

TRƯỜNG ĐẠI HỌC GIAO THÔNG VẬN TẢI TP. HỒ CHÍ MINH
KHOA ĐIỆN – ĐIỆN TỬ VIỄN THÔNG



ĐỒ ÁN

**ỨNG DỤNG BỘ ĐIỀU KHIỂN PID ĐIỀU KHIỂN NHIỆT ĐỘ CHO
BUỒNG ÁP TRÚNG, SỬ DỤNG VI ĐIỀU KHIỂN STM32F103C8T6,
MẠCH KÍCH TRIAC, GIÁM SÁT VÀ CÀI ĐẶT
BẰNG MÀN HÌNH LCD**

Học phần: **ĐỒ ÁN 1**

Chuyên ngành: **TỰ ĐỘNG HOÁ CÔNG NGHIỆP**

Giảng viên hướng dẫn: ThS. Lê Mạnh Thắng

Nhóm sinh viên thực hiện:

1. Trương Quang Bảo Khanh	2051050130	TD20B
2. Hoàng Minh Khôi	2051050134	TD20B
3. Nguyễn Duy Nhật	2051050154	TD20B
4. Bùi Tá Khang	2051050128	TD20B

TP. Hồ Chí Minh, 2023

LỜI CAM ĐOAN

Nhóm em xin cam đoan tất cả nội dung và thông tin trong bài báo cáo bộ môn Đồ Án 1 là trung thực và do nhóm tự thực hiện và lên ý tưởng trong suốt quá trình nghiên cứu tìm hiểu hoàn thiện đồ án.

Các thông tin trích dẫn trong báo cáo có nguồn gốc rõ ràng và được phép công bố. Nhóm em xin hoàn toàn chịu trách nhiệm về nội dung của bài báo cáo đồ án của mình.

Thành phố Hồ Chí Minh, ngày tháng năm 2023

Sinh viên thực hiện

Trương Quang Bảo Khanh

Hoàng Minh Khôi

Nguyễn Duy Nhật

Bùi Tá Khang

LỜI CẢM ƠN

Lời đầu tiên, nhóm em xin gửi lời cảm ơn chân thành đến thầy Lê Mạnh Thắng đã tận tình chỉ bảo, giúp đỡ nhóm hoàn thành môn học và báo cáo đồ án môn học Đồ Án 1.

Không thể quên được, nhóm nghiên cứu xin gửi lời biết ơn đến những đồng sinh thành dưỡng dục đã luôn hỗ trợ, động viên và cũng là niềm động lực lớn lao để nhóm có thể hoàn thành đề tài một cách tốt nhất.

Tuy đã có sự chuẩn bị, song bài báo cáo đồ án môn học không thể tránh khỏi những sai sót, nhóm chúng em rất mong được sự cảm thông từ thầy.

Nhóm chúng em xin chân thành cảm ơn!

MỤC LỤC

CHƯƠNG 1. TỔNG QUAN	1
1.1. Mở đầu – Lý do chọn đề tài:	1
1.2. Mục tiêu của công trình:	3
1.3. Phương pháp nghiên cứu:	3
CHƯƠNG 2. CƠ SỞ LÝ THUYẾT	5
2.1. Tổng quan về PID (Proportional Integral Derivative)	5
2.1.1. PID là gì?	5
2.1.2. Lịch sử ra đời của PID:	7
2.1.3. Các loại bộ điều khiển PID:	10
2.1.4. Vì sao lại cần PID?	10
2.1.5. Một số ứng dụng của bộ điều khiển PID:	11
2.1.6. Ứng dụng PID về điều khiển nhiệt độ:	13
2.1.7. Công thức PID:	14
2.2. Tổng quan về mạch phát hiện điểm 0:	15
2.2.1. Mạch phát hiện điểm 0 là gì?	15
2.2.2. Một số ứng dụng phổ biến của mạch phát hiện điểm 0:	15
2.2.3. Tại sao chúng ta cần sử dụng mạch phát hiện điểm 0?	16
2.2.4. Sơ đồ mạch phát hiện điểm 0:	18
2.3. Tìm hiểu về mạch kích Triac:	21
2.3.1. Mạch kích Triac là gì?	21
2.3.2. Sơ đồ mạch kích Triac:	23
2.3.3. Nguyên lý hoạt động của mạch kích Triac:	25
2.4. Tổng quan về Vi điều khiển STM32:	26
2.4.1. Giới thiệu về vi điều khiển STM32:	26
2.4.2. Các bước hoạt động của vi điều khiển STM32:	26
2.4.3. Chức năng và tính năng của vi điều khiển STM32:	27
2.4.4. Giới thiệu về STM32F103C8T6:	28
2.4.5. So sánh giữa STM32F103C8T6 và Arduino Uno R3:	32
2.5. Tổng quan về màn hình LCD 1602:	34
2.5.1. Màn hình LCD 1602 là gì?	34

2.5.2. Chức năng của từng chân LCD 1602:.....	35
2.6. Tổng quan về MAX31865 và PT100:.....	37
2.6.1. Max31865 và PT100 là gì?	37
2.6.2. Quá trình kết nối PT100 với MAX31865	39
2.7. Phần mềm STM32Cube IDE:	40
2.7.1. Tổng quan về STM32Cube IDE:	40
2.7.2. Cách tạo một Project mới trên STM32Cube IDE:	41
2.8. Phần mềm Easy EDA:.....	43
2.8.1. Tổng quan về phần mềm Easy EDA:.....	43
2.8.2. Cách tạo một project mới trên phần mềm Easy EDA:.....	44
CHƯƠNG 3. LẬP TRÌNH – THIẾT KẾT VÀ KẾT QUẢ ĐẠT ĐƯỢC	46
3.1. Yêu cầu cấu hệ thống – Sơ đồ khối – Chức năng của các khối:.....	46
3.1.1. Yêu cầu của hệ thống “buồng ấp trứng tự động”.....	46
3.1.2. Sơ đồ khối và chức năng mỗi khối.....	46
3.2. Quá trình xây dựng – hoạt động của hệ thống:.....	48
3.2.1. Giai đoạn 1: Xây dựng khối cảm biến nhiệt độ và khối hiển thị.....	48
3.2.2. Giai đoạn 2: Xây dựng mạch phát hiện điểm 0 và mạch kích Triac.....	48
3.2.3. Giai đoạn 3: bắt đầu viết thuật toán PID, viết thuật toán sử dụng cho hệ thống.....	48
3.3. Lưu đồ thuật toán hệ thống ấp trứng	49
3.4. Cấu hình và bắt đầu lập trình STM32F103C8T6 trên Cube IDE	50
3.4.1. Các thư viện sử dụng	50
3.4.2. Cấu hình Pinout cho STM32	51
3.4.3. Chuyển đổi thuật toán PID thành ngôn ngữ C/C++.....	56
3.5. Sơ đồ nguyên lý toàn bộ hệ thống và Layout PCB.....	58
3.6. Tiến hành đấu nối và kết quả đạt được	62
3.7. Kết luận và hướng phát triển:.....	64

MỤC LỤC HÌNH ẢNH

<i>Hình 1 Vi điều khiển STM32C8T6.....</i>	<i>1</i>
<i>Hình 2 Lò ấp trứng được dùng trong công nghiệp.....</i>	<i>2</i>
<i>Hình 3 Minh hoạ thuật toán PID</i>	<i>6</i>
<i>Hình 4 Lịch sử ra đời PID</i>	<i>7</i>
<i>Hình 5 Điều khiển mực nước bằng PID</i>	<i>12</i>
<i>Hình 6 Các công thức PID sử dụng trong đồ án</i>	<i>14</i>
<i>Hình 7 Sơ đồ mạch phát hiện điện áp điểm 0</i>	<i>18</i>
<i>Hình 8 Cầu Diode 2A 700V DB207 DIP-4 (2A vuông)</i>	<i>19</i>
<i>Hình 9 Diode Zener 1W 5.1V DIP 1N4733A</i>	<i>20</i>
<i>Hình 10 Opto PC817 DIP-4 (817)</i>	<i>21</i>
<i>Hình 11 Sơ đồ mạch kích Triac</i>	<i>23</i>
<i>Hình 12 MOC3023 DIP-6 Triac Driver Optocoupler</i>	<i>24</i>
<i>Hình 13 Triac BTA16 - 600B.....</i>	<i>24</i>
<i>Hình 14 Sơ đồ chân Triac BTA16 - 600B</i>	<i>25</i>
<i>Hình 15 Vi điều khiển STM32F103C8T6</i>	<i>26</i>
<i>Hình 16 Mô phỏng Vi điều khiển STM32F103C8T6</i>	<i>30</i>
<i>Hình 17 Sơ đồ chân của Vi điều khiển STM32F103C8T6.....</i>	<i>31</i>
<i>Hình 18 LCD 1602 xanh lá.....</i>	<i>35</i>
<i>Hình 19 LCD 1602 Xanh dương 5V.....</i>	<i>37</i>
<i>Hình 20 Mối tương quan giữa nhiệt độ và điện trở.....</i>	<i>38</i>
<i>Hình 21 MAX31865 Adafruit.....</i>	<i>39</i>
<i>Hình 22 PT100 và MAX31865</i>	<i>40</i>
<i>Hình 23 Giao diện làm việc trên phần mềm STM32Cube IDE</i>	<i>41</i>
<i>Hình 24 Tạo project mới trên STM32 Cube IDE.....</i>	<i>42</i>
<i>Hình 25 Tạo project mới trên STM32 Cube IDE.....</i>	<i>43</i>
<i>Hình 26 Giao diện làm việc trên phần mềm Easy EDA</i>	<i>44</i>
<i>Hình 27 Hướng dẫn tạo project mới trên Easy EDA.....</i>	<i>44</i>
<i>Hình 28 Hướng dẫn tạo project mới trên Easy EDA.....</i>	<i>45</i>
<i>Hình 29 Sơ đồ khối.....</i>	<i>46</i>
<i>Hình 30 Add thư viện LCD vào Project.....</i>	<i>50</i>

<i>Hình 31 Cấu hình tổng quát STM32F103C8T6</i>	<i>52</i>
<i>Hình 32 Cấu hình các ngắt ngoài và mức độ ưu tiên ngắt</i>	<i>53</i>
<i>Hình 33 Cấu hình timer2</i>	<i>54</i>
<i>Hình 34 Hàm delay micro giây với timer2</i>	<i>54</i>
<i>Hình 35 Cấu hình SPI.....</i>	<i>55</i>
<i>Hình 36 Tốc độ truyền tải SPI của MAX31865</i>	<i>56</i>
<i>Hình 37 Đặt các biến PID.....</i>	<i>57</i>
<i>Hình 38 Thuật toán PID code C/C++.....</i>	<i>57</i>
<i>Hình 39 Schematic toàn hệ thống</i>	<i>58</i>
<i>Hình 40 Sơ đồ PCB</i>	<i>59</i>
<i>Hình 41 Mặt trước board</i>	<i>60</i>
<i>Hình 42 Mặt sau board.....</i>	<i>61</i>
<i>Hình 43 Đầu nối testboard thực tế.....</i>	<i>62</i>
<i>Hình 44 Cài đặt nhiệt độ cho hệ thống.....</i>	<i>63</i>
<i>Hình 45 Buồng áp trứng tự động STM32F103C8T6</i>	<i>64</i>

Mục lục bảng

<i>Bảng 1 So sánh đặc tính STM32F103C8T6 và ARDUINO UNO R3.....</i>	<i>32</i>
<i>Bảng 2 So sánh thông số kỹ thuật giữa STM32F103C8T6 và ARDUINO UNO R3</i>	<i>33</i>
<i>Bảng 3 Lưu đồ thuật toán hệ thống được vẽ trên drawio.....</i>	<i>49</i>

TÓM TẮT CÔNG TRÌNH

Hiện nay, khoa học kỹ thuật ngày càng phát triển, con người đã có cũng như đang có nhiều bước tiến xa trong nhiều lĩnh vực, thực hiện được những việc mà trước đây tưởng chừng là không thể. Nhìn chung, tất cả các nỗ lực đó đều phục vụ cho nhu cầu đời sống con người ngày càng đầy đủ, tiện nghi hơn ; và đặc biệt, một trong số đó là lĩnh vực nghiên cứu về hệ điều khiển tự động, xu hướng phát triển toàn cầu trong những năm gần đây.

Ở Việt Nam nói riêng, tình hình nghiên cứu về hệ điều khiển tự động (PID) và vi điều khiển STM32 đã có những tiến bộ đáng kể. Các nghiên cứu tập trung vào cải tiến thuật toán PID và tích hợp nó vào nền tảng vi điều khiển STM32. Chúng ta đã có sự phát triển của các ứng dụng điều khiển như hệ thống nhiệt độ, robot và đèn giao thông ; Các dự án cũng đã khám phá khả năng kết hợp PID và STM32 với IoT và Machine Learning,... Tuy nhiên, tới thời điểm hiện tại, vẫn còn thách thức như tối ưu hóa PID cho ứng dụng phức tạp, đào tạo nguồn nhân lực và xây dựng phần cứng phù hợp...

Với mong muốn thực hiện một dự án vừa sức, mang tính làm quen với vi điều khiển STM32, bộ điều khiển PID, cảm biến nhiệt độ công nghiệp PT100, song song với tinh thần học hỏi, nghiên cứu và góp sức mình vào khoa học hệ thống nhúng. Nhóm nghiên cứu quyết định thực hiện đề tài **“Ứng dụng bộ điều khiển pid điều khiển nhiệt độ cho buồng áp trứng, sử dụng vi điều khiển stm32f103c8t6, mạch kích triac, giám sát và cài đặt trên màn hình tinh thể LCD”**.

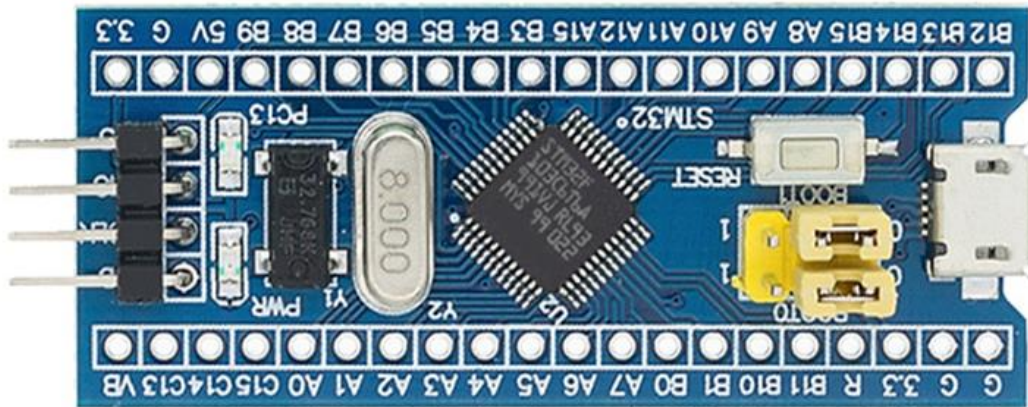
CHƯƠNG 1. TỔNG QUAN

Ở chương này, nhóm em sẽ trình bày lý do chọn đề tài, cũng như tính cấp thiết, tình hình nghiên cứu, mục tiêu, phương pháp nghiên cứu. Từ đó, người đọc sẽ dần hiểu được nguyên nhân, thực trạng cũng như phương án giải quyết của nhóm.

1.1. Mở đầu – Lý do chọn đề tài:

Ngày nay, việc ứng dụng khoa học công nghệ vào lĩnh vực phát triển nông nghiệp - đặc biệt là nhóm ngành tự động hoá – đã trở thành một hướng đi lớn mạnh, phát triển chính giúp thúc đẩy nền kinh tế.

Qua quá trình tìm hiểu, nhóm nhận thấy **STM32F103C8T6** là vi điều khiển giá thành rẻ, tốc độ xử lý nhanh (72MHz), có nhiều giao thức truyền thông như CAN, SPI, USART, v.v... được sử dụng nhiều trong các project thực tế.



Hình 1 Vi điều khiển STM32C8T6

Việc nghiên cứu STM32 giúp cho người nghiên cứu có cái nhìn toàn diện về vi điều khiển như thanh ghi, các hàm có đối số, RAM, v.v.. qua đó có kiến thức sâu về các hệ thống nhúng, rèn luyện tư duy debug, tuân tự v.v...

Cũng với góc nhìn và hướng đi đó, nhóm chúng em đã quyết định lựa chọn hướng nghiên cứu phát triển hệ thống lò nhiệt áp trứng, sử dụng vi điều khiển STM32 và bộ điều khiển PID làm cốt lõi để thực hiện công trình của mình.



Hình 2 Lò ấp trứng được dùng trong công nghiệp

Việc nghiên cứu 1 hệ thống ứng dụng thuật toán PID để áp trứng, lò nhiệt không phải là điều mới. Tuy nhiên thiết kế hệ thống đó bằng vi điều khiển giá rẻ **STM32F103C8T6** có tốc độ xử lý nhanh, nhiều chuẩn giao tiếp, ngoại vi là việc vẫn chưa được nghiên cứu sâu tại Trường Đại học Giao Thông Vận Tải Thành phố Hồ Chí Minh nói riêng và khu vực Thành phố Hồ Chí Minh nói chung. Chính vì những lý do trên, cộng hưởng với sự đồng thuận của giảng viên hướng dẫn Th.S Lê Mạnh Thắng, nhóm nghiên cứu đã chọn đề tài này.

1.2. Mục tiêu của công trình:

Mục tiêu đầu tiên nhóm đề ra là có thể hiểu được vi điều khiển **STM32F103C8T6** và biết cách chuyển đổi bộ điều khiển PID thành ngôn ngữ lập trình.

Xây dựng được hệ thống áp trứng tự động có thời gian xác lập dưới 2 phút, sai số xác lập dưới 0.09 độ C và độ vọt lố dưới 5%.

Cuối cùng cần phải nâng cao kiến thức, mở rộng trải nghiệm để tiếp xúc với các dự án lớn trong thực tế.

1.3. Phương pháp nghiên cứu:

Do đây là một đề tài khá mới mẻ với nhóm nên trong thời gian thực hiện công trình, chúng em đã gặp không ít khó khăn trong việc tiếp cận với các nguồn tài liệu.

Nhóm thực hiện công trình sử dụng **hai phương pháp chủ yếu:**

- **Phương pháp tham khảo tài liệu:** Nguồn tài liệu chủ yếu bằng tiếng Anh/tiếng Việt được tìm kiếm trên mạng Internet. Về thông tin nguồn tài liệu, nhóm em sẽ liệt kê chi tiết ở phần “**TÀI LIỆU LAM THẢO**”
- **Phương pháp lập trình, thiết kế mạch trên phần mềm tin học:** Song song với việc tham khảo tài liệu thì nhóm em đã sử dụng một số phần mềm tin học để có thể minh họa, phân tích và đào sâu vào cách thức xây dựng hệ thống áp dụng một cách chi tiết và rõ ràng nhất.

Một số phần mềm tin học nhóm nghiên cứu sử dụng có thể liệt kê như:

- + **Cube IDE** - dùng để lập trình, chạy code, debug code, nạp code.
- + **Easy EDA** – dùng để thiết kế mạch điện tử, mạch PCB hệ thống.

CHƯƠNG 2. CƠ SỞ LÝ THUYẾT

Ở chương này, nhóm nghiên cứu sẽ phân tích, khái quát về những vấn đề cơ sở như: Tìm hiểu về PID, Mạch phát hiện điểm 0, Mạch kích triac, vi xử lý STM32F103C8T6; tìm hiểu về màn hình LCD 1602, tìm hiểu MAX31865 và PT100, phần mềm Cube IDE, phần mềm Easy EDA, v.v... Từ đó, dẫn dắt người đọc đến những nội dung trực quan và chi tiết hơn của đề tài.

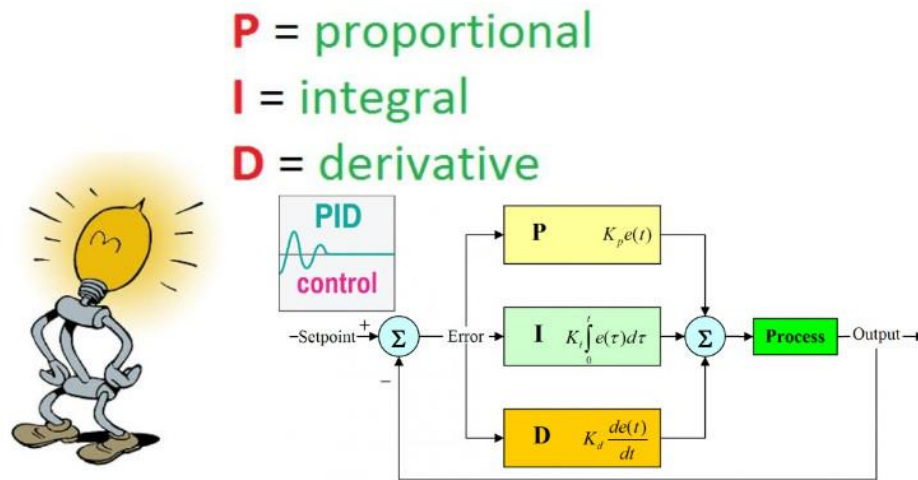
2.1. Tổng quan về PID (Proportional Integral Derivative)

2.1.1. PID là gì?

PID là tên viết tắt của Proportional Integral Derivative. Đây là một cơ chế phản hồi vòng điều khiển đã và đang được dùng phổ biến trong các hệ thống điều khiển công nghiệp. Bộ điều khiển này đã dùng dùng nhiều nhất trong các hệ thống vòng điều khiển kín (có tín hiệu phản hồi).

Bộ điều khiển này có thể tính toán được giá trị sai số sẽ là hiệu số giữa giá trị đặt mong muốn và giá trị của thông số biến đổi. PID có thể làm giảm tối đa tình trạng sai số bằng cách điều chỉnh các giá trị điều khiển ngay từ đầu vào.

Và để đạt được một kết quả tốt nhất, các thông số từ bộ điều khiển PID dùng trong tính toán cần phải điều chỉnh dựa vào tính chất của hệ thống. Đồng thời, nó cũng phụ thuộc vào những đặc thù của hệ thống.



Hình 3 Minh họa thuật toán PID

Trong đó:

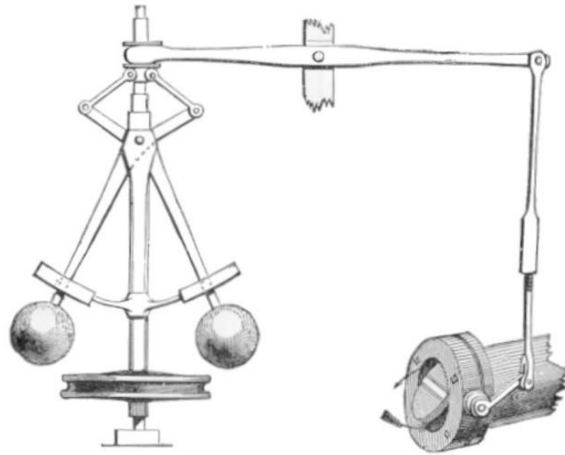
P – Proportional: Đây là phương pháp dùng để điều chỉnh tỉ lệ và tạo ra các tín hiệu điều chỉnh tỷ lệ với phần sai lệch đầu vào dựa vào thời gian lấy mẫu.

I – Integral: Đây là tích phân của sai số dựa vào thời gian lấy mẫu. Nó cũng là một phương pháp dùng để điều chỉnh nhằm tạo ra tín hiệu điều chỉnh sao cho độ sai lệch sẽ giảm tới 0. Từ đó, chúng ta cũng biết chính xác được tổng sai số tức thời theo thời gian hoặc các sai số tích lũy ở quá khứ. Khi mà thời gian càng ngắn thì việc điều chỉnh tích phân sẽ càng mạnh. Tương đương với đó chính là độ lệch sẽ càng thấp.

D – Dervative: Đây là vi phân sai lệch. Nó cùng để tạo ra các tín hiệu điều chỉnh sao cho tốc độ và tỷ lệ thay đổi sai lệch đầu vào. Thời gian càng dài thì

phạm vi để điều chỉnh vì phân sẽ càng lớn. Điều này cũng tương ứng với bộ điều chỉnh có thể đáp ứng tốt những thay đổi từ đầu vào càng nhanh chóng.

2.1.2. Lịch sử ra đời của PID:



Hình 4 Lịch sử ra đời PID

Điều khiển liên tục, trước khi bộ điều khiển PID được hiểu và thực hiện đầy đủ, có một trong những nguồn gốc của nó trong bộ điều khiển ly tâm, sử dụng trọng lượng quay để điều khiển một quá trình. Điều này được phát minh bởi Christiaan Huygens vào thế kỷ 17 để điều chỉnh khoảng cách giữa các cối xay trong cối xay gió tùy thuộc vào tốc độ quay, và do đó bù cho tốc độ thay đổi của thức ăn ngũ cốc.

Với việc phát minh ra động cơ hơi nước tĩnh áp suất thấp, cần phải điều khiển tốc độ tự động, và bộ điều khiển "con lắc hình nón" tự thiết kế của James Watt, một bộ bi thép quay được gắn vào trục chính thẳng đứng bằng các cánh

tay liên kết, đã trở thành một tiêu chuẩn công nghiệp. Điều này dựa trên khái niệm kiểm soát khe hở cối xay.

Tuy nhiên, điều khiển tốc độ xoay vẫn thay đổi trong các điều kiện tải trọng khác nhau, trong đó thiếu sót của cái mà ngày nay được gọi là điều khiển tỷ lệ là rõ ràng. Sai số giữa tốc độ mong muốn và tốc độ thực tế sẽ tăng lên khi tải tăng. Vào thế kỷ 19, cơ sở lý thuyết cho hoạt động của các thống đốc lần đầu tiên được James Clerk Maxwell mô tả vào năm 1868 trong bài báo nổi tiếng hiện nay của ông về các thống đốc. Ông đã khám phá cơ sở toán học cho sự ổn định kiểm soát, và tiến bộ một cách tốt hướng tới một giải pháp, nhưng đã kêu gọi các nhà toán học kiểm tra vấn đề. Vấn đề đã được kiểm tra thêm vào năm 1874 bởi Edward Routh, Charles Sturm, và vào năm 1895, Adolf Hurwitz, tất cả đều góp phần thiết lập các tiêu chí ổn định kiểm soát. Trong các ứng dụng tiếp theo, các bộ điều khiển tốc độ đã được tinh chỉnh thêm, đáng chú ý là bởi nhà khoa học người Mỹ Willard Gibbs, người vào năm 1872 đã phân tích về mặt lý thuyết thống đốc con lắc hình nón của Watt.

Vào khoảng thời gian này, việc phát minh ra ngư lôi Whitehead đặt ra một vấn đề điều khiển đòi hỏi phải kiểm soát chính xác độ sâu chạy. Chỉ riêng việc sử dụng một cảm biến áp suất sâu tỏ ra không đủ, và một con lắc đo khoảng cách phía trước và phía sau của ngư lôi được kết hợp với đo độ sâu để trở thành bộ điều khiển con lắc và thủy lực. Kiểm soát áp suất chỉ cung cấp một điều khiển tỷ lệ, nếu độ lợi điều khiển quá cao, sẽ trở nên không ổn định và đi vào tình trạng quá mức với sự mất ổn định đáng kể của việc giữ độ sâu. Con lắc đã

thêm cái mà ngày nay được gọi là điều khiển phản sinh, làm giảm dao động bằng cách phát hiện góc lặn / leo ngư lôi và do đó tốc độ thay đổi độ sâu. Sự phát triển này (được Whitehead đặt tên là "Bí mật" để không đưa ra manh mối nào về hành động của nó) là vào khoảng năm 1868.

Một ví dụ ban đầu khác về bộ điều khiển kiểu PID được Elmer Sperry phát triển vào năm 1911 để điều khiển tàu, mặc dù công việc của ông là trực quan hơn là dựa trên toán học.

Tuy nhiên, mãi đến năm 1922, một luật kiểm soát chính thức cho cái mà ngày nay chúng ta gọi là PID hoặc kiểm soát ba kỳ lần đầu tiên được phát triển bằng cách sử dụng phân tích lý thuyết, bởi kỹ sư người Mỹ gốc Nga Nicolas Minorsky. Minorsky đã nghiên cứu và thiết kế hệ thống lái tàu tự động cho Hải quân Hoa Kỳ và dựa trên phân tích của ông dựa trên những quan sát của một người lái tàu. Ông lưu ý rằng người lái tàu đã điều khiển con tàu không chỉ dựa trên lỗi hướng đi hiện tại mà còn dựa trên lỗi trong quá khứ, cũng như tốc độ thay đổi hiện tại; điều này sau đó đã được Minorsky đưa ra một phương pháp điều trị toán học. Mục tiêu của ông là sự ổn định, không phải kiểm soát chung, điều này đã đơn giản hóa vấn đề một cách đáng kể. Mặc dù kiểm soát tỷ lệ cung cấp sự ổn định chống lại các nhiễu nhỏ, nhưng nó không đủ để đối phó với sự xáo trộn ổn định, đáng chú ý là một cơn gió mạnh (do lỗi trạng thái ổn định), đòi hỏi phải thêm thuật ngữ tích phân. Cuối cùng, thuật ngữ phản sinh đã được thêm vào để cải thiện tính ổn định và kiểm soát.

Các thử nghiệm đã được thực hiện trên USS New Mexico, với các bộ điều khiển điều khiển vận tốc góc (không phải góc) của bánh lái. Kiểm soát PI mang lại yaw (sai số góc) duy trì $\pm 2^\circ$. Thêm phần tử D mang lại sai số ngáp $\pm 1/6^\circ$, tốt hơn hầu hết các người lái xe có thể đạt được.

Hải quân cuối cùng đã không áp dụng hệ thống này do sự kháng cự của nhân viên. Công việc tương tự đã được thực hiện và xuất bản bởi một số người khác trong những năm 1930.

2.1.3. Các loại bộ điều khiển PID:

Bộ điều khiển này có thể phân thành các loại:

- PI hay có tên gọi đầy đủ là *Proportional and Integral Controller* là bộ điều khiển tỉ lệ, tích phân.

- P là *Proportional Controller*, bộ điều khiển tỉ lệ.

- PID chính là viết tắt của *Proportional, Integral, and Derivative (PID) Controller*. Nó là bộ điều khiển tỉ lệ, tích phân, đạo hàm.

- PD là *Proportional and Derivative Controller* là bộ điều khiển đạo hàm.

2.1.4. Vì sao lại cần PID?

Bộ điều khiển PID (Proportional-Integral-Derivative) là một phần quan trọng trong hệ thống điều khiển tự động. Nó được sử dụng để điều chỉnh đầu ra của một hệ thống dựa trên sự đánh giá liên tục của sai số giữa giá trị đầu ra hiện tại và giá trị đặt trước (đầu vào).

Dưới đây là một số lý do chúng ta cần bộ điều khiển PID:

- **Độ ổn định:** Bộ điều khiển PID giúp duy trì sự ổn định của hệ thống. Nó phản hồi nhanh chóng đến các thay đổi trong đầu vào và điều chỉnh đầu ra để đảm bảo rằng hệ thống duy trì trạng thái cân bằng.

- **Độ chính xác:** Bộ điều khiển PID giúp giảm sai số giữa giá trị đầu ra thực tế và giá trị đặt trước. Bằng cách liên tục đánh giá sai số và điều chỉnh đầu ra, nó giúp hệ thống đạt được độ chính xác cao hơn.

- **Đáp ứng nhanh:** PID có khả năng điều chỉnh đầu ra nhanh chóng và linh hoạt đối với các thay đổi trong đầu vào. Điều này cho phép hệ thống phản ứng và thích ứng nhanh chóng với các điều kiện thay đổi, giúp đạt được hiệu suất tối ưu.

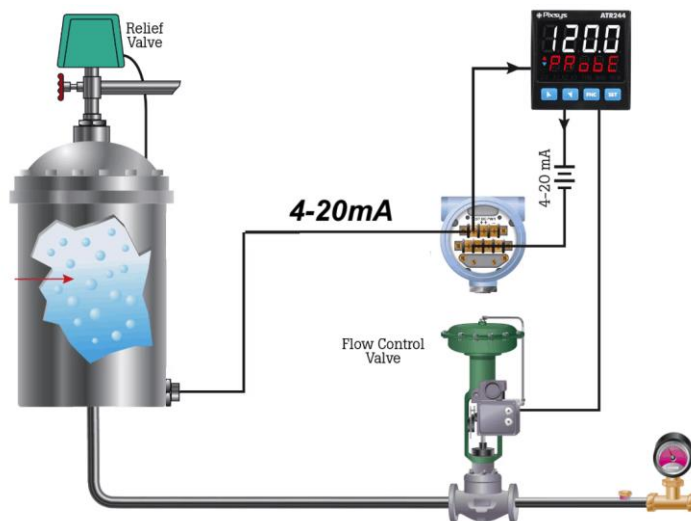
- **Dễ dàng thực hiện:** Bộ điều khiển PID có cấu trúc đơn giản và dễ dàng thực hiện trong các hệ thống điều khiển. Các thông số PID (proportional, integral, derivative) có thể được điều chỉnh để đáp ứng yêu cầu cụ thể của hệ thống.

- **Ứng dụng rộng rãi:** Bộ điều khiển PID được sử dụng phổ biến trong nhiều lĩnh vực như điều khiển công nghiệp, điều khiển quá trình, tự động hóa, robot học và nhiều ứng dụng khác. Sự linh hoạt và hiệu quả của PID làm cho nó trở thành một công cụ hữu ích trong việc điều khiển và điều chỉnh các hệ thống phức tạp.

2.1.5. Một số ứng dụng của bộ điều khiển PID:

Hiện nay PID được ứng dụng trong nhiều ngành nghề khác nhau.

- Giảm các sai số, hạn chế sự dao động hay là giảm thời gian xác lập và độ vọt lố...
- Sử dụng để điều khiển mực nước: bộ điều khiển được tự động hóa nhờ vào các thiết bị điện tử như cảm biến, van điều khiển...
- Điều khiển biến tần: Các thiết bị điện tử kết hợp ở đây gồm có: van điều khiển lưu lượng, cảm biến nhiệt độ, biến tần điều khiển....
- Kiểm soát lưu lượng khí qua đường ống
- Điều khiển PID trong PLC: Ở trong PLC thường sẽ được thiết kế sẵn các hàm dùng để điều chỉnh nhiệt độ, áp suất, lưu lượng...



Hình 5 Điều khiển mực nước bằng PID

2.1.6. Ứng dụng PID về điều khiển nhiệt độ:

Một ví dụ cụ thể về việc sử dụng bộ điều khiển PID để điều khiển nhiệt độ là *hệ thống điều khiển nhiệt độ của một lò nung*.

Trong một lò nung, bộ điều khiển PID có thể được sử dụng để duy trì nhiệt độ trong lò ở một mức ổn định. Các thành phần của bộ điều khiển PID được cấu hình như sau:

- **Proportional (P):** Đầu vào của thành phần P sẽ đo lường hiện tại của nhiệt độ trong lò nung và so sánh với giá trị đặt trước (setpoint). Sai số giữa hai giá trị này sẽ được nhân với hệ số tỷ lệ (gain) để tính toán giá trị điều chỉnh đầu ra của P.

- **Integral (I):** Thành phần I tích lũy các sai số trong thời gian, và điều chỉnh đầu ra dựa trên tổng sai số tích lũy. Điều này giúp giảm thiểu sai số ổn định dài hạn và đảm bảo rằng nhiệt độ trong lò nung đạt được giá trị đặt trước.

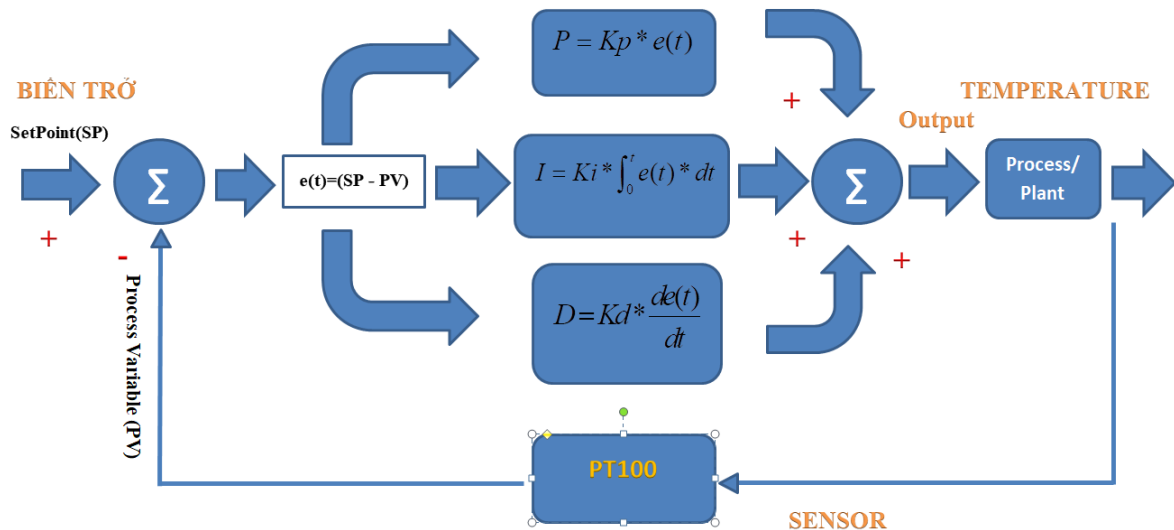
- **Derivative (D):** Thành phần D đo tốc độ thay đổi của nhiệt độ trong lò nung. Nếu nhiệt độ thay đổi quá nhanh, thành phần D sẽ tăng đầu ra để giảm thiểu độ dao động và giữ cho nhiệt độ ổn định hơn.

Bằng cách kết hợp ba thành phần trên, bộ điều khiển PID sẽ điều chỉnh đầu ra (ví dụ như công suất điện đưa vào lò) để duy trì nhiệt độ trong lò nung ở mức ổn định và gần giá trị đặt trước.

Trong quá trình hoạt động, bộ điều khiển PID liên tục đo nhiệt độ hiện tại, so sánh với giá trị đặt trước và tính toán giá trị điều chỉnh dựa trên các thông số

được cấu hình. Sau đó, nó gửi tín hiệu điều chỉnh cho hệ thống điện tử của lò nung để điều chỉnh công suất điện đưa vào và duy trì nhiệt độ ổn định.

2.1.7. Công thức PID:



Hình 6 Các công thức PID sử dụng trong đồ án

Bộ điều khiển PID là một bộ điều khiển vòng kín được sử dụng rộng rãi trong hệ thống điện, hệ thống tự động, điện tử. Mục tiêu của bộ điều khiển PID là điều chỉnh giá trị điều khiển ở ngõ ra **Output** sao cho sai lệch **Error $e(t) = (SP - PV)$** giữa giá trị đo được của hệ thống **PV (Process Variable)** với giá trị cài đặt **SP (SetPoint)** nhỏ nhất có thể (~ 0), đạt được sự ổn định và có đáp ứng nhanh.

Mục tiêu của bộ điều khiển PID là điều chỉnh giá trị điều khiển ở ngõ ra **Output** sao cho sai lệch **Error $e(t) = (SP - PV)$** giữa giá trị đo được của hệ

thông PV (Process Variable) với giá trị cài đặt **SP (SetPoint)** nhỏ nhất có thể (~ 0), đạt được sự ổn định và có đáp ứng nhanh.

2.2. Tổng quan về mạch phát hiện điểm 0:

2.2.1. Mạch phát hiện điểm 0 là gì?

Mạch phát hiện điểm 0 (zero crossing circuit) là một loại mạch điện được sử dụng để phát hiện sự thay đổi từ dương sang âm hoặc từ âm sang dương của một tín hiệu điện. Điểm 0 đề cập đến thời điểm khi tín hiệu điện chuyển từ một chiều sang chiều khác hoặc từ mức điện thế cao xuống mức điện thế thấp (hoặc ngược lại) trong chu kỳ tín hiệu.

Mạch phát hiện điểm 0 thường được sử dụng trong các ứng dụng như điều khiển động cơ, âm thanh, xử lý tín hiệu và các hệ thống đo lường. Mạch này có thể sử dụng để đếm số lượng điểm 0 trong một chu kỳ tín hiệu hoặc để đo tần số của tín hiệu dựa trên khoảng thời gian giữa các điểm 0 liên tiếp.

Có nhiều cách để thiết kế mạch phát hiện điểm 0, bao gồm sử dụng linh kiện như transistor, op-amp, hoặc vi điều khiển. Các mạch này thường sử dụng các nguyên lý như so sánh mức điện thế và sử dụng bộ lọc để loại bỏ nhiễu và chỉ tập trung vào sự thay đổi từ dương sang âm hoặc từ âm sang dương.

2.2.2. Một số ứng dụng phổ biến của mạch phát hiện điểm 0:

- **Đo tần số:** Mạch phát hiện điểm 0 có thể được sử dụng để đo tần số của một tín hiệu. Bằng cách đếm số lượng điểm 0 trong một khoảng thời gian, ta có thể tính được tần số của tín hiệu đó.

- **Điều khiển động cơ:** Mạch phát hiện điểm 0 có thể được sử dụng trong hệ thống điều khiển động cơ để xác định vị trí hoặc tốc độ của động cơ. Bằng cách theo dõi sự thay đổi từ dương sang âm hoặc từ âm sang dương, ta có thể xác định được khi nào động cơ đạt đến một điểm quan trọng trong quá trình quay.

- **Xử lý tín hiệu:** Mạch phát hiện điểm 0 có thể được sử dụng để xử lý tín hiệu âm thanh hoặc tín hiệu khác. Ví dụ, trong xử lý âm thanh, mạch này có thể được sử dụng để phân tách tín hiệu thành các phần dương và âm, hoặc để xác định các điểm tín hiệu đặc biệt như đỉnh sóng (peaks) hoặc vị trí trung tâm của các nhịp điệu.

- **Điều khiển hệ thống:** Mạch phát hiện điểm 0 có thể được sử dụng để xác định sự thay đổi trong một tín hiệu và điều khiển các hệ thống dựa trên sự thay đổi đó. Ví dụ, trong hệ thống đèn giao thông, mạch này có thể được sử dụng để phát hiện sự chuyển đổi từ đèn đỏ sang đèn xanh để kích hoạt các hành động tiếp theo.

Trên thực tế, ứng dụng của mạch phát hiện điểm 0 là rất đa dạng và phụ thuộc vào ngữ cảnh sử dụng của nó.

2.2.3. Tại sao chúng ta cần sử dụng mạch phát hiện điểm 0?

- Xác định trạng thái hoặc mức điện áp: Mạch phát hiện điểm 0 có thể xác định xem mức điện áp hoặc dòng điện có vượt quá ngưỡng cố định hay không. Điều này rất hữu ích trong việc xác định trạng thái của một hệ thống hoặc các

thiết bị, như xác định trạng thái hoạt động hay ngừng hoạt động của một máy móc.

- Bảo vệ quá tải: Mạch phát hiện điểm 0 có thể được sử dụng để bảo vệ các thành phần điện tử khỏi quá tải. Khi một mức điện áp hoặc dòng điện vượt quá ngưỡng quy định, mạch phát hiện điểm 0 có thể kích hoạt các cơ chế bảo vệ để ngắt hoặc giảm dòng điện, từ đó ngăn chặn các thiệt hại tiềm năng cho hệ thống.

- Điều khiển chuyển đổi: Mạch phát hiện điểm 0 cũng có thể được sử dụng để điều khiển các chuyển đổi hoặc thiết bị khác. Ví dụ, khi một nguồn điện áp đạt đến mức quy định, mạch phát hiện điểm 0 có thể kích hoạt chuyển đổi hoặc kích thích một hành động nhất định, chẳng hạn như bật đèn hoặc bật máy móc.

- Kiểm soát năng lượng: Mạch phát hiện điểm 0 có thể được sử dụng để kiểm soát năng lượng tiêu thụ của các thiết bị. Bằng cách xác định mức điện áp hoặc dòng điện, mạch phát hiện điểm 0 có thể điều chỉnh hoạt động của các linh kiện hoặc hệ thống để tiết kiệm.

* Nếu không sử dụng mạch phát hiện điểm 0 và kích bừa bãi mà không có hệ thống kiểm soát, có thể xảy ra những hậu quả không mong muốn như sau:

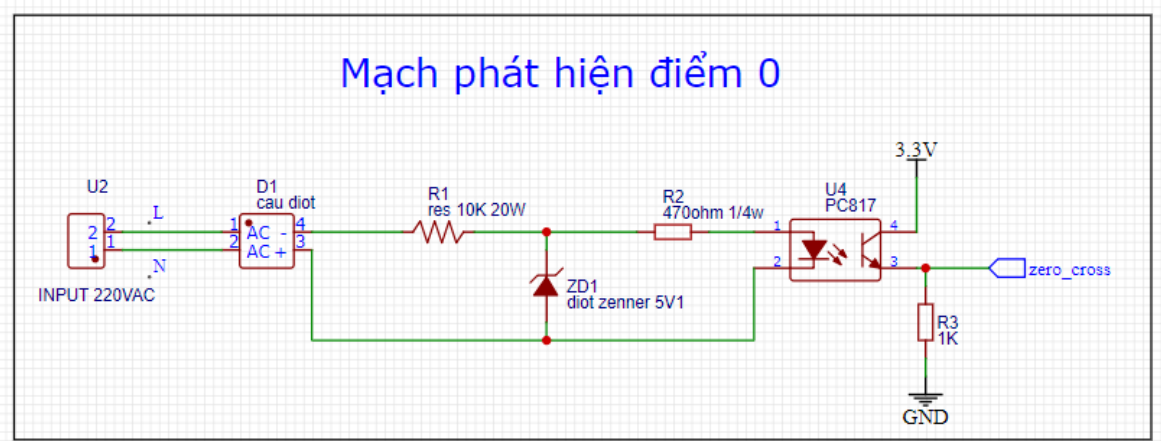
1. Quá tải và hỏng thiết bị: Khi không có mạch phát hiện điểm 0, không có cơ chế để ngăn chặn mức điện áp hoặc dòng điện vượt quá giới hạn an toàn. Điều này có thể dẫn đến quá tải và hỏng hóc các thành phần hoặc thiết bị điện tử. Ví dụ, một máy móc có thể bị hỏng hoặc bị cháy khi quá tải điện.

2. *Thiệt hại về tài sản:* Nếu không có mạch phát hiện điểm 0 để ngắt hoặc giảm dòng điện khi cần thiết, năng lượng điện sẽ tiếp tục truyền qua các thiết bị và có thể gây ra thiệt hại cho tài sản. Điều này có thể bao gồm hỏng hóc linh kiện, đèn bị cháy, máy móc hỏng, hoặc thậm chí gây cháy nổ.

3. *An toàn và nguy hiểm:* Việc kích bừa bãi điện mà không có kiểm soát có thể tạo ra một môi trường nguy hiểm. Nếu dòng điện không được điều khiển và ngừng khi cần thiết, có thể xảy ra tai nạn, gây chấn thương hoặc gây nguy hiểm đến tính mạng của con người.

4. *Tiêu thụ năng lượng không hiệu quả:* Mạch phát hiện điểm 0 có thể giúp kiểm soát và điều chỉnh năng lượng tiêu thụ của hệ thống. Nếu không có mạch này, hệ thống có thể tiêu thụ năng lượng không cần thiết hoặc không hiệu quả, dẫn đến lãng phí năng lượng và tăng chi phí hoạt động.

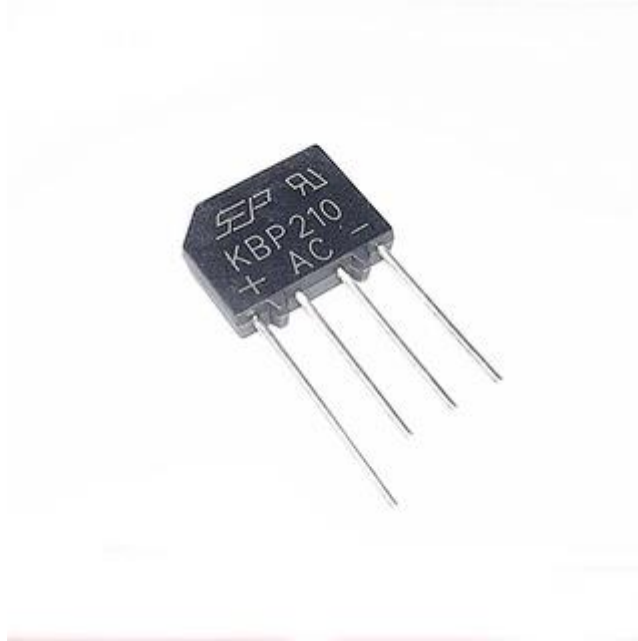
2.2.4. Sơ đồ mạch phát hiện điểm 0:



Hình 7 Sơ đồ mạch phát hiện điện áp điểm 0

Trong đó:

Cầu diot(D1): Dùng để chỉnh lưu dòng điện xoay chiều thành một chiều.



Hình 8 Cầu Diode 2A 700V DB207 DIP-4 (2A vuông)

Thông số kỹ thuật:

- Dòng điện định mức: 2A
- Điện áp tối đa: 700V
- Nhiệt độ hoạt động: -55°C đến 165°C

Diode Zener 5V(ZD1): Diode Zener thể hiện tính năng ổn áp khi được phân cực nghịch. Ban đầu, khi phân cực nghịch, dòng qua diode rất bé. Nhưng nếu tăng điện áp ngược lên 1 giá trị nào đó (V_Z) thì dòng tăng mạnh trong khi điện áp giữa 2 đầu diode gần như không đổi.



Hình 9 Diode Zener 1W 5.1V DIP 1N4733A

PC817(U4): là một loại optocoupler hoặc optoisolator, nó được sử dụng trong mạch phát hiện điểm 0 để cách ly và truyền tín hiệu giữa hai phần của mạch. Nhiệm vụ chính của PC817 trong mạch phát hiện điểm 0 là truyền tín hiệu từ phần nhận tín hiệu (đầu vào) sang phần xử lý (đầu ra) mà không có sự tiếp xúc vật lý trực tiếp giữa hai phần này.



Hình 10 Opto PC817 DIP-4 (817)

Các giá trị cần tính chọn trong mạch:

- Giá trị hiệu dụng khi qua cầu Diot: $U_{hd} = 220 - (0,7 \times 2) = 218,6 \text{ (V)}$
- Giá trị điện trở: $I_R = \frac{U_{hd}-5V}{10k} = \frac{218,6V-5V}{10^4} \approx 0,02 \text{ (A)}$
- Điện áp rơi trên điện trở 10K: $U_R = I_R \times 10K = 0,02 \times 10^4 = 200 \text{ (V)}$
- Công suất của trở: $P_R = I_R^2 \times 10k = 0,02^2 \times 10^4 = 4.10^{-4} \times 10^4 = 4 \text{ (W)}$

2.3. Tìm hiểu về mạch kích Triac:

2.3.1. Mạch kích Triac là gì?

Mạch kích Triac là một mạch điện được sử dụng để điều khiển độ sáng hoặc công suất của các thiết bị điện, chẳng hạn như đèn, quạt hay máy giặt.

Triac là một loại transistor hai hướng, cho phép dòng điện chạy qua nó theo cả hai chiều.

Triac được chế tạo từ hai transistor SCR (Silicon-Controlled Rectifier) kết hợp lại và chia sẻ một cổng điều khiển. Điều này cho phép Triac có khả năng điều khiển dòng điện chạy qua nó, bằng cách điều chỉnh thời gian mà nó dẫn điện trong mỗi nửa chu kỳ. Điều này làm cho Triac trở thành một công cụ hữu ích trong việc điều khiển tải điện biến thiên như đèn sáng.

Mạch kích Triac thường bao gồm các thành phần chính sau:

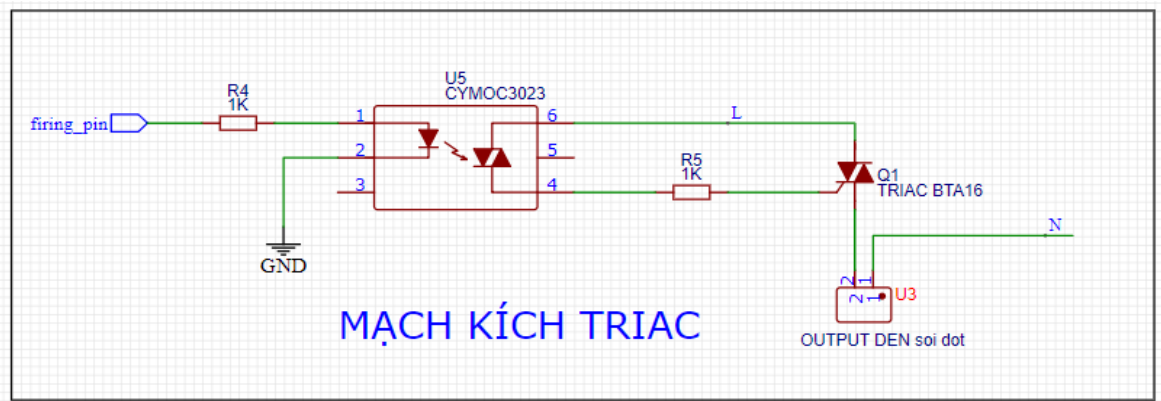
- **Triac:** Là linh kiện chính trong mạch kích. Nó có ba chân: cổng điều khiển (Gate), chân MT1 (Main Terminal 1) và chân MT2 (Main Terminal 2).

- **Ngõ vào điều khiển:** Được sử dụng để cung cấp tín hiệu điều khiển cho mạch kích Triac. Tín hiệu này thường được cung cấp từ mạch điều khiển hoặc mạch điện tử khác.

- **Optocoupler (quang cách ly):** Đôi khi được sử dụng trong mạch kích Triac để cách ly tín hiệu điều khiển và mạch công suất, bảo vệ các linh kiện nhạy cảm và nâng cao an toàn.

Khi tín hiệu điều khiển được cung cấp vào cổng điều khiển của Triac, nó sẽ bắt đầu dẫn điện và cho phép dòng điện chạy qua nó. Thời gian mà Triac dẫn điện trong mỗi nửa chu kỳ được điều chỉnh bởi tín hiệu điều khiển. Bằng cách điều chỉnh thời gian này, ta có thể điều chỉnh độ sáng hoặc công suất của thiết bị điện được kết nối với mạch kích Triac.

2.3.2. Sơ đồ mạch kích Triac:



Hình 11 Sơ đồ mạch kích Triac

• **CYMOC3023(U5):** có tác dụng cung cấp cách ly điện và chuyển đổi tín hiệu điều khiển từ dạng ánh sáng thành tín hiệu điện để kích hoạt kích triac chính.



Hình 12 MOC3023 DIP-6 Triac Driver Optocoupler

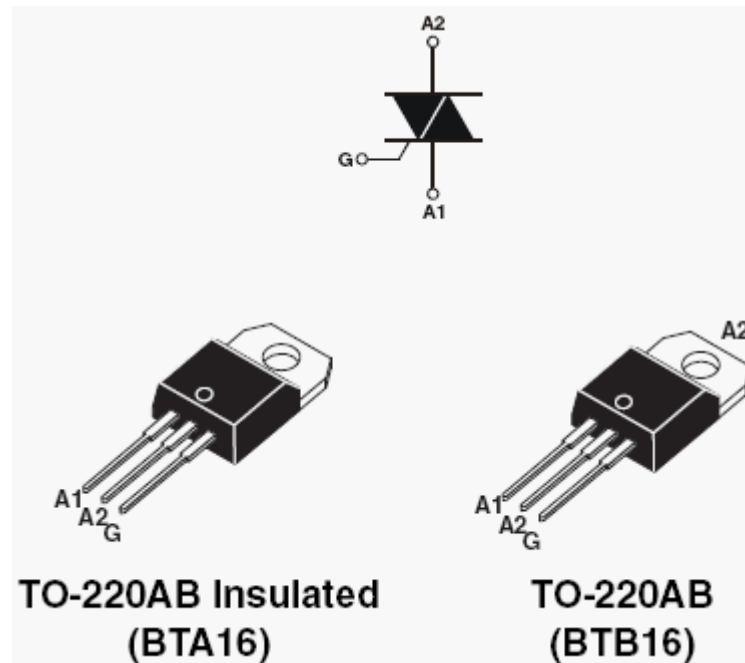
• **TRIAC BTA16(Q1):** điều khiển dòng điện và công suất đầu ra của các thiết bị điện thông qua việc điều chỉnh xung điện điều khiển áp dụng lên chân Gate của nó.



Hình 13 Triac BTA16 - 600B

Thông số kỹ thuật:

- Điện áp cực đại: 600V
- Dòng điện thuận cực đại: 16A
- Điện áp điều khiển mở van: 1.5V
- Dòng điều khiển mở van: 100mA
- Nhiệt độ làm việc: $-40^{\circ}\text{C} \sim 125^{\circ}\text{C}$



Hình 14 Sơ đồ chân Triac BTA16 - 600B

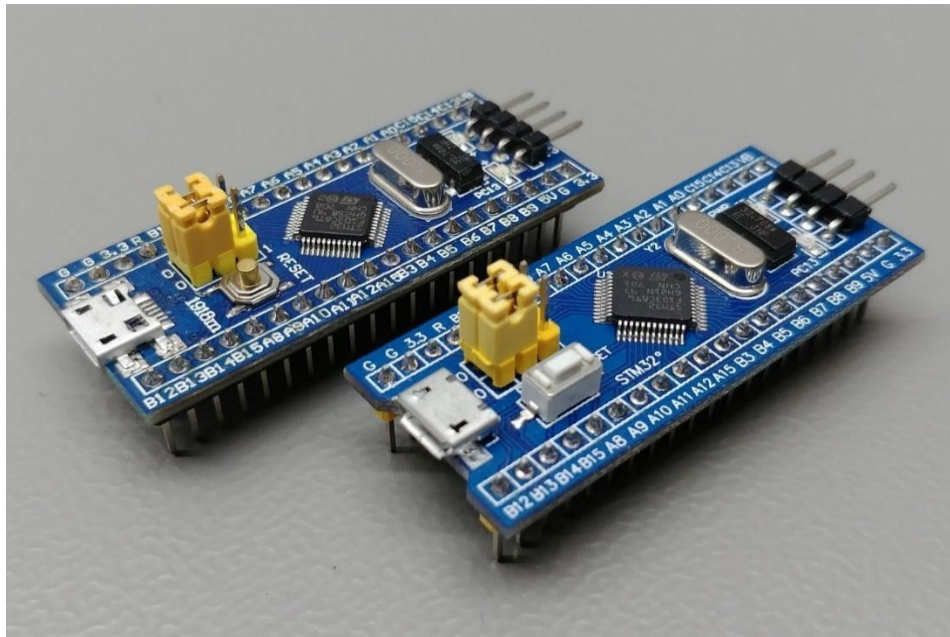
2.3.3. Nguyên lý hoạt động của mạch kích Triac:

Hoạt động của mạch kích triac dựa trên việc điều khiển kích triac thông qua tín hiệu điện áp áp dụng lên cổng Gate. Khi kích triac mở, nó cho phép dòng điện chạy qua nó và điều khiển dòng điện và công suất đầu ra của thiết bị điện liên quan.

2.4. Tổng quan về Vi điều khiển STM32:

2.4.1. Giới thiệu về vi điều khiển STM32:

STM32 là một dòng sản phẩm vi mạch đa chức năng được sử dụng rộng rãi trong lĩnh vực thiết kế điện tử. Loạt chip STM32 của STMicroelectronics hỗ trợ các giao thức giao tiếp như USB, UART, SPI và I2C, cung cấp tính linh hoạt và hiệu quả trong việc phát triển các ứng dụng đa năng. STM32 là một công nghệ đáng tin cậy, tiết kiệm năng lượng và được xem là giải pháp lý tưởng cho các nhà thiết kế điện tử hiện nay.



Hình 15 Vi điều khiển STM32F103C8T6

2.4.2. Các bước hoạt động của vi điều khiển STM32:

- **Khởi động:** Vi điều khiển được khởi động bằng cách cấp nguồn, và sau đó nạp chương trình thông qua quá trình nạp FLASH.

- **Thiết lập chế độ:** Sau khi khởi động, vi điều khiển được thiết lập để hoạt động trong chế độ yêu cầu, bao gồm đặt các thanh ghi và bộ đệm của nó để sẵn sàng cho các lệnh xử lý.

- **Xử lý:** Vi điều khiển STM32 xử lý các lệnh theo chương trình đã được nạp vào trong vi điều khiển, trong đó bao gồm các lệnh gọi hàm, điều khiển các bộ đệm và thanh ghi, và thực hiện các phép tính toán và xử lý thông tin.

- **Đáp ứng yêu cầu:** Vi điều khiển có thể đáp ứng các yêu cầu của hệ thống nhúng, bao gồm giao tiếp với các thiết bị khác, điều khiển và giám sát các thiết bị ngoại vi, đọc và ghi dữ liệu vào bộ nhớ, và thực hiện các chức năng khác như ngắt, đồng bộ, đo lường, điều khiển và xử lý tín hiệu.

2.4.3. Chức năng và tính năng của vi điều khiển STM32:

Vi điều khiển STM32 là một loại vi mạch đa chức năng được sử dụng rộng rãi trong các ứng dụng nhúng. Nó có nhiều tính năng và chức năng cơ bản như sau:

- **Kiến trúc:** Vi điều khiển STM32 có kiến trúc ARM Cortex và cung cấp nhiều tùy chọn cho các nhà phát triển để lựa chọn kiến trúc phù hợp với ứng dụng của họ.

- **Số chân:** Vi điều khiển STM32 cung cấp nhiều số lượng chân khác nhau cho các mục đích khác nhau. Nó cũng cung cấp nhiều chân GPIO để có thể điều khiển các thiết bị ngoại vi.

- **Kích thước:** Vi điều khiển STM32 có kích thước nhỏ và nhẹ, giúp nó dễ dàng tích hợp vào các ứng dụng nhỏ gọn.

- **Dung lượng bộ nhớ:** Vi điều khiển STM32 có dung lượng bộ nhớ lớn để lưu trữ chương trình và dữ liệu, giúp cho việc phát triển các ứng dụng nhúng dễ dàng hơn.

- **Tính năng bảo mật:** Vi điều khiển STM32 cung cấp nhiều tính năng bảo mật cho việc bảo vệ dữ liệu của người dùng.

- **Tính năng giao tiếp:** Vi điều khiển STM32 hỗ trợ nhiều chuẩn giao tiếp như USB, I2C, SPI, UART, CAN bus... giúp cho việc kết nối các thiết bị ngoại vi dễ dàng hơn.

- **Tính năng điều khiển PWM:** Vi điều khiển STM32 có tính năng điều khiển PWM giúp cho việc điều khiển tốc độ động cơ và độ sáng của đèn LED dễ dàng hơn.

2.4.4. Giới thiệu về STM32F103C8T6:

Tổng quan về Vi điều khiển STM32F103C8T6:

- STM32F103C8T6 là vi điều khiển 32bit, thuộc họ F1 của dòng chip STM32 hãng ST.

- Lõi ARM COTEX M3.

- Tốc độ tối đa 72Mhz.

- Bộ nhớ :

64 kbytes bộ nhớ Flash

20 kbytes SRAM

- Clock, reset và quản lý nguồn

Điện áp hoạt động từ 2.0 → 3.6V.

Sử dụng thạch anh ngoài từ 4Mhz → 20Mhz.

Thạch anh nội dùng dao động RC ở mode 8Mhz hoặc 40Khz.

- Chế độ điện áp thấp:

Có các mode: ngủ, ngừng hoạt động hoặc hoạt động ở chế độ chờ.

Cấp nguồn ở chân Vbat bằng pin ngoài để dùng bộ RTC và sử dụng dữ liệu được lưu trữ khi mất nguồn cấp chính.

- 2 bộ ADC 12 bit với 9 kênh cho mỗi bộ

Khoảng giá trị chuyển đổi từ 0 – 3.6 V

Có chế độ lấy mẫu 1 kênh hoặc nhiều kênh.

- DMA:

7 kênh DMA

Có hỗ trợ DMA cho ADC, UART, I2C, SPI.

- 7 bộ Timer:

3 Timer 16 bit hỗ trợ các mode Input Capture/ Output Compare/ PWM.

1 Timer 16 bit hỗ trợ để điều khiển động cơ với các mode bảo vệ ngắt Input, dead-time.

2 Watchdog Timer để bảo vệ và kiểm tra lỗi.

1 SysTick Timer 24 bit đếm xuống cho hàm Delay,....

- Có hỗ trợ 9 kênh giao tiếp:

2 bộ I2C.

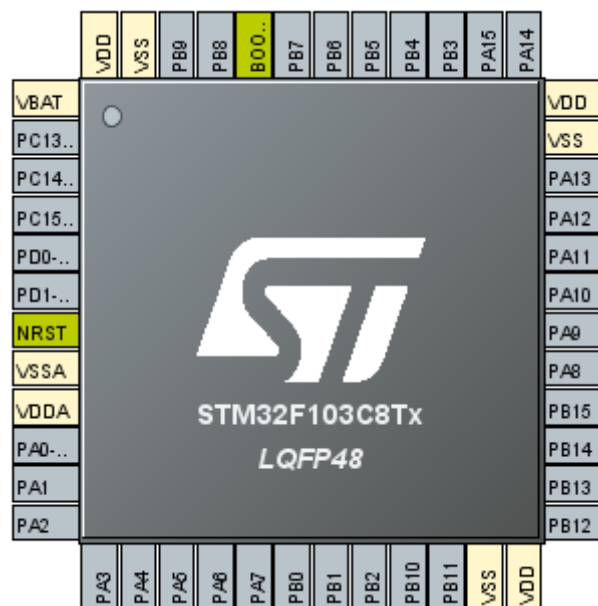
3 bộ USART

2 SPI

1 CAN

USB 2.0 full-speed interface

- Kiểm tra lỗi CRC và 96-bit ID.



Hình 16 Mô phỏng Vi điều khiển STM32F103C8T6

2.4.5. So sánh giữa STM32F103C8T6 và Arduino Uno R3:

TÊN	STM32S103C8T6	ARDUINO
KIẾN TRÚC VÀ BỘ VI XỬ LÝ	Là một vi điều khiển ARM Cortex-M3, với tốc độ xung nhịp cao và hiệu năng mạnh mẽ. Nó có bộ nhớ Flash lên đến 64KB và RAM lên đến 20KB.	Là một nền tảng phát triển dựa trên vi điều khiển AVR của Atmel. Tùy thuộc vào phiên bản, Arduino có các vi điều khiển với bộ nhớ và hiệu năng khác nhau.
CẤU HÌNH VÀ KHẢ NĂNG MỞ RỘNG	STM32F103C8T6 cung cấp nhiều chân I/O (đầu vào/đầu ra) và các tính năng nâng cao như UART, SPI, I2C, ADC, PWM, v.v. Nó cũng hỗ trợ nhiều nguồn ngoại vi và các chức năng phần cứng khác.	Arduino được thiết kế với một số chân I/O và cung cấp các giao tiếp phổ biến như UART, SPI và I2C. Tuy nhiên, Arduino có thể mở rộng qua việc sử dụng các module bổ sung hoặc sử dụng các board phát triển có khả năng kết nối với nhiều loại nguồn ngoại vi.
PHẦN MỀM VÀ MÔI TRƯỜNG PHÁT TRIỂN	Thường sử dụng môi trường phát triển tích hợp như Keil MDK hoặc STM32CubeIDE. Ngôn ngữ lập trình phổ biến là C/C++.	Arduino có một môi trường phát triển dễ sử dụng và thân thiện với người mới bắt đầu gọi là Arduino IDE. Nó sử dụng ngôn ngữ lập trình dựa trên Wiring, một biến thể của C/C++, và cung cấp một bộ thư viện phong phú cho các chức năng điều khiển.
ỨNG DỤNG	Thường được sử dụng trong các ứng dụng yêu cầu hiệu năng cao và kiểm soát chính xác, ví dụ như điều khiển robot, hệ thống tự động hóa, IoT, v.v.	Thường được sử dụng trong các ứng dụng đơn giản, các dự án học tập và thử nghiệm, và các ứng dụng nhỏ và đơn giản như đèn LED điều khiển, đo lường nhiệt độ, v.v.

Bảng 1 So sánh đặc tính STM32F103C8T6 và ARDUINO UNO R3

Features	STM32F103	ATMEGA328
Clock Frequency	72 Mhz	16 Mhz
I2C Buses	2	1
SPI Buses	2	1
CAN Bus	Yes	No
Analog Channel	10	8
PWM Channel	15	6
USART Buses	3	1
GPIO's	32	24
On Board RTC	Yes	No
Architecture	ARM Cortex M3 32 bit	AVR RISC 8 bit
ADC Resolution	12 bit	10 bit
Quantization Level	4096	1024
Flash Memory	64KB	32KB
SRAM	20KB	2KB
Debugging	Serial, JTAG	Serial
PWM Resolution	16 bit	10bit
Price	110	115

Bảng 2 So sánh thông số kỹ thuật giữa STM32F103C8T6 và ARDUINO UNO R3

Nhìn chung, STM32F103C8T6 mạnh hơn ARDUINO UNO R3 nhưng giá thành lại rẻ hơn nhiều, ngoài ra nhóm nghiên cứu quyết định tìm hiểu STM32F103C8T6 vì trước đó đã từng làm nhiều đồ án về ARDUINO.

2.5. Tổng quan về màn hình LCD 1602:

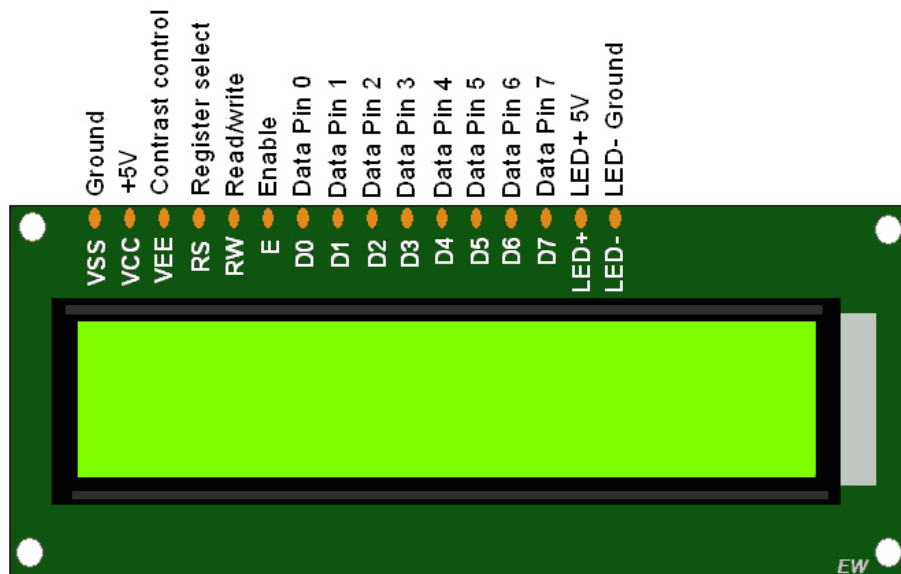
2.5.1. Màn hình LCD 1602 là gì?

LCD 1602 là màn hình tinh thể lỏng nhỏ dùng để hiển thị chữ hoặc số trong bảng mã ACSII. Màn hình sử dụng là màn hình text LCD 1602 xanh dương, có khả năng hiển thị 4 dòng với mỗi dòng là 20 ký tự, các ký tự có màu trắng và có LCD 1602 sử dụng MPU SPLC780D-01 điều khiển và truy xuất dữ liệu LCD 1602 nghĩa là loại có 4 dòng và mỗi dòng chỉ hiển thị được 20 ký tự. Đây là loại màn hình được sử dụng rất phổ biến trong các loại mạch điện.

Ngày nay, thiết bị hiển thị LCD 1602 (Liquid Crystal Display) được sử dụng trong rất nhiều các ứng dụng của VDK. LCD 1602 có rất nhiều ưu điểm so với các dạng hiển thị khác như: khả năng hiển thị ký tự đa dạng (chữ, số, ký tự đồ họa); dễ dàng đưa vào mạch ứng dụng theo nhiều giao thức giao tiếp khác nhau, tiêu tốn rất ít tài nguyên hệ thống, giá thành rẻ,...

Thông số kỹ thuật của LCD 1602:

- Điện áp MAX : 7V
- Điện áp MIN : - 0,3V
- Hoạt động ổn định : 2.7-5.5V
- Điện áp ra mức cao : > 2.4
- Điện áp ra mức thấp : <0.4V
- Dòng điện cấp nguồn : 350uA - 600uA
- Nhiệt độ hoạt động : - 30 - 75 độ C



Hình 18 LCD 1602 xanh lá

2.5.2. Chức năng của từng chân LCD 1602:

- **Chân số 1 - VSS:** chân nối đất cho LCD được nối với GND của mạch điều khiển.

- **Chân số 2 - VDD:** chân cấp nguồn cho LCD, được nối với VCC=5V của mạch điều khiển.

- **Chân số 3 - VE:** điều chỉnh độ tương phản của LCD.

- **Chân số 4 - RS:** chân chọn thanh ghi, được nối với logic "0" hoặc logic "1":

+ Logic "0": Bus DB0 - DB7 sẽ nối với thanh ghi lệnh IR của LCD (ở chế độ "ghi" - write) hoặc nối với bộ đếm địa chỉ của LCD (ở chế độ "đọc" - read).

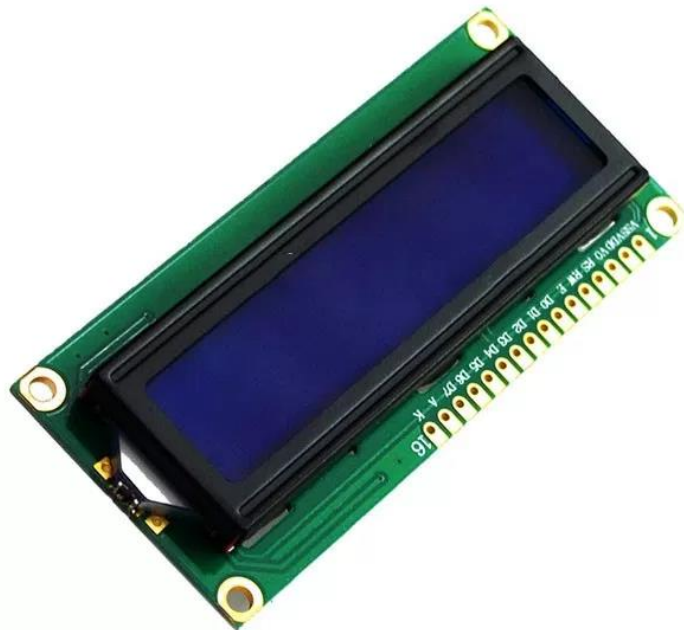
+ Logic “1”: Bus DB0 - DB7 sẽ nối với thanh ghi dữ liệu DR bên trong LCD.

- **Chân số 5 - R/W**: chân chọn chế độ đọc/ghi (Read/Write), được nối với logic “0” để ghi hoặc nối với logic “1” đọc.

- **Chân số 6 - E**: chân cho phép (Enable). Sau khi các tín hiệu được đặt lên bus DB0-DB7, các lệnh chỉ được chấp nhận khi có 1 xung cho phép của chân này như sau:

+ Ở chế độ ghi: Dữ liệu ở bus sẽ được LCD chuyển vào thanh ghi bên trong khi phát hiện một xung (high-to-low transition) của tín hiệu chân E.

+ Ở chế độ đọc: Dữ liệu sẽ được LCD xuất ra DB0-DB7 khi phát hiện cạnh lên (low-to-high transition) ở chân E và được LCD giữ ở bus đến khi nào chân E xuống mức thấp.



Hình 19 LCD 1602 Xanh dương 5V

- **Chân số 7 đến 14 - D0 đến D7:** 8 đường của bus dữ liệu dùng để trao đổi thông tin với MPU. Có 2 chế độ sử dụng 8 đường bus này là: Chế độ 8 bit (dữ liệu được truyền trên cả 8 đường, với bit MSB là bit DB7) và Chế độ 4 bit (dữ liệu được truyền trên 4 đường từ DB4 tới DB7, bit MSB là DB7).

- **Chân số 15 - A :** nguồn dương cho đèn nền.

- **Chân số 16 - K :** nguồn âm cho đèn nền.

2.6. Tổng quan về MAX31865 và PT100:

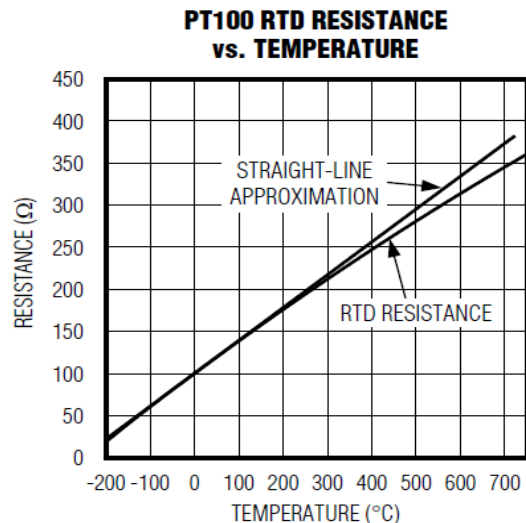
2.6.1. Max31865 và PT100 là gì?

MAX31865 là một bộ chuyển đổi giá trị điện trở sang kỹ thuật số số dễ sử dụng, được tối ưu hóa cho cảm biến nhiệt độ dựa trên điện trở platinum (RTD).

Cảm biến nhiệt độ PT100 là một loại cảm biến nhiệt độ dựa trên nguyên lý thay đổi điện trở của một RTD (Resistance Temperature Detector) được làm từ vật liệu platinum. PT100 có ý nghĩa là cảm biến có điện trở 100 ohm ở nhiệt độ 0°C.

PT100 sử dụng điện trở của RTD để đo nhiệt độ. Khi nhiệt độ tăng, điện trở của RTD tăng theo một quy luật cụ thể. Cảm biến PT100 được sử dụng rộng rãi trong các ứng dụng đo nhiệt độ chính xác, như trong các hệ thống kiểm soát nhiệt độ công nghiệp, phòng thí nghiệm, thiết bị y tế, và các ứng dụng khoa học và nghiên cứu khác. Hai giá trị được sử dụng rộng rãi nhất cho alpha (α) là

0.00385 và 0.00392, tương ứng với tiêu chuẩn IEC 751 (PT100) và tiêu chuẩn SAMA.



The resistance vs. temperature curve is reasonably linear, but has some curvature, as described by the Callendar-Van Dusen equation:

$$R(T) = R_0(1 + aT + bT^2 + c(T - 100)T^3)$$

where:

T = temperature ($^{\circ}\text{C}$)

$R(T)$ = resistance at T

R_0 = resistance at $T = 0^{\circ}\text{C}$

IEC 751 specifies $\alpha = 0.00385055$ and the following Callendar-Van Dusen coefficient values:

$$a = 3.90830 \times 10^{-3}$$

$$b = -5.77500 \times 10^{-7}$$

$$c = -4.18301 \times 10^{-12} \text{ for } -200^{\circ}\text{C} \leq T \leq 0^{\circ}\text{C}, 0 \text{ for } 0^{\circ}\text{C} \leq T \leq +850^{\circ}\text{C}$$

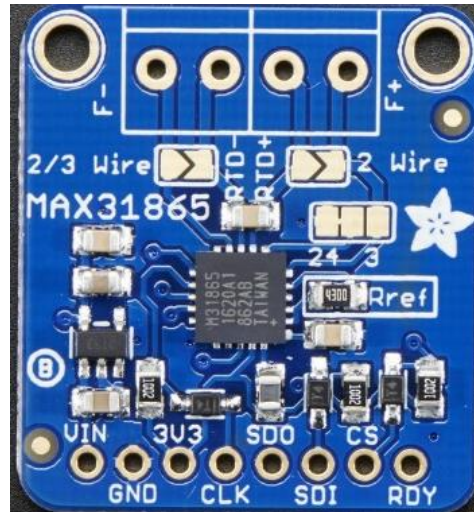
Hình 20 Mối tương quan giữa nhiệt độ và điện trở

Đường cong điện trở so với nhiệt độ có tính tuyến tính hợp lý, nhưng có một số độ cong, được mô tả bởi phương trình Callendar-Van Dusen ở hình trên.

Ở đề tài lần này, nhóm nghiên cứu sử dụng kết nối cảm biến PT100 loại 3 dây với MAX31865.

Dải đo trung bình của cảm biến PT100 dây dao động khoảng -40 đến 200 độ C, maximum có thể lên đến 400 độ C.

2.6.2. Quá trình kết nối PT100 với MAX31865



Hình 21 MAX31865 Adafruit

Vin - đây là chân cấp điện. Max31865 sử dụng điện áp 3V và các nhà thiết kế đã thêm 1 bộ ổn áp. Chính vì vậy với STM32F103C8T6 ta có thể cấp 3.3V hay 5V đều được.

3V3 - đây là chân đầu ra 3.3V từ bộ điều tiết áp, ta có thể lấy tối đa 100mA từ đây nếu cần.

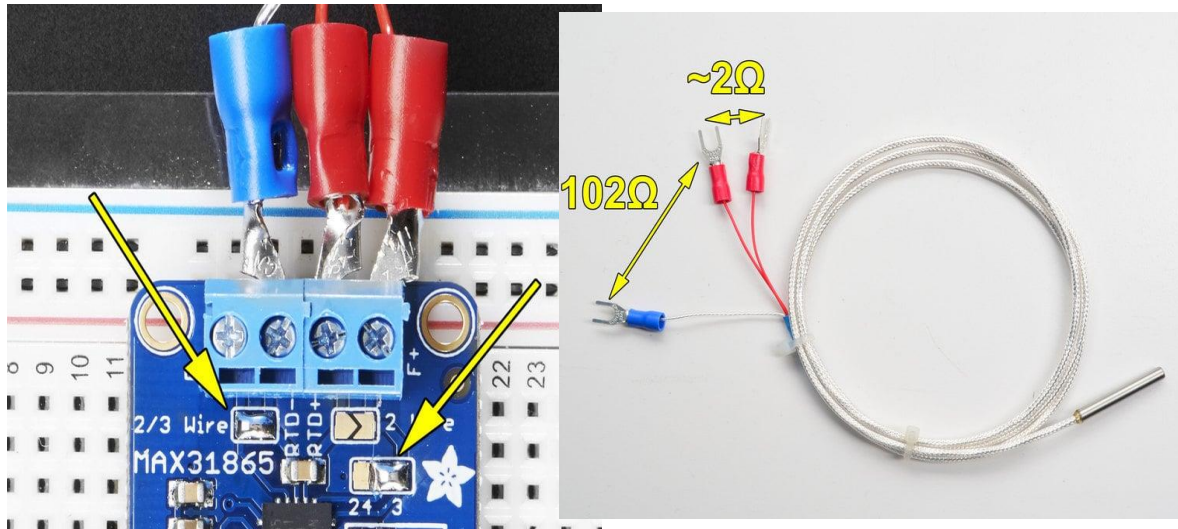
GND - chân mass chung cho nguồn và logic.

SCK - Đây là chân SPI Clock, là một chân INPUT của chip.

SDO - đây là chân Serial Data Out / Microcontroller In Sensor Out, dùng để gửi dữ liệu từ MAX31865 đến bộ xử lý STM32.

SDI - đây là chân Serial Data In / Microcontroller Out Sensor In, dùng để gửi dữ liệu từ bộ xử lý (STM32) của ta đến MAX31865.

CS - đây là chân Chip Select, khi có mức thấp, sẽ bắt đầu một truyền thông SPI. Đây là một chân đầu vào của chip.



Hình 22 PT100 và MAX31865

Sử dụng VOM để xác định các chân cần kết nối. Khi sử dụng loại cảm biến 3 dây, ta cần hàn kết nối tại các điểm mũi tên màu vàng – 2/3 Wire và “4 3”. Và không được quên phải loại bỏ kết nối giữa “2 4”.

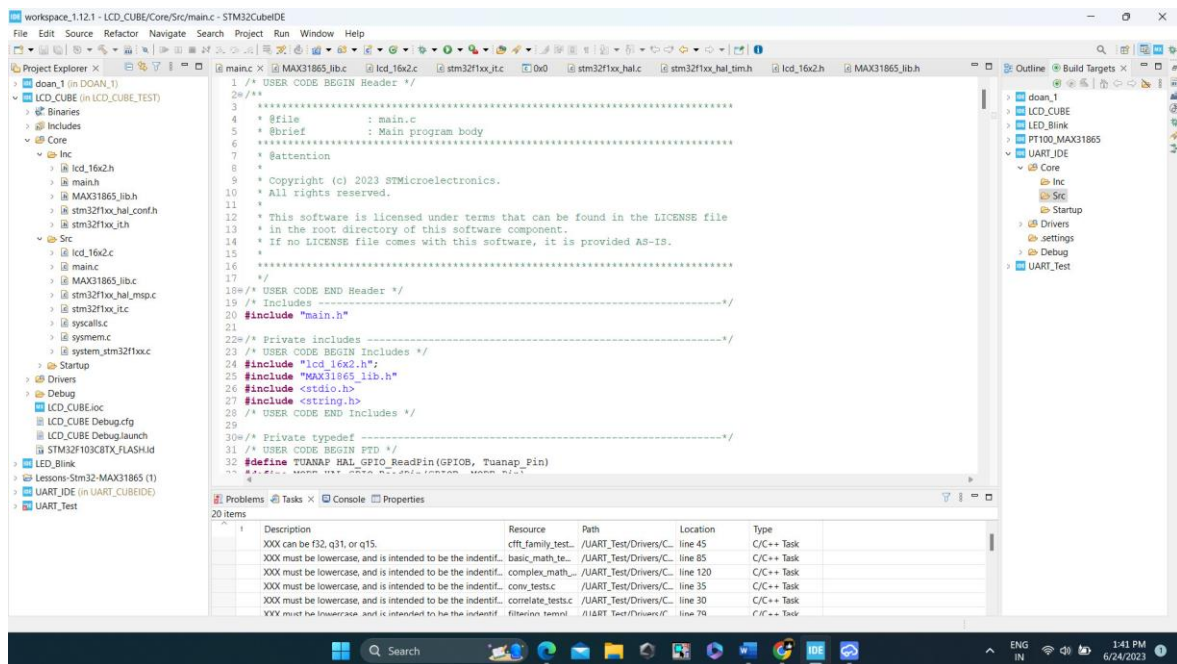
Sau khi làm đủ các bước trên, ta đã sẵn sàng để kết nối SPI giữa MAX31865 và STM32 để thực hiện lập trình.

2.7. Phần mềm STM32Cube IDE:

2.7.1. Tổng quan về STM32Cube IDE:

STM32Cube IDE là một môi trường phát triển tích hợp (IDE) được cung cấp bởi STMicroelectronics, một nhà sản xuất linh kiện điện tử hàng đầu. Cube IDE được thiết kế đặc biệt để hỗ trợ việc phát triển ứng dụng nhúng trên các vi điều khiển STM32 của STMicroelectronics.

Với STM32Cube IDE, bạn có thể phát triển và gỡ lỗi các ứng dụng nhúng cho các vi điều khiển STM32 bằng ngôn ngữ lập trình C/C++. IDE này cung cấp một loạt các công cụ và tính năng hữu ích để giúp bạn xây dựng ứng dụng nhúng một cách dễ dàng và hiệu quả.

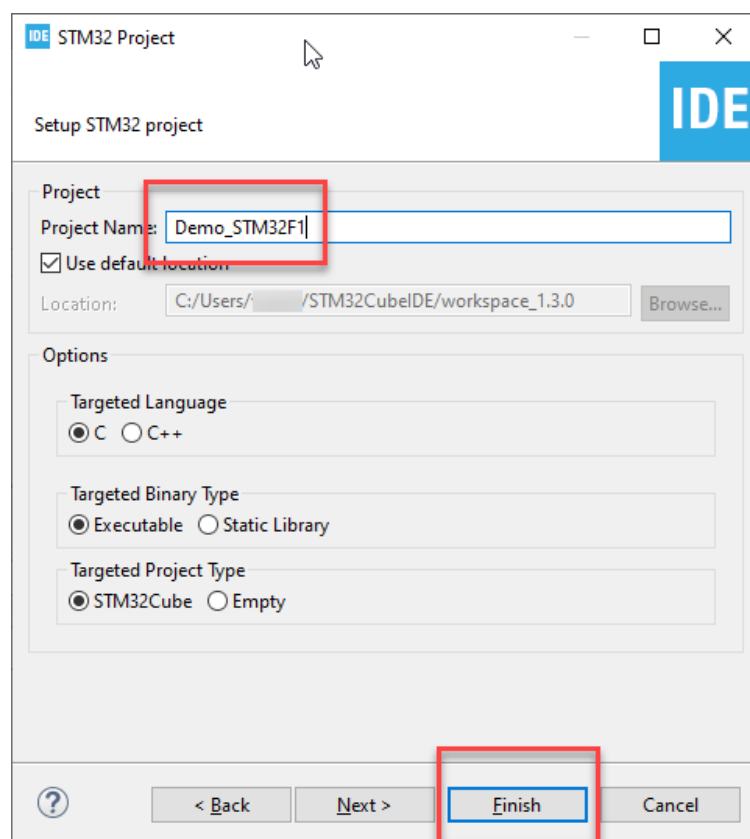


Hình 23 Giao diện làm việc trên phần mềm STM32Cube IDE

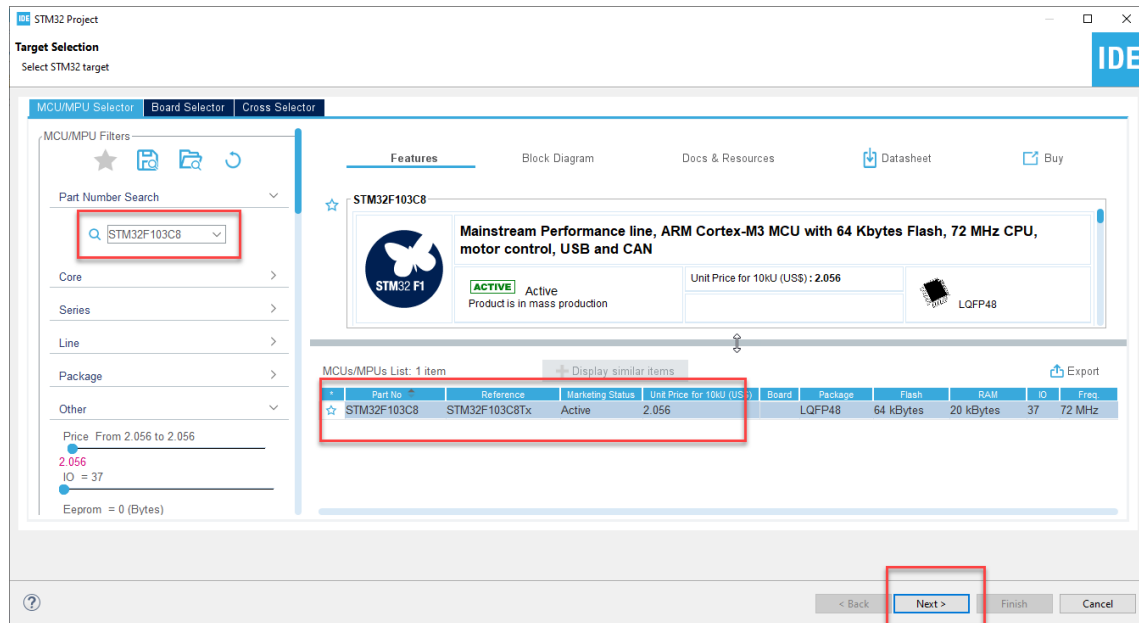
2.7.2. Cách tạo một Project mới trên STM32Cube IDE:

Sau khi mở phần mềm, để tạo một project mới, ta thực hiện thao tác:

Select File > New > STM32 Project. Chọn vào Target Selection, chọn MCU/MPU Selector và tìm chip STM32F1Cx, sau đó chọn Next.



Hình 24 Tạo project mới trên STM32 Cube IDE



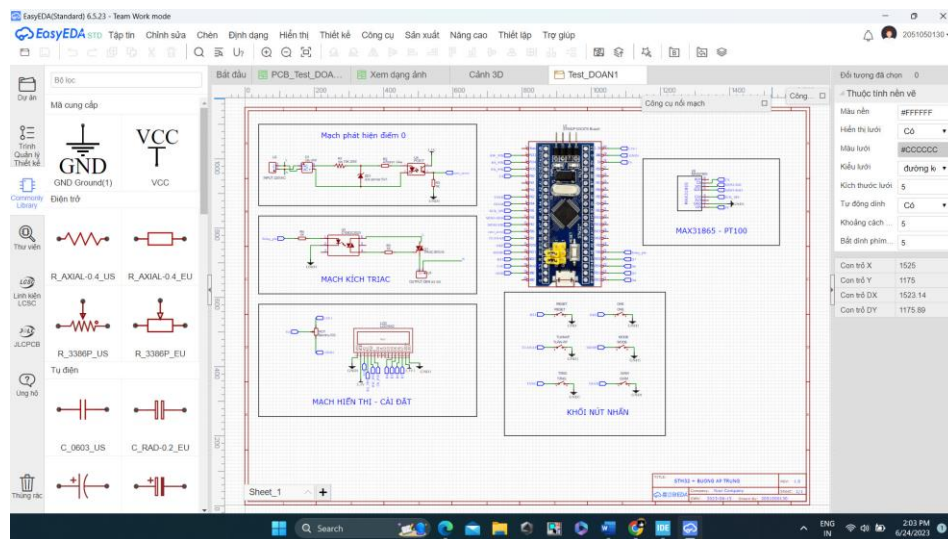
Hình 25 Tạo project mới trên STM32 Cube IDE

2.8. Phần mềm Easy EDA:

2.8.1. Tổng quan về phần mềm Easy EDA:

EasyEDA là một phần mềm thiết kế mạch điện trực tuyến (EDA - Electronic Design Automation) mạnh mẽ và dễ sử dụng. Nó cung cấp một môi trường đồ họa để bạn có thể thiết kế mạch điện, mô phỏng, và làm việc với các linh kiện điện tử.

Phần mềm Easy EDA cho phép bạn thiết kế mạch điện một cách trực quan, mô phỏng và mô phỏng ngược các linh kiện trong mạch điện trước khi tiến hành sản xuất, thiết kế bo mạch in PCB một cách chính xác,... Nhìn chung, đây là phần mềm phù hợp và dễ sử dụng đối với những người làm điện tử.

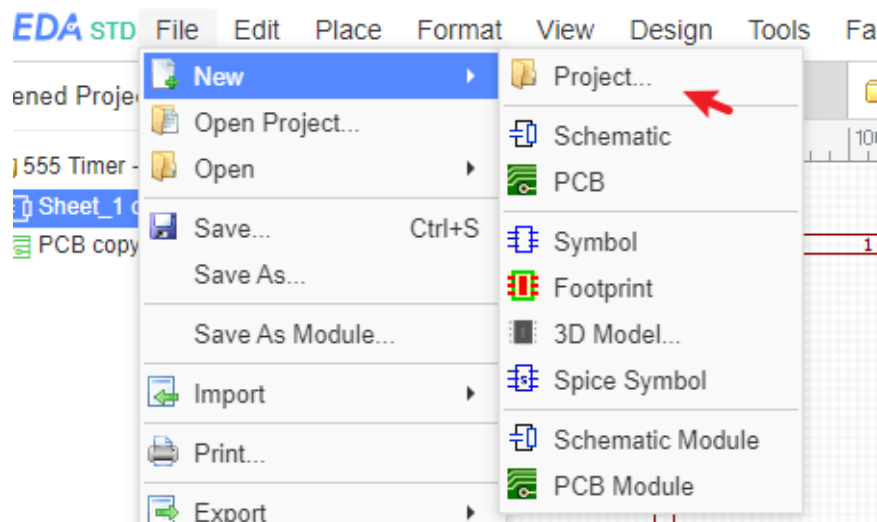


Hình 26 Giao diện làm việc trên phần mềm Easy EDA

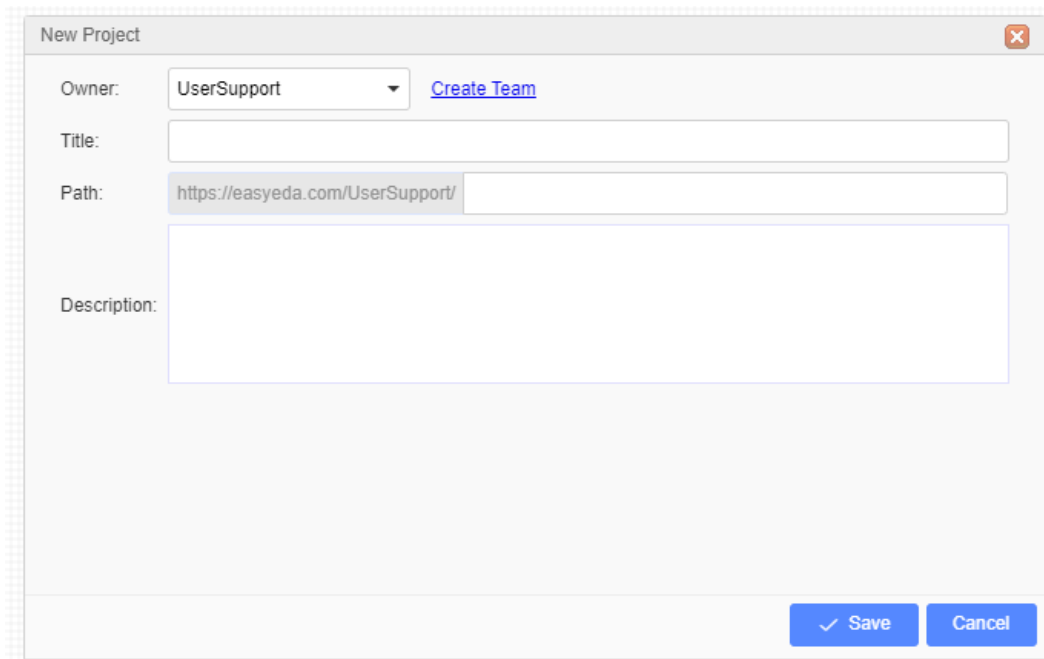
2.8.2. Cách tạo một project mới trên phần mềm Easy EDA:

Sau khi mở phần mềm, để tạo một project mới, ta thực hiện các thao tác:

File > New > Create a new project/Schematic..etc



Hình 27 Hướng dẫn tạo project mới trên Easy EDA



Hình 28 Hướng dẫn tạo project mới trên Easy EDA

Trong đó:

Owner: Đây là mục cho phép thay đổi chủ sở hữu của dự án; ta có thể chuyển chủ sở hữu cho nhóm mà mình đã tham gia.

Title: Tiêu đề hiển thị của dự án.

Path: EasyEDA cho phép ta đặt đường dẫn cho dự án, giúp dễ dàng chia sẻ với mọi người.

Description: Thêm một mô tả ngắn giúp bất kỳ ai bạn chia sẻ dự án này hiểu rõ về nội dung của dự án.

CHƯƠNG 3. LẬP TRÌNH – THIẾT KẾT VÀ KẾT QUẢ ĐẠT ĐƯỢC

Ở chương này, nhóm nghiên cứu sẽ tiến hành phân tích sơ đồ khối, lưu đồ thuật toán, sơ đồ nguyên lý hệ thống, chạy code – nạp code, cũng như đấu nối thực tế và kiểm thử thực tế. Từ đó, người đọc chắc chắn có thể hiểu sâu về hướng đi và quá trình thực hiện của nhóm, đồng thời nhìn thấy rõ những kết quả mà nhóm nghiên cứu đạt được.

3.1. Yêu cầu của hệ thống – Sơ đồ khối – Chức năng của các khối:

3.1.1. Yêu cầu của hệ thống “buồng áp trứng tự động”

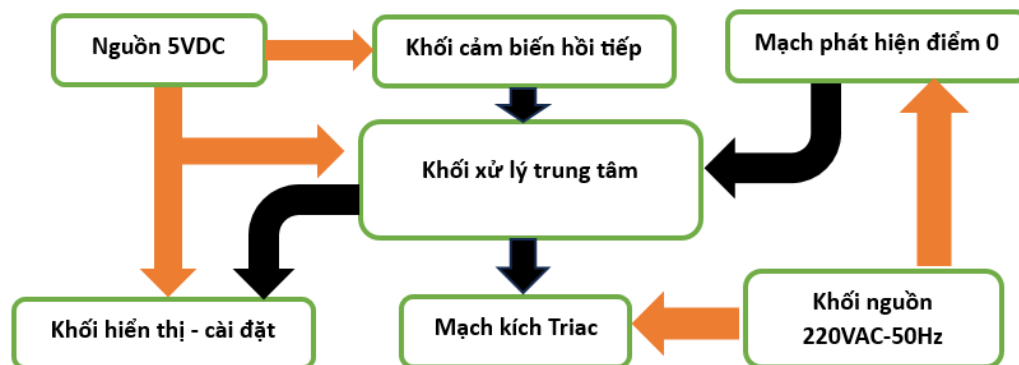
❖ Hệ thống có các chức năng sau:

- Cài đặt ngưỡng nhiệt độ ấp trứng tương ứng với 3 tuần ấp.
- Cài đặt ngưỡng nhiệt độ bất kì.
- Liên tục hiển thị và cập nhật nhiệt độ lên LCD 1602.

Nhiệt độ trong buồng sẽ liên tục bám theo nhiệt độ đặt với sai số xác lập dưới 0.04 độ C, thời gian xác lập dưới 1 phút (với nhiệt độ tuần ấp), độ vọt lố < 0.25%.

3.1.2. Sơ đồ khối và chức năng mỗi khối

Sơ đồ khối:



Hình 29 Sơ đồ khối

Trong đó, đường vàng, đường đen lần lượt là dây nguồn và dây tín hiệu.

Chức năng mỗi khối:

Khối nguồn 5VDC: Cấp nguồn cho khối xử lý trung tâm, khối hiển thị và khối cảm biến.

Khối cảm biến hồi tiếp: Sử dụng MAX31865 đọc tín hiệu điện trở từ cảm biến PT100, sau đó giao tiếp SPI với khối xử lý trung tâm.

Khối hiển thị - cài đặt: Sử dụng màn hình LCD 16x02 dùng để hiển thị nhiệt độ và cài đặt nhiệt độ cho buồng áp trướng.

Mạch phát hiện điểm 0: Mạch này sẽ lấy thời điểm tín hiệu điện AC gần bằng 0, ngay lúc này sẽ gửi tín hiệu đến khối xử lý trung tâm để thực hiện kiểm soát công suất cho nhiệt độ buồng áp trướng.

Mạch kích Triac: nhận tín hiệu điểm 0, kết hợp với thuật toán PID từ đó sẽ xuất ra tín hiệu Digital phù hợp để kiểm soát nhiệt độ buồng.

Khối nguồn 220VAC-50Hz: Sử dụng nguồn điện gia đình cấp nguồn cho bóng đèn sợi đốt và mạch phát hiện điểm 0.

Khối xử lý trung tâm: Sử dụng VĐK STM32F103C8T6 để lập trình chuyển đổi giá trị điện trở từ PT100 về giá trị nhiệt độ chuẩn xác, nhận tín hiệu điểm 0 kết hợp với thuật toán PID, ngắt ngoài v.v...

3.2. Quá trình xây dựng – hoạt động của hệ thống:

3.2.1. Giai đoạn 1: Xây dựng khối cảm biến nhiệt độ và khối hiển thị.

B1: Tìm hiểu về VDK STM32F103C8T6, lập trình chớp tắt led cơ bản, lý thuyết ngắt, timer, SPI, I2C, ADC.

B2: Chọn 1 phần mềm để lập trình VDK xuyên suốt đồ án, ở đây nhóm nghiên cứu sử dụng Cube IDE để lập trình, nạp code và debug.

B3: Đọc hiểu thư viện LCD.c và Max31865.c, add thư viện vào source code. Điều chỉnh giá trị #Define Rref trên MAX31865 đúng với giá trị đo được từ VOM. Từ đó ta sẽ có được giá trị nhiệt độ đúng nhất.

3.2.2. Giai đoạn 2: Xây dựng mạch phát hiện điểm 0 và mạch kích Triac.

B1: Cần tìm hiểu kỹ lý thuyết về 2 mạch này (đã được giải thích ở chương 1), sau khi nắm vững lý thuyết ta tiến hành đấu nối trên testboard.

B2: Ta kiểm tra mạch phát hiện điểm 0 có hoạt động tốt hay không bằng cách sử dụng ngắt ngoài ở STM32, đặt 1 biến đếm và cho biến đếm tăng lên mỗi khi xảy ra ngắt.

B3: Thực hiện debug và quan sát biến đếm có tăng đều với tần số 100Hz hay không, nếu không đều, cần xử lý chống nhiễu trên chân ngắt ngoài.

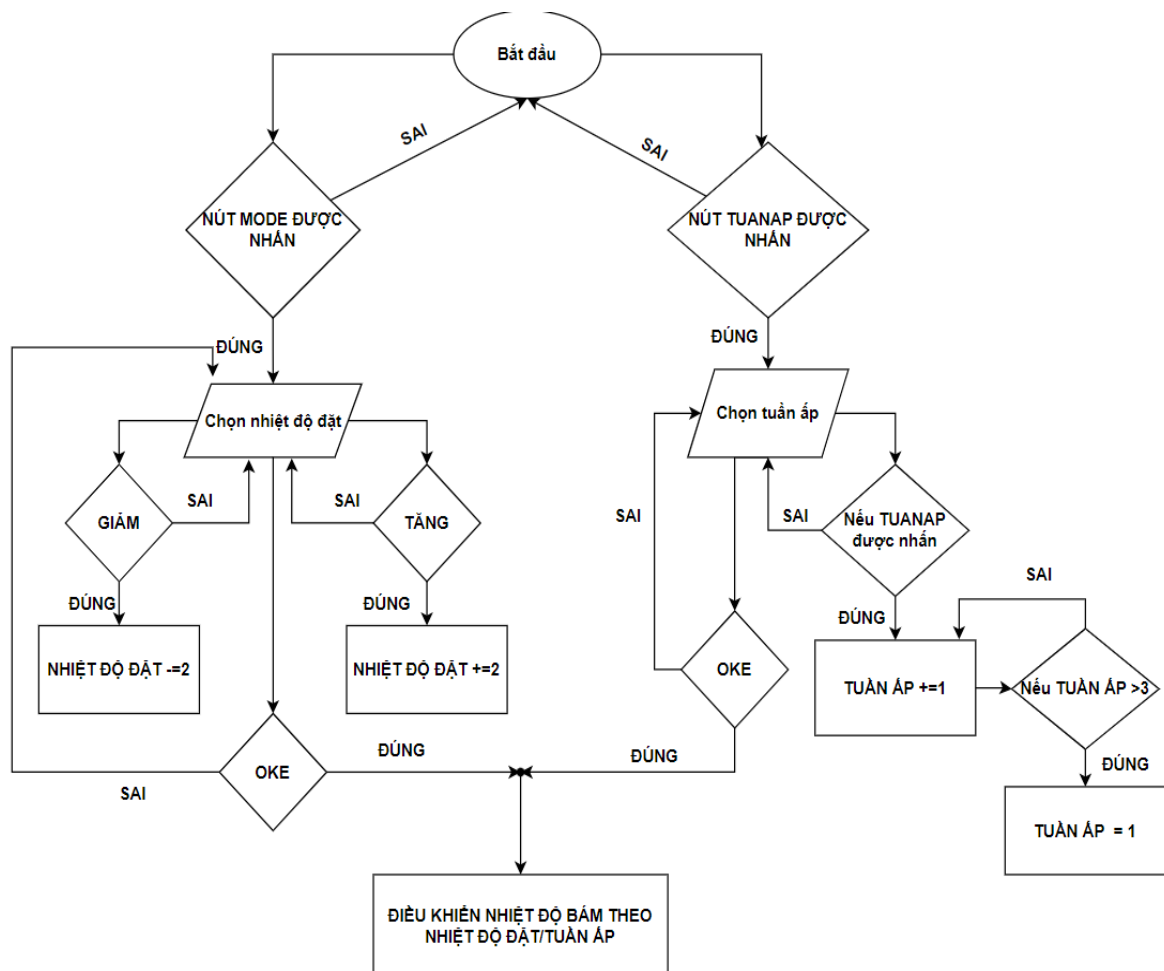
3.2.3. Giai đoạn 3: bắt đầu viết thuật toán PID, viết thuật toán sử dụng cho hệ thống.

B1: Tìm hiểu về PID, nắm rõ công thức và chuyển đổi thành code. Đặt các biến như PID_error, PID_value, v.v...

B2: Kích hoạt 1 timer trên STM32 với mục đích tạo delay Microsecond, từ đó kiểm soát công suất của bóng đèn.

B3: Sau khi hiệu chỉnh các tham số Kp, Ki, Kd ổn. Ta tiến hành lập trình giải thuật theo cài đặt người dùng.

3.3. Lưu đồ thuật toán hệ thống áp trướng



Bảng 3 Lưu đồ thuật toán hệ thống được vẽ trên drawio

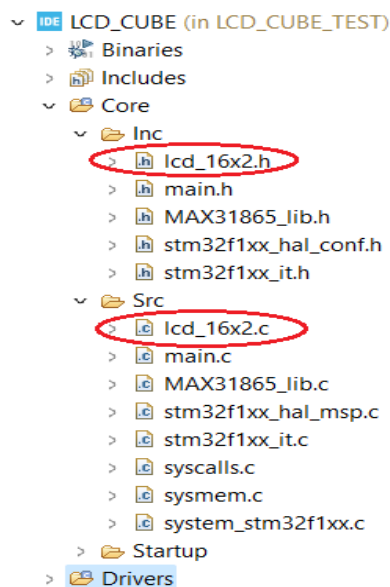
- Nút “BẮT ĐẦU” tượng trưng cho hành động cấp nguồn cho hệ thống.
- Ở giao diện ban đầu người dùng có thể chọn 1 trong 2 chế độ “TUANAP” hoặc “MODE”.
- Chế độ “TUANAP” chúng ta sẽ tiến hành cài đặt chọn nhiệt độ cho tuần thứ bao nhiêu để áp trứng không bị dị tật, sai sót.
- Chế độ “MODE” dùng để thử nghiệm cài đặt một nhiệt độ bất kì (cao hơn nhiệt độ môi trường).
- Cả hai chế độ sau khi được cài đặt xong sẽ bắt đầu đi vào trạng thái gia nhiệt theo thuật toán PID.

3.4. Cấu hình và bắt đầu lập trình STM32F103C8T6 trên Cube IDE

3.4.1. Các thư viện sử dụng

- Thư viện LCD_1602 và MAX31865.

Để dễ dàng lập trình cho màn hình LCD, ta cần add 2 thư viện trên vào cây thư mục của Project.



Hình 30 Add thư viện LCD vào Project

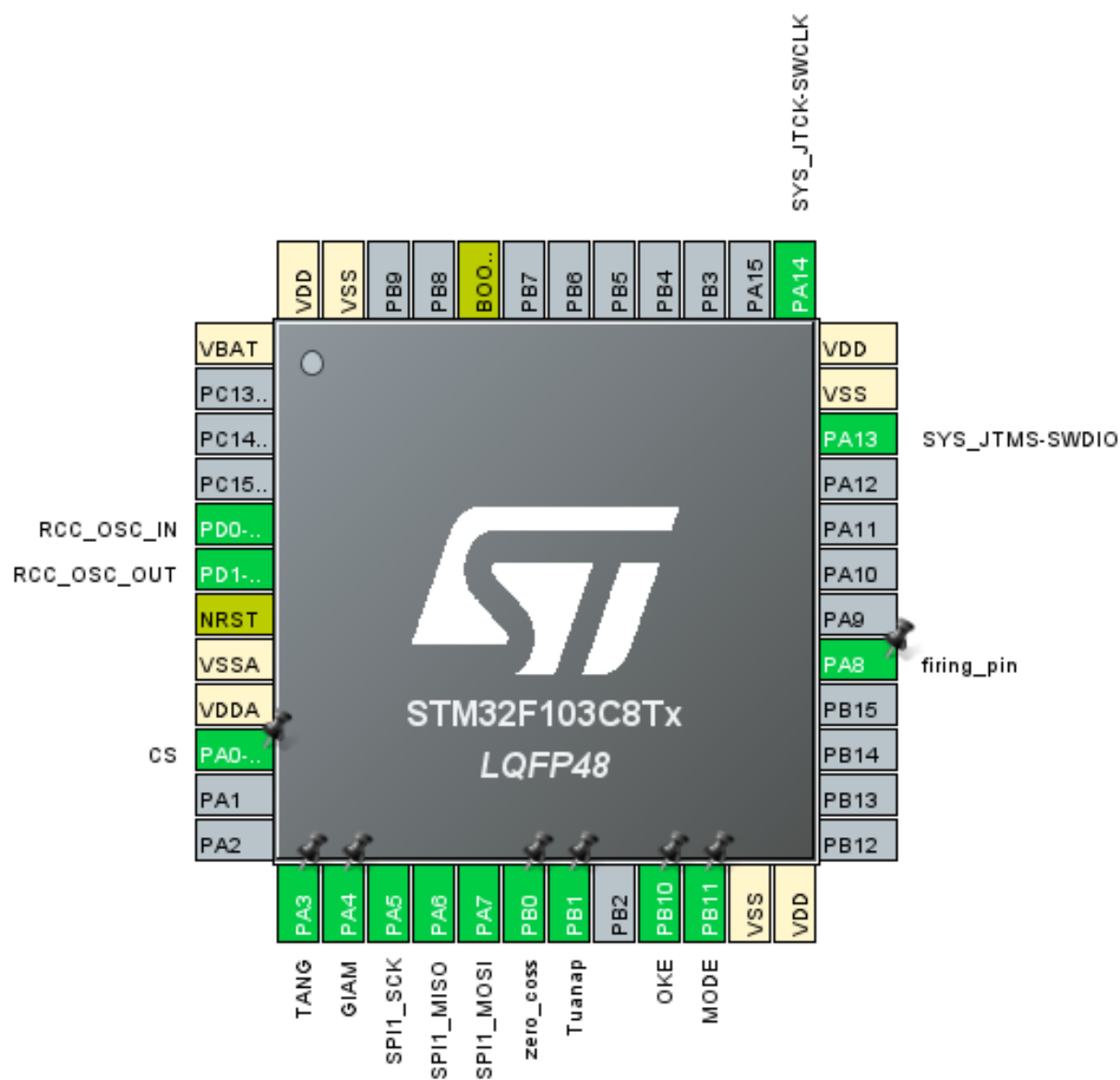
Vào Inc để add thư viện “.h” và vào Src để add thư viện “.c”. Ta làm tương tự với “MAX31865.h” và “MAX31865.c”.

3.4.2. Cấu hình Pinout cho STM32

Với STM32, nhà phát triển đã thêm tính năng cấu hình pinout cho VĐK trực tiếp trên phần mềm CubeMX. Nhờ vậy người lập trình có cái nhìn toàn diện hơn về hệ thống. VD: cấu hình tần số thạch anh, cấu hình timer, SPI, UART, ngắt, GPIO, v.v...

❖ Ở đề tài này chúng ta cần cấu hình như sau:

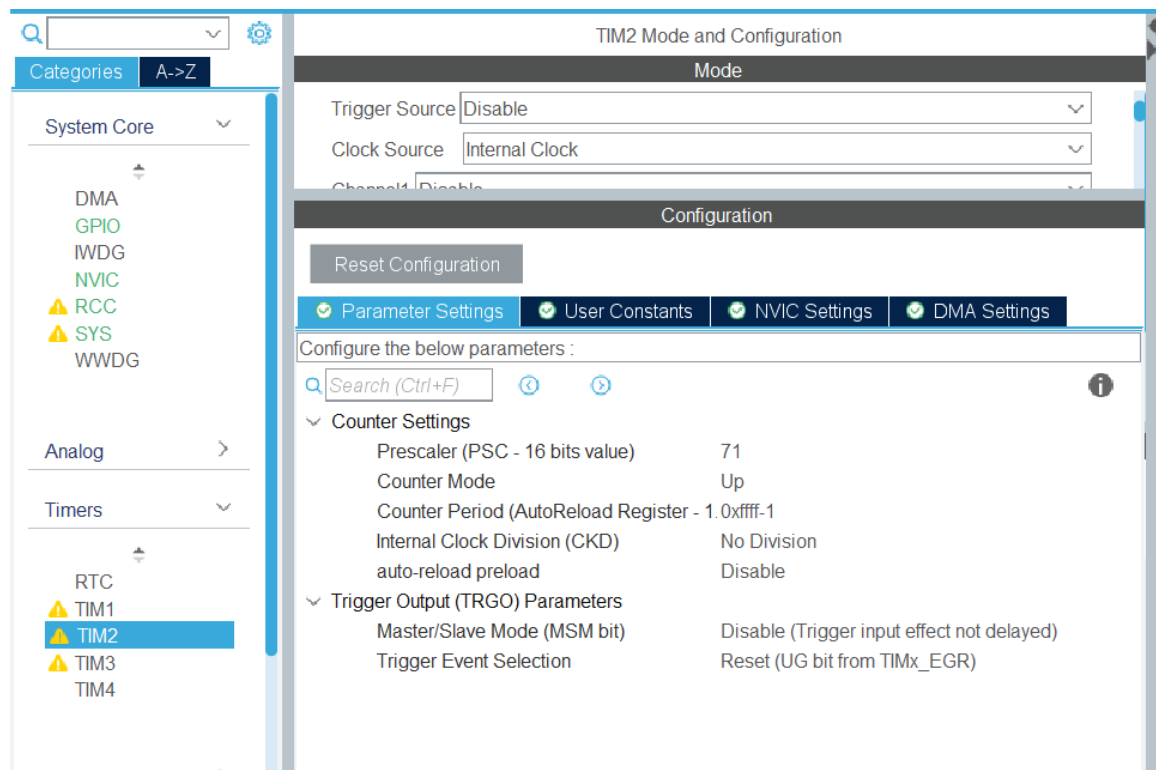
- Các chân giao tiếp SPI với MAX31865.
- Cấu hình ngắt ngoài để nhận tín hiệu đếm 0, nút Tăng, nút Giảm.
- Cấu hình Timer để viết hàm delay_us (dùng để kiểm soát công suất).
- Cấu hình các chân giao tiếp với LCD, chân OUTPUT firing_pin, và các chân chọn chế độ OKE, MODE, TUANAP.



Hình 31 Cấu hình tổng quát STM32F103C8T6

NVIC Code generation		
NVIC Interrupt Table		
Interrupt	Enabled	Preemption Priority
Non maskable interrupt	<input checked="" type="checkbox"/>	0
Hard fault interrupt	<input checked="" type="checkbox"/>	0
Memory management fault	<input checked="" type="checkbox"/>	0
Prefetch fault, memory access fault	<input checked="" type="checkbox"/>	0
Undefined instruction or illegal state	<input checked="" type="checkbox"/>	0
System service call via SWI instruction	<input checked="" type="checkbox"/>	0
Debug monitor	<input checked="" type="checkbox"/>	0
Pendable request for system service	<input checked="" type="checkbox"/>	0
Time base: System tick timer	<input checked="" type="checkbox"/>	15
PVD interrupt through EXTI line 16	<input type="checkbox"/>	0
Flash global interrupt	<input type="checkbox"/>	0
RCC global interrupt	<input type="checkbox"/>	0
EXTI line0 interrupt	<input checked="" type="checkbox"/>	1
EXTI line3 interrupt	<input checked="" type="checkbox"/>	0
EXTI line4 interrupt	<input checked="" type="checkbox"/>	0
TIM2 global interrupt	<input type="checkbox"/>	0
SPI1 global interrupt	<input type="checkbox"/>	0

Hình 32 Cấu hình các ngắt ngoài và mức độ ưu tiên ngắt



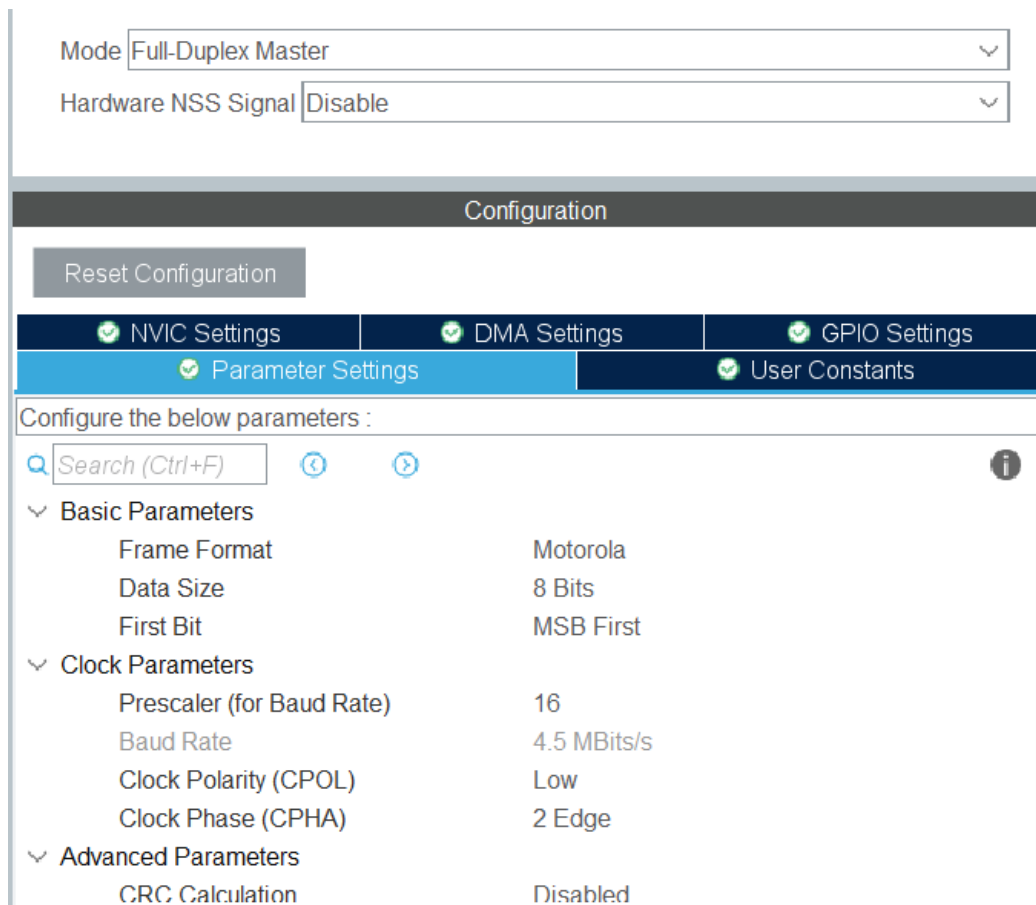
Hình 33 Cấu hình timer2

Vì tần số thạch anh là 72MHz, nên ta chọn bội số chia là 71 (“71+1” hệ thống sẽ tự cộng lên 1 cho bội số chia). Như vậy ta có tần số dao động là 1MHz. Đổi sang chu kỳ $T = 1\mu s$ cho mỗi lần đếm lên 16bit.

Chế độ đếm lên 16 bit ngẫu nhiên, với chế độ này ta đã có thể viết được hàm delay microsecond.

```
void delay_us (uint16_t us)
{
    HAL_TIM_SET_COUNTER(&tim2,0); // set the counter value a 0
    while ( __HAL_TIM_GET_COUNTER(&tim2) < us); // wait for the counter to reach the us
}
```

Hình 34 Hàm delay micro giây với timer2



Hình 35 Cấu hình SPI

Trong datasheet của MAX31865, nhà sản xuất đã ghi rõ tốc độ truyền dữ liệu tối đa của MAX31865 là 5MBit/s. Với tần số thạch anh 72MHz ta chọn bội số là 16, như vậy tốc độ truyền tải dữ liệu là 4,5Mbit/s.

AC Electrical Characteristics: SPI Interface

($3.0V \leq V_{DD} \leq 3.6V$, $T_A = -40^\circ C$ to $+125^\circ C$, unless otherwise noted. Typical values are $T_A = +25^\circ C$, $V_{DD} = V_{DVDD} = 3.3V$.) (Notes 3 and 7) (Figure 1 and Figure 2)

PARAMETER	SYMBOL	CONDITIONS	MIN	TYP	MAX	UNITS
Data to SCLK Setup	t_{DC}	(Notes 8, 9)	35			ns
SCLK to Data Hold	t_{CDH}	(Notes 8, 9)	35			ns
SCLK to Data Valid	t_{CDD}	(Notes 8, 9, 10)			80	ns
SCLK Low Time	t_{CL}	(Note 9)	100			ns
SCLK High Time	t_{CH}	(Note 9)	100			ns
SCLK Frequency	t_{CLK}	(Note 9)	DC		5.0	MHz
SCLK Rise and Fall	t_R, t_F	(Note 9)			200	ns
\overline{CS} to SCLK Setup	t_{CC}	(Note 9)	400			ns
SCLK to \overline{CS} Hold	t_{CCH}	(Note 9)	100			ns
\overline{CS} Inactive Time	t_{CWH}	(Note 9)	400			ns
\overline{CS} to Output High-Z	t_{CDZ}	(Notes 8, 9)			40	ns
Address 01h or 02h Decoded to \overline{DRDY} High	t_{DRDYH}	After RTD register read access (Note 9)		50		ns

Hình 36 Tốc độ truyền tải SPI của MAX31865

3.4.3. Chuyển đổi thuật toán PID thành ngôn ngữ C/C++

- ❖ Từ thuật toán PID ở chương 2, ta dễ dàng chuyển đổi lưu đồ thành code C như sau:
 - Đặt các biến toàn cục cần thiết.

```

float setpoint = 40.0f;
//PID variables
float PID_error = 0;
float previous_error = 0;
float elapsedTime, Time, timePrev;
int PID_value = 0;
//PID constants
int kp = 300;
float ki = 3;
int kd = 400;
float PID_p = 0;
float PID_i = 0;
float PID_d = 0;
/* USER CODE END PTD */

```

Hình 37 Đặt các biến PID

- ✓ “setpoint” là điểm nhiệt độ đặt tham chiếu.
- ✓ PID_error là biến lưu trữ sai số giữa nhiệt độ đặt và nhiệt độ thực tế.
- ✓ Previous_error, elapsedTime, Time, timePrev là các biến dùng để lưu trữ và tính toán cho PID_d.
- ✓ PID_value là biến lưu trữ giá trị PID: $PID_value = PID_p + PID_i + PID_d$.
- ✓ Kp,ki,kd lần lượt là hệ số tỷ lệ, hệ số tích phân và hệ số đạo hàm của điều khiển PID.

```
PID_error = setpoint - PT100_Temperature ; //Calculate the pid

if (PID_error > 3.1) //integral constant will only affect e
{ PID_i = 0; }

PID_p = kp * PID_error; //Calculate the P value
PID_i = PID_i + (ki * PID_error); //Calculate the I value
// timePrev = Time; // the previous time is
Time = HAL_GetTick(); // actual time read
elapsedTime = (Time - timePrev) / 1000;
PID_d = kd * ((PID_error - previous_error) / elapsedTime);
PID_value = PID_p + PID_i + PID_d;

//We define firing delay range between 0 and 8300. Read above
if (PID_value < 0) { PID_value = 0; }
if (PID_value > 8300) { PID_value = 8300; }
realTemp_LCD();
previous_error = PID_error;
```

Hình 38 Thuật toán PID code C/C++

- Biến PID_p là biến tỷ lệ, sai số càng lớn thì PID_p càng lớn và ngược lại.
- Biến PID_i là biến tích phân, biến này sẽ bắt đầu cộng dồn khi sai số bé hơn 3.1 độ C. Vì khi gần đạt ngưỡng nhiệt độ đặt, PID_p sẽ rất nhỏ dẫn

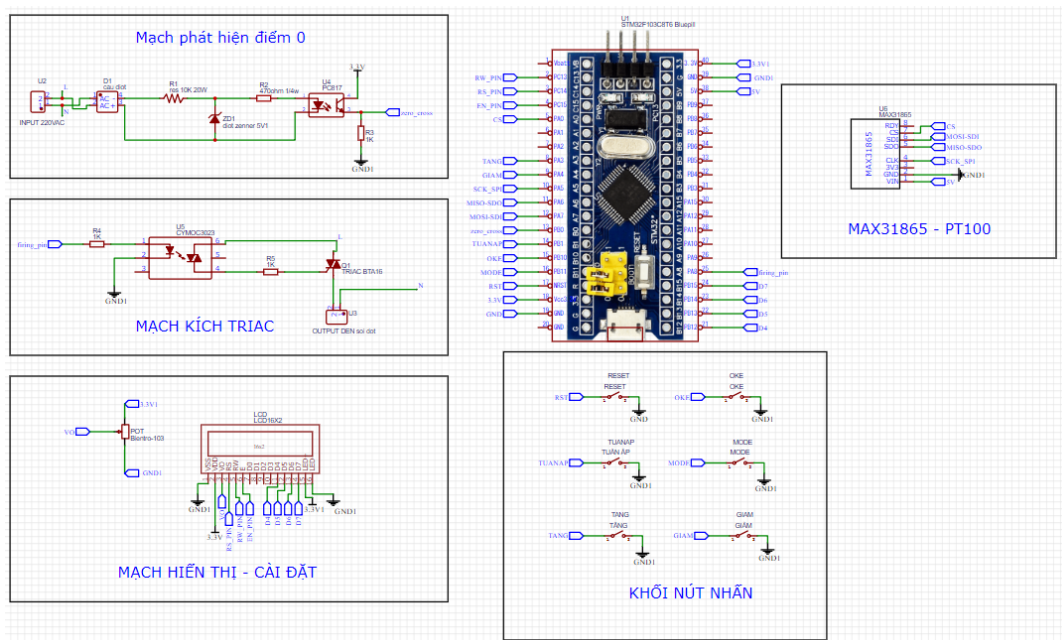
đèn không đủ nhiệt độ cấp cho buồng, khi này PID_i sẽ được tăng lên và đảm bảo nhiệt độ sẽ vẫn tăng đều, tránh vọt lố.

- PID_d là biến đạo hàm, đây là biến sẽ được so sánh với biến trước đó, nếu nhiệt độ có sự thay đổi lớn một cách đột ngột, PID_d sẽ đóng vai trò đảm bảo công suất đầu ra cho bóng đèn, giúp cho hệ thống không bị tăng hay giảm nhiệt độ đột ngột.

Các hệ số K_p, K_i, K_d sẽ được chọn qua quá trình thử nghiệm thực tế.

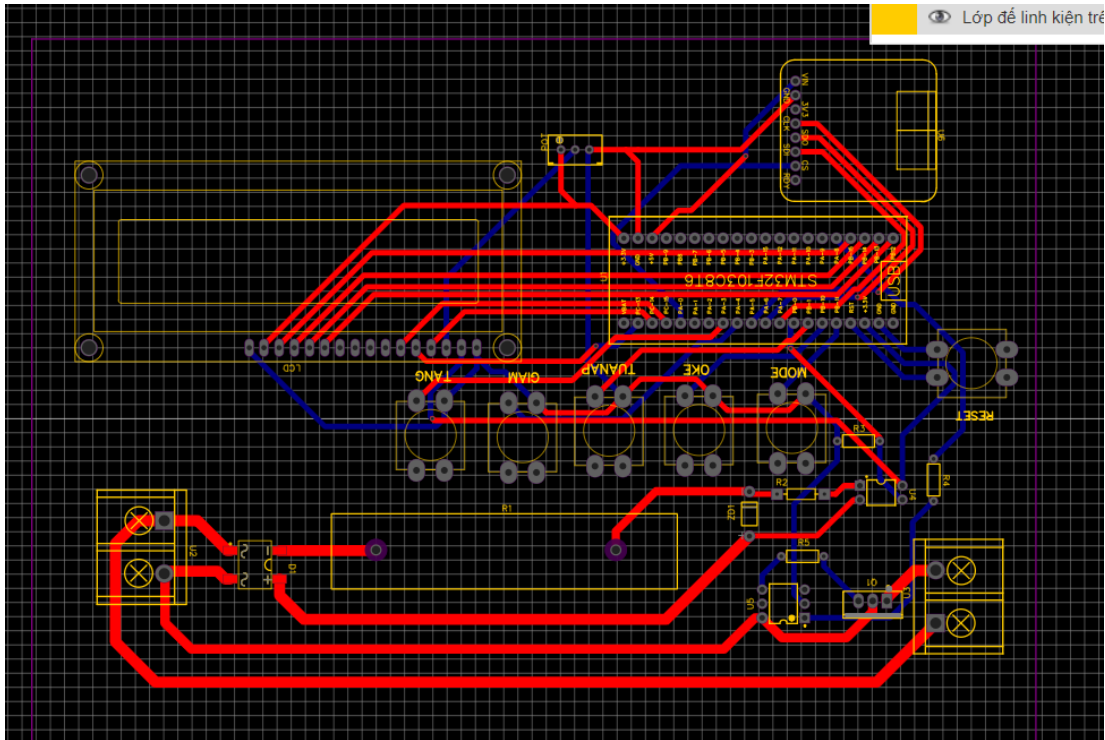
3.5. Sơ đồ nguyên lý toàn bộ hệ thống và Layout PCB

Sơ đồ nguyên lý và sơ đồ Layout PCB cho toàn bộ hệ thống được nhóm nghiên cứu vẽ trên phần mềm EASY EDA.



Hình 39 Schematic toàn hệ thống

- Vì đề tài sử dụng nhiều ngoại vi nên nhóm nghiên cứu chọn vẽ mạch PCB 2 lớp, sử dụng các lỗ Via để nhảy đường dây trên board.

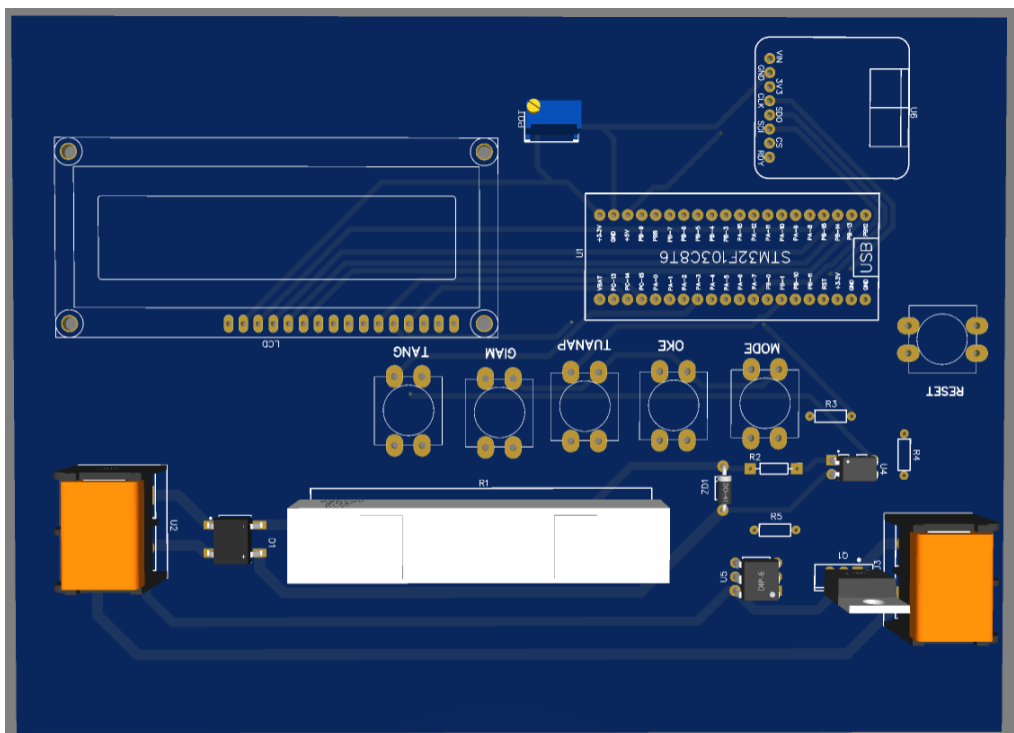


Hình 40 Sơ đồ PCB

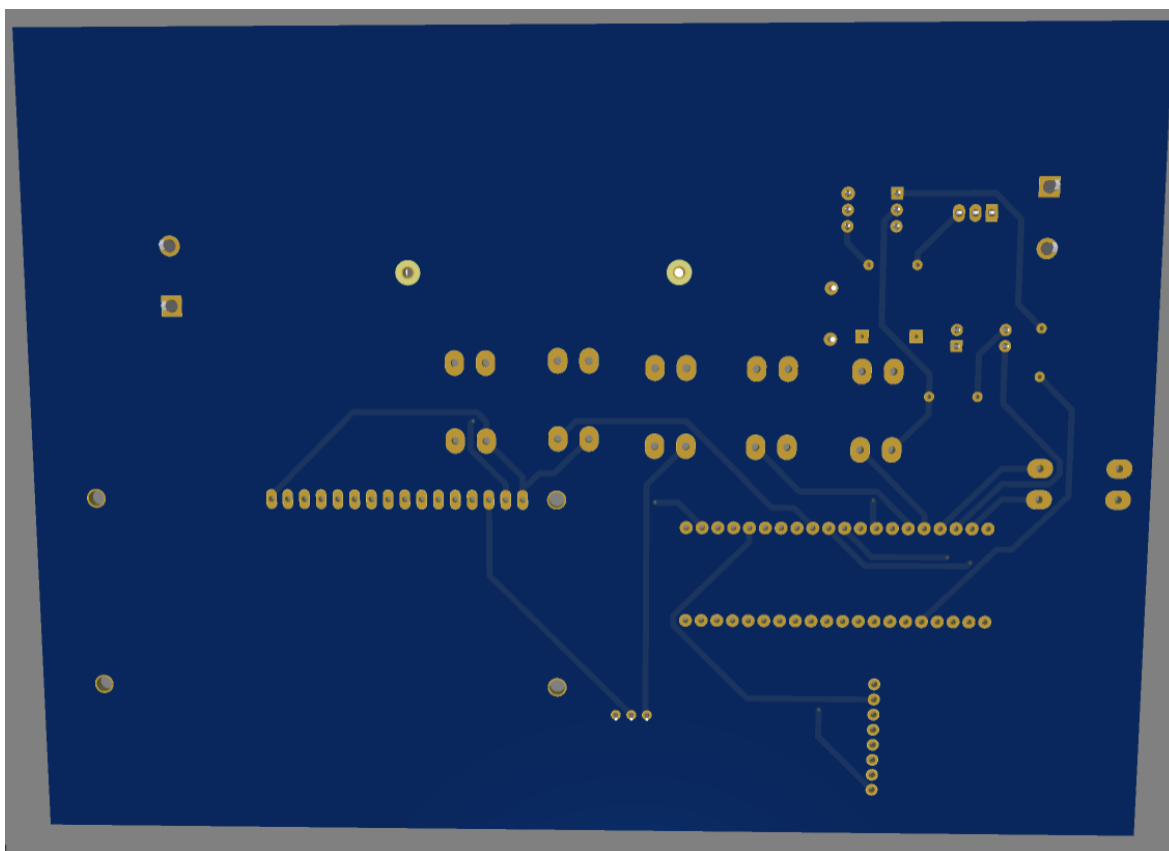
Sau khi vẽ sơ đồ nguyên lý, ta lưu file và convert Schematic thành PCB, ban đầu sau khi convert, các linh kiện được sắp xếp ngẫu nhiên. Việc của chúng ta là sắp xếp các linh kiện 1 cách hợp lý mà không gây nhiễu mạch hoặc ngược mạch.

VD: Điện trở 10K-20W khi hoạt động sẽ tỏa nhiệt rất lớn, vì vậy ta cần cách ly điện trở này để không gây hư hỏng các linh kiện khác. Hoặc khi sắp xếp INPUT nguồn xoay chiều, INPUT cảm biến, v.v... cần phải sắp xếp hướng hợp lý để không bị chân cắm hướng vào trong, gây cản trở khi hàn board.

- Sau khi thiết kế xong mạch PCB ta có thể xem trước mạch ở chế độ 3D.



Hình 41 Mặt trước board

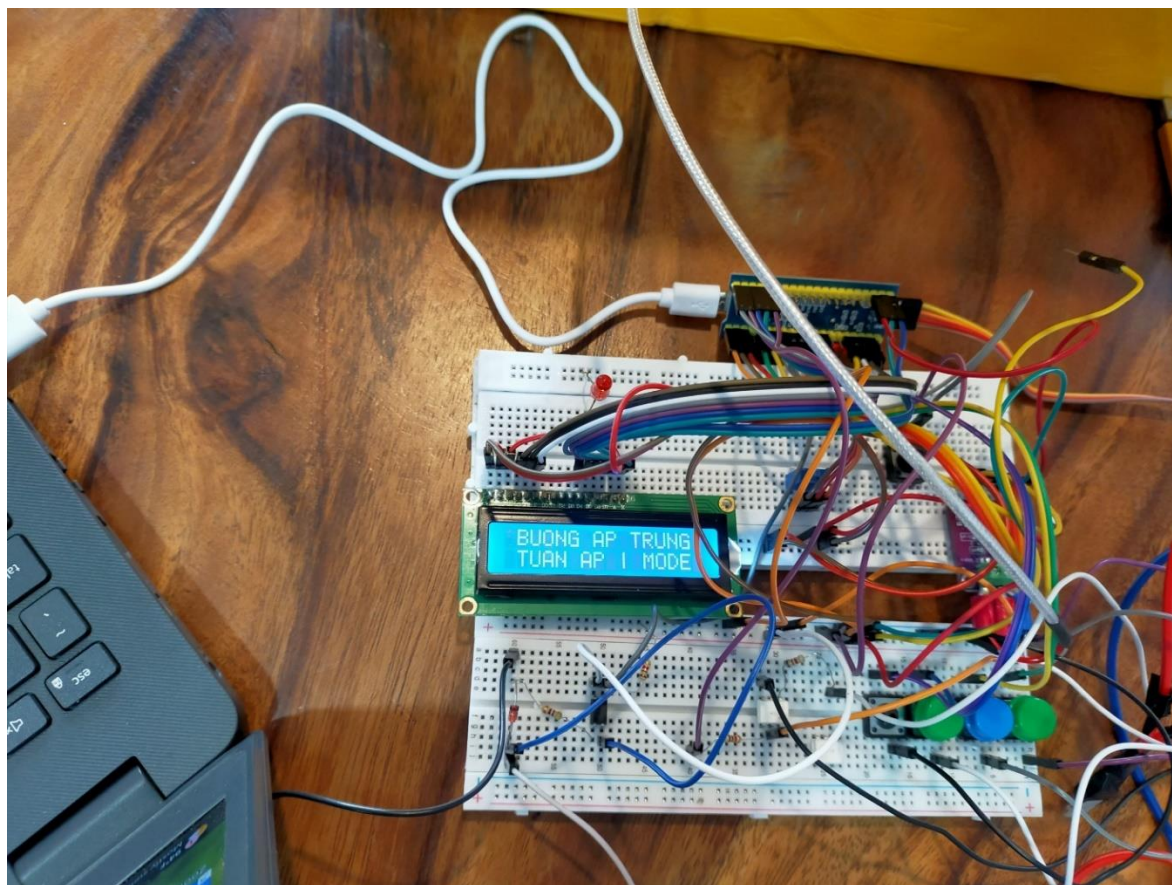


Hình 42 Mặt sau board

Sau khi thiết kế mạch, ta có thể xuất file Gerber cho các công ty làm mạch. Ở đồ án 1, nhóm nghiên cứu được sự cho phép của GVHD quyết định chỉ dừng lại ở mức chạy hệ thống buồng áp trứng trên test board.

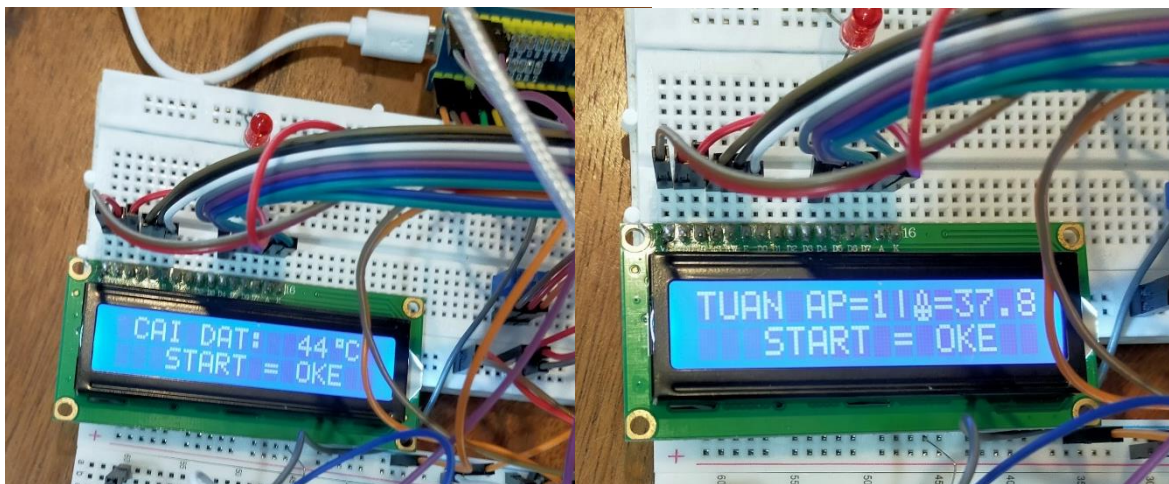
3.6. Tiến hành đấu nối và kết quả đạt được

Nhóm nghiên cứu sử dụng breadboard và jumper wire để kết nối các modul, linh kiện với nhau.



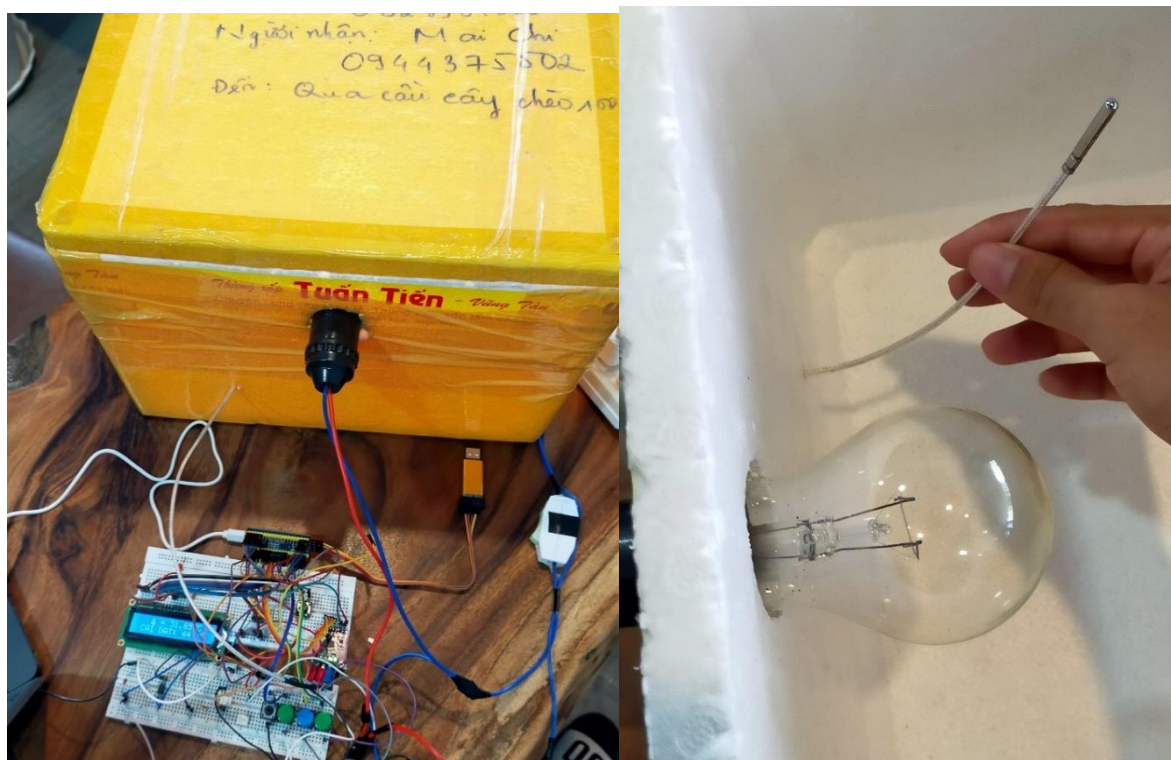
Hình 43 Đấu nối testboard thực tế

- Cài đặt chế độ MODE hoặc TUANAP.



Hình 44 Cài đặt nhiệt độ cho hệ thống

Sau khi cài đặt hệ thống sẽ vào trạng thái điều chỉnh nhiệt độ theo nhiệt độ đặt.



Hình 45 Buồng ấp trứng tự động STM32F103C8T6

3.7. Kết luận và hướng phát triển:

Nhóm đã hoàn thành yêu cầu từ đề án và rút ra kết luận chung. Hệ thống “ấp trứng tự động sử dụng vi điều khiển STM32F103C8T6 kết hợp bộ điều khiển PID” đạt được kết quả mong muốn và thỏa mãn các yêu cầu được đặt ra ban đầu. Sai số xác lập bé hơn 0.05 độ C, độ vọt lố bé hơn 0.3%, thời gian xác lập tùy thuộc vào nhiệt độ môi trường. Trong trường hợp nhiệt độ môi trường là 30 độ C, thì sau khi nhấn “OKE” thì thời gian để nhiệt độ đạt

trạng thái xác lập cho 3 tuần áp là khoảng dưới 1 phút. Qua đề tài lần này, mỗi thành viên trong nhóm học được nhiều kiến thức, nắm được các thao tác từ các ứng dụng lập trình, thiết kế phần cứng, lập trình cho board mạch và đặc biệt là hiểu thêm kiến thức về Vi điều khiển STM32F103C8T6.

Hướng phát triển đề tài:

- + Thực hiện đặt mạch in 2 lớp và hàn linh kiện.
- + Giao tiếp UART với PC để hiển thị biểu đồ nhiệt.
- + Thiết lập một số chế độ như WatchDog Timer tránh tình trạng nhiễu, gây hỗn loạn PID.
- + Thiết kế cơ khí, thực hiện áp trướng thực tế và thương mại hóa sản phẩm.

TÀI LIỆU THAM KHẢO

Sách tham khảo:

[1] MAX31865, Maxim Integrated and the Maxim Integrated logo are trademarks of Maxim Integrated Products, Inc.

[2] RM0008 Reference manual, PM0056, STM32F10xxx Cortex®-M3 programming manual.

Trang thông tin điện tử tham khảo:

[3] Electronoobs (2022), “220V AC HEATER PID TEMPERATURE CONTROL”, ELECTRONO OBS BLOGS, https://electronoobs.com/eng_arduino_tut39.php, truy cập ngày 31/05/2023.

[4] Nguyen Van Dong Hai (2021), “Dieu khien nhiet do”, Kênh Youtube Nguyen van dong hai, https://www.youtube.com/playlist?list=PLOTKVdGkFUunBxIkg9EgpGK_XEcb0fNYM, truy cập ngày 28/05/2023.