کوییز سوم

سیستمهای نهفته و بیدرنگ

استاد درس

دکتر حسینی منزه

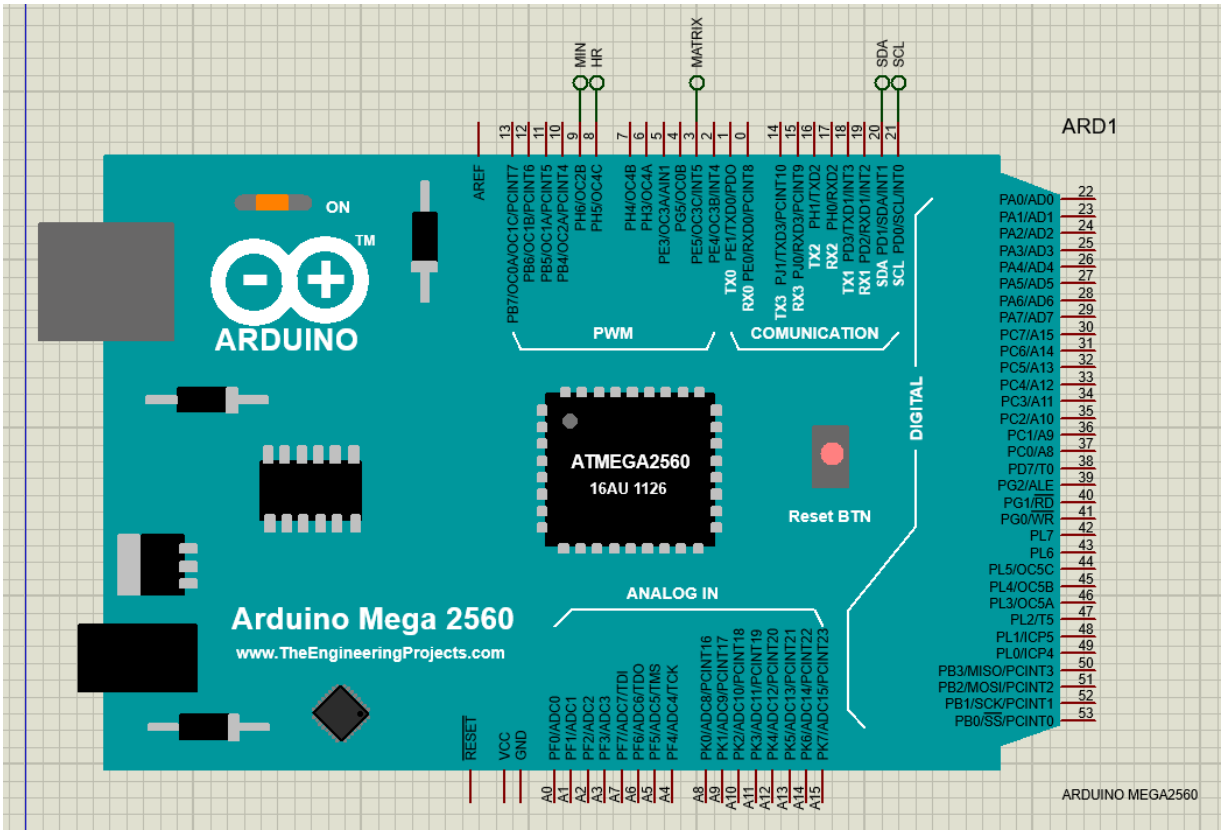
دانشجو

امیرحسین احمدی

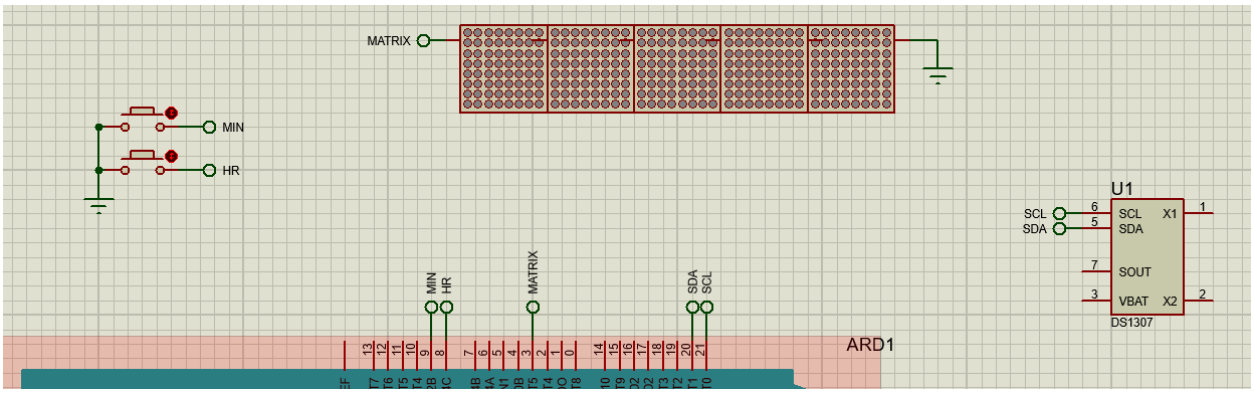
۹۷۵۲۲۲۹۲

بهار ۱۴۰۱

برای حل این سوال از یک پردازنده Arduino Mega 2560 کمک میگیریم که تصویر آن در نرم افزار Proteus را در زیر میبینید (برای اضافه کردن آن به Proteus نیاز به اضافه کردن تعدادی Library جداگانه بود.)



برای پیاده سازی این سوال، نیاز به یک کلاک خارجی برای گرفتن زمان داشتیم که از قطعه DS1307 برای این کار استفاده کردیم. همچنین برای پیاده سازی تابلوی روان ۵ قطعه WS2812X8X8 که به صورت سطری به هم متصل کرده ایم استفاده کردیم که هر کدام شامل ۸ * ۸ قطعه LED میباشند. برای تنظیم ساعت و دقیقه نیز از دو دکمه استفاده کردیم که زمین وصل شده اند.



همانگونه که میبینید، دو دکمه را به دو Pin 8 و 9 ، LED ها را به 3 Pin و کلاک را به پین های مربوطه وصل کرده ایم.

در کد، ابتدا هر کدام از Pin های گفته شده را تعریف میکنیم تا در ادامه نیاز به نوشتن اعداد نباشد. همچنین تعداد LCD های استفاده شده را تعریف میکنیم تا در ادامه بتوانیم در محاسبات استفاده کنیم. MAX_CHR نیز حداکثر مقدار حروفی است که میتوانیم نشان دهیم (به صورت متحرک) که در ادامه از آن استفاده خواهیم کرد.

```
/*-----------------------------------------------------------------------------------------*/

#define DATA_PIN 3 // Serial pin
#define MIN_PIN 9
#define HR_PIN 8

#define LCD_NUM 5
#define MAX_CHR 20
```

در ادامه متغییر های Global که در ادامه نیاز داریم را تعریف میکنیم. ابتدا یک آرایه به طول تعداد LED های موجود در تابلوی روان است که از نوع CRGB است. CRGB مربوط به FastLED است که میتوان در آن رنگ هر LED را تعیین کرد. یک آرایه عددی به اسم binary_result داریم که مقادیر 0 و ۱ میگیرد و نشان میدهد که LED خاموش است یا خیر. برای راحتی کار این آرایه متفاوت از leds در نظر گرفتیم. یک متغییر Boolean داریم که نشان میدهد که در حال نمایش زمان هستیم یا خیر. برای تمام مقادیر ثانیه، دقیقه، ساعت، سال، تاریخ و ماه یک متغییر گرفتیم زیرا در هنگام ارتباط به Clock خارجی و Update کردن آن باید کل این مقادیر را رد و بدل کنیم. در آخر یک آرایه داریم که به ازای هر کاراکتر قابل نمایش در کد ما ۹ متغییر را نگه میدارد که اولین متغییر برابر با مقدار اسکی آن کاراکتر است و ۸ متغییر بعدی اعدادی هستند که اگر به حالت Binary در نظر بگیریم، وضعیت روشن و خاموش بودن LED ها را نشان میدهند اگر بخواهیم کاراکتر مورد نظر را در یک ۸ در ۸ نمایش دهیم.

```
/*-----------------------------------------------------------------------------------------*/

CRGB leds[LCD_NUM * 8 * 8];
int binary_result[MAX_CHR][64];
bool display_time = false;
byte second, minute, hour, year, date, month;

uint8_t alphabets[855]={
    0x20, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,   //
    0x21, 0x18, 0x3c, 0x3c, 0x18, 0x18, 0x00, 0x18, 0x00,   //>!
    0x22, 0x6c, 0x6c, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,   //"
    0x23, 0x6c, 0x6c, 0xfe, 0x6c, 0xfe, 0x6c, 0x6c, 0x00,   //#
```

در تابع setup ابتدا FastLED را Initialize میکنیم و که در آن Pin خروجی، آرایه خروجی و سایز آرایه را به FastLED میدهیم و میزان روشنایی Pixel ها را نیز تعیین میکنیم. سپس Wire را که برای ارتباط با Clock خارجی به آن نیاز داریم begin میکنیم. دو Pin مربوط به تنظیم ساعت و دقیقه را به عنوان ورودی تنظیم میکنیم و تابع string_to_pixel_array را با اسم خودم صدا میزنم تا با تبدیل آن به آرایه binary آماده نمایش روی LCD ها شود.

```
void setup() {
    FastLED.addLeds<NEOPIXEL, DATA_PIN>(leds, LCD_NUM * 8 * 8);
    FastLED.setBrightness(255);

    Wire.begin(); // for clock

    pinMode(HR_PIN, INPUT_PULLUP);
    pinMode(MIN_PIN, INPUT_PULLUP);

    string_to_pixel_array("AmirHossein");
}
```

تابع string_to_pixel_array که پیش تر به آن اشاره شد به این صورت عمل میکند که به ازای هر کاراکتر در String داده شده به آن Index آن در آرایه alphabets که توضیح داده شد را پیدا میکند و به ترتیب تمام اعداد آن را با استفاده از get_binary_array به آرایه Binary تبدیل کرده و درون binary_result که توضیح داده شد میریزد.

```
void string_to_pixel_array (String name) {
    for (int i = 0; i < name.length(); i++) {
        int j = ((int) name[i] - 0x20) * 9; // Index of alphabets
        for (int r = 0; r < 8; ++r) {
            int binary_arr[8];
            get_binary_array (alphabets[j + r + 1], binary_arr);
            for (int c = 0; c < 8; ++c) {
                binary_result[i][r * 8 + c] = binary_arr[c];
            }
        }
    }
}
```

در تابع get_binary_array که پیشتر اشاره شد با تقسیم بر ۲ کردن مداوم عدد آرایه Binary را از آن استخراج میکند.

```
void get_binary_array(int number, int binary_arr[8]) {
    int i = 7;
    while (i >= 0) {
        binary_arr[i] = number % 2;
        number /= 2;
        i--;
    }
}
```

در تابع loop به این صورت عمل میکنیم که در صورتی که هنوز نمایش اسم تمام نشده (display_time برابر با false است) تابع shift_left را صدا میزنیم که Pixel های موجود در binary_result را یک واحد شیفت میدهد تا اسم به صورت متحرک باشد. در غیر این صورت ابتدا با صدا زدن get_time_details اطلاعات زمانی را از Clock میگیرد، سپس در صورت فشرده شدن دکمه های تنظیم زمان هر کدام از ساعت و دقیقه را تنظیم میکنیم و در نهایت با تبدیل کردن دقیقه و ساعت به String و فرستادن آن به string_to_pixel_array آن ها را در خروجی قرار میدهیم. در نهایت تابع display که آرایه binary_result را درون leds میریزد صدا میزنیم و با دستور FastLED.show تابع leds را خروجی میدهیم و در آخر مقداری Delay میدهیم تا نمایش Pixel ها را بتوانیم حس کنیم.

```
void loop() {
    if (!display_time) {
        shift_left();
    } else {
        get_time_details();

        if(!digitalRead(HR_PIN)) {
            while(!digitalRead(HR_PIN)){}
            increase_hour_value();
        }

        if(!digitalRead(MIN_PIN)) {
            while(!digitalRead(MIN_PIN)){}
            increase_minute_value();
        }

        string_to_pixel_array(time_to_string((int) hour, (int) minute));
    }

    display();

    FastLED.show();
    delay(10);
}
```

در تابع shift_left به ازای هر led آن را برابر با led سمت چپی باید قرار دهیم که برای خانه های هر کاراکتر در binary_result به خانه ی بعدی مراجعه میکنیم ولی برای خانه های آخر مقدار خانه های اول کاراکتر بعدی را بر میداریم. همچنین چک میکنیم که اگر تمام Pixel ها خاموش بودند یعنی نمایش اسم به پایان رسیده و display_time را True میکنیم.

```
void shift_left(){
    bool flag = false;
    for(int i = 0; i < MAX_CHR; ++i) {
        for(int j = 0; j < 64; ++j){
            if (j % 8 == 7) {
                binary_result[i][j] = binary_result[i + 1][j - 7];
            } else {
                binary_result[i][j] = binary_result[i][j + 1];
            }

            if (binary_result[i][j] == 1) {
                flag = true;
            }
        }
    }

    if (!flag)
        display_time = true;
}
```

در تابع time_to_string با گرفتن ساعت و دقیقه و با استفاده از تابع itoa برای تبدیل string به int ساعت و دقیقه را به فرمت HH:MM در میاوریم. (هر کدام کم تر از ۱۰ بود یک صفر پشت آن میگذاریم.)

```
String time_to_string(int h, int m) {
    char buffer[MAX_CHR];
    String res = "";

    // hour
    if (h < 10) res += "0";
    itoa(h, buffer, 10);
    res += buffer;
    res += ":";

    // minute
    if (m < 10) res += "0";
    itoa(m, buffer, 10);
    res += buffer;

    return res;
}
```

در تابع increase_minute_value مقدار دقیقه را یکی بالا برده و در صورت ۶۰ شدن صفر کرده و ساعت را بالا میبریم و در صورت ۲۴ شدن ساعت آن را نیز صفر میکنیم. مقدار آن را به BCD تبدیل کرده و با استفاده از wire به Clock ارسال میکنیم.

```
void increase_minute_value() {
    if ((int)minute + 1 == 60) {
        minute = 0;
        hour++;
    } else {
        minute += 1;
    }
    if (hour == 24){
        hour =0 ;
    }
    minute = ((minute / 10) << 4) + (minute % 10);
    hour = ((hour / 10) << 4) + (hour % 10);


    Wire.beginTransmission(0x68);              // Start I2C protocol with DS1307 address
    Wire.write(0);                             // Send register address
    Wire.write(0);                             // Reset sesonds and start oscillator
    Wire.write(minute);                        // Write minute
    Wire.write(hour);                          // Write hour
    Wire.write(1);                             // Write day (not used)
    Wire.write(date);                          // Write date
    Wire.write(month);                         // Write month
    Wire.write(year);                          // Write year
    Wire.endTransmission();                    // Stop transmission and release the I2C bus
}
```

در تابع increase_hour_value نیز مانند بالا عمل کرده فقط با این تفاوت که مقدار ساعت را یکی زیاد میکنیم.

```
void increase_hour_value() {
    if ((int)hour + 1 == 24) {
        hour = 0;
    } else {
        hour += 1;
    }
    minute = ((minute / 10) << 4) + (minute % 10);
    hour = ((hour / 10) << 4) + (hour % 10);


    Wire.beginTransmission(0x68);               // Start I2C protocol with DS1307 address
    Wire.write(0);                              // Send register address
    Wire.write(0);                              // Reset sesonds and start oscillator
    Wire.write(minute);                         // Write minute
    Wire.write(hour);                           // Write hour
    Wire.write(1);                              // Write day (not used)
    Wire.write(date);                           // Write date
    Wire.write(month);                          // Write month
    Wire.write(year);                           // Write year
    Wire.endTransmission();                     // Stop transmission and release the I2C bus
}
```

در تابع get_time_details مقادیر زمانی را با استفاده از wire خوانده و دو متغییر دقیقه و ساعت که برای ما اهمیت دارند را از BCD به Decimal تبدیل میکنیم تا بتوانیم از آن ها استفاده کنیم.

```
void get_time_details() {
    Wire.beginTransmission(0x68);               // Start I2C protocol with DS1307 address
    Wire.write(0);                              // Send register address
    Wire.endTransmission(false);                // I2C restart
    Wire.requestFrom(0x68, 7);                  // Request 7 bytes from DS1307 and release I2C bus at end of reading
    second = Wire.read();
    minute = Wire.read();                       // Read minuts from register 1
    hour   = Wire.read();                       // Read hour from register 2
    Wire.read();

    //BCD to Decimal
    minute = (minute >> 4) * 10 + (minute & 0x0F);
    hour   = (hour >> 4)   * 10 + (hour & 0x0F);
}
```

در نهایت در تابع display مقادیر binary_result را از اول تا جایی که میتوان درون leds میریزیم تا بتوانیم خروجی دهیم. اگر LED روشن باشد آن را قرمز و اگر خاموش باشد آن را مشکی میکنیم.

```
void display() {
    for(int i = 0; i < 5; ++i){
        for(int j = 0; j < 64; ++j){
            if(binary_result[i][j] == 1) {
                leds[i * 64 + j] = CRGB::Red;
            } else {
                leds[i * 64 + j] = CRGB::Black;
            }
        }
    }
}
```