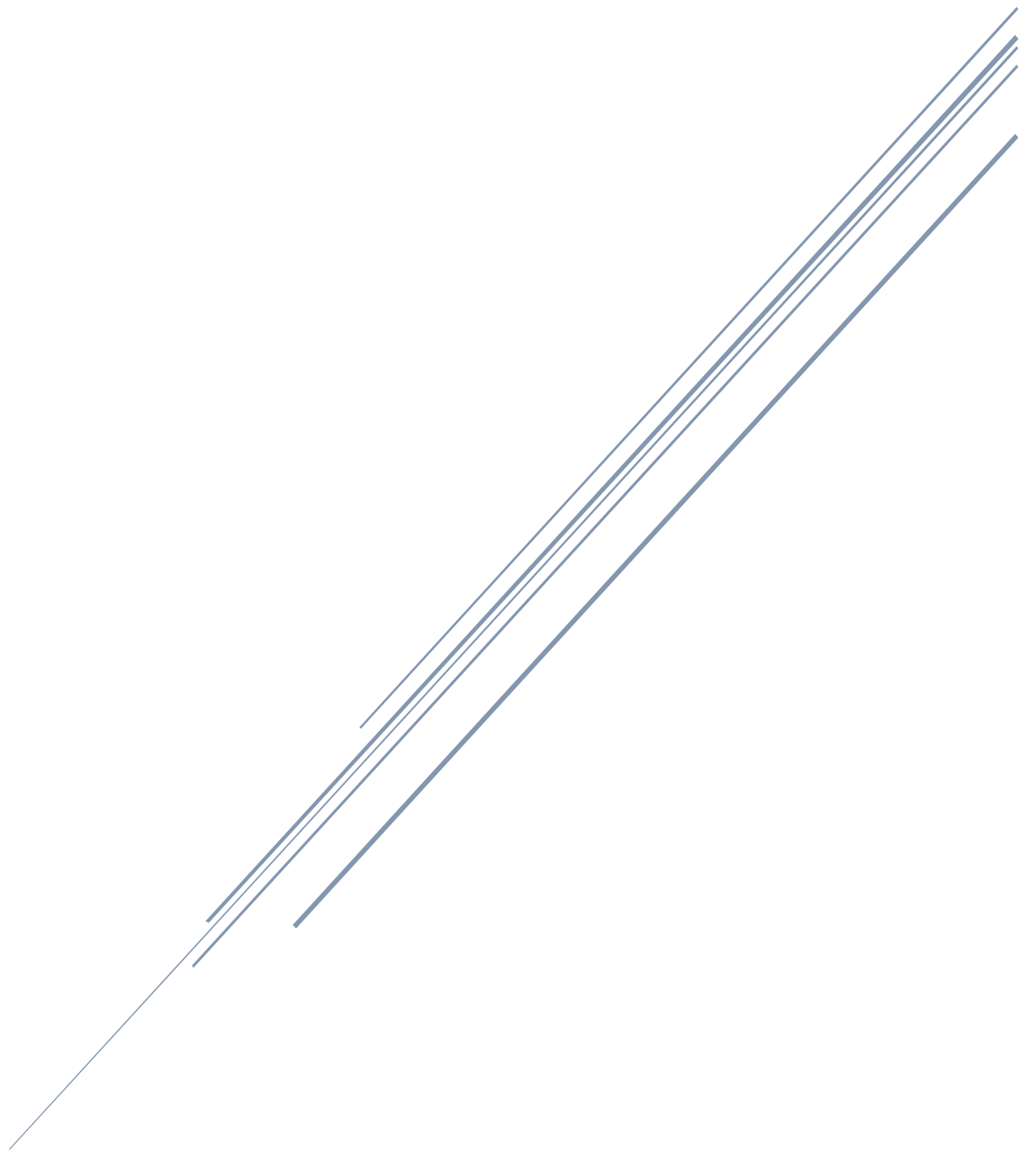


PROYECTO FINAL: SISTEMA DE CONTROL DE SENSORES MEDIANTE RASPBERRY PI Y ARDUINO



Contenido

1.	INTRODUCCIÓN	3
2.	RESUMEN.....	3
2.1.	ABSTRACT.....	3
3.	TECNOLOGÍAS UTILIZADAS	3
3.1.	Raspberry Pi 3B	3
3.2.	Node MCU V3.0	4
3.3.	Arduino Mega 2560	5
3.4.	Sensores y actuadores	6
3.4.1.	Sensor PIR HC-SR501	6
3.4.2.	Sensor de medida de nivel de agua.....	6
3.4.3.	Sensor de llama KY-026	7
3.4.4.	Sensor LDR KY-018.....	7
3.4.5.	Relé KY-019	8
3.4.6.	Buzzer activo KY-012	8
3.4.7.	Sensor de humedad y temperatura DHT11.....	9
3.4.8.	Resistencias, LEDs y cables Dupont	9
3.5.	Software utilizado	10
3.5.1.	Servidor Apache	10
3.5.2.	MySQL/MariaDB	10
3.5.3.	phpMyAdmin	11
3.5.4.	No-IP (Servicio DDNS)	11
3.5.5.	CoolTerm.....	12
3.5.6.	PHP y HTML	12
3.5.7.	VNC Viewer	12
3.5.8.	FileZilla	13
4.	FUNCIONAMIENTO	13
4.1.	Control de sensores y actuadores	13
4.2.	Envío de datos	14
4.3.	Visionado de datos	16
5.	CONCLUSIÓN	18
6.	MI PROYECTO TIENE FUTURO	18
7.	PRESUPUESTO.....	19

8. PROGRAMACIÓN	19
8.1. Código Arduino Mega	19
8.2. Código NodeMCU	22
8.3. Código PHP conexion.php	25
8.4. Código PHP EspPost.php.....	25

1. INTRODUCCIÓN

El proyecto consiste en la realización de un sistema de control de sensores mediante Raspberry Pi y Arduino. A su vez se ha creado un servidor web en el que se almacenan los datos de los sensores.

2. RESUMEN

En este proyecto se ha buscado desarrollar un sistema de control de sensores a través de internet. Para ello se ha creado una base de datos donde se almacenan los datos de estos sensores.

Para el desarrollo del proyecto se han utilizado 3 elementos principales: Raspberry Pi, Arduino y Node MCU. La Raspberry Pi será la encargada de alojar el sitio web donde se almacenan estos datos. A su vez, la Raspberry Pi está conectada al Arduino mediante el puerto serie. El Arduino se encarga de controlar los sensores y enviar los datos a la Raspberry Pi. Finalmente, la placa Node MCU enviará los datos de los sensores a la base de datos alojada en el servidor Apache creado.

El objetivo es poder visualizar el estado de los sensores en tiempo real a través de Internet.

2.1. ABSTRACT

This Project aims to develop a sensor control system via Internet. For this purpose, a database has been created to store the data.

For the Project development, three principal elements has been used: Raspberry Pi, Arduino and Node MCU. The Raspberry Pi is responsable to host the website, also the Raspberry Pi have the Arduino connected by serial port. The Arduino handles the sensor control and send the data to the Raspberry Pi. To sum up, the Node MCU send the data to a database hosted in the Apache server that was created.

The main objective is to view the sensor state via Internet at real time.

3. TECNOLOGÍAS UTILIZADAS

Para el desarrollo del proyecto se ha utilizado distintos elementos, tanto software como hardware, que se desarrollarán a continuación.

3.1. Raspberry Pi 3B

La Raspberry Pi es un ordenador de placa reducida, (SBC) de bajo costo desarrollado en el Reino Unido por la Raspberry Pi Foundation.



Posee distintos puertos y conexiones: conectores HDMI, Ethernet, USB conexión WiFi, Bluetooth, pines GPIO. A su vez, permite su uso como PC mediante un sistema operativo e interfaz gráfica. Para ello, los desarrolladores de esta placa han creado un sistema operativo específico para este ordenador llamado Raspberry Pi OS, basado en Debian, aunque se pueden utilizar otros sistemas operativos.

En la Raspberry se va a crear un servidor LAMP (Linux-Apache-MySQL-PHP), que se encargará de almacenar los datos recibidos por el Arduino y el Node MCU.

3.2. Node MCU V3.0

Node MCU es una placa de desarrollo abierto, tanto en software como hardware. Posee un microcontrolador para facilitar la programación de este y a su vez se compone de un SOC (System on Chip) ESP 8266.

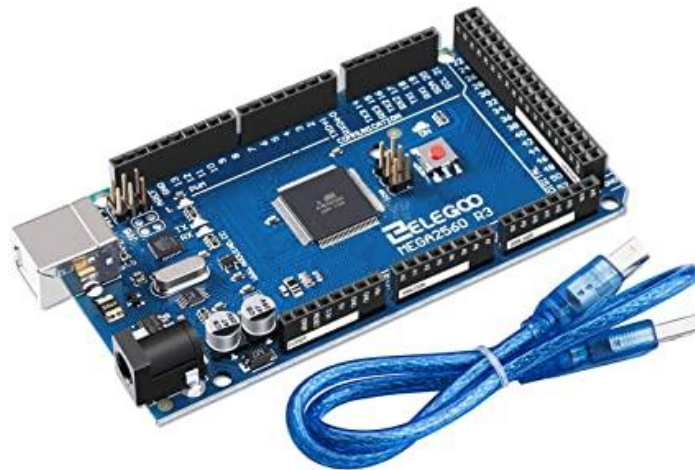


La programación de esta placa se realiza desde el IDE de Arduino permitiendo así programarla de la misma forma que un Arduino. La conexión al ordenador se realiza mediante un cable micro USB que permite también la alimentación del Node MCU.

Existen diferentes versiones de esta placa. En este caso se ha utilizado la Node MCU V3.0 basada en el módulo ESP8266-12E. En lugar de esta placa, se podría haber utilizado un chip ESP8266 conectado a un Arduino, lo que sería similar, pero la facilidad de utilización de esta placa, además de los pines que posee permite tener ambos elementos en conjunto.

3.3. Arduino Mega 2560

Arduino es una plataforma de creación de electrónica de código abierto, la cual está basada en hardware y software libre, flexible y fácil de utilizar para los desarrolladores. Esta plataforma permite crear diferentes tipos de microordenadores de una sola placa a los que la comunidad de creadores puede darles diferentes tipos de uso.



Realización del proyecto se ha utilizado la placa Arduino Mega 2560 debido a la gran flexibilidad que aporta, ya que posee un gran número de pines tanto de datos como de transmisión, recepción y alimentación.

Esta placa está basada en el microcontrolador ATmega 2560. Posee 54 pines digitales (15 de ellos PWM), 16 entradas analógicas, 4 UARTs (Transmisor-Receptor Asíncrono Universal), un oscilador de cristal de 16 MHz, una conexión USB, conexión de alimentación y un botón de RESET.

El Arduino Mega 2560 se encargará de controlar los sensores y actuadores utilizados, y enviara por el puerto serie los datos a la Raspberry Pi donde se subirán al servidor LAMP.

3.4. Sensores y actuadores

A continuación, se desarrollarán los elementos encargados tomar los datos y ejecutar las instrucciones deseadas. Estos sensores se conectan al Arduino y al Node MCU.

3.4.1. Sensor PIR HC-SR501



Este sensor de movimiento utiliza la tecnología infrarroja para detectar movimiento. Posee tres pines (alimentación, salida de datos y tierra) y dos potenciómetros con los que regular el rango de actuación del PIR (de 3 a 7 metros) y el tiempo que permanece la salida en HIGH cuando detecta movimiento (de 5 segundos a 5 minutos).

Su funcionamiento se basa en detectar cambios infrarrojos cuando se produce un movimiento. Dependiendo de la configuración de sensibilidad establecida, detectará un cambio infrarrojo como movimiento o no lo detectará.

Este dispositivo requiere de un minuto para inicializarse al ser alimentado. Durante este proceso es posible que este sensor emita señales de detección falsas. El dispositivo puede detectar movimiento dentro de un rango de 110° a una distancia de entre 3 y 7 metros.

En el desarrollo de este proyecto este sensor activará una alarma cuando detecte movimiento.

3.4.2. Sensor de medida de nivel de agua

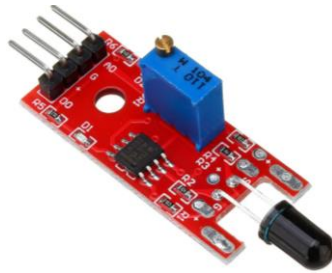


Este sensor permite medir la profundidad del agua (máximo 5 cm) re mediante un circuito amplificador formado por un transistor y varias rutas de PCB. Cuando está

puesto en el agua como estas rutas presentan una resistencia que cambia cuanto mayor sea la profundidad del agua. Luego, la señal de profundidad del agua es convertida en señal eléctrica y podemos conocer la profundidad debido al conversor analógico-digital de la placa Arduino Mega.

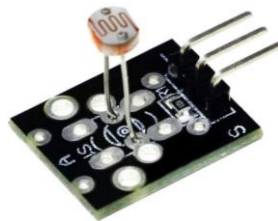
El voltaje de funcionamiento es de 5 voltios y posee una salida analógica. Los valores recibidos por este sensor serán enviados a la Raspberry Pi mediante el puerto serie.

3.4.3. Sensor de llama KY-026



El módulo KY-026 es un sensor de llama que por medio de un LED receptor infrarrojo detecta longitudes de onda de llama en un rango de 760nm a 1100nm y en un máximo de 60°. Las salidas de este sensor son digital y analógica e incluye un potenciómetro para ajuste de la sensibilidad del sensor. Puede trabajar tanto a 3,3V como a 5V. Los valores recibidos por este sensor serán enviados a la Raspberry Pi mediante el puerto serie.

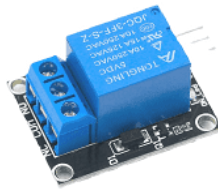
3.4.4. Sensor LDR KY-018



El sensor de luz LDR KY-018 es una resistencia variable con la luz que se puede conectar a Arduino de forma sencilla. El sensor proporciona una señal analógica que varía con la cantidad de luz recibida y puede utilizarse para medir luz ambiente (interior) o artificial con una linterna y también luz infrarroja o ultravioleta.

En la oscuridad la resistencia de la LDR puede ser muy elevada (aprox. de 1MΩ), mientras que puede llegar a caer dramáticamente con la presencia de luz (dependiendo de la intensidad de luz). El módulo incorpora una resistencia para adaptar la señal a Arduino a modo de divisor de tensión. La tensión de trabajo es de 5V.

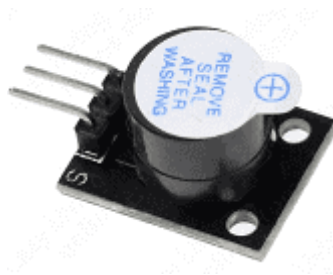
3.4.5. Relé KY-019



El Módulo Relé 5V KY-019 es un dispositivo electrónico que permite controlar dispositivos que trabajen máximo 250V /15A y 125V/10A AC y 30V/10A DC por medio de una señal de control de 5V, señalizando su activación por medio de un LED indicador en su PCB. Es utilizado en proyectos de IoT para controlar luces, contactos, electrodomésticos y otros dispositivos electrónicos. Comúnmente son usados en circuitos de control automático con una pequeña corriente de control y una gran corriente de operación.

Posee tres pines (5V, tierra y datos). En el desarrollo del proyecto se ha utilizado como actuador; cuando el sensor de llama detecta fuego, el relé se enciende, al cual se podría conectar, por ejemplo, una electroválvula encargada de permitir o no el paso de agua en un sistema antincendios.

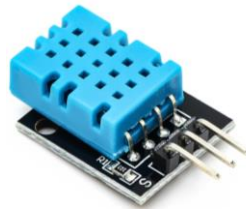
3.4.6. Buzzer activo KY-012



El Buzzer activo KY-012 permite reproducir un sonido de un solo tono (a diferencia de un buzzer pasivo). Incorpora un oscilador simple por lo que únicamente es necesario suministrar corriente al dispositivo para que emita sonido.

Posee tres pines: alimentación, señal y conexión a tierra. En el proyecto se ha utilizado como una alarma que emitirá un tono cuando el PIR detecte movimiento o cuando el sensor de llama detecte fuego.

3.4.7. Sensor de humedad y temperatura DHT11



El DHT11 es un sensor digital de temperatura y humedad relativa de bajo costo y fácil uso. Integra un sensor capacitivo de humedad y un termistor para medir el aire circundante, y muestra los datos mediante una señal digital en el pin de datos (no posee salida analógica).

Cuenta con tres pines: alimentación 3-5V, datos y conexión a tierra. Los datos obtenidos por este sensor son enviados a la base de datos alojada en el servidor LAMP.

3.4.8. Resistencias, LEDs y cables Dupont

Un led es un diodo emisor de luz de unión p-n. Si se aplica una tensión adecuada a los terminales, los electrones se recombinan con los huecos en la región de la unión p-n del dispositivo, liberando energía en forma de fotones. Este efecto se denomina electroluminiscencia, y el color de la luz generada viene determinado por la anchura de la banda prohibida del semiconductor.



Para el correcto funcionamiento de los leds se han utilizado resistencias de 100Ω. El objetivo de los leds es interpretar de una forma visible el estado de los sensores: se encenderán cuando el LDR no detecte suficiente luz y cuando el sensor de llama detecte fuego.



Todas las conexiones se han realizado mediante cables Dupont.



3.5. Software utilizado

Para la realización del proyecto se han utilizado diferentes programas que han permitido la correcta programación de las placas y la creación del servidor web.

3.5.1. Servidor Apache



El servidor HTTP Apache es un servidor web HTTP de código abierto, para plataformas Unix, Microsoft Windows, Macintosh y otras, que implementa el protocolo HTTP/1.12 y la noción de sitio virtual.

Nos permitirá convertir nuestra Raspberry en un servidor, que pueda responder a peticiones HTTP y que albergará la página web con la que interactuaremos con el sistema.

3.5.2. MySQL/MariaDB



MariaDB es un sistema de gestión de bases de datos derivado de MySQL con licencia GPL (General Public License). Es desarrollado por el fundador de MySQL, la fundación MariaDB y la comunidad de desarrolladores de software libre. Tiene una alta compatibilidad con MySQL ya que posee las mismas órdenes, interfaces, API y bibliotecas, siendo su objetivo poder cambiar un servidor por otro directamente.

3.5.3. phpMyAdmin



phpMyAdmin es una herramienta escrita en PHP con la intención de manejar la administración de MySQL a través de páginas web, utilizando un navegador web. Actualmente puede crear y eliminar bases de datos, crear, eliminar y alterar tablas, borrar, editar y añadir campos, ejecutar cualquier sentencia SQL, administrar claves en campos, administrar privilegios, exportar datos en varios formatos y está disponible en 72 idiomas.

3.5.4. No-IP (Servicio DDNS)



No-IP es una compañía que ofrece un servicio de DNS dinámica para servicios de pago y gratuitos. La función más usada del sistema de nombres de dominio (DNS) es la de relacionar una dirección IP a un nombre, por ejemplo, 87.67.124.56 al nombre de www.amazon.es.

El objetivo de este servicio es permitir el acceso al servidor desde fuera de la red local, ya que inicialmente, al crear el servidor, únicamente se puede acceder a el estando en la misma red local, ya que se accede mediante la dirección IP del dispositivo que aloja el servidor web (192.168.1.42).

Para conseguir acceso desde Internet, en primer lugar, se debe asociar un nombre de dominio a la dirección IP pública del router (84.56.78.21). Posteriormente, se debe realizar un redireccionamiento de puertos en la configuración del router redireccionando las peticiones que recibe el router a la dirección IP de la Raspberry Pi.

3.5.5. CoolTerm

CoolTerm es una aplicación simple de terminal de puerto serie (sin emulación de terminal) que está dirigida a aficionados y profesionales que necesitan intercambiar datos con hardware conectado a puertos serie como servocontroladores, kits robóticos, receptores GPS, microcontroladores, etc.

Se ha utilizado para enviar los datos que recibe la Raspberry por el puerto serie a un archivo de texto alojado en el servidor web.

3.5.6 PHP y HTML

Para conseguir una conexión correcta entre los dispositivos principales y el servidor, se han utilizado diferentes lenguajes de programación:

- **PHP:** es un lenguaje de código abierto muy popular especialmente adecuado para el desarrollo web y que puede ser incrustado en HTML. PHP está centrado en la programación de scripts del lado del servidor. Se ha utilizado para el envío de datos del Node MCU a la base de datos.
- **HTML:** HTML (Lenguaje de Marcas de Hipertexto, del inglés (HyperText Markup Language) es el componente más básico de la Web. Define el significado y la estructura del contenido web. Además de HTML, generalmente se utilizan otras tecnologías para describir la apariencia/presentación de una página web (CSS) o la funcionalidad/comportamiento (JavaScript). En el desarrollo del proyecto se ha utilizado para realizar las peticiones y consultas desde el Arduino al servidor web.

3.5.7. VNC Viewer



VNC es un programa que permite tomar el control del ordenador servidor remotamente a través de un cliente multiplataforma. Una vez instalado VNC en el ordenador, es posible acceder a él desde cualquier parte del mundo a través de Internet y desde cualquier dispositivo, como ordenadores o smartphones.

Se ha utilizado para obtener acceso remoto a la Raspberry Pi mediante un ordenador portátil.

3.5.8. FileZilla



FileZilla es un cliente FTP que nos permitirá conectar mediante este protocolo (File Transfer Protocol) con nuestro servidor FTP.

De este modo, podemos subir, descargar o modificar archivos de nuestro alojamiento de forma remota.

Además, las operaciones realizadas desde un cliente FTP no tienen las limitaciones que puede presentar un cliente online como WebFTP, por lo que se puede operar con ficheros pesados o con números altos de ficheros sin problemas.

4. FUNCIONAMIENTO

El proyecto se basa en tres partes principales: control de sensores y actuadores, envío de datos, y almacenamiento y visionado de datos.

4.1. Control de sensores y actuadores

Los encargados de controlar los sensores y actuadores son la placa Arduino Mega y la placa NodeMCU. Para ello, se han creado dos programas (uno por cada placa). El NodeMCU se encargará de leer el sensor DHT11 y enviar los datos a la base de datos, mientras que el Arduino Mega controlará todos los demás elementos, y enviará los datos al puerto serie, que serán recibidos por la Raspberry Pi.

El funcionamiento se basa en:

- **Sensor LDR:** mide el nivel de luz presente; en caso de que este sea menor al establecido; activará un led azul simulando una lámpara.
- **Sensor de llama:** cuando el sensor de llama detecte fuego, este enviará una señal al relé para que sea activado, lo que simularía un sistema antiincendios. Además, activa la alarma y un led de aviso.
- **Sensor de profundidad de agua:** este sensor mide la profundidad del agua presente, a través de una entrada analógica y nos permite visualizar el nivel de profundidad en el puerto serie. Esto simularía un sensor que midiera la cantidad de agua en una piscina, aunque también se podría utilizar como sensor de lluvia o como un detector de inundaciones.

- **Sensor PIR:** cuando detecta movimiento, activa la alarma. Al detectar movimiento envía el dato a través del puerto serie a la Raspberry Pi.
- **Buzzer activo:** este actuador simula la alarma de una vivienda, ya que emite un tono al detectar fuego y al detectar movimiento.
- **LEDs:** en el caso del led azul, simula una lámpara que se enciende al hacerse de noche o cuando hay poca luz. El led rojo se encenderá cuando se active la alarma, y se apagará cuando esta se desactive.

4.2. Envío de datos

Para el envío de datos entre las placas y la Raspberry Pi, ha sido necesario utilizar tanto el puerto serie como un programa en php.

- **Envío de datos desde Arduino a Raspberry Pi:** los datos que “imprimimos” en el puerto serie desde el Arduino, son enviados mediante USB a la Raspberry Pi. Para recibir estos datos, se a utilizado un programa llamado CoolTerm, que es un programa de terminal del puerto serie.

```

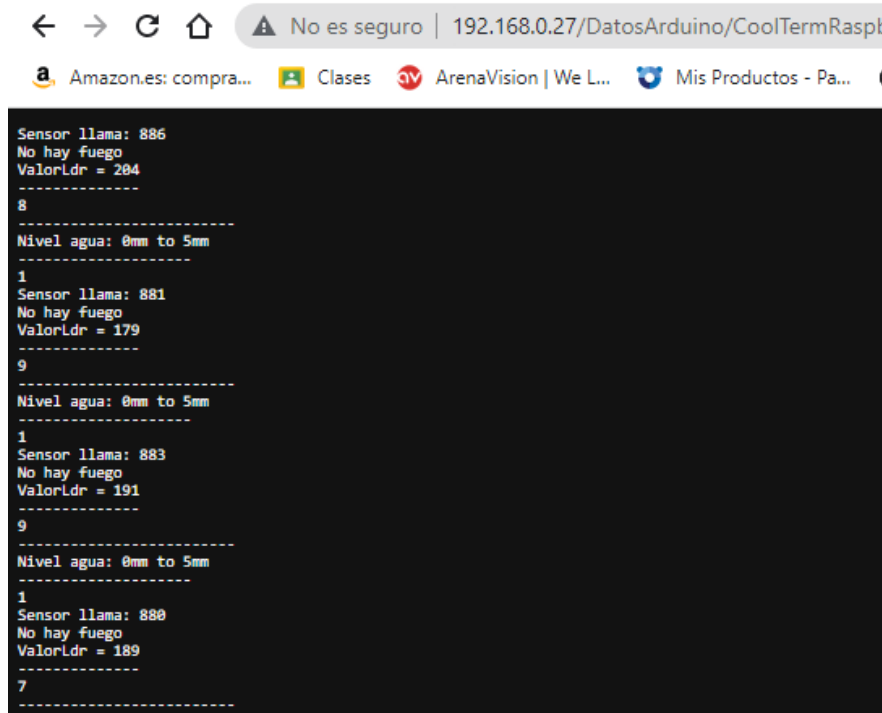
/dev/ttyACM0
No hay fuego
ValorLdr = 532
-----
13
-----
Nivel agua: 0mm to 5mm
-----
1
Sensor llama: 1017
No hay fuego
ValorLdr = 528
-----
14
-----
Nivel agua: 0mm to 5mm
☒ Autoscroll ☐ Mostrar marca temporal

```

Una vez estos datos son recibidos, se envían a un archivo de texto alojado en la carpeta “DatosArduino”, a la cual podemos acceder escribiendo 192.168.0.27/DatosArduino en el navegador.



Una vez dentro, si abrimos la carpeta podemos ver el archivo de texto creado por el programa y a través de él ver los datos del Arduino:



The screenshot shows a web browser window with the address bar displaying "192.168.0.27/DatosArduino/CoolTermRaspk". The browser's address bar also shows a warning "No es seguro" (Not safe). The main content area displays a text file with the following data:

```
Sensor llama: 886
No hay fuego
ValorLdr = 284
-----
8
-----
Nivel agua: 0mm to 5mm
-----
1
Sensor llama: 881
No hay fuego
ValorLdr = 179
-----
9
-----
Nivel agua: 0mm to 5mm
-----
1
Sensor llama: 883
No hay fuego
ValorLdr = 191
-----
9
-----
Nivel agua: 0mm to 5mm
-----
1
Sensor llama: 880
No hay fuego
ValorLdr = 189
-----
7
-----
```

- **Envío de datos NODE MCU a base de datos:** para enviar datos desde el Node MCU a el servidor web, necesitaremos código HTML para realizar una consulta al servidor. Para poder realizar esta consulta, son necesarios dos scripts en PHP: uno que cree la conexión con MySQL (*conexion.php*), y otro que inserte los datos en las tablas para su posterior visualización en phpMyAdmin (*EspPost.php*).



The screenshot shows a code editor with the file name "conexion.php". The code is as follows:

```
1  <?php
2
3      $user = "admin";
4      $pass = "eduardo02";
5      $server = "localhost";
6      $db = "DatosArduino";
7      $con = mysqli_connect($server, $user, $pass, $db);
8
9
10  ?>
```

conexion.php


```

EspPost.php
3 include 'conexion.php';
4
5 if ($con) {
6     echo "Conexion con base de datos exitosa! ";
7
8     if(isset($_POST['temperatura'])) {
9         $temperatura = $_POST['temperatura'];
10        echo "Estación meteorológica";
11        echo " Temperatura : ".$temperatura;
12    }
13
14    if(isset($_POST['humedad'])) {
15        $humedad = $_POST['humedad'];
16        echo " Humedad : ".$humedad;
17        date_default_timezone_set('Europe/Madrid');
18        $fecha_actual = date("Y-m-d H:i:s");
19
20        $consulta = "INSERT INTO TemperaturaHumedad(Temperatura,Humedad,fecha) VALUES ('$temperatura','$humedad', '$fecha_actual')";
21        // $consulta = "UPDATE DHT11 SET Temperatura='$temperatura',Humedad='$humedad' WHERE Id = 1";
22        $resultado = mysqli_query($con, $consulta);
23        if ($resultado){
24            echo " Registro dht11 en base de datos OK! ";
25        } else {
26            echo " Falla dht11! Registro BD";
27        }
28    }
}

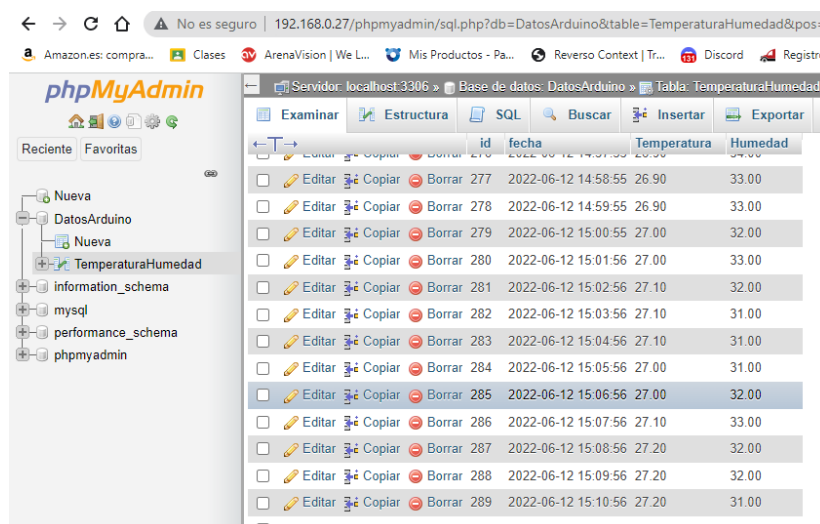
```

EspPost.php

Como se puede observar el programa *conexion.php* únicamente se encarga de conectarse con la base de datos. El programa *EspPost.php* se encarga de recibir los datos de humedad y temperatura desde el Arduino mediante el método POST y finalmente inserta los datos en las tablas de la base de datos.







4.3. Visionado de datos

Una vez enviados los datos, podemos acceder a phpMy Admin escribiendo *192.168.0.27/phpmyadmin* en el navegador, donde observaremos los datos recibidos.



	id	fecha	Temperatura	Humedad
<input type="checkbox"/>	277	2022-06-12 14:58:55	26.90	33.00
<input type="checkbox"/>	278	2022-06-12 14:59:55	26.90	33.00
<input type="checkbox"/>	279	2022-06-12 15:00:55	27.00	32.00
<input type="checkbox"/>	280	2022-06-12 15:01:56	27.00	33.00
<input type="checkbox"/>	281	2022-06-12 15:02:56	27.10	32.00
<input type="checkbox"/>	282	2022-06-12 15:03:56	27.10	31.00
<input type="checkbox"/>	283	2022-06-12 15:04:56	27.10	31.00
<input type="checkbox"/>	284	2022-06-12 15:05:56	27.00	31.00
<input type="checkbox"/>	285	2022-06-12 15:06:56	27.00	32.00
<input type="checkbox"/>	286	2022-06-12 15:07:56	27.10	33.00
<input type="checkbox"/>	287	2022-06-12 15:08:56	27.20	32.00
<input type="checkbox"/>	288	2022-06-12 15:09:56	27.20	32.00
<input type="checkbox"/>	289	2022-06-12 15:10:56	27.20	31.00

Para poder tener acceso al servidor desde fuera de la red local, se ha utilizado un servicio de DDNS (DNS Dinámico) llamado NoIP. Este programa permite acceder al servidor escribiendo en el navegador la IP pública del router o un nombre de dominio en lugar de la IP privada de la Raspberry Pi.

Nombre de host	Last Update	IP / Objetivo	Type	
 iotprueba.ddns.net Redemption in 5 days	May 11, 2022 01:12 PDT	10.0.0.2	A	<div>  Confirmar </div> <div>  Modificar </div>
 proyectoioitevebr.ddns.net Redemption in 4 days	May 26, 2022 13:05 PDT	192.168.0.29	A	<div>  Confirmar </div> <div>  Modificar </div>

Para que esto sea posible, es necesario realizar un redireccionamiento de puertos desde la web de configuración del router. El puerto necesario para la comunicación es el puerto 80 ya que es el puerto por defecto de los sistemas para publicar un servicio web por el protocolo HTTP.



UMTS

Redirección de puer...

DMZ

Redirección de puertos

La redirección de puertos permite que los equipos remotos se conecten a un dispositivo específico dentro de una LAN privada.

Nombre del servicio	Dirección IP	Protocolo	Puerto LAN	Puerto público	
ESP RASPI	192.168.0.29	TCP	80	80	  
raspi iot2	192.168.0.150	TCP	21	21	  

Una vez realizado el redireccionamiento, podemos acceder a el servidor escribiendo la IP pública del router en el navegador.

← → ↺ 🏠

⚠ No es seguro | 85.251.73.62/DatosArduino/

 Amazon.es: compra...
 Clases
 ArenaVision | We L...
 Mis Pro

Index of /DatosArduino

Name	Last modified	Size	Description
 Parent Directory		-	
 CoolTermRaspberryPi/	2022-06-12 14:20	-	
 SalidaSerie.txt	2022-06-12 13:30	0	
 recibeserie.py	2022-06-12 13:30	282	
 serie2.py	2022-06-12 14:10	253	

Apache/2.4.53 (Raspbian) Server at 85.251.73.62 Port 80

A su vez, podemos acceder mediante el dominio creado en NoIP (proyectoioitevebr.ddns.net).

5. CONCLUSIÓN

El objetivo final de este proyecto era poder controlar unos actuadores mediante unos sensores, y los valores de estos sensores visualizarlos a tiempo real remotamente desde Internet.

El coste del proyecto es relativamente bajo, ya que estos elementos son baratos. Por el contrario, la calidad y rendimiento de los sensores hace que no suele ser aplicable profesionalmente, es decir, alguien que quiera controlar toda la seguridad de su casa (abrepuertas, sensores PIR, alarmas, etc.) no va a recurrir a colocar estos sensores para ello, ya que además de necesitar bastantes placas de Arduino alimentadas constantemente por cada zona en la que haya sensores, el rendimiento de estos no es suficiente para un correcto funcionamiento.

Sin embargo, la flexibilidad de estas placas y sensores permite realizar múltiples proyectos (domotización, control de maquinaria, control de motores, servidores web, etc).

La solución al problema de los sensores y actuadores, es utilizar relés controlados por el Arduino, y en estos relés conectar lámparas, ventiladores, motores para persianas, sensores de mayor calidad, etc. Obviamente el precio aumentaría, pero la diferencia de calidad y posibilidades es muy grande.

Personalmente este proyecto me ha permitido aprender lenguajes de programación nuevos (php, HTML), aprender a montar un servidor en una Raspberry Pi, enviar datos a una pagina web, entre otras cosas.

6. MI PROYECTO TIENE FUTURO

Aunque la calidad de los sensores no sea aplicable profesionalmente hablando, como ya he mencionado, se pueden utilizar relés controlados por Arduino. Si el Arduino está conectado a Internet, puedes controlarlos remotamente.

Esto permitiría controlar toda la iluminación de tu hogar, controlar motores de persianas y toldos, controlar sensores, electroválvulas, etc.

Incluso industrialmente hablando, los Arduino son muy utilizados para controlar máquinas de producción y robots, debido a su bajo coste y a su fácil programación.

En definitiva, podemos utilizar Arduino para controlar relés conectados a diferentes elementos y a un precio relativamente bajo.

7. PRESUPUESTO

ELEMENTO	UNIDADES	PRECIO UD.	COSTE TOTAL
Raspberry Pi 3 Modelo B	1	59,00 €	59,00 €
Tarjeta MicroSD 32 GB Kingston + Adaptador MicroSD-SD	1	4,99 €	4,99 €
Sensor PIR HC-SR501	1	1,99 €	1,99 €
Módulo Relé KY-019	1	2,00 €	2,00 €
Sensor LDR KY-018	1	0,39 €	0,39 €
Buzzer activo KY-012	1	1,30 €	1,30 €
AZ Delivery NodeMCU V3	1	8,29 €	8,29 €
Resistencias, cables Dupont y LEDs			
Elegoo Mega 2560	1	24,99 €	24,99 €
Sensor llama KY-026	1	0,65 €	0,65 €
Sensor agua	1	0,97 €	0,97 €
Bandeja de pruebas NIMO Electronic	2	4,99 €	9,98 €
TOTAL:			112,86 €

8. PROGRAMACIÓN

8.1. Código Arduino Mega

```
int analogllama = A0;
int digillama = 24;
const int alarma = 7;
const int pinLed = 3;
const int pinLdr = A1;
const int pinLed2 = 5;
const int limite = 250;
const int pinagua = A5;
int input = 0;
int pirpin = 11;
int valorpir;
int espera = 1000;
int Relay = 26;

void setup() {
  Serial.begin(9600);
  pinMode(pinLed, OUTPUT);
  pinMode(pinLed2, OUTPUT);
  pinMode(pinLdr, INPUT);
  digitalWrite(pinLed, LOW);
  pinMode(alarma, OUTPUT);
  pinMode(digillama, INPUT);
  pinMode(analogllama, INPUT);

  pinMode(pirpin, INPUT);
  pinMode(pinagua, INPUT);
  digitalWrite(alarma, LOW);
  pinMode(Relay, OUTPUT);
}
```

```

void loop() {

valorpir = digitalRead(pirpin);
Serial.println(valorpir);
digitalWrite(alarma, valorpir);

llama();
ldr();
agua();

}

void ldr(){
input = analogRead(pinLdr);

if (input > limite) {
digitalWrite(pinLed2, HIGH);
Serial.println((String)"ValorLdr = " + input);
}
else {
digitalWrite(pinLed2, LOW);
Serial.println((String)"ValorLdr = " + input);
}

Serial.println("-----");
delay(1000);
}

void llama(){

int valorllama1 = analogRead(analogllama);
int valorllama2 = digitalRead(digillama);

Serial.println((String)"Sensor llama: " + valorllama1);
if (valorllama1 < 200){
Serial.println("-----");
Serial.println("Fuego detectado");
digitalWrite(alarma, HIGH);
digitalWrite(Relay, HIGH);
digitalWrite(pinLed, HIGH);

}

else {
Serial.println("No hay fuego");
digitalWrite(alarma, LOW);
digitalWrite(Relay, LOW);
digitalWrite(pinLed, LOW);
}
}

```

```

void agua(){

  int valagua = analogRead(pinagua);
  Serial.println(valagua);
  Serial.println("-----");

  if (valagua <= 5) {
    Serial.println("Nivel agua: 0mm - Vacío!");

    delay(espera);
  }
  else if (valagua > 6 && valagua <= 100) {
    Serial.println("Nivel agua: 0mm to 5mm");

    delay(espera);
  }
  else if (valagua > 100 && valagua <= 110) {
    Serial.println("Nivel agua: 5mm to 10mm");

    delay(espera);
  }
  else if (valagua > 110 && valagua <= 120) {
    Serial.println("Nivel agua: 10mm to 15mm");
  }
  else if (valagua > 120 && valagua <= 130) {
    Serial.println("Nivel agua: 15mm to 20mm");

    delay(espera);

  }
  else if (valagua > 140 && valagua <= 150) {
    Serial.println("Nivel agua: 20mm to 25mm");

    delay(espera);
  }
  else if (valagua > 160 && valagua <= 170) {
    Serial.println("Nivel agua: 25mm to 30mm");

    delay(espera);
  }
  else if (valagua > 180 && valagua <= 190) {
    Serial.println("Nivel agua: 30mm to 35mm");

    delay(espera);
  }
  else if (valagua > 200) {
    Serial.println("Nivel agua: 35mm to 40mm");

    delay(espera);
  }
  Serial.println("-----");
}

```

8.2. Código NodeMCU

```
#include <ESP8266HTTPClient.h>

#include <ESP8266WiFiMulti.h>

#include <WiFiClient.h>


//const char* ssid ="sagemcomFBA0";
//const char* password ="UMNEMZWCTLFT2Y";
const char* ssid ="MOVISTAR_E358";
const char* password ="3uTVs7snTAsRijkg3g73";
// Variables para lectura del DHT 11

float t;

float h;

float f;

float hif;

float hic;

#include "DHT.h"

#define DHTPIN 2

#define DHTTYPE DHT11 // DHT 11

DHT dht(DHTPIN, DHTTYPE);

WiFiClient client;


void setup() {

  Serial.begin(115200);

  Serial.println(F("DHT 11 prueba de conexión con el servidor"));

  dht.begin();


  WiFi.begin(ssid, password);

  Serial.print("Conectando...");

  while (WiFi.status() != WL_CONNECTED) {
```

```

delay(500);
Serial.print(".");
}
Serial.println("Conexión OK!");
Serial.print("IP Local: ");
Serial.println(WiFi.localIP());
}
void loop() {
    LecturaTH();
    EnvioDatos();

}
void LecturaTH(){
    h = dht.readHumidity();
    t = dht.readTemperature();
    f = dht.readTemperature(true);
    if (isnan(h) || isnan(t) || isnan(f)) {
        Serial.println(F("Failed to read from DHT sensor!"));
        return;
    }
    hif = dht.computeHeatIndex(f, h);
    // Calcula indice calor en grados centigrados (isFahreheit = false)
    hic = dht.computeHeatIndex(t, h, false);
}
void EnvioDatos(){
    if (WiFi.status() == WL_CONNECTED){
        HTTPClient http; // creo el objeto http
        String datos_a_enviar = "temperatura=" + String(t) + "&humedad=" + String(h);
    }
}

```



```

http.begin(client,"http://192.168.1.42/EspPost.php");
http.addHeader("Content-Type", "application/x-www-form-urlencoded");

int codigo_respuesta = http.POST(datos_a_enviar);
if (codigo_respuesta>0){
  Serial.println("Código HTTP: "+ String(codigo_respuesta));
  if (codigo_respuesta == 200){
    String cuerpo_respuesta = http.getString();
    Serial.println("El servidor respondió: ");
    Serial.println(cuerpo_respuesta);
  }
} else {
  Serial.print("Error enviado POST, código: ");
  Serial.println(codigo_respuesta);
}
http.end();
} else {
  Serial.println("Error en la conexion WIFI");
}
delay(60000);
}

```

8.3. Código PHP conexion.php

```
conexion.php
1  <?php
2
3      $user = "admin";
4      $pass = "eduardo02";
5      $server = "localhost";
6      $db = "DatosArduino";
7      $con = mysqli_connect($server, $user, $pass, $db);
8
9
10  ?>
```

8.4. Código PHP EspPost.php

```
EspPost.php
1  <?php
2
3  include 'conexion.php';
4
5  if ($con) {
6      echo "Conexion con base de datos exitosa! ";
7
8      if(isset($_POST['temperatura'])) {
9          $temperatura = $_POST['temperatura'];
10         echo "Estación meteorológica";
11         echo " Temperatura : ".$temperatura;
12     }
13
14     if(isset($_POST['humedad'])) {
15         $humedad = $_POST['humedad'];
16         echo " Humedad : ".$humedad;
17         date_default_timezone_set('Europe/Madrid');
18         $fecha_actual = date("Y-m-d H:i:s");
19
20         $consulta = "INSERT INTO TemperaturaHumedad(Temperatura,Humedad,fecha) VALUES ('$temperatura','$humedad', '$fecha_actual')";
21         // $consulta = "UPDATE DHT11 SET Temperatura='$temperatura',Humedad='$humedad' WHERE Id = 1";
22         $resultado = mysqli_query($con, $consulta);
23         if ($resultado){
24             echo " Registro dht11 en base de datos OK! ";
25         } else {
26             echo " Falla dht11! Registro BD";
27         }
28     }
29
30
31
32 } else {
33     echo "Falla! conexion con Base de datos ";
34 }
35
36
37 ?>
```

