**暨南大学本科实验报告专用纸**

课程名称 物联网应用课程设计 成绩评定

实验项目名称 基础实验-工厂环境检测系统 指导教师 李晓帆

实验项目编号 0814400101 实验项目类型 验证 实验地点 B305

学生姓名 郑晓鹏 学号 2022101950 学院 智能科学与工程学院 专业 物联网工程 实验时间 2024 年 10 月 15 日

**实验目的**

1. 熟悉实验开发平台，使用ESP8266模块实现自动连接WiFi并实现Web端实时显示环境温湿度，将数据通过串口发送给Arduino Mega控制LED灯亮灭。
2. 在原有基础上，在Arduino端增加继电器模块控制直流电机转动，模拟工厂开启通风扇散热；增加光敏模块检测环境的光照情况并控制LED补光灯进行补光；增加火焰传感器监测工厂的消防情况并控制蜂鸣器进行报警；增加了OLED小屏幕，对温湿度、继电器开关、消防报警信息进行实时显示，不需要查询Web端也能得知环境信息。
3. 实现ESP8266与Arduino Mega串口双向通信，将Arduino Mega检测的消防情况和继电器工作状态回传给ESP8266并显示在Web页面上。

**实验过程**

1. 编写ESP8266端程序，代码及详细注释如下：

|  |
| --- |
| #include <ESP8266WiFi.h>  #include <DHT.h>  // 定义温湿度传感器相关引脚 并将DHT类实例化  int pinDHT = D7;  DHT dht(pinDHT, DHT11);  // 定义WiFi模块作为STA连接的AP的标识名和密码  const char \*ssid = "Zzz";  const char \*password = "1234567890";  // 将WiFiServer类实例化，WiFi模块作为TCP服务器，端口号为80  WiFiServer server(80);  // 定义串口缓冲区、传感器逻辑判断标志等变量  #define BUFFER\_SIZE 64    // 宏定义串口接收缓冲区大小  char buffer[BUFFER\_SIZE]; // 串口接收缓冲区  int bufferIndex = 0;      // 缓冲区索引  int relayState = 0;       // 继电器状态  int fireFlag = 0;         // 消防状态  int lightFlag = 0;        // 补光灯状态  // 上电初始化配置函数  void setup()  {      Serial.begin(115200); // 串口初始化 波特率115200      pinMode(2, OUTPUT);   // 2号引脚配置为输出模式，板载LED灯      digitalWrite(2, 0);   // 2号引脚拉低，TX有数据后板载LED会闪烁      dht.begin();          // DHT11初始化      // WiFi模块作为STA连接到AP 并通过串口打印相关信息      Serial.println();      Serial.print("Connecting to ");      Serial.println(ssid);      WiFi.begin(ssid, password);      // 500ms查询一次连接状态，不成功打点      while (WiFi.status() != WL\_CONNECTED)      {          delay(500);          Serial.print(".");      }      // 连接成功 打印WiFi模块由AP分配到的地址      Serial.println("");      Serial.println("WiFi connected");      server.begin();      Serial.print("IP address: ");      Serial.println(WiFi.localIP());  }  // loop循环函数  void loop()  {      // 实例化WiFiClient类，调用server.available()监听是否有客户端连接      WiFiClient client = server.available();      // 没有客户端连接，延时2s后返回，重新执行loop()函数      if (!client)      {          delay(2000);          return;      }      // 定义温湿度浮点数变量，调用dht的方法解算传感器温湿度信息      float humidity = dht.readHumidity();      float temperature = dht.readTemperature();      // 检查读取是否成功 不成功则直接返回重新执行loop()函数      if (isnan(humidity) || isnan(temperature))      {          Serial.println("无法从DHT传感器读取数据!");          return;      }      // 串口打印温湿度数据      Serial.print("DHT11 ~ TEM-HUM : ");      Serial.print((int)temperature);      Serial.print("℃,");      Serial.print((int)humidity);      Serial.println("%");      delay(1000); // DHT11数据更新周期为1s      Serial.print(temperature);      Serial.print(",");      Serial.println(humidity);      // 如果串口RX有数据，阻塞接收串口数据      while (Serial.available() > 0)      {          char incomingByte = Serial.read(); // 读取一个字节的数据          // 如果是换行符或回车符，则处理缓冲区中的数据          if (incomingByte == '\n' || incomingByte == '\r')          {              buffer[bufferIndex] = '\0'; // 在字符串末尾添加空字符构成字符串              // 根据接收到的数据执行相应的操作              if (strcmp(buffer, "Relay ON") == 0)              {                  relayState = 1; // 继电器状态位置1              }              else if (strcmp(buffer, "Relay OFF") == 0)              {                  relayState = 0; // 继电器状态位置0              }              else if (strcmp(buffer, "Warning") == 0)              {                  fireFlag = 1; // 火焰标志位置1              }              else if (strcmp(buffer, "Normal") == 0)              {                  fireFlag = 0; // 火焰标志位置0              }              // 重置缓冲区索引以准备接收新的数据              bufferIndex = 0;          }          else // 如果不是换行符或回车符，则将字符添加到缓冲区中          {              if (bufferIndex < BUFFER\_SIZE - 1) // 确保不会溢出缓冲区              {                  buffer[bufferIndex++] = incomingByte;              }              else // 如果缓冲区已满 重置缓冲区以丢弃旧数据              {                  bufferIndex = 0;              }          }      }      // 刷新客户端缓存区 定义HTML页面转换成的字符串      client.flush();      String s = "HTTP/1.1 200 OK\r\n"                 "Content-Type: text/html\r\n\r\n"                 "<!DOCTYPE html>\n"                 "<html lang=\"en\">\n"                 "<head>\n"                 "    <meta charset=\"UTF-8\">\n"                 "    <meta http-equiv=\"Refresh\" content=\"5\">\n"                 "    <meta name=\"viewport\" content=\"width=device-width, initial-scale=1.0, minimum-scale=0.5, maximum-scale=2.0, user-scalable=yes\">\n"                 "    <meta http-equiv=\"X-UA-Compatible\" content=\"ie=edge\">\n"                 "    <title>工厂环境检测系统</title>\n"                 "    <style>\n"                 "        h2, h1 { line-height: 1%; }\n"                 "        body { margin: 0; padding: 0; width: 340px; background: LightCyan; }\n"                 "        .button { width: 100px; height: 100px; text-align: center; font-weight: 100; color: darkcyan; margin: 0 40px 40px 0; position: relative; border: 6px solid darkcyan; background: LightCyan; font-size: 20px; border-radius: 50%; }\n"                 "        .top1 { width: 360px; height: 45px; color: #FFF; border: 1px solid darkcyan; background: darkcyan; font-size: 25px; border-radius: 0%; }\n"                 "    </style>\n"                 "</head>\n"                 "<body>\n"                 "    <button class=\"button top1\">工厂环境检测系统</button>\n"                 "    <center style=\"left: 20px; position: relative;\">\n"                 "        <br/>\n"                 "        <button type=\"button\" class=\"button\" value=\"temp\">温度<span style=\"color: red; font-size: 25px;\">";      // 拼接温度和湿度数据      s += ((int)temperature);      s += "°C</span></button>\n";      s += "        <button type=\"button\" class=\"button\" value=\"humi\">湿度<span style=\"color: green; font-size: 25px;\">";      s += ((int)humidity);      s += "%</span></button>\n";      // 拼接继电器状态信息      s += "        <button type=\"button\" class=\"button\" value=\"humi\">继电器<span style=\"color: green; font-size: 25px;\">";      if (relayState == 1)      {          s += "<b>ON</b>";      }      else      {          s += "<b>OFF</b>";      }      s += "</span></button>\n";      // 拼接继电器状态信息      if (fireFlag == 1)      {          s += "        <button type=\"button\" class=\"button\" value=\"humi\">消防<br><span style=\"color: red; font-size: 25px;\">";          s += "<b>异常</b>";      }      else      {          s += "        <button type=\"button\" class=\"button\" value=\"humi\">消防<br><span style=\"color: green; font-size: 25px;\">";          s += "<b>正常</b>";      }      s += "</span></button>\n";      // 闭合HTML标签      s += "    </center>\n"           "</body>\n"           "</html>";      // 将HTML字符串发送给客户端      client.print(s);      delay(1);  } |

1. 编Arduino Mega端程序，代码及详细注释如下：

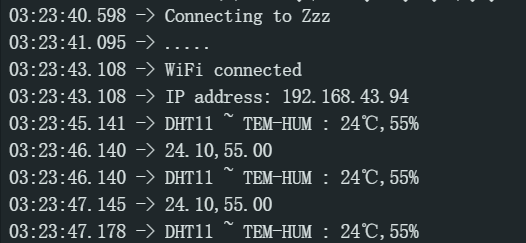
|  |
| --- |
| // OLED显示屏相关库  #include <Adafruit\_GFX.h>  #include <Adafruit\_SSD1306.h>  // 定义相关引脚  int tempLedPin = 13;        // 温度LED  int humLedPin = 12;         // 湿度LED  int relayPin = 22;          // 继电器LED  int beepPin = 23;           // 有源蜂鸣器  int firePin = A1;           // 火焰传感器(模拟量)  int lightSensorPin = A0;    // 光敏模块(模拟量)  int lightCompentatePin = 2; // 补光LED(PWM)  // 定义相关变量  int temperature = 0;       // 温度  int humidity = 0;          // 湿度  int flameVal = 500;        // 火焰传感器模拟量 值小有火  int lightDuty = 0;         // 补光LED占空比 0~255值越大灯越亮  int lightSensorVal = 1000; // 光敏模块模拟量 0~1000值越高光线越暗  int tempThreshold = 24;    // 温度阈值  int humThreshold = 30;     // 湿度阈值  int posX = 0;              // OLED显示屏初始X坐标  int posY = 32;             // OLED显示屏初始Y坐标  // 实例化Adafruit\_SSD1306  // OLED显示屏是128x64，通信采用硬件I2C(Wire)  Adafruit\_SSD1306 display = Adafruit\_SSD1306(128, 64, &Wire);  // 上电初始化配置函数  void setup()  {      // 串口初始化 波特率115200      Serial.begin(115200);  // 默认串口 直接打印在串口监视器上      Serial1.begin(115200); // 串口1 与ESP8266全双工通信      // 温湿度LED、继电器、补光灯、蜂鸣器引脚配置为输出模式      pinMode(tempLedPin, OUTPUT);      pinMode(humLedPin, OUTPUT);      pinMode(relayPin, OUTPUT);      pinMode(lightCompentatePin, OUTPUT);      pinMode(beepPin, OUTPUT);      // 火焰传感器、光敏模块引脚配置为输入模式      pinMode(firePin, INPUT);      pinMode(lightSensorPin, INPUT);      // 初始化引脚电平      digitalWrite(tempLedPin, LOW);              // 高电平点亮 熄灭      digitalWrite(humLedPin, LOW);               // 高电平点亮 关      digitalWrite(relayPin, LOW);                // 继电器高电平导通 关      digitalWrite(beepPin, HIGH);                // 有源蜂鸣器低电平触发 关      analogWrite(lightCompentatePin, lightDuty); // 占空比初值为0 关      // 初始化OLED显示屏      display.begin(SSD1306\_SWITCHCAPVCC, 0x3C);      display.clearDisplay(); // 清屏      // 设置文本样式      display.setTextSize(2);      display.setTextColor(SSD1306\_WHITE);      // 菜单 温湿度继电器      display.setCursor(0, 0);      display.println("TEMP:");      display.setCursor(0, 16);      display.println("HUMI:");      display.setCursor(0, 32);      display.println("RELAY:");      // 显示温湿度继电器初始化信息      display.setCursor(62, 0);      display.println(temperature);      display.setCursor(62, 16);      display.println(humidity);      display.setCursor(92, 32);      display.println("OFF");      // 温湿度指示灯显示ON/OFF      display.setCursor(92, 0);      display.println("OFF");      display.setCursor(92, 16);      display.println("OFF");      // 更新显示内容      display.display();      Serial.println("Arduino Mega Ready"); // 默认串口打印  }  void loop()  {      // 读取光敏模块模拟量 值越大光线越亮      lightSensorVal = analogRead(lightSensorPin);      // 数值归一化为补光灯占空比      if (lightSensorVal >= 600)      {          // 光线良好不需要照明          lightDuty = 0;          // 串口1发送开灯信息给ESP8266处理          Serial1.println("Light OFF");      }      else if (lightSensorVal <= 300)      {          // 光线特别昏暗 补光灯占空比为1          lightDuty = 255;          // 串口1发送开灯信息给ESP8266处理          Serial1.println("Light ON");      }      else      {          // 光线比较昏暗 补光灯占空比随光线动态调整          lightDuty = map(lightSensorVal, 800, 300, 0, 255);          // 串口1发送开灯信息给ESP8266处理          Serial1.println("Light ON");      }      // 根据占空比 PWM调节补光灯亮度      analogWrite(lightCompentatePin, lightDuty);      // 读取火焰传感器模拟量 模拟量越小距离火源越近      flameVal = analogRead(firePin);      if (flameVal <= 150) // 有火      {          // 串口1发送消防报警信息给ESP8266处理          Serial1.println("Warning");          // 有源蜂鸣器报警(低电平触发)          digitalWrite(beepPin, LOW);          // 显示屏显示报警信息          display.setTextColor(SSD1306\_WHITE);          display.setCursor(0, 48);          display.println("WARNING!");          display.display();      }      else      {          // 串口1发送消防正常信息给ESP8266处理          Serial1.println("Normal");          // 关闭有源蜂鸣器          digitalWrite(beepPin, HIGH);          // 显示器隐去报警信息          display.setTextColor(SSD1306\_BLACK);          display.setCursor(0, 48);          display.println("WARNING!");          display.display();      }      // 串口1与ESP8266通信      // 如果串口1接收缓冲区有信息      if (Serial1.available() > 0)      {          // 把串口1接收缓冲区开头直到换行符 存储到字符串data里          String data = Serial1.readStringUntil('\n');          // 由于发送的温湿度数据格式是以“,”分隔开的          // 获取“,”的下标方便做处理          int commaIndex = data.indexOf(',');          // 定义温湿度局部变量          int temperature1 = 0;          int humidity1 = 0;          // “,”的下标大于0 说明传输正确          if (commaIndex > 0)          {              // 将逗号前后的字符串转换成整型数据分别存储到温湿度局部变量里              temperature1 = data.substring(0, commaIndex).toInt();              humidity1 = data.substring(commaIndex + 1).toInt();          }          // 温湿度局部变量不等于0 说明有读取到数据          if (temperature1 != 0 && humidity1 != 0)          {              // 如果温度局部变量和全局变量不一致，说明温度有更新              // 有更新则刷新OLED上的相关数据 没有则保持不变 以免出现频闪              if (temperature != temperature1)              {                  // OLED隐藏显示温度信息                  display.setTextColor(SSD1306\_BLACK);                  display.setCursor(62, 0);                  display.println(temperature);                  display.display();                  // 将温度全局变量更新                  temperature = temperature1;              }              // 如果湿度局部变量和全局变量不一致，说明湿度有更新              if (humidity != humidity1)              {                  // OLED隐藏显示湿度信息                  display.setTextColor(SSD1306\_BLACK);                  display.setCursor(62, 16);                  display.println(humidity);                  display.display();                  // 将湿度全局变量更新                  humidity = humidity1;              }          }          // 刷新显示OLED上给温湿度信息          display.setTextColor(SSD1306\_WHITE);          display.setCursor(62, 0);          display.println(temperature);          display.setCursor(62, 16);          display.println(humidity);          display.display();          // 默认串口打印温湿度数据到串口监视器          Serial.print("Temperature: ");          Serial.println(temperature);          Serial.print("Humidity: ");          Serial.println(humidity);          // 温度大于等于阈值          if (temperature >= tempThreshold)          {              // 点亮温度LED指示灯 默认串口打印信息              digitalWrite(tempLedPin, HIGH);              Serial.println("Temperature LED ON");              // 打开继电器 串口1发送继电器状态给ESP8266处理              digitalWrite(relayPin, HIGH);              Serial1.println("Relay ON");              // OLED显示 继电器信息为 ON              display.setTextColor(SSD1306\_BLACK);              display.setCursor(92, 32);              display.println("OFF");              display.setTextColor(SSD1306\_WHITE);              display.setCursor(104, 32);              display.println("ON");              display.display();              // OLED显示 温度LED指示灯信息为 ON              display.setTextColor(SSD1306\_BLACK);              display.setCursor(92, 0);              display.println("OFF");              display.setTextColor(SSD1306\_WHITE);              display.setCursor(104, 0);              display.println("ON");              display.display();          }          // 温度小于阈值          else          {              // 熄灭温度LED指示灯 默认串口打印信息              digitalWrite(tempLedPin, LOW);              Serial.println("Temperature LED OFF");              // 关闭继电器 串口1发送继电器状态给ESP8266处理              digitalWrite(relayPin, LOW);              Serial1.println("Relay OFF");              // OLED显示 继电器信息为 OFF              display.setTextColor(SSD1306\_BLACK);              display.setCursor(104, 32);              display.println("ON");              display.setTextColor(SSD1306\_WHITE);              display.setCursor(92, 32);              display.println("OFF");              display.display();              // OLED显示 温度LED指示灯信息为 OFF              display.setTextColor(SSD1306\_BLACK);              display.setCursor(104, 0);              display.println("ON");              display.setTextColor(SSD1306\_WHITE);              display.setCursor(92, 0);              display.println("OFF");              display.display();          }          // 湿度大于等于阈值          if (humidity >= humThreshold)          {              // 点亮湿度LED指示灯 默认串口打印信息              digitalWrite(humLedPin, HIGH);              Serial.println("Humidity LED ON");              // OLED显示 湿度LED指示灯信息为 ON              display.setTextColor(SSD1306\_BLACK);              display.setCursor(92, 16);              display.println("OFF");              display.setTextColor(SSD1306\_WHITE);              display.setCursor(104, 16);              display.println("ON");              display.display();          }          // 湿度小于阈值          else          {              // 熄灭湿度LED指示灯 默认串口打印信息              digitalWrite(humLedPin, LOW);              Serial.println("Humidity LED OFF");              // OLED显示 湿度LED指示灯信息为 OFF              display.setTextColor(SSD1306\_BLACK);              display.setCursor(98, 16);              display.println("ON");              display.setTextColor(SSD1306\_WHITE);              display.setCursor(104, 16);              display.println("OFF");              display.display();          }      }  } |

1. 调试

按照代码中的定义配置手机热点，将手机作为AP，WiFi模块作为STA。打开手机热点，复位ESP8266后会进行扫描并自动连接。

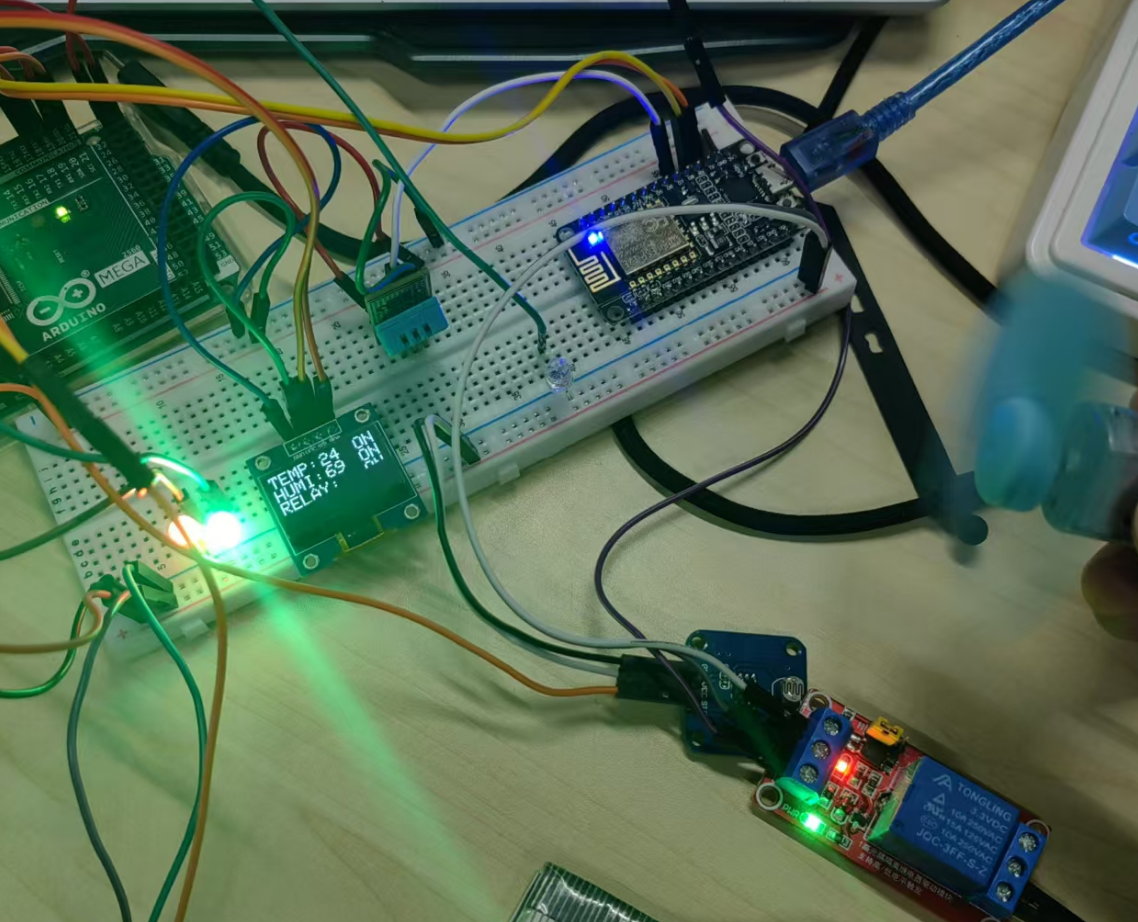
|  |
| --- |
| const char \*ssid = "Zzz"; // 热点名  const char \*password = "1234567890"; // 密码 |

成功连接后ESP8266会将AP分配给它的本地IPv4地址通过串口打印显示到串口监视器上(这里是192.168.43.94)，当服务器监听到有客户端后便会开始读取温湿度传感器数据，并将其打印到串口监视器并发送至Arduino Mega的串口1。



*串口打印IP地址、温湿度传感器数据*

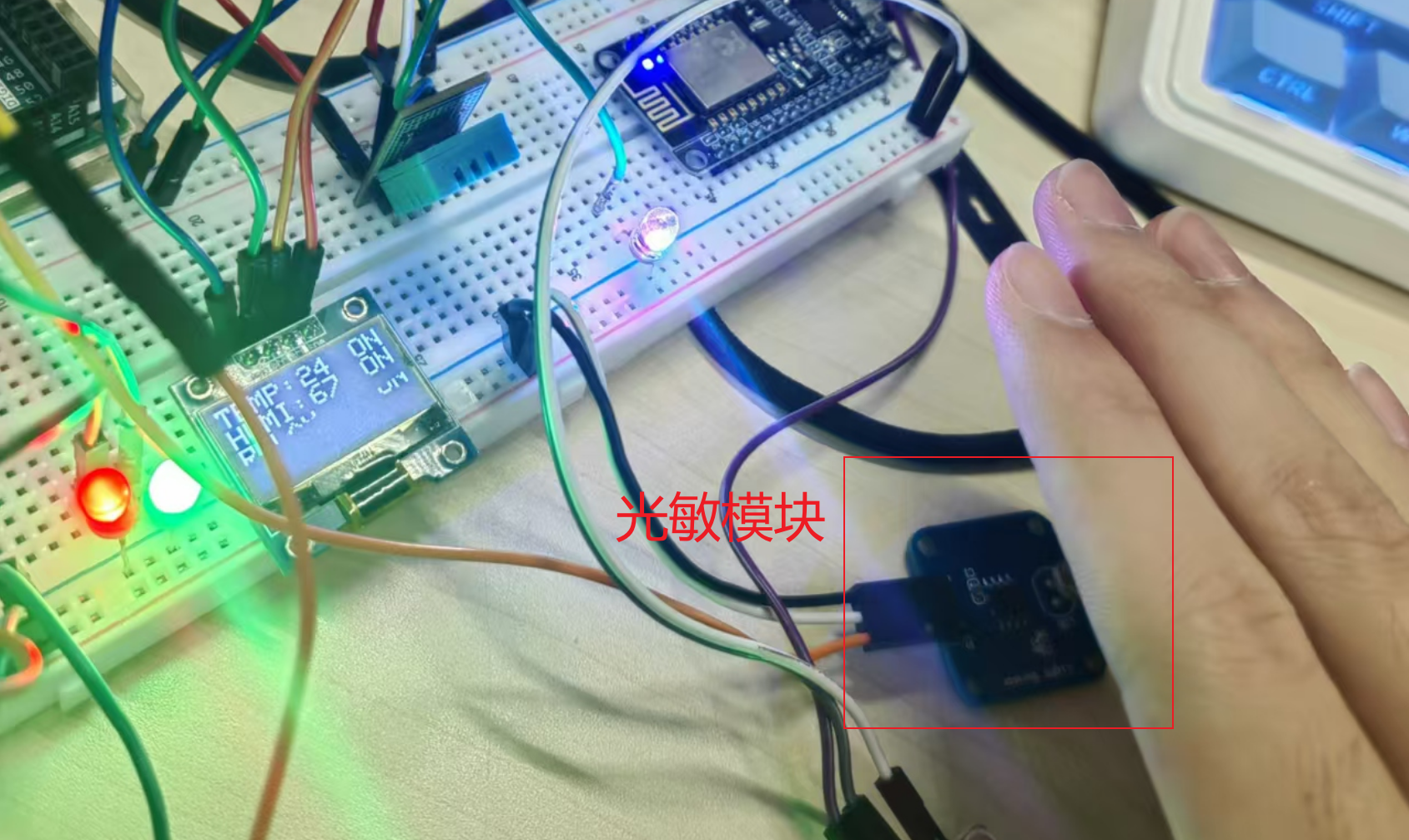
当Arduino Mega的串口1接收到的温湿度数据超过阈值后，会点亮对应的指示灯。此外，温度超过阈值后，会使得继电器输出端导通，进而控制直流电机工作进行通风散热；且会将温湿度数据、温湿度指示灯工作状态、继电器工作状态显示在OLED显示屏上，同时将继电器工作状态通过串口1回传给ESP8266。



*OLED实时显示温湿度数据 指示灯继电器工作状态*

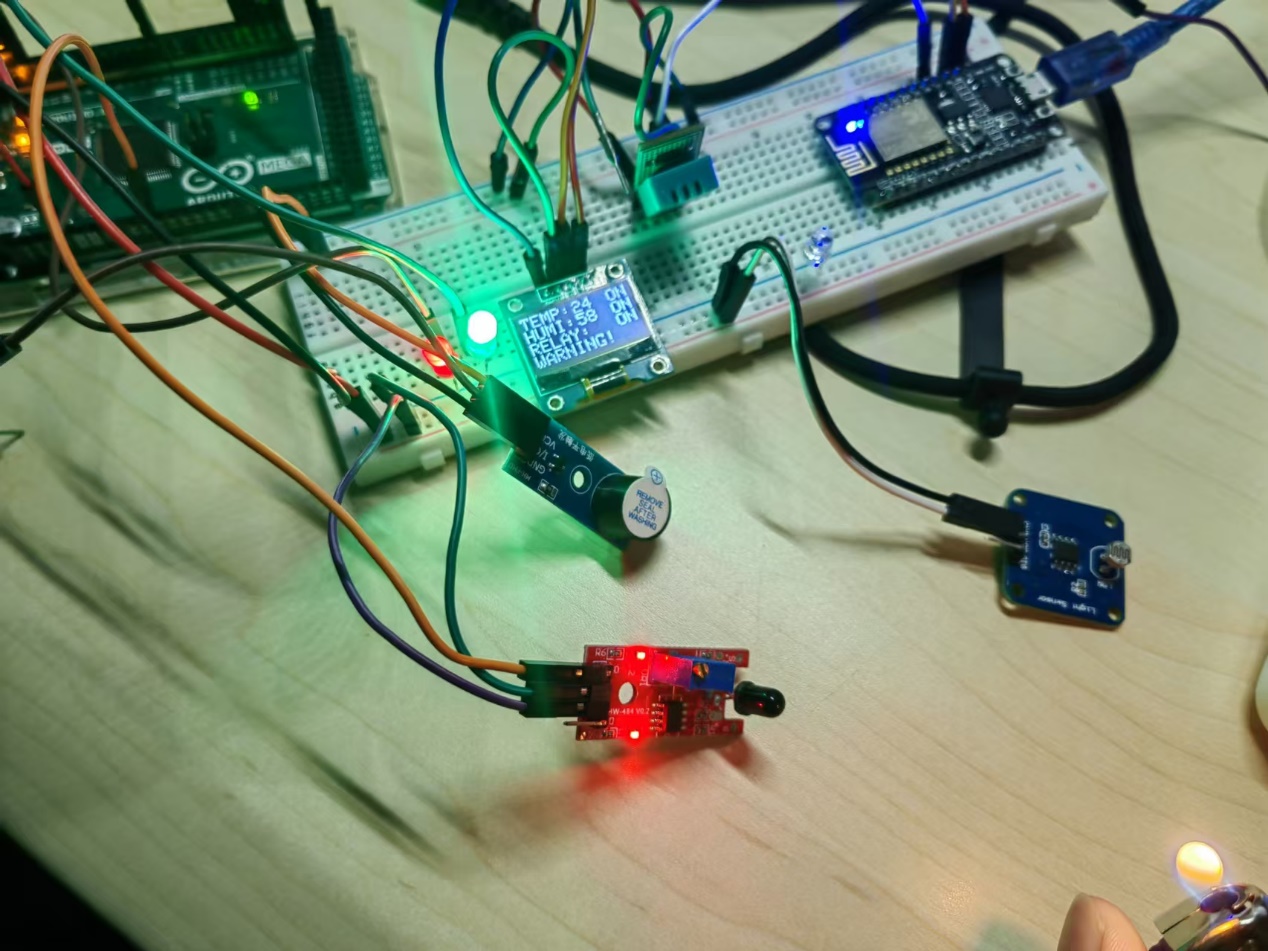
*温湿度超过阈值 指示灯亮 直流电机进行散热*

当光敏模块检测到环境温度变暗后，会根据昏暗程度调整PWM占空比进行不同亮度的补光。



*遮挡光敏模块的光敏电阻 遮挡程度不同补光灯亮度不同*

当火焰检测模块检测到火焰特有频率的光后，Arduino Mega会控制蜂鸣器报警，同时OLED显示屏会显示报警信息，并将消防情况信息回传给ESP8266处理。



*检测到火焰 蜂鸣器报警 OLED显示屏显示WARNING*

当ESP8266串口接收到来自Arduino Mega发送的数据后会进行处理，解算出关于继电器工作状态和工厂消防状态信息，并和温湿度信息一起实时显示在Web页面上。打开浏览器输入WiFi模块由AP分配得的IP地址(192.168.43.94)，可以访问到该前端页面，消防异常情况以醒目的红色警醒管理人员。



*Web端实时显示温湿度 继电器工作状态和消防情况*

**实验总结**

本实验完成了基本实验要求并联系实际应用进行拓展，设计了一个简单的工厂检测系统，通过ESP8266与Arduino串口双向通信，根据检测的工厂环境温湿度数据、光照情况和消防情况，控制温湿度指示灯、散热器(直流电机)、环境补光灯、消防报警器(蜂鸣器)工作，并实时显示在OLED显示屏和Web端上。

通过本次实验，复习了过去所学知识并将之应用于实际，加深了关于单片机、传感器、计算机网络相关知识的理解，还借此机会学习了前端页面设计相关知识。整个系统工作稳定符合预期，实验过程较为顺利，总体感觉良好。