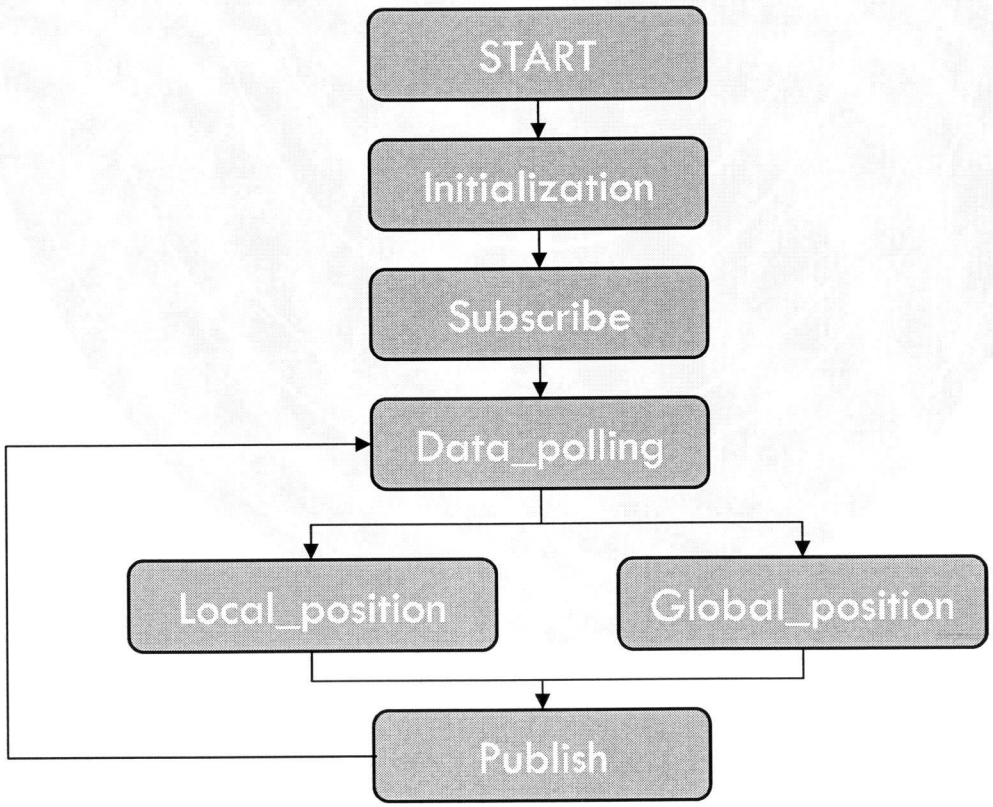
POSITION_ESTIMATOR

POSITION_ESTIMATOR_INAV_MAIN.C

동작의 기본 구조


START
Initialization
Subscribe
Data_polling
Local_position
Global_position
Publish

LOCAL_POSITION

- INPUT:
  - ACCELEROMETER
  - GPS
  - HOME_POSITION
  - PARAMS(사용자 정의 파라미터)
- OUTPUT:
  - X,Y,Z POSITION
  - X,Y,Z VELOCITY
  - YAW
  - EPH, EPV

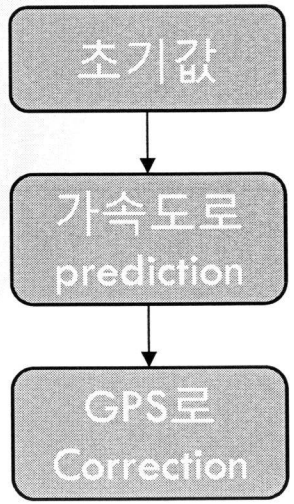
GLOBAL_POSITION

- INPUT:
  - ACCELEROMETER
  - GPS
  - HOME_POSITION
- OUTPUT:
  - GLOBAL LAT, LON, ALT
  - GLOBAL NED VELOCITY
  - GLOBAL YAW
  - EPH, EPV

LOCAL_XY POSITION & VELOCITY

- 초기값 P(0, 0) V(GPS.N_VEL, GPS. E_VEL)

```
X_EST[0] = 0.0F;                    //X_POSITION
X_EST[1] = GPS.VEL_N_M_S;           //X_VELOCITY
Y_EST[0] = 0.0F;                    //Y_POSITION
Y_EST[1] = GPS.VEL_E_M_S;           //Y_VELOCITY
```

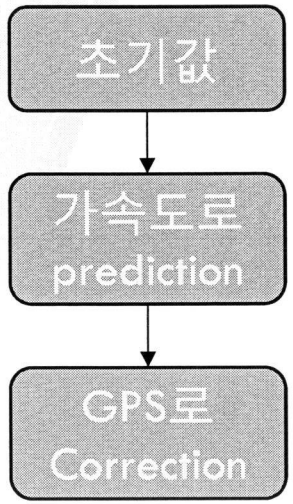


LOCAL_XY POSITION & VELOCITY

//가속도 센서의 값을 가져와 PREDICTION

```
void inertial_filter_predict(float dt, float x[2], float acc)
{
    if (isfinite(dt)) {
        if (!isfinite(acc)) {
            acc = 0.0f;
        }

        x[0] += x[1] * dt + acc * dt * dt / 2.0f;
        x[1] += acc * dt;
    }
}
```



The Kinematic Equations

$d = v_i * t + \frac{1}{2} * a * t^2$ $\qquad v_f^2 = v_i^2 + 2 * a * d$

$v_f = v_i + a * t$ $\qquad d = \frac{v_i + v_f}{2} * t$

$$x = v_x . \Delta t + \frac{1}{2} a_x . \Delta t^2 \qquad v_{x1} = a_x . \Delta t$$

$$y = v_y . \Delta t + \frac{1}{2} a_y . \Delta t^2 \qquad v_{y1} = a_y . \Delta t$$

$$z = v_z . \Delta t + \frac{1}{2} a_z . \Delta t^2 \qquad v_{z1} = a_z . \Delta t$$

LOCAL_XY POSITION & VELOCITY

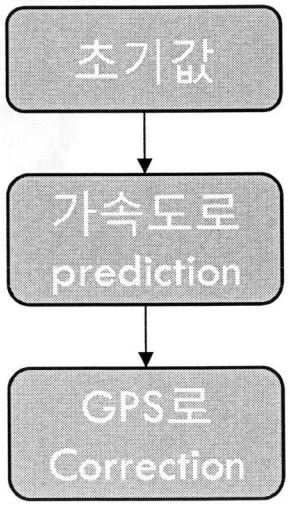
- GPS로 CORRECTION

INERTIAL_FILTER_CORRECT(CORR_GPS[0][0], DT, X_EST, 0, W_XY_GPS_P);

INERTIAL_FILTER_CORRECT(CORR_GPS[1][0], DT, Y_EST, 0, W_XY_GPS_P);

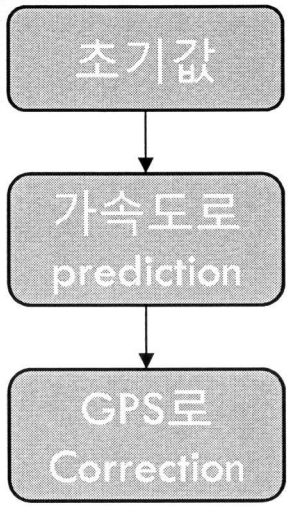
X_EST[0] += CORR [0][0] * DT * W_XY_GPS_P

Y_EST[0] += CORR [1][0] * DT * W_XY_GPS_P



LOCAL_XY POSITION & VELOCITY

- **INT EST_I = BUF_PTR - 1 - MIN(EST_BUF_SIZE - 1, MAX(0, (INT)(PARAMS.DELAY_GPS * 1000000.0F / PUB_INTERVAL)));**
- **IF (EST_I < 0) {**
- EST_I += EST_BUF_SIZE;
- }
- /* CALCULATE CORRECTION FOR POSITION */
- CORR_GPS[0][0] = GPS_PROJ[0] - EST_BUF[EST_I][0][0];
- CORR_GPS[1][0] = GPS_PROJ[1] - EST_BUF[EST_I][1][0];
- /* CALCULATE CORRECTION FOR VELOCITY */
- **IF (GPS.VEL_NED_VALID) {**
- CORR_GPS[0][1] = GPS.VEL_N_M_S - EST_BUF[EST_I][0][1];
- CORR_GPS[1][1] = GPS.VEL_E_M_S - EST_BUF[EST_I][1][1];

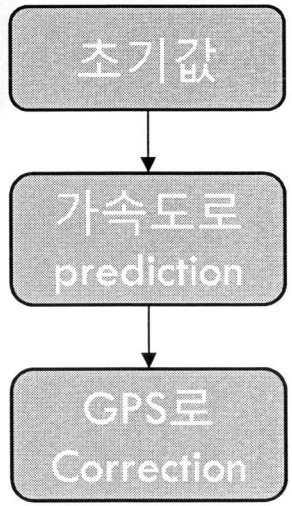


LOCAL_XY POSITION & VELOCITY

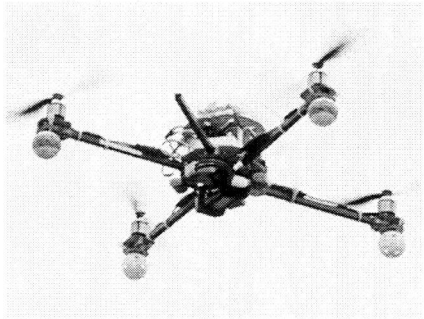
- **FLOAT W_XY_GPS_P = PARAMS.W_XY_GPS_P * W_GPS_XY;**
- **FLOAT W_XY_GPS_V = PARAMS.W_XY_GPS_V * W_GPS_XY;**

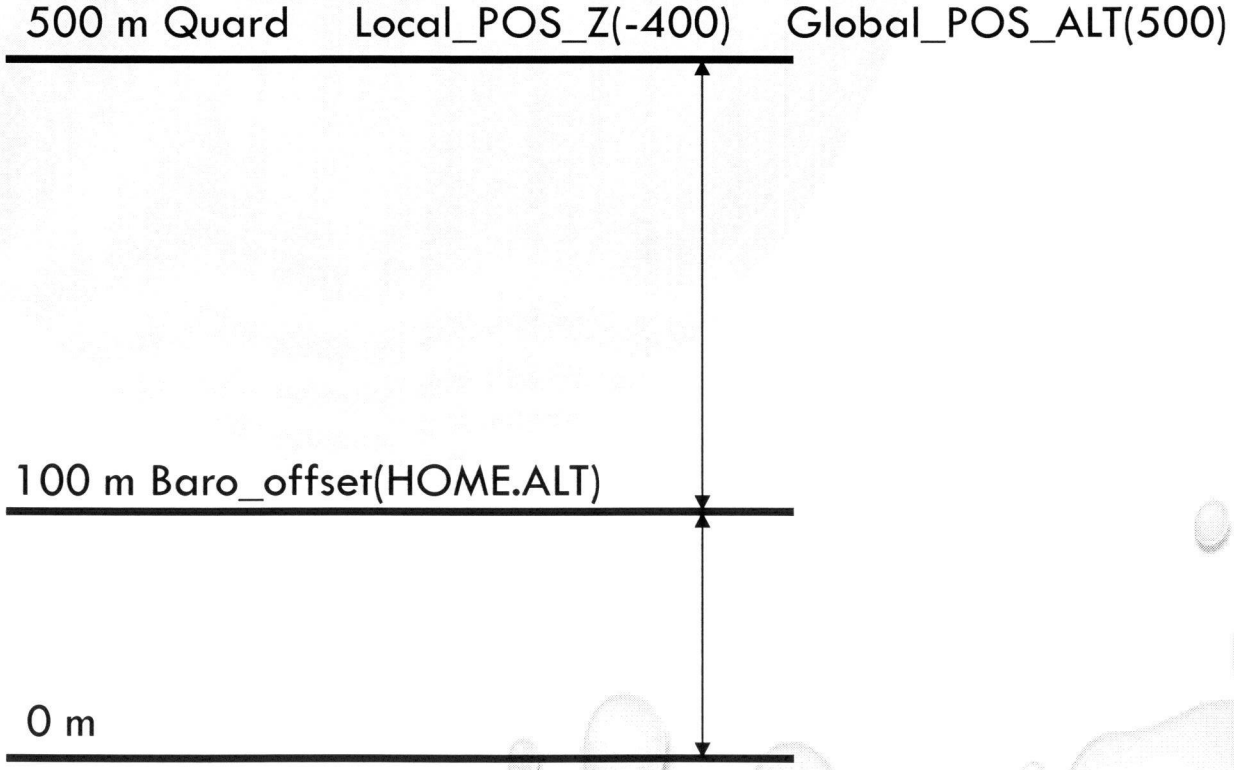
- W_GPS_XY = MIN_EPH_EPV / FMAXF(MIN_EPH_EPV, GPS.EPH);

- // MIN_EPH_EPV = 2.0F
- // MAX_EPH_EPV = 20.0F



LOCAL_Z POSITION & VELOCITY





LOCAL_Z POSITION & VELOCITY

- 기본적인 동작의 구조는 XY와 동일하나 BAROMETOR 가 추가됨
- BARO_OFFSET += SENSOR.BARO_ALT_METER; (초기에 여러 번 더한 뒤 평균을 구해 사용)
- CORR_BARO = BARO_OFFSET - SENSOR.BARO_ALT_METER - Z_EST[0];
- INERTIAL_FILTER_CORRECT(CORR_BARO, DT, Z_EST, 0, PARAMS.W_Z_BARO);

GLOBAL LAT, LON, ALT

- MAP_PROJECTION_REPROJECT(&REF, LOCAL_POS.X, LOCAL_POS.Y, &EST_LAT, &EST_LON);

- GLOBAL_POS.LAT = EST_LAT;
- GLOBAL_POS.LON = EST_LON;
- GLOBAL_POS.ALT = LOCAL_POS.REF_ALT - LOCAL_POS.Z;
- // LOCAL_POS.REF_ALT = HOME.ALT;

GLOBAL NED VELOCITY

- GLOBAL_POS.VEL_N = LOCAL_POS.VX;
- GLOBAL_POS.VEL_E = LOCAL_POS.VY;
- GLOBAL_POS.VEL_D = LOCAL_POS.VZ;

LOCAL & GLOBAL YAW

- LOCAL_POS.YAW = ATT.YAW;

- GLOBAL_POS.YAW = LOCAL_POS.YAW;

EPH, EPV

- LOCAL_POS.EPH = EPH;
- LOCAL_POS.EPV = EPV;

- GLOBAL_POS.EPH = EPH;
- GLOBAL_POS.EPV = EPV;

EPH, EPV

```
/* INCREASE EPH/EPV ON EACH STEP */
IF (EPH < MAX_EPH_EPV) {
    EPH *= 1.0F + DT;
}
IF (EPV < MAX_EPH_EPV) {
    EPV += 0.005F * DT;// ADD 1M TO EPV EACH 200S (BARO DRIFT)
}
```

GPS 유효 검사

```
IF (GPS_VALID) {
        IF (GPS.EPH > MAX_EPH_EPV || GPS.EPV > MAX_EPH_EPV || GPS.FIX_TYPE < 3) {
        GPS_VALID = FALSE;
        }
} ELSE {
        IF (GPS.EPH < MAX_EPH_EPV * 0.7F && GPS.EPV < MAX_EPH_EPV * 0.7F &&
        GPS.FIX_TYPE >= 3) {
        GPS_VALID = TRUE;
}
```

//GPS FIX: 0 = NO FIX, 1 = DEAD RECKONING ONLY, 2 = 2D FIX, 3 = 3D-FIX, 4 = GPS + DEAD RECKONING, 5 = TIME ONLY FIX

POSITION__ESTIMATOR__INAV__PARAMS

```
PARAM_DEFINE_FLOAT(INAV_W_Z_BARO, 0.5F);
PARAM_DEFINE_FLOAT(INAV_W_Z_GPS_P, 0.005F);
PARAM_DEFINE_FLOAT(INAV_W_Z_GPS_V, 0.0F);
PARAM_DEFINE_FLOAT(INAV_W_Z_VIS_P, 0.5F);
PARAM_DEFINE_FLOAT(INAV_W_Z_SONAR, 3.0F);
PARAM_DEFINE_FLOAT(INAV_W_XY_GPS_P, 1.0F);
PARAM_DEFINE_FLOAT(INAV_W_XY_GPS_V, 2.0F);
PARAM_DEFINE_FLOAT(INAV_W_XY_VIS_P, 7.0F);
PARAM_DEFINE_FLOAT(INAV_W_XY_VIS_V, 0.0F);
PARAM_DEFINE_FLOAT(INAV_W_XY_FLOW, 5.0F);
PARAM_DEFINE_FLOAT(INAV_W_XY_RES_V, 0.5F);
PARAM_DEFINE_FLOAT(INAV_W_GPS_FLOW, 0.1F);
PARAM_DEFINE_FLOAT(INAV_W_ACC_BIAS, 0.05F);
PARAM_DEFINE_FLOAT(INAV_FLOW_K, 0.15F);
PARAM_DEFINE_FLOAT(INAV_FLOW_Q_MIN, 0.5F);
PARAM_DEFINE_FLOAT(INAV_SONAR_FILT, 0.05F);
PARAM_DEFINE_FLOAT(INAV_SONAR_ERR, 0.5F);
PARAM_DEFINE_FLOAT(INAV_LAND_T, 3.0F);
PARAM_DEFINE_FLOAT(INAV_LAND_DISP, 0.7F);
PARAM_DEFINE_FLOAT(INAV_LAND_THR, 0.2F);
PARAM_DEFINE_FLOAT(INAV_DELAY_GPS, 0.2F);
PARAM_DEFINE_INT32(CBRK_NO_VISION, 0);
PARAM_DEFINE_INT32(INAV_ENABLED, 1);
```