**AIM:**

The aim of this project is to develop an innovative IoT-enabled military smart jacket that combines Artificial Intelligence (AI) and Internet of Things (IoT) technologies for real-time health monitoring, anomaly detection, and location tracking. This smart jacket is designed to enhance soldier safety, operational efficiency, and situational awareness in high-risk environments.

The jacket continuously monitors vital health parameters such as heart rate, blood pressure, ECG, $SpO_2$, and body temperature, using advanced sensors. AI algorithms analyze this data to detect early signs of health anomalies like heart attacks or blood clots, allowing timely intervention.

GPS technology provides real-time location tracking, aiding in better mission coordination and rescue operations. Additionally, environmental sensors track ambient conditions, including temperature, humidity, and toxic gases, ensuring soldiers are aware of hazardous surroundings.

Emergency features such as automatic SOS alerts and fall detection ensure critical incidents are reported instantly. With end-to-end encryption, the system secures sensitive data transmission.

This project offers a cost-effective, scalable solution for military and rescue operations, with potential extensions to civilian healthcare, providing predictive insights for improved safety and mission success.

**REQUIREMENTS:**

**HARDWARE:**

- **ESP32 Microcontroller:** Acts as the central microcontroller for IoT operations, processing inputs and communicating data to the cloud. Its versatility and low cost make it ideal for prototyping.
- **Health Sensors (MAX30102, AD8232):** Monitor vital signs such as heart rate, ECG, and $SpO_2$.
- **Temperature and Humidity Sensors:** Track environmental conditions to ensure soldier well-being.
- **GPS Module:** Provides real-time location tracking for mission coordination.
- Battery Pack: Powers the jacket with optimized power management for extended use.
- **Flexible Circuit Board:** Facilitates secure connections between sensors and the microcontroller.
- **LED Indicators:** Provide real-time feedback on system status.
- Fabric with Embedded Wiring: Ensures seamless integration of electronic components.

**SOFTWARE:**

- **Arduino IDE:** A user-friendly programming environment based on C++ for coding the ESP32 to collect sensor data and communicate with the cloud. Its open-source nature supports extensive community resources.
- **Python:** Powers the data analysis pipeline with its robust libraries, enabling seamless integration of machine learning tools for real-time data processing.
- **AWS IoT Core:** A cloud platform used for storing and analyzing sensor data, offering real-time insights and remote monitoring capabilities.
- **AI Models:** Lightweight AI models for predictive analytics and anomaly detection.

- **Web Dashboard:** Provides a user-friendly interface for real-time monitoring and alert management.

**THEORY:**

**IOT-BASED SOLDIER HEALTH AND LOCATION MONITORING**

The Internet of Things (IoT) forms the foundation of the IoT-enabled military smart jacket by enabling real-time health monitoring, anomaly detection, and location tracking. The jacket is embedded with multiple health sensors that continuously track vital parameters such as heart rate, ECG, $SpO_2$, and body temperature. When abnormal readings are detected, the system triggers an alert that is transmitted via LoRa or NB-IoT communication to a central command center. Additionally, GPS technology is integrated to provide real-time soldier location tracking, enhancing mission coordination and rescue efforts in critical situations. The collected data is processed on the edge using AI algorithms and further analyzed in the cloud for predictive insights.

**ESP32 MICROCONTROLLER**

The ESP32 microcontroller serves as the IoT hub of the system, interfacing with all health sensors, GPS, and environmental monitoring units. It processes sensor data in real-time and transmits it securely to cloud platforms such as AWS IoT Core. The ESP32 operates at low power, making it ideal for battery-powered wearable systems. Additionally, it supports Wi-Fi and Bluetooth for local communication and LoRa or NB-IoT for long-range data transmission, ensuring uninterrupted connectivity in remote military environments.

**AI-POWERED HEALTH ANALYTICS AND ENVIRONMENTAL MONITORING**

AI-driven analytics enhance the effectiveness of the smart jacket by identifying health anomalies and predicting potential risks. Machine learning models process historical and real- time data to detect patterns indicating fatigue, dehydration, or impending medical emergencies like heart failure or hypothermia. The environmental sensors monitor ambient temperature, humidity, and exposure to toxic gases, allowing soldiers to adapt to hazardous environments. AI models deployed on edge devices ensure real-time processing, while cloud- based analytics provide long-term health assessments.

**EMERGENCY ALERT SYSTEM AND DATA SECURITY**

The jacket includes automatic SOS alerts and fall detection to ensure immediate response in emergencies. If a soldier is injured or immobilized, the system automatically sends an emergency signal with location data to command centers and nearby medics. End-to-end encryption is implemented for secure communication, protecting sensitive military data from cyber threats.

By integrating IoT, AI, and real-time analytics, this smart jacket significantly enhances soldier safety, operational efficiency, and battlefield awareness, making it an invaluable tool for modern military operations.

**IMPLEMENTATION STEPS:**

1. **IOT SETUP**

   The first step involves assembling the IoT hardware to monitor soldiers' vital signs and environmental conditions. Health sensors (such as MAX30102 for $SpO_2$ and AD8232 for ECG) are embedded in the smart jacket to continuously measure vital parameters. Environmental sensors detect temperature, humidity, and toxic gases,
   while the GPS module tracks the soldier's location. These components are connected to the ESP32 microcontroller using a flexible PCB (Printed Circuit Board) or jumper wires. The ESP32 collects sensor data, processes it in real-time, and transmits it securely to the cloud using LoRa or NB-IoT communication modules. An Arduino sketch is written to handle data acquisition, processing, and transmission. This setup ensures continuous, reliable data collection for further analysis.

2. **DATA CAPTURE AND CLOUD INTEGRATION**

   Once the hardware is set up, sensor data is captured and transmitted to AWS IoT Core or similar cloud platforms. The ESP32 processes data locally for immediate alerts (such as fall detection or health anomalies) and sends batch data to the cloud for long- term analysis. The cloud platform aggregates and stores data, allowing for real-time and historical monitoring through a web dashboard. Secure protocols such as MQTT and TLS encryption ensure data security.

3. **AI-BASED HEALTH AND ENVIRONMENTAL ANALYSIS**

   AI models deployed on edge devices and in the cloud analyze sensor data to detect patterns indicating health risks or environmental threats. For example, a sudden increase in body temperature combined with a drop in oxygen levels could indicate heat exhaustion or hypoxia. Machine learning models analyze historical trends to predict potential health risks, such as heart conditions or fatigue. Anomalies trigger instant alerts to command centers, enabling rapid decision-making.

4. **EMERGENCY ALERT AND RESPONSE SYSTEM**

   The jacket includes an automatic SOS system that triggers in case of falls or medical emergencies. Upon detecting a fall or abnormal vitals, the ESP32 sends an alert with the soldier's real-time location to nearby medics or command units. Vibration motors or LED indicators on the jacket can alert the soldier or teammates nearby. This system ensures quick response times, potentially saving lives in critical situations.

5. **TESTING AND CALIBRATION**

   The final step involves rigorous testing under simulated battlefield conditions. Sensor calibration ensures accuracy, while field tests validate the system's durability and reliability. Calibration involves adjusting the sensitivity of health sensors and testing the responsiveness of the GPS and environmental monitoring systems. Continuous feedback helps optimize battery life, sensor performance, and data accuracy.

**DIAGRAMS:**

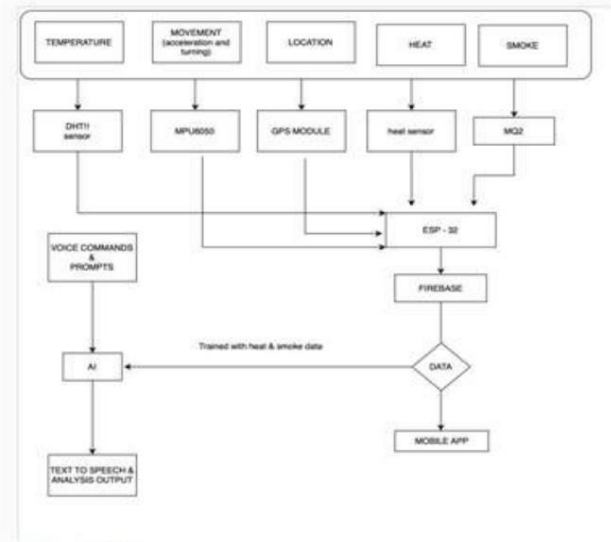


Fig 1.1 Architecture Diagram


FIG 1.0 proposed AI smart jacket

**SETUP:**

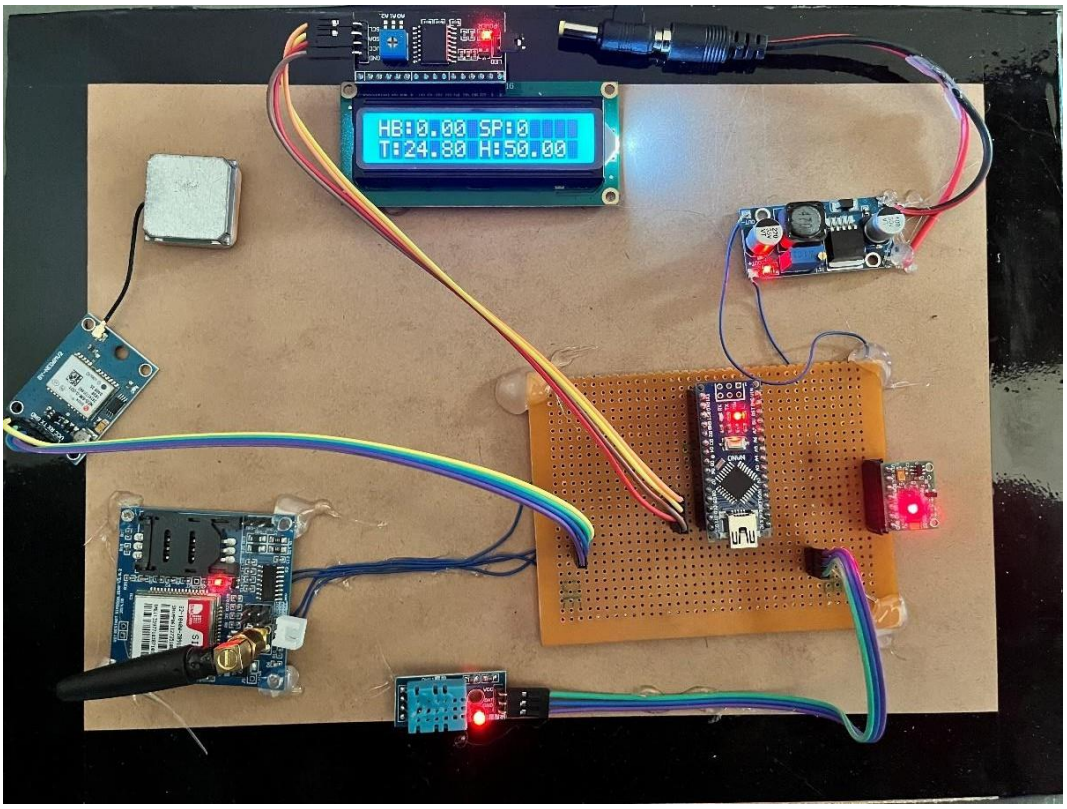


Fig.3 Simulated to test the IoT system

**CODE:**

**ARDUINO UNO IDE (C++)**

```
void loop()
{
  // Make sure to call update as fast as possible
  pox.update();
  if (millis() - tsLastReport > REPORTING_PERIOD_MS) {
    Serial.print("<H"); Serial.print(pox.getHeartRate()); Serial.print(">");
    Serial.print("<S"); Serial.print(pox.getSpO2()); Serial.println(">");
    lcd.setCursor(0, 0);
    lcd.print(F("HB:")); lcd.print(pox.getHeartRate()); lcd.print(F(" "));
    lcd.print(F("SP:")); lcd.print(pox.getSpO2()); lcd.print(F(" "));
    t = dht.readTemperature();
    h = dht.readHumidity();
    lcd.setCursor(0, 1);
    lcd.print(F("T:")); lcd.print(t); lcd.print(F(" "));
    lcd.print(F("H:")); lcd.print(h); lcd.print(F("   "));

    tsLastReport = millis();
  }
  if(t >45 || h > 80)
    {
      if(t > 45)
      {
        pinMode(13,HIGH);
        read_gps_location();
        send_msg_location(phone_number,"Temp Alert : ");
        initPulseSensor();
      }
      else if(h > 80)
      {
        pinMode(13,HIGH);
        read_gps_location();
        send_msg_location(phone_number,"Humidity Alert : ");
        initPulseSensor();
      }
    }
    else
    {
      pinMode(13,LOW);
    }
}
```

*Fig.4 Arduino Code*

**PYTHON (OPENCV & AI PROCESSING)**

```
import random
import time
import
requests

# Define Threshold Limits
TEMP_LIMIT = 38.0          # Body temperature in °C
HUMIDITY_LIMIT = 70.0      # Humidity in %
HEART_RATE_HIGH = 120      # Max Heart Rate in BPM
```

```
HEART_RATE_LOW = 50       # Min Heart Rate in
BPM SPO2_LIMIT = 95  # Min SpO2 in %

# Wi-Fi / Server Endpoint (Change this to your actual server)
SERVER_URL = "http://your-server.com/alert"

# Function to Simulate Reading Sensor Values
def read_sensor_values():
    # Replace random values with actual sensor readings if available
    temperature = random.uniform(35, 41) # Simulate Temp (°C)
    humidity = random.uniform(30, 85)      # Simulate Humidity (%)
    heart_rate = random.randint(40, 130) # Simulate Heart Rate (BPM)
    spo2 = random.randint(85, 100)         # Simulate SpO2 (%)

    print("====== Current Readings ======")
    print(f" Body Temp: {temperature:.2f} °C")
    print(f" Humidity: {humidity:.2f} %")
    print(f" Heart Rate: {heart_rate} BPM")
    print(f" SpO2: {spo2} %")

    return temperature, humidity, heart_rate, spo2

# Function to Run AI-Based Prediction
def predict_anomaly(temperature, humidity, heart_rate, spo2):
    # Basic threshold-based anomaly detection
    if (temperature > TEMP_LIMIT or
        humidity > HUMIDITY_LIMIT or
        heart_rate > HEART_RATE_HIGH or
        heart_rate < HEART_RATE_LOW or
        spo2 < SPO2_LIMIT):
        return True
    return False

# Function to Send Alert Message
def send_alert_message(message):
    payload = {
        "alert": message,
        "device_id": "ESP32_SIMULATED",
        "timestamp": time.strftime("%Y-%m-%d %H:%M:%S")
    }
    try:
        response = requests.post(SERVER_URL, json=payload)
        if response.status_code == 200:
            print("📩 Message Sent to Server!")
        else:
            print(f"⚠️ Failed to send message. Status code: {response.status_code}")
    except Exception as e:
        print(f"⚠️ Error sending message: {e}")

# Main Function
def main():
    print("💻 AI Health Monitoring Started")
    while True:
        # Read sensor values
        temp, hum, hr, sp = read_sensor_values()

        # Run AI prediction
        if predict_anomaly(temp, hum, hr, sp):
            print("🚨 ALERT: Abnormal Condition Detected!")
            alert_message = (
                f"Abnormal Condition! "
                f"Temp: {temp:.2f}°C, Humidity: {hum:.2f}%, "
                f"Heart Rate: {hr} BPM, SpO₂: {sp}%"
            )
            send_alert_message(alert_message)
        else:
```

```
        print("✅ All readings are normal.")

        # Wait for 5 seconds before next reading
        time.sleep(5)

# Run the Script
if __name__ == "__main__": main()
```

---

**OUTPUT:**

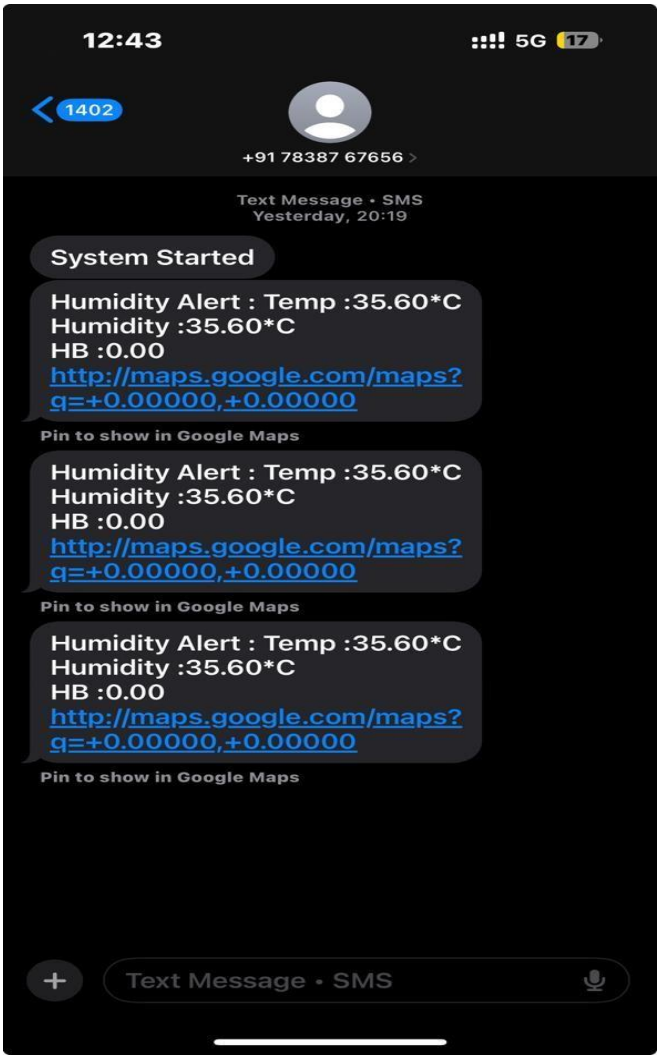


*Fig.5* System Architecture Diagram

**RESULT:**

The IoT-Enabled Military Smart Jacket has successfully demonstrated its potential as an innovative solution for real-time health monitoring, location tracking, and environmental sensing in military operations. By integrating advanced IoT hardware with AI-driven predictive analytics, the jacket continuously monitors vital health parameters, including heart rate, ECG, $SpO_2$, and body temperature, while also tracking location and environmental conditions.

The system accurately detects health anomalies such as blood clots, heart attacks, or heat exhaustion, providing real-time alerts for rapid medical intervention. AI-powered predictive analysis has shown promising results in identifying early signs of fatigue and other health risks, enhancing soldier safety and combat readiness.

Location tracking using GPS enables effective mission coordination, while the integration of automatic SOS alerts and fall detection ensures that emergencies are immediately reported to command centers. The use of LoRa or NB-IoT for long-range data transmission ensures reliable communication even in remote or hostile environments.

This smart jacket offers a scalable and cost-effective solution adaptable to various military and rescue scenarios. The project not only demonstrates the practical application of IoT and AI but also paves the way for smarter, safer, and more efficient defense technology.

**PRECAUTIONS:**

- **Secure Connections:** Ensure all wiring between ESP32, sensors, and GPS modules is properly secured to prevent loose contacts or short circuits that could disrupt data transmission or sensor readings.
- **Microcontroller Reset:** Press the reset button on the ESP32 before uploading the firmware to ensure it enters programming mode correctly, avoiding upload failures.
- **Stable Network Connectivity:** Verify that the LoRa or NB-IoT modules are functioning properly and connected to a reliable network to maintain uninterrupted data transmission, even in remote areas.
- **Sensor Calibration:** Regularly calibrate health sensors (MAX30102, AD8232) and environmental sensors to ensure accurate readings. Calibration should be done under different environmental conditions for optimal performance.
- **Battery Optimization:** Monitor battery performance and implement power management techniques to extend battery life, such as low-power modes or solar charging where applicable.
- **Robust Enclosure:** Ensure the jacket materials are durable enough to withstand extreme environmental conditions (e.g., heat, cold, humidity) without affecting sensor performance.
- **Data Security:** Implement end-to-end encryption and secure data storage practices to prevent unauthorized access to sensitive soldier data.
- **Threshold Tuning:** Run multiple field tests with different scenarios to fine-tune AI models and threshold values for anomaly detection, ensuring accurate alerts while minimizing false positives.
- **Regular Maintenance:** Schedule routine inspections of all hardware components to detect signs of wear or malfunction, ensuring long-term reliability.