

"Año del Fortalecimiento de la Soberanía Nacional"

Universidad Nacional Mayor de San Marcos

(Universidad del Perú, Decana de América)

Facultad de Ingeniería de Sistemas e Informática



Asignatura

Internet de las Cosas

Sección

1

Profesora

Rosas Cueva Jessica

Estudiantes

Barrientos Flores Sharon Camila	19200173
Chinchay Epiquin, Rodrigo Francisco	20200166
Flores Coronel, Elvis	18200023
Ñacari Elescano, Alan Jesus Valentino	19200035
Rodriguez Aybas, Jean Carlo	18200226

Lima, Perú

2022

ÍNDICE

CAPÍTULO I : INTRODUCCIÓN	3
1.1. Revisión del estado del arte	3
1.2. Planteamiento del problema	5
1.3. Objetivos del proyecto	6
CAPÍTULO II : MARCO TEÓRICO	7
2.1. Sistema hidropónico	7
2.2. Requerimientos de cultivo de tomate	8
2.3. IOT	10
2.4. Node Red	10
2.5. Base de datos	10
2.6. Broker Mosquitto	10
2.7. Arduino	11
CAPÍTULO III : COMPONENTES DEL SISTEMA DE RIEGO	12
3.1 ESP32	12
3.2 Sensor DTH11	13
3.3 Bomba de agua sumergible	14
3.4 Relé HW-482	15
3.5 Protoboard	16
CAPÍTULO IV : IMPLEMENTACIÓN DEL SISTEMA	18
CAPÍTULO V : RESULTADOS	29
CAPÍTULO VI : CONCLUSIONES	31
7. Bibliografía	32
8. Anexos	33

CAPÍTULO I : INTRODUCCIÓN

1.1. Revisión del estado del arte

–Design and Implementation of IoT based Automated Tomato Watering System Using ESP8266–

Rossy Nurhasanah, Lira Savina1, Zul Mahadi Nata y Ilham Zulkhair. Junio 2021. Medan, Indonesia. Publicado bajo la licencia de IOP Publishing Ltd. Journal of Physics: Conference Series. Volume 1898. 5 th International Conference on Computing and Applied Informatics (ICCAI 2020).

OBJETIVO: Diseñar e implementar un sistema de riego para tomates preciso, basado en IoT, y asu vez, en la humedad del suelo y la temperatura del aire, que se pueda controlar de forma remota mediante el uso de Internet, específicamente, a través del bot de la aplicación Telegram Messanger.

FUNCIONAMIENTO: Luego de haber configurado el bot de Telegram, y realizar la conexión con el módulo de ESP8266. El sistema encenderá la bomba de agua para regar automáticamente los cultivos de tomate si la humedad del suelo es menor al 60%. En caso de que la temperatura del aire sea mayor a 25 C° se en encenderá el ventilador. Posteriormente, se enviará una notificación mediante Telegram. El sistema también permite el control manual.

CONCLUSIONES: El sistema permitirá a los agricultores controlar y monitorear las plantas de tomate desde cualquier lugar a través de la aplicación Telegram. Sin embargo, se requiere siempre una conexión a Internet para su uso.

–Automated Irrigation Systems for Wheat and Tomato Crops in Arid Regions–

Hussein M. Al-Ghabari1, Fawzi S Mohammad, Mohamed SA El Marazky y Ahmed Zakaria Dewidar. Abril 2017. Publicado bajo la licencia de CC BY 4.0 Water SA vol.43 n.2 Pretoria.

OBJETIVO: Medir el rendimiento, durante 2 temporadas sucesivas, de un sistema automático de riego por goteo y aspersión, en condiciones climáticas extremadamente áridas para cultivos de tomate y trigo haciendo uso de un Módulo de

Evapotranspiración, y compararlo con un sistema de riego convencional.

RESULTADOS: Los cultivos regados con el sistema de riego automatizado presentaron un mayor nivel de ahorro de agua. Se llegó a utilizar hasta un 26% menos de agua que el sistema de riego convencional. Además, con el riego automático se obtuvieron características agronómicas superiores en lo que se refiere a la planta de Tomate: Altura, número de frutos, longitud y peso promedio del fruto.

CONCLUSIONES: Los sistemas de riego automáticos ofrecen un valiosa ventaja para el riego de plantas de trigo y tomate, pudiendo extenderse a otros cultivos similares. Se recomienda el uso de esta técnica para la cosecha de cultivos en zonas áridas, ya que proporciona sostenibilidad a largo plazo. Además, ofrecen un beneficio económico debido al ahorro sustancial de agua.

– Diseño e Implementación de un Sistema de Riego Automatizado y Monitoreo de Variables Ambientales mediante IoT en los Cultivos Urbanos de la Fundación Mujeres Empresarias Marie Poussepín –

Valeria Cortes Cadavid y Marco Fabian Vargas Garcia. 2020. Universidad Católica De Colombia, Facultad De Ingeniería Programa Ingeniería Electrónica Y Telecomunicaciones, Bogotá.

OBJETIVO: Desarrollar un sistema de riego automático basado en IoT, que sea capaz de monitorear las variables ambientales (Humedad del suelo, temperatura y humedad ambiental) de los cultivos de tomate Cherry de la Fundación Mujeres Empresarias Marie Poussepín, a través de una aplicación móvil. Y así, poder optimizar el uso del agua y mejorar el nivel de productividad.

FUNCIONAMIENTO: Luego de que se realice la lectura de los sensores. Se podrá visualizar la información de las variables en ThingSpeak. A su vez; se imprime un una pantalla LCD 16x2. Si se detecta que la humedad es menor al 65% se encenderá la electroválvula y regará los cultivos.

CONCLUSIONES: El realizar la cosecha en una zona controlada, mejora la calidad del cultivo y evita la propagación de plagas. Así mismo, el uso de un sistema automatizado y el monitoreo de las variables, optimizó el uso de agua, y permitió un mejor cuidado del cultivo. Fue necesario la creación de un manual de uso, ya que los usuarios finales eran personas con escasos conocimientos en el área.

–Desarrollo de un prototipo de sistema de riego automático para el cultivo de tomates

rojos: Caso San Pedro Apóstol Oaxaca–

C. Jessica Sánchez Arrazola. Enero 2017. Universidad Autónoma del Estado de México. Ecatepec de Morelos, Estado de México.

OBJETIVO: Diseñar un prototipo de riego automatizado de cultivo de tomate rojo, para controlar el agua que se bombea en base a la temperatura y humedad, en el invernadero del Sr. Hugo Pérez. Y posteriormente, mediante un programa en Java, poder visualizar la información recopilada por los sensores y exportar a un archivo en Excel.

FUNCIONAMIENTO: Tras tomar la lectura de las variables ambientales, estas se imprimirán en dos LCD 16x2. Uno, mostrando la temperatura y humedad relativa, mientras que el otro, indicará el estado de la humedad del suelo. Si el sensor de huemad detecta un valor mayor a 800, la bomba de agua se activará. A su vez, podrá realizarse el monitoreo y visualización de las variables, mediante gráficas en un programa en Java.

CONCLUSIONES: El desarrollo del prototipo resulta de fácil acceso económico hablando. Sin embargo, al no poseer una conexión inalámbrica limita el acceso a los datos por parte del usuario final.

1.2. Planteamiento del problema

El proyecto se centra en la implementación de un circuito de riego automatizado, donde se realizará un diseño de una arquitectura IoT para el control de sistemas hidropónicos así como el desarrollo de una aplicación web para facilitar la configuración y monitorización del sistema.

El motivo del presente proyecto viene con el fin de profundizar en el desarrollo de aplicaciones que unan hardware y software para así desarrollar herramientas de utilidad que hagan más fácil la vida de la gente. Como motivación principal es que llevamos el curso de Internet de las Cosas, nos llamó la atención el hecho de integrar tecnología en objetos de la vida cotidiana para que se conecten entre ellos a través de Internet, junto a un interés social en una alimentación más sana a través de cultivar en casa.

También notamos que un problema de la agricultura extensiva es que necesitan

grandes cantidades de tierra , las cuales no pueden abastecerse correctamente de un sistema de riego óptimo .

En efecto, hay un deterioro progresivo del suelo en los invernaderos y las zonas de producción en general, debido al uso excesivo de agua lo cual genera erosión en el campo agrícola provocando la pérdida de nutrientes, también hay un bajo control y prevención de la humedad del suelo, donde impide el crecimiento óptimo de los cultivos, obliga a los agricultores a optar por otros métodos de cultivos para solucionar estos problemas minimizando todo tipo de pérdidas.

Por lo tanto, se propone hacer una combinación de todos los temas llevados a cabo en el curso, con el objetivo de desarrollar un sistema hidropónico desde cero, creando una aplicación web para controlar el estado del sistema, analizar datos recopilados entre los distintos componentes mediante los protocolos adecuados y promover los cultivos sostenibles en los huertos.

1.3. Objetivos del proyecto

Implementar un sistema de riego basado en IoT, para abastecer de nutrientes y agua necesarios nuestra maceta casera hidropónica

CAPÍTULO II : MARCO TEÓRICO

Este capítulo contempla una descripción detallada del proyecto, en primer lugar se definirá sobre el sistema hidropónico, así como sus beneficios en los cultivos. Así mismo se explicarán los requerimientos de la planta y por último la tecnología implementada para su desarrollo.

2.1. Sistema hidropónico

2.1.1. Definición

Según Chow(2017) el sistema hidropónico es una nueva tecnología de cultivo que aplica soluciones nutritivas a través de un soporte artificial, esta tecnología ofrece una capacidad de reutilizar el agua y nutrientes. Así mismo Iberdrola (2021) argumenta que este sistema es una opción más sostenible frente a la agricultura tradicional debido a su escaso uso de recursos.

2.1.2. Beneficios

El cultivo hidropónico presenta ventajas económicas, sociales y ambientales en comparación a la agricultura común.

De acuerdo con (Moreno & Reyes, 2016), las ventajas se clasifican en mecánicas, ambientales y económicas:

Beneficios mecánicos

- Mayor control de calidad en el producto final. Esto permite que durante el proceso de crecimiento de la planta, se intervenga con los nutrientes faltantes para mejorar su calidad
- Aumento del número de cosechas por año. Este resultado se obtiene por los pocos recursos que se necesitan para implementar este sistema.
- Facilidad de uso por su automatización en el riego. El sistema permite controlar la cantidad de agua y nutrientes que necesita la planta

Beneficios ambientales

- Se utilizan subproductos o desechos orgánicos como sustratos
- Eco amigable por la no utilización de fertilizantes químicos y pesticidas

Beneficios económicos

- Ahorro en la mano de obra debido a su bajo uso de recursos
- Bajos costos para el control de plagas
- Incremento en los ingresos por la calidad de los cultivos

2.2. Requerimientos de cultivo de tomate

2.2.1. Temperatura

La temperatura es un factor importante para el desarrollo de la planta que influye en los procesos de germinación, crecimiento, floración, fructificación y maduración de los frutos.

Para el cultivo del tomate se consideran los siguientes rangos de temperatura para su correcto crecimiento:

Temperatura	Efectos sobre las plantas
Mínima 8-12° C	Dentro de este rango la temperatura tiende a afectar el procesos de toma de nutrientes y crecimiento; si la temperatura mínima se extiende en el tiempo la planta se debilita y se marchita, y si la temperatura llega en algún momento a estar por debajo de este valor, la planta sufre una progresiva decadencia posible muerte.
Óptima 21-27° C	Temperatura óptima para el desarrollo normal de la planta, los procesos bioquímicos se desarrollan normalmente; el crecimiento vegetativo, la floración y la fructificación son las esperadas.
Máxima 32-36° C	Los procesos bioquímicos y de toma de nutrientes están al máximo, son excesivos y agotadores para la planta, se detiene la floración; cuando estas temperaturas se extienden por un tiempo la planta muere.

Fuente: JARAMILLO, Jorge, Rodríguez, Viviana, Guzmán, Miryam, Zapata, Miguel, Rengifo, Teresita. Buenas Prácticas Agrícolas (BPA) en la producción de tomate bajo condiciones protegidas. Medellín, Colombia: CTP Print Ltda, 2007, p. 72

Según (Cortes & Vargas, 2020) los factores de riesgo que genera la exposición de la planta a altas temperaturas son:

- Variación en la elongación del ovario, lo que genera una deformación en el fruto.
- Variación en los estambres, lo que genera una mala polinización de la planta.
- Reducción en la formación de flores y frutos por inflorescencia

Así mismo (Cortes & Vargas, 2020) mencionan los factores de riesgo que conlleva la exposición a altas temperaturas de la planta, estos son:

- Reducción en la producción de polen, flores y frutos
- Mala fecundación de los frutos
- Asimetría en la forma de la inflorescencia

2.2.2. Humedad

La humedad es la cantidad de agua por volumen de tierra que hay en el terreno, este es uno de los factores que permiten el correcto desarrollo y crecimiento de la planta.

Citando a (Cortes & Vargas, 2020) la humedad ideal para el desarrollo del cultivo de tomate debe estar entre un 65% y 75% para su óptimo crecimiento y obtención final de un producto de buena calidad para su consumo y comercialización.

2.2.3. Ventilación

La ventilación es uno de los factores importantes para las plantas ya que según las corrientes de los vientos si son cálidos o fríos, se ve afectada la etapa de floración.

Este tipo de control en los cultivos se implementa en los invernaderos. Algunos de los beneficios que se obtienen son la producción de cosechas fuera de época debido al control que se tiene en los factores climáticos que inciden en el desarrollo de la planta, tal como la lluvia, caída del sol y circulación del aire (Cortes & Vargas, 2020).

2.2.4. Riego

Según (Cortes & Vargas, 2020) el tomate cherry tiene una elevada masa foliar, por lo cual debe estar en un estado óptimo de humedad, por ello recomiendan para esta planta el riego por goteo por sus diversas ventajas que son las siguientes:

- Esta modalidad de riego garantiza una aplicación exacta y localizada del agua a la raíz de la planta.
- Brinda un equilibrio entre el aire y agua que necesita la zona exacta de riego, lo que genera un beneficio en el ahorro de agua.
- Disminuye la infestación de malezas ya que con este tipo de riego se

evita el exceso de humedad en el terreno y por ende el desarrollo de las malezas.

2.3. IOT

2.3.1. Definición

Es la interconexión en red de diferentes objetos cotidianos, desarrollados por algún tipo de inteligencia que les permita responder a las necesidades de las personas. Esta tecnología es una evolución del Internet que añade interconectividad y una mejor percepción de la información.

Según (Salazar & Silvestre, 2018) la IOT ofrece una gran cantidad de oportunidades en el acceso a los datos, servicios como la educación, seguridad, transporte, entre otros. Esta tecnología ofrece una amplia distribución de red en dispositivos inteligentes que pueden ser personalizados según la necesidad del usuario.

2.4. Node Red

Esta herramienta de programación es un servicio creado por IBM que tiene como objetivo la integración de un hardware con otros servicios de internet; permite hacer uso de tecnologías complejas a través de una interfaz de fácil acceso.

Se trata de un motor de flujos con enfoque IOT para definir flujos de servicios a través de protocolos MQTT.

2.5. Base de datos

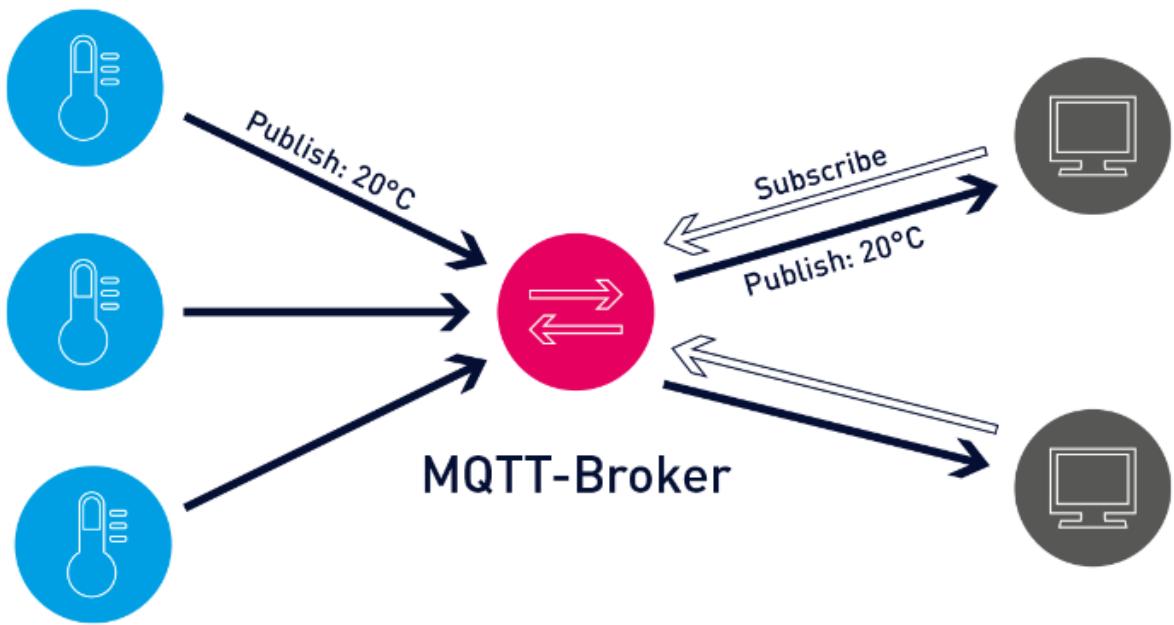
Es una recopilación de datos estructurados que se almacena en un sistema informático; su funcionamiento está controlado por un sistema de gestión de base de datos(DBMS) que requiere del uso de un software que sirva como interfaz entre la base de datos y programas a los usuarios finales.

2.6. Broker Mosquitto

MQTT es un protocolo controlado por eventos donde no hay transmisión de datos periódica o continua. Este protocolo de mensajería se utiliza principalmente para

comunicaciones de máquina a máquina (M2M) o conexiones del tipo internet de las cosas.

Un broker es el servidor con el que se comunican los clientes.



2.7. Arduino

Es una plataforma de código abierto de fácil uso y acceso para crear aplicaciones que permite conectar periféricos a las entradas y salidas de un microordenador, estos son circuitos integrados en los que se pueden grabar instrucciones con lenguaje de programación.

CAPÍTULO III : COMPONENTES DEL SISTEMA DE RIEGO

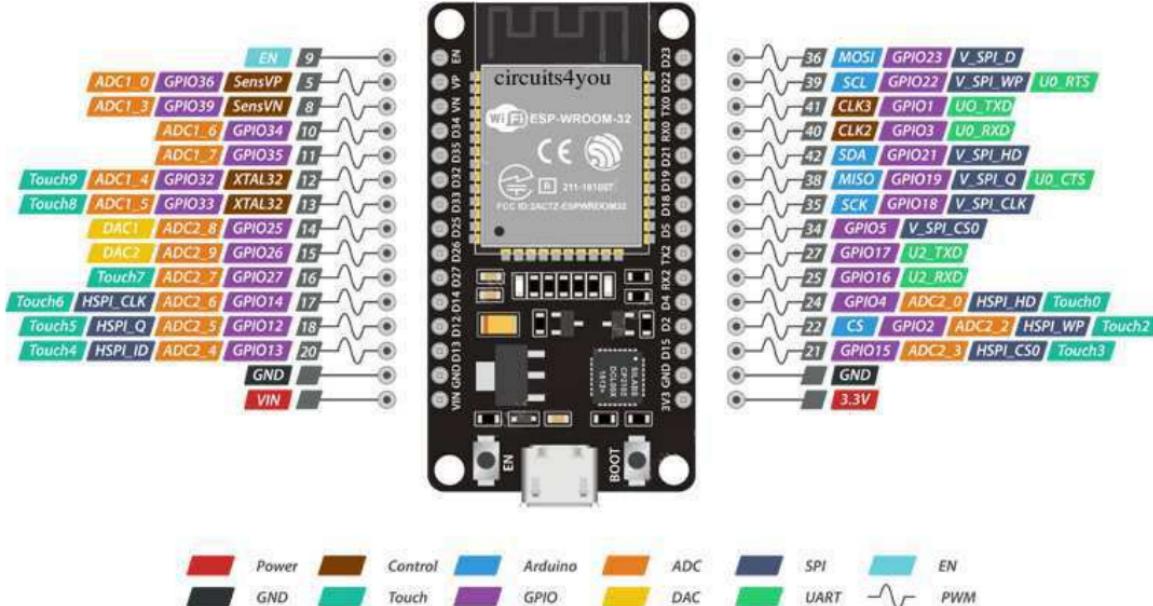
3.1 ESP32

El módulo ESP32 es un microcontrolador de bajo consumo y bajo costo. Este posee un microprocesador Tensilica Xtensa LX6 con una frecuencia hasta de 240 MHz. Además tiene conexión Wi-Fi y modo dual con Bluetooth.

De acuerdo a Bruno (2019): “ESP32 está altamente integrado con switch de antena , balun para RF, amplificador de potencia, amplificador de recepción con bajo nivel de ruido, filtros y módulos de administración de energía, totalmente integrados dentro del mismo chip!!“

Características principales:

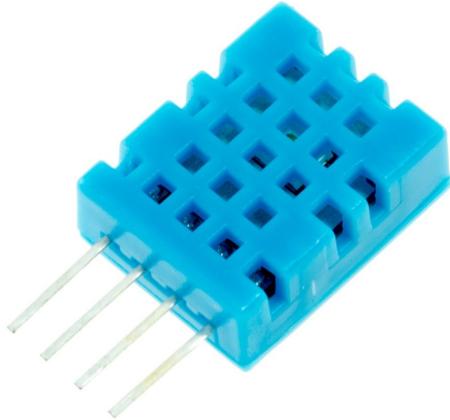
- Procesador principal: Tensilica Xtensa LX6 de 32 bits.
- Wi-Fi: 802.11 b / g / n / e / i (802.11n @ 2.4 GHz hasta 150 Mbit / s).
- Bluetooth: v4.2 BR / EDR y Bluetooth Low Energy (BLE).
- Frecuencia de Clock: Programable, hasta 240MHz.
- Rendimiento: hasta 600DMIPS.
- ROM: 448KB, para arranque y funciones básicas.
- SRAM: 520KiB, para datos e instrucciones



3.2 Sensor DTH11

El sensor DTH11 es un sensor digital que capta la temperatura y la humedad relativa. Es de bajo costo y fácil entendimiento, principalmente se emplea para fines académicos. El DTH11 integra un termistor para captar la temperatura del aire y un sensor de humedad. Los datos son mostrados por la señal digital en el pin de datos.

Para su funcionamiento se debe conectar el pin VCC de alimentación a 3-5V, el GND a Tierra (0V) y el pin que recolecta los datos se recolecta al pin digital del Arduino. La recolección de datos se hace cada 2 segundos.



ESPECIFICACIONES TÉCNICAS

- Voltaje de Operación: 3V - 5V DC
- Rango de medición de temperatura: 0 a 50 °C
- Precisión de medición de temperatura: ±2.0 °C
- Resolución Temperatura: 0.1°C
- Rango de medición de humedad: 20% a 90% RH.
- Precisión de medición de humedad: 5% RH.
- Resolución Humedad: 1% RH
- Tiempo de sensado: 1 seg.
- Interface digital: Single-bus (bidireccional)
- Modelo: DHT11
- Dimensiones: 16*12*5 mm
- Peso: 1 gr.
- Carcasa de plástico celeste

3.3 Bomba de agua sumergible

La bomba de agua sumergible crea un flujo de hasta 2L por minuto (70-120 L/h). Tiene un funcionamiento de 2,5v a 6v DC por medio USB. Incluye un motor interno de 800mA.



ESPECIFICACIONES TÉCNICAS

- Voltaje de funcionamiento : 2,5-6v DC.
- Altura bombeo máx.: 40-110 cm.
- Caudal bombeo máx.: 80-120 l/h.
- Diámetro salida Exterior: 7,5 mm.
- Diámetro salida Interior: 5 mm.
- Longitud cable: 20 cm.
- Tiempo continuo de trabajo; 500 horas.

3.4 Relé HW-482

El relé HW-482 es un interruptor electromagnético que está en función de la bobina y el electroimán.



ESPECIFICACIONES TÉCNICAS

- Módulo de relé: 5V
- Señal de control TTL: 5V-12V
- Voltaje máximo en AC: 220V

3.5 Protoboard

El protoboard es una placa de pruebas relacionadas a la electrónica. Su estructura llena de orificios permite la conexión de distintos componentes electrónicos y cables para trasladar la corriente eléctrica.

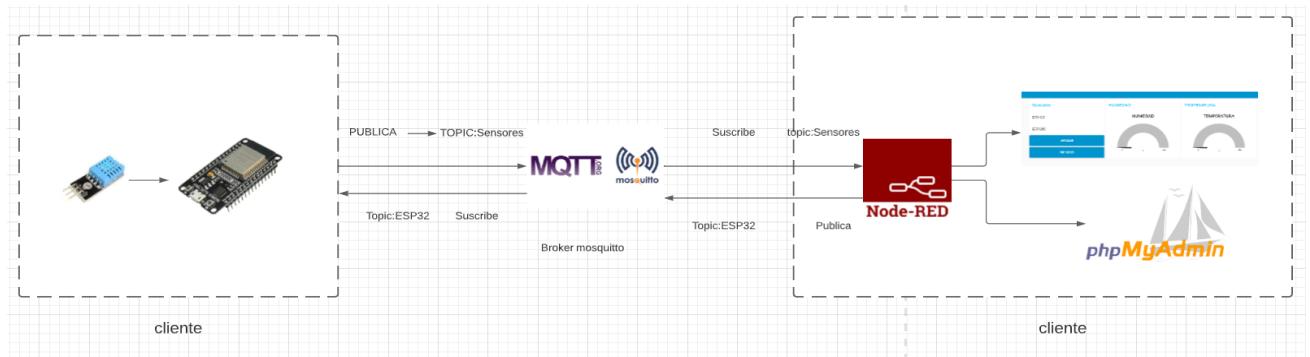


Características :

- Puntos: 830 puntos.
- Color: Blanco.
- Material: Plástico ABS.
- Compatible con cualquier componente o cable de 20-29 AWG (0.3-0.8mm)
- Longitud: 16.5 cm.
- Ancho: 5.5 cm.
- Altura: 1 cm.

CAPÍTULO IV : IMPLEMENTACIÓN DEL SISTEMA

- Primero mediante Node RED y MQTT preparamos nuestro servidor local con el puerto 9000 para poder conectar nuestro arduino al mismo puerto.

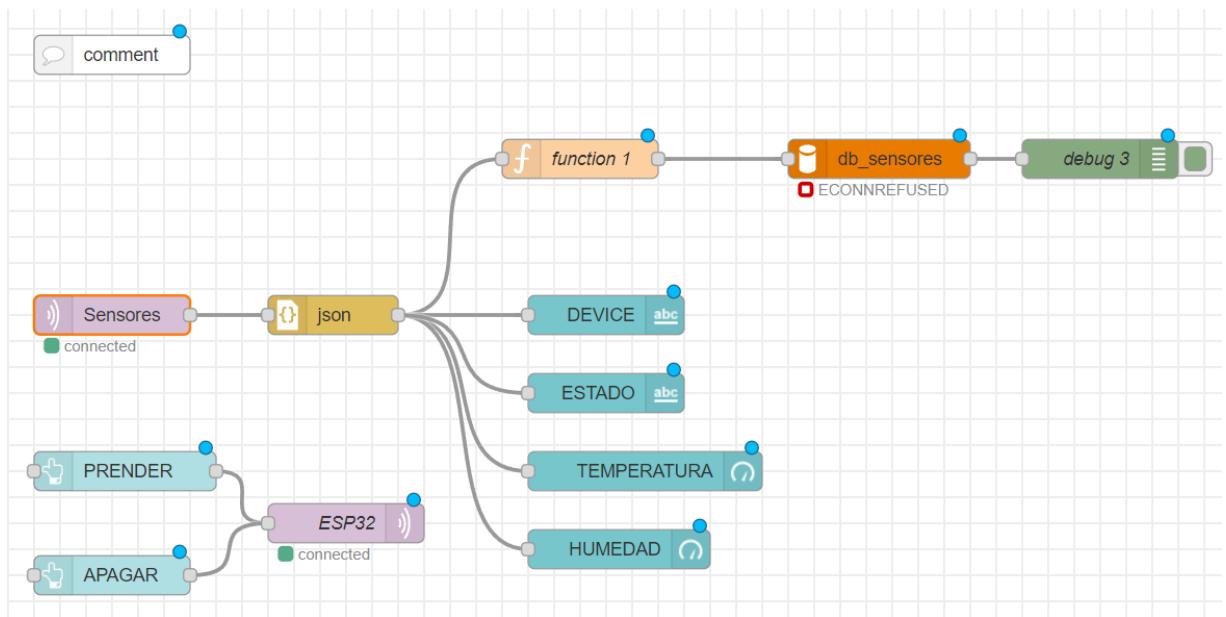


- Luego todo esto será procesado y llevado a PHPMyAdmin para poder generar gráficos estadísticos.

BASE DE DATOS

#	Nombre	Tipo	Cotejamiento	Atributos	Nulo	Predeterminado	Comentarios	Extra	Acción
1	ID	int(11)			No	Ninguna		AUTO_INCREMENT	
2	FECHA	datetime			No	current_timestamp()			
3	DEVICE	varchar(50)	utf8mb4_general_ci		No	Ninguna			
4	TEMPERATURA	float			No	Ninguna			
5	HUMEDAD	float			No	Ninguna			

- Se hizo la construcción de los nodos del Node Red para el envío de los datos.



Construcción de cada uno de los nodos:

Properties	
<input type="button" value="Name"/>	Name
<input type="button" value="Connection"/> <input type="button" value="Security"/> <input type="button" value="Messages"/>	
Server	broker.emqx.io
<input checked="" type="checkbox"/> Connect automatically	Port 1883
<input type="checkbox"/> Use TLS	
<input type="button" value="Protocol"/>	MQTT V3.1.1
<input type="button" value="Client ID"/>	Leave blank for auto generated
<input type="button" value="Keep Alive"/>	60
<input type="button" value="Session"/>	<input checked="" type="checkbox"/> Use clean session

Con este nodo se obtiene la información de los sensores para enviarlos a la base de datos.

The screenshot shows the Node-RED interface with two configurations:

- Top Configuration (json Node):**
 - Action:** Always convert to JavaScript Object
 - Property:** msg. payload
 - Name:** Name
- Bottom Configuration (function 1 Node):**
 - Name:** function 1
 - Script (On Message):**

```

1 if (msg.payload.TEMPERATURA!=0 && msg.payload.HUMEDAD!=0 ){
2   var query = "INSERT INTO `datos-esp32` (`ID`, `FECHA`, `DEVICE"
3   query = query + msg.payload.DEVICE+',';
4   query = query + msg.payload.TEMPERATURA + ',';
5   query = query + msg.payload.HUMEDAD + ')';
6   msg.topic=query;
7
8 }
9
10
11 return msg;

```

En este proceso se transforman los datos obtenidos de los sensores en objetos Json.



Properties



Group

[Home] Group 3



Size

auto

Label

DEVICE

Value format

`{{msg.payload.DEVICE}}`

Layout

label value

label value

label value

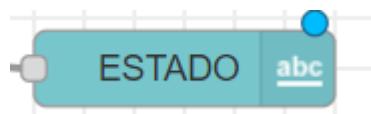
label value

label value

</> Class

Optional CSS class name(s) for widget

Name



Properties



Group

[Home] Group 3



Size

auto

Label

ESTADO

Value format

{{msg.payload.ESTADO}}

Layout

label value

label value

label value

label value

label value

Class

Optional CSS class name(s) for widget

Name



Properties



Group	[Home] Group 4	<input type="button" value="▼"/>	<input type="button" value="✎"/>				
Size	auto						
Type	Gauge	<input type="button" value="▼"/>					
Label	TEMPERATURA						
Value format	{{msg.payload.TEMPERATURA}}						
Units	°C						
Range	min <input type="text" value="0"/>	max <input type="text" value="50"/>					
Colour gradient							
Sectors	0	...	20	...	30	...	50
<input type="checkbox"/> Fill gauge from centre.							
</> Class	Optional CSS class name(s) for widget						
Name	<input type="text"/>						



Properties

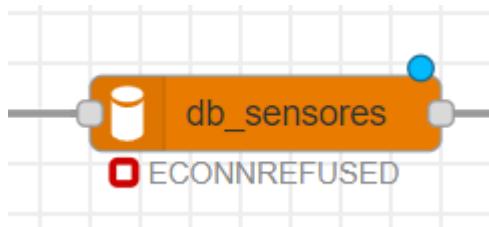


Group	[Home] Group 2	<input type="button" value="▼"/>	<input type="button" value="✎"/>	
Size	auto			
Type	Gauge	<input type="button" value="▼"/>		
Label	HUMEDAD			
Value format	{{{msg.payload.HUMEDAD}}}			
Units	%			
Range	min 0	max 100		
Colour gradient				
Sectors	0	... 70	... 80	... 100
<input type="checkbox"/> Fill gauge from centre.				
</> Class	Optional CSS class name(s) for widget			
Name	<input type="text"/>			

Para la conexión con la base de datos se instalo los nodos de MySQL



Y con esto instalado se procede a realizar la configuración



Properties

Host	127.0.0.1
Port	3308
User	
Password	
Database	db_sensores
Timezone	\pm hh:mm
Charset	UTF8
Name	Name

Tip: The timezone should be specified as \pm hh:mm or leave blank for 'local'.

La dirección que se muestra en el Host es una dirección de Lupa, lo que quiere decir que es una conexión así misma. Esto es debido a que MySql por términos de seguridad no está habilitado para conexiones remotas, pero como Node red se encuentra en la misma máquina, no habrá ningún inconveniente. El puerto de servicio de MySql es 3308. Luego se agrega el nombre del Database.



Properties

Output: msg. payload

To:

- debug window
- system console
- node status (32 characters)

Name: debug 3



Properties

Group: [Home] Group 3

Size: auto

Icon: optional icon

Label: PRENDER

Tooltip: optional tooltip

Color: optional text/icon color

Background: optional background color

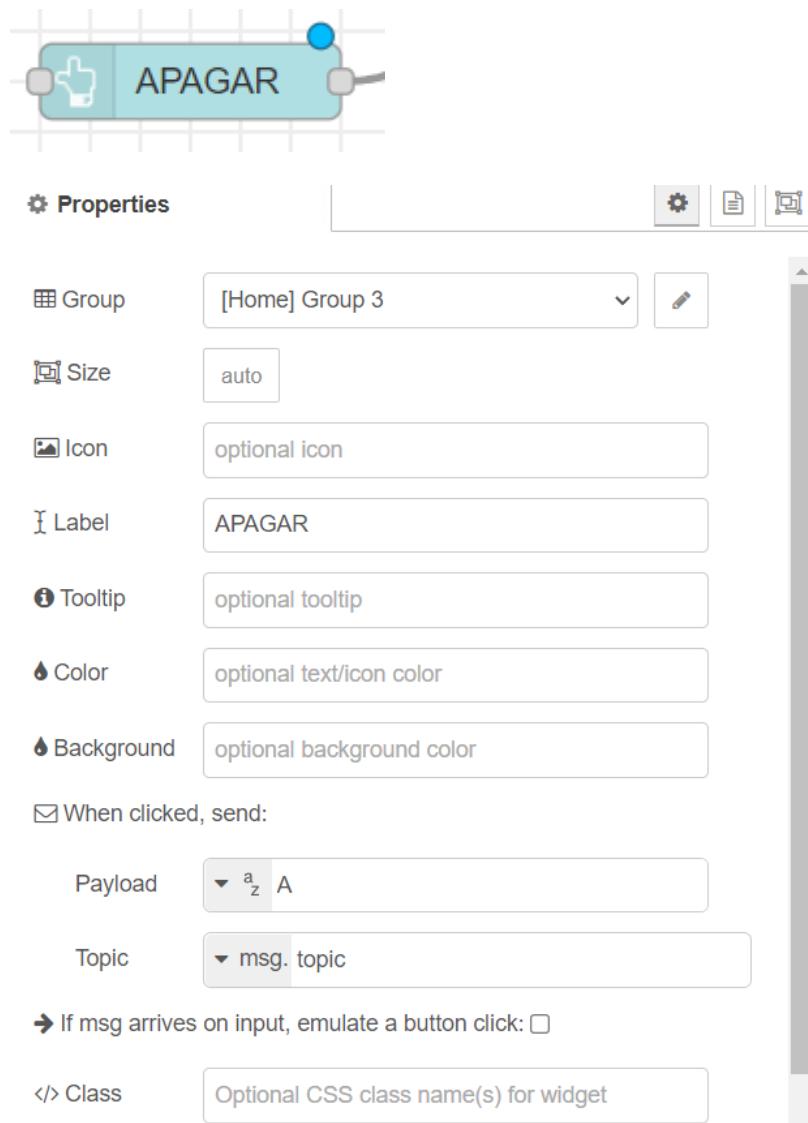
When clicked, send:

Payload: ▾ a_z P

Topic: ▾ msg. topic

➔ If msg arrives on input, emulate a button click:

</> Class: Optional CSS class name(s) for widget



Properties



🏷 Name

Name

Connection

Security

Messages

🌐 Server

broker.emqx.io

Port

1883

Connect automatically

Use TLS

⚙ Protocol

MQTT V3.1.1



🏷 Client ID

Leave blank for auto generated

❤️ Keep Alive

60

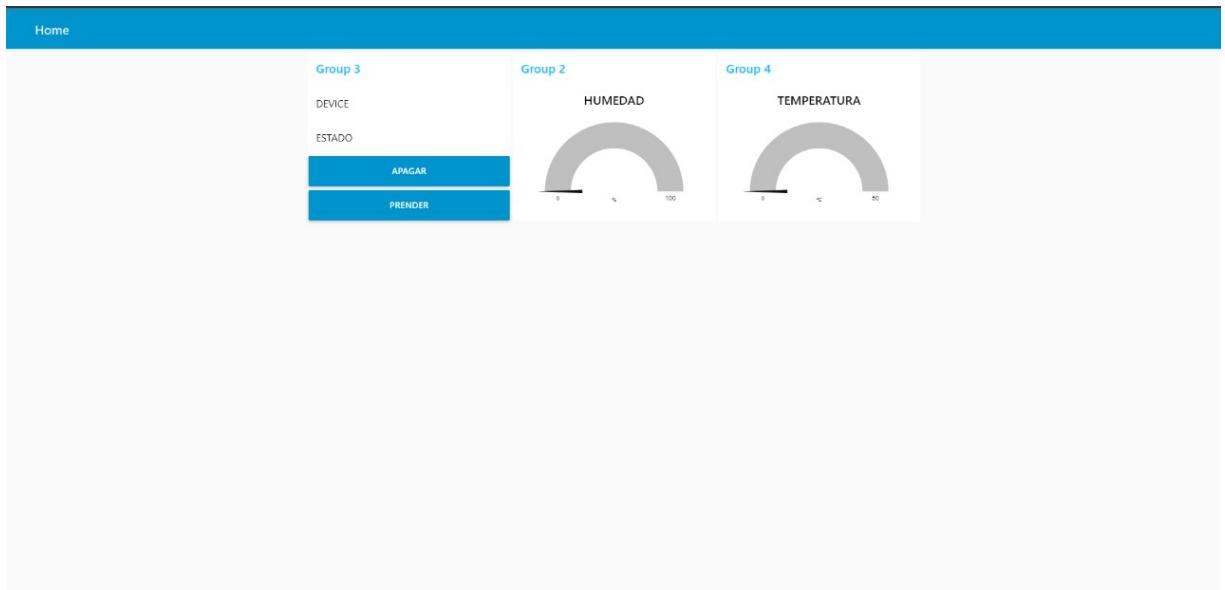
ℹ Session

Use clean session

CAPÍTULO V : RESULTADOS

Se obtuvieron los siguientes resultados:

Con la conexión a un servidor con el protocolo MQTT se obtuvo los siguientes resultados:



The screenshot shows the 'phpMyAdmin' interface with the URL 'localhost/phpmyadmin/index.php?route=/sql&server=1&db=db_sensores&table=datos-esp32&pos=0'. The left sidebar shows the database structure. The main area displays the 'datos-esp32' table with the following data:

ID	FECHA	DEVICE	TEMPERATURA	HUMEDAD
26	2022-12-29 23:32:21	ESP32	20	80
27	2022-12-29 23:32:23	ESP32	34	74
28	2022-12-29 23:32:25	ESP32	27	78
29	2022-12-29 23:32:27	ESP32	24	78
30	2022-12-29 23:32:29	ESP32	15	62
31	2022-12-29 23:32:31	ESP32	29	85
32	2022-12-29 23:32:33	ESP32	27	77
33	2022-12-29 23:32:35	ESP32	28	82
34	2022-12-29 23:32:37	ESP32	26	82
35	2022-12-29 23:32:40	ESP32	22	61
36	2022-12-29 23:32:41	ESP32	21	75
37	2022-12-29 23:32:43	ESP32	20	64
38	2022-12-29 23:32:45	ESP32	31	62
39	2022-12-29 23:32:47	ESP32	18	61
40	2022-12-29 23:32:49	ESP32	16	60
41	2022-12-29 23:32:51	ESP32	18	77
42	2022-12-29 23:32:53	ESP32	27	75
43	2022-12-29 23:32:55	ESP32	33	86
44	2022-12-29 23:32:57	ESP32	26	75
45	2022-12-29 23:32:59	ESP32	34	72
46	2022-12-29 23:33:01	ESP32	30	88
47	2022-12-29 23:33:03	ESP32	34	86
48	2022-12-29 23:33:05	ESP32	24	80
49	2022-12-29 23:33:07	ESP32	23	80

Por el lado de la conexión de manera local, el modo acces point ESP32



CAPÍTULO VI : CONCLUSIONES

- Se concluye que el control y monitoreo de la maceta hidropónica no solo se trata de recoger los datos por medio de los sensores, sino se da un tratamiento de ellos para poder presentarlo a los usuarios.
- La implementación de una aplicación web permite que el usuario pueda monitorizar los parámetros de su maceta.
- El uso del protocolo MQTT permitió recoger sencillamente los datos de los sensores y cumplir con los requerimientos del sistema.
- Se desarrolló un sistema de bajo costo y fácil implementación que permite el monitoreo, registro y control de variables físicas (Temperatura, humedad del ambiente) de una maceta hidropónica.
- Debido al modo en que se desarrolló el proyecto se puede agregar de manera fácil diferentes sensores y actuadores

7. Bibliografía

Chow, Y. N., Lee, L. K., Zakaria, N. A., & Foo, K. Y. (2017). New emerging hydroponic system. In *Symposium on Innovation and Creativity (iMIT-SIC)* (Vol. 2, pp. 1-4).

Iberdrola. (2021, April 22). QUÉ ES LA HIDROPONÍA Y SUS VENTAJAS. Iberdrola; Iberdrola.

<https://www.iberdrola.com/sostenibilidad/que-es-hidroponia-y-ventajas#:~:text=La%20hidropon%C3%ADA%20es%20un%20sistema,decir%2C%20prescinde%20de%20la%20tira>.

Moreno, A., & Reyes, J. L. (2016). Tópicos selectos de sustentabilidad: un reto permanente. México: AM Editores.

Salazar, J., & Silvestre, Y. (2018). INTERNET DE LAS COSAS.

https://upcommons.upc.edu/bitstream/handle/2117/100921/LM08_R_ES.pdf

Nurhasanah R. y otros, 2021 J. Phys.: Conf. Ser. 1898:

<https://iopscience.iop.org/article/10.1088/1742-6596/1898/1/012041/pdf>

Al-Ghobari, H.; El-Marazky, M.; Dewidar, A. Automated irrigation systems for wheat and tomato crops in arid regions. Abril 2017. Water SA vol.43 n.2 Pretoria:

<http://www.scielo.org.za/pdf/wsa/v43n2/18.pdf>

Cadavid, V. C., & Garcia, M. F. (2020). Diseño e Implementación de un Sistema de Riego Automatizado y Monitoreo de Variables Ambientales mediante IoT en los Cultivos Urbanos de la Fundación Mujeres Empresarias Marie Poussepín. Bogotá, Ecuador.

Arrazola, J. S. (2017). Desarrollo de un prototipo de sistema de riego automático para el cultivo de tomates rojos: Caso San Pedro Apóstol Oaxaca. Estado de México, México.

8. Anexos

CÓDIGO IMPLEMENTADO PARA CONECTAR A SERVIDOR

```
#include <ArduinoJson.h>
#include <WiFi.h>
#include <PubSubClient.h>
#include "DHT.h"
#define BUILTIN_LED 2
#define DHTPIN 14
#define DHTTYPE DHT11

// Variables con datos para la conexion a la red

const char* ssid = "LOS JARDINES";
const char* password = "orizon123";
const char* mqtt_server = "broker.emqx.io";
WiFiClient espClient;
PubSubClient client(espClient);
unsigned long lastMsg = 0;
#define MSG_BUFFER_SIZE      (50)
char msg[MSG_BUFFER_SIZE];
DHT dht(DHTPIN, DHTTYPE);
int value = 0;
String ESTADO;
const int relayPin=33;
bool releState=false;

void setup_wifi() {

    delay(10);
    // Empezamos conectandonos a una red WiFi
    Serial.println();
    Serial.print("Conectando ");
    Serial.println(ssid);

    WiFi.mode(WIFI_STA);
    WiFi.begin(ssid, password);

    while (WiFi.status() != WL_CONNECTED) {
        delay(500);
        Serial.print(".");
    }
}
```

```

randomSeed(micros());
//Imprime el ip address
Serial.println("");
Serial.println("WiFi connected");
Serial.println("IP address: ");
Serial.println(WiFi.localIP());
}

void callback(char* topic, byte* payload, unsigned int length) {
    Serial.print("Message arrived [");
    Serial.print(topic);
    Serial.print("] ");
    for (int i = 0; i < length; i++) {
        Serial.print((char)payload[i]);
    }
    Serial.println();

    // Encender el LED si se recibió un 1 como primer carácter
    if ((char)payload[0] == 'A') {
        digitalWrite(BUILTIN_LED, LOW); // Encienda el LED (tenga en cuenta que BAJO es
        el nivel de voltaje
        releState=false;

        // está activo bajo en el ESP32
    } else {
        digitalWrite(BUILTIN_LED, HIGH); // Apaga el LED haciendo que el voltaje sea ALTO
        releState=true;
    }
}

void reconnect() {
    // Bucle hasta que estemos reconectados
    while (!client.connected()) {
        Serial.print("Intentando la conexión MQTT... ");
        // Crear una identificación de cliente aleatoriaID
        String clientId = "ESP8266Client-";
        clientId += String(random(0xffff), HEX);
        // intento de conexión
        if (client.connect(clientId.c_str())) {
            Serial.println("conectado");

```

```

// Una vez conectado, publicar un anuncio...
client.publish("Sensores", "Estas conectado al servidor mqtt");
// se suscribe
client.subscribe("ESP32");
} else {
    Serial.print("fallido, rc=");
    Serial.print(client.state());
    Serial.println("inténtalo de nuevo en 5 segundos");
    // Espere 5 segundos antes de volver a intentarlo
    delay(5000);
}
}

void setup() {
pinMode(BUILTIN_LED, OUTPUT);    // Inicializa el led del esp32 en output
pinMode(relayPin,OUTPUT);        //Inicializa el pin del rele en output
digitalWrite(releState,HIGH);
Serial.begin(115200);
setup_wifi(); //Se conecta a la red con SSID y contraseña
client.setServer(mqtt_server, 1883); //Se conecta al servidor
client.setCallback(callback);
// Read humidity
float h = dht.readHumidity();
// Read temperature as Celsius
float t = dht.readTemperature();
delay(100);
}

void loop() {

if (!client.connected()) {
    reconnect();
}
client.loop();

unsigned long now = millis();
if (now - lastMsg > 2000) {
    lastMsg = now;

    // Lee la humedad
    float h = dht.readHumidity();
    // Lee la temp en Celsius
}
}

```

```

float t = dht.readTemperature();
//Condiciones para la activacion del riego
if (t >= 25 && h <= 20){
ESTADO="REGANDO";
digitalWrite(relayPin , LOW);
} else if (t >= 36 && h <= 65 ){
digitalWrite(relayPin , LOW);
ESTADO = "REGANDO";
}else if(releState){

ESTADO = "REGANDO";
digitalWrite(relayPin , LOW);
}else{
ESTADO = "APAGADO";
digitalWrite(relayPin,HIGH);
}

//Se crea el json con los datos del sensor
StaticJsonDocument<128> doc;

doc["DEVICE"] = "ESP32";
//doc["Anho"] = 2020;
//doc["Empresa"] = "Doge";
doc["TEMPERATURA"] = t;
doc["HUMEDAD"] = h;
doc["ESTADO"] = ESTADO;

String output;
//Serializa los datos del json
serializeJson(doc, output);

Serial.print("Publish message: ");
Serial.println(output);
//Se envian los datos serializados al servidor mosquito en el topico Sensores
client.publish("Sensores",output.c_str());

}
}

```

CÓDIGO IMPLEMENTADO PARA IMPLEMENTAR CONEXIÓN LOCAL

```
#include <WiFi.h>
#include "DHT.h"
#define DHTPIN 14
#define DHTTYPE DHT11
DHT dht(DHTPIN, DHTTYPE);
// Reemplace con sus credenciales de red
const char* ssid    = "ESP32";
const char* password = "12345678";

// Establezca el número de puerto del servidor web en 80
WiFiServer server(80);

// Variable para almacenar la solicitud HTTP
String header;
String ESTADO;
const int relayPin=33;
bool releState=false;

void setup() {
  Serial.begin(115200);

  pinMode(relayPin,OUTPUT);
  digitalWrite(releState,HIGH);
  // Conéctese a la red Wi-Fi con SSID y contraseña
  Serial.print("Configuración de AP (Punto de acceso)...");
  // Conéctese a la red Wi-Fi con SSID y contraseña
  WiFi.softAP(ssid, password);

  IPAddress IP = WiFi.softAPIP();
  Serial.print("AP IP address: ");
  Serial.println(IP);

  server.begin();
}

void loop(){
  WiFiClient client = server.available(); // Escuche los clientes entrantes
  // Leer humedad
  float h = dht.readHumidity();
  // Leer la temperatura como Celsius
```

```

float t = dht.readTemperature();
String t1= String(t);
String h1= String(h);
if (client) {           // Si un nuevo cliente se conecta,
    Serial.println("New Client."); // imprimir un mensaje en el puerto serie
    String currentLine = ""; // hacer una cadena para contener los datos entrantes
del cliente
    while (client.connected()) { // Bucle mientras el cliente está conectado
        if (client.available()) { // Si hay bytes para leer del cliente,
            char c = client.read(); // leer un byte, entonces
            Serial.write(c); // imprimirllo en el monitor serie
            header += c;
            if (c == '\n') { // si el byte es un carácter de nueva línea
                // si la línea actual está en blanco, tienes dos caracteres de nueva línea seguidos.
                // ese es el final de la solicitud HTTP del cliente, así que envíe una respuesta:
                if (currentLine.length() == 0) {
                    // Los encabezados HTTP siempre comienzan con un código de respuesta (por
ejemplo, HTTP/1.1 200 OK)
                    // y un tipo de contenido para que el cliente sepa lo que viene, luego una línea en
blanco:
                    client.println("HTTP/1.1 200 OK");
                    client.println("Content-type:text/html");
                    client.println("Connection: close");
                    client.println();

                    // enciende y apaga los GPIO
                    if (header.indexOf("GET /26/on") >= 0) {
                        Serial.println("GPIO 26 on");
                        ESTADO = "on";
                        releState=true;
                    } else if (header.indexOf("GET /26/off") >= 0) {
                        Serial.println("GPIO 26 off");
                        ESTADO = "off";
                        releState=false;
                    }

                    // Mostrar la página web HTML
                    client.println("<!DOCTYPE html><html>");
                    client.println("<head><meta name=\"viewport\" content=\"width=device-width," +
initial-scale=1\">");
                    client.println("<link rel=\"icon\" href=\"data:,\">");
                    // CSS para diseñar los botones de encendido/apagado

```

```

    client.println("<style>html { font-family: Helvetica; display: inline-block; margin: 0px
auto; text-align: center;}");
    client.println(".button { background-color: #4CAF50; border: none; color: white;
padding: 16px 40px;}");
    client.println("text-decoration: none; font-size: 30px; margin: 2px; cursor:
pointer;}");
    client.println(".button2 {background-color: #555555;}</style></head>");

    // Encabezado de página web
    client.println("<body><h1>ESP32 Web Server</h1>");

    // Muestra el estado actual y los botones ON/OFF para GPIO 26
    client.println("<p>rele - ESTADO " + ESTADO + "</p>");

    if (ESTADO=="off") {
        client.println("<p><a href=\"/26/on\"><button
class=\"button\">ON</button></a></p>");
    } else {
        client.println("<p><a href=\"/26/off\"><button class=\"button
button2\">OFF</button></a></p>");
    }

    client.println("<h2>Temperatura "+t1+"C</h2>");
    client.println("<h2>Humedad "+h1+"%</h2>");
    // La respuesta HTTP termina con otra línea en blanco
    client.println();
    // Salir del bucle while
    break;
} else { // si tiene una nueva línea, borre currentLine
    currentLine = "";
}
} else if (c != '\r') { // si obtuvo algo más que un carácter de retorno de carro,
    currentLine += c; // agregarlo al final de currentLine
}
}

// Borrar la variable de encabezado
header = "";
//Cerrar la conexión
client.stop();
Serial.println("Cliente desconectado.");
Serial.println("");

```

```
}

if (t >= 25 && h <= 20){

    digitalWrite(relayPin , LOW);
} else if (t >= 36 && h <= 65 ){

    digitalWrite(relayPin , LOW);

}else if(releState){

    digitalWrite(relayPin , LOW);
}

}else{

    digitalWrite(relayPin,HIGH);
}

}
```