Tecnológico de Costa Rica

Escuela de Ingeniería Electrónica

Programa de Licenciatura en Ingeniería Electrónica

Desarrollo de un conjunto de flujos de trabajo para la implementación de software a bordo de computadoras de guía, navegación y control espacial

Informe de Trabajo Final de Graduación para optar por el título de

Ingeniero en Electrónica con el grado académico de Licenciatura

David Duarte Sánchez

Borrador de 12 de septiembre de 2024

El documento Requisitos para la entrega de Trabajos Finales de Graduación a las bibliotecas del TEC indica que usted debe incluir la licencia de Creative Commons en la página siguiente de la portada.

Asegúrse entonces de elegir la licencia correcta, y ajustar el texto abajo a su selección.

Es necesario que descargue el ícono correcto en formato vectorial, y lo coloque en el directorio `fig/`.

Declaro que el presente documento de tesis ha sido realizado enteramente por mi persona, utilizando y aplicando literatura referente al tema e introduciendo conocimientos y resultados experimentales propios.

En los casos en que he utilizado bibliografía he procedido a indicar las fuentes mediante las respectivas citas bibliográficas. En consecuencia, asumo la responsabilidad total por el trabajo de tesis realizado y por el contenido del presente documento.

David Duarte Sánchez

Cartago, 12 de septiembre de 2024

Céd: 3-0507-0982

Instituto Tecnológico de Costa Rica
Escuela de Ingeniería Electrónica
Trabajo Final de Graduación
Acta de Aprobación

Defensa de Trabajo Final de Graduación
Requisito para optar por el título de Ingeniero en Electrónica
Grado Académico de Licenciatura

El Tribunal Evaluador aprueba la defensa del trabajo final de graduación denominado *Desarrollo de un conjunto de flujos de trabajo para la implementación de software a bordo de computadoras de guía, navegación y control espacial*, realizado por el señor David Duarte Sánchez y, hace constar que cumple con las normas establecidas por la Escuela de Ingeniería Electrónica del Instituto Tecnológico de Costa Rica.

Miembros del Tribunal Evaluador

Dr. Alfonso Chaves Jiménez Profesor Lector	Ing. William Marín Moreno Profesor Lector

Dr. Johan Carvajal Godínez
Profesor Asesor

Cartago, 12 de septiembre de 2024

Instituto Tecnológico de Costa Rica
Escuela de Ingeniería Electrónica
Trabajo Final de Graduación
Tribunal Evaluador
Acta de Evaluación

Defensa del Trabajo Final de Graduación
Requisito para optar por el título de Ingeniero en Electrónica
Grado Académico de Licenciatura

Estudiante: **David Duarte Sánchez** Carné: 2017239606

Nombre del proyecto: *Desarrollo de un conjunto de flujos de trabajo para la implementación de software a bordo de computadoras de guía, navegación y control espacial*

Los miembros de este Tribunal hacen constar que este trabajo final de graduación ha sido aprobado y cumple con las normas establecidas por la Escuela de Ingeniería Electrónica del Instituto Tecnológico de Costa Rica y es merecedor de la siguiente calificación:

Nota del Trabajo Final de Graduación: _____

Miembros del Tribunal Evaluador

Dr. Alfonso Chaves Jiménez Profesor Lector	Ing. William Marín Moreno Profesor Lector

Dr. Johan Carvajal Godínez
Profesor Asesor

Cartago, 12 de septiembre de 2024

Resumen

El resumen es la síntesis de lo que aparece en el resto del documento. Tiene que ser lo suficientemente conciso y claro para que alguien que lo lea sepa qué esperar del resto del trabajo, y se motive para leerla completamente. Usualmente resume lo más relevante de la introducción y contiene la conclusión más importante del trabajo.

Es usual agregar palabras clave, que son los temas principales tratados en el documento. El resumen queda fuera de la numeración del resto de secciones.

Evite utilizar referencias bibliográficas, tablas, o figuras en el resumen.

Palabras clave: GNC, Sistemas, procesadores embebidos, marcos de trabajo, model to model transofrmation, codigo embebido

Abstract

Same content as the Spanish version, just in English. Check this site for some help with the translation. For instance, the following is the automatic translation from a previous version of the "Resumen".

The abstract is the synthesis of what appears in the rest of the document. It has to be concise and clear enough so that someone reading it knows what to expect from the rest of the text, and is motivated to read it in full. It usually summarizes the most relevant parts of the introduction and contains the most important conclusion of the work.

It is usual to add keywords, which are the main topics covered in the document. The abstract is left out of the numbering of the rest of the sections.

Avoid using bibliographical references, tables, or figures in the abstract.

Keywords: word 1, word 2,

a mis queridos padres

Agradecimientos

El resultado de este trabajo no hubiese sido posible sin el apoyo de Thevenin, Norton, Einstein y mi querido amigo Ohm.

Usualmente se agradece aquí a la empresa o investigador que dio la oportunidad de realizar el trabajo final de graduación.

No debe confundir el agradecimiento con la dedicatoria. La dedicatoria es usualmente una sola línea, con la persona a quien se dedica el trabajo.

El agradecimiento es un texto más elaborado, de caracter personal, en donde se expresa la gratitud por la oportunidad, el apoyo brindado, la inspiración ofrecida, el acompañamiento moral, etc.

David Duarte Sánchez

Cartago, 12 de septiembre de 2024

Índice general

Índice de figuras

Índice de tablas

Revisar

Capítulo 1

Introducción

1.1. Proceso de diseño de los sistemas de Guía, Navegación y Control espacial

La implementacion de sistemas GNC en sistemas embebidos, conlleva una combinacion de hardware y software especializado, por un lado, los microcontroladores son los encargados de gestionar los clculos, mientras que los sensores proporcionan distintos tipos de datos por medio de las entradas. Por otro lado, los sistemas en tiempo real garantizan la respuesta en el momento requerido. Las aplicaciones de estos se pueden observar en drones, satelites y sondas espaciales [4].

Como se menciono anteriormente los sistemas GNC son fundamentales en las misiones espaciales, estan encargados de determinar la trayectoria optima para cumplir los objetivos de la mision, ademas de calcular la secuencia de maniobras necesarias, determinar la posicion, velocidad y orientacion, tambien se encargan de aplicar las acciones correctivas necesarias para mantener la trayectoria [15].

1.1.1. Requerimientos de los sistemas

Los requerimientos de los sistemas GNC incluyen: precision para determinar la posicion y orientacion del vehiculo con gran exactitud, robustez para funcionar de manera confiable y tolerar fallos o perturbaciones generadas por el entorno, autonomia para poder operar sin depender de la intervencion humana, flexibilidad para adaptarse a diferentes fases de la mision y un bajo consumo de potencia para minimizar el uso de los recursos limitados a bordo.

Para cumplir con los requerimientos mencionados anteriormente se debe definir con precision los requerimientos del sistema, como la precision necesaria para determinar la posicion del vehiculo, la robustez del sistema para resistir fallos, las restricciones energeticas y de recursos computacionales a bordo. Una vez solventados estos requerimientos el sistema

se plantea bajo una arquitectura modular la cual divide el sistema en bloques independientes para las funciones de guia, navegacion y control, facilitando el desarrollo, prueba y mantenimiento [1].

1.2. Sistemas embebidos para los sistemas GNC

El uso de sistemas embebidos ha transformado la navegación y el control aeroespacial. Estos sistemas integran hardware y software, permitiendo el procesamiento de datos en tiempo real, fundamental para la navegación precisa y el control de vuelo. Los sistemas embebidos gestionan sensores que recopilan información sobre altitud, velocidad y posición, permitiendo a pilotos y sistemas automáticos tomar decisiones rápidas y fundamentadas. Esta capacidad de respuesta es esencial en entornos cambiantes, como la aviación o el lanzamiento de cohetes.

Además, los sistemas embebidos facilitan la integración de múltiples funciones en un solo dispositivo, reduciendo el peso y volumen de los equipos a bordo, un factor crucial en la industria aeroespacial. Por ejemplo, en los sistemas de control de vuelo, los microcontroladores y procesadores embebidos pueden gestionar desde la navegación hasta la comunicación y el monitoreo de sistemas críticos, todo desde una única unidad. Esta integración mejora la eficiencia del espacio y minimiza la posibilidad de fallos al reducir el número de componentes individuales que podrían fallar.

Finalmente, la implementación de sistemas embebidos ha permitido avances significativos en la automatización y la inteligencia artificial aeroespacial. Los algoritmos embebidos procesan datos de manera eficiente, permitiendo la navegación autónoma y el control de vehículos sin intervención humana, especialmente relevante en misiones espaciales con comunicación limitada. Los sistemas embebidos mejoran la seguridad y eficiencia de las operaciones aeroespaciales, abriendo nuevas posibilidades para la exploración y el desarrollo de tecnologías futuras en este campo.

1.2.1. Marco de Trabajo Yocto

El Yocto Project es una iniciativa de código abierto que proporciona un conjunto de herramientas y recursos para crear sistemas operativos Linux personalizados, especialmente diseñados para dispositivos embebidos. Su objetivo principal es facilitar el desarrollo de software y la integración de componentes en una amplia variedad de hardware, permitiendo a los desarrolladores construir imágenes de sistema adaptadas a sus necesidades específicas. Utiliza BitBake, una herramienta que permite definir recetas para la construcción y empaquetado de software, lo que otorga gran flexibilidad y personalización. Además, soporta múltiples arquitecturas de hardware, como ARM, x86, MIPS y PowerPC, lo que lo hace adecuado para diferentes dispositivos, desde microcontroladores hasta sistemas más complejos. Al fomentar la reutilización de componentes y contar con una comunidad

activa que contribuye con mejoras y documentación, el Yocto Project se convierte en una solución ideal para el desarrollo de sistemas operativos en aplicaciones de IoT, electrónica de consumo y automatización industrial.

1.3. Donde se ubica dentro del flujo de control

Diagrama profe Johan (ver en proxima reunion)

1.4. Hardware en el loop

El Hardware en el loop es una tecnica fundamental en el desarrollo de sistemas GNC, ya que, permite simular el comportamiento del hardware en tiempo real, facilitando para los desarrolla-

dores la prueba y validacion del software sin requerir el hardware fisico, es esta forma permite probar de forma exhaustiva el software asegurando el funcionamiento del hardware simulado y permite la validacion de todo el sistema antes de su implementacion final. Esta implementacion genera una mayor precision en las pruebas, la posibilidad de validar la autonomia del sistema, ademas de reducir significativamente el tiempo y los costos de implementacion y prueba del hardware. En resumen, es una tecnica esencial en el desarrollo de sistemas GNC espaciales, per- mitiendo una validacion integral y eficiente de estos complejos sistemas antes de su despliegue en misiones reales [19] [21].

1.5. Objetivos y estructura del documento

El objetivo principal de este proyecto es desarrollar un conjunto de flujos de trabajo para la implementacion de software a bordo de computadoras de guia, navegacion y control espacial. Para lograr este objetivo se persiguen tres objetivos especificos. El primero consiste en Identificar una plataforma de hardware para el desarrollo de un modelo de ingenieria de una computadora de navegacion espacial.

El segundo se encarga de Establecer flujos de trabajo para el prototipado de algoritmos de control de orientacion y navegacion para aplicaciones espaciales con hardware en el loop. Y por ultimo el tecero consiste en Evaluar los casos de uso de una computadora de navegacion y control espacial mediante la implementacion de una aplicacion de referencia demostrativa (caso de la IMU)

Este documento incluye lo siguiente: en el capitulo 2 se presenta el marco teorico, donde se esbozan los fundamentos de la propuesta realizada. En el capitulo 3 se detalla la solucion propuesta y el modelo implementado para la solucion del problema. En el capitulo 4 se presentan los resultados obtenidos. Por ultimo, el capitulo 5 se presenta las conclusiones

de la investigacion y trabajo realizado, asi como recomendaciones y trabajo a futuro por desarrollar.

Capítulo 2

Marco teórico

En este capítulo se presentan los conceptos teóricos que subyacen la propuesta de desarrollo de un conjunto de flujos de trabajo para la implementación de software a bordo de computadoras de guía, navegación y control espacial. La información expuesta se deriva tanto de conocimientos propios como información bibliográfica.

2.1. Estimación

La estimación implica el uso de modelos matemáticos y algoritmos para calcular las variables de estado del sistema. Estas variables son esenciales para comprender el comportamiento del sistema y para tomar decisiones informadas sobre su control. La estimación puede realizarse de dos maneras:

- Lazo abierto: En este enfoque, se utilizan modelos de estimación predefinidos sin retroalimentación, lo que significa que las estimaciones no se ajustan en función de las mediciones reales.
- Lazo cerrado: Este método ajusta las estimaciones en función de las mediciones reales y las salidas del sistema, lo que permite una mayor precisión y adaptabilidad.

Esta es crucial en aplicaciones donde las mediciones directas son difíciles o costosas de obtener, por ejemplo en los sistemas hidráulicos, la estimación de variables de estado permite optimiza el rendimiento y la eficiencia del sistema, asegurando que se mantengan las condiciones deseadas a pesar de las perturbaciones externas o errores en las mediciones [1]. La estimación es un componente clave en los sistemas de control, ya que facilita la comprensión y el manejo de sistemas complejos. Su implementación permite una operación más eficiente y efectiva, mejorando su capacidad de respuesta ante diversa condiciones operativas [2].

2.2. Control

Como se mencionó anteriormente la estimación es un componente clave en los sistemas de control, ya que este se enfoca en el desarrollo y diseño de sistemas capaces de regular y controlar variables de un proceso de manera autónoma. Estos sistemas utilizan sensores, actuadores y algoritmos de control para mantener las variables de interés dentro de los rangos permitidos, mejorando de esta forma la eficiencia, precisión y confiabilidad de los procesos. Su aplicación abarca desde sistemas espaciales hasta biorreactores y sistemas de iluminación.

2.3. Procesadores embebidos

Los procesadores embebidos son microprocesadores especializados en tareas dentro de un sistema más complejo. A diferencia de los procesadores de propósito general, estos están optimizados para ofrecer eficiencia energética, un tamaño compacto y costo reducido. Algunas de las características de los procesadores embebidos se presentan a continuación:

- Integración de periféricos: Incorporan periféricos específicos de la aplicación en un único chip, incluyendo temporizadores, puertos de entrada/salida y controladores de memoria.
- Arquitecturas de bajo Consumo: Diseñados para maximizar la duración de la batería en dispositivos portátiles, lo que es esencial para la operatividad de dispositivos móviles.
- Tamaño compacto: Su diseño permite reducir costos y facilitar la integración en espacios limitados, lo que los hace ideales para aplicaciones donde el espacio es crítico.
- Capacidad de respuesta en tiempo real: Pueden responder a eventos externos de manera predecible y determinista, lo que es crucial en aplicaciones que requieren una respuesta rápida y precisa.

2.3.1. Cortex-A9

Los procesadores embebidos basados en la arquitectura ARM Cortex-A9 se utilizan en aplicaciones de alto rendimiento y capacidades avanzadas de procesamiento. Aunque esta arquitectura no es un procesador embebido, sino más bien una familia de núcleos de procesador diseñado por ARM Holdings, los SoC que incorporan estos núcleos han demostrado ser una solución popular para aplicaciones embebidas [3]. Algunas de sus características son :

- Arquitectura de 32 bits basada en ARMv7-A.
- Alto rendimiento adecuado para aplicaciones exigentes como sistemas operativos embebidos, procesamiento multimedia y gráficos.
- Características avanzadas como unidades de coma flotante, unidades de procesamiento NEON para procesamiento multimedia y soporte para virtualización.

Algunos SoC que incorporan núcleos Cortex-A9 son:

- Nvidia Tegra 3: Combina cuatro núcleos Cortex-A9 y una GPU.
- Texas Instruments OMAP 4: Familia de SoC que combina núcleos Cortex-A9 y DSP.
- Xilinx Zynq-7000: Integra núcleos Cortex-A9 con lógica programable FPGA.

2.3.2. Tarjeta de desarrollo ZedBoard

La ZedBoard es una tarjeta de desarrollo basada en el Xilinx Zynq-7000 que como se mencionó anteriormente integra núcleos Cortex-A9 con la lógica programable para Field Programmable Gate Array (FPGA). Esta plataforma es ideal para prototipar aplicaciones en el ámbito de sistemas embebidos. La tabla 2.1 resume las especificaciones que posee la tarjeta de desarrollo ZedBoard.

Tabla 2.1: Especificaciones generales de la tarjeta de desarrollo ZeadBoard

Especificación	Detalles
Procesador	Xilinx Zynq-7000 (XC7Z020)
Núcleos de Procesador	ARM Cortex-A9 de doble núcleo
Memoria DDR3	512 MB
Memoria Flash	256 MB QSPI
Almacenamiento	Tarjeta SD de 4 GB
Conectividad	Ethernet (10/100/1000 Mbps), USB OTG 2.0, USB-UART
Salidas de Video	HDMI (1080p), VGA de 8 bits, OLED 128x32
Audio	Códec de audio I2S
Puertos GPIO	54 pines GPIO
Interfaz de JTAG	Soporte para programación y depuración
Dimensiones	10.2 cm x 6.4 cm
Fuente de Alimentación	5V a través de conector de alimentación
Sistema Operativo	Soporte para Linux y otros sistemas embebidos
Expansión	Conectores Pmod y FMC para módulos adicionales

2.4. Marcos de trabajo

Los marcos de trabajo en sistemas embebidos son conjuntos de herramientas y bibliotecas que facilitan el desarrollo de aplicaciones en estos sistemas. Estos proporcionan una estructura que permite abordar los desafíos específicos que presentan los sistemas embebidos.

Los sistemas embebidos interactúan con su entorno físico, lo que requiere un diseño que no solo considere los resultados de las operaciones, sino también el cumplimiento de plazos y restricciones específicas. En este contexto, las propiedades no funcionales, como el consumo energético, la latencia, la fiabilidad y el manejo de recursos, son críticas para el diseño y optimización del rendimiento general del sistema [4]. Los frameworks juegan un papel fundamental al proporcionar herramientas y bibliotecas predefinidas, permitiendo a los desarrolladores centrarse en la lógica de la aplicación en lugar de lidiar con los detalles de bajo nivel del hardware, lo que acelera el proceso de desarrollo y reduce la posibilidad de errores. Ejemplos de frameworks populares en sistemas embebidos incluyen Robot Operating System (ROS), utilizado en aplicaciones de robótica, y FreeRTOS, un sistema operativo de tiempo real diseñado para microcontroladores y sistemas embebidos [5].

2.4.1. YOCTO

Yocto es un marco de trabajo (framework) popular utilizado en el desarrollo de sistemas embebidos, especialmente en la creación de distribuciones de Linux personalizadas para hardware específico. Yocto utiliza un proceso de construcción cruzada, lo que significa que el código se compila en una plataforma diferente a la que se ejecutará, permitiendo que el código se optimice para el hardware específico del sistema embebido [6].

Una de las principales ventajas de Yocto es su flexibilidad en la configuración del sistema, permitiendo a los desarrolladores seleccionar paquetes específicos, configurar opciones de compilación y personalizar el sistema operativo según sus necesidades. Además, Yocto fomenta la reutilización de código a través de capas, que son colecciones de recetas, configuraciones y parches que se pueden agregar o eliminar fácilmente del flujo de trabajo de construcción [6].

2.5. Transformación de modelo a modelo

La transformación de modelo a modelo se refiere a un proceso en el que un modelo se convierte en otro, manteniendo la esencia de su estructura y funcionalidad, pero adaptándose a nuevas necesidades o contextos. Este concepto es fundamental en la Ingeniería de Software, especialmente dentro de la Arquitectura Dirigida por Modelos (MDA), donde se busca facilitar la interoperabilidad y la portabilidad de sistemas a través de la transformación de modelos independientes de la computación (CIM) a modelos independientes

de la plataforma (PIM) y viceversa.

- Modelos de Datos a Modelos de Aplicación:
- Modelos de Negocio a Modelos de Implementación
- Modelos UML a Código Fuente

Para efectos de este trabajo el área de interés serán la transformación de UML a Código Fuente.

2.5.1. MATLAB Embedded Coder

El MATLAB Embedded Coder se adapta a esta definición de transformación de modelo a modelo, ya que permite a los usuarios generar código C y C++ a partir de modelos Simulink. Esto es especialmente útil en el desarrollo de sistemas embebidos, donde se requiere que los modelos de alto nivel se transformen en código que pueda ser ejecutado en hardware específico. Esta herramienta facilita la implementación de algoritmos y sistemas de control, asegurando que el modelo original se traduzca eficazmente en un formato que pueda ser utilizado en entornos de producción.

2.6. Código embebido

El código embebido se refiere a un tipo de software diseñado para operar en dispositivos con recursos limitados, como microcontroladores y sistemas embebidos. Este código es fundamental en la programación de dispositivos electrónicos, permitiendo que estos realicen tareas específicas, como gestionar un sistema de automatización industrial o incluso operar en dispositivos móviles. Se caracteriza por su ejecución en dispositivos con recursos limitados, su capacidad para controlar dispositivos electrónicos, el uso de lenguajes de bajo nivel, la optimización de recursos y la necesidad de garantizar tiempos de respuesta determinísticos.

2.7. Revisión literaria

En los últimos 4 años, las computadoras de guía, navegación y control han mostrado grandes avances en el desarrollo de sistemas autónomos.

2.7.1. Desarrollo de sistemas de navegación

En 2022, se presentó un sistema de planificación y control de navegación para vehículos autónomos en entornos urbanos. Este sistema permite la planificación de rutas basadas en

la posición actual del vehículo y su destino, utilizando un controlador clásico que asegura el seguimiento de la trayectoria mediante odometría y correcciones visuales. Los resultados se simularon utilizando herramientas como ROS y Gazebo, lo que demuestra la viabilidad de estos sistemas en entornos complejos [7].

2.7.2. Transformación de Lenguaje de Bloques a Código C

La traducción de código de control de lenguaje de bloques a C implica un proceso de conversión donde cada bloque visual se asocia con una estructura de código en C. Esto se puede hacer utilizando herramientas de software que generan automáticamente el código C a partir de la lógica definida en el entorno de bloques. Este proceso no solo facilita la programación, sino que también permite la optimización del código generado para mejorar el rendimiento en sistemas de navegación autónoma.

XOD

XOD es un entorno de programación visual basado en bloques que permite a los usuarios crear programas para microcontroladores como Arduino. Este software genera automáticamente código en C++ a partir de la lógica definida en bloques. Los usuarios pueden conectar componentes gráficamente y, al finalizar, acceder al código generado, que es abierto y personalizable. XOD es gratuito y permite la creación de nuevos nodos para componentes específicos, lo que facilita la adaptación a diferentes proyectos [8].

Visual Microcontroller

Este software proporciona un lenguaje de programación gráfico para microcontroladores, desarrollado en C#. Utiliza una interfaz gráfica que permite a los usuarios diseñar diagramas que representan la lógica de control. El sistema compila el código a partir de diagramas gráficos, generando código intermedio en C antes de llegar al código hexadecimal necesario para la programación del microcontrolador [9].

LabVIEW

LabVIEW es un entorno de desarrollo que utiliza un enfoque gráfico para la programación. Aunque es más conocido en el ámbito de la ingeniería, también permite la generación de código en C. LabVIEW facilita la creación de aplicaciones de control y adquisición de datos, y su capacidad para traducir diagramas de bloques a código C lo convierte en una opción útil para proyectos que requieren un control preciso de hardware.

Simulink

Como se mencionó anteriormente, Simulink, parte de MATLAB, proporciona un entorno gráfico para modelar, simular y analizar sistemas dinámicos. Permite a los usuarios crear modelos utilizando bloques y, posteriormente, generar código C automáticamente a partir de estos modelos. Esta herramienta es especialmente valiosa en aplicaciones de ingeniería donde se requiere un alto grado de precisión y control sobre el comportamiento del sistema.

2.8. Avances recientes en GNCs

En el marco del proyecto EROSS+ (European Robotic Orbital Support Services), se ha trabajado en el diseño de un sistema GNC altamente autónomo para misiones de servicio robótico en órbita. Este proyecto, que abarca desde 2021 hasta 2023, busca integrar técnicas avanzadas de navegación visual y control de cumplimiento para la captura y manipulación de satélites, mostrando un enfoque en la autonomía y la eficiencia operativa [10].

Otro desarrollo notable es el programa de NASA sobre GNC autónomo, que incluye sistemas para el transbordador espacial. Este programa se centra en la optimización de trayectorias de vuelo y la adaptación de sistemas GNC para diferentes condiciones de vuelo, lo que demuestra la importancia de la flexibilidad en el diseño de estos sistemas [11].

Además, la actividad VV4RTOS, apoyada por la Agencia Espacial Europea, se ha centrado en la verificación y validación de sistemas de control basados en optimización. Esto incluye el desarrollo de software GNC en tiempo real, lo que permite una validación más efectiva y segura de los sistemas diseñados [12].

2.8.1. Programación de Sistemas GNC

Los lenguajes de bloques, como Simulink, son comúnmente utilizados para diseñar y simular sistemas de control. Estos lenguajes permiten a los ingenieros visualizar el flujo de datos y las interacciones entre componentes de manera intuitiva. Sin embargo, la necesidad de traducir estos modelos a código C es crucial para su implementación en hardware real.

A pesar de los avances, existen desafíos significativos en la implementación de sistemas GNC. La variabilidad en los entornos operativos y la necesidad de adaptarse a condiciones cambiantes requieren algoritmos robustos y adaptativos. La optimización de estos sistemas es fundamental para asegurar su efectividad en misiones críticas.

Un estudio reciente sobre el sistema CubeNav destaca la importancia de desarrollar herramientas de análisis de navegación que faciliten las operaciones de GNC en misiones

de CubeSats. Este enfoque busca reducir la curva de aprendizaje y minimizar errores humanos, lo que es esencial para misiones de bajo presupuesto y alta complejidad [12].

Capítulo 3

Solución propuesta

Primero que todo, jamás utilice el título indicado arriba, sino algo relacionado con su solución: "Sistema de corrección de distorsión" o lo que competa a su tesis en particular.

Este capítulo puede separarse en varias secciones, dependiendo del problema concreto. Aquí los algoritmos o el diseño del sistema deben quedar lo suficientemente claros para que otra persona pueda re-implementar al sistema propuesto. Sin embargo, el enfoque no debe nunca concentrarse en los detalles de la implementación particular realizada, sino del diseño conceptual como tal.

Recuerdese que toda tabla y figura debe estar referenciada en el texto.

Capítulo 4

Resultados y análisis

En este capítulo se exponen los diseños experimentales realizados para comprobar el funcionamiento correcto del sistema. Por ejemplo, si se realiza algún sistema con reconocimiento de patrones, usualmente esta sección involucra las llamadas *matrices de confusión* donde se compactan las estadísticas de reconocimiento alcanzadas. En circuitos de hardware, experimentos para determinar variaciones contra ruido, etc. También pueden ilustrarse algunos resultados concretos como ejemplo del funcionamiento de los algoritmos. Puede mostrar por medio de experimentos ventajas, desventajas, desempeño de su algoritmo, o comparaciones con otros algoritmos.

Recuerde que debe minimizar los "saltos" que el lector deba hacer en su documento. Por tanto, usualmente el análisis se coloca junto a tablasy figuras presentadas, y debe tener un orden de tal modo que se observe cómo los objetivos específicos y el objetivo general del proyecto de tesis se han cumplido.

Capítulo 5

Conclusiones

Las conclusiones no son un resumen de lo realizado sino a lo que ha llevado el desarrollo de la tesis, no perdiendo de vista los objetivos planteados desde el principio y los resultados obtenidos. En otras palabras, qué se concluye o a qué se ha llegado después de realizado la tesis de maestría. Un error común es "concluir" aspectos que no se desarrollaron en la tesis, como observaciones o afirmaciones derivadas de la teoría directamente. Esto último debe evitarse.

Es fundamental en este capítulo hacer énfasis y puntualizar los aportes específicos del trabajo.

Es usual concluir con lo que queda por hacer, o sugerencias para mejorar los resultados.

Bibliografía

[1] J. S. G. Merchán, M. F. Rodríguez, G. J. C. Méndez y M. F. H. Morales, «Evaluación de modelos aproximados para el diseño de control automático en sistemas de riego a canal abierto,» 2019. dirección: https://api.semanticscholar.org/CorpusID:230388188.

[2] F. Mesa, R. Ospina-Ospina y G. Correa-Vélez, «Estimación de variables de estado (LA y LC) en sistemas de control,» *Revista UIS Ingenierías*, 2020. dirección: https://api.semanticscholar.org/CorpusID:228853996.

[3] F. Schwiegelshohn y M. Hübner, «Design of an attention detection system on the Zynq-7000 SoC,» *2014 International Conference on ReConFigurable Computing and FPGAs (ReConFig14)*, págs. 1-6, 2014. dirección: https://api.semanticscholar.org/CorpusID:18920120.

[4] P. G. de Aledo Marugán, «Simulación y verificación de propiedades no-funcionales para sistemas embebidos,» 2017. dirección: https://api.semanticscholar.org/CorpusID:67290306.

[5] J. M. Herrera-López, Á. Galán-Cuenca, I. García-Morales, M. Rollón, I. Rivas-Blanco y V. F. Muñoz, «Entorno de trabajo cíber-físico para cirugía laparoscópica,» *Revista Iberoamericana de Automática e Informática industrial*, 2023. dirección: https://api.semanticscholar.org/CorpusID:265202759.

[6] A. Leppakoski, E. Salminen y T. D. Hämäläinen, «Framework for industrial embedded system product development and management,» *2013 International Symposium on System on Chip (SoC)*, págs. 1-6, 2013. dirección: https://api.semanticscholar.org/CorpusID:21473510.

[7] C. Barrera-Ramírez, Ó. González-Miranda y J. M. Ibarra-Zannatha, «Sistema de planeación y control de navegación para un vehículo autónomo en un entorno urbano,» *Pädi Boletín Científico de Ciencias Básicas e Ingenierías del ICBI*, 2022. dirección: https://api.semanticscholar.org/CorpusID:260620410.

[8] S. R. Sánchez, «Programación de laboratorios de biología portátiles abiertos basados en Arduino con el lenguaje de programación visual XOD,» 2020. dirección: https://api.semanticscholar.org/CorpusID:215856445.

[9] C. Sacta y K. David, «Desarrollo de un lenguaje de programación gráfico para microcontroladores,» 2011. dirección: https://api.semanticscholar.org/CorpusID:170649818.

[10] D. Casu, V. Dubanchet, H. Renault, A. Comellini y P. Dandré, «EROSS+ Phase A/B1 Guidance, Navigation and Control design for In-Orbit Servicing,» *Papers of ESA GNC-ICATT 2023*, 2023. dirección: https://api.semanticscholar.org/CorpusID:267272107.

[11] A. J. Bordano, G. G. Mcswain y S. T. Fernandes, «Autonomous Guidance, Navigation and Control,» 1991. dirección: https://api.semanticscholar.org/CorpusID:108853424.

[12] P. Lourenço et al., «Verification & validation of optimisation-based control systems: methods and outcomes of VV4RTOS,» *Papers of ESA GNC-ICATT 2023*, 2023. dirección: https://api.semanticscholar.org/CorpusID:267287945.

Apéndice A

Demostración del teorema de Nyquist

El título anterior es solo un ejemplo ilustrativo. Éste teorema no ameritaría un apéndice pues es parte normal del currículum de Electrónica, pero apéndices usualmente involucran aspectos de esta índole, que se salen de la línea de la tesis, pero que es conveniente incluir por completitud.

Los anexos contienen toda información adicional que se considere pertinente agregar, como manuales de usuario, demostraciones matemáticas que se salen de la línea principal de la tesis, pero que pueden considerarse parte de los resultados del trabajo.

Índice alfabético