Last modification: November 6, 2024

RBE 2002 (B-24)
Lab 3 Activities
IMU and Filtering

Be sure to follow along with the Worksheet at the end of this document.

You will work in pairs to complete the activities and Worksheet. Do not take a 'divide-and-conquer' approach – it is important that each student masters the material.

In this lab you will explore the IMU, in particular the accelerometer and its limitations. You'll build a *complementary filter* that *fuses* readings from both the gyroscope (explored last week) and the accelerometer to take advantage of the strengths of each. The end result will be a filter that can reliably detect the pitch angle of the Romi as it drives up a slope.

Your work will culminate with using the filter to detect the ramp and platform.

Background

A reference document that recaps the material from lecture has been placed on Canvas.

Lab Activities

1 IMU Fundamentals

You have already explored the gyroscope in the previous lab. Here, you will go a little deeper into how the IMU works, focusing on the accelerometer.

1.1 Basic functionality

Procedure

1. Edit your code from last week to read and print out the raw accelerometer values. You might want to comment out the gyroscope outputs to make the data easier to read. Alternatively, you can edit your code so you can use the Teleplot extension in VScode. Whatever you decide, be sure to enclose all the print statements in `#ifdef __IMU_DEBUG__...#endif` statements so you can turn off the printing easily later.

2. Set your Romi on the table and observe the data from the accelerometer. Are the accelerometer values zero? What is the value of the accelerometer reading in the z-axis? How many g does this correspond to? **For the purpose of this exercise (and the datasheet) *g* refers to *gravities*, not grams!**
3. Do some experiments to determine/verify which direction the axes point. A good experiment is to accelerate the Romi in various directions and observe the output. Do the accelerometer axes coincide with the gyroscope from last week? Record your findings in Table 1.
4. With the Romi sitting on the workbench in its normal position, record the accelerometer reading for each axis in the first row of Table 2 (where it says, "Top facing up"). Round your answers in integer values of g. That is, you should enter $1\mathrm{g}$, $0\mathrm{g}$, or $-1\mathrm{g}$ in each square.
5. Without moving the Romi yet, work with your partner to make *predictions* for the readings on each axis if you were to point the front straight up. Record your predictions in the second row of Table 2 ("Front facing up").
6. After you have made your predictions, set the Romi on its end so the front is pointing straight up. Was your prediction correct? *Do not go back and change your answers!* That's not the point of this exercise, and you will *not* lose points if your predictions were incorrect.
7. Repeat the process for each of the positions in the table. Each time, make a prediction and then check your results. Discuss the answers with your partner to make sure you understand the results.

1.2 Settings and Calibration

Last week, you explored the output data rate (ODR) and the sensitivity for the gyroscope. While the ODR for the gyroscope and accelerometer can be different, for this lab, you'll set them to the same rate. If the rates are different, then you have to adjust the complementary filter steps to allow multiple prediction steps for each observation. The filter is actually more efficient in that case, but harder to code up. Another benefit is that if you set the two to run at the same rate, then testing the `STATUS` register is easier.

If you want to run the gyroscope and accelerometer at different rates, talk to your instructor.

Procedure

1. Set the accelerometer ODR to be the same as the gyroscope. See `LSM6::enableDefaults()` for the syntax and available data rates.
2. In your IMU library, add code to set pin `13` to HIGH right before the Romi reads the IMU data and then LOW immediately afterwards (same as last week).
3. Attach one channel of an oscilloscope to pin `13` and the other to the clock pin of the I2C bus (labelled `SCL` on the Romi pinout). Run the program and determine how long it takes for the IMU to take a reading.
4. Zoom the oscilloscope in and determine the clock rate on the `CLK` line. We'll talk more about I2C later.

5. Consult Table 16 (p. 38) in the datasheet to determine how many *bytes* of information have to be read from the IMU (all three axes of both the accelerometer and gyroscope). Considering that it takes one clock cycle on pin `SCL` to transfer one *bit*, how does the length of the read operation compare to the time needed transfer all the data?

6. Use the oscilloscope to verify that the accelerometer is reading at the ODR you have specified.

Show your results to the course staff for an electronic sign-off. Be ready to show your oscilloscope capture and explain your findings to course staff.

You wrote code last week to estimate the bias in the gyroscope readings; here you will do the same with the accelerometer.

Procedure

1. Update the code from last week to calculate the bias from the x and y axes of the accelerometer as running averages. Why don't we ask you to do the same on the z axis? You will probably need to add a variable to track the bias.

2. While you're at it, comment out the line where you estimate the bias for the y axis of the gyroscope. Later in this document, you will estimate the bias as part of the complementary filter, and while removing it at the start will help, we don't want to "mask" the results later.

1.3 Angle calculations

Here, you will calculate the pitch angle of the Romi using just the accelerometer readings.

Note: There are several options for where to manage and store the Romi orientation. We could have made `eulerAngles` a member of the `LSM6` class, which makes a lot of sense since the math is all being done with IMU readings. However, if you wanted to fuse the data from the IMU with other data (odometry, camera, etc.), then it might make more sense to store that information in class `Robot` or even class `Chassis`. For better or worse, we have chosen to store the data in `Robot`.

Procedure

1. If you did not do so above, remove bias filter for the y-axis of the gyroscope. We'll show that you can solve for bias as part of the filter.

2. In `Robot::HandleOrientationUpdate()`, add code to calculate the pitch angle based purely on the accelerometer readings. Assign the result to a *local* variable, `observedPitchAngle`.

3. Test your code to be sure you have the correct functionality. Is the timing correct? Is there noticeable noise? What happens if you gently tap the Romi? Do the angle readings jump? Why?

2 Sensor fusion

Here, you will build the functionality to perform sensor fusion, specifically, you'll implement a complementary filter to estimate the pitch angle.

2.1 The Complementary Filter

When you call `HandleOrientationUpdate()`, `eulerAngles.y` will contain the pitch angle from the previous iteration (e.g., $\hat{\theta}_y^{k-1}$).

Procedure

1. Calculate the predicted angle from the gyroscope readings in the same way that you calculated heading update last week (you may have done that already).Assign that calculation to a *local* variable, `predictedAngle`.
2. Add code to calculate the estimated (filtered) angle. Store that in `eulerAngles`, since you'll need it for the next iteration.

   Warning: Keeping track of your units and signs is paramount to a properly working filter!
3. Add code to print the three pitch angles: predicted, observed, and filtered. If you're using Teleplot, then you should also print `millis()` (it will ruin the Serial Plotter output if using Arduino, since it `millis()` will quickly wash out the other values).

   Pro tip: Enclose the print statements in compiler directives:

   ```
   #ifdef __DEBUG_IMU__
   ...
   #endif
   ```
4. Run the code with $\kappa = 1$ and plot the angles. Are you now relying the accelerometer or the gyroscope? How does the "filtered" angle (which isn't really being filtered at all) compare to the observed angle from the accelerometer?
5. Run the code with $\kappa = 0$. Which value are you using now? Set the Romi on your desk and don't move it. What happens to the "filtered" angle?
6. Set $\kappa = 0.5$ and run your program again. Move the Romi through pitching motions (but not too fast); gently tap the Romi. How does it do?
7. Experiment with different values of κ. Do you think it should tend towards 0 or 1? What did you find worked well?
8. Now pitch the Romi quickly. Does your filter keep up? You may want to experiment with the FS and ODR values.

Show your results to the course staff for an electronic sign-off. You will be expected to show a Serial Plotter output while you perform some of the experiments above. The SA will ask you about your parameters (κ, etc.) and how they affect the performance. They may also ask you questions related to the "What about...?" questions in the handout.

2.2 Dealing with bias

As discussed above, both the accelerometer and gyroscope are prone to an *offset*, or *bias*. You have seen methods for dealing with bias, but here you'll solve for the bias as part of the filter. That is, we can make an estimate of the gyroscope bias as *part of the solution*, so instead of calibrating it out (and having it change anyway), we can just determine the bias on the fly.

Procedure

1. Add code to implement the bias update. Start with $\epsilon = 0.01$.

   Warning: The magnitude of ϵ will depend heavily on how you manage the unit conversions. If you follow the equations from lecture, it will be much easier for course staff to help you, if needed.

2. Add code to print the bias, in addition to the angles that are already being printed.

3. Run your code. How does the bias change over time? What happens if you make $\epsilon = 1$? Can you explain what happens?

2.3 Troubleshooting guide

There are a number of things that can cause poor performance (or complete failure) of your filter. Some of the common ones are listed below, with some trouble shooting hints.

Not filtering out noise well? Try adjusting κ. Which do you want to favor more, the accelerometer or the gyroscope?

Not responsive enough? Try adjusting κ, but also increase the gyroscope full-scale, since it may be saturating.

Filtered angle goes the opposite way before it settles down? Classic sign error on your axes.

Gross over- or under-shoot? Check your scale factors. Be sure to convert properly between radians and degrees and include the timestep from the ODR.

The filtered angle oscillates slowly? The phenomenon shows up when the bias update is poorly scaled.

3 Challenge

Here you will program your Romi to drive up a slope and stop when it reaches level ground. You will use your complementary filter to detect the slope and flat part. Before you start coding, create a state transition diagram for the following behaviour:

- When a button on the remote control (or the Romi itself) is pressed (your choice), the robot starts to line follow towards the ramp,
- When it detects that it is going up the ramp, an LED will light, and
- When it reaches the top of the ramp (which will tip down), the Romi stops and the LED is turned off.

You will need to add a flag to indicate if you're on the ramp or not. What threshold(s) will you use for detecting the ramp? Do you think hysteresis could be useful here?

4 Extra Challenges

to go beyond expectations, consider attempting one or more of the following:

- Instead of stopping when it reaches level ground, program the robot to drive to the edge of the platform and turn around (to simulate dropping off the garbage). You could either command the wheels to move at the same speed or, for an additional challenge, use the gyroscope to follow a straight heading.
- Instead of just driving straight until your robot sees the ramp, command it to drive to a point not on a straight line to find it.
- Other challenges as you might see fit. Please discuss them with course staff before you embark on them.

Worksheet

5 Worksheet

Submit one Worksheet per pair.

Question

What IMU chip is used on the Romi? Who manufactures it?

LSM6DS33

iNEMO

Question

Nominally, what is the largest acceleration you can register with the IMU?

+/- 16G

Question

What is the default output data rate and full-scale range for the accelerometer?

13 Hz data rate

0.061 milli-gravities per least significant bit

Question

What is the default sensitivity for the accelerometer?

0.061 milli-gravities per least significant bit

Question

Be sure to complete Tables 1 an 2. Do the axes coincide?

	accelerometer	gyroscope (from last week)
x	Forward is positive	Roll to the right is positive
y	Left is positive	Pitch forward is positive
z	Positive is up	Yaw Counterclockwise is positive

Table 1: Table for noting the direction of each axis.

Orientation	a_x	a_y	a_z
Top facing up	535	75	16.7k
Front facing up	16.4k	432	1.1k
Left side facing up	835	16.2k	20
Right side facing up	269	-16.5k	909
Upside down	-305	-177	-15.9k
Rear facing up	-16.1k	-371	459

Table 2: Accelerometer readings for each orientation.

Question

How long does it take to read the IMU? Given the length of the reading, what is the maximum rate you could read the IMU?

2.5 milliseconds
400 Hz

Question

If all the IMU had to do was transfer data bits (that is, ignore overhead), how long would it take to send the necessary data with the given I2C clock speed (one bit per clock cycle)? How does that compare to the actual time needed?

2.0 milliseconds
960 Hz

Reading without overhead allows for faster IMU data aquisition

Question

What ODR did you settle on? Explain.

We settled on the ODR of 104, as it performed the best and allowed the romi to accurately measure the pitch of the robot

Question

When calculating the angle from the accelerometer only, describe the basic behavior of your angle calculator. Does is get the angle correct over a complete 360? How noisy is it (qualitatively)?

The angle is wrapped between -180 and 180 degrees, and it is very noisy.
When we tapped on the romi, the noise can rach upwards of 5 degrees.

Question

What happens with $\kappa = 0$? What about $\kappa = 1$?

When Kappa is 0, the filter only uses the gyro - this means it is very sensitive to bias.
When Kappa is 1, the filter only uses the accelerometer - this means it is very sensitive to noise.

Question

What was your final value for κ? How does your filter perform, qualitatively?

The final value for Kappa was 0.01, which allows the complimentary filter to filter out noise from the accelerometer, while also making it a little less susceptible to gyro bias.

Question

What did you choose for the `ODR` setting? Why did you choose that rate?

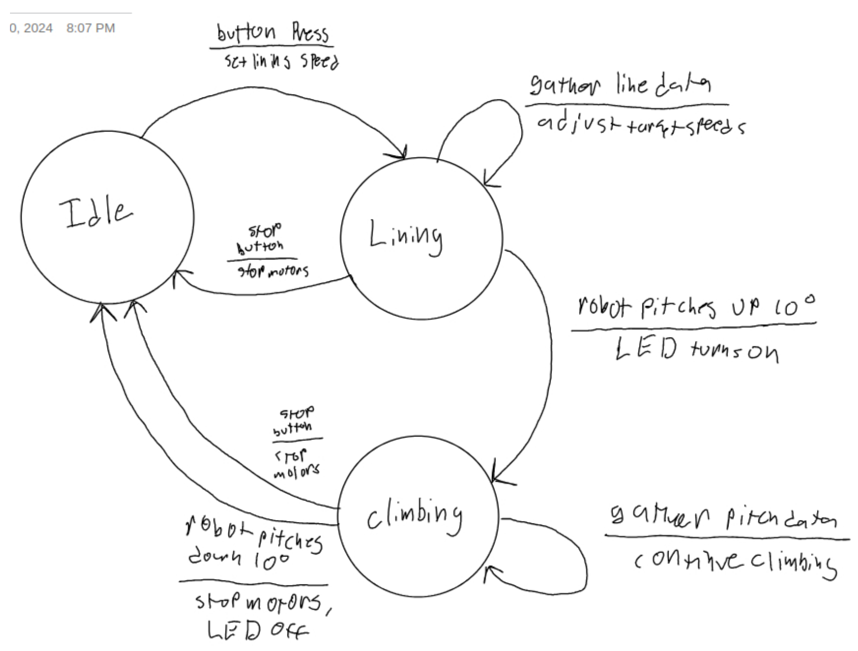
Question

What did you choose for the `FS` setting for the gyroscope? Why did you choose that value?

We chose 500 for the FS setting for the gyroscope, because it provides a decently high sensitivity while not impacting the resolution it can read too heavily.

Question

Neatly draw out your state machine for the challenge in Section 3.



Question

Create releases of your code. Tag and title them `week03-firstname-lastname` and paste clickable links here.

Question

Take a video of one of your Romis completing the challenge. Paste a clickable link here.

6 Wrapping up

Question

Team Member Contributions

1. Independently, describe how you contributed to the design tasks for this week's activities.

   Student A ______________________

   Student B ______________________

2. For the core activities, assign a weight (as a percentage) to each team member's contribution. The core activities will be worth a total of 90 points. To receive higher marks, you will need to complete one or more extra activities.

Student	Percentage

3. If either of you went beyond expectations for this lab, describe any extra work that you did. There is no requirement that both students work on the same activities, so be sure to identify who did what. Include evidence, either as a video or with code snippets or both.