Fall Prevention System - Code Implementation

1. Introduction

This document provides a detailed overview of the code implementation for the Fall Prevention System using the ESP32 microcontroller. It includes necessary libraries, frameworks, development tools, and step-by-step instructions for uploading and running the code.

2. Required Libraries and Frameworks

To ensure the smooth execution of the Fall Prevention System, the following libraries and frameworks must be installed:

2.1 MicroPython Libraries

- machine – To control ESP32 hardware components (GPIO, I2C, UART, etc.)
- time – For implementing delays
- bluetooth – For enabling Bluetooth communication
- ujson – To format data in JSON format
- mpu6050 – For interfacing with the accelerometer and gyroscope
- pulse_sensor – For heart rate monitoring
- edge_impulse_classifier – To process sensor data using the AI model
- micropyGPS – For GPS location tracking

2.2 Development Tools

- **MicroPython Firmware** – Required for running Python code on ESP32
- **Thonny IDE** or **VS Code with Pymakr Plugin** – Recommended for writing and flashing MicroPython code
- **ESP-IDF (Espressif IoT Development Framework)** – Optional for C-based development
- **Arduino IDE with ESP32 Board Support** – Alternative method to flash code in C++

3. Setting Up the Development Environment

3.1 Installing MicroPython on ESP32

1. Download the latest MicroPython firmware for ESP32 from MicroPython.org.
2. Flash the firmware using esptool.py:

```
esptool.py --port COMx erase_flash

esptool.py --chip esp32 --port COMx write_flash -z 0x1000 firmware.bin
```

3. Connect ESP32 to VS Code or Thonny for programming.

3.2 Installing Required Libraries

1. Install the necessary libraries using upip (MicroPython package manager):

   ```
   import upip
   upip.install('micropython-umqtt.simple')
   upip.install('micropython-mpu6050')
   ```

2. Manually upload other required Python files (e.g., edge_impulse_classifier.py, pulse_sensor.py).

4. Uploading and Running the Code

4.1 Uploading the Code

1. Copy the following Python script into Thonny IDE or VS Code:

```
# Fall Prevention System - ESP32 Code
from machine import Pin, I2C, UART
import time
import bluetooth
import ujson
from mpu6050 import MPU6050  # Accelerometer & Gyroscope
from pulse_sensor import PulseSensor  # Heart Rate Sensor
from edge_impulse_classifier import classify_fall  # AI Model for Fall Detection
from micropyGPS import MicropyGPS  # GPS Module

# Initialize sensors
i2c = I2C(scl=Pin(22), sda=Pin(21))
mpu = MPU6050(i2c)
heart_sensor = PulseSensor(Pin(34))
```

```
gps_uart = UART(1, baudrate=9600, tx=17, rx=16)  # GPS Module
my_gps = MicropyGPS()

# Bluetooth Setup
bt = bluetooth.Bluetooth()
bt.active(True)

# Emergency Contact Numbers
CARETAKER_PHONE = "+1234567890"  # Replace with actual caretaker number
EMERGENCY_PHONE = "+9876543210"  # Replace with nearest healthcare center

# Thresholds
HEART_RATE_THRESHOLD = 40  # bpm (Too low may indicate a problem)
GYROSCOPE_THRESHOLD = 300  # degrees/sec (Sudden movement may indicate a fall)

def read_gps():
    """Reads GPS data and returns coordinates."""
    while gps_uart.any():
        my_gps.update(gps_uart.read().decode('utf-8'))
    return my_gps.latitude, my_gps.longitude



def send_bluetooth_alert(location):
    alert_msg = ujson.dumps({
        "alert": "FALL_DETECTED",
        "location": location
    })
    bt.write(alert_msg)
    print("Bluetooth alert sent: ", alert_msg)

def make_emergency_calls():
    print(f"Calling caretaker at {CARETAKER_PHONE}...")
    time.sleep(5)  # Simulate call duration
    print(f"If no response, calling emergency services at {EMERGENCY_PHONE}...")
    time.sleep(5)

# Main Loop
while True:
    accel_data = mpu.get_acceleration()
    gyro_data = mpu.get_rotation()
    heart_rate = heart_sensor.get_heart_rate()
    gps_location = read_gps()

    # Combine sensor data
    sensor_data = {
        "acceleration": accel_data,
        "rotation": gyro_data,
        "heart_rate": heart_rate
    }

    # Predict fall using AI model
    fall_detected = classify_fall(sensor_data)
    # Additional conditions for alert
    if fall_detected or heart_rate < HEART_RATE_THRESHOLD or abs(gyro_data[0]) >
GYROSCOPE_THRESHOLD:
        print("Fall or abnormal condition detected! Sending alert...")
        send_bluetooth_alert(gps_location)
        make_emergency_calls()
        time.sleep(10)  # Avoid repeated alerts
    time.sleep(1)
```

2. Save the file as main.py.
3. Upload the script to ESP32 using Thonny IDE or VS Code with the Pymakr plugin.
4. Run the script and monitor the output in the Serial Console.

5. Troubleshooting

- If the ESP32 does not respond, check the COM port settings.
- Ensure all required libraries are installed correctly.
- If Bluetooth fails, verify that the paired device is discoverable.
- If GPS coordinates are not received, check the baud rate and connection pins.

6. Conclusion

This document provides a comprehensive guide for setting up, uploading, and running the Fall Prevention System code on an ESP32. The integration of AI-powered fall detection ensures timely alerts and emergency responses, enhancing elderly safety and healthcare efficiency.