Tecnológico de Costa Rica

Escuela de Ingeniería Electrónica

Programa de Licenciatura en Ingeniería Electrónica

Diseño e implementación de un sistema que simule el comportamiento dinámico de una planta prototipo de control automático utilizando redes neuronales artificiales

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

Ingeniero en Electrónica con el grado académico de Licenciatura

Jorge Andrés Brenes Alfaro

Cartago, 25 de noviembre, 2022

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 *Diseño e implementación de un sistema que simule el comportamiento dinámico de una planta prototipo de control automático utilizando redes neuronales artificiales*, realizado por el señor Jorge Andrés Brenes Alfaro 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

M.Sc. José Miguel Barboza Retana
Profesor Lector

Dr. Adolfo Chaves Jiménez
Profesor Lector

Dr. José Pablo Alvarado Moya
Profesor Asesor

Cartago, 25 de noviembre de 2022

Declaro que los resultados obtenidos en el presente trabajo de investigación, previo a la obtención del título de Licenciado en Ingeniería en Electrónica, son absolutamente originales, auténticos y personales.

Soy consciente de que el hecho de no respetar los derechos de autor y realizar una mala conducta científica; es decir, fabricación de datos falsos y plagio, conlleva sanciones universitarias y/o legales.

En tal virtud, declaro que el trabajo de investigación realizado sujeto a evaluación no ha sido presentado anteriormente para obtener algún grado académico o título, ni ha sido publicado en sitio alguno y los efectos legales y académicos que se puedan derivar del trabajo propuesto de investigación son y serán de mi sola y exclusiva responsabilidad legal y académica.

Jorge Andrés Brenes Alfaro

Cartago, 25 de noviembre de 2022

Céd: 3-0514-0729

Resumen

El presente proyecto busca simular mediante redes neuronales artificiales el comportamiento dinámico de un péndulo amortiguado a hélice (PAHM). Para el desarrollo de la red neuronal artificial mimetizadora (RNAM) se requiere la conexión de la planta con un sistema embebido NVIDIA Jetson TX2. Para la comunicación de los sistemas se hace uso del protocolo UART. El sistema principal es la Jetson TX2, que transmite el valor requerido para movilizar el motor que hace girar la hélice y recibe el ángulo del péndulo. Estos datos son almacenados para el entrenamiento de la RNAM.

Se desarrolla una planta sintética que corresponde al modelado matemático lineal de la PAHM, con el objetivo de producir datos sintéticos y comprobar con ellos si la RNAM logra aprender el comportamiento dinámico de este caso idealizado. Su respuesta es satisfactoria con un error absoluto medio de 0,0621° en sus predicciones.

Por otro lado, se implementa la RNAM con datos reales del PAHM, cuyos resultados también son satisfactorios prediciendo con un error absoluto medio de 1,0875°. A estos resultados se llega luego de un proceso de selección de modelos que optimiza hiperparámetros de los modelos neuronales artificiales, y de las estrategias de entrenamiento.

Se concluye que la RNAM tiene la capacidad de simular el comportamiento dinámico de la planta PAHM. En cuanto a su comparación con el modelo sintético lineal, el cual presenta un error absoluto promedio de 17,5177° con respecto a la respuesta real de la PAHM, se puede concluir que la aplicación de la RNAM aproxima con un menor error que el modelo matemático lineal.

Palabras clave: GRU, NVIDIA Jetson TX2, Modelo matemático, RNAM, UART.

Abstract

The present project seeks to simulate, through artificial neural networks, the dynamic behavior of a propeller damped pendulum (PAHM). For the development of the mimicking artificial neural network (RNAM), it is required the connection of the system with an NVIDIA Jetson TX2 embedded system. For the communication of the systems the UART protocol is used. The main system is the Jetson TX2, which transmits the value required to drive the motor that rotates the propeller and receives the angle of the pendulum. This data is stored for training the RNAM.

A synthetic system corresponding to the linear mathematical modeling of the PAHM is developed, with the objective of producing synthetic data and testing with them if the RNAM is able to learn the dynamic behavior of this idealized case. Its response is satisfactory with a mean absolute error of 0,0621° in its predictions.

On the other hand, RNAM is implemented with real PAHM data, whose results are also satisfactory, predicting with a mean absolute error of 1,0875°. These results are obtained after a model selection process that optimizes the hyperparameters of the artificial neural models and the training strategies.

It is concluded that the RNAM has the capability to simulate the dynamic behavior of the PAHM system. In comparison with the linear synthetic model, which presents an mean absolute error of 17,5177° with respect to the real response of the PAHM, it can be concluded that the RNAM application approximates with a lower error than the mathematical linear model.

Keywords: GRU, NVIDIA Jetson TX2, Mathematical model, RNAM, UART.

A mi querida familia por su apoyo incondicional

Agradecimientos

Me gustaría agradecer a mis queridos padres Lidieth y Enrique por su apoyo incondicional, preocupación, confianza y su empeño para que yo estudiase y fuera un profesional en el futuro. Les agradezco por los valores inculcados que me han hecho la persona que soy hoy en día y todas las enseñanzas y cariño que me han dado. De igual forma, agradecer a mi querido hermano José Enrique y hermanas Viviana, Karol y Andrea; quienes siempre han estado para mi incondicionalmente, me han apoyado todo el camino hasta el final. Infinitas gracias a mi familia por el cariño, la confianza y por motivarme a ser una mejor persona cada día. Un especial agradecimiento a Andrea y a mi cuñado Chad quienes han depositado mucha confianza en mi y me han dado la oportunidad de estudiar y obtener mi título universitario.

Agradecer a mi equipo de trabajo: Kimberly, Jason, Pablo y Salomón, quienes más que compañeros son parte de mi vida y un orgullo para mi el haberlos conocido y cumplir metas al lado de compañeros maravillosos como ustedes. Gracias por la amistad, ayuda y compañía brindada. Agradezco en especial a Kimberly quien siempre ha estado ahí apoyándome y motivándome para seguir adelante. Te agradezco todo tu cariño y por siempre estar presente tanto en los momentos felices como en los difíciles e impulsarme a avanzar y ser mejor.

Por supuesto, agradecer a mi asesor Pablo Alvarado por su paciencia, sabiduría, consejos y por confiar en mi como profesional. Le agradezco el tiempo dedicado para atender mis dudas y orientar mi trabajo.

Me agradezco a mí mismo por mi sacrificio, disciplina y esfuerzo para lograr cumplir mis metas y no rendirme en el camino. Por último, un agradecimiento general a todos los mencionados ya que nada de esto hubiera sido posible sin ustedes, este trabajo es el resultado de todo el apoyo brindado a mi persona y estoy infinitamente agradecido y agradezco a Dios, por ponerlos en mi camino.

Jorge Andrés Brenes Alfaro

Cartago, 29 de noviembre de 2022

Índice general

Índice de figuras III

Índice de tablas V

Lista de símbolos y abreviaciones VI

Lista de símbolos y abreviaciones VI

1. Introducción 1
- 1.1. Proceso de diseño en control . . . 1
- 1.2. Aprendizaje automático en control . . . 3
- 1.3. Red Neuronal Mimetizadora . . . 5
- 1.4. Objetivos y estructura del documento . . . 7

2. Marco teórico 9
- 2.1. Péndulo amortiguado a hélice (PAMH) . . . 9
  - 2.1.1. Descripción de la planta . . . 10
  - 2.1.2. Modelo matemático del PAHM . . . 11
  - 2.1.3. Kit CY8CKIT-059 PSoC 5 LP . . . 13
  - 2.1.4. Controlador de velocidad electrónico (ESC) . . . 14
- 2.2. Nvidia Jetson TX2 . . . 16
- 2.3. Protocolo de comunicación UART . . . 17
- 2.4. Redes neuronales artificiales (RNA) . . . 18
  - 2.4.1. Redes Neuronales Recurrentes . . . 20
  - 2.4.2. Unidades Recurrentes con Compuerta (GRU) . . . 22
- 2.5. Métricas de evaluación . . . 24

3. Red neuronal artificial mimetizadora (RNAM) 25
- 3.1. Conexión Jetson TX2 y PAHM . . . 25
  - 3.1.1. Implementación de hardware . . . 26
  - 3.1.2. Implementación de software . . . 28
- 3.2. Planta sintética . . . 32
- 3.3. Modelado de la RNAM . . . 34
  - 3.3.1. Preprocesamiento de los datos . . . 35
  - 3.3.2. Arquitectura de la RNAM . . . 36

3.3.3. Selección de modelo 37

4. Resultados y análisis **40**
4.1. Comprobación de no linealidad 40
4.2. Modelo empírico lineal 43
4.3. Análisis de la RNAM sintética 45
4.4. Análisis de la RNAM real 51

5. Conclusiones y recomendaciones **57**
5.1. Conclusiones 57
5.2. Recomendaciones 58
5.3. Trabajos a futuro 58

Bibliografía **59**

A. Obtención modelo empírico en Matlab **63**

B. Espacio de estados de la PAHM **65**

C. Resultados adicionales de la RNAM **67**
C.1. RNAM sintética 67
C.2. RNAM real 67

Índice de figuras

1.1. Proceso de diseño para el control de un sistema. 2
1.2. Interacción del aprendizaje reforzado. 4
1.3. Diagrama de etapas involucradas en la solución. 5
1.4. Diagrama de control para la PAHM. 6
1.5. Diagrama de etapas involucradas en la red neuronal mimetizadora. 7

2.1. Péndulo Simple. 9
2.2. Péndulo amortiguado a hélice. 10
2.3. Diagrama de fuerzas en la PAHM. 11
2.4. Kit CY8CKIT-059 PSoC 5LP 14
2.5. Comparación del PWM estándar con el *PWM One Shot*. 15
2.6. Kit de desarrollo NVIDIA Jetson TX2. 16
2.7. Conexión transmisor-receptor del protoclo UART. 17
2.8. Trama de datos del protocolo UART. 18
2.9. Red neuronal artificial multicapa. 19
2.10. Red neuronal recurrente con una capa oculta. 20
2.11. Red neuronal recurrente desarrollada. 21
2.12. Estructura de la red neuronal GRU. 24

3.1. Diagrama de comunicación entre la Jetson TX2 y el PSoC 5 LP. 26
3.2. Detalles de los pines del kit de prototipo PSoC 5LP. 27
3.3. Magnitud de la respuesta en frecuencia del PAHM. 29
3.4. Diagrama de conexión de bloques en el PSoC 5 LP. 29
3.5. Diagrama de flujo del programa ejecutado en el sistema PSoC 5 LP. 30
3.6. Interconexión de los módulos de la Jetson TX2. 31
3.8. Ejemplificación del PAHM como caja negra. 32
3.7. Diagrama de flujo del programa ejecutado en la Jetson TX2. 33
3.9. Rango de datos de estimación y validación. 34
3.10. Diagrama de etapas para la elaboración de la RNAM. 35
3.11. Etapas del preprocesamiento del conjunto de datos. 36
3.12. Arquitectura de la RNAM. 37
3.13. Arquitectura de la RNAM con 2 capas GRU. 38
3.14. Diagrama de flujo para la RNAM física. 39

4.1. Respuesta del PAHM ante las entradas escalón de magnitud 0,1 y 0,2. . . . 41

4.2. Aplicación de un factor ante la respuesta escalón. 41
4.3. Diez entradas para la estimulación del PAHM. 42
4.4. Diez respuestas entregadas por la PAHM ante la entrada de la figura 4.3. . 42
4.5. Promedio y desviación estándar de las respuestas del PAHM en la figura 4.4. 43
4.6. Predicción del modelo de ident y la respuesta medida en la planta física. . 44
4.7. Comparación de la respuesta del modelo matemático con la PAHM real. . . 44
4.8. Predicción de la prueba base de la RNAM sintética. 45
4.9. Predicciones con el uso de diferentes métricas en la función de pérdida. . . 47
4.10. Predicción de la RNAM entrenada con valores óptimos 48
4.11. Predicción de la RNAM sintética con preentrenamiento. 48
4.12. Predicciones dadas por la RNAM con el uso de dos capas GRU. 49
4.13. Predicción del modelo preentrenado por segunda vez. 50
4.14. Señal de entrada para las predicciones de la RNAM real. 51
4.15. Predicción del modelo de la RNAM real con la configuración óptima. . . . 52
4.16. Predicción del modelo con preentrenamiento de la RNAM real. 53
4.17. Predicciones dadas por la RNAM con el uso de dos capas GRU. 54
4.18. Predicción de la RNAM real preentrenada por segunda vez. 55
4.19. Ejemplificación de la pérdida MAE de los modelos. 56

A.1. Ventana para importar datos al ident. 63
A.2. Ventana para la obtención de la función de transferencia del ident. 64
A.3. Ventana correspondiente a la herramienta ident. 64

C.1. Predicción de la RNAM sintética 8/16. 68
C.2. Predicción de la RNAM sintética 64/16. 68
C.3. Predicción de la RNAM sintética 32/32. 69
C.4. Predicción de la RNAM sintética 32/96. 69
C.5. Predicción de la RNAM real 32/96. 70
C.6. Predicción de la RNAM real 64/16. 70

Índice de tablas

2.1. Especificaciones generales del kit CY8CKIT-059 PSoC 5 LP. 14
2.2. Especificaciones generales del kit de desarrollo NVIDIA Jetson TX2 17

3.1. Descripción del conector J17 para el sistema Jetson TX2. 27
3.2. Pines asignados para el PSoC 5 LP mediante *PSoC Creator*. 27

4.1. Pérdidas al variar hiperparámetros en la RNAM sintética 46
4.2. Pérdida de RNAM sintética evaluando preentrenamiento. 50
4.3. Pérdida de RNAM sintética con dos capas GRU. 50
4.4. Pérdidas al variar hiperparámetros en la RNAM real. 52
4.5. Pérdida de RNAM real evaluando preentrenamiento. 53
4.6. Pérdida de RNAM real con dos capas GRU evaluando preentrenamiento. . 55
4.7. Errores absolutos medios de las predicciones por los modelos de RNAM. . . 56

Capítulo 1

Introducción

1.1. Proceso de diseño en control

Actualmente, el control automático de plantas y procesos tiene una cuota de mercado creciente en la economía global, ya que gran parte de los procesos industriales y de manufactura moderna se encuentran automatizados [1]. En las fábricas modernas e instalaciones industriales se hace cada vez más necesario disponer de sistemas de control que permitan mejorar y optimizar los procesos, lo que ha repercutido en beneficios como la reducción de costos, el consumo de energía y tiempo, y el incremento en la calidad y volumen de producción. Además, el logro de una vida con mayores comodidades, aligerar la carga de operaciones manuales repetitivas y rutinarias, la eliminación de errores y aumento en la seguridad de los procesos [2], [3].

Esta área ha demostrado sus aplicabilidad como componente imprescindible en sistemas de vehículos motrices, espaciales y robóticos, en procesos modernos de fabricación y en cualquier operación industrial que requiera control de temperatura, presión, humedad y flujo [3]. Esto conlleva a promover el desarrollo y aplicación de técnicas modernas de control, impulsando el desarrollo en avances científicos y tecnológicos en áreas como domótica, procesos químicos, ingeniería mecánica, automovilismo, aeronáutica, espacial, entre otras [2].

El control automático ha jugado un papel central en el avance de la ingeniería y la ciencia, donde los sistemas bajo estudio son dinámicos y el conocimiento de la teoría de análisis de sistemas y de control proporciona una base para entender y controlar el comportamiento de dichos sistemas [2]. En los métodos convencionales de control aplicados, cada etapa requiere de intervención manual, en la que por medio de experimentación deben ajustarse parámetros, cuya validez depende de qué tan precisos son los modelos determinados previamente. En la figura 1.1 se muestran las etapas para el diseño convencional de control, donde los pasos cinco, seis y siete son procesos iterativos que se pueden optimizar de forma que se reduzca la configuración manual.

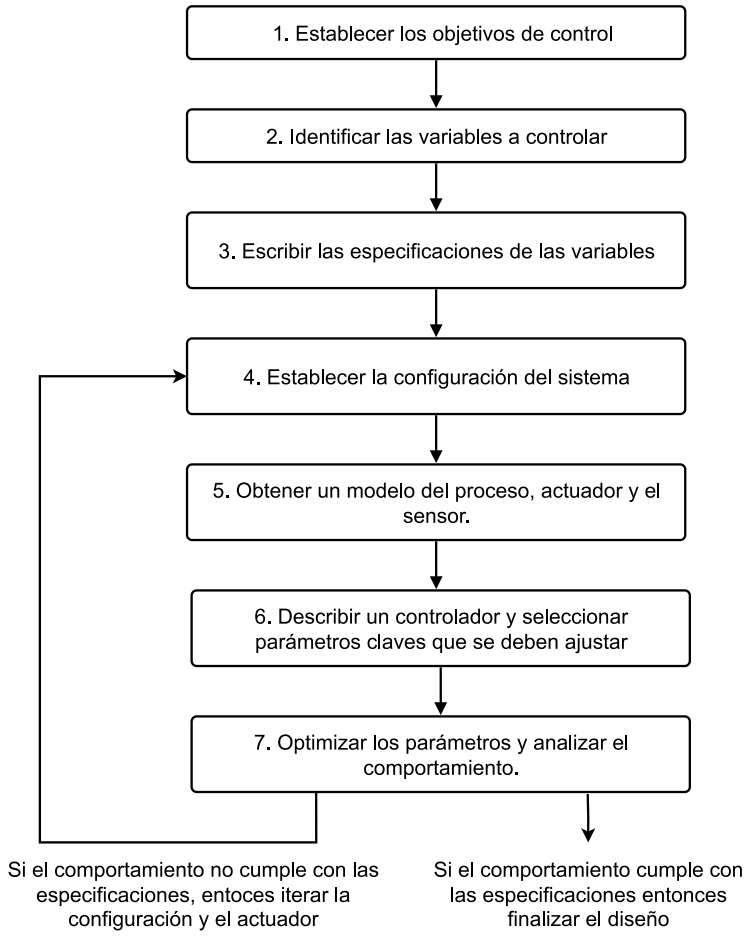


Figura 1.1: Proceso de diseño para el control de un sistema. Fuente: [4].

En sistemas de control, el proceso a controlar requiere de una representación que caracterice el proceso físico; es decir, un modelo matemático dependiente del tiempo que represente su comportamiento, tal como se observa en el paso cinco de la figura 1.1. Esta representación se puede obtener de forma teórica, lo que se denomina *modelado*; así como también de forma empírica, mediante experimentos sobre el sistema real, basándose en datos de entrada y salida del proceso, llamado *identificación*. Una ventaja de contar con un modelo es que permite predecir y estudiar el comportamiento dinámico y estático de los sistemas, sin exponer en el proceso al sistema físico a daños. Se puede determinar qué puede ocurrir, simulando con el modelo alguna condición, entrada inusual o perturbación [5].

Sin embargo, debido a la presencia de no linealidades, varianza en el tiempo y múltiples entradas y salidas en las plantas modernas de control, comparado a los sistemas LTI de una entrada y salida se vuelve más compleja la descripción del sistema, requiriendo una mayor cantidad de ecuaciones. Con esos sistemas se hace cada vez más difícil derivar modelos manualmente que logren un equilibrio aceptable entre precisión y velocidad [3], [6]. Los sistemas dinámicos en su rango completo de operación son con frecuencia no lineales y variantes en el tiempo, y en aplicaciones modernas de ingeniería, con su creciente complejidad, se presentan retos en la ejecución de control. Esto incluye por ejemplo, sistemas

de comando de vuelo, manipuladores de robot, sistemas de inyección de combustible de alto rendimiento, entre otros [7]. Este panorama ha incentivado la búsqueda de nuevas técnicas para la identificación de sistemas, en particular al desarrollo de métodos que crean los modelos directamente a partir de datos, basados en el aprendizaje automático (ML) y el aprendizaje profundo (DL) [6].

1.2. Aprendizaje automático en control

El control clásico que trata sistemas con una entrada y una salida pierde su potencialidad cuando se trabaja con sistemas que no son lineales ni invariantes en el tiempo [3]. A pesar de esto, se pueden controlar mientras la planta sea linealizable en puntos de operación específicos; no obstante, debido a la existencia de puntos de operación variantes en el tiempo y no linealidades en los modelos, la capacidad de controlar eficientemente el sistema se complica [8]. El control moderno se ha desarrollado para manejar la creciente complejidad de las plantas modernas y los requisitos cada vez más exigentes. Este se basa en el análisis del dominio temporal de los sistemas de ecuaciones diferenciales, simplificando el diseño de los sistemas de control porque se basa en un modelo del sistema real que se quiere controlar [3].

En sistemas dinámicos complejos, las no linealidades y perturbaciones impredecibles afectan el desempeño de los controladores [9]. La realización de modelos y controladores requieren experimentación, cuya exhaustividad y resultados están con frecuencia afectados por la no linealidad y la varianza en el tiempo de los sistemas, lo que conlleva a procesos iterativos que consumen tiempo, exploración y refinamiento de controles lineales para zonas de operación específicas. En consecuencia, surge la necesidad de diseñar e implementar estrategias innovadoras de control automático para los procesos no lineales [7], [9] y controlar el sistema en todo su espacio de operación sin la necesidad de un modelo matemático linealizado en torno de un punto de operación y minimizando la configuración manual.

En la figura 1.1 se observa que los pasos los pasos cinco, seis y siete referentes a la obtención de un modelo matemático, realizar el controlador y optimizar los parámetros de este, pueden ser optimizados de forma que se reduzca la configuración manual. Por ello, se exploran técnicas recientes involucradas con el aprendizaje automático que utilizan redes neuronales para construir modelos complejos a partir de datos y requieren de menor intervención humana. Por otro lado, en los cursos de la línea de control automático del plan de Licenciatura en Ingeniería en Electrónica del TEC se ha acumulado experiencia en el diseño de sistemas de control con las metodologías tradicionales, pero en años recientes con el crecimiento del aprendizaje automático han surgido alternativas que aún no han sido exploradas en la Escuela y que pueden representar un complemento interesante a esa línea desde el curso electivo de aprendizaje automático.

En la última década, ha habido un incremento exponencial en la aplicación de redes neu-

ronales artificiales (RNA) [6], [9] ya que constituyen una herramienta para el aprendizaje de relaciones complejas a partir de un conjunto representativo de ejemplos. Además, debido a sus capacidades de aproximación, su adaptabilidad y su tolerancia a fallos, las RNA presentan una alternativa en el modelado e identificación de sistemas no lineales [10], [11]. Son modelos paramétricos que, basándose en una medida del error y en un algoritmo para minimizarlo, aproximan una función no lineal. Además, las RNA son capaces de resolver problemas cuya solución por métodos convencionales resultan difíciles o insatisfactorias [11], [12].

Es por ello que han surgido propuestas para usar las RNA en la identificación del comportamiento de sistemas dinámicos no lineales desconocidos, donde, a partir de la respuesta del sistema real ante una entrada dada, la red sea capaz de modificar sus parámetros, aprendiendo una aproximación de la dinámica del sistema hasta conseguir un modelo fiable [13], [14]. De igual forma, las RNA ofrecen alternativas para el control de sistemas dinámicos no lineales que permiten mejorar el desempeño de un sistema en el mundo real sin la necesidad de un modelo matemático linealizado en un punto de equilibrio. De forma ideal, la red neuronal aprende con la diferencia entre la señal de control idónea y la señal calculada por la red neuronal, sin embargo, la señal de control idónea no se conoce y, por tanto, el error no se puede calcular. Una solución a ello son los algoritmos de aprendizaje reforzado, que permiten el aprendizaje con señales cualitativas del error en vez de cuantitativas [11].

El aprendizaje reforzado (RL), que ha mostrado en los últimos años avances de alto impacto social en sistemas como AlphaGo [15] y AlphaFold [16], ofrece algoritmos para buscar y desarrollar controladores de sistemas con dinámicas no lineales, estocásticas y que son desconocidas o inciertas [17]. Esta metodología busca que un agente sea capaz de encontrar la acción correcta de manera autónoma, explorando un espacio desconocido y determinando acciones a tomar mediante prueba y error. Dicho agente aprende por medio de recompensas y penalizaciones que obtiene de sus acciones, con el fin de actuar y crear la mejor estrategia posible, de forma que maximice la recompensa [18], [19]. Esta interacción es representada en la figura 1.2.

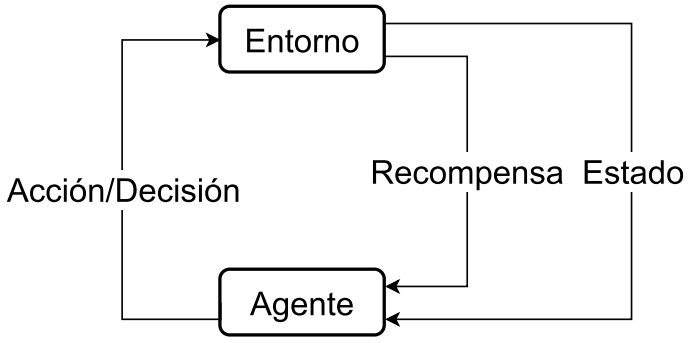


Figura 1.2: Interacción del aprendizaje reforzado. Fuente: [20].

Este enfoque no solo se ha utilizado para resolver juegos de Atari [21], [22], pero además ya existen propuestas en sistemas de control, en donde el control no es 100 % efectivo [18],

[20]. Esta metodología ajusta el control ante cambios en el entorno del sistema al aprender con las acciones, permitiendo automatizar el proceso más allá de lo alcanzable con los métodos tradicionales de control.

1.3. Red Neuronal Mimetizadora

Con lo mencionado anteriormente, se propone hacer uso de dos redes neuronales artificiales; una que mimetice el comportamiento del sistema de estudio y otra que lleve a cabo el control del sistema por medio de aprendizaje reforzado. El uso de estas dos RNA brindan la capacidad de aproximación y adaptabilidad del aprendizaje automático. En este trabajo se tratará exclusivamente el problema de la red neuronal mimetizadora (RNAM), cuyo objetivo es entonces aprender un modelo de comportamiento dinámico, implícito en la estructura neuronal, a partir de datos de entrada y salida. Esta red mimetizadora ofrece varias ventajas. Primero, permite experimentar con diversas entradas sin afectar directamente la planta, evitando el riesgo de que los experimentos sean tan violentos en los cambios que se dañe la planta. Segundo, usando la RNAM se puede usar un reloj más rápido que el tiempo real, acelerando los experimentos y eliminando la desventaja de tener que limitar la ejecución al tiempo real de la planta. Por ello, el volumen de datos disponible para el entrenamiento del modelo de aprendizaje reforzado es mayor con la red que modela al sistema real, que el disponible usando la planta en físico. Por último, con la ayuda de la RNAM se prescinde del planteamiento explícito del modelo que se necesitaría para el control clásico, pues las RNA se adaptan a cualquier tipo de estructura capturando sus características en el modelo aprendido [13].

La solución completa del diseño de un controlador con aprendizaje reforzado se representa con un diagrama de etapas en la figura 1.3. Primeramente, se recopilan datos de la planta física para entrenar la red mimetizadora. La planta a utilizar es un péndulo amortiguado a hélice (PAHM). Seguido de ello, se implementa y entrena el controlador, el cual ajusta el control ante cambios del entorno del sistema al aprender con acciones. Por último, el controlador se prueba con la red mimetizadora y posteriormente con la planta física.

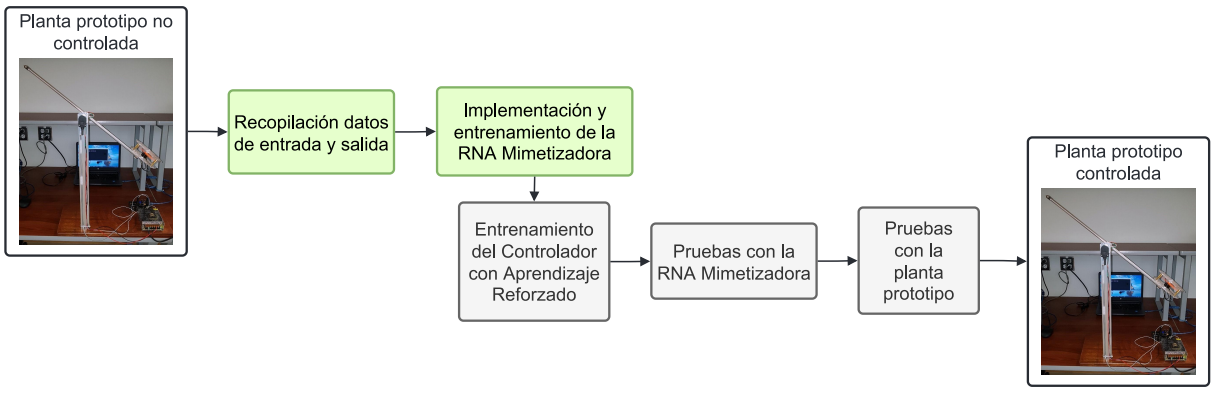


Figura 1.3: Diagrama de etapas involucradas en la solución. En verde se muestran las etapas realizadas para este trabajo y en gris las etapas a realizar en un trabajo futuro.

Un diagrama de control y sus bloques para el PAHM, se muestra en la figura 1.4. En dicho diagrama se visualizan cuatro bloques: un filtro, con el fin de limpiar la señal y eliminar ruido, los bloques de las RNA a utilizar, tanto del controlador como de la red mimética, el bloque de la planta cuya observabilidad está dada por el sensor incorporado para medir el ángulo del péndulo, partiendo de que la incertidumbre del sensor es mínima.

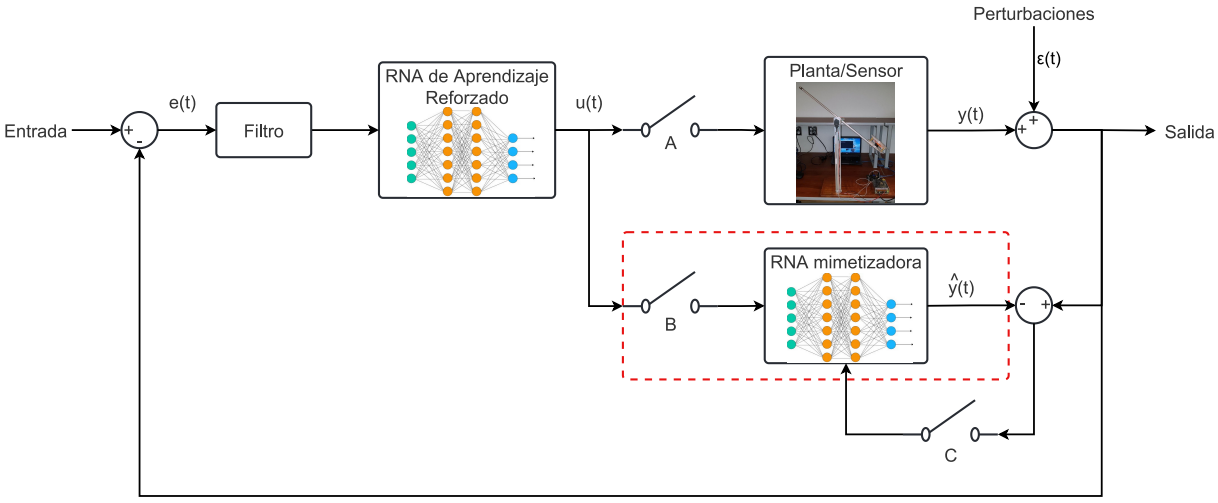


Figura 1.4: Diagrama de control para la PAHM. La línea punteada de color rojo muestra el bloque a dearrollar.

Se han agregado tres interruptores de configuración: A, B y C. Estos interruptores se cierran o abren de acuerdo a la etapa en la que se encuentre el proceso de diseño de control. Primeramente, se abren los tres interruptores y se estimula la planta para capturar sus reacciones ante diferentes entradas. Con los datos recolectados de entrada/salida se entrena la RNAM, forzando las mismas entradas y salidas, donde el ajuste de los parámetros se basa en la diferencia entre las salidas $y(t)$ y $\hat{y}(t)$ durante su entrenamiento. Seguidamente, con A y C abierto y B cerrado, se entrena el controlador de RL realizando pruebas con la RNAM como modelo de la planta física, sin ponerla en riesgo. Por último, se cierra A y se abren B y C para así controlar la planta física y evaluar el desempeño del controlador de RL. En adición, se refina la RNAM con el funcionamiento en paralelo de la planta y la red neuronal. Para ello, se cierran los tres interruptores A, B y C. El estado interno de la RNAM se ajusta para compensar efectos de las perturbaciones $\varepsilon(t)$ que tiene la planta real y que RNAM no se percata de que han ocurrido.

El presente documento está centrado en el desarrollo de la red neuronal mimetizadora, marcada con un cuadro rojo en la figura 1.4, la cual es necesaria para desarrollar e implementar en un futuro el controlador de aprendizaje reforzado. Para ello, se plantea el diagrama de solución de la figura 1.5. Para esta red mimetizadora se recopilarán datos de entrada y salida de la PAHM física y de una planta sintética realizada en base a un modelo matemático clásico. Con los conjuntos de datos recopilados se lleva a cabo el entrenamiento e implementación de la RNAM. Seguidamente, se aplica un conjunto de datos de prueba de una entrada desconocida para la RNAM, con el fin de evaluar la respuesta de la RNAM entrenada con datos de la PAHM y la respuesta dada por la RNAM entre-

nada con la planta sintética y así, verificar si la RNAM tiene la capacidad de aprender el comportamiento ideal del modelo y de la planta PAHM considerando la incertidumbre que añade el sensor.

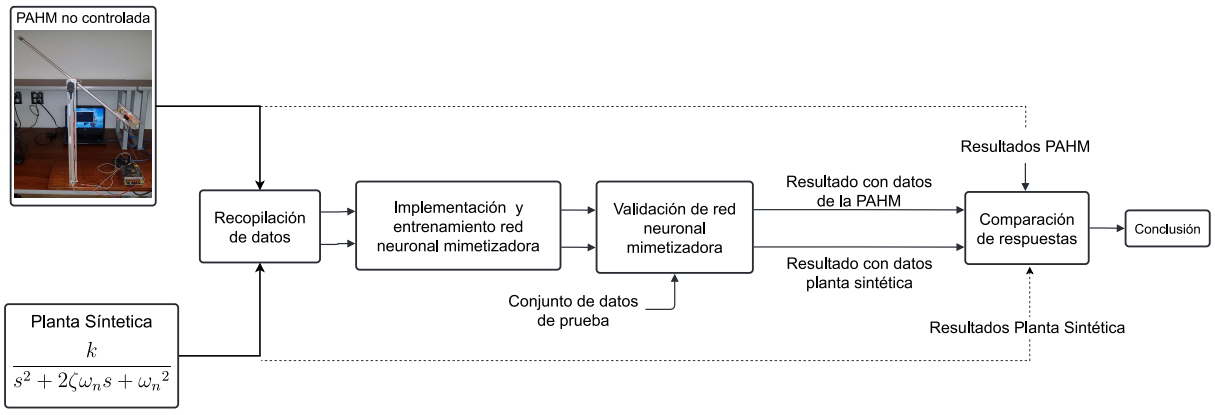


Figura 1.5: Diagrama de etapas involucradas en la red neuronal mimetizadora.

1.4. Objetivos y estructura del documento

El objetivo principal de este proyecto es desarrollar un sistema de aprendizaje automático que mimetice el comportamiento dinámico de la planta prototipo de péndulo amortiguado a hélice (PAHM) mediante el uso de redes neuronales artificiales. Para lograr este objetivo se persiguen cuatro objetivos específicos. El primero consiste en acoplar un sistema embebido de altas prestaciones al lazo de control. Esto es necesario debido a que el proceso completo, es decir, la red neuronal artificial de control y la mimetizadora en operación en tiempo real tienen demandas computacionales que usualmente requieren el uso de procesamiento con GPU. Además, el lazo de control incluye el diseño del manejo de datos desde la producción de la señal de entrada, actuador, sensor, el envío y vuelta de los datos al sistema embebido. Este sistema embebido se elige de forma que sea capaz de ejecutar el modelo seleccionado, y a su vez impone limitaciones a la complejidad de los algoritmos a seleccionar.

Como segundo objetivo específico se tiene el preparar un conjunto de datos para el entrenamiento de la red neuronal artificial mimetizadora de la planta real, particionado en datos de entrenamiento, validación y prueba. El tercer objetivo específico consiste en implementar una red neuronal artificial que mimetice el comportamiento de la planta en estudio. El último objetivo corresponde a evaluar la respuesta de la red neuronal artificial mimetizadora utilizando tanto una planta sintética como la planta prototipo real. Mediante métricas de desempeño y simulaciones se comprueba la eficacia del sistema para simular el comportamiento dinámico de la planta, ante la respuesta de una determinada entrada.

Este documento incluye lo siguiente: en el capítulo 2 se presenta el marco teórico, donde se esbozan los fundamentos de la propuesta realizada. En el capítulo 3 se detalla la solución propuesta y el modelo implementado para la solución del problema. En el capítulo 4 se presentan los resultados obtenidos. Por último, el capítulo 5 se presenta las conclusiones de la investigación y trabajo realizado, así 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 diseño e implementación de un sistema que simule el comportamiento dinámico de una planta prototipo de control automático utilizando redes neuronales artificiales. La información expuesta se deriva tanto de conocimientos propios como de información bibliográfica.

2.1. Péndulo amortiguado a hélice (PAMH)

Un péndulo simple consiste en una masa adherida a una barra de longitud fija que se encuentra suspendida en un punto de equilibrio. Esta masa se desplaza a un ángulo θ y se deja caer, generando un movimiento oscilatorio [23]. Dicho sistema se muestra en la figura 2.1. El péndulo amortiguado a hélice (PAHM) exhibe un comportamiento idéntico al péndulo simple, con la diferencia de que la masa es una hélice accionada por un motor en corriente directa, el cual mediante una fuerza de empuje mueve la masa de su punto de equilibrio. La planta PAHM se asume como un modelo simplificado de un vehículo no tripulado que se utiliza para la enseñanza de temas de dinámica y control de sistemas, donde se utiliza la fuerza de empuje para ubicar el péndulo en cualquier posición deseada o requerida por un usuario [24].

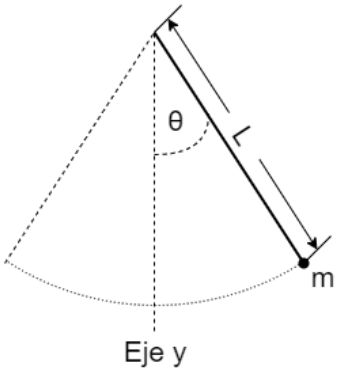


Figura 2.1: Péndulo Simple. Fuente: [25].

2.1.1. Descripción de la planta

La planta PAHM consta de un soporte rígido de aluminio extruído colocado perpendicularmente sobre una placa de madera. Sobre la parte superior de este soporte se sitúa, montado en rodamientos, el eje de pivote del péndulo. En adición, tiene un sensor de cuadratura óptico incremental, USDigital E5 de 3600 CPR (cuentas por revolución) acoplado al eje, el cual sirve para medir el ángulo. El péndulo mismo está constituido por un tubo hueco de aluminio que al final está unido a un motor sin escobillas Hobbymate 2204, gobernado por un controlador de velocidad, ESC, OneShot125 BLHeli de 15 A. El motor tiene adosada una hélice GEMFAN 5045BNR de tres aspas, protegida por rejillas cortadas en acrílico [26]. Además, cuenta con una fuente de poder de 12 V y 5 A, que alimenta al ESC y este a su vez, alimenta el motor. También utiliza un sistema embebido PSoC 5 LP y una placa para la conexión de sensores. Este sistema se muestra en la figura 2.2.

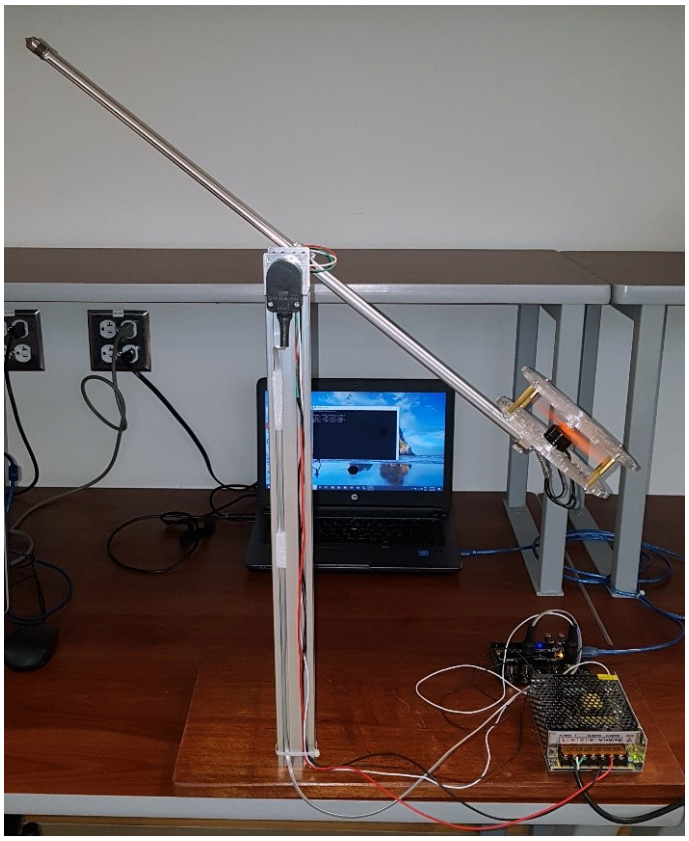


Figura 2.2: Péndulo amortiguado a hélice. Fuente: [26].

Además de ello, la planta PAHM exhibe las características de ser continua, estable y no lineal. Además, es un sistema SISO (*Single Input, Single Output*), cuya entrada es un valor codificado en PWM (modulación por ancho de pulso) que controla el motor de la hélice a través de un controlador de velocidad electrónico (ESC). El ESC utiliza el protocolo *PWM OneShot125*, lo que a grandes rasgos quiere decir que el número de revoluciones del motor será controlado por el ancho del pulso de entrada, de modo que un pulso con 125 μs de ancho representa 0 RPM y la velocidad de rotación sube linealmente con el ancho del pulso hasta alcanzar, con un pulso de ancho de 250 μs, el mayor número de revoluciones

$r_{\text{máx}}$, que con el motor utilizado es $r_{\text{máx}} = 2300\,\text{RPM}$. Así si w es el ancho del pulso en μs, el número de revoluciones:

$$r = \text{mín}\left(1, \text{máx}\left(0, \frac{w - 125\mu s}{125\mu s}\right)\right) r_{\text{máx}} \tag{2.1}$$

La salida del PAHM corresponde al ángulo del péndulo medido por el sensor de cuadratura. EL PAHM posee solo un grado de libertad, por lo se requiere de solo una ecuación para describir su movimiento y esta no necesita de intervención humana para reiniciarse por ser estable.

2.1.2. Modelo matemático del PAHM

Los modelos matemáticos son necesarios en ingeniería, pues proporcionan una caracterización del sistema, que permite predecir su comportamiento y diseñar controladores [5]. El modelo matemático de una planta se obtiene de dos formas: teóricamente o empíricamente. El modelo matemático teórico se realiza por medio de un análisis físico, el cual se plantea por medio de ecuaciones diferenciales que se obtienen a partir de leyes físicas que gobiernan un determinado sistema, por ejemplo: las leyes de Newton para sistemas físicos, las leyes de Kirchhoff para sistemas eléctricos, las ecuaciones de Euler-Lagrange para el balance de energías, entre otras [3], [27], [28].

En [29], [30] se realiza el análisis teórico del PAHM por medio de la aplicación de las leyes de Newton. El diagrama de fuerzas en la figura 2.3 sirve como punto de partida. El actuador corresponde al motor con una hélice en el extremo inferior del péndulo, el cual ejerce la fuerza de empuje $F_T(t)$. El ángulo $\theta(t)$ formado entre el péndulo y el eje vertical es la variable de salida.

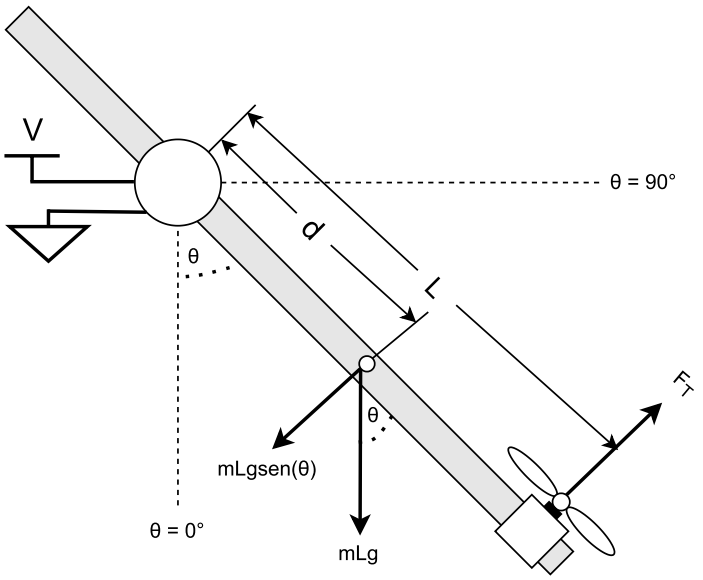


Figura 2.3: Diagrama de fuerzas en la PAHM. Fuente: [29].

Utilizando la ley de Newton para movimiento rotacional, que establece que el torque es igual al producto del momento inercial por la aceleración angular [31], y asumiendo que el ángulo es lo suficientemente pequeño como para poder aproximar $\text{sen}(\theta) \approx \theta$, se obtiene la ecuación diferencial:

$$\ddot{\theta}(t) + 2\zeta\omega_n\dot{\theta}(t) + \omega_n^2\theta(t) = K_T F_T(t) \tag{2.2}$$

Conociendo que la fuerza de empuje es a su vez proporcional a la señal de entrada $v(t)$, y utilizando la transformada de Laplace de las señales de entrada y salida, se obtiene la función de transferencia [30]:

$$\frac{\Theta(s)}{V(s)} = \frac{K}{s^2 + 2\zeta\omega_n s + \omega_n^2} \tag{2.3}$$

Así, el modelo del sistema es de segundo orden, donde ω_n es la frecuencia natural del sistema y ζ el factor de amortiguamiento, cuyo valor determina si la respuesta del sistema es subamortiguada ($0 < \zeta < 1$), críticamente amortiguada ($\zeta = 1$), sobreamortiguada ($1 < \zeta$), oscilatoria ($\zeta = 0$) o inestable ($\zeta < 0$).

En aplicaciones de control automático es común utilizar la función de transferencia en espacio de estados, el cual es una forma compacta de representar un sistema dinámico en función de n ecuaciones diferenciales de primer orden en vez de tener una sola ecuación diferencial de n-ésimo orden [32]. El espacio de estados de corresponde a:

$$\dot{\underline{\mathbf{x}}}(t) = \mathbf{A}\underline{\mathbf{x}}(t) + \mathbf{B}\underline{\mathbf{u}}(t) \tag{2.4}$$

$$\underline{\mathbf{y}}(t) = \mathbf{C}\underline{\mathbf{x}}(t) + \mathbf{D}\underline{\mathbf{u}}(t) \tag{2.5}$$

Al ser un sistema de segundo orden se tienen solamente dos estados, correspondientes a la posición angular y la velocidad angular:

$$\underline{\mathbf{x}}(t) = \begin{bmatrix} \theta(t) \\ \dot{\theta}(t) \end{bmatrix} \tag{2.6}$$

La salida está representada por $\underline{\mathbf{y}}(t) = [\theta(t)]$ y la entrada por $\underline{\mathbf{u}}(t)$. Las matrices $\mathbf{A}$, $\mathbf{B}$, $\mathbf{C}$ y $\mathbf{D}$ son:

$$\mathbf{A} = \begin{bmatrix} 0 & 1 \\ -\omega_n^2 & -\zeta\omega_n \end{bmatrix} \quad \mathbf{B} = \begin{bmatrix} 0 \\ K \end{bmatrix} \quad \mathbf{C} = \begin{bmatrix} 1 & 0 \end{bmatrix} \quad \mathbf{D} = [0] \tag{2.7}$$

Como se menciona anteriormente, la PAHM es un sistema SISO, lo que se refleja en el valor de matriz de salida $\mathbf{C}$ que indica que solo la posición angular es observable.

La otra forma de obtener un modelo matemático de una planta es de forma empírica, donde usualmente se desconoce total o parcialmente el sistema bajo estudio, por lo que se utiliza un modelo de caja negra. En este método se diseña una secuencia de entrada o estímulo capaz de generar una respuesta o salida, se registran los resultados para obtener el par entrada-salida $(x(t), y(t))$. De esta forma, se debe encontrar una estructura para que el modelo propuesto del sistema, bajo una entrada $x(t)$ cualquiera, aproxime la respuesta del sistema [5].

En la práctica se utilizan aproximaciones lineales de los sistemas como forma de simplificar el modelo y el control del sistema; sin embargo, las simplificaciones modelan de forma incompleta el comportamiento del sistema dado que existen condiciones como histéresis o caos que son imposibles de representar linealmente [5].

Una herramienta alternativa para modelar son las redes neuronales artificiales cuyas funciones de activación son no lineales, por lo que trabajan de forma no lineal desde el principio y resultan útiles para la identificación de sistemas no lineales, además de que se caracterizan por sus capacidades de adaptabilidad, tolerancia a fallos, entre otras [10], [11].

2.1.3. Kit CY8CKIT-059 PSoC 5 LP

La comunicación con el sensor y el motor de la PAHM se da mediante un microcontrolador, el cual es el kit CY8CKIT-059 PSoC 5 LP. El PSoC es un acrónimo de *programmable system on chip*, el cual pertenece a una familia de microcontroladores desarrollados por la empresa *Cypress*. Este cuenta con módulos tanto analógicos como digitales programables y reconfigurables [33], [34]. Esta placa posee un conector micro-USB para permitir el desarrollo de aplicaciones USB, condensadores CMOD integrados para permitir el desarrollo de *CapSense*, condensadores de derivación para garantizar conversiones ADC de alta calidad y condensadores de carga para conectar un oscilador de cristal externo de 32 kHz [33].

La tabla 2.1 resume las especificaciones que posee el kit del PSoC 5 LP.

Además, el kit incluye un KitProg que ofrece la funcionalidad de programación y depuración mediante la conexión USB. Se utiliza como programa de desarrollo una aplicación denominada *PSoC Creator*, la cual tiene una interfaz gráfica donde se visualizan los bloques tanto digitales y analógicos que posee el microcontrolador y cuáles de ellos se han utilizado en la aplicación que se desarrolla [33]. También permite escribir el código de programación para la aplicación en dos lenguajes de programación: C y ensamblador. El microcontrolador PSoC 5 LP se ilustra en la figura 2.4, donde se observa la tarjeta principal y el kitprog.

Tabla 2.1: Especificaciones generales del kit CY8CKIT-059 PSoC 5 LP. Fuente: [34].

Especificaciones generales	
Tensión de Operación	3.3 V a 5 V
Rendimiento	Controlador DMA. Procesador de filtro digital (DFB). Procesador de 32-bit ARM Cortex M3.
Almacenamiento	Memoria flash: 256 KiB. Memoria RAM: 64 KiB. EEPROM: 2 KiB.
Interfaces de Comunicación	UART, I2C, SPI, I2S
Frecuencia	DC a 80 MHz.
Temperatura	-40 °C a 85 °C
Puertos	USB, micro USB, J1 y J2 headers
Fabricante	Cypress Semiconductor

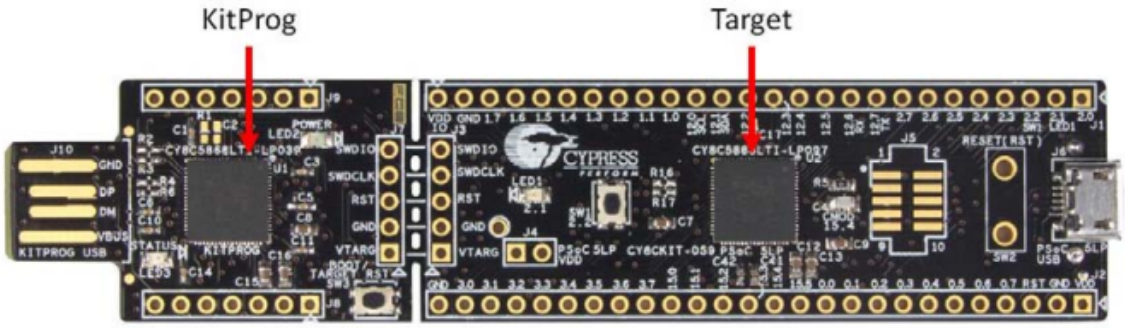


Figura 2.4: Kit CY8CKIT-059 PSoC 5 LP. Fuente: [33].

2.1.4. Controlador de velocidad electrónico (ESC)

Como se menciona en la sección 2.1.1, el PAHM utiliza un motor sin escobillas, lo que proporciona un mayor tiempo de vida, eficiencia y velocidad y menor mantenimiento y ruido. El funcionamiento de este motor se basa en la aplicación de corriente directamente al bobinado del estator, donde se generará un campo electromagnético que interacciona con el campo de los imanes permanentes del rotor, originando una fuerza que hace girar al rotor [35]. No obstante, al no tener escobillas se requiere una etapa de potencia adecuada para proporcionar corriente a un arreglo de bobinados y hacer que el rotor gire. Para ello, se hace uso de un controlador de velocidad electrónico (ESC, por sus siglas en inglés).

Un ESC es un circuito electrónico que controla un motor eléctrico con el fin de conseguir que el motor gire a la velocidad deseada. Este dispositivo determina en qué posición se encuentra el rotor en cada instante para conseguir que el sentido y la velocidad de conmutación de la corriente sean las adecuadas para provocar movimiento en el rotor. Su conexión se da por medio de tres cables, uno para cada fase del motor. Por otro lado, se conecta el ESC a la fuente del PAHM utilizando los dos cables correspondientes a la

alimentación y el tercer cable se conecta al microcontrolador para la transmisión de una señal de PWM que indica la velocidad de giro que debe alcanzar el motor [35]. Esta señal enciende y apaga rápidamente un conjunto de transistores permitiendo una variación suave, precisa y eficiente de la velocidad, dirección y del torque del motor eléctrico [36]. Entre más tiempo se dejen encendidos los transistores, mayor potencia y por ende, mayor velocidad.

Dado que el ESC envía la velocidad requerida al motor, tiene que haber un protocolo de comunicación entre ellos. El protocolo de PWM es frecuentemente utilizado para controlar la cantidad de energía que se envía a una carga y es utilizado en aplicaciones para regular la velocidad de giro de los motores, regulación de intensidad luminosa, controles de elementos termoeléctricos o fuentes conmutadas, entre otros usos. Los ESC utilizan un pulso de entrada periódico de ancho típicamente entre 1 ms y 2 ms para variar la potencia de cero a máxima, respectivamente. No obstante, esta frecuencia resulta lenta dependiendo de la aplicación, por lo que se han desarrollado protocolos alternativos a partir del PWM [35].

Uno de ellos es el *PWM One Shot* 125, el cual es utilizado en el ESC del PAHM utilizado en este proyecto. Este protocolo es similar al PWM estándar, excepto que los anchos de pulso se dividen por un factor de 8, lo que permite una comunicación más rápida en comparación al estándar, 8 veces más rápida, enviando datos en pulsos de 125 μs a 250 μs [37]. Tal y como se aprecia en la figura 2.5. Los pulsos de PWM con *One Shot* 125 tardan un total de 250 μs, donde para indicarle al ESC que debe detener el motor, el PWM debe de estar al 50 % de su ciclo, o lo que viene siendo 125 μs de cada 250 μs y para indicarle al ESC que debe poner el motor a su máxima velocidad, el PWM debe de estar al 100 % de su ciclo, lo que viene siendo 250 μs. Gracias a este protocolo es posible ejecutar un control sobre los motores más rápido. Existen otros protocolos más recientes como *PWM One Shot 42* que permiten una mayor velocidad [37].

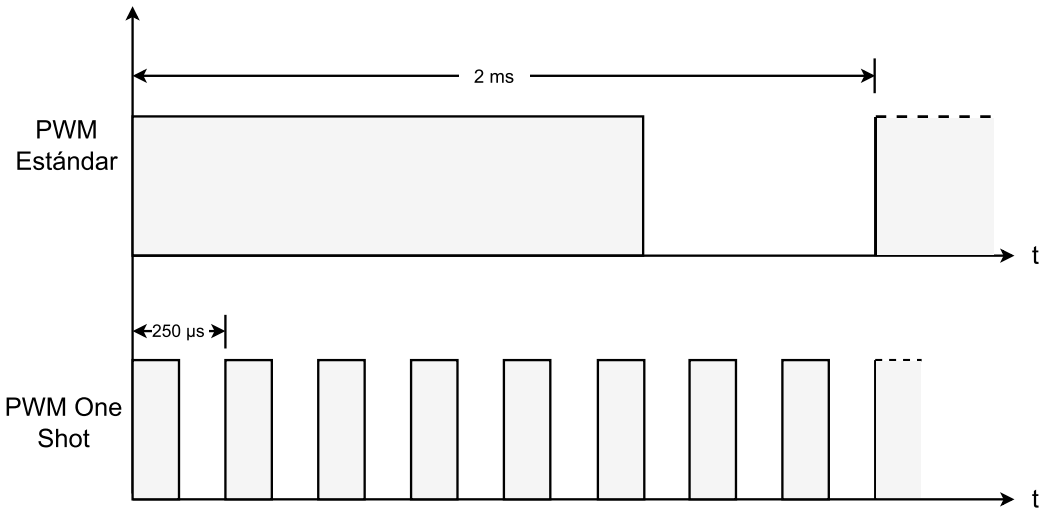


Figura 2.5: Comparación del PWM estándar con el *PWM One Shot* 125. Fuente: [35].

2.2. Nvidia Jetson TX2

Debido a la exigencia computacional que demandan los métodos de aprendizaje automático se opta por utilizar un kit de desarrollo NVIDIA Jetson TX2. Este sistema embebido de altas prestaciones se encarga de la implementación de las RNA para predicciones de datos de entrada. Además, se encarga de la lectura de los datos y proporciona la alimentación al PSoC 5 LP para su comunicación.

El módulo posee un procesador Tegra de NVIDIA que integra una unidad de procesamiento central (CPU) de arquitectura ARM y una unidad de procesamiento gráfico (GPU) de arquitectura NVIDIA Pascal, diseñado para maximizar el rendimiento y eficiencia energética para cargas de trabajo con necesidades de cálculo exigentes. Su sistema operativo es Linux4Tegra, basado en Ubuntu 18.04, el cual está disponible a través de la imagen de la tarjeta SD incluida, que está diseñada para ejecutar hardware NVIDIA. Por otro lado, el kit también cuenta con conectores PCLe, SATA, HDMI, USB y micro-USB, Ethernet, expansiones con pines de entrada-salida de propósito general (GPIO), cámara, entre otros que se documentan en [38]-[40]. Este kit de desarrollo se ilustra en la figura 2.6.

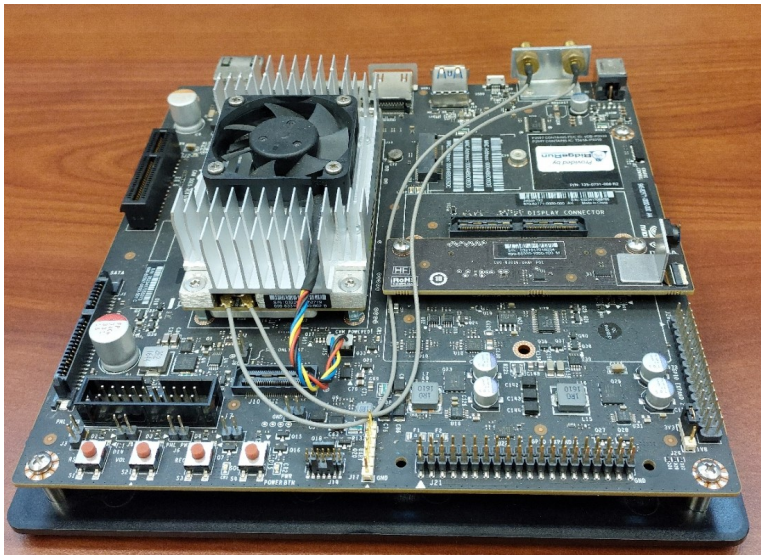


Figura 2.6: Kit de desarrollo NVIDIA Jetson TX2.

Debido a su capacidad computacional la Jetson TX2 se usa para aplicaciones de inteligencia artificial, visión por computadora y aprendizaje automático, siendo una solución a proyectos que requieren involucrar hardware y software. Ejemplo de aplicaciones donde se utiliza dicho embebido es en robots autónomos, drones, dispositivos médicos portátiles, vehículos inteligentes, ente otros.

En la tabla 2.2 se resumen las especificaciones que posee el kit de desarrollo NVIDIA Jetson TX2.

Tabla 2.2: Especificaciones generales del kit de desarrollo NVIDIA Jetson TX2. Fuente: [40].

Especificaciones generales	
CPU	Dual-core Denver 2 64 bit + Quad core ARM Cortex A57
GPU	256 NVIDIA CUDA cores
Memoria	4 GB 128-bit LPDDR4
Almacenamiento	16 GB eMMC 5.1
Red	10/100/1000 BASE-T Ethernet
Video	Decode 2x 4K60 Video Encode 1x 4K60
Camera	12 lanes MIPI CSI-2, D-PHY 1.2 (30 Gbps)
Interfaces periféricas	1x USB 3.0, 3x USB 2.0, 2x PCLe 3x UART, 2x SPI, 4x I2C, 1x CAN, 4 x I2S, GPIOs, 1x SD CARD/SDIO

2.3. Protocolo de comunicación UART

El transmisor-receptor asíncrono universal o UART (por sus siglas en inglés) es un protocolo de comunicación encargado de interconectar dos puertos seriales: uno que actúa como receptor y el otro como transmisor, donde se da el intercambio de datos en serie entre dos dispositivos. Dicho protocolo utiliza dos hilos de comunicación, uno para transmisión y otro para recepción, donde la comunicación puede ser *simplex* (los datos se envía en una sola dirección), *half-duplex* (se comunican uno a la vez) o *full duplex* (ambos transmiten y reciben al mismo tiempo) [41]. Esto se ilustra en la figura 2.7.

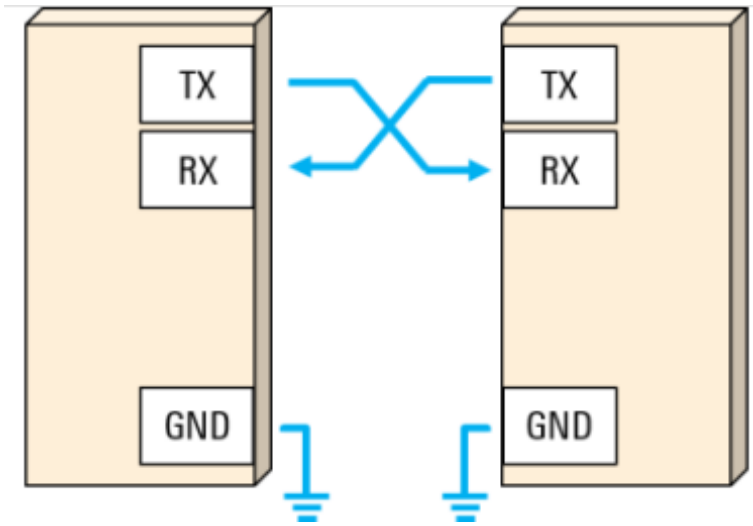


Figura 2.7: Conexión transmisor-receptor del protoclo UART. Fuente: [41], [42].

El transmisor (Tx) se encarga de empaquetar y enviar datos a través del puerto serial. Este toma un dato en paralelo de un tamaño establecido y lo empaqueta según el protocolo y lo envía *bit* a *bit* a la velocidad estipulada. Por el otro lado, el receptor (Rx) funciona de manera opuesta: se encarga de tomar un dato a través del puerto serial, desempaquetarlo y almacenarlo de manera paralela. La unidad de almacenamiento es en forma de cola, donde el primer dato en entrar es el primero en salir (FIFO, por sus siglas en inglés). Este tiene como objetivo evitar un desface entre la información que se envía y la que se recibe.

El protocolo UART posee parámetros para la transmisión de información. Debe definirse la longitud de palabra entre cero a ocho *bits*, es decir, la cantidad de *bits* que constituye la palabra transmitida, cuyo valor se encuentra entre 0 y 8 *bits*. El otro parámetro a definir es la tasa de baudios (*baudrate*), que determina la velocidad con que se transmite los datos a través del sistema. Las tasas más frecuentes son 9600 bit/s, 19200 bit/s, 38400 bit/s, 57600 bit/s, 115200 bit/s [42]. También se debe definir si existe o no control de flujo ya sea por *software* o *hardware*.

Además, el protocolo establece cómo realizar el empaquetamiento de los datos, donde la trama de bits está compuesta por un bit de inicio, cuyo valor es un cero lógico. Seguido de ello, viene la palabra de datos, le sigue un bit de paridad, el cual es opcional y su valor depende si se quiere paridad par o impar. Por último, un bit de parada, cuyo valor es un uno lógico. Esta secuencia se muestra en la figura 2.8.

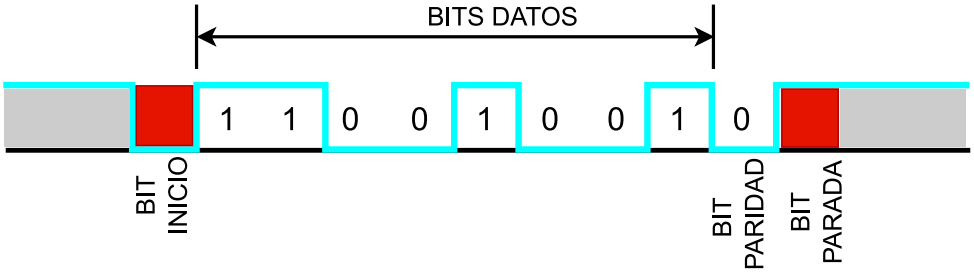


Figura 2.8: Trama de datos del protocolo UART. Fuente: [41], [42].

Otro aspecto a considerar es que si no se están transmitiendo datos, la línea de salida debe estar en nivel alto, denotando que existe comunicación entre dispositivos, pero no existe trasiego de datos [41], [42]. De esta manera, el bit de inicio le indica al receptor que una transmisión se ha iniciado.

2.4. Redes neuronales artificiales (RNA)

Las RNA se han inspirado a grandes rasgos en el funcionamiento básico de las neuronas biológicas del cerebro humano, conteniendo capas de unidades interconectadas o nodos. Estas redes ofrecen ventajas como adaptabilidad, tolerancia a fallas y uniformidad en el análisis y diseño, lo que permite utilizarlas para problemas complejos o tareas donde derivar relaciones lógicas entre entradas y salidas de forma explícita se dificulta, como el reconocimiento de patrones y el análisis predictivo [13], [43].

Una RNA está formada por una capa de entrada, un determinado número de capas ocultas (puede ser cero) y una capa de salida. En la figura 2.9 se muestra la organización de las capas de la RNA interconectadas. Entre cada una de estas capas existen matrices de pesos, que son los que le otorgan a la red neuronal su potencial de cálculo y capacidad de aprendizaje, debido a que varían a medida que el sistema aprende [44], [45]. En cada neurona se realiza una suma ponderada de las entradas x_i y un sesgo b. Lo que hace cada peso w_i es simplemente multiplicar a su entrada correspondiente y define la importancia relativa de cada entrada. Seguidamente, se somete la respuesta a una función de activación no lineal $f(\cdot)$ para producir la salida del nodo o neurona [44]-[46]:

$$y = f\left(\sum_{i=0}^{n} w_i x_i + b\right) \tag{2.8}$$

Esto se aplica a todas las neuronas de una capa hasta dar una salida vectorial $\underline{\mathbf{y}}$ correspondiente a la entrada vectorial $\underline{\mathbf{x}}$, con el sesgo vectorial $\underline{\mathbf{b}}$:

$$\underline{\mathbf{y}} = f\left(\mathbf{W}\underline{\mathbf{x}} + \underline{\mathbf{b}}\right)) \tag{2.9}$$

Finalmente, la RNA completa utiliza las salidas de una capa como entrada para la etapa siguiente:

$$\underline{\mathbf{y}} = f_N(\mathbf{W}_N f_{N-1}(\mathbf{W}_{N-1}...f_2(\mathbf{W}_2 f_1(\mathbf{W}\underline{\mathbf{x}} + \underline{\mathbf{b}}_1) + \underline{\mathbf{b}}_2)... + \underline{\mathbf{b}}_{N-1}) + \underline{\mathbf{b}}_N) \tag{2.10}$$

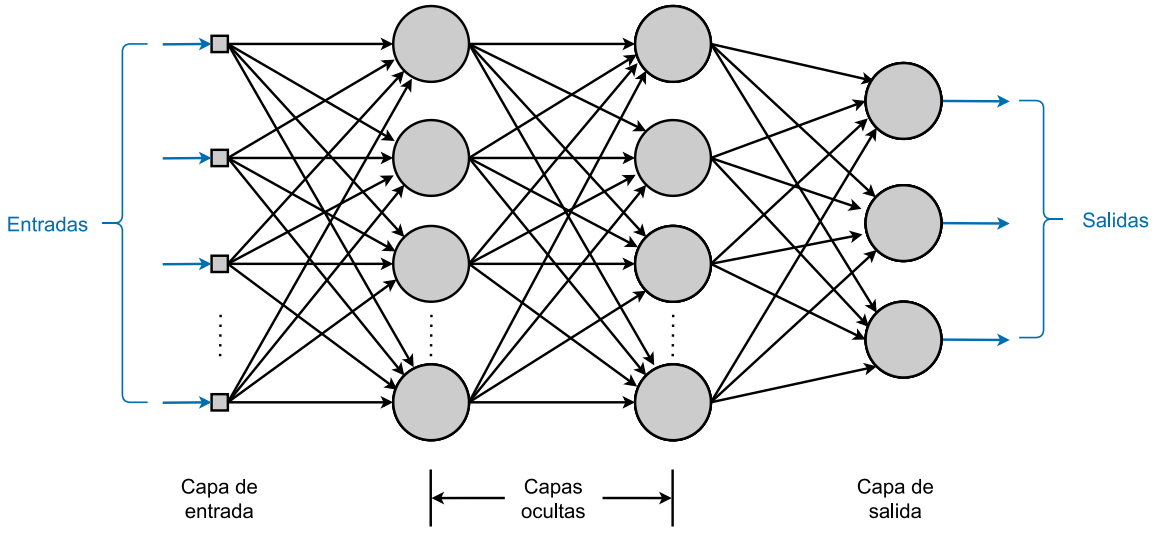


Figura 2.9: Red neuronal artificial multicapa. Cada círculo representa una neurona artificial, o unidad, que recibe un número de entradas para calcular una única salida. Esa salida es una entrada de las neuronas de la siguiente capa. Fuente: [45].

La RNA aprende variando sus parámetros, en particular los pesos, el cual es un proceso de optimización, en el que se busca el mínimo de una función de error, que mide la diferencia

entre lo que la RNA predice para una entrada dada, y el valor que en el conjunto de datos se ha establecido como valor de referencia para esa entrada. Esa función de error depende de los pesos de la RNA y de esa dependencia se calcula el ajuste que cada peso debe sufrir en cada paso del proceso de aprendizaje. Este tipo de RNA que transforma una entrada, pasando capa por capa hacia una salida se denomina "alimentada hacia adelante" (*feed-forward*).

2.4.1. Redes Neuronales Recurrentes

Las redes neuronales recurrentes (RNN) son modelos de aprendizaje diseñados para manejar la complejidad de la dependencia de secuencias temporales, es decir, tienen la capacidad para procesar y obtener información de datos secuenciales. Las RNN utilizan su estado interno para procesar secuencias de entrada de longitud variable [47]. Esto las hace aplicables a tareas como predicción de series temporales, procesamiento de lenguaje humano, codificación y reconocimiento del habla, inferencia gramatical, modelado de sistemas dinámicos, control de sistemas, entre otros [5], [45], [48].

A diferencia de las RNA, en las cuales las capas ocultas no reciben salidas de capas posteriores, lo que se interpreta como que no tienen memoria y se limitan a mapeos de uno a uno, las RNN realimentan salidas posteriores en las entradas, lo que introduce el concepto de "estado oculto", que depende del tiempo. Esto se esboza en la figura 2.10 donde $h(t)$ es el estado oculto de la red. Generalmente, las RNN extienden la arquitectura de una red multicapa alimentada hacia adelante con conexiones hacia atrás. Estas conexiones introducen la memoria interna de la red que se utiliza para complementar las propiedades del dato actual con información del pasado inmediato. Las RNN hacen mapeos de uno a muchos datos, de muchos a uno y de muchos a muchos [45], por ejemplo, subtitulando imágenes, identificando un hablante y traduciendo y transcribiendo el habla, respectivamente.

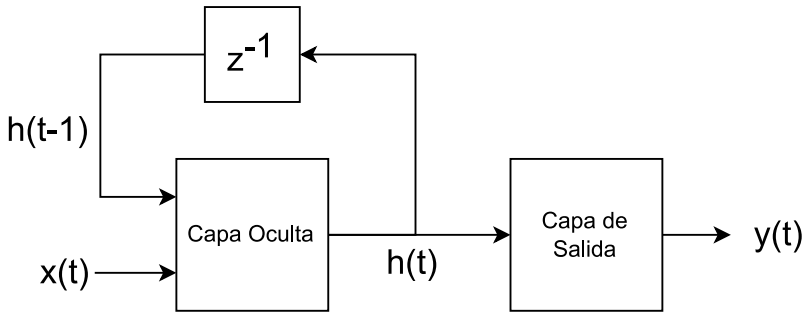


Figura 2.10: Red neuronal recurrente con una capa oculta. Fuente: [49], [50].

Son las conexiones hacia atrás entre capas lo que permite que la salida de una capa afecte a la entrada de la misma capa o de capa anteriores, mostrando así un comportamiento dinámico temporal. Ese comportamiento dinámico se suele representar como un desarrollo en pasos temporales como se muestra en la figura 2.11.

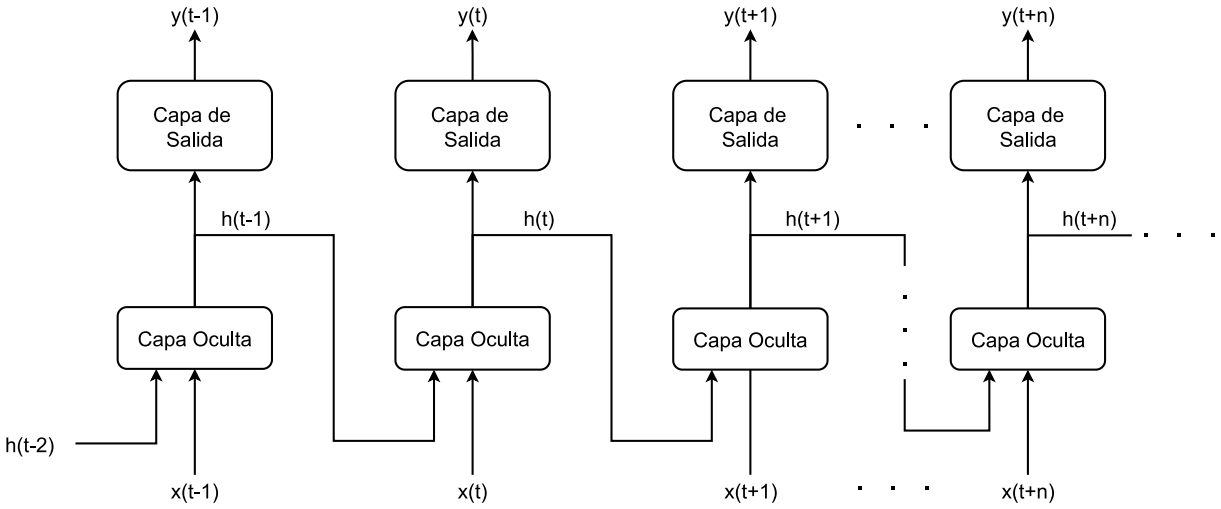


Figura 2.11: Red neuronal recurrente desarrollada. Fuente: [45].

En la figura 2.10 se nota que la salida $\underline{\mathbf{y}}(t)$ está influenciada por el estado $\underline{\mathbf{h}}(t)$. Dado que la entrada de la capa oculta es $\underline{\mathbf{x}}(t)$ y $\underline{\mathbf{h}}(t-1)$, se puede decir que $x(t-1)$ también influye en la salida $\underline{\mathbf{y}}(t)$ e incluso en salidas posteriores. Los parámetros aquí son las matrices de pesos de las capas, que se estiman a partir de los datos de entrenamiento. El estado y la salida se calculan con [5], [48]-[51]:

$$\underline{\mathbf{h}}(t) = \mathbf{W}_h f_h(\underline{\mathbf{h}}(t-1)) + \mathbf{W}_x \underline{\mathbf{x}}(t) + b = \mathbf{W}_H \begin{bmatrix} f_h\left(\underline{\mathbf{h}}(t-1)\right) \\ \underline{\mathbf{x}}(t) \\ 1 \end{bmatrix} \tag{2.11}$$

$$\underline{\mathbf{y}}(t) = \mathbf{W}_y f_y(\underline{\mathbf{h}}(t)) + b_y = \mathbf{W}_Y \begin{bmatrix} f_y(\underline{\mathbf{h}}(t)) \\ 1 \end{bmatrix} \tag{2.12}$$

donde $\underline{\mathbf{x}}(t)$ es la secuencia de entrada, $\underline{\mathbf{y}}(t)$ la salida, $\mathbf{W}_h$, $\mathbf{W}_x$ y $\mathbf{W}_y$ son los pesos o parámetros ajustables, $f_{y/h}(\cdot)$ representan funciones de activación no lineales y $\underline{\mathbf{h}}(t)$ corresponde al estado oculto de la RNN. Aquí $\mathbf{W}_H$ y $\mathbf{W}_Y$ son:

$$\mathbf{W}_H = \begin{bmatrix} \mathbf{W}_h & 0 & 0 \\ 0 & \mathbf{W}_x & 0 \\ 0 & 0 & b \end{bmatrix} \tag{2.13}$$

$$\mathbf{W}_Y = \begin{bmatrix} \mathbf{W}_y & 0 \\ 0 & b_y \end{bmatrix} \tag{2.14}$$

Se observa que (2.11) y (2.12) son similares al modelo de espacio de estados (2.4) y (2.5), lo que permite que las RNN se usen en el modelado e identificación de sistemas dinámicos, tanto lineales como no lineales. La capacidad de modelar sistemas dinámicos no lineales con RNN la introducen las funciones de activación no lineales $f_y(\cdot)$ y $f_h(\cdot)$.

Una red neuronal recurrente descrita por el modelo de espacio de estados (2.11) y (2.12), es un aproximador universal de sistemas dinámicos no lineales [49], [50]. Según [49], un sistema dinámico no lineal puede ser aproximado por una red neuronal recurrente con algún grado de precisión deseado y sin restricciones impuestas por el espacio de estados, siempre que la red esté equipada con un número adecuado de neuronas ocultas. Este hecho ha permitido la introducción de las RNN para aplicaciones del procesamiento de señales y control.

Un problema de RNA y RNN es que el proceso de entrenamiento termine en un mínimo local de la función de error. Esa función de error corresponde a una superficie multidimensional no convexa con un mínimo global y múltiples mínimos locales que pueden no corresponder a una solución aceptable del problema [52]. El problema reside entonces en que casi todos los algoritmos de optimización utilizados basados en el gradiente, corren el riesgo de quedar atrapados en esos mínimos locales.

Otro problema corresponde al llamado desvanecimiento de gradiente. Como se ha explicado, las RNN procesan datos secuenciales, donde el resultado final será una función que agregue todos los elementos de la secuencia. A medida que se procesan más elementos, la red tiene problemas en recordar información pasada, por lo que las RNN más sencillas tienen límites en la extensión de las secuencias que pueden aprender, siendo eficaces en rangos cortos [49], [52]. Para resolver este problema se propuso una red denominada *Memoria de Corto y Largo Plazo* (LSTM, por sus siglás en inglés) [53], que es capaz de lidiar con secuencias de mayor longitud. Con el fin de disminuir la complejidad de las redes LSTM surgen las *Unidades Recurrentes con Compuertas* (GRU, por sus siglás en inglés), las cuales son utilizadas para simular el comportamiento de la PAHM en el presente documento.

2.4.2. Unidades Recurrentes con Compuerta (GRU)

La red LSTM parte de la idea de que la red debe ser capaz de aprender a modelar la información temporal, para lo que utiliza *compuertas*. Estas compuertas son controladas por un valor entre cero (bloqueo) y uno (paso), que se regula con un producto, cuánto de la entrada a la compuerta pasa a su salida. El control de cada compuerta se realiza a través de la salida de una red neuronal con una función de activación sigmoidal $\sigma(\cdot)$:

$$\sigma(x) = \frac{1}{1 + e^{-x}} \tag{2.15}$$

Es así como cada compuerta aprende cuánta información debe dejar pasar o bloquear [54]. Las redes LSTM proponen una configuración con tres compuertas que regulan el flujo de dos tipos de estados internos [53].

La GRU fue propuesta en 2014 por Kyunghyun et al. [55], esto con el fin de que cada unidad recurrente capture de forma adaptativa dependencias de diferentes escalas de tiempo. De forma similar a la LSTM, la GRU tiene compuertas que modulan el flujo de

información de la unidad. La GRU es una arquitectura más simple que la LSTM ya que solamente posee un estado, está constituida por dos compuertas en lugar de tres y no posee celdas de memoria separadas [54]. Dichas características han hecho popular el uso de la GRU actualmente.

La figura 2.12 muestra la estructura de la GRU. Esta red trabaja con el vector de estado oculto $h(t)$ y con dos compuertas que modifican el flujo de dicho vector. La primer compuerta es denominada como la *compuerta de reinicio*. Su función es controlar el grado de información que se ignora o añade al nuevo estado oculto $\hat{h}(t)$. Cuando la compuerta de reinicio se acerca a 0, el nuevo estado oculto se ve obligado a ignorar el estado oculto anterior y a reiniciarse haciendo uso únicamente de la entrada actual. Esto permite que el estado oculto elimine cualquier información que se considere irrelevante en el futuro, permitiendo así una representación más compacta. El cálculo del control de esta compuerta se hace con [54]-[57]:

$$\underline{\mathbf{r}}(t) = \sigma\left(\mathbf{W}_r \begin{bmatrix} \underline{\mathbf{h}}(t-1) \\ \underline{\mathbf{x}}(t) \\ 1 \end{bmatrix}\right) \tag{2.16}$$

La segunda compuerta corresponde a la *compuerta de actualización*, la cual controla la cantidad de información que se añadirá del estado anterior $h(t-1)$ al estado oculto actual $\hat{h}(t)$. Esto ayuda a la RNN a recordar información a largo plazo. El control de la compuerta de actualización se calcula con [54]-[57]:

$$\underline{\mathbf{z}}(t) = \sigma\left(\mathbf{W}_z \begin{bmatrix} \underline{\mathbf{h}}(t-1) \\ \underline{\mathbf{x}}(t) \\ 1 \end{bmatrix}\right) \tag{2.17}$$

Cuanto mayor sea el valor de la compuerta, más información del estado oculto anterior se introduce. Se obtiene el estado oculto actual $\hat{h}(t)$ a través de una función de activación de tangente hiperbólica, la cual toma como entrada al resultado obtenido de la compuerta de actualización y la entrada actual [54]-[57]:

$$\underline{\hat{\mathbf{h}}}(t) = \tanh\left(\mathbf{W}_{\hat{h}} \begin{bmatrix} \underline{\mathbf{r}}(t) \odot \underline{\mathbf{h}}(t-1) \\ \underline{\mathbf{x}}(t) \\ 1 \end{bmatrix}\right) \tag{2.18}$$

donde $\odot$ denota el producto de Hadamard. Por último, se estima cuál es la salida de la red y a su vez el nuevo estado $\underline{\mathbf{h}}(t)$ que servirá para las futuras iteraciones. Para ello, se realiza la suma de la información que se dejó pasar del estado oculto anterior y el resultado del producto de Hadamard de la compuerta de actualización con el estado oculto actual $\underline{\hat{\mathbf{h}}}(t)$ [54]-[57]:

$$\underline{\mathbf{y}}(t) = \underline{\mathbf{h}}(t) = (1 - \underline{\mathbf{z}}(t)) \odot \underline{\mathbf{h}}(t-1) + \underline{\mathbf{z}}(t) \odot \underline{\hat{\mathbf{h}}}(t) \tag{2.19}$$

La ventaja de la GRU es que cada red oculta tiene compuertas de reinicio y actualización separadas. Cada red oculta aprende a captar las dependencias en diferentes escalas de tiempo. Las dependencias a corto plazo tenderán a activar con mayor frecuencia la compuerta de reinicio. Al contrario de esto, las dependencias a largo plazo producirán mayor activación de la compuerta de actualización [55].

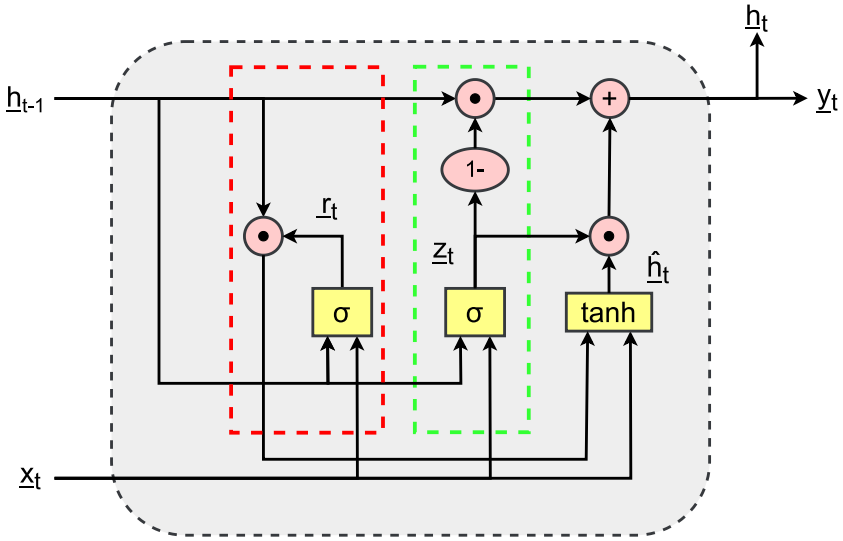


Figura 2.12: Estructura de la red neuronal GRU. La línea punteada de color rojo demarca la compuerta de reinicio y la de color verde la compuerta de actualización. Fuente: [56], [57].

2.5. Métricas de evaluación

Cada modelo de ML intenta resolver un problema con objetivos específicos utilizando conjuntos de datos particulares de la aplicación y, por lo tanto, se debe comprender el contexto antes de elegir una métrica. Para problemas de regresión, se utilizan las métricas de *error absoluto medio (MAE)* y *error cuadrático medio (MSE)*. Estas métricas miden la distancia entre el valor numérico predicho y el valor numérico de referencia.

Como su nombre indica, el MAE realiza un promedio de las diferencias absolutas entre los valores teóricos x_i y los valores predichos u obtenidos $\hat{x}_i$ [58]:

$$MAE = \frac{1}{n}\sum_{i=1}^{n}|x_i - \hat{x}_i| \tag{2.20}$$

Por otro lado, se tiene el MSE, cuya diferencia es que realiza el promedio de diferencias entre los valores teóricos x_i y los valores predichos u obtenidos $\hat{x}_i$ elevados al cuadrado [58]:

$$MSE = \frac{1}{n}\sum_{i=1}^{n}(x_i - \hat{x}_i)^2 \tag{2.21}$$

Capítulo 3

Red neuronal artificial mimetizadora (RNAM)

Para llevar a cabo la implementación del controlador basado en aprendizaje reforzado se debe desarrollar primeramente la RNAM, la cual es necesaria debido a que esta red cumple funciones de seguridad y reducción de tiempos de entrenamiento. Para ello, se dispone del sistema embebido de altas prestaciones, *NVIDIA Jetson TX2*, el cual es capaz de ejecutar el modelo seleccionado y a su vez impone limitaciones a la complejidad de los algoritmos a seleccionar. Por otro lado, en cuanto al PAHM posee un *kit CY8CKIT-059 PSoC 5 LP* que es utilizado para la comunicación con el sensor de cuadratura y el motor. Estos sistemas embebidos se seleccionan debido a su directa disponibilidad para el proyecto.

3.1. Conexión Jetson TX2 y PAHM

El sistema principal encargado de generar las señales de entrada al PAHM y de recopilar la información del sensor de ángulo es la Jetson TX2. Este se comunica con la PAHM a través del PSoC, donde el diagrama de comunicación se muestra en la figura 3.1. La Jetson TX2 transmite el valor PWM necesario para movilizar el motor, el PSoC recibe dicho valor y lo adecua al protocolo *PWM One Shot 125* y lo transmite al ESC como una tensión $v(t)$. El ESC es el encargado de convertir lo recibido a una señal de mayor potencia $p(t)$ para que el motor funcione. Este motor aplica una fuerza $f(t)$ sobre el péndulo que provoca que se desplace de su punto de equilibrio, lo que causa una inclinación del péndulo $\theta(t)$. El ángulo generado es medido por el sensor de cuadratura $\hat{\theta}(t)$, cuya señal es captada por el PSoC y transmitida de regreso a la sistema Jetson TX2.

Se utilizan dos sistemas embebidos aparte por dos razones. Primero, para dedicar la capacidad computacional del sistema Jetson TX2 a las redes neuronales artificiales; y segundo, para dedicar el sistema PSoC a tareas de bajo nivel para las que está mejor equipado que el primero; por ejemplo, para el manejo de la modulación PWM y la lectura de bajo nivel del sensor angular.

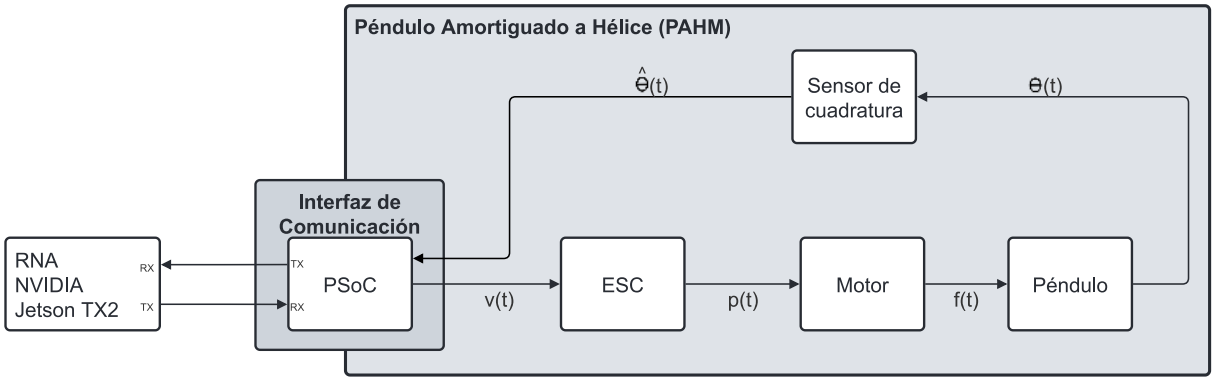


Figura 3.1: Diagrama de comunicación entre la Jetson TX2 y el PSoC 5 LP.

Para la transmisión de información entre los dos sistemas embebidos; es decir, el valor PWM para el control de velocidad y el ángulo medido por el sensor, se utiliza el protocolo de comunicación UART, elegido debido a su facilidad de conexión, es asíncrono lo que agiliza la transmisión y recepción de datos sin la necesidad de un reloj, posee *bits* de paridad que permiten detectar errores en la comunicación y es un protocolo documentado y utilizado. Además, los sistemas embebidos poseen integrado dicho protocolo, así como puertos designados a dicho propósito. Para la transmisión de datos se asigna una tasa de transmisión de 115200 bit/s, no se activa el *bit* de paridad y no se utiliza control de flujo de software o de hardware por tratarse de una conexión local de poca distancia y de bajo riesgo de interferencia en el canal de comunicación. De esta forma, solo se tienen las líneas de transmisión, de recepción y referencia a tierra.

3.1.1. Implementación de hardware

La Jetson TX2 tiene tres conectores de pines: J17, J21 y J30 [38]. De ellos, J17 es utilizado para la comunicación serial, específicamente el protocolo UART; sin embargo, el conector J21 también posee pines para dicho protocolo, así como pines de entrada y salida de propósito general (GPIO, por su siglas en inglés). En este caso, se hace uso de J17 denominado como conector de puerto serial, el cual posee 6 pines cuyas funciones se listan en la tabla 3.1.

El sistema embebido PSoC 5 LP, posee dos conectores de pines que disponen de puertos GPIO, de alimentación, de tierra y de comunicación serial, tal como se ilustra en la figura 3.2. Destacan los pines 12.6 y 12.7 correspondientes al transmisor y receptor del protocolo UART respectivamente. En cuanto a los pines para el ESC y el sensor de cuadratura, se utilizan los pines GPIO que son asignados por software mediante la aplicación *PSoC Creator*, los cuales se resumen en la tabla 3.2.

Tabla 3.1: Descripción del conector de 6 pines J17 de puerto serial para el sistema Jetson TX2. Fuente: [38].

# Pin	Nombre del Pin	Uso/Descripción
1	-	Tierra (*Ground*)
2	UART1_RTS#	Solicitud de envío (*Request to send*)
3	-	Sin Uso
4	UART1_RX	Receptor
5	UART1_TX	Transmisor
6	UART1_CTS#	Listo para envío (*Clear to send*)

Tabla 3.2: Pines asignados para el PSoC 5 LP mediante *PSoC Creator*.

Nombre	Puerto	Pin
LED_EJECUCION	P1[2]	13
PWM_1_OUT	P1[7]	19
Q1_A	P2[1]	63
Q1_B	P2[2]	64
Rx_1	P12[6]	20
Tx_1	P12[7]	21

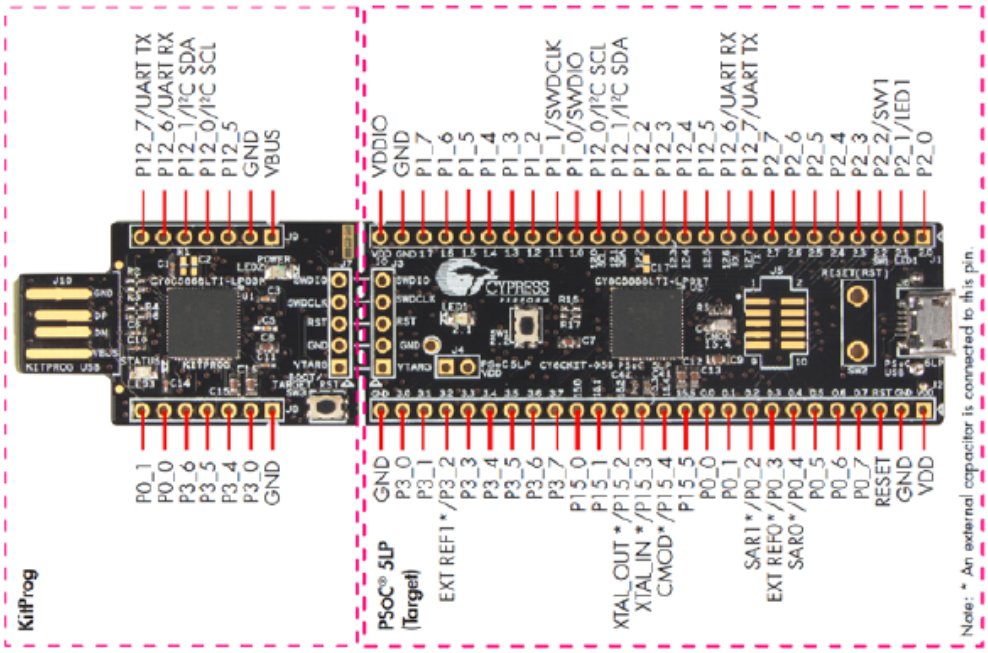


Figura 3.2: Detalles de los pines del kit de prototipo PSoC 5LP. Fuente [33].

3.1.2. Implementación de software

Para operar los sistemas embebidos se utilizan los lenguajes de programación C y Python, que permiten la obtención de datos y la comunicación serial. Para el sistema Jetson TX2, que utiliza el sistema operativo Ubuntu Linux, se hace uso del lenguaje Python debido a que proporciona bibliotecas incorporadas que abstraen la capa física del hardware, además de que considerado el lenguaje *de-facto* para tareas de aprendizaje automático [59], lo que es beneficioso para el desarrollo de la RNAM. Por otro lado, el PSoC 5 LP está diseñado para utilizar y ser programado en lenguaje C bajo la aplicación de *PSoC Creator*. Dicho lenguaje es más veloz que Python y permite acceder directamente al hardware del sistema.

Programación PSoC 5 LP

Como se ha mencionado, la programación del PSoC se hace mediante una aplicación dedicada que permite visualizar los recursos disponibles. Se utiliza un modulador PWM digital, el cual se configura para producir una señal con las características del protocolo *PWM One Shot 125*, con un periodo de activación de 20 ms. Usualmente, el bloque de PWM se conecta a una señal de reloj de 1 MHz permitiendo manejar el periodo entre 125 μs y 250 μs, sin embargo, este valor del reloj limita el valor máximo de pasos a 125, lo que es limitado para la función que se quiere realizar. Por ello, el valor de la señal de reloj se cambia a 8 MHz, aumentando así la cantidad máxima de pasos a mil.

Utilizando la función de transferencia (2.3) se estima la respuesta en frecuencia del PAHM, tal como se muestra en la figura 3.3. La frecuencia de corte de potencia mitad (-3 dB) es de 1,0465 Hz. Una buena práctica en control es utilizar una frecuencia de muestreo diez o veinte veces mayor, obteniendo una frecuencia de 20,93 Hz que se redondea a 25 Hz, siendo la frecuencia de corte. Se aplica el teorema de Nyquist obteniendo una frecuencia de muestro a utilizar de 50 Hz.

Adicionalmente, se tiene un bloque de decodificador de cuadratura digital y un temporizador (*timer*), donde se define que el periodo de muestreo del PAHM es de 20 ms. Asimismo, la frecuencia de muestreo es de 50 Hz, la cual se encuentra por encima de la frecuencia de Nyquist pero a su vez es lo suficientemente baja para que ambos sistemas embebidos tengan tiempo de realizar sus tareas, por cada muestra capturada, y que el ESC reaccione a los valores recibidos. Además, en un futuro permitirá generar una respuesta por parte del controlador basado en aprendizaje reforzado.

Una señal de reloj de 1 MHz se conecta a los relojes internos de estos dos bloques para satisfacer el periodo de muestreo asignado. Se conecta un bloque de interrupción cuya función es interrumpir el sistema para la lectura y escritura de datos; en este caso, para escribir el valor PWM recibido al ESC y leer el valor del ángulo del sensor del PAHM, ambos cada 20 ms. A ello se asigna un diodo LED que sea testigo de que la interrupción sea activada y verificar su funcionamiento. Cabe mencionar que los bloques PWM, decodificador y temporizador poseen una resolución de 16 *bits*. Por último, se tiene el bloque

UART, cuya configuración es un una tasa de 115200 *bit*/s, 1 *byte* a transmitir, un *bit* de parada y no hay *bit* de paridad ni control de flujo. El diagrama de conexión se presenta en la figura 3.4 y en la tabla 3.2 se encuentran los pines asignados.

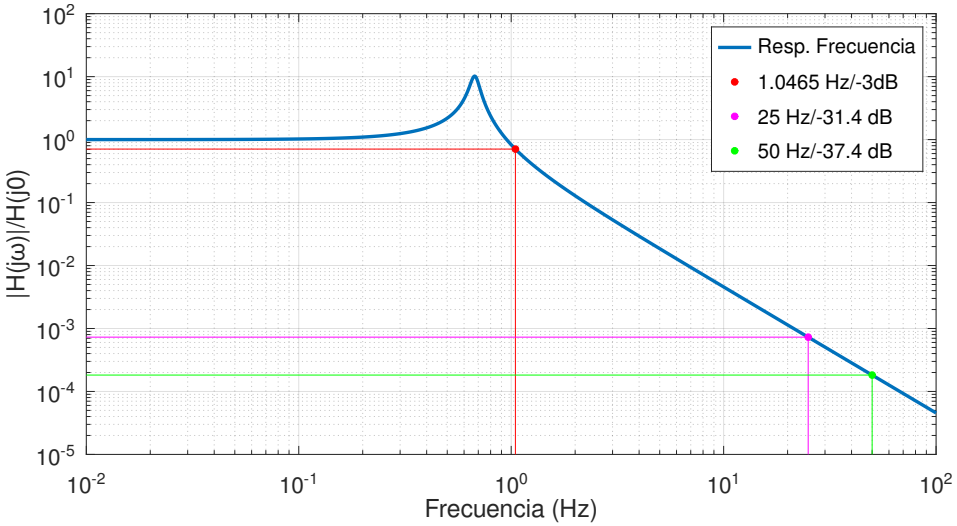


Figura 3.3: Magnitud de la respuesta en frecuencia del PAHM. De color rojo se muestra la frecuencia de potencia mitad, de color magenta se muestra la frecuencia de corte y de color verde la frecuencia de muestreo utilizada .

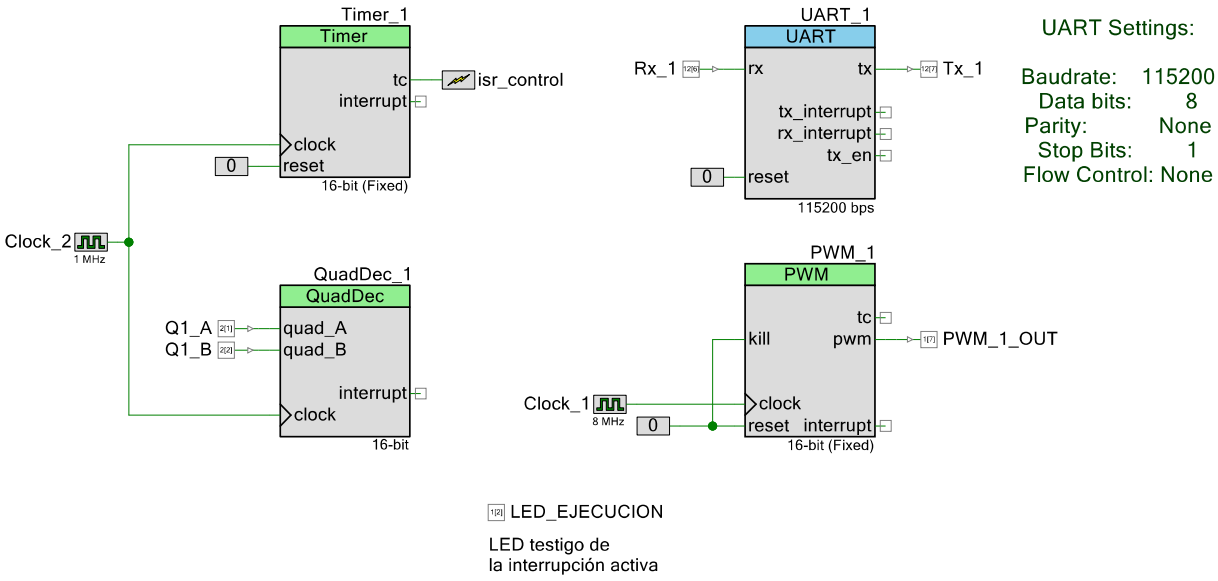


Figura 3.4: Diagrama de conexión de bloques en el PSoC 5 LP.

La figura 3.5 muestra el diagrama de flujo seguido por el programa para el sistema embebido PSoC. Primeramente, se definen variables globales que corresponden al valor PWM y el ángulo del PAHM. Seguidamente, se define la función de interrupción y se inicializan los bloques descritos anteriormente. Se limpian los búferes de transmisión y recepción y se inicia un ciclo infinito que recibe dos *bytes* correspondientes al valor PWM. El almacenamiento de los dos *bytes* en el PSoC usa otro *endianness* a como los recibe el sistema

Jetson TX2 por lo que es necesario reordenarlos. Además, se transmite el ángulo en cuatro *bytes*. Por último, el programa se detiene por un 20 ms antes de reiniciar el ciclo.

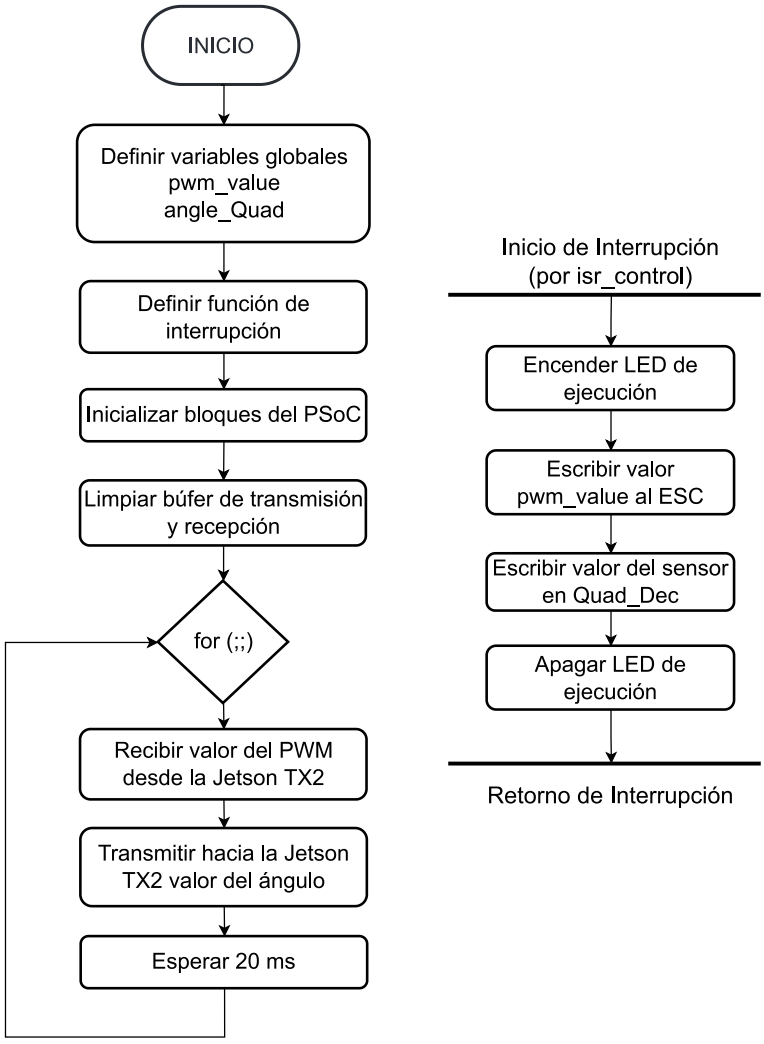


Figura 3.5: Diagrama de flujo del programa ejecutado en el sistema PSoC 5 LP.

Programación Jetson TX2

El sistema embebido de altas prestaciones es el encargado de la generación de señales y la recopilación de datos provenientes del PAHM para el entrenamiento de la RNAM, a través de la transmisión y recepción de datos con el PSoC. Para llevar a cabo esta tarea se diseñan cuatro módulos: un temporizador, un comunicador, un convertidor de datos y un recolector de datos. La interconexión de dichos módulos se muestra en la figura 3.6

Debido a que el PAHM opera con muestreo homogéneo cada 20 ms se necesita que el código se ejecute y dé respuesta en ese lapso de tiempo concreto. Por ello, es necesario un temporizador que llame una función cada determinado tiempo, encargada de transmitir y recibir valores en el tiempo requerido por el PAHM.

El temporizador utiliza un hilo secundario que está llamando a la función encargada de la

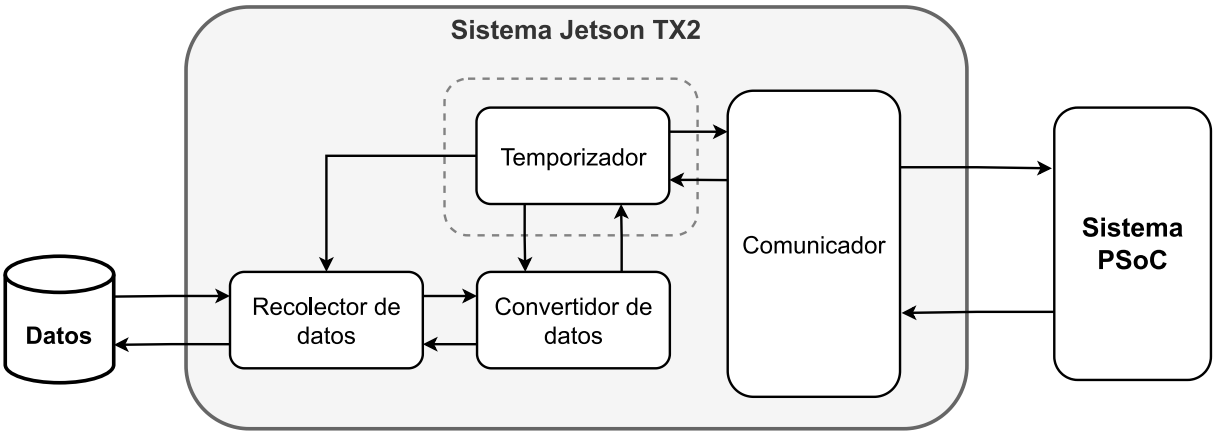


Figura 3.6: Interconexión de los módulos de la Jetson TX2.

transmisión y recepción de datos. En ese hilo secundario se vuelve a calcular el tiempo de cuándo debe volver a llamarse la función, haciendo que se repita el proceso cada 20 ms. Esto se encapsula en tres funciones, donde *start()* es la función que el hilo principal llama para arrancar un hilo secundario. La función que ejecuta ese hilo secundario es *_run()*, donde se hace el llamado a la función encargada de la comunicación y se vuelve a calcular el tiempo de llamada. El hilo principal puede detener el hilo secundario con la función *stop()*.

El módulo de conversión de datos tiene la tarea de transformar los datos desde y hacia el PAHM al formato utilizado por la RNAM y viceversa. El valor del PWM se normaliza entre cero y uno para asegurar la estabilidad de las RNAM y acelerar su aprendizaje. No obstante, el valor a transmitir al PSoC como valor PWM debe desnormalizarse, lo que se encapsula en una función *denormalizerPWM()*.

El módulo de comunicación se encarga de transmitir y recibir datos hacia y desde el PSoC. Este módulo utiliza dos funciones: la función *reset()* que limpia los búferes de entrada y salida para evitar que permanezcan datos residuales de anteriores transmisiones y perjudique la lectura de los datos. Posterior a ello, la función *Transmit_Receive()*, la cual se encarga del proceso de transmisión referido a empaquetar el valor PWM en dos *bytes* y transmitirlo al PSoC, y el proceso de recepción referido a desempaquetar los cuatro *bytes* recibidos desde el PSoC correspondientes al valor del ángulo medido por el sensor. Estos dos valores se conservan en un arreglo para uso del siguiente módulo.

Por último, el módulo de recolector de datos tiene como función recopilar y reinsertar los datos enviados y recibidos vía PSoC hacia y desde el PAHM. Además, se definen dos tipos de entradas: la *manual*, que, por medio de una interfaz gráfica con un deslizador permite cambiar manualmente el valor PWM a transmitir y almacenar para futuras reproducciones. Por otro lado, se tiene la entrada de *reproducción* (*playback*). Este tipo de entrada reutiliza la misma entrada de un experimento previo, para de este modo reproducir y así evaluar el efecto en la salida producido por factores variantes en el tiempo del PAHM ante la misma secuencia de entrada, así como determinar el promedio y la desviación estándar de las respuestas entregadas en repeticiones experimentales. Otra funcionalidad

es recolectar los datos y ponerlos a disposición en un archivo .csv con el fin de utilizarlos para el entrenamiento de la RNAM, lo cual se encapsula en la función *csv_doc()*.

Adicionalmente a los módulos, se realizan dos funciones de seguridad para evitar daños en la planta física: la función *turn_off()* se encarga de asignar un valor de cero al PWM con el fin de apagar el motor de la hélice en caso de emergencia y la función *pwm_set_safe_value()*, que actúa como un saturador evitando que el valor de PWM exceda un umbral de seguridad. En este caso, se determina experimentalmente que 0,25 es el valor de umbral normalizado que evita que el péndulo gire alrededor de su eje.

En la figura 3.7 se muestra el diagrama de flujo completo para el sistema Jetson TX2. Primeramente, se verifica si existe un nombre de archivo de entrada y salida. Luego, se define el puerto serial y la configuración utilizada; se utiliza en la Jetson TX2 al puerto /dev/ttyTHS2. En caso de definir un nombre de archivo de entrada, implica que el usuario desea reproducir un archivo de entrada, por lo que se procede a leer el archivo .csv correspondiente. En caso contrario, se aplica la entrada manual. En ambos casos se toma el valor PWM, se desnormaliza y se transmite. Simultáneamente, se recibe el valor del ángulo, cuyos valores se almacenan en un arreglo. Esto se realiza durante la ejecución del temporizador para asegurar que suceda cada 20 ms. Una vez acabada la transmisión se apaga el PAHM y se guardan los datos en un archivo .csv. Los códigos desarrollados para la implementación de software están a disposición en el sistema de control de versiones en [60].

3.2. Planta sintética

Como se menciona en la sección 2.1.2, los modelos matemáticos lineales describen en un rango de operación limitado el comportamiento de sistemas dinámicos, lo que da la posibilidad de predecir su comportamiento. Para el desarrollo de la solución se usa un modelo lineal para corroborar, bajo condiciones teóricas, que una RNAM es capaz de aprender el comportamiento descrito por ese modelo. Debido a que se desconocen parámetros como masas, longitudes, entre otros, necesarios para el modelado teórico del PAHM, se realizan pruebas de caja negra, considerando el PAHM como dicha caja. A partir de un estímulo en la entrada, se genera una reacción en la salida, con el fin de encontrar los parámetros que describan su comportamiento, como se muestra en la figura 3.8.

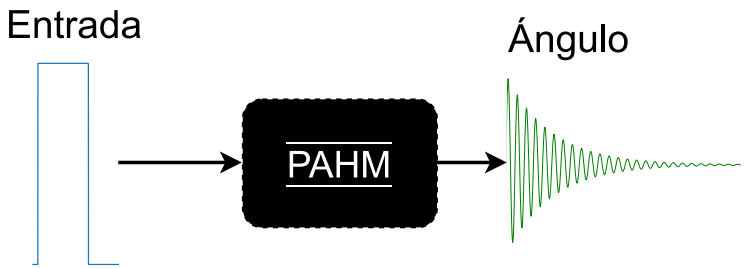


Figura 3.8: Ejemplificación del PAHM como caja negra.

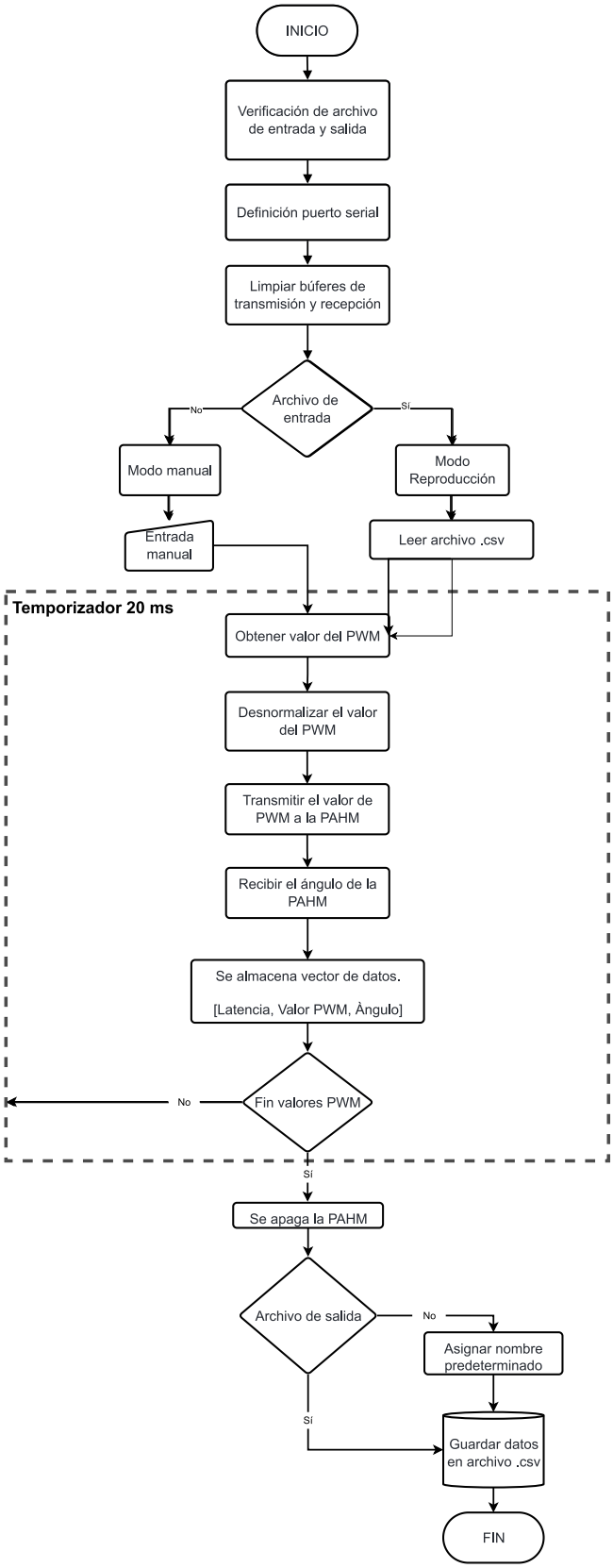


Figura 3.7: Diagrama de flujo del programa ejecutado en la Jetson TX2.

El comportamiento dinámico del PAHM se describe a través del modelo lineal analítico (2.3). No obstante, se desconocen los parámetros denominados como frecuencia fundamental, el factor de amortiguamiento y la ganancia, donde sus valores se determinan mediante el uso de métodos numéricos con el fin de encontrar los valores óptimos del sistema mediante la disminución de la diferencia entre la respuesta real del PAHM y la respuesta del modelo. Por lo tanto, se parte de realizar una identificación del sistema, es decir, a partir de un conjunto de datos del PAHM real se determina el valor de los parámetros del modelo. Se estimula entonces la PAHM con una señal escalón y se obtiene la respuesta del ángulo, donde se usa un valor normalizado del escalón de 0,1 para asegurar que el modelo lineal aproxima el comportamiento de la planta PAHM real.

Con los datos de entrada y salida se utiliza el sistema Matlab, que incorpora la herramienta *ident* para derivar modelos matemáticos de forma empírica. Los datos se separan en dos subconjuntos: uno asignado a la estimación del modelo matemático y otro para la validación de dicho modelo, como se muestra en la figura 3.9.

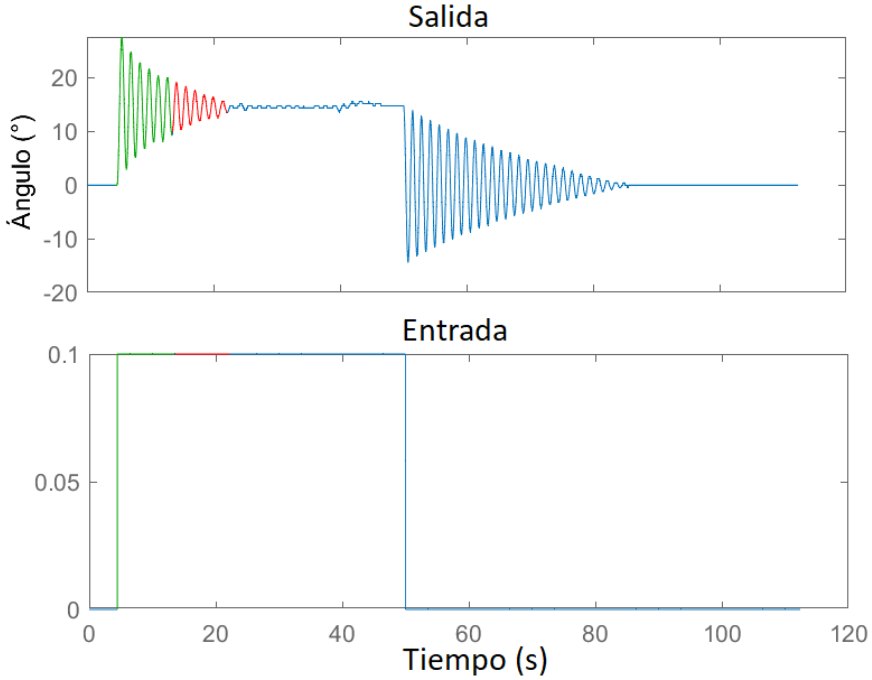


Figura 3.9: Rango de datos de estimación y validación. De color verde se encuentran los datos de estimación y de color rojo los datos de validación.

3.3. Modelado de la RNAM

El proceso de diseño de la RNAM tiene tres etapas: el preprocesamiento de los datos, selección e implementación del modelo neuronal y su entrenamiento, y, por último, la variación de hiperparámetros con el fin de encontrar valores óptimos que ayuden a mejorar el aprendizaje y predicción de la red neuronal. Estas etapas se resumen en la figura 3.10, donde el interruptor indica que no siempre se realiza variación en los hiperparámetros.

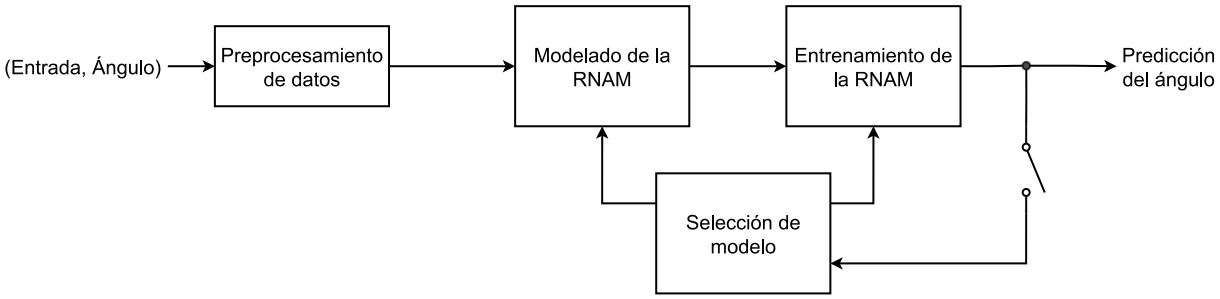


Figura 3.10: Diagrama de etapas para la elaboración de la RNAM.

3.3.1. Preprocesamiento de los datos

Para recopilar datos reales se estimula la entrada del PAHM con señales manuales que abarquen todo el espacio de operación, sin sobrepasar límites que puedan dañar la planta. Dichas entradas se repiten un mínimo de tres veces mediante el modo de reproducción con el fin de obtener múltiples respuestas de la PAHM ante una misma entrada. Esto permite verificar la presencia de efectos que provocan variaciones en la salida. Asimismo, esto permite aumentar la cantidad de datos para la RNAM, donde se procura que haya suficiente cantidad de datos para que la red no se sesgue a una predicción y con variedad en su entrada y salida para evitar el sobreajuste.

La normalización de valores ayuda a la redes neuronales a que converjan más rápido, sean más precisas en las predicciones y se reduce la posibilidad de que la superficie de error esté patológicamente deformada. Las entradas y salidas se normalizan entre 0 y 1, para lo cual se utiliza:

$$x_{normalizado} = \frac{x_i - x_{\text{mín}}}{x_{\text{máx}} - x_{\text{mín}}} \tag{3.1}$$

donde x_i es el valor actual, $x_{\text{mín}}$ el valor mínimo de los datos y $x_{\text{máx}}$ el valor máximo. Para el caso del valor PWM sin normalizar, los valores mínimo y máximo corresponden a 1000 y a 1250 respectivamente. Mientras para la salida, es decir, el ángulo, se utiliza un valor mínimo y máximo de $-120°$ y $120°$ respectivamente, cuyo valor fue definido al observar las respuestas obtenidas de los datos recopilados.

Los datos normalizados se separan en subconjuntos correspondientes a datos de entrenamiento, validación y prueba, donde el 60 % se le asigna a los datos de entrenamiento del conjunto total de datos. A los datos de validación y prueba se les asigna un 20 % del conjunto total de datos para cada uno. Los marcos de trabajo para las redes neuronales trabajan con tensores, cuya estructura depende de cada capa de la red.

Para el desarrollo de la RNAM se usa una red GRU, la cual ocupa un tensor en específico con la forma *[tamaño de lote, tamaño de secuencia, características]*. Para la entrada, el tamaño de lote (*batch_size*) usado es de uno; el tamaño de la secuencia (*timesteps*) corresponde a la cantidad de datos que se tiene para cada subconjunto; y las característi-

cas (*features*) tienen dimensión dos. Por lo tanto, el vector con los datos de entrada se transforma a un tensor con dimensiones $(1, N, 2)$. Ahora bien, la salida o las etiquetas, a diferencia de la entrada, utilizan una dimensión de uno en las características, correspondiendo una única respuesta por cada dato de entrada, por lo que el vector de etiquetas se transforma a un tensor con dimensiones $(1, N, 1)$.

Al tensor se le agrega el estado inicial de la planta, el cual es cero debido a que parte del reposo. Un ejemplo de representación de estos tensores es:

$$train\_data[0, :, :] = \begin{bmatrix} u_0^{(1)} & u_0^{(2)} \\ x_0 & 0 \\ x_1 & 0 \\ \vdots & \vdots \\ x_n & 0 \end{bmatrix} \tag{3.2}$$

donde la primer fila $u_0^{(1,2)}$ denota los estados iniciales. En la columna uno, a partir de la segunda fila hasta la n-ésima fila; es decir, x_0 hasta x_n, se colocan las entradas. La segunda columna posee valores de cero, con el fin de mantener constante la dimensión del tensor. En la figura 3.11 se resumen las etapas del preprocesamiento de datos.

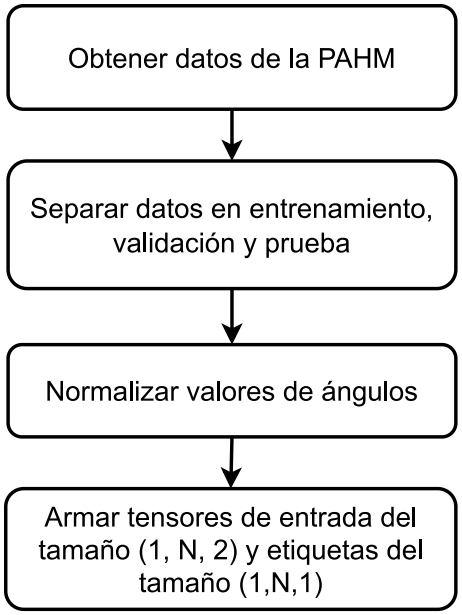


Figura 3.11: Etapas del preprocesamiento del conjunto de datos.

3.3.2. Arquitectura de la RNAM

Para la RNAM se selecciona el modelo *GRU*. Dicha red se elige por dos razones: su estructura matemática es apta para el campo de modelado y simulación de sistemas dinámicos, y su cantidad de parámetros y entradas es menor que el modelo LSTM, lo que lleva a menor tiempo de cálculo y entrenamiento.

La arquitectura de la RNAM consiste en una capa GRU y una capa densa. La capa GRU es la encargada de aprender la dinámica del sistema, donde se define el tamaño de entrada de esta capa como flexible con (*None*, 2), puesto que de otro modo a todos los vectores de entrenamiento, validación y prueba se les exigiría tener la misma cantidad de datos. Por otro lado, el número de columnas corresponde a la dimensión ya establecida en (3.2).

La salida de la capa GRU, que representa su estado interno, se conecta a la capa densa, la cual posee solo una neurona que se encarga de hacer la regresión del valor del ángulo a predecir, mediante la sumatoria de las entradas y su producto con el peso aprendido por la capa, como muestra en (2.8), donde se utiliza una capa de activación lineal. Para entrenamientos, se incorpora una capa de pérdida con las métricas MSE o MAE y se utiliza el optimizador Adam. La arquitectura final de la red se muestra en la figura 3.12.

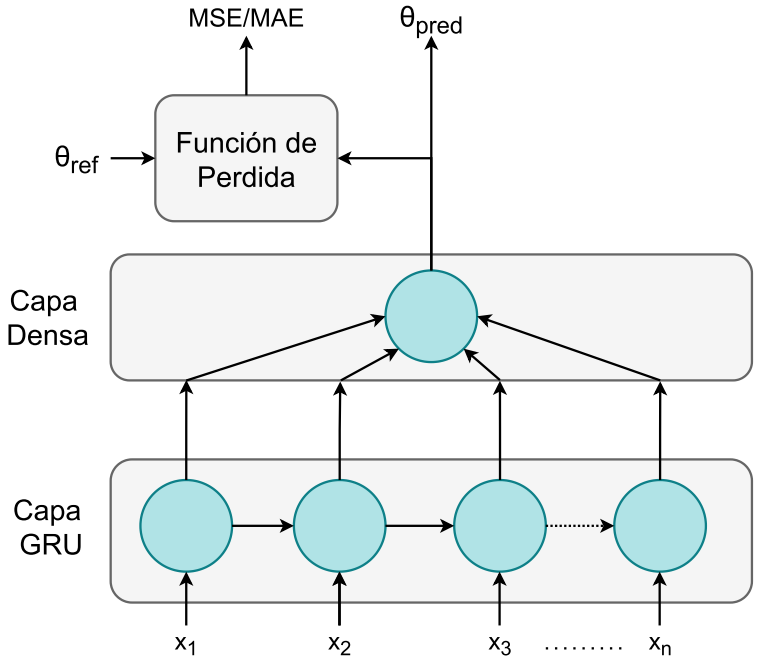


Figura 3.12: Arquitectura de la RNAM.

3.3.3. Selección de modelo

En el aprendizaje automático se denomina "selección de modelo" al proceso de variar los hiperparámetros y arquitectura de la red con el fin de mejorar la respuesta o predicción entregada por esta. Para ello, se modifica la capa GRU donde se elige la dimensión del espacio de estados y el tamaño de lote como los hiperparámetros. Además, se realiza un preentrenamiento, tomando esto como el entrenar una RNAM inicializando los pesos con los valores de otra red previamente entrenada con diferente función de pérdida.

En cuanto a las dos métricas de error se utilizan de forma que, en el primer entrenamiento se use MSE, la cual estimula la reducción de errores grandes de la red. Cuando el entrenamiento finaliza se cambia la métrica por MAE y se vuelve a entrenar, con la diferencia que se inicializan los pesos con los de la red entrenada anteriormente con MSE. Esto permite

reducir los errores pequeños, disminuyendo la pérdida de la red y obteniendo predicciones aproximadas al comportamiento dinámico de la planta PAHM real.

Del mismo modo, se realiza la prueba de usar dos capas GRU en la RNAM, dando mayor capacidad de aprendizaje temporal a cambio de mayor complejidad en el entrenamiento. De igual forma, se realiza un preentrenamiento con este modelado de la RNAM. En la figura 3.13 se muestra su arquitectura neuronal.

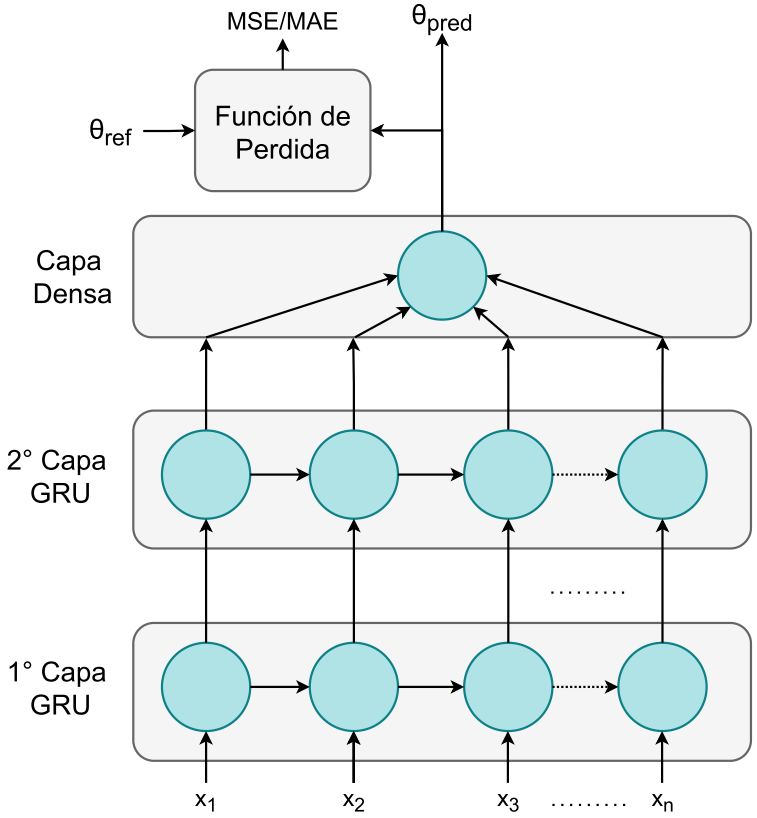


Figura 3.13: Arquitectura de la RNAM con 2 capas GRU.

Los modelos de RNAM propuestos se ponen a prueba con un escenario sintético y otro real. El escenario sintético, denominado como *RNAM sintética*, utiliza el modelo lineal empírico con el fin de evaluar si la red neuronal puede aprender el comportamiento dinámico idealizado del PAHM. Mientras el escenario real, denominado como *RNAM real*, consiste en el uso de datos físicos del PAHM, donde existe influencia de la incertidumbre proporcionada por el sensor de cuadratura.

Para el desarrollo de la RNAM tanto sintética como real se sigue el diagrama de flujo planteado en la figura 3.14. Primeramente, se obtienen los datos de la planta, ya sea física o sintética. Se lleva a cabo el preprocesamiento de dichos datos donde se separan, normalizan y se reconstruyen en tensores. Seguido, se diseña el modelo neuronal y se determina qué métrica se va a utilizar al igual que si se van a inicializar los pesos con algún modelo previo. Se entrena la RNAM, se guarda su modelo y se lleva a cabo la predicción, donde se obtiene un gráfico de pérdida como de predicción. Por último, se evalúa la respuesta con ambas métricas.

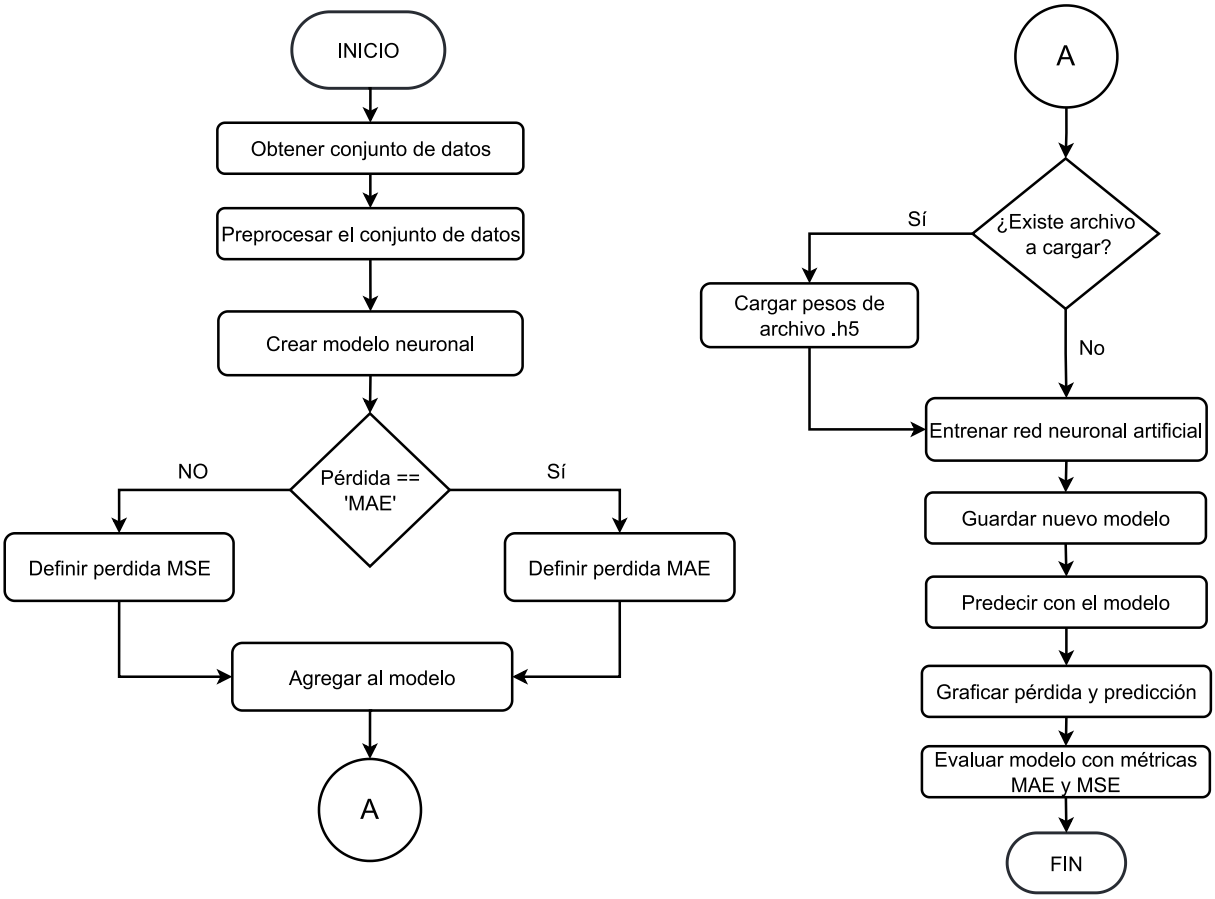


Figura 3.14: Diagrama de flujo para la RNAM física.

Los datos de entrada de la RNAM sintética se crean a partir de secuencias aleatorias. Sus etiquetas se crean a través de una función que discretiza el comportamiento dinámico del PAHM por el método de *backward Euler*, el cual utiliza las matrices del espacio de estados del modelo matemático. El tamaño del conjunto de datos generado corresponde a un total de 3500 datos, el cual se separa en subconjuntos de entrenamiento, validación y prueba con la distribución antes mencionada de 60 %/20 %/20 %. En cambio, la RNAM real al evaluar datos reales del PAHM y con el fin de abarcar los efectos de la planta a través del tiempo, se recopilan una mayor cantidad de datos: el conjunto total de datos tiene 279832 muestras.

Capítulo 4

Resultados y análisis

En el presente capítulo se presentan los resultados obtenidos con el desarrollo del sistema propuesto, centrándose en el desempeño y la capacidad de la RNAM para mimetizar o simular el comportamiento dinámico del PAHM. Para ello, se analiza la RNAM tanto en el escenario sintético como real mediante las métricas de la sección 2.5 y se presentan las gráficas de las predicciones obtenidas con el fin de evaluar los modelos. Asimismo, se lleva a cabo un análisis del PAHM, donde se realizan pruebas de su comportamiento, específicamente de linealidad, bajo la definición de sistemas LTI, con el fin de verificar el alcance del modelo lineal presentado en la sección 2.1. De igual forma, se expone y analiza el modelo empírico lineal obtenido con la herramienta ident, donde se discute el tipo de sistema definido por la ecuación y el porcentaje de error del modelo bajo la métrica MSE. Además, se evalúa la respuesta del modelo comparada con los datos reales del PAHM.

4.1. Comprobación de no linealidad

Primeramente, se lleva a cabo una prueba a fin de verificar el comportamiento no lineal del PAHM. Los sistemas LTI cumplen la condición de superposición, es decir, si $y_1(t)$ es la respuesta del sistema a una entrada $x_1(t)$ y $y_2(t)$ la salida a la entrada $x_2(t)$, la respuesta a $k_1x_1(t) + k_2x_2(t)$ es $k_1y_1(t) + k_2y_2(t)$. Para la prueba, la entrada $x_1(t)$ corresponde a un escalón de magnitud 0,1, mientras que k_1 corresponde a 2, con $x_2(t) = 0$. La respuesta obtenida del PAHM se muestra en la figura 4.1.

Al aplicar un factor de dos en el escalón se espera para un sistema lineal que su salida, $y_2(t)$, esté escalada por un factor de igual magnitud. No obstante, eso no se cumple por lo que el sistema no es lineal. Para comparar el comportamiento en el tiempo se procede a multiplicar la respuesta ante el escalón de 0,1 por un valor de 2,81 que acerca ambas respuestas, lo que se muestra en la figura 4.2. Se logra visualizar más allá de la divergencia en los tiempos de decaimiento, un comportamiento asimétrico en la oscilación hacia arriba y hacia abajo del nivel de convergencia en los primeros 2 s del proceso, atribuido a la no linealidad. Obsérvese que para los valores de ángulo observados como

respuesta a $x_2(t) = 0{,}2u(t)$, la aproximación $\text{sen}(x) \approx x$ utilizada en la derivación del modelo matemático lineal teórico ya no aplica lo que fundamenta el comportamiento no lineal observado.

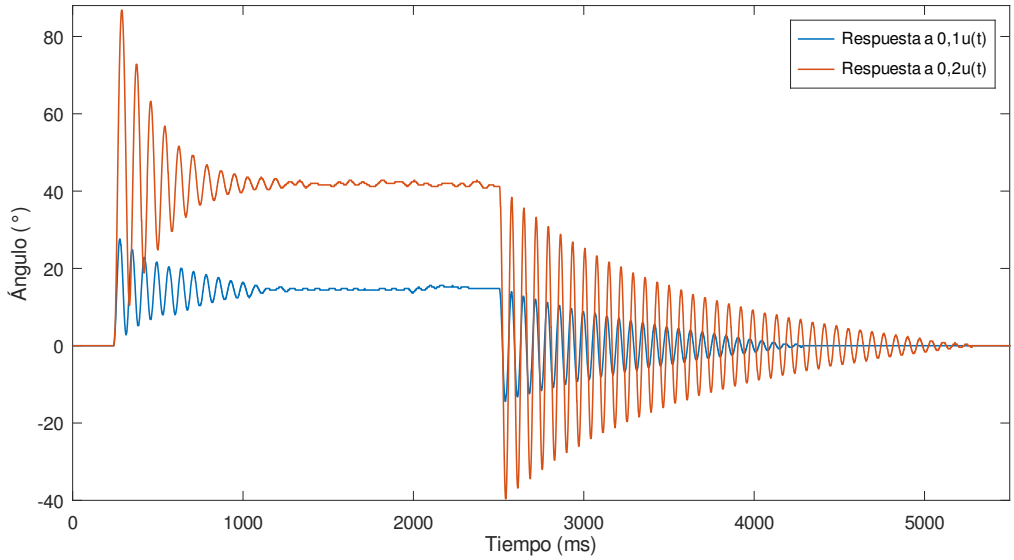


Figura 4.1: Respuesta del PAHM ante las entradas escalón de magnitud 0,1 y 0,2.

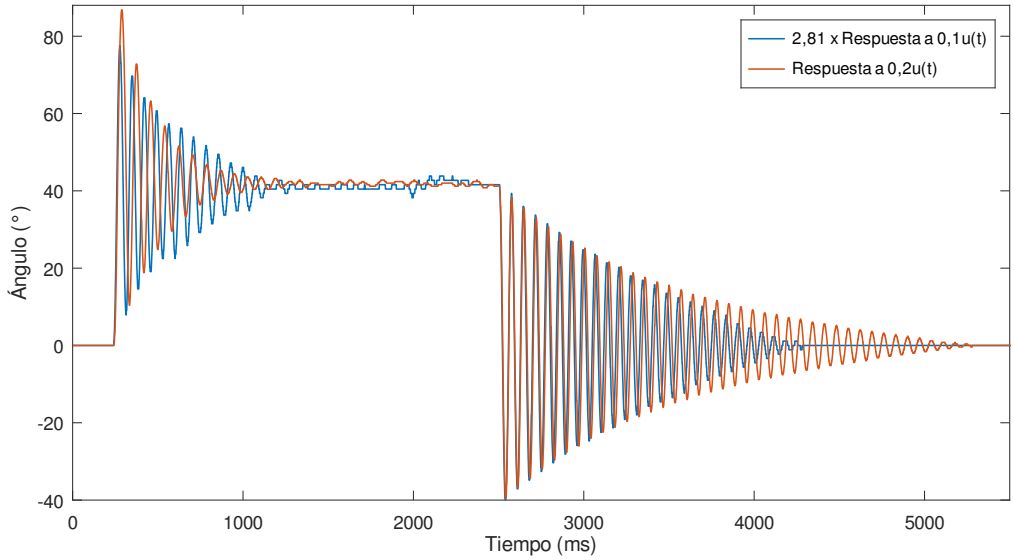


Figura 4.2: Aplicación de un factor ante la respuesta escalón.

Se procede a hacer una prueba de estabilidad, para observar la variación de la respuesta entregada por la planta y registrada por el sensor. Para ello, se estimula la planta diez veces con la misma entrada que se muestra en la figura 4.3. Las respuestas se muestran en la figura 4.4. Mediante la desviación estándar y el promedio de cada punto del conjunto de datos recopilado, se muestra en la figura 4.5 la variación y la tendencia de la planta PAHM. La respuesta promedio se considerada como el valor esperado de la respuesta real

entregada por el PAHM. Se observa que en el primer segundo de activación es donde existe una mayor desviación en los datos capturados y, conforme el tiempo avanza, dicha desviación disminuye, siendo notable solamente en los picos y valles de la respuesta.

Las variaciones se pueden atribuir tanto a perturbaciones en el entorno (como corrientes de aire) o efectos mecánicos de la planta misma. A nivel general, el sistema presenta variaciones en su comportamiento dinámico ante una misma entrada, por lo que el modelo neuronal a explorar debe poder lidiar con la varianza temporal observada.

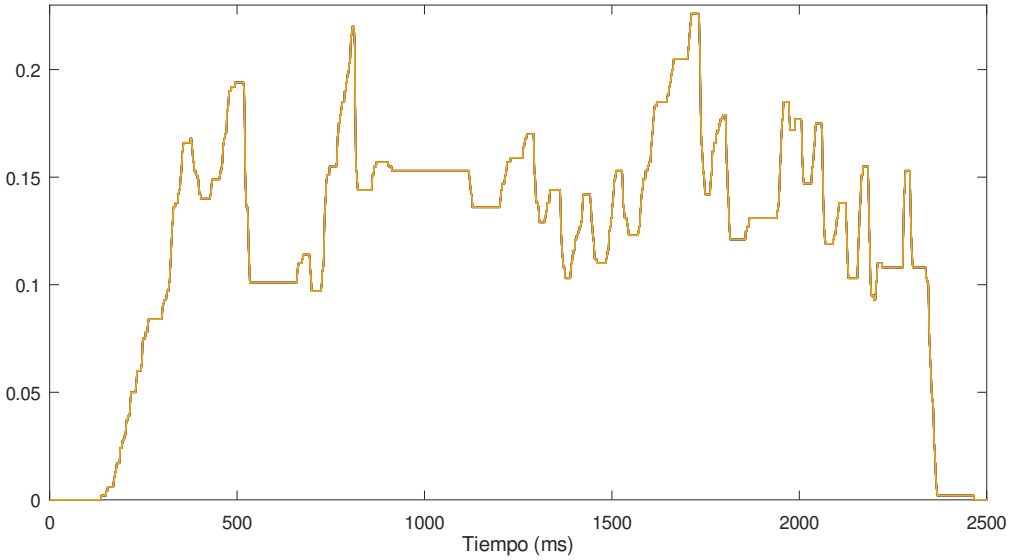


Figura 4.3: Diez entradas para la estimulación del PAHM.

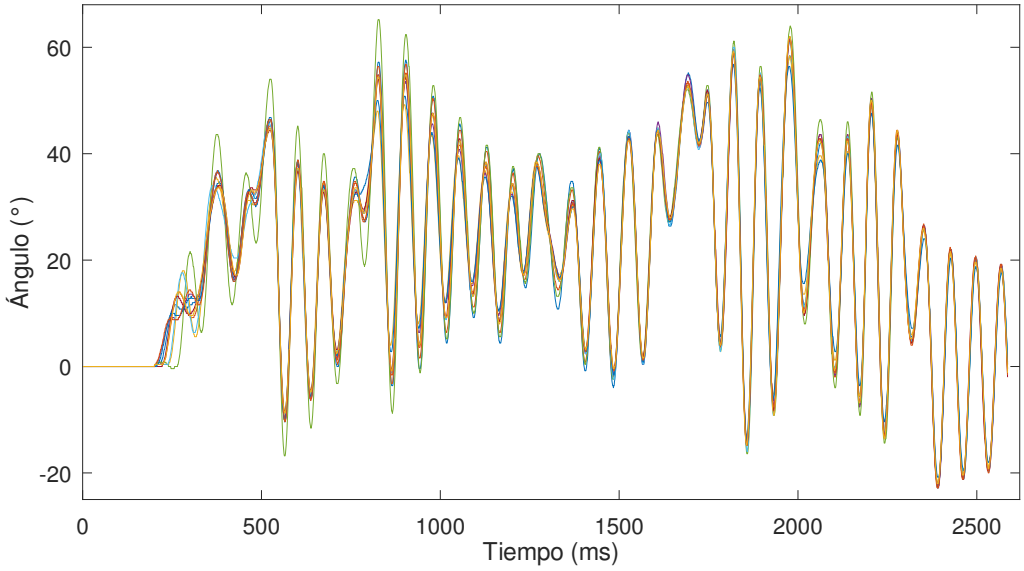


Figura 4.4: Diez respuestas entregadas por la PAHM ante la entrada de la figura 4.3.

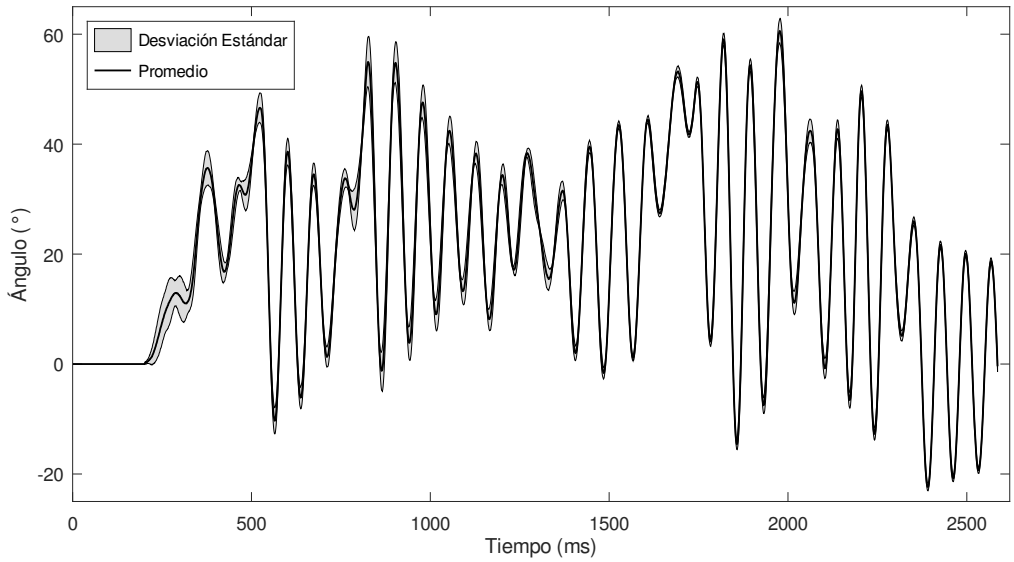


Figura 4.5: Promedio y desviación estándar de las respuestas del PAHM en la figura 4.4.

4.2. Modelo empírico lineal

Mediante la herramienta de ident se determina el modelo empírico lineal de la planta PAHM representado por:

$$H(s) = \frac{\theta(s)}{V(s)} = \frac{2965}{s^2 + 0{,}4151s + 17{,}97} \tag{4.1}$$

donde para los datos de salida capturados se usó como entrada $v(t) = 0{,}1u(t)$. Con los valores empíricos y mediante el despeje de variables de (2.3) se obtienen los valores empíricos de la frecuencia, el factor de amortiguamiento y la ganancia, los cuales son:

$$K = 2965 \quad \omega_n = 4{,}239 \quad \zeta = 0{,}04896 \tag{4.2}$$

Ello confirma que el sistema es subamortiguado, cuyo comportamiento refleja en la figura 4.1, donde se observa que como respuesta a un escalón el sistema oscila con un periodo $T = 2\pi/\omega_n = 1{,}48\,\text{s}$, y un tiempo de establecimiento al 2 % de $t_s = 4/\zeta\omega_n = 21\,\text{s}$. La hélice y fricciones tanto internas como externas amortiguan el comportamiento oscilatorio del péndulo, posicionándolo alrededor de 14° en el tiempo de establecimiento.

El modelo determinado posee un porcentaje de aproximación del 84,99 % respecto a los datos medidos con un valor de MSE de 1,973 grados al cuadrado. La predicción de la estimación dada por ident se muestra en la figura 4.6.

Debido a las aproximaciones usadas para derivar (2.3), el modelo matemático determinado no iguala el comportamiento real del PAHM en todo su rango de operación, lo que se muestra en la figura 4.7 donde se compara la respuesta dada por la planta física contra la

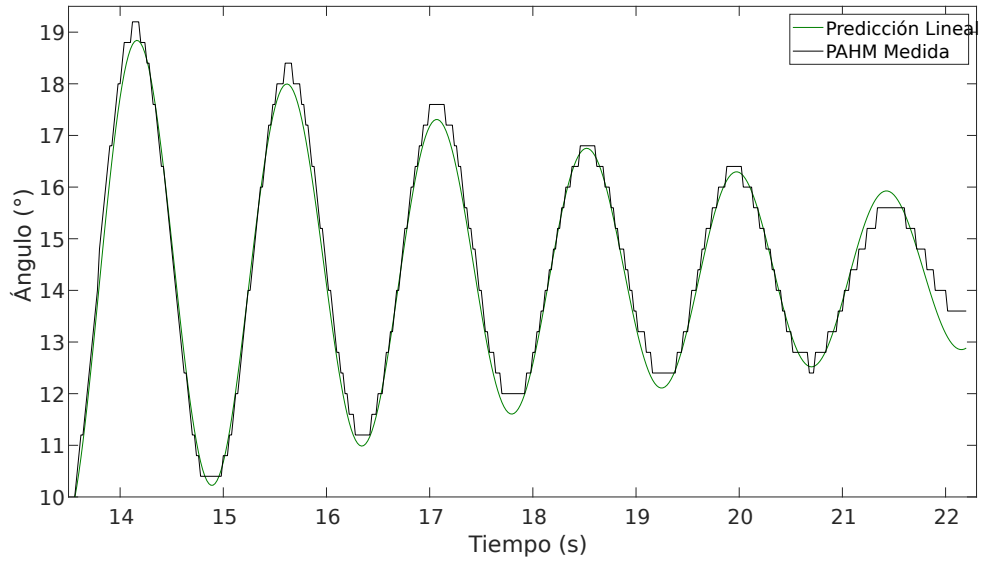


Figura 4.6: Predicción del modelo de ident y la respuesta medida en la planta física, para una entrada escalón.

respuesta del modelo ante una entrada manual. Se observa que el modelo sigue la tendencia del comportamiento dinámico de la planta PAHM; no obstante, posee un error absoluto promedio de 17,5177° con una diferencia máxima de 24,647°. Aún así, el comportamiento presentado por el modelo es esperado debido a que se estima bajo la aproximación de $\text{sen}(\theta) \approx \theta$ para ángulos pequeños, por lo que es esperado que la predicción se desvíe, donde el error incrementa un 1 % cada grado para ángulos mayores de 14°.

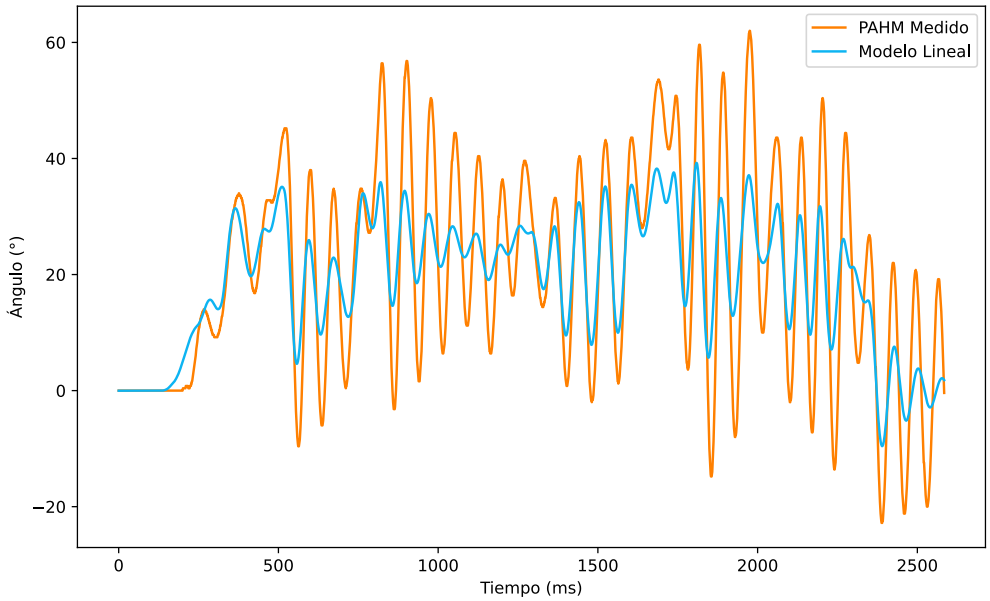


Figura 4.7: Comparación de la respuesta del modelo matemático con la PAHM real.

4.3. Análisis de la RNAM sintética

Para probar la RNAM se hace uso del modelo lineal ideal con el fin comprobar si la red puede simular su comportamiento. A este se le denominada *RNAM sintética*. La RNAM sintética presenta un menor tiempo de entrenamiento debido a que la cantidad de datos generados es menor que los datos recopilados de la PAHM física. Esto reduce el tiempo de entrenamiento, y permite entonces aumentar el número de pruebas para determinar valores óptimos de hiperparámetros.

Los siguientes experimentos trabajan con errores calculados sobre los ángulos normalizados según la sección 3.3.1, es decir, un valor de 1 equivale a 120°. Como la métrica MSE es un promedio sobre una transformación no lineal, los valores resultantes no pueden transformarse a un equivalente en grados exacto. Obsérvese que MSE, al utilizar cuadrados de valores menores a 1, se espera que produzca siempre valores menores a MAE para caracterizar una misma diferencia. La métrica MAE desnormalizado sí se puede interpretar como la diferencia promedio en grados.

El conjunto total de datos utilizado para la RNAM sintética es de 3250 muestras, el cual se separa en 2600 muestras para los datos de entrenamiento y 650 muestras para los datos de validación y prueba cada uno.

Se realiza una prueba base para observar si la red es capaz de asimilar la dinámica del modelo, para lo cual se configura un entrenamiento con 1500 épocas, un tamaño de lote de uno y 32 dimensiones en el espacio de estados. Se utiliza la métrica MAE como función de pérdida y se utiliza un intervalo de tiempo de dos segundos. El resultado obtenido se muestra en la figura 4.8, el cual a pesar de tener sus diferencias, logra aprender la tendencia general del comportamiento del modelo.

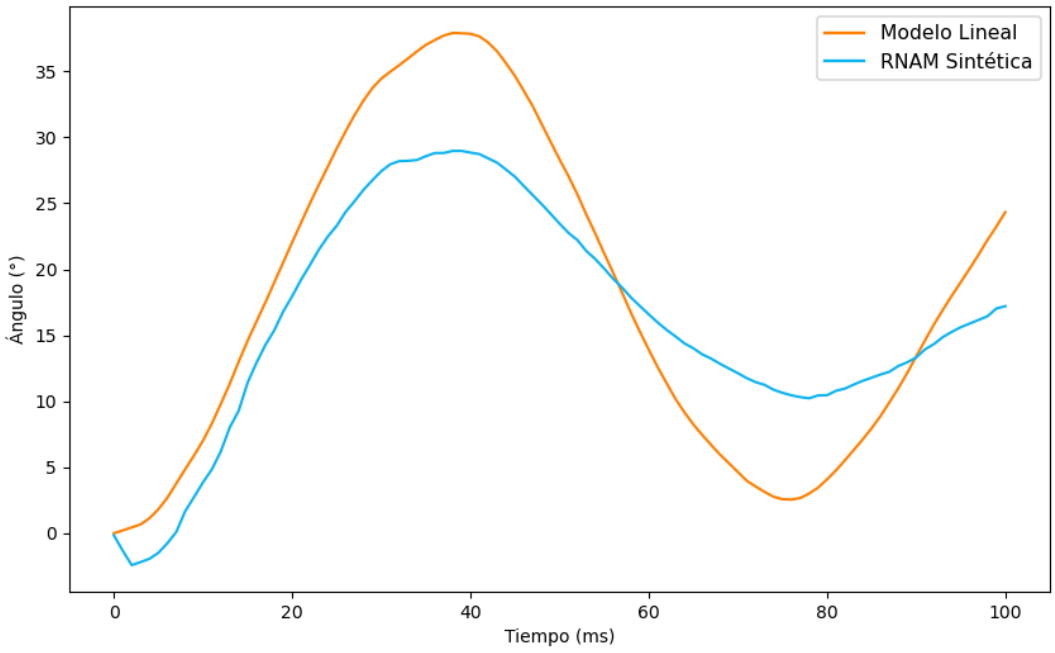


Figura 4.8: Predicción de la prueba base de la RNAM sintética.

Es necesaria la optimización de hiperparámetros para mejorar la respuesta obtenida. Para ello, se evalúan las métricas de la función de pérdida, entrenando con un tamaño de lote de 16, con 32 dimensiones en el espacio de estados y variando entre MSE y MAE. Para la métrica MAE, la pérdida obtenida es de 0,0105, mientras que para el MSE es de 0,0004349. La predicción obtenida se aproxima a la curva del PAHM, tal como se muestra en la figura 4.9. La métrica MSE se caracteriza por castigar errores de gran magnitud, y ese efecto se nota en la reducción del error con respecto a la RNAM entrenada con la métrica MAE.

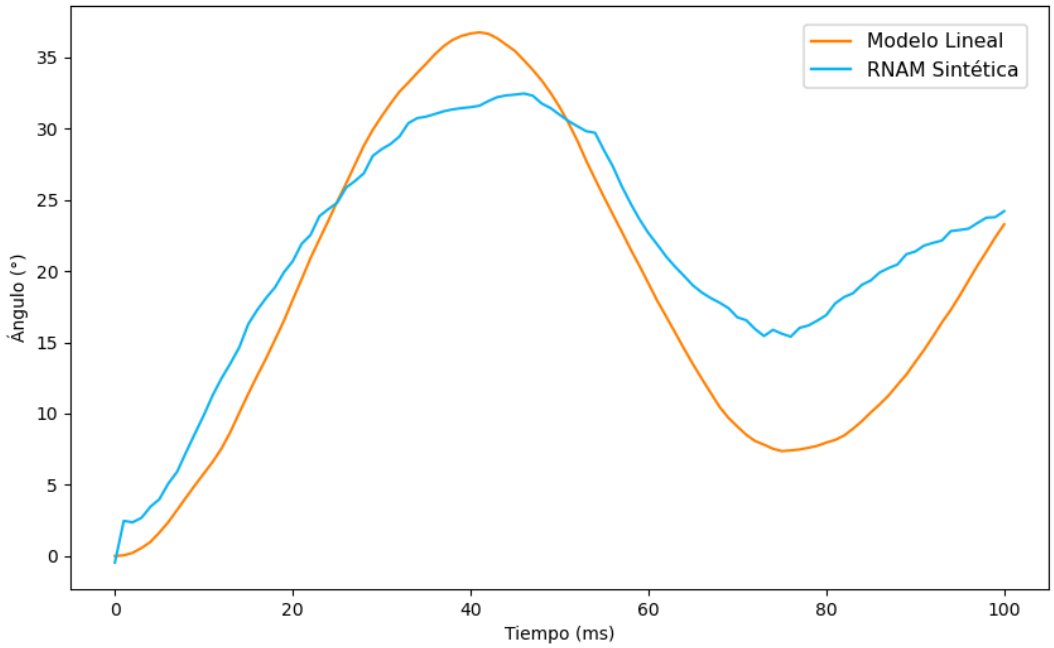
Los resultados obtenidos en la figura 4.9 ilustran la capacidad de la RNAM de simular el comportamiento dinámico teórico del PAHM. Con la métrica ya definida como MSE, se varían los hiperparámetros de la dimensión del espacio de estados y el tamaño de lote, cuyos resultados se resumen en la tabla 4.1. Se obtiene que el valor óptimo para la dimensión del espacio de estados y el tamaño de lote es de 32 y 16 respectivamente, donde la predicción obtenida por la RNAM, con la configuración óptima se muestra en la figura 4.10, con un intervalo de tiempo mayor a los doce segundos.

Tabla 4.1: Pérdida de entrenamiento y validación variando el tamaño de lote y la dimensión del espacio de estados. Los valores de pérdidas están en unidades de grados normalizados al cuadrado.

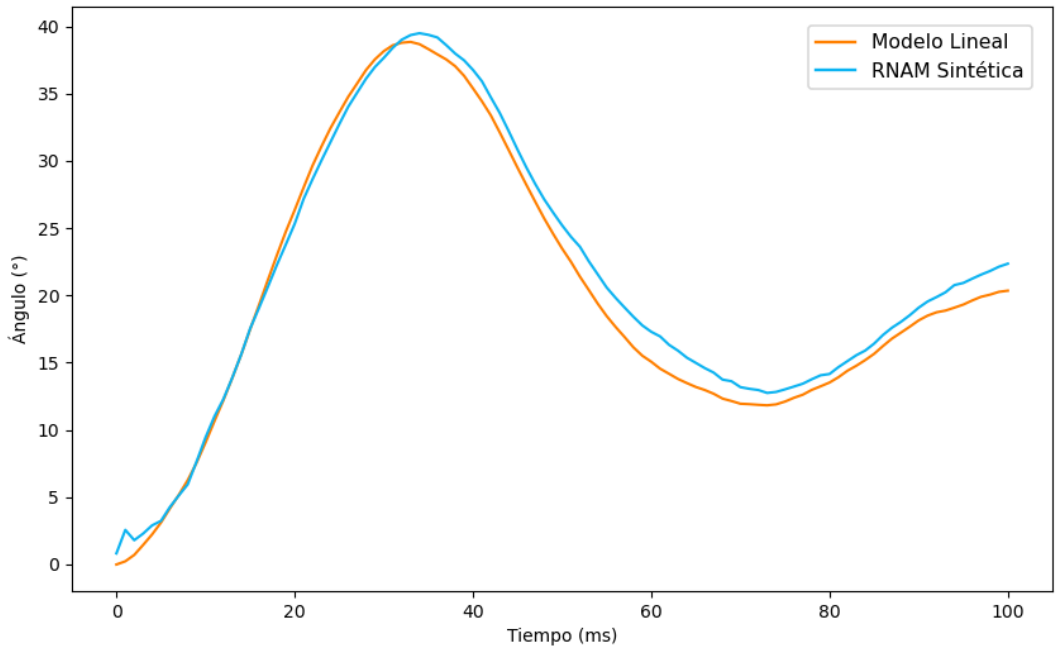
RNAM Sintética				
Configuración			**Resultados**	
Épocas	**Tamaño de lote**	**Dimensión de espacio de estados**	**Pérdida MSE**	**Pérdida de validación MSE**
1500	16	8	0,0006241	0,0009748
		16	0,0005660	0,001037
		32	**0,00008957**	**0,0001185**
		64	0,0002765	0,001004
		128	0,000257	0,0002714
	1	32	0,00834	0,02196
	16		**0,00008957**	**0,0001185**
	32		0,0001245	0,0001790
	96		0,0001241	0,0001464

La predicción dada en la figura 4.10 presenta un valor de MAE de 1,0408°. Aún así, con el fin de perfeccionar la respuesta de la RNAM sintética se realiza un preentrenamiento con la métrica MAE. Dicha acción disminuye el valor del MAE en las predicciones a 0,3663°. La predicción dada por la RNAM sintética preentrenada se muestra en la figura 4.11 y en la tabla 4.2 se resumen los valores de pérdida obtenidos con y sin preentrenamiento.

Ahora bien, con el fin de dar mayor capacidad de aprendizaje temporal, se añade una segunda capa GRU. Esta se configura con los valores óptimos determinados anteriormente y se realiza un preentrenamiento. En la figura 4.12 se muestran las predicciones resultantes y en la tabla 4.3 se resumen los valores de pérdida obtenidos. El MAE de la predicción sin preentrenamiento es de 1,0667°, mientras que la red con preentrenamiento presenta un error del 0,3223°.



(a) Predicción con métrica MAE.



(b) Predicción con métrica MSE.

Figura 4.9: Predicciones dadas por la RNAM con el uso de diferentes métricas en la función de pérdida.

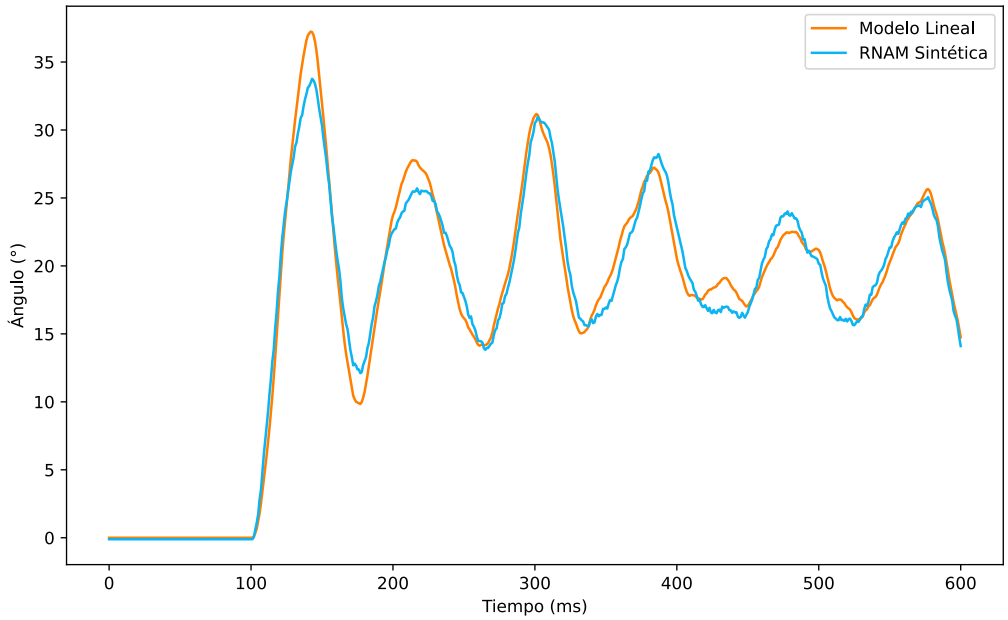


Figura 4.10: Predicción de la RNAM sintética entrenada con una dimensión de espacio de estados de la capa GRU de 32 y un tamaño de lote de 16.

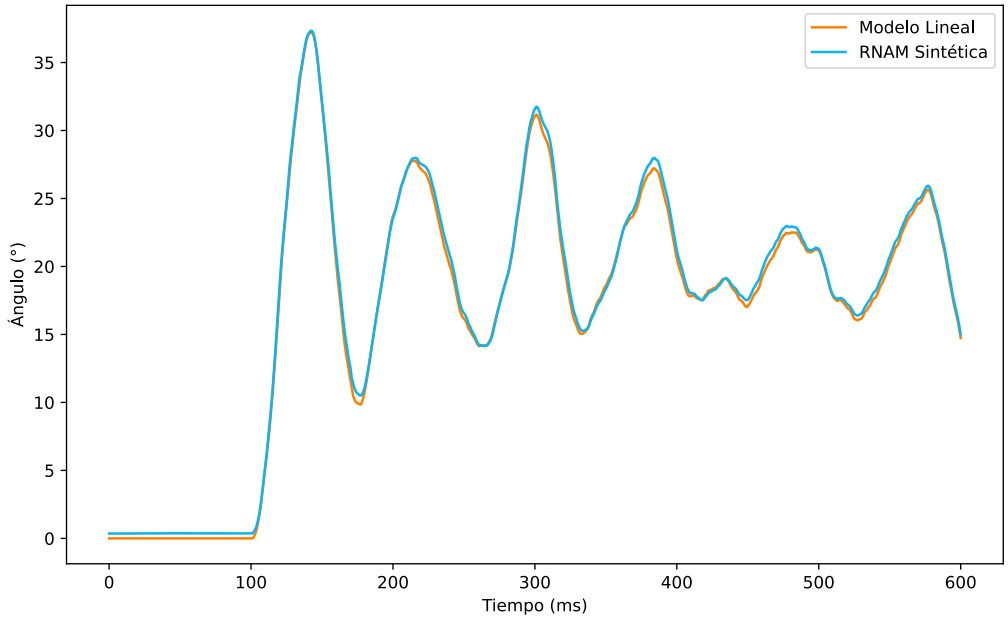
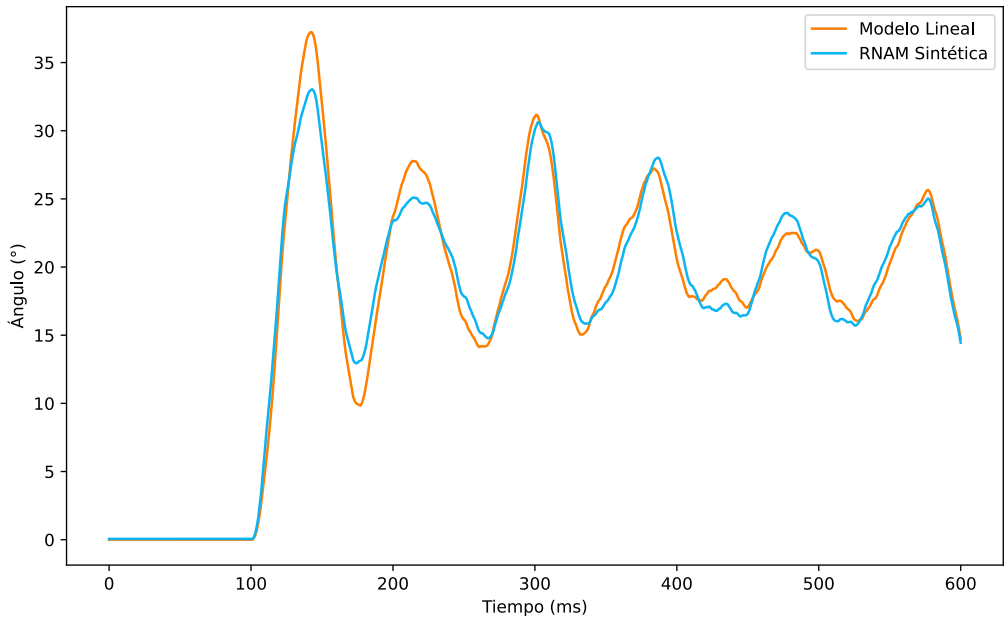
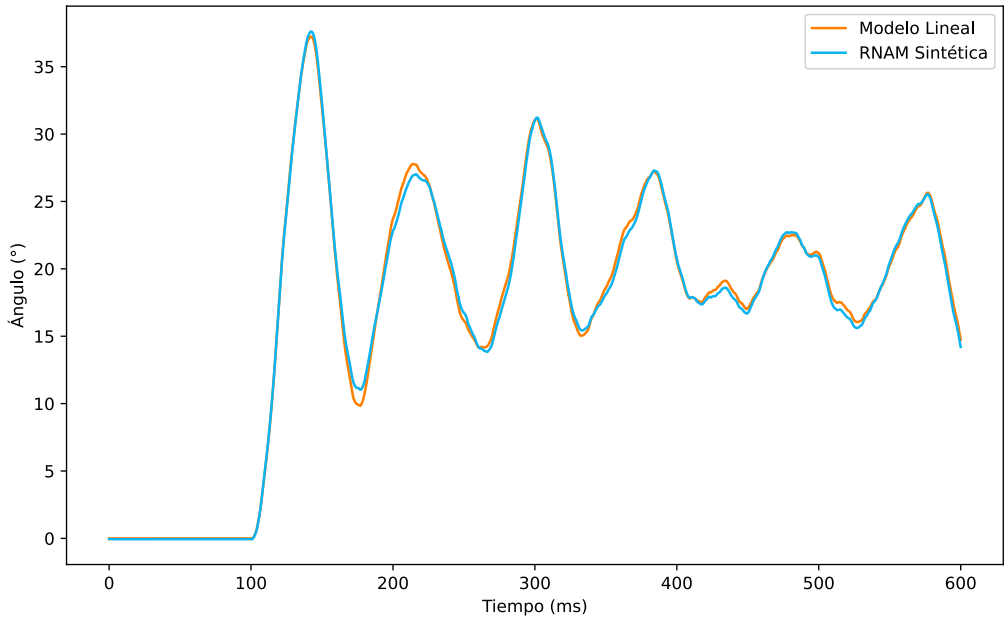


Figura 4.11: Predicción de la RNAM sintética con preentrenamiento.

Por último, se realiza un segundo preentrenamiento del modelo de una capa GRU con la métrica MSE en la función de pérdida, donde la predicción presenta un error absoluto del 0,0621°. Dicho modelo resulta preferible ya que presenta el menor MAE en sus



(a) Predicción con dos capas GRU sin preentrenamiento.



(b) Predicción con dos capas GRU con preentrenamiento.

Figura 4.12: Predicciones dadas por la RNAM con el uso de dos capas GRU.

Tabla 4.2: Pérdida de RNAM sintética evaluando preentrenamiento. Los valores para la métrica MSE están en unidades de grados normalizados al cuadrado, para la métrica MAE están en unidades de grados normalizado.

RNAM Sintética		
Métricas	**Sin preentrenamiento**	**Con preentrenamiento**
Pérdida MSE	0,0001146	-
Pérdida de validación MSE	0,0001799	-
Pérdida MAE	-	0,001176
Pérdida de validación MAE	-	0,001814

Tabla 4.3: Pérdida de RNAM sintética con dos capas GRU. Los valores para la métrica MSE están en unidades de grados normalizados al cuadrado, para la métrica MAE están en unidades de grados normalizado.

RNAM Sintética con 2 capas GRU		
Métricas	**Sin preentrenamiento**	**Con preentrenamiento**
Pérdida MSE	0,00005247	-
Pérdida de validación MSE	0,0001365	-
Pérdida MAE	-	0,001965
Pérdida de validación MAE	-	0,0027106

predicciones. La predicción del modelo se muestra en la figura 4.13.

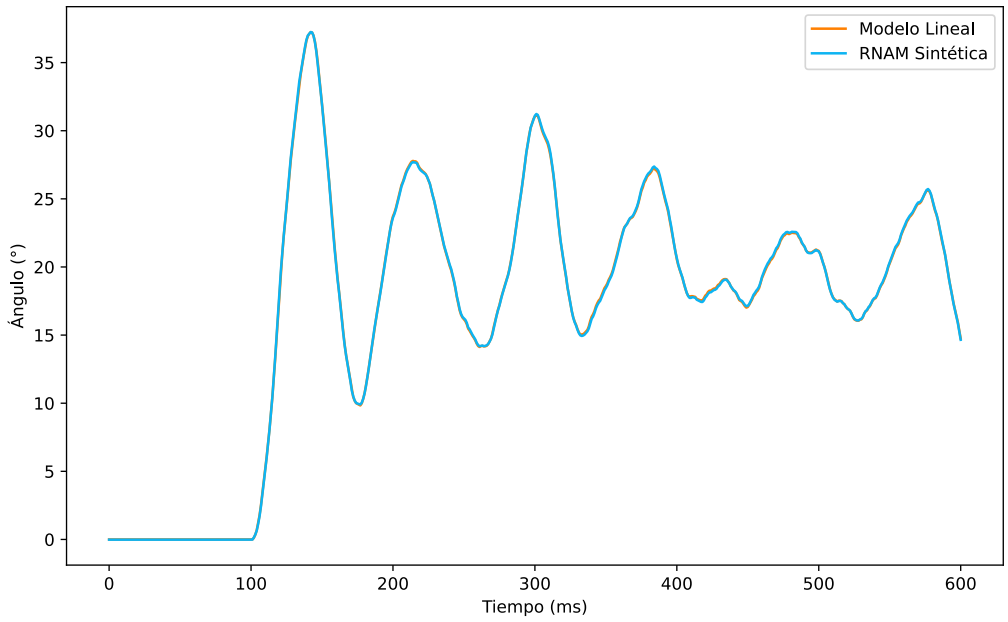


Figura 4.13: Predicción del modelo preentrenado por segunda vez.

Por lo tanto, el preentrenamiento mejora la predicción de la RNAM sintética tanto con una o dos capas GRU. El uso de otra capa GRU incrementa la cantidad de parámetros a calcular y por ende, el tiempo de entrenamiento. No obstante, la reducción en el error es de 0,044° por lo que se puede prescindir de su implementación. Además, con las predicciones obtenidas se determina que la RNAM sintética requiere de solo una capa GRU para aprender el comportamiento dinámico de dicho modelo matemático lineal, donde el preentrenamiento reduce más el MAE en comparación al uso de dos capas GRU.

4.4. Análisis de la RNAM real

Como se observa en la figura 4.7, el modelo lineal idealizado no comprende la totalidad del comportamiento dinámico del PAHM, para lo cual se necesita una ecuación más compleja que la presentada. Ahora bien, ya se analizó que la RNAM sintética es capaz de simular el comportamiento dinámico lineal del PAHM idealizado, por lo que ahora se analiza si la RNAM tiene la capacidad de simular el comportamiento dinámico del PAHM real. A esta red se le denomina *RNAM real*.

El conjunto de datos utilizado para la RNAM real contiene 279832 muestras, las cuales se separan en 167899 muestras para el entrenamiento de la red, 55966 para el conjunto de validación y 55967 para el conjunto de prueba. No obstante, se grafica un intervalo de tiempo de 50 s de ejecución con el fin de apreciar la respuesta de la RNAM real. Su entrada se muestra en la figura 4.14.

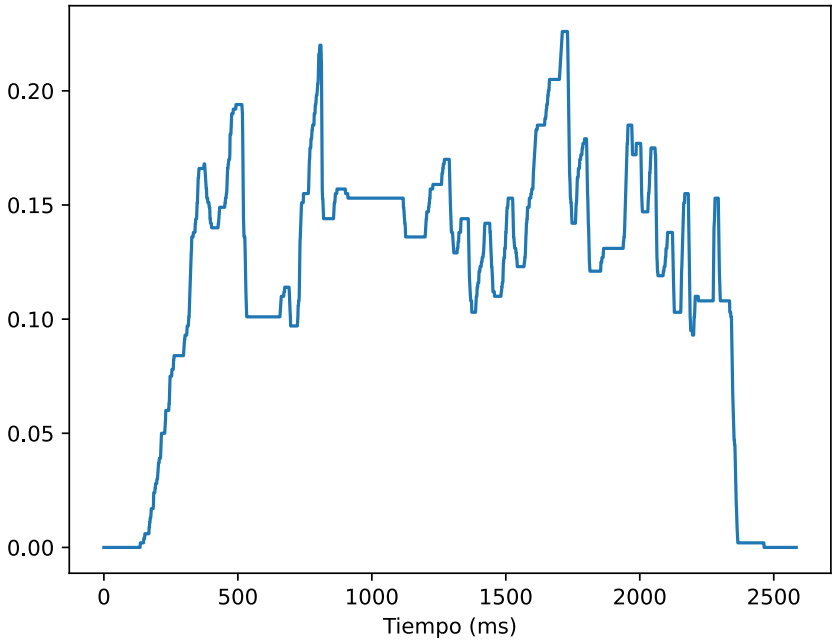


Figura 4.14: Señal de entrada para las predicciones de la RNAM real.

A pesar de determinar anteriormente la dimensión del espacio de estados de la GRU y el tamaño de lote, se repite la selección de hiperparámetros para el caso de datos provenientes del PAHM real. Los resultados obtenidos se resumen en la tabla 4.4. De esta forma, se confirma la configuración óptima con un tamaño de lote de 16 y un espacio de estados de dimensión de 32, observando los valores de pérdida tanto de entrenamiento como de validación. En la figura 4.15 se muestra la predicción de la red con dicha configuración, donde el MAE de la predicción es de 5,6776°.

Tabla 4.4: Pérdida de entrenamiento y validación variando el tamaño de lote y la dimensión del espacio de estados para la RNAM real. Los valores están en unidades de grados normalizados al cuadrado.

RNAM Real				
Configuración			**Resultados**	
Épocas	**Tamaño de lote**	**Dimensión de espacio de estados**	**Pérdida MSE**	**Pérdida de validación MSE**
1500	16	32	**0,0007967**	**0,0008837**
		64	0,002178	0,001737
	16	32	**0,0007967**	**0,0008837**
	32		0,0009579	0,0009934
	65		0,000997	0,001029
	128		0,0008253	0,001162
	64	128	0,001699	0,001267
	96	32	0,002034	0,001585

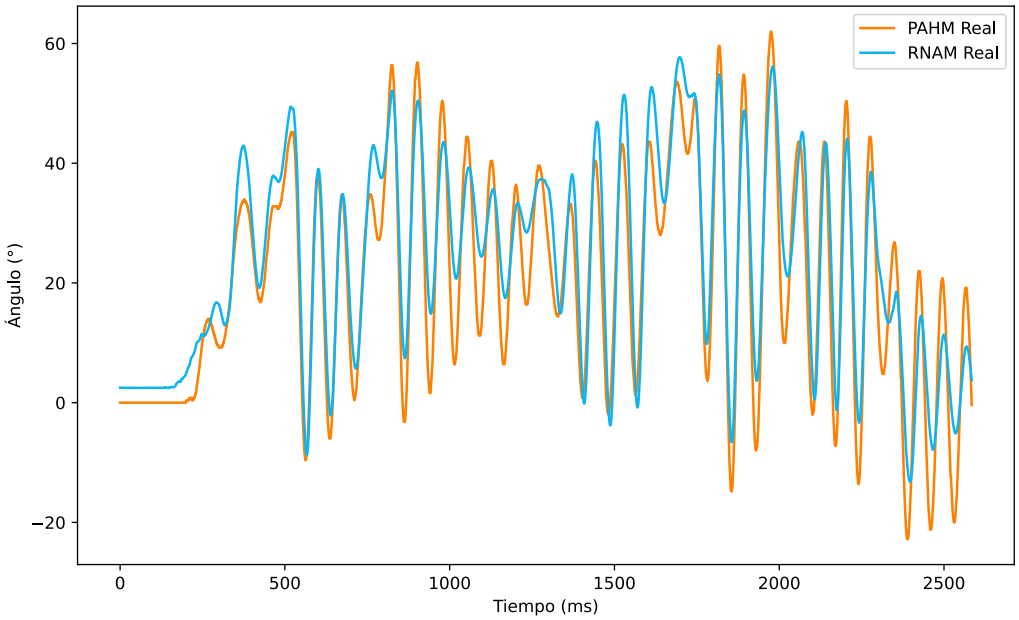


Figura 4.15: Predicción del modelo de la RNAM real entrenada con una dimensión de espacio de estados de la capa GRU de 32 y un tamaño de lote de 16.

Seguidamente, el modelo se preentrena con el fin de mejorar los resultados en la predicción del comportamiento de la PAHM. La tabla 4.5 resume los resultados en cuanto a las pérdidas obtenidas. La figura 4.16 muestra la predicción del modelo con preentrenamiento. Gracias a ello, el valor del MAE disminuye a 3,9045°.

Tabla 4.5: Pérdida de RNAM real evaluando preentrenamiento. Los valores para la métrica MSE están en unidades de grados normalizados al cuadrado. Para la métrica MAE las unidades son de grados normalizado.

RNAM Sintética con 2 capas GRU		
Métricas	**Sin preentrenamiento**	**Con preentrenamiento**
Pérdida MSE	0,0008198	-
Pérdida de validación MSE	0,0009934	-
Pérdida MAE	-	0,01363
Pérdida de validación MAE	-	0,01825

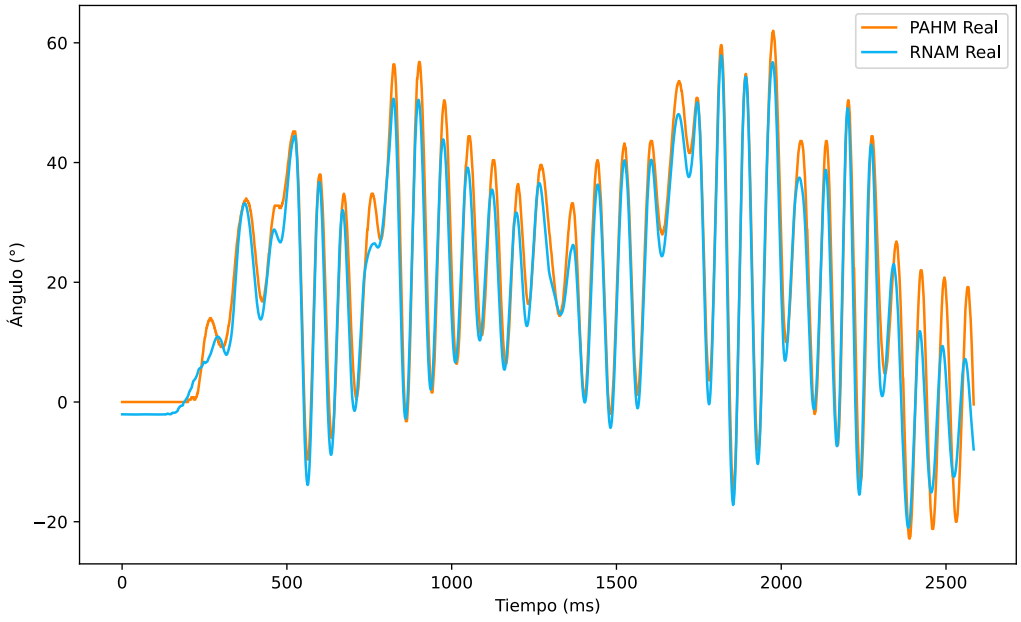
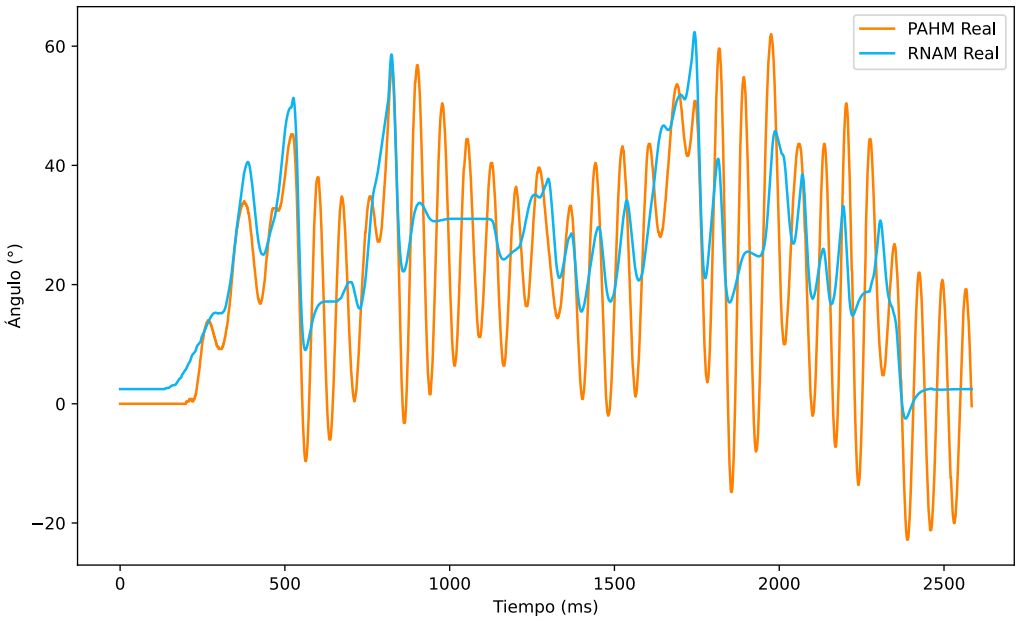


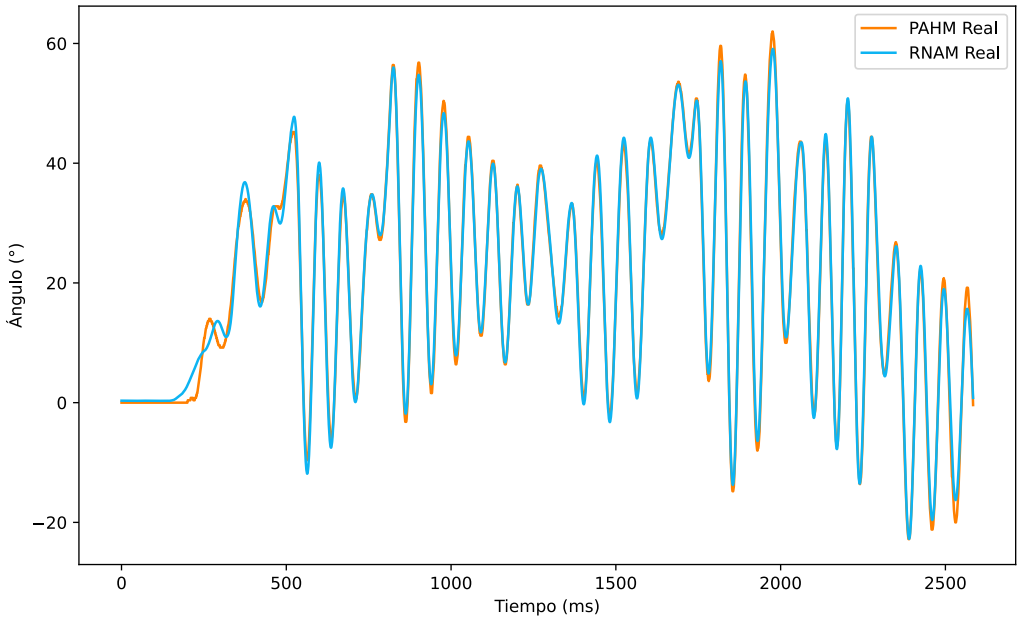
Figura 4.16: Predicción del modelo con preentrenamiento de la RNAM real.

También se añade otra capa GRU en el modelo neuronal. Los resultados se muestran en la tabla 4.6, donde se compara el modelo normal con el preentrenado. En la figura 4.17 se muestran las predicciones respectivas.

Se observa que, para el modelo sin preentrenamiento, la predicción se desvía de la respuesta real con un MAE de 10,3156°. Sin embargo, el modelo con preentrenamiento cumple su función simulando el comportamiento dinámico del PAHM, disminuyendo el valor del MAE a 1,0875°. A diferencia de la RNAM sintética, el uso de dos capas GRU con datos reales de la PAHM sí refleja cambios en cuanto a la predicción, donde para la RNAM



(a) Predicción con dos capas GRU sin preentrenamiento.



(b) Predicción con dos capas GRU y preentrenamiento.

Figura 4.17: Predicciones dadas por la RNAM con el uso de dos capas GRU.

Tabla 4.6: Pérdida de RNAM real con dos capas GRU evaluando preentrenamiento. Los valores para la métrica MSE están en unidades de grados normalizados al cuadrado. Para la métrica MAE las unidades son de grados normalizado.

RNAM Sintética con 2 capas GRU		
Métricas	**Sin preentrenamiento**	**Con preentrenamiento**
Pérdida MSE	0,001453	-
Pérdida de validación MSE	0,001342	-
Pérdida MAE	-	0,006929
Pérdida de validación MAE	-	0,01035

sintética mejora en MAE es de 0,0651°, mientras que en la RNAM real es de 2,817°. Comparando la figura 4.15 con la figura 4.17(b) se observa cómo el uso de dos capas GRU se ajusta más al comportamiento dinámico del PAHM real.

De igual forma que en la sección 4.3, se preentrena al mejor modelo optimizando al MSE, en este caso el modelo con dos capas GRU. A pesar de tener las mejores pérdidas de entrenamiento y validación de 0,0001316 y 0,0006078 respectivamente, el valor del MAE en la predicción es de 1,3295°. El resultado de esa predicción se muestra en la figura 4.18.

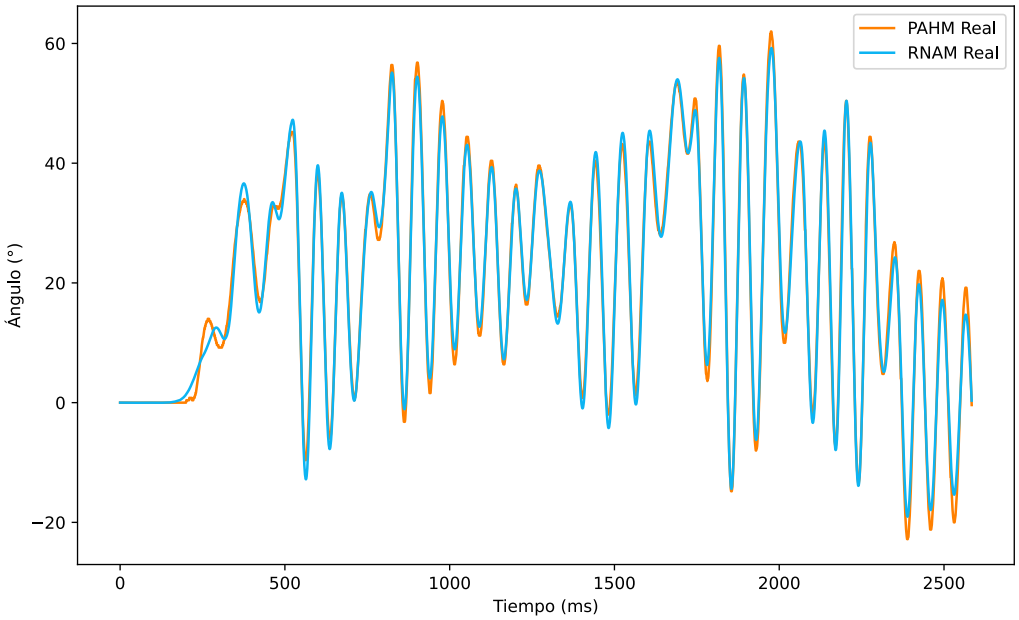


Figura 4.18: Predicción de la RNAM real preentrenada por segunda vez.

Considerando el tiempo de entrenamiento y que el resultado del MAE aumenta con el segundo preentrenamiento, se descarta. Por lo tanto, el modelo óptimo para simular el comportamiento dinámico de la planta PAHM física es la RNAM con dos capas GRU con preentrenamiento. A manera de resumen, en la tabla 4.7 se listan los errores obtenidos con cada modelo, donde el modelo base corresponde al modelo entrenado con un tamaño

de lote de 16, con 32 dimensiones en el espacio de estados y métrica MSE.

Tabla 4.7: Errores absolutos medios de las predicciones por los modelos de RNAM.

	RNAM Sintética	**RNAM Real**
Modelo base	1,0408°	5,6776°
Modelo base con preentrenamiento	0,3663°	3,9045°
Modelo con dos capas GRU	1,0667°	10,3156°
Modelo con preentrenamiento y con dos capas GRU	0,3223°	1,2416°
Modelo con segundo preentrenamiento	0,0621°	1,3295°

En cuanto a la pérdida de los modelos, se muestra en la figura 4.19 un ejemplo del proceso. Estos modelos siguen una tendencia a disminuir en las primeras 100 épocas, mientras el resto de épocas su valor está en constante variación y disminuye lentamente. Se observa una cercanía entre la curva de entrenamiento y la de validación, lo que indica que no hay sobreajuste en los modelos. Como ejemplo, se muestra la curva de pérdida de la RNAM real con dos capas GRU con preentrenamiento. Cabe aclarar que las demás pérdidas tienen comportamiento similar.

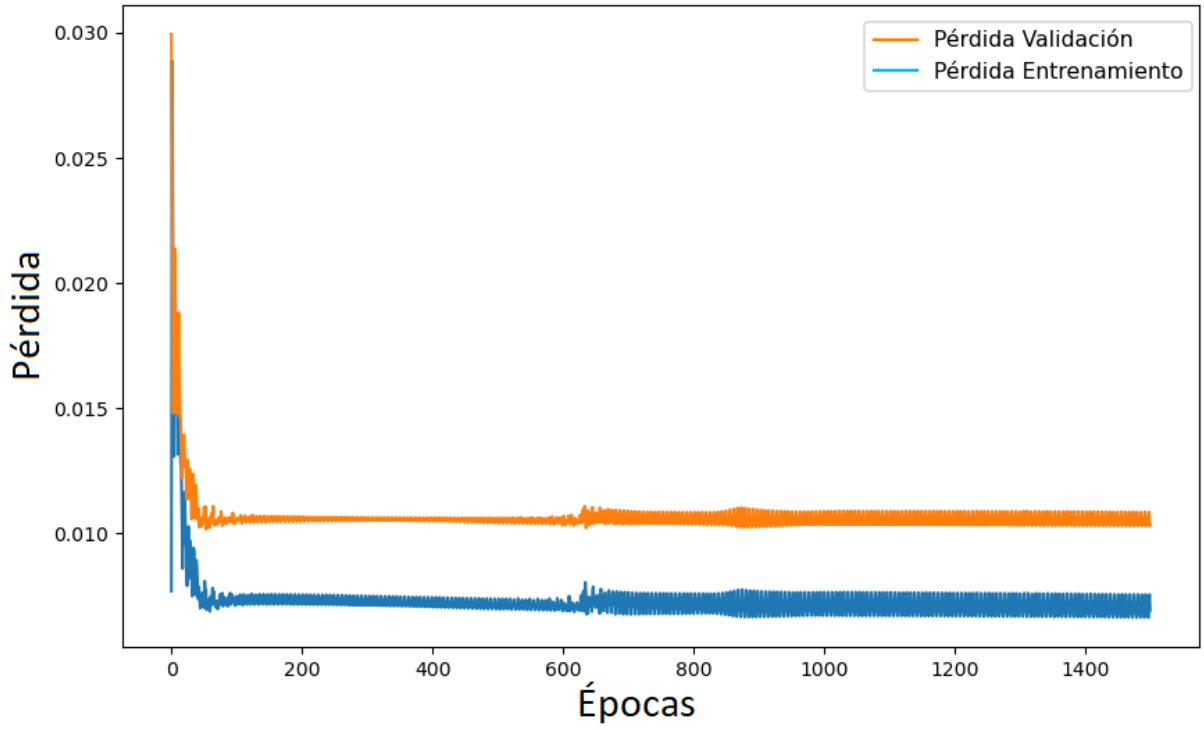


Figura 4.19: Ejemplificación de la pérdida MAE de los modelos.

Capítulo 5

Conclusiones y recomendaciones

5.1. Conclusiones

El presente trabajo propone el uso de redes neuronales artificiales para simular el comportamiento dinámico de un péndulo amortiguado a hélice (PAHM). Para ello, se acoplan dos sistemas embebidos: un PSoC 5 LP y un NVIDIA Jetson TX2. Se desarrolla una red neuronal artificial mimetizadora (RNAM) basada en el uso de capas GRU y una capa densa para regresión en su salida. Se evalúan dos escenarios que permiten cuantificar la capacidad de aprendizaje del comportamiento dinámico del PAHM de un modelo lineal idealizado, y de la planta real.

Para el comportamiento lineal idealizado, el mejor modelo encontrado presenta un MAE de 0,0621° en la predicción realizada. Para el comportamiento real, referido a la planta física con sus perturbaciones, no linealidades, varianza en el tiempo e incertidumbre del sensor de cuadratura, el mejor modelo encontrado presenta un MAE de 1,0875°.

De los resultados obtenidos se concluye que el preentrenamiento de la red es necesario, ya que reduce el MAE en las predicciones del modelo. En cuanto a la cantidad de capas GRU, para el escenario sintético es suficiente el modelado con una capa GRU, ya que el uso de dos capas GRU incrementa el tiempo de entrenamiento para una reducción del MAE de 0,044° en comparación al modelo con una capa GRU. Por otro lado, para el escenario real, el uso de dos capas GRU proporciona una reducción del error de 2,817° comparado con el modelo de una capa GRU.

El modelo lineal empírico permite aproximar el comportamiento dinámico del PAHM solo para ángulos pequeños. Se verifica que para ángulos mayores a 14° el error en las predicciones aumenta, lo que limita la aproximación del comportamiento del PAHM. Por otro lado, el uso de la RNAM permite una aproximación del comportamiento del PAHM real en todo su espacio de operación, lo cual es conveniente para el uso de este como un simulador o gemelo digital del PAHM. Además, a pesar del efecto de las no linealidades y variaciones en el tiempo del PAHM, logra aprender ese comportamiento dinámico y dar una respuesta aproximada.

En cuanto al diseño del sistema completo, este logra acoplar los dos sistemas embebidos, donde el sistema puede capturar los valores del sensor del ángulo y pasar la entrada al actuador con la frecuencia destinada de 50 Hz. También permite la simulación del comportamiento dinámico del PAHM. Una ventaja de la RNAM implementada es su independencia de la planta en particular, ya que solo con adquirir datos de plantas similares se puede reutilizar el sistema de aprendizaje automático sin mayor intervención.

5.2. Recomendaciones

Se recomienda pasar las funciones de seguridad de la Jetson TX2 al PSOC 5 LP. Esto por dos motivos: para dedicar el uso de la Jetson TX2 a la recolección de datos y eventualmente al sistema de control, y para asegurar la reacción en caso de una emergencia. En estos momentos el valor de ángulo debe ser procesado por el PSoC y transmitido a la Jetson TX2, se procesa el dato y responde con el valor PWM para detener la planta, el cual tiene que ser transmitido. No obstante, esto conlleva tiempo y riesgos potenciales que pueden evitarse al tener dichas funciones en el PSoC.

Se recomienda además, hacer una exploración más exhaustiva del espacio hiperparamétrico, para evaluar potenciales mejores modelos, aplicables a plantas más complejas que el PAHM.

5.3. Trabajos a futuro

Como trabajo a futuro del proyecto se encuentra el desarrollo del controlador basado en aprendizaje reforzado, para el cual se desarrolla la red neuronal artificial mimetizadora debido a factores de seguridad y velocidad de los entrenamientos. Además, se debe hacer la realimentación de la RNAM con el PAHM, de forma que esta pueda actualizar los pesos tras observar perturbaciones que no son observables por el entorno simulado.

Bibliografía

[1] Á. Almidón Elescano y E. Julian-Laime, *Sistemas de Control Automático I- Teoría y problemas aplicativos*. 2018.

[2] M. Perez, A. Perez Hidalgo y E. Perez Berenguer, *Introducción a los sistemas de control y modelo matemático para sistemas lineales invariantes en el tiempo*, Argentina, 2008.

[3] K. Ogata, *Ingeniería de control moderna*, 5.ª ed. Madrid: Prentice Hall, 2010.

[4] R. Dorf y R. Bishop, *Modern control systems*, 12.ª ed. Madrid: Prentice Hall, 2011.

[5] J. González Godoy, «Modelado de sistemas dinámicos usando redes neuronales recurrentes profundas,» Tesis doct., Centro de investigación y estudios avanzados del instituto politécnico nacional, México, 2019.

[6] C. Møldrup Legaard, T. Schranz, G. Schweiger y col., «Constructing neural network-based models for simulating dynamical systems,» *ACM Computing Surveys*, vol. 1, n.º 1, 2021.

[7] M. M. Serón y J. H. Braslavsky, «Sistemas no lineales,» *Departamento de Electrónica, Universidad Nacional de Rosario, Primer Cuatrimestre*, 2000.

[8] J. Gómez Martínez y G. Mendoza Avedaño, *Aplicación del control adaptativo a procesos industriales tipo SISO*, Bucaramanga, Colombia, 2009.

[9] E. Sarmiento Jurado, «Diseño e implementación de un controlador basado en redes neuronales con entrenamiento rápido para sistemas de control 2x2,» Tesis de mtría., Universidad del Norte, Barranquilla, Colombia, 2006.

[10] R. Valverde Gil y D. Gachet Páez, «Identificación de sistemas dinámicos utilizando redes neuronales,» *Revista Iberoamericana de Automática e Informática Industrial*, n.º 1697-7912, 2007.

[11] B. Morcego Seix, «Estudio de redes neuronales modulares para el modelado de sistemas dinámicos no lineales,» Tesis doct., Universitat Politècnica de Catalunya, Barcelona, España, 2000.

[12] J. Acosta, J. Fernández y L. Becerra, *Aplicación de las redes neuronales para la identificación de un sistema no lineal. Un caso práctico*, 2000.

[13] S. Torrubia Caravaca, «Redes neuronales multimodelo aplicadas al control de sistema,» Tesis de mtría., Universidad Autónoma de Barcelona, España, 2010.

[14] J. F. Hernández Pérez, «Redes neuronales dinámicas para la identificación y control adaptable para sistemas no lineales,» Tesis de mtría., Universidad Autónoma del Estado de Hidalgo, Pachuca, México, 2018.

[15] D. Silver, A. Huang, C. Maddison y col., «Mastering the game of Go with deep neural networks and tree search,» *nature*, vol. 529, n.º 7587, págs. 484-489, 2016.

[16] J. Jumper, R. Evans, A. Pritzel y col., «Highly accurate protein structure prediction with AlphaFold,» *Nature*, vol. 596, n.º 7873, págs. 583-589, 2021.

[17] L. Busoniu, T. de Bruin, D. Tolic, J. Kober e I. Palunko, «Reinforcement learning for control: performance, stability, and deep approximators,» *Annual Reviews in Control*, vol. 46, 2018, ISSN: 1367-5788.

[18] A. Díaz Latorre, «Aprendizaje por refuerzo para control de sistemas dinámicos,» Tesis de mtría., Universidad Autónoma de Occidente, Santiago de Cali, Colombia, 2019.

[19] J. Torres, en *Introducción al aprendizaje por refuerzo profundo. Teoría y práctica en Python*. Publicación Independiente, 2021, Capítulo 1.

[20] R. Sutton y A. Barto, *Reinforcement Learning: An Introduction*, 2.ª ed. Cambridge, MA: The MIT Press, 2018.

[21] V. Mnih, K. Kavukcuoglu, D. Silver y col., «Playing atari with deep reinforcement learning,» *arXiv preprint arXiv:1312.5602*, 2013.

[22] I.-A. Hosu y T. Rebedea, «Playing atari games with deep reinforcement learning and human checkpoint replay,» *arXiv preprint arXiv:1607.05077*, 2016.

[23] F. Moreno, J. Becerra, Y. Ayala y R. Alarcón, «Desarrollo de un módulo didáctico para control angular de un péndulo suspendido,» Cúcuta, Colombia, inf. téc., 2012.

[24] Y. Taskin, «Fuzzy PID controller for propeller pendulum,» *IU-Journal of Electrical & Electronics Engineering*, vol. 17, n.º 1, págs. 3175-3180, 2017.

[25] V. Gómez, *Práctica 2: Péndulo Simple. Determinación de la aceleración de la gravedad*, Valladolid, España, 2017.

[26] A. Ruiz y E. Interiano, *Proyecto corto: Identificación y control del péndulo amortiguado a hélice (PAMH)*, Cartago, Costa Rica, 2022.

[27] M. Ortiz Moctezuma, *Sistemas dinámicos en tiempo continuo: Modelado y simulación*. México: OmniaScience, 2015.

[28] F. Mazo Espiau, «Modelado Dinámico y Simulación del robot industrial Stäubli TX90,» 2011.

[29] F. E. M. García, J. Becerra e Y. E. A. García, «Desarrollo de un módulo didáctico para control angular de un péndulo suspendido,» *Respuestas*, vol. 17, n.º 2, págs. 48-54, 2012.

[30] A. Castaño Hernández, J. Moreno Beltrán, J. Hernández Pérez y R. Villafuerte Segura, «Diseño y control de un sistema balancín con motor y hélice de bajo costo,» *Pädi Boletín Científico de Ciencias Básicas e Ingenierías del ICBI*, vol. 5, n.º 10, 2018.

[31] H. Young y R. Freedman, *Física Universitaria*, 13.ª ed. Mexico: Pearson, 2013, vol. 1.

[32] Á. Varela Fernández, *Modelado y control en el espacio de estados*, U. P. de Valencia, ed. España, 2016.

[33] C. Semiconductor, *PSoC® 5LP prototyping kit guide*, San José, California, 2016.

[34] ——, *PSoC® 5LP: CY8C58LP family datasheet*, San José, California, 2016.

[35] O. Alarcón Balseca, «Controlador electrónico de velocidad para cuatro motores sin escobillas de un dron,» B.S tesis, 2019.

[36] E. Ávila, V. Cuásquer, L. Ortega y M. Pozo, «Diseño e implementación del control electrónico de velocidad con freno regenerativo para una plataforma autónoma móvil terrestre con tracción diferencial,» 2016.

[37] H. Abdelrahman, *Software integration of electronic speed controller (ESC) for an unmanned aerial robot*, 2021.

[38] NVIDIA, *NVIDIA Jetson TX1/TX2 Developer Kit Carrier Board*, Santa clara, California, 2017.

[39] ——, *Jetson TX2 Developer Kit*, Santa clara, California, 2020.

[40] ——, *Datasheet NVIDIA Jetson TX2 NX System-on-Module*, Santa clara, California.

[41] B. Díaz Mulas, «UART: Universal Asynchronous Receiver-Transmitter,» B.S. tesis, 2015.

[42] C. Álvarez Hernández, «Implementación de un ambiente de verificación UART mediante método UVM,» Lic. Tesis, Tecnológico de Costa Rica, Costa Rica, 2018.

[43] B. Yegnanarayana, *Artificial neural networks*. PHI Learning Pvt. Ltd., 2009.

[44] X. B. Olabe, «Redes neuronales artificiales y sus aplicaciones,» *Publicaciones de la Escuela de Ingenieros*, 1998.

[45] C. Arana, «Redes neuronales recurrentes: Análisis de los modelos especializados en datos secuenciales,» 2021.

[46] F. Izaurieta y C. Saavedra, «Redes neuronales artificiales,» *Departamento de Física, Universidad de Concepción Chile*, 2000.

[47] O. I. Abiodun, A. Jantan, A. E. Omolara, K. V. Dada, N. A. Mohamed y H. Arshad, «State of the art in artificial neural network applications: A survey,» *Heliyon*, vol. 4, n.º 11, 2018.

[48] R. Velázquez y G. Acuña, «Entrenamiento de redes neuronales recurrentes para sistemas dinámicos tipo NARMAX y NOE,» 2007.

[49] S. Haykin, *Neural Networks and Learning Machines*, 3.ª ed. Madrid: Prentince Hall, 2009.

[50] K. Patan, *Artificial Neural Networks for the Modelling and Fault Diagnosis of Technical Processes*. Berlin, Alemania: Springer, 2008.

[51] E. B. Solovyeva, «Types of recurrent neural networks for non-linear dynamic system modelling,» en *2017 XX IEEE International Conference on Soft Computing and Measurements (SCM)*, 2017, págs. 252-255.

[52] J. A. Pérez Ortíz, «Modelos predictivos basados en redes recurrentes neuronales de tiempo discreto,» Tesis doct., Universidad de Alicante, España, 2002.

[53] S. Hochreiter y J. Schmidhuber, «Long short-term memory,» *Neural computation*, vol. 9, n.º 8, págs. 1735-1780, 1997.

[54] J. Chung, C. Gulcehre, K. Cho e Y. Bengio, «Empirical evaluation of gated recurrent neural networks on sequence modeling,» *arXiv preprint arXiv:1412.3555*, 2014.

[55] K. Cho, B. Van Merriënboer, C. Gulcehre y col., «Learning phrase representations using RNN encoder-decoder for statistical machine translation,» *arXiv preprint arXiv:1406.1078*, 2014.

[56] J. J. Cea Morán, «Redes neuronales recurrentes para la generación automática de música,» Tesis de mtría., Universidad Politécnica de Madrid, España, 2020.

[57] G.-B. Zhou, J. Wu, C.-L. Zhang y Z.-H. Zhou, «Minimal gated unit for recurrent neural networks,» *International Journal of Automation and Computing*, vol. 13, n.º 3, págs. 226-234, 2016.

[58] C. J. Willmott y K. Matsuura, «Advantages of the mean absolute error (MAE) over the root mean square error (RMSE) in assessing average model performance,» *Climate research*, vol. 30, n.º 1, págs. 79-82, 2005.

[59] S. Cass, «The top programming languages: Our latest rankings put Python on top-again,» *IEEE Spectrum*, vol. 57, n.º 8, págs. 22-22, 2020.

[60] J. Brenes, *Trabajo_Final_Graduacion*, 2022. dirección: https://github.com/jorgebre98/Trabajo_Final_Graduacion.

Apéndice A

Obtención modelo empírico en Matlab

La planta sintética consiste en realizar un modelo matemático lineal, cuya ecuación va a tener la misma forma que (2.3). Para ello, se procede a identificar y estimar el modelo con la herramienta *ident* de Matlab, que permite derivar modelos de función de transferencia. Primeramente, se importan los datos a Matlab y se crea un objeto con "iddata" llamado PAHM, donde se especifica su entrada, salida y el tiempo de muestreo. En el espacio de trabajo de Matlab se ejecuta la orden de ident, donde se abre una ventana, se selecciona *Import data/Data object* para importar los datos a la herramienta, tal como se observa en la Fig. A.3 de color rojo. Seguido de ello, sale otra ventana la cual se llena con los datos del objeto creado, como se observa en la figura A.1.

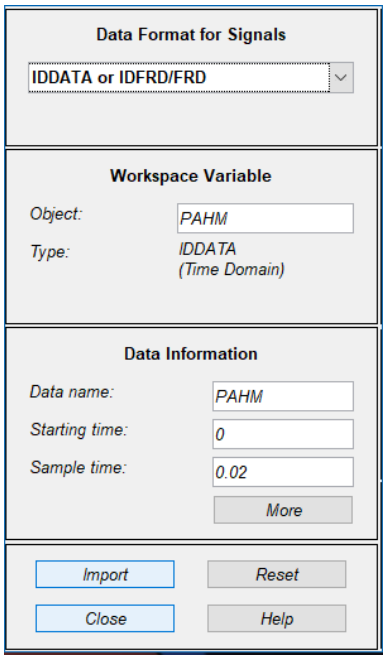


Figura A.1: Ventana para importar datos al ident.

Tras importar el objeto, se procede a dividir los datos en dos partes, seleccionando "← Preprocess/select range" como se muestra en la figura A.3 de color amarillo. De igual forma, en la figura 3.9 se observa la partición de datos realizada para el modelo.

Una vez realizado esto, se estima el modelo en "Estimate →/Transfer Functions Models", el cual da una función de transferencia. Se busca una función de transferencia que posea dos polos conjugados y una ganancia, por lo que se le indica, como se puede observar en la figura A.2. La función obtenida se puede exportar al espacio de trabajo de Matlab y de esta forma visualizarla, al igual que se puede hacer la conversión a espacio de estados, obteniendo los valores de las matrices **A**, **B**, **C** y **D**.

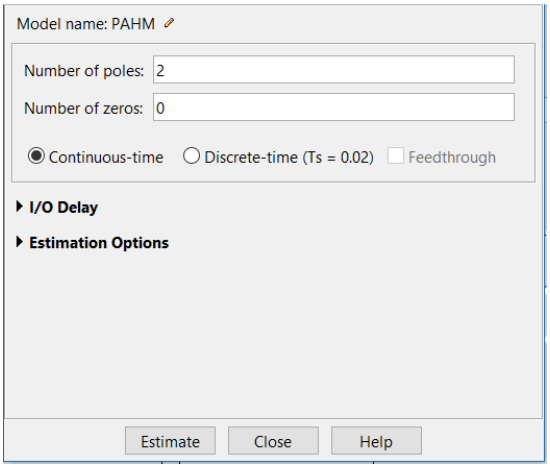


Figura A.2: Ventana para la obtención de la función de transferencia del ident.

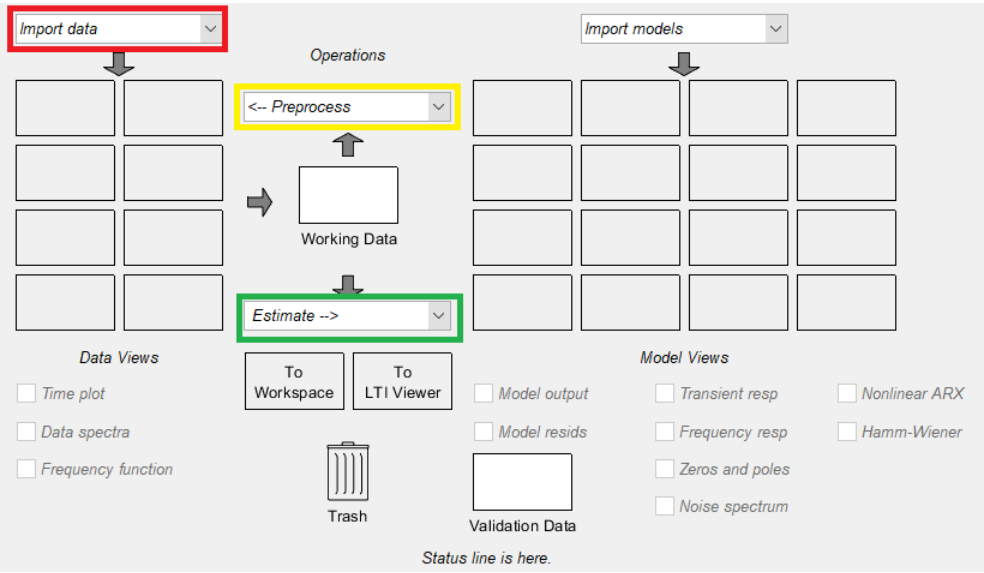


Figura A.3: Ventana correspondiente a la herramienta ident. De color rojo se encuentra la opción para importar los datos, en amarillo, para seleccionar el rango de datos de estimación y de validación y por último, en verde se muestra la opción para estimar el modelo mediante una función de transferencia.

Apéndice B

Espacio de estados de la PAHM

Tomando como punto de partida a (2.3), esta ecuación se manipula matemáticamente quedando como:

$$\Theta(s)(s^2 + 2\zeta\omega_n s + \omega_n^2) = KV(s) \tag{B.1}$$

donde se aplica la propiedad distributiva de la multiplicación y la transformada inversa Laplace, obteniendo:

$$\ddot{\theta}(t) + 2\zeta\omega_n\dot{\theta}(t) + \omega_n^2\theta(t) = KV(t) \tag{B.2}$$

Para representarla en variables de estado, use deben definir las siguientes variables de estado:

$$x_1(t) = \theta(t) \tag{B.3}$$

$$x_2(t) = \dot{x}_1(t) = \dot{\theta}(t) \tag{B.4}$$

$$x_3(t) = \dot{x}_2(t) = \ddot{\theta}(t) \tag{B.5}$$

De igual forma, se definen las ecuaciones diferenciales de primer orden:

$$\dot{x}_1(t) = x_2(t) \tag{B.6}$$

$$\dot{x}_2(t) = -2\zeta\omega_n x_2(t) - \omega_n^2 x_1(t) + KV(t) \tag{B.7}$$

Teniendo las variables de estado y las ecuaciones diferenciales definidas se obtiene el espacio de estados correspondiente a la PAHM:

$$\begin{bmatrix} \dot{x}_1(t) \\ \dot{x}_2(t) \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ -\omega_n^2 & -2\zeta\omega_n \end{bmatrix} \begin{bmatrix} x_1(t) \\ x_2(t) \end{bmatrix} + \begin{bmatrix} 0 \\ K \end{bmatrix} V(t) \tag{B.8}$$

Su salida corresponde a:

$$y(t) = \begin{bmatrix} 1 & 0 \end{bmatrix} \begin{bmatrix} x_1(t) \\ x_2(t) \end{bmatrix} \tag{B.9}$$

Apéndice C

Resultados adicionales de la RNAM

En este capítulo se muestran los resultados obtenidos con otros valores de tamaño de lote y dimensión del espacio de estados de la GRU que ayudaron a determinar los valores óptimos.

C.1. RNAM sintética

Primeramente, se expone la RNAM sintética donde se varía la dimensión del espacio de estados de la GRU y se utiliza un tamaño de lote de 16. Para ello, se presentan las configuraciones con dimensión del espacio de estados de 8 y 64. La RNAM con una dimensión del espacio de estados de 8 presenta un MAE de 3,2411° en sus predicciones, cuya predicción se muestra en la figura C.1. La RNAM con dimensión del espacio de estados de 64 presenta un MAE de 3,3110° y su predicción se muestra en la figura C.2.

No obstante, en la figura C.1 y C.2 se observa que la predicción no se ajusta al comportamiento dado por el modelo matemático de la PAHM. Esto implica que se requiere de una mayor cantidad de épocas para poder ajustar al modelo o un mayor tamaño del espacio de estados de la GRU o tamaño de lote. Ahora bien, se mantiene la dimensión del estado en 32 y se varía el valor del tamaño de lote, donde se presentan las configuraciones con valores de 32 y 96. La configuración con tamaño de lote de 32 presenta un MAE 3,2766° y el tamaño de lote de 96 presenta un MAE de 2,4161°. Se observa que mejora la predicción, aún más en el caso de utilizar un tamaño de lote de 96.

C.2. RNAM real

En cuanto a la RNAM real se presentan dos configuraciones. Una con dimensión del espacio de estados de 32 y un tamaño de lote de 96, la cual presenta un MAE de 10,5003°. Dicha predicción se muestra en la figura C.5.

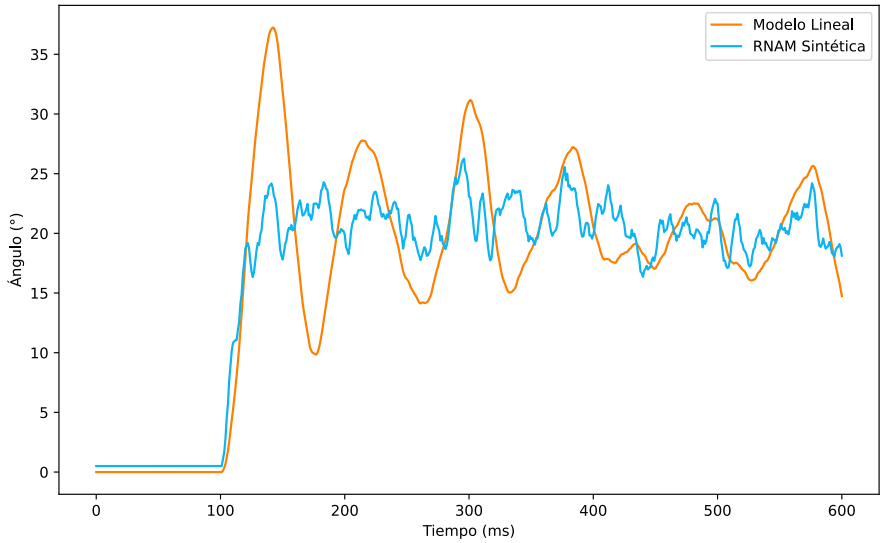


Figura C.1: Predicción de la RNAM sintética entrenada con dimensión del espacio de estados de 8 y tamaño de lote de 16.

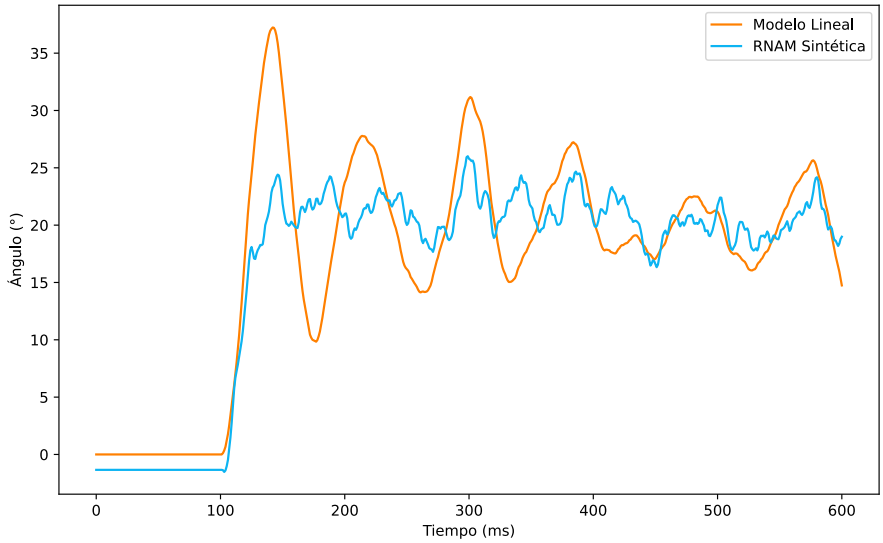


Figura C.2: Predicción de la RNAM sintética entrenada con dimensión del espacio de estados de 64 y un tamaño de lote de 16.

La otra configuración posee un tamaño de lote de 16 y una dimensión del espacio de estados de 64. Este presenta un error de 10,69°. La predicción obtenida se muestra en la figura C.6.

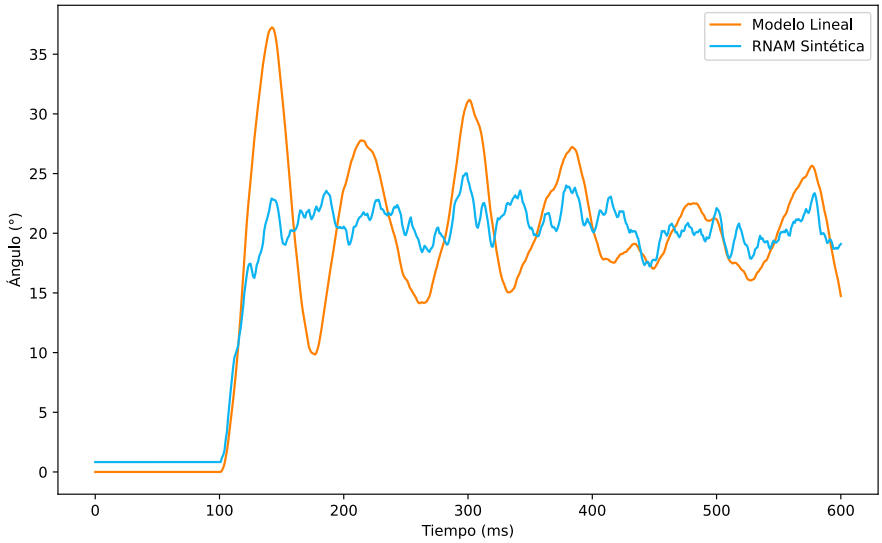


Figura C.3: Predicción de la RNAM sintética entrenada con dimensión del espacio de estado de 32 y un tamaño de lote de 32.

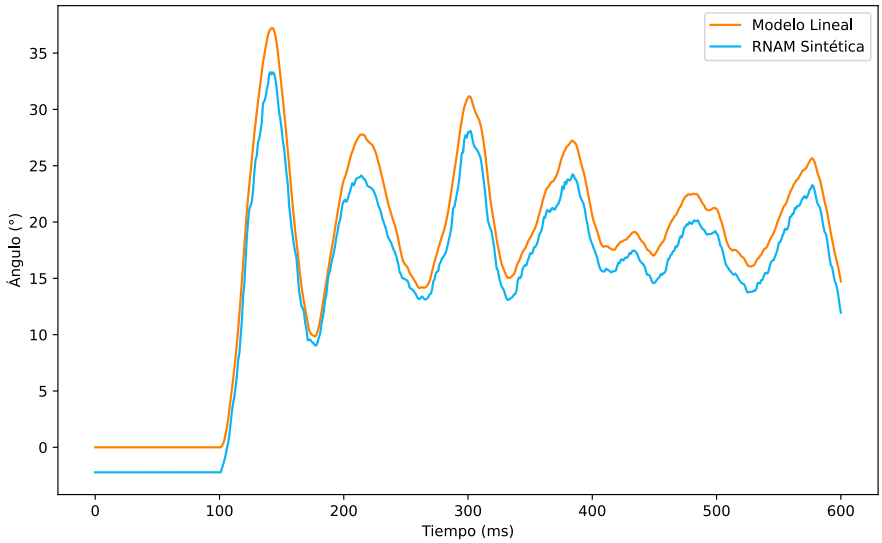


Figura C.4: Predicción de la RNAM sintética entrenada con dimensión del espacio de estado de 32 y un tamaño de lote de 96.

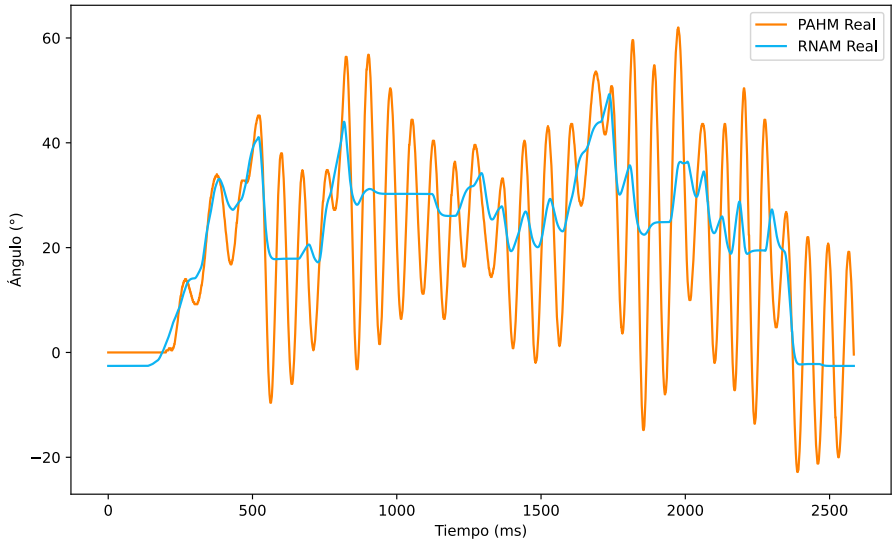


Figura C.5: Predicción de la RNAM real entrenada con dimensión del espacio de estado de 32 y un tamaño de lote de 96.

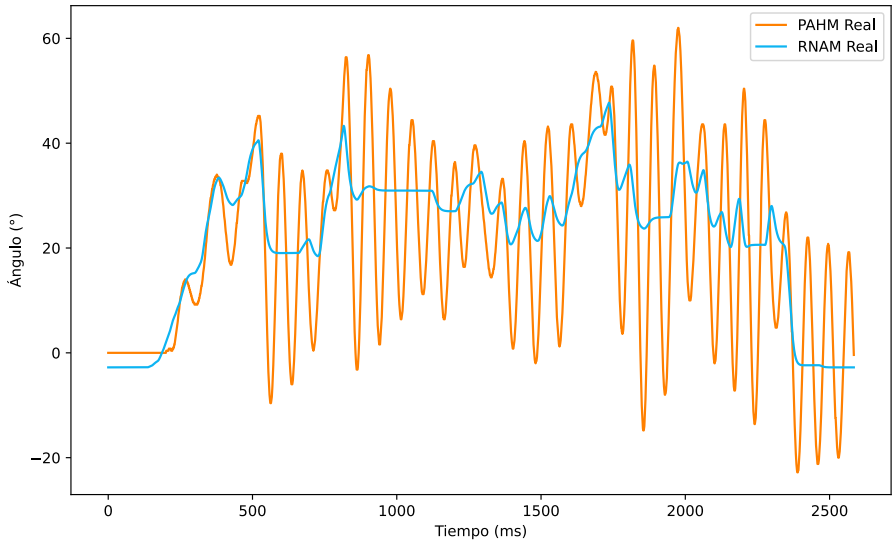


Figura C.6: Predicción de la RNAM real entrenada con dimensión del espacio de estado de 64 y un tamaño de lote de 16.