**LỜI NÓI ĐẦU**

Hiện nay, thời đại khoa học công nghệ ngày càng phát triền, thiết bị và công nghệ không ngừng đổi mới tiên tiến hiện đại góp phần nâng cao chất lượng máy móc thiết bị hoạt động hiệu quả năng suất. Trong đó vi điều khiển ngày càng ứng dụng rộng rãi trong tất cả các lĩnh vực, chỉ với một con chip vi mạch nhỏ có thể thực hiện nhiều chức năng với thao tác đơn giản dễ dàng có thể điều khiển các hệ thống phức tạp.

Trong nông nghiệp hiện nay, như ngành chăn nuôi gia cầm thì việc ứng dụng các hệ thống điều khiển tự động trong khâu sản suất là cần thiết , để ấp nở con giống từ trứng gia cầm cho nên thấy được tầm quan trọng của nó,chúng em đã lựa chọn đề tài “**MÁY ẤP TRỨNG**” và thực hiện hệ thống có thể điều khiển và ổn định nhiệt độ cho lò ấp trứng.

Hệ thống còn nhiều điểm chưa hoàn thiện nhưng với mong muốn học hỏi và sáng tạo chúng em hi vọng tương lai sẽ phát triển thêm nhiều chức năng cho hệ thống.

Qua bộ môn “**Kỹ thuật vi điều khiển**” chúng em đã học hỏi và hiểu biết thêm nhiều kiến thức về vi điều khiển STM32 và ứng dụng vào thực tế qua sản phầm này. Chúng em cảm ơn thầy “**Huỳnh Việt Thắng**” đã giảng dạy và giúp đỡ với bộ môn này đề bọn em hoàn thành đề tài.

**I. ĐẶT VẤN ĐỀ, LÍ DO CHỌN ĐỀ TÀI :**

Ngày nay, điều khiển tự động đã trở thành một nhu cầu không thể thiếu được trong tất cả các lĩnh vực cũng như ngành nông nghiệp, cũng như việc chăn nuôi gia cầm để tiết kiệm thời gian và công sức của con người trong với việc ấp trứng thủ công, một trong các yếu tố quan trọng là nhiệt độ, vì lí do tác động của nhiệt độ môi trường nên trứng gia cầm thường nở không đều và bị hỏng làm giảm năng suất chăn nuôi của người nông dân nên nhóm chúng em đã quyết định thực hiện đề tài máy ấp trứng sử dụng vi điều khiển STM32.

**II.CÁC SẢN PHẨM, GIẢI PHÁP, HỆ THỐNG ĐÃ CÓ:**

**1.CÁC SẢN PHẨM:**

- Kit STM32f411 Dicovery

- Cảm biến nhiệt độ DHT11

-Màn hình lcd 16x2

-Nguồn 220VAC -> 12VDC

- Quạt thông gió 220V/5W

- Đèn sợi đốt tạo nguồn nhiệt 220V/40W

- Động cơ đảo trứng 24V/4W

**2.GIẢI PHÁP HỆ THỐNG ĐÃ CÓ:**

- Hệ thống điều khiển và ổn định nhiện độ bằng phương pháp on/off, với ưu điểm đơn giản, dễ thi công

- Sử dụng cảm biến nhiệt độ độ ẩm DHT11 để lấy mẫu nhiệt độ trong lò ấp trứng hiển thị lên LCD từ đó ổn định nhiệt độ.

**III. MỤC TIÊU ĐỀ TÀI**

1. Hiểu rõ, nắm vững, và sử dụng được dòng vi điều khiển STM32
2. Nắm vững được tổ chức của vi điều khiển
3. Ứng dụng được dòng vi điều khiển STM32 vào thực tế thông qua đề tài.
4. Phát huy sự sáng tạo của bản thân kết hợp với các chức năng của vi điều khiển để tạo ra sản phẩm ứng dụng vào các lĩnh vực trong cuộc sống.

**IV. ĐẶC TÍNH KỸ THUẬT, CHỨC NĂNG SẢN PHẨM:**

**1.ĐẶC TÍNH KỸ THUẬT:**

- Lò ấp sử dụng nguồn điện 220 VAC

- Nhiệt độ là ấp ổn định ở mức 36-37 0C (mặc định)

- Sử dụng vi điều khiển STM32F411

- Đảo trứng sau mỗi 30 phút

**2. CHỨC NĂNG SẢN PHẨM:**

- Lò ấp trứng sử dụng phương pháp điều khiển và ổn định nhiệt độ theo phương pháp on/off ổn định chính xác nhiệt độ lò ấp ở khoảng 36-37 0C trong quá trình ấp, từ đó làm tăng tỉ lệ nở của trứng gia cầm.

- Chức năng đảo trứng sau mỗi 30 phút.

- Điều chỉnh khoảng nhiệt độ của lò ấp theo từng loại trứng.

- Hiển thị nhiệt độ độ ẩm của lò ấp tại thời điểm hiện tại.

**IV. SƠ ĐỒ KHỐI HỆ THỐNG**

KHỐI NGUỒN

KHỐI XỬ LÝ TRUNG TÂM

(STM32F411 DISCOVERY)

KHỐI NÚT NHẤN

KHỐI ĐIỀU KHIỂN NGUỒN NHIỆT

KHỐI ĐIỀU KHIỂN QUẠT THÔNG GIÓ

KHỐI ĐIỀU KHIỂN ĐỘNG CƠ ĐẢO TRỨNG

KHỐI CẢM BIẾN

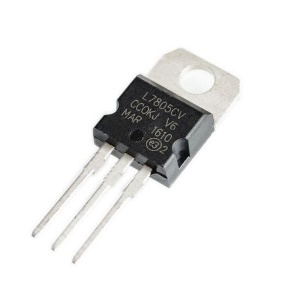
KHỐI LCD HIỂN THỊ

**V. THIẾT KẾ PHẤN CỨNG**

1. **SƠ ĐỒ MẠCH CÁC KHỐI:**

**1.1KHỐI NGUỒN:**



-Chọn module nguồn HI-LINK 220VAC – 12VDC/0.25A

-IC LM7805 ổn định áp ngõ ra 5VDC/0.25A cung cấp cho khối điều khiển, khối xử lý, khối hiển thị và khối nút nhấn.

- Chọn tụ lọc C3,C4 1000uF/25V để bù áp khi có hiện tượng sụt áp.

-Tụ C5,C6 lọc các tần số cao, các nhiễu cao tần.

**1.2-KHỐI XỬ LÝ TRUNG TÂM**

- Sử dụng Kit STM32F411 Discovery



**1.Giới thiệu Vi điều khiển STM32F411 Discovery:**

* Lõi: ARM® 32bit Cortex®-M4
* Nhiệt độ hoạt động : -40ºC – 85ºC
* Bộ nhớ: Bộ nhớ Flash 512KB , RAM 128KB trong gói LQFP100
* Tốc độ : 100MHz
* Nguồn cung cấp cho board : thông qua bus USB hoặc từ nguồn 5V bên ngoài.
* Nguồn cung cấp cho ngoại vi: 3V và 5V
* 8 đèn LED: LD1( đỏ/xanh) giao tiếp USB ; LD2 (đỏ) tín hiệu nguồn 3,3V ; 4 LED dành cho người dùng LD3(cam), LD4(xanh) ,LD5(đỏ) ,LD6(xanh dương)

2 đèn LED USB OTG: LD7(xanh) VBus và LD8( đỏ) quá dòng

* 2 nút nhấn (người dùng và reset)
* Bộ điều khiển DMA đa mục đích với bộ điều khiển DMA 16 luồng FIFO và hỗ trợ liên tục.
* 11 bộ timer: 6 bộ timer 16 bit , 2 bộ định thời 32bit tần số 100MHz với mỗi bộ có tối đa 4 IC/OC/PWM hoặc bộ đếm xung 2 bộ watchdog

timers and một bộ ngắt System Tick

* 81 cổng I/0, 16 line ngắt

- Các giao diện giao tiếp: I2C, USART, SPI/I2S, STDIO ,ISO.

**2. Các chức năng sử dụng:**

+ Các chân GPIO sử dụng:

* - PA3 : chân GPIO output điều khiển động cơ đảo trứng
* - PA5: chân GPIO output điều khiển quạt thông gió hút ra bên ngoài.
* - PA6 : chân GPIO output để đọc dữ liệu cảm biến DHT11
* - PA7:chân GPIO output điều khiển nguồn nhiệt bằng phương pháp on/off
* - PE8 : chân GPIO điều khiển nút nhấn SW1
* - PE10 : điều khiển khiển nút nhấn SW2
* - PE12 : điều khiển nút nhấn SW3
* - PE14 : điều khiển nút nhấn SW4
* - PC9: chân SDA (chân dữ liệu kết) nối module I2C làm trung gian giao tiếp VĐK và LCD.
* - PA8 : chân SCL (chân xung nhip đồng hồ) kết nối module I2C làm trung gian giao tiếp VĐK và LCD

+ Các bộ timer:

* - TIM3 : bộ đếm hẹn giờ đảo động cơ trứng.
* - TIM4: sử dụng cho hàm Delay điều khiển động cơ.
  1. **KHỐI ĐIỀU KHIỂN:**

-Khối điều khiển: Điều khiển nguồn nhiệt, điều khiển quạt thông gió.



-Từ tải đèn sợi đốt có công suất 220VAC/40W

-> dòng qua triac : I =

Chọn triac có Igh = 2I=2\*0.182 = 0.364

->chọn triac BTA16 có:

IT(RMS)= 16A

VDRM max = 400VAC

VGT max = 2V

VGT min = 0.2V

IGT MAX = 50mA

-Tính toán R7=R11 để kích đẫn triac:

Gỉả sử thời điểm kích dẫn triac nhỏ nhất là 0.2ms và lớn nhất là 10ms trong một nữa chu kì

Điện áp rơi trên VR7 = Vsin – Vtải – VGT

Xét thời điểmgóc kích nhỏ nhất (ứng với thời gian t = 0.2ms )

Vsin = 220220

Giả sử Vtải = 17V

Xét thời điểm t= 10(ms) thì điện áp Vsin có giá trị lớn nhất:

Vsin = 220220

Giả sử Vtải = 250V

Suy ra:

-> Chọn R7 = R11 = 1k

IGT max = 50mA

P = R\*=103 \*(50\*10-3)2 = 2.5w

-> Chọn trở 1k/2.5W

-Chọn opto MOC 3021 có các thông số:

Vin =1 - 3VDC

Iin = 10 - 20 mA

Vout max = 400 V

-Chọn Q2, Q3 là BJT C1815 có

Igh =

Để BJT dẫn bảo hòa thì Ib = 3Igh = 3\* 0.14 = 0.42 mA

BJT dẫn bảo hòa nên Vce = 0.2 V

Ta có: VR4 = 5v – VLED – Vin MOC3021 - Vce = 5 - 1.8 – 1 – 0.2 = 2V

Chọn dòng qua led là I LED =10 mA

R4 =

Chọn trở R4 = R10 = 220

Ta có : VR9 = VPA5 – Vbe = 4 – 0.7 = .07 V

-> R9 =

-> chọn R9 = R12=8.6K

**1.4. KHỐI ĐIỀU KHIỂN ĐỘNG CƠ ĐẢO TRỨNG:**



-Tải: Động cơ

Nguồn cấp: 12V-24V  
Tốc độ không tải: 120 RPM

Tính toán:

Từ tải ta chọn Relay 5V chịu tải 250V/10A

Để kích dẫn relay IRLmin = 5mA

Chọn IRL = 15mA

Chọn opto PC817 có thông số:

VoutPC817/dẫn = 0.2V

VinPC817 = 1.5V

BJT Q1 là C1815 có = 70

Igh/Q1 = = = 0.214 mA

Để BJT Q1 dẫn bảo hòa thì:

Ib/Q1 = 2 Igh/Q1 = 2\*0.214 = 0.428 mA

Ta có: VR2 = 5V - VoutPC817/dẫn –Vbe/Q1 = 5 – 0.2 – 0.7 = 4.1V

-> R2 = = = 9.5K

Chọn R2 = 8.6k

**1.5.KHỐI CẢM BIẾN**



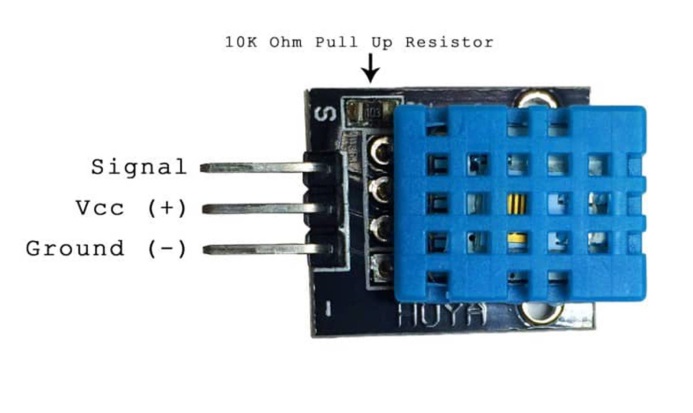
Thông số kỹ thuật:

+ Điện áp hoạt động : 3.3V-5V

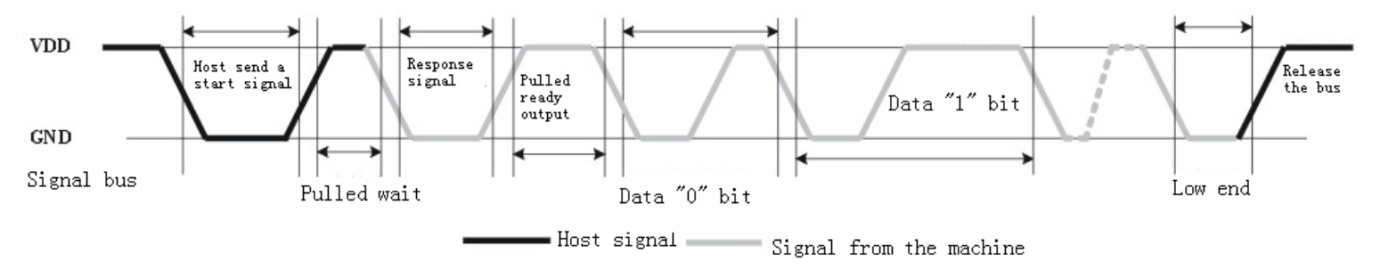
+ Nhiệt độ hoạt động: 0ºC – 50ºC, sai số ± 2ºC

+ Độ ẩm hoạt ẩm: 20%-90% RH, sai số ± 5%RH

+ Tần số lấy mẫu: 1Hz (mỗi giây một lần).



**Nguyên lý hoạt động:**



*Dạng tín hiệu giao tiếp cảm biến DHT11*

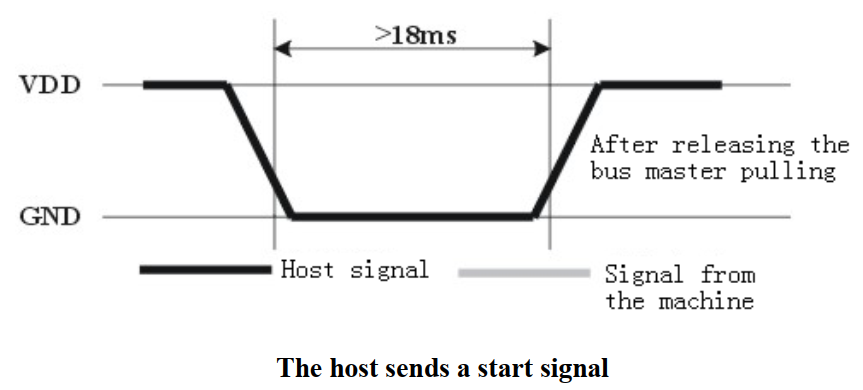
Để có thể giao tiếp với DHT11 theo chuẩn 1 chân vi xử lý thực hiện theo 2 bước:

+ Gửi tin hiệu muốn đo (Start) tới DHT11, sau đó DHT11 xác nhận lại.

+ Khi đã giao tiếp được với DHT11, Cảm biến sẽ gửi lại 5 byte dữ liệu và nhiệt độ độ ẩm đo được.

– Bước 1: Gửi tín hiệu Start

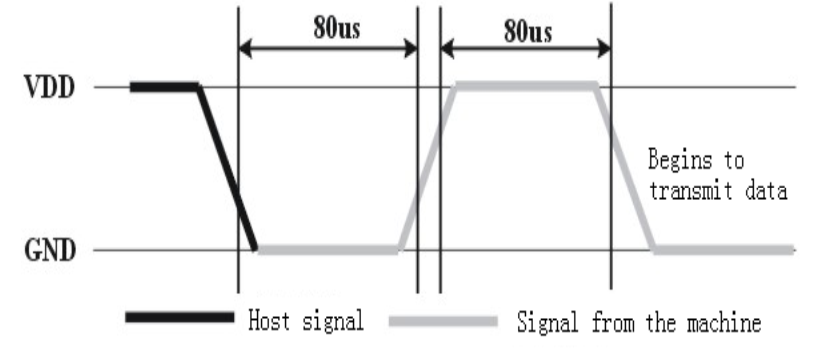
Vi điều khiển thiết lập chân DATA là GPIO Output PA6, kéo chân DATA xuống 0 trong khoảng thời gian 18ms.Khi đó DHT11 sẽ hiểu VĐK muốn đo giá trị nhiệt độ và độ ẩm.



+ VĐK đưa chân DATA lên 1, sau đó thiết lập lại là chân đầu vào.

+ DHT11 sẽ kéo chân DATA xuống thấp.

Chân DATA sẽ ở mức thấp 80us sau đó được DHT11 kéo lên cao trong 80us. Bằng việc giám sát chân DATA, VĐK có thể biết được có giao tiếp được với DHT11 không. Nếu tín hiệu đo được DHT11 lên cao, khi đó hoàn thiện quá trình giao tiếp của VĐK với cảm biến DHT11.



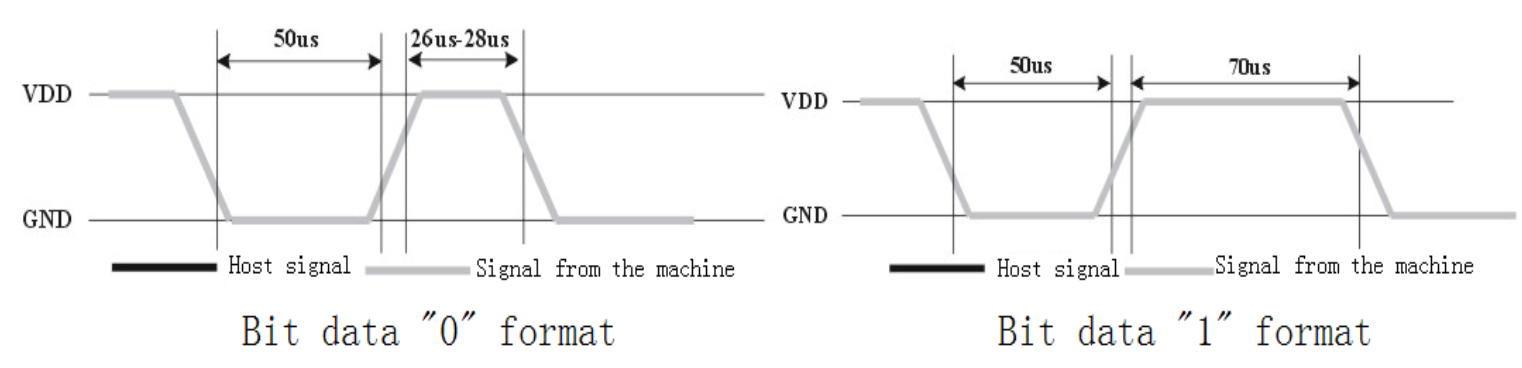
– Bước 2: Đọc giá trị trên DHT11

+ DHT11 sẽ trả giá trị nhiệt độ và độ ẩm về dạng dữ liệu 5 byte 40bits theo thứ tự .

* Byte 1: 8 bit biểu thị phần nguyên của độ ẩm (RH%)
* Byte 2: 8 bit biểu thị phần thập phân của độ ẩm (RH%)
* Byte 3: 8 bit biểu thị phần nguyên của nhiệt độ (TºC)
* Byte 4 : 8 bit biểu thị phần thập phân của nhiệt độ (TºC)
* Byte 5 : 8 bit check sum= (Byte 1+ Byte 2 +Byte3 +Byte4) thì giá trị đo là chính xác, nếu sai thì kết quả đo không có nghĩa.

+ Đọc dữ liệu:

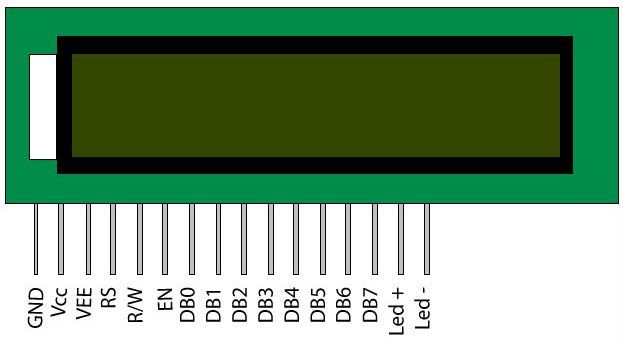
Sau khi giao tiếp được với DHT11, DHT11 sẽ gửi liên tiếp 40 bit 0 hoặc 1 về VĐK, tương ứng chia thành 5 byte kết quả của nhiệt độ và độ ẩm.



**1.6. KHỐI HIỂN THỊ:**

1.6.1. SƠ ĐỒ NGUYÊN LÝ:

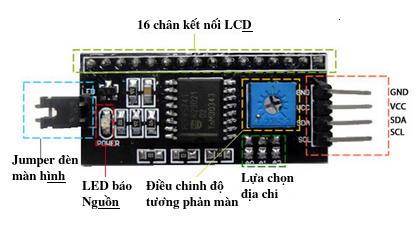


* Màn hình LCD 16x2:
* 



* Mạch chuyển đổi I2C cho LCD:

Hỗ trợ 2 chuẩn tốc độ là 100khz và 400 Khz.



* Điện áp hoạt động: 2.5-6V DC
* Hỗ trợ màn hình: LCD1602,1604,2004 (driver HD44780)
* Giao tiếp: I2C

*Sơ đồ đấu nối giao tiếp I2C và LCD 16x2*



**1.7. KHỐI NÚT NHẤN:**



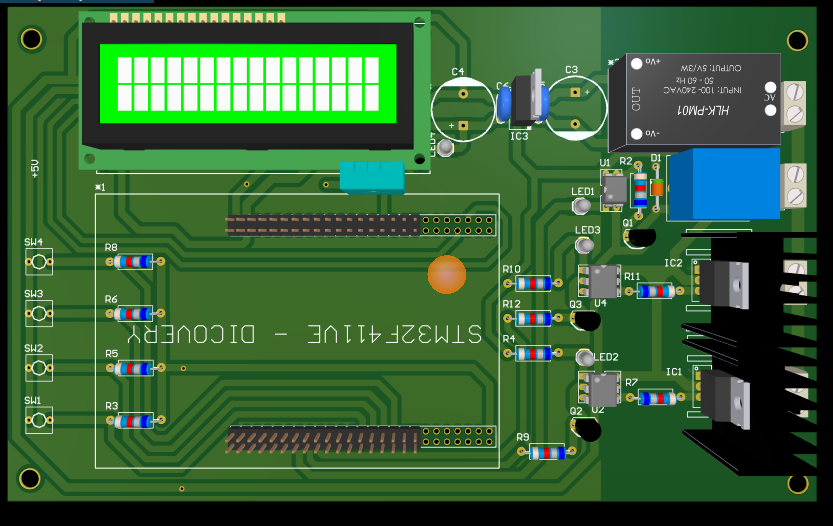
- Chức năng của các nút:

+ Nút nhấn SW1 và SW2: cài đặt mức trên và mức dưới cho nhiệt độ

+Nút nhấn SW3: Mode nhấn nút để đến màn hình cài đặt và quay về màn hình ban đầu hiển thị nhiệt độ độ ẩm.

+ Nút nhấn SW4: hiển thị màn hình cài đặt chế độ cho nhiệt độ

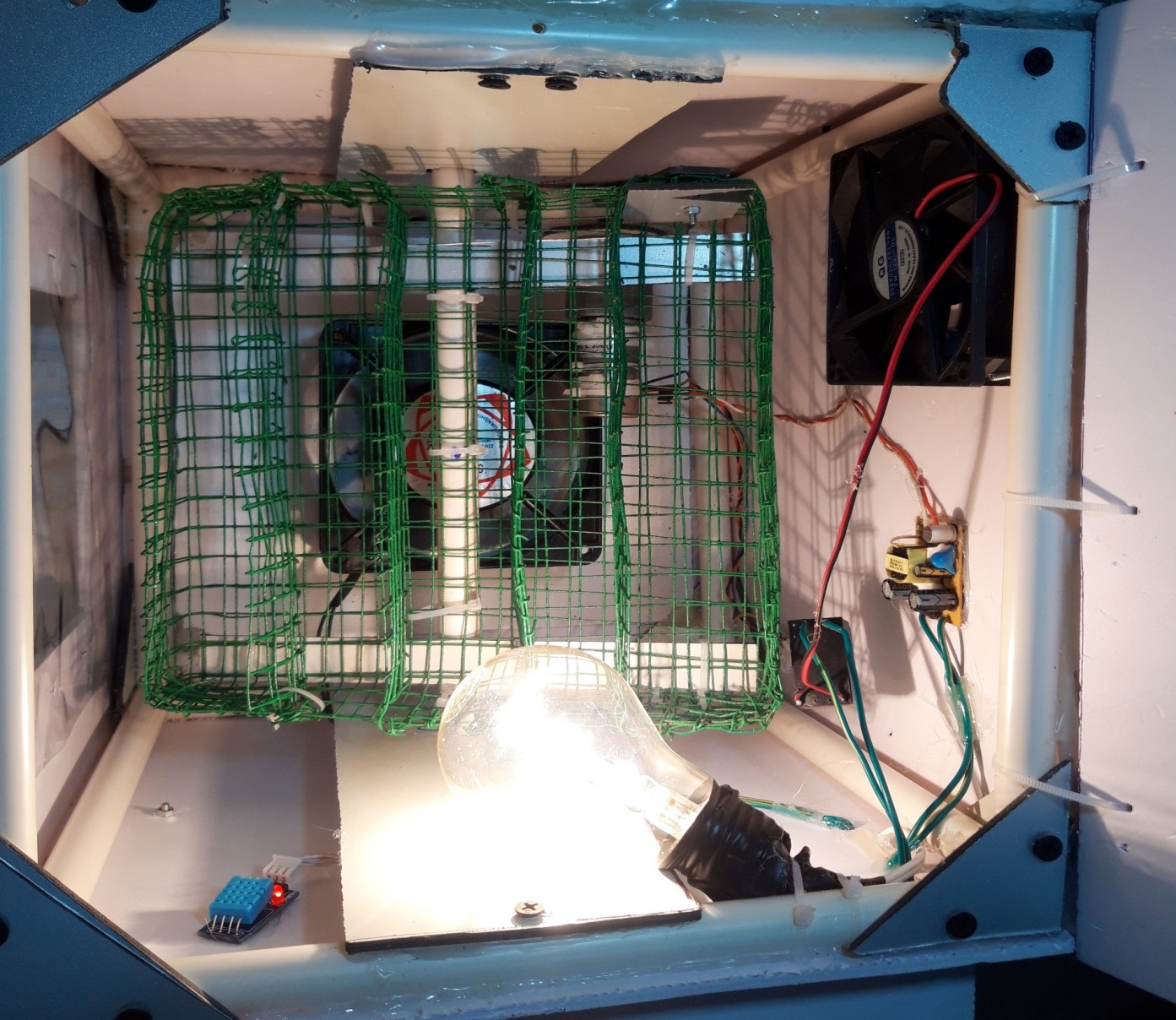
**1.8. MẠCH 3D:**



**1.9 THI CÔNG MẠCH:**



*Sơ đồ mạch*



*Mô hình máy ấp trứng*

**V.THIẾT KẾ PHẦN MỀM:**

* 1. **LƯU ĐỒ THUẬT TOÁN:**

DEM = DEM +1

ĐIỀU KHIỂN ĐỘNG CƠ ĐẢO TRỨNG(3S)

DEM >=5

NGẮT TIMER 10S

ĐIỀU KHIỂN NGUỒN NHIỆT, QUẠT THÔNG GIÓ ON

TEMP <T1

ĐIỀU KHIỂN NGUỒN NHIỆT, QUẠT THÔNG GIÓ OFF

TEMP >T2

HIỂN THỊ NHIỆT ĐỘ ĐỘ ẨM LÊN LCD

ĐỌC CẢM BIẾN NHIỆT ĐỘ, ĐỘ ẨM

NGẮT TỪ NÚT NHẤN

THỰC THI HÀM NGẮT

KHỞI TẠO CÁC BIẾN, TIMER, IO,TẦN SỐ DAO ĐỘNG.

Đ

S

Đ

S

Đ

**2.CHƯƠNG TRÌNH:**

/\* USER CODE END Header \*/

/\* Includes ------------------------------------------------------------------\*/

**#include** "main.h"

/\* Private includes ----------------------------------------------------------\*/

/\* USER CODE BEGIN Includes \*/

**#include** "DHT.h"

**#include** "stdio.h"

**#include** "i2c-lcd.h"

/\* USER CODE END Includes \*/

/\* Private typedef -----------------------------------------------------------\*/

/\* USER CODE BEGIN PTD \*/

/\* USER CODE END PTD \*/

/\* Private define ------------------------------------------------------------\*/

/\* USER CODE BEGIN PD \*/

/\* USER CODE END PD \*/

/\* Private macro -------------------------------------------------------------\*/

/\* USER CODE BEGIN PM \*/

/\* USER CODE END PM \*/

/\* Private variables ---------------------------------------------------------\*/

I2C\_HandleTypeDef hi2c3;

TIM\_HandleTypeDef htim3;

TIM\_HandleTypeDef htim4;

/\* USER CODE BEGIN PV \*/

/\* USER CODE END PV \*/

/\* Private function prototypes -----------------------------------------------\*/

**void** **SystemClock\_Config**(**void**);

**static** **void** **MX\_GPIO\_Init**(**void**);

**static** **void** **MX\_TIM3\_Init**(**void**);

**static** **void** **MX\_TIM4\_Init**(**void**);

**static** **void** **MX\_I2C3\_Init**(**void**);

/\* USER CODE BEGIN PFP \*/

/\* USER CODE END PFP \*/

/\* Private user code ---------------------------------------------------------\*/

/\* USER CODE BEGIN 0 \*/

**float** T1 = 36;

**float** T2= 37;

**char** t1 [20]={0};

**char** t2 [20]={0};

**char** Temp [20]= {0};

**char** tempa [20]= {0};

**char** humi [20]={0};

DHT\_DataTypedef DHT11\_Data;

**float** Temperature, Humidity;

**int** COUNT =0;

**int** set = 0;

**int** menu = 0;

**int** up = 0;

**int** down = 0;

**int** set\_temp;

**void** **delay**(uint16\_t time)

{

\_\_HAL\_TIM\_SET\_COUNTER(&htim4,0);

**while**(\_\_HAL\_TIM\_GET\_COUNTER(&htim4) < time);

}

**void** **HAL\_GPIO\_EXTI\_Callback**(uint16\_t GPIO\_Pin)

{

**if**(HAL\_GPIO\_ReadPin(NUT1\_GPIO\_Port, NUT1\_Pin) == 1)

{

**if**(set == 1)

{

T1 = T1 + 1;

}

**if**(set == 2)

{

T2 = T2 + 1;

}

}

**if**(HAL\_GPIO\_ReadPin(NUT2\_GPIO\_Port, NUT2\_Pin) == 1)

{

**if**(set == 1)

{

T1 = T1- 1;

}

**if**(set == 2)

{

T2 = T2- 1;

}

}

**if**(HAL\_GPIO\_ReadPin(NUT3\_GPIO\_Port, NUT3\_Pin) == 1)

{

menu =! menu;

}

**if**(HAL\_GPIO\_ReadPin(NUT4\_GPIO\_Port, NUT4\_Pin) == 1)

{

**if**(menu == 1)

{

set = set + 1;

**if**(set >= 3)

{

set = 0;

}

}

}

}

**void** **DAO\_TRUNG**()

{

**if**(COUNT >= 15 )

{

HAL\_GPIO\_WritePin(LED2\_GPIO\_Port, LED2\_Pin, 1);

HAL\_GPIO\_WritePin(DC\_DAO\_GPIO\_Port, DC\_DAO\_Pin, 1);

delay(2500);

HAL\_GPIO\_WritePin(LED2\_GPIO\_Port, LED2\_Pin, 0);

HAL\_GPIO\_WritePin(DC\_DAO\_GPIO\_Port, DC\_DAO\_Pin, 0);

COUNT = 0;

}

}

**void** **HAL\_TIM\_PeriodElapsedCallback**(TIM\_HandleTypeDef \*htim)

{

**if**(htim -> Instance == htim3.Instance)

{

COUNT = COUNT + 1;

}

}

**void** **menu\_tab**()

{

**if**(menu == 1 )

{

lcd\_clear\_display();

lcd\_goto\_XY(2, 3);

lcd\_send\_string("SET TEMP");

}

**if**(menu == 0)

{

lcd\_clear\_display();

set = 0;

up = 0;

down = 0;

set\_temp = 0;

lcd\_clear\_display();

lcd\_goto\_XY(1, 2);

lcd\_send\_string("TEMP:");

lcd\_goto\_XY(1, 7);

lcd\_send\_string(tempa);

lcd\_goto\_XY(1, 13);

lcd\_send\_string("C");

lcd\_goto\_XY(2, 0);

lcd\_send\_string("HUMI:");

lcd\_goto\_XY(2, 7);

lcd\_send\_string(humi);

lcd\_goto\_XY(2, 13);

lcd\_send\_string("%");

}

**if**(menu == 1 && set == 1 )

{

lcd\_clear\_display();

lcd\_goto\_XY(1, 0);

lcd\_send\_string("T up ");

lcd\_goto\_XY(2, 0);

**sprintf**(t1,"%.2f",T1);

lcd\_send\_string(t1);

lcd\_goto\_XY(1, 10);

lcd\_send\_string("T down ");

lcd\_goto\_XY(2, 10);

**sprintf**(t2,"%.2f",T2);

lcd\_send\_string(t2);

}

**if**(menu == 1 && set == 2 )

{

lcd\_clear\_display();

lcd\_goto\_XY(1, 1);

lcd\_send\_string("T up ");

lcd\_goto\_XY(2, 0);

**sprintf**(t1,"%.2f",T1);

lcd\_send\_string(t1);

lcd\_goto\_XY(1, 10);

lcd\_send\_string("T down ");

lcd\_goto\_XY(2, 10);

**sprintf**(t2,"%.2f",T2);

lcd\_send\_string(t2);

}

}

**void** **Sensor**()

{

DHT\_GetData(&DHT11\_Data);

Temperature = DHT11\_Data.Temperature;

Humidity = DHT11\_Data.Humidity;

delay(1500);

**sprintf**(tempa,"%.2f",Temperature);

**sprintf**(humi,"%.2f",Humidity);

**if**( Temperature > T1 && Temperature < T2 ||Temperature > T2 )

{

HAL\_GPIO\_WritePin(DEN\_GPIO\_Port, DEN\_Pin, 0);

HAL\_GPIO\_WritePin(QUAT\_GPIO\_Port, QUAT\_Pin, 1);

}

**if**(Temperature < T1 )

{

HAL\_GPIO\_WritePin(DEN\_GPIO\_Port, DEN\_Pin, 1);

HAL\_GPIO\_WritePin(QUAT\_GPIO\_Port, QUAT\_Pin, 0);

}

}

**void** **xung**()

{

HAL\_GPIO\_WritePin(led\_GPIO\_Port, led\_Pin, 1);

HAL\_Delay(50);

HAL\_GPIO\_WritePin(led\_GPIO\_Port, led\_Pin, 0);

HAL\_Delay(50);

}

/\* USER CODE END 0 \*/

/\*\*

\* @brief The application entry point.

\* @retval int

\*/

**int** **main**(**void**)

{

/\* USER CODE BEGIN 1 \*/

/\* USER CODE END 1 \*/

/\* MCU Configuration--------------------------------------------------------\*/

/\* Reset of all peripherals, Initializes the Flash interface and the Systick. \*/

HAL\_Init();

/\* USER CODE BEGIN Init \*/

/\* USER CODE END Init \*/

/\* Configure the system clock \*/

SystemClock\_Config();

/\* USER CODE BEGIN SysInit \*/

/\* USER CODE END SysInit \*/

/\* Initialize all configured peripherals \*/

MX\_GPIO\_Init();

MX\_TIM3\_Init();

MX\_TIM4\_Init();

MX\_I2C3\_Init();

/\* USER CODE BEGIN 2 \*/

HAL\_TIM\_Base\_Start\_IT(&htim3);

HAL\_TIM\_Base\_Start(&htim4);

lcd\_init();

lcd\_goto\_XY(1, 0);

lcd\_send\_string("MAY AP TRUNG ");

lcd\_goto\_XY(2, 4);

lcd\_send\_string("NHOM 5");

HAL\_Delay(1000);

lcd\_clear\_display();

/\* USER CODE END 2 \*/

/\* Infinite loop \*/

/\* USER CODE BEGIN WHILE \*/

**while** (1)

{

Sensor();

menu\_tab();

DAO\_TRUNG();

xung();

/\* USER CODE END WHILE \*/

/\* USER CODE BEGIN 3 \*/

}

/\* USER CODE END 3 \*/

}

/\*\*

\* @brief System Clock Configuration

\* @retval None

\*/

**void** **SystemClock\_Config**(**void**)

{

RCC\_OscInitTypeDef RCC\_OscInitStruct = {0};

RCC\_ClkInitTypeDef RCC\_ClkInitStruct = {0};

/\*\* Configure the main internal regulator output voltage

\*/

\_\_HAL\_RCC\_PWR\_CLK\_ENABLE();

\_\_HAL\_PWR\_VOLTAGESCALING\_CONFIG(PWR\_REGULATOR\_VOLTAGE\_SCALE1);

/\*\* Initializes the RCC Oscillators according to the specified parameters

\* in the RCC\_OscInitTypeDef structure.

\*/

RCC\_OscInitStruct.OscillatorType = RCC\_OSCILLATORTYPE\_HSE;

RCC\_OscInitStruct.HSEState = RCC\_HSE\_ON;

RCC\_OscInitStruct.PLL.PLLState = RCC\_PLL\_ON;

RCC\_OscInitStruct.PLL.PLLSource = RCC\_PLLSOURCE\_HSE;

RCC\_OscInitStruct.PLL.PLLM = 4;

RCC\_OscInitStruct.PLL.PLLN = 50;

RCC\_OscInitStruct.PLL.PLLP = RCC\_PLLP\_DIV2;

RCC\_OscInitStruct.PLL.PLLQ = 4;

**if** (HAL\_RCC\_OscConfig(&RCC\_OscInitStruct) != *HAL\_OK*)

{

Error\_Handler();

}

/\*\* Initializes the CPU, AHB and APB buses clocks

\*/

RCC\_ClkInitStruct.ClockType = RCC\_CLOCKTYPE\_HCLK|RCC\_CLOCKTYPE\_SYSCLK

|RCC\_CLOCKTYPE\_PCLK1|RCC\_CLOCKTYPE\_PCLK2;

RCC\_ClkInitStruct.SYSCLKSource = RCC\_SYSCLKSOURCE\_PLLCLK;

RCC\_ClkInitStruct.AHBCLKDivider = RCC\_SYSCLK\_DIV1;

RCC\_ClkInitStruct.APB1CLKDivider = RCC\_HCLK\_DIV1;

RCC\_ClkInitStruct.APB2CLKDivider = RCC\_HCLK\_DIV1;

**if** (HAL\_RCC\_ClockConfig(&RCC\_ClkInitStruct, FLASH\_LATENCY\_1) != *HAL\_OK*)

{

Error\_Handler();

}

}

/\*\*

\* @brief I2C3 Initialization Function

\* @param None

\* @retval None

\*/

**static** **void** **MX\_I2C3\_Init**(**void**)

{

/\* USER CODE BEGIN I2C3\_Init 0 \*/

/\* USER CODE END I2C3\_Init 0 \*/

/\* USER CODE BEGIN I2C3\_Init 1 \*/

/\* USER CODE END I2C3\_Init 1 \*/

hi2c3.Instance = I2C3;

hi2c3.Init.ClockSpeed = 100000;

hi2c3.Init.DutyCycle = I2C\_DUTYCYCLE\_2;

hi2c3.Init.OwnAddress1 = 0;

hi2c3.Init.AddressingMode = I2C\_ADDRESSINGMODE\_7BIT;

hi2c3.Init.DualAddressMode = I2C\_DUALADDRESS\_DISABLE;

hi2c3.Init.OwnAddress2 = 0;

hi2c3.Init.GeneralCallMode = I2C\_GENERALCALL\_DISABLE;

hi2c3.Init.NoStretchMode = I2C\_NOSTRETCH\_DISABLE;

**if** (HAL\_I2C\_Init(&hi2c3) != *HAL\_OK*)

{

Error\_Handler();

}

/\* USER CODE BEGIN I2C3\_Init 2 \*/

/\* USER CODE END I2C3\_Init 2 \*/

}

/\*\*

\* @brief TIM3 Initialization Function

\* @param None

\* @retval None

\*/

**static** **void** **MX\_TIM3\_Init**(**void**)

{

/\* USER CODE BEGIN TIM3\_Init 0 \*/

/\* USER CODE END TIM3\_Init 0 \*/

TIM\_ClockConfigTypeDef sClockSourceConfig = {0};

TIM\_MasterConfigTypeDef sMasterConfig = {0};

/\* USER CODE BEGIN TIM3\_Init 1 \*/

/\* USER CODE END TIM3\_Init 1 \*/

htim3.Instance = TIM3;

htim3.Init.Prescaler = 25000-1;

htim3.Init.CounterMode = TIM\_COUNTERMODE\_UP;

htim3.Init.Period = 10000-1;

htim3.Init.ClockDivision = TIM\_CLOCKDIVISION\_DIV1;

htim3.Init.AutoReloadPreload = TIM\_AUTORELOAD\_PRELOAD\_DISABLE;

**if** (HAL\_TIM\_Base\_Init(&htim3) != *HAL\_OK*)

{

Error\_Handler();

}

sClockSourceConfig.ClockSource = TIM\_CLOCKSOURCE\_INTERNAL;

**if** (HAL\_TIM\_ConfigClockSource(&htim3, &sClockSourceConfig) != *HAL\_OK*)

{

Error\_Handler();

}

sMasterConfig.MasterOutputTrigger = TIM\_TRGO\_RESET;

sMasterConfig.MasterSlaveMode = TIM\_MASTERSLAVEMODE\_DISABLE;

**if** (HAL\_TIMEx\_MasterConfigSynchronization(&htim3, &sMasterConfig) != *HAL\_OK*)

{

Error\_Handler();

}

/\* USER CODE BEGIN TIM3\_Init 2 \*/

/\* USER CODE END TIM3\_Init 2 \*/

}

/\*\*

\* @brief TIM4 Initialization Function

\* @param None

\* @retval None

\*/

**static** **void** **MX\_TIM4\_Init**(**void**)

{

/\* USER CODE BEGIN TIM4\_Init 0 \*/

/\* USER CODE END TIM4\_Init 0 \*/

TIM\_ClockConfigTypeDef sClockSourceConfig = {0};

TIM\_MasterConfigTypeDef sMasterConfig = {0};

/\* USER CODE BEGIN TIM4\_Init 1 \*/

/\* USER CODE END TIM4\_Init 1 \*/

htim4.Instance = TIM4;

htim4.Init.Prescaler = 50000-1;

htim4.Init.CounterMode = TIM\_COUNTERMODE\_UP;

htim4.Init.Period = 0xffff-1;

htim4.Init.ClockDivision = TIM\_CLOCKDIVISION\_DIV1;

htim4.Init.AutoReloadPreload = TIM\_AUTORELOAD\_PRELOAD\_DISABLE;

**if** (HAL\_TIM\_Base\_Init(&htim4) != *HAL\_OK*)

{

Error\_Handler();

}

sClockSourceConfig.ClockSource = TIM\_CLOCKSOURCE\_INTERNAL;

**if** (HAL\_TIM\_ConfigClockSource(&htim4, &sClockSourceConfig) != *HAL\_OK*)

{

Error\_Handler();

}

sMasterConfig.MasterOutputTrigger = TIM\_TRGO\_RESET;

sMasterConfig.MasterSlaveMode = TIM\_MASTERSLAVEMODE\_DISABLE;

**if** (HAL\_TIMEx\_MasterConfigSynchronization(&htim4, &sMasterConfig) != *HAL\_OK*)

{

Error\_Handler();

}

/\* USER CODE BEGIN TIM4\_Init 2 \*/

/\* USER CODE END TIM4\_Init 2 \*/

}

/\*\*

\* @brief GPIO Initialization Function

\* @param None

\* @retval None

\*/

**static** **void** **MX\_GPIO\_Init**(**void**)

{

GPIO\_InitTypeDef GPIO\_InitStruct = {0};

/\* GPIO Ports Clock Enable \*/

\_\_HAL\_RCC\_GPIOH\_CLK\_ENABLE();

\_\_HAL\_RCC\_GPIOA\_CLK\_ENABLE();

\_\_HAL\_RCC\_GPIOE\_CLK\_ENABLE();

\_\_HAL\_RCC\_GPIOD\_CLK\_ENABLE();

\_\_HAL\_RCC\_GPIOC\_CLK\_ENABLE();

/\*Configure GPIO pin Output Level \*/

HAL\_GPIO\_WritePin(GPIOA, DC\_DAO\_Pin|QUAT\_Pin|DEN\_Pin, *GPIO\_PIN\_RESET*);

/\*Configure GPIO pin Output Level \*/

HAL\_GPIO\_WritePin(GPIOD, led\_Pin|LED2\_Pin, *GPIO\_PIN\_RESET*);

/\*Configure GPIO pins : DC\_DAO\_Pin QUAT\_Pin DEN\_Pin \*/

GPIO\_InitStruct.Pin = DC\_DAO\_Pin|QUAT\_Pin|DEN\_Pin;

GPIO\_InitStruct.Mode = GPIO\_MODE\_OUTPUT\_PP;

GPIO\_InitStruct.Pull = GPIO\_NOPULL;

GPIO\_InitStruct.Speed = GPIO\_SPEED\_FREQ\_LOW;

HAL\_GPIO\_Init(GPIOA, &GPIO\_InitStruct);

/\*Configure GPIO pins : NUT4\_Pin NUT3\_Pin NUT2\_Pin NUT1\_Pin \*/

GPIO\_InitStruct.Pin = NUT4\_Pin|NUT3\_Pin|NUT2\_Pin|NUT1\_Pin;

GPIO\_InitStruct.Mode = GPIO\_MODE\_IT\_RISING;

GPIO\_InitStruct.Pull = GPIO\_NOPULL;

HAL\_GPIO\_Init(GPIOE, &GPIO\_InitStruct);

/\*Configure GPIO pins : led\_Pin LED2\_Pin \*/

GPIO\_InitStruct.Pin = led\_Pin|LED2\_Pin;

GPIO\_InitStruct.Mode = GPIO\_MODE\_OUTPUT\_PP;

GPIO\_InitStruct.Pull = GPIO\_NOPULL;

GPIO\_InitStruct.Speed = GPIO\_SPEED\_FREQ\_LOW;

HAL\_GPIO\_Init(GPIOD, &GPIO\_InitStruct);

/\*Configure GPIO pin : PD6 \*/

GPIO\_InitStruct.Pin = GPIO\_PIN\_6;

GPIO\_InitStruct.Mode = GPIO\_MODE\_INPUT;

GPIO\_InitStruct.Pull = GPIO\_NOPULL;

HAL\_GPIO\_Init(GPIOD, &GPIO\_InitStruct);

/\* EXTI interrupt init\*/

HAL\_NVIC\_SetPriority(*EXTI9\_5\_IRQn*, 2, 0);

HAL\_NVIC\_EnableIRQ(*EXTI9\_5\_IRQn*);

HAL\_NVIC\_SetPriority(*EXTI15\_10\_IRQn*, 1, 0);

HAL\_NVIC\_EnableIRQ(*EXTI15\_10\_IRQn*);

}

/\* USER CODE BEGIN 4 \*/

/\* USER CODE END 4 \*/

/\*\*

\* @brief This function is executed in case of error occurrence.

\* @retval None

\*/

**void** **Error\_Handler**(**void**)

{

/\* USER CODE BEGIN Error\_Handler\_Debug \*/

/\* User can add his own implementation to report the HAL error return state \*/

/\* USER CODE END Error\_Handler\_Debug \*/

}

**#ifdef** USE\_FULL\_ASSERT

/\*\*

\* @brief Reports the name of the source file and the source line number

\* where the assert\_param error has occurred.

\* @param file: pointer to the source file name

\* @param line: assert\_param error line source number

\* @retval None

\*/

**void** assert\_failed(uint8\_t \*file, uint32\_t line)

{

/\* USER CODE BEGIN 6 \*/

/\* User can add his own implementation to report the file name and line number,

tex: printf("Wrong parameters value: file %s on line %d\r\n", file, line) \*/

/\* USER CODE END 6 \*/

}

**#endif** /\* USE\_FULL\_ASSERT \*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* (C) COPYRIGHT STMicroelectronics \*\*\*\*\*END OF FILE\*\*\*\*/

**VI. KẾT QUẢ VÀ ĐÁNH GIÁ:**

1. **KẾT QUẢ:**

* Sản phẩm:

+ Mô hình máy ấp trứng.

+ Máy ấp trứng được cài đặt nhiệt độ 360C-370C trong suốt quá trình ấp.

+ Cảm biến nhiệt độ độ ẩm hiển thị lên LCD ở thời điểm hiện tại để theo dõi.

+ Một động cơ đảo trứng theo chu kì.

+ Có thể điều khiển , ổn định nhiệt độ độ ẩm theo chu kì ấp từng loại trứng bằng phương pháp on/off điều khiển nguồn nhiệt sử dụng đèn và quạt thông gió.

+ 4 nút nhấn để cài đặt chế độ của nhiệt độ và độ ẩm.

- Kiến thức : Hiểu một số chức năng của vi điều khiển , các bộ timer , chân I/O của STM32F411 ứng dụng vào máy ấp trứng và sáng tạo các sản phẩm khác trong tương lai.

1. **ĐÁNH GIÁ:**

* Sản phầm còn nhiều hạn chế như chưa lưu được giá trị vào bộ nhớ, cài đặt nhiệt độ độ ẩm theo ngày của chu kì ấp.
* Đạt được 80% yêu cầu đặt ra ban đầu của đề tài.

**VII. KẾT LUẬN VÀ HƯỚNG PHÁT TRIỀN CỦA ĐỀ TÀI:**

* Hệ thống máy ấp trứng vẫn chưa hoàn thiện và chúng em hi vọng trong tương lai sẽ phát triển máy ấp trứng với nhiều chức năng hơn để quá trình ấp hiệu quả và đạt năng suất cao hơn.
* Tăng công suất của bóng đèn và động cơ đảo trứng, mở rộng mô hình, có thể điều khiển nhiệt độ độ ẩm theo ngày và lưu các giá trị vào bộ nhớ để lưu trữ dữ liệu và theo dõi quá trình ấp trứng, hoàn thiện hệ thống để áp dụng mô hình vào thực tế sử dụng trong các hộ nông dân hay nông trại nhỏ trong ngành chăn nuôi gia cầm tạo được giá trị thương mại , tạo sản phẩm giá rẻ nhưng chất lượng tốt để phù hợp cho những người nông dân chăn nuôi gia cầm mô hình nhỏ.