Desarrollo de una plataforma en IoT para captura, almacenamiento, procesamiento y visualización de datos para monitoreo y trazabilidad en la producción de lechuga hidropónica.

Dahianna Mychell Hurtado Cubillos

20442016435

Universidad Antonio Nariño

Programa Ingeniería Electrónica

Facultad de Ingeniería Mecánica, Electrónica y Biomédica

Ibagué, Colombia

2023

Desarrollo de una plataforma en IoT para captura, almacenamiento, procesamiento y visualización de datos para monitoreo y trazabilidad en la producción de lechuga hidropónica.

Dahianna Mychell Hurtado Cubillos

Proyecto de grado presentado como requisito parcial para optar al título de:
Ingeniero electrónico

Director (a):
Ing. Sergio Orjuela

Línea de Investigación:
Nombrar la línea de investigación en la que se enmarca el trabajo de grado.

Universidad Antonio Nariño

Programa Ingeniería electrónica

Facultad de Ingeniería Mecánica, Electrónica y Biomédica

Ibagué, Colombia

2023

NOTA DE ACEPTACIÓN

El trabajo de grado titulado Desarrollo de una plataforma en IoT para captura, almacenamiento, procesamiento y visualización de datos para monitoreo y trazabilidad en la producción de lechuga hidropónica, Cumple con los requisitos para optar Al título de ingeniero electrónico.

_____

Firma del Tutor

_____

Firma Jurado

_____

Firma Jurado

Ibagué, 08 Noviembre 2023.

Contenido

Pág.

Lista de Figuras

Pág.

Lista de tablas

Pág.

(Dedicatoria)

A mis padres que con amor y esfuerzo me apoyaron siempre; a Dios por fortalecerme en los momentos más complejos, a mis familiares que aportaron en este proceso

Dahianna Mychell Hurtado Cubillos

Agradecimientos

Como primera medida a mis tutores los ingenieros Sergio Orjuela quien con paciencia y conocimiento me guiaron durante todo este proceso; a la Universidad Antonio Nariño por proporcionarme los recursos intelectuales y físicos para fortalecer mis conocimientos.

Resumen

El control y monitoreo de los procesos agroindustriales es hoy día, una herramienta muy efectiva para generar acciones en tiempo real para los productores, el proyecto presentado tiene como fin desarrollar una plataforma en IoT para captura, almacenamiento, procesamiento y visualización de datos para monitoreo y trazabilidad de variables de producción de lechuga hidropónica. La metodología empleada comprende una investigación aplicada diseñada en 5 etapas a partir de una investigación previa, la definición del sistema de comunicación, el diseño de la plataforma de datos, la visualización de variables y el funcionamiento del sistema; los resultados muestran que al integrar el NodeMCU basado en el ESP8266, la página web HTML, el servidor PHP y el servidor de base de datos phpMyAdmin se obtiene un mejor desempeño de datos para el agricultor.

Palabras clave: sistema IoT, servidor PHP, plataforma NodeMCU, Termohigrómetro dth 11, servidor de base datos.

Abstract

The control and monitoring of agro-industrial processes is today a very effective tool to generate real-time actions for producers, the project presented aims to develop an IoT platform to capture, store, process and visualize data for monitoring and traceability of hydroponic lettuce production variables. The methodology used comprises an applied research designed in 5 stages from a previous research, the definition of the communication system, the design of the data platform, the visualization of variables and the operation of the system; the results show that by integrating the NodeMCU based on the ESP8266, the HTML web page, the PHP server and the phpMyAdmin database server, a better data performance for the farmer is obtained.

Keywords: IoT system, PHP server, NodeMCU platform, dth 11 Thermohygrometer, database server.

Introducción

La transferencia tecnológica en la producción de cultivos orientada a mejorar el rendimiento de los mismos en función de poder controlar variables criticas asociadas a su proceso de germinación y maduración, es hoy día una tema de alta demanda en el sector agrícola; ejemplo de ello es el de la implementación de técnicas o métodos de producción como en el caso de los cultivos hidropónicos en donde las actividades agrícolas se simplifican y garantizan ambientes controlados; esto permite un mayor ahorro en consumos de agua, luz, tierra entre otros (Saavedra, 2018).

Sin embargo, esto no es posible si no se tienen las herramientas que permitan capturar, almacenar, procesamiento y visualización de datos para monitoreo y trazabilidad de variables de producción como por ejemplo crecimiento radicular, temperatura y humedad para su uso y análisis para optimizar la productividad.

Dentro de los estudios realizados que aportan soluciones para aumentar la eficiencia de los cultivos en cuanto a productividad a partir de IoT se tienen por ejemplo el propuesto por Gómez et al (2017), quienes realizaron un sistema IoT para el monitoreo de cultivos protegidos con la capacidad de recolectar información de parámetros relacionados con el desarrollo y crecimiento de los cultivos como humedad y temperatura. Para su elaboración utilizaron el protocolo MQTT empleando Paho un cliente de Python, así como un visualizador web para mostrar la información y capturar alertas relacionadas con el cultivo (Gómez, 2017).

La problemática abordada en el proyecto busca mediante Iot controlar variables asociadas al ciclo de desarrollo de la planta de lechuga en la finca las mercedes, dado que la temperatura y la humedad son claves como parámetros de crecimiento, de aquí la importancia

de que los datos disponibles de pueden monitorear y analizar en tiempo, lo cual permite un mejor desempeño del cultivo, menos cotos y optimización de insumos; la metodología empleada en este proyecto comprende una investigación aplicada diseñada en 5 etapas las cuales van desde la investigación previa, la definición del sistema de comunicación, preparación de la plataforma, visualización de variables y evaluación de la plataforma, el resultado del estudio plantea la integración del NodeMCU basado en el ESP8266, la página web HTML, el servidor PHP y el servidor de base de datos phpMyAdmin; el estudio concluye que transferencia de datos mediante sensor DHT11 es viable que estos pueden ser visualizados en tiempo real para el control de temperatura y humedad.

1. Capítulo 1

1.1 Planteamiento del problema

El desarrollo de tecnologías para comunicación, así como las aplicaciones de dispositivos electrónicos como los sistemas inalámbricos, bluetooth han impulsado a que los datos resultantes de las mediciones en tiempo real y para cualquier proceso estén disponibles en nubes de información, lo que permite generar nuevas formas de conexión facilitando la creación de redes globales y de ciberespacio para cualquier comunidad, esto es el uso de IoT el cual facilita una dinamismo entre los propios usuarios a tal punto que se sitúa como un actor clave de desempeño en múltiples tareas.

Sin embargo, en modelos de producción agrícola tradicional caracterizados por pequeños productores (fincas < 2.5 ha) cuyas barreras para acceder a tecnologías que les permitan tomar decisiones en cuanto a la trazabilidad del ciclo de producción de un cultivo resulta complejo, se pueden buscar alternativas bajo IoT que le permitan al agricultor fortalecer las experiencias y toma de decisiones basadas en las variaciones de factores críticos para garantizar un mejor de desempeño en los cultivos realizados en su granja.

Datos asociados al crecimiento de la planta, temperatura y humedad son fundamentales a la hora de desarrollar un cultivo en ambientes controlados como el caso de los métodos hidropónicos implementados en la finca las Mercedes como respuesta a mitigar las perdidas ocasionados por lo cambios climáticos en la producción de lechuga; sin embargo a pesar de lo anterior, el agricultor debe conocer la dinámica de evolución del cultivo según los datos de crecimiento, temperatura y humedad entre otras variables para este tipo de cultivos; lo anterior a la necesidad de poder mejorar los costos de producción en

un contexto en donde los elevados gastos de los insumos básicos como fertilizantes agua sumado a los de distribución y comercialización afectan el margen de rentabilidad, por ende la sostenibilidad de sus familias y proyectos de vida (Quintero, 2022).

Resulta importante realizar proceso para el control de variables asociadas al ciclo de desarrollo de la planta, dado que la temperatura y la humedad son claves como parámetros de crecimiento, de aquí la importancia de que los datos disponibles de pueden monitorear y analizar en tiempo real con el fin de que el productor pueda tomar decisiones en función de mantener el cultivo o suministrar insumos para asegurar el proceso de crecimiento, desarrollo y cosecha del producto.

De no controlarse estos parámetros, así como el no contar con información real de trazabilidad del mimo, se presentan riesgos de producción los cuales deben abordarse a través del uso de herramientas de interacción que le permita capturar, almacenar, procesar y visualizar datos que permitan realizar un adecuado proceso de monitoreo y trazabilidad de estas variables para la producción de lechuga.

Teniendo en cuenta lo anterior, surge el siguiente problema de investigación: ¿Qué factores y variables se deben considerar para garantizar la captura, almacenamiento, procesamiento y visualización de datos mediante el desarrollo de plataforma IoT para monitoreo y trazabilidad de parámetros de crecimiento para cultivos hidropónicos de lechuga en la finca las Mercedes de Neiva-Huila?

1.2 Objetivos

1.2.1 General

Desarrollar una plataforma en IoT para captura, almacenamiento, procesamiento y visualización de datos para monitoreo y trazabilidad de variables de producción de lechuga hidropónica.

1.2.2 Específicos

- Identificar las variables de almacenamiento, así como los requerimientos de comunicación inalámbrica y transmisión de datos.
- Identificar los componentes del sistema, así como la etapa de adquisición, control, comunicación y transmisión de datos inalámbrica con conexión a la nube para la plataforma basada en IoT.
- Establecer el protocolo de comunicación.
- Crear la estructura de almacenamiento de las variables en la base de datos MySQL y Visualizar en la página web.
- Evaluar y validar la plataforma en IoT en función de la gestion de datos (registros y gráficos de control).

1.3 Justificación

El IoT revoluciono la forma como se ven y operan las cosas en el contexto diario, mostro como a través de la comunicación y del manejo de los datos , estos pueden integrarse a partir de una red de equipos de cómputo, sistemas inalámbricos sensoricos (WSN), sistemas tecnológicos de comunicaciones, así como otros métodos de proceso de datos para el procesamiento de información clave en el control- proceso (Singh, 2019)

Las plataforma basadas en IoT, para aplicaciones agrícolas pueden tener un impacto muy positivo en el alcance de metas fijadas en cualquier ámbito, dado que se integran tecnologías que facilitan la trazabilidad a partir de la captura, almacenamiento, procesamiento y visualización de datos que consideran varios tipos de variables como el crecimiento de la planta, condiciones térmicas, humedad, suelos, variables fitosanitarias y nutricionales para el desarrollo de la planta (Kim, 2020). Así pues, la agricultura inteligente sostenible tiene como objetivo mantener la calidad del suelo, reducir la erosión del suelo, ahorro de los recursos hídricos y se ha aplicado al cultivo para preservar los recursos naturales sin comprometer la calidad de los requerimientos fundamentales, que además mitigan los problemas

Actualmente, el IoT es considerado un factor vital en la vida diaria, ofreciendo múltiples soluciones en diferentes ámbitos como en: el área de la salud, el comercio minorista, el tráfico, la seguridad, los hogares y ciudades inteligentes, la agricultura inteligente y de precisión, entre otros (Farooq, 2020), permitiendo el desarrollo de una infraestructura digital con potencial impacto en el desarrollo sostenible a nivel global.

Este proyecto desarrolla una plataforma basada en IoT, para captura, almacenamiento, procesamiento y visualización de datos para monitoreo y trazabilidad de variables de crecimiento, temperatura y humedad en agricultura de precisión, que permita

tomar decisiones en tiempo real y de esta manera garantizar una producción de calidad y con mínimas perdidas para el agricultor de la finca las Mercedes; adicionalmente el contar con una plataforma propia para la Universidad Antonio Nariño que permita el captura, almacenamiento, procesamiento y visualización de datos vía Web para que los estudiantes puedan disponer de este recurso para el desarrollo de proyectos de innovación como soporte en la estructuración de futuros proyectos de grado.

2. Capítulo 2

A continuación, se presenta el capítulo 2 en el cual se establece la descripción de estudios actuales, así como el marco teórico del estudio, fundamental en el desarrollo de propuestas definidas en los objetivos generales. La descripción realizada infiere sobre los aspectos esenciales sobre experiencias de aplicación del sistema IoT en otros ámbitos de trabajo, así como la explicación de los componentes necesarios para el desarrollo del proyecto como el caso de la plataforma de comunicación, el sensor, la página web, servidor y base de datos.

2.1.Estado del arte

Algunas de las investigaciones realizadas por múltiples autores comprenden:

Terrones (2018) presenta un sistema web para monitorear variables meteorológicas; lo anterior se logró a partir de la implementación de una Raspberry y un Arduino integradas mediante ZigBee y una página web desarrollada con PHP y MySQL para almacenar y visualizar información. Desventajas como uso de un solo nodo para efectuar el sensado y el uso complejo de tecnologías de ZigBee limitan su integración y operatividad con otros sistemas (Tovar, 2018)

Rojas et al (2018) diseñaron e implementaron un sistema automatizado para invernadero hidropónico para la producción de hortalizas a gran escala; para alcanzar este objetivo desarrollaron un sistema de control mediante Arduino leonardo al cual incorporaron sensores térmicos, de humedad DHT22, luminosos entre otros; el esquema de supervisión de variables parte del ingreso de parámetros eran controlados por sistemas como actuadoras,

bombas y aspersores; el resultado final permitió mejorar el desempeño en cuanto a reducción del tiempo de germinación, de consumo de agua y de crecimiento de hortalizas (Rojas, 2018).

Nyoman et al (2018) realizaron un prototipo de sistema para gestión analítica de cultivos hidropónicos mediante un microcontrolador Arduino Uno, Raspberry Pi 2 Model B así como web, dada su alta capacidad como interfaz responsiva. El Arduino Microcontroller Uno combinado con varios sensores relacionados con sensores, como ultrasónicos SR04-HC, sensor de pH, sensor de temperatura DS18B20, el sensor y el EC, así como las herramientas de apoyo para crear cultivos hidropónicos NFT como, grow light, bombas de agua, y aireador. Raspberry Microcomputers Pi 2 Model B en este sistema sirve como un lugar para almacenar la interfaz web/servidor; lo anterior permitió mejorar el desempeño frente a menos gastos y mayores ganancias de este cultivo (Nyoman, 2018).

Lakhiar et al (2018) diseñaron un sistema de sensores para monitorear la humedad y el suministro de nutrientes en cultivos aeropónicos e hidropónicos; el protocolo inalámbrico diseñado se basó en la placa de desarrollo Arduino. Su estudio concluyó que el sistema propuesto puede controlar la frecuencia de autorización de nutrientes en función del contenido de humedad de la cámara de la raíz. Sin embargo, el sistema puede transferir automáticamente toda la información recopilada a un servidor web y también compartirla en Twitter (Lakhiar, 2019).

2.2.Marco teórico

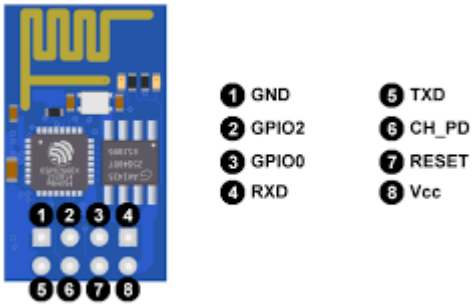
2.2.1. NodeMCU

NodeMCU es una plataforma IoT de código abierto. Incluye el firmware que se desarolla en el SoC Wi-Fi ESP8266 de Espressif Systems; la infraestructura corresponde al módulo ESP-12. La palabra "NodeMCU" hace referencia al firmware, su uso se da mediante lenguaje Lua, cuyo software controla circuitos eléctricos. El módulo ESP-01, trae consigo un sistema que facilita la comunicación con el ESP8266 a partir de comandos AT, a través de puertos seriales. Existen otros firmwares que permiten controlar el microcontrolador ESP8266, como, por ejemplo: microPython, Espruino, ESPBasic y Arduino (para usar esta última opción solo es necesario crear el Sketch y cargarlo (Robles, 2019).

En cuanto a su esquema de conexión para entradas y salidas se tiene:

Dentro de los aspectos negativos del ESP-01 resaltan los pines, dado que posee ocho de estos con tareas únicas.

Figura 4-1. E/S ESP-01



Fuente: programarfacil.com

Tabla 2-1 Pin E/S ESP-01

PIN	TIPO	USO
GND		toma a tierra.
2. GPIO2	pin digital 2	Entrada / salida.
3. GPIO0	pin digital 0	Entrada / salida.
4. RXD	pin digital datos puerto serie	Entrada - GPI03
5. TXD	pin digital datos puerto serie	Entrada - GPI01
6. CH_PD		pin on/of (0 V OFF y 3.3 V ON)
7. RESET		V RESET.

Fuente: (Robles, 2019).

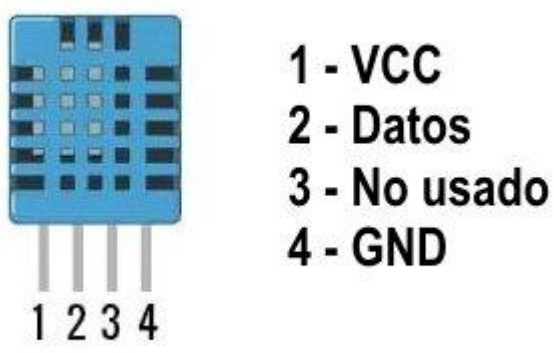
8. Vcc: pin que alimenta el microcontrolador, tienen cables hasta el ESP-12 lo que hace simple el acesso.

2.2.2. Termohigrómetro digital dth 11

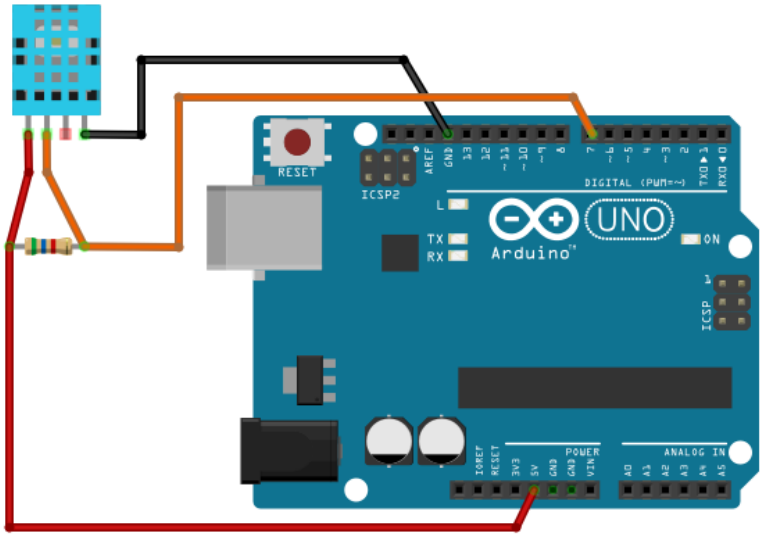
Dentro de sus características su señal digital con alto nivel de calibración, asegurando una altísima calidad y seguridad en el tiempo, cuenta con convertidor de 16 bits integrado, y sensores resistivos (NTC y humedad). Alta capacidad de respuesta rápida en las medidas de hasta un rango del 20 % a 95% de humedad y 0 °C a 50 °C de temperatura. Este sensor está estrictamente calibrado, sus coeficientes de calibrado se almacenan como programas en la memoria OTP, usados en detección de señales internas (Becerra, 2021).

Cuenta con protocolos que comunican mediante único hilo (protocolo 1-wire), esto conlleva a que se integren, su radio de transmisión es de hasta 20 mt de distancia, los pines del elemento se muestran a continuación:

Figura 4-2. E/S sensor dth11



Fuente: (https://robots-argentina.com.ar/, 2023)



Este módulo con DHT11 dispone de 3 pines: la toma de tierra GND, DATA para los datos, y la alimentación VCC (de 3,5V a 5V).

2.2.3. Página web HTML

HTML es el acrónimo de HyperText Markup Languaje. Es un lenguaje de programación que utiliza etiquetas para denir la posición forma y funcionamiento del

contenido de las páginas WEB, como texto, imágenes, videos, colores, hipervínculos y estructura; permitiendo a los navegadores WEB interpretar las posiciones de cada uno de los componentes para construir la página final que verá el usuario (Barbecho, 2022).

2.2.4. Servidor PHP

Se usa para desarrollos web, de fácil programación dado que opera también bajo código HTML; se opera desde una web a partir de PHP y tiene una alta capacidad de integración con cualquier sistema, de uso libre y el más conocido en equipos de cómputo.

Pese a que su enfoque es netamente dirigido a crear paginas web, posee un sistema de interfaz grafico para los usuarios

En este servidor los clientes reciben una web mediante un traductor PHP, quien ejecuta el script para obtener datas

Servidor de base de datos phpMyAdmin.

PhpMyAdmin es una herramienta escrita en PHP cuyo objetivo es el de administrar el MySQL mediante web, a partir de internet. Es multifuncional dado que crea y elimina bases de datos, tablas, campos, privilegios, exportar datos en varios formatos

2.2.5. Internet de las cosas

El Internet de las cosas (IoT) es un sistema que envía y recibe datos a partir de la integración de sistemas digitales y elementos físicos del diario vivir

Los dispositivos del IoT que se encuentran dentro de esos objetos pueden categorizarse como interruptores (es decir, envían las instrucciones a un objeto) o son

sensores (recopilan los datos y los envían a otro lugar) (Salinas, 2022). El internet de las cosas o IoT refiere a tecnologías que facilitan la interconexión de dispositivos electrónicos.

La tecnología IoT fija a los objetos una dirección IP y a través de una red estos se conectan, incluye dispositivos móviles hasta la infraestructura en una ciudad dándole una utilidad muy provechosa.

2.2.6. Agricultura de precisión

Conjunto de sistemas de apoyo a las actividades agrícolas y cuyo objetivo es generar un mejor desempeño de variables de caracter espacial y temporal.

Aquí también aparece el concepto de conservación de precisión (PC), que aborda específicamente el concepto de reducción del daño ambiental, como la disminución de la erosión del suelo o la pérdida de nutrientes (Universidad de Antioquia, 2023).

3. Capítulo 3

3.1.Tipo de investigación

Comprende una investigación aplicada, que de acuerdo con Sampieri (Hernández-Sampieri et al., 2014) la cual cumple con el propósito fundamental de resolver problemas, incluyendo como justificación adelantos y productos tecnológicos; a su vez, se marca en un diseño investigación-acción el cual tiene como precepto conducir a cambiar y por tanto este cambio debe incorporarse en el propio proceso de investigación, es decir, se indaga al mismo tiempo que se interviene. De acuerdo con el tipo de investigación, se empleará un instrumento para el levantamiento de datos, así como su tratamiento

3.2.Diseño metodológico

El diseño metodológico de este proyecto está establecido en 5 etapas las cuales se describen en seguida:

Etapa 1: Investigación previa

En esta etapa se identificaron:

- Variables de almacenamiento.
- Parametros de medición.
- Tipos de registros.
- Tipo de gráficos de función a emplear.
- Identificar otras necesidades de información con el productor.
- Identificación de sensores y actuadores mediante proceso de selección por criterios
- Identificación de tipo de microcontrolador mediante selección de criterios.

- Identificación de sistema de comunicación inalámbrica y de transmisión de datos mediante selección por criterios.
- Diseño de lazos de control. Se realiza el diseño lógico del sistema y se definieron los sensores y los respectivos actuadores.
- Diseño plano instrumentación. Se realiza el diseño del plano de instrumentación del sistema, así mismo el plano de periféricos de interconexión al módulo IoT.
- Realización del diagrama de flujo del sistema. Se define la secuencia o rutina de trabajo del sistema para posteriormente realizar el programa para su respectiva interpretación.
- Diseño electrónico. Se realiza el respectivo diseño electrónico teniendo en cuenta los diferentes componentes de entrada y salida del sistema, así como los componentes pasivos para el acondicionamiento de señales.

Etapa 2: Sistema de comunicación

Las actividades relacionadas comprenden:

- Evaluación y selección del protocolo a partir de variables como costo, confiabilidad y capacidad de transmisión.
- Caracterización de la estructura de almacenamiento.
- Identificar y seleccionar los elementos de software y hardware a partir de costo, facilidad de integración entre otros.
- -Diseño y elaboración del algoritmo.
- Priorización de los requerimientos.
- Evaluación económica para selección del algoritmo.
- -Establecer el emisor, mensaje, medio y receptor, así como la identificación

de la interfaz.

- Seleccionar el tipo de transmisión de datos.
- Establecer las capas de acceso.
- Determinar el tamaño de la red.

Etapa 3: Preparación de la plataforma

Comprende descripciones precisas de los componentes tanto en hardware como en software (arquitectura y operación).

Etapa 4: Visualización de las variables

El back-end almacena datos y facilita la interacción de usuario en intervalos de tiempo real.

Etapa 5: Evaluación de funcionamiento de la plataforma en IoT.

Se realizan pruebas a partir de la ejecución de los dispositivos integrados para monitoreo de variables.

3.3.Componentes del sistema

El sistema integral se compone de cuatro elementos interconectados: el NodeMCU basado en el ESP8266, la página web HTML, el servidor PHP y el servidor de base de datos phpMyAdmin. Su propósito fundamental radica en la monitorización en tiempo real de la temperatura y humedad de un entorno específico, presentando estos datos de manera visual en una interfaz web, así como enviándolos y almacenándolos posteriormente en el servidor de base de datos phpMyAdmin. Cada componente desempeña un papel crucial en la operación cohesionada del sistema.

Bloque 1: NodeMCU - basado en el chip ESP8266, corresponde al "cerebro" del sistema. Esta placa compacta y potente establece conexión WiFi con la red local a través de la biblIoTeca ESP8266WiFi, permitiendo la comunicación bidireccional con otros componentes. En un ciclo de bucle, el NodeMCU utiliza un sensor DHT11 para adquirir mediciones de temperatura y humedad, lo que garantiza una eficiente funcionalidad a un bajo costo.

Especificaciones de componentes:

- Microcontrolador: ESP8266
- Interfaz USB integrada
- Conectividad WiFi
- Pines digitales y analógicos
- Memoria flash incorporada

Figura 4-3. microcontrolador ESP

Fuente: Data sheet

En cuanto a su funcionamiento se tienen los siguientes usos:

- Establece una conexión WiFi con la red local mediante la biblIoTeca ESP8266WiFi.
- Lee los valores de temperatura y humedad del sensor DHT11 en un ciclo de bucle.
- Envía solicitudes GET al servidor PHP con los valores medidos.
- Muestra los valores actuales de temperatura y humedad en el Monitor Serie.
- Envía datos al servidor phpMyAdmin y a la página web, incluyendo temperatura, humedad y marca de tiempo.

Ventajas y beneficios:

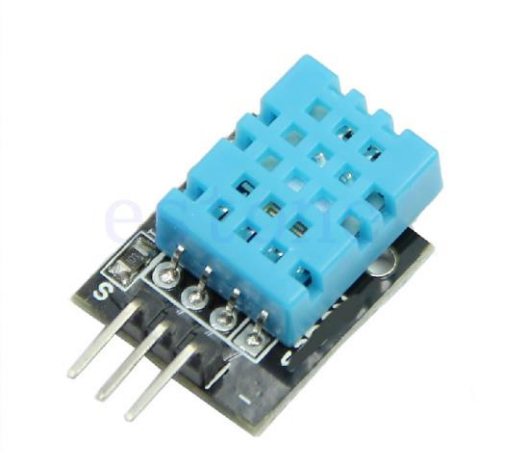
- **Conectividad WiFi:** Facilita la comunicación con la red local e Internet, permitiendo la transmisión de datos en tiempo real.
- **Potente CPU:** Suficiente capacidad de procesamiento para recopilar y manejar datos.
- **E/S Digital y Analógica:** Posibilita la interacción con sensores y actuadores.
- **Memoria Flash:** Almacena programas y datos de manera eficiente.
- **Comunidad Activa:** Amplia documentación y soporte gracias a la comunidad de desarrolladores.
- **Bajo Costo:** Económico en comparación con otras placas de desarrollo.
- **Versatilidad:** Puede conectarse a redes WiFi, acceder a Internet y comunicarse con servidores en la nube.
- **Integración:** NodeMCU incluye componentes esenciales como regulador de voltaje y USB.

- **Facilidad de Programación:** Utiliza el código Arduino para facilitar la programación y comunicación con otros componentes.

3.4.Integración dispositivos físicos

Dadas las necesidades de transmitir información por internet se seleccionó un módulo de comunicación inalámbrica compatible con la programación Arduino, dicho modulo corresponde a una “Tarjeta NodeMcu V3 para ESP8266”, la cual es la última versión de esta familia; adicional a esto se monitoreo la temperatura mediante dht11 un termohigrómetro digital.

Figura 4-4. Termohigrómetro digital dth 11

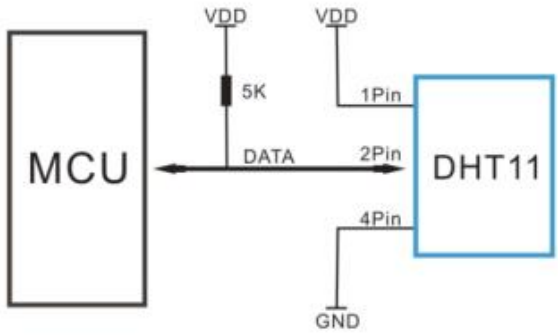


Fuente: Datasheet

A partir de este momento se desarrolla realiza el montaje en físico en una protoboard y se procedió a la programación; aquí el dispositivo dht11 y la tarjeta nodecmu se energizan por el mismo cable de programación del módulo nodemcu, aclarando que el mismo modulo tiene pines de conexión que suministra unos voltajes de 3.3 v, al cual se conecta el sensor

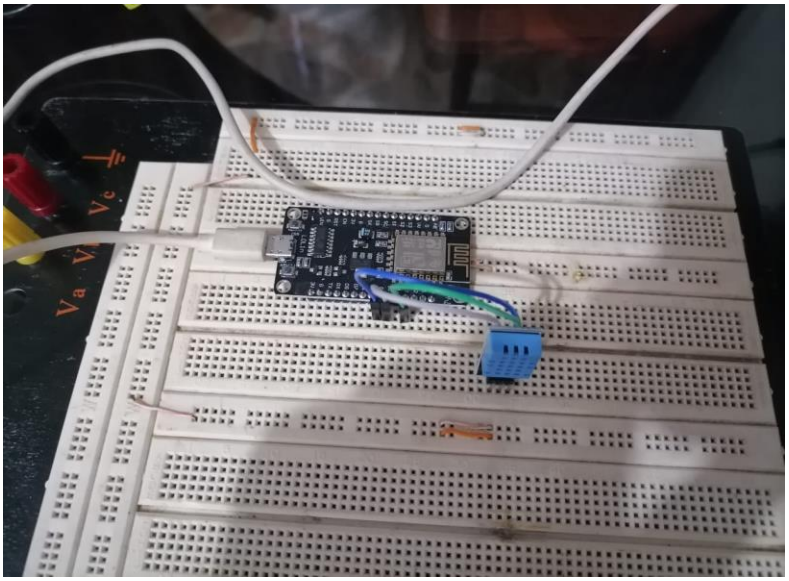
de temperatura para su funcionamiento y donde se establece que el pin 3 del módulo nodecmu es por donde se recibirán los datos digitales de temperatura y humedad del sensor.

Figura 4-5, Esquema de conexión entre el sistema de procesamiento de datos y el sensor DHT11



Fuente: Autor

Figura 4- 6. Montaje de dispositivos

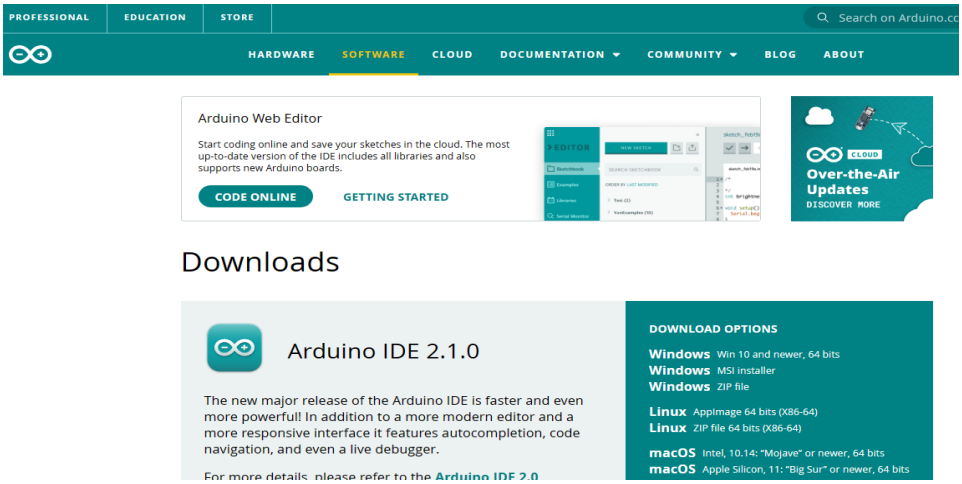


Fuente: Autor

A partir de la instalación física de dispositivos se realizó el proceso de descarga de la plataforma o software donde se programarán los dispositivos en este caso realizaremos

esto en el ID de Arduino de acuerdo a las características del equipo de cómputo y sistema operativo, cuya versión corresponde a arduino-ide_2.1.0_Windows_64bit , la página oficial donde se descargo es la https://www.arduino.cc/en/software.

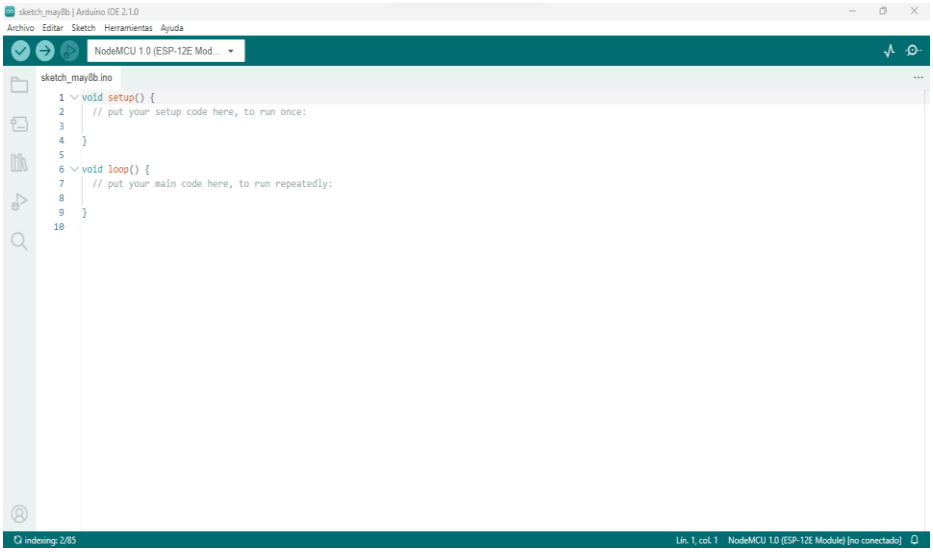
Figura 4-7. Proceso descarga plataforma



Fuente: https://www.arduino.cc/en/software.

Una vez instalado se realizó el proceso de inicialización del programa, como se indica en la figura 4-4.

Figura 4-8. Inicialización del programa



Fuente: https://www.arduino.cc/en/software.

A partir de este proceso se realizó la configuración del ID el cual permite gestionar la programación de nodemcu.

4. Capítulo 4

4.1.Resultados

Los resultados presentados a continuación se dan en función de la secuencia de desarrollo frente al software y hardware a partir de los siguientes bloques para iniciar el proceso de visualización de las variables de temperatura y humedad:

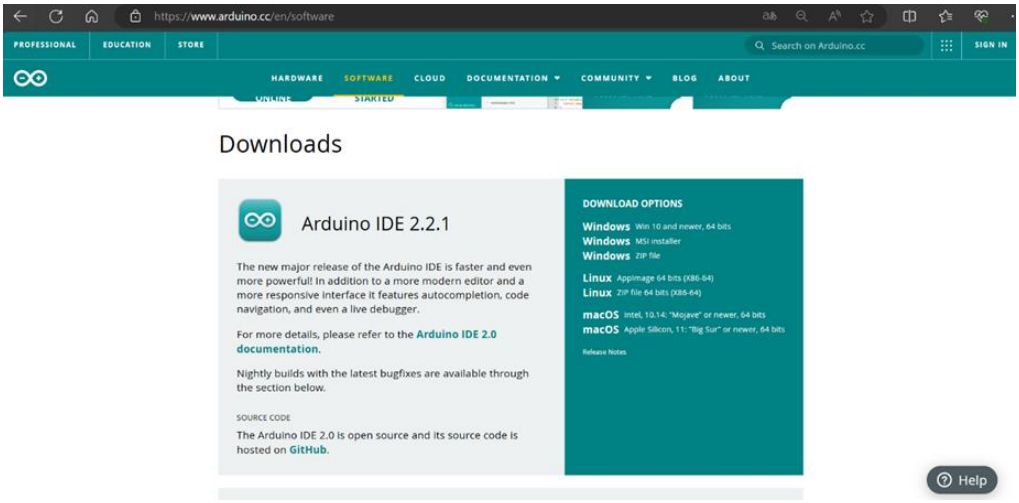
4.1.1. Bloque 1

Instalación Arduino:

Para instalar el programa de Arduino (Arduino IDE) en la computadora deben contemplarse los siguientes pasos:

Paso 1: Descarga Arduino IDE desde el sitio web oficial de Arduino en https://www.arduino.cc/en/software.

Figura 4-9. Descarga Arduino



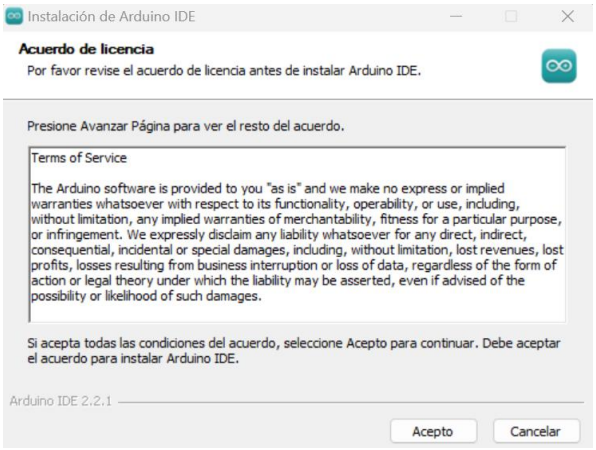
Fuente: https://www.arduino.cc/en/software.

Se descarga la versión de Arduino IDE adecuada para el sistema operativo (Windows, macOS o Linux), el cual para nuestro caso es Windows.

Paso 2: Instala Arduino IDE

Para Windows:

Se ejecuta el archivo de instalación que descargamos. Se siguen las instrucciones del instalador, Se dejan las configuraciones predeterminadas. Una vez completada la instalación, Arduino IDE estará listo para usar.

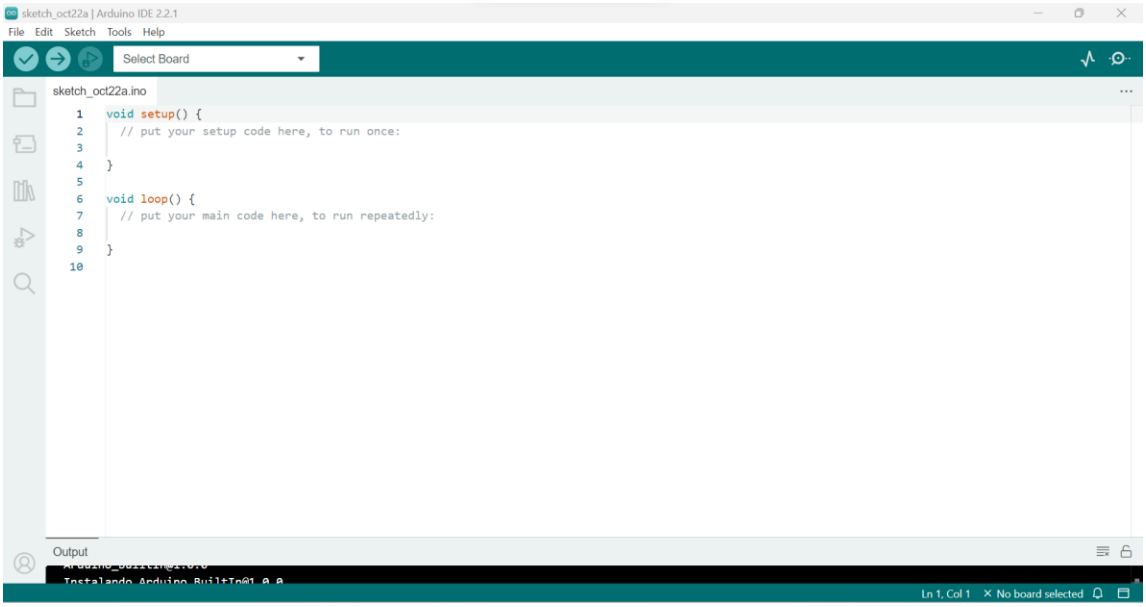


A partir de lo anterior se procede a ejecutar el Arduino:

Paso 3: Ejecuta Arduino IDE

Una vez que Arduino IDE esté instalado, se ejecuta desde el menú de aplicaciones (Windows).

Figura 4-10. Ejecución Arduino desde menú de aplicaciones.



A partir de lo anterior se realiza la conexión con el hardware

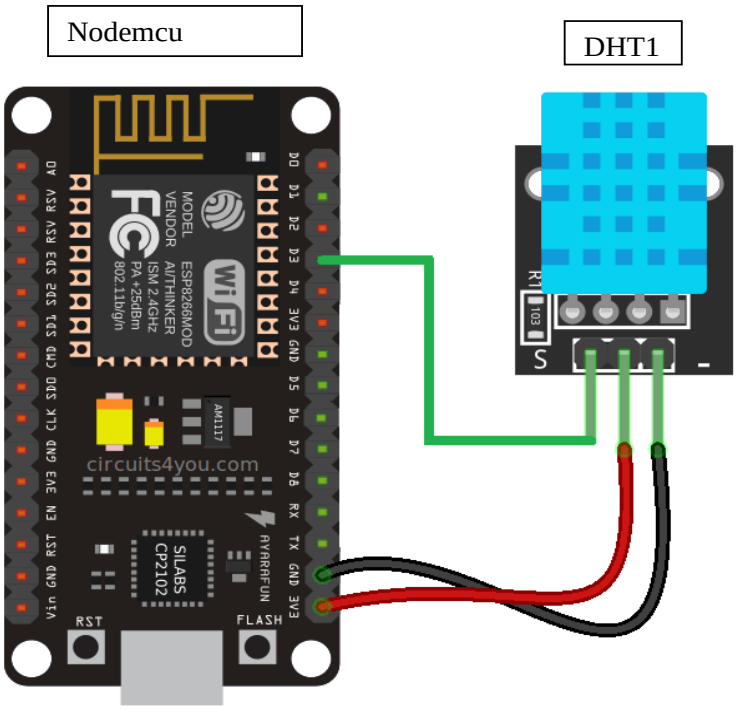
Paso 4: Conectar el Hardware

Conectamos el NodeMCU y el sensor DHT11 de la siguiente manera:

Conecta el pin VCC del DHT11 al pin 3.3V del NodeMCU.

Conecta el pin GND del DHT11 al pin GND del NodeMCU.

Conecta el pin DATA del DHT11 al pin D2/D3 del NodeMCU.

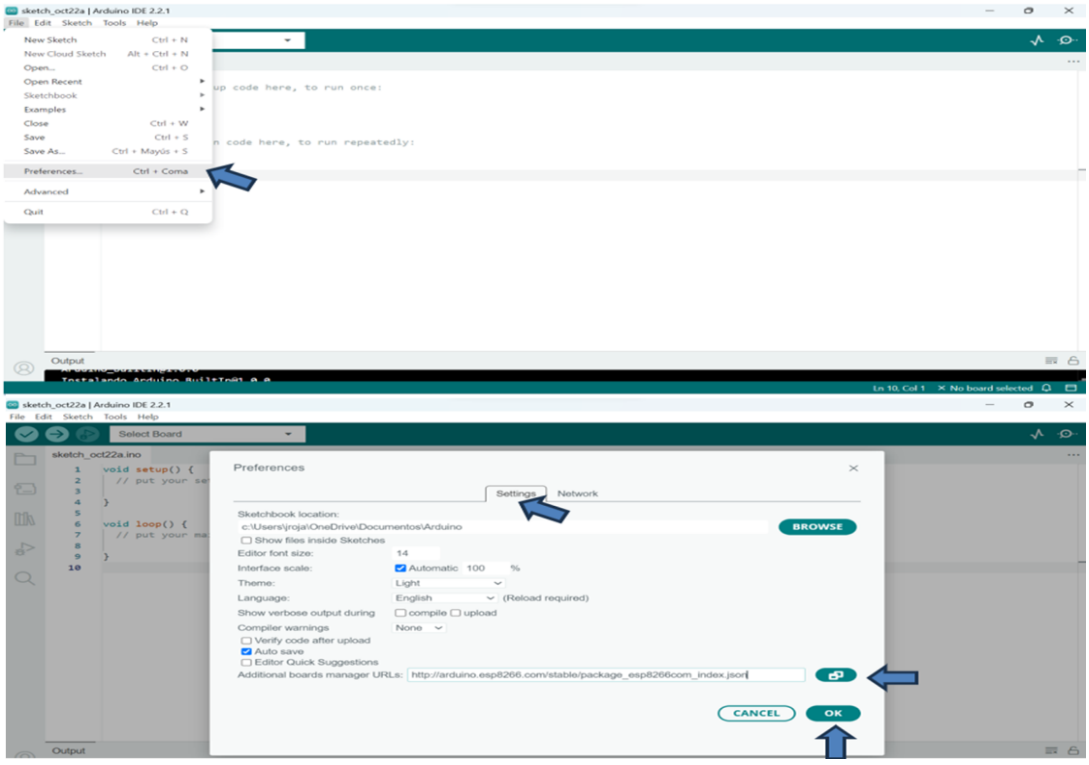


Paso 5: Instalar el Soporte de ESP8266 en Arduino IDE

Se Abre Arduino IDE "Archivo o Files" -> "Settings", en el campo "URLs adicionales de Gestor de Tarjetas o Additional boards managers URLs " en la parte inferior de la ventana de Settings, se agrega la URL: http://arduino.esp8266.com/stable/package_esp8266com_index.json dando clic en "OK" para cerrar la ventana de preferencias.

seguidamente "Herramientas" -> "Placa" -> "Gestor de tarjetas". Se Busca "esp8266" y haz clic en "Instalar" en la entrada correspondiente. Una vez que la instalación se complete, se cierra el Gestor de tarjetas.

Figura 4-11. Instalación carpetas

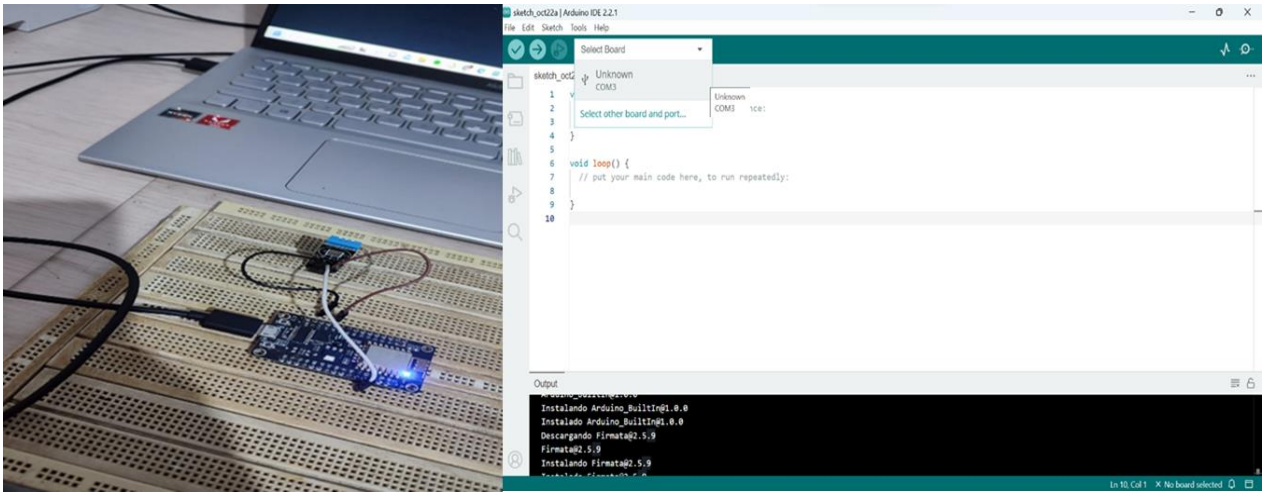


Fuente: Autor

Paso 6: Seleccionar la Tarjeta y el Puerto

Conectamos el montaje del circuito básico realizado en el paso 4 al computador y luego se va a a la parte superior de ide donde dice “select boards" y como ya tenemos conectado nuestro NodeMCU el automáticamente nos detecta el puerto que se a asignado por defecto (en nuestro caso puerto COM3) damos clic en este ítem.

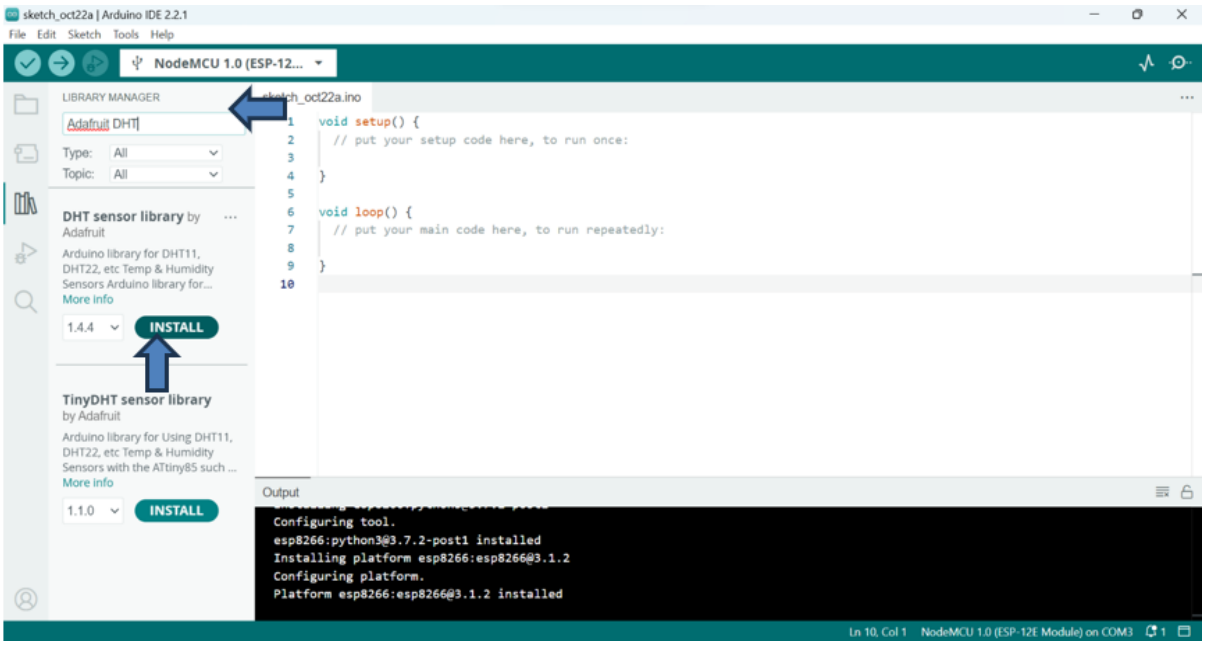
Figura 4-12. Selección de tarjeta y puerto



Fuente: Autor

Paso 7: Instalar la Biblioteca Adafruit DHT

Para trabajar con el sensor DHT11, se necesita contar con la biblioteca Adafruit DHT, el cual permite leer los datos digitalizados desde el Sensor DHT11 y luego poderlos interpretar y visualizar en un formato de fácil comprensión. Para esto se va al apartado en la parte superior del IDE "Sketch" -> "Incluir Biblioteca" -> "Gestionar Bibliotecas...", En el cuadro de búsqueda, escribe "Adafruit DHT". Se hace clic en el botón "Instalar" junto a la biblioteca "Adafruit DHT" de Adafruit.



Una se han determinado las librerías y el módulo a trabajar se procede a realizar el código (leer los datos del sensor DHT11 y enviarlos a la serie (Serial Monitor) en Arduino IDE) que luego será grabado en la memoria del dispositivo nodecmu para la lectura y funcionamiento de las variables de temperatura y humedad, este código se puede ver en el anexo 1.

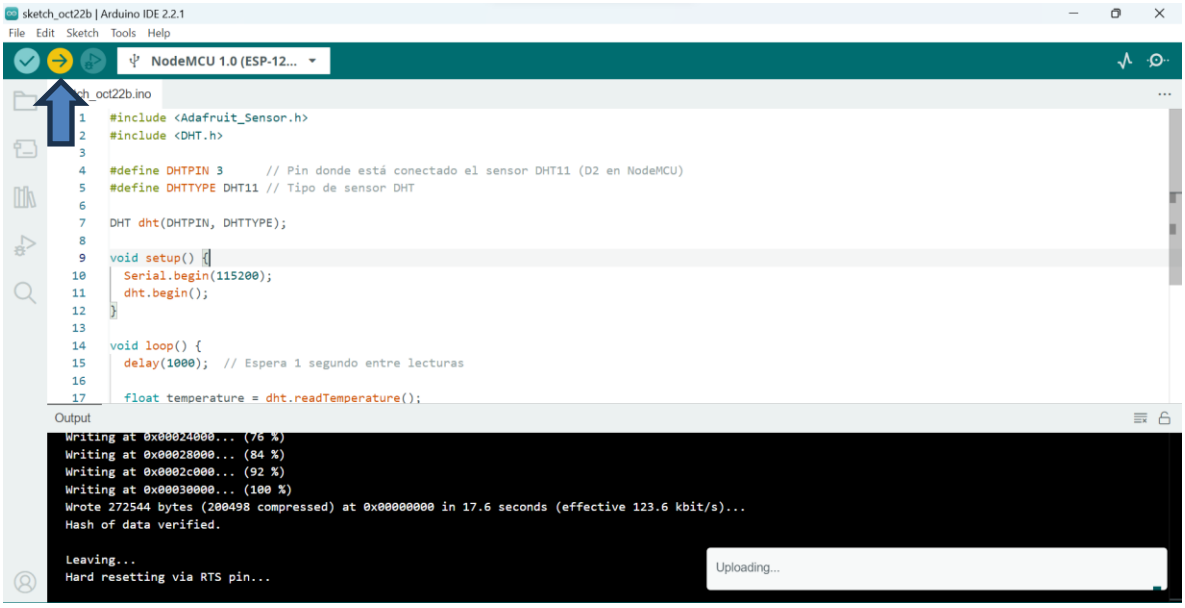
Paso 9: Cargar y Verificar el Código

Guarda el código en Arduino IDE.

Haz clic en el botón "Verificar" (ícono de check) para asegurarte de que no haya errores en el código.

Paso 10: Cargar el Código en el NodeMCU

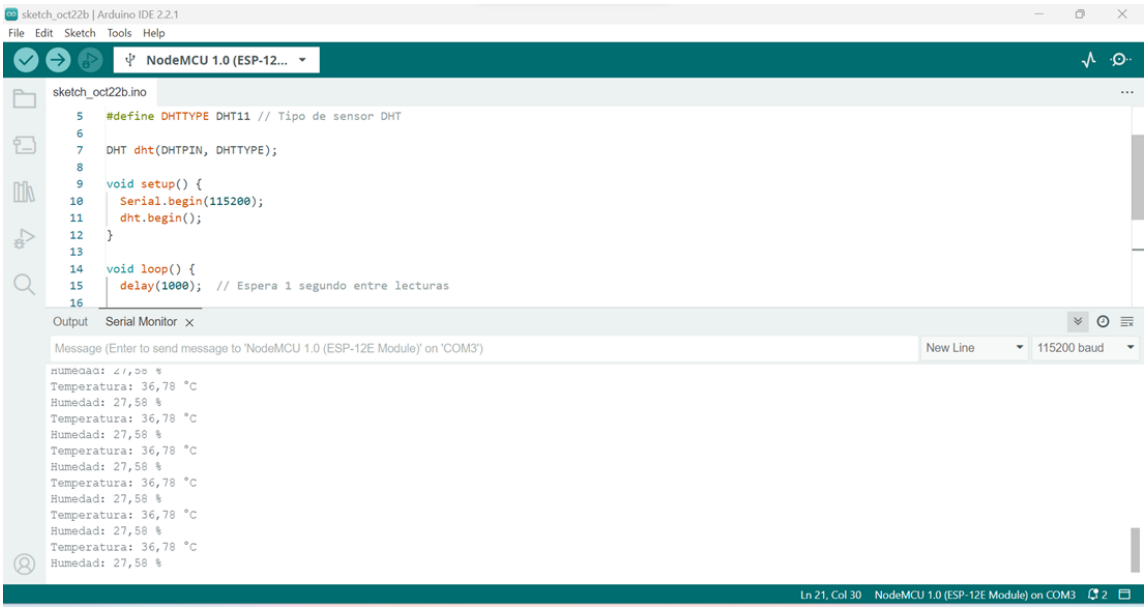
Con el NodeMCU conectado al puerto USB de la computadora, se hace clic en el botón "Subir" (flecha hacia arriba) en Arduino IDE.



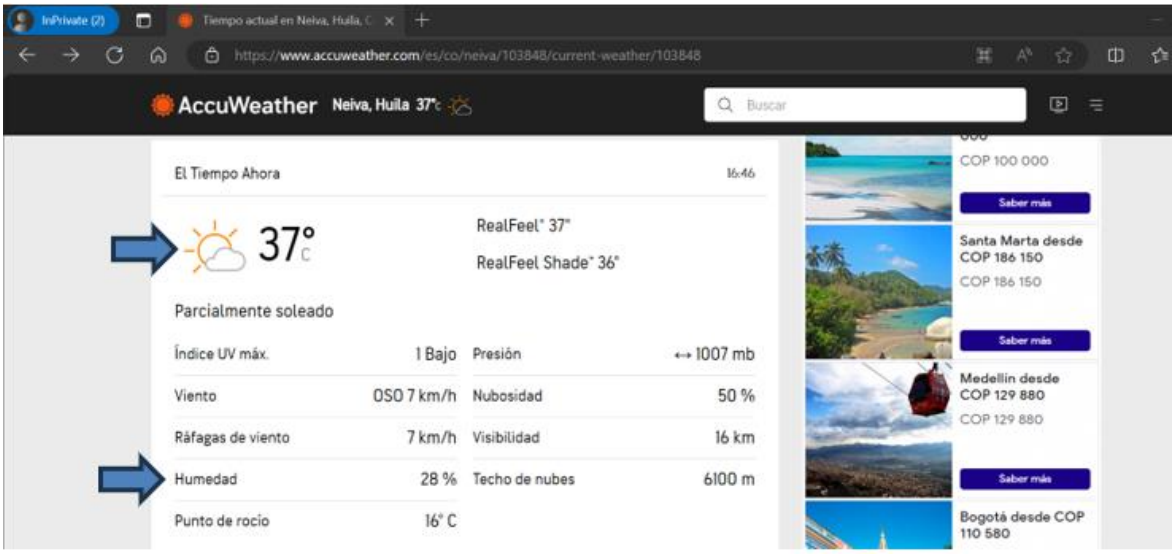
Cargado el código se pueden observar los resultados, a partir de la apertura del "Serial Monitor" en Arduino IDE haciendo clic en "Herramientas" -> "Monitor Serie".

La temperatura y la humedad del sensor DHT11 siendo mostradas en el monitor serie.

Figura 4-13. Lectura de temperatura y humedad del sensor DHT 11



Al comparar estos datos con los de la página web: https://www.accuweather.com/es/co/neiva/103848/current-weather/103848), la cual proporciona las variables de interés como temperatura y humedad se observa que el dispositivo lee los datos de manera correcta.



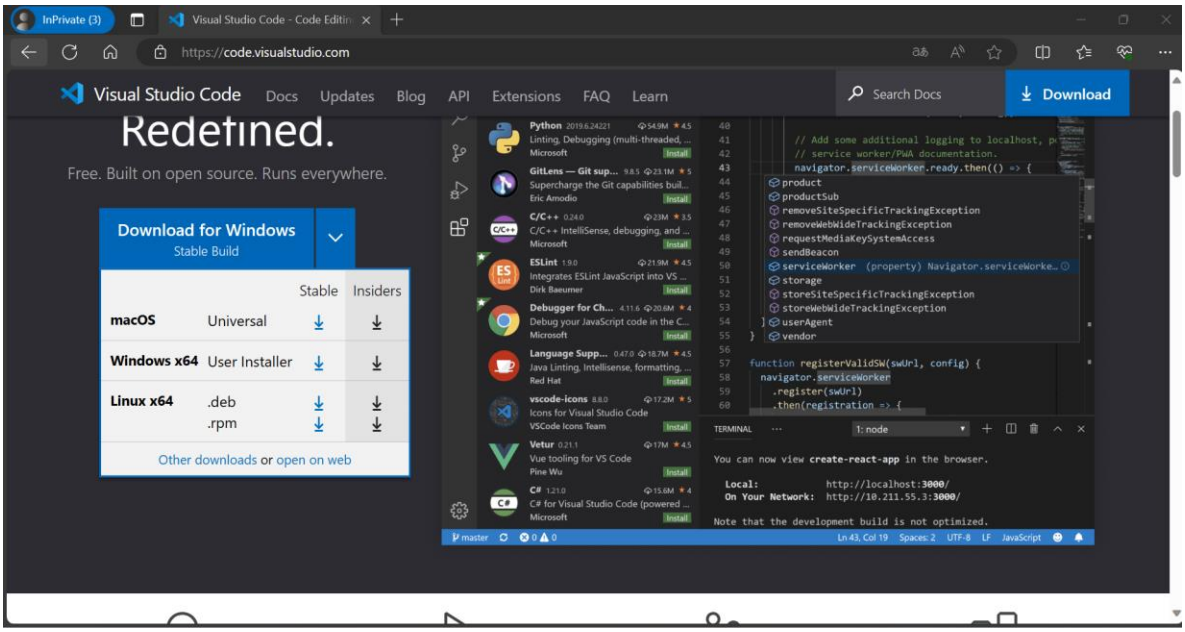
Con esto se termina la primera etapa del proyecto que es la configuración del ide de Arduino de acuerdo con el módulo del sensor la placa o tarjeta a utilizar del nodemcu, el bloque a desarrollar es el montaje de un servidor local y posteriormente crear una página HTML para que se cargue en dicho servidor local y poder enviar los datos a dicho servidor y mostrarlos a través de la página para enviar dichos datos a una.

4.1.2. Bloque 2

creación del servidor WEB y pagina WEB HTML. A continuación, se emplea el nodecmu como servidor web para la visualización de los datos del proyecto. En este orden de ideas necesitáremos instalar visual studio code , dado que es una herramienta de edición de cocido abierto para la creación de la página y demás archivos necesarios para la implementación de este proyecto:

Según lo anterior se procede a descarga e instalación de la herramienta de edición de código “visual estudio code”

Se va a la siguiente dirección url: https://code.visualstudio.com/ y le damos click en la opción “Download for Windows”→ WindowsX64 , descargamos la aplicación, la ejecutamos y dejamos la configuración por defecto.



Una vez instalada la herramienta se ejecuta visual studio code, a partir de la carpeta en donde se tiene el archivo que se ha creado anteriormente de Arduino que contiene el código para la lectura del dht11

Seleccionado el directorio de trabajo, se crea el nuevo archivo llamado “pageindex1” con extensión “.h” y de esta manera se crea el archivo que recibirá los datos provenientes del sensor para poder visualizarlos en la página.

Para poder leer los datos provenientes del señor DHT11 Agregamos al archivo creado el siguiente contenido:

```
const char SENSOR_page[] PROGMEM = R"=====(
<!DOCTYPE html>
<html>
<head>
  <meta name="viewport" content="width=device-width, initial-scale=1">
  <meta charset="UTF-8"> <!-- línea para especificar la codificación caracteres especiales -->
</head>
<body>
```

```
  <h1>Medición de Temperatura y Humedad</h1>
  <p>Temperatura: <span id="temperature">%TEMPERATURA% &#8451;</span></p>
  <p>Humedad: <span id="humidity">%HUMEDAD% %</span></p>
</body>
</html>
)";
```

El código proporcionado parece es un fragmento de código C++ que define una cadena de texto llamada SENSOR_page. Esta cadena de texto contiene un fragmento de código HTML que se utiliza en el proyecto relacionado con la medición de temperatura y humedad. Veamos el código en detalle:

1. **const char* SENSOR_page**: Esta línea declara una variable llamada **SENSOR_page** que es un puntero a una cadena de caracteres (en C++ se usa **const char*** para representar cadenas de texto).

2. **R"(** y **)"**: Estos caracteres especiales **R"(** y **)"** son parte de una característica de C++ llamada "raw string literal". Esto permite definir una cadena de texto que abarca varias líneas y contiene caracteres de escape sin necesidad de escaparlos manualmente.

3. El contenido del **SENSOR_page** es un fragmento de código HTML, y aquí está el significado de cada parte:

- **<!DOCTYPE html>**: Esto define la declaración DOCTYPE de HTML, lo que indica que se está utilizando HTML como lenguaje de marcado.

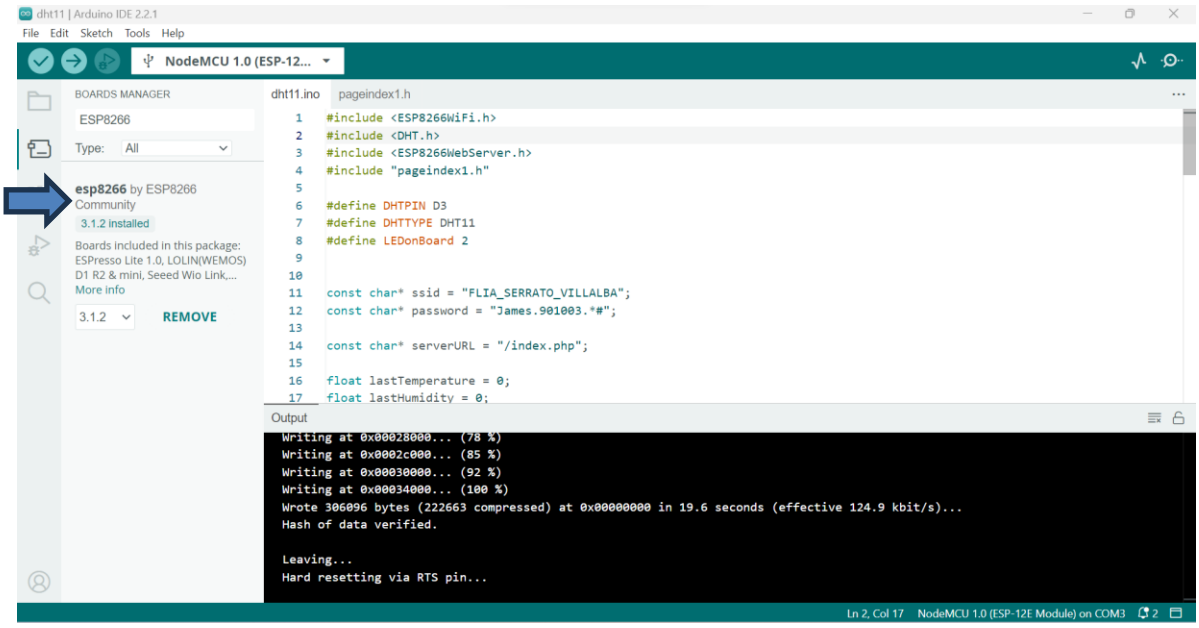
- **<html>**, **<head>**, y **<body>**: Estas son etiquetas HTML que definen la estructura básica de una página web. **<head>** generalmente contiene metainformación sobre la página, mientras que **<body>** contiene el contenido visible de la página.

- **<meta name="viewport" content="width=device-width, initial-scale=1">**: Esto es una etiqueta meta que se usa comúnmente en páginas web para

controlar cómo se ajusta la página en dispositivos móviles. Establece que el ancho de la página se adapta al ancho del dispositivo y el nivel de escala inicial es 1.

- **<h1>Medición de Temperatura y Humedad</h1>**: Esto es un encabezado de nivel 1 que muestra el título de la página, que es "Medición de Temperatura y Humedad".
- ** Temperatura: %TEMPERATURA% °C **: Esto crea un párrafo que muestra la etiqueta "Temperatura", seguida de un elemento de span con un id de "temperature". La parte **%TEMPERATURA%** parece ser un marcador de posición que se podría reemplazar dinámicamente con un valor de temperatura en el código fuente final. El "°C" indica que se muestra la temperatura en grados Celsius.
- ** Humedad: %HUMEDAD% % **: Similar al párrafo anterior, esto muestra la etiqueta "Humedad" con un elemento de span que tiene un id de "humidity". Al igual que antes, **%HUMEDAD%** parece ser un marcador de posición que se podría reemplazar con un valor de humedad en el código fuente final. El "%" al final indica que se muestra el valor de humedad en porcentaje.

Paso 2: Configurar el proyecto Arduino con librerías ESP8266 y código para el envío de datos se instala las librerías "ESP8266WiFi" para la conectividad Wi-Fi y la "ESP8266WebServer" para crear el servidor web en el ESP8266. Esto lo realizamos de igual manera como se realizó en el **bloque 1 paso 7** , pero este caso para las librerías mencionadas.



Una vez se tienen instaladas las librerías se procede a configurar el código para que se conecte a una red wifi y envíe posteriormente a la página web creada “pageindex1.h” (ver anexo 2 del código)

Este código incluye las siguientes bibliotecas

```
#include <ESP8266WiFi.h>
#include <DHT.h>
#include <ESP8266WebServer.h>
#include "pageindex1.h"
```

Estas líneas incluyen las bibliotecas necesarias para habilitar la funcionalidad WiFi en el ESP8266, controlar el sensor DHT11 y crear un servidor web. Los pines y variables correspondientes son:

```
#define DHTPIN D3
#define DHTTYPE DHT11
#define LEDonBoard 2

const char* ssid = "FLIA_SERRATO_VILLALBA";
const char* password = "James.901003.*#";
const char* serverURL = "/index.php";
float lastTemperature = 0;
float lastHumidity = 0;
```

En esta sección, se definen los pines utilizados para el sensor DHT11 y el LED incorporado, así como las credenciales de la red WiFi a la que se conectará el ESP8266 y la URL de la página web que servirá.

1. Declaración de objetos:

```
ESP8266WebServer server(80);
DHT dht(DHTPIN, DHTTYPE);
```

Aquí se declaran un objeto del tipo **ESP8266WebServer** y un objeto **DHT** para interactuar con el servidor web y el sensor DHT11, respectivamente.

2. Funciones de manejo de solicitudes: El código define varias funciones de manejo de solicitudes HTTP:

- **handleRoot()**: Esta función responde a la solicitud principal ("/") y envía una página web que muestra la temperatura y la humedad.
- **handleSaveData()**: Lee la temperatura y la humedad del sensor DHT11 y las almacena en las variables **lastTemperature** y **lastHumidity**. Luego, llama a **handleRoot()** para mostrar los datos en la página web.
- **handleTemperature()**: Responde a la solicitud de la ruta "/readTemperature" y envía la temperatura actual como texto plano.
- **handleHumidity()**: Responde a la solicitud de la ruta "/readHumidity" y envía la humedad actual como texto plano.

3. Configuración inicial en **setup()**:

- Se inicia la comunicación serial para la depuración.
- Se inicia el sensor DHT11.
- Se conecta a la red WiFi especificada en **ssid** y **password**.

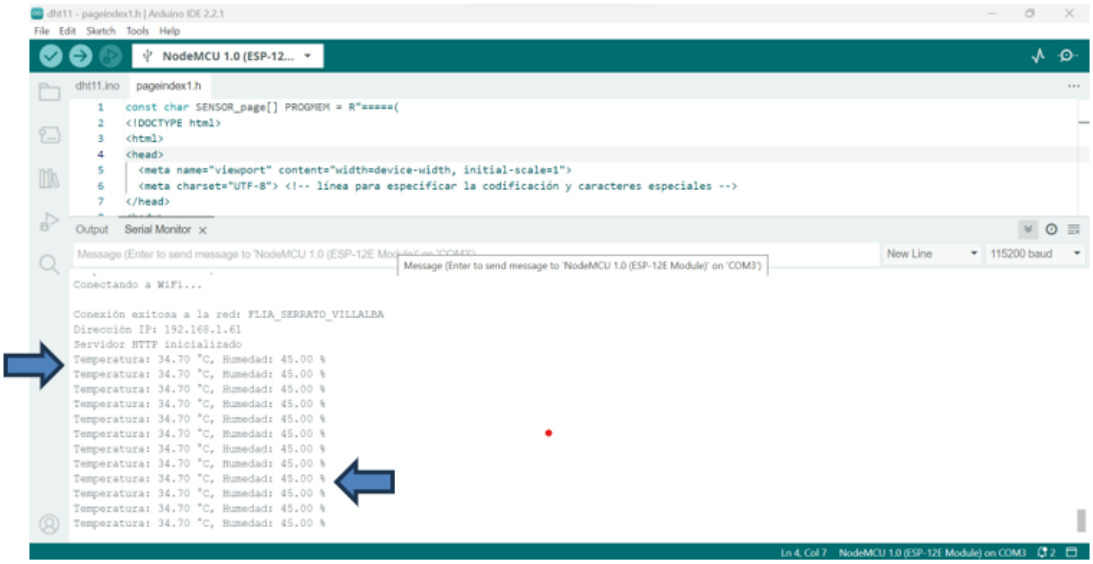
- Se configura el pin del LED incorporado.
- Se configuran las rutas de solicitud y se inicia el servidor web.

4. Bucle principal en **loop()**:

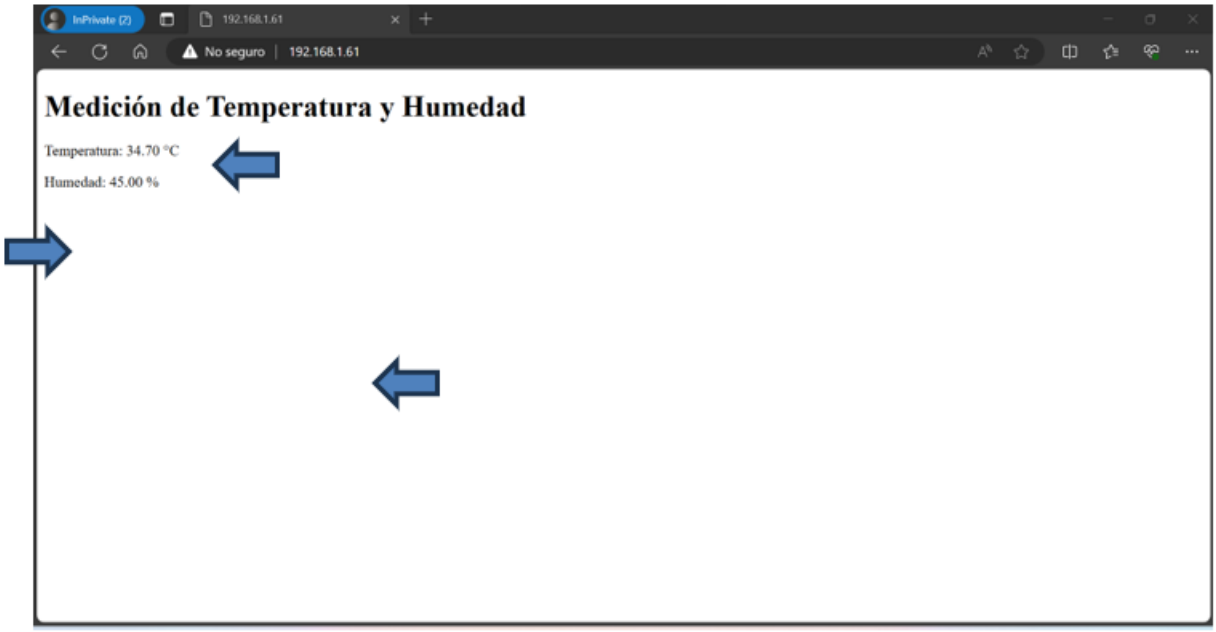
- **server.handleClient()** maneja las solicitudes entrantes del servidor web.
- **handleSaveData()** actualiza la temperatura y la humedad y actualiza la página web.
- Se leen la temperatura y la humedad del sensor DHT11 y se muestran en la consola.
- Hay un retardo de 1 segundos antes de repetir el ciclo.

En resumen, este código configura el ESP8266 para medir la temperatura y la humedad y proporciona una interfaz web para ver los datos en tiempo real. Las páginas web se generan a partir de una plantilla HTML almacenada en un archivo externo ("pageindex1.h"). a continuación, se visualiza la ejecución del código

Figura 4-14. ejecución de código



Aquí se copia la dirección IP que nos genera el dispositivo, en este caso “Dirección IP: 192.168.1.61” y la pegamos al navegador web “ Microsoft EDGE”, obteniendo la siguiente descripción.



Al cargar la página con la IP se comprueba que la lectura del sensor y la web son las mismas indicando que la lectura del sensor DHT de temperatura y humedad son enviados correctamente al servidor web creado como se indica a continuación

Figura 4-15. Visualización grafica de variables temperatura y humedad

De la visualización anterior presentada en la figura 4-15 se puede concluir que los datos transmitidos en tiempo real muestran de forma clara y grafica los valores de control, proporcionado al productor de la finca las mercedes información clave para realizar una buena gestion a nivel de control de desarrollo, costos y toma de decisiones asociadas al ciclo del cultivo. La información muestra visualmente cambios repentinos en ambas variables.

Al código pageindex1.h se le realizaron modificaciones para que los valores se visualizaran en tiempo real de manera grafica teniendo en cuenta la fecha y hora de la lectura el código que se generó es el siguiente.

4.1.3. Bloque 3

Creación de servidor para base de datos y almacenamiento de variables en phpmyadmin a través de XAMPP

Para instalar XAMPP y usar PHPMyAdmin se deben seguir los siguientes pasos:

Paso 1: Descargar XAMPP

Direccionamiento a URL o al sitio web de Apache: https://www.apachefriends.org/index.html Friends, de aquí se realiza la descarga de la versión de XAMPP adecuada para el sistema operativo (en nuestro caso Windows).

Paso 2: Instalar XAMPP

Se procede a ejecutar el instalador descargado y sigue las instrucciones del asistente de instalación.

Paso 3: Iniciar los servicios

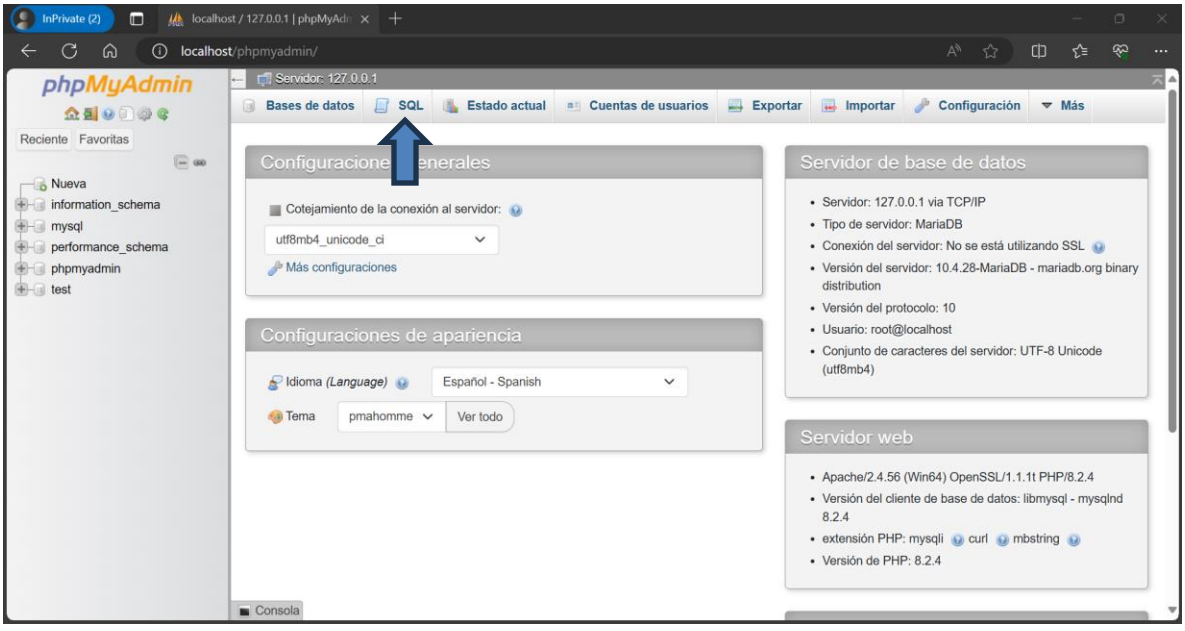
Una vez instalado, se abre XAMPP Control Panel el cual contiene una lista de servicios como Apache, MySQL, FileZilla, etc. Asegúrate de que Apache y MySQL estén marcados en verde. Si no lo están, haz clic en "Start" junto a cada uno para iniciar los servicios.

Paso 4: Creación de base de datos y variables de almacenamiento

Ahora tenemos el servidor local con phpmyadmin para la creación de base de datos y tablas para el almacenamiento de las variables provenientes del nodecmu como lo es la temperatura y la humedad, para esto se deben cumplir los siguientes pasos:

1. Se necesita crear una base de datos que se llamara "datos_senor", adentro de esta base se crea una tabla que se llame "datos" y dentro de esta tabla se creen las siguientes variables "id" es de tipo int (11) auto incremental y no nula " , "fecha_hora es de tipo datetime", "temperatura tipo float" y "humedad tipo float", cabe resaltar que etos parámetros de las variables se crearan de esta manera teniendo el tipo de formato en el que recibiremos las mismas desde el nodecmu.

2. Una vez definida el nombre de la base, el nombre de la tabla y las columnas que se tendrán en esta para el almacenamiento de los datos procedemos a crearla en phpmyadmin, nos dirigimos a la pestaña que dice “SQL” y digitamos lo siguiente:

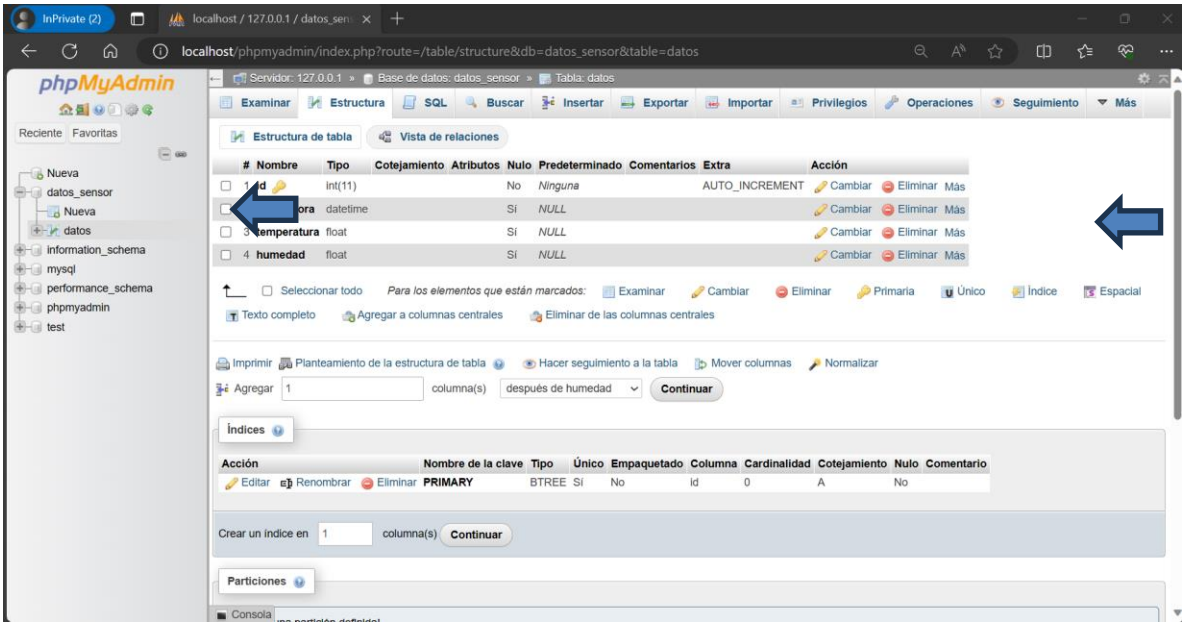


3. Se crea la base de datos para lo cual se digita “CREATE DATABASE datos_sensor;” y damos clic en continuar

4. Creación de la tabla y las columnas para el almacenamiento de datos: se selecciona la base de datos creada en el paso anterior “datos_sensor” y procedemos de igual manera como realizamos las indicaciones del punto 2. Y 3. Pero para este paso digitamos lo siguiente:

```
CREATE TABLE datos (
   id INT(11) AUTO_INCREMENT NOT NULL,
   fecha_hora DATETIME,
   temperatura FLOAT,
   humedad FLOAT,
   PRIMARY KEY (id)
);
```

Con estos pasos ya creamos nuestra base de datos con la tabla y columnas correspondientes para su posterior almacenamiento de datos



4.1.4. Bloque 4

Conectividad entre el nodecmu y phpMyAdmin. Para poder almacenar los datos en el servidor phpmyadmin se necesita crea un puente que solicite los datos provenientes del nodecmu y luego integrarlos en nuestra base de datos a través del servidor local y gestor de base de datos phpmyadmin , para esto se necesita crear un script o código con extensión ".php" (puente) (Anexó) y almacenarlo dentro de nuestra carpeta de directorio raíz de la instalación de xammp para que sirva de identificador y conector entre nuestro nodecmu y nuestra base de datos.

1. Creamos una carpeta que contendrá un archivo llamado "Proyecto_Sensor_DHT11" en el siguiente directorio "C:\xampp\htdocs"

2. Se ingresa a visual studi code , aquí se referencia el directorio anterior y se crea el archivo con extensión ".php" con el nombre de "index.php"

3. Creación de script para obtener las variables

Este código PHP realiza varias tareas relacionadas con la gestión de datos de temperatura y humedad que se envían a través de una solicitud GET y los almacena en una base de datos. Aquí está el detalle de lo que hace:

1. **Configuración de la Zona Horaria**:

- La primera línea establece la zona horaria a "America/Bogota" utilizando **date_default_timezone_set()**. Esto asegura que las fechas y horas se registren y se muestren en la zona horaria correcta.

2. **Verificación de Datos**:

- El código verifica si se han recibido los datos de temperatura (**temperature**) y humedad (**humidity**) a través de una solicitud GET utilizando **isset()**. Esto se hace para asegurarse de que los datos necesarios estén presentes en la URL.

3. **Obtención de Datos**:

- Si los datos de temperatura y humedad están presentes, se obtienen y se almacenan en las variables **$temperatura** y **$humedad**.

4. **Obtención de Fecha y Hora Actual del Servidor**:

- Se obtiene la fecha y hora actual del servidor utilizando la función **date()**. El formato utilizado es 'Y-m-d H:i:s'.

5. **Mensajes de Depuración**:

- Se muestran mensajes de depuración en la página para verificar los datos recibidos. Esto se hace utilizando **echo** para imprimir los valores de temperatura, humedad y la fecha y hora actuales.

6. **Conexión a la Base de Datos**:

- El código establece una conexión a una base de datos MySQL utilizando la extensión MySQL. Se proporcionan los detalles de la conexión, como la dirección del servidor, el nombre de usuario y la contraseña.

7. **Verificación de la Conexión a la Base de Datos**:

- Se verifica si la conexión a la base de datos se realizó con éxito. Si no, se muestra un mensaje de error y se detiene la ejecución del programa con **die()**.

8. **Inserción de Datos en la Base de Datos**:

- Se construye una consulta SQL para insertar los datos de temperatura, humedad y la fecha y hora actual en una tabla llamada "datos" en la base de datos.

9. **Verificación de la Inserción de Datos**:

- Se verifica si la inserción en la base de datos se realizó correctamente. Si es exitosa, se muestra un mensaje de éxito. De lo contrario, se muestra un mensaje de error que incluye detalles del error.

10. **Cierre de la Conexión a la Base de Datos**:

- Finalmente, se cierra la conexión a la base de datos utilizando **$conn->close()**.

11. **Manejo de Errores**:

- Si no se reciben todos los datos necesarios (temperatura y humedad), se muestra un mensaje de error indicando que no se recibieron los datos necesarios.

12. **Configuración de la Zona Horaria:**

- La primera línea establece la zona horaria a "America/Bogota" utilizando date_default_timezone_set(). Esto asegura que las fechas y horas se registren y se muestren en la zona horaria correcta.

13. **Verificación de Datos:**

- El código verifica si se han recibido los datos de temperatura (temperature) y humedad (humidity) a través de una solicitud GET utilizando isset(). Esto se hace para asegurarse de que los datos necesarios estén presentes en la URL.

14. **Obtención de Datos:**

- Si los datos de temperatura y humedad están presentes, se obtienen y se almacenan en las variables $temperatura y $humedad.

15. **Obtención de Fecha y Hora Actual del Servidor:**

- Se obtiene la fecha y hora actual del servidor utilizando la función date(). El formato utilizado es 'Y-m-d H:i:s'.

16. Mensajes de Depuración:

- Se muestran mensajes de depuración en la página para verificar los datos recibidos. Esto se hace utilizando echo para imprimir los valores de temperatura, humedad y la fecha y hora actuales.

17. Conexión a la Base de Datos:

- El código establece una conexión a una base de datos MySQL utilizando la extensión MySQLi. Se proporcionan los detalles de la conexión, como la dirección del servidor, el nombre de usuario y la contraseña.

18. **Verificación de la Conexión a la Base de Datos:**

- Se verifica si la conexión a la base de datos se realizó con éxito. Si no, se muestra un mensaje de error y se detiene la ejecución del programa con die().

19. Inserción de Datos en la Base de Datos:

- Se construye una consulta SQL para insertar los datos de temperatura, humedad y la fecha y hora actual en una tabla llamada "datos" en la base de datos.

20. Verificación de la Inserción de Datos:

- Se verifica si la inserción en la base de datos se realizó correctamente. Si es exitosa, se muestra un mensaje de éxito. De lo contrario, se muestra un mensaje de error que incluye detalles del error.

21. Cierre de la Conexión a la Base de Datos:

- Finalmente, se cierra la conexión a la base de datos utilizando $conn->close().

22. Manejo de Errores:

- Si no se reciben todos los datos necesarios (temperatura y humedad), se muestra un mensaje de error indicando que no se recibieron los datos necesarios.

En resumen, este código PHP recibe datos de temperatura y humedad a través de una solicitud GET, los almacena en una base de datos MySQL y proporciona información de depuración y manejo de errores en función del éxito o el fracaso de estas operaciones.

El método “GET” es uno de los métodos HTTP utilizados para solicitar recursos (como páginas web, archivos, datos, etc.) desde un servidor web. Se caracteriza por enviar los datos a través de la URL como parámetros como se describe a continuación:

1. **Solicitud de Recursos**: Cuando un cliente, como un navegador web, realiza una solicitud a un servidor web, puede utilizar el método GET para solicitar un recurso, como una página web. La solicitud se envía al servidor web a través de una URL.

2. **Datos en la URL**: Los datos que se envían con el método GET se incluyen en la URL como pares de clave-valor. Por ejemplo, una URL podría verse así: “http://ejemplo.com/recursos?parametro1=valor1&parametro2=valor2”

En este caso, "parametro1" y "parametro2" son las claves, y "valor1" y "valor2" son los valores.

3. **Visibles en la URL**: Debido a que los datos se incluyen en la URL, son visibles en la barra de direcciones del navegador. Esto significa que cualquier usuario que vea la URL también verá los datos que se están enviando, lo que puede no ser adecuado para información confidencial.

4. **Limitaciones de Tamaño**: Existe un límite en la cantidad de datos que se pueden enviar a través del método GET. Este límite varía según el servidor web y el navegador, pero en general, es menos que el método POST.

5. **Cacheable**: Las solicitudes GET son generalmente cacheables, lo que significa que los resultados de la solicitud se pueden almacenar en caché en el navegador o en servidores intermedios, lo que puede acelerar la carga de páginas web.

6. **Seguridad**: El método GET se utiliza principalmente para solicitudes de lectura y recuperación de información. No se utiliza para cambiar el estado del servidor ni para realizar acciones sensibles, ya que los datos son visibles y pueden ser manipulados fácilmente.

7. **Ejemplos de Uso**:

- Navegar por páginas web: Cuando ingresas una URL en tu navegador, se realiza una solicitud GET para obtener la página web correspondiente.
- Pasar parámetros a través de la URL: Muchas aplicaciones web utilizan GET para pasar datos en la URL, por ejemplo, cuando haces una búsqueda en un motor de búsqueda y ves resultados con parámetros en la URL.

En resumen, el método GET se utiliza para solicitar recursos y enviar datos a través de la URL. Es adecuado para solicitudes de lectura y recuperación de información, pero no es seguro para enviar datos confidenciales, ya que son visibles en la URL. Las solicitudes GET son ampliamente utilizadas en la navegación web y en aplicaciones donde la visibilidad de los datos no es un problema.

4.1.5. Bloque 5

Modificación y reestructuración código ARDUINO IDE para el envío de datos al servidor de base de datos según Código del anexo 4

Esto segmentos de códigos se adicionaron para enviar datos al servidor especificado en la URL **serverURL** utilizando una solicitud HTTP GET. A continuación, se describe lo que sucede en cada parte del bloque:

1. **WiFiClient client; // Crea una instancia de WiFiClient**:

- Se crea una instancia del objeto **WiFiClient** llamada **client**. Este objeto se utilizará para realizar la conexión con el servidor mediante HTTP.

2. **HTTPClient http;**

- Se crea una instancia del objeto **HTTPClient** llamada **http**. Este objeto se utilizará para realizar solicitudes HTTP, como GET o POST, al servidor.

3. **String url = serverURL;**:

- Se crea una variable **url** y se inicializa con el valor de la URL de destino que se ha especificado en **serverURL**.

4. **url += "?temperature=" + String(temperatura);**:

- Se concatenan parámetros a la URL. En este caso, se agrega el parámetro "temperature" junto con el valor de la variable **temperatura** en el formato adecuado para ser enviado a través de una solicitud GET. Esto se hace para enviar datos de temperatura al servidor.

5. **url += "&humidity=" + String(humedad);**:

- Se agrega otro parámetro a la URL, en este caso "humidity", junto con el valor de la variable **humedad**, nuevamente en el formato adecuado para ser enviado a través de una solicitud GET. Esto se hace para enviar datos de humedad al servidor.

6. **Serial.print("Enviando URL: ");**:

- Se imprime un mensaje de depuración en el puerto serie para indicar que se está enviando la URL al servidor.

7. **Serial.println(url);**:

- Se imprime la URL completa que se enviará al servidor. Esta línea de código es útil para depurar y verificar la URL que se está utilizando.

8. **http.begin(client, url); // Usa el objeto client como argumento para begin**:

- Se inicia una solicitud HTTP utilizando el objeto **http**. La URL que se ha construido y el objeto **client** se pasan como argumentos al método **begin()**. Esto prepara la solicitud HTTP para ser enviada al servidor.

9. **int httpCode = http.GET();**:

- Se realiza una solicitud GET al servidor mediante **http.GET()**. El código de respuesta del servidor se almacena en la variable **httpCode**.

10. **http.end();**:

- Se finaliza la solicitud HTTP. Esto libera recursos y cierra la conexión con el servidor.

11. **if (httpCode == 200) { ... } else { ... }**:

- Se verifica el código de respuesta del servidor. Si el código es 200 (que significa "OK"), se asume que la solicitud fue exitosa, y se realiza una serie de acciones,

como mostrar un mensaje de éxito, actualizar las últimas temperaturas y humedades (**lastTemperature** y **lastHumidity**), y llamar a la función **handleRoot()** (que probablemente actualiza la página web con los datos recién enviados).

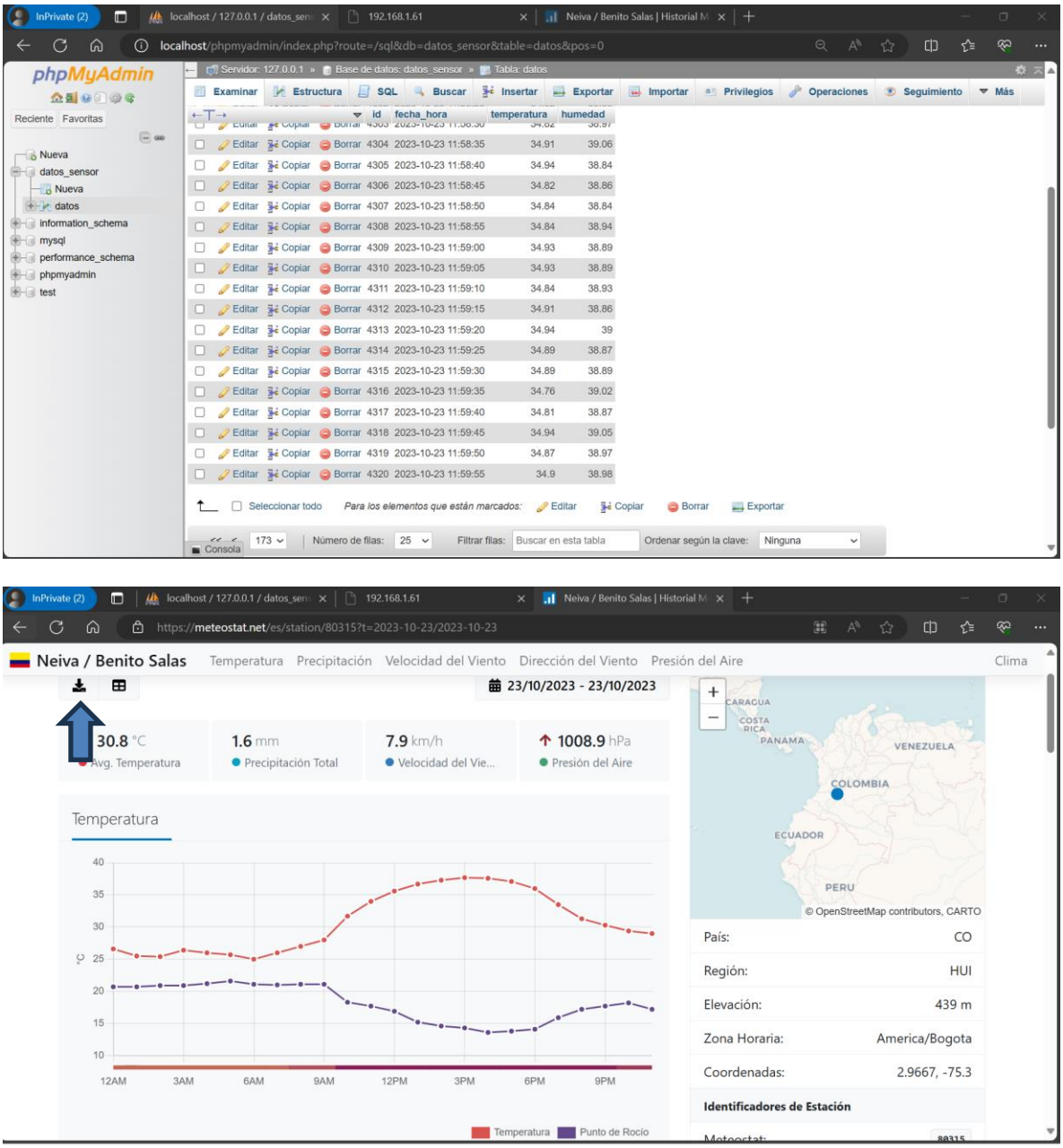
- Si el código de respuesta no es 200, se asume que la solicitud no fue exitosa y se muestra un mensaje de error que incluye el código de respuesta del servidor.

En resumen, este bloque de código se encarga de construir una URL que incluye datos de temperatura y humedad, enviar una solicitud GET al servidor especificado y manejar la respuesta del servidor en función del código de respuesta recibido.

4.1.6. Monitoreo y evaluación de datos

Se realizo un monitoreo y captura de datos el día 23/10/2023 desde las 6:00:00 am hasta las 12:00:00 m , con un intervalo de captura de datos de cada 5 segundos , se almacenaron un total de 4321 datos del sensor en la base de datos de phpmyadmin se tomaron como referencia para la validación de los mismos los datos provenientes de la estación climatológica del aeropuerto de la ciudad de Neiva Benito salas desde la url: https://meteostat.net/es/station/80315?t=2023-10-23/2023-10-23.

Figura 4-16. Reporte visual datos sobre medición temperatura y humedad



Fuente: Autor

A partir de los datos obtenidos y los datos de referencia sacamos el promedio por hora y calculamos la diferencia de valores entre ambos obteniendo como resultado una diferencia promedio de temperatura de 0,72 grados centígrados y para la humedad relativa 1,55%.

Tabla 4-1. Monitoreo de datos

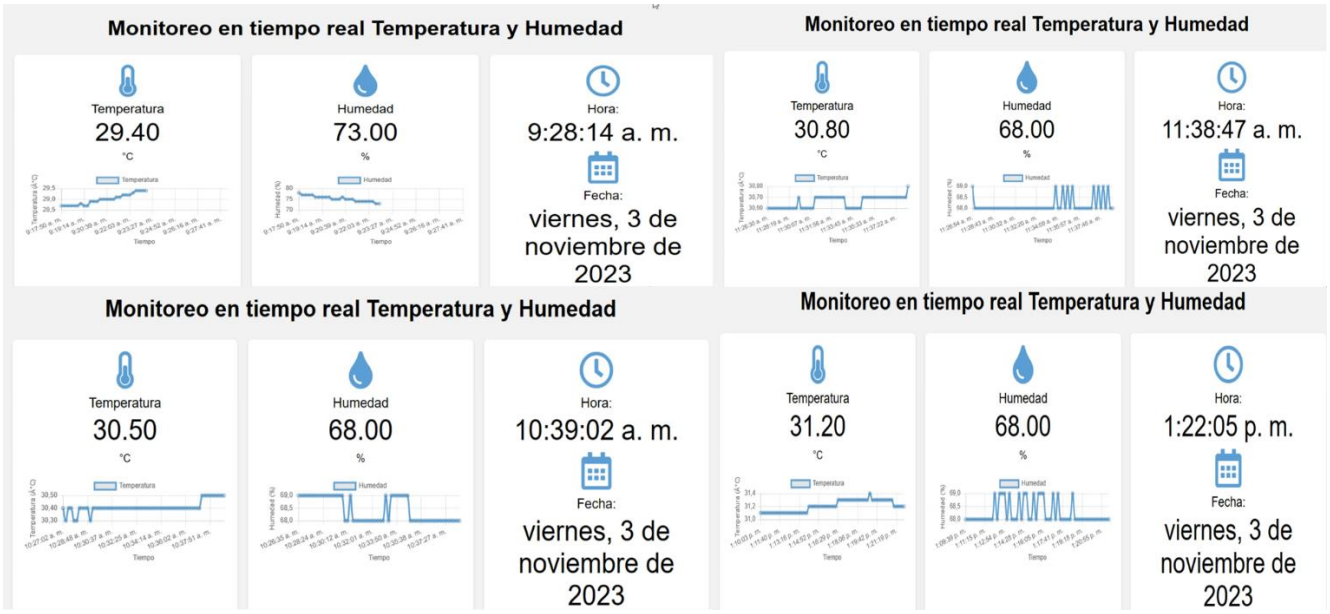
23/10/2023	Promedio sensor		Promedio Referencia		Diferencia	
hora	temperatura	Humedad	temperatura	Humedad	temperatura	Humedad
6:00:00	25,62	80,98	25,00	79,00	0,62	1,98
7:00:00	26,65	75,86	26,00	74,00	0,65	1,86
8:00:00	27,68	71,76	27,00	70,00	0,68	1,76
9:00:00	28,70	67,65	28,00	66,00	0,70	1,65
10:00:00	32,49	46,12	31,70	45,00	0,79	1,12
11:00:00	34,85	38,95	34,00	38,00	0,85	0,95
Total	29,33	63,55	28,62	62,00	0,72	1,55

3/11/2023	Promedio sensor		Promedio Referencia		Diferencia	
hora	temperatura	Humedad	temperatura	Humedad	temperatura	Humedad
6:00:00	26,56	80,45	26,12	79,13	0,44	1,32
7:00:00	27,41	80,15	26,89	79,15	0,52	1,00
8:00:00	28,38	80,05	27,56	79,10	0,82	0,95
9:00:00	29,27	73,87	28,87	72,10	0,40	1,77
10:00:00	30,43	68,33	29,96	67,55	0,47	0,78
11:00:00	30,64	68,42	29,86	67,23	0,78	1,19
12:00:00	30,92	68,33	30,52	67,75	0,40	0,58
13:00:00	31,24	67,83	30,85	67,12	0,39	0,71
14:00:00	31,81	63,67	31,2	62,98	0,61	0,69
15:00:00	32,27	59,50	31,95	58,69	0,32	0,81
16:00:00	32,27	59,50	31,89	58,89	0,38	0,61

Fuente: Autor

De acuerdo con estos resultados deducimos que el sistema funciona de manera óptima y eficiente teniendo en cuenta que los datos de referencia no manejan decimales se puede decir que el objetivo se cumple a cabalidad.

Figura 4-17. Visualización monitoreo



Conclusiones

Integración Exitosa de Sensores y Plataforma de Transmisión: La implementación de sensores de temperatura y humedad junto con la adquisición de datos de fecha y hora mediante un sistema basado en ESP8266 en lenguaje Arduino ha demostrado ser exitosa. La plataforma logro capturar, procesar y transmitir eficazmente estos datos a través de una conexión WiFi a una página web en HTML y a un servidor local de MySQL ejecutado en Apache. Esta integración proporciona una solución sólida para la monitorización en tiempo real de las condiciones ambientales y meteorológicas.

Datos Accesibles y Visualización en Tiempo Real: La transmisión de datos a la página web en HTML permitió la visualización en tiempo real de las variables de temperatura y humedad junto con el registro de fecha y hora. Esto brinda a los usuarios una forma accesible y conveniente de monitorear las condiciones ambientales y meteorológicas desde cualquier ubicación con acceso a Internet. La capacidad de acceder a estos datos en tiempo real es valiosa para tomar decisiones informadas y responder rápidamente a cambios en las condiciones.

Almacenamiento Seguro y Acceso a Largo Plazo: La conexión exitosa con el servidor local de MySQL a través de XAMP garantiza un almacenamiento seguro y accesible a largo plazo de los datos recopilados. Esto es esencial para el análisis retrospectivo, el seguimiento histórico de las condiciones y la generación de informes a largo plazo. La combinación de transmisión en tiempo real a una página web y almacenamiento en una base de datos MySQL ofrece un enfoque completo para el manejo de datos ambientales y meteorológicos en un entorno controlado.

Anexos

Anexo 1. Código NodeMCU

Paso 8: Escribir el Código

A continuación, escribiremos el código para

```
#include <Adafruit_Sensor.h>
#include <DHT.h>

#define DHTPIN 3      // Pin donde está conectado el sensor DHT11 (D2 en NodeMCU)
#define DHTTYPE DHT11 // Tipo de sensor DHT

DHT dht(DHTPIN, DHTTYPE);

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

void loop() {
  delay(1000);  // Espera 1 segundo entre lecturas

  float temperature = dht.readTemperature();
  float humidity = dht.readHumidity();

  Serial.print("Temperatura: ");
  Serial.print(temperature);
  Serial.print("°C\n");

  Serial.print("Humedad: ");
  Serial.print(humidity);
  Serial.print("%\n");
}
```

Anexo 2. Código página web

“pageindex1.h”

```
#include <ESP8266WiFi.h>
#include <DHT.h>
#include <ESP8266WebServer.h>
#include "pageindex1.h"

#define DHTPIN D3
#define DHTTYPE DHT11
#define LEDonBoard 2

const char* ssid = "FLIA_SERRATO_VILLALBA";
const char* password = "James.901003.*#";

const char* serverURL = "/index.php";

float lastTemperature = 0;
float lastHumidity = 0;

ESP8266WebServer server(80);
DHT dht(DHTPIN, DHTTYPE);

void handleRoot() {
  String page = SENSOR_page;
  page.replace("%TEMPERATURA%", String(lastTemperature));
  page.replace("%HUMEDAD%", String(lastHumidity));
  server.send(200, "text/html", page);
}

void handleSaveData() {
  float temperature = dht.readTemperature();
  float humidity = dht.readHumidity();

  if (!isnan(temperature) && !isnan(humidity)) {
    lastTemperature = temperature;
    lastHumidity = humidity;
  }

  handleRoot();
}

void handleTemperature() {
  float temperature = dht.readTemperature();
  if (!isnan(temperature)) {
    server.send(200, "text/plain", String(temperature));
  } else {
    server.send(500, "text/plain", "Error al leer la temperatura.");
  }
}
```

```
void handleHumidity() {
  float humidity = dht.readHumidity();
  if (!isnan(humidity)) {
    server.send(200, "text/plain", String(humidity));
  } else {
    server.send(500, "text/plain", "Error al leer la humedad.");
  }
}


void setup() {
  Serial.begin(115200);
  delay(10);
  dht.begin();

  WiFi.begin(ssid, password);
  Serial.println("");

  pinMode(LEDonBoard, OUTPUT);
  digitalWrite(LEDonBoard, HIGH);

  while (WiFi.status() != WL_CONNECTED) {
    delay(1000);
    Serial.println("Conectando a WiFi...");
    digitalWrite(LEDonBoard, LOW);
    delay(250);
    digitalWrite(LEDonBoard, HIGH);
  }
  digitalWrite(LEDonBoard, HIGH);

  Serial.println("");
  Serial.print("Conexión exitosa a la red: ");
  Serial.println(ssid);
  Serial.print("Dirección IP: ");
  Serial.println(WiFi.localIP());
  delay(10);

  while (WiFi.status() != WL_CONNECTED) {
    delay(1000);
    Serial.println("Conectando a WiFi...");
  }

  server.on("/", handleRoot);
  server.on("/index.php", handleSaveData);

  // Agregar los manejadores para las rutas de temperatura y humedad
  server.on("/readTemperature", handleTemperature);
  server.on("/readHumidity", handleHumidity);

  server.begin();
  Serial.println("Servidor HTTP inicializado");
```

```
}

void loop() {
  server.handleClient();
  handleSaveData();

  float temperature = dht.readTemperature();
  float humidity = dht.readHumidity();

  if (!isnan(temperature) && !isnan(humidity)) {
    Serial.print("Temperatura: ");
    Serial.print(temperature);
    Serial.print(" °C, Humedad: ");
    Serial.print(humidity);
    Serial.println(" %");
  }

  delay(1000);
}
```

Anexo 3.

```
const char SENSOR_page[] PROGMEM = R"=====(
<!DOCTYPE html>
<html>

<head>
  <meta name="viewport" content="width=device-width, initial-scale=1">                    <!-- Configuración de la vista del dispositivo -->
  <link href="https://use.fontawesome.com/releases/v5.7.2/css/all.css" rel="stylesheet">          <!-- Enlace a la hoja de estilos de Font Awesome -->
  <script src="https://cdn.jsdelivr.net/npm/chart.js"></script>                    <!-- Enlace al archivo de script de Chart.js -->
  <style>
    html {
      font-family: Arial, sans-serif;                                    /* Estilo de fuente y tipo de letra para el documento */
      display: flex;
      justify-content: center;
      align-items: center;
      height: 100%;
    }

    body {
      background-color: #f2f2f2;                                     /* Color de fondo del cuerpo */
      margin: 0;
      padding: 20px;
      display: flex;
      flex-direction: column;
      align-items: center; /* Centrar verticalmente los contenedores */
    }

    .card {
      background-color: #ffffff;                                    /* Color de fondo de las tarjetas */
      border-radius: 8px;
      box-shadow: 0px 2px 4px rgba(0, 0, 0, 0.1);                          /* Sombra de las tarjetas */
      padding: 20px;
      margin: 10px;
      text-align: center;
      flex-basis: 30%; /* Establecer el ancho de los contenedores */
    }

    h1 {
      font-size: 2.5rem;
      margin-bottom: 20px;
    }

    .sensor-icon {
      font-size: 4rem;
```

```
    color: #62a1d3;                                    /* Color del icono del sensor */
    margin-bottom: 10px;
   }

   .sensor-label {
    font-size: 1.5rem;
    margin-bottom: 5px;
   }

   .sensor-value {
    font-size: 3rem;
    margin-bottom: 10px;
   }

   .units {
    font-size: 1.2rem;
   }

   .timestamp {
    font-size: 1.2rem;
    margin-top: 20px;
   }


   .container {
    display: flex;
    justify-content: space-between;
    width: 100%;
   }

   .chart-container {
    width: 100%;
    max-width: 500px;
    margin-top: 20px;
   }


  </style>
</head>


<body>
  <h1>Monitoreo en tiempo real Temperatura y Humedad</h1>                <!-- Encabezado -->

  <div class="container">                                    <!-- Contenedor principal -->
    <div class="card">                                     <!-- Tarjeta 1 -->
      <div>
        <i class="fas fa-thermometer-half sensor-icon"></i>                 <!-- Icono de termómetro -->
        <div class="sensor-label">Temperatura</div>                        <!-- Etiqueta de temperatura --
>
        <div class="sensor-value" id="TemperatureValue">{temperature}</div>          <!-- Valor de
temperatura actualizada -->
```

```
      <div class="units">&deg;C</div>                    <!-- Unidades de temperatura -->
    </div>

    <div class="chart-container">                    <!-- Contenedor del gráfico -->
      <canvas id="temperatureChart"></canvas>              <!-- Gráfico de temperatura -->
    </div>
  </div>

  <div class="card">                                 <!-- Tarjeta 2 -->
    <div>
      <i class="fas fa-tint sensor-icon"></i>              <!-- Icono de gota de agua -->
      <div class="sensor-label">Humedad</div>              <!-- Etiqueta de humedad -->
      <div class="sensor-value" id="HumidityValue">{humidity}</div>          <!-- Valor de
humedad inicializa en cero-->
      <div class="units">%</div>                    <!-- Unidades de humedad -->
    </div>

    <div class="chart-container">
      <canvas id="humidityChart"></canvas>
    </div>
  </div>


  <div class="card">                                 <!-- Tarjeta 3 -->
    <div>
      <i class="far fa-clock sensor-icon"></i>             <!-- Icono de reloj -->
      <div class="sensor-label">Hora:</div>                <!-- Etiqueta de hora -->
      <div class="sensor-value" id="time"></div>           <!-- Valor de hora -->
    </div>

    <div>
      <i class="far fa-calendar-alt sensor-icon"></i>      <!-- Icono de calendario -->
      <div class="sensor-label">Fecha: </div>              <!-- Etiqueta de fecha -->
      <div class="sensor-value" id="date"></div>           <!-- Valor de fecha -->
    </div>
  </div>
 </div>
</body>



 <script>
  // Variables para almacenar datos de temperatura y etiquetas de tiempo
  var temperatureData = [];
  var timeLabels = [];
  var humidityData = [];

  // Crear gráfico de temperatura utilizando Chart.js
  var temperatureChart = new Chart(document.getElementById("temperatureChart"), {
    type: "line",
    data: {
      labels: timeLabels,                          // Etiquetas de tiempo en el eje X
```

```
    datasets: [
      {
        label: "Temperatura",                    // Etiqueta de la serie de datos
        data: temperatureData,                   // Datos de temperatura en el eje Y
        fill: false,                             // Sin relleno debajo de la línea
        borderColor: "#62a1d3",                  // Color del borde de la línea
        tension: 0.4                             // Tensión de la línea
      }
    ]
  },
  options: {
    responsive: true,                            // Hacer el gráfico responsivo
    maintainAspectRatio: false,                  // No mantener la proporción del aspecto
    scales: {
      x: {
        display: true,
        title: {
          display: true,
          text: "Tiempo"                         // Título del eje X (tiempo)
        }
      },
      y: {
        display: true,
        title: {
          display: true,
          text: "Temperatura (°C)"               // Título del eje Y (temperatura)
        }
      }
    }
  }
});

// Crear gráfico de humedad utilizando Chart.js
var humidityChart = new Chart(document.getElementById("humidityChart"), {
  type: "line",
  data: {
    labels: timeLabels,
    datasets: [
      {
        label: "Humedad",
        data: humidityData,
        fill: false,
        borderColor: "#62a1d3",
        tension: 0.4
      }
    ]
  },
  options: {
    responsive: true,
    maintainAspectRatio: false,
    scales: {
      x: {
```

```
        display: true,
        title: {
          display: true,
          text: "Tiempo"
        }
      },
      y: {
        display: true,
        title: {
          display: true,
          text: "Humedad (%)"
        }
      }
    }
  }
});


 // Actualizar los datos de temperatura, humedad, tiempo y fecha cada 10 segundos

   setInterval(function() {

     getTemperatureData();
     getHumidityData();

   }, 10000);

    setInterval(getTimeAndDate, 1000);


   // Obtener los datos de temperatura del sensor
function getTemperatureData() {
  var xhttp = new XMLHttpRequest();                                    // Se crea una instancia de
XMLHttpRequest, que es un objeto utilizado para realizar solicitudes HTTP asincrónicas.
  xhttp.onreadystatechange = function() {                              // Se define un manejador de evento
onreadystatechange para la solicitud XMLHttpRequest. Este manejador se activa cuando el estado de la
solicitud cambia.
    if (this.readyState == 4 && this.status == 200) {                  // Se verifica si la solicitud se ha
completado (readyState igual a 4) y si el código de estado de la respuesta es 200 (indicando una
respuesta exitosa).
      var temperature = parseFloat(this.responseText);                 // El valor de temperatura recibido
del servidor como una cadena de texto, lo convierte en un número decimal y lo asigna a la variable
temperature
      document.getElementById("TemperatureValue").innerHTML = temperature.toFixed(2);    //
Actualizar el valor de temperatura en la página

      temperatureData.push(temperature);                               // Agregar el valor de temperatura al
arreglo de datos
      if (temperatureData.length > 60) {
        temperatureData.shift();                                       // Limitar la longitud del arreglo a 60 (1
minuto)
      }
```

```
    var now = new Date();
    var timeLabel = now.toLocaleTimeString();                    // Obtener la etiqueta de tiempo actual
    timeLabels.push(timeLabel);                                  // Agregar la etiqueta de tiempo al arreglo de
etiquetas
    if (timeLabels.length > 60) {
      timeLabels.shift();                                        // Limitar la longitud del arreglo a 60 (1 minuto)
    }

    temperatureChart.update();                                   // Actualizar el gráfico de temperatura
   }
  };
  xhttp.open("GET", "readTemperature", true);                    // Realizar una solicitud GET para
obtener los datos de temperatura del servidor

  xhttp.send();
}

   // Obtener los datos de humedad del sensor
function getHumidityData() {
  var xhttp = new XMLHttpRequest();                              // Se crea una instancia de
XMLHttpRequest, que es un objeto utilizado para realizar solicitudes HTTP asincrónicas.
  xhttp.onreadystatechange = function() {                        // Se define un manejador de evento
onreadystatechange para la solicitud XMLHttpRequest. Este manejador se activa cuando el estado de la
solicitud cambia.
   if (this.readyState == 4 && this.status == 200) {             // Se verifica si la solicitud se ha
completado (readyState igual a 4) y si el código de estado de la respuesta es 200 (indicando una
respuesta exitosa).
    var humidity = parseFloat(this.responseText);                // El valor de temperatura recibido del
servidor como una cadena de texto, lo convierte en un número decimal y lo asigna a la variable
temperature
    document.getElementById("HumidityValue").innerHTML = humidity.toFixed(2);    // Actualizar el
valor de temperatura en la página

    humidityData.push(humidity);                                 // Agregar el valor de temperatura al arreglo de
datos
    if (humidityData.length > 60) {
      humidityData.shift();                                      // Limitar la longitud del arreglo a 60 (1 minuto)
    }

    var now = new Date();
    var timeLabel = now.toLocaleTimeString();                    // Obtener la etiqueta de tiempo actual
    timeLabels.push(timeLabel);                                  // Agregar la etiqueta de tiempo al arreglo de
etiquetas
    if (timeLabels.length > 60) {
      timeLabels.shift();                                        // Limitar la longitud del arreglo a 60 (1 minuto)
    }

    humidityChart.update();                                      // Actualizar el gráfico de temperatura
   }
  };
```

```
  xhttp.open("GET", "readHumidity", true);                    // Realizar una solicitud GET para
obtener los datos de temperatura del servidor
  xhttp.send();
}

   // funcion para obtener el timepoy actualizarlo
   function getTimeAndDate() {
     var now = new Date();
     var time = now.toLocaleTimeString();                     // Obtener la hora actual en formato de
cadena
     var date = now.toLocaleDateString("es-CO", {             // Obtener la fecha actual en
formato de cadena hora colombia
       weekday: "long",                                       // Día de la semana completo (lunes, martes,
etc.)
       year: "numeric",                                       // Año en formato numérico (ejemplo: 2023)
       month: "long",                                         // Mes completo (enero, febrero, etc.)
       day: "numeric",                                        // Día del mes en formato numérico (ejemplo: 1,
2, etc.)
       timeZone: "America/Bogota"                             // Zona horaria
     });

     document.getElementById("time").innerHTML = time;        // Mostrar la hora en la
página
     document.getElementById("date").innerHTML = date;        // Mostrar la fecha en la
página
   }

</script>

</html>
)=====";
```

Anexo código script

```
<?php

// Establecer la zona horaria a Colombia
date_default_timezone_set('America/Bogota');

// Verificar si se recibieron los datos de temperatura y humedad
if (isset($_GET['temperature']) && isset($_GET['humidity'])) {
    // Obtener los datos de temperatura y humedad
    $temperatura = $_GET['temperature'];
    $humedad = $_GET['humidity'];

    // Obtener la fecha y hora actual del servidor
    $fechaHora = date('Y-m-d H:i:s');

    // Agregar mensajes de depuración para verificar los datos recibidos
    echo "Temperatura recibida: " . $temperatura . "<br>";
    echo "Humedad recibida: " . $humedad . "<br>";
    echo "Fecha y hora actual del servidor: " . $fechaHora . "<br>";

    // Conectar a la base de datos
    $servername = "localhost"; // Cambiar por la dirección de tu servidor MySQL
    $username = "root"; // Cambiar por el usuario de tu base de datos
    $password = ""; // Cambiar por la contraseña de tu base de datos
    $dbname = "datos_sensor"; // Cambiar por el nombre de tu base de datos

    $conn = new mysqli($servername, $username, $password, $dbname);

    // Verificar si la conexión fue exitosa
    if ($conn->connect_error) {
```

```
        die("Conexión fallida: " . $conn->connect_error);
    }

    // Insertar los datos en la base de datos
    $sql = "INSERT INTO datos (fecha_hora, temperatura, humedad) VALUES ('$fechaHora', '$temperatura', '$humedad')";

    if ($conn->query($sql) === TRUE) {
        echo "Datos insertados correctamente en la base de datos.";
    } else {
        echo "Error al insertar los datos: " . $conn->error;
    }

    // Cerrar la conexión
    $conn->close();
} else {
    echo "Error: No se recibieron todos los datos necesarios.";
}
?>
```

Anexo 4. Código Arduino

```
#include <ESP8266WiFi.h>
#include <WiFiClient.h>
#include <ESP8266WebServer.h>
#include <DHT.h>
#include <ESP8266HTTPClient.h>
#include "pageindex1.h"

#define DHTPIN D3
#define DHTTYPE DHT11
#define LEDonBoard 2



const char* ssid = "FLIA_SERRATO_VILLALBA";
const char* password = "James.901003.*#";
//const char* serverURL =
"C:\\xampp\\htdocs\\Proyecto_Sensor_DHT11\\index.php"; // Cambia esto por la
URL de tu servidor y el script PHP que procesa los datos
const char* serverURL = "http://192.168.1.61/Proyecto_Sensor_DHT11/index.php";
// Cambia esto por la URL de tu servidor y el script PHP que procesa los datos




float lastTemperature = 0;
float lastHumidity = 0;

ESP8266WebServer server(80);
DHT dht(DHTPIN, DHTTYPE);

void handleRoot() {
  String s = SENSOR_page;
  s.replace("{temperature}", String(lastTemperature));
  s.replace("{humidity}", String(lastHumidity));
  server.send(200, "text/html", s);
}

void handleSaveData() {
  float temperatura = dht.readTemperature();
  float humedad = dht.readHumidity();

  if (isnan(temperatura) || isnan(humedad)) {
```

```
    Serial.println("Error al leer los datos del sensor DHT");
    return;
  }

  WiFiClient client; // Crea una instancia de WiFiClient
  HTTPClient http;
  String url = serverURL;
  url += "?temperature=" + String(temperatura);
  url += "&humidity=" + String(humedad);

  Serial.print("Enviando URL: ");
  Serial.println(url);

  http.begin(client, url); // Usa el objeto client como argumento para begin
  int httpCode = http.GET();
  http.end();

  if (httpCode == 200) {
    Serial.println("Datos enviados correctamente al servidor. ;)");
    lastTemperature = temperatura;
    lastHumidity = humedad;
    handleRoot();
  } else {
    Serial.print("Error al enviar los datos al servidor. Código de respuesta:
");
    Serial.println(httpCode);
  }
}

// Manejador para obtener los datos de temperatura
void handleTemperature() {
  float temperature = dht.readTemperature();
  server.send(200, "text/plain", String(temperature));
}

// Manejador para obtener los datos de humedad
void handleHumidity() {
  float humidity = dht.readHumidity();
  server.send(200, "text/plain", String(humidity));
}

void setup() {
  Serial.begin(115200);
  delay(10);
```

```
  dht.begin();

  WiFi.begin(ssid, password);
  Serial.println("");

  pinMode(LEDonBoard, OUTPUT);
  digitalWrite(LEDonBoard, HIGH);

  while (WiFi.status() != WL_CONNECTED) {
    delay(1000);
    Serial.println("Conectando a WiFi...");
    digitalWrite(LEDonBoard, LOW);
    delay(250);
    digitalWrite(LEDonBoard, HIGH);
  }
  digitalWrite(LEDonBoard, HIGH);

  Serial.println("");
  Serial.print("Conexión exitosa a la red: ");
  Serial.println(ssid);
  Serial.print("Dirección IP: ");
  Serial.println(WiFi.localIP());

  server.on("/", handleRoot);
  server.on("/index.php", handleSaveData);

  // Agregar los manejadores para las rutas de temperatura y humedad
  server.on("/readTemperature", handleTemperature);
  server.on("/readHumidity", handleHumidity);

  server.begin();
  Serial.println("Servidor HTTP inicializado");
}

void loop() {
  server.handleClient();

  handleSaveData();

  float temperatura = dht.readTemperature();
  float humedad = dht.readHumidity();

  if (isnan(temperatura) || isnan(humedad)) {
```

```
    Serial.println("Error al leer los datos del sensor DHT");
  } else {
    Serial.print("Temperatura: ");
    Serial.print(temperatura);
    Serial.print(" °C, Humedad: ");
    Serial.print(humedad);
    Serial.println(" %");
  }

  delay(1000);
}
```

A la codificación previa se implementa estos dos segmentos de codigo para cumplir la función del envió de datos :

```
  WiFiClient client; // Crea una instancia de WiFiClient

  HTTPClient http;
  String url = serverURL;
  url += "?temperature=" + String(temperatura);
  url += "&humidity=" + String(humedad);

  Serial.print("Enviando URL: ");
  Serial.println(url);

  http.begin(client, url); // Usa el objeto client como argumento para begin
  int httpCode = http.GET();
  http.end();

  if (httpCode == 200) {
    Serial.println("Datos enviados correctamente al servidor. ;)");
    lastTemperature = temperatura;
    lastHumidity = humedad;
    handleRoot();
  } else {
    Serial.print("Error al enviar los datos al servidor. Código de respuesta:
");
    Serial.println(httpCode);
  }
}
```

Referencias Bibliográficas

Barbecho, E. (2022). *Diseño e implmentación de una plataforma basada en Iot.* Cuenca (Ecuador): Universidad Politécnica Saleciana. Obtenido de https://red.uao.edu.co/bitstream/handle/10614/10213/T07859.pdf?sequence=5&isAllowed=y

Becerra, J. (2021). *Desarrollo de un sistema de control de tempertura y monitoreo de PH y humedad.* Santiago de Cali: Universidad del Valle. Obtenido de https://red.uao.edu.co/bitstream/handle/10614/10213/T07859.pdf?sequence=5&isAllowed=y

Farooq, M. (2020). *Role of IoT Technology in Agriculture: A Systematic Literature Review.* Seúl (Korea): MDPI.

Gómez, J. (2017). *NTERNET OF THINGS (IoT) SYSTEM FOR THE MONITORING OF PROTECTED CROP.* Monteria (Cordoba): Universidad de Córdoba.

https://robots-argentina.com.ar/. (17 de Agosto de 2023). Obtenido de https://robots-argentina.com.ar/didactica/medicion-de-temperatura-y-humedad-con-dht11/

Kim, W. (2020). *A Review of the Applications of the Internet of Things (IoT) for Agricultural Automation.* Hong Kong, (China): Researchgate.

Lakhiar, I. (2019). *Monitoring and Control Systems in Agriculture Using Intelligent Sensor Techniques: A Review of the Aeroponic System.* Zhenjiang (China): Hindawi.

Nyoman. (2018). *Hydroponic Management and Monitoring System for an IOT Based NFT Farm Using Web Technology.* Bali (Indonesia): Researchgate.

Quintero, G. (2022). *Efecto de dos condiciones de protección de cultivo sobre los índices de crecimiento y producción de tres variedades de lechuga (Lactuca sativa) tipo gourmet en la Sabana de Bogotá.* Bogotá: Universidad Nacional.

Robles, G. (14 de Agosto de 2019). *Sistema de medida de temperatura basado en NodeMCU y Android.* Madrid (España): Universidad Carlos III de Madrid. Obtenido de https://e-archivo.uc3m.es/bitstream/handle/10016/29308/TFG_Teresa_Castro_Blanco.pdf?sequence=1&isAllowed=y

Rojas, o. (2018). *DISEÑO E IMPLEMENTACIÓN DE UN SISTEMA AUTOMATIZADO PARA INVERNADERO HIDROPÓNICO.* Mariquita: UNAD.

Saavedra, G. (2018). *Manula de produccion d elechuga.* Santiago de Chile : INIA.

Salinas, Y. (2022). El impacto del internet de todas las cosas (IoT)en la vida cotidiana. *Revista multidisciplinar ciencia latina, 6*(2), 1369-1378. doi:https://doi.org/10.37811/cl_rcm.v6i2.1959

Singh, J. (2019). *Accountability in the IoT: Systems, Law, and Ways Forward.* Cambridge (Londres): Researchgate.

Tovar, J. (2018). *Internet de las cosas aplicado a la agricultura: estado actual.* Buenaventura (Colombia): Universidad de San Buenaventura.

Universidad de Antioquia. (12 de marzo de 2023). Obtenido de https://www.udea.edu.co/wps/portal/udea/web/inicio/extension/portafoliotecnologico/articulos/Agricultura_de_precision