DESIGN AND IMPLEMENTATION OF A WEATHER MEASUREMENT SYSTEM

Ayomide Ajayi

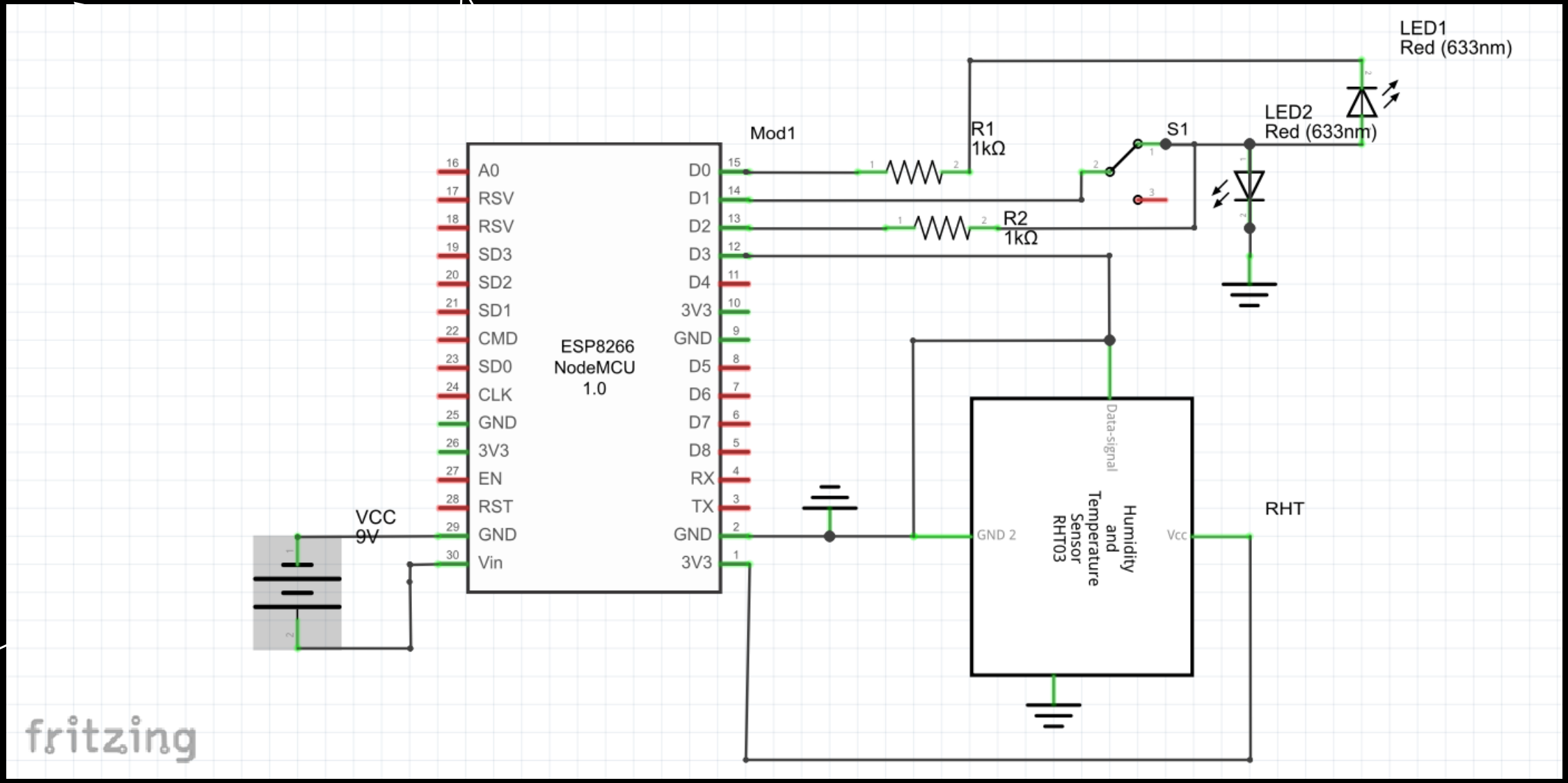
Embedded Systems Lecturer: Prof. Lajos Harasztosi
26/11/2023

OVERVIEW

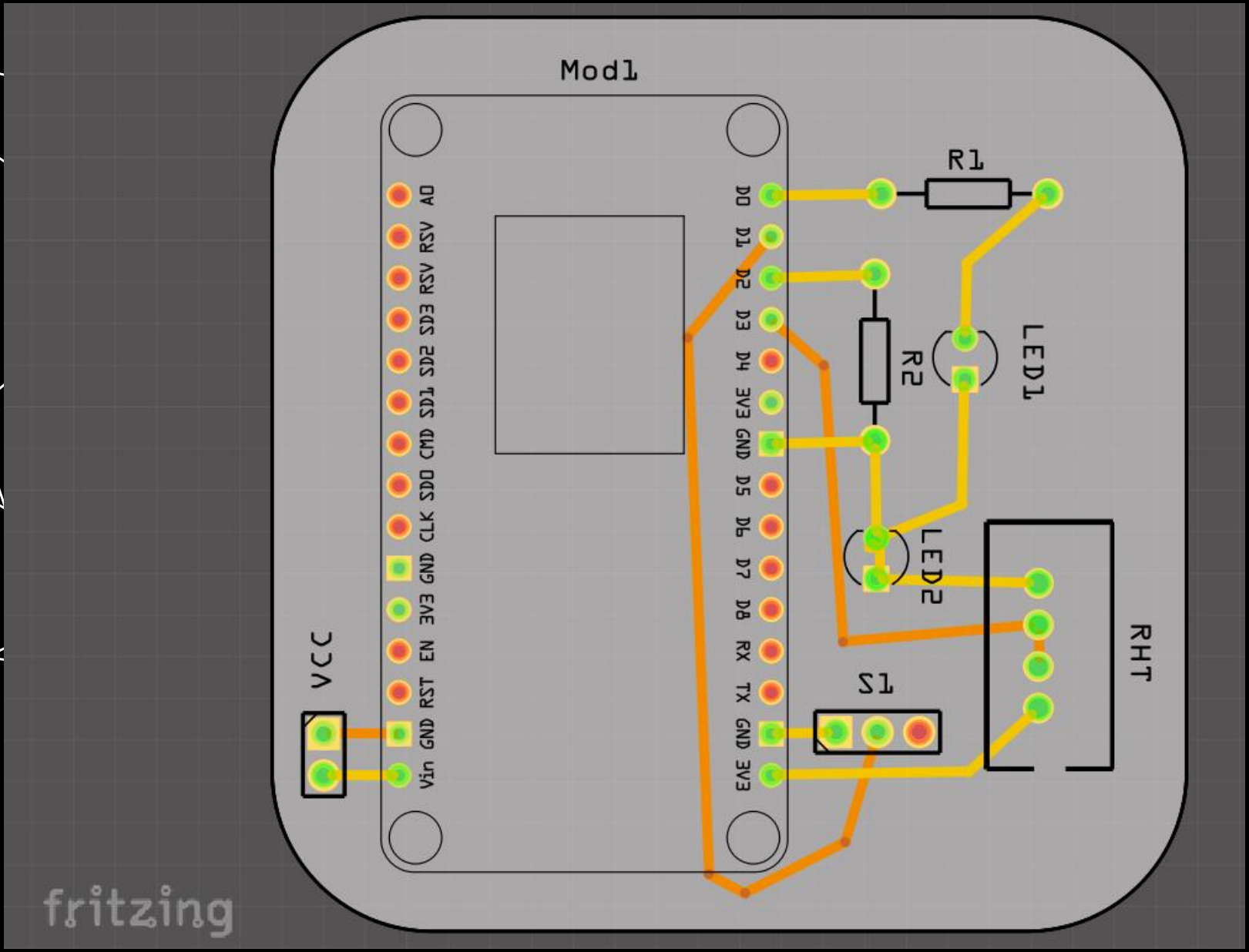
Design an autonomous weather monitoring system that sends temperature and humidity data to the cloud to be accessible from anywhere in the world.

Components used:
DHT22 - Temperature and Humidity sensor
Battery – Power supply
Resistors – two 1k Ohms resistors
Tact switch
2 LEDs – WiFi indicator and Power indicator
Node MCU – ESP8266 microcontroller

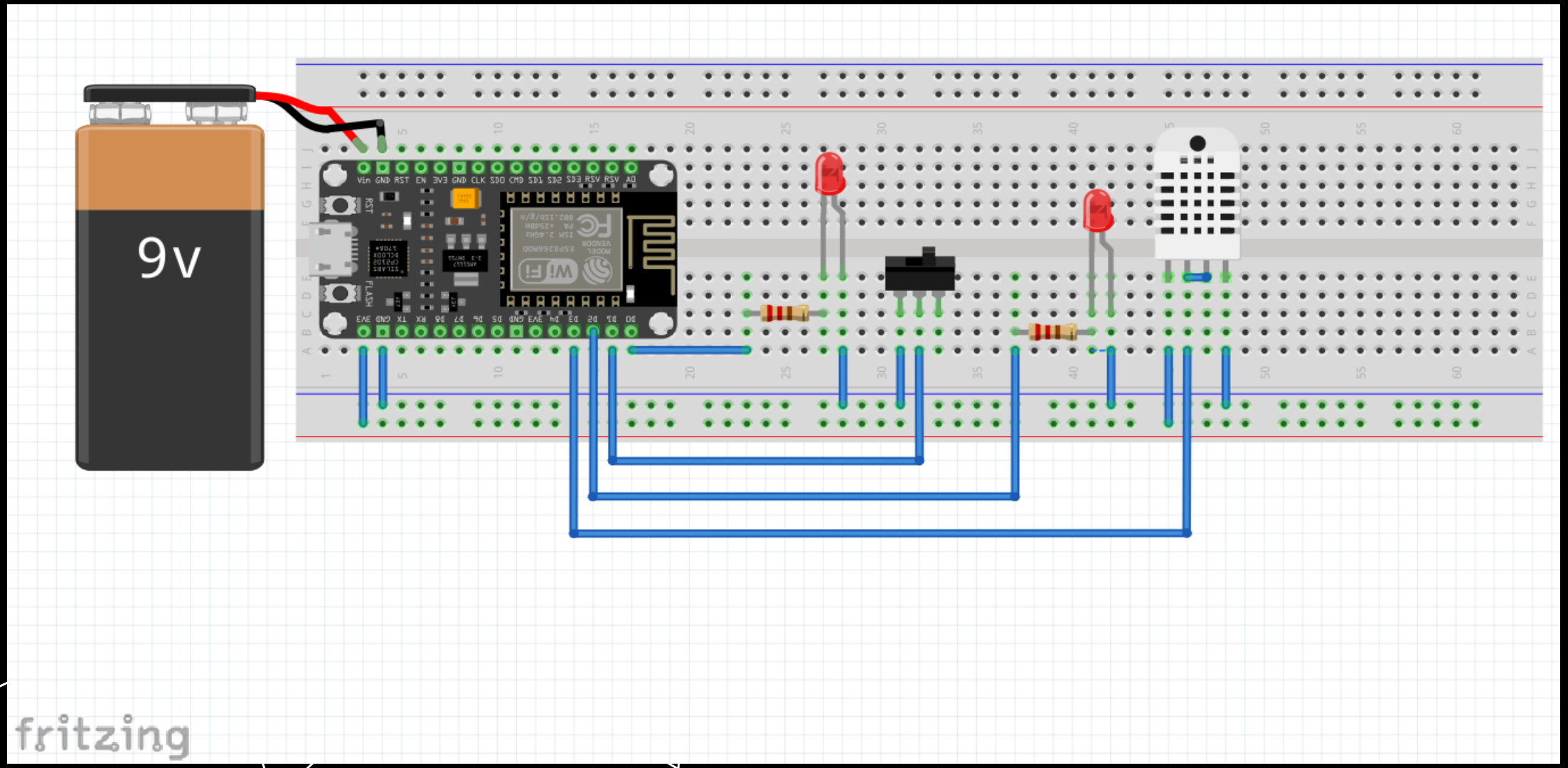
SCHEMATIC DIAGRAM



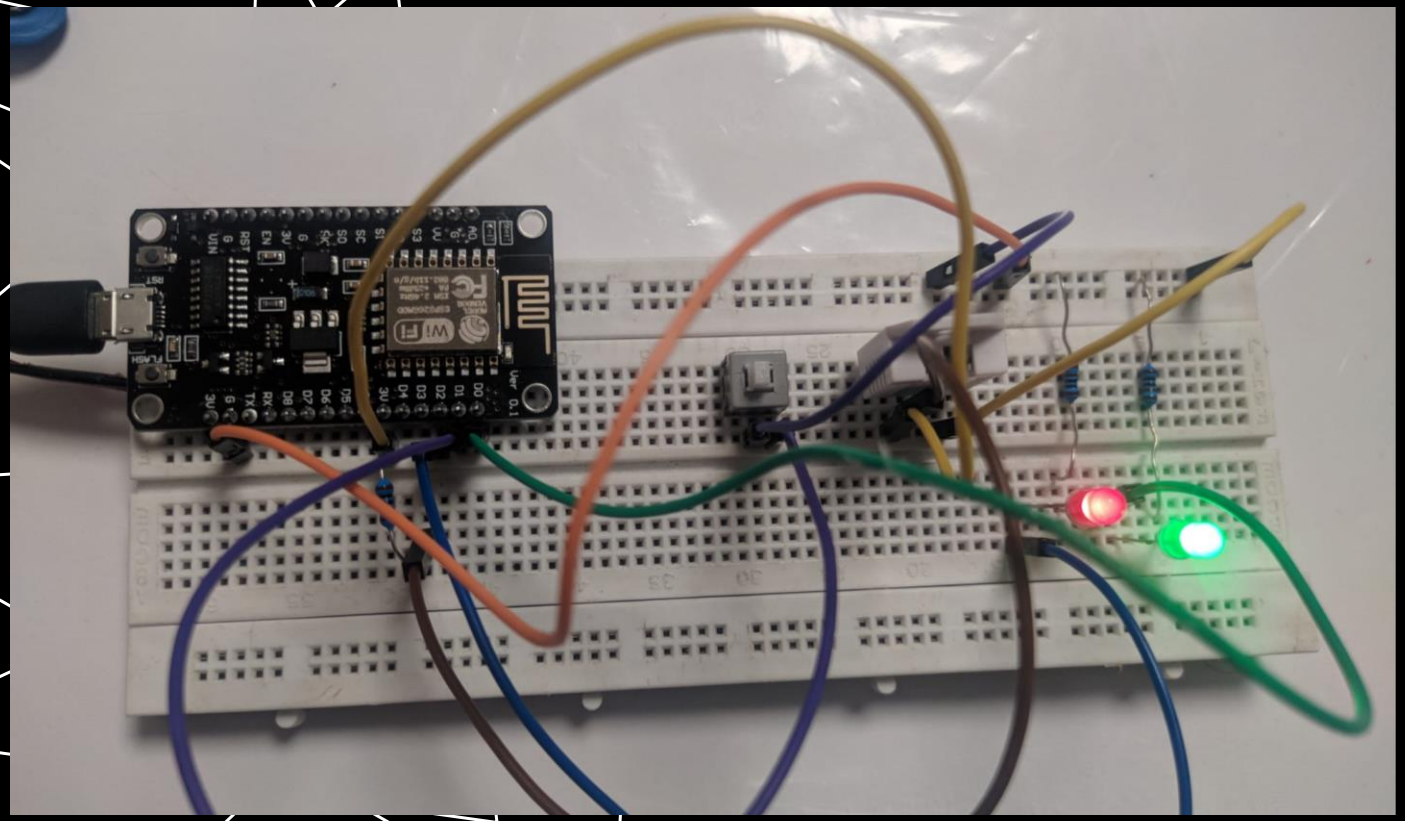
PRINTED CIRCUIT BOARD DESIGN



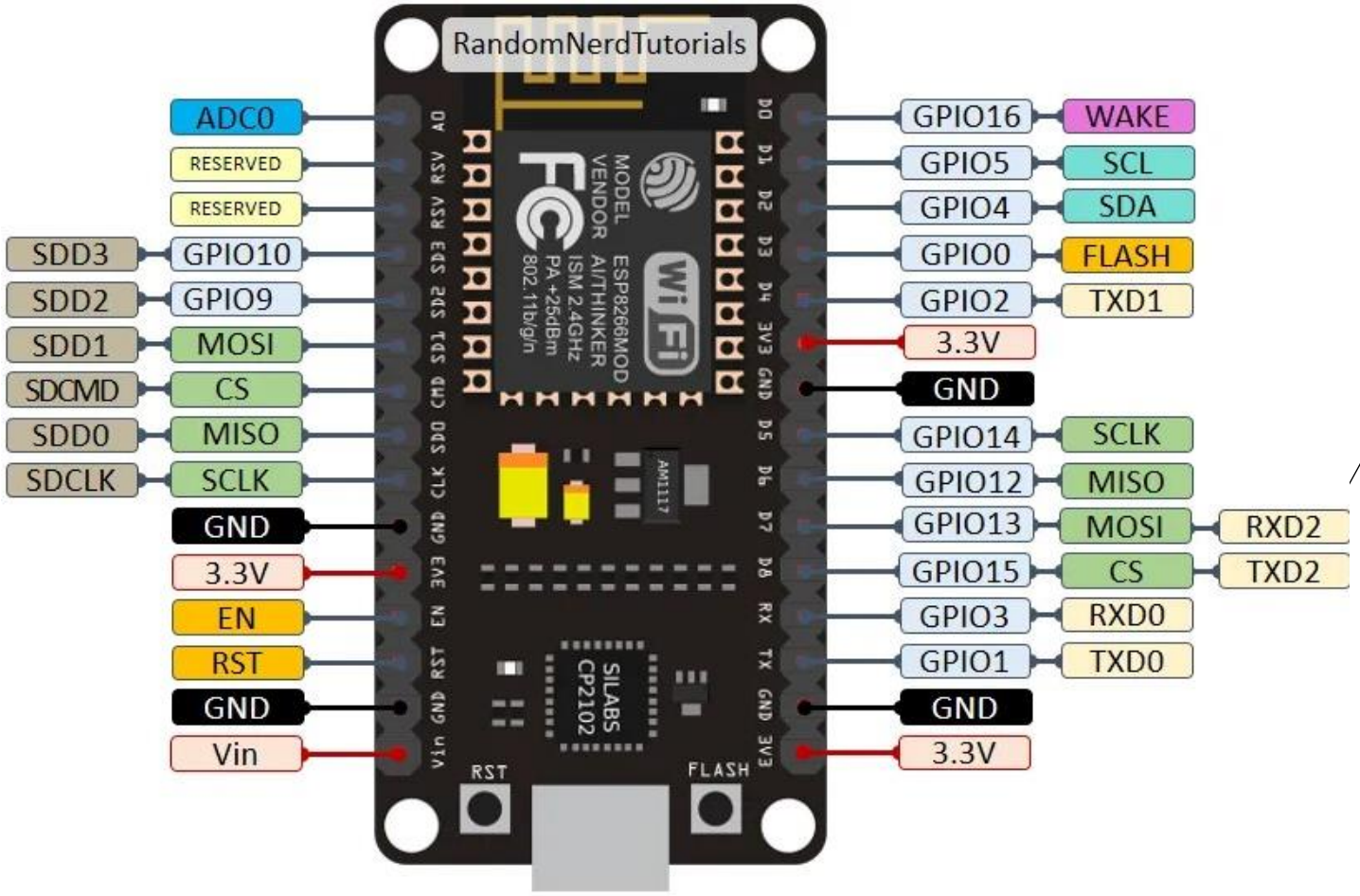
BREADBOARD CIRCUIT LAYOUT



CIRCUIT AT WORK



PINOUT DIAGRAM OF ESP8266 NODEMCU V2



OTHER COMPONENTS USED IN THE PROJECT

DHT22 Temperature and Humidity Sensor

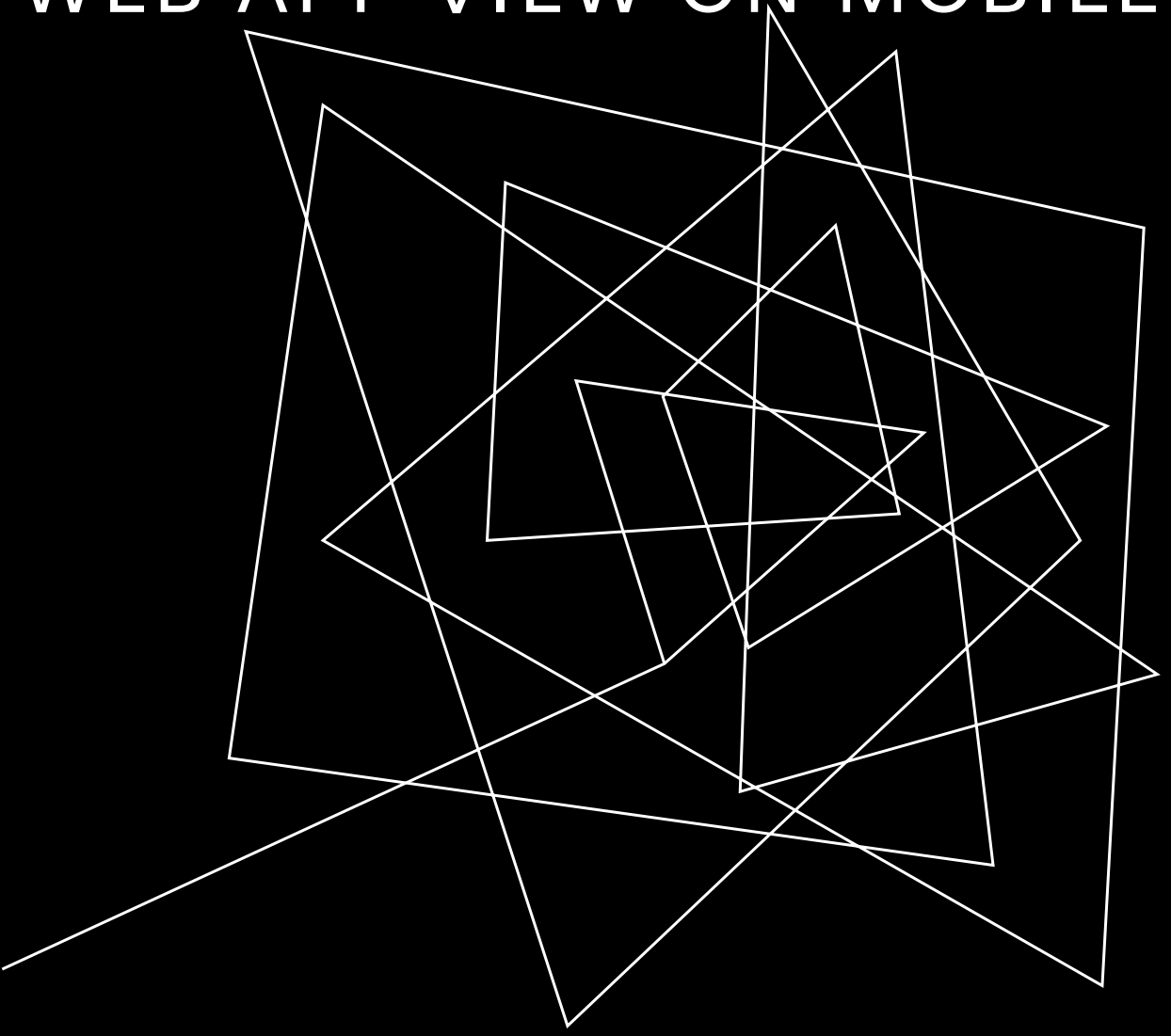
Resistors

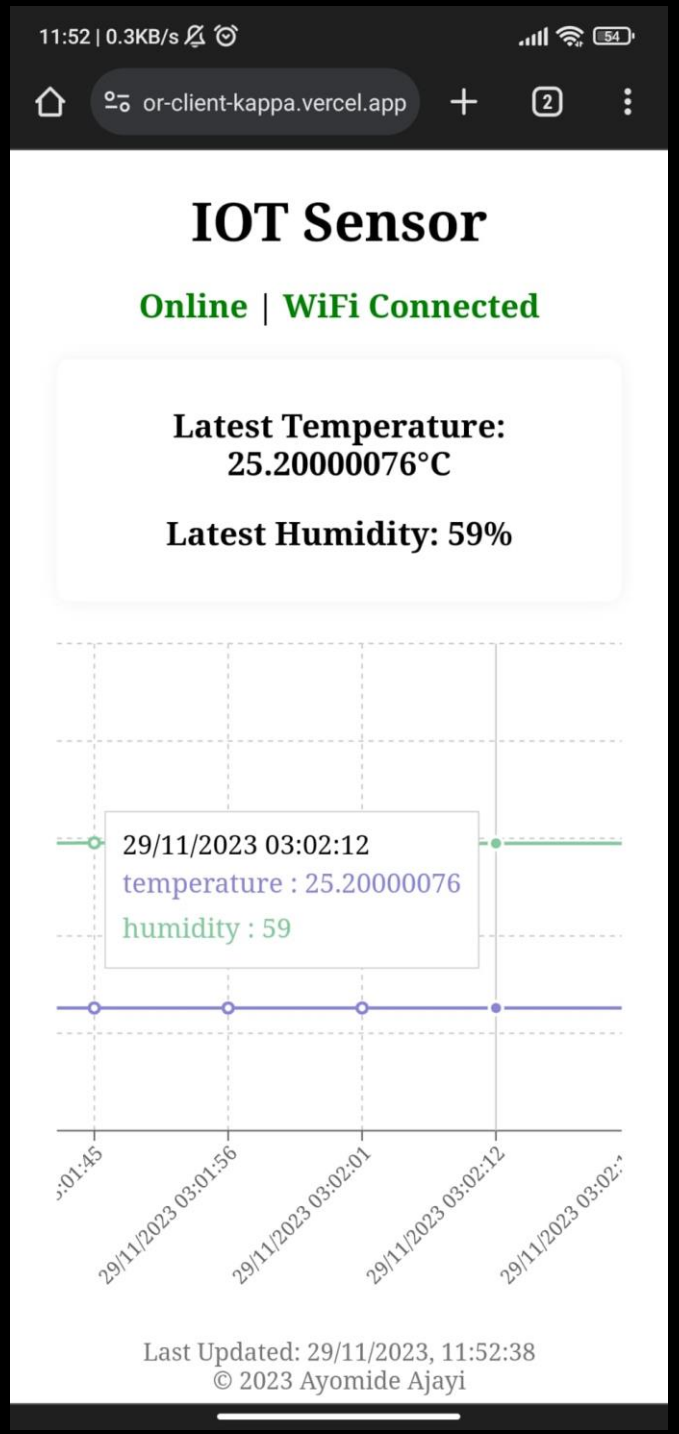
LEDs

Battery

Switch

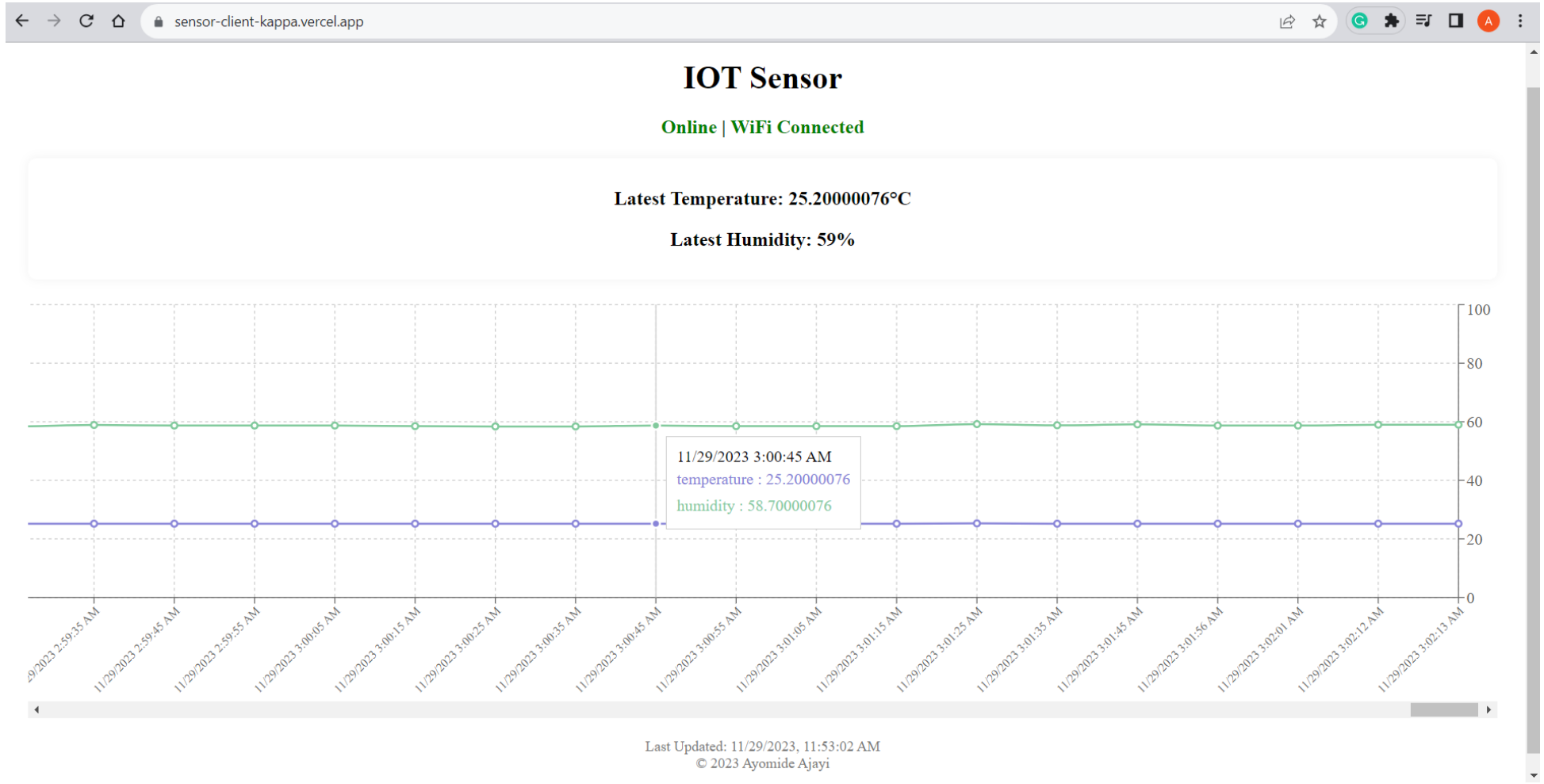
WEB APP VIEW ON MOBILE PHONE





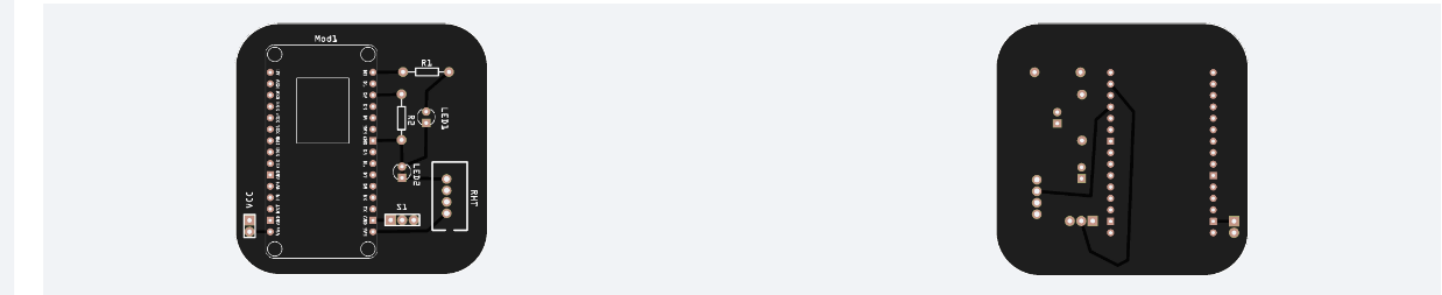
https://sensor-client-kappa.vercel.app/

WEB APP ON PC WITH AN EXTENDED CHART VIEW



https://sensor-client-kappa.vercel.app/

JLCPCB PCB ORDERING PROCESS USING GERBER FILE



← Back to Upload File

Detected 2 layer board of 55.96x55.93mm(2.2x2.2 inches) .

Gerber Viewer

Base Material	FR-4 / Flex / Aluminum / Copper Core / Rogers / PTFE Teflon
Layers	1 / 2 / 4 / High Precision PCB / 6 / 8 / 10 / 12 / 14 / 16 / 18 / 20
Dimensions	55.93 * 55.96 mm
PCB Qty	5
Product Type	Industrial/Consumer electronics / Aerospace / Medical

PCB Specifications

Different Design	1 / 2 / 3 / 4
Delivery Format	Single PCB / Panel by Customer / Panel by JLCPCB

Charge Details

Special Offer	$2.00
Via Covering	$0.00
Surface Finish	$0.00

Build Time

PCB: 3-4 days	$0.00

Calculated Price ~~$4.00~~ $2.00

Additional charges may apply for special cases

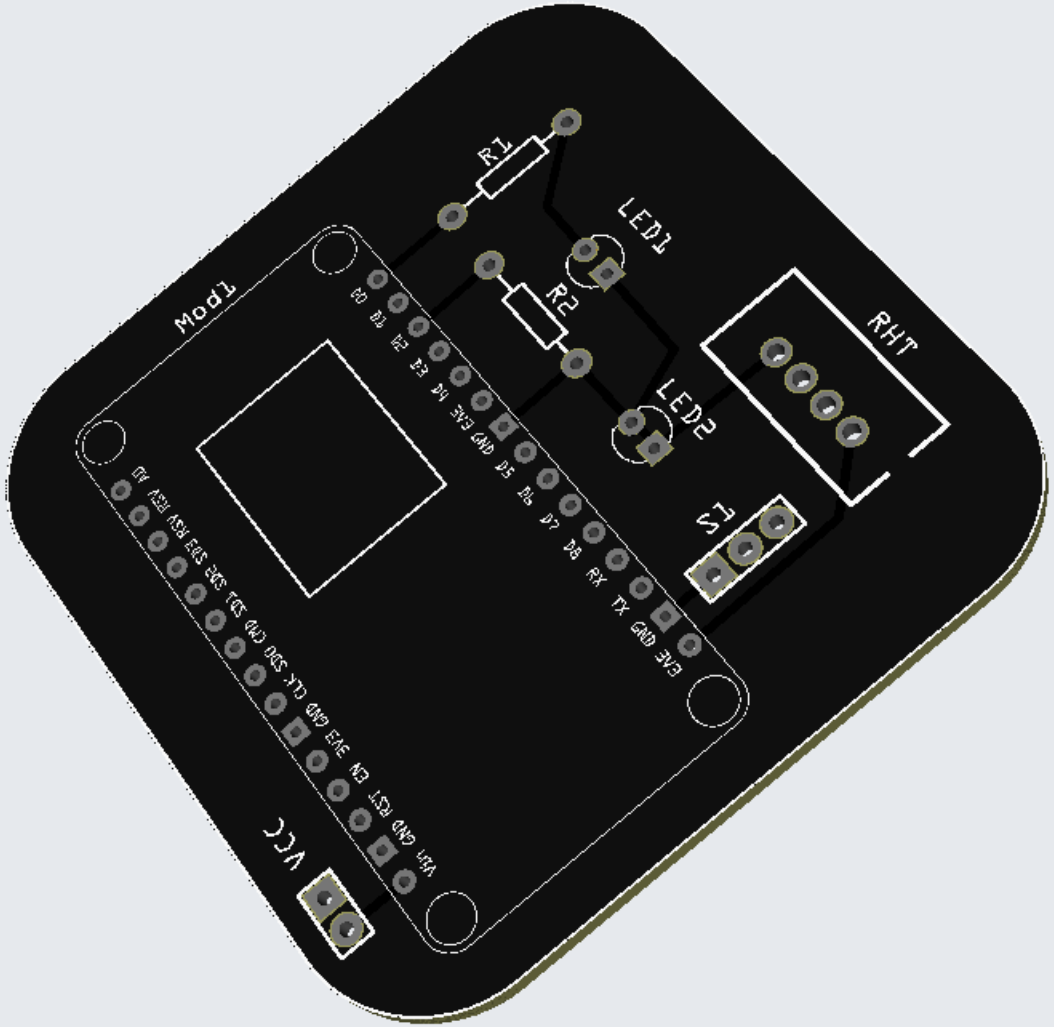
SAVE TO CART

Shipping Estimate

Charge: Choose destination country first

Weight 0.17kg

3D VIEW OF THE 2-LAYER PCB ON JLCPCB



https://github.com/ayo-ajayi/iot-sensor

FIRMWARE SCRIPT

main — iot-sensor / firmware / src / main.cpp

Code | Blame | 211 lines (180 loc) · 4.58 KB

```
void loop()
{
  unsigned long currentMillis = millis();

  if (digitalRead(SWITCH_PIN) == HIGH)
  {
    device.isOn = true;
    digitalWrite(LED_PIN, HIGH);

    if (!device.wifiConnected)
    {
      connectToWifi();
    }

    if (currentMillis - lastSensorReadTime >= sensorReadInterval)
    {
      lastSensorReadTime = currentMillis;
      readSensorData();
    }

    if (currentMillis - lastSendTime >= sendDataInterval)
    {
      lastSendTime = currentMillis;
      sendData();
    }
    return;
  }
  device.isOn = false;
  if (device.wifiConnected)
  {
    device.wifiConnected = false;
    sendData();
    delay(200);
    WiFi.disconnect();
    digitalWrite(WIFI_LED_PIN, LOW);
    Serial.println("WiFi is not connected");
  }
  digitalWrite(LED_PIN, LOW);
}
```

https://github.com/ayo-ajayi/iot-sensor

SERVER SCRIPT

main iot-sensor / server / main.go

Code Blame 180 lines (156 loc) · 4.9 KB Raw

```go
func main() {

    r.POST("/sensor-data", func(c *gin.Context) {
        sensorDataJson := struct {
            Temperature float64 `json:"temperature" binding:"required"`
            Humidity    float64 `json:"humidity" binding:"required"`
        }{}
        if err := c.ShouldBindJSON(&sensorDataJson); err != nil {
            c.JSON(400, gin.H{"error": err.Error()})
            return
        }

        sd := &SensorData{
            Humidity:    sensorDataJson.Humidity,
            Temperature: sensorDataJson.Temperature,
            UpdatedAt:   time.Now(),
            Id:          primitive.NewObjectID(),
        }

        if _, err := sensorDataCollection.InsertOne(context.Background(), sd); err != nil {
            c.JSON(500, gin.H{"error": err.Error()})
            return
        }
        c.JSON(200, gin.H{"message": "sensor data saved", "data": sd})
    })

    r.POST("/device-status", func(c *gin.Context) {
        deviceStatusJson := struct {
            IsOn          bool `json:"is_on"`
            WifiConnected bool `json:"wifi_connected"`
        }{}

        if err := c.ShouldBindJSON(&deviceStatusJson); err != nil {
            c.JSON(400, gin.H{"error": err.Error()})
            return
        }

        ds := &DeviceStatus{
            IsOn:          deviceStatusJson.IsOn,
            WifiConnected: deviceStatusJson.WifiConnected,
```

https://github.com/ayo-ajayi/iot-sensor

WEB APP SCRIPT

main iot-sensor / client / components / Dashboard.jsx

Code Blame 176 lines (163 loc) · 4.44 KB

```
const fetchData = async (endpoint) => {
  const response = await fetch(
    `${BaseUrl}${endpoint}?_ts=${new Date().getTime()}`
  );
  if (!response.ok) {
    throw new Error(`HTTP error! status: ${response.status}`);
  }
  return await response.json();
};

const Dashboard = () => {
  const [deviceStatus, setDeviceStatus] = useState({
    is_on: false,
    wifi_connected: false,
  });
  const [sensorData, setSensorData] = useState([]);
  const [lastUpdate, setLastUpdate] = useState("");
  const chartContainerRef = useRef(null);

  useEffect(() => {
    const getDeviceStatus = async () => {
      const statusData = await fetchData("device-status");
      setDeviceStatus(statusData);
    };

    const getSensorData = async () => {
      const data = await fetchData("sensor-data");
      const formattedData = data.map((sd) => ({
        ...sd,
        dateTime: `${new Date(sd.updated_at).toLocaleDateString()} ${new Date(
          sd.updated_at
        ).toLocaleTimeString()}`,
      }));
      setSensorData(formattedData);
      setLastUpdate(new Date().toLocaleString());
    };

    getDeviceStatus();
    getSensorData();
    const statusInterval = setInterval(getDeviceStatus, 10000);
    const sensorDataInterval = setInterval(getSensorData, 120000);
```

SOFTWARE USED FOR PROJECT:

PlatformIO [C++] – Microcontroller programming

Golang – Server Development

ReactJS – Client/Web Interface

MongoDB – Database for storing data

Fritzing – PCB Design

THANK YOU

ajayiayomide247@gmail.com