IMPLEMENTACIÓN DE UN CONTROLADOR PREDICTIVO BASADO EN EVENTOS PARA UN SISTEMA DE CONTROL EN RED INALÁMBRICO

CARLOS GERMAN PILLAJO ANGOS

UNIVERSIDAD PONTIFICIA BOLIVARIANA
POSTGRADOS ESCUELA DE INGENIERÍAS
MAESTRÍA EN INGENIERÍA
MEDELLÍN
2017

IMPLEMENTACIÓN DE UN CONTROLADOR PREDICTIVO BASADO EN EVENTOS PARA UN SISTEMA DE CONTROL EN RED INALÁMBRICO

CARLOS GERMAN PILLAJO ANGOS

Trabajo de grado para optar al título de Magister en Ingeniería

Asesor

ROBERTO HINCAPIÉ, PhD

UNIVERSIDAD PONTIFICIA BOLIVARIANA
ESCUELA DE INGENIERÍA
FACULTAD DE INGENIERÍA EN TECNOLOGÍAS DE LA INFORMACIÓN Y LA COMUNICACIÓN
MAESTRÍA EN INGENIERÍA
MEDELLÍN
2017

Marzo 2017

Carlos German Pillajo Angos

“Declaro que este trabajo de grado no ha sido presentado para optar a el título de maestría el cual es inédito y no ha sido presentado en ninguna otra universidad” Art 82 Régimen Discente de Formación Avanzada.

Firma

DEDICATORIA

Este proyecto de Tesis lo dedico a Dios padre creador y a mi familia

AGRADECIMIENTOS

Agradezco a todas las personas que han podido colaborar conmigo en la realización de este proyecto, gracias a mi tutor y mentor PhD Roberto Hincapié

Contenido

Índice de Figuras

Índice de Tablas……

RESUMEN

Los Sistemas de Control en Red Inalámbricos (WNCS, Wireless Network Control Systems), son sistemas espacialmente distribuidos donde la comunicación entre sensores, actuadores y controladores ocurre a través de una red de comunicación digital inalámbrica con ancho de banda limitado. Esta arquitectura distribuida permite que el sistema sea reconfigurable de forma más eficiente y, en general permite reducir los costes de instalación y mantenimiento debido a la disminución del cableado. Además, posibilita que se pueda recurrir a la intranet de una empresa o a Internet para comunicar los diferentes componentes del sistema de control, de tal manera que los elementos del sistema inalámbrico pueden estar distantes entre sí y cerrar el lazo de control a través de la red o de la nube. Los sensores de la planta transmiten su información de medición al controlador y el controlador transmite los datos de control a los actuadores a través de una red compartida inalámbrica, para lo cual los recursos de comunicación y/o fuentes de energía, por ejemplo, las baterías de los dispositivos inalámbricos, son limitados, por lo tanto se podría realizar un controlador inalámbrico que considere reducir las transmisiones desde del sensor al controlador y del controlador al actuador tanto como sea posible garantizando el comportamiento dinámico del lazo cerrado así como la estabilidad de la planta, para lo cual vamos a desarrollar e implementar un controlador PID basado en eventos aplicados a un sistema de control de una planta (Beam-Ball), en donde se aplicaran algoritmos predictivos.

PALABRAS CLAVE.- Algoritmos Heurísticos de sintonización PID, Control basado en eventos, Sistema Beam-Ball, NCS, WNCS.

CAPITULO 1

1 Introducción

El diseño de sistemas Cyberfisicos (CPS) en general y en particular WNCS, requieren una atención simultánea de los aspectos de comunicación, computación y de control, además de los fenómenos físicos. Cuando se utiliza la comunicación inalámbrica, una atención especial debe darse al recurso crítico de control de la comunicación, ya que es a la vez limitado y compartido entre otros dispositivos. Actualmente la tecnología inalámbrica es utilizada en muchas áreas como la electrónica de consumo, redes de sensores, automatización en la industria automotriz, etc. Los WNCS (figura 1.1) son un tipo de los NCS que son sistemas de control distribuido espacialmente, donde la comunicación entre sensores, controlador y actuador es compartida en una red inalámbrica en donde los WNCS tienen limitaciones como el ancho de banda, pero también tiene muchas ventajas como arquitectura flexible, bajo costo de mantenimiento, disminución del cableado y alta movilidad, según A.Haupt [1] Además, da la posibilidad de que se pueda recurrir a la intranet de una empresa o a Internet para comunicar los diferentes componentes del sistema de control, pudiendo estar el controlador lejos del proceso y cerrando el lazo de control a través de Internet.

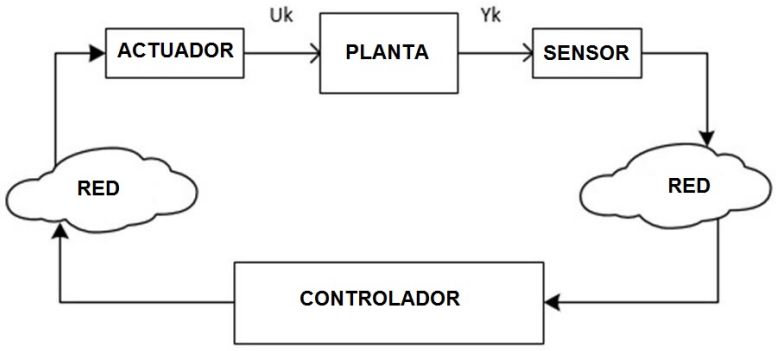


Figura 1.1 Esquema de un sistema de control basado en red inalámbrica

Tradicionalmente, todas las teorías de control moderno y clásico se centran en el estudio de sistemas dinámicos interconectados a través de canales de comunicación ideales. Mientras que la teoría de comunicaciones estudia la transmisión de información a través de canales imperfectos, en los que se pueden producir retardos y pérdidas de información. Por tanto, un WNCS estaría en la intersección de ambas teorías, como se menciona en [2].

Los sistemas de control en red, por su propia naturaleza, presentan varios problemas:

• Retardos variables: debido a que la información que en si es un conjunto de paquetes que viaja por las redes tarda tiempo en transmitirse a través de la red, pues esta información depende de condiciones muy variables como la congestión de la red o la calidad del canal de comunicación, también el tiempo de retardo se debe al tiempo que demora en muestrear la señal continua de la planta en el lado del cliente y el procesamiento de datos en el lado del controlador

• Desconexiones: dependiendo de la calidad de servicio de la red y sobretodo en redes inalámbricas son frecuentes las desconexiones momentáneas, que pueden ocasionar pérdida de información y que, por un momento, el sistema de control se quede sin servicio.

• Ancho de banda limitado: esto debido a que actualmente existen muchos dispositivos que pueden hacer uso de la red limitando la capacidad de transmisión.

Los problemas anteriormente citados se pueden agravar en el caso de procesos que presentan dinámicas inestables y rápidas, debido a que los procesos requieren tiempos de actuación menores que la sumatoria de retardos debido a las comunicaciones. El creciente uso de redes inalámbricas para sensores en aplicaciones de monitoreo se han masificado, no así para los sistemas de control retroalimentado en tiempo real pues su uso aun es incipiente. Las aplicaciones de supervisión no necesitan grandes requisitos de tiempo real y no sufren de problemas críticos relacionados con retardos en el tiempo, a comparación de los problemas típicos de control de retroalimentación. Para superar estos problemas de tiempo hay dos enfoques: La primera opción es optimizar la infraestructura de comunicación para un controlador dado, es decir, modificar la pila de comunicación o adquirir el hardware más costoso disponible para alcanzar una comunicación tan cerca a la ideal como sea posible, que todavía puede resultar en un comportamiento insatisfactorio debido a las limitaciones del hardware. La segunda opción es optimizar el controlador para una infraestructura de comunicación dada, es decir, desarrollar y aplicar algoritmos de control que funcionen y que estén basados en la teoría de WNCS, para de esta manera garantizar un comportamiento deseable en lazo cerrado.

En este trabajo exploramos la segunda opción, cuyo objetivo es el desarrollo e implementación de un controlador basado en eventos con algoritmo predictivo en un sistema de red inalámbrico es decir un WNCS. En un sentido más genérico, el firmware desarrollado e implementado en una tarjeta Arduino constituye el controlador en el cual se realiza el algoritmo PID predictivo basado en eventos donde los datos de control los transmite al sensor-actuador.

Tal como se menciona en [3], la retroalimentación es un concepto que por sí mismo ha revolucionado la manera en cómo se realiza el control, en ese mismo artículo también se señala que el controlador PID es la forma más utilizada de control por retroalimentación. Un aspecto importante que se debe considerar del controlador PID es la estructura del mismo en el cual se procesa el error, se sabe que todo control PID debe ser capaz de manejar las ganancias proporcional, derivativa e integral, sin embargo, existen diversas maneras de hacerlo, es decir, existen diversas estructuras de PID, para este proyecto se plantea la implementación basada en eventos, pero sin importar cual estructura se haya elegido para el controlador, se debe considerar que éste deber ser sintonizado, es decir, se debe elegir valores para cada una de las ganancias del controlador, esto es lo que se conoce como sintonización. Este proceso de sintonización es necesario que se realice con cada nueva planta o cuando el desempeño no es satisfactorio, esta es una característica del controlador que lo hace tan versátil.

En este proyecto se implementan dos técnicas de sintonización basadas en heurísticas, que resuelve el problema de la sintonización de un sistema Beam-Ball, el cual es un proceso que requiere realimentación para su estabilización. Los algoritmos genéticos y los algoritmos PSO son técnicas heurísticas, las cuales buscan encontrar una solución, partiendo de un espacio de búsqueda específico que se define en un principio y que va evolucionando conforme el algoritmo avanza. Ya se han reportado artículos donde se utiliza un algoritmo basado en heurísticas para la sintonización de las ganancias de controladores de tipo PID, así se puede observar en [4], [5] . Los resultados obtenidos en la mayoría de las ocasiones han sido satisfactorios o al menos prometedores, es importante remarcar que a pesar de que los resultados son satisfactorios, la mayoría de ellos provienen de simulaciones de los modelos matemáticos del proceso.

En este trabajo se abordarán cuatro secciones en la primera sección revisaremos los conceptos y características de los sistemas de control en red inalámbricos basado en eventos, para fundamentar la base teórica sobre la cual se presentará una alternativa de controlador PID. En la segunda sección se estudiará el controlador PID basado en eventos con estimadores para sistemas lineales y algoritmos de sintonización heurística, en la tercera sección se determinará el modelo de la planta sobre el cual se va a implementar un controlador PID predictivo y por último en la cuarta sección se diseñará e implementará el controlador PID predictivo inalámbrico y se enseñará las pruebas y resultados de comunicación y control.

CAPITULO 2

2 Sistemas de Control en Red Inalámbrico WNCS

En la actualidad muchos paradigmas se están desvaneciendo sobre todo en los sistemas de control gracias a los sistemas Cyberfisicos que son aquellos que involucran conocimientos en el área de control, comunicación y computación muchos de los retos actuales en el contexto del control de retroalimentación en tiempo real requieren que los canales de comunicación entre sensores, controladores y actuadores deben ser compartido con otras aplicaciones, se sabe que el compartir la información a través de la red tiene relación con propiedades que son fundamentales a los sistemas de control retroalimentado en tiempo real, tales como la velocidad de datos, los retardos, el uso del canal de comunicaciones, propiedades propias de la red que en su mayoría será incierta. Los sistemas de control actuales tratan de optimizar los recursos de comunicación esto debido a que las incertidumbres dadas sobre todo por las redes de comunicación inalámbricas hacen replantear los supuestos básicos en la implementación de los controladores en lazo cerrado de realimentación. Por lo tanto, el campo de los sistemas de control en red inalámbrica (WNCS) actualmente está en desarrollo y más aún sus implementaciones, debido a esto se están generando algoritmos que permitan garantizar las propiedades de control, tales como la estabilidad y el rendimiento, en presencia de la comunicación incierta.

Se debe tomar en cuenta la ocupación de los canales de comunicación inalámbrica, más aun cuando la distancia es grande si se va a utilizar para acciones de control remoto en donde también influye el escalamiento de los procesos. Los sistemas de control que se extienden a través de grandes distancias a menudo se desean ser controlados de manera descentralizada. Por lo tanto, la combinación de la teoría de sistemas de control en red (NCS) y la teoría de comunicación inalámbrica para el control abre un campo de investigación muy grande en temas sobre los que trata este proyecto. Antes de profundizar en la contribución específica de este proyecto, a continuación se dará una breve visión general de cómo los avances de telecomunicaciones, que ya han permitido algunas aplicaciones pertinentes a los sistemas de control, para este caso específico se plantea desarrollar algoritmos de control para un sistema beam and ball.

2.1 Antecedentes

Con la utilización a gran escala de los teléfonos inteligentes se ha provocado que la demanda de datos se dispare, esto ha producido que el campo de las telecomunicaciones crezca a pasos agigantados en un esfuerzo para satisfacer esta demanda. Hoy en día se puede acceder a una conexión de Internet cada vez más rápida en casi cualquier lugar del mundo mediante un teléfono inteligente, lo que hace tan sólo en 20 años para tener acceso a Internet se tenía se necesitaba un gran computador con un módem de acceso telefónico de conexión (56 Kbit / s). Estas redes de alta velocidad se pueden conectar fácilmente y pueden ser utilizadas en una gran variedad de aplicaciones relacionadas con el control, tales como redes inalámbricas de sensores (WSNs) y control de movimiento.

Las WSNs ya han tenido éxito en el uso de telecomunicación inalámbrica, estas redes se basan en dispositivos de bajo costo y consumo (nodos) que son capaces de obtener información de su entorno, procesarla localmente, y comunicarla a través de enlaces inalámbricos hasta un nodo central de coordinación como se describe ampliamente en [6], los avances en una variedad de aplicaciones entre las cuales está el sensado industrial [7],[8], aplicaciones para la salud [9], para la proyectos agrarios [10] por mencionar algunos.

En el pasado, los avances de telecomunicaciones fueron capaces de proporcionar una alta calidad de servicio para aplicaciones de control (por ejemplo, a través de un bus CAN dedicada), actualmente se pretende que los avances de telecomunicaciones más recientes como redes inalámbricas e internet de alta velocidad, imponen una necesidad de desarrollar la teoría de control para utilizar las comunicaciones de forma fiable en un contexto de sistemas de control en tiempo real.

Las WSN`s han demostrado funcionar satisfactoriamente en aplicaciones de monitoreo y supervisión eso se puede atribuir a su dinámica, mientas tanto los datos relacionados como las aplicaciones de control en tiempo real aún no ha despegado en sus aplicaciones es por eso que se debe primero tener mayor comprensión de la teoría de las WNCS la cual está relacionada con la comprensión de las propiedades de control, tales como estabilidad y el rendimiento, en correlación con las propiedades del medio incierto de comunicación.

Una de las importantes ventajas de las redes de información actuales es la capacidad de comunicarse a través de grandes distancias sin cables dedicados, ya que la distribución física de éstos a través de cientos de metros puede convertir a una aplicación en particular en muy costosa y poco flexible. Por otro lado la aplicación de un controlador centralizado el cual debe recoger los datos relacionados con el control con el fin de regular los sistemas que se extienden a través de cientos de metros, lo cual lo hace muy poco práctico debido a las distancias largas de cables de comunicación. El uso de las redes de comunicación actuales ofrece una opción para la aplicación de sistemas de control ya no basado en la instalación de cables dedicados entre todos los dispositivos de control.

Por lo tanto, la combinación de la teoría WNCS y la teoría de control descentralizada surge como un campo de investigación de gran relevancia, es por eso que en este proyecto se utiliza la teoría de WNCS para la implementación un controlador inalámbrico basado en eventos, si bien es cierto que los ingenieros de control tienen mayor confianza en las comunicaciones punto a punto cableadas, sin embargo mediante este proyecto se pretende dar una opción funcional para que se abra paso a proyectos de sistemas dinámicos de control en donde las comunicaciones entre sensor al controlador y del controlador al actuador no sean cableadas, es decir que se pueda liberar la dependencia de cableado dedicado para la comunicación, resolviendo de esta manera muchos desafíos actuales y abriendo muchas posibilidades para probar teorías que están surgiendo y que constituye un campo de investigación muy relevante, como se puede visualizar en [11],[12],[13].

2.2 Los WNCS

Los sistemas dinámicos de control necesitan comunicar información emitida por el sensor hacia el controlador y la información del controlador hacia el actuador, la implementación de los sistemas de control a través de redes digitales ha sido implementadas desde hace mucho tiempo, El creciente interés en el campo de los sistemas de control en red es porque en la actualidad existe una gran variedad de redes digitales disponibles en todas partes en los cuales se puede implementar el lazo de realimentación sin costo adicional, las conexiones inalámbricas facilitan la extensión del área de aplicación de los sistemas de control automático hacia objetos móviles, pues las medidas de las variables y el control de las variables pueden ahora ser transmitidas al controlador desde casi todos los lugares de una planta, así lo menciona en [1].

Una definición del término Sistemas de control en red inalámbrico (WNCS), se describe como: un sistemas de control en lazo cerrado que tienen que tomar en consideración los sistemas en red inalámbrico para la transmisión de datos, esta definición comprende que las redes de datos imponen desafíos que no pueden ser considerados de forma independiente del sistema de control de realimentación. Esto revela un cambio en la percepción de las redes de control de retroalimentación con WNCS convirtiéndose en una de las áreas de investigación más activas de hoy en día. En la siguiente sección, se describe brevemente la aparición de NCS en un nivel general.

2.2.1 Visión de los sistemas de control en red

Los sistemas de control realimentado han sido beneficiados de las redes digitales pues debido a estos los sistemas de comunicación se han descentralizado y distribuido sus acciones de control, pero en la actualidad los requisitos en tiempo real de aplicaciones industriales se necesitan de redes altamente especializadas, razón por lo cual desde hace varios años se han desarrollado buses de campo. Uno de los primeros buses de campo en el mercado era CAN (Controller Area Network), luego se introdujeron en el mercado buses de campo industriales como, Interbus y Device-Net, Profibus y Foundation Fieldbus. Por lo tanto, nos referimos a este tipo de redes como las redes deterministas, las cuales proporcionan soluciones de comunicación patentados para aplicaciones de control y son estándar en el control industrial en la actualidad.

Alrededor del año 2000, los esfuerzos se incrementaron a utilizar las redes con un comportamiento no determinístico para fines de control como lo describe detalladamente en [14]. El gran avance se debe a desarrollos tecnológicos como son las notables mejoras de las tecnologías de comunicación inalámbrica, por un lado, y los avances en las redes de computadoras en general y, en particular, Ethernet, por otra parte, que también allanó el camino para la expansión de Internet, y por lo tanto, se han realizado grandes esfuerzos para emplear estas redes en el control realimentado, la ventaja de las redes de ordenadores en general, tales como Internet, es que la infraestructura de comunicación ya existente puede ser utilizada sin costes de instalación adicionales, en contraste con los buses de campo. Éstas redes (WNCS)

también son no-propietarias, por lo que no hay derechos de licencia que tienen que ser pagados.

Las redes inalámbricas con tecnologías como: Bluetooth (IEEE 802.15.1), WiFi (IEEE 802.11), y ZigBee (IEEE 802.15.4) han experimentado un rápido progreso en los últimos años dando impulso a las aplicaciones de sistemas de control realimentado descritos en [15]. Por ejemplo, las redes inalámbricas aumentan el ámbito del control de retroalimentación a los objetos móviles tales como vehículos aéreos no tripulados (UAV) o redes de inteligencia ambiental [16], y permiten la colocación de actuadores y sensores en lugares que son difíciles de acceder. En este contexto, las redes inalámbricas no sólo reducen los costes de mantenimiento e instalación, sino que también aumentan la disponibilidad y la escalabilidad del sistema.

Por lo que, el uso de redes de datos inalámbricas en los sistemas de control de realimentación crea desafíos significativos. En contraste con los buses de campo deterministas, estas redes tienen características de transmisión estocásticas. Esto significa que no sólo se pueden producir pérdidas de transmisión, como se observa en las redes inalámbricas en particular, sino que también las transmisiones de datos pueden estar sujetos a retrasos desconocidos y variables en el tiempo. Por otra parte, si la red es compartida con otras aplicaciones, varias fuentes compiten por el acceso a la red que implique restricciones adicionales para el sistema de control. La experiencia ha demostrado que los efectos de red inducidos pueden tener un impacto negativo en el rendimiento de un sistema de retroalimentación. Por ejemplo, incluso pequeñas variaciones de los tiempos de transmisión pueden desestabilizar un sistema de control [17]

Por lo tanto, surge la necesidad de nuevas estrategias de control y herramientas de análisis que son capaces de hacer frente a los efectos de red no deterministas. La investigación en esta dirección se inició a principios de la década del 2000 y pronto se hizo conocido como Sistemas de control en red (NCS). Una característica de la investigación NCS es que tiene una complejidad compuesta tanto para los ingenieros de control como para los ingenieros de telecomunicaciones, debido a que el objetivo común es la combinación de la teoría de control, que describe sistemas dinámicos conectados por enlaces perfectos, con el área de la teoría de la comunicación, que se ocupa de la información transportada a través de enlaces imperfectos. Existe información más detallada sobre los NCS y la relación con otras áreas de investigación activas, tales como Sistemas Físicos cibernéticos y se puede encontrar de una manera general en [18],[19], también se describen algunos desafíos de control y comunicación en sistemas en tiempo real conectados a redes mencionados en [20].

2.2.2 Control a través de la red de comunicación

Se sabe que la teoría de control clásica se basa en el hecho de que la comunicación entre sensores, actuadores y controladores es ideal, lo que significa que los datos se comunican y se calculan con retrasos nulos o fijos pero debido a las limitaciones físicas, cualquier medio de comunicación digital induce un retraso distinto de cero y no periódicos, para minimizar los efectos de los retrasos propios de las redes de comunicaciones se puede optar por dos opciones: La primera opción es optar por

configurar una red dedicada. La segunda opción es el desarrollo de la teoría y el uso (WNCS) para ser capaz de especificar en qué condiciones más lentas, menos costoso, el hardware se pueden utilizar de forma fiable en el sentido de dejar de garantizar un comportamiento adecuado de circuito cerrado. En este proyecto se sigue la segunda opción ya que se presentan los resultados de la implementación de un WNCS, contribuyendo al desarrollo de esta teoría.

La teoría de control en red, estudia las propiedades de los sistemas de control, como se puede observar en la figura 2.1a, en el cual se puede evidenciar que en todas las aplicaciones de control realimentado en tiempo real la comunicación entre sensores actuadores y controladores no es perfecta (figura N2.1b) y mucho menos si esta comunicación se realiza mediante WNCS.

Varios efectos de red inducidos se pueden introducir en la realimentación del lazo cerrado cuando se utiliza comunicación inalámbrica como:

- la presencia de un medio de comunicación compartido,
- modificar los intervalos de muestreo / transmisión,
- variación de los retardos de transmisión,
- abandonos de paquetes,
- cuantificación.

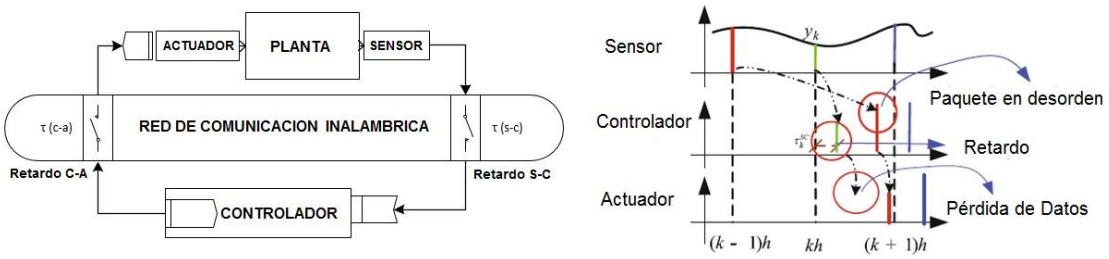


Figura 2.1 a) Estructura de una WNCS; b) Retardos en la comunicación de datos

Se conoce que cualquiera de estos fenómenos degrada el rendimiento e incluso pone en peligro la estabilidad del lazo cerrado, véase, por ejemplo [21], [2]. Dependiendo de la red o la aplicación a realizar, la influencia e importancia de cada uno de los efectos mencionados puede variar de forma significativa. Así en los sistemas que son capaces de transmitir los datos del sensor en un paquete no sufren de un medio de comunicación compartido pero podría tomar más tiempo para recoger todos los datos y preparar la transmisión del paquete, lo que resulta en tiempos de retardo dominantes, no asi en el caso en el que los sensores comparten un bus CAN a través de un cable, en donde la información de los sensores es compartida, pero los retardos de paquetes y peor aun las pérdidas se producen muy rara vez lo que se consigue alta calidad de servicio de la infraestructura de comuncación en el bus CAN. Un ejemplo es el sistema Beam and Ball en donde realizar la configuración de la comunicación,

estabilizar el sistema, tomar la información del sensor, realizar los algoritmos respectivos en el controlador y nuevamente enviar esa información al actuador en todo este proceso se presentan todos los efectos de los sistemas realimentados en red inalámbrica antes mencionados, es por esta razon y motivo de este proyecto en el cual se desarrolla e implenta un controlador PID predictivo basado en eventos capaz de estabilizar el sistema.

En este trabajo, concentramos nuestra atención en los siguientes efectos. El primer efecto, la presencia de un medio de comunicación común, requiere un protocolo de comunicación que relacione los datos con el control. Aunque otros autores han considerado el caso cuando el protocolo es estocástico, véase, por ejemplo [22], suponemos que el protocolo de comunicación es un proceso determinista, como también asumido en [23],[24]. Los siguientes dos efectos como la variación de los intervalos de transmisión y diferentes retardos de transmisión, por lo general no se consideran deterministas.

Algunos autores han centrado su estudio en el caso en donde la información estocástica está incluido en los modelos de los retrasos y los intervalos de transmisión a través de una función de distribución de probabilidad [25], o una cadena de Markov [26]. En este proyecto centramos nuestra atención en el caso 'distribución de probabilidad libre', como la sumatoria de retardos que también es estudiado en [27]. En este caso, los intervalos de transmisión variables en el tiempo y la incertidumbre y los retrasos de transmisión se toman de un conjunto acotado, sin pretender ningún conocimiento sobre la distribución de probabilidad particular. Por supuesto, si se puede obtener un modelo estocástico fiable y precisa de los canales de comunicación, siempre es más beneficioso usar esta información en el modelo y el análisis. Sin embargo, en general, un modelo estocástico exacto puede ser difícil de obtener, en cualquier caso, tener una variedad de herramientas disponibles que tengan en cuenta los diferentes supuestos del modelo de red es beneficioso ya que las redes exhiben comportamientos muy diferentes dependiendo de las configuraciones de red y los estándares de la red.

2.3 Control Basado en Eventos

El control basado en eventos se define cuando, el lazo de realimentación está cerrado sólo si un evento indica que el error de la señal controlada excede de un límite tolerable y desencadena una transmisión de datos desde los sensores a los controladores y los actuadores. Por lo tanto, el control basado en eventos es un método importante para la reducción de la carga de comunicación de una red digital. A continuación se presentan las ideas principales de control basado en eventos.

2.3.1 Introducción al control basado en eventos

Control basado en eventos es una metodología de control que está siendo desarrollado como un medio para reducir la comunicación entre los sensores, el

controlador y los actuadores en un lazo de control. Los instantes de muestreo no están determinados periódicamente por un reloj, sino por un detector de eventos, que se adapta el flujo de información en el lazo de retroalimentación con el comportamiento actual del sistema de lazo cerrado. Una comunicación entre los componentes se invoca sólo después de que un evento ha indicado que el error supera un límite tolerable. Este principio de funcionamiento difiere respecto a la retroalimentación de datos muestreados en donde los datos de los sensores se comunican con el controlador en tiempos de muestreo periódicos, es decir que la comunicación se lleva a cabo independientemente de la magnitud del error, en estas situaciones, los recursos de comunicación e informáticos se utilizan innecesariamente.

En la Figura 2.2 muestra los principales componentes de un lazo de control basado en eventos, en donde: La planta tiene una salida en el tiempo continuo y(t), una entrada de tiempo u(t) y el estado x(t), que se generan continuamente por el generador de entrada de control y es continuamente evaluado por el detector de eventos. Los enlaces de comunicación representados por las líneas de trazos sólo se utilizan después de que el detector de eventos ha indicado que el error supera un límite tolerable dado en algún tiempo t_k, donde un evento e_k dado por el estado actual $x(t_k)$ se transmite al controlador. El controlador determina la nueva entrada de u_k, que se utiliza en el generador de entrada de control para determinar la entrada continua u(t) en el intervalo de tiempo $[t_k, t_{k+1})$ hasta el próximo evento que se produce en el momento t_{k+1}.

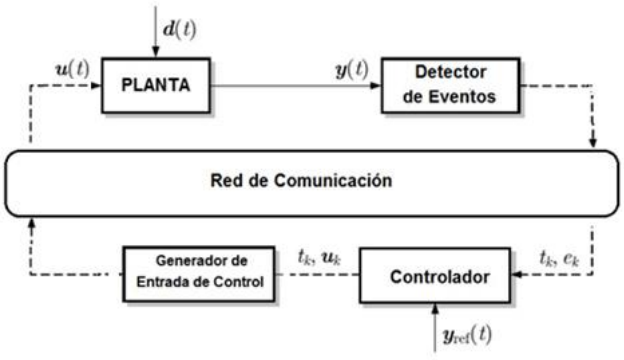


Figura 2.2 Lazo de control basado en eventos

En el control basado en eventos, se rompre el paradigma del supuesto fundamental de la teoría de control para datos muestreados que genera un esquema de disparo periódico invocado por un reloj. Por lo tanto, para el control basado en eventos los modelos de tiempo discreto bien conocidos de la planta, el controlador y el sistema de circuito cerrado no se puede aplicar, pero una nueva teoría tiene que ser desarrollada, que tiene en cuenta el comportamiento de los componentes asíncronos. Los nuevos objetivos principales de análisis y diseño de esta nueva teoría se refieren a la elección del generador de eventos y del generador de entrada de control.

El detector de eventos determina

- los instantes de tiempo t_k, (k = 0, 1, ...) en el que se invoca la siguiente comunicación entre el detector de eventos, el controlador y el generador de entrada de control, y
- la información que se comunica desde el sensor hacia el controlador.

El generador de entrada de control determina la señal u(t) de forma continua durante el intervalo de tiempo $t \in [t_k, t_{k+1})$ en función de la información obtenida en t_k tiempo.

Las principales preguntas que hay que responder

- En que tiempo t_k en un lazo de realimentación se tiene que cerrar mediante el uso de los enlaces de comunicación
- Cual es la información que el error e_k debería comunicar y
- Cómo el generador de entrada de control debería determinar la entrada de control u(t) entre dos eventos sucesivos.

Se han propuesto varios métodos diferentes para control basado en eventos en los últimos años, que distinguen con respecto a las respuestas dadas a estas preguntas. Algunos de ellos han sido publicados bajo diferentes nombres como el control manejado por evento, control activado por eventos, el muestreo de Lebesgue, control de banda muerta o envio de un delta de control. Las encuestas y las introducciones a estas técnicas se pueden encontrar en [28],[29]

Escenarios de aplicación. Existen varios motivos para el uso de retroalimentación basado en eventos:

a) Cuando el intercambio de información en el lazo de realimentación se debe reducir a la comunicación mínima que es necesaria para asegurar un rendimiento de sistema requerido. Si la información se transfiere hacia y desde el controlador en una red de comunicación digital inalámbrica, un flujo de información reducida disminuye el riesgo de una sobrecarga de la red. Para nodos inalámbricos, la actividad reducida ahorra energía;

b) Cuando existe sistemas en los que la estructura física requiere que las medidas o acciones de control tienen que ser tomadas en instantes de tiempo que dependen de la dinámica de la planta. Por ejemplo cuando existen medición de elementos de rotación y en base a su posición se toma acciones de control y;

c) Cuando, los protocolos de comunicación asíncrona y software en tiempo real hacen que no permita transferir y procesar información en momentos específicos del reloj, sino conducir a un comportamiento asíncrono de todos los componentes de un circuito de retroalimentación. Los sensores y actuadores del mismo modo se activan por los acontecimientos, ya que trabajan si llega alguna información nueva. Para este tipo de componentes el esquema de trabajo basado en eventos es más "natural" que el muestreo periódico.

Propiedades fundamentales de control basado en eventos.- El principio de funcionamiento orientado a eventos implica que la entrada a la planta u(t) se determina

por una combinación de control predictivo y de control de realimentación. En los tiempos de eventos t_k, la entrada $u(t_k)$ depende de un modo de lazo cerrado del estado actual $x(t_k)$ (siempre que los enlaces de comunicación no introducen un retardo de tiempo considerable), mientras que entre los dos tiempos de eventos t_k y t_{k+1} la entrada u(t) se genera como control en lazo abierto en función de datos pasados de u_k.

El objetivo del control basado en eventos es muestrear sólo si se debe evitar una degradación severa del rendimiento, la mayoría de los esquemas de control basados en eventos no pueden garantizar la estabilidad asintótica del sistema de circuito cerrado. En cambio, el estado de la planta x(t) deben mantenerse en los alrededores de Ω del estado de equilibrio x^. "La propiedad $x(t) \in \Omega$ para todo $t \geq T$ se llama la acotación final o estabilidad práctica del sistema en lazo cerrado. Por lo general, el tamaño del conjunto Ω depende del limite de umbral de eventos e", esta definicion dada en [1]

Comparación entre el control basados en eventos y Control de datos muestreados. Existe una diferencia entre el control basado en eventos y el control de datos muestreados, esto lo demuestran los resultados analíticos obtenidos para los sistemas de primer orden presentados en [30],[31],[32],[33]. Los trabajos citados anteriormente demuestran que el muestreo basado en eventos de hecho puede conducir a una reducción considerable de la comunicación dentro del lazo de control. Además, si el sistema está muy perturbado, el lazo de control basado en eventos puede tener un mejor rendimiento que el lazo de datos muestreados, ya que en esta situación se invoca la comunicación con más frecuencia que el reloj.

Se mensiona los métodos que son aplicables para los sistemas de orden superior y diferentes métodos de generación de eventos de entrada-activación y control. Los métodos diferentes se describen a continuación:

• **Realimentación de estado basado en eventos**: El generador de entrada de control está construido de manera que el sistema de lazo cerrado imita el comportamiento de un lazo de realimentación de estado continuo con precisión ajustable. Se muestra que el generador de entrada de control tiene que incluir un modelo del lazo de realimentación de estado continuo.

• **Control basado en eventos distribuido:** Para los sistemas interconectados, la idea de realimentación de estado basado en eventos puede o bien ser implementado como un controlador distribuido, lo que conduce al mismo rendimiento general del sistema como la retroalimentación centralizada, o se pueden aplicar por separado a los subsistemas aislados resultantes en un esquema de control basado en eventos descentralizados.

• **Control basado en optimización:** El controlador basado en eventos se puede obtener como la solución de un problema de control óptimo, si se divide el espacio de estados y la mejor entrada constante posible se aplica siempre que el estado permanece en el misma partición de espacio de estado.

• **Control basado en eventos de sistemas estocásticos:** Si se formula como un problema de optimización estocástica, el detector de eventos y el generador de entrada de control tienen que ser diseñadas de manera simultánea lo cual conduce a un problema de optimización muy complejo, sin embargo, ambos componentes se encuentran en una estructura de información anidado, en el que el detector de eventos conoce las x(t) del estado de la planta para todo el tiempo t y el generador de entrada de control sólo conoce el estado $x(t_k)$ en un t_k tiempo del evento, el problema en general se puede descomponer en dos problemas de optimización anidados que se pueden resolver de forma secuencial.

De los métodos anteriormente descritos en este trabajo se realizó el control basado en eventos de realimentación de estado, pues el detector de eventos esta dado en base al umbral del error y la planta sólo transmite datos si el error sobreapasa el límite umbral, en cambio el generador de entrada de control está en el lado del controlador y esta implementado de manera que el sistema de lazo cerrado imita el comportamiento de un lazo de realimentación de estado continuo con precisión ajustable. El generador de entrada de control tiene que ejecutar un algoritmo predictivo en base a los datos recibidos por el detector de eventos el cual esta en el lado del sensor en donde se incluye el modelo del lazo de realimentación de la planta de estado continuo.

2.4 Características de los WNCS

Los primeros sistemas de control en red son sistemas cableados, donde los sensores y controladores están conectados a través de un medio común cerrando cada uno de los muchos lazos de control con cables dedicados. Como uno de los parámetros más críticos para la estabilidad de un sistema de control es el retraso como lo demuestra en [34], el protocolo de red utilizado en tales sistemas de control deben satisfacer las restricciones de tiempo real. Algunos protocolos de cable que pueden trabajar bajo limitaciones de tiempo real son: Control de red de área (CAN) desarrollado para ser utilizado en la automatización de los vehículos, y PROFIBUS, diseñado para la fábrica y automatización de procesos. Estos sistemas con comunicación alámbrica estan siendo reemplazados ampliamente en la industria mediante los WNCS como se puede apreciar en [35], pero a pesar de que reducen el cableado y todos los costos asociados con ella, existe problemas con los retardos, perdida de paquetes aleatorias que todavía no se resuelven satisfactoriamente. Por lo tanto, una gran cantidad de investigación se ha centrado en los sistemas de control inalámbrico, en el que el cableado esta ausente entre los elementos que componen el sistema a controlar como el sensor, el controlador y el actuador.

2.4.1 Sistemas Inalámbricos, Las imperfecciones de la red inalámbrica

Una de las claras desventajas mas evidentes de utilizar el medio inalámbrico para la comunicación son los retardos de envio recepción de los paquetes de información conocido como jitter. Para un sistema de control inalámbrico de tiempo típico impulsado, a intervalos regulares, la planta se toman muestras y las muestras están programadas para ser entregados al controlador. Sin embargo, debido a la naturaleza estocástica del medio se introducen retardos, el controlador recibe paquetes a intervalos irregulares, como se ilustra en la figura 2.3

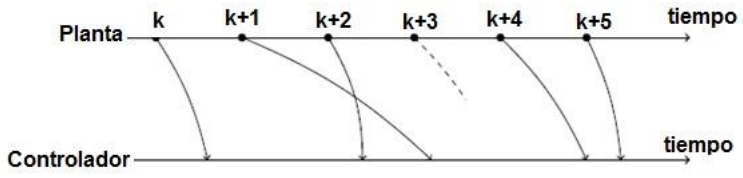


Figura 2.3 Transmisión inalámbrica irregular

Los factores que contribuyen a este comportamiento son

- Retardo de acceso de red: El tiempo necesario para que la red esté disponible para aceptar los datos
- Tiempo de retardo del paquete de datos: El tiempo para el transmisor coloque un paquete de datos en la red
- Retardo de transmisión: El tiempo que el paquete pasa dentro de la red la cantidad de estos retrasos depende de la anchura de banda del canal, tamaños de paquetes y perdidas en generales.

Otros problemas que se dan aparte del retardo de llegada de los paquetes al controlador, también hay un riesgo de paquetes que se pierden dentro de la red debido a la ruido, interferencia, fallos en los nodos y las colisiones como se describen en [36]. Cuando se detecta la pérdida de paquetes, se puede pedir a ser retransmitido, pero es muy probable que en el momento en que se vuelve a transmitir, los datos sean obsoletos. Por lo que, es más útil para transmitir un paquete nuevo en lugar del paquete antiguo, esto prácticamente implica que el sistema debe tolerar la pérdida de paquetes.

Frente a la pérdida de paquetes una aparente solución debería ser mantener el tiempo de muestreo bajo y el envío de forma redundante de muchos paquetes, de modo que el controlador pueda utilizar los que llegan a tiempo. Pero, dado el limitado ancho de banda del medio, el envío de paquetes a alta frecuencia dará lugar a más tráfico, por lo tanto más colisiones y retardos más largo. A demás, los sensores inalámbricos generalmente funcionan con energía limitada para mantener su esperanza de vida larga, lo cual entraría en conflicto con el envío de forma redundante de muchos paquetes. Por lo tanto, se han preferido las soluciones con mayor tiempo de muestreo.

Acceso a la red

El acceso al medio debe ser regulado cuando existen transmisiones de paquetes de control, sobre todo para mantener el retardo de acceso a la red y las colisiones bajo control. Para ello se puede utilizar una forma determinista la cual sera planificar los tiempos cuando el nodo transmite, conocido como acceso múltiple por división de tiempo (TDMA), de manera que cada nodo tiene un período específico cuando se puede transmitir. Otra forma determinista es dividir la banda de frecuencia de manera que cada nodo tiene una frecuencia exclusiva para transmitir, conocida esta técnica como división de frecuencia de acceso múltiple (FMDA). Estas formas de transmitir

determinan mayor límite de retraso, pero por lo general son difíciles de aplicar debido a la construcción de un planificador y a la sincronización de los nodos de modo que puedan cumplir con la programación del planificador.

Otra forma aleatoria para acceder al medio es poner en práctica protocolos de acceso aleatorio, donde un nodo que tiene un paquete para transmitir, primero comprueba si el medio está disponible, y luego envía de inmediato si lo es. Si el medio está ocupado, el nodo espera durante una cantidad de tiempo aleatorio y vuelve a intentar enviar. Estos protocolos de acceso aleatorio, inevitablemente aumentan retardo de acceso a la red, lo cual hace más difícil de ser empleados en los sistemas de control.

Al igual que en todas las redes de medio compartido, el control de acceso al medio (MAC) es una técnica importante que permite el funcionamiento exitoso de la red. Una tarea fundamental del protocolo MAC es evitar colisiones de nodos interferentes. Hay muchos protocolos MAC que se han desarrollado para las redes inalámbricas de voz y datos. Entre estos incluyen el acceso múltiple por división de tiempo (TDMA), acceso múltiple por división de código (CDMA) y protocolos basados en contención como IEEE 802.11

Estructura del Sistema

La representación de red compartida en la que los sensores, el controlador y el actuador están espacialmente separados unos de otros figura. 2.4, y la comunicación se realiza a través de la red. En esta arquitectura, los costos de la utilización de una red como retardo se introducen dos veces, una vez desde los sensores al controlador, y de nuevo desde el controlador al actuador. Habiendo considerado que los actuadores generalmente exigen alta potencia, esta arquitectura también es beneficioso en términos de fuente de alimentación del controlador. Por lo tanto, si la acción de control no se distribuye, es favorable para conectarlos. Esta es la arquitectura que se ha utilizado en este trabajo.

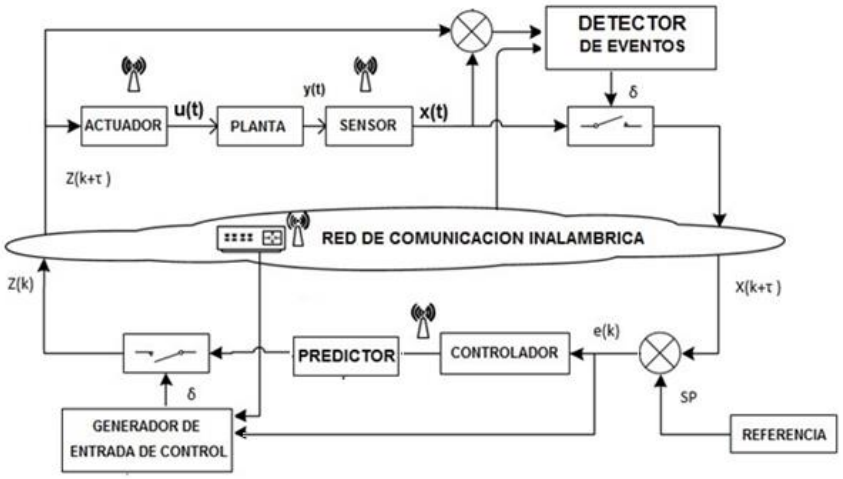


Figura 2.4 Estructura directa de la WNCS

2.4.2 Modelo de retardo

Existe diferentes propuestas para modelar el retardo inducido por la red y pérdidas de paquetes, una descripcion detallada lo tenemos en [37], en donde se ha descrito los siguientes modelos de retardo.

Retardo Constante.- La forma más determinista para modelar el retardo de red es la determinación de un valor y utilizar este como el retardo constante predeterminado, que sólo puede ser aplicable a casos deterministas. En este caso el valor medio del peor retardo puede ser utilizado como el retraso constante, para el análisis. Sin embargo este tipo de seguridad disminuirá el rendimiento ya que el retraso de control puede ser innecesariamente largo. Este modelo de retardo no está adaptado a las redes con protocolos de acceso al medio estocásticos. $\tau_1(t) = \tau_c$.

Retardo con Distribución Randomica Gauseana.- Para realizar análisis y representar las variaciones en los protocolos de acceso aleatorio, se asume un modelo de distribución normal que representa el retardo, donde la media y la varianza se seleccionan para efectos de simulación del comportamiento de los retardos en la red. $\tau_2(t) = x_1(t), X_1 \sim N(\mu, \sigma)$

Retardo Aleatorios Parcialmente Independiente.- En una red los retardos son estocásticos y pueden tener varias fuentes asi por ejemplo:

a) Tiempo de espera a que la red este activa;

b) Tiempo de espera para la transmisión de los mensajes ;

c) Si se producen errores de transmisión, puede ser necesaria una retransmisión ;

d) Debido a colisiones si dos nodos intentan enviar al mismo tiempo, la resolución de esto puede incluir una espera aleatoria.

Debido a que las actividades del sistema generalmente no están sincronizadas entre sí, el número de las causas de retardo enumeradas anteriormente que se producirán es aleatorio. Para tener en cuenta la aleatoriedad de los retrasos de red en el modelo, los retornos de tiempo se pueden modelar tomando como base una distribución probabilística. Para mantener el modelo simple de analizar se puede suponer que el retardo de transferencia sea independiente de los tiempos de retardo anteriores. En un sistema de comunicación real, sin embargo, el tiempo de transferencia se correlacionará usualmente con el último retardo de transferencia. Por ejemplo, la carga de la red, que es uno de los factores que afectan al retardo, varía típicamente con una constante de tiempo más lenta que el período de muestreo en un sistema de control, es decir, el tiempo entre dos transferencias.

En este proyecto vamos a permitir que el modelo tenga diferentes distribuciones de probabilidad para el retardo desde el sensor al controlador Tk^{sc}, y para el retardo desde el controlador al actuador, Tk^{ca}, como se indica en la figura.2.5

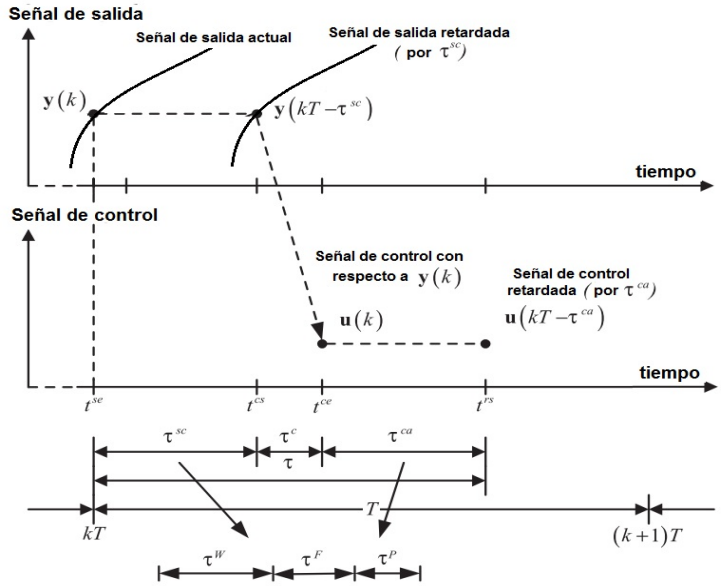


Figura 2.5 Diagrama de retardos causados en la red

2.4.3 Modelo de abandono de paquetes

En el caso de abandono o pérdidas de paquetes, la forma mas simple para modelar es tratarlo como un proceso de Bernoulli , donde la variable pérdida de paquetes es única e independiente. Cuanto mayor es la probabilidad de transmisión mejor es la condición de la red. Sin embargo, se ha observado que las pérdidas de paquetes generalmente aparecen en ráfagas [38], por lo tanto no es una buena aproximación para ignorar esta correlación y el paquete de modelo temporal de deserción por un modelo de pérdida independiente. En lugar de esto, un método más completo, se considera como el modelo de canal Gilbert-Elliot, que es un modelo de cadena de Markov de dos estados, compuesto de estados "buenos" y "malos". En el estado de "bueno", los paquetes se entregan sin un error, y la tasa de fracaso q, es la probabilidad de la transición al estado "malo", en la que se pierden los paquetes. Tasa de recuperación, la probabilidad de la transición al estado "bueno" se denota por p. Este modelo, representado en la Figura 2.6, captura la correlación temporal de las pérdidas de paquetes y se puede extender a las cadenas de Markov con más estados si es necesario.

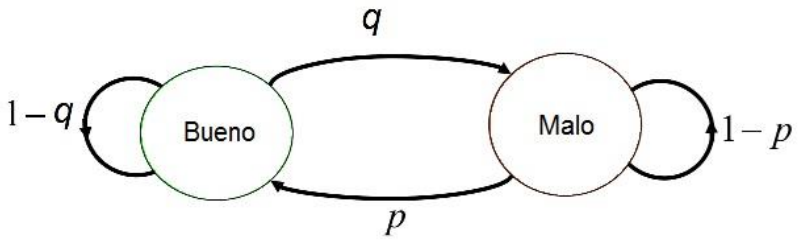


Figura 2.6 Modelo de pérdida de paquetes Gilbert-Elliot

CAPITULO 3

3 Controlador basado en eventos

3.1 Sistemas basados en eventos.

Los sistemas de control que se han utilizado hasta ahora en muchas áreas de la ingeniería se han basado en un muestreo periódico, para dichos sistemas existe toda una teoría probada y muchas herramientas matemáticas dando resultados teóricos bien establecidos, así por ejemplo en [39], en cambio el muestreo basado en eventos aparecen de manera natural en las redes de sensores inalámbricos (WSN), donde los sensores envían sus salidas en función de alguna condición. Además, las estrategias basadas en eventos se han utilizado en áreas tales como el control de los procesos industriales [40], la planificación de la trayectoria del robot [41], y el control del motor [42]. Sin embargo, el control basado en eventos ha sido utilizado principalmente en forma ad-hoc, debido a la falta de resultados teóricos, que han comenzado a estar disponibles por ejemplo en, [43], [31],[2]. Y últimamente, control basado en eventos también se está utilizando en sistemas distribuidos [44].

Los sistemas basados en eventos frecuentemente se clasifican en dos tipos, los sistemas activados por eventos [45] y sistemas auto-disparados [29],[11]. En los primeros por lo general no se tiene un modelo de la planta, donde los cambios medidos en el estado provocan cambios esporádicos del controlador, en los sistemas auto-disparados el controlador calcula el momento en que se activa el siguiente evento en base a una estimación del estado de la planta obtenida a partir de una función de estimación que tiene que ver con el comportamiento del proceso. Para el desarrollo de este trabajo se utilizó los sistemas auto-disparados

Algunos trabajos relacionados con los sistemas activados por eventos son los de [46], [47], donde se utiliza una estrategia periódica activado por eventos, en la que la condición de evento es verificada en un período de muestreo constante. En [32], los autores analizan la relación entre el control de datos muestreados y dos estrategias activadas por eventos, a saber, el control de banda muerta y el control activado por eventos basado en modelos, concluyendo que el enfoque basado en eventos garantiza la mejor estabilidad y aprovecha las propiedades de comunicación.

Razón por la cual la industria ha volcado sus esfuerzos en aprovechar las redes inalámbricas de comunicación para el control de muchos procesos, estandarizando el uso de controladores, buscando realizar implementaciones de controladores PID basados en eventos como se tiene en [48],[49]. Un estudio exhaustivo de los diferentes métodos propuestos en la literatura para el control PID basado evento puede encontrarse en [50].

Este proyecto se centra en la implementación de un controlador PID basado en el muestreo activado por eventos y en particular sobre el paso a nivel o el muestreo *send-on-delta*, que consiste en tomar una nueva muestra cuando un cambio mayor que un umbral definido δ se detecta en la señal. En la práctica, este sistema es equivalente a introducir una no linealidad

(NL) que se puede caracterizar como una cuantificación de N niveles con histéresis. El comportamiento de un esquema de control basado en el muestreo del paso a nivel se estudió considerando que la toma de muestras se encuentra después de la salida del proceso, el cual, presenta una configuración de un esquema de control basado en las transmisiones inalámbricas (figura 3.1), y tiene diferentes propiedades, el caso estudiado corresponde a una planta (P(s)=beam-ball) con un sensor inalámbrico que toma medidas de la variable de proceso (y=distancia) y los envía al controlador (C(s)), separado físicamente del sensor, luego los datos del sensor son procesados en el controlador y el resultado de este (u) son enviados al actuador que igualmente esta físicamente separado, pero está conectado a la planta.

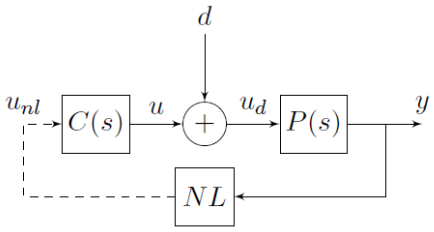


Figura 3.1 Esquema de un sistema de control basado en eventos

En este capítulo se caracteriza un enfoque sistemático el comportamiento del control basado en eventos con el proceso del controlador PID, considerando el retardo de tiempo, además se pretende dar una alternativa para el análisis y aplicación de controladores PID basado en eventos mediante comunicación inalámbrica, la obtención de resultados han sido confirmados con la práctica.

3.2 Muestreo basado en eventos

3.2.1 Sistemas de datos muestreados

Es conocido que una gran mayoría de sistemas en ingeniería son continuos en la naturaleza, pero gracias a la capacidad de las computadoras digitales para procesar datos discretos, los sistemas de tiempo continuo se controlan usando observaciones muestreadas tomadas en instantes discretos. Por lo tanto, los sistemas de control resultantes son un híbrido, que consiste en interactuar con componentes discretos y continuos como se muestra en la figura 3.2 Estos sistemas híbridos, en los que, el sistema a controlar evoluciona en tiempo continuo y el controlador evoluciona en tiempo discreto, se denominan sistemas de datos muestreados, como lo describe en [51]

La característica significativa del diseño de sistemas de datos muestreados que lo distingue de las técnicas estándar para el diseño de sistemas de control es que debe enfrentarse con modelos de plantas y leyes de control que se encuentran en diferentes dominios. Existen tres metodologías principales para el diseño y análisis de sistemas de datos muestreados que están representados gráficamente en la figura 3.2 donde G es un proceso de tiempo continuo y Kd es una ley de control de tiempo discreto. Los tres métodos comienzan con el principio del modelo

de tiempo continuo G y apuntan a diseñar el controlador de tiempo discreto Kd y analizar su rendimiento.

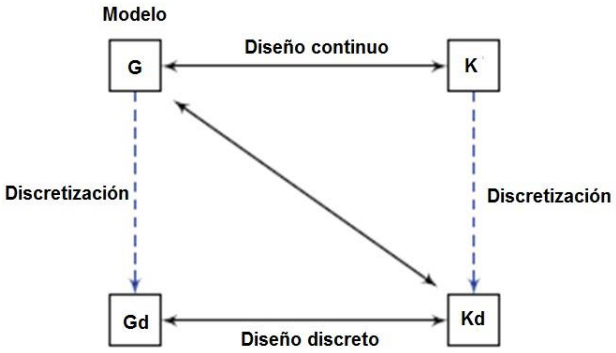


Figura 3.2 Enfoques de sistemas de datos muestreados

Las dos aproximaciones bien conocidas siguen los caminos alrededor del perímetro del diagrama. El primero es realizar todo el análisis y diseño en el dominio de tiempo continuo utilizando un sistema que se cree que es una aproximación cercana al sistema de datos muestreados. Esto se logra asociando cada controlador de tiempo continuo K con una aproximación de tiempo discreto Kd mediante un método de discretización; Síntesis y análisis del controlador se realizan entonces en tiempo continuo, con la suposición subyacente de que el comportamiento del sistema de lazo cerrado obtenido con controlador K refleja de cerca lo conseguido con la implementación de datos muestreados Kd. Por lo tanto, este método no aborda directamente la cuestión de la implementación en la etapa de diseño.

El segundo enfoque comienza discretizando el sistema de tiempo continuo G, dando una aproximación de tiempo discreto Gd, por lo tanto, ignorando el comportamiento entre muestras. Entonces el controlador Kd está diseñado directamente en tiempo discreto usando Gd, con la creencia de que el rendimiento de este sistema de tiempo puramente discreto se aproxima al del sistema de datos muestreados.

El tercer enfoque ha atraído considerable actividad de investigación, por su facilidad en la implementación, en este enfoque, el sistema G y la interconexión Kd del controlador se tratan directa y exactamente. Este trabajo se centra, en la medida de lo posible, en este enfoque, mientras que en algunos casos utilizaremos el segundo enfoque para explicar más sencillamente el diseño del controlador.

3.2.2 Las técnicas de muestreo y control basadas en eventos

En los últimos años han desarrollado estas técnicas los investigadores que integran tecnologías en varias áreas. La reducción de la información en el intercambio entre los agentes que intervienen en un lazo de control sean sensores, controladores y actuadores es de suma importancia, más aun cuando hay limitaciones en el flujo de información debido a que los datos se intercambian por las redes inalámbricas, por lo que reducir la carga de tráfico es

clave, pues cuanto más tráfico hay mayor es la posibilidad de pérdida de datos y se puede experimentar retardos aleatorios, por lo que una gran reducción en el tráfico es una problema esencial en las redes inalámbricas, si mayor es la reducción en el flujo de información, mayor será la disminución de las operaciones y transmisiones de cálculo y por lo tanto más largo es el tiempo de vida de las baterías que abastecen de energía al sistema.

Es importante definir previamente un evento el cual es algo que ocurre en un determinado tiempo, que se produce o se dispara cuando ocurre alguna condición booleana o lógica por ejemplo una condición genérica basada en eventos se representa como: $(Condición\ lógica\ es\ verdadera)$ó $(t_{evento} \geq t_{max})$

La condición lógica se detecta si algo pasa en un momento determinado, y por lo tanto se considera el término asíncrono de la expresión. Para el control de procesos, la condición lógica se compone de operaciones booleanas, donde las variables son las señales que el sensor recibe del proceso o la acción de control producida por un controlador. Esto es utilizado debido a que hay situaciones en las que la condición lógica no puede ser verdad y, por lo tanto, no se pueden producir eventos.

Porque la activación de un evento en el control de procesos significa, en general, la activación de un controlador para realizar una o varias acciones de control en función de la estrategia de control, un excesivo número de eventos, podría generar un gran número de acciones de control que posiblemente puedan dañar los actuadores o en su defecto saturar los canales de comunicación en el lazo de control, produciendo retrasos y abandonos de datos. Para poder evaluar y detectar el momento exacto en el tiempo cuando una condición lógica se cumple, se debe evaluar de forma continua en tiempo. Por lo que, una evaluación continua de la condición implica un muestreo continuo de las variables o funciones implicadas en la expresión lógica; pero los sistemas de control digitales actuales implementan estrategias de muestreo discretos síncronos. Por esta razón, los enfoques prácticos de control basado en eventos se implementan para muestrear las variables y evaluar las condiciones basadas en eventos tan rápido como sea posible. Esto se conoce como el enfoque compuesto, donde los eventos asíncronos se pre sincronizan por la vía rápida tomando muestras periódicas [52].

En la Figura 3.3 se presenta el resultado de la evaluación de una simple condición basada evento usando una señal continua obtenida, por ejemplo, de un sensor incorporado en un proceso industrial. En este ejemplo, la condición lógica asíncrona para evaluar es:

$$|y(t_{act}) - y(t_{last})| \geq \Delta \qquad (3.1)$$

donde $y(t_{act})$ es el valor actual de la salida del proceso $y(t)$, $y(t_{last})$ es el valor previo de la condición verdadera de y (t), y Δ se conoce como umbral o delta, que asume un valor menor que el rango dinámico de $y(t)$. Esto se expresa como $\Delta < |y(max) - y(min)|$. Las señales resultantes de la Figura 3.3 se obtienen mediante la aplicación de los valores de las señales que hacen que la verdadera condición lógica a una retención de orden cero (ZOH).

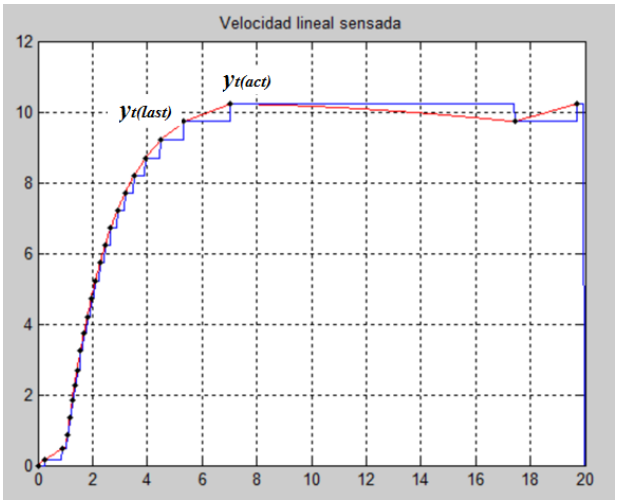


Figura 3.3 Muestreo basado en eventos

El proceso de generación de una señal discreta en el tiempo mediante la evaluación de una condición basada en eventos sobre una señal continua en el tiempo se conoce comúnmente como el muestreo basado en eventos. Otros nombres que se encuentran en la literatura de control son el muestreo de frecuencia variable [53], el muestreo adaptativo [54], el muestreo de banda muerta [55], el muestreo de Lebesgue [31], muestreo *sent-on-delta* [52], y muestreo por cruce de nivel [56].

Un resumen de las estrategias de muestreo basados en eventos se muestra en la Tabla 1.

Condición Lógica	Verdadero cuando
$\lvert \boldsymbol{y}(\boldsymbol{t}) - \boldsymbol{y}(\boldsymbol{t_{last}}) \rvert \geq \Delta$	La diferencia entre la salida actual y la salida anterior es mayor que Δ
$\int_{\boldsymbol{tant}}^{\boldsymbol{tact}} \lvert \boldsymbol{y}(\boldsymbol{t}) - \boldsymbol{y}(\boldsymbol{t_{last}}) \rvert \geq \Delta$	La integral de la diferencia entre la salida actual y la salida anterior entre el tiempo actual y anterior es verdadero cuando es mayor que Δ
$\lvert \hat{\boldsymbol{y}}(\boldsymbol{t}) - \boldsymbol{y}(\boldsymbol{t}) \rvert \geq \Delta$	La diferencia entre el valor estimado de la señal de salida y la salida es mayor que Δ
$\int_{\boldsymbol{tant}}^{\boldsymbol{tact}} \lvert \hat{\boldsymbol{y}}(\boldsymbol{t}) - \boldsymbol{y}(\boldsymbol{t}) \rvert \geq \Delta$	La integral de la diferencia entre el valor estimado de la señal de salida y la señal de salida es verdadero cuando es mayor que Δ
$\int_{\boldsymbol{tant}}^{\boldsymbol{tact}} \lvert \boldsymbol{y}(\boldsymbol{t}) - \boldsymbol{y}(\boldsymbol{t_{last}}) \rvert^2 \geq \Delta$	La integral del cuadrado del error entre la salida actual y la salida anterior entre el tiempo actual y anterior es verdadero cuando es mayor que Δ

Tabla 3.1 Condición lógica utilizada para el muestreo por eventos

¿Cómo funciona el muestreo basado en eventos en el marco del control automático de procesos? En los sistemas de control digital, las acciones de los agentes de control (sensores, controladores, actuadores) se rigen por una señal de reloj común. Esto da lugar a las acciones sincronizadas más de un intervalo (h_{nom}) de muestreo que incluye la toma de muestras del proceso por el sensor, la activación del controlador y la aplicación de la acción de control al actuador. Esta es la base para el muestreo de los sistemas de control. Pero, si se aplica el paradigma basado en eventos, diferentes eventos independientes causarían los agentes de control para llevar a cabo las acciones de una manera asíncrona. Una condición basada en eventos podría ser diferente para cada agente para detectar un cambio en una variable de sistema o una función de la variable de sistema. El ejemplo de esta situación es cuando un sensor detecta un cambio en una señal (el evento) y envía una muestra del proceso de salida al controlador (la acción).

A continuación se realiza una clasificación general de las estrategias de control basadas en eventos que se ha encontrado en la literatura, además se revisa la evolución de control PID basado en eventos, cabe señalar que, para conocimiento del autor, no existen implementaciones de controladores PID industriales basadas en eventos y sólo hay pocas publicaciones con resultados apoyados por procesos reales como se puede revisar en [57],[58] y [59].

3.2.3 Estrategias de control basado en eventos

Últimamente se han dado investigaciones en relación con el muestreo y control basado en eventos, especialmente las relacionadas con los sistemas de control en red inalámbrica. En este trabajo, los agentes pueden poseer los eventos de entrada para recibir información entre ellos, así por ejemplo: el controlador recibe información de la entrada del sensor, el actuador recibe información desde el controlador, los eventos para activar la transmisión de información depende de la condición lógica utilizada para el muestreo por eventos, así la información de una muestra del sensor van al controlador o la información procesada por el algoritmo de control va al actuador permitiendo realizar la acción de control, los controladores pueden tener diferentes algoritmos de control. Para proporcionar un esquema básico de los PID basado en eventos, partimos de una estructura simplificada del sistema con control basado en eventos presentado en [30] , que se muestra en la Figura 3.4 y descrito a continuación. Se considera un sistema genérico, una planta, un detector de evento y un generador de entrada de control.

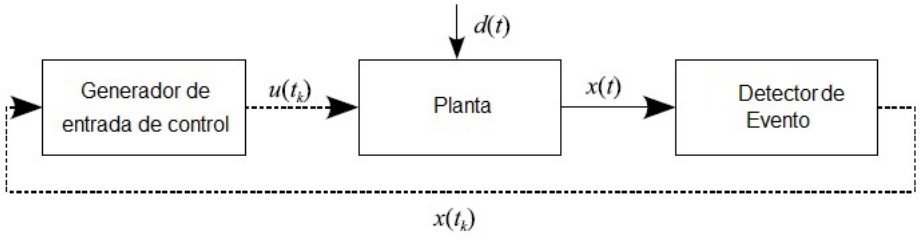


Figura 3.4 Diagrama de bloques simplificado de un WNCS.

El detector de evento envía los datos de salida del proceso, vector de estado para activar el generador de entrada de control cuando se produce un evento. El generador de entrada de control está inactivo entre dos eventos consecutivos y se activa mediante una invocación del detector de eventos (también es posible activarlo mediante una señal de tiempo, pero esto implicaría un estimador del estado de la planta hasta cuando exista nueva información disponible). Por lo tanto, el generador de entrada de control funciona en un lazo cerrado sólo a los eventos (retroalimentación). Del mismo modo, una vez que en el generador de entrada de control se ha producido la nueva acción de control, hay dos condiciones posibles que pueden ser utilizados para enviar los datos al actuador: cuando están disponibles o cuando una condición lógica situado en su salida es verdadera, es decir, una solución basada en eventos. Dependiendo de la condición basada en eventos ubicados en el detector de evento y el algoritmo de control incluido en el generador de entrada de control, se pueden distinguir tres categorías genéricas:

- En el primero, las acciones de control de realimentación se calculan cuando alguna variable varía más que un cierto valor (véase la Tabla 3.1 para diferentes condiciones basado en eventos). Por lo tanto, el generador de entrada de control produce una nueva acción de control cada vez que x(tk) cruza hacia arriba o abajo los niveles calculados de antemano (, -3Δ, -2Δ, -Δ, 0, Δ, 2Δ, 3Δ,...). La ley de control empleado en esta categoría es, en la mayoría de los casos, un PID en período variable de muestreo [43],[48],[60], pero hay enfoques basados en la retroalimentación de estado [61],[62], [63].

- En el segundo grupo, la acción de control se ajusta al máximo / mínimo, mientras que x (t) se queda fuera de una determinada banda muerta alrededor del set point ysp o una trayectoria de estado X, es decir, mientras que | y (tact) - ysp | ≥ Δ o X (tact) - Xs ≥ Δ ; en el mismo instante x (t) vuelve a entrar en la banda, surge un nuevo evento, y el generador de entrada de control produce una señal de control de alimentación directa para mover el proceso asintóticamente al valor de referencia o de la trayectoria [30],[64],[57]. La señal de control de alimentación directa se obtiene por medio de una retención generalizada que se pone a cero cada vez que x vuelve a entrar en la banda.

- Por último, es posible definir un tercer grupo de enfoques basados en eventos, donde se genera una acción de impulso para llevar el proceso al valor de consigna o al origen cuando X cruza los límites de la banda muerta; el resto del tiempo se pone a cero la acción de control [31], [65],[66].

Para resumir, el primer grupo mantiene una ley de control constante durante el período de muestreo no uniforme delimitada por dos eventos consecutivos, el segundo grupo es una ley de control variable, y el tercer grupo es uno impulsivo. Este proyecto se enmarca en el primer grupo

3.3 El controlador PID basado en eventos

Tomando en cuenta los diferentes tipos de políticas de control basadas en eventos que se han presentado anteriormente, ahora se aborda el tema de este capítulo. A continuación se da una

descripción de los controladores basados en eventos bajo la ley de control de PID (Figura 3.5) implementados en el generador de entrada de control. Su representación es:

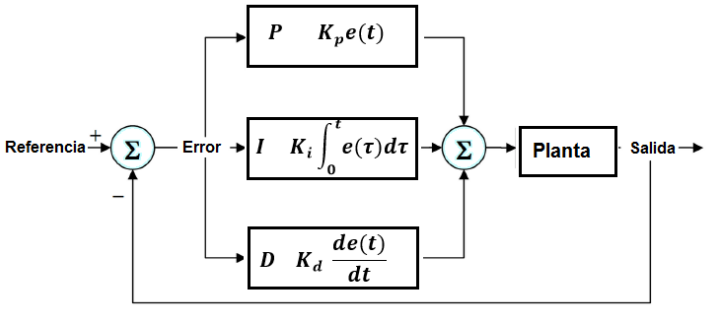


Figura 3.5 Diagrama de bloques y representación simplificada de un controlador PID

$$u(t) = K_p e(t) + \frac{K_p}{T_i}\int_0^t e(t)dt + K_p T_d \frac{de(t)}{dt}$$

Para la implementación digital de este controlador, se debe discretizar la ley de control. Esto se consigue normalmente utilizando diferencias hacia delante para el término integral de manera que es posible pre calcular la parte integral de tiempo tk + 1 y el uso de las diferencias hacia atrás para la parte de derivación. Por lo tanto, al indicar que el intervalo de muestreo como h_{nom}, el siguiente pseudo-código de este controlador PID es:

```
void pid_eventos(void)
{
  error = sp - Sdistancia;
  proporcional=error;
  integral = integral + ((error+error_anterior)/2.0)*(dt/1000.0);
  derivativo = (error - error_anterior)/(dt/1000.0);
  pid=(kp*proporcional)+(ki*integral) +(kd*derivativo);
  error_anterior=error;
}
```

Tenga en cuenta que el código anterior no está optimizado; por ejemplo, algunos coeficientes de la ley de control pueden ser pre-calculados para reducir el tiempo de cálculo. Además, el alcance de las variables se considera global. En todos los códigos del PID basado en eventos analizados en este capítulo se consideran que las comunicaciones entre el sensor, el detector de eventos, el generador de entrada de control, y el controlador son instantáneos; es decir, los retardos de comunicación y el tiempo de cálculo son insignificantes. En un enfoque basado en el evento teórico, el detector de evento debe examinar continuamente el sensor y evaluar la condición basada en eventos para invocar el generador de entrada de control cuando se convierte en verdad. Como se ha indicado antes, en las implementaciones digitales prácticas,

el detector de evento es disparado por tiempo con un periodo de muestreo fijo. En función de este periodo, el detector de eventos puede realizar una evaluación discreta o casi continua de la condición basada en eventos (es decir, un enfoque compuesto ó hibrido). Sin embargo, una frecuencia de muestreo superior induce mayor costo computacional.

El primer controlador PID basado en eventos reportados en la literatura se describe en [48]. El objetivo de este controlador es reducir el tiempo de la CPU para el cálculo de la ley de control el cual no proporciona una reducción global de la información intercambiada en el lazo de control. En realidad, como ocurre en las realizaciones prácticas de los enfoques basados en eventos, este controlador es una aproximación híbrida entre un tiempo y un controlador basado en eventos. La activación del detector de eventos es la base del tiempo con el período de h_{nom}, pero la invocación del generador de entrada de control no lo es. Si se cumple la condición basada en eventos en el detector de eventos entonces se invoca una nueva acción de control dada por el generador de entrada de control. Tenga en cuenta que si la frecuencia de activación del detector de eventos se aumenta al máximo en una implementación real, este controlador se convierte en un enfoque basado en eventos cuasi-pura, es decir en teoría continuo.

Como se ha descrito, en cada período de muestreo nominal h_{nom} el controlador lee la muestra del sensor, y el detector de evento evalúa la condición basada en eventos con esta información. Si la condición es verdadera, se produce un evento, y el generador de entrada de control calculará la acción de control usando una ley PID con muestreo periódico de la variable de entrada. La condición lógica es una función del error relativo. Por lo tanto, la condición es verdadera cuando el valor absoluto de la diferencia entre el valor actual del error $e(t_{act})$ y el valor del error anterior $e(t_{last})$ es mayor que un umbral predefinido Δ_e. Por lo tanto, un error de muestreo basado en send-on-delta se aplica al controlador. Además, el detector de evento

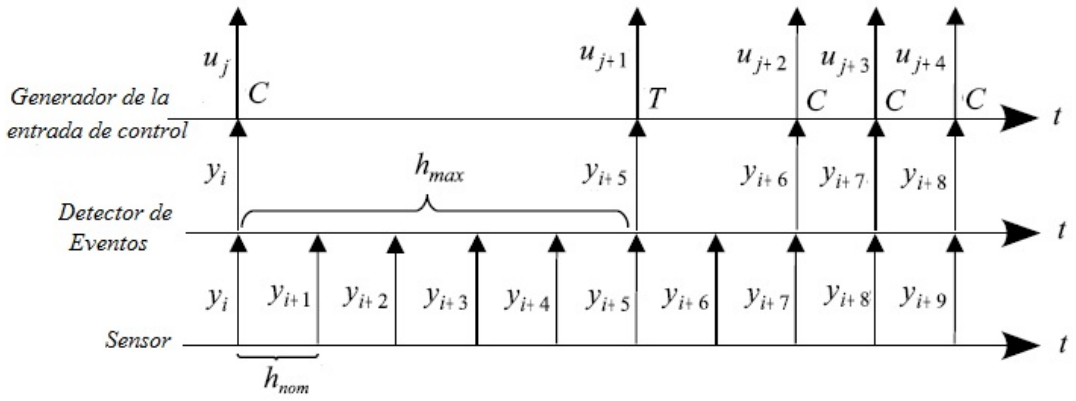


Figura 3.6 Intercambio de información en el enfoque PID basado en eventos.

incluye una condición de seguridad basado en el tiempo para garantizar un intervalo de tiempo máximo entre eventos consecutivos h_{max}. De esta manera, un nuevo valor de la señal de control se ve obligada a ser calculada cuando el tiempo transcurrido $h_{act} = t_{act} - t_{last}$ excede un límite dado h_{max}. Formalmente, el estado general basado en eventos se puede escribir como:

$$(|e(t_{act}) - e(t_{last})| \geq \Delta e) \quad or \quad (h_{act} \geq h_{max}) \tag{3.3}$$

El patrón de muestreo que produce la condición basada en eventos se denota como muestreo localmente no uniforme descrito en [61]. La condición basada en eventos utiliza un tiempo de muestreo h_{max} uniforme cuando el error no cambia u oscila sin problemas, pero utiliza un período de muestreo h_{act} no uniforme si la variación de error actual se mantiene localmente debajo de Δe antes de que el temporizador fijado a h_{max} se dispare. Cabe señalar que el período de muestreo no uniforme h_{act} es siempre un múltiplo de h_{nom}. La figura 3.6 presenta un ejemplo de la información intercambiada entre los agentes de control utilizando la condición basada en eventos definida en la ecuación (3.3). El generador de entrada de control se ejecuta sólo si se detecta el evento, lo que ahorra recursos de cálculo si no hay necesidad de actualizar la variable de control (por ejemplo, cuando el proceso está en el estado estacionario), en este caso estaríamos hablando de un PID asíncrono, pero si el tiempo cuando ocurre un evento aumenta y el generador de eventos no envia información el controlador deberá estimar una salida en base a un predictor que obedezca a la tendencia o a la función de estado de la planta.

3.4 Alternativas para el controlador PID basado en eventos

El algoritmo PID anterior basado en el enfoque híbrido en donde el detector de eventos actúa periódicamente y en base a una condición lógica o tiempo, dispara el dato al generador de la entrada de control en el cual se realiza el cálculo del algoritmo PID, en este caso se dice que es un PID asíncrono debido a que el dato de salida del PID sólo se calcula cuando se activa la condición lógica, en este trabajo también se ha implementado otras alternativas como:

a) El PID semi síncrono en donde para el cálculo de la salida del PID, se periodiza el intervalo de tiempo en que llega una nueva actualización de datos enviada por el detector de eventos, es decir el intervalo de tiempo entre eventos que llega al generador de la entrada de control se divide en un delta de tiempo entero en el cual se calcula el PID, pero existirá intervalos de tiempo en los que no siempre se dará un delta entero, es entonces en donde el PID generará su salida en base a la lectura actual de la entrada proporcionada por el detector de eventos.

```
// Cálculo de la función PID Semi Sincrono
void pid_eventos(void){
 error = sp - Sdistancia;
 proporcional=error;
 integral = integral + ((error+error_anterior)/2.0)*(t_semisincrono/1000.0);
 derivativo = (error - error_anterior)/(t_semisincrono/1000.0);
 pid=(kp*proporcional)+(ki*integral) +(kd*derivativo);
 error_anterior=error;
t_semisincrono=0;
}
```

b) el PID síncrono o periódico que en esencia es similar al semi síncrono con la diferencia que el cálculo de la salida del PID siempre se dará en base a un período delta seleccionado sin importar el tiempo de llegada del nuevo dato enviado por el generador de función, cuya información sirve para actualizar el dato para cálculo del PID.

```
// Cálculo de la función PID Sincrono
void pid_eventos(void){
error = sp - Sdistancia; //
proporcional=error;
integral = integral + ((error+error_anterior)/2.0)*(0.03);
derivativo = (error - error_anterior)/(0.03);
pid=(kp*proporcional)+(ki*integral) +(kd*derivativo);
error_anterior=error;
}
```

En la literatura también existen otras alternativas que pretenden minimizar el cálculo computacional. El uso de un estimador en función a las variables de estado de la planta realizando cálculos que predigan el comportamiento del PID es un medio eficaz de reducir el coste computacional. Sin embargo, algunas mejoras del algoritmo PID se describen en [60] para reducir el coste computacional y lograr un rendimiento similar al de un controlador PID convencional disparado por tiempo.

La minimización del coste computacional implica minimizar el número de eventos activados. La solución propuesta en [60] se basa en el cálculo de la señal de control sólo cuando la medición es demasiado lejos del valor de referencia. En la concepción original, la condición basada en eventos se basa en el error relativo y utiliza un detector de evento de disparo con h_{nom}. Ahora, la condición lógica del detector se basa en el error absoluto

$$|e(t_{act})| \geq \Delta e \qquad (3.4)$$

Para determinar cuando ocurra el evento, se proponen sustituir el detector disparador de tiempo por uno continuo para enviar una petición en el momento en que el error supera el umbral. En implementaciones reales, el detector de evento de tiempo continuo se emplea mediante la reducción del valor de h_{nom} (el enfoque hibrido) tanto como sea posible para simular un muestreo continuo.

Con esta nueva condición basada en caso de que el número de eventos se reduce al mínimo durante los intervalos de estado estable cuando el error permanece por debajo del umbral; Sin embargo, durante el período transitorio, el número de eventos crecerá. Para reducir el número de eventos, se introduce una nueva condición lógica. Se compone de la adición de un tiempo mínimo entre eventos, es decir, $h_{act} > h_{min}$, al detector de evento. De acuerdo con [64], este intervalo de tiempo mínimo podría ser elegido como el período de muestreo discreto correspondiente a un controlador de tiempo por alarma. Otra opción es encontrar un compromiso entre la reducción de h_{nom} y el número de eventos producidos. Otra de las mejoras en esta versión de bajo cálculo del algoritmo basado en eventos es la reducción del

error absoluto durante los intervalos de estado estable. Con la implementación actual, e(t) permanece por debajo de Δe, no produce acciones de control para reducir el error. En los pseudo-códigos, anteriores se puede apreciar que el detector de evento que se evalúa cada h_{nom}, y es una combinación de condiciones lógicas con dos propósitos diferentes, envía datos al generador de entrada de control que es en donde se ejecuta el PID se invoca con diferente periodicidad en función de la dinámica del proceso (en el estado estacionario o transitorio) y el tiempo transcurrido desde el último evento. Simulaciones en [64], muestran que el comportamiento de este controlador es mejor que la de los otros descritos en esta sección, con un rendimiento muy cerca de un homólogo PID disparado por tiempo utilizando el mismo conjunto de parámetros.

3.5 El objetivo del Control basado en eventos

A continuación se sintetiza el modelo basado en eventos presentado en [1], donde se introduce un enfoque de estado realimentado para el control basado en eventos con el objetivo de diseñar e imitar el comportamiento de un sistema de lazo cerrado continuo dado, el lazo de control basado en eventos como se indica en la Figura 3.4 el controlador está incorporado en el generador de entrada de control.

La planta es representada por el modelo de espacio de estados lineal

$$\dot{x}(t) = Ax(t) + Bu(t) + Ed(t)\,; \quad x(0) = xo \qquad (3.5)$$

$$y(t) = Cx(t) \qquad (3.6)$$

Donde $x \in \mathbb{R}^n$ denota el estado del sistema con el valor inicial x_o , $u \in \mathbb{R}^m$; $y \in \mathbb{R}^r$ son las entradas o salidas medidas, respectivamente y $d \in \mathbb{R}^t$ representa las perturbaciones externas.

El par (A, B) se asume controlable y el límite de la perturbación $d(t)$ será limitado como sigue:

$$\|d(t) \leq d_{max}\| \qquad (3.7)$$

La notación $||x||$ y $||A||$ denota un vector normal arbitrario o inducido de la matriz normal, y el valor absoluto es denotado por $|x|$. La expresión $||x(t)||$ denota un vector normal al tiempo t Si el futuro asumimos que:

• La dinámica de la planta se conocen con exactitud,

• El estado $x(t)$ es medible, y

• El intercambio de información entre el detector de eventos y el generador de entrada de control es instantánea y no impone restricciones a la información que debe transmitirse a los tiempos de eventos.

Por lo tanto, la razón para comunicar información a través de las flechas de trazos en la Fig. 2.3 se da principalmente por la situación de que la perturbación $d(t)$ ha provocado un comportamiento intolerable de la salida de control $y(t)$ o el estado de la planta $x(t)$.

Idea principal. Como característica principal del esquema propuesto, el detector de eventos utiliza un modelo del circuito de regulación continua para comparar los estados actuales de la planta $x(t)$ con el estado deseado que se produce en el sistema de lazo cerrado continuo. Si la diferencia entre los dos estados excede el límite superior $\bar{e}$, se activa un evento y actualiza el estado $x(t_k)$ para ser transmitido al generador de entrada de control. Como hecho importante adicional, el generador de entrada de control incorpora una función que depende del proceso para determinar el futuro de la entrada de control $u(t), (t \geq t_k)$. Se muestra que el lazo de control basado en eventos de estas características tiene las siguientes propiedades:

• El estado $x(t)$ del lazo de realimentación de estado basado en eventos es finalmente acotado en el sentido de que sigue siendo, para todos los tiempos t, en una zona acotada Ω_e del estado deseado $x_{CT}(t)$ del lazo de realimentación de estado continuo.

• La comunicación a través del canal de realimentación en el lazo de control basado en eventos es limitada y depende explícitamente de la perturbación $d(t)$.

• Tanto la exactitud de la aproximación del comportamiento del lazo de realimentación de estado continuo y el intervalo de tiempo mínimo entre dos eventos consecutivos (mínimo tiempo entre eventos) se puede ajustar cambiando el e umbral del detector de eventos con el fin de adaptar el estado de retroalimentación del lazo basado en eventos a las necesidades requeridas.

3.5.1 Realimentación de estado continúo

Las principales propiedades del lazo de realimentación de estado continuo que se utiliza como el sistema de referencia para evaluar el comportamiento del lazo de realimentación de estado basado en eventos. La planta 3.5, 3.6, junto con la retroalimentación de estado

$$u(t) = -Kx(t) \qquad (3.8)$$

Proporciona el sistema de circuito cerrado continuo

$$\dot{x}_{CT}(t) = (A - BK)x_{CT}(t) + Ed(t) \quad ; x_{CT}(0) = xo \qquad (3.9)$$

$$y_{CT}(t) = Cx_{CT}(t) \qquad (3.10)$$

El índice "CT" se utiliza para distinguir las señales de este modelo de las señales correspondientes del lazo de control basado en eventos considerado más adelante. Se supone que la matriz de realimentación de estado K se ha diseñado de manera que la matriz $\bar{A}$ es Hurwitz y el sistema de lazo cerrado tiene propiedades de atenuación de perturbaciones deseadas.

Como es $\bar{A}$ es Hurwitz y la perturbación $d(t)$ se supone que está limitada de acuerdo con la Ecuación 3.7, el estado del lazo de realimentación de estado continuo 3.9, 3.10 es GUUB de acuerdo con la siguiente definición:

Definición.- La solución de $x(t)$ del lazo de control continua 3.9, 3.10 se dice que es globalmente uniforme en última instancia delimitada (GUUB) si para cada $x_0 \epsilon \mathbb{R}^n$ existe una constante positiva p y un tiempo t tal que mantenga:

$$x(t) \epsilon \Omega_t = \{x : ||x|| \leq p\}, \qquad \forall t \geq \bar{t}$$

Entonces se dice que el lazo de control continuo 3.9, 3.10 está en última instancia, acotada. Para el lazo de control lineal continua 3.9, 3.10 el estado $x(t)$ es GUUB si la matriz $\bar{A}$ es de Hurwitz y la perturbación $d(t)$ es acotada.

Comportamiento del Estado-Realimentado de lazo continúo. La entrada de control generada por el controlador de estado-realimentado 3.5 está dada por:

$$u(t) = -Ke^{\bar{A}t}x_0 - \int_0^t Ke^{\bar{A}(t-\tau)}Ed(\tau)d\tau \qquad (3.11)$$

Esta ecuación muestra que la entrada $u(t)$ no sólo depende del estado inicia x_0 sino también en la entrada de perturbación $d(t)$. En el ajuste de control basado en eventos, este aspecto es importante. Si en el momento t_k el estado $x(t_K)$ es comunicado al generador de entrada de control, el generador de entrada de control es capaz de determinar la misma entrada de control
$u(t_K) = -Kx(t_K)$ como un controlador de estado de retroalimentación continua. Sin embargo, para todos los futuros tiempos $t > t_k$, el generador de entrada de control tiene que saber la perturbación $d(t)$ para $t > t_k$:

$$u(t) = -Ke^{\bar{A}(t-t_k)}x_{CT}(t_k) - \int_{t_k}^t Ke^{\bar{A}(t-\tau)}Ed(\tau)d\tau \quad , \quad t \geq t_k \qquad (3.12)$$

Este análisis pone de manifiesto dos hechos importantes:

• El control de realimentación de estado continuo 3.8 obtiene la información sobre la perturbación actual implícitamente por la comunicación continua del estado actual $x_{CT}(t)$.

• Cualquier regeneración sin la comunicación continua tiene que hacer suposiciones acerca de la perturbación a ser atenuadas. A menos que la alteración es medible, cualquier regeneración discontinua no puede tener el mismo rendimiento que el circuito de retroalimentación con la comunicación continua.

La idea principal del enfoque del estado de retroalimentación basado en eventos discute que lo siguiente es reemplazar la realimentación de estado continuo 3.8 por un controlador basado en eventos de modo que el estado x(t) del lazo de realimentación de estado basados en

eventos , para todos los tiempos t, en el vecindario $\Omega_e(x_{CT}(t))$ del estado deseado $x_{CT}(t)$ del lazo continuo de realimentación de estado 3.9, 3.10.

3.5.2 Realimentación de estados basado en eventos

Generador de Entrada de control. Una consecuencia directa del análisis en la sección anterior es el hecho de que por el momento t = tk la planta 3.5, 3.6 con la entrada de control 3.8 se comporta exactamente como el lazo de control continua 3.9, 3.10. Si el generador de entrada de control utiliza la ecuación 3.12 para determinar la entrada de control para t = tk, entonces se obtiene el mejor rendimiento posible. Para activar el generador de entrada de control y para utilizar esta ecuación, el estado tiene que ser medido y comunicado al generador de entrada de control, y se tiene que realizar un supuesto acerca de la perturbación.

En lo que sigue, el generador de entrada de control asume que la perturbación es constante

$$d(t) = \widehat{d_K} \quad para \ t \geq t_K$$

cuando se conoce la magnitud $\widehat{d_K}$. Por lo tanto, se utiliza la ecuación

$$u(t) = -Ke^{\bar{A}(t-t_k)}x(t_k) - K\bar{A}^{-1}\left(e^{\bar{A}(t-\tau)} - I_n\right)E\,\hat{d}_k, \quad t \geq t_k \qquad (3.13)$$

que sigue directamente de la ecuación. 3.12 para las perturbaciones constantes, hasta que se obtiene la información siguiente x (tk + 1). In denota la matriz identidad de tamaño n.

El generador de entrada de control determina la entrada 3.13 por medio de un modelo del sistema de lazo cerrado continuo 3.8

$$\dot{x}_s(t) = \bar{A}x_s(t) + E\hat{d}_k(t) \quad x_s(t_k{}^+) \ , t \geq t_k \qquad (3.14)$$

$$u(t) = -Kx_s(t) \qquad (3.15)$$

Aquí, xs se utiliza para denotar el estado del generador de entrada de control. Tenga en cuenta que la señal u (t) obtenido por la ecuación 3.13 es la misma que la solución de 3.14, 3.15.

El tiempo $t_k{}^+$ indica la actualización del modelo de estados Xs con las medidas estado x (tk), que el generador de entrada de control recibe del detector de eventos en el tiempo tk del evento. La figura 3.7 muestra el diagrama de bloques del generador de entrada de control. La forma adecuada para determinar el tiempo de evento tk y la estimación de la perturbación se presentan más adelante.

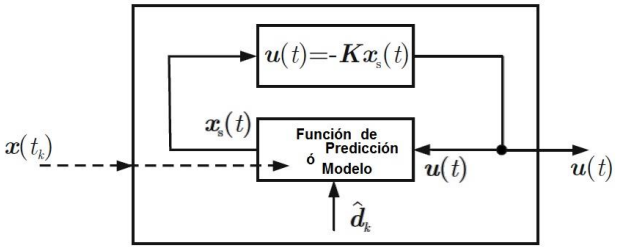


Figura 3.7 Generador de la entrada de control.

Comportamiento del Estado del lazo de realimentación basado en eventos. El análisis de este párrafo es válida para los generadores de eventos arbitrarios y métodos arbitrarios para estimar la magnitud de la perturbación dk. Se investiga el comportamiento del lazo de control basado en eventos en el intervalo de tiempo [tk, tk + 1) entre los consecutivos tiempos de eventos tk y tk + 1.

La planta 3.5, 3.6, junto con el generador de entrada de control 3.14, 3.15 se describe para el período de tiempo [tk, tk + 1) por el modelo de espacio de estado

$$\begin{pmatrix} \dot{x}(t) \\ \dot{x}_s(t) \end{pmatrix} = \begin{pmatrix} A & -BK \\ 0 & \bar{A} \end{pmatrix} \begin{pmatrix} x(t) \\ x_s(t) \end{pmatrix} + \begin{pmatrix} E \\ 0 \end{pmatrix} d(t) + \begin{pmatrix} 0 \\ E \end{pmatrix} \hat{d}_k \qquad (3.16)$$

$$\begin{pmatrix} \dot{x}(t_k^+) \\ \dot{x}_s(t_k^+) \end{pmatrix} = \begin{pmatrix} x(t_k) \\ x_s(t_k) \end{pmatrix} \qquad (3.17)$$

$$y(t) = (C \quad 0) \begin{pmatrix} x(t) \\ x_s(t) \end{pmatrix} \qquad (3.18)$$

Este modelo tiene en cuenta que el sistema de lazo cerrado está sujeta a la perturbación d(t), mientras que el generador de entrada de control utiliza la constante dk estimación de perturbación. La expresión xi (t + k) = xi (tk) se utiliza para indicar explícitamente que el estado respectivo no se cambia en el instante de tiempo correspondiente.

Detector de eventos. Los eventos se generan mediante la comparación de la medida de la trayectoria de estado $x(t)$ con la trayectoria de estado $x_s(t)$ que se producirían en el lazo de realimentación de estado continuo para la perturbación constante $d(t) = d_k$. Como el estado $x_s(t)$ es determinado en función de la ecuación 3.14 representa la señal deseada de referencia, la medida de estado x (t) debe ser mantenida en el entorno

$$\Omega_s(x_s(t)) = \{x: \|x - x_s(t)\| \leq \bar{e}\} \qquad (3.19)$$

de este estado con tamaño ajustable $\bar{e}$.

El detector de eventos desencadena un evento siempre que la diferencia entre las medidas del estado de la planta $x(t)$ y la referencia de estado $x_s(t)$ alcanza el evento umbral $\bar{e}$

$$\|x - x_s(t)\| = \bar{e} \qquad (3.20)$$

En este momento t, que denota el tiempo tk, la información de estado x(tk) se comunica al generador de entrada de control.

Con el fin de evitar una transmisión continua del estado xs(t) del generador de entrada de control al detector de eventos, una copia del generador de entrada de control está incluido en el detector de eventos de modo que el generador de eventos puede determinar el estado xs(t) por medio de la ecuación 3.14.

Resumen de los componentes. El lazo de realimentación de estado basado en eventos tiene la estructura representada en la Figura 3.8 Tiene los siguientes componentes:

• la planta 3.5, 3.6

• el generador de entrada de control 3.14, 3.15 y

• el detector de eventos que incluye una copia del generador de entrada de control 3.14, 3.15, y determina el tiempo del evento tk de acuerdo con la ecuación 3.20.

En los tiempos de eventos tk, (k = 0, 1, 2, ...) la información de estado medido x(tk) se envía desde el detector de eventos hacia el generador de entrada de control y se utiliza allí, así como en el detector de eventos es para actualizar el estado del modelo xs de acuerdo con xs (t + k) = x (tk) y determinar la nueva estimación de la perturbación dk. Dado el supuesto de la transmisión de datos se lleva a cabo en muy poco tiempo, los modelos del generador de entrada de control y el detector de eventos trabajan sincrónicamente.

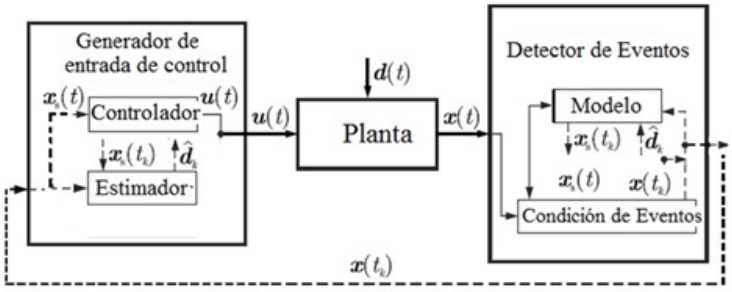


Figura 3.8 Lazo de control de realimentación de estado basado en eventos

Propiedades principales del lazo de realimentación de estados basado en eventos.- Las propiedades centrales a investigar al considerar el control basado en eventos concierne a la estabilidad y la comunicación a través del enlace de retroalimentación. Los principales resultados del análisis posterior son los siguientes:

• El estado x (t) del lazo de control basado en eventos es GUUB y no existe un límite superior en el error de aproximación en términos de emular el comportamiento del lazo de realimentación de estado continuo.

• Existe un límite inferior en el tiempo mínimo entre eventos.

• Si las perturbaciones son suficientemente pequeñas, no se genera ningún evento para t> 0

Comparación entre lazo de realimentación de estados basado en eventos y Lazo continuo

El siguiente teorema compara el lazo de realimentación de estados basado en eventos con el lazo de realimentación de estado continuo.

Teorema.- [67] La diferencia $e(t) = x(t) - x_{CT}(t)$ entre el estado $x(t)$ del lazo de realimentación de estado basado en eventos 3.5, 3.6, 3.14, 3.15, 3.20 y el estado $x_{CT}(t)$ del lazo de realimentación de estados continuo 3.9, 3.10 está acotada superiormente por:

$$\|e(t)\| \leq e_{max} = \bar{e} \int_0^{\infty} \left\| e^{\bar{A}\tau} BK \right\| d\tau \qquad (3.21)$$

Este teorema demuestra que el controlador basado en eventos se puede utilizar para imitar un sistema de retroalimentación de estado continuo con precisión arbitraria por elegir en consecuencia del umbral $\bar{e}$ de evento. Se puede utilizar para determinar cada límite a la aproximación error $||e(t)||$ superior tolerable, el umbral de eventos $\bar{e}$. El precio de una precisión más alta (e_{max} pequeña) es una comunicación más frecuente entre el detector de eventos y el generador de entrada de control. El estado $x(t)$ permanece en el conjunto.

$$x(t) \in \Omega_e\big(x_{CT}(t)\big) = \{x: \|x - x_{CT}(t)\| \leq e_{max}\} \qquad (3.22)$$

que describe una vecindad limitada del estado $x_{CT}(t)$ del lazo de realimentación de estado continuo para todos los tiempos t, por lo tanto, como el estado $x_{CT}(t)$ del lazo de realimentación de estado continuo es GUUB, el estado x(t) del lazo de realimentación de estado basado en eventos también es GUUB.

3.6 El uso de predictores en el control basado en eventos

En los enfoques anteriores, el detector de evento fue considerado disparado por tiempo (con un período de muestreo rápido) y el generador de entrada de control basado en evento. Además, se examinó la creación de problemas de tiempo de la naturaleza basada en eventos del generador de entrada de control; estos problemas incluyen períodos variables de muestreo, los grandes errores de aproximación en los términos integral y derivativa. Tales inconvenientes reducen la calidad del control y una heurística deben ser introducidas en la ley de control para resolverlos.

Una posible configuración que elimina los problemas de tiempo es que ambos elementos son disparados por tiempo pero ambos envían la información a su respectivo vecino adyacente cuando se cumple alguna condición basada en eventos. De este modo, el detector de evento evalúa cada h_s su estado basado en eventos (por ejemplo, | y (t_{act}) - y (t_{last}) | ≥ Δs), y el generador de entrada de control produce cada h_c una acción de control que se envía al actuador.

Como no hay retrasos en la comunicación y el tiempo de cálculo es insignificante o inferior a los períodos de muestreo h_s y h_c nominales, no hay problemas de tiempo en este enfoque. Sin embargo, el problema con esta configuración es que el generador de entrada de control no podía recibir el estado del proceso del sensor cada hc, y esta información debe ser obtenido mediante el empleo de un observador (Kalman, Luenberger) en el controlador para predecir el

valor del proceso de variable en cada intervalo de muestreo h_c . En este enfoque, el observador en realidad se ejecuta en un lazo abierto entre las muestras, y la variable de proceso se calcula utilizando el último estado recibido. La estimación de estado se actualiza cada vez que llega una nueva señal del sensor. Es evidente que la eficacia de la metodología depende en gran medida la exactitud del observador (lo que significa que los aumentos de esfuerzo de diseño de manera significativa) y que la presencia de perturbaciones pueden implicar una disminución significativa en el rendimiento. Una solución parcial a estos problemas se puede lograr mediante el empleo de un observador de orden reducido (primer orden), como en [68] ó mediante una función que describa el proceso, tal como se implementó en este proyecto.

Un enfoque similar puede ser implementado por la aplicación de un predictor de espacio de estado de primer orden de la planta para estimar la salida de la planta en un lazo abierto en ausencia de información actualizada desde el detector de evento. El predictor de primer orden simplificado puede ser escrita como:

$$\hat{x}(t) = A_s\hat{x}(t_{k+1}) + B_s u(t_{last}) \qquad (3.23)$$

$$y(t_k) = C_s x(t_k) \qquad (3.24)$$

donde As, Bs, y Cs son matrices de primer orden y $u(t_{last})$ es la última acción de control enviado al actuador. En este enfoque, el predictor es parte del generador de entrada de control, y por lo tanto, se evalúa cada hc con la información disponible. Al inicio de cada evaluación, se comprueba la llegada de un nuevo valor de la salida de proceso enviada desde el detector de eventos. Si esto ocurre, la ley PID calcula la acción de control. Sí hay llegadas, el controlador utiliza la última predicción del resultado del proceso. Después de eso, el predictor se evalúa con la nueva información, y la predicción de salida de proceso está listo para la siguiente invocación del detector de control.

El uso del predictor exige un pequeño h_c para reducir los errores de estimación, aunque esto produce un gran número de acciones de control que pueden saturar el enlace de comunicación o producir desgaste en los actuadores. Para mantener el tráfico de comunicación bajo, este enfoque introduce una condición basada en eventos en el generador de entrada de control para evaluar si vale la pena transmitir la nueva acción de control al actuador. La condición lógica es:

$$|u(t_{act}) - u(t_{last})| \geq \Delta_c \qquad (3.25)$$

3.7 Sintonización de los controladores PID

Pese a los avances en la teoría de control en las últimas décadas, los controladores PID ocupan un lugar dominante en la industria. Una encuesta realizada, describe que más del 97% de los controladores son PID descrito en [69], por lo que es llamado "el controlador de industrias de proceso por defecto" [70]. Esto es debido al hecho de que, dada la suficiente experiencia y tiempo, la estructura de un controlador PID es adaptable y fácil de ser sintonizado sin la necesidad de una descripción de la planta, solamente mediante la observación de la respuesta del sistema.

Pero, hay que señalar que los enfoques que se están utilizando del modelo PID son más seguros y más robustos en términos de estabilidad y rendimiento en una amplia gama de frecuencias. Además de ser ajustable y poseer gran confianza de la industria, los controladores PID son capaces de satisfacer tanto los requisitos de respuesta transitoria y de estado estacionario de muchos procesos. No hay nada que ganar con el uso de un controlador más complejo para las plantas cuyas dinámicas dominantes son de segundo orden, que cubre la mayor parte de las plantas industriales, a continuación se describen los métodos de sintonización más utilizados.

3.7.1 Clásica sintonización del PID

Consciente de la finalidad de los tres términos, junto con un par de reglas populares de pulgar puede ser adecuada para ajustar el controlador, mediante la observación de la respuesta del sistema. Sin embargo, es difícil juzgar que el controlador PID sintonizado de esa manera es el ideal. Además de esto, siguiendo las reglas de oro no siempre son apropiados. En [71], se ha demostrado que para algunas plantas, aumentando el término derivado se intensifica la ganancia a un nivel en el que el sistema se vuelve inestable. Esto es contrario a la creencia general de que la acción derivada aumenta la estabilidad. En el mismo estudio, este tipo de malentendidos y generalizaciones han dado como la razón al hecho de que la mayoría de los controladores PID en la industria tienen parte de derivación desconectada, por lo que no se utiliza toda la funcionalidad del controlador.

Debido a estas razones, es esencial disponer de una forma sistemática la sintonía de los controladores PID. Afortunadamente, en las últimas décadas numerosos enfoques novedosos para ajustar los parámetros PID se han propuesto. La mayoría de estos métodos tienen como objetivos de diseño

• La estabilidad

• La atenuación de perturbaciones de carga, que se puede medir utilizando la Integral del error absoluto (IAE)

$$IAE = \int_0^\infty |e(t)|dt \qquad (3.26)$$

• La respuesta transitoria que, junto con IAE, también se puede medir por los siguientes índices de rendimiento:

$$ITAE = \int_0^\infty t|e(t)|dt \qquad (3.27)$$

$$ISE = \int_0^\infty e^2(t)dt \qquad (3.28)$$

$$ITSE = \int_0^\infty te^2(t)dt \qquad (3.29)$$

A continuación se presentan algunos de los métodos mencionados para sintonizar controladores PID. Para un más detalle se pueden encontrar en [72].

3.7.2 Método del Ziegler-Nichols

El método de sintonización PID presentado por Ziegler y Nichols [73] se basa en la respuesta paso de lazo abierto del sistema. Se utiliza el hecho de que muchos sistemas en la industria de procesos pueden ser aproximadas por un retardo de primer orden, más un retardo de tiempo como:

$$G_p(s) = \frac{\alpha}{sL} e^{-sL} \qquad (3.30)$$

Ziegler y Nichols también introdujeron un método basado en .la respuesta de frecuencia del sistema de circuito cerrado bajo control proporcional puro. En este caso, la ganancia se incrementa hasta que el sistema de circuito cerrado se vuelve críticamente estable. En este punto, la ganancia máxima, se registra junto con el correspondiente período de oscilación, conocido como el período final. Sobre la base de estos valores Ziegler y Nichols calculan los parámetros de ajuste dados en tablas como se indica en [72]. Los métodos de ZN fueron diseñados para dar buenas respuestas ante perturbaciones. Un criterio de amplitud de amortiguación se utilizó en el diseño que da un coeficiente de amortiguamiento cerca de 0,2. Esto no es satisfactorio para muchos sistemas, ya que no da márgenes de fase y de ganancia satisfactorios. La sensibilidad máxima también es grande, dando los sistemas sensibles a las variaciones de los parámetros. Además, los métodos ZN no son fáciles de aplicar en su forma original en las plantas de trabajo.

3.7.3 Asignación de polos

El método analítico de asignación de polos se utiliza sobre todo cuando el sistema en cuestión es de orden inferior. Un enfoque común es adoptar un modelo de segundo orden y luego especificar un factor de amortiguamiento y la frecuencia natural deseada para el sistema. Estas especificaciones a continuación, se pueden cumplir mediante la localización de los dos polos del sistema en las posiciones que dan los requerimientos de funcionamiento de lazo cerrado. Como un ejemplo, la ecuación característica para un sistema aproximado por un modelo de primer orden.

$$G_p(s) = \frac{K_p}{1 + sT_1} \qquad (3.31)$$

bajo el control PI tomará la forma:

$$s^2 + s\left(\frac{1}{T_1} + \frac{K_p k_p}{T_1}\right) + \frac{K_p k_p}{T_1 T_i} = 0 \qquad (3.32)$$

Esto se puede comparar con el modelo general de segundo orden

$$s^2 + 2\zeta\omega s + \omega^2 = 0 \qquad (3.33)$$

y así obtener

$$K_p = \frac{2\zeta\omega T_1 - 1}{k} \quad , \quad T_i = \frac{2\zeta\omega T_1 - 1}{\omega^2 T_1} \tag{3.34}$$

En el caso en que se utiliza un sistema de segundo orden de la forma siguiente:

$$G_p(s) = \frac{K_p}{(1 + sT_1)(1 + sT_2)} \tag{3.35}$$

Un controlador PID de la forma

$$G_c(s) = \frac{K_p(1 + sT_i + s^2 T_i T_d)}{sT_i} \tag{3.36}$$

Se puede colocar arbitrariamente todos los polos en lazo cerrado. La ecuación característica del sistema se convierte entonces. Esto puede ser comparado con el siguiente, ecuación característica de tercer orden en general:

$$(s + \alpha\omega)(s^2 + 2\zeta\omega s + \omega^2) = 0 \tag{3.37}$$

El diseño de polo dominante se basa en el posicionamiento de polos dominantes del sistema en el plano complejo. Por ejemplo, tomando la función de transferencia de un sistema de realimentación unitaria.

$$G_{CL}(s) = \frac{G_c(s)G_p(s)}{1 + G_c(s)G_p(s)} \tag{3.38}$$

A continuación se puede fácilmente encontrar los polos y ceros del sistema de circuito cerrado resultante. En muchos casos, la dinámica del sistema dominante puede ser aproximada por la configuración polo-zero simple. El par de polos P1, P2 son conocidos como los polos dominantes. Polos y ceros que tienen partes reales mucho más negativa que las de los polos dominantes tienen poca influencia en la respuesta general del sistema. Para un controlador PI, como un ejemplo, los polos P1, P2 pueden ser:

$$P_1 = -\zeta\omega_0 + i\omega\sqrt{1 - \zeta^2} \quad , \quad P_2 = -\zeta\omega_0 - i\omega\sqrt{1 - \zeta^2} \tag{3.39}$$

Estos dan un sistema con la amortiguación requerida ζ , y la velocidad de respuesta dada por ω_0 .

3.7.4 Los algoritmos genéticos para la sintonización PID

Los algoritmos genéticos son una zona de rápida expansión en el diseño de sistemas de control. Un algoritmo de ajuste genética comienza generalmente sin conocimiento de la solución correcta y depende de las respuestas de su medio ambiente para dar un resultado aceptable. Se ha demostrado que los algoritmos genéticos son capaces de localizar regiones óptimas en dominios complejos evitando las dificultades, o incluso resultados erróneos en algunos casos, asociados con los métodos de descenso de gradiente y con sistemas de alto

orden. Para obtener los parámetros de ajuste del PID por lo general se tiene que minimizar un índice de desempeño. Esto, en la mayoría de los casos, es uno de los siguientes: 3.26, 3.27, 3.28 y 3.29.

El algoritmo genético requiere una población de aproximaciones iniciales, que pueden ser al azar, para iniciar la búsqueda. Luego, el algoritmo comprueba la aptitud de cada individuo (o cromosomas), a continuación se sigue un proceso de selección sigue, donde se eligen los individuos más aptos. Los individuos restantes son seleccionados probabilísticamente. Los individuos seleccionados se utilizan para producir la siguiente población, y el proceso se repite entonces hasta que se cumplan los requisitos de diseño. Este método es aplicable a una amplia gama de modelos de sistemas debido a su adaptabilidad. Los sistemas de orden superior no presentan un problema con este procedimiento de ajuste.

CAPITULO 4

4 PID basado en eventos para el sistema B-B

4.1 El modelo del sistema Beam & Ball

El sistema beam-ball (BB) es un sistema dinámico clásico el cual se ha utilizado para indicar teorías de control y aplicar técnicas, la importancia del sistema BB es porque es un sistema simple, no lineal e inestable en lazo abierto. Consiste en equilibrar la bola en una barra en una posición deseada, para estabilizar la bola se necesita un sistema de control realimentado, para lo cual se debe medir la posición de la bola y ajustar el motor que sujeta la barra a un ángulo requerido, que vaya acorde al diseño deseado.

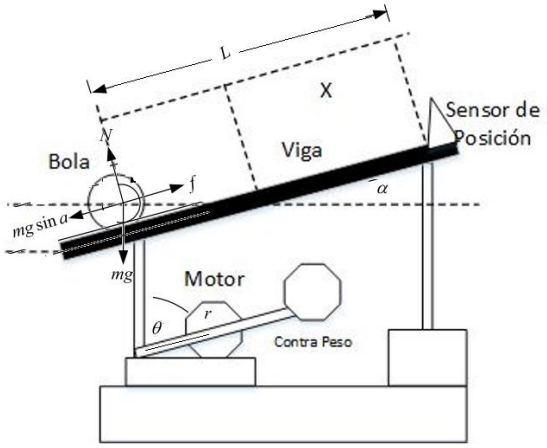


Figura 4.1 Modelo del sistema Beam - Ball

Para resolver esta situación muchas estrategias de control se han utilizado como: controladores PID [74],[75] , control de realimentación de estado, control difuso , control en redes neuronales, pero todas las técnicas anteriores se han considerado en un entorno de trabajo local, en este trabajo se utiliza la técnica de control PID basado en eventos la cual es una técnica que se basa en un entorno de trabajo distribuido en donde la planta, el sensor-actuador están en una posición física y el controlador está en otra distinta, alejada pero las dos están comunicándose vía red inalámbrica. El sistema BB considerado en este trabajo está en la figura 4.1, como puede observarse se compone de una viga que es sostenida por dos brazos el uno fijo y el otro móvil que se puede inclinar accionado por un servomotor DC encima esta una bola que puede rodar hacia adelante o hacia atrás dependiendo del ángulo de inclinación, el objetivo del control del sistema BB es encontrar el ángulo adecuado de manera que la bola pueda permanecer a una posición deseada x, ya que cuando se cambia el ángulo la acción de la gravedad hará que la bola ruede a lo largo de la viga. El sensor de posición, el cual proporciona la distancia de la bola a la referencia del brazo fijo.

El modelo simplificado de espacio de estados de acuerdo con las leyes de la física y aplicando matemáticas se deriva para el sistema BB de la siguiente manera: aplicando equilibrio de fuerzas mediante la ley de Newton

$$m\ddot{x}(t) = mgsin\alpha(t) - \mu mgcos\alpha(t) \qquad (4.1)$$

Donde: m es la masa de la bola, $x(t)$ es la posición de la bola en la viga, g es la aceleración de la gravedad, $\alpha(t)$ es el ángulo de la viga respecto a la horizontal, μ es la constante de fricción cuando la bola rueda por la viga.

Debido a que $\alpha(t)\ y\ \mu$ son relativamente muy pequeños casi siempre se asume que $\sin\alpha(t) \approx \alpha(t), \mu \approx 0$, por lo tanto se tiene de la ecuación anterior que

$$\ddot{x}(t) = g\alpha(t) \qquad (4.2)$$

el ángulo del brazo móvil $\theta(t)$, acorde a la relación geométrica dada por:

$$\alpha(t) = \frac{r}{L}\theta(t) \qquad (4.3)$$

donde r es el radio de palanca al brazo móvil L es la longitud de la viga al sustituir 4.3 en (2) se da:

$$\ddot{x}(t) = g\frac{r}{L}\theta(t) \qquad (4.4)$$

Suponiendo que el vector que describe el estado es: $x(t) = \left[x(t)\ \dot{x}(t)\ \theta(t)\ \dot{\theta(t)}\right]^T\ y\ u(t) = \ddot{\theta}(t)$. El modelo del sistema BB considerando un sistema continuo en espacio de estados se tiene:

$$\begin{bmatrix}\dot{x}(t)\\ \ddot{x}(t)\\ \dot{\theta}(t)\\ \ddot{\theta}(t)\end{bmatrix} = \begin{bmatrix}0 & 1 & 0 & 0\\ 0 & 0 & gr/L & 0\\ 0 & 0 & 0 & 1\\ 0 & 0 & 0 & 0\end{bmatrix}\begin{bmatrix}x(t)\\ \dot{x}(t)\\ \theta(t)\\ \dot{\theta}(t)\end{bmatrix} + \begin{bmatrix}0\\ 0\\ 0\\ 1\end{bmatrix}u(t) \qquad (4.5)$$

Para el sistema BB considerado en este trabajo r= 0.07[m], y L=0.456 [m], por lo tanto, el sistema de espacio de estados sería calculado con las siguientes ecuaciones de estado:

$$\begin{cases}x(t) = Ax(t) + bu(t)\\ y(t) = Cx(t)\end{cases} \qquad (4.6)$$

Donde:

$$A = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \end{bmatrix}, \quad b = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \end{bmatrix}, \quad y = \begin{bmatrix} x(t) \\ \theta(t) \end{bmatrix}, \quad C = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \qquad (4.7)$$

El modelo del sistema continuo puede ser expresado en forma discreta como:

$$\begin{cases} x(k+1) = A_d x(k) + b_d u(k) \\ y(k+1) = C_d x(k+1) \end{cases} \qquad (4.8)$$

Donde $A_d, b_d \; y \; C_d$ puede ser calculado acorde a período específico de muestreo

Modelo de espacio de estado aumentado.- cabe señalar que algunos elementos como la fricción, la no linealidad, la perturbación y características variables en el tiempo del sistema BB son descuidados en el modelo simplificado dado anteriormente. Para compensar estas dinámicas no modeladas la acción de control adicional (k) se introduce por lo tanto el modelo del sistema BB quedaría como:

$$\begin{cases} x(k+1) = A_d x(k) + b_d [u(k) + u_c(k)] \\ y(k+1) = C_d x(k+1) \end{cases} \qquad (4.9)$$

Donde se define un vector de estado aumentado que puede ser $xe(k) = [x^T(k) \;\; u_c(k)]^T$, por lo que el modelo anterior se puede reescribir así:

$$\begin{cases} x_e(k+1) = A_e x_e(k) + b_e u(k) \\ y(k+1) = C_e x_e(k+1) \end{cases} \qquad (4.10)$$

Donde

$$A_e = \begin{bmatrix} A_d & b_d \\ 0 & 1 \end{bmatrix}, b_e = \begin{bmatrix} b_d \\ 0 \end{bmatrix}, C_e = [C_d \quad 0_{2x1}] \qquad (4.11)$$

4.1.1 Metodología para la simulación e implementación

Este apartado de modelamiento e identificación experimental, se centra en el prototipo mostrado en la figura 4.2 Se construye e instrumenta con: sensores estándar; sensores infrarrojos encargados de medir la posición de la bola; el actuador servomotor encargado de posicionar la viga; y la unidad de control.

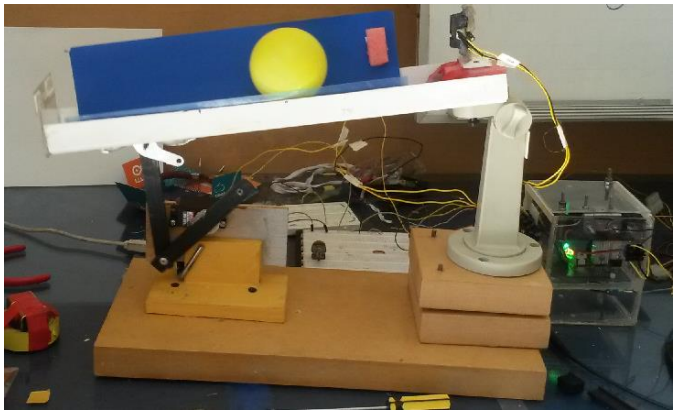


Figura 4.2 Sistema ball & beam implementado

En lo que concierne al diseño mecánico, es importante que el pivote pueda centrar a la viga, procurando que esté en equilibrio, de lo contrario la gravedad genera una no linealidad que hace el control más complicado. Por otro lado, para que las aproximaciones lineales asumidas en este trabajo funcionen correctamente.

Metodológicamente, se propone un modelamiento e identificación del prototipo experimental, asumiendo dos subsistemas: motor y bola; se obtiene la función de transferencia de la planta completa; luego se simula utilizando el software Matlab; posteriormente los controladores se implementan en el prototipo, y luego se mide la respuesta de los controladores al sistema ante una entrada de tipo escalón.

Para modelar e identificar el prototipo experimental, se asumen dos sub-sistemas [76]:

a) El primer subsistema está compuesto por: un servomotor; la transmisión; el sistema de brazos uno fijo y otro móvil; y la viga. Se considera la entrada como el ciclo útil del PWM del motor; y la salida, el ángulo de la viga, en radianes, con respecto a la horizontal. Nos vamos a referir a este sistema como sistema del motor.

b) El segundo subsistema está compuesto por la bola. Se considera como la entrada el ángulo de la viga con respecto a la horizontal; y la salida, la posición de la bola en la viga. Nos vamos a referir a este sistema como sistema de la bola.

Debido a que el peso de la bola no es muy grande, la posición de esta no afecta al sistema del motor, por lo tanto se modelan los dos sistemas independientemente y, simplemente, la salida del sistema del motor es la entrada del sistema de la bola. El modelo matemático del sistema BB está dado por la combinación del modelo del motor con carga y el modelo de la bola como se indica en [4], esto es:

$$\frac{\theta(s)}{l(s)} = \frac{k}{s(s+a)} \quad ; \quad \frac{X(s)}{\theta(s)} = \frac{\rho}{s^2} \tag{4.12}$$

De acuerdo a la ecuación anterior el modelo de la planta tiene tres polos en por lo que el sistema es inestable en lazo abierto. Por ello, el objetivo de diseñar un controlador es precisamente conseguir la estabilidad en lazo cerrado. En la figura 4.3 se muestra el diseño del controlador en diagrama de bloques para el sistema BB.

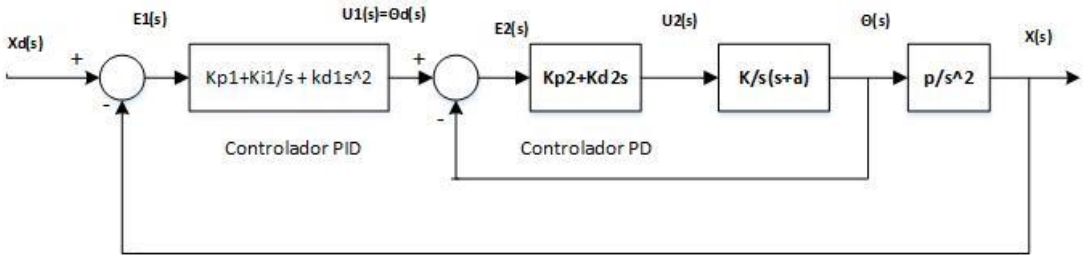


Figura 4.3 Diagrama de bloques para el control de Ball - Beam

Para el sistema BB se utiliza dos controladores, en este trabajo el controlador de tipo PID (controlador lazo externo) es el que permite posicionar la bola en el punto

deseado y un controlador de tipo PD (controlador lazo interno) que permite posicionar horizontalmente el riel. La nomenclatura que utiliza el para el sistema BB es la siguiente:

- $X_d(s)$ posición deseada de la bola.
- $X(s)$ posición de la bola.
- $\theta_d(s)$ posición deseada del riel.
- $\theta(s)$posición del riel.
- $E_1(s)$ error de posición de la bola.
- $E_2(s)$ error de la posición del riel.
- $U_1(s)$ señal de control del controlador PID.
- $U_2(s)$ señal de control del controlador PD.
- $kp1, kp2$ ganancias proporcionales.
- k_{i1} ganancia del integrador.
- $kd1, kd2$ ganancias derivativas.

Para el sistema BB se desea que $\lim_{t\to\infty} x(t) = x_d$ donde x_d es una constante, si la función de transferencia de lazo cerrado $X(s)/X_d(s)$ es estable. Por lo tanto, el problema es encontrar las ganancias del controlador primario así como el secundario que estabilicen el sistema.

Para el modelamiento se identificó las respuestas del sistema en conjunto para lo cual se midió la señal de salida del sistema como es la distancia de la bola al eje respecto a una señal de referencia, estos datos fueron utilizados para la identificación del sistema utilizando el Toolbox de Matlab (IDENT), para mayor detalle ver anexo A. De los modelos identificados, se seleccionó la función de transferencia, que representa mejor el comportamiento del sistema, así como la función del PID tomada de datos experimentales con las contantes $K_p = 8$; $K_i = 0.5$ y $K_d = 2.8$, se sabe que la función del controlador es:

$$PID = \left(K_p + sK_d + \frac{k_i}{s}\right) \tag{4.13}$$

por lo tanto:

$$Gt = PID * FTp \tag{4.14}$$

$$Gt = \frac{6.049}{s^2 + 1.091s + 6.552} \quad ; \quad PID = \frac{2.8s^2 + 8s + 0.5}{s} \tag{4.15}$$

$$FTp = \frac{6.049s}{((s^2 + 1.091s + 0.503) * (2.8s^2 + 8s + 0.5))} \tag{4.16}$$

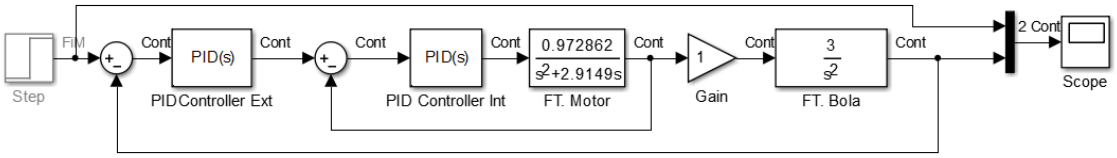


Figura 4.4 Diagrama de control en cascada simulado

De forma experimental se obtuvo las constantes de la función del motor, así como de la bola.

La tendencia actual de controlar los sistemas dinámicos en forma digital en lugar de analógica, se debe principalmente a la disponibilidad de dispositivos electrónicos digitales de bajo costo. En la figura 4.4. Se muestra el diagrama de bloques del sistema BB en tiempo continuo, en la transformada de Laplace.

En la figura 4.5 Se presenta el sistema en su representación (en la parte de anexo de este documento se muestra como se hace dicha transformación) tomando en cuenta que el tiempo de muestreo es de 30 ms y con los controladores antes presentados, tenemos:

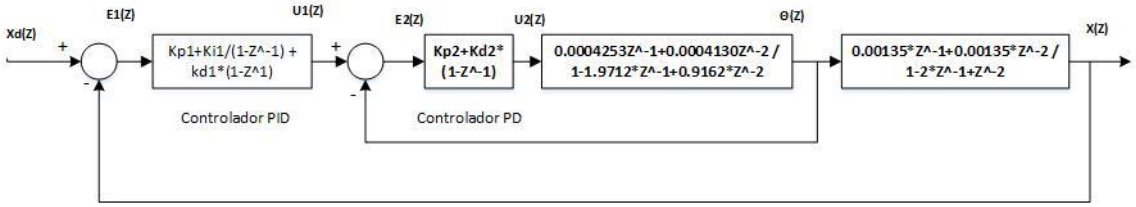


Figura 4.5 Sistema Ball - Beam en su representación Z

4.1.2 Diseño del controlador PID Predictivo para la planta BB

Para establecer criterios de predicción en cualquier sistema se debe de tener muy claro cómo interactúan las diferentes variables en la respuesta final. Cada variable tiene diferente grado de influencia en el resultado final, dependerá del diseñador considerar las de mayor o menor interacción en el fenómeno de estudio. En la realidad no existen fenómenos lineales, simplemente son aproximaciones. Las funciones de transferencia que expresan el comportamiento de un sistema presentan una gran complejidad por ende muchos diseñadores descartan este método para ahorrar varios recursos.

La expresión matemática de un fenómeno es de vital importancia para elaborar estrategias, soluciones y algoritmos de predicción para sistemas de control. En su esencia la estrategia de control predictivo hace uso de un modelo matemático interno y de una estrategia de optimización para predecir las salidas del sistema dentro de un intervalo de tiempo al que se le denomina horizonte de predicción.

El movimiento de la pelota sobre el riel es influenciado por varios factores, el más importante es el ángulo del servomotor al que esta sujetado la riel y el peso de la pelota. Este es el criterio central para poder elaborar un algoritmo que nos ayude a calcular un resultado futuro, el controlador tiene acción directa sobre el ángulo del servo es decir sobre la causa permitiendo intuir el valor del efecto (la posición de la pelota sobre el riel).

4.1.3 Algoritmo predictivo para el control PID basado en eventos para el sistema BB

Se desea implementar un algoritmo predictivo para un sistema BB el cual tiene los sensores y actuadores físicamente alejado del controlador PID basado en eventos, lo que implica que la acción de control se ejecutará cuando ocurra un evento en este caso el disparo del evento sucederá en el sensor y se dará en base al error entre las lecturas de la señal de salida del sensor (distancia) anterior respecto al actual, en base a este dato y si cumple una función lógica el sensor seguirá emitiendo datos que ingresaran al algoritmo localizado en el controlador, permitiendo de esta manera calcular la función PID, la que emitirá un dato que permite controlar la posición de la bola a una distancia predefina en la viga. El controlador calcula el valor del ángulo del servo resultado del algoritmo PID, durante la ausencia de información entre planta y controlador (por ser un sistema de control asíncrono) se pierde información de vital importancia para que algoritmo de control se ejecute correctamente (Figura 4.6). Obteniendo un sistema oscilante. En este trabajo se generó varias alternativas para el controlador PID basado en eventos descritos en el literal 3.4

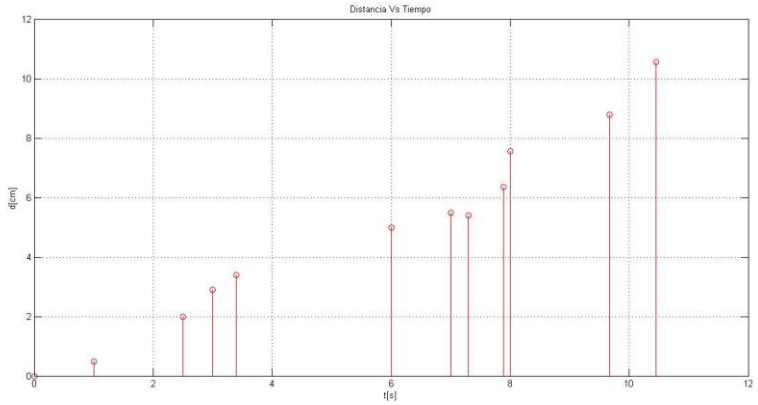


Figura 4.6 Muestreo asíncrono

4.1.4 Definición de los requerimientos

Para realizar el algoritmo PID requiere un dato emitido por el sensor de distancia inalámbrico con este dato al ingresar al algoritmo, se realizará el cálculo y minimización del error para efectos de estabilizar el sistema y emitirá un dato que es, el que se enviará al actuador vía comunicación wifi.

Durante la ausencia de datos se ingresan valores al PID que no vienen directamente de la planta, sino que son calculados por el controlador basándose en los últimos datos que llegaron (Figura 4.7).

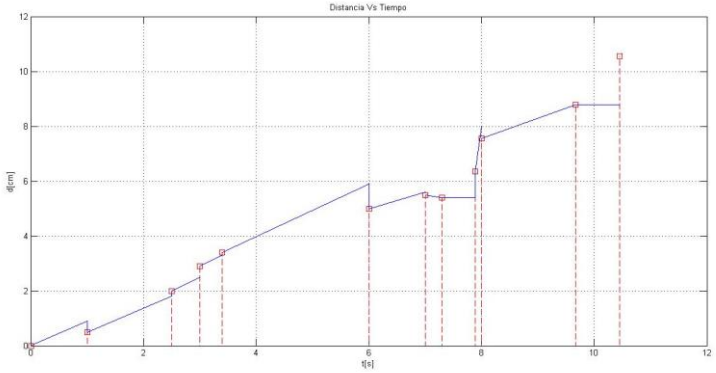


Figura 4.7 Muestreo asíncrono con información adicional durante la ausencia de comunicación

Los últimos valores de distancia que la planta envió y ángulo que calcula el PID en el controlador, son usados como parámetros para determinar una función lineal que se dispara cuando no existe comunicación entre cliente-servidor.

Para predecir la distancia se usa un contador que inicia y vuelve a cero entre actualizaciones de datos, el último valor de distancia (disf) es usado como un offset para que la función lineal inicie desde el último valor real. Para el definir la pendiente de la función lineal se utiliza el ángulo de inclinación de la riel definiendo la dirección y la velocidad con la que la pelota se moverá.

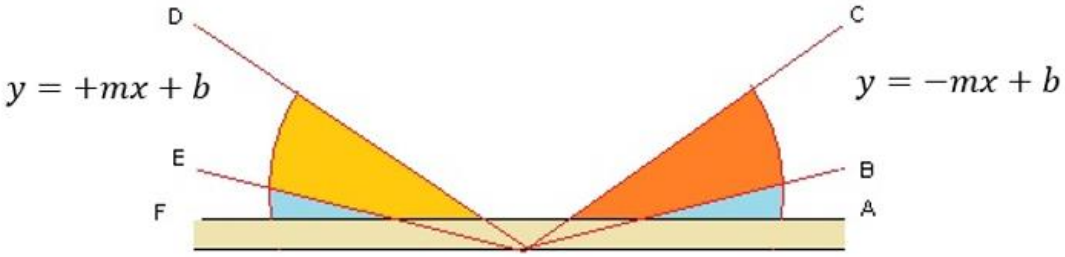


Figura 4.8 Dirección de la pelota por inclinación de la riel

En la figura 4.8 se presenta los posibles ángulos giro del servo motor que están comprendidos en los siguientes valores: [F, D] U [C, A]

Cuando el valor del ángulo este comprendido entre el intervalo [C, B] la dirección de la pelota será para la izquierda, cuando el intervalo sea de [E, D] la dirección de la pelota será para la derecha. Para los valores de los ángulo que están en el intervalo [F, E] U [B, A] la pelota no tendrá un movimiento significativo.

Con estos valores podemos determinar el signo de la pendiente: [C, B] pendiente negativa, [E, D] pendiente positiva, [F, E] U [B, A] la pendiente tendrá valor 0.

4.1.5 Identificación de los Módulos del algoritmo implementado

Los módulos implementados son:

- Configuración de la comunicación
- Definición de variables
- Inicio de algoritmo

- Configuración de envio-recepcion de datos
- Lectura y delimitación del dato requerido
- Conversión de variable
- Calculo del PID
- Registro e impresión de datos para análisis
- Fin

El algoritmo deberá cumplir con las siguientes características:

Finitud.- es decir el algoritmo termina tras un número finito de pasos

Definibilidad.- Se define paso a paso de forma clara.

Entrada.- Se define las entradas necesarias y las variables.

Salida.- Se define e identifica claramente la salida.

Efectividad.- Se define su efectividad por los registros de salida los cuales se los puede graficar.

Algoritmo Nodo Servidor Controlador

1. Configuración de comunicación Servidor en Arduino Ethernet y Cliente Wifi en Dragnio Yun.
2. Definición de protocolo de comunicación html
3. Entrada : $X_t^1 = \sum x_i^1$; Lee y almacena variable de distancia de sensor
4. Calcula el PID en base a los datos de lectura del sensor llegados luego de cumplir una función lógica $Z_{t'}^1 = Z_\tau^1(\hat{x})$; Procesa la información en el algoritmo de control PID con los datos estimados luego de realizar una periodicidad de entre eventos, datos reales llegados
5. Salida : $Z_{t+\tau}^1$; Envio del valor PID, ángulo al actuador remoto, dato enviado via wifi.

4.2 Algoritmo heurístico para la sintonización del PID basado en eventos en un WNCS

Los algoritmos heurísticos se caracterizan por ser métodos que buscan soluciones a funciones o a problemas de optimización. En la actualidad, los algoritmos heurísticos han incrementado considerablemente su aplicación, gracias a que encuentran soluciones satisfactorias a problemas complejos, a pesar de no ser capaces de probar que la solución encontrada es la óptima. En este proyecto se presenta dos métodos heurísticos, la optimización por cúmulo de partículas (PSO) y los algoritmos genéticos (AG).

4.2.1 Optimización mediante el cúmulo de partículas (PSO)

La metaheurística desarrollada por Eberhart y Kennedy [77], como un algoritmo estocástico el PSO fue originalmente desarrollado para la optimización en espacios continuos, pero también ha sido desarrollado para espacios discretos presentando un buen rendimiento cuando se aplica a funciones objetivos discontinuas, por lo que es utilizado en la optimización en muchos campos de ingeniería. El PSO simula el vuelo inteligente de un cumulo de aves, se ha caracterizado por una optimización global y una rápida convergencia. En este proyecto se utiliza el algoritmo PSO para sintonizar los parámetros del controlador PID.

Mediante el algoritmo PSO se realiza la sintonización y encuentra los parámetros óptimos del controlador PID como lo son las constantes Kp, Ki, Kd. Se implementa un controlador PID digital que viene de la desratización de un PID análogo y obedece a la relación: $u(k) = K_p e(k) + K_I \sum_{j=0}^{k} e(j) + K_D[e(k) - e(k-1)]$; donde Kp es el coeficiente proporcional, Ki es el coeficiente integral, Kd es el coeficiente diferencial, u(k) es la salida de control en un tiempo de muestreo k el cual es variable, pero para efectos de la simulación se considera un sistema continuo periódico.

Parámetros del algoritmo PSO

El PSO se caracteriza por ser un algoritmo iterativo y estocástico sobre un enjambre de individuos, denominados partículas inteligentes. La posición de cada partícula representa una solución potencial al problema de optimización. Una partícula está compuesta de tres vectores:

- El vector de posición $x_i = [x_{i1}, x_{i2}, \ldots\ldots, x_{in}]$ el cual guarda la posición actual de la partícula que se toma como referencia para calcular la nueva dirección de búsqueda.
- El vector $pBest_i$ ó $pBest_i = [p_{i1}, p_{i2}, \ldots\ldots, p_{in}]$ el cual guarda la posición de la mejor solución encontrada hasta esa generación, y representa la memoria de la partícula
- El vector de velocidad $v_i = [v_{i1}, v_{i2}, \ldots\ldots, v_{in}]$ el cual guarda el gradiente (dirección del vuelo de la partícula).

El flujo del algoritmo PSO se determina en el siguiente procedimiento: 1) Se inicializa la población de las partículas incluyendo los vectores de posición y velocidad aleatoriamente ; 2) evalúa la función objetivo para cada partícula; 3) Compara la función objetivo de cada partícula con la posición optima local si es la mejor guarda la posición de la partícula ; 4) Compara la función objetivo de cada partícula con la posición global si es la mejor guarda ; 5) Actualiza la velocidad y posición de la partícula dados en los pasos (3) y (4), y; 6) Si la condición de finalización termina, entonces evalúa otra partícula va al paso (2). El modelo matemático que representa el algoritmo PSO se presenta en la ecuación (3) que modifica la velocidad de la partícula o la dirección de búsqueda y en la ecuación (4) que modifica la posición de la partícula.

$$v_i^{k+1} = wv_i^k + c1\ rand_1\left(pBest_i - x_i^k\right) + c2 rand_2\left(gBest_i - x_i^k\right) \qquad (4.17)$$

$$x_i^{k+1} = x_i^k + v_i^{k+1} \qquad (4.18)$$

En donde: i=1,2,…m, siendo m el número total de partículas de la población, Vi es la velocidad de la partícula, pBest es la posición optima de la partícula, rand () es un número randómico entre 0 y 1; Xi es la posición de la partícula; c1,c2 y w son factores de aprendizaje siendo c1 una componente cognitivo, c2 el componente social y w el factor de inercia.

Para comenzar con el algoritmo PSO antes se define el número de partículas que harán la búsqueda, para esta variable no existe en la literatura un número recomendable pero como es de pensar si se ocupa una cantidad enorme de partículas entonces eso implica que el espacio de búsqueda será de igual forma, de tal manera, que se tendrá una mayor probabilidad de encontrar la solución en pocas iteraciones. Tener un número cuantioso de partículas puede resultar contraproducente debido a que al cuantificar la aptitud de cada partícula hará que el tiempo de búsqueda incremente considerablemente. Por otro parte, el número de partícula también pude estar limitado por la memoria de los dispositivos electrónicos. En este trabajo se eligió aves que buscaran la solución. Con un número máximo de iteraciones. Estos números se establecieron de acuerdo a pruebas hechas en simulaciones. Cabe mencionar que la dimensión del problema es de orden tres esto es debido a que se cuenta con tres variables [kp,ki,kd].

En la etapa de inicializar las variables también se definen los parámetros del PSO, en este caso c1=0.2 y c2=0.2 que son la razón o rapidez de aprendizaje del componente cognitivo y social, respectivamente. También se inicializó los parámetros R1 y R2 de manera aleatoria en un intervalo de [0 1]. Posteriormente se inicializó la posición y velocidad de cada partícula de manera aleatoria.

Una vez definido los parámetros del PSO y generados las partículas se cuantifica la aptitud de cada partícula. La cuantificación de la aptitud de cada partícula se hace a través de la integral del error al cuadrado o como se conoce comúnmente ISE (por sus siglas en inglés, Integral Squared Error) la intensión de este método es minimizar el error de salida, en la función del tiempo transcurrido.

$$ISE = \int_0^{\tau} e^2(t)dt \qquad (4.19)$$

El ISE, es relativamente insensible a pequeños errores, pero los grandes errores contribuyen fuertemente al valor de la integral. Consecuentemente, utilizar el ISE como criterio de optimización dará como resultado una respuesta con pequeños sobrepasos en el sistema pero con un largo tiempo de estabilización puesto que los pequeños errores a lo largo del tiempo contribuyen muy poco a la integral.

Existen otros criterios de optimización tales como utilizar la integral del valor absoluto del error, IAE (por sus siglas en inglés, *Integral Absolute Error,* ecuación 4.20).

$$IAE = \int_0^{\tau} |e(t)|dt \qquad (4.20)$$

Este criterio es más sensible a pequeños errores pero es menos sensible que el ISE a grandes errores. También se encuentra la integral del tiempo multiplicado por el valor absoluto del error, ITAE (por sus siglas en inglés, Integral Time Absolute Error ecuación 4.21)

$$ITAE = \int_0^{\tau} t|e(t)|dt \qquad (4.21)$$

El ITAE, es insensible a los errores iniciales, y a veces inevitables, pero penalizan fuertemente los errores que permanecen a lo largo del tiempo. La respuesta óptima definida por ITAE, consecuentemente, mostrará tiempos cortos en la estabilidad y un mayor sobrepaso en la respuesta del sistema que los otros criterios.

Una vez finalizado el proceso de cuantificar la aptitud de cada individuo o partícula se procede a buscar el individuo con la mejor aptitud, en otras palabras, como se desea encontrar el mínimo del sistema entonces se busca el individuo que presenta un número menor de ISE. Este individuo será el líder que guiará al resto de la población. En la primera generación cada individuo no cuenta con un historial de tal manera que su experiencia es su posición actual. De esta forma se determina la siguiente posición y velocidad de acuerdo a la ecuación 4.14 y 4.15, las que generan la nueva generación. Por ende se repite el procedimiento hasta cumplir con el número máximo de iteraciones, Cabe mencionar, si en una de las generaciones existe un individuo que exhibe una mejor aptitud que el líder actual, el individuo con la mejor aptitud será el nuevo líder que guiará al resto de la población. De igual manera, la experiencia de cada individuo va evolucionando. Por último, al terminar el proceso el líder será la solución del problema.

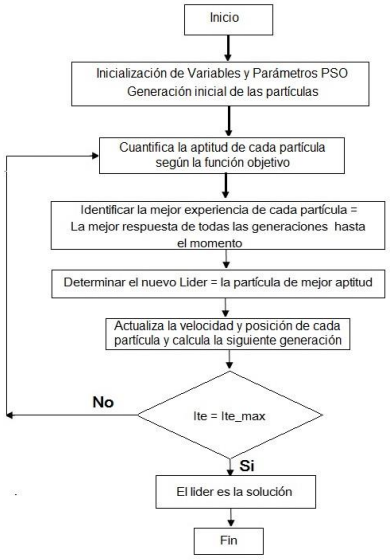


Figura 4.9 Diagrama de flujo para el algoritmo PSO.

4.2.2 Algoritmo PSO para sintonizar un controlador PID

```
1. Inicializa variables:
n = tamaño del cúmulo; n_bird= número de iteraciones; dim= dimensión del problema;
c1=c2= razón de aprendizaje; w = factor de inercia
Inicializa parámetros y rangos de búsqueda
2. Generación de una población inicial
3. Evaluación inicial de la población
4. Selecciona la mejor aptitud en base a una función objetivo
5. Calculo de la velocidad, posición
6. Evalúa una nueva partícula
for ite = 1 : n_bird % condición para finalizar la búsqueda.
7. Evaluar la aptitud de cada partícula
   for i = 1 : n
   aptitud_actual(i) = fModeloBB(pos_actual(:,i)) ;
   end
8. Encontrar la mejor aptitud local
   for i = 1 : n
      if aptitud_actual(i) < aptitud_local_best(i)
      aptitud_local_best(i) = aptitud_actual(i);
      pos_local_best(:,i) = pos_actual(:,i);
      end
   end
9. Encontrar la partícula con la mejor aptitud actual
[aptitud_actual_global_best, g] = min(aptitud_local_best);
Ap_act_glob_beite(ite)=aptitud_actual_global_best ;
10. Cambiar el líder o seguir con el mismo según su aptitud.
if aptitud_actual_global_best < aptitud_global_best
aptitud_global_best = aptitud_actual_global_best;
   for i = 1 : n
   pos_global_best(:,i) = pos_local_best(:,g);
   end
end
11. Calcular la velocidad
velocidad = w*velocidad + c1*(R1.*(pos_local_best - pos_actual)) +
c2*(R2.*(pos_global_best - pos_actual));
12. Calcular la posición
pos_actual = pos_actual + velocidad;
13. Evolución de las partículas
kpv1(ite) = pos_global_best(1,1);
kiv1(ite) = pos_global_best(2,1);
kdv1(ite) = pos_global_best(3,1);
end
[Jbest_min,I] = min(aptitud_actual) ;% Aptitudes mínimas
pos_actual(:,I); % La mejor solución
14. Solución encontrada por el PSO Controlador PID
```

```
kp1 = pos_actual(1,I); % Ganancia Proporcional
ki1 = pos_actual(2,I); % Ganancia integral
kd1 = pos_actual(3,I); % Ganancia Derivativo

k=[kp1 ki1 kd1];
```

4.2.3 Optimización mediante algoritmo genético

El algoritmo genético (AG) es una técnica de búsqueda que se utiliza para optimizar sistemas o encontrar soluciones a funciones. El AG está inspirado en la evolución de Charles Darwin. Es decir, se basa en los mecanismos de selección que utiliza la naturaleza, donde los individuos más aptos de una población son los que sobreviven al adaptarse más fácilmente a los cambios que se producen en su entorno. El principio básico del algoritmo genético fue establecido por John Holland en 1975.

De acuerdo a la literatura, existen muchas variantes de los algoritmos genéticos pero la esencia de esta técnica es la misma: dada una población de individuos (posibles soluciones del problema), la influencia del ambiente o el entorno ocasiona selección natural (la supervivencia del más apto), misma que da lugar a un incremento en la aptitud de la población. En una forma explícita, se diseña una función de calidad a ser maximizada o minimizada, posteriormente se crea de forma aleatoria un conjunto de soluciones candidatas, es decir, elementos del dominio de la función, y aplicar la función de calidad como una medida de aptitud abstracta. Basado sobre la aptitud de cada individuo, algunos de los mejores candidatos son seleccionados para pasar a la siguiente generación y aplicarles recombinación y/o mutación. La recombinación es un operador aplicado a dos o más candidatos seleccionados (llamados padres). Realizando la recombinación y la mutación se conduce a un conjunto de nuevos candidatos (hijos) que compiten junto con los padres por un lugar en la siguiente generación. Este proceso puede ser iterativo hasta que un candidato con suficiente calidad (una solución) sea encontrado o que algún criterio de finalización establecido sea previamente alcanzado (figura 4.10).

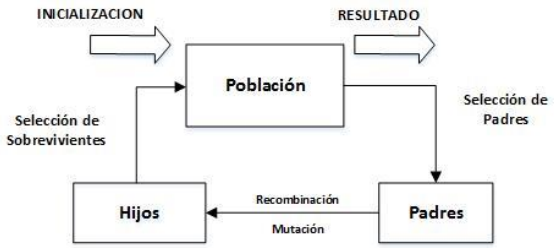


Figura 4.10 Algoritmo Genético

El algoritmo genético es una herramienta que pertenece a la rama de la inteligencia artificial, la cual ha probado ser útil para encontrar soluciones a problemas reales. Como se ha mencionado el AG es una herramienta de optimización. Un enfoque a problemas de control, la optimización de una función por lo general resulta ser el error

entre la referencia y la salida del sistema. En muchos sistemas se espera que la respuesta del sistema tenga tiempos cortos, sobrepasos pequeños y sin errores a lo largo del tiempo.

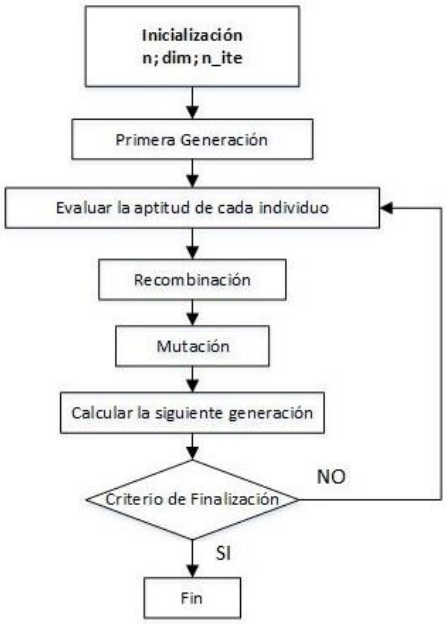


Figura 4.11 Diagrama de Flujo del Algoritmo Genético

Algoritmo Genético implementado

Primera generación.- Como en la mayoría de los programas el primer paso es inicializar las variables a utilizar. Sin embargo, en el algoritmo genético se genera la primera generación tomando en cuenta el espacio de búsqueda de la solución, el espacio de búsqueda se genera de forma aleatoria y cada elemento de esta generación se le conoce como individuo. Donde cada individuo es una posible solución al problema. Otro factor que se considera al generar la primera generación es el tamaño de la población es decir el número de individuos que competirán para sobrevivir y por último que tan aleatorio se pretende generar el espacio de búsqueda.

Evaluar aptitud de cada individuo.- Para encontrar la solución al problema de optimización, se debe evaluar cada individuo de la generación en una función de calidad o aptitud. Al someter cada individuo en la función de aptitud lo que se hace es cuantificar sus cualidades, de tal manera y bajo algún criterio se seleccionan a los individuos que participarán en la recombinación y generar nuevos candidatos que competirán por sobrevivir y ocupar un espacio en la siguiente generación o incluso puede llegar el caso de que uno de estos elementos cumpla con todas las cualidades buscadas y sea la solución.

La función de aptitud es un punto crítico del algoritmo genético. Esta función debe ser capaz de poner en manifiesto la aptitud de cada individuo para el problema que se desea optimizar.

Selección.- Después de conocer la aptitud de cada uno de los individuos se procede a elegir los individuos que serán padres. Para este proceso existe diferentes técnicas de

selección, una de ellas es seleccionar la mitad de la generación que haya mostrado ser la más apta para sobrevivir, a este proceso se le conoce como elitismo. Sin embargo, esta técnica en algunos problemas presenta convergencia prematura, en tal caso existen otras alternativas de hacer la selección de padres como la selección aleatoria o la conocida como ruleta.

Operador de recombinación.- Una vez hecha la selección de los padres se prosigue hacer el cruce entre padres, el operador que más se utiliza es seleccionar dos padres y a partir de ellos crear dos hijos, la generación de hijos se conoce como el proceso de recombinación. Existen diversas maneras de generar los hijos dependiendo de la codificación que se usa, en este trabajo se usan números reales y el operador de recombinación se hace de manera aritmética.

Mutación.- La mutación es opcional, en muchos de los casos es preferible contar con este operador, ya que puede llegar el caso en que el algoritmo genético llegue a una solución local, donde este operador ayuda a encontrar una solución global. Cabe mencionar que la mutación surge de manera aleatoria y uno de los individuos de la población mutará, es decir, cambiará completamente por otro agente del espacio de búsqueda que es generado de forma aleatoria.

Generación siguiente.- Para generar la siguiente generación se toma los padres que fueron seleccionados con respecto a su aptitud y los hijos que previamente fueron creados a partir de la recombinación de los padres y además de la mutación si este fue generado. La nueva generación cuenta con el mismo número de individuos como la generación pasada y están listos para competir por un puesto en la siguiente generación.

Criterio de finalización.- Este paso es esencial para que el algoritmo genético detenga la búsqueda, la búsqueda puede ser detenida si uno de los individuos presenta las aptitudes deseadas, es decir, el algoritmo genético llegó a la solución. O puede detenerse si ya cumplió con un número de generaciones, es decir, un número de iteraciones que previamente se ha elegido.

4.2.4 Algoritmo Genético para sintonizar un controlador PID

```
1. Inicialización: n= tamaño de la población; dim = dimensión del problema; n_ite=número de iteraciones (generaciones)
2. Rangos iniciales para constantes PID
3. Generación de un la población con distribución normal dentro del rango
for ite = 1 : n_ite % Condición para finalizar las iteraciones
4. Analizar la aptitud de cada uno de los individuos
for i = 1 : n
aptitud_actual(i) = fModeloBB(P(:,i)); % Función para la aptitud.
end
4. Ordenar de menor a mayor.
[FA ind] = sort(aptitud_actual);
 FAite(ite)=FA(1,1);
5. Selección de los padres que se reproducirán.
for i = 1 : n
   Px(i) = P(1,ind(i));
```

```
   Py(i) = P(2,ind(i));
   Pz(i) = P(3,ind(i));
end
6.Combinación y generación de los hijos.
   for i=1:n/4
      Hx1(i)=Px(i);
      Hy1(i)=Py((n/2+1)-i);
      Hz1(i)=Pz((n/2+1)-i);
      Hx2(i)=Px((n/2+1)-i);
      Hy2(i)=Py(i);
      Hz2(i)=Pz(i);
   end
   P1 = Px;
   P2 = Py;
   P3 = Pz;

   P1((n/2+1):(3/4*n))=Hx1;
   P2((n/2+1):(3/4*n))=Hy1;
   P3((n/2+1):(3/4*n))=Hz1;
   P1((3/4*n+1):n)=Hx2;
   P2((3/4*n+1):n)=Hy2;
   P3((3/4*n+1):n)=Hz2;
7. Nueva población.
P(1,1:n) = P1;
P(2,1:n) = P2;
P(3,1:n) = P3;
8. Individuo con mejor aptitud.
kpv1(ite) = P1(1);
kiv1(ite) = P2(1);
kdv1(ite) = P3(1);
end
9. Controlador PID
kp1 = P1(1) % Ganancia Proporcional
ki1 = P2(1) % Ganancia Integral
kd1 = P3(1) % Ganancia Derivativo
% Controlador PD
k=[kp1 ki1 kd1];
10. Gráficos
```

4.2.5 Resultados

A continuación se presenta los resultados que se obtuvo en las simulaciones.

Las simulaciones se realizaron en Matlab, para evaluar los algoritmos heurísticos se realizó sobre una función objetivo, que se obtuvo experimentalmente mediante la toolbox ident de matlab, en la cual se ingresaba unos valores de referencia y los valores de salida dada por la posición de la bola en la viga mediante el sensor de

distancia. La función objetivo evalúa las constantes del PID, bajo el criterio de la integral del error absoluto IAE

```
function[IAE]=fCtrlPlantaBB(k)
%********** FUNCION OBJETIVO DE UN PROCESO BEAN & BALL *********
clc;
num=[6.049 0];
den=[2.8 11.05 10.64 4.569 0.2515];
error=0;
dt=0.03; Tmax=5;
t=0:dt:Tmax;
s=tf('s');
Planta=tf(num,den);
sys_c0=feedback(Planta,1);
% Controlador PID con valores iniciales experimentales de :
%kp=8; ki=0.5;kd=2.8;
kp=k(1);ki=k(2); kd=k(3);
C= pid(kp,ki,kd);
%Sistema lazo cerrado realimentación Unitaria
sys_c1=feedback(C*Planta,1);
m= step(sys_c1,t);
%Calculo del error
error=1-m;
 IAE=trapz(t,abs(error));
```

La función de transferencia del sistema está dado por :

$$FT_{sistema} = \frac{6.049s}{2.8s^4 + 11.05s^3 + 10.64s^2 + 4.569s + 0.2515} \qquad (4.22)$$

Generando las siguientes respuestas:

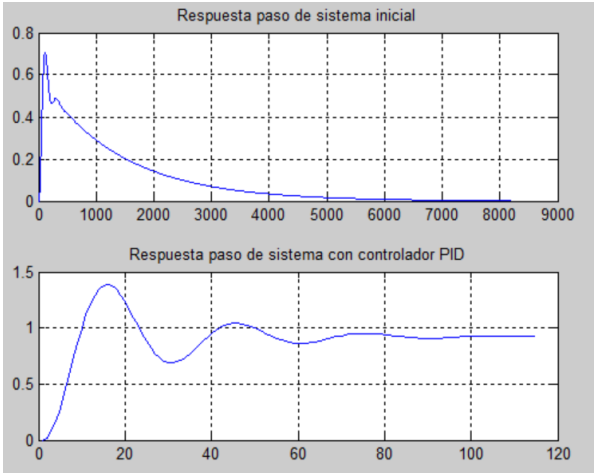


Figura 4.12 Respuestas del sistema a una señal paso

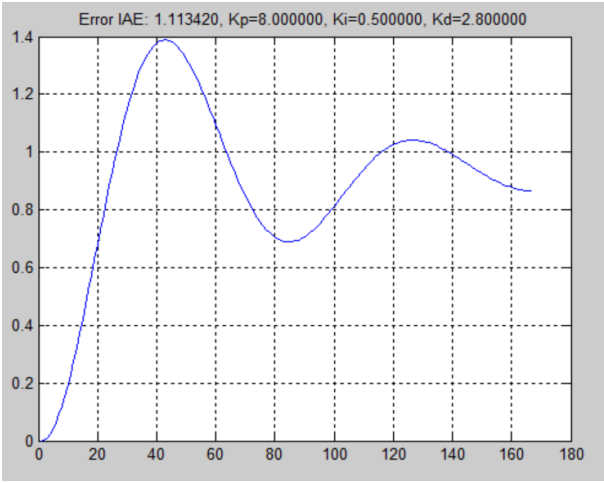


Figura 4.13 Respuesta del sistema con controlador PID

Los resultados generados por el algoritmo genético generaron los siguientes valores de las contastantes PID bajo el criterio del error IAE como se muestran en la siguientes figuras:

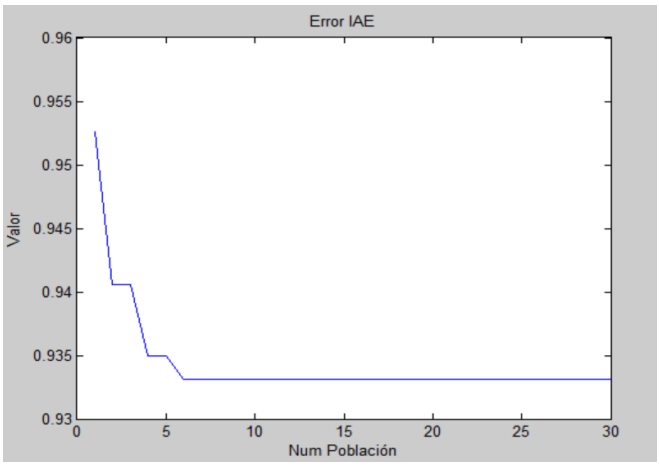


Figura 4.14 Error generado por el algoritmo genético bajo el criterio IAE

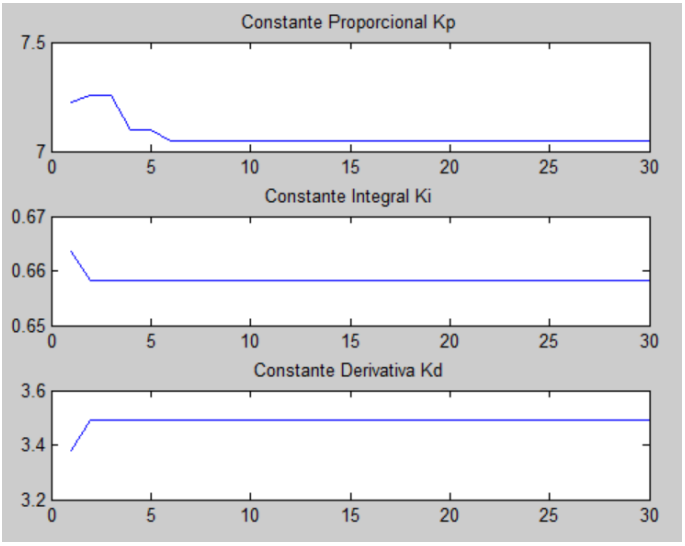


Figura 4.15 Constantes Kp,Ki,Kd con algoritmo genético.

Los resultados generados por el algoritmo PSO generaron los siguientes valores de las contastantes PID bajo el criterio del error IAE como se muestran en la siguientes figuras:

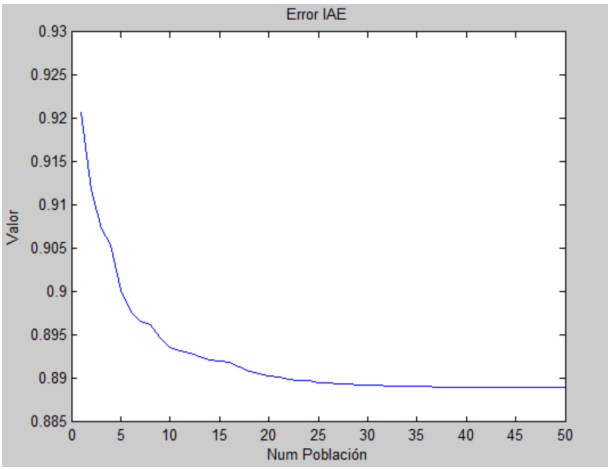


Figura 4.16 Error generado por el algoritmo PSO bajo el criterio IAE

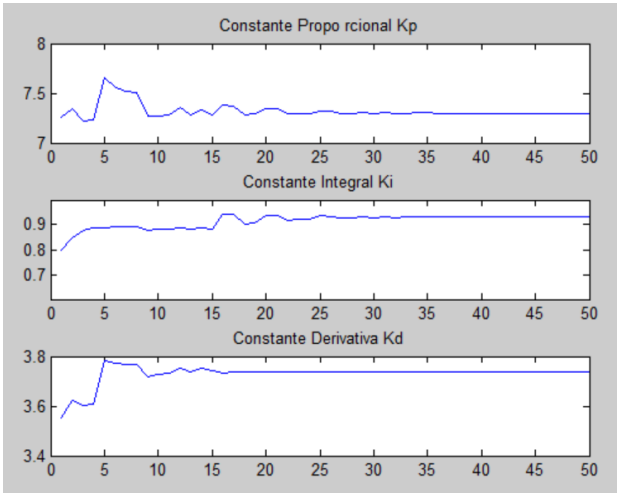


Figura 4.17 Constantes Kp,Ki,Kd con algoritmo PSO.

En la siguiente tabla 4.1 se muestran los valores obtenidos al utilizar los algoritmos heurísticos y evaluar en la funcion objetivo. En las Figura 4.12, se observa la respuesta de la funcion de transferencia del sistema obtenida de valores experimentales ante una entrada de tipo escalón, que se refiere a una posición deseada de la bola en la viga. En la figura 4.12(a) se observa que la respuesta de la simulación sin controlador como se puede apresiar es inestable, en la figura 4.12(b) se observa el comportamiento de la planta con controlador PID, pero se observa que tiene un sobre impulso SO del 38%, pero la integral del error absoluto (IAE) es demasiado alto de 11.134. (como se obserba en la primera columna de la tabla 4.1).

Valores obtenidos al analizar la Función de transferencia de la planta obtenida	Valores obtenidos al sintonizar con algoritmo Genetico	Valores obtenidos al sintonizar con algoritmo PSO
Kp = 8	Kp = 7.04644	Kp = 7.2954
Ki = 0.5	Ki = 0.6582	Ki = 0.9325
Kd = 2.8	Kd = 3.4903	Kd = 3.7377
IAE = 11.134	IAE = 0.9331	IAE = 0.8890
Maximo Sobreimpulso = 0.38	SO = 0.28	SO = 0.29
Tiempo pico = 12.6	Tp = 12	Tp = 11.7
Tiempo de subida = 0.75	Ts = 0.72	Ts = 0.72
	Tiempo de corrida = 49.79seg	Tiempo de corrida = 52.53seg

Tabla 4.1 Datos de los tiempos de retardo en PID asíncrono con error IAE

Al simular la funcion de la planta con los parametros obtenidos luego de sintonizar con las constantes Kp,Ki,Kd dadas por el algoritmo genético observamos que disminuye el sobre impulso al 28% pero baja el IAE radicalmente. De igual manera se tiene cuando se simula el comportamiento de la planta con las constantes obtenidas por el algoritmo PSO.
Los valores Kp,Ki,Kd obtenidos por los algoritmos heurísticos de sintonización estan dentro de un rango de busqueda que fueron configurados en base a los valores obtenidos experimentalmente como son : kp_ini=0;kp_fin=10; ki_ini=0;ki_fin=2; kd_ini=0;kd_fin=6. Al simular y correr los algoritmos dieron valores dentro del rango especificado es decir se limito el campo de busqueda de la mejor solución, de lo contrario el espacio de busqueda puede ser ilimitado, es decir nos puede dar cualquier valor que cumpla con la funcion objetivo, pero esos valores no siempre pueden implementarse en la realidad y por ende no se puede obtener un control óptimo.

CAPITULO 5

5 Controlador PID predictivo para WNCS

5.1 Descripción de la arquitectura de los dispositivos que conforman el WNCS

El cliente (sensor-actuador) se halla localizado directamente en la planta, en este nodo se usa Arduino Yun que permite comunicación por Wi-Fi. En un nodo remoto se halla Arduino Ethernet (controlador) que está conectado a un router y trabaja como servidor en la red. Cada dispositivo dentro de la red tiene una dirección IP única que sirve como parámetro de identificación. El servidor tiene una dirección fija a diferencia del cliente que tiene una dirección dinámica que es otorgada por el router por el protocolo DHCP.

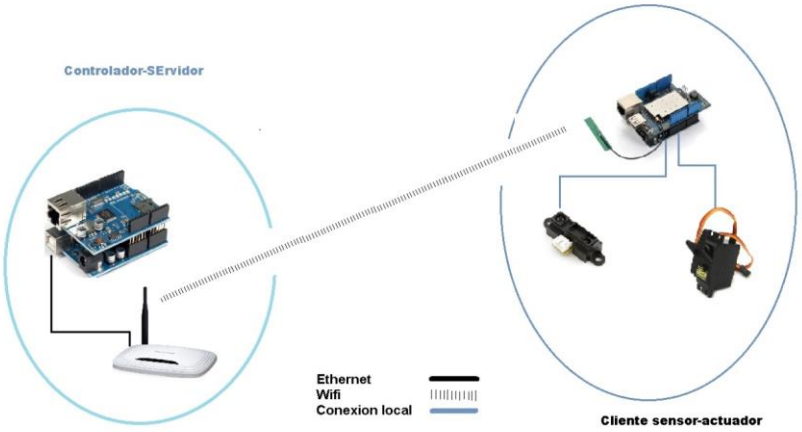


Figura 5.1 Elementos de Red

5.1.1 Elementos de red

Router.

Para el presente proyecto se usó un router marca tp-link (TL-WR841N). Se configura una red LAN con direcciones 192.168.0.X y submáscara 255.255.255.0. (Figura 5.2)

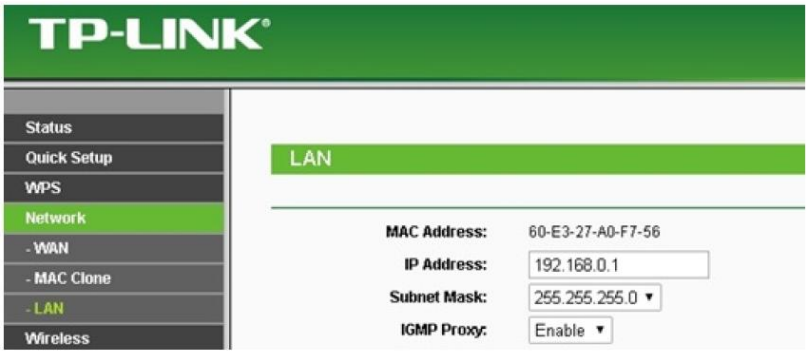


Figura 5.2 Configuración del Router tp-link

Se configura la red externa (WAN), dependiendo del proveedor de internet tendremos una dirección pública fija o dinámica. En este caso se selecciona una ip fija. (Figura 5.3)

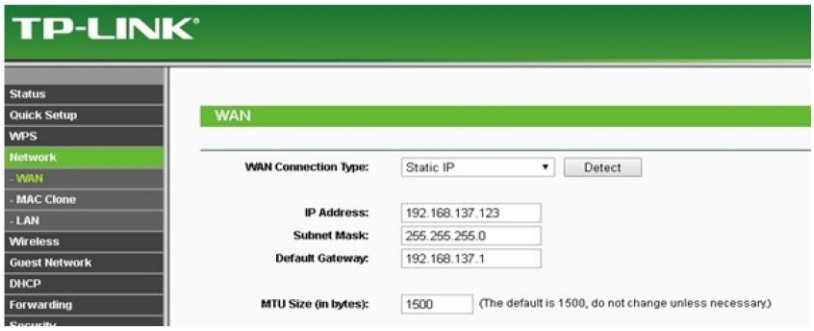


Figura 5.3 Configuración de red WAN

Se debe de configurar el protocolo DHCP que otorga direcciones IP a los diferentes elementos del sistema. Se define un pool de direcciones (192.168.0.100…… 192.168.0.199).(Figura 5.4)

Figura 5.4 Configuración del protocolo DHCP

5.1.2 Servidor-controlador

Arduino Ethernet Shield permite a una placa Arduino conectarse a internet. Se basa en el chip Wiznet W5100 ethernet. El W5100 Wiznet proporciona una red IP capaz de usar los protocolos TCP y UDP, tiene un estándar de conexión RJ-45. Características: Tensión de alimentación de 5V, Controlador Ethernet: W5100 con una memoria interna de 16Kb, Velocidad de trasmisión de 10/100Mb, Conexión con Arduino a través del Puerto SPI, presentado con más detalle en [78].

En este nodo se usa Arduino Ethernet que estará conectado directamente al router. Por ser un servidor su dirección IP es fija y no es asignada por el protocolo DHCP. Arduino Ethernet permite configurar desde su programación interna la dirección IP y MAC que se desean usar dentro de la red (figura 5.5).

servifor

```
#include <SPI.h>
#include <Ethernet.h>

byte mac[] = {0xDE, 0xAD, 0xBE, 0xEF, 0xFE, 0xED};
IPAddress ip(192, 168, 0, 155);
EthernetServer server(80);
```

Figura 5.5 Parámetros de red en Arduino

Los diferentes algoritmos de control son programados en este nodo. El controlador responde las diferentes peticiones del cliente (planta) encapsulando el resultado que se obtiene del PID en la respuesta del servidor hacia el cliente.

5.1.3 Cliente Sensor-Actuador

El Arduino Yún es el primer miembro de una nueva serie de placas Arduino que combinan la potencia de Linux junto con la sencillez característica de Arduino. Combina el chip del modelo Leonardo (ATMega32U4) junto con un módulo SOC (System-On-a-Chip) corriendo una distribución de Linux llamada Linino, basada en OpenWRT. Una de las características más interesantes es que soporta red cableada Ethernet y Wifi. Dispone de dos conexiones de red. Una red ethernet 10/100 mbps y otra Wifi (IEEE 802.11 b/g/n, 2,4GHz) que puede montarse como cliente o como punto de acceso.

En la planta se usa Arduino Yun que está conectado directamente al sensor y actuador, es decir, este nodo recibe la información de la planta en tiempo real. Para configurar Arduino Yun a la red de trabajo, primero se establece comunicación directamente vía Wifi con la red de nombre Dragino- XXX-XXXXX (nombre que viene dada por el fabricante). (figura 5.6)

Figura 5.6 Red wi-fi de Dragino Yun

En esta red se presenta una aplicación en la dirección IP 192.168.240.1 que permite configurar Arduino Yun a redes que se hallan dentro de su cobertura,

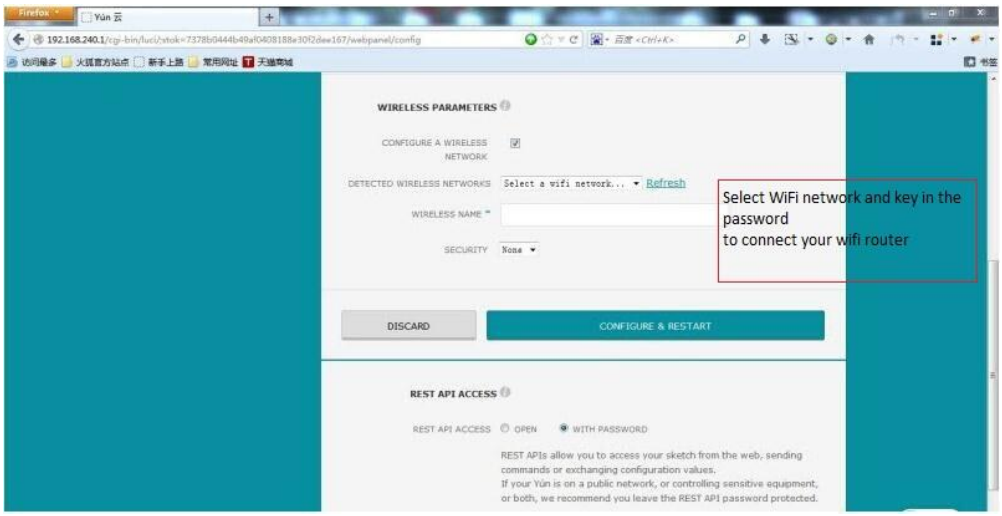


Figura 5.7 Configuración de Dragino Yun

Se identifica la red de trabajo y se introduce la contraseña. Una vez realizado esto el router otorga una dirección automáticamente a Arduino YUN.

La configuración de Arduino Yun como cliente es muy simple, se requiere de dos librerías inicialmente configuradas: HttpClient.h, que sirve para configurar el dispositivo como un cliente en la red, Bridge.h., Para comunicar el pequeño ATMega32U4 con el módulo Linux.

```
#include <Servo.h>
#include <Bridge.h>
#include <HttpClient.h>
#include <Console.h>
```

Figura 5.8 Librerías de Dragino Yun.

Para realizar peticiones al servidor se crea una variable tipo String que lleva la dirección del servidor y el valor de la variable del sensor (figura 5.9).

```
void loop() {

HttpClient client;
sensor();
envio = "http://192.168.0.155/dist:";
envio += distancia;

if (seguro== 0){

client.get(envio);
```

Figura 5.9 Dragnino Yun como cliente

Existe también un seguro que permite realizar peticiones al servidor solo si ciertas condiciones de la planta se cumplen (control por eventos). El cliente es el nodo que inicia la comunicación, el tiempo que existe entre actualización de información es de tipo asíncrono por la naturaleza de la red, incluso si no se llegan a cumplir las condiciones que activan el evento este tiempo puede ser infinito.

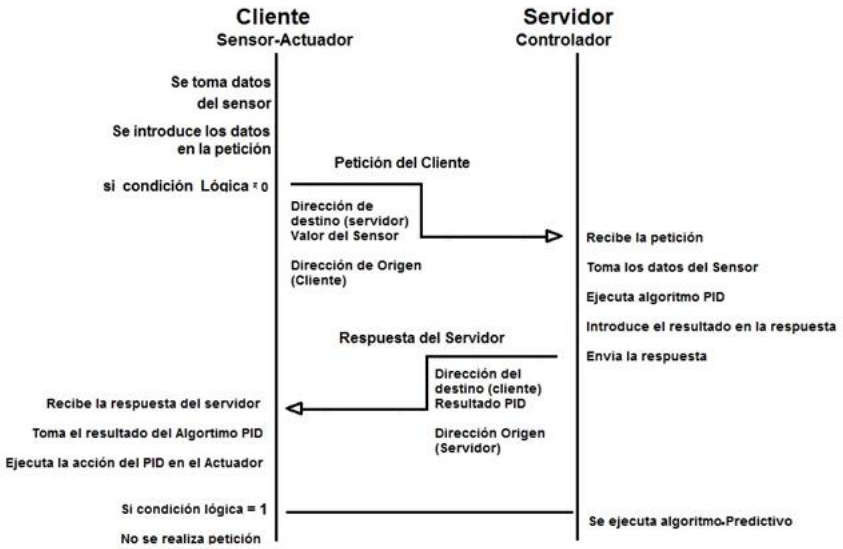


Figura 5.10 Comunicación cliente-servidor

5.2 Pruebas de funcionamiento de los algoritmos implementados en el controlador PID

El principal problema de un sistema de control en red es su tiempo de muestreo variable. La red de comunicación actualizara la información de una forma aleatoria por su naturaleza no determinística. Para procesos de control el periodo de muestreo es fundamental para tener un óptimo desempeño del algoritmo del controlador. Cuando más sensible se la variable medida a las acciones del actuador en el tiempo, se requiere de un tiempo de muestreo de menor valor. Por esta razón se desarrollaron las redes industriales, que son de carácter determinísticos. Existen dos áreas que presentan posibles soluciones: a) Comunicaciones: control por eventos, incremento de ancho de banda, subredes dedicadas, etc. b) Control: Algoritmos de control acoplados para trabajar sin tiempo fijo de actualización de datos, algoritmos predictivos, etc.

5.2.1 Algoritmo predictivo PID Asíncrono.

En el Algoritmo PID el tiempo de muestreo (T) permite calcular el componente integral y derivativo (figura 5.11). Se calcula el componente integral con la fórmula del área del trapecio, acumulando cada valor del área en la misma variable. T es la altura del trapecio, error es la base mayor y error_anterior sería la base menor. Para el factor derivativo T es la variación de tiempo que divide la variación de error.

```
void loop() {

 error = sp - distancia;
  proporcional=error;
  integral = integral + ((error+error_anterior)/2.0)*(T);
  derivativo = (error - error_anterior)/(T);

delay(T)

}
```

Figura 5.11 PID Asíncrono

La variable T es fija en su sistema clásico de control teniendo al error como el valor que cambia entre actualización de datos. En un sistema de control en red, el cliente (planta) realiza peticiones al servidor (controlador) que responderá con la respuesta del PID. El tiempo entre peticiones y respuestas es de tipo variable, es decir, T cambia con cada actualización de datos. Para el cálculo PID se tiene dos valores que cambian constantemente entre actualización de datos.

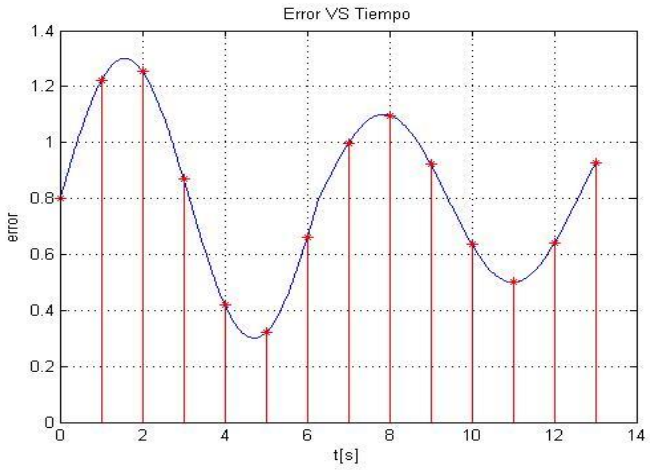


Figura 5.12 Muestreo síncrono

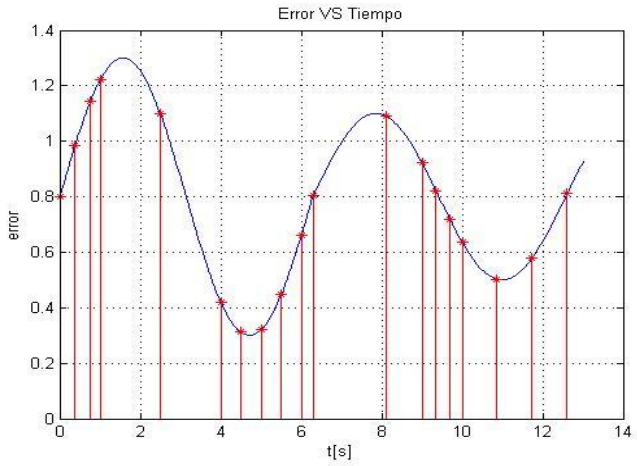


Figura 5.13 Muestreo Asíncrono

En un PID asíncrono es necesario obtener el valor de T para el cálculo del PID. Para tener el valor de T se usa un contador (dt) que incrementa cada milisegundo, en el que se mantiene el tiempo de actualización de información. El servidor al recibir información usa este valor almacenado para el cálculo PID y luego lo vuelve a cero para una futura petición (figura 5.14).

```
void loop() {

 EthernetClient client = server.available();
  if (client) {

  Servidor();
  error = sp - distancia;
  proporcional=error;
  integral = integral + ((error+error_anterior)/2.0)*(dt/1000);
  derivativo = (error - error_anterior)/(dt/1000);
  pid=(kp*proporcional)+(ki*integral) +(kd*derivativo);
  error_anterior=error;
   dt=0
  }


delay(1)
dt++
}
```

Figura 5.14 PID asíncrono

5.2.2 Algoritmo predictivo con PID síncrono y semi-sincrono

En un sistema donde razón de cambio de la variable medida respecto al tiempo es de un reducido valor, el PID asíncrono funciona óptimamente aun con la perdida de información. La acción del actuador no influye significativamente en el sistema en un corto tiempo.

Sin embargo existen sistemas que cambian sus parámetros en cortos lapsos de tiempo. Este es el principal problema de un sistema donde la red es no determinística, la perdida de información entorpece la acción de control. Toda la teoría de control justifica su trabajo en sistemas continuos o con periodos de muestreo fijos, aquí es donde se incorpora una nueva forma de acoplar un sistema totalmente asíncrono a uno síncrono o semi-síncrono, incorporando información en base a las respuestas del controlador durante el tiempo de actualización.

Durante la ausencia de datos el PID asíncrono necesita de información extra (tendencia) que se aproximara a la realidad incorporando datos en función de los resultados reales. Se introducirán valores ficticios para que el algoritmo de control trabaje incluso durante la ausencia de datos. Sin estos datos adicionales la respuesta del PID es abrupta y torpe, sin equilibrar el sistema. Beam-Ball es un sistema que requiere de tiempos de muestro muy bajos alrededor de 100ms, la variable medida cambia drásticamente en cortos lapsos de tiempo. Por esta razón se optó por trabajar con este sistema.

La lógica usada para calcular la función lineal de tendencia está previamente explicado en el literal 4.1.2 Diseño del controlador PID Predictivo para la planta BB. El muestreo semi-sincrono tiene un periodo fijo que esporádicamente aparecerá dependiendo de la naturaleza del sistema. Este periodo fijo no es constante pero al realizar un promedio de los tiempos de muestreo se observa que se encuentra próximo a este valor (figura 5.15)

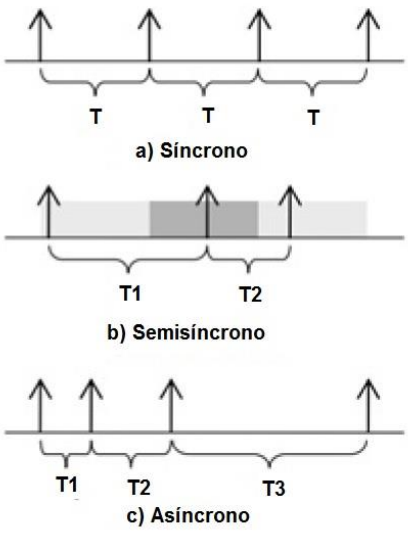


Figura 5.15 Tipos de muestreo

5.2.3 PID síncrono

Para convertir un sistema asíncrono en síncrono es necesario trabajar con información adicional durante la ausencia de datos reales. La función lineal estará constantemente trabajando en el sistema sin importar si la planta envía o no información. El PID tomara siempre los datos entregados por la función lineal que están basados en los datos de

salida y entrada del sistema (distancia, ángulo). De cierta manera el sistema no trabaja con datos auténticos sino una tendencia que presenta un error respecto a la realidad.

El programa realizado en Arduino cambia significativamente. El PID no solo trabaja cuando llegan datos, por ello se encuentra fuera de la condición (if (cliente)) que se activa cuando el servidor recibe una petición del cliente. También se crea el contador contador_f que ayuda a llevar el periodo virtual de trabajo (figura 5.16). Se crea un seguro para que el PID y la función lineal no trabajen cuando el cliente deje de enviar información si el tiempo de actualización de datos es mayor a 300 ms, tiempo de retardo máximo medido en la red implementada. El seguro se activara cuando el tiempo entre actualizaciones de datos sea mayor a 300ms, indicando que el sistema se equilibró (evento).

```
void loop() {
  if (client){
    seguro_dat=0;
    Servidor()
  }

delay(1);
 dt++;
contador_f++;
datos_falsos();
if (dt>=300 && seguro_dat==0){
dt=300;
contador_f=0;
seguro_dat=1;
pid=0;
}

}
```

Figura 5.16 PID síncrono

La función datos_falsos se cumple cada un milisegundo, la condición que existe al inicio de esta función usa el contador creado previamente para establecer un periodo fijo de trabajo. Solo cuando la variable contador_f es un múltiplo de 30 y el seguro sea igual a 0 el PID y la función lineal trabajaran.

El valor de tiempo de muestreo de 30ms fue tomado mediante análisis de trabajo con el sistema, el método usado directamente para el cálculo de la pendiente de la función lineal es empírico, basado directamente en prueba-error. Cabe recordar que se trata de un sistema estocástico.

El máximo tiempo de retardo que la red presente bordea los 300ms, es decir, se tendrían 10 datos adicionales entre actualización de información. También se podría mantener trabajando el sistema constantemente pero esto saturaría de información al PID obteniendo un resultado inestable.

La variable aumento_f se incrementa cada vez que ingresa al PID y sirve para llevar el valor del eje de las abscisas de la función lineal.

```
void datos_falsos(void){

  if (contador_f%30==0 && seguro_dat==0){

if (salida>= 105 && salida <= 140){
distancia_falsa= (-0.25*aumento_f)+distancia_pid;
if (distancia_falsa >22 ){
 distancia_falsa= 22;
}
if (distancia_falsa < 0){
 distancia_falsa= distancia_pid;
}
}


if (salida>= 45 && salida <= 90){

distancia_falsa= 0.25*aumento_f+distancia_pid;
if (distancia_falsa < 0){
 distancia_falsa= distancia_pid;
}
if (distancia_falsa >22 ){
 distancia_falsa= 22;
}

}

if (salida>90  && salida < 105){
distancia_falsa= distancia_pid;

}

  pid_eventos();
 aumento_f++;

if (aumento_f >=10){
aumento_f=0;

}
```

Figura 5.17 Función de datos falsos

La información que se usa entre actualización es dada por la función de lineal la cual trabaja con un nuevo contador aumento_f (figura 5.17).

- Cuando el ángulo del servo motor está dentro del intervalo [45 , 90) el valor de la pendiente es de 0.25.
- Cuando el ángulo del servo motor está dentro del intervalo (105 , 140] el valor de la pendiente es de -0.25.
- Cuando el ángulo del servo motor está dentro del intervalo [90 105] se mantiene el valor de la última distancia real.

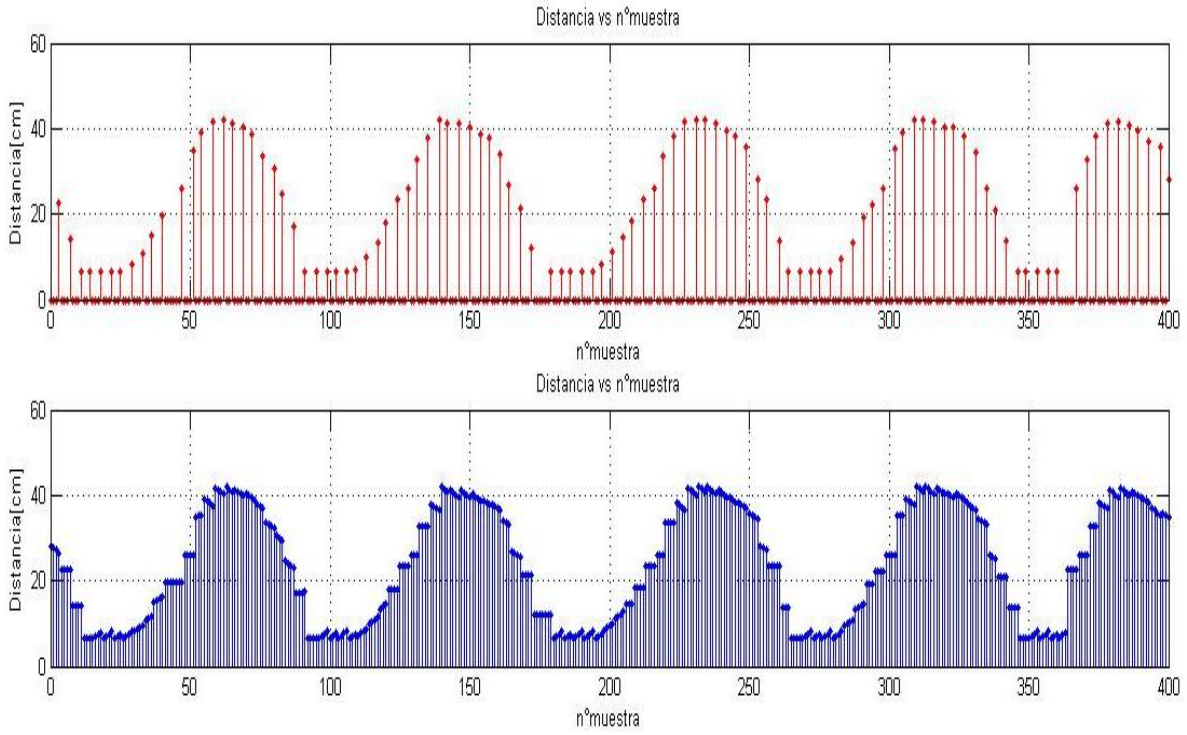


Figura 5.18 Datos reales de la planta (figura de arriba), introducción de nuevos datos con la función lineal(fig. abajo) Datos obtenidos al hacer oscilar el sistema

La función pid_eventos es llamada cada 30ms, por obvias razones cambia en comparación al PID asíncrono. La variable dt ya no se usa para el cálculo del PID, sino directamente el tiempo del periodo establecido (figura 5.19)

```
void pid_eventos(void){
  error = sp - distancia_falsa;
  proporcional=error;
  integral = integral + ((error+error_anterior)/2.0)*(0.03);
  derivativo = (error - error_anterior)/(0.03);
}
```

Figura 5.19 PID síncrono con período fijo

5.2.4 PID semi-síncrono

Para convertir al sistema en semi-síncrono se crea un nuevo contador t_semisincrono (figura 5.20). Este nuevo contador se usa para realizar el cálculo del pid, es decir, se tiene nuevamente un tiempo de muestreo variable.

```
void loop() {
  if (client){
    seguro_dat=0;
    Servidor()
  }

delay(1);
 dt++;
contador_f++;
t_semisincrono++;
datos_falsos();
if (dt>=300 && seguro_dat==0){
dt=300;
contador_f=0;
seguro_dat=1;
pid=0;
}

}
```

Figura 5.20 PID semi-síncrono

En la función de datos falsos se añade nuevos condiciones para que la función lineal y el pid trabajen (figura 5.21).

```
void datos_falsos(void){

  if (contador_f%30==0 && seguro_dat==0  || (dt==1 && seguro_dat==0)){
```

Figura 5.21 Condiciones para PID semi-síncrono

La función es exactamente la misma con la diferencia de que las condiciones de ingreso a la función de datos falsos es distinta. No solo se cumple el algoritmo cuando es un múltiplo de 30 sino cuando existe una petición del cliente, esta condición se puede conocer por medio de la variable dt que lleva el tiempo de actualización.

```
void pid_eventos(void){
  error = sp - distancia_falsa;
  proporcional=error;
  integral = integral + ((error+error_anterior)/2.0)*(t_semisincrono/1000.0);
  derivativo = (error - error_anterior)/(t_semisincrono/1000.0);
   t_semisincrono=0;
}
```

Figura 5.22 Condiciones para PID semi-síncrono

Para el cálculo PID se usa la variable t_semisincrono, una vez realizado el cálculo esta variable regresara a cero (Figura 5.22).

5.3 Cliente sensor-actuador

Un evento se puede definir como una condición lógica que bajo ciertas condiciones entrega un 0 o 1 lógico al sistema. Dependerá del diseñador establecer los criterios y parámetros de trabajo de la condición ya que no existe una norma definida, el método de calibración es basándose directamente en prueba y error al trabajar con el sistema. Esta condición directamente controla el envío-recepción de datos, liberando el medio de comunicación de trabajo innecesario.

Al trabajar bajo la arquitectura cliente-servidor el cliente es el que tiene el control del medio de comunicación, al realizar una petición el servidor responderá inmediatamente por el contrario bajo la ausencia de la misma se detendrán las comunicaciones. La condición permite que se realicen o no peticiones al servidor.

A continuación se analizan las diferentes condiciones aplicadas, cabe destacar que para el estudio de eventos se usó el PID asíncrono con el objetivo de observar la funcionalidad del sistema con los retardos propios de la red. Los diferentes límites y variaciones para las condiciones de eventos, fueron tomados observando que cumplan con los requisitos de equilibrio del sistema.

El PID trabajo con los siguientes coeficientes Kp=2.65, ki=0.01,kd=0.75.

5.3.1 Condición intervalo de trabajo temporizado

Se establece una franja de trabajo para activar la condición lógica cuando la pelota se encuentre dentro de un rango cercano al set point. Un contador se activa cuando se encuentra dentro de la franja, cuando alcanza un límite definido dejara de realizar peticiones.

```
if (seguro== 0){
client.get(envio);
while (client.available()) {
    char c = client.read();
    angulo += c;
    if (c == '\n')
    {
    angulo_in=angulo.toInt();
    angulo="";
    }
   Console.println(contador);
  }
    envio="";
}
```

Figura 5.23 Condición de eventos (seguro)

En el cliente se programa la condición seguro (figura 5.23) que permite enviar peticiones al servidor con los datos de la planta.

```
if (distancia > 12.0 && distancia <17.0 ){
  contador++;
}else{
  contador=0;
  seguro=0;
}
  if (contador>=11)
    {
      seguro =1;
    }

if (contador >= 12){
  contador=0;
}

  motor.write(angulo_in);
   delay(1);
```

Figura 5.24 Programación del intervalo temporizado

Se establece los intervalos de la franja (12,17) teniendo un set point de 15, el contador empieza a incrementar. Cuando el contador excede un valor definido el seguro se activa y las peticiones cesan. El valor de 11 es equivalente a 7 segundos, este valor se da porque el contador se demora al procesar la información que se envía y recibe del servidor (figura 5.24).

El seguro y el contador regresan a su estado inicial cuando una perturbación le permita salir de la franja de trabajo, permitiendo que se realicen nuevas peticiones para estabilizar el sistema. En la figura 5.25 se puede observar el funcionamiento del PID, el periodo de actualización varía dependiendo de la red, el sistema se estabilizo a los 5250 ms el pico máximo es de 22.98 [cm]. El contador detiene automáticamente el envió de datos, se observa que el proceso duro alrededor de 8229 [ms].

El tiempo que trabajo innecesariamente seria :

TI= tiempo de trabajo – tiempo de estabilidad = 2979 ms

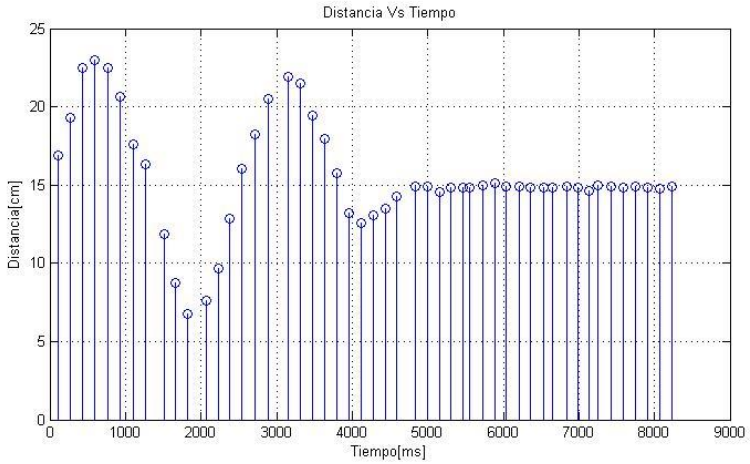


Figura 5.25 PID asíncrono con intervalo de trabajo temporizado

En la figura 5.26 se observa los retardos durante cada actualización de datos.

Máximo tiempo de retardo	253[ms]
Mínimo tiempo de retardo	91[ms]
Promedio	164.58[ms]

Tabla 5.1 Datos de los tiempos de retardo en el PID asíncrono con intervalo de trabajo temporizado

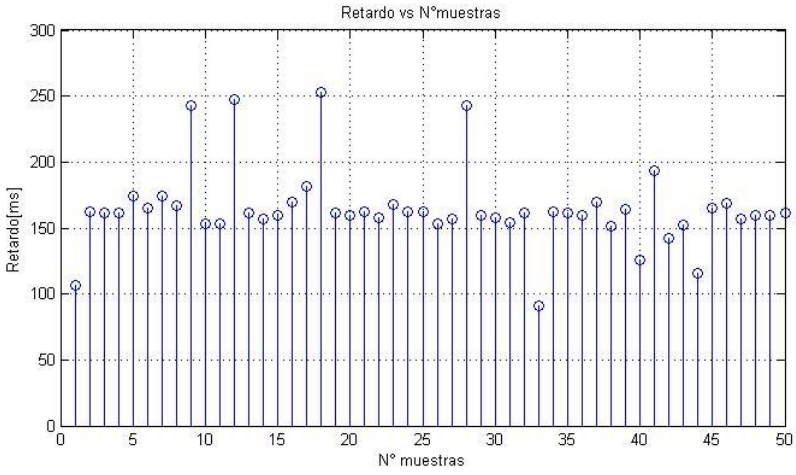


Figura 5.26 Retardo del PID asíncrono con intervalo de trabajo termporizado

5.3.2 Condición de variación de la variable medida

El contralador PID tiende a equilibrar el sistema, se observa que las oscilaciones del sistema tienen una menor magnitud en el tiempo. La razón de cambio entre el valor de la variable medida actualmente y(t) y la anterior y(tlast) tiene un valor menor al acercarse al set_point.

$$|y(t) - y(t_{last})| \geq \Delta \qquad (5.1)$$

Es lógico pensar que una vez equilibrado el sistema la razón de cambio o variación será cero. Esta condición es usada para disparar el evento que activa o desactiva las comunicaciones entre cliente y servidor.

A diferencia de la condición anterior no existe tiempo pero si hay una franja de trabajo. Esta franja donde se activa la condición de variación se encuentra situado entre intervalos del set_point. Se programa la variación entre cada actualización de datos en la variable vd (figura 5.27), la diferencia entre el valor actual y el anterior se almacenan en esta variable.

```
vd=abs(distancia-distancia_anterior);
envio = "http://192.168.0.155/dist:";
envio += distancia;

if (seguro== 0){

client.get(envio);


while (client.available()) {
    char c = client.read();
    angulo += c;
```

Figura 5.27 Variación de la variable medida (vd)

Cuando el seguro es cero el sistema sigue enviando peticiones, una vez que la pelota se encuentra en el intervalo de (13,16.5) y la variación es menor a 0.09 el seguro cambia a uno. La franja se continúa usando debido a que pueden existir variaciones muy pequeñas en distancias muy alejadas al set point deteniendo al sistema innecesariamente. Al no cumplir alguna de las condiciones el sistema solicitara nuevamente una petición al controlador (figura 5.28).

```
if (distancia> 13.0 && distancia <16.5 ){

if (vd <=0.09)
   {
     seguro=100;
   }


}else{
if (vd>= 0.1 ){
  seguro=0;
}
```

Figura 5.28 Condiciones para activación de eventos

En la figura 5.29 se puede observar el funcionamiento del PID, el periodo de actualización varía dependiendo de la red. El sistema se estabilizo a los 6903 [ms] el pico máximo es de 23.98 [cm]. El tiempo de trabajo total es 7630 [ms].

El tiempo que trabajo innecesario seria:

TI= tiempo de trabajo – tiempo de estabilidad; TI=727ms

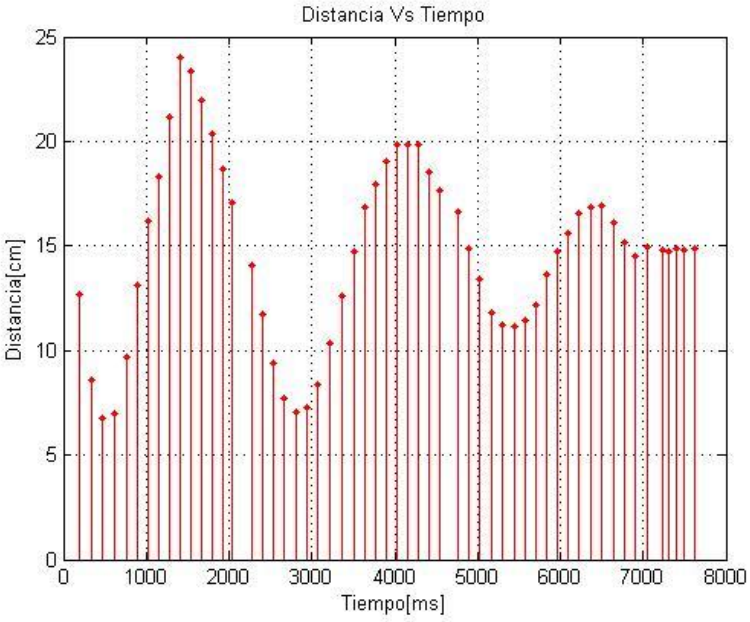


Figura 5.29 PID asíncrono con variación de variable medida.

En la figura 5.30 se observa los retardos durante cada actualización de datos.

Máximo tiempo de retardo	223[ms]

Mínimo tiempo de retardo	117[ms]
Promedio	138.22[ms]

Tabla 5.2 Datos de los tiempos de retardo en PID asíncrono con variación de la variable medida

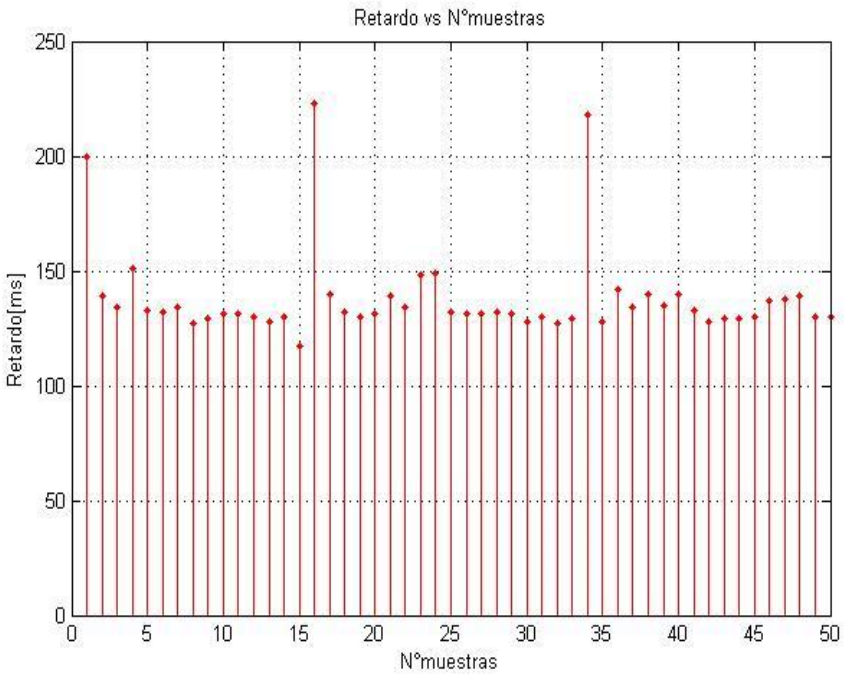


Figura 5.30 Retardos del PID asíncrono con variación de la variable medida

5.3.3 Condición de acumulación del error

El error es el parámetro que activara o desactivara el evento, es lógico pensar que cuando el sistema tiende a equilibrarse el valor del error disminuye paulatinamente. Al tener un sistema oscilante el error aumentara y disminuirá constantemente, es por esta situación que la acumulación del error solo entra a trabajar en una franja de trabajo.

$$\sum e(t) > \Delta \qquad (5.2)$$

El programa es exactamente el mismo con la diferencia de que la condición que activa o desactiva el evento trabaja con la acumulación del error

```
error=abs(15.0- distancia);
 contador_integral++;
if (error >=0 && error <2.5 ){
   if (valor>=7.0)
    {
      seguro =1;
    }
}else{
 if (valor<7.0){
  seguro=0;
  contador_integral=0;
  resultado=0.0;
  valor=0.0;
 }
}
if (contador_integral>=900){
  contador_integral++;
  valor=valor+error;

}


delay(1);
```

Figura 5.31 Programación del error integral.

Se calcula el error con el set_point establecido previamente, cuando el error se encuentra entre 0 y 2.5, es decir, en los rangos de 13.5 y 17.5 la variable contador _integral++ incrementa cada 1 milisegundo. Cada vez que pasa un segundo toma el error y lo acumula en la variable valor que es la acumulación del error, en la condición de la acumulación se observa que trabaja cuando es 900ms esto es debido a que la comunicación wi-fi introduce retardo aleatorio que se compensa con esta reducción de tiempo. El seguro que detiene la comunicación se activa cuando el valor de la integral es mayor a 7 caso contrario al no cumplirse alguna condición el sistema automáticamente pedirá nuevas acciones de control (figura 5.31)

En la figura 5.32 se puede observar el funcionamiento del PID, el periodo de actualización varía dependiendo de la red, el sistema se estabilizo a los 6195 [ms] el pico máximo es de 23.94 [cm]. El tiempo de trabajo total es 8365 [ms]

El tiempo que trabajo innecesariamente seria:

TI= tiempo de trabajo – tiempo de estabilidad; TI=2170ms

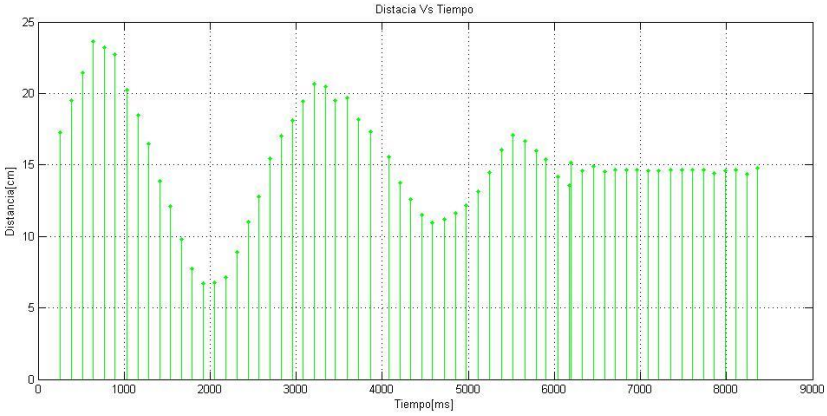


Figura 5.32 PID Asíncrono con error integral

En la figura 5.32 se observa los retardos durante cada actualización de datos.

Máximo tiempo de retardo	259 [ms]
Mínimo tiempo de retardo	18[ms]
Promedio	130.7[ms]

Tabla 5.3 Datos del tiempo de retardo en PID asíncrono con acumulación del error

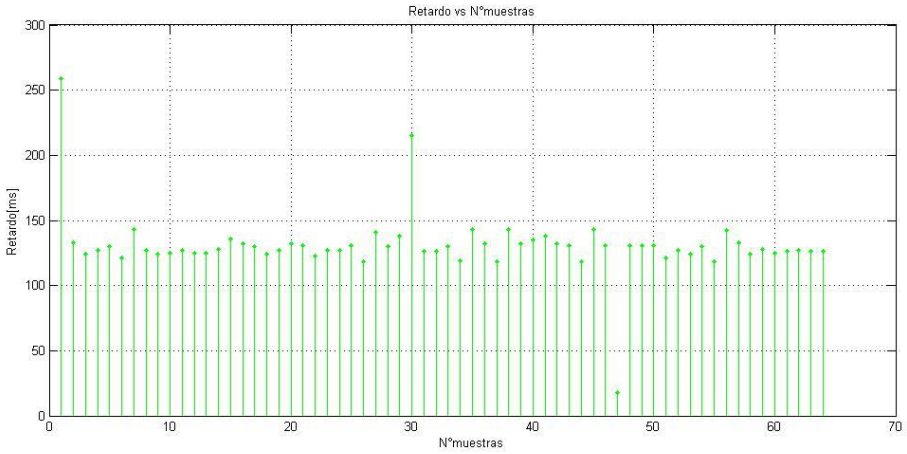


Figura 5.33 Retardos del PID asíncrono con error integral

5.3.4 Condición de acumulación del error al cuadrado

El error es el parámetro que activara el evento, a diferencia de la condición anterior el error es elevado al cuadrado. Cuando la variación del de este parámetro exceda un valor determinado, la planta detendrá la las peticiones al servidor.

$$\sum e^2(t) > \Delta \qquad (5.3)$$

El valor del error integral al cuadrado para activar el evento aumenta a 11 para estabilizar el sistema (figura 5.34).

```
  error =abs(distancia-15.0);
    error_cuadrado = error*error;
  contador_integral++;
  if (error >=0 && error <2.5 ){
   if (valor>=11.0)
    {
      seguro =1;
    }
}else{
 if (valor<11.0){
  contador=0;
  seguro=0;
contador_integral=0;
  resultado=0.0;
  valor=0.0;
 }
}

if (contador_integral >= 900){
  contador_integral=0;
  valor=valor+error_cuadrado;
}
```

Figura 5.34 Programación de error integral al cuadrado

En la figura 5.35 se puede observar el funcionamiento del PID, el periodo de actualización varía dependiendo de la red, el sistema se estabilizo a los 7217 [ms] el pico máximo es de 23.89 [cm]. El tiempo de trabajo total es 8020 [ms].

El tiempo que trabajo innecesariamente seria:

TI= tiempo de trabajo – tiempo de estabilidad; TI=803 [ms]

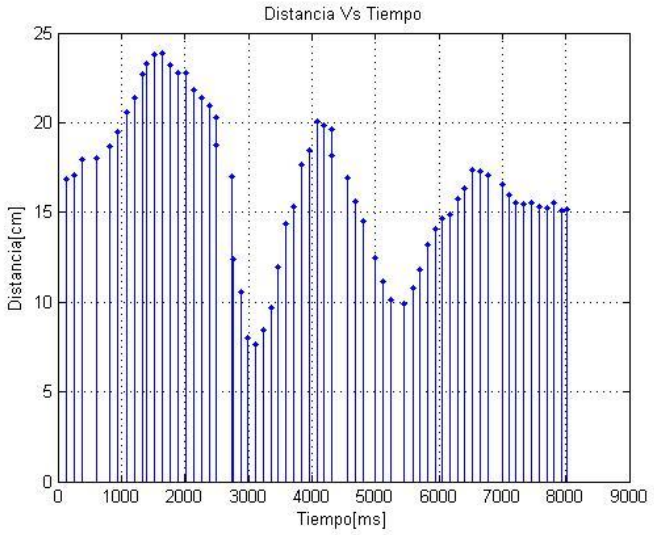


Figura 5.35 PID Asíncrono con error integral al cuadrado

En la figura 5.36 se observa los retardos durante cada actualización de datos.

Máximo tiempo de retardo	227 [ms]
Mínimo tiempo de retardo	21[ms]
Promedio	125.31[ms]

Tabla 5.4 Datos del tiempo de retardo en PID asíncrono con error integral al cuadrado

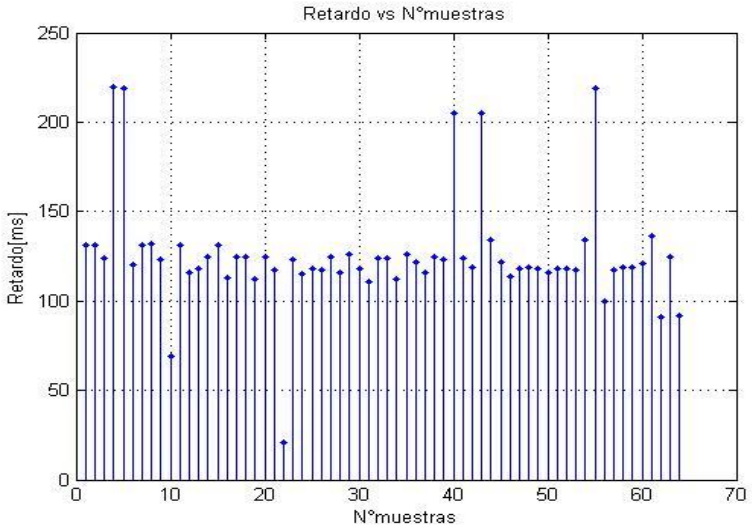


Figura 5.36 Datos de los tiempos de retardo en PID asíncrono con error integral al cuadrado

5.4 Definición de la mejor condición para el evento del sistema

Al trabajar con un sistema de control en una red no determinístico, el tiempo de estabilidad y trabajo son los parámetros que nos permiten definir que condición usa mejor los recursos del sistema, es decir, el algoritmo tiene la inteligencia necesaria para pedir o no información a la central de control.

Una de las grandes ventajas de control por eventos es ahorrar recursos del sistema. El evento a más de detener la comunicación entre planta y controlador también detiene los cálculos o algoritmos que no son necesarios cuando el sistema ha alcanzado el equilibrio. Para establecer que condición es la mejor se debe observar cuanto tiempo estuvo trabajando innecesariamente.

Tiempo de estabilidad es variante pero esto no depende de las condiciones de los eventos, esto se debe a la naturaleza aleatoria del tiempo de muestreo. Al observar la tabla 5.5 los valores de promedio del retardo se encuentra en un intervalo de (125.31 [ms], 164. 58 [ms]) y el valor máximo de retardo se encuentra en el intervalo de (227 [ms], 259 [ms]). Al obtener estos valores se puede afirmar que las diferentes condiciones para eventos trabajaron con una red que tiene una tendencia o cierta similitud al actualizar la información.

El tiempo innecesario de trabajo varía dependiendo de la condición para el evento, se puede observar en la tabla 5.5 que el menor valor es de la condición de variación de la variable medida por obvias razones es la que más ahorra recursos al sistema.

Condición	Tiempo de trabajo [ms]	Tiempo de estabilidad [ms]	Retardo (Promedio) [ms]	Máximo Retardo [ms]	Tiempo innecesario de trabajo[ms]
Intervalo de trabajo temporizado	8229	5250	164.58	253	2979
Variación de la Variable medida	7530	5003	138.22	223	727
Acumulación del error	8365	5195	130.7	259	2170
Acumulación del error al cuadrado	8020	7217	125.31	227	803

Tabla 5.5 Datos de las condiciones para eventos

5.5 PID síncrono con evento activado por variación de la variable medida.

Previamente se analizó el PID asíncrono para obtenerla mejor condición para activar el evento, se mantiene la condición en el nodo de la planta y se cambia el algoritmo de control en el controlador.

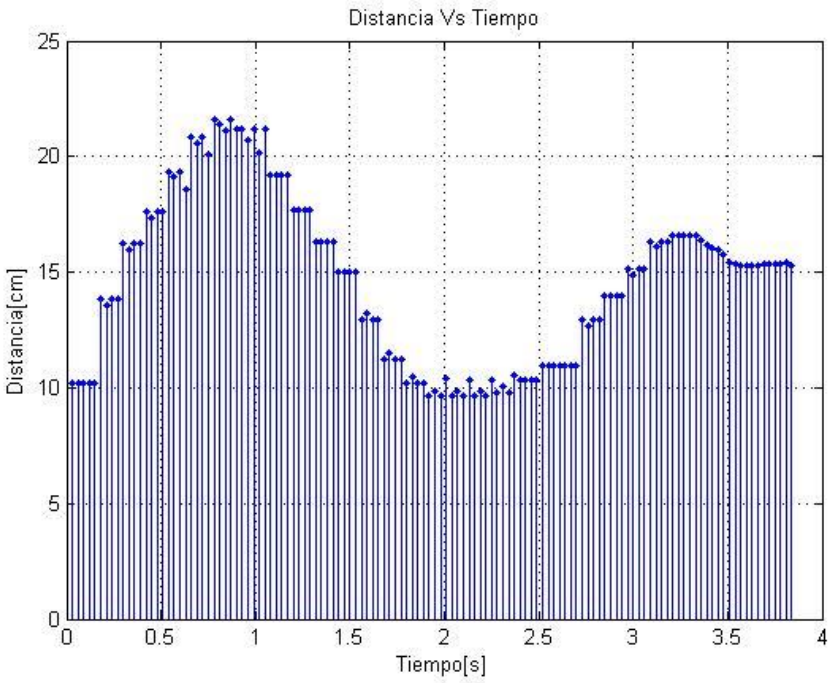


Figura 5.37 PID síncrono con evento activado por variación de la variable medida

El PID siempre está trabajando cada 30ms incluso durante la ausencia de información, se puede observar cómo se incluye información adicional al proceso de control (figura A.25).

Los coeficientes usados son kp=2.5,ki=0.09,kd=0.32. El tiempo de trabajo es de 3.84 [s] y de estabilidad es de 3.57 [s],el tiempo innecesario de trabajo es de 0.27[s]. Con estos resultados se obtiene un sistema más veloz y eficiente al equilibrar el sistema. En comparación al PID asíncrono el tiempo de estabilidad y trabajo innecesario bajan considerablemente.

Los datos introducidos por la función lineal complementan el cálculo del PID durante la ausencia de datos, al ver la figura 5.37 se observa una coherencia en el desplazamiento de la pelota. Es muy difícil diferenciar cual es el dato original o adicional.

5.6 PID semi-síncrono con evento activado por variación de la variable medida.

En el PID semi-síncrono el tiempo de muestreo no es fijo, pero existe una tendencia que mantiene un valor que se repite eventualmente. El PID trabaja cada 30 [ms] y también cada vez que un dato original de la planta llega a la central del control, es decir, existe nuevamente retardos aleatorios.

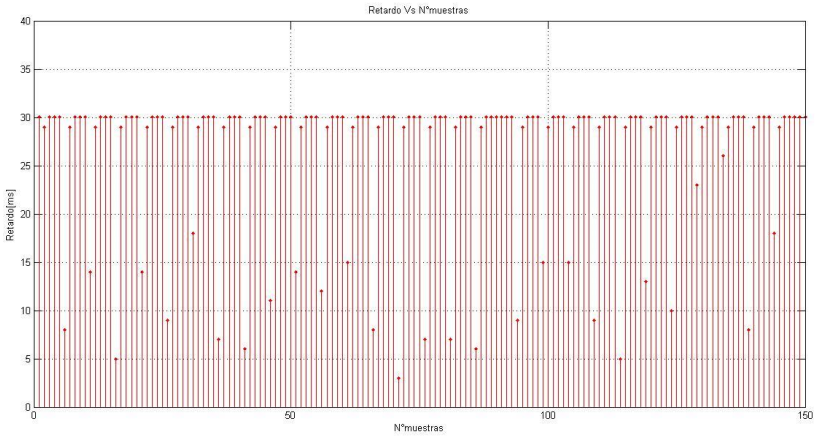


Figura 5.38 Retardo del PID semi-sincrono

En la figura 5.38 se observa los diferentes retardos de la red, el valor que se repite eventualmente es de 30[ms].

Promedio de los retardos del PID semisincrono	25.61[ms]

Tabla 5.6 Datos de los tiempos de retardo en PID semisíncrono con evento activado por variación de la variable medida.

En la figura 5.39 se observa el PID semi-síncrono, el tiempo de trabajo es de 5419 [ms] y el tiempo de estabilidad es de 4803 [ms], es decir, que el tiempo de trabajo innecesario es de 616 [ms].

Los coeficientes utilizados son kp=2.5,ki=0.09,kd=0.3

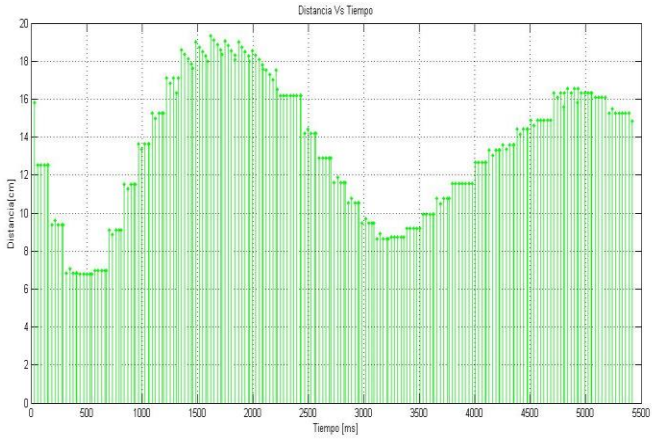


Figura 5.39 PID semi-síncrono con variación de la variable medida

Cada uno de los sistemas tiene diferentes propiedades debido al tiempo de actualización. Los coeficientes para los diferentes PID varían significativamente. Se puede observar que los parámetros de estabilidad y equilibrio del sistema varían. El PID con menor tiempo de estabilidad es el PID síncrono.

PID	kp	ki	kd	tiempo de trabajo[ms]	Tiempo de estabilidad[ms]	Tiempo innecesario[ms]
Asíncrono con variación en la variable medida	2.65	0.01	0.75	7530	5903	727
Semi-síncrono con variación en la variable medida	2.5	0.09	0.3	5419	4803	616
Síncrono con variación en la variable medida	2.5	0.09	0.32	3840	3570	270

Tabla 5.7 Datos del PID síncrono, semi-síncrono y asíncrono

La tabla 5.7 indica que el PID con mejor tiempo de estabilidad es el PID síncrono, es lógico pensar que los coeficientes cambiarían debido a que el valor integral y derivativo utilizan el tiempo de actualización para su cálculo. Los datos introducidos adicionalmente ayudan al PID a intuir una repuesta futura mejorando la reacción del sistema para equilibrarlo.

Se debe de tomar en cuenta que la red tiene un retardo oscilante con picos de 259[ms]. Otras redes pueden presentar otro tipo de comportamiento donde posiblemente los algoritmos utilizados deban ser acoplados para funcionar óptimamente.

6 CONCLUSIONES

- El estado de la técnica de control PID basado en eventos ha sido analizado y se ha proporcionado los pseudo códigos para poner en práctica los diferentes algoritmos PID, es evidente que el paradigma basado en eventos se puede adaptar a una ley de control, proporcionando muchas ventajas que tienen en cuenta la optimización de recursos, sin embargo, existe problemas en la implementación de un PID basado en eventos debido a que no hay un algoritmo estandarizado, así como existen diferentes maneras para producir la activación del generador de entrada de control. En este trabajo se ha considerado una alternativa útil en la aplicación de WNCS para la planta Beam-Ball.

- En este trabajo se realizó la simulación de la sintonización de un controlador PID para el proceso Beam-Ball, mediante dos algoritmos heurísticos: el Algoritmo Genético (AG) y la Optimización por Cúmulo de Partículas (PSO) los cuales presentan un desempeño satisfactorio para sintonizar controladores PID en línea, además de encontrar soluciones satisfactorias estos también encuentran la solución en pocas interacciones. Sin embargo, en la aplicación es primordial el desempeño del sistema, como es el caso del sistema Ball and Beam entonces el algoritmo PSO es la herramienta que presenta ventajas sobre el algoritmo genético.

- En los sistemas de control convencional el tiempo es una constante que influye en el cálculo, en el muestreo, en el proceso, etc. En los sistemas de control asíncrono en red el tiempo es una variable aleatoria sin ningún patrón o coherencia, la dificultad de estos sistemas es trabajar con dos variables para el cálculo del algoritmo de control (la variable medida y el tiempo de muestreo). Esperar una respuesta igual entre los dos sistemas será un imposible, sin embargo, el criterio para establecer un sistema de control óptimo se mantiene, es decir, el tiempo en que el sistema se equilibra, pero en los sistemas de control basados en eventos se debería considerar el “timestamp” del sistema, para lo cual se consideró utilizar contadores separados tanto en el generador de entrada como en el detector de eventos, de tal manera que el sistema que presenta el menor valor en estos tiempos se define como un sistema rápido, confiable y que ahorra recursos.

- En este proyecto se aporta con alternativas para la implementación del algoritmo PID en sistemas asíncronos, semi-síncrono y síncrono. Los coeficientes varían considerablemente debido a que las variables ki y kd dependen directamente del tiempo de muestro. El PID síncrono y semi-síncrono tienden a estabilizar el sistema de mejor manera respecto al PID asíncrono, esto es debido a la función lineal o tendencia introducen datos virtuales para compensar la ausencia de información de la planta, permitiendo ejercer un control más exacto y preciso ya que el controlador intuye una repuesta futura de la planta por su acción.

Recomendaciones y Trabajo Futuro

- Se recomienda que para realizar el diseño de controladores PID basados en eventos para los procesos, se debe asociar una condición basada en eventos de cada sensor o una combinación de varios sensores para activar el generador de entrada de control, también es importante definir las condiciones basadas en eventos, los valores de umbral para la activación del controlador.

- En realidad, como ocurre en las aplicaciones prácticas de los enfoques basados en eventos, este controlador es una aproximación híbrida entre uno de tiempo continuo y uno basado en eventos. Si se cumple la condición basada en eventos en el detector de eventos entonces se invoca una nueva acción de control dada por el generador de entrada de control. Tenga en cuenta que si la frecuencia de activación del generador de eventos se aumenta al máximo en una implementación real, este controlador se convierte en un enfoque basado en eventos cuasi-pura, es decir en teoría continuo.

- Para el control del sistema Beam-Ball, se implementó un prototipo, del cual se obtuvo los valores de entrada/salida experimentalmente, para con estos valores obtener la función de transferencia del sistema a simular, junto con el sistema de control, de esta manera se diseñó el sistema en cascada, el cual se convierte en punto de partida para posteriores desarrollos, por lo que se recomienda que para otras aplicaciones se debe partir de la medición de las variables, datos de entrada-salida para poder obtener una función de transferencia apegada a la realidad mediante la utilización de la herramienta ident de Matlab.

- En este trabajo los parámetros fueron estimados de forma experimental. Sin embargo, se sabe que los parámetros pueden sufrir cambios por un deterioro físico, a esto en control se dice que los parámetros son variantes en el tiempo. Por lo tanto, si los parámetros sufren algún cambio entonces el desempeño del sistema Ball- Beam también será afectado aun cuando el algoritmo basado en eventos presente soluciones satisfactorias. Para contrarrestar este problema, este trabajo puede ser complementado con sintonizador en línea que estime los parámetros.

- Condiciones para el control por eventos otorga una inteligencia para establecer comunicación entre el controlador-servidor, es de vital importancia elegir la mejor opción para el eficiente desempeño del sistema. Dependiendo de la condición seleccionada el tiempo de trabajo innecesario variará sometiendo a la red, procesadores, memoria, etc. a diferentes cargas de trabajo. Para el sistema BB implementado la condición de la variación de la variable medida es la mejor opción ya que el tiempo de trabajo innecesario presenta el menor valor respecto a las otras condiciones.

BIBLIOGRAFÍA

[1] J. Lunze, *Control Theory of Digitally Networked Systems*. 2014.

[2] J. P. Hespanha, P. Naghshtabrizi, and Y. Xu, “A survey of recent results in networked control systems,” *Proc. IEEE*, vol. 95, no. 1, pp. 138–172, 2007.

[3] K. J. Åström, C. C. Hang, and P. Persson, “Towards intelligent PID control,” *Annu. Rev. Autom. Program.*, vol. 15, no. PART 2, pp. 53–58, 1989.

[4] S. O. Santos, “Sintonización de un controlador PID basado en un algoritmo heurítio para el control de un Ball and Beam,” *J. Chem. Inf. Model.*, vol. 53, no. 9, pp. 1689–1699, 2014.

[5] C. Yajuan and W. Qinghai, “Design of PID controller based on PSO algorithm and FPGA,” *2011 2nd Int. Conf. Intell. Control Inf. Process.*, vol. 2, no. 2, pp. 1102–1105, 2011.

[6] U. de C. Fernandez Barcell, Manuel, “Introducción a las redes de sensores inalámbricas,” *Wirel. Sens. Netw.*, pp. 1–20, 2008.

[7] C. Chong, S. P. Kumar, and S. Member, “Sensor Networks : Evolution , Opportunities , and Challenges,” vol. 91, no. 8, 2003.

[8] M. Dunbar, “Plug-and-play sensors in wireless networks,” *IEEE Instrum. Meas. Mag.*, vol. 4, no. 1, pp. 19–23, 2001.

[9] E. E. Flores Carbajal, “Redes de Sensores Inalámbricas Aplicado a la Medicina,” p. 91, 2012.

[10] M. García, “Análisis y diseño de una red inalámbrica de sensores para un proyecto agrario Índice,” 2014.

[11] J. Almeida, C. Silvestre, and A. M. Pascoal, “Self-triggered state-feedback control of linear plants under bounded disturbances,” *Int. J. Robust Nonlinear Control*, vol. 25, no. 8, pp. 1230–1246, 2015.

[12] A. Hern, N. Master, and D. P. Stockholm, “Wireless Inverted Pendulum using IEEE,” 2011.

[13] S. K. Mazumber, *Wireless Networking Based Control*. Chicago, Illinois, USA: Springer, 2011.

[14] J. R. Moyne and D. M. Tilbury, “The emergence of industrial control networks for manufacturing control, diagnostics, and safety data,” *Proc. IEEE*, vol. 95, no. 1, pp. 29–47, 2007.

[15] M. Johansson and R. Jäntti, “Wireless networking for control: Technologies and models,” in *Networked Control Systems*, Springer, 2010, pp. 31–74.

[16] L. Litz, O. Gabel, and I. Solihin, “NCS-controllers for ambient intelligence networks-control performance versus control effort,” in *Proceedings of the 44th IEEE Conference on Decision and Control*, 2005, pp. 1571–1576.

[17] S. Bhuyan, B. Subudhi, and S. Ghosh, “Networked Control System: Uncertainty Modeling and Stabilization,” *Procedia Eng.*, vol. 38, pp. 3662–3667, 2012.

[18] P. Antsaklis and J. Baillieul, “Special issue on technology of networked control systems,” *Proc. IEEE*, vol. 95, no. 1, pp. 5–8, 2007.

[19] F.-Y. Wang and D. Liu, *Networked control systems : theory and applications*. 2008.

[20] J. Baillieul and P. J. Antsaklis, “Control and Communication Challenges in Networked Real-Time Systems,” *Proc. IEEE*, vol. 95, no. 1, pp. 9–28, 2007.

[21] Z. Wei, M. S. Branicky, and S. M. Phillips, “Stability of networked control systems,” *Control Syst. IEEE*, vol. 21, no. 1, pp. 84–99, 2001.

[22] D. Antunes, J. P. Hespanha, and C. Silvestre, “Stochastic networked control systems with dynamic protocols,” *Asian J. Control*, vol. 17, no. 1, pp. 99–110, 2015.

[23] T. Donkers, M. Heemels, N. Van De Wouw, and L. Hetel, “Stability Analysis of Networked Control Systems Using a Switched Linear Systems Approach,” 2011.

[24] G. C. Walsh, Y. Hong, and L. G. Bushnell, “Stability analysis of networked control systems,” *IEEE Trans. Control Syst. Technol.*, vol. 10, no. 3, pp. 438–446, 2002.

[25] M. Donkers, W. Heemels, and D. Bernardini, “Stability analysis of stochastic networked control systems,” *2010 Am. Control Conf.*, vol. 48, no. 5, pp. 3684–3689, 2012.

[26] Y. Shi and B. Yu, “Output Feedback Stabilization of Networked Control Systems With Random Delays Modeled by Markov Chains,” *Autom. Control. IEEE Trans.*, vol. 54, no. 7, pp. 1668–1674, 2009.

[27] M. B. G. Cloosterman, L. Hetel, N. van de Wouw, W. P. M. H. Heemels, J. Daafouz, and H. Nijmeijer, “Controller synthesis for networked control systems,” *Automatica*, vol. 46, no. 10, pp. 1584–1594, 2010.

[28] X. Wang and M. D. Lemmon, “Event-eriggering in distributed networked control systems,” *IEEE Trans. Autom. Contr.*, vol. 56, no. 3, pp. 586–601, 2011.

[29] P. Tabuada, “Event-triggered real-time scheduling of stabilizing control tasks,” *IEEE Trans. Automat. Contr.*, vol. 52, no. 9, pp. 1680–1685, 2007.

[30] K. J. Astrom, “Event Based Control,” Springer Science & Business Media, pp. 127–147.

[31] K. J. Åström and B. Bernhardsson, “Comparison of Riemann and Lebesque sampling for first order stochastic systems,” in *Proceedings of the 41st IEEE Conference on Decision and Control, 2002*, 2002, vol. 2, pp. 2011–2016.

[32] D. L. J. L. K. H. Johansson, “Comparison between sampled-data control , deadband control and model-based event-triggered control,” vol. 7, no. Adhs 12, pp. 7–12, 2012.

[33] M. Rabi, K. H. Johansson, and M. Johansson, “Optimal Stopping for Event-triggered sensing and actuation,” pp. 3607–3612, 2008.

[34] C.-Y. Kao and B. Lincoln, “Simple stability criteria for systems withtime-varying

delays," *Automatica*, vol. 40, no. 8, pp. 1429–1434, 2004.

[35] Z.-G. E. and E. M. Association, "Wireless Solutions in Automation Over," vol. 49, no. 0, 2011.

[36] Y. Tipsuwan and M.-Y. Chow, "Control methodologies in networked control systems," *Control Eng. Pract.*, vol. 11, no. 10, pp. 1099–1111, 2003.

[37] J. Nilsson and others, "Real-time control systems with delays," 1998.

[38] A. Willig, M. Kubisch, C. Hoene, and A. Wolisz, "Measurements of a wireless link in an industrial environment using an IEEE 802.11-compliant physical layer," *IEEE Trans. Ind. Electron.*, vol. 49, no. 6, pp. 1265–1282, 2002.

[39] K. J. Åström and B. Wittenmark, "Computer-Controlled Systems: Theory and Design," no. 3rd edition, p. 555, 2001.

[40] W. H. Kwon, Y. H. Kim, S. J. Lee, and K. N. Paek, "Event-Based modeling and control for the burnthrough point in sintering processes," *IEEE Trans. Control Syst. Technol.*, vol. 7, no. 1, pp. 31–41, 1999.

[41] T.-J. Tarn, N. Xit, and A. K. Bejczy~t, "Path-based Approach to Integrated Planning and Control for Robotic Systems*," vol. 96, no. 12, pp. 1675–1687, 1996.

[42] E. Hendricks, M. Jensen, a. Chevalier, and T. Vesterholm, "Problems in event based engine control," *Proc. 1994 Am. Control Conf. - ACC '94*, vol. 2, pp. 8–10, 1994.

[43] A. Pawlowski, J. L. Guzm, F. Rodr, M. Berenguel, D. Lenguajes, and D. Inform, "Event-Based Control and Wireless Sensor Network for Greenhouse Diurnal Temperature Control : A Simulated Case Study," *Emerg. Technol. Fact. Autom. 2008. ETFA 2008. IEEE Int. Conf.*, pp. 500–507, 2008.

[44] J. Weimer, J. Araújo, and K. H. Johansson, "Distributed event-triggered estimation in networked systems," *IFAC Proc. Vol.*, no. Adhs 12, pp. 178–185, 2012.

[45] H. Yu and P. J. Antsaklis, "Event-triggered output feedback control for networked control systems using passivity: Time-varying network induced delays," *Proc. IEEE Conf. Decis. Control*, pp. 205–210, 2011.

[46] R. Postoyan, A. Anta, W. P. M. H. Heemels, P. Tabuada, and D. Neˇ, "Periodic event-triggered control for nonlinear systems," vol. 58, no. 4, pp. 7397–7402, 2013.

[47] W. P. M. H. Heemels and M. C. F. Donkers, "Model-based periodic event-triggered control for linear systems," *Automatica*, vol. 49, no. 3, pp. 698–711, 2013.

[48] K. Årzén, "A simple event-based PID controller," *Proc. 14th IFAC World Congr.*, vol. 18, pp. 423–428, 1999.

[49] V. Vasyutynskyy and K. Kabitzsch, "Implementation of Pid Controller With Send-on-Delta Sampling," *Ukacc*, no. October, 2006.

[50] J. Sánchez, A. Visioli, and S. Dormido, “PID Control in the Third Millennium,” *chapter Event-based PID Control. Springer*, pp. 495–526, 2012.

[51] K. A. J. Xu, *Advanced Discrete- Time Control Designs and Applications*, Singapore. 2015.

[52] M. Miskowicz, “Send-On-Delta Concept: An Event-Based Data Reporting Strategy,” *Sensors*, vol. 6, no. 1, pp. 49–63, 2006.

[53] R.C. Dorf, “Adaptive Sampling Frequency for Sampled-Data Control Systems ;":,” *IEEE Trans. Automat. Contr.*, vol. AC-7, pp. 38–47, 1962.

[54] D. Laurent, J. Mitchell, and W. McDaniel, “Adaptive sampling technique,” *IEEE Trans. Automat. Contr.*, vol. 14, no. 2, pp. 200–201, 1969.

[55] P. G. Otanez, J. R. Moyne, and D. M. Tilbury, “Using deadbands to reduce communication in networked control systems,” *Proc. Am. Control Conf.*, vol. 4, pp. 3015–3020, 2002.

[56] M. Miskowicz, “Sampling of Signals in Energy Domain,” *Proc. 10th IEEE Int. Conf. Emerg. Technol. Fact. Autom.*, vol. 1, pp. 263–266, 2005.

[57] D. Lehmann and J. Lunze, “Extension and experimental evaluation of an event-based state-feedback approach,” *Control Eng. Pract.*, vol. 19, no. 2, pp. 101–112, 2011.

[58] M. Guarnes, S. Dormido, and A. Visioli, “Comparative Study of Event-Based Control Strategies: An Experimental Approach on a Simple Tank,” pp. 1973–1978, 2009.

[59] J. Sánchez, A. Visioli, and S. Dormido, “A two-degree-of-freedom PI controller based on events,” *J. Process Control*, vol. 21, no. 4, pp. 639–651, 2011.

[60] S. Durand, N. Marchand, S. Durand, N. Marchand, A. E. Pid, C. With, L. Computational, C. L. Fesquet, B. Torr, S. Durand, and N. Marchand, “To cite this version : An Event-Based PID Controller With Low Computational Cost,” 2010.

[61] W. Heemels, J. H. Sandee, and P. P. J. Van Den Bosch, “Analysis of event-driven controllers for linear systems,” *Int. J. Control*, vol. 81, no. 4, pp. 571–590, 2008.

[62] T. Henningsson and A. Cervin, “Comparison of LTI and event-based control for a moving cart with quantized position measurements,” *Proc. Eur. Control Conf*, pp. 3791–3796, 2009.

[63] M. Rabi and J. S. Baras, “Level-triggered control of a scalar linear system,” *2007 Mediterr. Conf. Control Autom. MED*, pp. 1–6, 2007.

[64] A. Cervin and K. J. Åström, “On limit cycles in event-based control systems,” *Proc. IEEE Conf. Decis. Control*, pp. 3190–3195, 2007.

[65] A. Cervin and T. Henningsson, “Scheduling of event-triggered controllers on a shared network,” *Proc. IEEE Conf. Decis. Control*, pp. 3601–3606, 2008.

[66] M. Rabi and K. H. Johansson, “Scheduling Packets packets for for Event-Triggered event-triggered control Scheduling Control,” *Proc. 10th Eur. Control*

Conf., pp. 3779–3784, 2009.

[67] J. Lunze and D. Lehmann, “A state-feedback approach to event-based control,” *Automatica*, vol. 46, no. 1, pp. 211–215, 2010.

[68] V. Vasyutynskyy and K. Kabitzsch, “First order observers in event-based PID controls,” *ETFA 2009 - 2009 IEEE Conf. Emerg. Technol. Fact. Autom.*, 2009.

[69] L. Desborough and R. Miller, “Increasing customer value of industrial control performance monitoring-Honeywell’s experience,” *AIChE Symp. Ser.*, no. Figure 1, pp. 169--189, 2002.

[70] B. Lennartson and B. Kristiansson, “Evaluation and tuning of robust PID controllers,” *IET Control theory Appl.*, vol. 3, no. 3, pp. 294–302, 2009.

[71] Yun Li, Kiam Heong Ang, and G. C. Y. Chong, “PID control system analysis and design,” *IEEE Control Syst. Mag.*, vol. 26, no. 1, pp. 32–41, 2006.

[72] P. Cominos, N. Munro, T. H. Lee, W. S. Ra, S. H. Jin, T. S. Yoon, and J. B. Park, “PID controllers: recent tuning methods and design to specification,” *IEE Proceedings-Control Theory Appl.*, vol. 149, no. 1, 2002.

[73] J. G. Ziegler and N. B. Nichols, “Optimum settings for automatic controllers,” *InTech*, vol. 42, no. 6, pp. 94–100, 1995.

[74] G. Sridharan, “Ball on Beam on Roller : A New Control Laboratory Device,” vol. 91, pp. 1318–1321, 2002.

[75] F. Ortiz, “Stability analysis of PD regulation for ball and beam system,” *Proc. 2005 IEEE Conf. Control Appl. 2005. CCA 2005.*, pp. 517–522, 2005.

[76] D. L. M. Lizarazo and J. A. T. Borja, “CONTROL DE UN BALL & BEAM CON MATLAB Y LEGO NXT,” *Visión Electrónica algo más que un estado sólido*, vol. 8, no. 2, pp. 39–48, 2015.

[77] R. Eberhart and J. Kennedy, “A new optimizer using particle swarm theory,” *Proc. Sixth Int. Symp. Micro Mach. Hum. Sci.*, pp. 39–43, 1995.

[78] J. M. R. Guitiérrez, “Arduino Ethernet Shield,” *Arduinio Comun.*, p. 3, 2010.

7 ANEXO A

7.1 Modelo Matemático de la planta Beam-Ball

Para obtener el modelo matemático que describe el comportamiento de la planta, es decir la función de transferencia de la planta se realizó experimentalmente de la siguiente manera, primero se implementó la planta con el controlador PID incluido en el cual se configuró los parámetros de una forma empírica experimental siguiendo criterios de estabilización de Zigler-Nichols, luego de que se logró estabilizar adecuadamente se tomó los datos, muestras de la entrada y de la salida del sistema. Cave recalcar que tal experimento para la toma de datos se lo hizo de manera local, es decir todos los elementos del sistema B-B, sensor-actuador, planta y controlador estaban físicamente unidos.

Los datos de salida del sistema se obtuvo de las mediciones que eran impresas por medio de arduino, mediante comunicación serial, a continuación estos datos se los copio a una hoja de Excel para realizar una tabla de datos tanto de entrada como de salida (archivo : funcion_pid_bloques.xls) luego se tomo esos datos en Matlab mediante el scrip : FunctionTrans.m

```
% PARA IDENTIFICAR LA FUNCION DE TRANSFERENCIA
% teniendo datos tomados de las mediciones reales.
% estos datos estan en un archivo de excel
% primero leemos el rango de datos que se desea de
% la funcion de excel
clear all; close;
datosuy=xlsread('funcion_pid_bloques.xlsx','Hoja1','B48:C135');
u=datosuy(:,2);
y=datosuy(:,1);
ident
```

Mediante la herramienta ident se realizó un ajuste de los datos para ver cuál era la función de transferencia que más se acoplaba a los datos tomados experimentalmente: siendo u los datos de entrada, y los datos de salida y a un tiempo de muestreo de T=0.01 [s], como se puede observar se realizó algunas aproximaciones siendo la tf1 la cual se estimó la Función de transferencia con 2 polos y 1 cero sin retardo en tiempo continuo

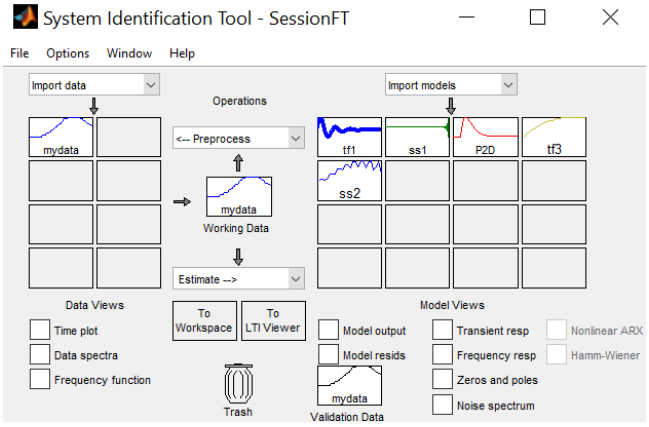


Figura 7.1 Herramienta iden para sacar FT

```
>> tf1
tf1 =
  From input "u1" to output "y1":
     1.068 s + 151.3
  ---------------------
  s^2 + 5.458 s + 163.9

Name: tf1
Continuous-time identified transfer function.
Parameterization:
   Number of poles: 2   Number of zeros: 1
   Number of free coefficients: 4
   Use "tfdata", "getpvec", "getcov" for parameters and their uncertainties.
Status:
Estimated using TFEST on time domain data "mydata".
Fit to estimation data: 76.63% (simulation focus)
FPE: 0.8398, MSE: 0.7475
```

Para corroborar se utilizó también el criterio de la función de transferencia del proceso

```
P2U =
Process model with transfer function:
                Kp
  G(s) = ---------------------
         1+2*Zeta*Tw*s+(Tw*s)^2

         Kp = 0.92319
         Tw = 0.3907
       Zeta = 0.21306
Name: P2U
Parameterization:
   'P2U'
```

Number of free coefficients: 3
Use "getpvec", "getcov" for parameters and their uncertainties.
Status:
Estimated using PROCEST on time domain data "mydata".
Fit to estimation data: 76.63% (prediction focus)
FPE: 0.823, MSE: 0.7476

Con estos parámetros se empezó a realizar la simulación

Para el modelamiento se identificó las respuestas del sistema en conjunto para lo cual se midió la señal de salida del sistema como es la distancia de la bola al eje respecto a una señal de referencia, estos datos fueron utilizados para la identificación del sistema utilizando el Toolbox de Matlab (IDENT), De los modelos identificados, se seleccionó la función de transferencia, que representa mejor el comportamiento del sistema, así como la función del PID tomada de datos experimentales con las contantes $K_p = 8\ ;\ K_i = 0.5\ y\ K_d = 2.8$, se sabe que la función del controlador es $PID = \left(K_p + sK_d + \frac{K_i}{s}\right)$, por lo tanto:

$$Gt = PID * FTp$$

$$Gt = \frac{6.049}{s^2 + 1.091s + 6.552}\ ; \quad PID = \frac{2.8s^2 + 8s + 0.5}{s}\ ;$$

$$FTp = \frac{6.049s}{\left((s^2 + 1.091s + 0.503) * (2.8s^2 + 8s + 0.5)\right)} \qquad (1)$$

Al realizar el siguiente script en matlab:

```
nump=[6.049 0];denp=[2.8 11.05 10.64 4.5695 0.2515]; % CON LA FUNCION IDENT
gpt=tf(nump,denp) % Función de transferencia del proceso
[p,z]=pzmap(gpt)
rlocus(gpt)          % Grafico de Polos y Ceros de la FT del proceso sin controlador
[A,B,C,D]=tf2ss(nump,denp)
error=0;
dt=0.01; Tmax=5;
t=0:dt:Tmax;
s=tf('s');
Planta=tf(nump,denp);
Kp=8; Ki=0.5;Kd=2.8;
Cpid= pid(Kp,Ki,Kd)
%Sistema lazo cerrado realimentación Unitaria
sys_c1=feedback(Cpid*Planta,1)
[p,z]=pzmap(sys_c1)
rlocus(sys_c1)
title(sprintf('LGR LazoCerrado  , Kp=%f, Ki=%f, Kd=%f',Kp,Ki,Kd));
step(sys_c1);
 m= step(sys_c1,t);
 % %Calculo del error
```

```
error=1-m;
IAE=trapz(t,abs(error))
```

Se obtuvo los siguientes resultados:

```
gpt =
                    6.049 s
  ---------------------------------------------------
  2.8 s^4 + 11.05 s^3 + 10.64 s^2 + 4.569 s + 0.2515

Continuous-time transfer function.
p =
  -2.7899 + 0.0000i
  -0.5463 + 0.4529i
  -0.5463 - 0.4529i
  -0.0639 + 0.0000i
z =
     0
Matrices de estado de la FT del proceso
A =
   -3.9464   -3.8000   -1.6320   -0.0898
    1.0000         0         0         0
         0    1.0000         0         0
         0         0    1.0000         0
B =
     1
     0
     0
     0
C =
         0         0    2.1604         0
D =
     0
```

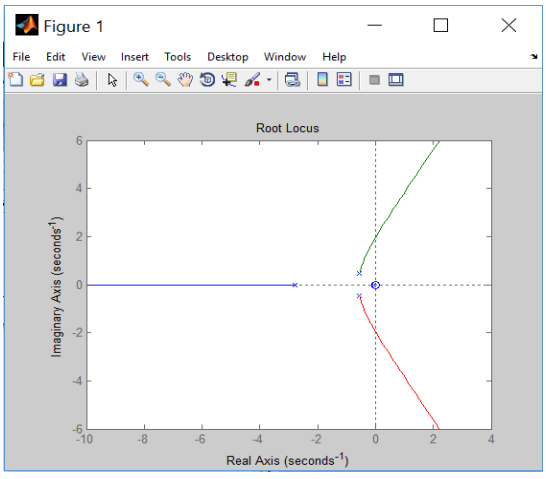


Figura 7.2 Gráfico de Polos y ceros de la FT sin Controlador

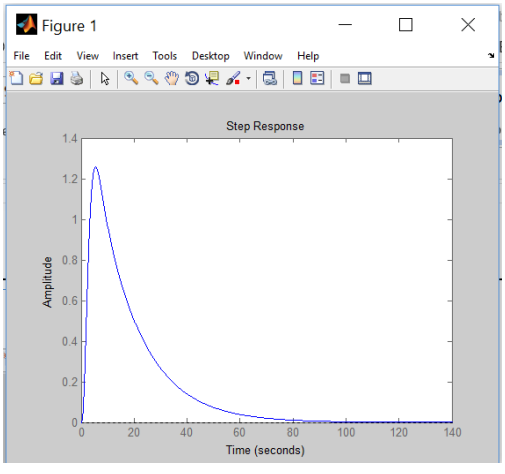


Figura 7.3 Repuesta paso de la planta sin Controlador

Sistema con controlador y realimentación unitaria

sys_c1 =

$$\frac{16.94\,s^3 + 48.39\,s^2 + 3.025\,s}{2.8\,s^5 + 11.05\,s^4 + 27.58\,s^3 + 52.96\,s^2 + 3.276\,s}$$

Continuous-time transfer function.

Polos y Ceros de la FT del proceso, con controlador y realimentación unitaria

p =

0.0000 + 0.0000i
-2.7917 + 0.0000i
-0.5454 + 2.5016i
-0.5454 - 2.5016i
-0.0639 + 0.0000i

z =

0
-2.7932
-0.0639

IAE =

1.1134

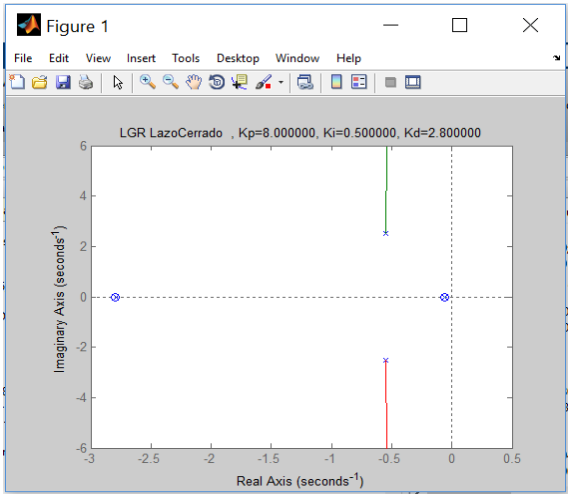


Figura 7.5 LGR de la planta con controlador PID continúo

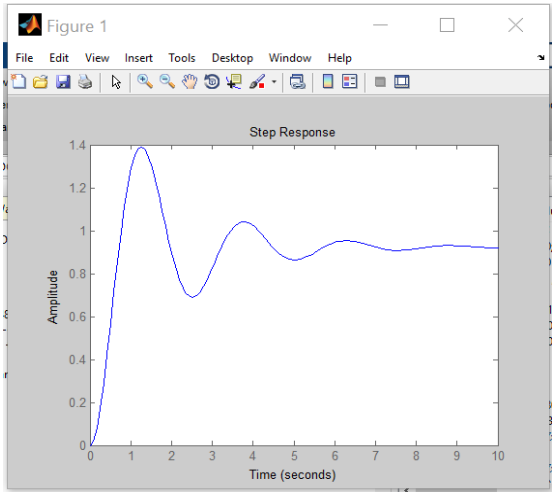


Figura 7.4 Respuesta a una señal paso del sistema con PID

7.2 Modelo discreto de la Planta Beam-Ball

También se realizó otro método en base a datos medidos y en función de ciertos parámetros necesarios en donde se obtuvo la función de transferencia del sistema motor y luego del sistema bola de la siguiente manera:

Modelo del motor

i) Se consideró el sistema motor mediante el siguiente modelo esquemático

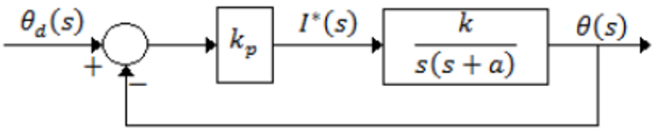


Figura 7.6 Control proporcional de posición para la riel

Donde la función de transferencia correspondiente es:

$$\frac{\theta(s)}{\theta_d(s)} = \frac{k_p k}{s^2 + as + k_p k} = \frac{w_n^2}{s^2 + 2\zeta w_n s + w_n^2} \qquad (A1)$$

Tal que:

$$k = \frac{w_n^2}{k_p}\ ,\ \ a = 2\zeta w_n \quad y \quad W_n = \sqrt{k_p k}$$

Donde Wn se conoce como la frecuencia natural y ζ es un factor de amortiguamiento.

$$\zeta = \sqrt{\frac{ln^2\left(\frac{M_p(\%)}{100}\right)}{ln^2\left(\frac{M_p(\%)}{100}\right) + \pi^2}} \qquad (A2)$$

$$w_d = \frac{1}{t_r}\left[\pi - tan^{-1}\left(\frac{\sqrt{1-\zeta^2}}{\zeta}\right)\right] \qquad (A3)$$

$$w_n = \frac{w_d}{\sqrt{1-\zeta^2}} \qquad (A4)$$

Con datos medidos se obtuvo:

Mp	0,09
tr	1,17
kp	5,9
k	0,972862709
a	2,914900359

Discretizando la función y considerando:

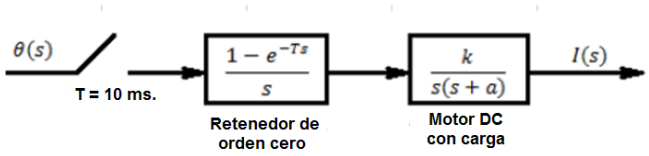


Figura 7.7 Diagrama de la Función de transferencia de un motor DC

Del modelo anterior se tiene:

$$G1(s) = \frac{\theta(s)}{I(s)} = \frac{1 - e^{-Ts}}{s} \frac{k}{s(s+a)} \qquad (A5)$$

A este modelo se le aplica la transformada z y se obtiene:

$$G_1(z) = \frac{\theta(z)}{I(z)} = Z\left[(1 - e^{-Ts}) \frac{k}{s^2(s+a)}\right] \qquad (A6)$$

$$G_1(z) = \frac{\theta(z)}{I(z)} = Z\left[(1 - e^{-Ts})\right] Z\left[\frac{k}{s^2(s+a)}\right] \qquad (A7)$$

Donde:

$$A = -\frac{k}{a^2} \; ; \quad B = \frac{k}{a} \quad y \quad C = \frac{k}{a^2}$$

Si:

$$Z[e^{Ts}] = z \;\; , \quad Z\left[\frac{1}{s}\right] = \frac{1}{1 - z^{-1}} \;\; , \quad Z\left[\frac{1}{s^2}\right] = \frac{Tz^{-1}}{(1 - z^{-1})^2} \;\; , \; Z\left[\frac{1}{(s+a)}\right] = \frac{1}{1 - e^{-aT}z^{-1}}$$

$$G_1(z) = (1 - z^{-1}) \left(\frac{A}{1 - z^{-1}} + \frac{BTz^{-1}}{(1 - z^{-1})^2} + \frac{C}{1 - e^{-aT}z^{-1}}\right) \qquad (A8)$$

Reemplazando los datos obtenidos anteriormente se tiene los parámetros k=0.972 , a = 2.914 del sistema Beam-Ball y tiempo de muestreo T= 10 ms, tenemos que:

A = -0.1144 ; B= 0.3337 ; C= 0.1144

Por lo tanto:

$$G_1(z) = \frac{\theta(z)}{I(z)} = \frac{0.0004253\, z^{-1} + 0.0041308\, z^{-2}}{1 - 1.916267 z^{-1} + 0.916267 z^{-2}} \qquad (A9)$$

La ecuación (A9) es la función de transferencia en transformada z del modelo del motor de DC, a continuación se presenta la función discreta del modelo anterior para su programación directa.

$$(1 - 1.916267z^{-1} + 0.916267z^{-2})\theta(z) = (0.0004253\, z^{-1} + 0.0041308\, z^{-2})I(z) \qquad (A10)$$

Aplicando la transformada z:

$$x(n+k) = z^{-k}X(z) \qquad (A11)$$

Se tiene:

$$\theta(n) = 1.916267\,\theta(n-1) - 0.916267\,\theta(n-2) + 0.0004253\,I(n-1) + 0.0041308\,I(n-2) \qquad A(12)$$

La ecuación anterior es la función que representa al modelo dinámico del motor DC y es el que se programa en cualquier sistema digital

Modelo de la bola

Modelo dinámico de la bola consiste en una bola que gira sobre un riel donde:

$$\frac{X(s)}{\theta(s)} = \frac{\rho}{s^2} \qquad (A13)$$

Para su conversión en la transformada z se diseña el siguiente modelo:

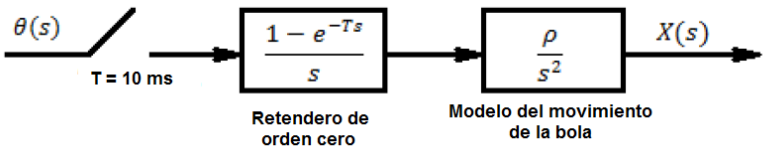


Figura 7.8 Diagrama del modelo de la bola con retenedor de orden cero

Del diagrama anterior se tiene:

$$G_2(s) = \frac{X(s)}{\theta(s)} = \frac{1 - e^{-Ts}}{s}\frac{\rho}{e^2} \qquad (A14)$$

Aplicando la transformada z a la ecuación anterior:

$$G_2(z) = \frac{X(z)}{\theta(z)} = Z\left[(1 - e^{-Ts})\left(\frac{\rho}{e^3}\right)\right] \qquad (A15)$$

$$G_2(z) = Z[(1 - e^{-Ts})]\ Z\left[\left(A\frac{2}{e^3}\right)\right] \qquad (A16)$$

Donde $A = \rho/2$; si:

$$Z[e^{Ts}] = z \quad y \quad Z\left[\frac{2}{s^3}\right] = \frac{T^2 z^{-1}(1 + z^{-1})}{(1 - z^{-1})^3} \qquad (A17)$$

Entonces:

$$G_2(z) = \frac{AT^2 z^{-1} + AT^2 z^{-2}}{1 - 2z^{-1} + z^{-2}} \qquad (A18)$$

Sustituyendo el parámetro Rho = 3 del BB entontes A=1.5 y si el tiempo de muestreo de T= 10ms, por lo tanto:

$$G_2(s) = \frac{3}{s^2} \qquad (A19)$$

$$G_2(z) = \frac{X(z)}{\theta(z)} = \frac{0.00015z^{-1} + 0.00015z^{-2}}{1 - 2z^{-1} + z^{-2}} \qquad (A20)$$

Para la programación directa utilizando la propiedad de la transformada z a la ecuación (A20), se tiene:

$$x(n) = 2\,x(n-1) - x(n-2) + 0.00015\,\theta(n-1) + 0.00015\,\theta(n-2) \qquad (A21)$$

La ecuación anterior es la función que representa al modelo dinámico del movimiento de la bola y es el que se programa en cualquier sistema digital

Realizando la respectiva simulación en simulink de las funciones de transferencia se tiene:

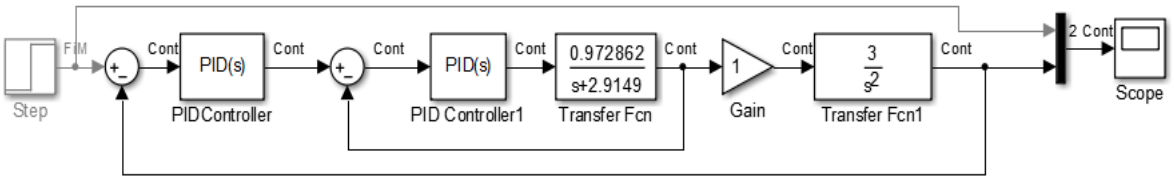


Figura 7.9 Diagrama de bloques en simulink del sistema Beam-Ball

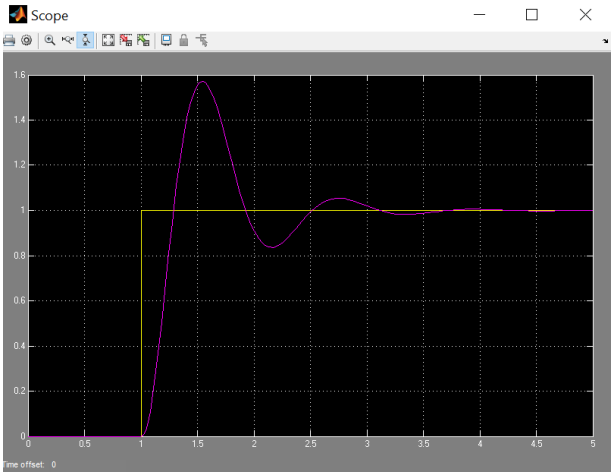


Figura 7.10 Respuesta del sistema BB ante una entrada paso en Simulik

8 ANEXO B

8.1 Código en arduino del Servidor-Controlador

A continuación se presentan los códigos de arduino implementados

8.1.1 Código del PID Asíncrono

```
///////// Inclusión de Librerías ///////////
#include <LiquidCrystal.h>
#include <SPI.h>
#include <Ethernet.h>
LiquidCrystal lcd(7, 8, 9, 10, 11, 12);
byte mac[] = {0xDE, 0xAD, 0xBE, 0xEF, 0xFE, 0xED};
IPAddress ip(192, 168, 0, 155);
EthernetServer server(80);
String readString,distancia_s;
float escala_1;
////////// Definición de variables /////////
int abajo,arriba,set,back;
int t_b;// tiempo de botones
int menu_v, aceptar;// variable menu
float dt,aumento_f,distancia_falsa,distancia_grafico;
int contador_f;
int linha = 0;
int LABEL = 1;
int valor = 0;
int inicio_d,inicio_c;
float distancia_pid, sp=14.0,
kp=2.70,ki=0.01,kd=0.79,error,proporcional,integral,derivativo,error_anterior,pid, salida ;
float compensador;
///////// Subrrutina de incialización ///////////
void setup()
{
  lcd.begin(16, 2);          // Fijamos el numero de caracteres y fil
  pinMode(2,INPUT);
  pinMode(3,INPUT);
  pinMode(4,INPUT);
  pinMode(5,INPUT);
  lcd.clear();
  delay(10);
  lcd.setCursor(2, 0);
  lcd.print("PID.Eventos");  // Aquí va el mensaje
  lcd.setCursor(5, 1);    // Ponte en la line 1, posición 5
  lcd.print("Menu_I") ;
  salida=100;
  Serial.begin(9600);
  Ethernet.begin(mac, ip);
  server.begin();
  Serial.println("CLEARDATA");           // Reset da comunicación serial
```

```
  Serial.println("LABEL,Hora,Distancia,dt");
  dt=0;
}
///////// Inicia el Programa //////
void loop() {
abajo=digitalRead(2);
arriba=digitalRead(3);
set=digitalRead(4);
back=digitalRead(5);
if (t_b%10==0 )  {
  botones_entrada();
  }
///////// Condiciones de Pulsadores para Menú //////
if (abajo== HIGH || arriba == HIGH || set== HIGH || back==HIGH)
{
  if (menu_v == 0 && aceptar==1)
    {
    inicio();
    }
  if (menu_v == 1 && aceptar==1)
    {
    kp_lcd();
    }
  if (menu_v == 2 && aceptar==1)
    {
    ki_lcd();
    }
  if (menu_v == 3 && aceptar==1)
    {
    kd_lcd();
    }
  if (menu_v == 0 && aceptar==0)
    {
    menu_0();
    }
  if (menu_v == 1 && aceptar==0)
    {
    menu_1();
    }
  if (menu_v == 2 && aceptar==0)
    {
    menu_2();
    }
  if (menu_v == 3 && aceptar==0)
  {
  menu_3();
  }
}
//////////////////////////////////////////////////////
if (menu_v==0 && aceptar==1 )
```

```
{
  EthernetClient client = server.available();
  if (client)
  {
    boolean currentLineIsBlank = true;
    while (client.connected())
    {
      if (client.available())
      {
      char c = client.read();
      if (readString.length() < 16)
        {
        readString += c;
        }
      if (c == '\n' && currentLineIsBlank)
      {
        aumento_f=0;
        contador_f=0;
        if (readString.indexOf("dist:")>0)
        {
          inicio_d=readString.indexOf(":");
          for (inicio_c=0; inicio_c <=5 ; inicio_c++)
            {
            distancia_s += readString[inicio_d+inicio_c+1];
            }
        }
        distancia_pid = distancia_s.toFloat();
        pid_eventos();
        client.println("HTTP/1.1 200 OK");
        client.println("Content-Type: text/html");
        client.println("Connection: close");
        client.println();
        client.println(salida);
        break;
      }
      if (c == '\n')
        {
        currentLineIsBlank = true;
        }
      else if (c != '\r')
        {
        currentLineIsBlank = false;
        }
      }
    }
    client.stop();
    readString="";
    dt=0;
    distancia_s="";
  }
```

```
  datos_falsos();
}
delay(1);
dt++;
contador_f++;
t_b++;
if (dt>=500)
  {
  dt=500;
  }
if (t_b>=255)
  {
  t_b=0;
  }
}
////////////////////////////////////////////////////////
void pid_eventos(void)
{
error = sp - distancia_falsa;
proporcional=error;
integral = integral + ((error+error_anterior)/2.0)*(dt/1000.0);
derivativo = (error - error_anterior)/(dt/1000.0);
pid=(kp*proporcional)+(ki*integral) +(kd*derivativo);
error_anterior=error;
if(pid <= -100 )
  {
  pid=-100;
  }
if(pid >= 100 )
  {
  pid=100;
  }
salida=map(pid,-100,100,150,45);
if(salida <= 45 )
  {
  salida=45;
  }
if(salida >= 140 )
{
salida=140;
}
}
////////////////////////////////////////////
void datos_falsos(void)
{
  if (contador_f%30==0){
    if (aumento_f==0) {
      distancia_grafico=distancia_pid  ;
      }
    else {
```

```
        distancia_grafico=0  ;
       }
     if (salida>= 105 && salida <= 140) {
     distancia_falsa= 0.25*aumento_f+distancia_pid;
       if (distancia_falsa >22 ) {
         distancia_falsa= 22;
       }
     }
     if (salida>= 45 && salida <= 90) {
     distancia_falsa= (-0.25*aumento_f)+distancia_pid;
       if (distancia_falsa < 0) {
         distancia_falsa= distancia_pid;
       }
     }
     if (salida>90  && salida < 105) {
       distancia_falsa= distancia_pid;
       }
     aumento_f++;
   }
   if (aumento_f >=255){
   aumento_f=0;
   contador_f=0,
   ki=0;
   }
}
////////////////////////////////////////
void inicio(void)
{
  lcd.clear();
  delay(1);
  lcd.setCursor(4, 0);
  lcd.print("p:");  // Aqi va el mensaje
  lcd.setCursor(7, 0);
  lcd.print(kp);
  lcd.setCursor(0, 1);    // Ponte en la line 1, posición 0
  lcd.print("i:") ;
  lcd.setCursor(3, 1);
  lcd.print(ki);
  lcd.setCursor(8, 1);    // Ponte en la line 1, posición 8
  lcd.print("d:") ;
  lcd.setCursor(11, 1);
  lcd.print(kd);
}
////////////////////////////////////////
void menu_0(void)
{
  lcd.clear();
  delay(1);
  lcd.setCursor(2, 0);
  lcd.print("PID.Eventos");  // Aquí va el mensaje
```

```
  lcd.setCursor(6, 1);    // Ponte en la line 1, posición 6
  lcd.print("Menu") ;
}
////////////////////////////////////
void kp_lcd(void)
{
  lcd.clear();
  delay(1);
  lcd.setCursor(0, 0);
  lcd.print("kp:");  // Aqi va el mensaje
  lcd.setCursor(0, 1);    // Ponte en la line 1, posición 0
  lcd.print(kp) ;
}
////////////////////////////////////
void ki_lcd(void)
{
  lcd.clear();
  delay(1);
  lcd.setCursor(0, 0);
  lcd.print("ki:");  // Aqi va el mensaje
  lcd.setCursor(0, 1);    // Ponte en la line 1, posición 0
  lcd.print(ki) ;
}
////////////////////////////////////
void kd_lcd(void)
{
  lcd.clear();
  delay(1);
  lcd.setCursor(0, 0);
  lcd.print("kd:");  // Aquí va el mensaje
  lcd.setCursor(0, 1);    // Ponte en la line 1, posición 0
  lcd.print(kd) ;
}
////////////////////////////////////
void menu_1(void)
{
  lcd.clear();
  delay(1);
  lcd.setCursor(1, 0);
  lcd.print("PID.Variables");  // Aquí va el mensaje
  lcd.setCursor(5, 1);    // Ponte en la line 1, posición 5
  lcd.print("kp:") ;
}
////////////////////////////////////
void menu_2(void)
{
  lcd.clear();
  delay(1);
  lcd.setCursor(1, 0);
  lcd.print("PID.Variables");  // Aquí va el mensaje
```

```
  lcd.setCursor(5, 1);    // Ponte en la line 1, posición 5
  lcd.print("ki:") ;
}
////////////////////////////////////
void menu_3(void)
{
  lcd.clear();
  delay(1);
  lcd.setCursor(1, 0);
  lcd.print("PID.Variables");  // Aquí va el mensaje
  lcd.setCursor(5, 1);    // Ponte en la line 1, posición 5
  lcd.print("kd:") ;
}
////////////////////////////////////
void botones_entrada(void)
{
if (arriba== HIGH && aceptar==0)  {
  menu_v++;
  }
else {
  menu_v=menu_v;
  }
if (abajo== HIGH && aceptar==0  ) {
  menu_v--;
  }
else {
  menu_v=menu_v;
  }
if (menu_v >= 3) {
  menu_v=3;
  }
if (menu_v <= 0){
  menu_v=0;
  }
if (set==HIGH  ){
  aceptar++;
  }
if (aceptar>=1 ){
  aceptar=1;
  }
if (back==HIGH ){
  aceptar--;
  }
if (aceptar<=0 ){
  aceptar=0;
  }
////////////////////////////////
if (aceptar == 1 && menu_v==1)
{
  if (arriba == HIGH){
```

```
    kp=kp+0.01;
    }
  if (abajo == HIGH){
    kp=kp-0.01;
    }
}
if (kp>=99){
  kp=99.0;
  }
if (kp<=0){
  kp=0.0;
  }
////////////////////////////////////
if (aceptar == 1 && menu_v==2)
{
  if (arriba == HIGH){
    ki=ki+0.01;
    }
  if (abajo == HIGH){
    ki=ki-0.01;
    }
}
if (ki>=99){
  ki=99.0;
  }
if (ki<=0){
  ki=0.0;
  }
////////////////////////////////////
if (aceptar == 1 && menu_v==3)
{
  if (arriba == HIGH){
    kd=kd+0.01;
    }
  if (abajo == HIGH){
    kd=kd-0.01;
    }
}
if (kd>=99){
  kd=99.0;
  }
if (kd<=0){
  kd=0.0;
  }
}
```

8.1.2 Código del PID Semi-síncrono

```
#include <LiquidCrystal.h>
#include <SPI.h>
```

```
#include <Ethernet.h>
LiquidCrystal lcd(7, 8, 9, 10, 11, 12);
byte mac[] = {0xDE, 0xAD, 0xBE, 0xEF, 0xFE, 0xED};
IPAddress ip(192, 168, 0, 155);
EthernetServer server(80);
String readString,distancia_s;
float escala_1;
////////Definición de variables /////////
int seguro_dat;
int abajo,arriba,set,back;
int t_b;// tiempo de botones
int menu_v, aceptar;// variable menu
float dt,aumento_f,distancia_falsa,distancia_grafico;
int contador_f;
 int linha = 0;
int LABEL = 1;
int valor = 0;
int inicio_d,inicio_c;
float distancia_pid, sp=14.0,
kp=2.5,ki=0.09,kd=0.30,error,proporcional,integral,derivativo,error_anterior,pid, salida ;
float compensador;
float t_semisincrono;
///// subrrutina de incialización /////
void setup()
{
  lcd.begin(16, 2);          // Fijamos el número de caracteres y filas
  pinMode(2,INPUT);
  pinMode(3,INPUT);
  pinMode(4,INPUT);
  pinMode(5,INPUT);
  lcd.clear();
  delay(10);
  lcd.setCursor(2, 0);
  lcd.print("PID.Eventos");  // Aquí va el mensaje
  lcd.setCursor(5, 1);    // Ponte en la line 1, posición 5
  lcd.print("Menu_I") ;
  salida=100;
  Serial.begin(9600);
  Ethernet.begin(mac, ip);
  server.begin();
  Serial.println("CLEARDATA");           // Resetea la comunicación serial
  Serial.println("LABEL,Hora,Distancia_falsa,dt");
  dt=0;
}
//////////////////////
void loop()
{
abajo=digitalRead(2);
arriba=digitalRead(3);
set=digitalRead(4);
```

```
back=digitalRead(5);
if (t_b%10==0 ){
  botones_entrada();
  }
////////////////////////////////////////////////
if (abajo== HIGH || arriba == HIGH || set== HIGH || back==HIGH){
  if (menu_v == 0 && aceptar==1){
      inicio();
      }
  if (menu_v == 1 && aceptar==1){
      kp_lcd();
      }
  if (menu_v == 2 && aceptar==1){
      ki_lcd();
      }
  if (menu_v == 3 && aceptar==1){
      kd_lcd();
      }
  if (menu_v == 0 && aceptar==0){
      menu_0();
      }
  if (menu_v == 1 && aceptar==0){
      menu_1();
      }
  if (menu_v == 2 && aceptar==0){
      menu_2();
      }
  if (menu_v == 3 && aceptar==0){
      menu_3();
      }
}
////////////////////////////////////////////////////
if (menu_v==0 && aceptar==1 )   {
  EthernetClient client = server.available();
  if (client) {
    boolean currentLineIsBlank = true;
    while (client.connected()) {
    if (client.available()) {
    char c = client.read();
    if (readString.length() < 16) {
      readString += c;
      }
    if (c == '\n' && currentLineIsBlank) {
    aumento_f=0;
    contador_f=0;
    seguro_dat=0;
////////////////////////////////////////////
    if (readString.indexOf("dist:")>0){
      inicio_d=readString.indexOf(":");
      for (inicio_c=0; inicio_c <=5 ; inicio_c++){
```

```
        distancia_s += readString[inicio_d+inicio_c+1];
        }
    }
    distancia_pid = distancia_s.toFloat();
////////////////////////////////////////////
    client.println("HTTP/1.1 200 OK");
    client.println("Content-Type: text/html");
    client.println("Connection: close");
    client.println();
    client.println(salida);
    break;
    }
    if (c == '\n') {
    currentLineIsBlank = true;
    }
    else if (c != '\r') {
    currentLineIsBlank = false;
    }
   }
 }
 client.stop();
 readString="";
 dt=0;
 distancia_s="";
 }
}
  delay(1);
  dt++;
  contador_f++;
  t_semisincrono++;
  t_b++;
  datos_falsos();
  if (t_b>=255){
    t_b=0;
    }
  if (t_semisincrono>=100){
    t_semisincrono=0;
    }
  if (dt>=600 && seguro_dat==0){
    dt=600;
    contador_f=0;
    t_semisincrono=0;
    seguro_dat=1;
    pid=0;
    }
}
////////// Módulo PID ////
void pid_eventos(void){
error = sp - distancia_falsa;
proporcional=error;
```

```
integral = integral + ((error+error_anterior)/2.0)*(t_semisincrono/1000.0);
derivativo = (error - error_anterior)/(t_semisincrono/1000.0);
pid=(kp*proporcional)+(ki*integral) +(kd*derivativo);
error_anterior=error;
t_semisincrono=0;
if(pid <= -100 ){
  pid=-100;
  }
if(pid >= 100 ){
  pid=100;
  }
salida=map(pid,-100,100,150,45);
if(salida <= 45 ){
  salida=45;
  }
if(salida >= 140 ){
  salida=140;
  }
}
//////// SUBRRUTINA DE DATOS
void datos_falsos(void){
if ((contador_f%30==0 && seguro_dat==0) || (dt==1 && seguro_dat==0)){
  if (aumento_f==0){
  distancia_grafico=distancia_pid  ;
  }else{
    distancia_grafico=0  ;
    }
  if (salida>= 105 && salida <= 140){
    distancia_falsa= (-0.25*aumento_f)+distancia_pid;
    if (distancia_falsa >22 ){
      distancia_falsa= 22;
      }
    if (distancia_falsa < 0){
      distancia_falsa= distancia_pid;
      }
  }
  if (salida>= 45 && salida <= 90){
    distancia_falsa= (0.25*aumento_f)+distancia_pid;
    if (distancia_falsa < 0){
      distancia_falsa= distancia_pid;
      }
    if (distancia_falsa >22 ){
      distancia_falsa= 22;
      }
  }
  if (salida>90  && salida < 105){
    distancia_falsa= distancia_pid;
    }
  pid_eventos();
  aumento_f++;
```

```
}
if (aumento_f >=10){
  aumento_f=0;
  }
}
//////////////////////
void inicio(void){
  lcd.clear();
  delay(1);
  lcd.setCursor(4, 0);
  lcd.print("p:");  // Aquí va el mensaje
  lcd.setCursor(7, 0);
  lcd.print(kp);
  lcd.setCursor(0, 1);    // Ponte en la line 1, posición 0
  lcd.print("i:") ;
  lcd.setCursor(3, 1);
  lcd.print(ki);
  lcd.setCursor(8, 1);    // Ponte en la line 1, posición 8
  lcd.print("d:") ;
  lcd.setCursor(11, 1);
  lcd.print(kd);
}
//////////////////////////
void menu_0(void){
  lcd.clear();
  delay(1);
  lcd.setCursor(2, 0);
  lcd.print("PID.Eventos");  // Aquí va el mensaje
  lcd.setCursor(6, 1);    // Ponte en la line 1, posición 6
  lcd.print("Menu") ;
}
//////////////////////////
void kp_lcd(void){
  lcd.clear();
  delay(1);
  lcd.setCursor(0, 0);
  lcd.print("kp:");  // Aquí va el mensaje
  lcd.setCursor(0, 1);    // Ponte en la line 1, posición 0
  lcd.print(kp) ;
}
//////////////////////
void ki_lcd(void){
  lcd.clear();
  delay(1);
  lcd.setCursor(0, 0);
  lcd.print("ki:");  // Aquí va el mensaje
  lcd.setCursor(0, 1);    // Ponte en la line 1, posición 0
  lcd.print(ki) ;
}
//////////////////////
```

```
void kd_lcd(void){
  lcd.clear();
  delay(1);
  lcd.setCursor(0, 0);
  lcd.print("kd:");  // Aqi va el mensaje
  lcd.setCursor(0, 1);    // Ponte en la line 1, posicion 6
  lcd.print(kd) ;
}
//////////////////////
void menu_1(void){
  lcd.clear();
  delay(1);
  lcd.setCursor(1, 0);
  lcd.print("PID.Variables");  // Aquí va el mensaje
  lcd.setCursor(5, 1);    // Ponte en la line 1, posición 5
  lcd.print("kp:") ;
}
//////////////////////
void menu_2(void){
  lcd.clear();
  delay(1);
  lcd.setCursor(1, 0);
  lcd.print("PID.Variables");  // Aquí va el mensaje
  lcd.setCursor(5, 1);    // Ponte en la line 1, posición 5
  lcd.print("ki:") ;
}
//////////////////////
void menu_3(void){
lcd.clear();
delay(1);
lcd.setCursor(1, 0);
lcd.print("PID.Variables");  // Aqi va el mensaje
lcd.setCursor(5, 1);    // Ponte en la line 1, posición 5
lcd.print("kd:") ;
}
//////////////////////
void botones_entrada(void){
  if (arriba== HIGH && aceptar==0){
    menu_v++;
    }else {
    menu_v=menu_v;
    }
  if (abajo== HIGH && aceptar==0  ){
    menu_v--;
    }else{
    menu_v=menu_v;
    }
  if (menu_v >= 3){
    menu_v=3;
    }
```

```
if (menu_v <= 0){
  menu_v=0;
  }
if (set==HIGH  ){
  aceptar++;
  }
if (aceptar>=1 ){
  aceptar=1;
  }
if (back==HIGH ){
 aceptar--;
}
if (aceptar<=0 ){
  aceptar=0;
}
if (aceptar == 1 && menu_v==1){
  if (arriba == HIGH){
    kp=kp+0.01;
  }
  if (abajo == HIGH){
      kp=kp-0.01;
  }
}
if (kp>=99){
  kp=99.0;
  }
if (kp<=0){
  kp=0.0;
  }
if (aceptar == 1 && menu_v==2){
  if (arriba == HIGH){
    ki=ki+0.01;
  }
  if (abajo == HIGH){
    ki=ki-0.01;
  }
}
if (ki>=99){
  ki=99.0;
}
if (ki<=0){
  ki=0.0;
}
if (aceptar == 1 && menu_v==3){
  if (arriba == HIGH){
    kd=kd+0.01;
  }
if (abajo == HIGH){
    kd=kd-0.01;
  }
```

```
  }
  if (kd>=99){
    kd=99.0;
  }
  if (kd<=0){
    kd=0.0;
  }
}
```

8.1.3 Código del PID Síncrono

```
#include <LiquidCrystal.h>
#include <SPI.h>
#include <Ethernet.h>
LiquidCrystal lcd(7, 8, 9, 10, 11, 12);
byte mac[] = {0xDE, 0xAD, 0xBE, 0xEF, 0xFE, 0xED};
IPAddress ip(192, 168, 0, 155);
EthernetServer server(80);
String readString,distancia_s;
float escala_1;
//////////////////////////
int seguro_dat;
int abajo,arriba,set,back;
int t_b;// tiempo de botones
int menu_v, aceptar;// variable menu
float dt,aumento_f,distancia_falsa,distancia_grafico;
int contador_f;
int linha = 0;
int LABEL = 1;
int valor = 0;
int inicio_d,inicio_c;
float distancia_pid, sp=15.0,
kp=2.5,ki=0.09,kd=0.32,error,proporcional,integral,derivativo,error_anterior,pid, salida ;
float compensador;
////////////// CONFIGURACION INICIAL ///
void setup() {
lcd.begin(16, 2);         // Fijamos el número de caracteres y filas
    pinMode(2,INPUT);
    pinMode(3,INPUT);
    pinMode(4,INPUT);
    pinMode(5,INPUT);
lcd.clear();
delay(10);
lcd.setCursor(2, 0);
lcd.print("PID.Eventos");  // Aquí va el mensaje
lcd.setCursor(5, 1);    // Ponte en la line 1, posición 5
lcd.print("Menu_I") ;
salida=100;
Serial.begin(9600);
Ethernet.begin(mac, ip);
```

```
server.begin();
Serial.println("CLEARDATA");
Serial.println("LABEL,Hora,d,a");
dt=0;
}
//////////////////
void loop() {
abajo=digitalRead(2);
arriba=digitalRead(3);
set=digitalRead(4);
back=digitalRead(5);
if (t_b%10==0 ){
  botones_entrada();
  }
////////////////////////////////////////////////
if (abajo== HIGH || arriba == HIGH || set== HIGH || back==HIGH){
  if (menu_v == 0 && aceptar==1){
    inicio();
    }
  if (menu_v == 1 && aceptar==1){
    kp_lcd();
    }
  if (menu_v == 2 && aceptar==1){
    ki_lcd();
    }
  if (menu_v == 3 && aceptar==1){
    kd_lcd();
    }
  if (menu_v == 0 && aceptar==0){
    menu_0();
    }
  if (menu_v == 1 && aceptar==0){
    menu_1();
    }
  if (menu_v == 2 && aceptar==0){
    menu_2();
    }
  if (menu_v == 3 && aceptar==0){
    menu_3();
    }
}
////////////////////////////////////////////////////
if (menu_v==0 && aceptar==1 )   {
EthernetClient client = server.available();
if (client) {
boolean currentLineIsBlank = true;
while (client.connected()) {
  if (client.available()) {
    char c = client.read();
    if (readString.length() < 16) {
```

```
      readString += c;
      }
    if (c == '\n' && currentLineIsBlank) {
      aumento_f=0;
      contador_f=0;
      seguro_dat=0;
      if (readString.indexOf("dist:")>0){
        inicio_d=readString.indexOf(":");
          for (inicio_c=0; inicio_c <=5 ; inicio_c++){
          distancia_s += readString[inicio_d+inicio_c+1];
          }
        }
      distancia_pid = distancia_s.toFloat();

      client.println("HTTP/1.1 200 OK");
      client.println("Content-Type: text/html");
      client.println("Connection: close");
      client.println();
      client.println(salida);
      break;
    }
    if (c == '\n') {
      currentLineIsBlank = true;
      }
    else if (c != '\r') {
      currentLineIsBlank = false;
      }
    }
}
client.stop();
readString="";
dt=0;
distancia_s="";
}
}
delay(1);
dt++;
contador_f++;
t_b++;
datos_falsos();
if (t_b>=255){
  t_b=0;
  }
if (dt>=300 && seguro_dat==0){
  dt=300;
  contador_f=0;
  seguro_dat=1;
  pid=0;
}
}
```

```
//////////////////////
void pid_eventos(void){
error = sp - distancia_falsa;
proporcional=error;
integral = integral + ((error+error_anterior)/2.0)*(0.03);
derivativo = (error - error_anterior)/(0.03);
if (distancia_pid > 13.0  && distancia_pid < 17.0){
pid=(kp*proporcional)+(ki*integral) +(kd*derivativo);
}else
{pid=(kp*proporcional)+(kd*derivativo);
}
error_anterior=error;
if(pid <= -100 ){
  pid=-100;
  }
if(pid >= 100 ){
  pid=100;
  }
salida=map(pid,-100,100,150,45);
if(salida <= 45 ){
  salida=45;
  }
if(salida >= 140 ){
  salida=140;
  }
}
//////////////////////////////
void datos_falsos(void){
  if (contador_f%30==0 && seguro_dat==0){
    if (aumento_f==0){
      distancia_grafico=distancia_pid  ;
      }else{
      distancia_grafico=0  ;
      }
    if (salida>= 105 && salida <= 140){
      distancia_falsa= (-0.25*aumento_f)+distancia_pid;
      if (distancia_falsa >22 ){
        distancia_falsa= 22;
        }
      if (distancia_falsa < 0){
        distancia_falsa= distancia_pid;
        }
      }
    if (salida>= 45 && salida <= 90){
      distancia_falsa= 0.25*aumento_f+distancia_pid;
      if (distancia_falsa < 0){
        distancia_falsa= distancia_pid;
        }
      if (distancia_falsa >22 ){
        distancia_falsa= 22;
```

```
        }
       }
     if (salida>90  && salida < 105){
       distancia_falsa= distancia_pid;
       }
     pid_eventos();
     aumento_f++;
   }
   if (aumento_f >=5){
     aumento_f=0;
   }
}
//////////////////////////
void inicio(void){
  lcd.clear();
  delay(1);
  lcd.setCursor(4, 0);
  lcd.print("p:");  // Aquí va el mensaje
  lcd.setCursor(7, 0);
  lcd.print(kp);
  lcd.setCursor(0, 1);    // Ponte en la line 1, posición 0
  lcd.print("i:") ;
  lcd.setCursor(3, 1);
  lcd.print(ki);
  lcd.setCursor(8, 1);    // Ponte en la line 1, posición 8
  lcd.print("d:") ;
  lcd.setCursor(11, 1);
  lcd.print(kd);
}
//////////////////////////
  void menu_0(void){
  lcd.clear();
  delay(1);
  lcd.setCursor(2, 0);
  lcd.print("PID.Eventos");  // Aqi va el mensaje
  lcd.setCursor(6, 1);    // Ponte en la line 1, posición 6
  lcd.print("Menu") ;
}
//////////////////////////
void kp_lcd(void){
  lcd.clear();
  delay(1);
  lcd.setCursor(0, 0);
  lcd.print("kp:");  // Aquí va el mensaje
  lcd.setCursor(0, 1);    // Ponte en la line 1, posición 0
  lcd.print(kp) ;
}
//////////////////////////
void ki_lcd(void){
  lcd.clear();
```

```
  delay(1);
  lcd.setCursor(0, 0);
  lcd.print("ki:");  // Aquí va el mensaje
  lcd.setCursor(0, 1);    // Ponte en la line 1, posición 0
  lcd.print(ki) ;
}
////////////////////////
void kd_lcd(void){
  lcd.clear();
  delay(1);
  lcd.setCursor(0, 0);
  lcd.print("kd:");  // Aquí va el mensaje
  lcd.setCursor(0, 1);    // Ponte en la line 1, posición 0
  lcd.print(kd) ;
}
////////////////////////
void menu_1(void){
  lcd.clear();
  delay(1);
  lcd.setCursor(1, 0);
  lcd.print("PID.Variables");  // Aqi va el mensaje
  lcd.setCursor(5, 1);    // Ponte en la line 1, posición 5
  lcd.print("kp:") ;
}
////////////////////////
void menu_2(void){
  lcd.clear();
  delay(1);
  lcd.setCursor(1, 0);
  lcd.print("PID.Variables");  // Aquí va el mensaje
  lcd.setCursor(5, 1);    // Ponte en la line 1, posición 5
  lcd.print("ki:") ;
}
////////////////////////
void menu_3(void){
  lcd.clear();
  delay(1);
  lcd.setCursor(1, 0);
  lcd.print("PID.Variables");  // Aqi va el mensaje
  lcd.setCursor(5, 1);    // Ponte en la line 1, posición 5
  lcd.print("kd:") ;
}
////////////////////////
void botones_entrada(void){
  if (arriba== HIGH && aceptar==0){
    menu_v++;
    }else{
    menu_v=menu_v;
    }
  if (abajo== HIGH && aceptar==0  ){
```

```
  menu_v--;
  }else {
  menu_v=menu_v;
  }
if (menu_v >= 3){
  menu_v=3;
  }
if (menu_v <= 0){
  menu_v=0;
  }
if (set==HIGH  ){
  aceptar++;
  }
if (aceptar>=1 ){
  aceptar=1;
  }
if (back==HIGH ){
  aceptar--;
  }
if (aceptar<=0 ){
  aceptar=0;
  }
if (aceptar == 1 && menu_v==1){
  if (arriba == HIGH){
    kp=kp+0.01;
    }
  if (abajo == HIGH){
    kp=kp-0.01;
    }
  }
if (kp>=99){
  kp=99.0;
  }
if (kp<=0){
  kp=0.0;
  }
if (aceptar == 1 && menu_v==2){
  if (arriba == HIGH){
    ki=ki+0.01;
    }
  if (abajo == HIGH){
    ki=ki-0.01;
    }
}
if (ki>=99){
  ki=99.0;
}
if (ki<=0){
  ki=0.0;
}
```

```
  if (aceptar == 1 && menu_v==3){
    if (arriba == HIGH){
      kd=kd+0.01;
    }
  if (abajo == HIGH){
      kd=kd-0.01;
    }
  }
  if (kd>=99){
    kd=99.0;
    }
  if (kd<=0){
    kd=0.0;
    }
}
```

8.2 Código en arduino en el Cliente Sensor/Actuador

A continuación se presentan los códigos de arduino implementados en el cliente, que para este trabajo es la planta en donde están el sensor y actuador, se presentan códigos en función de la condición lógica que dispara el evento. Para lo cual se realizó en base a los criterios:

8.2.1 Códigos del Cliente con criterio Intervalo

```
#include <Servo.h>
#include <Bridge.h>       //Libreria para conexion entre arduino mega y modulo Yung
#include <HttpClient.h>   //Libreria para generar peticiones de cliente al servidor
#include <Console.h>      //Libreria para visualizar lo del puerto serial
/////////
int sen_1,sen_2,angulo_in,seguro;
float vol_1,vol_2,dist_1 ,dist_2, distancia, dist_1i,voltaje;;
String dato,envio,angulo;
int contador;
Servo motor;
float distancia_actual, distancia_anterior,vd;
////////
void setup() {
  Bridge.begin();
  Console.begin();
  motor.attach(9);
  angulo_in=100;
  analogReference(EXTERNAL);
}
//////////
void loop() {
  HttpClient client;
  sensor();
  envio = "http://192.168.0.155/dist:";
  envio += distancia;
  if (seguro== 0){
```

```
    client.get(envio);
    while (client.available()) {
      char c = client.read();
      angulo += c;
      if (c == '\n')
        {
        angulo_in=angulo.toInt();
        angulo="";
        }
    }
    envio="";
  }
  ////////////////////
  if (distancia > 12.0 && distancia <17.0 ){
    contador++;
    }else{
    contador=0;
    seguro=0;
    }
  if (contador>=11){
    seguro =1;
    }
  if (contador > 21){
    contador=0;
    }
  motor.write(angulo_in);
  delay(1);
}
//////Subrrutina para lectura de sensores
void  sensor(void){
sen_1=analogRead(0);
voltaje=(sen_1*3.3)/1024.0;
distancia= 26.825*pow(voltaje,-1.227);
}
```

8.2.2 Código del Cliente con criterio de Error

```
#include <Servo.h>
#include <Bridge.h>       //Libreria para conexion entre arduino mega y modulo Yung
#include <HttpClient.h>   //Libreria para generar peticiones de cliente al servidor
#include <Console.h>      //Libreria para visualizar lo del puerto serial
//////
int sen_1,sen_2,angulo_in,seguro;
float vol_1,vol_2,dist_1 ,dist_2, distancia, dist_1i,voltaje;;
String dato,envio,angulo;
int contador;
Servo motor;
int contador_t;
float distancia_actual, distancia_anterior,vd;
////////
```

```
void setup() {
  Bridge.begin();
  Console.begin();
  motor.attach(9);
  angulo_in=100;
  analogReference(EXTERNAL);
  seguro=0;
}
////////////
void loop() {
HttpClient client;
sensor();
vd=abs(distancia-distancia_anterior);
envio = "http://192.168.0.155/dist:";
envio += distancia;
if (seguro == 0){
  client.get(envio);
  while (client.available()) {
    char c = client.read();
    angulo += c;
    if (c == '\n'){
      angulo_in=angulo.toInt();
      angulo="";
    }
  }
  envio="";
}
//////////////////
if (distancia> 13.0 && distancia <16.5 ){
  if (vd <=0.09){
    seguro=100;
    }
  }else{
  if (vd>= 0.1 ){
    seguro=0;
    }
  }
motor.write(angulo_in);
delay(1);
distancia_anterior= distancia;
}
/////////
void  sensor(void){
sen_1=analogRead(1);
voltaje=(sen_1*3.3)/1024.0;
distancia= 26.825*pow(voltaje,-1.227);
}
```

8.2.3 Código del Cliente con criterio de acumulación del Error

```
#include <Servo.h>
#include <Bridge.h>       //Libreria para conexion entre arduino mega y modulo Yung
#include <HttpClient.h>   //Libreria para generar peticiones de cliente al servidor
#include <Console.h>      //Libreria para visualizar lo del puerto serial
////////
int sen_1,sen_2,angulo_in,seguro;
float vol_1,vol_2,dist_1 ,dist_2, distancia, dist_1i,voltaje;;
String dato,envio,angulo;
int contador;
float error, resultado, valor;
int contador_integral;
Servo motor;
float distancia_actual, distancia_anterior,vd;
////////
void setup() {
  Bridge.begin();
  Console.begin();
  motor.attach(9);
  angulo_in=100;
  analogReference(EXTERNAL);
  contador_integral=0;
  valor=0.0;
}
////////////
void loop() {
  HttpClient client;
  sensor();
  envio = "http://192.168.0.155/dist:";
  envio += distancia;
  if (seguro== 0){
    client.get(envio);
    while (client.available()) {
      char c = client.read();
      angulo += c;
      if (c == '\n'){
        angulo_in=angulo.toInt();
        angulo="";
        }
      }
    envio="";
  }
  ////////////////////
  error =abs(distancia-15.0);
  if (error > -2.5 && error <2.5 ){
    contador++;
    }else{
    contador=0;
    seguro=0;
    contador_integral=0;
```

```
    resultado=0.0;
    valor=0.0;
    }
  if (valor>=7.0){
    seguro =1;
    }
  if (contador >= 11){
    contador=0;
    }
  motor.write(angulo_in);
  delay(1);
  //integral
  if (contador%2==0){
    contador_integral++;
    valor=valor+error;
    }
  if (contador_integral >= 10  ){
    contador_integral=0;
    }
}
////// Subrrutina para lectura de sensores
void  sensor(void){
sen_1=analogRead(0);
voltaje=(sen_1*3.3)/1024.0;
distancia= 26.825*pow(voltaje,-1.227);
}
```

8.2.4 Código del Cliente con criterio de la acumulación del Error al cuadrado

```
#include <Servo.h>
#include <Bridge.h>       //Libreria para conexion entre arduino mega y modulo Yung
#include <HttpClient.h>   //Libreria para generar peticiones de cliente al servidor
#include <Console.h>      //Libreria para visualizar lo del puerto serial
//////
int sen_1,sen_2,angulo_in,seguro;
float vol_1,vol_2,dist_1 ,dist_2, distancia, dist_1i,voltaje;;
String dato,envio,angulo;
int contador;
float error, resultado, valor,error_cuadrado;
int contador_integral ;
Servo motor;
float distancia_actual, distancia_anterior,vd;
////////
void setup() {
  Bridge.begin();
  Console.begin();
  motor.attach(9);
  angulo_in=100;
  analogReference(EXTERNAL);
  contador_integral=0;
```

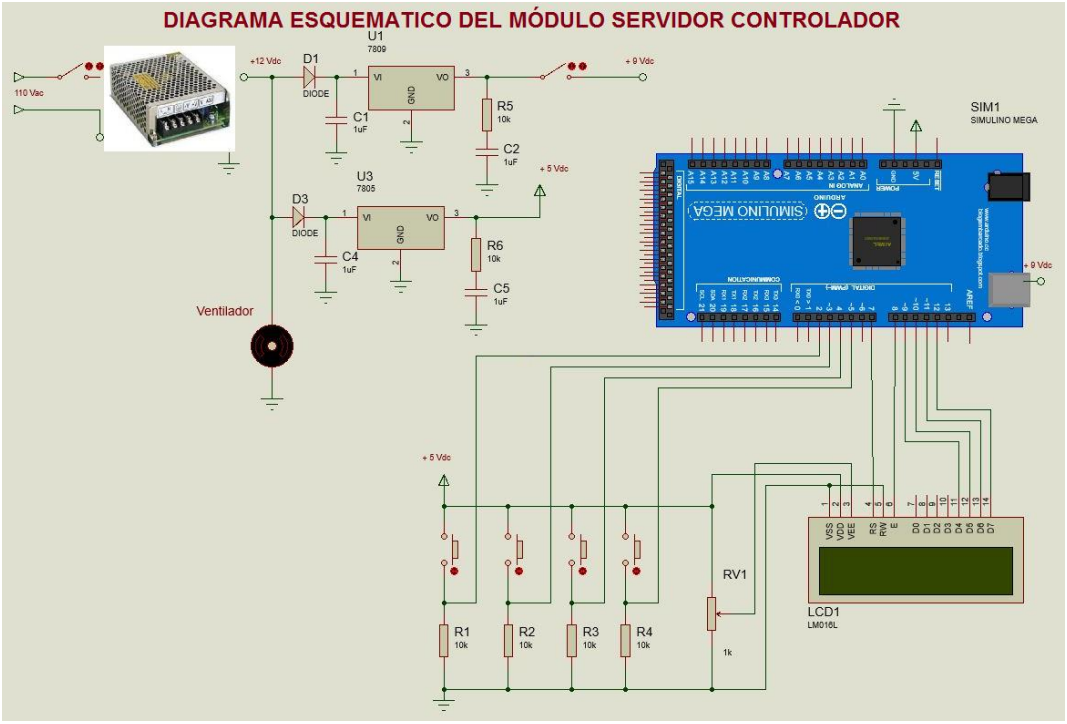
```
  valor=0.0;
}
//////////
void loop() {
  HttpClient client;
  sensor();
  envio = "http://192.168.0.155/dist:";
  envio += distancia;
  if (seguro== 0){
    client.get(envio);
    while (client.available()) {
      char c = client.read();
      angulo += c;
      if (c == '\n'){
        angulo_in=angulo.toInt();
        angulo="";
        }
    }
  envio="";
  }
  ////////////////////
  error =abs(distancia-15.0);
  error_cuadrado = error*error;
  if (error >-2.5 && error <2.5 ){
    contador++;
    }else{
    contador=0;
    seguro=0;
    contador_integral=0;
    resultado=0.0;
    valor=0.0;
    }
  if (valor>=11.0){
    seguro =1;
    }
  if (contador >= 11){
    contador=0;
    }
  motor.write(angulo_in);
  delay(1);
  //integral
  if (contador%2==0){
    contador_integral++;
    valor=valor+error_cuadrado;
    }
  ///
  if (contador_integral >= 10  ){
  contador_integral=0;
  }
}
```

```
////// Subrrutina para lectura de sensores
void  sensor(void){
sen_1=analogRead(0);
voltaje=(sen_1*3.3)/1024.0;
distancia= 26.825*pow(voltaje,-1.227);
}
```

9 ANEXO C

Diagramas esquemáticos.

9.1 Diagrama esquemático del módulo Servidor



9.2 Diagrama esquemático del módulo Cliente