Angle sensing, measurement and display with Arduino nano BLE sense IMU (accelerometer and gyroscope)

ELEC 391 Electrical Engineering Design Studio II

Let us consider an inverted pendulum (which will become your self-balanced robot later) that we want to balance using a certain control method (e.g. PID). This requires real-time measurements of the deviation angle θ (theta) that the pendulum exhibits in a non-equilibrium state. This assignment is asking you to use both the accelerometer and the gyroscope sensors to measure this angle.

1. Background

The accelerometer and the gyroscope sensors on the Arduino nano BLE sense IMU (**BMI270_BMM150**) provide us two independent methods to compute the angle θ. From a control theory standpoint, the angle θ is the error *e(t)* and is zero at equilibrium (vertical position) as shown in Figure 1. Both methods have advantages and disadvantages. They can be combined to provide a better, more stable measurement than any of them in isolation.

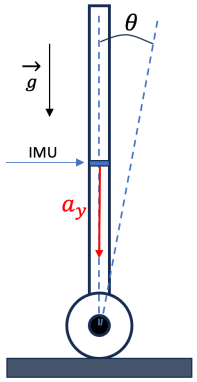

Figure 1 - robot at equilibrium

2. Computing angles with accelerometer readings

An accelerometer will always indicate the gravitational acceleration on its vertical axis, when it is either not moving along that axis, or moving at constant velocity. If the accelerometer's vertical axis is not aligned with the gravitational acceleration (i.e., the angle θ is different from zero), than the angle can be computed as:

$$\theta = arctan\frac{a_x}{a_y} \qquad (1)$$

If the tilt angle is very small, we can approximate the arctan function as a ratio (this is called a small angle approximation), and so:

$$\theta \approx \frac{a_x}{a_y} \quad (2)$$

where ax, ay are the accelerometer readings on the x (horizontal) and y (vertical) axes, respectively, as shown in Figure 2.

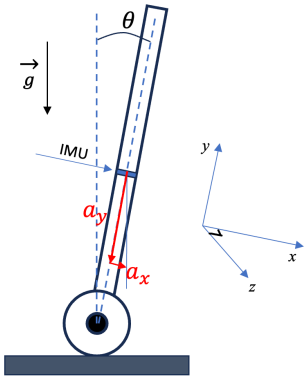

Figure 2 - robot imbalanced at θ angle

2.1 **Advantages of accelerometer reading**:

Because gravity always points in the vertical direction, the reading error will have a zero mean. The accelerometer does not accumulate any measurement bias (which gyroscopes typically do); that means that the measured acceleration along the y (vertical axis) is the true gravitational acceleration plus some zero-mean Gaussian noise.

2.2 **Disadvantages of accelerometer reading**:

The variance of the Gaussian noise can be quite large depending on the specific accelerometer. Other components of the mechanical system (e.g. motors) can also add noise to the readings. You will be able to observe the noise generated by the DC motor when you build your robot and perform readings with the motors turned on (not as part of this assignment).

3. Computing angles with gyroscope readings

The gyroscope measures the rotation rate (a.k.a. angular speed) in degrees/second. This is the same as the first order derivative of the angle with respect to time. So in order to compute the angle of rotation, we need to integrate, and also need to know the reference angle (θ_0 theta zero - integration constant). For a given sampling rate delta t (Δt), the tilt angle can be computed as:

$$\theta = \theta_0 + g_z \cdot \Delta t \qquad (3)$$

where g_z is the gyroscope reading corresponding to the z axis. This is also referred to *pitch* in the context of describing an aeroplane's movement to the air (together with *yaw* and *roll*).

3.1 **Advantages of gyroscope reading**:

The readings from gyroscopes are typically less noisy than those of the accelerometer, by a significant amount. This is true for short periods of time.

3.2 **Disadvantages of gyroscope reading**:

Due to the continuous integration, gyroscope measurements accumulate bias in time. This bias is driven by a random process and can offset the real measurement by either a positive or negative amount.

4. Complementary filter

We would like to combine the positive aspects of both sensor readings to achieve a more precise measurement. A complementary filter can achieve the accurate short-term reading of the gyroscope and the long-term stability of the accelerometer.

To implement a complementary filter for our angle measurement:

- Sample both gyroscope and accelerometer data
- Calculate the angle using accelerometer data
- Calculate the change in angle using gyro data
- Compute the complementary angle estimation by performing a weighted average of the accelerometer angle and the sum of the previous complementary angle estimate and the change in angle measured by gyroscope.

Choosing the complementary filter weights is part of tuning your system, and is done by repeated experiments. The greater the weight on the accelerometer, the *less bias* and *more noise* you see in your angle estimates. Start with an accelerometer weight of ~0.01 and a gyro weight of ~0.99. You can initialize the complementary angle using the accelerometer angle.
The output of the complementary filter can be expressed as:

$$\theta_n = k\left(\theta_{n-1} + \theta_{g,n}\right) + (1-k)\theta_{a,n} \qquad (4)$$

where k is a number from 0 to 1, θ_n is the filter output at discrete time n; θ_{n-1} is the output of the complementary filter at the previous discrete time n-1; $\theta_{g,n}$ is the current angle computed based on gyro reading; $\theta_{a,n}$ is the current angle computed based on accelerometer reading.

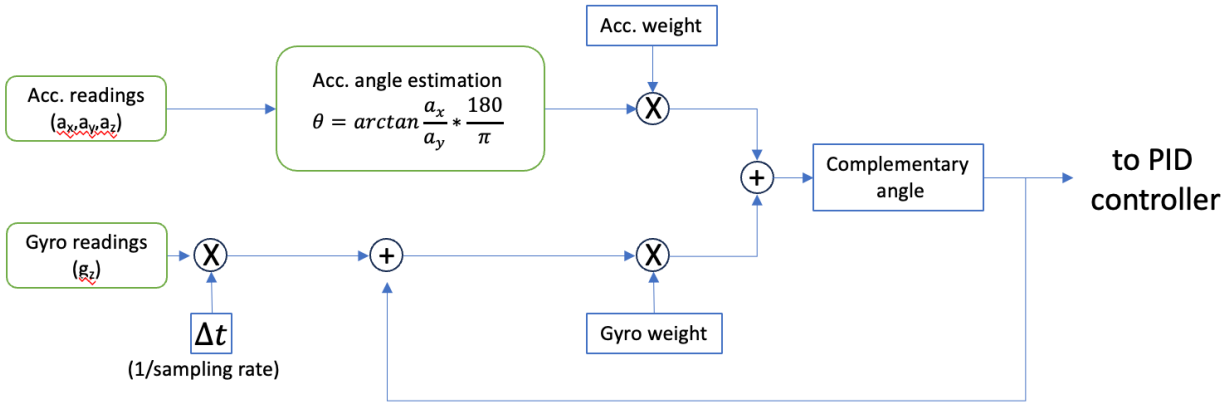

Figure 3 - Algorithm flow for acc/gyro complementary filter

Task 1 – Data Plotting [1 mark]

In order to check your results, validate your design choices, and debug your design, you will need a way to plot your sensor readings in a graphical manner. Your task 1 is to figure out a way to plot data in real time from an Arduino system (in particular the Nano 33 BLE sense) on

a PC or Mac screen. As a real-time sensor data source, you can use any of the sensors built-into the Arduino board, or random readings from any of the analog input pins. The built-in Serial Plotter of the Arduino IDE is **not** an acceptable solution. You should not use any external circuitry (resistors, potentiometers, etc.)
Hint: Python is one of the most useful languages for data manipulation and plotting. If you do not know Python, it is fine to use a LLM such as ChatGPT and direct it to write your code. If you use LLMs, submit your prompt(s) with this assignment, not only your code.
Grading: demonstrate your real-time data plot to the TA, answer TA's questions.
Deliverables:

- task1.ino (Arduino code)
- task1.py (Python code)

Task 2 – Computing angles with accelerometer readings [1 mark]
Write Arduino code to collect accelerometer data from the BMI270_BMM150 IMU and compute the tilt angle of the sensor. Display this angle value using the plotting method you developed for task 1. Angles must be calculated exactly as indicated earlier in this handout, no extra processing (i.e. filtering) is allowed.
Grading: demonstrate your tilt angle calculation and plotting to the TA, answer TA's questions.

Task 3 – Computing angles with gyro readings [1 mark]
Identical to task 3, but use gyroscope readings and calculations.
Grading: demonstrate your tilt angle calculation and plotting to the TA, answer TA's questions.

Task 4 – Calculate angles using a complementary filter [2 marks]
Using the complementary filter method, compute the tilt angle. You will have to experiment and tune the value of the K coefficient to obtain a stable and accurate reading. On the same plot, show the three datasets corresponding to the three methods, simultaneously.
Grading: demonstrate your complementary filter. Explain your tuning strategy for the k coefficient to the TA.

For all tasks, demonstrate your angle readings at three different angles: 0° (vertical), ±15°, and ±30°.

Submission: upload your code for all four tasks as required by the Canvas assignment. Your submission must be a single .zip archive that includes two files for each task, for a total of eight files. For each task you must submit an Arduino program called taskx.ino, and a Python script called taskx.py, where *x* is the task number. The name of the .zip archive must be group_yy.zip, where *yy* is your group number.