**ALGORITHMS**

**Algorithm 1 – Continuous Health Monitoring & Abnormality Detection**

**Scenario: Wristband is monitoring user vitals continuously. LED blinks for minor issues, SOS triggered for severe abnormalities.**

**BEGIN**

**INITIALIZE health\_sensor, ESP32, LED, Wi-Fi, SOS\_Module**

**CONNECT ESP32 to local computer for logging**

**SET safe\_ranges = {heart\_rate: [60, 100], spo2: [95, 100]}**

**LOOP FOREVER:**

**READ heart\_rate, spo2 FROM health\_sensor**

**DISPLAY readings ON VSCode monitor**

**LOG readings WITH timestamp**

**IF heart\_rate OR spo2 outside safe\_ranges:**

**IF deviation <= minor\_threshold:**

**ACTIVATE LED in blinking mode**

**SEND alert TO user display ("Check your health")**

**ELSE IF deviation > minor\_threshold:**

**ACTIVATE LED in red solid mode**

**CALL trigger\_SOS()**

**ELSE:**

**KEEP LED off**

**FUNCTION trigger\_SOS():**

**GATHER GPS location FROM navigation\_module**

**COMPOSE alert\_message WITH vitals and location**

**SEND alert\_message TO Ambulance, Police, Fire services**

**DISPLAY "Emergency Contacted" ON monitor**

**END**

**Algorithm 2 – Fall Detection & Immediate Emergency Response**

**Scenario: Elderly user wearing the band falls; system triggers SOS without waiting for manual input.**

**BEGIN**

**INITIALIZE accelerometer\_sensor, ESP32, Wi-Fi, SOS\_Module**

**SET fall\_threshold = predefined\_acceleration\_value**

**SET response\_time = 15 seconds**

**LOOP FOREVER:**

**READ acceleration\_values FROM accelerometer\_sensor**

**IF acceleration >= fall\_threshold:**

**ACTIVATE LED red solid mode**

**DISPLAY "Possible Fall Detected"**

**START countdown\_timer = response\_time**

**WHILE countdown\_timer > 0:**

**IF user presses CANCEL button:**

**LOG "Fall Alert Cancelled"**

**DEACTIVATE LED**

**BREAK**

**DECREMENT countdown\_timer**

**IF countdown\_timer == 0:**

**CALL trigger\_SOS()**

**FUNCTION trigger\_SOS():**

**READ vitals FROM health\_sensor**

**GATHER GPS location**

**COMPOSE emergency\_packet WITH vitals + location**

**SEND emergency\_packet TO emergency\_services**

**DISPLAY "Emergency Contacted"**

**END**

**Algorithm 3 – Health Monitoring with Integrated Map Navigation for Rescue**

**Scenario: Upon detecting severe abnormality, system sends live vitals + GPS to SOS and guides rescue team via map.**

**BEGIN**

**INITIALIZE health\_sensor, GPS\_module, ESP32, Wi-Fi, SOS\_Module**

**SET critical\_thresholds = {heart\_rate: <40 OR >150, spo2: <85}**

**LOOP FOREVER:**

**READ heart\_rate, spo2**

**IF heart\_rate OR spo2 in critical\_thresholds:**

**ACTIVATE LED red solid mode**

**DISPLAY "Critical Health Alert"**

**CALL send\_navigation\_assist()**

**FUNCTION send\_navigation\_assist():**

**GET current\_location FROM GPS\_module**

**GET nearest\_hospital FROM map\_navigation\_API**

**COMPOSE SOS\_packet WITH:**

**- current\_location**

**- nearest\_hospital\_route**

**- live\_vitals**

**SEND SOS\_packet TO:**

**- Ambulance Dispatch**

**- Police**

**- Fire Services**

**DISPLAY route TO nearest\_hospital ON rescue\_dashboard**

**END**

**Algorithm 4: Normal Operation – Health Monitoring & SOS Trigger**

**START**

**Initialize ESP32 Microcontroller**

**Initialize Health Sensor (MAX30100 or equivalent)**

**Initialize LED indicators (GREEN, YELLOW, RED)**

**Initialize SOS Software Module**

**Initialize Map Navigation Module**

**Connect ESP32 to Computer (VS Code Serial Monitor)**

**WHILE (System is ON) DO**

**Read heart\_rate, spo2 from Health Sensor**

**IF (Data is valid) THEN**

**IF (values within safe range) THEN**

**Turn ON GREEN LED**

**Log data to computer display**

**ELSE IF (minor abnormality detected) THEN**

**Turn ON YELLOW LED**

**Alert user via beep/vibration**

**ELSE IF (major abnormality detected) THEN**

**Turn ON RED LED**

**Trigger SOS sequence:**

**- Send health data & GPS coordinates**

**- Call Ambulance, Police, Fire Service**

**- Display location in Map Navigation Module**

**Log incident in system**

**ENDIF**

**ELSE**

**Display "Sensor Error" on Computer**

**ENDIF**

**Wait for 1 second**

**END WHILE**

**STOP**

**Algorithm 5: Health Sensor Failure Handling**

**START**

**Initialize ESP32 and system modules**

**Try connecting to Health Sensor**

**IF (Sensor not detected) THEN**

**Display "Sensor Connection Failed"**

**Blink RED LED continuously**

**Send notification to user’s mobile (via connected software)**

**Activate Backup Protocol:**

**- Use last known valid readings (if available)**

**- Request user to manually check health condition**

**- If user presses SOS button on wristband:**

**→ Trigger SOS call to emergency contacts with location**

**Log failure event for maintenance**

**ELSE**

**Proceed with normal health monitoring (Algorithm 1)**

**ENDIF**

**STOP**

**Algorithm 3: Network / SOS Communication Failure**

**START**

**Initialize ESP32, Health Sensor, LED indicators**

**Initialize SOS Software Module with communication channels:**

**- Primary: Internet-based API call**

**- Secondary: GSM-based call/text**

**- Tertiary: Offline Alert via stored emergency contacts**

**WHILE (System is ON) DO**

**Read health data from sensor**

**IF (Major abnormality detected) THEN**

**Attempt Primary SOS via Internet**

**IF (Primary SOS fails) THEN**

**Attempt Secondary SOS via GSM**

**IF (Secondary SOS fails) THEN**

**Store emergency message locally**

**Trigger Offline Alert Mode:**

**- Flash RED LED in distress pattern**

**- Play loud audible alarm**

**- Show location on device display**

**ENDIF**

**ENDIF**

**ELSE**

**Continue normal monitoring**

**ENDIF**

**Wait 1 second**

**END WHILE**

**STOP**