Code ghép module DS1307 DHT11 và Oled

#include <Wire.h>

#include <Adafruit\_GFX.h>

#include <Adafruit\_SH1106.h>

#include "RTClib.h"

#include <DHT.h>

#define SCREEN\_WIDTH 128

#define SCREEN\_HEIGHT 64

RTC\_DS1307 rtc;

#define OLED\_RESET     4

Adafruit\_SH1106 display(OLED\_RESET);

#define DHTPIN 2

#define DHTTYPE DHT11

DHT dht(DHTPIN, DHTTYPE);

const float NHIET\_DO\_CAO = 35.0; //Khai bao 1 bien nhiet do cao = 35 do C

byte Buzzer = 7;

byte \_day, \_month, \_hour24, \_hour12, \_minute, \_second, \_dtw;

int \_year;

byte hr24;

char st[3];

char \_nameoftheDay[9];

char \_monthsName[3];

const char daysOfTheWeek\_0[] PROGMEM = "Sunday";

const char daysOfTheWeek\_1[] PROGMEM = "Monday";

const char daysOfTheWeek\_2[] PROGMEM = "Tuesday";

const char daysOfTheWeek\_3[] PROGMEM = "Wednesday";

const char daysOfTheWeek\_4[] PROGMEM = "Thursday";

const char daysOfTheWeek\_5[] PROGMEM = "Friday";

const char daysOfTheWeek\_6[] PROGMEM = "Saturday";

const char\* const daysOfTheWeek\_table[] PROGMEM = {daysOfTheWeek\_0, daysOfTheWeek\_1, daysOfTheWeek\_2, daysOfTheWeek\_3, daysOfTheWeek\_4, daysOfTheWeek\_5, daysOfTheWeek\_6};

const char months\_name\_0[] PROGMEM = "Jan";

const char months\_name\_1[] PROGMEM = "Feb";

const char months\_name\_2[] PROGMEM = "Mar";

const char months\_name\_3[] PROGMEM = "Apr";

const char months\_name\_4[] PROGMEM = "May";

const char months\_name\_5[] PROGMEM = "Jun";

const char months\_name\_6[] PROGMEM = "Jul";

const char months\_name\_7[] PROGMEM = "Aug";

const char months\_name\_8[] PROGMEM = "Sep";

const char months\_name\_9[] PROGMEM = "Oct";

const char months\_name\_10[] PROGMEM = "Nov";

const char months\_name\_11[] PROGMEM = "Dec";

const char\* const months\_name\_table[] PROGMEM = {months\_name\_0, months\_name\_1, months\_name\_2, months\_name\_3,

                                                 months\_name\_4, months\_name\_5, months\_name\_6, months\_name\_7,

                                                 months\_name\_8, months\_name\_9, months\_name\_10, months\_name\_11,};

void setup() {

  Serial.begin(115200);

    pinMode(Buzzer, OUTPUT);

    if (! rtc.begin()) {

    Serial.println("Couldn't find RTC");

    while (1);

  }

  if (! rtc.isrunning()) {

    Serial.println("RTC is NOT running!");

  }

  dht.begin();

  display.begin(SH1106\_SWITCHCAPVCC, 0x3C);

  delay(1000);

  display.clearDisplay();

}

void loop() {

  display.clearDisplay();

  GetDateTime();

  display.clearDisplay();

  Digital\_Clock\_and\_Date(\_hour12, \_minute, \_second, \_dtw, \_day, \_month ,\_year, st);

  float temperature = dht.readTemperature();  // Đọc giá trị nhiệt độ từ DHT11

  float humidity = dht.readHumidity();  // Đọc giá trị độ ẩm từ DHT11

  display.setCursor(0, 40);

  display.setTextSize(1);

  display.setTextColor(WHITE);

  display.print("Temp: ");

  display.print(temperature);

  display.print(" C");

  display.setCursor(0, 50);

  display.print("Hum: ");

  display.print(humidity);

  display.print(" %");

  if (temperature > NHIET\_DO\_CAO) {

    Button\_Sound(1); // Bật buzzer

    delay(10000); // Giữ buzzer kêu trong 10 giây

    Button\_Sound(0); // Tắt buzzer

  }

  display.display();

}

void GetDateTime() {

  DateTime now = rtc.now();

  \_day = now.day();

  \_month = now.month();

  \_year = now.year();

  \_hour24 = now.hour();

  \_minute = now.minute();

  \_second = now.second();

  \_dtw = now.dayOfTheWeek();

  if (\_hour24 >= 12) {

    strncpy(st, "PM\0", sizeof(st));

    if (\_hour24 > 12) {

      \_hour12 = \_hour24 - 12;

    } else {

      \_hour12 = 12;

    }

  } else {

    strncpy(st, "AM\0", sizeof(st));

    if (\_hour24 == 0) {

      \_hour12 = 12;

    } else {

      \_hour12 = \_hour24;

    }

  }

}

void Digital\_Clock\_and\_Date(byte h, byte m, byte s, byte dtw, byte dy, byte mt, int yr, char\* st) {

  display.setTextColor(WHITE);

  display.setCursor(35, 0);

  if (h < 10) {

    display.print("0");

    display.print(h);

  } else {

    display.print(h);

  }

  display.print(":");

  if (m < 10) {

    display.print("0");

    display.print(m);

  } else {

    display.print(m);

  }

  display.print(":");

  if (s < 10) {

    display.print("0");

    display.print(s);

  } else {

    display.print(s);

  }

  display.print(" ");

  display.print(st);

  strcpy\_P(\_nameoftheDay,(char\*)pgm\_read\_word(&(daysOfTheWeek\_table[dtw])));//Để tiết kiệm bộ nhớ RAM lưu trữ dữ liệu này trong RAM có thể lưu trữ dữ liệu này trong bộ nhớ flash và truy cập dữ liệu đó bằng lệnh pgm\_read\_\*

  display.setCursor(0, 10);

  display.println(\_nameoftheDay);

  strcpy\_P(\_monthsName,(char\*)pgm\_read\_word(&(months\_name\_table[mt-1])));

  display.setCursor(50, 10);

  if (dy < 10) {

    display.print("0");

    display.print(dy);

  } else {

    display.print(dy);

  }

  display.print(", ");

  display.print(\_monthsName);

  display.setCursor(95,10);

  display.print(yr);

}

void Button\_Sound(byte snd) { //điều khiển đầu ra âm thanh của buzzer bằng cách sử dụng hàm digitalWrite(). Tham số snd cho biết trạng thái của buzzer: 1 để bật và 0 để tắt.

  digitalWrite(Buzzer, snd);

}

\*Nghiên cứu:

Dòng 22:

**byte \_day, \_month, \_hour24, \_hour12, \_minute, \_second, \_dtw;**- khai báo nhiều biến kiểu dữ liệu **byte**(là kiểu số nguyên không dấu, có thể lưu giá trị từ 0 đến 255) với các tên **\_day**, **\_month**, , **\_hour24**, **\_hour12**, **\_minute**, **\_second**, **\_dtw**. Các biến này có khả năng được sử dụng để lưu trữ thông tin về ngày và giờ.

**int \_year;-** khai báo một biến có kiểu dữ liệu int(là kiểu số nguyên có dấu, có thể lưu giá trị từ -2147483648 đến 2147483647) với tên \_year. Biến này có khả năng được sử dụng để lưu trữ thông tin về một năm. Biến **\_year** được sử dụng để lưu trữ giá trị của năm, và giá trị năm có thể lớn hơn 255 (giá trị tối đa mà một biến byte có thể lưu trữ). Vì vậy, để đảm bảo rằng biến có thể lưu trữ tất cả các giá trị năm, biến **\_year** được định nghĩa là kiểu **int**, có thể lưu trữ giá trị từ -32768 đến 32767.

**byte hr24;-** khai báo một biến kiểu dữ liệu bytecó tên hr24. Biến này có khả năng được sử dụng để lưu trữ thông tin về việc liệu thời gian có ở định dạng 24 giờ hay không.

**char st[3];**- khai báo một mảng các ký tự có tên **st** và kích thước là 3. Biến này có thể được sử dụng để lưu trữ một chuỗi ký tự. (Hiển thị AM hoặc PM). Dòng 116 và 123. Tại sao AM và PM chỉ cần 2 kí tự là đủ AM\0 hoặc PM\0 \0' đại diện cho ký tự null character, còn được gọi là ký tự kết thúc chuỗi. ta cần kết thúc chuỗi bằng ký tự null character để cho biết rằng chuỗi đã kết thúc.

Vì vậy, khi sử dụng chuỗi "PM", chúng ta cần kết thúc chuỗi bằng ký tự null character. Khi đó, chuỗi "PM" sẽ trở thành "PM\0".

Trong trường hợp này, ta đang sử dụng một chuỗi để đại diện cho định dạng giờ 12 giờ hoặc 24 giờ. Vì vậy, nếu không kết thúc chuỗi bằng ký tự null character, các lệnh khác có thể không hiểu được định dạng của chuỗi này.

Do đó, khi khai báo chuỗi "PM" trong đoạn mã trên, chúng ta cần thêm ký tự null character vào cuối chuỗi bằng cách sử dụng "\0". Kết quả là "PM\0"

**char \_nameoftheDay[9];**- khai báo một mảng các ký tự có tên **\_nameoftheDay**và kích thước là 9. Biến này có thể được sử dụng để lưu trữ tên của một ngày trong tuần.

**char \_monthsName[3];-** khai báo một mảng các ký tự có tên \_monthsNamevà kích thước là 3. Biến này có thể được sử dụng để lưu trữ tên của một tháng.

Dòng 30-38:

Các dòng code này định nghĩa một mảng các chuỗi ký tự (hay còn gọi là mảng các strings) để lưu trữ tên của các ngày trong tuần.

Các dòng code đầu tiên tạo ra các mảng daysOfTheWeek\_X (với X từ 0 đến 6) để lưu trữ tên của các ngày trong tuần tương ứng với thứ tự từ 0 đến 6, ví dụ daysOfTheWeek\_0 lưu trữ chuỗi ký tự "Sunday" (chủ nhật), daysOfTheWeek\_1 lưu trữ chuỗi ký tự "Monday" (thứ hai), và cứ như vậy.

Dòng code tiếp theo định nghĩa một mảng daysOfTheWeek\_table để lưu trữ tất cả các chuỗi ký tự tên của các ngày trong tuần. Mảng này được khai báo là const char\* const, có chứa các con trỏ (pointers) tới các chuỗi ký tự được lưu trữ trong các mảng daysOfTheWeek\_X. Mảng này cũng được đánh dấu là PROGMEM để lưu trữ trong bộ nhớ flash của vi điều khiển, giúp giảm tải cho bộ nhớ RAM của vi điều khiển.

dòng 157:

strcpy\_P(\_nameoftheDay,(char\*)pgm\_read\_word(&(daysOfTheWeek\_table[dtw])));

//Để tiết kiệm bộ nhớ RAM lưu trữ dữ liệu này trong RAM có thể lưu trữ dữ liệu này trong bộ nhớ flash và truy cập dữ liệu đó bằng lệnh pgm\_read\_\*

**strcpy\_P()** được sử dụng để sao chép chuỗi được lưu trữ trong bộ nhớ flash của Arduino (Program memory hay PROGMEM) vào bộ nhớ SRAM (Data memory).

strcpy\_P(\_nameoftheDay, (char\*)pgm\_read\_word(&(daysOfTheWeek\_table[dtw]))) được sử dụng để sao chép chuỗi tên ngày trong tuần từ bảng daysOfTheWeek\_table vào mảng \_nameoftheDay.

**strcpy\_P(\_monthsName,(char\*)pgm\_read\_word(&(months\_name\_table[mt-1])))** được sử dụng để sao chép chuỗi tên tháng từ bảng **months\_name\_table** vào mảng **\_monthsName**.

**pgm\_read\_word()** được sử dụng để đọc một từ (word) 16 bit từ bộ nhớ flash, và **(char\*)** được sử dụng để chuyển đổi con trỏ từ kiểu **const \_\_memx void\*** sang kiểu **char\***, để **strcpy\_P()** có thể sao chép chuỗi từ bộ nhớ flash vào bộ nhớ SRAM.

Trong đoạn code trên, việc sử dụng "\_hour24 > 254" trong điều kiện "if" là để kiểm tra xem giá trị của "\_hour24" có nhỏ hơn 0 hay không. Thông thường, khi một biến kiểu byte (8-bit) được giảm xuống giá trị âm, nó sẽ chuyển sang giá trị dương tương ứng với giá trị tuyệt đối của giá trị âm đó bằng cách bổ sung 2^(8), tức là 256. Vì thế, nếu "\_hour24" được giảm xuống giá trị âm, nó sẽ trở thành một số lớn hơn 23.

Do đó, trong trường hợp giá trị của "\_hour24" vượt quá 23 sau khi giảm đi một đơn vị, ta cần kiểm tra nếu giá trị của "\_hour24" lớn hơn 254 (2^8 - 2^1 - 1), thì nó có nghĩa là "\_hour24" đã bị chuyển sang giá trị dương và ta cần thiết lập lại nó bằng 23 để giữ giá trị giờ trong khoảng từ 0 đến 23.