MeshCore Sensor Data Integration

MeshCore Sensor Data Integration

MeshCore's SENSOR node variant transmits periodic environmental readings over the mesh using a lightweight binary packet format. This page describes how sensor data flows through the network, how to capture it at a room server, and how to build a Python integration that writes sensor records to InfluxDB or CSV for analysis.

SENSOR Variant Node Behaviour

A MeshCore node flashed with the SENSOR firmware variant enters a repeating sleep/wake cycle. On each wake event it reads all attached I2C sensors (BME680, SHTC3, or other supported devices), constructs a compact sensor payload, and transmits it as a standard MeshCore DATA packet addressed to the mesh broadcast ID. The node then returns to deep sleep until the next scheduled interval.

Sensor packets propagate through repeater nodes identically to chat messages, using the same store-and-forward and duplicate-suppression logic. Any node that receives the packet can decode it; a room server node provides a convenient aggregation point because it maintains persistent uptime and logs all traffic.

Sensor Data Flow Through the Mesh

1. SENSOR node wakes, reads BME680: T=21.8°C, H=62.3%, P=1011.4 hPa, IAQ=42
2. Node constructs a sensor DATA packet with a standard MeshCore header (sender ID, hop count, timestamp) and a typed sensor payload.
3. Packet is relayed hop-by-hop through repeater nodes toward the room server.
4. Room server receives and logs the packet. Its web interface displays the latest reading per node.
5. A background Python listener on the same machine reads room server output and writes to a time-series store.

Parsing Sensor Packets from Room Server Logs

The MeshCore room server logs incoming packets to stdout in a structured line format. A sensor packet line looks like:

```
2024-04-30T14:22:07Z SENSOR 0xDEADBEEF seq=1042 T=21.8 H=62.3 P=1011.4 IAQ=42 batt=3821
```

A simple Python parser to extract these fields:

```
import re, subprocess, datetime

SENSOR_RE = re.compile(
 r"(\S+) SENSOR (0x[0-9A-Fa-f]+) seq=\d+ "
 r"T=([\d.]+) H=([\d.]+) P=([\d.]+) IAQ=([\d.]+) batt=(\d+)"
)

def parse_sensor_line(line):
 m = SENSOR_RE.search(line)
 if not m:
 return None
 ts, node_id, temp, hum, pres, iaq, batt = m.groups()
 return {
 "time": ts,
 "node_id": node_id,
 "temperature": float(temp),
 "humidity": float(hum),
 "pressure": float(pres),
 "iaq": float(iaq),
 "battery_mv": int(batt),
 }
```

Writing to InfluxDB

```
from influxdb_client import InfluxDBClient, WriteOptions

INFLUX_URL = "http://localhost:8086"
INFLUX_TOKEN = "your-token-here"
INFLUX_ORG = "meshamerica"
```

```
INFLUX_BUCKET = "mesh_sensors"

client = InfluxDBClient(url=INFLUX_URL, token=INFLUX_TOKEN, org=INFLUX_ORG)
write_api = client.write_api(write_options=WriteOptions(batch_size=1))

def write_record(rec):
 point = (
 "environment"
 f",node_id={rec['node_id']}"
 f" temperature={rec['temperature']},"
 f"humidity={rec['humidity']},"
 f"pressure={rec['pressure']},"
 f"iaq={rec['iaq']},"
 f"battery_mv={rec['battery_mv']}"
 f" {int(rec['time_epoch'])}"
 )
 write_api.write(bucket=INFLUX_BUCKET, record=point)
```

Writing to CSV for Data Science Workflows

For ad-hoc analysis with pandas or R, a CSV sink is often more convenient than a full time-series database:

```
import csv, pathlib

CSV_PATH = pathlib.Path("/var/log/mesh_sensors.csv")

def append_csv(rec):
 write_header = not CSV_PATH.exists()
 with CSV_PATH.open("a", newline="") as f:
 w = csv.DictWriter(f, fieldnames=rec.keys())
 if write_header:
 w.writeheader()
 w.writerow(rec)
```

Load into pandas for analysis: `df = pd.read_csv("/var/log/mesh_sensors.csv", parse_dates=["time"])`. Standard pandas operations (resampling, rolling averages, correlation with external weather data) apply directly.

Integration with Python Data Science Tools

- **pandas** - Resample to hourly/daily averages, detect anomalies, compute sensor drift over time.
- **matplotlib / seaborn** - Plot temperature and humidity heatmaps across a sensor grid.
- **scikit-learn** - Train a simple regression to predict indoor temperature from outdoor readings. Useful for HVAC optimisation use cases.
- **Jupyter Notebook** - Combine data ingestion, analysis, and visualisation in a reproducible, shareable format ideal for community reporting.

Revision #2
Created 2026-05-03 06:11:41 UTC by Mesh America Admin
Updated 2026-05-03 13:02:52 UTC by Mesh America Admin