Project report:
Speed control system

Made by:
Bachar BARIDI – ESA
Badr Aldeen AL-KEBSI – Mécatronique
Aly HJEIJ - Mécatronique

Supervisors:

Frédric Antoni	Freddy Anstotz
Florent Nageotte	François Stock
François Schwartz	Denis Muller
Siham Touchal	Patrick Knobloch

Date:
2020/2021

Acknowledgements

We would like to thank the people who helped or advised us throughout this project.
First, we would like to thank the supervisors who were present during the various sessions and who were very helpful and attentive, but who were also available to us outside of these sessions. We would also like to thank our various teachers who may not have been present, but who were able to guide us when we needed them, and who answered our questions with enthusiasm.

Abstract

In this paper we are going to try to control the velocity of a car model (mini-car). We are starting by showing the design of the system through a block diagram. Then, we are going to explain the procedure and work that went into realizing this project step by step. After that, a short explanation of the code and how it works. Finally, we are going to discuss the problems we were faced with and what solutions we came up with and end the paper with a conclusion and presenting a reflection that each member had during the work on this project.

L3 Projet : Sujet 3

Table of contents

1. Introduction:

• Presentation:

Development of automobiles started from the early 17th century that were steam-powered cars.[1] These cars were very basic, and their systems were purely mechanical, and no electronical systems were introduced at the time. The first ever electronical application in automobiles were the vacuum tube car radios at the early 1930s.[2] And after that the development continued until electronics made around 30% of the value of a car in 2010.[3] Nowadays, there are many control systems in the car such as Fuel injection systems, knock control, chassis and wheel acceleration control and so on.

Speed control was used in the early 1900s and continued having many applications such as maintaining speed whether uphill or down or having a speed limiter and many other applications.

Our research and project presses on the subject of speed control and how to maintain it as a constant value and so our objectives and implementations are as follows:

• Objectives:

In this project, our main objective is to take a scale model and control its velocity so that it remains constant even in the event of external disturbances. These disturbances might be a change in the road, or a simple friction introduced by the user.

We will be using materials provided by the university that already exists in its labs and that takes away the difficulty of choice of materials.

• Realisation/implementation:

The way we are going to realize this objective is by first commanding the motor with a certain voltage. This voltage needs to be a variable that we can control its value manually. being able to control the amount of voltage going into the motor means that we can control its velocity.

After this, we are going to need to be able to read the real velocity of the tire commanded by the motor. This should be realized using an optical sensor that can send a signal of the velocity of the tire.

Lastly, we will need to take this velocity and send it to a code that will be able to readjust the input velocity according to the output velocity.

All these elements are going to be coded using the Arduino Uno card to program the needed functions.

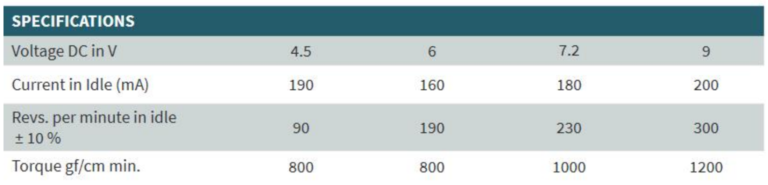
[1] (Chariot à vapeur du R. P. Verbiest , 1679-1681)
[2] (History of the Car Radio in Motor Cars, 2012-12-09)
[3] (statista, 2017)

2. Materials:

• Arduino Uno:

The main processing unit and the programmable card that is going to allow us to control different signals like the input voltage of the motors. It also allows us to read the signal from the optical sensors to measure the speed of the motors. It also allows us to integrate and complete the electrical circuit and more that will be shown later in this project.

• Car model / motors:



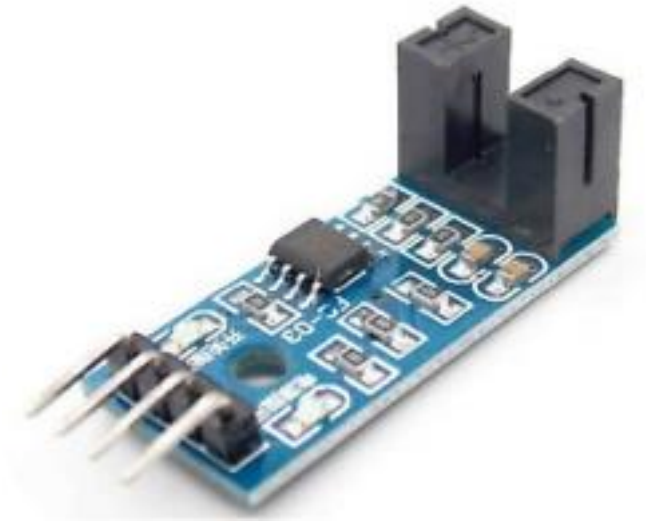
SPECIFICATIONS				
Voltage DC in V	4.5	6	7.2	9
Current in Idle (mA)	190	160	180	200
Revs. per minute in idle ± 10 %	90	190	230	300
Torque gf/cm min.	800	800	1000	1200

Figure 1 - Datasheet COM-Motor01 made by Joy-it

The car model was provided by the university and it consisted of the car body connected to 4 motors that control 4 wheels. These motors are DC motors with a supply of 3-9 V the specifications can be shown in the Figure 1 below.

- Sensors:

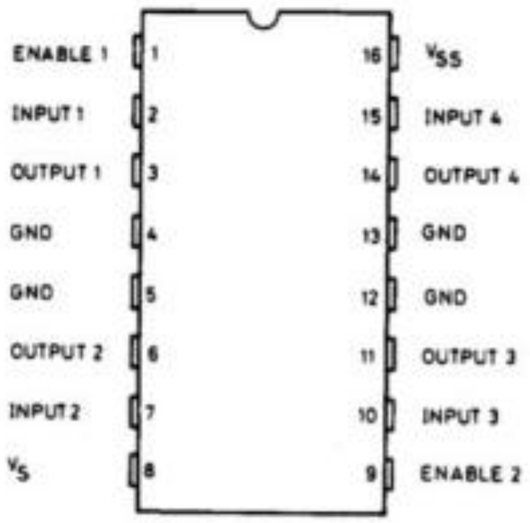
The car model was also mounted with 4 speed sensors, their type is LM393 and they measure speed using a rotation axe. It indicates the output status and has a fixed bolt hole for easy mounting on the wheel. The signal output is digital. Look at annexes for further specifications.



- L293D circuit:

The L293 device is a quadruple high-current half-H driver. It is designed to provide bidirectional drive currents of up to 600-mA at voltages from 4.5 V to 36 V.[4]

The reason we are using this device is to control the input voltage of the motors by connecting a controllable variable through the Arduino Card to one of the EN inputs and through this input, we can specify the voltage we need to enter the motor.



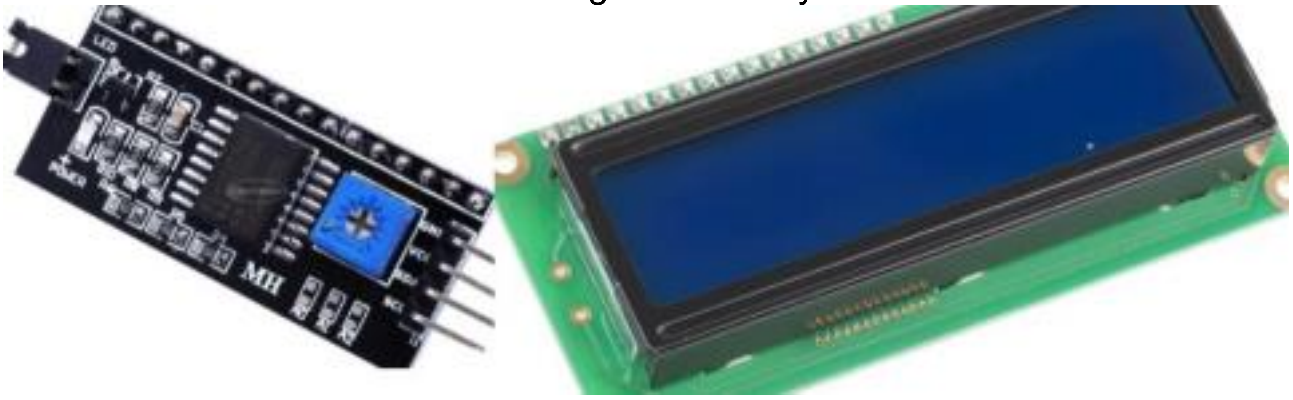
[4] (Instruments, Sept. 1986 [Revised Jan. 2016])

- Batteries 9V:

These batteries are used to supply the motors (they get connected to the L293D circuit that is connected to the motors) and also to supply the Arduino Card.

- LCD:

We chose an Lcd screen 16x2 to display the speed of the motor in rotation per minute and the static error. However, an Lcd screen alone proved to be very difficult to connect to the Arduino card (there were at least 10 wires to connect) so we also bought an I2C module. Its purpose was to connect the Lcd screen to the Arduino card using 4 wires only



3. Block Diagram:

• Open loop:

The Figure 2 shows the open loop of the system.

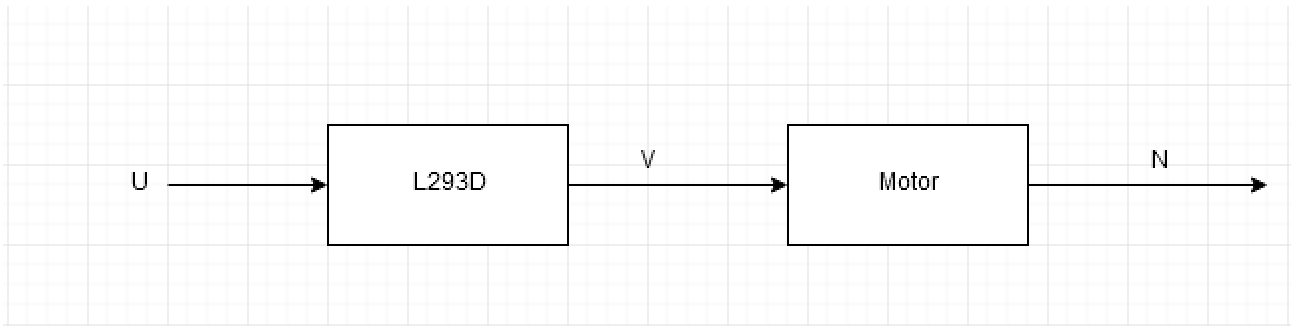


Figure 2 - Open loop of the system

U: Voltage supplied by the battery 9 Volt.

V: Output voltage from the L293D device where its value has been regulated by an amount of α where:
$\alpha \in [0,1]$

$$V = \alpha . U \qquad \textit{Eq. 1}$$

N: value of the angular velocity of the motor/wheel expressed as rotation per minute (rt/min)

• Closed loop:

To make a control system, it is required to make a closed loop where we take the output value N=Y and compare it to the desired value. The difference between these two values will be called the error. This error will be taken into a PID corrector that's going to try to eliminate this error in the most efficient way possible.

The Figure 3 shows the closed loop of the system and how to realize it.

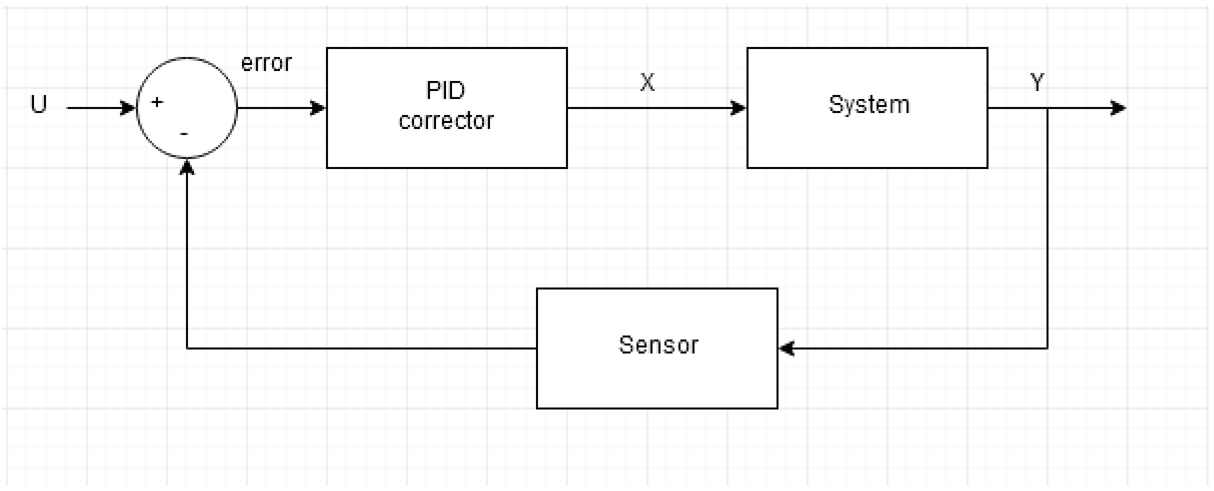


Figure 3 - Closed loop of the system

4. Procedure:

- Commanding the motors:

To command the motors, we had first to decide whether we are going to command all 4 of the motors or only 2 of them. This decision was delayed until after we realized that the circuit L293D can only command 2 motors. Now, if we had decided to command all 4 of them, we would have needed 2 different L293D devices so to save materials and to economize, we finally decided to only command 2 motors one on the left side and another on the right side.

At the beginning though, we only commanded one motor to understand how to do so using the L293D device and the Arduino. We looked up online tutorials and videos to have an idea and we started working on it until we were finally able to code a simple program that can control the input voltage of the motor and thus controlling its velocity.

From Eq.1 we know that the value of α is a percentage value.

This value can be coded in the Arduino Card as a value between 0 and 255 where is 0 is 0% and 255 is 100% the value that is between 0 and 255 is called "duty_cycle".

$$duty\_cycle = \alpha * 255 \Rightarrow \alpha = \frac{duty\_cycle}{255}$$

$$V = \alpha * U$$

$$\Rightarrow V = \frac{duty\_cycle * U}{255} \qquad Eq.\ 2$$

$$\Rightarrow duty_cycle = \frac{V}{U} * 255$$

GRAPHE HERE AND CIRCUIT

Interrupting sensor’s signal:

The sensor is made up of two parts: one with an LED and the other with a photoresistor. When the photoresistor receives light from the LED, it sends a 0 signal; when this light is cut off (by a wall), the sensor sends a 1.

The velocity of a given wheel is calculated by using a mini wheel with holes that rotates between the two parts of the sensor. The frequency of the sensor's signal is also the frequency of the wheel's rotation.

A simple code was implemented where every time the sensor sends a 1, a counter goes up. This function lasts a second, after which it calculates the frequency is easily obtained.[5]

PID control:

We will take the output velocity from the sensor and compare it to our reference value (desired value). The difference between these two values will be referred to as the error (ε). Now, to control this error value, we will employ the following correctors:

I. Proportional correction: This type of correction helps makes the system faster through reducing the rise time and the error but the bigger the value of this correction overshoot starts to appear and becomes problematic.

II. Integral correction: This component sums the error term over time and as a result, it drives the steady state error to zero. Although this component can eliminate the steady state error, it can strongly contribute to controller output overshoot. The solution to this is to add a derivative component.

III. Derivative correction: acts on the rate of change of the error signal. The more error changes or the longer the derivative time, the larger the derivative factor becomes. The effect of this is to counteract the overshoot caused by P and I.

The equation used for these correctors is:

$$C = K_P(\varepsilon) + K_I\left(\sum \varepsilon\right) + K_D(\Delta\varepsilon) \qquad \textit{Eq. 3}$$

Remark: The values of K_P, K_I, and K_D were calculated experimentally rather than mathematically.[6]

[5] Refer to “velocity calculation” in Coding for the equation used.
[6] Refer to “Problems and solutions” in Difficulties and Conclusion for more explanation.

- LCD: TODO
- FIXING: TODO

5. Final circuit:

HOW THE FINAL CIRCUIT LOOKED LIKE

6. Coding:

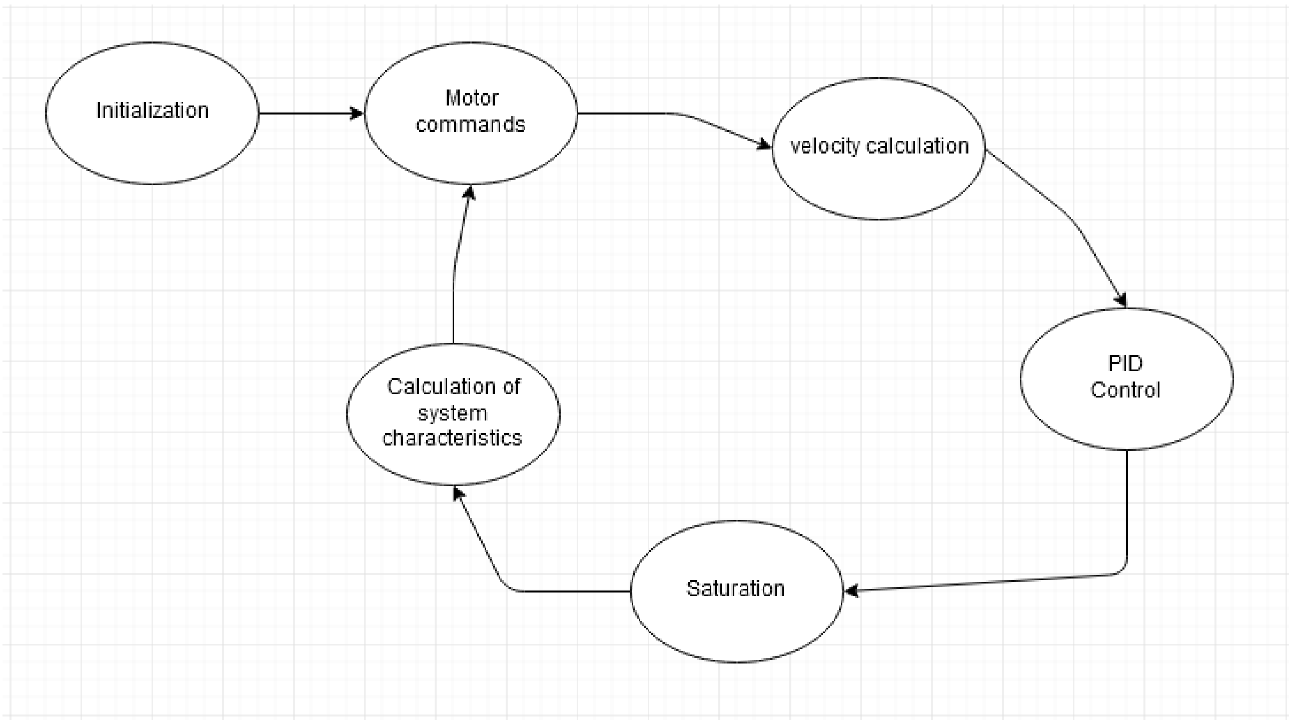


Figure 4 - Diagram explaining code's different function.

Figure 4 depicts how the Arduino Card was programmed. Each cercle corresponds to a function or a section of code that serves a specific purpose. The following is an explanation for each of these:

I. Initialization: This section of the code is where we declare the various variables we are using and assign pins to the Arduino as either input values or outputs for the purposes of either commanding the motors or reading sensor signals. It serves as the starting point for the loop that follows.

II. Motor commands: This is the section where we control the voltage input to the motors. We initially supply these motors with 50% of the supply voltage and then allow this value to be corrected in other iterations.

III. Velocity calculation: A function that takes the sensor's output signal and converts it to a velocity value using the equation: $N = \frac{n}{20} * 60$ (rot/min)
Where N: rotation per minute (Angular velocity) of the wheel.
n: number of signals sent by sensor per second.

IV. PID control: A function to apply the corrector PID as shown in the equation Eq.3

V. Saturation/Limiter: A basic function that limits the value of the correction so that the input voltage stays between 0 and 9V.

VI. System parameters: An experimental section that calculates the parameters of the system such as rise time, overshoot and settling time. These values are useful when optimizing the values of PID corrector. After this section, the program loops back and starts applying the values of the correction into the motors.

7. Difficulties and Conclusion:

- Problems and solutions:

a) Motors and Sensors:

The first problem we were presented with was the fact that we had 4 different motors and 4 different sensors. If we supplied each motor and read its sensor separately, we would run into complex mechanical problems such as balance.

Solutions:

To make things easier we decided to only control two motors and read the velocity value of only 1 motor. This way is the simplest and less expensive way to solving this problematic.

While this solution sounds simple and nice, we run into other problems where if a wheel started slowing down while the others are still having the same velocity due to road problems (holes/mud) then the car will get stuck. We decided that this problem should be solved manually due to its niche situation.

b) Coding the sensors (Velocity calculation):

While this part appeared simple and straight forward when we investigated at first, it proved to be way more difficult. At first, we thought about making a simple counter that anytime the sensor sends a high voltage value (1) this counter goes up. The problem that arises with this method is the fact that we did not know the inside clock of the Arduino Card. It took us about 3 weeks and multiple researches online until we found some functions that worked. The problem now was that we did not fully grasp how these functions worked to understand how to take the value of the frequency and convert it into velocity.

Solutions:

At the end, we had to use a custom function that has a delay inside of it so that the function works until there is no signal to be read (signal value = 0). This function still posed a problem for us. The problem now was when sometimes the wheel would stop due to friction or huge values of corrections, the program would sometimes stop working because it got stuck inside this function (since the wheel stopped while the sensor is sending a 0 signal)

This problem was finally solved by adding an addition condition where if the program got stuck in this function for a certain time, it forces itself out.

c) PID control and values of K_P, K_I and K_D:

When we wanted to implement the PID correction, we originally wanted to calculate the values of K_P, K_I and K_D mathematically rather than relying on experimental values since they would more precise and would work as demanded. When finding these values mathematically, a step response of the system on open loop is needed. The problem is that we were not able to have a proper step-response for this system.

The reason for not being able to find this step-response was because the motor's response system was so fast that we could not make a graph of it (the time for it to go from 0 to N was less than 50ms.) We were not able to make this graph because when we tried to make it (having the program function in less than 50ms) the sensor would start sending more signals than anticipated. In short, the sensor was not sensitive enough for us to make a step-response.

This problem is probably caused by the fact that the inside clock of the Arduino Card does not have a high enough frequency.

Solutions:

We decided that PID corrector should be measured experimentally by the method of trial and error.

Another thing we decided to do is to drop the derivative term in this corrector. The derivative term is a term that is heavily dependant on the step-response graph and should be mathematically calculated. It is way harder to try to make it work experimentally.

The reason we decided to drop the derivative term is because this term usually controls the stability of the system. It removes or controls the high values of overshoot and the oscillation that appear because of the proportional term and the integral one. Since our system is stable in its nature and since we don't mind it having overshoot values (getting the velocity past the desired value) this makes the derivative term the least important one.

- General conclusion: TODO
- Reflect (EACH OF US): TODO
- Possible continuation of the project: TODO

8. References: TODO

9. Annexes: TODO