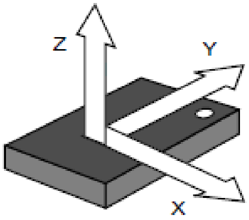
ANDROID SENSOR SYSTEM

Outline

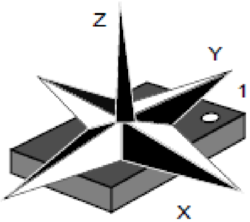
- Sensors in Android
  - SW
    - Android sensor framework and implementation
  - HW
    - 3-axis e-compass hardware introduction

Sensors in Android

- Example
  - Hardware
    - STMicro LSM303DLHC 3-axis e-compass module
      - 3-axis G-sensor (m/s^2) and 3-axis M-sensor (gauss)
      - 3-axis e-compass
        - Using G/M-sensor data to compute heading
  - Software
    - Android 4.0



DIRECTION OF DETECTABLE ACCELERATIONS



DIRECTION OF DETECTABLE MAGNETIC FIELDS

Software Architecture

Java Program

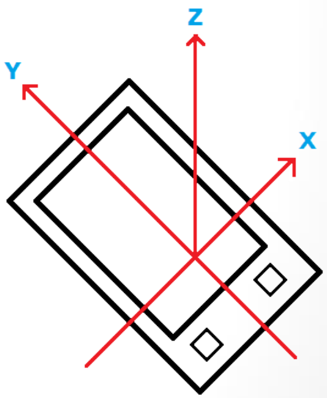
Android Framework

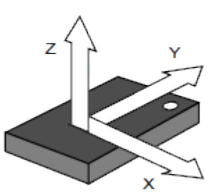
Sensor Library

Linux Driver

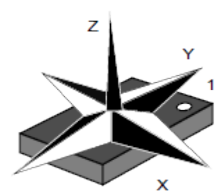
Java App on Android

- Android support several sensor types and provide unified interface to control them.
  - In android 4.0, there are 13 data types.
    - G-sensor, M-sensor, Gyro-sensor, Light sensor, …
  - Android defines several rules for app programmer
    - Data polling rate
      - SENSOR_DELAY_FASTEST (0 ms)
      - SENSOR_DELAY_GAME (20 ms)
      - SENSOR_DELAY_UI (60 ms)
      - SENSOR_DELAY_NORMAL (200 ms)
    - 3-axis dimension
    - etc.





DIRECTION OF DETECTABLE ACCELERATIONS



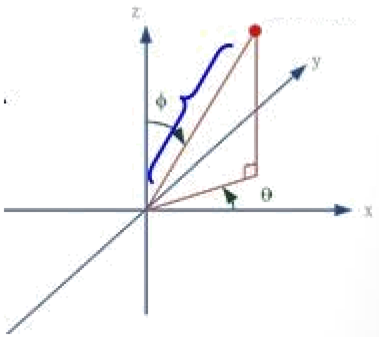
DIRECTION OF DETECTABLE MAGNETIC FIELDS

Java Program

- Classes and interface of android sensor framework
  - Sensor class
    - Instance of a specific sensor
      - Get sensor's capabilities
  - SensorEvent class
    - Instance of sensor event
      - Get raw data
  - SensorManager class
    - Instance of sensor service
      - Register/unregister, access, acquire orientation, …
  - SensorEventListener interface
    - Monitor sensor value/accuracy changed event

Sensors in Android Framework

- Sensors in android framework
  - User registers/unregisters listener for accessing sensor service
  - User proposes the need for data exporting
    - which sensor and data rate
  - User could only get
    - Static
      - Features of sensors
    - Dynamic
      - raw data
      - processed information by android (eg. orientation, …)



Sensor Library in Android

- Sensor library provides necessary callback functions for android to control sensors.
  - Sensor manager in android framework
    - Manages sensor resource for java apps
    - Passes control commands and dispatches data
  - In sensor library, we could implement extra features in Linux user-space.
    - Eg 1. orientation fixing
    - Eg 2. For cost-down, we may use cheap sensor module. However, we could develop some algorithm to improve its performance.

Callback Functions in Sensor Library

- Callback functions should be implemented
  - Open data source
    - Initial the sensor library
  - Close data source
    - Exit the sensor library
  - Activate
    - Start/Stop sensor
  - Set delay
    - Set the time interval of sensing
  - Poll
    - Poll all the sensors to get data
  - Wake
    - Stop sensor polling compulsively

Enter/Exit Sensor Library

Sensor State Setting

Run-time Data Retrieving

Sensor Data Packet

- **sensors_event_t structure**
  - **sensor type**
  - **timestamp**
  - **reserved**
  - **union {**

```
  float data[16];
 sensors_vec_t acceleration; /* (m/s^2) */
 sensors_vec_t magnetic; /* micro-Tesla (uT) */
 sensors_vec_t orientation; /* degrees */
 sensors_vec_t gyro; /* rad/s */
 float temperature; /* Celsius */
 float distance; /* centimeters */
 float light; /* lux */
 float pressure; /* hPa */
 float relative_humidity; /* percent */
}
```

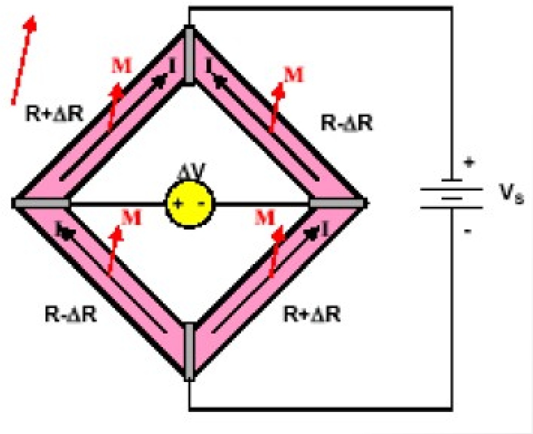
sensors_vec_t : union of (x,y,z), (azimuth, roll, pitch)

LSM303DLHC 3-axis E-Compass

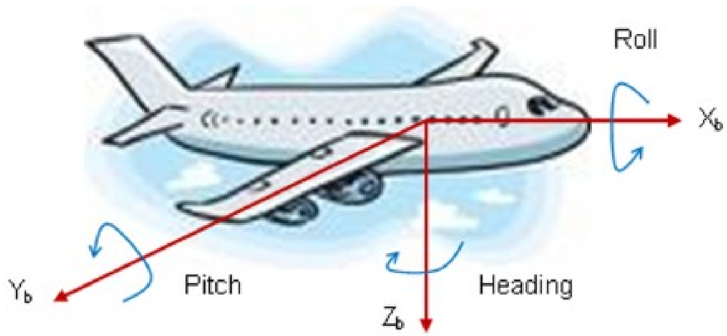
- Control hardware module and get data with I^2C bus.

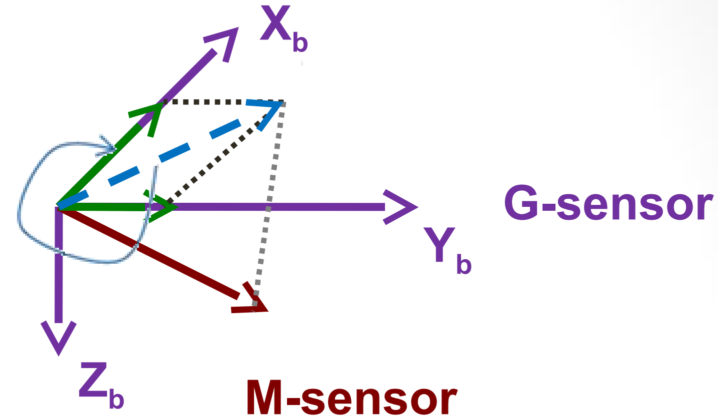
- G-sensor
  - Two interrupt line for special situation
  - Output data rate
  - Output data range

- M-sensor
  - Output data rate
  - Output data range
  - Enable/disable temperature exporting



Heading Computation





$$\text{Pitch} = \rho = \arcsin(-A_{x1})$$

$$\text{Roll} = \gamma = \arcsin(A_{y1}/\cos\rho)$$

$$M_{x2} = M_{x1}\cos\rho + M_{z1}\sin\rho$$

$$M_{y2} = M_{x1}\sin\gamma\sin\rho + M_{y1}\cos\gamma - M_{z1}\sin\gamma\cos\rho$$

$$M_{z2} = -M_{x1}\cos\gamma\sin\rho + M_{y1}\sin\gamma + M_{z1}\cos\gamma\cos\rho$$

$$\begin{aligned}
\text{Heading} = \psi &= \arctan\left(M_{y2}/M_{x2}\right) && \text{for } M_{x2} > 0 \text{ and } M_{y2} >= 0 \\
&= 180^\circ + \arctan\left(M_{y2}/M_{x2}\right) && \text{for } M_{x2} < 0 \\
&= 360^\circ + \arctan\left(M_{y2}/M_{x2}\right) && \text{for } M_{x2} > 0 \text{ and } M_{y2} <= 0 \\
&= 90^\circ && \text{for } M_{x2} = 0 \text{ and } M_{y2} < 0 \\
&= 270^\circ && \text{for } M_{x2} = 0 \text{ and } M_{y2} > 0
\end{aligned}$$

3-axis E-Compass Calibration

- 3-axis e-compass may not get right azimuth data
  - Misalignment
  - Magnetometer
    - Hard-iron distortion
      - It is a constant additive value to the output of each of the magnetometer axes.
    - Soft-iron distortion
      - Soft-iron distortion cannot be compensated with a simple constant; instead, a more complicated procedure is required.
- Calibration sequence is related to hardware

Calibration for LSM303DLHC

- Accelerometer
  - *All ST MEMS accelerometers are factory calibrated, allowing the user to avoid any further calibration for most of the applications now present in the market.*
  - Calibration Matrix :

$$\begin{bmatrix} A_{x1} \\ A_{y1} \\ A_{z1} \end{bmatrix} = [A\_m]_{3x3} \begin{bmatrix} 1/A\_SC_x & 0 & 0 \\ 0 & 1/A\_SC_y & 0 \\ 0 & 0 & 1/A\_SC_z \end{bmatrix} \cdot \begin{bmatrix} A_x - A\_OS_x \\ A_y - A\_OS_y \\ A_z - A\_OS_z \end{bmatrix}$$

$$= \begin{bmatrix} ACC_{11} & ACC_{12} & ACC_{13} \\ ACC_{21} & ACC_{22} & ACC_{23} \\ ACC_{31} & ACC_{32} & ACC_{33} \end{bmatrix} \cdot \begin{bmatrix} A_x \\ A_y \\ A_z \end{bmatrix} + \begin{bmatrix} ACC_{10} \\ ACC_{20} \\ ACC_{30} \end{bmatrix}$$

cal data

raw data