|  |
| --- |
| **2024 ALTIS NURA (SW사출팀)활동 보고서** |

|  |  |
| --- | --- |
| **활동 개요** | 낙하산 사출 조건 및 RAW데이터 필터링 |
| **일시** | 2024년 05월 19일 ~ 05월 26일 |
| **장소** | 407동 실험동 201호 |
| **참석자** | 양주호, 문시경 |
| 활동 내용 | **1. 로우패스필터 및 raw 데이터 적용 코드 (BMP280(고도)적용)**  **2. 칼만필터 및 raw 데이터 적용 코드 (MPU6050(각도), 엔코더)**  **3. 칼만필터 수식 및 raw 데이터 적용 코드 (BMI270(각도), 실제)**  **4. 사출조건 코드 (타이머, 각도, 고도)**  1**. 로우패스필터 및 raw 데이터 적용 코드 (BMP280(고도)적용)**  ○ 설명  - 버퍼 크기 10으로 해놨는데 bmp고도데이터가 저주파라 버퍼 크기 높을수록 좋을 건 없는데 실험 한 번 해보고 늘릴지 줄일지 정하겠음  - 센싱팀에서 로우데이터받아서 할 예정이므로 아래 필터부분을 제외한 실행코드는 실험 때만 쓰고, 시험발사 때는 필요없는 부분 지우고 센싱팀한테 받은 로우데이터 변수를 그대로 쓰겠음  - 가중치는 고도데이터는 현재시점 기준 최근 데이터가 더 중요하긴 한데 그렇다고 alpha값을 너무 크게 하면 데이터가 필터써도 튀는 부분이 있을 거고, 너무 낮추면 그래프는 부드러워지긴할텐데 최고점데이터찍는 시간이 지연되니까 일단 0.2로 두고 실험해서 바꿔보겠음  #include <Wire.h>  #include "i2c.h"  #include "i2c\_BMP280.h"  BMP280 bmp280;  #define BUFFER\_SIZE 10 // 버퍼 크기 10  float altitudeBuffer[BUFFER\_SIZE]; // 고도 데이터를 저장할 버퍼  int bufferIndex = 0; // 버퍼 인덱스 초깃값  void setup() {  Serial.begin(115200);  Serial.print("Probe BMP280: ");  if (bmp280.initialize()) Serial.println("Sensor found");  else {  Serial.println("Sensor missing");  while (1) {}  }  // BMP280 설정  bmp280.setPressureOversampleRatio(2);  bmp280.setTemperatureOversampleRatio(1);  bmp280.setFilterRatio(4);  bmp280.setStandby(0);  // BMP280 측정 시작  bmp280.setEnabled(0);  bmp280.triggerMeasurement();  }  void loop() {  bmp280.awaitMeasurement();  float altitude;  bmp280.getAltitude(altitude);  // 저주파 통과 필터 적용  float alpha = 0.2; // 가중치  altitudeBuffer[bufferIndex] = alpha \* altitude + (1 - alpha) \* altitudeBuffer[(bufferIndex + BUFFER\_SIZE - 1) % BUFFER\_SIZE]; //2번째항은 현재인덱스에 음수방지용으로 10더하고 1빼걸 10으로 나눠서 이전시점데이터 인덱스를 표현  bufferIndex = (bufferIndex + 1) % BUFFER\_SIZE; //데이터 담을 다음 버퍼로 바꾸기  // BMP280 다음 측정 트리거  bmp280.triggerMeasurement();  // 필터링된 고도 출력  Serial.print("Filtered Altitude: ");  Serial.println(altitudeBuffer[bufferIndex]);  delay(1000); // 1초 대기  }  **2. 칼만필터 및 raw 데이터 적용 코드 (MPU6050(각도), 엔코더)**  #include <Wire.h>  #include <MPU6050.h>  MPU6050 mpu;  // 칼만 필터 변수 및 상수  float angleX, angleY; // 추정된 각도  float biasX = 0, biasY = 0;  float P[2][2] = {0}, K[2];  const float Q\_angle = 0.001;  const float Q\_bias = 0.003;  const float R\_measure = 0.03;  float dt;  uint32\_t timer;  void setup() {  Serial.begin(115200);  Wire.begin();  mpu.initialize();  timer = micros();  }  void loop() {  // 센서에서 데이터 읽기  mpu.update();    float accX = mpu.getAccX();  float accY = mpu.getAccY();  float accZ = mpu.getAccZ();  float gyroX = mpu.getGyroX() - biasX;  float gyroY = mpu.getGyroY() - biasY;  float accAngleX = atan2(accY, accZ + PI) \* RAD\_TO\_DEG;  float accAngleY = atan2(accX, accZ + PI) \* RAD\_TO\_DEG;  dt = (micros() - timer) / 1000000.0f; // 시간차 계산  timer = micros();  // 자이로스코프 데이터를 각도로 변환  angleX += dt \* gyroX;  angleY += dt \* gyroY;  // 칼만 필터  // X축 칼만 필터 적용  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\_bias \* dt;  K[0] = P[0][0] / (P[0][0] + R\_measure);  K[1] = P[1][0] / (P[0][0] + R\_measure);  float y = accAngleX - angleX;  angleX += K[0] \* y;  biasX += K[1] \* y;  P[0][0] -= K[0] \* P[0][0];  P[0][1] -= K[0] \* P[0][1];  P[1][0] -= K[1] \* P[0][0];  P[1][1] -= K[1] \* P[0][1];  // Y축 칼만 필터 적용  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\_bias \* dt;  K[0] = P[0][0] / (P[0][0] + R\_measure);  K[1] = P[1][0] / (P[0][0] + R\_measure);  y = accAngleY - angleY;  angleY += K[0] \* y;  biasY += K[1] \* y;  P[0][0] -= K[0] \* P[0][0];  P[0][1] -= K[0] \* P[0][1];  P[1][0] -= K[1] \* P[0][0];  P[1][1] -= K[1] \* P[0][1];  // 값 출력  Serial.print("Kalman Angle X: ");  Serial.print(angleX);  Serial.print(" | Kalman Angle Y: ");  Serial.println(angleY);  delay(50);  }  **3. 칼만필터 수식 및 raw 데이터 적용 코드 (BMI270(각도), 실제)**  #include <Wire.h>  #include <BMI270.h> // 가정된 BMI270 라이브러리  BMI270 imu;  float angleX, angleY; // 추정된 각도  float biasX = 0, biasY = 0;  float P[2][2] = {0}, K[2];  const float Q\_angle = 0.001;  const float Q\_bias = 0.003;  const float R\_measure = 0.03;  float dt;  uint32\_t timer;  void setup() {  Serial.begin(115200);  Wire.begin();  imu.begin(); // BMI270 초기화  timer = micros();  }  void loop() {  imu.update(); // 데이터 업데이트    float accX = imu.readAccX(); // 데이터 읽기  float accY = imu.readAccY();  float accZ = imu.readAccZ();  float gyroX = imu.readGyroX() - biasX;  float gyroY = imu.readGyroY() - biasY;  float accAngleX = atan2(accY, sqrt(accX \* accX + accZ \* accZ)) \* RAD\_TO\_DEG; // 수정된 계산  float accAngleY = atan2(accX, sqrt(accY \* accY + accZ \* accZ)) \* RAD\_TO\_DEG; // 수정된 계산  dt = (micros() - timer) / 1000000.0f;  timer = micros();  // 자이로스코프 데이터를 각도로 변환  angleX += dt \* gyroX;  angleY += dt \* gyroY;  // 칼만 필터 알고리즘은 그대로 유지  // 값 출력  Serial.print("Kalman Angle X: ");  Serial.print(angleX);  Serial.print(" | Kalman Angle Y: ");  Serial.println(angleY);  delay(50);  }  **4. 사출조건 코드**  ○ 설명  - 아래 코드에는 칼만 필터를 적용해놨는데 실험 시 로우패스필터도 사용하여 어떤 필터가 적합한 지 결정할 것임  - 타이머, 기울기, 고도에 대한 3가지 조건  - 사용 센서  - BMP280 : 고도 측정, 칼만 필터 적용되어 증가하는 추세를 보이다가 감소로 바뀌면 사출 조건  - MPU6050 : 각도 측정, 지면 90도 기준 45도가 되면 사출 조건  - 타이머 : 코드 시작후 15초 후에 loop문 시작, loop문 시작 후 15초가 지나면 사출 조건  - 버튼 : 10번 핀에 연결, 10kΩ 저항 추가할 필요 있음, 강제 사출 조건  - LED : BMP가 데이터를 못받을 경우 깜빡이는 것, 사출 조건 달성시 깜빡이는 것 따로 존재  - x,y축 각도, 90도 기준 편차 각도, 실시간 고도는 시리얼 모니터로 출력됨  - 칼만 필터의 Q와 R 값은 실험을 기반으로 조율하여, 센서 특유의 노이즈와 환경에 가장 적합하게 만들어야 함  #include <Wire.h>  #include <Adafruit\_Sensor.h>  #include <Adafruit\_BMP280.h>  #include <MPU6050.h>  #define BUTTON\_PIN 10  #define LED\_PIN 13  #define ERROR\_LED\_PIN 4  #define SEA\_LEVEL\_PRESSURE 1013.25  const int START\_DELAY = 30000; // 시작 후 딜레이(ms)  const float CRITICAL\_TILT\_ANGLE = 45.0; // 위험 기울기 각도  Adafruit\_BMP280 bmp;  MPU6050 mpu;  unsigned long startTime;  // 칼만 필터 변수  float Q\_altitude = 0.1;  float R\_altitude = 0.5;  float P = 1.0;  float altitudeEstimate = 0.0;  float previousAltitudeEstimate = 0.0; // 이전 고도 추정값  bool increasing = true; // 고도가 증가 중인지 여부  void setup() {  Serial.begin(9600);  Wire.begin();  mpu.initialize();  startTime = millis();  if (!bmp.begin()) {  Serial.println(F("Could not find a valid BMP280 sensor, check wiring or try resetting your board."));  pinMode(ERROR\_LED\_PIN, OUTPUT);  while(true) {  digitalWrite(ERROR\_LED\_PIN, HIGH); // 오류 시 LED 켜기  delay(500);  digitalWrite(ERROR\_LED\_PIN, LOW); // 오류 표시 LED 깜박임  delay(500);  }  bmp.setSampling(Adafruit\_BMP280::MODE\_NORMAL, Adafruit\_BMP280::FILTER\_OFF);  pinMode(BUTTON\_PIN, INPUT);  pinMode(LED\_PIN, OUTPUT);  }  void loop() {  float altitude = bmp.readAltitude(SEA\_LEVEL\_PRESSURE); // 상수 사용  mpu.update(); // 데이터 업데이트  updateKalmanAltitude(altitude); // 칼만 필터를 사용한 고도 업데이트  float angleX = mpu.getAngleX();  float angleY = mpu.getAngleY();  float tiltAngle = sqrt(angleX \* angleX + angleY \* angleY);  float deviationAngle = abs(90 - tiltAngle); // 수직에서의 편차  // 고도가 감소하기 시작하는지 감지  if (increasing && (previousAltitudeEstimate > altitudeEstimate)) {  increasing = false;  }    // 출력  Serial.print("X Axis Angle: ");  Serial.print(angleX);  Serial.print(" degrees, Y Axis Angle: ");  Serial.print(angleY);  Serial.print(" degrees, Deviation from Vertical: ");  Serial.print(deviationAngle);  Serial.print(" degrees, Kalman Filtered Altitude: ");  Serial.print(altitudeEstimate);  Serial.println(" meters");  //타이머, 각도, 고도 조건 3개 넣은 것  if ((millis() - startTime > START\_DELAY) || (deviationAngle >= CRITICAL\_TILT\_ANGLE) || digitalRead(BUTTON\_PIN) || !increasing) {  blinkLED(5); // LED를 5번 깜빡이도록 호출  Serial.println("Condition met. LED blinked.");  }  previousAltitudeEstimate = altitudeEstimate; // 이전 고도 추정값 업데이트  delay(500);  }  void blinkLED(int blinkCount) {  for (int i = 0; i < blinkCount; i++) {  digitalWrite(LED\_PIN, HIGH);  delay(250);  digitalWrite(LED\_PIN, LOW);  delay(250);  }  }  void updateKalmanAltitude(float altitude) {  float K = P / (P + R\_altitude);  altitudeEstimate = altitudeEstimate + K \* (altitude - altitudeEstimate);  P = (1 - K) \* P + Q\_altitude; // 오차 공분산 업데이트  } |
| 활동 사진 |  |
| 활동 결과 | - 로우패스필터, 칼만필터 정리  - 사출조건설정 (타이머, 각도, 고도) |
| 계획 | - 5.27(월)에 위 코드를 bmp280에 적용하여 모터대신 led로 사출조건을 만족하는 지 확인  - 모터가 정해지면 그 모터에 맞게 조건에 따른 명령문을 바꿀것임(모터 작동하는 코드 준비해 둠) |

2024년 05월 26일

작성자 : 양주호 (인)