UNIVERSIDAD POLITÉCNICA SALESIANA

UNIDAD DE POSGRADOS

MAESTRÍA EN CONTROL Y AUTOMATIZACIÓN INDUSTRIALES

Proyecto de investigación
y desarrollo previo a la
obtención del Grado de Magíster
en Control y Automatización
Industriales

IMPLEMENTACIÓN DE UN CONTROLADOR PREDICTIVO BASADO EN MODELO CON RESTRICCIONES SOBRE UN MICROCONTROLADOR DE GAMA ALTA

Autor:

Galo Guzmán Guillén

Dirigido por:

Julio Zambrano A. Mg.

IMPLEMENTACIÓN DE UN CONTROLADOR PREDICTIVO BASADO EN MODELO CON RESTRICCIONES SOBRE UN MICROCONTROLADOR DE GAMA ALTA

IMPLEMENTACIÓN DE UN CONTROLADOR PREDICTIVO BASADO EN MODELO CON RESTRICCIONES SOBRE UN MICROCONTROLADOR DE GAMA ALTA

Autor:

GALO FERNANDO GUZMÁN GUILLÉN
Ingeniero Electrónico
Maestría en Control y Automatización Industriales
Unidad de Posgrados
Universidad Politécnica Salesiana

Dirigido por:

JULIO CÉSAR ZAMBRANO ABAD
Ingeniero Electrónico
Magíster en Automatización y Control Industrial

Cuenca - Ecuador

Datos de catalogación bibliográfica

GALO FERNANDO GUZMÁN GUILLÉN
Implementación de un controlador predictivo basado en modelo con restricciones sobre un microcontrolador de gama alta

Universidad Politécnica Salesiana, Cuenca - Ecuador, 2018

MAESTRÍA EN CONTROL Y AUTOMATIZACIÓN INDUSTRIALES

Formato 170 x 240 mm Páginas 86

Breve reseña del autor e información de contacto:

GALO FERNANDO GUZMÁN GUILLÉN
Ingeniero Electrónico
Maestría en Control y Automatización Industriales
galo.guzman@outlook.com

Dirigido por:

JULIO CÉSAR ZAMBRANO ABAD
Ingeniero Electrónico
Magíster en Automatización y Control Industrial
jzambranoa@ups.edu.ec

GUZMÁN GUILLÉN GALO F.
Implementación de un controlador predictivo basado en modelo con restricciones sobre un microcontrolador de gama alta.
IMPRESO EN ECUADOR - PRINTED IN ECUADOR

Índice general

Índice de figuras

Índice de cuadros

A mis padres

PREFACIO

En este proyecto de graduación se presenta los resultados de la investigación realizada para probar la factibilidad de los sistemas embebidos como plataformas para la implementación de controladores modernos.

Los resultados se obtuvieron mediante la implementación práctica de éstos algoritmos de control, aplicados a un “helicóptero” de laboratorio.

Este trabajo supone conocimientos previos de MATLAB, teoría de control clásica y moderna, microcontroladores, y lenguaje de programación C.

PROLOGO

Existe una gran variedad de estrategias de control que están siendo aplicadas en diferentes tipos de procesos, tanto en la industria, como en el área científica y académica. Sin duda el controlador más conocido es el PID, puesto a la facilidad relativa y a la ligereza que conlleva su implementación, mas no es un controlador óptimo.

Por otro lado están los controladores modernos, que durante mucho tiempo fueron implementados exclusivamente en plataformas con potencia de procesamiento considerable, debido a la carga de cálculo que tienen; tal es el caso de los ordenadores de escritorio, cuyo procesador tiene unidad de punto flotante. Dentro de éste grupo de controladores se puede nombrar al controlador predictivo basado en modelo ó MPC, el cual si es un controlador óptimo. El capítulo 1 comprende la base teórica de ésta estrategia de control.

Viendo el problema desde la aplicación, algunos procesos exigen el uso de plataformas ligeras y de bajo consumo para implementar sus controladores, no se puede montar un CPU convencional sobre drone (por dar un ejemplo). Hoy en día, la potencia de procesamiento de los microcontroladores ha subido significativamente, siendo factible implementar una estrategia de control moderno en éste tipo de dispositivos. En el capítulo 2 se da una breve introducción a los sistemas embebidos.

En éste proyecto se realizó la investigación e implementación de un algoritmo de control predictivo sobre un microcontrolador, el cual fue probado sobre un "helicóptero" estático, para medir el desempeño del mismo. Tras los resultados expuestos, se pretende incentivar al lector a que utilice estrategias de control modernas sobre plataformas embebidas; motivarlo a que profundice e investigue más sobre éste tema. En el capítulo 3 se realiza la implementación y se analizan los resultados de ésta investigación.

AGRADECIMIENTOS

A mi familia y seres queridos, por todo el apoyo entregado durante mis estudios y desarrollo de éste proyecto; al Ing. Mg. Julio Zambrano, director de tesis por brindarme su apoyo y transmitirme su experiencia en el tema; al Ing. MC. Ismael Minchala, quien me motivó a desarrollar éste tema; estimados amigos y colegas que han aportado con su granito de arena en mi crecimiento personal y profesional.

Capítulo 1

CONTROL PREDICTIVO BASADO EN MODELO

1.1. Introducción.

El Control Predictivo basado en Modelo (MPC) es una estrategia de control que se consolida en los años 70 en la industria petroquímica. En sus principios estaba orientado a los procesos con dinámica lenta, puesto que requería de una gran capacidad de procesamiento. El avance tecnológico, y la evolución de la técnica, han permitido extender su aplicación a procesos con dinámica rápida y compleja.

La popularidad de ésta estrategia de control dio lugar a algunas variantes, muchas de las cuales usan modelos de predicción lineal, cuya implementación es posible realizar en dispositivos con recursos limitados. Sin embargo, ésta estrategia siguió evolucionando, y en la actualidad existen variantes que utilizan modelos no lineales, posteriormente se presentan varias alternativas.

1.2. Estado del arte.

En la actualidad se pueden encontrar varios tipos de controladores operando en la industria, implementados en el área de la investigación, o en laboratorios; controladores tanto clásicos, como modernos.

Los controladores clásicos como el PID pueden ser programados fácilmente en un PC, un PLC, o en un microcontrolador básico sin mayor esfuerzo, o se pueden conseguir en el mercado por un precio razonable. Los controladores modernos, por otro lado, son más complejos de implementar, requieren de procesadores más potentes [40], aunque

también tienen muchas ventajas en eficiencia, flexibilidad, pueden ser multivariables y pueden tener otros beneficios [3], pero en general éstos controladores son más costosos. El Control Predictivo basado en Modelo es un ejemplo de estas técnicas de control.

Existen varias aplicaciones del MPC que se ejecutan sobre un PC, o un PLC, o cualquier plataforma de control existente. Algunas variantes utilizan programas de desarrollo bajo licencias que pueden ser costosas, e inclusive puede que éstas plataformas requieran de módulos extra, costosos y con un tamaño físico considerable. Aunque existen también alternativas libres, las cuales serán citadas posteriormente.

En el año 2006 se lanzó una propuesta para desarrollar un controlador predictivo embebido en un SoC (System-on-Chip) realizado en la Universidad Lehigh, cuya meta fue la de reducir la precisión en la fabricación del microprocesador al mínimo, de forma que permita tener un buen rendimiento, al igual que un consumo de energía mínimo. La reducción de la precisión permitiría integrar un módulo para procesamiento de números logarítmicos en un espacio de procesador reducido [6].

Últimamente es posible encontrar herramientas de software que permiten implementar el controlador en sistemas embebidos y ventajosamente también existen alternativas libres. Ésto es posible gracias al interés que despertó en la comunidad científica ésta técnica de control, por lo que se han realizado varios estudios acerca de la implementación del algoritmo en estos sistemas. Entre estos estudios, puede citarse el realizado por miembros del "Laboratory of Analysis and Control Systems, National School of Engineering of Tunis Belvedere". En el cual prueban un algoritmo MPC desarrollado e implementado por ellos en un microcontrolador STM32, y lo comparan con un PID sobre la misma plataforma; En éste estudio concluyen que el desarrollo eficiente de los algoritmos de control, produce una baja carga computacional y un alto rendimiento. A pesar de que el PID con anti-windup mostraba de igual manera un rendimiento alto, el MPC mostraba mejores resultados tales como la predicción de la salida y menos oscilación en la señal de control [3].

Existe también un estudio comparativo del PID con el MPC, donde se incluye la detección de fallas. Se concluye que en casos donde se utilice el MPC con detección de fallas, es recomendable programar los algoritmos de control en lenguaje C, dado su alto rendimiento a comparación de herramientas de alto nivel, como LabView. También propone el uso de un FPGA, que efectivamente representa un mayor rendimiento comparado con un microcontrolador, pero de igual manera representa un mayor costo [9].

Se puede citar otro estudio sobre la implementación del MPC realizada en un microcontrolador de gama baja con núcleo ARM, el cual funciona a una velocidad de 48 MHz y ejecuta un RTOS (Real Time Operating System). Se aplica a un robot tipo péndulo, modelado como un sistema invariante en el tiempo, con 8 estados y 2 entradas sujetas a restricciones [42].

Estudios más recientes sobre la implementación del control predictivo en sistemas em-

bebidos, plantean el uso de modelos de predicción basados en la respuesta al escalón, es decir, técnicas de control similares al DMC, donde se consigue un modelo de predicción recursivo. Ésta técnica permitiría optimizar el algoritmo de control, de forma que utilice la menor cantidad de recursos posible, lo cual sería una gran ventaja en sistemas con recursos limitados [26]. Más adelante puede encontrarse más información acerca de ésta técnica de control.

1.2.1. Variantes del método.

Existen varias formulaciones del control predictivo, las cuales han sido utilizadas en la industria durante mucho tiempo, a continuación se presentan éstas variantes y sus características.

Modelo del Proceso.

Las diferentes formulaciones existentes, están basadas en diferentes modelos del proceso. A continuación se citan algunas de ellas:

• Respuesta al escalón: Utilizado por el *DMC* y sus variantes, este modelo se basa en la respuesta al escalón del proceso, cuya salida está dada por:

$$y(t) = y_0 + \sum_{i=1}^{N} g_i \Delta u(t-i), \tag{1.1}$$

donde y_0 es el valor actual de la salida del proceso, g_i son los muestreos de los valores de la salida a consecuencia de la entrada escalón $\Delta u = u(t) - u(t-1)$, y N es el horizonte de predicción.

Tomando a los valores de $y_0 = 0$, entonces la función de predicción viene dada por:

$$\hat{y}(t+k|t) = \sum_{i=1}^{N} g_i \Delta u(t+k-i|t) \tag{1.2}$$

• Función de transferencia: Utilizado por el *GPC* y otros, su concepto se basa en la función de transferencia $G = B/A$ cuya salida es:

$$A(z^{-1})y(t) = B(z^{-1})u(t) \tag{1.3}$$

donde A y B son polinomios de orden *na* y *nb* respectivamente,

$$A(z^{-1}) = 1 + a_1 z^{-1} + a_2 z^{-2} + \cdots + a_{na} z^{-na}$$
$$B(z^{-1}) = b_1 z^{-1} + b_2 z^{-2} + \cdots + b_{nb} z^{-nb}$$

La función de predicción viene dada por:

$$\hat{y}(t+k|t) = \frac{B(z^{-1})}{A(z^{-1})} u(t+k-i|t) \tag{1.4}$$

• Espacio de estados: Utilizado por el *PFC*, se basa en un modelo es espacio de estados, de la forma:

$$x(t) = Ax(t-1) + Bu(t-1) \tag{1.5}$$
$$y(t) = Cx(t) \tag{1.6}$$

donde x es el **vector de estados**, A es la **matriz de estados**, B es la **matriz de control** y C es la **matriz de salida**. El modelo de predicción viene dado por:

$$\hat{y}(t+k|t) = C\hat{x}(t+k|t) = C[A^k x(t) + \sum_{i=1}^{k} A^{i-1} Bu(t+k-i|t)] \tag{1.7}$$

Optimización.

En éste contexto, se puede considerar una función de costo genérica, cuyo objetivo es que la salida futura y siga a una referencia w, dentro de un horizonte establecido, al mismo tiempo que penaliza al esfuerzo de control Δu necesario para conseguirlo. La expresión general sería:

$$J = \sum_{j=N_1}^{N_2} \delta(j)[\hat{y}(t+j|t) - w(t+j)]^2 + \sum_{j=1}^{N_u} \lambda(j)[\Delta u(t+j-1)]^2 \tag{1.8}$$

Donde N_1 y N_2 son los valores mínimo y máximo respectivamente del horizonte de predicción, N_u es el horizonte de control. Los coeficientes $\delta(j)$ y $\lambda(j)$ penalizan el comportamiento relativo a la referencia y la señal de control respectivamente, y $w(t+j)$ es una aproximación de la referencia.

Control por Matriz Dinámica - DMC.

El desarrollo de ésta variante tuvo lugar a final de los años 70, por Cutler y Ramaker [8], trabajadores de Shell Oil Co. y tuvo mucha acogida en la industria, especialmente la petroquímica.

La formulación del *DMC* está basada en el modelo de predicción 1.2, que es la respuesta finita al escalón del proceso, asumiendo que las perturbaciones son constantes a lo largo del horizonte.

Al desarrollar la ecuación 1.2, se obtiene un conjunto de valores g_i con los cuales se formula una matriz llamada Matriz Dinámica, de ahí el nombre de esta técnica.

$$G = \begin{bmatrix} g_1 & 0 & \cdots & 0 \\ g_2 & g_1 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ g_{N_u} & g_{N_u-1} & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ g_{N_2} & g_{N_2-1} & \cdots & g_{N_2-N_u+1} \end{bmatrix} \tag{1.9}$$

La dimensión de G viene dada por el horizonte de control N_u, mientras que N_2 viene dado por el horizontes de predicción. La matriz puede expresarse como:

$$\hat{y} = Gu + f \tag{1.10}$$

El objetivo del algoritmo de control es llevar la salida lo más cerca posible de la referencia, incluyendo una factor de penalización para los cambios en la entrada. Por lo que se aplica la función genérica 1.8.

En caso de no existir restricciones, el problema de control se resuelve directamente, obteniendo un vector u el cual contiene los valores de los futuros incrementos de la señal de control, expresado como:

$$u = (G^T G + \lambda I)^{-1} G^T (\omega - f) \tag{1.11}$$

De 1.11 se toma únicamente el primer valor del vector u como entrada para el proceso.

Mas información acerca de ésta técnica de control, puede encontrase en [39], [8], [17].

Control Predictivo Generalizado - GPC.

Ésta técnica fue desarrollada por Clarke *et al.*, en el año 1987 [17]. La predicción de la salida fue formulada con el modelo CARIMA (controller autoregressive and integrated moving-averange) representada en la ecuación:

$$A(z^{-1})y(t) = z^{-d}B(z^{-1})u(t-1) + C(z^{-1})\frac{e(t)}{1-z^{-1}} \tag{1.12}$$

En el Control Predictivo Generalizado, las predicciones vienen dadas en términos de $\{\Delta u(j); j \geq t\}$, por lo que una función más aproximada de la predicción de la salida vendría dada por:

$$\hat{y}(t+j|t) = G_j(z^{-1})\Delta u(t+j-1) + F_j(z^{-1})y(t) \tag{1.13}$$

En la ecuación 1.13 se puede observar que la predicción de la salida está en función de los valores anteriores de la salida, en términos de $F_j(z^{-1})y(t)$, y de los valores previos y futuros de la señal de control, en términos de $G_j(z^{-1})\Delta u(t+j-1)$.

Al considerar el error futuro, la predicción de la salida se expresa de la siguiente forma:

$$\hat{y}(t+j|t) = G_j(z^{-1})\Delta u(t+j-1) + F_j(z^{-1})y(t) + E_j(z^{-1})e(t+j) \tag{1.14}$$

En la ecuación 1.14 los valores de $\hat{y}(t+j|t)$ para $j > t$ dependen de las señales de control futuras $u(t+j)$.

A lazo abierto, e ignorando la secuencia de error futuro, se puede expresar la predicción óptima como:

$$\hat{y}(t+j|t) = G_j\Delta u(t) + F_j y(t) \tag{1.15}$$

La cual puede expresarse de la siguiente forma:

$$y = Gu(t) + F(z^{-1})y(t) + G'(z^{-1})\Delta u(t-1) \tag{1.16}$$

La ecuación 1.16 puede expresarse como:

$$y = Gu + f \tag{1.17}$$

Estas señales de control se utilizan para alcanzar el objetivo del *GPC* al minimizar la función de costo genérica 1.8.

La solución optima de la función de costo viene dada por:

$$\Delta u = (G^T G + \lambda I)^{-1} G^T (\omega - f) \tag{1.18}$$

La señal de control que se enviará al proceso es igual a:

$$u(t) = u(t-1) + \Delta u \tag{1.19}$$

Mas información acerca de ésta técnica de control, puede encontrase en [39], [8], [17].

Control Predictivo Funcional - PFC.

Este controlador fue desarrollado por Richalet a finales de los años 80 [17], aunque los fundamentos matemáticos fueron establecidos en el año 1968. Utiliza un modelo del proceso en espacio de estados, y puede funcionar con modelos no lineales, así como modelos lineales inestables. El *PFC* tiene dos características: los *puntos de coincidencia* y las *funciones base*.

Los *puntos de coincidencia* se utilizan para simplificar los cálculos considerando únicamente un subconjunto de puntos dentro del horizonte de predicción. La salida deseada y la salida predicha deben coincidir sólo dentro del subconjunto de puntos y no en todo el horizonte de predicción.

Las *funciones base* dependen de las características del proceso y del punto de funcionamiento deseado.

Esta técnica de control toma como base un modelo en espacio de estados como 1.5. La predicción se obtiene al agregar un termino de auto compensación:

$$\hat{y}(t+j|t) = y(t+j) + \hat{e}(t+j|t) \tag{1.20}$$

Entonces, la señal de control futuro se forma por una combinación lineal de las *funciones base*, obteniendo:

$$u(t+j) = \sum_{i=1}^{N_B} \mu_i(t) B_i(j) \tag{1.21}$$

La función de costo J es:

$$J = \sum_{i=1}^{N_H} [\hat{y}(t+h_i) - w(t+h_i)]^2 \tag{1.22}$$

Donde h_i corresponde al número total de *puntos de coincidencia*.

La señal de control viene dada por:

$$u(t) = \sum_{i=1}^{N_B} \mu_i(t) B_i(0) \tag{1.23}$$

Mas información acerca de ésta técnica de control, puede encontrase en [39], [8], [17].

1.2.2. Variantes comerciales.

Existen variantes comerciales del algoritmo de control predictivo, las cuales pueden ser adquiridas bajo licencia, y se puede citar las siguientes:

cpmPlus Expert Optimizer®.

Es una plataforma desarrollada por ABB soporta varias estrategias de control, incluído el MPC. Su programación se realiza por medio de diagramas de bloques y conexiones físicas. La interacción entre la plataforma y la planta se realiza a través de un servidor OPC (OLE para control de procesos) [1].

3dMPC™.

Es una plataforma desarrollada por ABB para la implementación de técnicas de control predictivo multivariable. Puede utilizarse para procesos no lineales, por medio de un conjunto de modelos que se adaptan al punto de funcionamiento. Cuenta con herramientas para sintonización, modelado y análisis fuera de línea. Permite la interacción con una interfaz de usuario a través de un servidor OPC.

Esta plataforma permite sintonizar la respuesta del proceso a cambios en el punto de funcionamiento, a perturbaciones y a la retroalimentación de control de forma independiente, razón por la cual lleva su nombre (3-Dimensional Multivarible Predictive Controller) [2].

Profit® Controller.

Esta plataforma desarrollada por Honeywell, permite implementar de forma sencilla controladores predictivos multivariable. Cuenta con una algoritmo de control robusto patentado, llamado RCA (Range Control Algorithm), que minimiza los efectos de la incertidumbre a la vez que determina los saltos más pequeños para conseguir un control óptimo.

Esta plataforma garantiza el control seguro de procesos industriales complejos y altamente interactivos [19].

Pavilion8®.

Esta plataforma desarrollada por Rockwell Automation sirve para implementar controladores predictivos, considerándolos como una capa de un completo sistema de control cuyo objetivo es reducir costos, reducir emisiones, mejorar la calidad e incrementar la producción. Utiliza modelos multivariable y las mediciones del proceso para calcular las acciones de control futuras, dentro de las limitaciones del proceso [5].

1.2.3. Variantes libres.

También es posible hoy en día encontrar variantes de código abierto y libre en Internet, se citan las siguientes:

ACADO Toolkit.

ACADO es un conjunto de algoritmos para el desarrollo de controladores modernos, brindando soporte para el MPC, NMPC, y otros. Tiene un estimador de estados y de parámetros. Incluye herramientas implementadas de forma eficiente, para simulaciones ODE y ADE, basadas en los integradores Runge-Kutta y BDF. Está escrito en C++, y puede instalarse en sistemas operativos Windows, OS X y Linux. Cuenta también con una interfaz para MATLAB.

Está disponible en línea, para mas información ver [20],[21],[4],[15],[37],[36].

μAO-MPC.

Esta herramienta desarrollada por Laboratory for Systems Theory and Automatic Control permite implementar el control predictivo en microcontroladores. Genera código en lenguaje C, altamente portable, y considerando las limitaciones propias de los microcontroladores, como baja memoria de datos y de programas, baja velocidad de procesamiento, entre otras. Soporta cálculos aritméticos en punto fijo y punto flotante. Ésta herramienta está escrita en Python, y provee una interfaz MATLAB/Simulink para el código generado [41].

jMPC Toolbox.

jMPC es un conjunto de herramientas para MATLAB orientado al desarrollo de controladores predictivos, implementa de forma sencilla MPC lineal, y permite simularlo en ambientes lineales y no lineales. Permite aplicar restricciones lineales tanto en las entradas como en las salidas. Permite probar diferentes algormitmos de programación cuadrática para resolver la función de costo. Esta herramienta fue desarrollada por Industrial Information and Control Centre con la colaboración de la Universidad de Auckland y la Universidad de Tecnología de Auckland [10].

SimuMPC.

SimuMPC es una plataforma de entrenamiento y simulación para control predictivo basada en MATLAB. Permite sintonizar controladores, identificar modelos de procesos y estimar sus estados.

La característica más relevante de ésta plataforma, es que utiliza por separado el ambiente de controlador y proceso, es decir en diferentes estaciones de trabajo ó PCs. Una se usa como controlador, y otra hace las veces de proceso, la comunicación entre las 2 estaciones es a través de UDP. A éste modo de sintonía se lo conoce como “modo en

línea", el cual permite simular situaciones reales que afectan a los procesos industriales, como el ruido, retardo y el tiempo de muestreo [39].

1.3. Fundamento teórico.

Para explicar en forma general la teoría del funcionamiento del Control Predictivo basado en Modelo (MPC), se puede tomar un caso cotidiano.[1]

El equipo de Investigación y Desarrollo de una empresa, planifica a diario las tareas correspondientes al diseño y desarrollo de un producto electrónico. La regla mas importante es que siempre planifican las actividades para las 8 horas de trabajo, aunque solo ejecutan lo que está planificado para la primera hora. Está actividad de planificación se efectúa en cada nueva hora hasta completar la tarea.

Se toma como punto de partida a el trabajo previo completado hasta las 8 a.m. y en base a esto se planifican las tareas para las siguientes 8 horas. Suponiendo que las tareas se dividen en: diseño de esquemas electrónicos, diseño del circuito impreso, codificación del firmware del dispositivo, etc. El cumplimiento de las tareas propuestas dependerá de varios factores, como el esfuerzo que se ponga en la tarea, el trabajo en equipo, el soporte que se reciba de terceros, etc. Estas son las variables manipuladas en el proceso de planificación. Por otro lado, existen limitaciones, como la habilidad para entender a fondo el problema, o la destreza de los ingenieros en el desarrollo software y hardware. Estas son las restricciones físicas y sistemáticas en la planificación. La información previamente obtenida, es primordial para la planificación.

Después de considerar todos los factores, se determina que la tarea de diseño del circuito impreso entrará en las siguientes 8 horas como una función de las variables manipuladas. Entonces se calculan las actividades necesarias, hora por hora, para cumplir con la tarea. Este calculo se basa en los antecedentes y en las restricciones, y se encuentra el mejor camino para alcanzar el objetivo. Como resultado se obtiene una proyección de todas las actividades de 8 a.m. a 4 p.m. Entonces se inicia ejecutando las actividades de la primera hora.

A las 9 a.m. se evalúan los avances de la primera hora. Esta información es utilizada para planificar la siguiente etapa de actividades. Posiblemente no se completaron todas las actividades planificadas, debido quizá a que el modelo no era el adecuado, o por que alguno de los integrantes con mayor experiencia se ausentó. Sin embargo, el equipo debe realizar la evaluación de los avances y usar ésta información actualizada para planificar sus actividades para las siguientes 8 horas. El objetivo puede seguir siendo el mismo o podría cambiar. El lapso de tiempo seguirá siendo el mismo, 8 horas. El proceso de

[1] Ejemplo basado en la explicación del libro [38]

planificación se repite, como si fueran las 8 a.m. dando como resultado una nueva proyección de actividades para las siguientes 8 horas. Entonces se ejecutan las actividades de la primera hora a las 9 a.m. Nuevamente a las 10 a.m. se revisan los avances y se usa la información actualizada para planificar las siguientes 8 horas. Todo este proceso se repite en cada hora, hasta alcanzar el objetivo original.

En éste ejemplo de planificación, se identifican tres elementos clave. El primero es la forma en como predecir lo que va a suceder (modelo); el segundo es el instrumento que permitirá evaluar las actividades actuales (medición); y el tercero es el instrumento que permitirá ejecutar las actividades planificadas (acción de control). Los problemas clave a tomar en cuenta son:

1. La planificación se realiza dentro de una ventana de tiempo fija de 8 horas.
2. Se necesita conocer el estado actual del sistema antes de la planificación.
3. Se hace la mejor aproximación para las 8 horas de trabajo, tomando en cuenta las restricciones, y la optimización se realiza en tiempo real con un horizonte de tiempo deslizante y la ultima información disponible.

Todo el proceso de planificación descrito aquí, es el principio de funcionamiento del MPC.

Ahora se introducen los siguientes conceptos: ventana de tiempo deslizante, horizonte de predicción, control por horizonte deslizante, y función objetivo.

1. Ventana de tiempo deslizante: ventana de tiempo comprendida desde el instante k_i hasta $k_i + T_p$. T_p es constante, mientras que k_i incrementa en cada paso, por ejemplo: En el caso anterior, siempre se planifica para un lapso de 8 horas $T_p = 8$, mientras que k_i incrementa en intervalos de 1 hora, empezando desde las 8 a.m. o sea $k_i = 8$.
2. Horizonte de predicción: indica que tan "lejos" se quiere predecir el futuro. Este parámetro es igual a la ventana de tiempo deslizante T_p; para su aplicación se designa como N.
3. Control por horizonte deslizante: la ventana de tiempo deslizante describe cual sería la trayectoria óptima de la futura señal de control, pero la aplicación directa de éstas acciones generaría discrepancias entre las futuras salidas predichas y las reales, sería un lazo abierto, por lo que, de hecho, solo se aplica la primera entrada de control a la planta, mientras que se ignora el resto. Posteriormente se evalúa el estado del sistema y se vuelve a generar un nuevo problema de optimización, cerrando el lazo de control. A esta estrategia se le llama *Control por horizonte deslizante*, debido a que el horizonte de predicción se va *deslizando* [28].
4. Función objetivo: Para realizar la planificación, es necesario conocer el estado actual en el instante k_i para predecir el futuro. Esta información se denota como

$x(k_i)$ el cual es un vector que contiene información importante, puede ser medido o estimado. También se requiere de un modelo que describa la dinámica del sistema. Un buen modelo dinámico permitirá predecir el futuro de forma precisa y coherente. Por último, para tomar la mejor decisión, se necesita de un criterio que refleje el objetivo. El objetivo está relacionado a una función de error que se basa en la diferencia entre la respuesta real y la deseada. Esta función objetivo a menudo es llamada como función de coste J; la acción acción de control óptima se calcula minimizando la función de coste, dentro de la ventana de optimización.

1.4. Formulación matemática del MPC.

A continuación se desarrolla la formulación del MPC, tomada del libro [38]. Para simplificar el estudio del MPC, se asume que el proceso corresponde a un sistema SISO modelado en espacio de estados y en tiempo discreto, de la forma:

$$x_m(k+1) = A_m x_m(k) + B_m u(k) \tag{1.24}$$

$$y(k) = C_m x_m(k) + D_m u(k) \tag{1.25}$$

Las expresiones 1.24 y 1.25 corresponden al modelo matemático del proceso, donde A_m representa la dinámica (matriz de estados), B_m es la matriz de entrada, C_m es la matriz de salida y D_m representa la matriz de transferencia directa, $x_m \in R^n$ es el vector de estados, $u \in R^m$ es el vector de entrada (control), $y \in R^p$ es el vector de salida.

La expresión 1.24 calcula los estados futuros del proceso $x_m(k+1)$, en función de los estados actuales $x_m(k)$ y de la señal de control aplicada $u(k)$. En base a ésta expresión, el estado actual se escribiría como:

$$x_m(k) = A_m x_m(k-1) + B_m u(k-1) \tag{1.26}$$

Al restar miembro a miembro éstas dos expresiones, se obtiene la siguiente expresión:

$$x_m(k+1) - x_m(k) = A_m(x_m(k) - x_m(k-1)) + B_m(u(k) - u(k-1)) \tag{1.27}$$

Donde,

$$\begin{aligned} x_m(k+1) - x_m(k) &= \Delta x_m(k+1) \\ x_m(k) - x_m(k-1) &= \Delta x_m(k) \\ u(k) - u(k-1) &= \Delta u(k) \end{aligned}$$

Al reemplazar éstas expresiones en 1.27, queda la siguiente expresión:

$$\Delta x_m(k+1) = A_m \Delta x_m(k) + B_m \Delta u(k) \tag{1.28}$$

La expresión 1.28 calcula la variación de los estados futuros, en función de la variación del estado actual y de la variación de la señal de control.

De forma similar, se ejecuta la misma operación para la expresión de salida 1.25, tomando en cuenta que, lo que interesa es predecir la variación de la salida, dando como resultado:

$$\begin{aligned} y(k+1) - y(k) &= C_m(x_m(k+1) - x_m(k-1)) \\ &= C_m \Delta x_m(k+1) \\ &= C_m A_m \Delta x_m(k) + C_m B_m \Delta u(k) \end{aligned}$$

Por último, es necesario conectar a la variable $\Delta x_m(k)$ con la salida $y(k)$, por lo que se define un nuevo vector de estados, de forma que:

$$x(k) = \begin{bmatrix} \Delta x_m(k) \\ y(k) \end{bmatrix} \tag{1.29}$$

Acomodando las operaciones anteriores, se obtiene la formulación del MPC, a partir del modelo en espacio de estados del proceso con un integrador embebido, el cual garantiza un error en estado estacionario nulo, como se indica a continuación:

$$\overbrace{\begin{bmatrix} \Delta x_m(k+1) \\ y(k+1) \end{bmatrix}}^{x(k+1)} = \overbrace{\begin{bmatrix} A_m & O_m^T \\ C_m A_m & 1 \end{bmatrix}}^{A} \overbrace{\begin{bmatrix} \Delta x_m(k) \\ y(k) \end{bmatrix}}^{x(k)} + \overbrace{\begin{bmatrix} B_m \\ C_m B_m \end{bmatrix}}^{B} \Delta u(k) \tag{1.30}$$

$$y(k) = \overbrace{\begin{bmatrix} O_m & 1 \end{bmatrix}}^{C} \begin{bmatrix} \Delta x_m(k) \\ y(k) \end{bmatrix} \tag{1.31}$$

La ecuación de salida original 1.25 considera la matriz D_m, pero debido al principio de horizonte deslizante, donde se requiere la información actual de la planta para las operaciones de predicción y de control, se tiene de forma implícita que $u(k)$ no afecta a $y(k)$, por lo que $D_m = 0$, dando como resultado la expresión 1.31 que carece del término dependiente de D_m.

Tanto en la ecuación 1.30 y 1.31, se observa a O_m que es un vector de ceros cuya dimensión corresponde al orden del sistema.

Por lo tanto, se define el triplete de matrices (A,B,C) que es conocido como el modelo aumentado, el cual será usado para el diseño del controlador predictivo.

1.4.1. Formulación para la predicción de los estados y las salidas.

Asumiendo que en el instante de muestreo k_i, con $k_i > 0$, el vector de estados $x(k_i)$ está disponible durante la medición, entonces $x(k_i)$ es capaz de proveer la información actual del proceso. La trayectoria de la variable de control futura está denotada por:

$$\Delta u(k_i), \Delta u(k_i+1), \ldots, \Delta u(k_i+Nc-1), \tag{1.32}$$

donde N_c se conoce como el *horizonte de control*, que es un parámetro que indica el número de predicciones de la variable de control. Con la información proporcionada por $x(k_i)$, se puede predecir una cantidad N_p de muestras de los estados futuros, donde N_p se conoce como *horizonte de predicción* (también conocido como *ventana de optimización*). Los estados futuros están denotados por:

$$x(k_i+1|k_i), x(k_i+2|k_i), \ldots, x(k_i+m|k_i), \ldots, x(k_i+N_p|k_i), \tag{1.33}$$

donde $x(k_i+m|k_i)$ es la predicción del estado en el instante k_i+m, tomando en cuenta la información actual del proceso $x(k_i)$. El horizonte de control N_c debe ser menor o igual al horizonte de predicción N_p.

A partir del modelo aumentado, en el instante de muestreo k_i se calculan las predicciones de los estados futuros de forma secuencial utilizando el conjunto de acciones de control futuras:

$$\begin{aligned}
x(k_i+1|k_i) &= Ax(k_i) + B\Delta u(k_i) \\
x(k_i+2|k_i) &= Ax(k_i+1|k_i) + B\Delta u(k_i+1) \\
&= A^2x(k_i) + AB\Delta u(k_i) + B\Delta u(k_i+1) \\
&\vdots \\
x(k_i+N_p|k_i) &= A^{N_p}x(k_i) + A^{N_p-1}B\Delta u(k_i) + A^{N_p-2}B\Delta u(k_i+1) \\
&\quad + \cdots + A^{N_p-N_c}B\Delta u(k_i+N_c-1)
\end{aligned}$$

De la misma forma, se calculan las predicciones de las salidas futuras, sustituyendo las predicciones de los estados:

$$\begin{aligned}
y(k_i+1|k_i) &= CAx(k_i) + CB\Delta u(k_i) \\
y(k_i+2|k_i) &= CA^2x(k_i) + CAB\Delta u(k_i) + CB\Delta u(k_i+1) \\
y(k_i+3|k_i) &= CA^3x(k_i) + CA^2B\Delta u(k_i) + CAB\Delta u(k_i+1) + CB\Delta u(k_i+2) \\
&\vdots \\
y(k_i+N_p|k_i) &= CA^{N_p}x(k_i) + CA^{N_p-1}B\Delta u(k_i) + CA^{N_p-2}B\Delta u(k_i+1) \\
&\quad + \cdots + CA^{N_p-N_c}B\Delta u(k_i+N_c-1)
\end{aligned}$$

Es claro ver que todas las predicciones están formuladas en términos de $x(k_i)$ y del desplazamiento de las acciones futuras de control $\Delta u(k_i + j)$, donde $j = 0, 1, \ldots N_c - 1$.

Entonces se definen los vectores:

$$Y = \begin{bmatrix} y(k_i+1|k_i) \\ y(k_i+2|k_i) \\ y(k_i+3|k_i) \\ \vdots \\ y(k_i+N_p|k_i) \end{bmatrix} \tag{1.34}$$

$$\Delta U = \begin{bmatrix} \Delta U(k_i) \\ \Delta U(k_i+1) \\ \Delta U(k_i+2) \\ \vdots \\ \Delta U(k_i+N_c-1) \end{bmatrix} \tag{1.35}$$

Al reducir las expresiones anteriores, se obtiene una formula generalizada de la predicción de la salida, que viene dada por la siguiente ecuación:

$$Y = Fx(k_i) + \Phi\Delta U \tag{1.36}$$

Donde F y Φ son matrices formuladas en base al triplete de matrices del sistema aumentado. La dimensión de F es $(N_p \times n)$ donde N_p es el horizonte de predicción y n corresponde al orden del sistema aumentado, y la dimensión de Φ es $(N_p \times N_c)$, donde N_c es el horizonte de control.

La matriz Φ es una matriz Toeplitz, la cual se crea a partir de calcular la primera columna, y el resto de columnas son el resultado del desplazamiento de la columna anterior.

$$F = \begin{bmatrix} CA \\ CA^2 \\ CA^3 \\ \vdots \\ CA^{Np} \end{bmatrix} \tag{1.37}$$

$$\Phi = \begin{bmatrix} CB & 0 & \cdots & 0 \\ CAB & CB & \cdots & 0 \\ CA^2B & CAB & \cdots & 0 \\ \vdots & \vdots & \cdots & \vdots \\ CA^{Np-1}B & CA^{Np-2}B & \cdots & CA^{Np-Nc}B \end{bmatrix} \tag{1.38}$$

1.4.2. Optimización.

Para una referencia dada $r(k_i)$ en un instante k_i, el objetivo del control predictivo es llevar la salida lo mas cerca posible de la referencia, dentro del horizonte de predicción. En el diseño, éste objetivo se consigue encontrando el mejor vector de acciones de control ΔU, como resultado de minimizar el error entre la referencia y la salida predicha.

El vector de datos que contiene la información de la referencia es

$$R_s^T = \overbrace{\begin{bmatrix}1 & 1 & \dots & 1\end{bmatrix}}^{N_p} r(k_i) \tag{1.39}$$

Entonces se define la función de costo J como

$$J = (R_s - Y)^T (R_s - Y) + \Delta U^T \overline{R} \Delta U \tag{1.40}$$

La ecuación 1.40 minimiza el error entre la referencia y la salida predicha, considerando el tamaño de los incrementos de ΔU. La matriz $\overline{R}$ indica la agresividad de la señal de control, y es el resultado del parámetro $\overline{R} = r_w I_{N_c x N_c}$, donde r_w puede tomar valores de $r_w \geq 0$. Si $r_w = 0$ el control es agresivo, entonces no se toma en cuenta el tamaño del salto ΔU, al minimizar el error $(R_s - Y)^T (R_s - Y)$; Por otro lado, al tener valores de r_w mayores, la minimización del error $(R_s - Y)^T (R_s - Y)$ es menos agresiva, con saltos de ΔU más suavizados.

Con el fin de encontrar el valor óptimo de ΔU que minimice la función de costo J, ésta se expresa de la siguiente forma:

$$J = (\overline{R}_s r(k_i) - Fx(k_i))^T (\overline{R}_s r(k_i) - Fx(k_i)) - 2\Delta U^T \Phi^T (\overline{R}_s r(k_i) - Fx(k_i)) + \Delta U^T \left(\Phi^T \Phi + \overline{R}\right) \Delta U \tag{1.41}$$

Reduciendo la ecuación 1.41, ésta queda expresada de la siguiente forma:

$$J = \Delta U^T \left(\Phi^T \Phi + \overline{R}\right) \Delta U - 2\Delta U^T \left(\Phi^T \overline{R_s} r(k_i) - \Phi^T Fx(k_i)\right) \tag{1.42}$$

La ecuación 1.42 es una representación más práctica de la función de costo, por lo que se utiliza ésta forma para la implementación.

Aplicando la primera derivada a la ecuación 1.42, se obtiene la siguiente expresión:

$$\frac{\partial J}{\partial \Delta U} = 2(\Phi^T \Phi + \overline{R}) \Delta U - 2\Phi^T (\overline{R_s} r(k_i) - Fx(k_i)), \tag{1.43}$$

la condición necesaria para minimizar la función de costo, se obtiene al igualar:

$$\frac{\partial J}{\partial \Delta U} = 0, \tag{1.44}$$

La solución óptima de la señal de control, da como resultado la ecuación:

$$\Delta U = (\Phi^T \Phi + \overline{R})^{-1}(\Phi^T \overline{R}_s r(k_i) - \Phi^T F x(k_i)) \tag{1.45}$$

Donde $(\Phi^T \Phi + \overline{R})$ es conocido como matriz Hessiana. $\Delta U \in R^{N_c}$ es el vector de incrementos futuros de la variable de control, su dimensión depende del horizonte de control y se utiliza únicamente el primer elemento para calcular la acción de control que se envía a la planta.

1.4.3. Optimización con restricciones.

Cuando la función de costo está sometida a restricciones, el problema de optimización se expresa como un problema de programación cuadrática, de la forma:

$$f(x) = \frac{1}{2} x^T E x + x^T F \tag{1.46}$$

$$Mx \leq \gamma \tag{1.47}$$

Para resolver éste problema cuadrático se utiliza el método de los multiplicadores de Lagrange, el cual permite encontrar el máximo y mínimo locales de una función sujeta a restricciones; y da como resultado una expresión de la forma:

$$J(x, \lambda) = \frac{1}{2} x^T E x + x^T F + \lambda^T (Mx - \gamma) \tag{1.48}$$

Cuando se satisface la condición $Mx = \gamma$, la expresión 1.48 es igual a la función objetivo original. Para minimizar la nueva función de costo, se aplica la primera derivada parcial con respecto a las variables x y λ e igualados a cero da como resultado las siguientes ecuaciones:

$$\frac{\partial J}{\partial x} = Ex + F + M^T \lambda = 0 \tag{1.49}$$

$$\frac{\partial J}{\partial \lambda} = Mx - \gamma = 0 \tag{1.50}$$

La resolución de éste sistema de ecuaciones permite encontrar las expresiones para calcular los valores de x y λ óptimos. A los elementos del vector λ se los conoce como multiplicadores de Lagrange.

$$\lambda = -\left(ME^{-1}M^T\right)^{-1}\left(\gamma + ME^{-1}F\right) \tag{1.51}$$

$$x = -E^{-1}\left(M^T \lambda + F\right) \tag{1.52}$$

La ecuación 1.52 puede ser expresada como dos términos, de la siguiente forma:

$$x = -E^{-1}F - E^{-1}M^T\lambda = x^0 - E^{-1}M^T\lambda \tag{1.53}$$

donde $x^0 = -E^{-1}F$ se conoce como la solución óptima global, la cual corresponde al mínimo de la función de costo original $f(x)$ sin restricciones (ver ecuación 1.46), mientras que el segundo término es una corrección producida por las restricciones.

Aplicación de las restricciones al control predictivo.

Una de las características más atractivas del control predictivo, es la incorporación de restricciones, cualidad que permite imponer limitaciones al momento del calcular la acción de control; limitaciones que se ven reflejadas en impedimentos de carácter numérico, tanto para las variables manipuladas como para las salidas del proceso.

Un proceso normalmente involucra un conjunto de componentes eléctricos, electrónicos, mecánicos etc. Los cuales tienen limitaciones físicas. Por ejemplo, una servo-válvula no puede abrirse a más del 100 %, o un amplificador no puede entregar un voltaje mayor que el de saturación. Entonces el algoritmo de control debe ser capaz de calcular una acción de control real, acorde a las restricciones físicas del proceso. En los problemas de ingeniería de control es muy común encontrar tres tipos de restricciones, que se son las siguientes:

- Restricción de la variación incremental de la señal de control.
- Restricción de la amplitud de la señal de control.
- Restricción de la salida.

Estas restricciones se expresan en la forma $Mx = \gamma$ donde x en éste caso, viene dado por el salto de la variable de control, quedando expresada como:

$$M\Delta U = \gamma \tag{1.54}$$

$$\begin{bmatrix} -I \\ I \\ -\rho_2 \\ \rho_2 \\ -\Phi \\ \Phi \end{bmatrix} \Delta U \leq \begin{bmatrix} -\Delta U^{min} \\ \Delta U^{max} \\ -U^{min} + \rho_1 u(k_i - 1) \\ U^{max} - \rho_1 u(k_i - 1) \\ -Y^{min} + Fx(k_i) \\ Y^{max} - Fx(k_i) \end{bmatrix} \tag{1.55}$$

Donde ΔU^{min}, ΔU^{max}, U^{min}, U^{max}, Y^{min} y Y^{max} son las restricciones mínimas y máximas de la tasa de cambio de la acción de control, amplitud de la acción de control y la amplitud de la salida respectivamente. I es una matriz identidad, ρ_1 es un vector unidad y ρ_2

es una matriz triangular inferior. Las dimensiones de I, ρ_1 y ρ_2 depende del número de puntos en donde se apliquen restricciones.

La solución óptima ΔU viene dada por la expresión 1.53, donde $x = \Delta U$, $E = (\Phi^T \Phi + \overline{R})$ y $F = -\left(\Phi^T \overline{R_s} r(k) - \Phi^T F x(k)\right)$.

Por lo tanto, reemplazando los valores se obtienen las siguientes expresiones:

$$\Delta U = (\Phi^T \Phi + \overline{R})^{-1} \left(\Phi^T \overline{R_s} r(k) - \Phi^T F x(k)\right) - (\Phi^T \Phi + \overline{R})^{-1} M^T \lambda \tag{1.56}$$

$$\Delta U = \Delta U^0 - (\Phi^T \Phi + \overline{R})^{-1} M^T \lambda \tag{1.57}$$

Donde ΔU^0 es la solución global del problema de optimización sin restricciones, expresado en la ecuación 1.45.

Capítulo 2

SISTEMAS EMBEBIDOS

En éste proyecto se desarrolló un algoritmo de control predictivo basado en modelo con restricciones, el cual fue implementado sobre un microcontrolador. Éste algoritmo define el comportamiento del proceso sobre el cual fue aplicado, haciendo uso de las mediciones (entradas del microcontrolador) y en base a éstas se definen las acciones de control (salidas del microcontrolador). Para ésto, se utilizó una tarjeta de desarrollo llamada ChipKit WiFire. En general, estos dispositivos son conocidos como **sistemas embebidos**, como se verá a continuación.

2.1. Introducción

Un sistema embebido es todo sistema electrónico que contiene al menos un dispositivo de control, o sea "un cerebro", de tal forma que sea invisible para el usuario final. Es decir, el controlador está **embebido** en el sistema sin que el usuario note su presencia[12].

Un sistema embebido puede ser un automóvil, un televisor, un teléfono celular, una lavadora, una refrigeradora, una licuadora, una cámara, un reproductor MP3, un reproductor de DVD, un enrutador de red inalámbrico, varios dispositivos Bluetooth como teclados, ratones, audífonos, etc...

Todos estos sistemas tienen algo en común, es decir, todos tienen entradas y salidas que sirven para interactuar con el entorno, así como un mecanismo de control que determina el comportamiento del sistema completo. Antiguamente en los años 1960, 70 y 80, estos mecanismos de control eran específicos de cada dispositivo, contenían una gran cantidad de circuitos electrónicos para comandar tanto entradas como salidas y desarrollarlos era una tarea larga y tediosa. Luego, estos sistemas fueron evolucionando de forma que reutilizaron la mayor cantidad de circuitos electrónicos posible, optimizando costos y

liberando carga de trabajo a los ingenieros de desarrollo. La mayor parte de los circuitos críticos que se reutilizaron, fueron incluidos en el cerebro del sistema.

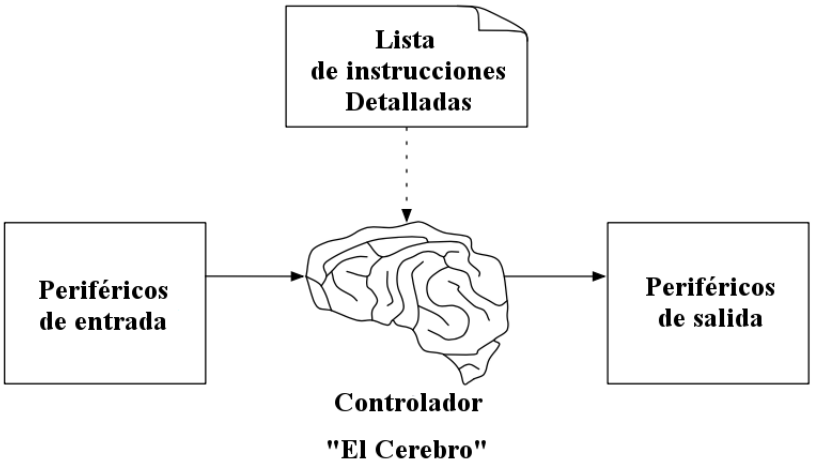


Figura 2.1: *Diagrama de bloques conceptual de un sistema embebido*[1]

Todos estos circuitos utilizados anteriormente como mecanismos de control, fueron reemplazados por, al menos, un sólo componente, sea éste un Dispositivo Lógico Programable (PLD), un microprocesador, o un microcontrolador.

En resumen, un sistema embebido es cualquier artefacto que contiene entradas y salidas, las cuales interactúan a través de un controlador, el cual tiene un programa que define su comportamiento.

El programa que define el comportamiento del sistema puede ser programado en lenguaje ensamblador directamente, o en la mayoría de los casos existen compiladores para C o C++.

2.2. Arquitectura de un sistema embebido

La evolución de los sistemas embebidos van de la mano con la filosofía de proveer un dispositivo de bajo consumo y bajo costo, por lo que cada microcontrolador o microprocesador tiene una arquitectura específica acorde al uso que se le vaya a dar.

La mayoría de los microcontroladores incluyen los módulos que son indispensables para el funcionamiento de un sistema embebido, como por ejemplo el CPU, la memoria

[1] Imagen traducida, tomada del libro [12]

RAM, la memoria FLASH, los buses de datos, los puertos GPIO, entre otros.

Adicionalmente incluyen módulos de comunicación como el USART, SPI, I^2C, USB, entre otros. También puede incluir módulos de conversión analógica/digital.

Algunos dispositivos incluyen módulos que pueden ser utilizados para el control de actuadores, como por ejemplo el PWM, que puede utilizarse para el control de motores, de servo actuadores, motores sin escobillas, etc.

2.3. Plataformas de desarrollo embebido

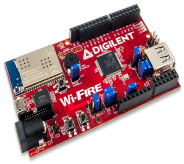
Las plataformas de desarrollo embebido, nacen como herramientas para facilitar el desarrollo de los programas de control para los sistemas embebidos. En la actualidad es posible encontrar varias de estas plataformas, algunas microcontroladas como el Arduino[27], otras microprocesadas como el Raspberry Pi[29], algunas con fines específicos por ejemplo los DSP de la marca Analog Devices. Existen también otras alternativas, que consisten en sistemas con microcontroladores de gama alta, los cuales funcionan a frecuencias relativamente altas, y disponen de FPU (Unidad de Punto Flotante), entre estos sistemas se puede citar los siguientes ejemplos: la tarjeta Discovery Kit de STMicroelectronics, la cual tiene un microcontrolador STM32F7 con el nucleo ARM®Cortex®-M7[33], ó la tarjeta Wi-Fire de Digilent, la cual tiene el microcontrolador PIC32MZ con el núcleo MIPS M5150[13].

En la tabla 2.1 se muestra información comparativa de las plataformas embebidas más populares del mercado, las características mas detalladas de cada dispositivo puede encontrarse en [35], [30], [13].

En ésta comparación se puede observar que la plataforma WiFire es superior en recursos a un Arduino, pero está por debajo de un Raspberry Pi. Ciertamente podría considerarse al Raspberry como la mejor opción, sin embargo éste dispositivo tiene sus limitaciones, ya que la respuesta en tiempo real se ve comprometida al requerir de un Sistema Operativo para su funcionamiento, pudiendo ser éste Linux o Windows 10. Por lo tanto, se considera que para ésta aplicación, la tarjeta WiFire es la mejor opción.

2.4. Descripción de la plataforma WiFire.

Para la implementación del controlador, se escogió la tarjeta de desarrollo Wi-Fire, de la marca Digilent, cuyo microcontrolador es el PIC32MZ2048EFG100 que forma parte de la familia PIC32MZ EF. Ésta tarjeta tiene también un convertidor USB-Serial que puede utilizarse para subir programas al microcontrolador o para comunicaciones. Es compatible con el hardware para Arduino. Puede ser programada con MPIDE o también

	 Arduino UNO	 Raspberry Pi 3	 Chipkit WiFire
Nucleo	Atmel ATmega328	ARM Cortex-A53	MIPS M5150
Frecuencia de reloj	16 MHz	1.2 GHz	200 MHz
Ancho de datos (bits)	8	32	32
RAM	2 KB	1 GB	512 KB
Flash	32 KB	16 GB, 32 GB (SD)	2 MB
GPIOs	20	40	43
Tiempo real	Si	No	Si
Precio	$23.38	$35	$79

Cuadro 2.1: *Tabla comparativa de plataformas embebidas*

con las herramientas propias de Microchip, como MPLAB®X IDE; funciona con todos los programadores/depuradores compatibles con MPLAB®X, como el PICKit^{TM}3[13]. La figura 2.2 muestra la apariencia física de la tarjeta donde puede verse su semejanza con el Arduino.

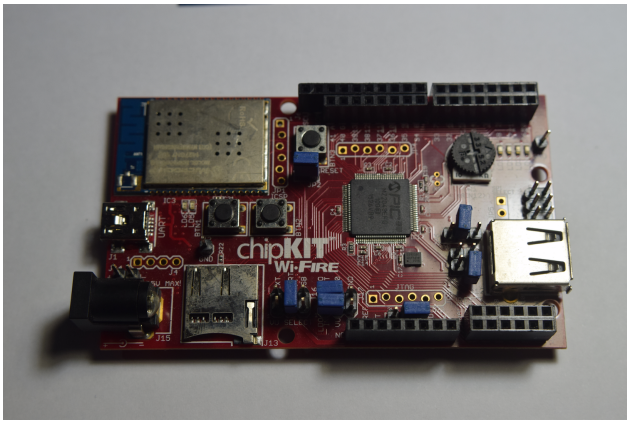


Figura 2.2: *Tarjeta de desarrollo ChipKit Wifire*

La tarjeta tiene las siguientes características:

- Microcontrolador Microchip®PIC32MZ2048EFG100 (200 MHz 32-bit MIPS M5150, 2MB Flash, 512K RAM).

- Modulo WiFi Microchip MRF24WG0MA.
- Conector para tarjeta Micro SD.
- Controlador USB 2.0 Hi-Speed OTG con conectores A y micro-AB.
- 43 pines I/O.
- 4 módulos SPI de 50 MHz, 6 módulos UART, 5 módulos I^2C.
- Cuatro LEDs para el usuario.
- Conexión a PC utilizando el USB A, USB micro B.
- 12 entradas analógicas.
- 3.3 V voltaje de operación.
- 7 V a 15 V voltaje de entrada(recomendado).
- 30 V voltaje de entrada (máximo).
- 0 V a 3.3 V rango del voltaje de entrada analógico.
- Fuente de alimentación conmutada de alta eficiencia de 3.3 V.

Descripción y arquitectura del PIC32MZ.

El PIC32MZ pertenece a la gama alta de microcontroladores de 32 bits de Microchip. Este dispositivo fue creado para brindar al desarrollador una poderosa plataforma para la implementación de aplicaciones embebidas complejas.

En la figura 2.3 se puede ver una comparación de los microcontroladores de 32 bits de Microchip, encabezada por la gama de alto rendimiento con la familia PIC32MZ EF FPU, cuyo núcleo es el MIPS M-Class. También forma parte de ésta gama, la familia SAMS, SAME y SAMV que tienen el núcleo ARM Cortex-M7, similares al STM32F7.

El PIC32MZ tiene 2 buses principales: El *Peripheral Buses* ó bus de periféricos, el cual se encarga de comunicar los módulos del microcontrolador con el bus principal, es decir, temporizadores, comparadores, puertos de entrada y salida, I^2C, SPI, USART, entre otros. Y el bus principal es el *High-Speed Bus Matrix* o Matriz de bus de alta velocidad, el cual comunica las interfaces de alta velocidad, como el bus de memoria, USB, CAN, Ethernet, Encriptación, incluso el bus de periféricos, etc. con el núcleo.

[2]Traducción de la comparación realizada por Microchip[22].
[3]Imagen tomada del sitio web de Microchip[23].

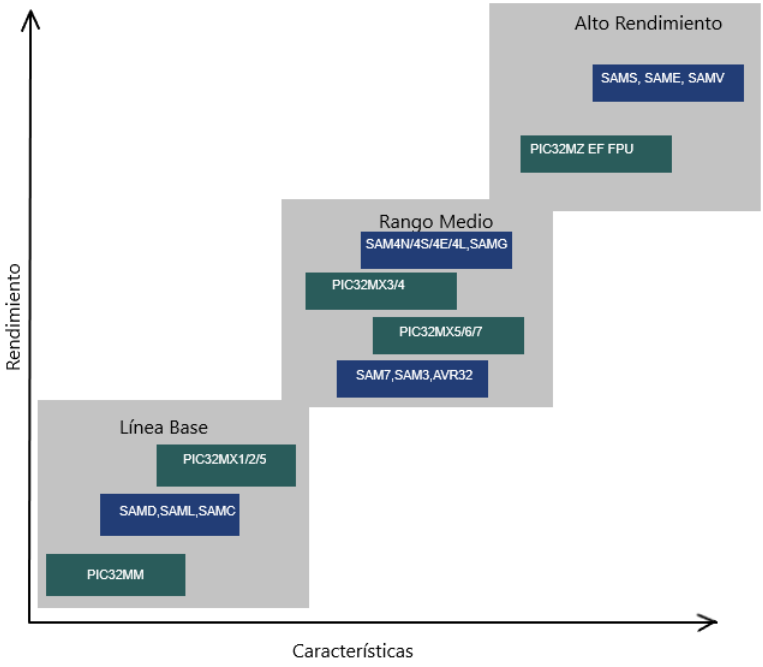


Figura 2.3: *Gama de microcontroladores de 32 bits de Microchip*[2].

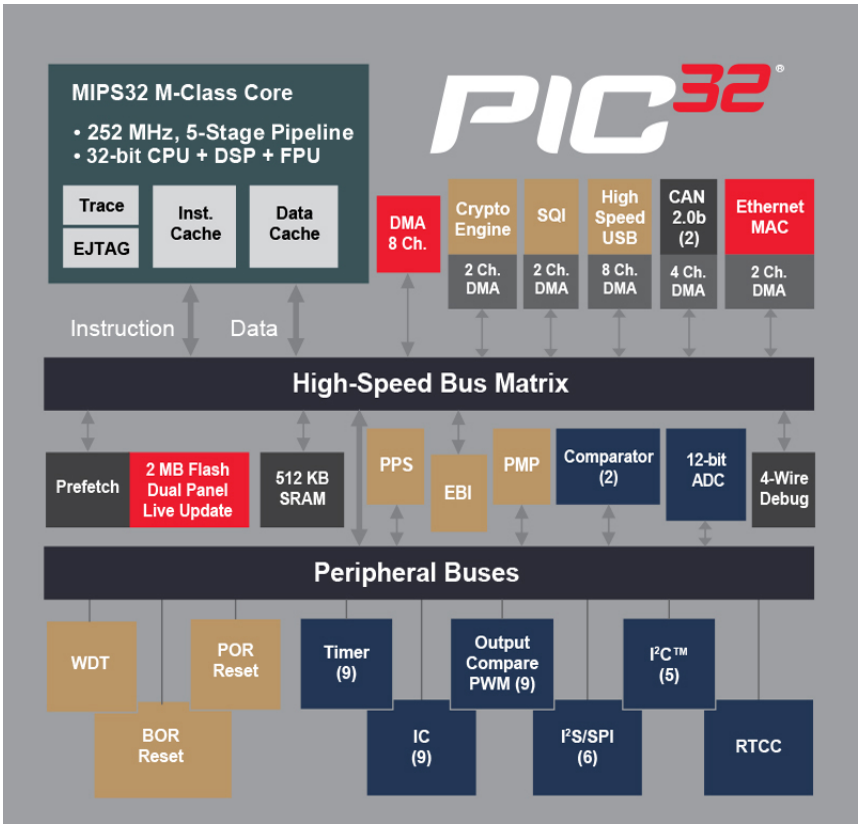


Figura 2.4: *Arquitectura interna del PIC32MZ EF*[3].

En la figura 2.4, se muestra la arquitectura interna de la familia de microcontroladores PIC32MZ EF, donde se puede observar el bloque del núcleo MIPS32 M-Class, que integra el CPU+DSP+FPU.

2.5. Herramientas de desarrollo.

Para el desarrollo del firmware del controlador se utilizó el IDE de Microchip MPLAB®X IDE v3.61, donde se escribió todo el código en lenguaje C. Para la compilación, se utilizó MPLAB®XC32 v1.43, que es el compilador propio de Microchip para el lenguaje C, y destinado a los microcontroladores de 32 bits. Para programar el firmware en el microcontrolador, se utilizó el programador PICKit^{TM}3.

Para la configuración del sistema se utilizó MPLAB®Harmony Integrated Software Framework, herramienta que sirve para configurar el hardware de forma óptima, y generar el código necesario para el funcionamiento básico del microcontrolador.

Para la implementación del algoritmo de control, se utilizó GSL - GNU Scientific Library, que es una librería escrita en lenguaje C, que contiene funciones para realizar operaciones matemáticas avanzadas. Ésta librería se distribuye de forma libre y gratuita, y se utiliza bajo licencia GNU GPL. [4]

Para crear una interfaz amigable en la cual capturar los datos enviados por el controlador, guardarlos y graficarlos, enviar datos y comandos al controlador, se utilizó Qt5, que es un framework multiplataforma escrito en el lenguaje C++, orientado a objetos. Contiene muchas herramientas para el desarrollo de aplicaciones, entre ellas Qt Designer, que permite crear interfaces gráficas de forma sencilla [14]. Existe una versión de Qt5 que se distribuye de forma gratuita y libre, bajo licencia GNU GPL. [5]

Para procesar los datos que fueron adquiridos por el controlador, y la identificación del modelo matemático del proceso, se utilizó Matlab.

[4] https://www.gnu.org/software/gsl/
[5] https://www1.qt.io/es/

Capítulo 3

IMPLEMENTACIÓN Y ANÁLISIS DE RESULTADOS

3.1. Construcción y configuración del proceso a controlar.

3.1.1. Construcción del proceso.

Para probar el desempeño del controlador embebido, se construyó un proceso similar al TRMS (Twin Rotor MIMO System) de la empresa Feedback Instruments (ver figura 3.1), con la diferencia de que tiene un solo grado de libertad, siendo este el ángulo de elevación θ, conocido como pitch.

El proceso está compuesto por las siguientes partes:

- Pedestal
- Actuador
- Sensor
- Controlador

En la figura 3.2 se puede ver el modelo tridimensional del proceso que se construyó para probar le desempeño del controlador. A continuación se describe cada una de las partes con sus respectivos parámetros de funcionamiento.

[1] Imagen tomada del manual del usuario del TRMS[24].

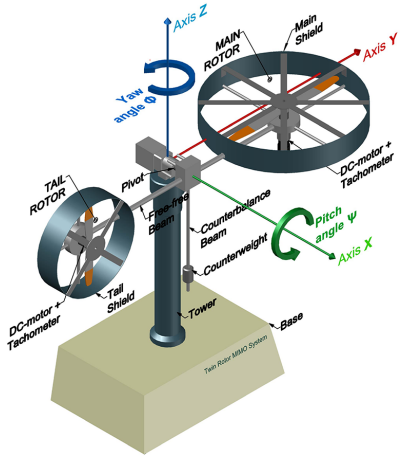


Figura 3.1: *Twin Rotor MIMO System de Feedback Instruments*[1].

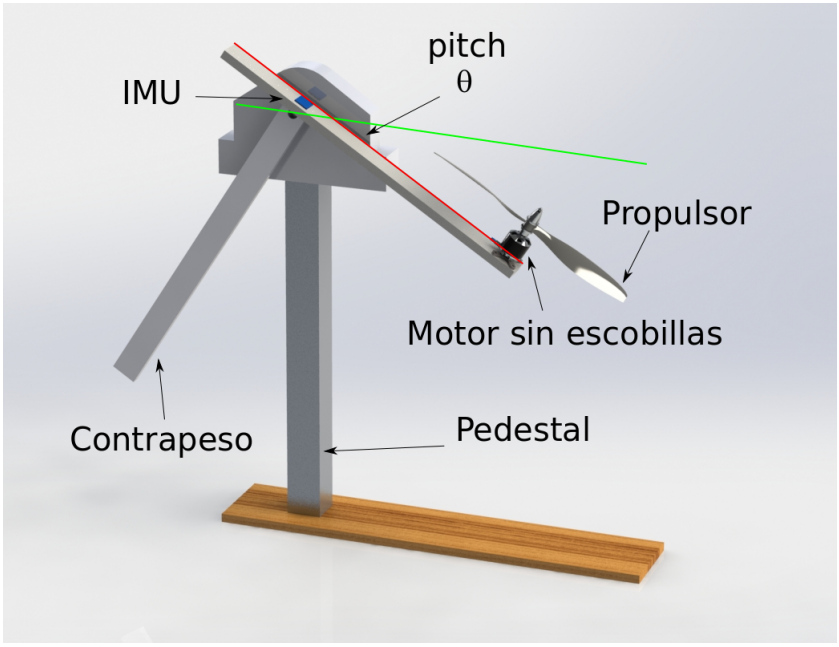


Figura 3.2: *Modelo 3D del proceso construido*

Pedestal.

Es la estructura mecánica sobre la cual está construido el proceso, consiste de un brazo pivotado a un soporte, éste brazo cuenta con un freno mecánico que amortigua el balanceo del mismo.

Actuador.

Está conformado por el motor sin escobillas en conjunto con el propulsor, y el módulo ESC. El actuador varía el ángulo de elevación θ del brazo mecánico, en función de la velocidad de giro del motor. El controlador envía una señal PWM al módulo ESC, el cual imprime la velocidad de giro al motor sin escobillas. La velocidad a la que gira el motor es proporcional a la fuerza de impulso del mismo, por lo tanto el ángulo θ es proporcional al ciclo de trabajo de la señal PWM.

El ESC (Electronic Speed Control) es un dispositivo que, como su nombre lo indica, sirve para controlar la velocidad de giro de un motor, generalmente sin escobillas. Usualmente se utiliza en dispositivos de modelismo, radio-control, drones, y otros. Para imprimir velocidad de giro al motor, se utiliza una señal PWM como entrada para éste dispositivo, la velocidad será proporcional al ciclo de trabajo de la señal PWM. La frecuencia a la que está configurada la señal es de 300 Hz.

El ESC requiere de un proceso de calibración con cada encendido. En un artefacto de radio-control esto se realiza llevando al máximo el acelerador y luego al mínimo. Por lo que se debe realizar el mismo proceso con el controlador en el encendido, en éste caso se programó esta función para que puede ser ejecutada con un comando.

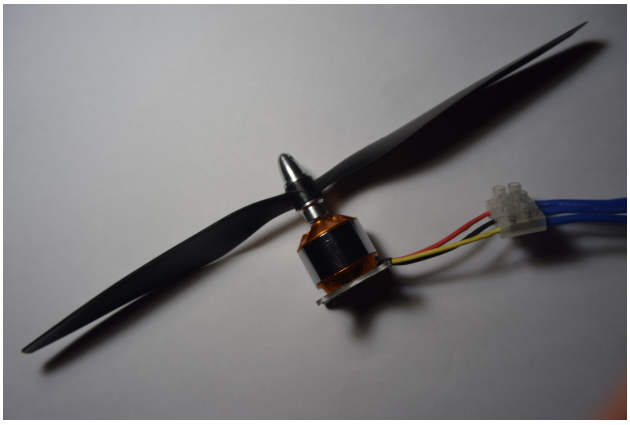


Figura 3.3: *Rotor y propulsor*

La figura 3.3 muestra el motor sin escobillas *A2212 1000KV*, con un propulsor llamado

1045.

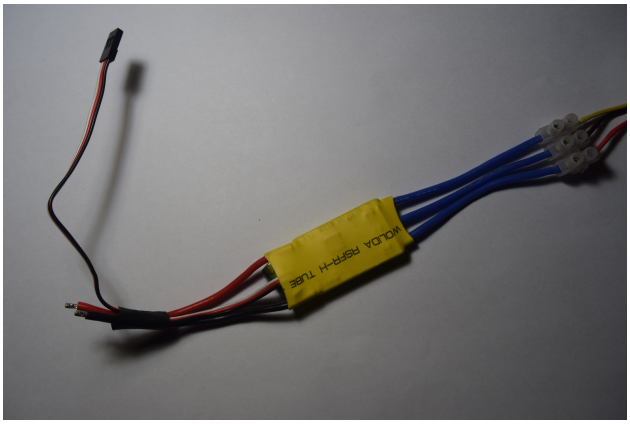


Figura 3.4: *Controlador de velocidad para el rotor*

La figura 3.4 muestra el controlador *ESC 30A*.

Sensor:

Para medir el ángulo θ, se utiliza el IMU ó Unidad de Medición Inercial, de la cual se utilizan 2 sensores, un acelerómetro y un giroscopio.

El acelerómetro entrega los datos de la aceleración para cada uno de los 3 ejes, tomando en cuenta también a la aceleración producida por la gravedad. El módulo está ubicado en el brazo mecánico de tal forma que mida el aceleración en los ejes x y z, los cuales forman el ángulo θ. A continuación se presenta la ecuación que vincula a las mediciones de la aceleración de los 2 ejes con el ángulo θ que se quiere medir [31].

$$\tan\theta_1 = \left(\frac{-acc_x}{acc_z}\right) \tag{3.1}$$

El ángulo se calcula despejando la ecuación anterior, y viene expresado en radianes, por lo que es necesario convertirlo a grados, en caso de requerirlo.

El ángulo medido únicamente con el acelerómetro es sensible al ruido producido por las vibraciones, por lo que se utiliza en conjunto con el giroscopio.

El giroscopio entrega la velocidad angular para cada uno de los 3 ejes. El módulo está ubicado de tal forma que mida la velocidad angular del eje y, entonces una componente del ángulo en función de la velocidad angular, viene dada por:

$$\theta_2 = \int \omega_y dt \tag{3.2}$$

La expresión que se utiliza en la práctica para calcular el ángulo, viene dada por la combinación lineal de las 2 expresiones anteriores y es:

$$\theta(k) = \alpha\theta_1 + \beta(\theta(k-1) + \omega_y dt)), \tag{3.3}$$

donde α y β son los pesos que tienen cada una de las componentes, que para éste caso fueron fijadas en 0,07 y 0,93. A la ecuación 3.3 se le conoce con el nombre de *filtro complementario* [16].

Para este proyecto se utilizó un módulo que incluye el sensor MPU6050 [25], que es un acelerómetro de 3 ejes y giroscopio de 3 ejes, el sensor HMC5883L [18] que es una brújula digital de 3 ejes, y el MS5611 [11] que es un barómetro. En total suma 10 ejes de libertad. La tarjeta se muestra en la figura 3.5.

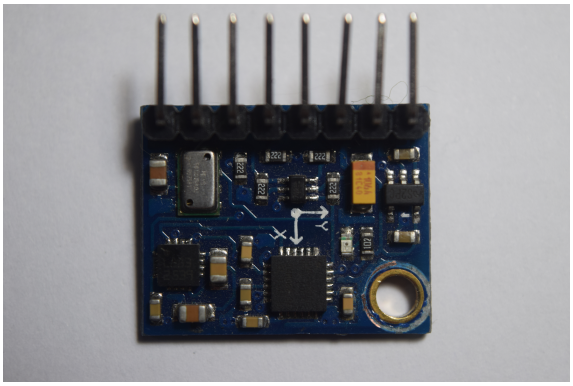


Figura 3.5: *Unidad de medición inercial*

Éste dispositivo se comunica con el microcontrolador a través de un puerto I^2C, configurado a una frecuencia de 400KHz.

Controlador.

El controlador MPC fue implementado en la tarjeta de desarrollo ChipKit WiFire, cuya descripción se encuentra en el Capitulo 2. Ésta tarjeta es la que se encarga de adquirir el ángulo medido con el IMU, procesar los datos con el algoritmo de control, y luego actuar sobre la velocidad de giro del motor a través de la señal PWM que se inyecta al ESC.

3.1.2. Configuración de la plataforma de desarrollo.

Para realizar la configuración de la plataforma, primero es necesario conocer y definir el funcionamiento del hardware de todos los módulos que conforman el controlador, en la sección 2.4 puede encontrarse la descripción de hardware de la plataforma de desarrollo, y en la sección 3.1.1 se encuentra la descripción del *ESC* y del *IMU*.

Conexiones eléctricas.

La tarjeta de desarrollo se alimenta a través del puerto USB, y luego reparte alimentación a través de los pines de voltaje de 3,3V y 5V, de donde se alimentan los módulos *IMU* y *ESC* respectivamente. El módulo del sensor (IMU) se conecta con el puerto de comunicación I^2C de la tarjeta de desarrollo, por otro lado, el módulo del actuador (ESC) se conecta a una salida *PWM* de la tarjeta de desarrollo(ver figura 3.6).

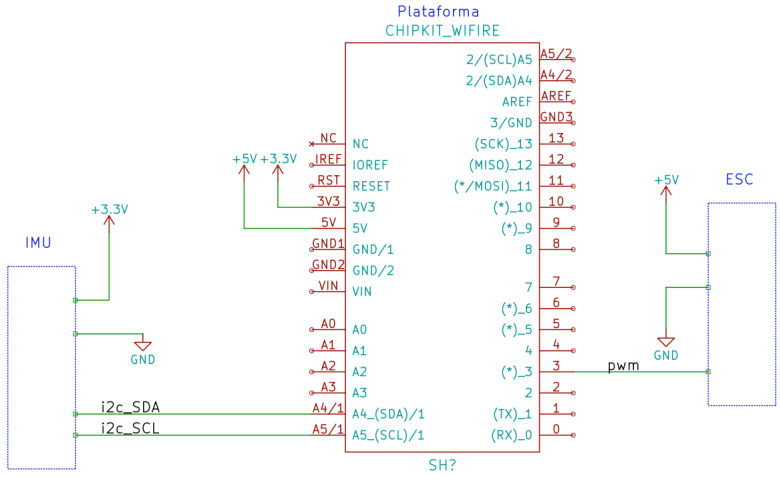


Figura 3.6: *Esquema de conexiones eléctricas*

Configuración del hardware.

Mediante la herramienta MPLAB®Harmony Configurator, se puede realizar la configuración del hardware de forma sencilla y eficiente. Ésta herramienta es un plugin para MPLAB X IDE, y puede ser descargado del sitio oficial de Microchip en la web.

El ChipKit WiFIRE, es una tarjeta reconocida por Microchip, por lo que existe un perfil por defecto en la interfaz de configuración, y puede ser escogida en el menú BSP Configuration, como se puede ver en la figura 3.7.

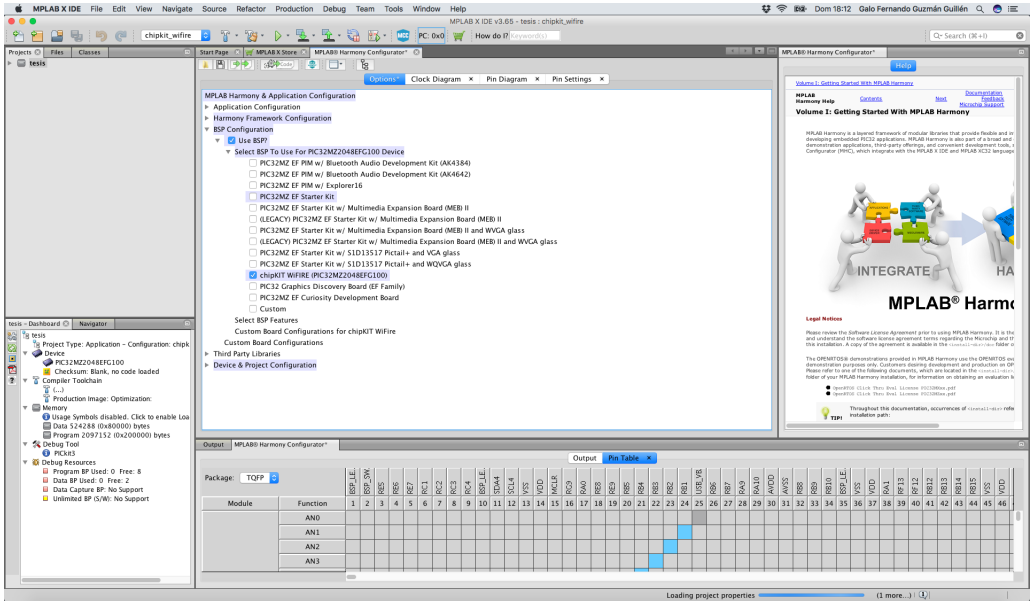


Figura 3.7: *Configuración de la tarjeta de desarrollo*

Al realizar ésta configuración, la tarjeta queda lista para su uso básico, es decir, se configuran los osciladores, relojes del sistema, interrupciones básicas, etc. La tarjeta puede operar en su máximo rendimiento, a una velocidad de procesador de 200 MHz.

La configuración de los periféricos se realiza en la sección *Drivers*, donde se configuran los puertos de comunicación I^2C, *USART* y el módulo *PWM*.

Para la comunicación con el módulo IMU, se utiliza el puerto de comunicación I^2C, el cual se configura para que funcione a una frecuencia de 400 KHz, como se puede ver en la figura 3.8.

La señal *PWM* funciona con 2 módulos, el uno es un temporizador, el cual sirve para fijar la frecuencia de la señal, y el otro es un módulo comparador, que sirve para fijar el ciclo de trabajo.

La configuración del módulo comparador, se muestra en la figura 3.9, el ciclo de trabajo de la señal *PWM* se establece en el campo *OC Pulse Width*, pero se deja en 0 ya que éste valor es calculado por el controlador, y corresponde a un factor de la señal de control $u(k)$.

Además del temporizador que se utiliza para la señal *PWM*, también se utiliza otro para fijar la frecuencia de muestreo. Los dos temporizadores utilizan un reloj de 100 MHz. El temporizador utilizado para el muestreo está identificado como *TMR_ID_1*, y está configurado para funcionar a un periodo de 0.0000025 segundos, entonces para conseguir

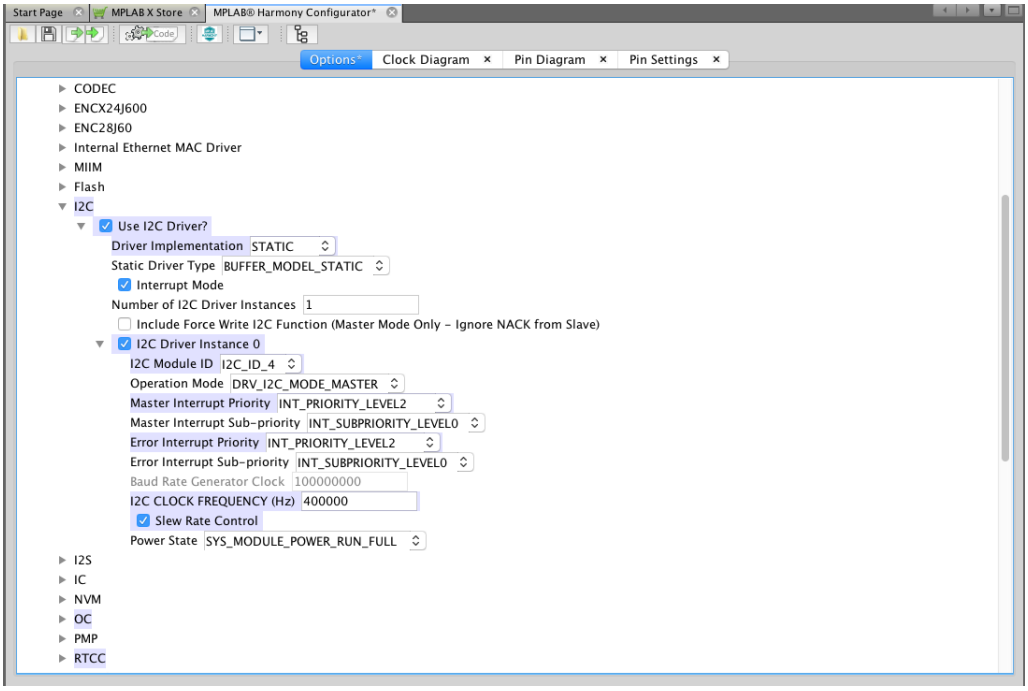


Figura 3.8: *Parámetros para la configuración del puerto de comunicaciones I^2C*

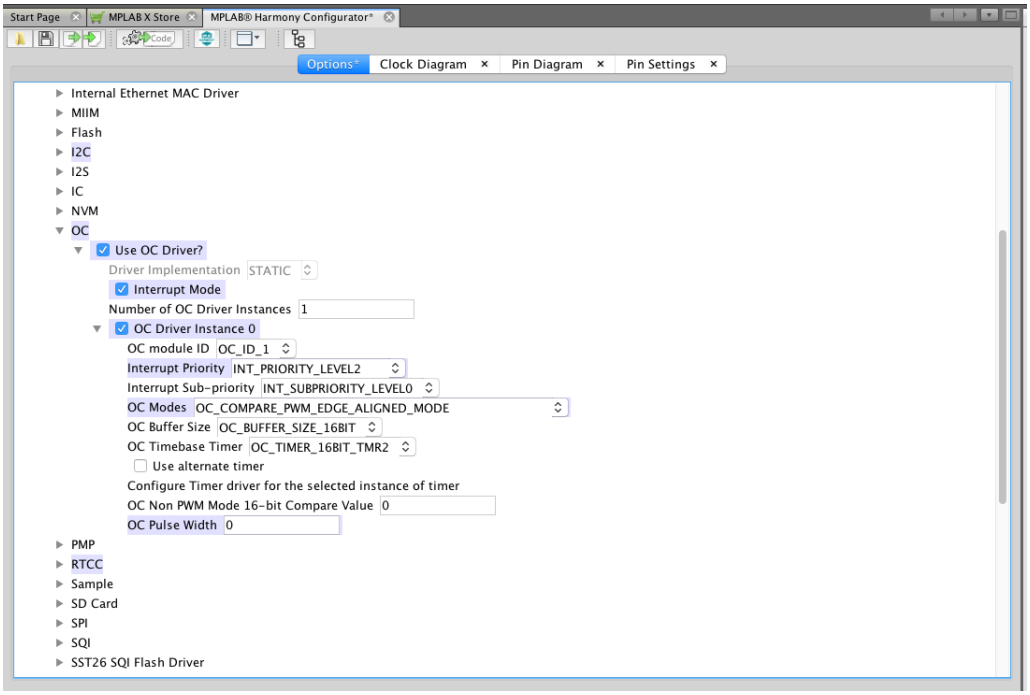


Figura 3.9: *Parámetros de configuración del módulo OC*

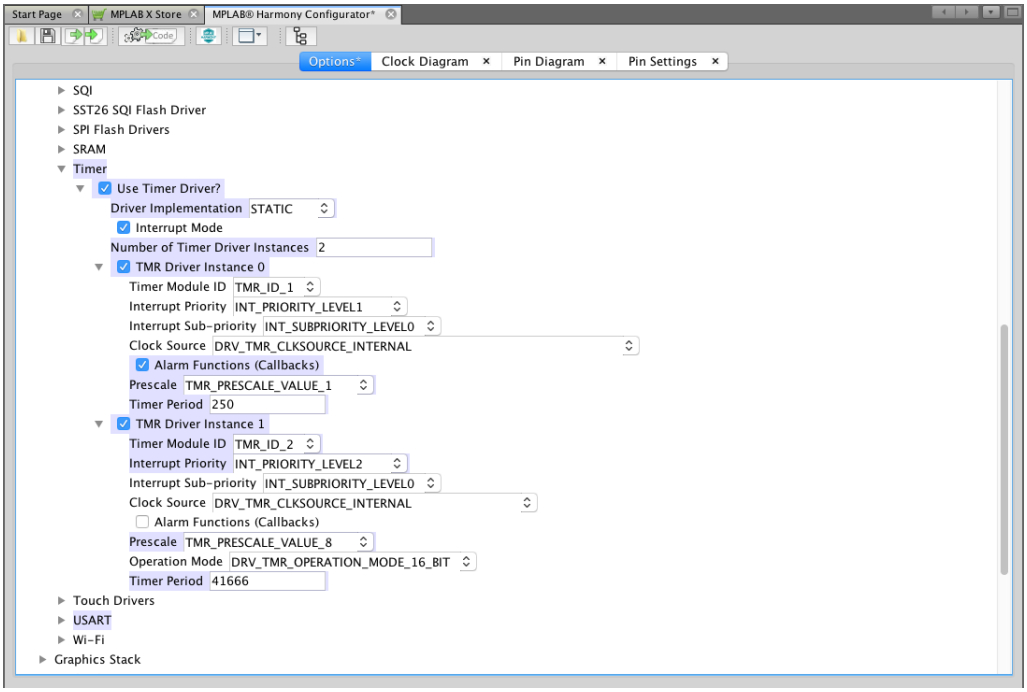


Figura 3.10: *Configuración de los temporizadores*

un tiempo de muestreo de 0.032 segundos, se utiliza un contador establecido en 12800.

Mientras que, para el módulo *PWM* se utiliza el temporizador identificado como *TMR_ID_2* configurado a una frecuencia de 300Hz, para ésto se divide el reloj del temporizador para 8, y luego se establece un contador de 41666 (ver figura 3.10).

Para la interfaz de comunicación con la PC, se configura el módulo *USART* identificado como *USART_ID_4* a una velocidad de 921600 baudios (ver figura 3.11).

Una vez que se ha completado la configuración del hardware, se crea el código fuente presionando el botón *Code*, y posteriormente se presiona el botón *Generate* (ver figura 3.12).

Con ésto queda listo el código fuente básico para la implementación del controlador.

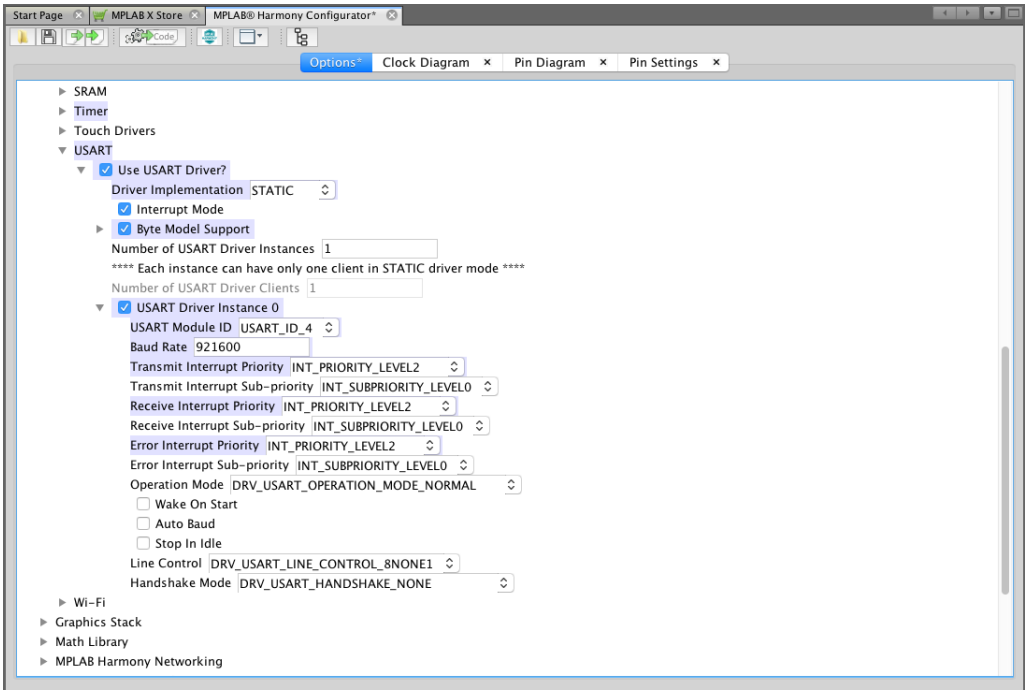


Figura 3.11: *Configuración del puerto serial*

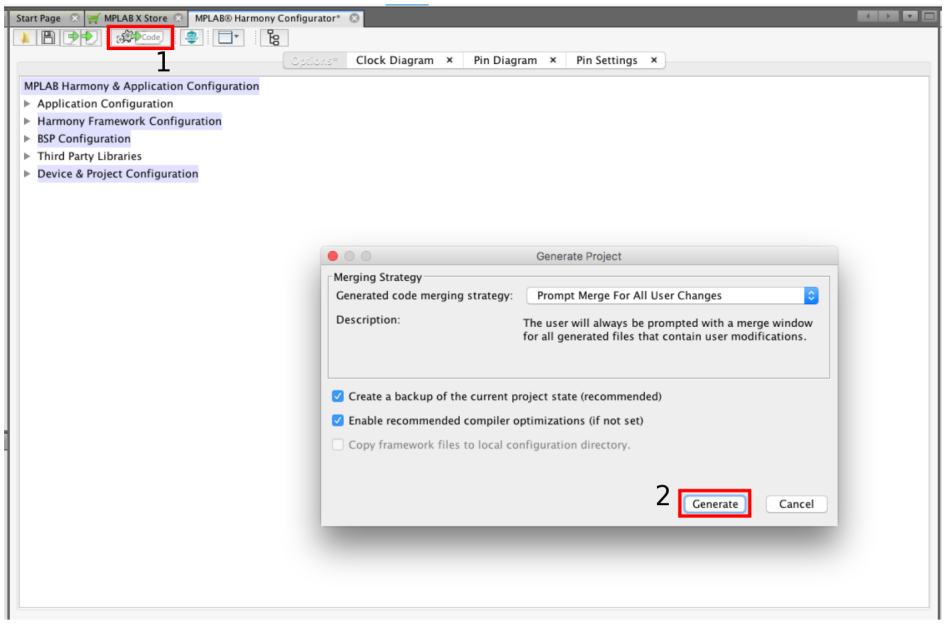


Figura 3.12: *Generando el código fuente básico*

3.2. Identificación del proceso.

El controlador MPC funciona sobre un modelo predictivo en espacio de estados, por lo que es necesario identificar el modelo del proceso, en la forma:

$$x(k+1) = Ax(k) + Bu(k) \tag{3.4}$$
$$y(k) = Cx(k) + Du(k) \tag{3.5}$$

La identificación del modelo matemático se realizó aplicando una secuencia binaria pseudoaleatoria (PRBS) creada en MATLAB con la función `idinput`. Esta señal se aplicó a la planta y se realizó la adquisición de un conjunto de datos para identificación y validación. Los datos de identificación se utilizaron con la función `n4sid` la cual estima el modelo lineal en espacio de estados por medio de un método de subespacios. Los datos de validación se utilizaron en el modelo identificado con la función `compare`, la cual devuelve una gráfica comparativa entre los datos de validación y la salida del modelo identificado, además de un indicador de ajuste de los datos de validación con respecto a la salida del modelo, ésto se tomará en cuenta para la validación.

Si bien el proceso es de naturaleza no lineal, con el método de identificación planteado no se busca conseguir un modelo que represente la no linealidad del mismo, sino su dinámica, por ésta razón se realiza la identificación con una señal PRBS, ya que éste método de muestreo en conjunto con el método de aproximación escogido, permite estimar un modelo que representa la dinámica del proceso.

3.2.1. Determinación del tiempo de muestreo.

Para determinar el tiempo de muestreo del controlador, primero fue necesario encontrar la frecuencia natural de oscilación del proceso, para lo cual se utilizó un método de conmutación de la señal de control para hacer que el proceso entre en resonancia.

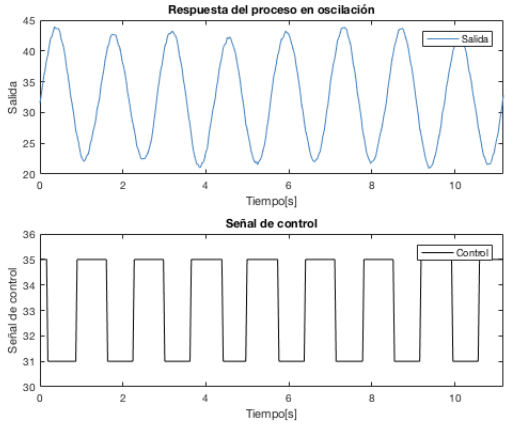


Figura 3.13: *Salida del proceso en oscilación y la señal de control que la produce*

En la figura 3.13 se puede observar la señal de control aplicada al proceso, con un valor de $u = 33\,\% \pm 2$, la cual conmuta entre $31\,\% < u < 35\,\%$, produciendo una oscilación en la salida en un rango de 20.95º $< \theta <$43.94º.

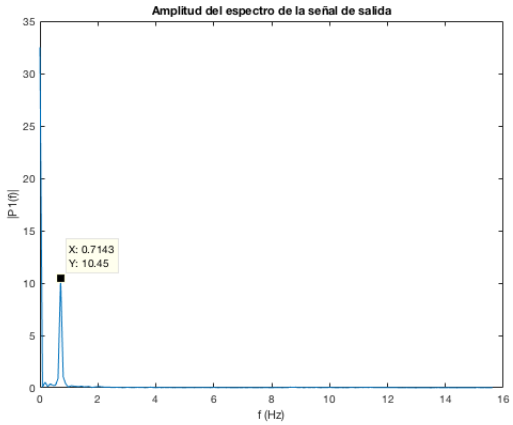


Figura 3.14: *Espectro de frecuencia de la señal de salida*

En la figura 3.14 se puede observar que la frecuencia natural de oscilación del proceso es de 0.7143 Hz, por lo que al aplicar el teorema del muestreo el cuál sostiene que "la frecuencia de muestreo debe ser mayor al doble de la frecuencia máxima", se encuentra que la frecuencia de muestreo debe ser al menos de 1.4286 Hz, aunque en la práctica suele utilizarse una frecuencia igual a 10 veces la frecuencia máxima, dando como resultado una frecuencia de muestreo de 7.143 Hz.

Inicialmente se utilizó una frecuencia de muestreo de 10 Hz para los experimentos realizados, dando resultados exitosos, pero con el fin de obtener una mayor resolución de las mediciones y para probar la fiabilidad del controlador y la capacidad de procesamiento de la plataforma embebida, se sobre dimensionó la frecuencia de muestreo a un valor de 31.25 Hz, el cual da como resultado un tiempo de muestreo de 32 milisegundos.

3.2.2. Adquisición de las señales.

Para la identificación, primero se hizo un muestreo de la salida del proceso con señales PRBS, las cuales fueron generadas tomando en cuenta el valor de la variable de control que produce que el brazo se mantenga en posición horizontal estable. El valor fue $u = 33\,\%$, ésto produce un ángulo aproximado de 32º con respecto al punto de equilibrio, calibrado en 0º. También es necesario determinar dentro de que rango la señal de control conmuta. Se establecieron 3 rangos para la identificación, por lo cual se obtendrán 3 modelos, estos rangos son $u \pm 2\,\%$, $u \pm 4\,\%$, $u \pm 6\,\%$. Con estos parámetros se construyeron las señales en MATLAB, ejecutando el siguiente código:

```
u=33;
du1=2;
du2=4;
du3=6;
prbsSignal1=idinput(10230,'prbs',[0, 0.1],[u-du1, u+du1]);
prbsSignal2=idinput(10230,'prbs',[0, 0.1],[u-du2, u+du2]);
prbsSignal3=idinput(10230,'prbs',[0, 0.1],[u-du3, u+du3]);
```

Se obtuvieron 3 vectores con 10230 muestras pseudoaleatorias que se aplicaron al proceso. La señales de control PRBS conmutan entre estos valores $[31,35]$, $[29,37]$, $[27,39]$ cada una de ellas. Las muestras de la respuesta del proceso a la señal PRBS, se utilizan para la identificación y para la validación.

En la figura 3.15 se muestra la respuesta del proceso y la primera señal PRBS aplicada. La respuesta del sistema a ésta señal PRBS, está dentro de un rango de $23.9686º \leq \theta \leq 42.9962º$, con una media de $\overline{\theta}$ =33.47º, y una desviación estándar de $\pm$ 2.71º.

En la figura 3.16 se muestra la respuesta a la segunda señal PRBS aplicada. La respuesta del sistema a ésta señal PRBS, está dentro de un rango de $15.0837º \leq \theta \leq 50.5045º$, con una media de $\overline{\theta}$ =33.72º, y una desviación estándar de $\pm$ 5.3146º.

En la figura 3.17 se muestra la respuesta a la tercera señal PRBS aplicada. La respuesta del sistema a ésta señal PRBS, está dentro de un rango de $4.1559º \leq \theta \leq 69.0422º$, con una media de $\overline{\theta}$ =34.2953º y una desviación estándar de $\pm$ 9.4626º.

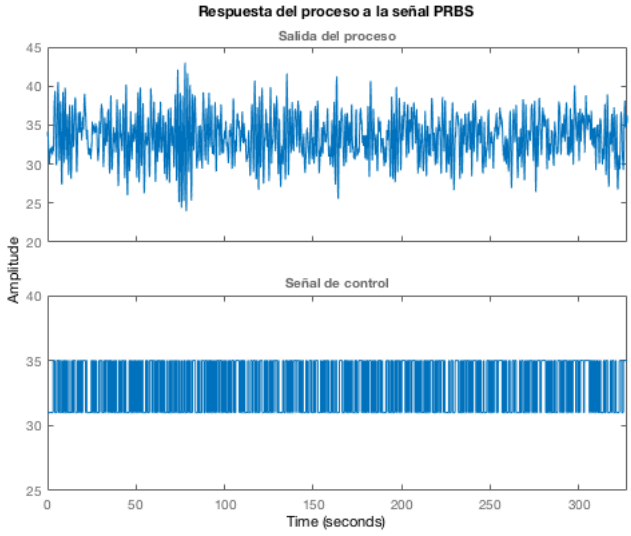


Figura 3.15: *Respuesta del proceso a la señal PRBS con u=33 % ± 2 %.*

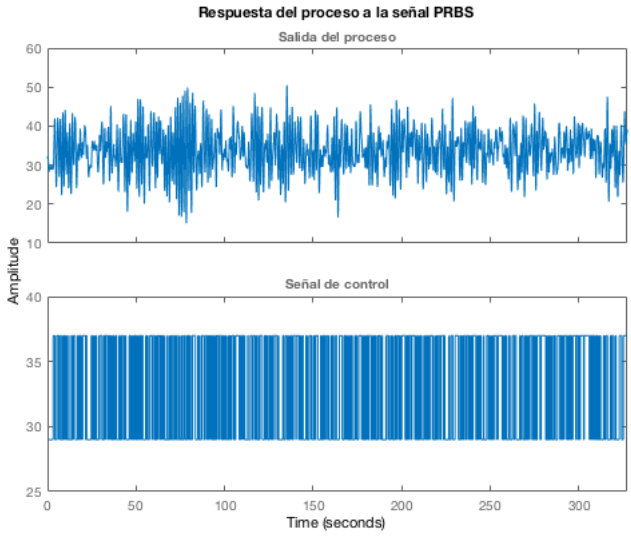


Figura 3.16: *Respuesta del proceso a la señal PRBS con u=33 % ± 4 %.*

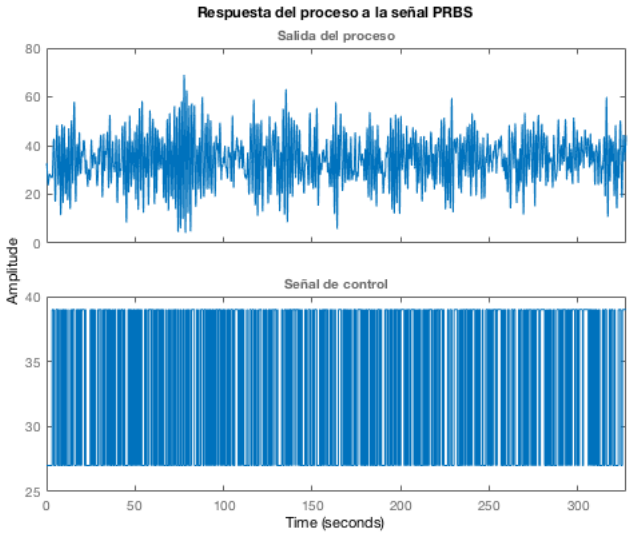


Figura 3.17: *Respuesta del proceso a la señal PRBS con u=33 % ± 6 %.*

De estos conjuntos de muestras se obtienen los datos para la identificación y validación, divididos en una proporción de 60 % y 40 % respectivamente. Los datos de identificación se utilizan con la función `n4sid`, y los datos de validación se utilizan con la función `compare`. Los conjuntos de muestras se forman con la función `iddata`, de la siguiente forma:

```
Ts=0.032;    %   Tiempo de muestreo del sistema

identData=iddata(yi,ui,Ts,'InputName','velocidad', \\
'OutputName','angulo');
validateData=iddata(yv,uv,Ts,'InputName','velocidad',\\
'OutputName','angulo');
```

Estos conjuntos contienen la información de entrada, salida y tiempo de muestreo. Se entiende que los pares de vectores `ui,yi`, y `uv,yv` contienen las muestras para identificación y validación respectivamente.

3.2.3. Identificación de los modelos.

Una vez levantada la información necesaria para la identificación, se procedió a realizar la misma por medio de la función `n4sid` de MATLAB, la cual recibe como parámetros

al conjunto de datos `identData`, el orden del modelo resultante, su forma, estados iniciales, entre otros.

```
%   comentarios intencionalmente sin acento
nx=3;    %   Orden del modelo matematico a identificar
opt = n4sidOptions('InitialState','estimate','N4Weight','CVA',\\
'N4Horizon',[15 0 40],'Focus','prediction','Display','on');
sys = n4sid(identData,nx,'Form','canonical','DisturbanceModel',\\
'none',opt);
```

Con ésta configuración, la función `n4sid` devuelve un modelo de tercer grado en espacio de estados, en la forma canónica observable, y con un tiempo de muestreo de 0.032 segundos, para cada conjunto de datos correspondientes a cada experimento. Los modelos obtuvieron una tasa de ajuste del 69.16 %, 78.64 % y 75.95 % respectivamente. Las matrices para cada uno de los modelos, son las siguientes:

La identificación da como resultado un modelo lineal que representa la dinámica de la planta para cada uno de estos rangos. El proceso es no lineal, por lo tanto, el modelo generará incertidumbre en el resultado final. Posteriormente se verá que esto no afecta al desempeño del controlador.

Modelo 1:

$$A = \begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ 0{,}8216 & -2{,}6206 & 2{,}7961 \end{bmatrix}$$

$$B = \begin{bmatrix} -0{,}0157 \\ -0{,}0075 \\ 0{,}0023 \end{bmatrix}$$

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

Modelo 2:

$$A = \begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ 0{,}8495 & -2{,}6775 & 2{,}8255 \end{bmatrix}$$

$$B = \begin{bmatrix} -0{,}0066 \\ -0{,}0011 \\ 0{,}0062 \end{bmatrix}$$

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

Modelo 3:

$$A = \begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ 0{,}8426 & -2{,}6648 & 2{,}8195 \end{bmatrix}$$

$$B = \begin{bmatrix} -0{,}0149 \\ -0{,}0085 \\ -0{,}0001 \end{bmatrix}$$

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

3.2.4. Validación de los modelos.

Para validar los modelos se utilizó el conjuntos de datos `validateData` adquirido para cada modelo. La validación consiste en inyectar la señal PRBS en el modelo identificado, y la salida producida se compara con la salida medida del proceso. MATLAB tiene una función que realiza ésta comparación:

```
1    %   comentarios intencionalmente sin acento
2    %   Validacion del modelo
3    compare(validateData, sys);
```

Ésta función crea una gráfica comparativa en la cual se puede observar la salida real del proceso (`validateData`) y la salida del modelo identificado (`sys`), incluye también un indicador que corresponde a la **raíz del error medio cuadrático normalizado (NRMSE)**, el cual es un valor adimensional y representa el porcentaje de ajuste que tiene el modelo identificado con respecto a la salida real, siendo el 100 % un ajuste perfecto. A continuación se muestran los resultados con cada uno de los modelos identificados.

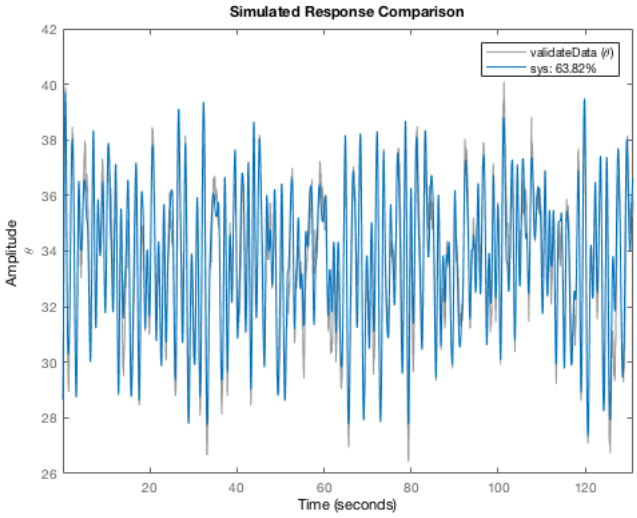


Figura 3.18: *Comparación de la respuesta del proceso a la señal PRBS 1, con los datos de validación. Respuesta real y simulada.*

La figura 3.18 muestra la comparación entre la salida real `validateData` y la salida simulada del modelo 1 `sys`, con un ajuste del 63.82 %.

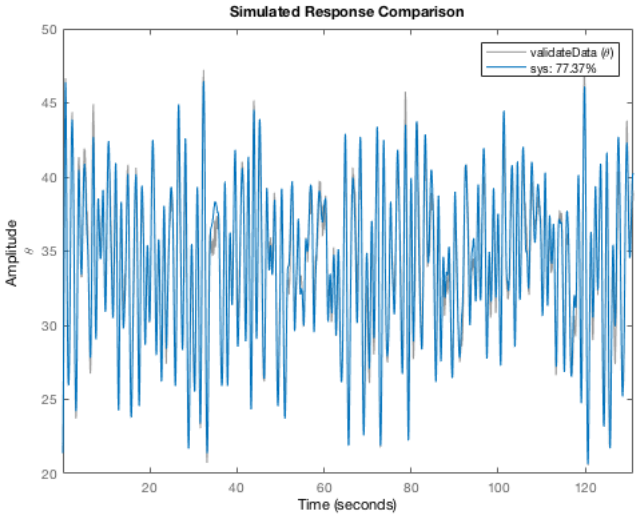


Figura 3.19: *Comparación de la respuesta del proceso a la señal PRBS 2, con los datos de validación. Respuesta real y simulada.*

La figura 3.19 muestra la comparación entre la salida real `validateData` y la salida simulada del modelo 2 `sys`, con un ajuste del 77.37 %.

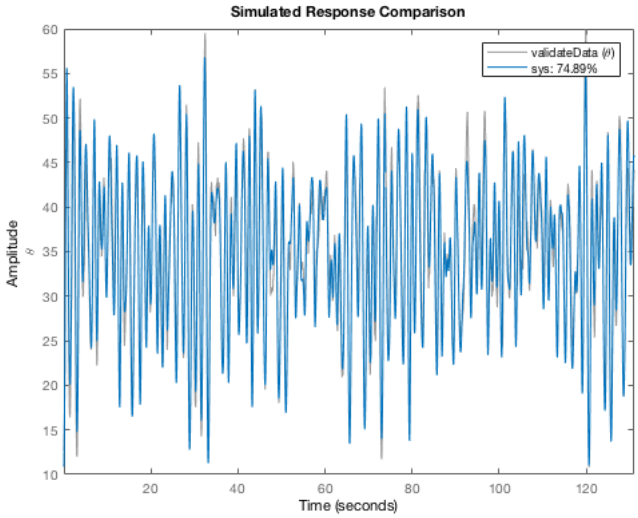


Figura 3.20: *Comparación de la respuesta del proceso a la señal PRBS 3, con los datos de validación. Respuesta real y simulada.*

La figura 3.20 muestra la comparación entre la salida real `validateData` y la salida simulada del modelo 3 `sys`, con un ajuste del 74.89 %.

Los resultados obtenidos con la función `compare` muestran que los modelos identificados tienen ajustes de 63.82 %, 77.37 % y 74.89 %, lo que significa que la estimación de la dinámica del proceso está por encima del 63 %, lo cual puede considerarse como bueno para los fines de éste proyecto. Se puede decir que el modelo 2 tiene el mejor ajuste dinámico con respecto a los otros 2 modelos.

Adicionalmente, se calculó la raíz del error medio cuadrático (RMSE - Root Mean Square Error) de la salida real del proceso y el modelo identificado en estado estable con un escalón de 33 %, aplicando la formula 3.6 donde y_i corresponde a la medición de la salida del proceso, y $\hat{y}_i$ corresponde a la salida del modelo identificado [7].

$$RMSE = \sqrt{\frac{\sum_{i=1}^{n} (y_i - \hat{y}_i)^2}{n}} \tag{3.6}$$

Este cálculo dio como resultado para el modelo 1 un RMSE=1.9042º, para el modelo 2 un RMSE=0.8881º y para el modelo 3 un RMSE=0.6192º. Lo que significa que en estado estable, el modelo 3 tiene un mejor ajuste con respecto a la salida real del proceso.

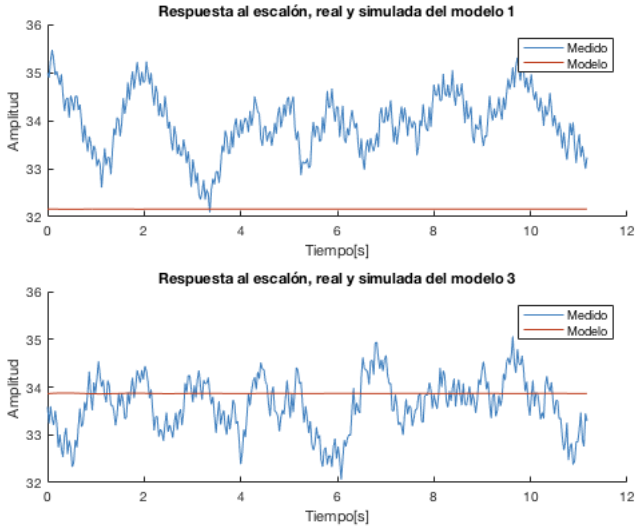


Figura 3.21: *Salida real del proceso y salida del modelo identificado en estado estable a lazo abierto, con un escalón del 33 %.*

En la figura 3.21 se puede observar que el modelo 1 presenta una mayor desviación de la salida real, a comparación del modelo 3 que tiene un mejor ajuste en estado estable.

Los resultados de validación indican que posiblemente el modelo 2 sea el más idóneo para la implementación, con un ajuste del 77.37 % dinámico, y un error medio de 0.8881º en estado estable. Sin embargo se verá posteriormente que el controlador es capaz de controlar el proceso con cualquiera de los 3 modelos.

Por último, como paso previo a la implementación, se comprueba la *controlabilidad* y la *observabilidad* de los modelos identificados, se utilizan las herramientas de Matlab `ctrb` y `obsv` respectivamente. Dando como resultado las siguientes matrices:

Modelo 1:

$$M = \begin{bmatrix} -0{,}0157 & -0{,}0075 & 0{,}0023 \\ -0{,}0075 & 0{,}0023 & 0{,}0133 \\ 0{,}0023 & 0{,}0133 & 0{,}0250 \end{bmatrix} \tag{3.7}$$

$$O = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{3.8}$$

Modelo 2:

$$M = \begin{bmatrix} -0{,}0066 & -0{,}0011 & 0{,}0062 \\ -0{,}0011 & 0{,}0062 & 0{,}0148 \\ 0{,}0062 & 0{,}0148 & 0{,}0244 \end{bmatrix} \tag{3.9}$$

$$O = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{3.10}$$

Modelo 3:

$$M = \begin{bmatrix} -0{,}0149 & -0{,}0085 & -0{,}0001 \\ -0{,}0085 & -0{,}0001 & 0{,}0098 \\ -0{,}0001 & 0{,}0098 & 0{,}0208 \end{bmatrix} \tag{3.11}$$

$$O = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{3.12}$$

Donde M es la matriz de controlabilidad, y su rango en todos los casos es de 3. Y O es la matriz de observabilidad y su rango para todos lo casos es 3. Por lo tanto los modelos identificados son controlables y observables.

3.3. Estructura e implementación del algoritmo de control.

3.3.1. Estructura.

El controlador está compuesto por un conjunto de tareas, las cuales deben ser realizadas de forma periódica, en un tiempo de muestreo establecido.

Estas tareas comprenden la lectura de la salida $y(k)$, la cual es ingresada al observador a la par que el $\Delta u(k)$. El observador estima los estados $\hat{x}(k)$ los cuales ingresan al controlador y con ésto se calcula el error con respecto a $r(k)$ y posteriormente el nuevo $\Delta u(k)$, el cual se adiciona al $u(k-1)$, formando el nuevo $u(k)$, que es enviado al proceso. La figura 3.22 muestra la estructura del lazo de control.

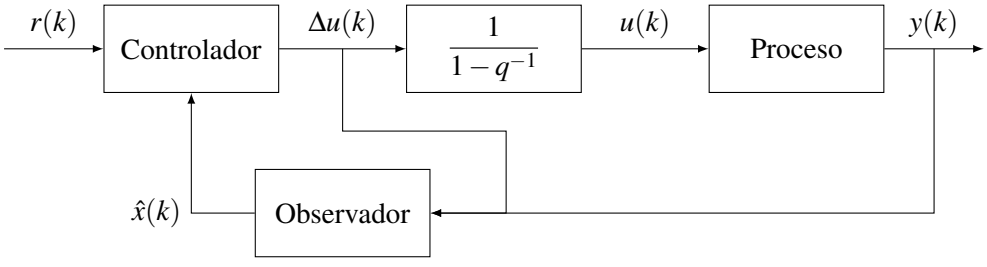


Figura 3.22: *Diagrama de bloques del proceso a lazo cerrado*

3.3.2. Algoritmo.

De acuerdo a la estructura del controlador, el algoritmo del bucle principal puede desglosarse en las siguientes tareas:

1. Leer el ángulo medido por el sensor IMU
2. Observar los estados del proceso
3. Calcular la función de costo J y el vector γ
4. Calcular el salto de la variable de control
5. Integrar la señal de control
6. Escribir la variable de control mediante una señal PWM

De todas las tareas, las más relevantes son la 3 y 4, ya que representan la técnica misma del controlador implementado. Adicional a éstas, también podría incluirse a la tarea 5 como parte del paso anterior. Estas 3 tareas conforman el bloque del controlador, como puede verse en la figura 3.22. El resto de tareas son comunes para las diferentes técnicas de control. El periodo de ejecución fue fijado en 0.032 segundos.

En el diagrama de flujo del controlador 3.23, se pueden observar las diferentes etapas del controlador. Existen operaciones que deben ejecutarse una sola vez al inicio del programa, y corresponden a la inicialización del sistema y a la inicialización del modelo.

La inicialización del sistema es la configuración del hardware del microcontrolador, cuyo procedimiento se describió anteriormente, en las configuraciones de la tarjeta de desarrollo.

La inicialización del modelo corresponde a las variables del controlador, hace mención al cálculo de las matrices del modelo aumentado, al cálculo de las matrices F, Φ, al cálculo de las matrices derivadas de éstas, $\Phi^T\Phi$, $\Phi^T\overline{R}_s$ y $\Phi^T F$, el cálculo del vector de ganancias del observador K_{ob} , etc.

El controlador está provisto de las funciones necesarias para poder realizar los cálculos de éstas variables de forma autónoma, prescindiendo del uso de otras herramientas.

Una vez que se ha completado la inicialización, el controlador ingresa al bucle de control, donde, en cada iteración se verifica el cumplimiento del tiempo de muestreo para dar paso a las tareas.

La tarea 1 realiza la lectura del sensor IMU, y entrega el resultado en una variable $y(k)$, éste valor representa el ángulo medido.

La tarea 2 es un observador de estados, el cuál recibe como parámetro el ángulo medido y el valor de ΔU que se calculó en la iteración anterior. Entrega como resultado el vector de estados estimados $\hat{x}(k)$.

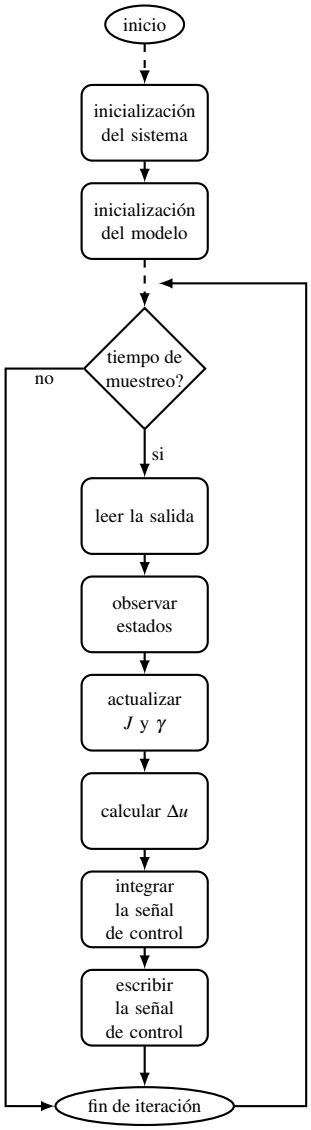


Figura 3.23: *Diagrama de flujo del algoritmo de control*

Las tareas 3 y 4 son el controlador como tal, aquí se realiza el cálculo de la función de costo y su minimización, lo que da como resultado el vector ΔU, posteriormente en la tarea 5 se hace la integración de la señal de control $u(k)$ utilizando sólo el primer valor del vector ΔU.

La tarea 6 escribe el valor de la señal de control $u(k)$ en el actuador del rotor. Completando el lazo de control.

Ya que el bloque del controlador es el más importante, se describirá más detalladamente su estructura. Como se mencionó anteriormente, aquí se calcula la función de costo J, ésto mediante la ecuación 1.42. Se tienen inicializadas como constantes a las matrices Hessiana, $\Phi^T \overline{R}_s$ y $\Phi^T F$, y como variables se tiene a la referencia $r(k)$ y el vector de estados estimados $\hat{x}(k)$.

A partir de éstas matrices, se calcula ΔU utilizando el método de programación cuadrática de Hildreth, el cual utiliza las ecuaciones 1.51 y 1.52 para minimizar la función de costo. El método de Hildreth devuelve el valor óptimo de ΔU de acuerdo a la ecuación 1.53.

En el diagrama 3.24, E es la matriz Hessiana, ΔU^0 es la solución global de la función de costo. Sí se cumple que ΔU^0 está dentro de las restricciones, entonces ΔU toma el valor de la solución global, caso contrario, se desplaza éste valor mediante un λ óptimo.

3.3.3. Implementación.

El código del algoritmo de control, está escrito en lenguaje C, y está compuesto por varias funciones de la librería GSL (GNU Scientific Library), realizando las respectivas modificaciones en el código para que pueda ser compilado para el microcontrolador.

La librería GSL contiene una gran variedad de funciones para realizar operaciones con vectores y matrices, éstas funciones pertenecen a un grupo llamado BLAS (Basic Linear Algebra Subprograms) que como su nombre indica, son funciones para álgebra lineal básica.

A continuación se describen las funciones más relevantes, que fueron utilizadas en la implementación. Ésta información fue sacada de la documentación de la librería GSL, la cual está disponible en línea [34].

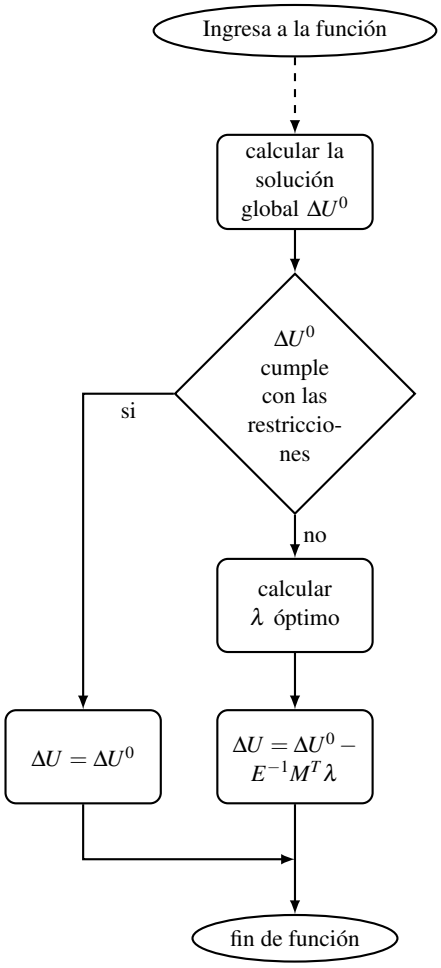


Figura 3.24: *Diagrama de flujo del algoritmo Hildreth*

Funciones de la librería GSL.

Funciones para suma y resta de matrices y vectores:

Función	gsl_matrix_add(gsl_matrix* a, const gsl_matrix* b)
Descripción	Realiza la suma de la matriz a + b, y guarda la respuesta en a

Función	gsl_vector_add(gsl_vector* a, const gsl_vector* b)
Descripción	Realiza la suma de los vectores a + b, y guarda la respuesta en a

Función	gsl_vector_sub(gsl_vector* a, const gsl_vector* b)
Descripción	Realiza la resta de los vectores a - b. y guarda la respuesta en a

Funciones para productos de matrices y vectores:

Función	gsl_blas_dgemm(CBLAS_TRANSPOSE_t TransA, CBLAS_TRANSPOSE_t TransB, double alpha, const gsl_matrix * A, const gsl_matrix * B, double beta, gsl_matrix * C)
Descripción	Realiza el producto matriz-matriz y acarrea la suma en $C = \alpha op(A) op(B) + \beta op C$, donde $op(A) = A, A^T, A^H$ para $TransA = CblasNoTrans, CblasTrans, CblasConjTrans$ y de igual manera para $TransB$

Función	gsl_blas_dgemv(CBLAS_TRANSPOSE_t TransA, double alpha, const gsl_matrix * A, const gsl_vector * x, double beta, gsl_vector * y)
Descripción	Realiza el producto matriz-vector y acarrea la suma $y = \alpha op(A)x + \beta y$, donde $op(A) = A, A^T, A^H$ para $TransA = CblasNoTrans, CblasTrans, CblasConjTrans$.

Función	gsl_blas_ddot(const gsl_vector * x, const gsl_vector * y, double * result)
Descripción	Realiza el producto escalar $x^T y$ para los vectores x y y, devuelve el resultado en $result$.

Funciones para calcular la matriz inversa:

Función	gsl_linalg_LU_decomp(gsl_matrix * A, gsl_permutation * p, int * signum)
Descripción	Factoriza a la matriz cuadrada A en la forma LU, $PA = LU$. Devuelve la descomposición en A, la cual tiene ahora los datos de U, y en p a la matriz de permutación P.

Función	gsl_linalg_LU_invert(const gsl_matrix * LU, const gsl_permutation * p, gsl_matrix * inverse)
Descripción	Calcula la inversa de la matriz A, la cual debe estar en la forma *LU* y su matriz de permutación. Devuelve en *inverse* el resultado.

Tipos de datos y parámetros: La librería GSL maneja sus propios tipos de datos, como se pudo ver en las funciones anteriores. Para la implementación de éste controlador, se utilizaron las siguientes:

Tipo	gsl_matrix
Descripción	Tipo de dato para matrices, por defecto utiliza variables de doble precisión.

Tipo	gsl_vector
Descripción	Tipo de dato para vectores, por defecto utiliza variables de doble precisión.

Algunas funciones también requieren que se les especifiquen parámetros de operación, por ejemplo, las funciones de producto. A continuación de describen los parámetros utilizados:

Parámetro	CblasNoTrans
Descripción	Indica que la matriz a la cual está vinculado el parámetro, se opera de forma normal, sin transposición.

Parámetro	CblasTrans
Descripción	Indica que la matriz a la cual está vinculado el parámetro, se transpone antes de realizar la operación.

En la guía en línea de la librería GSL se puede encontrar la información completa acerca de las funciones y tipos de datos, así como de sus inicializaciones.

Descripción del uso de las funciones en el algoritmo.

Implementar el algoritmo de control con las funciones antes mencionadas, no es una tarea complicada sí se tiene en claro qué operaciones se deben realizar.

Debido a que los recursos de un microcontrolador son limitados, comparados con una computadora, es fundamental reconocer la mayor cantidad de expresiones posibles que puedan ser calculadas como constantes, y dejar la menor cantidad posible de variables, de forma que se libere carga de procesamiento al microcontrolador. En las rutinas implementadas, se ha optimizado el código de forma que cumpla con éste requisito, por lo que la mayor carga de procesamiento se da en el arranque del controlador, donde se inicializan todas las variables necesarias para el funcionamiento, dejando como constantes a la mayor parte de matrices y vectores, que no requieren ser actualizados en cada iteración.

A continuación se exponen varios fragmentos del código que se utilizó para el cálculo de las matrices más importantes.

Código para calcular las matrices F, Φ y sus derivados:

```
1     gsl_vector_memcpy(CA, sys->C);
2     for (i = 1; i ≤ mpc->Np; i++)
3     {
4         gsl_blas_dgemv(CblasTrans, 1.0, sys->A, CA, 0.0, CAp);
5         for (j = 0; j < sys->C->size; j++) {
6             value = gsl_vector_get(CAp, j);
7             gsl_matrix_set(mpc->F, i - 1, j, value);
8         }
9     //  Para calcular Phi
10        gsl_blas_ddot(CAp, sys->B, &cab);
11        gsl_vector_set(CAB, i - 1, cab);
12        gsl_vector_memcpy(CA, CAp);
13    }
```

En base a la ecuación 1.37 se calcula la matriz F, en la primera línea se copia el vector $sys->C$ a un vector auxiliar CA. Luego dentro de un ciclo *for* de N_p iteraciones se realiza la operación $sys->A^TCA$, usando la función *gsl_blas_dgemv* cuyo resultado es el equivalente al primer producto $sys->C \times sys->A$ y éste resultado se guarda en CAp. Entonces se copia el vector CAp en la primera fila del vector $mpc->F$, ésta operación se realiza en el ciclo *for* interno. Para finalizar se actualiza el vector de CA, reemplazándolo por el valor de CAp y se continua iterando hasta completar las N_p filas de $mpc->F$, consiguiendo en las iteraciones siguientes, los vectores CA^2,CA^3,$\cdots$,CA^{Np}. También en ésta parte del código, se aprovecha ésta sucesión utilizándola para la formación de la matriz Φ, multiplicando cada uno de los vectores CA por el vector $sys->B$.

En base a la ecuación 1.38 se calcula la matriz Φ, considerando que el vector formado por los valores de CAB,CA^2B,$\cdots$,$CA^{Np-1}B$, ya fue calculado en el fragmento anterior. Entonces el vector Φ se forma con el siguiente código.

```
1   gsl_blas_ddot(sys->C, sys->B, &CB);
2   gsl_matrix_add_diagonal(mpc->Phi, CB);
3   for (j = 0; j < mpc->Nc; j++) {
4     for (i = 1; i < (mpc->Np - j); i++) {
5       value = gsl_vector_get(CAB, i - 1);
6       gsl_matrix_set(mpc->Phi, i + j, j, value);
7     }
8   }
```

Luego, es necesario calcular los vectores $\Phi^T\Phi$, $\Phi^T F$ y $\Phi^T\overline{R}_s$, para lo cual se ejecuta el siguiente fragmento de código.

```
1  gsl_blas_dgemm(CblasTrans, CblasNoTrans, 1.0, mpc->Phi, mpc->...
     Phi, 0.0, mpc->Phi_Phi);
2  gsl_blas_dgemm(CblasTrans, CblasNoTrans, 1.0, mpc->Phi, mpc->F...
     , 0.0, mpc->Phi_F);
3  gsl_blas_dgemv(CblasTrans, 1.0, mpc->Phi, mpc->Rs, 0.0, mpc->...
     Phi_Rs);
```

Donde $mpc->Phi$ es la variable que guarda la información de la matriz Φ, $mpc->F$ es la matriz F, $mpc->Rs$ es el vector de referencia, $mpc->Phi\_Phi$ es la variable donde se guarda el resultado de la operación $\Phi^T\Phi$, $mpc->Phi\_F$ guarda el resultado $\Phi^T F$, y por último la variable $mpc->Phi\_Rs$ donde ser guarda el resultado de $\Phi^T\overline{R}_s$.

Código para el observador: El observador está estructurado de acuerdo a la función de un observador normal, es decir, cumple con la ecuación:

$$\hat{x}(k+1) = A\hat{x}(k) + Bu(k) + K_e(y(k) - C\hat{x}(k)) \tag{3.13}$$

Donde, A, B, y C son las matrices del modelo aumentado, $\hat{x}$ es el vector de estados estimados, $u(k)$ es la variable de control aplicada al proceso, K_e es el vector de ganancias del observador que multiplica al error $(y(k) - C\hat{x})$.

Para calcular el vector de ganancias se utiliza el procedimiento de Acker para ubicación de polos, esto se realiza una vez al inicializar el sistema.

```
1  //  comentarios intencionalmente sin acento
2  //  inicializacion del observador
3  //  Variable para guardar A
4  gsl_matrix *ATrans = gsl_matrix_calloc(m, m); transpuesta
5  //  Realiza la transpuesta
6  gsl_matrix_transpose_memcpy(ATrans, sys->A);
7  //  Variable para guardar el vector de ganancias
8  obs->k = gsl_vector_calloc(m);
9  //  Calcula el vector de ganancias
10 acker(ATrans, sys->C, p, obs->k);
```

Entonces, la implementación en código de la ecuación 3.13, queda de la siguiente forma:

```
1  //  xe(k+1) = A*xe(k) + B*u(k) + Ke*(y(k) - C*xe(k))
2  //  ye=C*xe(k)
3  gsl_blas_ddot((const gsl_vector*) obs->sys->C, (const ...
     gsl_vector*) obs->xe, &obs->ye);
4  yc = y - obs->ye;
5  //  xc(k) = Ke*yc(k)
6  gsl_vector_memcpy(obs->xc, (const gsl_vector*) obs->k);
7  gsl_vector_scale(obs->xc, yc);
8  //  xe(k+1) = A*xe(k) + B*u(k) + xc(k)
9  gsl_vector_memcpy(obs->dxe, (const gsl_vector*) obs->sys->B);
10 gsl_blas_dgemv(CblasNoTrans, 1.0, (const gsl_matrix*) obs->sys...
     ->A, (const gsl_vector *) obs->xe, u, obs->dxe);
11 //  xe(k+1) + xc(k)
12 gsl_vector_add(obs->dxe, obs->xc);
13 //  Actualiza xe=dxe
14 gsl_vector_memcpy(obs->xe, (const gsl_vector*) obs->dxe);
```

Código para calcular la matriz Hessiana: Ahora, en éste fragmento de código se muestra como calcular la matriz inversa H^{-1}:

```
1  int s;
2  gsl_matrix *Hlu = gsl_matrix_alloc(m, n);
3  gsl_matrix *Hinv = gsl_matrix_alloc(m, n);
4
5  //  Copia la matriz H en Hlu, Hlu se factoriza en PH=LU
6  gsl_matrix_memcpy(Hlu, H);
7
8  gsl_permutation * p = gsl_permutation_alloc(x_size);
9
10 //  Descomponiendo Hlu en LU para operaciones posteriores
11 gsl_linalg_LU_decomp(Hlu, p, &s);
12 gsl_linalg_LU_invert(Hlu, p, Hinv);
```

Donde *Hinv* guarda el resultado de la matriz H^{-1}, éste es un ejemplo de como se calcula la inversa de la matriz Hessiana.

3.4. Aplicación del controlador y análisis de resultados.

Para aplicar el controlador al proceso, primero es necesario inicializar las variables de funcionamiento, que son matrices, vectores, entre otros. Estos parámetros se irán presentando conforme se avance en los pasos del algoritmo del controlador.

3.4.1. Inicialización y sintonización del controlador.

Ya que se realizarán las pruebas de desempeño con los 3 modelos identificados, se debe configurar al controlador con los 3 modelos en instancias distintas, por lo que se mostrarán los pasos para cada uno de estos.

Para todos y cada uno de los modelos, se utilizaron como parámetros de configuración los siguientes:

- Horizonte de control: $N_c = 4$.
- Horizonte de predicción: $N_p = 50$.
- Penalización al control: $r_w = 1000$.
- Restricción de control: $\Delta U^{max} = 10$.
- Restricción de control: $\Delta U^{min} = -10$.
- Restricción de control: $U^{max} = 60$.
- Restricción de control: $U^{min} = 0$.
- Polos del observador: [0,1 0,2 0,3 0,4]

Los valores del horizonte de control y del horizonte de predicción se determinaron en base a un procedimiento de sintonización para controladores de modelo predictivo propuesto por Rahul Shridhar y Douglas J. Cooper [32], donde se plantea aproximar la dinámica del proceso a un modelo de 1er orden con tiempo muerto (FOPDT ver ecuación 3.14). Luego aplicando la ecuación 3.15 se calcula el tiempo muerto discreto, necesario para calcular el horizonte de predicción con la ecuación 3.16. Posteriormente se plantea que el horizonte de control sea un valor de 1 a 6.

$$\frac{y(s)}{u(s)} = \frac{K_p e^{-\theta_p s}}{\tau_p s + 1} \tag{3.14}$$

$$k = \frac{\theta_p}{T_s} + 1 \tag{3.15}$$

$$N_p = \frac{5\tau_p}{T_s} + k \tag{3.16}$$

Para encontrar éste modelo de 1er orden, se utilizaron los mismos datos para identificación, obteniendo un modelo de 1er orden de la forma:

$$\frac{y(s)}{u(s)} = \frac{1{,}018e^{-0{,}256s}}{0{,}228s+1} \tag{3.17}$$

Donde $K_p = 1{,}018$, $\theta_p = 0{,}256$ y $\tau_p = 0{,}228$.

El valor de k necesario para calcular el horizonte de predicción, se obtiene utilizando el tiempo de muestreo $T_s = 0{,}032$, y da como resultado un valor igual 9. Entonces aplicando este valor a la ecuación 3.16 se obtiene un horizonte de control igual a 45. Sin embargo, se utilizó un valor de 50 para los experimentos.

Aunque el valor del horizonte de predicción sea alto y significativamente distante al horizonte de control, éste valor no afecta al rendimiento del microcontrolador, puesto que sirve únicamente para el cálculo de las matrices Φ, F y Rs. Las matrices del sistema que son derivadas de las anteriores, se calcularán como **constantes** y sus dimensiones dependen únicamente del horizonte de control.

Debido a que el controlador está implementando en un proceso con dinámica compleja, se ha ajustado el valor de r_w con un valor alto para que el control no sea agresivo, ésto fue determinado en base a experiencias.

A continuación se detalla la inicialización de las matrices y variables del proceso para cada uno de los modelos.

Modelo 1: Primero es necesario encontrar el modelo aumentado del proceso, es decir el modelo predictivo con integrador embebido, tal como se muestra en las ecuaciones 1.30 y 1.31.

$$A = \begin{bmatrix} 0 & 1{,}0000 & 0 & 0 \\ 0 & 0 & 1{,}0000 & 0 \\ 0{,}8216 & -2{,}6206 & 2{,}7961 & 0 \\ 0 & 1{,}0000 & 0 & 1{,}0000 \end{bmatrix}$$

$$B = \begin{bmatrix} -0{,}0157 \\ -0{,}0075 \\ 0{,}0023 \\ -0{,}0157 \end{bmatrix}$$

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

Se inicializan las matrices F, Φ y $\overline{R}_s$, aplicando los parámetros establecidos para el horizonte de control y horizonte de predicción, en las ecuaciones 1.37 y 1.38 y 1.39 respectivamente. Luego, con las matrices F, Φ y $\overline{R}_s$, se calcularon las matrices $\Phi^T\Phi$, $\Phi^T F$ y $\Phi^T\overline{R}_s$.

$$\Phi^T\Phi = \begin{bmatrix} 50{,}9672 & 50{,}6535 & 50{,}1782 & 49{,}5411 \\ 50{,}6535 & 50{,}5033 & 50{,}1901 & 49{,}7106 \\ 50{,}1782 & 50{,}1901 & 50{,}0405 & 49{,}7230 \\ 49{,}5411 & 49{,}7106 & 49{,}7230 & 49{,}5691 \end{bmatrix}$$

$$\Phi^T F = \begin{bmatrix} 13865 & -30474 & 17020 & 44 \\ 13847 & -30394 & 16947 & 43 \\ 13787 & -30222 & 16823 & 42 \\ 13684 & -29957 & 16648 & 42 \end{bmatrix}$$

$$\Phi^T \overline{R}_s = \begin{bmatrix} 43{,}8226 \\ 43{,}1415 \\ 42{,}4612 \\ 41{,}7746 \end{bmatrix}$$

Estas matrices sirven para formar la función de costo J tal como se expresa en la ecuación 1.42. La matriz Hessiana, como se indicó en el capitulo 1, es $\left(\Phi^T\Phi + \overline{R}\right)$ donde $\overline{R}$ es la matriz de pesos, que viene dada por $\overline{R} = I \times r_w$.

Como resultado se tiene que la matriz Hessiana es igual a:

$$Hessian = \begin{bmatrix} 1050{,}9672 & 50{,}6535 & 50{,}1782 & 49{,}5411 \\ 50{,}6535 & 1050{,}5033 & 50{,}1901 & 49{,}7106 \\ 50{,}1782 & 50{,}1901 & 1050{,}0405 & 49{,}7230 \\ 49{,}5411 & 49{,}7106 & 49{,}7230 & 1049{,}5691 \end{bmatrix}$$

Ya que las ecuaciones 1.51 y 1.52 utilizan como parámetro a la matriz Hessiana inversa, ésta se calcula como una constante.

$$Hessian^{-1} = \begin{bmatrix} 957{,}5 & -42{,}2 & -41{,}8 & -41{,}2 \\ -42{,}2 & 957{,}9 & -41{,}8 & -41{,}4 \\ -41{,}8 & -41{,}8 & 958{,}3 & -41{,}4 \\ -41{,}2 & -41{,}4 & -41{,}4 & 958{,}6 \end{bmatrix}$$

Luego se inicializa el observador, lo cual consiste en encontrar el vector de ganancias del observador, aplicando los polos establecidos. Esto da como resultado un vector de ganancias:

$$K_{ob} = \begin{bmatrix} 1{,}7990 & 2{,}7514 & 3{,}7580 & 2{,}7961 \end{bmatrix}$$

Las restricciones establecidas en éste lazo de control afectan a la señal de control U y a su tasa de cambio ΔU. La tasa de cambio de la señal de control, fue limitada para un valor de $\pm 10\,\%$, mientras que la señal de control fue limitada a un máximo de $60\,\%$ y un mínimo de $0\,\%$, como está expresado en las siguientes relaciones:

$$-10 \leq \Delta u(k) \leq 10$$

$$0 \leq u(k) \leq 60$$

Las restricciones se deben expresar en términos de la ecuación 1.54, lo que da como resultado la siguiente igualdad:

$$\begin{bmatrix} -1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ -1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 \end{bmatrix} \Delta U = \begin{bmatrix} 10 \\ 10 \\ 60 - u(k-1) \\ 0 + u(k-1) \end{bmatrix} \tag{3.18}$$

donde:

$$M = \begin{bmatrix} -1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ -1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 \end{bmatrix}$$

es constante, y

$$\gamma = \begin{bmatrix} 10 \\ 10 \\ 60 - u(k-1) \\ 0 + u(k-1) \end{bmatrix}$$

se actualiza en cada iteración. Las matrices de las restricciones son las mismas para los 3 modelos.

Modelo 2: Por analogía, para el modelo 2 se tienen las siguientes matrices:

$$A = \begin{bmatrix} 0 & 1{,}0000 & 0 & 0 \\ 0 & 0 & 1{,}0000 & 0 \\ 0{,}8495 & -2{,}6775 & 2{,}8255 & 0 \\ 0 & 1{,}0000 & 0 & 1{,}0000 \end{bmatrix}$$

$$B = \begin{bmatrix} -0{,}0066 \\ -0{,}0011 \\ 0{,}0062 \\ -0{,}0066 \end{bmatrix}$$

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

$$\Phi^T\Phi = \begin{bmatrix} 49{,}8060 & 49{,}5122 & 49{,}0621 & 48{,}4546 \\ 49{,}5122 & 49{,}3759 & 49{,}0821 & 48{,}6270 \\ 49{,}0621 & 49{,}0821 & 48{,}9459 & 48{,}6471 \\ 48{,}4546 & 48{,}6270 & 48{,}6471 & 48{,}5058 \end{bmatrix}$$

$$\Phi^T F = \begin{bmatrix} 16078 & -34739 & 19084 & 43 \\ 16064 & -34662 & 19009 & 43 \\ 16001 & -34477 & 18877 & 42 \\ 15887 & -34182 & 18685 & 41 \end{bmatrix}$$

$$\Phi^T \overline{R}_s = \begin{bmatrix} 43{,}3565 \\ 42{,}7007 \\ 42{,}0450 \\ 41{,}3815 \end{bmatrix}$$

$$Hessian = \begin{bmatrix} 1049{,}8 & 49{,}5 & 49{,}1 & 48{,}5 \\ 49{,}5 & 1049{,}4 & 49{,}1 & 48{,}6 \\ 49{,}1 & 49{,}1 & 1048{,}9 & 48{,}6 \\ 48{,}5 & 48{,}6 & 48{,}6 & 1048{,}5 \end{bmatrix}$$

$$Hessian^{-1} = \begin{bmatrix} 958{,}3 & -41{,}4 & -41{,}0 & -40{,}5 \\ -41{,}4 & 958{,}7 & -41{,}0 & -40{,}6 \\ -41{,}0 & -41{,}0 & 959{,}1 & -40{,}7 \\ -40{,}5 & -40{,}6 & -40{,}7 & 959{,}4 \end{bmatrix}$$

$$K_{ob} = \begin{bmatrix} 1{,}8284 & 2{,}8306 & 3{,}9096 & 2{,}8255 \end{bmatrix}$$

Modelo 3:

$$A = \begin{bmatrix} 0 & 1{,}0000 & 0 & 0 \\ 0 & 0 & 1{,}0000 & 0 \\ 0{,}8426 & -2{,}6648 & 2{,}8195 & 0 \\ 0 & 1{,}0000 & 0 & 1{,}0000 \end{bmatrix}$$

$$B = \begin{bmatrix} -0{,}0149 \\ -0{,}0085 \\ -0{,}0001 \\ -0{,}0149 \end{bmatrix}$$

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

$$\Phi^T \Phi = \begin{bmatrix} 54{,}8960 & 54{,}6258 & 54{,}1549 & 53{,}4828 \\ 54{,}6258 & 54{,}5555 & 54{,}2824 & 53{,}8031 \\ 54{,}1549 & 54{,}2824 & 54{,}2091 & 53{,}9277 \\ 53{,}4828 & 53{,}8031 & 53{,}9277 & 53{,}8458 \end{bmatrix}$$

$$\Phi^T F = \begin{bmatrix} 16302 & -35371 & 19473 & 45 \\ 16296 & -35304 & 19400 & 44 \\ 16234 & -35114 & 19259 & 44 \\ 16113 & -34797 & 19051 & 43 \end{bmatrix}$$

$$\Phi^T \overline{R}_s = \begin{bmatrix} 44{,}7195 \\ 44{,}1360 \\ 43{,}5475 \\ 42{,}9447 \end{bmatrix}$$

$$Hessian = \begin{bmatrix} 1054{,}9 & 54{,}6 & 54{,}2 & 53{,}5 \\ 54{,}6 & 1{,}0546 & 54{,}3 & 53{,}8 \\ 54{,}2 & 54{,}3 & 1054{,}2 & 53{,}9 \\ 53{,}5 & 53{,}8 & 53{,}9 & 1053{,}8 \end{bmatrix}$$

$$Hessian^{-1} = \begin{bmatrix} 954{,}8 & -44{,}9 & -44{,}5 & -43{,}9 \\ -44{,}9 & 955{,}1 & -44{,}6 & -44{,}2 \\ -44{,}5 & -44{,}6 & 955{,}4 & -44{,}4 \\ -43{,}9 & -44{,}2 & -44{,}4 & 955{,}7 \end{bmatrix}$$

$$K_{ob} = \begin{bmatrix} 1{,}8224 & 2{,}8155 & 3{,}8821 & 2{,}8195 \end{bmatrix}$$

3.4.2. Análisis de la respuesta en el punto de funcionamiento.

Anteriormente se definió que el proceso operaría en un punto, en el cual, el ángulo que forma el brazo con la horizontal es de 0º. Pero el sistema tiene su punto de equilibrio en un ángulo aproximado de -33º, razón por la cual se calibra al sistema con ese desplazamiento, por lo tanto, para que el brazo mantenga la posición horizontal, se debe establecer como punto de funcionamiento a un ángulo de 33º.

Se midió la respuesta del proceso en su punto de funcionamiento con los 3 modelos. Luego se hizo una comparación del error medio cuadrático para cada modelo con respecto a la referencia, para valorar el desempeño del controlador. Luego se analizó las señales de control para ver el comportamiento del controlador.

La señal de control está restringida para un valor máximo de $60\,\%$, pero debido a que la referencia no requiere una señal de control mayor a la restricción, el controlador aún es capaz de responder sin que la entrada llegue a saturarse. A su vez, la tasa de cambio de ΔU está restringida para saltos máximos de $\pm 10\,\%$, los cuales están suavizados por el valor de r_w alto.

Modelo 1:

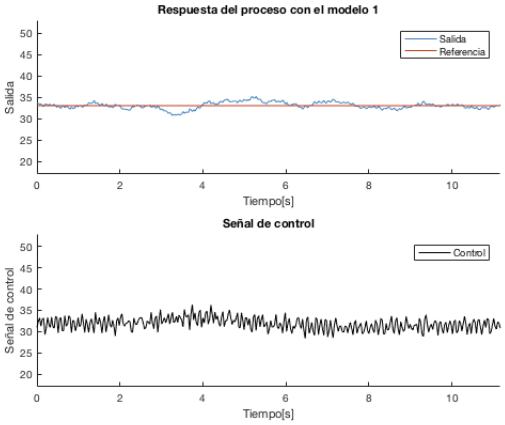


Figura 3.25: *Respuesta del proceso en su punto de funcionamiento con el modelo 1.*

En la figura 3.25 se puede observar la respuesta del proceso con el modelo 1, donde efectivamente la media de la salida es $\overline{y}$ =33.0294º, con un error medio cuadrático de 0.7944º, el rango máximo dentro del cual se desplazó la salida en ésta medición fue de $30.7076º \leq \theta \leq 35.1596º$.

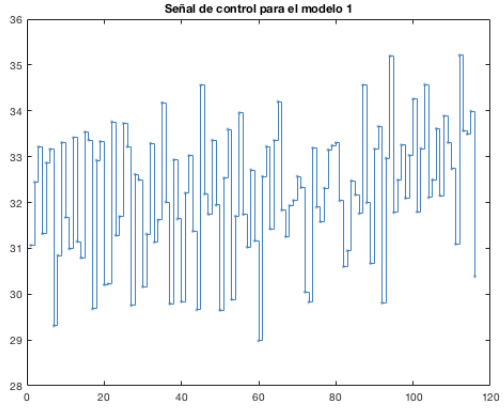


Figura 3.26: *Señal de control $u(k)$ para el modelo 1, mostrada en intervalos.*

En la figura 3.26 se puede observar un fragmento de la señal de control que sirve para mantener al proceso en el punto de funcionamiento con el modelo 1, la media de la señal completa tiene un valor de $\overline{u}$ =31.8374 %, y la señal se mueve dentro de un rango de $28.3956\,\% \leq u \leq 36.3348\,\%$.

Modelo 2:

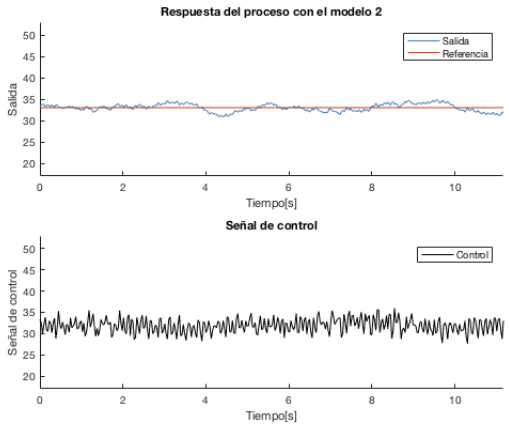


Figura 3.27: *Respuesta del proceso en su punto de funcionamiento con el modelo 2.*

En la figura 3.27 se puede observar la respuesta del proceso con el modelo 2, donde la media de la salida es $\overline{y}$ =32.9961º, con un error medio cuadrático de 0.8818º, el rango máximo dentro del cual se desplazó la salida en ésta medición fue de 30.8435º $\leq \theta \leq$ 34.8382º.

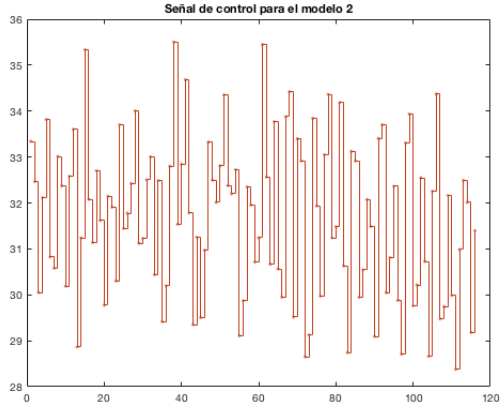


Figura 3.28: *Señal de control $u(k)$ para el modelo 2, mostrada en intervalos.*

En la figura 3.28 se puede observar un fragmento de la señal de control que sirve para mantener al proceso en el punto de funcionamiento con el modelo 2, la media de la señal completa tiene un valor de $\overline{u}$ =31.8700 %, y la señal se mueve dentro de un rango de 27.6421 % $\leq u \leq$ 35.9993 %.

Modelo 3:

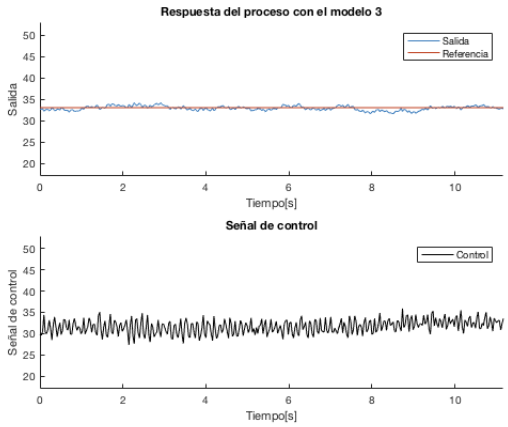


Figura 3.29: *Respuesta del proceso en su punto de funcionamiento con el modelo 3.*

En la figura 3.29 se puede observar la respuesta del proceso con el modelo 3, donde la media de la salida es $\overline{y}$ =32.8697º, con un error medio cuadrático de 0.5389º, el rango máximo dentro del cual se desplazó la salida en ésta medición fue de 31.5408º $\leq \theta \leq$ 34.1354º.

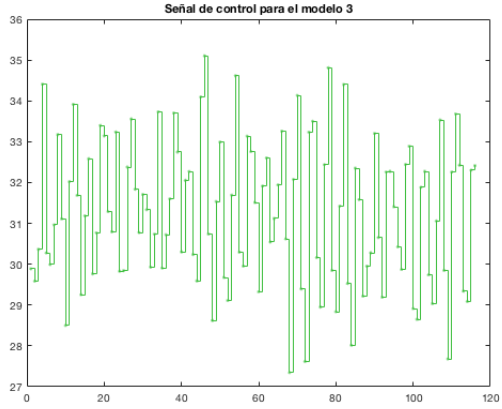


Figura 3.30: *Señal de control $u(k)$ para el modelo 3, mostrada en intervalos.*

En la figura 3.30 se puede observar un fragmento de la señal de control que sirve para mantener al proceso en el punto de funcionamiento con el modelo 3, la media de la señal completa tiene un valor de $\overline{u}$ =31.6411 %, y la señal se mueve dentro de un rango de 27.3454 % $\leq u \leq$ 35.8949 %.

	Modelo 1	Modelo 2	Modelo 3
Media	33.0294	32.9961	32.8697
RMSE	0.7944	0.8818	0.5389
Máximo	35.1596	34.8382	34.1354
Mínimo	30.7076	30.8435	31.5408
Desviación típica	0.7950	0.8830	0.5236

Cuadro 3.1: *Comparación de la respuesta de los 3 modelos en el punto de funcionamiento, salida y.*

Al observar los resultados de la tabla 3.1 se puede evidenciar que el modelo 2 presenta el mejor ajuste con respecto al punto de funcionamiento, aunque la respuesta de éste modelo es la más variante. Seguido por el modelo 1, cuyo ajuste es muy bueno y su desviación es aceptable. Por último es modelo 3 presenta un ajuste inferior a los modelos anteriores, pero su desviación es la más aceptable.

	Modelo 1	Modelo 2	Modelo 3
Media	31.8374	31.8700	31.6411
Máximo	36.3348	35.9993	35.8949
Mínimo	28.3956	27.6421	27.3454
Desviación típica	1.5083	1.7169	1.6786

Cuadro 3.2: *Comparación de las 3 señales de control en el punto de funcionamiento, entrada u.*

En la tabla 3.2 se puede observar que la señal de control con menor variabilidad es la del modelo 1, se podría decir que ésta señal es la menos exigente para el actuador.

Tiempo de optimización:

En éste punto del análisis, se midió el tiempo que le tomó al microcontrolador en procesar el algoritmo de optimización. En la figura 3.31 se puede ver un fragmento de las mediciones del tiempo que le toma al microcontrolador en resolver el algoritmo. Puede verse que el tiempo está distribuido en escalones, donde cada escalón representa una iteración del algoritmo. En general al microcontrolador le toma 0.225 ms en resolver el problema de optimización. El tiempo mínimo es de 0.222 ms, que posiblemente sea el tiempo que le toma al microcontrolador en encontrar la solución global. El tiempo más alto es de 0.232 ms, donde evidentemente el microcontrolador ejecuta el algoritmo de optimización iterativo.

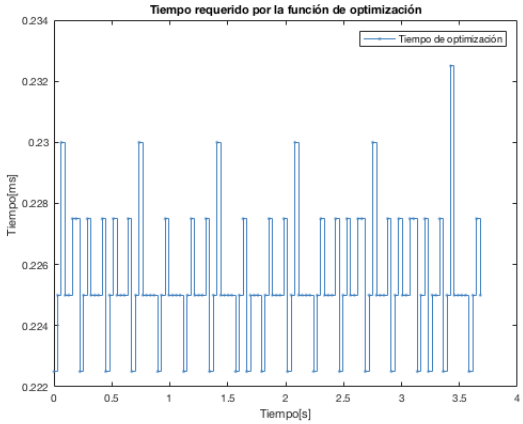


Figura 3.31: *Medición del tiempo de minimización.*

3.4.3. Análisis de la respuesta al cambio del punto de funcionamiento.

Se analizó la respuesta del proceso al cambio de su punto de funcionamiento. Se realizó en total 9 pruebas, en las cuales se cambió la referencia en saltos de ± 5º, ± 15º y ± 20º con respecto al punto central $r = 33$º. En todas las pruebas se aplicó una señal de control que cambia desde su punto central hasta un valor máximo, luego baja al mínimo y regresa al punto central.

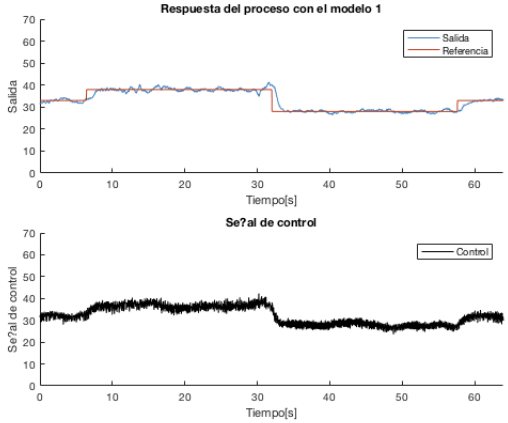


Figura 3.32: *Respuesta del proceso con el modelo 1 a los cambios de referencia, con saltos de ± 5º, salida y.*

En la figura 3.32 se muestra la respuesta del proceso cuando se aplica un cambio de referencia de ± 5. La forma de señal de control aplicada es similar en todos los 9 expe-

rimentos, cambiando únicamente la amplitud de los saltos. A continuación se muestran los resultados de las mediciones de los 9 experimentos realizados:

Saltos de $\pm 5^{o}$

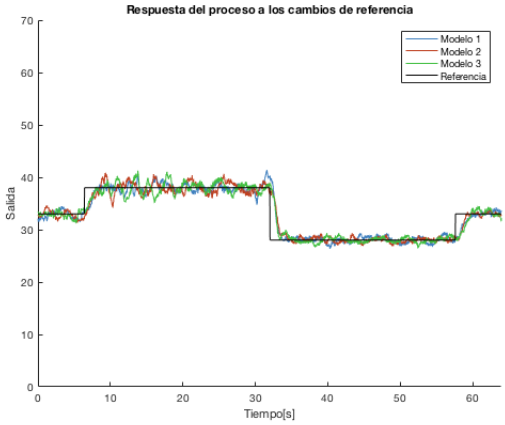


Figura 3.33: *Respuesta del proceso con los 3 modelos a los cambios de referencia, con saltos de $\pm 5^{o}$, salida y.*

En la figura 3.33 se muestra las mediciones de la salida con los 3 modelos, al aplicar los cambios de referencia de $\pm 5^{o}$ en cada caso.

	Referencia	Modelo 1	Modelo 2	Modelo 3
Media	33	33.0009	33.0103	33.0000
RMSE	0	1.5388	1.5063	1.5170
Máximo	38	41.3855	40.8699	41.2829
Mínimo	28	26.3771	26.8674	26.4520
Desviación típica	4.4733	4.4669	4.4692	4.5001

Cuadro 3.3: *Comparación de la respuesta de los 3 modelos cambiando su punto de funcionamiento, en saltos de $\pm 5^{o}$, salida y.*

En la tabla 3.3 se puede observar que el modelo 2 presenta el menor error cuadrático con respecto a la referencia y su desviación a comparación con la desviación de la referencia, es la más cercana. Sus máximos y mínimos son los más cercanos a la referencia. El modelo 2 tiene los mejores resultados.

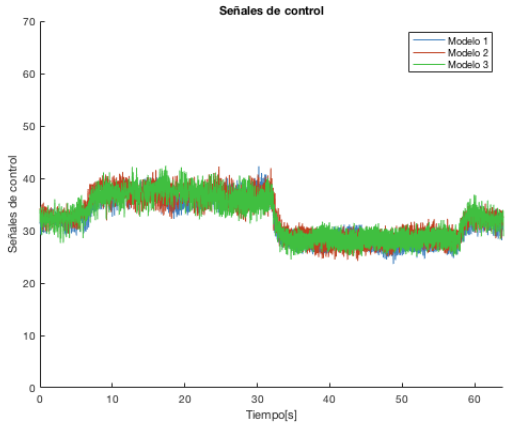


Figura 3.34: *Señales de control producidas con los cambios de referencia, con saltos de ±5º, entrada u.*

En la figura 3.34 se puede observar las señales de control producidas por los cambios de referencia de ±5º.

	Modelo 1	Modelo 2	Modelo 3
Media	32.1661	32.8042	32.6015
Máximo	42.2844	42.2436	42.3847
Mínimo	23.6475	24.2449	24.4847
Desviación típica	4.2058	4.2630	4.1519

Cuadro 3.4: *Comparación de las 3 señales de control en el cambio del punto de funcionamiento, en saltos de ±5º, entrada u.*

En la tabla 3.4 se puede ver que la señal de control del modelo 2 presenta la mayor dispersión.

Saltos de $\pm$15º

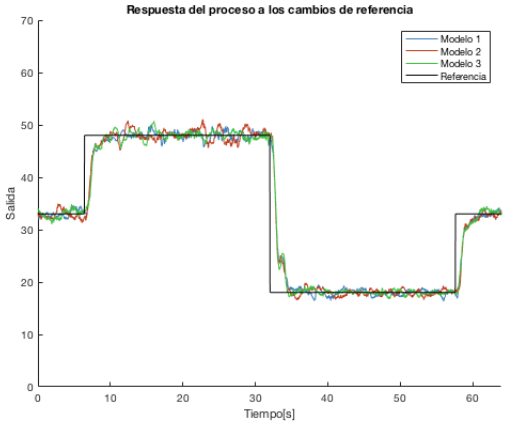


Figura 3.35: *Respuesta del proceso con los 3 modelos a los cambios de referencia, con saltos de $\pm$15º, salida y.*

En la figura 3.35 se muestra las mediciones de la salida con los 3 modelos, al aplicar los cambios de referencia de $\pm$15º en cada caso.

	Referencia	Modelo 1	Modelo 2	Modelo 3
Media	33	32.9919	32.9830	33.0097
RMSE	0	3.9510	4.0076	4.0105
Máximo	48	49.9632	51.0529	50.7290
Mínimo	18	16.3734	16.5813	16.7660
Desviación típica	13.4198	13.1409	13.2529	13.1702

Cuadro 3.5: *Comparación de la respuesta de los 3 modelos cambiando su punto de funcionamiento, en saltos de $\pm$15º, salida y.*

En la tabla 3.5 se puede observar que el modelo 1 tiene el menor error medio cuadrático, aunque su desviación comparada con la desviación de la referencia no es la más cercana, sino la del modelo 2. Los valores máximo y mínimo del modelo 3 son los mas cercanos a la referencia. En general podría decirse que los 3 modelos responden de forma similar.

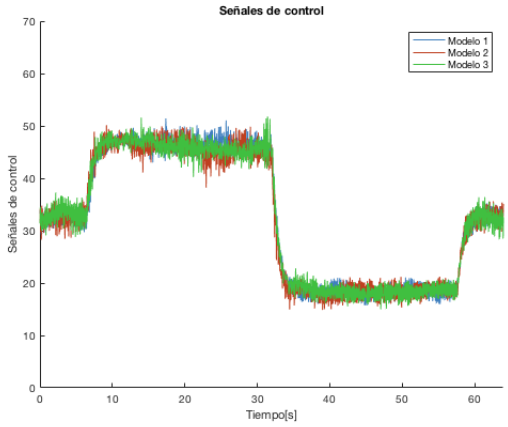


Figura 3.36: *Señales de control producidas con los cambios de referencia, con saltos de ±15º, entrada u.*

En la figura 3.36 se puede observar las señales de control producidas por los cambios de referencia de ±15º.

	Modelo 1	Modelo 2	Modelo 3
Media	32.5702	32.2407	32.5037
Máximo	51.4282	50.1443	51.7685
Mínimo	15.7219	14.8864	15.0002
Desviación típica	12.3229	12.1756	12.2116

Cuadro 3.6: *Comparación de las 3 señales de control en el cambio del punto de funcionamiento, en saltos de ±15º, entrada u.*

En la tabla 3.6 se puede observar que la señal con la menor desviación es la del modelo 2, aunque presenta una pequeña diferencia con respecto a las señales de los otros modelos.

Saltos de $\pm$20º

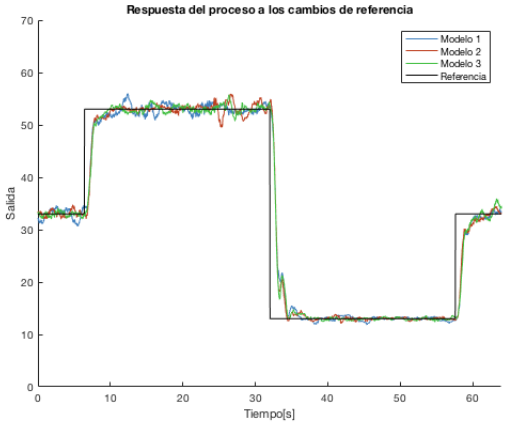


Figura 3.37: *Respuesta del proceso con los 3 modelos a los cambios de referencia, con saltos de $\pm$20º, salida y.*

En la figura 3.37 se muestra las mediciones de la salida con los 3 modelos, al aplicar los cambios de referencia de $\pm$20º en cada caso.

	Referencia	Modelo 1	Modelo 2	Modelo 3
Media	33	32.9983	33.0176	33.0068
RMSE	0	5.1521	5.2608	5.1627
Máximo	53	55.9765	55.9208	55.7517
Mínimo	13	11.8174	12.0317	12.3554
Desviación típica	17.8930	17.5569	17.6163	17.6016

Cuadro 3.7: *Comparación de la respuesta de los 3 modelos cambiando su punto de funcionamiento, en saltos de $\pm$20º, salida y.*

En la tabla 3.7 se puede observar que el modelo 1 tiene el menor error medio cuadrático, aunque la desviación que mas se aproxima a la desviación de la referencia es la del modelo 2. Los valores máximo y mínimo mas cercanos a la referencia son los del modelo 3. Se puede decir que los 3 modelos responde de forma similar.

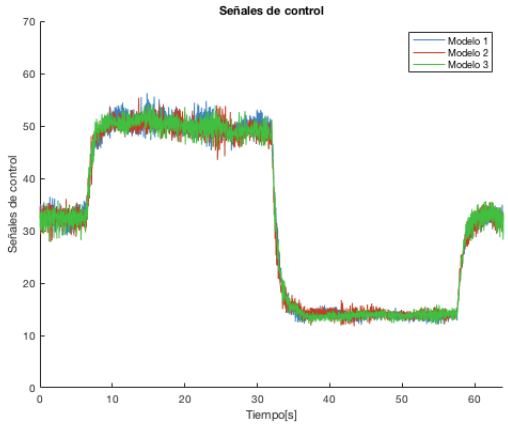


Figura 3.38: *Señales de control producidas con los cambios de referencia, con saltos de ±20º, entrada u.*

En la figura 3.38 se puede observar las señales de control producidas por los cambios de referencia de ±20º.

	Modelo 1	Modelo 2	Modelo 3
Media	32.1873	32.0207	32.1140
Máximo	56.2461	55.3495	54.0685
Mínimo	12.0050	11.7605	12.0175
Desviación típica	16.0021	15.6389	15.8577

Cuadro 3.8: *Comparación de las 3 señales de control en el cambio del punto de funcionamiento, en saltos de ±20º, entrada u.*

En la tabla 3.8 se puede observar también que la señal de control que tiene menor desviación es la del modelo 2.

Se puede concluir con estos experimentos, que los 3 modelos responden prácticamente igual a los cambios de referencia.

3.4.4. Análisis de la respuesta a las perturbaciones.

Se midió la respuesta real del proceso a las perturbaciones con los 3 modelos. Para este experimento se utilizó un peso de 75 gramos, el cual fue anclado al brazo mecánico hasta que el proceso alcance la referencia, y posteriormente el peso fue liberado.

	Modelo 1	Modelo 2	Modelo 3
Media	33.2558	33.6508	32.3034
RMSE	4.6337	6.6173	6.7085
Máximo	53.5596	65.1656	58.8025
Mínimo	15.7563	13.0649	10.2156
Desviación típica	4.6299	6.5906	6.6780

Cuadro 3.9: *Comparación de la respuesta de los 3 modelos a una perturbación, salida y.*

En la tabla 3.9 puede verse que el modelo 1 responde mejor a las perturbaciones, puesto que no sufre mucho sobre disparo, también puede verse que el error medio cuadrático es menor que los 2 modelos restantes, al igual que la desviación típica.

	Modelo 1	Modelo 2	Modelo 3
Media	39.9870	40.4610	39.8902
Máximo	56.4277	56.7068	53.2133
Mínimo	21.1446	28.0567	23.9206
Desviación típica	8.3736	7.8879	7.3165

Cuadro 3.10: *Comparación de las 3 señales de control producidas por las perturbaciones, entrada u.*

En la tabla 3.10 se puede observar que si bien el modelo 1 es el que mejor responde a las perturbaciones, también exige más al actuador, puesto que la señal de control de la misma tiene la más alta desviación.

Modelo 1:

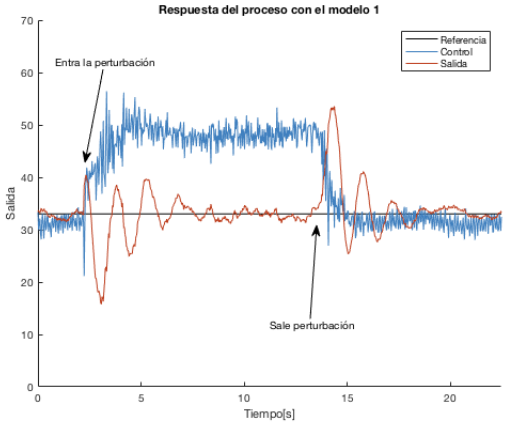


Figura 3.39: *Respuesta del proceso a una perturbación, señal de control y salida medida para el modelo 1.*

En la figura 3.39 se puede observar el ingreso de la perturbación aproximadamente en el segundo 2.5, inmediatamente el controlador empieza a aumentar la señal de control, se recupera la referencia en el segundo 7, luego en el segundo 14 se libera la carga y el controlador disminuye la señal de control, se recupera la referencia en el segundo 18.

Modelo 2:

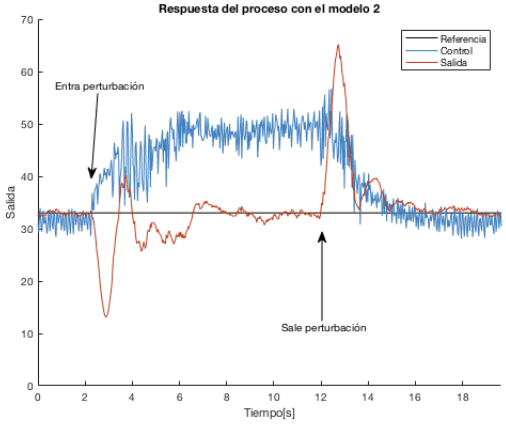


Figura 3.40: *Respuesta del proceso a una perturbación, señal de control y salida medida para el modelo 2.*

En la figura 3.40 se puede observar el ingreso de la perturbación aproximadamente en

el segundo 2, inmediatamente el controlador empieza a aumentar la señal de control, se recupera la referencia en el segundo 7, luego en el segundo 12 se libera la carga y el controlador disminuye la señal de control, se recupera la referencia en el segundo 15.

Modelo 3:

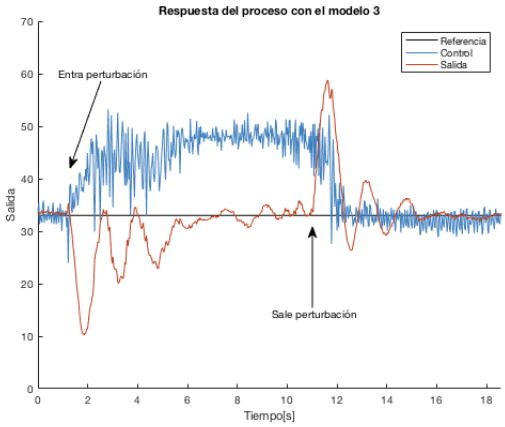


Figura 3.41: *Respuesta del proceso a una perturbación, señal de control y salida medida para el modelo 3.*

En la figura 3.41 se puede observar el ingreso de la perturbación aproximadamente en el segundo 1, inmediatamente el controlador empieza a aumentar la señal de control, se recupera la referencia en el segundo 6, luego en el segundo 11 se libera la carga y el controlador disminuye la señal de control, se recupera la referencia en el segundo 15.

Se puede concluir que el controlador es capaz de controlar el proceso con los 3 modelos, sin importar el rango dinámico de cada uno de ellos. La dinámica del proceso por ende puede ser representada por modelos lineales.

Capítulo 4

CONCLUSIONES Y RECOMENDACIONES

Los resultados obtenidos en el análisis anterior, indican el buen desempeño del controlador implementado.

Se puede concluir que, es totalmente factible implementar técnicas de control moderno sobre un microcontrolador de gama alta, y se pueden obtener buenos resultados en aplicaciones con procesos de dinámica rápida y compleja, como el expuesto en éste proyecto.

Este controlador utiliza un observador normal, por lo que se propone también realizar experimentos con un observador Kalman, el cual teóricamente es la mejor opción en la implementación de controladores de modelo predictivo.

Existe una gran variedad de microcontroladores que disponen de unidad de procesamiento para variables de punto flotante, así como microprocesadores, los cuales se pueden encontrar en plataformas de desarrollo conocidas como SBC (Single Board Computer). Ya que el algoritmo de control está escrito en lenguaje C, y utiliza una librería de código libre y abierto, su implementación en otros dispositivos para control embebido es totalmente factible y puede resultar interesante analizar estos resultados.

Podría también analizarse otras técnicas de predicción, como las basadas en la respuesta finita al escalón. También se pueden analizar otros algoritmos de optimización, implementar el código y medir su desempeño aplicando ésta u otras técnicas de control predictivo.

El controlador respondió bastante bien a los 3 modelos planteados para probar su desempeño, se puede decir que este tipo de controladores son muy robustos frente a la incertidumbre producida por los modelos con dinámicas tanto de rango amplio como de rango

reducido.

Bibliografía

[1] ABB. cpmPlus Expert Optimizer, Advanced optimization for your industry. https://library.e.abb.com/public/a3004e27e4286e52c12576be0038ce06/cpmPlus_Exp_Opt_3BHS%20291766_lr.pdf. [En línea; Accedido: 2017-09-10].

[2] ABB. Process improvement packages, multivariable control for the process industries – 3dmpc. https://library.e.abb.com/public/b788e3a9870e9a1ec1256a34007129d6/3BUS025043R0001_-_en_3dMPC_Multivariable_Controller.pdf. [En línea; Accedido: 2017-09-10].

[3] Amira Kheriji Abbes, Faouzi Bouania, and Mekki Ksouri. A microcontroller implementation of constrained model predictive control. *International Journal of Electrical, Computer, Energetic, Electronic and Communication Engineering*, Vol:5(No:5), 2011.

[4] D. Ariens, B. Houska, and H.J. Ferreau. Acado for matlab user's manual. http://www.acadotoolkit.org, 2010–2011.

[5] Rockwell Automation. Model predictive control, rockwell automation model predictive control delivers results. http://literature.rockwellautomation.com/idc/groups/literature/documents/br/rsbrp8-br001_-en-p.pdf. [En línea; Accedido: 2017-09-10].

[6] Leonidas G. Bleris, Jesus Garcia, Mayuresh V. Kothare, and Mark G. Arnold. Towards embedded model predictive control for system-on-a-chip applications. *Journal of Process Control*, pages 255–264, 2006.

[7] S.D.M. Borjas and C. Garcia. Subspace identification for industrial processes. *Uma Publicação da Sociedade Brasileira de Matemática Aplicada e Computacional.*, Vol:12(N.º3):183–194, 2011.

[8] E.F. Camacho and C. Bordons. *Model Predictive Control*. Springer, 2007.

[9] Marco A. Carpio. Evaluación de desempeño de los controladores digitales pid y predictivo tolerante a fallas, aplicados al control de nivel del líquido en un tanque. Master's thesis, Universidad Politécnica Salesiana, 2013.

[10] Industrial Information & Control Centre. Advanced Process Control Solutions, Model Predictive Control & Controller Tuning. http://www.i2c2.aut.ac.nz/PHP/getDownload.php?File=APC_Flyer.pdf&Type=Flyer. [En línea; Accedido: 2017-09-11].

[11] TE Conectivity. Ms5611 barometric pressure sensor, with stainless steel cap. http://www.te.com/commerce/DocumentDelivery/DDEController?Action=srchrtrv&DocNm=MS5611-01BA03&DocType=Data+Sheet&DocLang=English. [En línea; Accedido: 2017-10-01].

[12] Mitchell Thornton David Russell. *Introduction to Embedded Systems: Using ANSI C and the Arduino Development Environment*. Synthesis Lectures on Digital Circuits and Systems. Morgan and Claypool Publishers, 1 edition, 2010.

[13] Digilent. Wi-fire board reference manual. https://reference.digilentinc.com/_media/reference/microprocessor/wi-fire/wi-fire_rm_revd.pdf. [En línea; Accedido: 2017-10-01].

[14] L. Z. Eng. *Qt5 C++ GUI Programming Cookbook*. Packt Publishing, 2016.

[15] H.J. Ferreau, T. Kraus, M. Vukov, W. Saeys, and M. Diehl. High-speed moving horizon estimation based on automatic code generation. In *Proceedings of the 51th IEEE Conference on Decision and Control (CDC 2012)*, 2012.

[16] D. Gaydou, J. Redolfi, and A. Henze. Filtro complementario para estimación de actitud aplicado al controlador embebido de un cuadrimotor. *Proceedings of the Argentine Conference on Embedded Systems*, 2011.

[17] K. S. Holkar and L. M. Waghmare. An overview of model predictive control. *International Journal of Control and Automation*, Vol:3(N.º4), 2010.

[18] Honeywell. 3-axis digital compass ic hmc5883l. https://cdn-shop.adafruit.com/datasheets/HMC5883L_3-Axis_Digital_Compass_IC.pdf. [En línea; Accedido: 2017-10-01].

[19] Honeywerll. Product information note, profit controller. https://www.honeywellprocess.com/library/marketing/brochures/PIN-Profit-Controller-R440.pdf. [En línea; Accedido: 2017-09-10].

[20] B. Houska, H.J. Ferreau, and M. Diehl. ACADO Toolkit – An Open Source Framework for Automatic Control and Dynamic Optimization. *Optimal Control Applications and Methods*, 32(3):298–312, 2011.

[21] B. Houska, H.J. Ferreau, M. Vukov, and R. Quirynen. ACADO Toolkit User's Manual. http://www.acadotoolkit.org, 2009–2013.

[22] Microchip Technology Inc. 32-bit microcontroller families microchip. `http://ww1.microchip.com/downloads/en/DeviceDoc/30009904S.pdf`. [En línea; Accedido: 2017-10-10].

[23] Microchip Technology Inc. Architecture - 32-bit pic microcontrollers | microchip technology inc. `http://www.microchip.com/design-centers/32-bit/architecture/pic32mz-family`, 2017. [En línea; Accedido: 2017-10-11].

[24] Feedback Instruments. Engineering teaching solutions, twin rotor mimo. `http://www.feedback-instruments.com/pdf/brochures/33-007-PCI_datasheet_TwinRotorMIMO_MATLAB_10_2013.pdf`. [En línea; Accedido: 2017-09-24].

[25] InvenSense. Mpu-6000 and mpu-6050 product specification revision 3.4. `https://store.invensense.com/datasheets/invensense/MPU-6050_DataSheet_V34.pdf`. [En línea; Accedido: 2017-10-01].

[26] D. K. M. Kufoalor, L. Imsland, and T. A. Johansen. High-performance embedded model predictive control using step response models. *IFAC-PapersOnline*, 2015.

[27] M. Margolis. *Arduino Cookbook*. O'Reilly, 2011.

[28] Daniel Limón Marruedo. *Control predictivo de sistemas no lineales con restricciones: estabilidad y robustez*. PhD thesis, Escuela Superior de Ingenieros de la Universidad de Sevilla, 2002.

[29] S. Monk. *Raspberry Pi Cookbook*. O'Reilly, 2016.

[30] Ashwin Pajankar. *Raspberry Pi Supercomputing and Scientific Programming. MPI4PY, NumPy, and SciPy for Enthusiasts*. Apress, 2017.

[31] Freescale Semiconductor. Tilt sensing using a three-axis accelerometer. `https://www.nxp.com/docs/en/application-note/AN3461.pdf`. [En línea; Accedido: 2017-10-01].

[32] R. Shridhar and D. Cooper. A tuning strategy for unconstrained siso model predictive control. *Ind. Eng. Chem. Res.*, 1997.

[33] STMicroelectronics. Discovery kit with stm32f769ni mcu. `http://www.st.com/content/ccc/resource/technical/document/data_brief/group0/32/09/89/e4/4f/1d/48/3b/DM00276576/files/DM00276576.pdf/jcr:content/translations/en.DM00276576.pdf`. [En línea; Accedido: 2017-10-11].

[34] The GSL Team. GNU Scientific Library - GSL 2.4 Documentation. `https://www.gnu.org/software/gsl/doc/html/index.html`. [En línea; Accedido: 2017-08-02].

[35] Yi Zhu Tianhong Pan. *Designing Embedded Systems with Arduino*. Springer, 2017.

[36] M. Vukov, A. Domahidi, H. J. Ferreau, M. Morari, and M. Diehl. Auto-generated Algorithms for Nonlinear Model Predicitive Control on Long and on Short Horizons. In *Proceedings of the 52nd Conference on Decision and Control (CDC)*, 2013.

[37] M. Vukov, W. Van Loock, B. Houska, H.J. Ferreau, J. Swevers, and M. Diehl. Experimental Validation of Nonlinear MPC on an Overhead Crane using Automatic Code Generation. In *The 2012 American Control Conference, Montreal, Canada.*, 2012.

[38] Liuping Wang. *Model Predictive Control System Design and Implementation Using MATLAB*. Springer, 2009.

[39] Julio C. Zambrano. Plataforma basada en MATLAB®para entrenamiento en la sintonía de controladores predictivos. Master's thesis, Escuela Superior Politécnica del Litoral, 2014.

[40] Julio C. Zambrano and Ana I. González. Implementación de un algoritmo de control predictivo en espacio de estados sobre una plataforma de simulación desarrollada en MATLAB. *INGENIUS*, Vol:9(N.º9):5–14, 2013.

[41] P. Zometa, M. Kögel, and R. Findeisen. muAO-MPC: A free code generation tool for embedded real-time linear model predictive control. In *Proc. American Control Conference (ACC), 2013*, pages 5340–5345, Washington D.C., USA, 2013.

[42] Pablo Zometa, Markus Kögel, Timm Faulwasser, and Rolf Findeisen. Implementation aspects of model predictive control for embedded systems. *American Control Conference*, 2012.

UNIVERSIDAD POLITÉCNICA SALESIANA

UNIDAD DE POSGRADOS

MAESTRÍA EN CONTROL Y AUTOMATIZACIÓN INDUSTRIALES

Autor: Ing. Galo Guzmán G.

Director: Ing. Julio Zambrano A. Mg.

IMPLEMENTACIÓN DE UN CONTROLADOR PREDICTIVO BASADO EN MODELO CON RESTRICCIONES SOBRE UN MICROCONTROLADOR DE GAMA ALTA

Éste proyecto de investigación y desarrollo presenta los resultados de la implementación de un algoritmo de control predictivo basado en modelo sobre un microcontrolador de gama alta, con unidad de procesamiento de punto flotante. Para probar el desempeño del controlador, el mismo fue aplicado a un "helicóptero" estático. Se utiliza las funciones de álgebra lineal básica de la librería de código abierto GSL (GNU Scientific Library) para la estructura del algoritmo de control.

Se requiere de conocimientos previos de MATLAB, lenguaje C y teoría de control por parte del lector.