Instituto Tecnológico de Buenos Aires

22.85 - Sistemas de Control

Trabajo de Laboratorio N°3: Control Servo por Realimentación lineal de Estados

Grupo 1

Máspero, Martina	57120
Mestanza, Joaquín Matías	58288
Nowik, Ariel Santiago	58309
Panaggio Venerandi, Guido Martin	56214
Parra, Rocío	57669
Regueira, Marcelo Daniel	58300

Profesor

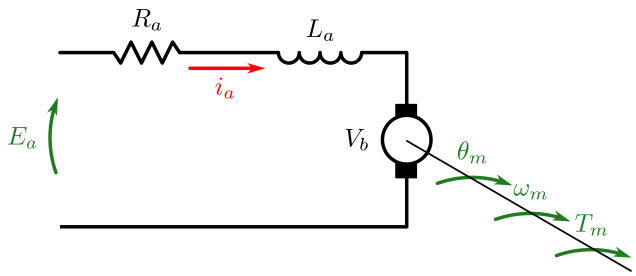
Nasini, Víctor Gustavo

Presentado: 13/11/2019

Índice

1. Análisis del Motor de CC

En primer lugar se considera el modelo circuital para el motor utilizado, teniendo en cuenta que los diferentes parámetros son datos provistos por la hoja de datos del QUANSER:



Las ecuaciones que caracterizan al sistema son:

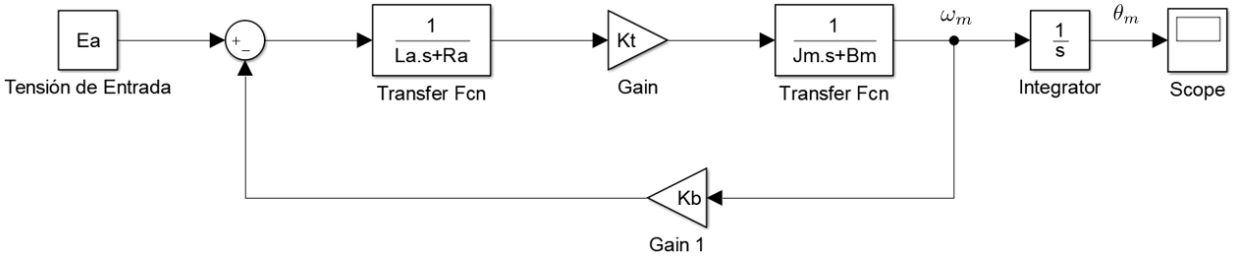
$$\begin{cases} E_a = R_a \cdot i_a + L_a \cdot \dot{i_a} + V_b \\ V_b = K_b \cdot \omega_m = K_b \cdot \dot{\theta_m} \\ T_m = J_m \cdot \ddot{\theta_m} + B_m \cdot \dot{\theta_m} \\ T_m = K_t \cdot i_a \end{cases}$$

De las cuales se puede obtener las funciones de transferencia de θ_m y ω_m respecto a la tensión de alimentación E_a. Considerando que $B_m = 0$, resulta:

$$\frac{\omega_m}{E_a} = \frac{\frac{K_t}{L_a \cdot J_m}}{S^2 + S \cdot \frac{R_a}{L_a} + \frac{K_t \cdot K_b}{L_a \cdot J_m}}$$

$$\frac{\theta_m}{E_a} = \frac{\frac{K_t}{L_a \cdot J_m}}{S \cdot \left(S^2 + S \cdot \frac{R_a}{L_a} + \frac{K_t \cdot K_b}{L_a \cdot J_m}\right)}$$

De donde se puede construir el diagrama en bloques (en general):



Donde el valor de las constantes provistas por la hoja de datos son:

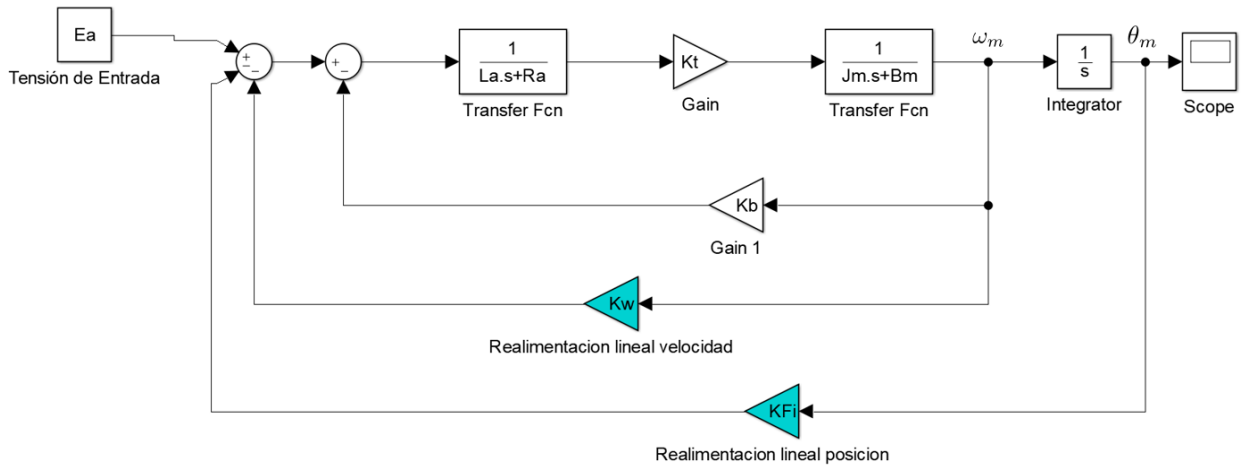
$$R_a = 2.6[\Omega] \qquad J_m = 3.87 \cdot 10^{-7}[Kg \cdot m^2] \qquad E_a = 6[V] \text{ (Máximo)} \qquad L_a = 180[\mu Hy]$$

$$K_t = 0.00767[Nm] \qquad K_b = 0.00767 \left[\frac{V}{rad \cdot seg}\right]$$

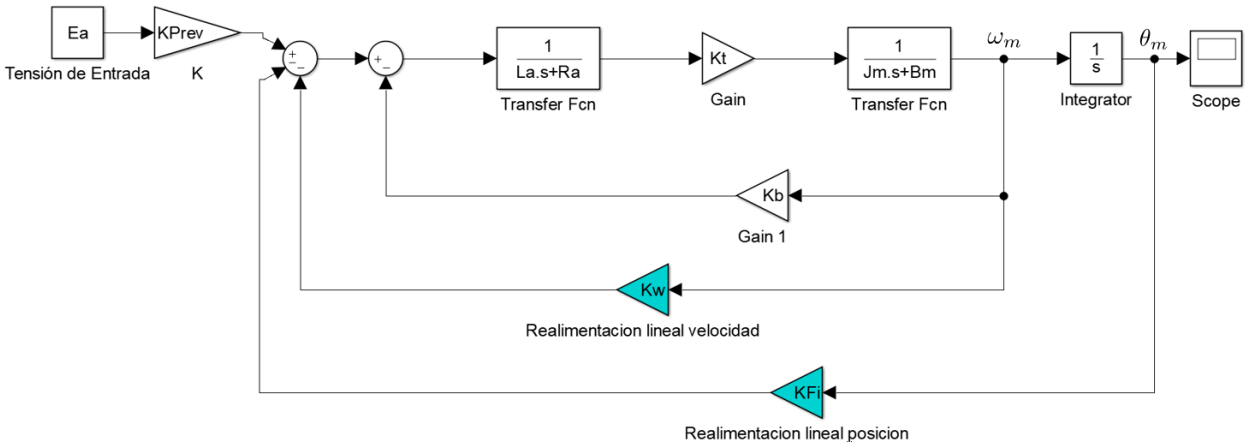
$$K_{Pot} = 35.2 \left[\frac{^\circ}{V}\right] \text{ (Sobre el potenciómtetro sin las resistencias de } 7.15K\Omega)$$

$$K_{Tach} = 1.5 \left[\frac{V}{1000RPM}\right]$$

Al sistema bajo análisis se le aplica realimentación lineal de estados para controlar la posición del motor, tomando como variables de estado a ω_m y a θ_m. Luego de aplicarla, el diagrama en bloques queda de la forma:



Para anular el error permanente se agrega delante un bloque de ganancia K_{Prev}:



Para hallar las constantes de realimentación lineal se aplica el método de Ziegler-Nichols en el controlador a implementar (Ver en), para luego comparar las mediciones que resultan con dichas constantes con la simulación. EDITAR EL CODIGO

2. Controlador Arduino - Código Fuente

Para realizar la realimentación lineal de estados, se implementó mediante un software en Arduino, cuyo código fuente se cita a continuación.

```
1
2 #include <L298N.h>
3 // Libreria para el manejo de la Placa Stepper
4
5 #define ADC_MID_RANGE 512 // El rango total es 1024, lo que hace es considerar el offset de
      la se\~nal a 2.5V
6 #define PIN_ENABLE 10 //
7 #define PIN_IN1 9
8 #define PIN_IN2 8
9
10 #define K_PREV 2.5
11
12 #define LIM_SUP 740
13 #define LIM_INF 490
14
15 #define K_TACH 1.5
16
17 // Constantes para realimentacion lineal de estados
18 #define K_FI 1
19 #define K_W 1
20
21 L298N * canalA=NULL;
22
23 class polePlacement {
24   private:
25     float kVel;
26     float kFi;
27     int pinW;
28     int pinFi;
29     int pinSetPoint;
30     L298N * motor;
31
32     configMotorSpeed(int mapedDuty);
33
34    public:
35    // Inicializador
36    polePlacement(int pinEnable, int pinIn1, int pinIn2,int pinW,int pinFi,int pinSetPoint,
       float kVel,float kFi);
37
38    runOnce();
39
40 };
41
42 polePlacement * MotorDrv;
43
44 void setup() {
45
46   MotorDrv = new polePlacement(PIN_ENABLE,PIN_IN1,PIN_IN2,A0,A1,A2,-4,2.1);
47
48 }
49
50 void loop() {
51
52   MotorDrv->runOnce();
53   delay(1); // Muestreo cada 1ms
54
55 }
56
57 polePlacement::polePlacement(int pinEnable, int pinIn1, int pinIn2,int pinW,int pinFi,int
      pinSetPoint,float kVel,float kFi){
58   this->kVel = kVel;
59   this->kFi = kFi;
60   this->pinFi = pinFi;
61   this->pinW = pinW;
62   this->pinSetPoint = pinSetPoint;
63   motor = new L298N(pinEnable,pinIn1,pinIn2);
64 }
65
66 polePlacement::runOnce(){
67
```

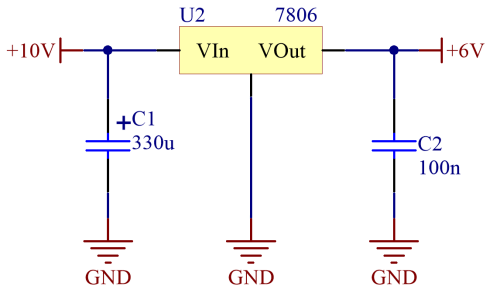
```
68    int medVel, medPos, medSetPoint, e;
69
70    medVel = analogRead(pinW) - ADC_MID_RANGE;
71    medPos = analogRead(pinFi);
72    medSetPoint = analogRead(pinSetPoint);
73
74    medSetPoint = K_PREV*(medSetPoint-ADC_MID_RANGE);
75
76    medPos = map(medPos, LIM_INF, LIM_SUP, 0, 1023) - ADC_MID_RANGE;
77
78    e = medSetPoint-(medPos*kFi)-(medVel*kVel);
79
80    configMotorSpeed(map(e, -512*K_TACH, 512*K_TACH, -255, 255));
81
82  }
83
84  polePlacement::configMotorSpeed(int mapedDuty){
85
86    if(mapedDuty > 255){
87      mapedDuty = 255;
88    }else if(mapedDuty < -255){
89      mapedDuty = -255;
90    }
91
92    if(mapedDuty < 0){
93      motor->setSpeed(-mapedDuty);
94      motor->forward();
95    }else{
96      motor->setSpeed(mapedDuty);
97      motor->backward();
98    }
99  }
```

3. Driver para señales

Dado que la implementación del controlador se realizó en Arduino, se debieron adaptar las señales de los sensores acorde a los niveles de tensión que admite el Arduino, para lo cual se realizó una placa adicional como driver.

3.1. Motor - Puente H

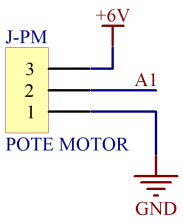
El motor acepta como máxima tensión de alimentación 6V, por lo que alimentando a la placa con ±10V, se toman los +10V y se pasan por un regulador LM7806, y de ahí se toma la alimentación para el puente H (para el control del motor):



Se utilizó la Placa Stepper para Arduino (AR-L298SHIELD), para lo cual se utilizó la librería L298.h mencionada en el código fuente.

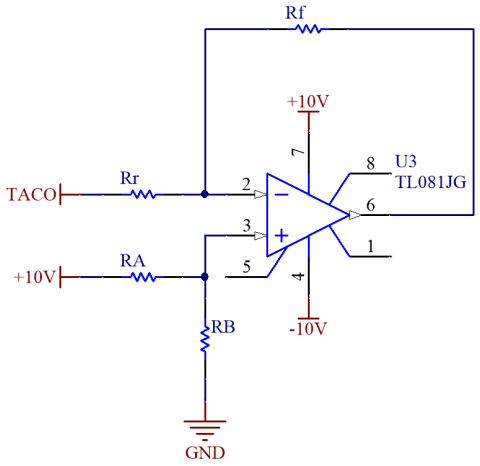
3.2. Potenciómetro (Posición)

Para el potenciómetro que permite medir la posición angular del motor, se lo alimenta con 6V y se conecta la salida a una de las entradas analógicas de Arduino (A1):



3.3. Tacómetro (Velocidad)

Para adaptar la señal del tacómetro, dado que provee valores entre -5V y +5V, se implementó con un amplificador operacional una función lineal, tal que la salida se mapee al intervalo de 0V a 5V (que es el rango admitido por el Arduino).



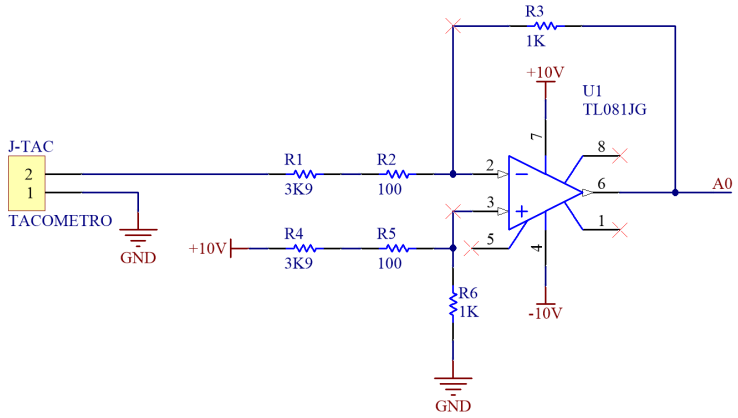
Considerando la transferencia de un amplificador inversor:

$$A_V = -\frac{R_f}{R_r}$$

El rango total inicial es de 20V, y el buscado es de 5V, por lo que el factor de escala es 4. De esta forma se toma $R_f = 1K$ y $R_r = 3K9 + 100$ (desdoblando esta última en dos resistencias en serie). Para el offset, se considera el divisor resistivo y la ganancia de un amplificador no inversor, tal que:

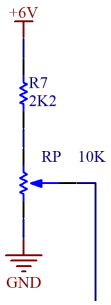
$$10V \cdot \frac{R_B}{R_A + R_B} \cdot \left(1 + \frac{1}{4}\right) = 2.5V$$

De esta forma el divisor resistivo resulta de $\frac{1}{5}$, por lo que se toma $R_B = 1K$ y $R_A = 3K9{+}100$ (desdoblando esta última en dos resistencias en serie). Finalmente, el circuito resultante es el siguiente:



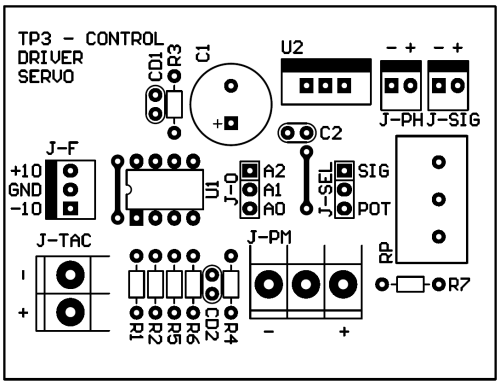
3.4. Señal de control por Potenciómetro

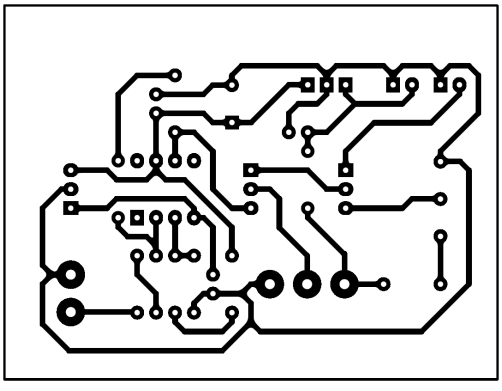
Para el potenciómetro de señal de control manual, utilizando la misma alimentación de 6V saliente del regulador, se le colocó una resistencia en serie tal que la máxima salida sea de 5V:



3.5. Driver PCB

Se muestra a continuacón el layout del diseño final de la placa.





4. Simulación y Mediciones

4.1. Entrada por señal

4.1.1. Rectangular

4.1.2. Triangular

4.1.3. Senoidal

4.2. Entrada por potenciómetro