**Internet of Things Fundamentals**

*Subject Project*

BS AI 6th Smester SP-25 (AIE-3079)

Date: 26-Jun-25

**Project Title:**

**SenseAI SmartFit: An IoT-Based Smart Health Wristband with Real-Time Web Monitoring**

**Group Name/no.:**

**Sense AI - 5**

**Team Members:**

|  |  |  |  |
| --- | --- | --- | --- |
| Members | Registration no | Name | Signature |
| **Member-1 (Leader)** | **22-NTU-CS-1362** | **Muhammad Hassaan Raza** |  |
| **Member-2** | **22-NTU-CS-1350** | **Kanza Kashaf** |  |
| **Member-3** | **22-NTU-CS-1360** | **Muhammad Faaiz Imtiaz** |  |
| **Member-4** |  |  |  |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Contributions in % of each Team Members for each component | | | | | |
|  | | Member-1 | Member-2 | Member-3 | Member-4 |
| Distribution Components | | Hassaan | Kanza | Faaiz | Name |
| Coding | ESP32-coding | 60% | 20% | 20% |  |
| Python Coding | 30% | 50% | 20% |  |
| UI Design | | 20% | 60% | 20% |  |
| Database | | 30% | 50% | 20% |  |
| Cloud Integration | | 20% | 60% | 20% |  |
| IoT Gateway | | - | - | - |  |
| Edge Processing | | 60% | 20% | 20% |  |
| Documentation | | 50% | 20% | 30% |  |
| Presentation  Design | | 50% | 20% | 30% |  |
| Hardware Integeration | | 30% | 30% | 40% |  |
| Order Taking App | | 15% | 70% | 15% |  |

*To be filled by the evaluator*

# Team-Based Evaluation (60 Marks)

|  |  |  |
| --- | --- | --- |
| Criteria | Obtained Marks | Out of |
| System Design & Architecture |  | 10 |
| Hardware Integration & Circuit Setup |  | 10 |
| IoT Gateway and Cloud Communication |  | 10 |
| Working Prototype Demonstration |  | 10 |
| Performance & Reliability Testing |  | 10 |
| Presentation |  | 10 |
| Total (Team-Based) |  | 60 |

# Individual-Based Evaluation (40 Marks per Member)

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | Member 1 | Member 2 | Member 3 | Member 4 |
| Criteria |  |  |  |  |
| Understanding of the Project & Role | /10 | /10 | /10 | /10 |
| Code Contribution and Explanation | /10 | /10 | /10 | /10 |
| Q/A VIVA | /10 | /10 | /10 | /10 |
| Documentation/Reporting & Communication | /10 | /10 | /10 | /10 |
| Total (Individual-Based) | /40 | /40 | /40 | /40 |
| Total Overall (60+40) | /100 | /100 | /100 | /100 |
| Weightage Lab Grade (50) |  |  |  |  |

# 1. Abstract / Executive Summary

SenseAI SmartFit is a comprehensive IoT-based smart health monitoring solution that combines embedded sensor hardware, edge AI processing, and cloud-based visualization into a compact wearable wristband. The project focuses on enhancing personal healthcare through real-time data acquisition and interpretation of vital signs including body temperature, heart rate, SpO₂ levels, and motion-based fall detection. The system is powered by the ESP32-C3 SUPER MINI SuperMini microcontroller, which is lightweight, low-power, and supports both Wi-Fi and BLE communication. On-device AI inference using TensorFlow Lite Micro makes the system edge-capable, minimizing latency and reducing dependence on continuous cloud interaction. The integrated web dashboard is built using Next.js and Firebase, offering intuitive visualization and secure user management. This solution is scalable and modular, allowing future enhancements like GPS integration, offline sync, and advanced health analytics.

# 2. Table of Contents

1. Abstract / Executive Summary  
2. Table of Contents  
3. Introduction  
4. Literature Review  
5. Methodology / System Design  
6. Implementation  
7. Results & Discussion  
8. Testing & Validation  
9. Conclusion & Future Work  
10. References  
11. Links

# 3. Introduction

## Background & motivation

With healthcare rapidly shifting towards remote and preventative paradigms, wearable technology has emerged as a cornerstone of modern health infrastructure. The convergence of IoT, AI, and cloud computing allows for intelligent, real-time, and proactive monitoring. In particular, the rising elderly population, increased prevalence of lifestyle-related diseases, and demand for smart diagnostics have made wearable health monitors highly relevant. SenseAI SmartFit addresses this need by offering a modular and extensible health tracking solution.

## Problem statement

Traditional methods of health monitoring are often limited by accessibility, manual data logging, and delayed anomaly detection. Additionally, many commercial wearables restrict customization and control over the data. There is a significant gap in the market for an open, programmable, AI-powered wearable that combines edge intelligence, secure storage, and real-time access. SenseAI SmartFit aims to fill this gap by integrating hardware, embedded software, cloud connectivity, and frontend interfaces into a single project.

## Project goals

* To develop a wearable wristband using ESP32-C3 SUPER MINI with vital health sensors.
* To train and deploy a fall detection AI model on the device.
* To send data to Firebase Realtime Database for logging and alerting.
* To build a user-friendly dashboard using Next.js with real-time charts and metrics.
* To implement a complete authentication and user management system using Firebase Auth.
* To ensure system modularity and future expandability.

# 4. Literature Review

## Relevant IoT/ESP32 concepts

* **ESP32-C3 SUPER MINI Supermini**: A RISC-V based microcontroller with onboard Wi-Fi/BLE capabilities suitable for wearable applications due to its small size and low power consumption.
* **TensorFlow Lite Micro (TFLM)**: Lightweight inference engine optimized for microcontrollers, enabling real-time AI at the edge.
* **Firebase**: Offers Realtime Database, Authentication, and Hosting features that simplify backend development.
* **Next.js**: React-based frontend framework that supports both server-side and client-side rendering, suitable for real-time dashboard interfaces.

## Similar projects/research

* **Xiaomi Mi Band**: Offers limited SDK access and no edge ML capabilities.
* **Fitbit Charge Series**: Closed ecosystem with limited raw data access.
* **HealthPatch MD**: Clinical-grade wearable with high accuracy, but very expensive.
* **Arduino-based Fall Detection Projects**: Mostly offline and lack cloud or frontend integration.

These limitations reinforce the need for a complete, customizable open-source wearable system like SenseAI SmartFit.

# 5. Methodology / System Design

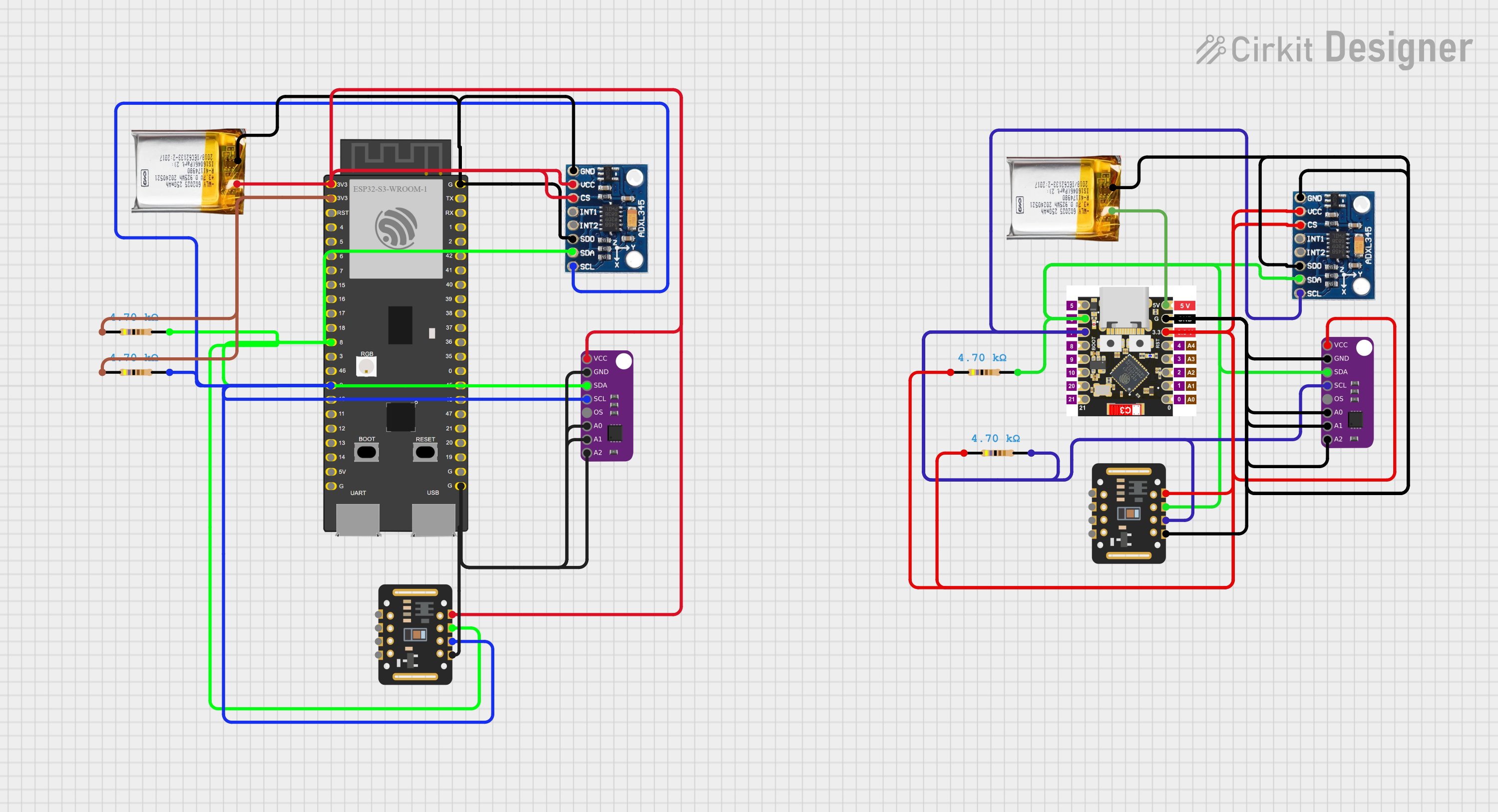
## 5.1 Hardware Components

### List of components (ESP32, sensors, actuators, etc.)

|  |  |  |  |
| --- | --- | --- | --- |
| Component | Model | Purpose | Library Used |
| MCU | ESP32-C3 SUPER MINI | Wi-Fi/BLE, GPIO, edge ML | WiFi.h, HTTPClient.h |
| Temperature Sensor | MAX30205 | High precision skin temperature | ClosedCube\_MAX30205 |
| Heart Rate & SpO₂ | MAX30102 | Red/IR LED-based pulse and oxygen data | SparkFun MAX3010x |
| Accelerometer | ADXL345 | X, Y axis motion + free-fall detection | Adafruit\_ADXL345\_U |
| Optional Sensor | DHT11/DHT22 | Environmental sensing (training only) | Adafruit\_DHT |

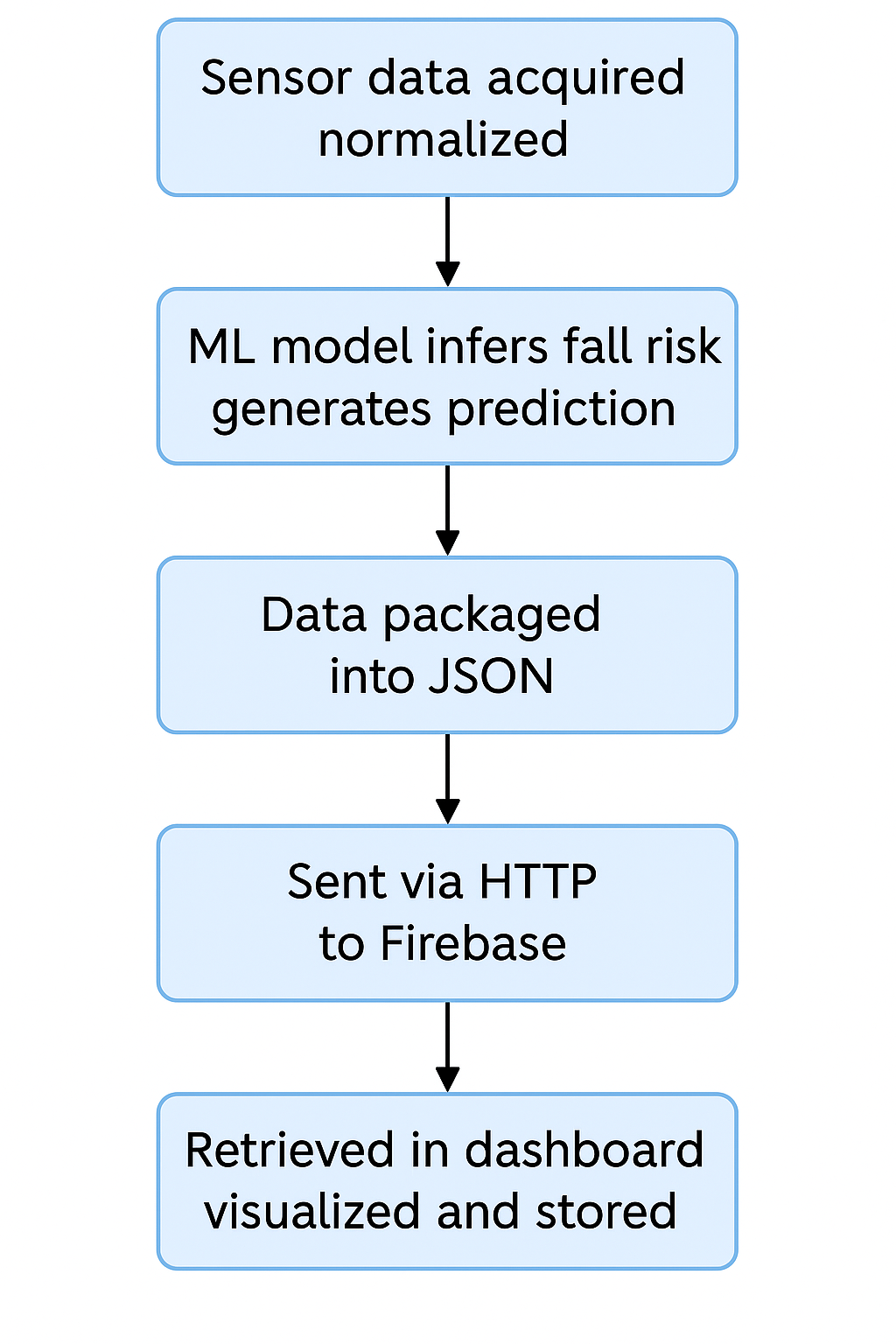
**Power Supply**: 3.7V Li-Po battery + voltage regulator  
**Display (Optional)**: OLED SSD1306 for visual feedback

### Circuit diagram (Fritzing/proteus/other) with labels



## 5.2 Software Design

### Flowchart/system architecture



### Libraries/tools used (Arduino IDE, PlatformIO, MQTT, etc.)

* **Embedded (Arduino/ESP32):**
* **WiFi.h – to connect ESP32 to the internet**
* **HTTPClient.h – to send data via HTTP**
* **Wire.h – for I2C sensor communication**
* **MAX3010x.h, ClosedCube\_MAX30205.h – for heart rate and temperature sensors**
* **Adafruit\_ADXL345\_U.h – for motion/fall detection**
* **tensorflow/lite/... – for Edge AI inference**
* **PlatformIO / Arduino IDE – development environment for uploading code**
* **Web (Frontend & Backend):**
* **Next.js 14 – your main frontend framework**
* **Firebase SDK – for authentication, database, storage**
* **Tailwind CSS – for styling the UI**
* **Chart.js – for graph visualizations**
* **React Context API – for managing data and auth state**

### Pseudocode (if applicable)

Start loop every 10 seconds:

Read temperature from MAX30205

Read heart rate and SpO2 from MAX30102

Read X and Y acceleration from ADXL345

Normalize acceleration values

Run ML model on normalized data

If fall probability > 0.5:

Set fall\_status to "Fall Detected"

Else:

Set fall\_status to "Stable"

Create JSON object with all readings and timestamp

Send JSON to Firebase Realtime DB

# 6. Implementation

## Step-by-step setup (wiring, configurations)

**Embedded System Setup**

* **Sensor Initialization**: Each sensor is initialized with retry conditions and error handling.
* **Time Synchronization**: configTime with NTP ensures all data timestamps are accurate.
* **Data Upload**: Sensor values including heart rate, temperature, and accelerometer readings are converted to JSON and sent over HTTP using Firebase token-based authentication.

**Edge AI Workflow**

* **Data Generation**: 15,000 samples of synthetic and real-world data were created.
* **Preprocessing**: Min-Max normalization applied to x and y values.
* **Model Architecture**: 2D input → Dense(32) → Dropout → Dense(16) → Dropout → Dense(1, sigmoid)
* **Model Evaluation**: Achieved 90%+ precision/recall on validation set.
* **Deployment**: .tflite → .h array using xxd tool.

**Frontend Implementation**

* **Page Components**: HeroSection, FeaturesSection, LiveMonitoring, OrderPage, Dashboard, Profile.
* **State Management**: Context API for auth and data handling.
* **DataContext**: Handles Firebase DB subscriptions and updates to health metrics.
* **Authentication**: Email/Password Firebase Auth with session persistence and route protection.

## Arduino Code:

#include <WiFi.h>

#include <HTTPClient.h>

#include <Wire.h>

#include "MAX30105.h"

#include "spo2\_algorithm.h"

#include <Adafruit\_ADXL345\_U.h>

#include <ClosedCube\_MAX30205.h>

#include "tensorflow/lite/micro/micro\_interpreter.h"

#include "tensorflow/lite/micro/micro\_mutable\_op\_resolver.h"

#include "tensorflow/lite/schema/schema\_generated.h"

#include "model\_data.h"

#include <time.h>

// WiFi credentials

const char\* ssid = "NTU FSD";

const char\* password = "";

const String USER\_UID = "NN4WDAbitlXXPefpPi8BFmL0etV2";

// Firebase Config

const String FIREBASE\_HOST = "senseai-db-a95b9-default-rtdb.firebaseio.com";

const String FIREBASE\_AUTH = "bpS289o5XH3TEcst0m4wIlaEbt4F95f6gRwxWIVI";

const String FIREBASE\_PATH = "/users/" + USER\_UID + "/rawSensorData.json";

// Sensor Config

#define SDA\_PIN 8

#define SCL\_PIN 9

#define BUFFER\_SIZE 100

MAX30105 pulseSensor;

Adafruit\_ADXL345\_Unified accel(12345);

ClosedCube\_MAX30205 tempSensor;

uint32\_t irBuf[BUFFER\_SIZE], redBuf[BUFFER\_SIZE];

// TensorFlow Lite Micro

constexpr int kTensorArenaSize = 10 \* 1024;

uint8\_t tensor\_arena[kTensorArenaSize];

const tflite::Model\* model = nullptr;

tflite::MicroInterpreter\* interpreter = nullptr;

tflite::MicroMutableOpResolver<3> resolver;

TfLiteTensor\* input = nullptr;

TfLiteTensor\* output = nullptr;

// Timing

const unsigned long SEND\_INTERVAL = 10000;

unsigned long lastSendTime = 0;

// Accelerometer normalization (adjust according to your dataset)

float X\_MIN = -17.0, X\_MAX = 17.0;

float Y\_MIN = -17.0, Y\_MAX = 17.0;

// ==================== SETUP ==================== //

void setup() {

  Serial.begin(115200);

  Wire.begin(SDA\_PIN, SCL\_PIN);

  Wire.setClock(100000);

  connectWiFi();

  configTime(18000, 0, "pool.ntp.org", "time.nist.gov");  // GMT+5

  while (time(nullptr) < 100000) delay(500);

  Serial.println("Time synced!");

  // Sensors Initialization

  if (!pulseSensor.begin(Wire, I2C\_SPEED\_FAST)) while (1) Serial.println("MAX30102 not found!");

  pulseSensor.setup(0x1F, 1, 2, 100, 411, 4096);

  Serial.println("MAX30102 OK");

  if (!accel.begin()) while (1) Serial.println("ADXL345 not found!");

  accel.setRange(ADXL345\_RANGE\_16\_G);

  Serial.println("ADXL345 OK");

  tempSensor.begin(0x48);

  Serial.println("MAX30205 OK");

  resetMAX30205();

  // TensorFlow Lite setup

  model = tflite::GetModel(model\_tflite);

  resolver.AddFullyConnected();

  resolver.AddRelu();

  resolver.AddLogistic();

  static tflite::MicroInterpreter static\_interpreter(model, resolver, tensor\_arena, kTensorArenaSize);

  interpreter = &static\_interpreter;

  if (interpreter->AllocateTensors() != kTfLiteOk) {

    Serial.println("Tensor allocation failed"); while (1);

  }

  input = interpreter->input(0);

  output = interpreter->output(0);

  Serial.println("Setup complete.");

}

// ==================== LOOP ==================== //

void loop() {

  if (millis() - lastSendTime >= SEND\_INTERVAL) {

    uploadDataToFirebase();

    lastSendTime = millis();

  }

}

// ==================== FUNCTIONS ==================== //

void connectWiFi() {

  WiFi.begin(ssid, password);

  Serial.print("Connecting to WiFi");

  while (WiFi.status() != WL\_CONNECTED) {

    delay(500);

    Serial.print(".");

  }

  Serial.println("\nWiFi connected.");

}

void resetMAX30205() {

  Wire.beginTransmission(0x48);

  Wire.write(0x01);

  Wire.write(0x80);

  Wire.endTransmission();

  delay(100);

}

String getFormattedTime() {

  time\_t now = time(nullptr);

  struct tm\* timeinfo = localtime(&now);

  char buffer[30];

  strftime(buffer, sizeof(buffer), "%Y-%m-%d %H:%M:%S", timeinfo);

  return String(buffer);

}

void uploadDataToFirebase() {

  if (WiFi.status() != WL\_CONNECTED) {

    Serial.println("WiFi not connected!");

    connectWiFi();

    return;

  }

  // ==================== READ SENSORS ==================== //

  float temperature = tempSensor.readTemperature();

  if (temperature < 10.0 || temperature > 45.0) {

    Serial.println("MAX30205 error! Retrying...");

    delay(100);

    resetMAX30205();  // Reset sensor before retry

    temperature = tempSensor.readTemperature();

    if (temperature < 10.0 || temperature > 45.0) {

      Serial.println("MAX30205 failed after retry. Setting temperature=null.");

      temperature = NAN;  // Mark as invalid

    }

  }

  // 2. Accelerometer

  sensors\_event\_t event;

  accel.getEvent(&event);

  float accel\_x = event.acceleration.x;

  float accel\_y = event.acceleration.y;

  // Normalize

  input->data.f[0] = (accel\_x - X\_MIN) / (X\_MAX - X\_MIN);

  input->data.f[1] = (accel\_y - Y\_MIN) / (Y\_MAX - Y\_MIN);

  interpreter->Invoke();

  float fall\_probability = output->data.f[0];

  String fall\_status = (fall\_probability > 0.5) ? "Fall Detected" : "Stable";

  // 3. MAX30102 Pulse Sensor

  for (int i = 0; i < BUFFER\_SIZE; i++) {

    while (!pulseSensor.available()) pulseSensor.check();

    redBuf[i] = pulseSensor.getRed();

    irBuf[i] = pulseSensor.getIR();

    pulseSensor.nextSample();

  }

  int32\_t spo2, hr;

  int8\_t spo2Valid, hrValid;

  maxim\_heart\_rate\_and\_oxygen\_saturation(irBuf, BUFFER\_SIZE, redBuf, &spo2, &spo2Valid, &hr, &hrValid);

  String timestamp = getFormattedTime();

  // ==================== BUILD JSON ==================== //

  String json = "{";

  json += "\"temperature\":";

  json += isnan(temperature) ? "null" : String(temperature, 2);

  json += ",";

  json += "\"accel\_x\":" + String(accel\_x, 2) + ",";

  json += "\"accel\_y\":" + String(accel\_y, 2) + ",";

  json += "\"fall\_probability\":" + String(fall\_probability, 4) + ",";

  json += "\"fall\_status\":\"" + fall\_status + "\",";

  json += "\"heartrate\":";

  json += (hrValid) ? String(hr) : "null";

  json += ",";

  json += "\"spo2\":";

  json += (spo2Valid) ? String(spo2) : "null";

  json += ",";

  json += "\"timestamp\":\"" + timestamp + "\"";

  json += ",\"user\_id\":\"" + USER\_UID + "\"}";

  Serial.println("Sending JSON:");

  Serial.println(json);

  // ==================== SEND TO FIREBASE ==================== //

  HTTPClient http;

  String url = "https://" + FIREBASE\_HOST + FIREBASE\_PATH + "?auth=" + FIREBASE\_AUTH;

  http.begin(url);

  http.addHeader("Content-Type", "application/json");

  int httpCode = http.POST(json);

  if (httpCode == HTTP\_CODE\_OK || httpCode == HTTP\_CODE\_CREATED) {

    Serial.println("Data sent to Firebase successfully.");

  } else {

    Serial.printf("Firebase Error. Code: %d\n", httpCode);

  }

  http.end();

}

## Challenges & solutions

1. I2C Communication Conflicts  
   Challenge: Interfacing multiple sensors over I2C resulted in NACK (No Acknowledgment) errors.  
   Solution: Added 4.7kΩ pull-up resistors to the SDA and SCL lines, which stabilized the I2C communication.
2. Unstable Readings from MAX30102 Sensor  
   Challenge: The MAX30102 library produced inconsistent and fluctuating SpO₂ values.  
   Solution: Integrated the spo2\_algorithm library, which provided more reliable and accurate sensor readings.
3. Unknown I2C Addresses  
   Challenge: Difficulty in identifying the correct I2C addresses for some sensors.  
   Solution: Used an I2C scanner utility to detect and confirm the correct addresses of all connected devices.
4. Firebase Data Write Path Issues  
   Challenge: Encountered user ID-related errors while writing data from the Next.js frontend to Firebase.  
   Solution: Ensured accurate tracking and construction of the Firebase database path using the authenticated user\_id, allowing successful data writes.

# 7. Results & Discussion

## Realtime Monitoring:

Four live updating charts showing heart rate, SpO₂, temperature, and motion.

A screenshot of a computer

AI-generated content may be incorrect.

## Dashboard – Summary:

A screenshot of a computer

AI-generated content may be incorrect.

## Fall Detection:

Real-time status "Fall Detected" or "Stable" displayed.

A screenshot of a computer

AI-generated content may be incorrect.

## User Interaction:

Users can log in, view their stats, and place wristband orders.

A screenshot of a computer

AI-generated content may be incorrect.

## Admin View:

Additional access to all user orders and device logs.

**A screenshot of a computer

AI-generated content may be incorrect.**

**A screenshot of a computer

AI-generated content may be incorrect.**

## Confusion matrix

A screenshot of a graph

AI-generated content may be incorrect.

## Firebase DB snapshot

**A screenshot of a computer

AI-generated content may be incorrect.**

# 8. Testing & Validation / Limitations

## Test cases

Each subsystem was tested individually and as part of the whole pipeline.

**Hardware Tests**:

* Sensor accuracy under varied environmental conditions
* ESP32 response under USB power

**Software Tests**:

* Firebase Auth Failure Modes
* LiveMonitoring chart refresh behavior
* Anomaly detection with synthetic test cases

**Edge AI Testing**:

* Evaluated using confusion matrix, ROC curve, and real fall emulation

## Limitations

* Model only detects sudden falls; gradual collapses may not trigger alerts.
* BLE not utilized; currently requires internet access.
* Battery management is not fully optimized.
* No haptic or audible alert (e.g., buzzer or vibration) implemented.

# 9. Conclusion & Future Work

## Key takeaways

SenseAI SmartFit demonstrates the potential of combining embedded health sensors, edge computing, cloud data infrastructure, and a responsive frontend in one unified system. The successful integration of all project modules showcases a practical and customizable solution for personal health monitoring.

## Potential improvements (e.g., adding AI, cloud integration)

**Future Enhancements**

* BLE & Mobile App: Enable offline data storage and syncing.
* GPS & Emergency SMS: Auto-alert caretakers upon fall detection.
* Edge Retraining: Update ML model from dashboard interface.
* Power Management: Add deep sleep and battery level monitoring.
* Community Platform: Allow users to share anonymized trends with physicians.

# 10. References

Citations (web/YouTube links, tutorials, research papers)

* TensorFlow Lite Micro: <https://www.tensorflow.org/lite/microcontrollers>
* Firebase Realtime DB: <https://firebase.google.com/docs/database>
* ESP32-C3 SUPER MINI Documentation: <https://docs.espressif.com/projects/esp-idf/en/latest/esp32c3>
* ADXL345 Datasheet & Tutorial: <https://learn.adafruit.com/adxl345-digital-accelerometer>
* MAX30102: <https://www.maximintegrated.com/en/products/sensors/MAX30102.html>

# 11. Links

## GitHub Repository Link (links from each member)

* Muhammad Hassaan Raza
* <https://github.com/HassaanRazaX/IoT_Labs/tree/main/Project>
* Kanza Kashaf
* <https://github.com/KanzaKashaf/IOT-Labs-1350>
* Muhammad Faaiz Imtiaz
* <https://github.com/mfaaizi/Project_final_6th>

## Video Demo:

<https://drive.google.com/file/d/12LW_ArNfJZc3zNxwpGK0QpqyHTzfsJXZ/view?usp=sharing>