

# Manual para el proyecto

# lefnica.

Antonio Miranda

David Langarica

Alan González

Enlace al repositorio:

<https://github.com/DavidLangarica/lefnica.git>

# Introducción al proyecto

**"Preservando la conservación de tus alimentos"**

## Sobre Iefnica.

Somos una empresa dedicada a la conservación de los alimentos, tenemos el principal objetivo de hacer que tus alimentos puedan durar más, con tan solo su ambiente de conservación, sin la necesidad de conservadores químicos o artificiales que puedan llegar a ser dañinos para la salud.

## Propósito

Evitar el desperdicio de alimentos, tratando de alargar su vida útil tanto como podamos con el ambiente de conservación.

## Misión

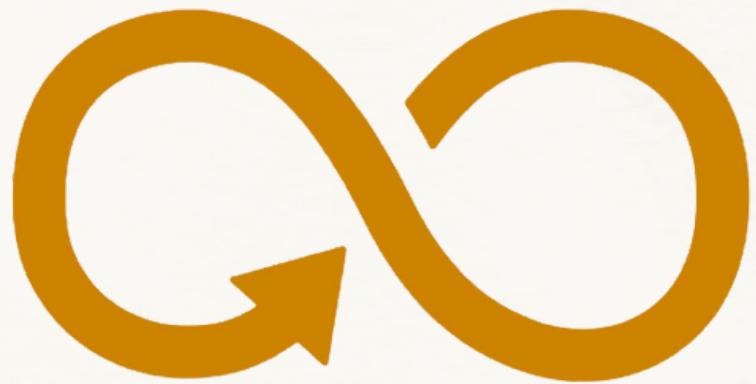
En Iefnica tenemos un lema, "Preservando la conservación de tus alimentos", que significa que te ayudaremos a mejorar la calidad de tus alimentos mediante su conservación, de este modo reduciendo el desperdicio de alimentos por todo el mundo, que por consecuente significara en un mundo más autosustentable en el que se evite el desperdicio.

# Objetivos de Desarrollo Sostenible

Para el desarrollo de Lefnica, utilizamos 2 Objetivos de Desarrollo Sostenible: Salud y Bienestar y Producción y consumo responsables



El Objetivo de Desarrollo Sostenible número 3, Salud y Bienestar se aplica en la función principal del proyecto, al buscar la mejor manera de almacenar los alimentos, estamos buscando mejorar la alimentación de las personas buscando la frescura y la conservación correcta de los alimentos.



El Objetivo de Desarrollo Sostenible número 12, Producción y Consumo Responsables trabaja en conjunto con el número 3, ya que al mejorar la forma de almacenamiento de los alimentos estamos evitando el desperdicio de los mismos.

Parte 1

# Base de datos

lefnica.

# Desarrollo de Base de Datos

**Para la base de datos, utilizamos la aplicación de MAMP, con la cual programamos la base de datos en MySQL**

Primero, instalamos MAMP (O XAMPP, ambas aplicaciones funcionan). Una vez instalada la aplicación, la ejecutamos para poder entrar a la página de PHPMyAdmin, dentro de esta y programando en MySQL, ejecutamos el siguiente código:

```

CREATE DATABASE LEFNICA;

USE LEFNICA;

#Creación de tablas

CREATE TABLE ALIMENTO(
Id_alimento INT AUTO_INCREMENT PRIMARY KEY NOT NULL,
Alimento VARCHAR(40) NOT NULL,
Humedad_ideal_rango_mayor VARCHAR(10),
Humedad_ideal_rango_menor VARCHAR(10),
Temperatura_ideal_rango_mayor VARCHAR(10),
Temperatura_ideal_rango_menor VARCHAR(10),
Tiempo TIME
);

CREATE TABLE NodeMCU(
Id_NODEMCU INT AUTO_INCREMENT PRIMARY KEY NOT NULL,
Temperatura INT,
Humedad INT,
Tiempo TIME
);

CREATE TABLE DATOS_ALIMENTO(
Id_table INT PRIMARY KEY NOT NULL
);

#Datos inciales

INSERT INTO ALIMENTO VALUES (1, "CARNE", 95, 90, 5, 4, 0);
INSERT INTO ALIMENTO VALUES (2, "VERDURAS", 90, 85, 13, 1, 1);
INSERT INTO ALIMENTO VALUES (3, "FRUTA", 90, 85, 13, 1, 2);
INSERT INTO ALIMENTO VALUES (4, "LEGUMINOSAS", 86, 88, 20, -30, 3);
INSERT INTO ALIMENTO VALUES (5, "PESCADO", 95, 90, 5, 4, 4);
INSERT INTO ALIMENTO VALUES (6, "POLLO", 95, 90, 4, 3, 5);

USE LEFNICA;

SELECT ALIMENTO.Alimento AS "Tipo de Alimento", ALIMENTO.Humedad_ideal_rango_menor AS "Humedad Mínima",
NodeMCU.Humedad AS "Humedad actual", ALIMENTO.Humedad_ideal_rango_mayor AS "Humedad Máxima",
ALIMENTO.Temperatura_ideal_rango_menor AS "Temperatura Mínima", NodeMCU.Temperatura AS "Temperatura actual",
ALIMENTO.Temperatura_ideal_rango_mayor AS "Temperatura Máxima"
FROM ALIMENTO, NodeMCU
WHERE ALIMENTO.Tiempo = NodeMCU.Tiempo

```

Una vez ejecutado el código, la base de datos está creada, ya contando con los datos sobre los tipos de alimentos disponibles y sus respectivas especificaciones para su conservación.

Se generaron varias tablas, las cuales contarán con información particular que se conectarán entre sí, entre ellas está la tabla de datos iniciales, una tabla de almacenamiento de datos que reciba mediante la conexión con el NodeMCU, desplegando los mismos en la página web.

Enlaces de para descargar MAMP o XAMPP:

- <https://www.mamp.info/en/windows/>
- <https://www.apachefriends.org/es/index.html>

En el repositorio de GitHub encontrarás el archivo .sql que puedes cargar en phpmyadmin para importar la base de datos con las tablas necesarias:

Archivos necesarios para esta parte: lefnica.sql

<https://github.com/DavidLangarica/lefnica.git>

# Parte 2

# Node MCU

lefnica.

# Node MCU

## Contenido

- Especificaciones del proyecto (8-9)
- Configuración del IDE de Arduino (9-11)
- Alambrado (12)
- Código del proyecto (12-15)
- Conexión al servidor (15-17)
- Iniciando el proyecto (17-18)

# Node MCU

## Especificaciones del proyecto

### **Micro controlador: ESP8266**

Ocupamos este microcontrolador programable desde el IDE de arduino, cuya principal ventaja sobre arduino es que puede conectarse a internet de forma nativa sin necesidad de un modulo aparte. Este lo ocuparemos para recopilar los datos de nuestro sensor y enviarlos a la base de datos.

### **Sensor de humedad: DHT-11 de tres terminales**

El sensor DHT-11 nos brinda una lectura precisa de la humedad y la temperatura, es importante recalcar que usamos el sensor con un adaptador de tres terminales, esto por que la libreria que ocupamos es diferente.

### **2 Leds de control**

Estos led's iran conectados al node y nos serviran como verificadores para cuando se conecte nuestro node a internet y cuando se conecte a la base de datos.

# Node MCU

## Especificaciones del proyecto

### Servidor local puenteado desde playit.gg

Con el fin de tener un proyecto completamente funcional, haremos uso de la plataforma play it para poder convertir nuestra laptop o pc en un servidor funcional.

## Configuracion del IDE de arduino

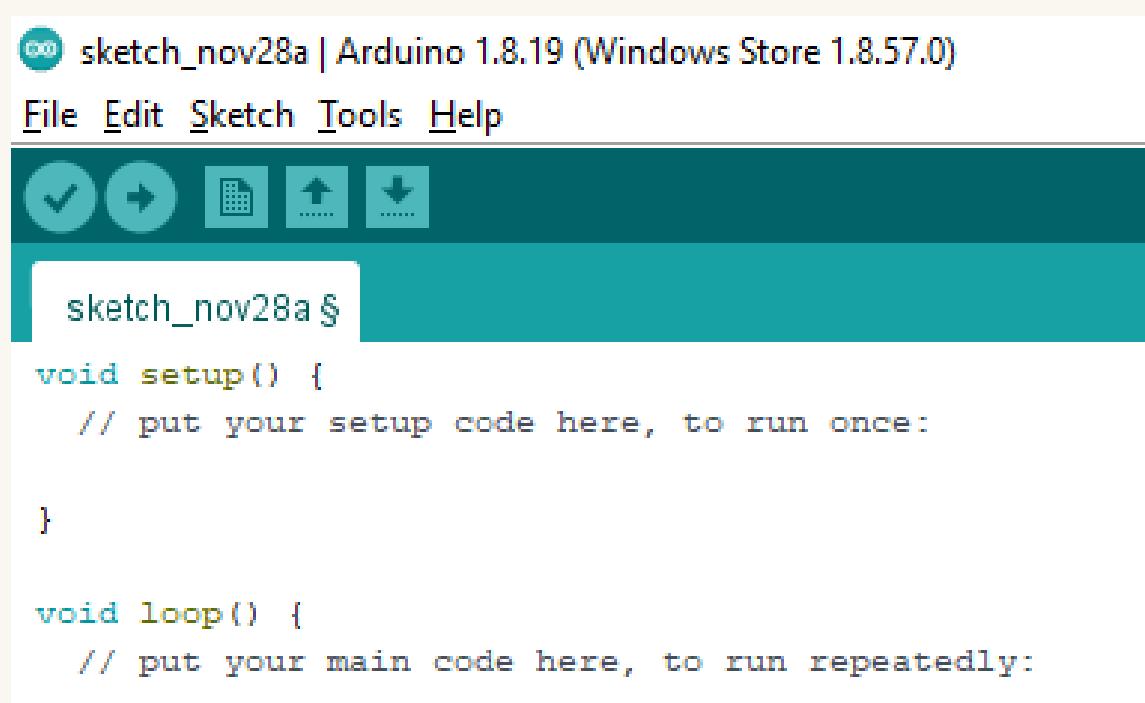
Al usar el node MCU primero que nada tendremos que descargar el IDE de arduino, en este caso es tan simple como bajarlo desde la [Microsoft store](#) despues de esto tendremos que descargar e instalar los drivers del node MCU (<https://www.silabs.com/>) que podras encontrar en el git de este proyecto ;). Para instalarlos basta con dar doble click en el instalador.

Con todo esto listo abriremos el IDE y lo configuraremos para poder trabajar con la board 8266.

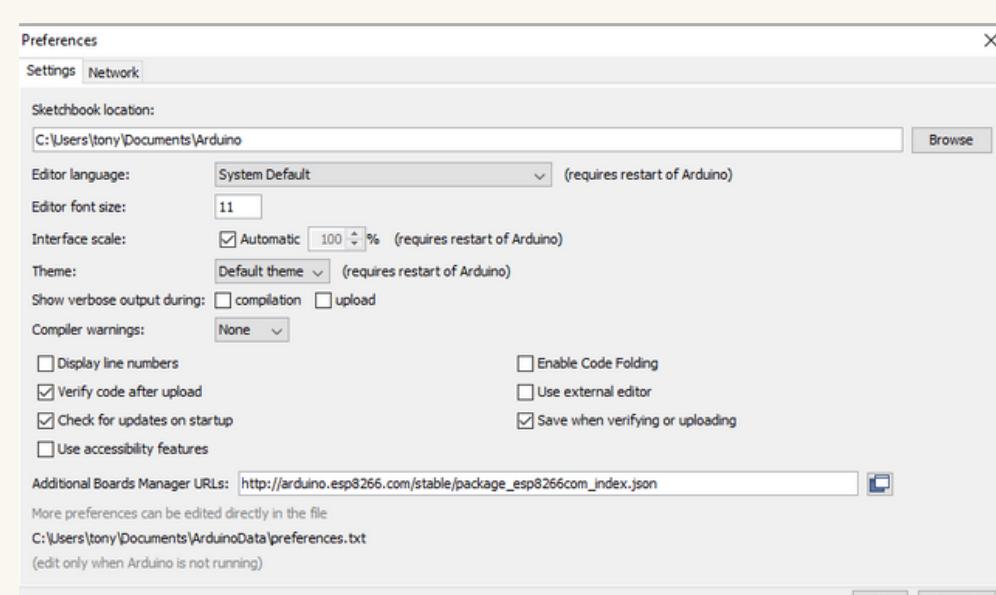
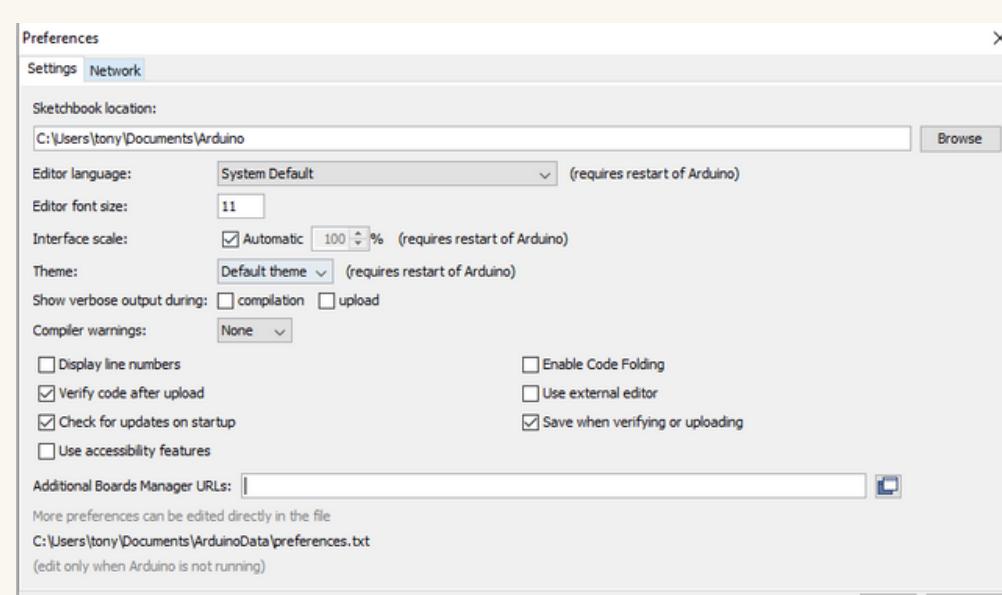
# Node MCU

## Configuracion del IDE de arduino

Como primer paso, accederemos a la pestaña de "File", donde seleccionaremos la sección de "Preferences"



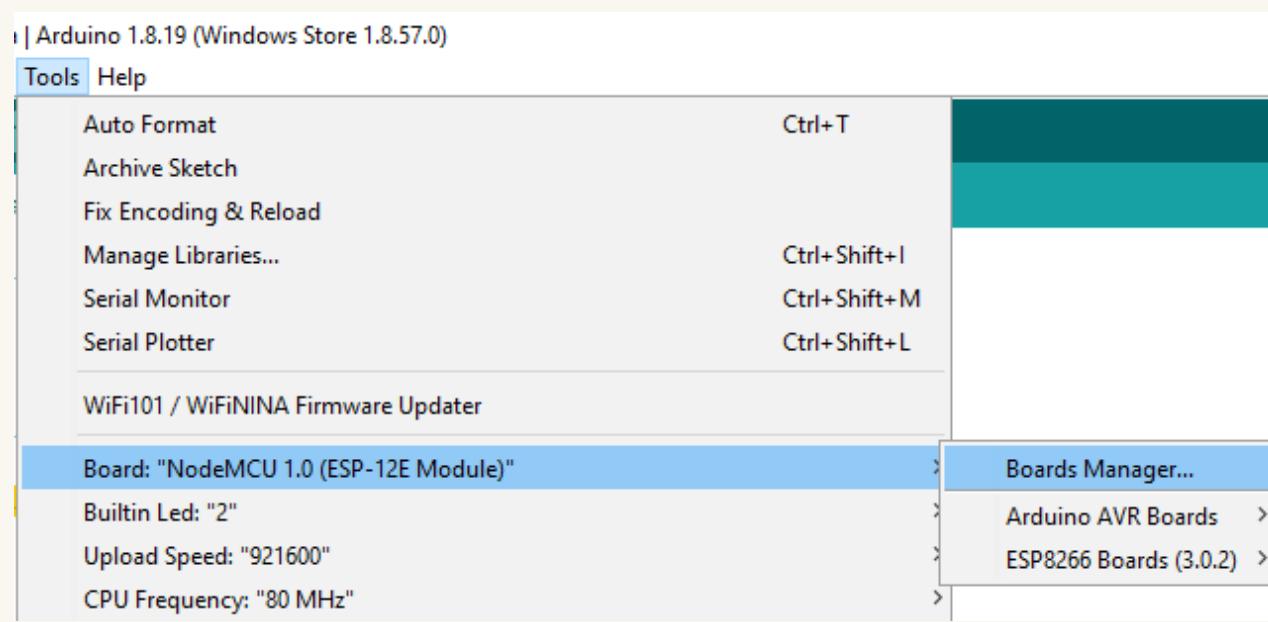
Ya aquí agregaremos la siguiente URL al apartado de Additional Boards Manager URLs, [http://arduino.esp8266.com/stable/package\\_esp8266com\\_index.json](http://arduino.esp8266.com/stable/package_esp8266com_index.json), que basicamente le indica donde buscar la board del nodeMCU



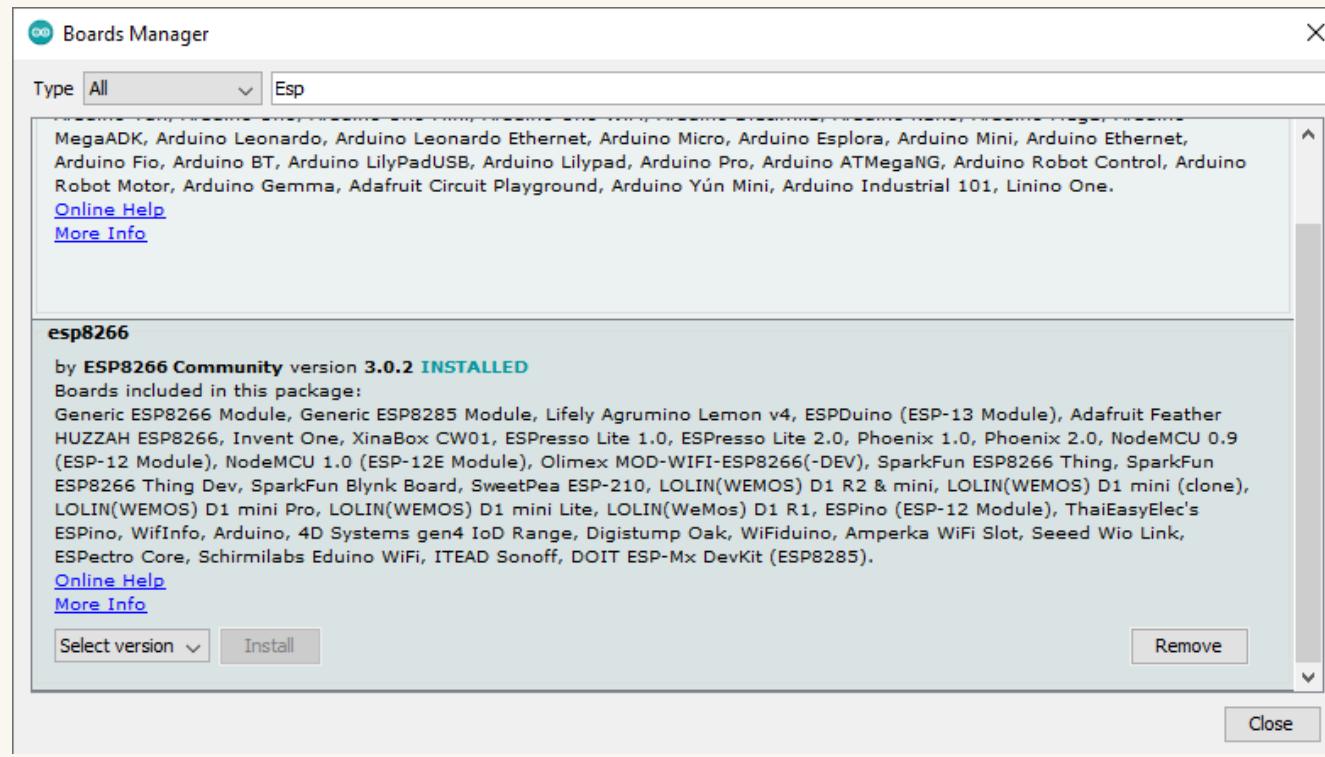
# Node MCU

## Configuracion del IDE de arduino

Ahora descargaremos la board desde el board manager, para llegar a este haremos  
Tools>Boards>Board manager



Descargamos la Tableta ESP8266

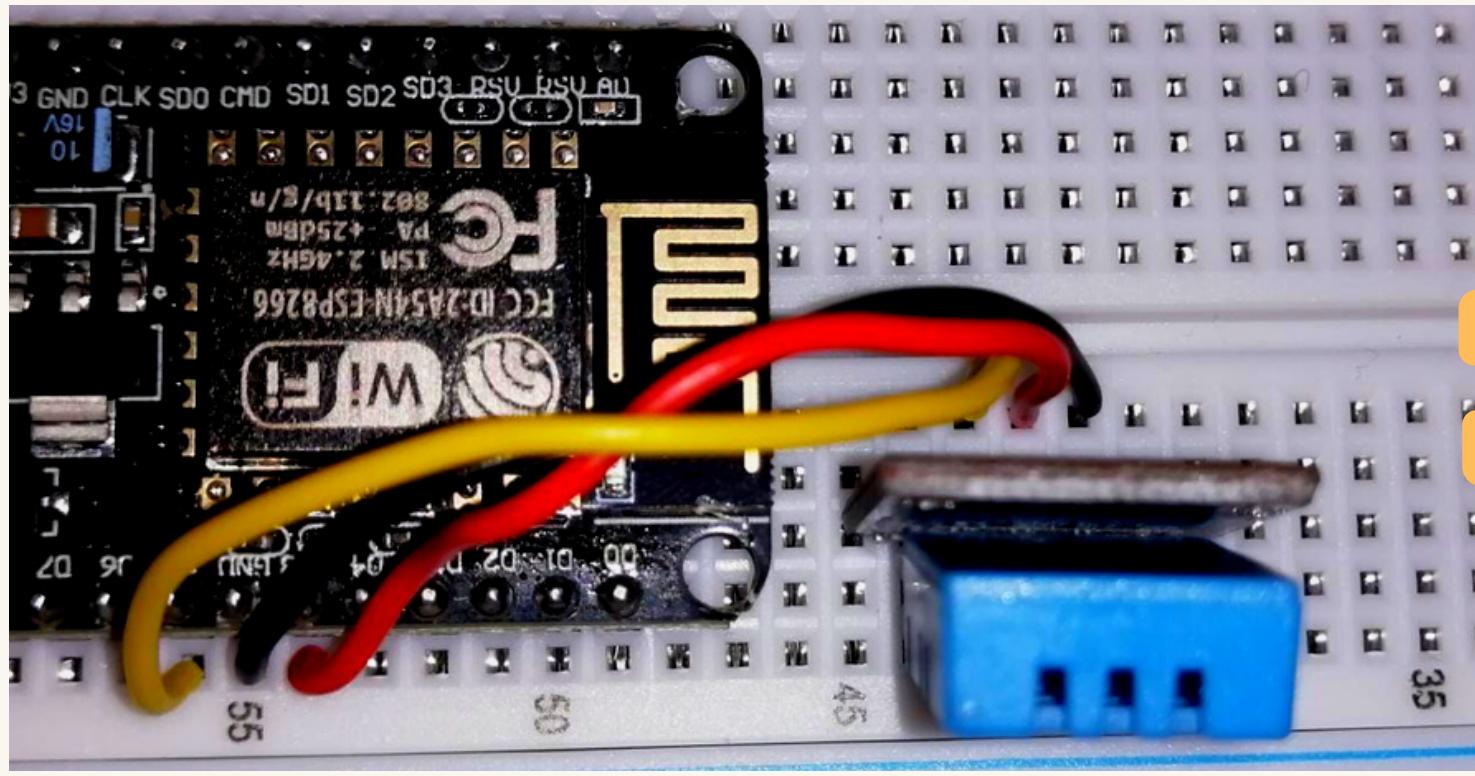


Listo, el nodeMCU esta listo al igual que le IDE de arduino.

# Node MCU

## Alambrado

El alambrado puede, si bien es una parte crucial del proyecto, esta lejos de ser la mas compleja, bastara con que sepas a que terminal de input estas conectando a la terminal de datos del sensor, lo demas bastara con que lo conectes a "galleta"



## Código del proyecto

El código relacionado al apartado del nodeMCU se divide en dos partes, Sensor y Servidor.

### Librerías y Variables públicas

```
#include <ESP8266WiFi.h>
#include <ESP8266HTTPClient.h>
#include "DHTesp.h"
```

Primero que nada incluimos las librerías correspondientes para trabajar con las funciones de internet y el HTTP client con el que podremos subir datos al servidor.

```
#define HOST "iot.dominiodemiranda.playit.gg:33396" //HOST URL
```

Le tendremos que brindar un HOST, este sera la URL que obtendremos posteriormente en el montaje del servidor.

```
#define WIFI_SSID "Tony"          // WIFI SSID
#define WIFI_PASSWORD "patoJarioso" // WIFI password here
```

Definiremos las variables para establecer una conexión wifi

te recomendamos hacer uso de un hotspot desde tu computadora.

```
#define DHTpin 14
```

Por último definimos el pin del sensor, el 14 es el 5D.

# Node MCU

## Código del proyecto

```
// Variables which will be uploaded to server
float val1 = 0;
float val2 = 0;
String sendval, sendval2, postData;

DHTesp dht; //Initialize DHT Sensor
```

Declaramos las variables que enviaremos al servidor.

Inicializamos el sensor como dht

Te recomendamos leer todo el manual una vez y regresar a esta parte, principalmente por que es probable te hayas equivocado en esto... o solo copia y pega :)

## Set Up

```
void setup(){
  Serial.begin(115200); // Information transfer rate from Arduino Code to NodeMCU
  Serial.println("Communication Started \n\n");
  delay(2000);

  dht.setup(DHTpin, DHTesp::DHT11); // GPIO14

  pinMode(LED_BUILTIN, OUTPUT); // Initialize NodeMCU LED

  WiFi.mode(WIFI_STA);
  WiFi.begin(WIFI_SSID, WIFI_PASSWORD); // Try to connect with WiFi

  Serial.print("Connecting to ");
  Serial.print(WIFI_SSID);

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

  Serial.println();
  Serial.print("Connected to: ");
  Serial.println(WIFI_SSID);
  Serial.print("IP Address is: ");
  Serial.println(WiFi.localIP()); // Print local IP address

  delay(500);
}
```

Confundido? no te apures estamos igual :)

En esta parte hacemos la inicialización de nuestro nodeMCU, por lo que este código solo se va a correr una vez igual que en arduino. Primero, establecemos la frecuencia de comunicación que manejará el nodeMCU, es casi siempre la misma así que no te apures por eso. Luego hacemos el set up de nuestro objeto dht, que inicializamos antes. Inicializamos el led que viene por defecto en el node. Por último nos tratamos de conectar a internet, saldrán puntitos en el monitor serial, si lo logramos saldrá un mensaje de que estamos conectados, la red a la que nos conectamos y la IP.

# Node MCU

## Código del proyecto

### Loop

```

void loop() {

    HTTPClient http;           // Http object of clas HTTPClient
    WiFiClient wclient;        // Wclient object of clas HTTPClient

    val1 = dht.getTemperature(); // Gets the values of the temperature
    val2 = dht.getHumidity();   // Gets the values of the humidity

    if(isnan(val1) || isnan(val2)){
        Serial.println("Error reading values from DHT Sensor");
        Serial.println("Sending 0's to Table");
        sendval = String(0);
        sendval2 = String(0);
    }
    else{
        // Convert float variables to string
        sendval = String(val1);
        sendval2 = String(val2);
    }
    delay(180000);
// We can post values to PHP files as example.com/dbwrite.php?name1=val1&name2=val2&name3=val3
    postData = "sendval=" + sendval + "&sendval2=" + sendval2;
    http.begin(wclient, "http://iot.dominiodemiranda.playit.gg:33396/dbwrite.php");
// Connect to host where dbwrite is located (PHP File to Write into SQL Database)
    http.addHeader("Content-Type", "application/x-www-form-urlencoded");
// Specify content-type header

    int httpCode = http.POST(postData);
// Send POST request to php file and store server response code in variable named httpCode
    Serial.println("Values sent were: Temperature = " + sendval + " && Humidity = "+sendval2 );

// If connection established successfully to database
    if (httpCode == 200){
        Serial.println("Values uploaded successfully.");
    }

// If failed to stablish then return and restart
    else {
        Serial.println(httpCode);
        Serial.println("Failed to upload values. \n");
        http.end();
        return;
    }

    delay(2000);
    digitalWrite(LED_BUILTIN, LOW);
    delay(2000);
    digitalWrite(LED_BUILTIN, HIGH);

//val+=1; val2+=10; // Incrementing value of variables
}

```

Aquí te voy san pedro :(

# Node MCU

## Código del proyecto

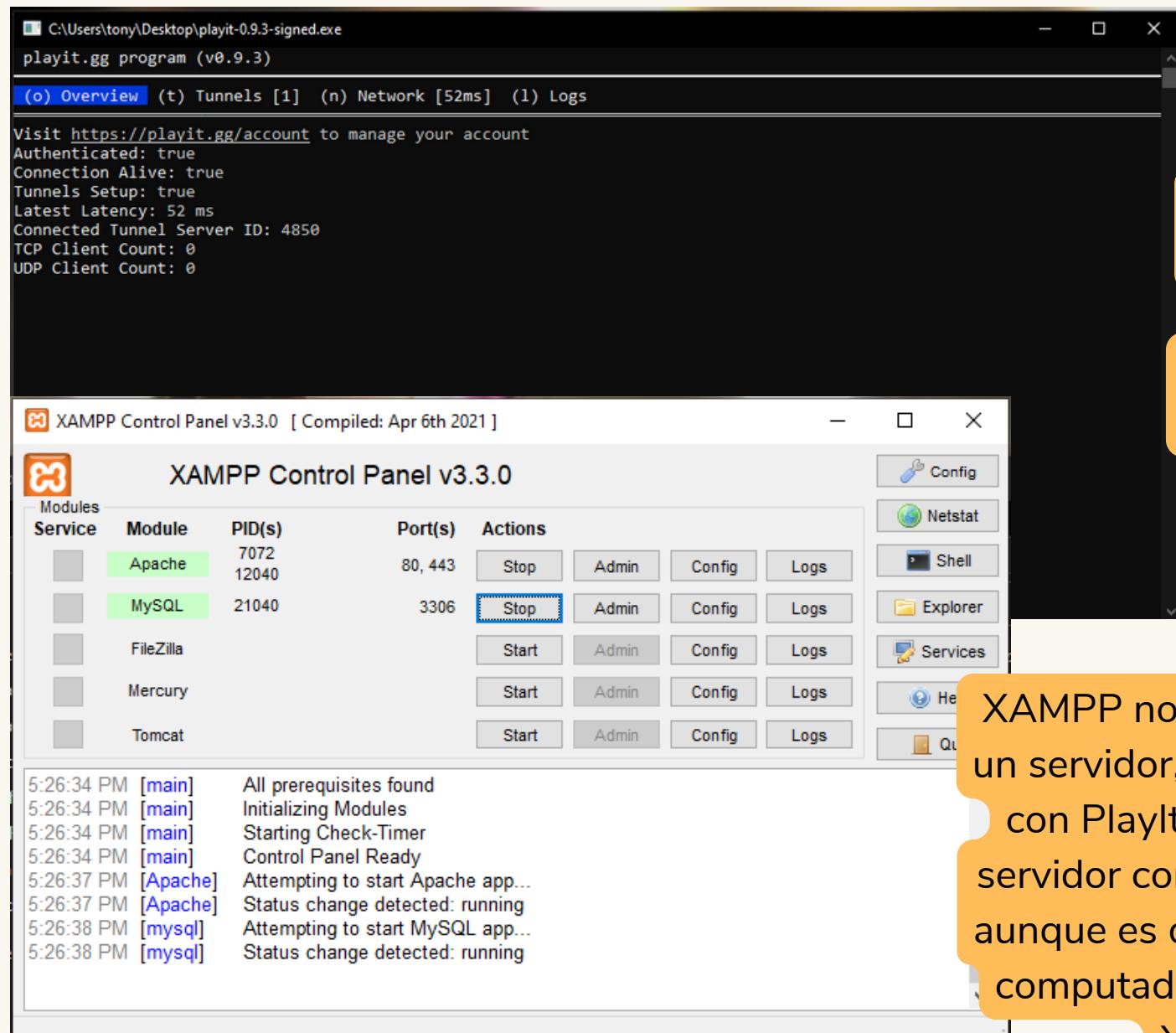
Todo el código de arriba es parte del Loop, todo lo que está ahí se repite infinitamente, o bueno mientras tengamos "galleta". Entonces lo primero que pasa es que hacemos dos objetos, HTTPClient y WiFiClient. Despues rescatamos los valores de temperatura y humedad en las variables Var1 y Var2. Si una de estas nos da un Nulo o Nan, nos mandara una alerta y se enviaran 0, de esta forma podemos saber si el sensor sufrio algun daño, sino envia los valores tal cual.

Esperamos 3 minutos antes de enviar valores a la pagina.

Armamos un paquete con los datos que vamos a mandar. Le decimos a que archivo de nuestro servidor lo enviara, y le manda un header. Por ultimo, le pedimos que nos diga si se logro subir la información a la base de datos, si lo logro nos mandara un mensaje de exito y volvera a iterar.

## Conexión al Servidor

Primero que nada, hay que conseguir un servidor, para esto haremos uso de playit.gg, un proyecto, aun en desarrollo que nos permite usar nuestra computadora como un servidor. Aparte de esto usaremos XAMP.



Playit.gg es gratuito sin embargo para este proyecto usamos una versión de pago para poder tener mejor control, en todo caso el precio estandar del servicio es de un dolar.

XAMPP nos permite emular un servidor, esto en conjunto con Playit.gg nos crea un servidor con dominio propio, aunque es dependiente de la computadora que controle XAMPP

# Node MCU

## Conexión al Servidor

Ahora, conectaremos la base de datos, ya creada en nuestro localhost con nuestro nodeMCU, para esto haremos uso de un archivo php, que ira en nuestra carpeta htdocs.

### dbwrite.php

```
<?php

$host = "localhost";
// Host: Is localhost because database hosted on the same place where PHP files are hosted
$dbname = "lefnica";           // Database name
$username = "tony";            // Database username
$password = "123";             // Database password

// Establish connection to MySQL database
$conn = new mysqli($host, $username, $password, $dbname);

// Check if connection established successfully
if ($conn->connect_error) {
    die("Connection failed: " . $conn->connect_error);
}

else {
    echo "Connected to mysql database. <br>";
}

// Get date and time variables
date_default_timezone_set('America/Mexico_City');
$d = date("Y-m-d");
$t = date("H:i:s");

// If values sent by NodeMCU code in Arduino are not empty then insert into MySQL database table

if(!empty($_POST['sendval']) && !empty($_POST['sendval2'])) ){
    $val = $_POST['sendval'];
    $val2 = $_POST['sendval2'];

    $sql = "INSERT INTO nodemcu(Temperatura,Humedad, Tiempo) VALUES ('".$val."','".$.$val2."','".$.$t."')";
    // nodemcu_table(val, val2, Date, Time)
    if ($conn->query($sql) === TRUE){
        echo "Values inserted in MySQL database table.";
    }

    else{
        echo "Error inserting values in MySQL: " . $sql . "<br>" . $conn->error;
    }
}

// Close MySQL connection
$conn->close();

?>
```

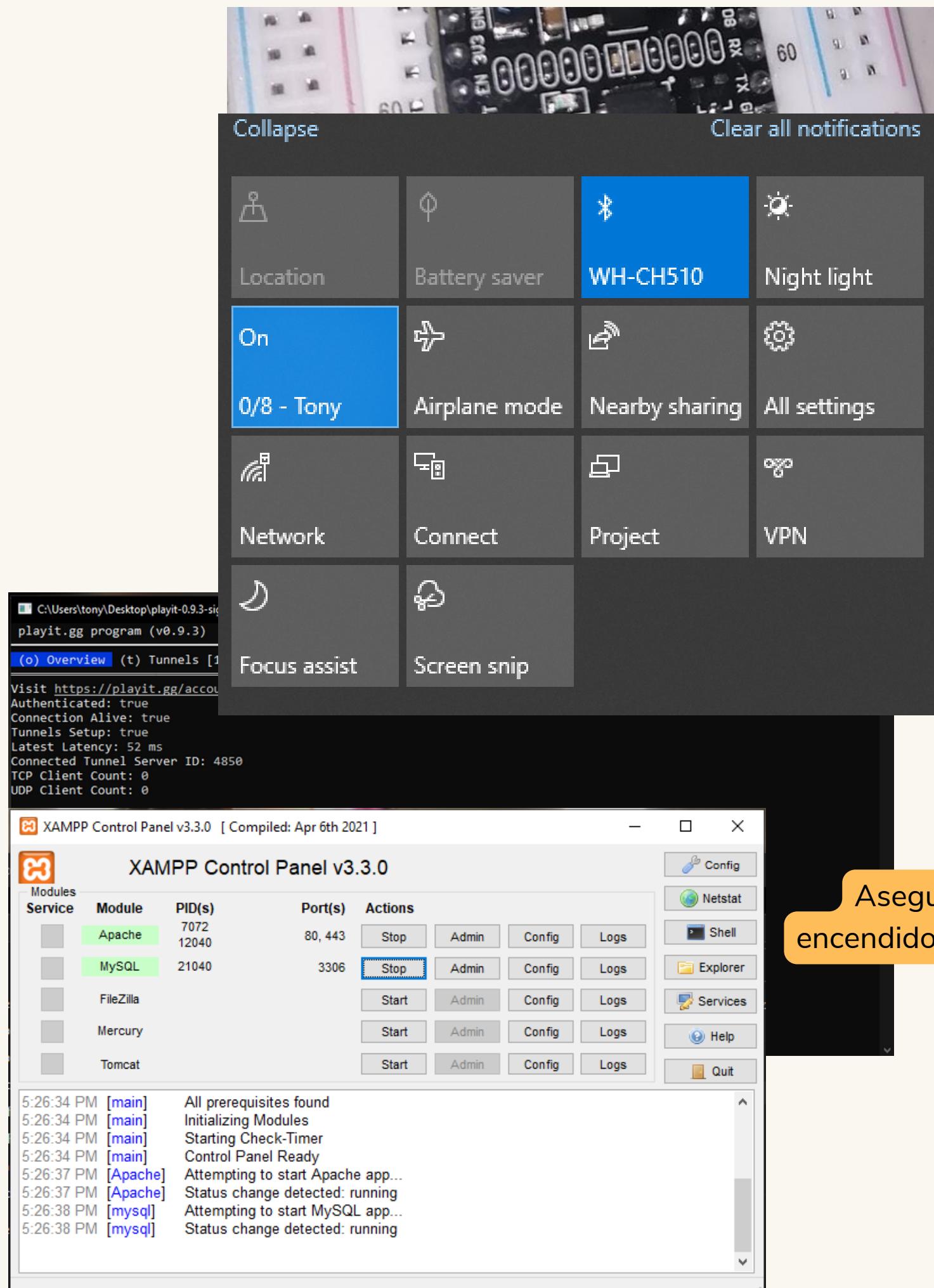
# Node MCU

## Conexión al Servidor

Esta código, inicia sesión en tu phpMyAdmin, SQL, y le hace un insert de los datos que rescatamos del nodeMCU, honestamente pasa mucho más, pero no es tema de este proyecto.

## Iniciando el proyecto

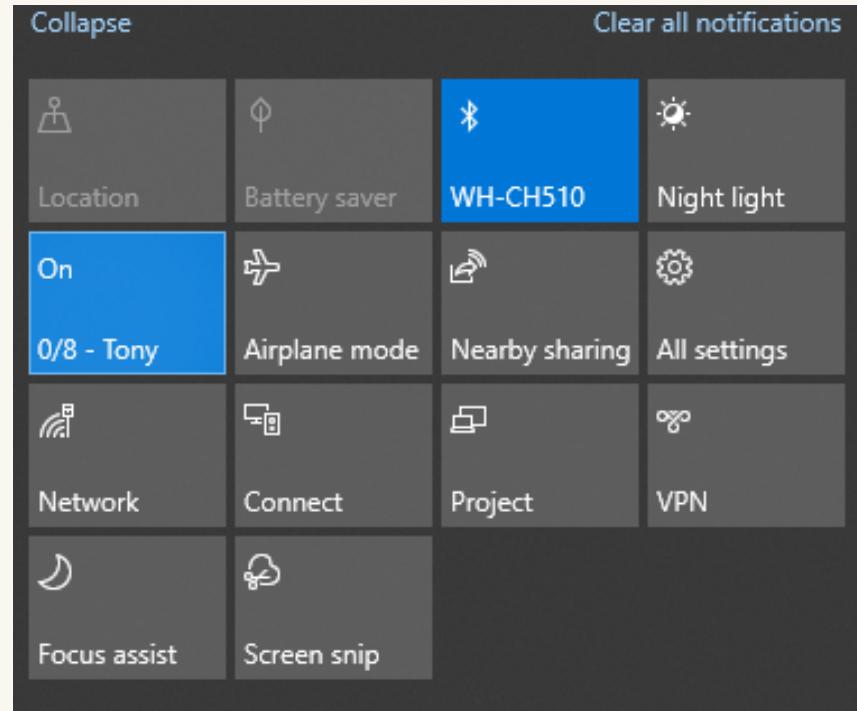
Conecta el node a voltaje



# Node MCU

## Iniciando el proyecto

Inicia el local Host



Abre el monitor serial o tu pagina para ver la magia

```
COM3
Connecting to Tony.....
Connected to: Tony
IP Address is: 192.168.1.37
Values sent were: Temperatura:23.000000000000002
Values uploaded successfully

----- CUT HERE FOR LOGS -----
Exception (9):
epc1=0x40212d08 epc2=0x00000000
>>>stack>>>
ctx: cont
sp: 3fffffc70 end: 3fffffc0
3ffffe00: 00000000 00000000
3ffffef0: 3fffeeb6c 00000000
3ffffe20: 0002f4eb 3fffeffd
3ffffe30: 0000001c7 00000000
3ffffe40: 0002f4eb 3fffffc0
3ffffe50: 00000012c 00000000
<----->
```

Autoscroll  Show timestamp

No te preocupes por los numeros raros, solo asegurate que se este enviando la informacion corretamente a la base de datos

**Tablero**

Escoger otro tipo de alimento

**Datos del alimento**

General

Tipo: VERDURAS

Nombre: rabiblos

Ambiente ideal

Temperatura: 13°C - 1°C

Humedad: 50% - 85%

**Descripción**

Aqui observarás un resumen gráfico y sencillo de los datos obtenidos con base en el tipo de alimento escogido previamente.

En la tarjeta "Datos del alimento" verás los datos ingresados; en "Gráficas" observarás las gráficas de comportamiento de la temperatura y humedad con respecto del tiempo; en "Estado" podrás ver si las condiciones actuales cumplen con los estándares para su conservación; finalmente, en "Reporte en vivo" verás el historial de datos registrados por el "LEF-MCU".

Para más información puedes dirigirte a la sección "Información" en la barra de navegación superior.

**Gráficas**

Temperatura vs Tiempo

Humedad vs Tiempo

**Estado**

Buenas tardes!

28 de noviembre de 2022

18:06

**Pruebas de cada sensor:**

Temperatura  Humedad

Tu alimento NO se encuentra en un ambiente óptimo para su conservación, te sugerimos cambiáro de lugar para asegurar su calidad.

**Reporte en vivo**

Tiempo	Temperatura	Humedad
14:47:11	23	66
14:43:56	23	66
14:40:42	23	66
14:37:28	23	66
14:34:14	23	66

Parte 3

# Página Web

lefnica.

# Página Web

## Contenido

- Pasos iniciales (21)
- Conexión con la base de datos (22)
- Gráficas (22-25)
- Tabla (25-28)

# El tablero

Para el proyecto de lefnica se representan los datos mediante un tablero (dashboard) en el que se presentan dos gráficas y una tabla de datos.

En el repositorio de GitHub brindado encontrarás los archivos necesarios para la página

## Pasos iniciales

Primero, se crea la estructura de la página con HTML y se importan los archivos externos de CSS para los estilos de los elementos.

Por motivos de espacio solo se muestra el elemento "head" de la página. A este archivo llamémosle "tablero.html"

```
<!DOCTYPE html>
<html lang="es">

<head>
  <meta charset="utf-8"/>
  <meta name="viewport" content="initial-scale=1, width=device-width"/>
  <title>Tablero</title>
  <meta name="description" content="">

  <link rel="icon" type="image/x-icon" href="lef_logo.png">

  <link rel="stylesheet" href="./global.css" />
  <link rel="stylesheet" href="./info-alimento.css"/>
  <link rel="stylesheet" href="./dashboard.css"/>
  <link rel="stylesheet" href="./team.css"/>

  <link rel="stylesheet"
    href="https://fonts.googleapis.com/css2?family=Montserrat:wght@400;500;600;700&display=swap"/>
  <link rel="stylesheet" href="https://fonts.googleapis.com/css2?family=Modak:wght@400&display=swap"/>
    <link      rel="stylesheet"      href="https://fonts.googleapis.com/css2?
family=Manrope:wght@400;600;700;800&display=swap"/>
  <link rel="stylesheet" href="https://fonts.googleapis.com/css2?family=Helvetica:wght@700&display=swap"/>
  <link rel="stylesheet" href="https://fonts.googleapis.com/css2?family=Inter:wght@400&display=swap"/>

    <link      rel="stylesheet"      href="https://fonts.googleapis.com/css2?
family=Material+Symbols+Sharp:opsz,wght,FILL,GRAD@20..48,100..700,0..1,-50..200"/>
    <link      rel="stylesheet"      href="https://fonts.googleapis.com/css2?
family=Material+Symbols+Outlined:opsz,wght,FILL,GRAD@48,400,0,0"/>
  <link href="https://fonts.googleapis.com/icon?family=Material+Icons+Outlined" rel="stylesheet"/>
</head>
```

# Conexión con la base de datos

Para conectar la página con la base de datos se requiere de PHP. El primer paso es cambiar la extensión del archivo "tablero.html" a PHP, quedando como "tablero.php".

Después se incluye lo siguiente arriba de "<!DOCTYPE html>". Fuera de todos los elementos de la página.

```
<?php
$host = "localhost";
$user = "root";
$password = "root";
$db = "lefnica";

$con = mysqli_connect($host, $user, $password, $db);

$sql = "SELECT * FROM `alimento`";
$all_categories = mysqli_query($con, $sql);

$id_selected = $_SESSION['id_select'];
$name_selected = $_SESSION['name_select'];

$sql_type = "SELECT * FROM `alimento` WHERE Id_alimento = '".$id_selected."'";
$result_type = mysqli_query($con,$sql_type);

$category_type = mysqli_fetch_array($result_type,MYSQLI_ASSOC);

$_SESSION['cat_type'] = $category_type;
?>
```

## Gráficas

Para las gráficas usaremos la librería CanvasJS, para importarla al código colocamos las siguientes líneas en el elemento "head".

```
<script src="https://ajax.googleapis.com/ajax/libs/jquery/3.6.0/jquery.min.js"></script>
<script type="text/javascript" src="https://canvasjs.com/assets/script/jquery-1.11.1.min.js"></script>
<script type="text/javascript" src="https://canvasjs.com/assets/script/canvasjs.min.js"></script>
```

Y agregamos las siguientes funciones de JavaScript en el mismo apartado "head".

```
<script type="text/javascript">

function addLoadEvent(func) {
    var oldonload = window.onload;
    if (typeof window.onload != 'function') {
        window.onload = func;
    } else {
        window.onload = function() {
            if (oldonload) {
                oldonload();
            }
            func();
        }
    }
}
```

```

function graph_temperatura() {
    var chart = new CanvasJS.Chart("t_chart",{
        axisY: {
            title: "Temperatura",
            titleFontFamily: "Tahoma",
        },
        axisX: {
            title: "Tiempo",
            titleFontFamily: "Tahoma",
        },
        data: [{{
            lineColor: "orange",
            color: "orange",
            type: "line",
            dataPoints : [],
        }}]
    });
}

$.getJSON("temperatura.php", function(data) {
    $.each((data), function(key, value){
        chart.options.data[0].dataPoints.push({label: value[0], y: parseInt(value[1])});
    });
    chart.render();
    updateChart();
});
}

function updateChart() {
    $.getJSON("temperatura.php", function(data) {
        chart.options.data[0].dataPoints = [];
        $.each((data), function(key, value){
            chart.options.data[0].dataPoints.push({label: value[0], y: parseInt(value[1])});
        });
        chart.render();
    });
}

setInterval(function(){updateChart()}, 1000);
}

function graph_humedad() {
    var chart = new CanvasJS.Chart("hum_chart",{
        axisY: {
            title: "Humedad",
            titleFontFamily: "Tahoma",
        },
        axisX: {
            title: "Tiempo",
            titleFontFamily: "Tahoma",
        },
        data: [{{
            lineColor: "orange",
            color: "orange",
            type: "line",
            dataPoints : [],
        }}]
    });
}

$.getJSON("humedad.php", function(data) {
    $.each((data), function(key, value){
        chart.options.data[0].dataPoints.push({label: value[0], y: parseInt(value[1])});
    });
    chart.render();
    updateChart();
});
}

function updateChart() {
    $.getJSON("humedad.php", function(data) {
        chart.options.data[0].dataPoints = [];
        $.each((data), function(key, value){
            chart.options.data[0].dataPoints.push({label: value[0], y: parseInt(value[1])});
        });
        chart.render();
    });
}

setInterval(function(){updateChart()}, 1000);
}

addLoadEvent(graph_temperatura);
addLoadEvent(graph_humedad);
</script>

```

En el elemento "body" se crearán dos div, uno para la gráfica de temperatura y otro para la de humedad. Y se les especificará tanto un id único a cada uno, como valores de ancho y alto específicos.

```
<div id="t_chart" style="height: 80%; width: 90%;"></div>
<div id="hum_chart" style="height: 80%; width: 90%;"></div>
```

Para obtener los datos de la base de datos se crean dos archivos con extensión ".php" uno para los datos de temperatura (temperatura.php) y otro para los datos de humedad (humedad.php). El archivo temperatura.php iría de la siguiente forma.

```
<?php

$username = "root";
$password = "root";
$dbname = "lefnica";

// Create connection
$conn = new mysqli("$username", $password, $dbname);
// Check connection
if ($conn->connect_error) {
    die("Connection failed: " . $conn->connect_error);
}

$sql = "SELECT Tiempo, Temperatura FROM nodemcu";
$result = $conn->query($sql);

if ($result->num_rows > 0) {
    // output data of each row
    $obj = array();
    while($row = $result->fetch_assoc()) {
        $element = array($row["Tiempo"], $row["Temperatura"]);
        array_push($obj, $element);
    }
    echo json_encode($obj);
} else {
    echo "0 results";
}
$conn->close();
?>
```

El archivo humedad.php es muy similar al de temperatura.php, solo se hacen uno ajustes menores para que el sistema filtre la columna de humedad en vez de la temperatura.

El código del archivo se muestra en la siguiente página.

```

<?php

$username = "root";
$password = "root";
$dbname = "lefnica";

// Create connection
$conn = new mysqli(",$username, $password, $dbname);
// Check connection
if ($conn->connect_error) {
    die("Connection failed: " . $conn->connect_error);
}

$sql = "SELECT Tiempo, Humedad FROM nodemcu";
$result = $conn->query($sql);

if ($result-> num_rows > 0) {
    // output data of each row
    $obj = array();
    while($row = $result->fetch_assoc()) {
        $element = array($row["Tiempo"],$row["Humedad"]);
        array_push($obj,$element);
    }
    echo json_encode($obj);
} else {
    echo "0 results";
}

$conn->close();
?>

```

Ahora las gráficas se desplegarán en los div destinados a ello y se actualizarán cada segundo. Para más información, la documentación de CanvasJS te dirá todo lo que tienes que saber para utilizarlo: <https://canvasjs.com/php-charts/>

## Tabla

Para tabular los datos se usará la librería DataTables, para importar dicha librería se incluyen las siguientes líneas al archivo "tablero.php" en el elemento "head".

```

<script type="text/javascript" src="https://cdn.datatables.net/1.13.1/js/jquery.dataTables.min.js"></script>
<link rel="stylesheet" href="https://cdn.datatables.net/1.13.1/css/jquery.dataTables.min.css"/>

```

Y agregamos las siguientes funciones en el mismo tag de <script> en el que se agregaron las gráficas.

```

function data_table(){
    var temp_len = 0;

    var table = $('#data_table').DataTable( {
        paging: false,
        searching: false,
        'responsive': true,
        order: [[0, 'desc']],
        columns: [
            { data: 'Tiempo' },
            { data: 'Temperatura' },
            { data: 'Humedad' }
        ],
    });
}

get_table();

function get_table(){

$.getJSON("size_db.php", function(data){

$('#tableh, #tableh1, #tableh2').css({
    "position": "sticky",
    "background-color" : "#dedede"
});

$.getJSON("table.php", function(max){

if(temp_len == data){
    return;
}
else{

    var last_time = max[temp_len]["Tiempo"];
    var last_temp = Math.floor(max[temp_len]["Temperatura"]);
    var last_hum = Math.floor(max[temp_len]["Humedad"]);

    change_state(last_temp,last_hum);

    table.row.add( {
        "Tiempo": last_time,
        "Temperatura": last_temp,
        "Humedad": last_hum,
    }).draw();

    $('#temp_actual').html(last_temp + "°C");
    $('#hum_actual').html(last_hum + "%");

    temp_len++;
}
});
});
});
setInterval(function(){get_table()}, 1000);
}

addLoadEvent(data_table);
}

```

Para obtener el número de filas de la base de datos, creamos un archivo llamado "size\_db.php" e ingresamos las siguientes líneas.

```
<?php

$host = "localhost";
$username = "root";
$password = "root";
$dbname = "lefnica";

// Create connection
$conn = new mysqli($host,$username, $password, $dbname);
// Check connection
if ($conn->connect_error) {
    die("Connection failed: " . $conn->connect_error);
}

$sql = "SELECT * FROM nodemcu;";
$result = $conn->query($sql);

$rows = mysqli_num_rows ( $result );

echo ($rows);

$conn->close();
?>
```

Para obtener los datos de temperatura, humedad y tiempo, en un arreglo para la tabla, creamos otro archivo PHP llamado "table.php" e ingresamos lo siguiente:

```
<?php

$host = "localhost";
$username = "root";
$password = "root";
$dbname = "lefnica";

// Create connection
$conn = new mysqli($host,$username, $password, $dbname);
// Check connection
if ($conn->connect_error) {
    die("Connection failed: " . $conn->connect_error);
}

$sql = "SELECT * FROM `nodemcu`";
$result = $conn->query($sql);

if ($result-> num_rows > 0) {
    // output data of each row
    $obj = array();
    // $row = $result->fetch_assoc();
    // $element = array('Tiempo' => $row["Tiempo"], 'Temperatura' => $row["Temperatura"], 'Humedad' => $row["Humedad"]);
    // array_push($obj,$element);
    while($row = $result->fetch_assoc()) {
        $element = array('Tiempo' => $row["Tiempo"], 'Temperatura' => $row["Temperatura"], 'Humedad' => $row["Humedad"]);
        array_push($obj,$element);
    }
    echo json_encode($obj);
} else {
    echo "0 results";
}

$conn->close();
?>
```

De vuelta en el archivo "tablero.php" creamos una tabla dentro del elemento "body", con el id "data\_table" de la siguiente forma:

```
<table class="report_table" id="data_table">
    <thead id="tab_head">
        <tr>
            <th id="tableh">Tiempo</th>
            <th id="tableh1">Temperatura</th>
            <th id="tableh2">Humedad</th>
        </tr>
    </thead>
</table>
```

Gracias a DataTables, nuestra tabla podrá ser interactiva y los datos necesarios serán cargados en la tabla destinada. Para conocer más de DataTables, puedes leer la documentación en el siguiente enlace:  
<https://datatables.net/manual/>

En el repositorio de GitHub encontrarás todos los archivos mencionados y presentados en esta sección. De igual forma se anexan los archivos gráficos y CSS necesarios:

Archivos necesarios para esta parte: public (carpeta), dashboard.css, global.css, humedad.php, info-alimento.css, range.php, size\_db.php, table.php, tablero.php, team.css, temperatura.php.

<https://github.com/DavidLangarica/lefnica.git>

Tip: Asegurate que todos los archivos de la página estén en una carpeta (ej. lefnica) en la carpeta "htdocs" de MAMP o XAMPP.

Enlaces de interés y consulta:

- <https://www.youtube.com/watch?v=LYQe8QdrQvo&list=PLc1g3vwxhg1Xr2r2VodhuthKo0GClc7V8&index=1>
- <https://www.youtube.com/watch?v=jKbjblt4NXA&list=PLTE0X123tz-yxigJHnmuEgqWeVMI7YI-W>
- <https://code.tutsplus.com/es/tutorials/how-to-use-ajax-in-php-and-jquery--cms-32494>



# lefnica.

Enlace al repositorio:

<https://github.com/DavidLangarica/lefnica.git>

Los padrinos mágicos © 2022