UIT2412 DIGITAL SYSTEMS AND MICROPROCESSOR LAB

Mini Project Report

Project Title: Arduino-based Climate Monitoring and Reporting Device (Climard)

Team Members:

1. Shanjay Athithya.G - 3122225002124
2. P.L. Singaram - 3122225002130

1. Introduction

1.1 Overview of Climate Monitoring Systems

Climate monitoring systems are essential tools designed to measure, record, and analyze various environmental parameters such as temperature, humidity, and light intensity. These systems utilize sensors to collect real-time data, which can then be processed and displayed for monitoring purposes. The collected data can also be sent to remote servers for further analysis and reporting.

1.2 Importance and Applications

Climate monitoring is critical in various fields due to its ability to provide accurate and timely information about environmental conditions. Key applications include:

- Agriculture: Monitoring soil and atmospheric conditions to optimize crop production.
- Weather Stations: Providing real-time weather data for forecasting and research.
- Building Automation: Managing HVAC systems to maintain optimal indoor climates.
- Research and Education: Collecting data for scientific studies and educational purposes.

The growing need for real-time environmental monitoring and data-driven decision-making highlights the significance of robust climate monitoring systems.

1.3 Objective of the Report

The primary goal of this report is to provide a comprehensive analysis of the design, development, and implementation of a climate monitoring device using an Arduino microcontroller. The specific objectives include:

- Design and Architecture: Detailing the overall design and architecture of the device, including the selection and integration of hardware components such as sensors and the Arduino board.
- Software Development: Describing the software development process, including the programming environment, algorithm design, and code implementation.
- Performance Evaluation: Evaluating the performance of the device through various tests and experiments, analyzing its ability to accurately measure and report environmental parameters.
- Challenges and Solutions: Discussing the challenges encountered during the development process and the solutions implemented to overcome them.

- Future Enhancements: Exploring potential improvements and future directions for enhancing the device's capabilities and expanding its applications.

This report aims to serve as a guide for anyone interested in understanding the fundamentals of climate monitoring technology and developing similar systems using Arduino.

2. System Design

Requirement Specifications

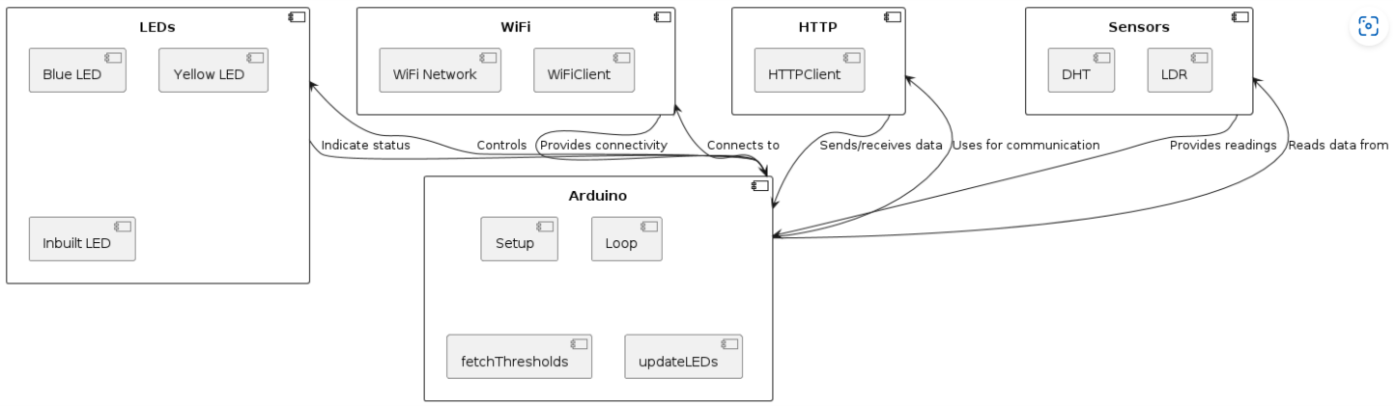
Hardware Requirements:

1. Arduino UNO
2. DHT11 Temperature and Humidity Sensor
3. LDR (Light Dependent Resistor)
4. 4 x LEDs (Red, Blue, Yellow, Green)
5. Breadboard
6. Jumper wires (male-to-male, male-to-female)
7. Resistors (10kΩ, 220Ω)
8. USB Cable for Arduino
9. Power Supply (Battery or Adapter)

Software Requirements:

1. Arduino IDE
2. Arduino drivers

Block diagram:



3. Wirings:

DHT11 Sensor:

- VCC → Arduino 5V
- GND → Arduino GND
- Data → Arduino Digital Pin 2

LDR:

- One end connected to 5V

- The other end connected to Analog Pin A0 and a 10kΩ resistor to GND

LEDs:

- Blue LED: Connected to Digital Pin 22 .
- Yellow LED: Connected to Digital Pin 23 .

4. Algorithm

Initialization Phase:

1. Define Pins:
    - Assign pin numbers for the sensors and LEDs.
2. Setup Function:
    - Initialize serial communication for debugging and monitoring data.
    - Set sensor and LED pins as outputs/inputs as required.

Main Loop:

1. Read Sensor Data:
    - Read temperature, humidity from DHT11.
    - Read light intensity from LDR.
2. Display Data:
    - Print the sensor data to the Serial Monitor.
3. Decision-Making Based on Data:
    - Compare sensor readings to predefined thresholds.
    - Control LEDs based on sensor data.
4. Send Data to Server:
    - Send sensor data to the remote server for logging and analysis.
5. Delay:
    - Wait for a specified period before taking the next measurement to ensure smooth operation.

5. Code:

```
#include <WiFi.h>

#include <HTTPClient.h>

#include <ArduinoJson.h>

#include <Adafruit_Sensor.h>

#include <DHT.h>



#define DHTPIN 15

#define DHTTYPE DHT11
```

```
#define LDRPIN 34

#define LED_BLUE 22

#define LED_YELLOW 23

#define LED_INBUILT 2


const char* ssid = "Bravo";

const char* password = "bravo220";

const char* serverName = "http://192.168.1.100:8000/endpoint/";

const char* thresholdsUrl = "http://192.168.1.100:8000/api/get_thresholds/";


DHT dht(DHTPIN, DHTTYPE);


// Threshold values

float TEMP_THRESHOLD = 37.0;

int LDR_THRESHOLD = 850;


void setup() {

  Serial.begin(115200);

  dht.begin();

  pinMode(LED_BLUE, OUTPUT);

  pinMode(LED_YELLOW, OUTPUT);

  pinMode(LED_INBUILT, OUTPUT);


  WiFi.begin(ssid, password);

  while (WiFi.status() != WL_CONNECTED) {

    delay(1000);

    Serial.println("Connecting to WiFi...");
```

```
  }
  Serial.println("Connected to WiFi");

  fetchThresholds();
}

void loop() {
  static unsigned long lastCheckTime = 0;
  unsigned long currentTime = millis();

  // Check thresholds every 10 seconds
  if (currentTime - lastCheckTime >= 10000) {
    lastCheckTime = currentTime;
    fetchThresholds();
  }

  digitalWrite(LED_INBUILT, HIGH);

  float humidity = dht.readHumidity();
  float temperature = dht.readTemperature();
  int ldrValue = analogRead(LDRPIN);

  Serial.print("Temperature: ");
  Serial.print(temperature);
  Serial.print(" °C, Humidity: ");
  Serial.print(humidity);
  Serial.print(" %, LDR: ");
```

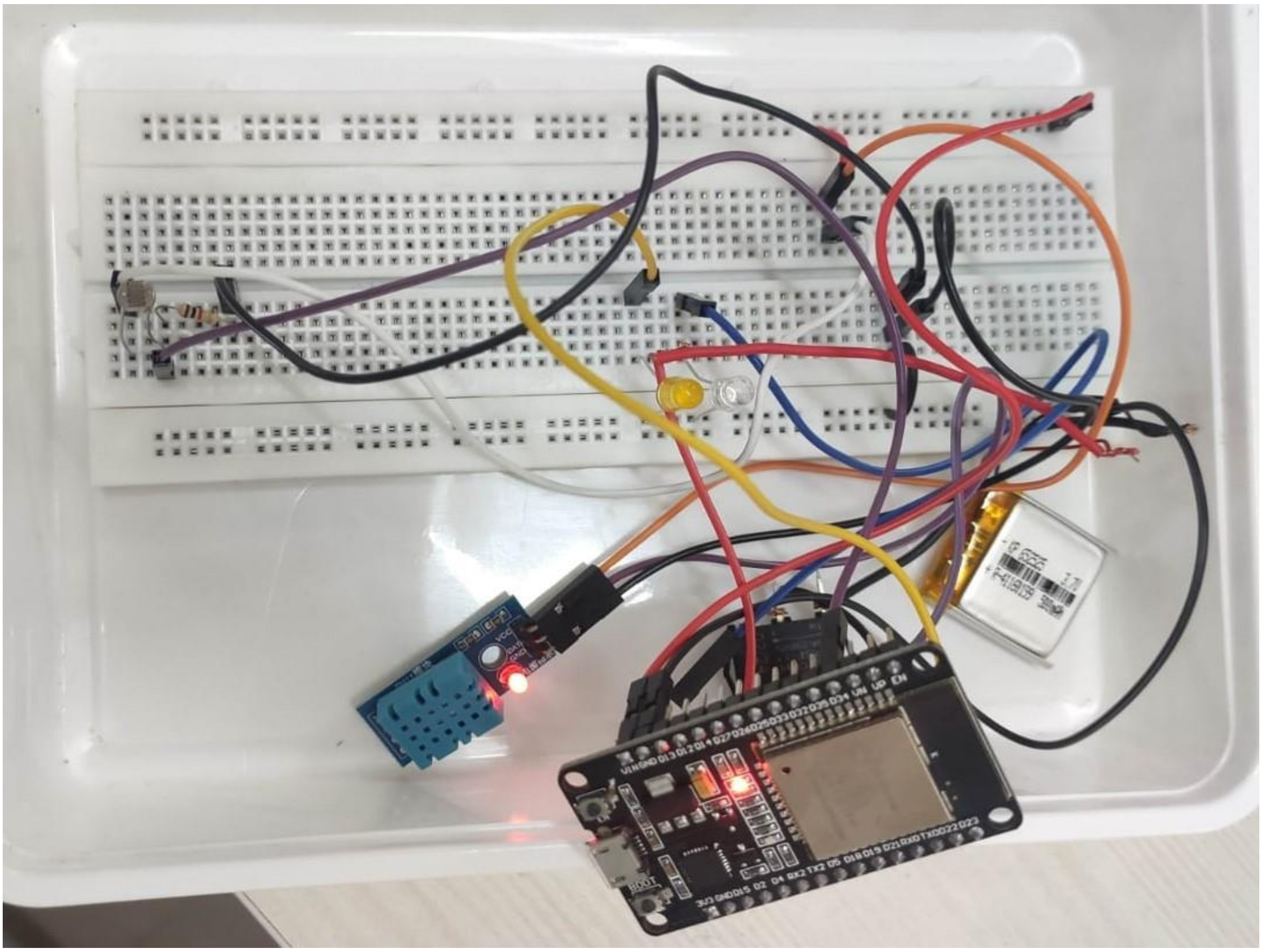
```
  Serial.println(ldrValue);


  if (temperature > TEMP_THRESHOLD) {
    digitalWrite(LED_BLUE, HIGH);
  } else {
    digitalWrite(LED_BLUE, LOW);
  }


  if (ldrValue > LDR_THRESHOLD) {
    digitalWrite(LED_YELLOW, HIGH);
  } else {
    digitalWrite(LED_YELLOW, LOW);
  }


  if (WiFi.status() == WL_CONNECTED) {
    HTTPClient http;
    http.begin(serverName);
    http.addHeader("Content-Type", "application/json");


    // Create JSON payload
    String jsonData = "{\"temperature\": " + String(temperature) + ", \"humidity\": " + String(humidity)
+ ", \"ldr\": " + String(ldrValue) + "}";


    int httpResponseCode = http.POST(jsonData);
    if (httpResponseCode > 0) {
      String response = http.getString();
      Serial.println(httpResponseCode);
      Serial.println(response);
```

```
    } else {
      Serial.print("Error on sending POST: ");
      Serial.println(httpResponseCode);
    }

    http.end();
  }

  delay(1000);
}

void fetchThresholds() {
  if (WiFi.status() == WL_CONNECTED) {
    HTTPClient http;
    http.begin(thresholdsUrl);
    int httpResponseCode = http.GET();
    if (httpResponseCode > 0) {
      String response = http.getString();
      Serial.println("Thresholds response: " + response);

      // Use ArduinoJson to parse the response
      DynamicJsonDocument doc(1024);
      deserializeJson(doc, response);
      float newTempThreshold = doc["temp_threshold"];
      int newLdrThreshold = doc["ldr_threshold"];

      // Update threshold values
```

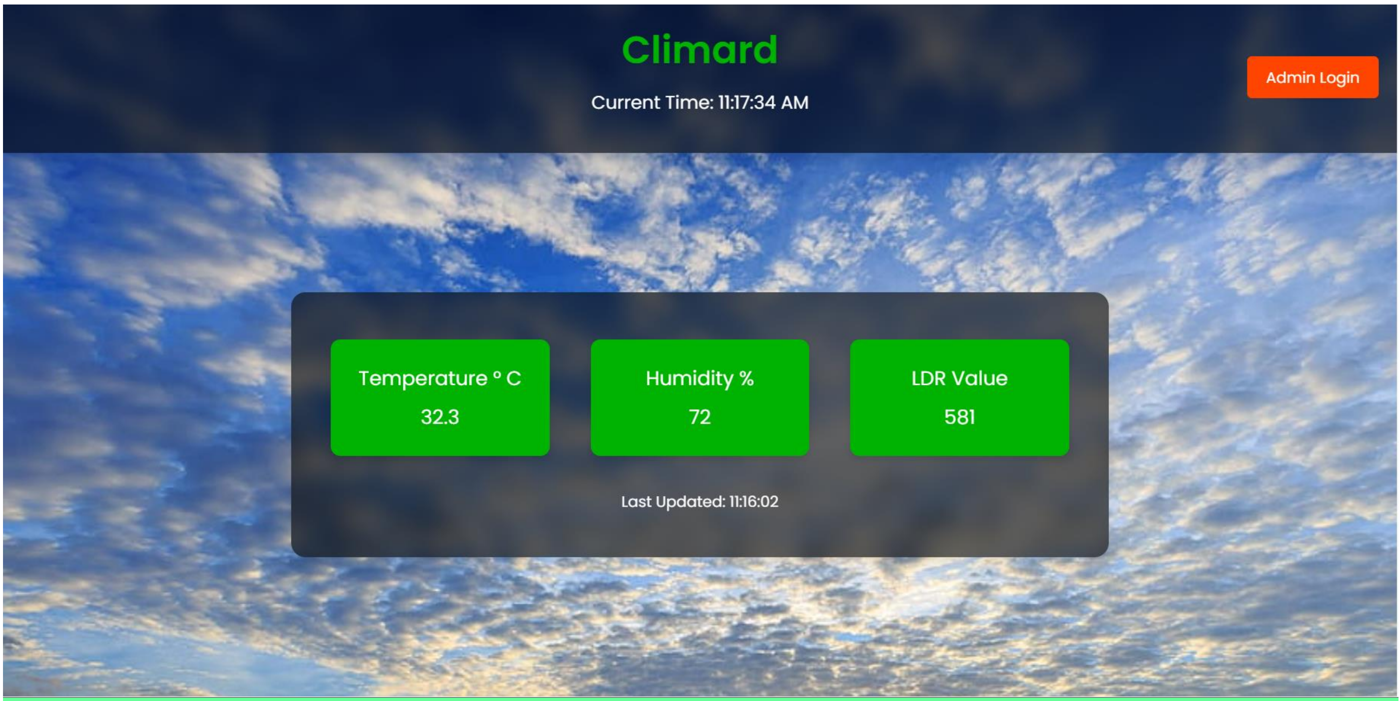
```
      TEMP_THRESHOLD = newTempThreshold;
      LDR_THRESHOLD = newLdrThreshold;

      // Control LEDs based on new thresholds
      updateLEDs();
    } else {
      Serial.print("Error on getting thresholds: ");
      Serial.println(httpResponseCode);
    }
    http.end();
  }
}

void updateLEDs() {
  float temperature = dht.readTemperature();
  int ldrValue = analogRead(LDRPIN);

  if (temperature > TEMP_THRESHOLD) {
    digitalWrite(LED_BLUE, HIGH);
  } else {
    digitalWrite(LED_BLUE, LOW);
  }

  if (ldrValue > LDR_THRESHOLD) {
    digitalWrite(LED_YELLOW, HIGH);
  } else {
    digitalWrite(LED_YELLOW, LOW);
```

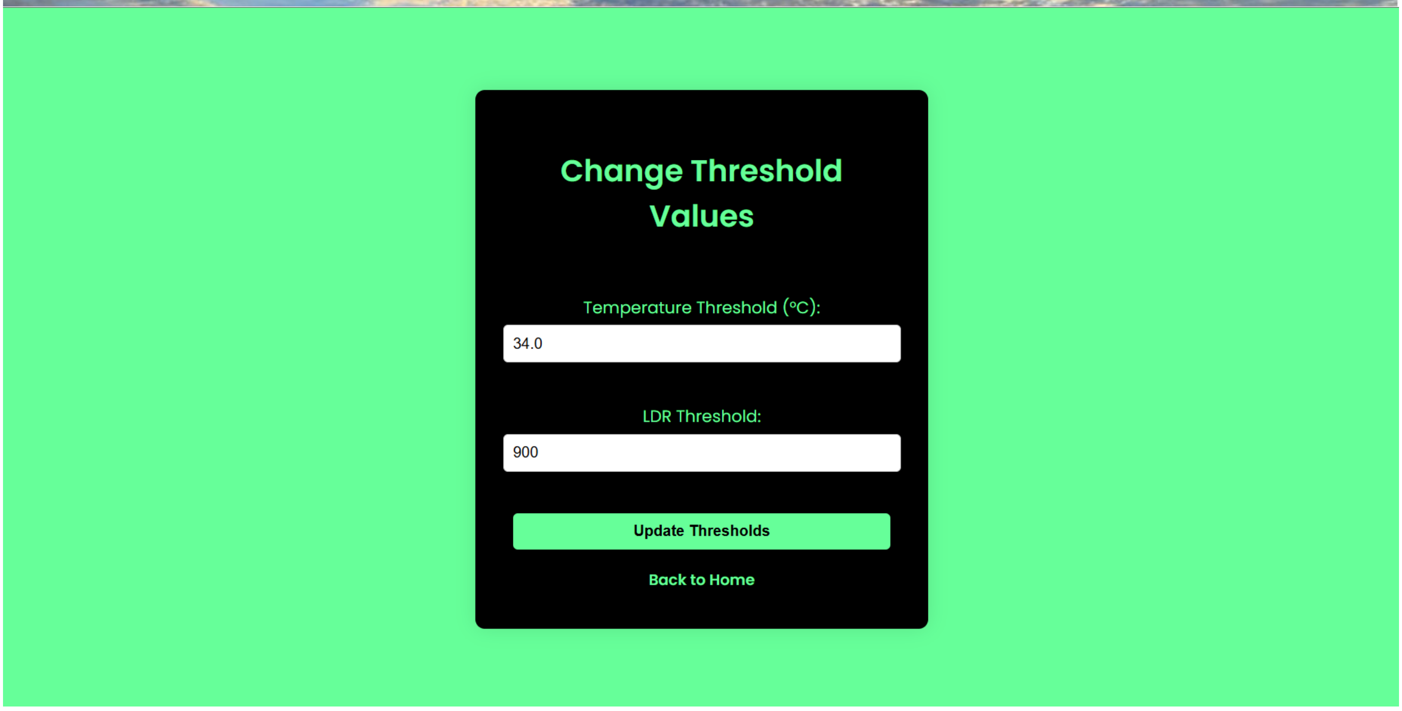
```
  }
}
```

6. Snapshot of project:



7. HTML Pages:





8. Future scope of the project:

- Integration with Additional Sensors: Adding more sensors such as CO2, PM2.5 for comprehensive environmental monitoring.
- Wireless Data Transmission: Using Bluetooth or GSM modules for wireless data transmission and remote monitoring.
- IoT Integration: Connecting to IoT platforms for real-time data logging, analysis, and control.
- Solar Power: Implementing solar power solutions to make the device self-sufficient.
- Mobile App Development: Creating a mobile application for real-time monitoring and control of the device.

9. GitHub repository :

Singaram-117/Climard: Climard is an Arduino IOT project to predict the weather in a room using ESP32 chip. (github.com)

10. References:

1. https://projecthub.arduino.cc/rajeshjiet/iot-based-weather-monitoring-system-using-arduino-a3334a
2. https://projectsfactory.in/product/iot-weather-monitoring-system-using-arduino/