*※보고서 제목 양식 예 : 이름\_20240403*

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

|  |  |
| --- | --- |
| **활동 개요** | 가속도 자이로 값으로 각도 추정하는 방법 |
| **일시** | 2024년 5월 5일 20시 |
| **작성자** | 문시경 |
| **활동 내용** | 가속도 자이로 값을 활용하여 각도를 추정하는 방법에 대하여 알아보고 사용가능한 방법을 찾아 실험해본다.  우선 사용할 수 있는 방법들을 조사해 보았고, 다음의 3가지 방법을 찾아내었다.  중력 가속도 이용 : 가속도계를 이용해 각도를 추정하는 방법은 중력 가속도를 활용하여 장치의 기울기를 판단하는 원리를 기반으로 한다. 각 축에 대한 가속도 데이터를 이용하여 롤 각(앞뒤 기울기)과 피치 각(좌우 기울기)을 계산할 수 있으며, 이를 위해 atan2 함수를 사용하여 전체 360도 범위의 각도를 얻을 수 있다. 그러나 가속도계를 사용한 각도 추정은 외부 가속도의 영향을 받을 수 있고, 장기적인 사용에서는 정밀도에 한계가 있기 때문에, 자이로스코프와 같은 다른 센서와 결합하여 사용함으로써 보다 정확하고 신뢰할 수 있는 데이터를 제공할 수 있다.  상보 필터 : 가속도와 자이로의 데이터를 결합하여 보다 정확한 각도 추정 값을 얻는 데 사용된다. 자이로스코프는 높은 샘플링 레이트로 인해 짧은 시간 동안은 매우 정확한 각속도를 제공하지만, 시간이 지남에 따라 통합 오류(드리프트)가 축적된다. 반면, 가속도계는 중력 벡터를 이용하여 절대적인 각도(기울기)를 측정할 수 있지만, 진동이나 외부 가속도의 영향을 받을 수 있다. 상보필터는 자이로스코프의 단기 정밀도와 가속도계의 장기 안정성을 결합하여 이 두 센서의 단점을 보완한다.  칼만 필터 : 선형 동적 시스템에서의 상태 추정 문제를 해결하기 위한 효율적인 재귀적 필터로, 시스템의 현재 상태와 측정한 데이터에서 오는 노이즈를 고려하여, 과거의 추정값들을 사용해서 현재 상태를 보다 정확하게 추정할 수 있다. 예측 단계와 업데이트 단계를 반복하며, 자이로스코프와 가속도계 데이터를 통합해 각도를 추정한다. 칼만 필터의 효율성은 프로세스 노이즈와 측정 노이즈 파라미터에 의해 크게 영향을 받는다. 각도 추정에서 이 필터는 자이로스코프의 빠른 응답성을 가속도계의 장기적 안정성과 결합하여 정확한 결과를 제공한다.  선택 이유 : 칼만 필터는 시스템의 상태를 추정할 때 과거와 현재의 측정값을 결합하여 보다 정확한 추정이 가능하며, 측정 오차나 외부 요인에도 덜 민감하여 변수의 발생에도 유용한 대처가 가능하다. 또한 실시간으로 데이터를 처리하고 필터링할 수 있어 위성이나 로켓에 사용한다고 가정할 때 매우 유용할 것이라 판단하여 선택하게 되었다.  이제 칼만 필터가 사용되는 과정을 생각하여 코드를 만든다. 칼만 필터에는 2가지 단계가 필요하다. 이전 시간에 추정된 상태에 따라 사용자 입력이 들어왔을 경우 예상되는 상태를 계산하는 '예측' 단계, 앞의 예측 상태와 실제 측정 상태를 토대로 정확한 상태를 계산하는 '보정' 단게이다. 칼만 필터에서는 이 예측과 보정을 계속하여 반복한다.    이것은 예측단계를 나타내는 수식이고,    이것은 보정단계를 나타내는 수식이다.  ^을 쓰고 있는 변수들은 정확한 값이 아닌 예측값을 의미한다. 아랫첨자로 쓰인 k등은 시점을 의미하며 여기서 k는 현시점, k-1은 이전 시점을 의미한다. 윗첨자로 쓰인 -는 이전 값을 의미한다. 따라서 는 현시점을 기준으로 예측된 이전값을 의미한다.  A, B, H는 상태를 변환시키기 위한 state transition model 이다. 여기서는 아래와 같습니다.        아래는 시시각각 변하는 변수들이다.  x는 각도값이다. 필터의 결과로 출력되는 변수이기도 하다.  u는 실측 자이로값이다. dt를 곱해 각도로 변환시켜 이용한다.  K는 칼만이득이다. 각도에 대한 값과 자이로에 대한 값이 있다.  P는 error covariance이다. 2x2의 배열을 가지고 있다.  Q는 process noise covariance이다. 각도에 대한 값과 자이로에 대한 값이 있다.  R은 measurement covariance로 잡음에 해당한다.  z는 실측 가속도 값이다. 예상치와 비교하여 공분산을 업데이트하는 데 사용된다.  I는 2x2의 단위행렬이다.  Q, R은 파라미터값으로 사용자가 지정해주며 여기서는 0.001, 0.003, 0.03으로 설정해 주었다.  사용코드 :  #include<Wire.h>  #definempu\_add0x68//mpu6050 address  classkalman {  public :  doublegetkalman(doubleacc, doublegyro, doubledt){  //project the state ahead  angle += dt \* (gyro - bias);  //Project the error covariance ahead  P[0][0] += dt \* (dt \* P[1][1] - P[0][1] - P[1][0] + Q\_angle);  P[0][1] -= dt \* P[1][1] ;  P[1][0] -= dt \* P[1][1] ;  P[1][1] += Q\_gyro \* dt ;  //Compute the Kalman gain  doubleS = P[0][0] + R\_measure ;  K[0] = P[0][0] / S ;  K[1] = P[1][0] / S ;  //Update estimate with measurement z  doubley = acc - angle ;  angle += K[0] \* y ;  bias += K[1] \* y ;  //Update the error covariance  doubleP\_temp[2] = {P[0][0], P[0][1]};  P[0][0] -= K[0] \* P\_temp[0] ;  P[0][1] -= K[0] \* P\_temp[1] ;  P[1][0] -= K[1] \* P\_temp[0] ;  P[1][1] -= K[1] \* P\_temp[1] ;  returnangle ;  };  voidinit(doubleangle, doublegyro, doublemeasure){  Q\_angle = angle ;  Q\_gyro = gyro ;  R\_measure = measure ;  angle = 0;  bias = 0;  P[0][0] = 0;  P[0][1] = 0;  P[1][0] = 0;  P[1][1] = 0;  };  doublegetvar(intnum){  switch(num){  case0:  returnQ\_angle ;  break;  case1:  returnQ\_gyro ;  break;  case2:  returnR\_measure ;  break;  }  };  private :  doubleQ\_angle, Q\_gyro, R\_measure ;  doubleangle, bias ;  doubleP[2][2], K[2] ;  };  kalman kal ;  longac\_x, ac\_y, ac\_z, gy\_x, gy\_y, gy\_z ;  doubledeg, dgy\_y ;  doubledt ;  uint32\_tpasttime ;  voidsetup(){  // put your setup code here, to run once:  Serial.begin(9600);  Wire.begin();  Wire.beginTransmission(mpu\_add);  Wire.write(0x6B);  Wire.write(0);  Wire.endTransmission(true);  kal.init(0.001, 0.003, 0.03);//init kalman filter  Serial.println();  Serial.print("parameter");  Serial.print("\t");  Serial.print(kal.getvar(0), 4);  Serial.print("\t");  Serial.print(kal.getvar(1), 4);  Serial.print("\t");  Serial.println(kal.getvar(2), 4);  }  voidloop(){  // put your main code here, to run repeatedly:  Wire.beginTransmission(mpu\_add);//get acc data  Wire.write(0x3B);  Wire.endTransmission(false);  Wire.requestFrom(mpu\_add, 6, true);  ac\_x = Wire.read()<< 8| Wire.read();  ac\_y = Wire.read()<< 8| Wire.read();  ac\_z = Wire.read()<< 8| Wire.read();  Wire.beginTransmission(mpu\_add);//get gyro data  Wire.write(0x43);  Wire.endTransmission(false);  Wire.requestFrom(mpu\_add, 6, true);  gy\_x = Wire.read()<< 8| Wire.read();  gy\_y = Wire.read()<< 8| Wire.read();  gy\_z = Wire.read()<< 8| Wire.read();  deg = atan2(ac\_x, ac\_z)\* 180/ PI ;//acc data to degree data  dgy\_y = gy\_y / 131.;//gyro output to  dt = (double)(micros()- pasttime)/ 1000000;  pasttime = micros();//convert output to understandable data  doubleval = kal.getkalman(deg, dgy\_y, dt);//get kalman data  Serial.print("kalman degree");  Serial.print("\t");  Serial.println(val);  }  아래는 해당 코드의 플로우 차트이다.    해당 코드에서는 dt가 중요하다. 여기서는 아두이노의 처리속도를 기준으로 하였으며, 측정결과 dt값은 0.02정도가 나왔다. 상수형태로 넣어줄 수도 있으며, 각 값을 입력할 때마다 출력되는 형태가 다르게 나온다. 입력해주는 숫자가 작을 수록 값의 변화(각도의 변화)가 더 느리며, 그래프가 더 완만하게 변하게 된다. 반대로 큰 값을 입력할 경우 매우 빠른 속도로 값이 변화하며, 중간중간 값이 튀는 경우도 생길 수 있다. 따라서 중간정도의 값을 찾아내는것이 중요하다.      각도가 잘 받아지는 모습이다. 위치를 이동시켜도 실시간으로 각도의 변화가 눈에 보이며 딜레이도 크게 보이지 않았다. |
| **활동 사진** | 활동 내용 마지막의 사진 참조. |
| **활동 결과** | 칼만 필터를 사용하여 mpu6050으로 실시간 각도 변화를 측정하는 방법을 생각하여 실습을 해보았고, 생각한대로 잘 된 것을 확인할 수 있었다. 센서와 컴퓨터 간의 거리가 먼 로켓이나 위성의 경우에는 각도를 받아오는 도중에 딜레이가 생길 수 있겠지만, 거리가 가까운 상황만을 한정하였을 때는 값을 수월하게 받아오는 것을 알 수 있었다. |
| **계획** | 칼만 필터 이외의 방법을 실험해 보고, 가장 좋은 방법을 선택하여 먼 거리에서 값을 받아오는 경우에도 큰 문제가 없는지 확인할 것. |
| **출처** | https://m.blog.naver.com/roboholic84/220421114302  https://m.blog.naver.com/roboholic84/220401407348  https://pajamacoder.tistory.com/8 |

2024년 5월 5일

작성자 : 문시경 (인)