UNIVERSIDAD TÉCNICA DEL NORTE

FACULTAD DE INGENIERÍA EN CIENCIAS APLICADAS

ESCUELA DE INGENIERÍA EN MECATRÓNICA

TRABAJO DE GRADO PREVIO A LA OBTENCIÓN DEL TÍTULO DE INGENIERO EN MECATRÓNICA

TEMA:

"CONTROL APERIÓDICO DE POSICIÓN DE UN SERVOMOTOR"

AUTOR: EDGAR JAVIER ENRRIQUEZ QUILUMBAQUIN

DIRECTOR: CARLOS XAVIER ROSERO CHANDI

IBARRA-ECUADOR

FEBRERO 2019

UNIVERSIDAD TÉCNICA DEL NORTE

BIBLIOTECA UNIVERSITARIA

AUTORIZACIÓN DE USO Y PUBLICACIÓN

A FAVOR DE LA UNIVERSIDAD TÉCNICA DEL NORTE

1. IDENTIFICACIÓN DE LA OBRA

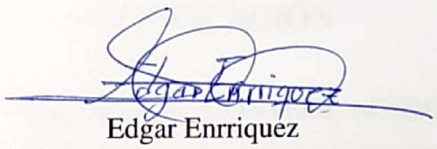
En cumplimiento del Art. 144 de la Ley de Educación Superior, hago la entrega del presente trabajo a la Universidad Técnica del Norte para que sea publicado en el Repositorio Digital Institucional, para lo cual pongo a disposición la siguiente información:

DATOS DEL AUTOR			
CÉDULA DE IDENTIDAD:	1725135071		
APELLIDOS Y NOMBRES:	EDGAR JAVIER ENRRIQUEZ QUILUMBAQUIN		
DIRECCIÓN:	ATUNTAQUI CALLE PICHINCHA Y MALDONADO		
EMAIL:	ejenriquezq@utn.edu.ec		
TELÉFONO FIJO:	062620116	**TELÉFONO MÓVIL:**	0969724906
DATOS DE LA OBRA			
TÍTULO:	"CONTROL APERIÓDICO DE POSICIÓN DE UN SERVOMOTOR"		
AUTOR:	EDGAR JAVIER ENRRIQUEZ QUILUMBAQUIN		
FECHA:	12-03-2019		
SÓLO PARA TRABAJOS DE GRADO			
PROGRAMA:	PREGRADO		
TÍTULO POR EL QUE OPTA:	INGENIERO EN MECATRÓNICA		
ASESOR/DIRECTOR:	CARLOS XAVIER ROSERO C.		

2. CONSTANCIAS

El autor manifiesta que la obra objeto de la presente autorización es original y se la desarrolló sin violar derechos de autor de terceros, por lo tanto la obra es original, y que es el titular de los derechos patrimoniales, por lo que asume la responsabilidad sobre el contenido de la misma y saldrá en defensa de la Universidad en caso de reclamación por parte de terceros.

Ibarra, a los 12 días del mes de marzo de 2019.



Edgar Enrriquez

C.I.: 1725135071

UNIVERSIDAD TÉCNICA DEL NORTE

FACULTAD DE INGENIERÍA EN CIENCIAS APLICADAS

CERTIFICACIÓN

En calidad de director del trabajo de grado "CONTROL APERIÓDICO DE POSICIÓN DE UN SERVOMOTOR", presentado por el egresado EDGAR JAVIER ENRRIQUEZ QUILUMBAQUIN, para optar por el título de Ingeniero en Mecatrónica, certifico que el mencionado proyecto fue realizado bajo mi dirección.

Ibarra, febrero de 2019

Carlos Xavier Rosero

DIRECTOR DE TESIS

UNIVERSIDAD TÉCNICA DEL NORTE

FACULTAD DE INGENIERÍA EN CIENCIAS APLICADAS

DECLARACIÓN

Yo, Edgar Javier Enrriquez Quilumbaquin con cédula de identidad Nro. 1725135071, declaro bajo juramento que el trabajo aquí escrito es de mi autoría; que no ha sido previamente presentado para ningún grado o calificación profesional; y, que he consultado las referencias bibliográficas que se incluyen en este documento.

A través de la presente declaración cedo mis derechos de propiedad intelectual correspondientes a este trabajo, a la Universidad Técnica del Norte - Ibarra, según lo establecido por la Ley de Propiedad Intelectual, por su Reglamento y por la normativa institucional vigente.

Ibarra, febrero de 2019

Edgar Javier Enrriquez Quilumbaquin

C.I.: 1725135071

Agradecimiento

Agradezco a cada persona que de una u otra manera ha estado pendiente de mi proceso de formación académica.

A mi hermana Victoria Enríquez por estar pendiente de mi desarrollo profesional, académico y por su contribución económica a lo largo de mi carrera como profesional .

A mi padre José Enrriquez por siempre darme ánimo a seguir sobre cualquier dificultad. A mi madre Rosa Quilumbaquin por inculcarme sus valores, principios, por enseñarme a tomar mis propias decisiones y siempre estar presente con el consejo correcto. Para Ana Bonilla por su comprensión y apoyo.

A cada docente que dedicó su tiempo a compartir su conocimiento, a mi director ingeniero Carlos Xavier Rosero por la confianza que me dio para desarrollar este proyecto.

A la Universidad Técnica del Norte, Facultad de Ingeniería en Ciencias Aplicadas, Carrera de Ingeniería en Mecatrónica y a sus respectivas autoridades, por su gratitud y su entera colaboración para culminar una meta más en mi vida.

Y en especial a la sustancia que impregna, penetra y llena todos los espacio del universo; mi Dios.

Dedicatoria

El presente trabajo esta dedicado al grupo SSPA, pasión por emprender que ha recopilado información sobre el estilo de vida de grandes mentes de la historia, para establecer un estilo de vida basado en elevar los principios del éxito: Salud espiritual, física, relaciones afectivas a largo plazo y libertad financiera a través del desarrollo de talleres de superación personal, formando grupos de trabajo y empresas a nivel nacional que quieran cambiar el estilo de vida de un sistema tradicional que limita el pensamiento a uno que explota la imaginación y creatividad.

Edgar Javier Enrriquez Quilumbaquin

Resumen

Aunque el control aperiódico es un tema que tienes algunos años de investigación, anteriormente era imposible implementarlo por los requerimientos en procesadores y cálculos matemáticos complejos de realizar. El avance tecnológico en software numéricos permite ahora nuevas alternativas para que la investigación en este campo avance.

El presente documento contiene el diseño e implementación de un controlador aperiódico de posición para un servomotor. Se empieza con una revisión literaria sobre controles aperiódicos actuales o llamados Event-based Control y se clasifican principalmente en: Event-Triggered control (control disparado por un evento) y Self-Triggered control (control auto disparado).

Luego se lleva acabó la identificación de un motor DC sobre este actúa el controlador de posición. Se utiliza la herramienta ident de matlab y se obtiene el modelo matemático del sistema en lazo abierto.

En la implementación se utiliza control aperiódico auto disparado (CAAD). Este se divide en dos etapas: la offline que es donde se encuentran los polinomios y valores de ganancias del controlador utilizando control LQR en matlab y la online que es donde se escribe en el microcontrolador los polinomios y ganancias anteriormente calculados y la teoría de muestreo aperiódico que calcula el periodo del siguiente muestreo basado en la estabilidad del sistema. Si el sistema es estable el periodo aumenta hasta alcanzar un valor máximo preestablecido y si es inestable o sigue un cambio de referencia disminuye hasta alcázar un valor mínimo también preestablecido en el diseño.

finalmente se demuestra que esta teoría de control se puede implementar en un servomecanismo ya que el sistema se estabiliza mientras varia el período de muestreo y se recomienda que en trabajos futuros se realicen pruebas físicas de desempeño.

Abstract

Although aperiodic control is not a new subject it have some years of research,but previously it was impossible to implement because it requiere high qualities in processors and complex mathematical calculations. Now technological advance in numerical software allows new alternatives to advance research in this field.

This document contains design and implementation of an aperiodic position controller for aservomotor. It begins with a reviw state of the art of current aperiodic controls or called .Eventbased Control.and it is classified mainly in: Event-Triggered control (control triggered by an event) and Self-Triggered control (self-triggered control).

Then identification of a DC motor is carried out. Position controller acts on it. Matlab ident tool is used to obtain the open-loop mathematical model system.

It uses self-triggered aperiodic control (CAAD) to implementaton . It is divided into two stages: offline which is where the polynomials and gains values of the controller they are calculate using LQR control in matlab and online one where the previously calculated polynomials and gains are written in the microcontroller and aperiodic sampling theory which calculates the period of the next sampling based on the stability of the system. If the system is stable, the period increases until a maximum preset value and if it is unstable or a reference change it decreases until it reaches a minimum value also preset in the design.

Finally, it is demonstrated that control theory can be implemented in a servomechanism since the system stabilizes while the sampling period varies and it is recommended to do physical performance tests in future work.

Índice general

Índice de figuras

Índice de tablas

Lista de Programas

Introducción

Motivación

Aunque el control aperiódico es un tema que tienes algunos años de investigación, anteriormente era imposible implementarlo por los requerimientos en procesadores y cálculos matemáticos complejos de realizar. El avance tecnológico en software numéricos permite ahora nuevas alternativas para que la investigación en este campo avance y encontramos avances realizado en los siguientes trabajos:

En [1] se presenta una comparación entre técnicas de control periódico y aperiódico, se concluye que el siguiente paso importante es validar estas técnicas en aplicaciones prácticas y encontrar nuevas preguntas y teorías de investigación.

En [2] se realiza una guía para implementar controladores auto-disparados y se presenta al control no periódico como una alternativa de eficiencia en el consumo de recursos.

En [3] se realiza la implementación de un control auto disparado en un sistema de primer orden en donde los resultados experimentales se alinean con los de la teoría .

Utilizando estos avances en control aperiódico, se probará si la metodología planteada es implementable en un sistema de segundo orden.

Objetivos

Objetivo Principal

- Desarrollar un controlador de posición basado en técnicas de control aperiódico para un servomotor.

Objetivos Específicos

- Investigar el estado del arte de técnicas de control aperiódico para seleccionar una y aplicarla en el control de posición de un servomotor.
- Identificar la función de transferencia del mecanismo a controlar utilizando software numérico.
- Diseñar el controlador de posición aperiódico
- Implementar el controlador de posición sobre un microcontrolador, con su respectivo análisis de resultados.

Antecedentes

En [1] se afirma que, los investigadores retoman el interés en el control basado en eventos por el consumo de recursos del control periódico en sistemas de control cableados oh inalámbricos es alto. En la ultima década se presentan avances importantes en control aperiódico. Por ejemplo: En [3] se realiza la implementación de un control auto disparado en un sistema de primer orden en donde los resultados experimentales se alinean con los de la teoría. Y en [2] se realiza una guía para implementar controladores auto-disparados y se presenta al control no periódico como una alternativa de eficiencia en el consumo de recursos.

Tambien en [1] se concluye que el siguiente paso importante es validar estas técnicas en aplicaciones prácticas y encontrar nuevas preguntas y teorías de investigación.

Problema

Los servomecanismos tales como servomotores son muy usados en ingeniería mecatrónica por tener la habilidad de localizar una posición en un rango de operación y permanecer estables en ella. Un servomotor es un dispositivo electromecánico usado para controlar el movimiento y posición exacta de una carga. Una referencia de posición es comparada con la posición actual del servomotor para ser corregida automáticamente[4]. Actualmente los servomotores son escogidos para desarrollar sistemas mecatrónicos compactos y no costosos [5]. Por ejemplo, los robots y también las máquinas de control numérico utilizan servomotores para mover rápidamente y con precisión un efector final (herramienta) hacia una posición deseada. [4]

Para controlar la posición de motores DC es dominante el uso del controlador PID convencional por ser sencillo y fácil de implementar [5]. Esta técnica de control se basa en modelar matemáticamente la dinámica de la planta y diseñar un controlador que garantice el funcionamiento deseado, realizando muestreos periódicos en los sensores de retroalimentación de los estados de la planta [6].

Los libros de control como por ejemplo [6]-[7] presentan únicamente técnicas de control periódicas como opción de implementación en plataformas digitales. Sin embargo, se han implementado técnicas de control aperiódico (preferiblemente llamadas control basado en eventos y control auto disparado) en las computadoras [1] . La literatura existente presenta como ventaja principal que al calcular un nuevo muestreo por cada evento de control se reduce el consumo de recursos de red, tiempo de procesamiento y batería [3].
Para validar este planteamiento se verifica si las soluciones propuestas en [2] son implementables en un sistema de segundo orden. Así, en trabajos futuros utilizando técnicas de control aperiódico para controlar la posición de un servomotor de bajo torque se puede comparar el consumo de recursos con un control periódico, garantizando el rendimiento adecuado en ambos casos.

Justificación

Par garantizar que un control de lazo cerrado se actualice eficientemente en un período, se necesita rapidez en los microprocesadores, redes con velocidades altas. El control no periódico determina un muestreo y procesamiento óptimo como solución a esta necesidad [2].

Las técnicas de control por eventos y control auto disparado disminuyen las acciones de control cambiando el tiempo de muestreo al máximo si la planta está inestable y al mínimo si la planta está estable logrando así disminuir el consumo de recursos computacionales, de red, de batería [3].

Alcance

Se implementará un controlador de posición basado en técnicas de control aperiódico, para un servomotor.

El servomotor constará de un motor DC, un codificador de cuadratura y una interfaz de potencia. El controlador se implementará sobre un microcontrolador.

Se hará la identificación del mecanismo a controlar con software numérico, así como la simulación del nuevo controlador.

Capítulo 1

Revisión literaria

1.1. Control por eventos

En los sistemas de control que se implementa actualmente en sistemas embebidos se calcula la acción de control y la supervisión de los estados del sistema para un periodo constante T como se muestra en la figura 1.1 haciendo que las redes de comunicación (inalámbricas o cableadas) estén ocupadas en todo momento y consumiendo recursos. Sin embargo, existen teorías de control aperiódicas o basadas en eventos que proponen una mejora en el rendimiento de recursos al variar el periodo T de muestreo según la atención que el sistema necesite.

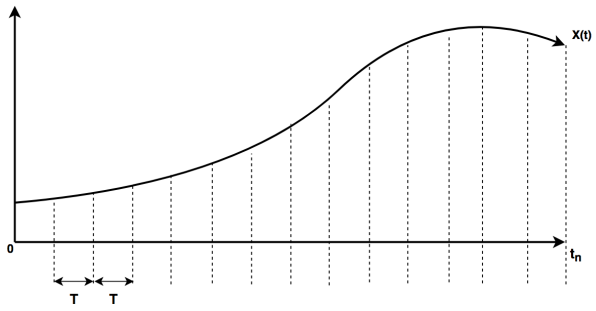


Figura 1.1: Periodo T de muestreo constante

A continuación se presenta un resumen de [1] que es la última actualización del estado del arte en debates de control periódico vs control aperiódico (control por eventos). Donde se habla de control disparado por eventos (event- triggered control) y control auto disparado (self-triggered control).

El control aperiódico se enfoca en el estudio de dos elementos indispensables: primero calcular la acción de control que requiere el sistema para estabilizar los estados y segundo implementar el mecanismo que varía el periodo de muestro en la que la acción de control se vuelve a calcular. En base a este mecanismo podemos diferenciar que el control disparado por eventos es reactivo ya que realiza la supervisión de una condición de disparo constantemente y cuando esta condición no se cumple se dispara un evento para controlar el sistema. Mientras el control auto disparado es proactivo debido que calcula el tiempo en que se va a disparar la siguiente acción de control basándose en la dinámica de la planta y los estados actuales.

1.2. Control disparado por eventos

Entonces la primera parte es calcular la acción de control como se vio en 1.1 para lo cual se considera una planta lineal 1.1

$$\frac{d}{dt}x_p = A_p x_p + B_p u, \tag{1.1}$$

donde $x_p \in \mathbb{R}^{n_p}, u \in \mathbb{R}^{n_u}$ son los estados y la señal de entrada. $A_p \in \mathbb{R}^{nxn}, B_p \in \mathbb{R}^{nxm}$ son las matrices que describen la dinámica del sistema.

Para la cual la ley de control se define como

$$u = Kx_p, \tag{1.2}$$

y la parte real de los valores propios de $A_p + B_p K$ son negativos entonces la planta es controlable .

Y la segunda parte es implementar un mecanismo para variar el periodo que actué cuando el rendimiento de la planta no es adecuado y así garantizar la estabilidad del sistema. Este el mecanismo se diseña para disparar la acción de control cuando se viole un desigualdad que se define en la ecuación 1.3 y es la función de Lyapunov denotada por V

$$\frac{d}{dt}V(x_p(t)) \leq -\sigma x_p^T Q x_p, \tag{1.3}$$

donde Q es positiva definida, y los valores de $-\sigma \in \left[0, 1\right]$. Considerando que $\sigma = 1$ calcula un rendimiento periódico y valores $\sigma < 1$ dan una velocidad de cambio más lenta para V .

En el control disparado por eventos los tiempos de muestreo τ_k se contabilizan cuando la acción se vuelve a calcular y se envían al actuador, este valor permanecen constantes hasta la próxima actualización. La ecuación 1.3 se puede reescribir considerando el error $e(t) = x_p(t_k) - x_p(t)$y se obtiene(1.4)

$$z^T(t_k)\psi z(t_k) = 0, \tag{1.4}$$

con

$$\psi = \begin{bmatrix} (\sigma - 1)\, PB_pK \\ K^T B_p^T \quad 0 \end{bmatrix}, z(t_k) = \begin{bmatrix} x_p(t_k) \\ e(t_k) \end{bmatrix}. \tag{1.5}$$

Sin embargo, el control disparado por eventos requiere de la supervisión constante de la desigualdad y de los estados de la planta y se usa cuando se diseña sistemas con hardware destinado para este propósito.

1.3. Control aperiódico auto disparado CAAD

Esta sección es un resumen de [8] donde se detalla la teoría de control auto disparado y se presenta un proceso para implementar que consta de una parte de diseño previo.

1.3.1. Dinámica en tiempo continuo representado en espacio de estados

Para el diseño se considere un sistema lineal invariante en el tiempo (LTI) descrito en 1.6

$$\begin{cases} \dot{x_{(t)}} = A_c x_{(t)} + B_c u_{(t)} \\ y_{(t)} = C x_{(t)} \end{cases}, given\, x_{(0)} = x_{(0)} \tag{1.6}$$

donde$x_{(t)} \in \mathbb{R}^{nxn}$ es el estado y $u_{(t)} \in \mathbb{R}^m$ señal de entrada de control. $A_c \in \mathbb{R}^{nxn}$ y $B_c \in \mathbb{R}^{nxm}$ describen la dinámica del sistema, y $C \in \mathbb{R}^{qxm}$ es la matriz de peso usada para leer los estado.Las variables m,n y q denotan las dimensiones de las matrices de estados, entradas y salidas respectivamente.

1.3.2. Dinámica en tiempo discreto representado en espacio de estados

La dinámica de tiempo continuo en (1.6) es discretizado utilizando métodos tomados de [9],

$$A_d = e^{Ac\tau},\ B_d = \int_o^{\tau} e^{Ac(\tau - t)}\, dt B_c, \tag{1.7}$$

entonces el sistema discreto se representa en 1.8

$$\begin{cases} \dot{x_{(k-1)}} = A_d x_{(k)} + B_d u_{(k)} \\ y_{(k)} = C x_{(k)} \end{cases}, given\, x_{(0)} = x_{(0)} \tag{1.8}$$

Los polos en tiempo continuo P_C se convierten en polos en tiempo discreto P_d a través de 1.9

$$P_d = e^{p_{c*\tau}} \tag{1.9}$$

, para profundizar en esto se puede revisar [9] .

1.3.3. Control regulador cuadrático lineal LQR

Para calcular la acción de control que es la primera parte a considerarse para el diseño de CAAD se usa el método de control óptimo LQR que permite encontrar una señal de entrada óptima en tiempo continuo y tiempo discreto que minimiza la función de costos en (1.10) y (1.11) respectivamente a través de los valores de la matriz K.

$$J_c = \int_0^{\infty} \left(x_{(t)}^T Q_c x_{(t)} + 2 x_{(t)}^T S_c u_{(t)} + u_{(t)}^T R_c u_{(t)} \right) dt, \tag{1.10}$$

$$J_d = \sum_0^{\infty} \left(x_{(k)}^T Q_d x_{(k)} + 2 x_{(k)}^T S_d u_{(k)} + u_{(k)}^T R_d u_{(k)} \right). \tag{1.11}$$

en donde $Q_c, Q_d \succeq 0 \in \mathbb{R}^{nxn}$ son positivos semi-definidos, $R_c, R_d \succ 0 \in \mathbb{R}^{nxn}$ son definitivos positivos, y

$$S_c, S_d \succ 0 \in \mathbb{R}^{nxm}$$

1.3.4. Teoría de muestreo para CAAD

Como segunda parte de la teoría para diseñar un CAAD, en esta sección se establece el método que se utiliza para variar el periodo y calcular cuando se ejecutará la acción de control nuevamente. La regla de muestreo es:

$$\tau_k = \tau_{max} \frac{1}{\frac{\tau_{max}}{\eta} \left| K_c \left(A_c + B_c K_c \right) x_{(k)} \right|^{\alpha+1}} \tag{1.12}$$

donde se define el límite máximo para los intervalos de muestreo por t_{max}; de manera similar se selecciona η que modifica la densidad en la secuencia de muestreo (η más pequeña produce instantes de muestreo más densos y viceversa).También, de acuerdo con [10] y [3] existen configuraciones óptimas para el exponente $\alpha \geqslant 0$ que influye en la densidad de las muestras establecidas; con $\alpha = 0$ el muestreo se vuelve periódico.

Además, a partir de [3] la señal de control expresado en forma de retroalimentación de estado lineal se expresa como

$$u_{(k)} = -k_{d(\tau_k)} x_{(k)}, \tag{1.13}$$

donde $K_{d(t_k)}$ se debe calcular en cada ejecución del controlador. Este valor se obtiene al resolver el problema de LQR en tiempo discreto con (1.11) considerando un período de muestreo fijo T_k sin embargo esto no es posible por el costo computacional que se necesita para resolver el control LQR.Se presenta mas adelante una método para resolver esto.

1.3.5. Consideraciones para la implementación

1.3.6. Problemas considerados

El primer problema tiene que ver con el cálculo de la matriz de ganancia del controlador $K_{d(t_k)}$ para (1.13) ya que,el procesos es computacionalmente costosos y se deben realizar en cada ejecución del controlador por lo tanto se vuelve difícil de implementar en tiempo real en un sistema integrado con baja capacidad computacional. Para resolver los problemas anteriores, se propone calcular aproximaciones utilizando software numérico que generen una ecuación

aproximada y luego implementar la aproximación en el sistema microprocesado con el fin de optimizar la utilización del procesador.

1.3.7. Precálculo de los tiempos de muestreo para calcular la matriz de ganancias $K_{d(t_k)}$

La estrategia propuesta en [8] incluye el cálculo de la matriz de ganancias $K_{d(t_k)}$ del controlador , y generar una función polinómica para imitar su comportamiento y evitar el costo de resolver el control LQR en cada tiempo de muestreo. El primer paso se basa en describir un conjunto de intervalos de muestreo $T \in \mathbb{R}^{1xs}$ dentro de un intervalo cerrado $[\tau_{min}, \tau_{max}]$ de la siguiente manera

$$T = \tau_{min,}\tau_{min,} + \tau_{g,}\tau_{min} + 2\tau_{g,\ldots}, \tau_{max} \tag{1.14}$$

donde τ_g es la granularidad del muestreo definida como la menor unidad de incremento para los intervalos de muestreo.Los tiempos de muestreo mínimo , τ_{min} y máximoτ_{max}, así como τ_g se eligen siguiendo las condiciones detalladas en [3]//

Para garantizar una correcta selección de los tiempos se debe respetar la expresión (1.15)

$$\begin{cases} \frac{\beta^{\alpha}}{\eta} \leq \frac{1}{\tau_{min}} - \frac{1}{\tau_{max}}, \ \beta := sup\,|K_c(A_c + B_cK_c)x|\, x \in X \\ \tau \leq \tau_g \end{cases} \tag{1.15}$$

1.3.8. Estrategia para calcular la matriz de ganancias del controlador

$K_{d(\tau_h)}$ se calcula para cada h^{th} elemento del conjunto de intervalos de muestreo T en (1.14) que llamaremos tau_h. Lo anterior implica que cada vez se debe calcular las matrices discretas $A_{d(\tau_h)}$, $B_{d(\tau_h)}$ por (1.7) con las cuales el problema de LQR de tiempo discreto se resuelve; este último estipula la minimización de la función de costo en (1.11). Por lo tanto, obtenemos un total de matrices de ganancia de controlador del tipo $K_{d(\tau_h)} \in \mathbb{R}^{mxn}$ porque se evalúan para cada uno de los posibles valores de τ_h dentro de el conjunto T. Estas matrices tienen la forma

$$K_{d(\tau_h)} = dlqr(A_{d(\tau_h)}, B_{d(\tau_h)}, Q_{d(\tau_h)}, R_{d(\tau_h)}) = \begin{bmatrix} kd_{11}^{(\tau_h)} & \dots & kd_{1n}^{(\tau_h)} \\ . & . & . \\ . & . & . \\ . & . & . \\ kd_{m1}^{(\tau_h)} & \dots & kd_{mn}^{(\tau_h)} \end{bmatrix} \tag{1.16}$$

donde el super índice (τ_h) indica la pertenencia del elemento de ganancia correspondiente a la matriz $K_{d(\tau_h)}$.Si los elementos de todas las matrices de ganancia se reagrupan de acuerdo con su posición, tenemos un grupo S_{K_d},donde n y m son las dimensiones de los estados y las entradas respectivamente,

$$S_{K_d} = \left\{ \left[kd_{11}^{(\tau_1)}, \dots, kd_{11}^{(\tau_s)} \right], \dots, \left[kd_{1n}^{(\tau_1)}, \dots, kd_{1n}^{(\tau_s)} \right], \dots, \left[kd_{m1}^{(\tau_1)}, \dots, kd_{m1}^{(\tau_s)} \right], \dots, \left[kd_{mn}^{(\tau_1)}, \dots, kd_{mn}^{(\tau_s)} \right] \right\}. \tag{1.17}$$

luego cada conjunto de entrenamiento en (1.17) se define en $\mathbb{R}^{1xs}$ y se usa para realizar un ajuste de curva polinomial para encontrar los coeficientes θ de polinomios de grado $K_{ij}(\tau_k)$.Por lo tanto, tenemos un total de mn polinomios

$$k_{ij(\tau_k)} = \theta_1^{(ij)} \tau_k^{d-1} + \theta_2^{(ij)} \tau_k^{d-1} + \dots + \theta_d^{(ij)} \tau_k + \theta_{d+1}^{(ij)}, \tag{1.18}$$

donde el super índice (ij) indica la pertenencia de los coeficientes θ al polinomio $K_{ij}(\tau_k)$); i-fila y j- columna muestran la posición de los polinomios en la matriz de ganancia. Tenga en cuenta el cambio de tau_k en lugar de $\tau_h$ya que el primero será el intervalo de muestreo actual calculado por la ecuación (1.12) en un controlador real. Por lo tanto de (1.16) a (1.18) se convierten en

$$K_{d(\tau_k)} = \begin{bmatrix} K_{11}(\tau_k) & \dots & K_{1n}(\tau_k) \\ . & . & . \\ . & . & . \\ . & . & . \\ K_{m1}(\tau_k) & \dots & K_{mn}(\tau_k) \end{bmatrix} \tag{1.19}$$

donde

$$k_{11}(\tau_k) = \theta_1^{11} \tau_k^d + \theta_2^{(11)} \tau_k^{d-1} + \dots + \theta_d^{(11)} \tau_k + \theta_{d+1}^{(11)} \dots \tag{1.20}$$

$$k_{1n}(\tau_k) = \theta_1^{1n}\tau_k^d + \theta_2^{(1n)}\tau_k^{d-1} + ... + \theta_d^{(1n)}\tau_k + \theta_{d+1}^{(1n)}... \tag{1.21}$$

$$k_{m1}(\tau_k) = \theta_1^{m1}\tau_k^d + \theta_2^{(m1)}\tau_k^{d-1} + ... + \theta_d^{(m1)}\tau_k + \theta_{d+1}^{(m1)}... \tag{1.22}$$

$$k_{mn}(\tau_k) = \theta_1^{mn}\tau_k^d + \theta_2^{(mn)}\tau_k^{d-1} + ... + \theta_d^{(mn)}\tau_k + \theta_{d+1}^{(mn)} \tag{1.23}$$

Para el diseño del servomotor con control aperiódico propuesto en este documento se selecciona entonces usar control auto disparado (self-triggered control).

1.4. Identificación del modelo matemático de un motor DC

El método de identificación de esta sección es un resumen de [11]. Es necesario conocer las características dinámicas de la planta las cuales se representan en un modelo matemático, para diseñar un controlador. El modelo matemático se puede deducir usando las leyes de la física lo que puede ser muy complicado para su obtención y comprobación.

1.4.1. Identificación por curva de reacción

Se presenta el método de curva de reacción que también se usa para la identificación de sistemas, consiste en determinar la función de transferencia aproximada de menor orden, a partir de la respuesta de la planta a una entrada escalón o impulso que se introduce en la variable de control $u(t)$ y registrar la respuesta $y(t)$ como se muestra en la figura 1.2. Con el registro se procede a proponer una función de transferencia en la que la respuesta a la misma entrada sea una curva similar a la obtenida en el proceso real.

El fundamento teórico de este análisis se basa en que si tenemos una función de transferencia

$$G(s) = \frac{Y(s)}{U(s)}, \tag{1.24}$$

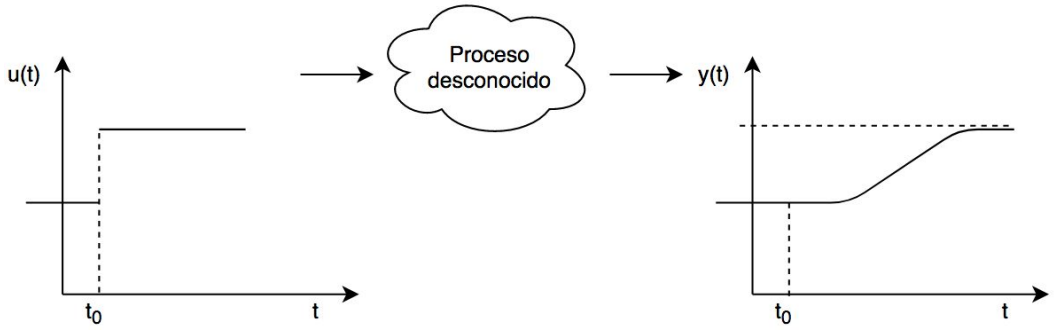


Figura 1.2: Obtención de una curva de reacción

y al conocer la entrada $U(s)$ podemos encontrar $Y(s)$ y por ende $y(t)$ si la entrada al sistema es un escalos $U(s) = \frac{1}{s}$ entonces la salida esta dada por (1.25)

$$y(t) = L^{-1}[\frac{G(s)}{s}] \tag{1.25}$$

La curva de reacción se utiliza mucho en la industria de procesos para caracterizar un sistema, al cual se le va a diseñar un sistema de control, sea de tipo PID o controladores sofisticados como control avanzado [11].

1.4.2. Curva de reacción para un sistema de primer orden

La función de transferencia para un sistema de primer orden (1.26)

$$G(s) = \frac{K}{\alpha s + 1} \tag{1.26}$$

donde K es la ganancia del sistema y α es la constante de tiempo. La respuesta a un escalón se puede ver en la figura 1.3 a partir de la cual podemos encontrar K y α así, la ganancia K se puede calcular a partir del valor final al que tiende la variable de salida

$$K = \frac{\Delta y}{\Delta u} = \frac{y(\infty) - \bar{y}}{\Delta u} \tag{1.27}$$

y la constante de tiempo α se calcula al obtener el valor de 63,2 % del valor final de la variable medida y la constante de tiempo α es el instante en que toma este valor como se ve en la figura 1.3.

$$y(\alpha) = 0{,}632K\Delta u \tag{1.28}$$

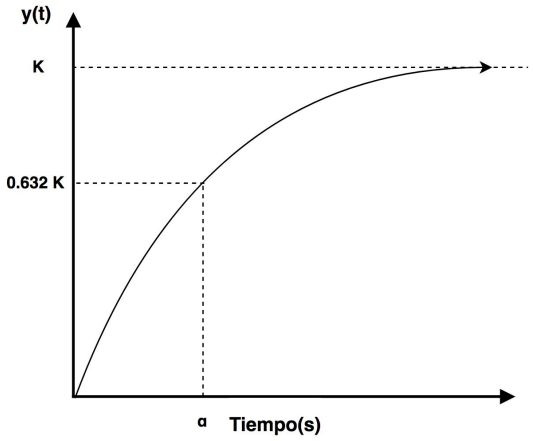


Figura 1.3: Respuesta de un sistema de primer orden a un escalón unitario

1.4.3. Representación de un motor DC en espacio de estados

En [6] y [11] encontramos una aproximación de la función de transferencia de un motor de corriente directa que relaciona el voltaje aplicado $u(t)$ con la posición de $y(t)$

$$G(s) = \frac{K}{s(\tau s + 1)} \tag{1.29}$$

La representación en espacio de estados (1.30) de la función de transferencia 1.29 se formula y simplifica usando la forma canónica controlable presentada en [6]

$$\begin{aligned} \dot{x}(t) &= \begin{bmatrix} \dot{x_1} \\ \dot{x_2} \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ 0 & \frac{-1}{\tau} \end{bmatrix} x(t) + \begin{bmatrix} 0 \\ \frac{K}{\tau} \end{bmatrix} u(t) \\ y(t) &= \begin{bmatrix} 1 & 0 \end{bmatrix} x(t) \end{aligned} \tag{1.30}$$

donde $x_1(t)$ representa la posición de la flecha y $x_2(t)$ su velocidad.

En [6] se muestra que si la planta ha controlar tiene un integrador se denomina servosistema de tipo 1 donde, y si la señal de control u y la señal de salida y son escalares, se puede igualar la salida a uno de los estados. En figura 1.4 se observa la configuración del servosistema tipo 1.

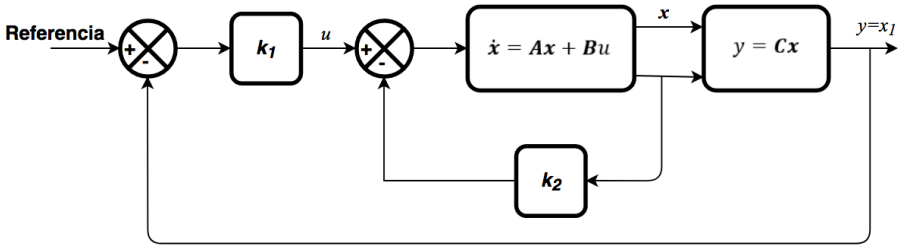


Figura 1.4: Servosistema tipo 1 cuando la planta tiene un integrador

En el gráfico se ve que

$$u = -\begin{bmatrix} 0\, k_2\, k_3 \ldots k_n \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ . \\ . \\ . \\ x_n \end{bmatrix} + k_1(r - x_1)$$

entonces

$$u = -(k_2 x_2 + k_3 x_3) + k^1(Referencia - x_1) = -\mathbf{K}x + k_1 Referencia \tag{1.31}$$

donde

$$K = \begin{bmatrix} k_1\, k_2 \ldots k_n \end{bmatrix}$$

por lo tanto otra representación del diagrama de bloques se muestra en la figura 1.5

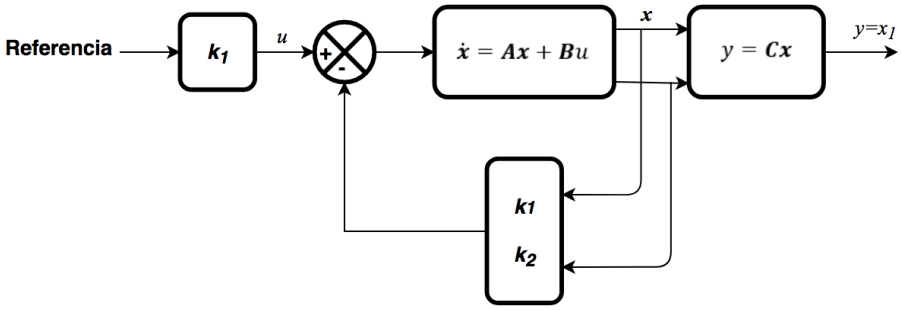


Figura 1.5: Servosistema tipo 1 con matriz de ganancia **K**

Según [12] el control en espacio de estados con realimentación tiene una deficiencia, el que no mejora le respuesta en estado estable por que tiene una ganancia constante lo que lo hace útil solo para sistemas reguladores en donde no se rastrea entradas. Sin embargo, la mayoría de los

sistemas de control deben rastrear entradas.

Como solución se introduce una acción integral junto con la matriz de ganancias para realimentación de estados la figura 1.6 muestra la acción integral en el servosistema de tipo 1.

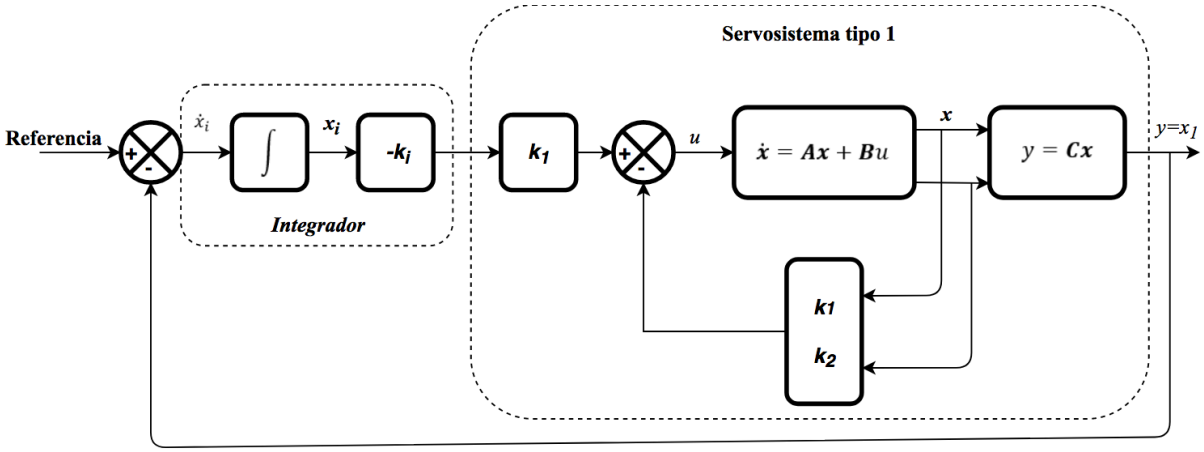


Figura 1.6: Servosistema tipo 1 con integrador

El sistema en espacio de estados 1.32 se extiende con un nuevo estado

$$\begin{aligned} \dot{x}(t) &= Ax(t) + Bu(t) \\ y(t) &= Cx(t) \end{aligned}, \tag{1.32}$$

así

$$\begin{aligned} \dot{\bar{x}}(t) &= \bar{A}\bar{x}(t) + \bar{B}\bar{u}(t) \\ y(t) &= \bar{C}\bar{x}(t) \end{aligned}, \tag{1.33}$$

donde

$$\dot{\bar{x}}(t) = \begin{bmatrix} \dot{x} \\ \dot{x}_i \end{bmatrix}$$

$$\bar{A} = \begin{bmatrix} \mathbf{A} & \mathbf{0} \\ -\mathbf{C} & 0 \end{bmatrix}$$

$$\bar{B} = \begin{bmatrix} \mathbf{B} \\ 0 \end{bmatrix}$$

$$\bar{K} = \begin{bmatrix} \mathbf{K} \, K_i \end{bmatrix}$$

entonces u para el servosistema tipo 1 y la acción integral

$$u(t) = -k_1 x_1 - k_2 x_2 - k_i k_1 x_i \tag{1.34}$$

Capítulo 2

Metodología

2.1. Identificación de la función de transferencia de un servomotor

En este capítulo se procede a identificar la función transferencia del servomotor sobre el cual se implementa el CAAD.

2.1.1. Proceso de identificación

En la figura 2.1 se muestran el proceso para obtener la función de transferencia del sistema sobre el cual se diseñará el controlador.

2.1.2. Configuración de hardware

Hardware utilizado

- Motor DC modelo IG220019X00015R [13]
- Encoder magnético de dos canales
- En la etapa de potencia se usa un DUAL FULL-BRIDGE DRIVER L298 [14]
- Arduino Uno
- Computadora con software numérico(Matlab).

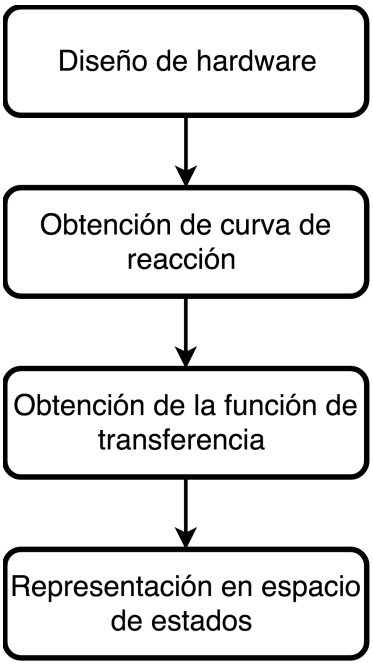


Figura 2.1: Proceso para la identificación de un sistema

En la figura 2.2 se muestran cómo está configurado el hardware del servomotor, en la figura 2.3 y en la figura 2.4 una fotografía real del hardware.

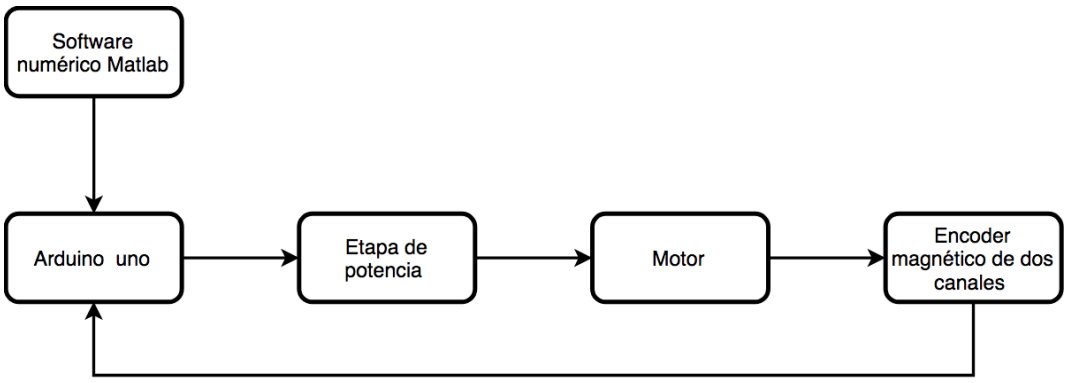


Figura 2.2: Configuración de hardware

2.1.3. Obtención de la curva de reacción

Para obtener la curva de reacción controlamos el voltaje que entra en la etapa de potencia con el conversor análogo – digital que tiene una resolución de 8 bit dando 255 divisiones para controlar 12V, que son $u(t)$ en el sistema y con los puertos digitales se registra los pulsos que

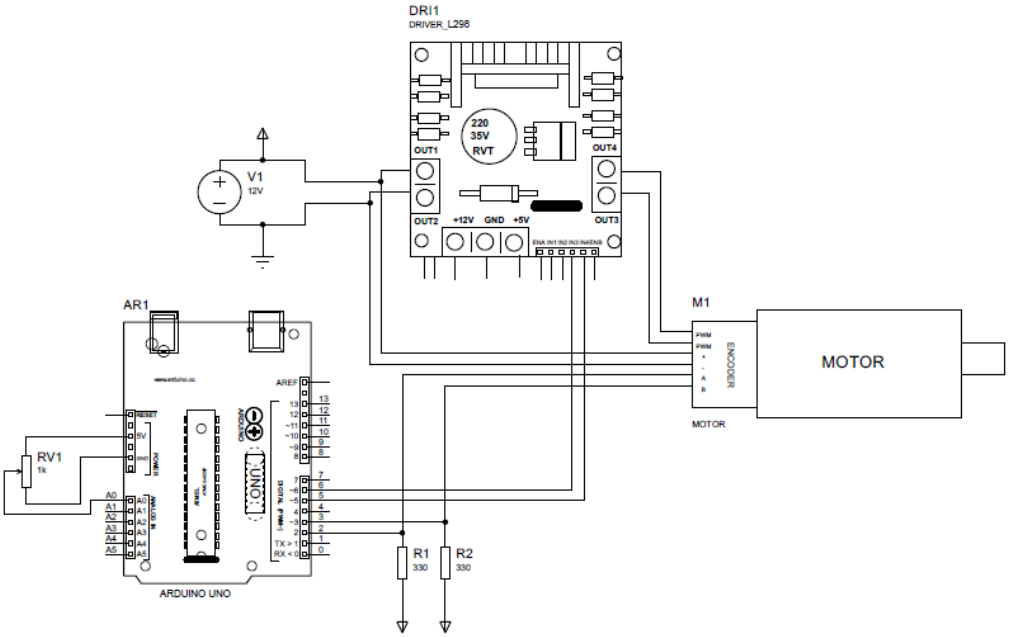


Figura 2.3: Fotografía del hardware

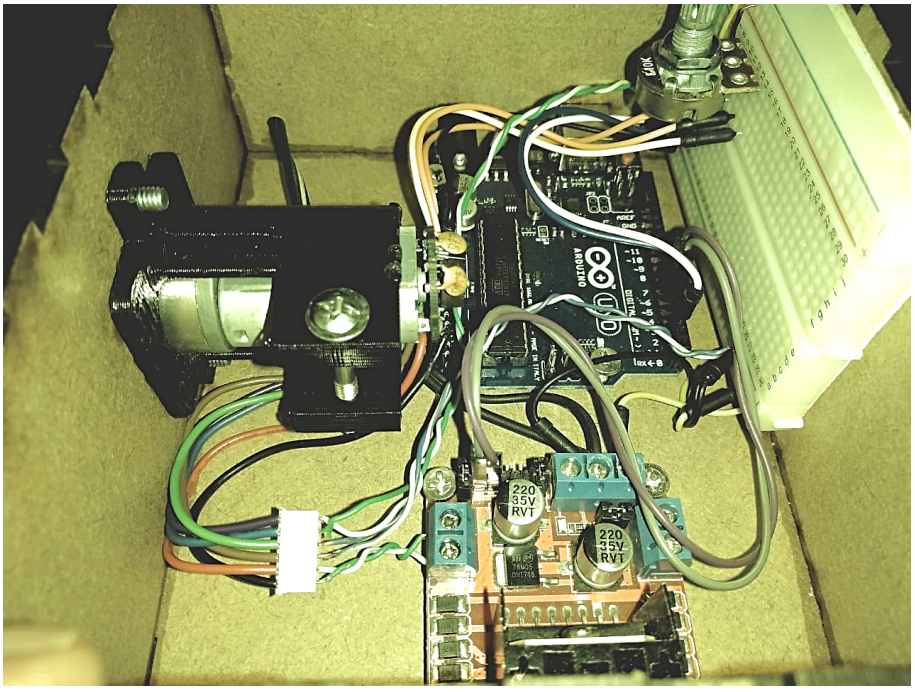


Figura 2.4: Fotografía del hardware

marca el encoder y se calcula la velocidad que es $y(t)$ del sistema en lazo abierto como se muestra en la figura 2.5.

Figura 2.5: Función de transferencia en lazo abierto

El tiempo de muestreo se controla con una interrupción de $100us$. Se registra entonces el valor entre 0 y 255 que da el microcontrolador para variar el voltaje y la velocidad que tiene el motor. Los datos de la tabla 2.1 adquiridos con matlab como se muestra en el apéndice A.1 son ingresados en el ident Matlab para generar y validar la curva de reacción que se muestra en la figura 2.6 .

Tabla 2.1: Datos para identificación del motor

Voltaje	RPM	Muestreo en microsegundos
255	0	97952
255	493,46	98980
255	651,08	97952
255	671,21	98980
255	664,26	98972
255	673,84	98972
255	669,49	98976
255	666,86	97952
255	676,49	98980
255	666,86	98972
255	664,24	97952
255	673,84	98976
255	669,49	98980
255	669,49	97948
255	673,84	98976
255	664,24	98976
255	664,24	97952
255	671,18	98976
255	664,24	98976

2.1.4. Obtención de la función de transferencia

La curva que se muestra en la figura 2.6 es la de un sistema de primer orden como se muestra en la figura 1.3 por lo tanto se procede a calcular K y α para encontrar la función de transferencia del motor $G(s)$ entre la salida que es la velocidad y la entrada que es el voltaje como se ve en la figura 2.7.

Entonces

$$K = \frac{671,2}{255} = 2,63$$

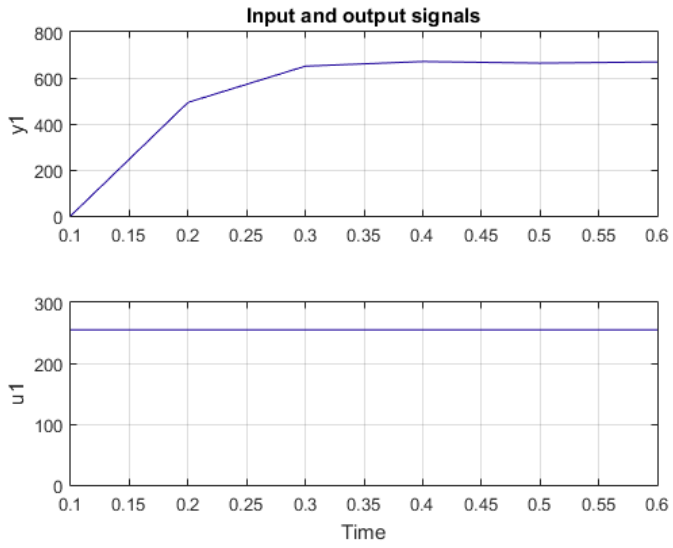


Figura 2.6: Curva de reacción del motor IG220019X00015R

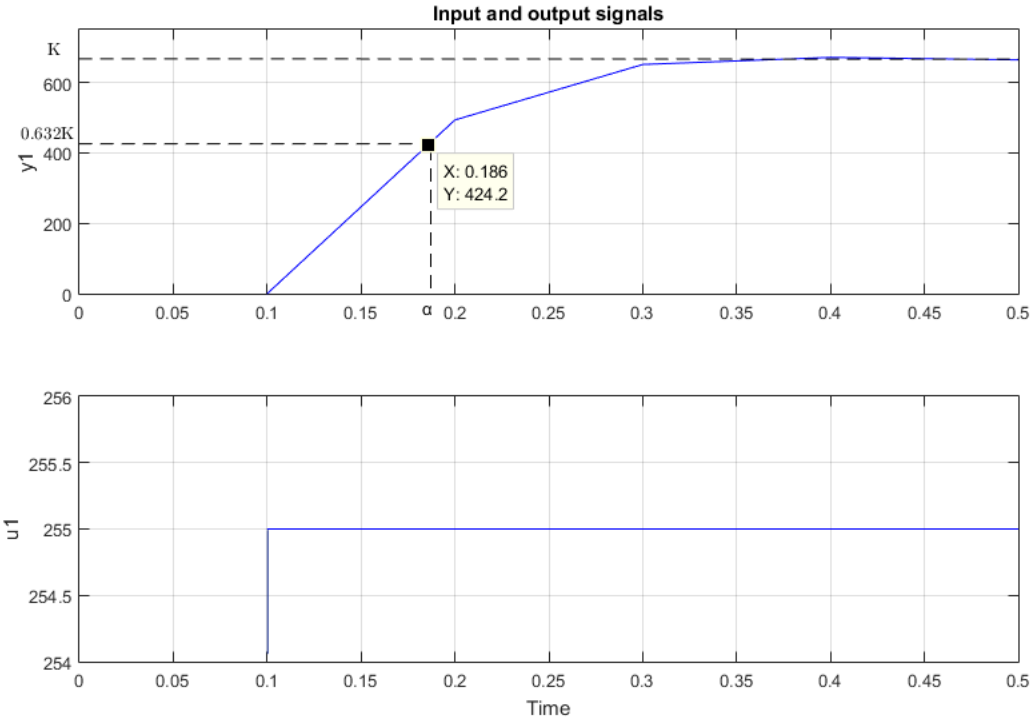


Figura 2.7: Curva de reacción del motor donde se muestra K y α

$$y(\alpha) = 0{,}632 * 2{,}63 * 255 = 424{,}2$$

para encontrar α tomamos del Valor de K y se verifica el tiempo en el que incurrió y se resta el tiempo de dónde empieza la curva

$$\alpha = 0{,}186 - 0{,}1 = 0{,}086$$

finalmente la función de transferencia buscada está dada por:

$$G(s) = \frac{2{,}63}{0{,}086s + 1}$$

Se hace una comparación con la función de transferencia obtenida en el ident de matlab

$$G(s) = \frac{2{,}644}{0{,}0719s + 1}$$

las cuales son semejantes y se utiliza la función obtenida con el ident de matlab.

2.1.5. Representación en espacio de estados

La función de transferencia encontraba está en relación del voltaje y la velocidad. Se requiere diseñar un servomotor y controlar la posición para lo que se procede a integrar la velocidad del sistema como se muestra en la figura 2.8.

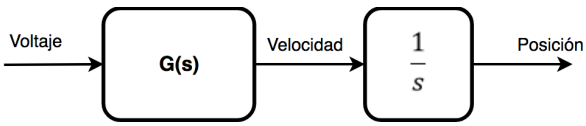


Figura 2.8: Diagramas de bloques de la función de transferencia para controlar la posición del motor

La función de transferencia para controlar la posición es

$$G(s) = \frac{2{,}644}{s(0{,}0719s + 1)}$$

entonces la representación en espacios estados es

$$\begin{bmatrix} \dot{x}_1 \\ \dot{x}_2 \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ 1 & -13{,}9 \end{bmatrix} x(t) + \begin{bmatrix} 0 \\ 36{,}77 \end{bmatrix} u(t)$$

$$y(t) = \begin{bmatrix} 1 & 0 \end{bmatrix} x(t)$$

2.1.6. Representación en espacio de estados con acción integral

Como se determinó en la ecuación (1.33) se debe encontrar las matrices extendidas

$$\bar{A} = \begin{bmatrix} 0 & 1 & 0 \\ 0 & -13{,}9 & 0 \\ -1 & 0 & 1 \end{bmatrix}$$

$$\bar{B} = \begin{bmatrix} 0 \\ 36{,}77 \\ 0 \end{bmatrix}$$

Estas matrices se utilizan en el diseño del controlador.

2.2. Diseño del control CAAD para un servomotor

Antes del diseño del control CAAD para un servo motor se diseña primero un control LQR periódico que se puede ver en el apéndice A.2 y en base a este control se procede a transportarlo a CAAD que se puede ver en el apéndice A.4.

2.2.1. Cálculo de la matriz de ganancia

Según la sección 1.3.3 hay que definir la matriz de pesos Q y R para resolver la función que minimice el costo, entonces con

$$Q = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 0{,}1 & 0 \\ 0 & 0 & 1 \end{bmatrix} y\, R = \begin{bmatrix} 1 \end{bmatrix},$$

se determina la matiz de ganancia K para el sistema de matrices extendidas de 3 estados.Se usa LQR en matlab obteniendo

$$k = \begin{bmatrix} 1{,}2781 & 0{,}0936 & -1{,}7253 \end{bmatrix}$$

.

2.2.1.1. Diagrama de Bode en lazo cerrado

El la figura 2.9 se puede ver que la frecuencia máxima para que el sistema sea estable es $3{,}21 rad/s$ y en [6] se recomienda frecuencias que sean de 10 a 20 veces esta frecuencia, para el control periódico se utiliza 15 veces esta frecuencia

$$f = 48{,}15 \frac{rad}{s} = 7{,}663 Hz$$

entonces el período es

$$T = \frac{1}{f} = \frac{1}{7{,}663} = 0{,}13 s$$

La respuesta simulada del sistema controlado con una referencia de 180 grados y con un periodo de $0{,}13 s$ se muestra en 2.10 se puede revisar el código del programa en el apéndice A.2

.

2.2.2. Diseño de CAAD

Para el diseño de un CAAD se necesitan dos partes, una que calcule la acción de control y otra que varié el periodo. Para la acción de control se utiliza el método calculado en la sección anterior y para la otra en la sección 1.3.4 se muestra el mecanismo para variar el periodo para lo cual es necesario definir el periodo máximo T_{max},el periodo mínimo T_{min} y la granularidad T_g. Del diagrama de bode tenemos el periodo máximo

$$T_{max} = \frac{1}{f} = \frac{1}{5{,}109} = 0{,}195 s$$

y el periodo mínimo

$$T_{min} = \frac{1}{f} = \frac{1}{10{,}218} = 0{,}097 s$$

En la sección 1.3.8 se explica como obtener los polinomios que remplazan a la matriz de ganancias en un control periódico. Los polinomios describen la curva que siguen las ganancias

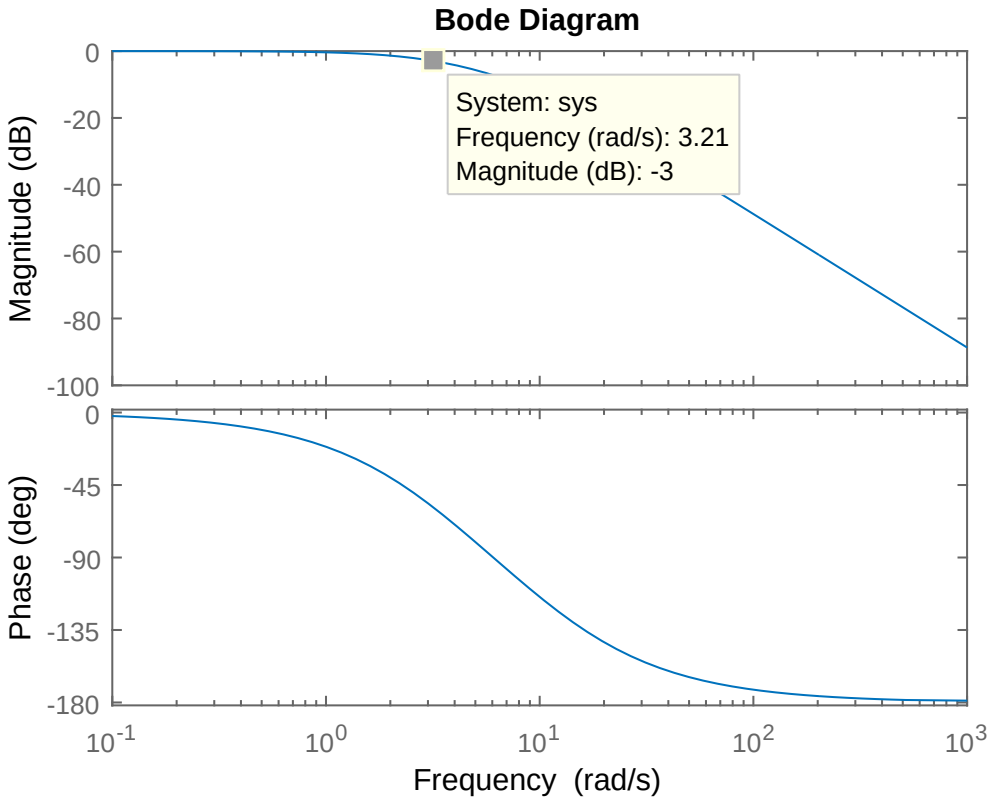


Figura 2.9: Diagrama de bode del servosistema con acción integral en lazo cerrado

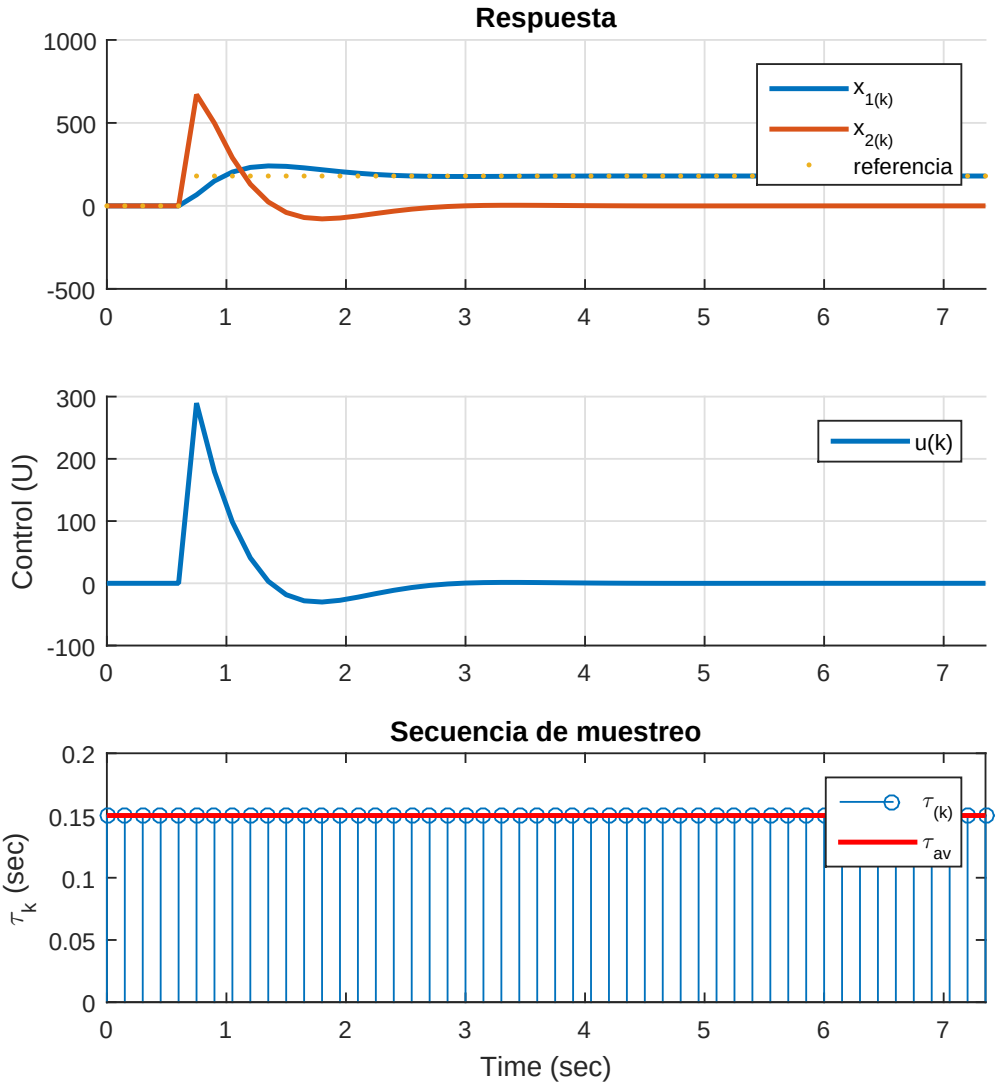


Figura 2.10: Simulación de respuesta al control optimo LQR

obtenidas desde el tiempo mínimo hasta el tiempo máximo pasando una granularidad en cada tiempo como se muestra en la figurar 2.11 y en la figura 2.12 se puede ver las curvas generadas y los polinomios son

$$K11 = (-0{,}9101 * tau_k) + 1{,}2529$$

$$K12 = (-0{,}815 * tau_k) + 0{,}0886$$

$$K13 = (1{,}4564 * tau_k) - 1{,}6864$$

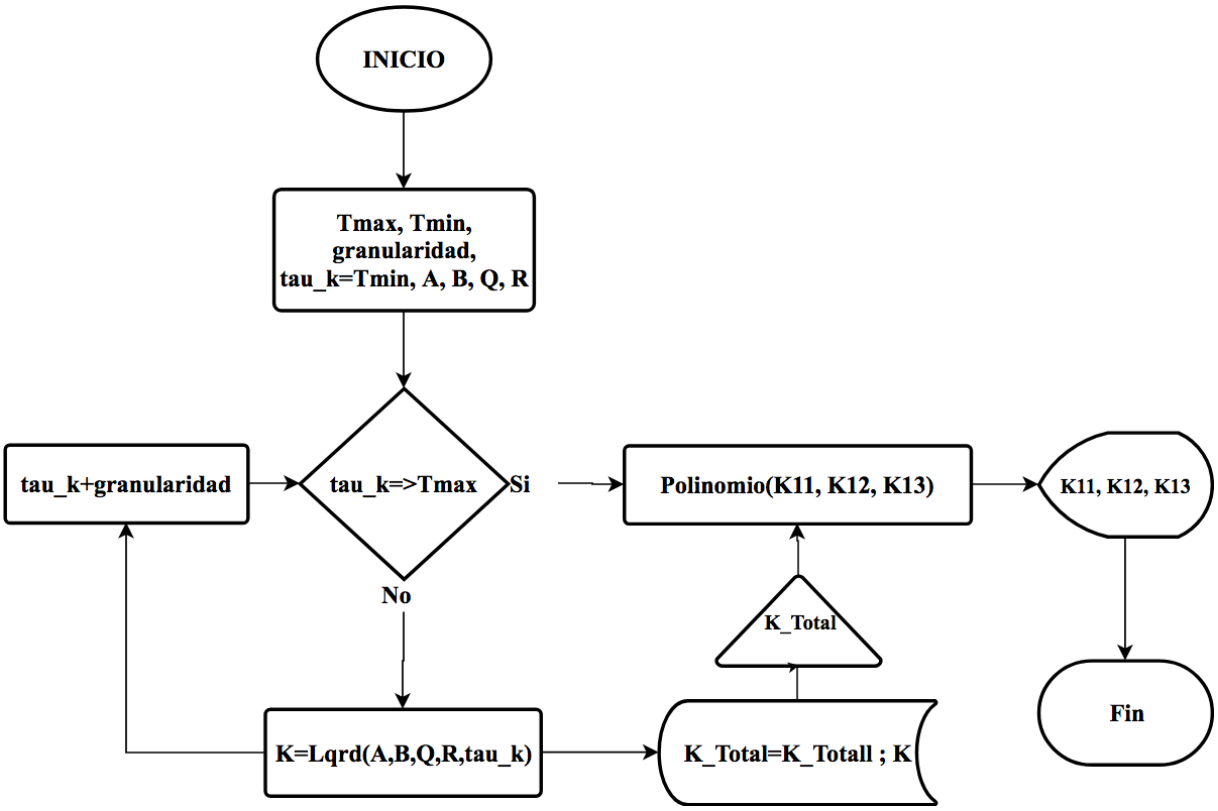


Figura 2.11: Obtención de la ganancia para cada período posible del CAAD

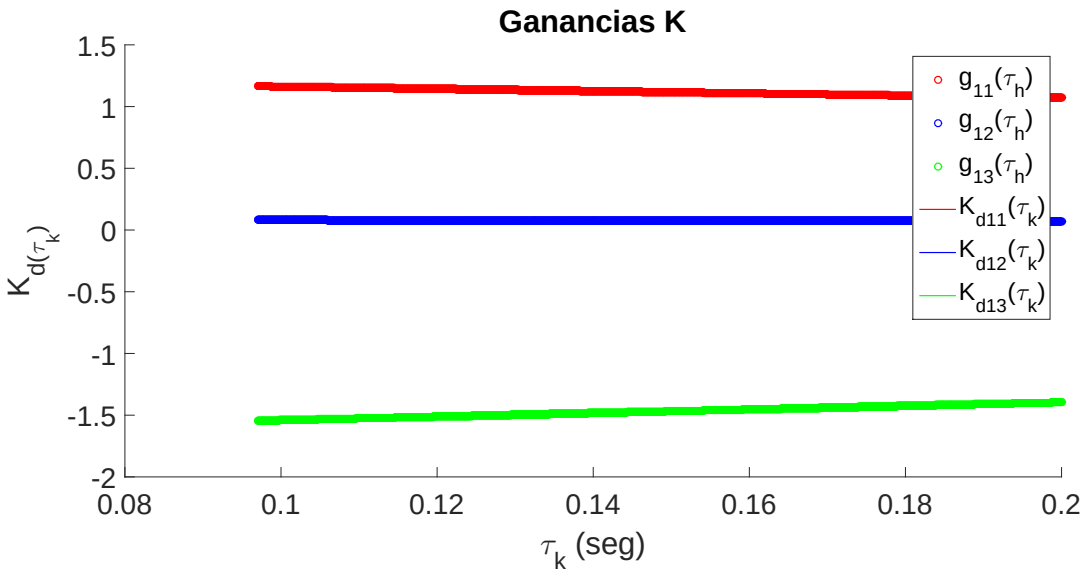


Figura 2.12: Curva de ganancias calculadas con LQR

2.2.3. Variación del periodo

En la sección 1.3.4 se demostró el método para variar el periodo el cual necesita predefinir α y η . En [3] se recomienda que $\alpha = 0{,}67$ y se selecciona un η que garantice que el tiempo mínimo se cumpla $\eta = 10$.

En la figura 2.13 se muestra la simulación de como el sistema responde al CAAD diseñado.

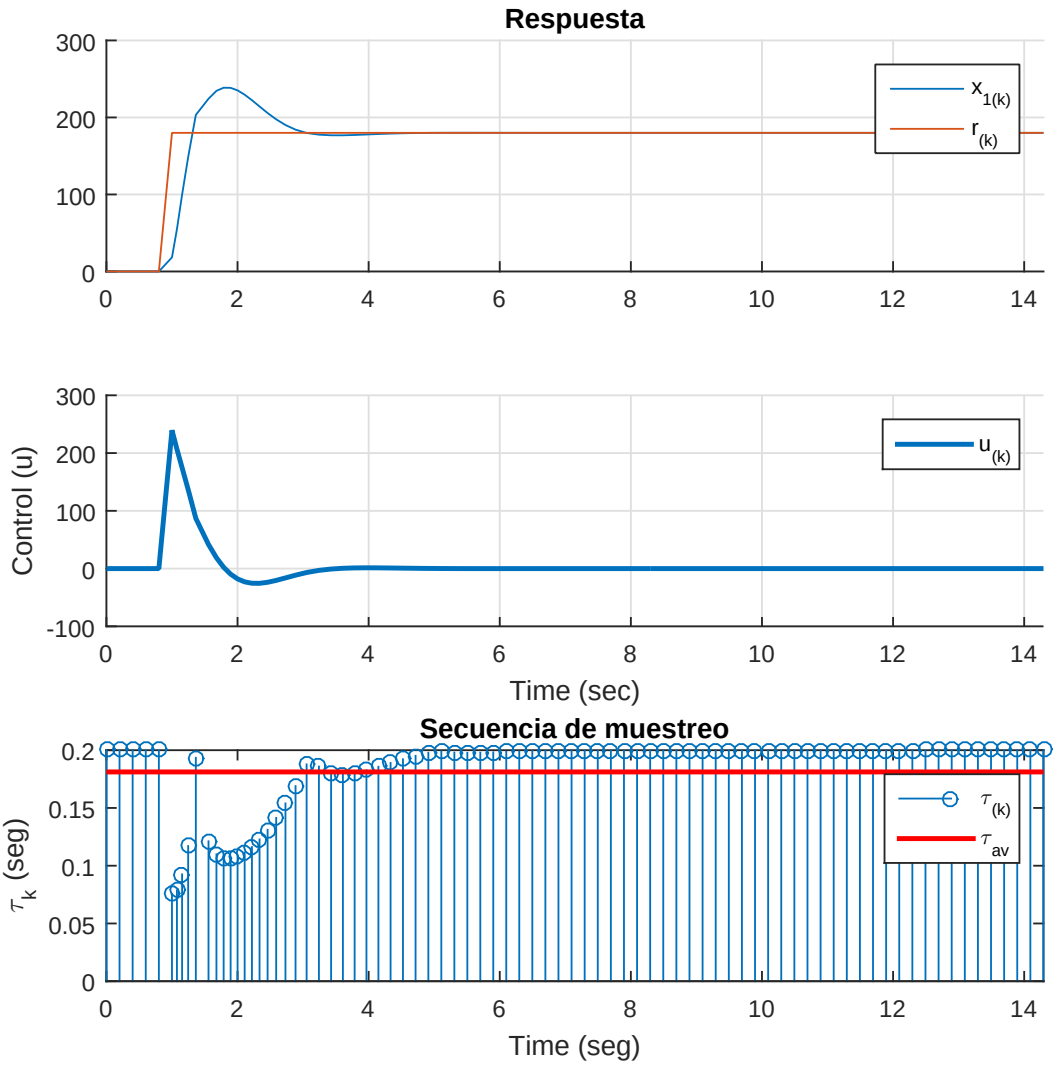


Figura 2.13: Simulación de CAAD

Capítulo 3

Implementación y resultados

La implementación se hace en el hardware definido en la seccion 2.1.2, se implementa el controlador periódico como el CAAD. La figura 3.1 se muestra el diagrama de la configuración del hardware y en la figura una foto real de todo el sistema.

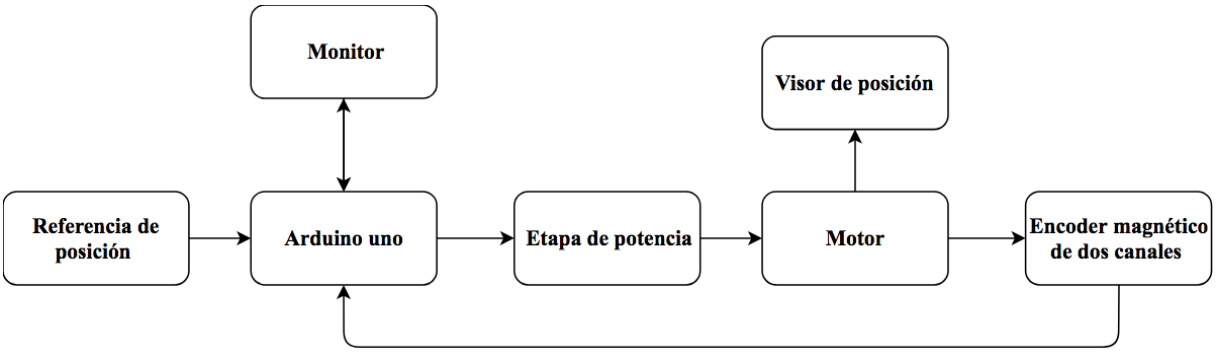


Figura 3.1: Configuración de hardware para control de posición

3.1. Implementación de control periódico

El período de muestreo se ejecuta por interrupción del timer uno del arduino para lo cual se calcula el TCNT(Timer/counter register) que define el tiempo de cada período

$$TCNT = 2^{16} - \frac{0{,}13 * 16000000}{1024} = 63504{,}75 \cong 63505.$$

el estado $x1$ es la posición que se calcula en base a un contador que se actualiza en la lectura del encoder magnético que tiene una resolución de 231 lo que significa que hay un

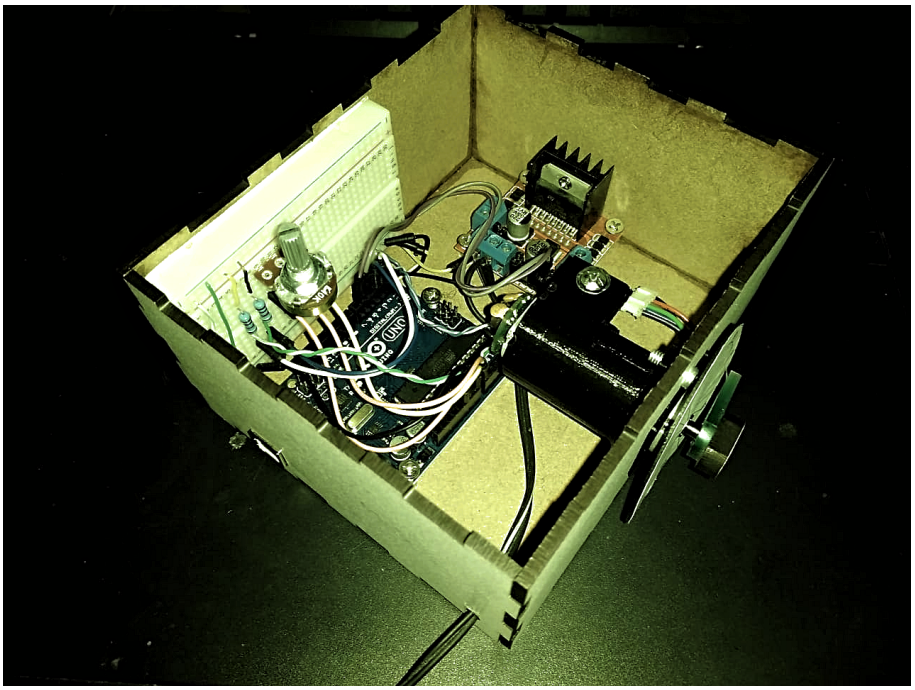


Figura 3.2: Fotografía del servomotor

error en la posición de referencia y la del servomotor de $e = \frac{360}{231} = 1{,}558\,grados$ entonces la posición sera calculada cada 0.13 segundos $\frac{(count*360)}{231}$ y el estado $x2$ es la velocidad que se encuentra derivando la posición respecto al tiempo $\frac{(P_{actual}-P_{pasada})}{tiempo}$ y finalmente el estado $x3$ es la acción integral que acumula el error $error = error + ((Referencia - x1) * perodo)$. La acción de control u se calcula con la ecuación (1.34) siendo la matriz K calculada en el diseño $k = \begin{bmatrix} 1{,}2781 & 0{,}0936 & -1{,}7253 \end{bmatrix}$, en la figura 3.3 se muestra le respuesta real del servomotor a la acción de control.

3.2. Implementación de CAAD

En la tabla 3.1 se muestran los parámetros que se usan el la implementación del CAAD. Los estados $x1, x2, x3$ y u se calculan como en el control periódico y el período como se muestra en la sección 1.12.

Los valores de tiempo están predefinidos ,sin embargo, hay que validar si estos valores se pueden generar en el microcontrolador. Si usamos el timer 1 que tiene una resolución de 16 bits

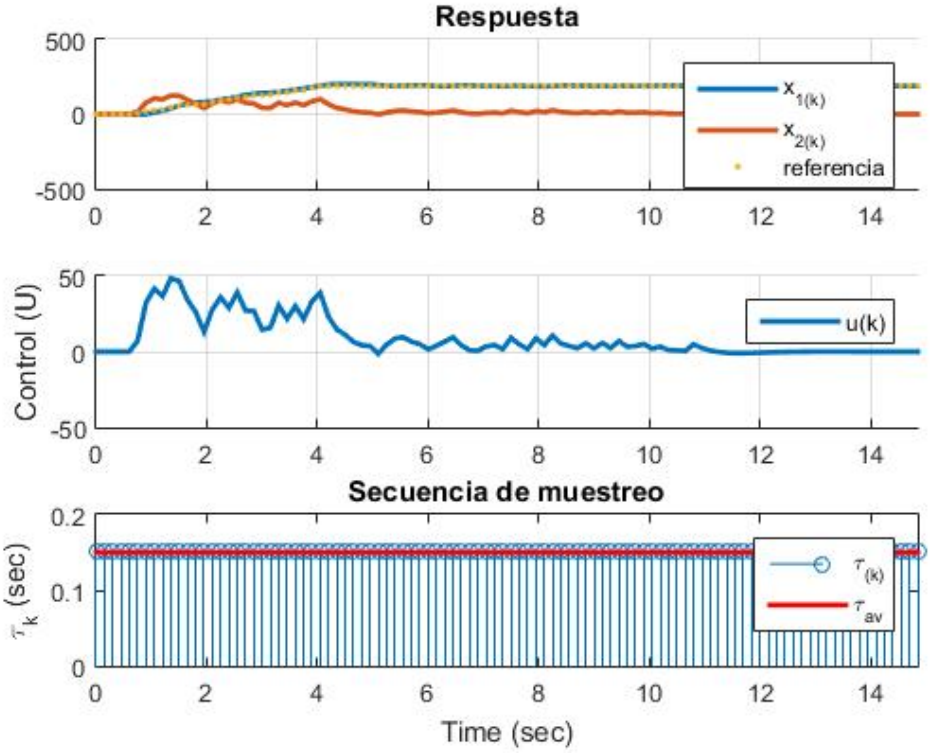


Figura 3.3: Respuesta real del servomotor a la acción de control

Tabla 3.1: Definición de valores usados en el CAAD

Tiempo máximo	0,19
Tiempo mínimo	0,04
α	0,67
η	10
A11	0
A12	1
A21	-13,9
B11	0
B12	36,77
k11	(-0,91 * período)+1,25
k12	(-0,081 * período)+0,088
Ki=K13	(1,456 * período)-1,686

y trabaja a una frecuencia de 16000000 Hz entonces se calcula el TCNT que define el tiempo . Para calcular el TCNT del timer

$$TCNT = 2^{16} - \frac{T * 16000000}{1024}$$

y para calcular el periodo

$$T = \frac{1}{16000000} * (2^{16} - TCNT) * 1024$$

entonces $TCNT_{T_{max}} = 62489,125$ y $TCNT_{T_{min}} = 64020,375$ de donde se tiene los tiempos reales que se pueden implementar $T_{max} = 0,195008$ y $T_{min} = 0,097024$y la granularidad mínima es $T_g = 0{,}000064$.

En la figura 3.4 , la figura 3.5 y la figura 3.6 se muestran imágenes obtenidas con el osciloscipio para validar el cambio del periodo en el microcontrolador.

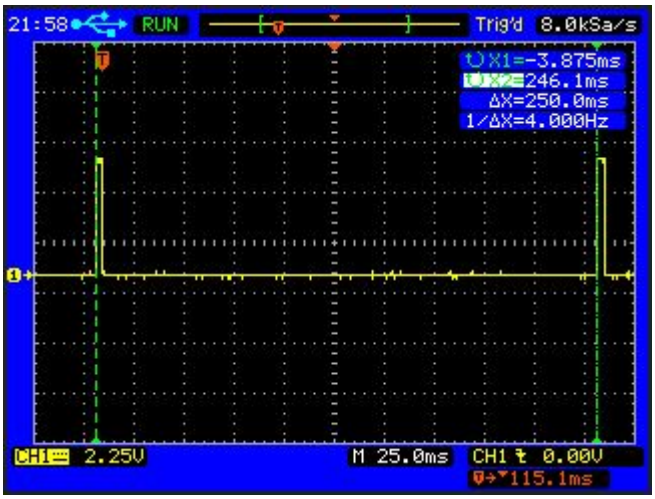


Figura 3.4: Período de muestreo máximo

Finalmente en la figura 3.7 se muestra la respuesta real del servomotor con CAAD.

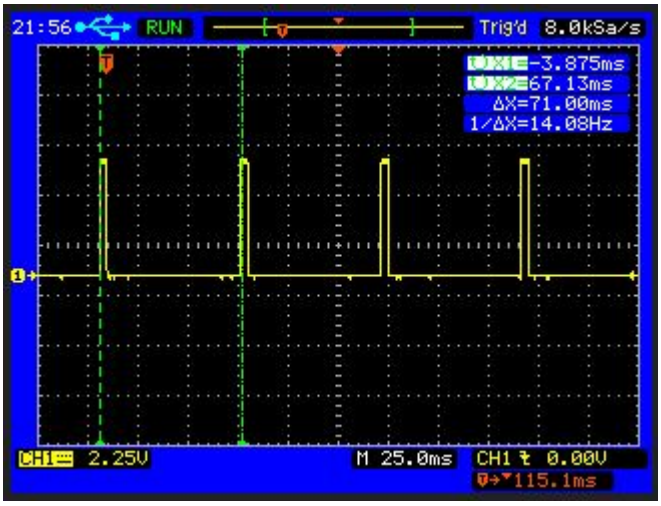


Figura 3.5: Período de muestro intermedio

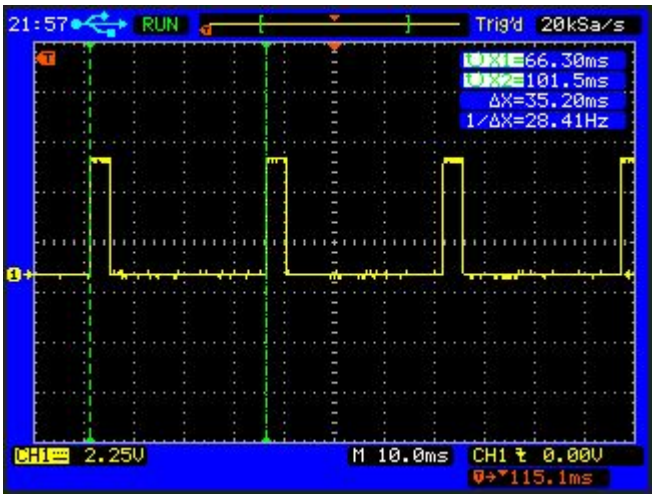


Figura 3.6: Período de muestro mínimo

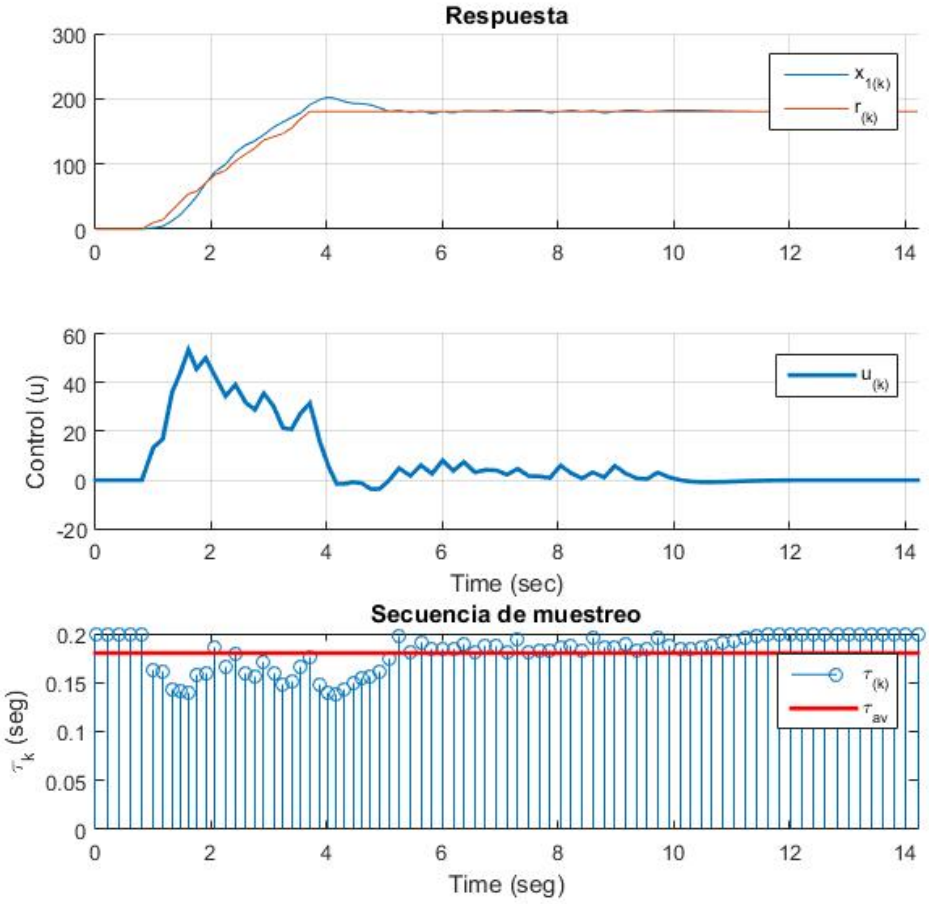


Figura 3.7: Respuesta real del servomotor al CAAD

Conclusiones y trabajos futuros

Conclusiones

En este trabajo de grado se selecciono de los controles basados en eventos al control auto disparado, para implementar por primera vez un control aperiódico en un servosistema en este caso un servomotor.En donde comprobamos que el método es implementable en este sistema.

Se llevó a cabo el diseño de la función de transferencia con el método de curva de reacción y se validó el resultado gráfico con software numérico donde se verificó que los resultados son semejantes. Logrando tener una función de transferencia que se utilizó para implementar el control aperiódico.

En la sección 1.3 se presento la metodología para diseñar un control auto disparado, el mismo que, tanto en la simulación vista en el figura 2.13 y las figuras 3.4 - 3.5 - 3.6 donde se muestra con varia el periodo de muestreo en el sistema, se verifica que cuando la plata es inestable el periodo es el máximo establecido y cuando es inestable disminuye.

Los resultados son positivos en el tema de implementar ya que se consigue que el servomotor se estabilice mientras varia el período de muestreo demostrado en la sección 3.2. Esto demuestra que es posible implementar un CAAD en servo sistemas.

Trabajos futuros

Comparando un control periódico vs el control aperíodico. El control aperíodico al disminuir el período de muestro a un período mínimo cuando el sistema es inestable hace que la

acción de control sea mas repetitiva logrando estabilizar en menor tiempo el sistema. Sin embargo, en el tiempo que realiza esta acción utiliza más recursos que con un control periódico. Y cuando es estable utiliza menos recurso ya que el periodo de muestreo aumenta. Por lo cual se propone que en trabajos futuros se plantee una método para medir el consumo de recurso y así validar tanto y si es que existe o no un ahorro de recursos en sistemas mayormente estables como en sistemas mayormente inestables.

Si bien en este trabajo se demuestra que es posible la implementación de un CAAD en un servomotor aún quedan parámetros por demostrar por ejemplo: que el control aperiódico implementado tienen menor uso de procesamiento, menor consumo energético, menor uso de red para la comunicación tomando en cuenta que al seguir una referencia la estabilidad de un servomotor puede ser constantemente variable, haciendo que el sistema mantenga periodos de muestreo menores que en control periódico.Lo que representaría un costo mayor en recurso; contradiciendo lo propuesto por el control aperiódico. Esto es motivo para probar el CAAD en un servosistema mas complejo por ejemplo un balancín. y finalmente se espera que se realicen pruebas físicas de desempeño del servo mecanismo.

Bibliografía

[1] WPMH Heemels, Karl Henrik Johansson, and Paulo Tabuada. An introduction to event-triggered and self-triggered control. In *Decision and Control (CDC), 2012 IEEE 51st Annual Conference on*, pages 3270–3285. IEEE, 2012.

[2] Implementation guidelines for the optimal-sampling-inspired self-triggered control.

[3] Manel Velasco, Pau Marti, and Enrico Bini. Optimal-sampling-inspired self-triggered control. pages 1–8, 2015.

[4] V. Kumar, P. Gaur, and A. P. Mittal. Trajectory control of dc servo using os-elm based controller. pages 1–5, Dec 2012.

[5] T. Wada, M. Ishikawa, R. Kitayoshi, I. Maruta, and T. Sugie. Practical modeling and system identification of r/c servo motors. pages 1378–1383, July 2009.

[6] Katsuhiko Ogata. *Ingeniería de control moderna*. Pearson Educación, 2003.

[7] *Control de realimentación de sistemas dinámicos*.

[8] Cristina Vaca Carlos Rosero, Juan Benavides. Implementation guidelines for the optimal-sampling-inspired self-triggered control.

[9] Karl J Åström and Björn Wittenmark. *Computer-controlled systems: theory and design*. Courier Corporation, 2013.

[10] Enrico Bini and Giuseppe M Buttazzo. The optimal sampling pattern for linear control systems. *IEEE Transactions on Automatic Control*, 59(1):78–90, 2014.

[11] Ricardo Fernández del Busto y Ezeta. *Análisis y diseño de sistemas de control digital*. McGraw-Hill Interamericana, 2013.

[12] Farid Golnaraghi and BC Kuo. *Automatic control systems*, volume 2. 2010.

[13] https://reference.digilentinc.com/_media/motor_gearbox/290-006_ig220019x00015r_-ds.pdf.

[14] https://www.sparkfun.com/datasheets/Robotics/L298_H_Bridge.pdf.

Apéndice A

Código de programas

A.1. Adquisición de datos para identificación del servomotor

Programa A.1: Adquisición de datos para identificación del servomotor

```
clear all, close all, clc

%Para comunicacin con el arduino. Se debe cambiar el puerto COM
   .
delete(instrfind({'Port'},{'COM3'}));
canal_serie= serial('COM3','BaudRate',115200,'Terminator','CR/
   LF')%Velocidad de comunicin y puerto de comunicacin
fopen(canal_serie);%abrir el canal

A=fscanf(canal_serie,'%f',[1,3]);% resivir la canidad de datos
   de la matris
clc;
i=1;
NumDatos=2000;% comuniquese hasta recibir esta cantidad de
   datos
```

```
Datos=zeros(NumDatos-1,3);%crear a matris para os datos a
   recivir

%%Para llenar la matris de datos
while i<NumDatos
    A=fscanf(canal_serie,'%f',[1,3]);

     Datos(i,1)=A(1,1);
     Datos(i,2)=A(1,2);
     Datos(i,3)=A(1,3);

     i=i+1;
end

 xlswrite('Rampaident.xlsx',Datos);% Crear un documento xlsx
    con los datos alacenados


 %%cerrar la comuncacin
fclose(canal_serie);
delete(canal_serie);
clear canal_serie;
```

A.2. Simulación de control periódico

Programa A.2: Simulación de control periódico

```
clear all, close all, clc,
%% Representacin en espacio de estados
```

```
A=[0 1;
    0 -13.9]
B=[0;
    36.77]
C=[1 0]
D=0;
%% Matriz extendida
G=[A [0;0]
    -C 1]

 H=[B;-D];
 I=[1 0 0]
 J=0;

%% Ganancia de control
Q = [1 0 0 ;
     0 0.1 0 ;
     0 0 1];
R =  [10];

K=lqr(G,H,Q,R)
Ns=50
%% Diagrama de bode
sys = ss(G, H, I, J);
 sys=feedback(sys,1);
 bode(sys)

%% Datos  iniciales
history_x = [];
history_y = [];
history_ref = [];
```

```
history_u = [];
history_time = 0;
history_tau_k = [];
x = [0; 0];
y=[0];
v=0;
vi=0;
error=0;
errorPas=0;
reference = 0;
t=0.15
for n=1:Ns
    % reference
   if(n>5)
        reference =180;
   end
    history_ref = [history_ref, reference];
    history_time = [history_time, history_time(end)+t];
    % Seal de control
    error=reference-x(1);
    v=(error*t)+v;
    u = ((-K(1)*(x(1)-reference))+(-K(2)*x(2)))+(-K(3)*v*K(1));
    vi=errorPas*t; % control u(k)
    history_u = [history_u, u];

    %Evolcin de la dinmica
    [Ad, Bd] = c2d(A, B,t);
    x = Ad*[x(1); x(2)] + Bd*u;
    history_x = [history_x, x];
    y=C*x;
    history_y = [ history_y y];
```

```
    history_tau_k = [history_tau_k, t];
end

%% Grficas
 figure

subplot(3,1,1)
grid on, hold on
plot(history_time(1:Ns), history_x(1,:), history_time(1:Ns),...
    history_x(2,:), history_time(1:Ns), history_ref(1,:),'.',
       ...
    'lineWidth', 2);
xlim([0 history_time(Ns)]),
title('Respuesta')
legend('x_{1(k)}','x_{2(k)}','referencia')

subplot(3,1,2)
grid on, hold on
plot(history_time(1:Ns), history_u(1,:),'lineWidth', 2);
xlim([0 history_time(Ns)])
ylabel('Control (U)')
legend('u(k)')
subplot(3,1,3)
stem(history_time(1:Ns), history_tau_k)
hold on
plot(history_time(1:Ns),...
mean(history_tau_k)*ones(size(history_time(1:Ns))),'r','
   lineWidth', 2)
xlim([0 history_time(Ns)])
title('Secuencia de muestreo')
xlabel('Time (sec)')
```

```
ylabel('{\tau}_k (sec)')
legend('\tau_{(k)}','\tau_{av}')
hold off
```

A.3. Control Periódico en Arduino

Programa A.3: Control Periódico en Arduino

```
1 //declaración de variables
2 volatile float error, errorInt, errorPas, errorAcu, errorCal;
3 volatile float u;
4 volatile float voltaje;
5 //int flag = 2;
6 //int salidal;
7 //volatile char S;
8 volatile byte state, statep;
9 volatile int analogInPin = A0;
10 volatile float  count = 0, countp = 0, desplasamiento = 0;
11 volatile float RPM = 0;
12 int x = 5;
13 int y = 6;
14 volatile float TimeP, timep, time, etime, timep1, timep2,
     etime2, etime1, T = 0;
15 volatile boolean A, B;
16 volatile int timeinterrup = 63973;
17 int Ref = 0;
18 int TimeRef = 100;
19
20 float x1 = 0.0; //states
```

```
21 float x2 = 0.0;
22 volatile float v = 0.0, vacu = 0.0;
23
24 volatile float k11 = 0.9657;
25 volatile float k12 = 0.0993;
26 volatile float Ki = -1.0662;
27
28
29
30 void setup() {
31
32   Serial.begin(115200);// Inicializa la comunicación serial
33
34   pinMode(x, OUTPUT); //In1 giro dle motor
35   pinMode(y, OUTPUT); //In2 giro del motor
36   pinMode(11, OUTPUT); //Ena Activa pwm drive
37   pinMode(A0, INPUT);
38
39
40   attachInterrupt(0, Achange, CHANGE); //denera una
        interrupcion al detectar un cambio en el encoder
41   attachInterrupt(1, Bchange, CHANGE); //denera una
        interrupcion al detectar un cambio en el encoder
42
43   timep = micros();
44   A = digitalRead(2);
45   B = digitalRead(3);
46   //Declaracion de los estados posibles entre la lectura de los
         canales
47   if ((A == HIGH) && (B == LOW))statep = 1;
48   if ((A == HIGH) && (B == HIGH))statep = 2;
```

```
49   if ((A == LOW) && (B == HIGH))statep = 3;
50   if ((A == LOW) && (B == LOW))statep = 4;
51   analogWrite(x, LOW);
52   analogWrite(y, LOW);
53
54   // Initialize Timer1
55   // normal mode
56   // T = (1/Tosc)*(2^resolution-TCNT1)*prescaler
57   // T = (1/16Mhz)*(2^16-64031)*1024= 100ms
58   noInterrupts(); // disable all interrupts
59   TCCR1A = 0;
60   TCCR1B = 0b00000101;
61
62   TCNT1 = timeinterrup;
63   OCR1A = 0x00;
64   OCR1B = 0x00;
65
66   TIMSK1 = (0 << OCIE1B) | (0 << OCIE1A) | (1 << TOIE1); //
       enable timer2 overflow interrupt
67   interrupts(); // enable all interrupts
68 }
69
70 ISR(TIMER1_OVF_vect, ISR_NOBLOCK ) // interrupt service routine
71 {
72   TCNT1 = timeinterrup; // preload timer
73   AccionControl();
74   envia_datos();
75 }
76
77
78 void AccionControl()
```

```
79 {
80
81
82   Ref = analogRead(A0); //referencia de posición
83   Ref = map(Ref, 0, 909, 0, 180); //tranformación a grados
84   desplasamiento = (count - countp) / 231; //revolución
85   countp = count;
86   etime = (micros() - timep) / 60000000; //delta t
87   timep = micros();
88   RPM = (desplasamiento) / (etime);
89   x1 = (count * 360) / 231;
90   x2 = RPM;
91   T = (1.0 / 16000000) * (pow(2, 16) - 64031.0) * 1024.0;
92   error = (Ref - x1);
93   if (abs(error) < 2)
94   {
95     error = 0;
96     errorInt = 0;
97   }
98   errorInt = errorInt + (T * error);
99   u = ((-k11 * (x1 - Ref)) + (-k12 * x2)) + (-Ki * errorInt *
       k11);
100  voltaje = u;
101
102  if (voltaje > 255)
103  {
104    voltaje = 255;
105  }
106  if (voltaje < -255)
107  {
108    voltaje = -255;
```

```
109   }
110
111   if (voltaje > 0)
112   {
113     analogWrite(x, voltaje);
114     digitalWrite(y, LOW);
115   } else if (voltaje < 0)
116   {
117     digitalWrite(x, LOW);
118     analogWrite(y, abs(voltaje));
119   }
120 }
121
122 void envia_datos()
123 {
124   Serial.print("Ref:");
125   Serial.flush();
126   Serial.print(" ");
127   Serial.flush();
128   Serial.print(Ref);
129   Serial.flush();
130   Serial.print(" ");
131   Serial.flush();
132   Serial.print("x1:");
133   Serial.flush();
134   Serial.print(" ");
135   Serial.flush();
136   Serial.print(x1);
137   Serial.flush();
138   Serial.print(" ");
139   Serial.flush();
```

```
140
141   Serial.print("Control u:");
142   Serial.flush();
143   Serial.print(" ");
144   Serial.print(u);
145   Serial.flush();
146   Serial.print(" ");
147   Serial.flush();
148   Serial.print("errorPas:");
149   Serial.flush();
150   Serial.print(" ");
151   Serial.flush();
152   Serial.print(errorPas);
153   Serial.flush();
154   Serial.print(" ");
155   Serial.flush();
156   Serial.print("errorInt:");
157   Serial.flush();
158   Serial.print(" ");
159   Serial.flush();
160   Serial.print(errorInt);
161   Serial.flush();
162   Serial.print(" ");
163   Serial.flush();
164   Serial.print("error:");
165   Serial.flush();
166   Serial.print(" ");
167   Serial.flush();
168   Serial.print(error);
169   Serial.flush();
170   Serial.print(" ");
```

```
171   Serial.flush();
172   Serial.print("v");
173   Serial.flush();
174   Serial.print(" ");
175   Serial.print(v);
176   Serial.flush();
177   Serial.print(" ");
178   Serial.flush();
179   Serial.println();
180   Serial.flush();
181 }
182
183 void Achange()//Fauncion que detecta el cabio de flanco en el
      canal
184 {
185
186   A = digitalRead(2); //Lectura del canal A
187   B = digitalRead(3); //Lectura del canal B
188
189   if ((A == HIGH) && (B == LOW))state = 1;
190   if ((A == HIGH) && (B == HIGH))state = 2;
191   if ((A == LOW) && (B == HIGH))state = 3;
192   if ((A == LOW) && (B == LOW))state = 4;
193
194   switch (state)
195   {
196     case 1: ;//Si el estedo actual es 1
197       {
198         //Comparación con el estado pasado
199
```

```
200         if (statep == 2) count--; //Si el estado pasado es 2
              decrementar
201         if (statep == 4) count++; //Si el estado pasado es 4
              incrementar
202         break;
203       }
204     case 2:// si el estado actual es 2
205       {
206         //Comparación con el estado pasado
207
208         if (statep == 1) count++; //Si el estado pasado es 2
              decrementar
209         if (statep == 3) count--; //Si el estado pasado es 4
              incrementar
210         break;
211       }
212     case 3:// si el estado actual es 3
213       {
214         //Comparación con el estado pasado
215
216         if (statep == 2) count++; //Si el estado pasado es 2
              decrementar
217         if (statep == 4) count--; //Si el estado pasado es 4
              incrementar
218         break;
219       }
220     default:
221       {
222         //Comparación con el estado pasado
223
```

```
224         if (statep == 1) count--; //Si el estado pasado es 1
               decrementar
225         if (statep == 3) count++; //Si el estado pasado es 3
               incrementar
226       }
227
228   }
229   statep = state; //Actualizamos la variable statep con el
         valor de state
230
231 }
232 void Bchange()//Fauncion que detecta el cabio de flanco en el
      canal
233 {
234
235   A = digitalRead(2); //Lectura del canal A
236   B = digitalRead(3); //Lectura del canal B
237
238   //Declaracion de estados entre las lecturas de los canaes
239   if ((A == HIGH) && (B == LOW))state = 1;
240   if ((A == HIGH) && (B == HIGH))state = 2;
241   if ((A == LOW) && (B == HIGH))state = 3;
242   if ((A == LOW) && (B == LOW))state = 4;
243   switch (state)
244   {
245     case 1: ;//Si el estado actual es 1
246       {
247         //Comparación con el estado pasado
248
249         if (statep == 2) count--; //Si el estado pasado es 2
               decrementar
```

```
250         if (statep == 4) count++; //Si el estado pasado es 4
              incrementar
251         break;
252       }
253     case 2:// si el estado actual es 2
254       {
255         //Comparación con el estado pasado
256
257         if (statep == 1) count++; //Si el estado pasado es 2
              decrementar
258         if (statep == 3) count--; //Si el estado pasado es 4
              incrementar
259         break;
260       }
261     case 3:// si el estado actual es 3
262       {
263         //Comparación con el estado pasado
264
265         if (statep == 2) count++; //Si el estado pasado es 2
              decrementar
266         if (statep == 4) count--; //Si el estado pasado es 4
              incrementar
267         break;
268       }
269     default:
270       {
271         //Comparación con el estado pasado
272
273         if (statep == 1) count--; //Si el estado pasado es 1
              decrementar
```

```
274         if (statep == 3) count++; //Si el estado pasado es 3
               incrementar
275       }
276
277   }
278   statep = state; //Actualizamos la variable statep con el
         valor de state
279
280 }
281
282 void loop() {
283
284 }
```

A.4. Cálculo de polinomios de ganancias para CAAD

Programa A.4: Cálculo de polinomios de ganancias para CAAD

```
clear all, close all, clc,
%%
%Representacin en espacio de estados
A=[0 1;
    0 -13.9];
B=[0;
    36.77];
C=[1 0];
D=0;
%%
%Calculo de matriz extendida
```

```
G=[A [0;0];
    -C 1];

 H=[B;-D];
 I=[1 0 0]
 J=0;
Q = [1 0 0 ;
     0 0.1 0 ;
     0 0 1];
R =  [10];
%%
%Diagrama de Bode
sys = ss(G, H, I, J);
sys=feedback(sys,1);
bode(sys)

%% Configuraciones
tau_max = 0.2;
tau_min = 0.097024;
granularity = 0.000064;
%abs(K*(G+H*K)*[1; 1;1])^alpha
beta =0.59;
alpha = 0.67;
eta = 8;
Ns = 80;  % Number of muestras para la simulacin

%% Garantia de que los perodos de muestro esten en el rango [
   tau_min, tau_max]

(beta^alpha)/eta;
1/tau_min - 1/tau_max;
```

```
%% Kd(tau_k) para todos los posibles tau_k
Kd_tau_k = [];
for tau_k = tau_min : granularity : tau_max
    Kd_tau_k = [Kd_tau_k; lqrd(G,H,Q,R,tau_k)];
end
%% Calculo de las funciones polonoicas para adaptar las
   ganancias del controlador
tau_k = [tau_min: granularity: tau_max]';
h = figure;

% nuevos polinomios para ajustar los elementos de la matriz de
   ganancia del controlador a lo largo de tau_k
K11 = polyfit(tau_k, Kd_tau_k(:,1), 1)
K12 = polyfit(tau_k, Kd_tau_k(:,2), 1)
K13 = polyfit(tau_k, Kd_tau_k(:,3), 1)

Kd_tk_1 = polyval(K11, tau_k);
Kd_tk_2 = polyval(K12, tau_k);
Kd_tk_3 = polyval(K13, tau_k);
hold on
plot(tau_k, Kd_tk_1, 'or', tau_k, Kd_tk_2, 'ob',tau_k, Kd_tk_3,
    'og','lineWidth',1);
plot(tau_k, Kd_tk_1, 'Color','r','lineWidth',1);
plot(tau_k, Kd_tk_2, 'Color','b','lineWidth',1);
plot(tau_k, Kd_tk_3, 'Color','g','lineWidth',1);
set(gca, 'FontSize', 24);
title('Gancias K')

xlabel('\tau_k (seg)');
ylabel('K_{d(\tau_k)}')
```

```
legend('g_{11}(\tau_h)','g_{12}(\tau_h)','g_{13}(\tau_h)','K_{
   d11}(\tau_k)','K_{d12}(\tau_k)','K_{d13}(\tau_k)');

hold off

%% Implementacion en el microcontrolador (lenguaje C)

syms A11 A12 A21 A22
syms B11 B21
syms k11 k12
syms x1 x2
syms r

a = [A11 A12; A21 A22];
b = [B11; B21];
l = [k11 k12];
X = [x1-r; x2];

 betaimplementacion=l*(a+b*l)*X;


%%

simulation1
```

A.5. Simulación de control aperiódico

Programa A.5: Simulación de control aperiódico

```
%% Simulacin sistema ideal

%% valores iniciales
history_tau_k = [];
history_X = [];
history_ref = [];
history_u = [];
history_time = 0;
error=0;
X = [0; 0];
reference = 0;
%%
for n=1:Ns

     if(n>5)
    reference=180;
   end
    history_ref = [history_ref, reference];

    tau_k = tau_max/(1+(tau_max/eta)*(abs(([Kd_tau_k(1)
       Kd_tau_k(2)])*(A+(B*([Kd_tau_k(1) Kd_tau_k(2)])))*[X(1)-
       reference ;(X(2))]))^alpha); % tau(k)
    tau_k = round(tau_k*1000000); % transformar a milisegundos
    tau_k = tau_k/1000000; % transformar a segundos
    history_tau_k = [history_tau_k, tau_k];
    history_time = [history_time, history_time(end)+tau_k];

    % Seal de control
    Kd_tau_k = lqrd(G,H,Q,R,tau_k); % K(tau(k))
```

```
    error=( tau_k*(reference-X(1)))+error;
     u = ((-Kd_tau_k(1)*(X(1)-reference))+(-Kd_tau_k(2)*X(2)))-
        Kd_tau_k(3)*error*Kd_tau_k(1);%
    history_u = [history_u, u];

     %Evolucin de la dinmica
     [Ad, Bd] = c2d(A, B, tau_k);
     X = Ad*[X(1); X(2)]  + Bd*u;  % x(k+1)
     history_X = [history_X, X];
end

%% Graficos del sistema ideal

figure

subplot(3,1,1)
grid on, hold on
plot(history_time(1:Ns), history_X(1,:), history_time(1:Ns),
   history_ref(1,:),'-');
xlim([0 history_time(Ns)]), %ylim([-1.1 1.4])
title('Respuesta')
legend('x_{1(k)}','r_{(k)}')

subplot(3,1,2)
grid on, hold on
plot(history_time(1:Ns),history_u(1,:),'lineWidth', 2);
xlim([0 history_time(Ns)]), %ylim([-1.1 1.4])
xlabel('Time (sec)')
ylabel('Control (u)')
legend('u_{(k)}')
```

```
subplot(3,1,3)
stem(history_time(1:Ns), history_tau_k)
hold on
plot(history_time(1:Ns), mean(history_tau_k)*ones(size(
   history_time(1:Ns))),...
    'r','lineWidth', 2)
xlim([0 history_time(Ns)])
title('Secuencia de muestreo')
xlabel('Time (seg)')
ylabel('{\tau}_k (seg)')
legend('\tau_{(k)}','\tau_{av}')

hold off
```

A.6. CAAD Implementado en Arduino

Programa A.6: CAAD Implementado en Arduino

```
1 //declaración de variables
2
3 char incomingByte;
4 //volatile float  Posi=0, PosiP=0;
5 volatile float error=0.0,errorpas=0.0,errorCal=0.0;
6 volatile float u;
7 volatile float voltaje;
8 volatile byte state, statep;
9 volatile int analogInPin=A0;
10 volatile float  count=0,countp=0,desplasamiento=0;
```

```
11 volatile float RPM=0;
12 int x=5;
13 int y=6;
14 volatile float TimeP,timep, time, etime,timep1,timep2,etime2,
     etime1, T=0;
15 volatile boolean A,B;
16 volatile int timeinterrup=63973;
17 int Ref=0;
18 float x1 = 0.0; //states
19 float x2 = 0.0;
20 float tau_k=0, tau_max = 0.15, tau_min = 0.04, alpha = 0.67,
     eta =10;
21 float A11=0;
22 float A12=1;
23 float A21=0;
24 float A22=-13.9;
25 float B_1=0;
26 float B_2=36.77;
27 float C_1=1;
28 float C_12=0;
29 volatile float k11=0.0;
30 volatile float k12=0.0;
31 volatile float k13=0.0;
32
33 void setup() {
34
35 Serial.begin(115200);// Inicializa ola comunicacionn serial
36 pinMode(x,OUTPUT);//In1 giro dle motor
37 pinMode(y,OUTPUT);//In2 giro del motor
38 pinMode(11,OUTPUT);//Ena Activa pwm drive
39 pinMode(A0,INPUT);
```

```
40
41 attachInterrupt(0,Achange,CHANGE);//genera una interrupción al
     detectar un cambio en el encoder
42 attachInterrupt(1,Bchange,CHANGE);//genera una interrupción al
     detectar un cambio en el encoder
43
44 timep=micros();
45 A= digitalRead(2);
46 B= digitalRead(3);
47 //Declaración de los estados posibles entre la lectura de los
     canales
48 if((A==HIGH)&&(B==LOW))statep=1;
49 if((A==HIGH)&&(B==HIGH))statep=2;
50 if((A==LOW)&&(B==HIGH))statep=3;
51 if((A==LOW)&&(B==LOW))statep=4;
52 analogWrite(x,LOW);
53 analogWrite(y,LOW);
54   noInterrupts(); // disable all interrupts
55   TCCR1A = 0;
56   TCCR1B = 0b00000101;
57
58   TCNT1 = timeinterrup;
59    OCR1A = 0x00;
60   OCR1B = 0x00;
61
62   TIMSK1 = (0<<OCIE1B) | (0<<OCIE1A) | (1<<TOIE1); // enable
       timer2 overflow interrupt
63   interrupts(); // enable all interrupts
64     pinMode(13, OUTPUT);
65 }
66
```

```
67 ISR(TIMER1_OVF_vect,ISR_NOBLOCK ) // interrupt service routine
68 {
69    digitalWrite( 13, HIGH);
70   TCNT1 = timeinterrup; // preload timer
71  AccionControl();
72  //envia_datos();
73   digitalWrite( 13, LOW);
74
75 }
76
77 #define teta1 -0.9101
78  #define teta2 1.2529
79  #define tetab1 -0.0815
80  #define tetab2 0.0886
81  #define tetac1 1.4564
82  #define tetac2 -1.6864
83 void AccionControl()
84 {
85
86    Ref = analogRead(A0);
87     Ref=map(Ref,0,909,0,180);
88   desplasamiento=(count-countp)/231;//revolucion
89      countp=count;
90   etime=(micros()-timep)/60000000;//delta t
91      timep=micros();
92    RPM=(desplasamiento)/(etime);
93   x1 =(count*360)/231;
94   x2 = RPM;
95   tau_k = tau_max/(1+((tau_max/eta)*pow(abs(x2*(k11*(A12 + B_1*
       k12) + k12*(A22 + B_2*k12)) - (k11*(A11 + B_1*k11) + k12*(
       A21 + B_2*k11))*(Ref-x1)),alpha)));
```

```
96   tau_k = round(tau_k*1000000); //transform into milliseconds
       in order to round
97   tau_k = tau_k/1000000;
98   timeinterrup=pow(2,16)-((tau_k*16000000)/1024);
99
100  k11= (teta1*tau_k)+teta2;
101  k12= (tetab1*tau_k)+tetab2;
102  k13= (tetac1*tau_k)+tetac2;
103
104 errorpas=(Ref-x1);
105 error=(errorpas*tau_k)+error;
106
107  if(errorpas<3&&errorpas>-3)
108  {
109    errorCal=0;
110    error=0;
111  }
112  u=((-k11*(x1-Ref))-k12*x2)-k13*error*k11;
113
114 voltaje=u;
115
116 if(voltaje>400)
117 {
118  voltaje=400;
119  }
120 if(voltaje<-4400)
121 {
122  voltaje=-400;
123  }
124
125  if (voltaje>0)
```

```
126 {
127   analogWrite(x,voltaje);
128     digitalWrite(y,LOW);
129 }else if(voltaje<0)
130   {
131  digitalWrite(x,LOW);
132    analogWrite(y,abs(voltaje));
133   }
134 }
135
136 void envia_datos()
137 {
138     Serial.print("tau_k");
139   Serial.flush();
140    Serial.print(" ");
141   Serial.flush();
142   Serial.print(tau_k);
143   Serial.flush();
144    Serial.print(" ");
145   Serial.flush();
146    Serial.print("x1");
147   Serial.flush();
148    Serial.print(" ");
149   Serial.flush();
150   Serial.print(x1);
151   Serial.flush();
152    Serial.print(" ");
153   Serial.flush();
154    Serial.print("x2");
155   Serial.flush();
156    Serial.print(" ");
```

```
157   Serial.flush();
158   Serial.print(x2);
159   Serial.flush();
160    Serial.print(" ");
161   Serial.flush();
162   Serial.print("U");
163  Serial.flush();
164   Serial.print(" ");
165 Serial.print(u);
166  Serial.flush();
167   Serial.println(" ");
168     Serial.flush();
169   }
170
171 void Achange()//Función que detecta el cambio de flanco en el
      canal
172 {
173
174   A=digitalRead(2);//Lectura del canal A
175   B=digitalRead(3);//Lectura del canal B
176
177   if((A==HIGH)&&(B==LOW))state=1;
178   if((A==HIGH)&&(B==HIGH))state=2;
179   if((A==LOW)&&(B==HIGH))state=3;
180   if((A==LOW)&&(B==LOW))state=4;
181
182   switch(state)
183   {
184     case 1: ;//Si el estedo actual es 1
185     {
186       //Comparación con el estado pasado
```

```
187
188       if(statep == 2) count--;//Si el estado pasado es 2
             decrementar
189       if(statep == 4) count++;//Si el estado pasado es 4
             incrementar
190       break;
191       }
192       case 2:// si el estado actual es 2
193       {
194           //Comparación con el estado pasado
195
196       if(statep == 1) count++;//Si el estado pasado es 2
             decrementar
197       if(statep == 3) count--;//Si el estado pasado es 4
             incrementar
198       break;
199         }
200           case 3:// si el estado actual es 3
201       {
202           //Comparación con el estado pasado
203
204       if(statep == 2) count++;//Si el estado pasado es 2
             decrementar
205       if(statep == 4) count--;//Si el estado pasado es 4
             incrementar
206       break;
207         }
208         default:
209         {
210           //Comparación con el estado pasado
211
```

```
212       if(statep == 1) count--;//Si el estado pasado es 1
              decrementar
213       if(statep == 3) count++;//Si el estado pasado es 3
              incrementar
214         }
215
216     }
217     statep = state; //Actualizamos la variable statep con el
            valor de state
218
219 }
220 void Bchange()//Fauncion que detecta el cabio de flanco en el
      canal
221 {
222
223  A=digitalRead(2);//Lectura del canal A
224 B=digitalRead(3);//Lectura del canal B
225
226   //Declaracion de estados entre las lecturas de los canaes
227   if((A==HIGH)&&(B==LOW))state=1;
228   if((A==HIGH)&&(B==HIGH))state=2;
229   if((A==LOW)&&(B==HIGH))state=3;
230   if((A==LOW)&&(B==LOW))state=4;
231   switch(state)
232   {
233     case 1: ;//Si elo estedo actual es 1
234     {
235       //Comparación con el estado pasado
236
237       if(statep == 2) count--;//Si el estado pasado es 2
              decrementar
```

```
238       if(statep == 4) count++;//Si el estado pasado es 4
             incrementar
239       break;
240       }
241       case 2:// si el estado actual es 2
242       {
243           //Comparación con el estado pasado
244
245       if(statep == 1) count++;//Si el estado pasado es 2
             decrementar
246       if(statep == 3) count--;//Si el estado pasado es 4
             incrementar
247       break;
248         }
249           case 3:// si el estado actual es 3
250       {
251           //Comparación con el estado pasado
252
253       if(statep == 2) count++;//Si el estado pasado es 2
             decrementar
254       if(statep == 4) count--;//Si el estado pasado es 4
             incrementar
255       break;
256         }
257         default:
258         {
259           //Comparación con el estado pasado
260
261       if(statep == 1) count--;//Si el estado pasado es 1
             decrementar
```

```
262         if(statep == 3) count++;//Si el estado pasado es 3
                incrementar
263           }
264
265       }
266       statep = state; //Actualizamos la variable statep con el
             valor de state
267
268   }
269
270 void loop() {
271
272 }
```