درس مدارهای واسط

نیم سال ۰۴ -۰۵

استاد: دکترفصحتی

مبینا رشیدی 40170564 – مبینا حیدری 401105861

دانشکده مهندسی کامپیوتر

فاز سوم پروژه کد 6

عنوان پروژه : کنترل گلخانه با استفاده از سخت‌افزار در حلقه HIL و مقایسه پروتکل‌های ارتباطی

۱ .وضعیت سخت‌افزار و اتصالات پیاده‌سازی فیزیکی

تامین قطعات:

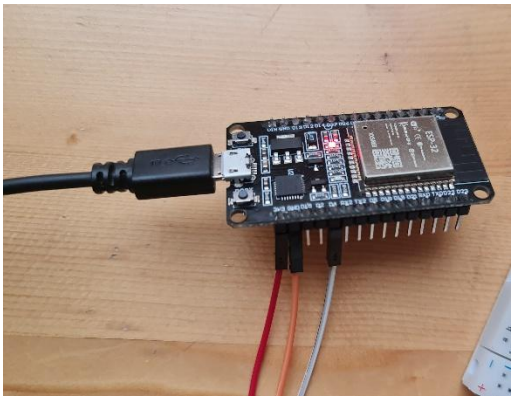
تمامی قطعات سخت‌افزاری تعریف شده در گام اول با موفقیت تهیه شده‌اند. لیست قطعات موجود به شرح زیر است:

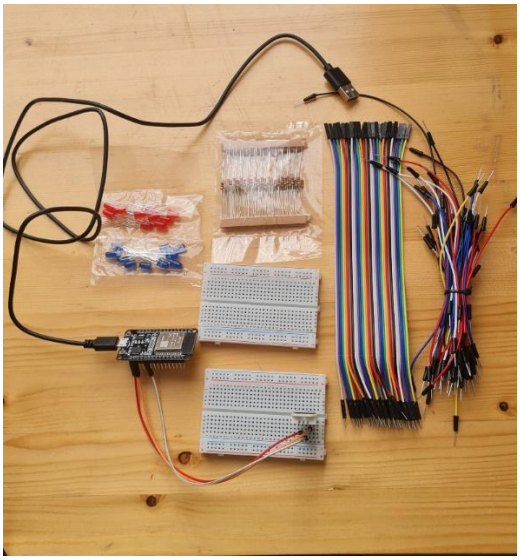
- **میکروکنترلر**: برد ESP32 مدل توسعه‌دهنده.
- **سنسور دما**: سنسور دیجیتال DHT22
- **عملگرها شبیه‌ساز**: دو عدد LED قرمز به عنوان هیتر، آبی به عنوان فن/خنک‌کننده.
- **ارتباطات**: کابل USB-to-Serial برای اتصال به کامپیوتر.
- **سایر**: برد بُردبوردBreadboard ، سیم‌های جامپر، مقاومت‌های محدودکننده جریان برایLED ها.

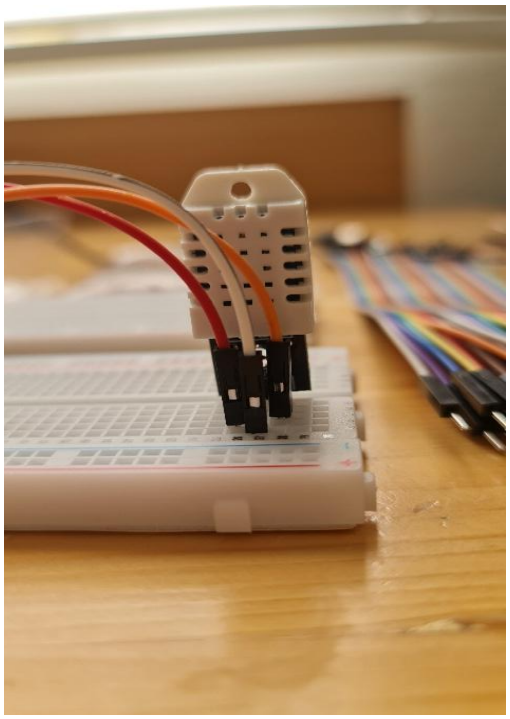
اسمبل مدار:

مدار سخت‌افزاری روی برد بُردبورد مونتاژ شده است. اتصالات به صورت زیر برقرار گردیده است:

- **سنسور:** DHT22 پین VDD به ۳,۳ ولت، پین GND به GND و پین Data به پین GPIO 4 esp32 متصل شده است.
- **ارتباط با کامپیوتر:** برد ESP32 از طریق پورت USB به کامپیوتر متصل شده تا ارتباط سریال UART برای ارسال داده‌های دما و دریافت فرمان‌های کنترل برقرار شود.







2. میکروکنترلر ESP32

این کد وظیفه دارد دما و رطوبت محیط را از سنسور DHT22 بخواند و آن‌ها را از طریق پورت سریال UART به نرم‌افزار Simulink ارسال کند. همچنین، فرمان‌های کنترلی روشن/خاموش کردن هیتر را از Simulink دریافت کرده و روی LED اعمال می‌کند.

- **تعاریف اولیه و تنظیمات Setup**

- **پین‌ها:** پین GPIO 4 برای ارتباط داده‌ای با سنسور و پین GPIO 2 برای کنترل LED هیتر تعریف شده‌اند.
- **پیکربندی سریال:** ارتباط با Baud rate 115200 برقرار می‌شود که سرعت مناسبی برای انتقال داده‌های بلادرنگ است.
- **حالت پین سنسور:** پین سنسور در ابتدا به صورت INPUT_PULLUP تنظیم شده تا نویزهای محیطی خنثی شوند.

- حلقه اصلی Loop و مدیریت ارتباط

در حلقه loop، برنامه به صورت دوره‌ای هر ۵ ثانیه دو کار اصلی انجام می‌دهد:

دریافت فرمان از Simulink:

- تابع Serial.available چک می‌کند آیا داده‌ای از کامپیوتر سیمولینک ارسال شده است یا خیر.
- اگر داده‌ای باشد یک کاراکتر، آن را می‌خواند.
- اگر کاراکترH باشد، یعنی فرمان "گرم کردن" صادر شده و LED هیتر روشن می‌شود HIGH. برای هر کاراکتر دیگری مثل C برای Cool، هیتر خاموش می‌شود LOW.

خواندن سنسور و ارسال داده:

- تابع readDHT فراخوانی می‌شود. اگر این تابع موفقیت‌آمیز باشد یعنی چک‌سام درست باشد، داده‌ها فرمت‌بندی شده و ارسال می‌شوند.
- **فرمت ارسال‌داده:** داده‌ها بین علامت < و > قرار می‌گیرند تا سیمولینک راحت‌تر بتواند شروع و پایان پیام را تشخیص دهد مثال: <25.5,60.2>.
- اگر خواندن سنسور با خطا مواجه شود، کلمه ERROR ارسال می‌شود.

۳ .پیاده‌سازی دستی پروتکل DHT22 تابعreadDHT

این بخش یکی از چالش‌برانگیزترین بخش‌های پیاده‌سازی نرم‌افزار بود.

برای غلبه بر این مشکل و پیشرفت پروژه بدون توقف، تصمیم گرفته شد پروتکل ارتباطی سنسور DHT22 به صورت "دستی Bit-banging" و بدون اتکا به کتابخانه‌های آماده پیاده‌سازی شود. در این روش، مستقیماً با استفاده از دستورات سطح پایین میکروکنترلر و تایمر دقیقESP32 ، سیگنال‌های الکتریکی روی پین داده کنترل و تحلیل می‌شوند.

مراحل اجرای این روش به شرح زیر است:

- **سیگنال شروع Start Signal**: میکروکنترلر پین داده را به مدت 1 میلی ثانیه در حالت LOW نگه‌می‌دارد تا سنسور را بیدار کند.
- **دریافت پاسخ و داده‌ها**: سنسور پس از پاسخ دادن، ۴۰ بیت داده ۵ بایت شامل دما و رطوبت را ارسال می‌کند.
- **تحلیل پالس‌ها**: برای تشخیص بیت‌های "۰" و "۱"، از تابع esp_timer_get_time استفاده شد تا طول پالس‌های میکروثانیه‌ای با دقت بالا اندازه‌گیری شود.
    1. اگر پالس High حدود 26 میکروثانیه باشد -> بیت 0
    2. اگر پالس High حدود 70 میکروثانیه باشد -> بیت 1
- **محاسبه چک‌سام**: برای اطمینان از صحت داده‌ها در نبود کتابخانه، الگوریتم محاسبه Checksum به صورت دستی کدنویسی شد تا داده‌های خراب شناسایی شوند.

۴ .تابع کمکی waitForPinState

این تابع برای صبر کردن است. مثلاً وقتی منتظریم سنسور خط را پایین بکشد، این تابع تا ۱۰۰ میکروثانیه صبر می‌کند. اگر در این زمان پین وضعیت مورد نظر را نگیرد، false برمی‌گرداند تا برنامه در یک حلقه بی‌نهایت گیر نکند.

خروجی در serial monitor

5 .پردازش داده‌ها در سیمولینک بلوک‌های MATLAB Function

برای مدیریت تبادل داده بین سیمولینک و میکروکنترلر، دو بلوک تابع MATLAB Function در مدل طراحی شده است:

1. **تابع اول Parse_Data**: پردازش داده دریافتی از ESP32

این تابع وظیفه دارد رشته‌ی متنی خامی که از پورت سریال UART می‌آید مثلاً <25.5,60.2>را بخواند و آن را به دو عدد قابل استفاده در سیمولینک دمای temp و رطوبت humتبدیل کند.

مراحل کاری:

- **تبدیل به رشته char**: ورودی که از بلوک Serial Receive می‌آید ممکن است فرمت بایت یا اعداد باشد. ابتدا آن را به یک رشته متنی تبدیل می‌کند تا بتوانیم کاراکترهای < و > را در آن پیدا کنیم.
- **استخراج محتوا strfind**: با استفاده از strfind محل علامت شروع < و پایان > را پیدا می‌کند. سپس متنی که بین این دو علامت قرار دارد یعنی 25.5,60.2 را جدا می‌کند.
- **جداسازی دما و رطوبت:**
  - محل ویرگول , را پیدا می‌کند.
  - قسمت قبل از ویرگول را به عنوان دما t_str و قسمت بعد از ویرگول را به عنوان رطوبت h_str جدا می‌کند.
- **تبدیل رشته به عدد تابع parseSimple**:

نکته فنی :در ابتدا تلاش شد از تابع استاندارد str2double برای تبدیل رشته به عدد استفاده شود، اما این تابع در هنگام کامپایل مدل، خروجی از نوع complex تولید می‌کرد و باعث بروز خطا در سیمولینک می‌شد. برای رفع این مشکل، یک الگوریتم تبدیل دستی تابع parseSimple نوشته شد که کاراکتر به کاراکتر رشته را می‌خواند و آن را به عدد اعشاری تبدیل می‌کند. این روش مشکل نوع داده را حل کرده و پایداری مدل را تضمین کرد.

منطق تابع: این تابع کاراکتر به کاراکتر رشته را می‌خواند. اگر کاراکتر نقطه . باشد، می‌فهمد که عدد اعشاری شده و برای ارقام بعدی تقسیم بر ۱۰، ۱۰۰ و ... انجام می‌دهد. در غیر این صورت، عدد را با ضرب در ۱۰ جابجا می‌کند مثل تبدیل "۲۵" به عدد ۲۵.

```
function [temp, hum] = fcn(data_str)
    % تعریف خروجی‌ها
    temp = 0.0;
    hum = 0.0;

    % تبدیل ورودی به رشته
    str = char(data_str(:)');

    % پیدا کردن محل < و >
    start_idx = strfind(str, '<');
    end_idx = strfind(str, '>');

    % اگر هر دو پیدا شدند
    if ~isempty(start_idx) && ~isempty(end_idx)
        s = start_idx(1);
        e = end_idx(1);

        % جدا کردن متن بین علامت‌ها
        if e > s
            content = str(s+1 : e-1);

            % پیدا کردن ویرگول
            comma = strfind(content, ',');

            if ~isempty(comma)
                c = comma(1);

                % جدا کردن دما و رطوبت
                t_str = content(1:c-1);
                h_str = content(c+1:end);

                % --- تبدیل دستی به عدد (بدون str2double) ---
                temp = parseSimple(t_str);
                hum = parseSimple(h_str);
            end
        end
    end
end

% تابع ساده برای تبدیل رشته به عدد
function num = parseSimple(str_val)
    num = 0;
    decimal_found = false;
    divisor = 10.0;

    for k = 1:length(str_val)
        ch = str_val(k);

        if ch == '.'
            decimal_found = true;
            continue;
        end

        % تبدیل کاراکتر به عدد (کد اسکی منهای 48)
        digit = double(ch) - 48;

        if decimal_found
            num = num + digit / divisor;
            divisor = divisor * 10.0;
        else
            num = num * 10.0 + digit;
        end
    end
end
```

2. **تابع دوم Generate_Command**: تبدیل فرمان کنترلی به کاراکتر

این تابع در سمت خروجی سیمولینک قرار دارد. ورودی آن سیگنال کنترلی است که از کنترلر PID می‌آید و معمولاً ۰ یا ۱ است و خروجی آن داده‌ای است که باید به پورت سریال فرستاده شود.

مراحل کاری:

- **تعریف نوع داده unit8**: خروجی را به صورت عدد صحیح ۸ بیتی بدون علامت تعریف می‌کند. این دقیقاً فرمت داده‌ای است که پورت سریال برای ارسال یک کاراکتر نیاز دارد.
- **شرط کنترل**: اگر control_signal برابر با 1 باشد یعنی کنترلر گفته هیتر را روشن کن، خروجی را برابر با کد اسکی کاراکتر H قرار می‌دهد.در غیر این صورت یعنی ۰ یا هر مقدار دیگری، خروجی را برابر با کد اسکی کاراکتر C برای Cool یا خاموش قرار می‌دهد.
- **نتیجه**: خروجی این بلوک یک عدد است مثلاً ۷۲ برای H که وقتی به بلوک Serial Send داده می‌شود، میکروکنترلر آن را به صورت کاراکتر H دریافت می‌کند و LED را روشن می‌کند.

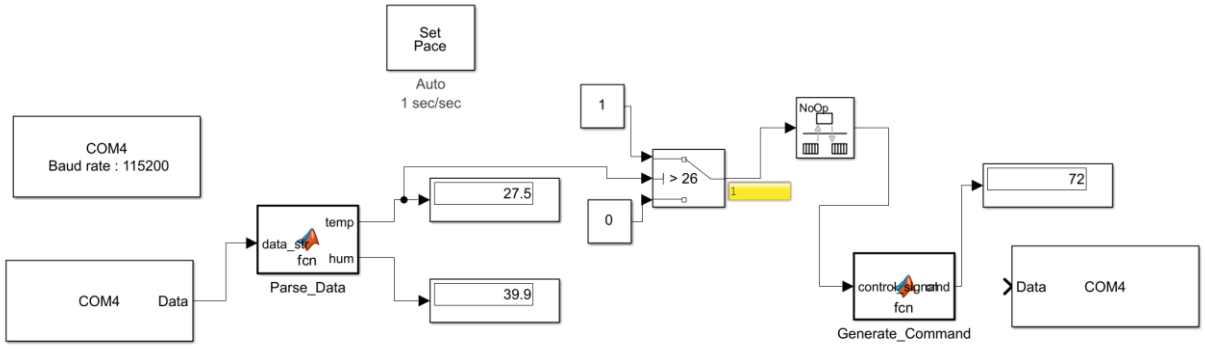
```
function cmd = fcn(control_signal)
    % تعریف نوع خروجی به صورت عددی ۸ بیتی بدون علامت
    cmd = uint8(0);

    if control_signal == 1
        cmd = uint8('H'); % به صورت عدد H کد اسکی کاراکتر
    else
        cmd = uint8('C'); % به صورت عدد C کد اسکی کاراکتر
    end
end
```

6. تست ارتباطات Communication - UART

ارتباط سریال UART بین سیمولینک و سخت‌افزار با موفقیت برقرار و تست شده است.

- **پیکربندی سیمولینک**: بلوک‌های Serial Configuration، Serial Receive و Serial Send در مدل Simulink/MATLAB اضافه شدند. پورت COM و Baud Rate ۱۱۵۲۰۰ تنظیم گردید.
- **عملکرد**: داده‌های دمای واقعی محیط توسط سنسور DHT22 خوانده شده، به سیمولینک ارسال می‌شود همچنین فرمان‌های صادر شده از کنترلر PID در سیمولینک مانند فرمان روشن کردن هیتر توسط ESP32 دریافت شده و LED مربوطه روشن می‌شود.
- **نتیجه**: تأخیر در ارتباط UART ناچیز بوده و داده‌ها به صورت بلادرنگ Real-time مبادله می‌شوند.



همانطور که در شکل بالا نشان داده شده دما و رطوبت توسط dht22 خوانده شده و ارسال شده در تابع parse_Data به دو مقدار دما و رطوبت جدا می شود مقدار دما در یک بلاک switch وارد میشود اگر بیشتر از یک مقدار خاصی باشد دستور 1 ارسال در غیر این صورت 0 ارسال میشود در تابع generate_command با گرفتن مقدار 1 نیز عدد 72 که کد اسکی H می باشدارسال میشود با دادن این فرمان به esp32 میتوان led را روشن کرد.

گام سوم پروژه

۱ .مقدمه و تغییر استراتژی نسبت به فاز قبل

در فاز اولیه پروژه، تمرکز اصلی بر روی "داده‌برداری (Data Logging) "بود و پارامترهای دما و رطوبت صرفاً از طریق پروتکل سریال (UART) مانیتور می‌شدند .در این گام، به منظور پیاده‌سازی یک سیستم کنترل **Hardware-in-the-Loop (HIL)** واقعی و بررسی دینامیک سیستم، تغییرات زیر در ساختار پروژه اعمال گردید:

1. **تمرکز تک‌متغیره :**جهت جلوگیری از پیچیدگی محاسباتی و تمرکز بر پایداری حلقه کنترل، پارامتر رطوبت از چرخه کنترلی حذف و تمام تمرکز سیستم بر روی **تنظیم دقیق دما** معطوف شد.
2. **ارتقای بستر ارتباطی :**ارتباط سیستم از حالت صرفاً سریال، به ساختار ترکیبی **Serial + TCP/IP (Wi-Fi)**ارتقا یافت تا اثر تأخیر شبکه بر سیستم کنترل بررسی شود.

3. **افزودن مدل مجازی(Virtual Plant) :** یک مدل ریاضی از گلخانه در سیمولینک طراحی شد تا رفتار ایده‌آل سیستم در کنار رفتار واقعی سخت‌افزار قابل مقایسه باشد.

۲ .تشریح ساختار پیاده‌سازی شده

الف) مدل‌سازی گلخانه مجازی(Virtual Greenhouse Model)

به منظور داشتن یک معیار مرجع (Reference) برای تست الگوریتم کنترلیPID ، رفتار حرارتی گلخانه به صورت یک سیستم مرتبه اول مدل‌سازی شد. این مدل در سیمولینک با تابع تبدیل زیر پیاده‌سازی گردیده است:

مدل‌سازی فیزیکی – (Digital Twin)

- **معادله دیفرانسیل**: رفتار سیستم بر اساس تابع تبدیل مدل شده است.

$$\frac{5}{20s+1}$$

  - **عدد ۵ (Gain):** نشان‌دهنده قدرت هیتر است. یعنی هیتر توانایی دارد دما را تا ۵ درجه نسبت به محیط افزایش دهد.
  - **عدد ۲۰ (Time Constant):** نشان‌دهنده اینرسی حرارتی یا ظرفیت گرمایی محیط است. یعنی ۲۰ ثانیه طول می‌کشد تا سیستم به ٪۶۳ تغییر دمای نهایی برسد (شبیه‌سازی کند گرم شدن محیط واقعی).
- **دمای پایه :**خروجی مدل با عدد ثابت ۲۵ جمع می‌شود که نشان‌دهنده دمای محیط (Ambient Temperature) است.
- **حلقه بازخورد (Feedback Loop):** برای جلوگیری از خطای Algebraic Loop در سیمولینک، از بلوک‌های تاخیر (Unit Delay یا 1/z) در مسیر برگشت سیگنال استفاده شده است.

ب) دریافت داده‌های واقعی(Real-World Feedback)

میکروکنترلر ESP32 به عنوان یک سرور TCP پیکربندی شده است که دمای محیط را از سنسور DHT22 قرائت کرده و به صورت بسته‌های داده (Data Packets) از طریق شبکه Wi-Fi به نرم‌افزار متلب ارسال می‌کند. همزمان، جهت دیباگ و اطمینان از صحت داده‌ها، این مقادیر بر روی پورت سریال نیز ارسال می‌شوند.

مکانیزم انتخاب منبع – (HIL Switch)

یک بلوک Manual Switch در مرکز دیاگرام قرار دارد که قابلیت حیاتی HIL را فراهم می‌کند:

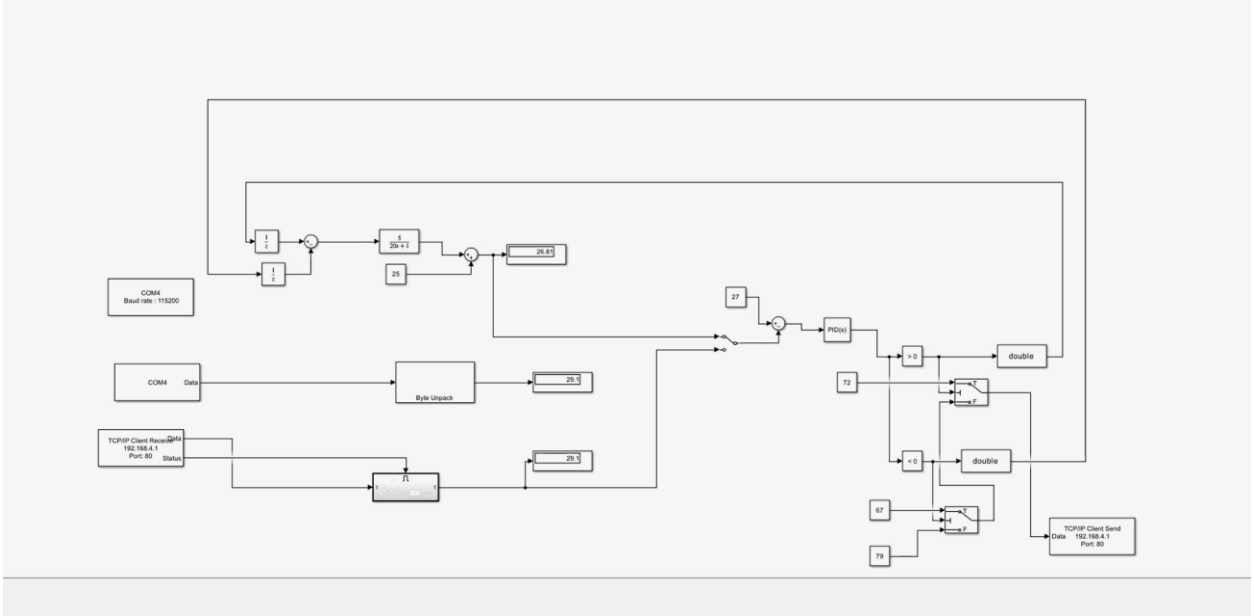
- **حالت ۱ (بالا):** ورودی کنترلر از "مدل ریاضی مجازی" گرفته می‌شود. (تست ایمن و تئوری).
- **حالت ۲ (پایین):** ورودی کنترلر از بلوک TCP/IP Receive گرفته می‌شود که دمای واقعی سنسور محیط را نشان می‌دهد.

کنترل‌کننده PID و منطق تصمیم‌گیری

خروجی سوئیچ وارد بلوک Discrete PID می‌شود. خروجی PID یک عدد بین ۱- تا ۱+ است که باید به فرمان‌های دیجیتال تبدیل شود:

- **اگر خروجی PID بزرگتر از صفر باشد (0>):** فرمان به بلوک هیتر می‌رود.
- **اگر خروجی PID کوچکتر از صفر باشد (0<):** فرمان به بلوک کولر می‌رود.

در نهایت، فرمان‌ها از طریق بلوک TCP/IP Send به میکروکنترلر ارسال می‌شوند تا LED ها (محرک‌ها) را روشن یا خاموش کنند.



همچنین جهت مشاهده عملکرد واقعی سیستم و هماهنگی بین مدل سیمولینک و سخت‌افزار، **فیلم اجرای پروژه** به پیوست این گزارش ارائه شده است.

3. معماری نرم‌افزار

الف) تنظیمات اولیه و کتابخانه‌ها

در ابتدای کد، کتابخانه‌های WiFi.h برای ارتباط شبکه و DHT.h برای سنسور دما فراخوانی شده‌اند. پین‌های کنترلی به صورت زیر تعریف شده‌اند:

- پین ۴: سنسور دما (DHT22)
- پین ۵ LED: قرمز (شبیه‌ساز هیتر)
- پین ۱۸ LED: آبی (شبیه‌ساز فن/کولر)

ب) پروتکل ارسال داده(Data Transmission)

برای ارسال دمای اعشاری (Float) به سیمولینک، از یک روش کارآمد استفاده شده است. به جای ارسال رشته متنی(String) ، بایت‌های خام متغیر float ارسال می‌شوند:

1. **هدر (Header):** کاراکتر A برای شروع بسته.
2. **بدنه (Payload):** ۴ بایت مربوط به متغیر دما.((uint8_t*)&temp)
3. **فوتر (Footer):** کاراکتر Z برای پایان بسته .این روش سرعت انتقال را بالا برده و پردازش در متلب را آسان می‌کند.

ج) دریافت فرمان و کنترل محرک‌ها(Actuators)

تابع controlActuators وظیفه دارد بر اساس دستور دریافتی از سیمولینک، وضعیت محیط را تغییر دهد:

- **دریافت کاراکتر 'H'**: روشن کردن هیتر.(Red LED High)
- **دریافت کاراکتر 'C'**: روشن کردن کولر.(Blue LED High)
- **دریافت کاراکتر '0'**: خاموش کردن همه سیستم‌ها.(Idle)

project.ino

```ino
#include <DHT.h>
#include <WiFi.h>

#define DHTPIN 4
#define DHTTYPE DHT22
DHT dht(DHTPIN, DHTTYPE);

#define RED_LED 5
#define BLUE_LED 18

const char* ssid = "HIL_Project_WiFi";
const char* password = "12345678";

WiFiServer server(80);

unsigned long previousMillis = 0;
const long interval = 2000;

void setup() {
  Serial.begin(115200);

  dht.begin();
  pinMode(RED_LED, OUTPUT);
  pinMode(BLUE_LED, OUTPUT);

  WiFi.softAP(ssid, password);

  IPAddress myIP = WiFi.softAPIP();

  Serial.print("AP IP address: ");
  Serial.println(myIP);

  server.begin();
}

void loop() {
  WiFiClient client = server.available();

  if (client) {
    while (client.connected()) {

      unsigned long currentMillis = millis();

      if (currentMillis - previousMillis >= interval) {
        previousMillis = currentMillis;

        float temp = dht.readTemperature();
        if (isnan(temp)) temp = 0.0;

        client.write('A');
        client.write((uint8_t*)&temp, 4);
        client.write('Z');

        Serial.write('A');
        Serial.write((uint8_t*)&temp, 4);
        Serial.write('Z');
      }

      if (client.available()) {
        char command = client.read();
        if (command == 'H' || command == 'C' || command == 'O') {
          controlActuators(command);
        }
      }

      delay(5);
    }

    client.stop();
  }
}

void controlActuators(char cmd) {
  if (cmd == 'H') {
    digitalWrite(RED_LED, HIGH);
    digitalWrite(BLUE_LED, LOW);
  } else if (cmd == 'C') {
    digitalWrite(RED_LED, LOW);
    digitalWrite(BLUE_LED, HIGH);
  } else if (cmd == 'O') {
    digitalWrite(RED_LED, LOW);
    digitalWrite(BLUE_LED, LOW);
  }
}

```

د) مدار بسته شده

در تصویر زیر مدار بسته شده را مشاهده می‌کنید: