UNIVERSIDAD DE ALCALÁ

Escuela Politécnica Superior

GRADO EN INGENIERÍA EN ELECTRÓNICA Y AUTOMÁTICA INDUSTRIAL

Trabajo Fin de Grado

Aplicación de Técnicas de Estimación Basadas en Eventos a Sistemas de Control en Red

Autor: Myrlene Félicité Bilo'o

Director: D. Felipe Espinosa Zapata

Co Director: D. Miguel Martínez Rey

TRIBUNAL:

Presidente: D. Ignacio Fernández Lorenzo

Vocal 1º: D. Manuel Guerra Martínez

Vocal 2º: D. Felipe Espinosa Zapata

CALIFICACIÓN: .....

FECHA:…….

Agradecimientos

A todos los que me han dado apoyo durante la realización de este proyecto, mi agradecimiento por vuestra disponibilidad, los ánimos y vuestra paciencia no es suficiente pero aun así, muchísimas gracias. Notamente a mi tío Jean Calvin Mbarga que sin él no habría realizado esos estudios, al director de este proyecto Felipe Espinoza Zapata, al co-director Miguel Martínez Rey, a Luana Bruno y a Anatole Anguele Anguele.

Índice

Resumen

Los estimadores de estados cumplen una función fundamental en el diseño de controladores en el espacio de estados. En los sistemas de control en los que sensor y controlador comparten una red inalámbrica, se tiende a utilizar técnicas de estimación aperiódica. En este TFG se realiza una comparativa entre los estimadores periódicos y los basados en eventos (aperiódicos), tomando como referencia el Filtro de Kalman. Se analizan y comparan dos técnicas de estimación basada en eventos: send-on-delta y send-on-area. Se describe el principio de funcionamiento y se aplica a la estimación de estados de un robot P3-DX.

Palabras claves:

Filtro de Kalman, Estimación de estados, Estimación aperiódica, Send-on-delta, Send-on-delta Integrado.

Abstract

Event based estimators play a fundamental role in the design of controllers in state space representation. In control systems where the sensor and controller share a wireless network, aperiodic estimation is using. During this end of grade work, it is done a comparison between periodic estimators and event based estimators (aperiodic), taking the Kalman Filter as a reference. The two event based estimators that are analysed and compared are the following: send-on-delta and send-on-area. The operating principle is described and applicate to estimated states of P3-DX robot.

Keywords:

Kalman Filter, State estimation, aperiodic Estimation, Send-on-delta, Send-on-area.

Resumen extendido

Hoy en día, las técnicas de estimación basada en eventos (EBE) están en su cénit, por la aplicación de las nuevas tecnologías a sistemas de control en red en distintos ámbitos como plantas industriales por ejemplo las cadenas de producción, las centrales eléctricas, el seguimiento de satélites, el seguimiento del tráfico que sea aéreo, marítimo, o terrestre, además en robótica donde el ámbito al que son utilizados puede ser espacial, industrial, doméstico, sanitario y lúdico. Hay también una grande aplicación de esos sistemas en los medios de transporte etc.

Dichas técnicas permiten que los sensores aporten información al algoritmo de control solo cuando sea necesario, con el correspondiente ahorro de recursos en el lazo de control, especialmente si la conexión entre los sistemas de sensado y control es remota. El enlace en cuestión es una red inalámbrica por lo que no existe contacto cableado entre el dispositivo destinado a sensar y el dispositivo que ejerce de controlador. Hay que precisar que las redes inalámbricas se diferencian entre si sea por su frecuencia de transmisión o la velocidad de esta última. Pero lo que más las vuelven útiles para los sistemas de control es que no requieren una grande inversión para su instalación como los sistemas cableados que requieren toda una reforma para que los cables no estorben el área de trabajo.

De ahí nace el interés en aplicar algunas técnicas como el *send-on-delta*, y su variante integrada llamada *send-on-area*, al robot Pioneer3-DX (P3-DX) cuyo modelo servirá de planta en este trabajo. Esos dos métodos de estimación son aperiódicos y para conseguir que sean así se aplica un umbral de disparo.

En general la estimación consta de dos fases, una de predicción basada en el modelo y otra de corrección a partir de la medida de los sensores. En el caso del FK la medida se realiza periódicamente y, por tanto, se corrige el estado de la planta cada periodo de muestreo (Ts). En el caso del *send-on-delta*, se actualizan los estados si y solo si la diferencia entre las variables medidas y las últimas enviadas supera un cierto umbral. En el caso del *send-on-area*, se integra la diferencia entre el valor medido y el último enviado y se corrigen el estado si la integral supera el umbral determinado. Cuando los estados se renuevan, hablamos de disparos o eventos.

Ahora se diferencian de las técnicas de estimación periódicas a la hora de la corrección, porque esta se dispara si y solo si se supera el umbral de disparo. Pero antes de emplear esos procesos se va a aplicar el filtro de Kalman al mismo robot y luego comparar los resultados obtenidos posteriormente de las estimaciones aperiódicas. Para poder comparar se van a establecer unos parámetros comunes en las tres técnicas porque de lo contrario no se podría contrastar los resultados entre sí.

La ganancia estática del filtro de Kalman, servirá de punto común en las tres técnicas empleadas pero antes de usarlo, se va analizar si no causaría otros impactos en los resultados y si sería igual de eficiente que calcular la ganancia de Kalman cada vez que hay estimación y corrección en el control, después de despejar esas dudas se procedería a su implementación y aplicación.

El experimento en cuestión se hace en el dominio temporal discreto, o sea, que los datos son recogidos cada cierto intervalo de tiempo conocido como el tiempo de muestreo, constante en el caso periódico (FK) y variable en el caso del SoD y del SoA. Como requisito el modelo de la planta de ser discretizado y hay que establecer un tiempo de sensado fijo que es de 10 ms en este trabajo. Pero durante las simulaciones se hará una diferencia entre el tiempo de sensado y el tiempo de muestreo, esa diferencia consiste en que el tiempo de sensado será el que tarda el sensor en captar nuevas medidas y el tiempo de muestreo será el que transcurre entre dos fases de corrección consecutivas en la estimación periódica. Existe una relación de proporcionalidad entre esos dos tiempos y el tiempo de muestreo debe ser mayor o igual al tiempo de sensado. Después de análisis se va a elegir tres tiempos de muestreo y por cada criterio de estudio se aplicarían esos últimos por cada tiempo de muestreo. Eso quiere decir que al tener tres periodos de muestreo distintos y tres técnicas de estimación, se va a calcular una ganancia estática por periodo de muestreo y aplicando esa ganancia a cada técnica se obtendrá nueve resultados por índice de comparación.

El objetivo que se requiere alcanzar al aplicar esas técnicas basadas en eventos es conseguir que el flujo de transmisión por la red inalámbrica sea reducido y que eso no provoque inestabilidades en el proceso controlado. Además se van a comparar los márgenes de errores generados por proceso.

En definitiva, se pretende responder a la pregunta:" ¿en qué circunstancias interesaría utilizar una u otra técnica?". El estudio realizado en este TFG permite afirmar que utilizando una técnica de estimación basada en eventos se puede llegar a controlar una planta de forma estable reduciendo las aportaciones del subsistema sensorial y, por tanto, ahorrando recursos del sistema de control.

El trabajo se subdivide en cuatros temas donde el primero es introductor y trata de las generalidades de los sistemas de control y en particular de los sistemas de control en red, además se introduce las redes sensoriales inalámbricas cuya importancia es crucial porque han permitido que el control se pueda realizar de forma remota y cada vez más fiables. En la misma lógica introductora se presenta las diferentes configuraciones de una red, los tipos de redes que se usan en la actualidad y las técnicas empleadas a continuación en el resto del trabajo. El segundo tema habla más detenidamente de los métodos usados, en primer lugar del *filtro de Kalman* exponiendo sus principios y particularidades, en segundo lugar del *send-on-delta* y para terminar con el *send-on-area*. Cada proceso es aplicado a un mismo modelo de seguimiento de un objeto en una dimensión como ejemplo de aplicación práctica después de describirlos con ecuaciones. Después de eso viene el tercer tema donde se estudia y se analiza las tres técnicas aplicándolas al modelo del robot P3-DX. Sin embargo la primera parte de este tema se centra en buscar índices de comparación entre las técnicas ya que de no ser así la comparación no tendría sentido debido a que las tres técnicas tienen unos criterios de disparos completamente diferentes. Al conseguir esos índices, se pasa a la determinación de los umbrales de disparo hay que recordar que por cada índice asociado a un tiempo de muestreo hay un umbral de disparo y para conseguirlo hay que realizar

muchas simulaciones empezando siempre con un umbral más bajo que se ve incrementado hasta conseguir el que cumple con las condiciones de simulación. Luego se procede a la obtención de los resultados en forma de tablas y gráficas porque con las gráficas se ve observa la distribución de los eventos y se interpreta mejor los resultados de las tablas. Se termina con la interpretación de esos resultados. Para terminar, el tema cuatro presenta las conclusiones generales resultantes de las simulaciones y los trabajos futuros que podrían derivar del estudio realizado. Los códigos utilizados para las simulaciones se encuentran en anexos junto con algunos anexos en matemática que se han considerado importante para entender algunos aspectos de las técnicas de estimación.

1- Introducción

Un sistema es una combinación de componentes que interactúan juntos para conseguir un objetivo determinado [1]. Por lo tanto puede considerarse como sistema cualquier conjunto de instrumentos que realizan una tarea, agrupa varios componentes y está compuesto de distintos elementos como las variables internas, las señales de control, las señales de perturbación, las salidas medibles y las entradas energéticas.

Para controlar un sistema, es necesario saber cómo responde frente a los cambios de los elementos que lo forman. En general, la respuesta de un sistema dado está formada por una componente estática y otra transitoria las cuales constituyen la dinámica del sistema mismo.

En ingeniería de control se emplean teorías basadas en comportamiento o teorías basadas en modelado para determinar la dinámica de un sistema.

En este trabajo se utilizarán las técnicas de estimación basadas en eventos. Éstas forman parte de la teoría de control basada en el modelado de la planta. El modelo es una representación matemática del comportamiento del sistema y para obtenerlo se requiere el registro de la respuesta real del sistema, frente a diferentes entradas.

1.1- Sistemas de control en red y red sensorial inalámbrica

Una clasificación básica de los sistemas de control lleva a establecer dos grandes grupos: lazo abierto y lazo cerrado. Los sistemas en lazo abierto, se caracterizan por una entrada no influenciada por la salida, no compensan perturbaciones ni ruido externo a la propia planta y presentan una alta sensibilidad a variaciones en el comportamiento de la misma (*figura 1.1).*

Los sistemas de control en lazo cerrado son más robustos a las perturbaciones, ruido e incertidumbres del modelo gracias a la realimentación (*figura 1.2).* En este grupo se sitúan los sistemas de control en red (NCS), los cuales utilizan un canal de comunicación compartido por múltiples procesos (*figura 1.3).* Estos últimos suelen utilizar redes sensoriales inalámbricas (WSN) (*figura 1.4)* para aportar información al dispositivo de control.

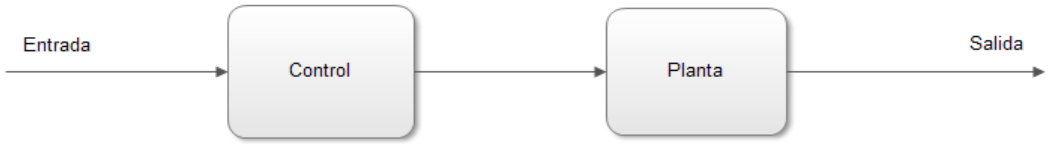

Figura 1. 1. Sistema de control en lazo abierto.

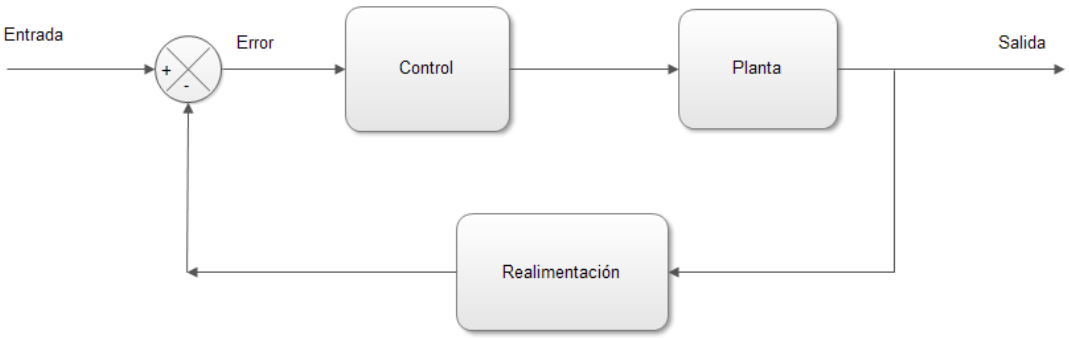

Figura 1. 2. Sistema de control en lazo cerrado.

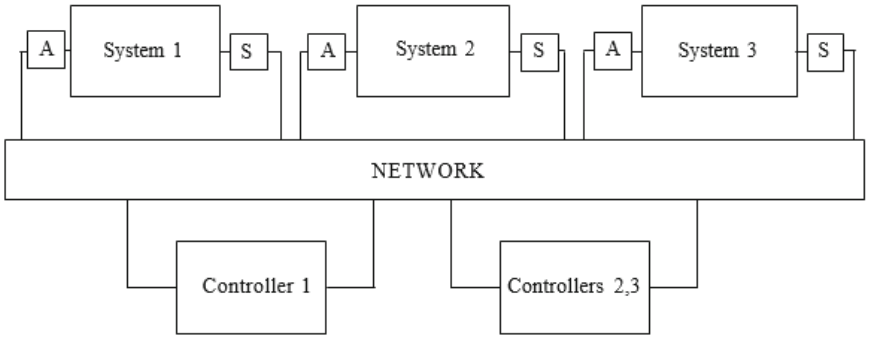

Figura 1. 3. Sistema de control en red.

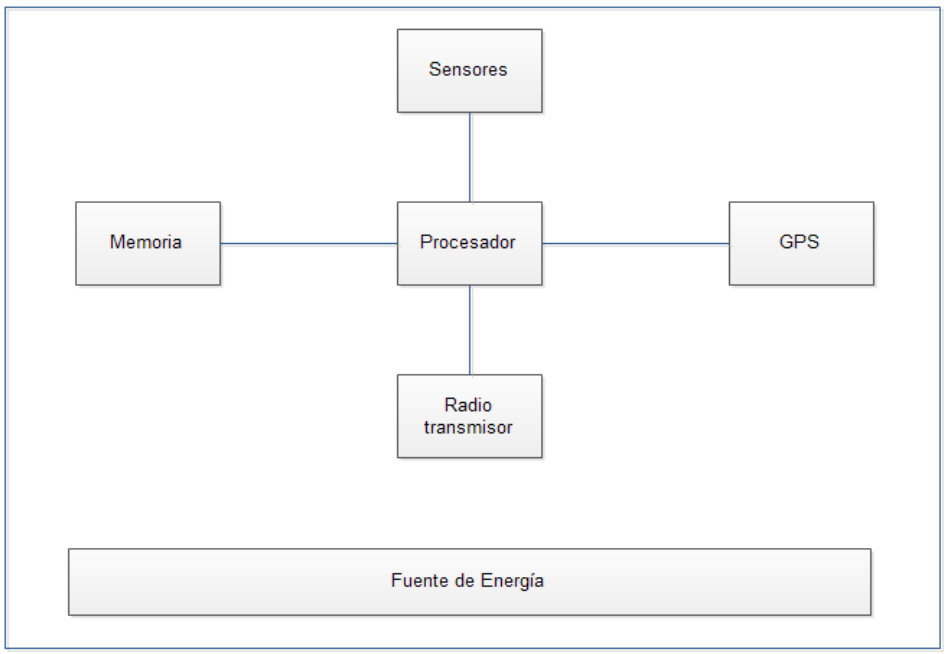

Figura 1. 4. Esquema básico de una red sensorial inalámbrica.

1.1.1- Sistemas de control en red

En un sistema de control en red la comunicación, entre los actuadores, los sensores y los dispositivos de control, se hace a través de un canal (cableado o inalámbrico). El bucle de control se cierra a través de este canal, que es el recurso distribuido y compartido del sistema (*figura 1.3*).

Un ejemplo de sistema de control en red podría ser un edificio donde la temperatura de cada sala/habitación (planta) se controla desde un centro remoto. En cada sala se instala un regulador de presencia de personas y de temperatura ambiente. De esta forma las salas ocupadas se mantienen a la temperatura fijada por el usuario o por el centro remoto, para ello son necesarios sensores que miden la temperatura cada cierto tiempo y la comunican al centro de control.

La información sensorial va en un sentido planta-control y señal de actuación (calefacción/refrigeración) en sentido control-planta .

Los sistemas de control en red tienen en común algunas características como el intercambio de señales de control de diferente naturaleza. En el ejemplo precedente, la actuación para conseguir la temperatura deseada tiene un determinado rango de valores (varios bits de información) mientras que la señal de presencia/ausencia solo requiere un bit. Otra característica es el tiempo crítico o retardo máximo del canal permitido para no desestabilizar el lazo realimentado.

Partes de un sistema de control en red

Planta: Los procesos de control se aplican a una determinada planta (proceso controlado) que tiene sus entradas, salidas, variables internas y señales externas no deseadas (perturbaciones y ruido). Ejemplo de planta controlada puede ser una máquina, o una parte de ella, que esté realizando una operación concreta, un robot móvil, un horno de calefacción, una cinta transportadora etc.

Sensor: Es un dispositivo que capta magnitudes físicas (mecánicas, eléctricas, etc) o químicas y proporciona una magnitud generalmente eléctrica que es función de estas. En sistemas de control en red es común la utilización de redes sensoriales inalámbricas (WSN) porque son de fácil instalación y mantenimiento, además simplifican el diseño del sistema al no requerir un cableado complejo. En el caso de robots móviles, los sensores pueden estar embarcados (encóder) o instalados en el entorno (cámaras de vídeo).

Actuador: Se encarga de convertir las órdenes del controlador en acciones sobre la planta. Dependiendo de la aplicación, son eléctricos, hidráulicos o neumáticos. En el caso de robots móviles, los actuadores suelen ser motores de continua o paso a paso.

Controlador: Es un dispositivo diseñado para detectar los desvíos de la señal controlada respecto a la señal de referencia, emitiendo una señal de corrección hacia los actuadores. En el caso de control remoto de un robot móvil el controlador suele ser un microcontrolador o un PC que forma parte de la infraestructura del entorno de control (no embarcado).

Canal de comunicación: Es el medio por el cual el controlador recibe información de los sensores y envía órdenes a los actuadores, en un sistema de control en red. Sirve también para transferir información a otros dispositivos no relacionados con el control, como centros de supervisión. El canal se dimensiona con un tamaño de paquetes suficiente para la información transmitida (sensado y control) y con un ancho de banda (paquetes/unidad de tiempo) suficiente para no interferir en la dinámica del lazo de control.

Principio de funcionamiento y limitaciones de un sistema de control en red

Cuando la planta está en funcionamiento, los sensores miden las variables que se están controlando y mandan los datos obtenidos a través del canal al controlador. Después de la recepción de los datos, el controlador los compara con la referencia y, en función del error resultante, envía o no señales de corrección a los actuadores. Entre el sensor y el controlador hay un codificador que permite la digitalización de los datos mientras que, entre el controlador y el actuador, está el decodificador para pasar la orden del formato digital discreto al formato de actuación (*figura 1.5)*. En caso de un

canal no ideal, pueden tener lugar retardos y/o pérdida de paquetes que contribuyen a empeorar el comportamiento del sistema en lazo cerrado.

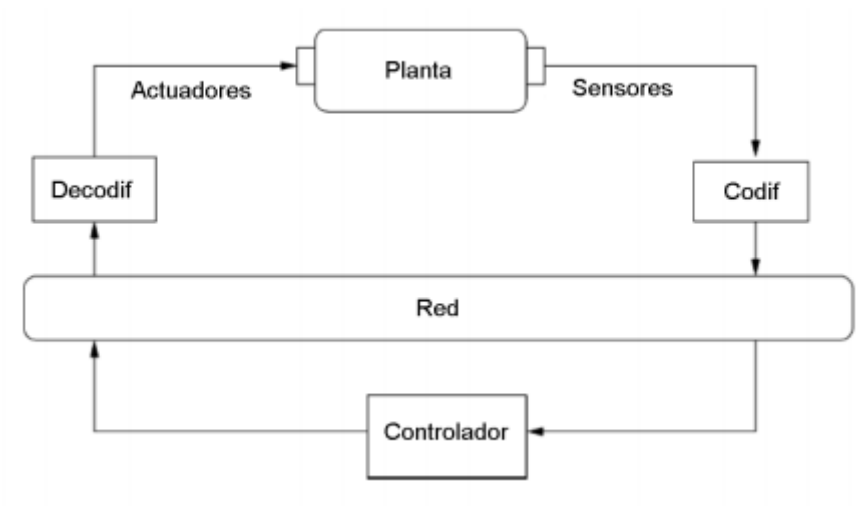

Figura 1. 5. Sistema de control en red de una planta

Anteriormente se ha hablado del canal de comunicación como un recurso concurrente; esto requiere de un protocolo que evite problemas como: pérdida de información, solape al enviar información de más de un dispositivo, retardo debido a la espera de que se termine una transmisión distinta a la que requiere la atención, pérdida de paquetes. La gestión de las transmisiones es similar a las interrupciones de mismo nivel de prioridad en programación en C, porque si una interrupción A está siendo atendida y llega otra B de misma prioridad, no se ejecutaría la rutina de atención de B hasta que se acabe de ejecutarse la de A.

Los NCS tienen dos configuraciones, una donde el sensor está integrado en la planta pero el control es remoto *figura 1.6* y otra, menos común, donde el controlador y sensor aun estando en el mismo sistema físico que la planta intercambian información a través de un canal de comunicación (*figura 1.7)*.

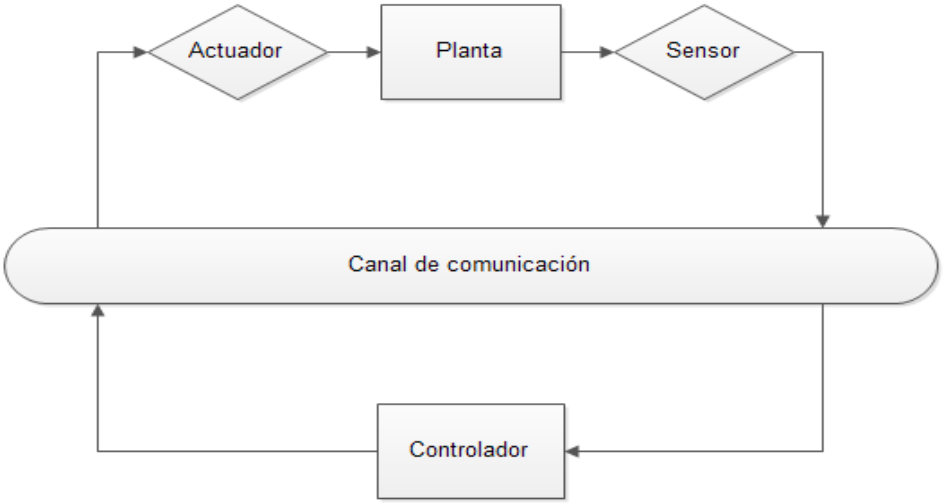

Figura 1. 6. Sistema de control en red con conexión directa entre planta y sensor.

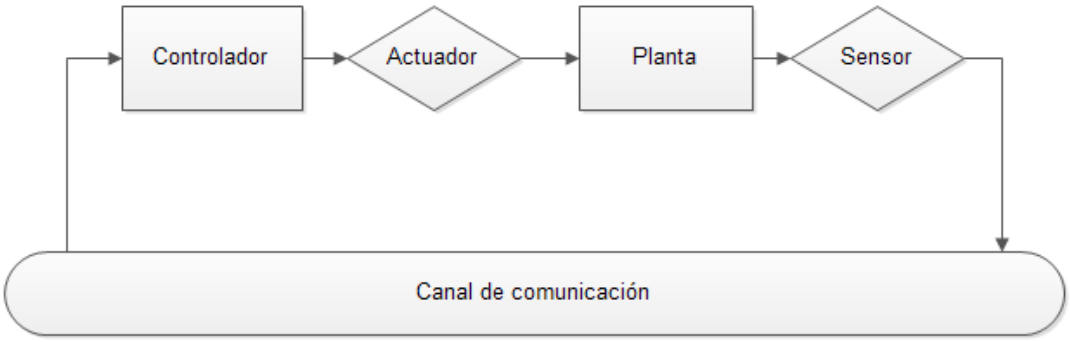

Figura 1. 7. Sistema de control en red con conexión directa entre controlador y actuador.

En cualquiera de las dos configuraciones, existe una limitación debida al ancho de banda del canal B, o a la transmisión de un número finito de datos por unidad de tiempo. Además hay una limitación de comunicación tal como lo demuestra el teorema de Shannon-Hartley para canales de comunicación [9]:

$$C = B * \log_2\left(1 + \frac{S}{N}\right) \quad (1.1)$$

C: la capacidad del canal en bits;

B: el ancho de banda del canal;

S: la potencia de la señal útil en W o mW;

N: la potencia de ruido presente en el canal en W o mW.

1.1.2- Red Sensorial inalámbrica (WSN)

De forma genérica una red sensorial inalámbrica integra módulos sensoriales formados por un sensor, una memoria, un dispositivo de localización, un radio transmisor, un procesador y una fuente de energía (*figura 1.4)*. En este apartado se habla de las características, las aplicaciones y los desafíos de las WSNs, seguido de los protocolos que utilizan, su topología y el estado de arte en su desarrollo.

Características, aplicaciones y desafíos de las WSNs

Cada componente de una WSN tiene las siguientes especificaciones:

- Procesador de bajo potencia que está limitado por los procesos que puede ejecutar;
- La memoria, al ser integrada en el dispositivo también tiene una capacidad de almacenamiento limitada.
- El radio transmisor con limitaciones de potencia y ancho de banda.
- Los sensores que se usan suelen ser escalares de baja potencia para captar magnitudes como la temperatura, presión, imagen o sonido.
- La fuente de energía para WSNs suelen ser baterías o pilas.

Las WSNs tienen múltiples aplicaciones: monitorización de procesos ecológicos, biológicos y sísmicos; vigilancia, rastreo de dispositivos; monitorización de procesos industriales y comerciales, espacios inteligentes, etc.

A pesar de esos avances sigue habiendo limitaciones que superar, tales como:

- Las limitaciones energéticas (pilas) que implican la relación rendimiento-autonomía como criterio de calidad de una WSN.
- La auto-reconfiguración porque son dispositivos remotos.
- La escalabilidad ya que existe una cantidad importante de nodos.
- Heterogeneidad para trabajar con dispositivos de varias prestaciones, sensores de diferentes modalidades y asegurar una jerarquía desplegable.
- Adaptabilidad en ajustarse con las condiciones de operación y en cambios de requerimientos de las aplicaciones.
- Seguridad y privacidad frente a la hostilidad del entorno de trabajo o por la potencial sensibilidad de la información recogida.

Topología de la red

La red de comunicación puede estar configurada en muchas topologías diferentes. En este TFG se comentan las cuatro principales [10]:

- La topología en estrella: es la más simple, cada nodo comunica directamente sus medidas al nodo de salida. Su realización es simple pero limitada por la poca escalabilidad y robustez que se consigue. Además en nodos alejados la interconexión es reducida (*figura 1.8.a)*.
- La topología en malla: permite que cada nodo esté conectado a otros nodos próximos. En caso de estar alejados, es necesario un enrutamiento, para ello se recurre a una malla arbitraria (*figura 1.8.b)* o una malla ordenada (*figura 1.8.c)*.
- La topología mixta: es una combinación de las dos primeras topologías, interesa utilizarla cuando la red es muy extensa ya que se reparte naturalmente en zonas locales de interconexión (*figura 1.8.d)*.

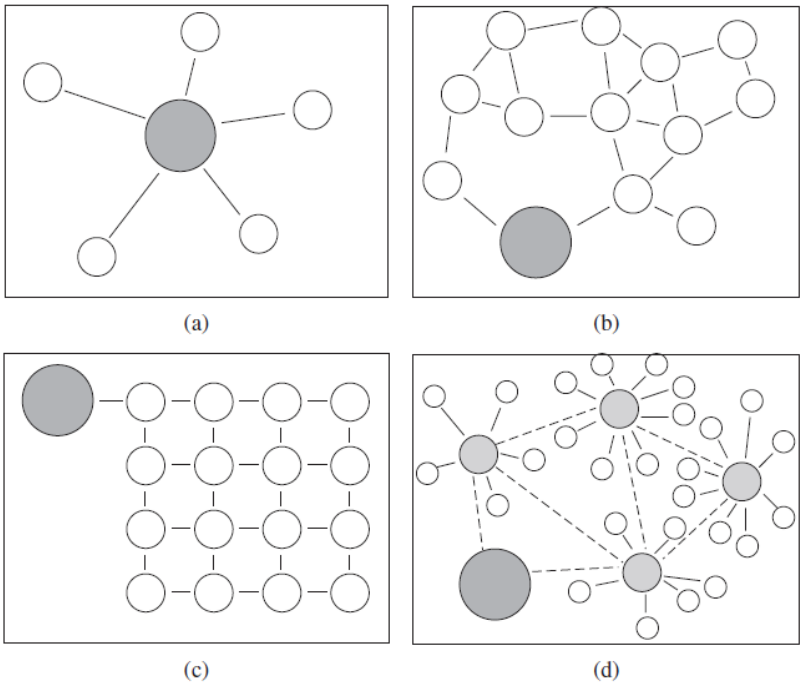

Figura 1. 8. Diferentes topologías de una red de comunicación en WSN: (a) en estrella, (b) en malla arbitraria, (c) en malla ordenada y (d) topología mixta.

Las redes pueden ser proactivas o reactivas. Las primeras monitorizan el entorno y mandan la información a una frecuencia constante, mientras que en las segundas, los nodos envían la nueva información solo cuando la variable monitorizada supera un determinado umbral.

Los WSNs operan en la banda libre de 2,4GHz, donde se encuentran tres estándares mayoritarios a saber:

- IEEE 802.11 para “wireless local area networks” (WLAN/Wi-Fi);
- IEEE 802.15.1 para “wireless personal area networks” (WPAN/Bluetooth);
- IEEE 802.15.4 para “low-rate wireless personal area networks” (LR-WPAN);

El IEEE 802.15.4, debido a que trabaja con bajas frecuencias de transmisión de datos y consume poca potencia, es idóneo para sistemas que utilizan las baterías como fuente de energía (WSNs). Existen diferentes arquitecturas de protocolos de comunicación inalámbricas basadas en este estándar, como ZigBee, ISA100.11a, WirelessHART y MiWi. Permiten cubrir con mayor flexibilidad, la variedad de aplicaciones.

1.2- Técnicas de estimación de estados basados en eventos

El proceso de estimación de estados aplicando la técnica del filtro de Kalman requiere de dos fases: una de predicción basada en el modelo de la planta, y otra de corrección a partir de las variables sensadas [2] [3]. Se habla de estimación basada en eventos cuando la información sensorial está disponible (suministrada por la red sensorial) solo y cuándo se cumple una cierta condición (evento) predeterminada por el diseñador.

Hacia 1960, Rudolf E. Kalman, desarrolló un algoritmo estimador de estados, conocido como filtro de Kalman (KF) y que es la referencia en estimación estocástica. A partir él, se han elaborado y publicado otras técnicas como las de estimación aperiódica: *send-on-delta* (SoD) y *send-on-delta* integrado (SoA). La estimación se hace de forma análoga en KF, *send-on-delta* y *send-on-delta* integrado, con una etapa de predicción y otra de corrección. En el caso de KF la corrección se hace de forma periódica, mientras que en el caso de las dos otras técnicas, la corrección se realiza si se cumple una determinada condición lo que permite reducir el número de accesos al canal de comunicación cuando el sistema sensorial y el de control están enlazados de forma inalámbrica.

1.3- Objetivos y organización

El auge de sistemas de control en red y sensores remotos, también conectados en red con el sistema de control, ha llevado a un creciente interés por las técnicas de estimación basada en eventos. El objetivo de este TFG es doble:

- analizar las técnicas más conocidas de estimación, periódica: filtro de Kalman, y basada en eventos: *send-on-delta* y el *send-on-delta* integrado; y
- aplicarlas al sensado remoto de robots P3-Dx (Pioneer3-DX) planteando una estructura como la de la figura 1.9.

Este trabajo se ha distribuido en cuatro temas donde el primero es la introducción de conceptos generales. El segundo se centra en los fundamentos matemáticos de las técnicas EBE mencionadas. En el tercer tema se aplican esas técnicas EBE a un sistema realimentado de control remoto de un robot. Y se cierra el trabajo con las conclusiones y las propuestas de trabajos futuros.

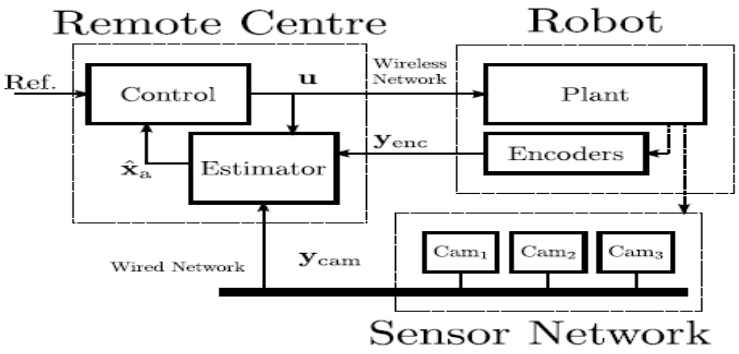

Figura 1. 98. Sistema de control en red de un robot. El control y el estimador basado en eventos se implementan en el centro remoto.

2- Técnicas de estimación

Estimar, según el diccionario, [10] significa valorar, y, a veces, se utiliza con el significado de calcular. Cuando se estima, se obtiene una magnitud mediante cálculos y suposiciones. Si los cálculos no son muy precisos, el valor será más grande o más pequeño que el valor estimado, mientras que si se ha hecho la estimación sabiendo cómo influyen todas las variables en la magnitud, el valor estimado será más cercano al valor real. Este tema se dedica en primer lugar al filtro de Kalman, seguido por dos técnicas de estimación aperiódica: *send-on-delta* y *send-on-delta* integrado. De aquí en adelante todo el estudio teórico se hará en dominio de tiempo discreto.

2.1- Filtro de Kalman

Una señal periódica es una señal que, al cabo de un intervalo de tiempo llamado periodo, se repite de la misma forma. Un ejemplo de señal periódica es el seno *figura 2.1*, de periodo 2π (π=3,14). Sin embargo la estimación periódica no significa que el valor se repetirá de la manera en cada periodo, sino que en un intervalo periódico de tiempo se calcula el valor estimado *figura 2.2*.

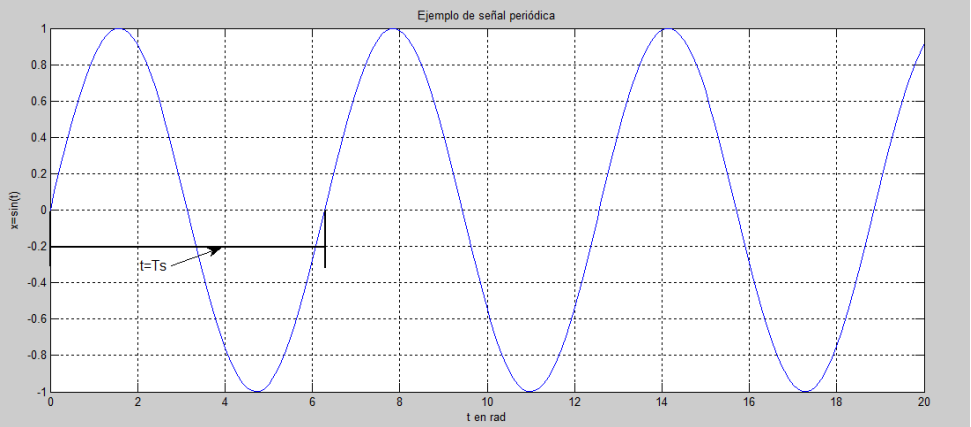

Figura 2. 1. Representación de la señal seno (x=sen (t)) donde Ts es el periodo y cada Ts se observa una réplica de la señal en el primer intervalo.

Se ve en la representación que el intervalo de tiempo de estimación es el mismo pero el valor estimado no tiene por qué coincidir con los valores anteriores aunque pueda dar un caso en el que haya dos resultados iguales. La estimación de la variable 1 está caracterizada por el cuadrado rojo y la estimación de la variable 2 por el triángulo azul.

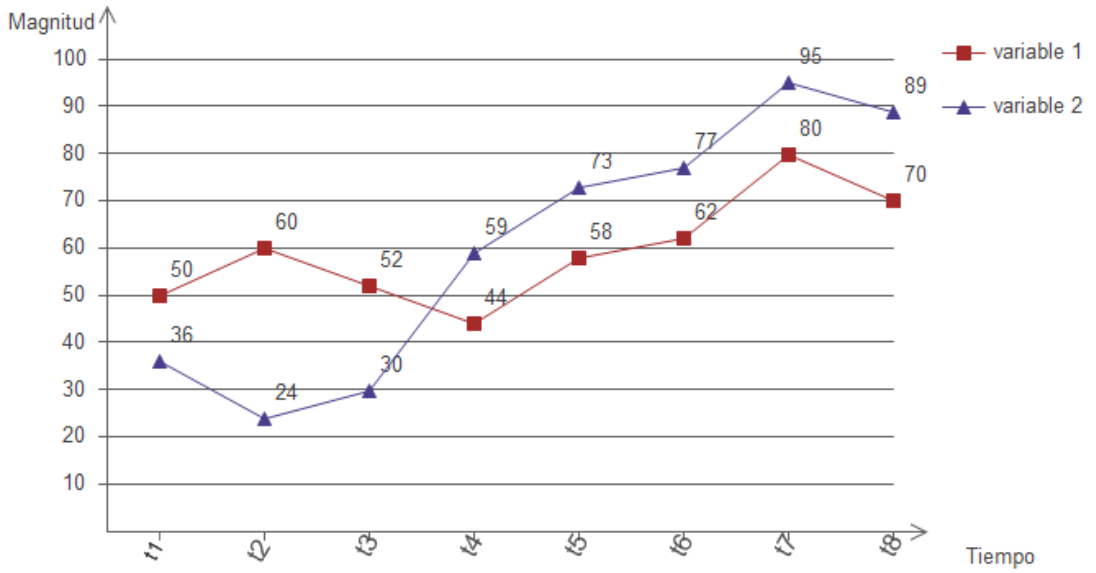

Figura 2. 2. Representación de dos variables estimadas periódicamente.

2.1.1-Fundamentos del Filtro de Kalman

En este bloque se describe el filtro de Kalman como estimador de estados periódico en sistemas con ruido (de estado y de medida) y se implementan las ecuaciones discretas que lo caracterizan.

Se supone un sistema lineal discreto y variante en el tiempo, representado por las siguientes ecuaciones:

$$x_{(k)} = F_{(k-1)}x_{(k-1)} + G_{(k-1)}u_{(k-1)} + w_{(k-1)}$$

$$y_{(k)} = H_{(k)}x_{(k)} + v_{(k)} \qquad (2.1)$$

Sabiendo que

- x $(_k)$ son las variables de estado del sistema,

- y $(_k)$ son las variables de salida;

- u $(_k)$ son las variables de entrada;

- v $(_k)$ y w $(_k)$ representan respectivamente el ruido de medida y el ruido del sistema.

Ambos son ruidos blancos, de media nula, incorrelados. Sus matrices de covarianzas son conocidas y son respectivamente $R_{(k)}$ y $Q_{(k)}$.

$$w\,(_k) \sim (0, Q_{(k)})$$

$$v(k) \sim (0, R_{(k)})$$

$$E\left[w_{(k)} w_{(j)}^{T}\right] = Q_{(k)}\delta_{(k-j)}$$

$$E\left[v_{(k)} v_{(j)}^{T}\right] = R_{(k)}\delta_{(k-j)}$$

$$E\left[v_{(k)} w_{(j)}^{T}\right] = 0 \qquad (2.2)$$

Donde $\delta\,(k - j)$ es la función delta de Kronecker, con las siguientes características:

- $\delta(k - j) = 1$ si $k = j$;
- $\delta(k - j) = 0$ si $k \neq j$;

En estimación de estados, si están disponibles todas las medidas de la salida hasta el instante k se puede hacer la estimación a posteriori $\hat{x}_{(k)}^{+}$. En caso de disponer de las medidas anteriores al instante k se puede hacer una estimación a priori $\hat{x}_{(k)}^{-}$, véase la *figura 2.3*.

$$\hat{x}_{(k)}^{+} = E\left[x_{(k)} | y_{(1)} y_{(2)} \ldots y_{(k)}\right] = \text{estimación a posteriori} \qquad (2.3)$$

$$\hat{x}_{(k)}^{-} = E\left[x_{(k)} | y_{(1)} y_{(2)} \ldots y_{(k-1)}\right] = \text{estimación a priori} \qquad (2.4)$$

$\hat{x}_{(k)}^{+}$ y $\hat{x}_{(k)}^{-}$ son estimaciones de las variables de estados $x_{(k)}$ solo que $\hat{x}_{(k)}^{-}$ se calcula antes de que las medidas en k estén disponibles y $\hat{x}_{(k)}^{+}$ cuando ya se han procesados las medidas del instante k. $\hat{x}_{(0)}^{+}$ es la estimación inicial de $x_{(k)}$.

La matriz de covarianza del error de estimación de $x_{(k)}$ está representada por $P_{(k)}$.

$P_{(k)}^{+}$ es la covarianza del error de estimación de $\hat{x}_{(k)}^{+}$;

$P_{(k)}^{-}$ es la covarianza del error de estimación de $\hat{x}_{(k)}^{-}$;

$$P_{(k)}^{-} = E\left[\left(x_{(x)} - \hat{x}_{(k)}^{-}\right)\left(x_{(k)} - \hat{x}_{(k)}^{-}\right)^{T}\right]$$

$$P_{(k)}^{+} = E\left[\left(x_{(x)} - \hat{x}_{(k)}^{+}\right)\left(x_{(k)} - \hat{x}_{(k)}^{+}\right)^{T}\right] \qquad (2.5)$$

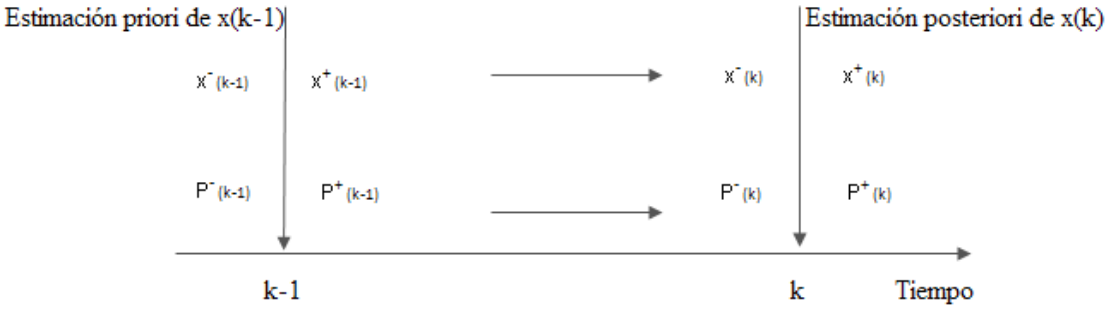

Figura 2. 3. Representación gráfica de la relación entre estimación a priori y/o a posteriori del estado y de la matriz de error de covarianza con el tiempo.

Para estimar los estados siguientes al instante inicial ($\hat{x}_{(0)}^{+}$= E ($x_{(0)}$)), se parte de la ecuación que describe la propagación de la media de los estados con el tiempo.

$$\hat{x}_{(1)}^{-}=F_{(0)}\hat{x}_{(0)}^{+}+G_{(0)}u_{(0)}$$

$$\hat{x}_{(2)}^{-}=F_{(1)}\hat{x}_{(1)}^{+}+G_{(1)}u_{(1)}$$

$$\cdots$$

$$\hat{x}_{(k-1)}^{-}=F_{(k-2)}\hat{x}_{(k-2)}^{+}+G_{(k-2)}u_{(k-2)}$$

$$\hat{x}_{(k)}^{-}=F_{(k-1)}\hat{x}_{(k-1)}^{+}+G_{(k-1)}u_{(k-1)} \qquad (2.6)$$

Del instante $(k-1)^{+}$ al instante $(k)^{-}$, al no recibir nuevas medidas en ese intervalo de tiempo, solo se puede recurrir al modelo de la dinámica del sistema para predecir los estados del sistema.

Para calcular la matriz de covarianza del error de estimación, se supone una matriz inicial $P_{(0)}^{+}$, cuyo valor dependerá de si el estado inicial $\hat{x}_{(0)}$ es conocido o no. En caso de ser conocido $P_{(0)}^{+}$=0, de lo contrario $P_{(0)}^{+}=\infty I$. Donde I es la matriz identidad. $P_{(0)}^{+}$ representa la incertidumbre de la estimación del estado inicial $x_{(0)}$.

$$P_{(0)}^{+}=E\left[\left(x_{(0)}-\hat{x}_{(0)}^{+}\right)\left(x_{(0)}-\hat{x}_{(0)}^{+}\right)^{T}\right] \qquad (2.7)$$

A partir del valor inicial de la matriz de covarianza del error de estados, se calcula la matriz de covarianza a priori con la siguiente expresión:

$$P_{(k)}^{-}=F_{(k-1)}P_{(k-1)}^{+}F_{(k-1)}^{T}+Q_{(k-1)} \qquad (2.8)$$

Las ecuaciones (2.6) y (2.8) forman la etapa de predicción de los estados y de la matriz de covarianza respectivamente. Las siguientes ecuaciones (2.9) representan la etapa de corrección.

$$K_{(k)} = P_{(k)}^{-} H_{(k)}^{T} \left(H_{(k)} P_{(k)}^{-} H_{(k)}^{T} + R_{(k)} \right)^{-1} \quad (2.9.i)$$

$$= \; P_{(k)}^{+} H_{(k)}^{T} R_{(k)}^{-1} \quad (2.9.ii)$$

$$\hat{x}_{(k)}^{+} = \hat{x}_{(k)}^{-} + K_{(k)} \left(y_{(k)} - H_{(k)} \hat{x}_{(k)}^{-} \right) \quad (2.9.iii)$$

$$P_{(k)}^{+} = \left(I - K_{(k)} H_{(k)} \right) P_{(k)}^{-} \left(I - K_{(k)} H_{(k)} \right)^{T} + K_{(k)} R_{(k)} K_{(k)}^{T} \quad (2.9.iv)$$

$$= \left[\left(P_{(k)}^{-} \right)^{-1} + H_{(k)}^{T} R_{(k)}^{-1} H_{(k)} \right]^{-1} \quad (2.9.v)$$

$$= \left(I - K_{(k)} H_{(k)} \right) P_{(k)}^{-} \quad (2.9.vi)$$

La matriz $K_{(k)}$ es la matriz de ganancia del filtro de Kalman. Su cálculo mediante la expresión (2.9.i) evita tener que disponer de la nueva medida, como ocurre con la expresión (2.9.ii).

En expresión (2.9.iii): $\hat{x}_{(k)}^{+} = \hat{x}_{(k)}^{-} + K_{(k)} \left(y_{(k)} - H_{(k)} \hat{x}_{(k)}^{-} \right)$, al término $y_{(k)} - H_{(k)} \hat{x}_{(k)}^{-}$ se le llama innovación y expresa la aportación de la nueva medida a la estimación (corrección) del estado. Se suele aproximar la media y la covarianza de la innovación mediante métodos estadísticos, y si esos valores son diferentes de los esperados significa que hay algún error en la implementación del filtro. El error puede venir de un modelado incorrecto del sistema o que el ruido considerado es erróneo.

2.1.2- Propiedades del filtro de Kalman

En este bloque, se habla de las propiedades importantes del filtro de Kalman. Se supone un sistema cuya dinámica es descrita por las ecuaciones anteriores (2.1 y 2.2):

$$x_{(k)} = F_{(k-1)} x_{(k-1)} + G_{(k-1)} u_{(k-1)} + w_{(k-1)}$$

$$y_{(k)} = H_{(k)} x_{(k)} + v_{(k)}$$

$$v_{(k)} \sim (0, Q_{(k)})$$

$$w_{(k)} \sim (0, R_{(k)})$$

$$E\left[w_{(k)}w_{(j)}^{T}\right] = Q_{(k)}\delta_{(k-j)}$$

$$E\left[v_{(k)}v_{(j)}^{T}\right] = R_{(k)}\delta_{(k-j)}$$

$$E\left[v_{(k)}w_{(j)}^{T}\right] = 0$$

El error entre el estado real $x_{(k)}$ y el estado estimado $\hat{x}_{(k)}$se denomina $\tilde{x}_{(k)}$, y el estimador es estable si el error de estimación:

$$\tilde{x}_{(k)} = x_{(k)} - \hat{x}_{(k)} \tag{2.10}$$

es nulo después de un régimen transitorio.

Suponiendo que se quiere hallar el estimador que minimiza el valor esperado de $\tilde{x}_{(k)}$:

$$\text{Min } E\left[\tilde{x}_{(k)}^{T}S_{(k)}\tilde{x}_{(k)}\right] \tag{2.11}$$

siendo $S_{(k)}$ una matriz definida positiva de ponderación elegida por el usuario. Si $S_{(k)}$ es diagonal con los elementos $S_{(k)}(1), S_{(k)}(2), \ldots, S_{(k)}(n)$; la suma ponderada sería igual a $S_{(k)}(1)E\left[\hat{x}_{(k)}^{2}(1)\right] + \ldots + S_{(k)}(n)E\left[\hat{x}_{(k)}^{2}(n)\right]$.

Consideraciones:

- Si $\{w_{(k)}\}$ y $\{v_{(k)}\}$ son gaussianos, de medias nulas, incorrelados y blancos, el filtro de Kalman es el estimador que minimiza la matriz de covarianza del error de estimación.
- Si $\{w_{(k)}\}$ y $\{v_{(k)}\}$ son correlados y coloreados, el filtro de Kalman ha de ser modificado para resolver el problema de minimización de $\tilde{x}_{(k)}$.
- Para sistemas no lineales, existen varias formulaciones del filtro de Kalman no lineales, como el filtro de Kalman extendido (EKF=Extended Kalman Filter) o el Unscented Kalman Filter (UKF).

Aplicación del filtro de Kalman a la estimación del seguimiento de un objeto

Se considera el movimiento de un objeto en una dimensión (línea recta). Se va a estimar la velocidad v(t) y la posición p(t) a partir de su aceleración a(t). Para la simulación se ha utilizado el fichero Matlab FK_1D.m.

Se sabe que en el dominio temporal

$$a(t) = \frac{d\,v(t)}{dt} = \frac{d^2p(t)}{dt^2}$$

Por lo tanto

$$p(t) = \iint a(t)dt \quad y \quad v(t) = \int a(t)dt$$

Los estados x(t) son la posición p(t) y la velocidad v(t) del objeto: $x(t) = \begin{bmatrix} p(t) \\ v(t) \end{bmatrix}$

A partir de una matriz de valores discretos de la aceleración (1D_aceleraction.mat), y sabiendo que como máximo puede valer 0,5 m/s^2. Se ha obtenido el modelo discreto siguiente donde la entrada u(k) es la aceleración del objeto, y el estado en tiempo discreto $x(k) = \begin{bmatrix} p(k) \\ v(k) \end{bmatrix}$, siendo Ts el periodo de muestreo.

$$x_{(k)} = F_{(k-1)}x_{(k-1)} + G_{(k-1)}u_{(k-1)} + w_{(k-1)}$$

$$y_{(k)} = H_{(k)}x_{(k)} + v_{(k)}$$

Donde y(k) es la position (x) del objeto.

Para el caso bajo estudio, con Ts = 0.1s, se tienen las matrices características:

$$F = \begin{bmatrix} 1 & Ts \\ 0 & 1 \end{bmatrix}; \; G = \begin{bmatrix} Ts^2/2 \\ Ts \end{bmatrix}; \; H = [1\ 0];$$

La matriz de covarianza del ruido de estado w se ha fijado en $Q = 0.02 * I_{2x2}$; y para el ruido de medida v se establece $R = 0.1e^{-3}$;

Y el estado inicial $x_{(0)} = [0\ 0]'$;

La representación de la matriz de aceleración, (1D_acceleration.mat) es la siguiente:

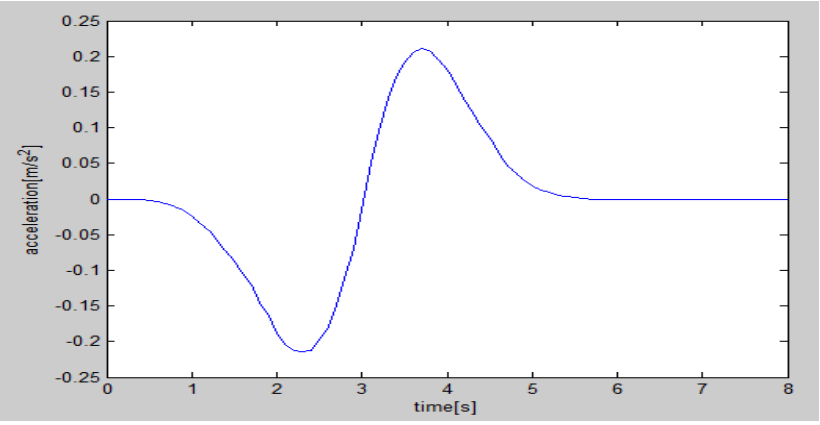

Figura 2. 4 Aceleración de referencia del objeto.

Sin aplicar técnicas de estimación y suponiendo que no hay ruido del sistema y tampoco de media, el primer estado que es la posición del objeto se representa de la forma siguiente:

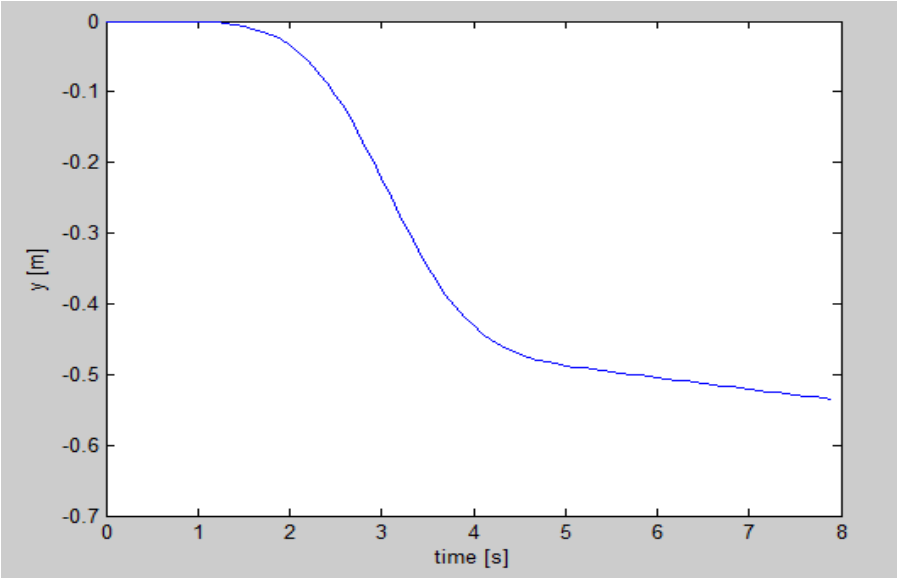

Figura 2. 5. Posición (x) del objeto.

Lo mismo para el segundo estado que es la velocidad del objeto:

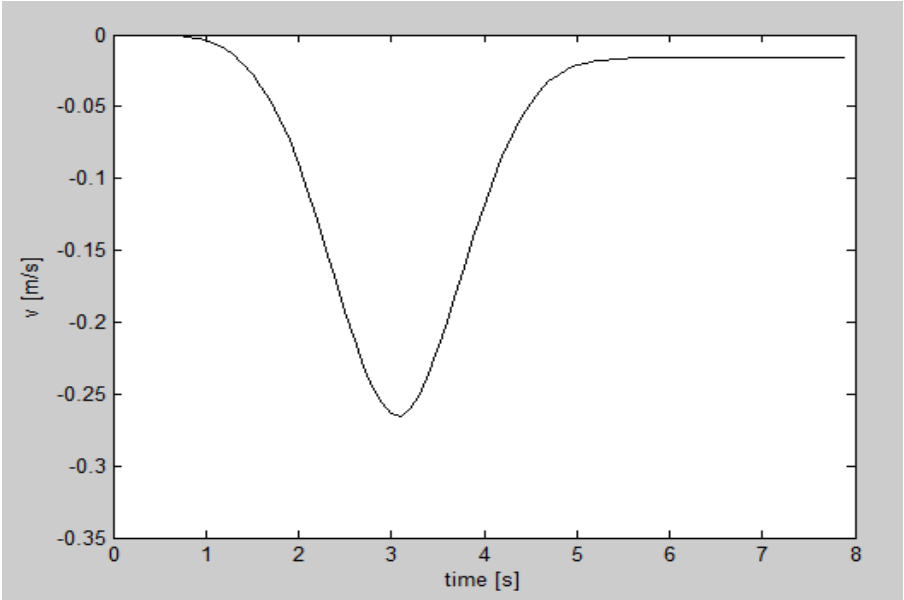

Figura 2. 6. Velocidad del objeto.

Teniendo en cuenta los ruidos de estado (w) y de medida (v), al usar el filtro de Kalman para estimar los estados se han obtenido las gráficas siguientes donde:

- Measurement: son los valores de la posición medidos por el sensor, con las mismas características de ruido R = $0.1e^{-3}$.
- Ideal: son los valores que tendría la posición sin ruido del sistema ni de medida.

- Estimated with periodical events: son los valores estimados de la posición aplicando el Filtro de Kalman.
- X-evento: son los disparos generados con FK.

El periodo de muestreo utilizado para tomar las medidas es de 0,1s.

Los valores eficaces de los errores de estimación de estados son los de la tabla siguiente

Tabla 1. Valores eficaces de los errores de estimación en posición y en velocidad usando el FK.

En posición $\varepsilon_{p,RMSE}$	En velocidad $\varepsilon_{v,RMSE}$	Número de disparos
0,00686 m	0,03729m/s	80

Donde:

$$\varepsilon_{p,RMSE} = \sqrt{\frac{1}{k}\sum_{i=0}^{k}\left[x_{(1,i)} - \hat{x}_{(1,i)}\right]^2}$$

$$\varepsilon_{v,RMSE} = \sqrt{\frac{1}{k}\sum_{i=0}^{k}\left[x_{(2,i)} - \hat{x}_{(2,i)}\right]^2}$$

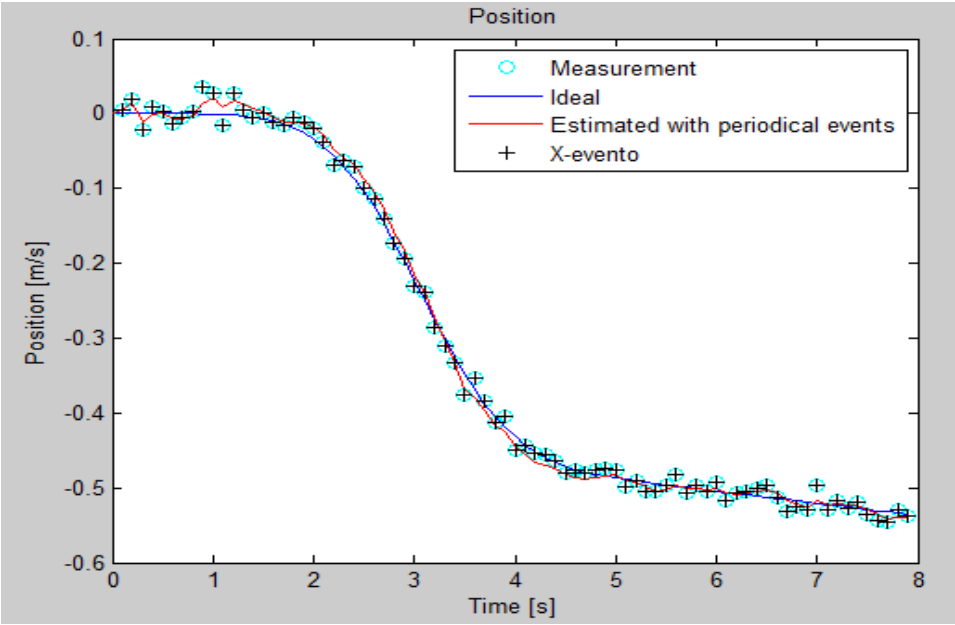

Figura 2. 7. Posición real (medida), estimada e ideal del objeto

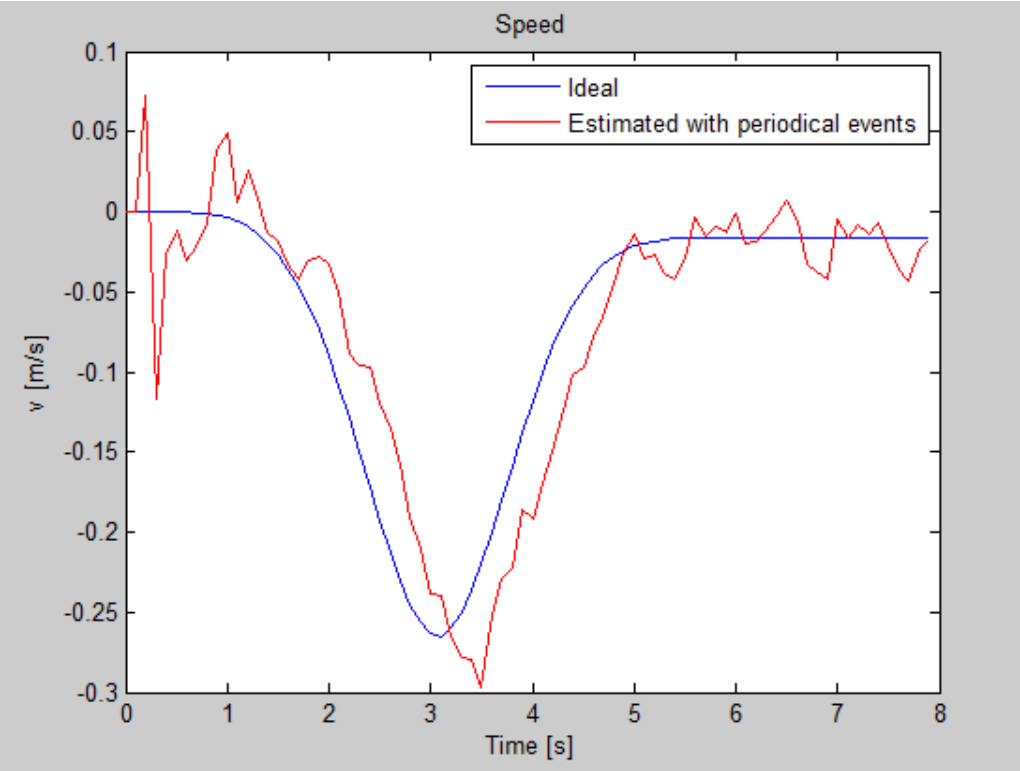

Figura 2. 8. Velocidad ideal y estimada del objeto.

Al observar la figura 2.7, se nota que hay diferencia entre la posición ideal (modelo sin ruido), la estimada con el filtro de Kalman y la medida debido a la incertidumbre del sistema y al ruido del sensor. Respecto a la velocidad (figura 2.8), su valor estimado es diferente del ideal por el ruido (incertidumbre) del sistema, téngase en cuenta que no es una variable medida. Al no tener medidas reales de la velocidad, aunque se corrija los estados cada tiempo de muestreo, su estimación acumula un error más acentuado que en posición donde si tenemos medidas reales.

El filtro de Kalman es periódico por lo que se puede calcular el número de eventos que se han producido. La duración de la simulación es T_final=8s, el tiempo de muestreo Ts=0,1s.

Número de eventos= $\frac{\text{T\_final}}{\text{Ts}} = \frac{8}{0,1} = 80$

2.1.3- Simplificación del filtro de Kalman para sistemas empotrados

Cuando se implementa el filtro de Kalman en un sistema empotrado, con memoria y capacidad de cálculo limitadas, si el sistema a estimar es invariante en el tiempo (F, G y H son constantes) y las covarianzas de los ruidos de medida y del proceso son invariantes en el tiempo (Q y R), se puede reemplazar el filtro de Kalman variante en el tiempo con el filtro de Kalman con ganancia estática (Steady-state Kalman filter). Este último tiene la ventaja de que no es necesario calcular la ganancia

de Kalman en tiempo real y tampoco las matrices de covarianza (a priori y a posteriori) del error de estimación. Se pueden determinar off-line.

Esa modificación es posible porque se demostró en {[3], (tema 5)} que la ganancia de Kalman converge en una ganancia estática tras varias muestras después del instante inicial

$$\lim_{k\to\infty} K_{(k)} = K_\infty \tag{2.11}$$

Para sistemas que tienen muchos estados, la determinación de esa ganancia estática evitaría un gran esfuerzo computacional, ya que no se calcularían matrices inversas en cada muestra k.

Como en los problemas de optimización, el mínimo de la matriz de covarianza a priori del error de estimación en régimen permanente P_∞, se obtiene a partir de la solución de la ecuación de Ricatti, y para el caso discreto (DARE= Discrete Algebraic Riccati Equation) resulta:

$$P_\infty = FP_\infty F^T - F\,(P_\infty H^T + M)\,(HP_\infty H^T + HM + M^T H^T + R)^{-1}\,(HP_\infty + M^T)\,F^T + Q \tag{2.12}$$

Con $M_{(k)} = Q_{(k)} H_{(k+1)}^T$, y asumiendo que Q>0, R>0, (definidas positivas) la ganancia estática correspondiente K_∞ es la siguiente:

$$K_\infty = (P_\infty H^T + M)(HP_\infty H^T + HM + M^T H^T + R)^{-1} \tag{2.13}$$

Y la estimación a posteriori (corregida por la nueva medida) del estado del sistema viene dada por la expresión:

$$\hat{x}_{(k)}^+ = (I - K_\infty H)F\hat{x}_{(k-1)}^+ + K_\infty y_{(k)} \tag{2.14}$$

El problema de estimación tiene una solución estable si los autovalores de $(I - K_\infty H)F$ se encuentran dentro del círculo unitario. En algunos sistemas, la ecuación de Riccati no converge a un valor estático, sino que converge a diferentes valores dependiendo de la condición inicial $P_{(0)}$.

2.2- Send-on-delta

En los sistemas de control en red, especialmente cuando los sensores aportan información a través de la red inalámbrica, la aportación aperiódica de información sensada contribuye a reducir los problemas del canal (retardos y pérdida de paquetes) así como a ahorrar energía a los módulos sensoriales. Por esta razón se utilizan técnicas de estimación no periódicas, de forma que la planta puede aplicar su solución de control

en tiempo real de forma periódica (con el vector de estado predicho), pero la corrección del estado estimado a partir de la red sensorial inalámbrica se realiza solo en determinadas condiciones e instantes de tiempo.

Esta sección describe el método de estimación aperiódica *send-on-delta*.

2.2.1-Principios del Send-on-delta

En la técnica *send-on-delta (SoD),* también llamado *level crossing o deadband,* el sensor es capaz de captar la información, procesarla y comprobar si supera o no un umbral. Como muestra la figura 2.9, en caso de superar el umbral fijado, se transmite la información al controlador y así se cierra el bucle de control. Y si no se supera el umbral no se envía nada y la planta sigue aplicando la ley de control con el estado predicho, sin ser corregido.

El umbral Δ es el valor que se utiliza como referencia. Determina la resolución de la información muestreada y se compara con la diferencia en valor absoluto de la información más reciente y la última en la que se provocó un evento.

$$\left|y_{(t_i)} - y_{(t_{i-1})}\right| = \Delta \tag{2.15}$$

El *send-on-delta* habilita el tráfico de información en el canal en función de los eventos. Sabiendo que un evento ocurre cuando se cumple la condición de la ecuación (2.15), en otras palabras cuando la señal monitorizada se desvía al menos un Δ. Como consecuencia, las señales que no varían o varían poco, obtienen una reducción significante de transmisiones por la red compartida.

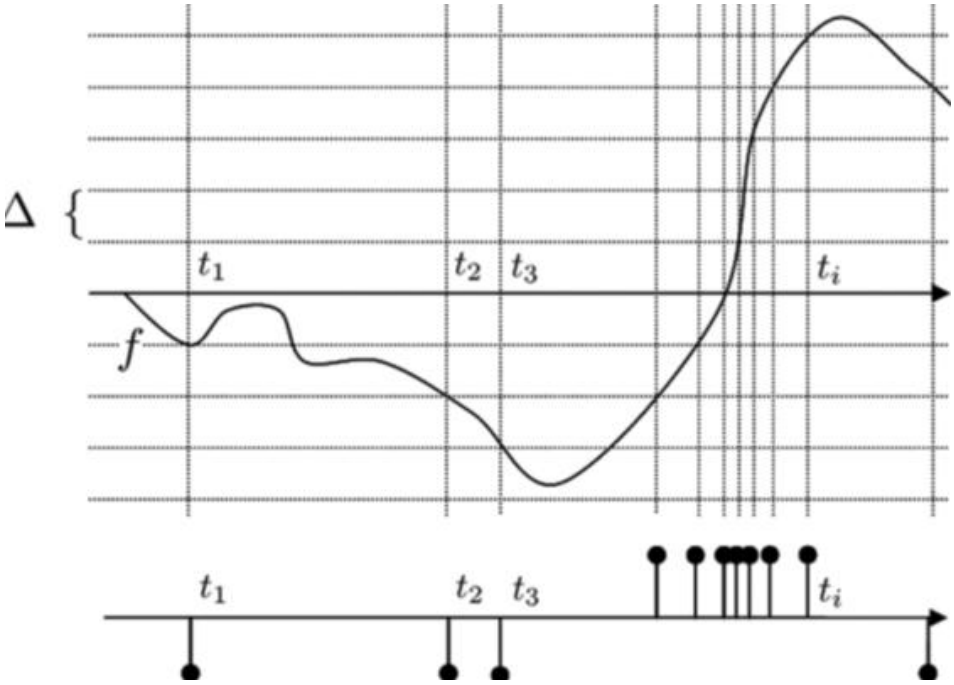

Figura 2. 9. Muestreo con el send-on-delta [16].

Aunque no se producen eventos de forma periódica, el sensor sigue tomando las medidas periódicamente con un periodo de muestreo que llamaremos T y una frecuencia de muestreo periódico $\lambda_T=1/T$. Además se puede calcular la frecuencia media λ de los eventos. La relación entre esas dos frecuencias es la efectividad p, y permite comparar la respuesta de *send-on-delta* para diferentes tipos de entradas y sistemas.

$$p = \frac{\lambda_T}{\lambda} \tag{2.17}$$

Se puede comprobar en [6] que, frente a una misma entrada, la efectividad es diferente en cada sistema. En la técnica de *send-on-delta*, cuando hay un cambio en la referencia, se producen los eventos hasta llegar a un estado de equilibrio y los intervalos de tiempo entre esos eventos dependen de la dinámica del sistema.

Aplicación de SoD al ejemplo del sistema de seguimiento de un objeto.

Tomando como referencia el ejemplo anterior de seguimiento de trayectoria en línea recta, se aplica la técnica SoD con umbral de 0.1m.

En el apéndice se incluye el fichero script de Matlab (SOD_1D.m) correspondiente a este caso de estudio, cuyos resultados se muestran a continuación. Las variables de la leyenda representan:

- Measurement: son los valores de la posición medidos por el sensor, con las mismas características de ruido $R = 0.1e^{-3}$.
- Real: son los valores reales que tiene la posición del móvil.
- Estimated: son los valores estimados de la posición aplicando el Filtro de Kalman con correcciones aperiódicas cuando el mecanismo de disparo SoD lo permite.
- X-evento: son los disparos generados con SoD.

En la representación de la figura 2.10, hay cruces donde se ha producido un evento y esos eventos se producen en una zona de variación de la posición, en este ejemplo los disparos se concentran en el intervalo de tiempo entre 2s y 5s. Cuando no hay una gran variación en la señal (a partir de 5s) no se producen eventos y la posición estimada va siendo cada vez más diferente de la posición real. Además se ve en este caso que el estado estimado no está siempre próximo al estado real como se vea en la figura 2.7 con el filtro de Kalman periódico, sino que va creciendo hasta que se produzca un evento y en ese momento se aproxima al estado real mediante la corrección y vuelve a desviarse hasta el siguiente disparo.

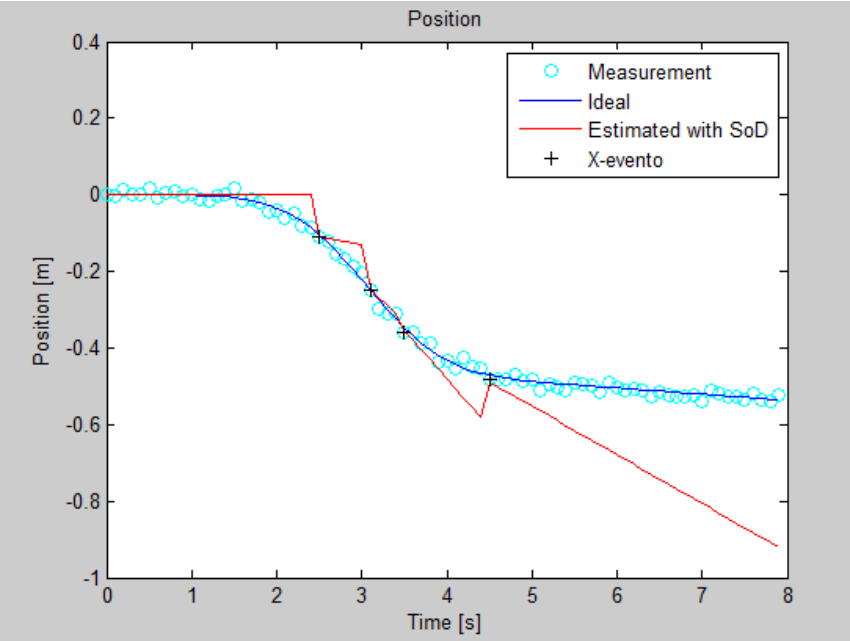

Figura 2. 9. Representación de las posiciones y de los errores aplicando el send-on-delta.

Ahora se va a representar la gráfica de la velocidad, para observar cómo es su evolución respecto a la figura 2.8, donde:

- Ideal representa los valores reales de la velocidad.
- Estimated representa los valores estimados de la velocidad.

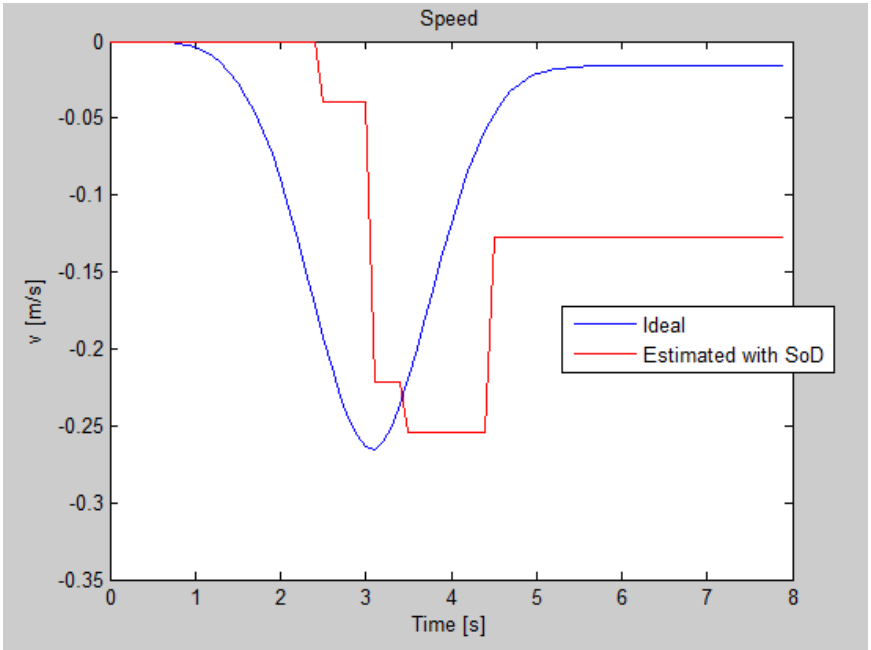

Figura 2. 10. Representación de la velocidad real estimada, con sus errores aplicando SOD.

Respecto a la gráfica de la velocidad aplicando el filtro de Kalman, seguimos apreciando un desfase entre los valores estimados y los valores reales. Además el área entre las dos curvas es más grande (figura 2.12) por lo que podemos deducir que el error será mayor.

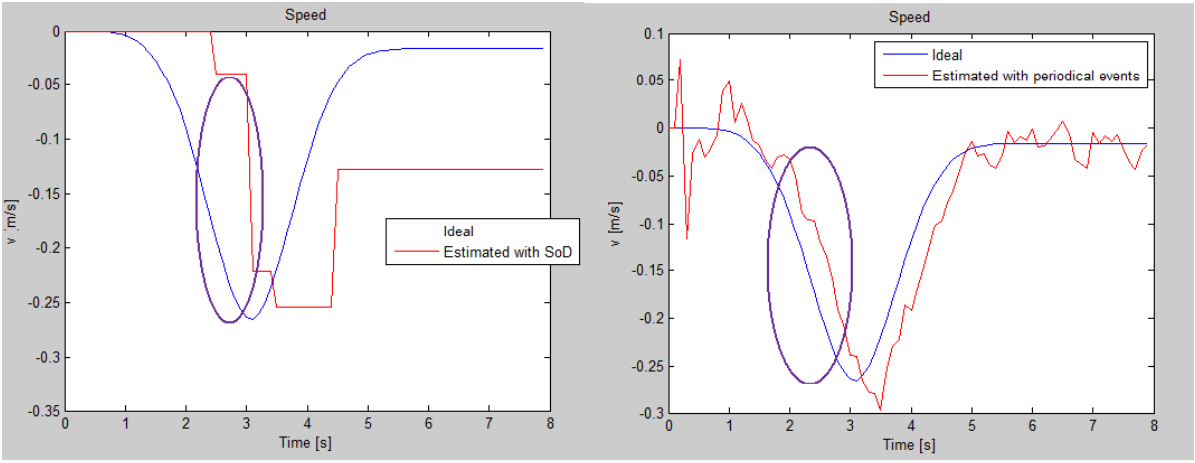

Figura 2. 11. Comparación de la diferencia entre el área entre la curva de estimación con SoD y la estimación periódica de la velocidad.

Los valores eficaces de los errores de posición y de velocidad se muestran en la tabla siguiente.

Tabla 2. Valores eficaces de los errores de estimación en posición y en velocidad usando el SoD y KF periódico

Técnica de estimación	$\varepsilon_{p,RMSE}$	$\varepsilon_{v,RMSE}$	Número de disparos
SoD	0,1873 m	0,1152m/s	4
KF períodico	0,00686 m	0,03729m/s	80

Comparando las gráficas anteriores, se confirma el hecho de que el *send-on-delta* reduce la frecuencia de corrección de los estados y por lo tanto las transmisiones por el canal de comunicación, pero hay inconvenientes. El principal es que cuando la señal no varía, no se producen eventos y la estimación de los estados se desfasa mucho de los estados reales, es decir se incrementa el error de estimación de estados. El otro problema es que el send-on-delta necesita dispositivos con protocolos de gestión que permitan a cada nodo habilitar la transmisión cuando proceda.

2.3- Send-on-delta integrado

El Send-on-delta integrado es un método de estimación aperiódica, pero se diferencia del SoD en la implementación del criterio que habilita la comunicación entre la red sensorial inalámbrica y el controlador. Esta sección describe los principios de este método.

2.3.1-Principios del Send-on-delta integrado

Esa técnica deriva de la anterior y es conocida también como *send-on-area* (SOA), o *send-on-delta integrado*. Tal y como muestra la figura 2.13, consiste en integrar en un intervalo de tiempo la diferencia en valor absoluto de la salida actual $y_{(t)}$ y de la última salida $y_{(t_{i-1})}$ en la que se facilitó datos al estimador para corregir los estados. Se produce un evento cuando el resultado de la operación anterior supera un umbral $\mu > 0$, umbral diferente al de send-on-delta Δ ($\mu \neq \Delta$).

$$\int_{t_{i-1}}^{t_i} \left| y_{(t)} - y_{(t_{i-1})} \right| dt = \mu \qquad (2.18)$$

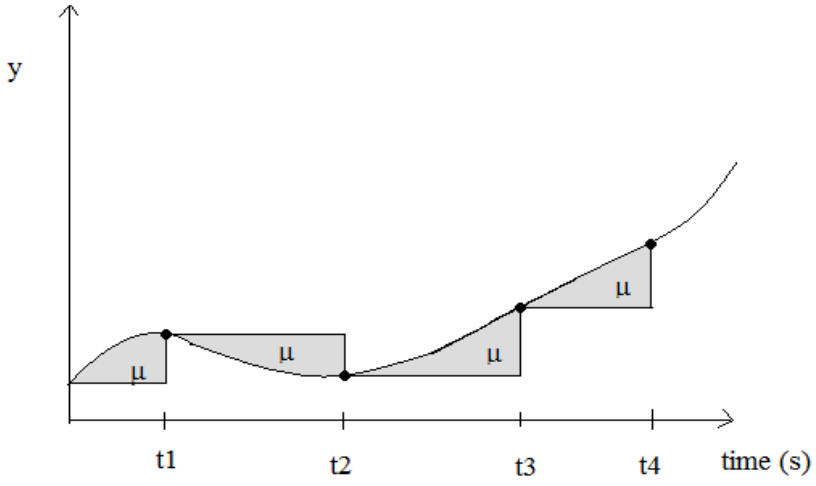

Figura 2. 12. Generador de eventos con send-on-delta integrado.

En la figura 2.13 se muestra que los disparos se generan en los momentos en que el área sombreada alcanza el valor prefijado. Esta técnica permite, al igual que en el send-on-delta, reducir la información que el módulo sensorial envía al elemento de control a través de la red que comparten. Esta técnica evita que la diferencia entre la señal sensada y la del último disparo se prolongue en el tiempo.

Aplicación de SOA al ejemplo de seguimiento de un objeto.

En este apartado se aplica la técnica de SoA al seguimiento de trayectoria en una dimensión con Delta=0.001m (*SODI_1D*)

Tomando como referencia el ejemplo inicial de seguimiento de un objeto en línea recta con aceleración variable en el tiempo, se aplica ahora la estimación de posición y velocidad basada en SoA.

En las siguientes figuras se utiliza la misma leyenda que en el ejemplo de SoD.

Calculando el valor medio del error de estimación de las dos componentes del vector de estado y comparando con las técnicas anteriores se tiene la tabla 3.

Tabla 3. Valores eficaces de los errores de estimación en posición y en velocidad usando el SoA

Técnica de estimación	$\varepsilon_{p,RMSE}$	$\varepsilon_{v,RMSE}$	Número de disparos
SoA	0,02112 m	0,05779m/s	11
SoD	0,1873 m	0,1152m/s	4
KF períodico	0,00686 m	0,03729m/s	80

En la *figura 2.14* se observa que se siguen produciendo eventos cuando se ha estabilizado el estado, porque el área va creciendo hasta alcanzar el umbral predefinido, mejorando así la estimación derivada del send-on-delta. Al tener una condición más estricta, el send-on-delta integrado va generar más eventos que el send-on-delta. Sin embargo, el error de posición con SoA es aproximadamente un 10% y el de velocidad un 40% con respecto al SoD.

Las conclusiones de este capítulo son:

- Las técnicas de estimación basadas en eventos, de interés en sistemas de control en red, reducen el número de comunicaciones entre el elemento sensor y el controlador frente a la solución clásica de estimación periódica.
- Independientemente del mecanismo elegido para aportar nueva información al estimador de estados, el filtro de Kalman es el estimador de referencia, aunque la fase de corrección se ejecute de forma aperiódica.
- El mecanismo de disparo send-on-delta implementado en el módulo sensorial, presenta una eficiencia fuertemente ligada a la dinámica del sistema, reduce el número de accesos al canal frente al caso periódico pero con un mayor error de estimación de estados.

- La técnica send-on-delta integrada cubre el sesgo asociado al send-on-delta cuando la medida no cambia con el tiempo. Generalmente proporciona mayor número de accesos al canal que SoD pero con menor error de estimación de estados resultante.

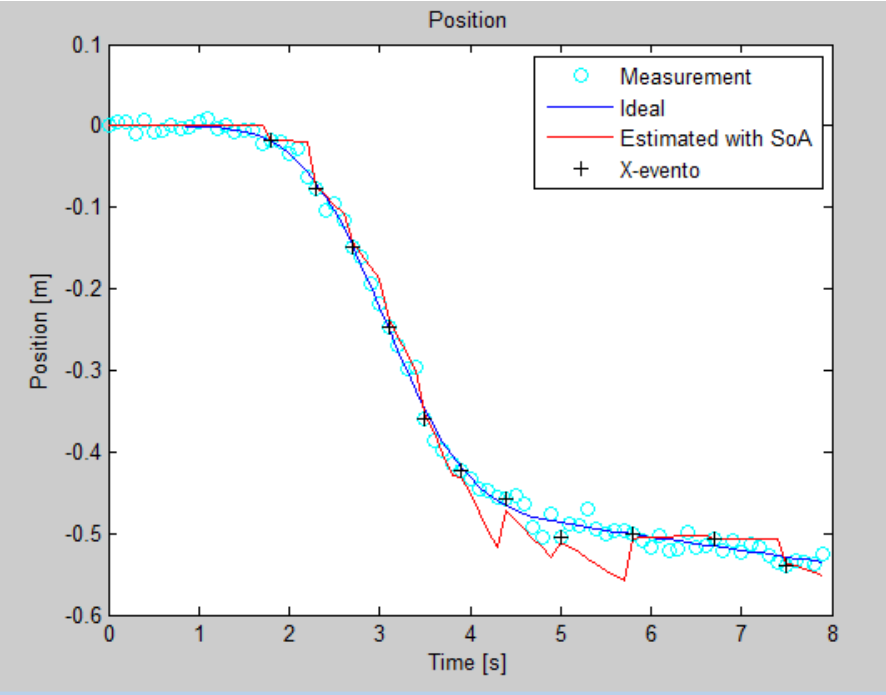

Figura 2. 13. Representación de las posiciones y de los errores aplicando el send-on-delta integrado.

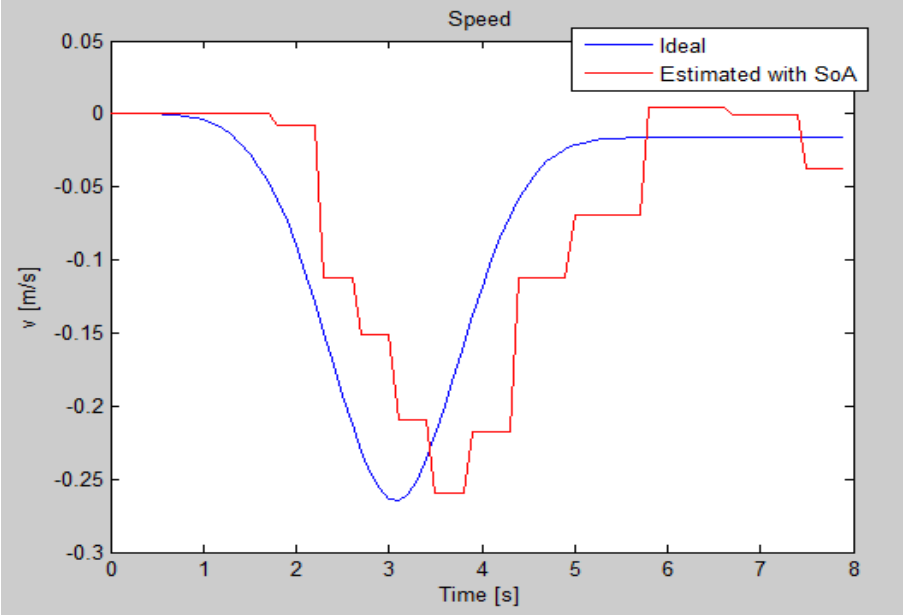

Figura 2. 14. Representación de la velocidad real y estimada, con sus errores aplicando el send-on-delta integrado.

3- Aplicación de técnicas de estimación basada en eventos al robot P3-DX

En este capítulo se aplican las técnicas de estimación basada en eventos descritas en el capítulo 2, al robot Pioneer 3-DX. Para ello se utiliza el modelo ha obtenido en trabajos previos [14] y [15].

3.1-Modelo robot P3-DX

Para modelar la planta, el procedimiento utilizado consiste en aplicar un vector de entrada cambiante en el tiempo y registrar el vector de salida. Se plantean una estructura paramétrica y se determinan los parámetros del modelo aplicando la técnica de identificación de mínimos cuadrados. El modelo del robot P3-DX utilizado es el obtenido en el trabajo [14]. Donde el vector de entrada está formado por las velocidades lineal y angular de actuación, el vector de salida por las velocidades lineal y angular de respuesta teniendo en cuenta la dinámica del robot y presenta cuatro estados internos: dos coinciden con las salidas y dos con las entradas retardadas. A continuación se indica el modelo de partida del robot P3-DX. En tiempo continuo resulta:

$$\dot{x}_{(t)} = A\,x_{(t)} + B\,u_{(t)} + w_{(t)}$$

$$y_{(t)} = C\,x_{(t)} + v_{(t)} \qquad (3.1)$$

$$A=\begin{bmatrix} -4{,}084 & -0{,}015 & 1664 & 0{,}7227 \\ -0{,}008 & -5{,}042 & 0{,}326 & 2023 \\ 0 & 0 & -200 & 0 \\ 0 & 0 & 0 & -200 \end{bmatrix};\quad B=\begin{bmatrix} -4{,}159 & -0{,}002 \\ -0{,}001 & -5{,}057 \\ 1 & 0 \\ 0 & 1 \end{bmatrix}$$

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

Este modelo se utilizará para evaluar la salida del sistema en tiempo continuo y decidir cuándo se han de generar los disparos o eventos; momentos en los que el sensor aporta información para corregir el estado predicho.

Discretizando este modelo para un periodo de muestro de 10ms, con la función *c2dm* de Matlab, resulta:

```
[F,G,H,D]=c2dm(A,B,C,D,0.01,'zoh)
```

$$F=\begin{bmatrix} 0{,}9599 & -0{,}0001 & 7{,}0017 & 0{,}0022 \\ -0{,}0001 & 0{,}9508 & 0{,}0010 & 8{,}4611 \\ 0 & 0 & 0{,}1353 & 0 \\ 0 & 0 & 0 & 0{,}1353 \end{bmatrix} \quad G=\begin{bmatrix} 0{,}0057 & 2{,}9742e^{-6} \\ 1{,}3429e^{-6} & 0{,}0070 \\ 0{,}0043 & 0 \\ 0 & 0{,}0043 \end{bmatrix}$$

$$H=\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix} \qquad D=\begin{bmatrix} 0 & 0 \\ 0 & 0 \end{bmatrix}$$

Para el sistema discretizado que responde al modelo:

$$x_{(k)} = F\, x_{(k-1)} + G\, u_{(k-1)} + w_{(k-1)}$$

$$y_{(k)} = H\, x_{(k)} + v_{(k)} \tag{3.2}$$

siendo $x \in R^4$, $u \in R^2$, $y \in R^2$.

Este modelo se utilizará para obtener los resultados de estimación periódica a partir del filtro de Kalman convencional.

3.2- Condiciones de análisis

El objetivo es comparar las tres técnicas de estimación ya comentadas (filtro de Kalman periódico, send-on-delta y el send-on-delta integrado) aplicadas a la estimación de estados de un robot P3-DX.

Para ello se plantea un esquema de trabajo como el indicado en la figura 3.1. La salida es continuamente evaluada. El generador de eventos decide, en función de la técnica aplicada, cuándo se incorpora la información del "sensor" al algoritmo de estimación para la corrección (estado a posteriori) del estado predicho (estado a priori).

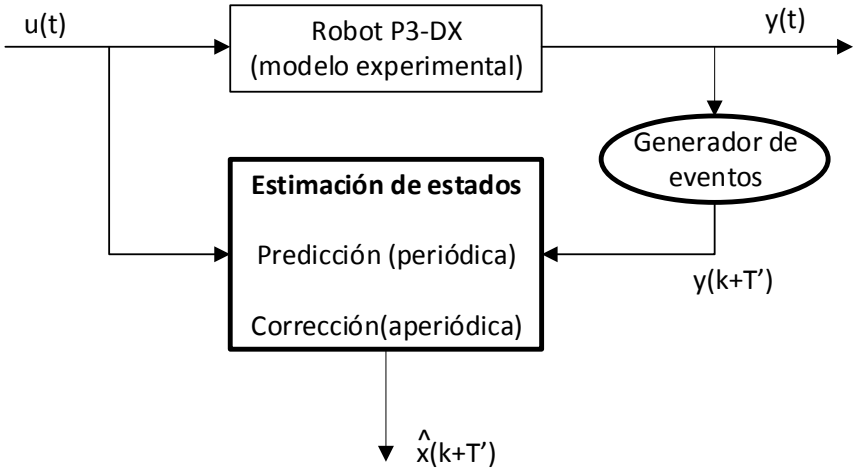

Figura 3. 1. Diagrama de bloques del estimador de estados basado en eventos.

En el caso real de un sistema de control en red, como el mostrado en la figura 1.5, el estimador forma parte del algoritmo de control implementado en el elemento controlador que actúa de forma remota sobre la planta. En otro tipo de aplicación, el sensor puede estar ubicado en la infraestructura (entorno) y los algoritmos de estimación y control se ejecutan en el propio robot conectado de forma remota con el sistema sensorial.

El estimador de estados está basado en el filtro de Kalman, periódico en un caso y aperiódico en otros (SoD y SoA). En todos ellos se ha considerado que el valor de la ganancia estática de Kalman (Kss o L_inf) es la misma y coincide con la evaluada en el caso periódico. Para obtener la ganancia estática es necesario disponer los siguientes parámetros: F, H, R y Q.

Para calcular dicha ganancia se ha recurrido a la función *dare* proporcionada por Matlab. DARE es el acrónimo de la ecuación algebraica discreta de Riccati de la que se habló en el apartado 2.1.1 del segundo tema.

Esta función proporciona la matriz de covarianza de error de estimación a priori (P_inf), y con ella se obtiene la ganancia de Kalman (L_inf), como se indica a continuación:

$$F\_inf = F^{l} \text{ y } Q\_inf = \sum_{i=0}^{l-1} F^{i} Q F^{i\ T}$$

$$[P\_inf, Y, G] = DARE\ (F\_inf^{\ T}, H^{T}, Q\_inf, R)$$

$$L\_inf = P\_inf * H^{T} (H * P\_inf * H^{T} + R)^{-1} \qquad (3.3)$$

Siendo l la relación entre Ts (el periodo de predicción) y T_m (el periodo de corección). Por ejemplo: Ts=0,01s y se envían las medidas al estimador cada T_m=1s, l= $\frac{Tm}{Ts}$ = 10.

En una primera aproximación, se observó que usando un umbral para la velocidad lineal y otro para la velocidad angular, se seguía produciendo otro inconveniente para poder comparar los datos obtenidos del análisis, ya que ajustar una de las condiciones afectaba a la otra, por lo que se optó por combinar las dos condiciones en una usando la siguiente expresión para el umbral de disparo:

$$E_y^{\ T}\, S\, E_y = \Delta \qquad (3.4)$$

Donde Ey= $\begin{bmatrix} E_{y1} \\ E_{y2} \end{bmatrix}$ y S = $\begin{bmatrix} k_1 & 0 \\ 0 & k_2 \end{bmatrix}$ con Ey_1 el error de velocidad lineal, Ey_2 el error en velocidad angular, k_1 y k_2 los coeficientes de ponderación de Ey_1 e Ey_2 espectivamente.

Si $k_1 = k_2 = 1 \rightarrow$ S= I la matriz identidad.

Por tanto, el punto de partida para los diferentes análisis que se realizan a continuación es: a) determinar la ganancia estática del filtro de Kalman, b) determinar los errores admitidos de salida (velocidades), c) calcular los umbrales de la técnica de estimación basada en eventos bajo estudio. Con ello se obtiene el número de eventos generado en cada caso.

- **Modulación del umbral de disparo**

Como se ha explicado en el segundo tema de este documento, el umbral de disparo define la resolución de la frecuencia de disparos, razón por la cual es importante que se elija de forma adecuada y dependiente de la aplicación que se está controlando. A partir de la ecuación (3.3), se obtiene que $\Delta=k_1 Ey_1^2 + k_2 Ey_2^2$ {Ey_1 en m/s; Ey_2 en rad/s; k_1 en $(m/s)^{-2}$; k_2 en $(rad/s)^{-2}$}. Resultando así un valor de umbral Δ adimensional.

Si se elige un umbral demasiado pequeño los eventos se producirían debido al ruido del sistema o del instrumento de medida, mientras que si ese umbral es demasiado grande los disparos se verían reducidos, aunque a costa de cometer un gran error entre los estados estimados.

Con los coeficientes de ponderación k_1 y k_2 se puede configurar cuál de las dos componentes de error tiene más peso en nuestra aplicación. Por ejemplo, si se quiere que el error en velocidad lineal sea el que domine en la generación de eventos, el coeficiente k_1 debe ser inferior a k_2 y viceversa.

- **Influencia del ruido del sensor**

Los equipos electrónicos de medida, por muy precisos que sean, suelen tener un pequeño ruido causado por:

– los componentes que lo forman;
– la influencia de entorno en el que trabajan;
– ambos aspectos.

El espacio de trabajo puede ser hostil debido a las radiaciones naturales o por las interferencias generadas por los equipos presentes en él.

La elección del sensor ha de estar condicionada por el nivel de incertidumbre de medida admitido en la aplicación.

3.3- Resultados

A continuación se indican los términos que aparecen en la leyenda de las diferentes figuras:

Measurement representa la velocidad lineal/angular medida por el sensor.

Real es la velocidad lineal/angular real, sin ruido de medida.

Estimated es la velocidad lineal/angular estimada.

Vdisparado representa los valores de la velocidad lineal alcanzada por el robot cuando se ha producido un evento.

Wdisparado representa los valores de la velocidad angular para los que se ha producido un evento.

$\boldsymbol{\sigma_v}$ y $\boldsymbol{\sigma_w}$ representan respectivamente la desviación típica del error de velocidad lineal y del error de velocidad angular. Los valores $2\sigma_v$ y $2\sigma_w$ se utilizan para evaluar si la propagación del error de estimación no sobrepasa estos límites.

Measurement error es el error de los valores medidos de la velocidad lineal/angular.

Estimation error es el error de los valores estimados de la velocidad lineal/angular.

El periodo básico de muestreo es Ts=0,01s, si bien se han realizado pruebas con otros valores: 10Ts, 25Ts y 40Ts.

El procedimiento seguido en las simulaciones es el siguiente:

- Definición de las condiciones de simulación.
  Se fijan los parámetros de simulación: tiempo de ejecución *T_final*, tiempo de muestreo *Ts*,
  Se indican las matrices características del modelo continuo *A, B, C*.
  Se determinan las matrices para la aplicación discreta del filtro de Kalman: $F= e^{A*Ts}$ y $G= A*[(F-I)*B]^{-1}$, *H=C.*
  Se definen los errores del sistema y de medida del sensor.
  Se fijan las condiciones iniciales de los estados (x_0, x_hat_0) y de la matriz de covarianza *P_0*. Y se calculan los valores estáticos: *P_inf* y *L_inf.*
  Se establecen las referencias de la velocidad lineal y de la velocidad angular en la variable ref.

- Determinación de la salida del sistema, a partir del modelo en tiempo continúo. Esta parte se ejecuta en un bucle desde *0* hasta *T_final*). Se determina los estados "reales" *x(k)* que permiten obtener las salidas *y(k) "*medidas" por el sensor de las velocidades lineal y angular.

- Cálculo el error de medida, en el mismo bucle anterior. Se ha definido una variable *i* donde se guarda los instantes de tiempo k donde se ha producido el último evento según la técnica considerada, inicialmente *i=1*. Esa variable i nos permite calcular el error entre el valor de la salida actual *y(k)* y el valor del último evento *y(i).*
  En el caso SoD el error se calcula con *rest(k)=|y(k)-y(i)|* y el valor que se compara con el umbral es $diferencia(k)= (rest(k))^T \cdot S \cdot rest(k)$. Como se ha explicado en el apartado 3.2.1.
  Para el caso SoA, se calcula el área entre *y(k)* e *y(i)*. Para ello se ha definido una variable $m(k)= (rest(k))^T \cdot S \cdot rest(k)$, sabiendo que *rest(k)=|y(k)-y(i)|* y con la función *trapz* de Matlab se obtiene el área que se asigna a la variable *integral(k).* Esa última variable es la que se compara con el umbral.

- Generación de eventos. En esta etapa se compara si la variación experimentada por la medida actual con respecto a la medida anterior es superior al umbral previsto. Si se cumple la condición de disparo se genera un evento y se hace la corrección de los estados estimados *x_hat(k)=x_pos(k),* la matriz de covarianza a posteriori *P_pos(k)* y se actualizan la entrada del servosistema *n(k)* y de la planta *u(k)* con los nuevos valores de *x_hat(k)*. Si no se cumple la condición de disparo *n(k)* y *u(k)* se calculan con los valores estimados en la fase de predicción: *x_hat(k)=x_pri(k)*.

- Cálculo de errores medios (RMS). Se calculan cuando se disponen ya de todos los valores de estados "reales" y estimados, una vez alcanzado *T_final*.

- Presentación de resultados en gráficas.

El diagrama siguiente representa el bucle de ejecución, por una parte de la medida periódica registrada por el sistema sensorial, y por otra el cálculo de la ley de control bien a partir del estado predicho (x_pri) o bien a partir del estado corregido (x_pos), entre 0 y *T_final*. La información aportada por el sistema sensorial remoto solo se lleva a cabo cuando se genera el correspondiente evento.

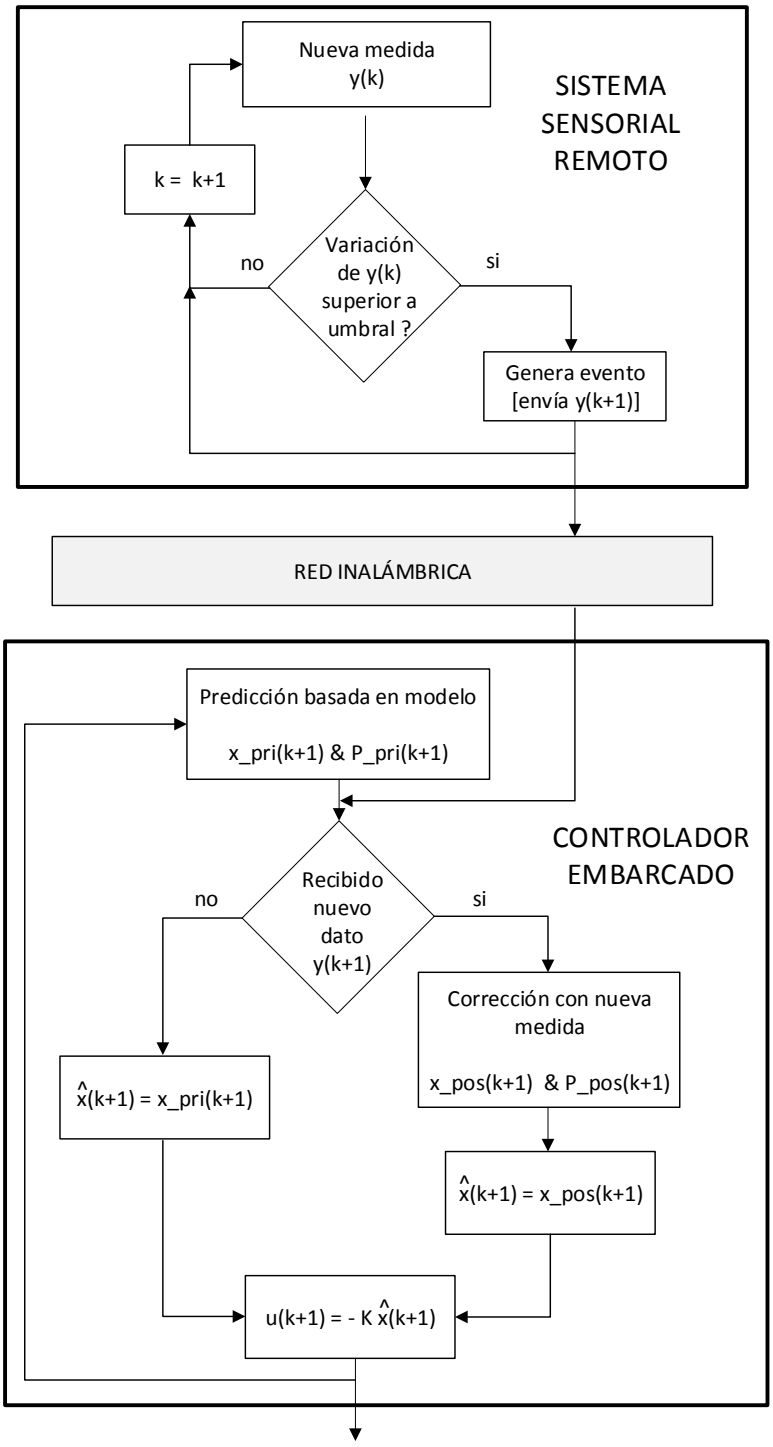

Figura 3. 2. Diagrama de bloque con la estructura de simulación.

Simulación 1: El objetivo es analizar el número de disparos necesarios, con las técnicas de estimación basada en eventos, para conseguir el mismo error de estimación de velocidad angular que aplicando el FK periódico.

Se fija ese periodo de disparo multiplicando Ts=10ms, por una constante N que puede valer 10, 25 o 40. Cada vez que se define un nuevo valor de N, se recalculan los valores de P_inf y L_inf.

Los elementos comunes a los tres casos (FK, SoD y SoA) son Ts, L, P_inf, L_inf, las condiciones iniciales x_0 y x_hat_0 y las referencias.

Cuando está ya hecho la simulación del FK y ya tenemos el error ω_{rms}, la siguiente simulación es la del SOD, seguida del SOA, donde se empieza siempre con un umbral mínimo y se va aumentando hasta conseguir el ω_{rms} anterior. Ese procedimiento necesita mucha paciencia y atención.

Donde $\omega_{rms} = \sqrt{\frac{1}{k}\sum_{i=0}^{k}\left[x_{(2,i)} - \hat{x}_{(2,i)}\right]^2}$ y $V_{rms} = \sqrt{\frac{1}{k}\sum_{i=0}^{k}\left[x_{(1,i)} - \hat{x}_{(1,i)}\right]^2}$ y

$k = \frac{T\_final}{Ts}$

Para un tiempo de muestreo de 10Ts

Técnica	Tm	Umbral (Δ)	V_{rms}	ω_{rms}	Nº disparos
FK	10Ts	-	0,0063	0,0141	999
SoD	variable	0,0015	0,0048	0,0141	497
SoA	variable	7e-5	0,0047	0,0141	532

Tabla 3.1: Resultados usando la ganancia estática y manteniendo un error de velocidad angular igual con un tiempo de muestreo de 10 Ts.

Analizando los resultados con 10Ts se comprueba que para el mismo error de estimación de la variable no medida (velocidad angular) el número de disparos requerido se reduce a la mitad. La diferencia no es relevante porque el tiempo entre muestreo es pequeño.

A continuación se muestra la evolución temporal de las diferentes señales registradas en simulación.

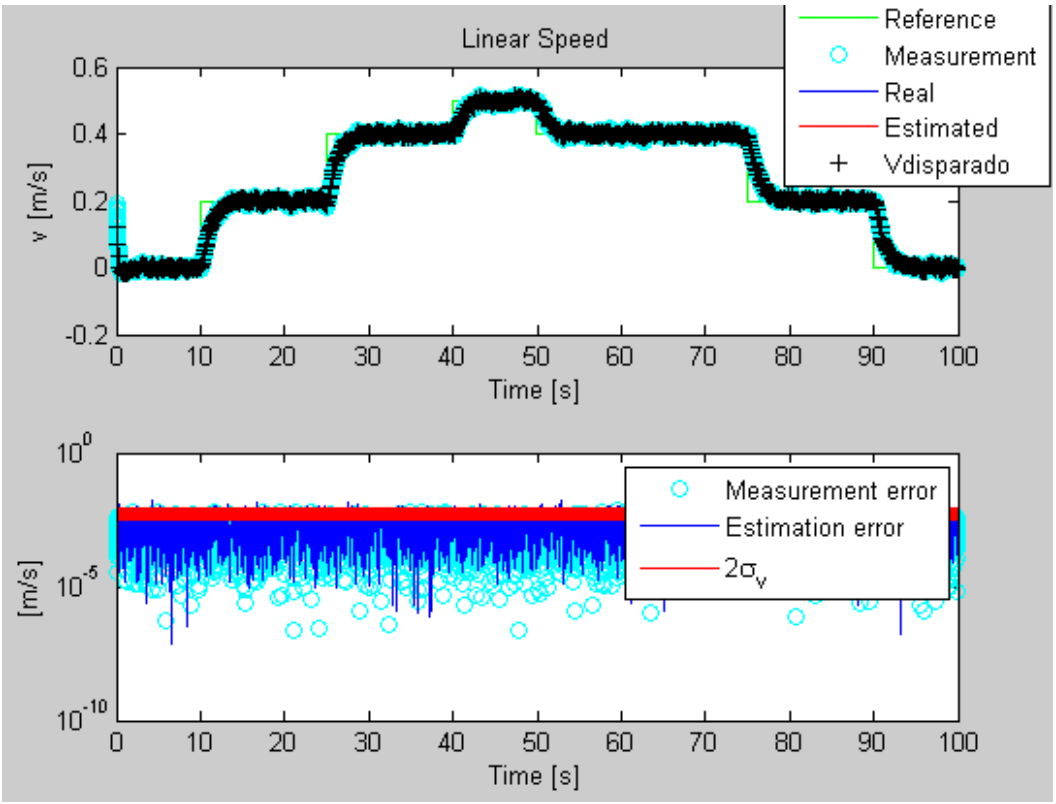

Figura 3. 3. Representación de la velocidad lineal y errores para el FK periódico con 10Ts.

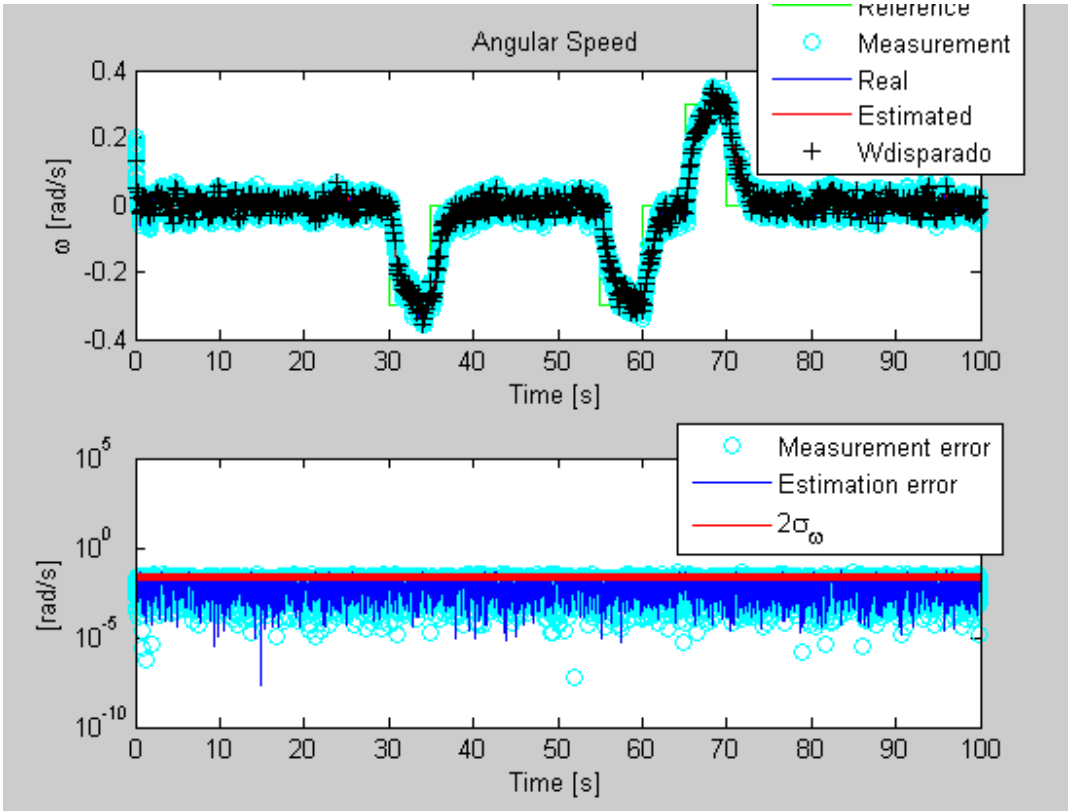

Figura 3. 4. Representación de la velocidad angular y sus errores para el FK a 10Ts.

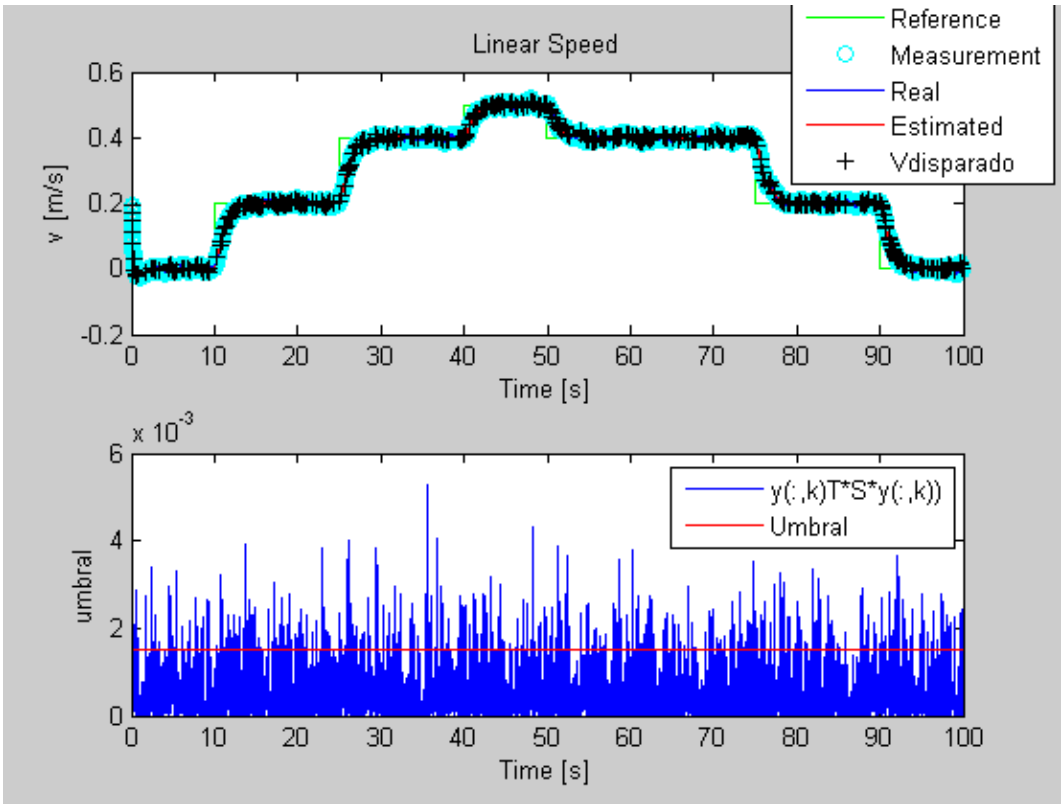

Figura 3. 5. Representación de la velocidad lineal y sus errores para el SoD a 10Ts.

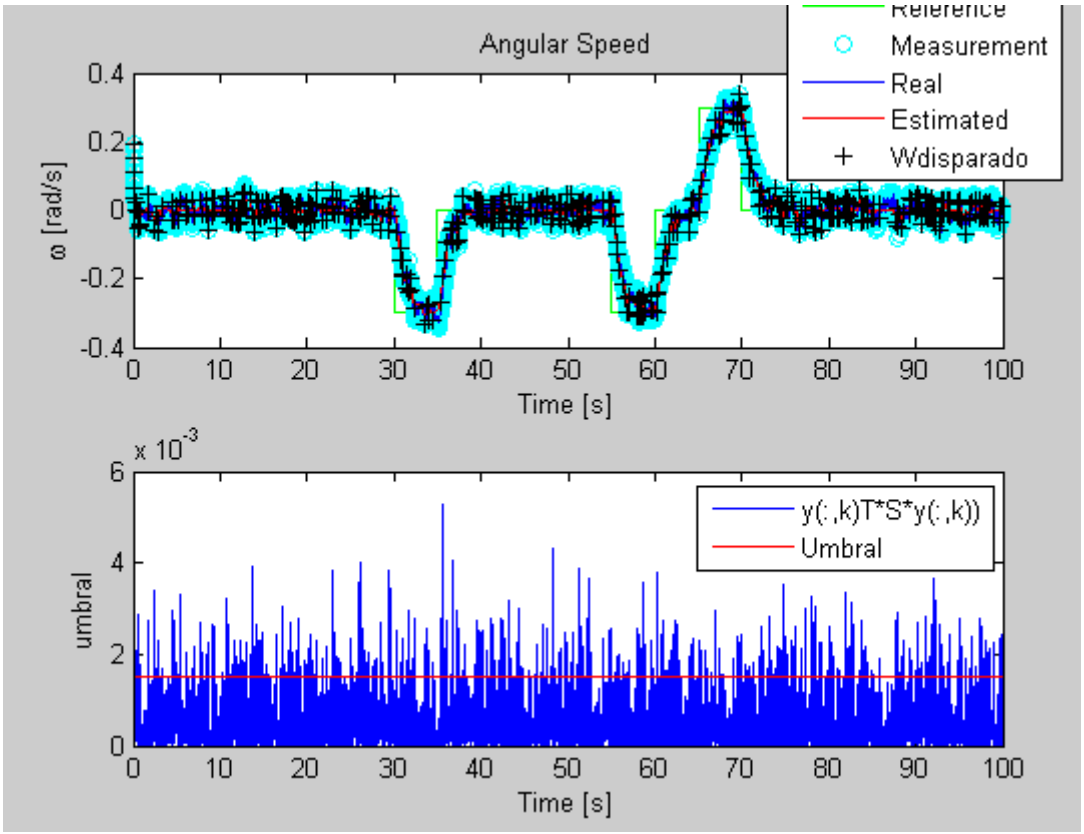

Figura 3. 6. Representación de la velocidad angular y sus errores para el SoD a 10Ts.

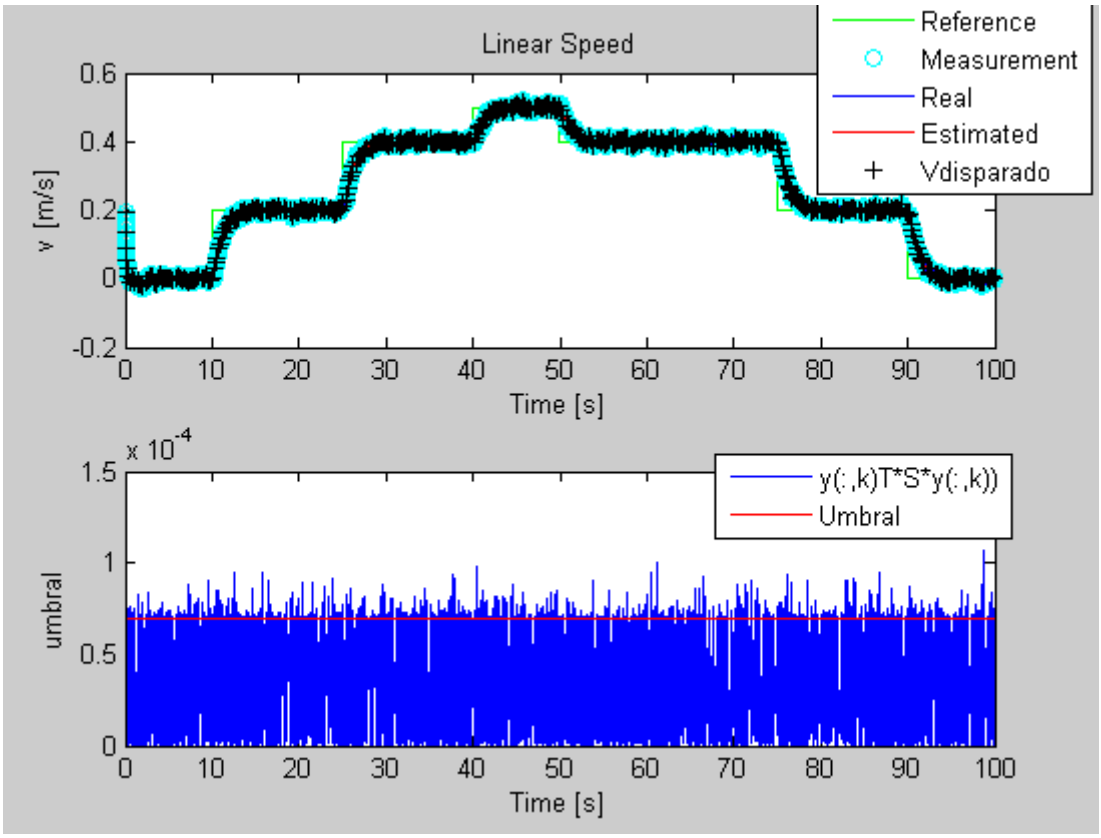

Figura 3. 7. Representación de la velocidad lineal y sus errores para el SoA a 10Ts.

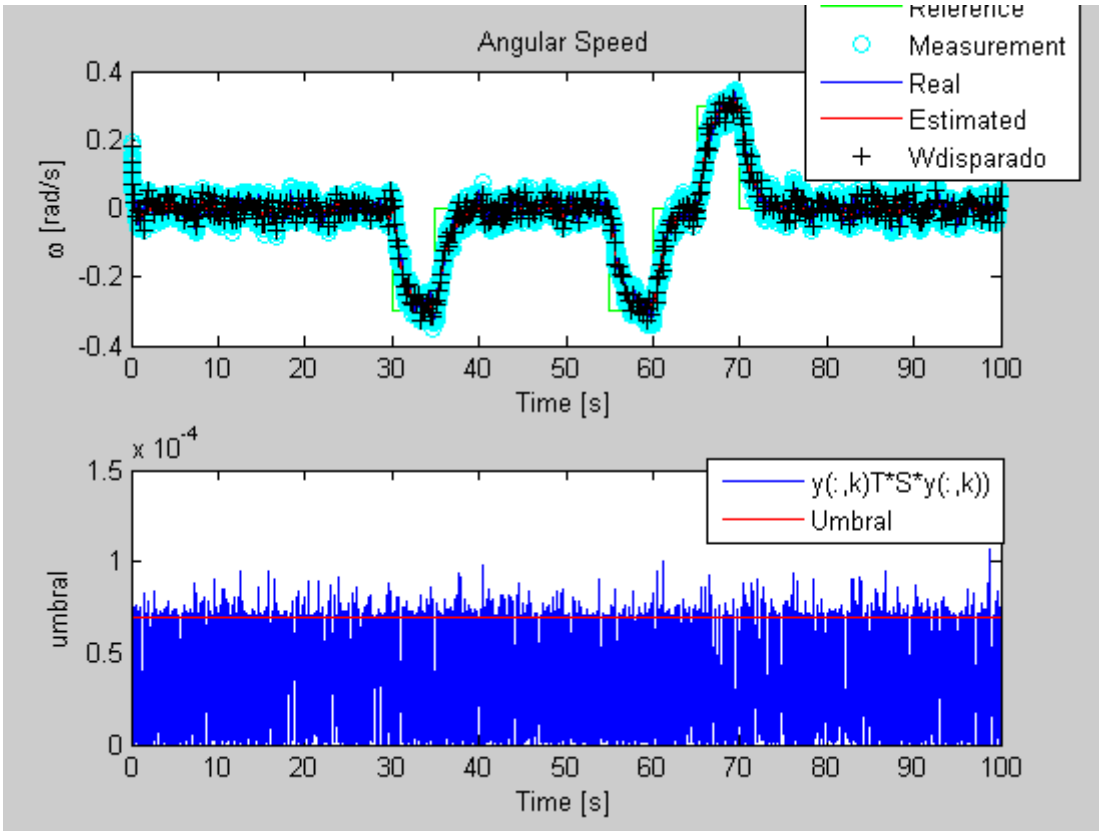

Figura 3. 8. Representación de la velocidad angular y sus errores para el SoA a 10Ts.

Se repite la simulación con un periodo de muestreo 25Ts, obteniéndose la siguiente tabla de resultados.

Técnica	Tm	Umbral (Δ)	V_{rms}	ω_{rms}	Nº disparos
FK	25Ts	-	0,0088	0,0174	399
SoD	variable	0,004	0,0056	0,0174	130
SoA	variable	6e-4	0,0054	0,0174	132

Tabla 3.2: Resultados usando la ganancia estática y manteniendo un error de velocidad angular igual con un tiempo de muestreo de 25 Ts.

Con un periodo de muestreo de 25Ts, se aprecia que los disparos se producen por la dinámica del sistema y, en menor medida, no por los ruidos. Para un mismo valor medio del error de estimación, con la técnica SoD el número de disparos se reduce al 22% y con SoA al 46%

A continuación se muestran la evolución temporal de las diferentes señales registradas en simulación.

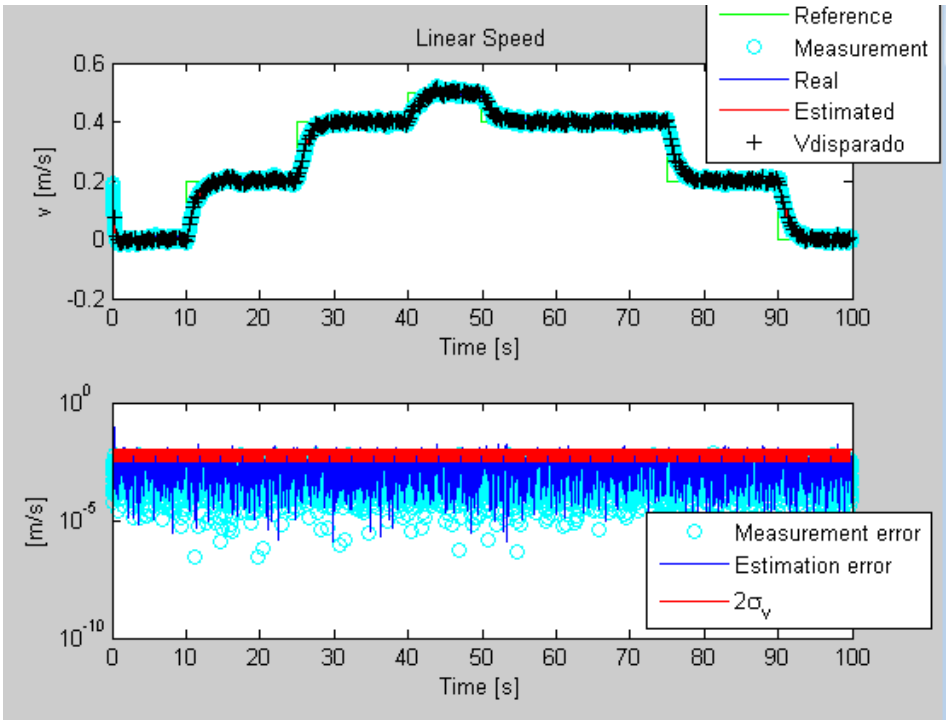

Figura 3. 9. Representación de la velocidad lineal y sus errores para el FK a 25Ts.

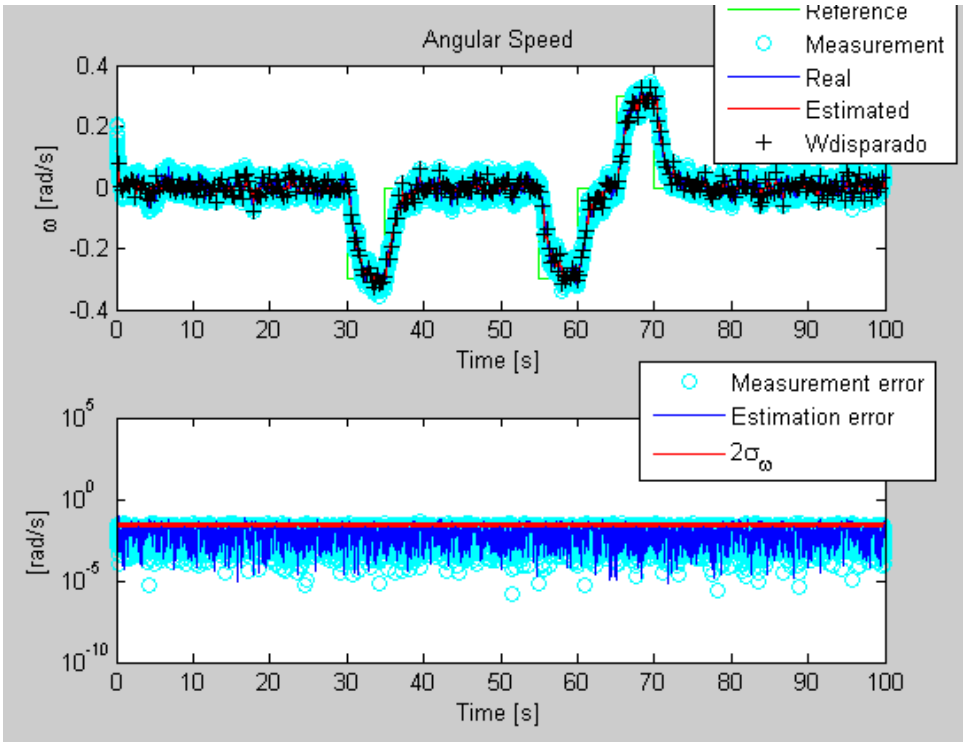

Figura 3. 10. Representación de la velocidad angular y sus errores para el FK a 25Ts.

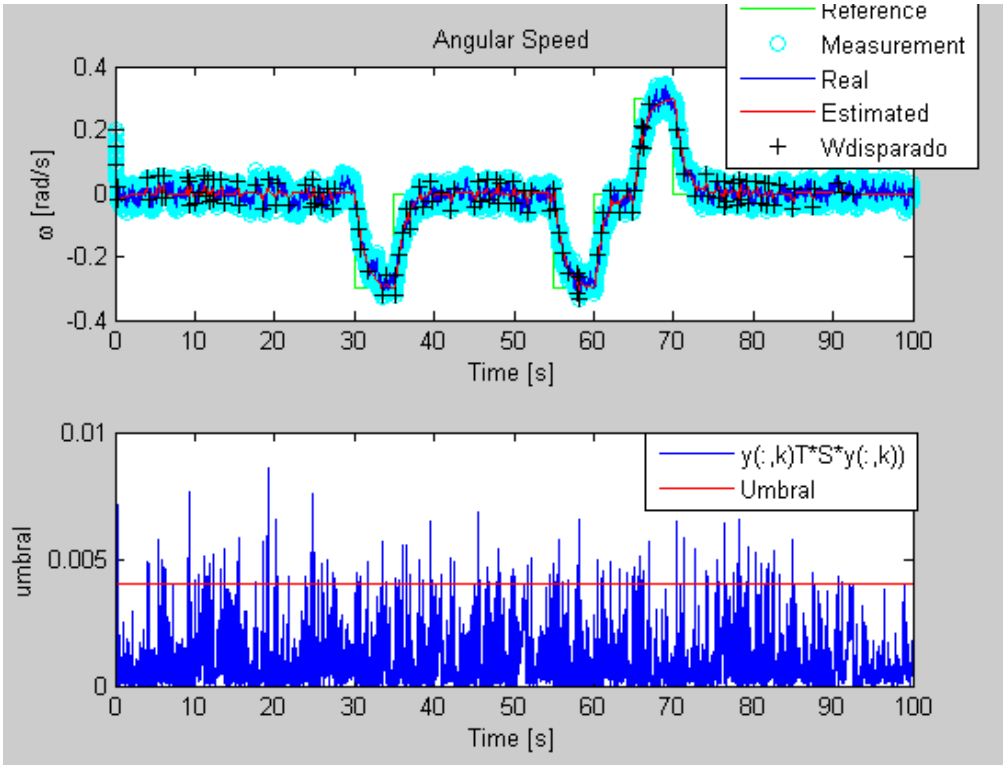

Figura 3. 11. Representación de la velocidad angular y sus errores para el SoD a 25Ts.

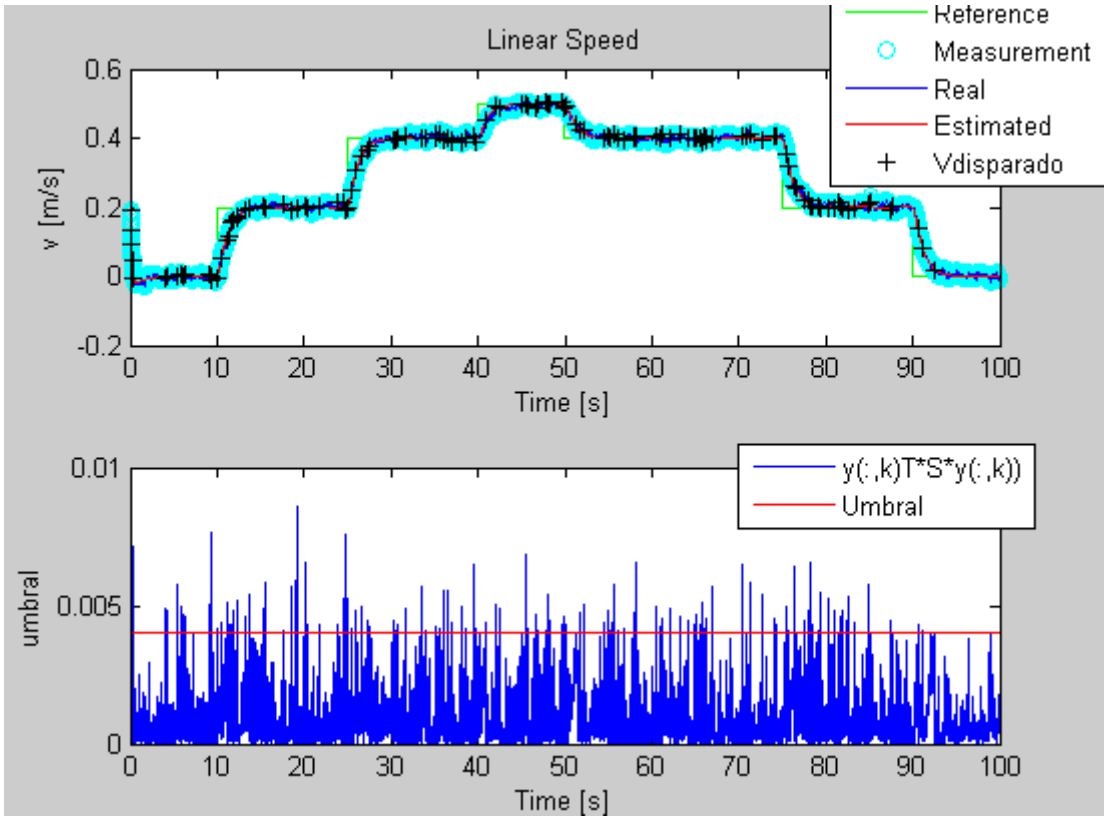

Figura 3. 12. Representación de la velocidad lineal y sus errores para el SoD a 25Ts.

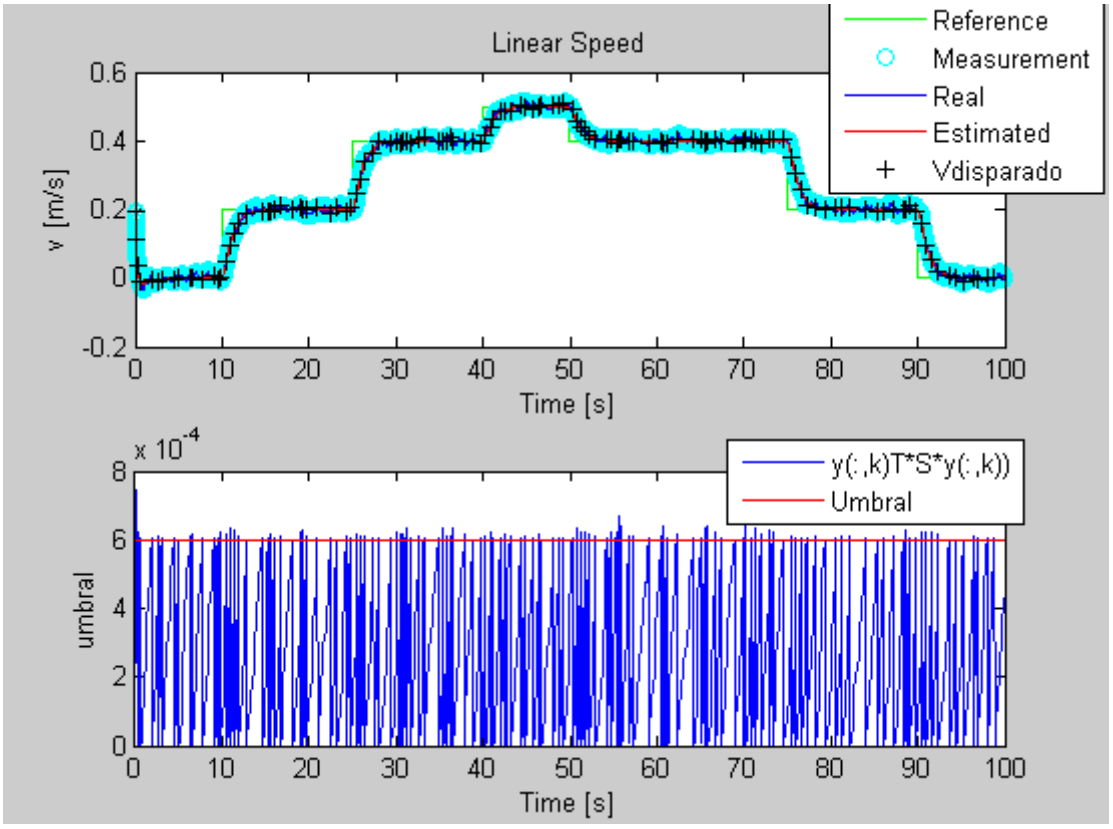

Figura 3. 13. Representación de la velocidad lineal y sus errores para el SoA a 25Ts.

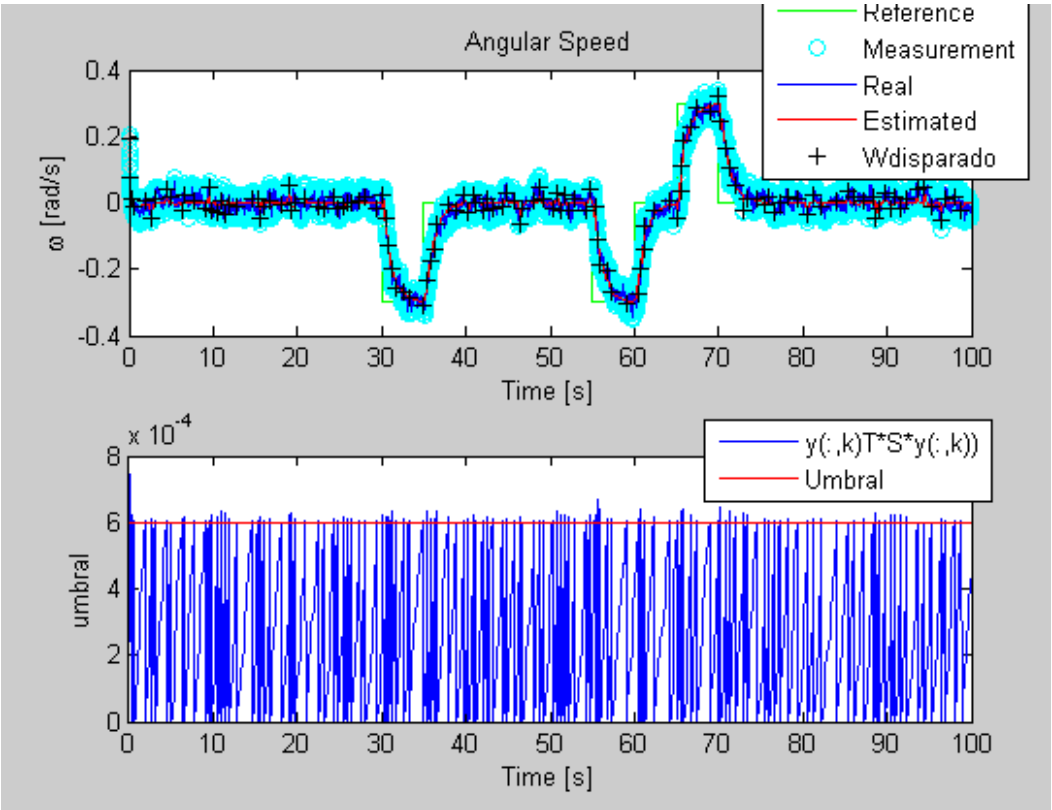

Figura 3. 14. Representación de la velocidad angular y sus errores para el SoA a 25Ts.

Se realiza otra simulación con periodo de muestreo 40Ts, obteniéndose los siguientes resultados.

Técnica	Tm	Umbral (Δ)	V_{rms}	ω_{rms}	Nº disparos
FK	40Ts	-	0,0087	0,0184	239
SOD	variable	0,009	0,00582	0,01837	48
SOA	variable	1.5e-3	0,0062	0,0184	85

Tabla 3.3: Resultados usando la ganancia estática y manteniendo un error de velocidad angular igual con un tiempo de muestreo de 40Ts.

Analizando los resultados se deduce que al incrementar el periodo de muestreo con el que se ejecuta la fase de predicción y se registra la salida del sistema, para un mismo error de estimación de la variable no medida, la reducción relativa del número de eventos en los que se corrige el estado predicho es mayor. En este caso el número de disparos con SoD se reduce al 4% y con SoA al 30% con respecto al caso periódico. Como en los casos anteriores, a continuación se muestran los registros temporales de las señales más relevantes obtenidas en simulación del robot P3-DX.

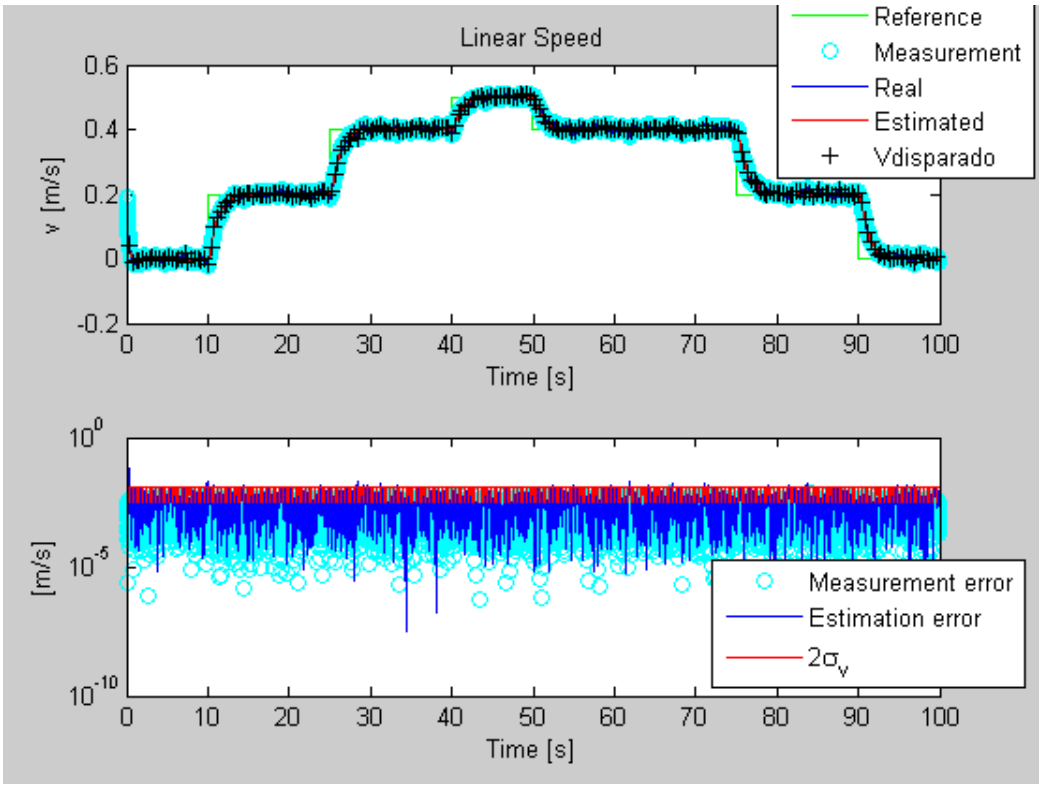

Figura 3. 15. Representación de la velocidad lineal y sus errores para el FK a 40Ts.

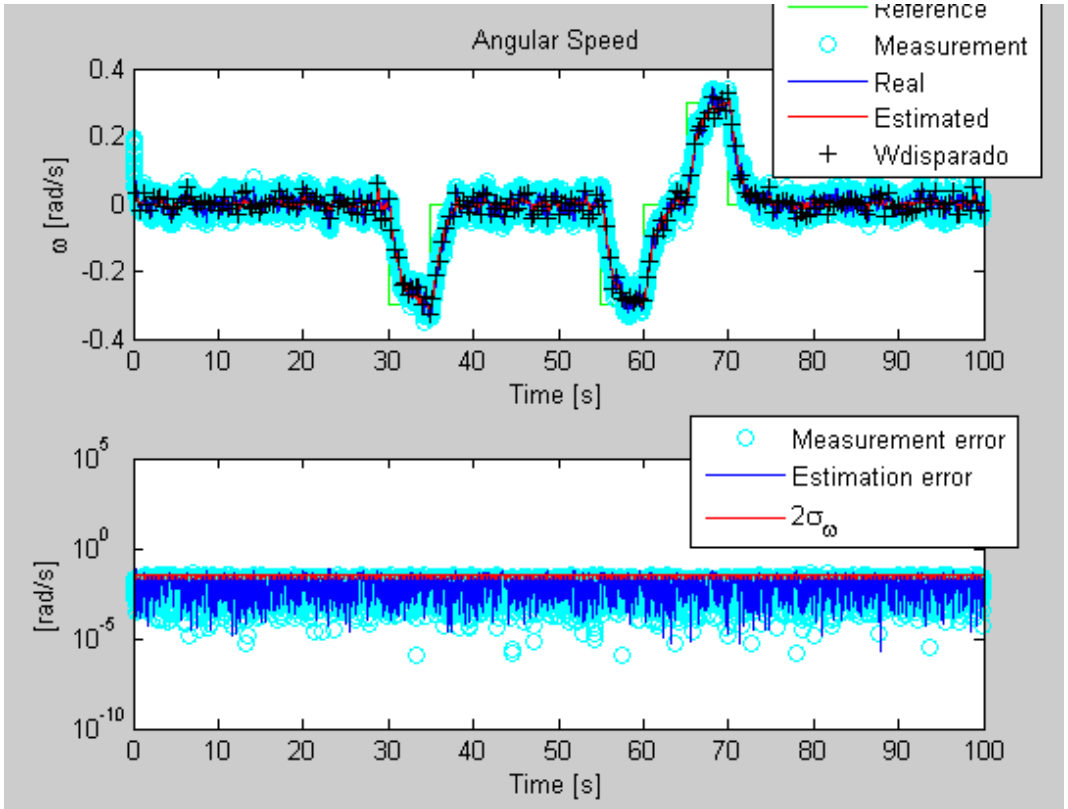

Figura 3. 16. Representación de la velocidad angular y sus errores para el FK a 40Ts.

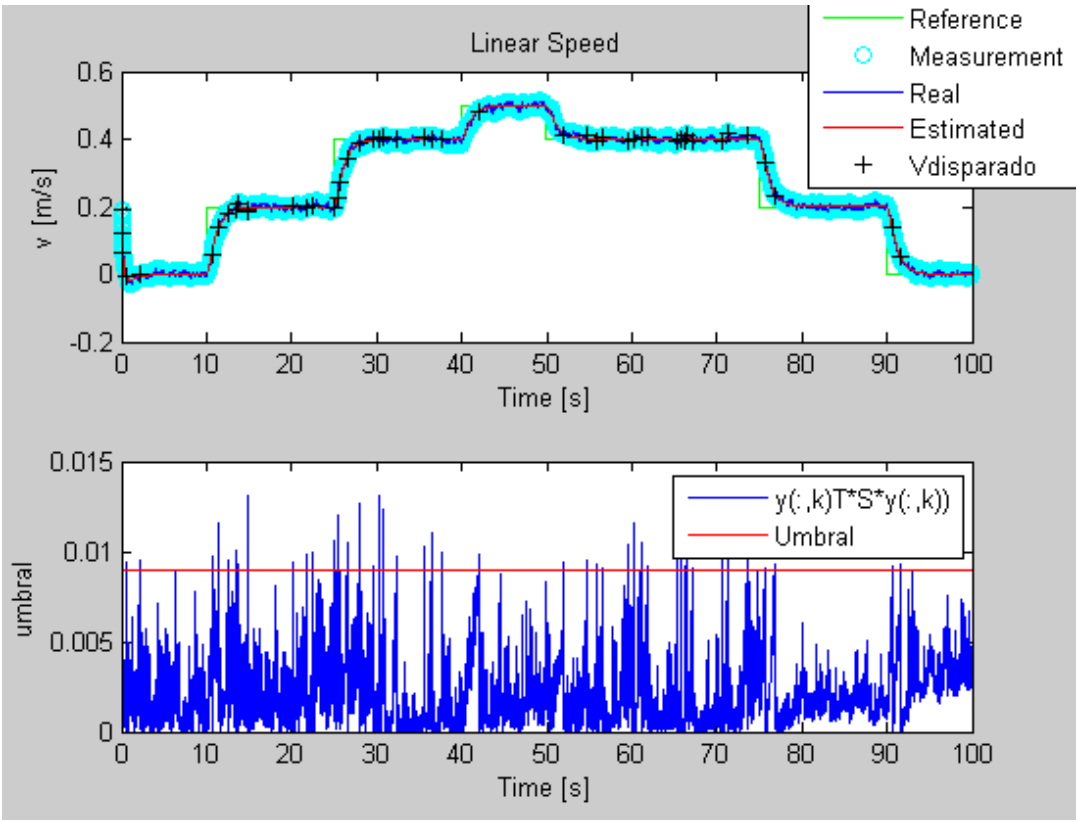

Figura 3. 17. Representación de la velocidad lineal y sus errores para el SOD a 40Ts.

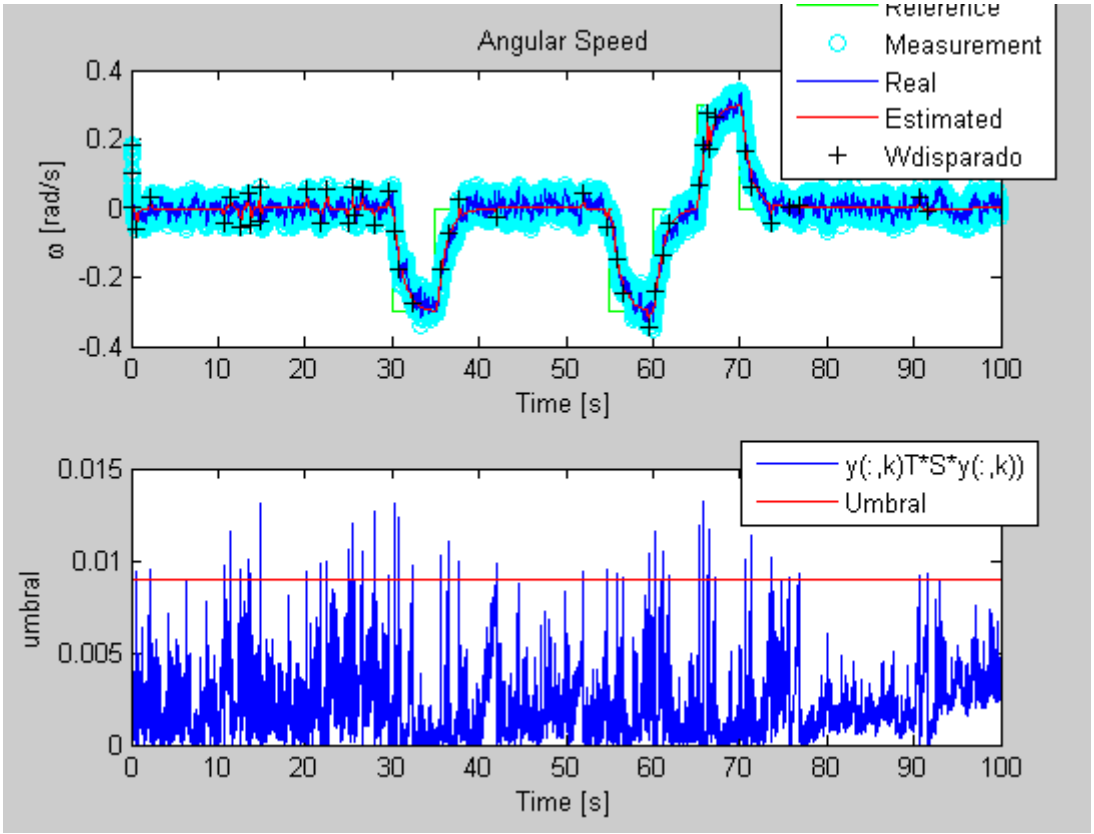

Figura 3. 18. Representación de la velocidad angular y sus errores para el SOD a 40Ts.

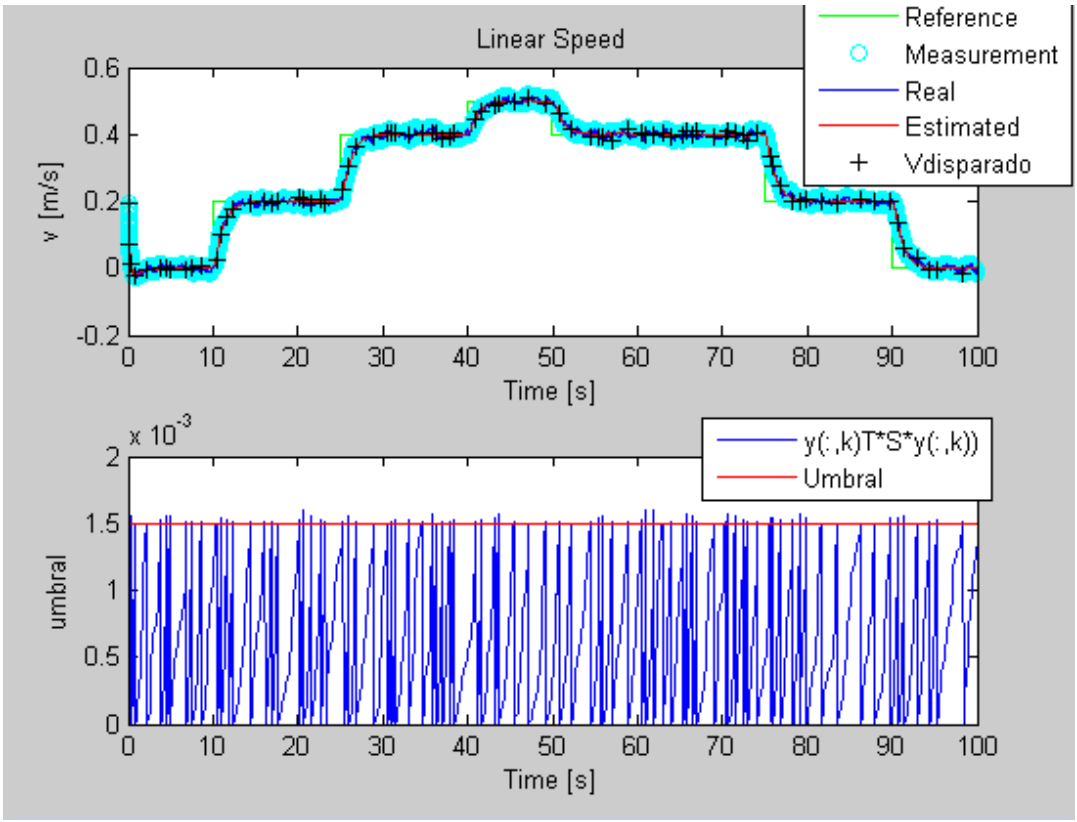

Figura 3. 19. Representación de la velocidad lineal y sus errores para el SOA a 40Ts.

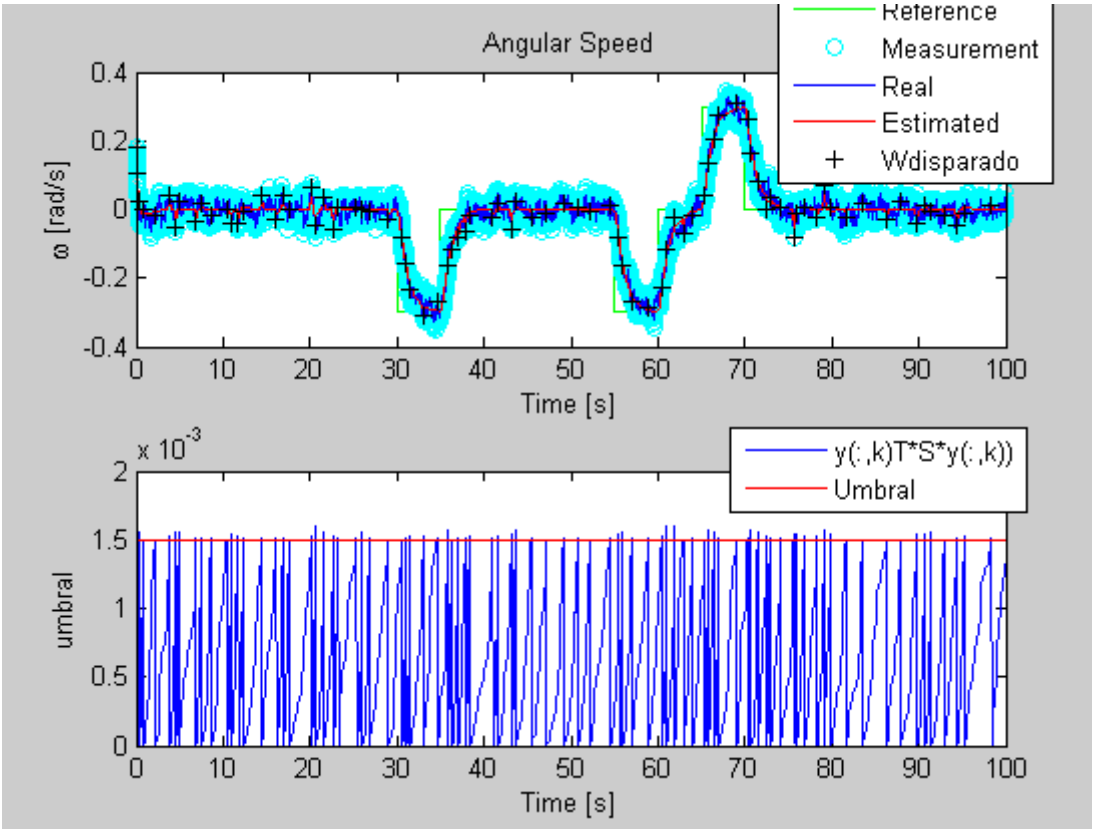

Figura 3. 20. Representación de la velocidad angular y sus errores para el SOA a 40Ts.

De la primera simulación se deduce que cuanto mayor es el periodo de muestreo con el que se registran datos de la planta (modelo) y se realiza la fase de predicción del estimador, menos afecta el ruido de medida; lo que contribuye a generar menos disparos para un mismo umbral.

Por otra parte, analizando los errores de estimación y de medida del sensor, se observa que el error producido en los intervalos de tiempo inmediatos a los de cambio de referencia (transitorios) es mayor que el correspondiente a los intervalos de régimen permanente, por lo que se planteó una segunda fase de simulación separando los errores en transitorio y en permanente.

Simulacion 2: El objetivo es analizar el diferente comportamiento de los estimadores basados en eventos en régimen transitorio y en régimen permanente.

El proceso es idéntico a la simulación 1. Para delimitar el régimen permanente se ha determinado el tiempo de establecimiento evaluando el tiempo que tarda la salida en alcanzar la referencia escalón de velocidad lineal con un error del 2% , tomando como referencia el caso de un tiempo de muestreo de 40 Ts.

Se analizó el tiempo de establecimiento para cada técnica y para cada valor de N. Esta parte se hizo evaluando los valores después de simulación y buscando el tiempo donde la velocidad lineal (salida) difería menos de un 2% con respecto a la de referencia. Los resultados se muestran en la tabla 3.4.

Técnica	10Ts	25Ts	40Ts
FK	4,3s	4,46s	4,52s
SOD	3,99s	4,17s	4,38s
SOA	4,09s	4,25s	4,24s

Tabla 3.4: Valores del tiempo de establecimiento para cada técnica y periodo de muestreo.

Como se puede observar, los valores de tiempo de asentamiento obtenidos son similares por lo que el tiempo considerado para este estudio es de T_{est}=4,21 como valor límite entre el régimen permanente y transitorio.

Se usan los mismos umbrales que en la simulación 1 porque aquí lo que queremos ver son los errores debido al transitorio.

Para Tm=10Ts

Técnica	umbral	V_{rms_Trans}	ω_{rms_Trans}	V_{rms}	ω_{rms}	Nº disparos transitorio	Nº disparos
FK	-	0,0423	0,0461	0,00339	0,0127	15	984
SoD	0,0015	0,0163	0,0211	0,0043	0,0140	12	532
SoA	7e-5	0,0169	0,0213	0,0043	0,0139	12	534

Tabla 3.5: Resultado separando los errores en transitorio y en permanente de la velocidad angular estimada tomando como referencia un tiempo de muestreo de 10 Ts.

Al calcular el error en velocidad angular en la parte transitoria y en la parte permanente con las mismas condiciones que la primera simulación, con 10Ts se observa que usando el filtro de Kalman el error en permanente se ha reducido de 0,0141 rad/s en 0,0127 rad/s. usando las otras técnincas (SoD y SoA), no hay mucha reducción. Para el send-on-delta pasamos de 0,0141 a 0,0140 y respecto al send-on-area varía de 0,0141 a 0,0139. Hay más error en transitorio con el FK pero los disparos se siguen produciendo en un mismo intervalo de tiempo sin depender de la dinámica del sistema.

Para Tm=25Ts

Técnica	umbral	V_{rms_Trans}	ω_{rms_Trans}	V_{rms}	ω_{rms}	Nº disparos transitorio	Nº disparos
FK	-	0,0513	0,0497	0,0044	0,0151	6	393
SoD	0,004	0,0169	0,0274	0,0054	0,0164	7	129
SoA	6e-4	0,0169	0,0235	0,00517	0,0173	5	130

Tabla 3.6: Resultado separando los errores en transitorio y en permanente angular igual con un tiempo de muestreo de 25 Ts.

Para 25Ts, en este caso se observa un comportamiento muy parecido al anterior.

Para Tm= 40Ts

Técnica	umbral	V_{rms_Trans}	ω_{rms_Trans}	V_{rms}	ω_{rms}	Nº disparos transitorio	Nº disparos
FK	-	0,0572	0,0556	0,00479	0,0178	3	246
SOD	0,009	0,0166	0,0278	0,0055	0,0178	4	50
SOA	1.5e-3	0,0164	0,0264	0,0053	0,0182	3	73

Tabla 3.7: Resultado separando los errores en transitorio y en permanente angular igual con un tiempo de muestreo de 40 Ts.

Los resultados de las tablas 3.6 y 3.7confirman las conclusiones comentadas para los valores numéricos de la tabla 3.5.

El FK es la técnica que más error acumula por el transitorio debido a que los disparos se producen en función del periodo, mientras que en las otras técnicas a saber el SoD y el

SoA, al producirse corrección de los estados cada vez que se supera el umbral, el error es más controlado y no crece desmesuradamente durante el transitorio.

Simulación 3: El objetivo de esta simulación es comprobar cuál es el efecto del ruido de estado y de medida en la comparativa de las tres técnicas de estimación presentadas.

Dado que la aplicación de las funciones de Matlab utilizadas hasta ahora no admiten valores nulos de la matriz Q, se ha multiplica el valor de las simulaciones anteriores por 10e-11, de forma que su contribución sea despreciable. Al igual que en simulación2, se diferencia entre régimen transitorio y régimen permanente.

Para Tm=10Ts

Técnica	umbral	V_{rms_Trans}	ω_{rms_Trans}	V_{rms}	ω_{rms}	Nº disparos transitorio	Nº disparos
FK	-	0,04214	0,0415	3,47e-13	1,4e-12	15	999
SoD	4e-5	0,01581	0,01658	4,9e-13	1,67e-12	32	400
SoA	1e-6	0,01581	0,0171	4,9e-13	1,73e-12	24	450

Tabla 3.8: Resultado separando los errores en transitorio y en permanente angular igual con un tiempo de muestreo de 10 Ts sin los ruidos.

Comparando con la tabla 3.5 se comprueba que, al no incorporar ruido en el estudio, el número de disparos se reduce de 512 a 400 en el caso SoD y de 534 a 450 en el caso SoA. También se aprecia que la aportación del régimen permanente al error medio de estimación de estados es despreciable.

Para Tm=25Ts

Técnica	umbral	V_{rms_Trans}	ω_{rms_Trans}	V_{rms}	ω_{rms}	Nº disparos transitorio	Nº disparos
FK	-	0,05259	0,04992	2,22e-9	3,462e-9	6	399
SoD	5e-4	0,0159	0,02102	5,58e-13	2,43e-12	2	115
SoA	5e-5	0,0161	0,0254	2,05e-11	2.53e-10	8	127

Tabla 3.9: Resultado separando los errores en transitorio y en permanente angular igual con un tiempo de muestreo de 25Ts sin los ruidos.

Si se compara con la tabla 3.6, la reducción del número de disparos no es tan significativa al prescindir de ruidos en la planta bajo estudio, pasando de 129 a 115 en el caso de SoD y de 130 a 127 en el caso de SoA. Con esos valores que no son muy diferentes podemos afirmar que usando el periodo 25Ts los disparos son más robustos a los ruidos del sistema y de medida y por lo tanto es el periodo idóneo para realizar una aplicación práctica con este modelo.

Para Tm= 40Ts

Técnica	umbral	V_{rms_Trans}	ω_{rms_Trans}	V_{rms}	ω_{rms}	Nº disparos transitorio	Nº disparos
FK	-	0,05523	0,0505	2,389e-9	2,736e-9	3	249
SOD	0,004	0,0193	0,1142	4,955e-7	2,64e-6	5	36
SOA	6e-4	0,020	0,121	6,02e-7	3,26e-6	4	56

Tabla 3.10: Resultado separando los errores en transitorio y en permanente angular igual con un tiempo de muestreo de 40 Ts sin los ruidos.

Los resultados de la tabla 3.10, también nos indica que se reduce el número de disparo. Si se dejan los umbrales iguales a los de las dos primeras simulaciones los resultados obtenidos son los siguientes:

Para Tm=10Ts

Técnica	umbral	V_{rms_Trans}	ω_{rms_Trans}	V_{rms}	ω_{rms}	Nº disparos transitorio	Nº disparos
FK	-	0,04214	0,0415	3,47e-13	1,4e-12	15	999
SoD	0,0015	0,0158	0,0177	5,80e-13	1,87e-12	6	64
SoA	7e-5	0,01581	0,0186	5,74e-13	1,87e-12	7	114

Tabla 3.11: Resultado separando los errores en transitorio y en permanente angular igual con un tiempo de muestreo de 10Ts sin los ruidos con los umbrales de la simulación 1.

Para Tm=25Ts

Técnica	umbral	V_{rms_Trans}	ω_{rms_Trans}	V_{rms}	ω_{rms}	Nº disparos transitorio	Nº disparos
FK	-	0,05259	0,04992	2,22e-9	3,462e-9	6	399
SoD	0,004	0,01622	0,02664	1,32e-9	1,05e-8	5	38
SoA	6e-4	0,01641	0,02991	4,596e-9	3,758e-8	5	56

Tabla 3.12: Resultado separando los errores en transitorio y en permanente angular igual con un tiempo de muestreo de 25Ts sin los ruidos con los umbrales de la simulación 1.

Para Tm= 40Ts

Técnica	umbral	V_{rms_Trans}	ω_{rms_Trans}	V_{rms}	ω_{rms}	Nº disparos transitorio	Nº disparos
FK	-	0,05523	0,0505	2,389e-9	2,736e-9	3	249
SOD	0,009	0,0199	0,1201	9,949e-7	3,780e-6	3	26
SOA	1,5e-3	0,2054	0,1237	8,459e-7	3,973e-6	3	43

Tabla 3.13: Resultado separando los errores en transitorio y en permanente angular igual con un tiempo de muestreo de 40Ts sin los ruidos con los umbrales de la simulación 1.

Usando los umbrales de la primera simulación observamos cómo se esperaba que hay una reducción más importante de los disparos. De Allí deducimos que una aplicación si el sistema y el medidor tienen poco ruido habrá menos disparos que si lo tuvieran para un mismo umbral.

4- Conclusiones y trabajos futuros

Este trabajo se ha llevado a cabo, para analizar el comportamiento de técnicas de estimación basada en eventos frente a la técnica clásica basada en el filtro de Kalman periódico. Se han analizado las técnicas de estimación aperiódica send-on-delta (SoD) y send-on-delta integrado o send-on-area (SoA); y se han aplicado a la estimación de estados de un robot P3-DX.

La efectividad de las técnicas de estimación basada en eventos depende de la elección del umbral de disparo. En nuestro caso, y aplicándolas el robot P3-DX con dos estados, se ha optado por establecer un umbral ponderando ambos estados en una única función escalar. La ponderación de los estados se deja a elección del diseñador.

En todos los casos la estimación de estados se realiza a partir de las dos etapas características del filtro de Kalman: predicción (periódica) y corrección (cuando se genera un evento según la técnica aplicada). Para poder comparar los resultados de las técnicas entre sí, se ha utilizado la ganancia estática del filtro común, recalculada para los diferentes casos de muestreo incluidos en la memoria (10Ts, 25Ts y 40Ts).

Tras los resultados de simulación obtenidos se concluye que las técnicas de estimación basadas en eventos contribuyen a reducir el intercambio de información entre el sistema sensorial y la unidad de control de una planta, cuando ambos subsistemas están enlazados de forma remota como ocurre en los sistemas de control en red (NCS).

Se ha comprobado que en sistemas con ruido, las técnicas de estimación aperiódica generan más eventos que cuando el ruido es despreciable. También se ha comprobado que la mayor contribución al error medio de estimación se genera cuando se producen cambios en las referencias, es decir en el régimen transitorio. Sin embargo, el error de estimación en el régimen transitorio se reduce tanto con SoD como con SoA frente al caso periódico.

La reducción del número de disparos (accesos al canal) es función de la dinámica de la planta, de la referencia aplicada a la misma y del ruido del sistema bajo estudio. La aplicación de SoA tiene más interés cuando las referencias presentas pocos cambios pues, si bien se generan más eventos, también se reduce el error de estimación frente al caso de SoD.

En trabajos futuros se propone validar los resultados simulados en experimentación con un robot P3-Dx real. También se puede investigar el efecto de la estimación aperiódica en el comportamiento del canal de comunicación inalámbrico, fundamentalmente el efecto en retardos y pérdidas de paquetes.

Anexos:

Variables aleatorias discretas

De niño, había un juego de dos equipos, el primer equipo era de los policías, y el segundo equipo era de los mercenarios. El juego consistía en que los mercenarios tenían que esconderse y les daban quince minutos para ello y luego los policías tenían que buscarles y arrestarles. También había un tiempo límite para encontrar a todos los mercenarios que era de media hora. Pero lo más importante de ese juego es como se elegían los miembros de cada grupo, y para ello vendaban los ojos del que decía los grupos y otro sin venda apuntaba a otro y preguntaba al de los ojos vendados a que equipo tenía que ir el que estaba apuntado sin decir su nombre, ni nada que le podía identificarlo. Como resultado todo el mundo sabía que podría ser policía o mercenario, pero nunca sabían con certeza cuál de los dos iba a ser.

Ese proceso de selección es un experimento aleatorio porque se sabe las posibles soluciones pero no la que va a salir en cada momento.

Otro ejemplo más común es el lanzamiento de un dado, se sabe que será una de sus seis caras pero no hay la certeza en la cara que va a salir precisamente. El valor del dado es una variable aleatoria debido a que es el resultado de un experimento aleatorio y además es discreto porque solo puede tener unos valores finitos a saber: 1; 2; 3; 4; 5 y 6.

Las variables aleatorias discretas como continuas, están asociadas a unos modelos matemáticos llamados funciones de distribución que indican la variabilidad de sus probabilidades. Antes de seguir se va definir con ecuaciones todos los conceptos enunciados en este bloque.

Definición: Una variable aleatoria discreta es una función que asocia a cada resultado del espacio muestral con valores finitos.

$$X=x_1; x_2;\ldots; x_i.$$

Ejemplo: lanzar une pieza de moneda y solo puede haber dos eventualidades que son cara o cruz.

$$L= \{\text{cara, cruz}\} \qquad \begin{cases} Y(\text{cara}) = 1 \\ Y(\text{cruz}) = 0 \end{cases}$$

Y es una variable aleatoria y L el espacio muestral.

La función de distribución F_X (t), es la probabilidad de que X tome valores menores o iguales a x.

$$F_X(t)= P(X \leq t) \quad \text{para todo t.}$$

La función de probabilidad de una variable aleatoria X es:

$$p_i = P(X=x_i) \text{ con } i=1,\ldots, k.$$

La función anterior tiene las siguientes propiedades:

- Para todo i, $p_i \geq 0$.
- $\sum pi = 1$.

La media μ de una variable aleatoria discreta X, también llamado esperanza matemática se define con la siguiente expresión:

$$\mu = E(X) = x_i p_i$$

La varianza de una variable aleatoria discreta X es una suma aritmética de la desviación de los valores x_i respecto a μ, elevado al cuadrado y ponderado por la función de probabilidad.

$$Var(X) = \sigma^2 = (x_i - \mu)^2 p_i = (\sum x_i^2 p_i) - \mu^2$$

Propiedades de la varianza:

- Sea X una constante, Var [X] =0.
- Sea a une constante, Var [aX] = a^2 Var [X].
- Sean a y b dos constantes, Var [aX+b] = a^2Var [X].
- La varianza de X no es aditiva Var $[\sum_{i=1}^{n} X_i] \neq \sum_{i=1}^{n} Var[X_i]$.
- Si X e Y son independientes Var [X+Y] = Var [X] + Var [Y].

Es difícil interpretar la varianza porque maneja unidades elevadas al cuadrado, para evitar ese problema se utiliza la desviación típica que es la raíz cuadrada de la varianza.

La desviación típica σ

$$\sigma = \sqrt{Var\ (X)}$$

Si σ =0, significa que todos los valores de x_i son iguales y que no existe dispersión.

Ejemplo:

Los resultados de un experimento aleatorio han sido los de la siguiente tabla:

i	1	2	3	4	5
X_i	2	4	1	7	3
p_i	$\frac{1}{11}$	$\frac{3}{11}$	$\frac{1}{11}$	$\frac{2}{11}$	$\frac{4}{11}$

- Determinación de la esperanza E(Xi)

$$E(Xi) = \frac{2+4+1+7+3}{5} = 3{,}4$$

- Determinación de la Varianza Var (Xi)

Var (Xi)= $(2-3{,}4)^2\frac{1}{11}+(4-3{,}4)^2\frac{3}{11}+(1-3{,}4)^2\frac{1}{11}+(7-3{,}4)^2\frac{2}{11}+$ $(3-3{,}4)^2\frac{4}{11}$ = 3,21 uds^2

- Determinación de la desviación típica σ

σ = $\sqrt{3{,}21}$ = 1,79 uds.

Códigos de programa

FK_1D

```
% 1D Object tracking with an Event-Based State Estimator

%% Set-up
clear all
%  --- Simulation parameters

Ts = 0.1;
T_final = 8;
k_final = T_final/Ts;
t_axis = 0:Ts:T_final-Ts;

% --- System parameters
F = [1 Ts; 0 1];
G = [Ts^2/2; Ts];
H = [1 0];
[M, N] = size(H); % Number of outputs and states

% --- Sensors and Noise
% - System noise
Q = 0.02;
load 1D_acceleration;
w = acceleration;

% - Sensor noise
R = 0.1e-3;
sigma_v = sqrt(R);

% --- Initial conditions
x_0 = [0 0];
x_hat_0 = zeros(N,1);
P_0 = diag([0.025 0.025]);


% Array initialization
x = zeros(N, k_final);      % Plant states
x(:,1) = x_0;

x_hat = x;                  % Estimated states
x_hat(:,1) = x_hat_0;

P = zeros(N,N,k_final);     % Estimation error covariance
P(:,:,1) = P_0;

y = zeros(M, k_final)*NaN;  % Sensor measurements

%% Compute simulation
for k = 2:k_final
    % Apply plant model
    x(:,k) = F * x(:,k-1) + G * w(k-1);

    % KF: prediction
```

```
    x_hat(:,k) = F * x_hat(:,k-1);
    P(:,:,k) = F * P(:,:,k-1) * F' + G * Q * G';

    % Sensor measurement
    y(:,k) = H * x(:,k) + sigma_v.*randn(M,1);

        L = P(:,:,k) * H' / (H * P(:,:,k) * H' + R);    % Kalman gain
        x_hat(:,k) = x_hat(:,k) +  L * (y(:,k) - H * x_hat(:,k));
        P(:,:,k) = (eye(N) - L * H) * P(:,:,k);

end

%% Results
fprintf('Position RMSE:  %g m\n', sqrt(mean((x(1,:)-x_hat(1,:)).^2)));
fprintf('Speed RMSE:     %g m/s\n\n', sqrt(mean((x(2,:)-
x_hat(2,:)).^2)));

% Plots
figure
plot(t_axis,y(1,:),'oc', t_axis,x(1,:),'b',
t_axis,x_hat(1,:),'r',t_axis,y(1,:),'+k');
title('Position')
xlabel('Time [s]')
ylabel('Position [m/s]')
legend('Measurement', 'Ideal', 'Estimated','X-evento')

figure
plot(t_axis,x(2,:),'b', t_axis,x_hat(2,:),'r');
title('Speed')
xlabel('Time [s]')
ylabel('v [m/s]')
legend('Ideal', 'Estimated')
```

SOD_1D

```
% 1D Object tracking with an Event-Based State Estimator

%% Set-up
clear all
%  --- Simulation parameters

Ts = 0.1;
T_final = 8;
k_final = T_final/Ts;
t_axis = 0:Ts:T_final-Ts;
Delta =0.1;
i=1;
num_disparo=0;
measurement=[];
% --- System parameters
F = [1 Ts; 0 1];
G = [Ts^2/2; Ts];
H = [1 0];

[M, N] = size(H); % Number of outputs and states
```

```
% --- Sensors and Noise
% - System noise
Q = 0.02;
load 1D_acceleration;
w = acceleration;

% - Sensor noise
R = 0.1e-3;
sigma_v = sqrt(R);

% --- Initial conditions
x_0 = [0 0];
x_hat_0 = zeros(N,1);
P_0 = diag([0.025 0.025]);


% Array initialization
x = zeros(N, k_final);      % Plant states
x(:,1) = x_0;

x_hat = x;                  % Estimated states
x_hat(:,1) = x_hat_0;

P = zeros(N,N,k_final);     % Estimation error covariance
P(:,:,1) = P_0;

y = zeros(M, k_final)*NaN;  % Sensor measurements
y(:,i)=0;
diff= zeros (M,k_final);

%% Compute simulation

for k = 2:k_final
    % Apply plant model
    x(:,k) = F * x(:,k-1) + G * w(k-1);

    % KF: prediction
    x_hat(:,k) = F * x_hat(:,k-1);
    P(:,:,k) = F * P(:,:,k-1) * F' + G * Q * G';

    % Sensor measurement
    y(:,k) = H * x(:,k) + sigma_v.*randn(M,1);

    diff=abs(y(:,k)-y(:,i));
     if (diff>Delta)

        i=k;
        num_disparo=num_disparo+1;
        measurement=cat(1,measurement,k);
        L = P(:,:,k) * H' / (H * P(:,:,k) * H' + R);    % Kalman gain
        x_hat(:,k) = x_hat(:,k) +  L * (y(:,k) - H * x_hat(:,k));
        P(:,:,k) = (eye(N) - L * H) * P(:,:,k);

     end
```

```
end

%% Results
fprintf('Position RMSE:  %g m\n', sqrt(mean((x(1,:)-x_hat(1,:)).^2)));
fprintf('Speed RMSE:     %g m/s\n\n', sqrt(mean((x(2,:)-
x_hat(2,:)).^2)));

% Plots
figure
plot(t_axis,y(1,:),'oc', t_axis,x(1,:),'b',
t_axis,x_hat(1,:),'r',t_axis(measurement),y(1,measurement),'+k');
title('Position')
xlabel('Time [s]')
ylabel('Position [m]')
legend('Measurement', 'Ideal', 'Estimated','X-evento')

figure
plot(t_axis,x(2,:),'b', t_axis,x_hat(2,:),'r');
title('Speed')
xlabel('Time [s]')
ylabel('v [m/s]')
legend('Ideal', 'Estimated')
```

SODI_1D

```
% 1D Object tracking with an Event-Based State Estimator
%% Set-up
clear all
close all
%  --- Simulation parameters
Ts = 0.1;
T_final = 8;
k_final = T_final/Ts;
t_axis = 0:Ts:T_final-Ts;
Delta=0.001;
i=1;
num_disparo=0;
measurement=[];
% --- System parameters
F = [1 Ts; 0 1];
G = [Ts^2/2; Ts];
H = [1 0];
[M, N] = size(H); % Number of outputs and states

% --- Sensors and Noise
% - System noise
Q = 0.02;
load 1D_acceleration;
w = acceleration;

% - Sensor noise
R = 0.1e-3;
sigma_v = sqrt(R);
```

```
% --- Initial conditions
x_0 = [0 0];
x_hat_0 = zeros(N,1);
P_0 = diag([0.025 0.025]);


% Array initialization
x = zeros(N, k_final);      % Plant states
x(:,1) = x_0;

x_hat = x;                  % Estimated states
x_hat(:,1) = x_hat_0;

P = zeros(N,N,k_final);     % Estimation error covariance
P(:,:,1) = P_0;

y = zeros(M, k_final)*NaN;  % Sensor measurements
y(:,i)=0;
integral=zeros(1, k_final);
rest=zeros(M,k_final);

%% Compute simulation

for k = 2:k_final
    % Apply plant model
    x(:,k) = F * x(:,k-1) + G * w(k-1);

    % KF: prediction
    x_hat(:,k) = F * x_hat(:,k-1);
    P(:,:,k) = F * P(:,:,k-1) * F' + G * Q * G';

    % ---- Send-on-Delta ----

    % Sensor measurement
    y(:,k) = H * x(:,k) + sigma_v.*randn(M,1);
     for a=i:k
        rest(:,a)=abs(y(:,a)-y(:,i));

    end

    t=0.01*(i:k);

    integral(k)=trapz(t,rest(i:k));

    if (~isfinite(integral(k)) || integral(k)>Delta)
        i=k;

        num_disparo=num_disparo+1;
        measurement=cat(1,measurement,k);
        L = P(:,:,k) * H' / (H * P(:,:,k) * H' + R);    % Kalman gain
        x_hat(:,k) = x_hat(:,k) +  L * (y(:,k) - H * x_hat(:,k));
        P(:,:,k) = (eye(N) - L * H) * P(:,:,k);
     end
    % ----------------------
```

```
end


%% Results
fprintf('Position RMSE:  %g m\n', sqrt(mean((x(1,:)-x_hat(1,:)).^2)));
fprintf('Speed RMSE:     %g m/s\n\n', sqrt(mean((x(2,:)-
x_hat(2,:)).^2)));

% Plots
figure
plot(t_axis,y(1,:),'oc', t_axis,x(1,:),'b',
t_axis,x_hat(1,:),'r',t_axis(measurement),y(1,measurement),'+k');
title('Position')
xlabel('Time [s]')
ylabel('Position [m]')
legend('Measurement', 'Ideal', 'Estimated','X-evento')

figure
plot(t_axis,x(2,:),'b', t_axis,x_hat(2,:),'r');
title('Speed')
xlabel('Time [s]')
ylabel('v [m/s]')
legend('Ideal', 'Estimated')
```

FK_Sim1

```
  % Speed tracking of robot with servo-system
% using an Event-Based State Estimator (EBSE)

%% Set-up
clear all
%  --- Simulation parameters

Ts = 10e-3;
T_final = 100;
k_final = T_final/Ts;
t_axis=0:Ts:T_final-Ts;
j=0;
l=10;

indice=[];
num_disparos=0;
Qn=zeros(4,4);
% --- System parameters
A = 1.0e+03 *[
  -0.004094199291309  -0.000015025848246   1.663549680532014
0.000722682271388
  -0.000008063153776  -0.005041683869938   0.000326285936661
2.022770901637336
                   0                   0  -0.200000000000000
0
                   0                   0                   0  -
0.200000000000000];
```

```
B = [-4.158874201330035  -0.001806705678471
     -0.000815714841653  -5.056927254093339
                       1                   0
                       0                   1];

H = [eye(2) zeros(2, 2)];

Ki= [  0.999999739408795  0.000721929696507
      -0.000721929696770  0.999999739408544];

Kr= [ -0.259902871873893   0.000493748739433  -2.167013573124378
0.004017102111344
       0.000526013153705  -0.222222011238482   0.003996171390493  -
2.256249376742330];


[M, N] = size(H); % Number of outputs and states

% Discrete plant
F = expm(A*Ts);
G = A\(F - eye(N))*B;

% --- Sensors and Noise
% - System noise
sigma_w = [2e-4 6e-4 2e-4 6e-4]';
Q = diag(sigma_w.^2);

% - Sensor noise
sigma_v = [1.75734*10e-4 1.054404*10e-3]';
R = diag(sigma_v.^2);

% --- Initial conditions
x_0 = [0.2 0.2 0 0];
x_hat_0 = zeros(N,1);
P_0 = diag([0.1 0.6 0.1 0.6]);

% Reference speeds
%  - 1. linear speed
ref = zeros(2,k_final);
ref(1, (10/Ts):(25/Ts)) = 0.2;
ref(1, (25/Ts):(40/Ts)) = 0.4;
ref(1, (40/Ts):(50/Ts)) = 0.5;
ref(1, (50/Ts):(75/Ts)) = 0.4;
ref(1, (75/Ts):(90/Ts)) = 0.2;
%  - 2. linear speed
w_0 = 0.3;
ref(2, (30/Ts):round(35.02/Ts)) = -w_0;
ref(2, (55/Ts):round(60.02/Ts)) = -w_0;
ref(2, (65/Ts):round(70.02/Ts)) = w_0;

% Array initialization
x = zeros(N, k_final);      % Plant states
x(:,1) = x_0;

x_hat = x;                  % Estimated states
x_hat(:,1) = x_hat_0;
```

```
P = zeros(N,N,k_final);     % Estimation error covariance
P(:,:,1) = P_0;
Fn=F^(l);
for a=0:1:l-1
    Qn=Qn+F^a*Q*(F^a)';
end
[P_inf,Y,G_estatica]=dare(Fn',H',Qn,R);
L_inf=P_inf*H'/(H*P_inf*H'+R);
y = zeros(M, k_final)*NaN;  % Sensor measurements

u = zeros(2, k_final);      % Control signal (plat inputs)
n = zeros(2, k_final);      % Integrator output


%% Compute simulation

for k = 2:k_final
    % Apply plant model
    x(:,k) = F * x(:,k-1) + G * u(:,k-1) + sigma_w.*randn(N,1);

    % KF: prediction
    x_hat(:,k) = F * x_hat(:,k-1) + G * u(:,k-1);
    %P(:,:,k) = F * P(:,:,k-1) * F' + Q;
    P(:,:,k)=P_inf;

    % ---- Send-on-Delta ----

    % Sensor measurement
    y(:,k) = H * x(:,k) + sigma_v.*randn(M,1);
    j=j+1;

    if (j==l)
        j=0;
        num_disparos=num_disparos+1;
        indice=cat(1,indice,k);
        % KF: correction
        %L = P(:,:,k) * H' / (H * P(:,:,k) * H' + R);    % Kalman gain
        x_hat(:,k) = x_hat(:,k) +  L_inf * (y(:,k) - H * x_hat(:,k));
        P(:,:,k) = (eye(N) - L_inf * H) * P(:,:,k) * (eye(N) - L_inf *
H)'+L_inf * R * L_inf';
    end
    % ----------------------

    % Compute next control signal
    n(:,k) = n(:,k-1) + (ref(:,k) - H * x_hat(:,k)) * Ts; % Integrate
error
    u(:,k) = Ki * n(:,k) + Kr * x_hat(:,k);
end


%% Results
fprintf('Linear speed RMSE:  %g m/s\n', sqrt(mean((x(1,:)-
x_hat(1,:)).^2)));
fprintf('Angular speed RMSE: %g rad/s\n\n', sqrt(mean((x(2,:)-
x_hat(2,:)).^2)));
```

```
% Plots
figure

subplot(2,1,1)
plot(t_axis,ref(1,:),'g', t_axis,y(1,:),'oc', t_axis,x(1,:),'b',
t_axis,x_hat(1,:),'r',t_axis(indice),y(1,indice),'+k');
title('Linear Speed')
xlabel('Time [s]')
ylabel('v [m/s]')
legend('Reference', 'Measurement', 'Real', 'Estimated','Vdisparado')

subplot(2,1,2)
semilogy(t_axis, abs(x(1,:) - y(1,:)),  'oc', ...
         t_axis, abs(x(1,:) - x_hat(1,:)),  'b',  ...
         t_axis, 2*squeeze(sqrt(P(1,1,:))), 'r')
xlabel('Time [s]')
ylabel('[m/s]')
legend('Measurement error', 'Estimation error', '2\sigma_v');

figure

subplot(2,1,1)
plot(t_axis,ref(2,:),'g', t_axis,y(2,:),'oc', t_axis,x(2,:),'b',
t_axis,x_hat(2,:),'r',t_axis(indice),y(2,indice),'+k');
title('Angular Speed')
xlabel('Time [s]')
ylabel('\omega [rad/s]')
legend('Reference', 'Measurement', 'Real', 'Estimated','Wdisparado')

subplot(2,1,2)
semilogy(t_axis, abs(x(2,:) - y(2,:)),  'oc', ...
         t_axis, abs(x(2,:)- x_hat(2,:)),   'b', ...
         t_axis, 2*squeeze(sqrt(P(2,2,:))), 'r')
xlabel('Time [s]')
ylabel('[rad/s]')
legend('Measurement error', 'Estimation error', '2\sigma_\omega');
```

SoD_Sim1

```
% Speed tracking of robot with servo-system
% using an Event-Based State Estimator (EBSE)

%% Set-up
clear all
%  --- Simulation parameters

Ts = 10e-3;
T_final = 100;
k_final = T_final/Ts;
t_axis=0:Ts:T_final-Ts;

Umbral=0.0015;
num=0;
```

```
l=10;
i=1;
Um=repmat(Umbral,1,k_final);
indice=[];
Qn=zeros(4,4);
% --- System parameters
A = 1.0e+03 *[
  -0.004094199291309  -0.000015025848246   1.663549680532014
0.000722682271388
  -0.000008063153776  -0.005041683869938   0.000326285936661
2.022770901637336
                   0                   0  -0.200000000000000
0
                   0                   0                   0  -
0.200000000000000];


B = [-4.158874201330035  -0.001806705678471
     -0.000815714841653  -5.056927254093339
                      1                   0
                      0                   1];

H = [eye(2) zeros(2, 2)];

Ki= [  0.999999739408795  0.000721929696507
      -0.000721929696770  0.999999739408544];

Kr= [ -0.259902871873893   0.000493748739433  -2.167013573124378
0.004017102111344
       0.000526013153705  -0.222222011238482   0.003996171390493  -
2.256249376742330];


[M, N] = size(H); % Number of outputs and states

% Discrete plant
F = expm(A*Ts);
G = A\(F - eye(N))*B;

% --- Sensors and Noise
% - System noise
sigma_w = [2e-4 6e-4 2e-4 6e-4]';
Q = diag(sigma_w.^2);

% - Sensor noise
sigma_v = [1.75734*10e-4 1.054404*10e-3]';
R = diag(sigma_v.^2);

% --- Initial conditions
x_0 = [0.2 0.2 0 0];
x_hat_0 = zeros(N,1);
P_0 = diag([0.1 0.6 0.1 0.6]);

% Reference speeds
%  - 1. linear speed
ref = zeros(2,k_final);
ref(1, (10/Ts):(25/Ts)) = 0.2;
```

```
ref(1, (25/Ts):(40/Ts)) = 0.4;
ref(1, (40/Ts):(50/Ts)) = 0.5;
ref(1, (50/Ts):(75/Ts)) = 0.4;
ref(1, (75/Ts):(90/Ts)) = 0.2;
%  - 2. linear speed
w_0 = 0.3;
ref(2, (30/Ts):round(35.02/Ts)) = -w_0;
ref(2, (55/Ts):round(60.02/Ts)) = -w_0;
ref(2, (65/Ts):round(70.02/Ts)) = w_0;

% Array initialization
x = zeros(N, k_final);      % Plant states
x(:,1) = x_0;

x_hat = x;                  % Estimated states
x_hat(:,1) = x_hat_0;

P = zeros(N,N,k_final);     % Estimation error covariance
P(:,:,1) = P_0;
Fn=F^(l);
for a=0:1:l-1
    Qn=Qn+F^a*Q*(F^a)';
end
[P_inf,Y,G_estatica]=dare(Fn',H',Qn,R);
L_inf=P_inf*H'/(H*P_inf*H'+R);
y = zeros(M, k_final)*NaN;  % Sensor measurements

u = zeros(2, k_final);      % Control signal (plat inputs)
n = zeros(2, k_final);      % Integrator output
diferencia=zeros(1, k_final);
rest=zeros(2,k_final);
S=[1.2 0;0 1];


%% Compute simulation


for k = 2:k_final
    % Apply plant model
    x(:,k) = F * x(:,k-1) + G * u(:,k-1) + sigma_w.*randn(N,1);

    % KF: prediction
    x_hat(:,k) = F * x_hat(:,k-1) + G * u(:,k-1);
    %P(:,:,k) = F * P(:,:,k-1) * F' + Q;
    P(:,:,k)=P_inf;

    % ---- Send-on-Delta ----

    % Sensor measurement
    y(:,k) = H * x(:,k) + sigma_v.*randn(M,1);
    rest(:,k)=abs(y(:,k)-y(:,i));
    diferencia(k)=rest(:,k)'*S*rest(:,k);
    if (~isfinite(diferencia(k)) || diferencia(k)>Umbral)

        i=k;

        num=num+1;
```

```
        indice=cat(1,indice,i);

        % KF: correction
        %L = P(:,:,k) * H' / (H * P(:,:,k) * H' + R);     % Kalman gain
        x_hat(:,k) = x_hat(:,k) +  L_inf * (y(:,k) - H * x_hat(:,k));
        P(:,:,k) = (eye(N) - L_inf * H) * P(:,:,k) * (eye(N) - L_inf *
H)'+L_inf * R * L_inf';
    end
    % ----------------------

    % Compute next control signal
    n(:,k) = n(:,k-1) + (ref(:,k) - H * x_hat(:,k)) * Ts; % Integrate
error
    u(:,k) = Ki * n(:,k) + Kr * x_hat(:,k);
end


%% Results
fprintf('Linear speed RMSE:  %g m/s\n', sqrt(mean((x(1,:)-
x_hat(1,:)).^2)));
fprintf('Angular speed RMSE: %g rad/s\n\n', sqrt(mean((x(2,:)-
x_hat(2,:)).^2)));


% Plots
figure

subplot(2,1,1)
plot(t_axis,ref(1,:),'g', t_axis,y(1,:),'oc', t_axis,x(1,:),'b',
t_axis,x_hat(1,:),'r',t_axis(indice),y(1,indice),'+k');
title('Linear Speed')
xlabel('Time [s]')
ylabel('v [m/s]')
legend('Reference', 'Measurement', 'Real', 'Estimated','Vdisparado')

subplot(2,1,2)
plot(t_axis,diferencia,'b',t_axis,Um,'r')
xlabel('Time [s]')
ylabel('umbral')
legend('y(:,k)T*S*y(:,k))', 'Umbral')

figure
subplot(2,1,1)
plot(t_axis,ref(2,:),'g', t_axis,y(2,:),'oc', t_axis,x(2,:),'b',
t_axis,x_hat(2,:),'r',t_axis(indice),y(2,indice),'+k');
title('Angular Speed')
xlabel('Time [s]')
ylabel('\omega [rad/s]')
legend('Reference', 'Measurement', 'Real', 'Estimated','Wdisparado')

subplot(2,1,2)
plot(t_axis,diferencia,'b',t_axis,Um,'r')
xlabel('Time [s]')
ylabel('umbral')
legend('y(:,k)T*S*y(:,k))', 'Umbral')
```

SoA_Sim1

```
% Speed tracking of robot with servo-system
% using an Event-Based State Estimator (EBSE)

%% Set-up
clear all
%  --- Simulation parameters

Ts = 10e-3;
T_final = 100;
k_final = T_final/Ts;
t_axis=0:Ts:T_final-Ts;

Umbral=6e-5;
num=0;
l=10;
i=1;
Um=repmat(Umbral,1,k_final);
indice=[];
Qn=zeros(4,4);
% --- System parameters
A = 1.0e+03 *[
  -0.004094199291309  -0.000015025848246   1.663549680532014
0.000722682271388
  -0.000008063153776  -0.005041683869938   0.000326285936661
2.022770901637336
                   0                   0  -0.200000000000000
0
                   0                   0                   0  -
0.200000000000000];


B = [-4.158874201330035  -0.001806705678471
     -0.000815714841653  -5.056927254093339
                      1                   0
                      0                   1];

H = [eye(2) zeros(2, 2)];

Ki= [  0.999999739408795  0.000721929696507
      -0.000721929696770  0.999999739408544];

Kr= [ -0.259902871873893   0.000493748739433  -2.167013573124378
0.004017102111344
       0.000526013153705  -0.222222011238482   0.003996171390493  -
2.256249376742330];


[M, N] = size(H); % Number of outputs and states

% Discrete plant
F = expm(A*Ts);
G = A\(F - eye(N))*B;
```

```
% --- Sensors and Noise
% - System noise
sigma_w = [2e-4 6e-4 2e-4 6e-4]';
Q = diag(sigma_w.^2);

% - Sensor noise
sigma_v = [1.75734*10e-4 1.054404*10e-3]';
R = diag(sigma_v.^2);

% --- Initial conditions
x_0 = [0.2 0.2 0 0];
x_hat_0 = zeros(N,1);
P_0 = diag([0.1 0.6 0.1 0.6]);

% Reference speeds
%  - 1. linear speed
ref = zeros(2,k_final);
ref(1, (10/Ts):(25/Ts)) = 0.2;
ref(1, (25/Ts):(40/Ts)) = 0.4;
ref(1, (40/Ts):(50/Ts)) = 0.5;
ref(1, (50/Ts):(75/Ts)) = 0.4;
ref(1, (75/Ts):(90/Ts)) = 0.2;
%  - 2. linear speed
w_0 = 0.3;
ref(2, (30/Ts):round(35.02/Ts)) = -w_0;
ref(2, (55/Ts):round(60.02/Ts)) = -w_0;
ref(2, (65/Ts):round(70.02/Ts)) = w_0;

% Array initialization
x = zeros(N, k_final);      % Plant states
x(:,1) = x_0;

x_hat = x;                  % Estimated states
x_hat(:,1) = x_hat_0;

P = zeros(N,N,k_final);     % Estimation error covariance
P(:,:,1) = P_0;
Fn=F^(l);
for a=0:1:l-1
    Qn=Qn+F^a*Q*(F^a)';
end
[P_inf,Y,G_estatica]=dare(Fn',H',Qn,R);
L_inf=P_inf*H'/(H*P_inf*H'+R);
y = zeros(M, k_final)*NaN;  % Sensor measurements


u = zeros(2, k_final);      % Control signal (plat inputs)
n = zeros(2, k_final);      % Integrator output
integral=zeros(1, k_final);
m=zeros(M,k_final);
rest=zeros(M,k_final);
S=[1.5 0;0 1];

%% Compute simulation

for k = 2:k_final
```

```
    % Apply plant model
    x(:,k) = F * x(:,k-1) + G * u(:,k-1) + sigma_w.*randn(N,1);

    % KF: prediction
    x_hat(:,k) = F * x_hat(:,k-1) + G * u(:,k-1);
    %P(:,:,k) = F * P(:,:,k-1) * F' + Q;
    P(:,:,k)=P_inf;

    % ---- Send-on-Delta ----

    % Sensor measurement
    y(:,k) = H * x(:,k) + sigma_v.*randn(M,1);
    for a=i:k
        rest(:,a)=abs(y(:,a)-y(:,i));
        m(a)=rest(:,a)'*S*rest(:,a);

    end

    t=0.01*(i:k);

    integral(k)=trapz(t,m(i:k));

    if (~isfinite(integral(k)) || integral(k)>Umbral)
        i=k;

        num=num+1;

        indice=cat(1,indice,i);

        % KF: correction
        %L = P(:,:,k) * H' / (H * P(:,:,k) * H' + R);    % Kalman gain
        x_hat(:,k) = x_hat(:,k) +  L_inf * (y(:,k) - H * x_hat(:,k));
        P(:,:,k) = (eye(N) - L_inf * H) * P(:,:,k) * (eye(N) - L_inf *
H)'+L_inf * R * L_inf';
    end
    % ----------------------

    % Compute next control signal
    n(:,k) = n(:,k-1) + (ref(:,k) - H * x_hat(:,k)) * Ts; % Integrate
error
    u(:,k) = Ki * n(:,k) + Kr * x_hat(:,k);
%     n(:,k) = n(:,k-1) + (ref(:,k) - H * x(:,k)) * Ts; % Integrate
error
%     u(:,k) = Ki * n(:,k) + Kr * x(:,k);
end


%% Results
fprintf('Linear speed RMSE:  %g m/s\n', sqrt(mean((x(1,:)-
x_hat(1,:)).^2)));
fprintf('Angular speed RMSE: %g rad/s\n\n', sqrt(mean((x(2,:)-
x_hat(2,:)).^2)));
```

```
% Plots
figure
subplot(2,1,1)
plot(t_axis,ref(1,:),'g', t_axis,y(1,:),'oc', t_axis,x(1,:),'b',
t_axis,x_hat(1,:),'r',t_axis(indice),y(1,indice),'+k');
title('Linear Speed')
xlabel('Time [s]')
ylabel('v [m/s]')
legend('Reference', 'Measurement', 'Real', 'Estimated','Vdisparado')

subplot(2,1,2)
plot(t_axis,integral,'b',t_axis,Um,'r')
xlabel('Time [s]')
ylabel('umbral')
legend('y(:,k)T*S*y(:,k))', 'Umbral')

figure
subplot(2,1,1)
plot(t_axis,ref(2,:),'g', t_axis,y(2,:),'oc', t_axis,x(2,:),'b',
t_axis,x_hat(2,:),'r',t_axis(indice),y(2,indice),'+k');
title('Angular Speed')
xlabel('Time [s]')
ylabel('\omega [rad/s]')
legend('Reference', 'Measurement', 'Real', 'Estimated','Wdisparado')

subplot(2,1,2)
plot(t_axis,integral,'b',t_axis,Um,'r')
xlabel('Time [s]')
ylabel('umbral')
legend('y(:,k)T*S*y(:,k))', 'Umbral')
```

Por el siguiente:

```
fprintf('Linear speed RMSE:  %g m/s\n', sqrt(mean((x(1,161:end)-
x_hat(1,161:end)).^2)));
fprintf('Angular speed RMSE: %g rad/s\n\n', sqrt(mean((x(2,161:end)-
x_hat(2,161:end)).^2)));

fprintf('Linear speed RMSE transitory:  %g m/s\n',
sqrt(mean((x(1,1:160)-x_hat(1,1:160)).^2)));
fprintf('Angular speed RMSE transitory: %g rad/s\n\n',
sqrt(mean((x(2,1:160)-x_hat(2,1:160)).^2)));
```

FK_Sim2

```
%% Set-up
clear all
%  --- Simulation parameters

Ts = 10e-3;
T_final = 100;
k_final = T_final/Ts;
t_axis=0:Ts:T_final-Ts;
j=0;
l=40;
```

```
indice=[];
num_disparos=0;
num_tr=0;
Qn=zeros(4,4);
% --- System parameters
A = 1.0e+03 *[
  -0.004094199291309  -0.000015025848246   1.663549680532014
0.000722682271388
  -0.000008063153776  -0.005041683869938   0.000326285936661
2.022770901637336
                   0                   0  -0.200000000000000
0
                   0                   0                   0  -
0.200000000000000];


B = [-4.158874201330035  -0.001806705678471
     -0.000815714841653  -5.056927254093339
                      1                   0
                      0                   1];

H = [eye(2) zeros(2, 2)];

Ki= [  0.999999739408795  0.000721929696507
      -0.000721929696770  0.999999739408544];

Kr= [ -0.259902871873893   0.000493748739433  -2.167013573124378
0.004017102111344
       0.000526013153705  -0.222222011238482   0.003996171390493  -
2.256249376742330];


[M, N] = size(H); % Number of outputs and states

% Discrete plant
F = expm(A*Ts);
G = A\(F - eye(N))*B;

% --- Sensors and Noise
% - System noise
sigma_w = [2e-4 6e-4 2e-4 6e-4]';
Q = diag(sigma_w.^2);

% - Sensor noise
sigma_v = [1.75734*10e-4 1.054404*10e-3]';
R = diag(sigma_v.^2);

% --- Initial conditions
x_0 = [0.2 0.2 0 0];
x_hat_0 = zeros(N,1);
P_0 = diag([0.1 0.6 0.1 0.6]);

% Reference speeds
```

```
%  - 1. linear speed
ref = zeros(2,k_final);
ref(1, (10/Ts):(25/Ts)) = 0.2;
ref(1, (25/Ts):(40/Ts)) = 0.4;
ref(1, (40/Ts):(50/Ts)) = 0.5;
ref(1, (50/Ts):(75/Ts)) = 0.4;
ref(1, (75/Ts):(90/Ts)) = 0.2;
%  - 2. linear speed
w_0 = 0.3;
ref(2, (30/Ts):round(35.02/Ts)) = -w_0;
ref(2, (55/Ts):round(60.02/Ts)) = -w_0;
ref(2, (65/Ts):round(70.02/Ts)) = w_0;

% Array initialization
x = zeros(N, k_final);      % Plant states
x(:,1) = x_0;

x_hat = x;                  % Estimated states
x_hat(:,1) = x_hat_0;

P = zeros(N,N,k_final);     % Estimation error covariance
P(:,:,1) = P_0;
Fn=F^(l);
for a=0:1:l-1
    Qn=Qn+F^a*Q*(F^a)';
end
[P_inf,Y,G_estatica]=dare(Fn',H',Qn,R);
L_inf=P_inf*H'/(H*P_inf*H'+R);
y = zeros(M, k_final)*NaN;  % Sensor measurements

u = zeros(2, k_final);      % Control signal (plat inputs)
n = zeros(2, k_final);      % Integrator output


%% Compute simulation

for k = 2:k_final
    % Apply plant model
    x(:,k) = F * x(:,k-1) + G * u(:,k-1) + sigma_w.*randn(N,1);

    % KF: prediction
    x_hat(:,k) = F * x_hat(:,k-1) + G * u(:,k-1);
    %P(:,:,k) = F * P(:,:,k-1) * F' + Q;
    P(:,:,k)=P_inf;

    % ---- Send-on-Delta ----

    % Sensor measurement
    y(:,k) = H * x(:,k) + sigma_v.*randn(M,1);
    j=j+1;

    if (j==l)
        j=0;
        if (k<=160)
            num_tr=num_tr+1;
        else
```

```
            num_disparos=num_disparos+1;
        end
        indice=cat(1,indice,k);
        % KF: correction
        %L = P(:,:,k) * H' / (H * P(:,:,k) * H' + R);    % Kalman gain
        x_hat(:,k) = x_hat(:,k) +  L_inf * (y(:,k) - H * x_hat(:,k));
        P(:,:,k) = (eye(N) - L_inf * H) * P(:,:,k) * (eye(N) - L_inf *
H)'+L_inf * R * L_inf';
    end
    % ----------------------

    % Compute next control signal
    n(:,k) = n(:,k-1) + (ref(:,k) - H * x_hat(:,k)) * Ts; % Integrate
error
    u(:,k) = Ki * n(:,k) + Kr * x_hat(:,k);
end


%% Results
fprintf('Linear speed RMSE:  %g m/s\n', sqrt(mean((x(1,161:end)-
x_hat(1,161:end)).^2)));
fprintf('Angular speed RMSE: %g rad/s\n\n', sqrt(mean((x(2,161:end)-
x_hat(2,161:end)).^2)));

fprintf('Linear speed RMSE transitory:  %g m/s\n',
sqrt(mean((x(1,1:160)-x_hat(1,1:160)).^2)));
fprintf('Angular speed RMSE transitory: %g rad/s\n\n',
sqrt(mean((x(2,1:160)-x_hat(2,1:160)).^2)));



% Plots
figure

subplot(2,1,1)
plot(t_axis,ref(1,:),'g', t_axis,y(1,:),'oc', t_axis,x(1,:),'b',
t_axis,x_hat(1,:),'r',t_axis(indice),y(1,indice),'+k');
title('Linear Speed')
xlabel('Time [s]')
ylabel('v [m/s]')
legend('Reference', 'Measurement', 'Real', 'Estimated','Vdisparado')

subplot(2,1,2)
semilogy(t_axis, abs(x(1,:) - y(1,:)),  'oc', ...
         t_axis, abs(x(1,:) - x_hat(1,:)),  'b',  ...
         t_axis, 2*squeeze(sqrt(P(1,1,:))), 'r')
xlabel('Time [s]')
ylabel('[m/s]')
legend('Measurement error', 'Estimation error', '2\sigma_v');

figure

subplot(2,1,1)
plot(t_axis,ref(2,:),'g', t_axis,y(2,:),'oc', t_axis,x(2,:),'b',
t_axis,x_hat(2,:),'r',t_axis(indice),y(2,indice),'+k');
title('Angular Speed')
xlabel('Time [s]')
```

```
ylabel('\omega [rad/s]')
legend('Reference', 'Measurement', 'Real', 'Estimated','Wdisparado')

subplot(2,1,2)
semilogy(t_axis, abs(x(2,:) - y(2,:)),  'oc', ...
         t_axis, abs(x(2,:)- x_hat(2,:)),   'b', ...
         t_axis, 2*squeeze(sqrt(P(2,2,:))), 'r')
xlabel('Time [s]')
ylabel('[rad/s]')
legend('Measurement error', 'Estimation error', '2\sigma_\omega');
```

SoD_Sim2

```
%% Set-up
clear all
%  --- Simulation parameters

Ts = 10e-3;
T_final = 100;
k_final = T_final/Ts;
t_axis=0:Ts:T_final-Ts;

Umbral=0.009;
num=0;
num_tr=0;

l=40;
i=1;
Um=repmat(Umbral,1,k_final);



indice=[];
Qn=zeros(4,4);
% --- System parameters
A = 1.0e+03 *[
  -0.004094199291309  -0.000015025848246   1.663549680532014
0.000722682271388
  -0.000008063153776  -0.005041683869938   0.000326285936661
2.022770901637336
                   0                   0  -0.200000000000000
0
                   0                   0                   0  -
0.200000000000000];


B = [-4.158874201330035  -0.001806705678471
     -0.000815714841653  -5.056927254093339
                      1                   0
                      0                   1];

H = [eye(2) zeros(2, 2)];

Ki= [  0.999999739408795  0.000721929696507
```

```
      -0.000721929696770   0.999999739408544];

Kr= [ -0.259902871873893    0.000493748739433   -2.167013573124378
0.004017102111344
       0.000526013153705   -0.222222011238482    0.003996171390493   -
2.256249376742330];


[M, N] = size(H); % Number of outputs and states

% Discrete plant
F = expm(A*Ts);
G = A\(F - eye(N))*B;

% --- Sensors and Noise
% - System noise
sigma_w = [2e-4 6e-4 2e-4 6e-4]';
Q = diag(sigma_w.^2);

% - Sensor noise
sigma_v = [1.75734*10e-4 1.054404*10e-3]';
R = diag(sigma_v.^2);

% --- Initial conditions
x_0 = [0.2 0.2 0 0];
x_hat_0 = zeros(N,1);
P_0 = diag([0.1 0.6 0.1 0.6]);

% Reference speeds
%  - 1. linear speed
ref = zeros(2,k_final);
ref(1, (10/Ts):(25/Ts)) = 0.2;
ref(1, (25/Ts):(40/Ts)) = 0.4;
ref(1, (40/Ts):(50/Ts)) = 0.5;
ref(1, (50/Ts):(75/Ts)) = 0.4;
ref(1, (75/Ts):(90/Ts)) = 0.2;
%  - 2. linear speed
w_0 = 0.3;
ref(2, (30/Ts):round(35.02/Ts)) = -w_0;
ref(2, (55/Ts):round(60.02/Ts)) = -w_0;
ref(2, (65/Ts):round(70.02/Ts)) = w_0;

% Array initialization
x = zeros(N, k_final);      % Plant states
x(:,1) = x_0;

x_hat = x;                  % Estimated states
x_hat(:,1) = x_hat_0;

P = zeros(N,N,k_final);     % Estimation error covariance
P(:,:,1) = P_0;
Fn=F^(l);
for a=0:1:l-1
    Qn=Qn+F^a*Q*(F^a)';
end
[P_inf,Y,G_estatica]=dare(Fn',H',Qn,R);
L_inf=P_inf*H'/(H*P_inf*H'+R);
```

```
y = zeros(M, k_final)*NaN;  % Sensor measurements

u = zeros(2, k_final);      % Control signal (plat inputs)
n = zeros(2, k_final);      % Integrator output
diferencia=zeros(1, k_final);
rest=zeros(2,k_final);
S=[1 0;0 1];


%% Compute simulation


for k = 2:k_final
    % Apply plant model
    x(:,k) = F * x(:,k-1) + G * u(:,k-1) + sigma_w.*randn(N,1);

    % KF: prediction
    x_hat(:,k) = F * x_hat(:,k-1) + G * u(:,k-1);
    %P(:,:,k) = F * P(:,:,k-1) * F' + Q;
    P(:,:,k)=P_inf;

    % ---- Send-on-Delta ----

    % Sensor measurement
    y(:,k) = H * x(:,k) + sigma_v.*randn(M,1);
    rest(:,k)=abs(y(:,k)-y(:,i));
    diferencia(k)=rest(:,k)'*S*rest(:,k);
    if (~isfinite(diferencia(k)) || diferencia(k)>Umbral)

        i=k;
        if (k<=160)
            num_tr=num_tr+1;
        else
        num=num+1;
        end
        indice=cat(1,indice,i);

        % KF: correction
        %L = P(:,:,k) * H' / (H * P(:,:,k) * H' + R);     % Kalman gain
        x_hat(:,k) = x_hat(:,k) +  L_inf * (y(:,k) - H * x_hat(:,k));
        P(:,:,k) = (eye(N) - L_inf * H) * P(:,:,k) * (eye(N) - L_inf *
H)'+L_inf * R * L_inf';
    end
    % ----------------------

    % Compute next control signal
    n(:,k) = n(:,k-1) + (ref(:,k) - H * x_hat(:,k)) * Ts; % Integrate
error
    u(:,k) = Ki * n(:,k) + Kr * x_hat(:,k);
end


%% Results
fprintf('Linear speed RMSE:  %g m/s\n', sqrt(mean((x(1,161:end)-
x_hat(1,161:end)).^2)));
fprintf('Angular speed RMSE: %g rad/s\n\n', sqrt(mean((x(2,161:end)-
x_hat(2,161:end)).^2)));
```

```
fprintf('Linear speed RMSE transitory:  %g m/s\n',
sqrt(mean((x(1,1:160)-x_hat(1,1:160)).^2)));
fprintf('Angular speed RMSE transitory: %g rad/s\n\n',
sqrt(mean((x(2,1:160)-x_hat(2,1:160)).^2)));


% Plots
figure

subplot(2,1,1)
plot(t_axis,ref(1,:),'g', t_axis,y(1,:),'oc', t_axis,x(1,:),'b',
t_axis,x_hat(1,:),'r',t_axis(indice),y(1,indice),'+k');
title('Linear Speed')
xlabel('Time [s]')
ylabel('v [m/s]')
legend('Reference', 'Measurement', 'Real', 'Estimated','Vdisparado')

subplot(2,1,2)
plot(t_axis,diferencia,'b',t_axis,Um,'r')
xlabel('Time [s]')
ylabel('umbral')
legend('y(:,k)T*S*y(:,k))', 'Umbral')

figure
subplot(2,1,1)
plot(t_axis,ref(2,:),'g', t_axis,y(2,:),'oc', t_axis,x(2,:),'b',
t_axis,x_hat(2,:),'r',t_axis(indice),y(2,indice),'+k');
title('Angular Speed')
xlabel('Time [s]')
ylabel('\omega [rad/s]')
legend('Reference', 'Measurement', 'Real', 'Estimated','Wdisparado')

subplot(2,1,2)
plot(t_axis,diferencia,'b',t_axis,Um,'r')
xlabel('Time [s]')
ylabel('umbral')
legend('y(:,k)T*S*y(:,k))', 'Umbral')
```

SoA_Sim2

```
%% Set-up
clear all
%  --- Simulation parameters

Ts = 10e-3;
T_final = 100;
k_final = T_final/Ts;
t_axis=0:Ts:T_final-Ts;

Umbral=1.5e-3;
num=0;
num_tr=0;

l=40;
```

```
i=1;
Um=repmat(Umbral,1,k_final);

indice=[];
Qn=zeros(4,4);
% --- System parameters
A = 1.0e+03 *[
  -0.004094199291309  -0.000015025848246   1.663549680532014
0.000722682271388
  -0.000008063153776  -0.005041683869938   0.000326285936661
2.022770901637336
                   0                   0  -0.200000000000000
0
                   0                   0                   0  -
0.200000000000000];


B = [-4.158874201330035  -0.001806705678471
     -0.000815714841653  -5.056927254093339
                      1                   0
                      0                   1];

H = [eye(2) zeros(2, 2)];

Ki= [  0.999999739408795  0.000721929696507
      -0.000721929696770  0.999999739408544];

Kr= [ -0.259902871873893   0.000493748739433  -2.167013573124378
0.004017102111344
       0.000526013153705  -0.222222011238482   0.003996171390493  -
2.256249376742330];


[M, N] = size(H); % Number of outputs and states

% Discrete plant
F = expm(A*Ts);
G = A\(F - eye(N))*B;

% --- Sensors and Noise
% - System noise
sigma_w = [2e-4 6e-4 2e-4 6e-4]';
Q = diag(sigma_w.^2);

% - Sensor noise
sigma_v = [1.75734*10e-4 1.054404*10e-3]';
R = diag(sigma_v.^2);

% --- Initial conditions
x_0 = [0.2 0.2 0 0];
x_hat_0 = zeros(N,1);
P_0 = diag([0.1 0.6 0.1 0.6]);

% Reference speeds
%  - 1. linear speed
ref = zeros(2,k_final);
ref(1, (10/Ts):(25/Ts)) = 0.2;
```

```
ref(1, (25/Ts):(40/Ts)) = 0.4;
ref(1, (40/Ts):(50/Ts)) = 0.5;
ref(1, (50/Ts):(75/Ts)) = 0.4;
ref(1, (75/Ts):(90/Ts)) = 0.2;
%  - 2. linear speed
w_0 = 0.3;
ref(2, (30/Ts):round(35.02/Ts)) = -w_0;
ref(2, (55/Ts):round(60.02/Ts)) = -w_0;
ref(2, (65/Ts):round(70.02/Ts)) = w_0;

% Array initialization
x = zeros(N, k_final);      % Plant states
x(:,1) = x_0;

x_hat = x;                  % Estimated states
x_hat(:,1) = x_hat_0;

P = zeros(N,N,k_final);     % Estimation error covariance
P(:,:,1) = P_0;
Fn=F^(l);
for a=0:1:l-1
    Qn=Qn+F^a*Q*(F^a)';
end
[P_inf,Y,G_estatica]=dare(Fn',H',Qn,R);
L_inf=P_inf*H'/(H*P_inf*H'+R);
y = zeros(M, k_final)*NaN;  % Sensor measurements


u = zeros(2, k_final);      % Control signal (plat inputs)
n = zeros(2, k_final);      % Integrator output
integral=zeros(1, k_final);
m=zeros(M,k_final);
rest=zeros(M,k_final);
S=[1 0;0 1];

%% Compute simulation

for k = 2:k_final

    % Apply plant model
    x(:,k) = F * x(:,k-1) + G * u(:,k-1) + sigma_w.*randn(N,1);

    % KF: prediction
    x_hat(:,k) = F * x_hat(:,k-1) + G * u(:,k-1);
    %P(:,:,k) = F * P(:,:,k-1) * F' + Q;
    P(:,:,k)=P_inf;

    % ---- Send-on-Delta ----

    % Sensor measurement
    y(:,k) = H * x(:,k) + sigma_v.*randn(M,1);
    for a=i:k
        rest(:,a)=abs(y(:,a)-y(:,i));
        m(a)=rest(:,a)'*S*rest(:,a);

    end
```

```
    t=0.01*(i:k);

    integral(k)=trapz(t,m(i:k));

    if (~isfinite(integral(k)) || integral(k)>Umbral)
        i=k;

        if (k<=160)
            num_tr=num_tr+1;
        else
        num=num+1;
        end

        indice=cat(1,indice,i);

        % KF: correction
        %L = P(:,:,k) * H' / (H * P(:,:,k) * H' + R);     % Kalman gain
        x_hat(:,k) = x_hat(:,k) +  L_inf * (y(:,k) - H * x_hat(:,k));
        P(:,:,k) = (eye(N) - L_inf * H) * P(:,:,k) * (eye(N) - L_inf *
H)'+L_inf * R * L_inf';
    end
    % ----------------------

    % Compute next control signal
    n(:,k) = n(:,k-1) + (ref(:,k) - H * x_hat(:,k)) * Ts; % Integrate
error
    u(:,k) = Ki * n(:,k) + Kr * x_hat(:,k);
%     n(:,k) = n(:,k-1) + (ref(:,k) - H * x(:,k)) * Ts; % Integrate
error
%     u(:,k) = Ki * n(:,k) + Kr * x(:,k);
end


%% Results
fprintf('Linear speed RMSE:  %g m/s\n', sqrt(mean((x(1,161:end)-
x_hat(1,161:end)).^2)));
fprintf('Angular speed RMSE: %g rad/s\n\n', sqrt(mean((x(2,161:end)-
x_hat(2,161:end)).^2)));

fprintf('Linear speed RMSE transitory:  %g m/s\n',
sqrt(mean((x(1,1:160)-x_hat(1,1:160)).^2)));
fprintf('Angular speed RMSE transitory: %g rad/s\n\n',
sqrt(mean((x(2,1:160)-x_hat(2,1:160)).^2)));


% Plots
figure
subplot(2,1,1)
plot(t_axis,ref(1,:),'g', t_axis,y(1,:),'oc', t_axis,x(1,:),'b',
t_axis,x_hat(1,:),'r',t_axis(indice),y(1,indice),'+k');
title('Linear Speed')
xlabel('Time [s]')
ylabel('v [m/s]')
legend('Reference', 'Measurement', 'Real', 'Estimated','Vdisparado')

subplot(2,1,2)
plot(t_axis,integral,'b',t_axis,Um,'r')
```

```
xlabel('Time [s]')
ylabel('umbral')
legend('y(:,k)T*S*y(:,k))', 'Umbral')

figure
subplot(2,1,1)
plot(t_axis,ref(2,:),'g', t_axis,y(2,:),'oc', t_axis,x(2,:),'b',
t_axis,x_hat(2,:),'r',t_axis(indice),y(2,indice),'+k');
title('Angular Speed')
xlabel('Time [s]')
ylabel('\omega [rad/s]')
legend('Reference', 'Measurement', 'Real', 'Estimated','Wdisparado')

subplot(2,1,2)
plot(t_axis,integral,'b',t_axis,Um,'r')
xlabel('Time [s]')
ylabel('umbral')
legend('y(:,k)T*S*y(:,k))', 'Umbral')
```

Índice de gráficas

Bibliografía

[1] Katsuhiko Ogata, "Ingeniería de control moderna",3ª edición, PEARSON.

[2] Arthur G.O. Mutambara, "Decentralized Estimation and Control for Multisensor Systems" CRC press 1998.

[3] Dan Simon, "Optimal State Estimation: Kalman, H∞, and Nonlinear Approaches", WILEY-INTERSCIENCE 2006.

[4] Joris Sijs, Mircea Lazar, "On Event Based State Estimation" Springer-Verlag Berlin Heidelberg, pp 336-350, 2009.

[5] Hespanha, J., Naghshtabrizi, P., Xu, Y. "A survey of recent results in networked control systems". Proceedings of the IEEE 95(1) (2007) 138–162.

[6] Miskowicz, M. "Send-on-delta concept: An event-based data reporting strategy", Sensors 6(1) (2006) 49–63

[7] Mallick, M., Coraluppi, S., Carthel, C. "Advances in asynchronous and decentralized estimation", In: Aerospace Conference, 2001, IEEE Proceedings. Volume 4.

[8] http://www.todomotores.cl/mecanica/el_motor.htm

[9] Claude E. Shannon, Warren Weaver, "The mathematical theory of communication", University of Illinois Press 1963.

[10] Bhaskar Krishnamachari, "Networking Wireless Sensors", Cambridge University Press, 2005.

[11] S. Fernández, J.M. Cordero, A. Córdoba, "Estadística Descriptiva", ESIC Editorial.

[12] Miskowicz, M. "Asymptotic Effectiveness of the Event-Based Sampling according to the Integral Criterion" ,Sensors 2007,7,16-37

[13] http://www.mobilerobots.com/Libraries/Downloads/Pioneer3DX-P3DX-RevA.sflb.ashx

[14] Carlos Santos, Manuel Mazo Jr., and Felipe Espinosa, "Adaptive Self-triggered Control of a Remotely Operated Robot "

[15] Carlos Santos, "Propuesta de control descentralizado con solapamiento para guiado en convoy de unidades P3-DX" 2010.