Complementary Filter

정보 교정을 위한 소프트웨어 필터 적용에 대한 고려 사항 공유

문단 시작 딱 좋아요. **MPU-6050(6축 IMU) + 소프트웨어 필터**로 낙상을 잡아내는 방법을 이론→실무 순서로 정리하고, 바로 **아두이노 우노용 예제 코드**(보정·필터·상태머신 포함)까지 드릴게요.

---

## 1) MPU-6050 + 필터로 낙상 감지: 핵심 개념 정리

문단 시작 **MPU-6050이 주는 값**

- 3축 가속도계 **a = (ax, ay, az)** : 중력 + 동작 가속도를 함께 측정(단위 g 또는 m/s²).
- 3축 자이로 **ω = (gx, gy, gz)** : 각속도(단위 °/s).
- 문제: 가속도는 **노이즈/진동/충격에 민감**, 자이로는 **시간이 지날수록 드리프트(영점 떠다님)**가 커집니다.

문단 시작 **왜 필터가 필요한가?**

- **자세(roll, pitch)** 를 계산하려면 가속도와 자이로를 **서로 보완**해야 합니다.
  - 저주파(느린 변화)는 가속도가 강함(중력 기준)
  - 고주파(빠른 변화)는 자이로가 강함(미세한 회전 변화)
- 대표 솔루션
  a. **보완(Complementary) 필터**: 간단, 우노에서 가볍게 돌아감
  b. **칼만(Kalman) 필터**: 정확하지만 연산량 큼(우노에서도 가능하나 코드·자원 부담↑)

---

### 1-1) Complementary Filter (권장: 우노/실시간)

문단 시작 **아이디어**

- 자이로로 이전 각도를 **적분**해 이어 붙이고(빠른 변화에 강함),
- 가속도 기반 각도(atan2)로 **장기 기준**을 조금씩 보정.
- 식(roll 예):

  $\text{roll} = \alpha (\text{roll} + \omega_x \cdot dt) + (1-\alpha)\cdot \text{roll\_acc}$

  - α(알파) = 0.90~0.98(경험치). 클수록 자이로 비중↑, 작을수록 가속도 비중↑.

- roll_acc = atan2(ay, az)·(180/π) 등으로 계산.

문단 시작 **장점/단점**

- 장점: 구현 간단, CPU·메모리 가벼움, 실무에서 충분히 잘 작동
- 단점: 튜닝 α 필요, yaw(방위각)는 자기센서 없으면 정확도 제한

---

### 1-2) Kalman Filter (선택: 더 정교하게)

문단 시작 **개념 요약**

- 상태벡터 예: [θ, θ˙]⊤[ \theta, \ \dot{\theta} ]^\top (자세각·각속도)
- **예측(Predict)**: 직전 상태를 자이로로 업데이트
- **갱신(Update)**: 가속도로 얻은 자세각(측정)으로 예측을 교정
- 공분산 행렬 PP와 프로세스/측정 잡음 Q,RQ, R를 사용해 **가중을 동적으로 결정**
- 결과: 드리프트 억제 + 노이즈 억제에 탁월

문단 시작 **언제 쓰나**

- 센서 노이즈/드리프트가 심하거나, 자세 추정의 안정성이 **매우** 중요할 때
- 우노에서도 가능하나, **STM32/ESP32** 같이 여유가 있는 보드에서 추천

---

### 1-3) 낙상(Fall) 패턴의 일반 구조

문단 시작 낙상은 보통 **세 단계 시퀀스**로 나타납니다.

1. **자유낙하 구간**: 가속도 합 ∣a∣=ax2+ay2+az2|a| = \sqrt{ax^2+ay^2+az^2} 이 **0.2~0.5 g** 근처로 급감(중력 상쇄)
2. **충격(임팩트)**: ∣a∣|a|가 **2.0~3.5 g 이상**으로 급상승(바닥/벽과의 충돌)
3. **무동작 + 비정상 자세**: 이후 수 초 동안 움직임 거의 없음 + **큰 기울기(예: pitch/roll > 60°)**

문단 시작 **실무 로직(상태머신)**

- `IDLE → FREEFALL → IMPACT → STILL → ALERT`
- 시간·임계값 윈도우를 둬 오탐 방지(예: 점프/앉기/침대 던짐 등)

---

## 2) MPU-6050 (6축) 낙상 감지 — 아두이노 우노 예제

문단 시작 **하드웨어**

- 보드: Arduino Uno
- 센서: MPU-6050 (I2C: SDA=A4, SCL=A5, VCC=5V(또는 3.3V, 모듈 사양 확인), GND)

- (선택) 부저/LED: 경보 표시

문단 시작 **기능**

- 초기 오프셋 보정(정지 상태에서 N회 평균)
- Complementary Filter로 roll/pitch 계산
- 낙상 상태머신: 자유낙하→충격→무동작(+큰 기울기)
- 시리얼 로그 + LED/부저 알림

```
// ------------------------------------------
// MPU-6050 낙상 감지 데모 (Arduino Uno)
// - Complementary Filter + 상태머신
// - 작성자 주석: 한국어 상세 설명 포함
// ------------------------------------------

#include <Wire.h>

// ====== 핀 설정 (필요 시 변경) ======
const int PIN_LED  = 8;   // 낙상 알림 LED
const int PIN_BUZZ = 9;   // 낙상 알림 부저

// ====== MPU6050 레지스터/주소 ======
const uint8_t MPU_ADDR = 0x68; // AD0=GND 기본 주소 0x68

// 레지스터 (필수만 사용)
const uint8_t REG_PWR_MGMT_1 = 0x6B;
const uint8_t REG_ACCEL_XOUT_H = 0x3B; // 가속도 X 상위 바이트부터 연속 읽기

// ====== 보정/스케일 ======
float accOffsetX=0, accOffsetY=0, accOffsetZ=0;
float gyroOffsetX=0, gyroOffsetY=0, gyroOffsetZ=0;

// MPU6050 기본 감도 (±2g, ±250°/s 기준)
const float ACC_SENS = 16384.0; // LSB/g
const float GYR_SENS = 131.0;   // LSB/(°/s)

// ====== 필터/상태 ======
float roll=0.0f, pitch=0.0f;     // 보완필터 결과 각도(°)
unsigned long lastMicros = 0;
const float alpha = 0.96f;       // 보완필터 비율(0.90~0.98 사이 튜닝)

// ====== 낙상 감지 파라미터 (현장 튜닝 권장) ======
const float FREEFALL_G_THRESH   = 0.35f; // g (|a|이 이 값 이하이면 자유낙하로 간주)
const uint16_t FREEFALL_MIN_MS  = 120;   // 자유낙하 최소 지속 시간

const float IMPACT_G_THRESH     = 2.5f;  // g (충격 임계)
const uint16_t IMPACT_MAX_MS    = 800;   // 자유낙하 후 이 시간 내에 충격 발생해야 함

const float STILL_G_VAR_THRESH  = 0.08f; // g 변동(표준편차 근사) 임계값
const uint16_t STILL_MIN_MS     = 3000;  // 충격 이후 이 시간 이상 정지하면 낙상 후보

const float TILT_DEG_THRESH     = 60.0f; // 충격 이후 자세 기울기(roll/pitch) 임계

// ====== 상태머신 ======
enum FallState { IDLE, FREEFALL, IMPACT, STILL, ALERT };
```

```
FallState state = IDLE;
unsigned long tEnterFreefall=0, tImpact=0, tStillStart=0;

// 최근 |a| 샘플로 정지 판단
const int WIN = 25;
float aMagWindow[WIN];
int winIdx = 0;
bool windowFilled = false;

// ====== 유틸 ======
int16_t read16(int reg) {
  Wire.beginTransmission(MPU_ADDR);
  Wire.write(reg);
  Wire.endTransmission(false);
  Wire.requestFrom(MPU_ADDR, 2, true);
  int16_t val = (Wire.read() << 8) | Wire.read();
  return val;
}

void readMPU(float& ax, float& ay, float& az, float& gx, float& gy, float& gz) {
  Wire.beginTransmission(MPU_ADDR);
  Wire.write(REG_ACCEL_XOUT_H);
  Wire.endTransmission(false);
  Wire.requestFrom(MPU_ADDR, 14, true); // ACC(6)+TEMP(2)+GYRO(6)

  int16_t rawAX = (Wire.read() << 8) | Wire.read();
  int16_t rawAY = (Wire.read() << 8) | Wire.read();
  int16_t rawAZ = (Wire.read() << 8) | Wire.read();
  Wire.read(); Wire.read(); // TEMP skip
  int16_t rawGX = (Wire.read() << 8) | Wire.read();
  int16_t rawGY = (Wire.read() << 8) | Wire.read();
  int16_t rawGZ = (Wire.read() << 8) | Wire.read();

  // 스케일링 (g, °/s) + 오프셋 제거
  ax = (rawAX / ACC_SENS) - accOffsetX;
  ay = (rawAY / ACC_SENS) - accOffsetY;
  az = (rawAZ / ACC_SENS) - accOffsetZ;

  gx = (rawGX / GYR_SENS) - gyroOffsetX;
  gy = (rawGY / GYR_SENS) - gyroOffsetY;
  gz = (rawGZ / GYR_SENS) - gyroOffsetZ;
}

void calibrate(int samples=500) {
  // 초기 정지 상태에서 평균값을 오프셋으로 사용
  long axSum=0, aySum=0, azSum=0;
  long gxSum=0, gySum=0, gzSum=0;

  for(int i=0;i<samples;i++){
    int16_t rawAX = read16(REG_ACCEL_XOUT_H);
    int16_t rawAY = read16(REG_ACCEL_XOUT_H+2);
    int16_t rawAZ = read16(REG_ACCEL_XOUT_H+4);
    int16_t rawGX = read16(REG_ACCEL_XOUT_H+8+2);
    int16_t rawGY = read16(REG_ACCEL_XOUT_H+10+2);
    int16_t rawGZ = read16(REG_ACCEL_XOUT_H+12+2);

    axSum += rawAX; aySum += rawAY; azSum += rawAZ;
    gxSum += rawGX; gySum += rawGY; gzSum += rawGZ;
```

```
    delay(3);
  }

  float axOf = (axSum/(float)samples) / ACC_SENS;
  float ayOf = (aySum/(float)samples) / ACC_SENS;
  float azOf = (azSum/(float)samples) / ACC_SENS;

  // 정지 상태에서 az는 +1g(또는 -1g) 근처여야 함 → 이를 기준으로 보정
  accOffsetX = axOf;
  accOffsetY = ayOf;
  accOffsetZ = azOf - 1.0f; // 중력 1g 제거

  gyroOffsetX = (gxSum/(float)samples) / GYR_SENS;
  gyroOffsetY = (gySum/(float)samples) / GYR_SENS;
  gyroOffsetZ = (gzSum/(float)samples) / GYR_SENS;
}

// |a| 윈도우 표준편차 근사(평균 절대편차 사용: 가벼운 연산)
float windowMAD() {
  int n = windowFilled ? WIN : winIdx;
  if(n<=1) return 0.0f;
  float mean=0;
  for(int i=0;i<n;i++) mean += aMagWindow[i];
  mean /= n;
  float mad=0;
  for(int i=0;i<n;i++) mad += fabs(aMagWindow[i]-mean);
  return mad/n;
}

void pushAMag(float a) {
  aMagWindow[winIdx++] = a;
  if(winIdx>=WIN){ winIdx=0; windowFilled=true; }
}

void setAlert(bool on){
  digitalWrite(PIN_LED, on?HIGH:LOW);
  if(on){
    tone(PIN_BUZZ, 2000, 400); // 2kHz 0.4s 경보
  }
}

// ====== SETUP ======
void setup() {
  pinMode(PIN_LED, OUTPUT);
  pinMode(PIN_BUZZ, OUTPUT);
  Wire.begin();
  Serial.begin(115200);

  // MPU 깨우기
  Wire.beginTransmission(MPU_ADDR);
  Wire.write(REG_PWR_MGMT_1);
  Wire.write(0x00); // sleep 해제
  Wire.endTransmission();

  delay(100);
  Serial.println(F("Calibrating... 센서를 정지 상태로 두세요"));
  calibrate(400);
  Serial.println(F("Done"));
```

```
163
164   lastMicros = micros();
165 }
166
167 // ====== LOOP ======
168 void loop() {
169   // dt 계산(초)
170   unsigned long now = micros();
171   float dt = (now - lastMicros) / 1e6f;
172   if(dt <= 0) dt = 1e-3f;
173   lastMicros = now;
174
175   // 센서 읽기
176   float ax, ay, az, gx, gy, gz;
177   readMPU(ax, ay, az, gx, gy, gz);
178
179   // 가속도 기반 각도 계산(중력 기준)
180   float rollAcc  = atan2(ay, az) * 180.0f / PI;               // X축 회전
181   float pitchAcc = atan2(-ax, sqrt(ay*ay + az*az)) * 180.0f/PI; // Y축 회전
182
183   // 보완 필터(자이로 적분 + 가속도 보정)
184   roll  = alpha*(roll  + gx*dt) + (1.0f-alpha)*rollAcc;
185   pitch = alpha*(pitch + gy*dt) + (1.0f-alpha)*pitchAcc;
186
187   // 가속도 크기(중력 포함) 계산 (단위: g)
188   float aMag = sqrt(ax*ax + ay*ay + az*az);
189   pushAMag(aMag);
190
191   // ---- 상태머신 ----
192   unsigned long t = millis();
193   switch(state){
194     case IDLE:
195       if(aMag < FREEFALL_G_THRESH){
196         state = FREEFALL;
197         tEnterFreefall = t;
198       }
199       break;
200
201     case FREEFALL:
202       if(aMag >= FREEFALL_G_THRESH){
203         // 자유낙하 지속시간 체크
204         if(t - tEnterFreefall >= FREEFALL_MIN_MS){
205           // 자유낙하 뒤 충격을 기다림
206           state = IMPACT;
207           tImpact = t; // 시작시간(타임아웃 계산용)
208         }else{
209           // 자유낙하 너무 짧았으면 취소
210           state = IDLE;
211         }
212       }else{
213         // 계속 자유낙하 상태 → 타임아웃 방지
214         if(t - tEnterFreefall > 1500) state = IDLE;
215       }
216       break;
217
218     case IMPACT:
219       if(aMag > IMPACT_G_THRESH){
220         // 충격 감지 → 정지 관찰 단계
```

```
        state = STILL;
        tStillStart = t;
      }else if(t - tImpact > IMPACT_MAX_MS){
        // 시간 내 충격 없으면 리셋
        state = IDLE;
      }
      break;

    case STILL: {
      // 정지 여부(가속도 변동 매우 작음) + 큰 기울기 확인
      float mad = windowMAD();
      bool lowMotion = (mad < STILL_G_VAR_THRESH);
      bool bigTilt   = (fabs(roll) > TILT_DEG_THRESH) || (fabs(pitch) > TILT_DEG_THRESH);

      if(lowMotion && (t - tStillStart > STILL_MIN_MS) && bigTilt){
        state = ALERT;
        setAlert(true);
        Serial.println(F("[ALERT] FALL DETECTED!"));
      }

      // 움직임이 다시 커지면 취소
      if(!lowMotion && (t - tStillStart > 500)){
        state = IDLE;
      }
      break;
    }

    case ALERT:
      // 여기서는 사람이 버튼을 누르거나, 일정시간 후 자동 해제 로직 추가 가능
      // 데모: 10초 후 자동 해제
      if((t - tStillStart) > 10000){
        setAlert(false);
        state = IDLE;
      }
      break;
  }

  // ---- 디버그 출력(필요 시 주석 처리) ----
  static unsigned long tLog=0;
  if(millis()-tLog > 200){
    tLog = millis();
    Serial.print("a|g: "); Serial.print(aMag,3); Serial.print(" g");
    Serial.print("  roll: "); Serial.print(roll,1);
    Serial.print("  pitch: "); Serial.print(pitch,1);
    Serial.print("  state: ");
    switch(state){
      case IDLE: Serial.println("IDLE"); break;
      case FREEFALL: Serial.println("FREEFALL"); break;
      case IMPACT: Serial.println("IMPACT"); break;
      case STILL: Serial.println("STILL"); break;
      case ALERT: Serial.println("ALERT"); break;
    }
  }
}

```

### 3) 튜닝·운영 팁

문단 시작 **임계값 보정**

- `FREEFALL_G_THRESH` : 0.3~0.5 g 사이 시도
- `IMPACT_G_THRESH` : 바닥/카펫 여부에 따라 2.0~4.0 g
- `STILL_G_VAR_THRESH` : 0.05~0.12 g 범위에서 오탐/미탐 줄이기
- `TILT_DEG_THRESH` : 50~70° (체형·착용부위에 따라 조정)

문단 시작 **착용/설치 위치**

- 손목·허리·가슴 스트랩 → 신체중심에 가까울수록 패턴 인식이 안정
- 실내용 데모는 **침대 프레임/허리벨트/가슴 스트랩** 등 고정

문단 시작 **오탐 방지**

- 상태머신에 **시간 윈도우**를 반드시 둘 것(이미 코드 반영)
- 충격 후 **음성 확인("괜찮으신가요?") + 버튼/음성 해제**를 UX로 결합
- 야간 화장실 이동 등 정상 패턴은 **이동 후 활동 지속**으로 STILL 단계에서 탈출

문단 시작 **필터 선택**

- 우노/간단 데모: **보완 필터**
- 고정밀·다중센서(자기센서 포함)·확장성: **칼만 필터** 또는 **Madgwick/Mahony** AHRS(ESP32 추천)

---

### 4) 보너스: Kalman 필터 관점의 로직 개요(의사코드)

문단 시작 (roll 축만 예시)

```
상태 x = [roll, bias]^T
입력 u = gyro_rate
측정 z = roll_acc

# 예측
roll_pred = roll_prev + (gyro_rate - bias_prev)*dt
bias_pred = bias_prev
P_pred = A*P_prev*A^T + Q

# 측정 갱신
K = P_pred*H^T * (H*P_pred*H^T + R)^-1
x = x_pred + K*(z - H*x_pred)
P = (I - K*H)*P_pred

```

문단 시작 여기서 `Q, R` (잡음 공분산)을 현장 노이즈에 맞춰 튜닝하면 **드리프트 억제 + 노이즈 완화**가 동적으로 최적화됩니다.

문단 시작 필요하시면 **부저 대신 BLE 푸시/LoRa 알림**, **데이터 로깅(CSV)**, **낙상 후 보호자 앱 호출(FCM)** 까지 붙인 버전도 바로 확장해 드릴게요.