



UNIVERSIDAD TECNOLÓGICA (UTEC)
UNIVERSIDADE FEDERAL DO RIO GRANDE (FURG)
UNIVERSIDAD NACIONAL DE RAFAELA (UNRAF)
POSGRADO EN ROBÓTICA E INTELIGENCIA ARTIFICIAL (PRIA)

Proyecto Final

Desarrollo de un Sistema embebido diseñado para la predicción de humedad en esponja fenólica de germinación hidropónica basado en RNN/LTSM

Gustavo Pablo Castro Abdallah

Proyecto final presentado a el Posgrado en Robótica e Inteligencia Artificial (PRIA) como requisito parcial para la obtención del grado de Especialista en Robótica e Inteligencia Artificial

Tutor: Prof. Ing. Marcelo José Rovai
Co-tutor: Prof. Dr. Marcelo Pias

Rivera, 2024

AGRADECIMIENTOS

Este trabajo final no hubiera sido posible sin el apoyo de varias personas e instituciones:

Al Magister Ingeniero Marcelo José Rovai por su labor como tutor durante la que ha demostrado no solo un gran conocimiento, sino también una comprensión y empatía sin las que el trabajo no hubiera sido posible.

Al Doctor Marcelo Pias pos su invaluable apoyo y compañamiento como co-tutor de este trabajo académico de posgrado.

A la Universidad Nacional de Rafaela, y en especial al director de las carreras de Mecatrónica y Automatización y Robótica Ingeniero Fernando Ferrer por su apoyo en todo momento.

A la empresa Desde el Llano S.A.S. por permitir el acceso a sus instalaciones de agricultura hidropónica y por todo el apoyo para poder realizar todas las tareas investigación relacionadas a este trabajo.

En especial a mi madre por ser un pilar fundamental, y a todos aquellos que durante este tiempo han ayudado a que este trabajo final sea hoy una realidad.

RESUMEN

CASTRO ABDALLAH, Gustavo Pablo. **Desarrollo de un Sistema embebido diseñado para la predicción de humedad en esponja fenólica de germinación hidropónica basado en RNN/LTSM.** 2024. 81 f. Proyecto Final – Posgrado en Robótica e Inteligencia Artificial (PRIA). Universidad Tecnológica (UTEC), Rivera.

Este proyecto se enmarca en la necesidad de innovar en la agricultura moderna, promoviendo la sostenibilidad y la eficiencia mediante el uso de tecnologías avanzadas. La hidroponía, un método de cultivo sin suelo, se presenta como una solución viable y sostenible ante los desafíos actuales de la agricultura, tales como la escasez de agua y la degradación del suelo. En este contexto, la capacidad de predecir la humedad del sustrato es crucial para optimizar el riego y mejorar la productividad de los cultivos.

En el presente trabajo final de posgrado se desarrolla un sistema embebido para la predicción de la humedad en esponja fenólica utilizada en la germinación hidropónica, basado en redes neuronales recurrentes de memoria a largo plazo (RNN-LSTM).

El estudio incluye una revisión exhaustiva del estado del arte de las redes neuronales y su implementación en sistemas embebidos, destacando técnicas de optimización como la cuantización y la poda, y el uso de librerías de código abierto como TensorFlow Lite y herramientas como Edge Impulse. Se desarrolla un datalogger basado en un microcontrolador ESP32, equipado con sensores ambientales y de humedad del sustrato, para recolectar datos durante un periodo de 66 días. Este dataset, compuesto por 1800 muestras para test y 800 para entrenamiento y validación, es normalizado utilizando la función MinMaxScaler de la librería Scikit-learn.

El capítulo de metodología detalla el proceso de diseño, implementación y entrenamiento del modelo LSTM, incluyendo la justificación del uso de esta arquitectura por su capacidad de manejar datos secuenciales y predecir valores futuros. Los resultados muestran que el modelo entrenado es capaz de predecir con precisión la humedad del sustrato, permitiendo una mejor gestión del riego en sistemas hidropónicos.

Finalmente, se discuten las contribuciones de este trabajo al campo de la agricultura hidropónica, destacando la importancia de contar con sistemas inteligentes para la supervisión y control de variables críticas, lo que puede llevar a un uso más eficiente de los recursos y a una mayor sostenibilidad en la producción agrícola.

Palabras-clave: RNN, LSTM, sistema embebido, esponja fenólica, hidroponia, agricultura, predicción, humedad.

ABSTRACT

CASTRO ABDALLAH, Gustavo Pablo. **Development of an embedded system for the prediction of humidity in hydroponic germination phenolic sponge based on RNN/LSTM.** 2024. 81f. Final Project – Postgraduate in Robotics and Artificial Intelligence (PRIA). Universidade Federal do Rio Grande (FURG), Rivera.

This project addresses the need to innovate in modern agriculture, promoting sustainability and efficiency through the use of advanced technologies. Hydroponics, a soilless cultivation method, emerges as a viable and sustainable solution to current agricultural challenges such as water scarcity and soil degradation. In this context, the ability to predict substrate moisture is crucial for optimizing irrigation and improving crop productivity.

This postgraduate final project develops an embedded system for predicting moisture in phenolic foam used in hydroponic germination, based on long short-term memory recurrent neural networks (RNN-LSTM).

The study includes a comprehensive review of the state of the art in neural networks and their implementation in embedded systems, highlighting optimization techniques such as quantization and pruning, and the use of libraries like TensorFlow Lite and tools like Edge Impulse. A datalogger based on an ESP32 microcontroller, equipped with environmental and substrate moisture sensors, is developed to collect data over a 66-day period. This dataset, comprising 1800 samples for testing and 800 for training and validation, is normalized using the MinMaxScaler function from the Scikit-learn library.

The methodology chapter details the design, implementation, and training process of the LSTM model, including the justification for using this architecture due to its ability to handle sequential data and predict future values. The results show that the trained model can accurately predict substrate moisture, enabling better irrigation management in hydroponic systems.

Finally, the contributions of this work to the field of hydroponic agriculture are discussed, highlighting the importance of having intelligent systems for monitoring and controlling critical variables, which can lead to more efficient use of resources and greater sustainability in agricultural production.

Keywords: RNN, LSTM, embedded system phenolic sponge, hydroponics, agriculture, prediction, humidity.

INDICE DE FIGURAS

| | | |
|----|---|----|
| 1 | Arquitectura Celda RNN vs. Arquitectura Celda LSTM | 23 |
| 2 | Seeed Studio XIAO ESP32S3 | 30 |
| 3 | Sensor Bosh BME280 | 32 |
| 4 | Espuma Fenólica | 32 |
| 5 | Espuma Fenólica TEN-LEAD - Características técnicas | 35 |
| 6 | Tipos de sondas FDR. (x): de placas planas, (y) de barras cilíndricas, (z): de anillos metálicos alrededor de un cilindro [Caicedo-Rosero et al., 2021] | 36 |
| 7 | Sensor Capacitivo de Medición de Humedad de Suelo V2.0 | 37 |
| 8 | Diagrama esquemático Sensor Capacitivo de medición de Humedad de Suelo | 37 |
| 9 | Ensamble de Sensor con carcasa de protección | 38 |
| 10 | Utilización del Sensor Capacitivo en sustratos | 38 |
| 11 | Ensayo del Sensor Capacitivo | 39 |
| 12 | Código Python, análisis regresión del sensor capacitivo | 40 |
| 13 | Resultados Regresión lineal Sensor Capacitivo | 40 |
| 14 | Esquemático Sistema Embebido | 41 |
| 15 | Diagrama Bloques Sistema Embebido | 41 |
| 17 | Datalogger | 46 |
| 18 | Datalog - Variables grabadas | 46 |
| 19 | Tipos de Redes LSTM | 51 |
| 20 | Pre-formateo de datos para la predicción | 52 |
| 21 | Curvas de pérdida: set de entrenamiento vs. validación | 56 |
| 22 | Comparación entre valores reales vs. valores predecidos por el modelo | 57 |
| 23 | Características Edge Impulse Python SDK | 58 |
| 24 | Hardware Sistema Embebido | 61 |
| 25 | Datos climáticos históricos Rafaela(Sta Fe- ARG) - fuente: <i>meteobr.com</i> | 64 |
| 26 | Dataset - Correlación entre variables de entrada | 65 |
| 27 | Sistema Embebido - Ensayo | 68 |
| 28 | Sistema embebido - Resultados de su ensayo | 70 |
| 29 | Circuito Electrónico del Datalogger | 79 |
| 30 | Circuito Electrónico del Sistema Embebido de Predicción de Humedad | 80 |

INDICE DE TABLAS

| | | |
|---|---|----|
| 1 | Comparativa Microprocesadores | 29 |
| 2 | Comparativa Placas de Desarrollo | 30 |
| 3 | BME280 - Características Generales | 31 |
| 4 | BME280 - Rango/Precisión/Resolución | 32 |
| 5 | Ensayo Esponja fenólica [Vout vs. HR] | 39 |
| 6 | Dataset - Rangos | 64 |
| 7 | Modelo LSTM - exactitud vs pasos futuros | 67 |
| 8 | Modelo LSTM - exactitud vs tamaño del vector de entrada [horas] | 68 |
| 9 | Comparación de Inferencia en familia de microprocesadores ESP32 | 68 |

LISTA DE ABREVIATURAS Y SIGLAS

| | |
|------|---------------------------------|
| CSV | Comma Separated Values |
| FDR | Frequency-Domain Reflectometry |
| FIFO | Firt Input First Output |
| IDE | Entorno de Desarrollo Integrado |
| IoT | Internet de las Cosas |
| LED | Light Emisor Diode |
| LSTM | Long short-term memory |
| ML | Machine Learning |
| NaN | Not A Number |
| PCB | Printed Circuit Board |
| RNN | Recurrent Neural Network |
| RTC | Real Time Clock |
| SDK | Software Development Kit |
| VWC | Volumetric Water Content |

SUMARIO

| | | |
|------------|---|-----------|
| 1 | Introducción | 10 |
| 1.1 | Contexto | 10 |
| 1.1.1 | Rol de la Hidroponía en la Agricultura | 10 |
| 1.2 | Justificación | 11 |
| 1.3 | Objetivos | 12 |
| 1.3.1 | Objetivo General | 12 |
| 1.3.2 | Objetivos Específicos | 12 |
| 1.4 | Contribuciones | 13 |
| 2 | REVISIÓN BIBLIOGRÁFICA | 14 |
| 2.1 | Germinación Hidropónica | 14 |
| 2.1.1 | Sus pilares fundamentales | 14 |
| 2.1.2 | Desafíos | 14 |
| 2.2 | Sensores de Humedad de suelo | 15 |
| 2.2.1 | Tecnologías y Técnicas relacionadas | 15 |
| 2.2.2 | Tipos Viables para Hidroponía | 16 |
| 2.3 | Redes Neuronales | 17 |
| 2.3.1 | Redes Neuronales en Predicción en Series Temporales | 17 |
| 3 | MARCO TEÓRICO | 19 |
| 3.1 | Plataformas de Desarrollo | 19 |
| 3.1.1 | Características | 19 |
| 3.2 | Sistemas Embebidos | 19 |
| 3.2.1 | Definición | 19 |
| 3.2.2 | Características | 19 |
| 3.3 | Entornos de Desarrollo Integrado (IDE) | 20 |
| 3.3.1 | Descripción | 20 |
| 3.3.2 | Características | 20 |
| 3.3.3 | Entornos de Desarrollo más Utilizados | 21 |
| 3.4 | Redes Neuronales | 21 |
| 3.4.1 | Redes Convolucionales (CNN) | 22 |
| 3.4.2 | Redes Neuronales Recurrentes (RNN) | 22 |
| 3.4.3 | Redes de Memoria larga de corto tiempo (LSTM) | 22 |
| 3.4.4 | Predicciones temporales con LSTM | 23 |
| 3.4.5 | Optimizaciones y Técnicas para LSTM en Sistemas Embebidos | 24 |
| 3.4.6 | Plataformas y Herramientas | 24 |
| 3.4.7 | Selección de una Red LSTM para el desarrollo del proyecto final | 24 |

| | |
|---|-----------|
| 4 METODOLOGÍA | 27 |
| 4.1 Diseño del Sistema | 28 |
| 4.1.1 Microcontrolador | 29 |
| 4.1.2 Sensor de Condiciones Ambientales | 30 |
| 4.1.3 Espuma Fenólica | 32 |
| 4.1.4 Sensor de Humedad de Esponja Fenólica | 34 |
| 4.1.5 Integración del Sistema Embebido | 40 |
| 4.2 Obtención del Dataset | 42 |
| 4.2.1 Concepto e Importancia del Dataset | 42 |
| 4.2.2 Generación del Dataset | 42 |
| 4.2.3 Estructura del Dataset y Variables Incluidas | 42 |
| 4.2.4 Importancia de Respetar la Periodicidad | 43 |
| 4.2.5 El Datalogger | 43 |
| 4.2.6 Pre-procesamiento del Dataset | 46 |
| 4.3 Implementación Red neuronal LSTM | 50 |
| 4.3.1 Redes Long Short-Term Memory (LSTM) | 50 |
| 4.3.2 Preparación de los datos para la predicción LSTM | 52 |
| 4.3.3 Creación de secuencias para LSTM (Features) | 52 |
| 4.3.4 Diseño y Entrenamiento del modelo LSTM | 53 |
| 4.4 Implementación RNN para el Sistema Embebido | 57 |
| 4.4.1 Create TFLite LSTM Model | 58 |
| 4.4.2 Implementación del modelo con Edge Impulse Python SDK | 59 |
| 4.5 Desarrollo del Sistema Embebido | 60 |
| 4.5.1 Desarrollo del hardware | 60 |
| 4.5.2 Desarrollo del Firmware | 61 |
| 5 RESULTADOS | 63 |
| 5.1 Evaluación del dataset | 63 |
| 5.2 Evaluación del Modelo LSTM | 65 |
| 5.2.1 Testeo de inferencia | 65 |
| 5.2.2 Testeo de Hiper-parámetros del modelo | 67 |
| 5.3 Evaluación del Sistema Embebido | 68 |
| 5.3.1 Energía | 68 |
| 5.3.2 Desempeño | 69 |
| 6 CONCLUSIONES y DISCUSIONES | 71 |
| 7 TRABAJOS FUTUROS | 73 |
| 7.1 Dataset | 73 |
| 7.2 Firmware | 73 |
| 7.3 Hardware | 73 |
| Bibliografía Consultada | 75 |
| ANEXOS | |
| A Circuitos Esquemáticos | 78 |
| B Repositorio GitHub | 81 |

1 INTRODUCCIÓN

1.1 Contexto

La hidroponia ha emergido como una técnica agrícola innovadora que ofrece una serie de beneficios significativos en comparación con los métodos tradicionales de cultivo en suelo. En este contexto, esta investigación se centra en un aspecto crucial de la hidroponia: mejorar la eficiencia en la etapa de germinación a través del control preciso de la humedad en el sustrato de cultivo, particularmente en esponjas fenólicas.

1.1.1 Rol de la Hidroponía en la Agricultura

En relación al rol de la hidroponia en la agricultura actual, es importante destacar su relevancia, beneficios y aplicaciones en el contexto agrícola contemporáneo [Hosseinzadeh et al., 2017].

1. Eficiencia de recursos: La hidroponía permite un uso más eficiente del agua al recircular continuamente soluciones nutritivas, lo que puede reducir el consumo de agua hasta en un 90 por ciento en comparación con la agricultura tradicional basada en el suelo [Boyland, 2020].
2. Aumento del Rendimiento y la Productividad: Al controlar de manera precisa los niveles de nutrientes y condiciones ambientales, la hidroponía puede aumentar significativamente el rendimiento de los cultivos en comparación con el cultivo en suelo. Esto se debe a que las plantas pueden acceder a nutrientes de manera más directa y eficiente.
3. Utilización de espacios limitados: La capacidad de cultivar en sistemas hidropónicos verticales y en interiores permite aprovechar eficazmente espacios limitados, como áreas urbanas o zonas con suelos poco fértiles, lo que abre nuevas oportunidades para la producción de alimentos local y sostenible.
4. Disminución de la erosión del suelo: Al prescindir del suelo como medio de cultivo, la hidroponía ayuda a reducir la erosión del suelo y la degradación de la tierra, lo que contribuye a la conservación de los recursos naturales y la biodiversidad.

5. Mayor control y precisión en los cultivos: La hidroponía permite un control preciso de las condiciones de crecimiento, incluyendo la temperatura, humedad, pH y niveles de nutrientes, lo que permite optimizar el ambiente para el crecimiento óptimo de las plantas y minimizar el riesgo de enfermedades y plagas.
6. Producción de Cultivos de alta calidad: Debido al ambiente controlado y la ausencia de contaminantes del suelo, los cultivos hidropónicos tienden a ser de alta calidad, con menos residuos de pesticidas y una mayor concentración de nutrientes y compuestos beneficiosos.
7. Sostenibilidad Ambiental: La hidroponia ofrece una alternativa sostenible a la agricultura convencional al reducir la huella hídrica, minimizar la utilización de productos químicos y pesticidas, y mitigar el impacto ambiental negativo asociado con la agricultura intensiva.
8. Resiliencia ante el cambio climático: La capacidad de cultivar en entornos controlados protege los cultivos de eventos climáticos extremos y variaciones en las condiciones climáticas, lo que aumenta la resiliencia del sistema agrícola frente al cambio climático.

En resumen, la hidroponia desempeña un papel crucial en la agricultura actual al ofrecer una serie de beneficios que abordan desafíos como la escasez de recursos, la degradación del suelo y los impactos del cambio climático. Su capacidad para producir alimentos de manera eficiente, sostenible y resiliente la convierte en una opción atractiva para el futuro de la agricultura.

1.2 Justificación

El desarrollo de tecnología que permita un control preciso de la humedad en esponjas fenólicas en sistemas hidropónicos se justifica por una serie de razones fundamentales, respaldadas por estudios previos en el campo de la agricultura sostenible y la hidroponía:

1. Eficiencia en el Uso de Recursos: La capacidad de la hidroponía para optimizar el uso de agua y nutrientes ha sido ampliamente documentada. Por ejemplo, según [Al-Chalabi and Kuo, 2018], los sistemas hidropónicos pueden reducir el consumo de agua en la agricultura hasta en un 90% en comparación con los métodos tradicionales de riego por suelo.
2. Optimización del Crecimiento de las Plantas: Investigaciones como el estudio de [Savvas and Gruda, 2018] han demostrado que la hidroponía permite un control preciso de las condiciones de cultivo, lo que conduce a un aumento significativo en el rendimiento y la calidad de los cultivos en comparación con los métodos de cultivo en suelo.

3. Reducción de Pérdidas de Cultivos: La capacidad de la hidroponía para prevenir problemas como el estrés hídrico y el exceso de riego puede ayudar a reducir las pérdidas de cultivos. Estudios como el de D. B. Hochmuth et al. (2018) han destacado la importancia de la gestión precisa del agua en la prevención de enfermedades y el aumento de la productividad de los cultivos en sistemas hidropónicos.
4. Resiliencia ante Condiciones Ambientales Cambiantes: La capacidad de la hidroponía para adaptarse a condiciones ambientales cambiantes ha sido reconocida en estudios como el de F. Leskovar et al. (2018), que señala que los sistemas hidropónicos ofrecen una mayor flexibilidad y capacidad de respuesta a eventos climáticos extremos en comparación con los métodos de cultivo en suelo.
5. Promoción de la Innovación Tecnológica: El desarrollo de sistemas embebidos basados en tecnologías avanzadas como las redes LSTM para la predicción de humedad en esponjas fenólicas impulsa la innovación tecnológica en el sector agrícola. Este enfoque ha sido respaldado por estudios como el de R. Shrestha et al. (2020), que explora el potencial de la inteligencia artificial y el aprendizaje automático en la agricultura de precisión.

La investigación en hidroponía y tecnologías asociadas contribuye al avance de la agricultura sostenible al proporcionar herramientas y tecnologías que permiten a los agricultores operar de manera más eficiente, rentable y respetuosa con el medio ambiente a largo plazo. Este punto ha sido respaldado por organizaciones como la Organización de las Naciones Unidas para la Agricultura y la Alimentación (FAO) en su informe sobre agricultura sostenible (FAO, 2018).

1.3 Objetivos

1.3.1 Objetivo General

Desarrollar un Sistema embebido para la predicción de humedad en esponja fenólica de germinación hidropónica con el objeto de mejorar la eficiencia durante este proceso productivo.

1.3.2 Objetivos Específicos

- Generar un dataset que incluya la interrelación de variables de ambiente y variables de germinación en entorno de producción de la fase de germinación de agricultura hidropónica
- Desarrollar, entrenar y realizar el deployment de una red neuronal tipo RNN/LSTM (Recurrent Neural Network / Long Short Time Memory) del tipo multivariable-univariable en un sistema embebido.

- Implementar un dispositivo de bajo costo y dimensiones con la capacidad de medir condiciones climáticas y de suelo, predecir escenarios futuros de necesidad de agua utilizando el modelo entrenado y emitir una alarma en caso que la predicción no sea favorable dentro de límites adecuados para el cultivo.
- Ensayar y establecer la fiabilidad del diseño en funcionamiento.

1.4 Contribuciones

Contar con un sistema de supervisión inteligente capaz de predecir la falta de humedad en un tiempo futuro en sistemas hidropónicos aportaría varias contribuciones significativas al sector de la agricultura por hidroponía:

- Optimización del Uso de Recursos: El sistema permitiría un uso más eficiente del agua al anticipar y prevenir la falta de humedad en las esponjas fenólicas, evitando así el riego excesivo y el desperdicio de agua.
- Reducción de Pérdidas de Cultivos: Al predecir y alertar sobre la falta de humedad en el sustrato, el sistema ayudaría a prevenir el estrés hídrico en las plantas, reduciendo las pérdidas de cultivos asociadas con este problema.
- Mejora en la Calidad y Rendimiento de los Cultivos: Al mantener niveles óptimos de humedad en el sustrato, el sistema contribuiría a un crecimiento más saludable y robusto de las plantas, lo que resultaría en una mayor calidad y rendimiento de los cultivos.
- Aumento de la Eficiencia Operativa: Al proporcionar alertas tempranas sobre la falta de humedad, el sistema permitiría a los agricultores tomar medidas preventivas de manera oportuna, evitando la pérdida de tiempo y recursos asociados con la corrección de problemas de humedad después de que hayan ocurrido.
- Promoción de la Sostenibilidad: Al mejorar la eficiencia y la productividad de los sistemas hidropónicos, el sistema de supervisión inteligente contribuiría a hacer la agricultura por hidroponía más sostenible y viable a largo plazo, lo que podría fomentar su adopción y expansión en diferentes regiones y contextos agrícolas.

En definitiva, el presente trabajo final persigue desarrollar un sistema embebido que pueda ser transformado en productor mínimo viable que pueda poner a disposición del pequeño y mediano productor estas contribuciones, mejorando la eficiencia operativa, reduciendo las pérdidas de cultivos y colaborando con la sostenibilidad ambiental y su sustentabilidad económica.

2 REVISIÓN BIBLIOGRÁFICA

2.1 Germinación Hidropónica

La germinación de semillas en sistemas hidropónicos es un proceso crucial que establece las bases para el crecimiento y desarrollo de las plantas en este sistema de cultivo. En esta sección, se analiza el estado actual de la germinación en la agricultura hidropónica, destacando tanto los avances logrados como los desafíos que aún persisten.

2.1.1 Sus pilares fundamentales

La eficiencia de la germinación en agricultura hidropónica se caracteriza por los siguientes pilares fundamentales:

- **Eficiencia en la Germinación:** Investigaciones como la de Li [Li et al., 2020] han demostrado que la germinación en sistemas hidropónicos puede ser altamente eficiente, con tasas de germinación comparables o incluso superiores a las obtenidas en suelo.
- **Control de Condiciones Ambientales:** La germinación en hidroponía permite un control preciso de las condiciones ambientales, como la temperatura, humedad y disponibilidad de nutrientes, lo que favorece un desarrollo inicial óptimo de las plántulas.
- **Impacto en la Producción:** Estudios como el de Silva [Silva et al., 2019] han demostrado que una germinación exitosa en sistemas hidropónicos puede tener un impacto positivo significativo en la producción final de cultivos, mejorando el rendimiento y la calidad de los mismos.

2.1.2 Desafíos

A pesar de los avances logrados, la germinación en agricultura hidropónica también enfrenta una serie de desafíos:

- **Manejo de la Humedad:** La gestión adecuada de la humedad durante el proceso de germinación es fundamental para garantizar un desarrollo óptimo de las plántulas.

Sin embargo, mantener niveles de humedad adecuados en el sustrato puede ser un desafío, especialmente en sistemas automatizados.

- Control de Enfermedades: La germinación en sistemas hidropónicos puede estar asociada con un mayor riesgo de enfermedades, como la pudrición de la raíz, debido a la alta humedad y la ausencia de competencia microbiana del suelo. Abordar este desafío requiere estrategias efectivas de control de enfermedades [Wu et al., 2021]
- Optimización de Nutrientes: La disponibilidad de nutrientes durante la fase de germinación es crucial para el desarrollo inicial de las plántulas. Sin embargo, garantizar un suministro adecuado de nutrientes sin provocar fitotoxicidad puede ser un desafío en sistemas hidropónicos [Sánchez-García et al., 2019]

2.2 Sensores de Humedad de suelo

La medición precisa de la humedad del suelo es fundamental para el monitoreo y control eficiente del riego en la agricultura. En esta sección, se analiza el estado actual de los sensores de humedad en suelo, centrándose en su aplicación en agronomía y su viabilidad para su uso en sistemas hidropónicos.

2.2.1 Tecnologías y Técnicas relacionadas

2.2.1.1 Principios de Funcionamiento

Los sensores de humedad en suelo utilizan una variedad de principios físicos y técnicas de medición para determinar la cantidad de agua presente en el suelo [Garg et al., 2016]. Algunos de los principales principios de funcionamiento son:

- **Capacitancia:** Los sensores de humedad basados en capacitancia son uno de los tipos más comunes de sensores utilizados en la medición de humedad del suelo. Estos sensores consisten en electrodos enterrados en el suelo que forman un capacitor con el agua presente en el suelo. La cantidad de agua afecta la capacitancia del sensor, que a su vez se convierte en una medida de la humedad del suelo. Estos sensores son especialmente adecuados para suelos con alta conductividad eléctrica y son relativamente inmunes a la salinidad y la compactación del suelo [Sharma, 2018]
- **Resistividad Eléctrica:** Los sensores de humedad en suelo basados en resistividad eléctrica miden la resistencia eléctrica del suelo, que está inversamente relacionada con la humedad del suelo. Cuanto más húmedo esté el suelo, menor será su resistencia eléctrica. Estos sensores suelen consistir en electrodos enterrados en el suelo que aplican una corriente eléctrica y miden la caída de tensión resultante para determinar la resistencia del suelo. Son particularmente útiles en suelos con baja

conductividad eléctrica y pueden ser sensibles a la salinidad y la compactación del suelo [Sharma, 2018]

- **TDR (Time Domain Reflectometry):** Los sensores de humedad en suelo basados en TDR utilizan pulsos de microondas para medir la humedad del suelo. Estos sensores emiten pulsos de microondas a través de una sonda enterrada en el suelo y miden el tiempo que tarda en reflejarse la señal de vuelta. Este tiempo de reflexión está relacionado con la constante dieléctrica del suelo, que a su vez está relacionada con la humedad del suelo. Los sensores basados en TDR son altamente precisos y pueden proporcionar mediciones a diferentes profundidades del suelo, pero tienden a ser más costosos y requieren una instalación más compleja [Sharma, 2018]
- **Tensiometría:** Aunque menos comunes en aplicaciones de monitoreo de humedad del suelo, los tensímetros miden la tensión del agua en el suelo, que está relacionada con la humedad del suelo. Los tensímetros consisten en un tubo poroso lleno de agua conectado a un manómetro. A medida que el suelo se seca, el agua es absorbida por el suelo y la tensión en el tubo poroso aumenta, lo que se refleja en el manómetro. Los tensímetros son particularmente útiles en suelos con capacidad de retención de agua y pueden proporcionar mediciones precisas en condiciones de baja conductividad eléctrica [Sharma, 2018]

2.2.1.2 Precisión y Fiabilidad

Los avances en tecnología han mejorado significativamente la precisión y fiabilidad de los sensores de humedad en suelo, permitiendo mediciones más exactas y consistentes en una variedad de condiciones de suelo y cultivo.

2.2.1.3 Integración con Sistemas de Monitoreo

Los sensores de humedad en suelo se pueden integrar con sistemas de monitoreo y control automatizados, permitiendo un seguimiento continuo y ajuste preciso del riego en función de las necesidades hídricas de los cultivos.

2.2.2 Tipos Viables para Hidroponía

Algunos tipos de sensores de humedad en suelo son especialmente viables para su aplicación en sistemas hidropónicos [Ling, 2004]

- **Sensores de Capacitancia:** Los sensores de capacitancia son comúnmente utilizados en hidroponia debido a su capacidad para medir la humedad del sustrato con precisión y su resistencia a la corrosión y la interferencia eléctrica.
- **Sensores de Tensiometría:** Aunque menos comunes en hidroponia, los sensores de tensiometría pueden ser útiles para medir la humedad en el sustrato en sistemas

hidropónicos donde se utiliza un sustrato con capacidad de retención de agua, como la fibra de coco..

2.3 Redes Neuronales

2.3.1 Redes Neuronales en Predicción en Series Temporales

En la investigación aplicada a la predicción de series temporales, las redes neuronales se han convertido en una herramienta poderosa y versátil. Se ha observado que, debido a su capacidad para modelar patrones no lineales y complejos, las Redes Neuronales Artificiales (ANN) y sus variaciones han demostrado ser altamente efectivas en este tipo de tareas.

2.3.1.1 Trabajos Previos con Redes Neuronales

Las ANN convencionales, como las Perceptron Multicapa (MLP), fueron de las primeras en ser utilizadas en predicciones temporales. Sin embargo, su desempeño se ha visto limitado en secuencias largas debido a la incapacidad de mantener una memoria a largo plazo en la estructura de la red. Esto condujo al desarrollo de arquitecturas más sofisticadas, como las Redes Neuronales Recurrentes (RNN) y su posterior mejora con las Redes de Memoria de Corto y Largo Plazo (LSTM), que han sido particularmente eficaces en el manejo de series temporales dependientes.

Trabajos como el de [Gers et al., 1999] introdujeron las LSTM como una solución al problema del desvanecimiento del gradiente en las RNN estándar, permitiendo que la red mantuviera información durante secuencias largas, lo cual es crítico en la predicción de series temporales. Esta arquitectura ha sido ampliamente utilizada en aplicaciones como la predicción de ventas, el pronóstico de demanda de energía y la predicción meteorológica.

Tradicionalmente, las redes convolucionales (CNN) han sido empleadas principalmente en tareas de visión por computadora, como la clasificación de imágenes, la detección de objetos y el procesamiento de imágenes. Sin embargo, en años recientes, se ha explorado su aplicación en series temporales, debido a su capacidad para detectar patrones locales mediante convoluciones. Aunque las CNN no son naturalmente recurrentes y no pueden mantener dependencias a largo plazo como las redes LSTM, han mostrado ser útiles en ciertos contextos de predicción de series temporales, particularmente cuando las dependencias son más cortas y los patrones locales son cruciales.

Investigaciones como la de [Borovskykh et al., 2017] han mostrado cómo las CNN pueden ser aplicadas con éxito en la predicción de series temporales, proponiendo la arquitectura de redes convolucionales aplicadas a secuencias de tiempo. Aunque las CNN no almacenan información a largo plazo, su capacidad de detectar características espaciales y locales en los datos ha llevado a combinarlas con redes LSTM en arquitecturas híbridas

(CNN-LSTM), donde las CNN extraen características y las LSTM modelan la temporalidad a largo plazo. Aunque las CNN no están diseñadas para manejar dependencias temporales largas, su capacidad de extraer patrones locales las ha hecho útiles en la predicción de series temporales, especialmente cuando se utilizan en combinación con redes recurrentes como las LSTM. Esto ha permitido aprovechar lo mejor de ambos mundos en modelos híbridos.

Varios estudios han demostrado la superioridad de las LSTM frente a modelos tradicionales, como el análisis de series temporales ARIMA o las redes neuronales feedforward. Por ejemplo, en el estudio de [Hochreiter and Schmidhuber, 1997], las LSTM mostraron una mejora significativa en la predicción de datos temporales complejos en comparación con otros enfoques.

2.3.1.2 Aplicaciones Específicas de LSTM

Entre los trabajos que destacan el uso de LSTM para predicciones temporales, se encuentran los de [Graves et al., 2013] en el reconocimiento de patrones de lenguaje, así como el de [Malhotra et al., 2015] , quienes emplearon LSTM para el monitoreo de máquinas industriales en tiempo real, prediciendo fallos a partir de las series de datos de sensores.

Otra aplicación relevante es el trabajo de [Brownlee, 2017] , que aborda el uso de LSTM en la predicción de series de tiempo económicas y financieras, mostrando que las LSTM superan ampliamente a las RNN convencionales y otros enfoques clásicos en términos de precisión de predicción.

2.3.1.3 LSTM en Sistemas Embebidos

Recientemente, el uso de LSTM en sistemas embebidos ha cobrado importancia debido al crecimiento de la tecnología TinyML. [Iodice, 2022] destacó en su libro "TinyML Cookbook" cómo las técnicas de optimización de hardware y modelos más ligeros permiten implementar LSTM en plataformas de bajo consumo, como el ESP32, logrando un balance entre capacidad de predicción y eficiencia energética en dispositivos con recursos limitados.

Existen varios casos de estudio que demuestran la viabilidad y efectividad de implementar RNN-LSTM en sistemas embebidos para aplicaciones del mundo real. Un ejemplo notable es el uso de LSTM para la predicción de series temporales en dispositivos de monitoreo ambiental [Tovar Macías, 2024], donde las restricciones de energía y procesamiento son críticas. Otro caso de uso incluye la detección de actividad humana mediante sensores portátiles, donde las RNN-LSTM procesan datos de acelerómetros y giroscopios para identificar patrones de movimiento en tiempo real [Sáez Bombín et al., 2018].

3 MARCO TEÓRICO

3.1 Plataformas de Desarrollo

3.1.1 Características

3.2 Sistemas Embebidos

Los sistemas embebidos juegan un papel fundamental en el desarrollo de tecnologías innovadoras en diversos campos, incluyendo la ingeniería robótica, la automatización industrial y la agricultura de precisión. Esta sección proporciona una visión detallada sobre los sistemas embebidos, comenzando con su definición y luego explorando sus características principales.

3.2.1 Definición

Los sistemas embebidos se definen como sistemas informáticos especializados diseñados para realizar funciones específicas dentro de un sistema más grande o un dispositivo electrónico. Estos sistemas están integrados en el hardware del dispositivo y están optimizados para eficiencia en recursos, tamaño compacto y bajo consumo de energía. Los sistemas embebidos se encuentran en una amplia variedad de dispositivos y aplicaciones, desde electrodomésticos y dispositivos médicos hasta vehículos autónomos y sistemas de control industrial.

3.2.2 Características

Las características clave de los sistemas embebidos incluyen:

- **Eficiencia en Recursos:** Los sistemas embebidos están diseñados para funcionar con recursos limitados, incluyendo memoria, procesamiento y energía. Esto requiere un diseño cuidadoso de software y hardware para optimizar el rendimiento y la eficiencia del sistema.
- **Tiempo Real:** Muchos sistemas embebidos requieren capacidades de tiempo real, lo que significa que deben responder a eventos y entradas del entorno dentro de

límites de tiempo estrictos. Esto es crucial en aplicaciones donde la precisión temporal es crítica, como en sistemas de control de vuelo de drones o en sistemas de monitoreo y control de procesos industriales.

- **Conectividad:** Con el avance de la Internet de las Cosas (IoT), los sistemas embebidos cada vez más requieren capacidades de conectividad para intercambiar datos con otros dispositivos o sistemas a través de redes inalámbricas o cableadas.
- **Fiabilidad y Seguridad:** Dado que muchos sistemas embebidos se utilizan en aplicaciones críticas donde el fallo del sistema puede tener consecuencias graves, la fiabilidad y seguridad son aspectos fundamentales. Esto implica la implementación de técnicas de diseño y pruebas para garantizar la estabilidad y la integridad del sistema en todo momento.
- **Flexibilidad y Escalabilidad:** Aunque los sistemas embebidos están diseñados para realizar funciones específicas, también deben ser flexibles y escalables para adaptarse a cambios en los requisitos del sistema o para soportar actualizaciones de software y hardware en el futuro.

3.3 Entornos de Desarrollo Integrado (IDE)

Los Entornos de Desarrollo Integrado (IDE, por sus siglas en inglés) son herramientas de software que proporcionan un conjunto integrado de funciones para el desarrollo de software. Estas herramientas ofrecen un entorno unificado que incluye un editor de código, herramientas de compilación, depuración, y otras características que facilitan el desarrollo y la depuración de aplicaciones.

3.3.1 Descripción

Los IDEs son esenciales en el proceso de desarrollo de software, ya que proporcionan un entorno centralizado para escribir, probar y depurar código. Estas herramientas suelen incluir características como resaltado de sintaxis, completado automático de código, navegación de código, integración con sistemas de control de versiones, y capacidades de depuración avanzadas.

3.3.2 Características

Algunas de las características comunes de los IDEs incluyen:

- **Editor de Código:** Ofrece un entorno para escribir y editar código fuente con resaltado de sintaxis y otras características de productividad.
- **Herramientas de Compilación:** Proporciona herramientas para compilar código fuente en ejecutables o bibliotecas.

- **Depuración:** Permite la detección y corrección de errores en el código a través de funciones de depuración interactiva.
- **Gestión de Proyectos:** Facilita la organización de archivos y recursos en proyectos de desarrollo.
- **Integración con Sistemas de Control de Versiones:** Permite la integración con sistemas como Git o SVN para el control de versiones del código fuente.
- **Automatización de Tareas:** Ofrece herramientas para automatizar tareas repetitivas, como la construcción y prueba de código.

3.3.3 Entornos de Desarrollo más Utilizados

Existen numerosos IDEs disponibles para diferentes lenguajes de programación y plataformas de desarrollo. Algunos de los IDEs más utilizados incluyen:

- **Visual Studio¹:** Desarrollado por Microsoft, es uno de los IDEs más populares para el desarrollo de aplicaciones en entornos Windows.
- **Eclipse²:** Un IDE de código abierto ampliamente utilizado en la industria del software, con soporte para múltiples lenguajes de programación.
- **IntelliJ IDEA³:** Un IDE desarrollado por JetBrains, conocido por su soporte avanzado para el desarrollo en Java y otros lenguajes.
- **PyCharm⁴:** Otro IDE de JetBrains, especializado en el desarrollo de aplicaciones en Python.
- **Arduino IDE⁵:** Un entorno de desarrollo específico para el desarrollo de proyectos con placas Arduino, que incluye herramientas para la compilación y carga de código en las placas Arduino.

3.4 Redes Neuronales

Las redes neuronales son modelos computacionales que se inspiran en la estructura y funcionamiento del cerebro humano, siendo capaces de aprender representaciones complejas a partir de datos de entrada. Se componen de capas de neuronas interconectadas, donde cada neurona procesa la información de entrada y la transmite a las siguientes capas mediante operaciones matemáticas y funciones de activación [Goodfellow et al., 2016].

¹<https://code.visualstudio.com/>

²<https://eclipseide.org/>

³<https://www.jetbrains.com/idea/>

⁴<https://www.jetbrains.com/pycharm/>

⁵<https://www.arduino.cc/en/software>

3.4.1 Redes Convolucionales (CNN)

Una red convolucional es simplemente una red neuronal que utiliza la convolución en lugar de la habitual multiplicación de matrices en al menos una de sus capas [Goodfellow et al., 2016]. En Machine Learning, las entradas a estas redes son arrays multi-dimensionales, denominados tensores. Además, estas convolucionales no las vamos a realizar solamente en una dimensión, sino en dos, es decir, vamos a utilizar un kernel bidimensional. Este tipo de redes neuronales, formadas, conceptualmente, por una serie de capas, utilizan lo que se conoce como “Weight sharing”, es decir, se comparten los pesos entre las entradas que contengan la misma información y se entrena de forma conjunta. Esto quiere decir que, por ejemplo, en imágenes, si en dos ejemplos que se introducen a la red aparece el mismo objeto, pero ubicado en una zona distinta en cada imagen, los pesos de esas dos entradas se compartirán y entrenarán de forma conjunta, ya que lo que se quiere es que la red aprenda lo que es el objeto, sin importar su ubicación en la imagen, es decir, se consigue una invarianza espacial. En este caso se realiza un “Weight sharing” espacial, realizado mediante convoluciones.

3.4.2 Redes Neuronales Recurrentes (RNN)

Al igual que las redes neuronales convolucionales, este tipo de redes utiliza el “Weight sharing”, pero en este caso, en vez de espacial, se comparte de manera temporal.

Al trabajar con secuencias, la red estará compuesta conceptualmente por una serie de celdas que mantendrán un cierto estado correspondiente a cada unidad de tiempo considerada. Al tratarse de una secuencia, la back-propagation con la que actualizamos los pesos de la red, interesaría poder hacerla hasta el inicio de la secuencia para tener una idea de la variación de toda la secuencia a lo largo del tiempo, sin embargo, en la práctica, con secuencias largas, esto no va a ser posible, realizándose esta propagación hasta donde se pueda.

Por ello, las RNN son un tipo especializado de red neuronal diseñado para modelar datos secuenciales, donde la salida de una neurona en un momento dado se convierte en una entrada para la misma neurona en el siguiente momento. Esto permite a las RNN capturar dependencias temporales en los datos, lo que las hace adecuadas para tareas como el procesamiento del lenguaje natural, la traducción automática y la predicción de series temporales. Sin embargo, las RNN tradicionales pueden tener dificultades para aprender dependencias a largo plazo debido al problema de desvanecimiento del gradiente [Hochreiter and Schmidhuber, 1997].

3.4.3 Redes de Memoria larga de corto tiempo (LSTM)

Las redes de memoria larga de corto plazo (LSTM) son una variante de las RNN diseñadas para abordar el problema del desvanecimiento del gradiente. Introducen unidades de memoria especializadas que permiten que la red mantenga y actualice la infor-

mación relevante a lo largo del tiempo, lo que les permite capturar dependencias a largo plazo en los datos de secuencia. Las LSTM han demostrado ser efectivas en una variedad de tareas de modelado de secuencias, incluyendo el procesamiento del lenguaje natural y la predicción de series temporales [Graves et al., 2013].

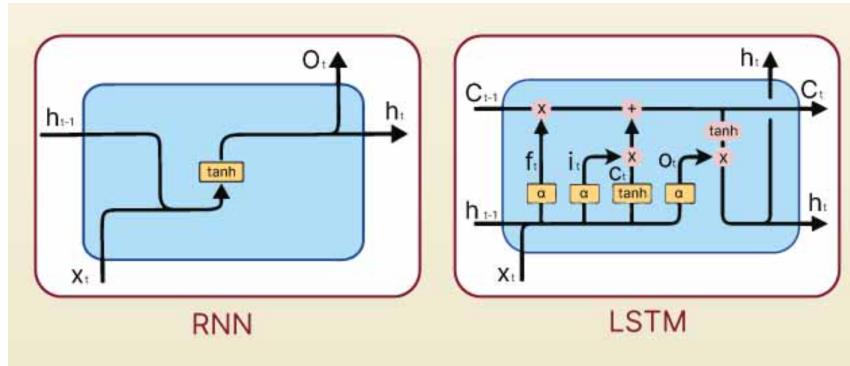


Figura 1: Arquitectura Celda RNN vs. Arquitectura Celda LSTM

Como se puede ver en la figura 1 , la celda LSTM trabaja como si fuera una pequeña red neuronal, permitiendo reducir el problema del vanishing gradient. Esta celda está constituida por una pequeña unidad de memoria que permite recordar el estado temporal de la red, además de unas unidades multiplicativas llamadas puertas para el control del flujo de la información. Cada celda contiene una “puerta de entrada” que controla el flujo de entrada de activations a la unidad de memoria, una “puerta de salida”, que controla el flujo de salida de activations de la unidad de memoria al resto de la red y una “puerta de olvido”, que escala el estado interno de la celda antes de añadirlo como una entrada a la celda a través de conexiones recurrentes propias de la celda, permitiendo resetear de forma adaptativa la memoria (el estado) de la celda.(Sak, H. et al., 2014), lo que hace que evite el problema del vanishing gradient. Además, contiene lo que es conocido como “conexiones mirilla” (peephole connections) entre la unidad de memoria y las puertas, lo que permite realizar todas las acciones indicadas con una cierta probabilidad dependiendo del contexto de la red y de las entradas y salidas anteriores, no como una decisión binaria.

3.4.4 Predicciones temporales con LSTM

Las redes LSTM son particularmente adecuadas para la predicción de series temporales debido a su capacidad para capturar dependencias temporales complejas en los datos. En el contexto del trabajo final de posgrado, las redes LSTM pueden utilizarse para predecir la humedad del sustrato fenólico en sistemas hidropónicos basándose en datos históricos de humedad del sustrato y variables ambientales. Al entrenar una red LSTM con datos históricos, la red puede aprender patrones temporales en los datos y realizar predicciones precisas sobre la humedad futura del sustrato.

3.4.5 Optimizaciones y Técnicas para LSTM en Sistemas Embebidos

Una de las principales estrategias para implementar RNN-LSTM en sistemas embebidos es la optimización del modelo. Esto incluye técnicas como la cuantización, que reduce la precisión de los pesos del modelo de flotante a entero, disminuyendo el uso de memoria y acelerando el proceso de inferencia sin sacrificar significativamente la precisión del modelo. La cuantización post-entrenamiento y la cuantización durante el entrenamiento son métodos específicos que se aplican para adaptar modelos grandes a dispositivos con recursos limitados.

Otra técnica crucial es la poda de redes neuronales, que implica eliminar conexiones y neuronas redundantes en el modelo. Esta técnica no solo reduce el tamaño del modelo, sino que también mejora la eficiencia computacional. La poda estructurada y no estructurada son variantes de esta técnica que se aplican dependiendo del diseño del hardware y los requisitos específicos de la aplicación [Iodice, 2022].

3.4.6 Plataformas y Herramientas

El desarrollo de plataformas y herramientas especializadas también ha facilitado la implementación de RNN-LSTM en sistemas embebidos. TensorFlow Lite y TensorFlow Lite for Microcontrollers son librerías de código abierto prominentes que permiten la conversión y despliegue de modelos de aprendizaje profundo en dispositivos con recursos limitados. Estas librerías soportan una variedad de técnicas de optimización y están diseñadas para integrarse fácilmente con microcontroladores como el ESP32 y los que poseen CPUs del tipo ARM Cortex-M.

Otra herramienta destacada es Edge Impulse Studio (<https://edgeimpulse.com>), una plataforma que permite el desarrollo y despliegue de modelos de machine learning en dispositivos embebidos. Edge Impulse proporciona un entorno integrado para la recolección de datos, el diseño de modelos y el 'deployment' a hardware embebido. Su compatibilidad con diversas plataformas de hardware y su enfoque en la facilidad de uso la hacen una opción atractiva para proyectos que buscan implementar modelos de RNN-LSTM en sistemas con recursos limitados. La plataforma soporta técnicas de optimización y permite una implementación eficiente y rápida, facilitando la integración de modelos en dispositivos como el ESP32 y otros microcontroladores comunes.

3.4.7 Selección de una Red LSTM para el desarrollo del proyecto final

Hay varias razones sólidas para defender el uso de LSTM (Long Short-Term Memory) en este proyecto de predicción de humedad en esponjas fenólicas para germinación hidropónica, en primer instancia desde el punto de vista profesional significa un desafío poder lograr desarrollarlo ya que recién en los meses de desarrollo del proyecto comenzaron a existir las herramientas que posibilitaban embeber este tipo de redes, lo cual también conlleva el alto grado de innovación de los objetivos propuestos; y desde el punto de vista

técnico se justificaría con los siguientes puntos:

1. Capacidad de manejar dependencias a largo plazo: Las LSTM están específicamente diseñadas para capturar y recordar patrones a largo plazo en series temporales. Esto es crucial en un entorno de germinación, donde las condiciones pasadas (horas o días anteriores) pueden tener un impacto significativo en la humedad actual y futura del sustrato.
2. Manejo de múltiples variables de entrada: El proyecto utiliza varias variables de entrada como temperatura, humedad relativa ambiental y presión atmosférica. Las LSTM son excelentes para manejar entradas multivariadas y capturar las complejas interacciones entre estas variables a lo largo del tiempo.
3. Predicción de series temporales: La predicción de humedad es esencialmente un problema de serie temporal. Las LSTM han demostrado un rendimiento superior en muchas tareas de predicción de series temporales en comparación con otros modelos.
4. Resistencia al problema de desvanecimiento del gradiente: A diferencia de las RNN simples, las LSTM son mucho menos susceptibles al problema de desvanecimiento del gradiente, lo que les permite aprender dependencias a más largo plazo de manera efectiva.
5. Capacidad de filtrar información irrelevante: La arquitectura de las LSTM con sus "puertas" permite al modelo aprender qué información es relevante para mantener y qué información puede ser descartada, lo cual es útil en un entorno con múltiples variables de entrada.
6. Adaptabilidad a cambios en patrones: En un entorno de germinación, pueden ocurrir cambios en los patrones debido a factores externos. Las LSTM tienen la capacidad de adaptarse a estos cambios en los patrones temporales.
7. Éxito probado en aplicaciones similares: Las LSTM han sido utilizadas con éxito en muchas aplicaciones de predicción en agricultura y sistemas de control, lo que respalda su elección para este proyecto.
8. Implementación en sistemas embebidos: Como se muestra en el proyecto, las LSTM pueden implementarse eficazmente en sistemas embebidos de bajo costo como los ESP32, lo que las hace adecuadas para aplicaciones prácticas en el campo.
9. Capacidad de manejar datos irregulares: En caso de que haya interrupciones en la recolección de datos o irregularidades en los intervalos de muestreo, las LSTM pueden manejar estas situaciones mejor que algunos otros modelos.

10. Balance entre complejidad y rendimiento: Las LSTM ofrecen un buen equilibrio entre la complejidad del modelo y el rendimiento predictivo, lo que las hace adecuadas para este tipo de aplicación práctica.

En resumen, la elección de LSTM para este proyecto se justifica por su capacidad para manejar efectivamente las complejidades de la predicción de humedad en un entorno de germinación hidropónica, considerando múltiples variables de entrada, capacidad de manejar datos irregulares y patrones temporales a largo plazo, mientras se mantiene la posibilidad de implementación en sistemas embebidos de bajo costo.

4 METODOLOGÍA

Durante el presente capítulo se describirá metodológicamente el desarrollo integral del proyecto final, conectando los objetivos del trabajo con las acciones, investigaciones, ensayos, desarrollos y metodologías implementadas a lo largo del proceso. Este recorrido meticuloso no solo detalla el camino seguido, sino que también ilumina el rigor y la precisión con los que se abordaron cada uno de los desafíos encontrados.

Desde el inicio, la ambición de este proyecto se estableció claramente: desarrollar un sistema embebido capaz de predecir la humedad en esponjas fenólicas utilizadas en la germinación hidropónica, empleando redes neuronales recurrentes de memoria a largo plazo (LSTM). Esta meta no solo requiere una comprensión profunda de los principios teóricos subyacentes, sino también la aplicación práctica de técnicas avanzadas de inteligencia artificial y el manejo de hardware especializado.

Para alcanzar estos objetivos, el trabajo se dividió en varias etapas interrelacionadas. En primer lugar, se llevó a cabo una exhaustiva revisión bibliográfica que abarcó desde los fundamentos de los sistemas embebidos hasta las últimas innovaciones en redes neuronales y predicción de series temporales. Esta base teórica robusta permitió la identificación de las mejores prácticas y enfoques metodológicos a seguir.

En la fase de investigación y diseño, se seleccionaron cuidadosamente los componentes de hardware y software. Se evaluaron diferentes plataformas de microcontroladores y sensores de humedad, considerando factores como la precisión, el consumo energético y la facilidad de integración. Paralelamente, se desarrollaron y probaron múltiples arquitecturas de redes LSTM, ajustando hiperparámetros y optimizando el rendimiento mediante técnicas de validación cruzada.

Los ensayos experimentales jugaron un papel crucial en la validación del sistema. Se realizaron numerosas pruebas en condiciones controladas, recopilando datos de humedad del sustrato fenólico, temperatura ambiente, presión atmosférica y humedad relativa. Estos datos fueron esenciales para entrenar y evaluar el modelo de predicción, garantizando su precisión y fiabilidad.

Cada desarrollo técnico estuvo acompañado de una rigurosa documentación y análisis de resultados. Los métodos de análisis de datos, las herramientas estadísticas y las

técnicas de visualización fueron empleadas para interpretar los resultados de manera clara y concisa. Esta documentación no solo sirve como evidencia del trabajo realizado, sino que también proporciona insights valiosos para futuras investigaciones.

Finalmente, la integración de todos los componentes en un sistema funcional y la validación del mismo en un entorno real concluyeron el proceso metodológico. Este capítulo detalla cada uno de estos pasos, proporcionando una visión completa del desarrollo del proyecto desde su concepción hasta su implementación final. En resumen, la metodología aquí descrita no solo persigue el cumplimiento de los objetivos planteados, sino que también refleja mi compromiso y pasión con la innovación en el campo de la inteligencia artificial aplicada a la agricultura hidropónica.

4.1 Diseño del Sistema

Para el diseño del sistema embebido el primer paso fundamental es identificar los componentes, las capacidades y características que los mismos debían poseer para poder integrarse y llevar adelante el objetivo de la predicción de humedad en el sustrato fenólico evaluando datos de variables físicas en tiempo real y llevando adelante una inferencia con una RNN LSTM para obtener la predicción de un dato futuro y que el mismo pueda servir para emitir una alarma en el caso que el valor predecido no se encuentre dentro de los valores ideales de germinación establecidos por los ingenieros agrónomos responsables del cuidado de los cultivos hidropónicos.

Componentes El sistema embebido de predicción de humedad de germinación debería tener la habilidad de:

- medir variables ambientales como la presión atmosférica, temperatura y humedad relativa ambiental,
- medir además la humedad relativa del sustrato de germinación,
- guardar los datos temporales de una ventana de medición que luego será el vector de ingreso al modelo de predicción,
- deberá poder realizar la inferencia con el modelo entrenado,
- comparar el valor predecido a futuro con valores definidos por los ingenieros agrónomos como humedad de germinación mínima y máxima
- la producción de alarmas ante el ingreso a zonas de valores no deseados para el proceso hidrópico relacionado.

Con dicha delimitación, los componentes mínimos para la integración en el sistema embebido deseado son: un Microcontrolador, un Sensor de variables ambientales, un Sensor de humedad de Esponja Fenólica, y un Sistema de alerta visual.

A continuación se desarrollará la investigación y ensayo de cada uno de ellos.

4.1.1 Microcontrolador

”El microcontrolador integra en un chip de microprocesador con memoria, interfaces de entrada/salida y otros dispositivos periféricos como temporizadores”[Bolton, 2013]

El factor mas importante de selección será la capacidad de poder correr una inferencia, lo cual significa poder correr en memoria una red pre-entrenada RNN/LSTM. Tomando como referencia las familias de microprocesadores que EdgeImpulse [EdgeImpulse®, 2024] posee como posibles dispositivos donde correr modelos de Inteligencia Artificial, entre los mas conocidos encontramos:

- Arduino Nano 33 BLE Sense (nRF52840 — Cortex-M4F 64MHz)
- Arduino Nicla Sense ME (nRF52832 — Cortex-M4 64MHz)
- Arduino Nicla Vision (STM32H747AI6 — Cortex-M7 480MHz)
- Arduino Portenta H7 (STM32H747XI — Cortex-M7 480MHz)
- Espressif ESP-EYE (ESP32 — Xtensa LX6 240MHz)
- Seeed XIAO ESP32 S3 Sense (ESP32S3 — Xtensa LX7 240MHz)
- Sony's Spresense (CXD5602 — Cortex-M4F 156MHz)

Haciendo foco en que el objetivo del proyecto se centra en poder generar un producto mínimo viable que pueda ser tenido como referencia para poder construir y utilizar por el mercado de productores de hortalizas hidropónicas, es fundamental poder elegir un MCU con costo relativamente bajo. Teniendo esto presente se selecciona la familia de microcontroladores ESP32 como base del proyecto. Dicha familia de MCU esta desarrollada por la empresa Espressif [Espressif®, 2024].

Espressif desarrolla varias familias de MCU, la Tabla 1 muestra la evaluación de las capacidades de las disponibles en Argentina.

| CHIP | ESP32 S3 | ESP32 C3 | ESP32 WROOM |
|--------------------------|-------------|----------|-------------|
| ARQUITECTURA | Xtensa LX7 | RISC-V | Xtensa LX6 |
| BUS | 32 Bits | 32 Bits | 32 Bits |
| SRAM on-chip | 512 KB | 400 KB | 520 KB |
| PSRAM external | hasta 16 MB | N/A | N/A |
| ROM on-chip | 384 KB | 384 KB | 448 KB |
| Frecuencia típica | 240MHz | 160MHz | 240MHz |
| Deep Sleep | 14uA | 44uA | 100 uA |

Tabla 1: Comparativa Microprocesadores

En pos de optimizar los tiempos de desarrollo y prototipado se analizan las opciones de placas de desarrollo ya existentes en el mercado que otorguen al proyecto simplificar

el diseño del circuito electrónica ya que las mismas integran junto al microcontrolador a los componentes mínimos necesarios para su funcionamiento.

Las tabla 2 muestra las opciones con mejores prestaciones accesibles en la actualidad :

| Placa de Desarrollo | Seeed Studio XIAO ESP32S3 | Seeed Studio XIAO ESP32C3 | ESP32-DEVKITM-1 |
|---------------------|---------------------------|---------------------------|-----------------|
| Fabricante | Seeed Studio | Seeed Studio | Spressif |
| Precio | USD 7.49 | USD 4.99 | USD 8.00 |
| Compra | Internacional | Internacional | Local |

Tabla 2: Comparativa Placas de Desarrollo

Analizando las características, tamaño y precio seleccionamos a la placa de desarrollo de la marca Seed Studio como las mas eficientes para el proyecto. Como características importantes se pueden destacar su bajo consumo que puede ser muy útil para el desarrollo de un sensor inteligente que sea energizado con una batería de Li-on, además de su tamaño reducido y de su bajo precio.

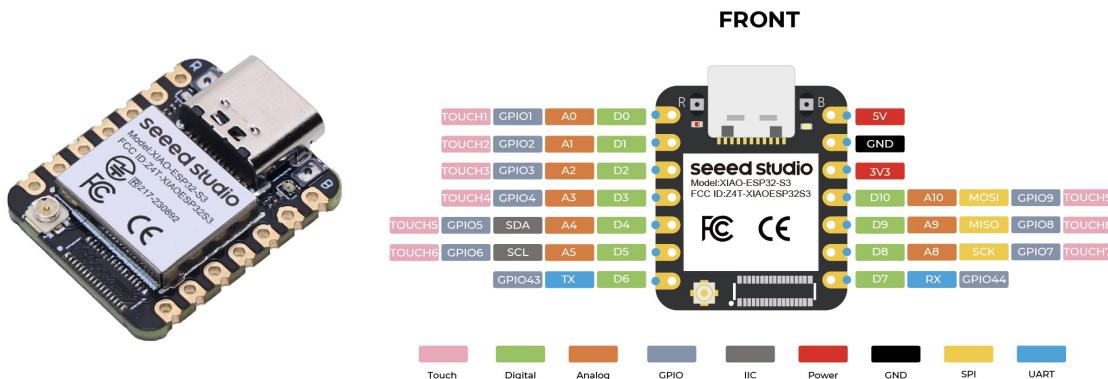


Figura 2: Seeed Studio XIAO ESP32S3

4.1.2 Sensor de Condiciones Ambientales

Como el objeto del dispositivo que se desea diseñar es predecir la humedad del sustrato de germinación, se presupone que dicha humedad dependerá del valor de otras co-variables asociadas presentes simultáneamente en el proceso de germinación.

De un análisis simple enumeramos las siguientes co-variables como primera instancia:

- Humedad de la esponja fenólica
- Humedad relativa ambiental
- Temperatura ambiental
- Presión Atmosférica
- Cantidad de cm3/m2 de riego

- Cantidad de agua que consume la semilla a lo largo de su germinación

Si consideramos que la variación en el tiempo de estas co-variables influyen en al variación de humedad que la esponja tendrá disponible para la germinación en el tiempo, es de vital importancia analizar cuales de ellas pueden ser adquiridas (medidas) e ingresadas al modelo LSTM para que el mismo pueda realizar una predicción con el menor error posible de la humedad futura de la variable principal, que como ya mencionamos es la humedad de la esponja fenólica.

En relación a la variable asociada a la cantidad de agua que consume la semilla a lo largo de su etapa de germinación inicial, la cual dependiendo de la especie de hortaliza puede variar entre 21 y 42 días, se descarta la posibilidad de que pueda desarrollarse un método para medir y adquirir la misma mediante un método simple y viable para la escala de inversión del proyecto, por lo cual se descarta.

La medición y adquisición de la humedad de la esponja fenólica de germinación se tratará en profundidad en la próxima sección.

Respecto al riego durante la germinación, se pudo relevar que el mismo se realiza de forma manual cuando el encargado de dicha función percibe que el color de la esponja empezó a cambiar aclarándose, lo que representaría que la cantidad de humedad de la misma ha disminuido. Como se puede identificar, éste es un criterio sumamente subjetivo ya que tanto la utilización de la humedad por parte de la semilla y la evaporación atmosférica que son las co-variables que influyen principalmente en la disponibilidad de agua (o sustrato líquido) que tendrá la esponja a disposición de las semillas no será un valor homogéneo en toda la superficie de germinación ya que dependerá tanto del tipo de semilla germinada, como de la posición de dentro de la zona de germinación, pudiendo notarse una mayor evaporación en los sectores de germinación de los extremos y pasillos , en relación que las zonas centrales. Esta característica deja en marcada evidencia la importancia de poder contar con un sensor que pueda alertar de que determinadas zona de germinación pueda necesitar riego con antelación a otras , transformando objetivamente el proceso de riego y logrado la mejora de la eficiencia en el proceso. En relación a la informalidad del proceso de riego, es inviable poder medir la cantidad de cm³/m² de riego y los momentos en el tiempo en los cuales se lo suministra para poder ser utilizado como una variable de entrada del modelo LSTM

En cuanto al resto de las co-variables, todas ellas ambientales, seleccionamos el sensor provisto por la empresa Bosh modelo BME280 por sus muy buenas prestaciones y bajo costo (aproximadamente USD 6.42) para la adquisición de las mismas. *Figura 3*

| Características Generales | |
|---------------------------|--------------|
| Voltaje de Funcionamiento | 1.71 a 3.6 V |
| Interfaz | I2C, SPI |
| Consumo de corriente | 3.6 µA |

Tabla 3: BME280 - Características Generales

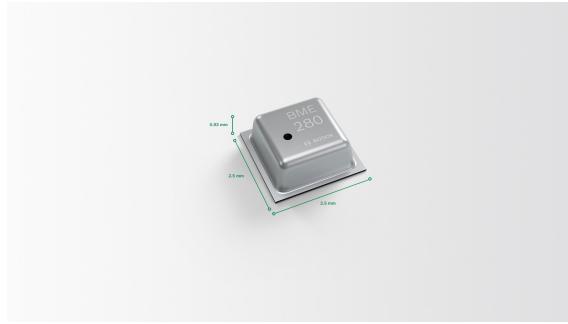


Figura 3: Sensor Bosh BME280

| | Sensor de Humedad | Sensor de Presión | Sensor de Temperatura |
|------------|-------------------|-------------------|-----------------------|
| Rango | 0 a 100 HR | 300 a 1100 hPa | -40 a 85 °C |
| Precisión | ± 3 HR | ± 1 hPa | ± 1 °C |
| Resolución | 0.008 HR | 0.18 Pa | 0.1 °C |

Tabla 4: BME280 - Rango/Precisión/Resolución

Dentro de sus características más importantes a destacar se encuentran: su bajo consumo, de alrededor de unos 3,6 μ A lo cual es ideal para aplicaciones alimentadas a través de baterías; su bus de comunicaciones I2C, el cual brinda eficiencia en la comunicación con el microcontrolador y requiere una muy baja cantidad de pines para ello, y su muy buena precisión en las tres variables que se desean utilizar como entradas del modelo: Temperatura , Humedad y Presión Atmosférica.

4.1.3 Espuma Fenólica

La espuma fenólica es un material sintético ampliamente utilizado en diversas aplicaciones industriales debido a sus propiedades únicas. En el contexto de la agricultura hidropónica, la espuma fenólica se destaca como un sustrato de germinación ideal debido a sus características físicas y químicas que favorecen el crecimiento de las plantas. Figura 4



Figura 4: Espuma Fenólica

4.1.3.1 Descripción del Material

La espuma fenólica es un tipo de plástico celular rígido, fabricado a partir de resinas fenólicas. Estas resinas son polímeros sintéticos derivados de la reacción de fenol con formaldehído. El resultado es un material ligero, con una estructura de celdas cerradas, que proporciona una excelente combinación de resistencia mecánica y estabilidad térmica.

4.1.3.2 Químicos Utilizados en su Fabricación

Los principales componentes químicos utilizados en la fabricación de la espuma fenólica incluyen:

- Fenol: Un compuesto aromático que actúa como uno de los reactivos base en la formación de resinas fenólicas.
- Formaldehído: Un aldehído simple que reacciona con el fenol en presencia de un catalizador ácido o básico para formar resinas fenólicas.
- Catalizadores: Ácidos o bases que aceleran la reacción entre el fenol y el formaldehído.
- Agentes Espumantes: Sustancias químicas que generan gas durante la reacción, creando la estructura celular de la espuma.

4.1.3.3 Proceso de Fabricación

El proceso de fabricación de la espuma fenólica generalmente sigue estos pasos:

- Reacción de Polimerización: El fenol y el formaldehído se mezclan en presencia de un catalizador para iniciar la reacción de polimerización, formando una resina líquida.
- Adición de Agentes Espumantes: Se añaden agentes espumantes a la resina líquida. Estos agentes generan gas, formando burbujas que crean la estructura celular de la espuma.
- Vertido en Moldes: La mezcla espumante se vierte en moldes donde se deja expandir y solidificar.
- Curado: La espuma se cura a altas temperaturas para completar la reacción de polimerización y estabilizar la estructura.
- Corte y Formado: La espuma curada se corta y forma en bloques o planchas según las especificaciones requeridas.

4.1.3.4 Características Ideales para la Germinación en Agricultura Hidropónica

La espuma fenólica posee varias características que la hacen ideal para la germinación de hortalizas en sistemas de agricultura hidropónica [Al-Malaika, 2005]:

- Retención de Humedad: La estructura celular de la espuma fenólica permite una excelente retención de agua, proporcionando un suministro constante de humedad a las raíces de las plantas.
- Aireación: A pesar de su capacidad de retención de agua, la espuma fenólica también permite una buena aireación, asegurando que las raíces reciban suficiente oxígeno.
- Estabilidad Química: La espuma fenólica es químicamente inerte, lo que significa que no reacciona con nutrientes o aditivos en la solución hidropónica, manteniendo un entorno de crecimiento estable.
- Estructura Consistente: La uniformidad de la estructura de la espuma fenólica asegura un desarrollo de raíces uniforme y evita la compactación que podría inhibir el crecimiento de las plantas.
- Ligereza y Manejo Fácil: Su baja densidad la hace fácil de manejar y manipular, facilitando el trasplante de plántulas sin dañar las raíces.

4.1.3.5 Otras Consideraciones

Además de las propiedades mencionadas, la espuma fenólica es resistente al moho y a la descomposición, lo que prolonga su vida útil en comparación con otros sustratos orgánicos. Sin embargo, su fabricación y uso deben manejarse adecuadamente para minimizar el impacto ambiental, ya que los componentes químicos involucrados pueden ser perjudiciales si no se gestionan correctamente.[Akelah, 2013]

4.1.3.6 Características técnicas

La figura 5 muestra un ejemplo de las características técnicas de las espumas fenólicas utilizadas en hidropónia, donde se puede destacar la capacidad de retener agua en relación a su propio volumen y el tiempo de retención del mismo como principales ventajas. [TEN-LEAD®, 2024]

4.1.4 Sensor de Humedad de Esponja Fenólica

Según lo arrojado por el estudio del estado de arte en relación a los sensores de medición de humedad en sustratos fenólicos de germinación, los sensores mas acordes son los Sensores de Capacitancia y los Sensores de Tensiometría. En pos de desarrollar la solución simple, económica y eficiente se selecciono a los Sensores de Capacitancia para su estudio y ensayo.

| Parámetro Del Producto | |
|---|---|
| Material | espuma fenólica |
| Densidad | 11-20kg/m ³ 40-50kg/m ³ |
| Espacio poroso | 90-98 por ciento |
| valor de pH | 6.5-7.4 |
| Conductividad eléctrica (ds/m, 20C grados) | 0.55 |
| Contracción | < 0.01% |
| Fuerza compresiva (kpa) | 20-31 |
| Agua totalmente disponible (porcentaje vol) | 50-61 por ciento |
| Tiempo de retención de agua(h) | 72-144 |
| Capacidad de retención de agua (porcentaje vol) | 50-91 por ciento |

Figura 5: Espuma Fenólica TEN-LEAD - Características técnicas

4.1.4.1 Contenido Volumétrico de Agua

De sus siglas en Inglés VWC (Volumetric Water Content) es el porcentaje de agua que hay en el suelo con respecto al volumen. Por ejemplo, si 1m³ está formado por 0,15m³ de aire, 0,35m³ de agua y 0,50m³ de minerales del suelo, el VMC será de 35%.

La VMC es un parámetro fundamental en la ciencia del suelo y la hidrología, ya que desempeña un papel crítico en la determinación de las propiedades físicas, químicas y biológicas del suelo y el movimiento del agua a través del perfil del suelo.

4.1.4.2 Principio de funcionamiento del Sensor de Capacitancia

Los sensores basados en la Técnica Capacitiva pertenecen a al grupo de los sensores basados en la Técnica de Reflectometría de Dominio de Frecuencia.

Para medir el VWC, estos sensores aprovechan el comportamiento atómico dentro de una molécula de agua al estar inmersa en un campo electromagnético. Una molécula de agua está compuesta por dos átomos de hidrógeno y uno de oxígeno, y los electrones de la molécula no están distribuidos uniformemente. El átomo de oxígeno tiene una electronegatividad mayor que los átomos de hidrógeno, lo que significa que atrae con más fuerza los electrones compartidos. Esto da lugar a una carga negativa parcial en el átomo de oxígeno y una carga positiva parcial en los átomos de hidrógeno, comportándose como un dipolo.

El principio de la Técnica de medición mediante la reflectometría del dominio de la frecuencia- FDR (Frequency-Domain Reflectometry) se basa en la integración de un circuito oscilante y un sensor capacitivo que está incrustado en el suelo [Caicedo-Rosero et al., 2021]

Esta técnica considera al suelo como el dieléctrico de un condensador, la constante dieléctrica del suelo varía en función de su contenido en agua, variando la capacitancia del condensador. Cuando este se condensador se integra en un circuito oscilador, las variaciones en la capacidad del condensador y la permitividad del medio conllevan a una variación en la frecuencia de la señal generada por el oscilador [Whalley et al., 1992],

[Robinson and Dean, 1993], [Gardner et al., 1998]. A partir de este principio se pueden mencionar dos tipos de técnicas bajo la denominación FDR: **los sensores capacitivos** (*en los cual se basa este trabajo final*) y los equipos FDR.

En los **sensores capacitivos** la permitividad del suelo, se determina midiendo el tiempo de carga del condensador, el cual se realiza mediante la salida del oscilador

Las sondas de estos equipos suelen presentarse en tres formas diferentes: placas planas, barras cilíndricas y anillos metálicos alrededor de un cilindro, Figura 6. En los sensores FDR con la configuración en anillo, la sonda se introduce dentro de un tubo de acceso ya instalado en el suelo para colocar varios sensores que puedan realizar medidas a diferentes profundidades. Avances tecnológicos en este tipo de sensores han servido para su miniaturización y/o análisis del VWC en el lugar de trabajo, además de mejores métodos de calibración con mayor exactitud y rapidez [Farahani et al., 2014]

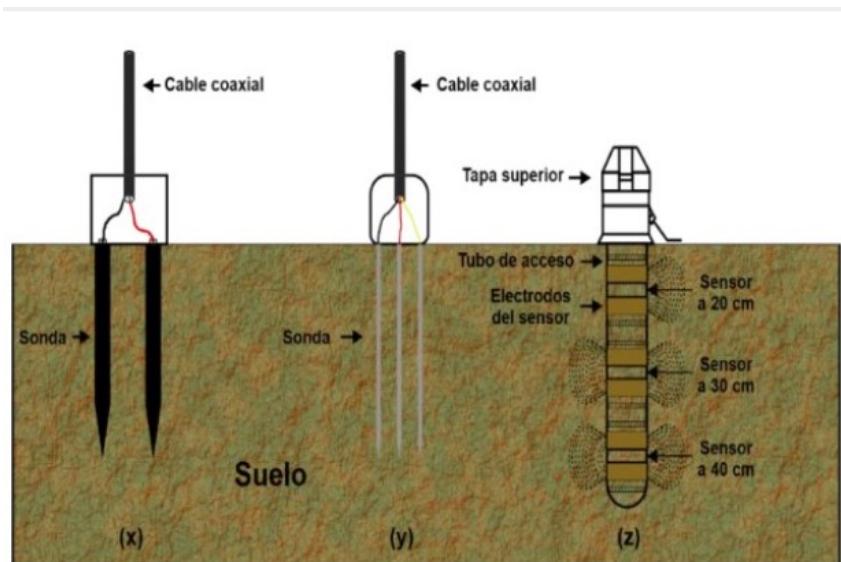


Figura 6: Tipos de sondas FDR. (x): de placas planas, (y) de barras cilíndricas, (z): de anillos metálicos alrededor de un cilindro [Caicedo-Rosero et al., 2021]

4.1.4.3 El sensor de Capacitancia

Se seleccionó el sensor de humedad de suelo del tipo capacitivo con sondas co-planares vendido para aplicaciones de IoT bajo la denominación "Capacitive Soil Moisture Sensor Module v2.0". Figura 7

Estos sensores utilizan un temporizador IC 555 y funcionan midiendo qué tan rápido (o lentamente) se carga un capacitor a través de una resistencia, pero en estos sensores el capacitor no es un componente literal, sino que está formado por dos trazas co-planares de PCB que están cerca una de la otra. Su capacitancia y, por lo tanto, su tasa de carga, cambian en respuesta a la permitividad del material que posean frente a ellas, en este proyecto específicamente cambiará según el VWC que posea el sustrato de germinación.

El sensor utiliza un 555 configurado como oscilador astable. Las ondas cuadradas

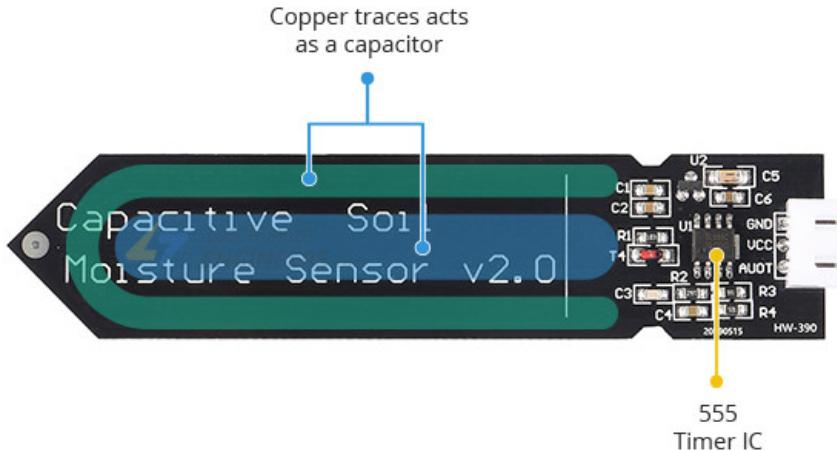


Figura 7: Sensor Capacitivo de Medición de Humedad de Suelo V2.0

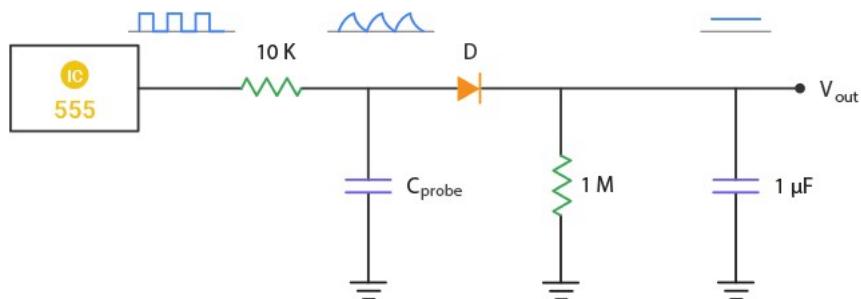


Figura 8: Diagrama esquemático Sensor Capacitivo de medición de Humedad de Suelo

generadas por el 555 se introducen en el integrador RC, a partir del cual se forma el condensador mediante la sonda de suelo. La señal del integrador es más bien una onda triangular, que se alimenta al rectificador y a un condensador de suavizado para producir una salida de CC. La Figura 8 muestra esquemáticamente el funcionamiento descrito.

Por lo cual la salida de la señal acondicionada del sensor V_{out} es proporcional al contenido de humedad del suelo VWC; si el sustrato está seco, el condensador se carga rápidamente, lo que da como resultado una mayor amplitud de la onda triangular y, posteriormente, produce un voltaje de salida más alto. Por el contrario, cuando el sustrato está mojado, el condensador se carga más lentamente, lo que da como resultado una amplitud más pequeña de la onda triangular, que a su vez genera un voltaje de salida más bajo.

El precio de este sensor comercial internacionalmente ronda entre los USD 0,79 - 1.10 en tiendas como AliExpress.

4.1.4.4 *Modelo de regresión*

El ultimo paso consiste en obtener el modelo analítico entre el valor de humedad de suelo VWC [%] y la salida de voltaje del sensor [Voltios].

Para ello se realizó el siguiente ensayo:

1. se imprimió una pequeña carcasa para la protección de los circuitos electrónicos del sensor y se instaló, Figura 9,

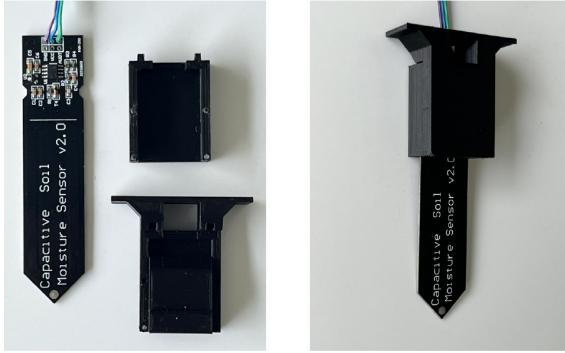


Figura 9: Ensamble de Sensor con carcasa de protección

2. se realizó el conexionado necesario para alimentar al sensor con 3.3V,
3. se conectó en la salida analógica del sensor un multímetro digital en modo voltímetro.,
4. se eligió una esponja de germinación estandar utilizada en una empresa local, se midió su volumen, y se midió su masa mediante una balanza de precisión +/-0,01g.
5. se saturó de solución de hidratación hidropónica al volumen elegido de esponja para el ensayo logrando su máxima retención de liquido, se midió nuevamente su masa obteniendo el valor de **87.58 gramos netos**, el cual se tomó como correspondiente al **99%** de humedad de retención de la esponja.
6. se colocó la esponja sobre el sensor longitudinalmente, en lugar de perpendicularmente como se utiliza habitualmente en suelos minerales para mayor inferencia de la humedad del sustrato en el sensor, para una mayor sensibilidad y de esta manera se asegura que la superficie del sensor en contacto con la esponja fenólica a ensayar sea siempre la misma (esponja haciendo tope en la carcasa plástica del sensor), Figura 10



Figura 10: Utilización del Sensor Capacitivo en sustratos

7. se colocó el sistema en el aire libre, bajo una galería, simulando aproximadamente la velocidad de evaporación que sucede en ambientes de producción

8. se midió periódicamente la variación de masa de la esponja [gramos] y el voltaje entregado por el sensor [voltios], Figura 11

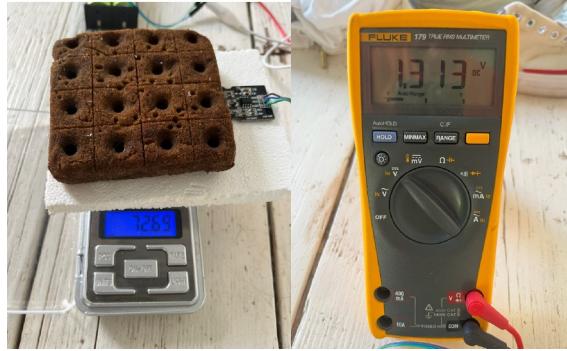


Figura 11: Ensayo del Sensor Capacitivo

9. se llevaron los valores a la tabla de referencia, Tabla 5,

| | | |
|---|-------------|-----------------|
| Masa de la esponja | 1.89 | g |
| Dimensiones esponja | 1,9x7,8x7,8 | cm |
| Volumen esponja | 115,6 | cm ³ |
| Capacidad de retención de líquido esponja | 79,29 | g |
| Capacidad de retención de líquido | 68,6 | % |
| Masa bandeja | 2,54 | g |
| Masa sonda | 27,42 | g |

| Salida Sonda [V] | Masa medida [g] | Masa neta de agua [g] | Retención de agua en la esponja [%] |
|------------------|-----------------|-----------------------|-------------------------------------|
| 0.880 | 119,43 | 87,58 | 99,00% |
| 0.886 | 117,05 | 85,2 | 96,31% |
| 0.904 | 113,86 | 82,01 | 92,70% |
| 0.986 | 107,75 | 75,9 | 85,80% |
| 0.997 | 104,9 | 73,05 | 82,58% |
| 1,027 | 103,85 | 72 | 81,39% |
| 1,058 | 102,01 | 70,16 | 79,31% |
| 1,103 | 98,6 | 66,75 | 75,45% |
| 1,142 | 91,49 | 59,64 | 67,42% |
| 1,17 | 87,2 | 55,35 | 62,57% |

Tabla 5: Ensayo Esponja fenólica [Vout vs. HR]

10. con un software desarrollado en Python (figura 12), se logra graficar los valores medidos, se busca la regresión lineal entre los valores y se comprueba la linealidad y las variables de identidad de dicha recta, obteniendo la ecuación de regresión entre la salida del sensor y la humedad relativa de la esponja fenólica. los resultados son mostrados en la Figura 13

Con un RMSE obtenido de 1.80 podemos tomar como válida a la recta de regresión obtenida, definiendo sus constantes: $b = 200.3753953040497$ y $m = -116.04687806958512$. Con dichos valores podremos obtener los valores de humedad relativa de la esponja de germinación desde el valor en voltios entregado por el sensor capacitivo.

Por el principio de funcionamiento y simpleza del sensor, si los mismos son construidos con estándares de calidad y con componentes de tolerancia 1%, se debería calcular el RMSE entre diferentes sensores del tipo en cuestión, esto determinará si se podrá usar la misma curva de regresión o si se debería realizar algún método de calibración para cada sensor.

```
# ENSAYO CON SONDA CAPACITIVA
import numpy as np
import matplotlib.pyplot as plt
from sklearn.metrics import mean_squared_error
from math import sqrt

# creating data
conductividad = np.array([0.880, 0.886, 0.904, 0.986, 0.997, 1.027, 1.058, 1.103, 1.142, 1.170])
HR = np.array([99, 99.31, 92.70, 85.80, 82.58, 81.39, 79.31, 75.45, 67.42, 62.57])

HR_predict = np.zeros(len(HR))

# create a simple scatterplot
plt.plot(conductividad, HR, "o")

# obtain the m (slope) and b(intercept) of the linear regression line
m, b = np.polyfit(conductividad, HR, 1)
print("Parameters of the regression line: \n")
print("m= ", m)
print("b= ", b)

# add a linear regression line to the scatterplot
plt.plot(conductividad, m * conductividad + b)

# calculate HR predict for values of array "conductividad"
for i in range(len(HR)):
    HR_predict[i] = (m * conductividad[i]) + b

# RMSE = sqrt(mean_squared_error(HR, HR_predict))
print("RMSE = ", RMSE)
```

Figura 12: Código Python, análisis regresión del sensor capacitivo

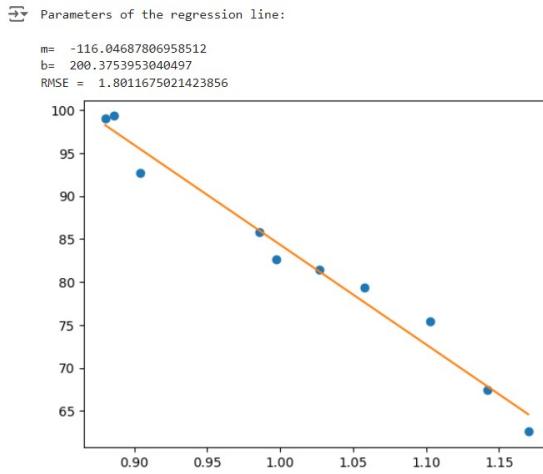


Figura 13: Resultados Regresión lineal Sensor Capacitivo

4.1.5 Integración del Sistema Embebido

El último paso contempló la integración de los componentes del sistema embebido, el cual en resumen poseerá:

- placa de desarrollo SEEED STUDIO XIAO ESP32S3
- sensor de condiciones ambientales BME280
- sensor de humedad de esponja fenólica
- LEDs de señalización

La figura 14 muestra el diagrama esquemático del sistema embebido
Las funciones del sistema embebido serán:

- la lectura de las variables de los sensores cada 10 minutos
- la normalización de los valores medidos

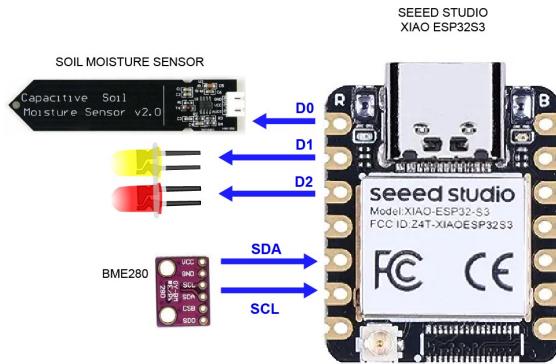


Figura 14: Esquemático Sistema Embebido

- almacenar las últimas 24 horas de datos sensados
- prepara la matriz de entrada al modelo de predicción previamente entrenado
- realizar la inferencia con los datos suministrados, y obtener la predicción de la variable "Humedad de sustrato" 30 minutos en el futuro.
- re-escalar la predicción para llevarlos a unidades originales
- comparar el valor predecido futuro con límites de alarmas, si el valor predecido esta por debajo del 50%HR de la esponja se encenderá el LED ROJO, y si están entre 50 y 75%HR encenderá el LED AMARILLO
- en el reinicio del sistema embebido AMBOS LED estarán encendidos. hasta que la matriz de entrada de datos del modelo aun no posea los 576 valores para poder realizar una inferencia.

La figura 15 muestra el diagrama de bloques y flujo objetivo del Sistema Embebido

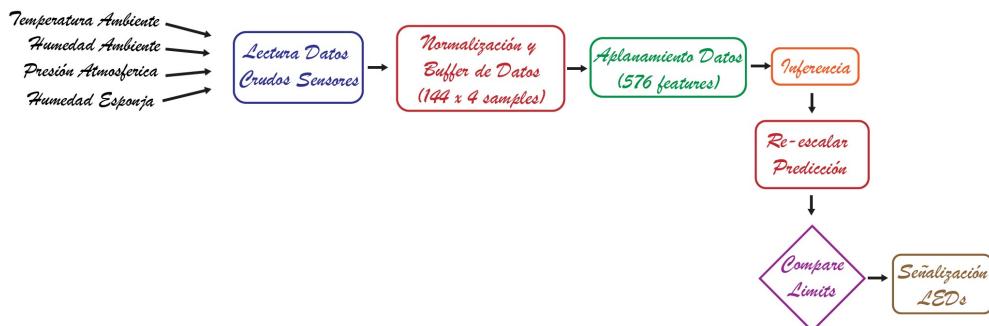


Figura 15: Diagrama Bloques Sistema Embebido

4.2 Obtención del Dataset

El dataset es un componente crucial en el desarrollo de cualquier modelo de aprendizaje automático, y su importancia se magnifica en el contexto del entrenamiento, evaluación y prueba de una red neuronal LSTM (Long Short-Term Memory). En este trabajo, el dataset constituye la base de datos sobre la cual se entrenará el modelo para predecir la humedad relativa de la esponja fenólica en un entorno de germinación hidropónica.

4.2.1 Concepto e Importancia del Dataset

El dataset es una colección estructurada de datos que se utiliza para alimentar al modelo de aprendizaje automático. En el caso de una red neuronal LSTM, que está diseñada para trabajar con datos secuenciales y dependencias temporales, la calidad y estructura del dataset son esenciales. La precisión y la capacidad predictiva del modelo dependen directamente de la calidad de los datos proporcionados durante las fases de entrenamiento, evaluación y prueba.

La red LSTM se entrena utilizando estos datos para aprender patrones y relaciones temporales entre las variables de entrada y la variable objetivo. Una vez entrenado, el modelo se evalúa y prueba utilizando segmentos del dataset que no se utilizaron durante el entrenamiento. Esto permite verificar su capacidad para generalizar y predecir correctamente sobre datos no vistos anteriormente.

4.2.2 Generación del Dataset

Es importante mencionar que el dataset específico requerido para este proyecto no se encontró en ninguna de las bases de datos internacionales de datasets para el entrenamiento de RNN, tales como Kaggle, UCI Machine Learning Repository y Google Dataset Search. Esta falta de datos disponibles resalta la necesidad de generar un nuevo dataset que cumpla con los requisitos específicos del proyecto. La generación de este dataset implica la recolección y registro meticuloso de datos en tiempo real de las variables relevantes.

4.2.3 Estructura del Dataset y Variables Incluidas

El dataset utilizado en este trabajo contiene series temporales de varias variables clave que afectan la humedad relativa de la esponja fenólica. Las variables incluidas en el dataset son:

- **Sello de Tiempo (Timestamp):** Esta variable marca el tiempo exacto en el que se registraron las demás mediciones. Es fundamental mantener la periodicidad y la secuencia temporal para que la red LSTM pueda captar adecuadamente las relaciones temporales entre las variables.

- **Humedad Relativa de la Esponja Fenólica:** Esta es la variable objetivo que el modelo intentará predecir. Representa el contenido de humedad presente en la esponja fenólica en un momento dado.
- **Temperatura Ambiente:** La temperatura del entorno en el cual se encuentra la esponja fenólica. Esta variable puede influir significativamente en la evaporación y, por ende, en la humedad de la esponja.
- **Humedad Relativa del Ambiente:** La cantidad de humedad presente en el aire circundante. Esta variable también puede afectar el intercambio de humedad entre la esponja y el ambiente.
- **Presión Atmosférica:** La presión del aire en el entorno. Aunque su impacto puede ser menos directo, la presión atmosférica puede influir en la evaporación y otros procesos físicos que afectan la humedad de la esponja.

4.2.4 Importancia de Respetar la Periodicidad

Para que la red LSTM pueda aprender correctamente de los datos, es crucial respetar la periodicidad y el orden temporal de las observaciones. Las LSTM están diseñadas para manejar dependencias a largo plazo en datos secuenciales, lo que les permite captar patrones temporales complejos que serían difíciles de modelar con otros tipos de redes neuronales.

La periodicidad en el dataset asegura que las secuencias temporales se mantengan intactas, permitiendo al modelo entender cómo evolucionan las variables a lo largo del tiempo. Cualquier interrupción o desorden en los datos temporales podría comprometer la capacidad del modelo para aprender y predecir con precisión.

En resumen, la construcción y organización del dataset son pasos fundamentales en el desarrollo de este proyecto. Un dataset bien estructurado no solo mejora el rendimiento del modelo, sino que también garantiza que las predicciones sean precisas y confiables, proporcionando una herramienta valiosa para la gestión de la humedad en sistemas de germinación hidropónica.

4.2.5 El Datalogger

El datalogger es un dispositivo esencial en este proyecto, diseñado para recolectar y almacenar datos de diversas variables ambientales y del sustrato fenólico, necesarios para entrenar y evaluar la red neuronal LSTM (Long Short-Term Memory). En esta sección, se describirá el concepto del datalogger, su diseño y los componentes utilizados, así como los desafíos encontrados durante la recolección de datos.

4.2.5.1 Concepto

Un datalogger es un dispositivo que registra datos en el tiempo o en relación con la ubicación, utilizando un sensor o instrumento. En este proyecto, el datalogger se encarga de medir y registrar variables ambientales críticas y la humedad del sustrato fenólico de germinación. Estos datos son fundamentales para el entrenamiento, validación y prueba del modelo de red neuronal LSTM.

4.2.5.2 Diseño del Datalogger

El diseño del datalogger incluye varios componentes electrónicos interconectados que permiten la adquisición precisa y continua de datos. Los componentes principales son:

- **Microcontrolador:** Por practicidad se utilizó una placa de desarrollo con el microcontrolador ESP32-WROOM-32, elegido por su capacidad de procesamiento,, y su eficiencia energética. El microcontrolador actúa como el cerebro del datalogger, controlando todos los demás componentes, procesando y almacenando los datos recogidos.
- **Reloj en Tiempo Real (RTC) DS3231:** Un módulo RTC se utiliza para proporcionar un sello de tiempo preciso para cada lectura de datos. El DS3231 es conocido por su alta precisión y estabilidad, lo que asegura que cada muestra tenga una marca temporal exacta.
- **Sensor de Variables Ambientales BME280:** Este sensor es capaz de medir la temperatura, la humedad relativa y la presión atmosférica. El BME280 es muy preciso y tiene un bajo consumo de energía, ideal para aplicaciones de monitoreo ambiental.
- **Sensor de Medición de Humedad de Suelo V2.0:** Este sensor mide la humedad del sustrato fenólico. Es confiable y preciso, proporcionando datos esenciales sobre el estado de humedad del sustrato de germinación.
- **Tarjeta de Memoria SD:** Se utiliza para almacenar los datos leídos por el datalogger. La tarjeta SD permite el almacenamiento de grandes cantidades de datos y facilita la transferencia de estos a un ordenador para su posterior análisis.
- **Pantalla LCD:** Un display LCD se incluye en el diseño para mostrar en tiempo real las lecturas de los sensores. Esto permite verificar y testear que el sistema funcione correctamente durante la fase de implementación.

4.2.5.3 Periodo de Recolección de Datos

El periodo de recolección de datos fue de 66 días. Durante este tiempo, el datalogger sufrió un daño en su alimentación eléctrica, lo cual limitó la cantidad de datos que se pudo

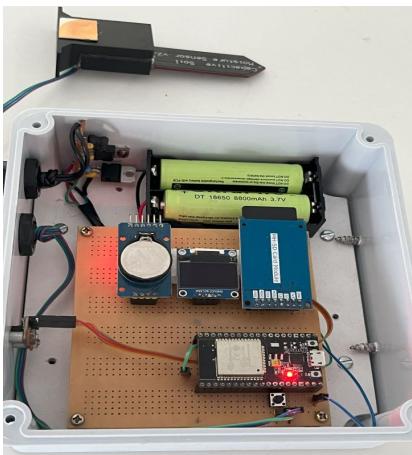
registrar. Originalmente, se planeaba recolectar una mayor cantidad de muestras, pero debido a este contratiempo el período efectivo de recolección se vio reducido logrando recolectar un total de 6474 muestras consecutivas (04/05/2024 al 18/06/2024) que serán utilizadas para el entrenamiento, validación y test del modelo. Se utilizó la misma tasa de muestreo que se estableció para el sistema embebido desarrollado, desarrollado en el punto 4.1.5

4.2.5.4 Consideraciones Adicionales

El diseño y la implementación del datalogger no solo se enfocaron en la precisión y la fiabilidad de los datos recolectados, sino también en la durabilidad y la resistencia del dispositivo en condiciones de campo. La elección de componentes como el ESP32 y el DS3231 asegura que el datalogger pueda operar de manera continua y precisa, mientras que el BME280 y el sensor de humedad de suelo garantizan que se recolecten datos ambientales relevantes y críticos para el modelo de predicción.

Además, se consideró la facilidad de acceso a los datos almacenados. El uso de una tarjeta SD permite que los datos recolectados se puedan extraer fácilmente y analizar sin interrupciones prolongadas del funcionamiento del datalogger.

En resumen, el datalogger fue diseñado para proporcionar datos de alta calidad y confiabilidad necesarios para entrenar y evaluar la red neuronal LSTM, y aunque se enfrentaron algunos desafíos durante la recolección de datos, se lograron obtener suficientes muestras para cumplir con los objetivos del proyecto.



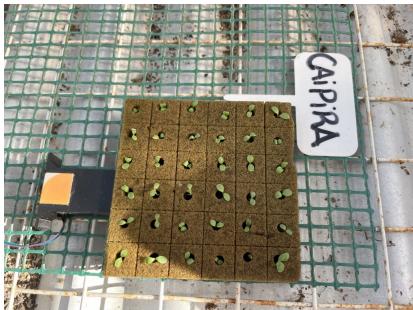
(a) Ensamblado



(b) Instalación

4.2.5.5 Funcionamiento

El Datalogger posee una doble fuente de alimentación: como alimentación principal posee una fuente regulada de 220VAC a 12VCC conectada a la red eléctrica, regulados a 9VCC gracias a un regulador 7809, y como fuente de contingencia baterías de Li-ion tipo 18650 con capacidad de 2500mAh, brindando una autonomía de aproximadamente



(a) Datalogging día 1



(b) Datalogging día 21

Figura 17: Datalogger

50 horas.

El firmware del ESP32 lleva adelante el sensado, y almacenamiento cada 10 minutos; para mayor precisión los períodos de tiempo son medidos con el reloj en tiempo real (RTC) DS3231. Los valores medidos son convertidos a caracteres y guardados en un archivo tipo TXT, separados por comas para una importación.

El almacenamiento se realiza en archivos diferentes cada día con el formato 20240201 (AñoMesDía), si el firmware detecta que el archivo no existe, lo genera y luego graba las variables medidas; si ya existe solo agrega las mediciones físicas. Cada grabación está acompañada del correspondiente TS (Timestamp o Sello de tiempo) en el cual fueron medidas. La figura 18 muestra el formato final de un archivo log generado.

```
date_time,tempC,HR,pressure,HRsoil,Vsensor
12/04/2024 16:32:00, 16.61, 89.40, 1016.31, 86, 0.982
12/04/2024 16:42:00, 16.61, 89.39, 1016.28, 86, 0.982
12/04/2024 16:52:00, 16.84, 88.42, 1016.14, 87, 0.981
12/04/2024 17:02:00, 17.00, 87.43, 1016.19, 88, 0.971
12/04/2024 17:12:00, 17.41, 86.36, 1016.25, 86, 0.982
```

Figura 18: Datalog - Variables grabadas

4.2.6 Pre-procesamiento del Dataset

Para la conformación y preparación del dataset se utiliza Python. En primer instancia, todos los archivos generados por el Datalogger son guardados dentro de una carpeta denominada ”/datalogs”. El primer paso es importar todos los archivos guardados con información relevada

```
extension = 'txt'
todos_los_archivos = [i for i in glob.glob('*.{}'.format(extension))]
```

Luego se combinan todos los archivos de la lista y se los concatena en un dataframe ”df”. Y como complemento se guarda una copia del mismo en formato ”.CSV” (Comma Separated Values)

Obteniendo el siguiente formato de dataframe:

```
df = pd.concat([pd.read_csv(f) for f in todos_los_archivos])

os.chdir(CURRENT_DIR)
df.to_csv("combinado_csv.csv", index=False, encoding='utf-8-sig')
```

| | date_time | tempC | HR | pressure | HRsoil | Vsensor |
|-----|---------------------|-------|-------|----------|--------|---------|
| 0 | 12/04/2024 16:32:00 | 16.61 | 89.40 | 1016.31 | 86 | 0.982 |
| 1 | 12/04/2024 16:42:00 | 16.61 | 89.39 | 1016.28 | 86 | 0.982 |
| 2 | 12/04/2024 16:52:00 | 16.84 | 88.42 | 1016.14 | 87 | 0.981 |
| 3 | 12/04/2024 17:02:00 | 17.00 | 87.43 | 1016.19 | 88 | 0.971 |
| 4 | 12/04/2024 17:12:00 | 17.41 | 86.36 | 1016.25 | 86 | 0.982 |
| ... | ... | ... | ... | ... | ... | ... |

Convierto la columna "date_time" a una nueva columna con formato y nombre "*datetime*" para poder usar las funciones del tiempo de la librería "*panda*" de Python

```
: df['datetime'] = pd.to_datetime(df['date_time'],format = '%d/%m/%Y %H:%M:%S')
```

A continuación se fija a la columna "*datetime*" como índice y se ordena cronológicamente el dataframe.

```
df = df.set_index('datetime')
df.sort_index(inplace=True)
```

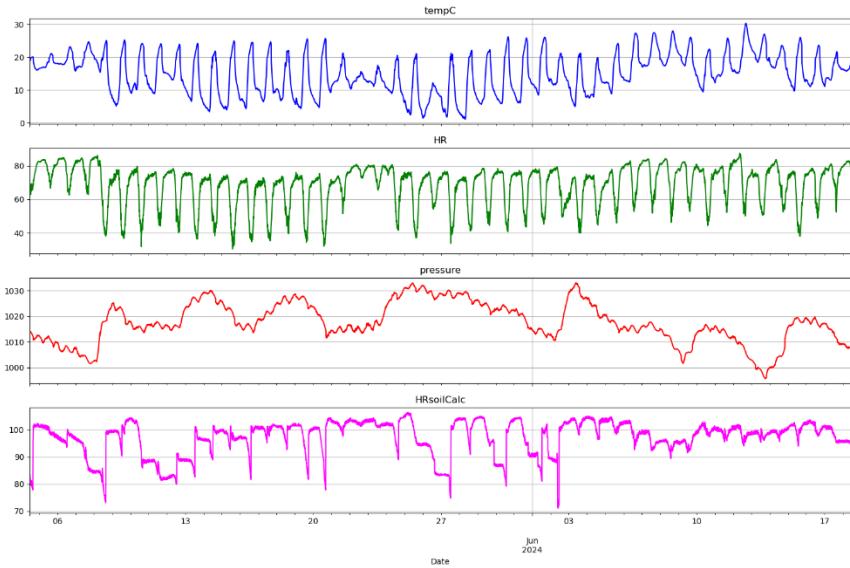
Las mediciones de humedad de la esponja fenólica de germinación fueron agregados a los archivos de datalog como números enteros, pero los mismos fueron calculados por el firmware del datalogger a través de la regresión hallada entre la tensión de salida del sensor y la humedad relativa de la esponja. A continuación se genera una nueva columna para el dataset denominada '*HRsoilCalc*' con los valores de humedad calculados a través de la regresión pero con precisión flotante para velar por una mejor precisión en el entrenamiento de la red LSTM

```
m = -116.0468780696
b = 200.375395304049
df['HRsoilCalc'] = (df['Vsensor']* m) + b
df = df[['date_time','tempC','HR','pressure','HRsoil','HRsoilCalc','Vsensor']]
```

Paso siguiente se borra del dataframe las columnas originales '*date_time*', '*HRsoil*', '*Vsensor*', las primeras dos serán remplazadas por las columnas '*datetime*' y '*HRsoilCalc*' respectivamente, mientras que '*Vsensor*' no será una variable de entrada al modelo.

```
df = df.drop(columns=['date_time', 'HRsoil', 'Vsensor'])
#df.columns
```

Para visibilizar la integridad del dataset, se grafica cada co-variable con colores distintos compartiendo la variable de tiempo en eje x. De esta manera se puede tener una visión general del grupo de datos obtenidos.



A continuación se realiza el análisis de datos faltantes. Los datos faltantes serán simplemente aquellas celdas dentro del dataframe que no contienen cantidades numéricas. Usualmente estas celdas contendrán valores tipo NaN (Not A Number).

```
print('Cantidad de NaNs:')
for column in df:
    nans = df[column].isna().sum()
    print(f'\tColumna {column}: {nans}')
```

El siguiente paso es Análisis de la periodicidad del dataset. Se debe verificar que entre muestras consecutivas del set de datos existe una diferencia temporal de exactamente 10 minutos (600 segundos), que es el período de muestreo del datalogger. Esta verificación es clave para garantizar que el modelo LSTM que implementaremos más adelante aprenderá a detectar adecuadamente los patrones en la serie de tiempo.

```
df_time_diffs = df.index.to_series().diff().dt.total_seconds()
print(df_time_diffs.value_counts())
```

4.2.6.1 Preparación los datos para el entrenamiento del modelo LSTM

El primer paso es realizar una función para separar nuestro dataset en entrenamiento, validación y prueba, aproximadamente tomamos un 80% para entrenamiento, un 10% para validación de la red neuronal, y un 10% para la prueba del modelo LSTM generado.

Ejecutamos la función y realizamos la generación de los tres sub datasets

Verificamos la continuidad de los datos en el tiempo de cada sub datasets

4.2.6.2 Escalamiento y Normalización del Dataset

El escalamiento y la normalización de datos son pasos fundamentales en el preprocesamiento de un dataset antes de entrenar una red neuronal. Estas técnicas transforman

```
def train_val_test_split(dataframe, tr_size=0.8, vl_size=0.1, ts_size=0.1):

    # Definir número de datos en cada subserie
    N = dataframe.shape[0]
    Nval = 1133    # Número de datos de validación
    Ntst = 1133     # Número de datos de prueba
    Ntrain = N - Ntst - Nval # Número de datos de entrenamiento

    # Realizar partición
    test = dataframe[0:Ntst]
    val = dataframe[Ntst:Nval+Ntst]
    train = dataframe[Nval+Ntst:]

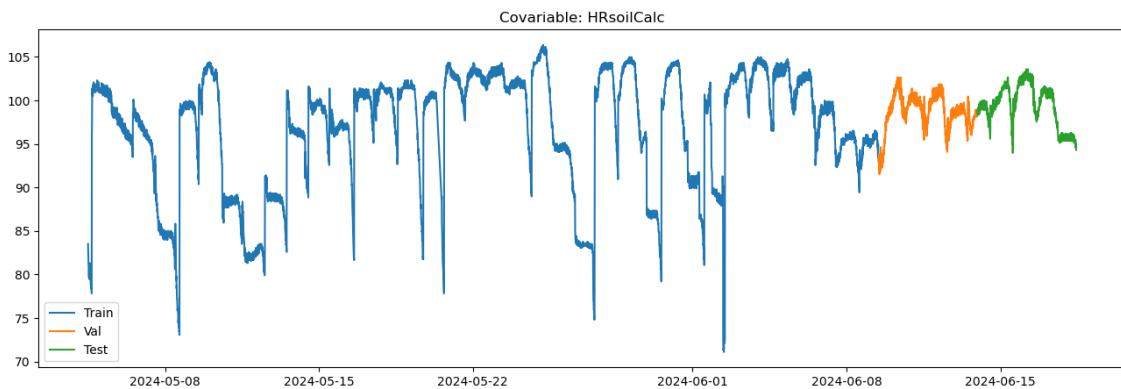
    return train, val, test
```

```
data_tr, data_vl, data_ts = train_val_test_split(df)

print(f'Tamaño set de entrenamiento: {data_tr.shape}')
print(f'Tamaño set de validación: {data_vl.shape}')
print(f'Tamaño set de prueba: {data_ts.shape}')
```

```
covar = 3                      # Índice de la covariante a graficar
col = df.columns[covar]

# Dibujar los sets de entrenamiento/validación/prueba para la covariante
fig, ax = plt.subplots(figsize = (16,5))
ax.plot(data_tr[col], label='Train')
ax.plot(data_vl[col], label='Val')
ax.plot(data_ts[col], label='Test')
ax.set_title(f'Covariante: {col}')
plt.legend();
```



los datos en un rango específico, mejorando la eficiencia del entrenamiento del modelo y asegurando que todas las características contribuyan de manera equitativa al proceso de aprendizaje. En este proyecto, se utiliza la función `MinMaxScaler` de la librería `sklearn` para escalar y normalizar los datos.

El escalamiento y la normalización mejoran el desempeño de las redes neuronales de varias maneras:

- **Convergencia Rápida:** Al escalar los datos, el proceso de optimización converge más rápidamente. Esto se debe a que los gradientes descendentes se estabilizan y permiten que los pesos de la red se actualicen de manera más uniforme.
- **Mejora en la Precisión:** Normalizar los datos reduce la posibilidad de que características específicas dominen el aprendizaje, permitiendo que el modelo preste

atención a todas las características de manera equitativa. Esto puede llevar a un mejor desempeño y precisión del modelo.

- **Consistencia de Datos:** Al mantener las características dentro del mismo rango, se asegura que los cálculos internos de la red, como los productos escalares, sean consistentes y estables.

Para llevar a cabo el escalamiento y normalización del dataset en este proyecto, se emplea la función `MinMaxScaler` de la librería `sklearn`. Esta función transforma los datos para que todos los valores de las características estén dentro de un rango específico, generalmente entre 0 y 1. La fórmula utilizada por `MinMaxScaler` es:

$$X_{\text{scaled}} = \frac{X - X_{\min}}{X_{\max} - X_{\min}}$$

donde X es el valor original de la característica, X_{\min} y X_{\max} son los valores mínimos y máximos de esa característica en el dataset, respectivamente. Este proceso asegura que todas las características contribuyan de manera equilibrada al entrenamiento del modelo.

```
scaler = MinMaxScaler(feature_range=(0, 1))
data_tr_n = scaler.fit_transform(data_tr)
data_vl_n = scaler.transform(data_vl)
data_ts_n = scaler.transform(data_ts)
```

La información de los escaladores de cada co-variable son fundamentales para la implementación del sistema embebido final, ya que los datos adquiridos en vivo por el sistema deben ser normalizados y escalados antes de ser ingresados al modelo, y luego la predicción realizada por el modelo deberá ser invertida su escala para poder ser utilizada para generar las alertas deseadas. A tal motivo estos valores son guardados en un archivo tipo texto para luego poder ser utilizados

```
os.chdir("./")
with open('scaler_params.txt', 'w') as file:
    file.write('scale:' + ','.join(map(str, scaler.scale_)) + '\n')
    file.write('min:' + ','.join(map(str, scaler.min_)) + '\n')
    file.write('data_min:' + ','.join(map(str, scaler.data_min_)) + '\n')
    file.write('data_max:' + ','.join(map(str, scaler.data_max_)) + '\n')
    file.write('data_range:' + ','.join(map(str, scaler.data_range_)) + '\n')
```

4.3 Implementación Red neuronal LSTM

4.3.1 Redes Long Short-Term Memory (LSTM)

Las redes LSTM como red neuronal recurrente (RNN), están diseñadas para manejar como entrada series de datos temporales de un grupo de variables simultáneamente, lo que nos se ajusta especialmente para al desarrollo de este proyecto donde se tienen varias

variable físicas como dato para el desarrollo de un modelo predictivo integral, el cual aprenderá de las intrincada relación entre ellas y su efecto combinado sobre la variable que se desea predecir. Incorporar redes LSTM dentro de proyectos de TinyML (Tiny Machine Learning) es bastante poderoso ya que permitirá el análisis de datos secuenciales y la generación de una predicción directamente en un microcontrolador.

Los modelos LSTM pueden ser configurados para diferentes tipos de tareas como se muestra en la figura 19 [Biases®, 2024] :

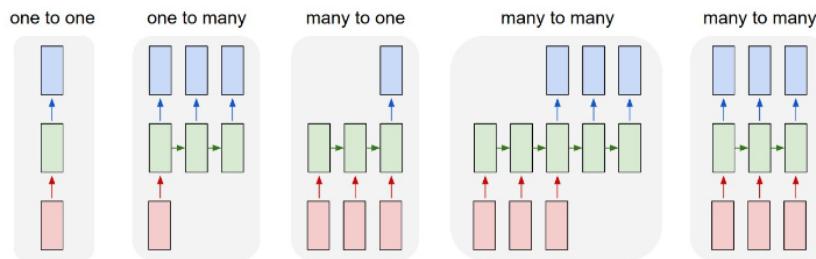


Figura 19: Tipos de Redes LSTM

1. **One-toOne (uno a uno):** para tareas en las que hay una única entrada y una única salida, a menudo se utiliza para tareas estándar de clasificación o regresión.
2. **One-to-Many (uno a muchos):** para tareas que requieren una entrada para generar una secuencia de salidas, como subtítulos de imágenes, donde una entrada de imagen genera una secuencia de palabras como salida.
3. **Many-to-One (muchos a uno):** se utiliza para tareas en las que una secuencia de puntos de datos conduce a una única salida. Es particularmente adecuado para el análisis de sentimiento o, como en su caso, el pronóstico del tiempo, donde una serie de observaciones a lo largo del tiempo conduce a un único valor pronosticado. La estructura "muchos a uno" permite al modelo observar una secuencia de puntos de datos, procesar las relaciones temporales dentro de esa secuencia y condensar toda esa información en una única salida predictiva, que sería la temperatura prevista en un momento futuro. punto.
4. **Many-to-many (muchos a muchos):** para tareas en las que una secuencia de entrada se asigna a una secuencia de salida, que puede sincronizarse o tener diferentes longitudes, como la traducción automática o el reconocimiento de voz.

En nuestro proyecto que utiliza un conjunto de parámetros relacionados con variables físicas del ámbito de germinación, al modelo LSTM "muchos a uno" es el que se ajusta naturalmente ya que se posee el objetivo de predecir un único valor de humedad de esponja fenólica futura a partir de una secuencia de observaciones pasadas. El modelo necesita comprender y recordar patrones en la secuencia de entrada (como ciclos

diarios o estacionales) para predecir el siguiente paso con precisión, esto es precisamente la función de los tipo de redes LSTM "Many to One".

4.3.2 Preparación de los datos para la predicción LSTM

Durante la preparación de datos para un modelo LSTM, es necesario transformar el conjunto de datos de series de tiempo en secuencias que el modelo puede utilizar para aprender los patrones que luego utilizará para realizar la predicción. Dado que los modelos LSTM están diseñados para trabajar con datos secuencias, necesitamos reformatear nuestro conjunto de datos en una estructura de entrada-salida donde la entrada es una secuencia de datos de las covariables que corresponden a un momento en el tiempo determinado, y la salida es el valor en instante de tiempo que se desea predecir. En este proyecto, definimos una ventana de 576 (24 x 144) pasos de datos en el tiempo para predecir la humedad de la esponja de germinación 30 minutos en el futuro (3 steps de tiempo), recordando que el período de muestreo de datos es de 10 min. Esto significa que utilizamos los datos de las últimas 24 horas (que equivale a una semana de datos) para predecir la humedad relativa de la espuma fenólica a la siguiente media hora. Figura 20

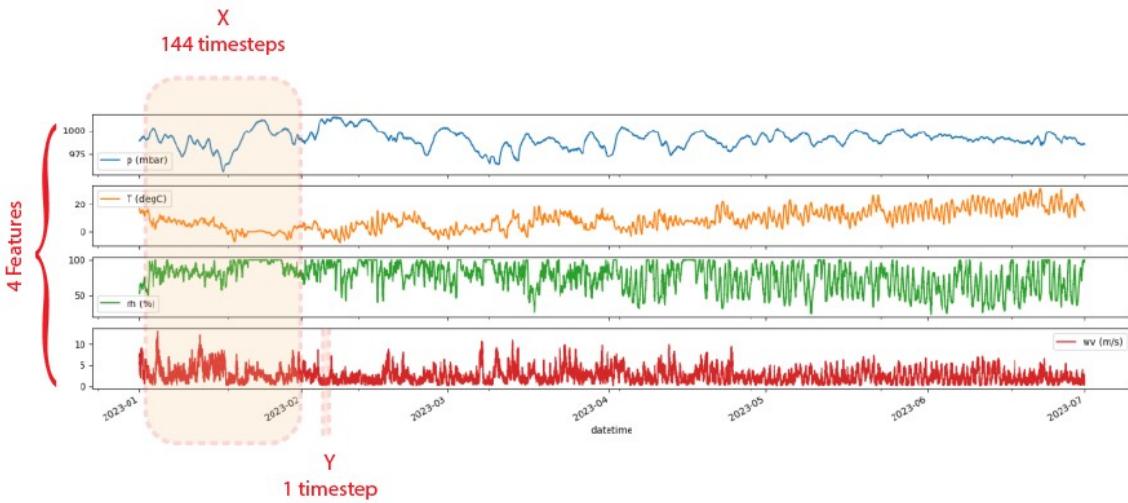


Figura 20: Pre-formateo de datos para la predicción

Cada secuencia de datos de entrada estará formado por esta ventana de datos, la cual para generar la próxima secuencia de datos se deslizara la ventana de datos con un paso igual a 1. De esta manera podremos generar la mayor cantidad de secuencias de datos para el uso durante el entrenamiento, validación y testeо de la red.

4.3.3 Creación de secuencias para LSTM (Features)

El primer paso es crear las secuencias de datos de 144 pasos de tiempo para los 4 features (Temperatura ambiente, Humedad Relativa del ambiente, Presión atmosférica y Humedad Relativa de la esponja de germinación. Definición de variables:

- **Tamaño de entradas (*INPUT_LENGTH*):** Esta variable define el tamaño de la secuencia de entrada al modelo LSTM, cada secuencia de entrada serán 144 muestras consecutivas representando 24 horas de información.
- **Paso a predecir (*FUTURE_STEP*):** Esta variable establece cuantos pasos en el futuro será el objetivo de predecir, y por lo tanto definirá el dato de salida a entregar en el momento del entrenamiento a la red LSTM.
- **Índice de Co-Variable a predecir (*OUTPUT_FEA_INDEX*):** esta variable estable en índice de la variable a predecir dentro del dataframe de datos.

Luego definimos una función "*crear_dataset_supervisado*" para generar las secuencias de datos de entrada y salida utilizadas durante el entrenamiento del modelo LSTM:

```
def crear_dataset_supervisado(array, input_length, future_step, out_feat_index):
    X, Y = [], []
    filas, cols = array.shape

    for i in range(len(array)-input_length-future_step):
        X.append(array[i:i+input_length, 0:cols])
        Y.append(array[i+input_length+future_step-1, out_feat_index])

    X = np.array(X)
    Y = np.array(Y)

    return X, Y
```

La salida de esta función serán dos arreglos NumPy:

- X: un arreglo de secuencias de entradas, cada una de 144 datos en el tiempo por cada co-variable de entrada.
- Y: un arreglo de Humedad Relativa de esponja fenólica, cada una corresponde a la humedad de la esponja 3 pasos en el futuro de cada secuencia de entrada.

En el último paso se aplica la función a cada uno de los dataset:

```
x_tr_n, y_tr_n = crear_dataset_supervisado(data_tr_n, INPUT_LENGTH, FUTURE_STEP, OUTPUT_FEA_INDEX)
x_vl_n, y_vl_n = crear_dataset_supervisado(data_vl_n, INPUT_LENGTH, FUTURE_STEP, OUTPUT_FEA_INDEX)
x_ts_n, y_ts_n = crear_dataset_supervisado(data_ts_n, INPUT_LENGTH, FUTURE_STEP, OUTPUT_FEA_INDEX)
```

4.3.4 Diseño y Entrenamiento del modelo LSTM

4.3.4.1 Diseño del Modelo

La arquitectura LSTM usa la API "Sequential" de Keras para implementar un modelo "many-to-one".

```
modelo = Sequential()
modelo.add(LSTM(N_UNITS, input_shape=INPUT_SHAPE))
modelo.add(Dense(1))
```

Donde:

- **LSTM Layer:** la primer capa del modelo es una capa LSTM con "N_UNITS"(hiper-parámetro) que en nuestro proyecto fue definido como 128, y el parámetro "input_shape" que indica las dimensiones del parámetro de entrada que el parámetro esperará.
- **Dense Layer:** Siguiente a la capa LSTM, tenemos una capa "Dense" con una sola neurona. Esta capa sirve como capa de salida y produce un único valor como salida. Esta configuración se usa comúnmente para tareas de regresión, como pronosticar un valor numérico continuo.

4.3.4.2 Compilado del modelo LSTM

Para el compilado del modelo se utiliza:

- **Optimizador: 'adam'.** El optimizador se encarga de la actualización de los pesos de la RNN. 'Adam' viene del inglés 'adaptative moment estimation' y es uno de los algoritmos de optimización más utilizados porque combina las mejores propiedades de los algoritmos AdaGrad y RMSProp para manejar gradientes dispersos en escenarios ruidosos. Adam es eficiente y requiere poca memoria, lo que lo hace adecuado para muchos problemas de redes neuronales, incluido el pronóstico de series temporales con LSTM.
- **Función de pérdida: 'mse'.** La función de pérdida, 'mse', significa error cuadrático medio. Mide la diferencia cuadrática promedio entre los valores reales y previstos. En el contexto de nuestro modelo, cuantifica qué tan cerca están la humedad del sustrato predecidas de las reales del conjunto de datos. Cuanto más bajo es este valor en el entrenamiento mejor será la precisión de las predicciones del modelo.

```
modelo.compile(
    optimizer = 'adam',
    loss = 'mse',
)
```

4.3.4.3 Entrenamiento del modelo

Después de compilar el modelo debemos entrenarlo proporcionando los arreglos de entrada y salida objetivo anteriormente preparados y otros parámetros de entrenamiento como el número de épocas y el tamaño del lote. Este proceso ajusta de forma iterativa los pesos del modelo para minimizar la pérdida, lo que idealmente da como resultado un modelo que puede predecir con precisión humedades de esponja de germinación futuras en función de la secuencia proporcionada de características de entrada.

Pero, para evitar el sobre-entrenamiento en los modelos de aprendizaje automático, utilizando Keras se definió una función de retorno denominada 'early_stopping' (detención

anticuada) utilizando Keras. La misma monitorea el rendimiento del modelo con el conjunto de datos de validación y detiene el proceso de entrenamiento si se detiene mejorando de la función de perdida.

```
from tensorflow.keras.callbacks import EarlyStopping
early_stopping = EarlyStopping(
    monitor='val_loss',
    patience=6,
    mode='min',
    restore_best_weights=True)
```

Donde estos son los parámetros utilizados:

- **monitor:** este parámetro especifica el valor a monitorear, que en nuestro caso es 'val_loss'. 'val_loss' refiere a la pérdida calculada del modelo utilizando el conjunto de datos de validación.
- **patience:** este parámetro (paciencia) define el número de épocas, sin mejora, después de las cuales se detendrá el entrenamiento. Al establecer patience=6 significa que si el valor de 'val_loss' no disminuye durante 6 épocas de entrenamiento consecutivas, el proceso de entrenamiento se detendrá. Esto le da al modelo cierto margen de maniobra para superar algunos diclos sin conseguir mejoras en el progreso del entrenamiento.
- **mode:** El modo 'min' significa que el entrenamiento se detendrá cuando la cantidad monitoreada ('val_loss') deje de disminuir. Esto tiene sentido porque menos es mejor en términos de pérdidas; y el objetivo principal es obtener el mínimo.
- **restaure_best_weights:** cuando se establece en 'True', esta opción restaura los pesos del modelo de la época con el mejor valor de la métrica monitoreada ('val_loss'). Esto significa que mantendremos el mejor estado del modelo incluso si el rendimiento del modelo se degrada en las épocas posteriores a la mejor (dentro del período de paciencia).

El Modelo se entrena gracias a la función 'model.fit()' utilizando el conjunto de datos de entrenamiento normalizados ('x_tr_n' , 'y_tr_n'), al mismo tiempo la evaluación se realizará con el conjunto de datos ('x_vl_n' , 'y_vl_n'), como parámetro función de retorno se pasará 'early_stopping', la cual si la función de pérdida durante la validación no mejora durante seis épocas consecutivas, el entrenamiento se detendrá antes de tiempo y los pesos del modelo volverán a los de la época con la pérdida de validación más baja, lo que evitará eficazmente el sobreentrenamiento , ahorrando recursos computacionales.

Este proceso de entrenamiento de detuvo en la época 11.

La evaluación del modelo calculada a través del error cuadrático medio (RMSE) sobre el conjunto de datos de entrenamiento nos arroja un valor de 0,00318 lo cual es muy

```

historia = modelo.fit(
    x = x_tr_n,
    y = y_tr_n,
    batch_size = 20,
    epochs = 30,
    validation_data = (x_vl_n, y_vl_n),
    callbacks=[early_stopping],
    verbose=2
)

```

prometedor para nuestro modelo, indicando que tendrá una buena performance prediciendo al humedad de la esponja fenólica en función de las variables físicas elegidas como entrada del mismo.

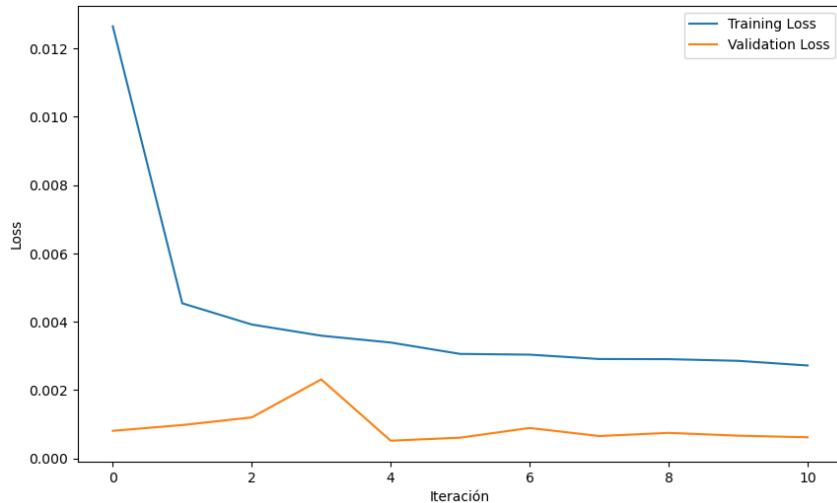


Figura 21: Curvas de pérdida: set de entrenamiento vs. validación

La Figura 21 gráfica los valores de pérdida (RMSE) del modelo durante las diferentes iteraciones de entrenamiento, tanto con el set de entrenamiento (azul), como para el set de validación (Naranja). Como se puede ver, las curvas tienen a mismo punto lo cual indica que no existe sobreentrenamiento del modelo. Esto y la baja magnitud de la función de perdida nos dan un indicio que el modelo podrá ser efectivo durante la predicción de valores desconocidos.

Con el modelo entrenado, calculamos el error cuadrático medio (RMSE) de cada conjunto de datos de nuestro dataset, obteniendo los siguientes valores:

- RMSE train: 0.00283
- RMSE val: 0.00052
- RMSE test: 0.00034

4.3.4.4 Predicciones con el modelo LSTM entrenado

El próximo paso consistió en poner a prueba el modelo entrenado con las primeras 200 muestras del conjunto de datos de test ('x_ts_n'). Tanto el set de datos real 'y_ts_n',

como los valores arrojados durante la predicción se encuentran normalizados, por lo cual para simular el funcionamiento final del sistema, realizamos la conversión inversa de a la escala original y graficamos los resultados, Figura 22 . En la misma se puede apreciar los valores reales capturados por el datalogger(azul) y los valores predecidos por el modelo (rojo) para las primeras 200 muestras del dataset de prueba (Time Steps)

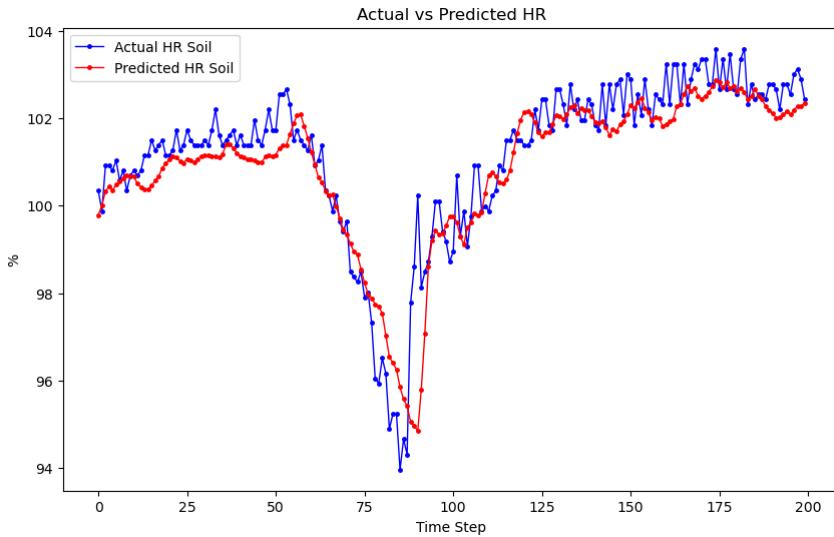


Figura 22: Comparación entre valores reales vs. valores predecidos por el modelo

4.4 Implementación RNN para el Sistema Embebido

Lo próximo es adaptar nuestro modelo entrenado para poder ser utilizado en el microprocesador ES32, para ello usaremos las herramientas de TensorFlow Lite, y TensorFlow Micro. Una vez obtenido el modelo en este formato utilizaremos las herramientas de Edge Impulse para poder realizar el deployment de nuestro modelo LSTM a nuestro sistema embebido. Para este proyecto, utilizamos el SDK Edge Impulse Python, una biblioteca que ayuda a desarrollar aplicaciones de aprendizaje automático (ML) para dispositivos con inteligencia de borde y de Internet de las cosas (IoT). Si bien Edge Impulse Studio es una excelente interfaz para guiarlo acompañar el proceso de recopilación de datos y entrenamiento de un modelo, el SDK de Python de edgeimpulse le permite traer su propio modelo (BYOM) mediante programación, desarrollado y entrenado en cualquier plataforma (Figura 23)

La conversión de un modelo de TensorFlow a TensorFlow Lite (TFLite) para su implementación en microcontroladores (TensorFlow Lite Micro) tiene algunas limitaciones y consideraciones a tener en cuenta a la hora de utilizar el SDK de Python desarrollado por Edge Impulse:

- 1. Soporte del operador:** no todas las operaciones de TensorFlow son compatibles con TFLite, y el soporte es aún más limitado para TensorFlow Lite Micro. A partir

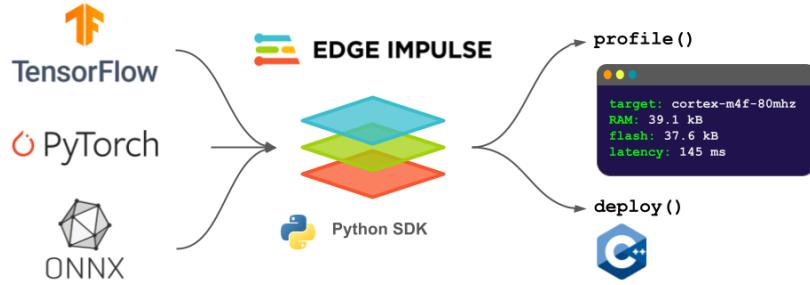


Figura 23: Características Edge Impulse Python SDK

de la última actualización, solo UnidireccionalLSTM es compatible con operaciones LSTM en TensorFlow Lite Micro, por lo que debemos asegurarnos de que nuestro modelo use solo este tipo de capa LSTM.

2. **Cuantificación:** la cuantificación es el proceso de reducir la precisión de los números utilizados para representar los parámetros de un modelo, lo cual es esencial para ejecutar modelos en dispositivos con precisión y memoria limitadas. Aunque los modelos float32 son compatibles y probados, los modelos cuantificados a veces pueden presentar desafíos, particularmente con TensorFlow Lite Micro, que puede no admitir completamente la cuantificación o no tener un soporte maduro para todas las operaciones en un formato cuantificado. Por tanto, no utilizaremos la cuantificación en este proyecto.

4.4.1 Create TFLite LSTM Model

Teniendo en cuenta las consideraciones expuestas, utilizamos el siguiente código para la conversión del modelo a modelo TFLite:

```
TF_LITE_MODEL_NAME = "tf_lite_model_UnidireccionallSTM.tflite"

run_model = tf.function(lambda x: modelo(x))
# This is important, let's fix the input size.
BATCH_SIZE = 1
STEPS = 144
INPUT_SIZE = 4
concrete_func = run_model.get_concrete_function(
    tf.TensorSpec([BATCH_SIZE, STEPS, INPUT_SIZE], modelo.inputs[0].dtype))

# model directory.
MODEL_DIR = "modelo_keras_lstm"
modelo.save(MODEL_DIR, save_format="tf", signatures=concrete_func)

converter = tf.lite.TFLiteConverter.from_saved_model(MODEL_DIR)
tflite_model = converter.convert()

# Save the converted model to file
with open(TF_LITE_MODEL_NAME, 'wb') as f:
    f.write(tflite_model)
```

Una vez obtenido el modelo en formato Tensor Flow Lite verificamos utilizando Neutron que todas las operaciones sean del tipo unidireccionales para no tener problemas a la hora de hacer el desarrollo para el microcontrolador con Edge Impulse Python SDK



4.4.2 Implementación del modelo con Edge Impulse Python SDK

4.4.2.1 Librería de Python Edge Impulse SDK

Instalación de la misma con el comando:

```
python -m pip install edgeimpulse
```

4.4.2.2 API Key Edge Impulse

Para usar el SDK de Python, primero se debe crear un proyecto en Edge Impulse y copiar la clave API. Luego en el Panel hacemos clic en la pestaña Claves para ver la clave API, y de allí la copiamos.

| NAME | API KEY | ROLE | CREATED |
|----------------|--|-------|---------------------|
| my-project-key | ei_dae27aebe0fc38473a80e12454752c13ed686283719361c2a3... | Admin | Yesterday, 13:07:26 |

En el proyecto, importamos el paquete y inicializamos la clave API:

```
import edgeimpulse as ei
ei.API_KEY = "ei_dae27aebe0fc38473a80e12454752c13ed686283719361c2a3..." # Change to your key
```

4.4.2.3 Cálculo de recursos necesarios de microcontrolador

Se calcula los recursos que el modelo necesitará una vez implementado para el microcontrolador ESP32 mediante la siguiente función de la librería:

```
try:
    profile = ei.model.profile(model=tflite_model,
                                device='espressif-esp32')
    print(profile.summary())
except Exception as e:
    print(f"Could not profile: {e}")
```

Obteniendo como resultado: Lo cual nos confirma que las capacidades de RAM (98K de 512K) y ROM (318K de 384K) del microcontrolador son suficientes, y nos predice un tiempo por inferencia de 22,99 Segundos lo que parece muy alto, pero que se deberá corroborar físicamente.

```

Target results for float32:
=====
{
  "device": "espressif-esp32",
  "tfliteFileSizeBytes": 276880,
  "isSupportedOnMcu": true,
  "memory": {
    "tflite": {
      "ram": 98891,
      "rom": 318008,
      "arenaSize": 98675
    },
    "eon": {
      "ram": 82264,
      "rom": 298792
    }
  },
  "timePerInferenceMs": 22990
}

```

4.4.2.4 Generación de la Biblioteca de Implementación

Para implementar nuestro modelo, utilizamos la función `deploy()` para convertir el modelo de TFLite a una de las salidas compatibles con Edge Impulse. Edge Impulse puede generar una serie de posibles bibliotecas de implementación y archivos binarios pre-compilados para una amplia variedad de hardware de destino, para este proyecto se usó 'Arduino'. También deberíamos definir el tipo de salida, por ejemplo Clasificación o Regresión. En nuestro caso, `Regression()`.

```

# Create Arduino Library with trained model
download_dir = "./"
deploy_filename = "lstm_float32_model_3step.zip"
deploy_bytes = None

try:
    deploy_bytes = ei.model.deploy(tflite_model,
                                   model_output_type=ei.model.output_type.Regression(),
                                   deploy_target='arduino')
except Exception as e:
    print(f"Could not deploy: {e}")

# Write the downloaded raw bytes to a file
if deploy_bytes:
    with open(deploy_filename, 'wb') as f:
        f.write(deploy_bytes.getvalue())

```

Este código nos guarda la librería "lstm_float32_model_3step.zip" para importar y utilizar con el framework de Arduino.

4.5 Desarrollo del Sistema Embebido

Una vez obtenido el modelo entrenado y probado en el microcontrolador, ensamblamos el circuito electrónico y utilizamos la librería generada por el SDK de Edge Impulse para desarrollar el Firmware encargado de llevar adelante la tarea objeto: predecir la humedad de la esponja de germinación.

4.5.1 Desarrollo del hardware

Según lo desarrollado en la Sección 4.1.5, se realiza la construcción del hardware del sistema embebido. Para esta fase de prueba se utilizará alimentación a través del puerto USB-C de XIAO ESP32-S3, y se conectará una batería de polímero de Litio de 3.7V

[600mA] en los bornes correspondientes del módulo XIAO como fuente redundante en caso de corte de energía eléctrica. Cabe rescatar que el módulo del microcontrolador ya posee integrado un circuito de carga para la misma utilizando su alimentación general.

La figura 24 muestra la integración del Hardware del sistema embebido, donde sus dimensiones finales del prototipo logrado fueron de 20 x 50 mm.



Figura 24: Hardware Sistema Embebido

4.5.2 Desarrollo del Firmware

Para el desarrollo del firmware se codificó en C++ utilizando el framework Arduino y su IDE.

4.5.2.1 Configuración de Alarmas

Recordando que la principal funcionalidad del sistema embebido será predecir la humedad de la esponja fenólica durante la germinación, el paso fundamental es conocer los umbrales adecuados para emitir la alarmas, para ello se consultó al Ingeniero Agrónomo encargado de la empresa donde se realizó el sensado de variables para el armado del dataset. El profesional responsable estableció que la humedad ideal para la germinación debe ser entre **65 y 95%** relativo a su capacidad máxima de retención, y remarcó que los problemas en la germinación por causa de la mala hidratación se comienzan a producir cuando las semillas están con niveles entre **65 y 70% HR**. Con esta información, establecimos las 2(dos) alarmas luminosas del sistema:

- **Alarma Amarilla:** cuando el sistema predice una HR de esponja entre el 65 y 75%
- **Alarma Roja:** cuando el sistema predice una HR por debajo de 65%

4.5.2.2 Utilización de Librerías

Para el manejo del sensor de variables atmosféricas BME280 se utilizó la librería desarrollada por Tyler Glenn [Glenn, 2021]. Estableciendo como unidades de medidas el Celcius[°C] para la temperatura, y Pascal[Pa] para la presión. Para la medición se utiliza el método de la clase desarrollada en la librería realizando un muestreo (cinco mediciones separadas 1 segundo) y calculando la media para cada variable.

Para la realizar la inferencia de nuestra librería exportada por el SDK de EdgeImpulse.

4.5.2.3 *Integración*

La otra co-variable de entrada al modelo es la medida por el sensor de Humedad de Esponja Fenólica, para ello se utiliza el seleccionado para esta función directamente conectado a una entrada analógica, y para la transducción de los valores de tensión a %HR se utiliza la función de regresión obtenida durante el ensayo del mismo.

Las variables son medidas cada 10 minutos, luego son normalizadas utilizando los escaladores y la formula obtenidos para cada magnitud durante el procesamiento del dataset 4.2.6.2.

Una vez normalizadas las mediciones se guardan en un '*buffer*' tipo FIFO de dimensiones 4 x 144, el cual guardará las ultimas 24 horas de mediciones.

El próximo paso es convertir es re-dimensionar el formato del buffer para poder ser utilizado como entrada del modelo LSTM, para ello lo "aplanamos" convirtiéndolo a un vector llamado '*features*' de dimensión igual a 576.

Inmediatamente después de ello se realiza la inferencia obteniendo el valor predecido, el cual es comparado con los límites de alarmas y en caso de encontrar el resultado dentro de las franjas de alarma establecidas, encenderá el LED relacionado.

5 RESULTADOS

A continuación se comparten los resultados observados durante el desarrollo de cada uno de los objetivos del proyecto

5.1 Evaluación del dataset

Al no encontrar de forma pública un dataset que contenga las co-variables necesarias para el desarrollo del proyecto, el paso principal fue construirlo a través de la obtención en un ambiente de germinación real, a través de un datalogger desarrollado para tal fin. La recolección de datos comenzó el 12 de abril del 2014 luego de algunos upgrade de firmware del datalogger corrigiendo algunos bugs y exactitud del RTC velando por la correcta periodicidad de las muestras. El 29 de abril la plantación fue azotada por una tormenta la cual incluyó a 'una cola de un tornado' según los lugareños, esto provocó serios daños y particularmente el datalogger sufrió una interrupción en su alimentación de energía, lo que conllevo que su batería se agotara al pasar unas horas, ya que el mismo no fue diseñado para una gran cantidad de horas de autonomía. La anomalía se detectó el 4 de mayo siguiente, lo que produjo una interrupción en la serie de tiempos de las muestras obtenidas. Luego de ello, el datalogger volvió a colocarse operativo hasta el 18 de junio.

Se obtuvieron las siguientes muestras:

- 2266 muestras consecutivas en el tiempo, entre el 12/04 y el 28/04
- 6474 muestras consecutivas en el tiempo, entre el 4/05 y el 18/06

Como para el entrenamiento y uso de las redes LSTM es fundamental que continuidad de las series de tiempos que se utilizan como entradas, solo se utilizó para el proyecto la segunda serie de muestras, de un total de 6474 muestras, cada una incluyendo temperatura y humedad relativa ambiente, presión atmosférica, y humedad relativa del sustrato de germinación.

El conjunto de datos utilizados pertenecen estacionalmente a el período de otoño, en la zona rural de la ciudad de Rafaela, Santa Fe, Argentina (-31.252622484615674, -61.491732297294064). Podemos analizar que el período de adquisición de datos influyó

en los rangos de las variables ambientales y sus límites, pudiendo observarse esto en la tabla 6:

| | Temperatura Ambiente | Humedad Relativa Ambiente | Presión Atmosférica | Humedad Relativa Esponja Fenólica |
|---------------|----------------------|---------------------------|---------------------|-----------------------------------|
| Mínimo | 1,11 °C | 30,46 % | 1001,54 hPa | 71,099 % |
| Máximo | 28,02 °C | 86,18 % | 1033,13 hPa | 106,377 % |
| Rango | 26,91 °C | 55,72 % | 31,59 hPa | 35,278 % |

Tabla 6: Dataset - Rangos

Según el archivo histórico de factores climáticos para el año 2023 y la ubicación donde se recabaron los datos, el rango de temperatura anual oscila en 46° C (-3 a 43°C) y el rango de humedad relativa en 80% (8 a 88 %HR), véase la figura 25 de referencia.

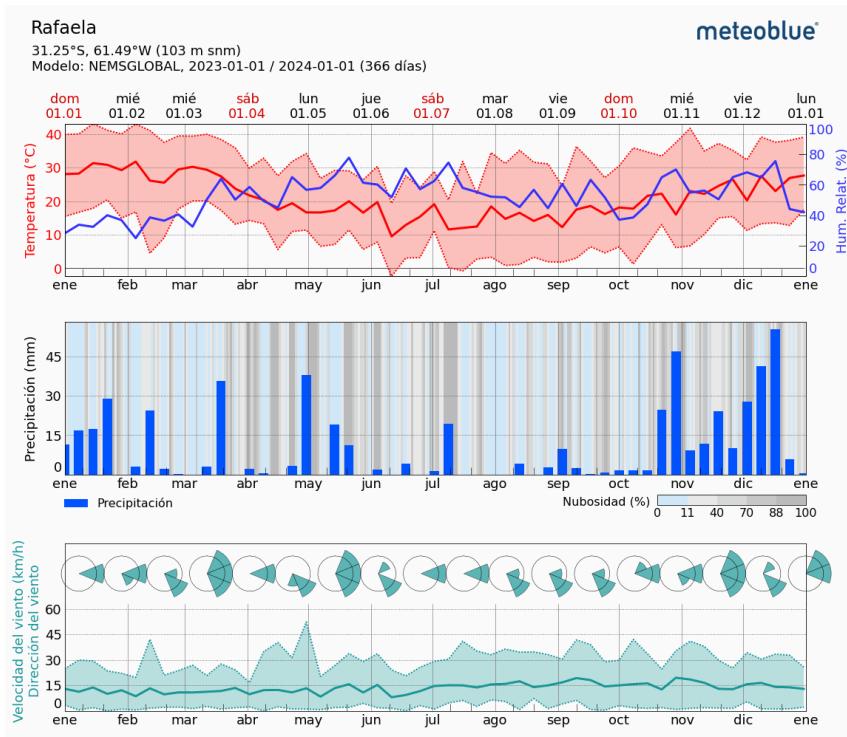


Figura 25: Datos climáticos históricos Rafaela(Sta Fe- ARG) - fuente: [meteoblue.com](https://www.meteoblue.com)

Un análisis importante obtenido es el rango y valores de la Humedad Relativa de la Esponja Fenólica, se observó que valores máximos de humedad relativa de la esponja recabados iban aumentando hasta llegar a valores por encima de la máxima capacidad de retención de líquido de la esponja (106%), este comportamiento se pudo asociar al crecimiento de raíces dentro de la esponja fenólica observando los máximos entre la tercera y cuarta semana de germinación, justo antes que las mismas fueran trasplantadas al sector de crecimiento productivo. Este fenómeno está íntimamente relacionado a que la presencia de las raíces ocasionan mejora en la permeabilidad eléctrica por lo cual el campo generado por el sensor , lo cual arrastra a mediciones que arrastran este error positivo, el cual estaría cubierto por la generación de una alarma amarilla por el sistema la cual esta desplazada 10%HR del valor mínimo solicitado por el Ingeniero Agrónomo .

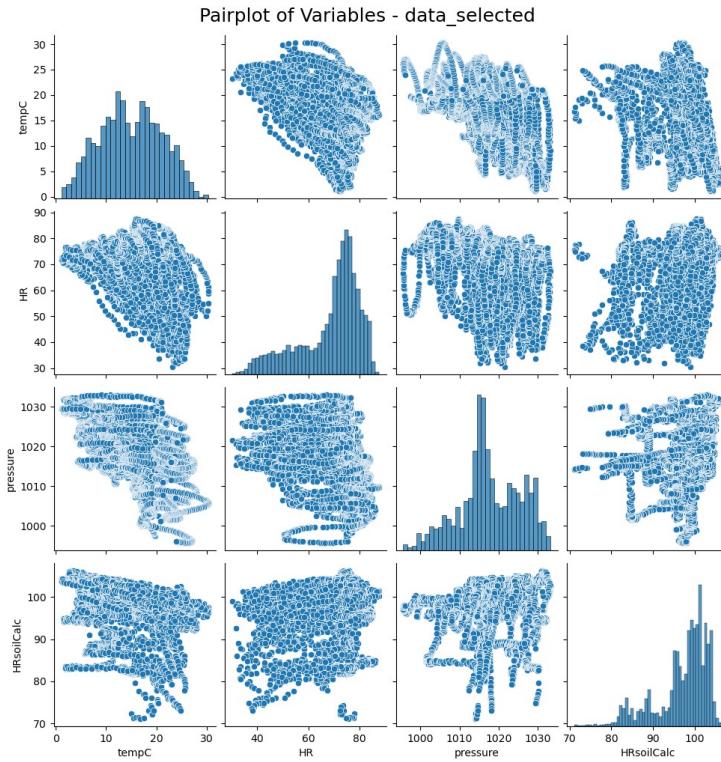


Figura 26: Dataset - Correlación entre variables de entrada

El último análisis se basa en analizar el comportamiento y la independencia de las co-variables entre ellas, esto ayudaría a poder encontrar variables que varíen en forma lineal entre ellas, lo cual ayudaría a diagnosticar variables que podrían no tener peso como entrada del modelo. Como se puede apreciar en el siguientes gráficos donde se comparan variable versus variable, no hay ninguna que posea su comportamiento directa y únicamente relacionado a otra de las seleccionadas como entrada del modelo. Figura 26

5.2 Evaluación del Modelo LSTM

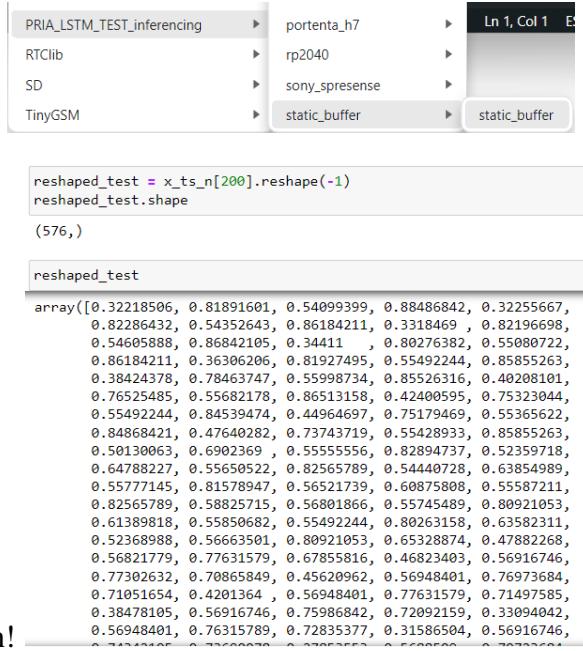
El primer punto evaluado es la exactitud del modelo LSTM. El modelo LSTM de tipo "many-to-one" en primer instancia se configuró para pronosticar el valor de HR del sustrato 3(tres) períodos en el futuro, entregando una buena exactitud.

5.2.1 Testeo de inferencia

El primer paso fue incluir la librería generada en el IDE utilizado, para este caso utilizamos el IDE de ARDUINO, para ello en un nuevo sketch nos dirigimos al menu 'sketch', seleccionamos la opción "*Include Library*", y luego "*Add .ZIP library*"

Una vez instalada la librería, dentro de la sección de "Examples" encontraremos un ejemplo para probar la inferencia bajo una libreria que lleva el nombre del proyecto de Edge Impulse (Examples\PRIA_LSTM_TEST_Inferencing\static_buffer \ static_buffer)

Para probar el modelo utilizando en el sketch "static_buffer", necesitaremos inicializar



la variable static const float features[] = con un conjunto de datos de prueba.

Debemos tener en cuenta que Edge Impulse implementa el modelo de tal manera que el tensor de entrada debe ser "plano", entonces, un punto de datos con una forma como (144, 4) debe reformarse a (576,). Para obtener el mismo, lo obtenemos desde Python:

El array generado como punto de prueba se copia y pega en el sketch:

```
1 #include <PRIA_LSTM_TEST_inferencing.h>
2
3
4 static const float features[] = {
5     0.32218506, 0.81891601, 0.540099399, 0.88486842, 0.32255667,
6     0.286432, 0.54352643, 0.86184211, 0.3318469 , 0.82196698,
7     0.605888, 0.86842105, 0.34411 , 0.80276382, 0.55080722,
8     0.184211, 0.36306206, 0.81927495, 0.5549244 , 0.85855263,
9     0.424378, 0.78463747, 0.55998734, 0.85526316, 0.40208101,
10    0.525485, 0.55682178, 0.86513158, 0.42400595, 0.75323044,
11    0.492244, 0.84539474, 0.44964697, 0.75179469, 0.55365622,
12    0.868421, 0.47640282, 0.73743179, 0.55428933, 0.85855263,
13    0.130063, 0.6902369 , 0.55555556, 0.82894737, 0.52359718,
14    0.788227, 0.55650522, 0.82565789, 0.54404728, 0.63854989,
15    0.777145, 0.81578947, 0.56521739, 0.60875808, 0.55587211,
16    0.565789, 0.58825715, 0.56801866, 0.55745489, 0.80921053,
17    0.389818, 0.55850682, 0.55492244, 0.80263158, 0.63852311,
18    0.368898, 0.56663501, 0.80921053, 0.65328874, 0.47882268,
19    0.821779, 0.77631579, 0.67855816, 0.46823403, 0.56916746,
20    0.302632, 0.70865849, 0.45620962, 0.56948401, 0.76973684,
21    0.051654, 0.4201364 , 0.56948401, 0.77631579, 0.71497585,
22    0.478105, 0.56916746, 0.75986842, 0.72092159, 0.33094042,
23    0.948401, 0.76315789, 0.72835377, 0.31586504, 0.56916746,
```

Inspeccionamos el valor real del conjunto de datos $y_ts_n[200]$ el cual es: 0.9046052631578951 . Este valor debería predecir nuestro microprocesador al pasarle el conjunto de datos de pruebas anteriores.

Conectamos el XIAO ESP32-S3 , cargamos y corremos el sketch de prueba, y obtenemos el siguiente resultado en el Monitor Serie:

Verificamos que el valor entregado por la inferencia para el conjunto de datos de prueba fue: **0.89170, lo que nos arroja un error de 0.0129 en relación al valor real de salida para este conjunto de datos, lo cual indicaría un error relativo del orden 1,426%**

```

static_buffer.ino
13 * limitations under the license.
14 *
15 */
16
17 /* Includes ----- */
18 #include <PRIA_LSTM_TEST_inferencing.h>
19
20 static const float features[] = {0.32218506, 0.81891601, 0.54099399, 0.88486842, 0.32255667,
21     0.82286432, 0.54352643, 0.86184211, 0.3318469 , 0.82196698,
22     0.54605888, 0.86842105, 0.34411 , 0.80276382, 0.55080722,
23     0.86184211, 0.36306206, 0.81927495, 0.55492244, 0.85855263,
24     0.38424378, 0.78463747, 0.55998734, 0.85526316, 0.40208101,
25     0.76525485, 0.55682178, 0.86513158, 0.42400595, 0.75323044,
26     0.55492244, 0.84539474, 0.44964697, 0.75179469, 0.55365622,
27     0.84868421, 0.47640282, 0.73743719, 0.55428933, 0.85855263,

```

Output Serial Monitor X

Not connected. Select a board and a port to connect automatically.

- - -

run_classifier returned: 0
Timing: DSP 0 ms, inference 1702 ms, anomaly 0 ms
Predictions:
value: 0.89170

(0.0129/0.90461). También se observa que el tiempo de latencia es de 1,7 segundos lo que es muy aceptable para este proyecto ya que se generará una predicción cada 10 minutos.

5.2.2 Testeo de Hiper-parámetros del modelo

A modo de aplicación, sería mucho mejor poder predecir los valores no deseados de humedad de espuma fenólica con el mayor tiempo de antelación posible, por lo cual en primer instancia se ensayó el modelo para predecir los valores desde T+1 a T+6, para contrastar como varía la exactitud del modelo a medida que deseamos predecir pasos mas futuros. El cuadro 7 muestra los resultados de dicho ensayo:

| RMSE | T+1 | T+2 | T+3 | T+4 | T+5 | T+6 |
|-------------------|---------|----------|---------|---------|---------|---------|
| Train | 0.00103 | 0.000221 | 0.00283 | 0.00439 | 0.00588 | 0.00569 |
| Validation | 0.00039 | 0.00044 | 0.00052 | 0.00064 | 0.00079 | 0.00079 |
| Test | 0.00023 | 0.00028 | 0.00034 | 0.00044 | 0.00054 | 0.00050 |

Tabla 7: Modelo LSTM - exactitud vs pasos futuros

Para la comparación de exactitud entre modelos se utilizó el error cuadrático medio en lugar del error relativo para independizarnos de la variable física y enfocar en la eficiencia , viendo que el RSME aumenta a medida que deseamos predecir un paso mas lejano en el futuro, y con podemos estipular que si el sistema embebido tratase de predecir la humedad 6 pasos en el futuro, su exactitud sería mucho menor contemplando que el RMSE aumenta entre 1,47 y 2 veces.

Otro ensayo importante al modelo es para evaluar la sensibilidad de la exactitud en relación a la cantidad de muestras del set de datos que recibe para hacer la predicción, inicialmente se seleccionó 144 muestras por variables, lo que correspondía a las ultimas 24 horas. Para evaluar la sensibilidad a este parámetro, se ensaya modelos que reciban 36hs , 12hs y 6hs de muestras para predecir, el valor siempre en el período de tiempo t+3, apreciándose los resultados en la tabla 8.

| RMSE | 36 Horas | 24 Horas | 12 Horas | 6 Horas |
|-------------------|----------|----------|----------|---------|
| Train | 0.00296 | 0.00283 | 0.00256 | 0.00259 |
| Validation | 0.00053 | 0.00052 | 0.00051 | 0.00049 |
| Test | 0.00044 | 0.00034 | 0.00038 | 0.00036 |

Tabla 8: Modelo LSTM - exactitud vs tamaño del vector de entrada [horas]



Figura 27: Sistema Embebido - Ensayo

Podemos ver en este caso que no existe una gran variación de la precisión del modelo si recibe series de variables en el tiempo con antelación de 6 a 24 horas.

Otro objetivo importante del proyecto fue lograr correr el modelo LSTM en un microcontrolador para poder ser integrado en un sistema embebido, esto fue logrado gracias al SDK provisto por Edge Impulse, y probado con éxito en una linea de microprocesadores simples y económicos como lo son los ESP32. A continuación se muestra los resultados de realizar la inferencia de nuestro modelo en los microprocesadores preseleccionados como posibles para el sistema embebido. Tabla 9 .

| | ESP32-S3 | ESP32-C3 | ESP32-WROOM |
|--|--------------------|----------------|--------------------|
| Arquitectura | Xtensa LX7 32 bits | RISC-V 32 bits | Xtensa LX6 32 bits |
| Tiempo para realizar la inferencia del modelo LSTM | 1,7 seg. | 17,2 seg. | 2,1 seg. |

Tabla 9: Comparación de Inferencia en familia de microprocesadores ESP32

En este sentido, cualquiera de los microcontroladores puede cumplir la función considerando que el tiempo entre predicciones es de 10 minutos, y la predicción objetivo es a 30 minutos.

5.3 Evaluación del Sistema Embebido

En cuanto al sistema embebido el análisis de desempeño es abordado desde varias puntos de vista. Figura 28

5.3.1 Energía

El diseño propuesto en funcionamiento desde el punto del consumo de energía poseerá dos regímenes:

5.3.1.1 Uso de energía durante el Período de Medición de variables

Durante este período los sensores necesitarán energía para realizar las mediciones de sus variables y el microcontrolador el manejo de esas funciones:

- Sensor BME208: menor a $3.6\mu\text{A}$
- Sensor Humedad de Sustrato: 0.1mA
- XIAO ESP32-S3: 22mA

La duración de este período es de aproximadamente 2 segundos, totalizando aproximadamente unos 22.1mA de consumo durante dicho tiempo.

5.3.1.2 Uso de energía durante el Periodo entre Mediciones

Durante este tiempo el sistema con el firmware estándar que se utilizó para la prueba necesitará proveer las siguientes corrientes:

- Sensor BME208: menor a $3.6\mu\text{A}$
- XIAO ESP32-S3: 22mA
- LEDS: pulsos de 10mA , con τ entre 75 y 150mS (según la señalización) sobre un T de 5000mS equivaldría a un consumo eficaz de 1.73mA

La duración de este período es de aproximadamente 598 segundos, totalizando unos 24mA durante este periodo, y pudiendo estipular el **consumo eficaz general a $23,96\text{mA}$** .

Con la batería de capacidad 600mA/h que posee el sistema posee unas **25 horas de autonomía**.

Un versión mejorada diseñada en base a funcionalidades de IoT podría lograr eliminar las señales lumínicas reemplazándolas por algún tipo de baliza transmitida (Ejemplo MQTT vía WiFi) y de esta forma poder colocar al microcontrolador en modo *Deep-Sleep* lo cual su consumo se reduciría a $14\mu\text{A}$, llevando al consumo durante este período a unos $17\mu\text{A}$, y el consumo eficaz total a unos $1,275\text{mA}$. Lo cual con la misma batería el sistema podría tener unas **470 horas, unos 19 días de autonomía antes de su recarga**.

5.3.2 Desempeño

Durante el testeo y ensayo del sistema embebido ensamblado el sistema presentó un buen desempeño. El mismo se ensayó en un ambiente montado en una galería simulando las condiciones climáticas del espacio de germinado, pero con la conexión a un puerto serial de un computador para poder ir evaluando el desempeño, los valores medidos, normalizados, y predecidos para evaluar el funcionamiento del sistema.

Como muestra la figura 28 en cada período de medición el sistema embebido fue programado especialmente para esta fase de testeo para que comparta los valores medidos

de las cuatro variables físicas , los valores normalizados que son ingresadas al buffer que luego se utiliza como entrada al modelo, y el valor predecido por el modelo para la humedad de la esponja fenólica, 30 minutos en el futuro.

The screenshot shows the Arduino Serial Monitor window. At the top, it says "PRIA_LSTM_ARDUINO.ino". Below that, there is code for defining constants:

```

23 const float airHRScalerMax = 86.18;
24 const float airHRScalerMin = 30.46;
25
26 const float moscalorMTn = 1001.51;

```

Below the code, the "Serial Monitor" tab is selected. It displays a message: "Message (Enter to send message to 'XIAO_ESP32S3' on 'COM9')". The main area shows sensor data and a prediction:

| | a | b | c | d |
|-----------------|-------------------------------|-------|---------|-------|
| 16:48:08.147 -> | v_ADC: 0.932 | | | |
| 16:48:08.147 -> | 25.04 | 29.02 | 1011.61 | 92.22 |
| 16:48:08.147 -> | 0.89 | -0.03 | 0.32 | 0.60 |
| 16:48:09.846 -> | Predicción re-escalada: 91.40 | | | |
| 16:58:08.161 -> | v_ADC: 0.940 | | | |
| 16:58:08.161 -> | 24.94 | 27.15 | 1011.63 | 91.29 |
| 16:58:08.161 -> | 0.89 | -0.06 | 0.32 | 0.57 |
| 16:58:09.846 -> | Predicción re-escalada: 89.56 | | | |
| 17:08:08.180 -> | v_ADC: 0.935 | | | |
| 17:08:08.180 -> | 24.94 | 26.65 | 1011.66 | 91.87 |
| 17:08:08.180 -> | 0.89 | -0.07 | 0.32 | 0.59 |
| 17:08:09.881 -> | Predicción re-escalada: 89.25 | | | |
| 17:18:08.188 -> | v_ADC: 0.939 | | | |
| 17:18:08.188 -> | 24.96 | 27.36 | 1011.68 | 91.41 |
| 17:18:08.188 -> | 0.89 | -0.06 | 0.32 | 0.58 |
| 17:18:09.842 -> | Predicción re-escalada: 89.53 | | | |

Figura 28: Sistema embebido - Resultados de su ensayo

Donde los valores de las variables medidas son mostrados según las siguientes referencias:

- a = Temperatura Ambiente
- b = Humedad Relativa Ambiente
- c = Presión Atmosférica
- d = Humedad Relativa Espuma de germinación
- e = Humedad Relativa predecida para la Espuma de germinación, 30 minutos en el futuro.

Como se puede ver, el sistema a las 16:48 horas predijo para las 17:18 (30 minutos en el futuro) una humedad de esponja fenólica de 91.40% , y a dicha hora podemos ver que la humedad de la misma fue 91,41%, siendo totalmente satisfactorio el resultado de funcionamiento.

6 CONCLUSIONES Y DISCUSIONES

A lo largo del presente trabajo final se pudieron desarrollar todas las tareas relacionadas a poder lograr todos los objetivos planteados, las cuales fueron en mayor medida instancias de aprendizaje y de afianzado de conocimientos obtenidos durante el cursado del PRIA. En muchas ocasiones las tareas y los tiempos fueron muchos mas extensos de lo planeado.

El primer objetivo específico del proyecto relacionado a generar un dataset real fue un desafío ya que debía realizarse en un espacio de producción cedido por una empresa de hidroponia, aunque ello conllevo que por contingencias climáticas o negligencias del personal de la empresa la serie datos en el tiempo guardado en el datalog presentaban discontinuidades, finalmente se logró un dataset que fue útil para el entrenamiento de la RNN. Dicho dataset puede accederse públicamente en el repositorio generado por este proyecto, ver Anexo B de este proyecto.

El hito mas importante y relacionado al cumplimiento del segundo objetivo quizás para este proyecto es lograr funcionar redes LSTM en dispositivos económicos como lo son los microcontroladores linea ESP32, y ello se logró gracias a que la firma Edge Impulse a comienzos del año en curso nos compartió el SDK para poder realizar el deployment desde el modelo LSTM desarrollado utilizando Keras en Python, a librerías de C# y específicamente para el framework de Arduino, simplificando luego la implementación en el sistema embebido. No sólo el poder realizar la implementación fue importante, sino poder testear su funcionamiento tanto en arquitecturas internas de microprocesadores de doble núcleo como la es la Xtensa de los ESP32S3 y ESP32WROOM, sino también en arquitecturas más simples como la RISC_V del ESP32C3. Una prueba de este cumplimiento puede encontrarse en los resultados expuestos en la sección homónima de este trabajo final, y replicable gracias al notebook de Python utilizado y compartido en el repositorio del proyecto dentro del Anexo B.

En relación al objetivo de realizar el desarrollo de una implementación del modelo en un sistema embebido de bajo costo y dimensiones, con alertas para que el cultivo se encuentre en margenes favorables de humedad de la esponja fenólica donde se desarrolla el sustrato, se llevo adelante la integración y armado de un prototipo con dimensiones

bajas como se pudo ver en la sección de resultados, y con dos alarmas lumínicas que fueron definidas por el usuario final (Ingeniero Agrónomo), y quien estableció que como objetivo base, el mismo fue cumplido dentro de lo que se esperaba para su funcionalidad.

La última fase del trabajo final se enfocó en poder validar junto al usuario final el funcionamiento del sistema embebido para lo cual, como se desarrolló en el capítulo de resultados, se sometió al sistema embebido a condiciones normales en la sección de germinación de la empresa de hidroponia pero con el sensor conectado a un terminal serial para ir corroborando que los valores predecidos y reales al pasar el tiempo tengan correlación, corroborando en el tiempo que ello sucedía, ver figura 28. Esta práctica pudo ayudar a validar la fiabilidad del sistema embebido en el entorno de trabajo para el cual debía ser diseñado, y por ende el último de los objetivos planteados para este trabajo final.

7 TRABAJOS FUTUROS

Durante el desarrollo del proyecto se pudo vislumbrar algunas directrices para mejoras futuras del sistema embebido.

7.1 Dataset

El dataset fue tomado en la estación de Otoño, lo cual no posee comportamientos muy característicos de las variables ambientales como los que se pueden encontrar en Verano y Invierno, que seguramente sesgan al modelo LSTM de poseer la misma precisión fuera del marco de valores comprendidos en el dataset con el cual fue entrenado. Un trabajo futuro muy importante sería poder obtener el dataset de 365 días como serie de tiempo para luego re-entrenar al red LSTM y poder velar porque la misma pueda mantener su exactitud durante las 4 estaciones del año.

7.2 Firmware

Una mejora sustancial sería poder integrar el sistema embebido a una plataforma inteligente que pueda emitir alarmas, informes, y notificaciones push a dispositivos móviles, lo cual podría ser muy útil ya que para en la proyección real de uso de estos sensores inteligentes (sistemas embebidos) deberían ser usados en cantidades de 20 a 30 sensores para una plantación de hidroponia tipo. Por lo tanto una mejora en el firmware que cambie el método de comunicación de lumínica (LEDs) a radial (MQTT over TCP) disminuiría el consumo del dispositivo brindando periodo de recarga muchos mas acordes, mejoraría la integrabilidad y el volumen espacial del sistema embebido diseñado para ser usado en producción.

7.3 Hardware

El hardware es otro punto a mejorar, principalmente y en primera instancia en el diseño de un gabinete pensado para contener como para velar por el correcto funcionamiento de los componentes del sistema embebido en un ambiente de regado por aspersión en

su gran mayoría, sin que ello impida la facil utilización y recarga de energía del mismo. Otro directriz de mejora sería el PCB del sensor, el cual podría integrar directamente el circuito de medición de humedad de sustrato con la utilización de trazas co-planares, sus componentes, el sensor de variables ambientales Bosh y el microcontrolador XIAO ESP32 en cualquiera de sus versiones. Esto disminuiría la cantidad de materiales producidos por terceros, optimizando el espacio volumétrico necesario y el diseño. Luego del desarrollo final del PCB que incluya el sensor capacitivo co-planar para la humedad de la esponja fenólica de germinación será necesario obtener la curva de regresión entre la tensión de salida del integrador del sensor y la humedad relativa de la esponja de el nuevo diseño, y como segundo paso futuro adecuado sería hacer el ensayo a varios sensores producidos en serie para establecer y cuantificar si existen diferencias entre las curvas de regresión, su error y si sería necesario incorporar un proceso de calibración para no perder precisión en el modelo y el funcionamiento sistema embebido.

BIBLIOGRAFIA CONSULTADA

- [Akelah, 2013] Akelah, A. (2013). *Functionalized polymeric materials in agriculture and the food industry*. Springer.
- [Al-Chalabi and Kuo, 2018] Al-Chalabi, A. and Kuo, J. (2018). A review of hydroponic systems and its comparison with soil based systems. *American Journal of Engineering Research*, pages 56–68.
- [Al-Malaika, 2005] Al-Malaika, S. (2005). Thermoplastic molding compositions and polymer additives. US Patent 6,936,204.
- [Biases®, 2024] Biases®, W. . (2024). Lstm rnn in keras. <https://wandb.ai/ayush-thakur/dl-question-bank/reports/LSTM-RNN-in-Keras-Examples-of-One-to-Many-Many-to-One-Many-to-Many-->
- [Bolton, 2013] Bolton, W. (2013). *Mecatrónica: sistemas de control electrónico en la ingeniería mecánica y eléctrica*. Alpha Editorial.
- [Borovykh et al., 2017] Borovykh, A., Bohte, S., and Oosterlee, C. W. (2017). Conditional time series forecasting with convolutional neural networks. *arXiv preprint arXiv:1703.04691*.
- [Boyland, 2020] Boyland, C. (2020). The future of farming: Hydroponics. *Princeton University*.
- [Brownlee, 2017] Brownlee, J. (2017). *Long short-term memory networks with Python*. Machine Learning Mastery.
- [Caicedo-Rosero et al., 2021] Caicedo-Rosero, L. C., Méndez-Ávila, F. d. J., Gutiérrez-Zeferino, E., and Flores-Cuautle, J. d. J. A. (2021). Medición de humedad en suelos, revisión de métodos y características. *Pädi Boletín Científico de Ciencias Básicas e Ingenierías del ICBI*, 9(17):1–8.
- [EdgeImpulse®, 2024] EdgeImpulse® (2024). Edge ai hardware. <https://docs.edgeimpulse.com/docs/edge-ai-hardware/edge-ai-hardware#mcu>.

- [Espressif®, 2024] Espressif® (2024). Edge ai hardware. <https://www.espressif.com/en>.
- [Farahani et al., 2014] Farahani, H., Wagiran, R., and Hamidon, M. N. (2014). Humidity sensors principle, mechanism, and fabrication technologies: a comprehensive review. *Sensors*, 14(5):7881–7939.
- [Gardner et al., 1998] Gardner, C., Dean, T., and Cooper, J. (1998). Soil water content measurement with a high-frequency capacitance sensor. *Journal of Agricultural Engineering Research*, 71(4):395–403.
- [Garg et al., 2016] Garg, A., Munoth, P., and Goyal, R. (2016). Application of soil moisture sensor in agriculture. In *Proceedings of Internation Conference on Hydraulic*, pages 8–10.
- [Gers et al., 1999] Gers, F. A., Schmidhuber, J., and Cummins, F. (1999). Learning to forget: Continual prediction with lstm. *Neural Computation*, 12(10):2451–2471.
- [Glenn, 2021] Glenn, T. (2021). BME280 Libreria. <https://github.com/finitespace/BME280>.
- [Goodfellow et al., 2016] Goodfellow, I., Bengio, Y., and Courville, A. (2016). *Deep learning*. MIT press.
- [Graves et al., 2013] Graves, A., Mohamed, A.-r., and Hinton, G. (2013). Speech recognition with deep recurrent neural networks. In *2013 IEEE international conference on acoustics, speech and signal processing*, pages 6645–6649. Ieee.
- [Hochreiter and Schmidhuber, 1997] Hochreiter, S. and Schmidhuber, J. (1997). Long short-term memory. *Neural computation*, 9(8):1735–1780.
- [Hosseinzadeh et al., 2017] Hosseinzadeh, S., Verheust, Y., Bonarrigo, G., and Van Hulle, S. (2017). Closed hydroponic systems: operational parameters, root exudates occurrence and related water treatment. *Reviews in Environmental Science and Bio/Technology*, (16):659–79.
- [Iodice, 2022] Iodice, G. M. (2022). *TinyML Cookbook: Combine Artificial Intelligence (AI) and the Internet of Things (IoT) for Edge Analytics*. Packt Publishing.
- [Li et al., 2020] Li, H., Ma, Y., Zhang, X., Liu, C., and Li, Y. (2020). Advances in seed germination and early seedling growth of wheat under salt stress. *Journal of Soil Science and Plant Nutrition*, pages 332–345.
- [Ling, 2004] Ling, P. (2004). A review of soil moisture sensors. *Assn. Flor. Prof. Bull*, 886:22–23.

- [Malhotra et al., 2015] Malhotra, P., Vig, L., Shroff, G., and Agarwal, P. (2015). Long short term memory networks for anomaly detection in time series. *Proceedings of the European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning (ESANN)*, 23:89–94.
- [Robinson and Dean, 1993] Robinson, M. and Dean, T. (1993). Measurement of near surface soil water content using a capacitance probe. *Hydrological processes*, 7(1):77–86.
- [Sáez Bombín et al., 2018] Sáez Bombín, S. et al. (2018). Reconocimiento de actividades físicas con sensores iniciales y redes neuronales de aprendizaje profundo. *Universidad de Valladolid*.
- [Savvas and Gruda, 2018] Savvas, D. and Gruda, N. (2018). Application of soilless culture technologies in the modern greenhouse industry – a review. *European Journal of Horticultural Science*, 83(5):280–293.
- [Sharma, 2018] Sharma, V. (2018). Methods and techniques for soil moisture monitoring.
- [Silva et al., 2019] Silva, E. M. S., Luz, P. B., Ferreira, S. R., and Figueiredo, F. (2019). Germinação de sementes de rabanete em diferentes substratos em condições hidropônicas. *Revista Ceres*, 66(6):432–437.
- [Sánchez-García et al., 2019] Sánchez-García, P., Ocampo-Jiménez, O. R., Hernández-Montiel, L. G., Mendoza-Nazar, P., Valdez-Aguilar, L. A., Ramírez-Silva, M. T., and Camacho-Ruiz, R. M. (2019). Hydroponic agriculture: Potential limitations and future perspectives. *sustainability*, 11(2):3377.
- [TEN-LEAD®, 2024] TEN-LEAD® (2024). Hydroponic growing cube. <https://es.tenleadpf.com/plant-growing-cube/hydroponic-growing-cube/grow-cubes-for-plant-propagation.html>.
- [Tovar Macías, 2024] Tovar Macías, S. (2024). Red neuronal lstm para pronosticar la radiación solar y la temperatura ambiente en el corto plazo utilizando pronósticos meteorológicos y datos sintéticos para una planta fotovoltaica. *Universidad de los Andes*.
- [Whalley et al., 1992] Whalley, W., Dean, T., and Izzard, P. (1992). Evaluation of the capacitance technique as a method for dynamically measuring soil water content. *Journal of agricultural engineering research*, 52:147–155.
- [Wu et al., 2021] Wu, Y., Cui, L., Ma, X., Zhang, H., and Li, C. (2021). Effects of controlled release fertilizer on seedling growth and bacterial community structure in ginseng soil under continuous cropping. *Journal of Ginseng Research*, 45(2):208–216.

A CIRCUITOS ESQUEMÁTICOS

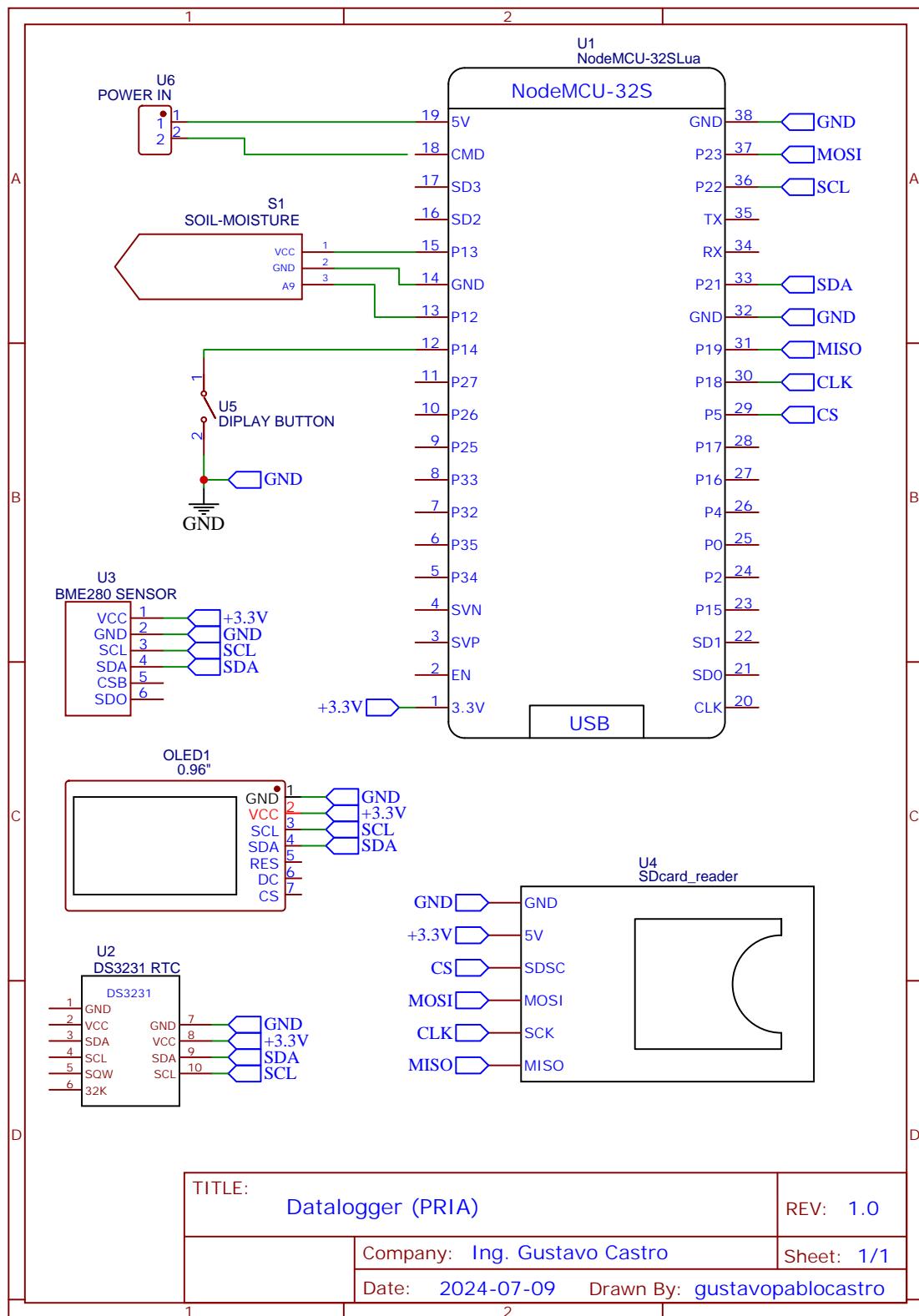


Figura 29: Circuito Electrónico del Datalogger

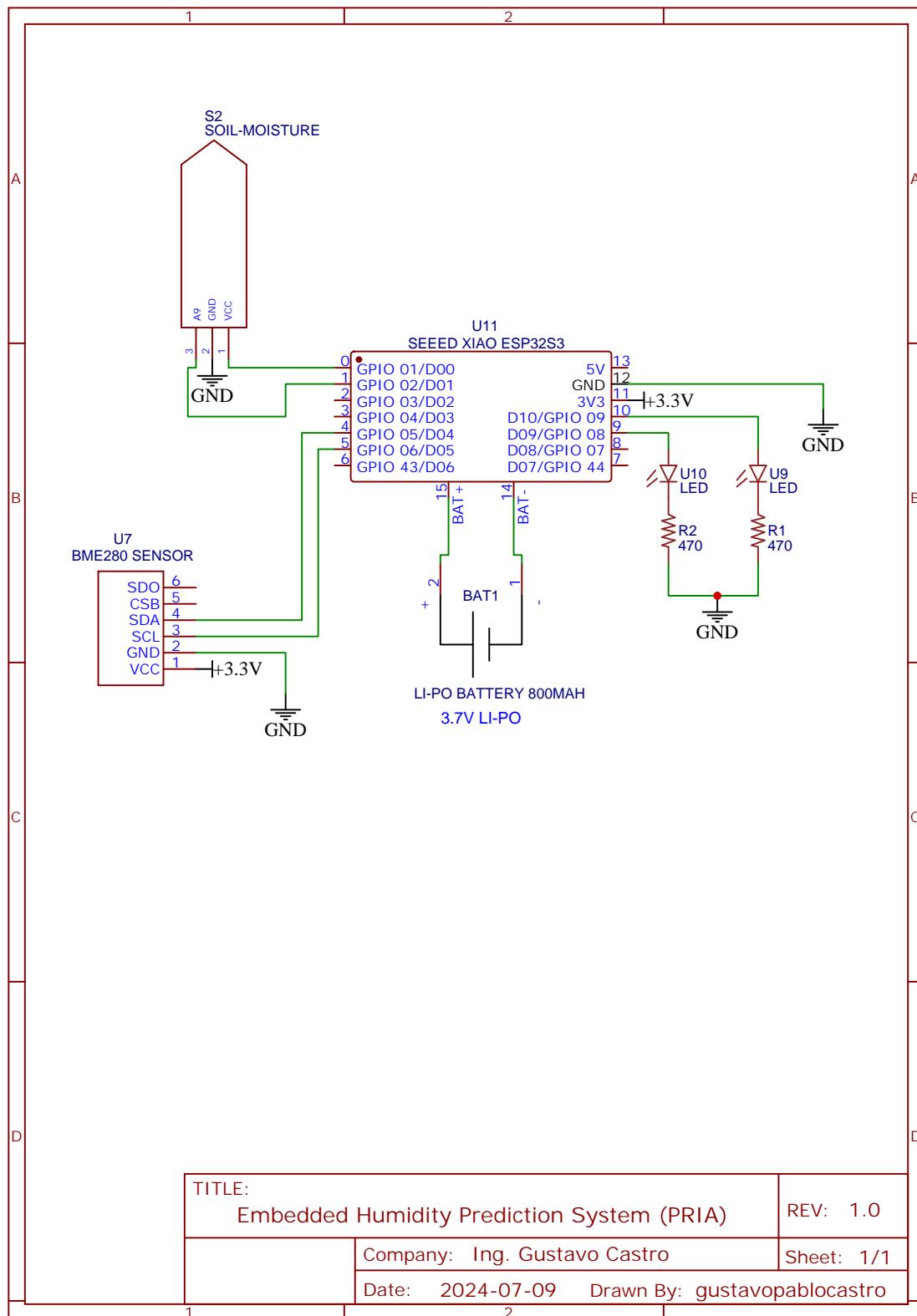


Figura 30: Circuito Electrónico del Sistema Embebido de Predicción de Humedad

B REPOSITORIO GITHUB

El presente anexo contiene toda la información relacionada al proyecto que pueda ser utilizada en un futuro por cualquier estudiante o investigador que desee continuar con este trabajo o basar uno propio con parte de este material.

Para ello se construyó un repositorio público *github*, el cual se puede acceder desde:
https://github.com/gustavopablocastro/ESP32_LSTM_Phenolic_Sponge_Moisture

En el mismo se podrá encontrar:

- El dataset recabado y utilizado en el presente trabajo
- El circuito electrónico y firmware del Datalogger (framework Arduino)
- El circuito electrónico y firmware del Sistema Embebido (framework Arduino)
- El Notebook (Python) utilizado para el pre-procesamiento del dataset, generación y entrenamiento de la RNN-LSTM y el deployment del modelo entrenado a una librería Arduino.

Estos materiales compartidos son los pilares fundamentales del presente trabajo final, cientos de horas de trabajo e investigación que deseo que pueda ayudar y aliviar cualquier trabajo futuro que pueda contribuir con una nueva mejora a nuestra sociedad y/o a nuestro entorno.