Reykjavík University

Mechatronics I

T-411-MECH

Individual Assignment 6

14/09/2020

Students:
Giacomo Menchi

Teacher:
Joseph T. Foley

Contents

1 Tasks

As usual, the code for this assignment is available at the following Github repo (though this time, it just contains a Hello World in Rust and an Arduino sketch for task 5):

`https://shorturl.at/hyWZ5`

1.1 Task 1

Task: List the pros and cons for every motor in your kit

Answer: In my kit there are three different motors: the servo, the stepper and the DC motor.

- The **servo motor** is very precise in position (angle) and velocity (rotation), is usually low power and can be electric, pneumatic or hydraulic: it has to tolerate sudden velocity changes while still remaining accurate and is commonly used for mechanical arms, cranes and other similar machines. Its caveats are that it is difficult to configure (specially using PWM) and it has to implement a closed loop mechanism, thus it always has to receive feedback about its current position to work correctly.
- The **stepper motor** is easier to configure than the previous one and turns itself by using a great number of steps, thus dividing its rotation in a series of discrete angles: this allows it to avoid using a feedback mechanism to check back if the position is accurate, differently from the other motors. On the other hand, they also produce a lot of heat, turn slow, and vibrate at every step. They are used in CDs, DVDs, and 3D printers mainly.
- The **DC motor** is a direct current machine which is way less accurate than the others (so it's not suggested when precision is needed) and is commonly used if I just want to keep a fast and stable rotation. It can also be used as an electricity generator (dynamo) and it's usually implemented in vehicles, fans and other similar systems.

1.2 Task 2

Task: You need a motor to move an arm to a precise position(+- 0.1°) the arm has a 280° of motion, which motor do you pick? Explain your choice briefly.

Answer: In order to move an arm with the requested features we could choose between a servo and a stepper motor: they usually have almost the same guaranteed precision (+- 0.1°), which is achieved in two different ways (as already explained in the previous task) and they could both reach the 280° motion, though only some kinds of servo motors can go beyond 180° (and those which do, are not that accurate).

In this case, therefore, the stepper motor could be preferred to the servo, even though the latter is usually suggested when working with mechanical arms.

1.3 Task 3

Task: You need a motor for wheels of a car, there is a lot of hills and irregularities. Which motor do you pick? Explain why this one.

Answer: The motor that is most suited for a car wheel is the DC one, since it provides a fast rotation and is able to keep its speed, without concerning about the accuracy. It is controlled via pulse width modulation (PWM) which sends brief electrical impulses to repeatedly turn the motor on and off, and since these pulses are very fast, the motor appears as if it were continuously spinning (I can raise the impulses "on" period, thus lowering the "off" one, which makes the motor turn faster, or vice versa). It wouldn't be suitable to use another type of motor since the others are better when needing to have lower speed and precision.

1.4 Task 4

Task: You have a low voltage application that is supposed to lock a deadlock using minimal electricity. Which motor do you pick?

Answer: If working with low voltages and minimal electricity, the servo motor is usually better to choose compared to the others, since its linear

actuators only consume energy when a motion has to be executed, and almost no energy when idle (ideal if we're talking about locking a deadlock), while the stepper motor always consumes the same amount of energy, even if no rotation is being executed (it's a huge waste of energy), and the DC motor is not suitable for the task (there is no continuous rotation with speed to keep).

1.5 Task 5

Task: Create a code to move a stepper motor 90° forward stop for a second then go 90° back.

Video: After a while, I managed to achieve this goal by creating an Arduino sketch which controls the stepper motor, here is the demonstration (the code is on the repo, under the Task_5 folder).

`https://shorturl.at/klvEY`

1.6 Task 6

Task: Read on PID Controller. Imagine your own space device that would have a PID loop. Describe how and why would PID be used there.

Answer: A PID Controller is a system which continuously checks if the device output is in the correct range, and in case it isn't, it applies corrections until the result is the expected one (this is done by using the mathematical formula below, in which the three P, I and D parameters are adjusted until the function has the right shape).

$$K_p e + K_d \frac{de}{dt} + K_i \int_0^t e(t)\,dt$$

Figure 1: PID formula.

In a spaceship, a PID loop would be necessary to automatically adjust the route every time it gets out of the desired bounds (a human would not

be able to do this, since the destination could be light years distant and a computer would be more efficient in calculations).

In order to do this, the direction (route) should be the input of our PID controller, it should then pass on to the controller itself and to the actuator, thus coming back to the beginning of the PID as a direction feedback: this feedback would be analyzed using the PID formula above (I is the parameter which is connected to past errors, thus to feedbacks) and adjusted accordingly in order to try fixing the previously identified error(s) (the schema and a more in-depth explanation of what I just wrote can be found in the following task).

1.7 Task 7

Task: Draw PID Controller diagram for the space device from the task 6. Specify which parameter of the device is controlled with PID and what's used for getting a feedback.

Answer: As specified before, the parameter controlled with PID is the spaceship direction (in this case described as coordinates X, Y and Z of the destination) and the feedback is the difference between the real coordinates and the actuated ones (ΔX = realX - actuatedX, same for ΔY and ΔZ).

Once we have the coordinates error in the feedback we can adjust those coordinates as explained before (and also as illustrated in the PID Controller diagram below) until the result is satisfactory.

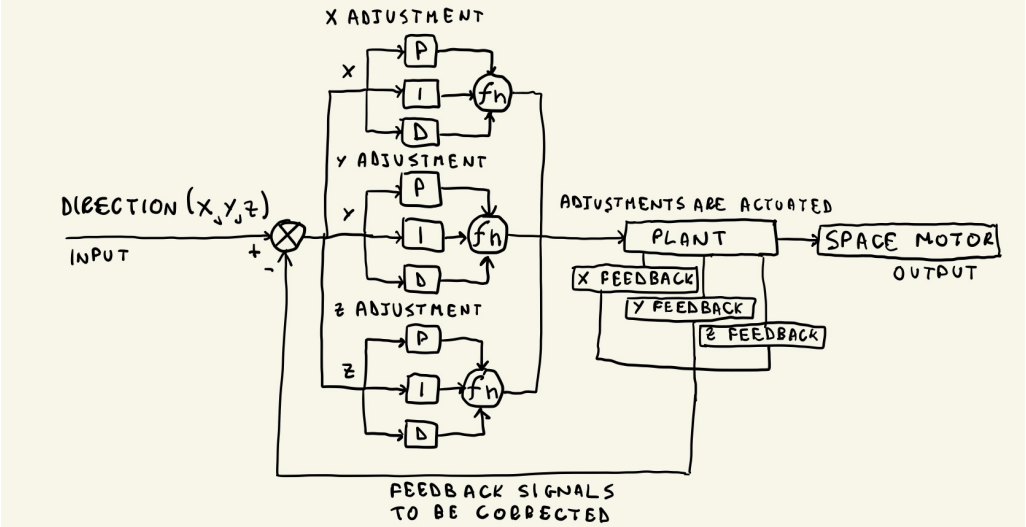

Figure 2: PID formula.