việc phân tích và so sánh các giải pháp này giúp lựa chọn công cụ phù hợp nhất cho dự án cụ thể.

### **2.6.1 Các thư viện đồ họa hiện có**

Thị trường thư viện đồ họa cho ESP32 khá đa dạng, với nhiều lựa chọn từ các thư viện đơn giản, nhẹ nhàng đến các framework GUI toàn diện. Dưới đây là phân tích chi tiết về các thư viện phổ biến nhất.

#### **2.6.1.1 TFT\_eSPI (Bodmer)**

TFT\_eSPI là một thư viện đồ họa mạnh mẽ và được tối ưu hóa cao cho ESP32 và các vi điều khiển khác, được phát triển bởi Bodmer. Thư viện này được thiết kế đặc biệt cho màn hình TFT sử dụng giao tiếp SPI.

Ưu điểm chính của TFT\_eSPI bao gồm hiệu suất cao nhờ tối ưu hóa mã assembly và sử dụng DMA, hỗ trợ nhiều loại controller màn hình (ILI9341, ST7789, ILI9488, ...), và tích hợp tốt với hệ sinh thái Arduino. Thư viện cung cấp các hàm vẽ cơ bản như điểm, đường, hình chữ nhật, hình tròn, và văn bản, cùng với hỗ trợ hiển thị hình ảnh và sprite (đối tượng đồ họa có thể di chuyển).

Tuy nhiên, TFT\_eSPI có một số hạn chế. Thư viện không cung cấp các widget GUI cao cấp như nút bấm, thanh trượt, hoặc menu, khiến việc xây dựng giao diện người dùng phức tạp trở nên khó khăn hơn. Cấu hình thư viện cũng khá phức tạp, đòi hỏi chỉnh sửa file `User_Setup.h` để phù hợp với phần cứng cụ thể.

TFT\_eSPI phù hợp nhất cho các ứng dụng yêu cầu hiệu suất cao, hiển thị đồ họa cơ bản, hoặc khi nhà phát triển muốn xây dựng GUI tùy chỉnh từ đầu.

#### **2.6.1.2 Adafruit GFX**

Adafruit GFX là một thư viện đồ họa phổ biến và dễ sử dụng, được phát triển bởi Adafruit Industries. Thư viện này cung cấp một API nhất quán cho nhiều loại màn hình

khác nhau, từ OLED đơn sắc đến TFT màu.

Ưu điểm của Adafruit GFX bao gồm tính đơn giản và dễ học, hỗ trợ rộng rãi từ cộng đồng, và tương thích với nhiều loại màn hình thông qua các thư viện driver riêng (như `Adafruit_ILI9341`, `Adafruit_SSD1306`). Thư viện cung cấp các hàm vẽ cơ bản tương tự như `TFT_eSPI`, cùng với hỗ trợ font và hiển thị bitmap.

Tuy nhiên, Adafruit GFX không được tối ưu hóa cho hiệu suất cao như `TFT_eSPI`, đặc biệt trên ESP32. Thư viện cũng thiếu các widget GUI và hệ thống quản lý sự kiện, đòi hỏi nhà phát triển phải xây dựng các thành phần này từ đầu.

Adafruit GFX là lựa chọn tốt cho người mới bắt đầu, các dự án đơn giản, hoặc khi cần tương thích với nhiều loại màn hình khác nhau trong cùng một codebase.

### **2.6.1.3 U8g2**

U8g2 là một thư viện đồ họa monochrome (đơn sắc) được tối ưu hóa cho các màn hình OLED và LCD đơn sắc. Mặc dù chủ yếu tập trung vào hiển thị đơn sắc, U8g2 vẫn là một lựa chọn quan trọng trong hệ sinh thái ESP32.

Ưu điểm của U8g2 bao gồm kích thước nhỏ gọn, tiêu thụ bộ nhớ thấp, và hỗ trợ rộng rãi cho các controller màn hình đơn sắc (`SSD1306`, `SH1106`, ...). Thư viện cung cấp nhiều font với kích thước khác nhau, hỗ trợ nhiều ngôn ngữ, và có cơ chế buffer kép để tránh hiện tượng nhấp nháy khi cập nhật màn hình.

Hạn chế chính của U8g2 là chỉ hỗ trợ màn hình đơn sắc, không phù hợp cho các ứng dụng yêu cầu hiển thị màu. Thư viện cũng thiếu các widget GUI và hệ thống quản lý sự kiện như các thư viện khác.

U8g2 là lựa chọn tuyệt vời cho các ứng dụng sử dụng màn hình OLED đơn sắc, đặc biệt khi tài nguyên hệ thống hạn chế hoặc khi tiêu thụ năng lượng là ưu tiên hàng đầu.

### **2.6.1.4 LVGL**

LVGL (Light and Versatile Graphics Library) là một thư viện GUI toàn diện, cung cấp không chỉ các hàm vẽ cơ bản mà còn bao gồm hệ thống widget phong phú, quản lý sự kiện, và nhiều tính năng cao cấp khác.

Như đã phân tích chi tiết trong phần trước, LVGL có nhiều ưu điểm nổi bật so với các thư viện khác. Thư viện cung cấp bộ widget phong phú (nút bấm, thanh trượt, bảng, biểu đồ, ...), hệ thống styles và themes linh hoạt, và hỗ trợ animation. LVGL được thiết kế để hoạt động hiệu quả trên các hệ thống nhúng với tài nguyên hạn chế, yêu cầu chỉ 64KB flash và 16KB RAM cho các tính năng cơ bản.

LVGL cũng có khả năng mở rộng cao, cho phép nhà phát triển tạo ra các widget tùy chỉnh hoặc mở rộng các widget có sẵn. Thư viện hỗ trợ nhiều loại thiết bị đầu vào (cảm

ứng, nút bấm, encoder) và có thể hoạt động với nhiều loại màn hình khác nhau thông qua hệ thống driver linh hoạt.

Tuy nhiên, LVGL có đường cong học tập dốc hơn so với các thư viện đơn giản như Adafruit GFX, và cấu hình ban đầu có thể phức tạp. Thư viện cũng yêu cầu nhiều tài nguyên hơn so với các giải pháp đơn giản hơn, mặc dù vẫn được tối ưu hóa cho hệ thống nhúng.

LVGL là lựa chọn lý tưởng cho các ứng dụng yêu cầu GUI phức tạp, chuyên nghiệp, với nhiều widget tương tác và hiệu ứng đồ họa.

## **2.6.2 So sánh và lựa chọn thư viện phù hợp**

Khi lựa chọn thư viện đồ họa cho dự án ESP32, cần cân nhắc nhiều yếu tố khác nhau để đưa ra quyết định phù hợp nhất.

### **2.6.2.1 Hiệu suất và tài nguyên**

Về hiệu suất render, TFT\_eSPI thường dẫn đầu nhờ tối ưu hóa assembly và sử dụng DMA, tiếp theo là LVGL với engine render được tối ưu hóa. Adafruit GFX và U8g2 có hiệu suất thấp hơn nhưng vẫn đủ cho nhiều ứng dụng.

Về tiêu thụ bộ nhớ, U8g2 là nhẹ nhất, tiếp theo là TFT\_eSPI và Adafruit GFX. LVGL yêu cầu nhiều bộ nhớ nhất, đặc biệt khi sử dụng nhiều widget và animation, nhưng vẫn được tối ưu hóa cho hệ thống nhúng.

### **2.6.2.2 Tính năng và khả năng mở rộng**

LVGL dẫn đầu về tính năng với bộ widget phong phú, hệ thống styles, và animation. TFT\_eSPI cung cấp các tính năng đồ họa cơ bản mạnh mẽ nhưng thiếu widgets. Adafruit GFX và U8g2 tập trung vào các chức năng vẽ cơ bản.

Về khả năng mở rộng, LVGL có kiến trúc module hóa cho phép mở rộng dễ dàng. TFT\_eSPI cũng khá linh hoạt nhưng đòi hỏi nhiều công sức hơn để mở rộng. Adafruit GFX và U8g2 có khả năng mở rộng hạn chế hơn.

### **2.6.2.3 Độ phức tạp và đường cong học tập**

Adafruit GFX có đường cong học tập thoải nhất, phù hợp cho người mới bắt đầu. U8g2 và TFT\_eSPI có độ phức tạp trung bình, trong khi LVGL có đường cong học tập dốc nhất do cung cấp nhiều tính năng và khái niệm phức tạp hơn.

#### 2.6.2.4 Hỗ trợ cộng đồng và tài liệu

Tất cả bốn thư viện đều có cộng đồng người dùng lớn và tài liệu tốt. Adafruit GFX có lợi thế về hướng dẫn và ví dụ từ Adafruit. LVGL có tài liệu toàn diện nhất với hướng dẫn, ví dụ, và tài liệu tham khảo API đầy đủ. TFT\_eSPI và U8g2 cũng có tài liệu tốt và cộng đồng hỗ trợ tích cực.

#### 2.6.2.5 Trường hợp sử dụng phù hợp

- **TFT\_eSPI:** Phù hợp nhất cho các ứng dụng yêu cầu hiệu suất cao, hiển thị đồ họa cơ bản, hoặc khi tài nguyên hệ thống hạn chế nhưng vẫn cần hiển thị màu.
- **Adafruit GFX:** Lý tưởng cho người mới bắt đầu, các dự án đơn giản, hoặc khi cần tương thích với nhiều loại màn hình khác nhau.
- **U8g2:** Tốt nhất cho các ứng dụng sử dụng màn hình OLED đơn sắc, đặc biệt khi tiêu thụ năng lượng và bộ nhớ là ưu tiên.
- **LVGL:** Phù hợp nhất cho các ứng dụng yêu cầu GUI phức tạp, chuyên nghiệp, với nhiều widget tương tác và hiệu ứng đồ họa.

### 2.6.3 Tại sao chọn LVGL cho dự án phát triển thư viện giao diện

Sau khi phân tích các giải pháp GUI hiện có, LVGL nổi bật như một lựa chọn tối ưu cho việc phát triển thư viện giao diện cho ESP32 vì nhiều lý do.

- **Tính năng phong phú:** LVGL cung cấp một bộ widget phong phú và hệ thống styles linh hoạt, cho phép tạo ra giao diện người dùng hấp dẫn và chuyên nghiệp mà không cần phát triển từ đầu. Điều này giúp tiết kiệm thời gian phát triển đáng kể và mang lại kết quả chất lượng cao hơn.
- **Hiệu suất tối ưu:** LVGL được tối ưu hóa cho các hệ thống nhúng như ESP32, cân bằng giữa tính năng phong phú và hiệu suất. Thư viện có thể hoạt động hiệu quả với tài nguyên hạn chế của ESP32, đồng thời vẫn cung cấp trải nghiệm người dùng mượt mà.
- **Khả năng mở rộng:** Kiến trúc module hóa của LVGL cho phép mở rộng và tùy chỉnh dễ dàng, lý tưởng cho việc phát triển thư viện giao diện tùy chỉnh. Nhà phát triển có thể xây dựng các widget mới hoặc mở rộng các widget có sẵn để đáp ứng nhu cầu cụ thể của dự án.

- **Hỗ trợ cộng đồng và tài liệu:** LVGL có cộng đồng người dùng lớn, tài liệu toàn diện, và được cập nhật thường xuyên. Điều này đảm bảo rằng thư viện giao diện được phát triển sẽ có nền tảng vững chắc và hỗ trợ lâu dài.

Với những ưu điểm trên, LVGL là nền tảng lý tưởng để phát triển thư viện giao diện cho ESP32, cung cấp sự cân bằng tối ưu giữa tính năng, hiệu suất, và khả năng mở rộng.

## Tài liệu tham khảo

- [1] Karl Johan Astrom, Richard M. Murray, *Feedback Systems: An Introduction for Scientists and Engineers*, Princeton University Press, 2012.
- [2] A Joukhadar, I Hasan, A Alsabbagh, M Alkouzbary, *Integral Lqr-Based 6dof Autonomous Quadcopter Balancing System Control*, International Journal of Advanced Research in Artificial Intelligence, Vol. 4, No.5, 2015.
- [3] Nur Hayati Sahrir, Mohd Ariffanan Mohd Basri, *Modelling and Manual Tuning PID Control of Quadcopter*, Control, Instrumentation and Mechatronics: Theory and Practice (pp.346-357).
- [4] Nguyễn Đình Huy, *Giáo trình giải tích 1*, Đại học Quốc gia TP. Hồ Chí Minh, 2020.
- [5] Faisal Iqbal, Hussamud Din, Byeungleul Lee, *Single Drive Multi-Axis Gyroscope with High Dynamic Range, High Linearity and Wide Bandwidth*, Micromachines 2019, 10(6), 410.
- [6] Heja Cengiz, *Quadcopter Modeling and Linear Quadratic Regulator Design Using Simulink*, Uppsala Universitet, 2024.
- [7] Ha Quang Thinh Ngo, Thanh Phuong Nguyen, Hung Nguyen, "A COMPLETE COMPARISON TO DESIGN COMPLEMENTARY FILTER AND KALMAN FILTER FOR AERIAL VEHICLE", *International Journal of Mechanical Engineering and Technology*, 2018.
- [8] Rio Ikhsan Alfian, Alfian Ma'arif, Sunardi Sunardi, "Noise Reduction in the Accelerometer and Gyroscope Sensor with the Kalman Filter Algorithm", *Journal of Robotics and Control (JRC)*, 2021.
- [9] Luis E. Romero, David F. Pozo, Jorge A. Rosales, "Quadcopter Stabilization by Using PID Controllers", *ACADEMIA*, 2014.

- [10] Yan Michalevsky, Dan Boneh, "Gyrophone: Recognizing Speech from Gyroscope Signals", *Proceedings of the 23rd USENIX Security Symposium*, 2014.
- [11] Dennis Freeman, Kevin Chen "Linear quadratic regulator (LQR) control" Source: [https://introcontrol.mit.edu/\\_static/spring23/lectures/lec08a-handout.pdf](https://introcontrol.mit.edu/_static/spring23/lectures/lec08a-handout.pdf).
- [12] "MPU-6000 and MPU-6050 Product Specification Revision 3.4", 2013. Source: [www.invensense.com](http://www.invensense.com).
- [13] "Hướng dẫn sử dụng cảm biến gia tốc MPU6050 với Arduino", 2023. Source: <https://arduino.vn/huong-dan-su-dung-cam-bien-gia-toc-mpu6050-voi-arduino/>.
- [14] "tkinter — Python interface to Tcl/Tk". Source: <https://docs.python.org/3/library/tkinter.html>.
- [15] "socket — Low-level networking interface". Source: <https://docs.python.org/3/library/socket.html>.
- [16] "Arduino Documentation". Source: <https://docs.arduino.cc/>.
- [17] "Distance Measurement with an Ultrasonic Sensor HY-SRF05", 2017. Source: [https://projecthub.arduino.cc/Nicholas\\_N/distance-measurement-with-an-ultrasonic-sensor-hy-srf05-bf2923](https://projecthub.arduino.cc/Nicholas_N/distance-measurement-with-an-ultrasonic-sensor-hy-srf05-bf2923).
- [18] "ESC Calibration". Source: <https://ardupilot.org/plane/docs/common-esc-calibration.html>.
- [19] Wikipedia, "Quadcopter". Source: <https://en.wikipedia.org/wiki/Quadcopter>.
- [20] STMicroelectronics, "Everything about STMicroelectronics' 3-axis digital MEMS gyroscopes". Source: <https://www.elecrow.com/download/TA0343.pdf>.
- [21] Teppo Luukkonen, "Modelling and control of quadcopter". Source: [https://sal.aalto.fi/publications/pdf-files/eluu11\\_public.pdf](https://sal.aalto.fi/publications/pdf-files/eluu11_public.pdf).
- [22] "Accelerometer and Gyroscopes Sensors: Operation, Sensing, and Applications". Source: <https://www.analog.com/en/resources/technical-articles/accelerometer-and-gyroscopes-sensors-operation-sensing-and-applications.html>.
- [23] "Single Drive Multi-Axis Gyroscope with High Dynamic Range, High Linearity and Wide Bandwidth". Source: <https://www.mdpi.com/2072-666X/10/6/410>.



- [24] Michel van Biezen, "SPECIAL TOPICS 1 - THE KALMAN FILTER". Source: <https://www.youtube.com/watch?v=CaCcOwJPYtQ&list=PLX2gX-ftPVXU3oUFNATxGXY90AULiqnWT&index=1>.
- [25] Kevin Jordan, "Self-Stabilizing Quadcopter UAV Using PID Control: Full Control Systems Project Presentation". Source: <https://www.youtube.com/watch?v=clyusOrMqbU>.
- [26] Curio Res, "How Gyroscope Sensor Works? | 3D Animated". Source: <https://www.youtube.com/watch?v=HJ-C4Incgpw>.
- [27] Blue Butterfly, "How to design and implement a digital low-pass filter on an Arduino". Source: <https://www.youtube.com/watch?v=REVP33SwwHE>.
- [28] Dr. KC Craig, "Quadrotor Equations of Motion and Control KCC Final 4 2023 Video". Source: <https://www.youtube.com/watch?v=REVP33SwwHE>.
- [29] thegioiic, "Tìm hiểu về chuẩn giao tiếp I2C". Source: <https://www.thegioiic.com/tin-tuc/tim-hieu-ve-chuan-giao-tiep-i2c>.
- [30] Timothy Hirzel, "Basics of PWM (Pulse Width Modulation)". Source: <https://docs.arduino.cc/learn/microcontrollers/analog-output/>.
- [31] Carbon Aeronautics, "Carbon Aeronautics Quadcopter Manual". Source: [https://github.com/CarbonAeronautics/Manual-Quadcopter-Drone/blob/main/Carbon\\_Aeronautics\\_Quadcopter\\_Manual.pdf](https://github.com/CarbonAeronautics/Manual-Quadcopter-Drone/blob/main/Carbon_Aeronautics_Quadcopter_Manual.pdf).
- [32] LinhKienRC, "Combo điều khiển TX – RX Flysky FS i6". Source: <https://bandochoi.net/combo-dieu-khien-tx-rx-flysky-fs-i6.html>.
- [33] nshopvn, "Module thu phát Wifi ESP8266 NodeMCU Lua CP2102". Source: <https://nshopvn.com/product/module-thu-phat-wifi-esp8266-nodemcu-lua-cp2102/>.
- [34] Sam Market, "Teensy 4.0 (Headers)". Source: <https://market.samm.com/teensy-4-0-headers-en>.
- [35] hshop, "Cảm biến GY-521 6DOF IMU MPU6050". Source: <https://hshop.vn/cam-bien-6-dof-bac-tu-do-gy-521-mpu6050>.
- [36] nshopvn, "Cảm biến áp suất IIC I2C và nhiệt độ của BMP280 3.3 V". Source: <https://nshopvn.com/product/cam-bien-ap-suat-iic-i2c-va-nhiet-do-cua-bmp280-3-3-v/>.

- [37] Nguyễn Hiền, "Mạch điều khiển tốc độ động cơ không chổi than ESC 30A 4V-16V (Pin 2S-4S) BEC 5V". Source: <https://dientunguyenhien.vn/show/3252>.
- [38] nshopvn, "Động cơ không chổi than A2212". Source: <https://nshopvn.com/product/dong-co-khong-choi-than-a2212/>.
- [39] FPT Jetking, "ESP8266 là gì? Tìm hiểu về module Wi-Fi phổ biến cho IoT". Source: <https://jetking.fpt.edu.vn/esp8266/>.
- [40] pjrc, "Using the Hardware Serial Ports". Source: [https://www.pjrc.com/teensy/td\\_uart.html](https://www.pjrc.com/teensy/td_uart.html)
- [41] x-engineer, "On-off control system". Source: <https://x-engineer.org/on-off-control-system>
- [42] webOS TV Developer, "Sensor Data for Motion Sensor". Source: <https://webostv.developer.lge.com/develop/guides/motion-sensor-sensor-data>.
- [43] Yahya Tawil, "Towards understanding IMU: Basics of Accelerometer and Gyroscope Sensors and How to Compute Pitch, Roll and Yaw Angles". Source: <https://atadiat.com/en/e-towards-understanding-imu-basics-of-accelerometer-and-gyroscope-sensors/>.
- [44] Nasser M. Abbasi, "Dynamics equations, kinematics, velocity and acceleration diagrams". Source: [https://www.12000.org/my\\_notes/dynamics\\_cheat\\_sheet/reportchapter2.htm](https://www.12000.org/my_notes/dynamics_cheat_sheet/reportchapter2.htm).