Enviado: 15/02/2020 *Recibido: 06/03/2020* *Aceptado: 23/04/2020*

UNIVERSIDAD DISTRITAL FRANCISCO JOSE DE CALDAS

VISIÓN ELECTRÓNICA

Algo más que un estado sólido

https://doi.org/10.14483/issn.2248-4728

VISIÓN ELECTRÓNICA

A CASE-STUDY VISION

PID-fuzzy of DC motors using Raspberry PI

PID-difuso de motores DC utilizando Raspberry PI

Sebastián Herrera-Aristizábal[1], Julio Alejandro Hincapié-Correa[2], Luis Hernando Ríos-González[3], Sebastián López Flórez[4]

[1] BSc. in Electrical Engineering, Universidad Tecnológica de Pereira, Colombia. E-mail: seherrera@utp.edu.co ORCID: https://orcid.org/0000-0001-6429-857X

[2] BSc. in Electrical Engineering, Universidad Tecnológica de Pereira, Colombia. ORCID: https://orcid.org/0000-0001-9611-0527

[3] BSc. in Electronic Engineering, Instituto Politécnico de Vladimir, Rusia. MSc. in Electrical Engineering, Universidad Tecnológica de Pereira, Colombia. E-mail: lhgonza@utp.edu.co ORCID: https://orcid.org/0000-0002-5565-0030

[4] BSc. in Mechatronic Engineering, Universidad Tecnológica de Pereira, Colombia. E-mail: sebastianlopezflorez@utp.edu.co ORCID: https://orcid.org/0000-0002-8890-9184

Cite this article as: S. Herrera-Aristizábal, J. A. Hincapié-Correa, L. H. Ríos-González and S. López-Flórez, "PID-fuzzy of DC motors using Raspberry PI", *Visión Electrónica*, vol. 14, no. 1, January 2020.

Abstract

This article describes the implementation of a fuzzy and a PID control system for the velocity and torque of a DC power motor, based in the variation of the armor current and its duty cycle. The control system has been applied to a medium power DC motor, using Python, and a low-cost embedded system - Raspberry Pi, this work is fundamental given the importance of implementing velocity and torque controllers that yield energy optimization and correct operation being achieved through the regulation of the duty cycle applied to the power of the DC motor.

Keywords: duty cycle, PID, PWM, Python, Raspberry.

Resumen

Este Articulo describe la implementación de un sistema de control difuso (FLC) y un controlador convencional proporcional-integral-derivativo (PID) para la velocidad y torque de un motor DC de potencia media, basado en la variación de corriente de armadura y en el control del ciclo de trabajo. El sistema de control ha sido aplicado a un motor DC de Potencia media, utilizando Python, y un sistema embebido de bajo costo - Raspberry Pi, dicho trabajo es fundamental dada la importancia de implementar controladores de velocidad y torque que permitirán la optimización de la energía para su correcto funcionamiento lográndose a través de la regulación del ciclo de trabajo aplicado a la potencia del motor DC.

Palabras clave: ciclo de trabajo, PID, PWM, Python, Raspberry.

1. Introduction

The DC Motor of medium power has been widely used in industrial and domestic processes such as electric wheelchairs, mobile robots and others applications. Many applications require precise speed control. The parameters of Motor DC changes in each moment of time. These

variations are due to inaccuracies in the detection of current, temperature increase and changes in operating conditions, as well as to the errors of some sensor.

In recent years new and different control techniques have been studying in order to improve the speed regulation of the DC Motor. In [1] the authors apply a diffuse gain tuner and a traditional PID (Proportional- Integral- Derivative) control technique to the control of a motor; in [2] a PID technique is used, where the authors include software and firmware elements applied to Xilinx and FPGAs to control a DC motor; in [3] a control is implemented by Pulse Width Modulation (PWM) using microcontrollers and implementing the entire closed loop feedback control circuit.

In this work has been addressed the problem of speed and torque control of DC Motor, implementing a hybrid model PID/ Fuzzy, using a low cost embedded system [4] and an efficient control technique, which is determined by comparing classic and modern control actions [1-3], [5].

2. Model of the motor DC

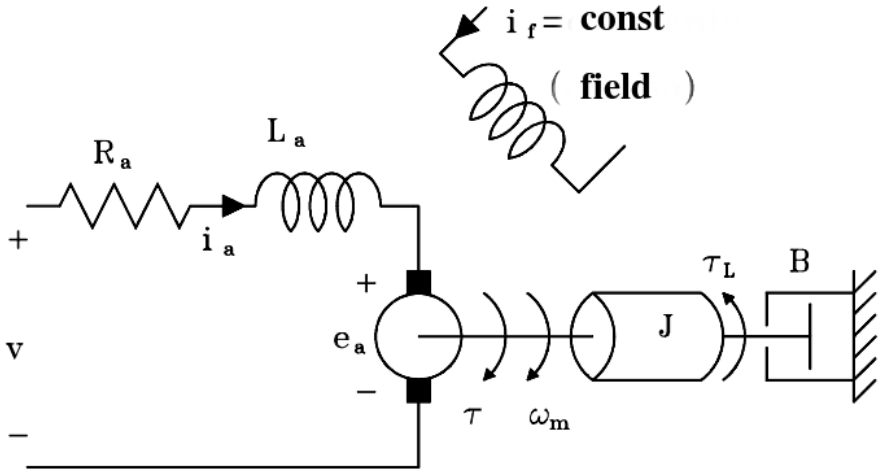
By relation of the electrical and mechanical equations, the DC motor model, was obtained, which the input is the applied voltage and the output is the rotational speed of the axis.

For this, it is necessary to know the follow parameters:

- Inertial Moment of the rotor (J).
- Damping coefficient of the mechanical system (B).
- Electromotive force constant $K=K_e=K_t$
- Armature resistance (R_a).
- Armature inductance (L_a).
- Input (v): Voltage source.
- Output (w_m): Rotational axis speed.

The basic scheme of model of the DC motor is shown in Figure 1.

Figure 1. Model of DC Motor. [6]



The differential equations of this system are:

$$K_e = K_a \phi \tag{1}$$

$$e_a = K_e w_m \tag{2}$$

$$K_t = K_b \phi \tag{3}$$

$$T = K_t i_a \tag{4}$$

w_m – Rotational axis speed.

ϕ – magnetic Fluxe.

T – Generated Torque.

Kirchhoff's second law

$$v - e_a = R_a i_a + L_a \frac{di_a}{dt} \tag{5}$$

Newton's second law

$$T - T_l = J \frac{dw_m}{dt} + B w_m \tag{6}$$

Transforming to the Laplace (7) y (8)

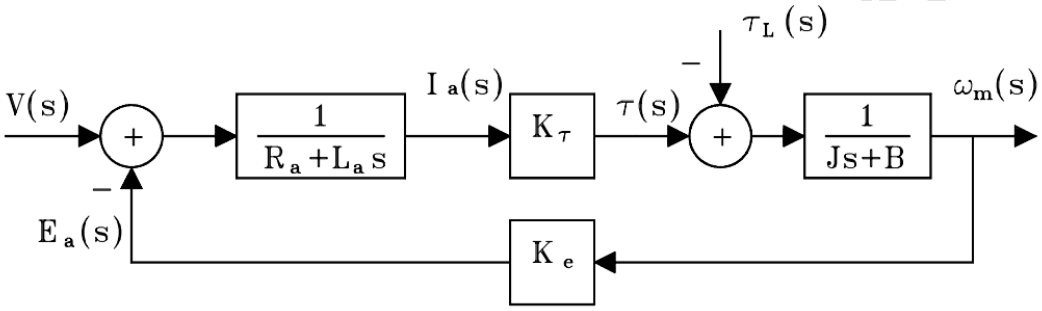
$$I_a(S) = \frac{V(S) - E_a(S)}{L_a S + R_a} \tag{7}$$

$$W_m(S) = \frac{T(S) - T_l(S)}{JS + B} \tag{8}$$

Replacing (2) in (7)

$$I_a(S) = \frac{V(S) - K_e W_m(S)}{L_a S + R_a} \tag{9}$$

Figure 2. Block Diagram of a Plant [7].



From the equations (7), (8) and (9) is obtained the block diagram representing the DC motor controlled in the armature (Figure 2) [7].

$$G_s = \frac{\dfrac{K_t}{L_a J S^2 + (L_a B + R_a J) S + R_a B}}{1 + \dfrac{K_t K_e}{L_a J S^2 + (L_a B + R_a J) S + R_a B}}$$

$$G_s = \frac{K_t}{L_a J S^2 + (L_a B + R_a J) S + R_a B + K_t K_e} \tag{10}$$

3. Parameter measurement

To measure the parameters shown in Equation (10), we proceed as follows:

3.1. Armor resistance

To calculate the resistance, the motor was fed with a low magnitude voltage and just before it started to work, were taken different voltage and current measurements by varying the position

of the rotor; The armor resistance calculation was then performed using ohm's law and averaging the results (Table 1) [8].

Table 1. Armor Resistance. [9]

MOTOR 180W		
E2 (V)	**I2 (A)**	**Ra (Ω)**
0,595	0,253	2.352
0,608	0,251	2.422
0,624	0,246	2.537
0,637	0,244	2.611
0,655	0,236	2.775
0,667	0,235	2.838
0,667	0,234	2.850
0,719	0,221	3.253
0,725	0,218	3.326
0,746	0,215	3.470
0,758	0,216	3.509
0,781	0,208	3.755
0,818	0,199	4.111
0,833	0,192	4.339
0,869	0,185	0.470
1,080	0,132	8.182
		3.564

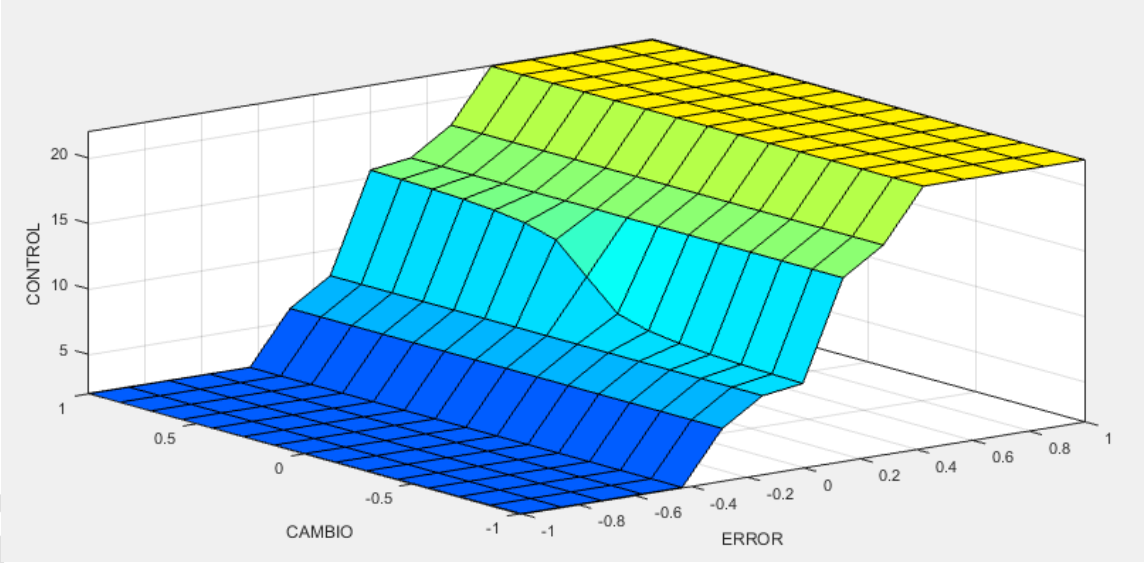
3.2. Armor Inductance

For the calculation of the inductance, the motor was fed with an AC source and varied taking into account that the current limits of the motor were not exceeded. The voltage and current readings were taken; The Impedance calculation was performed using ohm's law and averaging the results (Table 2) [8].

Table 2. Armor Inductance. [8]

MOTOR 180W		
E2 (V)	**I2 (A)**	**Z (Ω)**
0,993	0,422	2,35308057
1,69	0,855	1,97660819
2,09	1,09	1,91743119
2,26	1,86	1,21505376
2,762	2,265	1,21942605
2,932	3,194	0,9179712
2,604	2,942	0,88511217
3,086	3,427	0,90049606
3,354	3,518	0,9533826
3,731	3,647	1,02303263
		1,33615944

Figure 3. Armor Inductance. [8]



For the calculation of L (Armature inductance), with the equation $L = \frac{\sqrt{z^2 - R^2}}{2\pi f}$, we Obtained a negative inductance value, showing that the inductance value was too small to be calculated by means of this test (Figure 3). For the correct calculation of the Armature Inductance a parameter meter was used, obtaining the following result:

3.3. Intrinsic Constants of the Motor

To determine the value of the constant of electromotive force K, we utilize the equation: $K = \frac{Va - Ra * i}{\omega}$;

To minimize the calculation error, we measure the voltage, in intervals from 1 to the maximum voltage and obtain the average of K [10]. See Table 3.

Table 3. Intrinsic Constants of the Motor. [10]

MOTOR 180W					
E (V)	**I (A)**	**W (rpm)**	**W (rad/s)**	**K (V*s/rad)**	**B (N*m*s/rad)**
2,132	0,345	203,2	21,279104	0,0424061	0,00068753
3,255	0,381	365,3	38,254216	0,04959072	0,00049390
4,375	0,431	511	53,51192	0,05305072	0,00042728
5,367	0,473	638,8	66,895136	0,05502868	0,00038909
6,278	0,506	758,8	79,461536	0,05631069	0,00035857
7,314	0,536	891,9	93,399768	0,05785463	0,00033201
8,525	0,559	1052	110,16544	0,05929839	0,00030089
9,596	0,58	1189	124,51208	0,06046631	0,00028166
10,65	0,622	1330	139,2776	0,06054879	0,00027040
11,54	0,62	1447	151,52984	0,06157348	0,00025193
20,81	0,727	2687	281,38264	0,06474761	0,00016728
				0,05644328	0,00036005

3.4. Inertial Moment

To calculate the inertial moment we considered that the motor is a cylinder, and to calculate the moment of inertia of a cylinder is very easy, and is given by the following equation: $J = \frac{M * R^2}{2}$, where M is the mass of the motor, R the estimated radius of the engine [11].

$$J = \frac{1{,}8 * 0{,}049^2}{2} Kgm^2 = 2{,}1603x10^{-3} Kgm^2 \quad (11)$$

Value of the experimental parameters obtained Table 4.

Table 4. Experimental parameters of DC motor.

Ra (Ω)	3,56
K ($\frac{Vs}{Rad}$)	0,056
J (Kgm^2)	$2{,}1603x10^{-3}$
B ($\frac{Nms}{Rad}$)	3,6x10^{-4}
L (μH)	441,2

Source: own.

Replacing the value of the parameters in (10) and operating

$$G(S) = \frac{58737{,}82}{S^2+8069{,}07S+4633{,}67} \tag{12}$$

Where Equation (12) is equals the transfer function of the plant.

4. PID Control

The PID controllers are the most frequently used in the control of industry processes, where more than 95% of control loops use PID controllers. It's simple, versatility and ability to solve the basic problems in the industry processes with favorable dynamics and modest operating requirements make them an essential tool in process control. [12-13].

The transfer function of the PID controller given in Equation (13) and is partitioned as [12-13]:

$$G_c(s) = K_p + K_d S + \frac{K_i}{S}$$

$$G_c(s) = (1 + K_{d1}S)(K_{p2} + + \frac{K_{i2}}{S}) \tag{13}$$

The proportional constant of the PD part is 1, due that the PID controller only requires three parameters.

Equating both members of Equation (17) you have:

$$K_d = K_{d1}K_{d2} \tag{14}$$

$$K_d = K_{d1}K_{p2} \tag{15}$$

$$K_i = K_{i2} \tag{16}$$

The PI part selects the values of K_{i2} and K_{p2} that are required so that the system upload time is satisfied. The stable state error of the PI control system is improved in an order. The maximum over pulse at this stage is not considered and may be larger than desired [12-13].

The PD part is used to reduce the maximum over impulse. The value of K_{d1} is selected to find the maximum over pulse required. The values of Kp, Kd and Ki are found using Equations (14), (15) and (16) [12-13].

Exist a PID algorithm called speed PID, which has the following control law [11]:

$$U(s) = K\left(1 + \frac{1}{T_i S} + \frac{T_d S}{1+\frac{T_d}{N}S}\right)E(s) \tag{17}$$

4.1. Tuning of PID

For the implementation of the PID controller we utilize a block in SIMULINK / MATLAB (Figure 4) called PID Controller is used; This block tunes the controller gains automatically (Figures 5 and 6) [14].

Figure 4. Block Diagram of PID Control in [15].

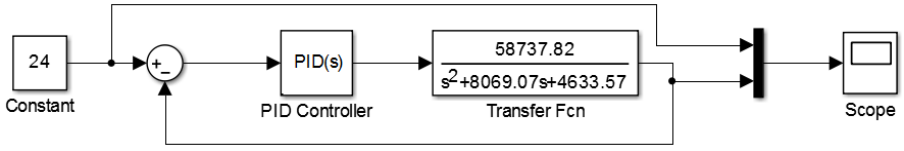


Figure 5. Automatic Tuning of constants in [15].

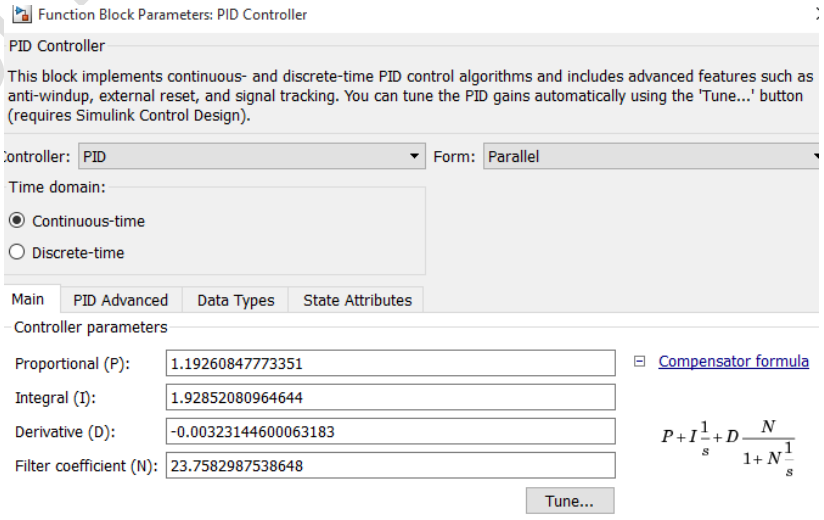
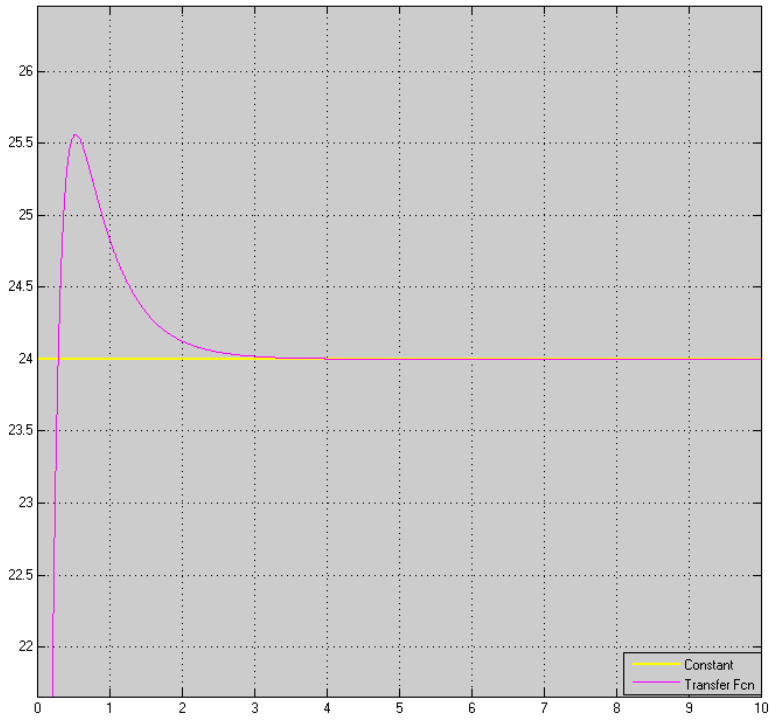


Figure 6. Response of the Controlled plant and reference signal in [15].



4.2. Digital PID control

To perform a digital control, the control law must be discretized; in this case the PID control law must be discretized, applying the Z transform to Equation (18), [11] is obtained:

$$\frac{U(z)}{E(z)} = K_p\left(1 + \frac{T}{T_i(1-Z^{-1})} + \frac{\frac{N*T}{T_d+N*T}(1-Z^{-1})}{1-\frac{T_d}{T_d+N*T}Z^{-1}}\right) \tag{18}$$

Where T is the sampling period.

To implement the PID Control in the embedded system Raspberry Pi it is necessary to transform Equation (18), in the domain Z, using equations in differences through the inverse Z transform, obtaining [11]:

$$U(k) = U_{k-1} + a * E_k + b * E_{k-1} + c * E_{k-2} \tag{19}$$

$$a = K_p + \frac{K_p*T}{T_i} + \frac{K_p*T_d}{T} \tag{20}$$

$$b = K_p + \frac{2*K_p*T_d}{T} \tag{21}$$

$$c = K_p + \frac{K_p * T_d}{T} \quad (22)$$

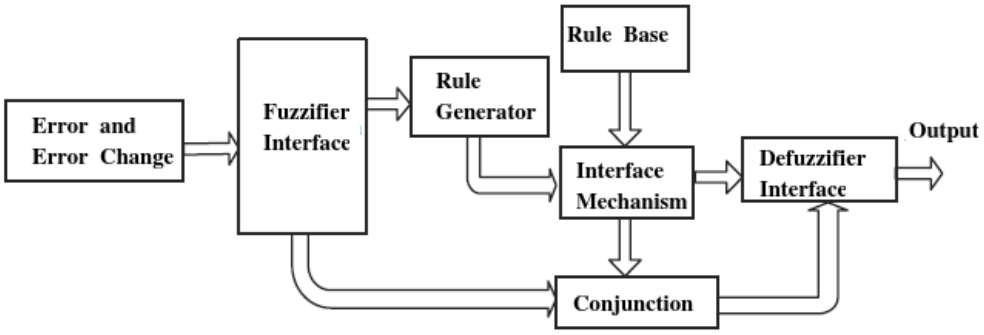
5. Fuzzy Control

The fuzzy control provides a formal methodology to represent, manipulate and implement the heuristic knowledge to control a system. Basically, the fuzzy controller is a system that makes decisions, operates in a closed cycle and real time. The fuzzy control receives information of the outputs of the plant to controller, compares it with the reference value, and then decides which the inputs of the plant [16].

A block diagram of a digital fuzzy control system is shown in Figure 7. The fuzzy controller consists of the following 6 elements [16]:

1) A base rules (“IF-THEN” rule set), which contain a fuzzy logical quantification of the expert's linguistic description to achieve optimal control.
2) An active rule generator, in this stage each input variable generates a memory address discretized and associated with its corresponding active rule stored in memory.
3) An inference mechanism (Also called “inference engine” or “fuzzy Inference module”), which emulates the expert's decision to interpret and apply knowledge of the best form to the control of the plant.
4) The Fusion interface, converts the controller inputs into information that the inference mechanism can easily use to activate and apply rules.
5) An aggregate stage, where the fuzzy sets are evaluated before being defusified and grouped.
6) The Defuzzifier Interface, which converts the conclusions of the inference mechanism to inputs suitable for the process

Figure 7. Fuzzy Controller. [6]



5.1. Implementation of a fuzzy controller in [15]

The implementation of the fuzzy controller, uses Mandani type, of the MATLAB [15] Fuzzy Logic Designer toolbox is used [17]; This toolbox allows to enter the control rules, membership functions, inference methods, merger and defusification [16-20].

For the implementation of the fuzzy controller, two inputs called ERROR and CHANGE were taken. The error was divided into 5 groups as follows:

- Vel_Alta: Trapezoidal function.
- Vel_Med_Alta: Triangular function.
- Vel_Cte: Triangular function.
- Vel_Med_Baja: Triangular function.
- Vel_Baja: Trapezoidal function.

The error scheme, see Figure 8 whose range ranges from -1 to 1, present these values per unit. It means that if the reference speed is 500 rpm, the feedback value is 600 rpm and if the base value is 1500 rpm, the error presented would be:

$$Error = \frac{V_{ref} - V_{salida}}{V_{Base}} \qquad Error = \frac{500-600}{1500} = -0.067 \tag{23}$$

The result of Equation (23), shows the error would be included in the Vel_Cte and Vel_Med_Baja category. This stage is known as fusification and inference.

The derivative of the error see Figure 8 was divided into two groups, and represent the sudden changes, translated as change of error, which, when the motor rotates at constant reference speed, can present. Their range ranges from -1 to 1, presenting these values per unit.

- E_Alto_Negativo: Triangular Function.
- E_Alto_Positivo: Triangular Function.

The response, or output, of the fuzzy controller is called CONTROL. In Figure 8 you can see the distribution of the universe for the control action, whose range varies from 0 to 24 Volts:

- Decrease_Much: Triangular Function.
- Decrease_Poor: Triangular function.
- Hold: Triangular function.
- Increase_Poor: Triangular function.
- Increase_Much: Triangular Function.

However, it is essential to assign a set of base rules so that there is consistency in the response. The Figure 8 shows the action of control to be performed depending on the value received, by the controller, as input. For example, it is presented that for an ERROR equal to 0 (Vel_Cte), CHANGE equal to 0, the CONTROL action must be 12V (Hold). This last stage is known as defusification and MATLAB / SIMULINK [15] uses the method of centroid. For practical purposes, was not built a control matrix as recommended in [16-18], since the objective of this project is to implement the result of these simulations in an embedded system and many control rules generate a quite high computing expense. The Figure 9 shows the characteristic surface of the diffuse controller and all possible combinations that the system can offer.

Figure 8. Rules of control [15].

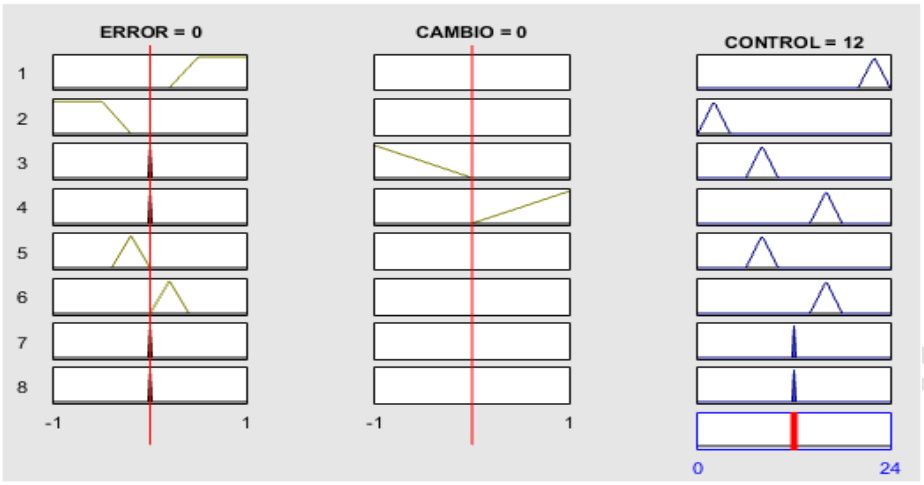
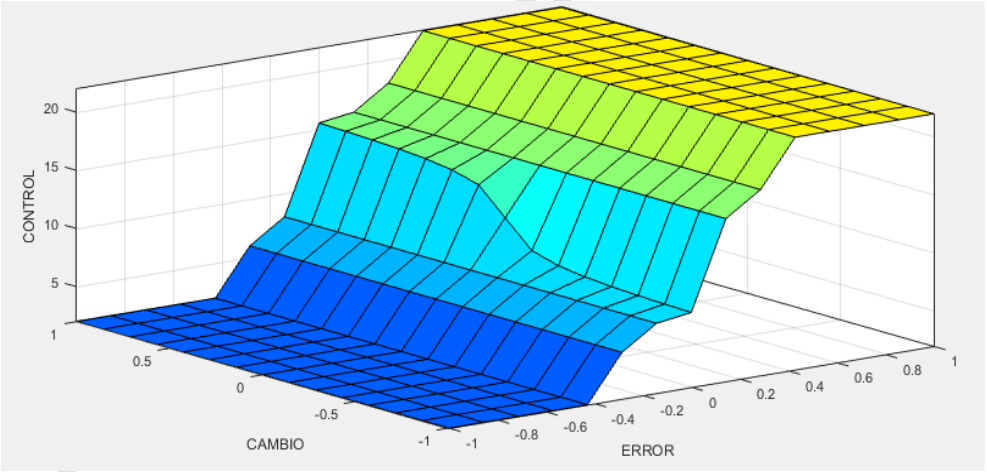


Figure 9. Control Surface [15].



Finally, the Figure 10 and Figure 11 show the block diagram of the fuzzy control system and the response of the system to a disturbance. The block calls DC motor, is the equivalent to the DC motor, whose values of resistance of armature, inductance of armature, moment of inertia and intrinsic parameters were found by means of laboratory tests and can be seen in Table 4. In addition to the parameters for the model of motor, in the block is a torque input to simulate variations of it at different times during simulation.

In fact, Figure 11 shows the response of the system to a considerable increase in torque, 7 seconds after the start of the simulation; and we see the as despite the transients, tries to follow the reference value.

Figure 10. Diagram of Blocks of Fuzzy control [15]

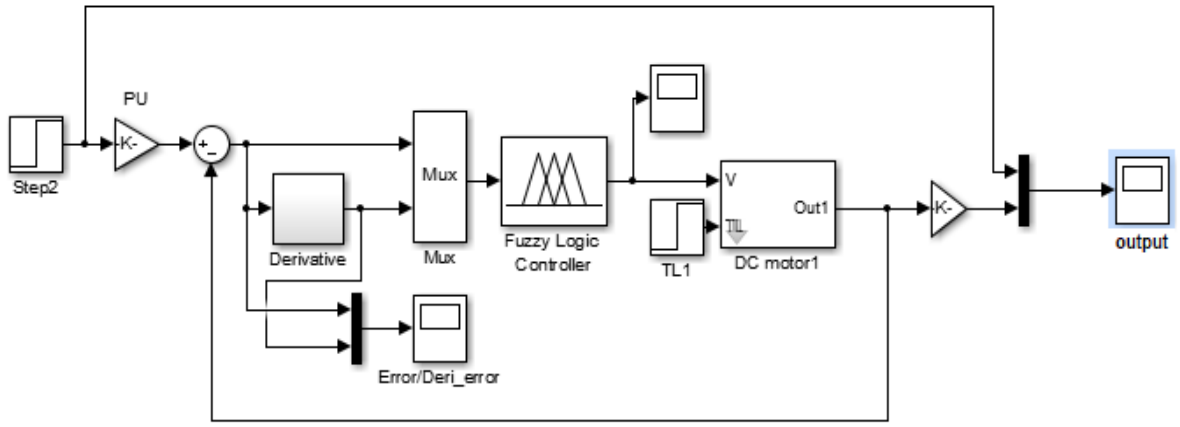
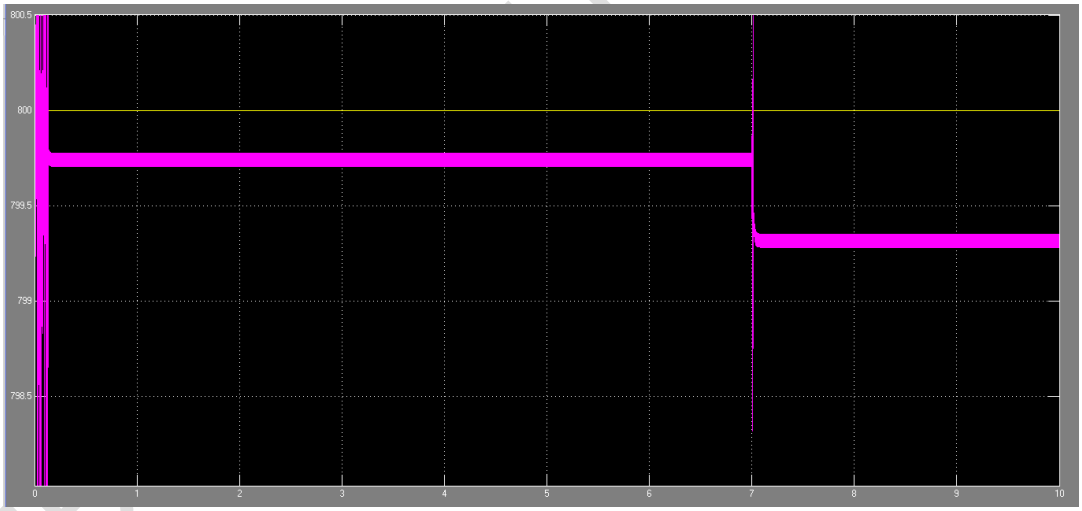


Figure 11. Graphic of Output signal and reference signal of fuzzy controller [15].



5.2. Tests of the controllers

7 cases were implemented for each control technique, varying the torque and reference speed, under the following parameters: constant torque and variable speed; variable torque and constant speed; variable torque and speed. Presenting the results in Tables 5 and 6.

Table 5. Test of the motor using fuzzy logic. Source: own.

Cases		Torque [Nm]	Ref. Speed [rpm]	Out. Speed [rpm]	Current [A]	Response Time [ms]
1	Start	1.35	100	100.10 - 100.04	10.03	7
	End	1.35	1500	1499.64 - 1499.57	10.03	
2	Start	2.10	100	99.78 - 99.71	13.72	8
	End	0.00	1500	1499.84 - 1499.77	2.12	
3	Start	0.00	1500	1499.84 - 1499.76	2.12	7.5
	End	0.00	100	99.93 - 99.96	2.12	
4	Start	0.00	1500	1499.84 - 1499.76	2.12	8
	End	2.10	100	99.62 - 99.54	13.72	
5	Start	0.00	1500	1499.84 - 1499.76	2.12	7
	End	1.35	1500	1499.64 - 1499.57	10.03	
6	Start	0.00	1000	999.76 - 999.69	2.12	7
	End	1.35	1000	999.54 - 999.49	10.03	
7	Start	0.00	500	499.86 - 499.78	2.12	7
	End	2.10	500	499.41 - 499.64	13.72	

Table 6. Test of the motor using PID control. Source: own.

Cases		Torque [Nm]	Ref. Speed [rpm]	Out. Speed [rpm]	Current [A]	Response Time [s]
1	Start	1.35	100	99.85	10.03	2.3
	End	1.35	1500	1500.5	10.03	
2	Start	2.1	100	99.85	13.72	2.3
	End	0	1500	1500.5	2.12	
3	Start	0	1500	1500.6	2.12	2.3
	End	0	100	99.5	2.12	
4	Start	0	1500	1501	2.12	2.3
	End	2.1	100	98	13.72	
5	Start	0	1500	1501.6	2.12	2.3
	End	1.35	1500	1499.4	10.03	
6	Start	0	1000	1002	2.12	2.3
	End	1.35	1000	99.5	10.03	
7	Start	0	500	500.8	2.12	2.3
	End	2.1	500	499.4	13.72	

5.3. Simulations analysis

Based on the results obtained in Tables 5 and 6, it can be concluded that the fuzzy controller responds faster than the PID controller. The first has an answer in milliseconds, while the PID controller has an answer in seconds. In addition, regarding the analysis of transients in the response signals of both controllers, it can be observed in case 4 that in the fuzzy controller it is almost non-existent, while in the PID controller it generates a longer transient.

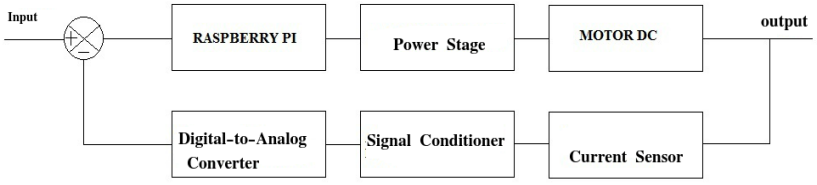
However, the fuzzy controller presents interference in case 5 in which the response of the system without charge is simulated in an increase in effort and the system remaining invariant in speed. Also we appreciated that the response of the PID controller is much cleaner and smoother. The possibility of interference being mitigated by the circuit elements used in the prototype is raised.

6. Implementation of the prototype of control of speed and torque of a DC motor using the embedded system Raspberry PI.

6.1. The scheme of the control of speed and torque of the DC Motor

The model of control of speed and torque of DC motor can be represented by integrate the system:

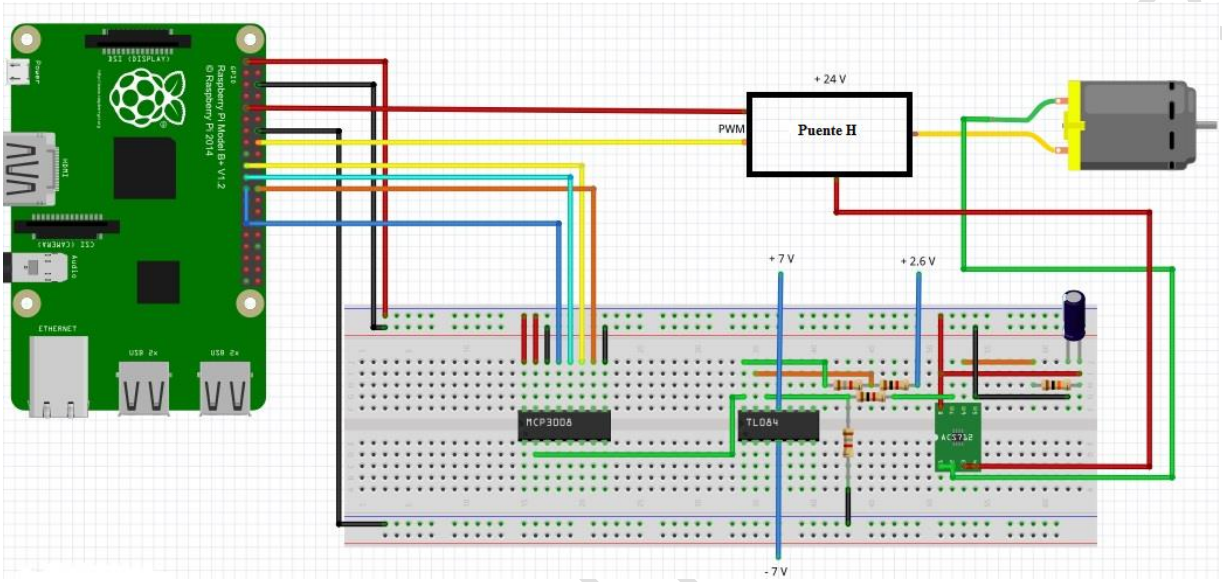
Figure 12. Scheme of Control of implemented system [6].



For the control of the DC Motor using the embedded system Raspberry Pi is necessary to use components that allow the relationship between the motor and the output signal emitted by Raspberry Pi (PWM pulse) [21-22] and the analog current reading with its corresponding conditioning and digital conversion so that it can be read by the embedded system (Figure 12) [23].

The Figure 13 shows connection of the prototype. The connection of the set of elements of the system is described: DC Motor, H Bridge, Current Sensor, Raspberry Pi, Analog / Digital Converter [22-25].

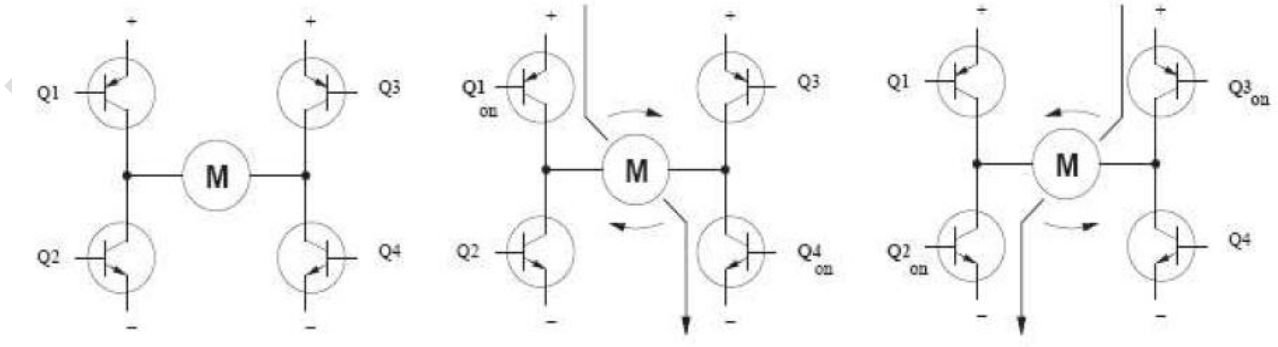
Figure 13. Connection of the prototype [26].



6.2. Power stage

The H bridge is a typical circuit used for motor control. A simplified schematic representation of the circuit is shown in Figure 14. For the rotation of motor, two of the diagonally opposite transistors are activated. Depending on the pair of activated transistors, the current flows in one or the other direction, which allows to control the direction of rotation of the motor [7, 27-29].

Figure 14. Representation of the H bridge [30].



The direction of rotation of motor is controlled by activating and deactivating pairs of diagonally opposite transistors. Thus, the current flows through the motor in two different paths: from Q1 to Q4 or from Q3 to Q2. Current flows through the motor in one or another direction, resulting in a clockwise or counterclockwise rotation of the motor [29].

The Pulse width modulation (PWM) is a technique that is based on the modification of the duty cycle of a periodic signal (for example sinusoidal or square). The duty cycle of a periodic signal is the relative width of its positive part in relation to the period [29].

$$D = \frac{\tau}{T} \tag{24}$$

Where:

D: It is the duty cycle.

τ : It is the pulse width, that is, the time in which the function is positive.

T: It is the period of the function.

The H bridge used in this work has the following schematics elaborated in the EAGLE [30] program, PWM reception, opto coupling stage, amplification and inversion, see Figure 15 and Bridge H, see Figure 16.

Figure 15. Schematic of H Bridge [30].

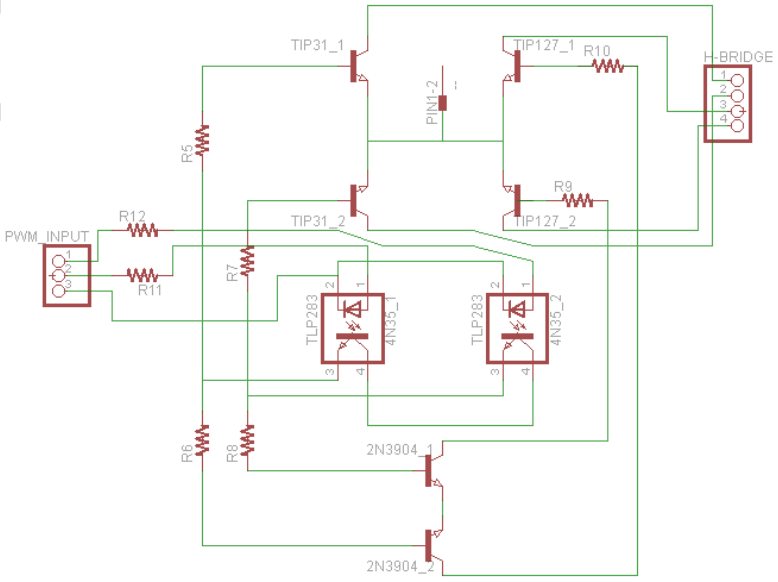
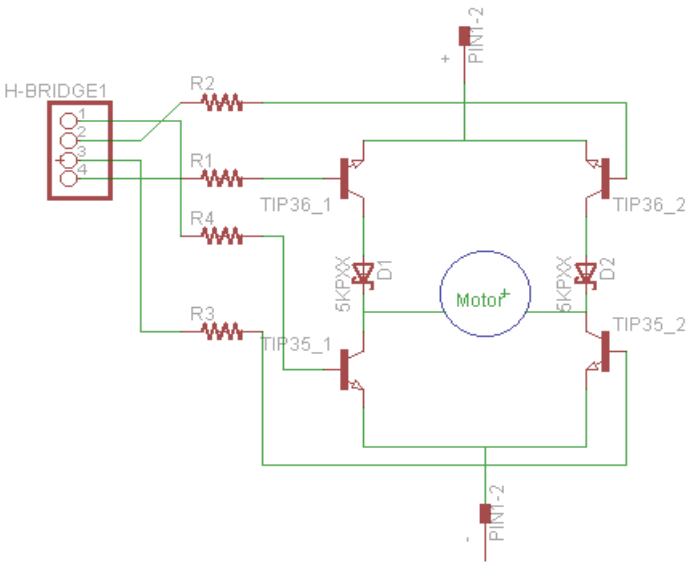
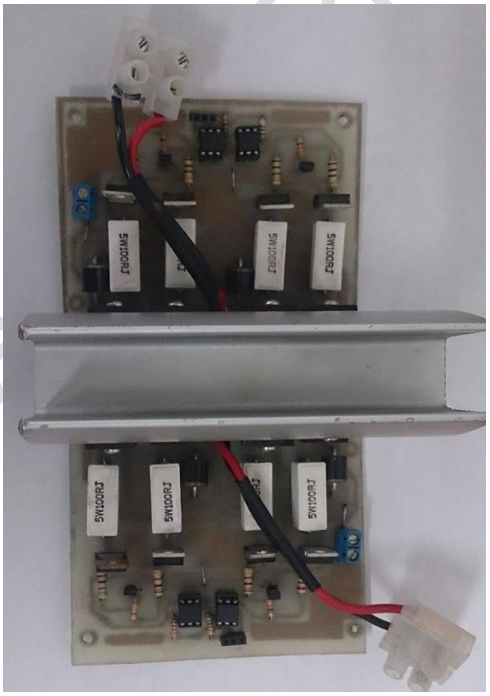


Figure 16. Circuit of H bridge and Motor [30].



In the Figure 17 shows the H bridge implemented.

Figure 17. The H bridge implemented.



Source: own.

6.3. Current sensor

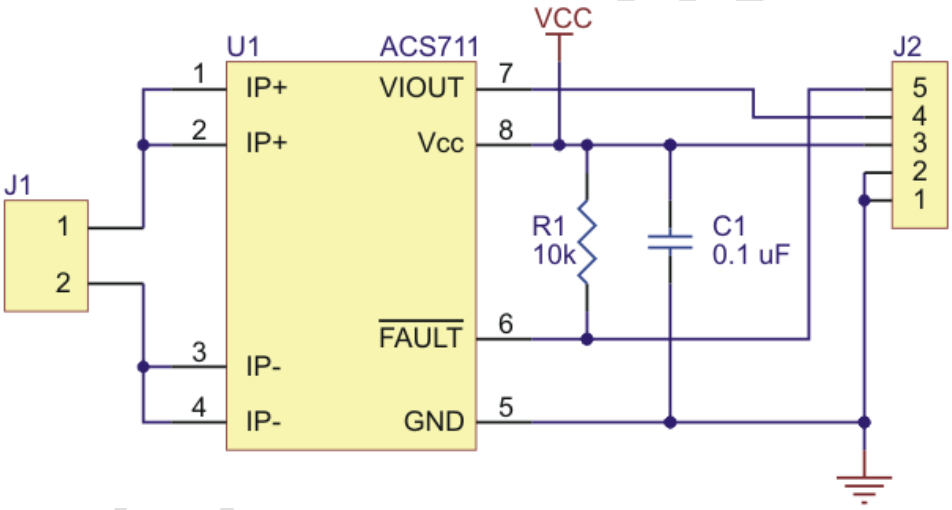
The sensor used the Allegro ACS711KEXLT-15AB-T is a linear current sensor based on Hall effect, it comes in a support board or unlocking board, with overcurrent fault output; (Figure 18).

This sensor has a voltage of 3 V to 5.5 V and an output sensitivity of 90 mV / A, when it is Vcc 3.3 V (or 136 mV / A when Vcc is 5 V) [31].

The sensor requires a 3V to 5.5V power supply connected between the Vcc and GND terminals. The sensor's output is an analog voltage, linearly proportional to the input current. The standby output voltage is Vcc / 2 and changes in relation to 90 mV per Amper of input current (when Vcc = 3.3 V), the relationship between the instantaneous input current i, and the sensor output voltage, VOUT , can be represented by the following equation [31]:

$$Vout = \frac{Vcc}{2} + i * \frac{Vcc}{36.7} \quad (25)$$

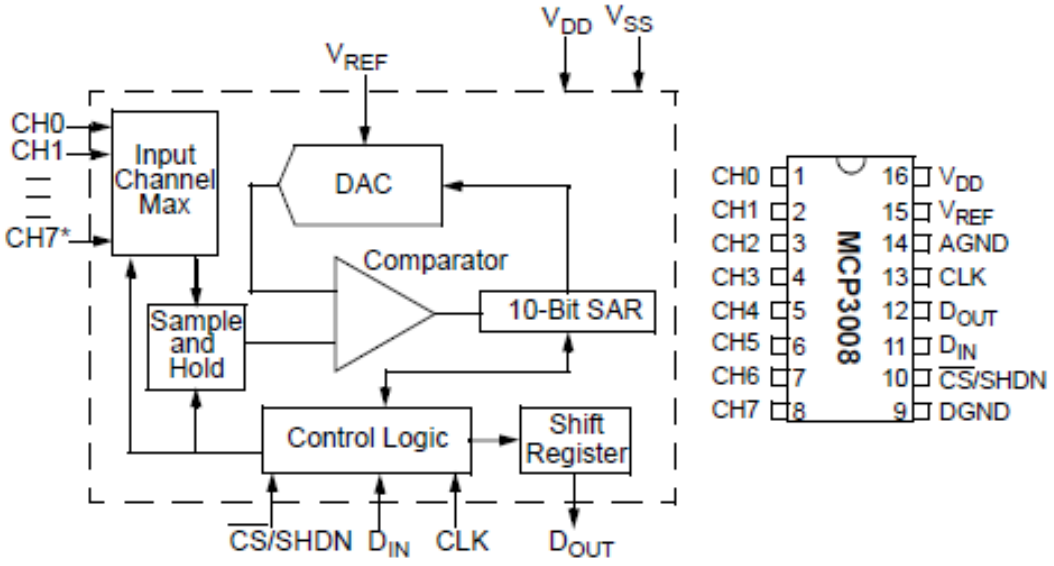
Figure 18. Circuit of current sensor [32].



6.4. MCP3008 analog-digital converter

The Mcp3008 device is a digital analog converter of successive 10-bit approximations (Figure 19), it is programmable to provide four pairs of pseudo-differential inputs or eight single-ended inputs. The communication with this device is achieved using a simple serial interface compatible with the SPI protocol (in this case Raspberry Pi [4, 7, 28]. This device is capable of reaching a conversion rate of up to 200 KSPS, operates over a wide voltage range (2.7 V - 5.5 V), its design allows operation with typical reserve currents of only 5 nA and typical active currents of 320 uA [33].

Figure 19. Scheme of Digital-analog converser [34].

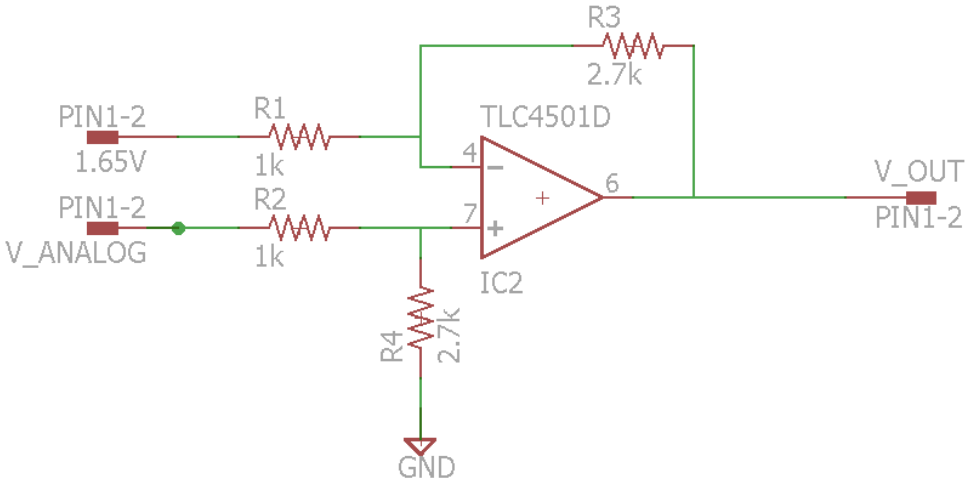


6.5. Signal Conditioning

To make an adequate reading of the current by means of the sensor, it is necessary to adecuate the output voltage to the values allowed by the Raspberry Pi embedded system [24], which are between 0V and 3.3V, for this, is necessary to implement a conditioning circuit of signal that matches the output voltage (Figure 20). The conditioning circuit used is a "subtractor" that operates under the following equation:

$$Vdig = 2.7(Vanal - 1.65) \tag{26}$$

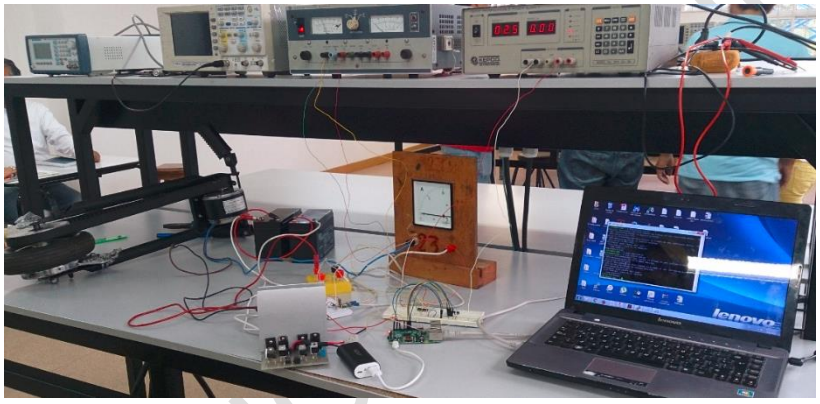
Figure 20. Circuit of the conditioning signal [30].



7. Results

The prototype was implemented, including the motor and its braking system, the embedded system Raspberry Pi [22, 24], H bridge, current sensor and signal conditioning, analog / digital converter, see Figure 19; Also, measuring elements, such as digital multimeter, analog ammeter, oscilloscope; and the different polarization voltages necessary for the correct operation of the electronic elements, see Figure 21.

Figure 21. The Implementation of the prototype of the control system of speed and torque of a DC motor.



The controllers were tested based on the results obtained in the simulations, Tables 5 and 6. It was observed that the response of the fuzzy controller was not constant even when the motor is without load, see Figure 22, It was determined that the output of the analog / digital converter oscillated around +/- 0.3V the reading value due that the current sensor delivers an analog voltage at its output with disturbances. The system is conditioned for the analog/digital converter receives a voltage between 0 and 3.3 V, and a variation of 0.3V represents 9% of the operating range. To protect the controller of these circuit disturbances, was installed a 0.1 µF capacitor at the converter input and the codes were reprogrammed so that the converter output variable, which is a position between 0 and 1023, do not return a different work cycle in the n-iteration of the program, as long as the measured variable does not leave a range of +/- 50 positions.

Figure 22. Controller stability. Source: own.

```
pi@raspberrypi ~ $ sudo python fuzzy3.py
Cambio
Nivel 511
---------------------------{}--------------------------------
CicloTrabajo 8   Voltaje A/D 1.65   Error -0.5
Nivel 511
---------------------------{}--------------------------------
CicloTrabajo 8   Voltaje A/D 1.65   Error -0.5
Nivel 511
---------------------------{}--------------------------------
CicloTrabajo 8   Voltaje A/D 1.65   Error -0.5
Nivel 511
---------------------------{}--------------------------------
CicloTrabajo 8   Voltaje A/D 1.65   Error -0.5
Nivel 511
---------------------------{}--------------------------------
CicloTrabajo 8   Voltaje A/D 1.65   Error -0.5
Nivel 511
---------------------------{}--------------------------------
CicloTrabajo 8   Voltaje A/D 1.65   Error -0.5
Nivel 511
---------------------------{}--------------------------------
CicloTrabajo 8   Voltaje A/D 1.65   Error -0.5
Cambio
Nivel 675
---------------------------{}--------------------------------
CicloTrabajo 16   Voltaje A/D 2.18   Error -0.339393939394
Cambio
Nivel 510
---------------------------{}--------------------------------
CicloTrabajo 8   Voltaje A/D 1.65   Error -0.5
Nivel 510
---------------------------{}--------------------------------
CicloTrabajo 8   Voltaje A/D 1.65   Error -0.5
Nivel 510
---------------------------{}--------------------------------
CicloTrabajo 8   Voltaje A/D 1.65   Error -0.5
```

The Figures 23 and 24 show the behavior of the PID and fuzzy controllers, respectively. It is observed that both respond efficiently to the stability of the system. The first was tested with a reference speed of 700 rpm whose theoretical response should be to keep a 50% duty cycle, as shown in Figure 23. The fuzzy controller was tested with a reference voltage of 0.65V, which translates into a negative error whose response must be to reduce the duty cycle to less than 10%. As shown in Figure 24 It should be noted that the measures implemented to protect the system of disturbances at the analog / digital converter input were effective.

Figure 23. Test results of the PID controller without load. Source: own.

```
pi@raspberrypi ~ $ sudo python PID25.py
---------------- 1 --------------------
Velocidad 700 rpm
Ciclo de trabajo 50 %
Cambio
---------------- 2 --------------------
a 12.6916896959 b 1.19106754995 c -0.0007704250264 Error 0.0141365824 Data 544
Current : (1.75V) 1.22 Amp Velocidad 700.85092196 RPM Referencia 700
Ciclo de trabajo 50 %
---------------- 3 --------------------
a 12.6916896959 b 1.19106754995 c -0.0007704250264 Error 0.0141365824 Data 544
Current : (1.75V) 1.22 Amp Velocidad 700.16854342 RPM Referencia 700
Ciclo de trabajo 50 %
---------------- 4 --------------------
a 12.6916896959 b 1.19106754995 c -0.0007704250264 Error 0.0141365824 Data 544
Current : (1.75V) 1.22 Amp Velocidad 700.66380612 RPM Referencia 700
Ciclo de trabajo 50 %
---------------- 5 --------------------
a 12.6916896959 b 1.19106754995 c -0.0007704250264 Error 0.0141365824 Data 544
Current : (1.75V) 1.22 Amp Velocidad 700.66380612 RPM Referencia 700
Ciclo de trabajo 50 %
---------------- 6 --------------------
a 12.6916896959 b 1.19106754995 c -0.0007704250264 Error 0.0141365824 Data 544
Current : (1.75V) 1.22 Amp Velocidad 700.66380612 RPM Referencia 700
Ciclo de trabajo 50 %
---------------- 7 --------------------
a 12.6916896959 b 1.19106754995 c -0.0007704250264 Error 0.0141365824 Data 544
Current : (1.75V) 1.22 Amp Velocidad 700.66380612 RPM Referencia 700
Ciclo de trabajo 50 %
---------------- 8 --------------------
a 12.6916896959 b 1.19106754995 c -0.0007704250264 Error 0.0141365824 Data 544
Current : (1.75V) 1.22 Amp Velocidad 700.66380612 RPM Referencia 700
Ciclo de trabajo 50 %
---------------- 9 --------------------
a 12.6916896959 b 1.19106754995 c -0.0007704250264 Error 0.0141365824 Data 544
Current : (1.75V) 1.22 Amp Velocidad 700.66380612 RPM Referencia 700
Ciclo de trabajo 50 %
---------------- 10 --------------------
a 12.6916896959 b 1.19106754995 c -0.0007704250264 Error 0.0141365824 Data 544
Current : (1.75V) 1.22 Amp Velocidad 700.66380612 RPM Referencia 700
Ciclo de trabajo 50 %
---------------- 11 --------------------
a 12.6916896959 b 1.19106754995 c -0.0007704250264 Error 0.0141365824 Data 544
Current : (1.75V) 1.22 Amp Velocidad 700.66380612 RPM Referencia 700
```

Figure 24. Test results of the fuzzy controller without load. Source: own.

```
pi@raspberrypi ~ $ sudo python fuzzy3.py
Cambio
Nivel 510
--------------------------{}--------------------
CicloTrabajo 8   Voltaje A/D 1.65   Error -0.5
Nivel 510
--------------------------{}--------------------
CicloTrabajo 8   Voltaje A/D 1.65   Error -0.5
Nivel 510
--------------------------{}--------------------
CicloTrabajo 8   Voltaje A/D 1.65   Error -0.5
Nivel 510
--------------------------{}--------------------
CicloTrabajo 8   Voltaje A/D 1.65   Error -0.5
Nivel 510
--------------------------{}--------------------
CicloTrabajo 8   Voltaje A/D 1.65   Error -0.5
Nivel 510
--------------------------{}--------------------
CicloTrabajo 8   Voltaje A/D 1.65   Error -0.5
Nivel 510
--------------------------{}--------------------
CicloTrabajo 8   Voltaje A/D 1.65   Error -0.5
Nivel 510
--------------------------{}--------------------
CicloTrabajo 8   Voltaje A/D 1.65   Error -0.5
Nivel 510
--------------------------{}--------------------
CicloTrabajo 8   Voltaje A/D 1.65   Error -0.5
Nivel 510
--------------------------{}--------------------
CicloTrabajo 8   Voltaje A/D 1.65   Error -0.5
```

At the time of demanding the motor, by means of the braking action, there were complications with the voltage delivered by the H- Bridge since it did not correspond to the value of the work cycle delivered by the embedded system, Raspberry Pi. It was concluded that the batteries had lost responsiveness.

Even so, when monitoring with the digital multimeter the output voltages of the multipurpose port configured as the output of the PWM programmed in Python, to the output voltage of Bridge H; and the values presented on the screen of system, it can be determined that both controllers met the system requirements by increasing the duty cycle. See Figure 25.

Figure 25. Test results of the fuzzy controller with load. Source: own.

```
pi@raspberrypi ~ $ sudo python fuzzy3.py
Cambio
Nivel 511
--------------------------{}--------------------------------
CicloTrabajo 75   Voltaje A/D 1.65   Error 0.30303030303
Nivel 511
--------------------------{}--------------------------------
CicloTrabajo 75   Voltaje A/D 1.65   Error 0.30303030303
Cambio
Nivel 656
--------------------------{}--------------------------------
CicloTrabajo 100   Voltaje A/D 2.12   Error 0.445454545455
Nivel 656
--------------------------{}--------------------------------
CicloTrabajo 100   Voltaje A/D 2.12   Error 0.445454545455
Nivel 656
--------------------------{}--------------------------------
CicloTrabajo 100   Voltaje A/D 2.12   Error 0.445454545455
Nivel 656
--------------------------{}--------------------------------
CicloTrabajo 100   Voltaje A/D 2.12   Error 0.445454545455
Nivel 656
--------------------------{}--------------------------------
CicloTrabajo 100   Voltaje A/D 2.12   Error 0.445454545455
Nivel 656
--------------------------{}--------------------------------
CicloTrabajo 100   Voltaje A/D 2.12   Error 0.445454545455
Nivel 656
--------------------------{}--------------------------------
CicloTrabajo 100   Voltaje A/D 2.12   Error 0.445454545455
```

8. Conclusions

The design of the controller was carried out using and modeling the different physical parameters of the system to be controlled (Table 4).

In addition, we implemented the necessary methodology for the control, as well as the theoretical basis on which the calculations are based, since, without a modeling of the physical system, it would be impossible to perform such design or the relevant simulations to verify the validity of the same.

An approach of a fuzzy logic controller and PID was presented for the control a DC motor, using Python programming and an embedded digital system, Raspberry Pi.

From the results acquired for the simulated systems it can be affirmed that the implementation with the best behavior was the fuzzy control since, in addition to providing the response more quickly, a more accurate estimate is achieved than the PID control.

Based on the results obtained in the final tests it can be deduced that, although the fuzzy controller has a much shorter response time compared to that of the PID, the stability of the PID control is more useful in applications of robust systems, since the principle of overlapping areas used in the process of defusification presents problems in systems with presence of disturbances and unnatural oscillations.

appealing

Acknowledgments

The Authors, thank the Technological University of Pereira, for the support in the realization of this project, which was sponsored by the Vice Rectory of Research and Extension.

References

[1] J. Teeter, M. Chow, and J. J. Brickley, “Use of a Fuzzy Gain Tuner for Improved Control of a DC Motor System with Nonlinearities”, in Proceedings of 1994 IEEE International

Conference on Industrial Technology - ICIT '94, pp. 258-262, 1994. https://doi.org/10.1109/ICIT.1994.467117

[2] B. Behnam and M. Mansouryar, “Modeling and simulation of a DC motor control system with digital PID controller and encoder in FPGA using Xilinx system generator”, in Proc. 2nd Int. Conf. Instrum. Control Autom, ICA, pp. 104–108, 2011. https://doi.org/10.1109/ICA.2011.6130138

[3] S. B. Noor, S. M. Uashi, and M. K. Hassan, “Microcontroller Performance for DC Motor Speed Control System”, in Proceedings. National Power Engineering Conference, PECon, pp. 104–109, 2003. https://doi.org/10.1109/PECON.2003.1437427

[4] Raspberry Pi Blog, “raspberry.org”. [Online]. Available at: http://www.Raspberry.org

[5] K. Ogata, “Ingeniería de control moderna”, Prentice Hall, pp. 567-596. 2003.

[6] W. I. Hameed, and K. A. Mohamad, “Speed control of separately excited dc motor using fuzzy neural model reference controller”, *International Journal of Instrumentation and Control Systems (IJICS)*, vol, 2, no. 4, pp. 27-39, 2012. https://doi.org/10.5121/ijics.2012.2403

[7] A. Dorzhigulov, B. Bissengaliuly, B. F. Spencer, J. Kim, and A. P. James, “ANFIS based quadrotor drone altitude control implementation on Raspberry Pi platform”, *Analog Integrated Circuits and Signal Processing*, vol. 95, no. 3, pp. 435-445, 2018. https://doi.org/10.1007/s10470-018-1159-8

[8] C. Jiménez, “Estimación básica de los parámetros del circuito equivalente de la máquina de corriente directa”, pp. 1–4, 2015. [Online]. Available at: https://es.scribd.com/document/257965184/Estimacion-Basica-de-Los-Parametros-Del-Circuito-Equivalente-de-La-Maquina-de-Corriente-Directa

[9] U. K. Bansal, and R. Narvey, “Speed control of DC motor using fuzzy PID controller”, *Advance in Electronic and Electric Engineering*, vol. 3, no. 9, pp. 1209-1220, 2013.

[10] S. Salvador, “Determinación de los parámetros de un motor de CD por medición física directa”, pp. 5-8, 2014. [Online]. Available at: https://www.academia.edu/9614705/Obtenci%C3%B3n_de_Par%C3%A1metros_de_un_Motor_de_CD

[11] R. Alexander Montenegro, C. Alberto, and F. Perdomo, “Diseño e Implementación de un Control PID Digital para Motor DC”, in 2do. Congreso Virtual de Microcontroladores y sus Aplicaciones, pp. 1–9, 2010.

[12] L. Moreno, S. Garrido, and C. Balaguer, “Ingeniería de control. Modelado y control de sistemas dinámicos”, Ed. Ariel, p. 460, 2003.

[13] A. O’Dwyer, “Handbook of PI and PID Controller Tuning Rules”, *IEEE Control Systems Magazine*, vol. 26, no. 1. 2006. https://doi.org/10.1109/MCS.2006.1580157

[14] MathWorks, “PID Controller”. [Online]. Available at: https://www.mathworks.com/help/simulink/slref/pidcontroller.html

[15] MathWorks, “MATLAB: the language of technical computing, visualization, programming installation guide for UNIX version 5”, Natwick: Math Works Inc., 1996.

[16] K. Passino and S. Yurkovich, “Fuzzy control”, Addison Wesley Longman, 1998.

[17] MathWorks, “Fuzzy Logic Toolbox”. [Online]. Available at: https://www.mathworks.com/help/fuzzy/

[18] L. Zadeh, “Fuzzy Sets”, University of California, Berkeley, California, USA, pp. 19-34, 1996. https://doi.org/10.1142/9789814261302_0001

[19] C. D. De Los Ríos and W. Ipanaqué, “Evaluación de estructuras y métodos de ajuste de reguladores PID-difusos”, Comunicaciones aceptadas en las XXVI jornadas de automática, pp. 4–26, 2004.

[20] L. Reznik, “Fuzzy Controller” in Victoria University of Technology Melbourne, Australia, Oxford, p. 287, 1997.

[21] A. Cobo, “Guía de Raspberry Pi”, 2013. [Online]. Available at: https://hardlimit.com/guia-raspberry-pi/

[22] M. Bejarano, “Conexión remota al Raspberry Pi usando SSH”, 2013. [Online]. Available at: http://www.frambuesapi.co/2013/%2009/25/tutorial-5-conexion-remota-al-raspberry-pi-usando-ssh/

[23] W. J. Tang, and S. Y. Cao, “A Fast Realization Method of Fuzzy PID Control for DC Motor”, in 37th Chinese Control Conference (CCC), pp. 5131-5135, 2018. https://doi.org/10.23919/ChiCC.2018.8483184

[24] A. Robinson, and M. Cook, “Raspberry Pi Projects”, John Wiley & Sons Inc., 2013.

[25] J. L. Rincón-Gaviria, “Control PID para el control de velocidad de un motor DC”, thesis, Universidad Tecnológica de Pereira, Colombia, 2014.

[26] G. Fischer, “Creativity and Distributed Intelligence”, in Report of Workshop on Creativity Support Tools, pp. 71-73, 2005.

[27] M. Boutouba, A. Ougli, and S. Miqoi, “Intelligent control for voltage regulation system via DC-DC Converter using Raspberry Pi 2 board”, *Wseas Transactions on Electronics journal*, vol. 8, pp. 41-47, 2017.

[28] A. Kholid, R. A. Fauzi, Y. Y. Nazaruddin, and E. Joelianto, “Power Optimization of Electric Motor using PID-Fuzzy Logic Controller”, in 6th International Conference on Electric Vehicular Technology (ICEVT), pp. 189-195, 2019. https://doi.org/10.1109/ICEVT48285.2019.8993984

[29] F. Moreno, “Diseño de un sistema de control de velocidad de un motor de corriente continua basado en acelerómetros”, Universidad Pontificia Comillas, p. 350, 2010.

[30] Autodesk, “EAGLE software”. [Online]. Available at: https://www.autodesk.com/products/eagle/overview

[31] Allegro mycrosystems, “ASC711”, pp. 1-16. 2013. [Online]. Available at: https://www.allegromicro.com/en/products/sense/current-sensor-ics/zero-to-fifty-amp-integrated-conductor-sensor-ics/acs711

[32] Circuitmaker Environment, “Software by PCB design software”. [Online]. Available at: https://circuitmaker.com/

[33] Microchip, "MCP3004/3008", pp. 1–40, 2008. [Online]. Available at: http://ww1.microchip.com/downloads/en/DeviceDoc/21295d.pdf

[34] D. S. Paulin, M. I. Jean, H. Djalo and E. Joseph, “Virtual Digital Control Scheme for a Duty-Cycle Modulation Boost Converter”, *Journal of computer science and control systems*, vol. 10, no. 2, pp. 22-27, 2017.