# Proyecto final: Diseño e implementación de controlador LQR para el péndulo de Furuta.

1st Christian Daniel Loja Chalco
*Universidad de Cuenca*
*Facultad de Ingeniería*
Cuenca, Ecuador
christian.loja@ucuenca.edu.ec

2rd Nathaly Elizabeth Ramón Lojano
*Universidad de Cuenca*
*Facultad de Ingeniería*
Cuenca, Ecuador
nathaly.ramonl@ucuenca.edu.ec

3th Freddy Eduardo Lopez Padilla
*Universidad de Cuenca*
*Facultad de Ingeniería*
Cuenca, Ecuador
freddy.lopez@ucuenca.edu.ec

***Resumen*—The Furuta pendulum is a classic nonlinear system in control theory, characterized by its high instability and complex coupled dynamics. Stabilizing this system in its vertical inverted position represents a significant challenge that validates different control strategies. This work presents the design and implementation of an LQR (Linear Quadratic Regulator) controller to stabilize a real Furuta pendulum prototype. Starting from a mathematical model obtained through Lagrange formulation, linearization around the equilibrium point is performed, and an optimal controller minimizing a quadratic cost function is designed. The practical implementation is carried out on an Arduino platform, integrating state estimation, control action computation, and motor actuation. The obtained results demonstrate the effectiveness of the LQR controller in maintaining local system stability, despite the limitations imposed by linearization approximations and the nonlinearities present in the real system. The study compares simulation results with experimental implementation, analyzing performance metrics such as overshoot, settling time, and control effort.**

***Index Terms*—Furuta pendulum, LQR control, linearization, optimal control, state feedback, real-time implementation, Arduino, digital control.**

## I. Introducción

El péndulo de Furuta representa un sistema no lineal clásico en la teoría de control, caracterizado por su alta inestabilidad y compleja dinámica acoplada. La estabilización de este sistema en su posición vertical invertida constituye un desafío significativo que permite validar diferentes estrategias de control. Este trabajo presenta el diseño e implementación de un controlador LQR (Regulador Lineal Cuadrático) para estabilizar un prototipo real de péndulo de Furuta. A partir de un modelo matemático obtenido mediante la formulación de Lagrange, se realiza la linealización alrededor del punto de equilibrio y se diseña un controlador óptimo que minimiza una función de costo cuadrática. La implementación práctica se lleva a cabo en una plataforma Arduino, donde se integra la estimación de estados, el cálculo de la acción de control y la actuación sobre el motor. Los resultados obtenidos demuestran la efectividad del controlador LQR para mantener la estabilidad local del sistema, a pesar de las limitaciones impuestas por las aproximaciones de linealización y las no linealidades presentes en el sistema real.

## II. Marco Teórico

### *II-A. Modelado del sistema y formulación de Lagrange*

El péndulo de Furuta es un sistema mecánico sub-actuado con dos grados de libertad: el ángulo del brazo rotatorio ($\phi$) y el ángulo del péndulo ($\theta$). La dinámica del sistema se deriva utilizando la formulación de Lagrange, la cual permite obtener las ecuaciones de movimiento a partir de las energías cinética y potencial del sistema.

Las ecuaciones de Lagrange para un sistema conservativo se expresan como:

$$\frac{d}{dt}\left(\frac{\partial L}{\partial \dot{q}_i}\right) - \frac{\partial L}{\partial q_i} = Q_i$$

donde $L = T - V$ es el Lagrangiano (diferencia entre energía cinética $T$ y potencial $V$), $q_i$ son las coordenadas generalizadas ($\theta$ y $\phi$), y $Q_i$ son las fuerzas generalizadas no conservativas (en este caso, el torque aplicado $u$).

A partir de las características geométricas y físicas del sistema, se definen los siguientes parámetros:

- $l = 0{,}413\ [m]$: Longitud del péndulo
- $M = 0{,}01\ [kg]$: Masa de la bola en el extremo del péndulo
- $J_P = 9 \times 10^{-4}\ [kgm^2]$: Momento de inercia del péndulo
- $r = 0{,}235\ [m]$: Radio del brazo rotatorio
- $J = 0{,}05\ [kgm^2]$: Momento de inercia del brazo
- $m = 0{,}02\ [kg]$: Masa adicional distribuida en el brazo

Con estos parámetros se definen las siguientes constantes intermedias que simplifican la escritura de las ecuaciones de movimiento:

$$a = J_P + Ml^2 = 2{,}60569 \times 10^{-3}\ [kgm^2] \quad (1)$$

$$b = J + Mr^2 + mr^2 = 0{,}05165675\ [kgm^2] \quad (2)$$

$$c = Mrl = 9{,}7055 \times 10^{-4}\ [kgm^2] \quad (3)$$

$$d = lg\left(M + \frac{m}{2}\right) = 0{,}08100582\ \left[kgm^2/s^2\right] \quad (4)$$

Las ecuaciones de movimiento no lineales resultantes son:

$$a\ddot{\theta} - a\dot{\phi}^2 \sin(\theta)\cos(\theta) + c\ddot{\phi}\cos(\theta) - d\sin(\theta) = 0 \tag{5}$$

$$\begin{aligned} c\ddot{\theta}\cos(\theta) - c\dot{\theta}^2\sin(\theta) + 2a\dot{\theta}\dot{\phi}\sin(\theta)\cos(\theta) \\ + \left(b + a\sin^2(\theta)\right)\ddot{\phi} = u \end{aligned} \tag{6}$$

donde $u$ representa el torque aplicado al brazo rotatorio.

### *II-B. Representación en espacio de estados*

Un sistema dinámico lineal se puede representar en espacio de estados mediante las ecuaciones:

$$\dot{x} = Ax + Bu, \quad y = Cx + Du$$

donde $x$ es el vector de estados, $u$ es la entrada de control, $y$ es la salida, y $A$, $B$, $C$, $D$ son matrices constantes.

Para el péndulo de Furuta, el vector de estados se define como:

$$x = \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix} = \begin{bmatrix} \theta \\ \dot{\theta} \\ \phi \\ \dot{\phi} \end{bmatrix}$$

### *II-C. Análisis de estabilidad mediante polos del sistema*

La estabilidad de un sistema lineal depende de la ubicación de sus polos en el plano complejo. Los polos del sistema son los autovalores (eigenvalores) de la matriz $A$, obtenidos resolviendo:

$$\det(sI - A) = 0 \tag{7}$$

Un sistema es estable si todos sus polos tienen parte real negativa (plano continuo) o módulo menor que 1 (plano discreto).

### *II-D. Controlabilidad del sistema*

La controlabilidad determina si es posible llevar el sistema desde cualquier estado inicial a cualquier estado final en tiempo finito mediante una entrada adecuada. El criterio de controlabilidad de Kalman establece que un sistema es completamente controlable si la matriz de controlabilidad:

$$\mathcal{C} = \begin{bmatrix} B & AB & A^2B & \dots & A^{n-1}B \end{bmatrix} \tag{8}$$

tiene rango completo (igual al número de estados $n$).

### *II-E. Controlador LQR (Linear Quadratic Regulator)*

El controlador LQR es una estrategia de control óptimo que minimiza un funcional de costo cuadrático:

$$J = \int_0^\infty \left(x^T Q x + u^T R u\right) dt \tag{9}$$

donde $Q$ es una matriz semidefinida positiva que penaliza las desviaciones del estado deseado, y $R$ es una matriz definida positiva que penaliza el esfuerzo de control.

La solución óptima del problema LQR resulta en una ley de control por realimentación de estados:

$$u = -Kx$$

donde la matriz de ganancias $K$ se obtiene resolviendo la ecuación algebraica de Riccati:

$$A^T P + PA - PBR^{-1}B^T P + Q = 0$$

y luego calculando:

$$K = R^{-1}B^T P$$

### *II-F. Discretización de sistemas continuos*

Para implementar un controlador diseñado en tiempo continuo en un sistema digital, es necesario discretizar el modelo. Dos métodos comunes son:

*II-F1. Método ZOH (Zero-Order Hold):* Asume que la entrada se mantiene constante entre instantes de muestreo. Las matrices discretas se calculan como:

$$A_d = e^{AT_s}, \quad B_d = \left(\int_0^{T_s} e^{A\tau} d\tau\right) B$$

donde $T_s$ es el período de muestreo.

*II-F2. Método de Tustin (Aproximación trapezoidal):* Utiliza la aproximación bilineal $s = \frac{2}{T_s}\frac{z-1}{z+1}$. Las matrices discretas se calculan como:

$$A_d = (I - A\frac{T_s}{2})^{-1}(I + A\frac{T_s}{2}), \quad B_d = (I - A\frac{T_s}{2})^{-1}BT_s$$

Ambos métodos transforman el sistema continuo $\dot{x} = Ax + Bu$ al sistema discreto $x[k+1] = A_d x[k] + B_d u[k]$, permitiendo su implementación en plataformas digitales como Arduino.

## III. Desarrollo

### *III-A. Linealización del modelo*

La linealización se realiza alrededor de la posición vertical del péndulo ($\theta = 0°$), utilizando aproximaciones de pequeño ángulo; las funciones trigonométricas quedan como: $\sin(\theta) \approx \theta$ y $\cos(\theta) \approx 1$. Reemplazándolos en las Ecuaciones 5 y 6:

$$a\ddot{\theta} - a\dot{\phi}^2\theta + c\ddot{\phi} - d\theta = 0$$

$$c\ddot{\theta} - c\dot{\theta}^2\theta + 2a\dot{\theta}\dot{\phi}\theta + b\ddot{\phi} + a\ddot{\phi}\theta^2 = u$$

Hay que tomar en cuenta que los productos entre las variables de interés: $\theta$, $\dot{\theta}$, $\phi$ y $\dot{\phi}$, se eliminan, ya que estos casos no deben ocurrir en un modelo lineal. Con todas estas consideraciones, las Ecuaciones 5 y 6 se reducen a lo siguiente:

$$a\ddot{\theta} + c\ddot{\phi} = d\theta \tag{10}$$

$$c\ddot{\theta} + b\ddot{\phi} = u \tag{11}$$

Las Ecuaciones 10 y 11 se reescriben como una matriz para hacer el espacio de estados:

$$\begin{bmatrix} a & c \\ c & b \end{bmatrix} \cdot \begin{bmatrix} \ddot{\theta} \\ \ddot{\phi} \end{bmatrix} = \begin{bmatrix} d\theta \\ u \end{bmatrix} \tag{12}$$

Para despejar $\ddot{\theta}$ y $\ddot{\phi}$, se invierte la siguiente matriz:

$$A = \begin{bmatrix} a & c \\ c & b \end{bmatrix}$$

$$A^{-1} = \frac{1}{ab - c^2} \begin{bmatrix} b & -c \\ -c & a \end{bmatrix}$$

Resuelto para $\ddot{\theta}$ y $\ddot{\phi}$:

$$\begin{bmatrix} \ddot{\theta} \\ \ddot{\phi} \end{bmatrix} = \frac{1}{ab - c^2} \begin{bmatrix} b & -c \\ -c & a \end{bmatrix} \cdot \begin{bmatrix} d\theta \\ u \end{bmatrix} \tag{13}$$

En base a la Ecuación 13, se extrae lo siguiente:

$$\begin{aligned} \ddot{\theta} &= \frac{bd}{ab - c^2}\theta - \frac{c}{ab - c^2}u \\ \ddot{\theta} &= \frac{(0{,}0517)(0{,}081)}{(0{,}0026)(0{,}0517) - (0{,}00097)^2}\theta - \\ &\frac{(0{,}00097)}{(0{,}0026)(0{,}0517) - (0{,}00097)^2}u \\ \ddot{\theta} &= (31{,}3071)\theta - (7{,}2614)u \end{aligned} \tag{14}$$

$$\begin{aligned} \ddot{\phi} &= -\frac{cd}{ab - c^2}\theta + \frac{a}{ab - c^2}u \\ \ddot{\theta} &= -\frac{(0{,}00097)(0{,}081)}{(0{,}0026)(0{,}0517) - (0{,}00097)^2}\theta + \\ &\frac{(0{,}0026)}{(0{,}0026)(0{,}0517) - (0{,}00097)^2}u \\ \ddot{\theta} &= -(0{,}5882)\theta + (19{,}4950)u \end{aligned} \tag{15}$$

La matriz de estados se define como:

$$x = \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix} = \begin{bmatrix} \theta \\ \dot{\theta} \\ \phi \\ \dot{\phi} \end{bmatrix} \tag{16}$$

Entonces:

$$\begin{aligned} \dot{x_1} &= x_2 = \dot{\theta} \\ \dot{x_2} &= \ddot{\theta} \\ \dot{x_3} &= x_4 = \dot{\phi} \\ \dot{x_4} &= \ddot{\phi} \end{aligned} \tag{17}$$

Así, el espacio de estados del modelo linealizado queda:

$$\dot{x} = \begin{bmatrix} \dot{x_1} \\ \dot{x_2} \\ \dot{x_3} \\ \dot{x_4} \end{bmatrix} = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 31{,}3071 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ -0{,}5882 & 0 & 0 & 0 \end{bmatrix} \cdot \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix} + \begin{bmatrix} 0 \\ -7{,}2614 \\ 0 \\ 19{,}4950 \end{bmatrix} \cdot u \tag{18}$$

En el programa MATLAB, es posible realizar el espacio de estados con el comando:

```
ss(A,B,C,D);
```

Donde:

$$A = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 31{,}3071 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ -0{,}5882 & 0 & 0 & 0 \end{bmatrix} \tag{19}$$

$$B = \begin{bmatrix} 0 \\ -7{,}2614 \\ 0 \\ 19{,}4950 \end{bmatrix} \tag{20}$$

$$C = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{21}$$

$$D = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \end{bmatrix} \tag{22}$$

La matriz C (Matriz 21) es una matriz identidad de 4x4, ya que se medirán u observarán todas las variables.
Como el motor no afecta instantáneamente la posición, la matriz D es cero (Matriz 22).

### *III-B. Inestabilidad y controlabilidad*

Antes de continuar en la programación del controlador en MATLAB, se debe comprobar que el sistema es inestable, pero controlable. Para ello, primero, se obtienen los polos del sistema, es decir, los eigenvalores de la Matriz 19, con la Ecuación 7.

$$sI - A = \begin{bmatrix} s & -1 & 0 & 0 \\ -31{,}3071 & s & 0 & 0 \\ 0 & 0 & s & -1 \\ 0{,}5882 & 0 & 0 & s \end{bmatrix} \tag{23}$$

Para calcular la determinante de la Matriz 23:

$$\det(sI - A) = \det \begin{bmatrix} s & -1 \\ -31{,}3071 & s \end{bmatrix} \cdot \det \begin{bmatrix} s & -1 \\ 0 & s \end{bmatrix}$$

Resulta en:

$$\det(sI - A) = s^2(s^2 - 31{,}3071) \tag{24}$$

Entonces, los polos del sistema son los siguientes:

$$\begin{aligned} s_{1,2} &= 0 \\ s_{3,4} &= \pm\sqrt{31{,}3071} = \pm 5{,}5953 \end{aligned} \tag{25}$$

Sí es necesario un controlador, ya que existe un polo en la zona positiva ($s_4 = +5{,}5953$).
Es posible utilizar el comando de MATLAB:

```
eig(A);
```

Para obtener los polos de esta primera parte.
Ahora, para verificar la controlabilidad, se resuelve la Matriz 8, siendo:

$$AB = \begin{bmatrix} -7{,}2614 \\ 0 \\ 19{,}4950 \\ 0 \end{bmatrix}$$

$$A^2 = \begin{bmatrix} 31{,}3071 & 0 & 0 & 0 \\ 0 & 31{,}3071 & 0 & 0 \\ -0{,}5882 & 0 & 0 & 0 \\ 0 & -0{,}5882 & 0 & 0 \end{bmatrix}$$

$$A^2B = \begin{bmatrix} 0 \\ -227{,}3325 \\ 0 \\ 4{,}2712 \end{bmatrix}$$

$$A^3 = \begin{bmatrix} 0 & 31{,}3071 & 0 & 0 \\ 980{,}1372 & 0 & 0 & 0 \\ 0 & -0{,}5882 & 0 & 0 \\ -18{,}4153 & 0 & 0 & 0 \end{bmatrix}$$

$$A^3B = \begin{bmatrix} -227{,}3325 \\ 0 \\ 4{,}2712 \\ 0 \end{bmatrix}$$

Finalmente, la matriz de controlabilidad es:

$$\begin{bmatrix} 0 & -7{,}2614 & 0 & -227{,}3325 \\ -7{,}2614 & 0 & -227{,}3325 & 0 \\ 0 & 19{,}4950 & 0 & 4{,}2712 \\ 19{,}4950 & 0 & 4{,}2712 & 0 \end{bmatrix} \quad (26)$$

Como se observa, su rango es igual a 4, el mismo número de estados, por tanto, el sistema sí es controlable.
La Matriz 26 es posible de obtener en MATLAB con:

```
ctrb(A,B);
```

Y para calcular su rango:

```
rank(ctrb(A,B));
```

### *III-C. Controlador LQR*

Se eligen matrices Q y R para prueba:

$$Q = \begin{bmatrix} 10 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0{,}1 \end{bmatrix} \quad (27)$$

$$R = \begin{bmatrix} 1 \end{bmatrix} \quad (28)$$

La Matriz 27 indica la priorización de la estabilización del péndulo, frente al movimiento del brazo, teniendo las variables $\theta$ y $\dot{\theta}$ una mayor ponderación (10 y 1) en comparación con sus semejantes $\phi$ y $\dot{\phi}$ (1 y 0.1).
Se resuelve la ecuación de Riccati con el comando:

```
[K_lqr, S_lqr, eig_lqr] = lqr(A, B, Q, R);
```

Donde $K_{LQR}$ es la ganancia de realimentación de estados óptima, $S_{LQR}$ es la solución completa de la ecuación de Riccati, y $eig_{LQR}$ es el conjunto de eigenvalores de: $A-K_{LQR}\cdot B$, es decir, los polos del sistema en lazo cerrado.
Ahora, la nueva matriz de ganancias será:

$$K = \begin{bmatrix} -28{,}6407 & -5{,}197 & -1 & -0{,}8264 \end{bmatrix} \quad (29)$$

Con ello, se tiene una nueva matriz A:

$$A_{LQR} = \begin{bmatrix} 0 & 1 & 0 & 0 \\ -176{,}6634 & -37{,}7373 & -7{,}2614 & -6{,}001 \\ 0 & 0 & 0 & 1 \\ 557{,}762 & 101{,}3154 & 19{,}495 & 16{,}1113 \end{bmatrix} \quad (30)$$

Los polos en lazo cerrado son:

$$\begin{aligned} s_{1,2} &= -3{,}5049 \pm (1{,}773)i \\ s_3 &= -3{,}5498 \\ s_4 &= -11{,}0663 \end{aligned} \quad (31)$$

Con estos resultados es posible concluir que el sistema ya es estable, pero más adelante se analizarán los resultados hasta este punto.

### *III-D. Discretización*

El controlador LQR de la anterior sub-sección es uno en tiempo continuo. Para utilizarlo dentro del prototipo real, se debe discretizar, en este caso, con un tiempo de muestreo: $T_s = 2$ [ms].
Entre los métodos de discretización escogidos para este trabajo son: ZOH y Tustin.
Con el Método ZOH:

$$K_d = \begin{bmatrix} -7{,}1404 & -1{,}3194 & -0{,}4115 & -0{,}2977 \end{bmatrix} \quad (32)$$

$$A_d = \begin{bmatrix} 0{,}9995 & 0{,}0019 & 0 & 0 \\ -0{,}4475 & 0{,}9072 & -0{,}2 & -0{,}016 \\ 0{,}0014 & 0{,}0003 & 1{,}0001 & 0{,}002 \\ 1{,}3685 & 0{,}2494 & 0{,}0538 & 1{,}0429 \end{bmatrix} \quad (33)$$

$$\begin{aligned} s_1 &= 0{,}9967 \\ s_{2,3} &= 0{,}9962 \pm (0{,}0018)i \\ s_4 &= 0{,}9856 \end{aligned} \quad (34)$$

Estos polos se encuentran en el Plano Z, y sus módulos indican que están dentro del circulo unitario, por tanto, el sistema es estable.
Con el Método de Tustin:

$$K_d = \begin{bmatrix} -7{,}0931 & -1{,}3107 & -0{,}4088 & -0{,}2958 \end{bmatrix} \quad (35)$$

$$A_d = \begin{bmatrix} 0{,}9996 & 0{,}0019 & 0 & 0 \\ -0{,}4476 & 0{,}9072 & -0{,}02 & -0{,}016 \\ 0{,}0014 & 0{,}0002 & 1{,}0001 & 0{,}002 \\ 1{,}3686 & 0{,}2494 & 0{,}0538 & 1{,}0429 \end{bmatrix} \tag{36}$$

$$\begin{aligned} s_1 &= 0{,}9933 \\ s_{2,3} &= 0{,}9925 \pm (0{,}0036)i \\ s_4 &= 0{,}9714 \end{aligned} \tag{37}$$

Estas respuestas indican que el sistema discreto tendrá una dinámica lenta, respuesta suave, o también entendido como poco esfuerzo de control.
Se continuarán con estos resultados para la implementación real, ya que garantizan estabilidad. En las posteriores respuestas puede que se modifiquen estos valores.

## IV. Análisis de resultados

### *IV-A. Verificación del modelo linealizado*

El documento base otorga la siguiente ganancia por realimentación de estados:

$$K = \begin{bmatrix} -7{,}5343 & -1{,}3465 & 0 & -0{,}2216 \end{bmatrix}$$

Aplicando esta ganancia al espacio de estados de la Ecuación 18:

$$\begin{aligned} A_{\text{NEW}} &= A - K \cdot B \\ A_{\text{NEW}} &= \begin{bmatrix} 0 & 1 & 0 & 0 \\ -23{,}4021 & -9{,}7774 & 0 & -1{,}6091 \\ 0 & 0 & 0 & 1 \\ 146{,}2928 & 26{,}25 & 0 & 4{,}3201 \end{bmatrix} \end{aligned} \tag{38}$$

Con el sistema iniciando en el estado de reposo, y el ángulo del péndulo en 5°, o 0.08727 radianes, se obtiene el siguiente comportamiento dentro de la simulación:

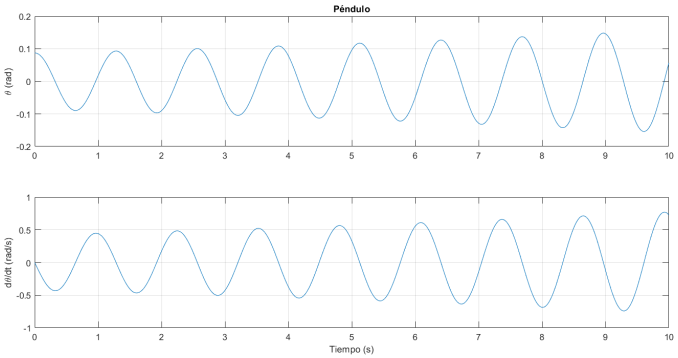


Figura 1. Comportamiento del péndulo con realimentación propuesta.

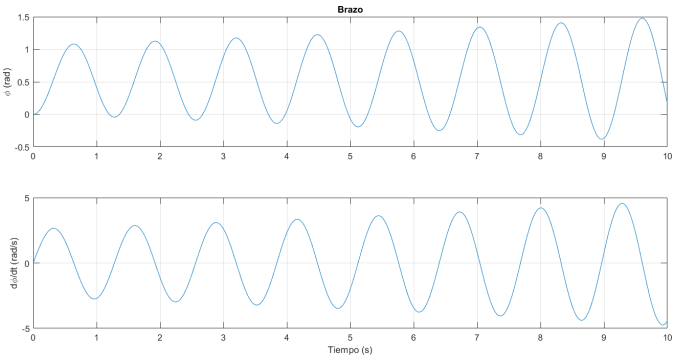


Figura 2. Comportamiento del péndulo con realimentación propuesta.

En las Figuras 1 y 2 se muestra el resultado en la posición y velocidad angular del péndulo y del brazo, presentando oscilaciones que no se amortiguan, sino que tienen una amplitud que crece lentamente. Con esta progresiva divergencia, se dice que el sistema çasi se estabiliza".
Este es un resultado esperable al observar sus polos:

$$\begin{aligned} s_1 &= 0 \\ s_2 &= -5{,}5782 \\ s_{3,4} &= 0{,}0604 \pm (4{,}9064)i \end{aligned} \tag{39}$$

En dichas gráficas se muestran las simulaciones realizadas en un intervalo de tiempo reducido (10 segundos), dado que el sistema fue linealizado en las cercanías del punto de equilibrio; un desvío mayor a $\pm 10°$, ó $\pm 0.1745$ radianes, ya no corresponde al rango de validez del modelo linealizado.
Las unidades de medidas utilizadas, tanto en los cálculos como en las gráficas, son radianes (rad) y radianes por segundo (rad/s).

### *IV-B. Validación del controlador LQR mediante simulación*

Con la Matriz 30 se obtienen los siguientes resultados:

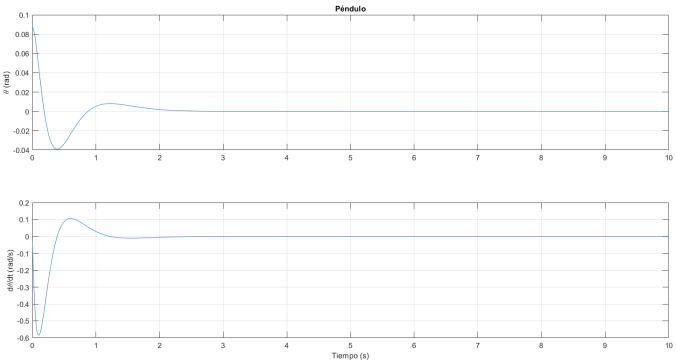


Figura 3. Comportamiento del péndulo con Matriz 30.

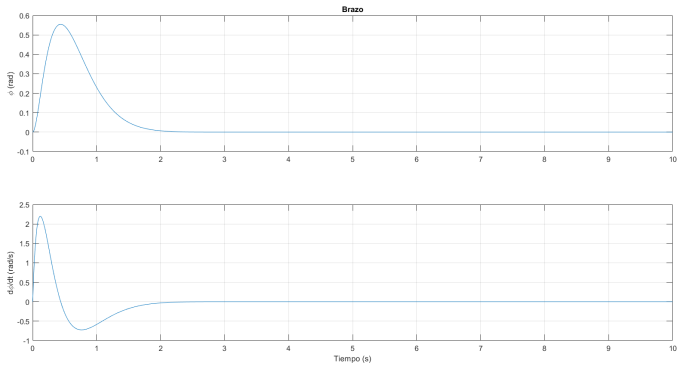


Figura 4. Comportamiento del brazo con Matriz 30.

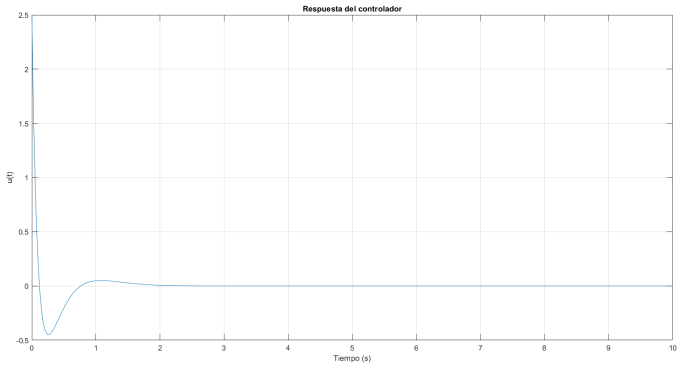


Figura 5. Respuesta del controlador con Matriz 30.

En las Figuras 3, 4 y 5 se observan sobreimpulsos, pero con una convergencia rápida, menor a 2 segundos, por tanto es un resultado válido. Sin embargo, para tener resultados más claros, se utiliza MATLAB.
En este punto se enfocará en la variable de máximo interés, $\theta$, el ángulo del péndulo.

```
theta=x(:,1);
```

Para obtener el último valor de la prueba, y con ello definir el error en estado estacionario ($e_{ss}$), se utiliza:

```
ess=abs(theta(end));
```

El resultado obtenido es $7{,}9307 \times 10^{-16}$, que es lo mismo que tener un error estacionario del $0\,\%$. En simulación, el controlador logra estabilizar el péndulo exactamente en el equilibrio deseado; en simulación le espera problemas con la fricción, el ruido y enfrentar la resolución del encoder.
Para obtener el sobreimpulso máximo:

```
Mp=(max(abs(theta))-theta(1))/theta(1)*100;
```

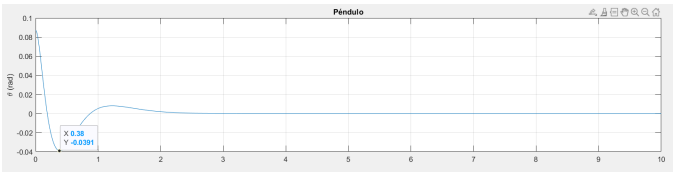


Figura 6. Valor mínimo en el ángulo del péndulo.

En la Figura se observa un ángulo de -0.0391 radianes, o $-2{,}2403°$. Es decir, que el péndulo inició en $5°$, pasó por $0°$, luego se desvió hasta $-2{,}2403°$, y finalmente se logró estabilizar en $0°$.
Considerando que el punto final deseado es de 0 radianes, entonces un desvío que termine en $\pi$ radianes es un sobreimpulso del $100\,\%$. En base a esto:

$$M_p = \frac{0{,}0391}{\pi} \cdot 100\,\% = 1{,}25\,\% \tag{40}$$

Dentro de la simulación, la respuesta presenta un sobreimpulso casi imperceptible.
En cuanto al esfuerzo máximo de control:

```
u_max=abs(max(u));
```

Obteniendo, 2.4994.
Se considera el rango de valores positivos y negativos en la respuesta del controlador. Tanto en la Figura 5 como obtenido en MATLAB, da un resultado cercano a 2.5, lo que quiere decir que el actuador requiere un esfuerzo moderado. Este análisis es necesario en la implementación real, ayudando en el criterio de elección de umbrales de saturación, para proteger el motor.
Ahora, con una matriz $R = [100]$, se obtiene:

$$K = \begin{bmatrix} -12{,}3494 & -2{,}211 & -0{,}1 & -0{,}1423 \end{bmatrix} \tag{41}$$

$$A_{LQR} = \begin{bmatrix} 0 & 1 & 0 & 0 \\ -58{,}3664 & -16{,}0548 & -0{,}7261 & -1{,}0336 \\ 0 & 0 & 0 & 1 \\ 240{,}1634 & 43{,}1033 & 1{,}9495 & 2{,}7749 \end{bmatrix} \tag{42}$$

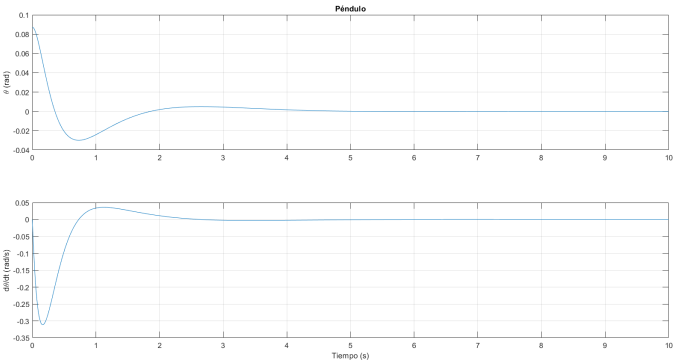


Figura 7. Comportamiento del péndulo con Matriz 42.

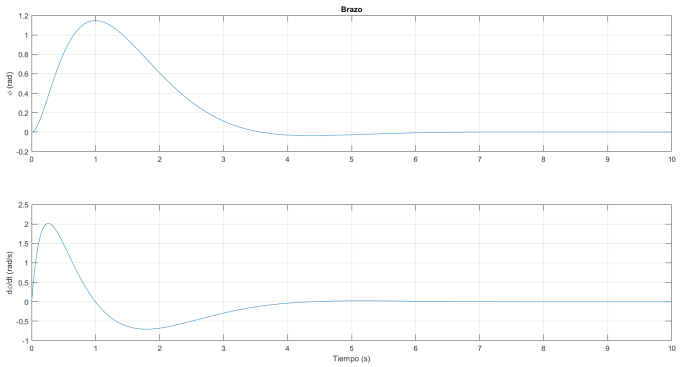


Figura 8. Comportamiento del brazo con Matriz 42.

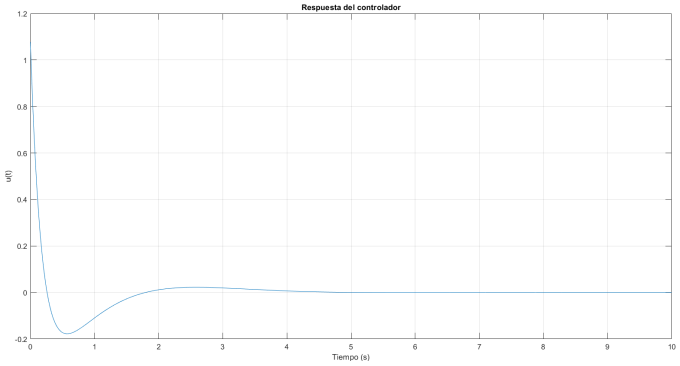


Figura 9. Respuesta del controlador con Matriz 42.

En las Figuras 7, 8 y 9 se tienen respuestas más amortiguadas a las anteriores, debido a la mayor penalización en el controlador, pero un LQR no busca mínimo sobreimpulso, sino equilibrio. Eso sí, los resultados son más lentos, ya que llega a la convergencia alrededor de los 4 segundos.
Para ser más exactos, el sobreimpulso obtenido es del $0{,}9528\,\%$, siendo un cuarto inferior al anterior caso. Mientras que el esfuerzo de control máximo es del 1.0777, que es un cambio significativo para el actuador. El problema está en su error de estado estacionario, ya que es superior en esta situación, $e_{ss} = 3{,}658 \times 10^{-6}$.
Entonces, al aumentar el valor de la Matriz R, se logra una disminución significativa tanto del sobreimpulso como de la acción máxima aplicada por parte del controlador (siendo este aspecto la prioridad de la Matriz R). Es verdad que existe un aumento en el error estacionario, pero este sigue siendo despreciable.

### *IV-C. Validación del control en el prototipo real*

Con los valores obtenidos con el modelo linealizado se procede a desarrollar un controlador LQR sobre la plataforma Arduino. Esto se realiza a partir de la solución encontrada para $K$ que garantiza estabilidad en lazo cerrado.

El controlador LQR implementado en el código Arduino (***Ver Anexos***) está diseñado para estabilizar un péndulo de Furuta en su posición vertical inestable. Su funcionamiento se basa en las siguientes etapas clave:

*IV-C1. Relación con el modelo linealizado:* El controlador utiliza la ley de control óptima derivada de la teoría LQR para sistemas lineales:

$$u(t) = -Kx(t) = -[k_1 \quad k_2 \quad k_3 \quad k_4] \begin{bmatrix} \theta \\ \dot{\theta} \\ \phi \\ \dot{\phi} \end{bmatrix}$$

donde:

- $\theta$: Ángulo del péndulo respecto a la vertical (rad)
- $\dot{\theta}$: Velocidad angular del péndulo (rad/s)
- $\phi$: Ángulo del brazo rotatorio (rad)
- $\dot{\phi}$: Velocidad angular del brazo (rad/s)

Las ganancias $K$ implementadas en el arreglo `const float K[4]` fueron calculadas resolviendo la ecuación algebraica de Riccati para el modelo linealizado del sistema alrededor del punto de operación $(\theta, \dot{\theta}, \phi, \dot{\phi}) = (0, 0, 0, 0)$.

El código implementa una arquitectura de control con los siguientes componentes:

*IV-C2. Estimación de estados:*

- **Péndulo**: El ángulo $\theta$ se obtiene mediante un potenciómetro, aplicando conversión ADC y linealización limitada a $\pm 0{,}7$ rad ($\approx 40°$).
- **Brazo**: La posición $\phi$ se calcula a partir de un encoder incremental con 2000 pulsos por revolución.
- **Velocidades**: $\dot{\theta}$ y $\dot{\phi}$ se estiman por diferenciación numérica con un filtro pasa-bajos de primer orden con un coeficiente $\alpha = 0{,}3$.

*IV-C3. Cálculo de la acción de control:* En la función `computeControlAction()`, se implementa:

$$u = -S \cdot (k_1\theta + k_2\dot{\theta} + k_3\phi + k_4\dot{\phi})$$

donde $S = 0{,}2$ es un factor de escalado para suavizar la respuesta inicial. Se incluye una zona muerta de $\pm 0{,}2$V para evitar oscilaciones por ruido de medición.

*IV-C4. Discretización temporal:* El controlador opera a 500 Hz (periodo $T_s = 2$ ms) mediante la interrupción por timer. Las ganancias $K$ ya están ajustadas para esta frecuencia de muestreo, considerando la discretización por retenedor de orden cero (ZOH) del modelo continuo original.

*IV-C5. Actuación:* La señal de control $u$ (en voltios) se convierte a señal PWM para un puente H, con saturación a $\pm 5$V para proteger el motor.

### *IV-D. Desviaciones del punto de linealización*

Cuando el estado del sistema se aleja significativamente del punto de linealización ($\theta = 0$), ocurren los siguientes efectos:

*IV-D1. Pérdida de optimalidad:* Las ganancias $K$ fueron optimizadas para el modelo linealizado válido alrededor de $\theta \approx 0$. Para ángulos grandes, el modelo real presenta:

- No linealidades.
- Acoplamientos dinámicos no modelados
- Efectos centrífugos despreciados en la linealización

Esto reduce la optimalidad del controlador, aunque puede mantener cierta capacidad de estabilización.

*IV-D2. Limitaciones en la estimación:* La función `estimatePendulumAngle()` limita artificialmente la lectura a $\pm 0{,}7$ rad para mantener cierta linealidad en la conversión ADC. Para ángulos mayores, esta aproximación introduce errores significativos.

*IV-D3. Mecanismo de seguridad:* El sistema incluye un **freno de emergencia** que se activa cuando $|\theta| > 30°$ (0.5236 rad). Esta medida previene daños al hardware cuando el controlador LQR, diseñado para pequeñas perturbaciones, no puede recuperar el sistema. Esta parte del diseño es importante puesto que evita oscilaciones violentas en el sistema.

*IV-D4. Degradación del desempeño:* Con ángulos moderados ($|\theta| < 30°$ pero $> 10°$), el controlador puede exhibir:

- Mayor sobreimpulso en la respuesta
- Tiempos de estabilización prolongados
- Posibles oscilaciones sostenidas

### *IV-E. Consideraciones de implementación*

- El factor `GAIN_SCALE = 0.75` permite atenuar inicialmente la acción de control para pruebas seguras.
- La diferenciación numérica para velocidades es sensible al ruido, por lo que es necesario una etapa de filtrado.
- La zona muerta introduce no linealidad pero mejora la vida útil del actuador.
- La saturación del actuador puede causar wind-up, esté parámetro no fue considerado en el diseño LQR.

Figura 10. Péndulo de Furuta en funcionamiento.

En la Figura 10 se muestra el péndulo de Furuta manteniendo el equilibrio en posición vertical. Este es el resultado del proceso de linealización y control mediante LQR. Las características extraídas del modelo no lineal han sido lo suficientemente precisas como para permitir que el péndulo logre mantenerse vertical. El controlador LQR es una herramienta bastante robusta para la ingeniería de control, ya que se evidenció cómo el sistema fue controlado casi al instante de introducir las ganancias K calculadas.

Un factor de escala *GAIN_SCALE* fue añadido en el código para ajustar la magnitud del vector de ganancias K; esto fue todo lo que se realizó durante el ajuste fino para mejorar el control del péndulo. Esto demuestra que el controlador LQR se compone de un método bastante preciso que define muy bien el comportamiento del sistema. Sin embargo, estas características se pierden si el sistema se aleja demasiado de su punto de linealización y, por ende, el controlador no puede actuar correctamente sobre zonas donde el comportamiento es muy distinto al lineal.

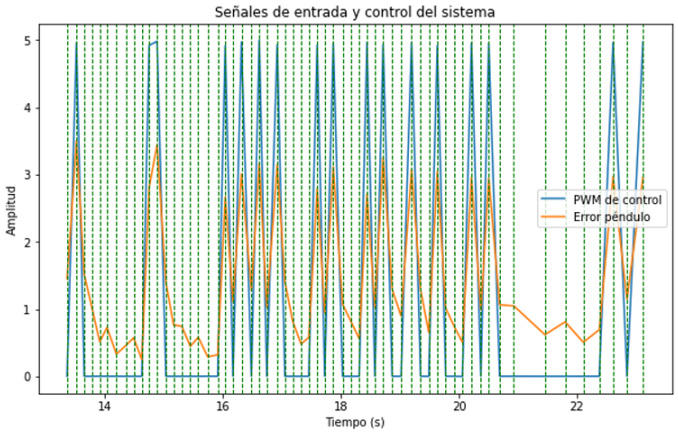


Figura 11. Señal PWM generada por el controlador LQR y Error en la posición del péndulo con respecto al punto de equilibrio.

En la Figura 11 se muestran las señales del valor absoluto del error en la posición del péndulo y la respuesta generada por el control LQR en esos instantes, expresada como una señal PWM.

La razón por la que se observa una señal PWM en lugar de una señal analógica en la salida se debe a la forma en que Arduino envía una señal analógica a través de sus pines. En realidad, Arduino genera una señal PWM con un ciclo de trabajo (*duty cycle*) específico en función del valor analógico requerido. El motor se moverá según el valor promedio de esta señal, por lo que el ciclo de trabajo del PWM es directamente proporcional al valor de voltaje analógico establecido en el código.

Refiriéndonos a la Figura 11, se observa que existe una señal PWM cuando el error excede un umbral. Esto se debe a que mediante mediciones sobre el péndulo se ha detectado que el valor de voltaje del potenciómetro no es completamente fijo

en su posición vertical, sino que existe un rango de valores dentro del cual el péndulo se mantiene en equilibrio.

Además, se ha establecido una medida de seguridad en el código con un valor umbral para el cálculo de la señal de control. Con un error muy grande, el ciclo de trabajo de la PWM sería elevado, lo que generaría un giro muy fuerte del motor, desestabilizando el sistema y provocando daños en los componentes. En la Figura 11 se observa que cuando el error (curva de color naranja) aumenta, se genera una señal PWM en cierto sentido de giro que reduce el error, manteniendo el péndulo relativamente en equilibrio hasta que el error vuelve a elevarse y el algoritmo de control actúa para reducirlo nuevamente.

## V. Conclusiones

El presente trabajo permitió diseñar, simular e implementar exitosamente un controlador LQR para estabilizar un péndulo de Furuta en su posición vertical invertida. La metodología inició con la obtención del modelo matemático no lineal mediante la formulación de Lagrange, seguida de su linealización alrededor del punto de equilibrio. El análisis de controlabilidad confirmó que el sistema linealizado puede ser estabilizado mediante realimentación de estados. El diseño del controlador LQR, mediante la solución de la ecuación algebraica de Riccati, generó ganancias óptimas que garantizan estabilidad en lazo cerrado con un equilibrio adecuado entre el desempeño del sistema y el esfuerzo de control. La discretización de estas ganancias mediante los métodos ZOH y Tustin permitió su implementación en tiempo real sobre la plataforma Arduino. Los resultados experimentales demostraron que el controlador LQR es capaz de mantener la estabilidad local del sistema, validando así el modelo linealizado y el diseño del controlador. Sin embargo, se observó que el desempeño se degrada cuando el péndulo se aleja significativamente del punto de linealización, debido a las no linealidades despreciadas en el modelo. Esta limitación fue mitigada mediante la implementación de mecanismos de seguridad como el freno de emergencia. El trabajo constituye una aplicación completa de técnicas de control digital, desde el modelado matemático hasta la implementación práctica, evidenciando tanto las capacidades como las limitaciones del control LQR aplicado a sistemas no lineales reales.

## Referencias


[1] E. Hernández Márquez, R. Silva Ortigoza, P. Cruz Francisco, X. Siordia Vásquez, J. O. Millán Tejeda, and U. Santos López, "Simulación del péndulo de furuta: ensamble versus modelo matemático," *Boletín UPIITA*, vol. 17, no. 93, p. n.p., November 2022, accessed: December 30, 2025. [Online]. Available: https://www.boletin.upiita.ipn.mx/index.php?option=com_content&view=article&id=2105:simulacion-del-pendulo-de-furuta-ensamble-versus-modelo-matematico&catid=1022:cyt-numero-93&Itemid=156

## ANEXOS

### *V-A. Repositorio de GitHub (Código y video demostrativo):*

Click aquí.

### *V-B. Código del controlador LQR (Arduino)*

```
// CONTROLADOR LQR PARA PENDULO DE FURUTA
#include <Arduino.h>
#include <MsTimer2.h>

#define PWM 9
#define IN1 11
#define IN2 10
#define ENCODER_A 2
#define ENCODER_B 4
#define POT A5

// Parámetros del sistema
const float Ts = 0.002;  // 2 ms = 500 Hz
const float GAIN_SCALE = 0.75;  // Escalar valores de K

// Ganancia LQR discretizada (ZOH con Ts=2ms)
//const float K[4] = {-7.1404, -1.3194, -0.4115, -0.2977};
const float K[4] = {-7.0931, -1.3107, -0.4088, -0.2958};

// Variables de estado estimadas
float theta = 0.0;        // x1: Ángulo del péndulo (rad)
float theta_dot = 0.0;    // x2: Velocidad del péndulo (rad/s)
float phi = 0.0;          // x3: Ángulo del brazo (rad)
float phi_dot = 0.0;      // x4: Velocidad del brazo (rad/s)

// Variables anteriores para filtrado
float theta_prev = 0.0;
float phi_prev = 0.0;

// Variables de medición
int pendulum_raw = 0;     // Valor crudo del potenciómetro
int arm_counts = 10000;   // Conteo del encoder (posición inicial)

// Factores de conversión
const float ADC_TO_RAD = 0.3352;      // Valor de calibración calculado con el péndulo en
    posición vertical
const float ENCODER_TO_RAD = 2.0 * PI / 2000.0;  // 2000 conteo/rev

// Saturación
const float MAX_VOLTAGE = 5.0;      // Voltaje máximo al motor
const float DEAD_ZONE = 0.2;        // Zona muerta para evitar oscilaciones

// Filtros
const float ALPHA = 0.3;           // Para filtrado de velocidades

void timerCallback();
void READ_ENCODER_A();
void READ_ENCODER_B();
float estimatePendulumAngle(int raw_adc);
void estimateStates();
float computeControlAction();
void applyMotorControl(float u);

void setup() {
  Serial.begin(115200);

  // Configuración de pines
  pinMode(IN1, OUTPUT);
  pinMode(IN2, OUTPUT);
  pinMode(PWM, OUTPUT);
```

```
  pinMode(ENCODER_A, INPUT);
  pinMode(ENCODER_B, INPUT);

  // Inicialización
  analogWrite(PWM, 0);
  digitalWrite(IN1, LOW);
  digitalWrite(IN2, LOW);

  // Timer para control a 500 Hz (2 ms)
  delay(200);
  MsTimer2::set(2, timerCallback);  // 2 ms
  MsTimer2::start();

  // Interrupciones del encoder
  attachInterrupt(digitalPinToInterrupt(ENCODER_A), READ_ENCODER_A, CHANGE);
  attachInterrupt(digitalPinToInterrupt(ENCODER_B), READ_ENCODER_B, CHANGE);
}

void loop() {
  // El control se ejecuta en timerCallback
  delay(100);
}

// Convertir lectura ADC a ángulo del péndulo (radianes)
float estimatePendulumAngle(int raw_adc) {
  // ADC_VERTICAL es el valor cuando el péndulo está vertical
  const int ADC_VERTICAL = 758;  // Set point (Valor capturado por el ADC cuando el péndulo
      está en posición vertical)

  // Convertir a radianes (usa el factor de calibración)
  float angle = (raw_adc - ADC_VERTICAL) * ADC_TO_RAD;

  // Limitar para linealidad (aproximación de pequeño ángulo)
  return constrain(angle, -0.7, 0.7);  // +-28.6 grados
}

// Estimar estados (ángulos y velocidades)
void estimateStates() {
  // 1. Leer sensores
  pendulum_raw = analogRead(POT);

  // 2. Estimar ángulos
  theta = estimatePendulumAngle(pendulum_raw);
  phi = (arm_counts - 10000) * ENCODER_TO_RAD;  // Posición relativa

  // 3. Estimar velocidades por diferenciación con filtro
  static unsigned long last_time = 0;
  unsigned long current_time = micros();
  float dt = (current_time - last_time) / 1e6;

  if (dt > 0 && dt < 0.1) {  // Evitar divisiones por cero o dt grandes
    // Diferenciación
    float theta_dot_raw = (theta - theta_prev) / dt;
    float phi_dot_raw = (phi - phi_prev) / dt;

    // Filtrado pasa-bajos
    theta_dot = ALPHA * theta_dot + (1 - ALPHA) * theta_dot_raw;
    phi_dot = ALPHA * phi_dot + (1 - ALPHA) * phi_dot_raw;
  }

  // 4. Guardar valores anteriores
  theta_prev = theta;
  phi_prev = phi;
  last_time = current_time;
}

// Calcular acción de control LQR
```

```
float computeControlAction() {
  float u = -GAIN_SCALE * (K[0] * theta +
                           K[1] * theta_dot +
                           K[2] * phi +
                           K[3] * phi_dot);

  // Aplicar zona muerta para evitar oscilaciones pequeñas
  if (fabs(u) < DEAD_ZONE) {
    u = 0;
  }

  // Saturación
  return constrain(u, -MAX_VOLTAGE, MAX_VOLTAGE);
}

// Aplicar control al motor
void applyMotorControl(float u) {
  // Determinar dirección
  if (u > 0) {
    digitalWrite(IN1, HIGH);
    digitalWrite(IN2, LOW);
  } else if (u < 0) {
    digitalWrite(IN1, LOW);
    digitalWrite(IN2, HIGH);
  } else {
    // Frenar cuando u=0
    digitalWrite(IN1, LOW);
    digitalWrite(IN2, LOW);
  }

  // Convertir voltaje a PWM (0-255 para 0-5V)
  int pwm_value = (int)(fabs(u) * 255.0 / MAX_VOLTAGE);
  pwm_value = constrain(pwm_value, 0, 255);

  analogWrite(PWM, pwm_value);
}

// Frenado de emergencia
bool emergencyBrake() {
  // Si el péndulo se inclina demasiado (> 30 grados)
  if (fabs(theta) > 0.5236) {  // 30 grados en radianes
    digitalWrite(IN1, LOW);
    digitalWrite(IN2, LOW);
    analogWrite(PWM, 0);
    return true;
  }
  return false;
}

// Función principal del timer
void timerCallback() {
  // 1. Estimación de estados
  estimateStates();

  // 2. Verificar freno de emergencia
  if (emergencyBrake()) {
    Serial.println("Frenado de Emergencia!");
    return;
  }

  // 3. Calcular acción de control LQR
  float u = computeControlAction();

  // 4. Aplicar al motor
  applyMotorControl(u);

  // 5. Monitoreo (Serial)
```

```
  static int counter = 0;
  if (counter++ % 10 == 0) {  // Enviar cada 20ms
    Serial.print("Theta: ");
    Serial.print(theta, 4);
    Serial.print(" Phi: ");
    Serial.print(phi, 4);
    Serial.print(" U: ");
    Serial.println(u, 4);
  }
}

// Interrupciones del encoder
void READ_ENCODER_A() {
  if (digitalRead(ENCODER_A) == digitalRead(ENCODER_B)) {
    arm_counts++;
  } else {
    arm_counts--;
  }
}

void READ_ENCODER_B() {
  if (digitalRead(ENCODER_A) == digitalRead(ENCODER_B)) {
    arm_counts--;
  } else {
    arm_counts++;
  }
}
```