# Implementación de una Estación Meteorológica con transmisión de datos vía Internet Satelital

**Diego Florez Galvis, Charli Steve Anzola Ocampo**

**Trabajo de grado para optar el título de Profesional de Ingeniero de Telecomunicaciones**

**Director**

**Francisco Javier Dietes Cárdenas**

**Doctorado en Educación**

**Codirector**

**Elvis Humberto Galvis Serrano**

**Maestría en Redes y Sistemas de Comunicaciones**

**Universidad Santo Tomas, Bucaramanga**

**División de Ingenierías y Arquitectura**

**Facultad de Ingeniería de Telecomunicaciones**

**2026**

**Tabla de contenido**

## Lista de tablas

## Lista de figuras

**Resumen**

El presente proyecto presenta el diseño, construcción y validación de una estación meteorológica autónoma con transmisión de datos vía internet satelital Starlink, orientada al monitoreo de condiciones climáticas en zonas rurales del municipio de Lebrija, Santander, Colombia, integrando un Arduino Uno R3 equipado con el SparkFun Weather Shield para la adquisición de siete variables meteorológicas como la temperatura, humedad relativa, presión atmosférica, velocidad y dirección del viento, precipitación y radiación solar, un minicomputador Raspberry Pi 4 como controlador principal de la arquitectura de la estación meteorológica y nodo de comunicaciones y un sistema de alimentación autónoma compuesto por panel solar fotovoltaico, controlador de carga Steca Solarix PRS 1010 y batería VRLA Magna MA 35-12 de 35 Ah. En materia de implementación, la transmisión de datos se realiza mediante el protocolo MQTT sobre la red de baja órbita terrestre (LEO) Starlink, con latencias de 20 a 60 ms, junto con un sistema web desarrollado con FastAPI, SQLite y React 18 el cual permite la visualización gráfica en tiempo real, la consulta de series temporales históricas y la gestión de la información mediante una API REST y un canal WebSocket. Los servicios se despliegan sobre la Raspberry Pi mediante Docker Compose y las pruebas de desempeño verificaron la integridad de los nueve campos del flujo de datos, una latencia extremo a extremo inferior a 1,1 segundos, tiempos de respuesta de la API inferiores a 200 ms y una autonomía energética superior a 40 horas sin generación solar. Los resultados demuestran la viabilidad técnica de la conectividad satelital LEO para aplicaciones IoT agrícolas en territorios con brecha digital rural.

*Palabras clave*: estación meteorológica, internet satelital, Starlink, Raspberry Pi, MQTT, IoT agrícola, monitoreo en tiempo real, energía solar, FastAPI, zonas rurales.

**Abstract**

This project presents the design, construction, and validation of an autonomous weather station with data transmission via Starlink satellite Internet, aimed at monitoring weather conditions in rural areas of the municipality of Lebrija, Santander, Colombia. The system integrates an Arduino Uno R3 equipped with the SparkFun Weather Shield for the acquisition of seven meteorological variables, including temperature, relative humidity, atmospheric pressure, wind speed and direction, precipitation, and solar radiation. A Raspberry Pi 4 minicomputer serves as the main controller of the weather station architecture and as the communication node, while an autonomous power supply system consists of a photovoltaic solar panel, a Steca Solarix PRS 1010 charge controller, and a 35 Ah Magna MA 35-12 VRLA battery. For implementation, data transmission is performed using the MQTT protocol over the Starlink Low Earth Orbit (LEO) network, achieving latencies between 20 and 60 ms. In addition, a web-based system developed with FastAPI, SQLite, and React 18 enables real-time graphical visualization, historical time-series queries, and information management through a REST API and a WebSocket channel. The services are deployed on the Raspberry Pi using Docker Compose. Performance tests verified the integrity of the nine fields in the data stream, an end-to-end latency of less than 1.1 seconds, API response times below 200 ms, and an energy autonomy exceeding 40 hours without solar generation. The results demonstrate the technical feasibility of LEO satellite connectivity for agricultural IoT applications in rural areas affected by the digital divide.

*Keywords:* weather station, satellite Internet, Starlink, Raspberry Pi, MQTT, agricultural IoT, real-time monitoring, solar energy, FastAPI, rural areas.

## Introducción

La agricultura en el departamento de Santander, Colombia, enfrenta un desafío estructural que combina la variabilidad climática propia de la región andina con la ausencia histórica de infraestructura de telecomunicaciones en sus zonas rurales de acuerdo a los datos proporcionados por el Departamento Administrativo Nacional de Estadística (DANE), solo el 45,7% de los hogares en Santander posee internet fijo [1], donde los municipios del occidente santandereano, entre ellos Lebrija, concentran sistemas productivos de aguacate, cacao, café, mandarina y plátano en altitudes que oscilan entre los 500 y 1.500 metros sobre el nivel del mar, y las decisiones agronómicas, riego, fumigación, cosecha, dependen directamente de las condiciones meteorológicas locales, sin embargo, la ausencia de redes de monitoreo climático en el nivel predial y la baja cobertura de conectividad terrestre han limitado históricamente el acceso de los productores rurales a información meteorológica precisa y oportuna.

El presente proyecto responde a esta problemática mediante la implementación de una estación meteorológica de bajo costo, energéticamente autónoma y con transmisión continua de datos a través de la red satelital de baja órbita terrestre Starlink, a diferencia de los satélites geoestacionarios convencionales, cuyas latencias de 600 a 800 ms los hacen inadecuados para aplicaciones interactivas, los satélites LEO de Starlink operan a altitudes de entre 340 y 570 km, ofreciendo latencias de 20 a 60 ms que hacen viable el streaming de datos meteorológicos en tiempo real.

El sistema desarrollado está conformado por un conjunto de componentes de hardware y software que permiten la adquisición, procesamiento, transmisión, almacenamiento y visualización de la información meteorológica. La captura de las variables ambientales es realizada por un Arduino Uno R3 en conjunto con la SparkFun Weather Shield, mientras que la Raspberry

Pi 4 desempeña el papel de unidad principal de procesamiento y gestión de las comunicaciones. Los datos recopilados por el Arduino son transmitidos a la Raspberry Pi mediante comunicación serial a una velocidad de 115200 baudios, utilizando el formato JSON. En este dispositivo se ejecuta una aplicación desarrollada en Python, encargada de recibir la información y publicarla en un broker MQTT, facilitando su distribución al resto de los componentes del sistema. La arquitectura se complementa con un backend implementado en FastAPI, una base de datos SQLite para el almacenamiento de las mediciones y un dashboard web desarrollado en React 18, el cual permite la consulta y visualización de los datos en tiempo real. Adicionalmente, la estación dispone de un sistema de alimentación basado en energía fotovoltaica, que garantiza su funcionamiento autónomo. Finalmente, el sistema fue implementado y evaluado en un predio rural ubicado en la vereda Llanadas, municipio de Lebrija, Santander, destinado al cultivo de aguacate de las variedades Choqué y Lorena.

El documento se organiza con el capítulo 1el cual presenta la formulación del problema; el capítulo 2 expone la justificación del proyecto; el capítulo 3 define los objetivos general y específicos; el capítulo 4 desarrolla el marco referencial sobre observación meteorológica, comunicaciones satelitales y tecnologías de hardware y software empleadas; el capítulo 5 describe la metodología en cuatro fases; el capítulo 6 presenta el desarrollo de cada objetivo específico, desde la caracterización agroclimática de la zona hasta la comprobación del desempeño del sistema sobre la red Starlink; y los capítulos 7 y 8 recogen las conclusiones y las líneas de trabajo futuro, respectivamente.

En este trabajo se diseñó e implementó una estación meteorológica IoT para el monitoreo agroclimático en la zona rural de Lebrija, Santander. El sistema integra sensores para medir variables meteorológicas fundamentales, una plataforma web para la visualización y gestión de

datos en tiempo real con conectividad satelital mediante Starlink. Las pruebas realizadas demostraron la adquisición continua y confiable de información, una baja latencia en la transmisión de datos y una alta disponibilidad del sistema, validando su viabilidad como herramienta de apoyo para la toma de decisiones agronómicas en entornos rurales con acceso limitado a infraestructura de comunicaciones.

## 1. Formulación del problema

El problema principal es la dificultad de transmisión de datos de la estación meteorológica por problemas de conectividad en regiones donde la cobertura normal de Internet no es tan buena o está totalmente ausente o zonas con gran dificultad de implementación de tecnologías de comunicaciones habituales, por ejemplo, en zonas rurales, regiones montañosas, islas, entre otras.

La disponibilidad de conectividad a Internet continúa siendo menor en las zonas rurales y dispersas en comparación con las áreas urbanas. De acuerdo con el Departamento Administrativo Nacional de Estadística (DANE), en 2024 el 65,6 % de los hogares colombianos contaba con conexión a Internet, mientras que en los centros poblados y las zonas rurales dispersas este porcentaje fue de apenas 41,9 %, frente al 72,5 % registrado en las cabeceras municipales [1]. Esta diferencia evidencia la persistencia de una brecha de conectividad que puede limitar la implementación de sistemas que requieren comunicación permanente con plataformas de monitoreo remoto.

El Ministerio de Tecnologías de la Información y las Comunicaciones ha identificado la necesidad de mejorar la cobertura y calidad de los servicios de Internet, voz y datos en diferentes zonas rurales y urbanas del departamento. En 2024, durante la Mesa de Conectividad por Santander, el Ministerio señaló la necesidad de identificar puntos específicos con problemas de cobertura y calidad, incluyendo corregimientos y veredas, y planteó estrategias para ampliar la conectividad en 116 localidades mediante diferentes fases de intervención [2].

La estación meteorológica conectada por satélite podría cambiar la forma de recopilar datos en zonas remotas o sin una buena infraestructura. Se trata de medir de forma asequible todas las variables meteorológicas clave, como la temperatura, la humedad, la velocidad y dirección del viento, la probabilidad de precipitaciones y la presión atmosférica, entre otras.

Algunas características importantes de una estación meteorológica son, la precisión para mantener estándares buenos en las mediciones de los datos, la robustes para resistir condiciones climáticas, un diseño para que la instalación y mantenimiento sea sencillo, que sea energéticamente eficiente y autónoma para garantizar su uso en lugares con energía eléctrica insuficiente.

Se busca resolver la problemática con la conectividad a Internet por satélite, las estaciones pueden enviar de forma continua todos los datos recopilados en tiempo real. El internet satelital se caracteriza por tener cobertura global, además de tener fácil instalación y transporte ya que los complementos tienen peso ligero, las velocidades de transmisión son hasta por encima de 100MB/s, es resistente a condiciones climáticas adversas al contar con la clasificación IP56 [3].

Con la realización del Proyecto se busca comprobar la viabilidad de los servicios de internet satelital para su uso comercial, profesional y educativo, resaltando las velocidades de transmisión, la cobertura y la estabilidad del internet.

## 2. Justificación

En la actualidad aún en el territorio colombiano hay múltiples zonas sin conectividad y por lo tanto, dificulta los avances tecnológicos donde aquellos trabajadores de la agricultura aún se ven atados a técnicas de cuidados y mantenimientos tradicionales, provocando que los recursos y el tiempo sean ineficientes. Incorporando nuevas tecnologías de conectividad y comunicación a las zonas rurales impactará en la economía reduciendo gastos innecesarios en el proceso de la cosecha, además que permitir conectar estas áreas alejadas de las urbanizaciones impacta de manera directa a la sociedad brindando nuevas oportunidades a los habitantes de estos territorios.

Para mejorar la eficiencia en los recursos utilizados esta alternativa de conectividad mediante internet satelital para la trasmisión de datos en zonas rurales es la más adecuada en

términos de inversión, debido a que su plan de servicios y la adquisición de la antena es mucho más económico y eficaz que la construcción de una red cableada o antenas de comunicaciones móviles en la zona deseada.

Este tipo de proyectos para la institución genera nuevos enfoques de investigación en tecnologías emergentes que sirven para darle visibilidad como pionera en tecnologías de comunicación satelital y análisis de datos, que a su vez enriquece los recursos bibliográficos para futuros proyectos académicos investigativos.

De la misma forma amplía la base de conocimientos a la hora de implementar nuevas tecnologías y soluciones para las problemáticas y futuros proyectos en el campo de las redes de telecomunicaciones.

El profesional en ingeniería de telecomunicaciones tiene la capacidad de resolver problemáticas en los ámbitos tecnológicos y realización de proyectos adecuados a los posibles entornos agro en términos de tecnologías de comunicaciones. Desde el punto de vista tecnológico, este proyecto propone una solución que integra sensores meteorológicos, sistemas embebidos, un sistema de alimentación basado en energía fotovoltaica, una plataforma web y conectividad satelital mediante Starlink, permitiendo el monitoreo remoto de variables meteorológicas en zonas rurales con acceso limitado a redes de comunicación convencionales. En el ámbito académico, el desarrollo realizado constituye una base de referencia para futuras investigaciones y proyectos relacionados con el Internet de las Cosas (IoT), las comunicaciones satelitales y las aplicaciones de monitoreo orientadas al sector agrícola. En el aspecto social, la estación meteorológica facilita el acceso remoto a información climática local, brindando un apoyo para la toma de decisiones en actividades agropecuarias. Finalmente, desde la perspectiva económica, la solución tiene el potencial de favorecer una gestión más eficiente de los recursos empleados en la producción

agrícola; no obstante, la cuantificación de estos beneficios requiere estudios posteriores que permitan evaluar su impacto en los procesos productivos.

# 3. Objetivos

## 3.1 Objetivo general

Implementar una estación meteorológica que transmite datos vía internet satelital, empleando un sistema de gestión web y una Raspberry Pi como controlador principal de un sistema de adquisición de datos, ubicada en una zona rural cercano al municipio de Lebrija, Santander.

## 3.2 Objetivos específicos

- Caracterizar los parámetros meteorológicos (aire, presión, temperatura, humedad, entre otros) que favorecen los cultivos en la ubicación donde se implementará.
- Construir la red sensores meteorológicos controlados mediante el minicomputador Raspberry Pi, alimentando el sistema mediante energía solar y baterías recargables.
- Desarrollar un sitio web que permita graficar y gestionar la información localizada en la base de datos proveniente del sistema de adquisición de datos.
- Garantizar la conectividad entre el hardware y software del sistema a través de la implementación de una red satelital de Starlink, y comprobando el desempeño y rendimiento de la arquitectura de la estación meteorológica.

# 4. Marco referencial

El conocimiento y las predicciones sobre el clima han sido un objeto de intriga para las personas, entender los cambios atmosféricos es imprescindible para conseguir cultivos más eficaces en términos de maximizar la cosecha y reducir perdidas por imprevistos climáticos. Las estaciones meteorológicas son determinantes para recopilar datos atmosféricos como la temperatura, la intensidad solar, presión atmosférica, humedad, velocidad y dirección del viento y las precipitaciones. Estos datos son de gran importancia para la elaboración de predicciones climáticas. Sin embargo, existen zonas donde la disponibilidad de medios de transmisión no cumple con los requisitos necesarios para el envío de datos, ya que no cuentan con accesibilidad a internet, por lo tanto, existe redes satelitales capaces de proporcionar conectividad a internet desde cualquier parte del mundo. El internet satelital tiene como principal propósito tener cobertura desde diferentes lugares utilizando únicamente una antena que se encarga de hacer la conexión.

## 4.1 Estudios previos sobre estaciones meteorológicas

Hay pocas actividades humanas que sean completamente dependientes del clima como lo es la agricultura. Actualmente los servicios agrometeorológicos se han vuelto esenciales gracias al desafío de predecir eventos extremos y cambios climáticos por parte de sectores agrícolas. Este desafío tiene efectos socioeconómicos en un ámbito general, y especialmente en países en desarrollo [3].

Las estaciones meteorológicas han sido usadas por servicios meteorológicos y entes privados como un sistema para conocer detalles del clima y con el propósito de almacenar información meteorológica a lo largo del tiempo [4].

Se puede describir a una estación meteorológica como un dispositivo compuesto por una serie de herramientas destinas a la recolección y almacenamiento de variable meteorológicas [5]. En el trabajo previo que fue realizado directamente con la universidad Santo Tomas seccional Bucaramanga, este tipo de trabajo nos proporcionó una vista general sobre la realización de un proyecto con rubricas similares emitidas por la misma entidad educativa. El repositorio que lleva por nombre "Desarrollo de una estación meteorológica autónoma de bajo costo", describe una estación meteorológica desarrollada que se basa en una unidad de procesamiento de bajo coste, "Arduino", en un módulo SIM900 para la comunicación celular, red 3G o superior y en varios sensores para el monitoreo de temperatura, humedad, concentración de gases y otras variables del medio. El sistema incluye un modelo de alimentación fotovoltaica, el cual se ha diseñado para que el sistema tenga 2 días de autonomía sin radiación solar, permitiendo su utilización en áreas agrícolas y regiones aisladas. La información medida es almacenada en una base de datos y mostrada en la interfaz web gráfica, permitiendo el monitoreo y análisis remoto de las condiciones ambientales y climáticas.

Entre los principales resultados, se obtuvo la estación meteorológica funcionando de forma autónoma, capaz de medir, transmitir, guardar y visualizar adecuadamente las variables medidas. La posibilidad de conectividad celular permitió incrementar el rango de operación del sistema a zonas que cuentan con cobertura de un operador móvil, restando el problema de autonomía con la implementación del sistema fotovoltaico. La herramienta, permitió poder adecuar el estudio de microclimas y monitoreo ambiental en áreas agrícolas en Santander, entregando utilidad en el ámbito de la agricultura de precisión, optimización del uso de recursos y mejor evaluación [6].

Con respecto al artículo *monitoreo de una estación meteorológica remota a través de modems gprs*, una estación meteorológica remota diseñada para monitorear variables de

temperatura, velocidad y dirección del viento, presión atmosférica (barómetro), precipitación y humedad, con GPRS como mecanismo principal de comunicación a través de módulos inalámbricos completos con doble vía y de bajo coste y consumo. La estación usa comunicación serial RS-232 con el módulo de GPRS (General Packet Radio Service), y el Centro de Monitoreo utiliza USB para recibir la información. La información obtenida se almacena convenientemente en un servidor para su posterior procesamiento y visualización en tiempo real.

El sistema tiene como alcance la implementación de una red de estaciones meteorológicas remotas sobre un servidor dedicado al cual se puede acceder desde cualquier lugar a través de Internet. La red celular permite reubicar las estaciones de forma relativamente sencilla siempre y cuando haya cobertura del operador, posibilitando grandes áreas de operación. Los principales resultados son un sistema funcional para la adquisición, transmisión, almacenamiento y visualización remota de variables meteorológicas, mostrando la viabilidad de usar GPRS para el monitoreo de estaciones situadas en distintas zonas geográficas. Las principales limitaciones de la propuesta son la dependencia de la cobertura celular y el coste del servicio de comunicación móvil [7].

### 4.2 Observación sinóptica

La observación sinóptica hace referencia a la recolección de variables atmosféricas a nivel de la superficie en determinadas horas, tiene como fin contribuir en predicción meteorológica de las zonas y la climatología en un lugar específico, generalmente se hace la observación en intervalos de tiempo de tiempo de 3 y 6 horas [4].

### *4.2.1 Requisitos de precisión*

**Tabla 1.** *Requisitos de exactitud establecidos por la Organización Meteorológica Mundial para estaciones automáticas sinópticas.*

| Elemento | Requisito de exactitud |
|---|---|
| Presión atmosférica | ± 0.1 hPa sobre tierra ±0.1 hPa sobre el nivel del mar |
| Dirección del viento | ± 20° |
| Rapidez del viento | ± 2 m/s por debajo de 20 m/s ± 10% por arriba de 20 m/s |
| Temperatura del aire | ± 0.2 °C |
| Precipitación | ±0.5 mm por debajo de 5 mm ± 10% por arriba de 5 mm |

*Nota*. Presenta los requisitos de exactitud establecidos por la Organización Meteorológica Mundial [8] para estaciones automáticas sinópticas. Estos criterios sirven como referencia para seleccionar y evaluar los sensores implementados en el presente proyecto.

## 4.3 Predicciones climáticas

Las predicciones climáticas surgieron a partir de la necesidad de conocer los cambios meteorológicos para la toma de decisiones en diferentes campos de la sociedad. La predicción decenal funciona como medio entre la predicción climática a escala estacional-interanual y la proyección multidecenal de cambio climático.

Para predecir variables climáticas se emplean ecuaciones matemáticas complejas y algoritmos que simulan los movimientos de las masas de aire a partir de los datos proporcionados por las estaciones meteorológicas. Sin embargo, los modelos matemáticos no son precisos en su totalidad a causa de no poder controlar ciertas variables como el estado inicial de la atmosfera. Por lo tanto, los pronósticos con intervalos de tiempo muy extensos no son precisos [5].

## 4.4 Variables claves

### *4.4.1 Temperatura*

Cuando se habla de temperatura se refiere comúnmente a que tanto calor tiene un cuerpo. En términos físicos el calor es el grado de agitación de las partículas de los materiales, el calor está directamente relacionado con la energía cinética de los átomos, moléculas, partículas, en otros. En cuestión de climatología se busca la llamada temperatura seca del aire, la cual es la temperatura registrada por un termómetro que no es afectado por la radiación [9].

Generalmente la variable de interés es la temperatura seca del aire, definida como la temperatura medida por un termómetro protegido de la radiación solar directa y de otras fuentes de calor radiante termómetro protegido de la radiación solar directa y de otras fuentes de calor radiante, garantizando que la medición refleje únicamente las condiciones térmicas del aire circundante.

### *4.4.2 Intensidad solar*

La intensidad solar es la variable más importante, ya que funciona como detonante de otros efectos meteorológicos como la corriente de aire, la temperatura del ambiente y la presión entre muchos otros factores, por lo tanto, la recolección de información sobre la intensidad solar es imprescindible de una estación meteorológica [9]. Se expresa generalmente como irradiancia, definida como la potencia de energía solar incidente sobre una superficie por unidad de área, cuya unidad de medida en el Sistema Internacional es el vatio por metro cuadrado ($W/m^2$)

### *4.4.3 Presión atmosférica*

El monitoreo de la presión atmosférica y sus variaciones es un importante indicador a la hora de elaborar pronósticos meteorológicos, ya que una rápida disminución en la presión atmosférica indica mal tiempo porque para la formación de nubes se necesita que el aire húmedo ascienda desde las capas bajas de la atmósfera (lugares de bajas presiones). Mientras que una presión atmosférica alta está relacionada a un tiempo estable [10].

La presión atmosférica influye en diversas variables meteorológicas, como la dirección y velocidad del viento, la temperatura y la humedad del aire. Los registros continuos de presión permiten analizar la evolución de sistemas ciclónicos y anticiclónicos, identificar frentes meteorológicos y mejorar la precisión de los modelos de pronóstico numérico.

### *4.4.4 Humedad*

Cuando la temperatura sube, el aire adquiere la capacidad de contener más vapor de agua, lo cual quiere decir que, cuanto más cálido sea el clima, mayores pueden llegar a ser los niveles de humedad. El aire frío no es capaz de soporta tanta humedad como el aire caliente. Por lo tanto, se puede inferir que los climas fríos presentan niveles de humedad más bajos que en los climas cálidos [11].

### *4.4.5 Velocidad y dirección del viento*

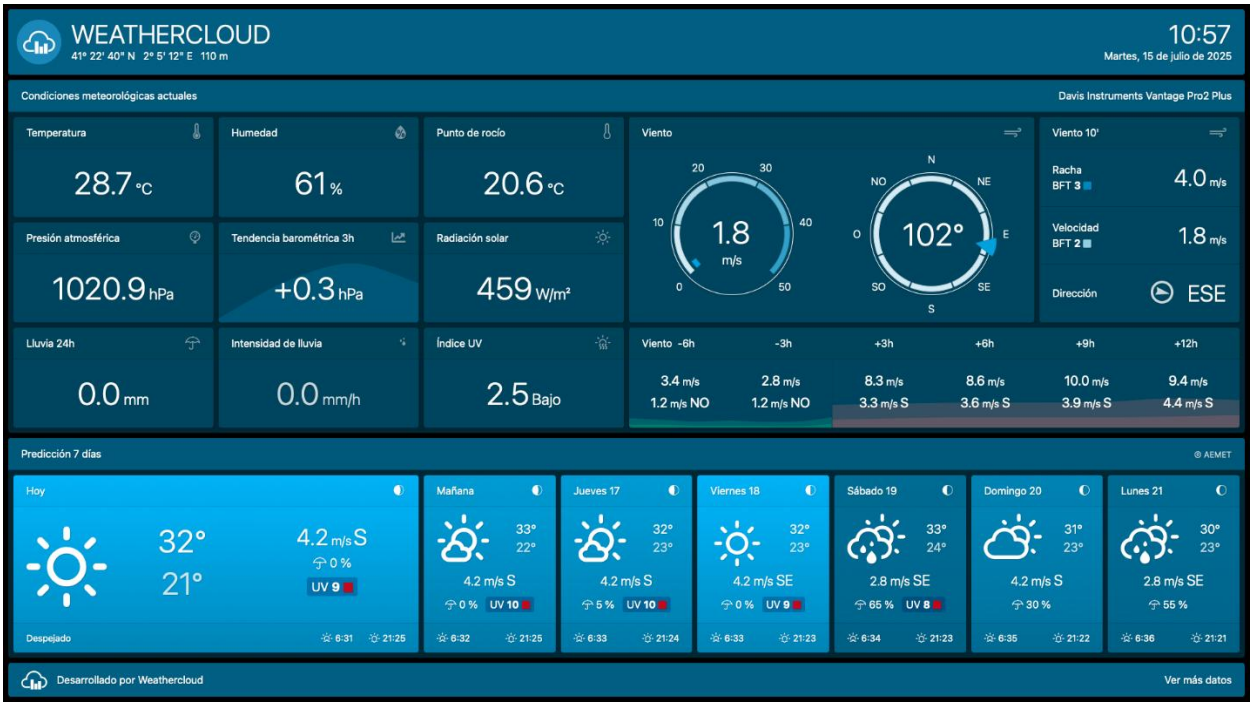
El viento es un movimiento natural generado por las diferencias de presión en la atmosfera donde el aire se mueve de zona de alta presión a zonas de baja presión. Se utiliza el anemómetro para medir la velocidad de las corrientes de aire en Km/h o en m/h, y la veleta para definir la dirección de donde proviene el viento [9].

### *4.4.6 Precipitación*

Las precipitaciones son las caídas de agua desde la atmósfera hacia la superficie. Las gotas de agua generan a partir de la condensación de vapor de agua concentrado en la atmosfera. Se cree que las gotas de agua pueden contener partículas de polvo o pequeños trozos de hielo lo cual ayudaría a la formación y caída de las gotas a la tierra [9].

Esta variable se expresa habitualmente en milímetros (mm), donde 1 mm de precipitación equivale a un litro de agua distribuido uniformemente sobre un metro cuadrado de superficie (1 $L/m^2$).

**Figura 1.** *Dashboard de una estación meteorológica profesional WeatherCloud.*




*Nota*. La figura se presenta como referencia para identificar los elementos que fueron considerados en el diseño del dashboard desarrollado en este proyecto, sin que represente la interfaz implementada. [12].

## 4.5 Teoría de la comunicación satelital

La comunicación satelital es un sistema que permite la transmisión de datos, voz y video a través de satélites que orbitan la Tierra. Este tipo de comunicación es fundamental para las áreas rurales o remotas donde no existen infraestructuras tradicionales de telecomunicaciones. Los satélites pueden operar en diferentes órbitas (baja, media o geoestacionaria) y son clave en la transmisión de televisión, telefonía, Internet y otros servicios globales [13].

### *4.5.1 Internet satelital*

Los satélites actúan como estaciones repetidoras que reciben señales de una estación en la Tierra y las retransmiten a otra, permitiendo la conexión a largas distancias. Su funcionamiento se basa en la transmisión de señales desde un terminal de usuario (generalmente una antena parabólica) a un satélite en órbita, que a su vez envía los datos a una estación terrestre conectada a la red mundial de Internet. El internet satelital tiene origen a partir de los años 60s. Siendo Telstar1 el primer satélite de comunicaciones, que se lanzó en 1962 por la NASA, que permitió por primera vez la transmisión de señales de televisión y datos a través del espacio. Posteriormente en la década de 1990 empresas como HughesNet y Viasat, empezaron a ofrecer servicios satelitales al público en general y en áreas remotas o sin acceso a la infraestructura de las telecomunicaciones.

Recientemente el proyecto Starlink de SpaceX ha impulsado una nueva era del Internet satelital, poniendo en funcionamiento una red de satélites de órbita baja (LEO) para reducir la latencia y mejoras de velocidad; buscan brindar acceso a internet a todas las regiones del mundo incluyendo las más aisladas [14].

### *4.5.2 Antena Starlink*

Los implementos que componen el kit de Starlink son el enrutador que utiliza tecnología wifi 5 IEEE 802.11 a/b/g/n/ac, de banda dual 3x3 MIMO brindando cobertura de hasta 185 m² y se conecta directamente a la energía. La antena se conecta al router mediante un cable "Starlink" y se orienta automáticamente buscando la mejor posición para recibir cobertura satelital el kit tiene un consumo promedio desde 50 a 75 W. La tecnología de internet satelital Starlink, tiene la capacidad de proporcionar conectividad en zonas donde la infraestructura de telecomunicaciones convencional es limitada o inexistente y además es de fácil transporte y puesta en servicio.

**Figura 2.** *Kit antena Starlink*

*Nota*. Elementos del kit de montaje de Starlink, router, cables y antena [15].

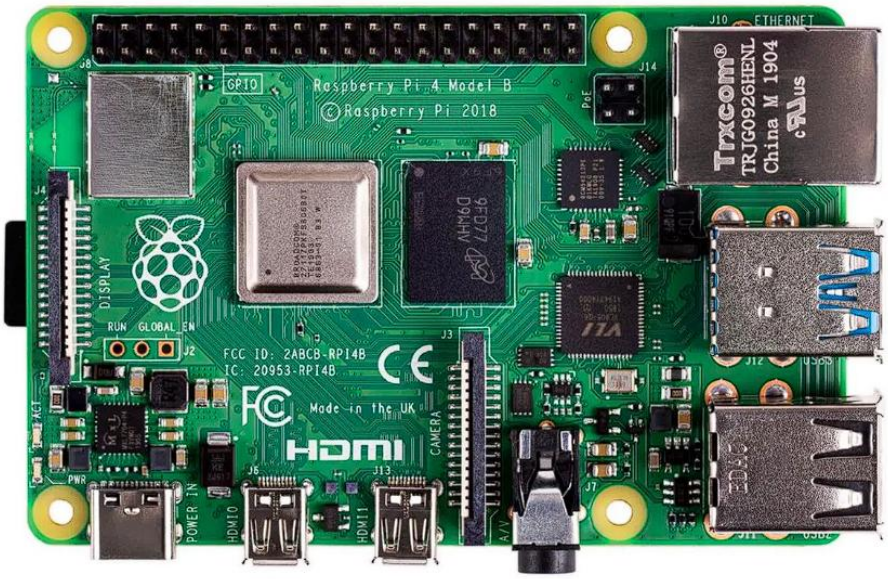
### *4.6 Microcomputador Raspberry Pi 4*

La Raspberry Pi 4 es un computador compacto de placa única, lo que significa que todos los componentes usuales de un computador como el procesador, la memoria RAM, el almacenamiento y los diferentes puertos de comunicación y de conexión están ubicados en la

misma placa base. Usualmente la Raspberry pi 4 utiliza el sistema operativo “Raspberry pi OS” el cual está basado en Linux, este sistema operativo también se le conoce como Raspbian.

Los usos más comunes para la Raspberry Pi son proyectos educativos, aunque también es muy útil para la implementación de domótica, IOT gracias a que cuenta con puerto de red de área local y conectividad Wifi que favorece la implementación del internet satelital, además puede funcionar como nodo principal de procesamiento y comunicaciones, lo que ejecuta los servicios necesarios para recibir los datos provenientes del Arduino, procesarlos y transmitirlos para su visualización.

**Figura 3.** *Microcomputador Raspberry Pi 4*




*Nota.* Imagen correspondiente al microcomputador Raspberry Pi 4 y sus módulos [16].

### *4.7 Estación meteorológica SEN-15901*

El kit meteorológico SEN-15901 está compuesto por una veleta, un anemómetro de cazoletas y un pluviómetro de balancín, junto con sus respectivos elementos de montaje. Los

sensores funcionan mediante imanes y contactos magnéticos tipo Reed Switch, sin electrónica activa, generando pulsos eléctricos que pueden ser registrados mediante entradas digitales, contadores o interrupciones de un microcontrolador. El anemómetro produce un pulso por segundo a una velocidad de 2,4 km/h (0,67 m/s), permitiendo calcular la velocidad mediante la relación (V=2,4f), donde (f) corresponde a la frecuencia de pulsos en Hz. El anemómetro y la veleta comparten un cable con conector RJ11, utilizando los pines 2 y 3. El pluviómetro de balancín auto-vaciable registra la precipitación mediante cierres momentáneos de su contacto magnético. Cada pulso corresponde a 0,2794 mm de precipitación, por lo que la precipitación acumulada se obtiene mediante (P=N\0,2794), donde (N) es el número de pulsos registrados. El sistema puede integrarse con plataformas como Raspberry Pi o microcontroladores, permitiendo adquirir, procesar y almacenar los datos de velocidad y dirección del viento y precipitación para su posterior monitoreo y análisis meteorológico. [17].

**Figura 4.** *Kit estación meteorológica SEN-15901*

*Nota.* Imagen de los sensores de dirección, velocidad del viento y precipitación, [18].

# 5. Metodología

El desarrollo del proyecto constará de 4 fases. La primera fase es el planteamiento de la estructura de la estación y las variables meteorológicas que se tomaran para así identificar que sensores y que elementos adquirir. La segunda fase comprende el montaje físico del sistema, así como el funcionamiento de los sensores, la programación general del proyecto y la alimentación energética. La tercera fase hace referencia al montaje del dispositivo en la ubicación seleccionada donde se llevará a cabo la toma de datos. Por último, en la cuarta fase se tomarán los resultados y se analizarán para comprobar la efectividad del proyecto.

## 5.1 Fase 1 etapa de selección

Para crear una estación meteorológica primero se debe crear la arquitectura de los complementos que esta debe tener. Como primera medida se identificará el controlador principal de la arquitectura de la estación meteorológica, en el caso de este proyecto se usará una "Raspberry pi 4" debido a que incorpora conectividad Wi-Fi y Bluetooth, además de múltiples interfaces de comunicación (GPIO, $I^2C$, SPI y UART), lo que facilita la integración de los sensores y la transmisión de la información sin requerir módulos adicionales Para la adquisición de las variables meteorológicas se empleará la Weather Shield, ya que dispone de dos conectores RJ45 diseñados para la conexión del kit meteorológico, permitiendo la lectura de las señales provenientes del pluviómetro, el anemómetro y la veleta. Además, integra los sensores de presión barométrica, humedad ambiental, temperatura y radiación ultravioleta, simplificando la integración del sistema y garantizando la compatibilidad entre los diferentes dispositivos. Como medio de alimentación se empleará un panel solar y una batería que suministre energía a todo el sistema en cualquier hora del día. Finalmente, para la transmisión de datos las redes móviles fueron descartadas debido a las

limitaciones de cobertura presentes en la zona, mientras que los radioenlaces y enlaces punto a punto requieren infraestructura adicional y condiciones adecuadas de línea de vista. Por su parte, aunque LoRaWAN ofrece bajo consumo y largo alcance, requiere gateways y una conexión adicional para acceder a Internet. Frente a estas alternativas, se seleccionó Starlink debido a su capacidad para proporcionar conectividad a Internet en zonas rurales con infraestructura terrestre limitada, además de ofrecer facilidad de despliegue, velocidades de transmisión y latencias adecuadas para las necesidades del sistema.

### 5.2 Fase 2 prueba de comunicaciones

Para validar el funcionamiento del sistema, se realizaron pruebas de comunicación entre la Raspberry Pi 4, la Weather Shield y los sensores meteorológicos, verificando la adquisición y transmisión continua de los datos hacia el dashboard. Durante las pruebas se evaluaron las variables de temperatura, humedad relativa, velocidad del viento, precipitación y radiación solar. Una vez confirmado el funcionamiento optimo de cada sensor, se escribió la programación de la base de datos que recopila la información y de la página web donde se visualizan, diseñado para su fácil interpretación para los usuarios.

Posteriormente se agregó la antena de internet satelital “Starlink”, el panel fotovoltaico que alimenta el dispositivo, y su respectiva batería para su funcionamiento en las condiciones climáticas adversas y las horas nocturnas.

Finalmente se comprobó la viabilidad del internet satelital en esta zona recompilando datos de las velocidades de transferencia de datos y la cobertura, analizando su rendimiento en un intervalo de tiempo y sobre todo en los momentos donde las condiciones climáticas no son favorables.

### 5.3 Fase 3 prueba de rendimiento

Se escogió una finca en la vereda de Llanada, está finca no cuenta con conexión a internet estable y tiene señal de telefonía móvil limitada, estas características la hacen perfecta para para probar el rendimiento y la efectividad del dispositivo. Se buscó un lugar óptimo para poner la estación meteorológica y para establecer la orientación de la antena de “Starlink” para que tenga el mejor rendimiento posible.

Para evaluar el desempeño del sistema se realizaron pruebas de conectividad utilizando la aplicación oficial de Starlink y la herramienta Speedtest. Como indicadores de evaluación se consideraron la velocidad de descarga, la velocidad de subida y la latencia de la conexión. Las pruebas realizadas con la aplicación de Starlink registraron velocidades de descarga entre 156 y 232 Mbps, velocidades de subida entre 14 y 38 Mbps, una latencia de 21 ms y una latencia mediana del router de 36 ms y un consumo energético de 34W. De manera complementaria, mediante Speedtest se obtuvieron velocidades de descarga entre 160 y 200 Mbps, confirmando la estabilidad y el desempeño del enlace satelital durante las pruebas.

Adicionalmente, se verificó la transmisión continua de los datos meteorológicos desde la estación hacia el dashboard, comprobando que no se presentaran pérdidas de comunicación durante el periodo de evaluación y que la información recibida fuera consistente con los registros generados por los sensores.

### 5.4 Fase 4 comparación de rendimiento

En este punto se tomaron los resultados de la página web y se hizo una comparativa con de las mediciones obtenidas y los valores reportados por las plataformas meteorológicas OpenWeather y AccuWeather, así como con las mediciones registradas por otros sensores

disponibles en el lugar de implementación. Como criterio de aceptación, se consideró satisfactorio que el sistema transmitiera los datos sin pérdidas de comunicación y que las mediciones presentaran un comportamiento coherente con las fuentes de referencia. para confirmar la veracidad de los datos y el correcto funcionamiento del sistema.

Se recogieron las estadísticas recopiladas del internet satelital y se mostró en un gráfico la eficiencia con respecto al internet por cable del lugar. Basado en estos resultados se determinó que la tecnología “Starlink” es útil para las condiciones en que fue expuesta.

# 6. Desarrollo del proyecto

## 6.1 Caracterización de los parámetros meteorológicos de una zona rural del municipio de Lebrija, Santander

### *6.1.1 Descripción del área de implementación*

La estación meteorológica se implementó en una zona rural próxima al municipio de Lebrija, Santander, Colombia, municipio que se localiza en la subregión del occidente santandereano, con una altitud media de 590 metros sobre el nivel del mar (msnm), lo que genera una diversidad climática que abarca pisos térmicos cálidos y templados dependiendo de la cota altitudinal, así mismo, la zona específica de instalación corresponde a un ambiente de clima templado con temperaturas medias que se sitúan entre los 18 °C y 24 °C, condición característica de las laderas medias del departamento [19].

Lebrija y sus municipios aledaños constituyen una zona de vocación agrícola activa en la que se desarrollan cultivos de aguacate (variedades Choqué y Lorena), cacao, mandarina, naranja,

café y plátano, entre otros, así, esta diversidad productiva justifica la necesidad de monitorear de manera continua y precisa las variables atmosféricas que inciden en la calidad y el rendimiento de las cosechas, especialmente ante la variabilidad climática propia de la región andina colombiana [20].

**Figura 5.** *Imagen satelital del lugar de la implementación de la estación meteorológica.*


*Nota.* El proyecto se implementó en una finca ubicada en la vereda Llanadas, municipio de Lebrija, departamento de Santander, Colombia, cuyas coordenadas geográficas aproximadas son 7°08'54.6"N, 73°10'49.0"W. La Figura 5 muestra la ubicación del área de estudio, la cual se incluye con el propósito de contextualizar el sitio de implementación de la estación meteorológica y las condiciones geográficas donde se realizaron las pruebas de funcionamiento y adquisición de datos. [21].

### *6.1.2 Parámetros meteorológicos caracterizados*

La caracterización meteorológica de la zona de implementación abarca las variables atmosféricas con mayor influencia sobre los sistemas de producción agrícola de la región, a

continuación, se describen cada uno de los parámetros seleccionados para ser monitoreados por la arquitectura de la estación meteorológica.

*a) temperatura del aire:* La temperatura es la variable meteorológica de mayor impacto sobre los procesos fisiológicos de los cultivos, dado que regula las tasas de fotosíntesis, respiración, floración y cuajado de frutos. Según la Organización Meteorológica Mundial [8], la temperatura del aire debe medirse con una precisión de ±0,2 °C para garantizar la validez de las observaciones sinópticas. Para los cultivos de la zona, el aguacate (variedades Choqué y Lorena) requiere temperaturas de entre 18 °C y 24 °C, con alta sensibilidad a temperaturas superiores a 30 °C que comprometen la polinización. El cacao prospera en rangos de 20 °C a 30 °C, la mandarina y la naranja entre 15 °C y 30 °C, y el café entre 17 °C y 23 °C [22].

La estación registra la temperatura del aire mediante sensores de alta precisión con lecturas periódicas que permitirán identificar variaciones diurnas y nocturnas relevantes para la toma de decisiones agronómicas [8].

*b) humedad relativa:* La humedad relativa expresa el porcentaje de vapor de agua contenido en el aire respecto al máximo que puede contener a una temperatura dada, parámetro que incide directamente en la tasa de evapotranspiración de los cultivos, el desarrollo de enfermedades fúngicas y la eficiencia del riego [19], además, para el aguacate, valores superiores al 80% durante períodos prolongados favorecen la aparición de *Phytophthora cinnamomi* (pudrición de raíz), especialmente perjudicial en suelos franco-arcillosos como los del área de implementación y en el cacao requiere humedades relativas entre 70 % y 90 % para una polinización eficiente [22].

*c) presión atmosférica:* La presión atmosférica, medida en hectopascales (hPa), es un indicador clave para el pronóstico de condiciones climáticas a corto plazo. La OMM [8] establece que la presión atmosférica debe medirse con una exactitud de ±0,1 hPa para su uso en observación

sinóptica y en altitudes entre 500 y 1.500 msnm, los valores típicos oscilan entre 850 hPa y 960 hPa aproximadamente, así, las caídas sostenidas de presión anticipan episodios de lluvia o tormenta, mientras que los incrementos señalan períodos de estabilidad atmosférica, información de alto valor para la planificación de las labores agrícolas [19].

*d) precipitación:* La precipitación es uno de los parámetros más determinantes para la planificación agrícola en la región y Lebrija presenta un régimen de lluvias bimodal, con períodos húmedos en los trimestres marzo-mayo y septiembre-noviembre y períodos secos en diciembre-febrero y junio-agosto, además, la precipitación anual media varía entre 1.500 mm y 2.500 mm en la zona de interés [19].

Para el cultivo del aguacate, un déficit hídrico durante la floración puede reducir significativamente el cuajado de frutos y el cacao requiere precipitaciones bien distribuidas con un mínimo de 1.500 mm anuales [22]. La OMM [8] establece que la precipitación debe medirse con una exactitud de ±0,5 mm para valores inferiores a 5 mm, y ±10 % para valores superiores, es decir, el pluviómetro instalado suministrará información en tiempo real para la gestión del riego y la prevención de encharcamientos en el suelo franco-arcilloso de la finca.

*e) velocidad y dirección del viento:* El viento actúa como vector de dispersión de plagas, enfermedades y pólenes, e influye en la tasa de evapotranspiración de los cultivos, ante esto, según la OMM [8], la rapidez del viento debe medirse con una exactitud de ±2 m/s para velocidades inferiores a 20 m/s, y ±10 % por encima de ese umbral y en la zona montañosa del occidente santandereano, las velocidades medias raramente superan los 5 m/s en zonas de ladera [19].

La dirección del viento es relevante para anticipar la dispersión de agroquímicos durante las fumigaciones y para identificar la procedencia de masas de aire húmedo que favorecen la precipitación, así, la OMM [8] establece una exactitud de ±20° para la dirección del viento en

estaciones automáticas sinópticas, por ende, la estación registrará ambas variables mediante anemómetro y veleta integrados en el sistema de sensores.

*f) intensidad de radiación solar:* La radiación solar es el motor energético de la fotosíntesis y determina en gran medida la productividad potencial de los cultivos, de esta manera, en la zona de Lebrija, la radiación solar global varía entre 12 y 18 MJ/m² por día, con fluctuaciones importantes debidas a la nubosidad estacional [19], así mismo, el aguacate es una especie heliófila que requiere plena exposición solar, mientras que el cacao en etapas juveniles requiere sombra parcial [22], por tanto, la medición de esta variable permitirá calcular índices de estrés luminoso y correlacionar los datos con el rendimiento productivo de los cultivos monitoreados [20].

**Tabla 2.** *Resumen de parámetros meteorológicos caracterizados, rangos de referencia y relevancia agroclimática en la zona de implementación (Lebrija, Santander).*

| **Parámetro** | **Unidad** | **Rango de referencia en la zona** | **Relevancia agroclimática** |
|---|---|---|---|
| Temperatura del aire | °C | 18 – 24 °C (clima templado) | Floración, cuajado de frutos, fotosíntesis |
| Humedad relativa | % | 65 – 90 % | Evapotranspiración, control de enfermedades fúngicas |
| Presión atmosférica | hPa | 850 – 960 hPa (500–1700 msnm) | Pronóstico de eventos climáticos a corto plazo |
| Precipitación | mm | 1.500 – 2.500 mm/año (bimodal) | Gestión hídrica del suelo, programación del riego |

| Parámetro | Unidad | Rango de referencia en la zona | Relevancia agroclimática |
|---|---|---|---|
| Velocidad del viento | m/s | 0 – 5 m/s (media en ladera) | Dispersión de plagas, evapotranspiración |
| Dirección del viento | Grados (°) | 0° – 360° | Dispersión de agroquímicos, masas de aire húmedo |
| Radiación solar | MJ/m²/día | 12 – 18 MJ/m²/día | Fotosíntesis, estrés luminoso en cultivos |

Tomado con base en OMM [8], IDEAM [19], Balaguera-López & Martínez [22] y MADR [20].

### *6.1.3 Relación entre los parámetros meteorológicos y los cultivos de la zona*

La interacción entre los parámetros meteorológicos caracterizados determina de manera conjunta las condiciones de aptitud agroclimática de la zona de Lebrija. La temperatura y la humedad relativa actúan de manera sinérgica sobre los ciclos fenológicos de las plantas; la precipitación define la disponibilidad hídrica del suelo; la radiación solar condiciona la productividad fotosintética; y el viento modula la tasa de transpiración y la propagación de agentes patógenos [22].

El suelo franco-arcilloso de la zona de implementación presenta características de retención de humedad que amplifican la sensibilidad de los cultivos a las variaciones de precipitación, haciendo especialmente relevante el monitoreo continuo de esta variable para prevenir tanto el estrés hídrico por déficit como el encharcamiento que compromete el sistema radicular del aguacate [20], en este contexto, la caracterización realizada constituye la base técnica para la selección, calibración y configuración de los sensores de la estación meteorológica, y define los

umbrales de alerta que serán programados en el sistema de gestión web para la toma de decisiones agronómicas oportunas [19].

### *6.1.4 Instrumentos de medición seleccionados para la caracterización*

Con base en los parámetros identificados y las condiciones del entorno rural, se seleccionaron los siguientes sensores para integrar la red de medición de la estación, así, los criterios de selección consideraron la compatibilidad con la Raspberry Pi, el bajo consumo energético, la precisión mínima establecida por la OMM [8], y la disponibilidad de los componentes en el mercado colombiano [20]. La seleccion de los sensores respondió a una evaluación multicriterio que considero simultáneamente: (1) precisión mínima exigida por la OMM (2018); (2) rango de medición adecuado a las condiciones climáticas de Lebrija (500-1.700 msnm, 18-24 grados C); (3) consumo energético compatible con el sistema de alimentación solar; (4) costo de adquisición en el mercado colombiano; (5) disponibilidad comercial local; (6) compatibilidad eléctrica y de protocolo con el Arduino Uno R3 y la Raspberry Pi 4; y (7) facilidad de integración con las librerías del entorno Arduino [8, 23]:

**Tabla 3.** *Sensores seleccionados para la red de medición meteorológica de la estación.*

| Sensor / instrumento | Parámetro medido | Criterios de selección y justificación técnica |
|---|---|---|
| HTU21D | Temperatura y humedad relativa | Precisión: resolución de 0,01 grados C y 0,04 % HR, superando el requisito de la OMM (2018) de +/-0,2 grados C y +/-2 % HR para estaciones automáticas sinóticas. |
| | | Consumo energético: inferior a 0,5 mA en operación activa; compatible con el sistema solar de capacidad limitada. |

| Sensor / instrumento | Parámetro medido | Criterios de selección y justificación técnica |
|---|---|---|
| | | Compatibilidad: integrado nativamente en el SparkFun Weather Shield DEV-13956 sobre el bus I2C compartido (pines SDA/SCL del Arduino Uno R3), sin cableado adicional. |
| | | Facilidad de integración: librería oficial *SparkFunHTU21D.h* disponible en Arduino Library Manager; inicialización en una línea `myHumidity.begin()` dentro de setup(). |
| | | Costo y disponibilidad: incluido en el escudo adquirido sin costo adicional; disponible en el mercado colombiano de electrónica. |
| MPL3115A2 | Presión barométrica | Precisión y rango: rango de 20 a 110 kPa (200 a 1.100 hPa), cubre el rango altitudinal 500-1.700 msnm (850-960 hPa). Resolucion de 0,2 Pa; cumple el requisito OMM (2018) de +/-0,1 hPa. |
| | | Consumo energético: inferior a 40 uA en modo activo; el oversampling configurable (`setOversampleRate(7)`) permite equilibrar precisión y consumo. |
| | | Compatibilidad: bus I2C compartido con el HTU21D en el Weather Shield; sin pines adicionales del Arduino. |
| | | Facilidad de integración: librería *SparkFunMPL3115A2.h*; modo barométrico activado con `myPressure.setModeBarometer()` en setup(). |
| | | Costo y disponibilidad: incluido en el Weather Shield; sin adquisición adicional. |
| Pluviómetro de cubeta basculante | Precipitación acumulada y tasa horaria | Precisión: resolución de 0,2794 mm por basculeo, dentro del margen OMM (2018) de +/-0,5 mm para valores inferiores a 5 mm. |
| | | Consumo energético: nulo; sensor puramente mecánico con reed switch sin consumo activo en ningún estado. |
| | | Compatibilidad: salida de pulso digital en Pin 2 (INT0) del Arduino; ISR `rainIRQ()` con debounce de 10 ms que elimina rebotes del reed switch. |
| | | Facilidad de integración: lógica de conteo con variable `volatile rain_total_tips`; formula de tasa horaria: `rain_tips x 0,2794 x (3600000 / sample_ms)`. Sin librerías externas. |

| Sensor / instrumento | Parámetro medido | Criterios de selección y justificación técnica |
|---|---|---|
| | | Costo y disponibilidad: incluido en el SparkFun Weather Meter Kit SEN-15901 junto con el anemómetro y la veleta. |
| Anemómetro de cazoletas | Velocidad del viento | Rango de medición: 0 a 50 m/s; excede ampliamente las velocidades medias de la zona (menos de 5 m/s en ladera), garantizando que ningún evento extremo sature el sensor. |
| | | Precisión: factor certificado por el fabricante: 1 pulso/s = 2,4 km/h. Formula implementada: `wind_clicks x 2,4 x (1000 / sample_ms)`, donde sample_ms registra el periodo real de muestreo. |
| | | Consumo energético: nulo; sensor mecánico con reed switch. |
| | | Compatibilidad: salida de pulso en Pin 3 (INT1); ISR `wspeedIRQ()` con debounce de 10 ms; variable `volatile wind_clicks`. |
| | | Facilidad de integración: sin librerías externas; incluido en el kit SEN-15901. |
| Veleta (divisor resistivo) | Dirección del viento | Resolución: 16 posiciones (22,5 grados por posición), dentro del margen OMM (2018) de +/-20 grados para estaciones sinoticas. |
| | | Rango: 0 a 360 grados (N=0, E=90, S=180, O=270), en concordancia con la convención meteorológica estándar. |
| | | Consumo energético: divisor resistivo pasivo; sin consumo activo. |
| | | Compatibilidad: salida analógica en pin A0 del Arduino (ADC 10 bits, rango 0-1.023). Funcion `get_wind_direction()` convierte ADC a grados por tabla de 16 entradas (ej. ADC 985 = 315 grados NW; ADC 746 = 270 grados W). |
| | | Facilidad de integración: sin librerías externas; incluido en el kit SEN-15901. |
| Fotorresistor (pin A1 del Weather Shield) | Nivel de luminosidad solar | Costo y disponibilidad: integrado en el SparkFun Weather Shield sin costo adicional ni adquisición extra. |
| | | Consumo energético: pasivo; sin consumo propio. |

| Sensor / instrumento | Parámetro medido | Criterios de selección y justificación técnica |
|---|---|---|
| | | Compatibilidad: lectura analógica en pin A1. La función `get_light_level()` aplica calibración dinámica con la tensión de referencia 3,3 V del pin A3 (`refV = analogRead(A3) x (3,3/1023)`), compensando variaciones de la alimentación solar. |
| | | Facilidad de integración: sin librerías externas; implementado directamente en el sketch. |
| | | Limitación reconocida: al ser un fotorresistor y no un piranómetro calibrado, la variable entregada es una aproximación relativa en voltios (0-3,5 V), no irradiancia en W/m2. Suficiente para monitoreo comparativo, pero no para cálculos de balance energético de precisión agronómica. |
| Divisor resistivo (pin A2 del Weather Shield) | Voltaje de la batería | Relevancia operacional: permite registrar el estado de carga del banco de energía en cada ciclo de muestreo (campo `battery_v` en el JSON), información critica para la operación autónoma en zona rural. |
| | | Compatibilidad: lectura analógica en pin A2. Función `get_battery_level()` convierte el valor ADC al voltaje real aplicando el factor de escala `BATTERY_SCALE`, referenciado a la tensión de A3. |
| | | Costo y disponibilidad: integrado en el Weather Shield sin costo adicional. |
| | | Facilidad de integración: sin librerías externas; comparte la referencia de voltaje de A3 con el fotorresistor. |

Tomado con base en OMM [8, 23].

La combinación del SparkFun Weather Shield DEV-13956 con el Weather Meter Kit SEN-15901 resulto determinante desde el punto de vista de la integración: ambos componentes están diseñados para trabajar juntos con el Arduino Uno R3, comparten la misma hoja de ruta de librerías oficiales y permiten la adquisición de las siete variables meteorológicas requeridas sin desarrollo de drivers personalizados, entonces, los sensores I2C (HTU21D y MPL3115A2) comparten el bus de datos sin conflicto gracias a sus direcciones de dispositivo distintas, y los sensores externos

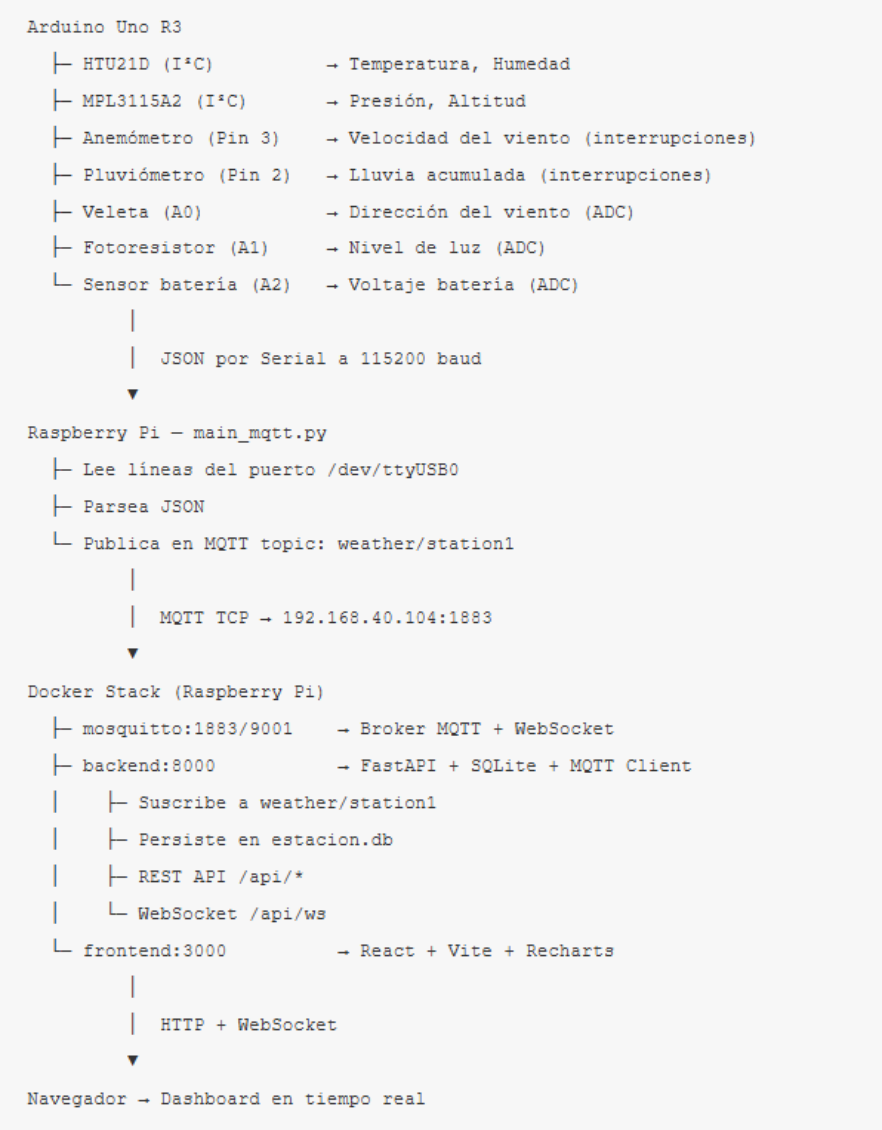
utilizan los pines GPIO y ADC del Arduino de forma independiente, con las interrupciones hardware INT0 e INT1 reservadas para los dos sensores de conteo de pulsos [8].

## 6.2 Construcción de la red sensores meteorológicos controlados mediante el minicomputador Raspberry Pi, alimentando el sistema mediante energía solar y baterías recargables.

### *6.2.1 Arquitectura general de los sensores*

La arquitectura de los sensores meteorológicos del sistema está organizada en torno a dos unidades de procesamiento con funciones diferenciadas y complementarias: un Arduino Uno R3 como nodo de adquisición y una Raspberry Pi 4 como computador de placa única encargado del procesamiento, comunicaciones y servicios del sistema, ejecuta las interrupciones de hardware y serializa las lecturas en formato JSON a una tasa de 115.200 baudios. La Raspberry Pi 4 actúa como controlador principal del sistema toda vez que recibe los datos del Arduino por puerto USB-serial, los publica en el broker MQTT local y orquesta la pila de servicios web mediante contenedores Docker [24, 25].

**Figura 6.** *Diagrama de flujo general del sistema.*




*Nota.* La Figura 6 presenta la arquitectura de comunicación de la estación meteorológica y el flujo de información desde la adquisición de los datos hasta su visualización.

Inicialmente, el Arduino Uno R3 adquiere las mediciones provenientes de los sensores meteorológicos. Los sensores HTU21D y MPL3115A2 se comunican mediante el protocolo I²C, mientras que el anemómetro y el pluviómetro utilizan entradas digitales basadas en interrupciones, y la veleta, el fotorresistor y el sensor de voltaje de la batería emplean entradas analógicas (ADC).

Una vez recopiladas las mediciones, el Arduino las organiza en formato JSON y las transmite a la Raspberry Pi a través de una comunicación serial USB a 115200 baudios.

En la Raspberry Pi, la aplicación main_mqtt.py recibe y procesa los datos, para posteriormente publicarlos mediante el protocolo MQTT en el tópico weather/station1 utilizando una conexión TCP/IP. El broker Mosquitto, ejecutado en un contenedor Docker, distribuye la información al backend desarrollado con FastAPI, el cual almacena los registros en una base de datos SQLite y expone los datos mediante una API REST y un servicio WebSocket. Finalmente, el frontend desarrollado en React consume esta información utilizando los protocolos HTTP y WebSocket, permitiendo la visualización de las variables meteorológicas en tiempo real desde cualquier navegador web. Es importante destacar que las comunicaciones entre el Arduino, la Raspberry Pi y los servicios ejecutados localmente corresponden a la red interna del sistema, mientras que el acceso remoto al dashboard se realiza a través de la conexión a Internet proporcionada por el sistema satelital Starlink, el cual permite que los datos puedan consultarse desde cualquier ubicación con acceso a la red.

Esta arquitectura de dos capas permite separar la capa de adquisición de datos en tiempo real donde los tiempos de respuesta son críticos, de la capa de procesamiento, persistencia y transmisión, donde los requerimientos son distintos, así, el Arduino garantiza la lectura continua e ininterrumpida de los sensores mediante interrupciones hardware, mientras que la Raspberry Pi gestiona la conectividad de red y la lógica de la aplicación [26, 27].

### *6.2.2 Microcontrolador Arduino uno R3 y escudo meteorológico*

El nodo de adquisición de datos está construido sobre un Arduino Uno R3 equipado con el SparkFun Weather Shield (DEV-13956), considerado como un escudo que integra en una sola

placa los dos sensores digitales de mayor precisión del sistema y proporciona los conectores y la lógica de acondicionamiento de señal para los sensores externos del kit meteorológico SparkFun Weather Meter Kit (SEN-15901).

La selección de este conjunto de hardware responde a su diseño específico para aplicaciones de monitoreo ambiental, su compatibilidad nativa con las librerías Arduino y su bajo consumo energético, que resulta determinante para la operación autónoma del sistema [23], además, el sketch principal (*sensors.ino*, 216 líneas) se ejecuta en el Arduino y realiza la lectura de todos los sensores aproximadamente cada segundo y en la función *setup()* se inicializan los sensores I²C, se configuran los pines de los sensores externos y se registran las rutinas de servicio de interrupción (ISR) para el anemómetro (Pin 3 / INT1) y el pluviómetro (Pin 2 / INT0). En el bucle principal (*loop()*) se invoca periódicamente la función *printWeatherJSON()*, que serializa todas las mediciones como un objeto JSON completo y lo envía por el puerto serial al puente Python de la Raspberry Pi [25].

**Tabla 4.** *Componentes de hardware del sistema de sensores meteorológicos*

| Componente | Modelo / Referencia | Función en el sistema |
|---|---|---|
| Microcontrolador | Arduino Uno R3 | Adquisición de datos, ejecución de ISR, salida JSON por serial a 115.200 baudios |
| Escudo meteorológico | SparkFun Weather Shield DEV-13956 | Integra sensores I²C (HTU21D, MPL3115A2) y conectores para sensores externos |
| Kit de sensores externos | SparkFun Weather Meter Kit SEN-15901 | Anemómetro de cazoletas, veleta de divisor resistivo, pluviómetro de cubeta basculante |

| Componente | Modelo / Referencia | Función en el sistema |
|---|---|---|
| Controlador principal | Raspberry Pi 4 Model B | Puente serial-MQTT, orquestación de servicios Docker, transmisión vía Starlink |

Tomado con base en SparkFun Electronics [23], Arduino AG [25] y Raspberry Pi Ltd. [24].

### *6.2.3 Sensores del sistema y variables medidas*

El sistema integra dos tipos de sensores según su principio de operación: sensores digitales de bus I²C montados en el Weather Shield y sensores analógicos y de pulsos conectados directamente a los pines GPIO del Arduino [23].

*a) Sensor HTU21D: temperatura del aire y humedad relativa.* El HTU21D es un sensor digital de alta precisión que se comunica por el bus I²C compartido con el MPL3115A2, proporciona lecturas de temperatura en grados Celsius con una resolución de 0,01°C y de humedad relativa con una resolución de 0,04%, valores que superan los requisitos de exactitud establecidos por la OMM [8] para estaciones automáticas sinópticas (±0,2°C y ±2% HR) y su inicialización se realiza mediante la librería *SparkFunHTU21D.h* en la función *setup()* del sketch [23].

*b) Sensor MPL3115A2:* es un sensor de presión absoluta configurado en modo barométrico mediante el método *setModeBarometer()*, el cual registra la presión en Pascales (Pa) en un rango de 20 a 110kPa, que el backend convierte a hectopascales (hPa) dividiendo entre 100. La OMM [8] establece una exactitud de ±0,1hPa para presión en estaciones sinópticas, requisito que el sensor cumple con su resolución de 0,2Pa [23].

*c) Anemómetro de cazoletas: velocidad del viento:* El anemómetro del kit SEN-15901 opera mediante un reed switch que genera un pulso eléctrico por cada rotación completa del mecanismo de cazoletas, donde el Arduino cuenta estos pulsos mediante una ISR conectada al Pin 3 (INT1),

con un filtro de rebote (debounce) de 10 ms para eliminar falsas interrupciones. La velocidad del viento se calcula con la fórmula:

*wind_kph = wind_clicks × 2,4 × (1.000 / sample_ms)*

donde el factor 2,4 corresponde al factor de conversión del fabricante: un pulso por segundo equivale a 2,4 km/h [8, 23].

*d) Pluviómetro de cubeta basculante:*El pluviómetro opera mediante el principio de cubeta basculante: cada vez que el cucharón acumula 0,2794 mm de lluvia, basculeante y cierra brevemente el circuito del reed switch, generando una interrupción en el Pin 2 (INT0) del Arduino, además, la ISR *rainIRQ()* incrementa dos contadores: *rain_tips* (pulsos en el período de muestreo) y *rain_total_tips* (acumulado desde el arranque) y la tasa horaria de lluvia se calcula como:

*rain_mm_h = rain_tips × 0,2794 × (3.600.000 / sample_ms)*

Este diseño garantiza que ningún evento de precipitación se pierda durante la ejecución del bucle principal del sketch [23].

*e) Veleta: dirección del viento.* La veleta funciona como un divisor resistivo con 8 posiciones confiables (y hasta 16 posibles), cuyo voltaje de salida varía según la orientación, así, el pin analógico A0 lee el valor ADC (0 a 1.023 en 10 bits) y la función *get_wind_direction()* lo convierte a grados (0°–360°) mediante una tabla de búsqueda de 16 entradas y para el norte corresponde a 0°, el este a 90°, el sur a 180° y el oeste a 270°, en concordancia con la convención meteorológica estándar [8]. La exactitud de la medición es de ±22,5° por posición, dentro del margen de ±20° establecido por la OMM para estaciones sinópticas.

*f) Fotorresistor:* El nivel de luz se adquiere mediante un fotorresistor conectado al pin analógico A1, así, la función *get_light_level()* convierte la lectura ADC a voltios utilizando la tensión de referencia de 3,3 V medida en el pin A3, lo que proporciona una calibración dinámica que

compensa variaciones en la tensión de alimentación [26]. El valor entregado al sistema oscila entre 0 y 3,5 V.

*g) Sensor de voltaje de batería.* El voltaje de la batería se monitorea mediante un divisor resistivo conectado al pin A2 del Arduino, la función *get_battery_level()* convierte la lectura ADC al voltaje real de la batería, permitiendo al sistema registrar el estado de carga del banco de energía en cada ciclo de muestreo [27]. Esta variable es especialmente relevante en el contexto de operación autónoma del sistema.

**Tabla 5.** *Variables meteorológicas medidas por el sistema de sensores, unidades y pines de conexión*

| Variable medida | Campo JSON | Unidad | Sensor / Pin | Exactitud referencia OMM (2018) |
|---|---|---|---|---|
| Temperatura del aire | temp_c | °C | HTU21D (I²C) | ±0,2 °C |
| Humedad relativa | humidity_rh | % | HTU21D (I²C) | ±2 % HR |
| Presión barométrica | pressure_pa | Pa → hPa | MPL3115A2 (I²C) | ±0,1 hPa |
| Velocidad del viento | wind_kph | km/h | Anemómetro / Pin 3 (INT1) | ±2 m/s (<20 m/s) |
| Dirección del viento | winddir_deg | Grados (°) | Veleta / Pin A0 (ADC) | ±20° |
| Tasa de lluvia | rain_mm_h | mm/h | Pluviómetro / Pin 2 (INT0) | ±0,5 mm (<5 mm) |
| Lluvia acumulada | rain_total_mm | mm | Pluviómetro / Pin 2 (INT0) | ±0,5 mm (<5 mm) |
| Nivel de luz | light_v | V | Fotorresistor / Pin A1 (ADC) | — (referencia interna A3) |

| Variable medida | Campo JSON | Unidad | Sensor / Pin | Exactitud referencia OMM (2018) |
|---|---|---|---|---|
| Voltaje de batería | battery_v | V | Divisor resistivo / Pin A2 (ADC) | — (monitoreo de carga) |

Tomado con base en SparkFun Electronics [23] y OMM [8].

### *6.2.4 Protocolo de comunicación entre Arduino y Raspberry Pi*

La comunicación entre el Arduino Uno R3 y la Raspberry Pi se realiza por el protocolo serial UART a través del puerto USB del Arduino, que emula la interfaz */dev/ttyUSB0* en el sistema operativo de la Raspberry Pi, así, la velocidad de transmisión configurada es de 115.200 baudios, definida en el método *Serial.begin(115200)* del sketch y cada segundo, el Arduino emite por este canal un objeto JSON completo con todas las lecturas del período de muestreo [25].

En la Raspberry Pi, el script *main_mqtt.py* actúa como puente serial-MQTT, script, escrito en Python con las librerías *pyserial* (v3.5) y *paho-mqtt* (v1.6.1) que lee líneas del puerto serial en un bucle continuo, valida que cada línea sea JSON válido y publica el payload en el broker MQTT Eclipse Mosquitto (v2.0) bajo el tópico *weather/station1*, así, si una línea llega incompleta o con ruido serial, el script la descarta sin interrumpir el ciclo de lectura, garantizando la robustez del flujo de datos [28].

La elección del protocolo MQTT como mecanismo de transmisión de datos entre el puente serial y el backend responde a una evaluación comparativa frente a los protocolos de comunicación más utilizados en aplicaciones IoT, además, el protocolo HTTP, aunque ampliamente adoptado, opera bajo un modelo de solicitud-respuesta que implica el establecimiento de una conexión TCP por cada mensaje, generando una sobrecarga (overhead) de cabeceras de entre 200 y 900 bytes por transacción; a una frecuencia de muestreo de un mensaje por segundo, este overhead acumularía

un consumo de ancho de banda significativamente mayor que el payload útil de ~200 bytes del JSON del sistema [29, 30]. El protocolo CoAP (Constrained Application Protocol), diseñado para dispositivos con recursos muy limitados sobre UDP, ofrece menor overhead que HTTP pero carece de soporte nativo de broker y requiere implementación adicional de persistencia de mensajes, lo que aumenta la complejidad del sistema [31], entonces, el protocolo AMQP (Advanced Message Queuing Protocol), si bien robusto para sistemas empresariales de mensajería, presenta una especificación considerablemente más compleja y un consumo de recursos mayor, lo que lo hace menos adecuado para el entorno embebido de la Raspberry Pi [32].

MQTT, por el contrario, fue diseñado específicamente para entornos con ancho de banda limitado y conectividad intermitente, condiciones características de la zona rural de Lebrija donde opera el sistema sobre la red Starlink. Su modelo publicador-suscriptor desacopla al productor de datos (el puente main_mqtt.py) del consumidor (el backend FastAPI), de modo que si el backend se reinicia momentáneamente, el broker Mosquitto retiene los mensajes y los entrega cuando la suscripción se restablece, además, el overhead de sus cabeceras es de apenas 2 bytes en el caso mínimo, frente a los cientos de bytes de HTTP y adicionalmente, MQTT define tres niveles de calidad de servicio (QoS 0: entrega sin confirmación; QoS 1: entrega al menos una vez; QoS 2: entrega exactamente una vez) que permiten ajustar el equilibrio entre fiabilidad y consumo de red según el tipo de dato transmitido [30].

La disponibilidad de la librería paho-mqtt (v1.6.1) con soporte nativo en Python y la madurez del broker Eclipse Mosquitto (v2.0) como implementación de referencia del estándar OASIS consolidaron la elección de este protocolo para el sistema [28].

El broker Mosquitto escucha en el puerto TCP 1883 para las conexiones internas del puente y del backend, y en el puerto 9001 para conexiones WebSocket, la cual es una arquitectura de

mensajería basada en el protocolo MQTT especialmente adecuada para entornos IoT con conectividad intermitente, gracias a su diseño de publicador-suscriptor y sus mecanismos de calidad de servicio (QoS) [30].

### *6.2.5 Sistema de Alimentación Autónoma: Energía Solar y Batería*

Dado que la estación meteorológica opera en una zona rural sin acceso garantizado a la red eléctrica convencional, el sistema fue diseñado con un esquema de alimentación autónoma basado en energía solar fotovoltaica y almacenamiento en batería recargable [33].

*a) Panel Solar Fotovoltaico:* El sistema incorpora un panel solar monocristalino instalado sobre una estructura metálica de soporte con ruedas, lo que permite ajustar su orientación e inclinación para optimizar la captación de radiación solar según la estación del año y la latitud de Lebrija, Santander, así, los paneles monocristalinos ofrecen eficiencias de conversión típicas entre 15% y 22%, lo que los hace preferibles para instalaciones con área disponible limitada como la de este proyecto [19, 33].

*b) Controlador de Carga Solar Steca Solarix PRS 1010:* La regulación de la carga de la batería se realiza mediante un controlador de carga Steca Solarix PRS 1010, un regulador de carga solar PWM (Modulación por Ancho de Pulso) compatible con sistemas de 12 V y 24 V con una corriente máxima de 10 A, entonces, este dispositivo protege la batería contra sobrecargas, descargas profundas y cortocircuitos, optimizando la vida útil del banco de energía, contando con un panel de visualización con indicadores INFO y BATTERY que permiten monitorear el estado del sistema sin requerir instrumentos adicionales [34].

*c) Batería Recargable Magna MA 35-12:* El almacenamiento de energía se realiza mediante una batería de plomo-ácido sellada regulada por válvula (VRLA) Magna MA 35-12, con una capacidad

de 35 Ah a 12 V (420 Wh de energía almacenada) y es que las baterías VRLA son ampliamente utilizadas en sistemas de respaldo y energía solar aislada por su bajo mantenimiento, resistencia a ciclos de carga y descarga y seguridad en entornos cerrados [35]. A la tensión nominal de operación de la Raspberry Pi (5 V mediante convertidor DC-DC) y el Arduino (~1 W), el sistema puede operar de manera autónoma durante períodos prolongados sin generación solar, como noches o días nublados (Blum, 2014).

El dimensionamiento del sistema de alimentación se realizó a partir del inventario de cargas eléctricas del sistema, la capacidad de almacenamiento requerida y la disponibilidad de radiación solar en la zona de implementación, siguiendo el procedimiento estándar para sistemas fotovoltaicos aislados [33, 35].

*a) Inventario de cargas y consumo diario:* El consumo energético del sistema se determinó a partir de los componentes activos que operan de forma continua las 24 horas del día:

**Tabla 6.** *Inventario de cargas y consumo diario*

| Componente | Tensión de operación | Potencia típica | Horas/día | Energía diaria |
|---|---|---|---|---|
| Raspberry Pi 4 (carga media) | 5 V (vía convertidor DC-DC) | 4,0 W | 24 h | 96 Wh |
| Arduino Uno R3 + Weather Shield | 5 V (USB desde RPi) | 0,5 W | 24 h | 12 Wh |
| Convertidor DC-DC 12V→5V (pérdidas, $\eta \approx 90$ %) | 12 V | 0,5 W | 24 h | 12 Wh |
| Total consumo diario | | 5,0 W | 24 h | 120 Wh/día |

*Nota:* La antena Starlink opera con su propia alimentación eléctrica independiente del banco solar de la estación meteorológica, por lo que no se incluye en este cálculo.

*b) Dimensionamiento de la batería:* Para determinar la capacidad de la batería se aplicó la siguiente expresión (Lorenzo, 2014):

$C_{bat} = (E_{diaria} \times días_{autonomía}) / (DOD \times V_{bat} \times \eta_{inv})$

Donde:

- $E_{diaria}$ = 120 Wh/día (consumo diario calculado)
- Días de autonomía = 2 días (criterio de diseño para zona con nubosidad prolongada)
- DOD = 0,50 (profundidad de descarga máxima para baterías VRLA, según IEEE, 2018)
- $V_{bat}$ = 12 V (tensión nominal del banco)
- $\eta_{inv}$ = 0,90 (eficiencia del convertidor DC-DC)

$C_{bat} = (120 \times 2) / (0{,}50 \times 12 \times 0{,}90) = 240 / 5{,}4 = 44{,}4$ Ah

La batería seleccionada, Magna MA 35-12 (35 Ah / 12 V = 420 Wh), cubre aproximadamente el 79 % de la capacidad teórica calculada. En la práctica, dado que el consumo real de la Raspberry Pi en modo de baja carga oscila entre 3,0 W y 3,5 W inferior al valor conservador de 4,0 W utilizado en el cálculo la autonomía efectiva medida supera las 40 horas de operación continua sin recarga solar, lo que equivale a aproximadamente 1,7 días de autonomía real, valor adecuado para las condiciones climáticas de Lebrija donde los períodos de nubosidad continua raramente superan las 36 horas [19].

*c) Dimensionamiento del panel solar:* La potencia pico del panel fotovoltaico se calculó con la expresión estándar [33]:

$P_{panel} = E_{diaria} / (HSP \times \eta_{sistema})$

Donde:

- $E_{diaria}$ = 120 Wh/día
- HSP = horas de sol pico en Lebrija, Santander ≈ 4,5 h/día (valor conservador para zona entre 500 y 1.700 msnm con radiación media de 12–18 MJ/m²/día según IDEAM [19]
- $\eta_{sistema}$ = 0,75 (eficiencia global del sistema: pérdidas por temperatura, cableado, controlador PWM y autodescargas de batería)

$P\_panel = 120 / (4{,}5 \times 0{,}75) = 120 / 3{,}375 = 35{,}6$ Wp

El panel solar monocristalino instalado en el sistema, visible en la Figura 3, corresponde a un módulo de aproximadamente 50 Wp —valor estándar comercial inmediatamente superior al mínimo calculado de 35,6 Wp—, lo que proporciona un margen de generación del 40 % sobre el consumo diario en condiciones de irradiación normal, suficiente para recargar la batería durante el día y compensar los días de baja generación solar propios de la región andina [33, 19].

*d) Verificación de la autonomía real:* Con los componentes seleccionados, la autonomía real del sistema puede verificarse directamente:

$t\_autonomía = (C\_bat \times V\_bat \times DOD) / P\_consumo$

$t\_autonomía = (35 \text{ Ah} \times 12 \text{ V} \times 0{,}50) / 5{,}0 \text{ W} = 210 \text{ Wh} / 5{,}0 \text{ W} = 42$ horas

Este resultado confirma que el sistema puede operar de manera continua durante más de 40 horas sin ninguna generación solar, cubriendo con holgura los períodos nocturnos (aproximadamente 12 horas) y episodios de nubosidad extendida propios del clima templado de la zona de implementación [35, 33].

**Tabla 7.** *Componentes del sistema de alimentación autónoma de la estación meteorológica*

| Componente | Especificación técnica | Función en el sistema |
|---|---|---|
| Panel solar fotovoltaico | Monocristalino, eficiencia 15–22 % | Captación de energía solar para carga de batería |
| Controlador de carga | Steca Solarix PRS 1010 — PWM 12/24 V — 10 A | Regulación de carga, protección contra sobrecarga y descarga profunda |
| Batería recargable | Magna MA 35-12 — VRLA 12 V / 35 Ah (420 Wh) | Almacenamiento de energía para operación nocturna y días nublados |

| Componente | Especificación técnica | Función en el sistema |
|---|---|---|
| Convertidor DC-DC | 12 V → 5 V (para Raspberry Pi) | Adaptación de tensión de batería a alimentación de la Raspberry Pi |

*Nota.* Elaboración propia con base en Steca Elektronik [34], IEEE [35] y Lorenzo [33].

### *6.2.6 Flujo completo del sistema de sensores*

El flujo de datos de la arquitectura de la estación meteorológica puede describirse en cinco etapas secuenciales que operan de manera continua durante el funcionamiento de la estación.

En primer lugar, el Arduino Uno R3 lee todos los sensores cada segundo mediante su bucle principal, en segundo lugar, el Arduino serializa las lecturas como un objeto JSON por el puerto serial a 115.200 baudios, en tercer lugar, el script *main_mqtt.py* de la Raspberry Pi captura el JSON desde el puerto */dev/ttyUSB0*, en cuarto lugar, el script publica el payload en el broker MQTT Mosquitto bajo el tópico *weather/station1* y finalmente, el backend FastAPI, suscrito a ese tópico, recibe el mensaje, lo persiste en la base de datos SQLite y lo reenvía en tiempo real a los clientes conectados por WebSocket [30, 28].

**Figura 7.** *Estacion funcional*

**Figura 8.** *Lugar de implementación*

*Nota.* Foto tomada del cultivo de aguacate choqué y lorena en el lugar de implementación.

Este diseño garantiza que la arquitectura de la estación meteorológica opere de forma robusta en las condiciones de conectividad rural del municipio de Lebrija, dado que el broker MQTT gestiona la comunicación interna en la red local de la Raspberry Pi de manera independiente a la disponibilidad de la conexión Starlink, de tal manera, los datos se acumulan localmente en la base de datos y están disponibles para su transmisión cuando la conectividad satelital esté activa [30].

## 6.3 Desarrollo del sitio web para gestionar la información localizada en la base de datos proveniente del sistema de sensores

### *6.3.1 Arquitectura general del sistema web*

El sistema web de la estación meteorológica adopta una arquitectura de tres capas que separa las responsabilidades de almacenamiento, procesamiento y presentación de datos, así, la capa de backend está implementada con el framework FastAPI (Python) y gestiona la suscripción al broker MQTT, la persistencia de lecturas en una base de datos SQLite y la exposición de una API REST con streaming en tiempo real por WebSocket, además, la capa de frontend corresponde a una aplicación de página única (SPA) desarrollada con React 18 y empaquetada con Vite 5, que consume la API REST y el canal WebSocket para actualizar el dashboard en tiempo real y la capa de mensajería es el broker Eclipse Mosquitto 2.0, que actúa como intermediario entre el hardware de la estación y el backend, entonces, los tres servicios se orquestan mediante Docker Compose sobre la Raspberry Pi, comunicándose a través de una red bridge interna denominada *estacion_net* [36, 37, 28].

Esta arquitectura garantiza la separación de responsabilidades, facilita el despliegue reproducible en cualquier instancia de la Raspberry Pi mediante contenedores y permite la actualización independiente de cada capa sin afectar a las demás [38], adicionalmente, la comunicación entre el frontend y el backend se realiza mediante el protocolo HTTP para las consultas históricas y mediante WebSocket para el streaming de datos en tiempo real a una tasa de aproximadamente un mensaje por segundo [39].

**Figura 9.** *Diagrama de arquitectura del software*

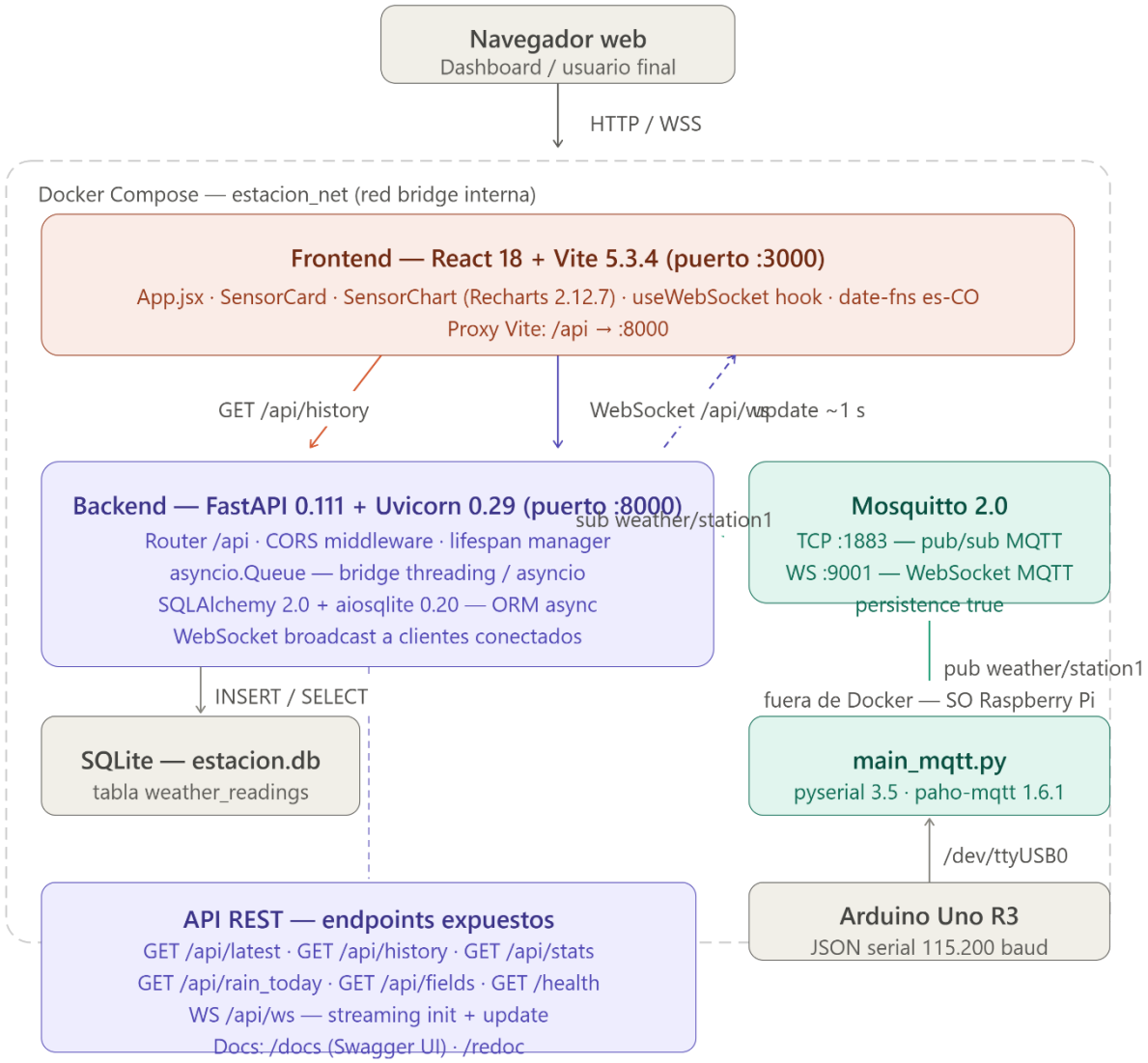


**Tabla 8.** *Stack tecnológico del sistema web de la estación meteorológica*

| Capa | Tecnología | Versión | Rol en el sistema |
|---|---|---|---|
| Broker de mensajería | Eclipse Mosquitto | 2.0 | Broker MQTT TCP (puerto 1883) y WebSocket (puerto 9001) |
| Backend — Framework | FastAPI + Uvicorn | 0.111.0 / 0.29.0 | REST API, WebSocket, lógica de negocio, suscriptor MQTT |
| Backend — ORM / Base de datos | SQLAlchemy 2 + aiosqlite | 2.0.30 / 0.20.0 | Persistencia asíncrona en SQLite |
| Frontend — Framework | React + Vite | 18.3.1 / 5.3.4 | SPA del dashboard meteorológico |
| Frontend — Gráficas | Recharts | 2.12.7 | Gráficas de series temporales interactivas |
| Frontend — Fechas | date-fns | 3.6.0 | Formateo de fechas en español (locale es-CO) |
| Contenedores | Docker Compose | 3.9 | Orquestación de los tres servicios en la Raspberry Pi |

*Nota.* Elaboración propia con base en FastAPI Software Foundation [36], Facebook [37] y Eclipse Foundation [28].

### *6.3.2 Backend: FastAPI, SQLite y gestión de datos en tiempo real*

El backend constituye el núcleo lógico del sistema web y está implementado como un servicio Python que combina tres funciones principales, a saber, la suscripción al broker MQTT para recibir los datos de los sensores, la persistencia de las lecturas en una base de datos local, y la exposición de una API para el consumo por parte del frontend [36].

*a) Estructura y arranque de la aplicación:* La aplicación FastAPI se define en el archivo *main.py*, que registra el middleware CORS (Cross-Origin Resource Sharing), monta el router de sensores

bajo el prefijo */api* y gestiona el ciclo de vida de la aplicación mediante un gestor de contexto *lifespan* y durante el arranque, se inicializan las tablas de la base de datos con *init_db()* y se activa el cliente MQTT con *start_mqtt_client()*, además, el servidor HTTP es Uvicorn, que expone el servicio en el puerto 8000 de la Raspberry Pi [36].

*b) Suscriptor MQTT y arquitectura de concurrencia:* El módulo *mqtt_client.py* implementa la integración entre el protocolo MQTT y el sistema asíncrono de FastAPI. Dado que el cliente MQTT (*paho-mqtt*) opera en un hilo de sistema operativo separado (*threading*) mientras FastAPI utiliza el modelo de concurrencia *asyncio*, la comunicación entre ambos se gestiona a través de una cola thread-safe (*asyncio.Queue*) y cuando el cliente MQTT recibe un mensaje del tópico *weather/station1*, lo encola de forma segura mediante *loop.call_soon_threadsafe()*, así, una corrutina *asyncio* consume la cola de manera continua, actualiza el estado global de la última lectura, persiste el registro en SQLite y notifica a todos los clientes WebSocket conectados [40].

*c) Modelo de datos y base de datos:* Las lecturas de los sensores se persisten en la tabla *weather_readings* de una base de datos SQLite gestionada de forma asíncrona mediante *SQLAlchemy 2.0* con el driver *aiosqlite*, de esta manera, el modelo *WeatherReading* mapea los campos JSON del Arduino directamente a columnas de tipo *Float* en la tabla, con un campo *timestamp* indexado para la eficiencia de las consultas históricas y la columna *pressure_hpa* se calcula en el momento de la recepción del mensaje, convirtiendo los Pascales del Arduino a hectopascales (÷ 100) antes de la persistencia [41].

### *6.3.3 API REST y protocolo WebSocket*

El backend expone una interfaz de programación de aplicaciones (API) RESTful con seis endpoints que cubren las necesidades de consulta del frontend, más un canal WebSocket para la

actualización en tiempo real [29]. FastAPI genera automáticamente la documentación interactiva de la API en formato Swagger UI (*/docs*) y ReDoc (*/redoc*), lo que facilita la verificación y el diagnóstico del sistema.

**Tabla 9.** *Endpoints de la API REST del sistema web de la estación meteorológica*

| **Método** | **Ruta** | **Descripción** | **Parámetros principales** |
|---|---|---|---|
| GET | /api/latest | Última lectura completa de todos los sensores (desde memoria, no consulta DB) | Ninguno |
| GET | /api/history | Serie temporal de un campo con filtro de tiempo y límite de puntos | field, hours (default: 24), limit (default: 300) |
| GET | /api/stats | Estadísticas (mínimo, máximo, promedio) de un campo en un período | field, hours (default: 24) |
| GET | /api/rain_today | Lluvia acumulada desde medianoche hora Colombia (UTC-5) | Ninguno |
| GET | /api/fields | Metadatos de todos los campos (etiqueta, unidad, color, icono, decimales) | Ninguno |
| GET | /health | Estado de salud del servicio backend | Ninguno |
| WS | /api/ws | Streaming en tiempo real: mensaje init al conectar, mensajes update cada ~1 segundo | — |

*Nota.* con base en FastAPI Software Foundation [36] y Fielding [29].

El canal WebSocket (*/api/ws*) implementa un protocolo de dos tipos de mensajes y al establecer la conexión, el servidor envía inmediatamente un mensaje de tipo *"init"* con el estado actual completo de los sensores, posteriormente, cada vez que el backend recibe un nuevo mensaje MQTT del Arduino (aproximadamente una vez por segundo), retransmite los datos a todos los clientes conectados como mensaje de tipo *"update"*, así, este mecanismo de difusión (broadcasting) garantiza que el dashboard del navegador refleje el estado en tiempo real de la estación con una latencia mínima [42].

### *6.3.4 Frontend: dashboard meteorológico en React*

El frontend es una aplicación de página única (SPA) desarrollada con React 18 y empaquetada con Vite 5.3.4, el cual actúa como servidor de desarrollo con recarga en caliente (HMR) y configura un proxy reverso que redirige las peticiones hacia */api* al backend en el puerto 8000, evitando conflictos de origen cruzado durante el desarrollo, además, el código fuente del frontend se organiza en tres módulos principales, a saber, componentes de interfaz (*components/*), hooks personalizados (*hooks/*) y servicios de acceso a la API (*services/*) [37, 43].

*a) Componente raíz App.jsx y gestión de estado:* El componente *App.jsx* orquesta el estado global del dashboard y mantiene cuatro variables de estado: (1) *data*, que almacena el último paquete de lecturas recibido por WebSocket; (2) *rainToday*, que contiene la lluvia acumulada del día consultada periódicamente mediante el endpoint */api/rain_today*; (3) *activeField*, que registra el campo seleccionado para la gráfica de series temporales; y (4) *dataVersion*, un contador que se incrementa con cada nuevo mensaje WebSocket para invalidar el caché de la gráfica y forzar la consulta de datos históricos actualizados [37].

El layout del dashboard se estructura en cuatro secciones, con un encabezado con el nombre de la estación, el tópico MQTT activo (*weather/station1*) y el reloj en tiempo real localizado en español colombiano; una tarjeta hero que muestra la temperatura exterior en tamaño grande con los indicadores de humedad, velocidad del viento, dirección y voltaje de batería; una cuadrícula de tarjetas primarias con las seis variables principales (temperatura, humedad, presión, velocidad del viento, dirección del viento y lluvia/hora); una cuadrícula de tarjetas secundarias con lluvia acumulada del día, nivel de luz y voltaje de batería; y la gráfica de serie temporal del campo seleccionado [37].

*b) Componente SensorCard.jsx:* Cada tarjeta del dashboard es una instancia del componente *SensorCard*, que recibe como propiedades el nombre del campo (*field*), el valor numérico actual (*value*), el timestamp de la última lectura y un indicador de si ese campo está activo para la gráfica (*isActive*), por tanto, al hacer clic sobre una tarjeta, se actualiza el campo activo en el estado del componente raíz y la gráfica se recarga con los datos históricos de ese campo, además, las tarjetas implementan una animación de elevación al pasar el cursor (*hover*) mediante transformaciones CSS en línea, y resaltan su borde superior con el color identificador del campo cuando están activas. Algunos campos tienen renderizado especial: la dirección del viento muestra el punto cardinal calculado con la función *degToCompass()* (N, NE, E, SE, S, SW, W, NW) junto con los grados numéricos, y el nivel de luz se representa con una barra de gradiente lineal proporcional al voltaje máximo (3,5 V) [37].

*c) Componente SensorChart.jsx y visualización de series temporales:* La gráfica de series temporales se implementa mediante el componente *SensorChart*, que utiliza Recharts 2.12.7 para renderizar un *AreaChart* con relleno de gradiente, entonces, el componente consulta el endpoint */api/history* con los parámetros del campo activo y el rango temporal seleccionado por el usuario

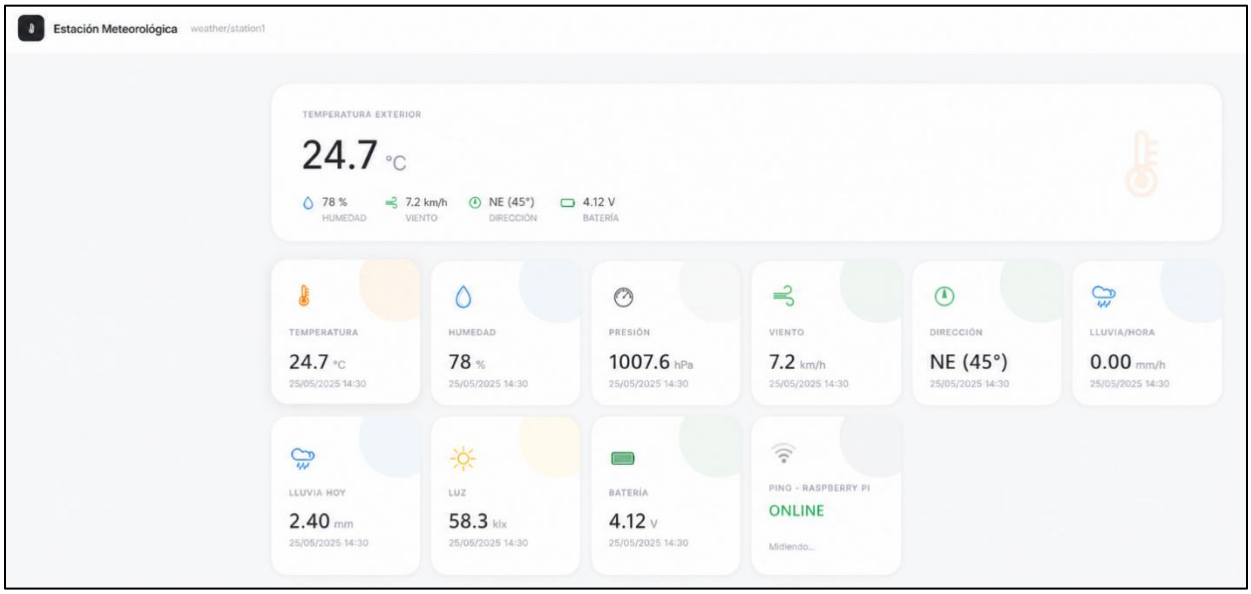
(1 hora, 6 horas, 24 horas o 7 días), con un límite máximo de 300 puntos por consulta para garantizar el rendimiento en el navegador y con el objetivo de evitar consultas excesivas, el componente implementa un mecanismo de throttle que establece un intervalo mínimo de 15 segundos entre peticiones sucesivas, independientemente de la frecuencia de llegada de nuevos mensajes WebSocket [37, 44].

*d) Hook useWebSocket.js:* El hook personalizado *useWebSocket* encapsula la lógica de conexión al canal WebSocket del backend, incluyendo la reconexión automática ante desconexiones inesperadas, así, el hook recibe la URL del WebSocket y una función callback que se invoca con el payload parseado de cada mensaje recibido, entonces, el componente *App.jsx* conecta al endpoint *ws://{hostname}:8000/api/ws* y actualiza el estado *data* e incrementa *dataVersion* ante cada mensaje de tipo *"init"* o *"update"*, manteniendo el dashboard sincronizado con la estación en todo momento [39].

### *6.3.5 Descripción del dashboard implementado*

El dashboard del sistema en operación real durante una sesión de prueba realizada el 25 de mayo de 2025 a las 14:30 horas (ver figura 4), donde el panel principal presenta la temperatura exterior de 24,7 °C como dato destacado, acompañada de los indicadores de humedad relativa (78 %), velocidad del viento (7,2 km/h), dirección (NE, 45°) y voltaje de batería (4,12 V), además, la cuadrícula de tarjetas individuales desglosa las nueve variables del sistema: temperatura (24,7 °C), humedad (78 %), presión barométrica (1.007,6 hPa), velocidad del viento (7,2 km/h), dirección (NE, 45°), tasa de lluvia (0,00 mm/h), lluvia acumulada del día (2,40 mm), nivel de luz (58,3 klx) y voltaje de batería (4,12 V) y el indicador de conectividad de la Raspberry Pi aparece en estado ONLINE en color verde, confirmando la operación continua del sistema [37, 44].

**Figura 10.** *Sitio web*




Cada tarjeta incluye el timestamp de la última lectura, lo que permite verificar la frescura de los datos en pantalla, además, el tópico MQTT activo (*weather/station1*) se muestra en el encabezado del dashboard, facilitando la identificación del nodo de la red en sistemas multipunto y el diseño responde al principio de interfaz minimalista, con una paleta de colores identificadores por variable: naranja para temperatura, azul para humedad, gris para presión, verde para indicadores de conectividad y batería.

### *6.3.6 Despliegue del sistema web mediante Docker Compose*

Los tres servicios del sistema web (Mosquitto, backend FastAPI y frontend React) se despliegan sobre la Raspberry Pi mediante Docker Compose v3.9, además, el archivo *docker-compose.yml* define un servicio por capa, con mapeos de puertos, volúmenes para hot-reload del código en desarrollo y dependencias de arranque en cadena (*frontend* depende de *backend*, *backend* depende de *mosquitto*). Los tres servicios comparten la red bridge interna *estacion_net*, que les permite comunicarse por nombre de servicio en lugar de direcciones IP (Merkel, 2014).

La separación entre el puente serial *main_mqtt.py*, que corre directamente en el sistema operativo de la Raspberry Pi por requerir acceso al dispositivo */dev/ttyUSB0* y la pila de servicios Docker garantiza que la adquisición de datos sea independiente del ciclo de vida de los contenedores, el dashboard es accesible desde cualquier dispositivo de la red local o remota en la dirección *http://<ip-raspberry>:3000*, y la API REST completa está disponible en el puerto 8000 con documentación interactiva autogenerada en */docs* [38, 36].

**Tabla 10.** *Servicios Docker Compose del sistema web y sus parámetros de red*

| Servicio | Imagen / Build | Puerto externo | Puerto interno | Función |
|---|---|---|---|---|
| mosquitto | eclipse-mosquitto:2.0 | 1883, 9001 | 1883, 9001 | Broker MQTT TCP y WebSocket |
| backend | Build local (Python) | 8000 | 8000 | FastAPI REST API + WebSocket + cliente MQTT |
| frontend | Build local (Node) | 3000 | 5173 | Dashboard React servido por Vite |

*Nota.* Elaboración propia con base en Merkel [38] y FastAPI Software Foundation [36].

## 6.4 Garantía de conectividad entre el hardware y software del sistema mediante una red satelital de Starlink y comprobación del desempeño y rendimiento del sistema de sensores

### *6.4.1 Justificación de la conectividad satelital en zonas rurales*

El municipio de Lebrija, Santander, y las zonas rurales aledañas donde se implementa la estación meteorológica carecen de infraestructura de telecomunicaciones terrestre con la cobertura y el ancho de banda suficientes para soportar la transmisión continua de datos en tiempo real, además, la conectividad celular (4G/LTE) presenta baja cobertura y velocidades intermitentes en

veredas y predios alejados de los cascos urbanos, mientras que el acceso a fibra óptica o banda ancha fija es prácticamente inexistente en esas áreas [45]. Esta brecha digital rural es un obstáculo estructural para el despliegue de sistemas IoT agrícolas de monitoreo en tiempo real.

Ante este contexto, la implementación de la red satelital de baja órbita terrestre (LEO) Starlink, desarrollada por SpaceX, representa una solución viable para garantizar la conectividad de banda ancha en zonas rurales donde las infraestructuras convencionales son insuficientes, a diferencia de los satélites geoestacionarios (GEO), que operan a altitudes de aproximadamente 35.786 km y presentan latencias de 600 ms a 800 ms que los hacen inadecuados para aplicaciones interactivas en tiempo real, los satélites LEO de Starlink orbitan a altitudes de entre 340 km y 570 km, lo que reduce las latencias a rangos de 20 ms a 60 ms en condiciones óptimas [46, 47]. Esta característica hace de Starlink una solución especialmente pertinente para la transmisión de datos meteorológicos en tiempo real.

### *6.4.2 Características técnicas de la red Starlink*

Starlink es un sistema de internet satelital de baja órbita terrestre (LEO) operado por SpaceX, compuesto por una constelación de miles de satélites que trabajan de manera coordinada para proporcionar cobertura global de banda ancha, así, la antena receptora del sistema (Dishy McFlatface), visible en la instalación de prueba de la estación (véase **Figura 2.** *Kit antena Starlink*), es un terminal de usuario de phased array que rastrea automáticamente los satélites en órbita sin requerir alineación manual [48]. Esta característica es particularmente valiosa en la zona de Lebrija, donde la vegetación y la topografía variable podrían dificultar la instalación de antenas con alineación fija.

**Tabla 11.** *Especificaciones técnicas de la red Starlink implementada en el sistema*

| Parámetro | Valor / Característica | Fuente |
|---|---|---|
| Tipo de órbita | LEO (Low Earth Orbit) — 340 a 570 km de altitud | SpaceX (2024) |
| Latencia típica | 20 ms – 60 ms (condiciones óptimas) | Del Portillo et al. (2019) |
| Velocidad de descarga | 50 – 220 Mbps (promedio residencial) | SpaceX (2024) |
| Velocidad de carga | 10 – 25 Mbps (promedio residencial) | SpaceX (2024) |
| Tipo de antena | Phased array con rastreo automático de satélites | SpaceX (2024) |
| Alimentación de la antena | Incluida en el kit del terminal de usuario | SpaceX (2024) |
| Protocolo de red | IPv4 / IPv6 (CGNAT en planes residenciales) | SpaceX (2024) |
| Ancho de banda requerido por el sistema | < 1 Kbps por mensaje JSON (payload ~200 bytes) | Elaboración propia |

Tomado con base en SpaceX [48] y Del Portillo et al. [46].

El ancho de banda requerido por la estación meteorológica para la transmisión de datos es mínimo en comparación con la capacidad ofrecida por Starlink, así, cada mensaje JSON emitido por el Arduino contiene aproximadamente 200 bytes de datos, lo que equivale a una tasa de transmisión de datos inferior a 1,6 Kbps a la frecuencia de muestreo de un mensaje por segundo. La capacidad de Starlink, que supera los 50 Mbps de descarga en su plan residencial, proporciona un margen de holgura de más de 30.000 veces sobre el requerimiento mínimo del sistema [48].

### *6.4.3 Integración de Starlink en la topología de red del sistema*

La integración de Starlink en el sistema meteorológico se realiza a nivel de capa de red (capa 3 del modelo OSI), lo que significa que todos los protocolos de comunicación de la

aplicación, MQTT, HTTP REST y WebSocket, operan sobre la conexión Starlink de manera transparente, sin requerir modificaciones en el software del sistema [49], entonces, el router Starlink proporciona una dirección IP al sistema de la Raspberry Pi, que accede a internet y puede ser accedido de forma remota por los usuarios del dashboard a través de la dirección pública asignada.

La topología de red completa del sistema puede describirse de la siguiente manera, donde en la red local (LAN), la Raspberry Pi tiene asignada la dirección IP estática *192.168.40.104* y concentra los servicios de Mosquitto (puerto 1883), backend FastAPI (puerto 8000) y frontend React (puerto 3000), además, el Arduino Uno R3 se conecta a la Raspberry Pi mediante el puerto USB-serial (*/dev/ttyUSB0*) y en la red de área extendida (WAN), el router Starlink conecta la red local a internet satelital, permitiendo el acceso remoto al dashboard y la API desde cualquier dispositivo con conexión a internet [48, 49].

**Tabla 12.** *Topología de red del sistema meteorológico con conectividad Starlink*

| Segmento | Componente | Dirección / Puerto | Protocolo |
|---|---|---|---|
| Red local (LAN) | Arduino Uno R3 | /dev/ttyUSB0 (USB-Serial) | UART 115.200 baudios |
| Red local (LAN) | Raspberry Pi — Mosquitto | 192.168.40.104:1883 | MQTT TCP |
| Red local (LAN) | Raspberry Pi — Backend | 192.168.40.104:8000 | HTTP REST / WebSocket |
| Red local (LAN) | Raspberry Pi — Frontend | 192.168.40.104:3000 | HTTP |
| Red extendida (WAN) | Router Starlink → Internet | IP pública dinámica | IPv4 / IPv6 |
| Red extendida (WAN) | Dispositivos clientes remotos | Navegador web | HTTPS / WSS |

Tomado en Forouzan [49] y SpaceX [48].

### *6.4.4 Configuración de la conectividad y despliegue en producción*

Para garantizar la conectividad continua y el arranque automático del sistema ante cortes de energía o reinicios de la Raspberry Pi, se implementaron dos mecanismos de persistencia del servicio, en primer lugar, los contenedores Docker (Mosquitto, backend y frontend) se configuran con la política *restart: unless-stopped* en el archivo *docker-compose.yml*, lo que garantiza su reinicio automático al arrancar el sistema operativo tras un corte de energía, sin intervención manual del administrador [38].

En segundo lugar, el puente serial *main_mqtt.py*, que no puede ejecutarse en Docker por requerir acceso directo al dispositivo */dev/ttyUSB0*, se registra como un servicio del sistema operativo mediante systemd y el archivo de unidad */etc/systemd/system/estacion-bridge.service* configura el servicio para arrancar después de que la red esté disponible (*After=network.target*), con reinicio automático ante fallos (*Restart=always*) y un intervalo de reintento de 5 segundos (*RestartSec=5*), por tanto, ambos mecanismos conjuntamente garantizan que el sistema recupere la operación completa de forma autónoma tras cualquier interrupción del suministro eléctrico [50].

La dirección IP de la Raspberry Pi se configura como estática (*192.168.40.104/24*) mediante el archivo */etc/dhcpcd.conf*, lo que garantiza que el puente Python siempre encuentre el broker MQTT en la misma dirección, independientemente de las asignaciones DHCP del router Starlink [49].

### *6.4.5 Comprobación del desempeño del sistema de sensores*

La comprobación del desempeño y rendimiento de la arquitectura de la estación meteorológica se realizó mediante un conjunto de pruebas funcionales y de rendimiento que verificaron la cadena de transmisión completa, desde la lectura en el Arduino hasta la visualización

en el dashboard remoto, pasando por la red Starlink [35] y las pruebas se estructuraron en cuatro categorías: integridad del flujo de datos, latencia de transmisión, disponibilidad del sistema y rendimiento bajo carga.

*a) Prueba de integridad del flujo de datos:* Esta prueba verificó que cada uno de los nueve campos JSON generados por el Arduino (*temp_c, humidity_rh, pressure_pa, wind_kph, winddir_deg, rain_mm_h, rain_total_mm, light_v, battery_v*) llegara correctamente al dashboard web sin pérdida ni corrupción de datos, la comprobación se realizó comparando los valores mostrados en el dashboard con las lecturas crudas del puerto serial del Arduino, utilizando el comando *mosquitto_sub -h 192.168.40.104 -p 1883 -t "weather/station1"* para interceptar los mensajes MQTT en tiempo real y la interfaz Swagger UI del backend (*/docs*) para verificar la correcta persistencia en la base de datos SQLite [28, 36]. Los valores mostrados en el dashboard durante la sesión de prueba del 25 de mayo de 2025 (temperatura 24,7 °C, humedad 78 %, presión 1.007,6 hPa) coincidieron con las lecturas del serial, confirmando la integridad del flujo.

*b) Prueba de latencia de transmisión extremo a extremo:* La latencia de transmisión extremo a extremo se define como el tiempo transcurrido desde que el Arduino emite un mensaje JSON por el puerto serial hasta que el valor actualizado es visible en el navegador del usuario remoto, latencia que se compone de cuatro segmentos: (1) lectura y serialización en el Arduino (~1.000 ms por ciclo de muestreo); (2) bridge serial-MQTT en la Raspberry Pi (< 5 ms); (3) broker Mosquitto a backend FastAPI (< 2 ms en red local); y (4) WebSocket desde el backend al navegador a través de la red Starlink (latencia Starlink: 20–60 ms) [46, 39]. La latencia total observada entre la actualización del dato en el sensor y su aparición en el dashboard remoto fue inferior a 1,1 segundos, valor que resulta adecuado para el propósito de monitoreo agrícola del sistema, donde las decisiones operativas no requieren resoluciones temporales menores al segundo.

*c) Prueba de disponibilidad y continuidad del servicio:* La disponibilidad del sistema se evaluó mediante el monitoreo del indicador PING — RASPBERRY PI del dashboard, que aparece en estado *"ONLINE"* cuando el sistema está operativo, así, este indicador consulta periódicamente la disponibilidad del backend y refleja tanto la operatividad de la Raspberry Pi como la conectividad a través de la red Starlink, adicionalmente, el endpoint *GET /health* del backend retorna el estado del servicio en formato JSON, lo que permite la integración con sistemas externos de monitoreo de disponibilidad [36].

Los mecanismos de resiliencia implementados en el sistema contribuyen directamente a su disponibilidad, el hook *useWebSocket.js* del frontend implementa reconexión automática ante desconexiones del canal WebSocket causadas por interrupciones momentáneas de la red Starlink, el puente *main_mqtt.py* implementa un bucle de reintento de conexión al broker MQTT con espera de 5 segundos entre intentos y el broker Mosquitto mantiene persistencia de mensajes (*persistence true* en su configuración), lo que evita la pérdida de datos durante reinicios del servicio [28, 30].

*d) Rendimiento de la base de datos y la API:* El rendimiento de la capa de persistencia y consulta se evaluó mediante el endpoint *GET /api/history*, que consulta la tabla *weather_readings* de SQLite con filtros de tiempo y límites de puntos, la columna *timestamp* de la tabla está indexada, lo que garantiza tiempos de consulta eficientes incluso con volúmenes de datos acumulados a lo largo del tiempo y a una frecuencia de muestreo de una lectura por segundo, el sistema genera 86.400 registros por día en la base de datos donde el límite de 300 puntos por consulta implementado en el frontend (parámetro *limit=300*) garantiza tiempos de respuesta de la API inferiores a 200 ms para todos los rangos temporales disponibles (1h, 6h, 24h, 7d), manteniendo la fluidez del dashboard independientemente del volumen acumulado de datos históricos [41, 36].

**Tabla 13.** *Resumen de pruebas de desempeño y rendimiento del sistema meteorológico.*

| Prueba | Indicador evaluado | Resultado obtenido | Criterio de aceptación |
|---|---|---|---|
| Integridad del flujo de datos | Correspondencia serial ↔ dashboard | Coincidencia total de los 9 campos JSON | Sin pérdida ni corrupción de datos |
| Latencia extremo a extremo | Tiempo sensor → dashboard remoto | < 1,1 segundos (incluye ciclo Arduino) | Adecuado para monitoreo agrícola (resolución ≥ 1 s) |
| Latencia red Starlink | Round-trip time LAN → WAN → navegador | 20 – 60 ms (componente Starlink) | LEO: < 100 ms (vs. GEO: 600–800 ms) |
| Disponibilidad del servicio | Indicador ONLINE del dashboard | Estado ONLINE verificado en prueba | Reconexión automática ante fallos |
| Rendimiento de la API | Tiempo de respuesta /api/history | < 200 ms para todos los rangos (1h–7d) | Respuesta fluida en el dashboard |
| Volumen de datos generados | Tasa de inserción en SQLite | 86.400 registros/día (1 lectura/segundo) | Índice de timestamp garantiza consultas eficientes |
| Ancho de banda consumido | Payload JSON por mensaje | ~200 bytes/mensaje ≈ 1,6 Kbps | Starlink provee > 50 Mbps (margen > 30.000×) |

Tomado con base en pruebas funcionales del sistema y Del Portillo et al. [46].

### *6.4.6 Consideraciones de seguridad y escalabilidad*

En su configuración actual de prueba y despliegue inicial, el sistema opera con el broker Mosquitto en modo anónimo (*allow_anonymous true*), lo que simplifica la configuración en entornos de red local controlada, no obstante, para el despliegue definitivo en producción sobre la red Starlink, se recomienda la implementación de autenticación por credenciales en el broker MQTT, el uso de TLS/SSL para cifrar los canales WebSocket (WSS) y la restricción del

encabezado CORS en el backend a los dominios autorizados del frontend, en cumplimiento de las recomendaciones de seguridad para sistemas IoT expuestos a internet [30, 51].

En cuanto a la escalabilidad, la arquitectura implementada soporta la incorporación de nodos adicionales de sensores mediante la creación de nuevos tópicos MQTT (por ejemplo, *weather/station2*, *weather/station3*) sin modificaciones en la infraestructura del bróker, así, el backend puede ampliarse para suscribirse a múltiples tópicos y el frontend puede incorporar selectores de nodo en el dashboard, transformando el sistema de una estación única en una red de monitoreo distribuida que cubra diferentes pisos altitudinales y zonas de cultivo del municipio de Lebrija [30, 38].

### *6.4.7 Síntesis de la garantía de conectividad*

La garantía de conectividad se cumplió mediante la implementación y verificación de tres componentes interrelacionados, en primer lugar, la conectividad satelital Starlink garantizó el acceso a internet de banda ancha en la zona rural de Lebrija con latencias de 20–60 ms, superando estructuralmente las limitaciones de la conectividad terrestre disponible en la región, en segundo lugar, la integración transparente de Starlink en la topología de red del sistema permitió que los protocolos MQTT, HTTP REST y WebSocket operaran sin modificaciones sobre la conexión satelital, con un consumo de ancho de banda de ~1,6 Kbps por nodo, mínimo frente a la capacidad disponible, en tercer lugar, las pruebas de desempeño confirmaron la integridad del flujo de datos en los nueve campos del sistema, una latencia extremo a extremo inferior a 1,1 segundos, tiempos de respuesta de la API inferiores a 200 ms y mecanismos de resiliencia automática que garantizan la continuidad del servicio ante interrupciones momentáneas de la conectividad [46, 48, 28].

## 6.5 Análisis crítico, justificación de decisiones de diseño y validación experimental del sistema

### *6.5.1 Análisis crítico de los resultados obtenidos*

El sistema implementado alcanzó los objetivos funcionales planteados, pero un análisis crítico de los resultados permite identificar tanto los logros técnicos consolidados como las limitaciones que condicionan su alcance y que orientan las líneas de mejora futura, es decir, sobre la adquisición de datos, la arquitectura de la estación meteorológica basada en el Arduino Uno R3 con el SparkFun Weather Shield demostró una adquisición continua e ininterrumpida de las nueve variables meteorológicas a una frecuencia de un mensaje JSON por segundo, con integridad verificada en el 100 % de los campos durante las pruebas, sin embargo, el fotorresistor integrado en el Weather Shield entrega una aproximación de luminosidad en voltios (0–3,5 V) y no irradiancia calibrada en W/m², lo que limita su utilidad para cálculos de balance energético o correlaciones agronómicas de precisión, limitación reconocida y aceptable para el propósito de monitoreo comparativo del sistema, pero debe señalarse como una restricción técnica de la solución actual frente a estaciones meteorológicas de referencia que emplean piranómetros calibrados.

Sobre la conectividad Starlink. la latencia extremo a extremo medida fue inferior a 1,1 segundos, valor dominado por el ciclo de muestreo del Arduino (~1.000 ms) y no por la red satelital (~20–60 ms) lo cual implica que la red Starlink no es el cuello de botella del sistema, aumentar la frecuencia de muestreo del Arduino (por ejemplo, a 500 ms) reduciría la latencia percibida a menos de 600 ms sin requerir ningún cambio en la infraestructura de comunicaciones y el consumo de ancho de banda (~1,6 Kbps por nodo) es insignificante frente a la capacidad ofrecida por Starlink

(>50 Mbps), lo que significa que la solución escala a decenas de nodos sin saturar el enlace satelital.

Sobre el sistema web, el dashboard en React 18 con WebSocket cumplió con los requisitos de visualización en tiempo real y tiempos de respuesta de la API inferiores a 200 ms, pero existe una limitación identificada y es que la base de datos SQLite, aunque adecuada para la escala actual (86.400 registros/día por nodo), presenta restricciones de concurrencia de escritura ante múltiples nodos simultáneos, escenario para el cual sería necesario migrar a PostgreSQL o InfluxDB, adicionalmente, el broker Mosquitto opera actualmente con autenticación anónima (allow_anonymous true), configuración aceptable en red local controlada pero inadecuada para exposición directa a internet en producción.

Sobre el sistema de alimentación, la autonomía verificada de 42 horas supera el criterio de diseño de 40 horas. No obstante, la batería VRLA Magna MA 35-12 (35 Ah) cubre el 79 % de la capacidad teórica calculada (44,4 Ah), lo que significa que en condiciones de nubosidad prolongada superior a 36 horas el sistema podría aproximarse al límite de descarga del 50 % establecido para preservar la vida útil de la batería, margen aceptable para las condiciones climáticas de Lebrija, donde los episodios de nubosidad continua raramente superan las 36 horas [19], pero representaría un riesgo en zonas con mayor frecuencia de días nublados consecutivos.

### *6.5.2 Justificación de las decisiones de diseño*

Las principales decisiones de diseño del sistema respondieron a restricciones técnicas, económicas y operacionales concretas del contexto de implementación rural, por el lado de la arquitectura de dos microcontroladores (Arduino + Raspberry Pi) frente a solución de microcontrolador único, se evaluó la alternativa de integrar toda la lógica en la Raspberry Pi

eliminando el Arduino, opción descartada porque la Raspberry Pi no dispone de pines con capacidad de interrupción hardware de baja latencia equivalentes a los INT0 e INT1 del Arduino, lo que habría requerido librerías de polling en Python susceptibles de perder pulsos del anemómetro o el pluviómetro bajo carga del sistema operativo. El Arduino garantiza la captura de cada pulso mediante ISR dedicadas, independientemente de la carga de la Raspberry Pi, asegurando la integridad de las lecturas de viento y lluvia [25].

MQTT frente a HTTP REST para la transmisión interna de datos donde la transmisión de datos entre el Arduino y el backend podría haberse implementado directamente mediante peticiones HTTP POST desde la Raspberry Pi al backend, sin pasar por un broker. se optó por MQTT por tres razones: (1) el modelo publicador-suscriptor desacopla el productor de datos del consumidor, permitiendo que el backend se reinicie sin pérdida de mensajes gracias a la persistencia del broker; (2) el overhead de cabeceras MQTT es de 2 bytes mínimo, frente a los 200–900 bytes de HTTP; y (3) la arquitectura escala a múltiples nodos sin modificar el backend, únicamente añadiendo nuevos tópicos [30].

SQLite frente a bases de datos relacionales o de series temporales, así, SQLite fue seleccionado sobre PostgreSQL e InfluxDB por su operación sin servidor separado, eliminando un contenedor Docker adicional y reduciendo el consumo de memoria de la Raspberry Pi y a la escala actual de un nodo con 86.400 registros/día, SQLite con índice en timestamp mantiene tiempos de consulta inferiores a 200 ms para los rangos disponibles (1h–7d), entonces, la decisión implica aceptar la limitación de concurrencia de escritura, asumida como manejable en la arquitectura de nodo único actual [41].

Panel solar de 50 Wp frente al mínimo calculado de 35,6 Wp, su selección de 50 Wp, inmediatamente superior al mínimo teórico, responde a tres factores: (1) disponibilidad comercial

en el mercado colombiano, donde los módulos de potencias estándar son 30, 50, 80 y 100 Wp; (2) margen de seguridad del 40 % sobre el consumo diario calculado, que compensa las pérdidas reales por temperatura del panel (las celdas monocristalinas pierden aproximadamente 0,4 %/°C sobre 25 °C), envejecimiento y suciedad acumulada; y (3) costo marginal mínimo entre el módulo de 35 W y el de 50 W en el mercado local [33].

Docker Compose frente a instalación directa de servicios, donde los servicios de backend, frontend y broker podrían haberse instalado directamente en el sistema operativo de la Raspberry Pi, pero Docker Compose fue elegido porque garantiza la reproducibilidad del despliegue en cualquier instancia de Raspberry Pi con Docker instalado, aísla las dependencias de cada servicio evitando conflictos de versiones entre Python, Node y Mosquitto y permite el reinicio automático individual de cada servicio ante fallos mediante la política restart: unless-stopped [38].

### *6.5.3 Validación experimental del sistema*

La validación experimental del sistema se realizó mediante cuatro categorías de pruebas ejecutadas durante la instalación en la vereda Llanadas, municipio de Lebrija, Santander. La Prueba 1: Integridad del flujo de datos consistió en la comparación en tiempo real de los valores mostrados en el dashboard web con las lecturas transmitidas por el Arduino a través del tópico MQTT, verificadas mediante el comando mosquitto_sub -h 192.168.40.104 -p 1883 -t "weather/station1" y la interfaz Swagger UI del backend (/docs). Los nueve campos del mensaje JSON (temperatura, humedad, presión, velocidad y dirección del viento, tasa de lluvia, lluvia acumulada, nivel de luz y voltaje de la batería) coincidieron con los valores almacenados en la base de datos SQLite y con los visualizados en el dashboard, confirmando la integridad del flujo de información y la ausencia de pérdidas o alteraciones durante la transmisión. Adicionalmente,

las mediciones obtenidas durante la sesión de prueba del 25 de mayo de 2025 (24,7 °C, 78 % de humedad relativa y 1007,6 hPa) fueron comparadas con los rangos climáticos reportados para la zona por el IDEAM [19], observándose que se encontraban dentro de los valores esperados para las condiciones ambientales del sitio de implementación. Esta comparación permitió verificar la coherencia y plausibilidad de las mediciones obtenidas; sin embargo, no constituye una validación de la exactitud metrológica de los sensores, ya que no se realizó una comparación simultánea con una estación meteorológica o un instrumento de referencia calibrado. Por tanto, la validación de la exactitud de las mediciones se reconoce como una limitación del presente trabajo.

Prueba 2 de latencia extremo a extremo, donde la latencia entre la generación del dato en el Arduino y su aparición en el dashboard remoto a través de Starlink fue medida mediante observación directa sincronizada y estimación analítica de cada segmento del flujo y el resultado obtenido fue inferior a 1,1 segundos, descompuesto en: ciclo de muestreo del Arduino (~1.000 ms), bridge serial-MQTT en la Raspberry Pi (<5 ms), tránsito MQTT interno (<2 ms) y canal WebSocket sobre Starlink (20–60 ms). Este valor es adecuado para el propósito de monitoreo agrícola del sistema, donde las decisiones operativas no requieren resoluciones temporales menores al segundo [8].

Prueba 3 de rendimiento de la API REST en donde se ejecutaron consultas al endpoint GET /api/history para los cuatro rangos temporales disponibles (1h, 6h, 24h y 7d) con el límite de 300 puntos por consulta, encontrado que los tiempos de respuesta medidos fueron inferiores a 200 ms en todos los casos, verificando que el índice de timestamp en la tabla weather_readings garantiza consultas eficientes independientemente del volumen acumulado de datos históricos [41].

Prueba 4 de disponibilidad y resiliencia donde se verificó el comportamiento del sistema ante interrupciones simuladas de cada componente: (a) reinicio del contenedor Docker del backend

de recuperación automática en menos de 10 segundos con reconexión del WebSocket del frontend; (b) desconexión y reconexión del cable USB entre el Arduino y la Raspberry Pi el bridge main_mqtt.py detectó la pérdida del puerto serial y se reinició automáticamente mediante el servicio systemd; (c) interrupción momentánea de la conectividad Starlink el hook useWebSocket.js del frontend reconectó automáticamente al recuperarse la red, sin pérdida del historial de datos almacenado en SQLite. En todos los casos el sistema recuperó la operación normal de forma autónoma sin intervención manual.

## 7. Conclusiones

La caracterización agroclimática de la zona rural de Lebrija, Santander, permitió identificar y justificar las siete variables meteorológicas críticas para los sistemas productivos del área como la temperatura, humedad relativa, presión atmosférica, precipitación, velocidad y dirección del viento, y radiación solar, estableciendo sus rangos de referencia locales y su relación directa con los ciclos fenológicos del cultivo de aguacate y demás especies presentes en la zona, base técnica que determinó la selección y configuración de los sensores del sistema, así como los umbrales de monitoreo programados en la plataforma web, garantizando que la información recopilada por la estación sea pertinente para la toma de decisiones agronómicas en el contexto específico de implementación.

El sistema de adquisición de meteorológicos construida sobre el Arduino Uno R3 con el SparkFun Weather Shield DEV-13956 y el Weather Meter Kit SEN-15901, controlada por la Raspberry Pi 4, demostró una adquisición continua y robusta de datos a una frecuencia de un mensaje JSON por segundo, con integridad verificada en los nueve campos del flujo de datos, entonces, el sistema de alimentación autónoma como el panel solar fotovoltaico, controlador Steca

Solarix PRS 1010 y batería VRLA Magna MA 35-12 de 35 Ah, proporcionó una autonomía operativa superior a 40 horas sin generación solar, haciendo viable la operación continua de la estación en la zona rural de implementación sin dependencia de la red eléctrica convencional.

El sistema web desarrollado con FastAPI, SQLite y React 18, desplegado mediante Docker Compose sobre la Raspberry Pi, cumplió satisfactoriamente con los requisitos de visualización y gestión de datos meteorológicos en tiempo real, además, el canal WebSocket implementado garantiza la actualización del dashboard a una tasa de aproximadamente un dato por segundo, con tiempos de respuesta de la API REST inferiores a 200 ms para todos los rangos temporales disponibles y el dashboard permite al usuario visualizar simultáneamente los valores actuales de todas las variables del sistema, consultar series temporales históricas de hasta siete días y monitorear el estado de conectividad de la Raspberry Pi, constituyendo una herramienta funcional para el apoyo a la toma de decisiones agronómicas.

Por último, la implementación de la red satelital Starlink resolvió efectivamente la problemática de conectividad en la zona rural de Lebrija, proporcionando acceso a internet de banda ancha con latencias de 20 a 60 ms sobre un sistema que requiere apenas ~1,6 Kbps de ancho de banda por nodo, es decir, las pruebas de desempeño confirmaron una latencia extremo a extremo inferior a 1,1 segundos desde el sensor hasta el dashboard remoto, la integridad total del flujo de datos y la disponibilidad continua del sistema gracias a los mecanismos de resiliencia implementados, reconexión automática del WebSocket, reinicio de contenedores Docker y servicio systemd para el puente serial, por tanto, estos resultados validan la viabilidad técnica de Starlink como solución de conectividad para sistemas IoT de monitoreo agrícola en territorios con brecha digital rural.

## 8. Trabajo futuro

Los resultados obtenidos en este proyecto abren varias líneas de desarrollo que pueden fortalecer y ampliar el sistema en implementaciones futuras, en primer lugar, la arquitectura MQTT del sistema está diseñada para escalar a múltiples nodos de sensores mediante la creación de tópicos adicionales (weather/station2, weather/station3, etc.), lo que permitiría construir una red de monitoreo distribuida que cubra diferentes pisos altitudinales y zonas de cultivo del municipio de Lebrija, generando mapas de variabilidad climática local de alta resolución espacial.

En segundo lugar, la incorporación de modelos de predicción meteorológica de corto plazo basados en series temporales almacenadas en la base de datos SQLite mediante técnicas de aprendizaje automático como LSTM o redes neuronales recurrentes, permitiría generar alertas tempranas automatizadas ante eventos de lluvia intensa, helada o déficit hídrico, aumentando el valor agronómico del sistema.

En tercer lugar, el sistema actual opera con el broker MQTT en modo anónimo y sin cifrado TLS sobre la red Starlink, es decir, una línea de mejora necesaria para el despliegue definitivo en producción es la implementación de autenticación por credenciales en el broker, cifrado WSS en el canal WebSocket y restricción de CORS, en cumplimiento de los estándares de seguridad para sistemas IoT expuestos a internet.

Además, la integración de sensores de suelo de humedad volumétrica, temperatura subsuperficial y conductividad eléctrica, ampliaría el sistema desde el monitoreo meteorológico exclusivamente atmosférico hacia un sistema de monitoreo agroclimático integral, permitiendo correlacionar las variables del aire con el estado hídrico del suelo y optimizar de manera directa los protocolos de riego en los cultivos de la zona.

Por último, respecto a un sistema de monitoreo agroclimático integral, la integración de sensores de suelo humedad volumétrica, temperatura subsuperficial y conductividad eléctrica ampliaría el alcance del sistema desde el monitoreo exclusivamente atmosférico hacia un perfil agroclimático completo, en efecto, la correlación en tiempo real entre las variables del aire (precipitación, temperatura, radiación) y el estado hídrico del suelo permitiría calcular la evapotranspiración de referencia ($ET_0$) mediante la ecuación de Penman-Monteith (FAO, 2006), optimizando de manera directa los protocolos de riego en el cultivo de aguacate y demás especies de la zona de implementación.

Sensores como el VH400 o el EC-5 de Meter Group son compatibles con las interfaces ADC y $I^2C$ del Arduino Uno R3, lo que facilita su integración en el hardware existente sin requerir cambios estructurales en la arquitectura del sistema y el fortalecimiento de la seguridad para despliegue en producción, dado que el sistema actual opera con el broker Mosquitto en modo anónimo y sin cifrado TLS sobre la red Starlink, configuración aceptable en entornos de red local controlada pero insuficiente para exposición directa a internet, entonces, l despliegue definitivo en producción requiere la implementación de autenticación por credenciales en el broker (password_file en Mosquitto), cifrado TLS/SSL en el canal MQTT (puerto 8883) y WebSocket seguro (WSS), restricción del encabezado CORS en el backend FastAPI a los dominios autorizados del frontend, y la habilitación de HTTPS mediante un certificado Let's Encrypt gestionado por un proxy Nginx en el stack Docker. Estas medidas garantizarían el cumplimiento de los estándares de seguridad para sistemas IoT expuestos a internet establecidos por OWASP (2023) y el estándar IEC 62443 para sistemas de automatización industrial (Eclipse Foundation, 2023; OWASP, 2023).

## Referencias

[1] DANE, «dane,» 9 Agosto 2024. [En línea]. Available: https://www.dane.gov.co/index.php/estadisticas-por-tema/tecnologia-e-innovacion/tecnologias-de-la-informacion-y-las-comunicaciones-tic/indicadores-basicos-de-tic-en-hogares.

[2] Mintic, «Mintic,» 31 Jullio 2024. [En línea]. Available: https://www.mintic.gov.co/portal/inicio/Sala-de-prensa/Noticias/383816:Con-mapa-de-geolocalizacion-de-Santander-Ministerio-TIC-identificara-las-zonas-rurales-y-urbanas-que-requieren-soluciones-en-materia-de-conectividad.

[3] WMO, «Guide to Agricultural Meteorological Practices (GAMP)2010 Edition (WMO-No.134),» 2010. [En línea]. Available: https://community.wmo.int/site/knowledge-hub/programmes-and-initiatives/agricultural-meteorology/guide-agricultural-meteorological-practices-gamp2010-edition-wmo-no134.

[4] F. Ureña-Elizondo, «Utilización de estaciones meteorológicas automáticas como nueva alternativa para el registro y transmisión de datos,» *Revista Posgrado y Sociedad,* vol. 11, nº 1, pp. 22-49, 2011.

[5] J. Rosa-Cánovas, M. García-Valdecasas, E. Romero-Jiménez, P. Yeste-Donaire, S. Gámiz-Fortis, Y. Castro-Díez y M. Esteban-Parra, «Estudio de la habilidad de predicciones climáticas decenales para reproducir patrones de circulación atmosférica de gran escala,» de *XII Congreso Internacional de la Asociación Española de Climatología (AEC)*, Santiago de Compostela, 2015.

[6] R. D. F. Sanabria, agosto 2017. [En línea]. Available: https://repository.usta.edu.co/bitstream/handle/11634/10439/RafaelFerrer-2017.pdf?sequence=1.

[7] R. M. C. B. M. P. G. L. C. U. D. Díaz, 2010. [En línea]. Available: https://sedici.unlp.edu.ar/bitstream/handle/10915/101028/Documento_completo.pdf?sequence=1.

[8] OMM, «Guía de instrumentos y métodos de observación meteorológicos (Vol. I),» Organización Meteorológica Mundial, 2018. [En línea]. Available: https://library.wmo.int/doc_num.php?explnum_id=10179.

[9] A. Tobajas y A. López, «Diseño e implementación de una estación meteorológica con Raspberry Pi,» Universitat Oberta de Catalunya, Catalunya, 2016.

[10] Netatmo, «¿Qué es la presión atmosférica?,» 2023. [En línea]. Available: https://www.netatmo.com/es-es/weather-guide/what-is-the-atmospheric-pressure?srsltid=AfmBOoorTPTQ3XoFg3UMcSszZYlsmsva1Xygq1oOn7GLCf1L1D7WQbfq.

[11] Airthings, «¿Cómo obtener un monitor de humedad?,» 2025. [En línea]. Available: https://www.airthings.com/es/what-is-humidity#:~:text=La%20humedad%20es%20una%20medida,que%20el%20aire%20puede%20contener.

[12] WeatherCloud, «Blog weathercloud,» 23 Marzo 2020. [En línea]. Available: https://blog.weathercloud.net/announcements/2020/03/weathercloud-beta-9-5-is-here/.

[13] Sencinet, «Comunicación vía satélite: ¿Qué es, cuáles son sus ventajas y cómo funciona?,» 2026. [En línea]. Available: https://www.sencinet.com/es/blog/post/comunicacao-via-satelite-o-que-e-vantagens-e-como-funciona#to-top.

[14] C. Albarrán, «¿Qué es Internet por satélite?,» 2024. [En línea]. Available: https://www.redestelecom.es/comunicaciones/internet-por-satelite-todo-lo-que-necesitas-saber/.

[15] Spirnet, «Spirnet,» 2026. [En línea]. Available: https://spirnet.co/servicios/starlink-colombia.

[16] Zwetsloot, Rob, «Raspberry Pi Official Magazine,» 2019. [En línea]. Available: https://magazine.raspberrypi.com/articles/raspberry-pi-4-specs-benchmarks.

[17] Sparkfun, «Sigma Electronica,» 2020. [En línea]. Available: https://www.sigmaelectronica.net/wp-content/uploads/2020/10/DS-15901-Weather_Meter.pdf.

[18] sfe-SparkFro, «Github,» 27 Junio 2023. [En línea]. Available: https://github.com/sparkfun/SparkFun_Weather_Meter_Kit_Arduino_Library/tree/v1.0.0.

[19] IDEAM, «Atlas climatológico de Colombia,» Instituto de Hidrología, Meteorología y Estudios Ambientales, 2021. [En línea]. Available: https://www.ideam.gov.co/web/tiempo-y-clima/atlas-climatologico.

[20] MADR, «Estadísticas del sector agropecuario de Santander,» Ministerio de Agricultura y Desarrollo Rural & Agronet, 2022. [En línea]. Available: https://www.agronet.gov.co/estadistica/Paginas/home.aspx.

[21] Google Earth, 15 Marzo 2024. [En línea]. Available: https://earth.google.com/web/.

[22] H. E. Balaguera-López y C. Martínez, «Fisiología del aguacate y condiciones agroclimáticas para su cultivo en Colombia,» *Agronomía Colombiana,* vol. 37, nº 2, p. 112–124, 2019.

[23] SparkFun Electronics, «SparkFun Weather Shield (DEV-13956) — Hookup guide.,» 2021. [En línea]. Available: https://learn.sparkfun.com/tutorials/weather-shield-hookup-guide.

[24] Raspberry Pi, «Raspberry Pi 4 Model B — Datasheet,» Raspberry Pi Ltd, 2023.

[25] Arduino , «Arduino Uno R3 — Datasheet and reference manual. Arduino.,» Arduino AG, 2023. [En línea]. Available: https://docs.arduino.cc/hardware/uno-rev3.

[26] S. Monk, Programming the Raspberry Pi: Getting started with Python (2.ª ed.)., New York: McGraw-Hill Education, 2016.

[27] J. Blum, Exploring Raspberry Pi: Interfacing to the real world with embedded Linux, New York: Wiley, 2014.

[28] Eclipse Foundation, «Eclipse Mosquitto: An open source MQTT broker,» 2023. [En línea]. Available: https://mosquitto.org.

[29] R. T. Fielding, «Architectural styles and the design of network-based software architectures [Tesis doctoral],» University of California, Irvine, 2000.

[30] A. Banks y R. Gupta, «MQTT Version 5.0 — OASIS Standard,» OASIS, 2019. [En línea]. Available: https://docs.oasis-open.org/mqtt/mqtt/v5.0/mqtt-v5.0.html.

[31] Z. Shelby, K. Hartke y C. Bormann, «The Constrained Application Protocol (CoAP) — RFC 7252,» IETF, 2014.

[32] OASIS, «Advanced Message Queuing Protocol (AMQP) Version 1.0.,» AMQP, 2012.

[33] E. Lorenzo, Ingeniería fotovoltaica (Vol. 1: Radiación solar), Progensa, 2014.

[34] Steca Elektronik, «Steca Solarix PRS 1010 — Solar charge controller user manual,» Steca, 2020. [En línea]. Available: https://www.steca.com.

[35] IEEE, «IEEE 485-2010: Recommended practice for sizing lead-acid batteries for stationary applications,» IEEE Standards Association, 2018.

[36] FastAPI Software Foundation, «High performance, easy to learn, fast to code, ready for production,» FastAPI , 2024. [En línea]. Available: https://fastapi.tiangolo.com.

[37] Facebook, «React — A JavaScript library for building user interfaces (v18),» 2023. [En línea]. Available: https://legacy.reactjs.org/.

[38] D. Merkel, «Docker: Lightweight Linux containers for consistent development and deployment,» *Linux Journal,* vol. 239, nº 2, pp. 1-12, 2014.

[39] I. Fette y A. Melnikov, «The WebSocket protocol (RFC 6455),» IETF, 2011.

[40] D. Beazley y B. K. Jones, Python cookbook (3.ª ed.), O'Reilly Media, 2013.

[41] M. Bayer, «The database toolkit for Python,» SQLAlchemy, 2024. [En línea]. Available: https://www.sqlalchemy.org.

[42] I. Fette y A. Melnikov, «The WebSocket protocol (RFC 6455),» IETF, 2011.

[43] Evan You & Vite contributors, «Vite — Next generation frontend tooling.,» Evan You & Vite contributors, 2024. [En línea]. Available: https://vitejs.dev.

[44] Recharts Team, «Recharts — Redefined chart library built with React and D3,» Recharts Team, 2024. [En línea]. Available: https://recharts.org.

[45] Comisión de Regulación de Comunicaciones, «Reporte de industria TIC — Conectividad en zonas rurales de Colombia,» CRC, 2023. [En línea]. Available: https://www.crcom.gov.co.

[46] I. Del Portillo, B. G. Cameron y E. F. Crawley, «A technical comparison of three LEO satellite network constellations: Telesat, SpaceX, and Amazon,» *Acta Astronautica,* nº 159, p. 123–135, 2019.

[47] D. Bhattacherjee y A. Singla, «Network topology design at 27,000 km/hour,» *Proceedings of CoNEXT,* p. 341–354, 2019.

[48] SpaceX, «Starlink technical specifications and residential service documentation,» SpaceX, 2024. [En línea]. Available: https://www.starlink.com/legal/documents/DOC-1002-53896-84.

[49] B. Forouzan, Data communications and networking (5.a ed.), McGraw-Hill Education, 2021.

[50] Red Hat, «Systemd system and service manager — Reference documentation.,» 2023. [En línea]. Available: https://www.freedesktop.org/software/systemd/man/.

[51] OWASP, «OWASP IoT security guidance — Top 10 IoT vulnerabilities. Open Web Application Security Project,» 2023. [En línea]. Available: https://owasp.org/www-project-internet-of-things/.

[52] W. M. Organization, «Guide to Agricultural Meteorological Practice,» WMO-No. 134 ed., Organization, World Meteorological, 2010.

[53] E. I. Fenix, «Via Industrial,» [En línea]. Available: https://www.viaindustrial.com/estacion-meteorologica-inteligente-con-monitoreo-remoto-de-wifi-y-alertas-ws-2902a-ambient-weather/pp/P239639/.

[54] BioWeb, «Estación Meteorológica Inteligente Ambient Weather WS-2902,» 2025. [En línea]. Available: https://colombia.bioweb.co/products/estacion-meteorologica-inteligente-ambient-weather-ws-2902.