Wearable 구현 고려 사항

조사 목적 및 할일 :

- 조사 목적 : 웨어러블 방식으로 구현 시 활용 가능한 기술 검토
- 할일 : Draft 조사 자료 공유

MPU(IMU) vs ADXL(가속도계) — 낙상 감지 관점 비교

문단 시작 **요약 한 줄**: **정확도**가 최우선이면 자이로가 포함된 **MPU(예: MPU-6050/MPU-9250/ICM-20948)**가 유리하고, **초저전력·초소형 웨어러블**이면 **ADXL(특히 ADXL362)**이 유리합니다. 현장에선 “MPU(실감지) + 문/레이더/버튼 보조”가 안정적입니다.

1) 센서 스펙/특성 비교

문단 시작 아래는 대표 칩을 가정해 비교했습니다: **MPU-6050(6축: 가속도+자이로)**, **ADXL345(3축 가속도)**, **ADXL362(초저전력 3축 가속도)**.

구분	MPU-6050 (IMU)	ADXL345 (Accel)	ADXL362 (Ultra-Low-Power Accel)
축	6축(가속도+자이로)	3축(가속도)	3축(가속도)
전력	중간(≈3~5mA)	낮음(≈140~200µA)	**매우 낮음(≈1.8µA @웨이크온)**
데이터	가속도 + **회전속도**	가속도	가속도
장점	**자유낙하→충격→정지 + 회전 패턴**을 동시 검출, 방향추정(roll/pitch)	저전력·저가·풍부한 예제, 활동/충격 INT 제공	**배터리 수개월~1년대** 웨어러블, HW 활동/무활동 트리거 강력
단점	전력·보정 부담, 가끔 **자이로 드리프트**	회전정보 부재 → “앉기/눕기”와 낙상 구분이 더 어려움	고주파 충격에 민감, 세밀한 포즈 추정은 한계

낙상 적합도	**높음(정확)**	중간(튜닝 필요)	**중간~높음(저전력 시나리오)**
가격/난이도	보통/중	저/저	보통/중

2) 낙상 감지 로직의 차이

문단 시작 **공통 패턴(클래식 3단계)**

1. **자유낙하(Free-fall)**: |a|가 1g(≈9.81m/s²)에서 **급감(예: 0.2~~0.5g 이하, 100~~300ms)**
2. **충격(Impact)**: |a|가 **급증(예: 2.0~~3.0g 이상, 20~~100ms)**
3. **무동작(Inactivity)**: 이후 **10~30초 저변화** + 비정상 자세(기울기) 유지

문단 시작 **MPU(가속도+자이로) 추가 조건**

- **회전속도(gyro)** 급증(예: **>300°/s**)을 “넘어지는 동작”의 보조 증거로 사용 → 앉기/누움과 **false positive**를 더 잘 구분.
- 추정자세(roll/pitch)가 평소와 크게 달라진 상태로 유지되면 신뢰도 ↑.

문단 시작 **ADXL(가속도 단독) 로직**

- 자유낙하+충격+무동작의 **임계값/시간창** 조합으로 판정.
- 자세 유지(기울기)는 **중력 벡터 방향**으로 근사 판단(자이로가 없어 회전 히스토리 정보는 제한).

---

3) 어떤 센서가 더 적합할까?

문단 시작 **실거주 낙상 경보 정확도**가 최우선 → **MPU(IMU)** 권장.

- 이유: 회전 이벤트(돌아가며 쓰러짐)와 방향 추정이 가능해 **오경보↓**.
- 예: 욕실·거실 활동 중 급격 회전 + 충격 + 무동작 패턴을 일관되게 포착.

문단 시작 **초저전력 웨어러블/패치형(배터리 수개월~1년)** → **ADXL362** 우선.

- 이유: 하드웨어 **Wake-Up/Activity/Inactivity** 인터럽트로 MCU를 거의 잠재워 **전력 극소화**.
- 정확도는 MPU보다 낮을 수 있으나, **다중센서 융합**(문열림, mmWave 존재감지, SOS버튼)으로 보완 가능.

---

4) 추천 구성(현장 관점)

문단 시작 - **팔/허리 착용 웨어러블**: ADXL362 + BLE 비콘(초저전력), 낙상 의심 시만 게이트웨이 깨우기.

문단 시작 - **실내용 정확 감지 노드**: MPU-6050/9250 + PIR/mmWave 보조 + 스피커 질의("괜찮으세요?") + 30초 무응답시 알림.
문단 시작 - **관제**: 로컬 임계값 + 서버 룰엔진(사용자별 평상시 활동 베이스라인) 동시 적용.

---

5) 아두이노 예제 코드 (둘 다 제공)

5-1. MPU-6050 기반 낙상 감지 (가속도+자이로)

문단 시작 아래 코드는 **자유낙하 → 충격 → 무동작 + 자이로 급증**을 조합해 간단 판정합니다. (I²C, 기본 레지스터 사용)

```cpp
// Arduino C++ 예제: MPU-6050 낙상 감지 (간단 임계값 버전)
// 하드웨어: Arduino Uno, MPU-6050 (VCC=5V 가능 모듈 or 3.3V, SDA=A4, SCL=A5)
// 라이브러리 없이 최소 레지스터 접근 (실무에선 MPU6050/MPU_DMP 라이브러리 권장)

#include <Wire.h>

const uint8_t MPU_ADDR = 0x68;

// --- 튜닝 파라미터 (사용자 환경에 맞게 조정) ---
const float G = 9.80665;
const float FREEFALL_G = 0.4;      // 자유낙하 임계(|a| < 0.4g)
const float IMPACT_G   = 2.5;      // 충격 임계(|a| > 2.5g)
const float GYRO_SPIKE = 300.0;    // 회전속도 임계(°/s)
const unsigned long INACT_MS = 10000; // 무동작 시간 10초

// 상태 추적
bool freefallDetected = false;
bool impactDetected   = false;
unsigned long impactTime = 0;

void mpuWrite(uint8_t reg, uint8_t val){
  Wire.beginTransmission(MPU_ADDR);
  Wire.write(reg); Wire.write(val);
  Wire.endTransmission();
}

void readMPU(int16_t &ax,int16_t &ay,int16_t &az,int16_t &gx,int16_t &gy,int16_t &gz){
  Wire.beginTransmission(MPU_ADDR);
  Wire.write(0x3B); // ACCEL_XOUT_H
  Wire.endTransmission(false);
  Wire.requestFrom(MPU_ADDR, 14, true);
  ax = (Wire.read()<<8)|Wire.read();
  ay = (Wire.read()<<8)|Wire.read();
  az = (Wire.read()<<8)|Wire.read();
  Wire.read(); Wire.read(); // temp skip
  gx = (Wire.read()<<8)|Wire.read();
  gy = (Wire.read()<<8)|Wire.read();
  gz = (Wire.read()<<8)|Wire.read();
}

void setup(){
```

```
  Serial.begin(115200);
  Wire.begin();
  mpuWrite(0x6B, 0x00); // PWR_MGMT_1: wake up
  mpuWrite(0x1B, 0x00); // GYRO_CONFIG: ±250°/s
  mpuWrite(0x1C, 0x00); // ACCEL_CONFIG: ±2g
  delay(100);
  Serial.println("MPU6050 fall detection start");
}

float norm3(float x, float y, float z){ return sqrtf(x*x + y*y + z*z); }

bool isInactive(){
  // 간단 무동작: 최근 N 샘플의 가속도 변화가 작으면 true
  static float prevA = 0;
  static unsigned long stableSince = millis();
  int16_t ax,ay,az,gx,gy,gz;
  readMPU(ax,ay,az,gx,gy,gz);
  // 스케일 변환: 데이터시트 기준(±2g, ±250°/s)
  float aX = ax/16384.0f, aY = ay/16384.0f, aZ = az/16384.0f; // 단위 g
  float aNorm = norm3(aX,aY,aZ);

  if (fabs(aNorm - prevA) < 0.05) { // 변화 0.05g 이내면 안정으로 간주
    if (millis() - stableSince > INACT_MS) return true;
  } else {
    stableSince = millis();
  }
  prevA = aNorm;
  return false;
}

void loop(){
  int16_t ax,ay,az,gx,gy,gz;
  readMPU(ax,ay,az,gx,gy,gz);

  float aX = ax/16384.0f, aY = ay/16384.0f, aZ = az/16384.0f;      // g
  float gX = gx/131.0f,   gY = gy/131.0f,   gZ = gz/131.0f;        // °/s
  float aNorm = norm3(aX,aY,aZ);
  float gNorm = norm3(gX,gY,gZ);

  // 1) 자유낙하
  if (!freefallDetected && aNorm < FREEFALL_G) {
    freefallDetected = true;
    Serial.println("[DEBUG] Free-fall detected");
  }

  // 2) 충격 + 자이로 스파이크
  if (freefallDetected && !impactDetected && (aNorm > IMPACT_G || gNorm > GYRO_SPIKE)){
    impactDetected = true;
    impactTime = millis();
    Serial.println("[DEBUG] Impact/gyro spike detected");
  }

  // 3) 무동작 유지 → 낙상
  if (impactDetected && (millis() - impactTime > INACT_MS) && isInactive()){
    Serial.println(">>> FALL DETECTED (MPU) <<<");
    // TODO: 알림 발송 / 부저 / MQTT publish
    // 상태 초기화(원한다면 유지)
    freefallDetected = impactDetected = false;
```

```
  }

  delay(20); // 50Hz
}

```

5-2. ADXL345 기반 낙상 감지 (가속도 단독)

문단 시작 가속도만 사용하므로 **임계값/시간창** 튜닝이 중요합니다. (I²C 기본 설정)

```cpp
// Arduino C++ 예제: ADXL345 낙상 감지 (가속도 기반)
// 하드웨어: Arduino Uno, ADXL345 (VCC=3.3~5V 모듈, SDA=A4, SCL=A5)

#include <Wire.h>
const uint8_t ADXL = 0x53;

const float FREEFALL_G = 0.4;
const float IMPACT_G   = 2.8;
const unsigned long INACT_MS = 12000;

bool freefall=false, impact=false;
unsigned long impactAt=0;

void write8(uint8_t reg, uint8_t val){
  Wire.beginTransmission(ADXL); Wire.write(reg); Wire.write(val); Wire.endTransmission();
}

void readXYZ(int16_t &x,int16_t &y,int16_t &z){
  Wire.beginTransmission(ADXL); Wire.write(0x32); Wire.endTransmission(false);
  Wire.requestFrom(ADXL, (uint8_t)6);
  x = (int16_t)(Wire.read() | (Wire.read()<<8));
  y = (int16_t)(Wire.read() | (Wire.read()<<8));
  z = (int16_t)(Wire.read() | (Wire.read()<<8));
}

void setup(){
  Serial.begin(115200);
  Wire.begin();
  write8(0x2D, 0x08); // POWER_CTL: Measure
  write8(0x31, 0x08); // DATA_FORMAT: ±2g, full res
  write8(0x2C, 0x0A); // BW_RATE: ~100Hz
  Serial.println("ADXL345 fall detection start");
}

float norm3(float x,float y,float z){ return sqrtf(x*x+y*y+z*z); }

bool inactive(){
  static float prev=0; static unsigned long since=millis();
  int16_t x,y,z; readXYZ(x,y,z);
  // 256 LSB/g (full res 대략치), 보정 필요
  float ax = x/256.0f, ay = y/256.0f, az = z/256.0f;
  float n = norm3(ax,ay,az);
  if (fabs(n - prev) < 0.06) {
    if (millis() - since > INACT_MS) return true;
  } else since = millis();
  prev = n; return false;
```

```
}

void loop(){
  int16_t x,y,z; readXYZ(x,y,z);
  float ax=x/256.0f, ay=y/256.0f, az=z/256.0f;
  float aNorm = norm3(ax,ay,az);

  if (!freefall && aNorm < FREEFALL_G){
    freefall=true; Serial.println("[DEBUG] Free-fall");
  }
  if (freefall && !impact && aNorm > IMPACT_G){
    impact=true; impactAt=millis(); Serial.println("[DEBUG] Impact");
  }
  if (impact && (millis()-impactAt > INACT_MS) && inactive()){
    Serial.println(">>> FALL DETECTED (ADXL) <<<");
    freefall=impact=false;
  }
  delay(10);
}

```

6) 결론

문단 시작 **정확도·안전 우선**: **MPU(가속도+자이로) 기반 낙상 감지**가 더 적합합니다. 회전 정보를 활용해 **앉기/눕기/물건 떨어짐**과 낙상을 구분하기 쉽습니다.
문단 시작 **초저전력 웨어러블·경제성**: **ADXL362** 같은 초저전력 가속도계로 **하드웨어 인터럽트 + 서버 측 보강 규칙**을 병행하면 좋습니다.
문단 시작 최종적으로는 **센서 융합**(IMU + mmWave/PIR + 문열림 + SOS버튼 + 음성 확인)이 **현실적인 오경보 저감**과 **책임성 있는 알림**에 가장 효과적입니다.