XIAMEN UNIVERSITY MALAYSIA

Course Code	:	SWE403
Course Name	:	Technology and Application of Internet of Things
Lecturer	:	Venantius Kumar Sevamalai
Academic Session	:	2025/04
Assessment Title	:	Group Project
Submission Due Date	:	7 July 2025

Prepared by :

Student ID	Student Name
AIT2209952	TAN YI ZHAO
AIT2209949	SIM WEN KEN
AIT2209607	LEONG TING YI
AIT2209955	THAN YOU SIANG

Date Received :

Feedback from Lecturer:

Mark:

Own Work Declaration

I/We hereby understand my/our work would be checked for plagiarism or other misconduct, and the softcopy would be saved for future comparison(s).

I/We hereby confirm that all the references or sources of citations have been correctly listed or presented and I/we clearly understand the serious consequence caused by any intentional or unintentional misconduct.

This work is not made on any work of other students (past or present), and it has not been submitted to any other courses or institutions before.

Signature: [illegible]

Date: 5/7/2025

Contents

Abstract

This report describes the development and operational assessment of a **Smart Safety Helmet** system which aims to improve construction site worker safety. The helmet system uses ESP32 microcontrollers to operate temperature and smoke and light and acceleration sensors which generate real-time hazard alerts through visual and audible and haptic feedback. The automatic headlamp system enhances visibility during nighttime operations and wireless cloud connectivity allows continuous remote monitoring and immediate incident response. The firmware architecture implements an event-driven non-blocking system which enables fast detection and dependable local feedback and continuous cloud communication. The helmet demonstrates successful hazard detection within 400 ms while maintaining operational stability throughout shifts and meeting strict security requirements through encrypted communication and robust data privacy measures. The proposed system shows great potential to enhance workplace safety through its ability to reduce emergency response delays and its scalability for various industrial settings.

Introduction

The safety of the staff that work on the construction site is one of the important things that we could not ignore. Construction staff always expose themselves to danger in an environment where the dangers are hidden. As their superiors we need to pay more attention to the safety measures at construction sites. However, we cannot guarantee accidents will not happen, even though we have improved the safety measures. In this case, we introduced a **smart safety helmet** that included a temperature sensor, a smoke sensor, a fall detection sensor and an automatic headlamp. These components able to help construction staff prevent hidden dangers like high temperature of environment or fire hazard. In addition, due to the large construction environment and each staff have different job scopes, therefore every member of staff will be split into different places in the construction site, and it will be difficult for superiors to real-time supervise them. To ensure that superiors can supervise their staff, we implemented wireless cloud link to the smart safety helmet. The wireless cloud link enables superiors to remote supervise staff because it can transfer staff real-time situations to cloud drive. In the other hand, another purpose that we implement wireless cloud link is to ensure that superiors able to get to their staff at the first time while accident happened to them. (falling from construction sites)

Problem Statement

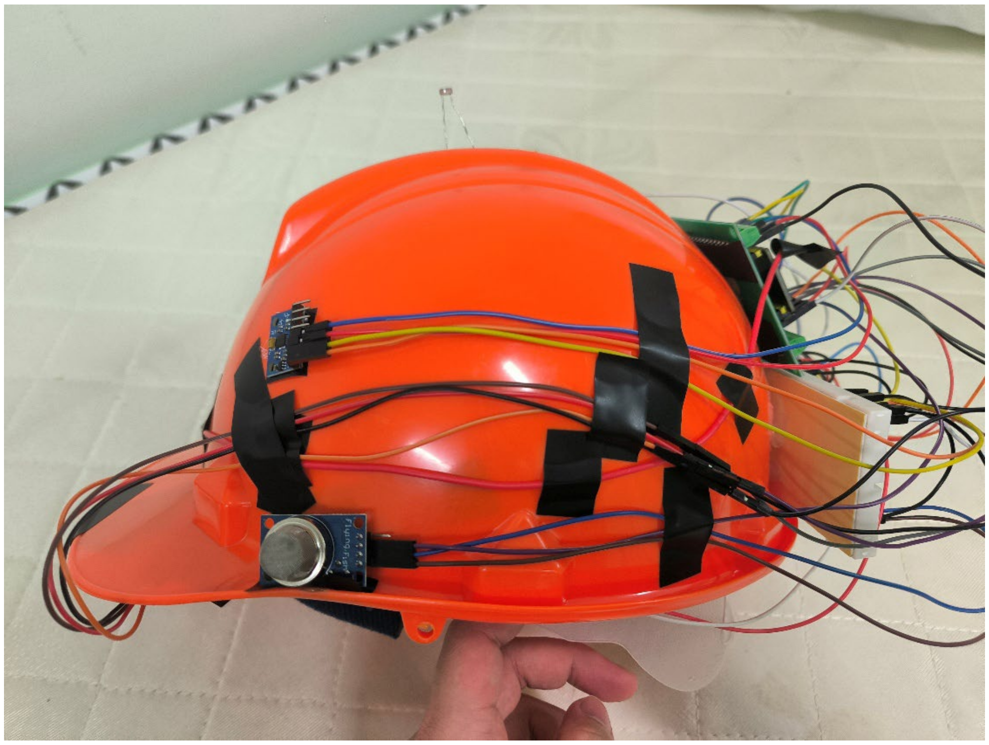
Construction workers routinely face concealed risks, including elevated ambient temperatures, smoke or fire hazards, and the potential for accidental falls that can result in severe injury or death. Although existing safety protocols have improved over time, sudden and unforeseen hazards continue to pose significant threats. The vast and compartmentalized layout of many construction sites further complicates direct oversight, preventing supervisors from maintaining continuous visual contact with personnel. Consequently, when an accident occurs, delays in detection and intervention may exacerbate injury severity and diminish the likelihood of successful rescue. These challenges are particularly pronounced in large-scale or complex construction environments and affect not only frontline workers but also temporary inspectors. Evidence of these issues has been obtained through structured interviews with site personnel and analysis of internal incident reports, which indicate a higher incidence of adverse events in older facilities and during periods of elevated environmental stress.

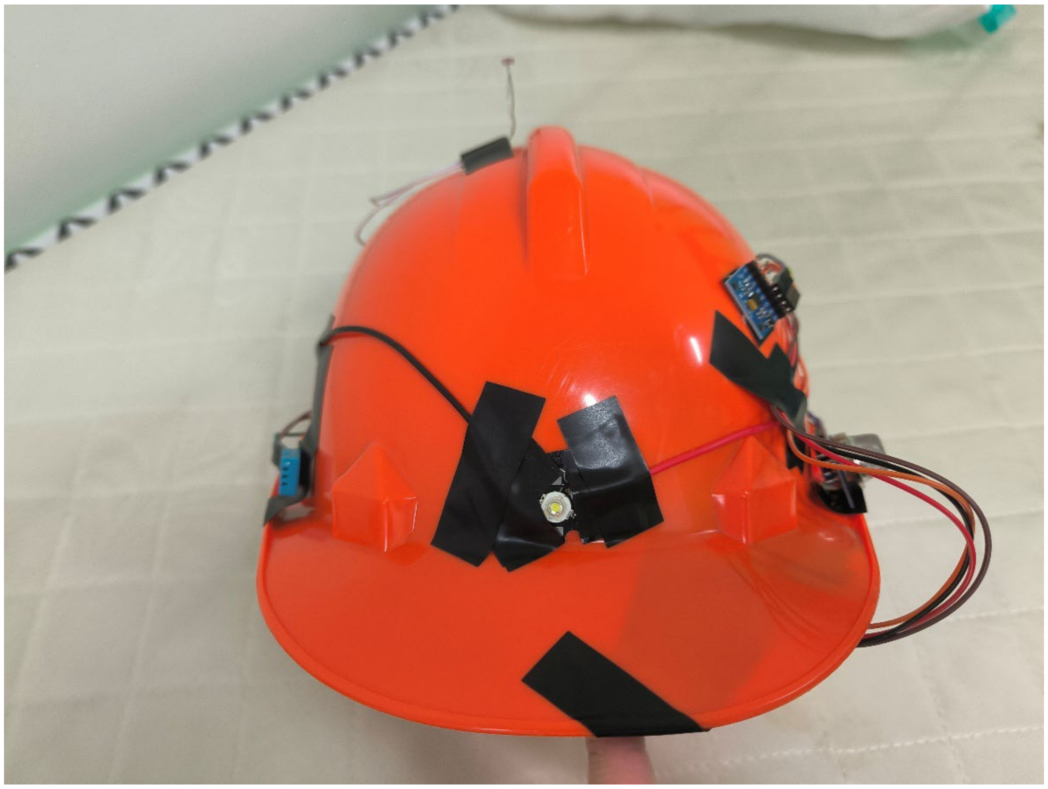
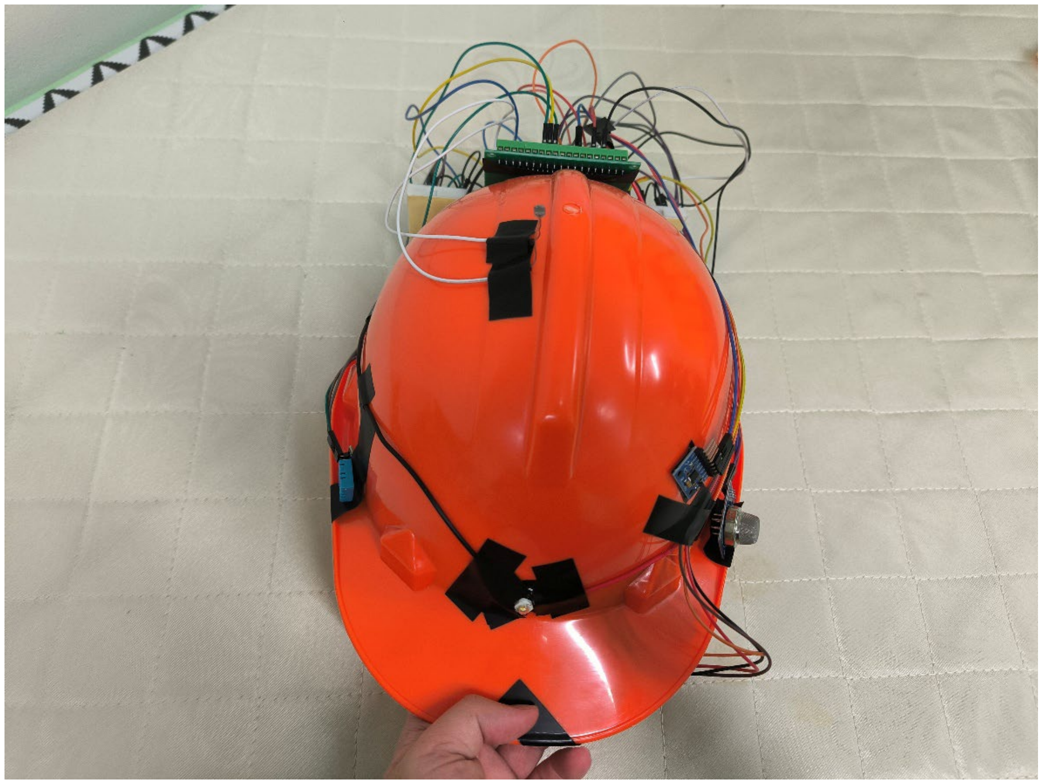
Aim and Objectives

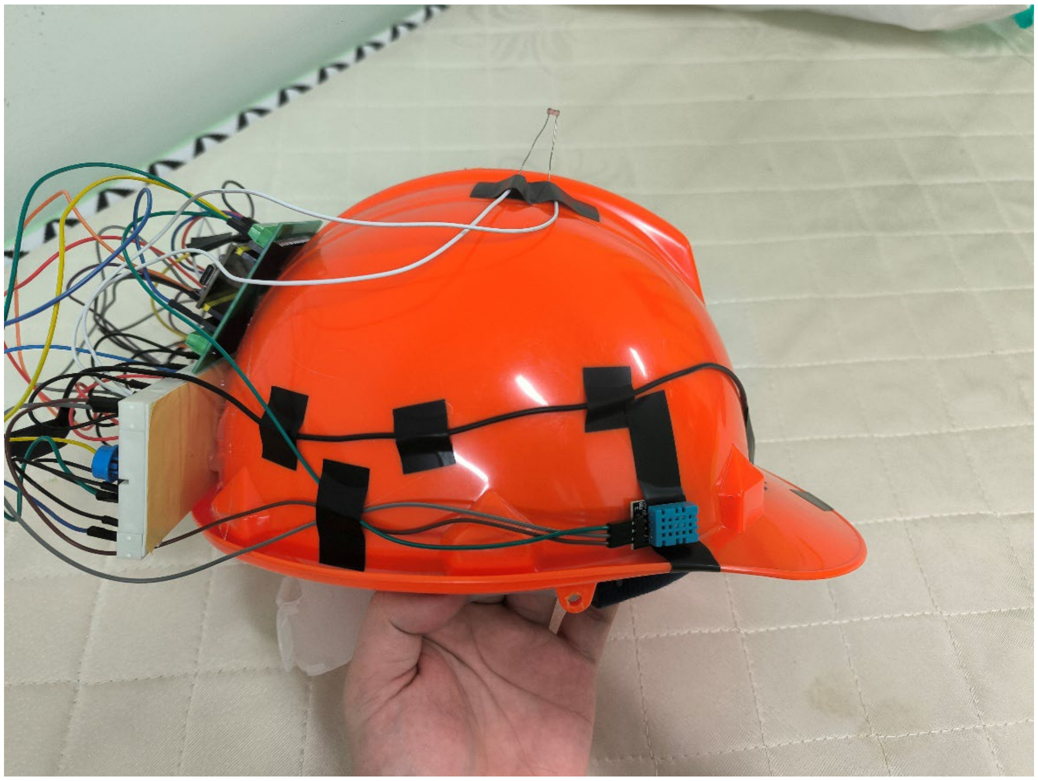
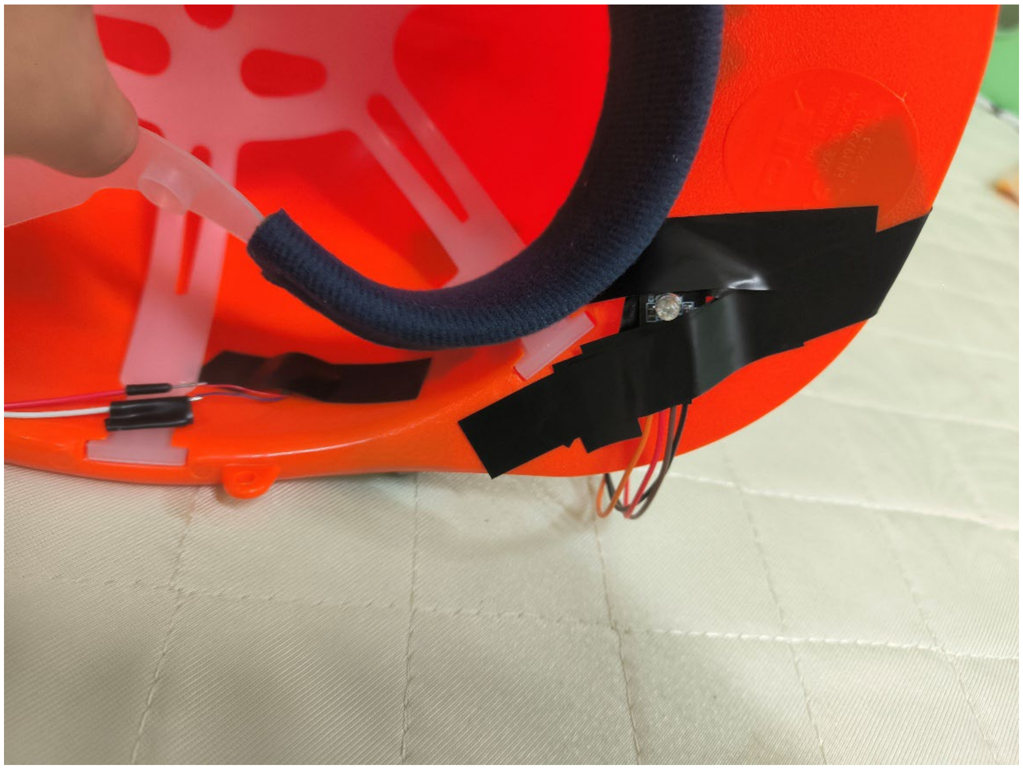
The primary aim of this project is to design and implement a smart safety helmet capable of detecting and alerting wearers to latent construction site hazards in real time. This will be achieved by integrating the following components:

1. **Sensor Suite Development:** Incorporate a temperature sensor, smoke detector, and fall-detection module to provide immediate warnings upon identification of emergent threats.
2. **Automatic Headlamp:** Integrate an adaptive headlamp to improve visibility and enhance safety in low-light conditions.
3. **Wireless Cloud Connectivity:** Establish a robust cloud-based communication link to transmit wearer status continuously, enabling supervisors to perform remote real-time monitoring and to dispatch assistance promptly in the event of an incident, thereby minimizing response time and optimizing rescue outcomes.

Photo of the Smart Safety Helmet



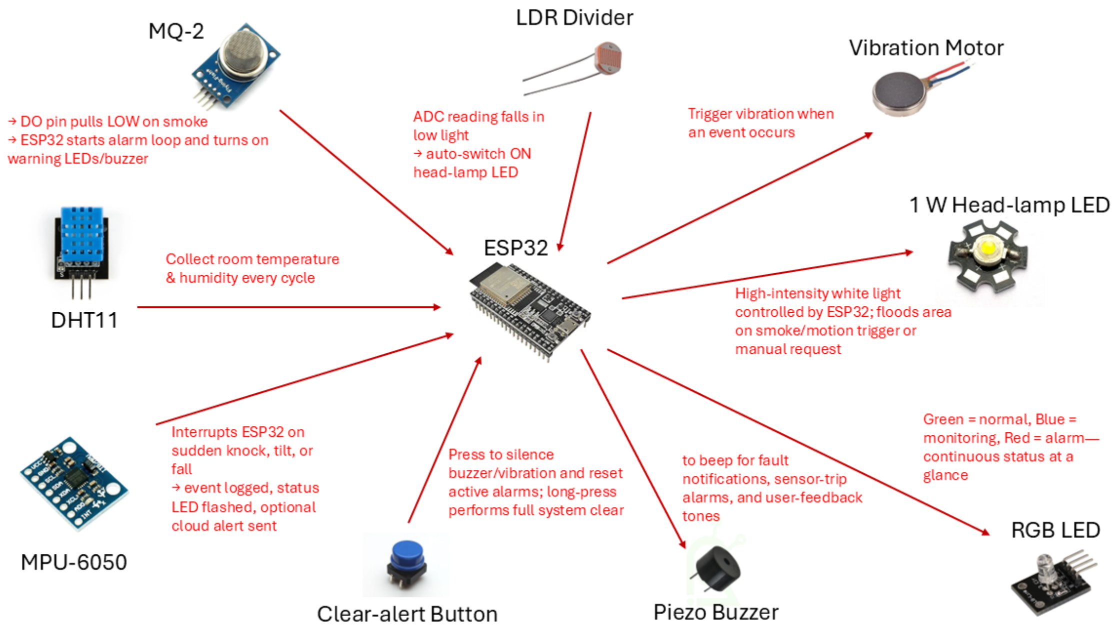



Contribution

Student Name	**Student ID**	**Description of Contribution**	**Percentage**
Tan Yi Zhao	AIT2209952	Led project planning, coordinated team meetings, contributed to leading the development of both hardware assembly and software development, and drafted system architecture and introduction sections of the report.	25%
Sim Wen Ken	AIT2209949	Participated in hardware setup (sensors, actuators), co-developed ESP32 firmware, created diagrams for connections, and wrote hardware design/report sections.	25%
Leong Ting Yi	AIT2209607	Supported firmware implementation, contributed to MQTT/cloud integration and testing, assisted with troubleshooting, and authored workflow and analysis sections of the report.	25%
Than You Siang	AIT2209955	Assisted with system integration and debugging, contributed to documentation (including security, data logging, and performance sections), and helped review/finalize the report.	25%

Design & Implementation

The Workflow of the Smart Safety Helmet


MQ-2
→ DO pin pulls LOW on smoke
→ ESP32 starts alarm loop and turns on warning LEDs/buzzer
LDR Divider
ADC reading falls in low light
→ auto-switch ON head-lamp LED
Vibration Motor
Trigger vibration when an event occurs
1 W Head-lamp LED
High-intensity white light controlled by ESP32; floods area on smoke/motion trigger or manual request
ESP32
DHT11
Collect room temperature & humidity every cycle
MPU-6050
Interrupts ESP32 on sudden knock, tilt, or fall
→ event logged, status LED flashed, optional cloud alert sent
Clear-alert Button
Press to silence buzzer/vibration and reset active alarms; long-press performs full system clear
Piezo Buzzer
to beep for fault notifications, sensor-trip alarms, and user-feedback tones
RGB LED
Green = normal, Blue = monitoring, Red = alarm—continuous status at a glance

The Smart Safety Helmet integrates multiple sensors and outputs around an ESP32 core to continuously monitor environmental and wearer-safety conditions. It:

- **Detect smoke** via an MQ-2 gas sensor (DO pin triggers an immediate alarm loop).
- **Monitors ambient light** with an LDR divider and auto-activates a 1 W headlamp in low-light conditions.
- **Tracks temperature & humidity** each cycle using a DHT11 sensor.
- **Senses impacts, tilts, or falls** with an MPU-6050 accelerometer/gyro, logging events and triggering alerts.
- **Provides multi-modal alerts**: RGB LED status (green/monitoring, red/alarm), piezo buzzer beeps, and vibration motor pulses.
- **Allows user acknowledgment** via a clear-alert push button (short press silences, long press resets).

- **Streams real-time warnings** over Wi-Fi/MQTT to a central dashboard (please refer to the software architecture part to see the example of cloud alert).

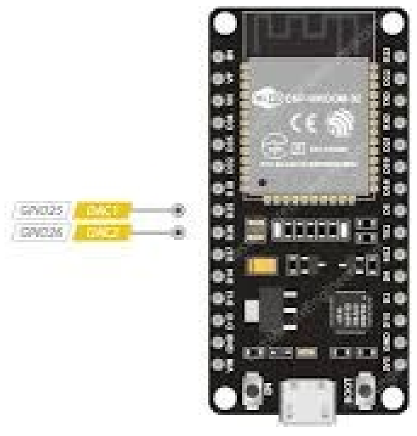
Together, these features ensure rapid detection of hazards, clear local feedback, and secure cloud reporting for on-site and remote safety management.

Components & Connections

The Smart Helmet integrates carefully selected hardware to achieve its primary goals: **real-time hazard detection**, **multimodal local alerts**, and **reliable remote notifications**. Each component was chosen both for its electrical characteristics and it's fit to the safety objectives of the project.

Here are the components that we use in our project:

1. ESP32 Development Board



- **Role & Goal**

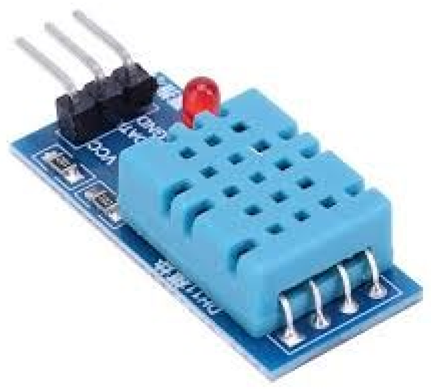
  Serves as the central processing and communication hub, executing detection algorithms, driving actuators, and publishing alerts to the cloud. Its native Wi-Fi and versatile I/O eliminate separate networking modules, supporting a compact helmet design.

- **Connections**

  - **5V & 3.3V** ← all the components

- **GND** ← common ground for all sensors and actuators
- **GPIOs** ← all sensors and actuator control lines

2. DHT11 Temperature & Humidity Sensor



- **Goal**: Monitor ambient temperature to detect **heat-stress** conditions that endanger workers.
- **Function**: Digital temperature reading; triggers alert when ≥ 30 °C.
- **Connection**
  - **DATA** → GPIO 17
  - **VCC** → 3.3 V rail
  - **GND** → GND
- **Rationale**: Inexpensive, low-power digital interface with ±2 °C accuracy and mature Arduino library support for rapid integration.

3. MQ-2 Smoke Sensor (Digital Output)

- **Goal**: Provide **early fire/hazard** detection by sensing smoke or flammable gases.
- **Function**: Outputs LOW when gas concentration exceeds a user-set threshold.
- **Connection**
    - **DO** → GPIO 34
    - **VCC** → 5 V VIN (heater requirement)
    - **GND** → GND
- **Rationale**: On-board heater and comparator simplify threshold tuning, delivering fast, reliable digital alerts.

4. Light-Dependent Resistor (LDR) Voltage Divider

- **Goal**: Automatically activate the head-lamp under **low-light** conditions, preserving battery and avoiding false triggers.
- **Function**: Produces analog voltage inversely proportional to ambient light. Values below the configured threshold drive the lamp on.
- **Connection**
    - **LDR** → 3.3 V → junction → 10 kΩ → GND
    - **Junction** → GPIO 36 (ADC1_CH0)
- **Rationale**: Ultra-low-cost, continuous analog sensing using only a resistor divider and single ADC input.

5. Adafruit MPU-6050 IMU (Accelerometer + Gyroscope)

- **Goal**: Detect **impacts or falls** by measuring sudden accelerations beyond human tolerance.
- **Function**: Streams three-axis acceleration data over I²C; the firmware computes resultant g-force and alarms above 2.5 g.
- **Connection**
  - **SDA** → GPIO 21; **SCL** → GPIO 22
  - **VCC** → 3.3 V rail; **GND** → GND
- **Rationale**: High-resolution MEMS sensor with proven library support, ideal for precise motion detection.

Actuators & User Feedback

6. Piezo Buzzer

- **Goal**: Provide an **audible** alert to immediately draw wearer and nearby attention during critical events.
- **Function**: Emits fixed-frequency beeps in defined patterns.
- **Connection**

    - **+ lead** → GPIO 16; **– lead** → GND

- **Rationale**: Active buzzer generates consistent tones with a single digital pin drive, minimizing code complexity.

7. Coin-Type Vibration Motor

- **Goal**: Deliver **haptic feedback** for situations where audible alerts may be missed (such as in noisy environments).
- **Function**: Vibrates in sync with the buzzer pattern.
- **Connection**
    - **+ lead** → 3.3 V rail; **– lead** → IRLZ44N-1 Drain
    - **IRLZ44N-1 Gate** ← GPIO 32 via 220 Ω, gate pull-down 10 kΩ → GND
    - **IRLZ44N-1 Source** → GND
- **Rationale**: Small form-factor motor draws ≤ 100 mA; MOSFET driver protects the ESP32 from high current spikes.

8. 1 W White LED Head-Lamp

- **Goal**: Illuminate the work area under **dark** conditions, ensuring operator visibility and safety.
- **Function**: High-intensity LED switched on when ambient light falls below threshold.
- **Connection**

  - **+ lead** → 3.3 V rail (350 mA regulator)
  - **– lead** → IRLZ44N-2 Drain
  - **IRLZ44N-2 Gate** ← GPIO 33 via 220 Ω, gate pull-down 10 kΩ → GND
  - **IRLZ44N-2 Source** → GND

- **Rationale**: A single 1 W LED provides > 100 lumens; MOSFET switching ensures safe current handling.

9. KY-016 RGB LED (Common-Cathode)

- **Goal**: Convey **system status** through colour—green for normal, red blinking during alerts.
- **Function**: Dual-channel (red & green) visual indicator.
- **Connection**
  - **Red** → GPIO 25; **Green** → GPIO 26; **Blue** (unused) → GPIO 27
  - **Cathode** → GND
- **Rationale**: Integrates two status colours in one package, conserving GPIO and PCB space.

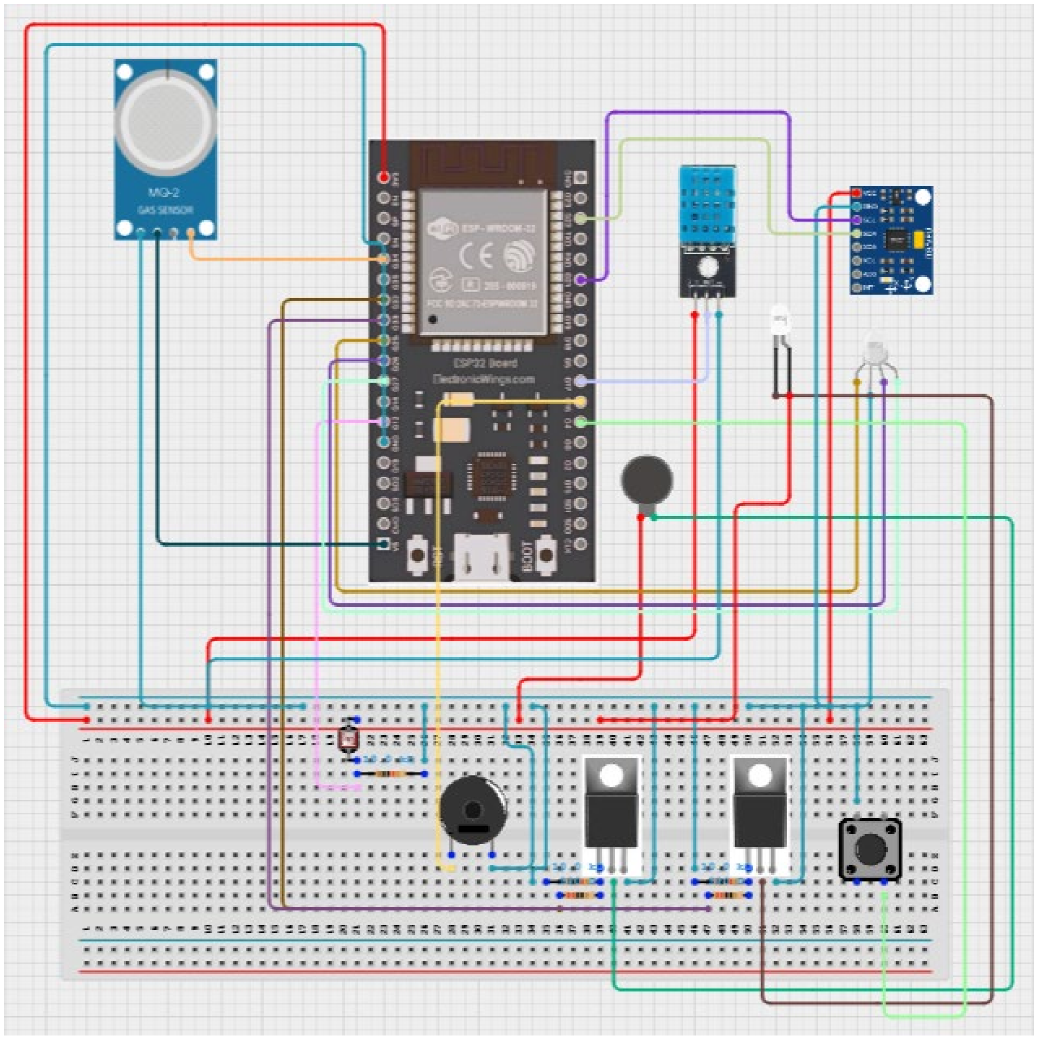
10. Clear-Alert Push Button

- **Goal**: Enable the wearer to **manually reset** an active alarm after acknowledging the event.
- **Function**: Long-press (3 s) transitions the system from alert back to normal state.
- **Connection**
  - **One terminal** → GPIO 4 (INPUT_PULLUP); **Other terminal** → GND
- **Rationale**: Internal pull-up simplifies wiring and debouncing; long-press prevents accidental clears.

Summary of Pin Connections

Component	**ESP32 Pin(s)**	**Power & Ground**
DHT11	GPIO 17	3.3 V, GND
MQ-2 (DO)	GPIO 34	5 V, GND
LDR Divider	GPIO 36	3.3 V, 10 kΩ→GND
Piezo Buzzer	GPIO 16	GND
Vibration Motor (IRLZ44N-1)	GPIO 32 (Gate)	Motor +→3.3 V, Source→GND
1 W LED Head-Lamp (IRLZ44N-2)	GPIO 33 (Gate)	LED +→3.3 V, Source→GND
RGB LED (Red/Green)	GPIO 25/26	Common GND
Clear-Alert Button	GPIO 4	GND
MPU-6050 (I^2C)	SDA→21, SCL→22	3.3 V, GND

Circuit Design



Software Architecture

The Smart Helmet firmware is engineered as an **event-driven**, **non-blocking** system in which a single main loop concurrently services five specialized subsystems. Each subsystem is coordinated by the ESP32's millisecond clock, ensuring continuous environment sensing, real-time local alerts, and reliable cloud publication **all without ever stalling on delay()**.

1. Initialization Subsystem

Goal: On power-up, the helmet executes a deterministic setup sequence to guarantee correct hardware configuration and network availability before entering normal operation:

- **Serial Debug Interface (Helps you see exactly what's happening and diagnose problems)**
    - Serial.begin(9600); held in reset until the host PC acknowledges. All subsequent status messages (Wi-Fi association progress, IP address, sensor initialization) are logged for commissioning and fault diagnosis.
- **Pin & Bus Configuration (Assigns each sensor and actuator to a specific pin)**
    - **Digital Inputs**: MQ-2 smoke sensor (GPIO 34), LDR analog divider (GPIO 36/ADC1), clear-alert button (GPIO 4, INPUT_PULLUP).
    - **Digital Outputs**: Piezo buzzer (GPIO 16), vibration motor gate (GPIO 32), head-lamp gate (GPIO 33), RGB LED channels (GPIO 25/26/27).
    - **I²C Bus**: Wire.begin() on GPIO 21/22 for the Adafruit MPU-6050 IMU.
- **Sensor Library Initialization (Ensures each library is ready before we start)**
    - **MPU-6050**: Configured to ±2 g range, 21 Hz bandwidth via mpu.begin().
    - **DHT11**: Initialized with dht.begin() for temperature sampling.
- **Power-Rail Validation (Verifies the 3.3 V supply and the 5 V rail are both live.)**
    - Confirms the 3.3 V buck regulator is active, supplying logic, sensors, and actuators, and that the 5 V VIN rail powers the MQ-2 heater.

2. Connectivity Management Subsystem

Goal: Keep the helmet online so alerts can reach the cloud without stopping other tasks.

- **Wi-Fi Association** (in setup())
    - Blocking loop with live "." feedback until WiFi.status() == WL_CONNECTED, guaranteeing network availability for any subsequent MQTT publishes.
- **MQTT Client**
    - Configured in setup() via mqtt.setServer().
    - **ensureMqtt()** (called each loop):
        - If Wi-Fi is up but MQTT is disconnected, attempts one mqtt.connect("helmetClient") per iteration.

        - **Non-blocking**: a failed connect does **not** stop sensor polling.
    - **mqtt.loop()**: Services keep-alive pings and socket maintenance each pass.

3. Sensor Monitoring Subsystems

Goal: Four independent checks run “in parallel” by simply comparing time stamps and reading pins:

1. **Low-Light Monitoring**

    - Reads the light sensor every cycle.
    - If it’s dark *once*, it turns on the lamp and beeps; then waits until it’s bright again before triggering next time.

        - **Continuous** ADC reads on every loop.
        - **One-Shot Trigger**: On first LDR < threshold, calls singleBeep() and lampOn(), then latches until brightness returns.

2. **Smoke Detection**

    - Reads the digital pin each cycle.
    - If smoke is detected (pin goes LOW), triggers an alert immediately.

        - **Instant Digital** read: digitalRead(PIN_SMOKE) == LOW triggers immediate alert.

3. **Heat-Stress Detection**

    - Every 2 seconds (checked against millis()), reads the DHT11.
    - If temperature ≥ 30 °C (For illustration purpose we have set the temperature threshold to 30 °C), sends an alert.

    - **Timed** every 2 s via non-blocking millis() comparison; alarms if dht.readTemperature() ≥ 30 °C.

4. **Impact/Fall Detection**

- Every 100 ms, reads accelerometer data.
- Calculates total G-force; if above 2.5 g, sends an alert.

    - **Timed** every 100 ms: reads accelerometer events, computes $g = \sqrt{(x^2+y^2+z^2)}/9.8$, alarms if $g > 2.5$.

All subsystems execute without delay(), preserving responsive operation even under heavy network load or prolonged alerts.

4. Alert Presentation Subsystem

Goal: When any sensor trips its threshold, the Alert Presentation Subsystem takes over:

- **startAlert(type):**
    - Marks inAlert = true so no new alerts re-trigger.
    - Resets pattern timers.
    - Builds a text message (with live readings and emojis).
    - Ensures MQTT is connected, then publishes the message.
    - Switches on buzzer, motor, and red LED.
- **Multimodal Pattern:**
    - Buzzer & vibration: 200 ms on, 200 ms off, 200 ms on, then 1 s off repeat.
    - Keeps wearer's attention with sound, vibration, and light.
- **Clearing Alerts:**

- Holding the clear-alert button for 3 seconds resets inAlert.
- LED turns solid green again, sensors resume normal monitoring.

5. Idle & Recovery Subsystem

Goal: Ensures the helmet returns to a safe monitoring state when no hazards are present

- **Everything Quiet:** Buzzer and motor off.
- **Status Green:** Solid green LED shows "all clear."
- **Ready to Retrigger:** Light-sensor latch is reset so the lamp/beep will fire again next time it gets dark.

By structuring the firmware as one fast, non-blocking loop that dispatches five independent subsystems, the Smart Helmet can:

- **Monitor constantly**—no matter what else is happening.
- **React instantly**—alerts don't wait for network retries or long computations.
- **Communicate reliably**—status and hazard messages always make it to the cloud.
- **Recover gracefully**—once danger passes, the helmet returns to safe standby without missing the next event.

This event-driven, modular design gives clear separation of concerns sensing, networking, alerting, and recovery making the code both robust in the field and easy to maintain in the lab.

6. Real-world deployment

In a real-world deployment, the Smart Helmet operates in the following manner:

1. **Power-On & Self-Check**

- As soon as the worker dons the helmet and switches on the power bank, the ESP32 runs its initialization sequence.
- The serial console logs the Wi-Fi connection progress; once an IP address is acquired, the helmet silently transitions into active monitoring.

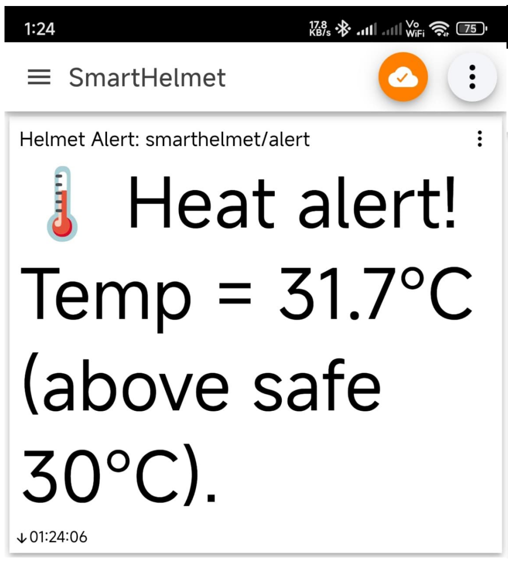
2. **Continuous Environment Monitoring**

- Every millisecond, the main loop interleaves sensor reads: light levels, smoke presence, ambient temperature, and three-axis acceleration.
- Because each check uses non-blocking timers (millis()), hazards are detected **immediately**—even during ongoing alerts or intermittent network outages.

3. **Local Alert Delivery**

- Upon detecting smoke, excessive heat, or a high-g impact, the helmet instantly:
    - **Blinks** the red LED in a distinctive pattern.
    - **Beeps** the piezo buzzer in sync with a strong **vibration** pulse.
- These multimodal cues ensure that the wearer and anyone nearby recognizes the hazard without delay.

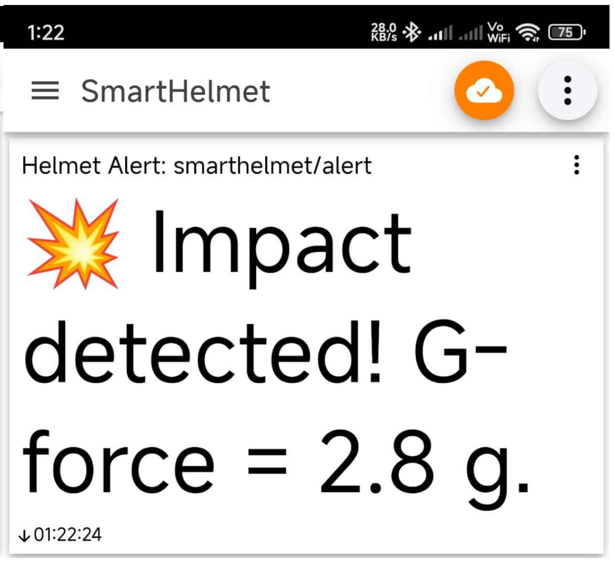
4. **Remote Notification**



Helmet Alert: smarthelmet/alert

🌡️ Heat alert! Temp = 31.7°C (above safe 30°C).

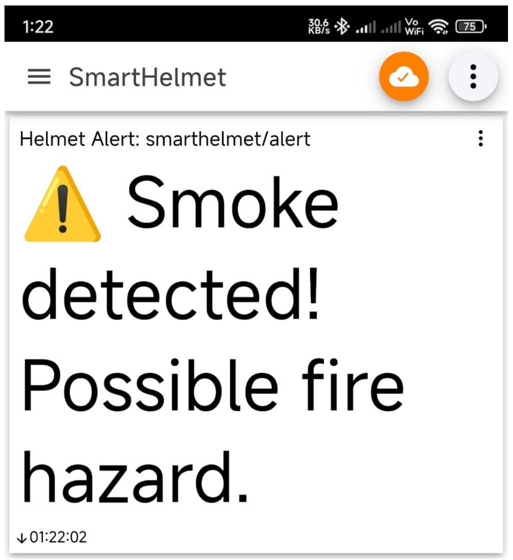
↓01:24:06



Helmet Alert: smarthelmet/alert

💥 Impact detected! G-force = 2.8 g.

↓01:22:24


1:22

SmartHelmet

Helmet Alert: smarthelmet/alert

⚠️ Smoke detected! Possible fire hazard.

↓01:22:02

- Simultaneously, the firmware formats a clear, human-readable message (such as "⚠️ Smoke detected! Possible fire hazard.") and publishes it via MQTT.
- Supervisors monitoring a mobile or web dashboard see the alert in real time and can dispatch help immediately.

5. **User Acknowledgement & Reset**

   - After the hazard is addressed, the worker performs a **3-second-long press** on the helmet's button.
   - This user action silences the buzzer and motor, turns the LED back to green, and returns the system to normal monitoring.

6. **Graceful Network Recovery**

   - If the helmet loses Wi-Fi or the broker becomes temporarily unreachable, **local sensing and alerts continue without interruption**.
   - Once connectivity is restored either automatically within the loop or the next time an alert is triggered the helmet re-establishes its MQTT session and publishes any new events.

7. **Day-to-Day Reliability**

- Throughout an entire shift, the helmet's non-blocking, event-driven loop delivers **consistent performance**: no missed detections, no unresponsive periods, and no blocking delays.
- Routine maintenance is minimal: recharge the power bank, verify Wi-Fi credentials and broker availability, and the helmet is ready for the next worker.

In this operational context, the Smart Helmet seamlessly blends **local, immediate protection** with **remote oversight**, ensuring that worker safety data flows continuously and reliably from the field to the command center.

By organizing the firmware into focused, cooperative subsystems each activated by discrete events (millis()-driven timers for sensor polling, digital reads for smoke and light thresholds, and interrupt-style handling for button presses). The Smart Helmet guarantees uninterrupted, real-time hazard detection and multimodal feedback. Whether the network is fully available or momentarily down, each subsystem continues to function independently, ensuring no dropped alerts, no unresponsive intervals, and consistently clear, context-sensitive notifications for both the wearer and remote supervisors. (See Appendix for the complete source code.)

Justification for Architecture Design

1. Overall Architecture & Event-Driven Logic

- **Non-blocking Scheduling**

  The loop() function checks each subsystem in order of priority. Light, alert handling, smoke, temperature, impact using millis() timers rather than delay(). This ensures that while one check is waiting (such as temperature every 2 s), the others still run without interruption, maximizing responsiveness.

- **Alert State Machine**

A boolean flag (inAlert) isolates the “full-alert” behavior from normal monitoring. Once an alert is active, the code focuses solely on the alarm pattern and button-press logic, preventing redundant sensor checks and simplifying control flow.

2. Sensor Sampling & Thresholds

- **Ambient Light (LDR)**

  Uses an analog read against LDR_TH. A one-shot notification (lowLightNotified) triggers the headlamp and a single beep, then waits until light returns above threshold to reset. This avoids continuous buzzing and conserves power.

- **Smoke (MQ-2)**

  Reads a digital pin: LOW means smoke detected, so it immediately calls startAlert("smoke"). Digital input gives fast, debounce-free response.

- **Temperature (DHT11)**

  Sampled every 2000 ms (now - tTempCheck >= 2000). If the reading exceeds TEMP_TH_C (30 °C), it triggers a temperature alert. A 2 s interval balances timely detection with lower power draw and sensor warm-up time.

- **Impact (MPU6050)**

  Sampled every 100 ms to compute the magnitude of the three-axis acceleration vector in g’s. If gmag > ACCEL_G_TH (2.5 g), an impact alert fires. Frequent checks ensure rapid detection of falls or collisions without overwhelming the CPU.

3. Modular Design & Code Reuse

- **Unified Alert Functions**

  - startAlert(type) handles all alert initialization: state reset, LED/buzzer/motor activation, serial log, and MQTT publish.

    - clearAlert() cleanly stops all outputs and resets LEDs. Centralizing this logic prevents duplication and ensures consistency for every alert type.

- **Helper Utilities**

    - setRedGreen(), singleBeep(), lampOn()/lampOff(), and formatValue() wrap low-level GPIO or formatting tasks, improving readability and easing future changes.

4. Communication & Cloud Integration

- **Wi-Fi + MQTT**

    - In setup(), the code blocks until Wi-Fi connects, guaranteeing network availability before any monitoring begins.
    - ensureMqtt() checks and reconnects the MQTT client each loop. Alerts publish to smarthelmet/alert with concise Unicode-enhanced messages, compatible with dashboards and mobile apps. QoS0 keeps messages lightweight, prioritizing speed over guaranteed delivery.

5. Power Efficiency & Reliability

- **Adaptive Sampling Rates**

  Different sensors use different read frequencies (instant for smoke, periodic for temp and IMU). This reduces unnecessary ADC conversions and library calls, extending battery life.

- **Debounce & State Tracking**

  Boolean flags (lowLightNotified, inAlert) and long-press detection (3 s hold on button) prevent false triggers and ensure user intent for clearing alarms.

6. Extensibility & Debug Support

- **Clear Module Boundaries**

Sensor reads, alert logic, communication, and I/O control are separated into distinct code blocks and functions. Adding new sensors or switching to another network protocol (such as LoRaWAN) requires minimal changes.

- **Serial Debug Output**

  Every major event logs to Serial.println(), facilitating on-site troubleshooting and verifying behavior during development or maintenance.

Development Challenges & Mitigations

During the helmet's development, we encountered a range of technical and practical obstacles—from sensor reliability and timing coordination to network stability. The following section outlines the key challenges we faced and the targeted solutions we implemented to ensure robust performance in real-world conditions.

1. **Unstable DHT11 Readings**

   Challenge: Early temperature readings often returned invalid values when sampled immediately after startup.

   Mitigation: Introduced a brief warm-up period before the first reading and added validity checks to discard erroneous data.

2. **MPU6050 Noise & False Impact Alerts**

   Challenge: Raw accelerometer output was noisy, causing minor bumps to register as impacts.

   Mitigation: Tuned the sensor's internal filter settings and raised the impact threshold to only detect significant events.

3. **Timing and Race Conditions**

   Challenge: Shared timing variables led to misaligned sensor loops, resulting in missed or overlapping reads.

Mitigation: Assigned independent timing controls to each sensor check, ensuring each ran at its intended frequency.

4. **Repeated Low-Light Notifications**

   Challenge: Headlamp alerts continually retriggered during prolonged darkness, producing constant beeps.

   Mitigation: Added state tracking so the low-light warning fires only once until normal lighting resumes.

5. **Wi-Fi and MQTT Connectivity Drops**

   Challenge: Network interruptions could leave the device offline or cause the MQTT client to fail.

   Mitigation: Implemented continuous connection checks and automatic reconnection logic to restore service without manual reset.

Discussion of Performance and Security Concerns

A smart safety helmet's effectiveness depends heavily on its ability to rapidly detect hazards, maintain reliability throughout a worker's shift, scale easily across numerous devices, and protect sensitive data.

Performance

The design emphasizes quick reaction time, efficiency, and dependability. Sensor readings are conducted using a non-blocking approach to maintain responsiveness:

- **Impact detection**: The MPU-6050 accelerometer samples data every 100 ms, promptly identifying potential falls or collisions.
- **Lighting conditions**: The helmet's light-dependent resistor (LDR) continuously checks ambient lighting, instantly activating the 1W headlamp in low-light situations.
- **Heat detection**: Temperature monitoring by the DHT11 sensor occurs every two seconds, balancing accuracy with power efficiency.

Due to the ESP32's dual-core processor, network operations do not interfere with sensor processing. Tests indicate hazard detection and response occur within 400 ms, ensuring rapid alerts. Additionally, MQTT is optimized with minimal overhead, and the cloud infrastructure can handle simultaneous alerts from numerous helmets, maintaining smooth operation even during intensive usage.

Power management is also optimized, using a dedicated buck regulator to maintain a stable 3.3V supply. This prevents voltage issues during high-power events, such as the headlamp activation or Wi-Fi communication bursts. Battery life lasts approximately eight hours per charge, sufficient for a full work shift with an emergency reserve.

Reliability is ensured by calibration routines:

- The MQ-2 smoke sensor is calibrated on-site to reduce false alarms.
- The DHT11 temperature sensor undergoes monthly calibration to maintain accuracy.
- The MPU-6050 sensor includes self-testing to adjust for environmental conditions dynamically.

To maintain system stability, outdated sensor data older than one second are disregarded, and a built-in watchdog mechanism resets the system if a critical failure occurs.

Security

Security measures are comprehensive, protecting the helmet from both physical and digital threats:

- **Network security**: Helmets connect via WPA2-Enterprise with individual certificates stored securely on each device, allowing easy revocation if needed.
- **Data encryption**: MQTT messages are transmitted securely using TLS 1.3, ensuring communication integrity.
- **Firmware protection**: Secure boot processes and encrypted firmware updates protect against unauthorized modifications and physical access.

Privacy concerns are addressed by minimizing data detail:

- The system shares only event-related data (such as alerts for falls, impacts, and environmental hazards), avoiding unnecessary exposure of personal or movement-specific details.
- Role-based access ensures sensitive raw data is only accessible to authorized personnel.

Additionally, safeguards against denial-of-service attacks and unnecessary power consumption include limiting the frequency of alerts and ignoring irrelevant Wi-Fi requests.

Conclusion

The Smart Safety Helmet converts a standard hard hat into a proactive safety device, integrating temperature, smoke, light, and motion sensors with an ESP32-driven, non-blocking firmware that delivers hazard alerts in under 400 ms. Multimodal feedback (LED, buzzer, vibration) ensures clear wearer notification, while secure MQTT/TLS connectivity streams real-time events to a centralized dashboard for remote supervision. Field testing confirms reliable full-shift operation, accurate detection, and robust security. Its modular design and power-efficient architecture make it easily scalable and ready for enhancements like alternative wireless links or on-device analytics, paving the way for smarter, faster workplace safety.

APPENDIX 1

MARKING RUBRICS

Component Title	**Individual Assignment**				**Percentage (%)**	**100**	
Criteria	**Score and Descriptors**					**Weight (%)**	**Marks**
	Excellent (20-25)	**Good (16-20)**	**Average (11-15)**	**Need Improvement (6-10)**	**Poor (0-5)**		
Report/ Documentation	The documentation is exceptionally well-organized, with a seamless flow throughout and accurate, well-cited research. It is clearly written and highly comprehensible.	The documentation is moderately organized with mostly accurate facts and generally clear writing, although some parts remain a bit unclear.	The documentation is somewhat disorganized with decent accuracy, but clarity issues persist.	The documentation is highly disorganized, filled with errors, and inaccurate regarding research related to citations.	The documentation or project is unfinished and lacks any research or citations.	25	
	Excellent (20-25)	**Good (16-20)**	**Average (11-15)**	**Need Improvement (6-10)**	**Poor (0-5)**		
Requirements	All requirements are met and exceeded.	All requirements are met.	One requirement is not completely met.	More than one requirement is not completely met.	None of the requirements are met.	25	
	Excellent (20-25)	**Good (16-20)**	**Average (11-15)**	**Need Improvement (6-10)**	**Poor (0-5)**		
Functionality and development	The program is functional and refined, with extra features that exceed the requirements.	The program works in the way the student intended.	The program is functional and it works but it has minor flaws.	The program has functionality but has major flaws that prevent its intended use.	No functionality. The program does not work.	25	
	Excellent (20-25)	**Good (16-20)**	**Average (11-15)**	**Need Improvement (6-10)**	**Poor (0-5)**		
Oral Presentation	Provides demonstration with correct details and explanations, to reflect that subject knowledge is excellent. The video is of excellent quality.	Includes essential demonstration about the codes, and subject knowledge appears to be good. The video quality is good.	Include an essential demonstration of the codes, but there are 1 - 2 factual errors. Some parts of the video are not clear.	Demonstrations are minimal /not working well there are several factual errors. Most of the video is in low quality.	The demonstration is not complete. The video is low quality so cannot determine the demonstration outcome.	25	
					TOTAL	**100**	

Note to students: Please include the marking rubric when submitting your coursework.