BATCH – 4

CAPSTONE PROJECT

SAFETY MONITORING IN HAZARDOUS GOODS WAREHOUSES

|  |  |  |
| --- | --- | --- |
| **Stage | Activity** | **Tool** | **Outcome** |
| **Stage 1:**  **Brainstorming** | **Group Discussion on requirements and solution model:**   * Challenges – Misidentification of parts, improper installations, manual verification delays   **FLASH CARD**   * Type: Industry 4.0, Smart Manufacturing * Domain: Automotive Quality Assurance & Process Optimization * Stakeholders: Line supervisors, Plant QA teams, OEMs, Automation teams * Technologies: PIR, IR, and Color sensors, MQTT, Cloud Computing, Machine Learning, Dashboards | A System that can ensure Safety in Hazardous Materials Warehouse that will be able to:   * Sensors collect data on temperature, humidity, gas, and fire. * Data is sent to ThingSpeak cloud and raw data is visualised * Machine Learning Algorithms will be using the data to help in predictive maintenance * Alerts are triggered via messages, buzzers, or indicator lights. * Automated actions are taken like turning off equipment or activating safety systems * All events are logged, and regular system checks ensure reliability. |
| Stage 2:  Idea Posting | **Sensors**:   * **DHT22** – for temperature and humidity * **Heat sensors** – for fire or high temperature detection   **Microcontroller**:   * Local edge processing (e.g., Raspberry Pi Pico W or ESP32)   **Cloud Platform**:   * **ThingSpeak** for data logging and visualization   **Technology Stack**  * **IoT Communication**: MQTT * **Cloud Services**: ThingSpeak * **Analytics**: Machine Learning (trend analysis, anomaly detection) * **Control Systems**: Alarms, relays, emergency lights | **Real-time Monitoring**:   * Continuous tracking of environmental parameters (temperature, humidity, gas levels).   **Predictive Alerts**:   * Machine Learning models analyze historical and real-time data to detect early signs of danger.   **Instant Alerts & Automation**:   * Automatic triggering of alarms, indicator lights, or system shutdowns upon anomaly detection.   **Data Processing**:   * Local processing on microcontrollers for quick response. * Cloud-based analysis via **ThingSpeak** for advanced insights.   **Efficient Data Communication**:   * Uses **MQTT protocol** for lightweight, real-time data transmission. |
| Stage 3:  Customer Mapping | **From the worker perspective**   1. Will this system immediately alert me if there’s a fire, gas leak, or other danger nearby? 2. What should I do when I receive an alert — is there a clear and simple safety protocol to follow? 3. Will the system still work and protect me if there’s a power failure or internet issue?   **From the Owner perspective**   1. How scalable is this system if I expand to more warehouses? 2. Can I monitor everything remotely from my phone or dashboard? 3. How often does the system require maintenance or calibration?   **From the Platform provider, Client/Supplier and Logistics manager**   1. How reliable is the system in detecting real-time hazards? 2. Can the system be customized based on the type of material stored or specific site conditions? | The system ensures comprehensive safety and efficiency across all stakeholders: Workers benefit from real-time monitoring, emergency routes, alerts, hazard segregation, and safety gear tailored to warehouse zones, along with automated safety mechanisms and emergency guidance. Owners gain cost-effective, durable industrial-grade sensors with phone alerts, automatic damage prevention. Clients and suppliers are assured of goods’ safety and easy accessibility, while platform providers focus on secure handling. Logistics operations prioritize quick, safe transportation, collectively safeguarding workers, goods, and processes. |
| **Stage 4:**  **Idea Mapping** |  | Observations & Gaps:  The ML model requires refinement to adapt to dynamic environmental norms of different warehouses.    Lack of a feedback loop from on-ground safety personnel limits improvements in system accuracy and usability.    There is over-reliance on cloud connectivity; implementing edge computing (e.g., on PICO PI W) for basic alerts would improve robustness.    Safety automation (sprinklers, exhausts, alarms) was proposed but not fully realized due to hardware limitations during the prototype stage.    Hazard mapping tools and heatmaps, while planned, require more real-world data to be fully effective. |
| **Stage 5:**  **Reflection** | The Safety Monitoring System for hazardous goods warehouses was evaluated based on its sensor integration, ML capability, real-time responsiveness, and user experience. The goal was to ensure proactive environmental monitoring and reduce accident risk through intelligent automation.   * System Functionality:   Successful integration of MQ2 and DHT22 sensors allowed for reliable collection of temperature, humidity, gas concentration, and fire detection data. Cloud connectivity (via Wi-Fi and MQTT/HTTP protocols) enabled real-time data logging and analysis.     * Machine Learning and Analytics:   Initial ML models implemented threshold-based anomaly detection and time-series forecasting for predictive alerts. While effective in identifying basic patterns, the models currently lack adaptive learning to adjust to evolving warehouse environments.   * User Experience and Dashboard:   The real-time dashboard proved useful for safety teams to visualize alerts, logs, and environmental trends. However, mobile app integration and voice alert systems are yet to be fully optimized for field use. | * Successfully integrates PIR, MQ2 and DHT22 sensors for reliable environmental monitoring. * Real-time data logging and alerts enabled through Wi-Fi and MQTT/HTTP cloud connectivity. * Basic ML models provide early hazard detection using threshold-based anomaly detection and forecasting. * ML models currently lack adaptive learning for evolving warehouse conditions. * Dashboard offers useful visualization of alerts and trends for safety teams. * Mobile app integration and voice alerts need improvement for better field usability. * Overall, the system delivers core safety functions but requires enhancements in ML adaptability and user accessibility. |
| **Stage 6:**  **Design of Modules** | **1. Sensor Module**  * **Purpose:** Collect environmental data. * **Components:**   + **MQ2 Gas Sensor** – Detects Gas presence.   + **PIR Sensor**– Measures Distance   + **DHT22** – temp & humidity sensor (cheap backup or for cross-checking). * **Interface:** Analog/Digital → GPIO (Pi Pico W)  **2. Processing & Control Module**  * **Purpose:** Process sensor data, apply thresholds/ML, trigger actions. * **Components:**   **Raspberry Pi Pico W**   * + **Functions:**     - Real-time data processing.     - Anomaly detection (threshold or lightweight ML model).     - Command control (e.g., turn on buzzer, emergency light). * **Interface:** Connects to all sensor modules and communication modules.  **3. Connectivity Module**  * **Purpose:** Send data to cloud and receive remote commands. * **Protocol:**   + MQTT (for lightweight, real-time updates)   + HTTP (for REST-based API communication) * **Wi-Fi:** Pico Pi in-built * **Fallback/Extension (optional):**   + GSM module (if Wi-Fi is unreliable)  **4. Alert & Response Module**  * **Purpose:** Trigger alerts on anomalies * **Components:**   + Buzzer / alarm   + Emergency lights (LED strip or beacon)   + Relay module (for controlling power to specific areas or devices)   + Phone alert  **5. Power Management Module**  * **Purpose:** Provide stable power to all components. * **Components:**   + 5V/3.3V regulator   + Rechargeable battery (optional)   + Protection circuit   + USB/Buck converter  **6. Enclosure & Deployment Module**  * **Purpose:** House all electronics safely * **Components:**   + Plastic/acrylic/metal box   + Cable management   + Mounting system (wall/pole-mounted)   + Weatherproofing (IP-rated if needed) | [Design](https://lucid.app/lucidspark/bcce0e5f-a7d5-45ed-80bf-d56829d25fb6/edit?page=0_0&invitationId=inv_44227b57-4ee3-48eb-835e-490a70e1e25a) |
| **Stage 7:**  **Resource Identification** | **Sensors:**  DHT22 (temp & humidity) - 121  MQ2 Gas Sensor 97  PIR Sensor 72  **Display:**  OLED Display 224  **Microcontroller:**  Raspberry Pi Pico W – 579  **Miscellaneous:**  Jumper Wires – 214  Breadboard - 54  **Total Budget** = 1307 | Hardware Components:   * Raspberry Pi Pico W * DHT 22 * MQ2 Gas Sensor * PIR Sensor * OLED Display * Jumper Wires * Breadboard   Software Components:   * ThingSpeak * Wokwi * GitHub |
| Stage 8:  Planning | 1. Sensor data Aquisition and cloud integration:   * Simulated sensor data such as temperature, vibration, or usage hours are collected and transmitted securely to the cloud. Real-time values are streamed via protocols like MQTT or HTTP to mimic IoT deployment.   2. Hardware implementation:   * Physical sensors (e.g., DHT22, gas sensor) are interfaced with a microcontroller (like Pi Pico W) to collect live operational parameters. Data is transmitted to the cloud for analysis and backup.   3. Fetching data and preprocessing:   * Raw sensor data is fetched from the cloud or local storage. Cleaning, normalization, and feature extraction are performed to prepare data for the predictive model.   4. Failure prediction:   * Raw sensor data is fetched from the cloud or local storage. Cleaning, normalization, and feature extraction are performed to prepare data for the predictive model. | 5. Predictive maintenance scheduling:   * Predicted failure timelines help schedule maintenance activities before breakdowns. This reduces downtime and optimizes maintenance costs.   6. Data analytics:   * Historical and real-time data are analysed to identify patterns, usage trends, and risk zones. Dashboards provide insights for decision-making and system optimization.   7. Version control:   * Project code, ML models, and data pipelines are tracked using Git. Version control ensures collaborative development, rollback safety, and organized codebase management. |
| Stage 9:  Re Design | |  | | --- | | 1. Raspberry Pi Pico W 2. Lightweight ML models (e.g., decision tree, moving average) 3. Feedback Button Module 4. ThingSpeak (Cloud Platform) 5. MQTT Protocol 6. Relay Module / LED & Buzzer | | Implement local/edge computing logic for basic safety checks.  Improve prediction accuracy for dynamic environments.  Gather user feedback on false positives/negatives to refine model.  Continue cloud-based data visualization and alert generation.  Efficient real-time data transmission and remote alert handling.  Enable layered alert responses and safety system activations. |
| Stage 10: Execution framework | 1. Data Flow 2. Control Logic 3. State Handling 4. Testing Protocols 5. Data Pipeline | Sensors → Pico W → Threshold check → MQTT publish → ThingSpeak/Cloud ML → Alert back to Pico  If alert=true, activate buzzer + LED + log timestamp + notify mobile  Defines states: Safe, Warning, Emergency. Create finite state machine logic on microcontroller.  Unit test each module, then integrate and run system-level tests with simulated hazards.  Store historical data -> run periodical ML training -> auto-update thresholds or models |
| Stage 11:  Micromodules | 1. Sensor Module 2. Control Module 3. Connectivity Module 4. Alert Module 5. ML/Analytics Module 6. Dashboard Module | Flame sensor script, DHT22 validation  Anomaly detection subroutine, buzzer trigger function, emergency switch override  MQTT publish/subscribe script, fallback GSM handler  LED pattern logic, sound escalation (buzzer level control), mobile alert API  Preprocessor (cleaning data), Simple ML model (thresholds, trend prediction), Accuracy monitor  ThingSpeak feed update, alert highlight, download logs button |