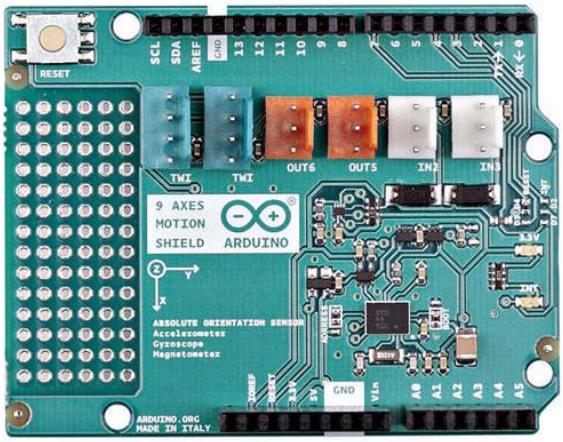
Robotics Design Project: Lab Session 4

Ungraded formative exercise

Introduction

In this session you will use information from the nine-axis sensor shield to control how your robot behaves. The goal of this lab session is to have a robot capable of a meaningful navigation task, fusing both distance and orientation data. The nine axis sensor is a intricate shield with rich functionality, and teams are encouraged to explore this for themselves. The official *GitHub* is a useful resource: https://github.com/arduino-org/arduino-library-nine-axes-motion

Task 1: Sensing orientation



From Brightspace, download and run the example sketch ‘session4.ino’ This simple programme should continually display the Euler angles and the linear acceleration measured by the sensor. The measured angles should correspond to the below aviation diagram, where the terms *heading* and *yaw* are equivalent:

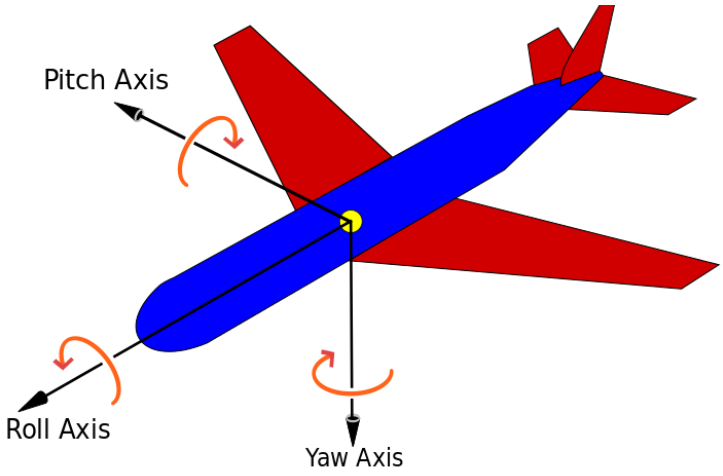
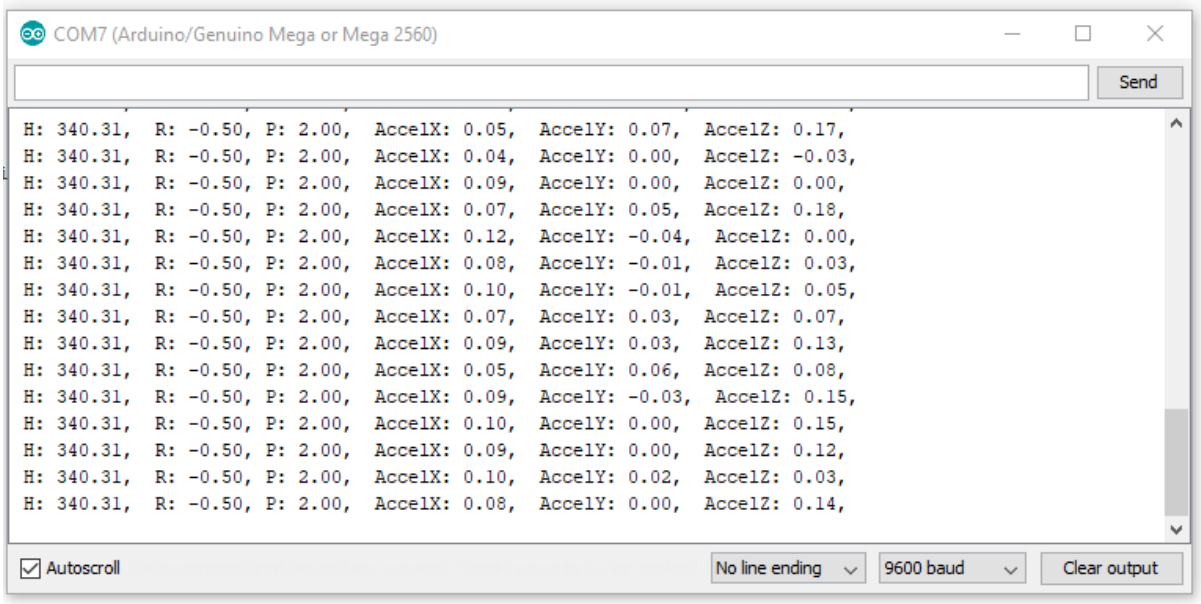


Image credit: *Auswise* and *Jrzw*

Use the provided sketch to verify that your nine axis sensor shield is working as expected. The sketch outputs data both to the LCD screen and via the serial port: use the serial monitor functionality (CTRL+SHIFT+M) to see how this works. You should see something like the below:



Once you are satisfied that you understand the data that this shield provides, you need to write a programme that implements *controlled steering* of your robot, as below:

- This programme should rotate your robot in 45° increments
- It should initially start with the motors off, waiting for the push of a white button
- When a button is pressed, the robot should rotate itself through precisely 45° in a controlled fashion (hint: a suitable while loop could be useful here)
- Once it has subtended 45° it should halt, and wait for another press of a white button, whereupon it should repeat the manoeuvre
- Be careful about the accumulation of error: the robot should take due account of its measured starting heading before commencing the rotation
- For reasons I've never been able to fully diagnose, there needs to be around a 25 ms delay between interactions with the nine-axis sensor and the LCD screen. You can see this little hacky workaround in the example sketch.

Creating a controlled steering function is a programming task of moderate complexity, and so it would be a good idea to sketch out a flow chart, or some pseudo-code, before writing actual Arduino commands.

One tricky point you'll need to handle is the abrupt change that occurs in heading when rotating incrementally from 359.9° to 0°. This discontinuity means that simple *greater than* or *less than* comparisons of heading values may not always be insightful.

Perhaps you could write functions that tell you:

- What heading is *n*° clockwise of my current orientation?
- Is the heading *x*° clockwise of the heading *y*°?

Having dependable, tested functions of this sort prepared should equip you well to write a while loop that rotates your robot to a specified heading. Also, it is often easier and more efficient to work with integers rather than floating point values – not only does this free up memory and allow faster calculations, it also permits operations like modulo division. So working with an integer number of centi-degrees might be a good approach here.

Challenge 4

In this challenge, your robot needs to complete laps of the test table, without hitting the walls. This challenge could form the basis of a competitive navigation strategy. You will need a facility to measure how far you have travelled, as developed in lab session 3.

- Your robot should start in a corner of the table, facing towards the opposite end
- When a button is pressed, it should drive straight down the longside of the table, reasonably close to the wall
- As it drives, your robot should continually measure out the distance travelled, so that it can stop after around 2400 mm, which is the length of the table
- Once it reaches the far end of the table, it should halt and complete a controlled 90° turn, and then traverse the short side of the table, a length of ~1220 mm.
- When it returns to its initial position, it should stop and await a button press before repeating the lap

This challenge will require a sketch of reasonable complexity. Some hints to help with this are below:

- Defining suitable functions will make your code more re-usable and easier to read. For instance, it would be helpful to have a function that can translate a raw number of sensed axle rotations into a mm value:

```
float TurnsToDistance(int AxleTurns)
{
return (AxleTurns * 50.5); //Assume wheel circumference is 50.5 mm
}
```

- A function that rotates the robot through a specified angle would be very useful
- As would a function that drives over a specified distance (if your robot has a tendency to veer or drift, this function could also use the nine axis sensor data to maintain a consistent heading)
- For a clean programme, the content of your `void loop()` would be mainly calls to these neatly packaged functions: you want to avoid wall-of-text spaghetti code.

Deliverable

This is an ungraded, formative exercise so no deliverables are due. Indeed, a full implementation of the functionality requested here may require multiple lab sessions of work to complete. Be sure to save whatever code you create in a durable way: team file lockers are available on Brightspace for this purpose.

Tidying

Before you leave the laboratory, please put all your Lego parts and your robot into the large plastic box provided. Please leave the robot accessible, with nothing on top of it, so that we can get access to it to charge its battery.