Industrial Wearable AI — Idea & Approach (Strategic Plan)

Document type: Strategic idea, product definition, and build approach
Purpose: Define what to build, how it works, whom it targets, and how to make the idea more efficient, powerful, and production-ready.
Based on: Existing patent idea and planning docs; this document adds recommendations and a concrete execution plan.

Table of Contents

---

1. Executive Summary

1.1 Refined Vision

We are building: A **human-centric factory intelligence platform** that turns each worker's wrist into a real-time sensor of *human–machine interaction*, not just "movement." The output is **actionable intelligence** for supervisors and owners: who is active, blocked, fatigued, or at ergonomic risk — and why (e.g. humidity, thread breaks, process bottleneck).

Core thesis: Factories fail less because of machines and more because **human contribution is invisible**. This system makes human labor measurable, predictable, and optimizable at SME cost.

1.2 Why This Approach Is Stronger

Original idea	Recommended evolution
"Wearable for data collection"	**Platform:** Wearable → Edge → AI → Dashboard → Decisions; emphasis on *insights* and *actions*, not just data.
One-month PoC only	**Phased path:** Month 1 = proof-of-concept; Months 2–3 = pilot in one role (e.g. sewing); Month 4+ = production-ready for one factory, then scale.
Generic "textile unit"	**Lock one use-case first:** Sewing operator (wrist motion + idle vs active). Expand to cutter, packer, etc. only after one role works.
"Basic ML"	**Structured ML pipeline:** Labeled dataset → Activity classifier (primary) → Rule-based ergonomic/fatigue layer → Optional productivity model. Clear evaluation (accuracy, latency, false-alert rate).
Dashboard as output	**Supervisor-first UX:** One screen = "Who needs attention now?"; second = "Why is line slow today?"; third = trends and compliance.
Low cost assumed	**Explicit cost tiers:** Ultra-low (ESP32 + phone/laptop only); Standard (+ RPi edge + cloud); Optional cloud for multi-site.

1.3 One-Line Pitch

"Google Analytics for human labor in factories — starting with one wristband per worker, one job (sewing), and one month to prove it works."

---

2. What We Should Build (Product Definition)

2.1 Product Name (Working)

Industrial Wearable AI / **Human Factory Intelligence (HFI)** / **ShopFloor Pulse** — pick one for branding; technically we refer to the **platform**.

2.2 Scope: What Is In and Out

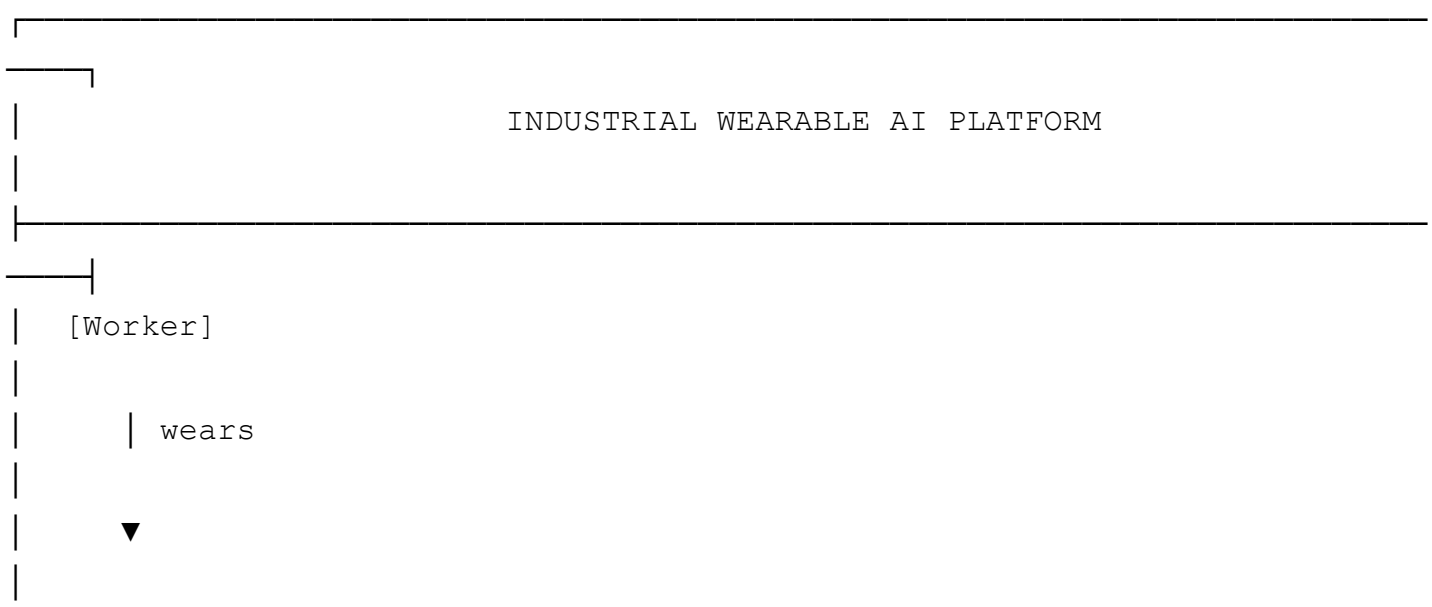
In scope (MVP → V1):

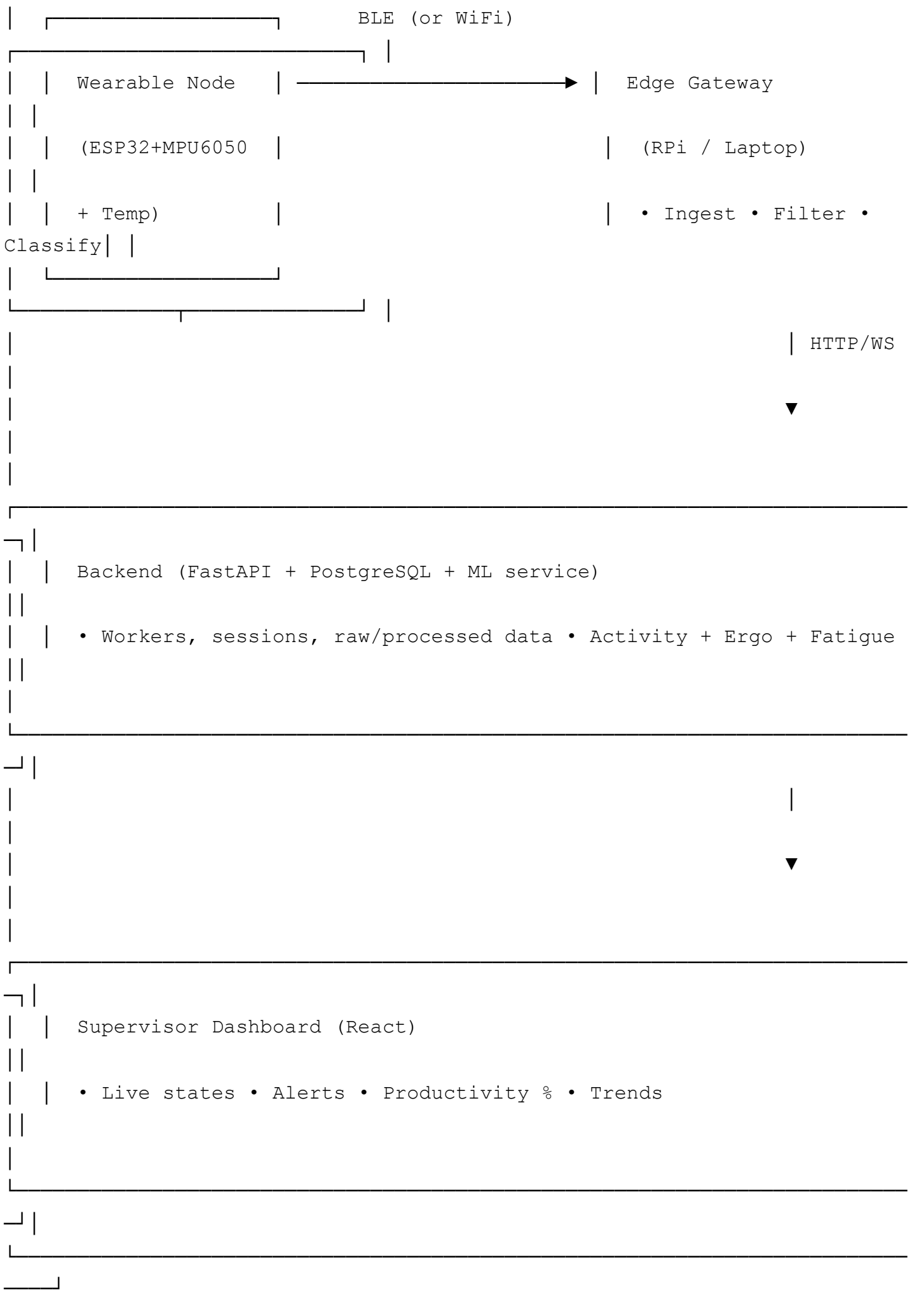
- **Wearable node:** Single form-factor (wrist band) with IMU (accelerometer + gyro) + temperature. Optional: heart rate later. **No AI on device** — only sense and stream.
- **One job role first:** Sewing machine operator. Clear motion vocabulary: sewing, idle, adjusting/fabric alignment, error/rework, break.
- **Edge gateway:** Receives BLE stream(s), runs noise filtering + segmentation + **activity classification** (Sewing | Idle | Adjusting | Error). Sends compact feature stream or labels to backend.
- **Backend:** API (FastAPI), auth, storage (PostgreSQL), ML pipeline (train offline, serve activity model + rule-based ergonomic/fatigue). Optional WebSocket for live stream.
- **Supervisor dashboard:** Web app (React). Primary view: **live worker states** (active / idle / adjusting / error / break) and **alerts** (fatigue risk, ergonomic risk). Secondary: simple productivity (active % per worker/shift) and trend charts.
- **Deployment:** Single-factory, on-prem edge + optional cloud sync for backup/analytics. No multi-tenant SaaS in MVP.

Out of scope (for later):

- Multiple job roles (cutter, packer, etc.) until sewing is validated.
- Full "productivity intelligence" (bottleneck root cause, humidity correlation) — start with correlation in dashboard, not complex causal ML.
- Heart rate hardware in MVP (add when fatigue model is prioritized).
- Mobile app for workers (optional later; MVP = supervisor web only).
- Compliance/audit reporting (V1.1).

2.3 System Boundary Diagram




BLE (or WiFi)
Wearable Node
(ESP32+MPU6050
+ Temp)
Edge Gateway
(RPi / Laptop)
• Ingest • Filter • Classify
HTTP/WS
Backend (FastAPI + PostgreSQL + ML service)
• Workers, sessions, raw/processed data • Activity + Ergo + Fatigue
Supervisor Dashboard (React)
• Live states • Alerts • Productivity % • Trends

2.4 Success Criteria for "Working"

- **Technical:** Wearable streams 10–50 Hz IMU + temp; edge classifies activity with ≥85% accuracy (on held-out labeled data); dashboard shows live state per worker with <5 s latency; no hard crashes in 8-hour shift.
- **User:** Supervisor can answer in <30 seconds: "Who is idle right now?" and "Who is at risk (fatigue/ergo)?" and "What was active % for the shift?"
- **Business:** One factory (or lab) runs a 1–2 week pilot with 2–5 workers; owner/supervisor agrees it is "useful for daily decisions."

---

3. Whom It Targets (Audience & Positioning)

3.1 Primary Target (Who Pays / Decides)

Segment	Description	Why they care	How we reach them
Textile SME owners (India, SEA)	Garment, powerloom, finishing units, 20–200 workers	Productivity and safety without big IT spend	Pilots, industry associations, word-of-mouth
Factory supervisors / floor managers	Day-to-day line management	"Who is blocked? Who is tired? Why is output low?"	Dashboard is built for them; training + support
Institutions / researchers	Engineering colleges, innovation labs	Proof of concept, publications, patents	Academic partnerships, hackathons (e.g. SIH), papers

3.2 Secondary (Who Uses / Benefits)

Role	Need	What they get
Workers	Safety, fair assessment, less "supervisor guess"	Indirect: fewer injuries (ergo alerts), clearer link between effort and visibility (optional future: worker-facing summary)
HR / compliance	Attendance, overtime, injury risk	V1.1: reports and audit trails
Management	Line efficiency, bottleneck visibility	Dashboard: active %, trends, alerts

3.3 Geographic & Segment Focus

- **First market:** Indian textile SMEs (garment, powerloom clusters). English + one local language for UI later.
- **Expansion:** Same product can later target Bangladesh, Vietnam, Sri Lanka (garment); then non-textile (assembly, warehouses, logistics) with new job-specific models.

3.4 Positioning Statement

For textile SME owners and supervisors **who** need visibility into worker activity and safety without expensive automation, **our platform** turns low-cost wearables into real-time human–machine intelligence **so that** they can reduce idle time, prevent injuries, and optimize the line — **unlike** machine-only IoT or CCTV, we measure the human side of production at a fraction of the cost.

---

4. How It Will Work (End-to-End Flow)

4.1 Data Flow (Step by Step)

1. **Worker** wears wristband (dominant hand for sewing). Powers on at shift start.
2. **Wearable** samples IMU (e.g. 25 Hz) + temperature (e.g. 0.2 Hz), batches into small packets, sends via BLE to gateway. No processing on device.
3. **Edge gateway** (RPi or laptop in same room):
    - Receives packets, validates, strips duplicates.
    - **Noise filter:** e.g. low-pass on accel/gyro.
    - **Segmentation:** sliding windows (e.g. 2–5 s) with overlap.
    - **Activity model:** each window → feature vector → classifier → label (Sewing | Idle | Adjusting | Error | Break).
    - Optionally runs **rule-based ergonomic/fatigue:** e.g. "idle > 2 min" → Idle alert; "sustained high wrist angle" → Ergo risk. Sends **labels + optional risk flags** to backend (not raw 25 Hz).
4. **Backend** stores:
    - Worker, session (shift), device id.
    - Time-series of labels (and optional raw for debugging).
    - Aggregates: active %, idle %, alert count per worker/session.
5. **Dashboard** subscribes (WebSocket or short polling):
    - **Live view:** each worker = current state + risk badge.
    - **Alerts panel:** fatigue/ergo alerts with time and worker.
    - **Shift view:** bar or table of active % per worker; simple trend (e.g. active % over last 7 days).

4.2 User Stories (Supervisor)

- "When I open the dashboard, I see which workers are active, idle, or at risk **right now**."

- "When someone is idle for more than X minutes, I get an alert so I can check if they're blocked."
- "At end of shift, I see active % per worker so I can compare performance and fairness."
- "When the system says 'ergonomic risk,' I know to check posture or rotation — **before** an injury."

4.3 Worker Experience (Non-intrusive)

- Wear band at start of shift; remove at end. No interaction needed with device.
- No personal data beyond "worker ID" and motion/temp (no camera, no voice). Explain "safety and productivity insight" in simple terms.
- Optional: one green/amber/red LED on band for "you're in risk" (V1.1) so worker can self-correct.

---

5. Technical Approach (Recommendations)

5.1 Architecture Principles

- **Edge does classification;** backend does storage, aggregation, and optional re-training. Reduces bandwidth and keeps latency low.
- **Wearable is dumb:** only sense + send. Keeps cost and complexity low; all intelligence in edge/cloud.
- **One activity model per job role.** First model: sewing operator. Same pipeline, different training data for cutter/packer later.
- **Offline-first labeling and training;** inference can be edge (real-time) and/or backend (replay, analytics).

5.2 Recommended Stack (Concrete)

Layer	Choice	Rationale
Wearable	ESP32 + MPU6050 (6-axis) + DHT11/DS18B20	Cheap, well-documented, BLE + WiFi. No ML on device.
Firmware	Arduino C++ or ESP-IDF	Simple loop: read IMU/temp → pack JSON → send BLE.
Edge	Raspberry Pi 4 or laptop (Python)	Python for ML (sklearn/lightweight model). Receives BLE, runs filter + segment + classify, POST/WS to backend.
Backend API	FastAPI	Async, WebSocket support, OpenAPI.
Database	PostgreSQL	Time-series of labels + aggregates; workers, sessions, devices.
ML (training)	Python: Pandas, scikit-learn or XGBoost	Activity = classification (window → label). Train on labeled CSV; export model (e.g. joblib/pickle) for edge.

Layer	Choice	Rationale
ML (inference on edge)	Same model in Python on RPi	If too slow, consider ONNX or lightweight C++ inference later.
Dashboard	React + Recharts (or Chart.js) + WebSocket or polling	Single-page: live map, alerts, shift summary.
Auth	JWT or simple API key for pilot	Multi-user and roles in V1.1.

5.3 Data Model (Minimal)

- **workers:** id, name, role (e.g. sewing), device_id (optional).
- **sessions:** id, worker_id, start_ts, end_ts, shift_label.
- **activity_events:** session_id, ts, label (Sewing|Idle|Adjusting|Error|Break), optional risk_flag.
- **aggregates:** session_id, active_pct, idle_pct, alert_count (computed on write or batch).

5.4 Security & Privacy

- **Data:** Motion and environmental data only; no PII beyond worker ID. Store in factory-owned or controlled backend; no unnecessary cloud in MVP.
- **Access:** Dashboard behind login; API keys for edge. No public endpoints.
- **Compliance:** Explain to workers what is collected and why (safety + productivity). Optional consent form for pilot.

---

6. AI/ML Strategy (Making It Accurate and Maintainable)

6.1 Model 1: Activity Recognition (Core)

- **Input:** Fixed-length windows (e.g. 2–5 s) of (ax, ay, az, gx, gy, gz) — optionally temp.
- **Output:** One label per window: Sewing | Idle | Adjusting | Error | Break.
- **Features:** Mean, std, min, max, zero-crossing rate, etc. per axis; or small raw windows if using a tiny CNN (later).
- **Algorithm:** RandomForest or XGBoost for MVP; possible 1D CNN if we have enough data and need better accuracy.
- **Data:** Labeled by **ground truth:** video of operator + sync timestamps; tag windows to labels. Need at least ~30–60 min of labeled data per operator (multiple operators better) for a stable model.
- **Evaluation:** Hold-out 20%; report accuracy and per-class F1. Target ≥85% accuracy; if not, add data or features.

6.2 Model 2: Ergonomic Risk (Rule-Based First)

- **Input:** Segment-level labels + gyro (wrist angle proxy) or derived posture.
- **Rules (examples):** "If wrist angle > X° for > Y min → Ergo risk"; "If static posture (low variance) for > Z min → Ergo risk."
- **Output:** Binary risk flag per segment or per minute. Dashboard shows "Ergonomic risk" for that worker.
- **Evolution:** Replace or augment rules with a small classifier when we have enough "injury near-miss" or expert-labeled risk data.

6.3 Model 3: Fatigue (Rule-Based First)

- **Input:** Activity stream + temperature + optional heart rate (later).
- **Rules (examples):** "If Idle % > threshold and high temp → possible fatigue"; "If long sustained sewing without break → fatigue risk."
- **Output:** Fatigue risk flag. Dashboard: "Fatigue risk."
- **Evolution:** Add HR when hardware is available; train a small classifier on "tired" vs "fresh" labels if we can collect them.

6.4 Data Pipeline (Labeling → Training → Deployment)

1. **Collect:** ESP32 streams to CSV (or DB) with timestamps; simultaneously record video (or manual log) of activity.
2. **Label:** Sync video to timestamps; assign label per window (semi-automated tool: play video, click label for segment). Export CSV: (start_ts, end_ts, label).
3. **Feature extraction:** From raw CSV, build windows and features; merge labels. Train/val/test split.
4. **Train:** Train classifier; evaluate; if OK, export model (joblib).
5. **Deploy:** Edge loads model; runs on live stream; sends labels to backend. Retrain when new data is available (e.g. weekly).

6.5 Improving Over Time

- **Active learning:** Flag low-confidence windows; supervisor or labeler confirms; add to training set.
- **Per-factory calibration:** Fine-tune on 1–2 days of labeled data from that factory (different machines, postures).
- **A/B tests:** Compare rule-based vs ML-based ergo/fatigue when we have enough labels.

---

7. Phased Build Plan (From PoC to Production)

Phase 0: Lock Use-Case and Design (Week 0)

- **Decide:** "We are building for **sewing operator** only. One wearable per person; wrist motion + idle/active/error."
- **Design:** Wireframes for dashboard (live view, alerts, shift summary). Data schema. API contract (REST + WebSocket).
- **Output:** One-page product spec + schema + wireframes.

Phase 1: Hardware & Data Pipeline (Weeks 1–2)

- **Hardware:** Order ESP32, MPU6050, DHT11/DS18B20; assemble wrist band; firmware: read sensors → BLE broadcast (or connect to one gateway).
- **Edge (laptop):** Python script: receive BLE → log CSV (timestamp, ax, ay, az, gx, gy, gz, temp). Validate 8-hour run without dropouts.
- **Labeling:** Record 30–60 min video + CSV; label segments (Sewing, Idle, Adjusting, Error, Break). Build labeled dataset.
- **Output:** Working wearable, CSV pipeline, labeled dataset (e.g. 500–2000 windows).

Phase 2: Activity Model & Edge Intelligence (Weeks 2–3)

- **Features + model:** Feature extraction from windows; train RandomForest/XGBoost; evaluate (accuracy, F1). Target ≥85%.
- **Edge app:** Same Python: receive BLE → filter → segment → classify → output labels (and optional risk rules). Send to backend via HTTP POST or WebSocket.
- **Backend (minimal):** FastAPI: POST /events (batch of labels); store in PostgreSQL. Optional: GET /workers, /sessions.
- **Output:** End-to-end: wearable → edge → DB. No dashboard yet.

Phase 3: Dashboard & Alerts (Weeks 3–4)

- **Dashboard (React):** Login (simple). Live view: list of workers with current state (from latest events). Alerts panel: ergo/fatigue from rules. Shift summary: active % per worker (from aggregates).
- **WebSocket or polling:** Dashboard gets updates every 2–5 s.
- **Output:** Supervisor can use dashboard in lab or pilot.

Phase 4: Pilot in Real Environment (Weeks 4–6)

- **Deploy:** 2–5 workers, one shift per day, 1–2 weeks. Edge = RPi or laptop on-site; backend on same machine or small server.

- **Support:** Fix bugs; tune alerts (reduce false positives). Collect feedback: "Is this useful? What's missing?"
- **Output:** Pilot report: accuracy in field, supervisor satisfaction, one testimonial or letter of interest.

Phase 5: Production-Ready (Months 2–3)

- **Stability:** Auto-restart edge service; basic monitoring (disk, BLE connection drops). Backup DB.
- **Docs:** User guide (supervisor), deployment guide (install edge + backend).
- **Optional:** Multi-worker BLE (multiple bands to one gateway); calibration flow (assign device to worker).
- **Output:** Package that another factory can install with minimal support.

Phase 6: Scale & Expand (Month 4+)

- **Second role:** Cutter or packer — new labeled dataset, new activity model; same pipeline.
- **Multi-site:** Backend in cloud; edge at each site pushes to cloud; dashboard shows multiple factories (optional).
- **Monetization:** Pilot free; then subscription per worker/month or one-time license per factory.

---

8. Risks & Mitigations

Risk	Mitigation
Activity model accuracy low in real factory	Collect more labeled data from target environment; add per-factory calibration; use confidence scores and "Unknown" class.
BLE range / dropouts	One gateway per room or line; buffer on wearable; reconnect logic. Test in actual floor layout.
Supervisor doesn't trust or use dashboard	Involve supervisor in design; make first screen "who needs attention now"; reduce alert noise (tune thresholds).
Worker resistance ("tracking")	Frame as safety + fairness; no surveillance (no camera); transparent communication; optional anonymized analytics.
Edge device (RPi) unreliable	Use UPS; auto-restart; log to local file + sync to backend when online.
Cost overrun	Stick to ESP32 + RPi/laptop; no custom PCB in MVP.

---

9. Success Metrics

Stage	Metric	Target
PoC (Month 1)	Activity classification accuracy (held-out)	≥85%
PoC	End-to-end latency (motion → dashboard)	<5 s
Pilot	Supervisor can answer "who is idle?" in	<30 s
Pilot	False alert rate (ergo/fatigue)	<20% of alerts
Pilot	Supervisor satisfaction (survey)	"Useful" or "Very useful"
Production	Uptime (edge + backend)	≥99% during shift
Business	Willingness to pay (letter of intent or paid pilot)	≥1 factory

---

10. Differentiation & Moat

- **Human-centric, not machine-centric:** We measure human–machine interaction; others measure machines or cameras. Clear patent angle: "human motion as primary factory sensor."
- **SME-first:** Low cost, minimal IT, quick deploy. Not competing with full MES/ERP.
- **Actionable, not just data:** Dashboard answers "who needs attention?" and "why?" — not just raw graphs.
- **Extensible:** Same platform can add cutter, packer, warehouse picker with new models; same dashboard and backend.
- **Moat over time:** Labeled datasets per role and per geography; tuned models; supervisor habits and trust.

---

11. Future Scope (Beyond V1)

- **More roles:** Cutter, packer, QC, helper — same wearable, new activity models.
- **Root-cause analytics:** Correlate idle % with humidity, machine breakdown, thread type — "Station 4 slow because …".
- **Worker-facing app:** Optional: worker sees own active % and tips (e.g. "Take a break").
- **Compliance & audit:** Export reports for HR, safety, certifications.
- **Other industries:** Warehouses (pick/pack), assembly lines, logistics — same Move/Hold/Repeat paradigm.
- **Hardware v2:** Smaller band, longer battery, optional heart rate, LoRa for large floors.

---

12. Summary: What to Build First

1. **One wearable:** ESP32 + MPU6050 + temp; BLE stream to edge.
2. **One job:** Sewing operator; labels: Sewing | Idle | Adjusting | Error | Break.
3. **One pipeline:** Edge (Python) does filter → segment → classify → send to backend.
4. **One backend:** FastAPI + PostgreSQL; store events and aggregates.
5. **One dashboard:** React — live states, alerts, shift active %.
6. **One pilot:** 2–5 workers, 1–2 weeks, one factory or lab.
7. **One success criterion:** Supervisor says "I use this to decide who needs help and why."

After that: add ergo/fatigue rules, improve model with more data, then add second role and scale.

This document is the recommended idea and approach. Implementation should follow the phased plan and iterate based on pilot feedback and metrics.