Nota de Aceptación

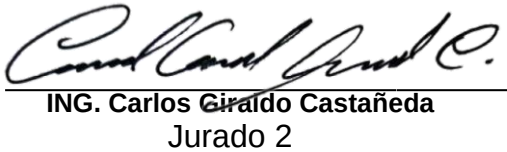
Aprobado por el Comité de Trabajo de Grado en cumplimiento de los requisitos exigidos por la Pontificia Universidad Javeriana para optar el título de Ingeniero Electrónico.

Dr. CAMILO ROCHA
Decano de la Facultad de Ingeniería y Ciencias

Director Carrera Ingeniería Electrónica.

Dr. LUIS EDUARDO TOBÓN LLANO
Director(a) Trabajo

Dr. Dimas Mavares
Jurado 1



ING. Carlos Giraldo Castañeda
Jurado 2

Acta de Correcciones al Proyecto de Grado
Ingeniería Electrónica

Fecha: 25/Abril/2021

Autor: Juan Camilo Zambrano Meza

Nombre del Proyecto de Grado: Modelamiento de la propagación inalámbrica en cultivos de arroz.

Director: Luis Eduardo Tobón Llano

Como indica el artículo 2.27 de las Directrices de Trabajo de Grado, he verificado que el estudiante indicado arriba ha implementado todas las correcciones que los Jurados del Proyecto de Grado definieron que se efectuaran, como consta en el Acta de Calificación correspondiente.

_____

Firma de Director(a) del Proyecto de Grado

Pontificia Universidad
JAVERIANA
Cali

Facultad de Ingeniería
y Ciencias
Ingeniería Electrónica

MODELAMIENTO DE LA PROPAGACIÓN INALÁMBRICA EN CULTIVOS DE ARROZ

JUAN CAMILO ZAMBRANO MEZA

TRABAJO DE GRADO

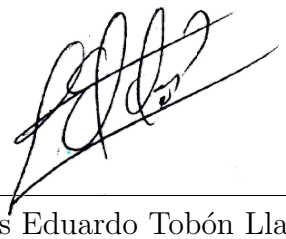
Director:
Luis Eduardo Tobón Llano

SANTIAGO DE CALI
Marzo 2021

Nota de Aceptación

Aprobado por el Comité de Trabajo de Grado en cumplimiento de los requisitos exigidos por la Pontificia Universidad Javeriana para optar al título de Ingeniero Electrónico.

Dr. Hernán Camilo Rocha
Decano Facultad de Ingeniería y Ciencias



Dr. Luis Eduardo Tobón Llano
Director Carrera Ingeniería Electrónica
Director Trabajo de Grado

Carlos Andrés Giraldo
Jurado 1

Dr. Dimas Mavares
Jurado 2

AGRADECIMIENTOS

Agradezco a mi madre y hermano por su amor y apoyo incondicional, a mis abuelos por su paciencia y sabiduría, a mis amigos por su tiempo y acompañamiento y al universo por ponerme en el camino en el que me encuentro hoy en día.

Gracias a los profesores y colaboradores de la universidad, por permitirnos tener una vida académica sin muchas complicaciones. Especial agradecimiento al profesor Luis Eduardo Tobón, por todo el acompañamiento y apoyo que me prestó para llevar a cabo esta investigación.

RESUMEN

En el ámbito de la agricultura de precisión los modelos y características de los canales inalámbricos son un recurso valioso para garantizar que una red de sensores tenga una buena comunicación. Se llevó a cabo un experimento que consiste en transmitir una señal inalámbrica a través de un cultivo de arroz variando la distancia entre antenas y su altura, en las etapas de macollamiento, alargamiento del entrenudo y producción del grano. Se comparó la variación de atenuación para diferentes distancias de transmisión y se analizó la comparación entre los valores de RSSI estimados y los medidos con base a diferentes modelos de propagación. Se usó un análisis de regresión para ajustar los modelos de espacio libre y dos rayos convencional al canal estudiado, determinar un modelo Log-distancia y plantear un modelo de dos rayos ajustado. El RE (error relativo) para los modelos de espacio libre y dos rayos se obtuvo entre 7.07 – 7.41 % y 5.34 - 28.79 % respectivamente lo que hace que estos modelos no sean adecuados para la estimación de RSSI en cultivos de arroz. Los rangos de RE (error relativo) para el modelo Log-distancia y el de dos rayos ajustado varió entre 6.06-7.01 % y 4.33-5.76 % respectivamente. El modelo Log-distancia y el de dos rayos ajustado tuvieron un mejor desempeño para modelar el RSSI de un canal inalámbrico presente en un cultivo de arroz. El modelo de dos rayos ajustado mejoró el rendimiento del modelo de dos rayos convencional obteniendo la mayor precisión entre todos los modelos estudiados. A medida que la planta se desarrolla se incrementan las atenuaciones en el canal presente a través de un cultivo de arroz y aumentar la altura de las antenas incrementa el rango de transmisión de un enlace. Esta investigación sirve como base para el modelamiento de canales inalámbricos para redes de sensores dentro de cultivos de arroz.

ABSTRACT

In the field of precision agriculture, the models and characteristics of wireless channels are a valuable resource to ensure that a sensor network has good communication. An experiment was carried out that consists of transmitting a wireless signal through a rice crop by varying the distance between antennas and their height, in the stages of tillering, jointing and grain production. The attenuation variation for different transmission distances was studied and the comparison between the estimated and measured RSSI values based on different propagation models was analyzed. Regression analysis was used to fit the free space and conventional two-ray models to the studied channel, determine a Log-distance model, and propose a fitted two-ray model. The RE (relative error) for the free space and two-ray models was obtained between 7.07 - 7.41 % and 5.34 - 28.79 % respectively, which makes these models not suitable for estimating RSSI in rice crops. The RE ranges for the Log-distance model and the adjusted two-ray model varied between 6.06-7.01 % and 4.33-5.76 % respectively. The Log-distance model and the adjusted two-ray model had a better performance to model the RSSI of a wireless channel present in a rice crop. The adjusted two-ray model improved the performance of the conventional two-ray model obtaining the highest precision among all the models studied. As the plant develops, the attenuations in the channel present through a rice crop increase and increasing the height of the antennas increases the transmission range of a link. This research serves as a basis for the modeling of wireless channels for sensor networks within rice crops.

Keywords: Precision Agriculture, Wireless Channel characteristics, Rice fields, Enviromental conditions, Two ray adjusted model, RSSI, Received power.

ÍNDICE GENERAL

ÍNDICE DE FIGURAS

ÍNDICE DE TABLAS

CAPÍTULO 1

INTRODUCCIÓN

A medida que avanzan los años la población de seres humanos crece generando que la demanda para alimentar personas crezca, por lo que se hace necesario tener los recursos suficientes para suplirla. La principal forma de obtener alimentos es mediante la agricultura, la cual se puede definir como la acción en la cual el ser humano administra los recursos naturales de una forma concreta con el objetivo de producir alimentos u otros bienes [1]; son muchas las necesidades que los humanos suplen mediante la aplicación de la agricultura, esta es una práctica esencial de la sociedad, entre más se avance en este campo se va a lograr un beneficio mucho mayor para la humanidad. Hacer eficiente la agricultura llevaría a hacer progresos en campos como el de la alimentación, algunos productos textiles e incluso en cultivos energéticos. Es aquí donde entra a jugar un papel importante la agricultura de precisión, este término hace referencia a un método que permite optimizar ciertos factores referentes a un cultivo mediante el monitoreo de distintas variables dentro y fuera de este; estas variables pueden ser condiciones climáticas, condiciones del suelo, color de la planta, técnicas de cultivo, etc. Dicho método ayuda a seguir con total exactitud el proceso de crecimiento que está llevando cada una de las plantas que hacen parte de un cultivo. El monitoreo que se realiza mediante la correcta aplicación de tecnologías de información, mencionado anteriormente, permite tener un control total sobre lo que está pasando en cualquier parte del cultivo, lo que facilita a los agricultores su tarea, brinda mayor eficiencia a la hora de corregir problemas, ayuda a preservar recursos y no malgastarlos, entre otros beneficios[2]. Para realizar un buen monitoreo se puede hacer uso de una red de sensores inalámbricos que funcione en tiempo real y permita saber en todo momento cómo las variables mencionadas están cambiando, es aquí donde se debe hacer un énfasis en lograr que la comunicación inalámbrica que se va a llevar a cabo a partir de los datos recolectados en el cultivo se transmita siempre de la forma más correcta posible.

La comunicación inalámbrica es aquella donde el mensaje que se pasa entre transmisor y receptor se realiza en un medio de propagación no guiado, es decir, se tiene un medio como el aire para realizar la comunicación. Para poder establecer esta comunicación entre dos dis-

positivos hay diferentes factores que se deben tener en cuenta: el tipo de antena que se va a usar, la localización del transmisor y receptor, la frecuencia que se va a usar para transmitir, el tipo de modulación, el medio por el cual se transmitirá la señal, entre otras. Todos estos factores determinan en mayor o menor medida la calidad de la transmisión, dado que un solo cambio en la posición de una antena, el uso incorrecto de un tipo de modulación o incluso no tener en cuenta los obstáculos del medio en el enlace puede ocasionar que la señal transmitida no contenga todos los datos que se quieren utilizar. Como se dijo antes, no son muchos los estudios que se han realizado con respecto a este tema, en [3] y [4] se realiza un modelo estadístico a partir de mediciones realizadas en zonas con una alta densidad de vegetación, en [5] verificaron si los modelos de medio existentes describen con una alta precisión la propagación de una señal a través de cultivos de papa, en [13] se plantea un modelo de dos pendientes que predice la potencia recibida en una antena, en un cultivo de arroz, para aplicaciones de WSN. Ninguno de estos estudios se realizan en un país como Colombia, cuyo entorno y dinamismo de las condiciones climáticas pueden afectar de forma directa el medio de transmisión de un enlace inalámbrico dentro de un cultivo. Es este dinamismo el cual puede causar que en un proceso de agricultura de precisión se lleve a cabo una transmisión deficiente de datos lo que llevaría a obtener resultados pobres al implementar esta técnica. Algunos de los factores que pueden llegar a afectar a la transmisión de datos dentro de un cultivo son el follaje de las plantas, las características del suelo, la humedad del entorno, etc. En un país de clima tropical como Colombia estos factores pueden ser mucho más dinámicos que en otros, de ahí la importancia de realizar un estudio como este.

En base a todo lo mencionado con anterioridad, este investigación se lleva a cabo para realizar un modelado del canal de transmisión que se encuentra en un cultivo de arroz que puede ser usado en el diseño de una red inalámbrica de sensores (WSN). Esta aplicación específica se estudia puesto que aporta al desarrollo del programa de investigación ÓMICAS, al cual la universidad Javeriana Cali hace parte, que busca contribuir a la seguridad alimentaria y una mejor producción agrícola a nivel mundial. Para lograrlo, se están realizando siete proyectos que incluyen desde el estudio genotipico de las plantas de arroz y caña hasta los impactos que estas tienen sobre el medio ambiente y viceversa. Uno de estos proyectos se denomina Fenómica el cual consiste en el desarrollo de una plataforma multimodal para el análisis fenotípico de cultivos, cuya aplicación facilita el proceso de evaluación y diagnóstico de estos mediante el uso de tecnologías de información. Se busca conocer características del suelo, planta y atmósfera para lograr cuantificar la incidencia de variables del cultivo en el medio ambiente. Teniendo en cuenta la finalidad de este programa el modelo se realizará con base a una frecuencia de operación de 433 Mhz la cual hace parte de la banda blanca, una banda que no es muy utilizada en el territorio nacional, un ancho de banda de 5 Mhz y este debe describir la potencia recibida en una antena de un enlace inalámbrico tomando como referencia la etapa de crecimiento de las plantas, la altura de las antenas y la distancia entre estas. La exactitud del modelo para predecir la potencia recibida es comparada con otros modelos ya existentes.

1.1. Objetivos

1.1.1. Objetivo General

Predecir el comportamiento de un canal inalámbrico presente a través de un cultivo de arroz mediante el planteamiento de un modelo semi-empírico para la implementación de una red de sensores inalámbricos.

1.1.2. Objetivos Específicos

- Identificar el efecto del proceso de crecimiento de las plantas, la altura de las antenas y la distancia entre estas en la propagación de una señal inalámbrica en un enlace punto a punto a través de un cultivo de arroz.
- Proponer un modelo semi-empírico que describa el comportamiento de una señal inalámbrica recibida en un enlace punto a punto a través de un cultivo de arroz.
- Evaluar la predicción de la potencia recibida en un enlace inalámbrico del modelo planteado mediante la comparación con otros modelos ya existentes.

CAPÍTULO 2

MARCO TEÓRICO

2.1. Glosario

- **Agricultura:** La agricultura se puede definir como todo aquel conjunto de actividades realizadas por los humanos para obtener recursos como verduras, frutas, hortalizas, cereales, etc. Dentro de estas actividades está el cuidado y adecuación del suelo, seguir de cerca el proceso de crecimiento de las plantas, entre otras. La agricultura es una técnica que permite la sostenibilidad alimenticia de las sociedades.[6]

- **Agricultura de precisión:** Termino que se refiere a la técnica usada en la agricultura que por medio de las tecnologías de la información logra obtener datos de un cultivo para conocer el estado del mismo, esto con el objetivo de mejorar la productividad de un cultivo(calidad y cantidad) reduciendo la dosis de insumos necesarios para que este crezca de forma saludable [2].

- **Antena:** Una antena es un dispositivo electrónico cuyo principal objetivo es recibir o transmitir señales desde o hacia el espacio libre. Todo dispositivo que lleve a cabo una comunicación inalámbrica debe hacer uso de una antena para la transmisión de datos.[9]

- **API:** Por sus siglas en inglés Application programming interface, una API se puede definir como un conjunto de procedimientos o funciones que permiten a un dispositivo de software acceder a información o servicio que otra aplicación tenga disponible[31].

- **Coeficiente de reflexión:** Coeficiente que indica la relación en cuanto amplitud de una señal incidente en un medio con la señal reflejada por el mismo. Para el caso de reflexión entre agua y aire este tiene un valor de -1. [19]

- **Comunicación inalámbrica:** Se define como comunicación inalámbrica a cualquier comunicación entre dos dispositivos que no necesite de un medio físico para llevarse a

cabo. La transmisión de datos se realiza por un medio no guiado, como el aire, y se hace uso de técnicas modulación para poder diferenciar las señales que se propagan a través del medio.[8]

- **IoT:** Internet de las cosas, por sus siglas en español, constituye la técnica por medio de la cual se conectan a internet objetos de uso cotidiano con el objetivo de gestionarlos e identificarlos.
- **RSSI:** Indicador de fuerza de la señal recibida, por sus siglas en español, es una medida que estima la calidad con la que un dispositivo recibe una señal inalámbrica.[32]
- **Monocultivo:** Un monocultivo hace referencia a, como su nombre lo indica, un cultivo cuyas plantaciones solo cuentan con una especie de plantas. Puede ser de arroz, caña, plátano, pero solo cuenta con una especie de planta. Se llevan a cabo debido a las condiciones óptimas que un lugar puede llegar a tener para que un determinado tipo de planta crezca de forma correcta y sana.[7]
- **Medio de transmisión:** Medio de transmisión hace referencia al canal que admite una comunicación entre dos dispositivos. Este medio puede ser alámbrico o inalámbrico, dependiendo de la forma como se desee transmitir los datos para aplicación que se busque llevar a cabo.[10]
- **Propiedades de las ondas:** Cuando una onda (electromagnética, mecánica, sonora) se propaga a través del espacio libre puede llegar a verse afectada por los distintos obstáculos que se encuentran en el mismo, dando origen a fenómenos como la reflexión, difracción, atenuación, refracción, etc. Estos fenómenos pueden llegar a afectar drásticamente la propagación de una señal lo que puede ocasionar problemas en la transmisión de la misma.
- **Path Loss:** Los modelos de pérdidas por trayectoria describen la atenuación de una señal a lo largo del camino de trasmisión, entre transmisor y receptor, con respecto a distintas variables como la distancia, entre otros parámetros.[11]
- **Red de sensores inalámbricos:** Una red de sensores inalámbricos se puede definir como una red conformada de un número determinado de nodos que transmiten señales de radio entre sí, esto con el objetivo de monitorear y analizar variables físicas en diferentes aplicaciones.[14]
- **SGBD:** Los Sistemas de gestión de bases de datos son el software que, al seguir un modelo de sistema de bases de datos, permiten al usuario administrarla y utilizarla, mediante el lenguaje de programación SQL (Lenguaje de consulta estructurada). [29]

2.2. Antecedentes

Un estudio que permita caracterizar un canal inalámbrico presente en un cultivo de arroz no se a realizado en Colombia, pero si existen estudios sobre cultivos y plantaciones donde

se busca conocer lo que el canal hace a una señal propagada. A continuación, se muestran las investigaciones que buscan cumplir con un objetivo parecido al definido para el proyecto que se está exponiendo en este documento:

- J.Theles et al. en su trabajo "Radio wave propagation in potato fields"(2011)[5], estudiaron la propagación de una señal a través de un cultivo de papa dadas diferentes condiciones del entorno. Se verificó qué condiciones afectan la señal, en este caso, se concluyeron dos ideas importantes, la primera expresa que el rango de transmisión de una señal inalámbrica es menor a los 10 metros si se presenta un follaje denso en el cultivo, por otro lado, se concluyo que la humedad general en el entorno ayuda a que las señales se propaguen de una manera más fácil.

- H. Klaina et al. en su trabajo "Characterization of Near-Ground Radio Propagation Channel for Wireless Sensor Network with Application in Smart Agriculture"(2017)[12], identificaron diferentes factores que afectan la propagación de una señal a través de un cultivo de caña. Se realizó un estudio generando transmisión de señales variando diferentes condiciones (altura de las antenas, tamaño de los cultivos, estado del suelo y frecuencia de trasmisión). El principal factor que se estudio es de la distancia entre las antenas, evaluando factores como el RSSI y las pérdidas por trayectoria. Los resultados a los que llegaron les permitió realizar las siguientes conclusiones: Entre más alto se ubique las antenas, se obtendrán menores pérdidas por trayectoria;la frecuencia de transmisión que mejor funciona contra las adversidades y obstáculos es la de 868 MHz, además, no se necesita demasiada potencia de transmisión con esta frecuencia; Por último, a pesar de que las pérdidas por trayectoria se ven reducidas debido a la altura, el RSSI aumenta, sin importar incluso la distancia de separación entre las antenas.

- F. Wang y K. Sarabandi en su trabajo "A Physics-Based Statistical Model for Wave Propagation Through Foliage."(2007)[3], verificaron si uno de los modelos ya planteados para predecir la transmisión a través de un bosque predice correctamente la propagación de la señal y plantearon un modelo que pueda permitir predecir la propagación de una señal de manera correcta con respecto a ciertos parámetros como la distancia de las antenas, el follaje de los árboles del bosque, entre otros. En esta investigación se concluye que el modelo ya planteado que se había verificado funciona solo en ciertas aplicaciones, debido a que es un modelo de dispersión sencilla, esto significa que no tiene en cuenta una distancia muy grande de propagación, además, pudieron verificar que el modelo planteado en el documento cumple con el objetivo de predecir el comportamiento del medio.

- Gao, Z. et al. en su trabajo "Wireless Channel Propagation Characteristics and Modeling Research in Rice Field Sensor Networks."(2018)[13] propusieron un modelo semiempírico de un canal inalámbrico presente a través de un cultivo de arroz. Este modelo era para una frecuencia de transmisión de 2.4 Ghz, dado que la aplicación que se tiene pensada para el modelo es para implementar una WSN. El modelo se realizó basándose en parámetros como la altura de las antenas del enlace, la distancia entre estas y la

etapa de crecimiento de la planta de arroz, y su finalidad es predecir las perdidas por trayectoria ocasionadas por el canal. Dadas las condiciones de crecimiento de la planta el modelo se realizo de dos pendientes debido a que se tienen 3 puntos de inflexión dentro de estas condiciones. Concluyeron que el modelo de dos pendientes es mucho más exacto y tiene mejor aplicabilidad que los que existentes.

2.3. Modelos de Propagación y Potencia Recibida

La potencia recibida para un enlace en el canal en cuestión se determinó a partir de los valores de RSSI medidos en dBm y con base a la siguiente fórmula:

$$RSSI = 10 \times \log_{10}(P_r(d)) \tag{2.1}$$

La ecuación 2.1 muestra la relación entre la potencia recibida en función de la distancia y el RSSI. La ecuación para calcular potencia recibida cambia de modelo a modelo, a continuación se describen los modelos estudiados y la ecuación con la que se puede calcular la potencia recibida para cada uno.

2.3.1. Modelo de espacio libre:

Modelo ideal que se basa únicamente en los efectos que ocasiona variar la frecuencia y la distancia a la transmisión inalámbrica dentro de un canal, entre otras asunciones. Este no considera los efectos de fenómenos como la reflexión, refracción, difracción o absorción de la señal por parte de obstáculos, como en este caso los que ocasionan el terreno donde se planta el arroz y la planta misma[16]. Si se quiere conocer la exactitud con la que un modelo caracteriza algo tan complicado como el ambiente alrededor de un cultivo de arroz utilizar el modelo de espacio libre puede servir como base, dada su caracterización ideal. La ecuación que este modelo utiliza para calcular la potencia recibida es la de Friis, como se muestra a continuación:

$$P_r(d) = P_t \times G_r \times G_t \times \left(\frac{\lambda}{4 \times \pi \times d}\right)^2 \tag{2.2}$$

Reemplazando 2.2 en 2.1 se obtiene que para el modelo de espacio libre el RSSI se puede describir por medio de la siguiente ecuación:

$$Pr(dBm) = P_t(dBm) + G_t(dBm) + G_r(dBm) + 20 \times [(\log_{10}(\lambda) - \log_{10}(4 \times \pi) - (\log_{10} d)] \tag{2.3}$$

Donde λ es la longitud de onda de la señal transmitida, P_t es la potencia transmitida en dBm, G_t y G_r las ganancias de las antenas transmisora y receptora en dBm y d es la distancia entre las antenas.El valor de P_t se aproxima por medio del método de mínimos cuadrados.

2.3.2. Modelo de dos rayos:

Modelo de dos rayos: Modelo que estima las perdidas por trayectoria de un canal basado en la altura a la que están ubicadas las antenas, la distancia entre estas y la frecuencia utilizada para transmitir. Toma en cuenta un camino de transmisión alterno al de la línea de visión entre las antenas, el cual es un reflejo en el suelo del enlace[17]. Las ondas de los dos caminos tienen diferencias en fase y la onda recibida es una mezcla entre ellas. En el caso de este experimento, el modelo tendría en cuenta las ondas que se transmiten por la línea de visión y las que se reflejen en el dosel de la planta de arroz, la distancia y la altura entre las antenas. Una mejor forma de entenderlo es por medio de la figura 2.1:

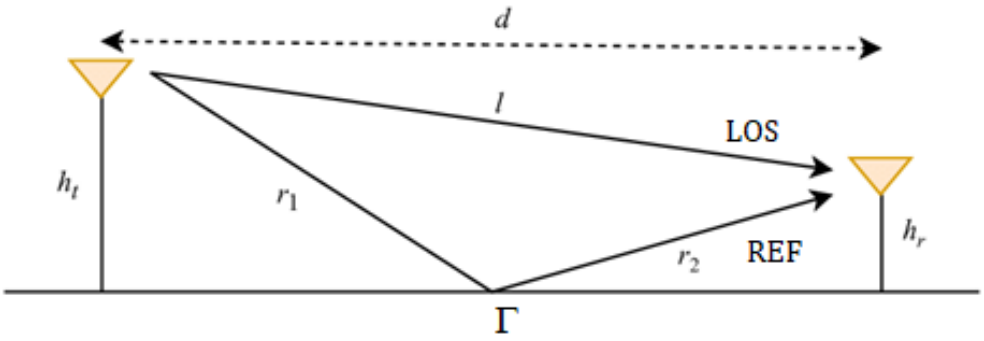


Figura 2.1: Visualización del modelo de dos rayos.

Donde l es la distancia entre la antena receptora y trasmisora, r_1 y r_2 son las distancias de estas al punto de reflexión, Γ el coeficiente de reflexión del suelo, LOS la trayectoria de linea de visión y REF la trayectoria reflejada. Con base en el enlace representado en la figura 2.1 se puede derivar la fórmula para calcular potencia recibida según este modelo.

$$P_r = P_t \left[\frac{\lambda}{4 \times \pi}\right]^2 \times \left[\frac{\sqrt{G_{los}}}{l} + \frac{\Gamma \times \sqrt{G_{ref}} \times e^{(-j \times \phi)}}{r_1 + r_2}\right]^2 \tag{2.4}$$

Siendo λ la longitud de onda de la señal transmitida, G_{los} la ganancia neta en la trayectoria LOS, G_{ref} la ganancia neta en la dirección del haz reflejado, l la distancia que viaja el rayo de transmisión LOS, $r_1 + r_2$ la distancia que viaja el rayo reflejado, Γ es el coeficiente de reflexión del suelo y ϕ es la diferencia de fase entre las dos señales recibidas. La ecuación se puede simplificar principalmente mediante geometría, proceso que se va a describir a continuación:

$$(r_1 + r_2 - l) = \sqrt{(h_t + h_r)^2 + d^2} - \sqrt{(h_t - h_r)^2 + d^2} \tag{2.5}$$

Teniendo h_t como la altura de la antena transmisora, h_r la altura de la antena receptora y d la distancia entre las antenas. Con la ecuación 2.5 se derivó la siguiente fórmula para la diferencia de fase entre señales recibidas:

$$\phi = \frac{2 \times \pi \times (r_1 + r_2 - l)}{\lambda} \tag{2.6}$$

Si se reemplazan 2.5 y 2.6 en 2.3 y se asume que la altura de las antenas es despreciable en comparación a la distancia que hay entre estas, que las ganancias de las antenas son iguales y que el coeficiente de reflexión del suelo es -1, la fórmula queda reducida a lo siguiente:

$$P_r \approx P_t \times \left[\frac{\lambda \times \sqrt{G}}{4 \times \pi \times d}\right]^2 \times \left[\frac{4 \times \pi \times h_r \times h_t}{\lambda \times d}\right]^2 = \left[\frac{\sqrt{G} \times h_r \times h_t}{d^2}\right]^2 \times P_t \tag{2.7}$$

Reemplazando 2.8 en 2.1 se obtiene que la formula para estimar el RSSI basado en la potencia recibida para el modelo de dos rayos es:

$$P_r(dBm) = P_t(dBm) + G(dBm) + 20 \times (\log_{10}(h_r \times h_t)) - 40 \times \log_{10}(d) \tag{2.8}$$

En este caso, la fórmula para el RSSI derivada del modelo de dos rayos depende solo de la ganancia de las antenas, la distancia entre ellas y altura de estas, asumiendo que son distintas. El valor de P_t se aproxima por medio del método de mínimos cuadrados.

2.3.3. Modelo Log-distancia:

Basado en el modelo de espacio libre, el modelo Log-Distance expresa las pérdidas por trayectoria de un canal como una ecuación lineal de la distancia entre antenas tomada logarítmicamente y un índice que muestra el coeficiente de las pérdidas por trayectoria de este, denominado índice de las perdidas por trayectoria. Este índice varía según el medio y afecta la señal en la antena receptora; si hay reflexión o refracción, cuanta intensidad tiene el enlace de punto de vista, etc [18]. Por lo general debido a la presencia de señales planas con diferentes ángulos y fases de llegada en la antena receptora se genera atenuación o distorsión de la señal transmitida. Este modelo tiene en cuenta cambios como diferencias en las alturas de las plantas de arroz presentes en el canal y en la cobertura del dosel de estas para garantizar el representar de la mejor forma lo que estos generan a la transmisión. Así, la fórmula para calcular las pérdidas por trayectoria a partir de este modelo es la siguiente:

$$\begin{gathered} P_L = P_{L(d0)} + 10 \times n \times \log_{10}\left(\frac{d}{d_0}\right) + \chi, \\ d_f \leq d_o \leq d \end{gathered} \tag{2.9}$$

Donde $P_{L(d0)}$ representa las pérdidas por trayectoria calculadas para una distancia de referencia, n es el índice de pérdidas por trayectoria, χ es una variable aleatoria Gaussiana con media cero y con desviación estándar σ, utilizada únicamente cuando se desean estudiar los efectos de ensombrecimiento en el canal. Cabe aclarar que esta ecuación es para calcular las perdidas por trayectoria para distancias ubicadas en el campo lejano de la antena transmisora, lo que está representado por d_f, d_0 es la distancia de referencia y $P_{L(d0)}$ son las pérdidas por trayectoria de referencia (distancia de referencia cercana). Este es calculado a partir del uso de la fórmula de las pérdidas por trayectoria basada en el modelo de Espacio libre mencionada anteriormente o tomando directamente las medidas en campo para esta distancia que, en el

caso de que el enlace sea una microcelda como la que se está usando en el experimento, puede ser de 1 metro a 10 metros. Simplificando un poco la ecuación 2.9 se obtiene que:

$$P_{L(d)} = K + 10 \times n \times \log_{10}(d) \tag{2.10}$$

Donde K es la suma de $P_L(d0)$ y χ, n es el índice de pérdidas por trayectoria, el cual estará relacionado en este estudio con la altura de las plantas de arroz, el ambiente de alrededor del cultivo y la cobertura del dosel de estas. Los valores de K y n se aproximan por medio del método de mínimos cuadrados.

2.3.4. Modelo de dos rayos ajustado:

Las distintas condiciones ambientales que un cultivo de arroz posee hacen que la caracterización de un canal inalámbrico a través del mismo sea una tarea complicada. Para estos casos incluso como esté polarizada la antena influye de manera drástica en la caracterización. La rugosidad del suelo de un cultivo de arroz y su tamaño y las dimensiones del sistema usados para hacer el experimento(altura y distancia) presentan escala similar; dada la longitud de onda de la señal transmitida y lo mencionando con anterioridad para el caso de esta investigación el efecto de polarización de la antena no tiene tanta relevancia en el modelamiento, debido a que se producen distintos fenómenos de propagación en el canal, no solo una reflexión especular. Como se explicó anteriormente, el modelo de dos rayos asume ciertos parámetros para simplificar la fórmula de potencia recibida por la antena receptora, pero presenta una dependencia a la altura a la que estén las antenas, la distancia entre estas y al medio en el cual se está reflejando la señal transmitida; de este modo, la fórmula para calcular las perdidas por trayectoria se deriva de la ecuación 2.4, con la notación que se muestra a continuación:

$$\begin{gathered} P_r = P_t \times \left[\frac{\lambda}{4 \times \pi}\right]^2 \times \left[\frac{\sqrt{G_{los}}}{d_{los}} + \frac{\Gamma \times \sqrt{G_{ref}} \times \mathrm{e}^{(-j \times \phi)}}{d_{ref}}\right]^2 \\ d_{los} = \sqrt{d^2 + (h_t - h_r)^2},\ d_{ref} = \sqrt{d^2 + (h_t + h_r)^2},\ \phi = \frac{2 \times \pi \times (d_{ref} - d_{los})}{\lambda} \end{gathered} \tag{2.11}$$

Si se desea adecuar el modelo al medio que se quiere estudiar en este experimento no se pueden tener las mismas consideraciones mencionadas con anterioridad; por lo tanto, para este caso se toma que $h_t = h_r$, $G_{los} = G_{ref}$ y Γ es el parámetro que variará según haya un cambio en la altura de la planta del arroz y en la cobertura del dosel de estas, no se asume que $\Gamma = -1$. Con estas asunciones las ecuaciones quedan de la siguiente forma: $d_{los} = d,\ d_{ref} = \sqrt{d^2 + 4 \times h^2},\ \phi = \frac{2\pi \times (\sqrt{d^2 + (4 \times h^2)} - d)}{\lambda}$

Si se toman $A = \mathrm{e}^{(-j \times \phi)} = \cos(\phi) - j\,\mathrm{sen}(\phi)$ y $B = \sqrt{d^2 + (4 \times h^2)}$ se puede decir que la potencia recibida para el modelo de dos rayos ajustado es expresada mediante la siguiente ecuación:

$$P_r = P_t \times \left[\frac{\lambda}{4\pi}\right]^2 \times (G) \times \left|\left(\frac{1}{d}\right) + \left(\frac{\Gamma \times A}{B}\right)\right|^2 \tag{2.12}$$

Dado que A es un número con un exponencial complejo, se procede a multiplicar este número por su complejo conjugado, para eliminar la parte imaginaria y contar con todo lo que aporta la diferencia de fase entre señales recibidas ($\Delta\phi$) en la estimación de la potencia recibida, como se muestra a continuación:

$$P_r = P_t \times \left[\frac{\lambda}{4\pi}\right]^2 \times (G) \times \left(\left[\frac{1}{d} + \frac{\Gamma \times e^{(-j\times\phi)}}{B}\right] \times \left[\frac{1}{d} + \frac{\Gamma \times e^{(j\times\phi)}}{B}\right]^*\right) \tag{2.13}$$

En la ecuación 2.13 el $*$ denota el complejo conjugado del término que incluye el número A. Realizando esta operación se obtiene entonces que la potencia recibida para el modelo de dos rayos ajustados se puede calcular mediante la ecuación que se muestra a continuación:

$$P_r = P_t \times \left[\frac{\lambda}{4\pi}\right]^2 \times (G) \times \left[\frac{1}{d^2} + \frac{2 \times \Gamma \times \cos(\phi)}{d \times B} + \frac{\Gamma^2}{B^2}\right] \tag{2.14}$$

Finalmente, reemplazando 2.15 en 2.1 se obtiene que la estimación del RSSI para el modelo de dos rayos ajustado se puede realizar mediante la ecuación:

$$\begin{aligned} P_r(dBm) = P_t(dBm) + 20 \times \log_{10}\left[\frac{\lambda}{4\pi}\right] + (G(dBm)) \\ +10 \times \log_{10}\left[\frac{1}{d^2} + \frac{2 \times \Gamma \times \cos(\phi)}{d \times B} + \frac{\Gamma^2}{B^2}\right] \end{aligned} \tag{2.15}$$

Debido a que se quiere lograr la mayor exactitud posible en el modelo propuesto, los valores de Γ y P_t se aproximarán mediante el método de mínimos cuadrados.

2.3.5. Métricas para la evaluación de modelos

El RMSE (por sus siglas en español error de media cuadrática) y RE(en español error relativo) fueron usados como indicadores de precisión para los modelos de propagación y potencia recibida. A medida que estos valores se acercan a 0 la precisión de la estimación incrementa. Las formulas para calcularlos son las siguientes:

$$RMSE = \sqrt{\frac{1}{n} \times \sum_{n=1}^{n} (P_{R(di)} - P_{RE(di)})^2} \tag{2.16}$$

$$RE = \sqrt{\frac{1}{n} \times \sum_{n=1}^{n} \frac{(P_{R(di)} - P_{RE(di)})^2}{P_{RE(di)}}} \tag{2.17}$$

Donde n son el numero de datos experimentales obtenidos, $P_{R(di)}$ son los valores de potencia medidos y $P_{RE(di)}$ los estimados por los modelos.

CAPÍTULO 3

DISEÑO E IMPLEMENTACIÓN

3.1. Requerimientos del experimento

Teniendo en cuenta que la finalidad de este proyecto de grado es encontrar un modelo semi empírico de un canal inalámbrico presente a través de un cultivo, se diseñó un experimento que permitiera relacionar la altura y la distancia entre dos antenas con el RSSI de una señal piloto transmitida entre estas. Es importante variar la altura del enlace y la distancia entre antenas dado que las plantas dentro del cultivo de arroz van a cambiar de tamaño considerablemente a medida que se produce su proceso de maduración, lo que modifica la cantidad de obstáculos presentes a través del canal. Para poder realizar un procesado y análisis de los datos que garantice buenos resultados en el modelado de un canal se necesita que la transmisión se realice de forma fiable y que el almacenamiento de los datos pueda hacerse en tiempo real; para esto se debe hacer uso de una base de datos que permita almacenar valores de distancia, altura y el RSSI de la señal transmitida; dado lo mencionado con anterioridad, el sistema diseñado hace uso de transmisión de radio punto a punto y también comunicación Wi-Fi para conectarse a una base de datos en un servidor local.
El sistema esta conformado por un transmisor, un Arduino Uno y un módulo LoRa sx1278, y un receptor, conformada por un NodeMCU v3 y un módulo LoRa sx1278; mediante el uso de una estructura que se detallará más adelante se realiza la variación de la distancia entre el receptor y transmisor y de la altura en la que se desee colocar estos. Se debe hacer uso de alimentación portable debido al lugar (cultivo de arroz) sobre el que se va a realizar el análisis.

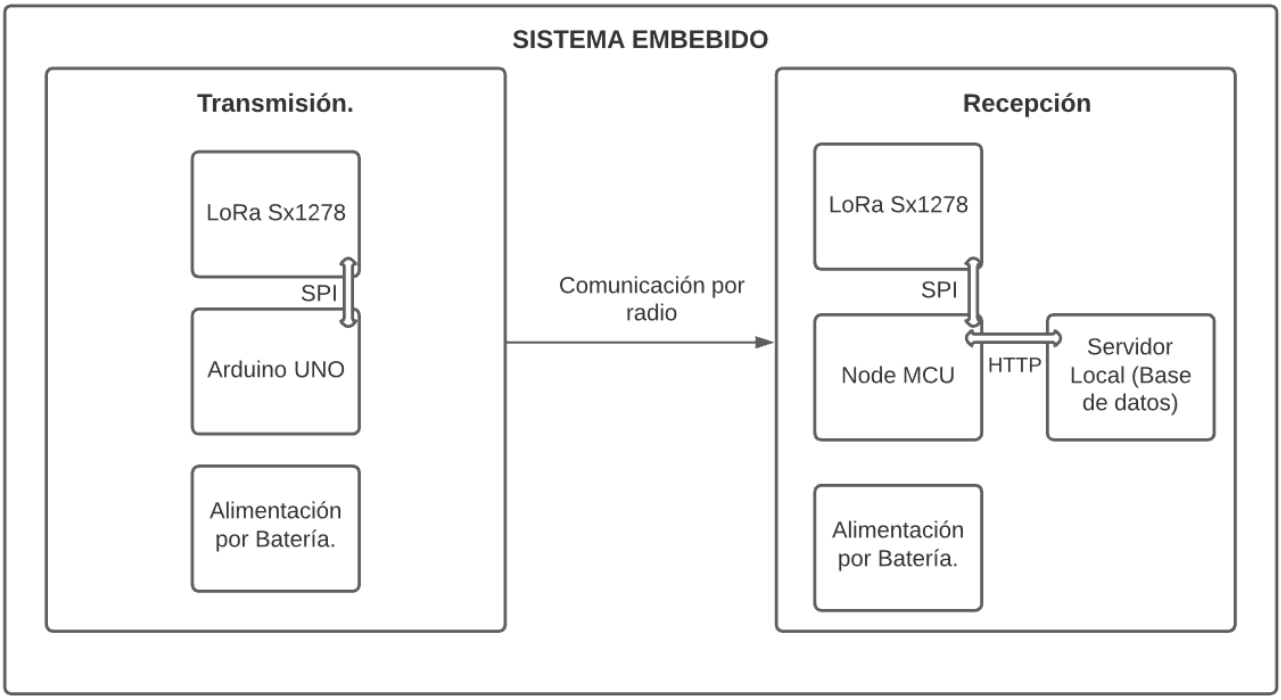


Figura 3.1: Diagrama de bloques del sistema embebido propuesto

El termino sistema embebido hace referencia a un sistema de computación que se diseña con el objetivo de cumplir con ciertas actividades específicas, a diferencia de otros dispositivos que cumplen un rango más variado de propósitos [15]. Por lo general se conforman de una placa base, en la que van incluidos la mayoría de los componentes, y módulos, los cuales tienen diferentes características y amplían el campo de acción de esta. Estos sistemas pueden ser programados por medio de diferentes lenguajes, tanto el propio del microcontrolador que se encarga de procesar en la placa base como otros de uso más general. Así, para el uso de un sistema embebido se deben tener en cuenta factores como la mantenibilidad de este, la eficiencia en cuanto a consumo de energía, tamaño, peso del dispositivo dependiendo de la aplicación, el tamaño del código y la confiabilidad, dado que se les usa para realizar tareas específicas las cuales en algunas ocasiones pueden determinar resultados en procesos más grandes.

3.2. Diseño y toma de decisiones

Para el sistema embebido se tuvieron en cuenta ciertos criterios en la selección de los microcontroladores, el tipo de comunicación a usar y el almacenamiento de datos. Estos se explican mejor con base a las tablas de comparación entre dispositivos escogidos:

COMUNICACIÓN INALAMBRICA (RADIO)						
Dispositivo	RSSI	Ancho de banda	Consumo	Robustez y Escalabilidad	Alcance	Total
LoRa	4,5	4,5	5	4,7	5	4,715
ZigBee	5	5	4,5	4	3,5	4,425
WiFi	3	4,5	5	3,5	2,5	3,75

Tabla 3.1: Tabla de comparación de dispositivos para la comunicación inalámbrica

De la Tabla 3.1 se puede inferir que los criterios utilizados para escoger el tipo de comunicación inalámbrica fueron la capacidad para conocer el RSSI en el módulo, el ancho de banda, el consumo energético del módulo, la robustez y escalabilidad de una red basada en este tipo de comunicación y el alcance esta. De los tres dispositivos el que obtuvo un mayor puntaje fue el LoRa, dado que da la facilidad al usuario de medir el valor del RSSI para cada uno de los paquetes que se envíen a través de un enlace, el ancho de banda de transmisión lo hace ideal para la aplicación pensada (transmisión de datos medidos por sensores), tiene un consumo energético reducido en comparación a las otras tecnologías y la facilidad con la que se puede establecer un enlace permite pensar en establecer redes de sensores inalámbricas robustas y escalables.

MICROCONTROLADOR				
Dispositivo	Protocolos disponibles	Corriente I/O	Alimentación	Total
UNO	5	4,5	5	4,8
Nano	5	3	5	4,2
NodeMCU	5	4,5	5	4,8

Tabla 3.2: Tabla de comparación de dispositivos para actuar como microcontrolador

De la Tabla 3.2 se deduce que para escoger microcontrolador a usar los criterios a tener en cuenta fueron los protocolos disponibles, la corriente que los pines de Entradas/Salidas llegan a suplir y la alimentación que requieren para funcionar. Dos de estos microcontroladores fueron escogidos, el arduino UNO y el Node MCU v3, dado que cumplen con las características de la misma forma. Con respecto a los protocolos disponibles los tres tienen soporte para SPI, protocolo usado por el módulo de transmisión. El problema con el Nano se presento en la corriente que suplen sus pines de Entradas/Salidas, no alcanza a suplir la corriente necesaria para que el módulo LoRa funcione correctamente como transmisor.

ALMACENAMIENTO DE DATOS				
Dispositivo	Capacidad de	Tiempo real	Manejo/Imple	Total
Base de datos	5	5	4	4,75
Módulo SD	4,5	5	2,5	4,2
EEPROM	3,5	2	4	3,025

Tabla 3.3: Tabla de comparación de dispositivos para el almacenamiento de datos

De la Tabla 3.3 se concluye que en cuanto almacenamiento de datos, para el caso de este sistema embebido los criterios de selección fueron la capacidad de almacenamiento, si este se puede hacer en tiempo real y que hace falta para que el dispositivo o técnica funcione correctamente. El mayor puntaje lo obtuvo el almacenamiento de datos por medio de las bases de datos, dado a que tiene una capacidad de almacenamiento proporcional al servidor en el que se encuentren alojadas, los datos se pueden guardar en tiempo real mediante el uso de protocolos de internet y su implementación es enteramente virtual a excepción de un módulo que permita conectar al sistema embebido con el servidor. El módulo SD usa el mismo protocolo de comunicación que el módulo de comunicación inalámbrica, por lo que no se puede implementar en el mismo microcontrolador, la EEPROM de un microcontrolador no ofrece la misma capacidad de almacenamiento que los otros dos dispositivos, además, no se pueden almacenar datos en tiempo real.

Tecnología LoRa

Para una aplicación como una red de sensores, la comunicación por radiofrecuencia es una herramienta que ofrece varias ventajas, entre ellas el amplio ancho de banda y los costos relativamente bajos a los que está sujeto un sistema de transmisión de este tipo. Aunque hay varias de estas tecnologías que se pueden usar en estas aplicaciones es LoRa, de Semtech, una de las mejores opciones.

Figura 3.2: Módulo LoRa SX1278 Ra-02.

El módulo escogido (ver Figura3.2) logra ser óptimo para aplicaciones que requieran un alto rango (2 y 15 km) y que presenten una gran robustez, gracias a que cuentan con un

modem que, usando la técnica de modulación patentada por el fabricante del dispositivo, logra garantizar una sensibilidad de -148 dBm y posee un amplificador LNA (De bajo ruido) de -20 dbM. En consecuencia, para aplicaciones de IoT, automatización, seguridad, entre otras, la alternativa de usar esta tecnología llama la atención, principalmente por el nivel de optimización que se le da al consumo energético, lo que permite aprovechar y alargar el ciclo útil de las baterías usadas para alimentar el sistema embebido que se diseñe[20].El módulo opera en las bandas ISM(Industrial, Scientific and Medical) disponibles a nivel mundial, teniendo diferentes valores según la ubicación global[21], en este caso se usa la frecuencia de 433 MHz. En cuanto a consumo de corriente, cabe mencionar que el módulo puede llegar a consumir 12mA cuando se encuentra en modo recepción y 0mA cuando se encuentra modo sleep lo que lo hace, como se mencionó antes, optimizar la energía que se le administra. Facilita al usuario el análisis del RSSI y otras características de la recepción de la señal gracias a que cuenta con Verificación por redundancia cíclica (CRC).
La comunicación entre en módulo LoRa y el microcontrolador que se quiera usar se hace por SPI (Serial peripherical interface) el cual es un protocolo de comunicación serial sincrónico full duplex, basado en una interfaz de maestro-esclavo. Para usar el protocolo, se debe definir uno de los dispositivos como esclavo y el otro maestro, para que este último sea el que controla la transferencia de la información mediante la generación de señales como la de control y reloj[22]. En el caso de LoRa, este protocolo se hace con una conexión de 4 líneas, siendo más popular que la conexión de tres. Estas cuatro líneas son:

- **NSS:** Generada por el maestro. Permite seleccionar a que dispositivo esclavo se le va a transmitir. Permite hacer uso de topografías con múltiples esclavos conectados a un maestro.
- **MISO:** Transfiere datos del esclavo al maestro.
- **MOSI:** Transfiere datos del maestro al esclavo.
- **SCLK:** Generada por el maestro. Señal de reloj para sincronizar la transferencia

Adicional a esta conexión, a nivel lógico se requiere la configuración de ciertos valores para el correcto funcionamiento del protocolo, una velocidad de la señal del reloj menor o igual a 10Mbit/s, la polarización de la misma mediante el bit CPOL y la fase mediante el bit CPHA. De este manera, para que el módulo LoRa funcione como esclavo se debe asignar 0 a los bits CPOL y CPHA , esto corresponde al modo 0 del protocolo SPI, habiendo 4 de estos con las distintas combinaciones entre los bits mencionados anteriormente[23].

Por lo tanto, para utilizar el módulo basta con realizar la conexión mostrada en la figura 3.3. Lo último a tener en cuenta es la conexión de la antena al módulo, puesto que sin esta no va a servir. Se utilizó una antena dipolo tipo Ipex con una ganancia de -5dBi para una frecuencia de 433 Mhz.

Módulo wifi y sistema receptor

Dado que se escogió almacenar los datos del experimento en una base de datos, se hizo

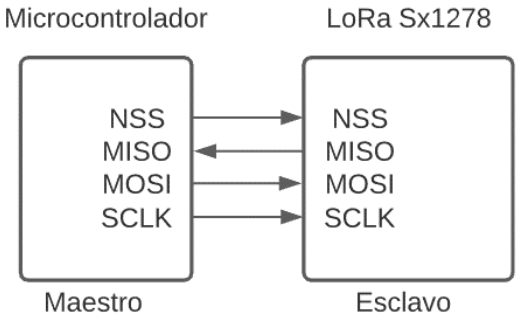


Figura 3.3: Conexión entre maestro y esclavo del protocolo SPI.

necesario hacer uso de un módulo Wi-Fi para conectarse a una API alojada en un servidor local. Se propone en primer lugar el uso del ESP8266 de Espressif, pero debido a disponibilidad de pines en el uno de los microcontroladores y la dificultad para programarlo por medio de comandos AP hizo que se buscara una segunda opción, en este caso, el NodeMCU v3, una tarjeta de desarrollo basada en el ESP8266, que, mediante el uso de ciertas librerías permite al usuario programar este último de forma sencilla, sin usar comandos AP directamente. En la Figura 3.4 se muestra la placa de desarrollo escogida:

Figura 3.4: Tarjeta de desarrollo Node MCU v3.

La placa NodeMCU cuenta con un microprocesador del fabricante Tensilica, Xtensa Lx106, que funciona a un voltaje de 3.3V y con una frecuencia de 80MHz. Con respecto a los pines y al funcionamiento general, la tarjeta cuenta con 16 entradas/salidas digitales y una sola entrada analógica, además, soporta protocolos como el SPI, UART y I2C. Esta tarjeta funciona con alimentación por cable USB o directamente con una batería, soportando voltajes de entrada entre 7 y 12 V, y se puede programar mediante el IDLE de Arduino. Para finalizar, gracias a los 128 Kb de memoria RAM y 4MB de memoria flash, tecnologías

Wi-Fi y Bluetooth incluidas internamente y modo de sueño profundo, NodeMCU es una de las mejores opciones para llevar a cabo proyectos de IoT.

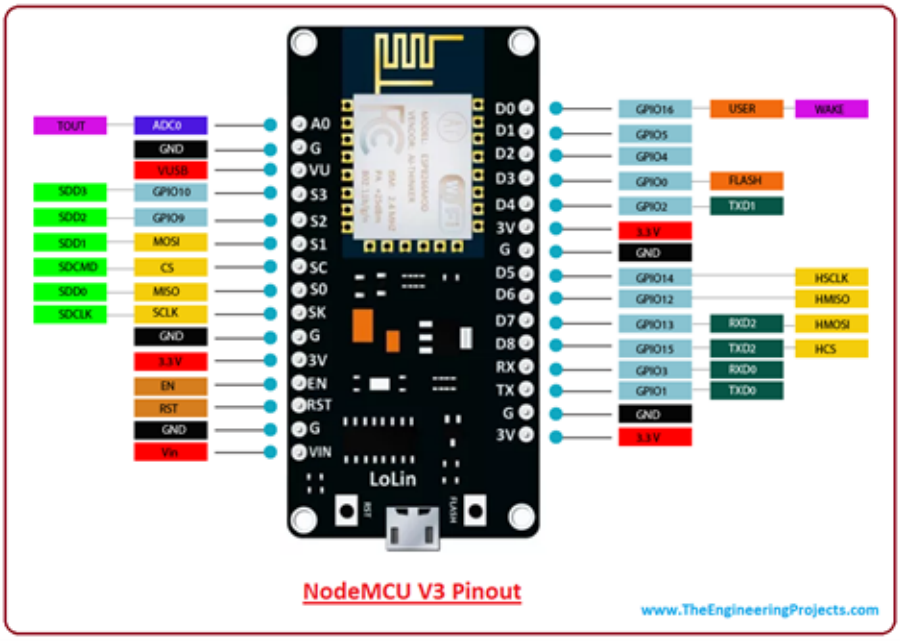


Figura 3.5: Diagrama de pines tarjeta Node MCU v3.

Como se evidencia en la figura 3.5, los pines que se deben usar para realizar la conexión SPI entre el dispositivo de radio escogido y el NodeMCU son las entradas/salidas digitales 12, 13, 14 y 15 [24].

Sistema transmisor

En aplicaciones donde se requiera hacer uso de sistemas embebidos y microcontroladores, Arduino, sin lugar a duda, es uno de los fabricantes más mencionados. En este caso se hace uso de la placa Arduino UNO R3. Esta tarjeta hace uso del microcontrolador ATmega328P de Atmel, que utiliza un voltaje nominal de 5V y una frecuencia de 16 Mhz. Permite al usuario hacer uso de 14 entradas/salidas digitales pudiendo configurar 6 de estas como PWM, 6 entradas analógicas, permite comunicación SPI, UART y I2C. Para la alimentación, funciona con voltajes de 7 a 12 V, ya sea externamente o mediante el cable USB utilizado para su programación, la cual se hace desde el software del fabricante (Arduino IDLE)[25].

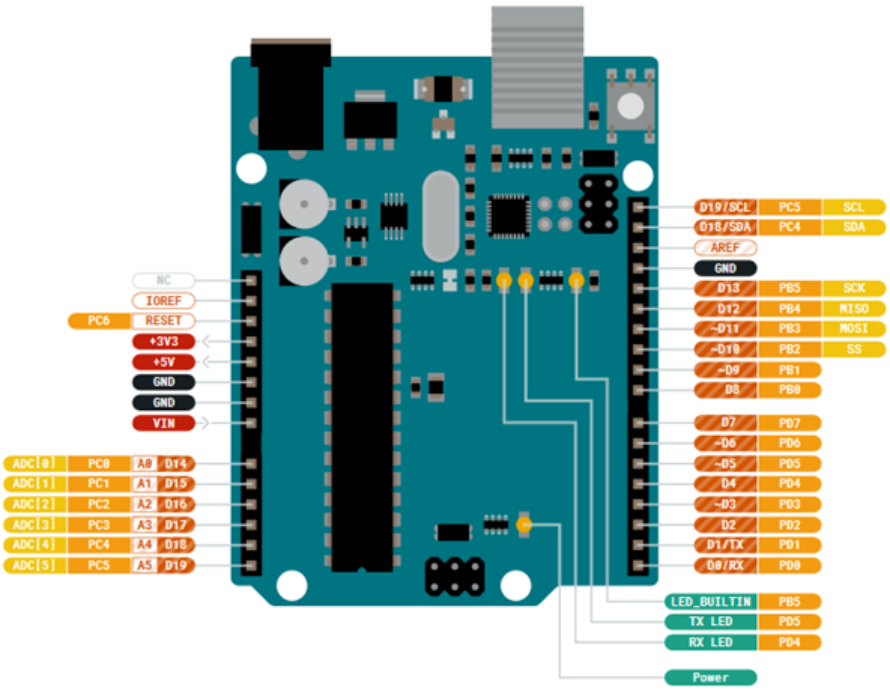


Figura 3.6: Diagrama de pines tarjeta Arduino UNO.

En el caso de esta tarjeta, la conexión SPI se realiza a los pines digitales D10, D11, D12 y D13 [26], lo cual se evidencia mejor en su diagrama de pines (ver Figura 3.6).

Almacenamiento de datos

Dado que la finalidad el experimento es poder analizar los datos de forma correcta se debe utilizar una tecnología que permita la manipulación de una gran cantidad de datos y que los almacene de forma integra, además, debe ser un método que permita el almacenamiento en tiempo real. Así, en un principio se pensó en hacer uso de la memoria interna de los microcontroladores usados, la EEPROM[27]; El problema surge a partir de la capacidad de almacenamiento pequeña en comparación a otras tecnologías (NodeMCU 512 bytes y Arduino Uno 1 Kb), además de permitir al usuario hacer un número limitado de ciclos de lectura y escritura, número al que se puede llegar fácilmente si se tiene en cuenta la velocidad de procesamiento de un microcontrolador. Lo que nos indica que sería buen almacenamiento, pero de forma duradera, no momentánea y veloz, por ejemplo, para guardar el nombre y la contraseña de una red Wi-Fi que se quiera usar en un proyecto.
Teniendo en cuenta lo anterior, se encontró la segunda opción, hacer uso de un módulo SD y mediante una memoria de este tipo almacenar los datos del experimento y sencillamente extraerlos de la misma para su posterior procesado y análisis. El problema con esta alternativa surge a partir del protocolo de comunicación que utiliza, siendo este el SPI, igual que el módulo LoRa. Para hacer uso de este módulo en este caso habría que conectarlo en la parte receptora del sistema, que es donde se calcula el RSSI de la señal enviada, lo que no dejaría

pines para realizar la conexión del módulo de radio. Por esta razón no es viable utilizar este dispositivo. En la Figura 3.7

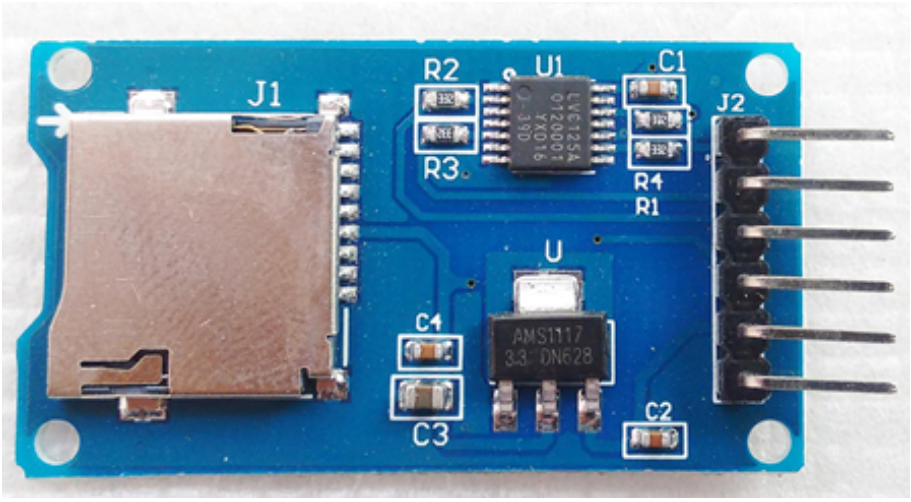


Figura 3.7: Módulo micro SD Catalex.

Finalmente se decidió realizar una base de datos. Esta alternativa hace una organización eficiente de los datos (Tablas), permite guardar datos en tiempo real, hacer uso de una capacidad elevada de almacenamiento y tener fácil acceso a la información con la que se quiera trabajar. Así, las bases de datos no son más que un conjunto de datos estructurado bajo un modelo que pueden llegar a reflejar la relación entre estos. Esta estructura permite ser modificada y leída por varios usuarios o uno solo, pero siempre manteniendo la integridad y seguridad entre cada interacción con la misma. Se pueden almacenar distintos tipos de variables, lo que permite caracterizar entidades y atributos de la vida real[28]. Este es un concepto que, aunque se escucha mucho, tiene un grado de complejidad el poder implementarlo. No basta solo con tener una base de datos, si se quieren modificar los datos de la misma de manera remota y sin necesidad de tener que hacer uso directamente del software que las gestiona hace falta el alojar esta en un servidor de internet.
Lo único que se debe garantizar en el sistema para hacer uso de esta alternativa es un módulo Wi-Fi capaz de conectarse a la base de datos por medio del servidor mencionado (por esta razón se escogió la tarjeta NodeMCU). Para el caso de este sistema, se creó el servidor local usando un computador portátil. Por lo tanto, se deben hacer uso de conceptos como Interfaz de Programación de Aplicaciones (API por sus siglas en inglés) y Sistema Gestor de Bases de Datos (SGBD), los cuales ayudan a hacer uso de esta herramienta de una forma correcta y eficiente según la necesidad que se tenga. En este caso se escogió el sistema gestor MYSQL y el framework para desarrollar APIs Django, basado en el lenguaje de programación Python. En el apartado de anexos se explica más a fondo como se logró almacenar los datos del experimento realizado en una base de datos.

3.3. Implementación y evaluación

Habiendo descrito los dispositivos y métodos utilizados que conforman el sistema, se pasará a explicar la estructura e implementación del mismo para poder llevar a cabo el experimento de medir la potencia recibida a través de un canal de arroz de forma correcta. Según lo diseñado en el experimento las antenas se colocaron cada una en un poste que permite cambiar la altura a las que estas están y al mismo tiempo la distancia entre las mismas. Por otro lado, dadas las condiciones ambientales que caracterizan un cultivo de arroz (suelo húmedo y fangoso, plantas con follaje denso, entre otras) y pensando en hacer el experimento lo más sencillo posible, se decidió poner el sistema transmisor y receptor en cajas que protegen los dispositivos de las condiciones ambientales. Las conexiones entre dispositivos usadas para el transmisor se muestran en la figura 3.8

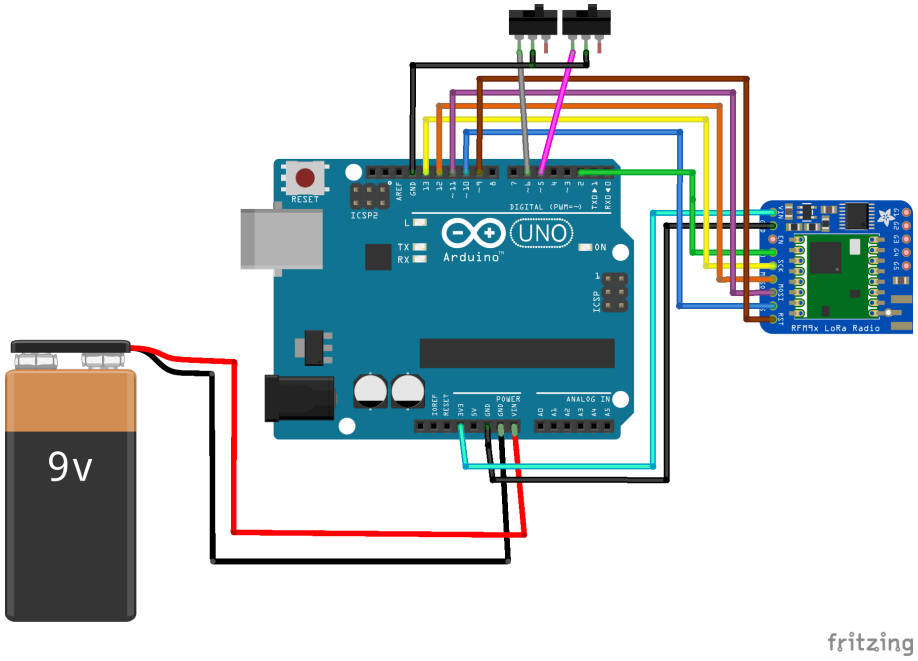


Figura 3.8: Conexiones del sistema transmisor para el experimento

Debido a que una antena en modo transmisión consume más corriente que si estuviera en recepción, el módulo LoRa que actúa como transmisor se conectó al Arduino Uno. Se alimenta el Arduino UNO con una batería de 9 V y este a su vez alimenta el módulo LoRa con un voltaje de 3.3V, la comunicación entre estos se hace por medio del protocolo SPI. Se utilizó un interruptor para iniciar el envío de los paquetes y el otro para variar un contador que representa la distancia en el sistema. Para el receptor se realizaron las conexiones mostradas en la figura 3.9

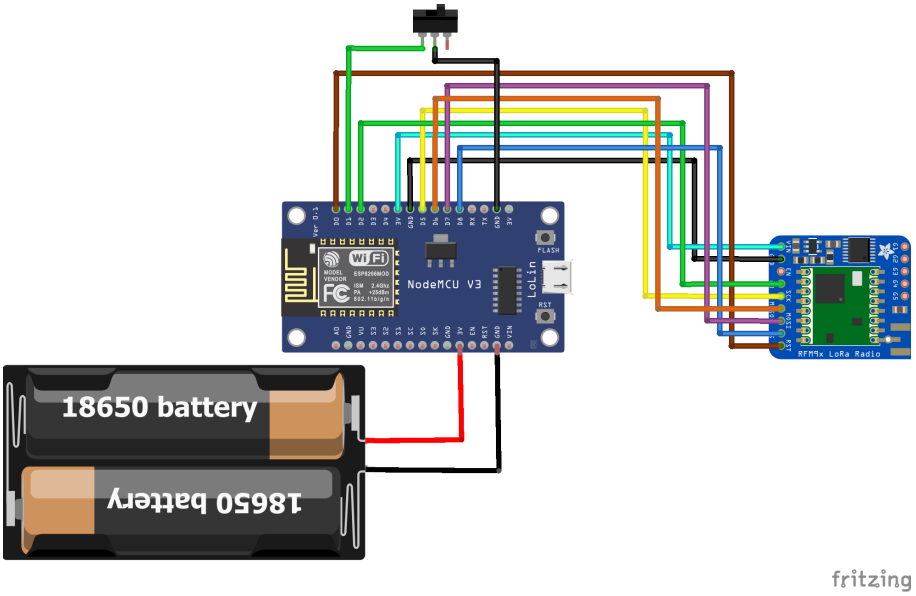
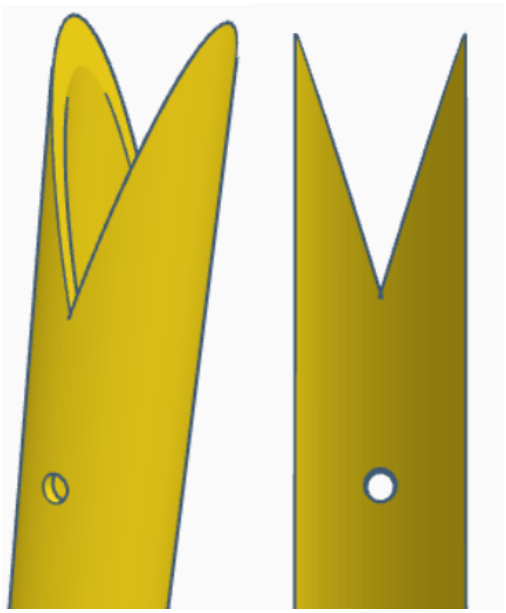


Figura 3.9: Conexiones del sistema receptor para el experimento

Para el receptor se utiliza una batería de Li-ion 18650 para suplir de energía al NodeMCU v3 debido a que este soporta voltajes de entrada hasta de 5V, la batería suple 4.2V. El microcontrolador alimenta con 3.3V al módulo LoRa y estos dos se comunican con el protocolo SPI. El interruptor varía un contador que representa el cambio de altura en el sistema. El nodeMCU actúa en modo SOFT AP, conectándose a la red wifi sobre la cual se pone en funcionamiento el servidor local con la finalidad de enviar peticiones HTTP a este último. Para poder establecer una comunicación entre el NodeMCU y el módulo lora se debe inicializar en el primero el protolo SPI. Así, los dispositivos se deben fijar a las caras internas de la caja para asegurar que no ocurran problemas de conexión ni de contacto. En la figura 3.10 se muestra la caja con el sistema receptor en su interior atornillada a uno de los soportes durante las pruebas finales del experimento.

Figura 3.10: Sistema receptor dentro de la caja atornillada a un soporte

En el caso de los postes se decidió usar dos soportes de la misma altura y ajustar la altura de las cajas mediante un tornillo a largo de estos logrando así variar los dos parámetros de interés para el estudio.



(a) Vista diagonal (b) Vista frontal

Figura 3.11: Corte diseñado para el soporte

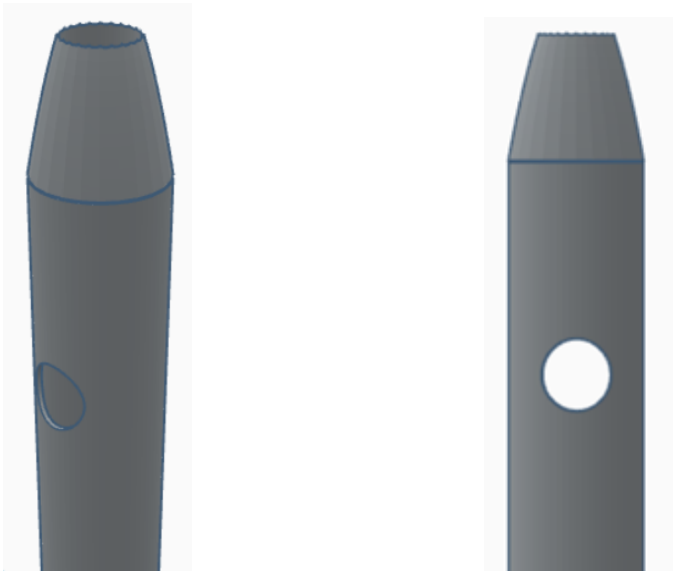
En la figura 3.11 se muestra el primer corte diseñado para el experimento, se usaron tubos de PVC debido a la facilidad de este material para ser cortado. La finalidad del corte es permitir que el tubo sea ingresado y retirado del suelo de forma fácil y rápida, gracias a las dos puntas que actúan como estacas. Junto con las cajas diseñadas y fabricadas, se realizó la primera prueba del sistema en la cancha de fútbol de la Pontificia Universidad Javeriana, lo cual se evidencia en la Figura 3.12.

(a) Caja y soporte de cerca

(b) Cajas a la misma altura con una distancia de separación

Figura 3.12: Pruebas tubo de PVC en la cancha de fútbol

El problema que se presentó en este caso es qué el material del tubo empieza a ceder debido a factores como la condición en la que este el suelo y cuánto tiempo esté plantado en este último. El césped de la cancha de fútbol estaba seco en el momento de realizar la prueba, lo que deformó las puntas del final del tubo y no permitió realizar más cambios de distancia en el enlace. Teniendo en cuenta este desempeño se optó por utilizar un tubo galvanizado, con un diámetro menor al de PVC para una mayor estabilidad. Debido al diámetro del tubo no se pudo realizar el mismo corte, en cambio, se realizo un bisel para ingresarlo y retirarlo del suelo de forma fácil, siguiendo el diseño en la figura 3.13:



(a) Vista diagonal

(b) Vista frontal

Figura 3.13: Diseño del bisel para el tubo galvanizado

Una vez se llegó a este soporte con la modificación necesaria se probó con el experimento diseñado en la cancha de fútbol de la Pontificia Universidad Javeriana, donde se desempeño de manera exitosa a la hora de dar estabilidad a las cajas para que el enlace inalámbrico se pueda realizar de forma confiable.

CAPÍTULO 4

PRUEBAS Y ANÁLISIS DE RESULTADOS

4.1. Experimento

Para llevar a cabo el experimento final se contó con los dos soportes hechos en tubo galvanizado, las dos cajas de acrílico que van atornilladas al tubo y un computador para el almacenamiento de los datos. El experimento consiste en clavar los soportes al suelo separados por una distancia fija, con las cajas a una misma altura y proceder a enviar un número de señales piloto, en este caso "paquetes", a través del enlace inalámbrico. Las distancias se varían según la distancia de estudio deseada y las alturas se varían teniendo en cuenta el rango del tubo, en cada una de estas configuraciones se realizan tres mediciones de potencia recibida. La primera prueba del experimento se llevo a cabo en la cancha de fútbol de la Universidad Javeriana Cali, un día soleado en la mañana.

(a) Caja y soporte de cerca

(b) Cajas a la misma altura con una distancia fija de separación

Figura 4.1: Pruebas tubo galvanizado en la cancha de fútbol

Se probó que las antenas y el sistema embebido funcionaran correctamente y que los

soportes se mantuvieran firmes y fueran fácil de desplazar. Se empezó con una distancia de 1.48m entre las dos antenas y a una altura de 1.05 m del suelo, siguiendo de la siguiente manera:

Distancias (m)	Alturas(m)
1.48	1.05
2.80	0.953
4.57	0.747
9.20	0.55
18.10	0.449
	0.353
	0.25

Tabla 4.1: Distancias y alturas usadas para la primera prueba

La tabla 4.1 muestra que para esta prueba el número de variaciones de distancia fue de cinco y el de altura de siete, además, se mandaron solo tres paquetes por cada configuración. Para verificar que el sistema funcionara de manera correcta se utilizó el software para programación científica Spyder, basado en el lenguaje Python. En el apartado anexos se muestra el proceso llevado a cabo para realizar el análisis de los datos. Se graficaron los datos en un mapa de calor donde los ejes de este representan las distintas alturas y distancias estudiadas. El resultado se muestra a continuación:

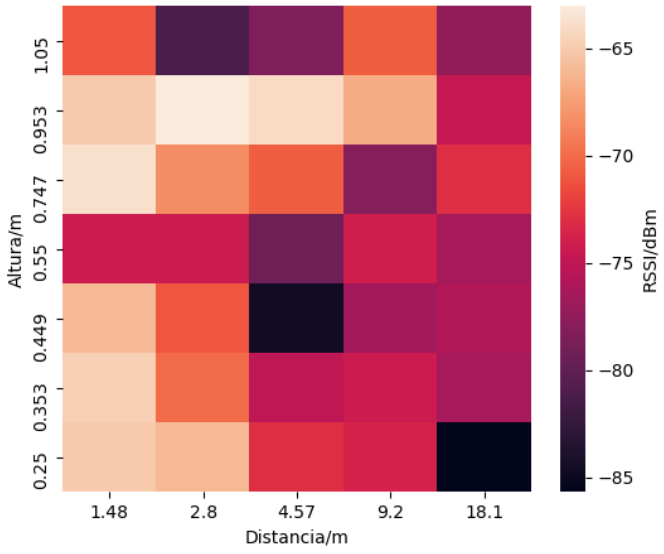


Figura 4.2: Mapa de calor de potencia recibida prueba cancha de fútbol.

Para la Figura 4.2 los tonos claros representan donde la potencia con la que se recibió

la señal fue más fuerte y entre más oscuros sean estos mayor atenuación en la señal hubo, se puede decir entonces que a medida que se incrementa la distancia entre antenas la señal presenta una mayor atenuación, y que la altura a la que el enlace presentó una mejor transmisión es la de 0.953m, debido a que es la fila en el mapa de calor que, a lo largo de las cinco distancias, presenta tonos más claros.
Para el experimento final las pruebas se realizaron en la Finca el Éden, ubicada a las afueras de la ciudad de Jamundi, Valle del Cauca, Colombia. Se realizaron mediciones de potencia recibida en la antena receptora en tres distintos cultivos de arroz con tres diferentes etapas de crecimiento de la planta y enviando siete paquetes por cada configuración del enlace, cabe aclarar que se tuvo en cuenta que las mediciones se realizaran durante días soleados y a la misma hora, para evitar que efectos ocasionados por el clima generaran diferencias entre mediciones realizadas.Las tres etapas escogidas dentro del proceso de crecimiento de la planta de arroz fueron Macollamiento (aprox. 30 días de crecimiento), Alargamiento del entrenudo (aprox. 60 días de crecimiento) y Producción del grano de arroz (aprox. 120 días de crecimiento). Se escogieron estas tres etapas debido a la importancia dentro del proceso de crecimiento de la planta que tienen y a que son donde mejor se evidencia diferencia entre altura y follaje de esta[30]. A continuación se muestra la tabla de las distancias y alturas tenidas en cuenta para este experimento:

Distancias (m)	**Alturas(m)**
1.02	0.979
2.538	0.886
5.07	0.687
10.14	0.488
20.10	0.379
40.07	0.285
	0.192

Tabla 4.2: Distancias y alturas usadas para el experimento final

En el siguiente apartado se discuten los resultados obtenidos.

4.2. Análisis de resultados

4.2.1. Análisis de las tendencias de variación de la atenuación

Los mapas de calor para notar diferencias entre los resultados obtenidos para las diferentes etapas se pueden evidenciar en la Figura 4.3

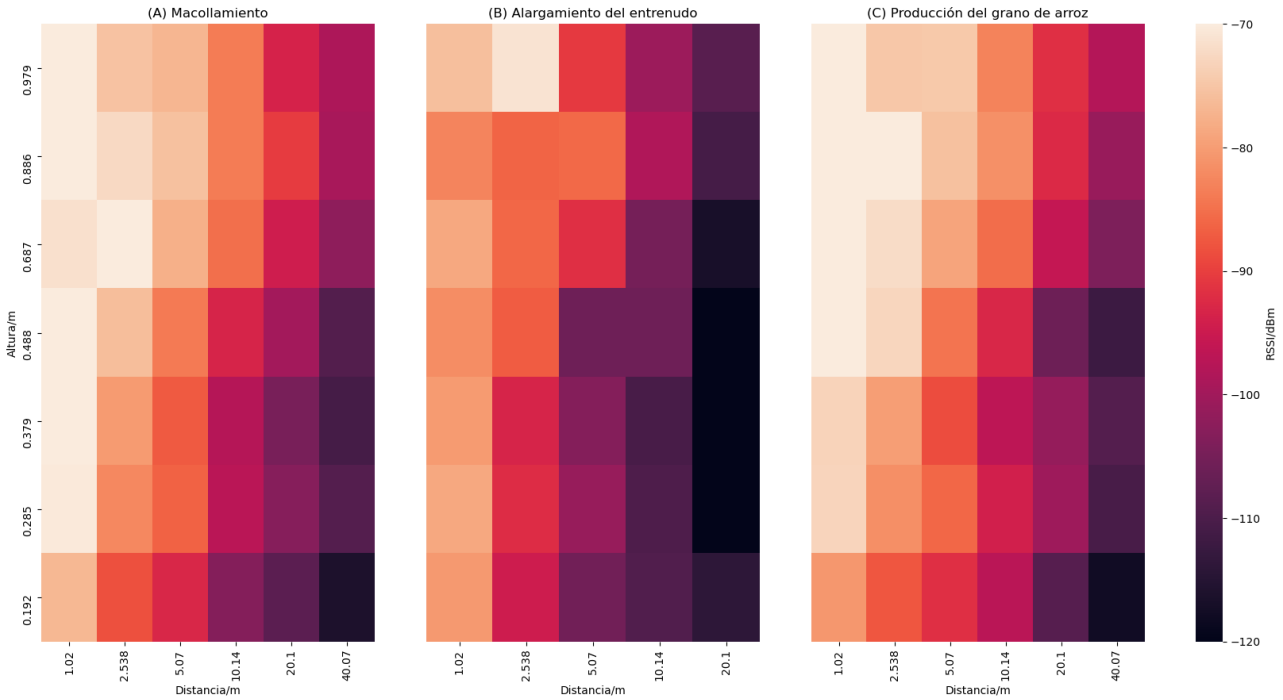


Figura 4.3: Mapa de calor de potencia recibida para los 3 cultivos estudiados.

La Figura 4.3 muestra la atenuación generada por el canal para las etapas de (A) macollamiento, (B) alargamiento del entrenudo y (C) producción del grano de arroz. Cabe aclarar que para la etapa de alargamiento del entrenudo se tomaron medidas hasta una distancia de 20 metros debido a problemas de logística en el lugar del experimento. Teniendo en cuenta que los colores claros representan un nivel de atenuación bajo en comparación a los oscuros, este nivel incrementa a medida que se incrementa la distancia en el enlace llegando a tener valores cercanos a los -120 dBm. Aunque existen diferencias en la atenuación para las tres etapas, se presentan similitudes en cuanto a tendencias de variación de esta a gran escala. La potencia recibida aumenta de manera directamente proporcional a la altura de las antenas e inversamente a la distancia. Se presentan distintos patrones a una menor escala, como puntos con atenuación muy baja en comparación a los que están su alrededor y viceversa, además, se genera más atenuación en el canal a medida que ocurre el desarrollo de la planta de arroz debido a los cambios de follaje y la altura de esta.

4.2.2. Resultado y análisis para modelos de potencia recibida

Resultados y análisis modelo de espacio libre

Se usó el modelo de espacio libre para estimar la potencia recibida en un enlace presente a través de un cultivo de arroz con 3 diferentes etapas de crecimiento. Este modelo se basa únicamente en la distancia entre las antenas, la frecuencia de transmisión inalámbrica y condiciones ideales del medio para modelar el valor de potencia recibida; Se ajustó el parámetro

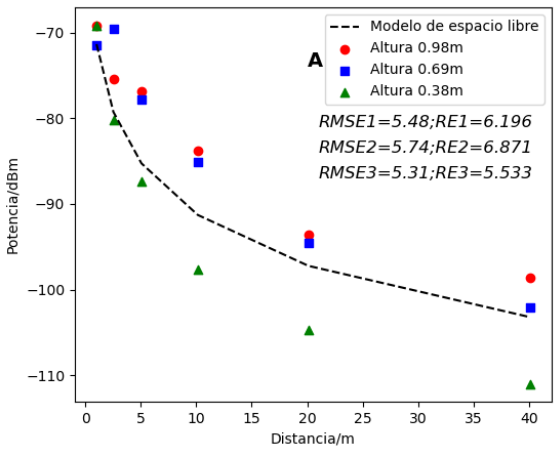
P_t de la ecuación (2.2) para cada una de las etapas mediante el método de mínimos cuadrados, dando como resultado los parámetros mostrados en la tabla 4.3.

$$Pr_{EL}(dBm) = P_t(dBm) + 2 \times G_{tr}(dBm) + 20 \times [(\log_{10}(\lambda) - \log_{10}(4 \times \pi) - (\log_{10} d)]$$

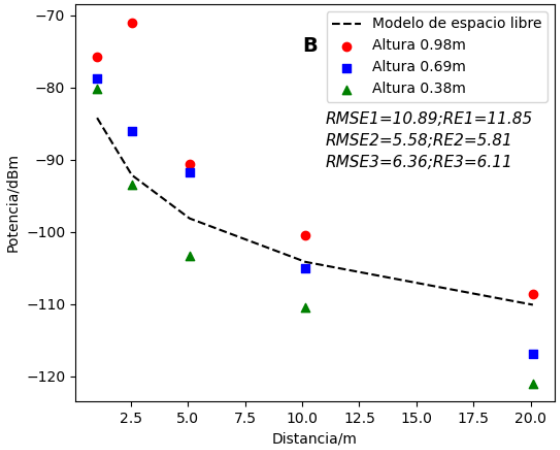
Etapa	**Parámetro Pt(dBm)**
Macollamiento	55.990
Alarg. entrenudo	66.692
Prod. grano de arroz	56.037

Tabla 4.3: Parámetros ajustados para la ecuación del modelo espacio libre en las tres etapas

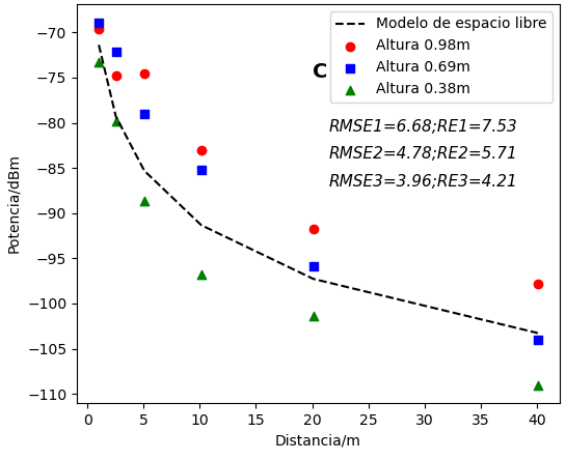
En la Figura 4.4 se muestra el resultado de la estimación de potencia recibida basada en el modelo de espacio libre para tres alturas diferentes.



(a) Macollamiento



(b) Alargamiento del entrenudo



(c) Producción del grano

Figura 4.4: Potencia recibida vs. Distancia según el modelo de espacio libre con varias alturas para la etapa de (a) Macollamiento; (b) Alargamiento del entrenudo; (c) Producción del grano.

Del análisis de la Figura 4.4 se infiere que para los casos de alturas fijas el modelo obtuvo un rango de RMSE entre 3.96 – 10.89 % y error relativo entre 4.21 – 11.85 %. En la primera etapa los valores fueron relativamente bien modelados dadas las distancias entre los valores para las primeras dos alturas con respecto a los estimados por el modelo, en contraste, en la etapa de alargamiento del entrenudo con el incremento de la distancia se obtiene una peor estimación y cuando las antenas se encontraron a 0.98m del suelo se obtuvo la peor aproximación, se evidencia como el ambiente de este cultivo afectó de manera drástica a la transmisión en esta altura en comparación a las otras dos etapas. El cuadro de comparación entre los valores estimados y medidos se muestra en la figura 4.5.

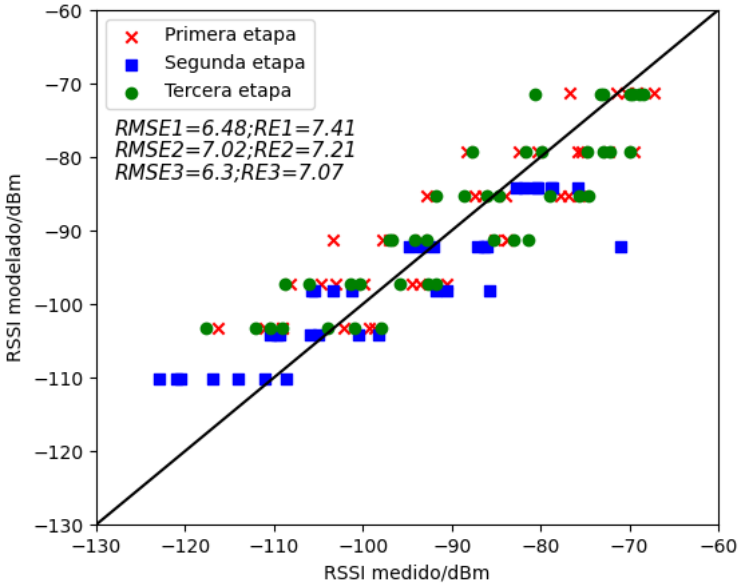


Figura 4.5: Cuadro comparativo de los valores estimados con el modelo de espacio libre contra los valores medidos experimentalmente para las 3 etapas.

De la figura 4.5 se puede concluir que la aproximación de este modelo presenta rangos de error de 6.3 – 6.48 % para el RMSE y de 7.07 – 7.41 % para el RE. Para este análisis se define una linea $x = y$ y se representan gráficamente los datos a comparar punto a punto con el objetivo de identificar las similitudes entre los valores medidos y los estimados. El mayor error de aproximación se presentó en la segunda etapa debido a diferencias en condiciones externas (tipo de suelo y humedad de este, desarrollo y crecimiento particular para una planta) en las que se encontraron los cultivos; Estos rangos de error y los que fueron analizados más arriba nos indican que el modelo no puede ser usado directamente para estimar la potencia recibida en un canal inalámbrico presente a través de un cultivo de arroz. Para modelar un canal inalámbrico es importante tener en cuenta no solo distancia y frecuencia de transmisión, el análisis del modelo de espacio libre nos indica que es necesario hacer uso de más parámetros.

Resultados y análisis modelo de dos rayos

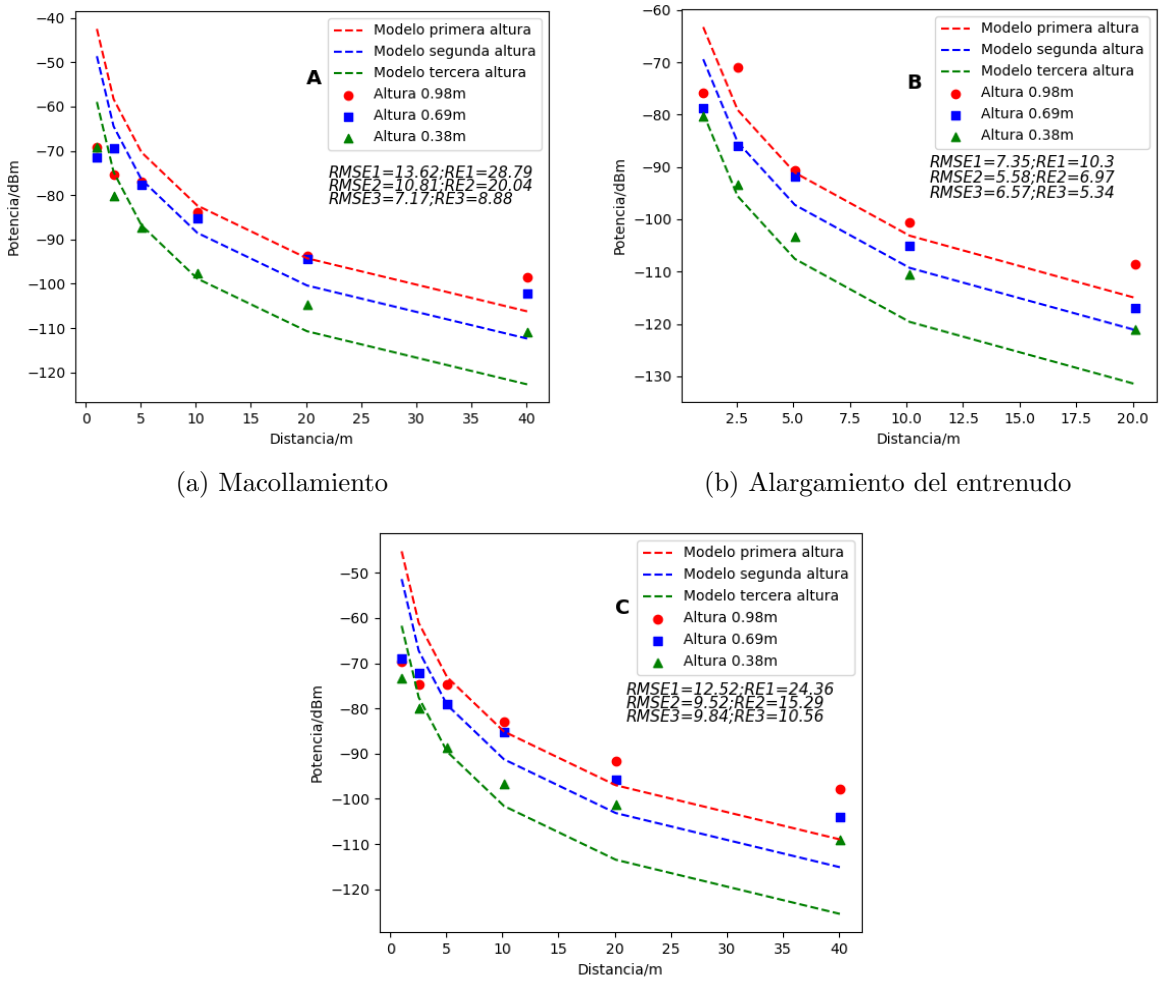
Se uso el modelo de dos rayos con reflexión en el suelo para estimar la potencia recibida en un enlace presente a través de un cultivo de arroz con 3 diferentes etapas de crecimiento. Para este modelo se realizó un ajuste por método de mínimos cuadrados para el parámetro P_t en la ecuación (2.8), obteniendo diferentes resultados para cada una de las etapas, como se muestra en la tabla 4.4

$$P_{r(TR)}(dBm) = (P_t)(dBm) + 2 \times G(dBm) + 40 \times (\log_{10}(h_{rt})) - 40 \times \log_{10}(d)$$

Etapa	Parámetro Pt(dBm)
Macollamiento	141.7544
Alarg. entrenudo	154.7426
Prod. grano de arroz	144.5023

Tabla 4.4: Parámetros ajustados para la ecuación del modelo de dos rayos en las tres etapas

Este modelo además de tener en cuenta parámetros como la distancia entre antenas y la frecuencia de transmisión del enlace, depende de la altura a la que las antenas estén ubicadas, como se evidencia en la figura 4.6.



(a) Macollamiento

(b) Alargamiento del entrenudo

(c) Producción del grano

Figura 4.6: Potencia recibida vs. Distancia según el modelo de dos rayos con varias alturas para la etapa de (a) Macollamiento; (b) Alargamiento del entrenudo; (c) Producción del grano.

El análisis de la Figura 4.6 indica con el modelo de dos rayos se estimó la potencia recibida

con RMSE en el rango de 5.58 – 13.62 % y RE de 5.34 - 28.79 %. Como se mencionó en el apartado de marco teórico, el modelo de dos rayos asume ciertas condiciones para llegar a la ecuación de perdidas por trayectoria, entre ellas que la altura de las antenas es despreciable con respecto a la distancia entre estas o que el suelo presenta un coeficiente de reflexión ideal, en este caso en particular no se cumplen la mayoría de estas asunciones, ni siquiera que las antenas estén a distintas alturas por lo tanto para este caso en particular el modelo de dos rayos no sirve para estimar la potencia recibida en el enlace. Para las tres etapas y a diferentes alturas el modelo estima de forma pobre los valores en largas distancias, teniendo una caída pronunciada en los tres casos a partir de los 20 metros, estimando valores muy distintos a los obtenidos en el experimento. La altura de 0.69m obtuvo la mejor estimación en promedio a lo largo de las tres etapas estudiadas, lo que indica incluir la altura de las antenas para la estimación del modelo influye en los resultados obtenidos. A continuación se muestra la comparación de los datos estimados por el modelo, teniendo en cuenta la totalidad de alturas y distancias, contra los hallados experimentalmente.

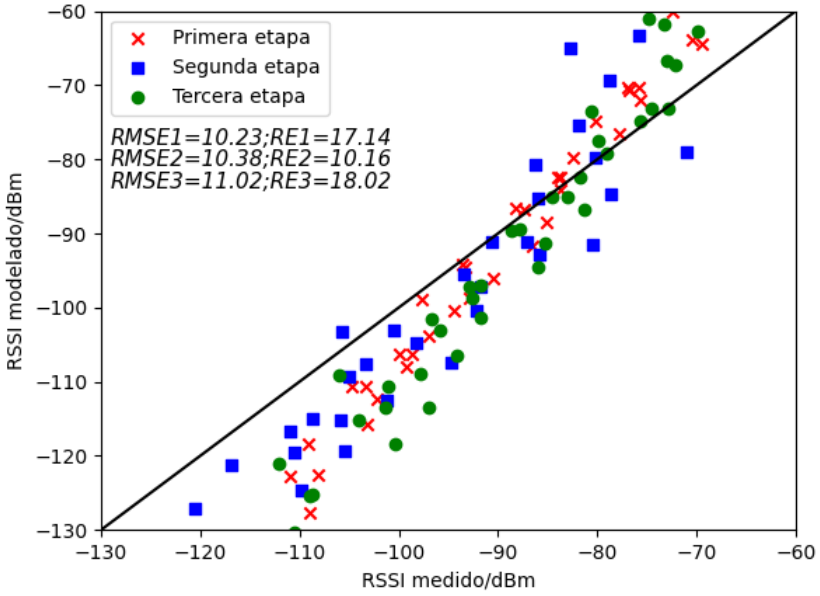


Figura 4.7: Cuadro comparativo de los valores estimados con el modelo de dos rayos contra los valores medidos experimentalmente para las 3 etapas.

En la figura 4.7 se define una linea $x = y$ y se representan gráficamente los datos a comparar punto a punto con el objetivo de analizar las similitudes entre los valores medidos y los estimados. Se infiere que al estimar la potencia recibida en el canal con el modelo de dos rayos el RMSE es de 10.23-11.02 % y el de RE es de 17.14-18.02 %. En general el desempeño del modelo fue pobre. La etapa de producción del grano obtuvo el mayor RMSE con un valor de 11.02 %, en parte, debido a factores externos de cada uno de los cultivos estudiados. Con estos rangos y los anteriores podemos concluir que para este caso no se puede estimar directamente la potencia recibida a través del enlace haciendo uso del modelo de dos rayos clásico.

Resultados y análisis modelo de Log-Distancia

Se utilizó el método de aproximación por mínimos cuadrados para hallar los valores de K y n con el fin de ajustar la ecuación (2.10) del modelo Log-distance lo mejor posible al canal estudiado. En este caso se calcularon estos dos valores para las 3 etapas de crecimiento. En la tabla 4.5 se muestran los resultados obtenidos.

$$PL_{LD}(d) = (K) + 10 \times (n) \times \log_{10} d$$

Etapa	Parámetro K(dBm)	Índice de p. trayectorias, n
Macollamiento	-68.0040	-2.3788
Alarg. entrenudo	-82.15	-1.9660
Prod. grano de arroz	-68.8325	-2.2838

Tabla 4.5: Parámetros ajustados para la ecuación del modelo Log-distancia en las tres etapas

De este forma, a medida que la planta se desarrolla se evidencia como cambia el n (índice de perdidas por trayectoria), volviéndose cada vez menos negativo. Esto significa que el canal está haciendo que se generen más perdidas en la transmisión debido, en parte, a que las plantas cambian en tamaño y la cobertura que ocasiona su dosel se vuelve más densa, generando un incremento en fenómenos como la refracción, reflexión y difracción de la señal inalámbrica transmitida. El factor de pérdidas por trayectoria del canal inalámbrico presente a través de un cultivo de arroz aumenta en proporción al desarrollo o crecimiento de la planta. En el caso de las mediciones para la segunda etapa el ambiente de este cultivo generó que hubiera mas atenuación en comparación a las otras dos mediciones, por eso el valor de n es mucho más negativo en este caso.

Basados en las ecuaciones anteriores, se procedió a estimar la potencia recibida en el enlace inalámbrico usando el modelo Log-Distance de una pendiente para cada una de las etapas. La Figura 4.8 muestra el desempeño del modelo estimando la potencia recibida:

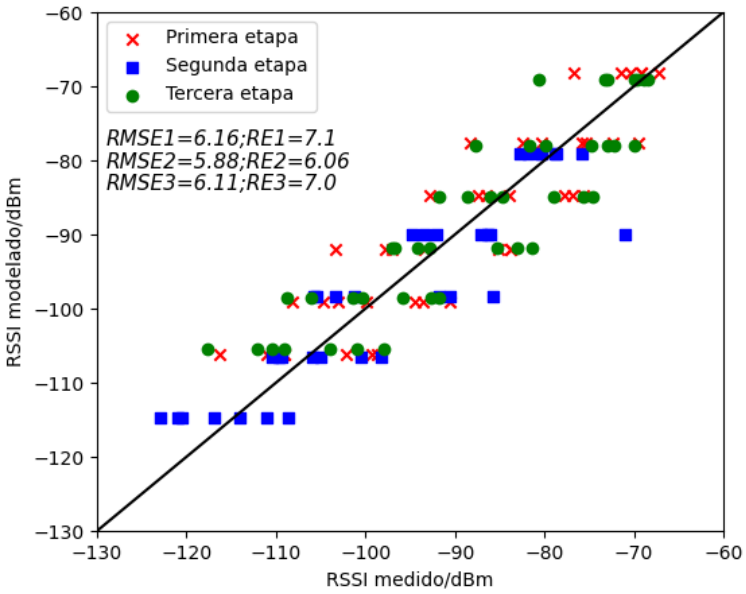
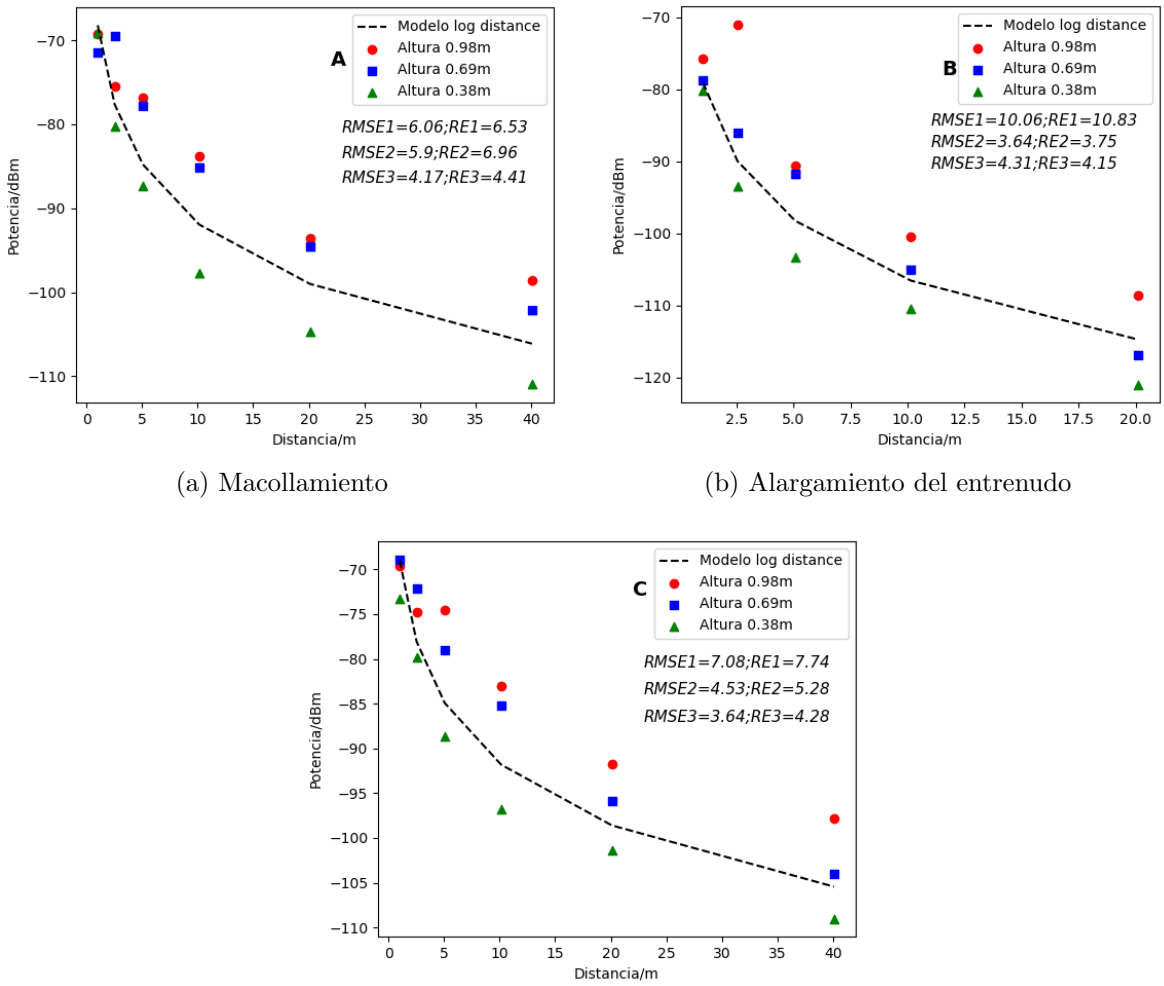


Figura 4.8: Cuadro comparativo de los valores estimados con el modelo Log-distancia contra los valores medidos experimentalmente para las 3 etapas.

En la Figura 4.8 se define una linea $x = y$ y se representan gráficamente los datos a comparar punto a punto con el objetivo de identificar las similitudes entre los valores medidos y los estimados. Se puede afirmar que al estimar la potencia recibida con el modelo Log-distance el error RMSE estuvo en el rango de 6.11-6.16 % y el RE entre 6.06-7.01 %. Este modelo presenta unos rangos de errores altos a medida que la planta se desarrolla, siendo la primera etapa la peor estimada. El ajuste que se realiza para calcular el factor de perdidas por trayectoria, ajuste que caracteriza el canal de una forma más específica, ayuda a disminuir el error en comparación al modelo de espacio libre, qué, como este, solo tiene en cuenta la distancia y la frecuencia de transmisión de la señal. En la Figura 4.9 se analiza el desempeño del modelo para 3 diferentes alturas:



(a) Macollamiento

(b) Alargamiento del entrenudo

(c) Producción del grano

Figura 4.9: Potencia recibida vs. Distancia según el modelo Log-Distancia con varias alturas para la etapa de (a) Macollamiento; (b) Alargamiento del entrenudo; (c) Producción del grano.

Como se puede observar en la figura 4.9, el modelo log-distance no varía según hay cambios en la altura. El mayor error de aproximación para este modelo se dio en la etapa de alargamiento del entrenudo con la primera altura debido a condiciones externas características de cada cultivo estudiado. En las etapas de macollamiento y producción del grano de arroz el valor del RMSE disminuye a medida que se reduce la altura de las antenas en el enlace. Se evidencia entonces como incluir la caracterización del factor de pérdidas por trayectoria influye en la estimación de la potencia recibida en un enlace inalámbrico para un canal presente a través de un cultivo de arroz y como los cultivos estudiados para la primera y tercera etapa pueden llegar a tener condiciones ambientales similares en comparación a la segunda etapa.

Resultados y análisis modelo de dos rayos ajustado

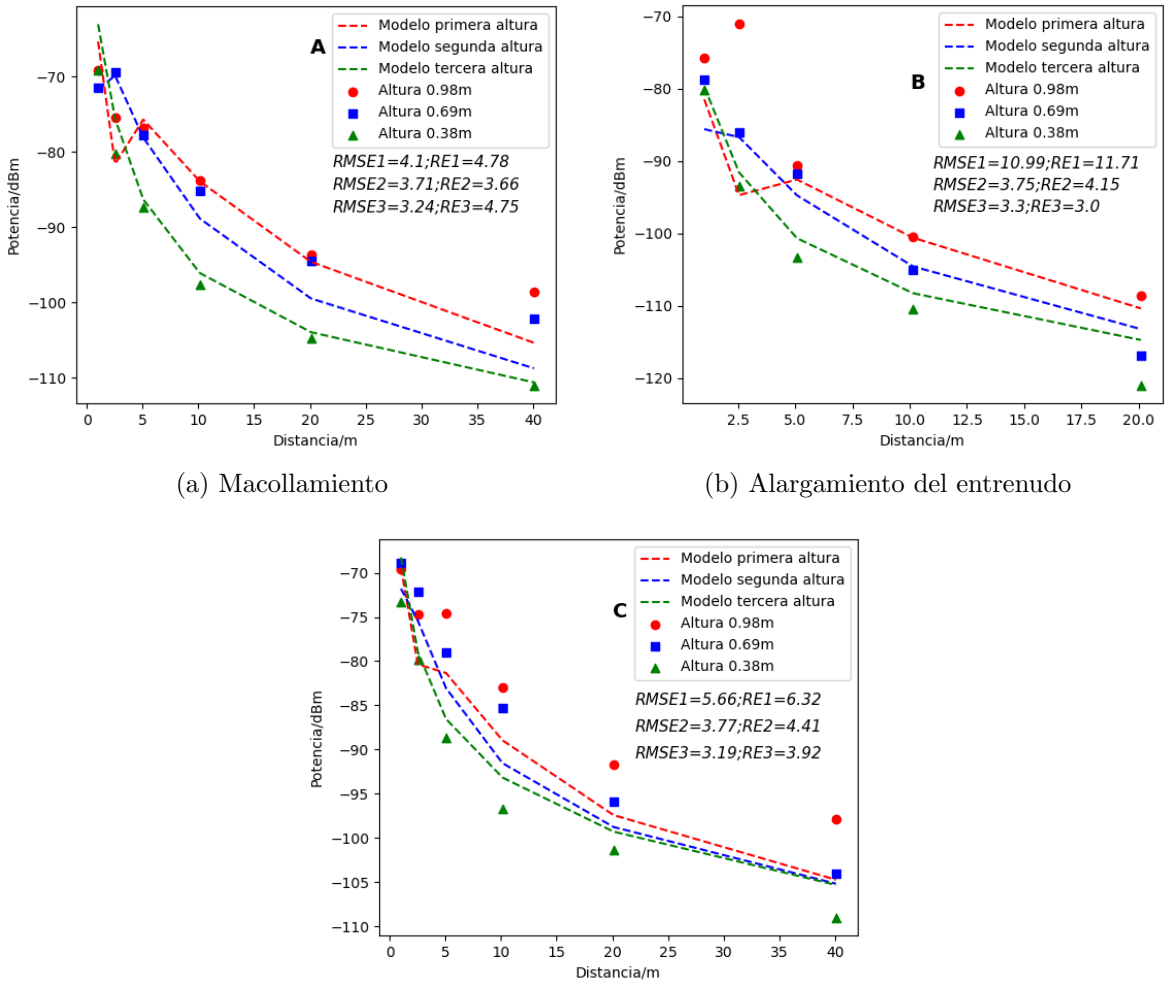
Para ajustar los parámetros de la mejor forma se utilizó el método de cuadrados mínimos en la ecuación 2.11 para cada una de las etapas. Dado que el coeficiente de reflexión (Γ) puede tener valores entre -1 y 1 este rango se especificó en el ajuste para que los valores estimados tuvieran relación con los de la realidad. En la tabla 4.6 se muestran los resultados obtenidos.

$$Pr_{MP}(d)(dBm) = (P_t)(dBm) + 20 \times \left[\log_{10}\left(\frac{\lambda}{4 \times \pi}\right)\right] + G(dBm) + 10 \times \log_{10}\left[\left(\frac{1}{d^2}\right) + \left(\frac{2 \times (\Gamma) \times A}{d \times B}\right) + \left(\frac{\Gamma^2}{B^2}\right)\right]$$

Etapa	**Parámetro Pt(dBm)**	**Coeficiente reflexión, Γ**
Macollamiento	-46.0870	-0.7644
Alarg. entrenudo	-61.8687	-0.5460
Prod. grano de arroz	-49.4575	-0.3436

Tabla 4.6: Parámetros ajustados para la ecuación del modelo propuesto en las tres etapas

Al analizar la tabla 4.6 se puede mencionar el cambio que hay en el coeficiente de reflexión para cada una de las etapas. Idealmente, en un medio que refleje la totalidad de la señal incidente el valor de $\Gamma = -1$, lo que significa que hay un cambio de fase pero no de amplitud en la señal; en el caso de los cultivos estudiados en la primera etapa de desarrollo se obtuvo un valor de $-0{,}7644$, en la segunda uno de $-0{,}5460$ y en la tercera de $-0{,}3436$, esto indica que a medida que avanza el ciclo de crecimiento de la planta el coeficiente se aleja de la reflexión ideal, ocasionando que la señal transmitida sea afectada por fenómenos de transmisión ocasionados por la estructura,altura y el dosel de las hojas de la planta de arroz. Dado que este modelo es basado en el de dos rayos, la altura a la que están ubicadas las antenas en el enlace influyen en la estimación que este realiza, lo que se evidencia a continuación:



(a) Macollamiento

(b) Alargamiento del entrenudo

(c) Producción del grano

Figura 4.10: Potencia recibida vs. Distancia según el modelo de dos rayos ajustado con varias alturas para la etapa de (a) Macollamiento; (b) Alargamiento del entrenudo; (c) Producción del grano.

Según la figura 4.10, al estudiar la estimación que el modelo de dos rayos ajustado hace sobre los datos experimentales esta obtuvo un rango de RMSE entre 3.19-10.99 % y de RE 3.0-11.71 %. La precisión de la estimación en cada una de las etapas aumenta de forma monótona a medida que se reduce la altura de las antenas, en el caso de la altura de 0.38 m se obtuvo una mejor estimación a lo largo de todas las etapas. El mayor error de aproximación se obtuvo para la etapa de alargamiento del entrenudo con la altura de 0.98 m debido a, como se mencionó con anterioridad, las condiciones externas y ambientales características del cultivo estudiado para esta. La densidad del follaje de la planta de arroz es menor en el tallo, lo que podría explicar la forma de la función del modelo para las primeras alturas. Visualmente en la etapa de macollamiento se evidencia que la estimación fue mejor dado que el modelo intenta recrear los cambios bruscos en los datos obtenidos experimentalmente (entre las distancias

2.5 m y 5 m para las dos primeras alturas). El modelo no realiza una estimación correcta cuando se configura el enlace en la última altura, como se muestra en la Figura 4.11.

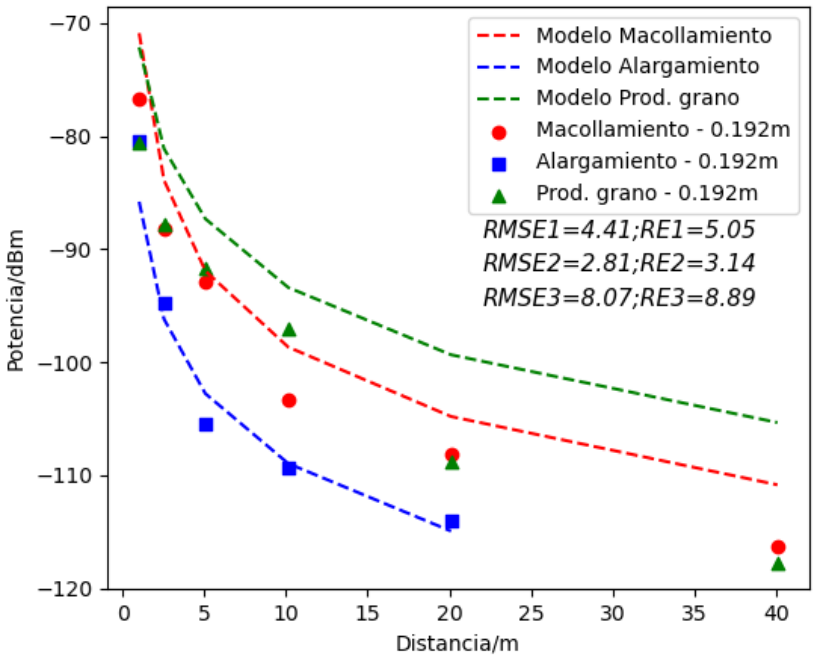


Figura 4.11: Potencia recibida estimada vs. distancia del modelo de dos rayos ajustado a una altura de 0.192 para las tres etapas de desarrollo.

Dadas las condiciones del suelo de un cultivo de arroz el modelo no logra realizar una estimación correcta debido a todos los factores que están influyendo en la trayectoria del rayo reflejado. La humedad, agua, barro, follaje de la planta entre otras condiciones contribuyen a una pobre transmisión de la señal, lo que no permite una buena estimación. El mejor ajuste se realiza para la etapa de alargamiento del entrenudo, en el caso de las otras dos etapas a partir de cierta distancia se presenta una gran diferencia entre valores estimados y medidos. En la figura 4.12 se muestra el cuadro comparativo para el modelo de dos rayos ajustado.

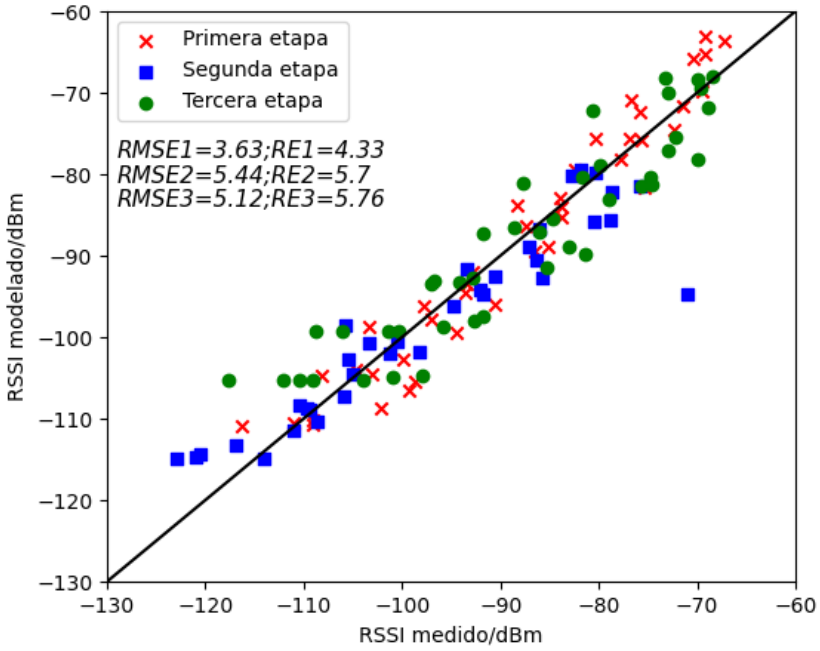
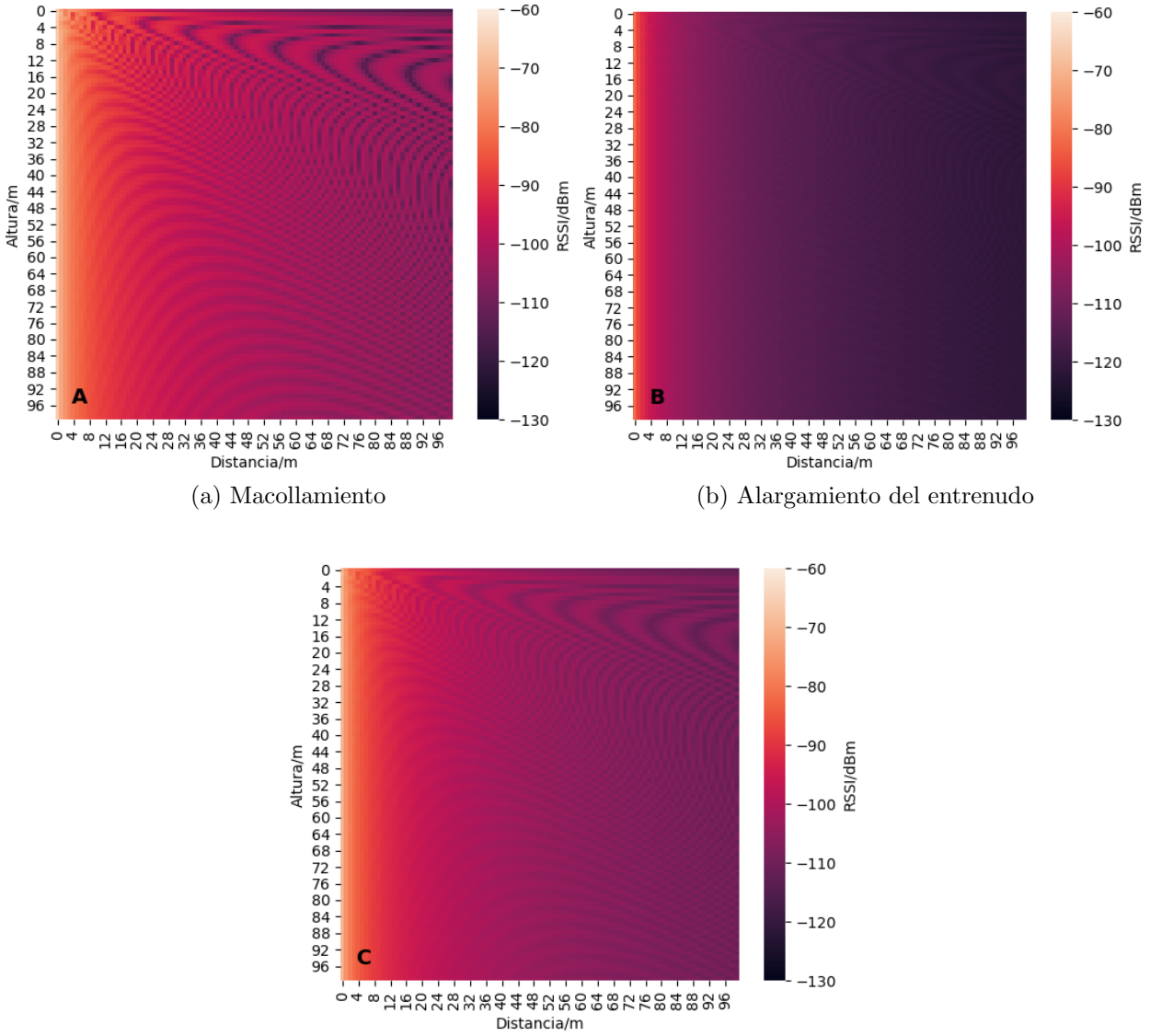


Figura 4.12: Cuadro comparativo de los valores estimados con el modelo de dos rayos ajustado contra los valores medidos experimentalmente para las 3 etapas.

De la Figura 4.12 se concluye que el estimar la potencia recibida con el modelo de dos rayos ajustado a las condiciones del experimento se hizo con un rango de RMSE entre 3.63-5.12 % y de RE entre 4.33-5.76 %, con estos rangos y los anteriores podemos afirmar que en este caso el modelo ajustado propuesto estima con mayor precisión la potencia recibida que los modelos de Espacio Libre, Dos Rayos y Log-Distance, lo que indica que para canales que se encuentren en campos de arroz el modelo de dos rayos ajustado es más apropiado que los modelos convencionales. El mayor error se da en la segunda etapa, lo que da a entender que las distintas condiciones ambientales en las que se encuentren los cultivos juegan un papel crucial y tienen mucha incidencia en cómo se transmite una señal inalámbrica. La caracterización que se realiza para las tres etapas de crecimiento por medio del coeficiente de reflexión(Γ) y la misma altura de las antenas permite llegar a una mejor estimación del canal.

Para entender de una mejor manera el comportamiento del modelo y los cultivos, se simuló un caso en el cual se incrementa la altura y distancia en una unidad hasta llegar a 100, dando como resultado los siguientes gráficos:



(a) Macollamiento

(b) Alargamiento del entrenudo

(c) Producción del grano

Figura 4.13: Simulación estimación de potencia con el modelo de dos rayos ajustado para la etapa de (A) Macollamiento; (B)Alargamiento del entrenudo; (C)Producción del grano.

De la Figura 4.13 se puede concluir que la mayor atenuación de la señal se genera en la segunda etapa, pero al mismo tiempo la primera presenta una menor atenuación con respecto a la segunda y tercera, llegando esta a tener un poco más de relación en cuanto a la atenuación generada por el medio con la tercera etapa que con la segunda. La primera etapa de crecimiento de la planta no genera tanta atenuación en comparación a las otras dos, por lo que se confirma que el avance del proceso de crecimiento del arroz genera más atenuación en el canal inalámbrico. Los resultados mostrados en la Figura 4.13 muestran por qué un estudio de este tipo debe ser realizado bajo condiciones ambientales similares, teniendo en

cuenta que si se quiere estudiar diferentes etapas de desarrollo deben hacerse en el mismo cultivo y el equipo que se use debe asegurar una buena transmisión inalámbrica.

CAPÍTULO 5

CONCLUSIONES

Realizar un estudio sobre la atenuación de un canal es una herramienta muy importante para diseñar e implementar una red de sensores inalámbricos de calidad, aún más si se piensa hacer uso de esta para una aplicación como la agricultura de precisión. El efecto que tiene el crecimiento y desarrollo de una planta sobre un enlace inalámbrico presente a través de un cultivo se puede estudiar por medio de modelos de propagación. Así, contar con un modelo que tenga en cuenta estas condiciones particulares y la aplicación final debe ser usado para cada caso particular.

En esta investigación se diseño e implementó un experimento para comparar la atenuación generada por un canal sobre un enlace con diferentes alturas de antenas y en tres etapas de desarrollo de una planta de arroz. El RSSI medido presentó una disminución directamente proporcional a la altura de las antenas e inversa a la distancia entre estas. A medida que se desarrolla un cultivo de arroz el ambiente de transmisión del canal presente en este se deteriora. El modelo de espacio libre probó no tener la mayor precisión para estimar la atenuación de canales presentes en un cultivo de arroz con un RE de 7.07-7.41 % . El modelo de dos rayos obtuvo la peor estimación de los modelos utilizado en un enlace presente a través de un cultivo de arroz con RE de 10.23-11.02 % pero mostró que el parámetro de la altura de las antenas influye de manera positiva en la caracterización de este. El modelo Log-Distancia tiene una buena aplicación en la estimación de la atenuación de canales presentes en cultivos de arroz debido a la caracterización que se realiza del índice de pérdidas por trayectoria pero se obtuvo un error relativo mayor qué 6 %. El modelo de dos rayos ajustado mejoró el rendimiento del modelo de dos rayos convencional y se obtuvo un error relativo menor que 6 %, por lo que este modelo tiene un mejor desempeño que los otros estudiados en el ambiente de un cultivo de arroz. Este modelo puede servir como base para el modelamiento de canales para la aplicación de redes de sensores inalámbricos. Realizar un experimento donde se estudie un solo cultivo a lo largo de su desarrollo puede contribuir a una mejor caracterización por parte del modelo propuesto.

5.1. Trabajos futuros

Basados en los resultados obtenidos a lo largo de esta investigación, una buena continuación para la misma sería realizar el estudio sobre un solo cultivo analizando las tres etapas seleccionadas a lo largo del tiempo. En el caso del modelo propuesto el estudio del coeficiente de reflexión Γ a profundidad puede llevar a obtener una mejor caracterización del estado de un cultivo, puesto que este contiene información de algunas propiedades eléctricas(como la impedancia) y la geometría del material donde se refleja la señal. Incluir más alturas también puede ser un punto crucial, para distinguir de forma correcta cual es la altura necesaria para que una red de sensores inalámbricos funcione a de manera eficiente y como afecta el desarrollo de la planta a los diferentes fenómenos de transmisión ocurridos a través del canal. Además, sería interesante descubrir que tanto se ve afectada la señal dependiendo de distintos parámetros como el numero de hojas de las plantas, los macollos que puedan tener, si tienen en ese momento grano o no, el color de estas, condiciones ambientales como la humedad del aire y del suelo, que haya distintos tamaños de planta en una misma etapa; esto llevaría a poder caracterizar como y cuáles son los obstáculos presentes a través de un canal inalámbrico, es decir, conocer el estado de las plantas que obstaculizan ese enlace sabiendo como afectan estas a la transmisión por medio de este.

BIBLIOGRAFÍA

[1] Hernández Xolocotzi, E. (1988). La agricultura tradicional en México. Comercio Exterior, [online] (Vol. 38, num. 8), p.673. Available at: http://revistas.bancomext.gob.mx/rce/magazines/189/2/RCE2.pdf [Accessed 19 May 2019].

[2] E. García and F. Flego, Agricultura de precisión , Revista Ciencia y Tecnología, p. 100, 2008.[Accessed 6 March 2019].

[3] F. Wang and K. Sarabandi, A Physics-Based Statistical Model for Wave Propagation Through Foliage, IEEE Transactions on Antennas and Propagation, vol. 55, no. 3, pp. 958-968, 2007. Available: 10.1109/tap.2007.891841 [Accessed 7 March 2019].

[4] N. Sabri, S. Aljunid, M. Salim, R. Kamaruddin, R. Ahmad and M. Malek, Path Loss Analysis of WSN Wave Propagation in Vegetation , Journal of Physics: Conference Series, vol. 423, p. 012063, 2013. Available: 110.1088/1742-6596/423/1/012063 [Accessed 7 March 2019].

[5] J. Thelen, D. Goense, K. Langendoen. Radio wave propagation in potato fields , Wireless Sensor Network, Vol.2 No.12, January 10, 2011. [Accessed 6 March 2019].

[6] A. Saez Domingo, La Agricultura y su evolución a la agroecología. Valencia: Obrapropia, 2009, p. 3.

[7] C. Vidal, El monocultivo y sus consecuencias. Blog sobre medio ambiente y ecología , Ecoclimatico.com, 2008. [Online]. Available: http://www.ecoclimatico.com/archives/el-monocultivo-ysus-consecuencias-822. [Accessed 01 May 2019].

[8] La conectividad Inalámbrica: un enfoque para el alumno , Temas para la educación, no. 6, 2010. Available: https://www.feandalucia.ccoo.es/docu/p5sd6785.pdf. [Accessed: 01 May 2019].

[9] J. Huidobro, Antenas de telecomunicaciones , Revista Digital de ACTA, p. 1, 2013. Available: https://www.acta.es/medios/articulos/cienciasytecnologia/020001.pdf [Accessed 1 May 2019].

[10] C. Olivares, Investigación Documentada - Medios de Transmisión. Managua, 2015, p. 10.

[11] Path Loss, Wireless communication.nl, 1995. [Online]. Available: http://www.wirelesscommunication.nl/refe [Accessed 01 May 2019].

[12] H. Klaina, A. Alejos, O. Aghzout and F. Falcone, Characterization of Near-Ground Radio Propagation Channel for Wireless Sensor Network with Application in Smart Agriculture , Proceedings, vol. 2, no. 3, p. 110, 2017. Available: 10.3390//ecsa-4-04922 [Accessed 1 May 2019].

[13] Gao, Z., Li, W., Zhu, Y., Tian, Y., Pang, F., Cao, W. and Ni, J. (2018). Wireless Channel Propagation Characteristics and Modeling Research in Rice Field Sensor Networks. Sensors, [online] 18(9), p.3116. Available at: 10.3390/s18093116. [Accessed 3 Feb 2020].

[14] Zennaro, M. (2012). Introduction to Wireless Sensor Networks. [ebook] Trieste, p.5. Available at: https://www.academia.edu/2236615/Introduction_to_Wireless_Sensor_Networks [Accessed 5 Sep. 2019].

[15] Ingeniería de los sistemas embebidos. 1st ed. Madrid, España, pp.6-10. Available at: http://www.ieec.uned.es/investigacion/Dipseil/PAC/archivos/Informacion_de_referencia_ISE5_3_1.pdf [Accessed 5 February 2021].

[16] Isi.edu. 2021. 18.1 Free space model. [online] Available at: https://www.isi.edu/nsnam/ns/doc/node217.html [Accessed 7 December 2020].

[17] GaussianWaves. 2021. Two ray ground reflection model - GaussianWaves. [online] Available at: https://www.gaussianwaves.com/2019/03/two-ray-ground-reflection-model/ [Accessed 3 January 2021].

[18] Idc-online.com. 2021. LOG DISTANCE PATH LOSS OR LOG NORMAL SHADOWING MODEL. [online] Available at: http://www.idc-online.com/technical_references/pdfs/electronic_engineering/Log_Distance_Path_Loss_or_Log_Normal_S [Accessed 2 February 2021].

[19] Schlemberg. Oilfield Glossary. n.d. Coeficiente de reflexión. [online] Available at: https://www.glossary.oilfield.slb.com/es/terms/r/reflection_coefficient [Accessed 2 February 2021].

[20] 2021. Datasheet LoRa SX1278. [ebook] Semtech, pp.2-15. Available at: ¡https://cdn-shop.adafruit.com/product-files/3179/sx1276_77_78_79.pdf¿[Accessed 2 December 2020].

[21] Lora.readthedocs.io. 2021. LoRa — LoRa documentation. [online] Available at: https://lora.readthedocs.io/en/latest/ [Accessed 3 December 2020].

[22] Rodriguez Munca, J., 2016. DISPOSITIVO LoRa DE COMUNICACIÓN A LARGO ALCANCE Y BAJO CONSUMO ENERGÉTICO PARA APLICACIONES DEL ÁMBITO DEL DESARROLLO. Master. Universidad Politecnica de Madrid.

[23] Nañez, Y., 2021. INGENIERIA EN MICROCONTROLADORES Protocolo SPI(Serial Peripherical Interface). [online] Academia.edu. Available at: https://www.academia.edu/5249761/INGENIERIA_EN_MICROCONTROLADORES_Protocolo_SPI_S [Accessed 3 December 2020].

[24] Weatherstation. 2021. ESP8266/NodeMCU with FT81x based 7"display. [online] Available at: https://weatherhelge.wordpress.com/2017/01/18/esp8266-with-ft81x-based-7-display/ [Accessed 3 Febrero 2021].

[25] Farnell.com. 2021. Datasheet Arduino UNO. [online] Available at: https://www.farnell.com/datasheets/1682209.pdf [Accessed 3 November 2020].

[26] Circuit Digest. 2021. Interfacing SX1278 (Ra-02) LoRa Module with Arduino. [online] Available at: https://circuitdigest.com/microcontroller-projects/arduino-lora-sx1278-interfacing-tutorial [Accessed 2 September 2020].

[27] Robots-argentina.com.ar. 2021. Uso de la EEPROM de Arduino — Robots Didácticos. [online] Available at: http://robots-argentina.com.ar/didactica/uso-de-la-eeprom-de-arduino/ [Accessed 7 December 2020].

[28] Oracle.com. 2021. ¿Qué es una base de datos?. [online] Available at: https://www.oracle.com/co/database/what-is-database/ [Accessed 8 February 2021].

[29] IONOS Digitalguide. 2021. Introducción al sistema gestor de base de datos (SGBD). [online] Available at: https://www.ionos.es/digitalguide/hosting/cuestiones-tecnicas/sistema-gestor-de-base-de-datos-sgbd/ [Accessed 7 February 2021].

[30] Moldenhauer, K. and Slaton, N., 1– Rice Growth and Development. 1st ed. [ebook] pp.7-14. Available at: http://www.agri971.yolasite.com/resources/RICE%20GROWTH.pdf [Accessed 10 February 2021].

[31] Dorairajan, C., 2021. What is an API? — (API) Application Program Interface Definition. [online] API Friends. Available at: https://apifriends.com/api-management/what-is-an-api/ [Accessed 11 December 2021].

[32] Speedcheck.org. 2021. ¿Qué es un RSSI?. [online] Available at: https://www.speedcheck.org/es/wiki/rssi/ [Accessed 2 January 2021].

[33] Arduino. 2018. Arduino - SPITransaction. [online] Available at: https://www.arduino.cc/en/Tutorial/SPITransaction [Accessed 18 December 2020].

[34] Random Nerd Tutorials. n.d. ESP8266 NodeMCU HTTP GET and HTTP POST with Arduino IDE — Random Nerd Tutorials. [online] Available at: https://randomnerdtutorials.com/esp8266-nodemcu-http-get-post-arduino/ [Accessed 18 December 2020].

CAPÍTULO 6

ANEXOS

6.1. Anexo A: Almacenamiento de datos

Para empezar a usar el framework se deben instalar diferentes dependencias desde la consola de comandos de Windows 10. Para que funcione correctamente previamente debe haberse instalado el programa de Python, en este caso la versión 3.8. Una vez instalado Python, se procede a crear un entorno virtual de desarrollo, para que las librerías de Python no interfieran con lo instalado por defecto en la máquina, de tal forma que se puede hacer uso de los recursos del computador. Una vez creado el ambiente virtual se realiza la instalación de django y mysql, siguiendo una serie de comandos que se deben ingresar en la consola, esto para asegurarnos de que se usaran las versiones y librerías deseadas. Lo siguiente es crear el proyecto. Una vez creado el proyecto, se debe crear una aplicación dentro de este. El proyecto general nos sirve para administrar cosas como el servidor local que se va a usar para correr el sitio web o la API a crear, pero es la configuración de la aplicación la que es más específica a lo que se quiera hacer. La creación de esta aplicación nos da acceso realmente al backend de la API, desde aquí se administra como y que información mostrar en la página web. Basta solo con registrar esta aplicación creada (En este caso "Transmi") en el proyecto mencionado anteriormente. Además, debido a que se quieren realizar peticiones HTTP para tener acceso a los datos de la base de datos sin necesidad de estar dentro del programa que se debe registrar la aplicación "rest_framework". Esto se muestra en la Figura 6.1

```
INSTALLED_APPS = [
    'django.contrib.admin',
    'django.contrib.auth',
    'django.contrib.contenttypes',
    'django.contrib.sessions',
    'django.contrib.messages',
    'django.contrib.staticfiles',
    'rest_framework',
    'Transmi',
]
```

Figura 6.1: Registro de la aplicación en el proyecto.

Django utiliza programación orientada a objetos, este tipo de programación permite agrupar datos con las mismas características o propiedades, lo que da un orden y ayuda a no ser tan redundante en el código. Estos objetos se encuentran dentro de un modelo, el cual es una clase especial de objeto de Django que permite almacenar estos datos en una base de datos. En este caso el modelo se denomina "Medicion" y dentro tiene objetos como el RSSI, Altura, distancia y fecha y hora de creación. En la Figura 6.2 se muestra la definición de este.

```
from django.db import models

class Medicion(models.Model):
 RSSI = models.IntegerField()
 altura = models.IntegerField()
 distancia = models.IntegerField()
 fecha_creacion = models.DateTimeField(auto_now_add=True)

 def __str__(self):
     return str(self.RSSI)
```

Figura 6.2: Definición del modelo "Medicion".

Pero antes de que funcione correctamente se debe crear la base de datos que estará ligada al modelo. Se puede ver cada medición que se va a tomar como una fila de la base de datos y las columnas serían los objetos creados anteriormente. Para ligar esta base al modelo se deben ingresar los datos de esta en el archivo de configuración de Django, como se indica en la Figura 6.3

```
DATABASES = {
    'default': {
        'ENGINE': 'django.db.backends.mysql',
        'NAME': 'cultivo',
        'USER': 'root',
        'PASSWORD':'Shaska12345',
        'HOST': 'localhost',
        'PORT': '3306',

    }
}
```

Figura 6.3: Definición de la base de datos.

Así, el nombre de la base de datos es cultivo, esta alojada en el servidor local creado anteriormente en la instalación de Django, y se utilizan el usuario y la clave definidos en MySQL para la administración de la base de datos. Finalmente, mediante el uso de ciertos comandos en la consola le decimos al modelo que se ligue a esta base de datos, lo que se evidencia en la Figura 6.4

id	RSSI	altura	distancia	fecha_creacion
0	0	0	0	2020-12-16 15:07:10.993414
1	0	0	0	2020-12-16 15:07:02.423793

Figura 6.4: Base de datos con el modelo ligado.

Teniendo finalmente el modelo que se va a usar conectado a la base de datos solo queda definir los aspectos de la API tipo rest que se van a necesitar. Anteriormente se agregó la aplicación "rest_framework" al archivo de configuración del proyecto, es aquí donde se va a hacer uso de esta. Para empezar, se definen como van a ser los serializadores que se van a usar. Un serializador es un código que se encarga de convertir los datos que se van a mandar a formato JSON, ideal para la transmisión de datos por internet; convierte nuestros modelos a este formato para que puedan ser guardados en la base de datos de forma correcta. En este caso, el serializador usado se enseña en la Figura 6.5:

```
from rest_framework import serializers

from Transmi.models import Medicion

class MSerializer(serializers.ModelSerializer):
    class Meta:
        model = Medicion
        fields = ('RSSI','altura','distancia','id')
```

Figura 6.5: Definición serializador usado.

Se importa el modelo creado anteriormente para que los campos necesarios sean serializados, antedicho, el RSSI de la señal, la altura a la que estén las antenas, la distancia entre

las mismas y la id para tener un orden en la base de datos. Teniendo el serializador creado se procede a la configuración de las vistas de la API rest.
Las vistas son las encargadas de manejar las solicitudes que se le hacen a la API, para dar una respuesta según lo que se requiera; es aquí donde se pueden crear, modificar o hasta borrar entradas dentro de la base de datos si es lo que se ha definido en las vistas. Así, esta es la que se encarga de procesar la solicitud HTTP que se va a mandar para, en el caso de este sistema, crear una nueva fila de datos en la base por cada paquete que se esté enviando. Para lograrlo, el código se expone en la Figura 6.6

```python
from rest_framework import status
from rest_framework.decorators import api_view
from rest_framework.response import Response
from Transmi.models import Medicion
from Transmi.serializers import MSerializer


@api_view(['GET','POST'])
def medidas_list(request, format=None):

    if request.method == 'GET':
        medidas= Medicion.objects.all()
        serializado = MSerializer(medidas, many=True)
        return Response (serializado.data)
    elif request.method == 'POST':
        serializado=MSerializer (data=request.data)
        if serializado.is_valid():
            serializado.save()
            return Response(serializado.data, status=status.HTTP_201_CREATED)
        return Response(serializado.errors, status=status.HTTP_400_BAD_REQUEST)


@api_view(['GET','PUT','DELETE'])
def medidas_detail(request, pk, format=None):
    try:
        medidas = Medicion.objects.get(pk=pk)
    except Medicion.DoesNotExist:
        return Response(status=status.HTTP_404_NOT_FOUND)

    if request.method == 'GET':
        serializado = MSerializer(medidas)
        return Response(serializado.data)

    elif request.method == 'PUT':
        serializado = MSerializer(medidas, data=request.data)
        if serializado.is_valid():
            serializado.save()
            return Response(serializado.data)
        return Response(serializado.errors, status=status.HTTP_400_BAD_REQUEST)

    elif request.method == 'DELETE':
        medidas.delete()
        return Response(status=status.HTTP_204_NO_CONTENT)
```

Figura 6.6: Definición vistas usadas para el proyecto.

Pero aún faltan definir las rutas que se quieren seguir para la petición HTTP, es decir, los urls que se van a usar para que se escoja la vista adecuada. En este caso tenemos dos vistas, una general que permite la creación de una nueva entrada en la base de datos o obtener todos los campos de la misma y la segunda que permite, específicamente, ver una entrada,

modificarla o borrarla, esto según el ID mencionado arriba; es decir, la primera nos da la opción modificar o ver cualquier valor de la base de datos mediante una sola petición, la segunda es en el caso que se quiera editar o ver un valor específico.

```
from django.urls import path
from rest_framework.urlpatterns import format_suffix_patterns
from Transmi import views

urlpatterns = [
 path('mediciones/', views.medidas_list),
 path('mediciones/<int:pk>/', views.medidas_detail),
]

urlpatterns = format_suffix_patterns(urlpatterns)
```

Figura 6.7: Definición rutas usadas en el proyecto.

Así, como se evidencia en la Figura 6.7 que se definieron dos rutas o url; la primera (*mediciones/*) permite obtener toda la lista de los datos o crear una nueva entrada y la segunda ($mediciones/ < int : pk > /$) permite el ingreso de una id específica para modificar, borrar o crear una nueva entrada en ese lugar. Una forma de verificar que las rutas y las vistas funcionen correctamente es realizando una petición HTTP de prueba, que se puede hacer fácilmente con distintos software, en este caso se usó PostMan. Antes de mostrar como es una petición cabe aclarar un aspecto del servidor. Este no tiene una ip estática, por defecto se utiliza la dirección 127.0.0.1 en el puerto 8000, pero estos valores se le pueden cambiar para poder mandar peticiones HTTP de forma remota, es decir, según la red a la que estén conectados el computador y el dispositivo del que se envía la petición así mismo se debe definir la ip sobre la que estará corriendo el servidor. Estos son los distintos Host sobre los cuales el servidor funciona, y antes de ponerlo en marcha debe definirse en el archivo de configuración del proyecto los Host a usar según la red a la que se esté conectado, como se indica en la Figura 6.8.

```
ALLOWED_HOSTS = ['127.0.0.1','localhost','192.168.1.2','192.168.1.53','10.70.16.200']
```

Figura 6.8: Definición Host permitidos por el servidor.

De esta forma “localhost”, “127.0.0.1”, “192.168.1.53” son direcciones con las cuales el servidor puede ser inicializado, es la ultima la que nos muestra que según a la red que se esté conectado así mismo debe hacerse un cambio en el nombre del Host, para que la petición remota se pueda realizar de forma correcta. Finalmente, se puede inicializar el servidor para verificar su correcto funcionamiento. En este caso el computador esta conectado a una red domestica que le da una ip de 192.168.1.53, una vez definida en la lista de host que se pueden usar procedemos a inicializar el servidor mediante el uso del comando *python manage.py runserver 192.168.1.53:8000*, como se muestra en la Figura 6.9

```
Microsoft Windows [Versión 10.0.18363.1256]
(c) 2019 Microsoft Corporation. Todos los derechos reservados.

C:\Users\zambr>cd Documents

C:\Users\zambr\Documents>cd prueba

C:\Users\zambr\Documents\Prueba>env\Scripts\activate

(env) C:\Users\zambr\Documents\Prueba>cd miproyecto

(env) C:\Users\zambr\Documents\Prueba\miproyecto>python manage.py runserver 192.168.1.53:8000
Watching for file changes with StatReloader
Performing system checks...

System check identified no issues (0 silenced).
January 06, 2021 - 16:40:38
Django version 3.0.8, using settings 'miproyecto.settings'
Starting development server at http://192.168.1.53:8000/
Quit the server with CTRL-BREAK.
```

Figura 6.9: Inicialización del servidor local.

En la Figura 6.9 se evidencia como el servidor está corriendo bajo el host de la IP asignada al computador al conectarse a una red local. A continuación, haciendo uso del software PostMan, se realizará una petición para poner unos valores en la base de datos, los cuales se verificarán después, seguimos pues las rutas creadas anteriormente y se envía la siguiente petición:

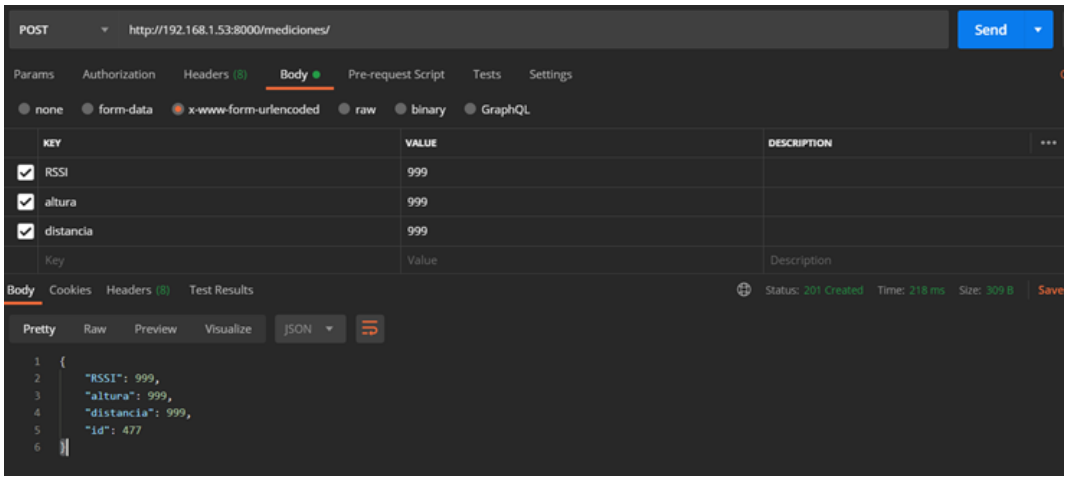


Figura 6.10: Envío de petición HTTP con software especializado.

Aunque el software permite hacerlo de forma sencilla, en realidad la petición consta de toda la ruta definida anteriormente con los valores que se quieren enviar. En el Anexo B con código final de la parte receptora del sistema se evidencia como es la estructura de la petición realmente. En la Figura 6.11 se recibió la confirmación del paquete recibido en la consola y en la 6.12 se verificó la información de este en la base de datos.

```
[06/Jan/2021 16:44:32] "POST /mediciones/ HTTP/1.1" 201 50
```

Figura 6.11: Notificación paquete recibido en el command prompt.

☑	477	999	999	999	2021-01-06 21:44:32.038479

Figura 6.12: Paquete guardado en base de datos.

Aunque el reloj interno de la base de datos y el computador no están configurados a la misma hora se evidencia como el servidor recibe y procesa correctamente las peticiones HTTP que se le envían, haciendo posible el almacenamiento en tiempo de real de los datos obtenidos a partir del experimento en una base de datos.

6.2. Anexo B: Códigos sistema transmisor y receptor

El código usado para el sistema se divide en dos, la transmisora y receptora, donde cada una se encarga de una parte importante del mismo. Mediante el uso de dos interruptores se controla el envío de paquetes(7 por cada accionamiento del interruptor) y el cambio de distancia en el sistema, el cual es un contador cuyo valor se asigna a una de las distancias que se definieron previamente en el experimento. Posteriormente, este valor del contador es la información que se transmite en el paquete enviado por el módulo LORA, para lograr tener en la base de datos el valor de la distancia que se está cambiando a medida que se desarrolla el experimento. A continuación se muestra el código usado.

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

int counter = 0;
int START=6;// Declaraci n de pines
int sdis=5;
int distancia=0;//Inicializaci n del contador
int edistancia;       // distancia
int estart;            // empezar
int prevd=HIGH;
int prevs=HIGH;



void setup() {
  Serial.begin(9600);
  while (!Serial);
// Se inicializa el m dulo LoRa
  pinMode(START,INPUT_PULLUP);
  pinMode(sdis,INPUT_PULLUP);
  Serial.println("LoRa Sender");
```

```
23
24    if (!LoRa.begin(433E6)) {
25      Serial.println("Starting LoRa failed!");
26      while (1);
27    }
28
29  }
30
31  void loop() {
32    //Se obtiene el valor de los pines digitales para empezar con el
        contador o la transmisi n
33    edistancia=digitalRead(sdis);
34    estart=digitalRead(START);
35    Serial.println(edistancia);
36    Serial.println(estart);
37    //Contador para el valor de la distancia
38    if (edistancia == LOW && prevd == HIGH){
39      distancia=distancia+1;
40      Serial.println("La distancia es: ");
41      Serial.println(distancia);
42      }
43    prevd=edistancia;
44
45    if (estart == LOW && prevs == HIGH){
46    delay(5000);
47    while (counter<=7){
48    Serial.print("La distancia es:");
49    Serial.println(distancia);
50    //A partir de aqu  se define y env a el paquete
51    LoRa.beginPacket();
52    LoRa.print(distancia);
53    LoRa.endPacket();
54
55    counter++;
56
57    delay(5000);
58    }
59    counter=0;
60      }
61    prevs=estart;
62    delay(1000);
63  }
```

Listing 6.1: Codigo TR

Para el sistema receptor se decidió usar el NodeMCU, debido al módulo Wi-Fi incluido, como se mencionó antes. En primer lugar, se conecta el módulo Wi-Fi a la red local que se está usando, además, hay que tener en cuenta la dirección web del servidor local, la cual cambia dependiendo de la red a la que se esté conectado. Se procede a inicializar el módulo LoRa, para el cual se deben tener en cuenta ciertos aspectos. Cuando se conecta un módulo

LoRa a placas como el Arduino uno, la inicialización no falla debido a que el microcontrolador sabe con qué parámetros configurar la comunicación SPI que utiliza LoRa, por otro lado, la tarjeta NodeMCU no hace esto automáticamente, razón por la cual se deben agregar ciertas líneas de código adicionales. SPISettings es un comando que permite al usuario configurar el puerto SPI mediante la declaración de 3 valores: la velocidad de comunicación, si el bit más significativo de un byte enviado es el primero o el último y el modo SPI a utilizar[33]. En este caso, se utilizó una velocidad de 10 Mhz, el bit más significativo el primero y el modo SPI 0, como se mencionó anteriormente. Una vez se consigue inicializar el módulo este siempre está listo para recibir un paquete. Dentro de esta parte del sistema se varía el valor de la altura en la que se colocan las antenas mediante un contador, se utiliza la señal piloto para conocer el valor de la distancia obtenido en la parte transmisora, se calcula el RSSI de la señal transmitida y se hace el envío de los tres datos (RSSI, altura y distancia) a la base de datos, lo que se logra por medio de una petición HTTP. Esta petición es de tipo POST, puesto que se quieren crear nuevas entradas en la base de datos[34]. A continuación se muestra el código usado.

```
#include <SPI.h>
#include <ESP8266WiFi.h>
#include <ESP8266HTTPClient.h>
#include <LoRa.h>

#define SS 15
#define RST 16
#define DI0 4
int altura=0;
int paltura=5;
int ealtu;
int preva=HIGH;
//Se define la red a la que se quiere conectar, el url del servidor al
    cual se enviaran los datos y la configuraci n SPI
const char* ssid = "El rolo"; //Enter SSID
const char* password = "internet123";
//const char* ssid = "High-Fi"; //Enter SSID
//const char* password = "Shaska12345";
const char* serverName= "http://192.168.43.114:8000/mediciones/";
SPISettings settingsA(10000000, MSBFIRST, SPI_MODE0);

void setup() {
  Serial.begin(9600);
  //se conecta a la red wi-fi
  WiFi.begin(ssid, password);
  pinMode(paltura,INPUT_PULLUP);
  while (WiFi.status() != WL_CONNECTED)
  {
     delay(500);
     Serial.print("*");
  }
  Serial.print("The IP Address of ESP8266 Module is: ");
  Serial.print(WiFi.localIP());
```

```
33    digitalWrite(SS, LOW);
34    //Se empieza la transacci n SPI para inicializar el modulo
35    SPI.beginTransaction(settingsA);
36    Serial.println("");
37    Serial.println("LoRa Receiver");
38    LoRa.setPins(SS,RST,DI0);
39      if (!LoRa.begin(433E6)) {
40      Serial.println("Starting LoRa failed!");
41      delay(100);
42 //     while (1);
43    }
44   digitalWrite(SS, HIGH);
45
46 }
47
48
49
50 void loop() {
51    ealtu=digitalRead(paltura);
52    Serial.println(ealtu);
53    //Contador para llevar el registro de la altura
54    if(ealtu == LOW && preva == HIGH){
55      delay(1000);
56      altura=altura+1;
57      Serial.println("La altura es: ");
58      Serial.println(altura);
59      }
60    preva=ealtu;
61
62    int packetSize = LoRa.parsePacket();
63    if (packetSize) {
64 // Se recibio un paquete
65      Serial.print("Received packet '");
66      // Se lee lo que viene en el paquete y se guarda en una variable
67 String LoRaData="";
68      while (LoRa.available()) {
69       LoRaData=LoRaData+(char)LoRa.read();
70      }
71 //       print RSSI of packet
72      Serial.print("La distancia es");
73      Serial.println(LoRaData);
74      Serial.print("' with RSSI ");
75      Serial.println(LoRa.packetRssi());
76      int data=LoRa.packetRssi();
77
78 //     A partir de aqu  es la petici n http
79      HTTPClient http;
80      http.begin(serverName);
81      http.addHeader("Content-Type", "application/x-www-form-urlencoded");
82      String httpRequestData= "RSSI="+String(data)+"&altura="+ String(
       altura)+"&distancia="+ LoRaData;
83      int httpResponseCode = http.POST(httpRequestData);
```

```
84      Serial.print("HTTP Response code: ");
85      Serial.println(httpResponseCode);
86      http.end();
87      }
88 }
```

Listing 6.2: Codigo RX

6.3. Anexo C: Códigos análisis de datos

```
1 # -*- coding: utf-8 -*-
2
3 import pandas as pd
4 import numpy as np
5 import seaborn as sns
6 import math as m
7 import matplotlib.pyplot as plt
8 import sklearn as sk
9 import matplotlib.transforms as mtransforms
10 from sklearn.metrics import mean_squared_error
11 from scipy.optimize import curve_fit
12 from lmfit import Model, Parameter, report_fit
13
14 mediciones=pd.read_csv(r'C:\Users\zambr\pruebas\ret2.csv')
15 mediciones1=pd.read_csv(r'C:\Users\zambr\pruebas\retpe.csv')
16 mediciones2=pd.read_csv(r'C:\Users\zambr\pruebas\retse.csv')
17 mediciones3=pd.read_csv(r'C:\Users\zambr\pruebas\rette.csv')
18 # Columnas de la base de datos en vectores para las longitudes
19 rs1=mediciones1.RSSI
20 al1=mediciones1.altura
21 dis1=mediciones1.distancia
22 rs2=mediciones2.RSSI
23 al2=mediciones2.altura
24 dis2=mediciones2.distancia
25 rs3=mediciones3.RSSI
26 al3=mediciones3.altura
27 dis3=mediciones3.distancia
28
29 #Volver los datos de la base de datos en una matriz, sacando de la media para
       cada uno
30 dpc=mediciones.pivot_table(index='altura', columns='distancia', values='RSSI',
        aggfunc='mean')
31 a=mediciones1.pivot_table(index='altura', columns='distancia', values='RSSI',
       aggfunc='mean')
32 b=mediciones2.pivot_table(index='altura', columns='distancia', values='RSSI',
       aggfunc='mean')
33 sha=mediciones3.pivot_table(index='altura', columns='distancia', values='RSSI'
       , aggfunc='mean')
34
```

```
# GRAFICA DE TODOS LOS MAPAS DE CALOR
yticklabels=[0.979,0.886,0.687,0.488,0.379,0.285,0.192]
xticklabels=[1.02,2.538,5.07,10.14,20.10,40.07]
xticklabels2=[1.02,2.538,5.07,10.14,20.10]
f,(ax1,ax2,ax3,axcb)=plt.subplots(1,4,gridspec_kw={'width_ratios'
    :[1,1,1,0.08]})
ax1.get_shared_y_axes().join(ax2,ax3)
g1=sns.heatmap(a,vmin=-120,vmax=-70,cbar=False, ax=ax1,yticklabels=yticklabels
    ,xticklabels=xticklabels)
g1.set_ylabel('Altura/m')
g1.set_xlabel('Distancia/m')
g1.set_title('(A) Macollamiento')
g2=sns.heatmap(b,vmin=-120,vmax=-70,cbar=False, ax=ax2,xticklabels=
    xticklabels2)
g2.set_ylabel('')
g2.set_xlabel('Distancia/m')
g2.set_title('(B) Alargamiento del entrenudo')
g2.set_yticks([])
g3=sns.heatmap(sha,vmin=-120,vmax=-70,ax=ax3,cbar_ax=axcb,xticklabels=
    xticklabels,cbar_kws={'label': 'RSSI/dBm'})
g3.set_ylabel('')
g3.set_xlabel('Distancia/m')
g3.set_title('(C) Producci n del grano de arroz')
g3.set_yticks([])
#Valores de distancia y altura para la prueba de la cancha
vdc=np.array([1.48,2.80,4.57,9.20,18.10])
vac=np.array([1.05,0.953,0.747,0.55,0.449,0.353,0.25])
yticklabelsc=[1.05,0.953,0.747,0.55,0.449,0.353,0.25]
xticklabelsc=[1.48,2.80,4.57,9.20,18.10]
# MAPA DE CALOR PRUEBA CANCHA
fig, mmmpc=plt.subplots(figsize=(6,5))
mmmpc=sns.heatmap(dpc,cbar=True,yticklabels=yticklabelsc,xticklabels=
    xticklabelsc,cbar_kws={'label': 'RSSI/dBm'})
mmmpc.set_ylabel("Altura/m")
mmmpc.set_xlabel("Distancia/m")

######Funcion para rmse
def rmse(prediction,target):
    return np.sqrt(((prediction-target)**2).mean())
######Funcion para error relativo
def re(prediction,target):
    return np.sqrt(((((prediction-target)/target)**2)).mean())

# Valores para las pruebas finales
vd=np.array([1.02,2.538,5.07,10.14,20.10,40.07])
vd2=np.array([1.02,2.538,5.07,10.14,20.10])
va=np.array([0.979,0.886,0.687,0.488,0.379,0.285,0.192])
gtr=5 ##ganancia antenas
c=3e8
f=433e6
gam=c/f
rad=4
```

```
s=(len(va),len(vd))
s2=(len(va),len(vd2))
#matrices con ceros para calcular modelos
mtr=np.zeros(s)
mtrse=np.zeros(s2)
mFS=np.zeros(s)
mFS2=np.zeros(s2)
mld=np.zeros(s)
mld2=np.zeros(s2)
mld3=np.zeros(s)
mmm1=np.zeros(s)
mmm2=np.zeros(s2)
mmm3=np.zeros(s)
# Volver matrices de las bases de datos con la media saca en vectores
vmm=a.values.flatten()
vmm2=b.values.flatten()
vmm3=sha.values.flatten()

pru=vd[dis1]
lva=va[al1]

pru2=vd2[dis2]
lva2=va[dis2]

pru3=vd[dis3]
lva3=va[dis3]
# Seleccion de alturas en vectores
# Altura 0,98m
epre1=vmm[0:6]
esre1=vmm2[0:5]
etre1=vmm3[0:6]
#Altura 0,69m
epre2=vmm[12:18]
esre2=vmm2[10:15]
etre2=vmm3[12:18]
# Altura 0.38
epre3=vmm[24:30]
esre3=vmm2[20:25]
etre3=vmm3[24:30]
#Altura 0.192
epre4=vmm[36:42]
esre4=vmm2[30:35]
etre4=vmm3[36:42]


## Espacio libre
xes=1
def fes(d,a):
    return -a-(20*(1.0992 + (np.log10(d)) - (gtr*0.1) - (m.log10(gam))))
# Primera etapa
pes1, escov1 =curve_fit(fes,pru,rs1,xes)
#Segunda etapa
```

```
pes2, escov2 =curve_fit(fes,pru2,rs2,xes)
#Tercera etapa
pes3, escov3 =curve_fit(fes,pru3,rs3,xes)
print(pes1,pes2,pes3)

# primera y tercera etapa
for i in range(0,len(va)):
    hpfs=va[i]
    for j in range(0,len(vd)):
        mFS[i][j]= 20 * ( 1.0992 + (m.log10(vd[j])) - (gtr*0.1) - (m.log10(gam
    )) )

# segunda etapa
for i in range(0,len(va)):
    hpfs=va[i]
    for j in range(0,len(vd2)):
        mFS2[i][j]= 20 * ( 1.0992 + (m.log10(vd2[j])) - (gtr*0.1) - (m.log10(
    gam)) )

nmFS=-pes1-mFS
nmFS2=-pes2-mFS2
nmFS3=-pes3-mFS

# Volver matrices calculadas en vectores
vffs=nmFS.flatten('C')
vffs2=nmFS2.flatten('C')
vffs3=nmFS3.flatten('C')
eprm1=vffs[0:6]
esrm1=vffs2[0:5]
etrm1=vffs3[0:6]

# Error espacio libre
rmfs=rmse(vmm, vffs)
rmfs2=rmse(vmm2, vffs2)
rmfs3=rmse(vmm3, vffs3)
refs=re(vmm, vffs)
refs2=re(vmm2, vffs2)
refs3=re(vmm3, vffs3)
# CUADRO COMPARATIVO ESPACIO LIBRE
fig, ax = plt.subplots(figsize=(6,5))
ax.scatter(vmm, vffs, marker='x',color='red', label="Primera etapa")
ax.scatter(vmm2, vffs2, marker=',', color='blue', label="Segunda etapa")
ax.scatter(vmm3, vffs3, color='green',label="Tercera etapa")
ax.legend()
rrmfs=round(rmfs,2)
rrefs=round(refs*100,2)
rrmfs2=round(rmfs2,2)
rrefs2=round(refs2*100,2)
rrmfs3=round(rmfs3,2)
rrefs3=round(refs3*100,2)
ax.text(-128,-77,r'RMSE1={};RE1={}'.format(rrmfs,rrefs), style='italic',
    fontsize=11)
```

```
ax.text(-128,-80,r'RMSE2={};RE2={}'.format(rrmfs2,rrefs2),style='italic',
    fontsize=11)
ax.text(-128,-83,r'RMSE3={};RE3={}'.format(rrmfs3,rrefs3),style='italic',
    fontsize=11)
ax.plot([0, 1],[0,1], color='black', transform=ax.transAxes)
ax.set_xlim([-130,-60])
ax.set_ylim([-130,-60])
ax.set_xlabel('RSSI medido/dBm')
ax.set_ylabel('RSSI modelado/dBm')


# Errores todas las etapas primera altura
rmfp1=rmse(epre1,eprm1)
rmfs1=rmse(esre1,esrm1)
rmft1=rmse(etre1,etrm1)
refp1=re(epre1,eprm1)
refs1=re(esre1,esrm1)
reft1=re(etre1,etrm1)

# Errores todas las etapas segunda altura
rmfp2=rmse(epre2,eprm1)
rmfs2=rmse(esre2,esrm1)
rmft2=rmse(etre2,etrm1)
refp2=re(epre2,eprm1)
refs2=re(esre2,esrm1)
reft2=re(etre2,etrm1)

# Errores todas las etapas tercera altura
rmfp3=rmse(epre3,eprm1)
rmfs3=rmse(esre3,esrm1)
rmft3=rmse(etre3,etrm1)
refp3=re(epre3,eprm1)
refs3=re(esre3,esrm1)
reft3=re(etre3,etrm1)


## Two ray
xtr=1
def ftr(X,a):
    h,d=X
    return -a-(40*np.log10(d)-20*(gtr)-40*np.log10(h))
# primera etapa
ptro,trcov=curve_fit(ftr,(lva,pru),rs1,xtr)
# segunda etapa
ptro2,trcov2=curve_fit(ftr,(lva2,pru2),rs2,xtr)
# tercera etapa
ptro3,trcov3=curve_fit(ftr,(lva3,pru3),rs3,xtr)


for i in range(0,len(va)):
    hptr=va[i]
    for j in range(0,len(vd)):
```

```
            mtr[i][j]=(40*m.log10(vd[j])-20*(gtr)-40*m.log10(hptr))

for i in range(0,len(va)):
    hptr=va[i]
    for j in range(0,len(vd2)):
        mtrse[i][j]=(40*m.log10(vd2[j])-20*(gtr)-40*m.log10(hptr))

nmtr=-ptro-mtr
nmtr2=-ptro2-mtrse
nmtr3=-ptro3-mtr
# Volver matrices calculadas en vectores
vftr=nmtr.flatten('C')
vftr2=nmtr2.flatten('C')
vftr3=nmtr3.flatten('C')

# Error two ray
rmtr=rmse(vmm, vftr)
retr=re(vmm, vftr)
rmtr2=rmse(vmm2, vftr2)
retr2=re(vmm2, vftr2)
rmtr3=rmse(vmm3, vftr3)
retr3=re(vmm3, vftr)
# CUADRO COMPARATIVO DOS RAYOS
fig, bx = plt.subplots()
bx.scatter(vmm, vftr, marker='x',color='red', label="Primera etapa")
bx.scatter(vmm2, vftr2, marker=',', color='blue', label="Segunda etapa")
bx.scatter(vmm3, vftr3, color='green',label="Tercera etapa")
bx.legend()
rrmtr=round(rmtr,2)
rretr=round(retr*100,2)
rrmtr2=round(rmtr2,2)
rretr2=round(retr2*100,2)
rrmtr3=round(rmtr3,2)
rretr3=round(retr3*100,2)
bx.text(-129,-78,r'RMSE1={};RE1={}'.format(rrmtr,rretr), style='italic',
    fontsize=11)
bx.text(-129,-81,r'RMSE2={};RE2={}'.format(rrmtr2,rretr2),style='italic',
    fontsize=11)
bx.text(-129,-84,r'RMSE3={};RE3={}'.format(rrmtr3,rretr3),style='italic',
    fontsize=11)
bx.plot([0, 1],[0,1], color='black', transform=bx.transAxes)
bx.set_xlim([-130,-60])
bx.set_ylim([-130,-60])
bx.set_xlabel('RSSI medido/dBm')
bx.set_ylabel('RSSI modelado/dBm')

# Altura 0,98 m
tprm1=vftr[0:6]
tsrm1=vftr2[0:5]
ttrm1=vftr3[0:6]
##Altura 0,488m
tprm2=vftr[12:18]
```

```
tsrm2=vftr2[10:15]
ttrm2=vftr3[12:18]
##Altura 0,192m
tprm3=vftr[24:30]
tsrm3=vftr2[20:25]
ttrm3=vftr3[24:30]
# Errores todas las etapas primera altura
rtmp1=rmse(epre1,tprm1)
rtms1=rmse(esre1,tsrm1)
rtmt1=rmse(etre1,ttrm1)
retp1=re(epre1,tprm1)
rets1=re(esre1,tsrm1)
rett1=re(etre1,ttrm1)

# # Errores todas las etapas segunda altura
rtmp2=rmse(epre2,tprm2)
rtms2=rmse(esre2,tsrm2)
rtmt2=rmse(etre2,ttrm2)
retp2=re(epre2,tprm2)
rets2=re(esre2,tsrm2)
rett2=re(etre2,ttrm2)


# # Errores todas las etapas tercera altura
rtmp3=rmse(epre3,tprm3)
rtms3=rmse(esre3,tsrm3)
rtmt3=rmse(etre3,ttrm3)
retp3=re(epre3,tprm3)
rets3=re(esre3,tsrm3)
rett3=re(etre3,ttrm3)


###Log-distance

x0=[0,0.5]
def func(d,K,n):
    return (K + 10*n*(np.log10(d)))
# Primera etapa
ptr, pcov =curve_fit(func,pru,rs1,x0)
# Segunda etapa
ptr2, pcov2 =curve_fit(func,pru2,rs2,x0)
# Tercera etapa
ptr3, pcov3 =curve_fit(func,pru3,rs3,x0)


for i in range(0,len(va)):
    hpld=va[i]
    for j in range(0,len(vd)):
        mld[i][j]= (ptr[0] + 10 * (ptr[1])*m.log10(vd[j]))
        # mld[i][j]= (ldpf['K'] + 10 * ldpf['n']*m.log10(vd[j]))

for i in range(0,len(va)):
```

```
334     hpld=va[i]
335     for j in range(0,len(vd2)):
336         mld2[i][j]= (ptr2[0] + 10 * (ptr2[1])*m.log10(vd2[j]))
337
338 for i in range(0,len(va)):
339     hpld=va[i]
340     for j in range(0,len(vd)):
341         mld3[i][j]= (ptr3[0] + 10 * (ptr3[1])*m.log10(vd[j]))
342
343 nmld=mld
344 nmld2=mld2
345 nmld3=mld3
346 # Volver matrices calculadas en vectores
347 vfld=nmld.flatten('C')
348 vfld2=nmld2.flatten('C')
349 vfld3=nmld3.flatten('C')
350
351 #Error log-distance
352 rmld=rmse(vmm, vfld)
353 reld=re(vmm, vfld)
354 rmld2=rmse(vmm2, vfld2)
355 reld2=re(vmm2, vfld2)
356 rmld3=rmse(vmm3, vfld3)
357 reld3=re(vmm3, vfld3)
358 lprm1=vfld[0:6]
359 lsrm1=vfld2[0:5]
360 ltrm1=vfld3[0:6]
361 # CUADRO COMPARATIVO LOG-DISTANCE
362 fig, cx = plt.subplots(figsize=(6,5))
363 cx.scatter(vmm, vfld, marker='x',color='red', label="Primera etapa")
364 cx.scatter(vmm2, vfld2, marker=',', color='blue', label="Segunda etapa")
365 cx.scatter(vmm3, vfld3, color='green',label="Tercera etapa")
366 cx.legend()
367 rrmld=round(rmld,2)
368 rreld=round(reld*100,2)
369 rrmld2=round(rmld2,2)
370 rreld2=round(reld2*100,2)
371 rrmld3=round(rmld3,2)
372 rreld3=round(reld3*100,2)
373 cx.text(-129,-78,r'RMSE1={};RE1={}'.format(rrmld,rreld), style='italic',
      fontsize=11)
374 cx.text(-129,-81,r'RMSE2={};RE2={}'.format(rrmld2,rreld2),style='italic',
      fontsize=11)
375 cx.text(-129,-84,r'RMSE3={};RE3={}'.format(rrmld3,rreld3),style='italic',
      fontsize=11)
376 cx.plot([0, 1],[0,1], color='black', transform=cx.transAxes)
377 cx.set_xlim([-130,-60])
378 cx.set_ylim([-130,-60])
379 cx.set_xlabel('RSSI medido/dBm')
380 cx.set_ylabel('RSSI modelado/dBm')
381
382
```

```
383
384
385 # Errores todas las etapas primera altura
386 rmlp1=rmse(epre1,lprm1)
387 rmls1=rmse(esre1,lsrm1)
388 rmlt1=rmse(etre1,ltrm1)
389 relp1=re(epre1,lprm1)
390 rels1=re(esre1,lsrm1)
391 relt1=re(etre1,ltrm1)
392
393 # Errores todas las etapas segunda altura
394 rmlp2=rmse(epre2,lprm1)
395 rmls2=rmse(esre2,lsrm1)
396 rmlt2=rmse(etre2,ltrm1)
397 relp2=re(epre2,lprm1)
398 rels2=re(esre2,lsrm1)
399 relt2=re(etre2,ltrm1)
400
401 # Errores todas las etapas tercera altura
402 rmlp3=rmse(epre3,lprm1)
403 rmls3=rmse(esre3,lsrm1)
404 rmlt3=rmse(etre3,ltrm1)
405 relp3=re(epre3,lprm1)
406 rels3=re(esre3,lsrm1)
407 relt3=re(etre3,ltrm1)
408
409 ###MI modelo ome
410 def dsig(h,d):
411     return ((2*m.pi)*(d-(np.sqrt(d**2+(4*(h**2))))))/(gam)
412
413 # Creacion matrices analisis zeros 100 pts
414 R1=np.zeros(s)
415 R2=np.zeros(s)
416 R3=np.zeros(s)
417
418
419 cc=(20*m.log10(gam/(4*m.pi)))+(gtr)
420
421 def mmod(X,ro,rm,PT):
422     h,d=X
423     ds=((2*m.pi)*((np.sqrt(d**2+(4*(h**2))))-d))/(gam)
424     R=ro
425     A=np.cos(ds)
426     B=np.sqrt(4*(h**2)+(d**2))
427     return (PT)+(cc)+10*(np.log10((1/d**2)+((2*R*A)/(d*B))+(R**2/B**2)))
428
429 mmodel=Model(mmod, independent_vars=['X'])
430 # Ajuste primera etapa
431 mresul=mmodel.fit(rs1,X=(lva,pru),ro=Parameter('ro',value=-1,min=-1,max=1),rm=
      Parameter('rm',value=-1,min=-1,max=1),PT=-100)
432 mpf=mresul.values
433 # Ajuste segunda etapa
```

```
mresul2=mmodel.fit(rs2,X=(lva2,pru2),ro=Parameter('ro',value=-1,min=-1,max=1),
    rm=Parameter('rm',value=-1,min=-1,max=1),PT=-100)
mpf2=mresul2.values
# Ajuste tercera etapa
mresul3=mmodel.fit(rs3,X=(lva3,pru3),ro=Parameter('ro',value=-1,min=-1,max=1),
    rm=Parameter('rm',value=-1,min=-1,max=1),PT=-100)
mpf3=(mresul3.values)

# Primera etapa
for i in range(0,len(va)):
    for k in range(0,len(vd)):
        mmm1[i][k]=mmod((va[i],vd[k]),mpf['ro'],mpf['rm'],mpf['PT'])

# SEGUNDA ETAPA
for i in range(0,len(va)):

    for k in range(0,len(vd2)):
        mmm2[i][k]=mmod((va[i],vd2[k]),mpf2['ro'],mpf2['rm'],mpf2['PT'])

# TERCERA ETAPA
for i in range(0,len(va)):
    for k in range(0,len(vd)):
        mmm3[i][k]=mmod((va[i],vd[k]),mpf3['ro'],mpf3['rm'],mpf3['PT'])

# Simulaci n 100 distancias y alturas
N=100
din = np.linspace(1, 100, N, endpoint=True)
ain = np.linspace(1, 100, N, endpoint=True)
frt=(len(din),len(ain))
mfi=np.zeros(frt)

for i in range(0,len(ain)):
    for k in range(0,len(din)):
        mfi[i][k]=mmod((ain[i],din[k]),mpf['ro'],mpf['rm'],mpf['PT'])
mfi2=np.zeros(frt)

for i in range(0,len(ain)):
    for k in range(0,len(din)):
        mfi2[i][k]=mmod((ain[i],din[k]),mpf2['ro'],mpf2['rm'],mpf2['PT'])

mfi3=np.zeros(frt)

for i in range(0,len(ain)):
    for k in range(0,len(din)):
        mfi3[i][k]=mmod((ain[i],din[k]),mpf3['ro'],mpf3['rm'],mpf3['PT'])

# GRAFICA SIMULACION 100 ALTURAS 100 DISTANCIAS
# fig, mtp=plt.subplots(figsize=(6,5))
# mtp=sns.heatmap(mfi,vmin=-130,vmax=-60,cbar=True,cbar_kws={'label': 'RSSI/
    dBm'})
# mtp.set_ylabel("Altura/m")
# mtp.set_xlabel("Distancia/m")
```

```
# mtp.text(4, 96, r'A',fontsize=14, fontweight='bold')
# fig, mtp2=plt.subplots(figsize=(6,5))
# mtp2=sns.heatmap(mfi2,vmin=-130,vmax=-60,cbar=True,cbar_kws={'label': 'RSSI/dBm'})
# mtp2.set_ylabel("Altura/m")
# mtp2.set_xlabel("Distancia/m")
# mtp2.text(4, 96, r'B',fontsize=14, fontweight='bold')
# fig, mtp3=plt.subplots(figsize=(6,5))
# mtp3=sns.heatmap(mfi3,vmin=-130,vmax=-60,cbar=True,cbar_kws={'label': 'RSSI/dBm'})
# mtp3.set_ylabel("Altura/m")
# mtp3.set_xlabel("Distancia/m")
# mtp3.text(4, 96, r'C',fontsize=14, fontweight='bold')


# Volver matrices calculadas en vector
vfmm=mmm1.flatten('C')
vfmm2=mmm2.flatten('C')
vfmm3=mmm3.flatten('C')
# Errores mi modelo
rmmm=rmse(vmm,vfmm)
remm=re(vmm,vfmm)
rmmm2=rmse(vmm2,vfmm2)
remm2=re(vmm2,vfmm2)
rmmm3=rmse(vmm3,vfmm3)
remm3=re(vmm3,vfmm3)

##Altura 0,98 m
mprm1=vfmm[0:6]
msrm1=vfmm2[0:5]
mtrm1=vfmm3[0:6]
##Altura 0,69m
mprm2=vfmm[12:18]
msrm2=vfmm2[10:15]
mtrm2=vfmm3[12:18]
##Altura 0,38m
mprm3=vfmm[24:30]
msrm3=vfmm2[20:25]
mtrm3=vfmm3[24:30]
##Altura 0,192m
mprm4=vfmm[36:42]
msrm4=vfmm2[30:35]
mtrm4=vfmm3[36:42]
# Errores todas las etapas primera altura
rmmp1=rmse(epre1,mprm1)
rmms1=rmse(esre1,msrm1)
rmmt1=rmse(etre1,mtrm1)
remp1=re(epre1,mprm1)
rems1=re(esre1,msrm1)
remt1=re(etre1,mtrm1)

### Errores todas las etapas segunda altura
```

```
533 rmmp2=rmse(epre2,mprm2)
534 rmms2=rmse(esre2,msrm2)
535 rmmt2=rmse(etre2,mtrm2)
536 remp2=re(epre2,mprm2)
537 rems2=re(esre2,msrm2)
538 remt2=re(etre2,mtrm2)
539
540 ### Errores todas las etapas tercera altura
541 rmmp3=rmse(epre3,mprm3)
542 rmms3=rmse(esre3,msrm3)
543 rmmt3=rmse(etre3,mtrm3)
544 remp3=re(epre3,mprm3)
545 rems3=re(esre3,msrm3)
546 remt3=re(etre3,mtrm3)
547
548 rmmp4=rmse(epre4,mprm4)
549 rmms4=rmse(esre4,msrm4)
550 rmmt4=rmse(etre4,mtrm4)
551 remp4=re(epre4,mprm4)
552 rems4=re(esre4,msrm4)
553 remt4=re(etre4,mtrm4)
554
555 # CUADRO COMPARATIVO MI MODELO
556 fig, dx = plt.subplots(figsize=(6,5))
557 dx.scatter(vmm,vfmm, marker='x',color='red', label="Primera etapa")
558 dx.scatter(vmm2,vfmm2, marker=',', color='blue', label="Segunda etapa")
559 dx.scatter(vmm3,vfmm3, color='green',label="Tercera etapa")
560 dx.legend()
561 rrmmm=round(rmmm,2)
562 rremm=round(remm*100,2)
563 rrmmm2=round(rmmm2,2)
564 rremm2=round(remm2*100,2)
565 rrmmm3=round(rmmm3,2)
566 rremm3=round(remm3*100,2)
567 dx.text(-129,-78,r'RMSE1={};RE1={}'.format(rrmmm,rremm), style='italic',
      fontsize=11)
568 dx.text(-129,-81,r'RMSE2={};RE2={}'.format(rrmmm2,rremm2),style='italic',
      fontsize=11)
569 dx.text(-129,-84,r'RMSE3={};RE3={}'.format(rrmmm3,rremm3),style='italic',
      fontsize=11)
570 dx.plot([0, 1],[0,1], color='black', transform=dx.transAxes)
571 dx.set_xlim([-130,-60])
572 dx.set_ylim([-130,-60])
573 dx.set_xlabel('RSSI medido/dBm')
574 dx.set_ylabel('RSSI modelado/dBm')
575
576
577
578 # GRAFICAS MODELO CONTRA EXPERIMENTAL ALTURA FIJA
579
580 ##########################################################################ESPACIO
      LIBRE
```

```
# PRIMERA
fig , mm=plt.subplots(figsize=(6,5))
mm.plot(vd, eprm1, '—k', label="Modelo de espacio libre")
mm.scatter(vd, epre1, marker='o', color='red', label="Altura 0.98m" )
mm.scatter(vd, epre2, marker='s', color='blue', label="Altura 0.69m" )
mm.scatter(vd, epre3, marker='^', color='green', label="Altura 0.38m" )
mm.legend()
romp1=round(rmfp1,2)
reep1=round(refp1*100,3)
romp2=round(rmfp2,2)
reep2=round(refp2*100,3)
romp3=round(rmfp3,2)
reep3=round(refp3*100,3)
mm.text(21,-81,r'RMSE1={};RE1={}'.format(romp1,reep1),fontsize=12, style='
    italic')
mm.text(21,-84,r'RMSE2={};RE2={}'.format(romp2,reep2),fontsize=12, style='
    italic')
mm.text(21,-87,r'RMSE3={};RE3={}'.format(romp3,reep3),fontsize=12, style='
    italic')
mm.text(20, -74, r'A',fontsize=14, fontweight='bold')
mm.set_xlabel('Distancia/m')
mm.set_ylabel('Potencia/dBm')
# SEGUNDA
fig , mm2=plt.subplots(figsize=(6,5))
mm2.plot(vd2, esrm1, '—k', label="Modelo de espacio libre")
mm2.scatter(vd2, esre1, marker='o', color='red', label="Altura 0.98m" )
mm2.scatter(vd2, esre2, marker='s', color='blue', label="Altura 0.69m" )
mm2.scatter(vd2, esre3, marker='^', color='green', label="Altura 0.38m" )
mm2.legend()
rosp1=round(rmfs1,2)
resp1=round(refs1*100,2)
rosp2=round(rmfs2,2)
resp2=round(refs2*100,2)
rosp3=round(rmfs3,2)
resp3=round(refs3*100,2)
mm2.text(10, -75, r'B',fontsize=14, fontweight='bold')
mm2.text(11,-85,r'RMSE1={};RE1={}'.format(rosp1,resp1),fontsize=11, style='
    italic')
mm2.text(11,-88,r'RMSE2={};RE2={}'.format(rosp2,resp2),fontsize=11, style='
    italic')
mm2.text(11,-91,r'RMSE3={};RE3={}'.format(rosp3,resp3),fontsize=11, style='
    italic')
mm2.set_xlabel('Distancia/m')
mm2.set_ylabel('Potencia/dBm')

# TERCERA
fig , mm3=plt.subplots(figsize=(6,5))
mm3.plot(vd, etrm1, '—k', label="Modelo de espacio libre")
mm3.scatter(vd, etre1, marker='o', color='red', label="Altura 0.98m" )
mm3.scatter(vd, etre2, marker='s', color='blue', label="Altura 0.69m" )
mm3.scatter(vd, etre3, marker='^', color='green', label="Altura 0.38m" )
mm3.legend()
```

```
rotp1=round(rmft1,2)
retp1=round(reft1*100,2)
rotp2=round(rmft2,2)
retp2=round(reft2*100,2)
rotp3=round(rmft3,2)
retp3=round(reft3*100,2)
mm3.text(21.5,-81,r'RMSE1={};RE1={}'.format(rotp1,retp1),fontsize=11,style='italic')
mm3.text(21.5,-84,r'RMSE2={};RE2={}'.format(rotp2,retp2),fontsize=11,style='italic')
mm3.text(21.5,-87,r'RMSE3={};RE3={}'.format(rotp3,retp3),fontsize=11,style='italic')
mm3.text(20, -75, r'C',fontsize=14, fontweight='bold')
mm3.set_xlabel('Distancia/m')
mm3.set_ylabel('Potencia/dBm')

##########################################################################################Two ray
# # PRIMERA
fig, tmm=plt.subplots(figsize=(6,5))
tmm.plot(vd, tprm1, '—r', label="Modelo primera altura")
tmm.plot(vd, tprm2, '—b', label="Modelo segunda altura")
tmm.plot(vd, tprm3, '—g', label="Modelo tercera altura")
tmm.scatter(vd, epre1, marker='o', color='red', label="Altura 0.98m" )
tmm.scatter(vd, epre2, marker='s', color='blue', label="Altura 0.69m" )
tmm.scatter(vd, epre3, marker='^', color='green', label="Altura 0.38m" )
tmm.legend()
tp1=round(rtmp1,2)
tetp1=round(retp1*100,2)
tp2=round(rtmp2,2)
tetp2=round(retp2*100,2)
tp3=round(rtmp3,2)
tetp3=round(retp3*100,2)
tmm.text(22,-76,r'RMSE1={};RE1={}'.format(tp1,tetp1), fontsize=11, style='italic')
tmm.text(22,-79,r'RMSE2={};RE2={}'.format(tp2,tetp2), fontsize=11, style='italic')
tmm.text(22,-82,r'RMSE3={};RE3={}'.format(tp3,tetp3), fontsize=11, style='italic')
tmm.text(20, -55, r'A',fontsize=14, fontweight='bold')
tmm.set_xlabel('Distancia/m')
tmm.set_ylabel('Potencia/dBm')

# # SEGUNDA
fig, tmm2=plt.subplots(figsize=(6,5))
tmm2.plot(vd2, tsrm1, '—r', label="Modelo primera altura")
tmm2.plot(vd2, tsrm2, '—b', label="Modelo segunda altura")
tmm2.plot(vd2, tsrm3, '—g', label="Modelo tercera altura")
tmm2.scatter(vd2, esre1, marker='o', color='red', label="Altura 0.98m" )
tmm2.scatter(vd2, esre2, marker='s', color='blue', label="Altura 0.69m" )
tmm2.scatter(vd2, esre3, marker='^', color='green', label="Altura 0.38m" )
tmm2.legend()
```

```
tps1=round(rtms1,2)
tetps1=round(rets1*100,1)
tps2=round(rtms2,2)
tetps2=round(rets2*100,2)
tps3=round(rtms3,2)
tetps3=round(rets3*100,2)
tmm2.text(11,-90,r'RMSE1={};RE1={}'.format(tps1,tetps1),fontsize=11, style='italic')
tmm2.text(11,-93,r'RMSE2={};RE2={}'.format(tps2,tetps2),fontsize=11, style='italic')
tmm2.text(11,-96,r'RMSE3={};RE3={}'.format(tps3,tetps3),fontsize=11, style='italic')
tmm2.text(10, -75, r'B',fontsize=14, fontweight='bold')
tmm2.set_xlabel('Distancia/m')
tmm2.set_ylabel('Potencia/dBm')


## TERCERA
fig, tmm3=plt.subplots(figsize=(6,5))
tmm3.plot(vd, ttrm1, '--r', label="Modelo primera altura")
tmm3.plot(vd, ttrm2, '--b', label="Modelo segunda altura")
tmm3.plot(vd, ttrm3, '--g', label="Modelo tercera altura")
tmm3.scatter(vd, etre1, marker='o', color='red', label="Altura 0.98m" )
tmm3.scatter(vd, etre2, marker='s', color='blue', label="Altura 0.69m" )
tmm3.scatter(vd, etre3, marker='^', color='green', label="Altura 0.38m" )
tmm3.legend()
tpt1=round(rtmt1,2)
tetpt1=round(rett1*100,2)
tpt2=round(rtmt2,2)
tetpt2=round(rett2*100,2)
tpt3=round(rtmt3,2)
tetpt3=round(rett3*100,2)
tmm3.text(21,-77,r'RMSE1={};RE1={}'.format(tpt1,tetpt1),fontsize=11, style='italic')
tmm3.text(21,-80,r'RMSE2={};RE2={}'.format(tpt2,tetpt2),fontsize=11, style='italic')
tmm3.text(21,-83,r'RMSE3={};RE3={}'.format(tpt3,tetpt3),fontsize=11, style='italic')
tmm3.text(20, -59, r'C',fontsize=14, fontweight='bold')
tmm3.set_xlabel('Distancia/m')
tmm3.set_ylabel('Potencia/dBm')


################################################################################Log distance
## PRIMERA
fig, lmm=plt.subplots(figsize=(6,5))
lmm.plot(vd, lprm1, '--k', label="Modelo log distance")
lmm.scatter(vd, epre1, marker='o', color='red', label="Altura 0.98m" )
lmm.scatter(vd, epre2, marker='s', color='blue', label="Altura 0.69m" )
lmm.scatter(vd, epre3, marker='^', color='green', label="Altura 0.38m" )
lmm.legend()
```

```
lt1=round(rmlp1,2)
letpt1=round(relp1*100,2)
lt2=round(rmlp2,2)
letpt2=round(relp2*100,2)
lt3=round(rmlp3,2)
letpt3=round(relp3*100,2)
lmm.text(23,-81,r'RMSE1={};RE1={}'.format(lt1,letpt1),fontsize=11, style='italic')
lmm.text(23,-84,r'RMSE2={};RE2={}'.format(lt2,letpt2),fontsize=11, style='italic')
lmm.text(23,-87,r'RMSE3={};RE3={}'.format(lt3,letpt3),fontsize=11, style='italic')
lmm.text(22, -73, r'A',fontsize=14, fontweight='bold')
lmm.set_xlabel('Distancia/m')
lmm.set_ylabel('Potencia/dBm')


## SEGUNDA
fig, lmm2=plt.subplots(figsize=(6,5))
lmm2.plot(vd2, lsrm1, '--k', label="Modelo log distance")
lmm2.scatter(vd2, esre1, marker='o', color='red', label="Altura 0.98m" )
lmm2.scatter(vd2, esre2, marker='s', color='blue', label="Altura 0.69m" )
lmm2.scatter(vd2, esre3, marker='^', color='green', label="Altura 0.38m" )
lmm2.legend()
lts1=round(rmls1,2)
letst1=round(rels1*100,2)
lts2=round(rmls2,2)
letst2=round(rels2*100,2)
lts3=round(rmls3,2)
letst3=round(rels3*100,2)
lmm2.text(11,-85,r'RMSE1={};RE1={}'.format(lts1,letst1),fontsize=11, style='italic')
lmm2.text(11,-88,r'RMSE2={};RE2={}'.format(lts2,letst2),fontsize=11, style='italic')
lmm2.text(11,-91,r'RMSE3={};RE3={}'.format(lts3,letst3),fontsize=11, style='italic')
lmm2.text(11.5, -78, r'B',fontsize=14, fontweight='bold')
lmm2.set_xlabel('Distancia/m')
lmm2.set_ylabel('Potencia/dBm')

## TERCERA
fig, lmm3=plt.subplots(figsize=(6,5))
lmm3.plot(vd, ltrm1, '--k', label="Modelo log distance")
lmm3.scatter(vd, etre1, marker='o', color='red', label="Altura 0.98m" )
lmm3.scatter(vd, etre2, marker='s', color='blue', label="Altura 0.69m" )
lmm3.scatter(vd, etre3, marker='^', color='green', label="Altura 0.38m" )
lmm3.legend()
ltt1=round(rmlt1,2)
lettt1=round(relt1*100,2)
ltt2=round(rmlt2,2)
lettt2=round(relt2*100,2)
ltt3=round(rmlt3,2)
```

```
lettt3=round(relt3*100,2)
lmm3.text(23,-81,r'RMSE1={};RE1={}'.format(ltt1,lettt1),fontsize=11, style='italic')
lmm3.text(23,-84,r'RMSE2={};RE2={}'.format(ltt2,lettt2),fontsize=11, style='italic')
lmm3.text(23,-87,r'RMSE3={};RE3={}'.format(ltt3,lettt3),fontsize=11, style='italic')
lmm3.text(22, -73, r'C',fontsize=14, fontweight='bold')
lmm3.set_xlabel('Distancia/m')
lmm3.set_ylabel('Potencia/dBm')

#######Mi modelo


# PRIMERA
fig, gm=plt.subplots(figsize=(6,5))
gm.plot(vd, mprm1, '—r', label="Modelo primera altura")
gm.plot(vd, mprm2, '—b', label="Modelo segunda altura")
gm.plot(vd, mprm3, '—g', label="Modelo tercera altura")
gm.scatter(vd, epre1, marker='o', color='red', label="Altura 0.98m" )
gm.scatter(vd, epre2, marker='s', color='blue', label="Altura 0.69m" )
gm.scatter(vd, epre3, marker='^', color='green', label="Altura 0.38m" )
gm.legend()
mtp1=round(rmmp1,2)
metp1=round(remp1*100,2)
mtp2=round(rmmp2,2)
metp2=round(remp2*100,2)
mtp3=round(rmmp3,2)
metp3=round(remp3*100,2)
gm.text(22,-82,r'RMSE1={};RE1={}'.format(mtp1,metp1),fontsize=11, style='italic')
gm.text(22,-85,r'RMSE2={};RE2={}'.format(mtp2,metp2),fontsize=11, style='italic')
gm.text(22,-88,r'RMSE3={};RE3={}'.format(mtp3,metp3),fontsize=11, style='italic')
gm.text(20, -67, r'A',fontsize=14, fontweight='bold')
gm.set_xlabel('Distancia/m')
gm.set_ylabel('Potencia/dBm')

# SEGUNDA
fig, gm2=plt.subplots(figsize=(6,5))
gm2.plot(vd2, msrm1, '—r', label="Modelo primera altura")
gm2.plot(vd2, msrm2, '—b', label="Modelo segunda altura")
gm2.plot(vd2, msrm3, '—g', label="Modelo tercera altura")
gm2.scatter(vd2, esre1, marker='o', color='red', label="Altura 0.98m" )
gm2.scatter(vd2, esre2, marker='s', color='blue', label="Altura 0.69m" )
gm2.scatter(vd2, esre3, marker='^', color='green', label="Altura 0.38m" )
gm2.legend()
mts1=round(rmms1,2)
mets1=round(rems1*100,2)
mts2=round(rmms2,2)
mets2=round(rems2*100,2)
```

```
mts3=round(rmms3,2)
mets3=round(rems3*100,2)
gm2.text(11,-91,r'RMSE1={};RE1={}'.format(mts1,mets1),fontsize=11, style='italic')
gm2.text(11,-94,r'RMSE2={};RE2={}'.format(mts2,mets2),fontsize=11, style='italic')
gm2.text(11,-97,r'RMSE3={};RE3={}'.format(mts3,mets3),fontsize=11, style='italic')
gm2.text(10, -80, r'B',fontsize=14, fontweight='bold')
gm2.set_xlabel('Distancia/m')
gm2.set_ylabel('Potencia/dBm')

## TERCERA
fig, gm3=plt.subplots(figsize=(6,5))
gm3.plot(vd, mtrm1, '—r', label="Modelo primera altura")
gm3.plot(vd, mtrm2, '—b', label="Modelo segunda altura")
gm3.plot(vd, mtrm3, '—g', label="Modelo tercera altura")
gm3.scatter(vd, etre1, marker='o', color='red', label="Altura 0.98m" )
gm3.scatter(vd, etre2, marker='s', color='blue', label="Altura 0.69m" )
gm3.scatter(vd, etre3, marker='^', color='green', label="Altura 0.38m" )
gm3.legend()
mtt1=round(rmmt1,2)
mett1=round(remt1*100,2)
mtt2=round(rmmt2,2)
mett2=round(remt2*100,2)
mtt3=round(rmmt3,2)
mett3=round(remt3*100,2)
gm3.text(22,-85,r'RMSE1={};RE1={}'.format(mtt1,mett1),fontsize=11, style='italic')
gm3.text(22,-88,r'RMSE2={};RE2={}'.format(mtt2,mett2),fontsize=11, style='italic')
gm3.text(22,-91,r'RMSE3={};RE3={}'.format(mtt3,mett3),fontsize=11, style='italic')
gm3.text(20, -75, r'C',fontsize=14, fontweight='bold')
gm3.set_xlabel('Distancia/m')
gm3.set_ylabel('Potencia/dBm')

# tercera altura todas las etapas
fig, gm4=plt.subplots(figsize=(6,5))
gm4.plot(vd, mprm4, '—r', label="Modelo Macollamiento")
gm4.plot(vd2, msrm4, '—b', label="Modelo Alargamiento")
gm4.plot(vd, mtrm4, '—g', label="Modelo Prod. grano")
gm4.scatter(vd, epre4, marker='o', color='red', label="Macollamiento – 0.192m" )
gm4.scatter(vd2, esre4, marker='s', color='blue', label="Alargamiento – 0.192m" )
gm4.scatter(vd, etre4, marker='^', color='green', label="Prod. grano – 0.192m" )
gm4.legend()
mt41=round(rmmp4,2)
met41=round(remp4*100,2)
mt42=round(rmms4,2)
```

```
852 met42=round(rems4*100,2)
853 mt43=round(rmmt4,2)
854 met43=round(remt4*100,2)
855 gm4.text(22,-89,r'RMSE1={};RE1={}'.format(mt41,met41),fontsize=11, style='
        italic')
856 gm4.text(22,-92,r'RMSE2={};RE2={}'.format(mt42,met42),fontsize=11, style='
        italic')
857 gm4.text(22,-95,r'RMSE3={};RE3={}'.format(mt43,met43),fontsize=11, style='
        italic')
858 gm4.set_xlabel('Distancia/m')
859 gm4.set_ylabel('Potencia/dBm')
```

pruebitasi.py