AI-Powered EV Battery Fire Prevention System – Hands-On Plan (Shareable Guide)

> Audience: Non-EE founders, student teams (Android, AI/ML, Hardware). Scope: Start with **two single lithium-ion cells** on the bench, stream real-time health to a **mobile app** via **MQTT**, then scale the same design to a small pack.

---

0) Safety First (read before buying)

- Work only with **one cell at a time** to begin. Never short a cell. Keep a **Class D/ABC fire extinguisher**, **sand/ceramic tray**, **LiPo-safe bag/metal box**, **safety glasses**, **heat-resistant gloves** nearby.
- Stay within datasheet limits: **charge to 4.20V (NMC/NCA)** or **3.65V (LFP)**, **cutoff ≥2.8–3.0V** (chemistry-dependent), **discharge current ≤0.5–1C** for home rigs, **cell temp ≤55 °C**.
- Work in ventilated area; never leave charge/discharge unattended.
- For packs: **do not mix cell brands/ages/SOH**; use a **BMS**; keep balance leads tidy and insulated.

---

1) System Overview (aligned with your diagram)

- **Team-3 (Data Acquisition)**: ESP32-based rig reads **Voltage/Current/Temperature** from each cell, computes **Capacity (mAh)**, **Internal Resistance (IR)**, basic **SOC/SOH**, and **publishes JSON** to MQTT (Wi-Fi).
- **MQTT Broker**: Eclipse Mosquitto (laptop/RPi/cloud).
- **Team-2 (AI/ML)**: Subscribes to telemetry → builds features → trains models (SOH regression, anomaly detection) → optional **inference service** (FastAPI) that sends results back via MQTT/HTTP.
- **Team-1 (Android)**: Subscribes to topics → live dashboard, trends, alerts; can also send **commands** (start/stop tests, set current).

---

2) What to Buy (for 2 single-cell rigs)

(Indicative India pricing; pick equivalent brands as available)

A) Safety & Bench Essentials

- LiPo-safe bag or metal ammo box; ceramic/metal tray – **₹1,000–1,800**
- Safety glasses and gloves – **₹400–800**
- Digital multimeter (basic, reliable) – **₹1,000–2,500**
- Heat-resistant tape (Kapton), cable ties, heat-shrink – **₹200–400**

B) Power, Charge & Load

- **Bench CC/CV supply** 0–30 V / 3–5 A (with current & voltage limit) – **₹3,000–8,000**
- OR **smart hobby charger** (e.g., SkyRC B6AC v2) for 1S Li-ion/LFP – **₹4,000–6,000**

- **Programmable electronic load** (easy) – **₹2,500–6,000** (e.g., DL24P)
- *DIY alternative*: logic-level MOSFET + 2–5 Ω/50 W resistor + heatsink + PWM – **₹400–800**

C) Cells & Fixtures

- Two new **18650 or 21700** cells (same chemistry; start with **NMC 2500–3000 mAh** or **LFP 2800–3300 mAh**) – **₹600–1,200** total
- **Cell holders** (18650/21700) – prefer **4-wire/Kelvin** style – **₹150–350** each
- Inline mini blade **fuses** (3–5 A) + holders – **₹150–300**

D) Sensing & Control (per cell)

- **ESP32 dev board** (ESP32-WROOM) – **₹800–1,200** × 2
- **Current/voltage sensor**:
- Easy: **INA219** module (0–3.2 A typical with included shunt) – **₹250–400**
- Better: **INA226** + **10 mΩ / 3–5 W precision shunt** – **₹400–800** total
- **High-res ADC (optional)**: **ADS1115 16-bit** for Kelvin voltage sense – **₹250–350**
- **Temperature**: **DS18B20** probe or **NTC 10 kΩ** + divider – **₹150–250**
- **Wiring**: 18–20 AWG silicone wire, banana→alligator leads, Dupont jumpers – **₹400–700**

E) Optional Gateway / Edge

- **Raspberry Pi 4/5** (runs Mosquitto + AI inference) – **₹4,500–9,000**
- *Not mandatory if you use a laptop for broker & AI.*

Approx total (2 rigs, budget path): ₹8k–₹15k (excluding RPi).

---

3) Wiring (single-cell rig)

1. **Discharge path**: Cell(+) → shunt (INAx26 sense) → electronic load/MOSFET → Cell(−).
2. **Kelvin voltage sense**: Two thin sense wires **directly at cell terminals** → ADC/INA VIN+ / VIN−.
3. **Temperature**: Tape DS18B20/NTC to the cell can (halfway up). Route signal to ESP32 (with pull-up for DS18B20).
4. **Fuse**: Put a 3–5 A fuse in series with the positive lead close to the cell holder.
5. **ESP32**: I^2C to INA219/226 (& ADS1115 if used). One GPIO (PWM) to load MOSFET gate (if DIY load). Common ground.
6. **Charging**: Use bench supply/charger **separately from discharge rig**. Never connect charger and load simultaneously.

   Tip: Keep power leads short and thick; sense leads thin and away from current path. Label everything.

---

4) Firmware (ESP32 – Team-3)

Sampling & Control - Loop @ **10 Hz** (start simple): read **V (mV), I (mA), T (°C)**. - Integrate capacity during discharge: `mAh += I_mA * dt_hours`. - **IR pulse test** (quick health): step current by ΔI for ~1–2 s, capture ΔV immediately after step → `R_int = ΔV/ΔI`. - Safety: if `T>55 °C` or `V<V_cutoff` or `I>I_limit` → **STOP** (disable load, publish fault).

MQTT Topics (example) - Telemetry: `ev/battery/<rigId>/cell/<cellId>/telemetry` - Events/Alarms: `ev/battery/<rigId>/cell/<cellId>/events` - Commands (ESP32 subscribes): `ev/battery/<rigId>/cell/<cellId>/cmd` - Status/heartbeat: `ev/battery/<rigId>/cell/<cellId>/status`

Command Set (JSON payload)

```
{ "cmd": "start_capacity", "i_set_mA": 1500, "v_cutoff_mV": 3000 }
{ "cmd": "stop" }
{ "cmd": "ir_pulse", "i_base_mA": 200, "i_step_mA": 1000, "dur_ms": 1500 }
```

Telemetry Payload (publish every 100–200 ms during tests)

```
{
  "ts": "2025-09-02T10:30:15.125Z",
  "rigId": "rig01", "cellId": "cellA",
  "chem": "NMC", "rated_mAh": 3000,
  "V_mV": 3985, "I_mA": 1480, "T_C": 32.4,
  "soc_pct": 62.4, "cum_mAh": 1125.7,
  "r_int_mohm": 42.7,
  "flags": {"discharging": true, "fault": false}
}
```

Derived Metrics - **Capacity (mAh)** at cutoff.
- **SOH_cap (%)** = `capacity_measured / capacity_rated_new × 100`. - **R_int (mΩ)** from pulse; track trend vs baseline.

---

5) MQTT Broker & Cloud (DevOps)

- **Mosquitto** on laptop/RPi: default port **1883**; enable **username/password**.
- Retain telemetry last-will for status; set QoS=1 for commands/events; telemetry QoS=0/1.
- Optional: bridge to cloud (AWS IoT/Core, EMQX, HiveMQ) later.
- Log to **Parquet/CSV** using a small Python subscriber for dataset creation.

---

6) Android App (Team-1)

Stack: Kotlin + Jetpack Compose, **MQTT client** (HiveMQ or Eclipse Paho), **MPAndroidChart** for plots, **Room** for history.

Screens 1. **Connect**: broker host, creds; choose rig/cell topics; persist settings. 2. **Live Dashboard**: tiles (V/I/T/SOC/Capacity/IR/SOH), traffic-light health. 3. **Charts**: real-time & historical (V/I/T/time; capacity curve; IR over cycles). 4. **Controls**: Start Capacity, IR Pulse, Stop; choose discharge current & cutoff. 5. **Alarms**: High temp, undervoltage, overcurrent; push notifications.

Local Validation - Show **sampling rate** & **last message age**.
- Warn if telemetry stalls >2 s.

---

7) AI/ML Pipeline (Team-2)

Phase-A (rule-based, fast) - Features: `V, I, T, dV/dt, dT/dt, capacity_so_far, IR`.
- Rules: temp > 55 °C or dT/dt spike → **anomaly**; IR above baseline by >25% → **aging**; capacity < specified % → **degradation**.

Phase-B (supervised model) - Labels: `SOH_cap` from full capacity tests; `IR_ref` from early cycles.
- Train a regressor (**XGBoost/RandomForest**) to predict `SOH` from short discharge segments + IR + temp features (reduces need for full cycle every time).

Phase-C (inference service) - Serve via **FastAPI**; endpoint `/infer` takes a recent window of telemetry, returns `soh`, `risk_score`, `explanations`.
- Publish back to MQTT: `ev/battery/<rigId>/cell/<cellId>/ai`.

Data Hygiene - Calibrate sensors first (zero-current offset, shunt gain).
- Downsample to 5–10 Hz; use **resampled timebase**; handle missing data.

---

8) Step-by-Step Procedure (single cell)

1. **Rig assembly**: mount cell in holder; wire shunt & load; route Kelvin sense; attach temp sensor; insert fuse.
2. **Sanity checks** (no cell yet): power ESP32 only → verify sensor I²C; verify MQTT publish; dry-run commands.
3. **Insert cell** (partially charged ~3.7–3.9 V). Confirm correct polarity; measure with DMM.
4. **Charge to full** (CC/CV) using charger; **rest 30–60 min** (for OCV stabilization).
5. **Start capacity test** from Android: set **0.5 C** current & **cutoff 3.0 V** (NMC) / **2.8 V** (LFP). Watch live V/I/T.
6. **Completes**: app shows **capacity (mAh)**, curve, and computed **SOH_cap**. Auto-save JSON/CSV.
7. **IR pulse**: at mid-SOC (40–70%), run **ΔI step** to compute `R_int`. Repeat 3×, take median.
8. **Monthly repeat**: track capacity fade & IR rise; export plots for AI dataset.

---

9) Scale to Two Cells & Mini-Pack

- Duplicate rig (rig02/cellB). Run tests independently.
- For **2S pack demo**: keep **cells matched**; use a **2S BMS board** (with balance). Measure **pack current** with one shunt, and **per-cell voltages** via **ADS1115** through a safe resistor divider network referenced to pack negative (*only if you're comfortable; else stick to single-cell until supervision is available*).

---

10) Calibration & Accuracy

- **Voltage**: compare ADC vs DMM at 3 points (e.g., 3.0/3.7/4.2V). Store linear gain/offset.
- **Current**: run known current through shunt (use load's ammeter) and adjust INA gain.
- **Temperature**: two-point check (ice-water ~0 °C, warm ~45 °C).
- Document calibration constants per rig in firmware.

---

11) Acceptance Criteria (MVP)

- MQTT telemetry from **both cells** at **≥5 Hz** with <2 s gaps.
- Capacity measurement error **≤±5%** vs hobby charger log.
- IR pulse repeatability **≤±10%** across 3 runs.
- Android app: live dashboard, charts, command control, history export.
- AI Phase-A: rules flag obvious anomalies (over-temp, undervoltage, high-IR).

---

12) Week-by-Week Plan (6 weeks)

Week-1
- Buy parts, set up Mosquitto, repo structure, Android project skeleton.
- Flash ESP32 "Hello MQTT" & sensor read. Safety training.

Week-2
- Complete single-cell rig wiring. Telemetry → MQTT → Android live tiles.
- Implement commands (start/stop/ir_pulse). Basic safety cutoffs.

Week-3
- Capacity integrator + CSV logging; Android charts; first full discharge test.
- Start Python subscriber → save Parquet/CSV dataset; notebooks for EDA.

Week-4
- Build second rig; calibration of both. Implement IR pulse routine & median logic.
- Rule-based anomalies & Android alerts.

Week-5
- Feature engineering; train first SOH regressor; cross-validation; - Package FastAPI inference & publish `…/ai` topic; Android consumes AI results.

Week-6
- (Optional) 2S pack demo with BMS; app view for multi-cell; write v1 report & demo video.

---

13) Documentation & Data Schemas

File naming: `cellId_cycleYYMMDD_runN.csv`.
CSV columns: `ts,V_mV,I_mA,T_C,cum_mAh,soc_pct,r_int_mohm,flags`.

Metadata JSON (per test)


```
{ "rigId":"rig01", "cellId":"cellA", "chem":"NMC", "rated_mAh":3000,
  "cutoff_mV":3000, "i_set_mA":1500, "notes":"0.5C capacity test" }
```


---

14) Common Pitfalls & Fixes

- **Voltage jumps/noise** → twist sense wires; move them away from power leads; average 3–5 samples.
- **Hot resistors/MOSFET** → add heatsink/fan; reduce current.
- **Capacity too low unexpectedly** → cell not fully charged/rested; cutoff too high; current not held constant; shunt calibration off.
- **IR inconsistent** → ensure steady baseline current; measure ΔV within ~50–150 ms of step; repeat and median.
- **Wi-Fi drops** → keep local buffer (ring-queue) and resend on reconnect.

---

15) Next Steps (what we can deliver quickly)

- ESP32 firmware skeleton (sensors, MQTT, commands, safety).
- Android Compose UI template (connect, dashboard, charts, commands).
- Python subscriber + starter notebook for EDA & rule-based alerts.
- Wiring diagram PDF + BOM spreadsheet with links.

Notes

- Start with **single cells only** until the team demonstrates consistent, safe test cycles.
- When moving to packs, use **proper BMS front-ends** (TI BQ769x, ADI/LTC68xx) and isolation practices; reassess safety.

FINALIZED STEP-BY-STEP PLAN (v2)

> Use this as the student handout. It includes roles, week-wise milestones, acceptance tests, MQTT/JSON specs, repo layout, and safety gates. Start with **single cells only**.

A) Roles & Deliverables

Team-1 (Android App) - Deliver a Compose app with: Connect, Live Dashboard, Charts, Controls, History Export. - Subscribe to MQTT, parse JSON, show alarms, send Commands.

Team-2 (AI/ML) - Deliver a Python pipeline: MQTT subscriber → dataset (CSV/Parquet) → EDA → Rule engine → SOH regressor → FastAPI inference → publish `…/ai` topic.

Team-3 (Hardware/ESP32) - Deliver two single-cell rigs with calibrated sensing; ESP32 firmware: telemetry, commands, safety cutoffs, IR pulse, capacity integrator.

B) Procurement Checklist (per *rig*)

- ESP32 dev board; INA219/INA226 + 10 mΩ shunt; DS18B20 probe; cell holder (Kelvin preferred); fuse (3–5 A) + holder; DIY load (MOSFET + 2–5 Ω/50 W resistor + heatsink) or programmable load; 18–20 AWG silicone wire; Kapton tape; heat-shrink; Dupont jumpers.
- Shared: Bench CC/CV supply or hobby charger; LiPo-safe bag/metal box; ceramic tray; safety glasses + gloves; DMM; small fan/heatsink for load.
- Optional lab gateway: Raspberry Pi 4 with Mosquitto + FastAPI.

C) Lab Setup

1) Place LiPo bag/metal box on ceramic tray; keep extinguisher handy. 2) Configure Mosquitto broker (laptop or RPi). Create user/pass. Enable persistence. 3) Label benches: **rig01**, **rig02**... Stickers for cell IDs: **cellA**, **cellB**.

Mosquitto quickstart (Linux/macOS)

```
brew install mosquitto   # or apt/yum
mosquitto -v              # dev mode; for prod use a config with
password_file
# Test
mosquitto_sub -t 'test' &
mosquitto_pub -t 'test' -m 'hello'
```

D) Wiring (single cell)

- Cell(+) → shunt → load/MOSFET → Cell(−).
- Kelvin sense wires from **cell terminals** to INA/ADS inputs.
- DS18B20 taped mid-can; route to ESP32 (3.3 V, GND, Data with 4.7 kΩ pull-up).
- 3–5 A fuse in series near holder.
- Keep power leads short; sense leads separated and twisted.

E) ESP32 Firmware (Team-3)

Libraries: `WiFi`, `PubSubClient` (or `AsyncMqttClient`), `OneWire` + `DallasTemperature`, driver for INA219/226.

Loop (10 Hz) 1. Read **V/I/T**.
2. If `discharging`: integrate capacity `mAh += I_mA * dt_h`.
3. Safety gates: `T>55°C` or `V<Vcut` or `I>Ilim` → stop + publish fault.

IR Pulse - Apply baseline current, step +ΔI for `dur_ms` (e.g., 1000–1500 ms). Measure ΔV right after step; compute `R_int=ΔV/ΔI`.

MQTT - Publish telemetry @ 5–10 Hz.
- Subscribe to `…/cmd` topic (JSON commands below).
- Heartbeat every 5 s on `…/status`.

F) MQTT Topics & JSON (FINAL)

Base: `ev/battery/<rigId>/cell/<cellId>/…`

Publish - `telemetry`

```
{ "ts":"2025-09-02T10:30:15.125Z", "rigId":"rig01", "cellId":"cellA",
  "chem":"NMC", "rated_mAh":3000,
  "V_mV":3985, "I_mA":1480, "T_C":32.4,
  "soc_pct":62.4, "cum_mAh":1125.7,
  "r_int_mohm":42.7, "discharging":true,
  "fault":null }
```

- `events` (alarms/info)

```
{ "ts":"…", "severity":"WARN", "code":"OVERTEMP", "detail":"T=58.2" }
```

- `status` (heartbeat)

```
{ "ts":"…", "fw":"1.0.0", "ok":true }
```

Subscribe (commands) — topic: `…/cmd`

```
{ "cmd":"start_capacity", "i_set_mA":1500, "v_cutoff_mV":3000 }
{ "cmd":"ir_pulse", "i_base_mA":200, "i_step_mA":1000, "dur_ms":1500 }
{ "cmd":"stop" }
```

AI Output (Team-2 publishes) — topic: `…/ai`

```
{ "ts":"…", "soh_pct":91.8, "risk":0.12,
  "explain":{"ir_mohm":44, "dTdt":0.8, "cap_pred":2860} }
```

G) Android App (Team-1) – Definition of Done

- **Connect Screen**: broker url, creds, topic base saved in `DataStore`.
- **Live Dashboard**: tiles (V/I/T/SOC/Capacity/IR/SOH) update ≥5 Hz; color status.
- **Charts**: V/I/T vs time; capacity curve; IR trend (last N tests).
- **Controls**: buttons → publish commands; show command acks.
- **Alarms**: toast + notification; log view.

- **History**: Room DB; export CSV for a selected cycle.
- **Offline**: cache last 1 h of telemetry.

H) AI/ML (Team-2) – Definition of Done

1. **Subscriber** writes Parquet/CSV with schema: `ts, V_mV, I_mA, T_C, cum_mAh, soc_pct, r_int_mohm, rigId, cellId`.
2. **EDA** notebook: plots, summary stats; save baseline IR per cell.
3. **Rules v1**: OVERTEMP, UNDERVOLT, HIGH_IR (>25% vs baseline), CAP_LOW (<80% rated). Publish `events`.
4. **Regressor v1**: Train/test split; metrics (MAE, R^2); model.pkl.
5. **FastAPI** `/infer`: accepts 60–120 s window; returns `soh_pct` & `risk`. Publishes to `…/ai`.

I) Calibration Procedure (record constants per rig)

- **Voltage**: compare at 3.0/3.7/4.2 V against DMM → compute linear gain/offset.
- **Current**: set 1.0 A on electronic load; read INA value → adjust shunt/gain.
- **Temperature**: two-point (ice water ~0 °C, warm ~45 °C) → save slope/offset.
- Store constants in a small `cal.json` in ESP32 flash.

J) Week-by-Week Schedule (6 Weeks)

Week-1 - Safety briefing & sign-off; broker install; repo bootstrap; hardware inventory. - ESP32 reads INA + DS18B20; publishes test telemetry → MQTT Explorer check. **Exit**: Telemetry visible; Android connects & shows dummy tiles.

Week-2 - Finish wiring; add commands; implement safety cutoffs; capacity integrator. - Android charts (realtime); CSV export.
Exit: One full capacity test captured end-to-end.

Week-3 - Build second rig; perform calibration on both; implement IR pulse + median. - Python subscriber stores datasets; EDA v1. **Exit**: Capacity error ≤±5% vs charger; IR repeatability ≤±10%.

Week-4 - Rule-based anomaly detector; Android notifications; historical views. - Begin regressor training; pick features; cross-validation. **Exit**: Rules firing correctly on induced cases (low cutoff, heated cell with hairdryer at safe temp, etc.).

Week-5 - Package FastAPI inference; integrate Android with `…/ai` topic. - Add session management: cycle IDs, notes, metadata. **Exit**: App shows AI SOH & risk in near-realtime.

Week-6 - Polish UX; write lab report; optional 2S pack demo with off-the-shelf BMS. **Exit**: Demo video + README + reproducible runbook.

K) Acceptance Criteria (MVP)

- Telemetry at ≥5 Hz; gaps <2 s; reconnect auto-recovers.
- Capacity within ±5% of reference; IR within ±10% repeatability.
- App can run start/stop/ir_pulse; charts + exports work; alarms visible.
- AI rules + regressor produce consistent SOH/risk; FastAPI latency <1 s.

L) Repo Layout (monorepo)

```
ev-fire-prevention/
  firmware/esp32/              # PlatformIO/Arduino sketch
  android/                      # Compose app
  ai/
    subscribers/                # MQTT→CSV/Parquet
    notebooks/                  # EDA & models
    service/fastapi/            # inference server
  docs/
    wiring.md, safety.md, bom.xlsx, api.md
```

M) Safety Gates (must pass before next stage)

1. **Wiring check** by mentor (photo + polarity test) before first power-on.
2. **Dry-run** with no cell: firmware telemetry/mqtt ok.
3. **First cell** at mid-SOC only; no charge/discharge until mentor approval.
4. **Never** charge & discharge connected simultaneously; attend cell at all times.

N) Demo Day Script (5–7 min)

1. Introduce rig & app; show live V/I/T.
2. Start capacity test at 0.5 C; show curves/alarms.
3. Run IR pulse; show IR computation; compare to baseline.
4. Show AI SOH & risk; export CSV; open a quick plot.
5. Summarize learnings, errors, and next steps (pack-level BMS).

Ready to execute. If you want, I can now generate: - PlatformIO **ESP32 firmware scaffold** with all MQTT topics/commands, - Android **Compose starter** with tiles+charts, - A small **Python subscriber** + sample notebook for Team-2.