Escuela Politécnica Superior de Linares

Grado en Ingeniería Telemática

UNIVERSIDAD DE JAÉN
Escuela Politécnica Superior de Linares

Trabajo Fin de Grado

MONITORIZACIÓN DE VARIABLES MEDIOAMBIENTALES MEDIANTE REDES DE SENSORES INALÁMBRICOS. PREDICCIÓN DEL PERIODO DE MUESTREO.

Alumno: **D. AMINE ALRHARAD.**

Tutor: Prof. D. MANUEL ÁNGEL GADEO MARTOS
Depto.: INGENIERÍA DE TELECOMUNICACIÓN

JUNIO, 2019

Escuela Politécnica Superior de Linares

Trabajo Fin de Grado

Curso 2018-2019

MONITORIZACIÓN DE VARIABLES MEDIOAMBIENTALES MEDIANTE REDES DE SENSORES INALÁMBRICOS. PREDICCIÓN DEL PERIODO DE MUESTREO.

Alumno: D. Amine Alrharad.

Tutor: Prof. D. Manuel Ángel Gadeo Martos.

Depto.: Ingeniería de Telecomunicación

Junio, 2019

Visto bueno a la defensa del TFG

Firma del autor:

Firma del tutor:

GADEO MARTOS MANUEL ANGEL - 25987883E

Firmado digitalmente por GADEO MARTOS MANUEL ANGEL - 25987883E
Nombre de reconocimiento (DN): c=ES, serialNumber=25987883E, sn=GADEO MARTOS, givenName=MANUEL ANGEL, cn=GADEO MARTOS MANUEL ANGEL - 25987883E
Fecha: 2019.06.20 19:25:05 +02'00'

Índice

ÍNDICE DE FIGURAS

Índice de tablas

CAPÍTULO 1. RESÚMEN

Este trabajo fin de grado tiene como objetivo principal la determinación del periodo de muestreo en un nodo de una red de sensores inalámbrica, que permita la reducción del consumo energético, minimizando el error entre el dato actual ofrecido por el sensor y el dato real.

Para tal fin, se va a buscar una tecnología de redes de sensores inalámbricas que permita leer los parámetros medioambientales (temperatura, humedad y luminosidad), y enviarlos a un servidor para almacenarlos, procesarlos y mostrarlos al usuario.

Finalmente, a fin de que nuestro sistema determine de forma automática el periodo de muestreo, se va a usar un sistema borroso basado en reglas.

ABSTRACT

This final degree project has as main objective the determination of the sampling period in a node of a wireless sensor network, which allows the reduction of energy consumption, minimizing the error between the current data offered by the sensor and the real data.

For this purpose, we will look for a wireless sensor network technology that allows us to read the environmental parameters (temperature, humidity and brightness), and send them to a server to store, process and show them to the user.

Finally, in order to automate the determination of the sampling period, a fuzzy system based on rules will be used.

CAPÍTULO 2. INTRODUCCIÓN

2.1 Introducción

Hoy en día, varios sectores como la agricultura, industria, domótica, medicina, seguridad, etc…, necesitan datos en tiempo real para optimizar sus procesos y tomar las decisiones oportunas.

Una de las tecnologías que está emergiendo es "la red de sensores inalámbrica" (Wireless Sensor Network, WSN), su amplio rango de aplicaciones y su bajo coste hace de esta tecnología la solución ideal para tal fin.

A continuación, se van a presentar una serie de ejemplos de la aplicación de las redes de sensores inalámbricas.

2.1.1. Agricultura de precisión

El término agricultura de precisión hace referencia a la aplicación del internet de las cosas (Internet of Things, IoT) al sector agrícola.

El uso de las redes de sensores inalámbricas en este sector permite monitorizar parámetros ambientales, como la temperatura, la humedad y la luminosidad, en tiempo real y controlar los dispositivos de forma remota, todas estas ventajas permiten optimizar la producción, la calidad y la sostenibilidad medioambiental.

2.1.2 Monitorización ambiental

El control ambiental es muy importante y esta tecnología es ideal cuando se desea monitorizar las condiciones medioambientales en varios entornos, como puede ser los invernaderos, edificios, cadenas de frío, etc. Las características a monitorizar pueden ser la temperatura, la humedad, la luminosidad, etc.

2.1.3 Domótica

En el sector domótico, se necesita saber en todo momento el valor de ciertos parámetros para garantizar el bienestar de las personas.

Por ejemplo, se necesita saber el valor de la temperatura para encender o apagar el aire acondicionado, la luminosidad, para subir o bajar las persianas, etc.

Según lo anteriormente expuesto en los puntos **2.1.1, 2.1.2 y 2.1.3,** se puede afirmar que las redes de sensores inalámbricas tienen varias aplicaciones, desde la agricultura hasta la automatización de viviendas.

Un aspecto muy importante a la hora de hablar de las WSNs es el consumo energético. En la mayoría de los casos, este tipo de redes funciona de manera aislada y continuada todo el día, lo que supone un consumo energético elevado.

2.2 Estado del arte

2.2.1 Monitorización de parámetros medioambientales

En la actualidad, existen varias empresas que ofrecen el servicio de monitorización en tiempo real de parámetros medioambientales mediante redes de sensores inalámbricas, tales como la temperatura, la humedad, la luminosidad, etc…, así como la posibilidad de la gestión remota por internet de los nodos. A continuación, se presentan las empresas más destacadas del sector.

2.2.1.1 AGROCONECTA

Utilizan redes de sensores inalámbricas para monitorizar parámetros ambientales y climatológicos. Los datos recogidos por los sensores son transmitidos a un servidor en la nube, lo que permite a los usuarios de este servicio consultarlos en tiempo real desde cualquier dispositivo que tenga conexión a internet.

También, ofrece la posibilidad de acceder al histórico, y obtener los datos de forma gráfica, informes, hojas de cálculo, así como el cálculo de los parámetros estadísticos más importantes. [1]

2.2.1.2 SOFTCRITS

Ofrece un conjunto de soluciones tecnológicas, entre ellas la monitorización inalámbrica de variables climáticas de edificios para la optimización energética de sus instalaciones. Estos sistemas tienen múltiples aplicaciones en el mundo actual, desde el control remoto de temperatura, humedad, luz…, hasta la eficiencia productiva en instalaciones de agricultura. [2]

2.2.1.3 SMARTY PLANET

Empresa especializada en el campo de las redes de sensores inalámbricas al servicio de las ciudades inteligentes (Smart cities). Desarrollan y producen su propio hardware y software para las soluciones que ofrecen.

Entre las soluciones que ofrecen, se encuentra “Smarty Meteo”, que son estaciones meteorológicas para Smart cities. [3]

2.2.1.4 SAYME

Ofrece soluciones tecnológicas de redes de sensores inalámbricas de **bajo consumo**. Tanto el hardware como el software ofrecido lo desarrolla la propia empresa. Utiliza varias tecnologías, como GPRS, ZigBee, LoRa, etc. [4]

Ofrecen soluciones en varios campos. A continuación, se presentan algunos:

- Gestión eficiente del alumbrado.
- Procesos industriales.
- Agricultura.
- Smart cities.

Según la información proporcionada, todas las empresas antes mencionadas ofrecen el servicio de monitorización en tiempo real de parámetros ambientales y climatológicos con

periodos de tiempo fijos y controlados por el administrador del sistema manualmente, lo que supone un desperdicio de la energía, puesto que los datos capturados en algunas ocasiones no presentan una gran variabilidad y sería necesario cambiar el periodo de muestreo para ahorrar energía.

La solución que se propone en este TFG es el desarrollo de un sistema basado en la lógica borrosa (Fuzzy Rule-Based Systems, FRBS), que, a partir de los datos capturados, nos permita obtener un periodo de muestreo con un consumo energético ajustado y un error razonable.

2.2.2 Fuzzy Rule-Based Systems (FRBS)

Un Sistema Borroso, en sentido amplio, es un sistema basado en Lógica Borrosa, donde ésta puede utilizarse como base para la representación de diferentes formas de representación de conocimiento, o para modelar las interacciones existentes entre las variables de un sistema [5].

Un Controlador Borroso es un sistema experto, que incorpora conocimiento humano en sus bases de conocimiento a través de un conjunto de reglas de control y de funciones de pertenencia.

La Lógica Borrosa tiene gran utilidad, ya que ella nos permite tratar problemas demasiado complejos, mal definidos o para los cuales no existen modelos matemáticos precisos.

Gracias a este tipo de lógica se ha permitido modelizar y resolver situaciones consideradas intratables desde el punto de vista de la Lógica Clásica, en nuestro caso, tenemos una red de sensores inalámbricos capturando datos sobre variables medioambientales a una frecuencia constante y queremos diseñar un controlador borroso que permita variar el periodo de captura, aplicando unas reglas.

A continuación, se va a introducir la estructura básica de un controlador borroso:

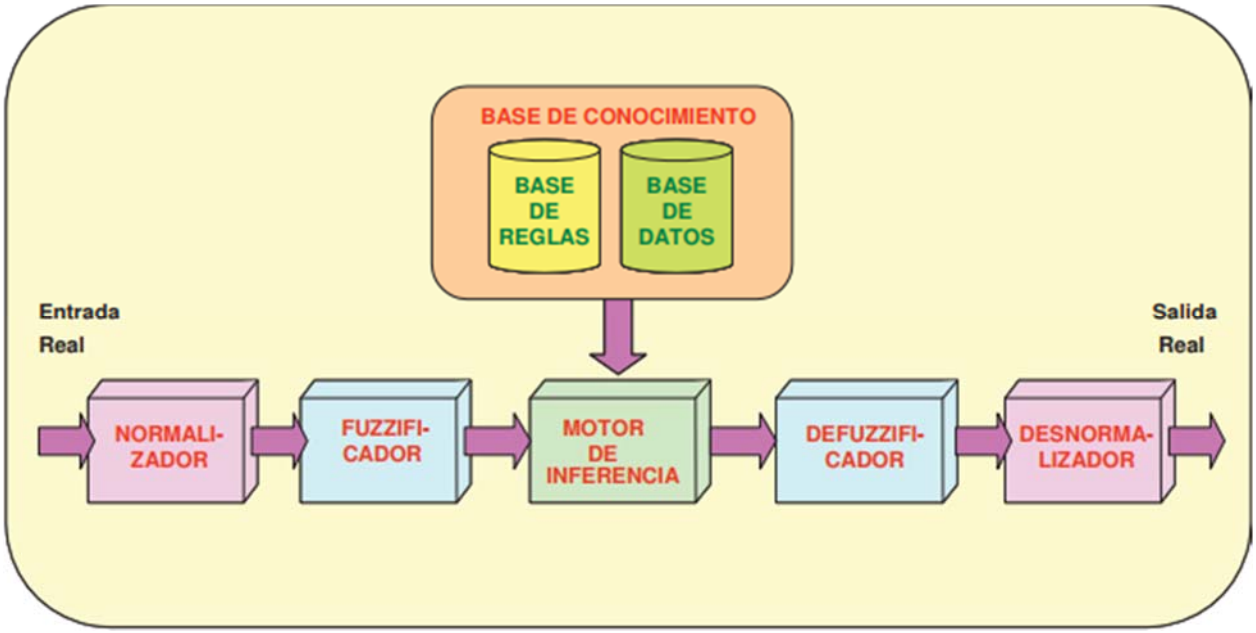

Figura 2.1 Estructura genérica de un controlador borroso [5]

Como se puede ver en la **figura 1**, los elementos básicos que constituyen un controlador borroso son los siguientes:

- **Normalizador**: este módulo normaliza los valores tomados por las variables de entrada (temperatura, humedad y luminosidad) al intervalo de valores entre 0 y 1.
- **Fuzzificador**: en este elemento se transforma el valor normalizado, tomado por las variables de entrada, a una información borrosa, asignándole un grado de pertenencia a cada uno de los conjuntos borrosos definidos en cada variable.
- **Motor de inferencia**: en este módulo se obtiene un conjunto borroso para cada variable de salida (en nuestro caso, la variable de salida es el periodo de muestreo), a partir de los conjuntos borrosos de las variables de entrada, aplicando una serie de reglas predefinidas.
- **Base de conocimiento:** Este módulo contiene dos tipos de información, almacenadas en las siguientes bases de datos:
    - **Base de datos:** contiene información sobre los conjuntos borrosos, que es utilizada en el proceso de defuzzificación.
    - **Base de reglas:** Contiene un conjunto de reglas lingüísticas, que serán utilizadas en el motor de inferencias.
- **Defuzzificador:** Como resultado al proceso de inferencia se obtiene un conjunto borroso. Teniendo en cuenta que, para controlar un sistema, no se puede aplicarle un conjunto borroso, entonces, será necesario un elemento que transforma los conjuntos borrosos inferidos para cada variable de salida a un valor real para aplicarlo al sistema a controlar.
- **Desnormalizador:** Este módulo transforma los valores tomados por las variables de salida que están entre 0 y 1, al rango de valores adecuados.

CAPÍTULO 3. OBJETIVOS

En este apartado se van a presentar los objetivos principales del TFG.

- Configurar y programar un nodo de una red de sensores inalámbricos para capturar periódicamente datos sobre parámetros medioambientales (temperatura, humedad y luminosidad).
- Diseñar un servidor web en la nube que permita el almacenamiento, acceso a los datos capturados, así como la gestión de los nodos y usuarios del sistema.
- Diseñar un algoritmo que a partir de las variables medioambientales de entrada obtenga un periodo de muestreo que permita obtener un consumo energético ajustado con un error razonable en la medición de las variables medioambientales.

CAPÍTULO 4. TECNOLOGÍAS UTILIZADAS

4.1 Elección de la tecnología de redes de sensores

4.1.1 Introducción

Una red de sensores inalámbrica, se basa en dispositivos de bajo coste y consumo (se llaman nodos) que son capaces de obtener información de su entorno, procesarla localmente, y comunicarla a través de enlaces inalámbricos hasta un nodo central de coordinación [6].

La red de sensores inalámbrica está formada por numerosos nodos distribuidos, que utilizan sensores para leer distintos parámetros en distintos puntos, entre ellas la temperatura, la humedad, la luminosidad, etc.

Los nodos son unidades autónomas que constan de un microcontrolador, una fuente de energía (por ejemplo, una batería), un radio-transceptor (RF) y un elemento sensor.

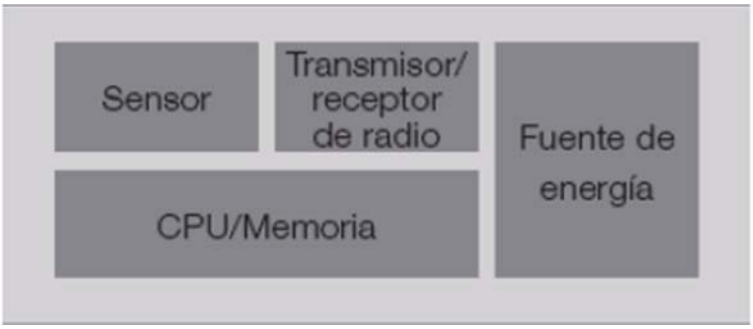

Figura 4.1 Estructura de un nodo de una WSN

Debido a las limitaciones en las fuentes de energía, los nodos se construyen teniendo en cuenta la conservación de energía, y generalmente pasan mucho tiempo en modo "durmiente" (sleep) de bajo consumo de potencia.

4.1.2 Tecnologías inalámbricas para redes de sensores inalámbricas

Para transmitir los parámetros ambientales (temperatura, humedad y luminosidad) leídos por los sensores al servidor, se va a hacer uso de una tecnología inalámbrica. A continuación, vamos a presentar un análisis comparativo de los estándares inalámbricos más importantes, para poder elegir la opción más adecuada.

4.1.2.1 Zigbee

- Baja velocidad de transferencia: 250Kbps.
- Bajo consumo.
- Seguridad.

- Capacidad para soportar un gran número de nodos.
- Alcance: 10-100m.

4.1.2.2 WiFi

- infraestructura ya instalada que transfiere datos con rapidez y permite manejar grandes cantidades de datos.
- Coste: bajo.
- Alcance: Aproximadamente 100m.
- Velocidad de transferencia: puede llegar hasta 600 Mbps.
- Consumo energético: Bajo-Medio.

4.1.2.3 Bluetooth

- Corto alcance: menos de 10m.
- Velocidad de transferencia: puede llegar hasta 1 Mbps.
- Consumo energético: Medio.

4.1.2.4 Red de telefonía móvil

- Velocidad de transferencia: 35-170kps (GPRS), 384Kbps-2Mbps (UMTS), 3-10Mbps (LTE).
- Consumo energético: muy alto.
- Coste: muy alto.
- Alcance: cubre grandes áreas (km).

4.1.2.5 Sigfox

- Alcance: 30-50km (ambientes rurales), 3-10km (ambientes urbanos).
- Velocidad de transferencia: 10-1000bps.
- Consumo: bajo.
- Coste: alto.

4.1.2.6 Elección

Después de hacer un análisis comparativo de las tecnologías inalámbricas más importantes, se ha optado por usar la tecnología WIFI en las comunicaciones entre los nodos y el servidor por las siguientes razones:

- **Bajo coste**: infraestructura presente en varios entornos de la vida cotidiana (domésticos, comerciales, etc.).
- **Alta velocidad de transferencia**.
- **Alta fiabilidad**.
- **Consumo energético bajo:** mA cuando está transmitiendo y µA en modo Deep sleep.

4.1.2 Alternativas para la implementación de la red de sensores inalámbrica

Para montar una red de sensores inalámbrica, se debe tener en cuenta varios factores, entre ellos, el coste económico, el consumo energético, la documentación disponible, la flexibilidad, etc. Actualmente, existen en el mercado varias opciones, entre ellas destacamos.

4.1.2.1 Arduino

Arduino es una plataforma de código abierto (open-source) basada en hardware y software flexibles y fáciles de usar. Dispone de varias entradas analógicas y digitales que le permiten leer información de su entorno, así como conectarse a internet mediante módulos externos.

El microcontrolador de la placa se programa usando el "Arduino Programming Language" y el "Arduino Development Environment, ADE". El software se puede descargar gratuitamente. [7]

Arduino presenta varias ventajas, las más importantes son: [8]

- **Barato**: Las placas Arduino son relativamente baratas comparadas con otras plataformas microcontroladoras. La versión menos cara del módulo Arduino cuesta menos de 20 euros [9].
- **Multiplataforma**: El software de Arduino se ejecuta en sistemas operativos Windows, Mac y Linux. La mayoría de los sistemas microcontroladores están limitados a Windows.
- **Entorno de programación simple y claro**: El entorno de programación de Arduino es fácil de usar e intuitivo.
- **Código abierto y software extensible**: El software Arduino está publicado como herramientas de código abierto. El lenguaje puede ser expandido mediante librerías C++.
- **Código abierto y hardware extensible**: Arduino está basado en microcontroladores ATMEGA8 y ATMEGA168 de Atmel. Los planos para los módulos están publicados bajo licencia Creative Commons, por lo que diseñadores experimentados de circuitos pueden hacer su propia versión del módulo, extendiéndolo y mejorándolo. Incluso usuarios relativamente inexpertos pueden construir la versión de la placa del módulo para entender cómo funciona y ahorrar dinero.

Figura 4.2 Arduino UNO

4.1.2.2 BeagleBone

Beaglebone es una tarjeta de desarrollo de bajo costo desarrollada por la organización Beagleboard.org, la cual está enfocada en estimular el uso de software y hardware open source, así como el conocimiento y el intercambio de ideas. Es una plataforma que corre bajo un sistema operativo Linux (cabe señalar que actualmente existen varias distribuciones de Linux para las plataformas BeagleBone), y que cuenta con diversas entradas y salidas de propósito general (GPIO – General Purpose Input/Output) las cuales cuentan con diversas funciones entre las cuales se encuentran (Entrada/Salida Digitales, Entradas Analógicas, Salidas con PWM (Pulse-Width Modulation), soporte para I2 (Internet2) & SPI (Serial Peripheral Interface)). Además, cuenta con un puerto Ethernet para la comunicación en red con otros dispositivos y un puerto USB 2.0 para la comunicación con otros dispositivos. [10]

Figura 4.3 BeagleBone

4.1.2.3 Módulos ESP-XX

Al igual que con Arduino, donde se trabaja con la placa o circuito integrado, con el ESP8266 ocurre exactamente lo mismo. El fabricante AI-Thinker proporciona la serie ESP con diferentes modelos para diferentes usos. A parte han ido surgiendo diferentes placas que incorporan algún módulo ESP. [11]

Figura 4.4 Diferentes modelos del ESP

4.1.2.3.1 ESP-01

Se trata del módulo más popular. El precio oscila entre los 2€ y los 4€ [9]. Tiene disponible dos pines GPIO digitales para controlar sensores y actuadores. También se puede llegar a utilizar para este uso los pines RX (pines de recepción) y TX (pines de transmisión) si no se utilizan para la comunicación a través del puerto serie. Se puede programar a través de un adaptador serie/USB o con el cableado adecuado, a través de Arduino. Los conectores que vienen por defecto, no permiten conectarlo a la protoboard.

Figura 4.5 Módulo ESP-01

4.1.2.3.2 ESP-05

Quizás sea el módulo más simple de toda la gama. Está destinado a ser un Shield WiFi para Arduino. Su precio ronda los 7€ [9]. La disposición de los pines nos permite un fácil conexionado con la protoboard. Por el contrario, no dispone de ningún puerto GPIO accesible.

Figura 4.6 Módulo ESP-05

4.1.2.3.3 ESP-12

Este módulo permite hacer bastantes más cosas que los módulos anteriores. Su precio ronda los 5€ [9]. Tenemos acceso a 11 puertos GPIO de los cuales uno, es analógico. La

configuración en modo dormido es muy sencilla. Esto nos permitirá ahorrar mucha energía. Por el contrario, la conexión con la protoboard no es muy amigable. Necesitamos soldar los pines o comprar un adaptador, aunque también hay que soldar.

Figura 4.7 Módulo ESP-12

4.1.2.3.4 ESP-201

Su precio ronda los 4€ [9]. En principio solo podemos acceder a 11 puertos GPIO, pero tras unas modificaciones, podríamos acceder a un par más de ellos. Lo podemos encajar fácilmente en una protoboard y permite el acople de una antena externa para tener más alcance.

Figura 4.8 Módulo ESP-201

4.1.2.3.5 NodeMCU

El NodeMCU es el módulo más característico de este tipo. Su precio ronda los 6€ [9]. A diferencia de los otros módulos, viene con todo lo necesario para empezar a trabajar de forma autónoma. Incluye un adaptador serie/USB y se alimenta a través del Micro USB. Está basado en el ESP-12 y la última versión oficial es la 3.

Figura 4.9 NodeMCU

4.1.2.4 Elección

Para montar la red de sensores inalámbrica, se van a usar los microcontroladores "NodeMCU" como nodos, debido a los siguientes motivos:

- Bajo coste, 6 euros aproximadamente [9].
- Mucha documentación disponible en la red.
- Son de código abierto.
- Alta calidad.
- Flexibilidad.

4.2 Sensores

Este apartado trata sobre los sensores que se van a usar en el TFG, para monitorizar los parámetros medioambientales: temperatura, humedad, y luminosidad. Se puede hacer la lectura de varios parámetros, pero para nuestra aplicación serán suficientes estos tres.

En la elección de los sensores se ha optado por usar sensores que sean baratos y que tengan una calidad aceptable en la lectura de los parámetros.

4.2.1 Sensores de temperatura y humedad

Para la medición de la temperatura y la humedad se han elegido los sensores de la familia DHT. Este tipo de sensores proporciona de forma digital la temperatura y la humedad, con diferente precisión según el modelo. Básicamente hay dos variantes: DHT11 y DHT22.

Las características del DHT11 son [12]:

- Muy barato.
- Funciona con 3,3 y 5V de alimentación.
- Rango de temperatura: de 0º a 50º con 5% de precisión.
- Rango de humedad: de 20% al 80% con 5% de precisión.
- Una muestra por segundo.
- Bajo consumo.
- Devuelve la medida en ºC.

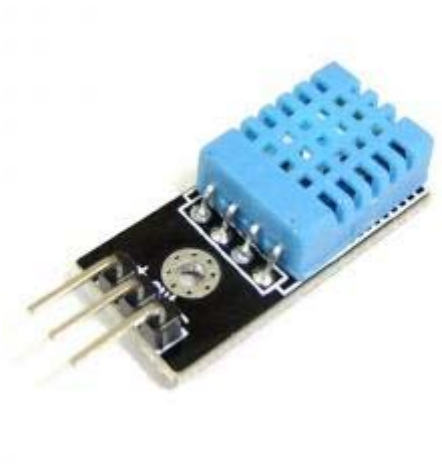

Figura 4.10 Sensor DHT11

En cuanto al DHT22 [12]:

- Barato.
- Funciona con 3,3V y 5V de alimentación.
- Rango de temperatura: de -40º a 125º ± 0,5ºC.
- Rango de humedad: de 0% al 100% con 5% de precisión.
- Lee dos veces por segundo.
- Bajo consumo.
- Devuelve la medida en ºC.

Figura 4.11 Sensor DHT22

Como se puede apreciar, los dos sensores ofrecen una buena calidad con un precio relativamente bajo. Para nuestra aplicación el **DHT11** cumple con los requisitos mínimos que necesitamos, por lo que será el candidato elegido para leer la temperatura y la humedad del ambiente.

4.2.2 Sensor de luminosidad

Para la obtención de la luminosidad se va a usar una LDR (Light Dependent Resistor).

LDR en una resistencia que varía en función del nivel de luz recibido, está disminuye con el aumento de intensidad lumínica incidente.

Este sensor tiene una media-baja precisión, pero para nuestra aplicación es adecuado, porque solo necesitamos saber el nivel de luz en tres márgenes: bajo, medio y alto.

A continuación, se presenta una foto del sensor y una tabla donde aparezcan sus principales características:

Figura 4.12 LDR

Specification	Light resistance (10Lux) (KΩ)	Dark resistance (MΩ)	γ_{10}^{100}	Response time (ms)		Illuminance resistance Fig. No.
				Increase	Decrease	
Φ5 series	4-7	0.5	0.5	30	30	2
	5-10	0.8	0.6	30	30	2
	10-20	2	0.6	20	30	3
	20-30	3	0.7	20	30	4
	30-50	4	0.8	20	30	4
	50-100	8	0.9	20	30	5
	100-200	15	0.95	20	30	6

Tabla 4.1 Hoja de características LDR (1)

Specification	Type	Maximum Voltage	Maximum power	Environmental temperature	Spectrum peak value
Φ5 series	GL5516	150	90	-30~+70	540
	GL5528	150	100	-30~+70	540
	GL5537-1	150	100	-30~+70	540
	GL5537-2	150	100	-30~+70	540
	GL5539	150	100	-30~+70	540
	GL5549	150	100	-30~+70	540

Tabla 4.2 Hoja de características LDR (2)

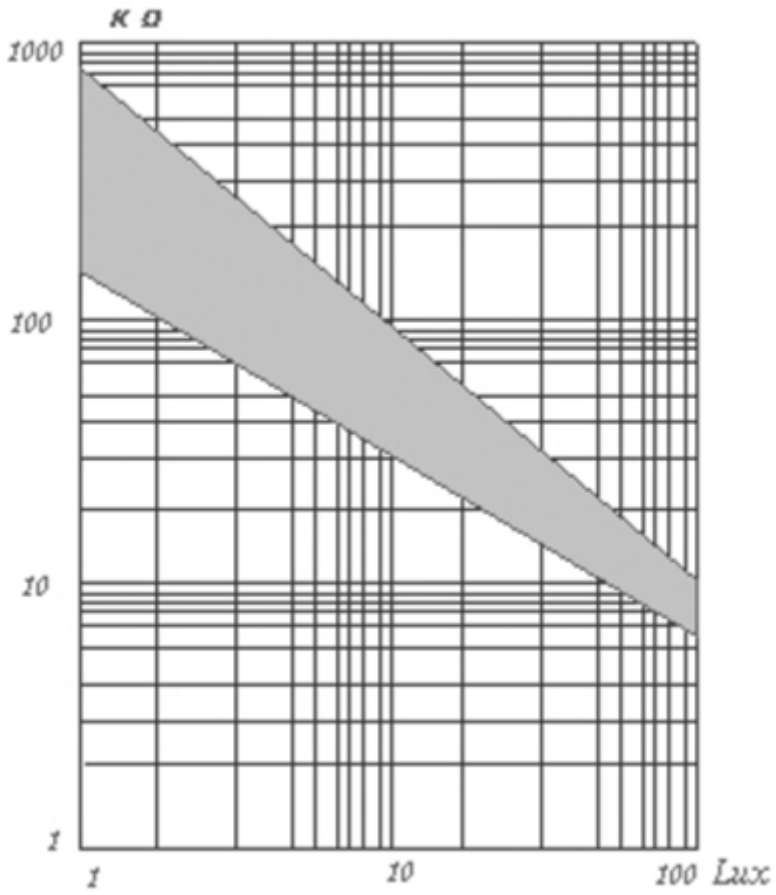

Figura 4.13 Variación de la Resistencia en función de luxes

4.3 Base de datos

4.3.1 Sistema Gestor de Base de Datos SGBD

Un sistema gestor de base de datos o SGBD (DBMS, en inglés) es un sistema que permite la creación, gestión y administración de bases de datos, así como la elección y manejo de las estructuras necesarias para el almacenamiento y búsqueda de la información del modo más eficiente posible. [13]

En la actualidad, existe multitud de SGBD, la mayoría de ellos son relacionales. A continuación, se van a mostrar los gestores de bases de datos más usados.

4.3.1.1 MySQL

Es un sistema de gestión de base de datos relacional, multihilo y multiusuario, seguramente el más usado en aplicaciones creadas como software libre.

Por un lado, se ofrece bajo la licencia GNU GPL, pero, empresas que quieran incorporarlo en productos propios pueden comprar a la empresa una licencia que les permita ese uso.

Ventajas:

- Velocidad al realizar las operaciones.
- Bajo costo en requerimientos para la elaboración de bases de datos.
- Facilidad de configuración e instalación.

4.3.1.2 Microsoft SQL Server

Es un sistema de gestión de bases de datos relacionales basado en el lenguaje "Transact-SQL", capaz de poner a disposición de muchos usuarios grandes cantidades de datos de manera simultánea.

Es un sistema propietario de Microsoft. Sus principales características son:

- Soporte de transacciones.
- Escalabilidad, estabilidad y seguridad.
- Soporta procedimientos almacenados.
- Incluye también un potente entorno gráfico de administración, que permite el uso de comandos DDL y DML gráficamente.
- Permite trabajar en modo cliente-servidor donde la información y datos se alojan en el servidor y los terminales o clientes de la red sólo acceden a la información.
- Además, permite administrar información de otros servidores de datos.

Su principal desventaja es el precio, aunque cuenta con una versión EXPRESS que permite usarlo en entornos pequeños (Aprox. unos 4GB de información y varios millones de registros por tabla).

4.3.1.3 Oracle

Es un sistema de gestión de base de datos relacional (o RDBMS por el acrónimo en inglés de Relational Data Base Management System), fabricado por Oracle Corporation.

Tradicionalmente Oracle ha sido el SGBD por excelencia, considerado siempre como el más completo y robusto, destacando por:

- Soporte de transacciones.
- Estabilidad.
- Escalabilidad.
- Es multiplataforma.

También siempre ha sido considerado de los más caros, por lo que no se ha estandarizado su uso como otras aplicaciones.

Al igual que SQL Server, Oracle cuenta con una versión EXPRESS gratis para pequeñas instalaciones o usuarios personales.

4.3.1.4 Microsoft Access

Es un sistema de gestión de bases de datos Relacional creado por Microsoft para uso personal de pequeñas organizaciones.

Se ha ofrecido siempre como un componente de la suite Microsoft Office, aunque no se incluye en el paquete "básico".

Entre las principales funcionalidades destacadas podemos indicar que:

- Permite crear tablas de datos indexadas.
- Modificar tablas de datos.
- Relaciones entre tablas (creación de bases de datos relacionales).
- Creación de consultas y vistas.
- Consultas de acción (INSERT, DELETE, UPDATE).
- Formularios.
- Entorno de programación a través de VBA (Visual Basic for Applications).
- Llamadas a la API de windows.

4.3.1.5 PostgreSQL

Es un sistema de gestión de base de datos relacional orientado a objetos y libre, publicado bajo la licencia BSD (Berkeley Software Distribution).

Como muchos otros proyectos de código abierto, el desarrollo de PostgreSQL no es manejado por una empresa y/o persona, sino que es dirigido por una comunidad de desarrolladores que trabajan de forma libre y/o apoyada por organizaciones comerciales. La comunidad PostgreSQL es denominada el PGDG (PostgreSQL Global Development Group).

Sus principales características son:

- Alta concurrencia: mediante un sistema denominado MVCC (Acceso concurrente multiversión).
- Amplia variedad de tipos nativos: provee nativamente varios soportes.
- Ahorros considerables de costos de operación.
- Estabilidad y confiabilidad.

4.3.1.6 DB2

Este SGBD es propiedad de IBM, bajo la cual se comercializa el sistema de gestión de base de datos. Utiliza XML como motor, además el modelo que utiliza es el jerárquico en lugar del modelo relacional que utilizan otros gestores de bases de datos. Es el único de los gestores, antes comentados, que no es relacional.

Sus características más importantes son:

- Permite el manejo de objetos grandes (hasta 2 GB).
- La definición de datos y funciones por parte del usuario, el chequeo de integridad referencial.
- SQL recursivo, soporte multimedia: texto, imágenes, video, audio; queries paralelos, backup/recuperación on−line y offline.
- Recuperación utilizando accesos de sólo índices.
- Tablas de resumen.
- Tablas replicadas.
- Uniones hash.

Su principal desventaja es el precio, está dirigido solo a grandes empresas con necesidades de almacenamiento y procesamiento muy altas.

Al igual que SQL Server y Oracle, dispone de una versión EXPRESS gratis pero no de libre distribución.

Existen muchos más gestores de bases de datos en el mercado, pero estos como se ha comentado son los más usados.

Todos son relacionales (a excepción del BD2) y comparten por tanto el lenguaje de consulta (con algunas variantes propias) que es SQL.

4.3.2 Elección del sistema gestor de base de datos

Entre los sistemas gestores de bases de datos, explicados anteriormente, se ha elegido **MySQL**, debido a las siguientes razones:

- Bajo coste.
- Facilidad de configuración e instalación.
- Velocidad al realizar las operaciones.
- Documentación disponible.

4.4 Servidor de hosting

Este apartado trata sobre el procedimiento seguido para la elección del hosting web para nuestra aplicación, así como los pasos seguidos para su creación, configuración, y la creación de la base de datos asociada.

4.4.1 Elección del servidor

Para alojar nuestro servidor disponemos de varias posibilidades, entre ellas, hay la posibilidad de implementarlo en nuestro equipo personal (localhost), o bien, montarlo en la nube. Se ha descartado la primera opción, puesto que se tiene que dejar el ordenador encendido y conectado a internet las 24h, lo que supone un gran consumo de energía.

Se ha optado por alojar el servidor web en la nube, particularmente, se ha usado "**000webhost**" en el modo gratuito para que nuestra aplicación cumple con los requisitos de calidad con un coste razonable.

4.4.2 Configuración del servidor

4.4.2.1 Creación de una cuenta gratuita en "000webhost"

Primero, abrimos un navegador e introducimos la siguiente dirección web:

https://www.000webhost.com/

Figura 4.14 000webhost (1)

Posteriormente, le damos a "Sign Up for FREE" y nos redirige a otra página para darnos de alta.

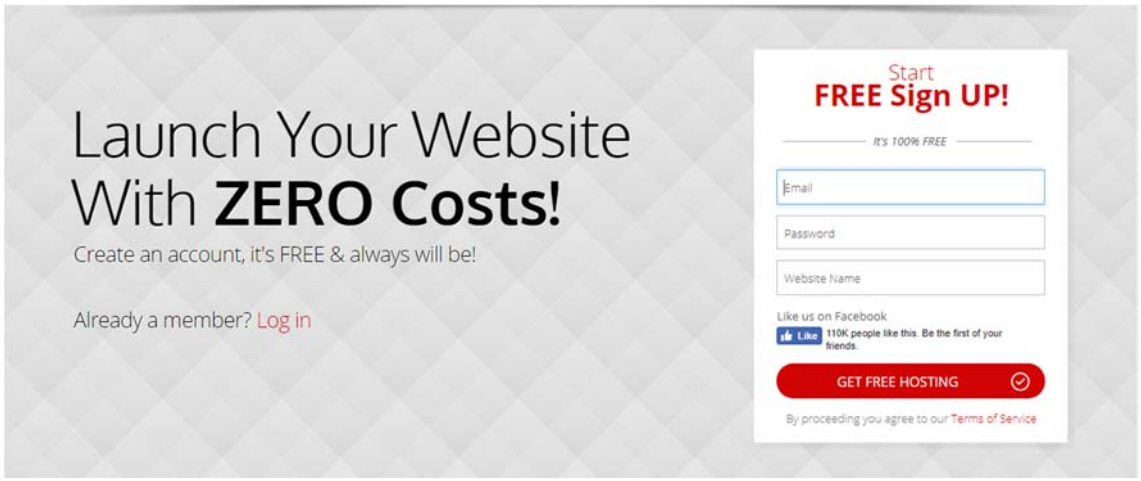

Figura 4.15 000webhost (2)

Después de registrarse, nos sale una pantalla en la cual podemos crear un nuevo sitio web.

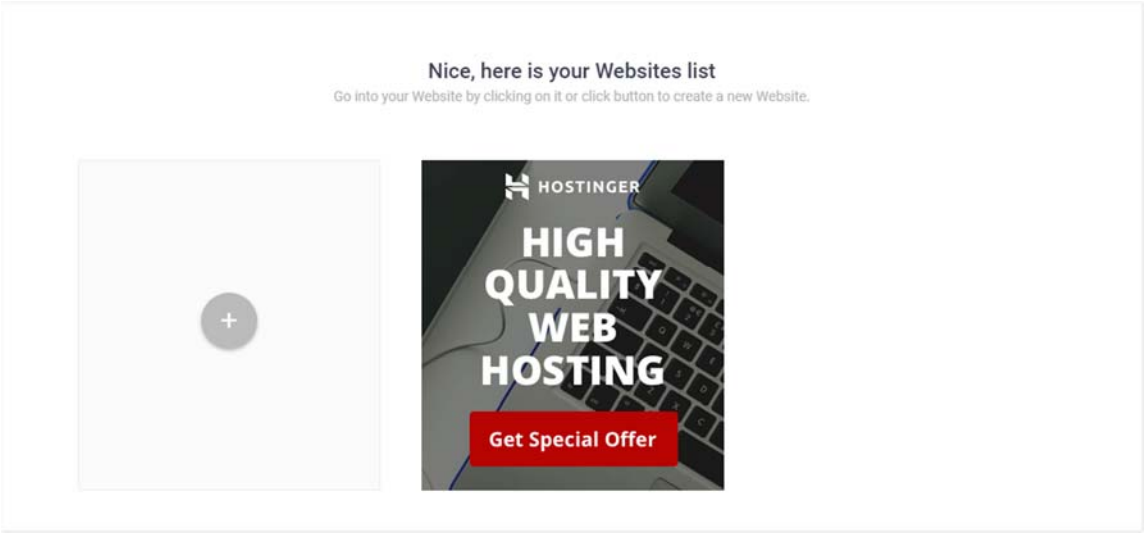

Figura 4.16 000webhost (3)

Le damos a crear nuevo sitio web. En seguida nos sale un formulario para introducir el nombre del sitio web y la contraseña para entrar.

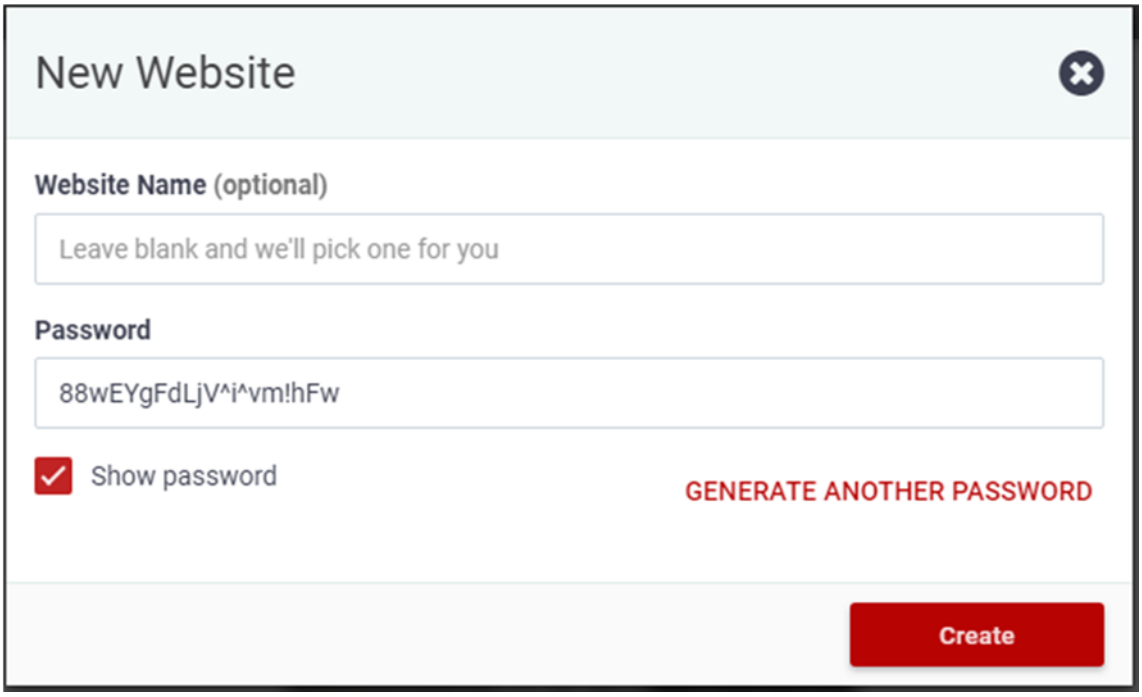

Figura 4.17 000webhost (4)

Después de darle a “Create”, nos redirige al panel de configuración de la página web.

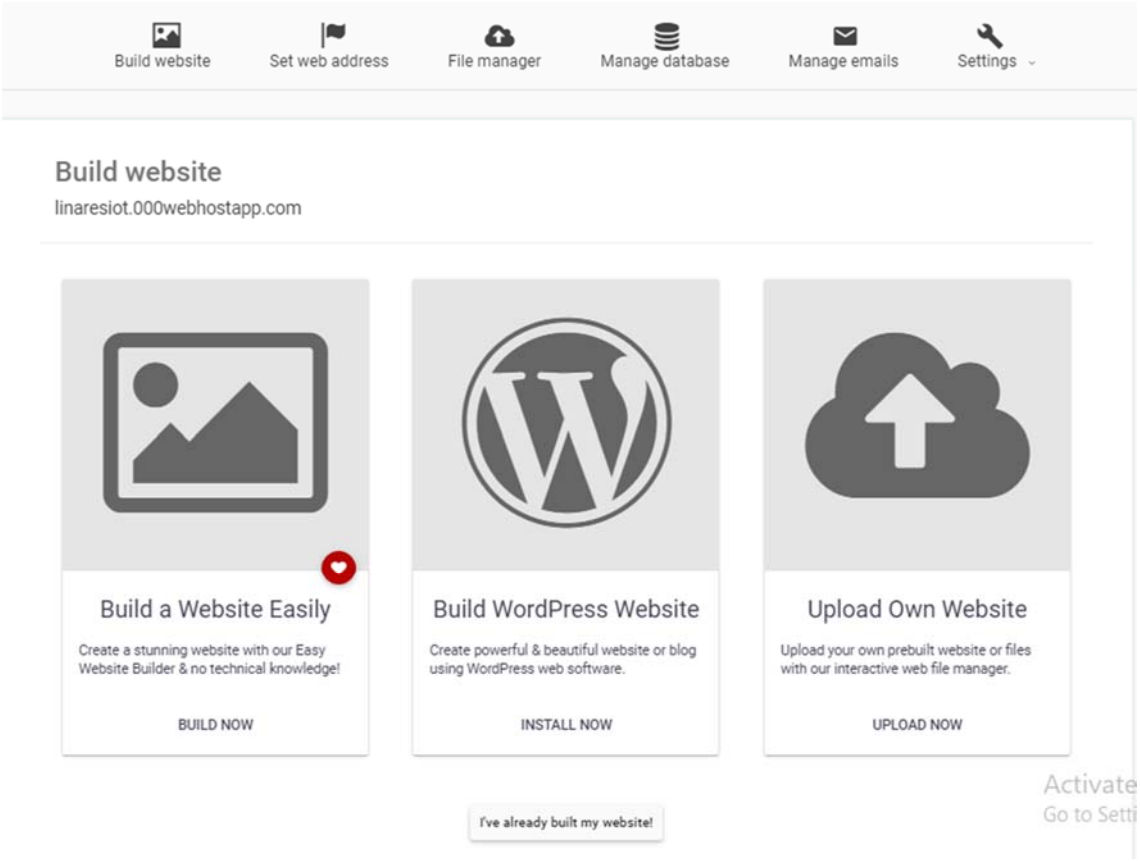

Figura 4.18 000webhost (5)

Vamos a usar el panel de control para crear y configurar la base de datos asociada al servidor.

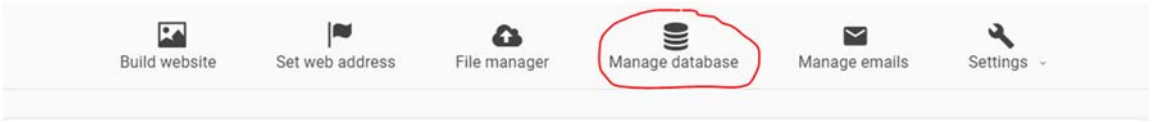

Figura 4.19 000webhost (6)

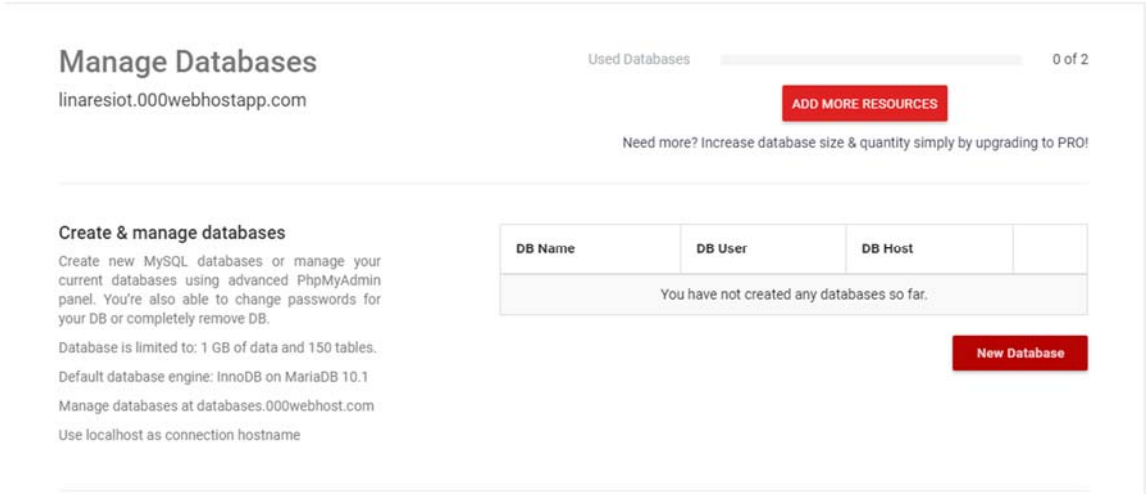

Figura 4.20 000webhost (7)

Creamos una nueva base de datos.

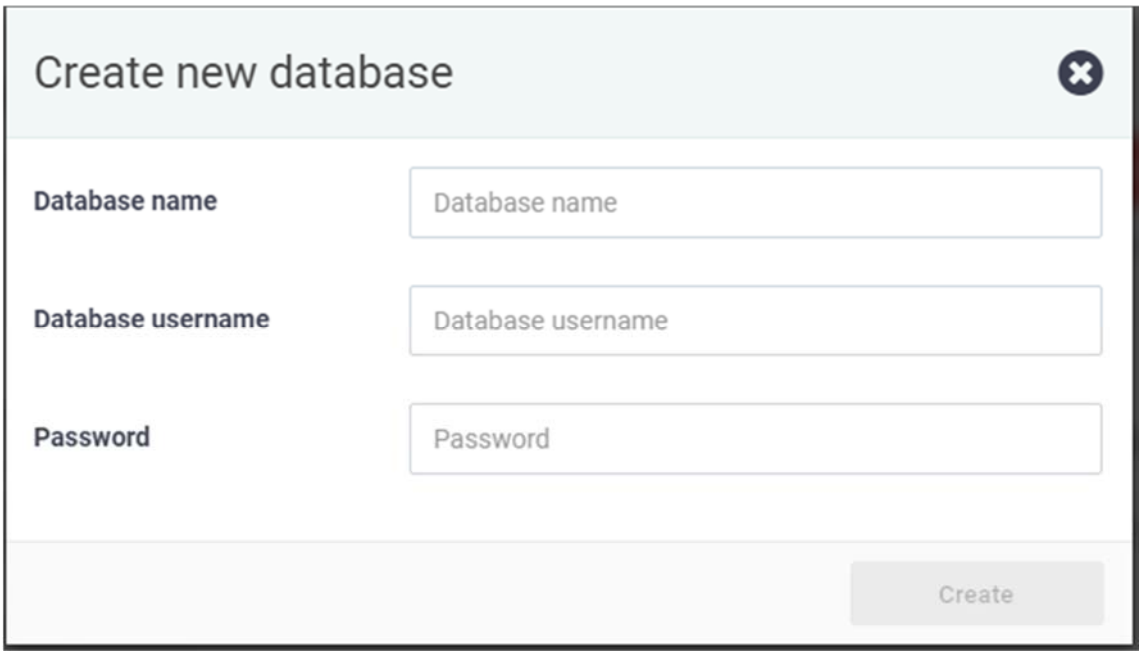

Figura 4.21 000webhost (8)

Después de crear la base de datos, podemos manejarla usando phpmyadmin.

Figura 4.22 000webhost (9)

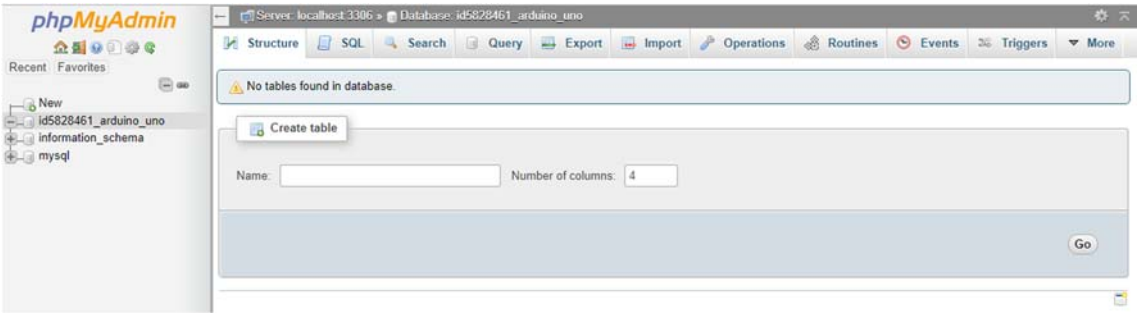

Figura 4.23 000webhost (10)

4.5 Servidor web

En el servidor web, vamos a necesitar implementar varios scripts que puedan extraer los datos enviados en las peticiones http de los nodos, procesarlos y guardarlos en la base de datos, así como, mostrarlos al usuario en varios formatos. También, vamos a dar la posibilidad de filtrar los datos según la fecha y la hora y poder buscar sólo las peticiones enviadas por un determinado nodo.

A continuación, se van a presentar todas las herramientas usadas para implementar la funcionalidad de nuestro programa.

4.5.1 Entornos de desarrollo IDE

La elección del entorno de desarrollo, IDE, está estrictamente relacionada con la tecnología que se va a utilizar, por lo tanto, se hará una introducción de las mismas. Los lenguajes de programación que vamos a usar son: PHP, HTML, Y Javascript.

Existen una multitud de IDEs que se adecuen a nuestras necesidades, tales como Netbeans, eclipse, visual studio, Monodevelop, etc. A continuación, vamos a presentar una breve introducción de los mismos. [14]

4.5.1.1 Netbeans

Muy difundido y conocido por una gran cantidad de programadores. Es una herramienta multilenguaje y multiplataforma en la cual podemos desarrollar software de calidad. Con él se puede crear aplicaciones web, y cuenta con plugins para trabajar en Android.

4.5.1.2 Eclipse

Es uno de los IDEs más usados, es multiplataforma y multilenguaje. Sin embargo, hay que tomar en cuenta que en Eclipse es necesario agregar varios plugins para que funcione al cien por cien.

4.5.1.3 Visual studio

Si lo que se desea es desarrollar para Windows 8 y sus versiones anteriores utilizando tecnología de Microsoft, la mejor opción es Visual Studio. Dentro de este IDE podremos desarrollar utilizando el entorno .net.

4.5.1.4 Monodevelop

Diseñado para Linux, Windows y Mac OS X. Soporta los lenguajes C#, Visual Basic, .Net, C / C + + y Vala. Por muchos programadores no es conocido, sin embargo, en su documentación, nos podemos hacer una idea de todo su potencial y comenzar a utilizar este IDE para .net.

4.5.1.5 Elección

Se ha optado por usar **eclipse**, ya que tiene una gran cantidad de bibliotecas y es un IDE muy utilizado en el sector.

4.5.2 Modelo cliente servidor

Las comunicaciones entre los nodos y el servidor web se van a efectuar siguiendo el modelo cliente-servidor, el cliente envía un mensaje HTTP (petición) y el servidor envía una respueta.

Para ello, los nodos deben saber la dirección IP y el puerto al que deben realizar la petición. Para ello, se ha asignado una IP y un puerto fijo al servidor.

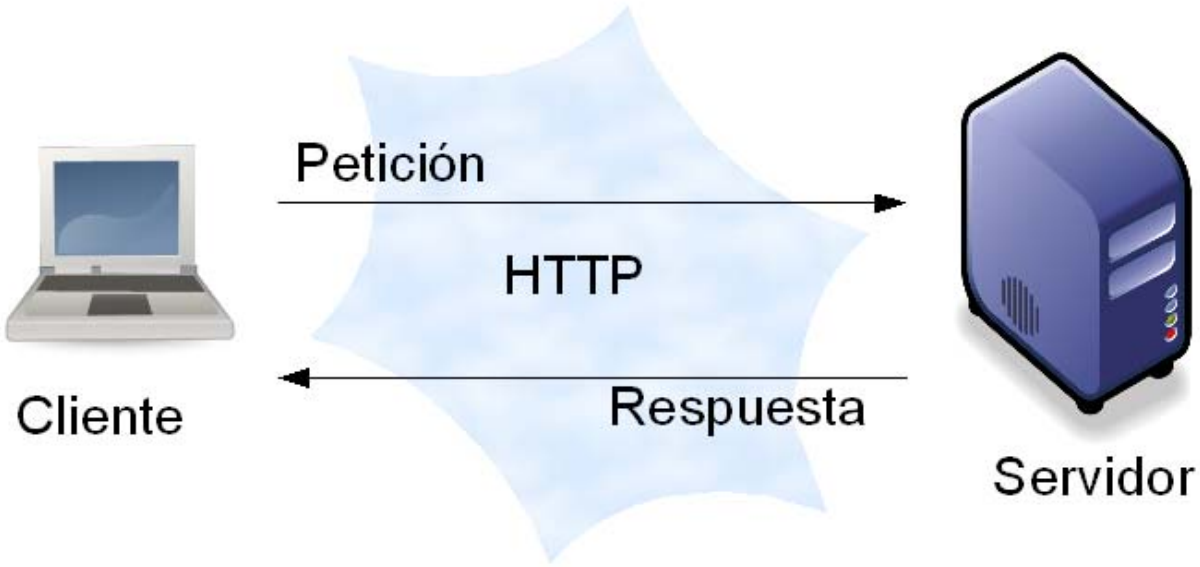

Figura 4.24 Esquema cliente-servidor

4.5.2.1 El protocolo HTTP

El protocolo que se va a usar en las interacciones entre los nodos y el servidor es el protocolo HTTP, Hyper Text Transfer Protocol, el cual pertenece al nivel de aplicación del modelo OSI.

Este protocolo está formado por una serie de cabeceras que nos permitan definir varios parámetros, tanto en la petición como en la respuesta, así como añadir información adicional. Además de texto, este protocolo nos permite transmitir archivos multimedia.

Las peticiones del protocolo HTTP pueden utilizar distintos métodos, siendo las principales: GET y POST. Ambos sirven para acceder a un recurso del servidor, la diferencia es que el primero codifica los datos adicionales en la URL, mientras que el segundo lo hace en un campo posterior a las cabeceras, denominado request body.

Las respuestas aparte de una serie de cabeceras, se definen por la primera línea, que nos da un código de estado, que nos indica la existencia de errores o no. Posteriormente, se añaden los datos requeridos en el request body.

En la interacción entre el cliente y el servidor se va a usar el método POST, ya que el método GET sólo nos permita transmitir un número limitado de datos, según el navegador web sea capaz de leer. Igualmente, el método POST añade una capa de seguridad al no mostrar en la url los datos a transmitir. A continuación, se mostrará un ejemplo:

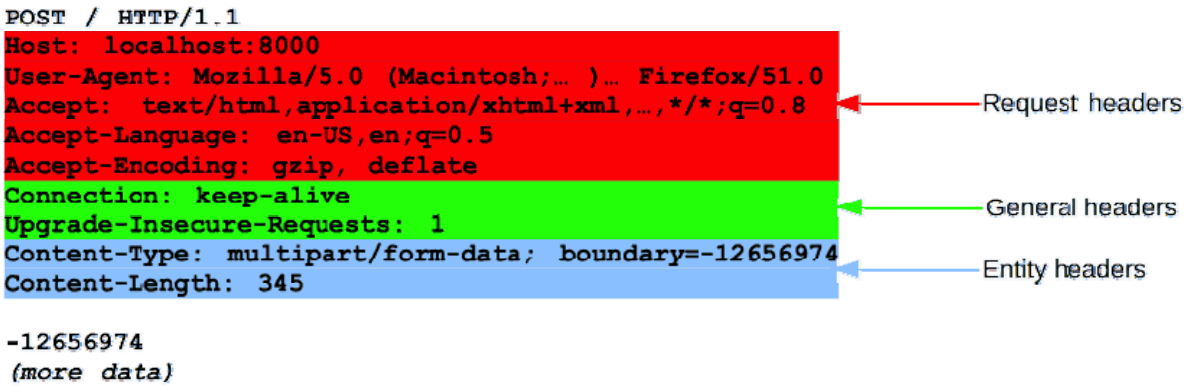

Figura 4.25 Ejemplo de mensaje de petición tipo POST

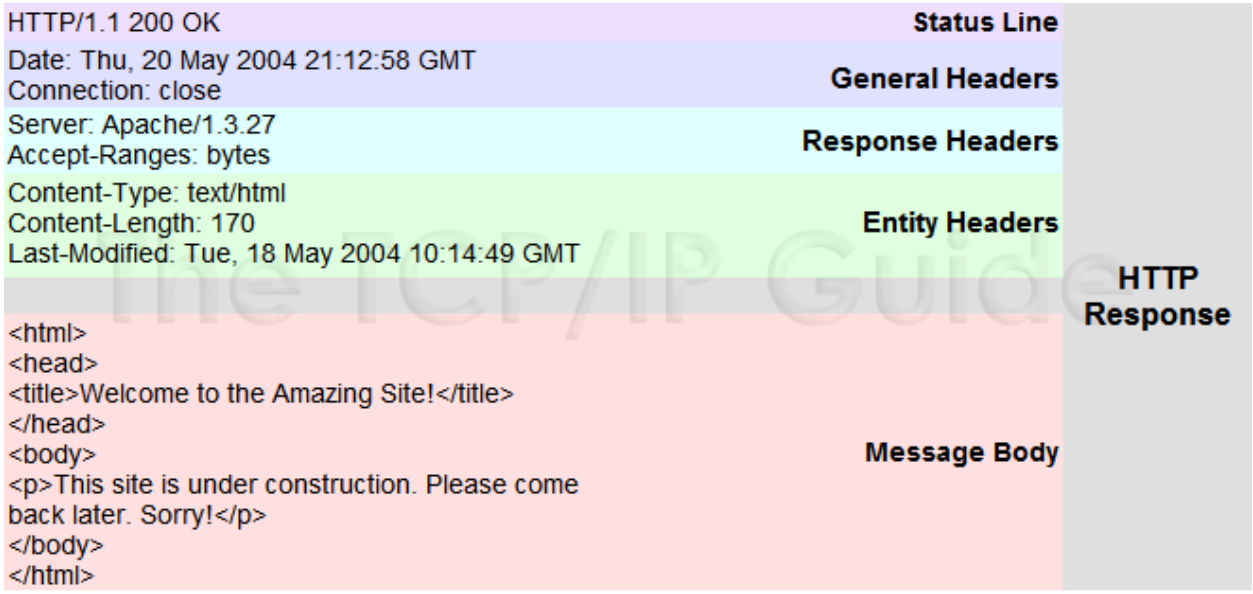

Figura 4.26 Ejemplo de mensaje de respuesta HTTP

4.5.3 Librerías de representación

Una de las funcionalidades de nuestra aplicación web es la representación gráfica de los datos leídos por los sensores. Para ello, se ha decidido utilizar una librería JavaScript que nos permita representar los datos de forma sencilla y con una apariencia visual atractiva. A continuación, se representarán las soluciones encontradas: [15] [16]

4.5.3.1 Chart.js

Una de las opciones más populares. Funciona sobre el elemento <canvas> de HTML5, que permite dibujar figuras a través de JavaScript. Ofrece seis tipos de gráficos diferentes, todos responsivos e interactivos (al tocar un sector del gráfico o pasar el mouse por encima se genera un tooltip con más información, también, podemos definir eventos según nuestras propias necesidades).

Figura 4.27 Representación gráfica de chart.js

4.5.3.2 Chartist.js

Potente biblioteca basada en SVG (Scalable Vector Graphics), un formato de imagen escalable y flexible. Ofrece una buena variedad de tipos de gráficos, con características de interactividad y animación, que podemos personalizar a través de Sass (Syntactically Awesome Stylesheets), un preprocesador de CSS. Incluso podemos aplicarles media queries si necesitamos adaptarlos a múltiples dispositivos. Un detalle interesante es que no requiere de bibliotecas externas (como jQuery) para funcionar, ahorrando recursos. Como aspecto negativo, varias características no son soportadas por Internet Explorer.

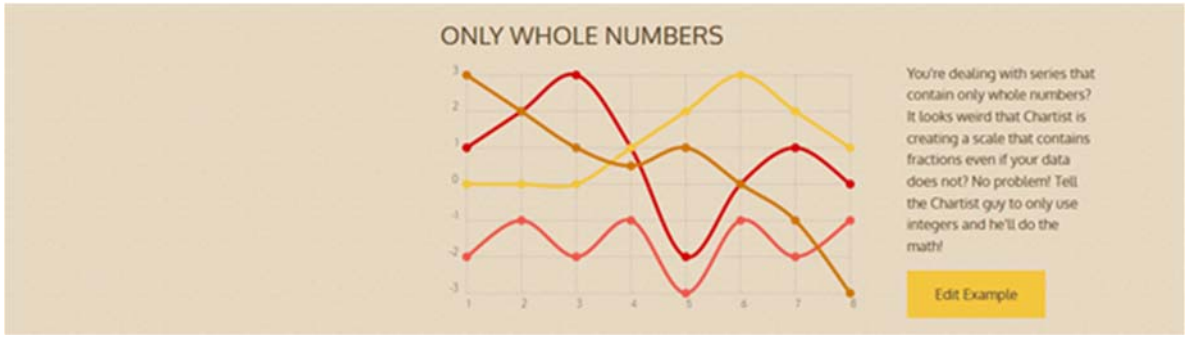

Figura 4.28 Representación gráfica de chartist.js

4.5.3.3 Google Chart

La opción de Google está basada en HTML5 y SVG, aunque puede reemplazar este lenguaje de gráficos con VML (Vector Markup Language) para versiones antiguas de Internet Explorer. Esto garantiza una buena compatibilidad para la mayoría de las plataformas. Hay una enorme cantidad de gráficos para elegir (incluso basados en mapas), todos ellos animados e interactivos. En comparación con otras herramientas similares, es muy fácil de utilizar, incluso para personas sin conocimientos de desarrollo. Los gráficos se pueden exportar a imágenes en PNG e incluso a otras aplicaciones.

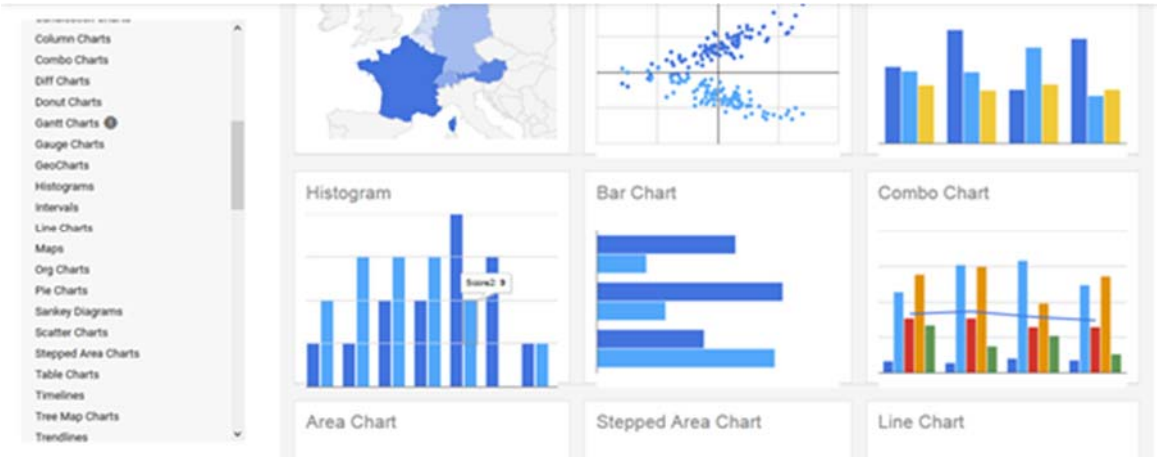

Figura 4.29 Representación gráfica de Google chart

4.5.3.4 D3.js

A diferencia de las herramientas ya mencionadas, D3.js no es una simple herramienta para generar gráficos: es una biblioteca para manipular elementos del DOM a partir de conjuntos de datos. La mayoría de las bibliotecas generan gráficos con una estructura predefinida, listos para usar, a partir de los datos que ingresamos. D3.js, sin embargo, nos permite elegir y manipular los elementos que consideremos más adecuados para representar nuestros datos, así sea una tabla, una imagen en SVG o un elemento . Esto garantiza una enorme libertad y control sobre el resultado final, aunque también convierte a esta herramienta en la más complicada de las otras alternativas que se han mencionado anteriormente. Una buena opción para cuando se tienen grandes volúmenes de datos y/o las demás herramientas no pueden ofrecernos el diseño visual que queremos.

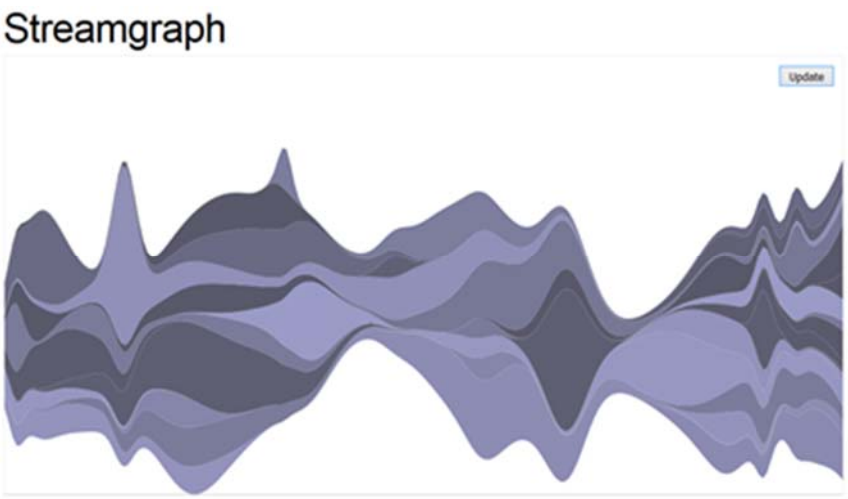

Figura 4.30 Representación gráfica de D3.js

4.5.3.5 Highcharts

Esta librería permite crear gráficas interactivas fácilmente para proyectos web. Usada por muchos desarrolladores, Highcharts es la solución más rápida y flexible. Por la cantidad de funciones que ofrece, esta librería es de pago, pero si el proyecto no usa la librería con fines comerciales, se puede usar de manera gratuita bajo la licencia de Creative Commons Attribution-NonCommercial 3.0.

- Usa SVG.
- Soporte para multiples exploradores.
- Altamente personalizable.
- Muchos gráficos.

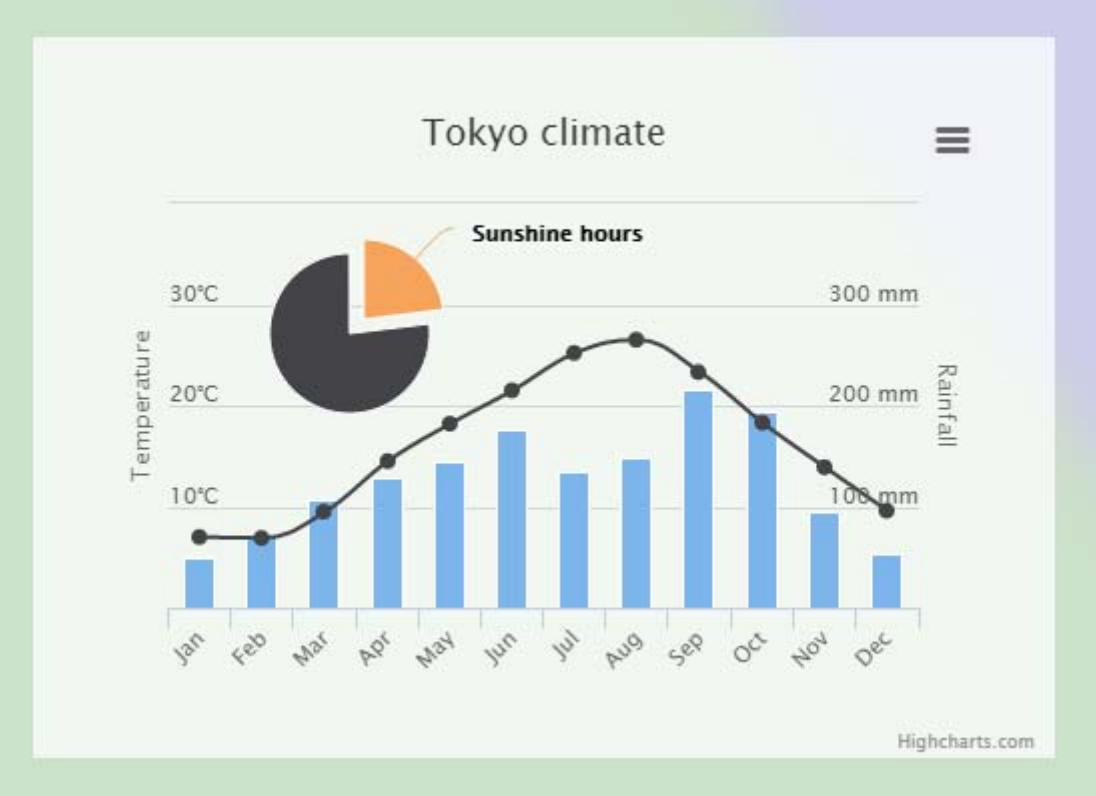

Figura 4.31 Representación gráfica de Highcharts

4.5.3.6 Elección

Después de ver las diferentes alternativas disponibles para la representación de datos de los sensores de forma gráfica, se ha decidido usar **HighCharts** por los siguientes motivos:

- Fácil de usar.
- Intuitiva.
- Nos permite hacer un filtro de los resultados en función de la fecha y hora del día.
- Gratuita.
- Nos da la posibilidad de descarga en varios formatos.

CAPÍTULO 5. IMPLEMENTACIÓN

En este capítulo, se va a mostrar, en detalle, el código usado en el trabajo fin de grado, así como los pasos seguidos para realizar la captura y el almacenamiento de información medioambiental.

5.1 Programación del NodeMCU

En este apartado, se va a mostrar el código que se está ejecutando en los nodos.

5.1.1 Adaptación de sensores

En esta sección, se va a explicar cómo programar los sensores, para que puedan leer información medioambiental. Para ello, vamos a hacer uso de algunas librerías, que mencionamos a continuación.

5.1.1.1 Sensor DHT11

Para el uso de este sensor, se va a hacer uso de la librería "**DHT.h**", la cual nos permita leer de forma sencilla los valores de temperatura y humedad. A continuación, vamos a mostrar el código usado en los nodos.

Primero, se ha de incluir la librería <DHT.h> y definir el pin (en nuestro caso, el pin D4) donde se van a recibir los datos:

```
#include <DHT11.h>
#include <ESP8266WiFi.h>

DHT11 dht11(D4);
```

Figura 5.1 Programación DHT11 (1)

Seguidamente, se ha de aplicar el método "read" de la librería DHT11 para leer la temperatura y la humedad:

```
int err;
if ((err = dht11.read(hum, temp)) == 0)
{
  Serial.print("Temperatura: ");
  Serial.print(temp);
  Serial.print(" Humedad: ");
  Serial.print(hum);
  Serial.println();
}
else
{
  Serial.println();
  Serial.print("Error Num :");
  Serial.print(err);
  Serial.println();
}
```

Figura 5.2 Programación DHT11 (2)

Vamos a explicar el código de la **figura 5.2**. Si el error vale 0, significa que el sensor DHT11 ha realizado la lectura con éxito. En caso contrario, nos da un error en la lectura, especificando el tipo de error.

A continuación, se va a mostrar un esquema de cómo está conectado el NodeMCU con el sensor de temperatura y humedad dht11:

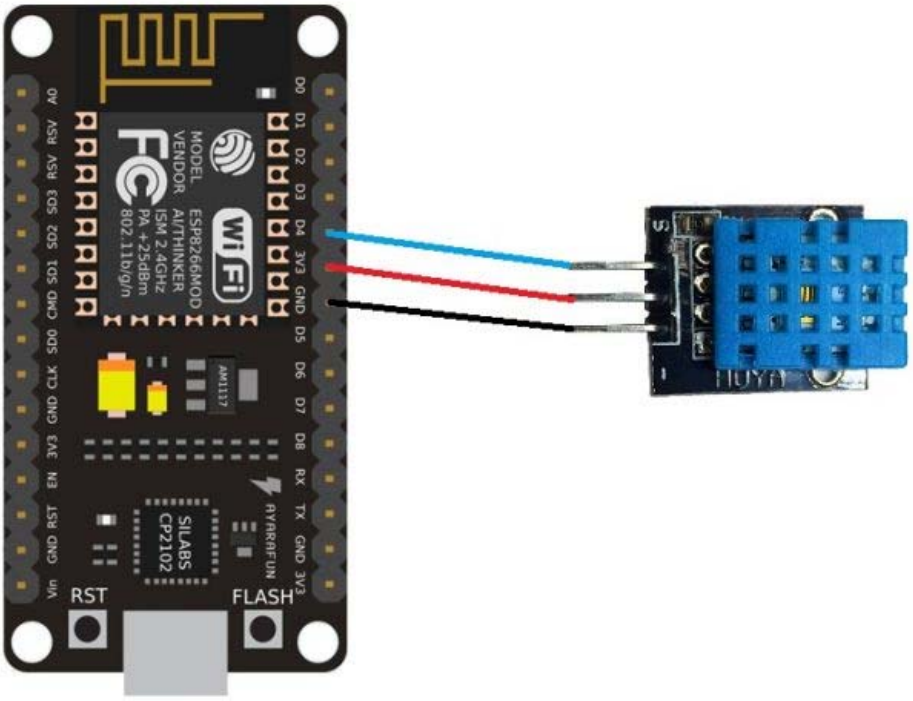

Figura 5.3 Conexión NodeMCU y DHT11

5.1.1.2 El sensor de luminosidad LDR

Como se ha mencionado en el **capítulo 4**, para obtener el valor de la luminosidad, se va a hacer uso de una LDR.

Vamos a tener que aplicar una adaptación para poder obtener el nivel de luz incidente sobre la célula de la LDR.

Matemáticamente, la relación entre la iluminancia y la resistencia de una LDR sigue una función potencial.

$$\frac{I}{I_0} = (\frac{R}{R_0})^{-gamma}$$

Siendo R0 la resistencia a una intensidad I0, ambas conocidas.

La constante gamma es la pendiente de la gráfica logarítmica, o la pérdida de resistencia por década. Su valor es típicamente entre 0.5 y 0.8.

A continuación, se presenta el esquema eléctrico:

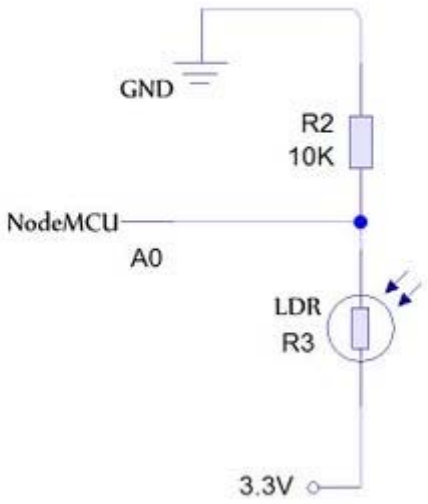

Figura 5.4 Conexión NodeMCU y LDR (1)

Y el montaje en una protoboard:

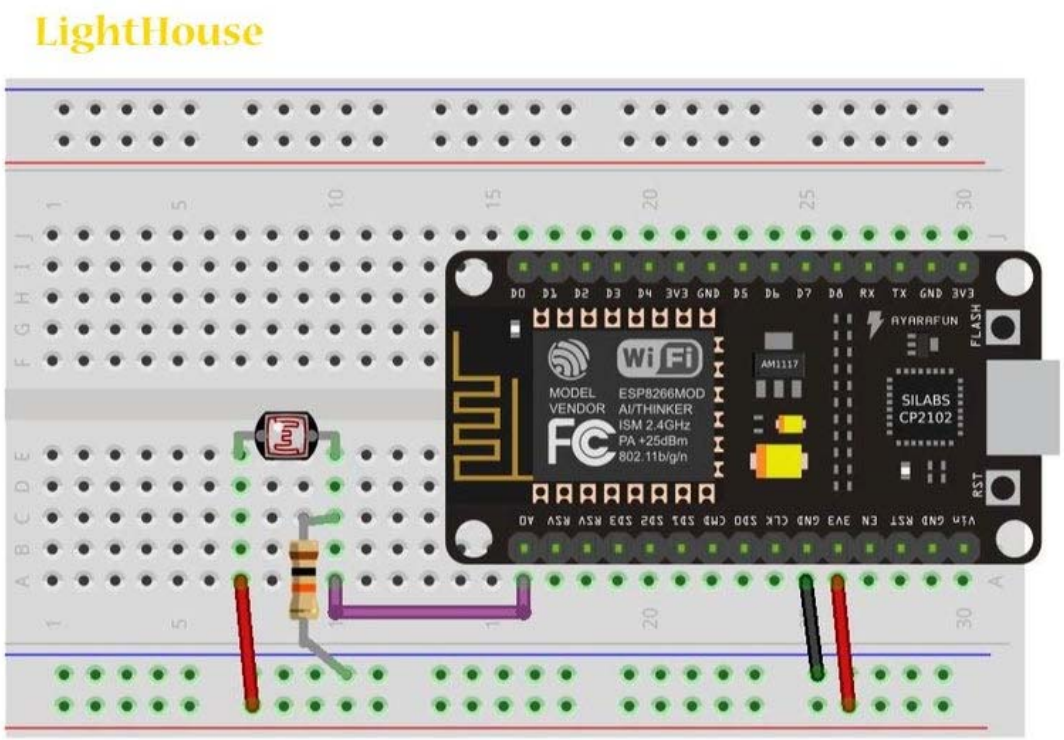

Figura 5.5 Conexión NodeMCU y LDR (2)

Finalmente, se va a presentar el código empleado en el nodo para la lectura de iluminación recibida.

```
const long A = 1000;     //Resistencia en oscuridad en KΩ
const int B = 15;        //Resistencia a la luz (10 Lux) en KΩ
const int Rc = 10;       //Resistencia calibracion en KΩ
const int LDRPin = A0;   //Pin del LDR

int V;
int lum;
```

Figura 5.6 Programción LDR (1)

En la **figura 5.6**, se han declarado todas las variables necesarias para la lectura de la iluminación.

```
V = analogRead(LDRPin);
lum = ((long)V*A*10)/((long)B*Rc*(1024-V));
Serial.print("Luminosidad: ");
Serial.print(lum);
Serial.println();
```

Figura 5.7 Programción LDR (2)

En la primera línea, hacemos uso del método “analogRead”, para realizar la lectura del pin analógico A0 del NodeMCU. Luego, aplicamos una fórmula para la obtención de la iluminación recibida.

5.1.2 Conexión inalámbrica a una red

Para enviar los datos medioambientales capturados al servidor, debemos estar conectados a internet. Como se ha mencionado en el **capítulo 4**, vamos a usar la tecnología WIFI para conectarnos a internet. A continuación, vamos a explicar el código usado.

```
#include <DHT11.h>
#include <ESP8266WiFi.h>

DHT11 dht11(D4);

const char* ssid     = "xxxxxxxx";
const char* password = "xxxxxxxx";

const char* host = "jaeniot.000webhostapp.com";
```

Figura 5.8 Conexión WIFI (1)

En la **figura 5.8**, incluimos la librería "ESP8266WiFi", que tiene los métodos necesarios para conectarse a internet. También, definimos la red al que queremos conectar (ssid), y su clave de acceso (password).

```
Serial.println();
Serial.println();
Serial.print("Connecting to ");
Serial.println(ssid);

WiFi.begin(ssid, password);

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

Serial.println("");
Serial.println("WiFi connected");
Serial.println("IP address: ");
Serial.println(WiFi.localIP());
```

Figura 5.9 Conexión WIFI (2)

Como se ve en la **figura 5.9**, usando el método "begin" de la librería "ESP8266WiFi", nos conectamos a internet.

5.1.3 Conexión al servidor y envió de peticiones HTTP

En esta sección, vamos a conectarnos a un servidor web, y vamos a enviar peticiones HTTP con los datos de temperatura, humedad, luminosidad, e identificador único del nodo.

```
const char* host = "jaeniot.000webhostapp.com";
```

Figura 5.10 Servidor web

```
Serial.print("connecting to ");
Serial.println(host);

// Use WiFiClient class to create TCP connections
WiFiClient client;
const int httpPort = 80;
if (!client.connect(host, httpPort)) {
  Serial.println("connection failed");
  return;
}
```

Figura 5.11 Conexión al servidor web

Como podemos ver en la **figura 5.11**, primero definimos un cliente WIFI y el puerto 80. Luego, mediante el método "connect" de la librería "ESP8266WiFi" nos conectamos al servidor en el puerto 80.

```
String url = "/cliente/sensores.php";
String dato0 = "?Temperatura=";
String dato1 = "&Humedad=";
String dato2 = "&Luminosidad=";
String dato3 = "&IDN=";

Serial.print("Requesting URL: ");
Serial.println(url);
```

Figura 5.12 URL petición HTTP

En la **figura 5.12**, definimos la URL de la petición HTTP.

```
client.print(String("GET ") + url + dato0 + temp + dato1 + hum + dato2 + lum + dato3 + idn + " HTTP/1.1\r\n" +
             "Host: " + host + "\r\n" +
             "Connection: close\r\n\r\n");
unsigned long timeout = millis();
while (client.available() == 0) {
  if (millis() - timeout > 5000) {
    Serial.println(">>> Client Timeout !");
    client.stop();
    return;
  }
}
```

Figura 5.13 Envió petición HTTP

Como podemos apreciar en la **figura 5.13**, la URL está compuesta por 4 variables (temperatura, humedad, luminosidad, e identificador del nodo), y el tipo de petición HTTP es GET.

Cuando el cliente no está disponible, salta un mensaje de error y se para el cliente.

```
while (client.available()) {
  String line = client.readStringUntil('\r');
  Serial.print(line);
}
```

Figura 5.14 Respuesta del servidor

Mediante el código de la **figura 5.14**, imprimimos por pantalla todo lo que nos llega del servidor.

```
Serial.println();
Serial.println("closing connection");

Serial.println("I'm awake, but I'm going into deep sleep mode for 30 seconds");
ESP.deepSleep(30e6);
```

Figura 5.15 Cerrar la conexión con el servidor y entrar en Deep Sleep

Finalmente, se cierra la conexión con el servidor, y el nodo entra en el modo "Deep Sleep". Más adelante en la memoria, vamos a hablar con detalle sobre aspectos de consumo energético de los nodos.

5.1.4 Deep sleep

Uno de los objetivos de nuestro TFG es ahorrar en el consumo energético. Para ello, vamos a usar el modo "Deep sleep" cuando el cliente no está transmitiendo datos al servidor.

El primer paso que tenemos que hacer es conectar el pin reset (RST) con el GPIO16 (D0).

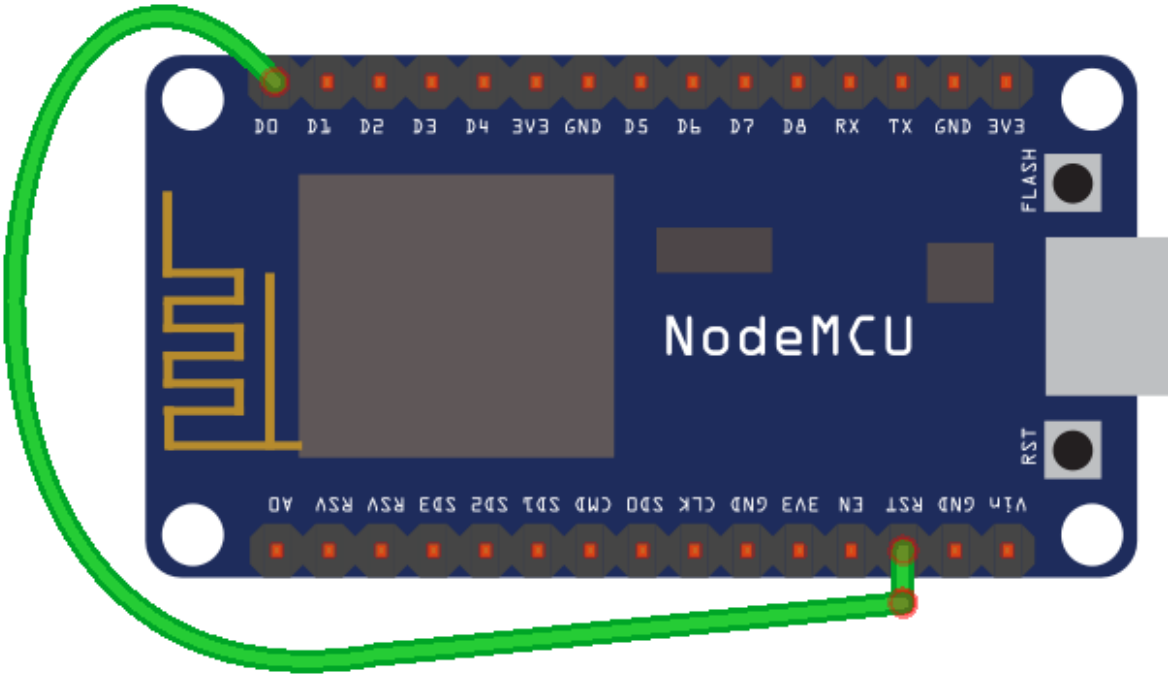

Figura 5.16 NodeMCU Deep Sleep

Si echamos un vistazo al pinout del NodeMCU, podemos saber que el pin GPIO16 tiene una función especial, mediante la cual podemos despertar el nodo si está en el modo “Deep sleep”.

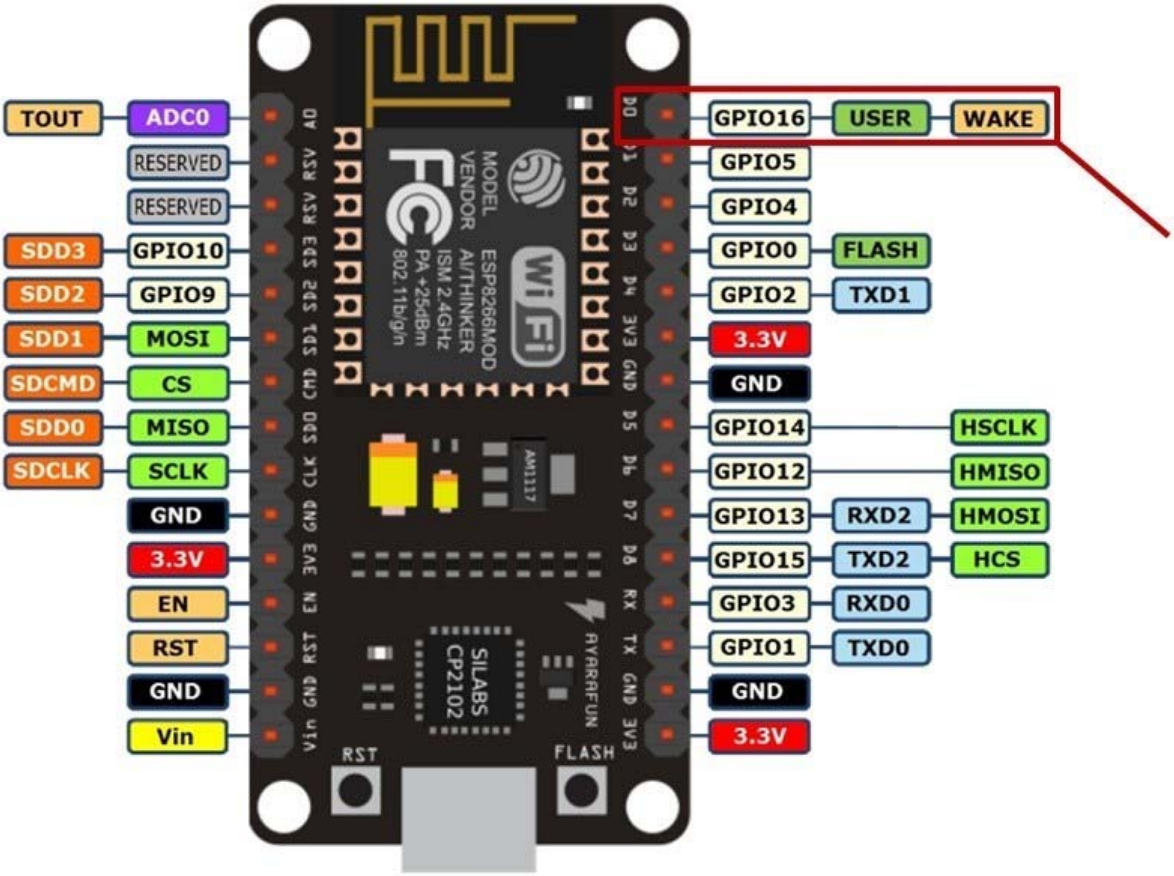

Figura 5.17 Pinout NodeMCU

El pin de RST está siempre a nivel alto mientras el NodeMCU está funcionando. Sin embargo, cuando el pin RST está a nivel bajo, se reinicia el nodo.

Si establecemos un temporizador con el microcontrolador ESP8266, una vez se vence el temporizador, el GPIO16 envía una señal de nivel bajo. Esto significa que conectando el pin GPIO16 con el pin RST, podemos despertar el nodo cada vez que el temporizador venza.

A nivel de código, ya hemos explicado, en la **sección 5.1.3**, cómo se puede dormir un NodeMCU en el modo “Deep Sleep”.

5.2 Servidor

En el servidor, se ha intentado seguir, en gran medida, el paradigma MVC (Modelo Vista Controlador). En este apartado, vamos a explicar con detalle la implementación de estos tres aspectos.

5.2.1 Modelo

Esta sección trata sobre los aspectos relacionados con el diseño de la base de datos, denominada "tfg". A continuación, se presentarán todas las tablas que componen la base de datos.

5.2.1.1 Tabla users

En está tabla, tenemos toda la información relacionada con los usuarios del sistema. A continuación, se muestra la estructura de la tabla.

Nombre	Tipo
id	int(10)
username	varchar(100)
email	varchar(100)
user_type	varchar(100)
password	varchar(100)
fecha_alta	timestamp(6)

Tabla 5.1 users

Como podemos ver en la **figura 5.18**, la tabla "users" está constituida por seis campos:

- **Id**: clave primaria.
- **Username**: nombre de usuario.
- **Email**: correo electrónico.
- **User_type**: usuario normal u admin.
- **Password**: contraseña.
- **Fecha_alta**: se asigna de forma automática cuando el usuario se registra.

5.2.1.2 Tabla nodos

En está tabla, tenemos toda la información relacionada con los nodos del sistema. A continuación, se muestra la estructura de la tabla.

Nombre	Tipo
id	int(10)
nombre	varchar(100)
id_unico	varchar(100)
fecha_alta	timestamp(4)

Tabla 5.2 nodos

Como podemos ver en la **figura 5.19**, la tabla "users" está constituida por cuatro campos:

- **Id**: clave primaria de la tabla.
- **Nombre**: nombre asignado al nodo.
- **Id_unico**: identificador único asignado por el administrador a cada nodo del sistema.
- **Fecha_alta**: asignado automáticamente cuando el administrador dé de alta al nodo.

5.8.1.3 Tabla monitorización

En está tabla, se guardan todos los datos enviados por los nodos. A continuación, se muestra la estructura de la tabla.

Nombre	Tipo
id	int(20)
temperatura	float
humedad	float
luminosidad	float
date	date
time	time
id_nodo	varchar(15)
chart_time	timestamp

Tabla 5.3 monitorizacion

Como podemos ver en la **figura 5.20**, la tabla "monitorización" está constituida por ocho campos:

- **Id**: clave primaria.
- **Temperatura**: temperatura ambiente enviada por los nodos.
- **Humedad**: humedad ambiente enviada por los nodos.
- **Luminosidad**: nivel de luz enviado por los nodos.
- **Date**: fecha de envió de los datos.
- **Time**: instante de envío de los datos.
- **Id_nodo**: identificador único del nodo que ha enviado los datos.
- **Chart_time**: campo reservado para usarse exclusivamente por la librería HighChart.

5.2.2 Controladores y vistas

En el servidor, hemos mezclado código PHP con código HTML. Así que, las vistas y los controladores estarán en el mismo archivo. Cabe destacar, que no estamos usando ningún framework en particular.

En esta sección, vamos a citar las funciones importantes involucradas en las diferentes interacciones del servidor.

5.2.2.1 functions.php

En este apartado, vamos a describir la funcionalidad de las diferentes funciones definidas en el script functions.php.

5.2.2.1.1 Función register

Está función permite registrar un usuario en la base de datos. A continuación, vamos a presentar el código usado, y vamos explicándolo por partes.

```php
// función para registrar un usuario
function register(){
    // llamar a estas variables con la palabra clave "global" para que estén disponibles en la función
    global $db, $errors, $username, $email;

    // recibir los valores enviados desde el formulario. llamar a la función e()
    // definida más abajo para que podamos usar estos valores en sentencias sql.
    $username    =  e($_POST['username']);
    $email       =  e($_POST['email']);
    $password_1  =  e($_POST['password_1']);
    $password_2  =  e($_POST['password_2']);

    // comoprobar si el usuario no existe
    $nuevo_usuario = mysqli_query($db, "select username from users where username='$username'");
    $nuevo_usuario2 = mysqli_query($db, "select email from users where email='$email'");

    if(mysqli_num_rows($nuevo_usuario)>0)
    {
        array_push($errors, "El nombre de usuario ya existe");
    }
    elseif(mysqli_num_rows($nuevo_usuario2)>0)
    {
        array_push($errors, "El email ya existe");
    }
    // Si no esta registrado el nodo continua el script
```

Figura 5.18 funcion register (1)

El primer paso es declarar las variables: $db, $errors, $username, $email, como variables globales para que estén disponibles en toda la función.

El segundo paso, es recibir los valores enviados desde el formulario para la creación de un nuevo usuario, y mediante la función “e”, eliminamos los caracteres especiales de los valores recibidos para poder usarlos en una sentencia SQL.

Luego, comprobamos si el usuario que intentemos crear, existe o no en la base de datos, mediante el nombre de usuario y email.

```php
// Si no esta registrado el nodo continua el script
else
{

// validación del formulario: asegurar que el formulario se ha rellenado correctamente
if (empty($username)) {
    array_push($errors, "Username is required");
}
if (empty($email)) {
    array_push($errors, "Email is required");
}
if (empty($password_1)) {
    array_push($errors, "Password is required");
}
if ($password_1 != $password_2) {
    array_push($errors, "The two passwords do not match");
}
```

Figura 5.19 función register (2)

Si el usuario no existe en la base de datos, pasamos a la validación del formulario. Si dejamos algún campo del formulario sin rellenar, nos da un aviso de que este campo es requerido. También, nos da un mensaje de error si las dos contraseñas no coinciden.

```php
// registrar el usuario si no hay errores en el formulario
if (count($errors) == 0) {
    $password = md5($password_1);//encriptar la contraseña antes de guardarla en la base de datos

    if (isset($_POST['user_type'])) {
        $user_type = e($_POST['user_type']);
        $query = "INSERT INTO users (username, email, user_type, password)
                  VALUES('$username', '$email', '$user_type', '$password')";
        mysqli_query($db, $query);
        $_SESSION['success']  = "Nuevo usuario creado exitosamente!!";
        header('location: users.php');
    }else{
        $query = "INSERT INTO users (username, email, user_type, password)
                  VALUES('$username', '$email', 'user', '$password')";
        mysqli_query($db, $query);

        // obtener el id del usuario creado
        $logged_in_user_id = mysqli_insert_id($db);

        $_SESSION['user'] = getUserById($logged_in_user_id); // poner usuario registrado en sesión
        $_SESSION['success']  = "Está conectado ahora!!";
        header('location: index.php');
    }
}
}
```

Figura 5.20 función register (3)

Si el formulario está libre de errores, guardamos los datos del nuevo usuario en la base de datos, encriptando la contraseña antes de guardarla. Si la creación del nuevo usuario se ha hecho mediante un administrador, se redirige a la vista “users.php”. Mientras que, si la creación de la nueva cuenta se ha hecho mediante el propio usuario, se redirige a la página principal (index.php).

5.2.2.1.2 Función insertar

La función "insertar" se encarga de registrar un nuevo nodo en el sistema. Sigue los mismos pasos que la función "register" explicada en el **apartado 5.2.2.1.1**.

5.2.2.1.3 Función actualizar

La función "actualizar" es la encargada de modificar los datos de un usuario existente. Mediante la función $_REQUEST, se busca al usuario por su id, y nos da la posibilidad de modificar sus datos. También, sigue los mismos pasos que las dos funciones anteriores.

5.2.2.1.4 Función actualizarNodo

De igual forma que la función "actualizar", explicada en el **apartado 5.2.2.1.3,** la función "actualizarNodo" permite modificar los datos de un nodo existente en el sistema.

5.2.2.2 sensoresWS.php

Este script es el encargado de recibir los valores enviados desde los nodos y guardarlos en la base de datos. A continuación, vamos a explicar el código usado.

```
// Conexión con la base de datos
$con = mysqli_connect($dbhost, $dbuser, $dbpass, $dbname);
 // Leemos los valores que nos llegan por GET
$valor1 = mysqli_real_escape_string($con, $_GET['temp']);
$valor2 = mysqli_real_escape_string($con, $_GET['hum']);
$valor3 = mysqli_real_escape_string($con, $_GET['lum']);
// Esta es la instrucción para insertar los valores
$query = "INSERT INTO monitorizacion (temperatura, humedad, luminosidad)
        VALUES('$valor1', '$valor2', 'valor3')";
// Ejecutamos la instrucción
mysqli_query($con, $query);
mysqli_close($con);
```

Figura 5.21 sensoresWS.php

Como se puede ver en la **figura 5.21**, el primer paso es conectarnos a la base de datos. Luego, mediante la función "mysqli_real_escape_string", eliminamos los caracteres especiales en los valores de temperatura, humedad, y luminosidad recibidos desde los nodos, y creamos la sentencia sql y la ejecutamos, y finalmente cerramos la conexión con la base de datos.

5.2.2.3 register.php

Este script contiene un formulario HTML, mediante el cual un nuevo usuario puede registrarse y obtener las credenciales de acceso (usuario y contraseña).

5.2.2.4 login.php

Este script contiene un formulario HTML, mediante el cual un usuario, con las credenciales de acceso, puede autenticase.

5.2.2.5 nodos.php

Esta vista contiene una tabla con todos los nodos dados de alta por el administrador del sistema. Para un usuario normal, con permisos limitados, la tabla solo contiene tres campos: nombre del nodo, su identificador único, y su fecha de alta. Mientras que, para un usuario admin, la tabla contiene dos campos más, uno para editar los datos del nodo, y el otro para eliminar el nodo definitivamente.

5.2.2.6 insert.php

Este script permite dar de alta a un nuevo nodo en la red de sensores inalámbricos. Contiene un formulario HTML con dos campos: nombre del nodo y su identificador único. Esta vista es exclusivamente para un usuario admin.

5.2.2.7 deleteNode.php

Este script permite borrar un nodo del sistema.

5.2.2.8 editNode.php

Este script permite editar los datos de un nodo de la red de sensores inalámbrica. Contiene un formulario HTML.

5.2.2.9 users.php

Esta vista contiene una tabla con todos los usuarios registrados en el sistema. Esta tabla contiene los siguientes campos: el nombre de usuario, su email, tipo de cuenta: usuario normal o admin, su fecha de alta, y dos campos más para editar sus datos y para eliminar su cuenta. Esta vista solo la pueden ver los usuarios con todos los permisos (admin). También, da la posibilidad de crear una nueva cuenta.

5.2.2.10 create_user.php

Este script contiene un formulario HTML para la creación de un nuevo usuario. El formulario contiene cinco campos: nombre de usuario, email, tipo de usuario (usuario normal o admin), contraseña, y otro campo para la confirmación de la contraseña.

5.2.2.11 delete.php

Este script permite borrar la cuenta de un usuario definitivamente.

5.2.2.12 edit.php

Esta script nos permite editar los datos de un usuario registrado en el sistema. Contiene un formulario HTML.

5.2.2.13 RandomClass.php

Este script contiene todas las funciones necesarias para la representación de los datos ambientales (temperatura, humedad y luminosidad) de forma gráfica. A continuación, vamos a explicar, a nivel de código, las funciones más importantes.

```php
function conectarBD(){
        $server = "localhost";
        $usuario = "id8146483_admin";
        $pass = "admin";
        $BD = "id8146483_tfg";
        //variable que guarda la conexión de la base de datos
        $conexion = mysqli_connect($server, $usuario, $pass, $BD);
        //Comprobamos si la conexión ha tenido exito
        if(!$conexion){
           echo 'Ha sucedido un error inexperado en la conexion de la base de datos<br>';
        }
        //devolvemos el objeto de conexión para usarlo en las consultas
        return $conexion;
}
```

Figura 5.22 función conectarBD

Está función crea y devuelve un objeto de conexión a la base de datos y chequea el estado de la misma.

```php
function desconectarBD($conexion){
        //Cierra la conexión y guarda el estado de la operación en una variable
        $close = mysqli_close($conexion);
        //Comprobamos si se ha cerrado la conexión correctamente
        if(!$close){
           echo 'Ha sucedido un error inexperado en la desconexion de la base de datos<br>';
        }
        //devuelve el estado del cierre de conexión
        return $close;
}
```

Figura 5.23 función desconectarBD

La función “desconectarBD” se encarga de cerrar la conexión con la base de datos.

```php
function getArraySQL($sql){
    //Creamos la conexión
    $conexion = $this->conectarBD();
    //generamos la consulta
    if(!$result = mysqli_query($conexion, $sql)) die();

    $rawdata = array();
    //guardamos en un array multidimensional todos los datos de la consulta
    $i=0;
    while($row = mysqli_fetch_array($result))
    {
        //guardamos en rawdata todos los vectores/filas que nos devuelve la consulta
        $rawdata[$i] = $row;
        $i++;
    }
    //Cerramos la base de datos
    $this->desconectarBD($conexion);
    //devolvemos rawdata
    return $rawdata;
}
```

Figura 5.24 función getArraySQL

Está función se encarga de devolver un array multidimensional con el resultado de una consulta sql.

```
function getAllInfo(){
    $conexion = $this->conectarBD();
    $sqlPrueba ="SELECT valor_prueba FROM tabla_prueba ORDER BY id DESC LIMIT 1";
    $result = mysqli_query($conexion, $sqlPrueba);
    $row = mysqli_fetch_assoc($result);
    //Creamos la consulta
    $sql = "SELECT * FROM monitorizacion WHERE id_nodo='". $row['valor_prueba']."'";
    //$sql = "SELECT * FROM ArduinoEthernet2;";
    //obtenemos el array con toda la información
    return $this->getArraySQL($sql);
}
```

Figura 5.25 función getAllInfo

Está función se encarga de ejecutar una sentencia sql para obtener los parámetros medioambientales guardados en la tabla "monitorización" para un determinado nodo.

5.3 Algoritmo para la determinación del periodo de muestreo óptimo.

Como se ha indicado en el **apartado 2.2.2**, el algoritmo para la determinación del periodo de muestreo óptimo está basado en un sistema borroso o "fuzzy" basado en reglas.

Las variables de entrada van a ser la variabilidad instantánea y la variabilidad media de los parámetros ambientales leídos por los sensores de temperatura, humedad y luminosidad. La variable de salida va a ser el periodo de muestreo.

5.3.1 Conjuntos borrosos

En este apartado, se van a definir los conjuntos borrosos, tanto para las variables de entrada, como para las variables de salida. El modelo geométrico que se va a usar para determinar la pertenencia de los valores previamente normalizados entre 0 y 1, es el trapezoidal.

El trapezoide está definido por sus límites inferior a, y superior d, y los límites de soporte inferior b, y superior c, tal que a<b<c<d.

En este caso, si los valores de b y c son iguales, se obtiene una función triangular.

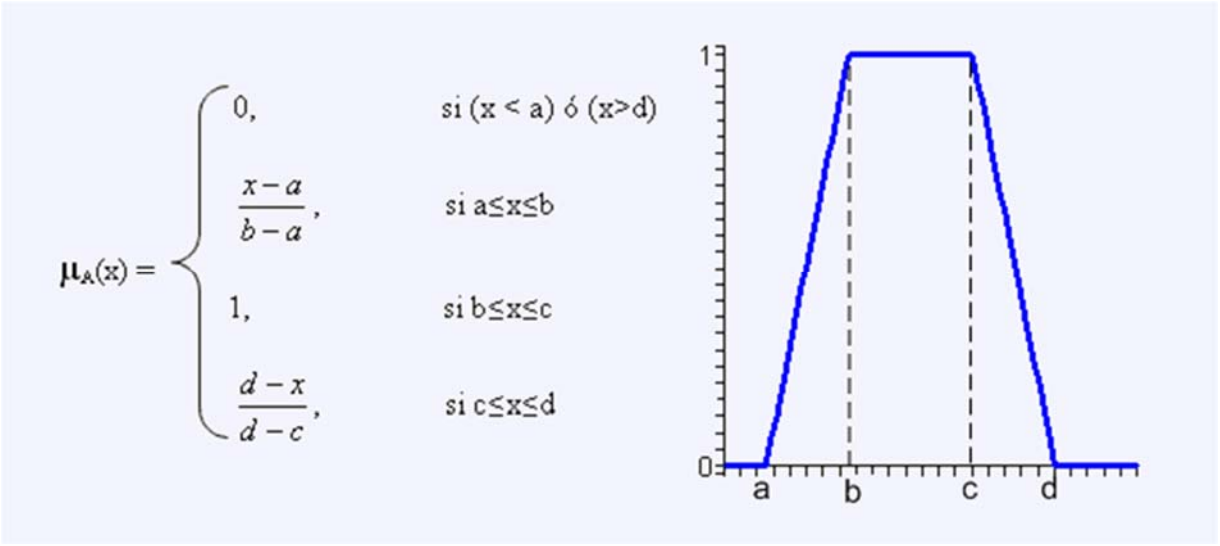

Figura 5.26 Función de pertenencia de un trapezoide

5.3.1.1 La variabilidad instantánea

En el cálculo de la variabilidad instantánea están involucrados tres variables: la temperatura, la humedad y la luminosidad. Para calcular la variabilidad instantánea vamos a seguir los pasos que se detallan a continuación.

- **Paso 1:** Normalizar los parámetros ambientales leídos por los sensores entre 0 y 1.
- **Paso 2:** Calcular la variabilidad instantánea para cada parámetro ambiental.
- **Paso 3:** Calcular la variabilidad instantánea total.

5.3.1.1.1 Normalización de parámetros medioambientales

Como se ha mencionado anteriormente en el **capítulo 4**, el sensor usado para medir la temperatura y la humedad es el DHT11. Este sensor, mide la temperatura en un rango entre 0 y 50 grados y la humedad entre 20% y 90% [17].

Para normalizar los valores de la temperatura y la humedad, simplemente vamos a dividir el valor de la temperatura entre 50 y el valor de la humedad entre 90.

Para medir la luminosidad se ha usado una LDR. Sus valores máximos suelen estar alrededor a 500 Luxes. Para normalizar el valor de la luminosidad, vamos a dividir entre 500.

5.3.1.1.2 Cálculo de la variabilidad instantánea de cada parámetro

La variabilidad instantánea es la diferencia entre el valor tomado por un parámetro en un instante "t" y otro instante "t-1". Por ejemplo, si la temperatura en el instante $t_6 = 26º$ y en el instante $t_7 = 28º$. El primer paso es normalizar los valores de temperatura: $t_6 = \frac{26}{50} = 0.52$, $t_7 = \frac{28}{50} = 0.56$.

El segundo paso es calcular la variabilidad instantánea: $v_i = t_7 - t_6 = 0.56 - 0.52 = 0.04$.

5.3.1.1.3 Cálculo de la variabilidad instantánea total

Para calcular la variabilidad instantánea total V_i, vamos a aplicar la siguiente fórmula.

$$Vi = \frac{1}{3} * Vt + \frac{1}{3} * Vh + \frac{1}{3} * Vl$$

Siendo Vt, Vh y Vl, la variabilidad instantánea de temperatura, humedad y luminosidad respectivamente.

5.3.1.1.4 Conjuntos borrosos de la variabilidad instantánea

Los valores de la variabilidad instantánea los podemos clasificar en tres conjuntos borrosos: **variabilidad instantánea baja, media y alta**.

- **El conjunto borroso de la variabilidad instantánea baja** está definido por los siguientes puntos (valores normalizados): A = 0; B = 0; C = 0,08; D = 0,12.
- **El conjunto borroso de la variabilidad instantánea media** está definido por los siguientes puntos (valores normalizados): A = 0,08; B = 0,12; C = 0,25; D = 0,29.
- **El conjunto borroso de la variabilidad instantánea alta** está definido por los siguientes puntos (valores normalizados): A = 0,25; B = 0,29; C = 1; D = 1.

5.3.1.2 La variabilidad media

La variabilidad media de los parámetros medioambientales V_m de n muestras, es el sumatorio de las variabilidades instantáneas V_i desde el instante t = 0 hasta el instante t = n.

Para el cálculo de la variabilidad media V_m, se va a usar la siguiente fórmula:

$$\mathrm{Vm} = \frac{1}{n}\sum_{i=0}^{n} \mathrm{Vi}$$

Los valores de la variabilidad media los podemos clasificar en tres conjuntos borrosos: **variabilidad media baja, media y alta**.

- **El conjunto borroso de la variabilidad media baja** está definido por los siguientes puntos (valores normalizados): A = 0; B = 0; C = 0,08; D = 0,12.
- **El conjunto borroso de la variabilidad media media** está definido por los siguientes puntos (valores normalizados): A = 0,08; B = 0,12; C = 0,25; D = 0,29.
- **El conjunto borroso de la variabilidad media alta** está definido por los siguientes puntos (valores normalizados): A = 0,25; B = 0,29; C = 1; D = 1.

5.3.1.3 Salida

Como se ha mencionado anteriormente, la salida del sistema borroso va a ser el periodo de muestreo. Los valores de la salida se dividirán en cinco conjuntos borrosos: **muy bajo, bajo, medio, alto y muy alto**.

Los conjuntos borrosos de la salida serán los siguientes:

- **El conjunto borroso de la variabilidad media muy baja** está definido por los siguientes puntos (valores normalizados): A = 0; B = 0; C = 0,2; D = 0,3.

- **El conjunto borroso de la variabilidad media baja** está definido por los siguientes puntos (valores normalizados): A = 0,2; B = 0,3; C = 0,4; D = 0,5.
- **El conjunto borroso de la variabilidad media media** está definido por los siguientes puntos (valores normalizados): A = 0,4; B = 0,5; C = 0,6; D = 0,7.
- **El conjunto borroso de variabilidad media alta** está definido por los siguientes puntos (valores normalizados): A = 0,6; B = 0,7; C = 0,8; D = 0,9.
- **El conjunto borroso de la variabilidad media muy alta** está definido por los siguientes puntos (valores normalizados): A = 0,8; B = 0,9; C = 1; D = 1.

Después de obtener el periodo de muestreo (salida del sistema borroso), se va a hacer un proceso de desnormalización para obtener el valor final que se va a usar en los nodos. A continuación, vamos a explicar el proceso de desnormalización usado.

La relación entre el periodo (entre 5 min y 60 min) y el valor de tau (entre 0 y 1) es lineal.

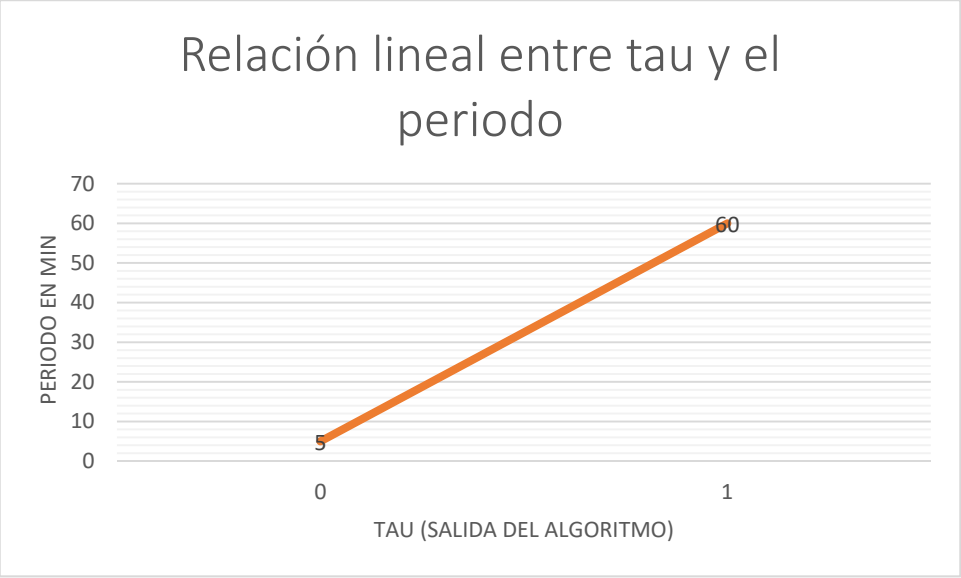

Figura 5.27 Relación lineal entre tau y el periodo

La ecuación de la función representada en la **figura 5.27** es:

$$y = ax + b$$

Vamos a determinar el valor de a y b.

$$5 = a * (0) + b$$

$$\boldsymbol{b = 5}$$

$$y = ax + 5$$

$$60 = a * (1) + 5$$

$$a = 60 - 5$$

$$\boldsymbol{a = 55}$$

$$\boldsymbol{y = 55x + 5}$$

Mediante la **ecuación 5.1**, podemos obtener el periodo de muestreo que vamos a usar en los nodos.

A continuación, se va a presentar un ejemplo para explicar el proceso de desnormalización anterior.

Vamos a suponer que la salida del algoritmo es **tau = 0.92**.

$$y = 55 * 0.92 + 5$$

$$y = 55.6\ min$$

El periodo que se va a usar en los nodos es de **55.6 min**.

5.3.2 Conjunto de reglas:

1. Si la variabilidad instantánea es **baja** y la variabilidad media es **baja**, el periodo de muestreo es **muy alto**.
2. Si la variabilidad instantánea es **media** y la variabilidad media es **baja**, el periodo de muestreo es **bajo**.
3. Si la variabilidad instantánea es **alta** y la variabilidad media es **baja**, el periodo de muestreo es **medio**.
4. Si la variabilidad instantánea es **baja** y la variabilidad media es **media**, el periodo de muestreo es **bajo**.
5. Si la variabilidad instantánea es **media** y la variabilidad media es **media**, el periodo de muestreo es **medio**.
6. Si la variabilidad instantánea es **alta** y la variabilidad media es **media**, el periodo de muestreo es **alto**.
7. Si la variabilidad instantánea es **baja** y la variabilidad media es **alta**, el periodo de muestreo es **medio**.
8. Si la variabilidad instantánea es **media** y la variabilidad media es **alta**, el periodo de muestreo es **alto**.
9. Si la variabilidad instantánea es **alta** y la variabilidad media es **alta**, el periodo de muestreo es **muy bajo**.

A continuación, se va a presentar una tabla donde vienen estas reglas de forma resumida:

V_{media}/V_i	*Baja*	*Media*	*Alta*
Baja	Muy alto	Bajo	Medio
Media	Bajo	Medio	Alto
Alta	Medio	Alto	Muy Bajo

Tabla 5.4 Reglas FRBS

5.3.3 Librería motorv2

Para la implementación de la lógica borrosa, se va a hacer uso de la librería motorv2, desarrollada en C++. Está librería va a efectuar todas las operaciones necesarias y como resultado final nos devuelva el periodo de muestreo.

Está librería estará formada por dos archivos: motorv2.h y motorv2.cpp. En motorv2.h estarán definidas todas las clases, así como todos los métodos que se van a usar.

Tenemos definidas dos clases: la clase principal, que será motorv2 y otra clase peso para calcular el peso de cada regla.

```
//
// File:   motorv3.h
//Author: Amine Alrharad
//
//Created on 24 de septiembre de 2018, 18:50
//
// si no está definida, defino la libreria
#ifndef motorv3_h
#define motorv3_h
// hay que incluir esta libreria siempre
#include <Arduino.h>
#include <Datos.h>
class peso {
    // al calcular el peso de una regla, nos devuelve el centro del grupo difuso y su área (masa).
    public:
        double centro;
        double masa;
  };
// defino la clase
class motorv3{
 // defino los métodos publicos
public:
    motorv3();
    double Pertenencia(double X, double a, double b, double c, double d);
    peso CalculaPeso (double altura, double a, double b, double c, double d);
    double MotorInfiere (float vi,float vm,int num_reglas );
};

#endif
```

Figura 5.28 Motorv2.h

En el archivo motorv2.cpp, se va a implementar un método principal llamado motorInfiere() que, dados los datos de variabilidad instantánea, la variabilidad media y el número de reglas, nos calcula el periodo de muestreo óptimo.

Lo primero que hace este método es extraer los puntos que definen los conjuntos borrosos de entrada para la variabilidad instantánea y la variabilidad media respectivamente, y luego calcula su grado de pertenencia.

```
for (iregla=0; iregla < num_reglas; iregla++ ) {

    grado[iregla] = 1.0; // inicialmente  1, miramos cada antecedente, y nos quedamos con el minimo

        for (i=0;i<((num_puntos-4)/4);i++){ // en cada fila , 4 puntos por conjunto borroso, resto 4 porque son los del conjunto borroso de salida
                                            // y despues divido entre 4 para saber el número de antecedentes

            iC=i*4; // nos indica el valor inicial (a) del conjunto borroso que vamos a utilizar
         // extraigo los puntos que definen el conjunto borroso
            v1 = BDm[iregla][iC];
            v2 = BDm[iregla][iC+1];
            v3 = BDm[iregla][iC+2];
            v4 = BDm[iregla][iC+3];
         // extraigo el grado de pertenencia
            if(i==0){
                grado_aux = Pertenencia(vi, v1, v2, v3, v4);
            } else{
                grado_aux = Pertenencia(vm, v1, v2, v3, v4);
                  }
        // compruebo si es menor al que tenía, si es así, es el nuevo grado de pertenencia
            if ( grado_aux < grado[iregla] ){
                grado[iregla] = grado_aux;   //Grado mínimo pertenencia
                }
        }
```


Figura 5.29 MotorInfiere (1)

Luego, extrae los puntos que definen el conjunto borroso de salida, para calcular el peso de cada regla y su aportación al total.

```
     // ahora extraigo los puntos que definen el conjunto borroso de salida, que serán los últimos 4 puntos de cada fila

          iC= num_puntos-4;
          v1 = BDm[iregla][iC];
          v2 = BDm[iregla][iC+1];
          v3 = BDm[iregla][iC+2];
          v4 = BDm[iregla][iC+3];

     // calculo el peso de la regla en función del grado mínimo de pertenencia de los antecedentes

          resultado = CalculaPeso ( grado[iregla],v1, v2, v3, v4);

    // voy sumando el producto del centro de gravedad por la masa (área) del conjunto borroso

          suma_centro_por_area = suma_centro_por_area + resultado.centro*resultado.masa;

    // voy sumando la masa (área) de cada conjunto borroso

          sumaarea=sumaarea + resultado.masa;
```

Figura 5.30 MotorInfiere (2)

Y finalmente, el proceso de desnormalización.

```
if (sumaarea > (float) 0.0 ){
    // finalmente , obtengo el valor de salida, defuzzificando mediante el proceso de suma ponderada de centros de gravedad
    // que es el producto de la masa por el centro entre la suma del área total

    return ( suma_centro_por_area / sumaarea );
    }
else{
    return ( 0.0 );
     }
```


Figura 5.31 MotorInfiere (3)

Para calcular la pertenencia de un valor a un conjunto borroso, se usa el método Pertenencia.

```
double motorv3::Pertenencia(double X, double a, double b, double c, double d)  {
   // Esta función calcula la pertenencia a un grupo borroso del valor X definido por los puntes a,b,c y d
   // siendo los puntos que definen un trapecio a y d inicio y final que valen 0, y b y c , los puntos
   // que valen 1.

   // resuelve la ecuación del trapecio para el valor X (eje de abscisas ) obteniendo el valor de pertenencia
   // (eje de ordenadas)

  double pert=0.0;
//   Serial.print("Valor dentro de la funcion de pertenencia: "); Serial.println(X);
  if ((X>a)&&(X<d))
  {
      if (X<b)
          pert=(X-a)/(b-a);
      else if (X<=c)
          pert=1;
      else
          pert=(X-d)/(c-d);//revisar
  }
//  Serial.print("Pertenencia: ");Serial.println(pert);
  return pert;
  }
```

Figura 5.32 Método Pertenencia()

Y para obtener la aportación de cada regla al final, se va a usar el método CalculaPeso() que usa como entradas los puntos que definen el conjunto borroso de salida y el grado mínimo de pertenencia.

Lo que hace este método es dividir el trapezoide en tres áreas: dos triángulos y un rectángulo y calcula su centro de gravedad y su área, para que luego hace una suma ponderada del centro por el área.

```
peso motorv3::CalculaPeso (double altura, double a, double b, double c, double d){
// la altura corresponde con el valor mínimo de pertenencia a su correpodiente conjunto borroso teniendo en cuenta
// cada variable de entrada.

// calcula el área y centro resolviendo la ecuacion del conjunto borroso con forma de trapecio
// para el valor de altura correspondiente


double r2=(double)2.0/(double)3.0; // preguntar por esto porque no lo entiendo
//
double centro,area,h,c1,a1,c2,a2,c3,a3;
peso result = peso();
h=altura;
centro=0.0; area=0.0;

c1=a+(b-a)*r2; // calcula el centro del triangulo izquierdo al dividir el trapecio
c2=b+(c-b)/2;// calcula el centro del rectangulo al dividir el trapecio
c3=c+(1-r2)*(d-c);// calcula el centro del triangulo derecho al dividir el trapecio

a1=(b-a)*h/2; // calcula el triángulo izquierdo al dividir el trapecio
a2=(c-b)*h; // calcula el rectángulo central al dividir el trapecio
a3=(d-c)*h/2;  // calcula el triangulo derecho al divir el trapecio

area=a1+a2+a3; // el área total es la suma de las áreas

if (area>0)
    centro=(c1*a1+c2*a2+c3*a3)/area; // si ponderamos el centro parcial por su masa parcial entre la masa total, obtendremos el centro total
else{
    area=0;
    centro=0;
}

result.centro = centro;
result.masa = area;

 return result;
}
```


Figura 5.33 Método CalculaPeso()

Finalmente, se ha implementado una matriz multidimensional como base de reglas. Esta matriz está constituida por 9 filas, una fila para cada regla, y por 12 columnas, 4 columnas por la variabilidad instantánea, 4 columnas por la variabilidad media, y los últimos 4 columnas por el periodo de muestreo.

```
double BDm [][12]={
                {0.00, 0.01, 0.02, 0.25, 0.00, 0.01, 0.02, 0.25, 0.89, 0.97, 0.98, 1},
                {0.23, 0.49, 0.51, 0.76,  0.00, 0.01, 0.02, 0.25, 0.10, 0.19, 0.20, 0.30},
                {0.75, 0.85, 0.87, 1, 0.00, 0.01, 0.02, 0.25, 0.29, 0.49, 0.51, 0.71},

                {0.00, 0.01, 0.02, 0.25, 0.23, 0.49, 0.51, 0.76, 0.10, 0.19, 0.20, 0.30},
                {0.23, 0.49, 0.51, 0.76, 0.23, 0.49, 0.51, 0.76, 0.29, 0.49, 0.51, 0.71},
                {0.75, 0.85, 0.87, 1, 0.23, 0.49, 0.51, 0.76, 0.70, 0.79, 0.81, 0.90},

                {0.00, 0.01, 0.02, 0.25, 0.75, 0.85, 0.87, 1, 0.29, 0.49, 0.51, 0.71},
                {0.23, 0.49, 0.51, 0.76, 0.75, 0.85, 0.87, 1, 0.70, 0.79, 0.81, 0.90},
                {0.75, 0.85, 0.87, 1, 0.75, 0.85, 0.87, 1, 0.00, 0.01, 0.02, 0.11}
            };
```

Figura 5.34 Base de reglas

CAPÍTULO 6. PRUEBAS DE FUNCIONALIDAD

En este capítulo, se va a intentar demostrar la efectividad del algoritmo de predicción del periodo de muestreo frente a datos con diferentes variabilidades. Para tal fin, vamos a trabajar con dos conjuntos de datos: datos con poca variabilidad (datos homogéneos), y datos con mucha variabilidad (datos heterogéneos).

Para cada conjunto de datos, se va a usar un periodo fijo y variable (algoritmo de predicción del periodo de muestreo) en la captura de datos.

Para que se vea con mayor claridad el funcionamiento del algoritmo de predicción del periodo de muestreo, se va a considerar una sola variable, en este caso, la temperatura.

6.1 Conjunto de datos homogéneos

6.1.1 Periodo de muestreo fijo

6.1.1.1 Error cuadrático medio

En esta sección, se van a realizar lecturas de temperatura con un periodo fijo (**cada 5min**). A continuación, se presentarán las 25 muestras capturadas:

Instantes de muestreo (min)	Temperatura
0	10.00
5	10.00
10	11.00
15	12.00
20	11.00
25	9.00
30	7.00
35	6.00
40	4.00
45	3.00
50	2.00
55	2.00
60	3.00
65	4.00
70	4.00
75	5.00
80	5.00
85	6.00
90	6.00
95	6.00
100	6.00

105	7.00
110	8.00
115	8.00
120	9.00

Tabla 6.1 datos de temperatura (periodo 5min)

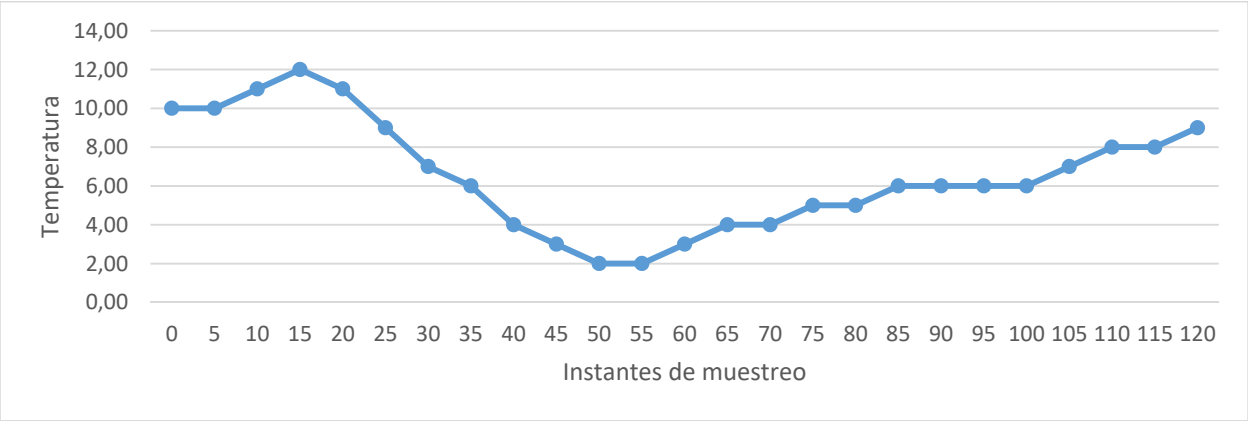

Figura 6.1 Evolución de temperatura (tabla 6.1)

La segunda captura se va a realizar cada **12min**, y los resultados son los que se presentan en la **figura 6.2**:

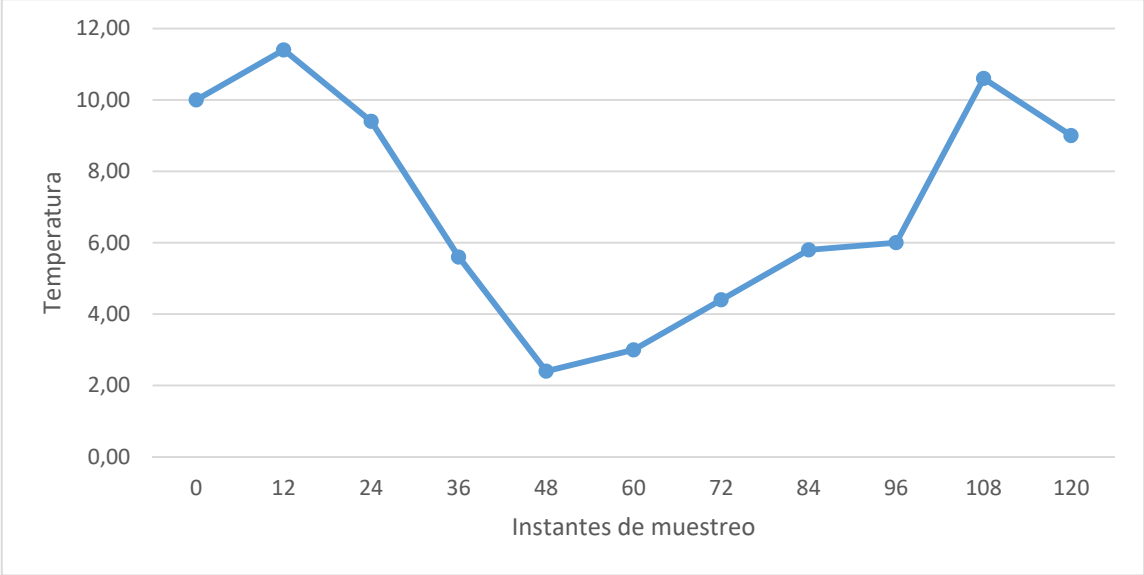

Figura 6.2 Evolución de temperatura (periodo 12min)

Se va a considerar que la temperatura entre dos puntos evoluciona de acuerdo a la **figura 6.2**.

Según la **figura 6.2**, los valores de temperatura cada 5min, son los que se detallan en la **tabla 6.2**:

Instante de muestreo (min)	Temperatura
0	10.00
5	10.83
10	11.66
15	10.90
20	10.06
25	9.09
30	7.50
35	5.91
40	4.54
45	3.20
50	2.50
55	2.75
60	3.00
65	3.58
70	4.17
75	4.75
80	5.34
85	5.81
90	5.90
95	5.98
100	7.54
105	9.45
110	10.34
115	9.67
120	9.00

Tabla 6.2 datos de temperatura (periodo 12min)

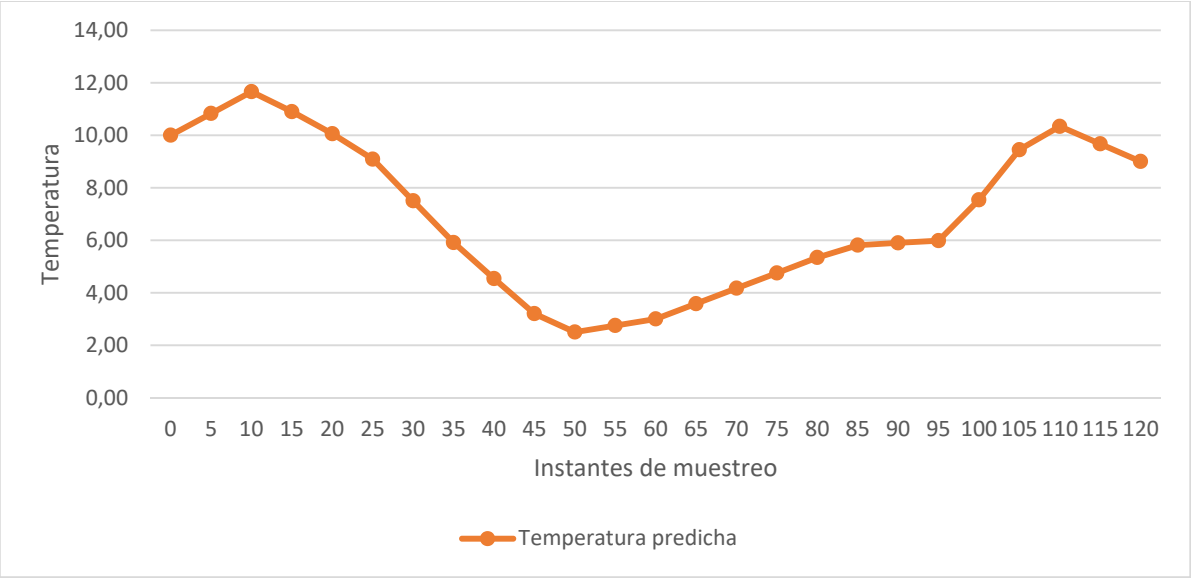

Figura 6.3 evolución de temperatura (tabla 6.2)

Lo que vamos a hacer ahora, es calcular la diferencia entre los valores de temperatura de la **tabla 6.1** y los valores de temperatura de la **tabla 6.2**, para que, posteriormente, calcular el error cuadrático medio (RMSE).

Para el cálculo del error cuadrático medio, se ha usado la siguiente fórmula:

$$RMSE = \sqrt{\frac{\sum_{i=1}^{n}(P_i - O_i)^2}{n}}$$

Siendo, P_i el valor predicho, O_i el valor observado, y n el número de muestras.

Instante de muestreo (min)	Temperatura observada	Temperatura predicha	Diferencia	RMSE
0	10.00	10.00	0.00	0.93
5	10.00	10.83	0.83	
10	11.00	11.66	0.66	
15	12.00	10.90	1.10	
20	11.00	10.06	0.94	
25	9.00	9.09	0.09	
30	7.00	7.50	0.50	
35	6.00	5.91	0.09	
40	4.00	4.54	0.54	
45	3.00	3.20	0.20	
50	2.00	2.50	0.50	
55	2.00	2.75	0.75	
60	3.00	3.00	0.00	
65	4.00	3.58	0.42	
70	4.00	4.17	0.17	
75	5.00	4.75	0.25	
80	5.00	5.34	0.34	
85	6.00	5.81	0.19	
90	6.00	5.90	0.10	
95	6.00	5.98	0.02	
100	6.00	7.54	1.54	
105	7.00	9.45	2.45	
110	8.00	10.34	2.34	
115	8.00	9.67	1.67	
120	9.00	9.00	0.00	

Tabla 6.3 Error cuadrático medio (periodo fijo 12min)

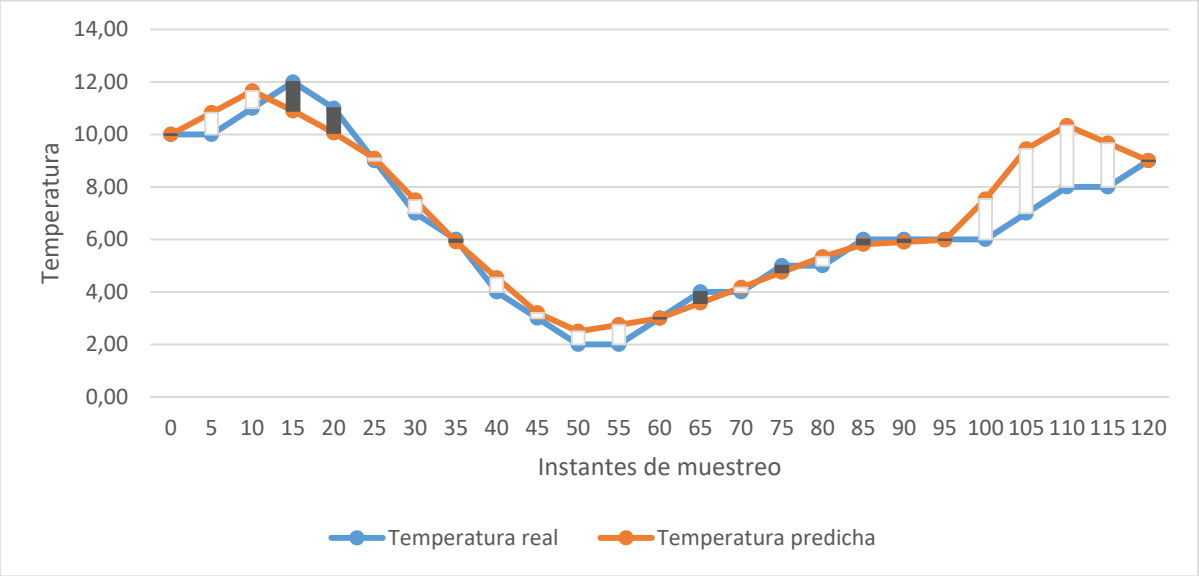

Figura 6.4 Error cuadrático medio (tabla 6.3)

6.1.1.2 Consumo energético

En esta sección, se va a calcular el consumo energético de un nodo con un periodo de muestreo fijo (5min).

El consumo energético de un nodo cuando está transmitiendo está alrededor de 50mA, mientras que, consume alrededor de 20µA cuando está en el modo "Deep-sleep". Para calcular el consumo total de un nodo, se va a tomar en cuenta las siguientes consideraciones:

- El tiempo que tarda un nodo para transmitir los datos al servidor es de 1 segundo.
- El nodo se encuentra en el modo "Deep-sleep", mientras no está transmitiendo.

Ahora, vamos a calcular cuánto realmente consume nuestro sistema cada 5min.

$$5min * 60 = 300\ segundos$$

$$20\mu A = 0.02mA$$

$$1 * 50 + 299 * 0.02 = 55.98mA$$

El sistema consume alrededor de 56mA cada 5min.

En la **tabla 6.1,** hemos tomado 25 muestras. El consumo total en 2 horas sería:

$$56mA * 25 = \mathbf{1400mA}$$

El consumo energético total en 2 horas es de **1400mA**.

6.1.2 Periodo de muestreo variable

6.1.2.1 Error cuadrático medio

En este apartado, se va a aplicar el algoritmo de predicción del periodo de muestreo sobre los datos capturados en el **apartado 6.1.1**. A continuación, se presentarán los periodos óptimos (algoritmo de predicción) entre dos lecturas consecutivas:

Periodo óptimo (min)	Instantes de muestreo (min)	Temperatura
8.00	8.00	10.60
7.77	15.77	11.85
7.77	23.54	9.58
7.77	31.31	6.74
6.13	37.44	5.02
6.13	43.57	3.29
7.77	51.34	2.00
6.13	57.47	2.49
7.77	65.24	3.00
7.77	73.01	4.60
8.00	81.01	5.20
7.77	88.78	6.00
7.77	96.55	6.00
8.00	104.55	2.91
7.77	112.32	8.00
7.00	119.32	8.86

Tabla 6.4 Periodos de muestreo obtenidos con el algoritmo y temperaturas

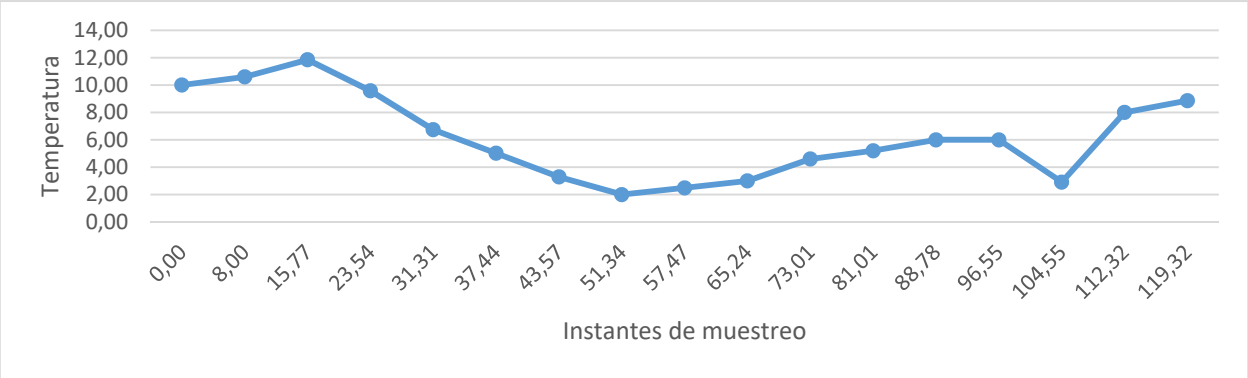

Figura 6.5 Evolución temperatura (tabla 6.4)

Igual que en el **apartado 6.1.1**, se va a calcular la diferencia entre los valores reales y los valores predichos de temperatura, con el fin de calcular el error cuadrático medio.

Instantes de muestreo (min)	Temperatura real	Temperatura predicha	Diferencia	RMSE
0	10	10	0	0.66
5	10	10.38	0.38	
10	11	10.92	0.08	
15	12	11.72	0.28	
20	11	10.87	0.13	
25	9	9.04	0.04	
30	7	7.21	0.21	
35	6	5.71	0.29	
40	4	4.30	0.30	
45	3	3.05	0.05	
50	2	2.22	0.22	
55	2	2.30	0.30	
60	3	2.66	0.34	
65	4	2.99	1.01	
70	4	3.99	0.01	
75	5	4.74	0.26	
80	5	5.12	0.12	
85	6	5.61	0.39	
90	6	6	0	
95	6	6	0	
100	6	4.68	1.32	
105	7	3.20	3.80	
110	8	6.48	1.52	
115	8	8.33	0.33	
120	9	8.86	0.14	

Tabla 6.5 Error cuadrático medio (periodo variable)

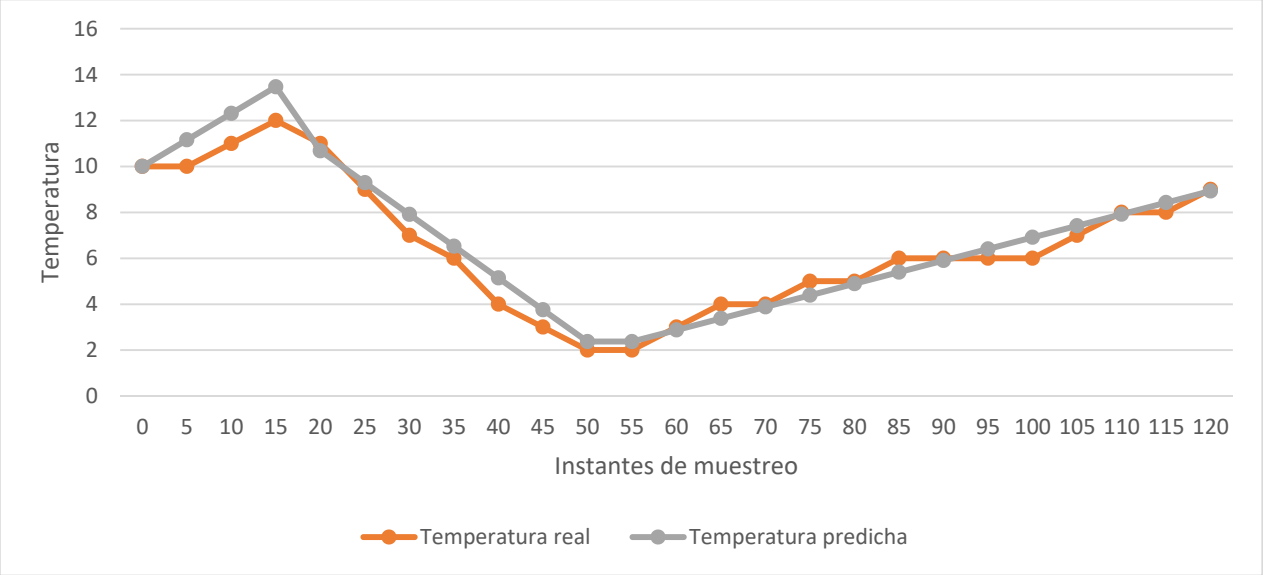

Figura 6.6 Error cuadrático medio (tabla 6.5)

Como se puede apreciar en los resultados, el error cuadrático medio es menor usando el algoritmo de predicción del periodo de muestreo (0.66), mientras que, usando un periodo de muestreo fijo, obtenemos un error más grande (0.93).

6.1.2.2 Consumo energético

En esta sección, vamos a calcular el consumo energético total de un nodo, usando un periodo de muestreo variable.

Periodo en segundos	Tiempo usado en la trasmisión en segundos	Consumo en mA	Tiempo en el modo Deep-sleep en segundos	Consumo en mA	Consumo total en mA
480	1	50	479	0.02	59.58
466.2	1	50	465.2	0.02	59.304
466.2	1	50	465.2	0.02	59.304
466.2	1	50	465.2	0.02	59.304
367.8	1	50	366.8	0.02	57.336
367.8	1	50	366.8	0.02	57.336
466.2	1	50	465.2	0.02	59.304
367.8	1	50	366.8	0.02	57.336
466.2	1	50	465.2	0.02	59.304
466.2	1	50	465.2	0.02	59.304
480	1	50	479	0.02	59.58
466.2	1	50	465.2	0.02	59.304
466.2	1	50	465.2	0.02	59.304
480	1	50	479	0.02	59.58
466.2	1	50	465.2	0.02	59.304
420	1	50	419	0.02	58.38

Tabla 6.6 Consumo energético con periodo variable

El consumo total del nodo en 120min es de **942mA**.

Como podemos apreciar, el consumo total del nodo ha bajado **458mA** en comparación con el **apartado 6.1.1.2** (1400mA) en solo 2 horas.

6.2 conjunto de datos con mucha variabilidad

En este caso, tenemos un conjunto de datos de temperatura con mucha variabilidad. Para obtener está variabilidad entre los datos capturados, hemos tenido que someter el sensor de temperatura a diferentes condiciones climáticas, y hemos obtenido los siguientes resultados:

{9, 14, 8, 7, 6, 16, 15, 6, 5, 11, 4, 4, 3, 12, 2, 3, 12, 6, 11, 17, 11, 12, 11, 10, 10}

6.2.1 Periodo de muestro fijo

6.2.1.1 Error cuadrático medio

Se va a seguir el mismo procedimiento del **apartado 6.1.1**.

Instantes de muestreo	Temperatura real	Temperatura predicha	Diferencia	RMSE
0	9,00	9,00	0,00	3,42
5	14,00	8,50	5,50	
10	8,00	8,00	0,00	
15	7,00	7,00	0,00	
20	6,00	6,00	0,00	
25	16,00	10,50	5,50	
30	15,00	15,00	0,00	
35	6,00	10,00	4,00	
40	5,00	5,00	0,00	
45	11,00	4,50	6,50	
50	4,00	4,00	0,00	
55	4,00	3,50	0,50	
60	3,00	3,00	0,00	
65	12,00	2,50	9,50	
70	2,00	2,00	0,00	
75	3,00	7,00	4,00	
80	12,00	12,00	0,00	
85	6,00	11,50	5,50	
90	11,00	11,00	0,00	
95	17,00	11,00	6,00	
100	11,00	11,00	0,00	
105	12,00	11,00	1,00	
110	11,00	11,00	0,00	
115	10,00	10,50	0,50	
120	10,00	10,00	0,00	

Tabla 6.7 Error cuadrático medio

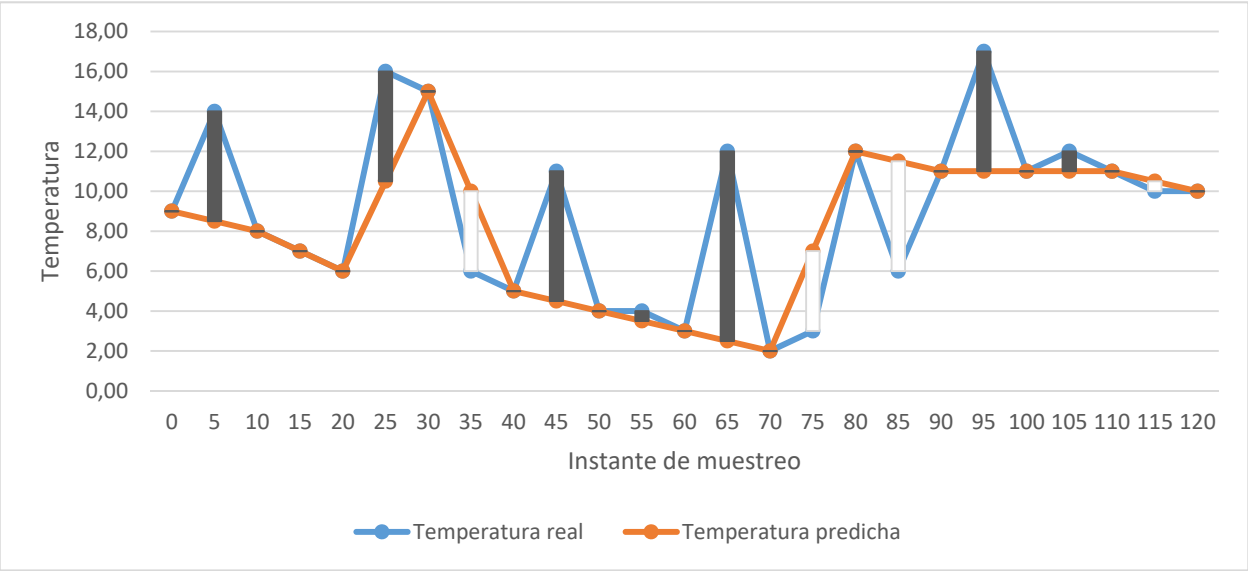

Figura 6.7 Error cuadrático medio (tabla 6.6)

6.2.1.2 Consumo energético

En esta sección, el consumo energético es similar al **apartado 6.1.1.2**.

6.2.2 Periodo de muestreo variable

6.2.2.1 Error cuadrático medio

En este apartado, se va a aplicar el algoritmo de predicción del periodo de muestreo sobre el conjunto de datos de temperatura antes mencionado (**apartado 6.2**), y se van a obtener los periodos de muestreo óptimos.

Instantes de muestreo (min)
0.00
5.38
10.38
16.05
21.05
26.18
31.05
35.89
41.05
46.63
50.92
58.00
61.05
66.81
70.71
76.30

80.65
85.49
90.47
95.38
100.38
106.54
111.39
116.25
123.00

Tabla 6.8 Instantes de muestreo

A continuación, se va a calcular el error cuadrático medio.

Instante de muestreo (min)	Temperatura real	Temperatura predicha	Diferencia	RMSE
0	9.00	9.00	0.00	1.25
5	14.00	13.54	0.46	
10	8.00	7.20	0.08	
15	7.00	6.79	0.21	
20	6.00	8.10	2.10	
25	16.00	15.76	0.24	
30	15.00	13.11	1.89	
35	6.00	5.82	0.18	
40	5.00	6.26	1.26	
45	11.00	8.72	2.28	
50	4.00	4.00	0.00	
55	4.00	3.40	0.60	
60	3.00	4.89	1.89	
65	12.00	8.38	3.62	
70	2.00	2.14	0.14	
75	3.00	5.34	2.34	
80	12.00	11.22	0.78	
85	6.00	6.49	0.49	
90	11.00	11.56	0.56	
95	17.00	16.54	0.46	
100	11.00	11.08	0.08	
105	12.00	11.69	0.31	
110	11.00	10.72	0.28	
115	10.00	10.00	0.00	
120	10.00	10.00	0.00	

Tabla 6.9 Error cuadrático medio

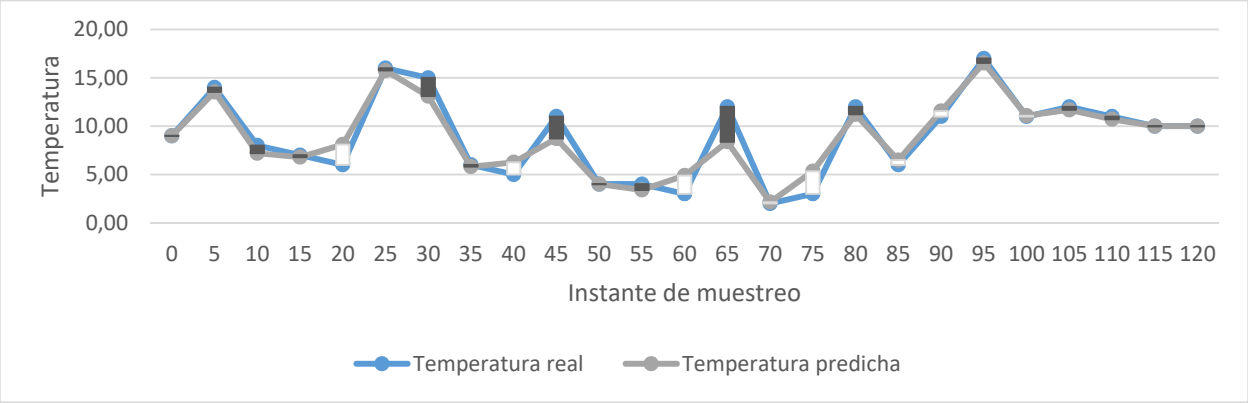

Figura 6.8 Error cuadrático medio (tabla 6.8)

6.2.2.2 Consumo energético

Siguiendo los mismos pasos de la **sección 6.1.2.2,** vamos a calcular el consumo energético total de un nodo durante 2 horas.

Periodo en segundos	Tiempo usado en la trasmisión en segundos	Consumo en mA	Tiempo en el modo Deep-sleep en segundos	Consumo en mA	Consumo total en mA
300	1	50	299	0.02	55.98
340.2	1	50	339.2	0.02	56.784
300	1	50	299	0.02	55.98
307.8	1	50	306.8	0.02	56.136
292.2	1	50	291.2	0.02	55.824
290.4	1	50	289.4	0.02	55.788
309.6	1	50	308.6	0.02	56.172
334.8	1	50	333.8	0.02	56.676
257.4	1	50	256.4	0.02	55.128
424.8	1	50	423.8	0.02	58.476
183	1	50	182	0.02	53.64
345.6	1	50	344.6	0.02	56.892
234	1	50	233	0.02	54.66
335.4	1	50	334.4	0.02	56.688
261	1	50	260	0.02	55.2
290.4	1	50	289.4	0.02	55.788
298.8	1	50	297.8	0.02	55.956
294.6	1	50	293.6	0.02	55.872
300	1	50	299	0.02	55.98
369.6	1	50	368.6	0.02	57.372
291	1	50	290	0.02	55.8
291.6	1	50	290.6	0.02	55.812
405	1	50	404	0.02	58.08

Tabla 6. 10 Consumo energético con periodo variable

El consumo total del nodo en 120min es de **1290mA**. Como podemos apreciar, el consumo ha bajado en **110mA** en comparación con el apartado 6.2.1.2 (1400mA)**.**

6.3 Conclusión

Después de ver todos los resultados, tanto del error cuadrático medio, como del consumo energético, deducimos que es mejor aplicar el algoritmo de predicción del periodo de muestreo cuando tenemos un conjunto de datos de gran variabilidad, mientras que, cuando tenemos un conjunto de datos con poca variabilidad, es mejor utilizar un periodo de muestreo fijo.

CAPÍTULO 7. CONCLUSIONES Y LÍNEAS DEL FUTURO

7.1 Conclusiones

En este apartado, se va a verificar el cumplimiento de los objetivos del TFG. A continuación, se presentan los puntos conseguidos.

- Se han configurado y programado varios nodos (NodeMCU), para que puedan leer información medioambiental (temperatura, humedad, y luminosidad), y enviarla a la base de datos de forma periódica.
- Se ha diseñado un servidor web, mediante el cual se puede consultar la información medioambiental almacenada en la base de datos de forma gráfica y en formato de tabla. También, se puede gestionar los nodos y los usuarios del sistema.
- Se ha diseñado un algoritmo para la predicción del periodo de muestreo, que, a partir de las variables medioambientales, obtenga el periodo de lectura óptimo (consumo energético ajustado con error razonable).

7.2 Líneas del futuro

- Desarrollar una aplicación móvil que tenga la misma funcionalidad que la aplicación web, con el fin de darle a los clientes un acceso más rápido y flexible a la información.
- Usar otras tecnologías para enviar información al servidor, como puede ser WIMAX, Zigbee, etc.
- Diseñar un SFA (Sistema Fotovoltaico Autónomo) para alimentar el sistema, que nos permita ahorrar más en los costes.
- Realizar un análisis más profundo a nuestro sistema borroso para poder introducir mejoras que nos permitan obtener un resultado más preciso.

CAPÍTULO 8. BIBLIOGRAFÍA

[1] https://agroconecta.es/ (último acceso: 05-12-2018)

[2] http://www.softcrits.es/ (último acceso: 05-12-2018)

[3] http://www.smartyplanet.com/ (último acceso: 05-12-2018)

[4] http://www.sayme.es/ (último acceso: 05-12-2018)

[5] Tesis doctoral: Estrategias evolutivas para la adquisición de conocimiento en controladores borrosos temporales difuminados, aplicadas al encaminamiento adaptativo en redes de comunicaciones, Manuel Angel Gadeo Martos, 2009.

[6] http://www.mfbarcell.es/conferencias/wsn.pdf

[7] https://www.arduino.cc (Último acceso: 10/12/2018)

[8] http://www3.ubu.es/ubuinvestiga/arduino-basico/

[9] https://www.amazon.es/ (Último acceso: 10/12/2018)

[10] https://mecatronicauaslp.wordpress.com/2014/02/28/introduccion-a-beaglebone/

[11] https://programarfacil.com/podcast/esp8266-wifi-coste-arduino/

[12] https://www.prometec.net/sensores-dht11/

[13]https://revistadigital.inesem.es/informatica-y-tics/los-gestores-de-bases-de-datos-mas-usados/

[14] https://www.vix.com/es/btg/tech/13220/los-mejores-ides-y-editores-para-programadores

[15] http://www.4rsoluciones.com/blog/bibliotecas-javascript-para-crear-graficos-interactivos/

[16] https://es.ourcodeworld.com/articulos/leer/71/top-5-mejores-librerias-para-crear-graficas-en-javascript

[17] https://www.mouser.com/ds/2/758/DHT11-Technical-Data-Sheet-Translated-Version-1143054.pdf

CAPÍTULO 9. ANEXOS

9.1 Manuales

En esta sección, se van a presentar dos manuales, un manual de usuario, en el cual se explica de forma simplificada al usuario cómo manejar la aplicación web alojada en el servidor, y un manual de instalación y mantenimiento de los nodos, donde se detalla cómo gestionar los nodos.

9.1.1 Manual de usuario

Cabe destacar que hay dos tipos de usuarios: un usuario normal, con permisos limitados, y un administrador con todos los permisos disponibles.

9.1.1.1 Usuario normal

En esta sección, se van a explicar todos los pasos seguidos por un usuario normal, desde la creación de la cuenta, hasta el uso de todas las herramientas que ofrece la aplicación.

El primer paso, es la creación de una nueva cuenta. Para ello, escribimos la siguiente dirección en el navegador web:

https://jaeniot.000webhostapp.com/pruebas/register.php

Nos sale un formulario para recoger nuestros datos:

Figura 9.1 Formulario de registro

Rellenamos el formulario con los datos correspondientes, y le damos a registrar.

Directamente, se redirigirá a la página principal (index.php).

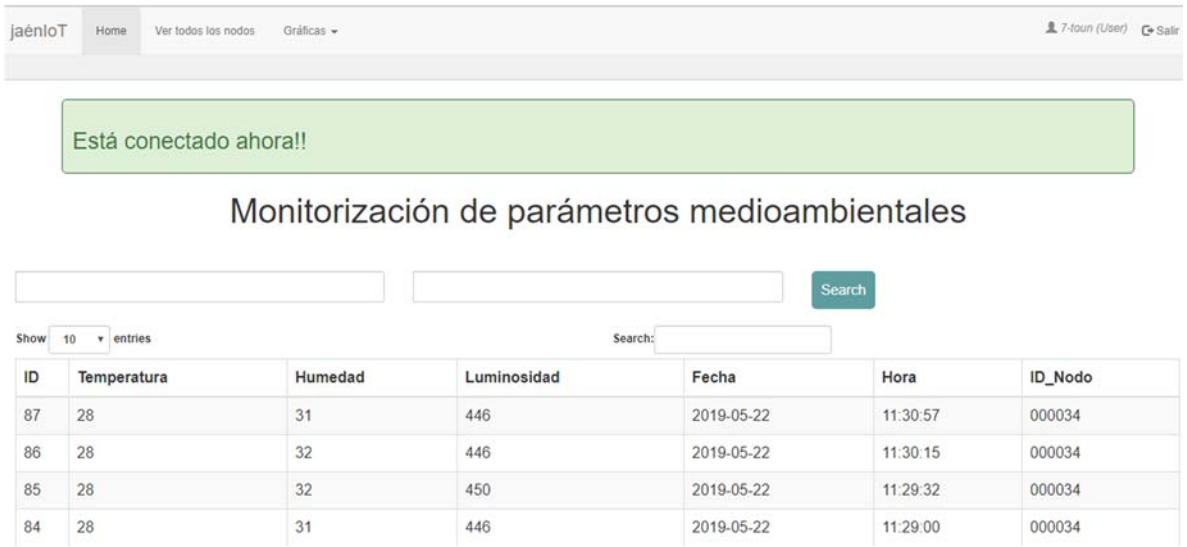

Figura 9.2 Página principal

Como podemos ver en la **figura 9.2**, los datos capturados por los nodos se pueden consultar de dos formas: en una tabla o de forma gráfica. También, tenemos la posibilidad de filtrar los datos entre dos determinadas fechas, y buscar los parámetros ambientales capturados por un determinado nodo.

Asimismo, tenemos implementado un mecanismo de paginación que nos permita mostrar los resultados de 10 en 10 en formato de tabla. La tabla muestra los datos de temperatura, humedad, luminosidad, fecha, hora e identificador único de cada nodo.

En la barra de navegación, a la derecha, podemos saber el nombre de usuario y su rol entre paréntesis. También, podemos cerrar la sesión pulsando el botón "salir".

Para saber los nodos del sistema y sus correspondientes identificadores, hacemos clic en la pestaña "Ver todos los nodos":

Figura 9.3 Ver todos los nodos

En seguida, se redirige a la página nodos.php.

ID	Nombre	identificador unico	Fecha de alta
2	arduino uno	000031	2019-01-15 18:14:22.0000
3	arduino due	000032	0000-00-00 00:00:00.0000
4	nodemcu	000034	2019-05-18 12:49:22.8719

Figura 9.4 nodos.php

Como se puede apreciar en la **figura 9.4**, tenemos toda la información relativa a los nodos del sistema: nombre del nodo, identificador único del nodo, y su fecha de alta.

Otra herramienta que tenemos en la barra de navegación, es un desplegable que nos permita ver los datos de temperatura, humedad, y luminosidad de forma gráfica.

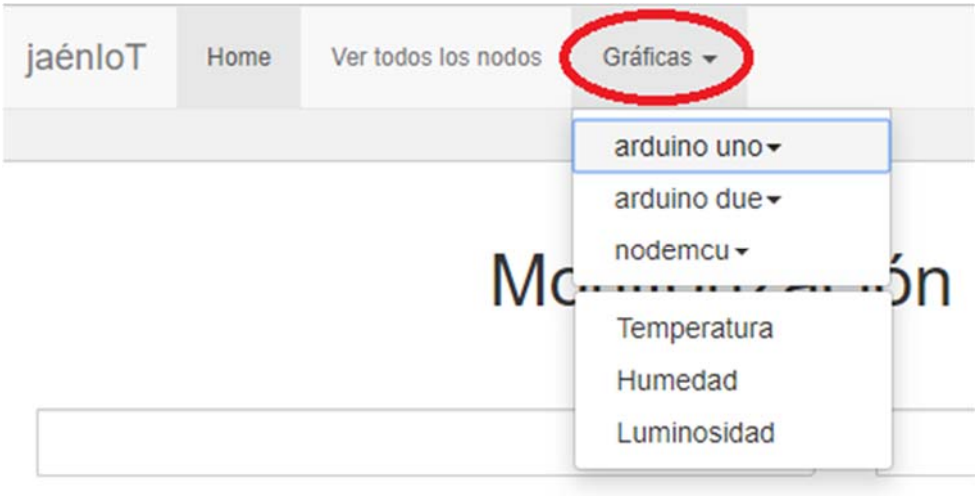

Figura 9.5 Gráficas

A continuación, se va a mostrar un ejemplo de la evolución de la temperatura en forma gráfica.

Figura 9.6 Ejemplo de evolución de temperatura

En la gráfica, tenemos un eje de abscisas que representa el tiempo, y un eje de ordenadas que representa los valores del parámetro monitorizado, en este caso, la temperatura. En la parte inferior, disponemos de un cursor, que, variando su anchura, podamos ver más o menos datos.

En la parte superior derecha, tenemos un botón para imprimir la gráfica en varios formatos:

- Imagen PNG y JPEG.
- SVG vectorial.
- Documento PDF.

9.1.1.2 Usuario admin.

El usuario admin tiene la posibilidad de gestionar los usuarios y los nodos del sistema (dar de alta, modificar, y borrar). La única diferencia entre la página principal de un usuario normal y un usuario admin es la pestaña de administración.

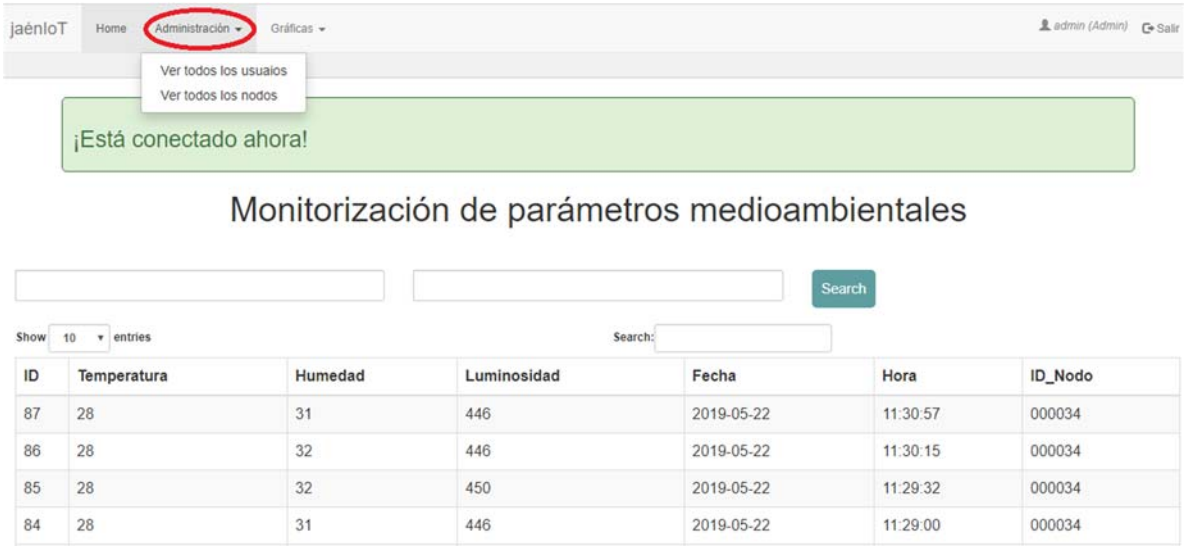

Figura 9.7 Página principal usuario admin

Para gestionar los usuarios del sistema, pinchamos sobre "ver todos los usuarios" del desplegable "administración". En seguida, se redirige a la página de usuarios (users.php).

Figura 9.8 users.php

Como se puede ver en la **figura 9.8**, tenemos toda la información relativa a los usuarios:

- Nombre de usuario.
- Email.
- Tipo de usuario: usuario normal o admin.
- Fecha de alta.

También, tenemos la posibilidad de borrar un usuario, o editar sus datos.

En la parte inferior, tenemos un enlace que nos redirige a un formulario para la creación de un nuevo usuario.

19	zineb	zineb@el.com	user
20	zarga	zarga@monamour.com	user
21	7-toun	toun@gmail.com	user

+ Añadir nuevo usuario

Figura 9.9 añadir un nuevo usuario

Figura 9.10 Creación de un nuevo usuario

De igual forma, tenemos otra página (nodos.php) para la gestión de nodos.

Figura 9.11 Nodos Del Sistema

En la página nodos.php, tenemos toda la información relativa a un nodo:

- Nombre del nodo.
- Identificador único del nodo.
- Fecha de alta del nodo.

Asimismo, tenemos la posibilidad de borrar un nodo, o editar sus datos.

En la parte superior izquierda de la página, tenemos un enlace que nos redirige a un formulario para insertar un nuevo nodo.

Figura 9.12 añadir un nuevo nodo

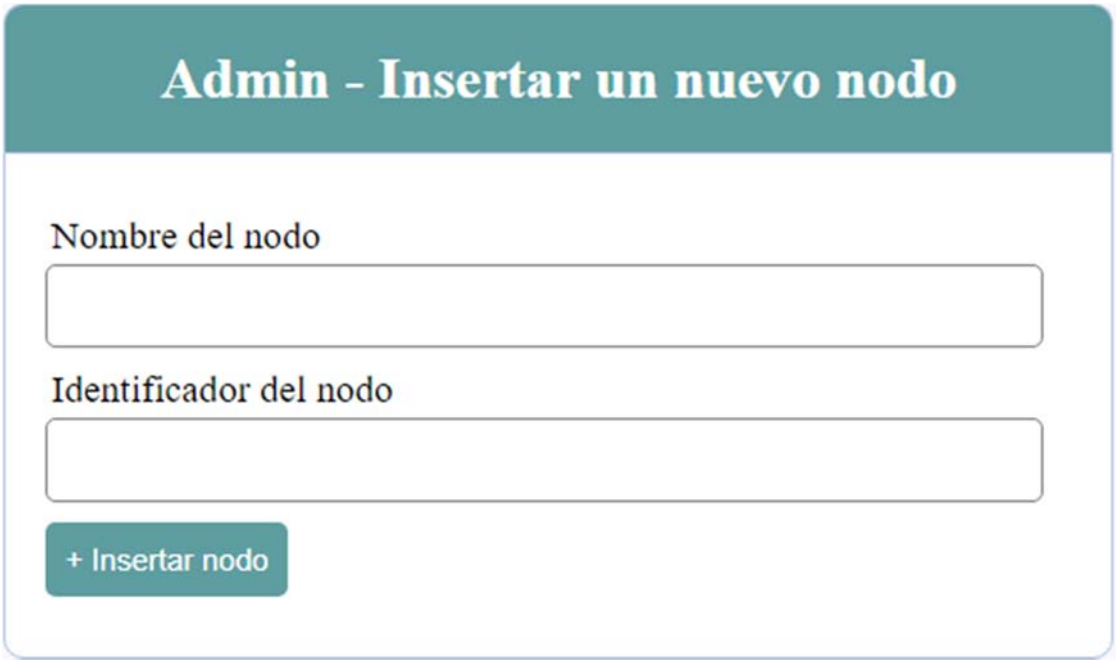

Figura 9.13 Inserción de un nuevo nodo

9.1.2 Manual de instalación y mantenimiento de los nodos.

En esta sección, se va a explicar al usuario final del servicio, como subir el código a los nodos y que modificaciones son necesarias para el correcto funcionamiento de los mismos.

9.1.2.1 Descarga e instalación del IDE de Arduino.

El primer paso es la descarga y la instalación del IDE (Entorno de Desarrollo Integrado) de Arduino. Para ello, abrimos un navegador web y escribimos la siguiente dirección:

https://www.arduino.cc/en/Main/Software

Nos va a salir la siguiente pantalla:

Figura 9.14 Descarga IDE Arduino (1)

Como podemos ver en la **figura 9.14**, tenemos varias opciones, dependiendo del sistema operativo que tengamos (Windows, Mac OS X, Linux). Si tenemos instalado el sistema operativo Windows, es conveniente elegir la opción "**Windows Installer, for Windows XP and up**", porque en ella vienen todos los drivers integrados en el instalador, y no hace falta instalarlos por separado.

Figura 9.15 Descarga IDE Arduino (2)

Después de la descarga, seguimos todos los pasos hasta el final de la instalación. A continuación, vamos a mostrar algunas capturas del proceso de instalación.

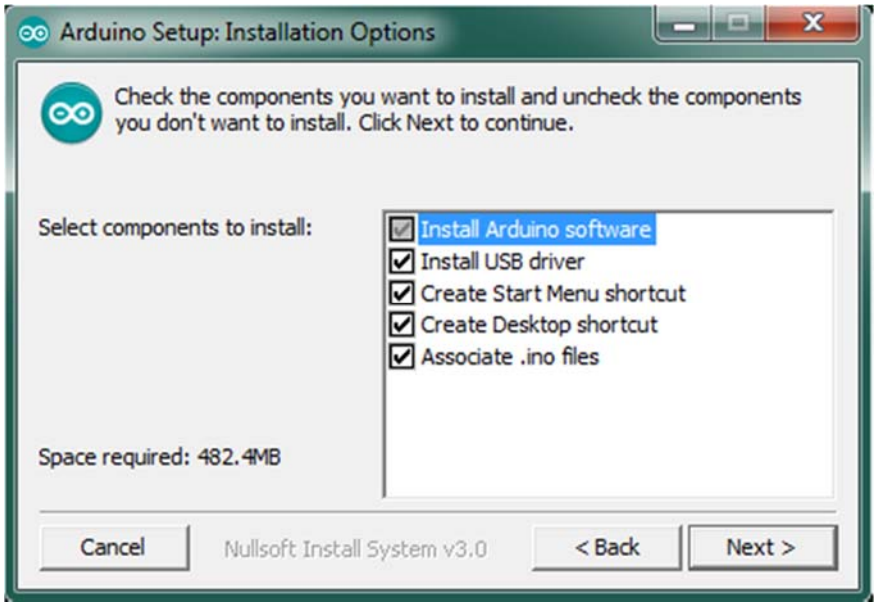

Figura 9.16 Proceso de instalación (1)

Como se ve en la **figura 9.16**, debemos marcar todas las casillas, para instalar todas las herramientas necesarias.

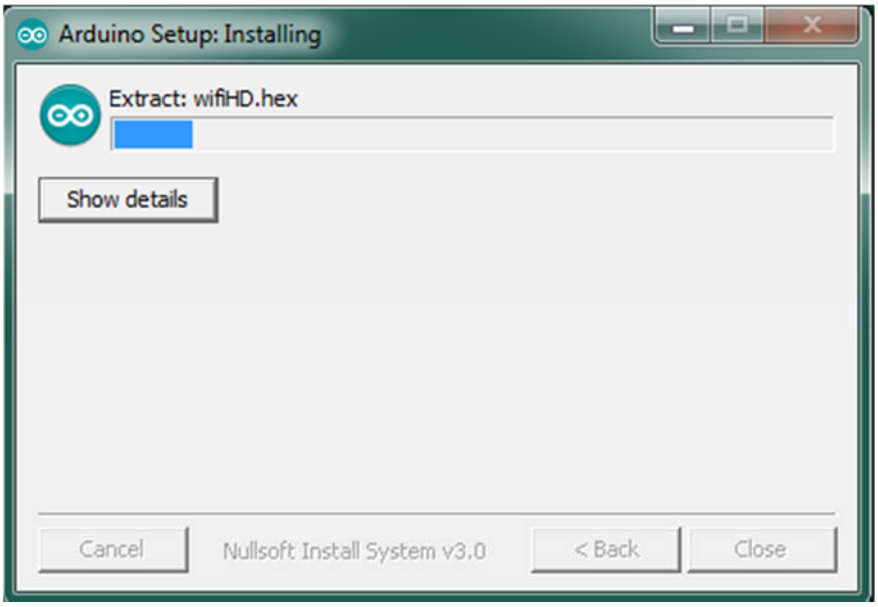

Figura 9.17 Proceso de instalación (2)

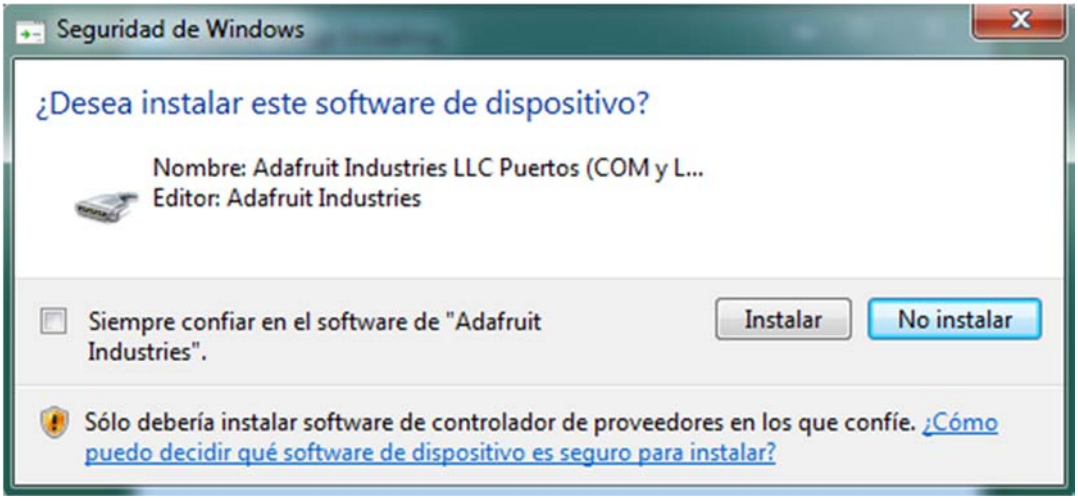

Figura 9.18 Proceso de instalación (3)

Al final del proceso de instalación, el IDE de Arduino va a tener la siguiente apariencia:

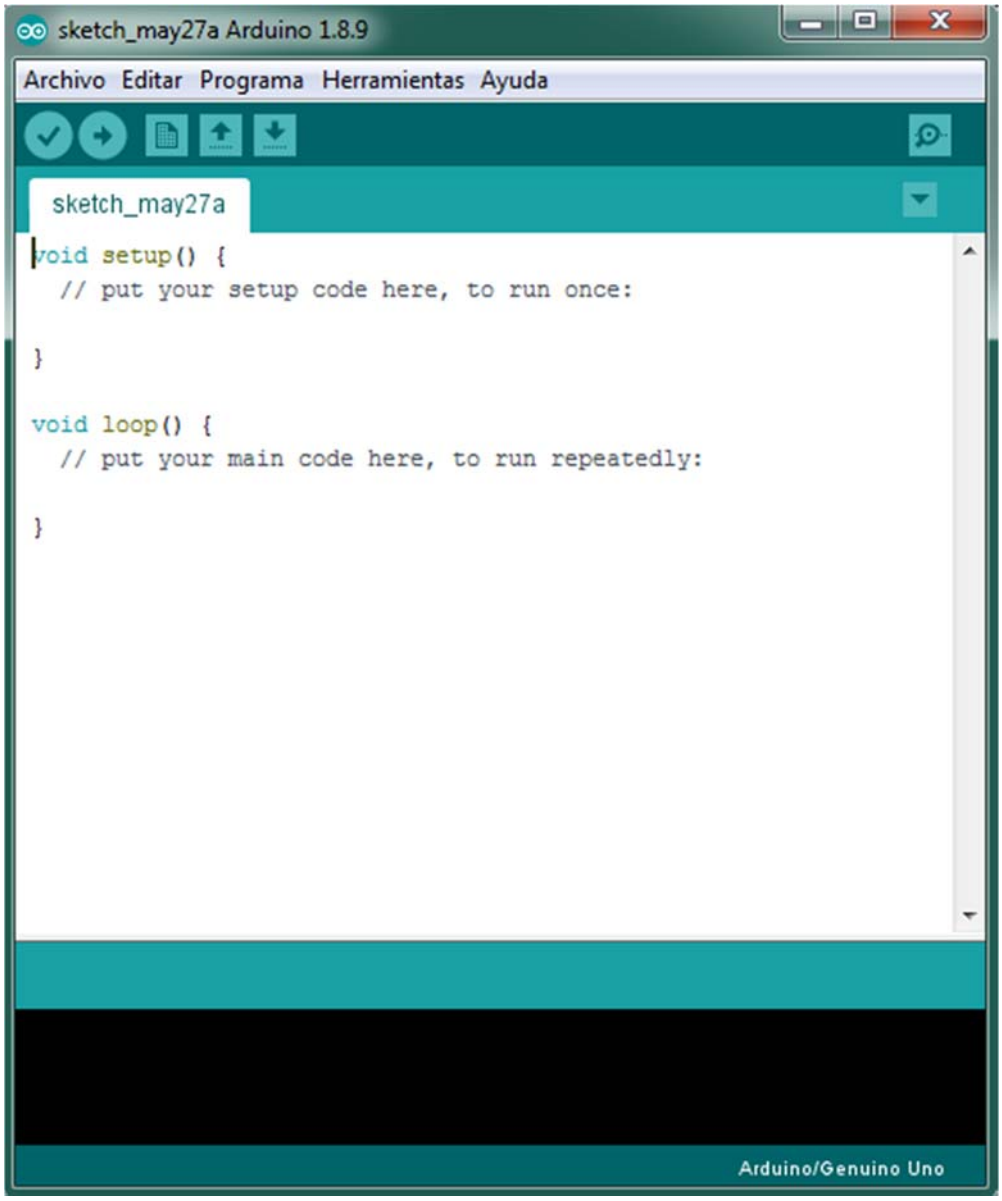

Figura 9.19 IDE de Arduino

9.1.2.2 Descargar e instalar el pluggin ESP8266.

Para la descarga e instalación del pluggin ESP8266, abrimos el IDE de Arduino, y vamos a **Archivo → Preferencias**:

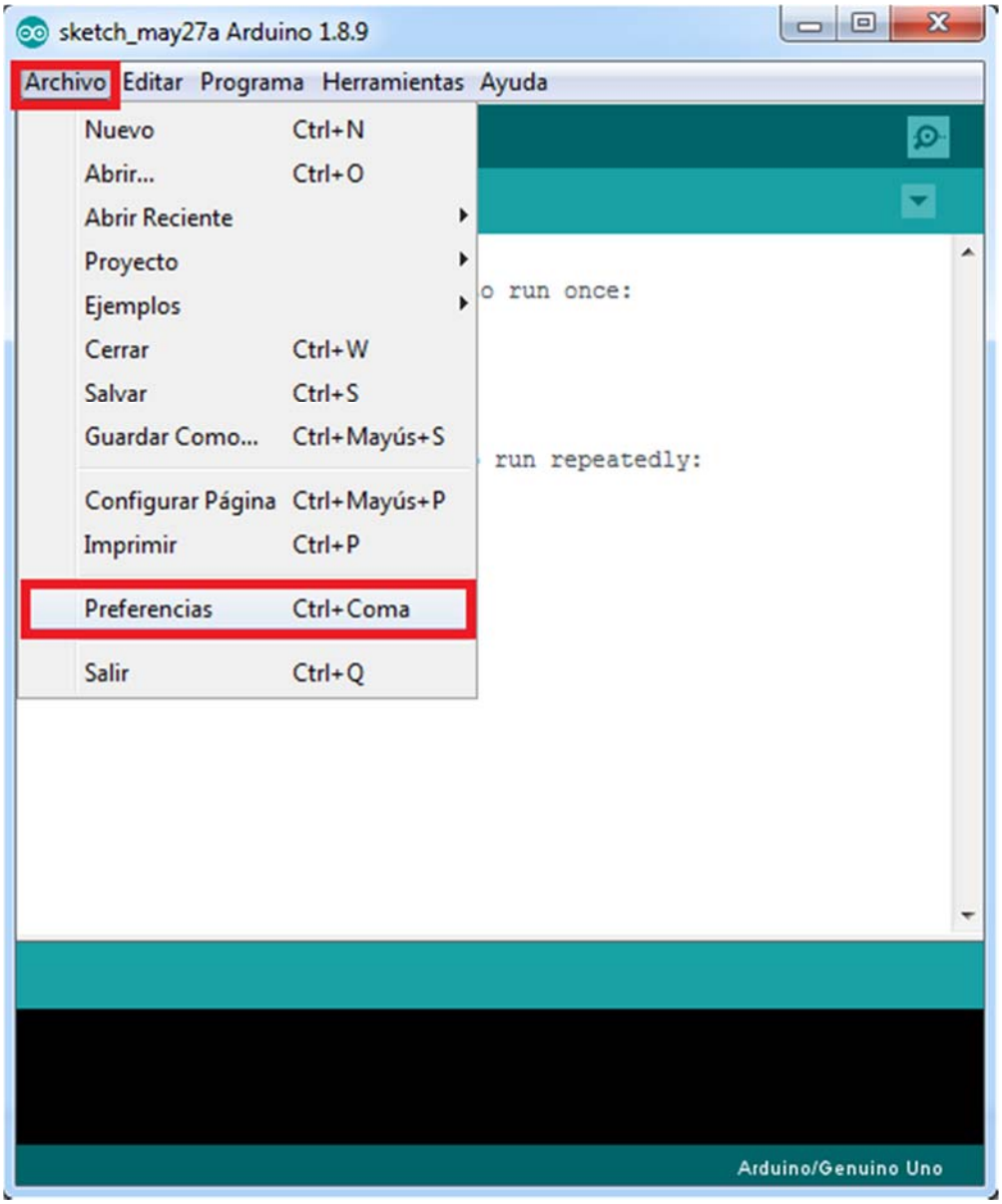

Figura 9.20 pluggin de esp8266 (1)

Después de pinchar en Preferencias, aparecerá la siguiente página de configuración:

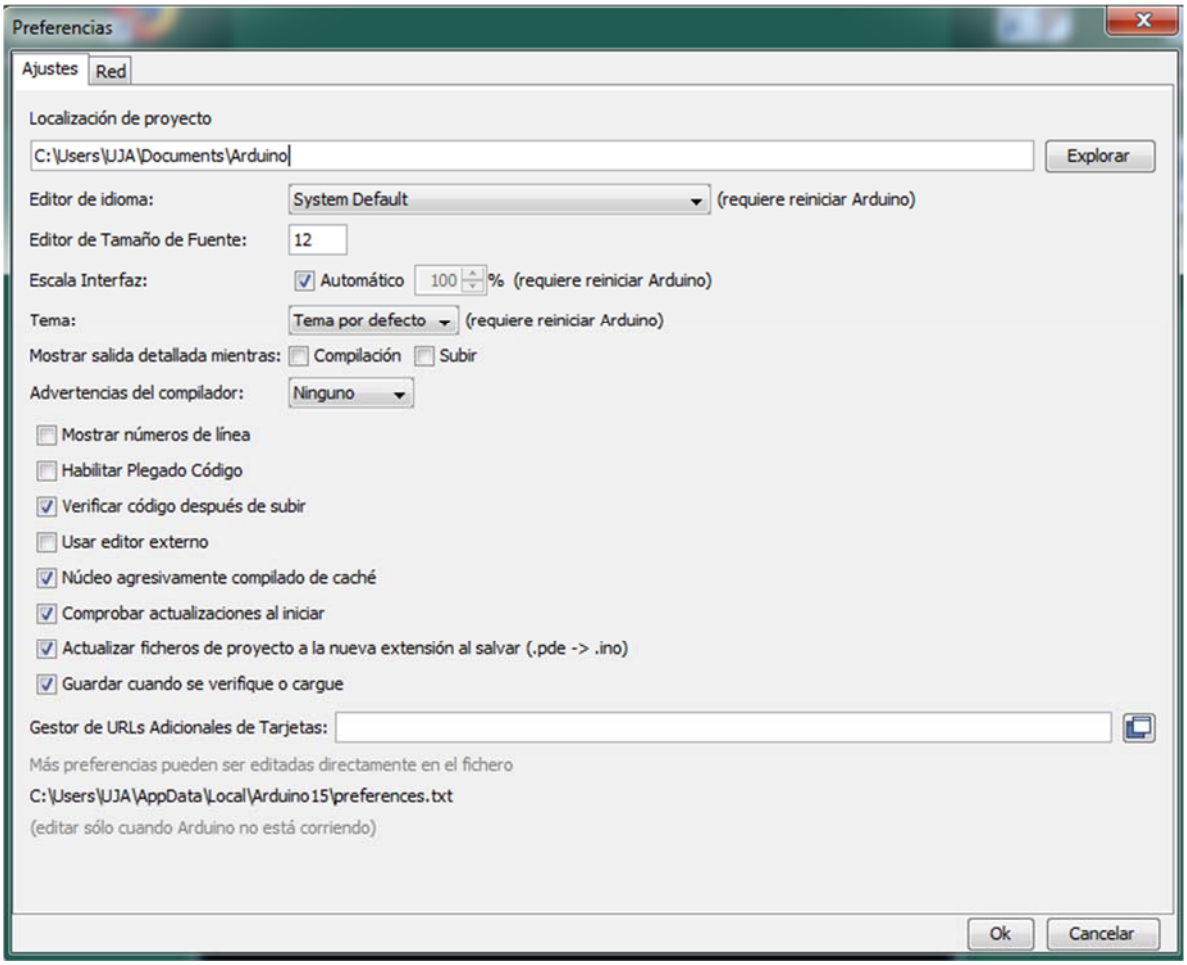

Figura 9.21 pluggin de esp8266 (2)

En "**Gestor de URLs Adicionales de Tarjetas"**, escribimos la siguiente dirección:

http://arduino.esp8266.com/stable/package_esp8266com_index.json

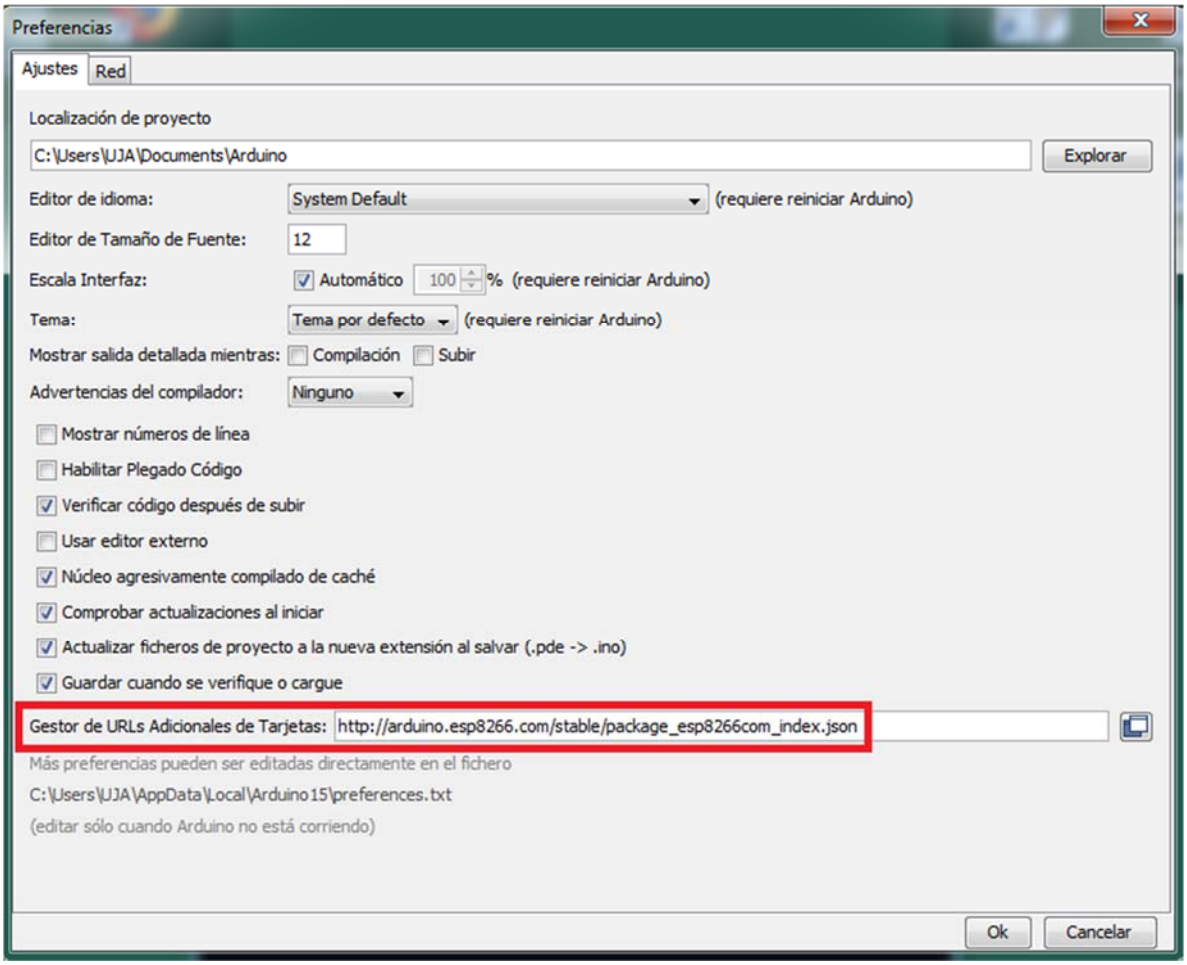

Figura 9.22 pluggin de esp8266 (3)

Le damos al Ok, y vamos a **Herramientas → Placa → Gestor de tarjetas…**, y buscamos "**esp8266**":

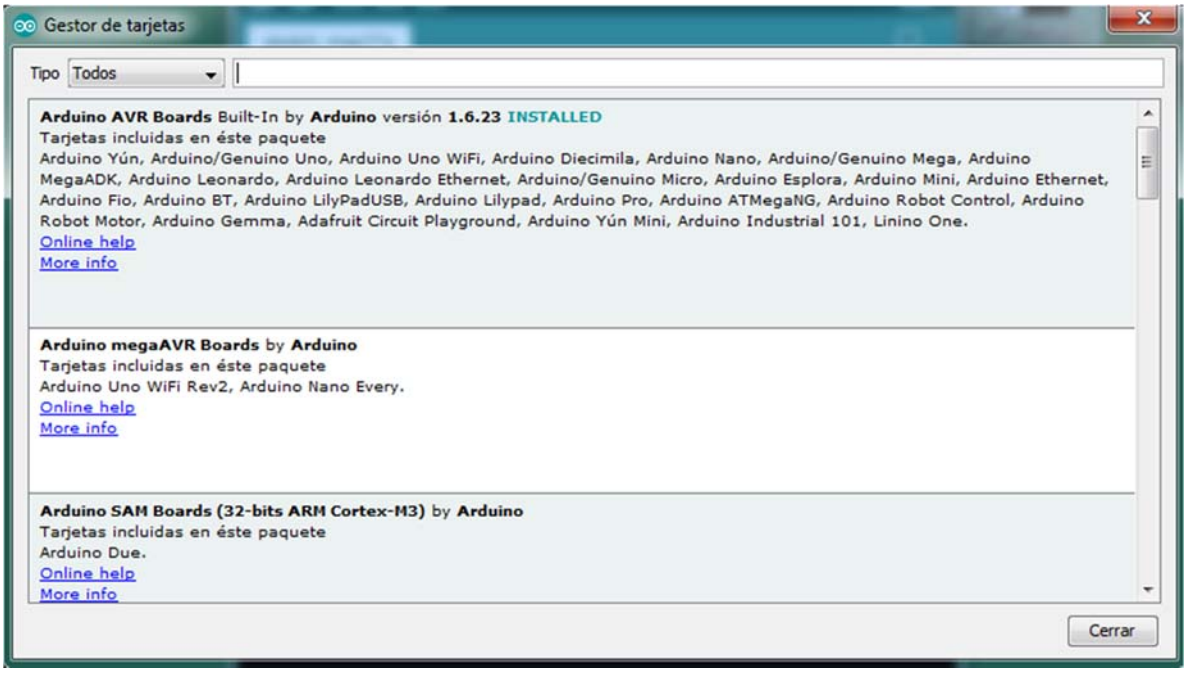

Figura 9.23 pluggin de esp8266 (4)

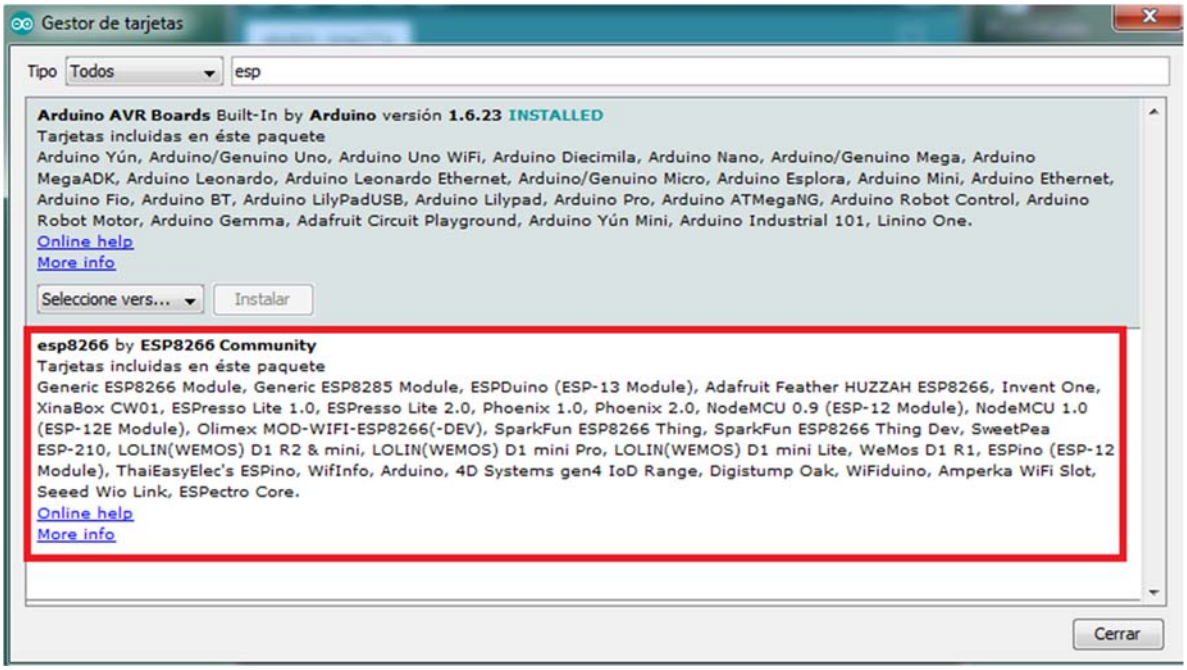

Figura 9.24 pluggin de esp8266 (5)

Le damos a **instalar** y ya podemos compilar y ejecutar sketchs en los NodeMCU.

9.1.2.3 Programación de los nodos.

Después de conectar el NodeMCU al ordenador mediante un cable USB nano, abrimos el IDE de Arduino y seleccionamos la placa (en nuestro caso, NodeMCU 1.0 (ESP-12E Module)), y el puerto correspondiente.

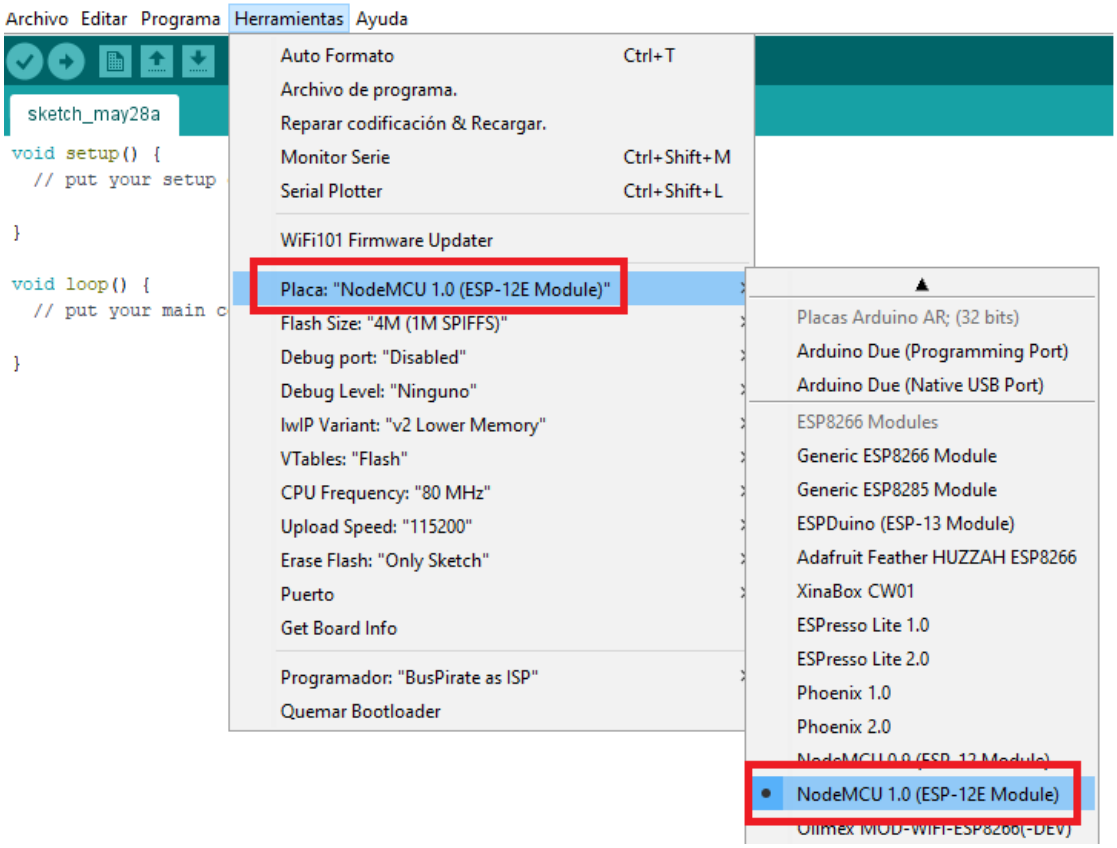

Figura 9.25 Programación NodeMCU (1)

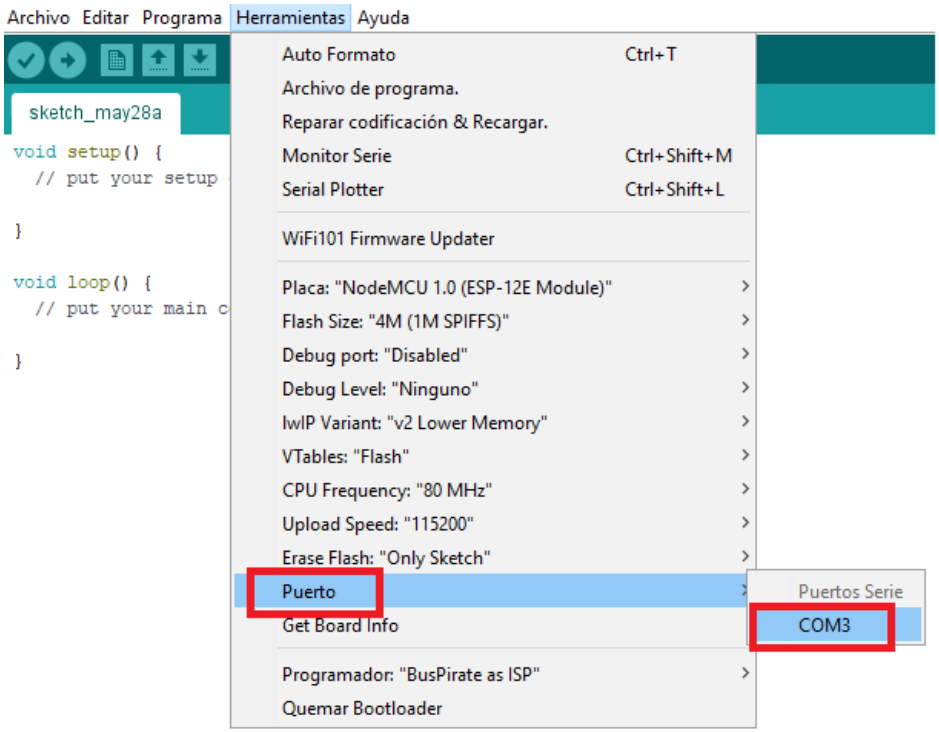

Figura 9.26 Programción NodeMCU (2)

También, debemos configurar los siguientes parámetros:

- **Flash Size**: “4M (1M SPIFFS)”.
- **Debug port**: “Disabled”.
- **Debug Level**: “Ninguno”.
- **lwIP Variant**: “v2 Lower Memory”.
- **VTables**: “Flash”.
- **CPU Frequency**: “80MHz”.
- **Upload Speed**: “115200”.
- **Erase Flash**: “Only Sketch”.

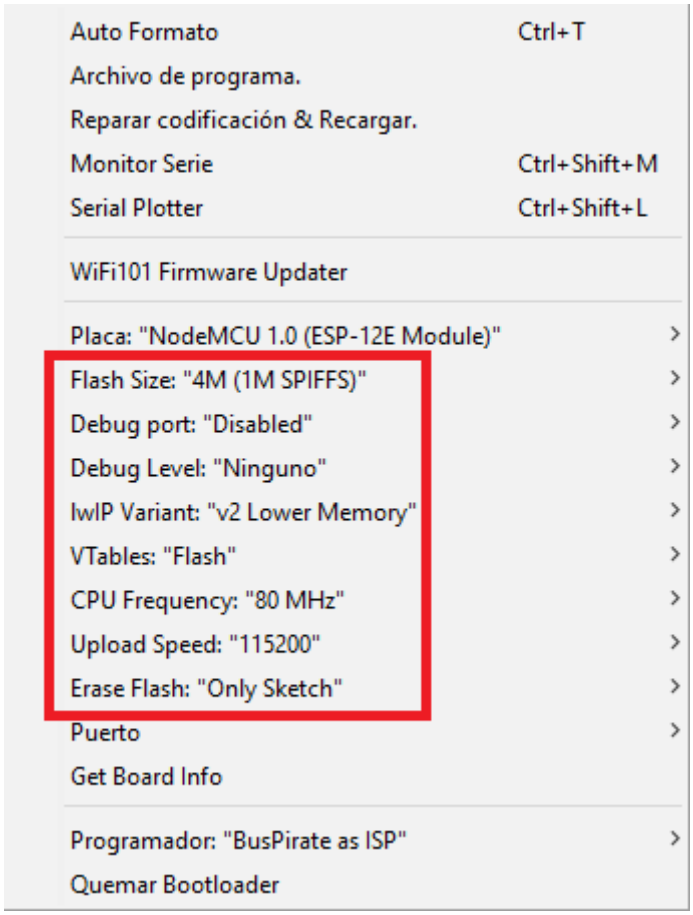

Figura 9. 27 Programación NodeMCU (3)

Ya hemos configurado todos los parámetros necesarios para compilar y ejecutar nuestro sketch. A continuación, vamos a mostrar las modificaciones necesarias dentro del código para poder conectar al servidor y enviar los parámetros ambientales.

```
#include <DHT11.h>
#include <ESP8266WiFi.h>

DHT11 dht11(D4);

const char* ssid     = "xxxxxxxx-xxxx";
const char* password = "xxxxxxxx";

const char* host = "xxxxxxxxx.xxxx.xxx";

void setup()
{
  Serial.begin(9600);
  // We start by connecting to a WiFi network

  Serial.println();
  Serial.println();
  Serial.print("Connecting to ");
  Serial.println(ssid);

  WiFi.begin(ssid, password);
```

Figura 9.28 Sketch NodeMCU

Como podemos ver en la **figura 9.28**, solo tenemos que cambiar tres parámetros dentro del código: nombre de la red wifi (ssid), contraseña de acceso (password), y dirección del servidor (host). Este último, se puede dejar sin cambio, si el cliente quiere usar nuestro servidor gratuito "**jaeniot**".

9.2 Presupuesto

En este apartado, se va a calcular el coste de los equipos usados en este TFG.

Equipo	Unidades	Precio unitario en €	Precio total en €
NodeMCU	1	7,49	7,49
Sensor DHT11	1	1,44	1,44
Sensor LDR	1	1,75	1,75
Resistencia 10k	1	0,15	0,15
Fuente de alimentación MB102 y accesorios	1	11,28	11,28
Regulador de tensión	1	3	3

El coste de los equipos **para un solo nodo** de la red de sensores inalámbricos es de **25,11 euros**.

A este coste le tenemos que añadirle el coste de los recursos humanos.

Actividad	Horas	Precio unitario en €	Precio total en €
Diseño	450	30	13500
Configuración	4	30	120
Instalación	1	25	25

El coste total de diseño, configuración e instalación es de **13645 euros**.

Ahora, vamos a proceder a calcular el coste total con IVA.

Coste	Precio en €
Equipos	25,11
Recursos humanos	13645
Total sin IVA	13670,11
IVA	21%
Total con IVA	16540,83

El coste total del trabajo fin de grado asciende a **DIECISÉIS MIL QUINIENTOS CUARENTA EUROS CON OCHENTA Y TRES CENTIMOS**.

También, tenemos un coste indirecto mensual que es la conexión a internet, para conectar los nodos con el servidor.

Coste	Precio en €
Conexión internet	25 euros

El coste indirecto es de **VEINTE CINCO EUROS**.