Smart Guardian: Wearable Fall Detection & Health Monitor

Source Code & System Design Documentation

Generated by Cirkit AI

December 13, 2025

Contents

1 System Architecture & OS Design **2**
- 1.1 Execution Loops . . . 2
- 1.2 State Machine: Heart Rate Monitor . . . 2

2 Hardware Configuration **3**
- 2.1 Wiring Diagram . . . 3

3 Required Libraries **3**

4 Tool 1: Hardware Diagnostic Sketch **4**

5 Tool 2: Feature Calibration (Serial Plotter) **5**

6 Production Firmware: Smart Guardian OS **6**

1 System Architecture & OS Design

The Smart Guardian firmware is designed as a **Multi-Rate Real-Time System** running on the ESP32 (TTGO T-Display). Unlike simple loop-based sketches, it employs a non-blocking architecture to ensure safety-critical features (Fall Detection) are never interrupted by slower tasks (Heart Rate measurement or WiFi transmission).

1.1 Execution Loops

The system divides tasks into three execution frequencies:

- **High-Speed Loop (Continuous):** Reads the MPU6050 accelerometer. Calculates Vector Magnitude (SVM). Detects Free Fall and Impacts immediately.
- **Medium-Speed Loop (State Machine):** Manages the MAX30102 Heart Rate sensor. It uses a "Sliding Window" buffer to process data in small chunks without blocking the main processor.
- **Low-Speed Loop (2000ms):** Manages WiFi connections and sends JSON payloads to the web server. Updates the TFT display to prevent flickering.

1.2 State Machine: Heart Rate Monitor

To prevent UI glitches and false readings, the Bio-Sensor logic follows these states:

1. **IDLE:** Sensor is in low power. Waiting for finger detection (IR Threshold $> 50,000$).
2. **MEASURING:** Finger detected. Fills a 100-sample buffer. Updates rolling average every 25 samples. Displays live animation.
3. **RESULT:** After 60 seconds of stable readings, locks the final average BPM and sends it to the server.

2 Hardware Configuration

Microcontroller: TTGO T-Display (ESP32)
Sensors: MPU6050 (Motion), MAX30102 (Pulse/SpO2)
Display: Built-in ST7789 IPS TFT

2.1 Wiring Diagram

Sensor Pin	TTGO Pin	Notes
VCC (Both)	3.3V	Do not use 5V
GND (Both)	G	Common Ground
SDA (Both)	GPIO 21	I2C Data
SCL (Both)	GPIO 22	I2C Clock
INT (MPU6050)	Not Used	Optional for wake-up

3 Required Libraries

Ensure the following libraries are installed in the Arduino IDE:

- `TFT_eSPI` by Bodmer (Configured for TTGO T-Display)
- `Adafruit MPU6050` by Adafruit
- `Adafruit Unified Sensor` by Adafruit
- `SparkFun MAX3010x Pulse and Proximity Sensor Library`
- `ArduinoJson` by Benoit Blanchon
- `HTTPClient` (Built-in ESP32)

4 Tool 1: Hardware Diagnostic Sketch

Use this code first to verify wiring. It scans I2C addresses and prints raw sensor data.

```
#include <Wire.h>
#include <Adafruit_MPU6050.h>
#include <Adafruit_Sensor.h>
#include "MAX30105.h"

#define I2C_SDA 21
#define I2C_SCL 22

Adafruit_MPU6050 mpu;
MAX30105 particleSensor;

void setup() {
  Serial.begin(115200);
  while (!Serial);
  delay(1000);

  Serial.println("\n====== SYSTEM DIAGNOSTIC START ======");
  Wire.begin(I2C_SDA, I2C_SCL);

  // I2C Scanner
  byte error, address;
  int nDevices = 0;
  for(address = 1; address < 127; address++ ) {
    Wire.beginTransmission(address);
    error = Wire.endTransmission();
    if (error == 0) {
      Serial.print("I2C found at 0x");
      if (address < 16) Serial.print("0");
      Serial.println(address, HEX);
      nDevices++;
    }
  }
  if (nDevices == 0) Serial.println("No I2C devices found. CHECK WIRING!");

  // Init Sensors
  if (!mpu.begin()) Serial.println("MPU6050 Failed!");
  else Serial.println("MPU6050 OK!");

  if (!particleSensor.begin(Wire, I2C_SPEED_FAST)) Serial.println("MAX30102 Failed!");
  else {
    Serial.println("MAX30102 OK!");
    particleSensor.setup();
  }
}

void loop() {
  sensors_event_t a, g, temp;
  mpu.getEvent(&a, &g, &temp);
  long irValue = particleSensor.getIR();

  Serial.print("Accel Z: "); Serial.print(a.acceleration.z);
  Serial.print("\t | Bio IR: "); Serial.println(irValue);
  delay(500);
}
```

5 Tool 2: Feature Calibration (Serial Plotter)

Upload this to tune thresholds. Open Tools > Serial Plotter to see live graphs of impacts and heartbeats.

```
#include <Wire.h>
#include <Adafruit_MPU6050.h>
#include <Adafruit_Sensor.h>
#include "MAX30105.h"

// Configuration
const float FALL_TRIGGER = 25.0;
const float STILL_THRESHOLD = 0.5;

Adafruit_MPU6050 mpu;
MAX30105 particleSensor;

float lastSVM = 0;
float motionVariance = 0;

void setup() {
  Serial.begin(115200);
  Wire.begin(21, 22);

  if (!mpu.begin()) while(1);
  mpu.setAccelerometerRange(MPU6050_RANGE_16_G);

  if (!particleSensor.begin(Wire, I2C_SPEED_FAST)) while(1);
  particleSensor.setup();
  particleSensor.setPulseAmplitudeRed(0x0A);
}

void loop() {
  sensors_event_t a, g, temp;
  mpu.getEvent(&a, &g, &temp);
  long irValue = particleSensor.getIR();

  // Calculate Features
  float svm = sqrt(sq(a.acceleration.x) + sq(a.acceleration.y) + sq(a.acceleration.z));
  float delta = abs(svm - lastSVM);
  motionVariance = (motionVariance * 0.9) + (delta * 0.1);
  lastSVM = svm;

  float visualizedIR = (irValue > 50000) ? (irValue - 50000) / 100.0 : 0;

  // Plotter Output
  Serial.print("Impact:"); Serial.print(svm); Serial.print(",");
  Serial.print("MotionVar:"); Serial.print(motionVariance); Serial.print(",");
  Serial.print("HeartRaw:"); Serial.print(visualizedIR); Serial.print(",");
  Serial.print("FallLimit:"); Serial.println(FALL_TRIGGER);

  delay(50);
}
```

6 Production Firmware: Smart Guardian OS

This is the final integrated code including Display, WiFi, Web Server, Fall Detection, and Heart Rate Monitoring.

```
/*
 * PROJECT: Smart Guardian - Integrated Firmware
 * HARDWARE: TTGO T-Display (ESP32), MPU6050, MAX30102
 */

#include <Wire.h>
#include <TFT_eSPI.h>
#include <SPI.h>
#include <Adafruit_MPU6050.h>
#include <Adafruit_Sensor.h>
#include <WiFi.h>
#include <HTTPClient.h>
#include <ArduinoJson.h>
#include "MAX30105.h"
#include "spo2_algorithm.h"

// ================= CONFIGURATION =================
const char *ssid = "EFG";
const char *password = "123456789";
const char *serverUrl = "http://10.175.23.246:3000/api/readings";
const char *deviceId = "TTGO-T-Display-001";

#define SDA_PIN 21
#define SCL_PIN 22

// Thresholds
const float FALL_TRIGGER = 25.0; // Impact G-force
const float STEP_THRESHOLD = 1.2;
const float SLEEP_VARIANCE = 0.05;
const unsigned long MEASURE_DURATION = 60000; // 60 Seconds

// Objects
TFT_eSPI tft = TFT_eSPI();
MAX30105 particleSensor;
Adafruit_MPU6050 mpu;

// Variables
float currentSVM = 0;
float dynamicAccel = 0;
bool fallDetected = false;
int stepCount = 0;
String sleepState = "Awake";
float motionVariance = 0;

// Heart Rate Buffer
uint32_t irBuffer[100];
uint32_t redBuffer[100];
int32_t bufferLength = 100;
int32_t spo2;
int8_t validSPO2;
int32_t heartRate;
int8_t validHeartRate;

// State Machine
enum HRState { IDLE, MEASURING, RESULT };
HRState hrState = IDLE;
unsigned long hrTimerStart = 0;
int last5BPM[5] = {0};
int bpmIndex = 0;
int finalAvgBPM = 0;
int displayBPM = 0;
int samplesRecorded = 0;
unsigned long lastWebSend = 0;

// Graphics
const unsigned char PROGMEM heart_bmp[] = {
  0x00, 0x00, 0x18, 0x18, 0x3C, 0x3C, 0x7E, 0x7E, 0xFF, 0xFF, 0xFF, 0xFF,
  0xFF, 0xFF, 0xFF, 0xFF, 0x7E, 0x7E, 0x3C, 0x3C, 0x18, 0x18, 0x08, 0x10,
```

```
  0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00
};

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

  tft.init();
  tft.setRotation(1);
  tft.fillScreen(TFT_BLACK);
  tft.setTextColor(TFT_WHITE, TFT_BLACK);
  tft.drawString("Booting System...", 10, 10, 2);

  WiFi.begin(ssid, password);
  while (WiFi.status() != WL_CONNECTED) { delay(500); }

  Wire.begin(SDA_PIN, SCL_PIN);
  mpu.begin();
  mpu.setAccelerometerRange(MPU6050_RANGE_16_G);

  particleSensor.begin(Wire, I2C_SPEED_FAST);
  byte ledBrightness = 60;
  byte sampleAverage = 4;
  byte ledMode = 2;
  int sampleRate = 100;
  int pulseWidth = 411;
  int adcRange = 4096;
  particleSensor.setup(ledBrightness, sampleAverage, ledMode, sampleRate, pulseWidth, adcRange
    );

  tft.fillScreen(TFT_BLACK);
  drawStaticUI();
}

void loop() {
  processMotion();
  processHeartRate();

  if (millis() - lastWebSend > 2000) {
    sendDataToServer();
    lastWebSend = millis();
  }
}

void processMotion() {
  sensors_event_t a, g, temp;
  mpu.getEvent(&a, &g, &temp);

  currentSVM = sqrt(sq(a.acceleration.x) + sq(a.acceleration.y) + sq(a.acceleration.z));
  dynamicAccel = currentSVM - 9.8;

  if (currentSVM > FALL_TRIGGER) {
    fallDetected = true;
    showFallAlert();
  }

  static bool stepFlag = false;
  if (dynamicAccel > STEP_THRESHOLD && !stepFlag) {
    stepCount++;
    stepFlag = true;
    updateStepUI();
  } else if (dynamicAccel < 0.5) stepFlag = false;

  static float lastVarSVM = 0;
  float delta = abs(currentSVM - lastVarSVM);
  motionVariance = (motionVariance * 0.95) + (delta * 0.05);
  lastVarSVM = currentSVM;

  if (motionVariance < SLEEP_VARIANCE) sleepState = "Sleep";
  else sleepState = "Awake";
}

void processHeartRate() {
  particleSensor.check();
```

```
  while(particleSensor.available()) {
    if(samplesRecorded >= 100) {
       for(int i=0; i<99; i++) {
         redBuffer[i] = redBuffer[i+1];
         irBuffer[i] = irBuffer[i+1];
       }
       samplesRecorded = 99;
    }
    redBuffer[samplesRecorded] = particleSensor.getFIFORed();
    irBuffer[samplesRecorded] = particleSensor.getFIFOIR();
    samplesRecorded++;
    particleSensor.nextSample();
  }

  long irValue = particleSensor.getIR();

  switch (hrState) {
    case IDLE:
      if (irValue > 50000) {
        hrState = MEASURING;
        hrTimerStart = millis();
        tft.fillRect(0, 40, 240, 90, TFT_BLACK);
        tft.drawString("Measuring...", 60, 50, 2);
        samplesRecorded = 0;
      }
      break;

    case MEASURING: {
      if (irValue < 50000) { hrState = IDLE; resetHRUI(); break; }

      if (samplesRecorded == 100 && (millis() % 250 < 20)) {
          maxim_heart_rate_and_oxygen_saturation(irBuffer, bufferLength, redBuffer, &spo2, &
    validSPO2, &heartRate, &validHeartRate);
          if (validHeartRate && heartRate > 40 && heartRate < 200) {
            displayBPM = heartRate;
            last5BPM[bpmIndex] = heartRate;
            bpmIndex = (bpmIndex + 1) % 5;
            updateLiveHR(heartRate);
          }
      }

      int progress = map(millis() - hrTimerStart, 0, MEASURE_DURATION, 0, 240);
      tft.fillRect(0, 125, progress, 5, TFT_GREEN);

      if (millis() - hrTimerStart > MEASURE_DURATION) {
        long sum = 0;
        int validCount = 0;
        for(int i=0; i<5; i++) {
          if(last5BPM[i] > 0) { sum += last5BPM[i]; validCount++; }
        }
        if (validCount > 0) finalAvgBPM = sum / validCount;
        else finalAvgBPM = displayBPM;
        hrState = RESULT;
        showFinalResult();
      }
      break;
    }

    case RESULT:
      if (irValue < 50000) { hrState = IDLE; resetHRUI(); }
      break;
  }
}

void drawStaticUI() {
  tft.setTextDatum(TC_DATUM);
  tft.setTextColor(TFT_CYAN, TFT_BLACK);
  tft.drawString("SMART GUARDIAN", 120, 5, 2);
  tft.drawLine(0, 25, 240, 25, TFT_DARKGREY);
  tft.setTextDatum(TL_DATUM);
  tft.setTextColor(TFT_SILVER, TFT_BLACK);
  tft.drawString("Steps:", 5, 110, 2);
  tft.print(stepCount);
```

```
}

void updateStepUI() {
  tft.fillRect(50, 110, 60, 20, TFT_BLACK);
  tft.setTextColor(TFT_WHITE, TFT_BLACK);
  tft.drawString(String(stepCount), 50, 110, 2);
}

void updateLiveHR(int bpm) {
  tft.fillRect(80, 40, 80, 40, TFT_BLACK);
  tft.setTextDatum(MC_DATUM);
  tft.setTextColor(TFT_GREEN, TFT_BLACK);
  tft.drawNumber(bpm, 120, 60, 6);
  tft.drawString("BPM", 120, 90, 2);
  static bool beatToggle = false;
  if(beatToggle) tft.drawBitmap(20, 50, heart_bmp, 16, 16, TFT_RED);
  else tft.fillRect(20, 50, 16, 16, TFT_BLACK);
  beatToggle = !beatToggle;
}

void showFinalResult() {
  tft.fillScreen(TFT_BLACK);
  drawStaticUI();
  tft.setTextColor(TFT_YELLOW, TFT_BLACK);
  tft.setTextDatum(MC_DATUM);
  tft.drawString("MEASUREMENT DONE", 120, 40, 2);
  tft.setTextColor(TFT_WHITE, TFT_BLACK);
  tft.drawString("AVG HR:", 120, 70, 2);
  tft.drawNumber(finalAvgBPM, 120, 100, 6);
}

void resetHRUI() {
  tft.fillScreen(TFT_BLACK);
  drawStaticUI();
  updateStepUI();
  displayBPM = 0;
  for(int i=0; i<5; i++) last5BPM[i]=0;
}

void showFallAlert() {
  tft.fillScreen(TFT_RED);
  tft.setTextColor(TFT_WHITE, TFT_RED);
  tft.setTextDatum(MC_DATUM);
  tft.drawString("FALL DETECTED!", 120, 67, 4);
  sendDataToServer();
  delay(3000);
  fallDetected = false;
  resetHRUI();
}

void sendDataToServer() {
  if (WiFi.status() == WL_CONNECTED) {
    HTTPClient http;
    http.begin(serverUrl);
    http.addHeader("Content-Type", "application/json");
    String json = "{";
    json += "\"device_id\":\"" + String(deviceId) + "\",";
    json += "\"fall_status\":" + String(fallDetected ? "true" : "false") + ",";
    json += "\"steps\":" + String(stepCount) + ",";
    int bpmToSend = (hrState == RESULT) ? finalAvgBPM : displayBPM;
    json += "\"heart_rate\":" + String(bpmToSend) + ",";
    json += "\"sleep_state\":\"" + sleepState + "\",";
    json += "\"spo2\":" + String(spo2);
    json += "}";
    int responseCode = http.POST(json);
    if (responseCode > 0) Serial.println("JSON Sent");
    else Serial.println("WiFi Fail");
    http.end();
  }
}
```