To: Christopher Peters
From: Abdullah Khalid

Purpose

The goal of this assignment was to create a system that reads sensor data on an Arduino Uno R4 WiFi board and wirelessly communicates with a Node-RED interface and a Python client. Based on the server response, the board dynamically adjusts LEDs, creating a closed-loop wireless control system.

Methodology / Approach

Part A involved setting up sensors (GY-87 IMU and HC-SR04 distance sensor) and LEDs on the Arduino and printing sensor data to the serial monitor. Part B integrated Node-RED to receive sensor data via HTTP and respond with LED values through a GUI interface. The Arduino sent a JSON payload every 10 seconds, and Node-RED parsed and returned updated LED control values. Part C replaced Node-RED with a Python client, which mimicked the same JSON exchange: sending data and controlling LEDs based on user-defined logic. Both the Arduino and Node-RED/Python used a local Wi-Fi network for communication, and JSON was the primary data format. Key libraries used included WiFiS3.h and ArduinoJson.h for the Arduino, and requests for the Python client.

Results

The wireless system was successful overall. All the functions worked as expected and any communication on the client was sent to the server. Challenges were mostly related to WiFi configurations at Drexel because the dragonfly3 WiFi was not working with the Arduino. Mobile Hotspot was used as the connection.

Appendix

```
Part A:

#include <WiFiS3.h>
#include <Wire.h>
```

```
#include <Adafruit_MPU6050.h>
#include <Adafruit_BMP085.h>
#include <QMC5883LCompass.h>

#define RED_LED_PIN 7
#define BLUE_LED_PIN 6
#define TRIG_PIN 10
#define ECHO_PIN 9

Adafruit_MPU6050 mpu;
Adafruit_BMP085 bmp;
QMC5883LCompass compass;

char ssid[] = "Abdullah Khalid";
char pass[] = "QSPS2992@is2992";

WiFiServer server(80);

void setup() {
  Serial.begin(9600);
  while (!Serial);

  pinMode(RED_LED_PIN, OUTPUT);
  pinMode(BLUE_LED_PIN, OUTPUT);
  pinMode(TRIG_PIN, OUTPUT);
  pinMode(ECHO_PIN, INPUT);

  Wire.begin();
  mpu.begin();
  bmp.begin();
  compass.init();

  WiFi.begin(ssid, pass);
  while (WiFi.status() != WL_CONNECTED) {
  Serial.print("Trying... Status = ");
  Serial.println(WiFi.status());
  delay(1000);
}

  Serial.println("\nWiFi connected!");
  Serial.print("IP address: ");
  Serial.println(WiFi.localIP());

  server.begin();
}

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

```
  if (client) {
    Serial.println("Client connected");
    String req = client.readStringUntil('\r');
    Serial.println(req);
    client.flush();

    int redVal = getValue(req, "red").toInt();
    int blueVal = getValue(req, "blue").toInt();

    analogWrite(RED_LED_PIN, constrain(redVal, 0, 255));
    digitalWrite(BLUE_LED_PIN, blueVal > 0 ? HIGH : LOW);

    String json = getSensorData();
    client.println("HTTP/1.1 200 OK");
    client.println("Content-Type: application/json");
    client.println("Connection: close");
    client.println();
    client.println(json);

    delay(10);
    client.stop();
    Serial.println("Client disconnected");
  }
}

String getValue(String req, String key) {
  int start = req.indexOf(key + "=");
  if (start == -1) return "";
  int end = req.indexOf("&", start);
  if (end == -1) end = req.length();
  return req.substring(start + key.length() + 1, end);
}

String getSensorData() {
  sensors_event_t a, g, t;
  mpu.getEvent(&a, &g, &t);
  compass.read();
  float temp = bmp.readTemperature();
  float pressure = bmp.readPressure();
  float altitude = bmp.readAltitude();
  float distance = readDistanceCM();
  int heading = compass.getAzimuth();
  char dir[4]; compass.getDirection(dir, heading);

  String json = "{";
  json += "\"temp\":" + String(temp, 1) + ",";
  json += "\"pressure\":" + String(pressure, 0) + ",";
  json += "\"altitude\":" + String(altitude, 1) + ",";
```

```
  json += "\"distance\":" + String(distance, 1) + ",";
  json += "\"heading\":" + String(heading) + ",";
  json += "\"direction\":\"" + String(dir) + "\"";
  json += "}";
  return json;
}

float readDistanceCM() {
  digitalWrite(TRIG_PIN, LOW);
  delayMicroseconds(2);
  digitalWrite(TRIG_PIN, HIGH);
  delayMicroseconds(10);
  digitalWrite(TRIG_PIN, LOW);
  long duration = pulseIn(ECHO_PIN, HIGH);
  return duration * 0.0343 / 2;
}
```

Part B:

```
#include <WiFiS3.h>
#include <ArduinoJson.h>

char ssid[] = "Abdullah Khalid";
char pass[] = "QSPS2992@is2992";

const int redPin = 5;
const int bluePin = 4;

void setup() {
  Serial.begin(9600);

  pinMode(LED_BUILTIN, OUTPUT);
  pinMode(redPin, OUTPUT);
  pinMode(bluePin, OUTPUT);

  WiFi.begin(ssid, pass);
  Serial.print("Connecting to WiFi");
  while (WiFi.status() != WL_CONNECTED) {
    delay(1000);
    Serial.print(".");
  }

  Serial.println("\nWiFi connected!");
  Serial.print("IP Address: ");
```

```
  Serial.println(WiFi.localIP());
}

void loop() {
  if (WiFi.status() != WL_CONNECTED) {
    WiFi.begin(ssid, pass);
    delay(3000);
    return;
  }

  WiFiClient client;

  if (client.connect("172.20.10.5", 1880)) {
    StaticJsonDocument<256> jsonDoc;
    jsonDoc["temp"] = 20.3;
    jsonDoc["pressure"] = 101980;
    jsonDoc["distance"] = 15.0;

    String jsonStr;
    serializeJson(jsonDoc, jsonStr);


    client.println("POST /sensor HTTP/1.1");
    client.println("Host: 172.20.10.5");
    client.println("Content-Type: application/json");
    client.print("Content-Length: ");
    client.println(jsonStr.length());
    client.println();
    client.println(jsonStr);

    // Wait for server response
    unsigned long timeout = millis();
    while (client.available() == 0) {
      if (millis() - timeout > 3000) {
        Serial.println("Client Timeout");
        client.stop();
        return;
      }
    }


    String response = "";
    while (client.available()) {
      response += client.readStringUntil('\r');
    }

    Serial.println("📩 Response:");
    Serial.println(response);
```

```
    int jsonStart = response.indexOf('{');
    if (jsonStart != -1) {
      String jsonPart = response.substring(jsonStart);
      StaticJsonDocument<128> ledDoc;
      DeserializationError error = deserializeJson(ledDoc, jsonPart);
      if (!error) {
        int redVal = ledDoc["red"];
        int blueVal = ledDoc["blue"];

        analogWrite(redPin, redVal);
        digitalWrite(bluePin, blueVal);
      } else {
        Serial.println("JSON parse error");
      }
    }

    client.stop();
  } else {
    Serial.println(“Connection failed.");
  }

  delay(10000);
}
```

```
Part C:

#include <WiFiS3.h>
#include <WiFiServer.h>
#include <ArduinoJson.h>

char ssid[] = "Abdullah Khalid";       // Your WiFi
char pass[] = "QSPS2992@is2992";       // Your password

WiFiServer server(80);

const int redPin = 5;
const int bluePin = 4;

void setup() {
  Serial.begin(9600);
  pinMode(redPin, OUTPUT);
  pinMode(bluePin, OUTPUT);
```

```
  WiFi.begin(ssid, pass);
  Serial.print("Connecting to WiFi");
  while (WiFi.status() != WL_CONNECTED) {
    delay(1000);
    Serial.print(".");
  }

  Serial.println("\nWiFi connected!");
  Serial.print("📡 IP address: ");
  Serial.println(WiFi.localIP());

  server.begin();
}

void loop() {
  WiFiClient client = server.available();
  if (client) {
    Serial.println("Client connected");

    String request = "";
    while (client.connected()) {
      if (client.available()) {
        request = client.readStringUntil('\n');
        if (request == "\r") break;
      }
    }


    String body = "";
    while (client.available()) {
      body += char(client.read());
    }

    Serial.println("📩 Body: " + body);


    StaticJsonDocument<128> doc;
    DeserializationError error = deserializeJson(doc, body);
    if (!error) {
      int redVal = doc["red"];
      int blueVal = doc["blue"];

      analogWrite(redPin, redVal);
      digitalWrite(bluePin, blueVal);

      Serial.print("Red: "); Serial.println(redVal);
      Serial.print("Blue: "); Serial.println(blueVal);
```

```
    } else {
      Serial.println("JSON parse error");
    }


    client.println("HTTP/1.1 200 OK");
    client.println("Content-Type: text/plain");
    client.println("Connection: close");
    client.println();
    client.println("OK");

    delay(100);
    client.stop();
    Serial.println("Client disconnected");
  }
}
```

Python for Part C:

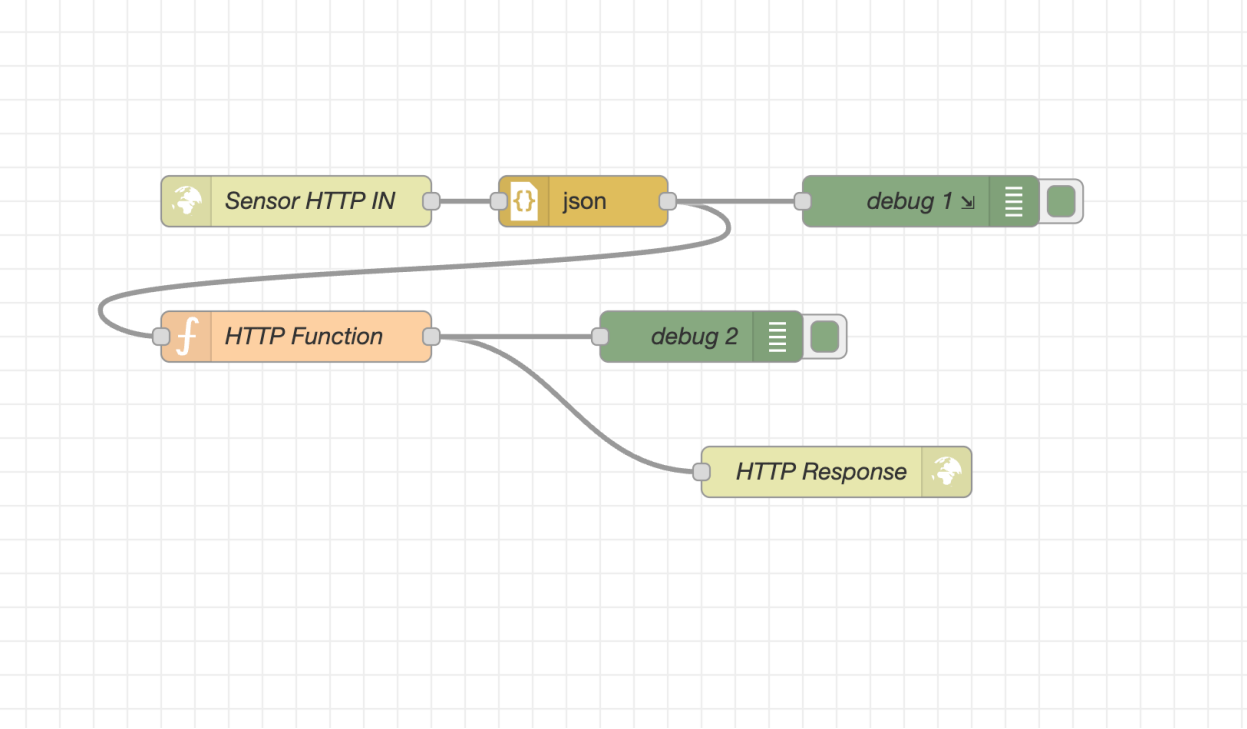
```
import requests
import time

url = "http://172.20.10.7"  # Arduino's IP

payload = {
    "red": 128,
    "blue": 1
}

while True:
    try:
        response = requests.post(url, json=payload, timeout=5)
        print(f"Sent data: {payload}")
        print(f"Response: {response.text}")
    except Exception as e:
        print("Error:", e)

    time.sleep(10)
```


Sensor HTTP IN
json
debug 1
HTTP Function
debug 2
HTTP Response