IOT CASE STUDY

2211CS020384
SREEJA

1) Interface a Camera Module to Create an Attendance Monitoring System of Your Class Room.

1. Hardware Selection:

- **Microcontroller:** ESP32 (with enough processing power and camera interface).
- **Camera Module:** OV2640 or OV7670.
    - OV2640 is recommended because it integrates well with the ESP32-CAM board.

2. Required Components:

- ESP32-CAM board (with OV2640)
- MicroSD card (optional for logging)
- Power supply (5V)
- External LEDs for low-light conditions (optional)

3. Embedded C Code for Camera Initialization and Capture:

Here's a simple outline to capture an image and store it.

Code:

```
#include "esp_camera.h"
#include <WiFi.h>

#define CAMERA_MODEL_AI_THINKER // Define the camera module

// Wi-Fi credentials
const char* ssid = "YourSSID";
const char* password = "YourPassword";

// Camera Pins Configuration for AI Thinker
#define PWDN_GPIO_NUM     -1
#define RESET_GPIO_NUM    -1
#define XCLK_GPIO_NUM      0
#define SIOD_GPIO_NUM     26
#define SIOC_GPIO_NUM     27

#define Y9_GPIO_NUM       35
#define Y8_GPIO_NUM       34
#define Y7_GPIO_NUM       39
#define Y6_GPIO_NUM       36
#define Y5_GPIO_NUM       21
#define Y4_GPIO_NUM       19
#define Y3_GPIO_NUM       18
#define Y2_GPIO_NUM        5
#define VSYNC_GPIO_NUM    25
#define HREF_GPIO_NUM     23
#define PCLK_GPIO_NUM     22
```

```
void startCameraServer();

void setup() {
  Serial.begin(115200);

  // Initialize Wi-Fi connection
  WiFi.begin(ssid, password);
  while (WiFi.status() != WL_CONNECTED) {
    delay(500);
    Serial.print(".");
  }
  Serial.println("");
  Serial.println("Wi-Fi connected");

  camera_config_t config;
  config.ledc_channel = LEDC_CHANNEL_0;
  config.ledc_timer = LEDC_TIMER_0;
  config.pin_d0 = Y2_GPIO_NUM;
  config.pin_d1 = Y3_GPIO_NUM;
  config.pin_d2 = Y4_GPIO_NUM;
  config.pin_d3 = Y5_GPIO_NUM;
  config.pin_d4 = Y6_GPIO_NUM;
  config.pin_d5 = Y7_GPIO_NUM;
  config.pin_d6 = Y8_GPIO_NUM;
  config.pin_d7 = Y9_GPIO_NUM;
  config.pin_xclk = XCLK_GPIO_NUM;
  config.pin_pclk = PCLK_GPIO_NUM;
  config.pin_vsync = VSYNC_GPIO_NUM;
  config.pin_href = HREF_GPIO_NUM;
  config.pin_sscb_sda = SIOD_GPIO_NUM;
  config.pin_sscb_scl = SIOC_GPIO_NUM;
  config.pin_pwdn = PWDN_GPIO_NUM;
  config.pin_reset = RESET_GPIO_NUM;
  config.xclk_freq_hz = 20000000;
  config.pixel_format = PIXFORMAT_JPEG;

  // Init with high specs to capture VGA image size
  config.frame_size = FRAMESIZE_VGA;
  config.jpeg_quality = 12;
  config.fb_count = 1;

  // Camera init
  esp_err_t err = esp_camera_init(&config);
  if (err != ESP_OK) {
    Serial.printf("Camera init failed with error 0x%x", err);
    return;
  }

  // Start camera server for web streaming (optional)
  startCameraServer();
  Serial.println("Camera ready! Use 'http://<your ESP32 IP address>' to connect");
}

void loop() {
  // Capture and save logic or process frames for facial recognition
}
```

Steps for Face Recognition Integration:

1. **Face Detection Libraries:** Use **ESP-WHO**, an optimized face detection/recognition library for ESP32.
2. **Server Integration:** Capture and process images on an external server for more complex operations like facial recognition using Python frameworks (OpenCV).
3. **Logging and Attendance:** Maintain a database to track recognized faces and record attendance timestamps.

Additional Considerations:

- **Accuracy:** For robust performance, use cloud processing or edge AI with TensorFlow Lite.
- **Lighting:** Ensure proper lighting in the classroom for better face recognition accuracy.
- **Privacy:** Follow regulations related to data storage and privacy for biometric systems.

2) IoT in Logistics and Fleet Management: Analyze how IoT technologies optimize logistics operations,from real-time tracking of shipments to predictive maintenance of transportation fleets.

IoT in Logistics and Fleet Management: Overview

IoT technologies are revolutionizing logistics and fleet management by providing real-time visibility, predictive maintenance, and operational efficiency improvements. IoT sensors, GPS trackers, and wireless communication modules allow continuous data collection and analytics to streamline operations.

Key IoT Features for Logistics Optimization

1. **Real-Time Tracking:** GPS modules track vehicle locations and shipment progress.
2. **Predictive Maintenance:** Sensors monitor vehicle conditions to anticipate failures.
3. **Environmental Monitoring:** Temperature, humidity, and vibration sensors protect sensitive cargo.
4. **Driver Behavior Monitoring:** Accelerometers and gyroscopes track acceleration, braking, and driving patterns.
5. **Fuel Management:** Flow meters and telemetry systems monitor fuel usage.

Model Design for IoT-Based Fleet Management

- **Microcontroller:** ESP32 (sufficient for real-time communication and sensor interfacing)
- **Communication Module:** GSM (SIM800L) or LoRa for long-range communication
- **Sensors:**
    - GPS Module (NEO-6M)
    - Temperature Sensor (DHT22)
    - Accelerometer (MPU6050)
    - Fuel Level Sensor (capacitive type)

Embedded C Code for GPS Tracking and Data Transmission:

```
#include <TinyGPS++.h>
#include <SoftwareSerial.h>

#define RXPin 4
#define TXPin 3
#define GPSBaud 9600

// Create a TinyGPS++ object
TinyGPSPlus gps;

// Set up a SoftwareSerial object for the GPS module
SoftwareSerial gpsSerial(RXPin, TXPin);

void setup() {
  Serial.begin(115200);
  gpsSerial.begin(GPSBaud);

  Serial.println("GPS Tracker Initialized");
}

void loop() {
  // Read data from the GPS module
  while (gpsSerial.available() > 0) {
    gps.encode(gpsSerial.read());
  }

  if (gps.location.isUpdated()) {
    Serial.print("Latitude: ");
    Serial.println(gps.location.lat(), 6);
    Serial.print("Longitude: ");
    Serial.println(gps.location.lng(), 6);
  }

  // Simulate sending data over a network
  sendToServer(gps.location.lat(), gps.location.lng());
  delay(1000); // Adjust delay as necessary
}

void sendToServer(double latitude, double longitude) {
  // Placeholder function for sending GPS data
  Serial.print("Sending to Server: ");
  Serial.print("Lat: ");
  Serial.print(latitude, 6);
  Serial.print(" Long: ");
  Serial.println(longitude, 6);
}
```

Additional Integration Points

Environmental Monitoring:

1. Interface temperature and humidity sensors to monitor perishable goods.
2. Example: Use DHT22 with appropriate libraries for real-time environmental data.

Predictive Maintenance:

1. Monitor engine parameters like temperature, pressure, and vibration using relevant sensors.
2. Use an accelerometer (MPU6050) for vibration detection and fault prediction.

Communication Protocols:

1. MQTT for efficient data transmission to cloud servers.
2. GSM/LoRa modules for remote areas.

Cloud Integration

- Utilize platforms like AWS IoT, Azure IoT Hub, or ThingsBoard for data analytics and dashboard creation.
- Implement machine learning models to predict maintenance schedules based on collected data.

3) IoT in Healthcare for Remote Patient Monitoring:examine the applications of IoT in healthcare, specifically focusing on how it enables remote patient monitoring improves healthcare delivery,and enhances patient outcomes.

IoT in Healthcare for Remote Patient Monitoring

IoT is transforming healthcare by enabling remote patient monitoring (RPM), improving care delivery, and enhancing patient outcomes. Devices such as wearable sensors, smart medical devices, and cloud integration allow healthcare professionals to monitor patients outside traditional healthcare settings.

Key Applications of IoT in Healthcare

1. **Vital Sign Monitoring:** Continuous tracking of heart rate, blood pressure, oxygen levels, and body temperature.
2. **Chronic Disease Management:** Monitoring glucose levels for diabetic patients or respiratory rates for patients with asthma or COPD.
3. **Emergency Alerts:** Automatic alerts for healthcare providers in case of abnormal readings.
4. **Medication Adherence:** Smart pill bottles that remind patients to take medications and alert caregivers.
5. **Data Analytics & AI:** Machine learning models to detect patterns and predict health risks.

Model for Remote Patient Monitoring System

- **Microcontroller:** ESP32 (supports Wi-Fi/Bluetooth for real-time communication)
- **Sensors:**
    - Heart Rate & SpO2 Sensor (MAX30100 or MAX30102)
    - Body Temperature Sensor (DS18B20 or LM35)
    - ECG Monitoring (AD8232 module)
- **Communication:** Wi-Fi for cloud data transfer via MQTT or HTTP.
- **Cloud Integration:** Google Firebase, AWS IoT, or ThingsBoard for data visualization and alerts.

Embedded C Code Example for Heart Rate and Temperature Monitoring:

```
#include <Wire.h>
#include <Adafruit_Sensor.h>
#include <Adafruit_MAX30102.h>
#include <WiFi.h>
#include <HTTPClient.h>

#define WIFI_SSID "YourSSID"
#define WIFI_PASSWORD "YourPassword"
#define SERVER_URL "http://example.com/upload"

Adafruit_MAX30102 heartSensor;
float bodyTemperature = 0.0;

void setup() {
  Serial.begin(115200);

  // Initialize Wi-Fi
  WiFi.begin(WIFI_SSID, WIFI_PASSWORD);
  while (WiFi.status() != WL_CONNECTED) {
    delay(500);
    Serial.print(".");
  }
  Serial.println("Wi-Fi connected");

  // Initialize Heart Rate Sensor
  if (!heartSensor.begin()) {
```

```
        Serial.println("MAX30102 not detected. Please check wiring.");
        while (1);
    }
    Serial.println("Heart Rate Sensor Initialized");

    // Initialize Temperature Sensor
    pinMode(A0, INPUT);  // Analog input for temperature sensor if using LM35
}

void loop() {
    // Read Heart Rate and SpO2
    int hr, spO2;
    heartSensor.getHeartRate(&hr);
    heartSensor.getSpO2(&spO2);

    // Simulate temperature reading (replace with actual sensor code as needed)
    bodyTemperature = analogRead(A0) * (5.0 / 1023.0) * 100;

    // Display readings
    Serial.print("Heart Rate: ");
    Serial.print(hr);
    Serial.print(" BPM, SpO2: ");
    Serial.print(spO2);
    Serial.print(" %, Body Temperature: ");
    Serial.print(bodyTemperature);
    Serial.println(" °C");
```

```
    // Send Data to Server
    sendDataToServer(hr, spO2, bodyTemperature);

    delay(5000); // Delay for next reading
}

void sendDataToServer(int hr, int spO2, float temperature) {
    if (WiFi.status() == WL_CONNECTED) {
        HTTPClient http;
        http.begin(SERVER_URL);

        // Prepare payload
        String payload = "{\"heart_rate\":" + String(hr) +
                         ",\"spo2\":" + String(spO2) +
                         ",\"temperature\":" + String(temperature) + "}";
        http.addHeader("Content-Type", "application/json");

        int httpResponseCode = http.POST(payload);
        if (httpResponseCode > 0) {
            Serial.println("Data sent successfully");
        } else {
            Serial.println("Error in sending data");
        }
        http.end();
    } else {
        Serial.println("Wi-Fi Disconnected");
    }
```

}

Additional System Features

1. **Data Visualization:**
    1. Display health parameters on a smartphone or cloud dashboard.
2. **Alert System:**
    1. Trigger SMS/email alerts for abnormal values.
3. **Battery Management:**
    1. Include low-power optimization to extend device battery life.

Benefits of IoT-Enabled RPM

1. **Improved Patient Outcomes:** Early detection of health issues leads to better management.
2. **Cost Efficiency:** Reduced hospital visits and improved resource allocation.
3. **Enhanced Access:** Provides care to patients in rural and remote areas.

4) IoT and Augmented Reality for Enhanced Experiences: Exploring the convergence of IoT and Augmented reality to create immersive and interactive experiences,such as AR-assisted maintenance or guided tours.

IoT and Augmented Reality for Enhanced Experiences

The convergence of **IoT** and **Augmented Reality (AR)** enables immersive, interactive experiences that combine real-time sensor data with visual overlays. These applications are reshaping industries such as maintenance, manufacturing, and tourism.

Key Applications

1. **AR-Assisted Maintenance:**
    1. Visual guidance for complex repairs by overlaying real-time data from IoT sensors on machinery.
    2. Remote expert support using AR headsets.
2. **Guided Tours:**
    1. Interactive museum or historical site tours with AR glasses or mobile apps showing contextual information.
    2. IoT beacons provide location-based content.
3. **Smart Home/Factory Monitoring:**
    1. Real-time display of environmental conditions (temperature, humidity) or equipment status.

Model for AR-Enabled IoT System

- **IoT Devices:** ESP32 for environmental monitoring and control.
- **AR Device/Software:** Mobile apps or AR glasses to visualize sensor data.
- **Communication:** Wi-Fi or Bluetooth communication between IoT devices and AR apps.
- **Sensors:** Temperature, humidity, proximity, vibration sensors, etc.
- **Data Integration:** JSON data sent from IoT devices to the AR system.

Embedded C Code for IoT Data Transmission to AR System:

```
#include <WiFi.h>
#include <HTTPClient.h>
#include <Adafruit_Sensor.h>
#include <DHT.h>

#define DHTPIN 4        // Pin for the DHT sensor
#define DHTTYPE DHT22   // Sensor type (DHT22)

// Replace with your network credentials
const char* ssid = "YourSSID";
const char* password = "YourPassword";

const char* serverUrl = "http://example.com/receive";  // AR system server URL

DHT dht(DHTPIN, DHTTYPE);

void setup() {
  Serial.begin(115200);

  // Connect to Wi-Fi
  WiFi.begin(ssid, password);
  while (WiFi.status() != WL_CONNECTED) {
    delay(500);
    Serial.print(".");
  }
  Serial.println("Wi-Fi connected");

  dht.begin();
  Serial.println("DHT Sensor Initialized");
}

void loop() {
  // Read temperature and humidity
  float temperature = dht.readTemperature();
  float humidity = dht.readHumidity();

  if (isnan(temperature) || isnan(humidity)) {
    Serial.println("Failed to read from DHT sensor!");
    return;
  }

  // Send data to AR system
  sendDataToARSystem(temperature, humidity);
```

```
  delay(5000);  // Delay between readings
}

void sendDataToARSystem(float temperature, float humidity) {
  if (WiFi.status() == WL_CONNECTED) {
    HTTPClient http;
    http.begin(serverUrl);

    // Prepare payload
    String payload = "{\"temperature\":" + String(temperature) +
                     ",\"humidity\":" + String(humidity) + "}";

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

    int httpResponseCode = http.POST(payload);

    if (httpResponseCode > 0) {
      Serial.println("Data sent successfully to AR system");
    } else {
      Serial.println("Error in sending data");
    }

    http.end();
  } else {
    Serial.println("Wi-Fi Disconnected");
  }
}
```

AR Visualization Integration

- **AR App Development:**
  - Use platforms like Unity 3D with ARKit (iOS) or ARCore (Android) to visualize real-time data.
  - Integrate with IoT data endpoints using RESTful APIs.
- **Data Visualization:**
  - Overlay sensor data as real-time labels on objects or as 3D charts.
  - Interactive alerts for sensor thresholds in maintenance scenarios.

Benefits of IoT-AR Integration

1. **Enhanced User Experience:** Interactive and engaging interfaces improve learning and operational efficiency.
2. **Improved Efficiency:** AR-assisted maintenance reduces error rates and training times.
3. **Remote Monitoring:** Real-time IoT data visualization through AR minimizes downtime.

5) Wearable IoT Devices for Health and Fitness: Analyze the impact of werable IoT devices, such as fitness trackers and smartwatches, on personal health monitoring,execrise routines,and preventive healthcare.

Wearable IoT Devices for Health and Fitness

Wearable IoT devices, such as fitness trackers and smartwatches, have revolutionized personal health monitoring by providing real-time insights into physical activity, sleep patterns, heart rate, and other health metrics. They empower users to track fitness goals, encourage healthy lifestyles, and aid in preventive healthcare.

Impact on Personal Health Monitoring

1. **Real-Time Data Collection:** Continuous monitoring of heart rate, activity levels, and sleep quality.
2. **Personalized Fitness Routines:** Data-driven insights optimize workout plans.
3. **Preventive Healthcare:** Early detection of irregular health patterns helps in preventive care.
4. **Motivation & Engagement:** Gamification elements like activity goals and achievements motivate users.
5. **Remote Health Monitoring:** Integration with healthcare services for chronic disease management.

Model for a Wearable IoT Health Monitoring Device

- **Microcontroller:** ESP32 for sensor interfacing and Bluetooth/Wi-Fi communication.
- **Sensors:**
    - Heart Rate & SpO2 Sensor (MAX30102)
    - Accelerometer (MPU6050) for activity tracking
    - Temperature Sensor (LM35) for skin temperature monitoring
- **Communication:** Bluetooth for connection to smartphones or fitness apps.
- **Power Source:** Rechargeable battery.

Embedded C Code Example for Heart Rate and Activity Monitoring:

```
#include <Wire.h>

#include <Adafruit_Sensor.h>

#include <Adafruit_MAX30102.h>

#include <MPU6050_light.h>

#include <BLEDevice.h>

#include <BLEServer.h>

#include <BLEUtils.h>

#include <BLE2902.h>
```

```
Adafruit_MAX30102 heartSensor;

MPU6050 mpu(Wire);


// BLE service and characteristic UUIDs

#define SERVICE_UUID        "12345678-1234-1234-1234-1234567890ab"

#define HEART_RATE_UUID     "12345678-1234-1234-1234-1234567890cd"

#define STEP_COUNT_UUID     "12345678-1234-1234-1234-1234567890ef"


BLECharacteristic heartRateCharacteristic(HEART_RATE_UUID,
BLECharacteristic::PROPERTY_NOTIFY);

BLECharacteristic stepCountCharacteristic(STEP_COUNT_UUID,
BLECharacteristic::PROPERTY_NOTIFY);


int stepCount = 0;


void setup() {

   Serial.begin(115200);


   // Initialize sensors

   if (!heartSensor.begin()) {

      Serial.println("MAX30102 not detected. Please check wiring.");

      while (1);

   }

   mpu.begin();


   // Start BLE

   BLEDevice::init("WearableHealthTracker");

   BLEServer *pServer = BLEDevice::createServer();

   BLEService *pService = pServer->createService(SERVICE_UUID);
```

```
    pService->addCharacteristic(&heartRateCharacteristic);
    pService->addCharacteristic(&stepCountCharacteristic);

    heartRateCharacteristic.addDescriptor(new BLE2902());
    stepCountCharacteristic.addDescriptor(new BLE2902());

    pService->start();
    BLEDevice::startAdvertising();

    Serial.println("Wearable Health Tracker Initialized");
}

void loop() {
    // Update heart rate sensor readings
    int heartRate;
    heartSensor.getHeartRate(&heartRate);

    // Update accelerometer for step counting (simplified logic)
    mpu.update();
    if (mpu.getAccX() > 1.2 || mpu.getAccY() > 1.2) {
        stepCount++;
    }

    // Display readings
    Serial.print("Heart Rate: ");
    Serial.print(heartRate);
```

```
    Serial.print(" BPM, Step Count: ");
    Serial.println(stepCount);

    // Notify BLE characteristics
    heartRateCharacteristic.setValue(heartRate);
    heartRateCharacteristic.notify();

    stepCountCharacteristic.setValue(stepCount);
    stepCountCharacteristic.notify();

    delay(1000);  // Adjust delay for smoother operation
}
```

Key Functional Features

1. **Heart Rate Monitoring:** Provides real-time heart rate feedback to users.
2. **Step Counting:** Tracks daily activity levels to encourage fitness goals.
3. **Bluetooth Connectivity:** Enables communication with smartphone apps for data logging and visualization.

Enhancements and Advanced Features

1. **Sleep Monitoring:** Add algorithms to monitor sleep stages using accelerometer and heart rate data.
2. **Calorie Tracking:** Estimate calories burned based on activity data.
3. **Cloud Integration:** Send data to cloud platforms for remote monitoring and analytics.
4. **User Notifications:** Add haptic feedback or LED indicators for alerts.

Benefits of Wearable IoT Devices

1. **Enhanced Awareness:** Users can better understand their health metrics.
2. **Preventive Healthcare:** Early detection of irregular patterns aids in timely interventions.
3. **Personalized Fitness:** Real-time feedback motivates and tailors exercise routines.