

```

1  from time import sleep
2  import dht
3  from machine import Pin, I2C, ADC
4  from lcd_api import LcdApi
5  from i2c_lcd import I2cLcd
6  import network
7  import uasyncio as asyncio
8  import socket
9
10 ssid = 'DE-LAB'
11 password = None
12
13 sta = network.WLAN(network.STA_IF)
14 sta.active(True)
15 sta.connect(ssid, password)
16
17 while not sta.isconnected():
18     pass
19
20 print('Connection successful')
21 print(sta.ifconfig())
22
23 def read_gas_level():
24     adc = ADC(1)
25     gas_value = adc.read_u16()
26
27     co2_ppm = gas_value * 20 - 60
28     o2_ppm = gas_value * 10 - 30
29
30     return co2_ppm, o2_ppm
31
32 def read_temperature_humidity():
33     dht_pin = Pin(2, Pin.IN)
34     d = dht.DHT11(dht_pin)
35
36     try:
37         d.measure()
38         return d.temperature(), d.humidity()
39     except OSError as e:
40         print("Failed to read from DHT sensor:", e)
41         return None, None
42
43 def display_on_lcd(temp, humidity, co2, o2, air_quality_value):
44     I2C_ADDR = 0x27
45     I2C_NUM_ROWS = 2
46     I2C_NUM_COLS = 16
47
48     i2c = I2C(0, sda=Pin(16), scl=Pin(17), freq=400000)
49     lcd = I2cLcd(i2c, I2C_ADDR, I2C_NUM_ROWS, I2C_NUM_COLS)
50
51     lcd.clear()
52     lcd.move_to(2,0)
53     lcd.putstr(f"EnviroMonitor")
54     sleep(5)
55     lcd.clear()
56     lcd.move_to(0,0)
57     lcd.putstr("Real-time Environmental Sensing")
58     sleep(5)
59     lcd.clear()
60     lcd.move_to(0,0)
61     lcd.putstr(f"Temperature:")
62     lcd.move_to(0,1)
63     lcd.putstr(f"{temp} Celsius")
64     sleep(3)
65     lcd.clear()
66     lcd.move_to(0,0)
67     lcd.putstr(f"Humidity:")
68     lcd.move_to(0,1)

```

```

69     lcd.putstr(f"{humidity}%")
70     sleep(3)
71     lcd.clear()
72     lcd.move_to(0,0)
73     lcd.putstr(f"Carbon-di-Oxide:")
74     lcd.move_to(0,1)
75     lcd.putstr(f"{co2} ppm")
76     sleep(3)
77     lcd.clear()
78     lcd.move_to(0,0)
79     lcd.putstr(f"Oxygen:")
80     lcd.move_to(0,1)
81     lcd.putstr(f"{o2} ppm")
82     sleep(3)
83
84     lcd.clear()
85     lcd.move_to(0,0)
86     lcd.putstr(air_quality_value)
87
88 def air_quality(CO2):
89     if CO2 < 800:
90         return "Fresh air"
91     else:
92         return "Not a Fresh air"
93
94
95 def determine_weather_condition(temp):
96     if temp > 30:
97         return "Hot"
98     elif temp > 20:
99         return "Moderate"
100    else:
101        return "Cool"
102
103 def web_page(temp, hum, co2, o2, air_quality_value):
104     weather_condition = determine_weather_condition(temp)
105
106     html = f"""<!DOCTYPE html>
107     <html lang="en">
108     <head>
109         <meta charset="UTF-8">
110         <meta name="viewport" content="width=device-width, initial-scale=1.0">
111         <title>Environmental Sensors Dashboard</title>
112         <style>
113             body {{
114                 background: url('https://media1.giphy.com/media/dYtHPYJxZblCcWPwcQ/giphy.gif') no-
repeat;
115
116                 background-size: cover;
117                 font-family: 'Segoe UI', Tahoma, Geneva, Verdana, sans-serif;
118                 margin: 0;
119                 padding: 0;
120                 background-color: #f2f2f2;
121                 display: flex;
122                 justify-content: center;
123                 align-items: center;
124                 height: 100vh;
125             }}
126
127             .container {{
128                 width: 90%;
129                 max-width: 800px;
130                 background-color: #fff;
131                 border-radius: 10px;
132                 box-shadow: 0 4px 8px rgba(0, 0, 0, 0.1);
133                 padding: 20px;
134                 margin: 20px;
135                 animation: slideIn 0.5s ease-out; /* Entrance animation */
136                 background: url('https://media1.giphy.com/media/dYtHPYJxZblCcWPwcQ/giphy.gif')
no-repeat;

```

```
136     background-size: cover;
137   }}
138
139   .header {{
140     color: white;
141     text-align: center;
142     font-size: 2rem;
143     margin-bottom: 20px;
144   }}
145
146   .main-content {{
147     text-align: center;
148     margin-bottom: 20px;
149   }}
150
151   .main-content img {{
152     width: 120px;
153     height: auto;
154     margin-bottom: 10px;
155   }}
156
157   .main-content .temperature {{
158     color: white;
159     font-size: 4rem;
160     font-weight: bold;
161   }}
162
163   .sub-content {{
164     display: flex;
165     justify-content: space-between;
166     flex-wrap: wrap;
167   }}
168
169   .sensor {{
170     background-color: transparent;
171     backdrop-filter: blur(16px);
172     border: 1px solid #ddd;
173     border-radius: 8px;
174     padding: 15px;
175     width: 45%;
176     margin-bottom: 20px;
177     box-shadow: 0 2px 4px rgba(0, 0, 0, 0.1);
178     animation: fadeIn 0.5s ease-out; /* Entrance animation */
179     position: relative;
180     overflow: hidden;
181     transition: transform 0.3s, box-shadow 0.3s;
182   }}
183
184   .sensor::before {{
185     content: '';
186     position: absolute;
187     top: 50%;
188     left: 50%;
189     width: 0;
190     height: 0;
191     background-color: rgba(255, 255, 255, 0.4);
192     border-radius: 50%;
193     transform: translate(-50%, -50%);
194     transition: width 0.3s, height 0.3s;
195   }}
196
197   .sensor:hover {{
198     transform: translateY(-5px);
199     box-shadow: 0 4px 8px rgba(0, 0, 0, 0.2);
200   }}
201
202   .sensor:hover::before {{
203     width: 200%;
204     height: 200%;
```

```

205     }}
206
207     .sensor .sensor_header {{
208         font-weight: bold;
209         font-size: 1.2rem;
210         margin-bottom: 10px;
211         text-align: center;
212     }}
213
214     .sensor .sensor_body {{
215         font-size: 1.5rem;
216         color: #333;
217         text-align: center;
218     }}
219
220     @keyframes slideIn {{
221         from {{
222             transform: translateY(-20px);
223             opacity: 0;
224         }}
225         to {{
226             transform: translateY(0);
227             opacity: 1;
228         }}
229     }}
230
231     @keyframes fadeIn {{
232         from {{
233             opacity: 0;
234         }}
235         to {{
236             opacity: 1;
237         }}
238     }}
239
240     @media screen and (max-width: 600px) {{
241         .sensor {{
242             width: 100%;
243         }}
244         .sensor .sensor_header {{
245             font-size: 1rem;
246         }}
247         .sensor .sensor_body {{
248             font-size: 1.2rem;
249         }}
250     }}
251 </style>
252 </head>
253 <body>
254     <div class="container">
255         <div class="header">Environmental Sensors Dashboard</div>
256
257         <div class="main-content">
258             <div class="temperature">{temp}°C</div>
259             <div>{weather_condition}</div>
260         </div>
261
262         <div class="sub-content">
263             <div class="sensor humidity">
264                 <div class="sensor_header">Humidity</div>
265                 <div class="sensor_body">{hum}%</div>
266             </div>
267             <div class="sensor CO2">
268                 <div class="sensor_header">CO2 - Percentage</div>
269                 <div class="sensor_body">{co2} ppm</div>
270             </div>
271             <div class="sensor O2">
272                 <div class="sensor_header">O2 - Percentage</div>
273                 <div class="sensor_body">{o2} ppm</div>

```

```
274         </div>
275         <div class="sensor quality">
276             <div class="sensor_header">Air Quality</div>
277             <div class="sensor_body">{air_quality_value}</div>
278         </div>
279     </div>
280 </div>
281 </body>
282 </html>
283 """
284     return html
285
286
287 addr = socket.getaddrinfo('0.0.0.0', 80)[0][-1]
288 s = socket.socket()
289 s.bind(addr)
290 s.listen(1)
291
292 print('Listening on', addr)
293
294 async def main():
295     while True:
296         co2_value, o2_value = read_gas_level()
297         print(f"CO2 (ppm): {co2_value:.2f}, O2 (ppm): {o2_value:.2f}")
298
299         temp, hum = read_temperature_humidity()
300         print("Temperature:", temp, "C")
301         print("Humidity:", hum, "%")
302         air_quality_value = air_quality(co2_value)
303         display_on_lcd(temp, hum, co2_value, o2_value, air_quality_value)
304
305
306         cl, addr = s.accept()
307         print('Client connected from', addr)
308         cl_file = cl.makefile('rwb', 0)
309         while True:
310             line = cl_file.readline()
311             if not line or line == b'\r\n':
312                 break
313
314             response = web_page(temp, hum, co2_value, o2_value, air_quality_value)
315             cl.send('HTTP/1.1 200 OK\r\nContent-Type: text/html\r\n\r\n')
316             cl.send(response)
317             cl.close()
318
319 loop = asyncio.get_event_loop()
320 loop.create_task(main())
321 loop.run_forever()
322
```