ĐOÀN THANH NIÊN CỘNG SẢN HỒ CHÍ MINH

**BAN CHẤP HÀNH TP. HỒ CHÍ MINH**

**——🙠🙢🟏🙠🙢 ——**

**CÔNG TRÌNH DỰ THI**  
**GIẢI THƯỞNG SINH VIÊN NGHIÊN CỨU KHOA HỌC EURÉKA**  
**LẦN THỨ XIX NĂM 2017**

TÊN CÔNG TRÌNH:

**MÀNG BỌT KHÍ NGHỆ THUẬT**

LĨNH VỰC NGHIÊN CỨU: **CÔNG NGHỆ THÔNG TIN**

CHUYÊN NGÀNH: **ĐIỆN TỬ VIỄN THÔNG**

Mã số công trình: …………………………….

Mã số công trình: …………………………….

(Phần này do BTC Giải thưởng ghi)

***Ghi chú:***(Không ghi tên trường, tên sinh viên thực hiện, giáo viên hướng dẫn ở trang bìa và trong toàn bộ các trang của đề tài nghiên cứu. Đề nghị sinh viên ghi rõ công trình nghiên cứu thuộc lĩnh vực, nhóm ngành nào theo thể lệ của Ban tổ chức.)

# **MỞ ĐẦU**

Đã từ rất lâu, biểu diễn nước, bọt khí nghệ thuật đã không còn đơn thuần là cái gì đó tốn kém và chỉ mang tính chất biểu diễn nữa. Thay vào đó, người ta đã để ý hơn tới hiệu quả kinh tế và mỹ quan của nó. Điều đặc biệt ở các công trình này là khả năng thu hút sự tò mò thích thú cho người xem. Đó cũng chính là điều mà các nhà chiến lược quảng bá hình ảnh rất được quan tâm.

Ở nước ta hiện nay, một số công trình nghệ thuật nước đã có mặt từ rất lâu như Nhạc nước, Màn hình nước chiếu tia laze … nhưng “BUBBLE GRAPHIC DISPLAY” đối với nhiều người vẫn còn khá xa lạ và mới mẻ, vì trong nước chưa có một cơ quan tổ chức nào thực hiện đề tài này.

Nội dung của đề tài nghiên cứu này trình bày cách thức tiếp cận và xây dựng “Hệ thống điều khiển bọt khí nghệ thuật” sử dụng vi điều khiển STM32F4 giao tiếp với máy tính.

MỤC LỤC

[MỞ ĐẦU 2](#_Toc468533813)

[MỤC LỤC 3](#_Toc468533814)

[CHƯƠNG 1: GIỚI THIỆU 5](#_Toc468533815)

[1.1. Giới thiệu về Bubble Graphic Display 5](#_Toc468533816)

[1.2. Nội dung, mục tiêu và giới hạn của đề tài 6](#_Toc468533817)

[*1.2.1. Nội dung, mục tiêu đề tài* 6](#_Toc468533818)

[*1.2.2. Giới hạn đề tài* 6](#_Toc468533819)

[CHƯƠNG 2: KIẾN THỨC NỀN TẢNG 7](#_Toc468533820)

[2.1. Đặt vấn đề 7](#_Toc468533821)

[2.2. Vi điều khiển STM32F4 và một số linh kiện điện tử 7](#_Toc468533822)

[*2.2.1. Vi điều khiển STM32F4* 7](#_Toc468533823)

[2.2.1.1. Giới thiệu về Vi điều khiển STM32F4 7](#_Toc468533824)

[2.2.1.2. Các tính năng của STM32F4 9](#_Toc468533825)

[2.2.1.3. Giới thiệu KIT STM32F4 DISCOVERY 10](#_Toc468533826)

[*2.2.2. IC dịch 595* 11](#_Toc468533827)

[2.2.2.1. Giới thiệu về IC dịch 595 11](#_Toc468533828)

[2.2.2.2. Cấu tạo và hoạt động của IC dịch 595 12](#_Toc468533829)

[2.2.2.3. Sơ đồ chân và chức năng 12](#_Toc468533830)

[*2.2.3. MOSFET* 13](#_Toc468533831)

[2.2.3.1. Giới thiệu về MOSFET 13](#_Toc468533832)

[2.2.3.2. Cấu tạo và kí hiệu MOSFET 14](#_Toc468533833)

[2.2.3.3. Nguyên tắc hoạt động 14](#_Toc468533834)

[*2.2.4. Giao tiếp Vi điều khiển - PC theo chuẩn nối tiếp RS232* 15](#_Toc468533835)

[2.3. Kiến thức cơ bản về ảnh số và xử lý ảnh 16](#_Toc468533836)

[*2.3.1. Giới thiệu* 16](#_Toc468533837)

[*2.3.2. Lý thuyết ảnh màu cơ bản* 17](#_Toc468533838)

[2.3.2.1. Biểu diễn màu (Color Representation) 17](#_Toc468533839)

[2.3.2.2. Không gian màu (Color Space) 17](#_Toc468533840)

[2.3.2.3. Biểu diễn ảnh (Image Representation) 19](#_Toc468533841)

[*2.3.3. Các cách thức chuyển đổi ảnh cơ bản* 20](#_Toc468533842)

[2.3.3.1. Chuyển đổi ảnh màu sang ảnh xám 20](#_Toc468533843)

[2.3.3.2. Chuyển đổi ảnh xám sang ảnh nhị phân 21](#_Toc468533844)

[2.4. Tổng kết chương 21](#_Toc468533845)

[CHƯƠNG 3: HỆ THỐNG BUBBLE GRAPHIC DISPLAY 22](#_Toc468533846)

[3.1. Phân tích yêu cầu hệ thống 22](#_Toc468533847)

[3.2. Thiết kế tổng quan hệ thống 24](#_Toc468533848)

[CHƯƠNG 4: THIẾT KẾ MẠCH VÀ XÂY DỰNG CHƯƠNG TRÌNH ĐIỀU KHIỂN HỆ THỐNG BUBBLE GRAPHIC DISPLAY 25](#_Toc468533849)

[4.1. Thiết kế mạch điều khiển 25](#_Toc468533850)

[*4.1.1. Sơ đồ khối hệ thống và sơ đồ khối Kit STM32F4 Discovery* 25](#_Toc468533851)

[*4.1.2. Khối mạch Công suất* 27](#_Toc468533852)

[*4.1.3. Khối mạch 595* 30](#_Toc468533853)

[4.2. Xây dựng chương trình xử lý ảnh và điều khiển 31](#_Toc468533854)

[*4.2.1. Sơ đồ khối* 31](#_Toc468533855)

[*4.2.2. Chương trình xử lý ảnh trên máy tính* 32](#_Toc468533856)

[4.2.2.1. Giới thiệu 32](#_Toc468533857)

[4.2.2.2. Phần xử lý ảnh 32](#_Toc468533858)

[*4.2.2.2.1 Sơ đồ giải thuật* 32](#_Toc468533859)

[*4.2.2.2.2. Chuyển đổi kích thước ảnh theo tỷ lệ* 33](#_Toc468533860)

[*4.2.2.2.3. Chuyển đổi ảnh màu sang ảnh nhị phân* 34](#_Toc468533861)

[*4.2.2.2.4. Chuyển đổi ảnh nhị phân sang mảng 2 chiều* 35](#_Toc468533862)

[4.2.2.3. Phần truyền dữ liệu qua cổng COM 36](#_Toc468533863)

[*4.2.3. Chương trình điều khiển trên STM32F4* 40](#_Toc468533864)

[4.2.3.1. Giới thiệu 40](#_Toc468533865)

[4.2.3.2. Xây dựng chương trình điều khiển cho STM32F4 40](#_Toc468533866)

[4.3. Thiết kế lắp đặt dàn Solenoid 42](#_Toc468533867)

[4.4. Kết quả thực nghiệm thu được 43](#_Toc468533868)

[*4.4.1. Hệ thống mạch điều khiển* 43](#_Toc468533869)

[*4.4.2. Chương trình xử lý ảnh* 44](#_Toc468533870)

[*4.4.3. Kết quả thực nghiệm* 46](#_Toc468533871)

[4.5. Đánh giá 47](#_Toc468533872)

[5.1. Kết luận 48](#_Toc468533873)

[5.2. Hướng phát triển 48](#_Toc468533874)

[TÀI LIỆU THAM KHẢO 49](#_Toc468533875)

# 

# **CHƯƠNG 1: GIỚI THIỆU**

## **1.1. Giới thiệu về Bubble Graphic Display**

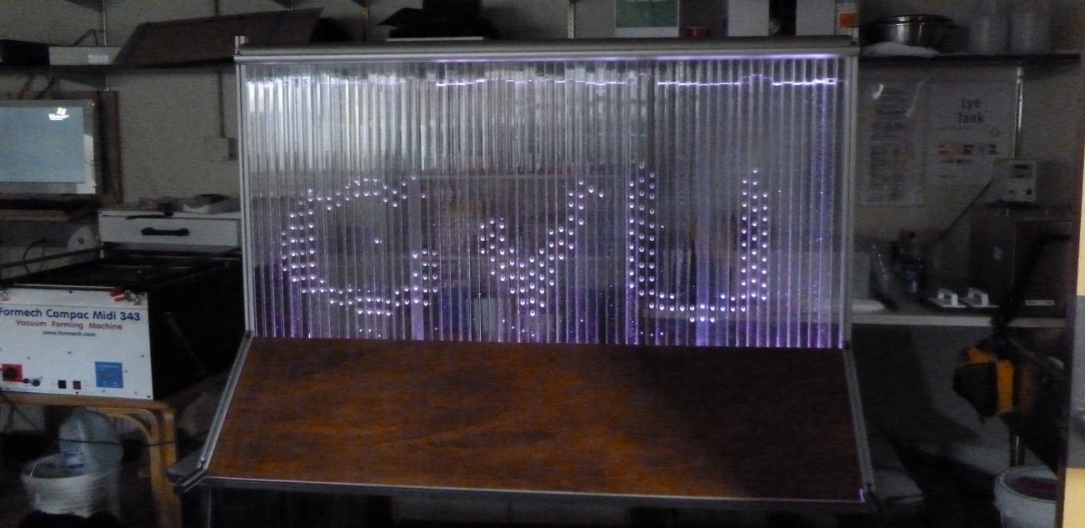
Hiện nay trên thế giới có rất nhiều công trình biểu diễn nghệ thuật tương tự như Bubble Graphic Display (Hệ thống điều khiển bọt khí nghệ thuật): Nhạc nước nghệ thuật, màn hình nước Laze, màn hình nước … Waterfall Graphic Display (hay còn gọi là Màn hình nước nghệ thuật) là một công trình biểu diễn nước rơi nghệ thuật với sự kết hợp của nước, vi mạch và lập trình. Máy tính điều khiển nước rơi thành hình, chữ, hiệu ứng … Màn hình nước nghệ thuật thường được lắp đặt biểu diễn ở những nơi công cộng như công viên, quảng trường … hoặc trong các trung tâm hội trợ triển lãm, khu thương mại để kích thích sự tò mò, thu hút người xem. Cho đến thời điểm hiện tại ở nước ta thì Bubble Graphic Display vô cùng lạ lẫm, mới mẻ vì chưa có cá nhân hay tổ chức nào thực hiện.



Hình 1.1: Màn hình nước nghệ thuật

Năm 2013 có nhóm nghiên cứu trên thế giới đã hiện thực thành công **BUBBLE GRAPHIC DISPLAY** sử dụng KIT phát triển ARDUINO. Những hình ảnh được hiển thị rất rõ ràng, đẹp mắt, chi phí rẻ hơn so với Waterfall Graphic Display.

*Nguồn*: <https://www.summet.com/blog/tag/bubbledisplay/>



Hình 1.2: Ảnh minh hoạ về Bubble Graphic Display

## **1.2. Nội dung, mục tiêu và giới hạn của đề tài**

### *1.2.1. Nội dung, mục tiêu đề tài*

Trong nội dung đề tài “*Bubble Graphic Display*” này, đề tài sẽ nghiên cứu tìm hiểu thiết kế mạch điện tử điều khiển sử dụng vi điều khiển STM32F4 Discovery thuộc dòng STM32 của hàng STMicroelectronics và xây dựng một chương trình xử lý trên máy tính qua giao tiếp với cổng COM. Máy tính sẽ xử lý dữ liệu đầu vào (văn bản, hình ảnh) đồng thời giao tiếp với mạch điện tử để điều khiển nước rơi tự do để đạt hiệu quả hiển thị. Ngoài hiển thị thông điệp văn bản, hình ảnh, một số hiệu ứng khác cũng có thể được lập trình. Với yêu cầu đề ra như vậy, đề tài tập trung giải quyết những vấn đề sau:

* Nghiên cứu thiết kế mạch điều khiển sử dụng Vi điều khiển STM32F4.
* Xây dựng chương trình xử lý ảnh trên máy tính.
* Lắp đặt một hệ thống “*Bubble Graphic Display*” thu nhỏ dài khoảng 1m gồm 20 van chạy thử nghiệm.

### *1.2.2. Giới hạn đề tài*

Để xây dựng một hệ thống “*Bubble Graphic Display*” hoàn chỉnh là một vấn đề lớn và tốn khá nhiều chi phí và thời gian bên cạnh đó do một số thiết bị (chẳng hạn như van Solenoid) ở Việt Nam chất lượng chưa cao cho nên hình ảnh hiển thị vẫn chưa sắc nét. Do đó đề tài chỉ dừng lại ở việc xây dựng một hệ thống “*Bubble Graphic Display*” căn bản điều khiển trực tiếp bởi máy tính, máy tính sẽ xử lý ảnh và gửi xuống STM32F4 điều khiển van Solenoid đóng mở để tạo hiệu quả hiển thị. Còn các khía cạnh nâng cao như điều khiển hệ thống “*Bubble Graphic Display*” qua Internet hoặc xử lý ảnh trực tiếp trên KIT từ thẻ nhớ… sẽ được nghiên cứu về sau.

# **CHƯƠNG 2: KIẾN THỨC NỀN TẢNG**

## **2.1. Đặt vấn đề**

Hệ thống “*Bubble Graphic Display*” là sự kết hợp chặt chẽ giữa lập trình, mạch điện tử và van khí điện từ. Đây là ba yếu tố quan trọng chính để xây dựng một hệ thống “*Bubble Graphic Display*” căn bản.

Ý tưởng thiết kế mạch cho hệ thống “*Bubble Graphic Display*” dựa trên hoạt động của mạch quang báo. Mạch quang báo sử dụng IC dịch để điều khiển một số lượng lớn LED. Màn hình nước cũng vậy, hệ thống cũng phải điều khiển đóng mở một số lượng lớn van điện từ. Chỉ có sự khác biệt là van Solenoid có công suất lớn hơn rất nhiều so với LED. Do đó ta cần tìm hiểu các linh kiện để thiết kế mạch công suất để điều khiển trực tiếp các van này. Để giao tiếp giữa máy tính và vi điều khiển ta vẫn sử dụng chuẩn RS232, một chuẩn rất thông dụng trong điều khiển.

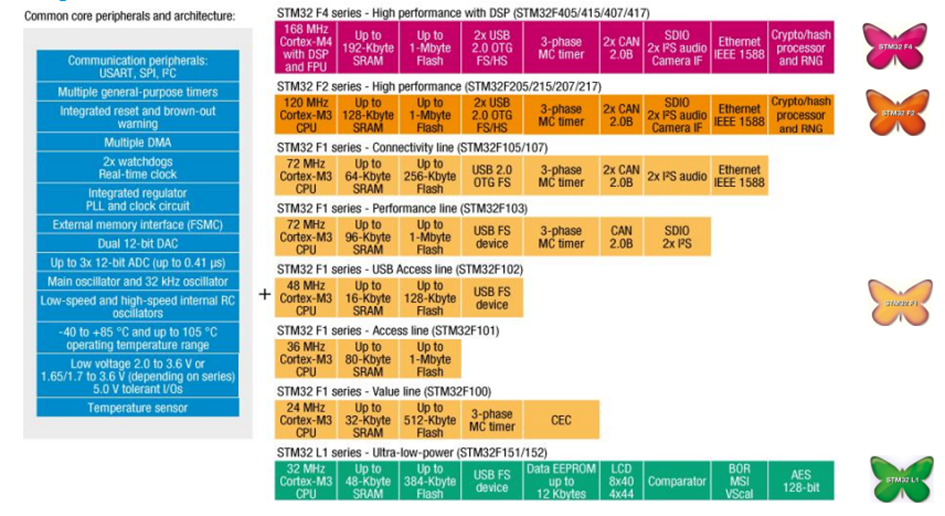
Về mảng lập trình, ta tìm hiểu một số kiến thức về xử lý ảnh trên máy tính căn bản cùng với đó là một số công thức chuyển đổi ảnh thông dụng. Do môi trường.NET hỗ trợ khá đầy đủ các thư viện xử lý ảnh và giao tiếp RS232 nên .NET và ngôn ngữ C# sẽ chọn để phát triển ứng dụng trên máy tính PC.

## **2.2. Vi điều khiển STM32F4 và một số linh kiện điện tử**

### *2.2.1. Vi điều khiển STM32F4*

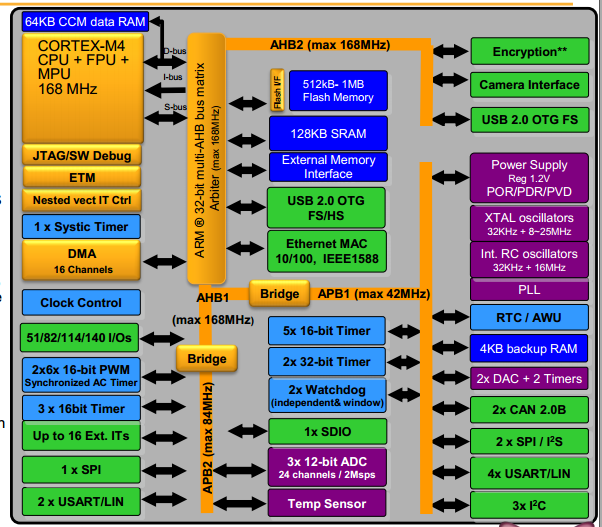
#### 2.2.1.1. Giới thiệu về Vi điều khiển STM32F4

Dòng ARM Cortex là một bộ xử lí thế hệ mới đưa ra một kiến trúc chuẩn cho nhu cầu đa dạng về công nghệ. Không giống như các chip ARM khác, dòng Cortex là một lõi xử lí hoàn thiện, đưa ra một chuẩn CPU và kiến trúc hệ thống chung. Dòng Cortex gồm có 3 phân nhánh chính: dòng A dành cho các ứng dụng cao cấp, dòng R dành cho các ứng dụng thời gian thực như các đầu đọc và dòng M dành cho các ứng dụng vi điều khiển và chi phí thấp. STM32 được thiết kế dựa trên dòng Cortex-M, dòng Cortex-M được thiết kế đặc biệt để nâng cao hiệu suất hệ thống, kết hợp với tiêu thụ năng lượng thấp, CortexM được thiết kế trên nền kiến trúc mới, do đó chi phí sản xuất đủ thấp để cạnh tranh với các dòng vi điều khiển 8 và 16-bit truyền thống.

****

Hình 2.1: Các dòng chip STM32

ST đã đưa ra thị trường 4 dòng vi điều khiển dựa trên ARM7 và ARM9, nhưng STM32 là một bước tiến quan trọng trên đường cong chi phí và hiệu suất (price/performance), giá chỉ gần 1 Euro với số lượng lớn, STM32 là sự thách thức thật sự với các vi điều khiển 8 và 16-bit truyền thống. STM32 đầu tiên gồm 14 biến thể khác nhau, được phân thành hai nhóm: dòng Performance có tần số hoạt động của CPU lên tới 72Mhz và dòng Access có tần số hoạt động lên tới 36Mhz. Các biến thể STM32 trong hai nhóm này tương thích hoàn toàn về cách bố trí chân (pin) và phần mềm, đồng thời kích thước bộ nhớ FLASH ROM có thể lên tới 128K và 20K SRAM.



Hình 2.2: Kiến trúc của STM32F4

Dòng STM32 có hai nhành, nhánh Performance hoạt động với xung nhịp lên đến 72Mhz và có đầy đủ các ngoại vi, nhánh Access hoạt động với xung nhịp tối đa 36Mhz và có ít ngoại vi hơn so với nhánh Performance.

#### 2.2.1.2. Các tính năng của STM32F4

1. **Sự tinh vi**

Thoạt nhìn thì các ngoại vi của STM32 cũng giống như những vi điều khiển khác, như hai bộ chuyển đổi ADC, timer, I2C, SPI, CAN, USB và RTC. Tuy nhiên mỗi ngoại vi trên đều có rất nhiều đặc điểm thú vị. Ví dụ như bộ ADC 12-bit có tích hợp một cảm biến nhiệt độ để tự động hiệu chỉnh khi nhiệt độ thay đổi và hỗ trợ nhiều mode chuyển đổi. Mỗi bộ timer có 4 khối capture compare, mỗi khối timer có thể liên kết với các khối timer khác để tạo ra một mảng các timer tinh vi.

1. **Sự an toàn**

Ngày nay các ứng dụng hiện đại thường phải hoạt động trong môi trường khắc khe, đòi hỏi tính an toàn cao, cũng như đòi hỏi sức mạnh xử lý và càng nhiều thiết bị ngoại vi tinh vi. Để đáp ứng các yêu cầu khắc khe đó, STM32 cung cấp một số tính năng phần cứng hỗ trợ các ứng dụng một cách tốt nhất. Chúng bao gồm một bộ phát hiện điện áp thấp, một hệ thống bảo vệ xung clock và hai bộ watchdogs.

1. **Tính bảo mật**

Một trong những yêu cầu khắc khe khác của thiết kế hiện đại là nhu cầu bảo mật mã chương trình để ngăn chặn sao chép trái phép phần mềm. Bộ nhớ Flash của STM32 có thể được khóa để chống truy cập đọc Flash thông qua cổng debug. Khi tính năng bảo vệ đọc được kích hoạt, bộ nhớ Flash cũng được bảo vệ chống ghi để ngăn chặn mã không tin cậy được chèn vào bảng vector ngắt.

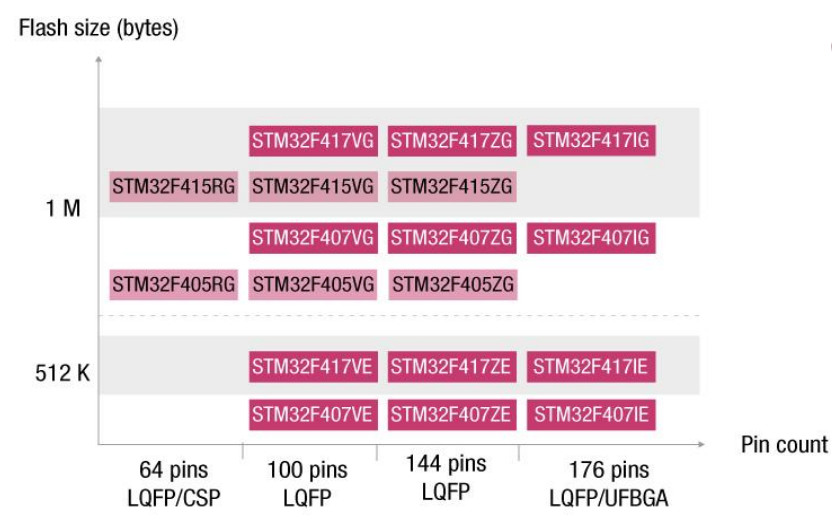
1. **Phát triển phần mềm**

Nếu như đã sử dụng một vi điều khiển dựa trên lõi ARM, các công cụ phát triển đã được hỗ trợ tập lệnh Thumb-2 và dòng Cortex. Ngoài ra ST cũng cung cấp một thư viện điều khiển thiết bị ngoại vi, một bộ thư viện phát triển USB như là một thư viện ANSI C và mã nguồn đó là tương thích với các thư viện trước đó được công bố cho vi điều khiển STR7 và STR9. Có rất nhiều RTOS mã nguồn mở và thương mại và middleware (TCP/IP, hệ thống tập tin, v.v.) hỗ trợ cho họ Cortex. Dòng Cortex-M3 cũng đi kèm với một hệ thống gỡ lỗi hoàn toàn mới gọi là CoreSight. Truy cập vào hệ thống CoreSight thông qua cổng truy cập Debug (Debug Access Port), cổng này hỗ trợ kết nối chuẩn JTAG hoặc giao diện 2 dây (serial wire-2 Pin), cũng như cung cấp trình điều khiển chạy gỡ lỗi.

1. **Dòng Performance và Access của STM32**

Họ STM32 có hai nhánh đầu tiên riêng biệt: dòng Performance và dòng Access. Dòng Performance tập hợp đầy đủ các thiết bị ngoại vi và chạy với xung nhịp tối đa 72MHz. Dòng Access có các thiết bị ngoại vi ít hơn và chạy tối đa 32MHz. Quan trọng hơn là cách bố trí chân (pins layout) và các kiểu đóng gói chip (package type) là như nhau giữa dòng Access và dòng Performance. Điều này cho phép các phiên bản khác nhau của STM32 được hoán vị mà không cần phải sửa đổi sắp sếp lại footprint (mô hình chân của chip trong công cụ layout bo mạch) trên PCB (Printed Circuit Board).

Ngoài hai dòng Performance và Access đầu tiên, hiện nay ST đã đưa ra thị trường thêm hai dòng USB Access và Connectivity như hình bên dưới.



Hình 2.3: Bộ nhớ và quy cách đóng gói dòng STM32F4

#### 2.2.1.3. Giới thiệu KIT STM32F4 DISCOVERY

Để thuận tiện cho việc học và nghiên cứu Vi xử lý ARM STM32, lựa chọn Kit Vi xử lý STM32F4 Discovery sẽ là sự thuận tiện cho việc học và nghiên cứu các chức năng cũng như việc lập trình các dòng ARM.

Ở đây hệ thống sẽ sử dụng vi điều khiển STM32F407VGT6, 100LQFP package

***Thông tin chung về sản phẩm:***

* Lõi: ARM Cortex M4F 32 bit.
* Bộ nhớ: 1MB Flash, 192 KB RAM.
* Tích hợp sẵn mạch nạp và debug ST-Link V2.
* Điện áp: sử dụng trực tiếp nguồn từ cổng USB, qua IC ổn áp 3.3V để nuôi mạch.
* Có sẵn cảm biến gia tốc: LIS302DL, ST MEMS motion sensor, 3-axis.
* Có sẵn bộ xử lý âm thanh: MP45DT02, ST MEMS audio sensor, omni-irectional digital microphone.
* Có sẵn bộ: CS43L22, audio DAC with integrated class D speaker driver.
* Có cổng Micro USB OTG.
* Tổng số I/O:
* ADC: 3x12 bit, 24 kênh 2.44MSPS
* DAC: 2x12 bit, 2 kênh
* DMA: Điều khiển 16 kênh DMA
* Timer: 10x16 bit, 2x32 bit
* Giao diện kết nối: 3xI2C, 6xUSART, 3xSPI, 2xCAN, 1xUSB 2.0 OTG FS/HS, 1x USB 2.0 OTG FS, Ethernet MAC 10/100 with IEEE 1588, …
* Kiểu chân: LQFP100

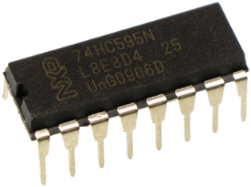


Hình 2.4: Kit STM32F4 Discovery

### *2.2.2. IC dịch 595*

#### 2.2.2.1. Giới thiệu về IC dịch 595

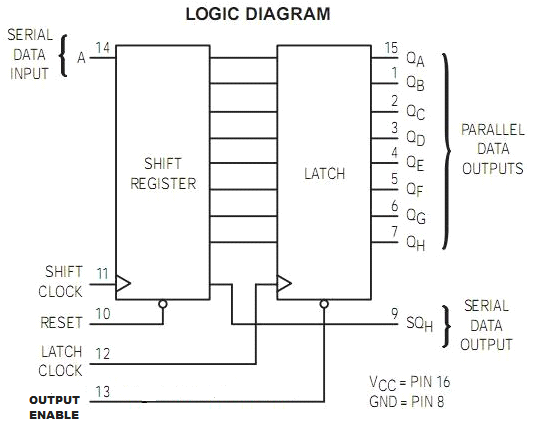
IC ghi dịch 8 bit kết hợp chốt dữ liệu, đầu vào nối tiếp đầu ra song song, thường dùng trong các mạch quét LED 7, LED Matrix … để tiết kiệm số chân vi điều khiển tối đa (3 chân). Có thể mở rộng số chân vi điều khiển bao nhiêu tùy thích mà không có IC nào có thể làm được bằng việc mắc nối tiếp đầu vào dữ liệu các IC với nhau.



Hình 2.5: IC dịch 595

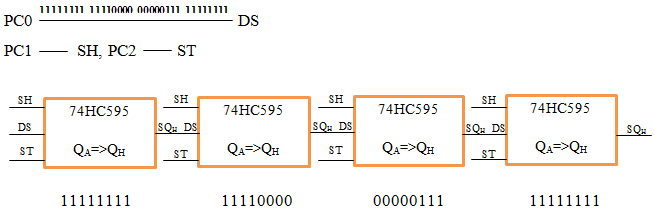
#### 2.2.2.2. Cấu tạo và hoạt động của IC dịch 595

Sơ đồ khối của IC dịch 595 được mô tả như trong Hình 2.6:



Hình 2.6: Sơ đồ khối IC dịch 595

Cách hoạt động của IC dịch 595 được mô tả trực quan như trong Hình 2.7:

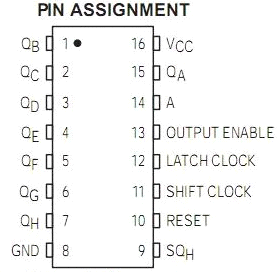


Hình 2.7: Nguyên tắc hoạt động IC dịch 595

Bit được đưa vào chân DS (A), chân SH (SHIFT CLOCK) có nhiệm vụ cho phép bit đó có được phép ghi vào thanh ghi dịch hay không, chân ST (LATCH CLOCK) có nhiệm vụ cho phép bit xuất ra ngoài hay không. Một IC 595 lưu trữ được 8 bit, nếu bit thứ 9 được đưa vào thì nó sẽ đẩy bit ở chân SQH (bit vào đầu tiên) ra chân SQH, bit này sẽ làm bit đầu vào cho thanh ghi dịch tiếp theo. Thông thường khi sử dụng nhiều IC 595 điều khiển, ta thường nối chân SQH của IC 595 này với chân DS của IC 595 tiếp theo.

#### 2.2.2.3. Sơ đồ chân và chức năng

Sơ đồ các chân IC dịch 595 được thể hiện trong hình 2.8.



Hình 2.8: Sơ đồ chân IC 595

Giải thích ý nghĩa hoạt động của một số chân quan trọng:

* Chân 14 (Input): Đầu vào dữ liệu nối tiếp. Tại một thời điểm xung clock chỉ đưa vào được 1 bit.
* Các chân QA=>QH trên các chân (15, 1, 2, 3, 4, 5, 6, 7) (Output): Xuất dữ liệu khi chân 13 tích cực ở mức thấp và có một xung tích cực ở sườn âm tại chân chốt 12.
* Chân 13 (Output Enable): Chân cho phép tích cực ở mức thấp (0). Khi ở mức cao, tất cả các đầu ra của 74HC595 trở về trạng thái cao trở, không có đầu ra nào được cho phép.
* Chân 9 (SQH): Chân dữ liệu nối tiếp. Nếu dùng nhiều 74HC595 mắc nối tiếp nhau thì chân này đưa vào đầu của con tiếp theo khi đã dịch đủ 8 bit.
* Chân 11 (Shift Clock): Chân vào xung clock. Khi có 1 xung clock tích cực ở sườn dương (từ 0 lên 1) thì 1 bit được dịch vào IC.
* Chân 12 (Latch Clock): Chân vào xung clock chốt dữ liệu. Khi có 1 xung clock tích cực ở sườn dương thì cho phép xuất dữ liệu trên các chân output. Lưu ý có thể xuất dữ liệu bất kỳ lúc nào ta muốn, ví dụ đầu vào chân 14 được 2 bit khi có xung clock ở chân 12 thì dữ liệu sẽ ra ở chân QA và QB (chiều dịch dữ liệu từ QA=>QH).
* Chân 10 (Reset): Khi chân này ở mức thấp (mức 0) thì dữ liệu sẽ bị xóa trên chip.

### *2.2.3. MOSFET*

#### 2.2.3.1. Giới thiệu về MOSFET

Metal Oxide Semiconductor Field Effect Transistor (MOSFET) là transistor hiệu ứng trường, một Transistor đặc biệt có cấu tạo và hoạt động khác với Transistor thông thường mà ta đã biết. Hình 2.9 thể hiện hình dáng và cấu tạo của một MOSFET trên thực tế. MOSFET có nguyên tắc hoạt động dựa trên hiệu ứng từ trường để tạo ra dòng điện, là linh kiện có trở kháng đầu vào lớn thích hợp cho khuyếch đại các nguồn tín hiệu yếu, MOSFET được sử dụng nhiều trong các mạch nguồn monitor, nguồn máy tính.



Hình 2.9: MOSFET

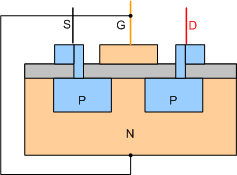
#### 2.2.3.2. Cấu tạo và kí hiệu MOSFET

Ký hiệu Mosfet trong sơ đồ nguyên lý như sau:



Hình 2.10: Ký hiệu MOSFET

#### 2.2.3.3. Nguyên tắc hoạt động



Hình 2.11. Cấu tạo MOSFET

Cấu tạo của MOSFET ngược Kênh N gồm các thành phần sau:

* G: Gate gọi là cực cổng.
* S: Source gọi là cực nguồn.
* D: Drain gọi là cực máng.

Trong đó: G là cực điều khiển được cách ly hoàn toàn với cấu trúc bán dẫn còn lại bởi lớp điện môi cực mỏng nhưng có độ cách điện cực lớn dioxit-silic (SiO2). Hai cực còn lại là cực gốc (S) và cực máng (D). Cực máng là cực đón các hạt mang điện.

MOSFET có điện trở giữa cực G với cực S và giữa cực G với cực D là vô cùng lớn, còn điện trở giữa cực D và cực S phụ thuộc vào điện áp chênh lệch giữa cực G và cực S (UGS).

Khi điện áp UGS = 0 thì điện trở RDS rất lớn, khi điện áp UGS > 0 do hiệu ứng từ trường làm cho điện trở RDS giảm. Điện áp UGS càng lớn thì điện trở RDS càng nhỏ.

### *2.2.4. Giao tiếp Vi điều khiển - PC theo chuẩn nối tiếp RS232*

Vấn đề giao tiếp giữa vi điều khiển và máy tính rất quan trọng trong các ứng dụng điều khiển, đo lường … Ghép nối qua cổng nối tiếp RS232 là một trong những kỹ thuật được sử dụng rộng rãi để ghép nối các thiết bị ngoại vi với máy tính. Nó là chuẩn giao tiếp nối tiếp dùng đinh dạng không đồng bộ, kết nối nhiều nhất là 2 thiết bị, chiều dài kết nối lớn nhất cho phép đảm bảo dữ liệu là 12.5 đến 25.4 m, tốc độ 20 kbit/s. Ý nghĩa chuẩn truyền thông nối tiếp là trong một thời điểm chỉ có một bit được gửi đi dọc đường truyền.

Trên STM32F4 sử dụng truyền thông nối tiếp không đồng bộ (UART) thường để chỉ thiết bị phần cứng, không phải chỉ một chuẩn giao tiếp. Với chuẩn RS232 trên các máy tính cá nhân là sự kết hợp của chip UART (chẳng hạn như STM32F4) và chip chuyển đổi mức điện áp.

Tín hiệu từ chip UART thường theo mức TTL:

* Ngõ vào: - Mức thấp (0): 0V ─ +0.8V

- Mức cao (1): +2V ─ +5V

* Ngõ ra: - Mức thấp (0): 0 ─ +0.5V

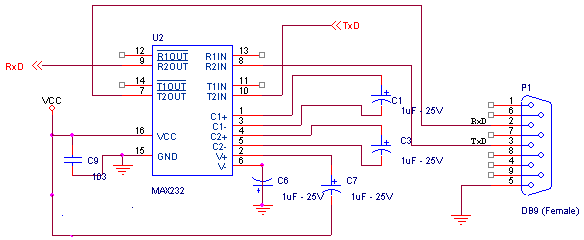
- Mức cao (1): +2V ─ +5V

Tín hiệu theo chuẩn RS232:

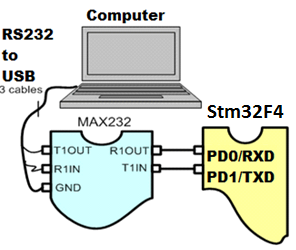
* Mức thấp (0): −12V ─ −3V
* Mức cao (1): +3V ─ +12V

Như thế, rõ ràng cần một “cầu chuyển” kết nối giữa 2 chuẩn này. Trên thị trường có loại IC chuyên dụng làm việc này. Đó là IC MAX232 chuyển UART-RS232 được sử dụng nhiều nhất.

Sơ đồ ghép nối với MAX232 được thể hiện trong hình sau:



Hình 2.12: Sơ đồ ghép nối MAX 232 với cổng COM



Hình 2.13: Kết nối RS232 với PC

## **2.3. Kiến thức cơ bản về ảnh số và xử lý ảnh**

### *2.3.1. Giới thiệu*

Để xây dựng 1 hệ thống Bubble Graphic Display thì vấn đề xử lý hình ảnh đóng vai trò quan trọng. Hệ thống yêu cầu phải xây dựng chương trình cho phép người dùng khi đưa một tấm ảnh màu với kích thước bất kì vào đều có thể hiển thị ra ngoài Màn hình bọt khí được.

Mục đích của việc xử lý ảnh trong chương trình điều khiển “bubble”:

* Chuyển ảnh màu sang ảnh nhị phân (ảnh trắng đen).
* Thay đổi kích thước ảnh theo tỉ lệ.
* Xử lý ảnh đen trắng trên pixels chuyển về mảng các giá trị 0, 1.

Để xây dựng 1 chương trình xử lý ảnh đơn giản, trước hết ta cần tìm hiểu thêm một số kiến thức cơ bản trong lập trình xử lý hình ảnh. Ở đây sử dụng ngôn ngữ C# và .NET framework.

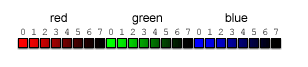
### *2.3.2. Lý thuyết ảnh màu cơ bản*

#### 2.3.2.1. Biểu diễn màu (Color Representation)

Màu sắc trên máy tính thưởng được biểu diễn bằng 2 cách thông dụng: 8-bit grayscale và 24-bit color.

Đối với thang màu xám 8-bit grayscale chứa 256 cấp độ của màu xám , trong đó 0 thể hiện cho màu đen và 255 thể hiện cho màu trắng ,các giá trị còn lại thể hiện các màu sắc khác nhau .

Còn đối với thang màu 24-bit là lưu 3 bytes thể hiện 3 thành phần màu bao gồm: đỏ (R), xanh (G) và xanh dương (B). Sự kết hợp của ba thành phần RGB có thể tạo nên một màu bất kỳ:



Hình 2.14: Biểu diễn màu [4]

Ngoài cách biểu diễn màu theo ba thành phần RGB còn có nhiều cách biểu diễn khác: HSI (Hue, Saturation, Intensity), CMY (Cyan, Magenta, Yellow), Normalized RG, CIE, YIQ …

#### 2.3.2.2. Không gian màu (Color Space)

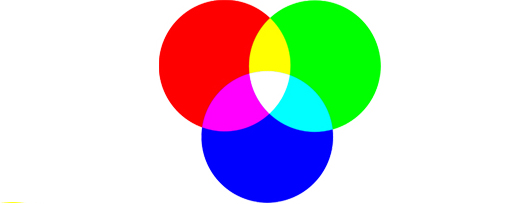
Không gian màu là một mô hình toán học dùng để mô tả các màu sắc trong thực tế được biểu diễn dưới dạng số học. Trên thực tế có rất nhiều không gian màu khác nhau được mô hình để sử dụng vào những mục đích khác nhau. Ba không gian màu cơ bản và được sử dụng phổ biến nhất sẽ được giới thiệu, bao gồm RGB, HSV, và CMYK.

**Không gian màu RGB:**

RGB là không gian màu rất phổ biến được dùng trong đồ họa máy tính và nhiều thiết bị kĩ thuật số khác. Ý tưởng chính của không gian màu này là sự kết hợp của 3 màu sắc cơ bản R, G, B để mô tả tất cả các màu sắc khác.

Nếu như một ảnh số được mã hóa bằng 24 bit, nghĩa là 8 bit cho kênh R, 8 bit cho kênh G, và 8 bit cho kênh B, thì mỗi kênh màu này sẽ nhận giá trị từ 0 đến 255. Với mỗi giá trị khác nhau của các kênh màu kết hợp với nhau ta sẽ được một màu khác nhau, như vậy ta sẽ có tổng cộng 2563 ≈ 1.66 triệu màu sắc.

Ví dụ: màu đen là sự kết hợp của các kênh màu (R, G, B) với giá trị tương ứng (0, 0, 0), màu trắng có giá trị (255, 255, 255), màu vàng có giá trị (255, 255, 0), màu tím đậm có giá trị (64, 0, 128).

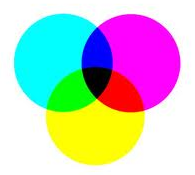


Hình 2.15: Không gian màu RBG [4]

**Không gian màu CMYK:**

CMYK là không gian màu được sử dụng phổ biến trong ngành công nghiệp in ấn. Ý tưởng cơ bản của hệ không gian này là dùng 4 màu sắc cơ bản để phục vụ cho việc pha trộn mực in. Trên thực tế, người ta dùng 3 màu là xanh lơ (Cyan – C), hồng xẫm (Magenta- M), và vàng (Yellow-Y) để biểu diễn các màu sắc khác nhau.

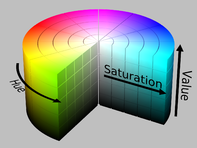
Ví dụ: Nếu lấy màu hồng xẫm cộng với vàng sẽ ra màu đỏ, màu xẫm kết hợp với xanh lơ sẽ cho xanh lam ... Sự kết hợp của 3 màu trên sẽ cho ra màu đen.



Hình 2.16: Không gian màu CYMK [4]

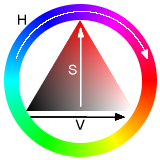
**Không gian màu HSV:**

HSV là không gian màu được dùng nhiều trong việc chỉnh sữa ảnh, phân tích ảnh và một phần của lĩnh vực thị giác máy tính. Hệ không gian này dựa vào 3 thông số sau để mô tả màu sắc: màu sắc (Hue - H), độ đậm đặc (Saturation - S), và giá trị cường độ sáng (Value - V). Không gian màu này thường được biểu diễn dưới dạng hình trụ hoặc hình nón như trong Hình 2.25:



Hình 2.17: Không gian màu HSV [4]

Không gian màu biểu diễn màu sắc (Hue) theo vòng tròn như hình 2.26 bên dưới:

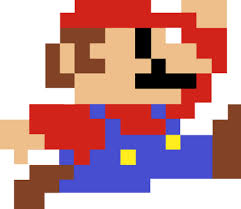


Hình 2.18: Hình tròn biểu diễn màu sắc (Hue) [4]

Như vậy với mỗi giá trị (H, S, V) sẽ cho ta một màu sắc mà ở đó mô tả đầy đủ thông tin về máu sắc, độ đậm đặc và độ sáng của màu đó.

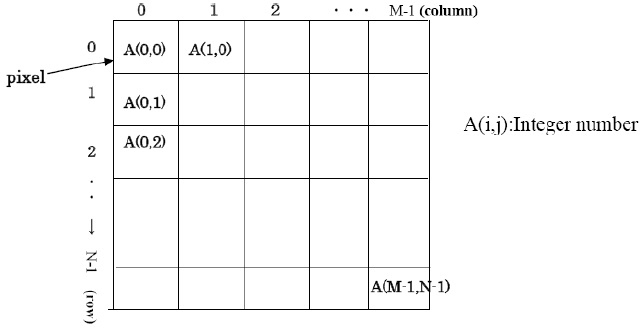
#### 2.3.2.3. Biểu diễn ảnh (Image Representation)

Ảnh số trên máy tính là một tập hợp các điểm ảnh nhỏ, mỗi điểm ảnh gọi là 1 pixe, mỗi điểm ảnh thể hiện một màu sắc nhất định và biểu diễn bằng một giá trị các bit.



Hình 2.19: Biểu diễn ảnh

Ví dụ trong hình 2.27, ảnh nhân vật Mario được tạo thành từ nhiều điểm ảnh khác nhau. Mỗi điểm ảnh được thể hiện bởi một màu sắc nhất định. Các điểm ảnh được sắp xếp như một mảng 2 chiều.

[](https://sites.google.com/site/dangtrieu/xulyanh/DigitalImage.jpg?attredirects=0)

Hình 2.20: Biểu diễn vị trí các pixel trên ảnh

**Phân loại: Có 3 loại ảnh:**

* Ảnh nhị phân (binary image): giá trị mỗi pixel là 0 hoặc 1, nghĩa là trắng hoặc đen. Trong thực tế khi xử lý trên máy tính thì người ta dùng ảnh xám (xem khái niệm bên dưới) để biểu diễn ảnh nhị phân và lúc này 2 giá trị là 0 hoặc 255.
* Ảnh xám (gray image): giá trị mỗi pixel nằm trong giải giá trị từ 0 đến 255, nghĩa là cần 8 bit hay 1 byte để biểu diễn mỗi pixel này.
* Ảnh màu (color image): mỗi pixel có giá trị gồm 3 RGB. Mỗi kênh màu có giá trị từ 0 đến 255, nghĩa là mỗi pixel cần 24 bit hay 3 byte để biểu diễn.



Ảnh màu  Ảnh xám Ảnh nhị phân

Hình 2.21: Các loại ảnh

### *2.3.3. Các cách thức chuyển đổi ảnh cơ bản*

#### 2.3.3.1. Chuyển đổi ảnh màu sang ảnh xám

Để chuyển từ ảnh màu sang ảnh xám, ta chỉ cần nhân giá trị các kênh màu của mỗi pixel với các hệ số cho trước:

|  |
| --- |
| *Y = 0.3\*Red + 0.59\*Green + 0.11\*Blue [1].* |

Xây dựng chương trình:

    + Kích thước của bức ảnh là X\*Y.

+ uchar c[3\*X][Y] là ma trận 2 chiều chứa dữ liệu của bức ảnh màu, 3 màu xếp theo thứ tự R, G, B.

    + uchar g[X][Y] là ma trận 2 chiều chứa dữ liệu của bức ảnh xám.

    + Duyệt từng pixels của cả bức ảnh (raster scan) bằng 2 vòng lặp và gán giá trị ảnh màu nhân với hệ số vào ảnh xám.

|  |
| --- |
| *for (i = 0; i <= X; i++) {*  *for (j = 0; j <= Y; j++) {*  *g[i][j] = 0.3\*c[3\*i][j] + 0.59\*c[3\*i+1][j] + 0.11\*c[3\*i+2][j];*  *} }* |

#### 2.3.3.2. Chuyển đổi ảnh xám sang ảnh nhị phân

Thao tác này được gọi là đặt ngưỡng, ta chỉ cần chọn một giá trị ngưỡng (threshold) thích hợp, pixel xám nào có giá trị bằng hoặc dưới ngưỡng đó ta cho bằng 0, trên ngưỡng đó ta cho bằng 255.

|  |
| --- |
| *if g[i][j] <= threshold         b[i][j] = 0; // b[i][j] là mảng 2 chiều chứa dữ liệu của bức ảnh nhị phân.     else         b[i][j] = 255;* |

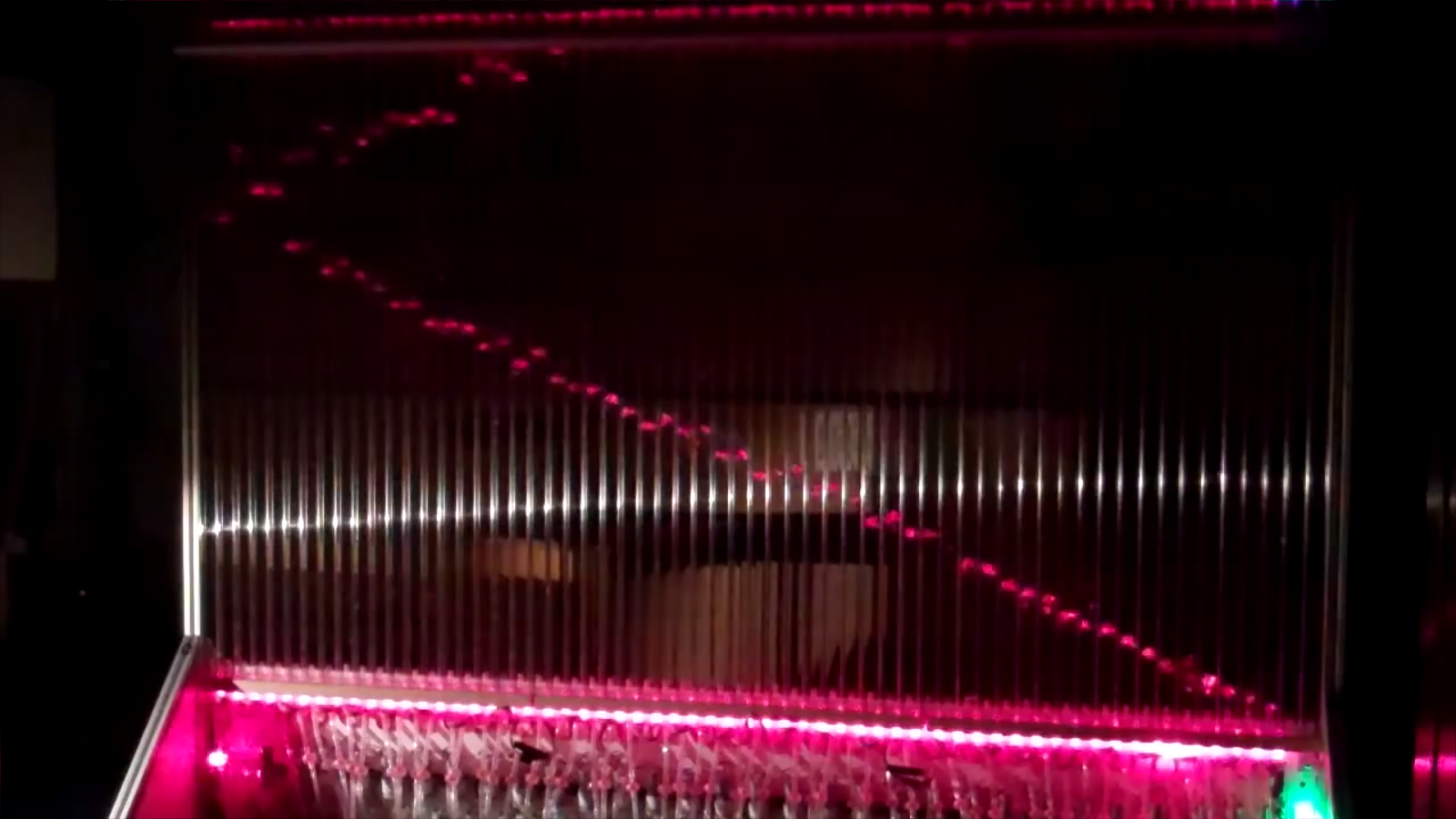
## **2.4. Tổng kết chương**

Như vậy, trong chương này đã đề cập đến một số linh kiện điện tử thông dụng sẽ được sử dụng để thiết kế mạch điều khiển và một số kiến thức căn bản hỗ trợ cho việc lập trình xử lý ảnh số trên máy tính. Đây là những kiến thức quan trọng làm nền tảng cho việc xây dựng một hệ thống Bubble Graphic Display.

# **CHƯƠNG 3: HỆ THỐNG BUBBLE GRAPHIC DISPLAY**

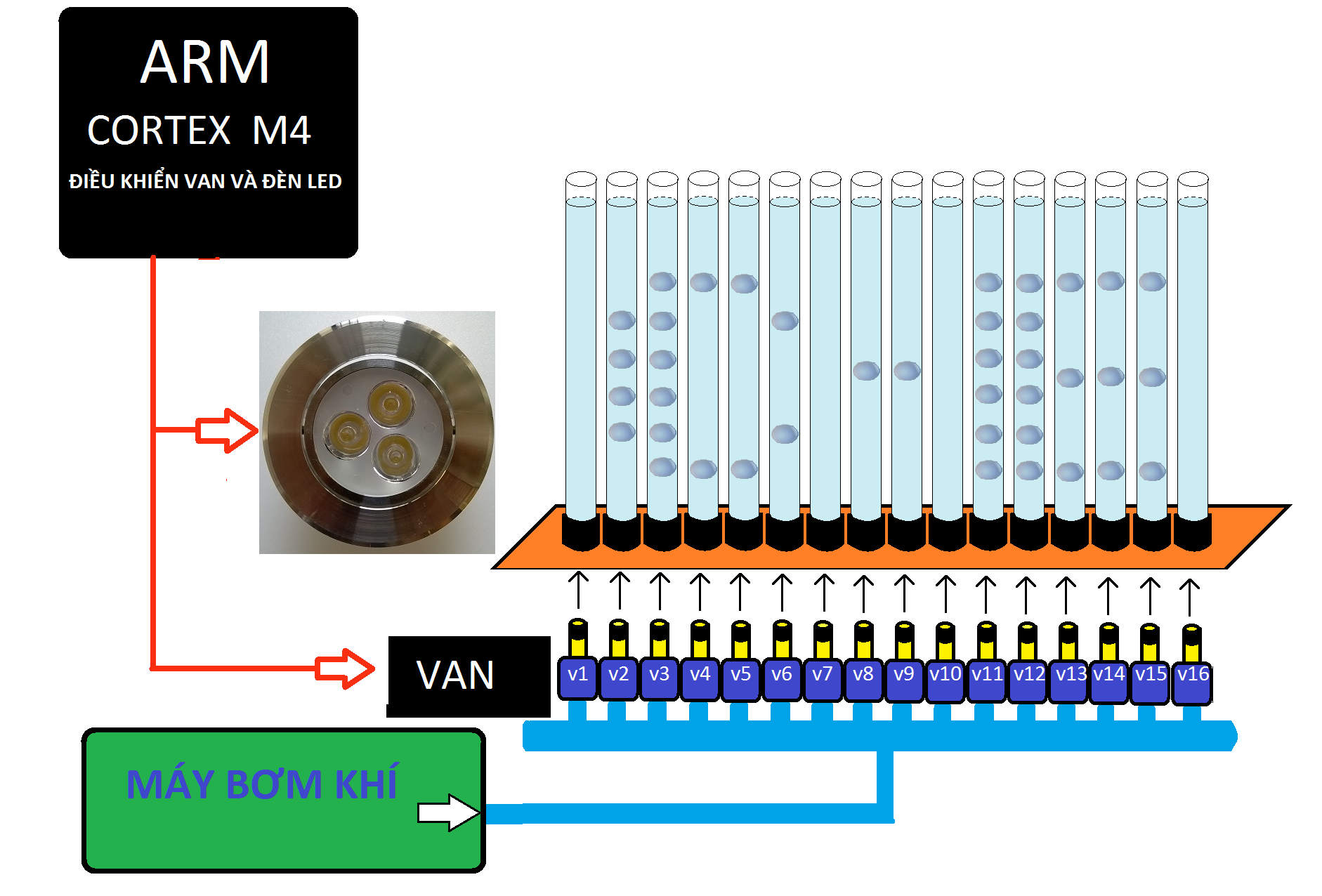
## **3.1. Phân tích yêu cầu hệ thống**

Bubble Graphic là hệ thống mà khi bọt khí nỗi tự do trong nước, các bọt khí kết hợp với nhau tạo thành hình ảnh, hiệu ứng …



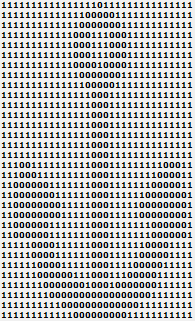
Hình 3.1: Bubble Graphic Display hoàn chỉnh

Các bọt khí được tạo bởi sự đóng mở liên tục của các van Solenoid (hay còn gọi là van điện tử). Các van Solenoid được đặt sát với nhau theo một hàng ngang, mỗi đầu van cách nhau khoảng 3cm. Các van này được điều khiển trực tiếp bởi một hệ thống mạch bên dưới. Khi van đóng mở ở tốc độ cao trong một khoảng thời gian ngắn (tính bằng ms) sẽ tạo ra bọt khí nỗi lên dưới áp lực đẩy khí từ máy bơm. Các bọt khí khi kết hợp lại sẽ tạo ra những hình ảnh, chữ số theo ý muốn.



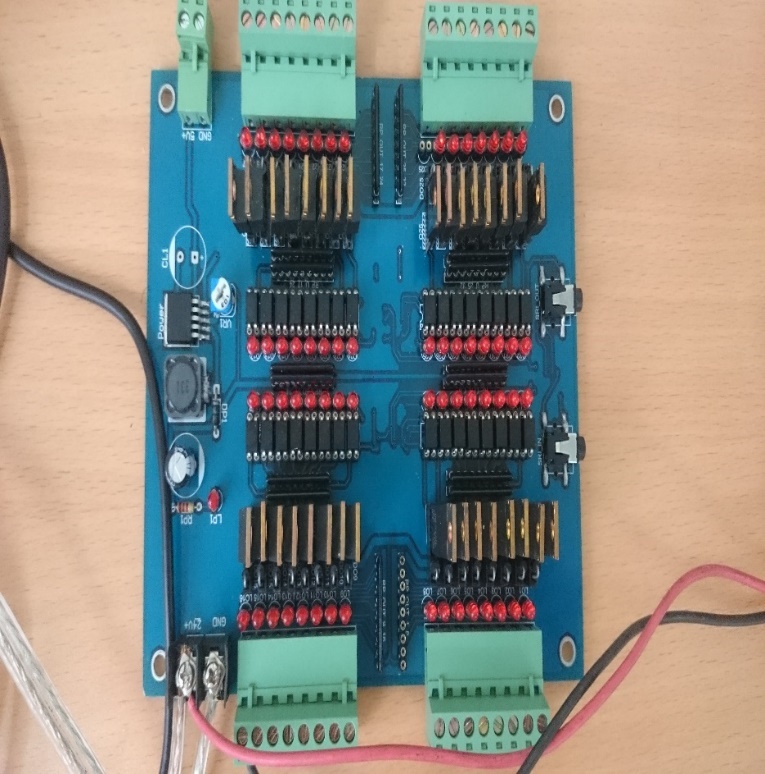
Hình 3.2: Hoạt động của van Solenoid

Ý tưởng xây dựng hệ thống dựa trên hoạt động của màn hình quang báo, hình ảnh được chuyển sang dạng nhị phân sau đó chuyển sang dạng mảng 2 chiều chỉ bao gồm các bit 0 và 1, những điểm ảnh có màu đen sẽ mang bit 0 còn điểm ảnh màu trắng mang bit 1, vi điều khiển dựa vào các tín hiệu này để điều khiển van Solenoid đóng mở. Nếu bit 0 thì van mở ngược lại thì đóng. Thời gian đóng mở van xảy ra trong một khoảng thời gian delay rất nhỏ, tạo thành nhiều bọt khí nỗi tự do liên tục. Các điểm bọt khí này khi liên kết với nhau sẽ tạo thành hình ảnh.



Hình 3.3: Ảnh nhị phân chuyển sang mảng 2 chiều

Việc xử lý ảnh này được thực hiện hoàn toàn trên máy tính, máy tính sẽ thực hiện toàn bộ việc xử lý ảnh sau đó mới gửi tín hiệu xuống hệ thống mạch điều khiển van. Vấn đề gặp phải là các van solenoid có điện áp khá lớn (khoảng 24V), vì thế khi hoạt động cùng lúc thì công suất của hệ thống trở nên rất lớn. Do đó không thể điều khiển các van này trực tiếp bằng Vi điều khiển được, mà sẽ được xử lý bởi các mạch Công suất. Các mạch Công suất này sẽ dựa vào tín hiệu từ Vi điều khiển và trực tiếp điều khiển hệ thống van solenoid.



Hình 3.4: Mạch công suất điều khiển van

Một vi điều khiển thông thường có số lượng PORT giới hạn, trong khi đó trong hệ thống Màn hình nước luôn đòi hỏi một số lượng lớn van và phải có khả năng mở rộng thêm sau này. Để giải quyết vấn đề này, hệ thống mạch sử dụng nhiều IC 595 mắc nối tiếp. Với cách này, Màn hình nước có thể mở rộng số lượng van bao nhiêu tuỳ ý.

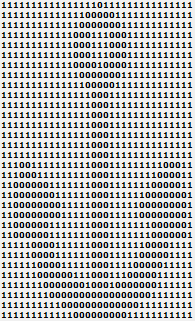
## **3.2. Thiết kế tổng quan hệ thống**

Như vậy qua phân tích các yêu cầu đề ra cho một hệ thống Màn hình nước. Ta có một cái nhìn tổng quan về hệ thống như sau:

Hình ảnh sau xử lý

Hình ảnh

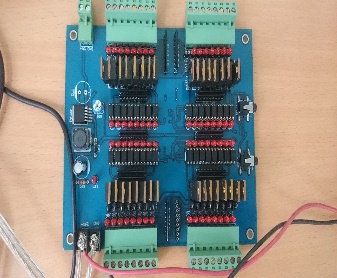
Máy tính xử lý





Gửi tín hiệu xuống mạch điều khiển

Mạch điều khiển van đóng mở

Hình 4: Sơ đồ chi tiết của hệ thống

Như vậy, máy tính sẽ load hình ảnh vào phân tích và sau đó gửi tín hiệu điều khiển cho mạch điều khiển. Mạch điều khiển dựa vào tín hiệu trên sẽ kích vào các mạch Công suất hoạt động, các mạch Công suất sẽ trực tiếp đóng mở các van.

# **CHƯƠNG 4: THIẾT KẾ MẠCH VÀ XÂY DỰNG CHƯƠNG TRÌNH ĐIỀU KHIỂN HỆ THỐNG BUBBLE GRAPHIC DISPLAY**

## **4.1. Thiết kế mạch điều khiển**

### *4.1.1. Sơ đồ khối hệ thống và sơ đồ khối Kit STM32F4 Discovery*

Như đã phân tích ở chương 3, hệ thống mạch đóng vai trò điều khiển trực tiếp Màn hình nước, hệ thống mạch sẽ nhận tín hiệu gửi từ máy tính và sau đó xử lý, điều khiển van Solenoid đóng mở. Trong hệ thống mạch ta tập trung xây dựng 3 khối chính như sau: Khối mạch Vi điều khiển, Khối mạch 595, Khối mạch Công suất.

**Sơ đồ khối hệ thống như sau:**

Máy tính

Kit STM32F4 Discovery

Khối mạch 595

Khối mạch Công suất 1

Khối mạch Công suất 2

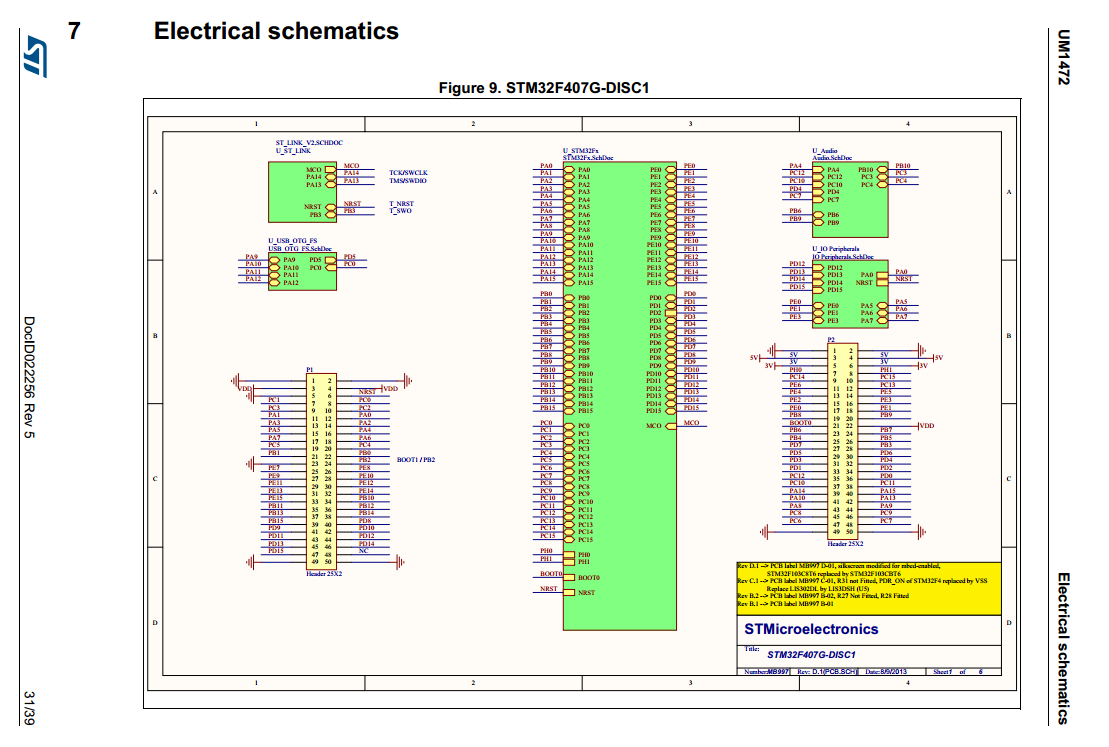
Khối mạch Công suất n

**………**…

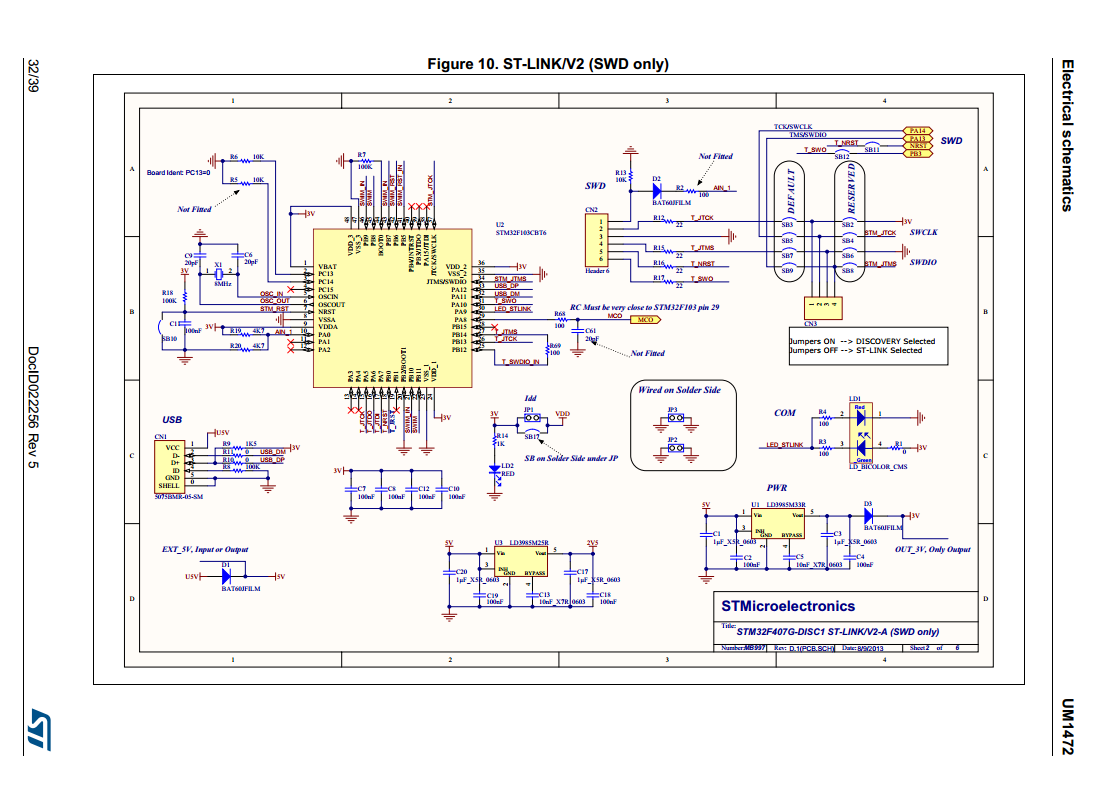
Van Solenoid

Hình 4.1: Sơ đồ khối Hệ thống mạch điều khiển

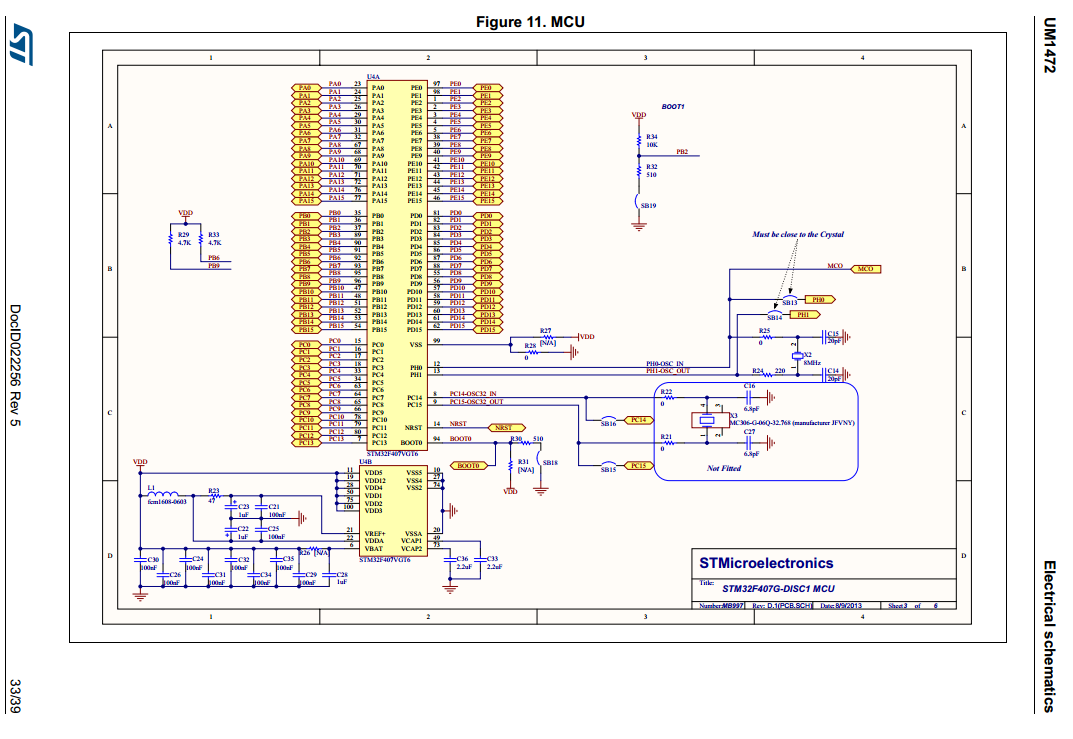
**Sơ đồ khối Kit STM32F4 Discovery:**



Hình 4.2: Sơ đồ PORT Kit STM32F4 DISCOVERY



Hình 4.3: Sơ đồ khối mạch nạp ST-LINK(SWD only)



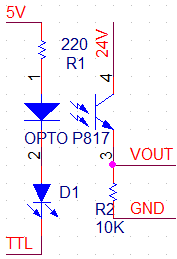
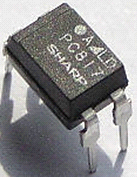
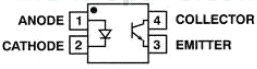
Hình 4.4: Sơ đồ Khối MCU Kit STM32F4 DISCOVERY

### *4.1.2. Khối mạch Công suất*

Khối mạch Công suất là khối thực hiện trực tiếp việc đóng mở Van Solenoid. Một Màn hình nước thông thường có số lượng van Solenoid rất lớn, hoạt động đóng mở ở công suất cao; do đó khối Công suất phải được tính toán các thông số điện áp một cách kĩ càng. Một mạch Công suất có thể điều khiển đóng mở cho 32 van Solenoid. Khối mạch Công suất nhận tín hiệu từ khối 595 điều khiển trực tiếp ra van Solenoid.

Trong mạch Công suất sử dụng MOSFET IRF\_540 kích Van Solenoid và Opto P817 có nhiệm vụ kích MOSFET và bảo vệ Vi điều khiển.

Opto PC817:



Hình 4.5: Cấu tạo chân, ký hiệu Opto và sơ đồ hoạt động [9]

Việc tính toán các thông số điện áp cho mạch trên được thực hiện như sau. Gọi:

* R1 là điện trở cho input của opto.
* Vled\_test là forward voltage (độ giảm điện áp thuận) qua đèn D1 dùng để kiểm tra dòng điện qua opto.
* Vled\_opto là forward voltage qua đèn trong opto.
* I là dòng điện qua mạch (qua R1, qua đèn trong opto và cũng là qua đèn dùng để kiểm tra Opto).
* R2 là điện trở tạo dòng cho điện áp đầu ra VOUT.

Đèn LED được sử dụng để kiểm tra với cường độ dòng điện định mức 20mA (vì đèn thường chỉ chỉ hoạt động tốt với dòng 5-20mA) nên có thể dùng Opto thông dụng PC817 với cường độ dòng qua LED của Opto định mức là 50mA (theo datasheet). Khi đó, Vled\_opto = 1.2 ─ 1.4V (được kiểm tra khi cho I = 20mA).

Để đèn của Opto này đủ sáng ở mức làm Opto hoạt động tốt, đồng thời cho đèn kiểm tra Opto sáng vừa đủ theo nhu cầu, vậy có thể chọn I = 10mA.

Với các giá trị trên, độ giảm điện thế cần có qua R là:

|  |
| --- |
| *UR = 5 – (Vled\_test  + Vled\_opto) (V).* |

Theo giá trị đo được đối với con PC817 sử dụng, **Vled\_opto**= **1.2V**. Và với đèn đỏ mờ được dùng trong khóa luận này có giá trị 2V.

Như vậy, độ giảm điện thế cần phải có qua điện trở cho Opto với đèn màu đỏ để kiểm tra là:

|  |
| --- |
| *UR = 5 – (2 + 1.2) = 1.8 (V).* |

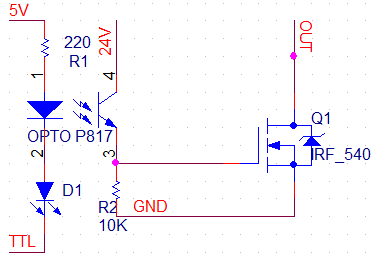
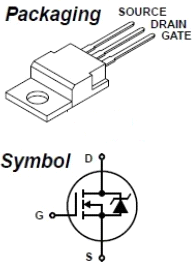
Theo đó, giá trị điện trở cho input của Opto với đèn để kiểm tra là:

|  |
| --- |
| *R = 1.8/I =1.8/0.01 = 180 (Ω).* |

Trong khóa luận này sử dụng điện trở 220Ω, tính ngược trở lại có được I ≈ 8mA.

Theo như sơ đồ, khi điện áp TTL có mức 0 thì Opto hoạt động tốt, kết quả thu được VOUT = 24V với dòng xuất ra theo là 2.4mA (đảm bảo kích MOSFET hoạt động tốt, không bị cháy với dòng cao).

MOSFET IRF540:



Hình 4.6: Ký hiệu, cấu tạo chân MOSFET và sơ đồ hoạt động [9]

Theo datasheet của Opto PC817, CTR (Current Transfer Ratio – Tỉ lệ vận chuyển dòng) từ 50 đến 600%. Giả sử giá trị thực tế cho PC817 dùng có CTR chỉ là 50% đi chăng nữa, IC qua opto khi Opto hoạt động ở chế độ khuếch đại:

|  |
| --- |
| *IC = (50/100).10 = 5 (mA).* |

Với giá trị này của IC, cũng là I qua điện trở phân áp cho MOSFET (cũng là điện trở chống hiện tượng cực gate của MOSFET bị float khi Opto ngắt mạch) (trong hình 4.5 là R2), điện áp qua điện trở này là:

|  |
| --- |
| *UR2 = IC.R2 = (0.005).10000 = 50 (V).* |

Trong khi:

|  |
| --- |
| *UR2 + UCE = 24 (V).* |

Opto sẽ hoạt động ở chế độ bão hòa, IC sẽ không thể được khuếch đại lên đến giá trị trên.

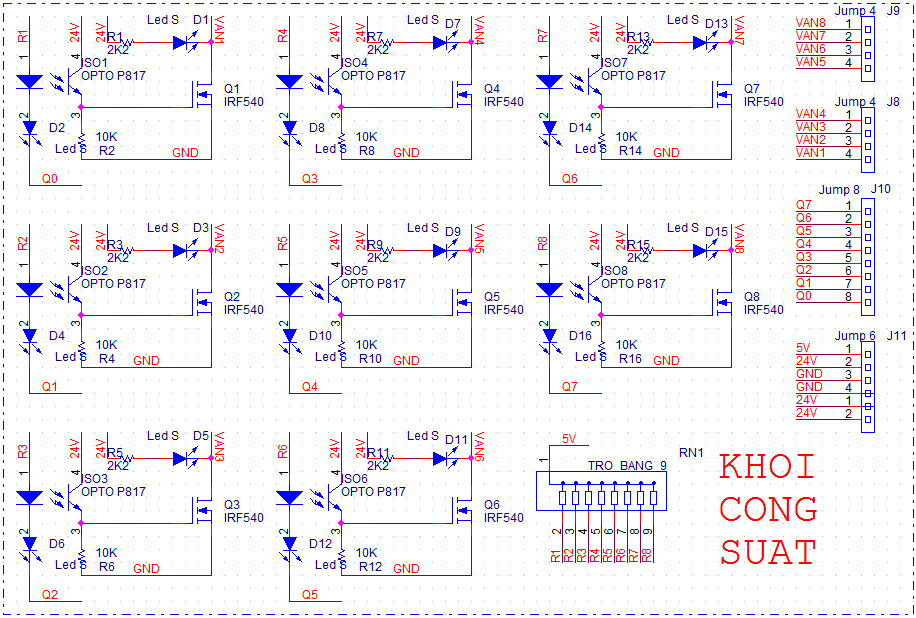
Theo datasheet của Opto, VCE(sat) (điện áp bão hòa giữa cực C và cực E của transistor trong Opto) có giá trị từ 0.1 đến 0.2V. Ta cứ giả sử với con PC817 của chúng ta, nó bằng 0.2V luôn thì hiệu điện thế qua R2 phân áp cho MOSFET (cũng là điện thế đặt vào cực G của MOSFET) là:

|  |
| --- |
| *UR2 = VG = 24 – 0.2 = 23.8 (V).* |

Với hoạt động của power MOSFET, nếu VGS ≥ VGS(th) (điện áp giữa cực G và cực S lớn hơn hoặc bằng điện áp ngưỡng (điện áp thềm) giữa cực G và cực S) thì MOSFET được “kích”, tức bắt đầu cho dòng điện từ cực D sang cực S. Thông thường VGS(th) có giá trị xấp xỉ 5V. Giá trị VGS cũng chính là VG.

Với phạm vị đề tài, bỏ qua các tính toán khác liên quan. Khi đó chỉ xem MOSFET chỉ thực hiện chức năng như 1 công tắc đóng mạch.

Sau khi tính toán, ta đưa ra sơ đồ nguyên lý cho một mạch công suất như sau:



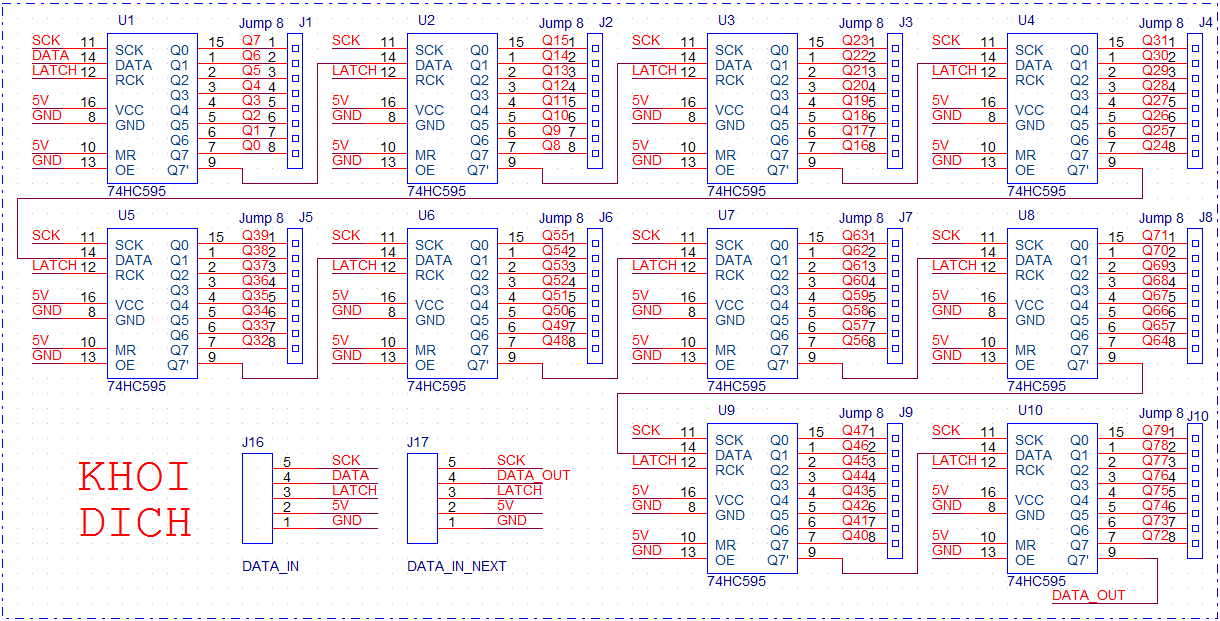
Hình 4.7: Sơ đồ nguyên lý Khối mạch Công suất

Với một mạch Công suất như vậy điều khiển đóng mở được 8 van Solenoid. Trong đó mỗi van Solenoid có điện áp là 24V.

### *4.1.3. Khối mạch 595*

Khối mạch 595 là khối trung gian nhận bit gửi từ khối Vi điều khiển sau đó lưu lại. Sau khi khối Vi điều khiển gửi đủ dữ liệu cần thiết, khối mạch 595 sẽ gửi tín hiệu cho khối Công suất thực hiện việc đóng mở van Solenoid.

Khối mạch 595 là bao gồm các IC 595 nối liền với nhau. Chân Q7’ IC595 này nối với chân DS IC595 kế tiếp. Các chân SCK và LATCH nối chung với nhau, các chân MR nối vào 5V và các chân OE nối GND. Dưới đây là sơ đồ nguyên lý cho mạch sử dụng 10 IC 595 điều khiển 80 van Solenoid:



Hình 4.8: Sơ đồ nguyên lý Khối mạch 595

## **4.2. Xây dựng chương trình xử lý ảnh và điều khiển**

### *4.2.1. Sơ đồ khối*

Chương trình điều khiển Bubble Graphic Display bao gồm 2 phần:

* Chương trình xử lý ảnh trên máy tính.
* Chương trình điều khiển Solenoid trên Kit STM32F4 DISCOVERY.

Sơ đồ điều khiển chương trình được thực hiện theo sơ đồ như sau:

Phần mềm xử lý ảnh trên máy tính

Mạch điều khiển

Van Solenoid

Image load từ máy tính

Hình 4.9: Sơ đồ điều khiển chương trình

### *4.2.2. Chương trình xử lý ảnh trên máy tính*

#### 4.2.2.1. Giới thiệu

Phần mềm xử lý ảnh và điều khiển trên máy tính có nhiệm vụ thực hiện 2 chức năng cơ bản là: xử lý ảnh, truyền dữ liệu sau khi xử lý xuống Vi điều khiển và một số chức năng phụ như cho phép người sử dụng chạy nhiều hình cùng một lúc hoặc điều chỉnh số lần lặp lại của một hình … Trước hết cần xây dựng 2 chức năng cơ bản của chương trình là xử lý ảnh và truyền dữ liệu.

#### 4.2.2.2. Phần xử lý ảnh

##### *4.2.2.2.1 Sơ đồ giải thuật*

Mục đích của việc xử lý ảnh là để chuyển đổi bất kì loại ảnh nào mà người sử dụng đưa vào sang dạng ảnh nhị phân (ảnh trắng đen) đồng thời thay đổi kích thước hình ảnh cho phù hợp với số lượng Solenoid điều khiển có trên hệ thống. Sau khi có được hình ảnh nhị phân theo kích thước phù hợp, chương trình sẽ chuyển hình ảnh này sang dạng ma trận chỉ gồm các bit 0 và 1; trong đó các pixel màu đen sẽ có giá trị là 0 và các pixel màu trắng có giá trị là 1. Sơ đồ giải thuật xử lý ảnh như sau:

Hình ảnh

Số Solenoid

Thay đổi tỉ lệ ảnh theo số Solenoid

Chuyển sang ảnh nhị phân

Chuyển sang mảng 2 chiều gồm các bit 0 và 1

Hình 4.10: Sơ đồ giải thuật chương trình xử lý ảnh

Chương trình xử lý ảnh và điều khiển cho hệ thống *Bubble Graphic Display* được xây dựng bằng ngôn ngữ lập trình C#, sử dụng thư viện *System.Drawing* trong .NET.

##### *4.2.2.2.2. Chuyển đổi kích thước ảnh theo tỷ lệ*

Trong hệ thống *Waterfall Graphic Display*, hệ thống sẽ dựa vào số pixel của tấm ảnh để điều khiển đóng mở Solenoid trong một khoảng thời gian nhất định. Đối với pixels có giá trị màu là đen thì cho Solenoid mở còn với pixels có giá trị màu là trắng thì ngược lại. Do đó kích thước của tấm ảnh bất kì thì chiều ngang của nó phải có số lượng pixel tương ứng với số lượng Solenoid có trên hệ thống. Vấn đề đặt ra là với mỗi tấm ảnh bất kì khi được đưa vào chương trình thường có kích thước to nhỏ khác nhau. Do đó ta phải xây dựng một hàm chuyển đổi kích thước ảnh theo tỉ lệ phù hợp với số lượng Solenoid trên hệ thống. Số Solenoid này có thể thay đổi phù hợp với từng kích thước hệ thống khác nhau trong thực tế.

Vậy kích thước của tấm hình phù hợp với yêu cầu hệ thống được tính như sau:

Chiều rộng = *số Solenoid.*

Chiều dài = *chiều dài ảnh ban đầu \* (số Solenoid / chiều rộng ảnh ban đầu).*

Ví dụ: một ảnh có kích thước là 239x137 và số Solenoid trên hệ thống là 80 cái.

Kích thước ảnh kết quả như sau: Chiều rộng = 80; Chiều dài = 137\*(80/239) = 46.

Để vẽ lại hình với kích thước mới ta sử dụng đối tượng Graphic có trong thư viện System.Drawing của C#:

*//Tạo một Bitmap với kích thước mới*

Bitmap bitmap = new Bitmap(dWidth,dHeight);

*//Khởi tạo đối tượng Graphic trong C#*

Graphics g = Graphics.FromImage((Image)bitmap);

*//Vẽ lại ảnh với kích thước mới*

g.DrawImage(sImage,0,0,dWidth,dHeight);

Với:

* sImage: là ảnh ban đầu
* dWidth: là chiều rộng ảnh kết quả
* dHeight: là chiều dài ảnh kết quả

Như vậy, kết quả thu được một ảnh giống như ảnh ban đầu nhưng có chiều rộng bằng với số Solenoid của hệ thống và kích thước tỉ lệ với kích thước ảnh ban đầu.

##### *4.2.2.2.3. Chuyển đổi ảnh màu sang ảnh nhị phân*

Hệ thống Waterfall Graphic Display hoạt động dựa vào màu sắc điểm ảnh của ảnh nhị phân. Ảnh nhị phân là ảnh chỉ có 2 màu đen và trắng. Đối với điểm ảnh là màu đen thì hệ thống kích Solenoid mở còn điểm ảnh màu trắng thì ngược lại. Do đó một ảnh màu khi đưa vào chương trình cần phải được chuyển sang ảnh nhị phân để xử lý các bước tiếp theo.

Để thực hiện việc chuyển ảnh màu sang ảnh nhị phân phải thực hiện theo thứ tự như sau: ảnh màu 🡪 ảnh xám 🡪 ảnh nhị phân. Để chuyển đổi ảnh màu sang ảnh xám sử dụng công thức như ở phần 2.3.3.1:

|  |
| --- |
| *Y = 0.3\*Red + 0.59\*Green + 0.11\*Blue* |

Hiện thực công thức trên vào phần lập trình trong chương trình như sau:

for (y = 0; y <Height; y++){

for (x = 0; x <Width; x++){

*//Lấy màu điểm ảnh ở vị trí (x,y)*

c = GetPixel(x, y)

*//Lấy mức xám cho điểm ảnh tại vị trí (x,y)*

gray = Convert.ToInt32(c.R \* 0.287 + c.G \* 0.599 + c.B \* 0.114);

*// Gán lại mức xám gray cho điểm ảnh*

}

}

Chuyển đổi ảnh xám sang ảnh nhị phân người ta thường gọi là thresholding, đặt ra một giá trị threshold thích hợp sau đó so sánh mức xám của điểm ảnh với giá trị threshold đó, nếu nhỏ hơn thì gán màu cho điểm ảnh là 0 còn lớn hơn thì giá trị là 255. Trong chương trình ta chọn giá trị ngưỡng threshold bằng **128**.

for (y = 0; y < Height; y++){

for (x = 0; x < Width; x++){

c = GetPixel(x, y);

if (gray < **128**){

*//Nếu nhỏ hơn 128 thì xét màu là trắng*

bitmap.SetPixel(x, y, Color.FromArgb(0, 0, 0));

}

else {

*//Ngược lại xét màu là đen*

bitmap.SetPixel(x, y, Color.FromArgb(255, 255,255));

}

}

}

##### *4.2.2.2.4. Chuyển đổi ảnh nhị phân sang mảng 2 chiều*

Vi điều khiển STM32F4 chỉ điều khiển đóng mở Solenoid dựa vào tín hiệu 1 và 0, trạng thái mở nếu là 0 và trạng thái đóng là 1. Do đó ảnh nhị phân phải được chuyển sang dạng mảng 2 chiều với chỉ các giá trị là 0 với 1 trước khi gửi các giá trị này xuống mạch điều khiển. Đối với điểm ảnh màu đen sẽ mang giá trị 0 và ngược lại. Một mảng 2 chiều tạm được sử dụng để lưu trữ rồi lấy mảng đó làm kết quả trả về. Sau dó duyệt từng điểm ảnh theo chiều dài và chiều rộng của ảnh.

for (y = 0; y < Height ; y++)

{

for (x = 0; x < Width ; x++)

{

c = bm.GetPixel(x, y);

if (c==255)

{

rs\_array[y,x] = 0;

}

else

{

rs\_array[y,x] = 1;

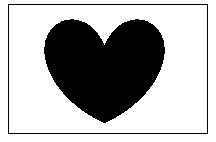
}

}

}

return rs\_array;

Với *Height* và *Width* là chiều cao và chiều rộng của ảnh nhị phân đã được thay đổi tỷ lệ. Mảng 2 chiều có số dòng bằng *Height* và số cột bằng *Width*. Ta duyệt theo chiều cao và chiều rộng của mảng. Vị trí nào có điểm ảnh bằng 255 thì tại vị trí đó ghi vào trong mảng là giá trị 0 và ngược lại với điểm ảnh bằng 0 thì ghi vào mảng giá trị 1.



Hình 4.11: Ảnh trái tim

Ví dụ, ta có ảnh trong Hình 4.10 với kích thước là 228x253 và Số Solenoid hệ thống có là 56 cái, thì giá trị *Width* và *Height* tương ứng là:

* Width = 56.
* Height= 253 \* (56/228) = 62.

Như vậy mảng kết quả ta cần sẽ khai báo như sau:

*int [,] rs\_array=new int [Height,Width];*

Sau khi duyệt từng điểm ảnh trên ảnh lấy thông tin ghi vào mảng ta được mảng *rs\_array* như sau:

**11111111111111111111111111111111111111111111111111111111**

**11111111111111111000011111111111110000111111111111111111**

**11111111111111100000000111111111000000001111111111111111**

**11111111111111000000000011111110000000000111111111111111**

**11111111111110000000000001111100000000000011111111111111**

**11111111111110000000000000111000000000000011111111111111**

**11111111111100000000000000010000000000000001111111111111**

**11111111111100000000000000010000000000000001111111111111**

**11111111111100000000000000000000000000000001111111111111**

**11111111111100000000000000000000000000000001111111111111**

**11111111111100000000000000000000000000000001111111111111**

**11111111111100000000000000000000000000000001111111111111**

**11111111111100000000000000000000000000000001111111111111**

**11111111111110000000000000000000000000000011111111111111**

**11111111111110000000000000000000000000000011111111111111**

**11111111111111000000000000000000000000000111111111111111**

**11111111111111000000000000000000000000000111111111111111**

**11111111111111100000000000000000000000001111111111111111**

**11111111111111110000000000000000000000011111111111111111**

**11111111111111110000000000000000000000011111111111111111**

**11111111111111111000000000000000000000111111111111111111**

**11111111111111111100000000000000000001111111111111111111**

**11111111111111111110000000000000000111111111111111111111**

**11111111111111111111000000000000001111111111111111111111**

**11111111111111111111110000000000011111111111111111111111**

**11111111111111111111111000000000111111111111111111111111**

**11111111111111111111111110000011111111111111111111111111**

**11111111111111111111111111101111111111111111111111111111**

**1111111111111111111111111****1111111111111111111111111111111**

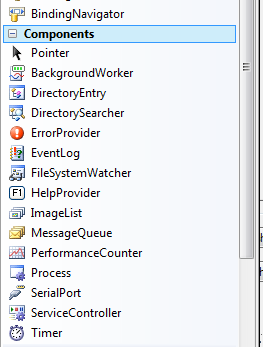
Hình 4.12: Mảng kết quả ảnh trái tim

Nhìn trên mảng kết quả, tương ứng với mỗi vị trí điểm ảnh màu đen sẽ thay bằng giá trị 0 và vị trí điểm ảnh màu trắng thay bằng giá trị 1.

#### 4.2.2.3. Phần truyền dữ liệu qua cổng COM

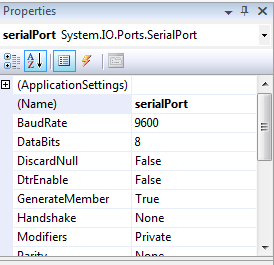
Trong hệ thống Waterfall Graphic Display, máy tính kết nối với mạch điều khiển thông qua cổng Serial Port hay còn gọi là cổng COM. Giao tiếp thông qua cổng COM là giao tiếp theo tiêu chuẩn nối tiếp RS232. Chuẩn này khá chậm nếu đem so sánh với chuẩn USB. Mặc dù vậy chuẩn RS232 vẫn được sử dụng nhiều trong giao tiếp giữa PC với Vi điều khiển do tính đơn giản của nó, một phần là do sự hỗ trợ của thư viện trong .NET nên việc lập trình trở nên đơn giản, gọn gàng hơn rất nhiều.

Trong .NET hỗ trợ sẵn đối tượng SerialPort trong thư viện *System.IO* để giao tiếp RS232 thông qua cổng COM. Để sử dụng đối tượng SerialPort từ cột ToolBox của Visual Studio 2008 phần Components kéo ra giao diện 1 đối tượng tên SerialPort:



Hình 4.13: Chọn Serial Port

Chỉnh *BaudRate* và *DataBits* cho phù hợp với cổng COM khai báo trên mạch:



Hình 4.13: Các thông số của Serial Port

Để bắt đầu gửi dữ liệu cần phải khai báo tên cổng COM mà PC đang sử dụng. Ví dụ như máy tính sử dụng cổng COM3 để giao tiếp:

serialPort.PortName = “COM3”;

serialPort.Open();

Gửi dữ liệu xuống mạch ta dùng phương thức *Write:*

serialPort .[Write (Byte[], Int32,](http://msdn.microsoft.com/en-US/library/ms143551(v=vs.80).aspx) **[Int32](http://msdn.microsoft.com/en-US/library/ms143551(v=vs.80).aspx)**[)](http://msdn.microsoft.com/en-US/library/ms143551(v=vs.80).aspx)

Trong đó:

* Byte[] là mảng lưu dữ liệu để gửi
* Int32 là vị trí phần tử đầu tiên trong mảng Byte
* **Int32** là số phần tử sẽ được gửi của mảng Byte

Đóng cổng COM sau khi kết thúc chương trình:

*//Kiểm tra xem cổng COM có đang mở hay không*

if (serialPort.IsOpen)

{

*//Đóng cổng COM lại*

serialPort.Close();

}

Trở lại với chương trình điều khiển trên máy tính của hệ thống Waterfall Graphic Display. Sau khi ảnh được xử lý và chuyển thành mảng 2 chiều bao gồm các bit 0 và bit 1 thì sẽ được gửi xuống mạch để xử lý đóng mở Solenoid.

Vấn đề đặt ra là bộ nhớ tạm của Vi điều khiển chỉ có giới hạn nhất định trong khi đó hình ảnh không có kích thước cố định nên không thể truyền một lúc cả mảng 2 chiều xuống vi điều khiển được. Để giải quyết vấn đề này, thay vì gửi cả mảng 2 chiều thì gửi lần lượt xuống vi điều khiển từng dòng dữ liệu. Dòng này sau khi được xử lý xong thì máy tính sẽ gửi dòng tiếp theo. Thời gian gửi và xử lý mỗi dòng rất nhỏ chỉ được tính bằng ms. Như vậy, cứ sau khi một dòng được gửi xuống chương trình sẽ tạo một khoảng thời gian delay chờ cho mạch xử lý xong rồi mới gửi tiếp. Ví dụ, ảnh sau khi được chuyển đổi sang mảng 2 chiều sẽ như hình 4.14:

I = 0

**1111111111111111111111111111111111111111**

**1111111111111111111101111111111111111111**

**1111111111111111111000011111111111111111**

**1111111111111111100000001111111111111111**

**1111111111111110000000000011111111111111**

**1111111111111100000000000000111111111111**

**1111111111110000000000000000011111111111**

**1111111111000000000000000000000111111111**

**1111111111000000000000000000000011111111**

**1111111111000000000000000000000111111111**

**1111111111100000000000000000000111111111**

**1111111111100000000000000000001111111111**

**1111111111100000000000000000001111111111**

**1111111111110000000000000000001111111111**

**1111111111110000000000000000011111111111**

**1111111111110000000000000000011111111111**

I = 23

**1111111111111000000000000000011111111111**

**1111111111111000000000000000111111111111**

I = 24

**1111111111111100000000000000111111111111**

**1111111111111111111111111111111111111111**

**1111111111111111111111111111111111111111**

I = 25

**1111111111111111111111111111111111111111**

Hình 4.14: Gửi mảng theo từng dòng

Mảng này có số cột = 40 và số dòng = 26. Chương trình sẽ gửi dòng ở vị trí I = 25 xuống trước, sau một khoảng thời gian delay sẽ gửi tiếp dòng ở vị trí I = 24, cứ như vậy đến khi đến vị trí I = 0.

Để tạo một khoảng delay sử dụng đối tượng *Timer* được hỗ trợ sẵn trong .NET. Cũng như SerialPort, một đối tượng Timer có thể được kéo thả từ Toolbox vào giao diện chương trình.

Khai báo thời gian delay cho đối tượng Timer (tính bằng ms):

timer1.Interval = 10;

Khởi động một đối tượng Timer:

timer1.Start();

Mã xử lý theo thời gian delay của đối tượng Timer trong hàm:

private void timer1\_Tick(object sender, EventArgs e){

*//…… Viết mã tại đây*

}

Giải thuật gửi dữ liệu theo sơ đồ như sau:

I = Số dòng-1

Gửi dòng thứ I

if (I < 0)

Sai

I = I - 1

Đúng

Hình 4.15: Sơ đồ giải thuật gửi dữ liệu từ cổng COM

Đặt mã truyền dữ liệu trong hàm xử lý của đối tượng Timer:

private void timer1\_Tick(object sender, EventArgs e){

*// Khai báo một byte Buffer*

byte[] Buffer = new byte[size\_buffer];

for (int i = 0; i < size\_buffer; i++){

*//..Ghi dữ liệu của dòng vào Buffer*

}

*//Gửi dữ liệu xuống board*

serialPort.Write(Buffer, 0, size\_buffer);

*//Dừng Timer nếu đã gửi hết số dòng*

if (this.num\_display <= 0) {

timer1.Stop();

}

}

Như vậy toàn bộ việc xử lý thời gian delay và truyền gửi hình ảnh dạng mảng 2 chiều được thực hiện hoàn toàn bằng chương trình của máy tính. Vi điều khiển chỉ cần nhận dữ liệu do máy tính gửi xuống và điều khiển cho Solenoid đóng mở. Vấn đề còn lại là xây dựng chương trình cho STM32F4 nhận dữ liệu từ cổng COM và điều khiển Solenoid hoạt động.

### *4.2.3. Chương trình điều khiển trên STM32F4*

#### 4.2.3.1. Giới thiệu

Chương trình điều khiển trên STM32F4 chỉ đơn giản là nhận dữ liệu từ máy tính gửi xuống qua cổng COM sau đó gửi dữ liệu này cho khối 595 và khối Công suất điều khiển van Solenoid.

#### 4.2.3.2. Xây dựng chương trình điều khiển cho STM32F4

Chương trình xử lý trên STM32F4 bao gồm 2 phần: Phần nhận dữ liệu từ máy tính và phần quét 595.

Để nhận dữ liệu từ máy tính cần khai báo một mảng lưu dữ liệu như sau:

*unsigned char rx\_buffer[NUMS]*

Trong đó NUMS là số IC 595 sử dụng trong mạch.

Khi có dữ liệu gửi từ máy tính gửi xuống, Vi điều khiển sẽ thực hiện một ngắt. Ta chỉ việc thực hiện ghi dữ liệu từ thanh ghi vào mảng *rx\_buffer* trong quá trình thực hiện ngắt.

*interrupt [USART\_RXC] void usart\_rx\_isr(void)*

*{*

*if(rx\_index<NUMS){*

*rx\_buffer[rx\_index]=UDR;*

*rx\_index++;*

*}else {*

*rx\_index=0;*

*}*

*}*

Để đưa dữ liệu ra khối 595 sử dụng kĩ thuật quét 595 như sau:

* Trước hết, xây dựng hàm tạo xung cho chân SH (SCK) và ST (LATCH) của IC 595:

*//Tao xung nhip chan SCK de ghi DATA vao 595*

*void sck(){*

*SCK=1;*

*SCK=0;*

*}*

*//Tao xung nhip chan LATCH de xuat du lieu tu 595*

*void latch(){*

*LATCH=1;*

*LATCH=0;*

*}*

* Thực hiện quét từng byte vào mạch 595:

*//Quet 595*

*void Quet(unsigned char x){*

*unsigned char temp;*

*for(n=0;n<8;n++)*

*{*

*temp=x;*

*temp=temp&0x80;*

*if(temp==0x80){*

*DATA=1;*

*}else {DATA=0;}*

*sck();*

*x=x<<1;*

*}*

*}*

*//DATA là chân DS của IC 595.*

* Kết hợp quét dữ liệu lưu trong *rx\_buffer* và điều khiển mạch 595 kích điều khiển van Solenoid như sau:

*void Display(){*

*if(rx\_index==NUMS){*

*for(k=NUMS;k>0;k--){*

*Quet(rx\_buffer[k-1]);*

*}*

*latch();*

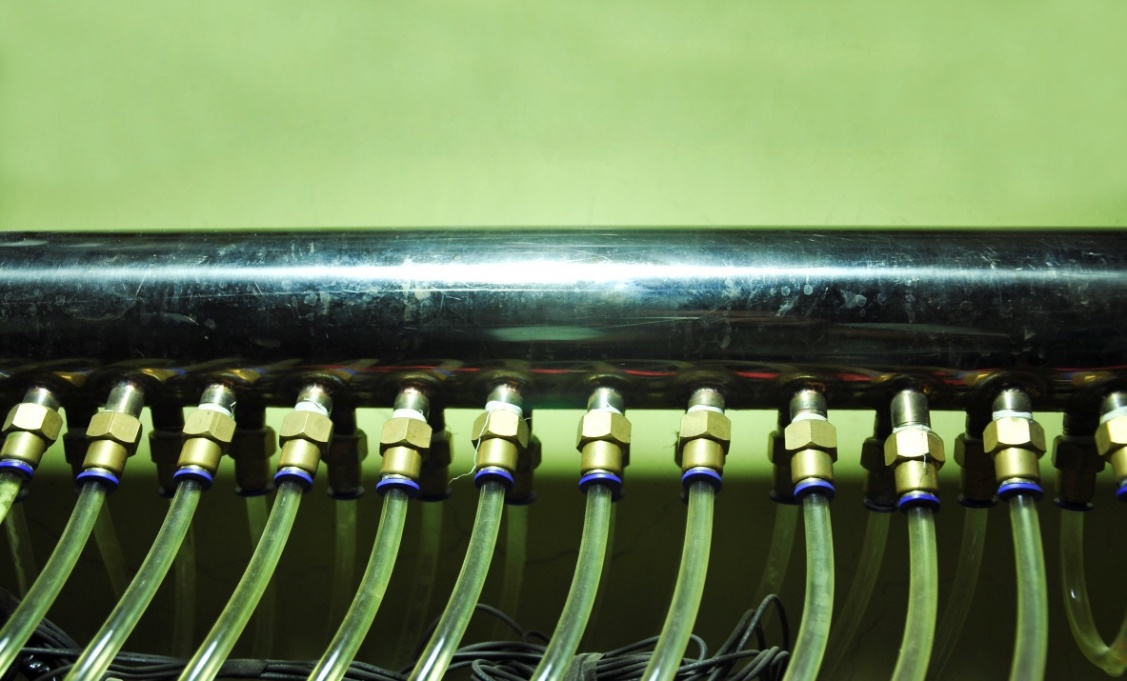
*}*

*}*

Như vậy, chương trình điều khiển cho STM32F4 cơ bản đã được xây dựng xong. Chương trình này thực hiện việc nhận dữ liệu từ máy tính qua cổng COM ghi vào một mảng rồi từ mảng thực hiện quét ra IC 595.

## **4.3. Thiết kế lắp đặt dàn Solenoid**

Bubble Graphic Display sử dụng van Solenoid của hãng STNC. Van này có điện áp là 24V-50Mhz, 8W. Các van này lắp đặt liên tiếp nhau thành một hàng ngang. Phía trên là ống chứ nước đặt sát nhau.

**

Hình 4.16: Ống chứa khí bơm đẩy lên các đầu van.



Hình 4.17: Khung.

Màn hình nước được treo cao hơn mặt đất khoảng 3m có bơm nước bơm liên tục vào ống Inox. Dây điện từ các van được nối trực tiếp vào các mạch Công suất của hệ thống điều khiển.

## **4.4. Kết quả thực nghiệm thu được**

### *4.4.1. Hệ thống mạch điều khiển*

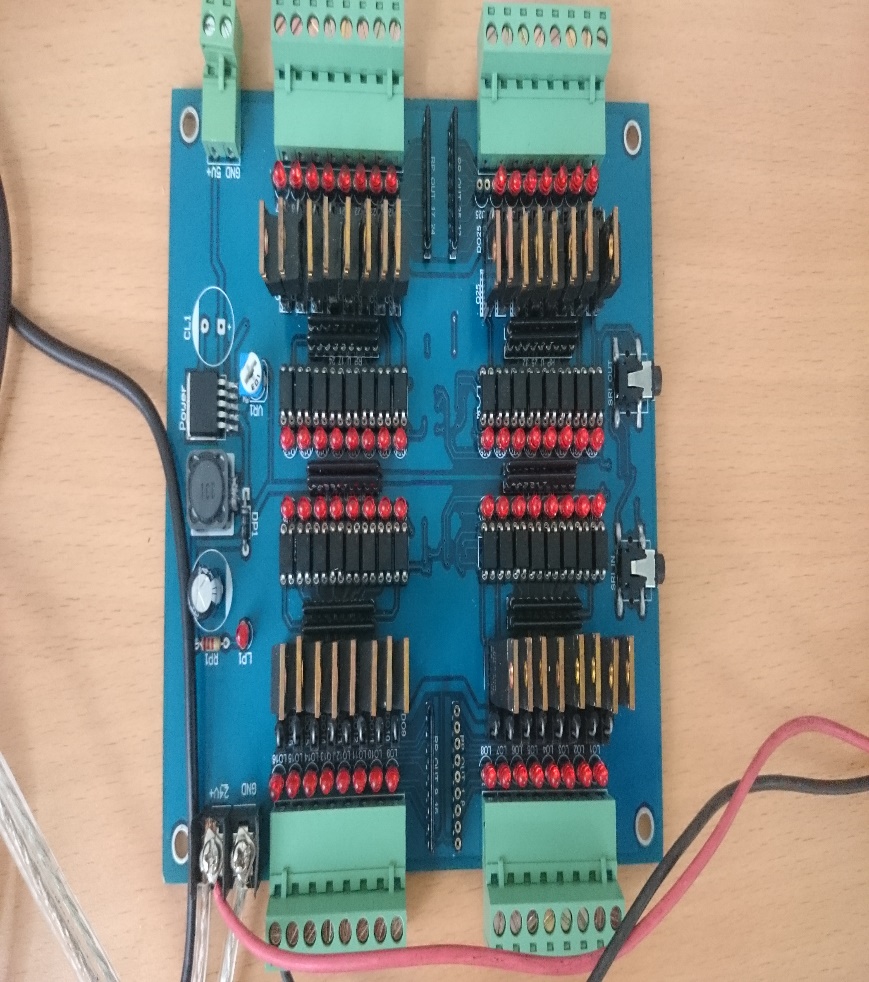
Sau khi thiết kế, chạy layout và làm mạch, hàn linh kiện, kết quả thu được như sau:

* Khối mạch Vi điều khiển sử dụng Kit STM32F4 DISCOVERY:



Hình 4.18: Mạch Vi điều khiển

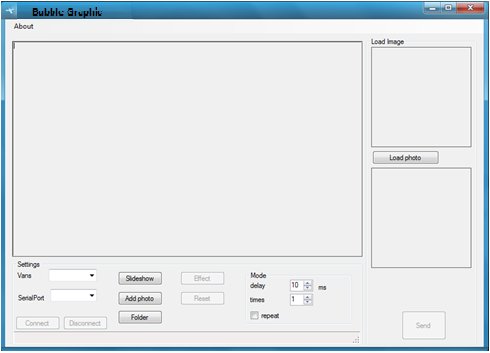
* Khối mạch 595 + Khối mạch công suất:



Hình 4.19: Mạch Công suất

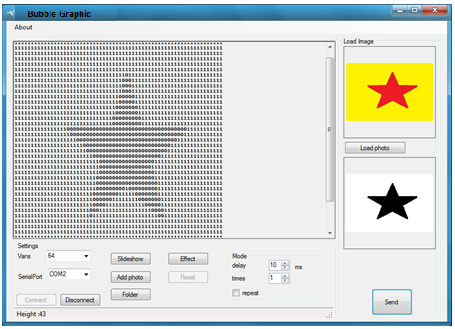
### *4.4.2. Chương trình xử lý ảnh*

Chương trình xử lý ảnh trên máy tính có giao diện chương trình như sau:



Hình 4.20: Giao diện chương trình phần mềm điều khiển

Load ảnh vào chương trình. Chương trình chuyển ảnh màu sang ảnh trắng đen và mảng 2 chiều.



Hình 4.21: Giao diện chương trình khi load hình, thiết lập thông số

Mảng này sẽ được gửi xuống mạch điều khiển qua cổng COM 2.

Chú thích các chức năng của chương trình:

|  |  |  |
| --- | --- | --- |
| **STT** | **Tên** | **Chức năng** |
| 1 | Vans | Chọn số Van Solenoid có trên hệ thống. |
| 2 | SerialPort | Chọn tên cổng COM của máy. |
| 3 | Connect | Kết nối cổng COM với mạch điều khiển. |
| 4 | Disconnect | Đóng kết nối. |
| 5 | Delay | Chọn thời gian delay. |
| 6 | Times | Chọn số lần rớt ảnh. |
| 7 | Repeat | Lặp đi lặp lại chương trình rơi ảnh. |
| 8 | Slideshow | Rơi danh sách ảnh do người dùng chọn từ trước |
| 9 | Effect | Rơi 1 số hiệu ứng/ ảnh đã tích hợp trong chương trình |
| 10 | Add Photo | Thêm ảnh vào danh sách ảnh hiển thị |
| 11 | Load Photo | Chọn 1 ảnh vào chương trình |
| 12 | Send | Gửi mảng sau xử lý xuống mạch điều khiển |
| 13 | Folder | Chọn thư mục chứa các ảnh cần hiển thị |

Bảng 4.3: Liệt kê các chức năng của chương trình phần mềm

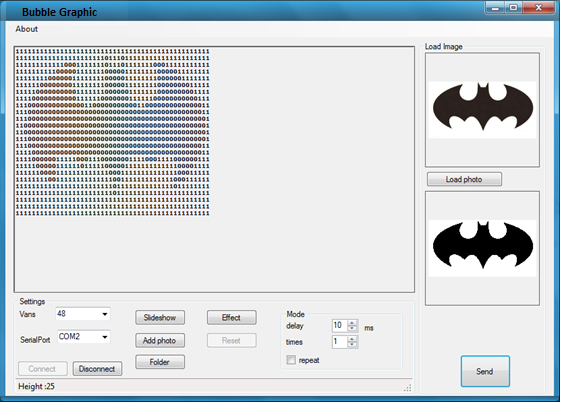
### *4.4.3. Kết quả thực nghiệm*

**Chuẩn bị phần mềm:**

Chạy chương trình xử lý ảnh trên PC nối cổng COM từ PC vào mạch điều khiển. Khai báo cổng COM và số van Solenoid trên hệ thống.

Khai báo giá trị delay.

Load hình ảnh vào chương trình:



Hình 4.22: Xử lý 1 ảnh trên phần mềm và gửi xuống mạch điều khiển

**Chuẩn bị phần cứng:**

Cung cấp nguồn 24V vào mạch điều khiển, kết nối các Khối mạch với nhau, nối dây từ các mạch công suất vào các van Solenoid. Sử dụng một bơm khí mini bơm khí liên tục vào các van. Kết quả thu được:



Hình 4.23: Máy bơm khí mini



Hình 4.24: Hình ảnh sản phẩm thực tế

## **4.5. Đánh giá**

**Ưu điểm:** Mạch chạy ổn định, chương trình xử lý khá chính xác, van Solenoid đóng ngắt nhanh, bọt khí nổi đúng với hình ảnh load trong chương trình.

**Khuyết điểm:** Do bơm khí cho Van chưa tốt và màn hình tương đối nhỏ nên hình ảnh khi phân giải bị hạn chế, chính vì thế chỉ có thể hiển thị một số hình đơn giản. Hệ thống còn khá đơn sơ, cồng kềnh trong khi vận chuyển.

**CHƯƠNG 5: KẾT LUẬN VÀ HƯỚNG PHÁT TRIỂN**

## **5.1. Kết luận**

Sau khi hoàn thành đề tài này, về cơ bản đã xây dựng được một mô hình Bubble Graphic Display nhỏ có thể tạo hình chữ, hình ảnh và một số hiệu ứng. Mặc dù hình ảnh trên màn hình nước vẫn chưa đẹp như mong muốn, song đó là bước khởi đầu để phát triển lên thành hệ thống Bubble Graphic Display lớn hơn sau này. Dưới đây là những kết quả mà khoá luận đạt được.

* Về mặt lý thuyết:
  + Nắm được cơ bản cấu tạo và nguyên lý hoạt động một số linh kiện như MOSFET, IC dịch 595…
  + Tính toán và thiết kế mạch Công suất có thể áp dụng điều khiển thiết bị có Công suất lớn.
  + Tìm hiểu thêm một số kĩ thuật cơ bản xử lý ảnh trong lập trình.
* Về mặt thực nghiệm:
  + Thiết kế hệ thống mạch điều khiển một dàn van Solenoid khoảng 20 van. Hệ thống có khả năng mở mộng thành một Bubble Graphic Display lớn hơn mà không cần phải thiết kế lại mạch.
  + Xây dựng chương trình điều khiển Bubble Graphic Display trên máy tính. Cho phép điều khiển hình ảnh với bất kì kích thước nào.
  + Cho chạy thử nghiệm Màn hình nước thực nghiệm và đạt được một số kết quả khả quan.

## **5.2. Hướng phát triển**

Bên cạnh các kết quả đạt được, đề tài vẫn còn gặp nhiều hạn chế cần phải khắc phục thêm. Đó cũng chính là hướng nghiên cứu mở rộng trong tương lai:

* + Nghiên cứu giao tiếp giữa Vi điều khiển với Internet để người dùng có thể điều khiển từ xa.
  + Nghiên cứu xử lý ảnh lấy từ thẻ nhớ ngay trên mạch điều khiển để giảm bớt sự cồng kềnh của hệ thống.
  + Mở rộng Bubble Graphic Display để hình ảnh có thể trở nên rõ nét hơn.

# **TÀI LIỆU THAM KHẢO**

Computer Vision anh Application (2012), Không gian màu (Color space), *Blog Computer Vision and Application*

STMicroelectronics (2012), STM32F4DISCOVERY schematics, *Website of STMicroelectronics*

STMicroelectronics (2016), RM0090 Reference manual, *Website of STMicroelectronics*

TH. S LÊ MINH PHƯƠNG (2012), Tổng quan về điện tử công suất, *Bài giảng môn Điện tử công suất*

Tilen Majerle (2016), Libraries and tutorials for STM32Fxxx series, *Website of* *STM32F4 Discovery*