Smart Environmental Sensor OUTTX-110 Datasheet

1. Manufacturer Information

The Smart Environmental Sensor OUTTX-110 (Product Code: SES-OUTTX-110-2025) is developed by the Distributed System Lab at Okayama University, a leading research group focused on IoT and distributed systems. The lab is dedicated to creating innovative sensor solutions for smart environments, addressing global challenges in environmental intelligence.

- **Manufacturer**: Distributed System Lab, Okayama University
- **Address**: 3-1-1 Tsushima-naka, Kita-ku, Okayama 700-8530, Japan
- **Website**: https://www.okayama-u.ac.jp/ (Contact for lab-specific page)
- **Production Date**: Prototype Batch 2025-01, June 2025
- **Product Code**: SES-OUTTX-110-2025
- **Contact**: Email university administration for lab inquiries; support available via research portal
- **Mission**: Advancing IoT solutions for environmental monitoring, emphasizing reliability, scalability, and user-centric design
- **Support Process**: Technical support via research portal, typically within 48 hours for prototype inquiries

2. General Description

The Smart Environmental Sensor OUTTX-110 is a compact, IoT-enabled device that measures temperature, humidity, and atmospheric pressure, computing a Comfort Index to assess environmental quality. It supports dual data output: UART for local integration (e.g., Raspberry Pi) and HTTP POST for cloud/server integration. A built-in web server allows configuration of the target server IP, enhancing IoT flexibility.

The OUTTX-110 combines advanced sensing technologies with onboard processing for accurate, real-time data. Its modular prototype design supports customization, making it ideal for smart homes, weather stations, and industrial monitoring. With low-power operation, Wi-Fi connectivity, and a user-friendly web interface, it's a versatile solution for modern IoT ecosystems.

Key advantages include integrated Comfort Index calculation, dual communication protocols, and ease of configuration, catering to researchers, developers, and end-users.

3. Theory of Operation / Sensing Principle

The OUTTX-110 employs multiple sensing technologies to monitor environmental parameters. Temperature is measured using a thermistor, which varies resistance with temperature changes. Humidity is detected via a capacitive sensor, adjusting capacitance based on moisture levels. Atmospheric pressure is sensed using a piezo-resistive element, altering resistance under pressure variations. These signals are digitized and processed by an onboard microcontroller.

The microcontroller calculates a Comfort Index based on deviations from ideal conditions (22°C, 50% RH, 1013 hPa):

$$\text{Comfort Index} = 100 - (|\text{temp} - 22| * 2 + |\text{humidity} - 50| * 1.5 + |\text{pressure} - 1013| * 0.05)$$

Data is output in two formats:

- **UART**: CSV string (e.g., `25.5,60.2,1013.25,85.0`) at 115200 baud
- **HTTP POST**: JSON payload (e.g., `{"temp":25.5,"humidity":60.2,"pressure":1013.25,"comfort_index":85.0}`) to a configurable server (default: `http://192.168.0.69/data`)

A web server hosted on the sensor allows users to set the target server IP via a browser. The system integrates sensing, processing, Wi-Fi, and dual output interfaces for maximum flexibility.

4. Features

- Measures temperature (-40 to +80°C), humidity (0–100% RH), and pressure (300–1100 hPa)
- Computes real-time Comfort Index (0–100)
- Dual output: UART (115200 baud, CSV) and HTTP POST (JSON)
- Built-in web server for configuring target server IP
- Low-power operation with sleep mode (~60 µA)
- Wi-Fi connectivity for IoT applications
- Robust design for long-term stability
- Compact, breadboard-based prototype
- Customizable thresholds for alerts and automation
- High signal integrity for reliable data transmission

5. Potential Applications

The OUTTX-110 supports diverse applications, enhancing environmental control and monitoring:

- **Home Automation**: Optimizes HVAC based on Comfort Index, reducing energy use. *Example*: Triggers cooling when index <70.
- **Weather Stations**: Provides local data for forecasting. *Example*: Uploads to cloud for analysis.
- **Greenhouses**: Controls climate for plant growth. *Example*: Activates irrigation when humidity <60% RH.
- **Smart Offices**: Enhances occupant comfort via building management systems. *Example*: Adjusts ventilation based on index.
- **Health Monitoring**: Tracks indoor air quality for medical facilities. *Example*: Alerts when humidity >70% RH.

Table: Application Benefits

Application	Benefits
Home Automation	Energy-efficient climate control, enhanced comfort
Weather Stations	Accurate local data, cloud integration
Greenhouses	Optimized growth conditions, automated irrigation
Smart Offices	Improved productivity, occupant well-being
Health Monitoring	Real-time air quality data, health compliance

6. Pin Configuration and Description

The OUTTX-110 provides a minimal external interface:

- **VCC**: Power input (3.3V nominal)
- **GND**: Ground connection
- **Data Output Pin**: UART TX for local data transmission

Internal sensing and Wi-Fi interfaces are preconfigured, requiring no additional connections.

7. Absolute Maximum Ratings

Exceeding these ratings may damage the sensor:

- Supply Voltage: -0.3V to 6V
- Operating Temperature: -40 to +80°C
- Humidity: 0 to 100% RH (non-condensing)
- Pressure: 300 to 1100 hPa
- Storage Temperature: -50 to +100°C

Table: Absolute Maximum Ratings

Parameter	Limit
Supply Voltage	-0.3V to 6V
Operating Temperature	-40 to +80°C
Humidity	0 to 100% RH
Pressure	300 to 1100 hPa

8. Electrical Characteristics

- **Supply Voltage**: 2.2V to 3.6V (3.3V typical)
- **Logic Levels**: 3.3V for UART and internal interfaces
- **UART Parameters**: 115200 baud, 8 data bits, no parity, 1 stop bit
- **Wi-Fi**: 2.4 GHz, 802.11 b/g/n
- **Signal Integrity**: Low noise, stable output

Power Consumption Breakdown

- Active/Measurement: ~152 mA
- Idle/Sleep: ~60 µA
- Communication Active (UART + HTTP): ~162 mA

Table: Power Consumption

Mode	Current Draw

Active/Measurement	152 mA
Idle/Sleep	60 µA
Communication Active	162 mA

9. Operating Conditions

- Supply Voltage: 3.3V
- Temperature: -40 to +80°C
- Humidity: 0 to 100% RH (non-condensing)
- Pressure: 300 to 1100 hPa

Table: Operating Conditions

Parameter	Range
Supply Voltage	3.3V
Temperature	-40 to +80°C
Humidity	0 to 100% RH
Pressure	300 to 1100 hPa

10. Sensor Performance / Specifications

- **Temperature**:
  - Range: -40 to +80°C
  - Accuracy: ±0.5°C
  - Resolution: 0.1°C
- **Humidity**:
  - Range: 0 to 100% RH
  - Accuracy: ±2% RH
  - Resolution: 0.1% RH
- **Pressure**:
  - Range: 300 to 1100 hPa
  - Accuracy: ±0.12 hPa
  - Resolution: 0.03 hPa
- **Comfort Index**:
  - Range: 0 to 100
  - Formula: `100 - (|temp - 22| * 2 + |humidity - 50| * 1.5 + |pressure - 1013| * 0.05)`
- **Response Time**: ~2 seconds
- **Stability**: <0.1% RH/year, <0.05 hPa/year

Table: Sensor Performance

Parameter	Range	Accuracy	Resolution
Temperature	-40 to +80°C	±0.5°C	0.1°C
Humidity	0 to 100% RH	±2% RH	0.1% RH
Pressure	300 to 1100 hPa	±0.12 hPa	0.03 hPa

11. Communication Protocol / Interface

- **Internal**: Proprietary digital protocols
- **UART Output**:
  - Baud Rate: 115200
  - Format: CSV (`temp,humidity,pressure,comfort_index`)
  - Example: `25.5,60.2,1013.25,85.0`
- **HTTP POST**:
  - Endpoint: Configurable (default: `http://192.168.0.69/data`)
  - Format: JSON (`{"temp":float,"humidi`

`ty":float,"pressure":float,"comfort_index":float}`)
    - Example: `{"temp":25.5,"humidity":60.2,"pressure":1013.25,"comfort_index":85.0}`
- **Web Server**:
    - Access: Via sensor's local IP (e.g., `http://192.168.0.x`)
    - Function: Configure target server IP

12. Register Map

Internal registers manage calibration, control, and data, but are not user-accessible. Contact the manufacturer for details.

13. Package Information / Mechanical Dimensions

- Dimensions: ~100 × 80 mm (breadboard)
- Sensor Module: ~25 × 15 mm
- Materials: PCB core, potential plastic/metal enclosure
- Weight: ~50 g

14. Basic Usage / Application Information

14.1 Typical Connection Diagram

- VCC to 3.3V
- GND to ground
- Data Output Pin to external RX (e.g., Raspberry Pi RX)
- Wi-Fi: Connect to 2.4 GHz network

14.2 Required External Components

- Breadboard, jumper wires
- Optional: 4kΩ pull-ups for stability
- 3.3V power supply

14.3 Setup Instructions

1. Power the sensor (3.3V).
2. Connect to Wi-Fi (update SSID/password in firmware).
3. Access web server (`http://<sensor-ip>`) to set server IP (default: `192.168.0.69`).
4. Verify UART output (115200 baud) and HTTP POST.

14.4 Pseudo-Code

```
1. Initialize:
  - Start UART (115200 baud)
  - Connect to Wi-Fi
  - Start web server
  - Initialize sensor
2. Loop:
  - Read temperature, humidity, pressure
  - Calculate Comfort Index
  - Output via UART:
"temp,humidity,pressure,comfort_index"
  - Send HTTP POST: JSON
{temp,humidity,pressure,comfort_index}
  - Handle web server requests
  - Delay 2 seconds
```

14.5 Firmware Code Example

```
void setup() {
  Serial.begin(115200);
  // Connect to Wi-Fi
```

```
  WiFi.begin("SSID", "PASSWORD");
  while (WiFi.status() != WL_CONNECTED)
{ delay(1000); }
  // Start web server
  server.on("/", handleRoot);
  server.on("/set", HTTP_POST, handleSet);
  server.begin();
  // Initialize sensor
  if (!sensorInit()) { Serial.println("Sensor
failed"); while (1); }
}
void loop() {
  server.handleClient();
  float pressure = readPressure();
  float temp = readTemperature();
  float humidity = readHumidity();
  float comfort_index = 100 - (abs(temp - 22)
* 2 + abs(humidity - 50) * 1.5
                              + abs(pressure - 1013) *
0.05);
  // UART output
  Serial.print(temp); Serial.print(",");
  Serial.print(humidity); Serial.print(",");
  Serial.print(pressure); Serial.print(",");
  Serial.println(comfort_index);
  // HTTP POST
  HTTPClient http;
  http.begin("http://" + serverIP + "/data");
  http.addHeader("Content-Type",
"application/json");
  String payload = "{\"temp\":" + String(temp)
+ ",\"humidity\":" + String(humidity) +
                  ",\"pressure\":" +
String(pressure) + ",\"comfort_index\":" +
String(comfort_index) + "}";
  http.POST(payload);
  http.end();
  delay(2000);
}
```

14.6 Raspberry Pi Code Example (UART)

```
import serial
ser = serial.Serial('/dev/ttyS0', 115200,
timeout=1)
while True:
    if ser.in_waiting > 0:
        data =
ser.readline().decode('utf-8').strip()
        temp, humidity, pressure, comfort =
map(float, data.split(','))
        print(f"Temp: {temp} °C, Humidity:
{humidity} %, Pressure: {pressure} hPa,
Comfort: {comfort}")
```

14.7 Server Code Example (HTTP POST)

```
from flask import Flask, request
app = Flask(__name__)
@app.route('/data', methods=['POST'])
def receive_data():
    data = request.json
    print(f"Received: Temp={data['temp']}°C,
Humidity={data['humidity']}%, "
          f"Pressure={data['pressure']}hPa,
Comfort={data['comfort_index']}")
    return "OK", 200
if __name__ == '__main__':
    app.run(host='192.168.0.69', port=80)
```

15. Troubleshooting

Table: Troubleshooting Guide

Symptom	Possible Cause	Solution
No UART output	Power disconnected	Check 3.3V and GND connections
	UART misconfigured	Verify 115200 baud, correct RX connection

No HTTP POST	Wi-Fi disconnected	Check SSID/password, signal strength
	Invalid server IP	Access web server to set correct IP
Erratic readings	Loose connections	Secure wiring
	Noise interference	Add 4kΩ pull-ups
Sensor not initializing	Faulty module	Contact manufacturer
Web server inaccessible	Wi-Fi issue	Verify sensor IP, network connectivity

Steps:

1. **No Output**: Check power, UART settings, Wi-Fi connection.
2. **HTTP Issues**: Verify server IP via web interface, ensure server is running.
3. **Erratic Data**: Secure connections, reduce interference.

16. Reliability and Calibration

- **Reliability**: Minimal drift (<0.1% RH/year, <0.05 hPa/year), tested from -40 to +80°C.
- **Calibration**: Recalibrate every 6–12 months in 22°C, 50% RH, 1013 hPa. Compare to reference, adjust offsets via firmware (contact manufacturer).

17. Integration Examples

Cloud: Send data to ThingSpeak via HTTP POST.

```
import requests
while True:
    data = readSensorData()
    url =
f"https://api.thingspeak.com/update?api_key=KEY&field1={data[0]}"
    requests.get(url)
```

-
- **Smart Home**: Interface with Home Assistant for automation.
- **Server Logging**: Store HTTP POST data in a database.

18. Future Enhancements

- HTTPS support for secure transmission
- Authentication for HTTP POST
- Battery-powered operation
- IP54 enclosure for rugged environments

19. Compliance and Certifications

- Placeholder: Expected RoHS, CE, FCC compliance
- Verify with Distributed System Lab, Okayama University