Got it — I updated the **entire GrainHero IoT plan** you pasted to use **VOC-based detection** (BME680 / tVOC) as the primary pest/spoilage proxy instead of CO_2, and I integrated the necessary changes everywhere (dataset mapping, sensor choices, calibration, thresholds, ML features, control logic, placement, sampling, etc.). This is a full, single-source replacement so you can copy it straight into your FYP doc or repo.

🌾 GrainHero – Complete System Design & Improvements Summary

(VOC-based pest/spoilage detection integrated — Final Integrated Explanation with Dataset, Hardware, Logic & AI Flow)

🧭 1. System Overview

GrainHero is an AIoT Smart Silo Management System that monitors, predicts, and controls spoilage risk in stored grains (rice & wheat). It combines IoT sensors (now VOC-first), data logging, and AI models to detect early signs of spoilage or pest infestation using environmental signals.

🧱 2. Hardware & Physical Design

⚙️ Prototype Build

- **Controller:** ESP32 board (with SD card + Wi-Fi).

- **Silo model:** Transparent plastic jar (5L+).
- **Fan:** 12V DC actuator fan for temperature and humidity control.
- **Probe:** Vertical, removable probe inserted from the top:
    - **Lower end (in-grain):** temperature, humidity, grain moisture sensors.
    - **Upper end (headspace above grain):** ambient temperature, humidity, **VOC sensor (tVOC / VOC index)** — primary pest/spoilage proxy (BME680 recommended).
- **Indicators:** Circular LED ring (Green / Yellow / Red) + buzzer for real-time alerts.
- **Wiring:** Modular and neat for quick removal/transfer between silos.

📋 3. Dataset & Parameters (as seen in your CSV)

The dataset has **11 parameters + meta labels**. All references to CO_2 are replaced by **VOC_index (or tVOC)**.

Parameter	Description	Source	Data Type

Temperature	Temperature of air/core	Direct	Sensor (SHT31 / BME680)
Humidity	Relative humidity (%) inside grain/headspace	Direct	Sensor (SHT31 / BME680)
Storage_Days	Days since stored	Derived	System clock / manual
Spoilage_Label	(Safe / Risky / Spoiled)	Derived	Manual / AI
Grain_Type	Rice = 1, Wheat = 2	Manual	Metadata
Airflow	Air movement proxy inside silo	Derived	From fan state (ON/OFF, RPM, PWM)

Dew_Point	Condensation threshold temp	Derived	Magnus formula (T + RH)
Ambient_Light	Light level above grain	Direct (opt)	BH1750 / LDR
Pest_Presence	Inferred insect/mold activity	Derived	**VOC patterns + T/RH + moisture**
Grain_Moisture	% water inside grain mass	Direct	Capacitive probe
Rainfall	External rainfall intensity	API	OpenWeatherMap / PMD

Parameter groups:

- **Direct (sensor):** Temperature, Humidity, Ambient_Light, Grain_Moisture, VOC_index.

- **Derived:** Dew_Point, Airflow, Pest_Presence (VOC-based).
- **External:** Rainfall (API).
- **Labels/Meta:** Spoilage_Label, Grain_Type, Storage_Days.

🌧️ 4. Handling of Grain Moisture & Rainfall

🌾 Grain Moisture

- **Sensor:** Capacitive moisture probe (e.g., SEN0193 / YL-69 style) placed in the lower half of grain.
- **Calibrations:** Dry baseline 0–10% → Risk >13–14% → Spoilage >=15%.
- **Output:** Analog → convert to % via calibration curve, record in `Grain_Moisture`. Feeds ML and triggers controls.

🌧️ Rainfall

- **No hardware sensor** — pulled from **weather API** (OpenWeatherMap / PMD).
- **Used for:** Prevent venting during rain or very high external humidity (disable fan actuation).
- **Update cadence:** 30–60 minutes; appended to dataset as `Rainfall`.

💨 5. Airflow & Fan Control (Derived Parameter)

🌀 Why Derived

Airflow measurement replaced by **fan telemetry** (on/off, PWM, RPM, current) — cheaper and robust for prototype.

⚙️ Calculation (example)

$$\text{Airflow} = \text{Fan\_Speed\_Factor} \times \text{Fan\_Duty\_Cycle}$$

- 1.0 = full airflow at 100% fan speed, 0.0 = fan OFF.

🔁 Fan Logic (VOC-aware)

- **ON** when: `RH_core > 65% AND Grain_Moisture > 14%` OR when ML risk score is high and external conditions permit.
- **OFF** when: `RH_core < 62% AND Grain_Moisture < 13.5%`.
- **Skip ventilation** if: external `Rainfall > 0` OR `Ambient_RH > 80%`.
- **Do not ventilate** if `T_core - Dew_Point_core < 1°C` (risk of condensation).
- **Hysteresis & min on/off times** to avoid short-cycling.
- **Log** fan_state, fan_duty, fan_rpm for ML.

💧 6. Dew Point Computation

- **Method:** Magnus formula calculated every 5 minutes from measured temperature & humidity.
- **Interpretation:** If `T_core - T_dew < 1°C` → condensation risk → avoid venting and flag for inspection.
- **Stored** as `Dew_Point` in dataset.

🌫️ 7. VOC Sensing (Primary Spoilage / Pest Proxy)

🌾 Why VOC

- **VOC (tVOC / VOC_index)** spikes (ethanol, aldehydes, ammonia, etc.) are early indicators of microbial activity and insect metabolism — often earlier and more cost-effective than CO_2 for your prototype.

🧠 Implementation

- **Primary sensor: BME680** (recommended): provides **T**, **RH**, **pressure**, **VOC_index** in one module.
- **Optional alternative: CCS811 + SHT31** for separate tVOC + eCO_2 + precise compensation if you later want CO_2.
- **Placement:** headspace ~5–10 cm above grain surface, away from direct fan stream, covered by breathable mesh to reduce dust.
- **Calibration:** 24-hour clean-air baseline → compute `VOC_relative = VOC_current - VOC_baseline_24h`. Monitor baseline drift

weekly/monthly.

- **Usage in dataset:** save `VOC_index` in existing gas/VOC column. Use `VOC_relative` and rate-of-change features for ML and derived `Pest_Presence`.

📈 **Example VOC thresholds (start values — tune during field tests)**

- **Yellow (early risk):** `VOC_relative_5min > 150` AND `VOC_rate_5min > 20`

- **Red (high risk):** `VOC_relative_5min > 300` OR (`VOC_relative_30min > 100` AND `Grain_Moisture > 14%`)

- Apply `rain_flag` and dew point checks to avoid false positives.

📊 8. Data Logging & Sampling Strategy

- **Raw sampling:** every **30 seconds** (T, RH, VOC, moisture, fan telemetry).

- **Averaging:** 5-minute averages stored to CSV (per dataset rule). Keep raw 30s logs on SD for ML expansion.

- **Per-row fields:** timestamp, silo_id, batch_id, T_core_avg_5m, RH_core_avg_5m, T_amb_avg_5m, RH_amb_avg_5m, Grain_Moisture_avg_5m, fan_state, fan_duty, VOC_index_avg_5m, dew_point_core, rainfall_last_hour, spoilage_label.

- **Extra processing:** rolling features (5m, 30m, 6h) and deltas (ΔTemp, ΔRH, VOC_rate).

- **Volume:** 8,640 5-min rows/month per silo; expand to 320k+ using raw 30s logs across time or parallel rigs.

🧠 9. ML Model Integration (VOC-first)

- **Model:** XGBoost as baseline (tabular); later ensemble with time-series models or image model if you add camera.

- **Inputs:** T_core, RH_core, dew_point, airflow/fan telemetry, VOC_index (and VOC_relative & rates), Grain_Moisture, rainfall_forecast, derived deltas.

- **Outputs:** Spoilage risk class (Safe / Risky / Spoiled). ML learns `Pest_Presence` implicitly from VOC + environmental patterns.

- **Key features:** VOC_relative, VOC_rate_5min/30min/6h, T_core - T_amb, dT_core/dt, dew_gap, moisture_trend_6h, fan_state_last_30min.

- **Training:** label rows with manual inspection (photo + oven-dry moisture) to build ground truth.

💡 10. Control Intelligence (VOC-informed)

- **Hybrid control:** deterministic rules + ML-guided advisories.

- **Alert mapping:** Green = Safe; Yellow = Early risk (VOC rise); Red = Spoiled (sustained VOC spike + moisture).
- **Actuation:** When Red or ML risk high, run fan (unless rain or high ambient RH or condensation risk). Buzzer if red persists > 10 minutes.
- **Logs:** All actuation decisions logged for model retraining and audit.

🔌 11. Hardware Summary (VOC-first)

Component	Function
ESP32 (with SD)	Central controller, logging, Wi-Fi
BME680 (recommended)	Temp, RH, pressure, **VOC_index** (primary pest proxy)
CCS811 + SHT31 (optional)	eCO_2 + tVOC + precise T/RH (upgrade path)
Capacitive moisture probe	In-grain moisture %
BH1750 (optional)	Ambient light (lid/tamper detection)

Fan (12V) + relay	Ventilation / dehumidifying actuator
LED ring + buzzer	On-site visual and audio alerts

🔒 12. Calibration & Maintenance (VOC-specific)

- **VOC baseline:** run sensor in clean air for 24 hours, store baseline. Compute rolling 24-hour baseline on device/backend.
- **Drift checks:** If baseline shifts > 20% weekly, schedule recalibration or hardware check.
- **Moisture probe:** calibrate with oven-dry samples for rice/wheat.
- **Maintenance:** clean mesh covers weekly; replace sensors if noise increases; monthly baseline review.
- **Power:** 3.3V sensors, isolated 5V for fan.

🌦️ 13. External API Integration

- Periodically fetch weather (rainfall, ambient humidity, temperature) every 30–60 minutes.
- **rain_flag** blocks venting if rain/humidity high. Append weather data to CSV rows for model context.

🚀 14. High-Level Impact Summary (VOC-based)

- **Lower cost & simplicity:** BME680 provides VOC + T + RH in one module.
- **Unified pest & spoilage detection:** VOC-first approach (with T/RH/moisture) reliably flags biological activity.
- **Smart fan logic:** dew-point + weather-aware ventilation reduces false positives and condensation risk.
- **Dataset unchanged:** VOC fits into existing gas/VOC column; derived features enrich ML without new columns.
- **Energy-efficient & explainable:** deterministic rules + ML ensemble.
- **Cross-crop compatible:** works for both rice & wheat.
- **FYP-ready:** demonstrates integrated AIoT, clear control logic, and practical prototypes.

🔧 Practical add-ons (VOC thresholds & example CSV row)

Suggested starting VOC rules (to tune in-field)

- **Yellow:** `VOC_relative_5m > 150 AND VOC_rate_5m > 20`

- **Red:** `VOC_relative_5m > 300 OR (VOC_relative_30m > 100 AND Grain_Moisture > 14)`

- Always suppress venting if `rain_flag == 1` or `Ambient_RH > 80%` or `T_core - Dew_Point_core < 1°C`.

Example 5-minute averaged CSV / MQTT JSON

- {
- "timestamp":"2025-11-13T10:05:00Z",
- "silo_id":"SILO01",
- "batch_id":"BATCH_A",
- "T_core":28.4,
- "RH_core":68.2,
- "T_amb":27.2,
- "RH_amb":66.9,
- "Grain_Moisture":13.8,
- "fan_state":1,
- "fan_duty":60,
- "VOC_index":210,
- "VOC_relative":95,
- "dew_point_core":23.1,
- "rainfall_last_hour":0,
- "spoilage_label":"Safe"
- }

If you want, I can now:

- generate the **ESP32 Arduino/PlatformIO skeleton** that reads **BME680 + capacitive moisture + T/RH probe**, computes 30s

samples → 5min averages, computes dew point & VOC_relative, logs to SD, publishes MQTT JSON and enforces the fan control rules above; **or**

- produce a **formatted one-page PDF** of this updated plan suitable for your FYP slides; **or**

- produce a **quick threshold tuning worksheet** and a short **test protocol** (what to run for the first 2 weeks to collect labeled data).
-