

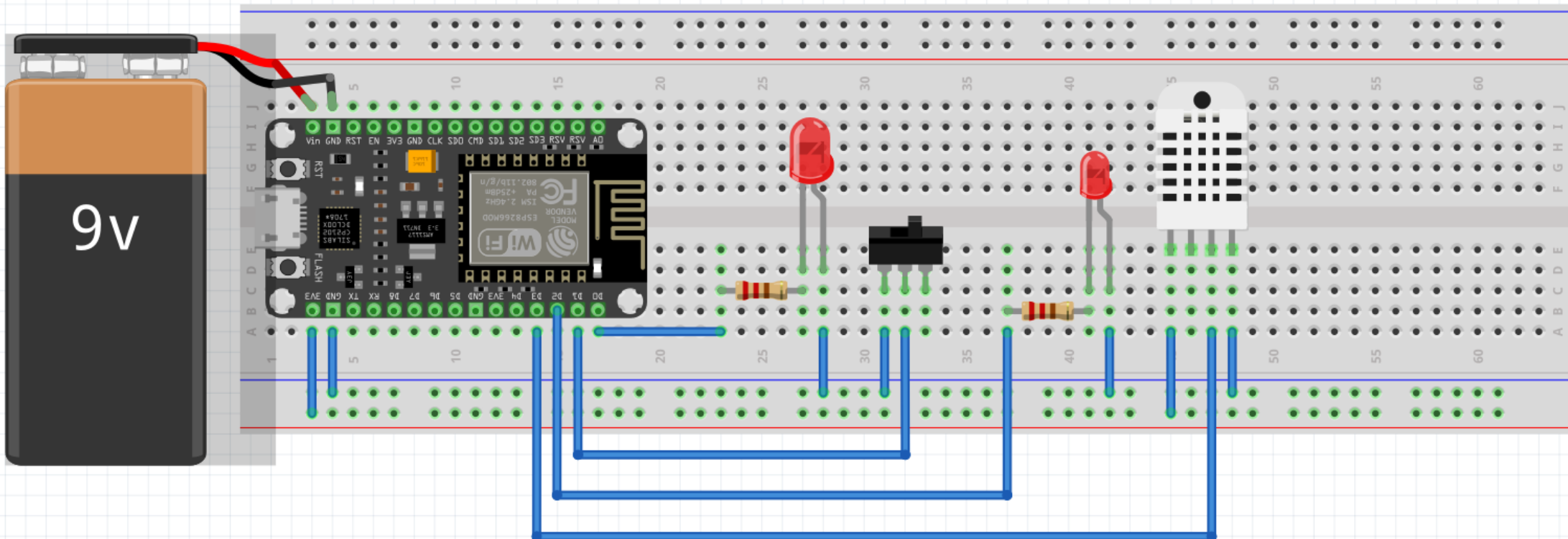
# DESIGN AND IMPLEMENTATION OF A WEATHER MEASUREMENT SYSTEM

Ayomide Ajayi

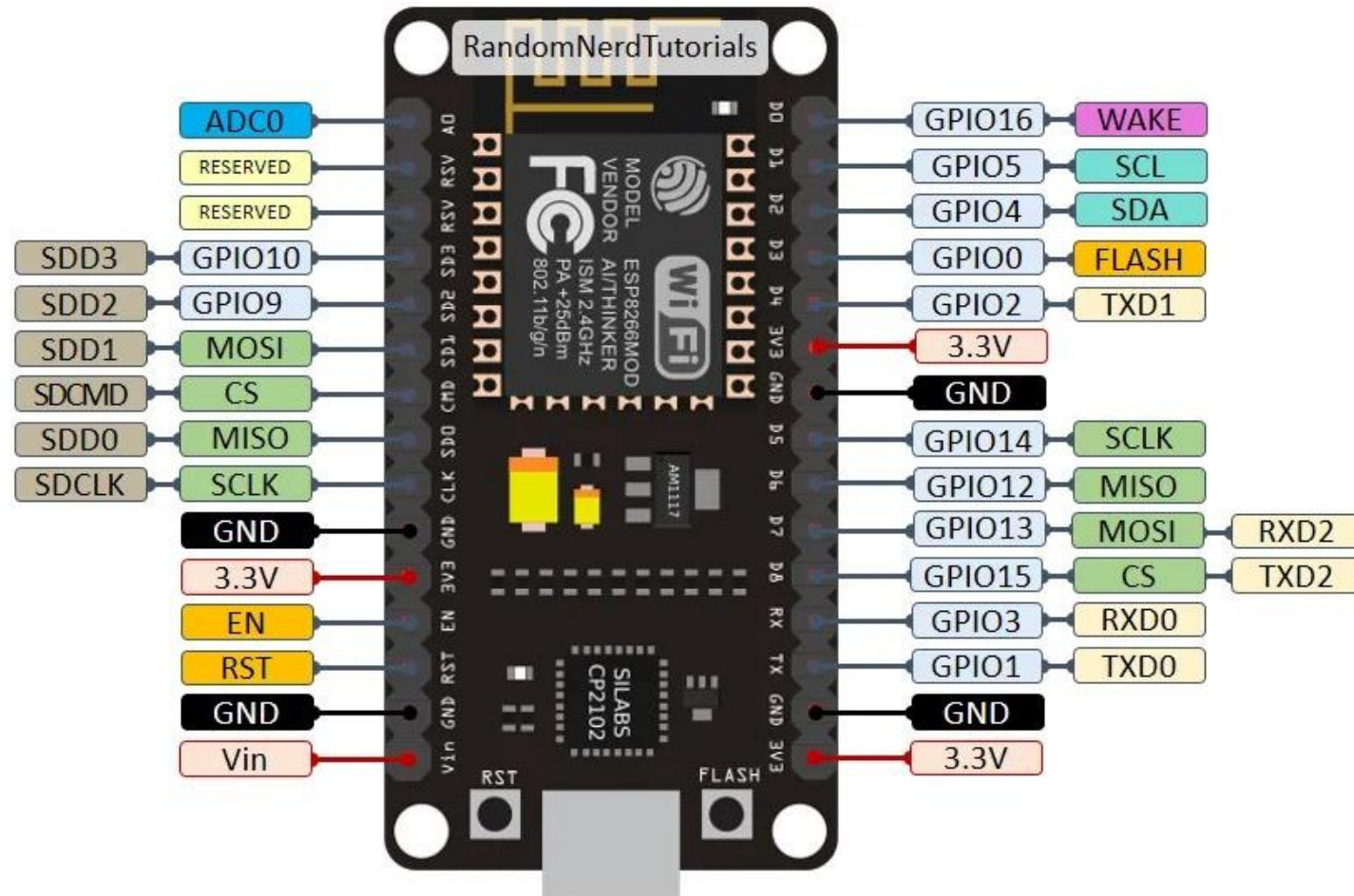
Embedded Systems Lecturer: Prof. Lajos Harasztosi

Electrical Engineering, UniDeb

26/11/2023

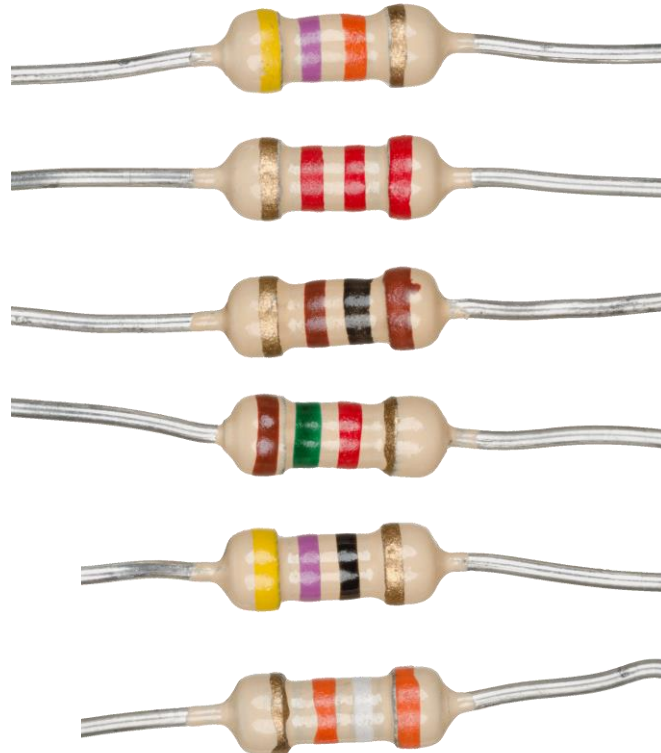


# PINOUT DIAGRAM OF ESP8266 NODEMCU V2



# OTHER COMPONENTS USED IN THE PROJECT

Resistors



LEDs



DHT22 Temperature and Humidity Sensor

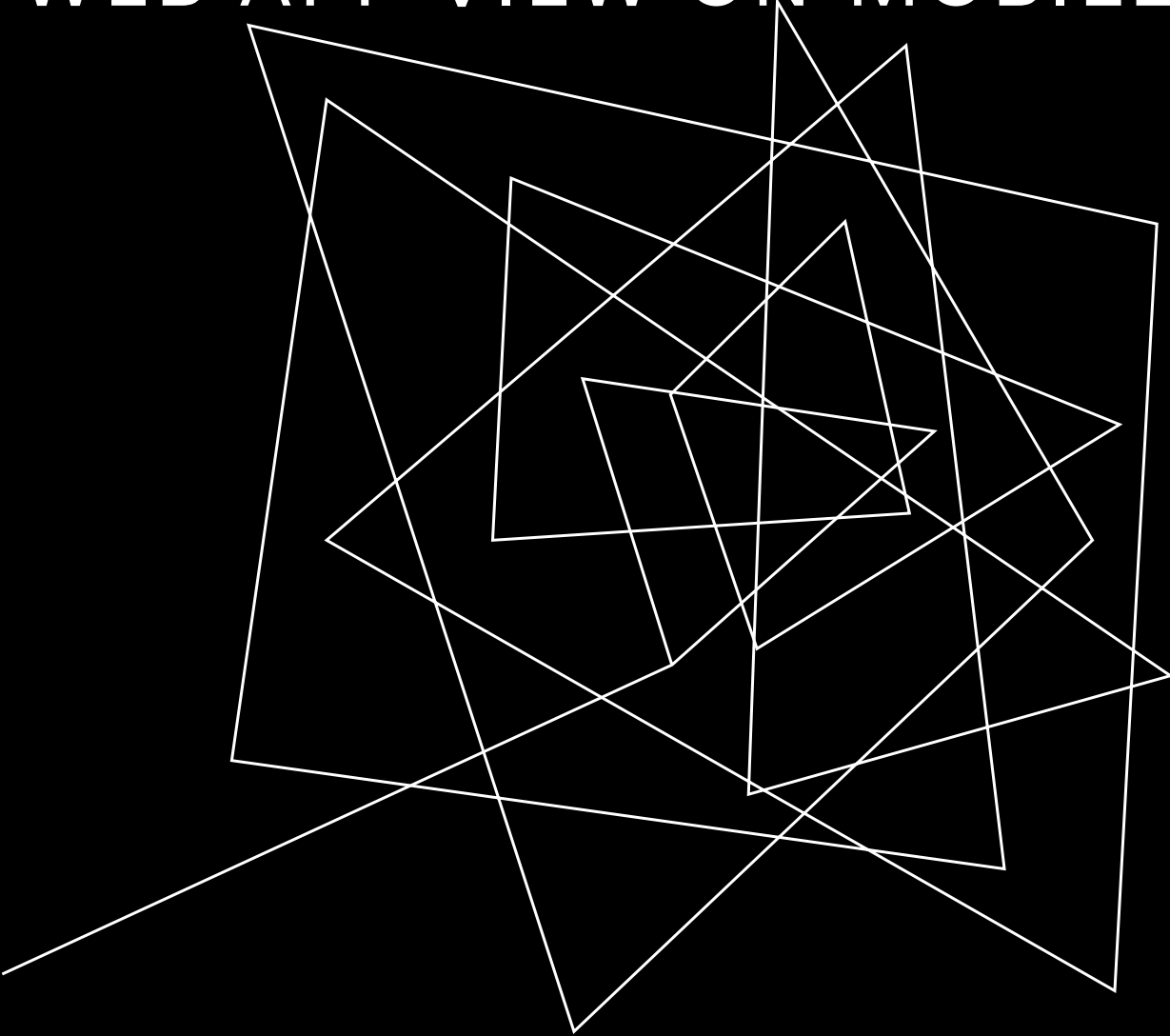


Battery

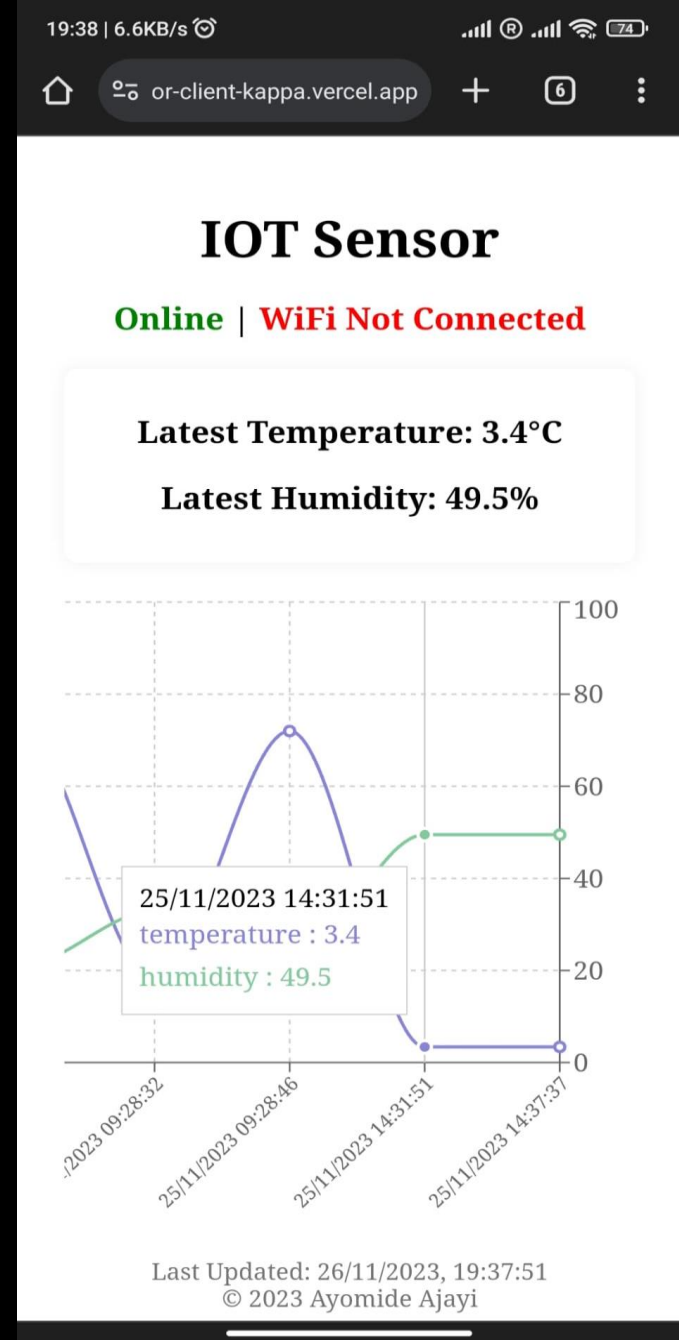


Switch

# WEB APP VIEW ON MOBILE PHONE



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



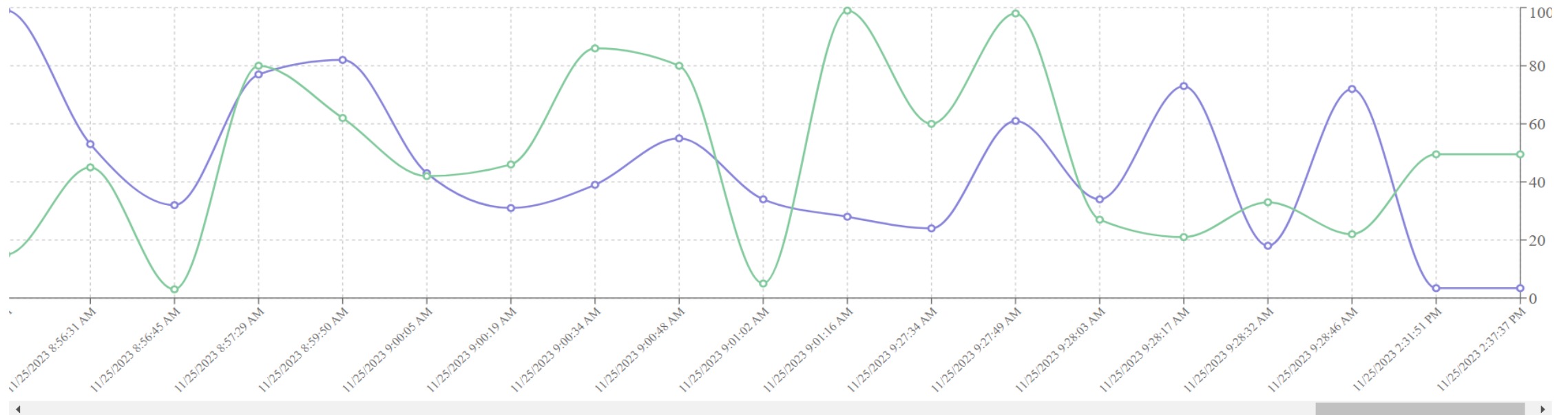
# WEB APP ON PC WITH AN EXTENDED CHART VIEW

## IOT Sensor

Online | WiFi Not Connected


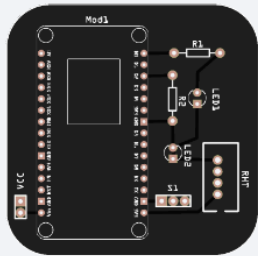
Latest Temperature: 3.4°C

Latest Humidity: 49.5%



<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

Via Covering

Surface Finish

\$2.00

\$0.00

\$0.00

Build Time

PCB: 3-4 days

\$0.00

Calculated Price

~~\$4.00~~

\$2.00

Additional charges may apply for [special cases](#)

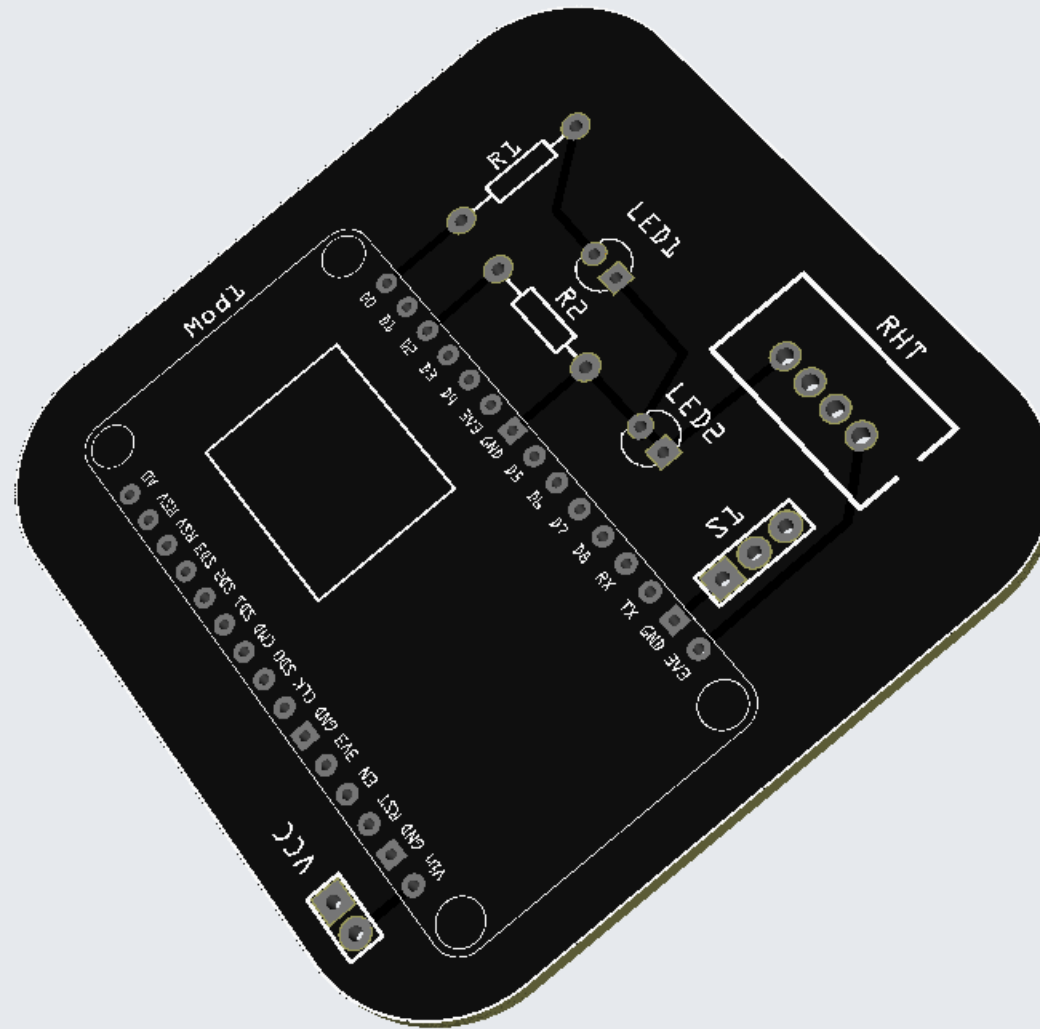
SAVE TO CART

Shipping Estimate

Charge: [Choose destination country first](#)

Weight 0.17kg

## 3D VIEW OF THE 2-LAYER PCB ON JLCPCB





<https://github.com/ayojay/iot-sensor>

## FIRMWARE SCRIPT

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

Code Blame 211 lines (180 loc) · 4.58 KB

70
71 void loop()
72 {
73     unsigned long currentMillis = millis();
74
75     if (digitalRead(SWITCH_PIN) == HIGH)
76     {
77         device.isOn = true;
78         digitalWrite(LED_PIN, HIGH);
79
80         if (!device.wifiConnected)
81         {
82             connectToWifi();
83         }
84
85         if (currentMillis - lastSensorReadTime >= sensorReadInterval)
86         {
87             lastSensorReadTime = currentMillis;
88             readSensorData();
89         }
90
91         if (currentMillis - lastSendTime >= sendDataInterval)
92         {
93             lastSendTime = currentMillis;
94             sendData();
95         }
96         return;
97     }
98     device.isOn = false;
99     if (device.wifiConnected)
100     {
101         device.wifiConnected = false;
102         sendData();
103         delay(200);
104         WiFi.disconnect();
105         digitalWrite(WIFI_LED_PIN, LOW);
106         Serial.println("WiFi is not connected");
107     }
108     digitalWrite(LED_PIN, LOW);
109 }
110
```

<https://github.com/ayojayi/iot-sensor>

## SERVER SCRIPT

```
main | iot-sensor / server / main.go
Code Blame 180 lines (156 loc) · 4.9 KB Raw Copy Download
16 func main() {
95
96     r.POST("/sensor-data", func(c *gin.Context) {
97         sensorDataJson := struct {
98             Temperature float64 `json:"temperature" binding:"required"`
99             Humidity      float64 `json:"humidity" binding:"required"`
100         }{}
101         if err := c.ShouldBindJSON(&sensorDataJson); err != nil {
102             c.JSON(400, gin.H{"error": err.Error()})
103             return
104         }
105
106         sd := &SensorData{
107             Humidity:      sensorDataJson.Humidity,
108             Temperature:   sensorDataJson.Temperature,
109             UpdatedAt:     time.Now(),
110             Id:             primitive.NewObjectID(),
111         }
112
113         if _, err := sensorDataCollection.InsertOne(context.Background(), sd); err != nil {
114             c.JSON(500, gin.H{"error": err.Error()})
115             return
116         }
117         c.JSON(200, gin.H{"message": "sensor data saved", "data": sd})
118     })
119
120     r.POST("/device-status", func(c *gin.Context) {
121         deviceStatusJson := struct {
122             IsOn          bool `json:"is_on"`
123             WifiConnected bool `json:"wifi_connected"`
124         }{}
125
126         if err := c.ShouldBindJSON(&deviceStatusJson); err != nil {
127             c.JSON(400, gin.H{"error": err.Error()})
128             return
129         }
130
131         ds := &DeviceStatus{
132             IsOn:          deviceStatusJson.IsOn,
133             WifiConnected: deviceStatusJson.WifiConnected,
```

<https://github.com/ayojay/iot-sensor>

## WEB APP SCRIPT

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

Code Blame 176 lines (163 loc) · 4.44 KB

14  const fetchData = async (endpoint) => {
15      const response = await fetch(
16          `${BaseUrl}${endpoint}?_ts=${new Date().getTime()}`
17      );
18      if (!response.ok) {
19          throw new Error(`HTTP error! status: ${response.status}`);
20      }
21      return await response.json();
22  };
23
24  const Dashboard = () => {
25      const [deviceStatus, setDeviceStatus] = useState({
26          is_on: false,
27          wifi_connected: false,
28      });
29      const [sensorData, setSensorData] = useState([]);
30      const [lastUpdate, setLastUpdate] = useState("");
31      const chartContainerRef = useRef(null);
32
33      useEffect(() => {
34          const getDeviceStatus = async () => {
35              const statusData = await fetchData("device-status");
36              setDeviceStatus(statusData);
37          };
38
39          const getSensorData = async () => {
40              const data = await fetchData("sensor-data");
41              const formattedData = data.map((sd) => ({
42                  ...sd,
43                  dateTime: `${new Date(sd.updated_at).toLocaleDateString()} ${new Date(
44                      sd.updated_at
45                  ).toLocaleTimeString()}`,
46              }));
47              setSensorData(formattedData);
48              setLastUpdate(new Date().toLocaleString());
49          };
50
51          getDeviceStatus();
52          getSensorData();
53          const statusInterval = setInterval(getDeviceStatus, 10000);
54          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

A series of white, overlapping geometric lines and polygons on a black background, located on the left side of the slide.

# THANK YOU

[ajayiayomide247@gmail.com](mailto:ajayiayomide247@gmail.com)