*※보고서 제목 양식 예 : 이름\_20240403*

|  |
| --- |
| **2024 ALTIS SW 개인 활동 보고서** |

|  |  |
| --- | --- |
| **활동 개요** | 가속도 자이로 값으로 각도 추정하는 방법 |
| **일시** | 2024년 04월 13일 03시 |
| **작성자** | 양주호 |
| **활동 내용** | 1. MPU-6050를 이용한 각도추정  ○ 가속도센서  - 가속도센서는 3축의 중력가속도를 포함하여 총 가속도를 측정한다. 중력가속도는 일정한 크기와 방향을 가지기 때문에 센서가 각 축에 수직 방향일 때 중력 가속도 성분이 최대가 되고, 물체가 수직에서 벗어나면 중력 가속도의 성분이 변화한다. 이러한 물체의 움직임에 따른 중력 가속도 변화량을 측정하여 각도를 추정한다. 가속도센서를 이용해 동적 물체에 경우 가속도와 물체의 중력이 함께 측정되므로 정확한 추정이 어렵기 때문에 정적인 상태일수록 정확한 각도의 데이터를 추정할 수 있음.  - 코드 (롤,피치만 구하는 예제) #include<Wire.h>  const int MPU\_ADDR = 0x68; // I2C통신을 위한 MPU6050의 주소  int16\_t AcX, AcY, AcZ, Tmp, GyX, GyY, GyZ; // 가속도(Acceleration)와 자이로(Gyro)  double angleAcX;  double angleAcY;  const double RADIAN\_TO\_DEGREE = 180 / 3.14159;  void setup() {  initSensor();  Serial.begin(9600);  delay(200);  }  void loop() {  getAngleXY();  Serial.print("Angle x : ");  Serial.print(angleAcX);  Serial.print("\t\t Angle y : ");  Serial.println(angleAcY);  delay(20);  }  double getAngleXY() {  getData();  // 삼각함수를 이용한 롤(Roll)의 각도 구하기  angleAcX = atan(AcY / sqrt(pow(AcX, 2) + pow(AcZ, 2)));  angleAcX \*= RADIAN\_TO\_DEGREE;  // 삼각함수를 이용한 피치(Pitch)의 각도 구하기  angleAcY = atan(-AcX / sqrt(pow(AcY, 2) + pow(AcZ, 2)));  angleAcY \*= RADIAN\_TO\_DEGREE;  }  void initSensor() {  Wire.begin();  Wire.beginTransmission(MPU\_ADDR); // I2C 통신용 어드레스(주소)  Wire.write(0x6B); // MPU6050과 통신을 시작하기 위해서는 0x6B번지에  Wire.write(0);  Wire.endTransmission(true);  }  void getData() {  Wire.beginTransmission(MPU\_ADDR);  Wire.write(0x3B); // AcX 레지스터 위치(주소)를 지칭합니다  Wire.endTransmission(false);  Wire.requestFrom(MPU\_ADDR, 14, true); // AcX 주소 이후의 14byte의 데이터를 요청  AcX = Wire.read() << 8 | Wire.read(); //두 개의 나뉘어진 바이트를 하나로 이어 붙여서 각 변수에 저장  AcY = Wire.read() << 8 | Wire.read();  AcZ = Wire.read() << 8 | Wire.read();  Tmp = Wire.read() << 8 | Wire.read();  GyX = Wire.read() << 8 | Wire.read();  GyY = Wire.read() << 8 | Wire.read();  GyZ = Wire.read() << 8 | Wire.read();  }  ○ 자이로센서  - 자이로센서는 3축의 각속도를 이용하여 각도를 추정한다. 자이로센서로 각 축의 각속도를 측정하고, 측정한 각속도는 시간에 대한 데이터이기 때문에 시간에 대해 적분하면 각 축의 회전 각도 변화량이 나온다. 3축의 각속도를 적분한 값을 종합하여 현재 물체의 각도를 추정한다. 다만 시간에 대해 적분을 하기 때문에 각도 데이터를 쌓는 과정에서 오차가 발생할 때 그 다음 각도 계산 과정에서 그 오차를 포함하여 계산하기 때문에 오차가 누적될 수 있다.  - 코드  #include<Wire.h>  const int MPU\_ADDR = 0x68; // I2C통신을 위한 MPU6050의 주소  int16\_t AcX, AcY, AcZ, Tmp, GyX, GyY, GyZ; // 가속도(Acceleration)와 자이로(Gyro)  double angleAcX, angleAcY, angleAcZ;  double angleGyX, angleGyY, angleGyZ;  const double RADIAN\_TO\_DEGREE = 180 / 3.14159;  const double DEG\_PER\_SEC = 32767 / 250; // 1초에 회전하는 각도  // GyX, GyY, GyZ 값의 범위 : -32768 ~ +32767 (16비트 정수범위)  unsigned long now = 0; // 현재 시간 저장용 변수  unsigned long past = 0; // 이전 시간 저장용 변수  double dt = 0; // 한 사이클 동안 걸린 시간 변수  double averAcX, averAcY, averAcZ;  double averGyX, averGyY, averGyZ;  void setup() {  initSensor();  Serial.begin(115200);  caliSensor(); // 초기 센서 캘리브레이션 함수 호출  past = millis(); // past에 현재 시간 저장  }  void loop() {  getData();  getDT();  angleGyX += ((GyX - averGyX) / DEG\_PER\_SEC) \* dt;  angleGyY += ((GyY - averGyY) / DEG\_PER\_SEC) \* dt;  angleGyZ += ((GyZ - averGyZ) / DEG\_PER\_SEC) \* dt;    Serial.print("Angle Gyro X:");  Serial.print(angleGyX);  Serial.print("\t\t Angle Gyro y:");  Serial.print(angleGyY);  Serial.print("\t\t Angle Gyro Z:");  Serial.println(angleGyZ);  delay(20);  }  void initSensor() {  Wire.begin();  Wire.beginTransmission(MPU\_ADDR); // I2C 통신용 어드레스(주소)  Wire.write(0x6B); // MPU6050과 통신을 시작하기 위해서는 0x6B번지에  Wire.write(0);  Wire.endTransmission(true);  }  void getData() {  Wire.beginTransmission(MPU\_ADDR);  Wire.write(0x3B); // AcX 레지스터 위치(주소)를 지칭합니다  Wire.endTransmission(false);  Wire.requestFrom(MPU\_ADDR, 14, true); // AcX 주소 이후의 14byte의 데이터를 요청  AcX = Wire.read() << 8 | Wire.read(); //두 개의 나뉘어진 바이트를 하나로 이어 붙여서 각 변수에 저장  AcY = Wire.read() << 8 | Wire.read();  AcZ = Wire.read() << 8 | Wire.read();  Tmp = Wire.read() << 8 | Wire.read();  GyX = Wire.read() << 8 | Wire.read();  GyY = Wire.read() << 8 | Wire.read();  GyZ = Wire.read() << 8 | Wire.read();  }  // loop 한 사이클동안 걸리는 시간을 알기위한 함수  void getDT() {  now = millis();  dt = (now - past) / 1000.0;  past = now;  }  // 센서의 초기값을 10회 정도 평균값으로 구하여 저장하는 함수  void caliSensor() {  double sumAcX = 0 , sumAcY = 0, sumAcZ = 0;  double sumGyX = 0 , sumGyY = 0, sumGyZ = 0;  getData();  for (int i=0;i<10;i++) {  getData();  sumAcX+=AcX; sumAcY+=AcY; sumAcZ+=AcZ;  sumGyX+=GyX; sumGyY+=GyY; sumGyZ+=GyZ;  delay(50);  }  averAcX=sumAcX/10; averAcY=sumAcY/10; averAcZ=sumAcY/10;  averGyX=sumGyX/10; averGyY=sumGyY/10; averGyZ=sumGyZ/10;  }  ○ 필터링 (상보필터)  - 가속도센서는 센서 특성상 고주파 영역에서 노이즈가 많이 발생하고 자이로센서는 저주파 영역에 대한 데이터의 정확도가 낮다. 상보필터는 가속도센서의 저주파 영역, 자이로센서의 고주파 영역에서의 장점을 활용하는 필터이고 각 센서로부터 얻은 데이터에 가중치를 적용해 보정하는 원리이다. - 코드  #include<Wire.h>  const int MPU\_ADDR = 0x68; // I2C통신을 위한 MPU6050의 주소  int16\_t AcX, AcY, AcZ, Tmp, GyX, GyY, GyZ; // 가속도(Acceleration)와 자이로(Gyro)  double angleAcX, angleAcY, angleAcZ;  double angleGyX, angleGyY, angleGyZ;  double angleFiX, angleFiY, angleFiZ;  const double RADIAN\_TO\_DEGREE = 180 / 3.14159;  const double DEG\_PER\_SEC = 32767 / 250; // 1초에 회전하는 각도  const double ALPHA = 1 / (1 + 0.04);  // GyX, GyY, GyZ 값의 범위 : -32768 ~ +32767 (16비트 정수범위)  unsigned long now = 0; // 현재 시간 저장용 변수  unsigned long past = 0; // 이전 시간 저장용 변수  double dt = 0; // 한 사이클 동안 걸린 시간 변수  double averAcX, averAcY, averAcZ;  double averGyX, averGyY, averGyZ;  void setup() {  initSensor();  Serial.begin(115200);  caliSensor(); // 초기 센서 캘리브레이션 함수 호출  past = millis(); // past에 현재 시간 저장  }  void loop() {  getData();  getDT();  angleAcX = atan(AcY / sqrt(pow(AcX, 2) + pow(AcZ, 2)));  angleAcX \*= RADIAN\_TO\_DEGREE;  angleAcY = atan(-AcX / sqrt(pow(AcY, 2) + pow(AcZ, 2)));  angleAcY \*= RADIAN\_TO\_DEGREE;  // 가속도 센서로는 Z축 회전각 계산 불가함.    // 가속도 현재 값에서 초기평균값을 빼서 센서값에 대한 보정  angleGyX += ((GyX - averGyX) / DEG\_PER\_SEC) \* dt; //각속도로 변환  angleGyY += ((GyY - averGyY) / DEG\_PER\_SEC) \* dt;  angleGyZ += ((GyZ - averGyZ) / DEG\_PER\_SEC) \* dt;  // 상보필터 처리를 위한 임시각도 저장  double angleTmpX = angleFiX + angleGyX \* dt;  double angleTmpY = angleFiY + angleGyY \* dt;  double angleTmpZ = angleFiZ + angleGyZ \* dt;  // (상보필터 값 처리) 임시 각도에 0.96가속도 센서로 얻어진 각도 0.04의 비중을 두어 현재 각도를 구함.  angleFiX = ALPHA \* angleTmpX + (1.0 - ALPHA) \* angleAcX;  angleFiY = ALPHA \* angleTmpY + (1.0 - ALPHA) \* angleAcY;  angleFiZ = angleGyZ; // Z축은 자이로 센서만을 이용하열 구함.  Serial.print("AngleAcX:");  Serial.print(angleAcX);  Serial.print("\t FilteredX:");  Serial.print(angleFiX);  Serial.print("\t AngleAcY:");  Serial.print(angleAcY);  Serial.print("\t FilteredY:");  Serial.println(angleFiY);  Serial.print("\t AngleGyZ:");  Serial.print(angleGyZ);  Serial.print("\t FilteredZ:");  Serial.println(angleFiZ);  // Serial.print("Angle Gyro X:");  // Serial.print(angleGyX);  // Serial.print("\t\t Angle Gyro y:");  // Serial.print(angleGyY);  // Serial.print("\t\t Angle Gyro Z:");  // Serial.println(angleGyZ);  // delay(20);  }  void initSensor() {  Wire.begin();  Wire.beginTransmission(MPU\_ADDR); // I2C 통신용 어드레스(주소)  Wire.write(0x6B); // MPU6050과 통신을 시작하기 위해서는 0x6B번지에  Wire.write(0);  Wire.endTransmission(true);  }  void getData() {  Wire.beginTransmission(MPU\_ADDR);  Wire.write(0x3B); // AcX 레지스터 위치(주소)를 지칭합니다  Wire.endTransmission(false);  Wire.requestFrom(MPU\_ADDR, 14, true); // AcX 주소 이후의 14byte의 데이터를 요청  AcX = Wire.read() << 8 | Wire.read(); //두 개의 나뉘어진 바이트를 하나로 이어 붙여서 각 변수에 저장  AcY = Wire.read() << 8 | Wire.read();  AcZ = Wire.read() << 8 | Wire.read();  Tmp = Wire.read() << 8 | Wire.read();  GyX = Wire.read() << 8 | Wire.read();  GyY = Wire.read() << 8 | Wire.read();  GyZ = Wire.read() << 8 | Wire.read();  }  // loop 한 사이클동안 걸리는 시간을 알기위한 함수  void getDT() {  now = millis();  dt = (now - past) / 1000.0;  past = now;  }  // 센서의 초기값을 10회 정도 평균값으로 구하여 저장하는 함수  void caliSensor() {  double sumAcX = 0 , sumAcY = 0, sumAcZ = 0;  double sumGyX = 0 , sumGyY = 0, sumGyZ = 0;  getData();  for (int i=0;i<10;i++) {  getData();  sumAcX+=AcX; sumAcY+=AcY; sumAcZ+=AcZ;  sumGyX+=GyX; sumGyY+=GyY; sumGyZ+=GyZ;  delay(50);  }  averAcX=sumAcX/10; averAcY=sumAcY/10; averAcZ=sumAcY/10;  averGyX=sumGyX/10; averGyY=sumGyY/10; averGyZ=sumGyZ/10;  } |
| **활동 사진** | 1. 롤, 피치의 각도를 구하기 위한 삼각함수 수식      1. 가속도 센서를 이용한 각도 계산 결과      1. 자이로 센서를 이용한 각도 계산 결과      1. 상보 필터를 적용한 각도 계산 결과 |
| **활동 결과** | (활동 사진에 첨부한 계산 결과 참조) |
| **참고 문헌** | [【 아두이노 완공 #29 】 MPU6050 자이로 가속도 센서 , 바로 이 영상으로 시작하세요~! (youtube.com)](https://www.youtube.com/watch?v=BTpwAFfeg0I) |

2024년 4월 13일

작성자 : 양주호 (인)