GTM-200 User Guide

INDEX

1. FEATURE ····· 3

2. UART Protocol ····· 4

3. Data Output ····· 4

4. AXIS ASSIGNMENT ····· 6

5. SETTING ····· 7

6. COMMAND ····· 9

7. APPLICATION ····· 12

1. FEATURE

3축 자이로스코프, 3축 가속도센서, 3축 지자기센서가 내장된
9축 IMU/AHRS 센서 모듈

연결 방식: USB C (USB 2.0)
통신 속도 지원 범위: 9600bps ~ 921600bps
데이터 갱신 및 데이터 출력 속도: 100Hz
  ** 출고 시 설정 가능 범위: 1 ~ 1,000Hz
데이터 출력 형식: Euler angles(*Pitch,Roll,Yaw)
  ** 출고 시 설정 가능 형식:
    Euler angles, Quaternion, sensor raw data
자이로 민감도:
  range: ±2000dps
  resolution: 0.061dps (2000/32768)
가속도 민감도:
  range: ±16g
  resolution: 0.00048828g (16/32768)
지자기 민감도:
  range: 4900uT
  resolution: 0.15uT (4900/32768)

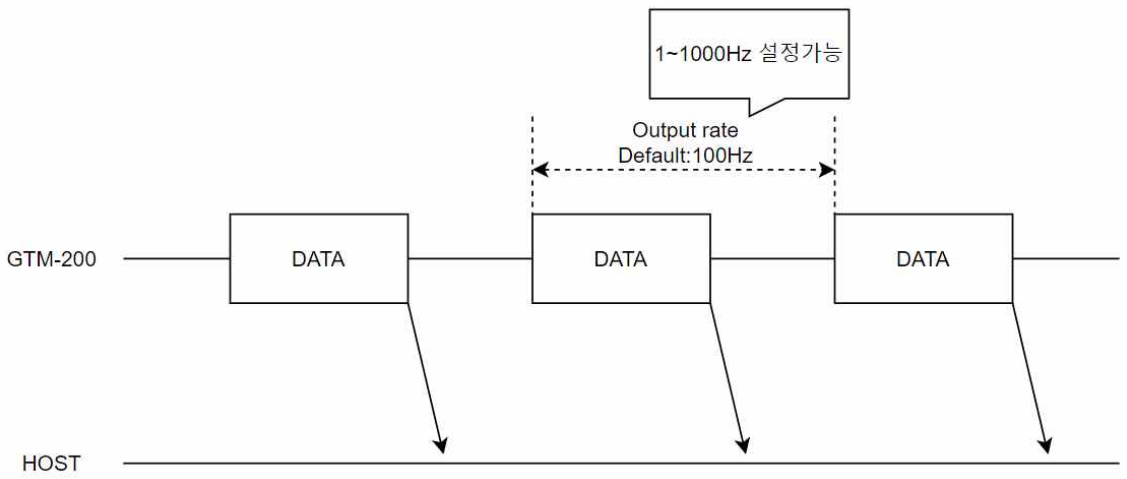
지자기센서 활성/비활성화 기능 (default: 비활성화)
자이로, 가속도, 지자기센서 캘리브레이션 기능
지자기센서 활성/비활성화 상태 확인 기능
프로그램 버전 확인 기능
전원 연결 시 자이로, 가속도 캘리브레이션 자동 실행 (3초 대기 후 사용가능)

2. UART Protocol

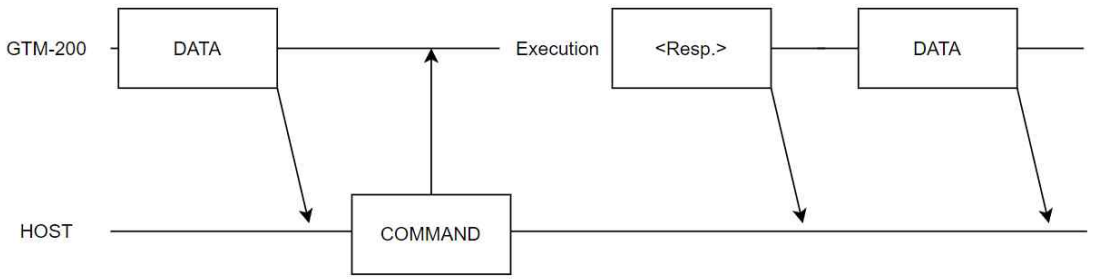
BAUDRATE : 9600bps ~ 921600bps(default:115200bps)
PARITYBIT : No Parity
DATABIT : 8 Data bits
STOPBIT : 1 Stop bit
No H/W flow controls

3. Data Output

3-1. 일반적인 데이터 출력



3-2. COMMAND Operations



**<Resp.>소요 시간(6-1. COMMAND CODE LIST) 참조

3-3 OUTPUT DATA FORMAT

SOL	DATA 1	sp	DATA 2	sp	DATA 3	EOL
*	ascii data 1 (Pitch)	,	ascii data 2 (Roll)	,	ascii data 3 (Yaw)	CR LF

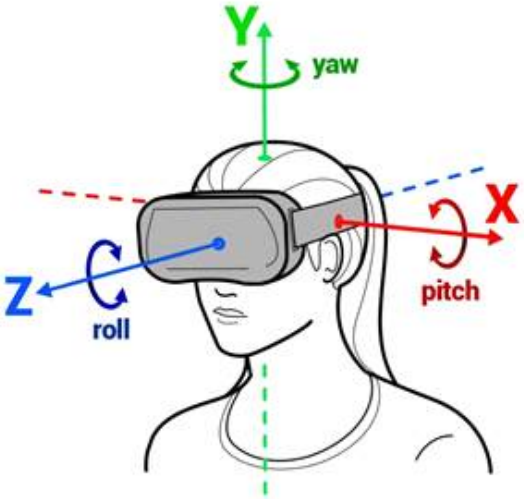
** 출고 시 설정 가능 형식: Euler angles, Quaternion, sensor raw data
Default: Euler angles

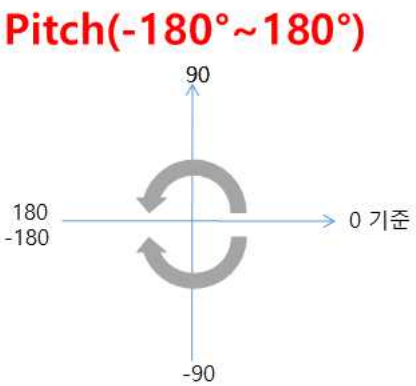
ex) *-2.20,-0.21,138.37(CR)(LF)
Pitch: -2.20°, Roll: -0.21°, Yaw: 138.37°

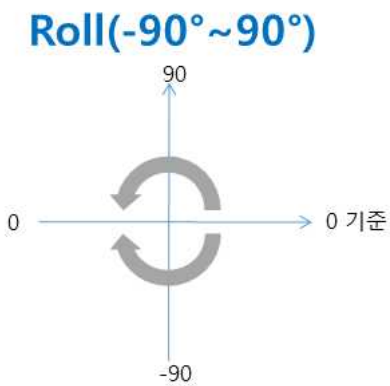
T: 0.01s (100Hz)

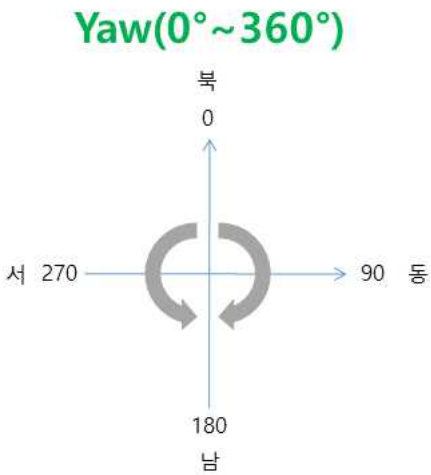
```
14:03:34.102 *-2.20,-0.21,138.37
14:03:34.112 *-2.20,-0.21,138.37
14:03:34.121 *-2.20,-0.21,138.37
14:03:34.131 *-2.20,-0.21,138.37
14:03:34.141 *-2.20,-0.21,138.37
14:03:34.151 *-2.20,-0.21,138.37
14:03:34.161 *-2.20,-0.21,138.37
14:03:34.171 *-2.20,-0.21,138.37
14:03:34.182 *-2.20,-0.21,138.37
14:03:34.191 *-2.20,-0.21,138.37
14:03:34.201 *-2.20,-0.21,138.36
14:03:34.212 *-2.20,-0.21,138.36
14:03:34.221 *-2.20,-0.21,138.36
14:03:34.231 *-2.19,-0.21,138.36
14:03:34.241 *-2.19,-0.21,138.36
14:03:34.251 *-2.20,-0.21,138.36
14:03:34.261 *-2.20,-0.21,138.36
14:03:34.271 *-2.20,-0.21,138.36
14:03:34.281 *-2.20,-0.21,138.37
14:03:34.292 *-2.19,-0.21,138.37
14:03:34.301 *-2.20,-0.21,138.37
14:03:34.311 *-2.20,-0.21,138.37
```

4. AXIS ASSIGNMENT









5. SETTING

5-1. 기본 세팅

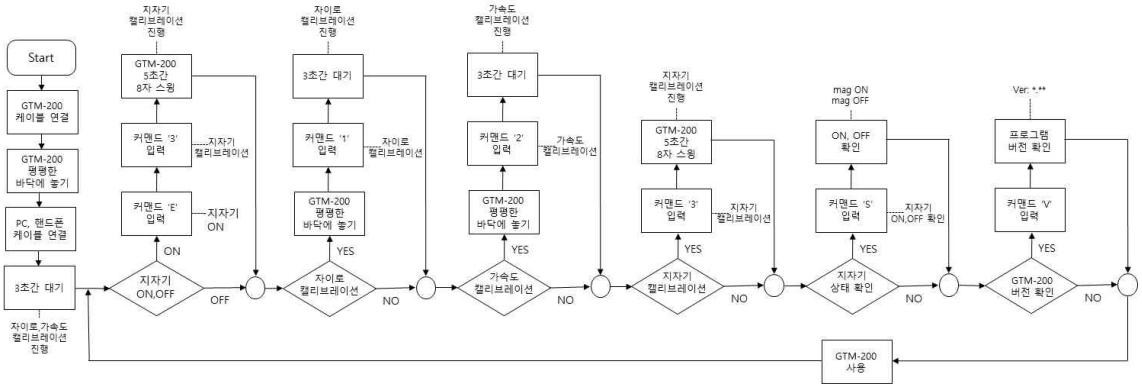
Output data: *Pitch,Roll,Yaw
자이로센서: ON
가속도센서: ON
지자기센서: OFF
**전원 공급시 GTM-200의 정면의
Yaw값이 270으로 표시됨
Pitch 범위: ±180˚
Roll 범위: ±90˚
Yaw 범위: 0˚~360˚

5-2. 사용 규정

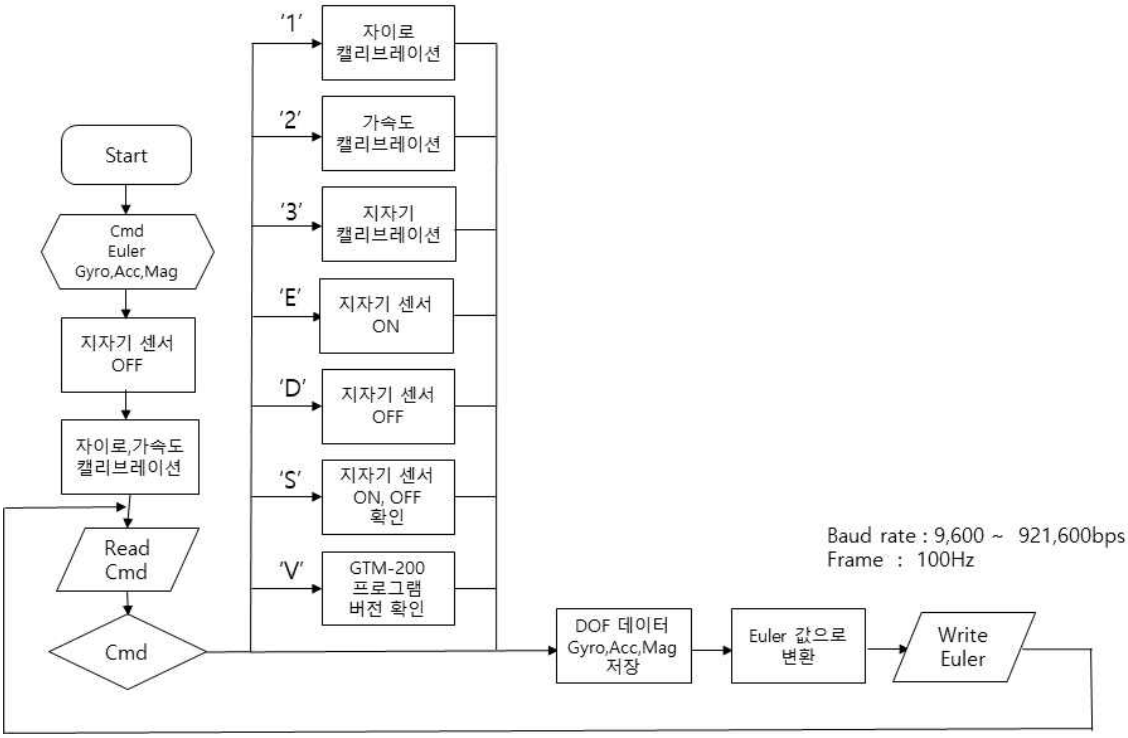
1. GTM-200에 케이블 연결
2. GTM-200를 흔들리지 않는 평평한 바닥에 내려놓기
3. 스마트폰, PC에 케이블 연결
4. 3초 대기 (자이로,가속도 캘리브레이션 진행)
5. 사용

*기기 사용시 꼭 위 사항을 준수하시기 바랍니다.

5-3. User Operation Procedure



5-4. Internal 9DOF Operation Procedure



6. COMMAND

6-1. COMMAND CODE LIST

설명 / Command	동작	기기 상태	Comment
1	자이로센서 캘리브레이션	평평한 장소에 흔들리지 않게 벗어두고 진행	정지상태 3초 이상 유지 <Resp.>: 3s
2	가속도센서 캘리브레이션		
3	지자기센서 캘리브레이션	기기를 들고 8자 그리기	8자 스윙 5초 이상 실행 <Resp.>: 5s
E	지자기센서 활성화	최초 1회 지자기센서 캘리브레이션 실행(이후 상관없음)	주위 환경에 따른 지자기 오류를 대처하기 위해 ON/OFF 모드 지원 <Resp.>: 1s
D	지자기 센서 비활성화	상관없음	
S	Status (지자기 ON,OFF)표기		1초간 mag ON,OFF 상태 표기 <Resp.>: 1s
V	GTM-200 Version 표기		1초간 Ver:*.** 표기 <Resp.>: 1s

6-2. COMMAND DETAILS

6-2-1. 자이로센서 캘리브레이션

자이로센서 x,y,z축 캘리브레이션을 합니다.

캘리브레이션시 기기는 반드시 움직임이 없는 정지 상태에 있어야 합니다.

[설정 순서]

1. 기기를 방향/자세 상관없이 움직임이 없는 상태로 놓습니다.
2. '1' 명령어 입력 후 3초간 대기합니다.
3. 기기를 착용하고 다시 사용합니다.

6-2-2. 가속도센서 캘리브레이션

가속도센서 x,y,z축 캘리브레이션을 합니다.

캘리브레이션시 기기는 평평한 면에 바르게 놓여야합니다.

[설정 순서]

1. 기기를 기울어짐 없이 평평한 면에 바르게놓습니다.
2. '2' 명령어 입력 후 3초간 대기합니다.
3. 기기를 착용하고 다시 사용합니다.

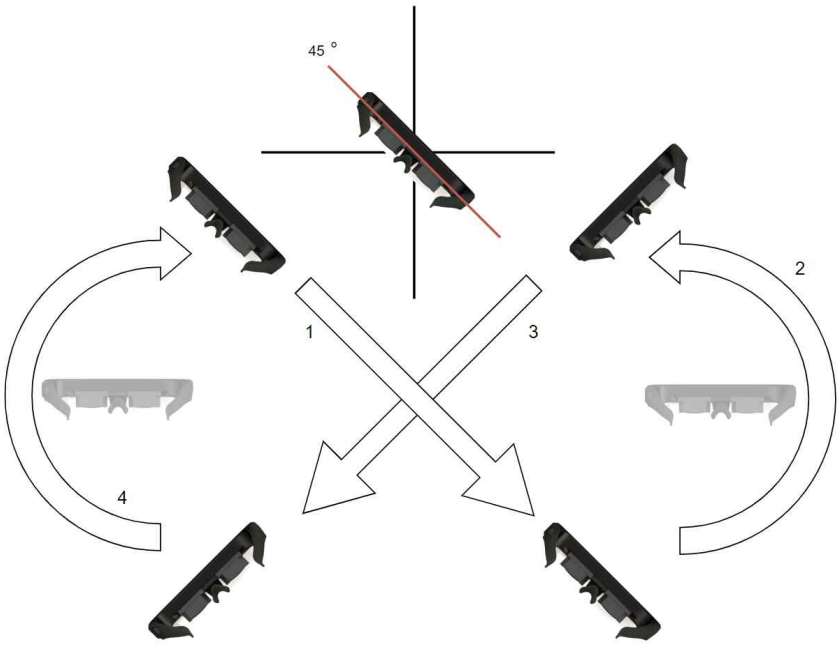
6-2-3. 자이로센서 캘리브레이션

자이로센서 x,y,z축 캘리브레이션을 합니다.

캘리브레이션시 기기는 45˚ 이상 기울여 가며 8자로 움직여 주어야 합니다.

[설정 순서]

1. 기기를 양손으로 잡아 듭니다.
2. '3' 명령어 입력 후 아래의 그림과 같이 8자를 그리며 5초 이상 움직여 줍니다.
3. 기기를 착용하고 다시 사용합니다.



6-2-4. 자이로센서 활성화

지자기센서를 활성화하여 자북을 탐지할 수 있게 합니다.
최초 1회 실행 시에는 지자기 캘리브레이션을 해주어야 합니다.
처음 자북과 일치하는데 몇 초간 시간이 걸립니다.
[설정 순서]
1. ‘E’ 명령어 입력합니다.
2. 기기를 사용합니다.

6-2-5. 자이로센서 비활성화

지자기센서를 비활성화하여 자이로,가속도센서만을 사용합니다.
시간변화에 따라 약간의 드리프트 현상이 누적될 수 있습니다.
[설정 순서]
1. ‘D’ 명령어 입력합니다.
2. 기기를 사용합니다.

6-2-6. Status(지자기 ON,OFF)표기

지자기센서의 활성화,비활성화 상태를 1초간 표기합니다.
표기 중에는 Pitch,Roll,Yaw 데이터가 출력되지 않습니다.
[설정 순서]
1. ‘S’ 명령어 입력합니다.
2. 1초간 출력되는 mag ON,OFF 메시지를 확인

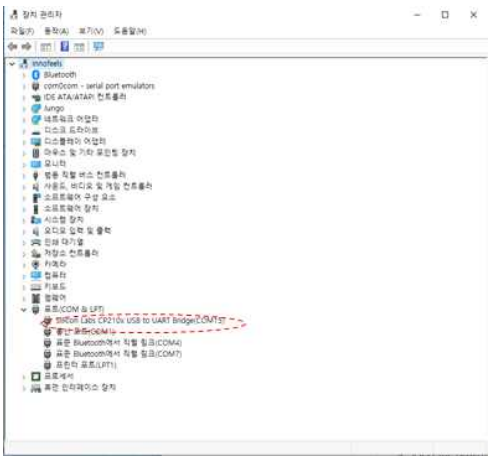
6-2-7. GTM-200 Version 표기

GTM-200의 현재 버전을 1초간 표기합니다.
표기 중에는 Pitch,Roll,Yaw 데이터가 출력되지 않습니다.
[설정 순서]
1. ‘V’ 명령어 입력합니다.
2. 1초간 출력되는 Ver:*.** 메시지를 확인

7. APPLICATION

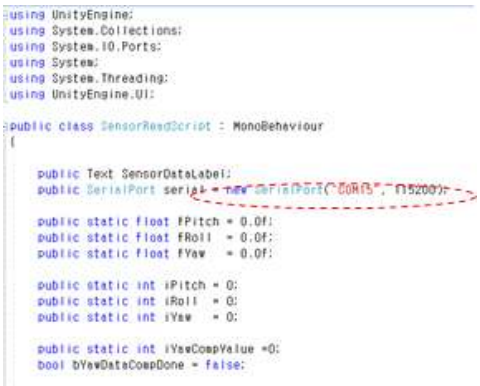
7-1. UNITY Environment

1. GTM-200 USB-C 연결
2. 장치관리자 포트 확인



3. DemoDSensor 예제

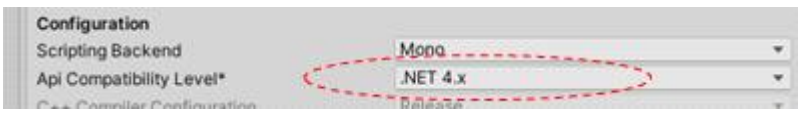
-> Assets>Scripts>SensorReadScripts.cs (COM port 설정)



```
using UnityEngine;
using System.Collections;
using System.IO.Ports;
using System;
using System.Threading;
using UnityEngine.UI;

public class SensorReadScript : MonoBehaviour
{

    public Text SensorDataLabel;
    public SerialPort serial = new SerialPort("COM3", 115200);

    public static float fPitch = 0.0f;
    public static float fRoll  = 0.0f;
    public static float fYaw   = 0.0f;

    public static int iPitch = 0;
    public static int iRoll  = 0;
    public static int iYaw   = 0;

    public static int iYawCompValue =0;
    bool bYawDataCompDone = false;
```

4. Edit -> Project Settings -> Player -> Other Settings:

Api Compatibility Level*



5. 9DOF 센서 데이터 해석

-> Assets>Scripts>SensorReadScripts.cs

매 센서값 입력 시 Event 발생

```
void SerialEvent(SerialPort aPort)
{
    string inputText = aPort.ReadLine(); //the ascii value of the "1" character

    if (inputText != null)
    {

        string[] split_text;
        split_text = inputText.Split(',');

        string[] temp_text;
        temp_text = split_text[0].Split('=');

        fRoll  = float.Parse(temp_text[1]);
        fPitch = float.Parse(split_text[1]);
        fYaw   = float.Parse(split_text[2]);

        iRoll  = (int)fRoll;
        iPitch = (int)fPitch;
        iYaw   = (int)fYaw;


        if (bYawDataCompDone == false)
        {
            iYawCompValue = iYaw;
            bYawDataCompDone = true;
        }

        // after YAW 보정 후
        else
        {

            iYaw = iYaw - Math.Abs(iYawCompValue);
            //if (iYaw < 0) iYaw = 360 + iYaw;
        }

        SensorDataLabel.text = "[ DOF ]" + " R:" + iRoll + " P:" + iPitch + " Y:" + iYaw + " YC:" + iYawCompValue + "\r\n";

    }
}
```

Roll, Pitich, Yaw & 초기 Yaw 값

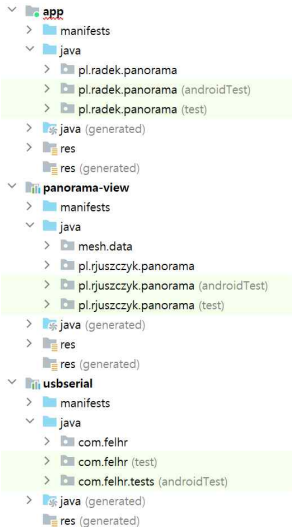
```
[ DOF ] P:-4 R:4 Y:-1 YC:-173
```

GTM-200 센서 계산

```
//
// V1
//iRoll = (int)fRoll;
//iPitch = (int)fPitch;
//iYaw = (int)fYaw;

// V2
iRoll = (int)fPitch;
iPitch = (int)fRoll*(-1);
iYaw = (int)fYaw;
```

7-2. ANDROID STUDIO Environment

1. 구성



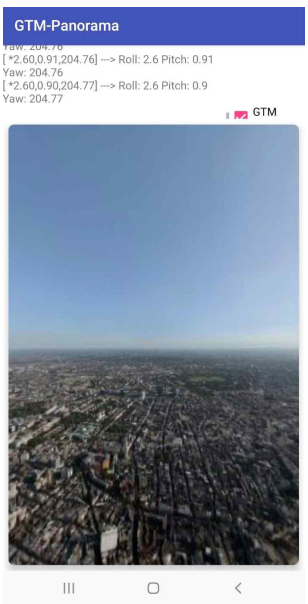
2. 9DOF 센서 데이터 해석

-> panorama-view>java>pl.rjuszczyk.panorama>gyroscope>GyroscopeHandler

```java
// This timestep's delta rotation to be multiplied by the current
// rotation after computing it from the gyro sample data.
if (timestampOldCalibrated != 0 && stateInitializedCalibrated) {
    final float dT = (timestamp - timestampOldCalibrated) * NS2S;

    // Axis of the rotation sample, not normalized yet.
    //float axisX = gyroscope[0];
    //float axisY = gyroscope[1];
    //float axisZ = gyroscope[2];

    float axisX = gyroscope[1];
    float axisY = gyroscope[0];
    float axisZ = gyroscope[2];

    // Calculate the angular speed of the sample
    float omegaMagnitude = (float) Math.sqrt(axisX * axisX + axisY * axisY + axisZ * axisZ);

    // Normalize the rotation vector if it's big enough to get the axis
    if (omegaMagnitude > EPSILON) {
        axisX /= omegaMagnitude;
        axisY /= omegaMagnitude;
        axisZ /= omegaMagnitude;
    }

    //Log.d("sensor", "x= " + axisX + " y= " + axisY + "    z= " + axisY);

    // Integrate around this axis with the angular speed by the timestep
    // in order to get a delta rotation from this sample over the
    // timestep. We will convert this axis-angle representation of the
    // delta rotation into a quaternion before turning it into the
    // rotation matrix.
    float thetaOverTwo = omegaMagnitude * dT / 2.0f;

    float sinThetaOverTwo = (float) Math.sin(thetaOverTwo);
    float cosThetaOverTwo = (float) Math.cos(thetaOverTwo);

    deltaRotationVectorCalibrated[0] = sinThetaOverTwo * axisX;
    deltaRotationVectorCalibrated[1] = sinThetaOverTwo * axisY;
    deltaRotationVectorCalibrated[2] = sinThetaOverTwo * axisZ;
    deltaRotationVectorCalibrated[3] = cosThetaOverTwo;

    SensorManager.getRotationMatrixFromVector(deltaRotationMatrixCalibrated, deltaRotationVectorCalibrated);

    currentRotationMatrixCalibrated = matrixMultiplication(
            currentRotationMatrixCalibrated,
            deltaRotationMatrixCalibrated);
```

3. 메시지 처리

-> app>java>pl.radek.panorama>MainActivity

```java
if(ready) {
    //--
    String data = (String) msg.obj;

    String[] raw_text;
    raw_text = data.split( regex: "\\n");

    String[] split_text;
    split_text = data.split( regex: ",");

    String[] temp_text;
    temp_text = split_text[0].split( regex: "\\*");

    panoramaGLSurfaceView.fRoll = 0;
    panoramaGLSurfaceView.fPitch = 0;
    panoramaGLSurfaceView.fYaw = 0;

    if(isStringDouble(temp_text[1])){
        panoramaGLSurfaceView.fRoll = Double.parseDouble(temp_text[1]);
    }

    if(isStringDouble(temp_text[1])){
        panoramaGLSurfaceView.fPitch = Double.parseDouble(split_text[1]);
    }

    if(isStringDouble(temp_text[1])){
        panoramaGLSurfaceView.fYaw = Double.parseDouble(split_text[2]);
    }


    if (bYawDataCompDone == false)
    {
        panoramaGLSurfaceView.finitialYaw = panoramaGLSurfaceView.fYaw;
        bYawDataCompDone = true;
    }

    //// after YAW 보정 후
    //else
    //{
    //    panoramaGLSurfaceView.fYaw = panoramaGLSurfaceView.fYaw + Math.abs(panoramaGLSurfaceView.finitialYaw);
    //    if (panoramaGLSurfaceView.fYaw < 0) panoramaGLSurfaceView.fYaw = 360 + panoramaGLSurfaceView.fYaw;
    //}

    panoramaGLSurfaceView.gtm_sensor_updated = true;

    String SensorData = "[ " + raw_text[0] + "] " + "--->" + " Roll: " + panoramaGLSurfaceView.fRoll + " Pitch: " + panoramaGLSurfaceView.fPitch + " Yaw: " + panoramaGLSurfaceView.fYaw + "\r\n";
    //String SensorData = "--->" + " Roll: " + fRoll + " Pitch: " + fPitch + " Yaw: " + fYaw + "\r\n";
```

[*Roll, Pitich, Yaw] ---> Roll: Pitich: Yaw: