

Analizador de Suelo Basado en Lógica Difusa para Determinar el Estado de Salud de una Planta de Chile

Fuzzy Logic-Based Soil Analyzer to Determine the Health Status of a Chili Plant

Ordoñez-Carrete, R.A.¹; Lara-Cardoso, J.²; Ovalle-Gamboa, L.R.³; Hernández-Flores, C.⁴; Arjona-López, M. A.⁵

Datos de Adscripción:

¹ Ordoñez Carrete, Ricardo Alfonso
Tecnológico Nacional de México - Instituto Tecnológico Superior de Lerdo.
Departamento de Posgrado. Lerdo, Durango, México.
ROR ID: <https://ror.org/04bjbt12>
mm2310879@itslerdo.edu.mx
ORCID: <https://orcid.org/0009-0005-0669-1070>

² Lara Cardoso, Jorge
Tecnológico Nacional de México - Instituto Tecnológico Superior de Lerdo.
Departamento de Posgrado. Lerdo, Durango, México.
ROR ID: <https://ror.org/04bjbt12>
jorge_lara_c@yahoo.com
ORCID: <https://orcid.org/0000-0002-2746-7044>

³ Ovalle Gamboa, Luis Ricardo
Tecnológico Nacional de México - Instituto Tecnológico Superior de Lerdo.
Departamento de Posgrado. Lerdo, Durango, México.
ROR ID: <https://ror.org/00xbw4a70>
luis.og@itslerdo.edu.mx
ORCID: <https://orcid.org/0000-0003-4935-2376>

⁴ Hernández Flores, Concepción
TecNM Instituto Tecnológico de La Laguna. División de Estudios de Posgrado
e Investigación. Torreón, Coahuila, México.
ROR ID: <https://ror.org/00xbw4a70>
con.hernandez@ieee.org
ORCID: <https://orcid.org/0000-0002-4757-5309>

⁵ Arjona López, Marco Antonio
TecNM Instituto Tecnológico de La Laguna. División de Estudios de Posgrado
e Investigación. Torreón, Coahuila, México.
ROR ID: <https://ror.org/00xbw4a70>
marjona@ieee.org
ORCID: <https://orcid.org/0000-0003-1826-4066>

Resumen - Este trabajo presenta el desarrollo de un analizador de suelo basado en lógica difusa para determinar el estado de salud de una planta de Chile. El analizador se basa en el sensor de suelo JXBS-3001-TR, capaz de monitorear siete parámetros: pH, temperatura, humedad y conductividad eléctrica (PH-T-H-CE), así como los niveles de nutrientes de nitrógeno, fósforo y potasio (NPK). La información generada se guarda en un archivo con formato .CSV, que a su vez se almacena en una memoria SD colocada en una placa de expansión de registro de datos conectada a una tarjeta Arduino Mega2560. Así, se crea la base de datos que eventualmente se transmitirá de forma inalámbrica a través de un transceptor LoRa Reyax RYLR998, con protocolo MODBUS RTU RS485, a una computadora para su análisis mediante lógica difusa. Para facilitar el proceso de carga de la información en formato CSV, se desarrolló una interfaz gráfica intuitiva y útil, mientras que para su análisis se desarrolló un código en Python que contiene la fuzzificación, las reglas difusas, las funciones de membresía y la defuzzificación. Dentro de este programa también se compararon las lecturas experimentales con parámetros predefinidos, considerados saludables para la planta de

chile, a fin de determinar su grado de pertenencia a los estados enfermo, sano u óptimo. El presente estudio provee la base que permitirá, en un trabajo futuro, tomar acciones de control orientadas a convertir una siembra enferma en una sana y así reducir las pérdidas económicas en la cosecha de un cultivo.

Palabras Clave - Código Python, Lógica Difusa, MODBUS RS485, Planta de Chile, Sensor de Suelo, Transceptor LoRa.

Abstract - This paper presents the development of a fuzzy logic-based soil analyzer to determine the health status of a chili plant. The analyzer is based on the JXBS-3001-TR soil sensor capable of monitoring seven parameters: pH, temperature, moisture, and electrical conductivity (PH-T-M-EC), as well as the levels of nitrogen, phosphorus, and potassium (NPK) nutrients. The generated information is saved in a .CSV file, which in turn is stored on an SD card located inside a data logger shield connected to an Arduino Mega2560 board. Thus, a database is created that will eventually be wirelessly transmitted via a Reyax RYLR998 LoRa transceiver using the MODBUS RTU RS485 protocol to a computer for analysis using fuzzy logic. To facilitate loading the information in CSV format, an intuitive and useful graphical interface was developed; for analysis, Python code was developed containing fuzzification, fuzzy rules, membership functions, and defuzzification. Within this program, the experimental readings were also compared against predefined parameters considered healthy for the chili plant, thus determining its degree of belonging to the state diseased, healthy, or optimal. The present study provides the foundation for future work to take control actions oriented to convert a diseased sowing into a healthy one, thereby reducing economic losses in a crop harvest.

Keywords - Chilli Plant, Fuzzy Logic, LoRa Transceiver, MODBUS RS485, Python Code, Soil Sensor.

I. INTRODUCCIÓN

Las plagas en los cultivos son un problema que afecta el rendimiento de las cosechas, generando pérdidas millonarias para los agricultores (Oerke et al., 1996), y también afectan a la salud humana, ya que producen enfermedades (Cáceres et al., 2015). El uso creciente de químicos para controlarlas también tiene efectos negativos en la salud de los consumidores y, además, efectos secundarios como la desmineralización de la tierra, los daños al medio ambiente y la reducción de la fauna e

insectos naturalmente benéficos (Fao, 2021). Los métodos tradicionales de control de plagas, como los pesticidas y plaguicidas, también deterioran significativamente el medio ambiente y la fauna local (Savary et al., 2019). Otras soluciones, como el uso de feromonas y bioinsecticidas, conllevan menos efectos secundarios, pero tienen la desventaja de que solo combaten a un grupo específico de insectos, por lo que resulta costoso aplicarlas a la amplia variedad de plagas existentes (Phillips, 2014). Por otro lado, los insectos de control biológico, que normalmente son depredadores naturales de la plaga incidente, pueden ayudar a combatirla, pero su uso frecuente puede provocar que estos depredadores se conviertan posteriormente en otra plaga. Para combatir de manera general la plaga presente en algún cultivo, primero es necesario identificarla de forma específica en una de las plantas cultivadas (Feitó et al., 2015).

El presente trabajo se enfoca fundamentalmente en determinar el estado de salud de una planta de chile mediante la teoría de la lógica difusa. Esta investigación está organizada de la siguiente manera: en la sección II se describe el sensor de suelo utilizado para monitorear las variables de pH, temperatura, humedad y conductividad eléctrica, así como los niveles de los nutrientes nitrógeno, fósforo y potasio (Musa et al., 2024). También presenta el diseño de la fuente de voltaje de 15V utilizada para alimentarlo, así como el módulo MAX485 empleado para realizar la comunicación del sensor con una tarjeta Arduino transmisora, la cual controla un módulo Reyax RYLR998 para enviar la información de manera inalámbrica a otra Arduino receptora para finalmente almacenar los datos monitoreados en una memoria SD dentro de una placa de expansión de registro de datos. Asimismo, porciones del código desarrollado y de las librerías utilizadas, así como de la programación de lógica difusa en Python se describen en esta sección. La interfaz gráfica diseñada para facilitar la carga de los datos almacenados en los archivos CSV (comma separated values o valores separados por comas), así como los resultados obtenidos como son las gráficas de pertenencia del pH de la planta analizada y los resultados de calificación de su estado de salud se muestran en la sección III. Finalmente, las conclusiones se presentan en la sección IV.

II. PARTE TÉCNICA DEL ARTÍCULO

El sistema transmisor de información mostrado en la Figura 1 está conformado por el sensor de suelo JXBS-3001-TR, su fuente de alimentación, el adaptador MAX485, la tarjeta Arduino UNO y el módulo LoRa Reyax RYLR998 que envía los datos de manera inalámbrica.

El sistema receptor de información mostrado en la Figura 2 está compuesto por un módulo Reyax RYLR998 que recibe los datos de forma inalámbrica, el microcontrolador de una tarjeta Arduino Mega2560 y el módulo de registro de datos (data logger shield), junto con un reloj en tiempo real (RTC, real-time clock) para agregar una estampa de tiempo a las mediciones.

A continuación, se describe cada uno de los módulos que integran tanto el sistema transmisor como el sistema receptor de información:

Figura 1
Componentes del sistema transmisor.

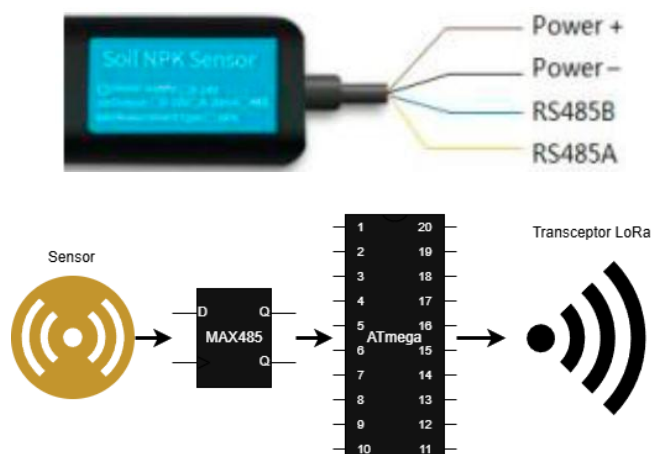
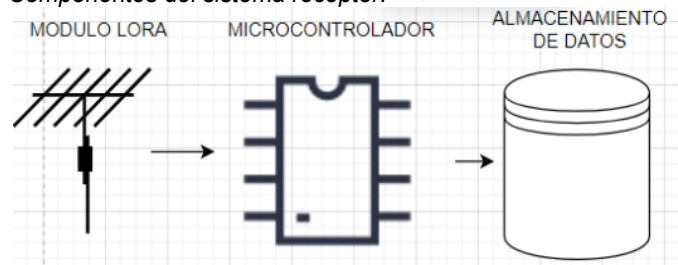


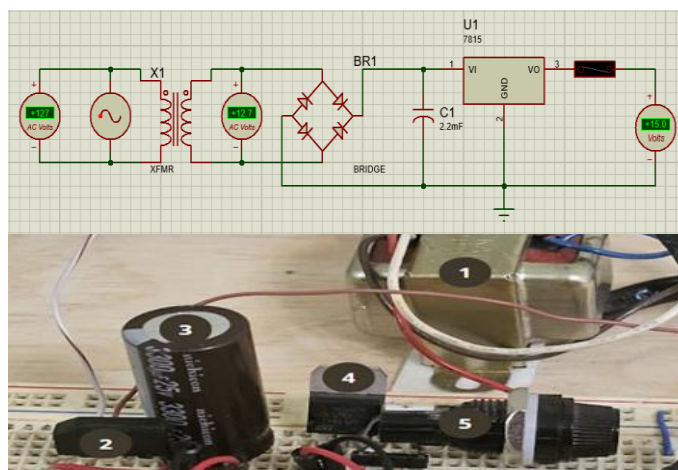
Figura 2
Componentes del sistema receptor.



2.1 Fuente de Voltaje para el Sensor de Suelo

Se implementó una fuente de voltaje de CD (corriente directa) para proveer de alimentación al sensor de suelo JXBS-3001-TR. En la Figura 3, se muestra el diseño de la fuente de alimentación que suministrará los 15V al sensor de suelo; observe tanto el circuito simulado en Proteus como el prototipo experimental.

Figura 3
Fuente de alimentación del sensor de suelo



2.2 Módulo MAX485 y Comunicación MODBUS RS485 - TTL

El sensor de suelo JXBS-3001-TR se comunica mediante el protocolo MODBUS a través de una interfaz RS-485, que transmite información a partir de señales diferenciales. Este tipo de comunicación utiliza dos conductores, comúnmente denominados A y B, que transportan señales en fase opuesta. En lugar de basarse en el voltaje respecto a tierra, el sistema interpreta los datos a partir de la diferencia de voltaje entre ambas líneas. Gracias a esto, se logra una mayor resistencia al ruido eléctrico y una transmisión de datos más estable y extensa, alcanzando distancias de hasta 1,200 metros, lo cual resulta ideal para ambientes industriales con alta interferencia electromagnética (Modbus Organization, 2006).

Debido a que el sensor de suelo tiene una interfaz RS-485 es necesario utilizar el adaptador de comunicación RS485-TTL mostrado en la Figura 4 para poder conectarlo a la tarjeta Arduino, específicamente al microcontrolador Atmega328P. Este módulo adapta las señales TX y RX en señales diferenciales y al mismo tiempo las convierte de TTL a RS485 y viceversa.

La Tabla 1 muestra los pines del módulo MAX485 junto con la función de cada uno.

La Figura 5 muestra la ventana de configuración en Modbus Poll para realizar una comunicación MODBUS RTU RS485 entre la PC y el sensor de suelo, mientras que la Figura 6 muestra la ventana en Modbus Poll para realizar la lectura de una dirección, en este caso, la 12 correspondiente al parámetro de humedad del sensor JXBS-3001-TR.

Figura 4

Módulo de comunicación MAX485.



Tabla 1

Pines del módulo MAX485.

Pin	Función
VCC	Alimentación del módulo 5 Vdc
A	Línea de transmisión de datos no invertida
B	Línea de transmisión de datos invertida
GND	Tierra
DI	Entrada de datos
DE	Habilita la transmisión de datos
RE	Habilita la recepción de datos
RO	Salida de datos

2.3 LoRa y Transmisión Inalámbrica de Datos

La tecnología que se utiliza para la transmisión inalámbrica de los datos monitoreados por el sensor de suelo es la de largo alcance LoRa (Long-Range). La especificación LoRaWAN es un estándar de red de baja potencia y amplia cobertura (LPWA) que permite la conexión inalámbrica de diversos dispositivos a internet, ya sea en redes regionales, nacionales o globales.

Figura 5

Configuración en Modbus Poll para la comunicación con el sensor.

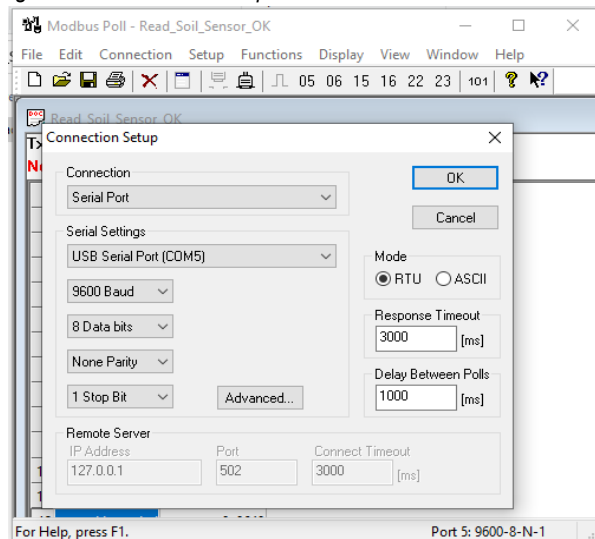
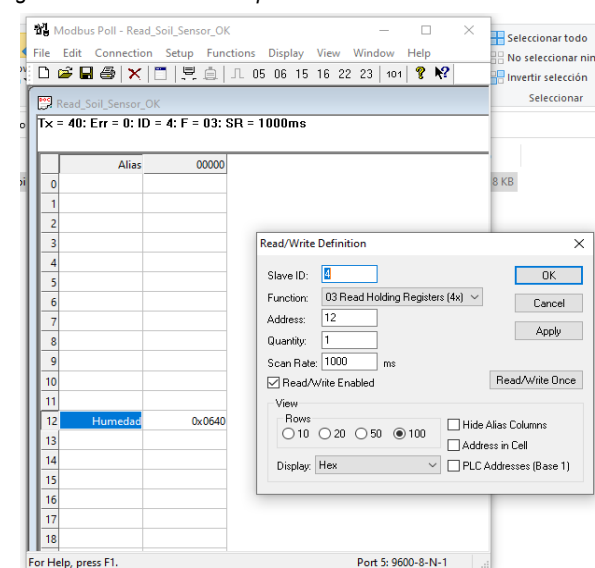


Figura 6

Configuración en Modbus Poll para leer la dirección 12 del sensor.



Esta tecnología opera en bandas de frecuencia no licenciadas y destaca por su largo alcance de hasta 15 kilómetros en zonas rurales, su bajo consumo energético y su capacidad para conectar múltiples dispositivos a una sola red (LoRa Alliance, 2023; (Mejías, 2021).

El módulo utilizado para esta transmisión de datos es el Reyax RYLR998, que se muestra en la Figura 7. Este módulo utiliza un voltaje de alimentación de 3.3 Vdc, está basado en el protocolo LoRa y se controla mediante comandos AT a través de una interfaz UART (Universal Asynchronous Receiver-Transmitter/Transmisor-Receptor Asíncrono Universal). Los datos enviados por este módulo y los recibidos posteriormente por otro módulo igual siguen una estructura específica que incluye información sobre la dirección, la longitud del mensaje, el contenido y las métricas de la señal. Gracias a este transceptor

LoRa empleado y a los programas de aplicación que pueden ser desarrollados por los usuarios, los módulos RYLR998 permiten implementar distintas arquitecturas de red, como punto a punto, punto a multipunto o multipunto a multipunto. Para lograr la comunicación entre módulos LoRa, estos deben compartir el mismo identificador de red (NETWORKID). En caso de que el valor de la dirección (ADDRESS) del receptor pertenezca a un grupo distinto, la comunicación entre dispositivos no será posible (Reyax, 2023).

Figura 7

Módulo de comunicación inalámbrica LoRa Reyax RYLR998.



Los comandos AT que se utilizan son los siguientes:

- AT+ADDRESS: Permite asignar una dirección para identificar los distintos módulos, ya sea para la transmisión o la recepción de datos.
- AT+NETWORKID: Es uno de los comandos más importantes cuando se tienen múltiples dispositivos, ya que permite la comunicación entre ellos. Si no se encuentran en la misma red, no pueden comunicarse entre sí, aun cuando tengan la misma frecuencia.
- AT+BAND: Configura la frecuencia de la banda con la cual se comunicarán los transceptores, comúnmente se utiliza una frecuencia de 868 MHz. Que todos los módulos compartan el mismo ancho de banda también es vital para que la comunicación entre ellos sea exitosa.
- AT+IPR: Configurar la tasa de baudios de la UART para poder comunicarse con el controlador en uso. En el presente trabajo, basado en una tarjeta Arduino Uno, el baudrate se configuró en 9600.
- AT+Send: Envía los datos. Se compone de la siguiente sintaxis, <Dirección>, <Longitud>, <Datos>.

La Figura 8 muestra un ejemplo de la estructura del comando AT+SEND.

Figura 8

Ejemplo del comando AT+SEND.

AT+SEND=50,5,HELLO

La estructura de los datos recibidos +RCV es la siguiente: <Dirección>, <Longitud>, <Datos>, <RSSI>, <SNR>, donde la dirección corresponde al transmisor, la longitud es la cantidad de bits de los datos, los datos son la información recibida, el RSSI es el indicador de la intensidad en dBm de la señal recibida y el SNR es la relación señal/ruido, expresada en dB.

La Figura 9 muestra un ejemplo de la estructura de los datos cuando un transceptor LoRa recibe información.

Figura 9

Ejemplo de recepción de información en un transceptor LORA.

+RCV=50, 5, HELLO, -99, 40

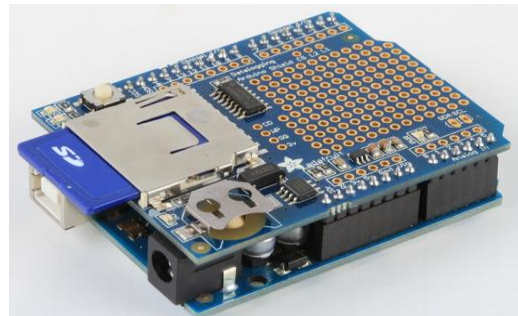
2.4 Módulo de Registro de Almacenamiento de Datos

La Figura 10 muestra un Arduino data logger shield, un módulo que amplía las capacidades de la placa Arduino, permitiéndole registrar datos de sensores u otras fuentes en una tarjeta de memoria SD para su posterior análisis. Este módulo se acopla directamente a la placa Arduino y proporciona tanto un lector de tarjetas SD como un reloj de tiempo real (RTC), lo que le permite guardar las lecturas con una marca temporal precisa (Monk, 2017).

La principal función de este shield es capturar los datos del sensor de suelo de manera continua y periódica, y almacenarlos en un archivo con formato CSV, el cual es fácilmente legible en Excel. El RTC está basado en el chip DS3231, y se alimenta con una batería de respaldo tipo botón para mantener la hora incluso cuando el Arduino pierde su alimentación, asegurando así la marca de tiempo de los datos almacenados en el registro.

Figura 10

Arduino data logger shield.



La información acumulada y almacenada en la SD se enviará semanalmente a otro transceptor LoRa conectado a una tarjeta Arduino Mega2560, que se encargará de vaciar la base de datos en una computadora para su análisis posterior. Esta modalidad de transmisión inalámbrica de información se planteó considerando el escenario cuando el sensor de suelo esté instalado en la parcela de cultivo alejado de una red de internet (Klein & Kuenzer, 2021).

Posteriormente, los datos recopilados se guardan en un archivo .CSV que contiene en sus columnas las mediciones de las variables de fecha/hora, temperatura, humedad, pH, conductividad eléctrica, nitrógeno, fósforo y potasio, tal y como se muestra en la Figura 11.

Figura 11

Contenido del archivo CSV que muestra los datos almacenados.

Hora	Temperatur	Humedad	PH	EC	Nitrogeno	Fosforo	Potasio
19/5/2025(8:	26	99	6	266	36	51	102
19/5/2025(8:	27	99	6	270	36	51	102
19/5/2025(8:	27	99	6	270	36	51	102
19/5/2025(8:	27	99	6	270	36	51	103
19/5/2025(8:	27	99	6	270	36	51	102
19/5/2025(8:	27	99	6	270	36	51	103
19/5/2025(8:	27	99	6	270	36	51	103
19/5/2025(8:	27	99	6	270	36	51	103
19/5/2025(8:	27	99	6	270	36	51	102

2.5 Programación en Arduino IDE

La programación desarrollada está dividida en dos, una para el sistema transmisor y otra para el sistema receptor. Para el primer sistema encargado del monitoreo, la recepción e interpretación de la información recopilada por el sensor y su envío comienzan con el código de las siguientes librerías:

```
1. #include <SoftwareSerial.h>
2. #include "Modbus.h"
3. #include "ModbusSerial.h"
4. int cconfig=0;
5. #define rx2 7
6. #define tx2 8
7. SoftwareSerial rserial(rx2,tx2);
```

La librería SoftwareSerial.h permite crear un puerto serie adicional en la tarjeta Arduino; en este caso, se definen los pines 7 y 8 como RX y TX, respectivamente. En estos pines se conectarán el TX y RX del módulo LoRa para así enviar inalámbricamente la información recopilada por el sensor.

```
1. void setup() {
2.   Serial.begin(9600);
3.   rserial.begin(9600);
4.   delay(2000);
5.   rserial.println("AT+ADDRESS=1");
6.   delay(500);
7.   rserial.println("AT+NETWORKID=11");
8.   delay(500);
9.   rserial.println("AT+BAND=865000000");
10.  delay(500);
11.  rserial.println("AT+IPR=9600");
12.  delay(500);
13.  verificareyax();
```

En el void setup se inicializa el puerto serial predefinido de la tarjeta, desde el cual se monitorean los valores leídos por el sensor. También se inicializa el puerto serial adicional creado para conectar la UART del módulo LoRa. Observe que es necesario configurar el "IPR" del módulo a 9600 para que puedan comunicarse correctamente. Una vez configurados los puertos seriales, se configura el módulo LoRa enviando los comandos AT descritos previamente en el apartado C. Posteriormente, se ejecuta la función llamada "verificareyax", la cual consiste en el código mostrado a continuación:

```
1. void verificareyax()
2. {
3.   while (cconfig<3)
4.   {
5.     rserial.println("AT+ADDRESS?");
6.     delay(500);
7.     rserial.println("AT+NETWORKID?");
8.     delay(500);
9.     rserial.println("AT+BAND?");
10.    delay(500);
```

```
11. rserial.println("AT+IPR?");
12. delay(500);
13. String resp=rserial.readString();
14. Serial.println(resp);
15. delay(500);
16. cconfig=cconfig+1;
17. Serial.println("Configurado");
18. }}
```

Dentro de esta función se pregunta, a través de los comandos AT enviados serialmente entre la UART y el Arduino, por la configuración del módulo LoRa. Las respectivas preguntas se realizan tres veces para corroborar que efectivamente se haya aplicado la configuración establecida en el void setup. En cuanto al void loop el código es el siguiente:

```
1. void loop() {
2.   if (Serial.available())
3.   {
4.     String datosr=Serial.readString();
5.     // Leer hasta el salto de línea
6.     datosr = Serial.readStringUntil('\n');
7.     datosr.trim(); // Eliminar espacios en blanco
8.     Serial.print("Datos Recibidos "); //mostrar datos recibidos
9.     Serial.println(datosr);
10.    rserial.println("AT+SEND=17,"+ 10.
        String(datosr.length())+" "+datosr);
11.  }
12.  else
13.  {
14.    Serial.println("Error en el MAX485");
15.    delay(10000);
16.    rserial.println("AT+SEND=17,6,Error");
17.    delay(10000);
18.  }}
```

Aquí se convierte la información recibida por el puerto serial; una vez confirmada la habilitación de la lectura serial, los datos recibidos se convierten en una variable del tipo "string" y se almacenan en la variable de los datos a enviar serialmente desde el Arduino al módulo LoRa a través del segundo puerto serial creado. Para esto se utilizan las estructuras definidas mediante los comandos AT. Observe que en lugar de establecer una longitud fija, se define datosr.length como la cantidad de bits de los datos almacenados en la variable datos r y el mensaje enviado de manera inalámbrica es "datosr".

Para el sistema receptor se utilizan las siguientes librerías:

```
1. #include <SD.h>
2. #include "RTCLib.h"
3. #include <time.h>
4. #include <SoftwareSerial.h>
```

La librería SD.h incluye las instrucciones necesarias para configurar el data logger shield, mientras que la librería RTCLib permite configurar el módulo RTC. La librería time.h sirve para configurar la hora, y finalmente la librería SoftwareSerial.h permite crear un puerto serie adicional.

```
1. #define rx3 7
2. #define tx3 8
3. SoftwareSerial rserial(rx3,tx3);
4. RTC_DS1307 rtc;
5. File dataFile;
6. bool header=false;
7. char daysOfTheWeek[7][12] = {"Sunday", "Monday", "Tuesday",
8.   "Wednesday", "Thursday", "Friday", "Saturday"};
9. const int CS_PIN= 10;
```

Las líneas de código anteriores definen constantes globales que serán utilizadas en todas las funciones del programa.

```
1. void setup() {
2.   Serial.begin(9600); rserial.begin(9600);
3.   rserial.println("AT+ADDRESS=17"); delay(500);
4.   rserial.println("AT+NETWORKID=11"); delay(500);
5.   rserial.println("AT+BAND=865000000"); delay(500);
6.   rserial.println("AT+IPR=9600"); delay(500);
7.   pinMode(CS_PIN, OUTPUT);
8.   if (!SD.begin(CS_PIN)) {
9.     Serial.println("Error al inicializar la tarjeta SD");
10.    return; delay(10000);
11.    Serial.println("Tarjeta SD inicializada correctamente");
12.    delay(1000);
13.    dataFile = SD.open("datos11.csv", FILE_WRITE);
14.    if (dataFile) {
15.      Serial.println("Archivo abierto correctamente.");
16.      dataFile.println("Hora, Temperatura, Humedad, PH, EC, Nitrógeno,
17.        Fósforo, Potasio"); // Cabecera del CSV
18.    } else {
19.      Serial.println("Error al abrir el archivo.");
20.    } if (!rtc.begin()) {
21.      Serial.println("No se pudo iniciar"); Serial.flush();
22.      while (1) delay(10);
23.    } if (!rtc.isrunning()) {
24.      Serial.println("Reloj no encontrado");
25.      rtc.adjust(DateTime(F(__DATE__), F(__TIME__)));
```

Las líneas anteriores muestran las instrucciones de configuración a ejecutar una sola vez durante el setup e incluyen la inicialización de los puertos seriales, la configuración del módulo UART y la configuración del data logger shield, donde primero se verifica que exista comunicación con la Arduino y, si sí la hay, se imprime serialmente un mensaje de que se inició correctamente. Posteriormente, se crea un archivo CSV, con los encabezados de las variables a guardar, como son la hora, la temperatura, la humedad, etc. y se cierra el archivo. Por último, se configura el reloj, se verifica que esté avanzando y, si no, se puede guardar la fecha y la hora en la memoria flash del Arduino.

En la función void loop se ejecuta lo siguiente:

```
1. void loop() {
2.   if (rserial.available() > 0) {
3.     // Leer los datos de la comunicación serial
4.     datosr = rserial.readStringUntil('\n'); // Leer hasta salto línea
5.     datosr.trim(); // Eliminar espacios en blanco
6.     int primer_coma = datosr.indexOf(',');
7.     int segunda_coma = datosr.indexOf(',', primer_coma)+1;
8.     int inicio_datos = datosr.indexOf(',', segunda_coma)+1;
9.     String datosp = datosr.substring(inicio_datos);
10.    datosp.trim();
11.    int ultima_coma = datosp.lastIndexOf(',');
12.    int penultima_coma = datosp.lastIndexOf(',', ultima_coma-1);
13.    datosp = datosp.substring(0, penultima_coma);
14.    if (datosr.length() > 0) {
15.      dataFile = SD.open("datos11.csv", FILE_WRITE);
16.      if (dataFile) {
17.        dataFile.println(datosp); // Guardar datos recibidos en nueva línea
18.        dataFile.close(); // Cerrar el archivo
19.        Serial.print("Datos guardados: ");
20.        Serial.println(datosp); } else {
21.        Serial.println("Error al abrir el archivo para escribir.");
```

Donde primero se evalúa si el puerto serial de la UART está abierto, y si se guarda la información en la variable "datosr". Dado que el dato recibido tiene el formato mostrado en la Figura 7, se procede a eliminar los datos antes y después de ciertas comas, para que queden solamente los datos útiles de la lectura del

sensor, los cuales se almacenan en la variable "datosp". Posteriormente, se confirma que la variable no esté vacía y se almacena en el archivo CSV previamente creado para luego almacenarlo en la SD.

2.6 Programación de Lógica Difusa en Python

La programación de la lógica difusa se realizó en el lenguaje de Python utilizando la IDE de Visual Studio Code, la cual no solo permite programar y crear notebooks de Python sino también ejecutar las líneas de código de manera individual. La versión utilizada para este trabajo es la versión Python 3.12, mientras que las librerías implementadas son las siguientes:

- Numpy
- Pandas
- Skfuzzy
- Matplotlib
- Tkinter

Estas librerías son fundamentales para la aplicación desarrollada en base al código de Python, la cual, resumida a grandes rasgos, consiste en una interfaz gráfica de usuario (GUI - graphical user interface) que permite comparar dos archivos CSV, donde uno de ellos contiene los datos de la planta de chile sana. De este primer archivo se obtienen los valores para los límites de las funciones de membresía de tipo trapezoidal que se utilizaron para definir la pertenencia de los conjuntos de las plantas analizadas. El segundo archivo CSV, contiene los datos de la planta de chile bajo prueba y para la cual se analizará su estado de salud. El resultado de salida de la GUI desarrollada es otro archivo CSV que muestra la puntuación de salud obtenida por la planta analizada para cada uno de los conjuntos de valores monitoreados, así como el estado de salud sana o enferma. En caso de resultar enferma, también se detalla qué parámetro está generando la deficiencia identificada.

En la Figura 12 se muestra parte del código implementado para definir la función de membresía del nivel del nutriente del fósforo, para la cual se consideraron tres tipos de pertenencia: bajo, óptimo y alto.

Figura 12

Funciones de membresía para el fósforo.

```
# Funciones de membresía para Fósforo
p_mean = means['fosforo']
fosforo['bajo'] = fuzz.trapmf(fosforo.universe, [1e-6, 1e-6, p_mean*0.8, p_mean])
fosforo['optimo'] = fuzz.trapmf(fosforo.universe, [p_mean*0.8, p_mean, p_mean, p_mean*1.2])
fosforo['alto'] = fuzz.trapmf(fosforo.universe, [p_mean, p_mean*1.2, p_mean*1.2, 1e6])
```

Una vez definidas las 7 funciones de membresía correspondientes a los 7 parámetros del sensor de suelo: pH, temperatura, humedad, conductividad eléctrica, nitrógeno, fósforo y potasio, se establece la base de reglas que permitirá analizar la información y, posteriormente, determinar el estado de salud de la planta bajo prueba.

La Figura 13 muestra las líneas de código en Python que definen las reglas para implementar un sistema de deficiencias, donde, a partir de la pertenencia de los valores de temperatura y humedad, se puede identificar la falta o el exceso de agua.

Figura 13

Base de reglas para el sistema de deficiencias de

Reglas para el sistema de deficiencias

```
rule6 = ctrl.Rule(humedad['bajo'] | temperatura['alta'], deficiencias['falta_agua'])
rule7 = ctrl.Rule(humedad['alto'] | temperatura['baja'], deficiencias['exceso_agua'])
rule8 = ctrl.Rule(
    humedad['optimo'] & temperatura['optima'] & ec['optimo'],
    deficiencias['ninguna'])
```

Todo el proceso matemático de la lógica difusa se lleva a cabo con la librería Skfuzzy, donde solo se requiere definir las funciones de membresía y las bases de reglas, y así evaluar los resultados de pertenencia de cada variable analizada, tal y como se muestra en la Figura 14.

Figura 14

Evaluación en Python del estado de salud de la planta de chile.

```
# Evaluación del estado de salud
for param in ['PH', 'Nitrogeno', 'Fosforo', 'Potasio']:
    if pd.isna(row[param]):
        raise ValueError(f"Valor faltante en {param}")

fuzzy_systems['salud_sim'].input['PH'] = row['PH']
fuzzy_systems['salud_sim'].input['Nitrogeno'] = row['Nitrogeno']
fuzzy_systems['salud_sim'].input['Fosforo'] = row['Fosforo']
fuzzy_systems['salud_sim'].input['Potasio'] = row['Potasio']

fuzzy_systems['salud_sim'].compute()
salud_score = fuzzy_systems['salud_sim'].output['Salud']
salud_status = 'Sana' if salud_score > 40 else 'Enferma'
```

III. RESULTADOS Y DISCUSIÓN

Los resultados finales se presentan y se discuten en esta sección.

La Figura 15 muestra el sensor de suelo enterrado en la maceta de la planta de chile analizada, mientras que la Figura 16 presenta la GUI desarrollada, que permite cargar archivos CSV de forma fácil y rápida.

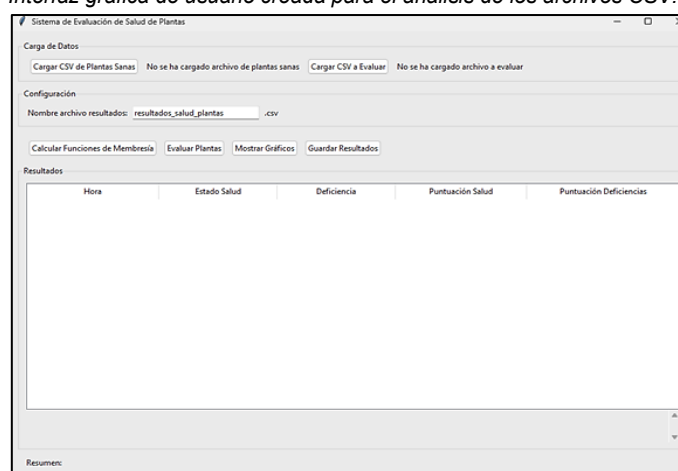
Figura 15

Sensor de suelo enterrado en la maceta de la planta de chile.



Figura 16

Interfaz gráfica de usuario creada para el análisis de los archivos CSV.



Esta GUI consiste en los siguientes 6 botones:

- 1) Botón "Cargar CSV de planta sana". En este archivo se definen las condiciones óptimas de la planta que se requiere analizar; puede ser cualquiera, pero en este caso se centra en la planta de chile.
- 2) Botón "Cargar CSV de planta a analizar". Este archivo contiene las mediciones del sensor de suelo obtenidas de la planta bajo prueba, con las cuales se determinará su condición de salud: enferma, sana u óptima.
- 3) Botón "Calcular funciones de membresía". Aquí se calculan las medias de las funciones de membresía correspondientes a los siete parámetros medidos por el sensor de suelo. La Figura 17 muestra el resultado obtenido al presionar este botón.
- 4) Botón "Evaluar plantas". Ejecuta todo el análisis matemático con base en la lógica difusa programada.
- 5) Botón "Mostrar gráficos". Despliega las gráficas de pertenencia de las siete variables monitoreadas en la planta de chile, incluyendo también las condiciones óptimas correspondientes. La Figura 18 muestra la gráfica de pertenencia del parámetro pH. Se pueden observar las funciones trapezoidales tanto la izquierda, que define la acidez del suelo, como la derecha, que define su alcalinidad. En medio, se tiene una función triangular que indica qué tan óptimo es el pH del suelo, considerando 6.0 como el valor ideal. A partir del análisis de pertenencia de cada variable del sensor de suelo, se determina la gráfica de deficiencias que se muestra en la Figura 19.
- 6) Botón "Guardar resultados". Crea un archivo CSV de salida que contenga los resultados de la planta de chile analizada, y donde, fila por fila, para cada conjunto de las lecturas realizadas, se despliegue su condición de salud obtenida, ya sea enferma, sana u óptima, tal y como se muestra en la Figura 20. En este caso, la planta de chile analizada obtuvo una puntuación de salud de 17.06 y una puntuación de deficiencias de 87.18, por lo que se determinó que está enferma principalmente debido al exceso de agua.

Es importante mencionar que estos seis botones deben presionarse en la secuencia ascendente descrita anteriormente (1-6), pues si no se ejecutan en este orden, se mostrará un mensaje de error y la ventana de los gráficos aparecerá en blanco.

Figura 17

Valores medios de las funciones de pertenencia de los siete parámetros monitoreados de la planta de chile.

Medias calculadas:	
ph:	6.00
nitrogeno:	36.00
fosforo:	50.38
potasio:	101.46
humedad:	99.00
temperatura:	26.03
ec:	264.83

Figura 18

Gráfica de pertenencia del pH de la planta de chile.

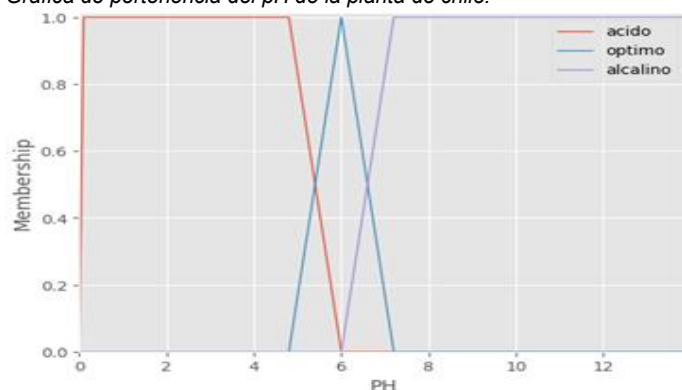


Figura 19

Gráfica de deficiencias de la planta de chile.

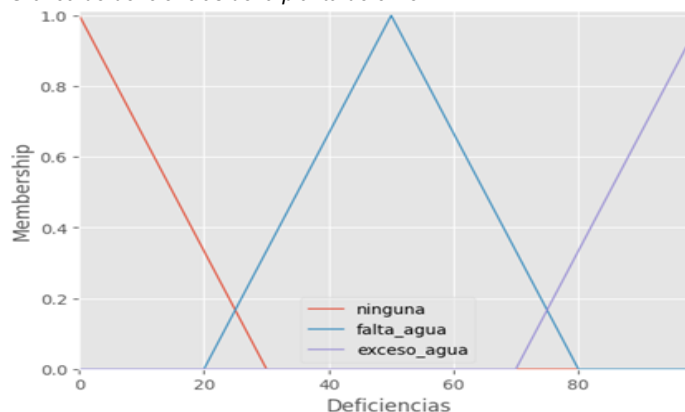


Figura 20

CSV de resultados con calificación del estado de salud.

Resultados					
Hora	Estado Salud	Deficiencia	Puntuación Salud	Puntuación Deficiencias	
9/5/2025(16:12:28)	Enferma	Exceso agua	17.06	87.10	
9/5/2025(16:12:46)	Enferma	Exceso agua	17.06	87.10	
9/5/2025(16:13:3)	Enferma	Exceso agua	17.06	87.10	
9/5/2025(16:13:21)	Enferma	Exceso agua	17.06	87.10	
9/5/2025(16:13:38)	Enferma	Exceso agua	17.06	87.10	
9/5/2025(16:13:56)	Enferma	Exceso agua	17.06	87.10	
9/5/2025(16:14:14)	Enferma	Exceso agua	17.06	87.10	
9/5/2025(16:14:31)	Enferma	Exceso agua	17.06	87.10	
9/5/2025(16:15:6)	Enferma	Exceso agua	17.06	87.10	

IV. CONCLUSIONES

Este trabajo presentó el desarrollo de un analizador de suelo basado en lógica difusa para determinar el estado de salud de una planta de chile. La integración del sensor de suelo y el sistema de transmisión LoRa Reyax RYLR998 resultó ser una solución eficiente y confiable para el monitoreo remoto de las condiciones del suelo. Asimismo, la conexión de la tarjeta Arduino Mega2560 con el módulo MAX485 permitió la compatibilidad en hardware para la lectura de los datos provenientes del sensor JXBS-3001-TR. El data logger shield permitió almacenar correctamente los datos en una tarjeta SD y luego transmitirlos y guardarlos en una computadora mediante la creación de archivos en formato CSV para su posterior análisis. El sistema desarrollado fue capaz de monitorear en tiempo real las variables de temperatura, humedad, pH y conductividad eléctrica, así como los niveles de los nutrientes nitrógeno, fósforo y potasio. La programación en Python, junto con la lógica difusa implementada para analizar la información del suelo, permitió realizar un diagnóstico eficaz del estado de salud de la planta de chile y clasificarla como enferma, sana u óptima, además de identificar las deficiencias presentes.

La solución propuesta presenta escalabilidad, bajo consumo de energía y adaptabilidad a diversas condiciones agrícolas y ambientales. Con una interfaz gráfica intuitiva y herramientas de visualización eficientes, el sistema implementado representa una herramienta práctica y accesible para investigadores, agrónomos y, eventualmente, agricultores, para que puedan, a futuro, tomar acciones orientadas a mantener un cultivo sano y reducir pérdidas económicas durante la cosecha.

V. AGRADECIMIENTOS

Los autores agradecen al Instituto Tecnológico Superior de Lerdo (ITSL) el apoyo brindado para el desarrollo del presente trabajo de investigación. Asimismo, se agradece al Departamento de Ingeniería Ambiental por el apoyo brindado al facilitar el sensor de suelo JXBS 3001-TR, componente fundamental para el desarrollo de este trabajo.

VI. REFERENCIAS

- Cáceres, F., Camilo, A., Amaya Hurtado, D., & Ramos Sandoval, O. L. (2015). Procesamiento de imágenes para reconocimiento de daños causados por plagas en el cultivo de Begonia semperflorens (flor de azúcar). *Acta Agronómica*, 64(3), 273–279. <https://doi.org/10.15446/acag.v64n3.42657>
- FAO. (2021). Revisión científica del impacto del cambio climático en las plagas de las plantas. In *FAO en nombre de la Secretaría de la CIPF* eBooks. <https://doi.org/10.4060/cb4769es>
- Feitó Cespón, M., Martínez Curbelo, G., Covas Varela, D., y Barrera García, A. (2015). Temperature control for preventing postharvest plagues in grain conservation. *Ingeniería y Desarrollo*, 33(2), 216–237. <https://doi.org/10.14482/inde.33.2.6281>
- Klein, I., Oppelt, N., y Kuenzer, C. (2021). Application of Remote Sensing Data for Locust Research and Management—A Review. *Insects*, 12(3), 233. <https://doi.org/10.3390/insects12030233>

- Kumar, N., Kumar, N. P., y Flammini, F. (2023). A Comparative Performance Evaluation of Various Classification Models for Detection and Classification of Flying Insects. *Interdisciplinary Description of Complex Systems*, 21(1), 52–68. <https://doi.org/10.7906/indexs.21.1.5>
- LoRa Alliance. (2023). LoRaWAN Specification. Recuperado de <https://loraalliance.org/>
- Mejías Rojas, D. A. (2021). Diseño de un Protocolo de Redes Mesh basado en LoRa [Trabajo Fin de Máster, Universitat Politècnica de València]. *RiuNet*. <http://hdl.handle.net/10251/173566>
- Modbus Organization. (2006). *MODBUS over Serial Line: Specification and Implementation Guide V1.02*. https://modbus.org/docs/Modbus_over_serial_line_V1_02.pdf
- Monk, S. (2017). *Programming Arduino: Getting Started with Sketches* (2nd ed.). McGraw-Hill Education.
- Musa, P., Sugeru, H., y Wibowo, E. P. (2024). Wireless Sensor Networks for Precision Agriculture: A Review of NPK Sensor Implementations. *Sensors*, 24(1), 51. <https://doi.org/10.3390/s24010051>
- Oerke, E.-C., Dehne, H.-W., Schonbeck, F. y Weber, A. (1996). Crop production and crop protection: Estimated losses in major food and cash crops. *Agricultural Systems, Elsevier*, vol. 51(4), 493-495. <https://doi.org/10.1016/C2009-0-00683-7>
- Phillips, A. J. (2014). Integrated sensing of soil moisture at the field-scale: Measuring, modeling and sharing for improved agricultural decision support. *Master's thesis, University of Calgary, Calgary, Canada*. <https://doi.org/10.11575/PRISM/26574>
- Reyax. (23 de octubre de 2023). *REYAX RYLR998/RYLR498 LoRa Transceiver Module User Manual*. <https://reyax.com/products/RYLR998>
- Savary, S., Willocquet, L., Pethybridge, S. J., Esker, P., McRoberts, N., y Nelson, A. (2019). The global burden of pathogens and pests on major food crops. *Nat. Ecol. Evol.* 3:430-439. <https://doi.org/10.1038/S41559-018-0793-Y>

No.	Rol	Descripción
1	Conceptualización [Conceptualization]	Ricardo Alfonso Ordoñez Carrete, Jorge Lara Cardoso, Luis Ricardo Ovalle Gamboa, Concepción Hernández Flores, Marco Antonio Arjona López
2	Curación de datos [Data curation]	Ricardo Alfonso Ordoñez Carrete
3	Análisis formal [Formal analysis]	Ricardo Alfonso Ordoñez Carrete, Luis Ricardo Ovalle Gamboa, Concepción Hernández Flores
4	Obtención de fondos [Funding acquisition]	Jorge Lara Cardoso, Concepción Hernández Flores, Marco Antonio Arjona López
5	Investigación [Investigation]	Ricardo Alfonso Ordoñez Carrete, Jorge Lara Cardoso, Marco Antonio Arjona López
6	Methodología [Methodology]	Jorge Lara Cardoso, Luis Ricardo Ovalle Gamboa
7	Administración del proyecto [Project administration]	Jorge Lara Cardoso, Luis Ricardo Ovalle Gamboa
8	Recursos [Resources]	Jorge Lara Cardoso, Marco Antonio Arjona López
9	Software [Software]	Ricardo Alfonso Ordoñez Carrete, Luis Ricardo Ovalle Gamboa, Concepción Hernández Flores
10	Supervision [Supervision]	Jorge Lara Cardoso, Luis Ricardo Ovalle Gamboa, Marco Antonio Arjona López
11	Validación [Validation]	Ricardo Alfonso Ordoñez Carrete, Jorge Lara Cardoso, Luis Ricardo Ovalle Gamboa, Concepción Hernández Flores, Marco Antonio Arjona López
12	Visualización [Visualization]	Ricardo Alfonso Ordoñez Carrete, Jorge Lara Cardoso, Luis Ricardo Ovalle Gamboa, Concepción Hernández Flores, Marco Antonio Arjona López
13	Redacción - borrador original [Writing – original draft]	Ricardo Alfonso Ordoñez Carrete
14	Redacción - revisión y edición [Writing – review & editing]	Ricardo Alfonso Ordoñez Carrete, Jorge Lara Cardoso, Luis Ricardo Ovalle Gamboa, Concepción Hernández Flores, Marco Antonio Arjona López