Automatique Numerique

Report

Erasmus student

Dmytro Savchuk

for

3A IA2R

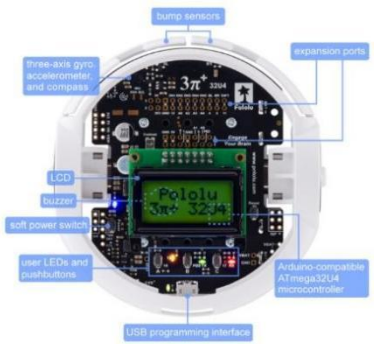
15/04/2022

Content

I - Introduction 3
II - Simple digital corrector 4
II.a. - Strategy explanation 4
II.b. - Short explanation of results 5
III - Proportional corrector with derivative action 5
III.a. - Setting up the corrector and testing on the track 5
III.a. - Short explanation of results 6
IV -Additional settings 6
V - Conclusion 7

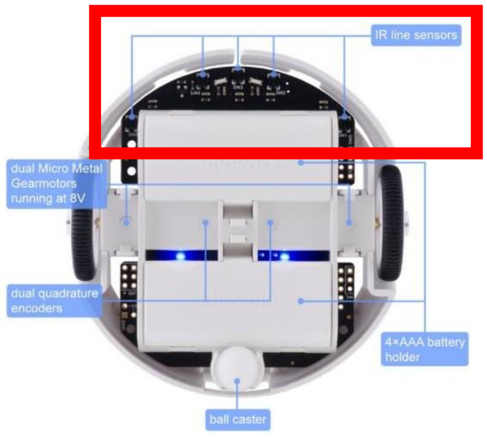
I - Introduction

The robot used in this tutorial is the same one used on the 5A of the first semester (PID controller for high-speed line-following robots). It is the 3pi+ 32U4 robot from Pololu which is a complete and powerful mobile platform based on an Arduino compatible ATmega32U4 microcontroller. This robot is easily programmed with the Arduino IDE via a micro-USB cable.


bump sensors
expansion ports
three-axis gyro, accelerometer, and compass
LCD
buzzer
soft power switch
user LEDs and pushbuttons
Arduino-compatible ATmega32U4 microcontroller
USB programming interface

In terms of sensors, the robot contains five sensors. The five line sensors on the front of the robot help the 3ft+ robot distinguish a non-reflective (black) line from a reflective (white) surface.

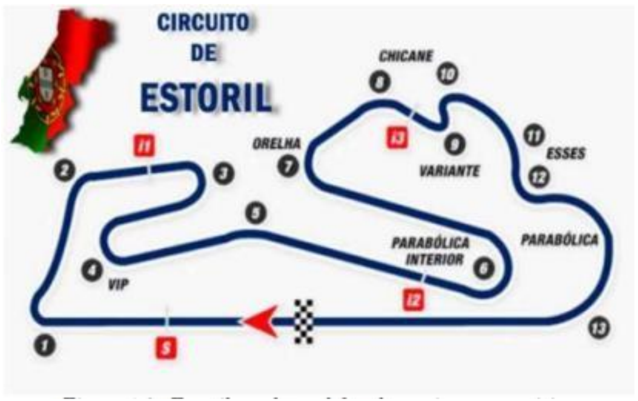
Each reflectance sensor consists of a downward-facing analogue infrared (IR) emitter LED (Light-Emitting Diode) combined with a phototransistor capable of detecting the infrared light reflected from the LED. Three of the five sensors are placed in the middle and the last two are located at the end of the two sides



Regarding the actuators, the robot is driven by two motors coupled to its two wheels, one on the right and one on the left.

The robot moves straight ahead when both wheels are rotating at the same speed. The robot moves to the right or to the left if the speed of one of its motors turns at a lower speed than the second.

The objective of this laboratory work is to create a control system that allows the robot to drive around the F1 race track in Estoril, Portugal as fast as a professional driver would. (as fast as possible)



II - Simple digital corrector

II.a. - Strategy explanation

The first strategy tested is to use a simple P controller that will allow the robot to follow the black line on the track with a nominal speed that must be chosen before the circuit is run and that must allow the robot to move forward in complete stability without causing it to abandon the path.

The robot operates with the help of sensors that are oriented towards the bottom of the robot and have a higher value on a black surface but a lower value on a white surface, which allows it to detect and follow the black line on the white track. These sensors send a robot position value of 2000 to the middle of the black line.

So the usefulness of this controller is the elimination of the error picked up by these sensors and which can cause the robot to be unstable during its operation.

In order to use this type of controller we need to choose a value for the gain Kp. Below is a table that summarizes our study with the proportional controller and represents the different values of Kp chosen with their results.

II.b. - Short explanation of results

Looking at the results obtained with a simple digital proportional controller, we can say that the best values that allowed the 3pi+32U4 robot to run the circuit in a stable way and as fast as possible are 100 for the nominal speed and 15 for the Kp gain since the duration of a turn was 22.8s which is the lowest compared to the other tests.

In addition, we notice that the addition of the sampling period did not serve to keep the stability of our robot. Indeed, when we increase the sampling period, we go from a "continuous" system to a sampled system. Thus the measurements of the robot's position on the line are made at regular intervals, and the greater the interval, the less likely it is that the robot will be able to maintain itself on the line.

With the delay at 100 the robot is just going out, this is normal as the robot is slower to correct. In real life, you need a delay that is low enough, but not too low, so as not to consume too much power. In this case, the delay value should be low enough, otherwise it is immediately unstable.

So we can say that the use of the P corrector is very limited and in this situation it is interesting to choose a more precise and efficient corrector. This is why we need to make a study with a PD controller.

III - Proportional corrector with derivative action

III.a. - Setting up the corrector and testing on the track

The second strategy is therefore to use a proportional corrector with derivative action. This new corrector will make it possible to reduce the instability of the simple proportional corrector.

A few tests allow us to obtain this series of results :

Speed	Algorithm	KP	KD	Time/turns (s)
120	PD	13	0	21.43
100	PD	13	130	19.43
180	PD	13	170	FAIL
130	PD	13	170	15.43

III.a. - Short explanation of results

Initially, it can be seen that the robot's time/revolution has improved in cases where the settings allow it to stay on track. We can also note that we left the sampling period at 0 to have the fastest possible correction.

The first tests are used to adjust the PD corrector in order to obtain the best ratio between the proportional and derivative actions in order to have an optimal correction (to avoid that the robot loses speed or goes out of the track because of jerks). Finally, we can see that the best time/lap without exit is achieved at the speed of 110 because when this speed is exceeded with this type of corrector, the tracking of the track by the robot is very much a matter of luck. We therefore need to find another type of correction to apply in order to increase the speed.

IV - Additional settings

When the limits of the PD corrector are reached, it is possible to think of adapting the speed of the robot according to the obstacles it will encounter. Indeed, it is interesting that the robot takes speed in straight lines and slows down when it arrives on a curve, like a real car.

The aim is therefore to adapt the robot's speed in an automated way. The pair was not necessarily very competent in Arduino programming, the choice was made not to create an array containing the values of the different speed levels that the robot can reach and then to use it in a for or while loop etc. even if in the end this solution can be very effective.

V - Conclusion

To conclude, the "basic" or at least frequently used correctors such as the P and PD correctors quickly show their limits when the path to be followed becomes more complicated and the speed increases. For example, even with a PD corrector, the robot will quickly tend to go off the track or to jerk too much when only 25% of the maximum speed of the robot is exceeded. Moreover, once the 50% is exceeded there is no chance that the robot will pass certain turns.

To counter this problem, the most obvious solution is to adapt the speed of the robot to its environment (straight lines, hairpins, etc.). Another parameter that could be taken into account if the robot were equipped with suitable sensors is the distance to the next obstacle. This would allow a broader view of the terrain and to know at what point to slow down depending on the obstacle the robot detects (wide bend or hairpin bends for example).

Finally, there are also more precise correctors such as the PID, but on a course as difficult as the one in the test, speed variation remains essential to achieve the best time without going off the track.