FYS3240/4240

Lab4: **Use of IMU & magnetometer sensors**

Introduction

See the lectures on Arduino Nano 33 and inertial navigation before you start.

Exercise 1 - Understand IMU datasheet & library (No coding required!)

1. Open the IMU datasheet and the file **LSM9DS1.cpp** (IMU library file)
2. Make sure that you see the connection between the data sheet and the code in the library (the big picture).
3. Which hex value (0x) do you need to set in the accelerometer register if you wanted to change the sample rate to 50 Hz and the range to +/- 2 g?
   - Note: 50 Hz to 1 kHz is a common sample rate range for IMUs.
4. If you wanted the **readAcceleration** function to return an integer instead of a float, how could you change the library file?
5. What would be the benefit of sending integers instead of float values over serial communication to the PC?
6. How to send the serial data as binary values (instead of text strings), and how to make sure that the LabVIEW-program receives the measurement data correctly?
   - Se chapter 6 in the Arduino Cookbook.
   - No coding is required!

Exercise 2 – Tilt-sensor using accelerometer

Make an Arduino program that read accelerometer data, calculates the tilt angles roll (φ) and pitch (θ) when the Arduino board has zero or almost zero linear acceleration, and send the calculated roll and pitch angles over serial communication as ascii strings with a tab ('\t') between and '\r\n' at the end (similar to the simpleAccelerometer example program). Open the sketch **simpleAccelerometer**, and save it as **tilt_measurements**.

1. Implement the following equations (from the lectures) on the Nano 33, using the functions atan and atan2 to calculate the pitch and roll angles (see note below):

Pitch angle: $$\tan\theta_{xyz} = \left(\frac{-G_{px}}{\sqrt{G_{py}^2 + G_{pz}^2}}\right)$$

Roll angle: $$\tan\phi_{xyz} = \left(\frac{G_{py}}{G_{pz}}\right)$$

(Note: 3-2-1 = Z-Y-X sequence assumed)

Where $\boldsymbol{G} = -\boldsymbol{f}$. The accelerometer raw output $\boldsymbol{f} = \begin{bmatrix} f_x & f_y & f_z \end{bmatrix}^T$ is negated to give value +1g in any axis aligned with the earth's downward gravitational field.

However, for the Nano 33 we first need to convert the accelerometer measurements from accelerometer sensor frame S to a defined body frame B (see lectures and lab 3), to get a right-handed coordinate system. Therefore, we can set $G_{px} = -f_x$, $G_{py} = -f_y$ and $G_{pz} = f_z$

Note: The equations for roll and pitch have an infinite number of solutions at multiples of 360°. It is therefore a standard convention to restrict the solutions for roll to the range -180° to 180°, and the pitch angle is limited to the range -90° to 90°. This ensures only one unique solution. Therefore, ATAN2 (with output angle range -180° to 180°) and ATAN (with output angle range -90° to 90°) are used.

2. Read AN3461 page 1-12 to see the detailed derivation of these equations and tilt measurements in general.
3. Send only the calculated roll, pitch and yaw angle as ascii text strings on the format 'roll_angle \t pitch_angle \t yaw_angle \r\n' (as before). Since we cannot measure yaw when the board is static you set the yaw angle to zero.
4. Open the LabVIEW program **SerialDataRead_3CH.vi or serial monitor/plotter**, and verify that you get reasonable values for the angles when you tilt the board. If not, correct your code. Not that it is not possible to separate a bias from a tilt unless we know from an external measurement that the sensor/board actually is perfectly leveled.

Exercise 3 – Tilt-compensated magnetic compass

Read chapter 6.16 in the Arduino Cookbook before you start.

1. Open the sketch **simpleMagnetometer** and save it as **tilt_compensated_compass**.
2. On the Nano 33, calculate the heading (yaw) ψ using only the magnetometer data, send the heading angle to the PC and see how the heading is affected if the Nano 33 board is not horizontal.

   $$\psi = \arctan\left(\frac{B_y}{B_x}\right)$$

   Note: In lab 3 we found the transformation matrix R^{BM} that we need to apply to convert the magnetometer measurements from the magnetometer frame M to the same body frame B as the accelerometer:

   $$R^{BM} = \begin{bmatrix} -1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & -1 \end{bmatrix}$$

   This transformation of magnetometer data to the body frame is required before we calculate the heading angle ψ (relative to magnetic north).

3. Read the Honeywell paper "Applications of Magnetic Sensors for Low Cost Compass Systems".
4. Implement the following equations to transform the magnetometer measurements to the horizontal plane (h), based on the tilt angle measurements:

   $$Bh_x = B_x \cos(\theta) + B_y \sin(\phi)\sin(\theta) - B_z \cos(\phi)\sin(\theta)$$
   $$Bh_y = B_y cos(\phi) + B_z sin(\phi)$$

   Note: If you compare with equation 2 in the Honeywell paper, they use the opposite Greek letters for roll and pitch. We follow our notation, which is the most common practice.

   Note: The equations for roll, pitch and yaw have an infinite number of solutions at multiples of 360°. It is therefore a standard convention to restrict the solutions for roll and yaw to the range -180° to 180°, and the pitch angle is limited to the range -90° to 90°. This

ensures only one unique solution. Therefore, ATAN2 (with output angle range -180° to 180°) and ATAN (with output angle range -90° to 90°) are used.

5. Calculate the heading based on these tilt compensated magnetometer measurements (see point 2).
6. Send the heading angle to a computer and verify that you now get improved results when you tilt the board.

Exercise 4 – 2D magnetometer calibration

1. Open the sketch **simpleMagnetometer.**
2. Rotate the magnetometer 360 degrees in the horizontal plane (around the z-axis), and save the data at the same time. You can for instance use the LabVIEW program from Lab3, **SerialDataReadWrite_3CH_parallel.vi,** to save the data to a file.
3. Write a program in your language of choice (Matlab, Python, LabVIEW, …) to plot By vs. Bx (both in uT). Find (graphically) the offsets ΔBx and ΔBy of the "circle" from (0,0). What is the offset values in uT?
    - Hint: If you want to solve this by fitting the data to a circle, you could in LabVIEW search for the example *circle fit*, which is using the function *Fitting on a Sphere.vi* to determine the offset values.
4. Open the sketch **tilt_compensated_compass** and save it as **tilt_compensated_compass_calibrated.** Subtract the calibrated offset values ΔBx and ΔBy from the tilt corrected magnetometer measurements Bhx and Bhy. Did it change the heading result?
5. Add magnetic <u>declination</u> correction to the program in point 4, see chapter 6.16 in the Arduino Cookbook.
6. You now have a tilt compensated electronic compass, giving the direction towards true North. However, in general it is difficult to get a stable and accurate heading, especially indoor. Can you explain why?

Optional (<u>not</u> required!)

To make the heading (yaw) angle more stable we could filter out noise using a <u>low pass filter</u> or use a <u>moving average filter.</u>

- 1. order low pass filter: $y[n] = (1-a)*y[n-1] + a*x[n]$
    - E.g. with a = 0.2
- Moving average filter (e.g. with three points):

$$y[n] = \frac{1}{3}(x[n] + x[n-1] + x[n-2])$$

Where *x[n]* is the measurement at time *n* and *y[n]* is the filter output at time *n*.

What to hand in

- Answers to question 1.3, 1.4, 1.5 and 1.6.
- The program **tilt_measurements** from exercise 2
- The program **tilt_compensated_compass** from exercise 3.
- The program **tilt_compensated_compass_calibrated** from exercise 4.
- The program you wrote to find the offset value, and an image (e.g. a print screen) of the "By vs. Bx circle".

- Answers to question 4.3, 4.4 and 4.6