JOSÉ LUIS ORTIZ – FERNANDO XAVIER VALENCIA – CARLOS PATRICIO BOSMEDIANO – JESSICA XIMENA TORRES – ALEX DANILO BASTIDAS – LETTY MAGDALENA GARCÍA

ALUMNI EDITORA
2024

PRIMERA EDICIÓN

INTERNET DE LAS COSAS: MANUAL DE PRÁCTICAS PARA EL AULA Y EL LABORATORIO

INSTITUTO SUPERIOR TECNOLÓGICO
17 DE JULIO

Internet de las Cosas: Manual de Prácticas para el Aula y el Laboratorio

Autores

José Luis Ortiz Arciniega

- Ingeniero en Electrónica y Redes de Comunicación
- Máster Universitario en Ingeniería Electrónica

https://orcid.org/0000-0003-3707-5252

jlortiz@ist17dejulio.edu.ec

Fernando Xavier Valencia Barahona

- Tecnólogo en Sistemas
- Ingeniero en Electrónica y Redes de Comunicación

https://orcid.org/0009-0006-5889-4918

fvalencia@ ist17dejulio.edu.ec

Carlos Patricio Bosmediano Cárdenas

- Ingeniero en Electrónica y Redes de Comunicación
- Magister en Telecomunicaciones

https://orcid.org/0000-0003-3707-5252

cbosmediano@ist17dejulio.edu.ec

Internet de las Cosas: Manual de Prácticas para el Aula y el Laboratorio

Autores

Jessica Ximena Torres Reyes

- Ingeniera en Electrónica y Redes de Comunicación
- Máster en Gestión y Administración de Tecnologías de la Información

https://orcid.org/0009-0007-7601-7588

jtorres@ist17dejulio.edu.ec

Alex Danilo Bastidas Jácome

- Ingeniero en Mecatrónica
- Magister en Mecatrónica mención en Procesos Industriales

https://orcid.org/0009-0008-0705-3697

dbastidas@ist17dejulio.edu.ec

Letty Magdalena García López

- Licenciado en Comunicación Social

https://orcid.org/0009-0007-2769-1453

lgarcia@ ist17dejulio.edu.ec

Internet de las Cosas: Manual de Prácticas para el Aula y el Laboratorio

Catalogación Bibliográfica

Autores	• José Luis Ortiz Arciniega • Fernando Xavier Valencia Barahona • Carlos Patricio Bosmediano Cárdenas • Jessica Ximena Torres Reyes • Alex Danilo Bastidas Jácome • Letty Magdalena García López
Título	Internet de las Cosas: Manual de Prácticas para el Aula y el Laboratorio
Descriptores	Internet de las cosas (IoT), Software, Hardware, Redes de comunicación, Innovación tecnológica
Dewey	621.38
Thema	UBW
Publicación	Enero 2025
Edición	Primera
ISBN	978-9942-7307-4-9
DOI	https://doi.org/10.70625/alumned/7
Editorial	Alumni Editora
Pais - Ciudad	Ecuador - Atuntaqui
Formato	Adobe Acrobat Reader
Páginas	135

Cámara Ecuatoriana del Libro

Internet de las Cosas: Manual de Prácticas para el Aula y el Laboratorio

ALUMNI EDITORA

Peer Review Process

Revisión de Pares

Este libro ha sido evaluado mediante un proceso de revisión por pares externos bajo el formato de doble ciego. En consecuencia, la investigación presentada en esta obra cuenta con el respaldo de expertos en la materia, quienes han emitido un juicio imparcial basado en criterios científicos, garantizando así la solidez académica del contenido.

Peer Review

This book has undergone a peer review process by external academics using a double-blind system. Consequently, the research presented in this work has the endorsement of subject matter experts, who have provided an impartial assessment based on scientific criteria, ensuring the academic rigor of the content.

Declaración del Editor

Declaración del Editor

Alumni Editora declara para todos los efectos legales, que:

Esta publicación implica únicamente una cesión temporal de los derechos de autor y de publicación, sin que ello constituya responsabilidad solidaria en la creación de los manuscritos publicados en conformidad con la Ley de Propiedad Intelectual y las normativas legales aplicables.

Autoriza y fomenta que los autores firmen acuerdos con repositorios institucionales con el fin exclusivo de difundir la obra, siempre que se reconozca adecuadamente la autoría y la edición, y que no existan fines comerciales involucrados.

Todos los libros electrónicos publicados son de acceso abierto y, por lo tanto, no se venden en el sitio web de Alumni Editora, ni en plataformas asociadas, de comercio electrónico u otros medios virtuales o físicos, eximiéndose de la transferencia de derechos de autor a los autores.

Todos los miembros del consejo editorial cuentan con el grado académico de cuarto nivel y están vinculados a instituciones de educación superior, conforme a las recomendaciones de las entidades de evaluación académica nacionales e internacionales para la obtención de estándares de calidad editorial.

Alumni Editora no transfiere, comercializa, ni autoriza el uso de los nombres, correos electrónicos u otros datos personales de los autores para fines distintos a la difusión de esta obra.

Declaración del Autor

Declaración del Autor

El autor de la obra declara: 1. no poseer ningún interés comercial que pueda representar un conflicto de interés en relación con el presente documento publicado; 2. Asegura haber participado activamente en la elaboración del manuscrito, específicamente en la concepción del estudio, la obtención de datos y/o su análisis e interpretación; la redacción o revisión del documento para garantizar su relevancia intelectual y la aprobación final del manuscrito antes de su envío; 3. Certifica que el contenido publicado está libre de datos o resultados fraudulentos; 4. Confirma que todas las citas y referencias de datos e interpretaciones de investigaciones previas son correctas; 5. Reconoce haber declarado todas las fuentes de financiamiento recibidas para la investigación; 6. Autoriza la publicación de la obra, que incluye su inclusión en catálogos, asignación de ISBN, DOI, otros índices, diseño visual, portada, maquetación interior, y su posterior difusión según lo dispuesto por Alumni Editora.

Prólogo

El avance de las tecnologías de la información y comunicación ha dado paso a una revolución tecnológica sin precedentes, caracterizada por la interconexión de dispositivos inteligentes que facilitan la interacción con el entorno de manera eficiente y autónoma. En este contexto, el Internet de las Cosas (IoT, por sus siglas en inglés) emerge como una de las tecnologías más disruptivas de nuestra era, con aplicaciones que abarcan desde la gestión de hogares inteligentes hasta la optimización de procesos industriales. Este libro se presenta como una guía práctica diseñada para estudiantes y profesionales que deseen explorar, aprender y aplicar los fundamentos del IoT, utilizando herramientas modernas y accesibles para resolver problemas del mundo real.

El contenido de esta obra se ha estructurado cuidadosamente para proporcionar una formación integral que combina teoría y práctica, orientada al diseño e implementación de sistemas IoT. Cada capítulo aborda un conjunto específico de habilidades y conocimientos, desde la configuración de placas de desarrollo como la ESP32 y la integración de sensores y actuadores, hasta el uso de protocolos de comunicación, bases de datos y plataformas de visualización. Además, se han diseñado prácticas aplicadas que reflejan situaciones reales, tales como la gestión de edificios inteligentes, el control remoto de dispositivos, y la implementación de alertas en tiempo real, permitiendo que los estudiantes adquieran competencias prácticas alineadas con las necesidades del sector tecnológico.

Una de las características distintivas de este libro es su enfoque en el uso de software ampliamente adoptado en el ámbito profesional, como Arduino IDE para la programación de dispositivos, Node-RED para la gestión de flujos, InfluxDB como base de datos, y Grafana para la creación de dashboards personalizados. Asimismo, se incluye la integración de plataformas de mensajería como Telegram para el envío de alertas y la gestión remota de dispositivos, promoviendo la adopción de soluciones tecnológicas avanzadas y de fácil implementación. De esta manera, los lectores no solo desarrollan habilidades técnicas, sino que también adquieren una comprensión profunda de las aplicaciones del IoT en entornos profesionales.

Este libro no solo busca formar técnicamente a los estudiantes, sino también inspirarlos a imaginar y construir soluciones innovadoras que contribuyan al desarrollo sostenible y a la mejora de la calidad de vida en sus comunidades. Al

explorar los conceptos, herramientas y casos prácticos aquí presentados, los lectores estarán preparados para enfrentar los retos tecnológicos del futuro, desarrollar proyectos IoT con impacto real y posicionarse como profesionales altamente competitivos en un mercado laboral dinámico y en constante evolución.

Los Autores

Tabla de Contenido

Tabla de contenido

Tabla de Contenido

Introducción

Este libro tiene como objetivo principal proporcionar una herramienta de aprendizaje práctico que permita a los estudiantes desarrollar habilidades técnicas y competencias profesionales en el ámbito del Internet de las Cosas (IoT). A través de una serie de guías prácticas cuidadosamente diseñadas, los estudiantes podrán:

- **Comprender los fundamentos de IoT**: explorar conceptos clave como sensores, actuadores, protocolos de comunicación, y plataformas de desarrollo para IoT.
- **Aplicar la teoría en entornos reales**: las prácticas están diseñadas para conectar el conocimiento teórico con escenarios reales, brindando una experiencia de aprendizaje significativa.
- **Desarrollar soluciones IoT**: Los estudiantes aprenderán a implementar proyectos desde la configuración de hardware y software hasta la gestión y análisis de datos en tiempo real.
- **Fortalecer el pensamiento crítico y la resolución de problemas**: cada práctica incluye desafíos técnicos que impulsan a los estudiantes a tomar decisiones fundamentadas y a enfrentar problemas reales de manera creativa.
- **Prepararse para la vida profesional**: el enfoque práctico del libro garantiza que los estudiantes estén preparados para integrar IoT en entornos laborales, contribuyendo a la solución de problemas tecnológicos y sociales.
- **Fomentar la innovación tecnológica**: a través de ejercicios complementarios, el libro busca inspirar a los estudiantes a desarrollar aplicaciones innovadoras en áreas como la domótica, la agricultura inteligente, las ciudades inteligentes, y la salud.

El propósito de este libro no se limita únicamente al desarrollo de habilidades técnicas. También busca promover una comprensión holística de cómo la tecnología IoT puede ser un motor de transformación en diferentes contextos, fomentando la integración de conocimientos y el aprendizaje basado en proyectos.

Este manual está alineado con las necesidades de la industria y las tendencias tecnológicas actuales, lo que garantiza que el contenido sea pertinente y de alto valor para los estudiantes y futuros profesionales en el campo de Redes y Telecomunicaciones, ya que se refuerza la utilización de software libre y placas de prototipado de hardware libre.

El desarrollo de proyectos prácticos en IoT es esencial para consolidar el aprendizaje, ya que permite a los estudiantes experimentar con tecnologías emergentes que conectan dispositivos físicos a la nube, desarrollar habilidades para configurar sensores, actuadores y redes de comunicación, comprender el impacto de IoT en áreas como la automatización, el monitoreo remoto y la gestión inteligente de recursos.

CAPÍTULO I

Introducción y antecedentes

ALUMNI EDITORA

Introducción

El Internet de las Cosas (IoT) es una de las tecnologías más disruptivas de la actualidad, transformando la forma en que los dispositivos interactúan entre sí y con el mundo que nos rodea. Este concepto se refiere a la conexión de objetos físicos a internet, permitiendo que recopilen, compartan y analicen datos en tiempo real para optimizar procesos, mejorar la toma de decisiones y facilitar la vida cotidiana.

La creciente implementación de IoT abarca sectores tan diversos como la industria, la salud, la agricultura, el transporte, y las ciudades inteligentes, desde dispositivos domésticos conectados hasta soluciones de monitoreo en tiempo real para sistemas críticos, IoT ofrece oportunidades infinitas para resolver problemas y mejorar la eficiencia en diferentes contextos [1].

El libro se enfoca en proporcionar una guía práctica para entender y trabajar con IoT desde una perspectiva aplicada. Las prácticas incluidas están diseñadas para facilitar el aprendizaje de los componentes y tecnologías fundamentales que conforman un sistema IoT. A través de proyectos concretos, los estudiantes aprenderán a:

- Configurar y programar dispositivos como la ESP32, una placa microcontroladora versátil y potente de hardware libre.
- Utilizar herramientas como Arduino IDE para programación, Node-RED para la gestión de flujos de datos, InfluxDB y Grafana para almacenamiento y visualización de datos en tiempo real, todas herramientas de software libre.
- Implementar el protocolo de comunicación MQTT, esencial para la transmisión eficiente de datos entre dispositivos IoT.
- Configurar bots en Telegram para generar alertas y controlar actuadores mediante comandos desde la aplicación.

Este enfoque práctico tiene como objetivo no solo desarrollar habilidades técnicas, sino también fortalecer el pensamiento crítico y la capacidad para resolver problemas. Además, se busca que los estudiantes comprendan cómo estas tecnologías pueden integrarse en aplicaciones reales que impacten de manera positiva en la sociedad.

En un mundo donde IoT está cada vez más presente, este manual busca ser una guía que conecte la teoría con la práctica, preparando a los futuros

profesionales para afrontar los desafíos y aprovechar las oportunidades de esta apasionante tecnología.

Herramientas Tecnológicas

Para el diseño, configuración e implementación de sistemas IoT, es fundamental utilizar herramientas tecnológicas que integren de manera eficiente el software y el hardware, permitiendo la creación de aplicaciones interactivas y funcionales [2]. En un entorno educativo, se priorizan herramientas de código abierto, lo cual elimina la necesidad de que los estudiantes inviertan en licencias o software propietario de alto costo. A pesar de emplear tecnologías libres, la comunidad de desarrolladores ha logrado optimizar estas herramientas, otorgándoles la capacidad y el rendimiento necesarios para cumplir con los objetivos esenciales del IoT.

Entre estas herramientas, destacan las placas de prototipado de hardware como Arduino, ESP32, ESP8266, Raspberry Pi, BeagleBone, entre otras, que facilitan el desarrollo de aplicaciones IoT a bajo costo, especialmente en entornos de programación de código abierto [3]. Aunque las placas Arduino ofrecen numerosas ventajas, en este manual se opta por utilizar la placa ESP32, debido a que integra conectividad Wi-Fi, que facilita la conexión con Internet, y su compatibilidad con el entorno de desarrollo integrado (IDE) de Arduino.

Por otro lado, diversas herramientas informáticas, como Node-RED, InfluxDB y Grafana, juegan un papel clave en el control, gestión, almacenamiento y visualización de aplicaciones IoT, estas herramientas cuentan con bibliotecas y códigos previamente desarrollados que permiten su integración en sistemas IoT de manera rápida y eficiente [4]. A continuación, se presenta una descripción detallada de las herramientas utilizadas en este libro.

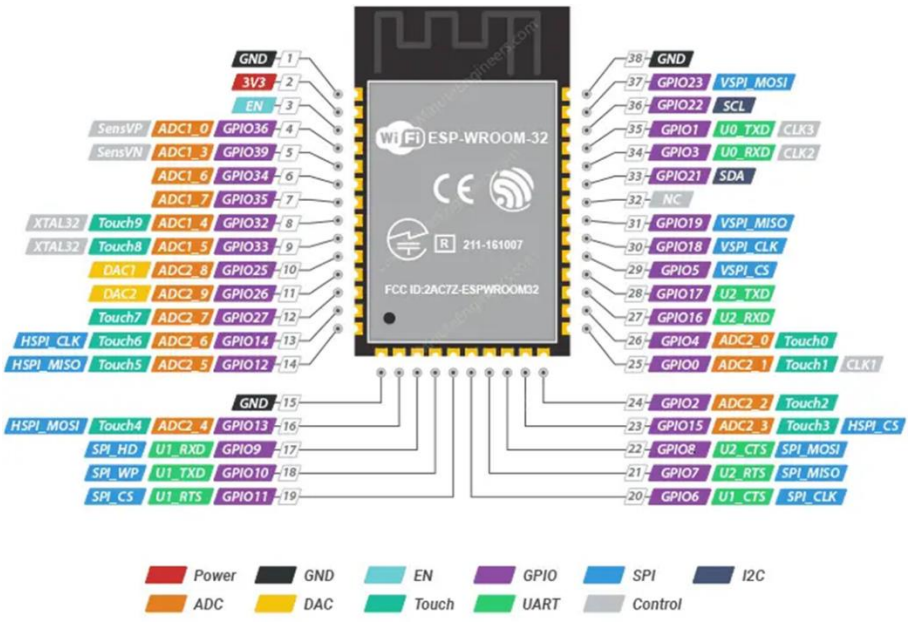
ESP32

Según la página web oficial del fabricante, ESP32 es una familia de chips SoC de bajo costo y bajo consumo energético, que integra tecnología Wi-Fi y Bluetooth en modo dual, este chip utiliza un microprocesador Tensilica Xtensa LX6, disponible en versiones de un solo núcleo o doble núcleo, y está equipado con interruptores de antena, balun de radiofrecuencia, amplificador de potencia,

amplificador receptor de bajo ruido, filtros y módulos de gestión de energía; desarrollado por Espressif Systems y fabricado por TSMC utilizando su proceso de 40 nm, el ESP32 es el sucesor del SoC ESP8266 [5].

Para las prácticas desarrolladas en este libro, se utiliza una placa de prototipado de hardware ESP32-WROOM-32 que según su ficha técnica [6] (datasheet) es un módulo MCU potente y genérico que combina Wi-Fi, Bluetooth® y Bluetooth LE, orientado a una amplia variedad de aplicaciones, que van desde redes de sensores de bajo consumo hasta tareas más exigentes, como la codificación de voz, transmisión de música y decodificación de MP3. En el núcleo de este módulo se encuentra el chip ESP32-D0WDQ6. El chip integrado está diseñado para ser escalable y adaptable. Cuenta con dos núcleos de CPU que pueden ser controlados individualmente, y la frecuencia de reloj de la CPU se puede ajustar de 80 MHz a 240 MHz. El chip también posee un coprocesador de bajo consumo que puede utilizarse en lugar de la CPU para ahorrar energía mientras realiza tareas que no requieren gran poder de procesamiento, como la supervisión de periféricos. El ESP32- WROOM-32 integra una amplia gama de periféricos, que incluyen sensores táctiles capacitivos, interfaz de tarjeta SD, Ethernet, SPI de alta velocidad, UART, I2S e I2C. En la figura 1 se detalla la configuración de pines de este módulo.

Figura 1
Definición de Pines (Pinout) ESP32-WROOM-32



Como se puede observar en el Pinout del ESP32, el módulo cuenta con 38 pines de los cuales se tiene suficientes entradas y salidas acondicionadas para señales digitales y analógicas que permiten controlar sensores y actuadores conforme a las necesidades del diseño IoT.

Arduino IDE

El entorno de desarrollo integrado (IDE) de Arduino es una aplicación multiplataforma compatible con Windows, macOS y Linux, desarrollada en el lenguaje de programación Java, este entorno permite escribir y cargar programas en placas compatibles con Arduino y, mediante núcleos de terceros, también puede utilizarse con placas de desarrollo de otros fabricantes, como en este caso los módulos ESP32 [7].

Soporta los lenguajes de programación C y C++, aplicando reglas específicas para estructurar el código. El IDE incluye una biblioteca de software derivada del proyecto Wiring, que facilita el uso de procedimientos comunes de entrada y salida (E/S). Para desarrollar programas, el usuario solo necesita definir dos funciones principales: una para iniciar el boceto y otra para el ciclo principal del programa.

El IDE también utiliza el programa avrdude, que convierte el código ejecutable en un archivo de texto codificado en formato hexadecimal, este archivo es cargado en la placa Arduino mediante un gestor de carga integrado en el firmware de la placa, también se utiliza para simulaciones en programas dedicados como Tinkercad y Proteus Design Suite [7]. Esto permite comprobar el correcto funcionamiento del código en ambientes simulados antes de su implementación en la placa.

Instalación y Configuración

Al tratarse de software libre, la descarga es gratuita desde su página web oficial www.arduino.cc y su instalación no requiere de licencias ni llaves de activación, se puede realizar una donación voluntaria al equipo desarrollador en caso de considerarlo. En la página web se muestran varias opciones de descarga siendo la última versión la 2.3.4. Además, se puede acceder a versiones anteriores en caso de que la computadora personal del estudiante no sea compatible con la última versión. Para evitar cualquier inconveniente se recomienda descargar la

versión 1.8.19, ya que es la más estable para cualquier sistema operativo y no requiere mucha carga computacional.

Para descargar esta versión, nos dirigimos a la pestaña de software en la página oficial de Arduino y bajamos hasta encontrar la opción Legacy IDE (1.8.X). En las opciones de descarga, seleccionamos el sistema operativo de la computadora personal y hacemos clic para ejecutar la descarga.

Una vez descargado el archivo, lo descomprimimos de ser el caso, y ejecutamos el instalador con permisos de administrador, seguimos todos los pasos y al finalizar tendremos el IDE instalado y listo para su configuración, debido a que no incluye soporte nativo para las placas ESP32. Para lo cual es necesario realizar los siguientes pasos:

Paso 1. Agregar la URL del gestor de tarjetas:

- Abre el IDE de Arduino.
- Ve a Archivo → Preferencias (en macOS, está en Arduino → Preferencias).
- En la sección Gestor de URLs de tarjetas adicionales, añade la siguiente URL:https://espressif.github.io/arduino-esp32/package_esp32_index.json

Paso 2: Instalar el paquete de ESP32

- Ve a Herramientas → Placa → Gestor de Tarjetas.
- En el buscador, escribe esp32.
- Selecciona el paquete esp32 by Espressif Systems y haz clic en Instalar.
- Espera a que finalice la instalación.

Una vez realizados estos pasos, ya podremos conectar cualquier placa ESP32 y en el IDE de Arduino seleccionar el modelo y el puerto al que se conectó para poder cargar los códigos desarrollados para las diferentes aplicaciones IoT.

Node-RED

Node-RED es una plataforma de desarrollo basada en flujos, diseñada para simplificar la integración de hardware, APIs y servicios en línea en aplicaciones de IoT [8]. Inicialmente desarrollada por IBM, Node-RED ha ganado una amplia adopción gracias a su enfoque visual, que permite a los desarrolladores construir aplicaciones complejas mediante una interfaz gráfica intuitiva, ya que en lugar de escribir código extensivo, los usuarios pueden conectar bloques o nodos que

representan funciones, entradas y salidas, formando flujos que definen el comportamiento de sus aplicaciones [8].

El editor de Node-RED es accesible a través de un navegador web, lo que lo hace multiplataforma y fácil de usar desde cualquier dispositivo conectado. Los flujos creados son almacenados en formato JSON, lo que permite compartirlos o reutilizarlos fácilmente. Además, la plataforma está basada en Node.js, lo que garantiza un alto rendimiento y la posibilidad de ejecutar aplicaciones tanto en servidores robustos como en dispositivos de bajo consumo, como placas Raspberry Pi o en este caso el módulo ESP32-WROOM-32.

Una de las principales ventajas de Node-RED es su capacidad para integrar diversos protocolos y tecnologías, como MQTT, HTTP, WebSocket, y los estudiantes pueden conectar nodos predefinidos que ya incluyen funcionalidades comunes, como la conexión con servicios en la nube, el procesamiento de datos o la gestión de dispositivos físicos, haciendo que no sean necesarios conocimientos avanzados en programación o JavaScript para implementar flujos de control y automatización complejos [9].

Este programa de código abierto tiene la capacidad de gestionar datos en tiempo real, lo cual es clave en el desarrollo de aplicaciones de monitoreo y control. Por ejemplo, se puede utilizar para capturar información de sensores, procesarla, almacenarla en bases de datos y luego visualizarla en dashboards personalizados.

Instalación y Configuración como Servicio

Para el desarrollo de proyectos IoT, se recomienda instalar Node-RED en dispositivos dedicados como Raspberry Pi, ya que ofrecen un bajo costo, consumo energético reducido y la flexibilidad necesaria para ejecutar aplicaciones de manera continua. Sin embargo, para las prácticas descritas en este manual, se optará por instalar Node-RED localmente en el computador personal del estudiante o docente. Esta configuración permitirá un entorno de desarrollo accesible y cómodo para probar los flujos sin necesidad de hardware adicional. Además, Node-RED se configurará como un servicio en el sistema operativo del computador, asegurando que se inicie automáticamente y esté disponible en segundo plano para garantizar una experiencia fluida durante las prácticas. Para lo cual se deben seguir los siguientes pasos:

Paso 1: Requisitos previos:

- Tener instalado Node.js (versión 14 o superior recomendada). Se descarga desde la página oficial: https://nodejs.org/.
- Disponer de acceso a la terminal o consola de comandos en el sistema operativo (CMD, PowerShell, Terminal en macOS o Linux).

Paso 2: Instalación de Node-RED

- Una vez que Node.js esté instalado, abre la terminal de tu computador.
- Ejecuta el siguiente comando para instalar Node-RED de manera global: npm install -g --unsafe-perm node-red
- Este comando descarga e instala Node-RED utilizando el gestor de paquetes de Node.js (npm).
- Una vez completada la instalación, verifica que Node-RED se instaló correctamente ejecutando: node-red –version
- Esto debería mostrar la versión instalada de Node-RED.
- Puedes iniciar Node-RED escribiendo simplemente: node-red
- Esto abrirá Node-RED en tu navegador web por defecto en la dirección http://localhost:1880

Paso 5: Descargar NSSM (Non-Sucking Service Manager):

- NSSM es una herramienta que permite crear servicios en Windows para aplicaciones que no tienen esta funcionalidad por defecto.
- Descárgalo desde https://nssm.cc/download asegurándose de elegir la última versión y descomprímelo.

Paso 4: Configuración como servicio

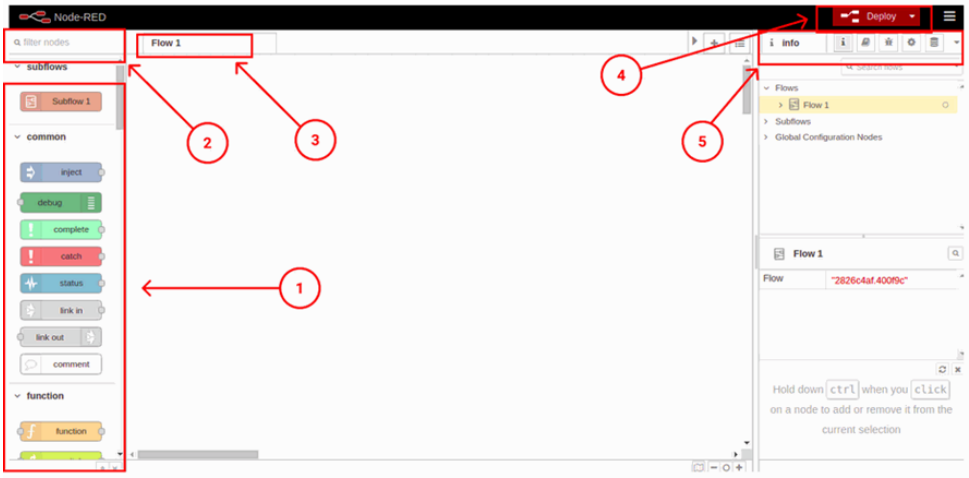
- Copiar el archivo .exe de nssm dentro de la carpeta de instalación de Node-RED, la cual se encuentra en C:\Users\user\AppData\Roaming\npm
- Abrir una ventana de cmd desde esa ubicación y ejecutar el comando: nssm install Node-RED.
- Se abrirá una ventana donde debes configurar:
    - Path: C:\Users\Usuario\AppData\Roaming\npm\node-red.cmd
    - Startup Dir: C:\Users\Usuario\.node-red\
    - Arguments: --settings "C:\Users\Usuario\.node-red\settings.js"
- Haz clic en "Install Service".

Paso 4: Iniciar el servicio:

- Abre el Administrador de Servicios de Windows (services.msc).

- Busca el servicio llamado "Node-RED", haz clic derecho y selecciona Iniciar.
- Configura el tipo de inicio como Automático para que se inicie al encender la computadora.
- Una vez configurado como servicio, Node-RED estará disponible en http://localhost:1880 automáticamente cada vez se encienda el computador.
- La interfaz de Node-RED tiene las siguientes partes:

Figura 2
Partes de la interfaz de Node-RED



1. Nodos de programación: los nodos son los bloques que se utilizan para programar. Para ello, se tendrá que arrastrar el nodo deseado al centro de la interfaz en el “Flow” (flujo) deseado y se configurará dependiendo la aplicación.
2. Buscador de nodos: el buscador facilita el encontrar el nodo deseado. Eso viene bien si en un futuro se instalan muchos paquetes y por ende se tiene una amplia lista de nodos.
3. Pestaña de “Flows”: aquí se puede visualizar en todo momento en que flujo se está trabajando, y dando dos clics encima de la pestaña se entra a la ventana de configuración de este.
4. Botón “Deploy”. Una vez programado o realizados los cambios deseados se pulsa el botón “Deploy” (Instanciar) para volcar el programa y ejecutarlo. Al lado de este botón se encuentra un menú ≡ donde se tienen

varias opciones, con una de ellas se puede importar/exportar “Flows”. También se pueden instalar nuevos paquetes desde “Manage Palette” y pestaña “Install”.

5. Panel de información y configuración: en este panel se pueden encontrar varias funciones como la información de cada flujo, información de ayuda del nodo seleccionado, errores encontrados en el flujo programado, configuración de los nodos, diseño de programación, configuración y edición del dashboard.

InfluxDB

InfluxDB es una base de datos de series temporales diseñada específicamente para manejar grandes volúmenes de datos generados por sensores, métricas, eventos y cualquier tipo de información que varíe con el tiempo [10]. Es una herramienta clave en el desarrollo de aplicaciones IoT, ya que permite almacenar, consultar y analizar datos de manera eficiente. Su diseño optimizado para consultas temporales facilita la visualización y el análisis de datos a través de plataformas como Grafana.

Instalación y Configuración como Servicio

Para garantizar un rendimiento óptimo y una implementación cercana a entornos reales, se recomienda instalar InfluxDB en dispositivos dedicados, como una Raspberry Pi o un servidor local. Sin embargo, para estas prácticas, se realizará la instalación en el computador personal para facilitar el aprendizaje y la configuración inicial. Una vez instalado, se configurará como un servicio para que pueda ejecutarse de manera automática.

Paso 1: Descargar el instalador de InfluxDB

- Ve al sitio oficial de InfluxData: https://portal.influxdata.com/downloads/
- Selecciona la opción para Windows. Esto descargará un archivo ZIP con los binarios.
- Una vez descargado el archivo ZIP, descomprímelo en una carpeta de tu elección, por ejemplo: C:\InfluxDB.

Paso 2: Configuración como servicio:

- Abre una terminal o un explorador de archivos con permisos de administrador.
- Navega hasta la carpeta donde descomprimiste NSSM y ejecuta: nssm install InfluxDB
- Se abrirá una ventana donde debes configurar:
    - Path: La ruta del ejecutable de InfluxDB (por ejemplo, C:\ruta\a\influxd.exe).
    - Startup directory: La carpeta donde se encuentra el ejecutable.
    - Arguments: Usualmente puedes dejarlo vacío.
- Haz clic en "Install Service".

Paso 3: Iniciar el servicio:

- Abre el Administrador de Servicios de Windows (services.msc).
- Busca el servicio llamado "InfluxDB", haz clic derecho y selecciona Iniciar.
- Configura el tipo de inicio como Automático para que se inicie al encender la computadora.

Paso 4: Verificar la instalación:

- Abre un navegador web y visita: http://localhost:8086
- Sigue las instrucciones para crear una cuenta de administrador y configurar la base de datos inicial (opcional).

Grafana

Grafana es una plataforma de análisis y visualización de datos, especialmente útil para monitorear sistemas y aplicaciones en tiempo real. Es comúnmente utilizada junto con InfluxDB para la visualización de datos almacenados en bases de datos de series temporales, como los generados por sistemas IoT. Con Grafana, los usuarios pueden crear paneles (dashboards) interactivos que permiten visualizar datos en tiempo real, facilitando el análisis y la toma de decisiones basadas en estos datos.

Instalación y Configuración

Al igual que InfluxDB, se recomienda instalar Grafana en dispositivos dedicados como una Raspberry Pi para implementar soluciones IoT reales. Sin

embargo, para el propósito de estas prácticas, se procederá con la instalación local en el computador personal, lo que permitirá una configuración más accesible para los estudiantes y facilitará la integración con las bases de datos y otras herramientas necesarias en el entorno de prácticas.

Paso 1: Descargar el instalador de Grafana

- Ve al sitio oficial de Grafana: https://grafana.com/grafana/download
- Selecciona la opción para Windows y descarga el instalador (archivo .msi).

Paso 2: Ejecutar el instalador

- Haz doble clic en el archivo .msi descargado.
- Sigue el asistente de instalación y selecciona las opciones predeterminadas. Grafana se instalará en C:\Program Files\GrafanaLabs\grafana.

Paso 3: Iniciar el servicio de Grafana

- Una vez completada la instalación, el servicio de Grafana debería iniciarse automáticamente. Si no:
    - Abre el menú Inicio y busca Services.
    - Encuentra el servicio Grafana, haz clic derecho y selecciona Iniciar.

Paso 4: Acceder a la interfaz de Grafana

- Abre un navegador web y visita: http://localhost:3000
- Inicia sesión con las credenciales predeterminadas:
    - Usuario: admin
    - Contraseña: admin
- Cambia la contraseña en el primer inicio de sesión.

Protocolo MQTT

El protocolo MQTT (Message Queuing Telemetry Transport) es un protocolo de mensajería ligero y eficiente diseñado para dispositivos y aplicaciones en entornos de redes con recursos limitados, como los que encontramos en la Internet de las Cosas (IoT) [11]. MQTT es ideal para aplicaciones donde la baja latencia, la fiabilidad y la mínima huella de ancho de banda son esenciales.

Este protocolo sigue un modelo de comunicación publicador-suscriptor, lo que significa que los dispositivos o aplicaciones se suscriben a temas de interés y pueden recibir mensajes publicados por otros dispositivos sobre esos mismos temas [11]. El uso de un broker MQTT facilita la transmisión de los mensajes entre los dispositivos. se comunican a través de un servidor central denominado broker. El broker actúa como intermediario entre los dispositivos, gestionando el envío y recepción de mensajes, asegurando que los datos se transmitan de manera eficiente y segura.

Un broker es una pieza clave en el funcionamiento del protocolo MQTT, ya que se encarga de recibir los mensajes publicados por los clientes y distribuirlos a aquellos que se han suscrito a los temas (topics) correspondientes. Además de manejar las conexiones y el flujo de mensajes, el broker también puede garantizar diferentes niveles de calidad de servicio (QoS) para asegurar que los mensajes sean entregados de manera adecuada, incluso en condiciones de red inestables. Entre los ejemplos más conocidos de brokers MQTT se encuentran Mosquitto, HiveMQ y EMQX. Estos brokers permiten gestionar la comunicación entre miles de dispositivos en una red de manera eficiente y escalable.

El protocolo funciona bajo un modelo de mensajes pequeños y bajo consumo de energía, siendo muy eficiente para redes con capacidad limitada, como redes de sensores o redes móviles. MQTT utiliza tres niveles de QoS para asegurar la entrega de los mensajes:

- QoS 0: El mensaje se envía una sola vez y no se confirma su recepción.
- QoS 1: El mensaje se garantiza que se entrega al menos una vez.
- QoS 2: El mensaje se garantiza que se entrega exactamente una vez, sin duplicados.

Instalación de Mosquitto Broker

En el contexto de estas guías prácticas, utilizaremos Mosquitto como nuestro broker MQTT. Mosquitto es un broker ligero, de código abierto y ampliamente utilizado en proyectos de IoT debido a su facilidad de instalación y configuración, así como su buen rendimiento en entornos de bajo ancho de banda y alto volumen de mensajes. Además, al ser un software de código abierto, Mosquitto ofrece una gran flexibilidad para personalizar y adaptar su

funcionamiento a las necesidades del proyecto. A continuación, se describe cómo instalar Mosquitto en un sistema operativo Windows:

Paso 1: Descargar el instalador

- Descarga la última versión estable de Mosquitto desde su página oficial: https://mosquitto.org/download/

Paso 2: Instalación del broker

- Una vez descargado el archivo, ejecutar el instalador .exe y seguir las instrucciones del asistente de instalación.
- Asegurarse de seleccionar la opción para instalar el servicio de Mosquitto si se desea que se ejecute automáticamente al iniciar el sistema.

Paso 3: Configuración de Mosquitto

- Durante la instalación, se generará un archivo de configuración predeterminado (mosquitto.conf).
- Puedes editar este archivo para personalizar parámetros como los puertos, la persistencia de los mensajes y la autenticación de usuarios si es necesario. Para que el broker funcione correctamente en un computador como servidor local se deben agregar las siguientes líneas de código:

listener 1883

allow_anonymous true

- Se deben crear dos reglas en el firewall de tu computador para permitir las entradas y salidas en el puerto 1883.
- Asegúrate de reiniciar tu computador para que los cambios realizados en el archivo de configuración y en el firewall se ejecuten.

Paso 4: Iniciar el servicio

- Después de la instalación, el servicio de Mosquitto debería iniciarse automáticamente.
- Si no es así, puedes iniciarlo manualmente a través de los servicios de Windows o desde la línea de comandos con el comando mosquitto -v para iniciar el broker en modo de depuración.

Una vez completada la instalación, Mosquitto estará listo para gestionar la comunicación entre los dispositivos que utilicen el protocolo MQTT.

Antecedentes

La asignatura de Internet de las Cosas es parte del cuarto nivel de la carrera de Tecnología Superior en Redes y Telecomunicaciones del Instituto Superior Tecnológico 17 de Julio, y se encuentra dentro de la unidad curricular profesional. Esta asignatura requiere que los estudiantes posean conocimientos previos adquiridos en las asignaturas de Sistemas Embebidos (de segundo nivel) y Redes de Sensores (de tercer nivel). Estos cursos brindan una base sólida en conceptos de programación de microcontroladores y en el manejo de redes de dispositivos interconectados, elementos esenciales para comprender y desarrollar aplicaciones IoT.

La asignatura de Sistemas Embebidos proporciona a los estudiantes los conocimientos necesarios para trabajar con microcontroladores y dispositivos electrónicos, comprendiendo su funcionamiento y cómo interactúan con el software. Estos conceptos son clave para las prácticas de IoT, ya que implican la programación y configuración de placas como, en este caso el módulo ESP32. Los estudiantes deberán saber cómo conectar y configurar sensores y actuadores, así como desarrollar aplicaciones que integren hardware y software de manera efectiva.

Por otro lado, en Redes de Sensores, los estudiantes adquieren habilidades para diseñar y gestionar redes de sensores que permiten recopilar datos de diferentes entornos y transmitir esta información a otros dispositivos o plataformas. Esta base es crucial para las prácticas de IoT, donde se debe garantizar que los dispositivos se conecten de manera eficiente y estable a la red, así como para el uso de protocolos de comunicación. A lo largo de las prácticas, los estudiantes utilizarán herramientas como Node-RED y plataformas de bases de datos como InfluxDB para gestionar y visualizar los datos transmitidos por los sensores.

Con estos antecedentes, los estudiantes estarán capacitados para enfrentar los desafíos prácticos de IoT, integrando conocimientos de electrónica y programación para desarrollar soluciones efectivas. Este libro de guías prácticas está diseñado para guiarlos paso a paso en el proceso de aprendizaje y aplicación de tecnologías IoT, asegurando que los estudiantes puedan aprovechar al máximo los recursos disponibles en esta área de estudio, con cada capítulo dedicado a una práctica específica.

CAPÍTULO II

Diseño de Experiencia-Interacción para Aplicaciones de IoT

ALUMNI EDITORA

Fundamentación

La interacción entre dispositivos IoT y los usuarios requiere el desarrollo de interfaces gráficas que sean intuitivas, funcionales y adaptables a las necesidades de cada aplicación [12]. En esta práctica, los estudiantes tendrán la oportunidad de diseñar y configurar una interfaz interactiva utilizando Node-RED, integrando sensores y actuadores conectados a la placa ESP32. Durante el desarrollo de las actividades, los participantes trabajarán en la creación de flujos visuales que permitan gestionar dispositivos en tiempo real, explorando las capacidades de personalización y control que ofrece esta herramienta.

El diseño de la experiencia de usuario (UX) en aplicaciones IoT implica la creación de interfaces que faciliten la interacción con dispositivos conectados. Estas interfaces permiten a los usuarios controlar y monitorear diversos dispositivos a través de sensores, actuadores y redes de comunicación [13]. Para ello, se emplean plataformas como Node-RED, un entorno de desarrollo visual basado en flujos que facilita la programación y control de dispositivos IoT de manera sencilla y eficiente.

Los flujos de Node-RED estarán compuestos por nodos de entrada (sensores) y nodos de salida (actuadores o notificaciones), permitiendo diseñar un sistema de monitoreo y control en tiempo real, similar a los utilizados en aplicaciones reales de domótica, monitoreo ambiental o gestión de energía en edificios inteligentes.

La importancia de esta práctica radica en que permite a los estudiantes aprender a diseñar aplicaciones interactivas para dispositivos IoT, un campo que está en constante crecimiento y es crucial en sectores como la automatización del hogar, la salud digital, la agricultura de precisión y la industria 4.0. Los conocimientos adquiridos en esta práctica brindan habilidades para desarrollar interfaces de usuario efectivas, lo cual es esencial para garantizar que los sistemas IoT sean fáciles de usar y accesibles para usuarios no especializados. El diseño UX es un factor crítico en la adopción de estas tecnologías. Una interfaz mal diseñada puede dificultar la interacción y el control efectivo de los dispositivos, lo que limita el potencial del sistema IoT. En la tabla 1, se describen las aplicaciones en el campo laboral y/o profesional:

Tabla 1

Aplicaciones profesionales de la práctica 1.

Aplicación	Descripción
Domótica	Los sistemas IoT en hogares inteligentes requieren interfaces que permitan a los usuarios controlar luces, termostatos, cámaras de seguridad, etc., de manera remota.
Monitoreo ambiental	Las aplicaciones de IoT en este campo, como la medición de calidad del aire, requieren interfaces para visualizar datos en tiempo real y generar alertas basadas en los parámetros monitoreados.
Salud digital	En la telemedicina y el monitoreo remoto de pacientes, las interfaces de usuario permiten a los profesionales de salud monitorear constantes vitales de los pacientes a través de dispositivos IoT.
Automatización industrial	Los sistemas de monitoreo de maquinaria y procesos industriales requieren interfaces interactivas para visualizar el estado de los dispositivos IoT y facilitar la toma de decisiones en tiempo real.

Objetivos

Objetivo General

Diseñar una interfaz gráfica interactiva en Node-RED que permita la conexión de sensores y actuadores, facilitando la interacción con dispositivos IoT a través de flujos de datos visuales y en tiempo real.

Objetivos Específicos

Configurar nodos de entrada y salida en Node-RED, vinculando sensores y actuadores conectados a la placa ESP32.

Implementar elementos de diseño de interfaces (botones, sliders, gráficos) que permitan al usuario controlar dispositivos IoT.

Probar la funcionalidad de la interfaz en un ambiente controlado utilizando la placa ESP32.

Preparación Previa

Antes de iniciar esta práctica, es fundamental que los estudiantes revisen los conceptos básicos de IoT y su impacto en la experiencia del usuario. Comprender cómo los dispositivos IoT interactúan con el entorno y con las personas permitirá diseñar aplicaciones más intuitivas y efectivas. Además, es esencial explorar el rol de las interfaces gráficas en este contexto, enfocándose en su capacidad para simplificar la interacción entre los usuarios y los dispositivos conectados.

También se recomienda investigar a profundidad el uso de Node-RED, desde su instalación y configuración básica hasta el manejo de nodos para construir flujos interactivos. Esta herramienta será clave en la práctica, ya que posibilita la creación de interfaces visuales que conectan sensores y actuadores de forma dinámica. Adicionalmente, los estudiantes deben familiarizarse con los principios del diseño de interfaces gráficas, priorizando aspectos como la funcionalidad, la claridad visual y la facilidad de uso, especialmente en entornos IoT donde la experiencia de usuario desempeña un papel crucial.

Procedimiento

En esta práctica, se desarrollará una solución IoT que permite la lectura de datos ambientales provenientes de un sensor DHT11/DHT22, incluyendo parámetros como temperatura y humedad relativa, y su procesamiento para generar respuestas automáticas en función de umbrales predefinidos. Cuando los valores superen los límites establecidos, se activarán indicadores visuales a través de LEDs, los cuales funcionarán como alertas físicas inmediatas. Por lo tanto, Asegúrate de instalar las librerías necesarias desde el Gestor de Librerías del Arduino IDE, buscando “DHT sensor library” de Adafruit.

Adicionalmente, los datos capturados se enviarán mediante comunicación serial a Node-RED, donde se diseñará un dashboard interactivo. Este tablero

mostrará en tiempo real los valores del sensor utilizando Gauges y generará notificaciones visuales en caso de activarse las alertas, como indicadores en LEDs virtuales y mensajes de texto. Esta práctica integra hardware y software para ofrecer una solución funcional y visualmente intuitiva que resalta la importancia de la interacción y monitoreo en sistemas IoT.

Materiales

Los materiales tanto software como hardware que se utilizan en esta práctica son los siguientes:

Hardware:

- Placa ESP32 (se trabaja con el módulo ESP32-WROOM-32).
- Sensor DHT11 o DHT22 (para medir temperatura y humedad).
- Resistencia de 10 kΩ (para conexión del sensor DHT).
- LEDs (mínimo dos de diferente color, para indicar alertas de umbrales).
- Resistencias de 220 Ω (para protección de los LEDs).
- Protoboard (para realizar las conexiones).
- Cables de conexión (tipo jumper: macho-macho y macho-hembra).
- Fuente de alimentación USB-micro USB (para la placa ESP32).

Software:

- Arduino IDE (para programar la ESP32).
- Node-RED (para el diseño del dashboard y la interacción visual).
- Drivers USB-UART (en el caso que el sistema operativo no reconozca la placa ESP32 utilizada).
- Dashboard de Node-RED (instalado como complemento para crear interfaces gráficas).

Opcional:

- Multímetro digital (para verificar conexiones y niveles de voltaje).
- Caja de conexiones o soporte (para organizar el prototipo durante la práctica).

Conexiones del Hardware

Para implementar esta práctica, sigue estas indicaciones para realizar las conexiones correctamente en la placa ESP32 como se muestra en la figura 3:

Conexión del Sensor DHT11/DHT22

- **Pin de alimentación (VCC)**: Conectar al pin **3.3V** de la ESP32.
- **Pin de datos (DATA)**: Conectar a un pin digital de la ESP32 (por ejemplo, el pin GPIO25).
    - Agregar una resistencia de 10 kΩ entre el pin **VCC** y el pin **DATA** para garantizar la comunicación estable del sensor.
- **Pin de tierra (GND)**: Conectar al pin **GND** de la ESP32.

Conexión de los LEDs de Alerta

- **Ánodo (+)**:
    - Conectar el terminal positivo de cada LED a un pin digital de la ESP32. Por ejemplo:
        - LED naranja (alerta de temperatura): **GPIO14**.
        - LED azul (alerta de humedad): **GPIO12**.
- **Cátodo (-)**: Conectar el terminal negativo de cada LED a **GND** de la ESP32 a través de una resistencia de 220 Ω (para limitar la corriente y proteger el LED).

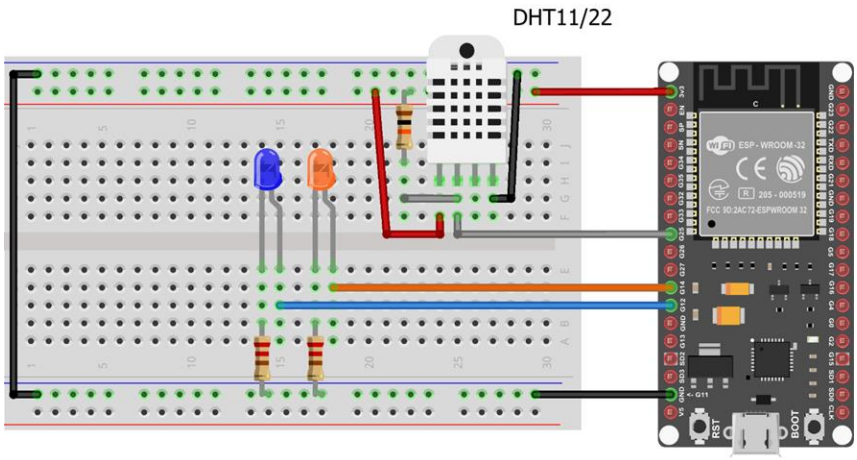
Conexión USB

- Conecta la placa ESP32 al computador mediante un cable **USB micro-B o USB-C** (dependiendo del modelo de la placa). Esto permitirá tanto la programación como la alimentación eléctrica de la placa.

Verificación de Conexiones (Opcional)

- Antes de energizar la ESP32, utiliza un multímetro para verificar la continuidad de las conexiones.
- Asegúrate de que no existan cortocircuitos en la Protoboard y que los pines estén conectados correctamente.

Figura 3
Conexiones de la Práctica 1.



Configuración del Software

Para conseguir los objetivos de la práctica, es necesario desarrollar un código en el IDE de Arduino para cargarlo a la placa ESP32 y esta pueda monitorear la temperatura y humedad relativa. También se debe poder controlar 2 LEDs de alerta de diferentes colores que se encenderán si se supera un umbral específico. Por otro lado, en Node-RED se deben configurar los nodos necesarios para visualizar los datos del sensor y proporcionar las alertas respectivas. A continuación, se explican a detalle estas configuraciones.

Código de Arduino

El código completo se encuentra escrito en el Anexo 1, en esta sección se detalla cada parte del código para tener una mejor comprensión de este.

- **Inclusión de Librerías:**
  - Se deben incluir las librerías necesarias, en este caso la librería DHT.h.

```
// Incluir las librerías necesarias
#include <DHT.h>
```

- **Definición de Pines y Umbrales**:
  - Se definen los pines a los que están conectados el sensor DHT y los LEDs.

- Los umbrales de temperatura y humedad se configuran a 30°C y 70%, respectivamente.

```
// Definición de pines
#define SENSOR_PIN 26      // Pin de conexión para el sensor DHT11/22
#define LED_TEMP 12        // Pin para el LED de temperatura
#define LED_HUM 13         // Pin para el LED de humedad

// Definición de umbrales
#define TEMP_UMBRAL 30     // Umbral de temperatura (en grados Celsius)
#define HUM_UMBRAL 70      // Umbral de humedad (en porcentaje)
```

- **Configuración del Tipo de Sensor:**

```
// Configurar el tipo de sensor DHT
#define DHT_TYPE DHT22         // O puedes usar DHT11 dependiendo de tu sensor
DHT dht(SENSOR_PIN, DHT_TYPE); // Crear objeto para el sensor DHT
```

- **Inicialización del Sensor, Pines y Comunicación Serial:**
  - Se incluye la librería <DHT.h> y se configura el tipo de sensor (DHT11 o DHT22).
  - En el setup(), se inicializan los pines para los LEDs y se configura el sensor DHT.
  - Se inicializa la comunicación serial a 115200 baudios (velocidad compatible con ESP32).

```
void setup() {
  // Inicializar la comunicación serial
  Serial.begin(115200);

  // Inicializar los pines de los LEDs como salida
  pinMode(LED_TEMP, OUTPUT);
  pinMode(LED_HUM, OUTPUT);

  // Inicializar el sensor DHT
  dht.begin();
}
```

- **Lectura de Temperatura y Humedad:**
  - En el loop(), se leen los valores de temperatura y humedad del sensor.
  - Si las lecturas son válidas, se envían al monitor serial.

```
void loop() {
  // Leer los valores de temperatura y humedad
  float temperatura = dht.readTemperature();  // Temperatura en grados Celsius
  float humedad = dht.readHumidity();         // Humedad en porcentaje

  // Verificar si las lecturas fueron exitosas
  if (isnan(temperatura) || isnan(humedad)) {
    Serial.println("Error al leer del sensor DHT");
    return;
  }
```

- **Control de LEDs:**
  - Se calcula si los valores de temperatura y humedad superan los umbrales definidos.
  - Si el valor supera el umbral, el LED correspondiente se enciende. Si no, se apaga.

```
// Control LEDs
// Enciende LED de temperatura si supera el umbral
int led_temp = temperatura > TEMP_UMBRAL ? 1 : 0;
// Enciende LED de humedad si supera el umbral
int led_hum = humedad > HUM_UMBRAL ? 1 : 0;

// Escribir el estado en los pines de los LEDs
digitalWrite(LED_TEMP, led_temp);  // Encender o apagar LED de temperatura
digitalWrite(LED_HUM, led_hum);    // Encender o apagar LED de humedad
```

- **Envío de Datos por Comunicación Serial:**
  - Se imprimen los datos necesarios por comandos en el monitor serial.
  - Se envían comas "," para separar cada valor y poder procesarlos en Node-RED.

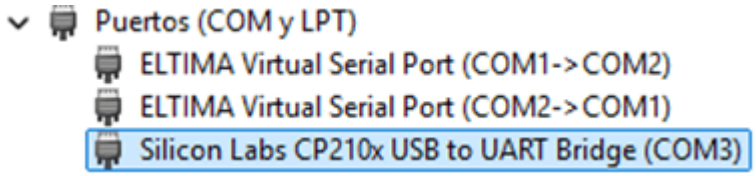
```
// Imprimir los valores leídos en el monitor serial
Serial.print(temperatura);
Serial.print(",");
Serial.print(humedad);
Serial.print(",");
Serial.print(led_temp);
Serial.print(",");
Serial.println(led_hum);
```

- **Retraso:**
  - Se agrega un retraso de 2 segundos entre cada lectura para evitar lecturas demasiado rápidas del sensor.

- Se cierra el void loop con la llave respectiva.

```
  // Esperar 2 segundos antes de leer nuevamente
  delay(2000);
}
```

- **Compilación y Subida del Código:**
  - Compilar el código para comprobar que no existen errores, en el caso de haberlos, revisar bien el código y depurar los errores.
  - Conectar la placa ESP32 a un puerto USB del computador y comprobar que puerto se asigna.



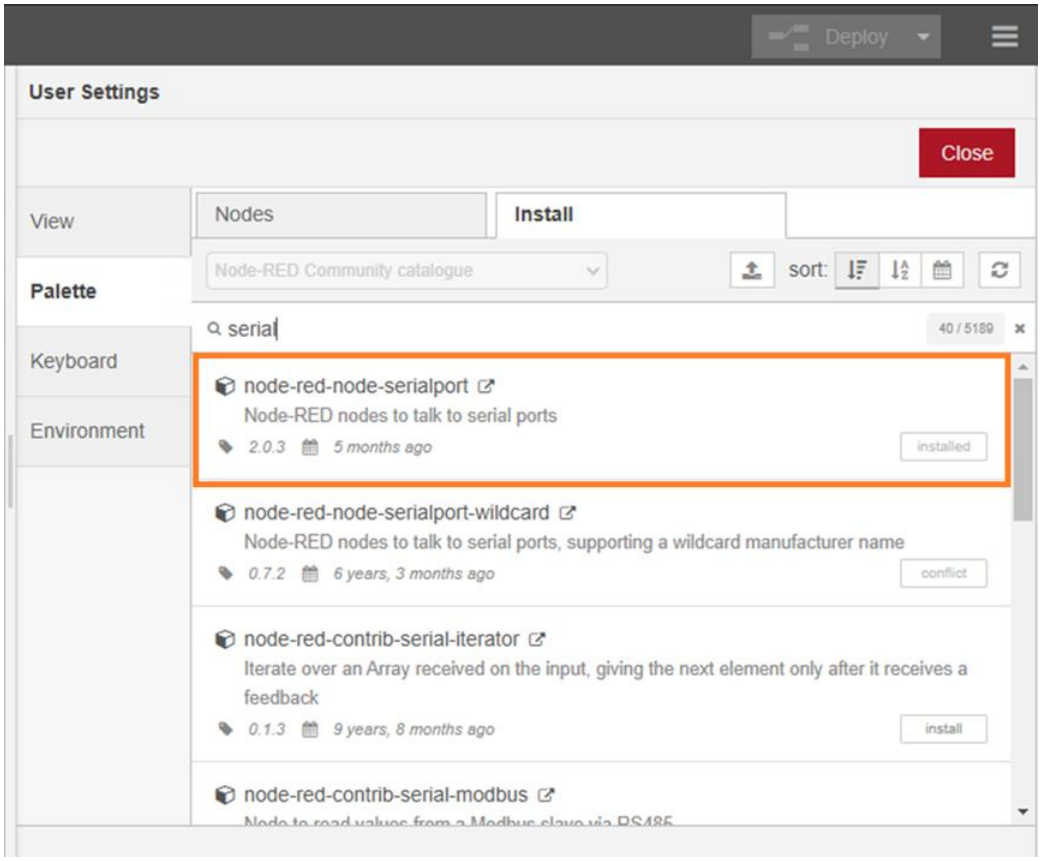
  - Seleccionar el puerto específico en el IDE y subimos el código a la ESP32.

Flujo de Node-RED

Para el diseño del flujo de la Práctica 1 se deben realizar configuraciones previas en Node-RED, debido a que se necesitan nodos que no vienen instalados por defecto y también el panel de visualización (dashboard). Entonces, una vez abierta la interfaz del programa mediante un navegador web y la dirección: http://localhost:1880/ debemos dirigirnos a la sección de configuración ≡ y en la opción “manage palette” (administrar paleta) seleccionar la pestaña “Install” (instalar), para buscar e instalar (ejemplo figura 4) los siguientes nodos:

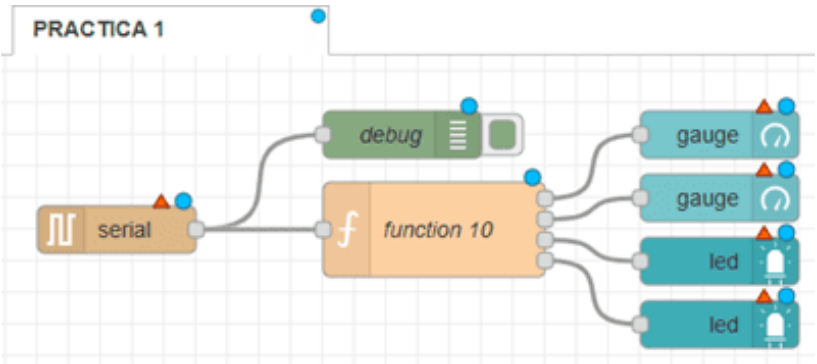
- node-red-node-serialport
- node-red-dashboard
- node-red-contrib-ui-led

Figura 4
Ejemplo de instalación del nodo serialport.



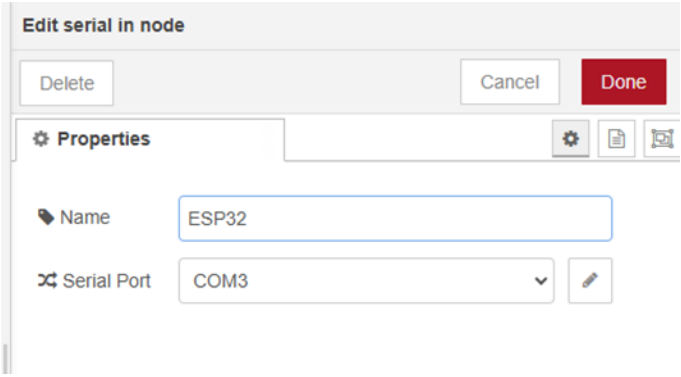
Con los nodos necesarios instalados se procede con el diseño y configuración del flujo que lo denominaremos “PRACTICA 1”, seleccionamos y arrastramos al área de trabajo los nodos: serial In, debug, function (con 4 salidas), gauge (2 nodos), led (2 nodos) y los conectamos como se muestra en la figura 5:

Figura 5
Conexión de nodos en Node-RED para la práctica 1.

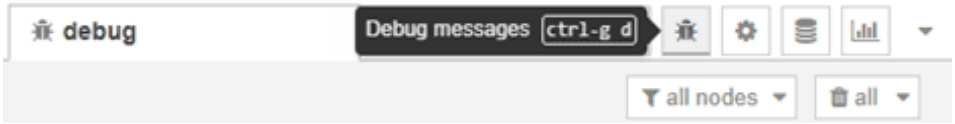


Los nodos no están configurados y por eso se marcan las alertas en forma de circunferencias azules y triángulos rojos. Para corregir estos errores se procede con la configuración de cada nodo haciendo doble clic en cada uno y llenando los espacios específicos.

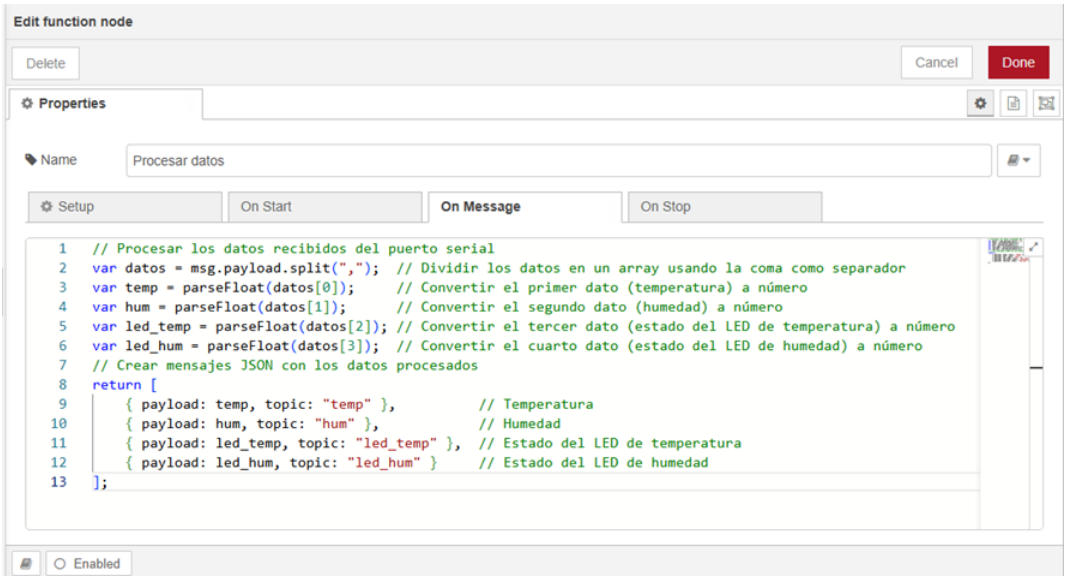
- **Nodo serial In:**
  - Configurar el nodo con el nombre ESP32 y seleccionar el puerto específico al que se conectó la placa (en este caso el COM3), hacer clic a “Done” para finalizar la configuración del nodo.



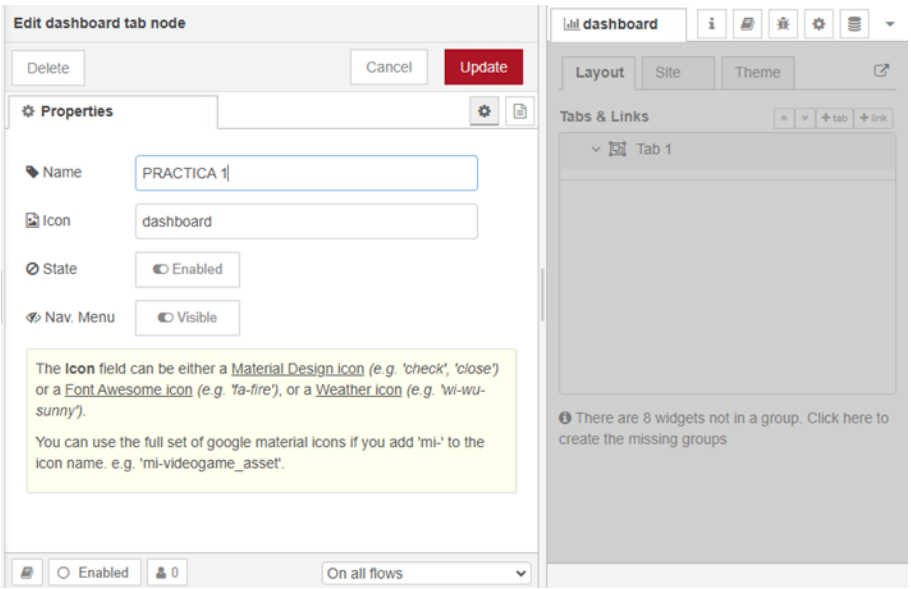
- **Nodo debug**
  - Este nodo no necesita configuración ya que solo muestra los datos recibidos en la pestaña “Debug messages” ubicada en el panel de información y configuración.



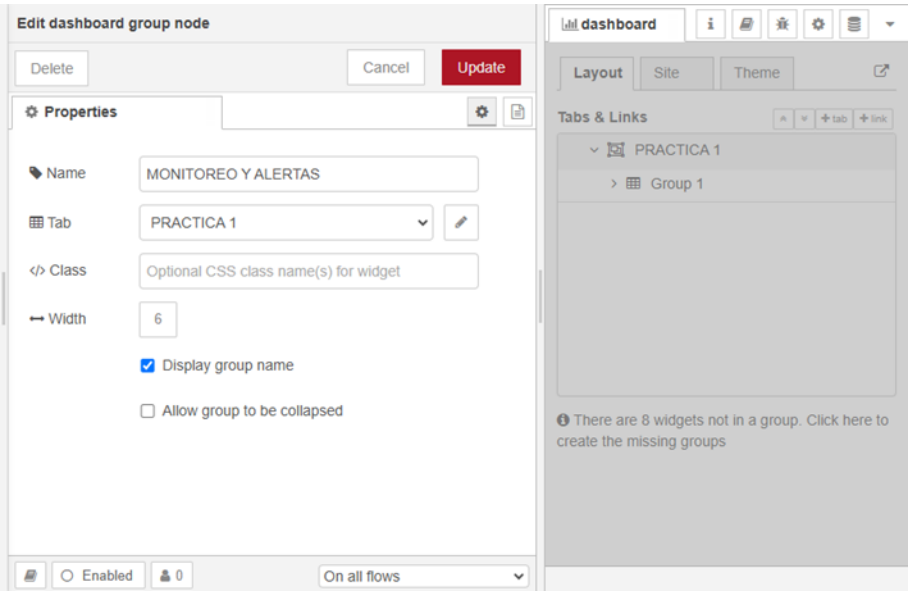
- **Nodo function:**
  - Este nodo sirve para procesar los datos enviados por la ESP32 y adaptarlos para su visualización en el dashboard.
  - Configuramos un nombre como por ejemplo “Procesar Datos” y en la pestaña “On Message” introducimos el código en JavaScript. Hacer clic a “Done” para finalizar la configuración del nodo.



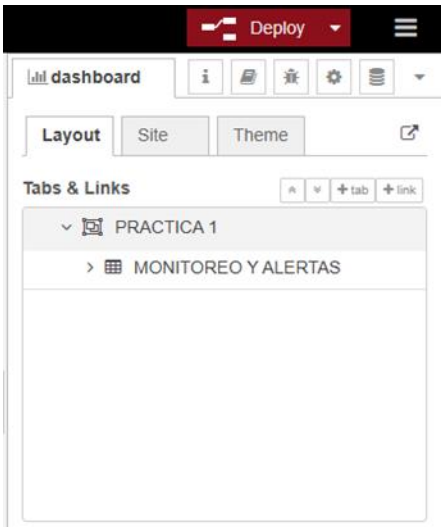
- **Nodos gauge**
  - Para configurar estos nodos es necesario agregar un grupo y una pestaña en el dashboard a los cuales van a pertenecer. Desde el panel de información y configuración, en la sección dashboard agregamos una nueva pestaña (tab) y la nombramos PRACTICA 1.



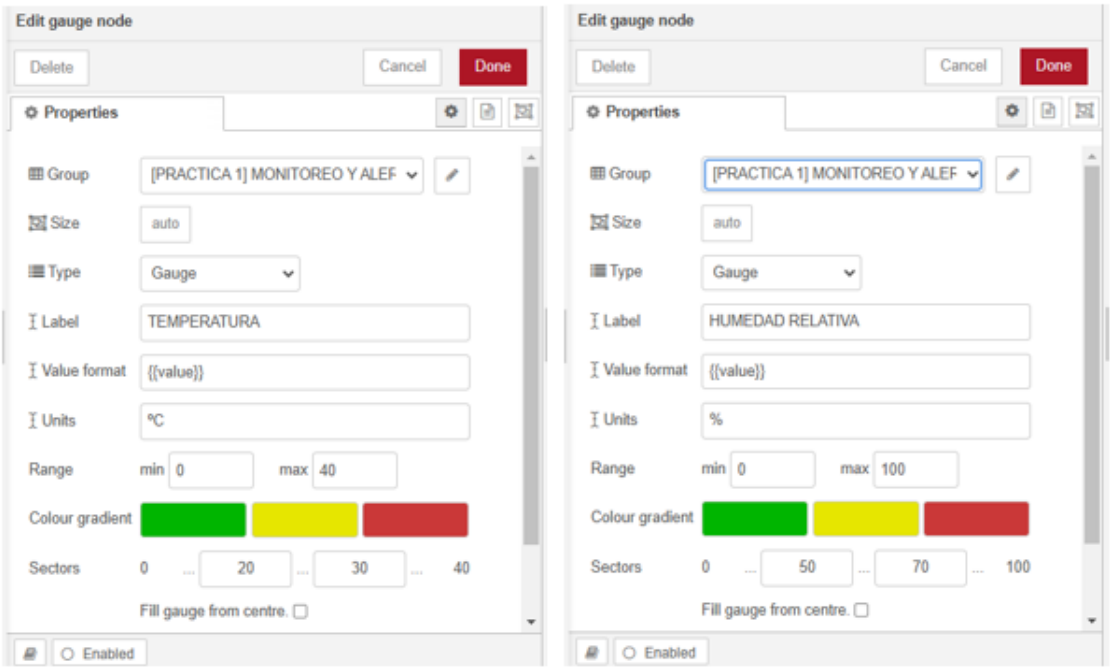
  - Damos clic en Update y verificamos que se haya agregado correctamente. Ahora agregamos un nuevo grupo dentro de la pestaña creada y lo denominamos MONITOREO Y ALERTAS.



- Oprimimos Update y comprobamos que el grupo y la pestaña se hayan agregado al panel de configuración.

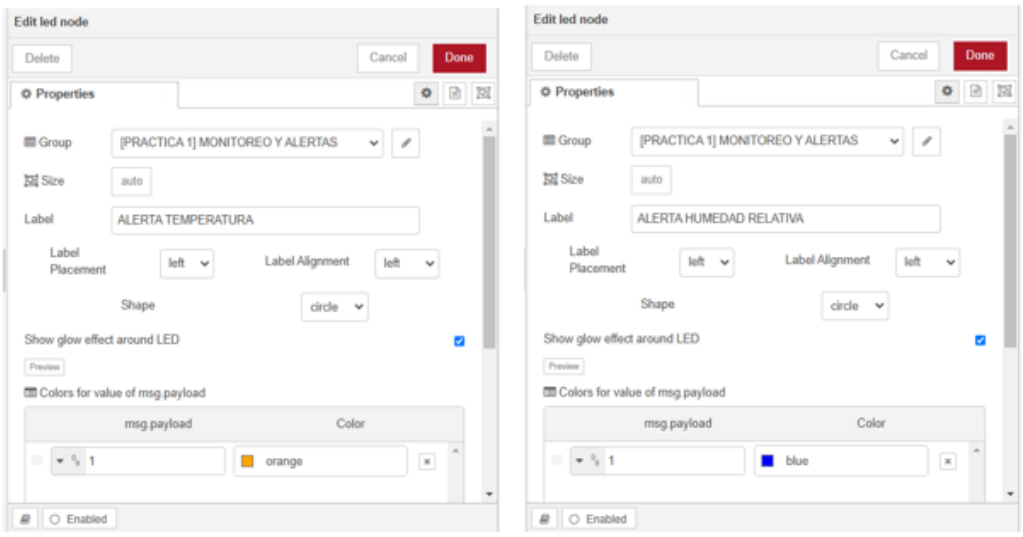


- Con esto configurado ya podemos seleccionar los gauges y agregarlos al grupo MONITOREO Y ALERTAS, configurando uno para temperatura y el otro para humedad relativa.



- Notar que se pueden establecer rangos con colores distintivos para hacer que los gauges tengan mejor presentación. Hacer clic en Done para finalizar la configuración.

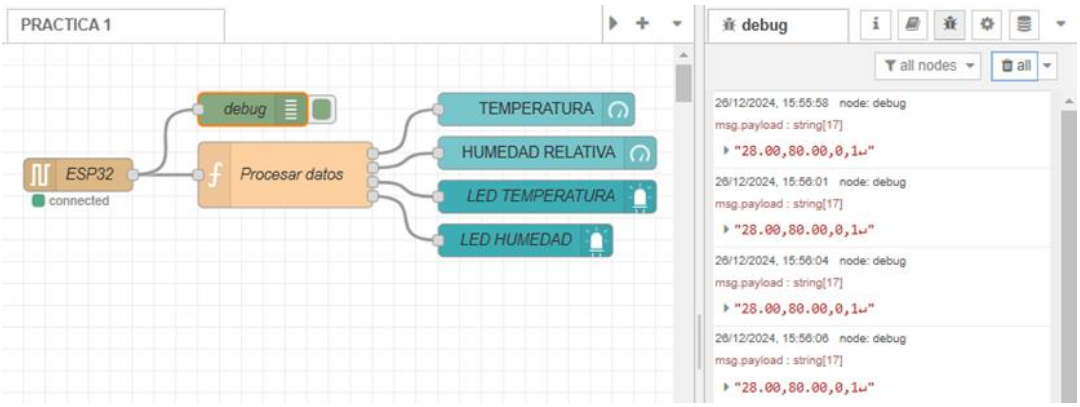
- **Nodos led**
  - Al igual que los gauges, los nodos led necesitan pertenecer a un grupo dentro del dashboard, por lo que se agregan al grupo ya creado y se configuran de la siguiente forma:



- Hacer clic en Done y ya tendríamos todos los nodos configurados.

Con todos los nodos configurados, se procede ha instanciar la configuración dando clic en el botón Deploy. Se puede notar que todas las alertas han desaparecido y nos quedaría el flujo como se muestra en la figura 6. Además, la información enviada por la placa ESP32 ya se puede observar en el panel de información mediante la pestaña debug.

Figura 6
Flujo de la Práctica 1 configurado.

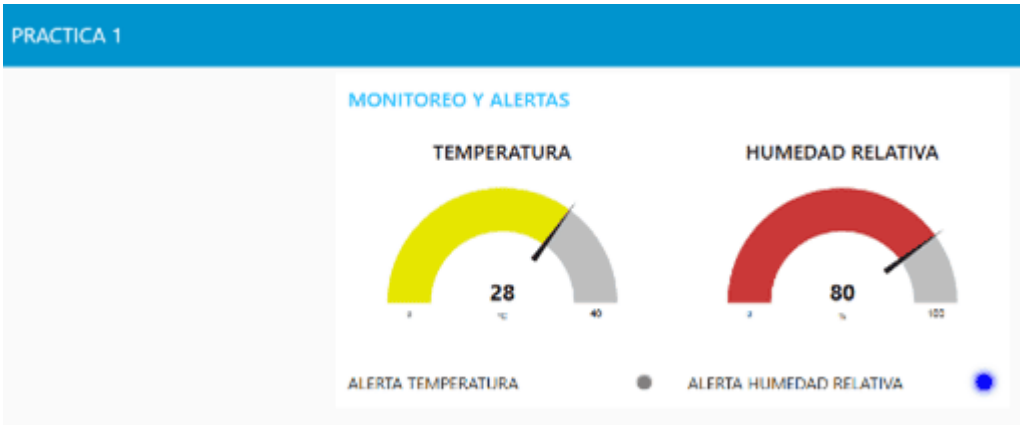


Panel de Visualización

Para el monitoreo de las variables y la visualización de alertas, se inicializa el dashboard. Puede hacerse desde el panel de configuración en la pestaña dashboard oprimiendo el botón o desde una nueva pestaña del navegador con la dirección: http://localhost:1880/ui/. En la figura 7 se muestran los resultados de la configuración en Node-RED para la visualización de temperatura y humedad relativa, se nota que el LED azul está encendido debido a que la humedad sobrepasa el umbral establecido del 70%.

Los dashboard pueden configurarse dependiendo de la aplicación y el diseño siempre puede variar de acuerdo con las especificaciones técnicas del proyecto.

Figura 7
Dashboard de la Práctica 1.



Resultados de la Práctica

Con las conexiones de hardware y las configuraciones del software se puede comprobar el correcto funcionamiento del sistema IoT, el cual muestra como cambian los datos de temperatura y humedad conforme estos varían en la vida real, mostrando alertas mediante el uso de LEDs para que el usuario tenga una experiencia más interactiva.

Los estudiantes deben seguir paso a paso el desarrollo de esta práctica para que puedan documentar el proceso de desarrollo del proyecto, incluyendo la configuración del hardware, el código fuente y la configuración de Node-RED, proporcionando una explicación detallada de cada paso en un informe de laboratorio, incluyendo fotografías del circuito y capturas de pantalla de la interfaz gráfica de Node-RED. Además, deben anexar el código de programación.

Evaluación del Aprendizaje

El docente calificará todo el proceso y desarrollo de la práctica estableciendo una calificación final sobre 10 puntos en base a la rúbrica presentada en la tabla 2:

Tabla 2
Rúbrica de evaluación.

Criterio de Evaluación	Nivel de Desempeño
Diseño e implementación del circuito	2,5 puntos
Configuración de la interfaz de Node-RED	2,5 puntos
Integración del hardware y software para el monitoreo	2,5 puntos
Desarrollo del informe de laboratorio con claridad y que contenga toda la información	2,5 puntos

Actividades Complementarias

Con el objetivo de practicar y mejorar el diseño UX se plantean las siguientes actividades para que el estudiante desarrolle:

- **Nodos Adicionales**
  - Agrega más nodos que permitan generar alertas visuales con nodos text y auditivas con nodos audio out.
- **Modificación de Umbrales**
  - Ajusta los valores de los umbrales de temperatura y humedad en el código de Arduino para diferentes condiciones ambientales (por ejemplo, climas cálidos o fríos).
  - Observa y analiza cómo cambian las alertas en Node-RED (Gauges y LEDs).

- **Integración de un Sensor Adicional**
  - Añade un sensor adicional (como un sensor de luz LDR o un sensor de gas MQ-2).
  - Modifica el código de Arduino y el flujo de Node-RED para integrar las lecturas del nuevo sensor, mostrando sus datos en un Gauge adicional y generando alertas si los valores superan un umbral definido.

- **Control de Actuadores Adicionales**
    - Conecta un actuador adicional, como un relé que controle un dispositivo externo.
    - Modifica el flujo en Node-RED para que el actuador se active o desactive en función de las lecturas del sensor o mediante un botón en la interfaz gráfica.
- **Creación de un Dashboard Personalizado**
    - Diseña un Dashboard en Node-RED con una temática específica, utilizando gráficos personalizados, sliders y botones adicionales.
    - Incluye elementos que permitan visualizar tendencias de datos en tiempo real o históricos.

Cuestionario

- ¿Qué elementos de hardware y software se utilizaron para implementar la interacción entre sensores y actuadores en esta práctica? Explique brevemente la función de cada uno.

_____
_____
_____
_____
_____
_____
_____

- ¿Qué nodos se utilizaron en Node-RED para representar las lecturas del sensor y las alertas de umbrales? ¿Cómo se configuraron para lograr su funcionalidad?

_____
_____
_____
_____
_____

- En el código de Arduino, ¿cómo se determina si se debe encender o apagar los LEDs en función de los valores de temperatura y humedad? Explique la lógica utilizada.

- ¿Qué datos procesa el nodo "function" en Node-RED, y cómo se organiza esta información para ser enviada al resto del flujo?

- Durante la simulación de condiciones críticas (umbrales superados), ¿cómo respondió el sistema (lecturas en el dashboard, encendido de LEDs y alertas)? ¿Qué mejoras sugerirías para optimizar su desempeño?

CAPÍTULO III

Programación y administración de energía

ALUMNI EDITORA

Fundamentación

La administración de energía en dispositivos IoT es un aspecto crucial, especialmente para aquellos que operan con baterías o en ambientes de recursos limitados. Uno de los enfoques principales para reducir el consumo energético es el uso de modos de ahorro de energía, como el modo Deep Sleep en los microcontroladores ESP32, que minimiza el consumo al desactivar componentes no esenciales mientras el dispositivo no está en uso activo [6].

La práctica se enfoca en enseñar cómo programar y evaluar el impacto del ahorro de energía mediante herramientas como Arduino IDE, que permite controlar los modos de operación del ESP32, y sensores analógicos y digitales para medir el consumo energético. Este ejercicio conecta la teoría de administración energética con su implementación en sistemas reales.

La gestión energética en IoT es esencial para garantizar que los dispositivos puedan operar de manera continua y eficiente, especialmente en aplicaciones remotas o autónomas. Los estudiantes aprenderán técnicas prácticas para optimizar el consumo energético, un conocimiento vital para diseñar sistemas IoT sostenibles.

Además, esta práctica les ayudará a comprender cómo planificar el despliegue de dispositivos IoT, considerando factores como la autonomía energética y las condiciones del entorno, asegurando así una implementación exitosa. En la tabla 3 se presentan las aplicaciones profesionales de la práctica 2.

Tabla 3
Aplicaciones profesionales de la práctica 2.

Aplicación	Descripción
Sensores agrícolas	Dispositivos IoT utilizados en agricultura de precisión requieren operar con bajo consumo energético para maximizar la duración de las baterías en entornos remotos.
Monitoreo ambiental	Los dispositivos de monitoreo, como estaciones meteorológicas, deben optimizar su uso de energía para operar durante largos periodos sin intervención humana.

Automatización residencial	La optimización energética mejora la sostenibilidad y reduce los costos de operación en hogares inteligentes.

Objetivos

Objetivo General

Optimizar el consumo energético de sistemas IoT basados en ESP32 para aplicaciones agrícolas mediante la programación de modos de bajo consumo y el uso eficiente de sensores digitales y analógicos.

Objetivos Específicos

Configurar sensores digitales y analógicos para la medición de variables agrícolas clave (temperatura, humedad del suelo y luz ambiental).

Implementar técnicas de ahorro de energía en la programación del ESP32, como el uso del modo Deep Sleep.

Enviar datos recopilados a través del protocolo MQTT a un broker, garantizando eficiencia en la comunicación.

Preparación Previa

La práctica 2 requiere una sólida preparación previa para comprender conceptos fundamentales relacionados con la administración eficiente de energía en sistemas IoT, especialmente al utilizar la placa ESP32 en aplicaciones agrícolas. Un punto clave a revisar es el modo Deep Sleep de la ESP32, una funcionalidad diseñada para reducir el consumo energético. Este modo permite desactivar componentes internos, como la CPU, mientras mantiene elementos esenciales como el RTC (reloj de tiempo real) activos, lo que habilita al dispositivo a despertar después de un período programado. Los estudiantes deben investigar cómo funciona este modo, cuáles son sus beneficios en entornos de bajo consumo, y cómo implementarlo mediante programación. Además, deben estudiar ejemplos prácticos que ilustren cómo utilizar el modo Deep Sleep en sistemas que recolectan datos a intervalos regulares.

En aplicaciones agrícolas, los sensores digitales y analógicos juegan un rol crucial para monitorear variables como temperatura, humedad ambiental, humedad del suelo y luz. Es esencial que los estudiantes comprendan las diferencias entre sensores digitales, como el DHT11/DHT22, y sensores analógicos, como los de humedad del suelo o LDR (resistencias dependientes de luz). También es importante que conozcan cómo conectar estos sensores a la ESP32, identificando correctamente los pines de entrada digital y analógica, así como interpretar las señales generadas para convertirlas en información útil. Esta comprensión ayudará a diseñar sistemas agrícolas inteligentes y eficientes que aprovechen al máximo los datos recolectados.

Es imprescindible revisar el protocolo MQTT, que será utilizado para transmitir los datos recolectados de los sensores hacia una plataforma central. Este protocolo funciona bajo un modelo de suscriptor/publicador, permitiendo una comunicación eficiente y organizada mediante tópicos. Los estudiantes deben investigar cómo funciona MQTT, comprender la importancia del uso de tópicos para segmentar los mensajes y aprender a configurar un broker MQTT como Mosquitto. Además, deberán realizar pruebas simples de conexión para asegurar que los datos de los sensores puedan transmitirse correctamente y optimizar la comunicación para minimizar el uso de ancho de banda y energía, factores críticos en aplicaciones IoT agrícolas. Esta preparación teórica es fundamental para garantizar el éxito en la implementación de la práctica.

Procedimiento

En esta práctica, los estudiantes configurarán y programarán una placa ESP32 para la recolección de datos de sensores digitales y analógicos utilizados en entornos agrícolas, como sensores de temperatura, humedad y humedad del suelo. La información recolectada se transmitirá utilizando el protocolo MQTT hacia un broker, permitiendo su integración en una plataforma de monitoreo en tiempo real. Además, se explorarán estrategias de administración de energía para maximizar la eficiencia del sistema, haciendo uso del modo Deep Sleep de la ESP32 para reducir el consumo energético cuando el dispositivo no está recolectando datos. Por lo tanto, Asegúrate de instalar las librerías necesarias desde el Gestor de Librerías del Arduino IDE, buscando “PubSubClient” de Nick O’Leary.

El procedimiento incluye la programación de la ESP32 para leer datos de sensores, el envío de esta información mediante MQTT, y la configuración de tópicos específicos en el broker. Los estudiantes implementarán configuraciones prácticas que les permitan gestionar tanto los recursos energéticos como las comunicaciones, garantizando un funcionamiento eficiente y continuo en un entorno simulado. Se verificará el funcionamiento del sistema mediante la visualización de datos en un cliente MQTT y se evaluará el impacto de las configuraciones energéticas en el rendimiento del sistema.

Materiales

Los materiales tanto software como hardware que se utilizan en esta práctica son los siguientes:

Hardware:

- 1 placa ESP32.
- 1 sensor digital DHT11/DHT22 (temperatura y humedad).
- 1 sensor analógico YL-69 de humedad del suelo.
- 1 sensor analógico de luz LDR.
- 2 resistencias de 10 kΩ.
- 3 LEDs de diferentes colores
- 3 resistencias de 220Ω.
- 1 Protoboard.
- Cables jumper.

Software:

- Arduino IDE (configurado para ESP32).
- Broker MQTT (Mosquitto instalado localmente o en un servidor).
- Node-RED para visualización de datos.
- MQTT Explorer o cualquier cliente MQTT para pruebas.

Opcional:

- Multímetro digital (para verificar conexiones y niveles de voltaje).
- Caja de conexiones o soporte (para organizar el prototipo durante la práctica).

Conexiones del Hardware

Para implementar esta práctica, sigue estas indicaciones para realizar las conexiones correctamente en la placa ESP32 como se muestra en la figura 8:

Conexión del Sensor DHT11/DHT22

- **Pin de alimentación (VCC)**: Conectar al pin **3.3V** de la ESP32.
- **Pin de datos (DATA)**: Conectar a un pin digital de la ESP32 (por ejemplo, el pin GPIO25).
    - Agregar una resistencia de 10 kΩ entre el pin **VCC** y el pin **DATA** para garantizar la comunicación estable del sensor.
- **Pin de tierra (GND)**: Conectar al pin **GND** de la ESP32.

Conexión del Sensor de Humedad del Suelo YL-69

- **Pin de alimentación (VCC)**: Conectar al pin **3.3V** de la ESP32.
- **Pin de datos (SIG)**: Conectar a un pin ADC de la ESP32 (por ejemplo, el pin GPIO15).
- **Pin de tierra (GND)**: Conectar al pin **GND** de la ESP32.

Conexión Sensor Analógico de Luz LDR

- Conectar un extremo del LDR al pin **3.3V** de la ESP32 y el otro a un divisor de voltaje con una resistencia de 1kΩ o de 10kΩ hacia **GND**. El punto medio del divisor va a un pin ADC de la ESP32 (por ejemplo, el pin GPIO13).

Conexión de los LEDs de Alerta

- **Ánodo (+)**:
    - Conectar el terminal positivo de cada LED a un pin digital de la ESP32. Por ejemplo:
        - LED naranja (alerta de temperatura): **GPIO27**.
        - LED azul (alerta de humedad del suelo): **GPIO14**.
        - LED amarillo (alerta oscuridad): **GPIO12**.

- **Cátodo (-)**: Conectar el terminal negativo de cada LED a **GND** de la ESP32 a través de una resistencia de 220 Ω (para limitar la corriente y proteger el LED).

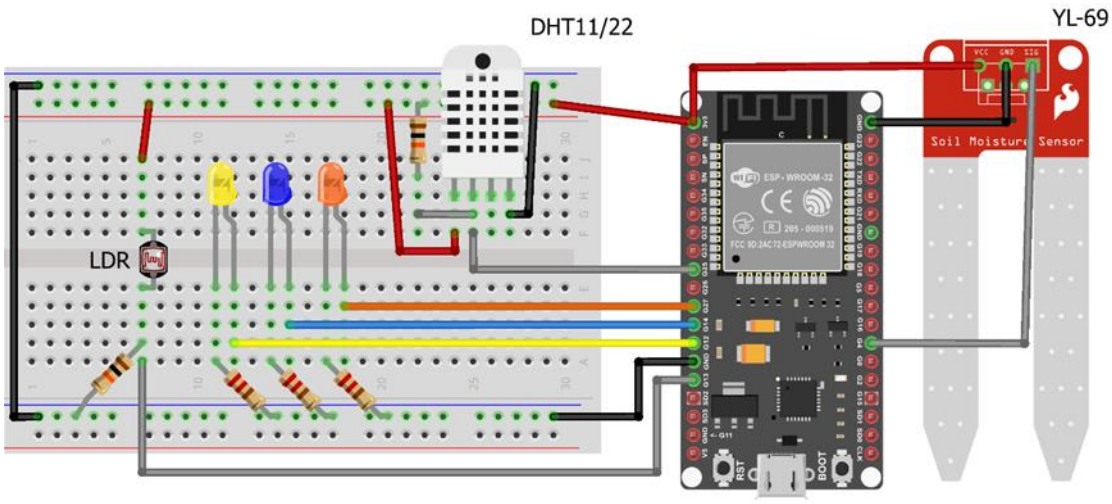
Conexión USB

- Conecta la placa ESP32 al computador mediante un cable **USB micro-B o USB-C** (dependiendo del modelo de la placa). Esto permitirá tanto la programación como la alimentación eléctrica de la placa.

Verificación de Conexiones (Opcional)

- Antes de energizar la ESP32, utiliza un multímetro para verificar la continuidad de las conexiones.
- Asegúrate de que no existan cortocircuitos en la Protoboard y que los pines estén conectados correctamente.

Figura 8
Conexiones de la Práctica 2.



Configuración del Software

Para conseguir los objetivos de la práctica, es necesario desarrollar un código en el IDE de Arduino para cargarlo a la placa ESP32 y esta pueda monitorear la temperatura, humedad relativa, humedad del suelo y luz ambiental. También se debe poder controlar 3 LEDs de alerta de diferentes colores que se encenderán si se supera un umbral específico. Otro aspecto importante es incluir la funcionalidad de Deep Sleep en la ESP32. Este modo

permite ahorrar energía apagando casi todos los módulos de la ESP32, excepto aquellos que se necesiten para despertarla, como el temporizador interno o un pin GPIO. En este caso, configuraremos el Deep Sleep para que la ESP32 se despierte cada cierto tiempo, realice las lecturas de los sensores, publique los datos y luego vuelva al modo de ahorro de energía. Por otro lado, en Node-RED se deben configurar los nodos necesarios para visualizar los datos de los sensores y proporcionar las alertas respectivas.

Código de Arduino

El código completo se encuentra escrito en el Anexo 2, en esta sección se detalla cada parte del código para tener una mejor comprensión de este.

- **Inclusión de Librerías:**
  - Se deben incluir las librerías necesarias, en este caso la librería DHT.h, Wifi.h y PubSubClient.h.

```
// Incluir las librerías necesarias
#include <DHT.h>                    // Librería para el sensor DHT
#include <WiFi.h>                   // Librería para conectividad WiFi
#include <PubSubClient.h>           // Librería para el protocolo MQTT
```

- **Pines, configuraciones y Umbrales**:
  - Se definen los pines a los que están conectados los sensores y los LEDs.
  - Los umbrales de temperatura y humedad de suelo se configuran a 30°C y 20%, respectivamente.
  - Se establece un tiempo de 60 segundos pata el modo Deep Sleep.

```
// Pines,configuraciones y umbrales
#define DHTPIN 25                   // Pin GPIO donde está conectado el DHT
#define DHTTYPE DHT22               // Tipo de sensor (DHT11 o DHT22)
#define SENSOR_SUELO 4              // Pin ADC donde está conectado el sensor de humedad del suelo
#define LDR_PIN 13                  // Pin ADC donde está conectado el sensor LDR
#define LED_TEMP 27                 // GPIO para el LED indicador de temperatura
#define LED_HUM 14                  // GPIO para el LED indicador de humedad
#define LED_LUZ 12                  // GPIO para el LED indicador de luz
#define TEMP_UMBRAL 30.0            // Umbral de temperatura crítica
#define HUM_UMBRAL 20.0             // Umbral de humedad de suelo crítica
#define LUZ_UMBRAL 3000             // Umbral de luz para determinar cuando oscurece
#define FACTOR_uS_S 1000000ULL      // Factor de conversión de microsegundos a segundos
#define TIME_TO_SLEEP 60            // Tiempo de Deep Sleep en segundos
```

- **Configuración de Protocolos:**
  - Para el protocolo Wi-Fi se deben definir el nombre de la red y la contraseña, para que la ESP32 se conecte correctamente.
  - Para el protocolo MQTT se define la dirección IP en la que se encuentra instalado el broker, en este caso se debe buscar que dirección tiene el computador personal. Si se utilizan brokers en la nube escribir ahí la dirección.

```
// Configuración de WiFi
const char* ssid = "Tu_SSID";                // Nombre de la red WiFi
const char* password = "Tu_Contraseña";      // Contraseña de la red WiFi

// Configuración MQTT
const char* mqtt_server = "localhost";       // Dirección IP del broker MQTT
const char* topic_temp = "agri/temp";        // Tópico para temperatura
const char* topic_hum = "agri/hum";          // Tópico para humedad
const char* topic_suelo = "agri/suelo";      // Tópico para humedad del suelo
const char* topic_luz = "agri/luz";          // Tópico para luz ambiental

DHT dht(DHTPIN, DHTTYPE);           // Inicialización del sensor DHT
WiFiClient espClient;               // Cliente WiFi
PubSubClient client(espClient);     // Cliente MQTT
```

- **Configuraciones del setup:**
  - Se inicializa la comunicación serial a 115200 baudios (velocidad compatible con ESP32).
  - Se inicializa el sensor DHT configurado.
  - Se configuran los pines de los sensores analógicos como entrada (INPUT) y los pines de los LEDs como salida (OUTPUT).
  - Se inicializa la comunicación Wi-Fi y se configura el servidor MQTT en el puerto 1883.
  - Se inicializa un método para leer y publicar los datos de los sensores.
  - Se configura el modo Deep Sleep cada 60 segundos (o el tiempo que configures en TIME_TO_SLEEP) para realizar su tarea y vuelve al modo de ahorro de energía automáticamente.

```
void setup() {
  Serial.begin(115200);                // Iniciar comunicación serial
  dht.begin();                         // Iniciar el sensor DHT

  pinMode(SENSOR_SUELO, INPUT);        // Configurar pin del sensor de suelo
  pinMode(LDR_PIN, INPUT);             // Configurar pin del sensor LDR
  pinMode(LED_TEMP, OUTPUT);           // Configurar LED de temperatura como salida
  pinMode(LED_HUM, OUTPUT);            // Configurar LED de humedad como salida
  pinMode(LED_LUZ, OUTPUT);            // Configurar LED de luz como salida

  setupWiFi();                         // Conexión WiFi
  client.setServer(mqtt_server, 1883); // Configuración del servidor MQTT

  // Leer y publicar datos antes de entrar en Deep Sleep
  leerSensoresYPublicar();

  // Configurar Deep Sleep
  Serial.println("Entrando en modo Deep Sleep...");
  esp_sleep_enable_timer_wakeup(TIME_TO_SLEEP * FACTOR_uS_S); // Configurar tiempo de sueño
  esp_deep_sleep_start(); // Entrar en Deep Sleep
}
```

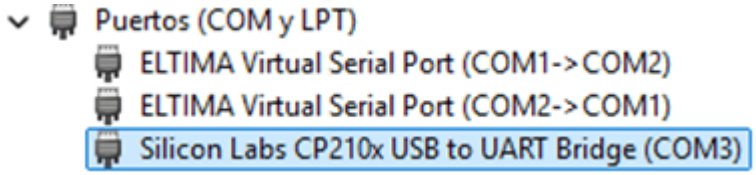
- **Funciones para Wi-Fi y MQTT**
  - Se configuran 2 funciones para la correcta conexión de los protocolos de comunicación.

```
void setupWiFi() {
  // Conexión a la red WiFi
  delay(10);
  Serial.println("Conectando a WiFi...");
  WiFi.begin(ssid, password);      // Iniciar conexión con las credenciales de red
  while (WiFi.status() != WL_CONNECTED) {
    delay(1000);                   // Esperar hasta que se establezca la conexión
    Serial.print(".");             // Imprimir un punto como indicador de progreso
  }
  Serial.println("\nConexión WiFi establecida"); // Confirmar conexión exitosa
}

void reconnect() {
  // Reconexión al broker MQTT en caso de desconexión
  while (!client.connected()) {
    Serial.print("Conectando al broker MQTT...");
    // Intentar conectar al broker MQTT
    if (client.connect("ESP32_Client")) {
      Serial.println("Conectado al broker MQTT"); // Confirmar conexión exitosa
    } else {
      Serial.print("Fallo, rc=");                 // Mostrar código de error
      Serial.print(client.state());
      Serial.println(" Intentando de nuevo en 5 segundos");
      delay(5000);                                // Esperar antes de reintentar
    }
  }
}
```

- **Función para lectura y publicación de datos:**
  - Se configura una función para leer los datos de los sensores y enviarlos mediante tópicos MQTT al broker para su publicación.
  - Dentro de la función también se establece el control de los LEDs de alerta.
  - En el loop no es necesario escribir nada.

```
void leerSensoresYPublicar() {
  if (!client.connected()) {
    reconnect();
  }
  client.loop();  // Mantener conexión MQTT activa

  // Leer sensores
  float temp = dht.readTemperature();        // Temperatura en °C
  float hum = dht.readHumidity();            // Humedad en %
  int hum_suelo = analogRead(SENSOR_SUELO);  // Lectura analógica del sensor de humedad del suelo
  int luz = analogRead(LDR_PIN);             // Lectura analógica del sensor LDR

  // Verificar errores de lectura
  if (isnan(temp) || isnan(hum)) {
    Serial.println("Error al leer el DHT");
    return;
  }

   // Publicar datos en MQTT
   client.publish(topic_temp, String(temp).c_str());       // Publicar temperatura
   client.publish(topic_hum, String(hum).c_str());         // Publicar humedad
   client.publish(topic_suelo, String(hum_suelo).c_str());  // Publicar humedad del suelo
   client.publish(topic_luz, String(luz).c_str());       // Publicar luz ambiental

   // Control de LEDs
   // Encender LED si la temperatura supera el umbral
   digitalWrite(LED_TEMP, temp > TEMP_UMBRAL ? HIGH : LOW);
   // Encender LED si la humedad está por debajo del umbral
   digitalWrite(LED_HUM, hum < HUM_UMBRAL ? HIGH : LOW);

   // Control del LED de luz
   if (luz < LUZ_UMBRAL)            // Si la luz es baja
     digitalWrite(LED_LUZ, HIGH);  // Enciende el LED de luz
    else
     digitalWrite(LED_LUZ, LOW);   // Apaga el LED de luz

   // Mostrar datos en el monitor serial
   Serial.print("Temp: "); Serial.print(temp); Serial.print(" °C, ");
   Serial.print("Hum: "); Serial.print(hum); Serial.print(" %, ");
   Serial.print("Suelo: "); Serial.print(hum_suelo); Serial.print(" , ");
   Serial.print("Luz: "); Serial.println(luz);
}
 void loop() {
   // La ESP32 no ejecutará código en el loop porque entra en Deep Sleep después del setup
 }
```

- **Compilación y Subida del Código:**
  - Compilar el código para comprobar que no existen errores, en el caso de haberlos, revisar bien el código y depurar los errores.
  - Conectar la placa ESP32 a un puerto USB del computador y comprobar que puerto se asigna.

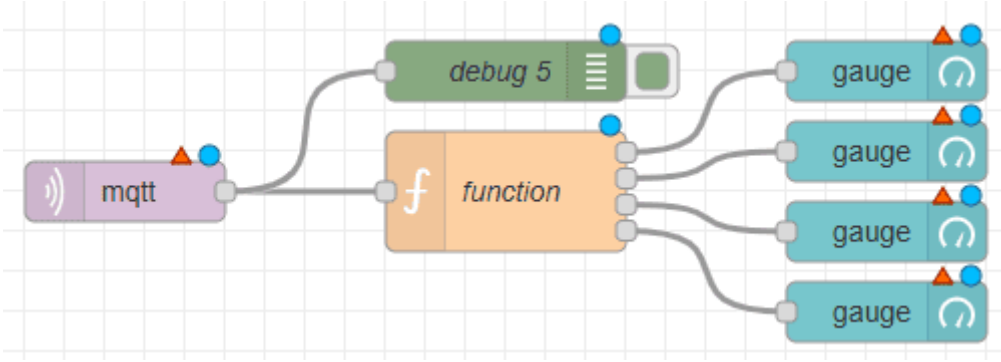


- Seleccionar el puerto específico en el IDE y subimos el código a la ESP32.

Flujo de Node-RED

Para esta práctica no se necesitan instalar nodos extra, debido a que Node-RED ya implementa por defecto los nodos MQTT dentro de la categoría "network". Por lo tanto, se procede con el diseño y configuración del flujo que lo denominaremos "PRACTICA 2", seleccionamos y arrastramos al área de trabajo los nodos: mqtt in, debug, function (con 4 salidas), gauge (4 nodos) y los conectamos como se muestra en la figura 9:

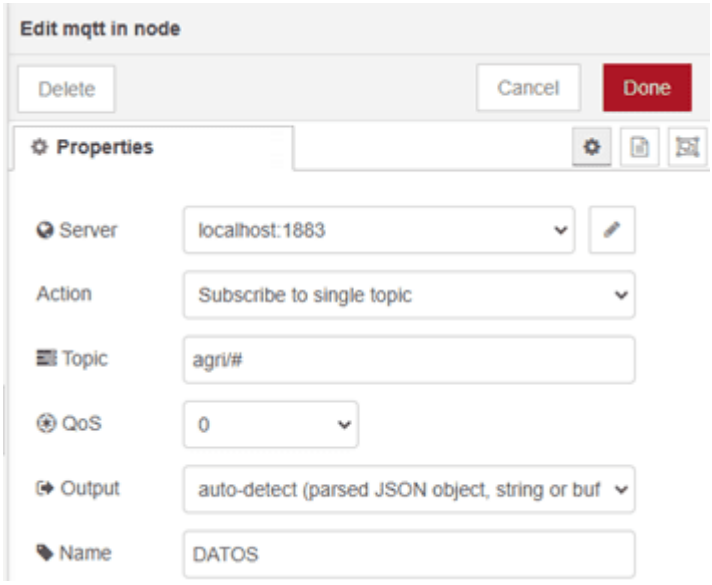
Figura 9
Conexión de nodos en Node-RED para la práctica 2.



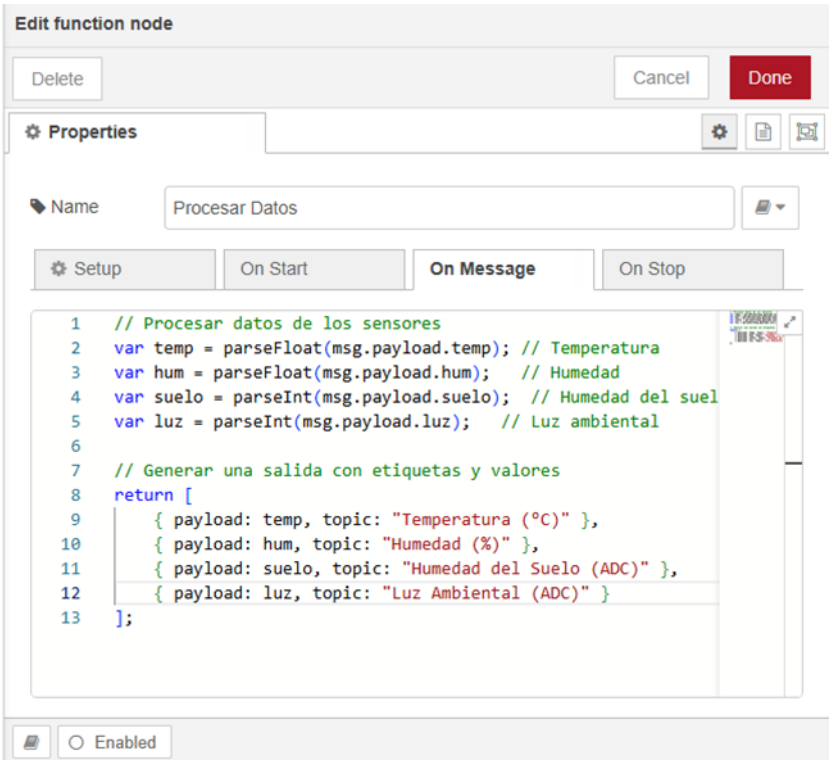
Como aprendimos en la práctica 1, los nodos no están configurados y por eso se marcan las alertas en forma de circunferencias azules y triángulos rojos. Para corregir estos errores se procede con la configuración de cada nodo haciendo doble clic en cada uno y llenando los espacios específicos.

- **Nodo mqtt in:**
  - Haz clic en el lápiz al lado del campo Server para añadir tu broker MQTT.
  - Server: Dirección IP del broker MQTT (por ejemplo, 192.168.1.100), localhost en este caso.

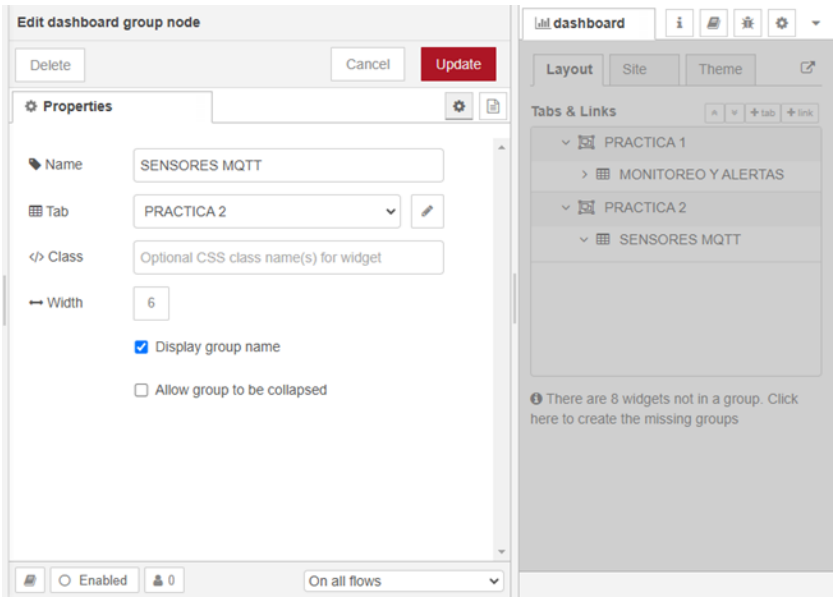
- Port: El puerto MQTT (1883).
- Username/Password: Si configuraste credenciales en el broker, ingrésalas aquí.
- Topic: nos suscribimos a todos los tópicos agri/#
- QoS: Déjalo en 0 (nivel de calidad del servicio más básico).
- Hacer clic a “Done” para finalizar la configuración del nodo.



- **Nodo function:**
  - Este nodo sirve para procesar los datos enviados por la ESP32 y adaptarlos para su visualización en el dashboard.
  - Configuramos un nombre como por ejemplo “Procesar Datos” y en la pestaña “On Message” introducimos el código en JavaScript. Hacer clic a “Done” para finalizar la configuración del nodo.



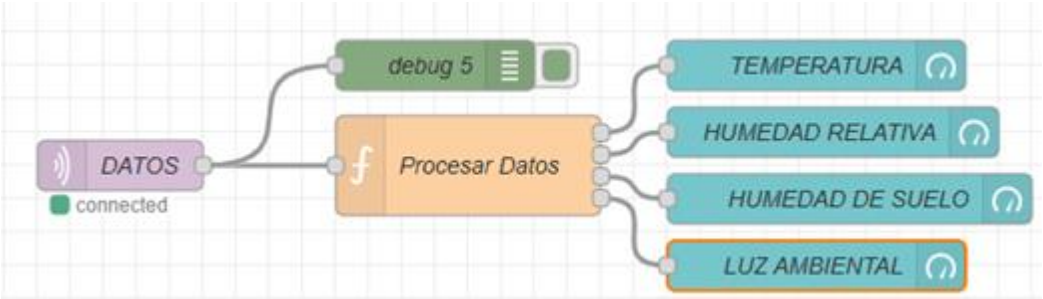
- **Nodos gauge**
  - Siguiendo las instrucciones aprendidas en la práctica 1, configuramos una nueva pestaña denominada PRACTICA 2 y un grupo dentro de esta llamado SENSORES MQTT.



  - Configurar los gauges para cada sensor dentro del nuevo grupo creado y estableciendo rangos con colores distintivos.

Con todos los nodos configurados, se procede ha instanciar la configuración dando clic en el botón Deploy. Se puede notar que todas las alertas han desaparecido y nos quedaría el flujo como se muestra en la figura 10:

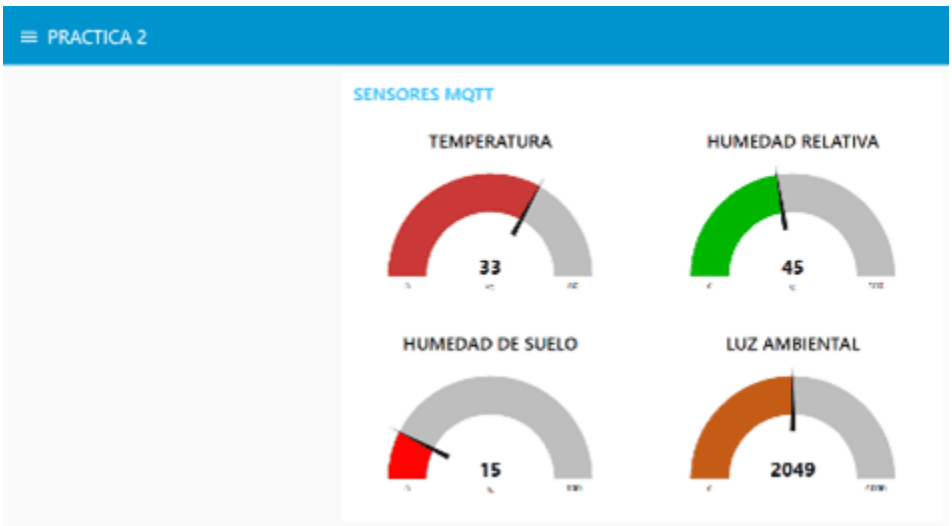
Figura 10
Flujo de la Práctica 2 configurado.



Panel de Visualización

En la figura 11 se muestran los resultados de la configuración en Node-RED para la visualización de los sensores agrícolas conectados, esto es posible gracias a la comunicación MQTT establecida. Los datos se actualizarán cada 60 segundos debido a la implementación del modo Deep Sleep en la placa ESP32.

Figura 11
Monitoreo de variables de agricultura.



Resultados de la Práctica

Los resultados que se deben obtener al aplicar la práctica incluyen una recopilación y monitoreo exitoso de las variables ambientales definidas. En

primer lugar, el sensor DHT22 debe proporcionar lecturas precisas de temperatura y humedad, publicando los datos en los tópicos MQTT correspondientes. Estas mediciones deben visualizarse en Node-RED utilizando indicadores tipo Gauge que muestren la variación de las condiciones en tiempo real. Además, los LEDs de temperatura y humedad deben responder de manera lógica, encendiéndose cuando las mediciones superen o bajen los umbrales configurados, demostrando que el control programado funciona correctamente.

En cuanto al sensor de humedad del suelo, debe entregar valores analógicos que se reflejen adecuadamente en los tópicos MQTT y en Node-RED, permitiendo la visualización de la humedad en tiempo real. El LDR, por su parte, debe detectar cambios en las condiciones de iluminación ambiental y activar el LED correspondiente al oscurecer, verificando que el sistema es capaz de reaccionar a las variaciones de luz con precisión.

Finalmente, al activar el modo Deep Sleep en la ESP32, el sistema debe reducir considerablemente el consumo energético durante los períodos de inactividad, despertándose para realizar lecturas y publicaciones en intervalos predefinidos. Esto asegura que el sistema sea eficiente en términos de energía y viable para aplicaciones en ambientes agrícolas donde la autonomía energética es crucial. Estos resultados demuestran la integración efectiva de hardware y software, confirmando que el sistema cumple con los objetivos planteados en la práctica.

Evaluación del Aprendizaje

El docente calificará todo el proceso y desarrollo de la práctica estableciendo una calificación final sobre 10 puntos en base a la rúbrica presentada en la tabla 4:

Tabla 4
Rúbrica de evaluación.

Criterio de Evaluación	Nivel de Desempeño
Diseño e implementación del circuito	2,5 puntos
Configuración de Node-RED y el broker MQTT	2,5 puntos

Integración del hardware y software para el monitoreo	2,5 puntos
Desarrollo del informe de laboratorio con claridad y que contenga toda la información	2,5 puntos

Actividades Complementarias

Con el objetivo de practicar y mejorar en el control de energía se plantean las siguientes actividades para que el estudiante desarrolle:

- **Nodos Adicionales**
  - Agregar nodos tipo led para mostrar las alertas en el dashboard.
- **Integración de un Sensor Adicional**
  - Añadir sensores adicionales, como sensores de CO_2 o de presión atmosférica, y adaptar el código para incluirlos en el sistema.
  - Modifica el código de Arduino y el flujo de Node-RED para integrar las lecturas de los nuevos sensores, mostrando sus datos en Gauges adicionales y generando alertas si los valores superan un umbral definido.
  - Evaluar cómo afectan estos datos adicionales al monitoreo y control de las variables ambientales.
- **Control de Actuadores Adicionales**
  - Conecta un actuador adicional, como un relé que controle un dispositivo externo para activar el riego automático en caso de que la humedad del suelo esté por debajo del umbral establecido
  - Modifica el flujo en Node-RED para que el actuador se active o desactive en función de las lecturas del sensor o mediante un botón en la interfaz gráfica.

Cuestionario

- ¿Qué es el modo Deep Sleep en la ESP32 y cuál es su importancia en esta práctica?
  a. Explica cómo contribuye al ahorro energético.
  b. Menciona al menos dos configuraciones necesarias para activarlo.

- ¿Qué rol desempeña el LDR en este sistema, y cómo se interpreta su salida analógica en términos de luminosidad?

a. Describe los valores típicos que genera el LDR en condiciones de luz y oscuridad.

b. ¿Qué acciones realiza el sistema cuando detecta oscuridad?

- ¿Qué ventajas ofrece el uso de MQTT en la comunicación de este sistema?

a. Detalla por qué es adecuado para aplicaciones IoT como la diseñada en esta práctica.

b. ¿Qué comando se utiliza para suscribirse a todos los tópicos en Mosquitto?

CAPÍTULO IV

Metodología de despliegue y servicios IoT

ALUMNI EDITORA

Fundamentación

El despliegue de soluciones en IoT [3] sigue varias etapas clave. En la planificación inicial, se definen los objetivos del sistema, los problemas a resolver y los servicios a ofrecer. Esto incluye la selección de hardware como sensores, actuadores y dispositivos como las placas ESP32, además de la identificación de los requisitos de conectividad y protocolos adecuados como WiFi, Bluetooth o LoRa. Es esencial considerar la escalabilidad para futuros crecimientos y la seguridad de los datos desde esta etapa. En la arquitectura del sistema, se diseña la estructura en niveles: el nivel de percepción (sensores y actuadores), el nivel de red (protocolos como MQTT o HTTP), el nivel de procesamiento (computación en la nube o en el borde) y el nivel de aplicación (interfaces para usuarios finales) [14]. También se asegura la interoperabilidad entre los dispositivos y servicios. Durante la implementación, se procede con la instalación física de los componentes, la configuración del software utilizando herramientas como Arduino IDE, Node-RED, InfluxDB y Grafana, y pruebas iniciales para validar la funcionalidad del sistema. Finalmente, en el monitoreo y mantenimiento, se realiza una supervisión continua mediante dashboards, se implementan actualizaciones remotas OTA y se optimiza el rendimiento del sistema según sea necesario.

Los servicios más comunes en IoT incluyen la gestión de datos, comunicación, visualización y seguridad. La gestión de datos abarca la recolección, almacenamiento y análisis de información proveniente de sensores. Para el almacenamiento, se suelen usar bases de datos como InfluxDB o sistemas NoSQL. En cuanto a la comunicación, se emplean protocolos ligeros como MQTT para una transmisión eficiente y sistemas de mensajería mediante brokers como Mosquitto. La visualización y control se realiza a través de dashboards personalizados que permiten monitorear el estado de los dispositivos y configurar automatizaciones. La seguridad es un aspecto crítico y se asegura mediante el uso de cifrado TLS/SSL, autenticación, certificados digitales y firewalls que protegen la red de posibles ataques [15]. En la tabla 5, se describen las aplicaciones en el campo laboral y/o profesional:

Tabla 5
Aplicaciones profesionales de la práctica 3.

Aplicación	Descripción
Agricultura de precisión	Proyectos de monitoreo ambiental en tiempo real, como medir humedad del suelo, temperatura, luz solar y otros parámetros cruciales para optimizar el riego y los ciclos de cultivo. Estas soluciones permiten reducir costos, mejorar los rendimientos y promover prácticas agrícolas sostenibles.
Gestión de edificios inteligentes	Desarrollar sistemas que controlan iluminación, climatización, consumo energético y seguridad. Por ejemplo, se puede diseñar un sistema que ajuste automáticamente las luces en función de la luminosidad ambiental o que controle la temperatura de los espacios en función de las condiciones externas.
Monitoreo industrial y mantenimiento predictivo	Desarrollar soluciones para monitorear máquinas y procesos en tiempo real, detectar fallos antes de que ocurran, y mejorar la eficiencia operativa. Los datos recopilados y visualizados en Grafana pueden ser utilizados para tomar decisiones basadas en información precisa y actualizada.
Salud y bienestar	Monitorización remota de pacientes, la creación de dispositivos portátiles para medir parámetros de salud, o en sistemas de asistencia en el hogar. Estas herramientas facilitan el análisis continuo y seguro de datos críticos.
Redes y telecomunicaciones	Implementar soluciones como monitoreo de tráfico en redes, control remoto de infraestructura tecnológica, y optimización de servicios mediante el análisis de grandes volúmenes de datos.

Objetivos

Objetivo General

Diseñar e implementar un sistema IoT para el control automático de iluminación y temperatura en un entorno simulado de un edificio inteligente.

Objetivos Específicos

Configurar sensores y actuadores en una placa ESP32 para medir luminosidad, temperatura y activar luces y ventiladores según condiciones programadas.

Implementar la comunicación de datos entre dispositivos mediante MQTT.

Almacenar las mediciones en InfluxDB para su análisis.

Crear dashboards interactivos en Grafana para monitorear y controlar los sistemas.

Preparación Previa

La preparación previa para la ejecución de esta práctica requiere comprender los principios fundamentales de IoT, incluyendo la interacción entre hardware y software en un sistema distribuido. Es esencial conocer cómo los microcontroladores como la ESP32 funcionan, sus capacidades de conectividad WiFi, y cómo pueden integrar sensores y actuadores para recopilar y procesar datos. Además, se debe entender el funcionamiento básico de los sensores utilizados, como el DHT22 para medir temperatura y humedad y el LDR para medir luz ambiental, incluyendo sus características técnicas y la forma en que generan datos analógicos o digitales.

Otro aspecto clave es el conocimiento del protocolo MQTT, que permite la comunicación eficiente entre dispositivos IoT. Los estudiantes deben familiarizarse con conceptos como clientes, tópicos y la publicación/suscripción, ya que esto facilitará el flujo de datos desde la ESP32 hasta el broker y su redistribución hacia aplicaciones como Node-RED e InfluxDB. También es importante tener una comprensión básica de los sistemas de bases de datos temporales como InfluxDB, que se emplean para almacenar datos históricos

provenientes de sensores y permiten consultas específicas para el análisis y visualización en tiempo real.

Finalmente, se necesita una base sólida en la gestión de flujos con Node-RED, un entorno gráfico que simplifica la integración de dispositivos, bases de datos y protocolos como MQTT. También es fundamental entender cómo Grafana se utiliza para diseñar dashboards interactivos y personalizables que presentan datos procesados de forma visualmente atractiva y comprensible. La familiaridad con estos conceptos asegura que los estudiantes puedan implementar un sistema IoT completo, desde la recolección de datos hasta su análisis y visualización.

Procedimiento

El procedimiento para implementar el sistema de gestión de iluminación y temperatura en edificios inteligentes inicia con la configuración del hardware. Se conecta un sensor de luz LDR al pin analógico de la ESP32, un sensor de temperatura y humedad DHT22 al pin digital correspondiente, un LED para simular el sistema de iluminación y un ventilador pequeño para el control de temperatura. La ESP32 actúa como el dispositivo principal que gestiona la recolección de datos, el análisis de condiciones y el control de los actuadores. Todo el hardware se alimenta adecuadamente y se configuran los pines en el código para asegurar su correcto funcionamiento. En el software, se programa la ESP32 en Arduino IDE para que lea las mediciones de los sensores, controle el encendido y apagado del LED y el ventilador según los umbrales definidos, y envíe los datos al broker MQTT mediante tópicos.

Posteriormente, se implementa la gestión de datos y visualización. Node-RED se configura como intermediario para procesar y mostrar los datos en tiempo real, permitiendo la supervisión y control manual del sistema. InfluxDB almacena las mediciones históricas de luminosidad y temperatura para su análisis, mientras que Grafana se utiliza para diseñar dashboards interactivos que muestran gráficos en tiempo real y permiten monitorear el rendimiento del sistema. Con este procedimiento, se logra un sistema funcional que simula la automatización de iluminación y control de temperatura en un edificio inteligente, integrando sensores, actuadores y herramientas de software clave para la gestión IoT.

En este procedimiento, el Módulo MOSFET IRF520N actúa como un interruptor electrónico controlado por la ESP32. Este dispositivo permite controlar cargas más grandes, como el ventilador de 5V, mediante una señal de bajo voltaje proveniente de la ESP32. El MOSFET se conecta entre la fuente de alimentación del ventilador y el ventilador mismo. Cuando la ESP32 envía una señal de "encendido" al pin SIG del MOSFET, este permite que fluya corriente desde la fuente de 5V hacia el ventilador, activándolo. Por el contrario, cuando la señal es "apagado", el MOSFET interrumpe el flujo de corriente, desactivando el ventilador. Este control eficiente de la corriente permite que el ventilador funcione sin sobrecargar la ESP32, que solo se encarga de enviar la señal de control al MOSFET.

Materiales

Los materiales tanto software como hardware que se utilizan en esta práctica son los siguientes:

Hardware:

- 1 placa ESP32.
- 1 sensor digital DHT11/DHT22 (temperatura y humedad).
- 1 sensor analógico de luz LDR.
- LED para iluminación.
- Ventilador de 5V.
- Módulo IRF520N comercial (MOSFET para el control del ventilador).
- Fuente externa de 5V.
- 1 resistencia de 10 kΩ.
- 1 resistencia de 220Ω.
- 1 Protoboard.
- Cables jumper.

Software:

- Arduino IDE (configurado para ESP32).
- Broker MQTT (Mosquitto instalado localmente o en un servidor).
- Node-RED para visualización de datos.
- InfluxDB para base de datos.
- Grafana para diseño de dashboards.

Opcional:

- Multímetro digital (para verificar conexiones y niveles de voltaje).
- Caja de conexiones o soporte (para organizar el prototipo durante la práctica).
- Fuente de 12V (Para ventiladores de mayor voltaje).

Conexiones del Hardware

Para implementar esta práctica, sigue estas indicaciones para realizar las conexiones correctamente en la placa ESP32 como se muestra en la figura 13:

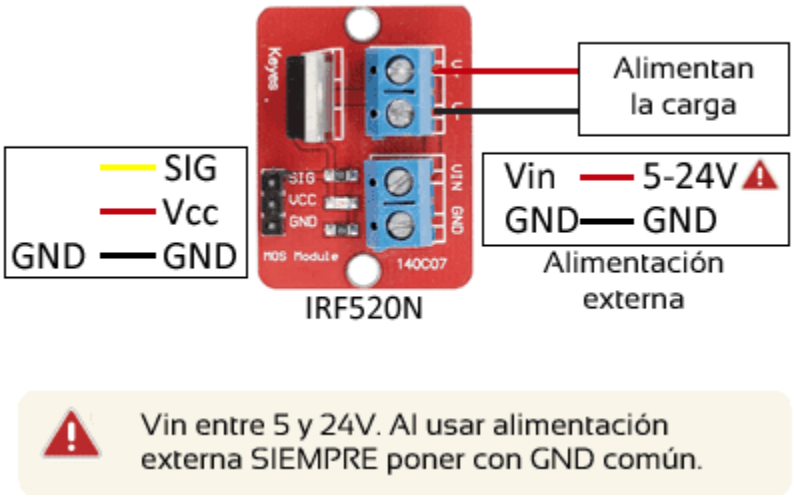
Conexión del Sensor DHT11/DHT22

- **Pin de alimentación (VCC)**: Conectar al pin **3.3V** de la ESP32.
- **Pin de datos (DATA)**: Conectar a un pin digital de la ESP32 (por ejemplo, el pin GPIO25).
    - Agregar una resistencia de 10 kΩ entre el pin **VCC** y el pin **DATA** para garantizar la comunicación estable del sensor.
- **Pin de tierra (GND)**: Conectar al pin **GND** de la ESP32.

Módulo IRF520N (para el ventilador de 12V)

- **Pin de alimentación (VCC)**: Conectar al pin **3.3V** de la ESP32.
- **Pin de datos SIG (Gate del MOSFET)**: Conectar a un pin digital de la ESP32 (por ejemplo, el pin GPIO13).
- **Pin de tierra (GND)**: Conectar al pin **GND** de la ESP32.
- **Entrada V+**: Conectar al pin positivo del ventilador.
- **Entrada V-**: Conectar al pin negativo del ventilador.
- **Entrada VIN**: Conectar el cable positivo de la fuente externa de 5V.
- **Entrada GND**: Conectar el cable negativo de la fuente externa de 5V.
- Seguir las indicaciones de la figura 12**.**

Figura 12
Conexiones del módulo IRF520N.



Conexión Sensor Analógico de Luz LDR

- Conectar un extremo del LDR al pin **3.3V** de la ESP32 y el otro a un divisor de voltaje con una resistencia de 1kΩ o de 10kΩ hacia **GND**. El punto medio del divisor va a un pin ADC de la ESP32 (por ejemplo, el pin GPIO14).

Conexión del LED

- **Ánodo (+)**:
  - Conectar el terminal positivo del LED a un pin digital de la ESP32 (por ejemplo, el pin GPIO27).
- **Cátodo (-)**: Conectar el terminal negativo del LED a **GND** de la ESP32 a través de una resistencia de 220 Ω (para limitar la corriente y proteger el LED).

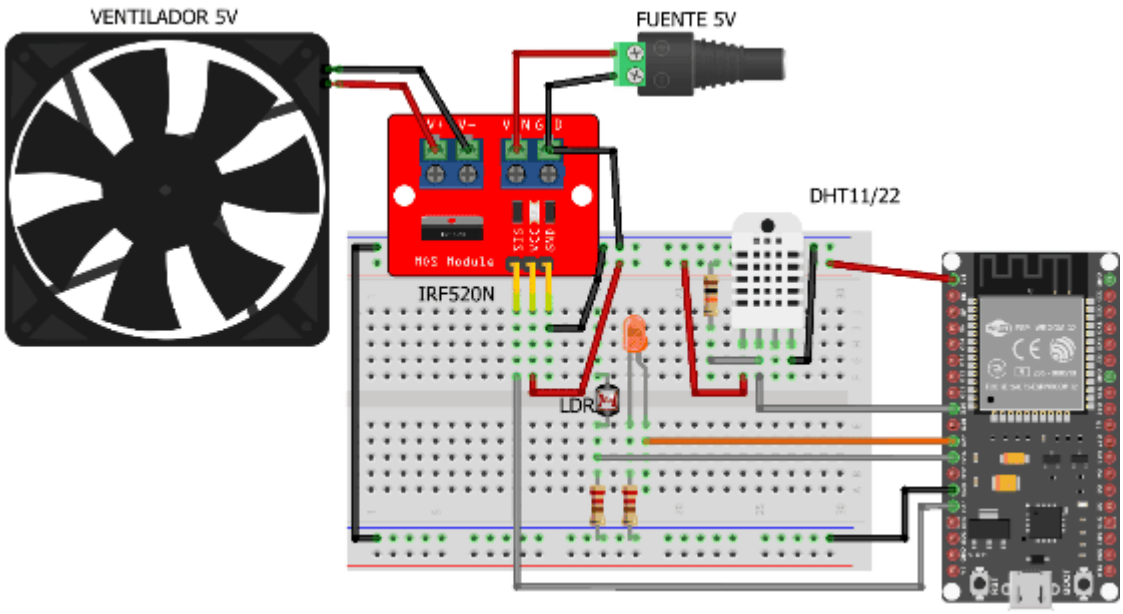
Conexión USB

- Conecta la placa ESP32 al computador mediante un cable **USB micro-B o USB-C** (dependiendo del modelo de la placa). Esto permitirá tanto la programación como la alimentación eléctrica de la placa.

Verificación de Conexiones (Opcional)

- Antes de energizar la ESP32, utiliza un multímetro para verificar la continuidad de las conexiones.
- Asegúrate de que no existan cortocircuitos en la Protoboard y que los pines estén conectados correctamente.

Figura 13
Conexiones de la Práctica 2.



Configuración del Software

Para que el sistema funcione correctamente, se deben hacer algunas configuraciones en el software que integran todos los componentes del proyecto, como el Wi-Fi, MQTT, y las herramientas de monitoreo como Node-RED y Grafana.

Primero, es necesario configurar la red Wi-Fi en el código del ESP32, donde se deben ingresar las credenciales correctas de la red, como el nombre de la red (SSID) y la contraseña, para que el dispositivo pueda conectarse a la red. Esto es fundamental para que el ESP32 pueda enviar los datos de los sensores al servidor MQTT. Posteriormente, en la configuración de MQTT, se debe indicar la dirección IP del broker MQTT, que es el servidor encargado de recibir y distribuir los datos entre los dispositivos conectados. En el código, se definen los tópicos

MQTT, como `home/temp` para la temperatura y `home/luz` para la luz ambiental, a través de los cuales el ESP32 publicará los datos.

Para gestionar y visualizar los datos, se utilizan herramientas como Node-RED y Grafana. En Node-RED, se deben crear flujos que reciban los datos del broker MQTT y los procesen según sea necesario. Node-RED también permite gestionar la lógica de control, como activar o desactivar el ventilador según los umbrales de temperatura. Por otro lado, en InfluxDB se almacenan los datos enviados por los sensores para que se puedan analizar posteriormente. Finalmente, en Grafana, se deben configurar paneles de control (dashboards) para visualizar los datos en tiempo real, permitiendo monitorear la temperatura y la luz ambiental, además de activar alarmas o notificaciones cuando se sobrepasen los umbrales configurados. Estas configuraciones garantizan que el sistema opere de manera eficiente y que se pueda monitorear de forma remota.

Código de Arduino

El código completo se encuentra escrito en el Anexo 3, en esta sección se detalla cada parte del código para tener una mejor comprensión de este.

- **Inclusión de Librerías:**
  - Se deben incluir las librerías necesarias, en este caso la librería DHT.h, Wifi.h y PubSubClient.h.

```
// Incluir las librerías necesarias
#include <DHT.h>                  // Librería para el sensor DHT
#include <WiFi.h>                 // Librería para conectividad WiFi
#include <PubSubClient.h>         // Librería para el protocolo MQTT
```

- **Pines, configuraciones y Umbrales**:
  - Se definen los pines a los que están conectados los sensores, el MOSFET y el LED.
  - Los umbrales de temperatura e iluminación se configuran a 30°C y 3000 respectivamente.

```
// Pines, configuraciones y umbrales
#define DHTPIN 25          // Pin GPIO donde está conectado el DHT
#define DHTTYPE DHT22      // Tipo de sensor (DHT11 o DHT22)
#define LDR_PIN 14         // Pin ADC donde está conectado el sensor LDR
#define LED_LUZ 17         // GPIO para el LED indicador de luz
#define PIN_VENT 13        // Pin GPIO para controlar el MOSFET que activa el ventilador
#define TEMP_UMBRAL 30.0   // Umbral de temperatura crítica para activar el ventilador
#define LUZ_UMBRAL 3000    // Umbral de luz (LDR) para encender el LED
```

- **Configuración de Protocolos:**
  - Para el protocolo Wi-Fi se deben definir el nombre de la red y la contraseña, para que la ESP32 se conecte correctamente.
  - Para el protocolo MQTT se define la dirección IP en la que se encuentra instalado el broker, en este caso se debe buscar que dirección tiene el computador personal. Si se utilizan brokers en la nube escribir ahí la dirección.

```
// Configuración de WiFi
const char* ssid = "Tu_SSID";              // Nombre de la red WiFi
const char* password = "Tu_Contraseña";    // Contraseña de la red WiFi

// Configuración MQTT
const char* mqtt_server = "localhost";     // Dirección IP del broker MQTT
const char* topic_temp = "home/temp";      // Tópico para temperatura
const char* topic_light = "home/luz";      // Tópico para luz ambiental

DHT dht(DHTPIN, DHTTYPE);          // Inicialización del sensor DHT
WiFiClient espClient;              // Cliente WiFi
PubSubClient client(espClient);    // Cliente MQTT
```

- **Configuraciones del setup:**
  - Se inicializa la comunicación serial a 115200 baudios (velocidad compatible con ESP32).
  - Se inicializa el sensor DHT configurado.
  - Se configuran los pines de los sensores analógicos como entrada (INPUT) y los pines del LED y del MOSFET como salida (OUTPUT).
  - Se inicializa la comunicación Wi-Fi y se configura el servidor MQTT en el puerto 1883.

```
// Configuración de WiFi
const char* ssid = "Tu_SSID";              // Nombre de la red WiFi
const char* password = "Tu_Contraseña";    // Contraseña de la red WiFi

// Configuración MQTT
const char* mqtt_server = "localhost";     // Dirección IP del broker MQTT
const char* topic_temp = "home/temp";      // Tópico para temperatura
const char* topic_luz = "home/luz";        // Tópico para luz ambiental

DHT dht(DHTPIN, DHTTYPE);          // Inicialización del sensor DHT
WiFiClient espClient;              // Cliente WiFi
PubSubClient client(espClient);    // Cliente MQTT
```

- **Funciones para Wi-Fi y MQTT**

- Se configuran 2 funciones para la correcta conexión de los protocolos de comunicación.

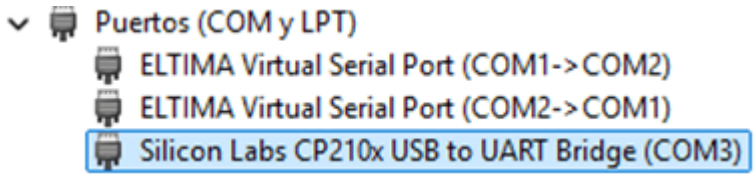
```
void setupWiFi() {
  // Conexión a la red WiFi
  delay(10);
  Serial.println("Conectando a WiFi...");
  WiFi.begin(ssid, password);      // Iniciar conexión con las credenciales de red
  while (WiFi.status() != WL_CONNECTED) {
    delay(1000);                   // Esperar hasta que se establezca la conexión
    Serial.print(".");             // Imprimir un punto como indicador de progreso
  }
  Serial.println("\nConexión WiFi establecida"); // Confirmar conexión exitosa
}

void reconnect() {
  // Reconexión al broker MQTT en caso de desconexión
  while (!client.connected()) {
    Serial.print("Conectando al broker MQTT...");
    // Intentar conectar al broker MQTT
    if (client.connect("ESP32_Client")) {
      Serial.println("Conectado al broker MQTT"); // Confirmar conexión exitosa
    } else {
      Serial.print("Fallo, rc=");                 // Mostrar código de error
      Serial.print(client.state());
      Serial.println(" Intentando de nuevo en 5 segundos");
      delay(5000);                                // Esperar antes de reintentar
    }
  }
}
```

- **Configuración del loop:**
  - Se establece la conexión MQTT.
  - Se realiza la lectura de los sensores.
  - Se configura el control de los actuadores (Ventilador y LED).
  - Se publican los datos en el broker MQTT.
  - Se muestran los datos en el monitor serial

```
void loop() {
  if (!client.connected()) {
    reconnect();
  }
  client.loop(); // Mantener conexión MQTT activa

  // Leer sensores
  float temp = dht.readTemperature(); // Temperatura en °C
  float luz = analogRead(LDR_PIN);    // Lectura analógica del sensor LDR

  // Verificar errores de lectura
  if (isnan(temp)) {
    Serial.println("Error al leer el DHT");
    return;
  }
```

```
  // Control de ventilador
  if (temp > TEMP_UMBRAL)
    digitalWrite(PIN_VENT, HIGH); // Activar el ventilador si la temperatura supera el umbral
    else
    digitalWrite(PIN_VENT, LOW);  // Desactivar el ventilador si la temperatura está por debajo del umbral

  // Control de LED para luz
  if (light < LUZ_UMBRAL)
    digitalWrite(LED_LUZ, HIGH); // Activar el LED de luz si la iluminación es baja
    else
    digitalWrite(LED_LUZ, LOW);  // Desactivar el LED de luz si la iluminación es suficiente

  // Publicar datos en MQTT
  client.publish(topic_temp, String(temp).c_str());  // Publicar temperatura
  client.publish(topic_luz, String(luz).c_str()); // Publicar luz ambiental

  // Mostrar datos en el monitor serial
  Serial.print("Temp: "); Serial.print(temp); Serial.print(" °C, ");
  Serial.print("Light: "); Serial.println(light);

  delay(5000); // Intervalo de 5 segundos
}
```

- **Compilación y Subida del Código:**
  - Compilar el código para comprobar que no existen errores, en el caso de haberlos, revisar bien el código y depurar los errores.
  - Conectar la placa ESP32 a un puerto USB del computador y comprobar que puerto se asigna.

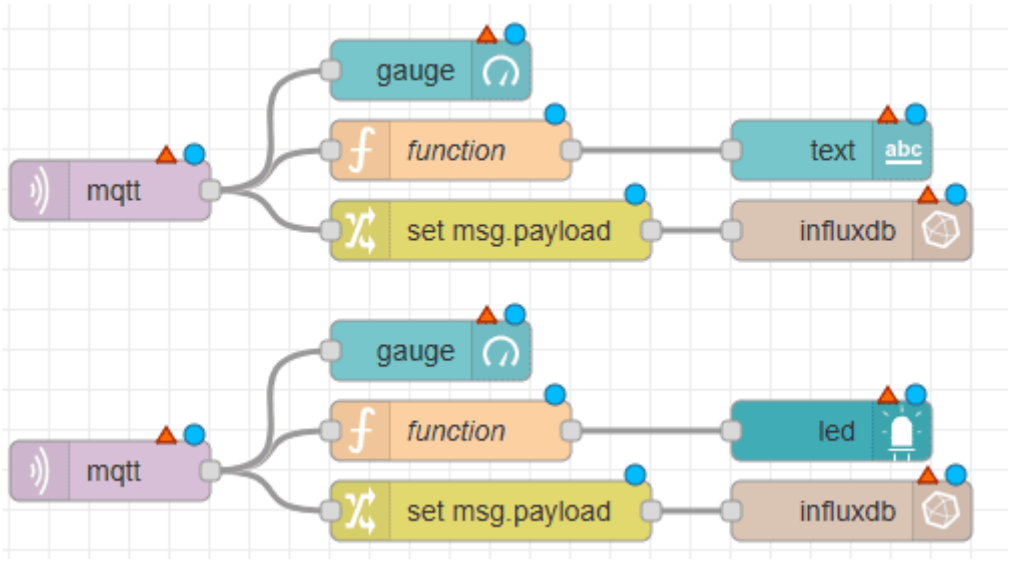


  - Seleccionar el puerto específico en el IDE y subimos el código a la ESP32.

Flujo de Node-RED

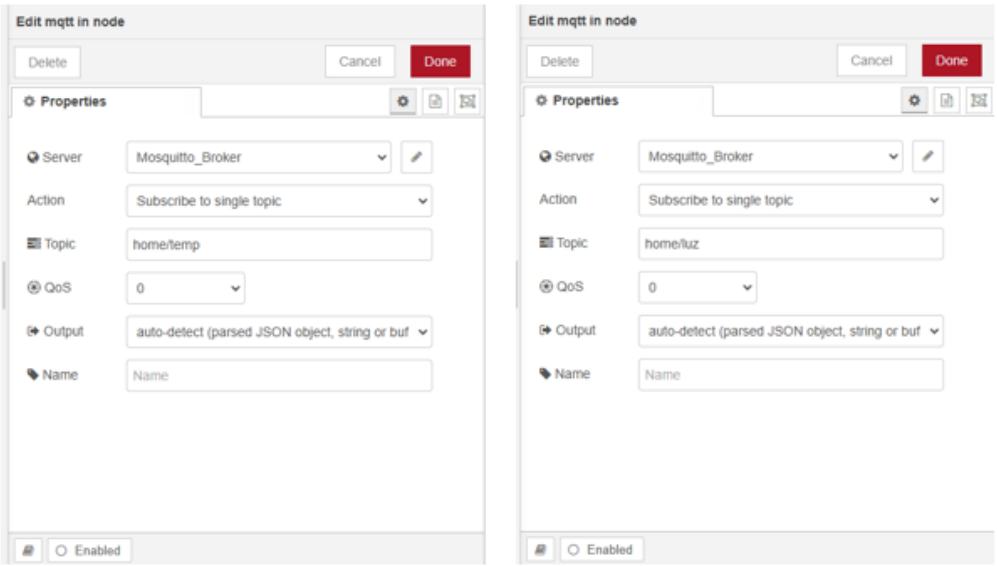
Para esta práctica se necesita instalar los nodos para la conexión con InfluxDB, nos dirigimos a la sección de configuración ≡ y en la opción “manage palette” (administrar paleta) seleccionar la pestaña “Install” (instalar), para buscar e instalar el nodo “node-red-contrib-influxdb”. Una vez instalados los nodos necesarios, se procede con el diseño y configuración del flujo que lo denominaremos “PRACTICA 3”, seleccionamos y arrastramos al área de trabajo los nodos necesarios y los conectamos como se muestra en la figura 14:

Figura 14
Conexión de nodos en Node-RED para la práctica 3.

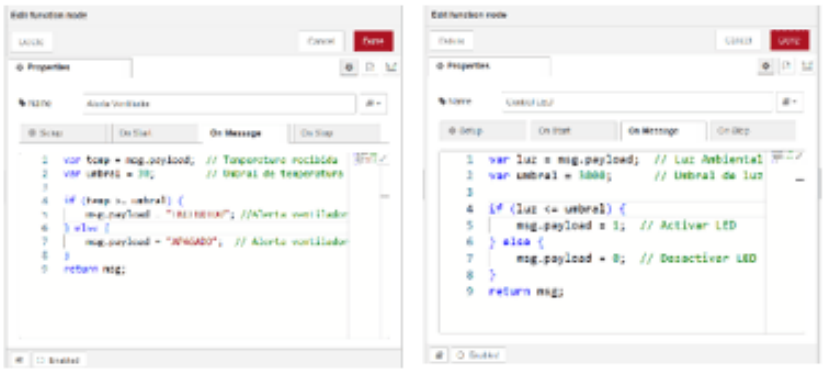


Se utilizan 2 nodos mqtt in para facilitar la configuración, ya que cada nodo se suscribirá a un tópico para mostrar los datos de cada sensor. Los nodos function se utilizan para mostrar de manera correcta el estado de los actuadores (ventilador y LED).

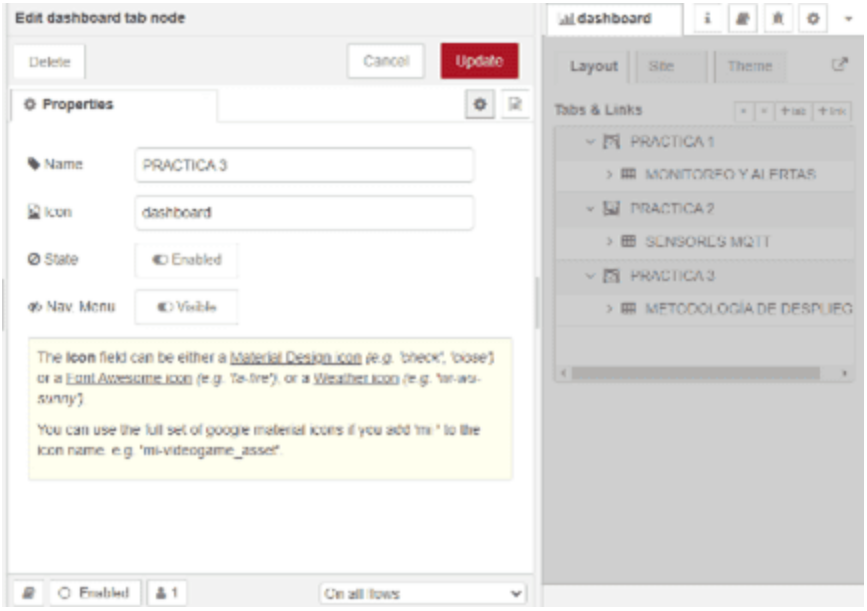
- **Nodos mqtt in:**
  - Debido a que ya creamos el broker en la práctica 2, solo lo seleccionamos y en cada nodo mqtt in nos suscribimos a un tópico:



- **Nodos function:**
  - Se configuran los 2 nodos function para procesar los datos provenientes de la ESP y poder controlar las alertas del Ventilador y del LED.
  - Cada nodo tendrá su propia configuración en JavaScript:

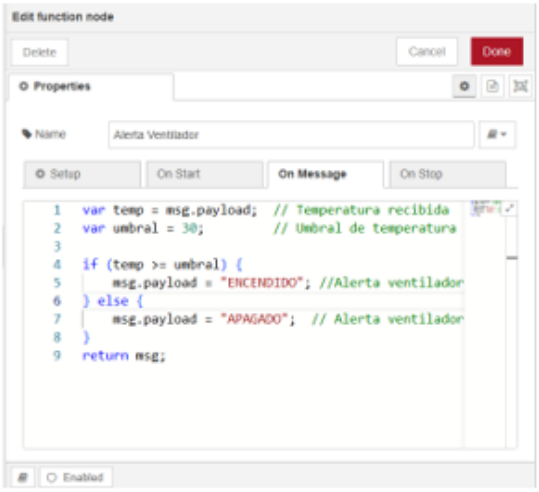


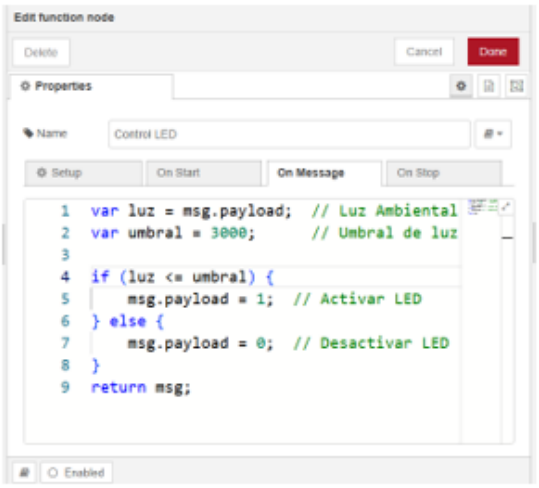
- **Nodos gauge**
  - Siguiendo las instrucciones aprendidas en la práctica 1, configuramos una nueva pestaña denominada PRACTICA 3 y un grupo dentro de esta llamado METODOLOGIA DE DESPLIEGUE.



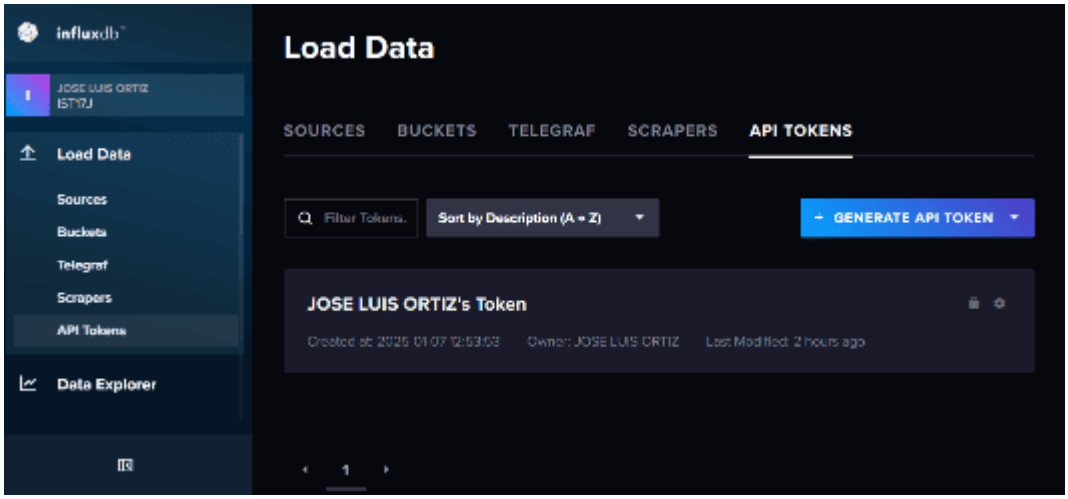
  - Configurar los gauges para cada sensor dentro del nuevo grupo creado y estableciendo rangos con colores distintivos.

- **Nodos de alerta**
    - Configuramos el nodo LED para que se encienda cuando reciba un “1” y se apague cuando reciba un “0”.
    - El nodo text solo recibe la información del nodo “Alerta Ventilador” e imprime el estado del ventilador.

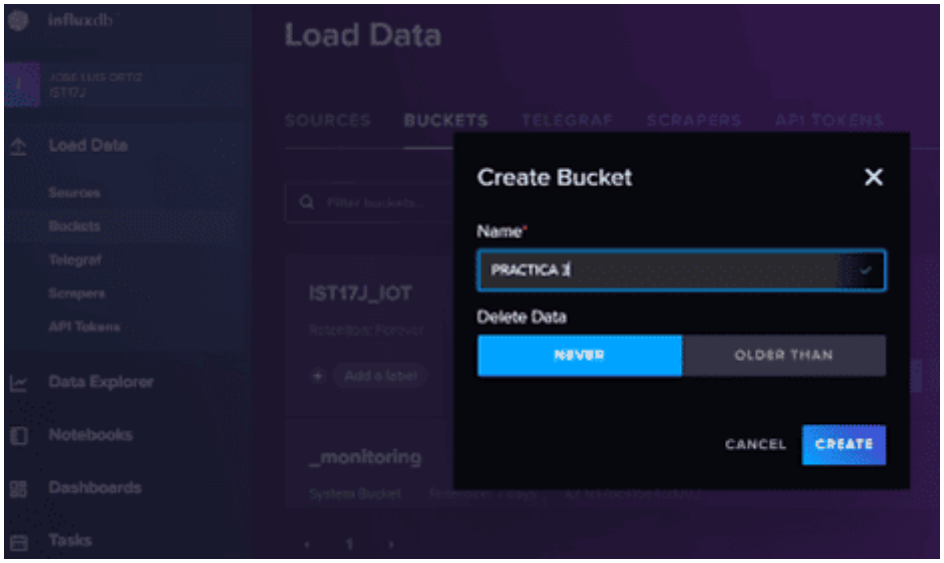




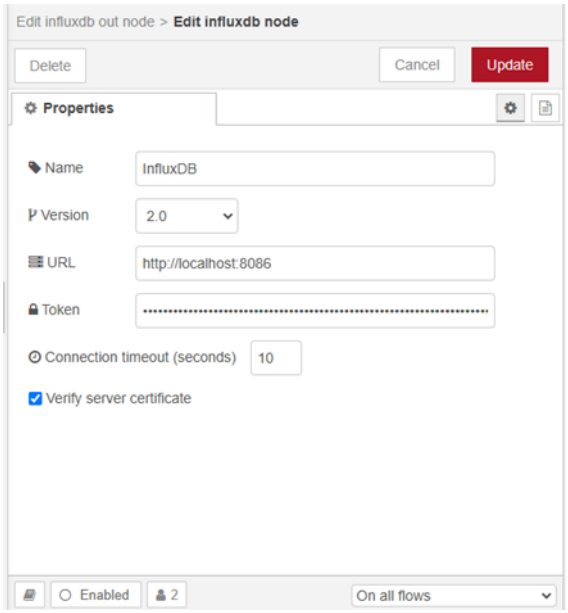
- **Nodos influxdb in**
    - Para la configuración de estos nodos necesitamos configurar la interfaz de InfluxDB para generar los Tokens necesarios y crear los buckets en donde se almacenarán los datos. Para lo cual ingresamos a InfluxDB mediante un navegador y la dirección: http://localhost:8086/
    - Una vez dentro de la interfaz, nos dirigimos al panel de configuraciones ubicado en la parte izquierda y dentro de la pestaña Load Data seleccionamos API Tokens:



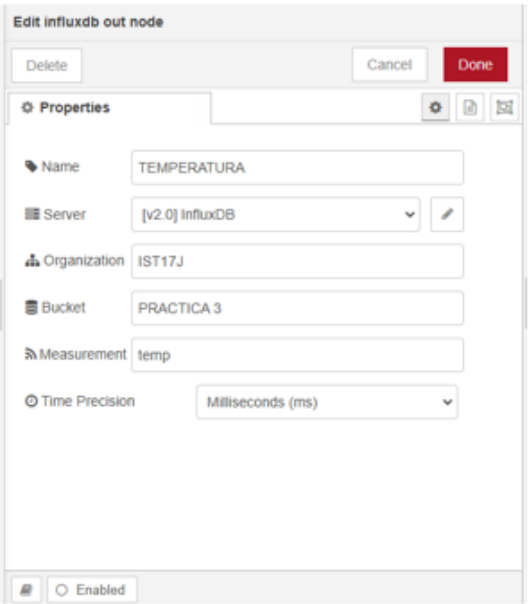
- Generamos un nuevo Token y nos aseguramos de copiarlo y guardarlo en una ubicación conocida (por ejemplo un bloc de notas), en la descripción colocamos un nombre de referencia como Práctica3.
- Ahora nos dirigimos a la sección BUCKETS y creamos un nuevo Bucket con el nombre PRACTICA 3:

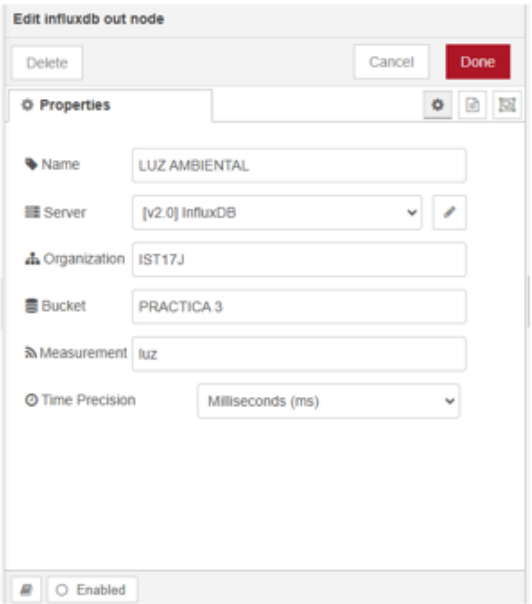


- Con el Token generado y el Bucket creado podemos configurar los nodos en Node-RED, para lo cual agregamos el servidor de nombre InfluxDB, versión 2.0 y ubicado en la dirección IP respectiva, en este caso localhost, el puerto es el 8086, copiamos el Token generado y guardamos presionando en el botón UPDATE:



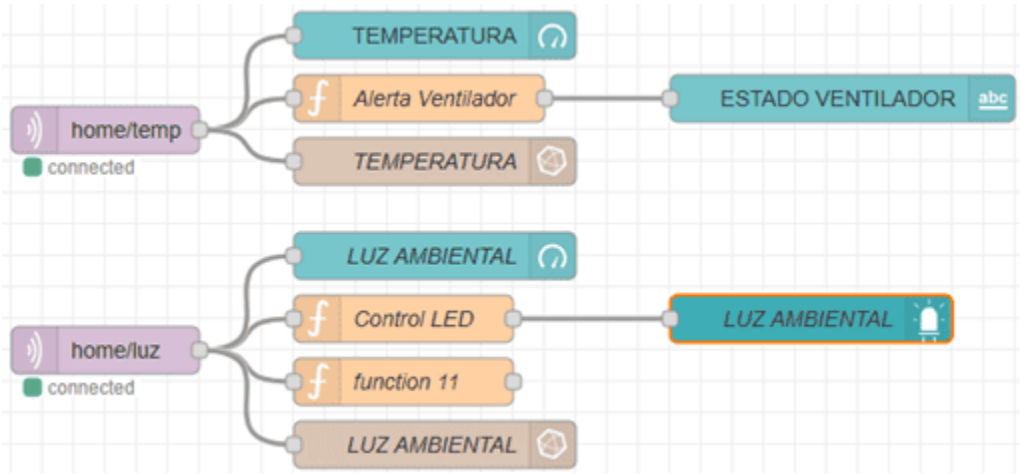
- Ahora se desplegarán las opciones del nodo y lo configuramos conforme los datos establecidos en InfluxDB:





Con todos los nodos configurados, se procede a instanciar la configuración dando clic en el botón Deploy. Se puede notar que todas las alertas han desaparecido y nos quedaría el flujo como se muestra en la figura 15:

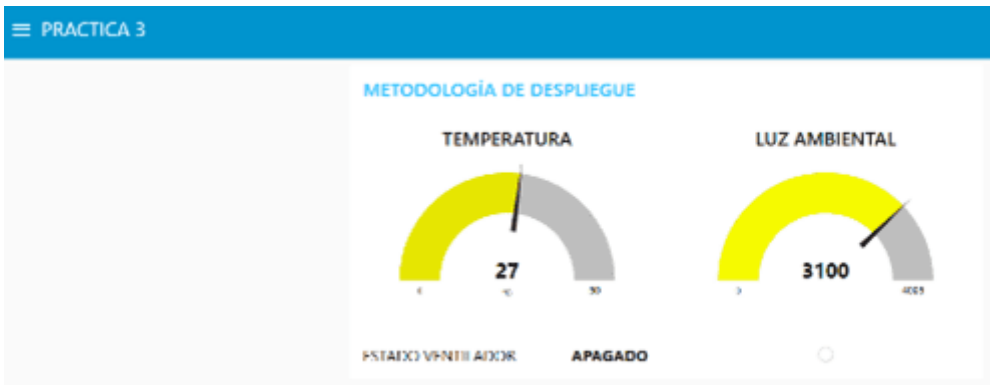
Figura 15
Flujo de la Práctica 3 configurado.



Panel de Visualización

En la figura 16 se muestran los resultados de la configuración en Node-RED para la visualización de los sensores y actuadores configurados, esto es posible gracias a la comunicación MQTT establecida. Se puede observar que tanto la temperatura como la luz ambiental están por debajo de los umbrales establecidos, por lo que el ventilador y el LED están apagados.

Figura 16
Monitoreo de variables de agricultura.

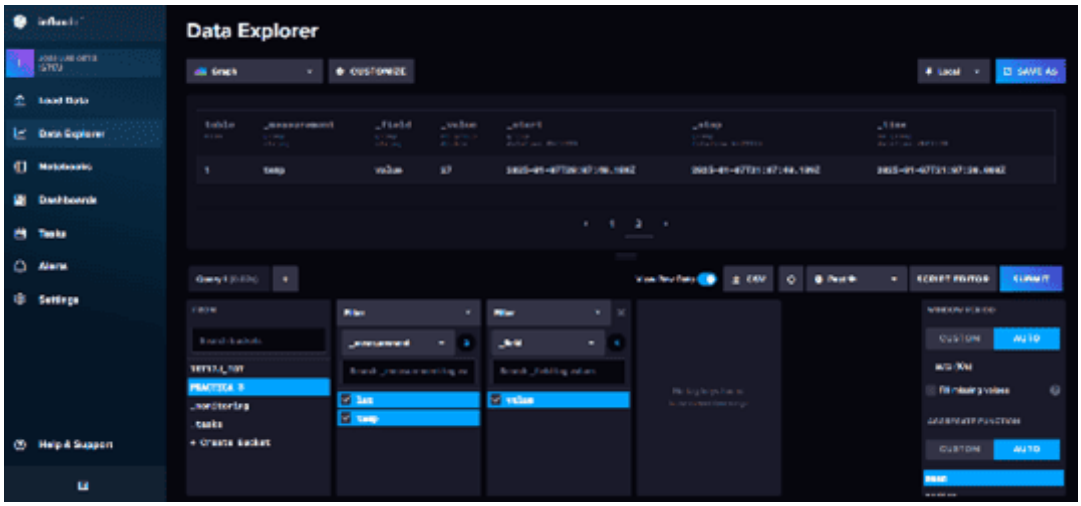


Interfaz de InfluxDB

Dentro de InfluxDB nos dirigimos al panel de control y seleccionamos la pestaña Data Explorer, nos ubicamos en el Bucket PRACTICA 3 y observamos

que ya aparecen las variables temp y luz, las cuales se enviaron desde Node-RED. Para poder visualizarlas, las seleccionamos y presionamos el botón SUBMIT. Como se muestra en la figura 17, InfluxDB permite visualizar los datos en crudo o descargarlos en formato CSV para su posterior análisis. También es posible crear Dashboards pero esa no es la función principal de esta aplicación.

Figura 17
Almacenamiento de datos en InfluxDB

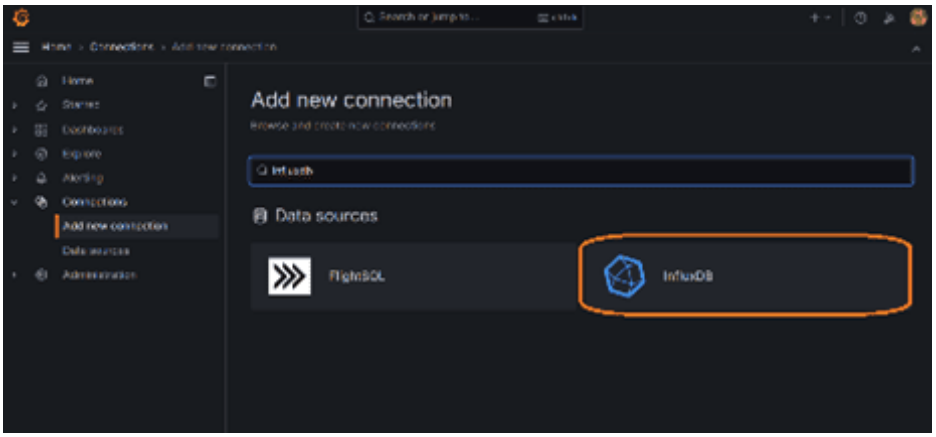


Dashboards en Grafana

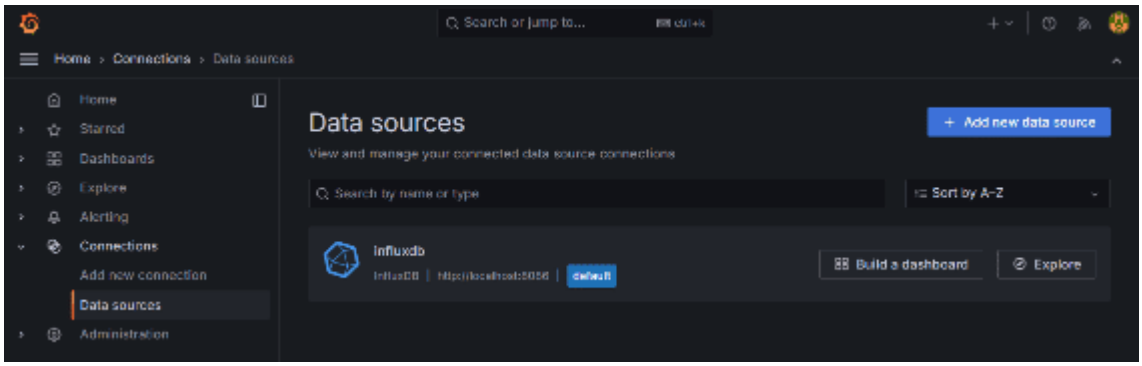
Para poder diseñar dashboards personalizados dependiendo de la aplicación y del cliente la mejor opción es Grafana, debido a que muestra los datos en tiempo real, permite la vinculación desde InfluxDB para observar los datos almacenados y no se vincula directamente con el sistema de gestión que es Node-RED, brindando independencia al administrador del sistema. Para ingresar en la interfaz de Grafana, lo hacemos mediante un navegador y la dirección: http://localhost:3000/

Después de realizar las configuraciones iniciales de registro, nos dirigimos al panel de control ubicado en la parte izquierda y en la pestaña “Connections” seleccionamos la opción “Add new connection”. En la barra de búsqueda escribimos InfluxDB y seleccionamos el API correspondiente como se muestra en la figura 18:

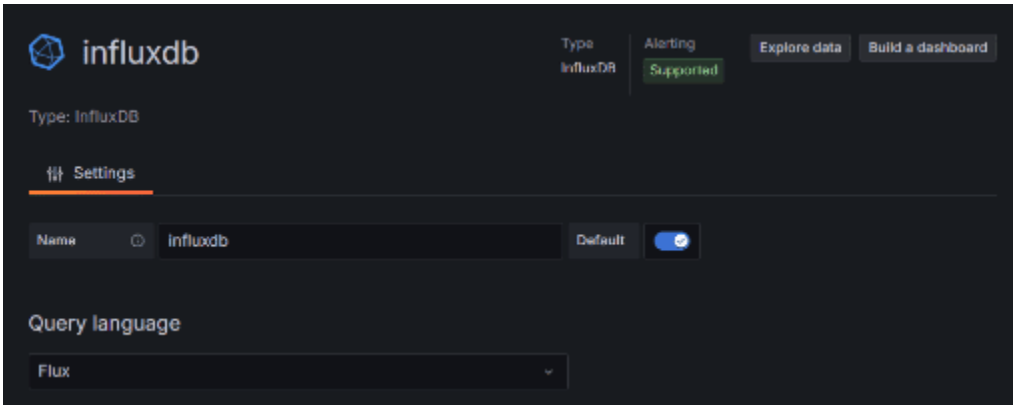
Figura 18
Selección de InfluxDB para monitoreo de datos en Grafana



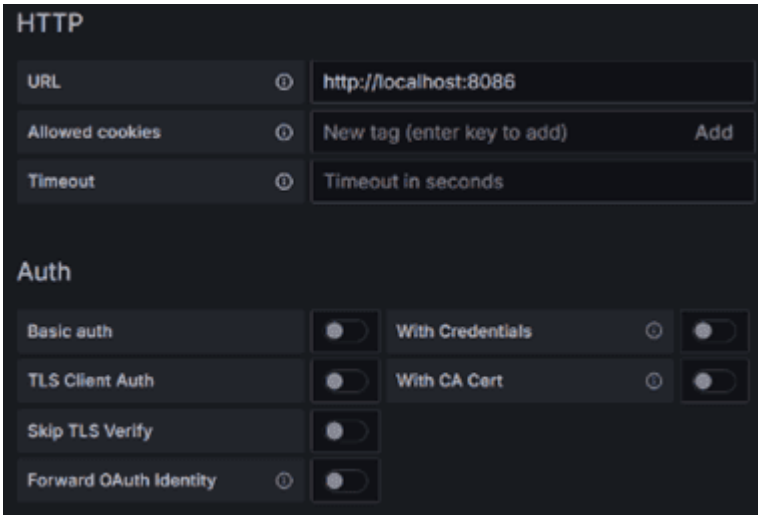
- **Fuente de datos**
  - Una vez seleccionado InfluxDB como conexión se puede observar que aparece como fuente de datos en la opción “Data sources”



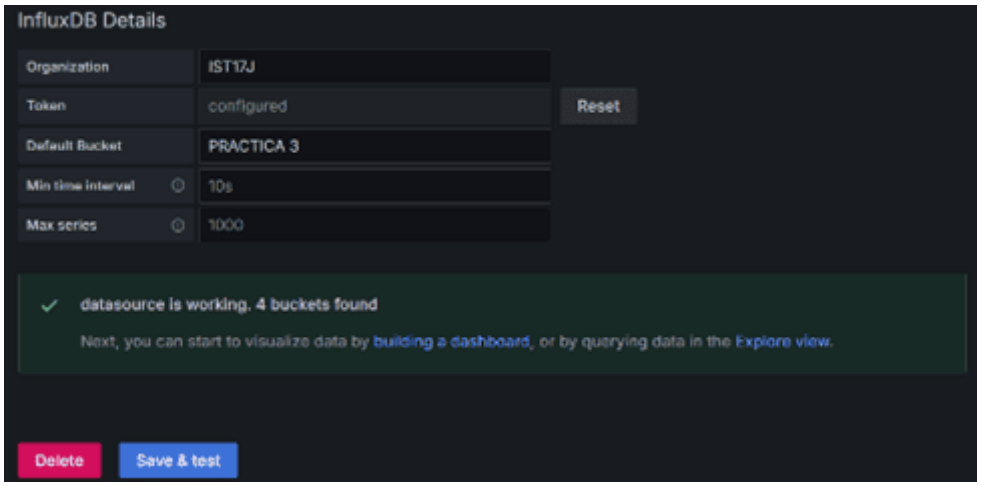
  - Hacemos clic en esta opción para proceder con la configuración de entrada de datos, seleccionando como Query language la opción Flux:



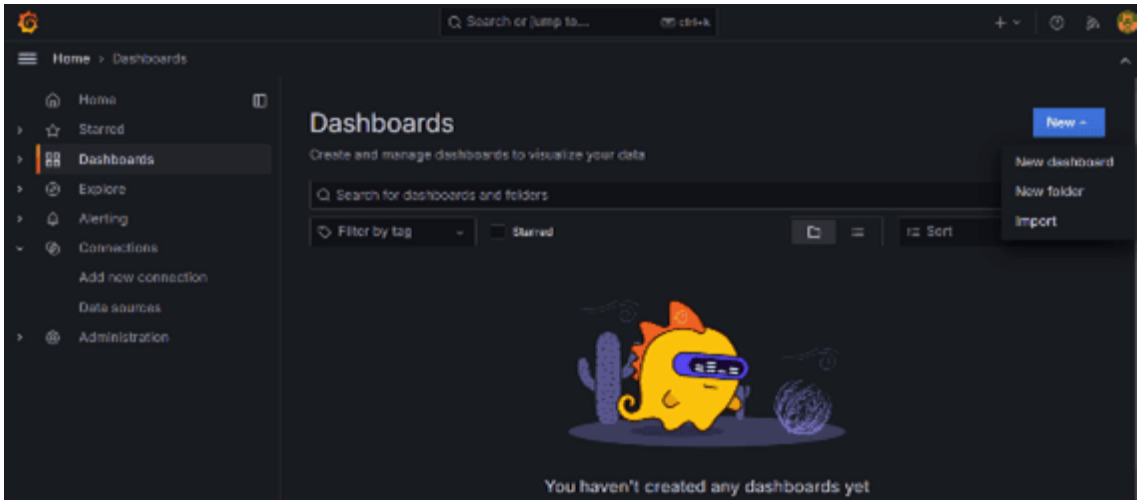
- Seguimos desplazando hacia abajo para configurar las opciones del servidor Influx y la autenticación en caso de ser necesario:



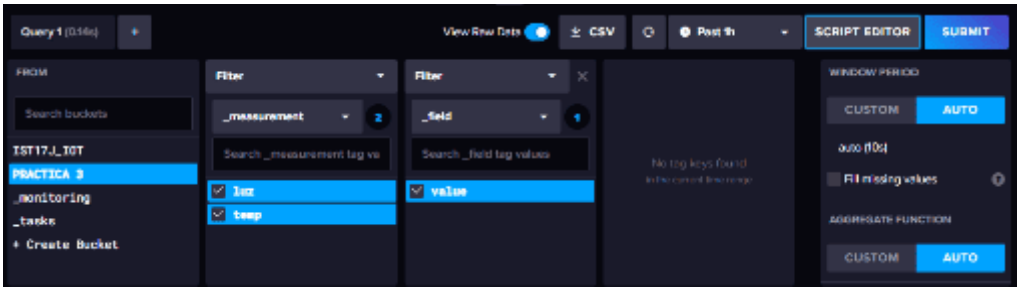
- Por último, configuramos los detalles de la BDD a la que accederemos desde Grafana, colocando la información de la Organización, el Token y el nombre del Bucket, presionamos Save & test y tendríamos todo configurado:



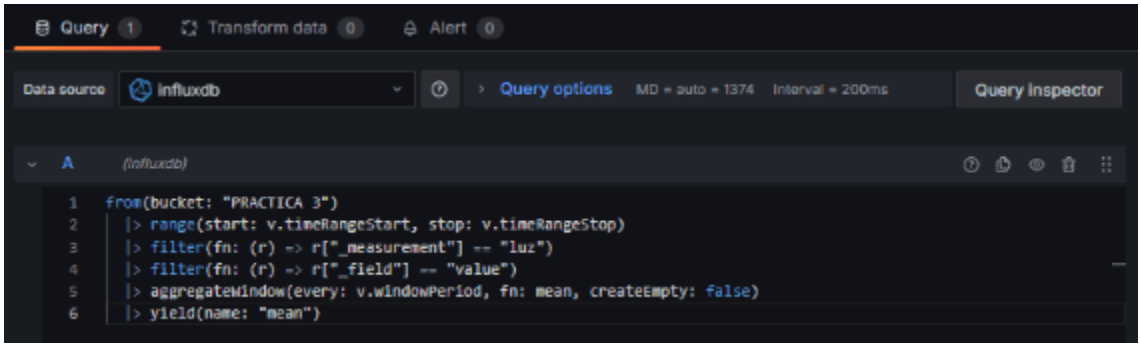
- **Diseño de Dashboards**
    - Seleccionamos la pestaña Dashboards en el panel de configuraciones y presionamos el botón New para crear una nueva ventana de visualización:



    - Ahora presionamos Add visualization y seleccionamos InfluxDB como fuente de datos. Se desplegarán todas las opciones para configurar un dashboard dependiendo de los datos que se envíen desde la BDD. Para lo cual tenemos que agregar el código JSON que se genera en Influx haciendo clic en SCRIPT EDITOR:

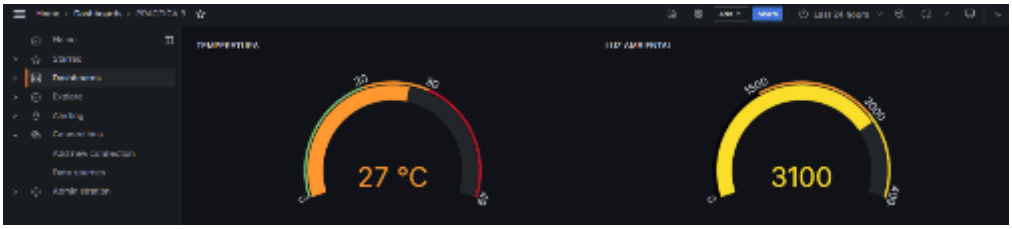


    - Esto hará que se genere el código que debemos pegar en la sección Query del dashboard de Grafana:



- Seleccionamos el tiempo de visualización histórico de datos, un gráfico acorde a la variable, lo nombramos según el dato monitoreado y guardamos el dashboard con el nombre PRACTICA 3.

- Una vez guardado el Dashboard podemos hacer las configuraciones necesarias para que los gráficos presentados sean lo más representativos para el usuario, colocando nombres, unidades, rangos, variando los colores, el tamaño entre muchas otras cosas que nos permite Grafana. Además, tenemos que agregar todas las variables que se van a monitorear, siguiendo los pasos antes explicados.



Grafana permite más personalización de los dashboards y al no estar enlazado directamente con Node-RED que es nuestro sistema de gestión, el administrador del sistema tiene mayor libertad y privacidad para gestionarlo, siendo que el usuario final solo tendría acceso a los dashboards, ya que esta aplicación nos permite generar enlaces para compartir o exportar los paneles de visualización en formato JSON.

Resultados de la Práctica

En esta práctica se busca integrar un sistema funcional para la gestión y monitoreo de sensores y actuadores utilizando Node-RED como herramienta de administración, InfluxDB como base de datos para el almacenamiento histórico, y Grafana para la visualización de datos mediante dashboards personalizados. El sensor DHT22 proporciona lecturas precisas de temperatura, mientras que el sensor LDR monitorea los niveles de luz ambiental. Estos valores son procesados por la ESP32, que envía la información a través del protocolo MQTT hacia Node-RED, donde se gestiona las reglas de control para los actuadores: el ventilador y el LED. El ventilador se activa automáticamente cuando la temperatura supera el umbral configurado (30 °C), y el LED se enciende cuando el nivel de luz ambiental es insuficiente. Esto permite automatizar el control de las condiciones ambientales, emulando un sistema de gestión de edificios inteligentes.

El flujo de trabajo en Node-RED permite centralizar la administración del sistema, facilitando la configuración de reglas de control y el envío de datos a InfluxDB para su almacenamiento. Cada medición de temperatura y luz es registrada con precisión, estructurada en series temporales en la base de datos. Esto permite disponer de un histórico confiable para análisis posterior. Grafana, por su parte, proporciona dashboards interactivos y visualmente atractivos que permiten a los usuarios finales observar en tiempo real tanto las mediciones de los sensores como el estado de los actuadores. Este enfoque completo demuestra

la capacidad de integración de hardware y software en un sistema IoT, aportando soluciones eficientes y escalables para la gestión inteligente de entornos físicos.

Evaluación del Aprendizaje

El docente calificará todo el proceso y desarrollo de la práctica estableciendo una calificación final sobre 10 puntos en base a la rúbrica presentada en la tabla 6:

Tabla 6
Rúbrica de evaluación.

Criterio de Evaluación	Nivel de Desempeño
Diseño e implementación del circuito	2 puntos
Configuración de Node-RED e InfluxDB	2 puntos
Diseño del dashboard en Grafana	2 puntos
Integración del hardware y software para el monitoreo	2 puntos
Desarrollo del informe de laboratorio con claridad y que contenga toda la información	2 puntos

Actividades Complementarias

Con el objetivo de practicar y mejorar en la metodología de despliegue y servicios:

- **Integración de Sensores Adicionales**
  - Sensor de movimiento (PIR): Incorporar un sensor PIR para detectar presencia, útil para activar luces o alarmas solo cuando haya personas presentes, mejorando la eficiencia energética.
  - Sensor de calidad del aire (MQ-135): Monitorear los niveles de contaminación del aire, como CO2 o compuestos volátiles, y activar ventiladores o alertas si se detectan valores peligrosos.
  - Modifica el código de Arduino y el flujo de Node-RED para integrar las lecturas de los nuevos sensores, mostrando sus

datos y generando alertas si los valores superan un umbral definido.
    - Evaluar cómo afectan estos datos adicionales al monitoreo y control de las variables.

- **Control de Actuadores Adicionales**
    - Conecta un actuador adicional, como un relé que controle un dispositivo externo para abrir o cerrar ventanas dependiendo de las mediciones del sensor de calidad de aire.
    - Modifica el flujo en Node-RED para que el actuador se active o desactive en función de las lecturas del sensor o mediante un botón en la interfaz gráfica.

- **Modos de Ahorro de Energía**
    - Deep Sleep en ESP32: Configurar el modo de sueño profundo de la ESP32 para apagar componentes no esenciales cuando no se requiera monitoreo constante. Por ejemplo, la ESP32 podría despertarse solo en intervalos específicos para tomar mediciones.
    - Control de iluminación adaptativa: Implementar un sistema en el que las luces solo se activen cuando los niveles de luz ambiental sean bajos y haya presencia detectada por un sensor PIR.
    - Monitoreo del estado de carga: Integrar un sensor de corriente o voltaje para monitorear el consumo energético del sistema y optimizar su uso.

Cuestionario

- ¿Por qué se utiliza un módulo MOSFET IRF520N para controlar el ventilador y no se conecta directamente al pin de la ESP32?

_____

_____

_____

_____

- Describe cómo se gestiona la comunicación entre los sensores y Node-RED.

- Explica cómo se configura InfluxDB para almacenar los datos de temperatura y luz en una base de datos.

- ¿Qué ajustes adicionales se deben hacer en el código para implementar modos de ahorro de energía en la ESP32?

- ¿Consideras que este sistema es escalable para gestionar edificios más grandes? Justifica tu respuesta.

_____
_____
_____
_____
_____
_____
_____

CAPÍTULO V

Integración de Alertas y Control mediante Telegram

ALUMNI EDITORA

Fundamentación

Telegram es una plataforma ideal para el desarrollo de sistemas IoT debido a sus características abiertas y flexibles. A través del uso de bots (programas automatizados dentro de Telegram), es posible crear interfaces de usuario simples para notificaciones y control remoto [16]. Esto elimina la necesidad de desarrollar aplicaciones móviles adicionales, lo que ahorra tiempo y recursos.

El bot de Telegram interactuará con la ESP32 a través del protocolo MQTT y Node-RED, funcionando como una extensión del sistema IoT. Esta integración permitirá ampliar el alcance del proyecto y demostrar cómo las tecnologías modernas pueden facilitar la interacción con dispositivos IoT en tiempo real. En la tabla 7, se describen las aplicaciones en el campo laboral y/o profesional:

Tabla 7
Aplicaciones profesionales de la práctica 4.

Aplicación	Descripción
Monitoreo Remoto de Sistemas IoT	Esta funcionalidad es especialmente útil en edificios inteligentes, sistemas industriales o agrícolas, donde el monitoreo y control remoto son esenciales.
Gestión de Alarmas	Las alertas en tiempo real permiten responder rápidamente a situaciones críticas, mejorando la seguridad y eficiencia operativa.
Optimización de Recursos	La capacidad de configurar el sistema desde Telegram reduce la necesidad de supervisión constante y permite realizar ajustes a distancia.

Objetivos

Objetivo General

Implementar un sistema IoT que integre sensores, actuadores, y alertas en tiempo real mediante Telegram, permitiendo la supervisión y control remoto de parámetros ambientales y dispositivos en un entorno automatizado.

Objetivos Específicos

Configurar un bot de Telegram que permita recibir alertas en tiempo real basadas en los datos de sensores ambientales.

Desarrollar un sistema que permita controlar los actuadores (ventiladores y luces) mediante comandos enviados desde Telegram.

Diseñar la conexión e integración de sensores, actuadores y la ESP32, garantizando la comunicación eficiente con Node-RED y Telegram.

Programar la ESP32 para que envíe datos a través del protocolo MQTT y responda a comandos desde Telegram.

Preparación Previa

Antes de iniciar esta práctica, los estudiantes deben tener conocimientos sólidos sobre la programación de la ESP32, el uso del protocolo MQTT, y la configuración de herramientas como Node-RED, InfluxDB y Grafana. Esto es esencial para garantizar que comprendan cómo se integran los datos de los sensores y los actuadores dentro de un sistema IoT. Además, deben estar familiarizados con los conceptos básicos de comunicación en red, como el envío y recepción de datos entre dispositivos y servidores, lo que permitirá comprender la interacción entre Telegram y la plataforma IoT.

Es necesario que los estudiantes conozcan el funcionamiento de los bots en Telegram. Esto incluye la creación y configuración de un bot utilizando el BotFather en Telegram, y la forma en que los bots interactúan con aplicaciones externas mediante el uso de tokens de autenticación. Asimismo, deben comprender cómo los comandos enviados a través de un bot pueden ser procesados en Node-RED para ejecutar acciones específicas en el sistema IoT. Este conocimiento previo es fundamental para implementar tanto las alertas en tiempo real como el control remoto de actuadores desde la aplicación de Telegram.

Finalmente, los estudiantes deben asegurarse de tener instalados todos los paquetes y nodos necesarios en Node-RED, como “node-red-contrib-telegrambot”, y haber configurado correctamente un broker MQTT para la comunicación con la ESP32. También es importante que tengan una cuenta activa en Telegram y hayan realizado pruebas básicas de envío y recepción de mensajes.

Procedimiento

Esta práctica permite implementar un sistema IoT avanzado que combina la supervisión de parámetros ambientales, el control de actuadores y el envío de alertas en tiempo real a través de Telegram. El sistema utiliza una placa ESP32 como microcontrolador principal, que recopila datos de un sensor de temperatura y humedad (DHT22) y un sensor de luz (LDR). La información obtenida se procesa para activar un ventilador cuando la temperatura excede un umbral crítico y un LED indicador cuando los niveles de luz ambiental son insuficientes. Además, los usuarios pueden controlar manualmente estos actuadores mediante comandos enviados desde Telegram, proporcionando una solución completa de monitoreo y control remoto. Por lo tanto, Asegúrate de instalar las librerías necesarias desde el Gestor de Librerías del Arduino IDE, buscando “UniversalTelegramBot” e instalando su última versión.

La configuración de Telegram requiere crear un bot en la plataforma utilizando el BotFather, asignar un nombre y un token de autenticación al bot, y obtener el chat ID del usuario. Estos parámetros se integran en Node-RED para permitir que el sistema envíe notificaciones automáticas cuando los sensores detecten condiciones críticas, como un aumento de temperatura o una disminución de la luz. Asimismo, el bot permite recibir comandos específicos, como "ON_VENT" u "OFF_LED", para activar o desactivar manualmente el ventilador y el LED desde Telegram.

En el software, Node-RED desempeña un papel central en la gestión del flujo de datos y la comunicación con Telegram, ya que las alertas y los comandos se gestionan mediante nodos de Telegram en Node-RED.

Materiales

Los materiales tanto software como hardware que se utilizan en esta práctica son los siguientes:

Hardware:

- 1 placa ESP32.
- 1 sensor digital DHT11/DHT22 (temperatura y humedad).
- 1 sensor analógico de luz LDR.
- LED para iluminación.

- Ventilador de 5V.
- Módulo IRF520N comercial (MOSFET para el control del ventilador).
- Fuente externa de 5V.
- 1 resistencia de 10 kΩ.
- 1 resistencia de 220Ω.
- 1 Protoboard.
- Cables jumper.
- Smartphone con Telegram instalado.

Software:

- Arduino IDE (configurado para ESP32).
- Broker MQTT (Mosquitto instalado localmente o en un servidor).
- Node-RED para visualización de datos.
- Telegram para control y alertas.

Opcional:

- Multímetro digital (para verificar conexiones y niveles de voltaje).
- Caja de conexiones o soporte (para organizar el prototipo durante la práctica).
- Fuente de 12V (Para ventiladores de mayor voltaje).
- InfluxDB para base de datos.
- Grafana para diseño de dashboards.

Conexiones del Hardware

Para implementar esta práctica, sigue las indicaciones mostradas en la práctica 3, ya que se utilizan los mismos materiales y solo se hacen cambios en el software. Por lo tanto, se debe utilizar el esquema de conexiones de la figura 13.

Configuración del Software

En esta práctica, se aprovecha la configuración de software ya establecida en la práctica 3, donde se utilizaron herramientas como el código de Arduino, el protocolo MQTT y Node-RED. Sin embargo, para esta implementación se añaden nuevas funcionalidades relacionadas con Telegram, lo que permite generar alertas en tiempo real y controlar los actuadores a través de comandos enviados desde esta plataforma. El cambio principal en el sistema se realiza en el código de

Arduino, donde se integran librerías y funciones específicas para manejar la comunicación con un bot de Telegram. Esto habilita el envío de mensajes automatizados cuando las condiciones de los sensores superen ciertos umbrales y permite al usuario controlar el ventilador y el LED mediante comandos.

En Node-RED, además de mantener los nodos MQTT y el dashboard configurados en la práctica anterior, se integran nodos específicos para manejar la interacción con Telegram. Estos nodos se encargan de recibir comandos enviados por los usuarios al bot y de reenviar alertas generadas por los sensores. Para completar esta configuración, es necesario crear un bot en Telegram, lo cual se realiza utilizando el BotFather, una herramienta oficial de Telegram para la gestión de bots. Una vez creado el bot, se obtiene un token único que permite vincular el bot con el sistema. Este token se utiliza tanto en Node-RED como en el código de Arduino para autenticar las comunicaciones y garantizar que las acciones sean realizadas de manera segura.

Código de Arduino

El código completo se encuentra escrito en el Anexo 4, en esta sección se detalla cada parte del código para tener una mejor comprensión de este.

- **Inclusión de Librerías:**
  - Se deben incluir las librerías necesarias, en este caso la librería DHT.h, Wifi.h, PubSubClient.h, UniversalTelegramBot.h y WiFiClientSecure.h:

```
// Incluir las librerías necesarias
#include <DHT.h>                // Librería para manejar el sensor de temperatura y humedad DHT
#include <WiFi.h>               // Librería para conectividad WiFi
#include <PubSubClient.h>       // Librería para el protocolo MQTT
#include <UniversalTelegramBot.h> // Librería para interactuar con Telegram
#include <WiFiClientSecure.h>   // Librería para la comunicación HTTPS necesaria para Telegram
```

- **Pines, configuraciones y Umbrales**:
  - Se definen los pines a los que están conectados los sensores, el MOSFET y el LED.
  - Los umbrales de temperatura e iluminación se configuran a 30°C y 3000 respectivamente.

```
// Pines, configuraciones y umbrales
#define DHTPIN 25          // Pin GPIO donde está conectado el sensor DHT22
#define DHTTYPE DHT22      // Tipo de sensor utilizado (DHT11 o DHT22)
#define LDR_PIN 14         // Pin ADC donde está conectado el sensor de luz LDR
#define LED_LUZ 17         // GPIO para el LED indicador de luz
#define PIN_VENT 13        // Pin GPIO conectado al módulo MOSFET que controla el ventilador
#define TEMP_UMBRAL 30.0   // Umbral de temperatura crítica para activar el ventilador
#define LUZ_UMBRAL 3000    // Umbral de luz para encender el LED
```

- **Configuración de Protocolos:**
  - Para el protocolo Wi-Fi se deben definir el nombre de la red y la contraseña, para que la ESP32 se conecte correctamente.
  - Para el protocolo MQTT se define la dirección IP en la que se encuentra instalado el broker, en este caso se debe buscar que dirección tiene el computador personal. Si se utilizan brokers en la nube escribir ahí la dirección.
  - Para el bot de Telegram se incluye el token generado en la aplicación y el chat ID.

```
// Configuración de WiFi
const char* ssid = "Tu_SSID";            // Nombre de la red WiFi
const char* password = "Tu_Contraseña";  // Contraseña de la red WiFi

// Configuración de MQTT
const char* mqtt_server = "192.168.0.10"; // Dirección IP del broker MQTT
const char* topic_temp = "home/temp";     // Tópico para publicar datos de temperatura
const char* topic_luz = "home/luz";       // Tópico para publicar datos de luz

// Configuración de Telegram
const char* botToken = "Tu_Token_Bot_Telegram"; // Token del bot de Telegram
const char* chatID = "Tu_Chat_ID";              // ID del chat donde se enviarán mensajes

// Inicialización de librerías y objetos
DHT dht(DHTPIN, DHTTYPE);          // Objeto para interactuar con el sensor DHT
WiFiClientSecure clientTelegram;  // Cliente seguro para comunicación HTTPS
UniversalTelegramBot bot(botToken, clientTelegram); // Objeto del bot de Telegram
WiFiClient espClient;              // Cliente WiFi para MQTT
PubSubClient client(espClient);    // Cliente MQTT
```

- **Configuraciones del setup:**
  - Se inicializa la comunicación serial a 115200 baudios (velocidad compatible con ESP32).
  - Se inicializa el sensor DHT configurado.
  - Se configuran los pines de los sensores analógicos como entrada (INPUT) y los pines del LED y del MOSFET como salida (OUTPUT).

- Se inicializa la comunicación Wi-Fi y se configura el servidor MQTT en el puerto 1883.

```
void setup() {
  Serial.begin(115200);                // Iniciar comunicación serial para depuración
  dht.begin();                         // Iniciar el sensor DHT
  pinMode(LDR_PIN, INPUT);             // Configurar pin del sensor LDR como salida
  pinMode(PIN_VENT, OUTPUT);           // Configurar el pin del ventilador como salida
  pinMode(LED_LUZ, OUTPUT);            // Configurar el pin del LED como salida
  setupWiFi();                         // Conectar a la red WiFi
  client.setServer(mqtt_server, 1883); // Configurar el servidor MQTT
}
```

- **Funciones para Wi-Fi y MQTT**
  - Se configuran 2 funciones para la correcta conexión de los protocolos de comunicación Wi-Fi y MQTT.

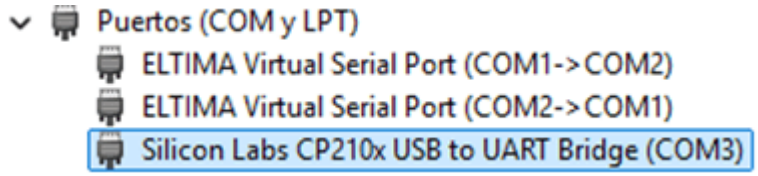
```
void setupWiFi() {
  // Conexión a la red WiFi
  delay(10);
  Serial.println("Conectando a WiFi...");
  WiFi.begin(ssid, password);      // Iniciar conexión con las credenciales de red
  while (WiFi.status() != WL_CONNECTED) {
    delay(1000);                   // Esperar hasta que se establezca la conexión
    Serial.print(".");             // Imprimir un punto como indicador de progreso
  }
  Serial.println("\nConexión WiFi establecida"); // Confirmar conexión exitosa
}

void reconnect() {
  // Reconexión al broker MQTT en caso de desconexión
  while (!client.connected()) {
    Serial.print("Conectando al broker MQTT...");
    // Intentar conectar al broker MQTT
    if (client.connect("ESP32_Client")) {
      Serial.println("Conectado al broker MQTT"); // Confirmar conexión exitosa
    } else {
      Serial.print("Fallo, rc=");                 // Mostrar código de error
      Serial.print(client.state());
      Serial.println(" Intentando de nuevo en 5 segundos");
      delay(5000);                                // Esperar antes de reintentar
    }
  }
}
```

- **Configuración del loop:**
  - Se establece la conexión MQTT.
  - Se realiza la lectura de los sensores.
  - Se configura el control de los actuadores (Ventilador y LED) y se envían los estados al bot de Telegram.

- Se publican los datos en el broker MQTT.
- Se muestran los datos en el monitor serial.
- Se configura el control de actuadores desde Telegram

```
void loop() {
  if (!client.connected()) {
    reconnect();
  }
  client.loop(); // Mantener conexión MQTT activa

  // Leer sensores
  float temp = dht.readTemperature(); // Temperatura en °C
  float luz = analogRead(LDR_PIN);    // Lectura analógica del sensor LDR

  // Verificar errores de lectura
  if (isnan(temp)) {
    Serial.println("Error al leer el DHT");
    return;
  }

// Control del ventilador
if (temp > TEMP_UMBRAL) {
  digitalWrite(PIN_VENT, HIGH); // Activar ventilador si la temperatura supera el umbral
  bot.sendMessage(chatID, "Temperatura alta: ventilador encendido", "Markdown");
} else {
  digitalWrite(PIN_VENT, LOW);  // Desactivar ventilador si la temperatura está dentro del rango
  bot.sendMessage(chatID, "Temperatura normal: ventilador apagado", "Markdown");
}

// Control del LED basado en la luz ambiental
if (luz < LUZ_UMBRAL) {
  digitalWrite(LED_LUZ, HIGH); // Encender LED si la luz es insuficiente
  bot.sendMessage(chatID, "Iluminación baja: LED encendido", "Markdown");
} else {
  digitalWrite(LED_LUZ, LOW);  // Apagar LED si la iluminación es suficiente
  bot.sendMessage(chatID, "Iluminación adecuada: LED apagado", "Markdown");
}

// Publicar datos en MQTT
client.publish(topic_temp, String(temp).c_str()); // Publicar temperatura
client.publish(topic_luz, String(luz).c_str());   // Publicar nivel de luz

// Mostrar datos en el monitor serial
Serial.print("Temperatura: "); Serial.print(temp); Serial.print(" °C, ");
Serial.print("Luz: "); Serial.println(luz);

  // Escuchar comandos en Telegram
  int numNewMessages = bot.getUpdates(bot.last_message_received + 1);
  for (int i = 0; i < numNewMessages; i++) {
    String text = bot.messages[i].text; // Leer comando recibido
    if (text == "/ON_FAN") {
      digitalWrite(PIN_VENT, HIGH);     // Encender ventilador desde Telegram
      bot.sendMessage(chatID, "Ventilador encendido", "Markdown");
    } else if (text == "/OFF_FAN") {
      digitalWrite(PIN_VENT, LOW);      // Apagar ventilador desde Telegram
      bot.sendMessage(chatID, "Ventilador apagado", "Markdown");
    } else if (text == "/ON_LED") {
      digitalWrite(LED_LUZ, HIGH);      // Encender LED desde Telegram
      bot.sendMessage(chatID, "LED encendido", "Markdown");
    } else if (text == "/OFF_LED") {
      digitalWrite(LED_LUZ, LOW);       // Apagar LED desde Telegram
      bot.sendMessage(chatID, "LED apagado", "Markdown");
    }
  }

  delay(5000); // Intervalo de 5 segundos entre lecturas
}
```

- **Compilación y Subida del Código:**
  - Compilar el código para comprobar que no existen errores, en el caso de haberlos, revisar bien el código y depurar los errores.
  - Conectar la placa ESP32 a un puerto USB del computador y comprobar que puerto se asigna.



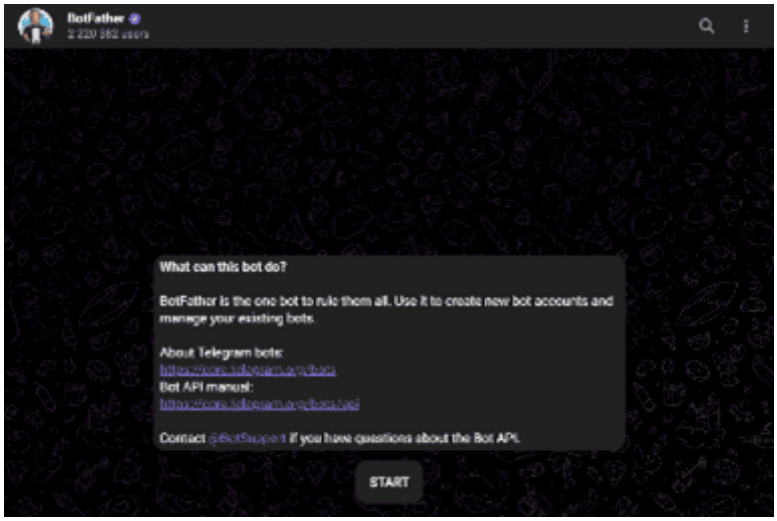
  - Seleccionar el puerto específico en el IDE y subimos el código a la ESP32.

Configuración de Telegram

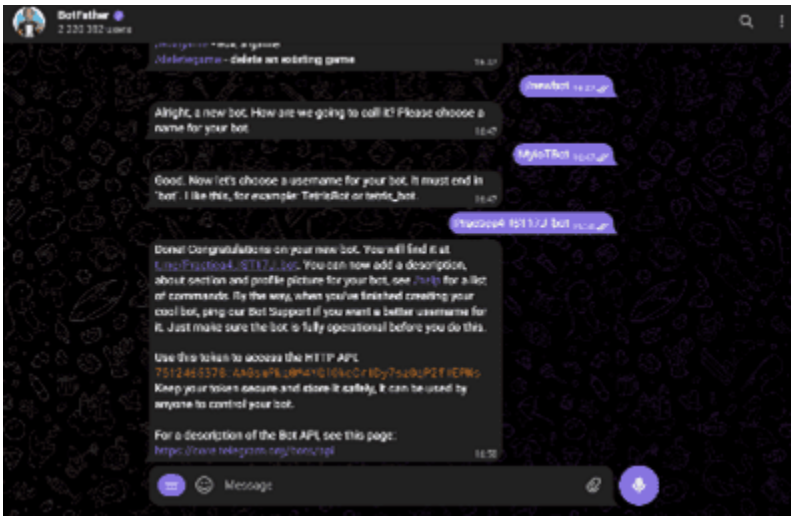
Creación de un Bot en Telegram:

La configuración de Telegram requiere crear un bot en la plataforma utilizando el BotFather, asignar un nombre y un token de autenticación al bot, y obtener el chat ID del usuario. Estos parámetros se integran en Node-RED para permitir que el sistema envíe notificaciones automáticas cuando los sensores detecten condiciones críticas, como un aumento de temperatura o una disminución de la luz. Asimismo, el bot permite recibir comandos específicos, como "ON_FAN" u "OFF_LED", para activar o desactivar manualmente el ventilador y el LED desde Telegram, para lo cual:

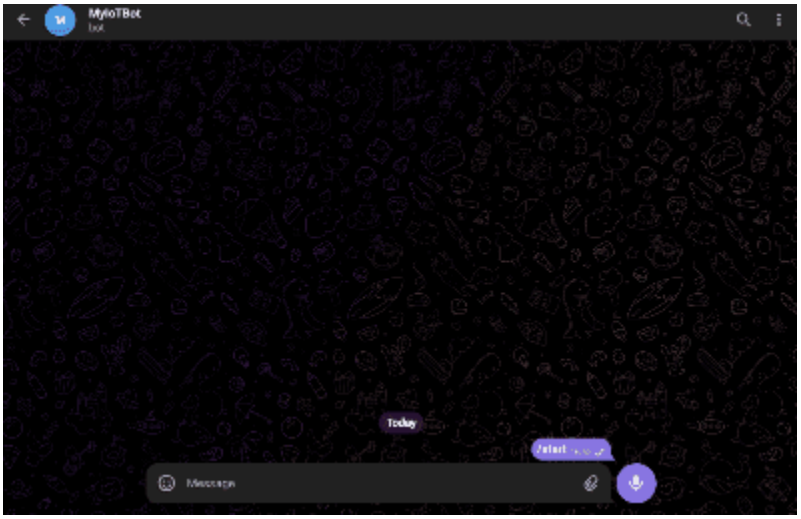
- Abre Telegram desde el smartphone o desde la aplicación web, busca el bot llamado BotFather y presiona el botón START:



- En el chat con BotFather, envía el comando /newbot para crear un nuevo bot.
- Asigna un nombre y un nombre de usuario único para tu bot (por ejemplo, MyIoTBot y Practica_4ISTJ_bot respectivamente).



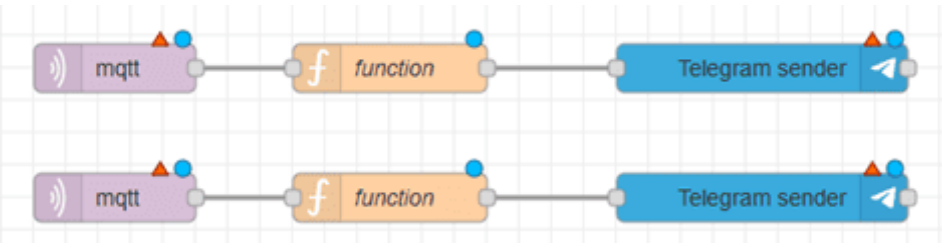
- Una vez creado, BotFather te proporcionará un token de acceso único.
- Copia este token, ya que lo necesitarás más adelante.
- Una vez creado el bot, ya podemos acceder al chat ID según el nombre de usuario que asignamos, por ejemplo: t.me/ Practica_4ISTJ_bot y presionamos el botón START:



Flujo de Node-RED

Para esta práctica se necesita instalar los nodos para la conexión con Telegram, nos dirigimos a la sección de configuración ≡ y en la opción “manage palette” (administrar paleta) seleccionar la pestaña “Install” (instalar), para buscar e instalar el nodo “node-red-contrib-telegrambot”. Una vez instalados los nodos necesarios, se procede con el diseño y configuración del flujo que lo denominaremos “PRACTICA 4”, en primer lugar configuraremos Node-RED para enviar los datos de los sensores a Telegram para lo cual seleccionamos y arrastramos al área de trabajo los nodos necesarios y los conectamos como se muestra en la figura 19:

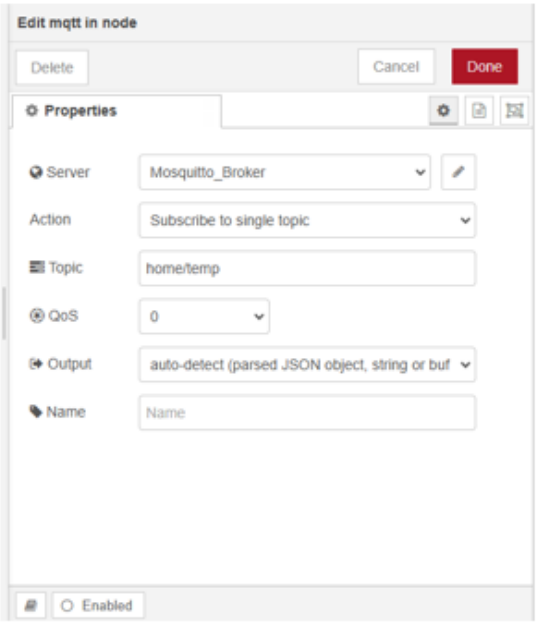
Figura 19
Conexión de nodos en Node-RED para la práctica 4 envío de mensajes.

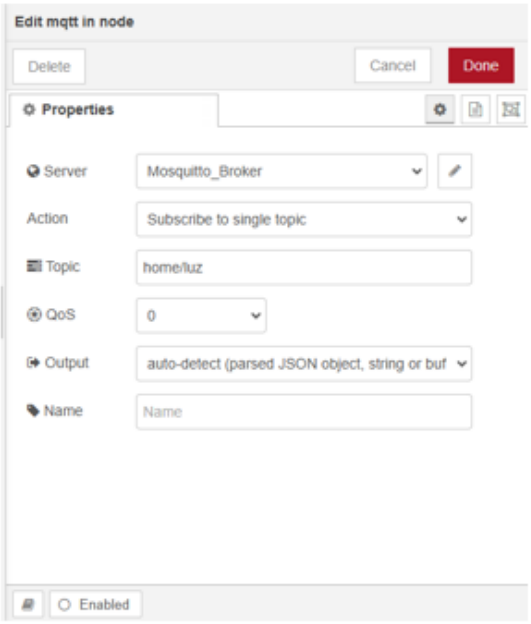


Se utilizan 2 nodos mqtt in para facilitar la configuración, ya que cada nodo se suscribirá a un tópico para mostrar los datos de cada sensor. Los nodos function se utilizan para enviar de manera correcta los mensajes a Telegram.

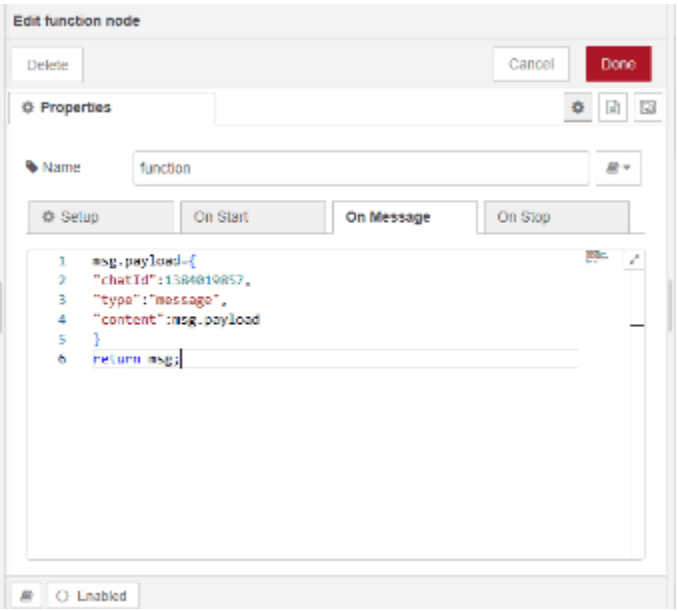
- **Nodos mqtt in:**

- Debido a que ya creamos el broker en la práctica 2, solo lo seleccionamos y en cada nodo mqtt in nos suscribimos a un tópico:

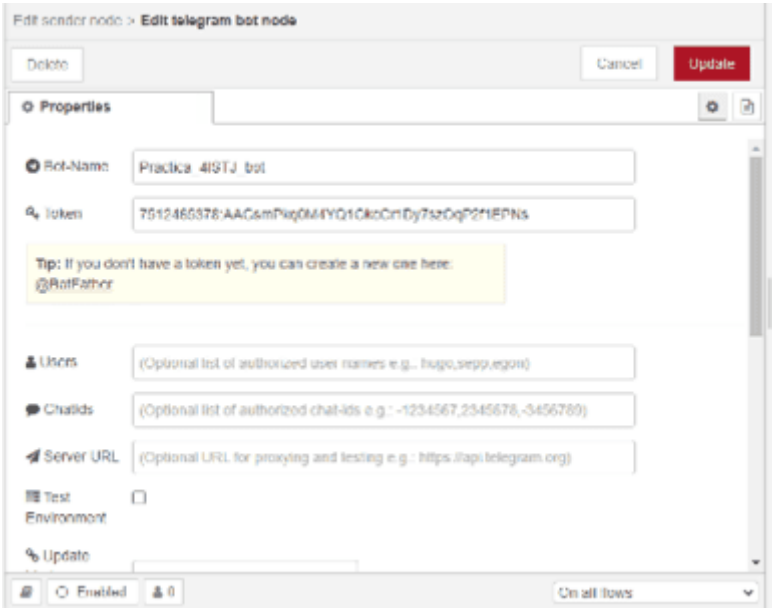




- **Nodos function:**
  - Se configuran los 2 nodos function para procesar los datos provenientes de la ESP y poder enviarlos a Telegram.
  - Es necesario indicar el chat ID, el tipo del mensaje y el contenido.



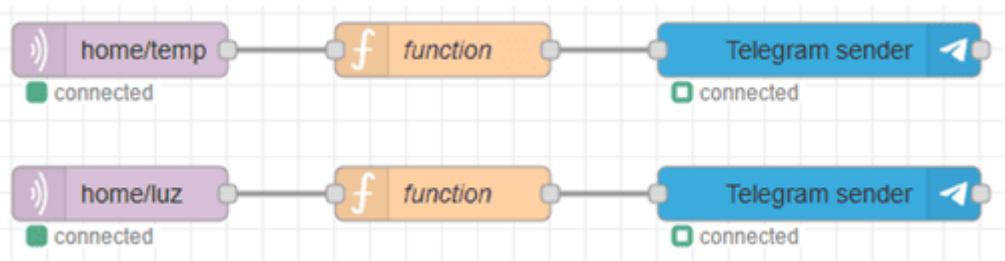
- **Nodos Telegram sender**
    - Para enviar los datos a Telegram es necesario agregar el bot que creamos en estos nodos, para lo cual seleccionamos el botón de lápiz y le indicamos nombre de usuario del bot y el token.



    - Presionamos el botón de Update y luego el botón de Done.

Con todos los nodos configurados, se procede a instanciar la configuración dando clic en el botón Deploy. Se puede notar que todas las alertas han desaparecido y nos quedaría el flujo como se muestra en la figura 20:

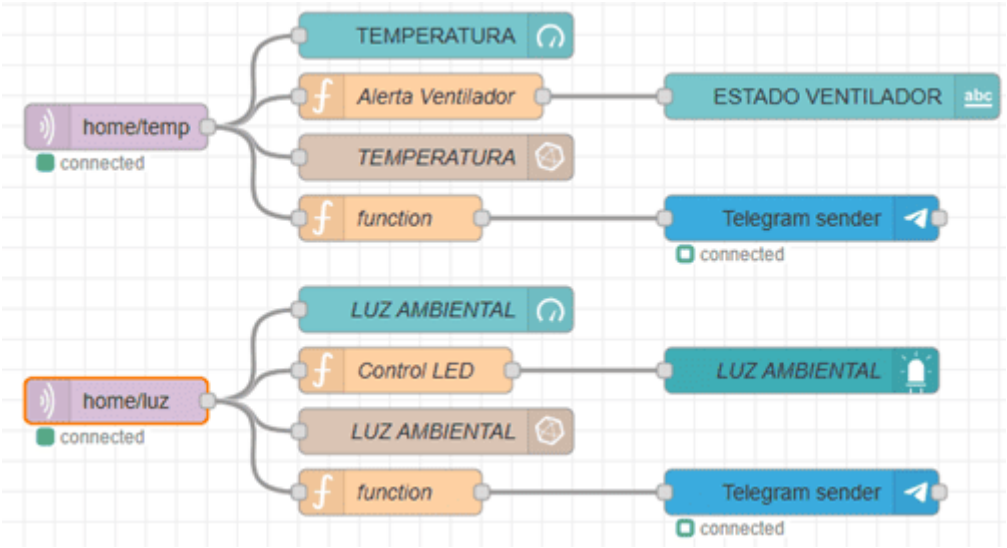
Figura 20
Flujo de la Práctica 4 configuración inicial.



De esta manera en Telegram empezaremos a recibir los datos de las variables que estamos monitoreando, en este caso la temperatura y la luz

ambiental. Se puede reutilizar la configuración de la práctica 3 para mostrar en gauges de Node-RED y enviar los datos a InfluxDB para su almacenamiento, los nodos quedarían conectados como se muestra en la figura 21:

Figura 21
Flujo de la Práctica 4 actualizado



No es necesario realizar configuraciones adicionales en Node-RED para los comandos de Telegram, ya que toda la lógica de control y la gestión de los comandos se encuentran implementadas directamente en el código de la ESP32. Esto incluye el manejo de los mensajes recibidos desde el bot de Telegram, el procesamiento de los comandos enviados por el usuario, y la activación o desactivación de los actuadores en función de las instrucciones recibidas. Al delegar esta funcionalidad directamente al microcontrolador, se simplifica la configuración en Node-RED, dejando su uso únicamente para la visualización de los datos recopilados por los sensores y el monitoreo del estado del sistema. Esta integración eficiente permite que el sistema sea más autónomo y flexible en su funcionamiento.

Resultados de la Práctica

Al completar esta práctica, se espera que los estudiantes integren con éxito un sistema IoT que combine el monitoreo en tiempo real de sensores ambientales con el control remoto de actuadores utilizando Telegram como medio principal de interacción. A través del diseño e implementación, los estudiantes configuran

un bot en Telegram que no solo envía alertas en tiempo real cuando se superan los umbrales de temperatura o iluminación, sino que también permite a los usuarios enviar comandos para controlar los actuadores, como un ventilador o un LED indicador.

En el aspecto técnico, los estudiantes desarrollan habilidades en la configuración del protocolo MQTT, la implementación de flujos en Node-RED y la programación en Arduino para manejar la lógica del sistema. Como resultado, los sensores de temperatura y luz transmitirán datos continuamente al broker MQTT, permitiendo que Node-RED procese la información y la utilice para enviar mensajes o activar actuadores según corresponda. Adicionalmente, se espera que los datos recopilados se visualicen en tiempo real en un dashboard de Node-RED y que las alertas importantes se envíen de forma eficiente al chat del bot en Telegram.

Los estudiantes tendrán una comprensión más profunda de cómo combinar tecnologías de comunicación y automatización para resolver problemas reales en aplicaciones de domótica o edificios inteligentes. Los resultados de la práctica proporcionarán un sistema IoT funcional que podría adaptarse fácilmente a diferentes escenarios del mundo laboral, demostrando la aplicabilidad y la versatilidad de las herramientas utilizadas.

Evaluación del Aprendizaje

El docente calificará todo el proceso y desarrollo de la práctica estableciendo una calificación final sobre 10 puntos en base a la rúbrica presentada en la tabla 6:

Tabla 8
Rúbrica de evaluación.

Criterio de Evaluación	Nivel de Desempeño
Diseño e implementación del circuito	2 puntos
Configuración de Node-RED y Telegram	2 puntos
Envío y recepción de alertas en tiempo real	2 puntos
Integración del hardware y software para el monitoreo y alertas	2 puntos

Desarrollo del informe de laboratorio con claridad y que contenga toda la información	2 puntos

Actividades Complementarias

Con el objetivo de practicar y mejorar en el monitoreo y alertas en tiempo real se plantean las siguientes actividades para que el estudiante desarrolle:

- **Integración de Sensores Adicionales**
  - Integrar todo tipo de sensores para monitorear diferentes variables dependiendo del sistema que se desee implementar.
- **Control de Actuadores Adicionales**
  - Conectar actuadores adicionales relacionados con los nuevos sensores monitoreados, para poder controlar mediante Telegram sus funcionalidades.
- **Modos de Ahorro de Energía**
  - Implementar el modo Deep Sleep para mejorar la eficiencia energética.
- **Almacenamiento en BDD**
  - Integrar InfluxDB para el almacenamiento en BBD temporales para el análisis en bruto y posterior de toda la información.
- **Dashboards en Grafana**
  - Diseñar paneles de visualización en Grafana que muestren de manera interactiva las mediciones de los sensores y los estados de los actuadores.

Cuestionario

- Describa cómo se establecen las conexiones entre los nodos en Node-RED para el envío de mensajes hacia Telegram.

_____

_____

_____

_____

_____

- ¿Qué ventajas tiene el uso de un bot de Telegram para la gestión y control del sistema IoT?

- ¿Por qué es necesario utilizar Node-RED como intermediario entre los sensores y Telegram?

- ¿Cómo se podría mejorar el sistema para incluir más sensores o actuadores?

- ¿Qué aplicaciones prácticas tiene este sistema en el ámbito profesional o laboral?

Referencias

[1] E. Baccelli, Internet de las cosas (IoT) Retos sociales y campos de investigación científica en relación con la IoT, Santiago: Inría, 2022.

[2] M. Zapata, L. Topón-Visarrea y E. Tipán, Fundamentos de Automatización y Redes Industriales, Quito: Editorial Universidad Tecnológica Indoamérica, 2021.

[3] A. Román, J. Román-Herrera, S. Sandoval, M. Andrade y E. Ramos, Internet de las cosas Teoría y práctica, Colima: Universidad de Colima, 2023.

[4] FutuRed, «Libro de Comunicaciones,» de *I Edición Congreso de Redes Inteligentes*, Madrid, 2022.

[5] Espressif Systems, « ESP32 Overview,» 2024. [En línea]. Available: https://www.espressif.com/en/products/socs/esp32.

[6] Espressif Systems, «Technical Documents | Espressif Systems,» 13 Febrero 2023. [En línea]. Available: https://www.espressif.com/en/support/documents/technical-documents.

[7] Arduino, «Overview of the Arduino IDE 1 | Arduino Documentation,» 17 Enero 2024. [En línea]. Available: https://docs.arduino.cc/software/ide-v1/tutorials/Environment/?_gl=1*q5ya5h*_up*MQ..*_ga*MTkzNjc5Nzc1OC4xNzM2ODYzMDM4*_ga_NEXN8H46L5*MTczNjg2MzAzNy4xLjAuMTczNjg2MzAzNy4wLjAuNTkyNjE0OTEy.

[8] OpenJS Foundation, «About: Node-RED,» 2019. [En línea]. Available: https://nodered.org/about/.

[9] OpenJS Foundation, «User Guide: Node-RED,» 2019. [En línea]. Available: https://nodered.org/docs/user-guide/.

[10] InfluxData, «Get started with InfluxDB,» 2022. [En línea]. Available: https://docs.influxdata.com/influxdb/v2/get-started/.

[11] AWS, «¿Qué es el MQTT? - Explicación del protocolo MQTT - AWS,» 2024. [En línea]. Available: https://aws.amazon.com/es/what-is/mqtt/.

[12] Granollers et al., «Actas del XXIII Congreso Internacional de Interacción Persona-Ordenador,» de *INTERACCIÓN 2023*, Lleida, 2023.

[13] Gunawan et al., «Prototype Design of Smart Home System using Internet of Things,» *Indonesian Journal of Electrical Engineering and Computer Science,* vol. 7, nº 1, pp. 107-115, 2017.

[14] Abdulmalek et al., «IoT-Based Healthcare-Monitoring System towards Improving Quality of Life: A Review,» *Healthcare (Basel),* vol. 10, nº 10, pp. 1-32, 2022.

[15] Secretaría de Estado de Seguridad, Guía sobre Seguridad en Dispositivos IoT, Madrid: Ministerio del Interior de España, 2023.

[16] J. Inestroza, «Acercamiento al uso de Bots como interfaz para Internet de las Cosas,» *South Florida Journal of Development,* vol. 2, nº 1, pp. 1159-1177, 2020.

Anexos

Código de la Práctica 1

```
// Incluir las librerías necesarias
#include <DHT.h>

// Definición de pines
#define SENSOR_PIN 26     // Pin de conexión para el sensor DHT11/22
#define LED_TEMP 12       // Pin para el LED de temperatura
#define LED_HUM 13        // Pin para el LED de humedad

// Definición de umbrales
#define TEMP_UMBRAL 30    // Umbral de temperatura (en grados Celsius)
#define HUM_UMBRAL 70     // Umbral de humedad (en porcentaje)


// Configurar el tipo de sensor DHT
#define DHT_TYPE DHT22       // O puedes usar DHT11 dependiendo de tu sensor
DHT dht(SENSOR_PIN, DHT_TYPE); // Crear objeto para el sensor DHT

void setup() {
  // Inicializar la comunicación serial
  Serial.begin(115200);

  // Inicializar los pines de los LEDs como salida
  pinMode(LED_TEMP, OUTPUT);
  pinMode(LED_HUM, OUTPUT);
```

```
  // Inicializar el sensor DHT
  dht.begin();
}

void loop() {
  // Leer los valores de temperatura y humedad
  float temperatura = dht.readTemperature();  // Temperatura en grados Celsius
  float humedad = dht.readHumidity();       // Humedad en porcentaje

  // Verificar si las lecturas fueron exitosas
  if (isnan(temperatura) || isnan(humedad)) {
    Serial.println("Error al leer del sensor DHT");
    return;
  }

  // Control LEDs
  int led_temp = temperatura > TEMP_UMBRAL ? 1 : 0;  // Enciende LED de temperatura si supera el umbral
  int led_hum = humedad > HUM_UMBRAL ? 1 : 0;       // Enciende LED de humedad si supera el umbral

  // Escribir el estado en los pines de los LEDs
  digitalWrite(LED_TEMP, led_temp);  // Encender o apagar LED de temperatura
  digitalWrite(LED_HUM, led_hum);    // Encender o apagar LED de humedad

  // Imprimir los valores leídos en el monitor serial
  Serial.print(temperatura);
  Serial.print(",");
  Serial.print(humedad);
  Serial.print(",");
```

```
  Serial.print(led_temp);
  Serial.print(",");
  Serial.println(led_hum);

  // Esperar 2 segundos antes de leer nuevamente
  delay(2000);
}
```

Código de la Práctica 2

```
// Incluir las librerías necesarias
#include <DHT.h>                // Librería para el sensor DHT
#include <WiFi.h>               // Librería para conectividad WiFi
#include <PubSubClient.h>       // Librería para el protocolo MQTT

// Pines,configuraciones y umbrales
#define DHTPIN 25               // Pin GPIO donde está conectado el DHT
#define DHTTYPE DHT22           // Tipo de sensor (DHT11 o DHT22)
#define SENSOR_SUELO 4          // Pin ADC donde está conectado el sensor de humedad del suelo
#define LDR_PIN 13              // Pin ADC donde está conectado el sensor LDR
#define LED_TEMP 27             // GPIO para el LED indicador de temperatura
#define LED_HUM 14              // GPIO para el LED indicador de humedad
#define LED_LUZ 12              // GPIO para el LED indicador de luz
#define TEMP_UMBRAL 30.0        // Umbral de temperatura crítica
#define HUM_UMBRAL 20.0         // Umbral de humedad de suelo crítica
#define LUZ_UMBRAL 3000         // Umbral de luz para determinar cuando oscurece
#define FACTOR_uS_S 1000000ULL  // Factor de conversión de microsegundos a segundos
#define TIME_TO_SLEEP 60        // Tiempo de Deep Sleep en segundos

// Configuración de WiFi
const char* ssid = "Tu_SSID";           // Nombre de la red WiFi
const char* password = "Tu_Contraseña";  // Contraseña de la red WiFi

// Configuración MQTT
const char* mqtt_server = "localhost";   // Dirección IP del broker MQTT
const char* topic_temp = "agri/temp";    // Tópico para temperatura
```

```
const char* topic_hum = "agri/hum";       // Tópico para humedad
const char* topic_suelo = "agri/suelo";   // Tópico para humedad del suelo
const char* topic_luz = "agri/luz";       // Tópico para luz ambiental

DHT dht(DHTPIN, DHTTYPE);         // Inicialización del sensor DHT
WiFiClient espClient;             // Cliente WiFi
PubSubClient client(espClient);   // Cliente MQTT

void setup() {
  Serial.begin(115200);             // Iniciar comunicación serial
  dht.begin();                      // Iniciar el sensor DHT

  pinMode(SENSOR_SUELO, INPUT);       // Configurar pin del sensor de suelo
  pinMode(LDR_PIN, INPUT);            // Configurar pin del sensor LDR
  pinMode(LED_TEMP, OUTPUT);          // Configurar LED de temperatura como
salida
  pinMode(LED_HUM, OUTPUT);           // Configurar LED de humedad como
salida
  pinMode(LED_LUZ, OUTPUT);           // Configurar LED de luz como salida

  setupWiFi();                      // Conexión WiFi
  client.setServer(mqtt_server, 1883); // Configuración del servidor MQTT

  // Leer y publicar datos antes de entrar en Deep Sleep
  leerSensoresYPublicar();

  // Configurar Deep Sleep
  Serial.println("Entrando en modo Deep Sleep...");
  esp_sleep_enable_timer_wakeup(TIME_TO_SLEEP * FACTOR_uS_S); //
Configurar tiempo de sueño
```

```
  esp_deep_sleep_start(); // Entrar en Deep Sleep
}

void setupWiFi() {
  // Conexión a la red WiFi
  delay(10);
  Serial.println("Conectando a WiFi...");
  WiFi.begin(ssid, password);      // Iniciar conexión con las credenciales de red
  while (WiFi.status() != WL_CONNECTED) {
    delay(1000);                   // Esperar hasta que se establezca la conexión
    Serial.print(".");             // Imprimir un punto como indicador de progreso
  }
  Serial.println("\nConexión WiFi establecida"); // Confirmar conexión exitosa
}

void reconnect() {
  // Reconexión al broker MQTT en caso de desconexión
  while (!client.connected()) {
    Serial.print("Conectando al broker MQTT...");
    // Intentar conectar al broker MQTT
    if (client.connect("ESP32_Client")) {
      Serial.println("Conectado al broker MQTT"); // Confirmar conexión exitosa
    } else {
      Serial.print("Fallo, rc=");               // Mostrar código de error
      Serial.print(client.state());
      Serial.println(" Intentando de nuevo en 5 segundos");
      delay(5000);                             // Esperar antes de reintentar
    }
  }
```

```
}

void leerSensoresYPublicar() {
  if (!client.connected()) {
    reconnect();
  }
  client.loop();  // Mantener conexión MQTT activa

  // Leer sensores
  float temp = dht.readTemperature();       // Temperatura en °C
  float hum = dht.readHumidity();           // Humedad en %
  int hum_suelo = analogRead(SENSOR_SUELO);  // Lectura analógica del sensor de humedad del suelo
  int luz = analogRead(LDR_PIN);            // Lectura analógica del sensor LDR

  // Verificar errores de lectura
  if (isnan(temp) || isnan(hum)) {
    Serial.println("Error al leer el DHT");
    return;
  }

  // Publicar datos en MQTT
  client.publish(topic_temp, String(temp).c_str());       // Publicar temperatura
  client.publish(topic_hum, String(hum).c_str());         // Publicar humedad
  client.publish(topic_suelo, String(hum_suelo).c_str());  // Publicar humedad del suelo
  client.publish(topic_luz, String(luz).c_str());       // Publicar luz ambiental

  // Control de LEDs
  // Encender LED si la temperatura supera el umbral
```

```
  digitalWrite(LED_TEMP, temp > TEMP_UMBRAL ? HIGH : LOW);
  // Encender LED si la humedad está por debajo del umbral
  digitalWrite(LED_HUM, hum < HUM_UMBRAL ? HIGH : LOW);

  // Control del LED de luz
  if (luz < LUZ_UMBRAL)           // Si la luz es baja
    digitalWrite(LED_LUZ, HIGH);  // Enciende el LED de luz
  else
    digitalWrite(LED_LUZ, LOW);   // Apaga el LED de luz

  // Mostrar datos en el monitor serial
  Serial.print("Temp: "); Serial.print(temp); Serial.print(" °C, ");
  Serial.print("Hum: "); Serial.print(hum); Serial.print(" %, ");
  Serial.print("Suelo: "); Serial.print(hum_suelo); Serial.print(" , ");
  Serial.print("Luz: "); Serial.println(luz);
}

void loop() {
  // La ESP32 no ejecutará código en el loop porque entra en Deep Sleep después del setup
}
```

Código de la Práctica 3

```
// Incluir las librerías necesarias
#include <DHT.h>              // Librería para el sensor DHT
#include <WiFi.h>             // Librería para la conectividad WiFi
#include <PubSubClient.h>     // Librería para el protocolo MQTT

// Pines, configuraciones y umbrales
#define DHTPIN 25        // Pin GPIO donde está conectado el DHT
#define DHTTYPE DHT22    // Tipo de sensor (DHT11 o DHT22)
#define LDR_PIN 14       // Pin ADC donde está conectado el sensor LDR
#define LED_LUZ 17       // GPIO para el LED indicador de luz
#define PIN_VENT 13      // Pin GPIO para controlar el MOSFET que activa el ventilador
#define TEMP_UMBRAL 30.0  // Umbral de temperatura crítica para activar el ventilador
#define LUZ_UMBRAL 3000   // Umbral de luz (LDR) para encender el LED

// Configuración de WiFi
const char* ssid = "Tu_SSID";          // Nombre de la red WiFi
const char* password = "Tu_Contraseña";  // Contraseña de la red WiFi

// Configuración MQTT
const char* mqtt_server = "localhost";   // Dirección IP del broker MQTT
const char* topic_temp = "home/temp";    // Tópico para temperatura
const char* topic_luz = "home/luz";      // Tópico para luz ambiental

DHT dht(DHTPIN, DHTTYPE);        // Inicialización del sensor DHT
WiFiClient espClient;            // Cliente WiFi
PubSubClient client(espClient);   // Cliente MQTT
```

```
void setup() {
  Serial.begin(115200);              // Iniciar comunicación serial
  dht.begin();                       // Iniciar el sensor DHT

  pinMode(PIN_VENT, OUTPUT);         // Configurar el pin del ventilador como salida
  pinMode(LDR_PIN, INPUT);           // Configurar pin del sensor LDR como salida
  pinMode(LED_LUZ, OUTPUT);          // Configurar el LED de temperatura como salida

  setupWiFi();                       // Conexión WiFi
  client.setServer(mqtt_server, 1883); // Configuración del servidor MQTT
}

void setupWiFi() {
  // Conexión a la red WiFi
  delay(10);
  Serial.println("Conectando a WiFi...");
  WiFi.begin(ssid, password);      // Iniciar conexión con las credenciales de red
  while (WiFi.status() != WL_CONNECTED) {
    delay(1000);                   // Esperar hasta que se establezca la conexión
    Serial.print(".");             // Imprimir un punto como indicador de progreso
  }
  Serial.println("\nConexión WiFi establecida"); // Confirmar conexión exitosa
}

void reconnect() {
  // Reconexión al broker MQTT en caso de desconexión
  while (!client.connected()) {
```

```
  Serial.print("Conectando al broker MQTT...");
  // Intentar conectar al broker MQTT
  if (client.connect("ESP32_Client")) {
   Serial.println("Conectado al broker MQTT"); // Confirmar conexión exitosa
  } else {
   Serial.print("Fallo, rc=");          // Mostrar código de error
   Serial.print(client.state());
   Serial.println(" Intentando de nuevo en 5 segundos");
   delay(5000);                        // Esperar antes de reintentar
  }
 }
}

void loop() {
 if (!client.connected()) {
  reconnect();
 }
 client.loop(); // Mantener conexión MQTT activa

 // Leer sensores
 float temp = dht.readTemperature(); // Temperatura en °C
 float luz = analogRead(LDR_PIN);    // Lectura analógica del sensor LDR

 // Verificar errores de lectura
 if (isnan(temp)) {
  Serial.println("Error al leer el DHT");
  return;
 }
```

```
  // Control de ventilador
  if (temp > TEMP_UMBRAL)
    digitalWrite(PIN_VENT, HIGH); // Activar el ventilador si la temperatura supera el umbral
    else
    digitalWrite(PIN_VENT, LOW);  // Desactivar el ventilador si la temperatura está por debajo del umbral

  // Control de LED para luz
  if (light < LUZ_UMBRAL)
    digitalWrite(LED_LUZ, HIGH); // Activar el LED de luz si la iluminación es baja
    else
    digitalWrite(LED_LUZ, LOW);  // Desactivar el LED de luz si la iluminación es suficiente

  // Publicar datos en MQTT
  client.publish(topic_temp, String(temp).c_str());  // Publicar temperatura
  client.publish(topic_luz, String(luz).c_str()); // Publicar luz ambiental

  // Mostrar datos en el monitor serial
  Serial.print("Temp: "); Serial.print(temp); Serial.print(" °C, ");
  Serial.print("Light: "); Serial.println(light);

  delay(5000); // Intervalo de 5 segundos
}
```

Código de la Práctica 4

```
// Incluir las librerías necesarias
#include <DHT.h>                // Librería para manejar el sensor de temperatura y humedad DHT
#include <WiFi.h>               // Librería para conectividad WiFi
#include <PubSubClient.h>       // Librería para el protocolo MQTT
#include <UniversalTelegramBot.h> // Librería para interactuar con Telegram
#include <WiFiClientSecure.h>   // Librería para la comunicación HTTPS necesaria para Telegram

// Pines, configuraciones y umbrales
#define DHTPIN 25        // Pin GPIO donde está conectado el sensor DHT22
#define DHTTYPE DHT22    // Tipo de sensor utilizado (DHT11 o DHT22)
#define LDR_PIN 14       // Pin ADC donde está conectado el sensor de luz LDR
#define LED_LUZ 17       // GPIO para el LED indicador de luz
#define PIN_VENT 13      // Pin GPIO conectado al módulo MOSFET que controla el ventilador
#define TEMP_UMBRAL 30.0  // Umbral de temperatura crítica para activar el ventilador
#define LUZ_UMBRAL 3000   // Umbral de luz para encender el LED

// Configuración de WiFi
const char* ssid = "Tu_SSID";          // Nombre de la red WiFi
const char* password = "Tu_Contraseña";  // Contraseña de la red WiFi

// Configuración de MQTT
const char* mqtt_server = "192.168.0.10"; // Dirección IP del broker MQTT
const char* topic_temp = "home/temp";    // Tópico para publicar datos de temperatura
const char* topic_luz = "home/luz";      // Tópico para publicar datos de luz
```

```
// Configuración de Telegram
const char* botToken = "Tu_Token_Bot_Telegram"; // Token del bot de Telegram
const char* chatID = "Tu_Chat_ID";           // ID del chat donde se enviarán mensajes

// Inicialización de librerías y objetos
DHT dht(DHTPIN, DHTTYPE);         // Objeto para interactuar con el sensor DHT
WiFiClientSecure clientTelegram;  // Cliente seguro para comunicación HTTPS
UniversalTelegramBot bot(botToken, clientTelegram); // Objeto del bot de Telegram
WiFiClient espClient;             // Cliente WiFi para MQTT
PubSubClient client(espClient);   // Cliente MQTT

void setup() {
  Serial.begin(115200);             // Iniciar comunicación serial para depuración
  dht.begin();                      // Iniciar el sensor DHT
  pinMode(LDR_PIN, INPUT);          // Configurar pin del sensor LDR como salida
  pinMode(PIN_VENT, OUTPUT);        // Configurar el pin del ventilador como salida
  pinMode(LED_LUZ, OUTPUT);         // Configurar el pin del LED como salida
  setupWiFi();                      // Conectar a la red WiFi
  client.setServer(mqtt_server, 1883); // Configurar el servidor MQTT
}

void setupWiFi() {
  delay(10);
  Serial.println("Conectando a WiFi...");
  WiFi.begin(ssid, password); // Iniciar conexión con las credenciales de red
  while (WiFi.status() != WL_CONNECTED) {
    delay(1000);            // Esperar hasta que la conexión se establezca
```

```
    Serial.print(".");        // Indicar progreso con puntos en el monitor serial
  }
  Serial.println("\nConexión WiFi establecida");
  clientTelegram.setCACert(TELEGRAM_CERTIFICATE_ROOT); // Configurar certificado para Telegram
}

void reconnect() {
  // Reconectar al broker MQTT si la conexión se pierde
  while (!client.connected()) {
    Serial.print("Conectando al broker MQTT...");
    if (client.connect("ESP32_Client")) { // Intentar conexión
      Serial.println("Conectado al broker MQTT");
    } else {
      Serial.print("Fallo, rc=");
      Serial.print(client.state());       // Mostrar estado de la conexión
      Serial.println(" Intentando de nuevo en 5 segundos");
      delay(5000);
    }
  }
}

void loop() {
  if (!client.connected()) {
    reconnect(); // Reconectar si la conexión MQTT se pierde
  }
  client.loop(); // Mantener la conexión MQTT

  // Leer sensores
  float temp = dht.readTemperature(); // Temperatura en ºC
```

```
  float luz = analogRead(LDR_PIN);    // Leer valor analógico del sensor LDR

  // Verificar errores en la lectura del DHT
  if (isnan(temp)) {
    Serial.println("Error al leer el sensor DHT");
    return;
  }

  // Control del ventilador
  if (temp > TEMP_UMBRAL) {
    digitalWrite(PIN_VENT, HIGH); // Activar ventilador si la temperatura supera el umbral
    bot.sendMessage(chatID, "Temperatura alta: ventilador encendido", "Markdown");
  } else {
    digitalWrite(PIN_VENT, LOW);  // Desactivar ventilador si la temperatura está dentro del rango
    bot.sendMessage(chatID, "Temperatura normal: ventilador apagado", "Markdown");
  }

  // Control del LED basado en la luz ambiental
  if (luz < LUZ_UMBRAL) {
    digitalWrite(LED_LUZ, HIGH); // Encender LED si la luz es insuficiente
    bot.sendMessage(chatID, "Iluminación baja: LED encendido", "Markdown");
  } else {
    digitalWrite(LED_LUZ, LOW);  // Apagar LED si la iluminación es suficiente
    bot.sendMessage(chatID, "Iluminación adecuada: LED apagado", "Markdown");
  }

  // Publicar datos en MQTT
```

```
  client.publish(topic_temp, String(temp).c_str()); // Publicar temperatura
  client.publish(topic_luz, String(luz).c_str());   // Publicar nivel de luz

  // Mostrar datos en el monitor serial
  Serial.print("Temperatura: "); Serial.print(temp); Serial.print(" °C, ");
  Serial.print("Luz: "); Serial.println(luz);

  // Escuchar comandos en Telegram
  int numNewMessages = bot.getUpdates(bot.last_message_received + 1);
  for (int i = 0; i < numNewMessages; i++) {
    String text = bot.messages[i].text; // Leer comando recibido
    if (text == "/ON_FAN") {
      digitalWrite(PIN_VENT, HIGH);     // Encender ventilador desde Telegram
      bot.sendMessage(chatID, "Ventilador encendido", "Markdown");
    } else if (text == "/OFF_FAN") {
      digitalWrite(PIN_VENT, LOW);      // Apagar ventilador desde Telegram
      bot.sendMessage(chatID, "Ventilador apagado", "Markdown");
    } else if (text == "/ON_LED") {
      digitalWrite(LED_LUZ, HIGH);      // Encender LED desde Telegram
      bot.sendMessage(chatID, "LED encendido", "Markdown");
    } else if (text == "/OFF_LED") {
      digitalWrite(LED_LUZ, LOW);       // Apagar LED desde Telegram
      bot.sendMessage(chatID, "LED apagado", "Markdown");
    }
  }

  delay(5000); // Intervalo de 5 segundos entre lecturas
}
```

JOSÉ LUIS ORTIZ – FERNANDO XAVIER VALENCIA – CARLOS PATRICIO BOSMEDIANO – JESSICA XIMENA TORRES – ALEX DANILO BASTIDAS – LETTY MAGDALENA GARCÍA

ALUMNI EDITORA
2024

PRIMERA EDICIÓN

INTERNET DE LAS COSAS: MANUAL DE PRÁCTICAS PARA EL AULA Y EL LABORATORIO

ISBN: 978-9942-7307-4-9

INSTITUTO SUPERIOR TECNOLÓGICO
17 DE JULIO