Internet de las cosas para seguridad y domótica

Ing. Alfonso Jacinto Agama Chico, MACI.

Internet de las cosas para seguridad y domótica
Universidad Técnica de Babahoyo

ISBN: 978-9942-606-19-8 (eBook)

Editado por:
Universidad Técnica de Babahoyo
Avenida Universitaria Km 2.5 Vía a Montalvo
Teléfono: 052 570 368

Babahoyo, Ecuador
www.utb.edu.ec
E-mail: editorial@utb.edu.ec
Este texto ha sido sometido a un proceso de evaluación por pares externos.
Diseño y diagramación, montaje y producción editorial
Universidad Técnica de Babahoyo

Babahoyo – Los Ríos – Ecuador

Alfonso Jacinto Agama Chico

Facultad de Administración Finanzas e Informática

Universidad Técnica de Babahoyo

aagama@utb.edu.ec

https://orcid.org/0000-0002-2839-5126

Carlos Alfredo Cevallos Monar

Facultad de Administración Finanzas e Informática

Universidad Técnica de Babahoyo

acevallos@utb.edu.ec

https://orcid.org/0000-0002-6751-6998

Ricardo Vicente García Paredes

Facultad de Administración Finanzas e Informática

Universidad Técnica de Babahoyo

rgarciap@utb.edu.ec

https://orcid.org/0000-0002-1831-891X

Erick Alfredo Saa Litardo

Facultad de Administración Finanzas e Informática

Universidad Técnica de Babahoyo

esaa841@fafi.utb.edu.ec

https://orcid.org/0000-0002-8997-106X

Emerson Gabriel Baldeón Navarrete

Facultad de Administración Finanzas e Informática

Universidad Técnica de Babahoyo

ebaldeon039@fafi.utb.edu.ec

https://orcid.org/0000-0001-9043-3034

Octavio Alexander Bajaña Ortiz

Facultad de Administración Finanzas e Informática

Universidad Técnica de Babahoyo

obajana894@fafi.utb.edu.ec

https://orcid.org/0000-0002-5020-3100

José Manuel Arias Moran

Facultad de Administración Finanzas e Informática

Universidad Técnica de Babahoyo

jarias@fafi.utb.edu.ec

https://orcid.org/0000-0001-5263-5604

Juan Fernando Saa Ayala

Facultad de Administración, Finanzas e Informática

Universidad Técnica de Babahoyo

jsaayala@fafi.utb.edu.ec

https://orcid.org/0000-0001-7996-0757

PRESENTACIÓN

Bienvenidos al apasionante mundo del Internet de las Cosas (IoT,) y su aplicación en la seguridad y la domótica. Este libro ha sido creado con el propósito de proporcionar a estudiantes y docentes de carreras relacionadas con las tecnologías de la información una valiosa fuente de consulta para el desarrollo de aplicaciones en este fascinante campo.

El Internet de las Cosas se ha convertido en una tendencia revolucionaria, que conecta objetos cotidianos a la red, permitiéndoles intercambiar información y tomar decisiones inteligentes de manera autónoma. Este libro se ha concebido como una guía práctica y accesible, diseñada para facilitar a los lectores la creación de sus propias soluciones de IoT a través de una serie de pasos claros y el uso de herramientas disponibles en el estudio.

En el corazón del libro se encuentra en el Capítulo IV, donde se aborda el desarrollo de aplicaciones concretas en el campo de la seguridad y la domótica. A través de ejemplos prácticos y explicaciones detalladas, los lectores aprenderán a crear soluciones personalizadas que respondan a sus necesidades específicas.

Es importante destacar que este libro no habría sido posible sin el valioso aporte de los docentes y estudiantes de la Universidad Técnica de Babahoyo, pertenecientes a la Facultad de Administración, Finanzas e Informática. Fruto del arduo trabajo realizado en un Semillero y Proyecto de Investigación sobre el Internet de las Cosas, esta obra es el resultado de su dedicación y conocimiento en la materia.

En conclusión, "Internet de las Cosas para Seguridad y Domótica" se presenta como un recurso esencial para aquellos interesados en adentrarse en el fascinante mundo del IoT. Esperamos que este libro sea una guía valiosa y una fuente de inspiración para el desarrollo de nuevas soluciones y proyectos que contribuyan al avance de esta emocionante tecnología.

¡Adelante, sumérgete en el universo del Internet de las Cosas y déjate inspirar por las infinitas posibilidades que ofrece!

Alfonso Agama, Ricardo García, Carlos Cevallos, Erick Saa, Emerson Baldeón, José Arias, Octavio Bajaña, Juan Saa.

Universidad Técnica de Babahoyo

Internet de las cosas para seguridad y domótica

INTRODUCCIÓN

El presente estudio se realizó con el propósito de permitirle a los estudiantes y docentes de las carreras de tecnologías de la información y afines, contar con una fuente de consulta para el desarrollo de aplicaciones en el internet de las cosas, para poder crear sus propias soluciones de manera sencilla, mediante una serie de pasos y herramientas que podrán encontrar en el presente estudio. El presente trabajo consta de 5 capítulos, los cuales son los siguientes:

Capítulo I: Introducción al internet de las cosas, en este capítulo se encuentran los fundamentos sobre todo lo relacionado a poder interconectar dispositivos mediante el internet para hacer la vida del ser humano más sencilla.

Capitulo II: Los módulos ESP 32/8266 y dispositivos complementarios, este capítulo trata sobre conocer las características de los módulos ESP 32 Y ESP 8266, y los dispositivos como sensores, librerías y aplicaciones con las que se pueden combinar para obtener una solución informática.

Capitulo III: Fundamentos de programación, en este capítulo se dan a conocer las bases teóricas para programar en el entorno de desarrollo de Arduino y el lenguaje de programación Python, además de algunas temáticas complementarias para poder crear soluciones mediante la programación.

Capítulo IV: Desarrollo de aplicaciones, este capítulo contiene algunos ejemplos de aplicaciones utilizando todo lo aprendido en los anteriores capítulos, tales como aplicaciones usando Web Servers, autentificaciones, dispositivos electrónicos, entre otros, con el propósito de entender por medio de la practica todo lo aprendido en el Semillero de Investigación.

Capítulo V: Protocolos de comunicación para el IoT, en este capítulo se abordan los aspectos teóricos para conocer cómo se realiza la comunicación entre distintos dispositivos al mismo tiempo.

Todos los capítulos mencionados contienen información relevante sobre los dispositivos que están marcando tendencia actualmente y que prometen ayudar en un futuro a la humanidad en distintos aspectos de su vida cotidiana, laboral e industrial.

ÍNDICE

CAPÍTULO I – Internet de las cosas

Introducción al IoT

El internet de las cosas, más conocido como IoT, se refiere a la posibilidad de interconectar dispositivos físicos cotidianos con el internet, tales como electrodomésticos, luces, accesorios personales como relojes, entre otros dispositivos, con el propósito de que estos objetos puedan intercambiar información entre sí, sin necesidad de la interacción de los seres humanos o computadoras para realizarlo.

El proceso de comunicación de estos dispositivos IoT se realizan a través de un tipo de conectividad, como un cable, Bluetooth, WiFi, entre otros, gracias a esto, los dispositivos que intervienen en la comunicación recolectan la información, la procesan y posteriormente son analizadas para ayudar al usuario a sacar conclusiones, como por ejemplo el hábito de encender/apagar las luces de su hogar.

Figura 1

Dispositivos conectados bajo un mismo Ecosistema IoT

Nota. Tomada de *Smart IoT* [Imagen], por MagazIEEE Ecuador, 2021 (https://r9.ieee.org/ecuador-magaz/domotica-smart-iot). CC BY 2.0

El uso del internet de las cosas es muy diverso, puede usarse tanto en aspectos personales como un hogar inteligente o también se puede emplear en empresas para mejorar sus modelos comerciales, afianzar las relaciones con los clientes, estudiar el hábito de los consumidores, entre otras actividades, para asegurar la continuidad y la mejora del negocio.

Internet de las cosas para hogares inteligentes SHIoT y domótica

Los avances tecnológicos en los últimos años han sido sorprendentes, actualmente se está experimentando una evolución en el uso de nuestros dispositivos, a través de gadgets para los hogares, acercándonos hacia la era del internet de las cosas. El propósito de esta nueva tecnología es poder automatizar diferentes actividades dentro del hogar, proporcionando una mayor comodidad para el usuario, es así como esto recibe el nombre de hogar inteligente.

Un hogar inteligente se refiere a la incorporación de tecnologías dentro del hogar, pero también hace referencia a las comunicaciones que permiten gestionar y automatizar las tareas cotidianas del hogar desde un mismo sistema. Estas tecnologías informáticas comunicadas les permiten a los usuarios del hogar tener una mejor calidad de vida, teniendo en cuenta que los objetivos principales de un hogar inteligente son: mejorar la comodidad, seguridad y ahorrar energía. (Custodio & Wilfredo, 2016)

Figura 2

Hogar controlado por internet

Nota. Tomada de *Casas Inteligentes* [Imagen], por El Economista, 2021 (https://eleconomista.com.ar/economia/casas-inteligentes-boom-inmobiliario-crecimiento-n41973). CC BY 2.0

CAPÍTULO II – Los módulos ESP32 y ESP8266

Los ESP32 y ESP8266

ESP8266 es un sistema en un chip (SoC por sus siglas en ingles), que contiene distintos componentes en un mismo circuito integrado teniendo entre sus características principales un procesador de 32 bits y un chip con conectividad wifi con gestión de pila TCP/IP.

La tarjeta ESP32 es un microcontrolador con las capacidades incorporadas de poner establecer una conexión mediante una red wifi o bluetooth, esta característica es la principal diferencia de su antecesora la ESP8266. Sus creadores la describen como un sistema en un Chip de bajo poder y bajo costo, diseñada para dispositivos móviles y aplicaciones de IoT.

Tanto ESP32 como ESP8266 cuentan con dos presentaciones la primera es simplemente el microcontrolador el cual necesita integrarse a una tarjeta para poder realizar las conexiones y empezar a trabajar, la segunda presentación es una tarjeta fabricada que cuenta con sus respectivos pines.

Características

ESP8266 es un chip encapsulado de propósito general que cuenta con conectividad wifi además cuenta con un procesador de 32 bits que funciona entre velocidades de 80 MHz y 160 MHz, lamentablemente no cuenta con una memoria flash por lo cual tiene que agregársela al momento de empezar a trabajar, es importante recalcar que la memoria varía entre modelos de 1 MB a 8 MB, siendo el máximo 16 MB

La ESP32 al ser una versión mejorada de la ESP8266 cuenta con varias mejoras en sus componentes siendo la principal su conectividad inalámbrica ya que no solo cuenta con wifi sino le añade el uso de bluetooth, además de poseer:

- Interruptores de antena
- Balun de RF
- Amplificador de potencia
- Amplificador de recepción de bajo ruido
- Filtros y módulos de administración de energía

También cuenta con funciones para ahorrar energía, así como pines sensibles al tacto que servirán para activarla después de entrar en modo de sueño profundo. En la Tabla 1 se mostrarán las características técnicas principales de las tarjetas ESP8266 y ESP32:

Tabla 1

Características Técnicas ESP8266 VS ESP32

Características Técnicas ESP32 VS ESP8266		
Características	ESP8266	ESP32
Microprocesador	Xtensa Single-core 32-bit L106	Xtensa Dual-Core 32-bit LX6 con 600 DMIPS
Alimentación	3.0 a 3.6V	2.2 a 3.6V
Wi-Fi	HT20	HT40
Bluetooth	No Posee	Bluetooth 4.2 y BLE
Frecuencia operativa	80 MHz	160 MHz
SRAM	No posee	448 KB
Flash	No posee	520 KB
GPIO	17	34
PWM(Hardware)	No posee	No posee
PWM(Software)	8 canales	16 canales
SPI	2	4
I2C	1	2
I2S	2	2
UART	2	2

ADC	10-bits de resolución	12-bits de resolución
CAN	No	Si
Interfaz MAC Ethernet	No	Si
Sensor de tacto	No	Si
Sensor de temperatura	No	Si (versiones antiguas)
Sensor de efecto hall	No	Si
Temperatura de trabajo	-40ºC a 125ºC	-40ºC to 125ºC

Nota: Se detallan las principales diferencias entre ESP8266 VS ESP32.

Esta tabla ha sido adaptada de “ESP32 Wifi y Bluetooth en un solo chip”, por ProgramarFacil.com, 2022 (https://programarfacil.com/esp8266/esp32).

Los módulos de ESP8266 y ESP32 como ya se describió con anterioridad cuentan con dos presentaciones, hasta ahora hemos hablado de las tarjetas prefabricada que cuentan con sus pines listo para realizar conexiones, la otra presentación es solo el microprocesador o como los describe la empresa fabricante “Sistema en Chip”, el cual tiene sus componentes comprimidos en un solo chip, en estos chips wroom encontramos los siguientes.

Chip WROOM

ESP-WROOM-02D: Es un módulo basado en ESP8266 que cuenta con optimización en RF (radiofrecuencia), sus dimensiones son 18x20x3.2 milímetros, cuenta con 18 pines una memoria flash de 2,4 MB y una antena PCB.

Figura 3

Tarjeta ESP WROOM 02D

Nota. Tomada de *ESP-WROOM-02D* [Imagen], Reichelt Elektronik, 2022 (https://www.reichelt.com/de/en/wifi-smd-module-esp8266ex-2-mb-spi-3-3-v-18-x-20-x-3-2-mm-esp-wroom-02d-p299996.html). CC BY 2.0

ESP-WROOM-02U: Al igual que el módulo anterior su infraestructura también está basada en ESP8266 y cuenta con la optimización RF además un conector UFL para conectar una antena externa, es un poco más pequeño que el modelo 02D con dimensiones de 18x14.3x3.2 milímetros, la cantidad de pines y la memoria flash es la misma, varia en la antena ya que este cuenta con una antena IPEX, gracias a esta antena se puede mejorar el alcance de la señal wifi, estas operan a una frecuencia de 2,4GHz con una ganancia de 3dB.

Figura 4

Tarjeta ESP-WROOM-02U

Nota. Tomada de *ESP-WROOM-02U* [Imagen], Reichelt Elektronik, 2022 (https://www.reichelt.com/pl/en/wifi-smd-module-esp8266ex-2-mb-spi-3-3-v-18-x-14-3-x-3-2-mm-esp-wroom-02u-p299999.html). CC BY 2.0

Módulos Comerciales

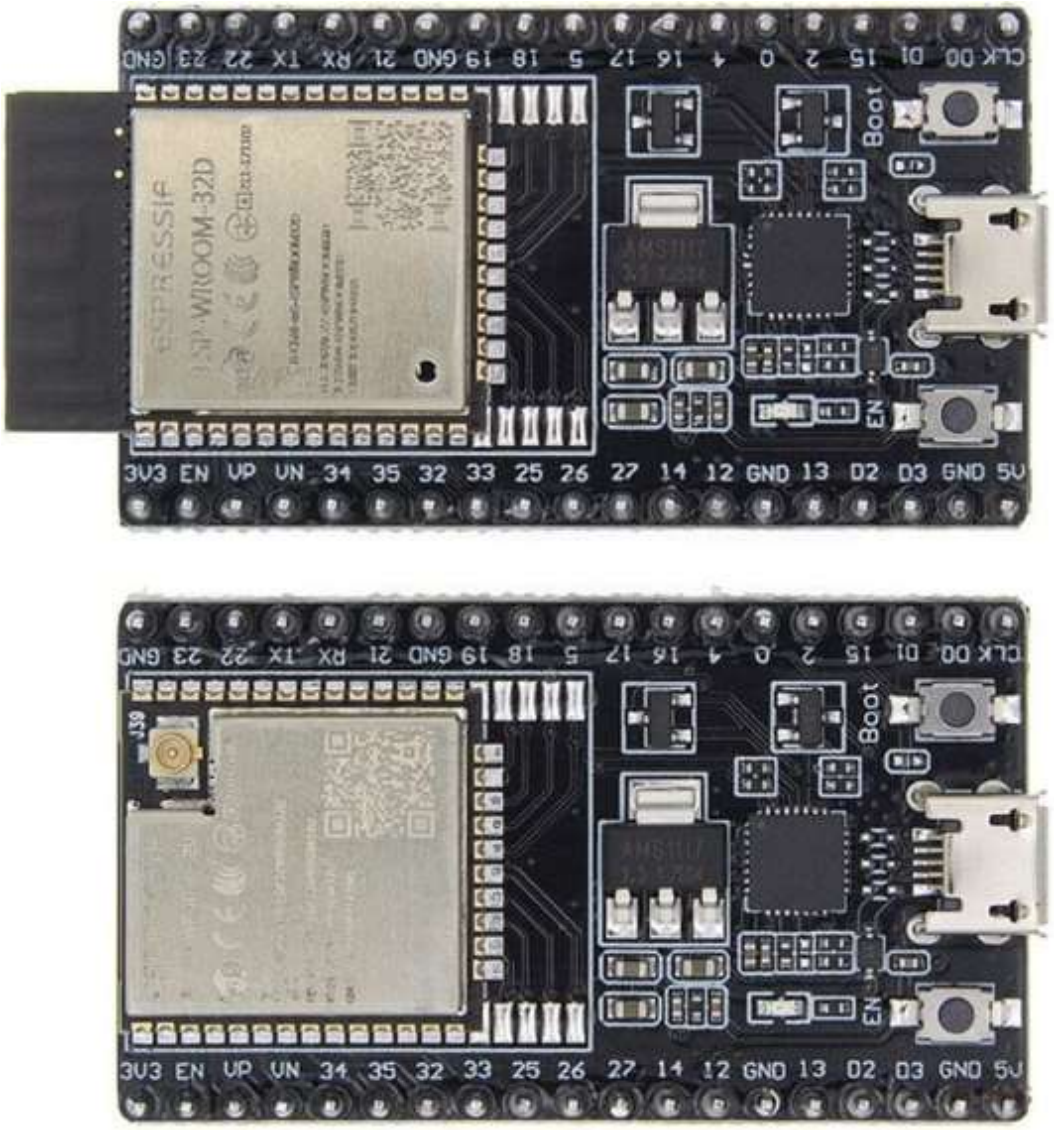
Las tarjetas ESP32 y ESP8266 debido a su tamaño y flexibilidad para trabajar se encuentran presentes en una gran cantidad de módulos que podemos encontrar en el mercado, cada uno con características propias que lo hacen más apto para ciertos trabajos o entorno en el que se desenvolverán, a continuación, veremos un representante de cada una de las tarjetas

ESP32

En la siguiente imagen veremos los componentes principales de la tarjeta de desarrollo ESP32-V4

Figura 5

Tarjeta de desarrollo ESP32-V4



Nota. Tomada de *ESP32 DevKitC V4 ESP32 WROOM 32D y 32U* [Imagen], Unit Electronics, 2022 (https://uelectronics.com/producto/esp32-devkitc-v4-esp32-wroom-32d-32u/). CC BY 2.0

Tabla 2

Componentes Principales Tarjeta de Desarrollo ESP32 V4

Componentes Principales Tarjeta de Desarrollo ESP32 V4	
Componente	Descripción
ESP32-WROOM-32	Modulo principal ESP32
Botón Enable	Botón de reseteo
Botón Boot	Botón de descarga, tenerlo presionado junto a enable inicia el modo de descarga de firmware a través del puerto serial
Puente USB a UART	Chip que permite la transferencia de hasta 3 Mbps
Puerto Micro USB	Interfaz USB. Fuente de alimentación además de interfaz de comunicación entre la computadora y el módulo ESP32
Led Encendido	Se enciende cuando se conecta la tarjeta a una fuente de 5V
I/O	Pines de entrada y salida

Nota: Se detallan los componentes principales de la tarjeta ESP 32.

Esta tabla ha sido adaptada de “ESP32-DevKitC V4 Getting Started Guide”, por ESPRESSIF, 2022 (https://docs.espressif.com/projects/esp-idf/en/latest/esp32/hw-reference/esp32/get-started-devkitc.html).

Esta tarjeta cuenta con tres fuentes opciones para poder trabajar, a través del puerto micro USB y los pines a tierra de 5 y 3.3V.

Antes de seguir con la exposición de las características de esta tarjeta mostraremos el significado de algunos términos que tal vez generen confusión a los novatos de la electrónica

- **I/O:** Entrada y Salida por sus siglas en ingles Input/Output
- **GPIO:** Entrada y Salida de propósito general
- **GND:** Tierra
- **PWM:** Modulación de Ancho de Pulso

En la figura 6 podemos observar la disposición de los pines y sus distintas funciones, así como en la tabla 3 los pines del lado izquierdo y la tabla 4 los del lado derecho

Tabla 3

Disposición de Pines Izquierdos de la Tarjeta de Desarrollo ESP32 V4

Disposición de Pines Izquierdos de la Tarjeta de Desarrollo ESP32 V4			
Numero	Nombre	Tipo	Función
1	3,3V	Poder	Alimentación de 3.3V
2	Enable	I	CHIP_PU, Reseteo
3	IO36	I	GPIO36, ADC1_CH0, S_VP
4	IO39	I	GPIO39, ADC1_CH3, S_VN
5	IO34	I	GPIO34, ADC1_CH6, VDET_1
6	IO35	I/O	GPIO35, ADC1_CH7, VDET_2

7	IO32	I/O	GPIO32, ADC1_CH4, TOUCH_CH9, XTAL_32K_P
8	IO33	I/O	GPIO33, ADC1_CH5, TOUCH_CH8, XTAL_32K_N
9	IO25	I/O	GPIO25, ADC1_CH8, DAC_1
10	IO26	I/O	GPIO26, ADC2_CH9, DAC_2
11	IO27	I/O	GPIO27, ADC2_CH7, TOUCH_CH7
12	IO14	I/O	GPIO14, ADC2_CH6, TOUCH_CH6, MTMS
13	IO12	I/O	GPIO12, ADC2_CH5, TOUCH_CH5, MTDI
14	GND	Tierra	Tierra
15	IO13	I/O	GPIO13, ADC2_CH4, TOUCH_CH4, MTCK
16	IO9	I/O	GPIO9, D2
17	IO10	I/O	GPIO10, D3
18	IO11	I/O	GPIO11, CMD
19	5V	Poder	Alimentación de 5V

Nota: Se detalla la disposición de los Pines Izquierdos de la Tarjeta de Desarrollo ESP32 V4.

Esta tabla ha sido adaptada de “ESP32-DevKitC V4 Getting Started Guide”, por ESPRESSIF, 2022 (https://docs.espressif.com/projects/esp-idf/en/latest/esp32/hw-reference/esp32/get-started-devkitc.html).

Tabla 4

Disposición de Pines Derechos de la Tarjeta de Desarrollo ESP32 V4

Disposición de Pines Derechos de la Tarjeta de Desarrollo ESP32 V4			
Número	Nombre	Tipo	Función
1	GND	Tierra	Tierra
2	IO23	I/O	GPIO23
3	IO22	I/O	GPIO22
4	IO1	I/O	GPIO1, U0TXD
5	IO3	I/O	GPIO3, U0RXD
6	IO31	I/O	GPIO21
7	GND	Tierra	Tierra
8	IO19	I/O	GPIO19
9	IO18	I/O	GPIO18
10	IO5	I/O	GPIO5
11	IO17	I/O	GPIO17
12	IO16	I/O	GPIO16
13	IO4	I/O	GPIO4 , ADC2_CH0, TOUCH_CH0
14	IO0	I/O	GPIO0, ADC2_CH1, TOUCH_CH1, Boot
15	IO2	I/O	GPIO2, ADC2_CH2, TOUCH_CH2

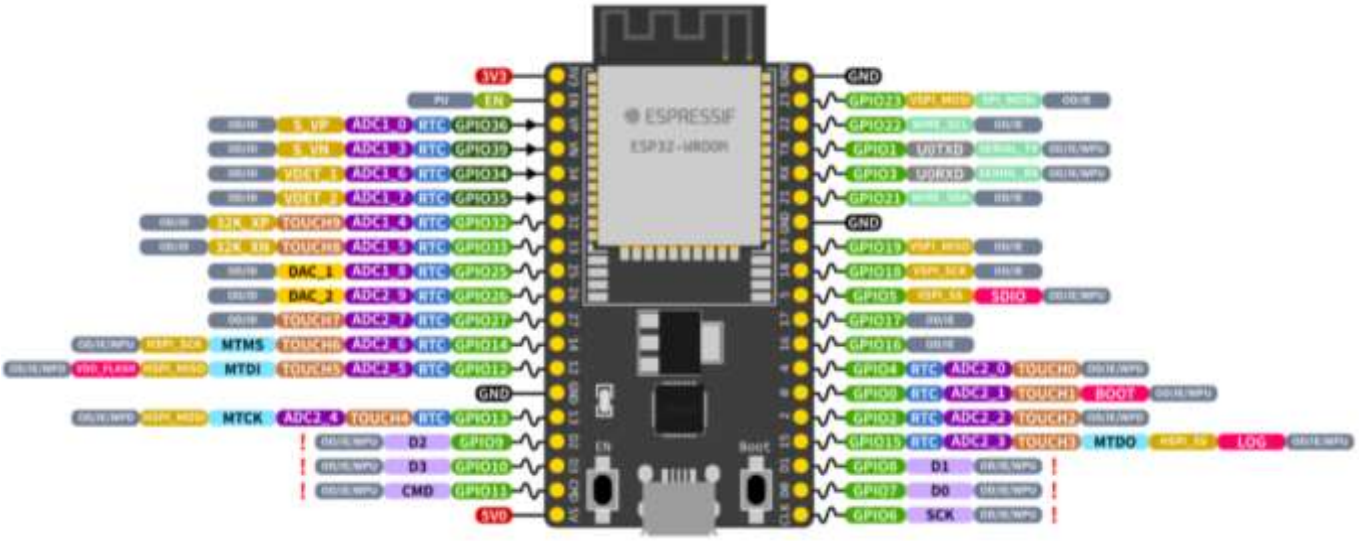
16	IO15	I/O	GPIO15, ADC2_CH3, TOUCH_CH3, MTDO
17	IO8	I/O	GPIO8, D1
18	IO7	I/O	GPIO7, D0
19	IO6	I/O	GPIO6, SCK

Nota: Se detalla la disposición de los Pines Derechos de la Tarjeta de Desarrollo ESP32 V4.

Esta tabla ha sido adaptada de "ESP32-DevKitC V4 Getting Started Guide", por ESPRESSIF, 2022 (https://docs.espressif.com/projects/esp-idf/en/latest/esp32/hw-reference/esp32/get-started-devkitc.html).

Figura 6

Disposiciones de Pines tarjeta de desarrollo ESP32-V4



Nota. Adaptada de *ESP32-DevKitC Pin Layout* [Imagen], Espressif, 2022 (https://docs.espressif.com/projects/esp-idf/en/latest/esp32/hw-reference/esp32/get-started-devkitc.html) CC BY 2.0

ESP8266

Este chip se encuentra presente en varias placas de desarrollo, pero nos enfocaremos en la última versión lanzada al mercado de la placa de desarrollo NODEMCU V3, ya que es uno de los módulos comerciales más populares

Figura 7

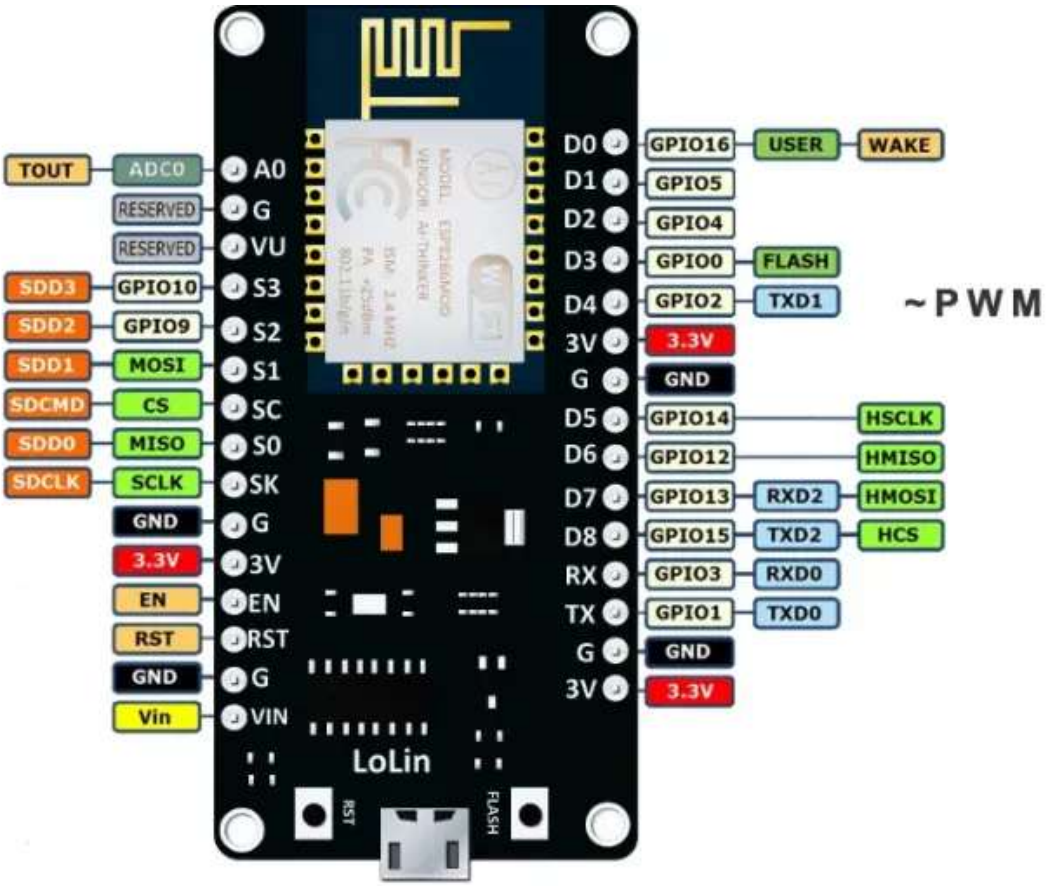
Tarjeta de Desarrollo NodeMCU V3

Nota. Tomada de *NodeMCU V3* [Imagen], Sigma Electrónica, 2022 (https://www.sigmaelectronica.net/producto/nodemcu-v3/). CC BY 2.0

Comparte muchas similitudes con la ESP32 que vimos anteriormente por lo cual nos centraremos en los aspectos más generales de esta tarjeta de desarrollo.

Figura 8

Disposición de pines NODEMCU V3



Nota. Tomada de *NodeMCU V3 Pinout* [Imagen], TM Microcontrollers, 2022 (https://www.teachmemicro.com/nodemcu-pinout/). CC BY 2.0

Tabla 5

Disposición de Pines de la Tarjeta de Desarrollo NODEMCU V3

Disposición de Pines de la Tarjeta de Desarrollo NODEMCU V3		
Pin	GPIO	Función
D0	GPIO16	PWM,Resistencia Pull Down
D1	GPIO5	SCL(I2C)
D2	GPIO4	SDA(I2C)

D3	GPIO0	Conectado a botón Flash
D4	GPIO2	Led integrado, TX1
D5	GPIO14	SLCK(SPI)
D6	GPIO12	MISO(SPI)
D7	GPIO13	MOSI(SPI)
D8	GPIO15	CS(SPI)
RX	GPIO3	No usable si se usa UART
TX	GPIO1	Debugging
A0	ADC0	Entrada análoga

Nota: Se detalla la disposición de Pines de la Tarjeta de Desarrollo NODEMCU V3.

Esta tabla ha sido adaptada de “NODEMCU, LA POPULAR PLACA DE DESARROLLO CON ESP8266”, por Luis Lamas, 2022 (https://www.luisllamas.es/esp8266-nodemcu).

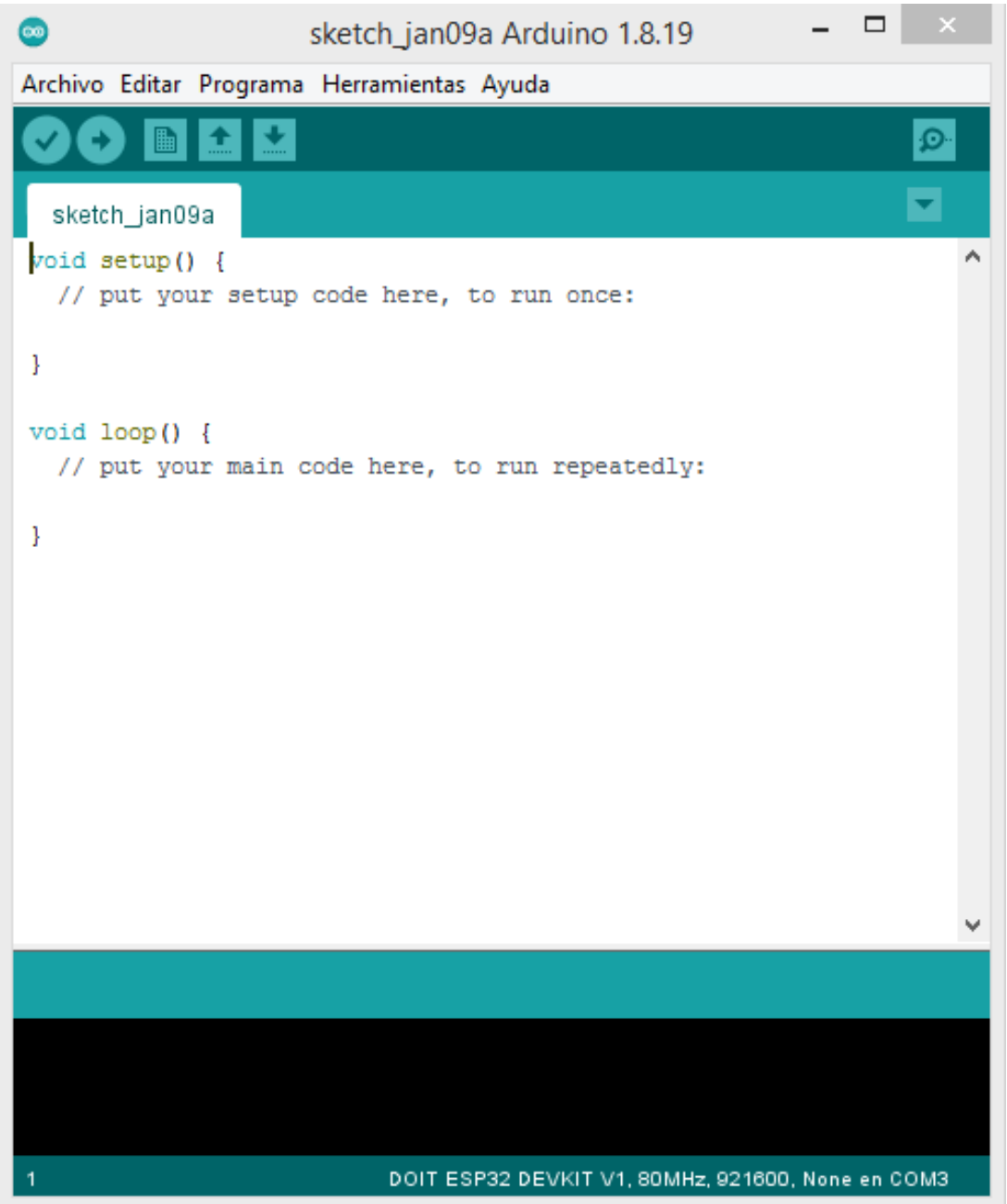
Entornos de Desarrollo

Para poder trabajar con las tarjetas ESP32 y ESP8266 contamos con algunos ambientes de desarrollo en los que podemos escribir nuestro código siendo los más populares Arduino IDE y ESP-IDF, aunque esto no lo hace los único en su clase, también contamos con algunos programas que nos ayudaran a la hora de escribir código, ahora veremos una breve reseña de ellos.

Arduino IDE: Es un software de licencia libre que permite escribir programas y subirlos directamente a tu tarjeta, aunque inicialmente solo cuenta con las librerías necesarias para trabajar con Arduino se puede agregar los archivos necesarios para que reconozca la tarjeta ESP

Figura 9

Ventana principal de Arduino IDE

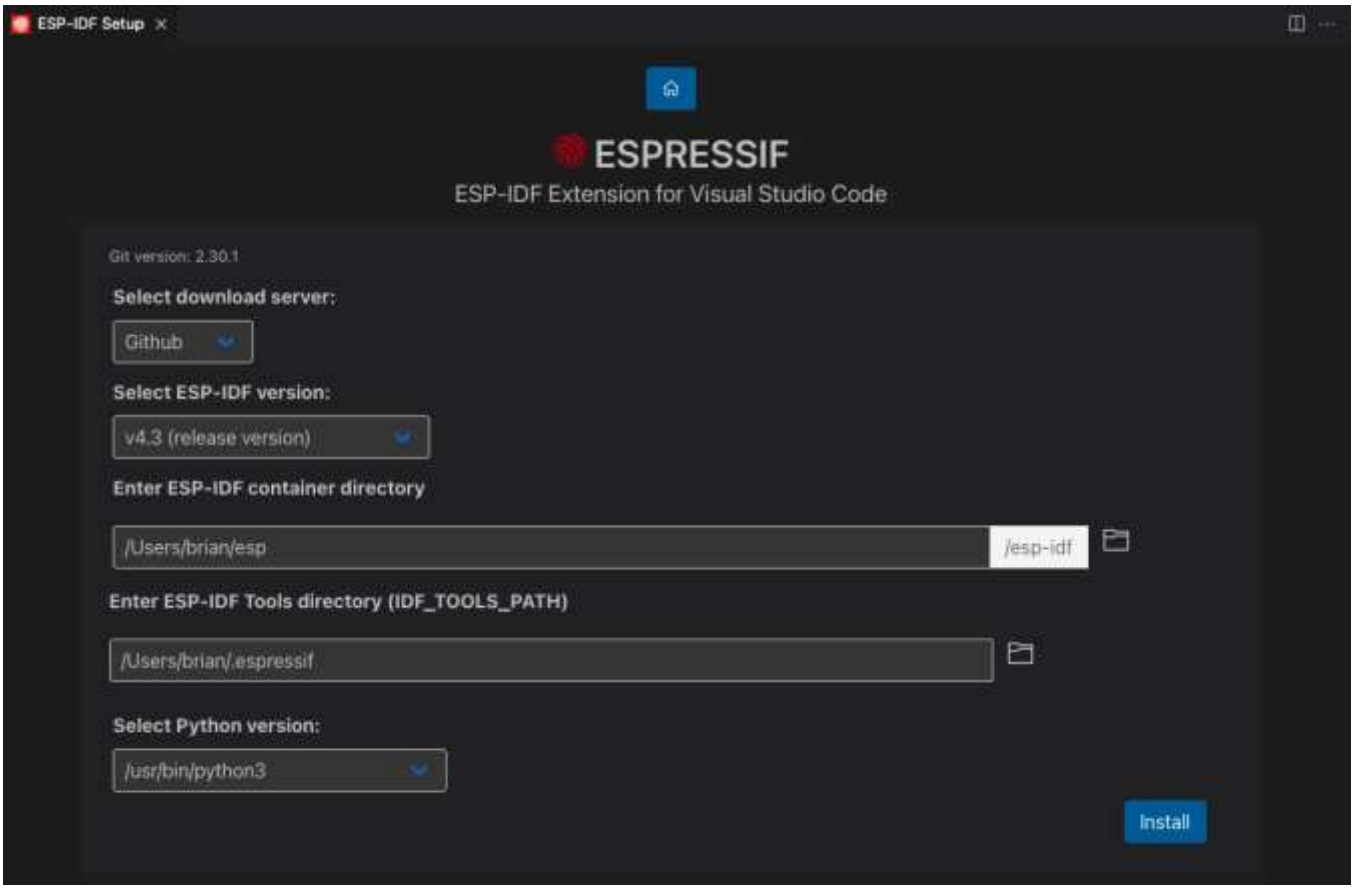


ESP-IDF: Desarrollado por Espressif es un entorno de desarrollo para los hardware que hagan uso de los chips ESP32 y ESP8266, su nombre son las siglas para Espressif IoT Development Framework, a diferencia de Arduino IDE que necesita agregarle la librería para trabajar con ESP esto

lo hace de forma nativa, lo cual ayuda a tener un mejor control de los procesos que se desee implementar.

Figura 3

ESP-IDF en Visual Studio Code

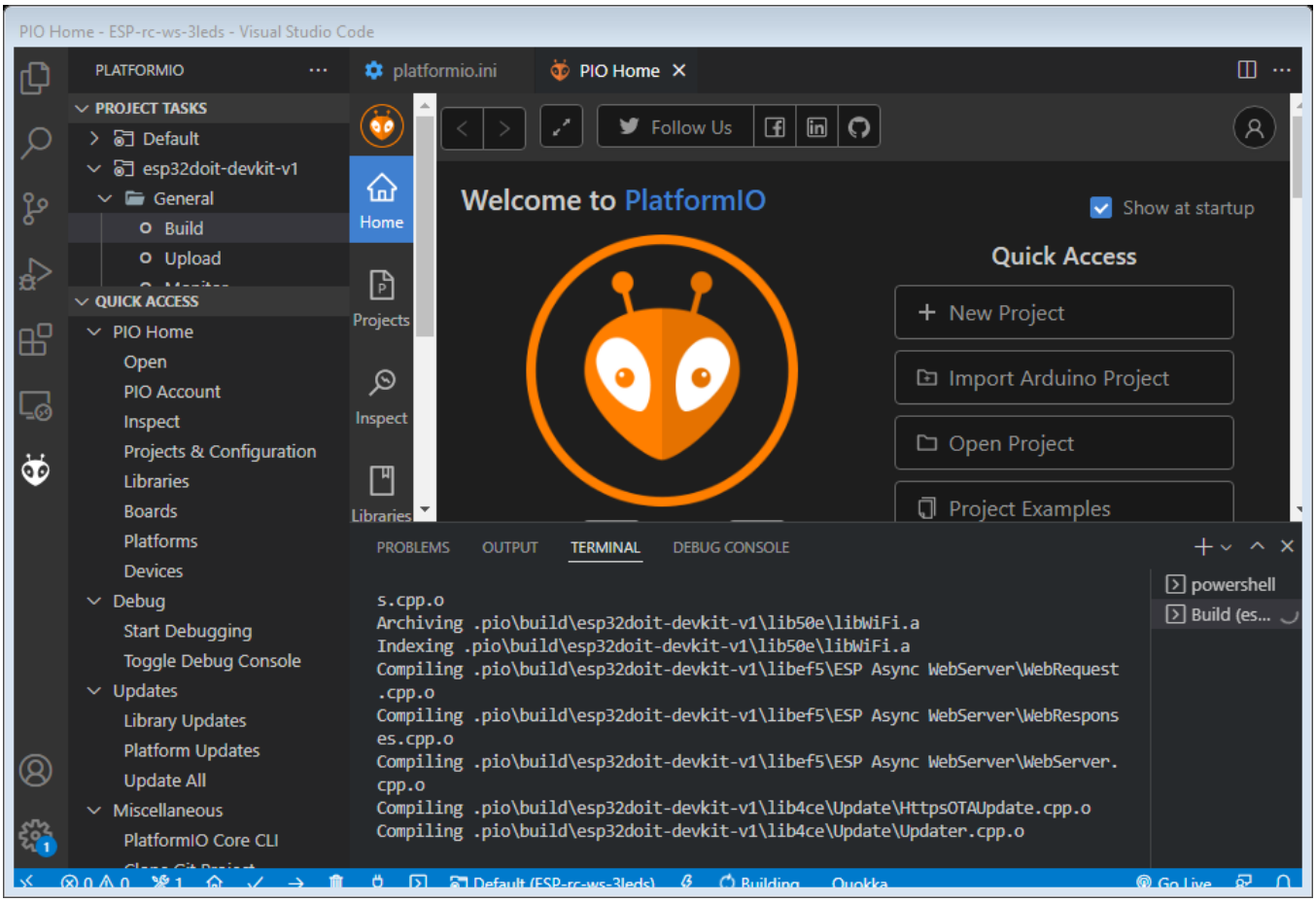


Nota. Tomada de *VSCode-ESP-IDF-Extension* [Imagen], GitHub Espressif, 2022 (https://github.com/espressif/vscode-esp-idf-extension/blob/master/docs/tutorial/install.md). CC BY 2.0

PlatformIO: Sus desarrolladores la describen como una herramienta multiplataforma de arquitectura cruzada para sistemas embebidos, ideal para todo aquel que desee escribir código para productos embebidos, provee a los desarrolladores con un entorno de desarrollo capaz de soportar distintos kits de desarrollo de software (SDKs) o frameworks además de incluir con herramientas sofisticadas para debugging y testeo de unidades.

Figura 11

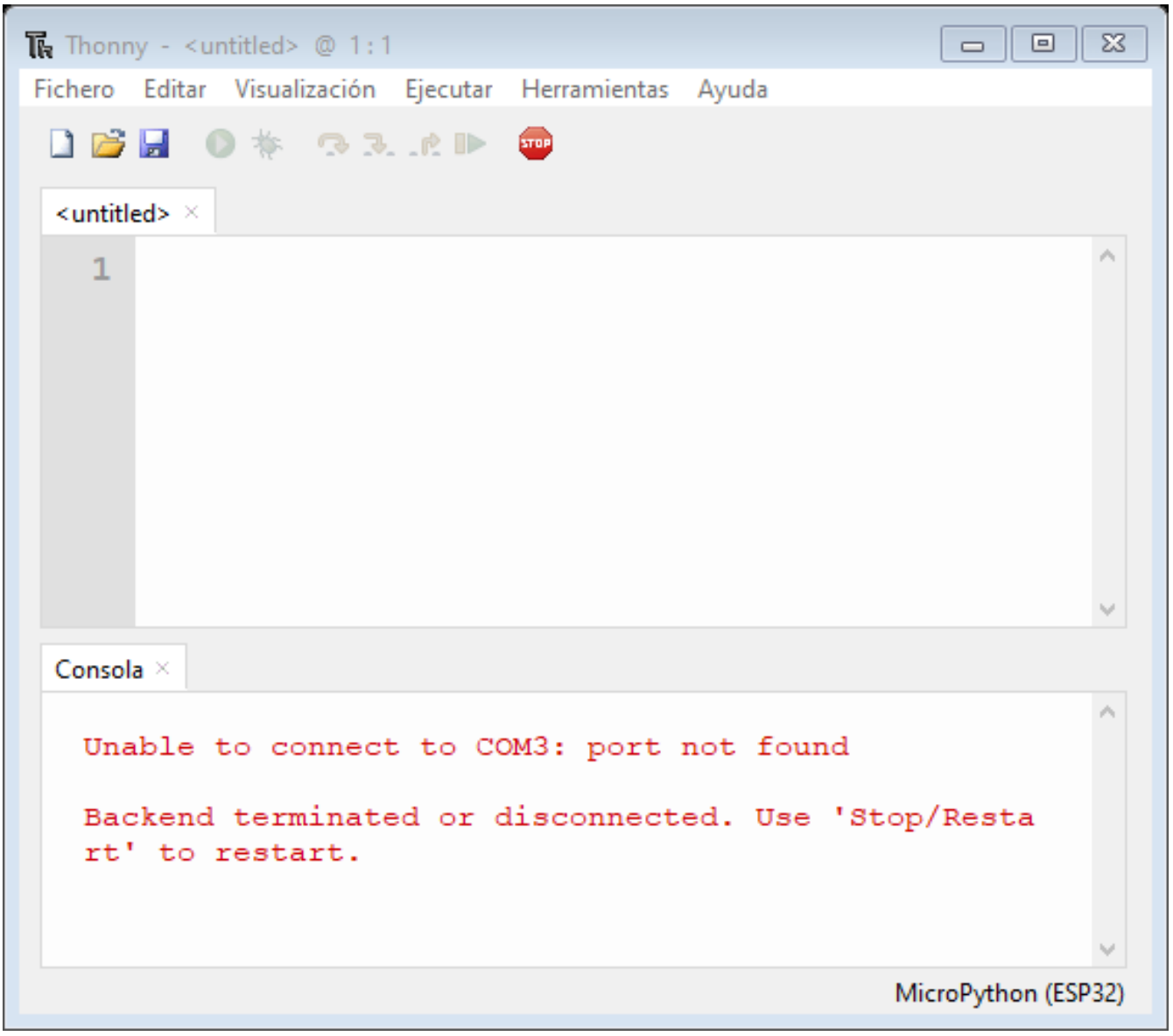
PlataformIO en Visual Studio Code



Thonny: Es un Entorno de Desarrollo Integrado (IDE por sus siglas en inglés) de Micropython que nos facilitara el proceso de escribir código para las tarjetas ESP32 y ESP8266, es compatible con Windows, MacOS y Linux, de fabrica viene instalado con Raspbian OS para Raspberry Pi, todo esto viene incluido en su instalador el cual pesa menos de 20 Mb.

Figura 42

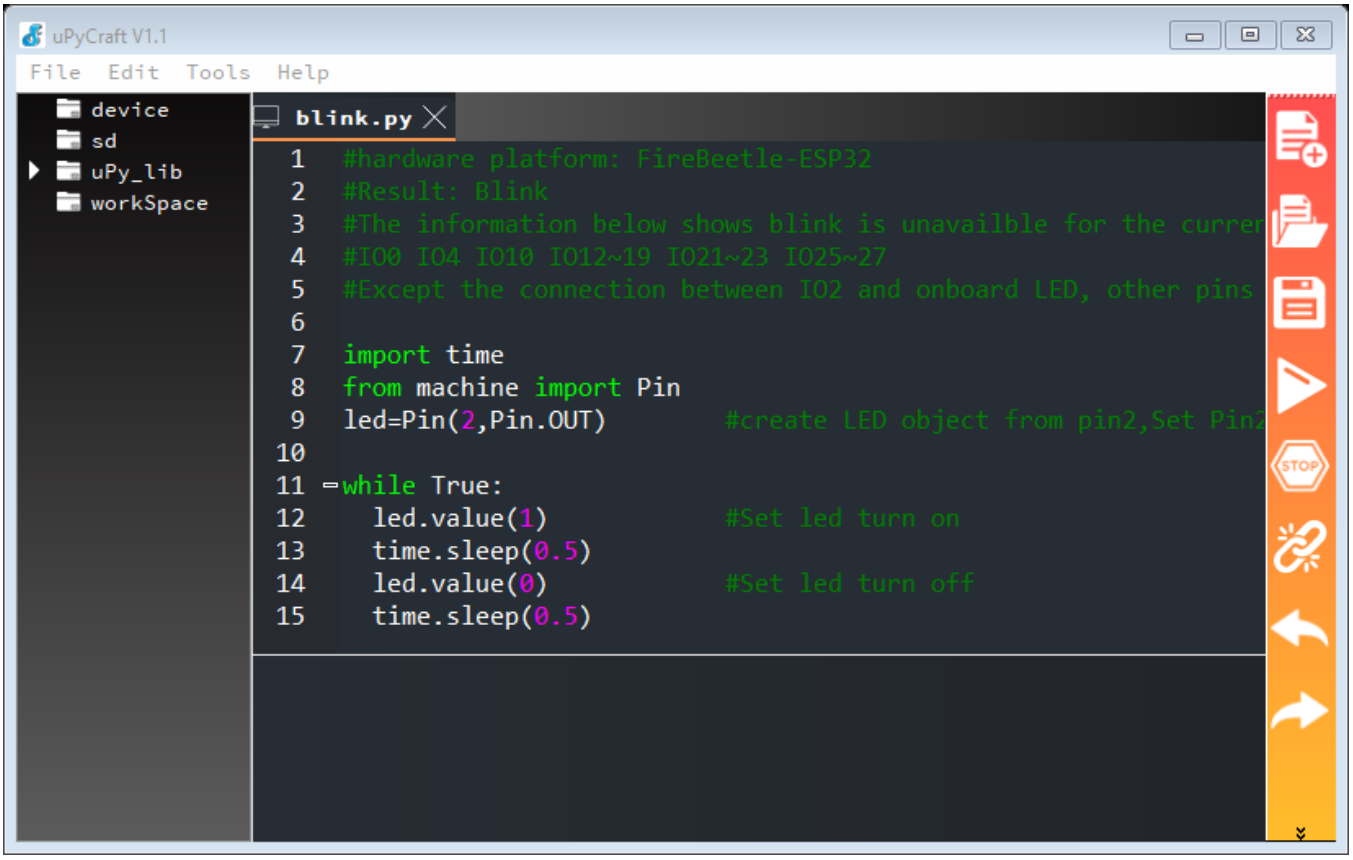
Ventana Principal Thonny



uPyCraft: Al igual que Thonny, este es una IDE que trabaja con mycropython, la principal ventaja que tiene con respecto a su competidor es que cuenta con una funcionalidad integrada que permite subir el código que escribimos directamente a la tarjeta ESP32 o ESP8266.

Figura 53

Ventana Principal uPyCraft



ESP32 con Arduino IDE

Instalando Arduino IDE

Para instalar Arduino IDE seguiremos los siguientes pasos

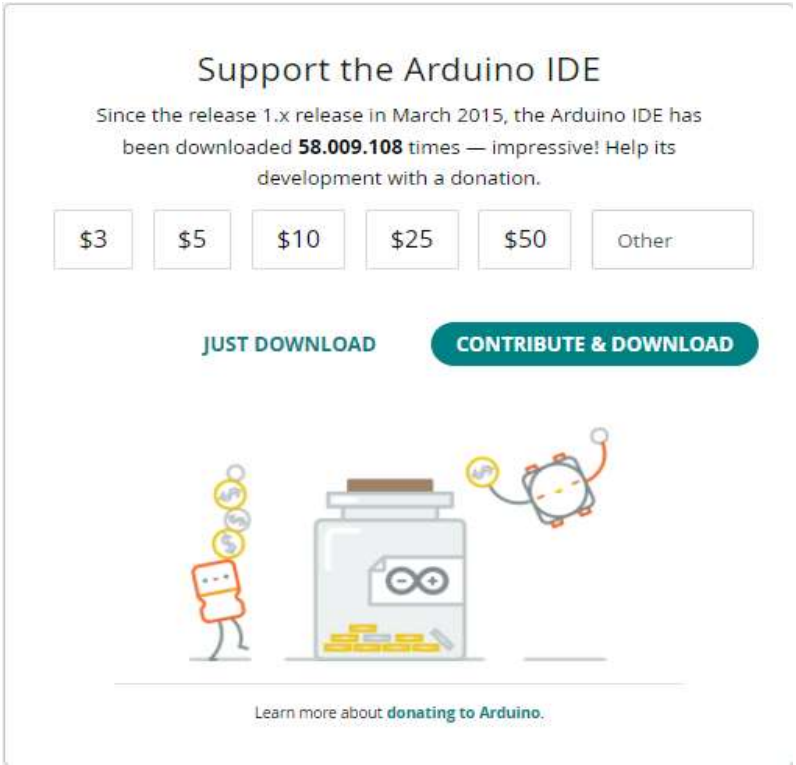
- Ingresamos a la página oficial de Arduino: https://www.arduino.cc/en/software
- Seleccionamos la última versión estable, al momento de escribir este documento es la 1.8.19
- Escogemos la versión a descargar según nuestro sistema operativo, la siguiente ventana nos dará la opción de donar o simplemente descargar, iremos por la segundo dando click en just download.

Figura 64

Sitio oficial de Arduino

Figura 75

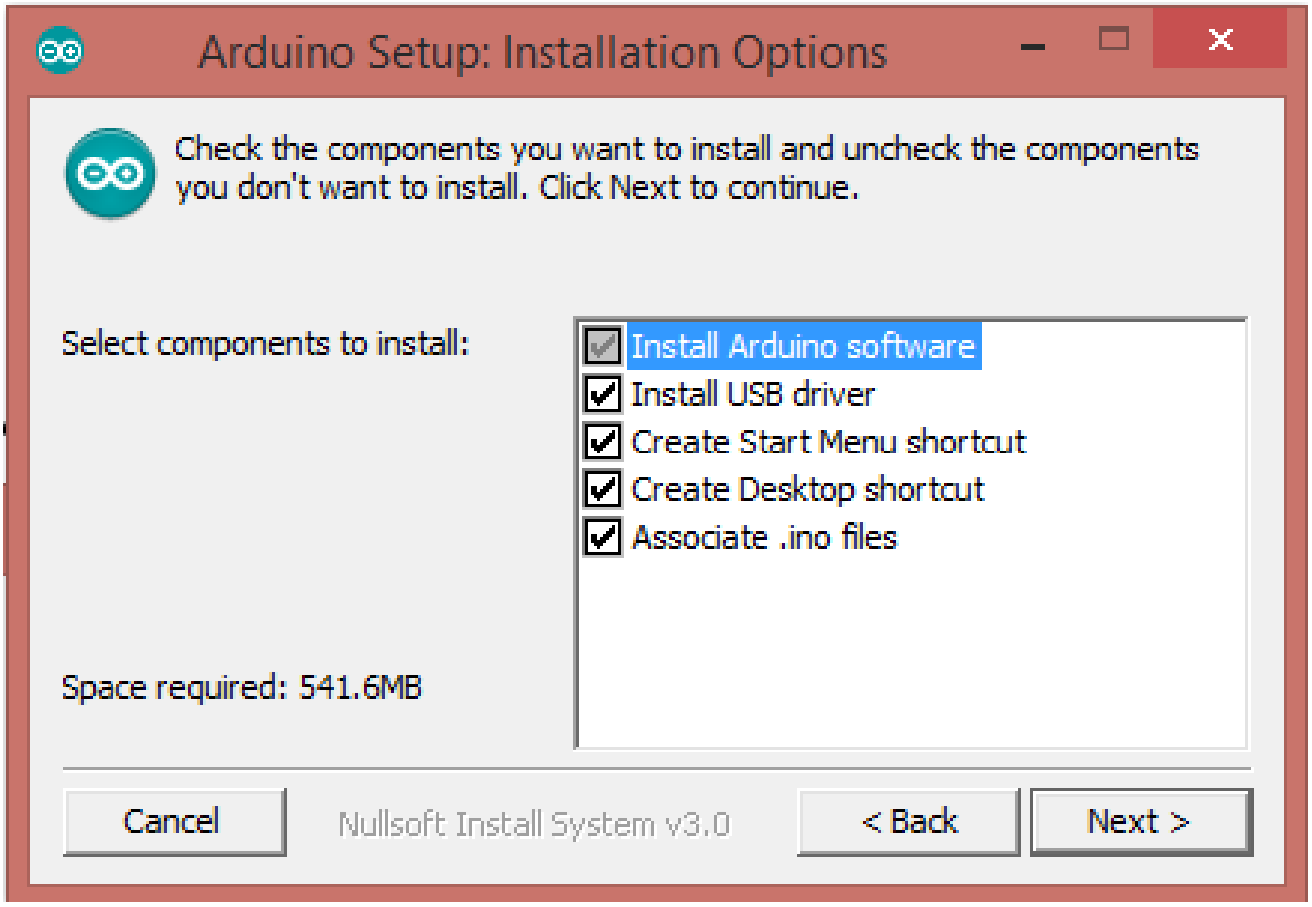
Ventana de descarga de Arduino IDE



- Ejecutamos el archivo .exe que descargamos
- Procedemos con la instalación aceptando los términos y condiciones
- Seleccionamos los componentes que deseemos instalar, como recomendación seleccionar todos.

Figura 16

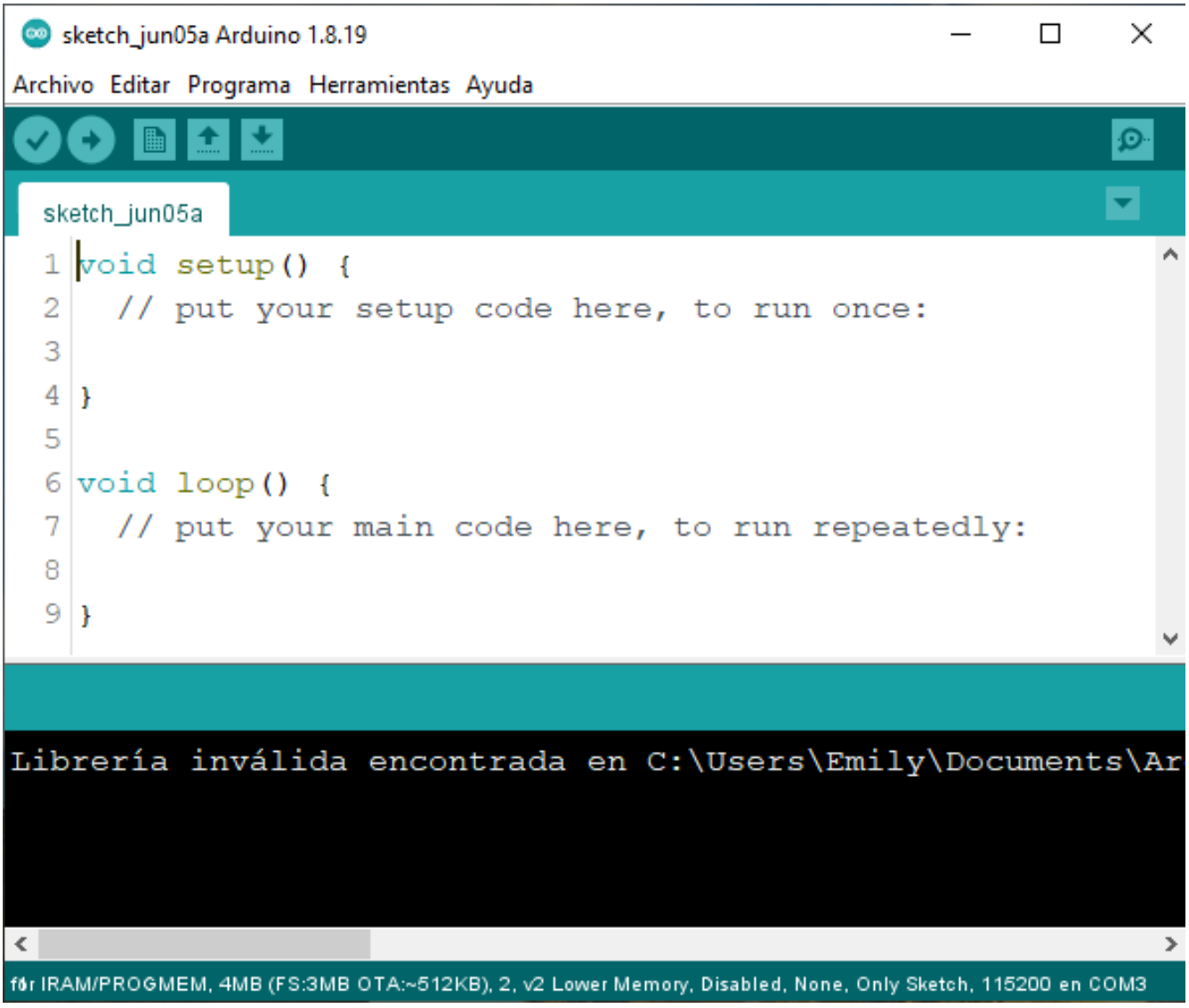
Componentes a instalar



- Escogemos la carpeta en que deseamos instalar Arduino IDE
- Esperamos a que se instale el programa
- Finalización de la Instalación
- Ejecutamos el programa
- Ventana principal de Arduino IDE, aquí ya podremos empezar a escribir nuestro código y realizar configuraciones

Figura 17

Ventana principal de Arduino IDE



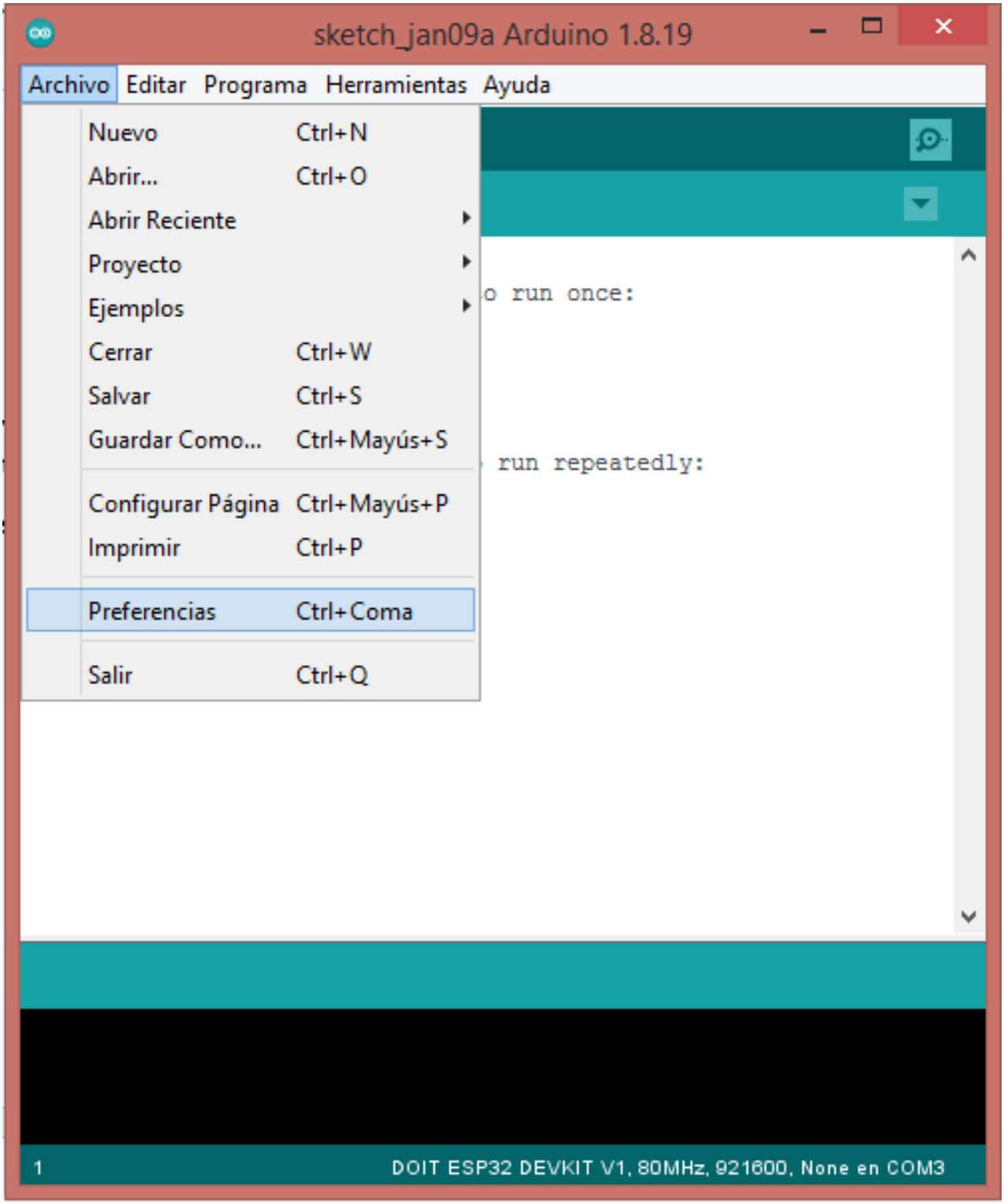
Una vez terminada la instalación ya podremos pasar a instalar los componentes necesarios para trabajar con la tarjeta ESP32.

Instalando la tarjeta ESP32

- Abrir la pestaña de preferencias, Archivo > Preferencias

Figura 18

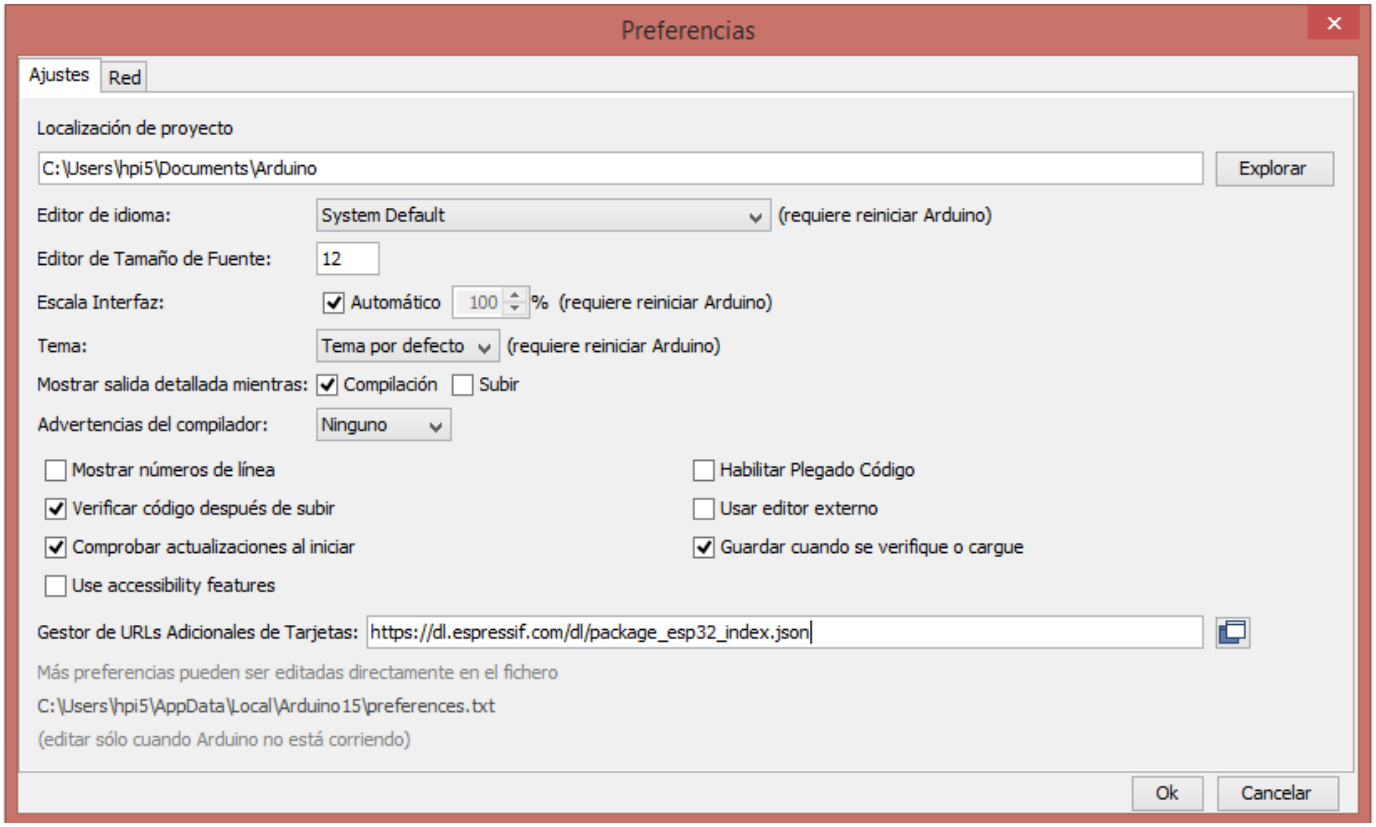
Pestaña archivo de Arduino IDE



- Copiamos https://dl.espressif.com/dl/package_esp32_index.json en el apartado de Gestor de URLs adicionales de Tarjetas, en caso de querer instalar otra tarjeta simplemente separar las direcciones con comas.

Figura 19

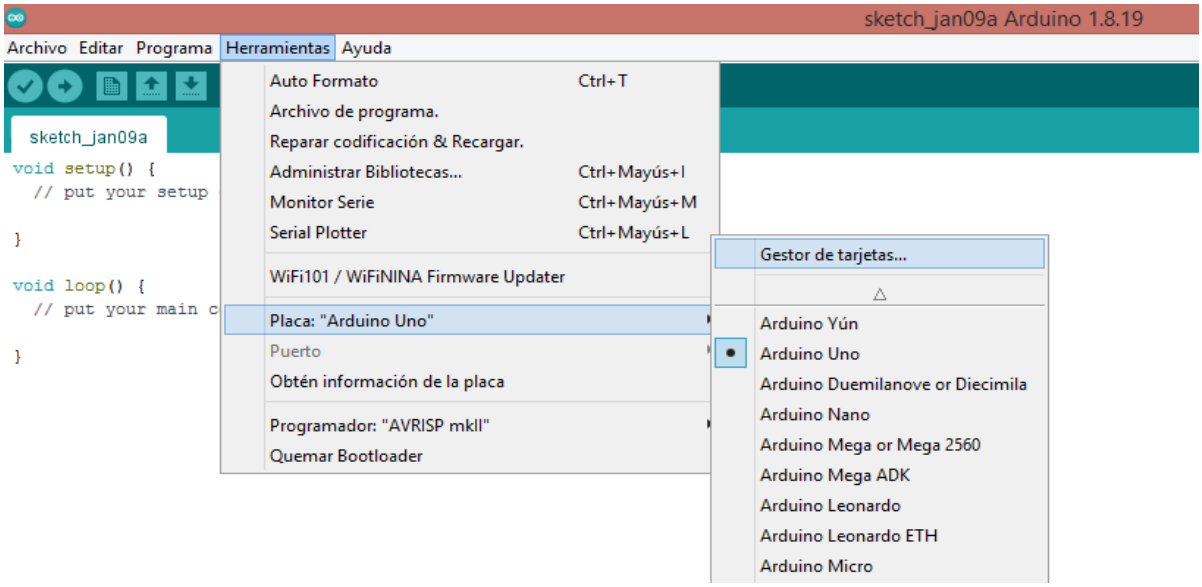
Ventana preferencias



- Dirigirse a Herramientas > Placa > Gestor de tarjetas

Figura 20

Selección de tarjeta en la pestaña herramientas



- Buscar ESP32 y presionar instalar

Figura 21

Búsqueda de la tarjeta ESP32 en la ventana Gestor de Tarjetas

Instalación de la tarjeta ESP32

Figura 22

Descarga del firmware para la tarjeta ESP32

- Una vez instalado reinicia el Arduino IDE

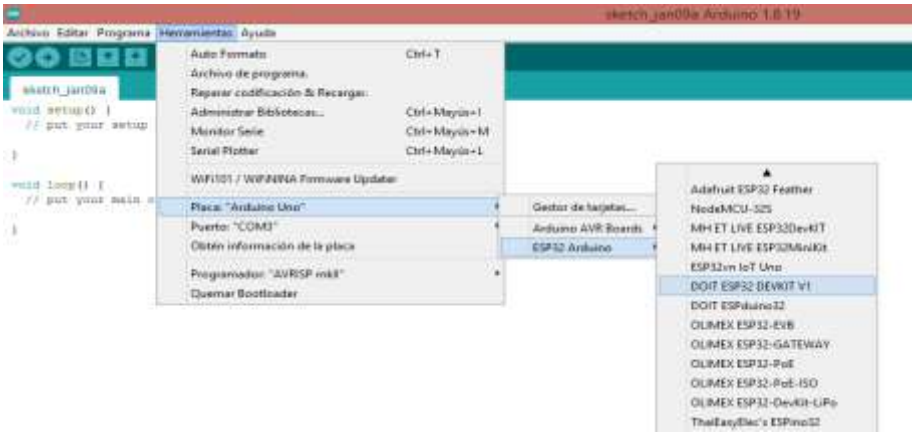
Prueba de la instalación

Para la realización de esta prueba necesitaras conectar tu tarjeta ESP32 a tu computadora, una vez echo eso sigue estos pasos

- Dirigirse a Herramientas > Placas y seleccionamos DOIT ESP32 DEVKIT V1

Figura 23

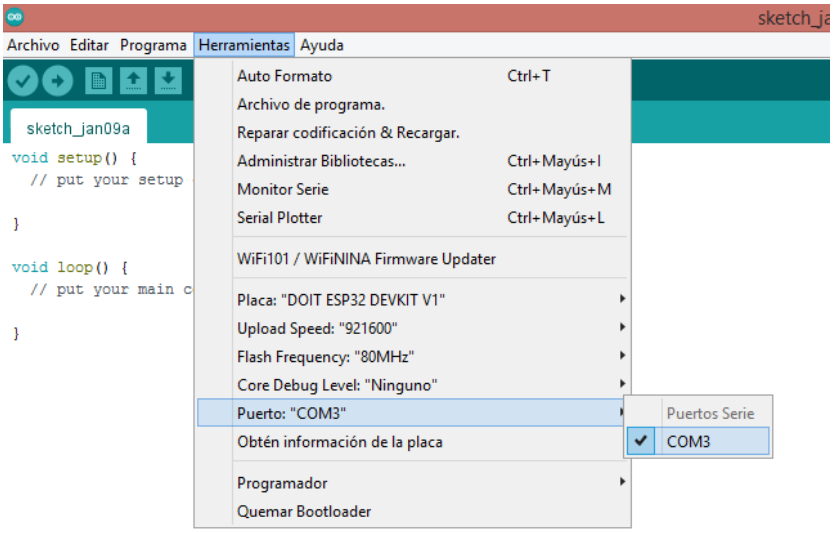
Selección de la tarjeta ESP32



- Seleccionar el puerto (en caso de no verlo, dirigirse al apartado de Errores Comunes)

Figura 24

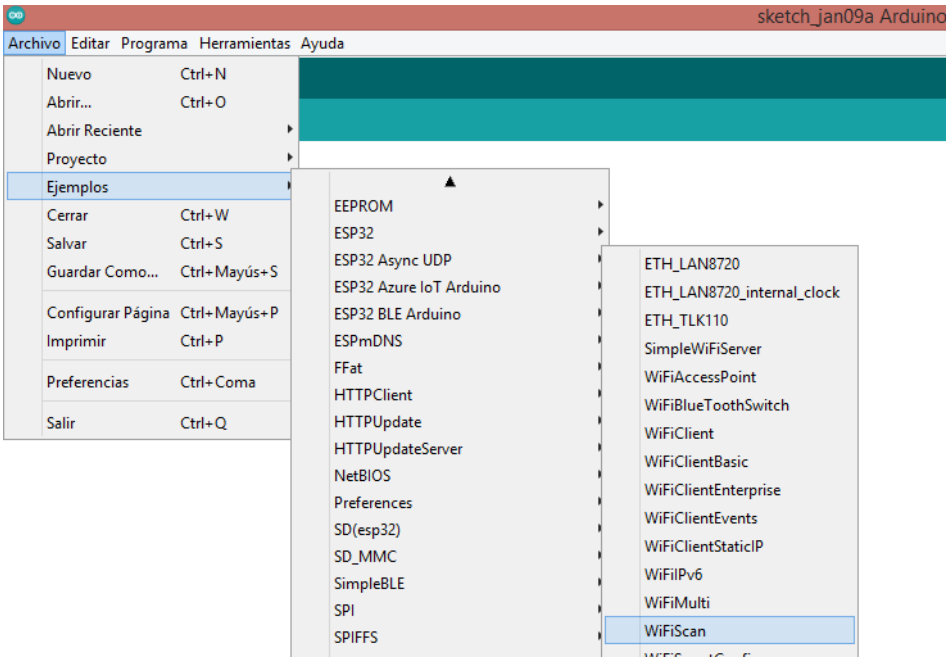
Selección del puerto COM



- Abrir la carpeta de ejemplos en Archivo > Ejemplos > Wifi (ESP32) > Wifi Scan

Figura 25

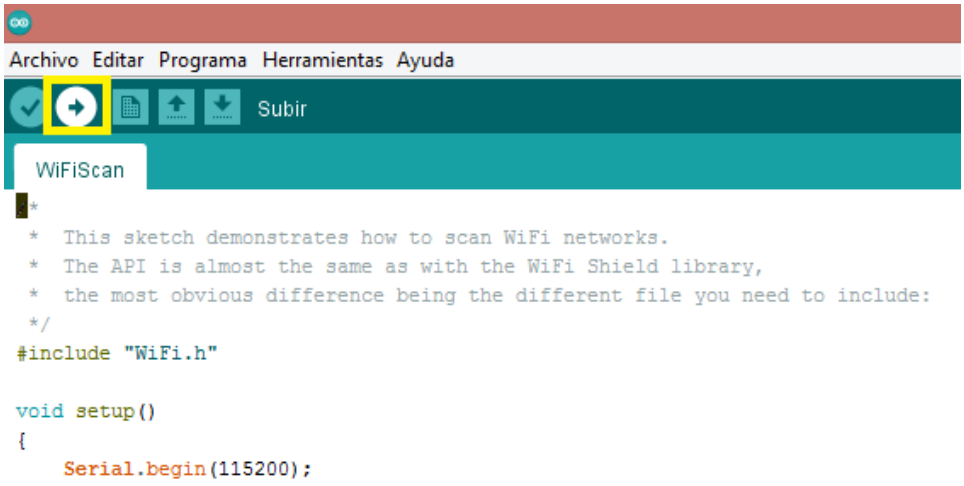
Selección de Ejemplo



- Presionar el botón de subir y seguido presione el botón BOOT en su tarjeta (si no sabe dónde se encuentra el botón diríjase al apartado de Errores Comunes)

Figura 26

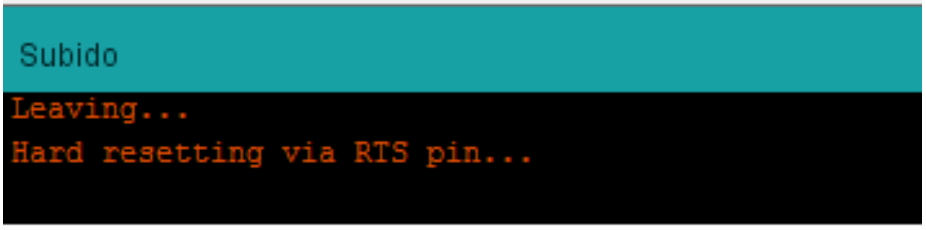
Subida del programa a la tarjeta ESP32



- Espere que el programa se sube a la tarjeta
- Si todo sale bien, se observará el mensaje de subido

Figura 27

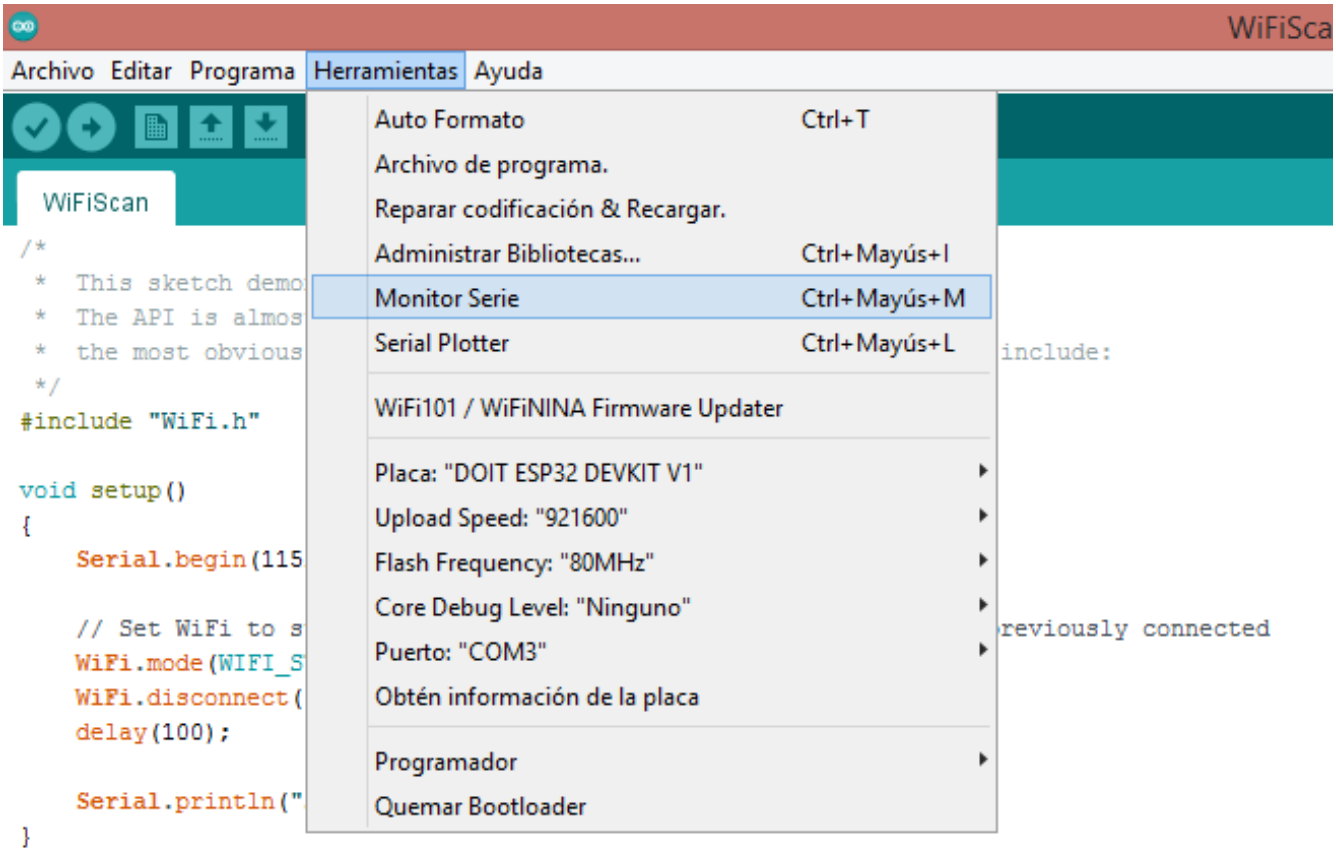
Carga exitosa del programa



- Abrimos el monitor serial de Arduino con los baudios en 115200

Figura 28

Monitor Serial en la pestaña herramientas



- Al presionar Enable en nuestra tarjeta nos mostrará información en el monitor

Figura 29

Mensaje en el Monitor Serial

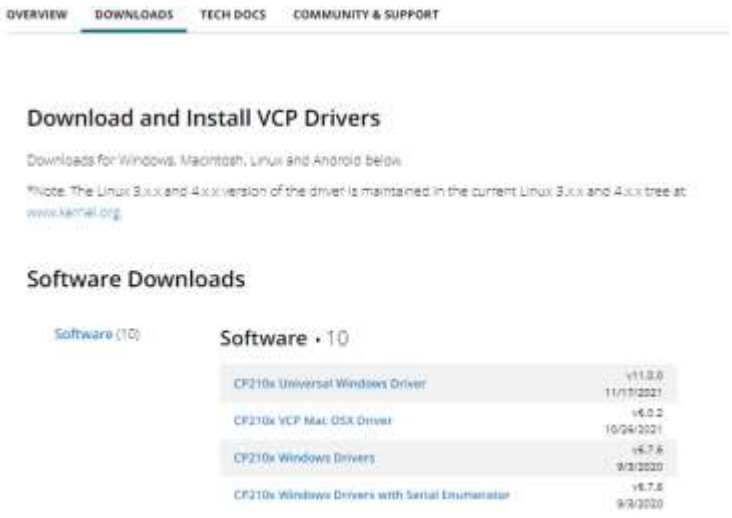
Problemas Comunes

No detecta el puerto:

Si no le aparece el puerto COM en el Arduino IDE necesitara instalar los drivers USB de la tarjeta lo puede descargar desde: https://www.silabs.com/products/development-tools/software/usb-to-uart-bridge-vcp-drivers

Figura 30

Descarga de drivers

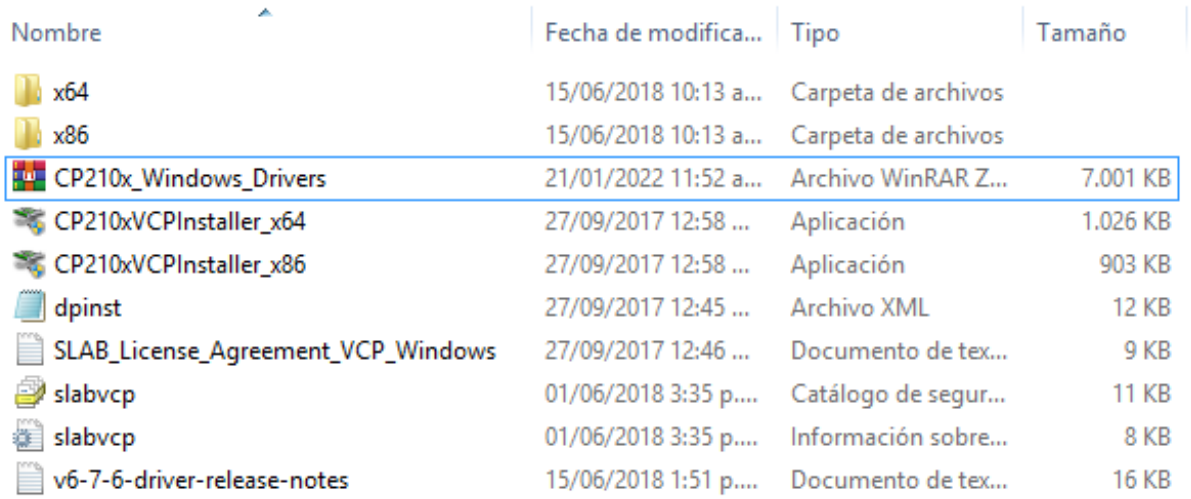


Nota: Universal Windows Driver es para Windows 10 o superiores, si queremos usar en versiones inferiores tales como Windows 8,8.1 o 7 debemos descargar Windows Drivers

- Extraemos el archivo que descargamos

Figura 31

Archivos extraídos

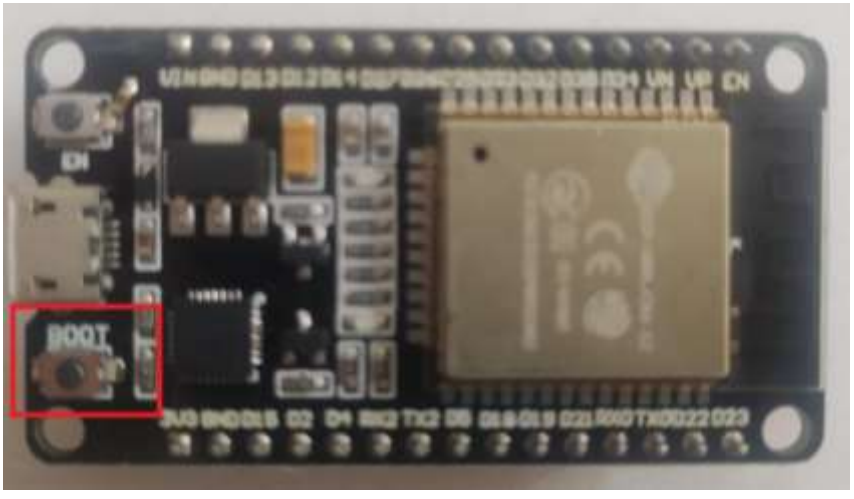


Seleccionamos la versión de nuestro S.O ya sea 32 bits (x86) o 64 (x64), ejecutamos el instalador y seguimos los pasos que nos indica.

- **"Failed to connect to ESP32: Timed out… Connecting…":** Si le aparece este mensaje de error, hay dos formas de solucionarlo
- Se olvidó de presionar el botón BOOT en su tarjeta ESP32

Figura 32

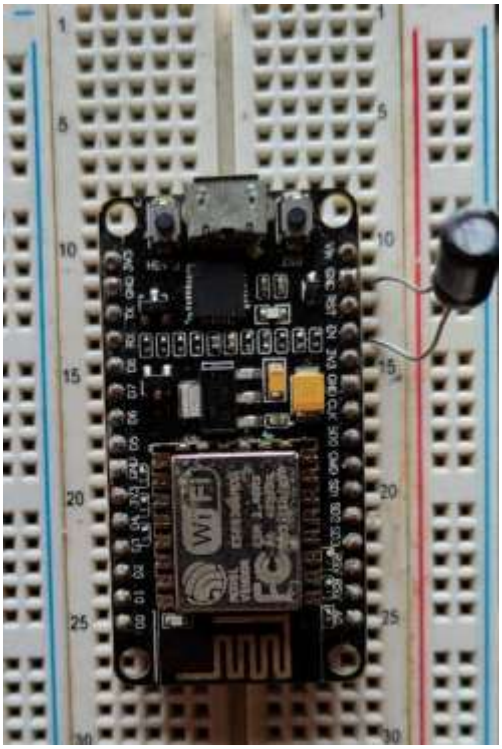
Botón BOOT en la tarjeta ESP32



- Conectar un capacitor de 10μF con el positivo en el pin de Enable y el negativo en GND

Figura 33

Capacitor conectado a la tarjeta ESP32



WEB SERVER

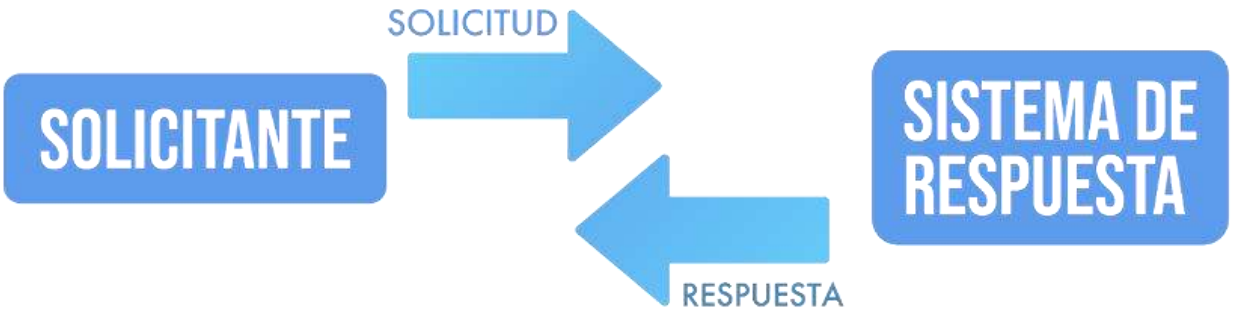
El ESP32 es un dispositivo que cuenta con dos núcleos donde se puede intercambiar información de manera simultánea en la nube o servidor web. Este módulo está completo de muchas funciones, como una de ellas es la combinación wifi y Bluetooth.

Usando estas funciones del ESP32 le mostraremos cómo se puede utilizar este módulo al conjunto de algunos sensores.

Solicitud-Respuesta

Figura 34

Esquema solicitud - respuesta

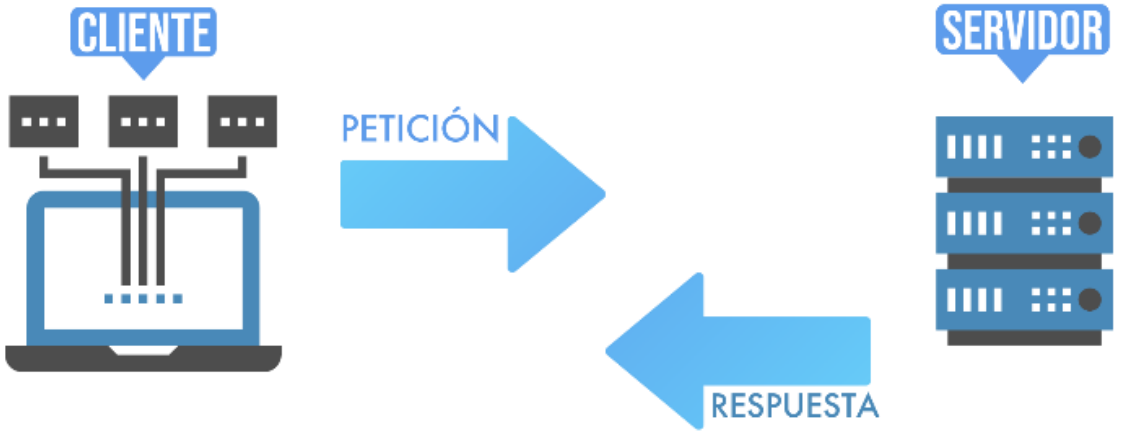


Este es un protocolo, donde se establece una comunicación que está diseñada para intercambiar mensaje, donde el cliente hace una petición (Solicitud) y el servidor responde (respuesta). Este es un patrón de comunicación es simple pero poderoso, especialmente en arquitecturas cliente-servidor.

Cliente – Servidor

Figura 35

Sistema Cliente - Servidor



La arquitectura de cliente servidor en simples palabras podemos definir, que el sistema proporciona múltiples procesos donde cliente solicita o hace una petición, y en el servidor le proporciona aquella petición en un tiempo estimado.

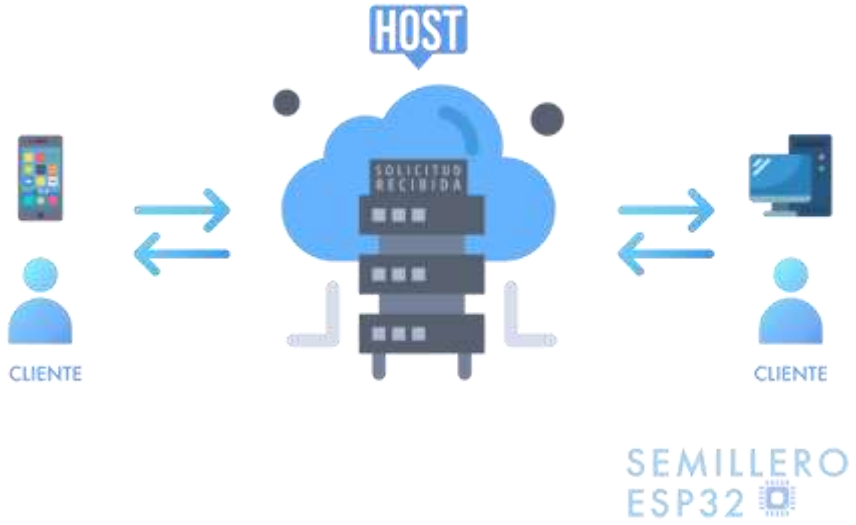
Cliente-Servidor, protagonizan un protocolo que construye un puente de diálogo entre dos puntos que interactúan, de un lado un PC, Smartphone u otro dispositivo, mediante un software específico utilizado por un usuario (por ejemplo, un navegador de Internet o un programa FTP), y del otro una estructura informática (denominando puntualmente al servidor, entendido como una potente computadora diseñada para estar activa y responder solicitudes de modo continuo). (Alsina, 2019)

Servidor host

Un servidor host o hosting, no es más que un servicio de alojamiento para nuestra aplicación web o página web. Un servidor es una computadora física que funciona ininterrumpidamente para que tu sitio web esté disponible todo el tiempo para cualquier persona que quiera verlo. (Gustavo B, 2021)

Figura 36

Configuración Host



Dirección IP

Figura 37

Representación de una dirección IP.

Cada dispositivo conectado a Internet cuenta con una dirección IP. Las direcciones IP son como los números de teléfono y tienen el mismo propósito. Cuando se pone en contacto con alguien, su número de teléfono identifica quién es usted y asegura a la persona que responde que es quien dice ser. Las direcciones IP hacen exactamente lo mismo cuando está en línea. (Andy Patrizio , 2019)

Servidor Web ESP32

El servidor web es un lugar para enviar y recibir información, procesar la información y almacenarla. El servidor web también puede mostrar esta información en una página web. El servidor se comunica con el usuario a través de un protocolo denominado Protocolo de transferencia de hipertexto (HTTP).

Cuando se envía una solicitud a este servidor (por ejemplo, se busca su dirección en el navegador), el servidor devuelve un código como respuesta (por ejemplo, el código 200, que significa que la conexión se ha establecido correctamente, o el código 404, que indica que la dirección no es correcta).

Station Mode

Cuando el ESP32 está configurado como una estación Wi-Fi, puede conectarse a otras redes (como su enrutador). En este escenario, el enrutador asigna una dirección IP única a su placa ESP. Puede comunicarse con el ESP usando otros dispositivos (estaciones) que también están conectados a la misma red consultando la dirección IP única del ESP.

Figura 38

Representación del Modo estación del Wi-Fi con el ESP32

Access point Mode

Cuando configura su tablero ESP32 como un punto de acceso, puede conectarse usando cualquier dispositivo con capacidades Wi-Fi sin conectarse a su enrutador. Cuando configura el ESP32 como un punto de acceso, crea su propia red Wi-Fi y los dispositivos (estaciones) Wi-Fi cercanos pueden conectarse a ella, como su teléfono inteligente o computadora. Por lo tanto, no necesita estar conectado a un enrutador para controlarlo.

Esto también puede ser útil si desea tener varios dispositivos ESP32 hablando entre sí sin la necesidad de un enrutador.

Figura 39

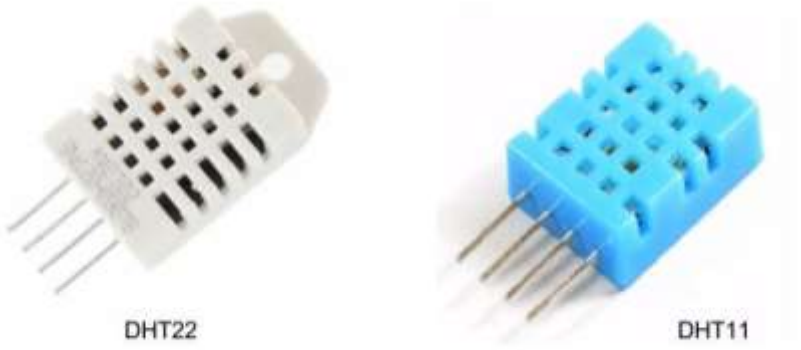
La tarjeta ESP32 en modo Access Point

Sensores, características, librerías necesarias y aplicaciones

Sensor DHT11 - DHT22

Figura 40

Sensores DHT22 y DHT11



Nota. Tomada de *DHT22 y DHT11 sensores de temperatura y humedad* [Imagen], por Arduino para todos, 2016 (http://arduparatodos.blogspot.com/2016/12/dht22-y-dht11-sensores-de-temperatura-y.html). CC BY 2.0

DHT11 y DHT22 Los sensores DHT11 y DHT22 son los más básicos y los más utilizados para implementarlos con Arduino, estos sensores están compuestos en dos partes, un sensor de humedad

capacitivo y un termistor, también constan de un circuito integrado básico en el interior que hace la conversión de analógico a digital y este envía una señal digital con la temperatura y la humedad (datasheetspdf, 2013).

La librería a usar es: DHT sensor library by Adafruit

DHT11

- Alimentación de 3.3V a 5VDC
- Corriente máxima 2.5mA durante la conversión
- Lectura de humedad con un +/- 5% de precisión
- Lectura de temperatura con un +/- 2°C de precisión
- Capaz de medir humedad de 20% a 80%
- Capaz de medir temperatura de 0 a 50°C
- No más de 1 Hz en velocidad de muestreo (una vez cada segundo)
- Dimensiones: 15.5mm x 12mm x 5.5mm

DHT 22

- Alimentación de 3.3V a 5VDC
- Corriente máxima 2.5mA durante la conversión
- Lectura de humedad con un +/- 2% a 5% de precisión
- Lectura de temperatura con un +/- 0.5°C de precisión
- Capaz de medir humedad de 0% a 100%
- Capaz de medir temperatura de -40°C a 125°C
- No más de 0.5Hz en velocidad de muestreo (una vez cada dos segundos)
- Dimensiones: 15.1mm x 25mm x 7.7mm

AHT20

Figura 41

Sensor AHT20

Nota. Tomada de *AHT20 Integrated temperature and humidity Sensor* [Imagen], por Guangzhou ASAIR Electronic Co., Ltd., 2022 (http://www.aosong.com/m/en/products-32.html). CC BY 2.0

AHT20, como una nueva generación de sensores de temperatura y humedad, ha establecido un nuevo estándar en tamaño e inteligencia. Está incrustado en un paquete sin plomo plano de doble fila apto para soldadura por reflujo, con un fondo de 3 x 3 mm y una altura de 1,0 mm. El sensor emite señales digitales calibradas en el estándar IAHT20, ya que una nueva generación de sensores de temperatura y humedad ha establecido un nuevo estándar en tamaño e inteligencia (aosong, s.f.).

Librería a usar: Adafruit AHTx0

Características del sensor:

- Uso con alimentación/lógica de 3.3 V o 5 V
- I2C dirección 0x38
- Rango de temperatura de funcionamiento: -40 a 85 °C (+ -1 °C de precisión típica en todo el rango)
- Rango de humedad relativa operativa: 0 a 100% de HR (+ -3% de precisión típica en todo el rango)

DS18B20

Figura 42

Sensor DS18B20

Nota. Tomada de *Sensor de Temperatura Digital DS18B20* [Imagen], por Naylamp Mechatronics SAC, 2022 (https://naylampmechatronics.com/sensores-temperatura-y-humedad/16-sensor-de-temperatura-digital-ds18b20.html). CC BY 2.0

El DS18B20 es un sensor digital de temperatura que utiliza el protocolo 1-Wire para comunicarse, este protocolo necesita solo un pin de datos para comunicarse y permite conectar más de un sensor en el mismo bus (naylampmechatronics, s.f.).

Librería a usar: OneWire

Características del sensor DS18B20:

- Rango de temperatura: -55 a 125°C
- Resolución: de 9 a 12 bits (configurable)
- Interfaz 1-Wire (Puede funcionar con un solo pin)
- Identificador interno único de 64 bits
- Múltiples sensores puede compartir el mismo pin
- Precisión: ±0.5°C (de -10°C a +85°C)
- Tiempo de captura inferior a 750ms
- Alimentación: 3.0V a 5.5V

BME280

Figura 43

Sensor BME280

Nota. Tomada de *Sensor de Presión, Temperatura y Humedad BME280* [Imagen], por Naylamp Mechatronics SAC, 2022 (https://naylampmechatronics.com/sensores-posicion-inerciales-gps/357-sensor-de-presion-temperatura-y-humedad-bme280.html). CC BY 2.0

El sensor BME280 integra en un solo dispositivo sensores de presión atmosférica, temperatura y humedad relativa, con gran precisión, bajo consumo energético y un formato ultra compacto. Basado en tecnología BOSCH piezo-resistiva con gran robustez EMC, alta precisión y linealidad, así como con estabilidad a largo plazo. Se conecta directamente a un microcontrolador a través de I2C o SPI (Naylamp Mechatronics, s.f.).

Librerías a usar: Para Arduino – cactus-io-BME280-SPI, Adafruit BME280 Library, Adafruit Sensor Master.

Características del sensor:

- Voltaje de Operación: 1.8V - 3.3V DC
- Interfaz de comunicación: I2C o SPI (3.3V)
- Rango de Presión: 300 a 1100 hPa (0.3-1.1bar)
- Resolución: 0.16 Pa
- Precisión absoluta: 1 hPa
- Rango de Temperatura: -40°C a 85°C
- Resolución de temperatura: 0.01°C
- Precisión Temperatura: 1°C

- Rango de Humedad Relativa: 0-100% RH
- Precisión de HR: +-3%
- Rango de altura medible: 0-9100 metros
- Ultra-bajo consumo de energía
- Completamente calibrado
- Frecuencia de Muestreo: 157 Hz (máx.)

BME680

Figura 44

Sensor BME680

Nota. Tomada de *Sensor de Presión, Temperatura, Humedad y Gas BME680* [Imagen], por Naylamp Mechatronics SAC, 2022 (https://naylampmechatronics.com/sensores-posicion-inerciales-gps/650-sensor-de-presion-temperatura-humedad-y-gas-bme680.html). CC BY 2.0

Es el primer sensor de gas que integra sensores de gas, presión, humedad y temperatura de alta linealidad y alta precisión. Está especialmente desarrollado para aplicaciones móviles y dispositivos portátiles donde el tamaño y el bajo consumo de energía son requisitos críticos (Bosch Sensortec, s.f.).

Librerías a usar: I2C como SPI son compatibles.

Características del sensor:

- Voltaje de alimentación: 3.3-5 VDC
- Interfaz de comunicación: I2C o SPI (3.3V)
- I2C-Dirección: 0x76/0x77
- Rango de Presión: 300-1100 hPa

- Resolución Presión: 0.18 Pa
- Precisión presión: +-0.12 hPa (aprox. +-1m)
- Rango de Temperatura: -40°C a 85°C
- Resolución de temperatura: 0.01°C
- Precisión Temperatura: +-1°C
- Rango de Humedad Relativa: 0-100% RH
- Resolución: 0.008% RH
- Precisión de HR: +-3%
- Ultra-bajo consumo de energía
- Completamente calibrado
- Frecuencia de Muestreo: 157 Hz (máx.)

HC-SR04

Figura 45

Sensor HC-SR04

Nota. Tomada de *Sensor de Ultrasonido HC-SR04* [Imagen], por Naylamp Mechatronics SAC, 2022 (https://naylampmechatronics.com/sensores-proximidad/10-sensor-ultrasonido-hc-sr04.html). CC BY 2.0

El uso de este módulo es bastante sencillo debido a que se toda la electrónica de control, transmisión y recepción se encuentra contenida en PCB. El usuario solamente debe enviar un pulso de disparo y medir en tiempo alto del pulso de respuesta, solamente se requieren 4 hilos para completar la interfaz con el módulo de sensor HC-SR04 (Geek Factory, 2022).

Librería a usar: HC-SR04lib

Características del sensor:

- Alimentación de 5 volts
- Interfaz sencilla: Solamente 4 hilos Vcc, Trigger, Echo, GND
- Rango de medición:2 cm a 400 cm
- Corriente de alimentación: 15 mA
- Frecuencia del pulso: 40 Khz
- Apertura del pulso ultrasónico: 15º
- Señal de disparo: 10uS
- Dimensiones del módulo: 45x20x15 mm.

MPU-6050

MPU6050 es un Acelerómetro Giroscopio 6DOF. El circuito integrado de InvenSense MPU6050 contiene un acelerómetro y giroscopio MEMS en un solo empaque. Es uno de los sensores más precisos del mercado con sus 16-bits de resolución, lo que significa que divide el rango dinámico en 65536 fracciones (Por ej. con ±2g que equivale a 4g de aceleración se tendría una sensibilidad de 4g/65536=61ug), los cual aplica para cada eje X, Y y Z al igual que en la velocidad angular. El sensor es ideal para diseñar control de robótica, medición de vibración, sistemas de medición inercial (IMU), detector de caídas, sensor de distancia y velocidad, y muchas cosas más (HeTPro, s.f.).

Figura 46

Sensor MPU-6050

Nota. Tomada de *Módulo Acelerómetro y giroscopio MPU6050* [Imagen], por Naylamp Mechatronics SAC, 2022 (https://naylampmechatronics.com/blog/45_tutorial-mpu6050-acelerometro-y-giroscopio.html). CC BY 2.0

Librería a usar: I2Cdev y MPU6050

Características del sensor:

- Salida digital de 6 ejes.
- Giroscopio con sensibilidad de ±250, ±500, ±1000, y ±2000dps
- Acelerómetro con sensibilidad de ±2g, ±4g, ±8g y ±16g
- Algoritmos embebidos para calibración
- Sensor de temperatura digital
- Entrada digital de video FSYNC
- Interrupciones programables
- Voltaje de alimentación: 2.37 a 3.46V
- Voltaje lógico: 1.8V±5% o VDD
- 10000g tolerancia de aceleración máxima

Sensor de ventana y puerta con cable, interruptor magnético aleatorio, tipo N.C, MC-38

Sensor magnético para puertas y ventanas. El mecanismo del sensor MC-38 es normalmente cerrado (NC), por lo cual, manda un 1 lógico cuando ambas partes del sensor están en contacto y 0 cuando están separadas (UNIT Electronics, 2022).

Características

- Modelo: MC-38
- Voltaje de alimentación máxima: 100V
- Corriente máxima: 0.5 A
- Potencia nominal: 3 W
- Mecanismo: Normalmente Cerrado (NC)
- Distancia de activación mínima: 15 mm
- Distancia de activación máxima: 25 mm
- Largo del cable: 25 cm
- Dimensiones: 27 mm x 14 mm x 8 mm
- Peso: 16 gramos

Figura 47

Sensor magnético MC-38

Nota. Tomada de *Sensor Magnético de puerta MC-38* [Imagen], por Naylamp Mechatronics SAC, 2022 (https://naylampmechatronics.com/sensores-proximidad/215-sensor-magnetico-de-puerta-mc-38.html). CC BY 2.0

Sensor de presencia infrarrojo PIR SR602

Este mini sensor PIR basado en MH-SR602 es ideal para detectar movimiento a una distancia de 0 a 3,5 m. Este módulo tiene alta sensibilidad, respuesta rápida, bajo consumo de energía estática y un tamaño pequeño, por lo que es fácil de instalar (MACTRONICA, 2022).

Características

- Distancia de detección: hasta 5 metros; distancia recomendada: 0-3,5 m.
- Salida: nivel alto, H = 3.3V, L = 0V Fuente de
- alimentación de CC: 3.3V-15V
- Corriente inactiva: 20uA
- El tiempo de alto nivel de la salida de este producto es ajustable, de 2.5 segundos a 1 hora, el tiempo de salida establecido en la fábrica.
- Son 2,5 segundos y puede cambiar una resistencia de chip si necesita cambiarla. Lea el valor de resistencia correspondiente al tiempo de retardo típico.
- Tiempo de bloqueo, 2 segundos, no ajustable.

Figura 48

Sensor PIR SR602

Nota. Tomada de *Sensor de Movimiento PIR MH-SR602* [Imagen], por Mactrónica, 2022 (https://www.mactronica.com.co/sensor-de-movimiento-pir-mh-sr602-sr602). CC BY 2.0

Fuente de alimentación inteligente HLK-5M03, HLK-5M05, HLK-5M12

Figura 49

Módulos de alimentación HLK-5M03, HLK-5M05, HLK-5M12

Nota. Tomada de *HLK-5M AC DC Convertidor 5W Fuente de Alimentación HLK-5M03/05/12* [Imagen], por UNIT Electronics, 2022 (https://uelectronics.com/producto/hlk-5m-ac-dc-convertidor-5w/). CC BY 2.0

HLK-5M AC DC Convertidor 5W es una pequeña fuente de alimentación que soporta un voltaje de entrada AC desde 100VAC hasta 240VAC y entrega una salida de voltaje DC fija. Contamos con diferentes modelos desde HLK-5M03 con voltaje de salida de 3.3V y corriente de salida de 1.5A,

HLK-5M05 con voltaje de salida de 5V y corriente de salida de 1A y HLK-5M12 con voltaje de salida de 12V y corriente de salida de 0. 42ª (UNIT Electronics, 2022).

Características

- Especificaciones generales:
- Tipo: Convertidores AC DC
- Marca: Hi-Link
- Modelo: Familia HLK-5M
- Color: Negro
- Material de Shell: Plástico
- Pines: 4 (2 para entrada AC) y (2 para voltaje de salida DC)
- Voltaje / Corriente entrada INPUT: 100-240V AC
- Frecuencia: 50 / 60Hz
- Potencia de salida: 5W
- Bajo rizado y bajo nivel de ruido
- Salida de protección contra sobrecarga y cortocircuito
- Protección: cortocircuito / sobrecarga / sobretensión
- Ciclo de vida: 100,000 horas
- HLK-5M03 3.3V DC 1.5A 5W

Modelo: HLK-5M03

- Voltaje de salida fijo: 3.3V DC
- Corriente de salida: Máx. 1.5A
- Led indicador: PWR
- Dimensiones (mm)
- HLK-5M05 5V DC 1A 5W

Modelo: HLK-5M05

- Voltaje de salida fijo: 5V DC
- Corriente de salida: Máx. 1A
- Led indicador: PWR
- Dimensiones (mm)

- HLK-5M12 12V DC 0.42mA 5W

  Modelo: HLK-5M12

- Voltaje de salida fijo: 12V DC
- Corriente de salida: Máx. 0.42A
- Led indicador: PWR
- Dimensiones (mm)

Reloj RTC DS3231

Es un módulo de tiempo real RTC (Real Time Clock) que podrás integrar a tus proyectos para que tengan la capacidad de almacenar y llevar la cuenta de fecha y hora, este módulo es compacto y de fácil utilización ya que tiene una interfaz de comunicación I2C para establecer comunicación con tarjetas de desarrollo o microcontroladores (UNIT Electronics, 2022).

Figura 50

Módulo RTC DS3231

Nota. Tomada de *Módulo RTC DS3231 Reloj de Tiempo Real* [Imagen], por UNIT Electronics, 2022 (https://uelectronics.com/producto/modulo-rtc-ds3231-reloj-de-tiempo-real/). CC BY 2.0

Características

- Voltaje de Alimentación DC: 3.3V ~ 5V
- Tipo de Comunicación: I2C
- Bajo consumo de energía
- Compatible con Arduino, Raspberry Pi y otras tarjetas.
- Exactitud Reloj: 2ppm

- Dirección I2C del DS3132: Read(11010001) Write(11010000)
- Memoria EEPROM AT24C32 (4K * 8bit = 32Kbit = 4KByte)
- RTC de alta precisión DS3231 con oscilador interno
- Puede ser usado en cascada con otro dispositivo I2C, la dirección del AT24C32 puede ser modificada (por defecto es 0x57)
- No incluye batería CR2032
- Dimensiones: 38 mm x 22 mm x 12 mm

Módulo de relé de 1, 2, 4 y 8 canales, 5V / 12V

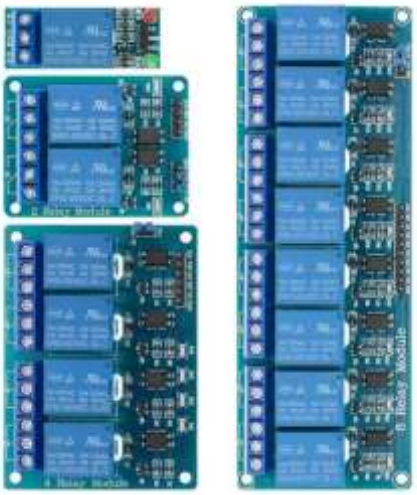
Dentro de la gran variedad de proyectos que podemos realizar con Arduino, podemos llegar a desear controlar componentes de alto voltaje o alto amperaje, como bombillas o bombas de agua, los cuales no pueden ser manejados directamente con Arduino. En estos casos es necesario utilizar Relays o Relés, estos dispositivos permiten controlar cargas de alto voltaje con una señal pequeña.

Características

- Voltaje de Operación: 5V DC
- Señal de Control: TTL (3.3V o 5V)
- Nº de Relays (canales): 1 CH – 2 CH – 4 CH – 8CH
- Capacidad máx: 10A/250VAC, 10A/30VDC
- Corriente máx: 10A (NO), 5A (NC)
- Tiempo de acción: 10 ms / 5 ms
- Para activar salida NO (normalmente abierto): 0 Voltios

Figura 51

Módulos de relé



Nota. Tomada de *Módulo Relevador 5V de 1 a 8 Canales* [Imagen], por UNIT Electronics, 2022 (https://uelectronics.com/producto/modulo-relevador-5v-de-1-a-8-canales/). CC BY 2.0

CAPITULO III – Fundamentos de programación

Definición de Arduino

Arduino es una plataforma de desarrollo basada en una placa electrónica de hardware libre que integra un microcontrolador programable y una serie de buses. Estos permiten establecer una conexión entre el microcontrolador y varios sensores y actuadores de una manera muy sencilla, principalmente utilizando un cable DuPont. (Arduino.cl, s.f.)

Figura 52

Tarjeta Arduino UNO R3

Nota. Tomada de *Arduino UNO* [Imagen], por MCI Electronics, 2022 (https://arduino.cl/arduino-uno/). CC BY 2.0

Para llevar a cabo la programación del microcontrolador Atmega328P que trae la placa Arduino UNO que se muestra en la Figura 51, se requiere el lenguaje C/C++. Para dicho efecto a continuación se presentan los fundamentos o conocimientos básicos de este lenguaje de programación.

Introducción al lenguaje C/C++

El lenguaje C creado por Bjarne Stroustrup a principios de la década de 1980 en Bell Labs, C puede considerarse un subconjunto de C, porque mantiene la velocidad y la eficiencia, facilita la comunicación entre el software, el software y el hardware, y muchas otras características que agregan soporte al modelo de programación orientado a objetos, este lenguaje fue creado con el propósito de

poder resolver muchas simulaciones estrictas dependientes de eventos que otros lenguajes no pueden lograr con el buen desempeño de C. (Rivera, 2020)

Tipos de Datos

En el ambiente Arduino es basado realmente en C++, que, con sus bibliotecas de soporte, puede asumir varios parámetros relativos al microcontrolador, para poder omitir parte del proceso de programación.

Existe una cantidad de tipos de datos muy distintos que suele definir C++ (Tabla 6), en donde a continuación se muestra una lista de tipos de datos que comúnmente se usa en el ambiente de Arduino, con sus respectivos tamaños de memoria. (Arduino.cl, s.f.)

Tabla 6

Tipos de datos

Tipos de Datos	Memoria que ocupa	Rango de valores
boolean	1 byte	0 o 1 (True o False)
byte / unsigned char	1 byte	0 -- 255
char	1 byte	-128 -- 127
int	2 bytes	-32.768 – 32.767
word / unsigned int	2 bytes	0 – 65.535
long	2 bytes	-2.147.483.648 – 2.147.483.647
unsigned	4 bytes	0 – 4.294.967.295
Float / double	4 bytes	-3,4028235E+38 – 3,4028235E+38
string	1 byte + x	Array de caracteres
array	1 byte + x	Colección de variables

Nota: Se detallan los tipos de datos y los valores numéricos y de memoria que pueden ocuparse.

Variables globales

En Arduino, una variable global es una variable visible que se puede extender a todo el programa, visible a todas las funciones y comandos, y estas variables deben declararse al principio del programa antes de la configuración anula la estructura *void setup().*

Variables locales

Estas variables son aquellas en que su visibilidad se puede limitar al bloque ”local” actual, donde las variables locales se pueden definir dentro de una función o en un bucle, estas pueden ser utilizadas dentro de una función ya declarada. (Arduino, s.f.)

Arrays

Una matriz es una colección de variables a las que se accede mediante el número de índice. Las matrices en el lenguaje de programación C para escribir bocetos de Arduino pueden ser complicadas, pero para trabajar con matrices simples es relativamente simple.

Para la creación de una variable, se deben tomar en cuenta todos los métodos en forma válida para declarar una matriz.

```
int myInts[6];
  int myPins[] = {2, 4, 8, 3, 6};
  int mySensVals[5] = {2, 4, -8, 3, 2};
  char message[6] = "hello";
```

Acceso a un arreglo de disco

Las matrices están indexadas a cero, es decir, referente a la inicialización de la matriz anterior, el primer elemento de la matriz está en el índice, por lo tanto:

mySensVals[0] == 2, mySensVals[1] == 4, y así sucesivamente.

También significa que, en una matriz con diez elementos, el índice nueve es el último elemento. Por lo tanto:

```
int myArray[10]={9, 3, 2, 4, 3, 2, 7, 8, 9, 11};
// myArray[9]    contiene 11
// myArray[10]   no es válido y contiene información aleatoria (otra dirección de memoria)
```

Boolean

Una variable booleana en sí, es una variable lógica la cual se puede definir que uno de los dos valores sea true o false. En Arduino el boolean está definido como un alias de tipo no estándar, la cual suele recomendarse usar el tipo estándar bool.

El tipo de datos booleano admite uno de dos valores, verdadero o falso. (Cada variable booleana ocupa un byte de memoria).

Sintaxis:

bool valorbool = true;

bool interruptor = false;

Byte

Un byte puede almacenar un número sin signo de hasta 8 bits, en un rango de 0 a 255.

Sintaxis: byte var = val;

Char

Es utilizado para almacenar un valor de caracteres. Los primeros caracteres se escriben entre comillas sencillas, como ‘A’ y para cadenas o múltiples caracteres se usan comillas dobles “ABC”.

Se puede observar también que estos caracteres son almacenados como números, que esto significa que se puede hacer procesos aritméticos con caracteres.

Sintaxis: char var = val;

Ejemplo:

char myChar = ‘A’;

char myChar = 65;

Double

Número de coma flotante de doble precisión, su tamaño es de 4 bytes, que la doble implementación es exactamente la misma que la flotante, sin ganar precisión.

Sintaxis: double var = val;

Float

Es un tipo de dato para números de coma flotantes, que tienen un punto decimal. Estos números de coma flotante son utilizados frecuentemente para aproximación de valores analógicos y continuos, porque son una solución mayor a la de los números enteros.

Sintaxis: float var = val;

Int

Estas variables son del tipo de dato principal para almacenamiento de números, en Arduino un int suele almacenar un valor de 16 bits.

Sintaxis: int var = val;

Long

Son variables largas de tamaño extendido para almacenar números de hasta 32 bits o 4 bytes. Si al hacer matemáticas con enteros de al menos uno de los valores, sea constante entera seguida de la L de variable tipo largo, suele forzarse a ser larga.

Sintaxis: long var = val;

Ejemplo: long speedOfLight_km_s = 300000L;

Short

Un short es un tipo de dato de 16 bits, que en la mayoría de Arduino puede almacenar 2 bytes.

Sintaxis: short var = val;

Ejemplo: short ledPin = 13

Size_t

Es un tipo de datos capaz de representar un tamaño de cualquier objeto en bytes

Sintaxis: Size_t var = val;

String

Son cadenas de texto que se pueden representar de dos maneras, que puede usar el tipo de dato string que forma parte del núcleo versión 0019, o también que puede crear una cadena a partir de una matriz tipo char y terminarla en null.

Declaraciones válidas para cadena:

- char Str1[15];
- char Str2[8] = {'a', 'r', 'd', 'u', 'i', 'n', 'o'};
- char Str3[8] = {'a', 'r', 'd', 'u', 'i', 'n', 'o', '\0'};
- char Str4[] = "arduino";
- char Str5[8] = "arduino";
- char Str6[15] = "arduino";

String()

Tiene la función de construir una instancia a partir de la clase string, también existen varias versiones para construir cadenas a partir de diferentes tipos de datos.

Sintaxis:

- String(val)
- String(val, base)
- String(val, decimalPlaces)

```
String stringOne = "Hello String";
String stringOne = String('a');
String stringTwo = String("This is a string");
String stringOne = String(stringTwo + " with more");
String stringOne = String(13);
String stringOne = String(analogRead(0), DEC);
String stringOne = String(45, HEX);
String stringOne = String(255, BIN);
String stringOne = String(millis(), DEC);
String stringOne = String(5.698, 3);
```

Void

La palabra clave solo se usa en declaraciones de función, indicando que se espera que la función no devuelva ninguna información a la función desde donde se la llamo.

Sintaxis:

```
void setup() {

}
```

Ejemplo:

```
void loop() {
}
```

Word

Es una palabra en la cual es posible almacenar números sin el signo de hasta máximo 16 bits.

Sintaxis: word var = val;

Ejemplo: word w = 1000;

Operadores Aritméticos y de Comparación

Los operadores aritméticos de la Tabla 7 suelen trabajar con operaciones básicas como suma, resta, multiplicación, división y modulo, nos ayuda a realizar cálculos matemáticos para una tarea en específico.

Tabla 7

Operadores aritméticos

x ++;	// equivale a x = x + 1 (incrementa x en 1)
x --;	// equivale a x = x – 1 (decrementa x en 1)
x += y;	// equivale a x = x + y (x es igual a x + el valor de y)
x -= y;	// equivale a x = x – y (x es igual a x – el valor de y)
x *= y;	// equivale a x = x * y (x es igual a x por el valor de y)
X /= y;	// equivale a x= x / y (x es igual a x dividido por y)

Los operadores de comparación de la Tabla 8 permiten establecer una diferencia o igualdad de tamaño o valor entre dos variables del mismo tipo.

Tabla 8

Operadores de comparación

x == y; //x será igual a y
x != y; //x será distinto de y
x <   y; //x será menor que y
x >   y; //x será mayor que y
x <= y; //x será menor o igual que y
x >= y; //x será mayor o igual que y

Operadores Lógicos o booleanos

Son utilizados para hacer comparaciones entre sus expresiones, que al realizarlas estas nos devuelven sea verdadero si cumple su condicionamiento (true) o falso si no cumple (false), en Arduino se trabajan solo con 3 operadores lógicos (Tabla 9). (codificada, 2019)

Tabla 9

Operadores Lógicos

Operadores Lógicos	
!	No lógico (NOT)
&&	“Y” lógico (AND)
||	“O” lógico (OR)

Estructuras de control

Estas estructuras funcionan de una manera similar a la de los lenguajes de programación más populares, en Arduino para realizar proyecto es imposible no usar estas estructuras de control ya que nos muy necesarias.

Condicional if

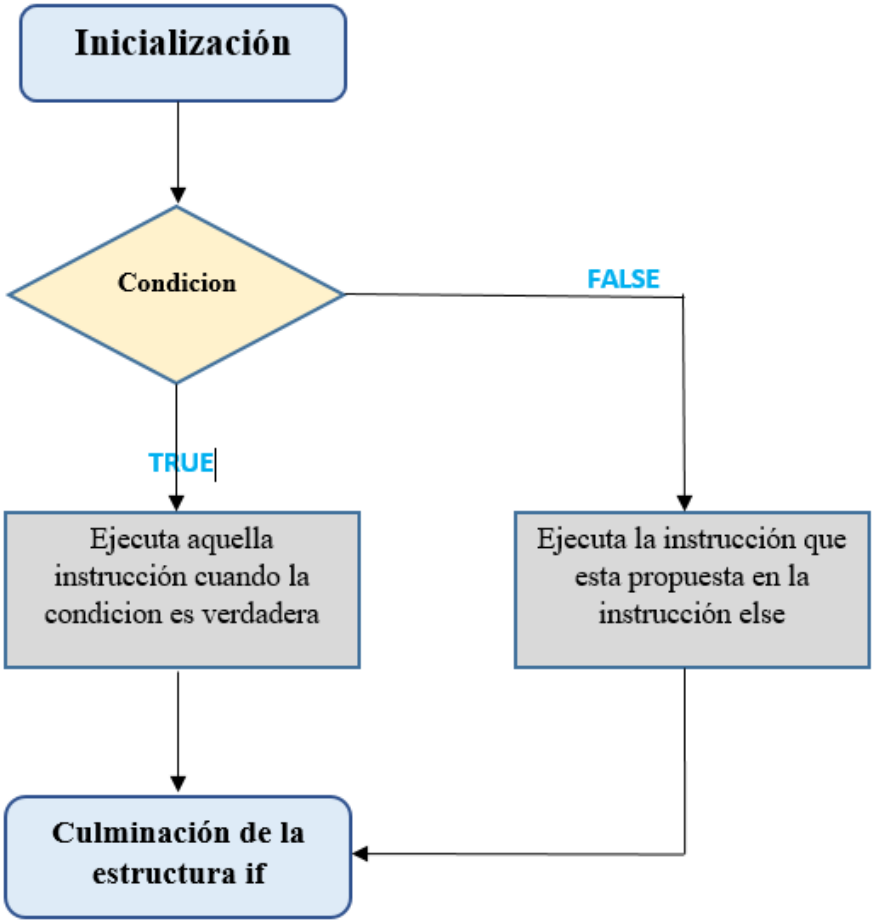
Comprueba si existe una condición y ejecuta la siguiente instrucción dada si la condición es verdadera “true”.

Sintaxis:

```
if (condicion) {
        //declaracion(s)
 }
```

Figura 53

Condicional if



Ejemplo:

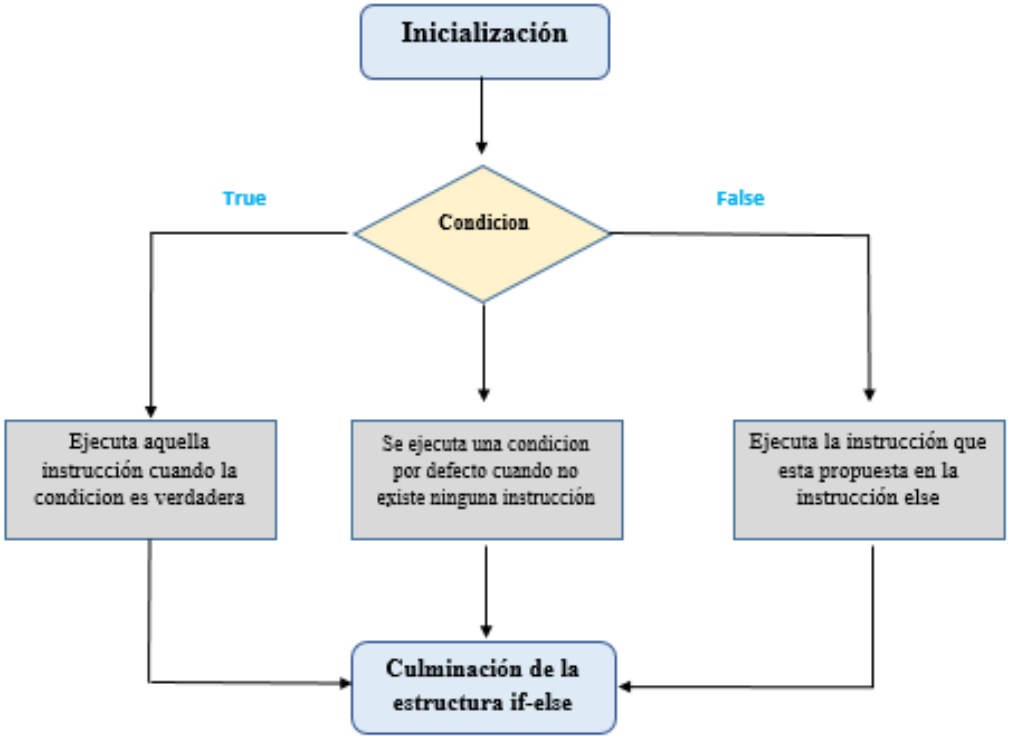
```
if (x > 120) digitalWrite(LEDpin, HIGH);

if (x > 120)
digitalWrite(LEDpin, HIGH);

if (x > 120) {digitalWrite(LEDpin, HIGH);}

if (x > 120) {
  digitalWrite(LEDpin1, HIGH);
  digitalWrite(LEDpin2, HIGH);
}
```

Estructura if-else

Tiene más control sobre los flujos de código que las instrucciones básicas, permite agrupar múltiples pruebas, en esta estructura se pueden definir también lo que comúnmente se le dice if anidado.

Figura 54

Estructura if-else



Sintaxis:

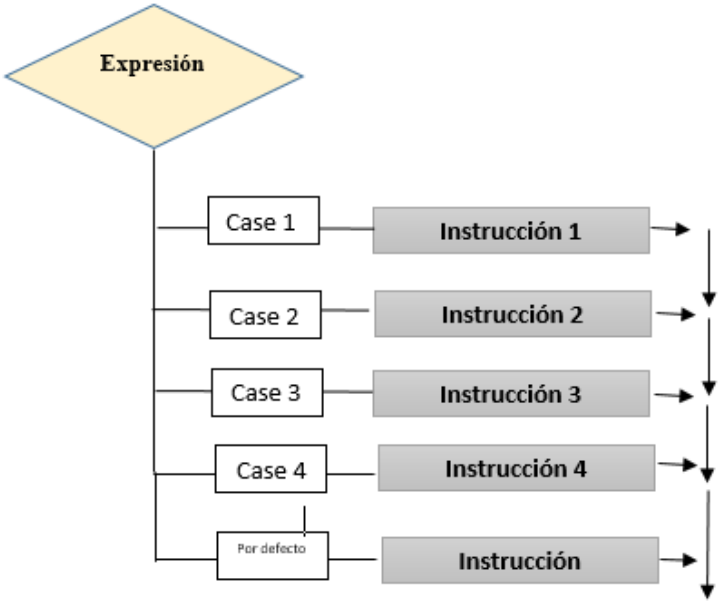
```
if (condition1) {
  // Hacer algo A
}else if (condition2) {
  // Hacer algo B
} else {
  // Hacer algo C
}
```

Estructura switch

Funciona de igual manera que las instrucciones if, switch case puede controlar los flujos de programas al permitir que se pueda especificar códigos donde se ejecutan diversas condiciones. En el caso de que este interruptor switch encuentre instrucciones se ejecutara la condición especificada cuyo valor coincida con el de la variable.

Figura 55

Estructura switch



Sintaxis:

```
switch (var) {
  case label1:
    // statements
    break;
  case label2:
    // statements
    break;
  default:
    // statements
    break;
}
```

Ciclo for

Se utiliza para realizar repeticiones de bloques de instrucciones cerradas, suele usarse con un contador de incrementos para incrementar y finalizar un bucle. Estas instrucciones son útiles para todo tipo de operaciones repetitivas y en muchas ocasiones se combinan en matrices para colecciones de datos.

Sintaxis:

```
for (initialization; condition; increment) {
  // statement(s);
}
```

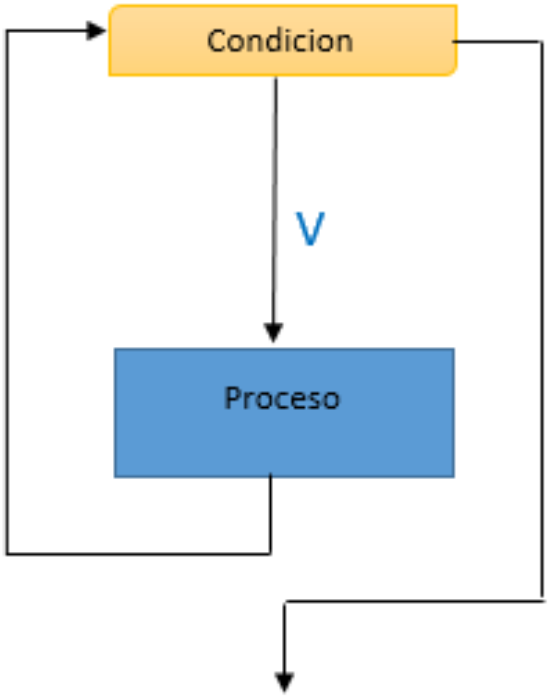
Ejemplo:

```
void setup() {
  // no se necesita configuracion
```

```
}
void loop() {
  for (int i = 0; i <= 255; i++) {
    analogWrite(PWMpin, i);
    delay(10);
  }
}
```

Figura 56

Ciclo for



Ciclo while

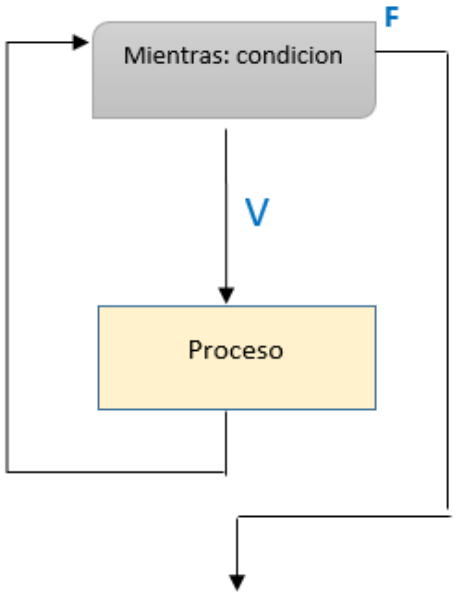
Un bucle se ira repitiendo continuamente y de forma infinita, siempre y cuando la expresión dentro del paréntesis devuelva un valor falso, se debe de modificar la variable probada para este bucle while salga y concluya su condición, esta podría estar asignada en el código como una variable incremental.

Sintaxis:

```
while (condition) {
  // statement(s)
}
```

Figura 57

Ciclo while



Ejemplo:

```
var = 0;
while (var < 200) {
  // hace algo repetitivo 200 veces
  var++;
}
```

Ciclo do while

Este bucle funciona de igual manera que el while, con una pequeña diferencia que cuando la condición se prueba al final del bucle, por lo que este bucle do siempre se ejecuta al menos 1 vez.

Sintaxis:

```
do {
  // statement block
} while (condition);
```

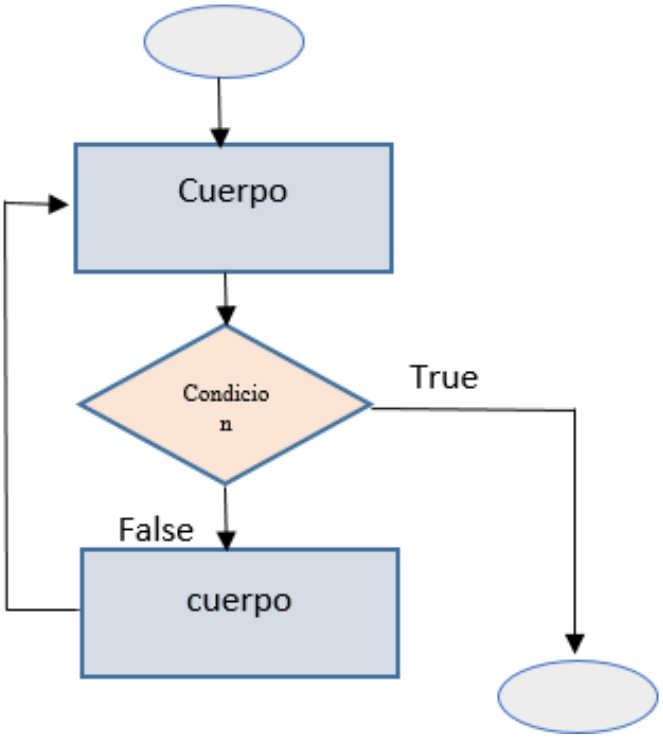
Tabla 10

Ejemplo usando el ciclo do while

```
1  int x = 0;
2  do {
3    delay(50);          // espera a que los sensores se estabilicen
4    x = readSensors();  // revisa los sensores
5  } while (x < 100);
```

Figura 58

Ciclo do-while



Continue

Esta sentencia tiene como función omitir el resto de iteración de un bucle es decir de (do, for, o while). Luego de eso sigue la comprobación de aquella expresión condicional del bucle y procedida de cualquier iteración posterior.

Tabla 11

Ejemplo usando la sentencia continue

```
1  for (int x = 0; x <= 255; x ++) {
2    if (x > 40 && x < 120) {  // crea salto en valores
3      continue;
4    }
5
6    analogWrite(PWMpin, x);
7    delay(50);
8  }
```

Break

Es utilizado para salir de un bucle do, for, while o de una instrucción switch.

Tabla 12

Ejemplo usando la sentencia break

```
1  int threshold = 40;
2  for (int x = 0; x < 255; x++) {
3    analogWrite(PWMpin, x);
4    sens = analogRead(sensorPin);
5    if (sens > threshold) {     // rescata la detección del sensor
6      x = 0;
7      break;
8    }
9    delay(50);
10 }
```

Return

Termina una función y retorna un valor de una función a la función llamada, si esta lo desea.

Sintaxis: return value;

Tabla 133

Ejemplo usando la sentencia return

```
1  int checkSensor() {
2    if (analogRead(0) > 400) {
3      return 1;
4    }
5    else {
6      return 0;
7    }
8  }
```

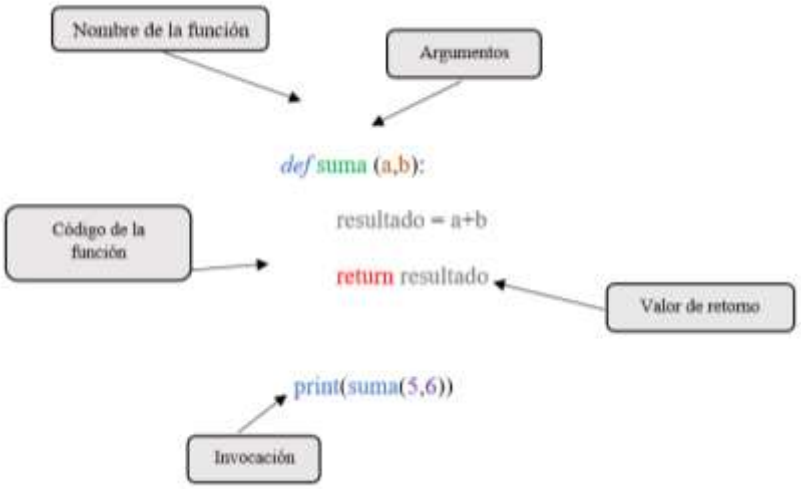
Funciones

Estas funciones son bloques de código que tiene nombre y conjunto de instrucciones donde son ejecutadas cuando se llama la función. Las funciones *setup()* y *loop()* son de las que comúnmente se han descrito. Estas funciones pueden ser escritas para realizar tareas repetitivas que son útiles para reducir código en un programa.

Las funciones son declaradas asociadas a un tipo de valor, este valor devuelve una función, si una función es de tipo int devuelve un dato numérico de tipo entero, pero si la función no devuelve valor, entonces se debe colocar delante la palabra “void” que significa “función vacía”. (wordpress, 2016)

Figura 59

Funciones



Arreglos

Cadena de caracteres y tipo String

String es un tipo de dato que se usa para guardar cadenas de caracteres, estos caracteres son datos de 8 bits representados en ASCII, las variables string también deben siempre terminar en carácter null. Es de mucha utilidad cuando queremos armar mensajes de texto concatenados para ser enviados por el puerto serial, en Arduino el tipo de dato string se almacena en un vector de datos de tipo carácter. (Herramientas tecnologicas profesionales, s.f.)

Sintaxis: string nombreVariable;

Ejemplo de string-concatenación:

Tabla 14

Ejemplo de string-concatenación:

1	String ejemplo1 = "Hola ";
2	String ejemplo2 = "Mundo!";
3	String resultado;

```
4   void setup() {
5     resultado = ejemplo1+ejemplo2;
6     Serial.begin(9600);
7     Serial.print("El resultado de la concatenacion es: ");
8     Serial.println(resultado);
9
10  }
11
12  void loop() {
13    // agrega código principal aquí, para que se ejecute rápidamente:
14
15  }
```

Struct C++

Struct es un tipo de datos definido por el usuario disponible en C. Permite a los usuarios combinar elementos de datos de (posiblemente) diferentes tipos de datos bajo un solo nombre.

La estructura C es diferente de la matriz en que una matriz solo puede contener datos de tipos de datos similares, por otro lado, la estructura C puede almacenar datos de múltiples tipos de datos. (Educative, s.f.)

Tabla 15

Sintaxis del tipo de dato Struct

```
1   struct structure_name;
2   {
3       //data_type variable 1
4     //data_type variable 2
5       //data_type variable 3
6     …
7   };
```

Tabla 16

Ejemplo usando el tipo de dato Struct

```
1    struct employee
2    {
3        char name[27];
4     int age;
5        float salary;
6     …
7    };
```

Fundamentos de diseño de páginas web

Para el desarrollo de páginas web se requiere el uso de varias tecnologías como lo son HTML, CSS y JAVASCRIPT, cada una con su propio conjunto de reglas y funciones dentro de la página. HTML se encarga de la estructura, CSS de la presentación y estilo de dicha estructura y el contenido en la pantalla por último JAVASCRIPT se encarga de la funcionalidad de la página, todas estas tecnologías y algo más veremos más adelante a lo largo de este documento.

HTML5

HTML5 es la versión vigente de HTML siglas que significan HyperText Markup Language o en español Lenguaje de Marcas de Hipertexto, es el componente fundamental de toda página web dado que mediante su uso se define la estructura sobre la cual se montara el contenido del sitio web a diseñar. La estructura debe proporcionar forma, organización y flexibilidad, y debe ser sólida como los cimientos de un edificio. Para trabajar y crear sitios web y aplicaciones con HTML5, primero debemos entender cómo se construye la estructura. Construir una base sólida nos ayudará más adelante, cuando se integren las tecnologías restantes para obtener la mejor página web posible. Los documentos HTML están estrictamente organizados cada parte de los documentos se diferencian, declaran e identifican mediante etiquetas específicas a continuación veremos cómo construir la estructura global de un documento HTML y los nuevos elementos semánticos incluidos en HTML5.

DOCTYPE

<!DOCTYPE html>

Mediante el uso de esta etiqueta indicamos el tipo de documento que estamos creando, de esta manera iniciamos la creación de nuestro documento HTML5 es importante recalcar que esta debe ser la primera línea del archivo, para que el navegador pueda identificar el documento.

html

```
<!DOCTYPE html>
<html lang=”es”>
</html>
```

Una vez declarado el tipo de documento procedemos con la base de la página web, si se permite el uso de una analogía sería la de un árbol ya que HTML tiene su raíz en la etiqueta html, dentro de esta haremos uso del atributo lang para especificar el idioma en el que escribiremos el contenido de la página.

head

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

Dentro de la etiqueta html tenemos que separar el contenido en dos, la parte superior será el encabezado delimitado por la etiqueta head, todo lo que pongamos entre las etiquetas de apertura y cierre no se va a mostrar con excepción del título que se mostrara en la pestaña de nuestro navegador.

meta

```
<!DOCTYPE html>
<html lang="es">
<head>
 <meta charset="iso-8859-1">
 <meta name=”description” content=”Ejemplo de HTML5”>
 <meta name=”keywords” content=”HTML5, CSS3, Javascript”>
</head>
</html>
```

Esta etiqueta es de las nuevas características implementadas en HTML5, en ella podemos definir el conjunto de caracteres a utilizar en la página, también podemos añadir una descripción o palabras claves para que el navegador o motor de búsqueda pueda tener una vista previa del contenido del sitio.

title

```
<!DOCTYPE html>
<html lang="es">
<head>
 <meta charset="iso-8859-1">
 <meta name="description" content="Ejemplo de HTML5">
 <meta name="keywords" content="HTML5, CSS3, JavaScript">
 <title>Este texto es el título del documento</title>
</head>
</html>
```

Definimos el título del documento, este será visible en la pestaña del navegador que estemos usando.

link

```
<!DOCTYPE html>
<html lang="es">
<head>
 <meta charset="iso-8859-1">
 <meta name="description" content="Ejemplo de HTML5">
 <meta name="keywords" content="HTML5, CSS3, JavaScript">
 <title>Este texto es el título del documento</title>
 <link rel=”stylesheet” href=”misestilos.css”>
</head>
</html>
```

Es utilizada para agregar archivos externos a nuestros documentos, estos pueden variar desde hojas de estilo CSS, código de JAVASCRIPT o simplemente agregar iconos e imágenes, haremos usos de los atributos rel y href, en el primero definiremos el tipo de relación entre el documento local y el archivo a agregar.

body

```
<!DOCTYPE html>
<html lang="es">
<head>
</head>
<body>
</body>
</html>
```

La segunda mitad de nuestro documento, todos los elementos que pongamos en esta sección serán visibles en el sitio.

Nota: Para utilizar una etiqueta tendremos que hacer uso de una etiqueta de apertura y una de cierre para delimitar el campo de acción de la misma y así como mantener el orden de nuestro sitio web

Figura 60

Estructura común de un sitio web

Nota. Tomada de *Etiquetas HTML* [Imagen], Ingeniería Systems, 2022 (http://www.ingenieriasystems.com/2015/07/Etiquetas-header-nav-section-aside-y-footer-en-html.html). CC BY 2.0

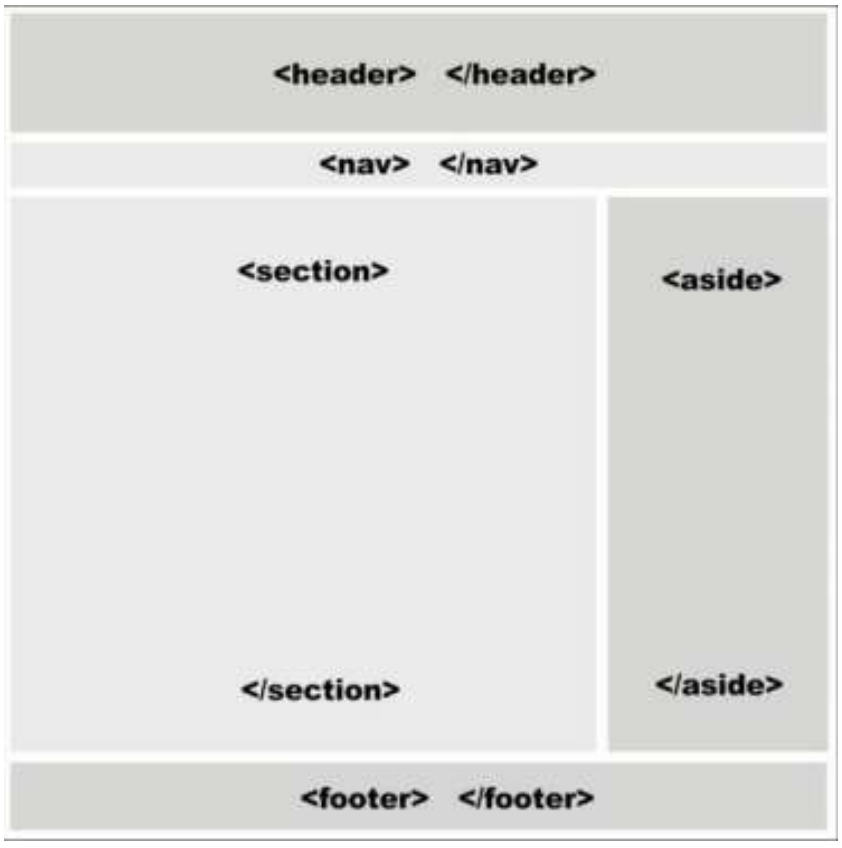
Estructura del cuerpo

Aunque cada diseñador web tiene total libertad de realizar su propio diseño, es muy común encontrarse con una estructura similar a la figura 1, dentro de este modelo nos encontramos con:

- **Cabecera:** Parte superior donde encontramos el logo, título y subtitulo del sitio web
- **Barra de navegación:** Barra con un menú o enlaces para facilitar la navegación dentro de la página
- **Información principal:** El contenido principal de la página se encuentra en esta sección he ahí su posición en el centro, muchas veces es dividido en filas y columnas
- **Barra lateral:** Dependiendo de las necesidades del diseñador esta sección podría contar con enlaces destacados como noticias o artículos destacados
- **Institucional:** También conocido como pie de página, aquí ponemos información acerca de la institución, autor, redes sociales e información adicional que el diseñador desee agregar

Figura 61

Estructura común de un sitio web en etiquetas HTML5



Nota. Tomada de *Etiquetas HTML* [Imagen], Ingeniería Systems, 2022 (http://www.ingenieriasystems.com/2015/07/Etiquetas-header-nav-section-aside-y-footer-en-html.html). CC BY 2.0

header

```
<!DOCTYPE html>
<html lang="es">
<head>
 <meta charset="iso-8859-1">
 <meta name="description" content="Ejemplo de HTML5">
 <meta name="keywords" content="HTML5, CSS3, JavaScript">
 <title>Este texto es el título del documento</title>
 <link rel="stylesheet" href="misestilos.css">
</head>
<body>
 <header>
 <h1>Este es el título principal del sitio web</h1>
 </header>
</body>
</html>
```

Header o encabezado es una de las nuevas etiquetas que se agregaron a HTML, no confundirse con head, el propósito del encabezado es el de proveer al usuario con información introductoria, este indica el principio del cuerpo del documento.

nav

```
<!DOCTYPE html>
<html lang="es">
<head>
 <meta charset="iso-8859-1">
 <meta name="description" content="Ejemplo de HTML5">
 <meta name="keywords" content="HTML5, CSS3, JavaScript">
 <title>Este texto es el título del documento</title>
 <link rel="stylesheet" href="misestilos.css">
</head>
<body>
 <header>
```

```
<h1>Este es el título principal del sitio web</h1>
</header>
<nav>
<ul>
<li>principal</li>
<li>fotos</li>
<li>videos</li>
<li>contacto</li>
</ul>
</nav>
</body>
</html>
```

La barra de navegación es generada a través de la etiqueta nav, dentro de ella encontraremos distintos links que nos redirigirán a distintas secciones dentro del mismo documento o uno externo

section

```
<!DOCTYPE html>
<html lang="es">
<head>
<meta charset="iso-8859-1">
<meta name="description" content="Ejemplo de HTML5">
<meta name="keywords" content="HTML5, CSS3, JavaScript">
<title>Este texto es el título del documento</title>
<link rel="stylesheet" href="misestilos.css">
</head>
<body>
<header>
<h1>Este es el título principal del sitio web</h1>
</header>
<nav>
<ul>
<li>principal</li>
```

```
 <li>fotos</li>
 <li>videos</li>
 <li>contacto</li>
 </ul>
 </nav>
 <section>
 </section>
</body>
</html>
```

Esta etiqueta sirve para delimitar una sección en nuestro documento, esto lo usaremos para establecer nuestra sección principal

aside

```
<!DOCTYPE html>
<html lang="es">
<head>
 <meta charset="iso-8859-1">
 <meta name="description" content="Ejemplo de HTML5">
 <meta name="keywords" content="HTML5, CSS3, JavaScript">
 <title>Este texto es el título del documento</title>
 <link rel="stylesheet" href="misestilos.css">
</head>
<body>
 <header>
 <h1>Este es el título principal del sitio web</h1>
 </header>
 <nav>
 <ul>
 <li>principal</li>
 <li>fotos</li>
 <li>videos</li>
 <li>contacto</li>
```

```
 </ul>
 </nav>
 <section>
 </section>
 <aside>
 <blockquote>Mensaje número uno</blockquote>
 <blockquote>Mensaje número dos</blockquote>
 </aside>
</body>
</html>
```

Este elemento no cuenta con una posición definida, puede estar tanto a la izquierda como a la derecha de la página, sirve para establecer información adicional que creamos relevante.

footer

```
<!DOCTYPE html>
<html lang="es">
<head>
 <meta charset="iso-8859-1">
 <meta name="description" content="Ejemplo de HTML5">
 <meta name="keywords" content="HTML5, CSS3, JavaScript">
 <title>Este texto es el título del documento</title>
 <link rel="stylesheet" href="misestilos.css">
</head>
<body>
 <header>
 <h1>Este es el título principal del sitio web</h1>
 </header>
 <nav>
 <ul>
 <li>principal</li>
 <li>fotos</li>
 <li>videos</li>
```

```
<li>contacto</li>
</ul>
</nav>
<section>
</section>
<aside>
<blockquote>Mensaje número uno</blockquote>
<blockquote>Mensaje número dos</blockquote>
</aside>
<footer>
Derechos Reservados &copy; 2010-2011
</footer>
</body>
</html>
```

El último elemento, pero no el menos importante, aquí indicaremos información sobre la institución y el autor del sitio, así como información que el diseñador considere importante.

Etiquetas usadas dentro del cuerpo

article

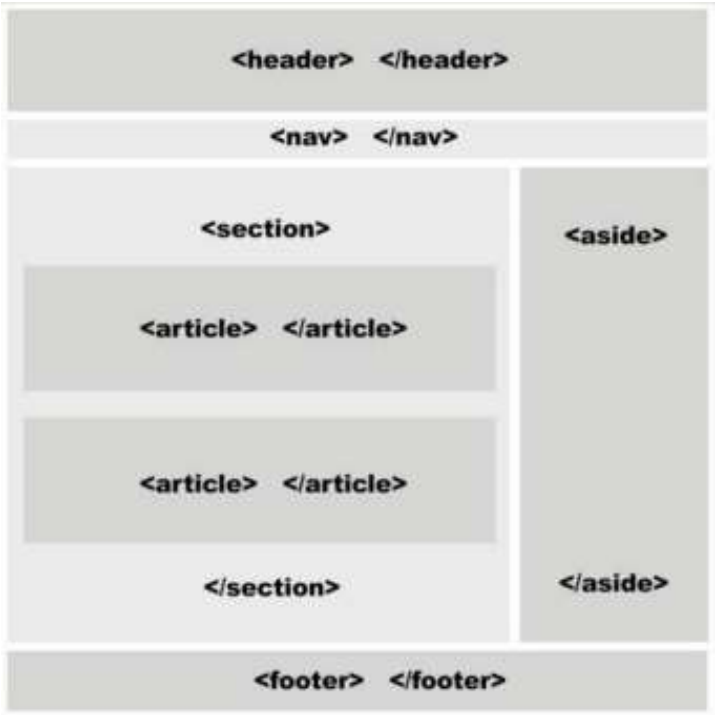
```
<!DOCTYPE html>
<html lang="es">
<head>
<meta charset="iso-8859-1">
<meta name="description" content="Ejemplo de HTML5">
<meta name="keywords" content="HTML5, CSS3, JavaScript">
<title>Este texto es el título del documento</title>
<link rel="stylesheet" href="misestilos.css">
</head>
<body>
<header>
<h1>Este es el título principal del sitio web</h1>
</header>
```

```
<nav>
<ul>
<li>principal</li>
<li>fotos</li>
<li>videos</li>
<li>contacto</li>
</ul>
</nav>
<section>
<article>
Este es el texto de mi primer mensaje
</article>
<article>
Este es el texto de mi segundo mensaje
</article>
</section>
<aside>
<blockquote>Mensaje número uno</blockquote>
<blockquote>Mensaje número dos</blockquote>
</aside>
<footer>
Derechos Reservados &copy; 2010-2011
</footer>
</body>
</html>
```

La etiqueta article tiene la finalidad de contener contenido independiente, no hay que dejarse llevar por su nombre no se limita a artículos de noticias, también puede contener mensajes de foros, entradas de un blog, comentarios de un usuario etc.

Figura 62

Etiquetas article dentro de la estructura



Nota. Tomada de *Etiquetas HTML* [Imagen], Ingeniería Systems, 2022 (http://www.ingenieriasystems.com/2015/07/Etiquetas-header-nav-section-aside-y-footer-en-html.html). CC BY 2.0

hgroup

Las etiquetas h como h1, h2 hasta h6 sirven para crear una línea de cabecera o subtítulos debajo de la misma, si queremos hacer esto es buena práctica agruparlas con la etiqueta hgroup, estas etiquetas deben preservar su jerarquía es decir para usar un subtitulo con h2 primero necesitamos declarar el titulo con h1

```
<!DOCTYPE html>
<html lang="es">
<head>
 <meta charset="iso-8859-1">
 <meta name="description" content="Ejemplo de HTML5">
 <meta name="keywords" content="HTML5, CSS3, JavaScript">
```

```
 <title>Este texto es el título del documento</title>
 <link rel="stylesheet" href="misestilos.css">
</head>
<body>
 <header>
 <h1>Este es el título principal del sitio web</h1>
 </header>
 <nav>
 <ul>
 <li>principal</li>
 <li>fotos</li>
 <li>videos</li>
 <li>contacto</li>
 </ul>
 </nav>
 <section>
 <article>
 <header>
 <hgroup>
 <h1>Título del mensaje uno</h1>
 <h2>Subtítulo del mensaje uno</h2>
 </hgroup>
 <p>publicado 10-12-2011</p>
</header>
 Este es el texto de mi primer mensaje
 <footer>
 <p>comentarios (0)</p>
 </footer>
 </article>
 <article>
 <header>
 <hgroup>
 <h1>Título del mensaje dos</h1>
```

<h2>**Subtítulo del mensaje dos**</h2>
</hgroup>
publicado 15-12-2011
</header>
Este es el texto de mi segundo mensaje
<footer>
comentarios (0)
</footer>
</article>
</section>
<aside>
<blockquote>**Mensaje número uno**</blockquote>
<blockquote>**Mensaje número dos**</blockquote>
</aside>
<footer>
Derechos Reservados © **2010-2011**
</footer>
</body>

figure y figcaption

Estas etiquetas sirven para ser más específicos a la hora de incluir un archivo multimedia como imágenes, videos, gif, figcaption nos sirve para poner una breve descripción del contenido agregado, para agregar una imagen hacemos uso de la etiqueta img y sus atributos src donde indicamos la dirección local o externa del archivo.

<!DOCTYPE html>
<html lang="**es**">
<head>
<meta charset="**iso-8859-1**">
<meta name="**description**" content="**Ejemplo de HTML5**">
<meta name="**keywords**" content="**HTML5, CSS3, JavaScript**">
<title>**Este texto es el título del documento**</title>
<link rel="**stylesheet**" href="**misestilos.css**">

```
</head>
<body>
<header>
<h1>Este es el título principal del sitio web</h1>
</header>
<nav>
<ul>
<li>principal</li>
<li>fotos</li>
<li>videos</li>
<li> >contacto</li>
</ul>
</nav>
<section>
<article>
<header>
<hgroup>
<h1>Título del mensaje uno</h1>
<h2>Subtítulo del mensaje uno</h2>
</hgroup>
<p>publicado 10-12-2011</p>
</header>
Este es el texto de mi primer mensaje
<figure>
<img src=”http://minkbooks.com/content/myimage.jpgg”>
<figcaption>
Esta es la imagen del primer mensaje
</figcaption>
</figure>
<footer>
<p>comentarios (0)</p>
</footer>
</article>
```

<article>
<header>
<hgroup>
<h1>**Título del mensaje dos**</h1>
<h2>**Subtítulo del mensaje dos**</h2>
</hgroup>
publicado 15-12-2011
</header>
Este es el texto de mi segundo mensaje
<footer>
comentarios (0)
</footer>
</article>
</section>
<aside>
<blockquote>**Mensaje número uno**</blockquote>
<blockquote>**Mensaje número dos**</blockquote>
</aside>
<footer>
Derechos Reservados © **2010-2011**
</footer>
</body>

CSS

Son las siglas en inglés para hojas de estilo en cascada (cascading style sheets) es la tecnología encargada de darle estilos a las páginas web, su integración con HTML es esencial para el diseño de sitios web, aunque cada elemento HTML cuenta con un estilo por defecto este no suele ser suficiente para suplir las necesidades del diseñador.

Los elementos son ordenados de acuerdo a su tipo entre estos tenemos

- **Elementos block:** Son bloques donde cada uno se posiciona abajo del otro
- **Elementos inline:** Se posicionan uno alado del otro, en la misma línea

Elementos como section, nav, header, footer son elementos de bloques y cada uno es posicionado abajo del anterior.

Conceptos básicos sobre estilos

Para la aplicación de estilos CSS tenemos tres maneras distintas de hacerlas estas son:

Estilos en línea

Es la forma más sencilla de aplicarlo para hacerlo haremos uso del atributo style de cada una de las etiquetas HTML, en el siguiente código se modificará el tamaño de fuente de un párrafo

```
<!DOCTYPE html>
<html lang="es">
<head>
 <title>Este es el título del documento</title>
</head>
<body>
 <p style=”font-size: 20px”>Mi texto</p>
</body>
</html>
```

Estilos embebidos

De esta manera agregamos los estilos en la cabecera del documento HTML, el inconveniente es que mientras más elementos tengamos el archivo se hará demasiado extenso para nuestro propio bien

```
<!DOCTYPE html>
<html lang="es">
<head>
 <title>Este texto es el título del documento</title>
 <style>
 p { font-size: 20px }
 </style>
</head>
<body>
 <p>Mi texto</p>
```

```
</body>
</html>
```

Archivos externos:

Declarar los estilos de esta manera ahorra espacio y se podría decir es la forma predilecta de los desarrolladores para agregar estilos a sus páginas, estos archivos se pueden agregar gracias a la etiqueta link en la cabecera del documento.

```
<!DOCTYPE html>
<html lang="es">
<head>
 <title>Este texto es el título del documento</title>
 <link rel="stylesheet" href="misestilos.css">
</head>
<body>
 <p>Mi texto</p>
</body>
</html>
```

Referencias CSS

Si queremos personalizar aún más nuestro documento HTML y queremos que las etiquetas tengan su estilo propio que no afecte a sus pares como por ejemplo el párrafo introductorio de color azul y los demás de rojo, lo podemos hacer gracias a las referencias y en CSS hay algunas maneras para seleccionar que etiquetas serán afectadas por los estilos

Referencia por la palabra clave del elemento

Mediante esta referencia afectamos a todas las etiquetas con el mismo nombre ejemplo en el código siguiente todas las etiquetas p tendrán un tamaño de fuente de 20

```
p { font-size: 20px }
```

Referencia por el atributo id

El atributo id sirve como un identificador de un elemento, cabe recalcar que solo puede existir un elemento con un id, este no puede repetirse, para referenciarlo haremos uso del #.

```
#texto1 { font-size: 20px }
```

```
<!DOCTYPE html>
<html lang="es">
<head>
 <title>Este texto es el título del documento</title>
 <link rel="stylesheet" href="misestilos.css">
</head>
<body>
 <p id=”texto1”>Mi texto</p>
</body>
</html>
```

Referencia por el atributo class

Funciona de manera similar a id pero este si permite repetirse, con este agruparemos varios elemento con características en común he de ahí el nombre de clase, para referenciarlo antepondremos el punto antes del nombre de la clase

```
.texto1 { font-size: 20px }
```

```
<!DOCTYPE html>
<html lang="es">
<head>
 <title>Este texto es el título del documento</title>
 <link rel="stylesheet" href="misestilos.css">
</head>
<body>
 <p class=”texto1”>Mi texto</p>
 <p class=”texto1”>Mi texto</p>
 <p>Mi texto</p>
</body>
```

```
</html>
```

Referencia por pseudo clases

Si queremos hacer la referencia aún más específica haremos el uso de pseudo clases como en el siguiente caso

```
<!DOCTYPE html>
<html lang="es">
<head>
 <title>Este texto es el título del documento</title>
 <link rel="stylesheet" href="misestilos.css">
</head>
<body>
 <div id="wrapper">
 <p class="mitexto1">Mi texto1</p>
 <p class="mitexto2">Mi texto2</p>
 <p class="mitexto3">Mi texto3</p>
 <p class="mitexto4">Mi texto4</p>
 </div>
</body>
</html>
```

Fijémonos en el div con el id wrapper dentro de el tiene cuatro párrafos cada uno con una clase distinta, estos elementos son hijos de la etiqueta div, gracias a las pseudoclases podremos referenciarlos sin conocer su atributo clase mediante el siguiente código.

```
p:nth-child(2){
 background: #999999;
}
```

Para llamar a una pseudo clase usamos los dos puntos y en este caso nth-child(2) que especifica al segundo hijo y le cambiamos el fondo por un color gris.

Referencia Universal

Si queremos que reglas específicas afecten a todo el documento haremos uso del selector universal *, todo lo que este declarado entre sus llaves afectara a todo el documento sin importar id, clase o elemento.

```
*{
 margin: 0px;
}
```

Nuevo Selectores

Dentro de las novedades de CSS3 nos encontramos con el selector > y + cada uno con características específicas de cada uno.

Selector de padre hijo: > sirve para indicar que el elemento de la derecha se vere afectado siempre y cuando tenga como padre al elemento de la izquierda

```
div > p.mitexto2{
 color: #990000;
}
```

Selector adyacente: + selecciona el elemento de la derecha siempre y cuando este antecedido por el elemento de la izquierda

```
p.mitexto2 + p{
 color: #990000;
}
```

JAVASCRIPT

Javascript es un lenguaje interpretado por el navegador, esto quiere decir que el código se ejecuta directamente sin la necesidad de ser compilado, este será el encargado de la funcionalidad de la página. Al igual que CSS hay tres maneras de poner integrarlo a nuestro documento HTML.

En línea

El método más simple para poder integrar el código, lo haremos mediantes algunos atributos de las etiquetas HTML como onClick, onMouseOver, etc., estos atributos son manejadores de eventos que ejecutan código de acuerdo a las acciones del usuario.

```
<!DOCTYPE html>
```

```
<html lang=”es”>
<head>
 <title>Este texto es el título del documento</title>
</head>
<body>
 <div id=”principal”>
 <p onclick=”alert(‘hizo clic!’)”>Hacer Clic</p>
 <p>No puede hacer clic</p>
 </div>
</body>
</html>
```

En el código expuesto cada vez que presionemos sobre el párrafo se mostrara un mensaje con el texto hizo clic, caso opuesto al que no tiene el atributo onClick ya que no hará nada

Embebido

Si nos vemos en la necesidad de declarar funciones o trabajar con código más extenso una de las opciones que tenemos es hacer uso de la etiqueta script, esta actua exactamente como style y podemos escribir código js dentro de sus etiquetas

```
<!DOCTYPE html>
<html lang="es">
<head>
 <title>Este texto es el título del documento</title>
 <script>
 function mostraralerta(){
 alert('hizo clic!');
 }
 function hacerclic(){
 document.getElementsByTagName('p')[0].onclick=mostraralerta;
 }
 window.onload=hacerclic;
 </script>
</head>
```

```
<body>
 <div id=”principal”>
 <p>Hacer Clic</p>
 <p>No puede hacer Clic</p>
 </div>
 </body>
</html>
```

Cabe recalcar que en este caso sí importa la ubicación de la etiqueta script, si la ubicamos antes del cuerpo de la página esta se cargara primero y algunas funciones pueden no ejecutarse, pero si la ponemos al final del cuerpo la página cargara primero y de ahí el código js.

En este código de ejemplo se hizo uso de la función getElementsByTagName() esto se usa para referenciar elementos al igual que CSS hay algunas maneras para hacerlo, estas son:

- **getElementsByTagName:** Hace referencia a un elemento por su nombre de etiqueta.
- **getElementById:** Permite referenciar a un elemento por el valor de su atributo id.
- **getElementsByClassName:** Nos permite referenciar un elemento por el valor de su atributo class.

Archivos Externos

Es completamente normal que el código se extienda por líneas y líneas y poner esto dentro de un solo documento seria incomodo, por eso se agrega archivos externos de la misma manera que con CSS

```
<!DOCTYPE html>
<html lang="es">
<head>
 <title>Este texto es el título del documento</title>
 <script src=”micodigo.js”></script>
</head>
<body>
 <div id=”principal”>
 <p>Hacer Clic</p>
 <p>No puede hacer Clic</p>
 </div>
```

```
</body>
</html>
```

Nuevo Selectores

Al igual que con CSS se han agregado nuevas maneras de hacer referencias a elementos HTML en este caso se presentan en la forma de:

querySelector()

Esta retorna el primer elemento que concuerda con el selector especificado en el ejemplo a continuación devolverá el primer hijo del elemento con el id principal

```
function hacerclic(){
 document.querySelector(“#principal p:first-
 child”).onclick=mostraralerta;
}
```

querySelectorAll()

Similar al anterior, pero devuelve todos los elementos que concuerden con el selector especificado, este es devuelto como un arreglo asi que tenemos que especificar si queremos trabajar con uno en particular, como en el ejemplo trabajamos con el primer elemento de ese arreglo

```
function hacerclic(){
 var lista=document.querySelectorAll(“#principal p”);
 lista[0].onclick=mostraralerta;
}
```

Manejadores de eventos

Como se explicó con anterioridad el código js se ejecuta después de que el usuario ejecuta un acción en particular, para poder realizar esto necesitamos del uso de manejadores de eventos, tenemos tres manera de hacerlo en línea, como propiedades y con addEventListener(), las dos primeras formas ya las vimos siendo en línea la propiedad onClick y las propiedades los getElements así que nos centraremos en la última.

addEventListener()

Es considerado como la mejor forma para manejar eventos, este método hace uso de tres argumentos primero el nombre del evento, la función a ser ejecutado y un valor booleano para indicar como un evento será disparado cuando se superpongan elementos. Gracias a esto podremos ejecutar los eventos que necesitemos en nuestro sitio web

```
<!DOCTYPE html>
<html lang="es">
<head>
 <title>Este texto es el título del documento</title>
 <script>
 function mostraralerta(){
 alert('hizo clic!');
 }
 function hacerclic(){
 var elemento=document.getElementsByTagName('p')[0];
 elemento.addEventListener(‘click’, mostraralerta, false);
 }
 window.addEventListener(‘load’, hacerclic, false);
 </script>
</head>
<body>
 <div id=”principal”>
 <p>Hacer Clic</p>
 <p>No puede hacer Clic</p>
 </div>
</body>
</html>
```

Uso de Librerías

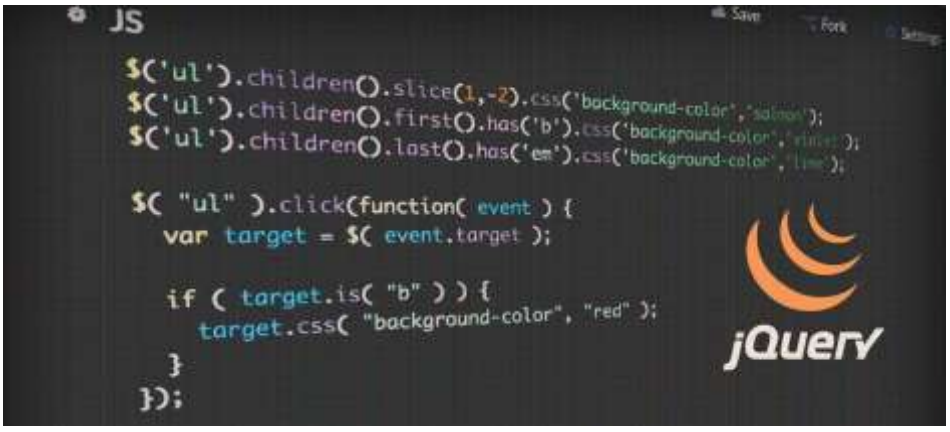
Una librería en el ámbito de programación es un conjunto de archivos desarrollados en un lenguaje determinado que cuentan con funciones que agilizan el desarrollo de un programa o sitio web, estas tienen un propósito específico por esto los métodos que cuentan son especializados para cumplir dicho propósito, en el desarrollo web tenemos algunas librerías de uso común como:

jQuery

jQuery es la más utilizada de todas las bibliotecas de JavaScript. Entre otras razones, esto se debe a que permite escribir código jQuery en cualquier tipo de navegador, y hay muchos complementos para eso. Las bibliotecas de código abierto se incluyen en muchos sistemas de gestión de contenido, como WordPress, Drupal o Joomla!. jQuery se usa principalmente como una interfaz DOM conveniente y ofrece muchas funciones: los selectores CSS3 le permiten seleccionar y manipular fácilmente elementos en una página web, pero jQuery es especialmente popular debido a su capacidad para integrar consultas Ajax con HTTP (sin necesidad de recargar la página web).

Figura 63

jQuery



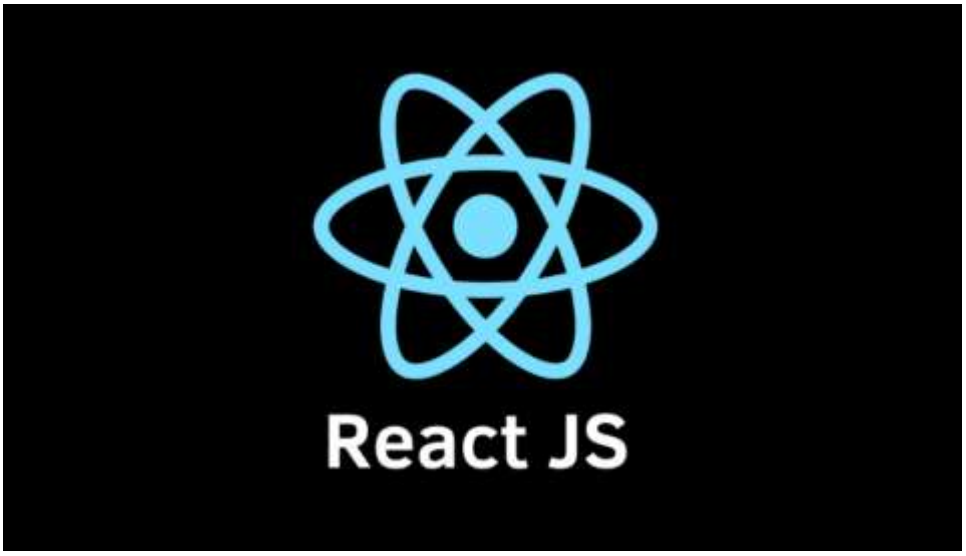
Nota. Tomada de *Ayudawp* [Imagen], Ayudawp, 2022 (https://ayudawp.com/wp-content/uploads/2009/03/codigo-jquery.jpg). CC BY 2.0

React

React se utilizó por primera vez en Facebook News Feed en 2011. En 2013, se lanzó como una biblioteca JavaScript de código abierto para crear interfaces de usuario. Lo que hace que React sea especial es que puede usarse no solo en el lado del cliente web, sino también en el lado del servidor o durante el desarrollo de aplicaciones. Esto es gracias al uso de DOM virtual, que también facilita el análisis de aplicaciones web. Asimismo, esta biblioteca de JavaScript destaca por su flujo de datos unidireccional esta técnica proporciona un código estable donde los cambios en el código de nivel inferior no afectan al código de nivel superior. Por lo tanto, los cambios solo pueden tener efectos en la dirección opuesta.

Figura 64

React



Nota. Tomada de *Sigdeletras* [Imagen], *Sigdeletras*, 2022 (https://sigdeletras.com/images/blog/202004_react_leaflet/react.png). CC BY 2.0

Uso de Frameworks

Framework es la palabra en inglés para estructura o marco de trabajo, en el ámbito de la programación un framework ofrece soporte para la elaboración de un proyecto una especie de modelo que servirá como punto de inicio para el desarrollo del software, son de gran utilidad para realizar aplicaciones web de gran complejidad, dentro de los más populares tenemos:

AngularJS

Este framework desarrollado por Google es sin duda alguna el más popular entre sus pares. es utilizado para crear aplicaciones web de una sola página (aplicaciones web que constan de un solo documento HTML), al igual hace su principal competidor, este es, la biblioteca de Facebook React. A causa del patrón MVVM (Model-View-ViewModel), se pueden desarrollar aplicaciones web especializadas en la interacción con los usuarios.

Figura 65

AngularJS

Nota. Tomada de *Okhosting* [Imagen], *Okhosting*, 2022 (https://okhosting.com/wp-content/uploads/2015/08/angularjs.jpg). CC BY 2.0

Angular

Angular, también conocido comúnmente como Angular 2, es el sucesor de AngularJS. Está orientado principalmente al desarrollo de aplicaciones web de una sola página, aunque Google ha realizado algunos cambios muy importantes en la segunda versión. En este sentido, la gran diferencia es que para programar ya no se usa JavaScript, sino TypeScript. Dado que el lenguaje de secuencias de comandos de Microsoft se basa en JavaScript y lo admite, no existen restricciones inmediatas en el desarrollo de JS. Además, Angular se adapta de tal manera que desarrollar aplicaciones en varias plataformas (escritorio, móvil, tableta) no es un problema.

Figura 66

Angular

Nota. Tomada de *SG* [Imagen], *SG*, 2022 (https://sg.com.mx/sites/default/files/styles/480x319/public/images/angular-logo.png?itok=k-dyLJ_m). CC BY 2.0

Vue.js

Vue.js es de la misma manera un framework de JavaScript enfocado en el desarrollo de aplicaciones web de una sola página que recuerda a Angular y React. Los desarrolladores de este proyecto relativamente nuevo han diseñado Vue.js de tal manera que iniciarse con él sea

relativamente sencillo. Así es posible, por ejemplo, integrar plantillas en HTML. Asimismo, Vue.js es mucho más flexible que otros frameworks.

Figura 67

Vue Js

Nota. Tomada de *VictorRoblesWeb* [Imagen], *VictorRobles*, 2022 (https://victorroblesweb.es/wp-content/uploads/2017/03/vuejs2-victorroblesweb.jpg). CC BY 2.0

API Rest

Una API es un conjunto de protocolos con la finalidad de permitir la comunicación entre dos sistemas, permitiendo de esta manera que un sitio o aplicación web cuente con varias funciones, Rest es una arquitectura que hace el uso de hipermedios como json, xml para la transmisión de datos es gracias a esto que cuenta con gran aceptación ya que esta característica le permite tener con una gran escalabilidad al enviar la información por estos medios y no alojarla en el servidor.

Figura 68

Ejemplo de Arquitectura Rest

Nota. Tomada de *GaussWeb* [Imagen], *GaussWeb*, 2022 (http://gausswebapp.com/wp-content/uploads/2014/09/imagen_rest.png). CC BY 2.0

Para que una Api pueda considerarse Restful debe cumplir con las siguientes condiciones:

- Arquitectura cliente-servidor compuesta de clientes, servidores y recursos, con la gestión de solicitudes a través de HTTP.
- Comunicación entre el cliente y el servidor sin estado, lo cual implica que no se almacena la información del cliente entre las solicitudes de GET y que cada una de ellas es independiente y está desconectada del resto.
- Datos que pueden almacenarse en caché y optimizan las interacciones entre el cliente y el servidor.
- Una interfaz uniforme entre los elementos, para que la información se transfiera de forma estandarizada. Para ello deben cumplirse las siguientes condiciones:
  - Los recursos solicitados deben ser identificables e independientes de las representaciones enviadas al cliente.
  - El cliente debe poder manipular los recursos a través de la representación que recibe, ya que esta contiene suficiente información para permitirlo.
  - Los mensajes autodescriptivos que se envíen al cliente deben contener la información necesaria para describir cómo debe procesarla.
  - Debe contener hipertexto o hipermedios, lo cual significa que cuando el cliente acceda a algún recurso, debe poder utilizar hipervínculos para buscar las demás acciones que se encuentren disponibles en ese momento.
- Un sistema en capas que organiza en jerarquías invisibles para el cliente cada uno de los servidores (los encargados de la seguridad, del equilibrio de carga, etc.) que participan en la recuperación de la información solicitada.
- Código disponible según se solicite (opcional), es decir, la capacidad para enviar códigos ejecutables del servidor al cliente cuando se requiera, lo cual amplía las funciones del cliente.

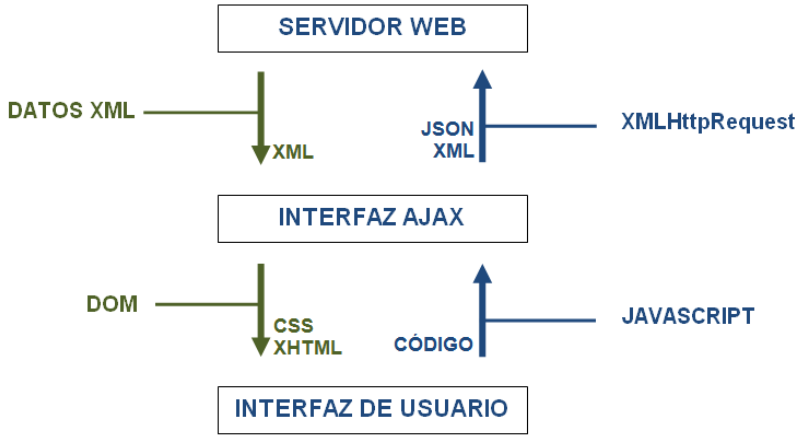
AJAX

Ajax es un acrónimo para Asynchronous JavaScript and XML en español es JavaScript Asincrónico y XML y son técnicas que permiten que la aplicación web funcione de manera asíncrona, relegando todas las solicitudes hacia el servidor al segundo plano. Un ejemplo sencillo de esto es la función de autocompletado de google mientras vamos escribiendo las sugerencias se van adaptando en tiempo real pero la página no cambia, para que Ajax funcione se necesita:

- **HTML/XHTML:** como el lenguaje principal
- **DOM**
- **XML:** para intercambiar datos
- **XMLHttpRequest**
- **Javascript**

Figura 69

Esquema de Ajax



Nota. Tomada de *Laredinfinita* [Imagen], *La red infinita*, 2022 (https://laredinfinita.files.wordpress.com/2014/04/estructura-ajax.pngCC BY 2.0

Json

Son las siglas en inglés para JavaScript Object Notation o en español Notación de Objetos de JavaScript es un formato ligero para el intercambio de datos, fácil de leer tanto para los

programadores como para las maquinas, Json hace uso de convenciones que los programadores están más que acostumbrados por eso lo hace ideal para compartir datos mediante una API Rest gracias a su ligereza y flexibilidad, como vemos en los ejemplos a continuación hay varias formas de escribir los datos.

```
{
   "rojo":"#f00",
   "verde":"#0f0",
   "azul":"#00f",
   "cyan":"#0ff",
   "magenta":"#f0f",
   "amarillo":"#ff0",
   "negro":"#000"
}
```

```
{
   "arrayColores":[{
         "rojo":"#f00",
         "verde":"#0f0",
         "azul":"#00f",
         "cyan":"#0ff",
         "magenta":"#f0f",
         "amarillo":"#ff0",
         "negro":"#000"
      }
   ]
}
```

Dentro de las características de JSON tenemos:

- JSON es solo un formato de datos.
- Requiere usar comillas dobles para las cadenas y los nombres de propiedades. Las comillas simples no son válidas.
- Una coma o dos puntos mal ubicados pueden producir que un archivo JSON no funcione.

- Puede tomar la forma de cualquier tipo de datos que sea válido para ser incluido en un JSON, no solo arreglos u objetos. Así, por ejemplo, una cadena o un número único podrían ser objetos JSON válidos.
- A diferencia del código JavaScript, en el que las propiedades del objeto pueden no estar entre comillas, en JSON solo las cadenas entre comillas pueden ser utilizadas como propiedades.

Websockets

WebSocket es un protocolo de red basado en TCP que especifica cómo se deben intercambiar los datos entre redes. Dado que es un protocolo confiable y eficiente, es utilizado por casi todos los clientes. El protocolo TCP establece una conexión, llamada socket, entre dos puntos finales de comunicación, de esta manera, el intercambio de datos puede tener lugar en ambas direcciones. En una conexión bidireccional, como la creada por WebSocket (y en ocasiones también websocket o web socket), los datos se intercambian en ambas direcciones al mismo tiempo. El beneficio de este intercambio es un acceso más rápido a los datos. Específicamente, WebSocket permite la comunicación directa entre una aplicación web y un servidor WebSocket. En otras palabras: el sitio web solicitado se muestra en tiempo real.

CAPÍTULO IV: Desarrollo de aplicaciones

Aplicación de WebSocket con Arduino IDE

Uso de websockets usando el sensor dht11 y un led mediante el módulo esp32

El presente proyecto consiste en hacer uso de la tecnología web socket, en este caso para medir la temperatura y humedad que el sensor dht11 recoja y también identificar el estado del led que usaremos, mediante la acción que el cliente realice al usar la aplicación web.

Tabla 17

Código general del programa

Archivo principal (dh.ino)

```
1   #include <WiFi.h>
2   #include <WebServer.h>
3   #include <WebSocketsServer.h>
4   #include<DHT.h>
5   //-----------------------------------------------
6   DHT dht(4, DHT11);
7   //-----------------------------------------------
8   const char* ssid = "NAVARRETE";
9   const char* password = "baldeon8";
10  //-----------------------------------------------
11
12  WebServer server(80);
13
14  WebSocketsServer webSocket = WebSocketsServer(81);
15
16  //-----------------------------------------------
17
18  #define LED 2
19
20  boolean LEDonoff=false; String JSONtxt;
21
22  //-----------------------------------------------
23
```

```
#include "webpage.h"

#include "functions.h"

//=================================================
================
void setup()

{

  Serial.begin(115200);

  pinMode(LED, OUTPUT);

  dht.begin();

  //-----------------------------------------------

  WiFi.begin(ssid, password);

  while(WiFi.status() != WL_CONNECTED)

  {

    Serial.print("."); delay(500);

  }

  WiFi.mode(WIFI_STA);

  Serial.print(" Local IP: ");

  Serial.println(WiFi.localIP());

  //-----------------------------------------------

  server.on("/", handleRoot);

  server.on("/grid", validar);

```

```
  server.on("/luces", opcLuces);

            server.begin();          webSocket.begin();
webSocket.onEvent(webSocketEvent);
}

//=================================================
================
void loop()

{

  webSocket.loop(); server.handleClient();

  //-----------------------------------------------

  static unsigned long l = 0;

  unsigned long t = millis();

  if((t-l) > 1000)

  {

    if(LEDonoff == false) digitalWrite(LED, LOW);

    else digitalWrite(LED, HIGH);

    String LEDstatus = "OFF";

    if(LEDonoff == true) LEDstatus = "ON";

     //-----------------------------------------------

                  String        TEMPvalString        =
String(dht.readTemperature());
                   String        HUMvalString        =
String(int(dht.readHumidity()));
    //-----------------------------------------------
```

```
6
5         JSONtxt  = "{\"TEMP\":\""+TEMPvalString+"\",";
7
5         JSONtxt += "\"HUM\":\""+HUMvalString+"\",";
8
5         JSONtxt += "\"LEDonoff\":\""+LEDstatus+"\"}";
9
6         webSocket.broadcastTXT(JSONtxt);
0
6       }
1
6     }
2
```

El código expuesto en la Tabla 18 servirá para conectarnos a nuestra red Wifi:

Tabla 18

Código de credenciales de acceso a la red Wifi

Credenciales de acceso a la red Wifi	
1	`const char* ssid = "NAVARRETE";//creamos una constate del nombre del wifi al que nos conectaremos`
2	`const char* password = "baldeon8";//creamos una constante con la contraseña del wifi al que nos vamos a conectar`

Posteriormente, a partir del siguiente código que se muestra en la Tabla 19, la placa ESP 32 tratara de conectarse a la red wifi y si la conectividad es correcta nos mostrara la ip de nuestra placa para mostrar las páginas web.

Tabla 19

Código para tratar de conectar a la red Wifi

Código para la conexión a la red wifi	
1	`WiFi.begin(ssid, password);//tratara de conectarse a la red con el nombre y contraseña que le indicamos`
2	`while(WiFi.status() != WL_CONNECTED)//tratara de conectarse y mientras eso sucede va a mostrar un mensaje (la linea siguiente)`
3	`{`

```
4          Serial.print("."); delay(500);//mostrara el mensaje . cada 5 segundos (emula la accion de carga para que el usuario sepa que esta tratando de conectarse)
5        }
6        WiFi.mode(WIFI_STA);//muestra el estado de conexion con la red
7        Serial.print(" Local IP: ");//mostrara el mensaje indicado
8        Serial.println(WiFi.localIP());//mostrara la ip local asignada
```

A partir del código de la Tabla 20 se obtendrán la lectura de humedad y temperatura del sensor DHT11 y el estado del led.

Tabla 20

Código para la lectura y envío de datos de temperatura y humedad, y led.

Código de lectura y envió de datos de temperatura y humedad, y el led

```
1  String TEMPvalString = String(dht.readTemperature());//lee la temperatura que detecta el sensor y la guarda en la variable TEMPvalString
2                          String HUMvalString = String(int(dht.readHumidity()));//lee la humedad que detecta el sensor y la guarda en la variable HUMvalString
3      //-----------------------------------------------
4       /*Agregamos los valores ya extraidos de las variables anteriores y los mandamos a traves de la variable JSONtext para poder obtenerlos en las paginas web que las necesiten*/
5      JSONtxt  = "{\"TEMP\":\""+TEMPvalString+"\",";
6      JSONtxt += "\"HUM\":\""+HUMvalString+"\",";
7      JSONtxt += "\"LEDonoff\":\""+LEDstatus+"\"}";
8      webSocket.broadcastTXT(JSONtxt);
```

Es necesario crear un fichero llamado ‘functions.h’ que contendrá los métodos que se usaran para llamar a los métodos que permita redirigir a los usuarios hacia las páginas webs que correspondas.

Tabla 21

Código del archivo 'functions.h'

functions.h

```
1   void index(){
2     server.send(200,"text/html", InicioWeb);
3   }
4   void validar(){
5     server.send(200,"text/html", grid);
6   }
7   void handleRoot(){
8     index();
9   }
10  void opcLuces(){
11
12    server.send(200,"text/html", Mainweb);
13
14  }
15
16  void  webSocketEvent(uint8_t  num,  WStype_t  type,
17  uint8_t *payload, size_t welength){
18    String payloadString = (const char *)payload;
19
20    Serial.print("payloadString= ");
21
22    Serial.println(payloadString);
23
24
25
26    if(type == WStype_TEXT) //receive text from client
27
28    {
29
30      byte separator=payloadString.indexOf('=');
31
32                          String          var          =
33  payloadString.substring(0,separator);
34      Serial.print("var= ");
```

```
2
2         Serial.println(var);
3
2                         String          val          =
4     payloadString.substring(separator+1);
2         Serial.print("val= ");
5
2         Serial.println(val);
6
2         Serial.println(" ");
7
2
8
2         if(var == "LEDonoff")
9
3         {
0
3           LEDonoff = false;
1
3           if(val == "ON") LEDonoff = true;
2
3         }
3
3       }
4
3     }
5
```

El código de la Tabla 22 permite la recepción de los datos previamente enviados de temperatura, humedad y led.

Tabla 22

Código para la recepción de datos de temperatura, humedad y led.

	Código de la recepción de datos de temperatura, humedad y led
1	`void webSocketEvent(uint8_t num, WStype_t type, uint8_t *payload, size_t welength)`
2	`{`

```
3         String payloadString = (const char *)payload;
4         Serial.print("payloadString= ");
5         Serial.println(payloadString);//muestra un mensaje con el
      valor que contiene la variable payloadString
6
7         if(type == WStype_TEXT) // evalua si recibe un texto del
      cliente (nosotros)
8         {
9             byte separator=payloadString.indexOf('=');//obtiene el
      indice de donde se encuentre el objeto = y lo guarda en una
      variable llamada separator
10            String var = payloadString.substring(0,separator);//en
      la variable var se guardara el texto ya separado del
      payloadString
11            Serial.print("var=  ");//muestra un mensaje con la
      palabra var
12            Serial.println(var); //muestra lo que contiene la
      variable var
13            String val = payloadString.substring(separator+1);//va a
      separar
14            Serial.print("val= ");
15            Serial.println(val);//muestra lo que contiene la palbra
      val
16            Serial.println(" ");
17            if(var == "LEDonoff")//va a comparar si la variable var
      contiene el texto LEDonoff y si es asi ejecutara el codigo
      de adebajo
18            {
19              LEDonoff = false;//inicializa la variable LEDonOff en
      falso
20              if(val == "ON") LEDonoff = true; //evalua si la
      variable val dice on y en caso de ser verdad va a cambiar a
      true la variable LEDonoff
21            }
22        }
```

Por último, es importante crear un archivo llamado webpage.h, la cual contendrá todo lo relacionado a las páginas web que se les mostrará a los usuarios.

Tabla 23

Código del archivo 'webpage.h'

webpage.h

```
1   //=====================
2   //HTML code for webpage
3   //=====================
4   //----------------------------
5   const char InicioWeb[] PROGMEM =
6   R"=====(
7   <!DOCTYPE html>
8   <html>
9   <head>
10    <title></title>
11
12        <link rel="stylesheet" type="text/css"
    href="https://emerkaskadldak.000webhostapp.com/agama
    /style.css">
13
14
15  </head>
16
17  <body>
18
19  <!-- Form-->
20
21  <div class="form">
22
23    <div class="form-toggle"></div>
24
25    <div class="form-panel one">
26
27      <div class="form-header">
28
29        <h1>Inicio de sesion</h1>
```

```
        </div>

        <div class="form-content">


            <div class="form-group">

              <label for="username">Usuario</label>

                    <input  type="text"  id="email_field"
name="username" required="required"/>
            </div>

            <div class="form-group">

              <label for="password">Clave</label>

               <input type="password" id="password_field"
name="password" required="required"/>
            </div>

            <div class="form-group">

                      <button   onclick="login()">Iniciar
sesion</button>
            </div>


        </div>

      </div>


    </div>

    <script
```

```
src="https://www.gstatic.com/firebasejs/4.8.1/firebase.js"></script>


  <script>

    // Initialize Firebase

    var config = {

                              apiKey:            "AIzaSyD-7t5dXC7rsT8ZIISeh0v9xH8EQ1etLUQ",
    authDomain: "pruebafire-aaf01.firebaseapp.com",

     databaseURL:  "https://pruebafire-aaf01-default-rtdb.firebaseio.com",
    projectId: "pruebafire-aaf01",

    storageBucket: "pruebafire-aaf01.appspot.com",

    messagingSenderId: "83050799887",

                                                  appId: "1:83050799887:web:9f8ccca8a655583829547a"
    };

    firebase.initializeApp(config);

  </script>


                                                <script src="https://emerkaskadldak.000webhostapp.com/agama/index.js"></script>
</body>

</html>

```

```
    )=====";
    //--------------
```

Tabla 24

Código de JavaScript sobre el funcionamiento de las lecturas de datos del sensor DHT11 y el led

Código JavaScript para la muestra del funcionamiento de las lecturas de datos de temperatura, humedad y led.

```
<!----------------------JavaScript---------------------->
<script>
  InitWebSocket()//se llama a la funcion que inicia el web socket
  function InitWebSocket()
  {
                              websock = new WebSocket('ws://'+window.location.hostname+':81/');  //se le asigna la ruta en donde estara escuchando el web socket
     websock.onmessage=function(evt)
     {

         JSONobj = JSON.parse(evt.data);//convierte los datos recibidos en archivo json para ser accedidos facilmente
                document.getElementById('btn').innerHTML = JSONobj.LEDonoff;//colocamos en el boton llamado btn lo que traiga el valor LEDonOff (en este caso mostrara ON u Off dependiendo de lo que traiga)
              if(JSONobj.LEDonoff == 'ON')//si la variable LEDonoff contiene el valor ON, se aplicaran unos estilos para indicar que esta encendido, caso contrario aplicara otros estilos respectivos
           {

document.getElementById('btn').style.background='#FF0000';
              document.getElementById('btn').style["boxShadow"] = "0px 0px 0px 8px #FF0000";
           }
```

```
17                 else
18                 {
19      document.getElementById('btn').style.background='#111111';
                    document.getElementById('btn').style["boxShadow"]
20      = "0px 0px 0px 8px #111111";
21                 }
                    document.getElementById('Tempvalue').innerHTML   =
                JSONobj.TEMP;//en el objeto llamado Tempvalue se le asignara
22      la variable TEMP que habiamos visto anteriormente
23
                   document.getElementById('Tempvalue2').innerHTML   =
                JSONobj.TEMP;//en el objeto llamado Tempvalue2 se le asignara
24      la variable TEMP que habiamos visto anteriormente
               var temp = parseInt(JSONobj.TEMP * 9.5); //se realiza
                una operacion a partir de los valores obtenidos de la
25      temperatura
               document.getElementById("dynRectangle1").style.width =
                temp+"px";//se obtiene el objeto dynRectangle1 para asignarle
26      un tamaño a la barra que se mostrara en el sitio
27
                    document.getElementById('Humvalue').innerHTML   =
                JSONobj.HUM;//igual que en el caso anterior, se haran
                referencia a los objetos Humvalue y Humvalue dos para
28      agregarles las variables
                    document.getElementById('Humvalue2').innerHTML   =
                JSONobj.HUM;//que hacen referencia a la humedad que el sensor
                detecta por medio de las variables que habiamos enviado
29      anteriormente
               var hum = parseInt(JSONobj.HUM * 4.8);//se realiza una
30      operacion a partir de los valores obtenidos de la humedad
               document.getElementById("dynRectangle2").style.width =
                hum+"px"; //se obtiene el objeto dynRectangle2 para asignarle
31      un tamaño a la barra que se mostrara en el sitio
32
33                    document.getElementById('FANbtn').innerHTML    =
```

```
        JSONobj.LEDonoff;//en el boton llamado FANbtn se mostrara un
        texto con el valor que contenga LEDonoff (se mostrara on u
        off)
34          }
35        }
36        //-----------------------------------------------------
37        function fanONOFF()// se crea una funcion fanONOFF
38        {
              FANbtn = 'LEDonoff=ON'; //se crea una variable FANbtn
39      establecimiendo un texto
               if(document.getElementById('btn').innerHTML  ==  'ON')
        //seleccionamos el objeto llamado btn y le comparamos si su
40      texto es ON, si es asi la variable FANbtn cambiara
41          {
42            FANbtn = 'LEDonoff=OFF';
43
44          }
              websock.send(FANbtn);//y  posteriormete  se  enviara  a
45      traves del web socket dicha variable FANbtn
46        }
47      </script>
```

Front End de la aplicación

Para acceder al sitio es necesario colocar la ip que la placa nos asigne, en este caso será la ip 192.168.100.137, y se nos mostrará la página web principal del proyecto (Figura 70), la cual será un formulario de inicio de sesión, en donde cada usuario tendrá que introducir correctamente sus credenciales de acceso para poder acceder.

Figura 70

Pantalla de inicio de sesión

Tabla 25

Código de la página de inicio de sesión

	Pagina de inicio de sesión del programa
1	//======================
2	//HTML code for webpage
3	//======================
4	//---------------------------
5	const char InicioWeb[] PROGMEM =
6	R"=====(
7	<!DOCTYPE html>
8	<html>
9	<head>
10	<title></title>
11	<link rel="stylesheet" type="text/css"

```
        href="https://emerkaskadldak.000webhostapp.com/agama
        /style.css">
1
2
1   </head>
3
1   <body>
4
1   <!-- Form-->
5
1   <div class="form">
6
1     <div class="form-toggle"></div>
7
1     <div class="form-panel one">
8
1       <div class="form-header">
9
2         <h1>Inicio de sesion</h1>
0
2       </div>
1
2       <div class="form-content">
2
2
3
2           <div class="form-group">
4
2             <label for="username">Usuario</label>
5
2                 <input  type="text"  id="email_field"
6   name="username" required="required"/>
2         </div>
7
2         <div class="form-group">
8
2           <label for="password">Clave</label>
9
3            <input type="password" id="password_field"
0   name="password" required="required"/>
```

```
            </div>

            <div class="form-group">

                        <button   onclick="login()">Iniciar sesion</button>
            </div>


        </div>

      </div>


</div>

<script src="https://www.gstatic.com/firebasejs/4.8.1/firebase.js"></script>


  <script>

    // Initialize Firebase

    var config = {

                              apiKey:           "AIzaSyD-7t5dXC7rsT8ZIISeh0v9xH8EQ1etLUQ",
    authDomain: "pruebafire-aaf01.firebaseapp.com",

     databaseURL:  "https://pruebafire-aaf01-default-rtdb.firebaseio.com",
    projectId: "pruebafire-aaf01",

    storageBucket: "pruebafire-aaf01.appspot.com",
```

```
        messagingSenderId: "83050799887",

                                                    appId:
"1:83050799887:web:9f8ccca8a655583829547a"
        };

        firebase.initializeApp(config);

    </script>


                                                <script
src="https://emerkaskadldak.000webhostapp.com/agama/
index.js"></script>
</body>

</html>

)=====";

//--------------
```

Si el usuario que está intentando ingresar al sitio coloca sus credenciales de la manera correcta en el formulario anterior, el sitio lo enviara a la siguiente página (Figura 71), la cual es el menú de opciones del sitio, para poder elegir hacía que opciones de la casa desea continuar, para poder controlarla.

Figura 71

Pantalla de menú de opciones

Tabla 26

Código de la página de menú de opciones

	Página de menú de opciones
1	const char grid[] PROGMEM =
2	R"=====(
3	<!DOCTYPE html>
4	<html lang="es">
5	<head>
6	<meta charset="UTF-8">
7	<meta name="viewport" content="width=device-width, initial-scale=1.0">

```
        <title>Document</title>
        <style>
            // font stuff

@import url(https://fonts.googleapis.com/css?family=Source+Sans+Pro:400,200,300,600,700,900);
// colour stuff

@white: #ffffff;

@lightBG: #dce1df;

@salmon: #ff6666;

@teal: #0096a0;

@tealMid: #0ebac7;

@tealContrast: #33ffff;

@tealShade: #007c85;

@darkGrey: #4f585e;



        </style>

            <link rel="stylesheet" type="text/css" href="https://emerkaskadldak.000webhostapp.com/grid_folder/grid.css">
</head>

<body>

<div class="cards">

```

```
28      <h1 class="centrar">Casa Inteligente</h1>
29
30      <div class="card">
31
32          <div class="card-title">
33
34            <h2 class="centrar">Entretenimiento</h2>
35
36          </div>
37
38          <div class="card__image-holder">
39
40              <form action="./luces" method="post">
41
42                                          <input        type="image"
    src="https://artecconsultants.com/wp-
    content/uploads/2020/12/bolas-de-discoteca-1024x682.jpg"
     alt="discoteca" width="100%"/>
43              </form>
44
45          </div>
46
47        </div>
48
49      <div class="card">
50
51          <div class="card-title">
52
53            <h2 class="centrar">Entretenimiento</h2>
54
55          </div>
56
57          <div class="card__image-holder">
58
59                                    <img         class="card__image"
60  src="https://artecconsultants.com/wp-
    content/uploads/2020/12/bolas-de-discoteca-1024x682.jpg"
    alt="discoteca" width="100%" />
61          </div>
62
```

```
46        </div>
47
48        <div class="card">
49
50          <div class="card-title">
51
52            <h2 class="centrar">Entretenimiento</h2>
53
54          </div>
55
56          <div class="card__image-holder">
57
58                                    <img         class="card__image"
    src="https://artecconsultants.com/wp-
    content/uploads/2020/12/bolas-de-discoteca-1024x682.jpg"
    alt="discoteca" width="100%" />
59          </div>
60
61
62
63        </div>
64
65        <div class="card">
66
67          <div class="card-title">
68
69            <h2 class="centrar">Entretenimiento</h2>
70
71          </div>
72
73          <div class="card__image-holder">
74
75                                    <img         class="card__image"
    src="https://artecconsultants.com/wp-
    content/uploads/2020/12/bolas-de-discoteca-1024x682.jpg"
    alt="discoteca" width="100%" />
76          </div>
77
78
79
```

```
      </div>

      <div class="card">

        <div class="card-title">

          <h2 class="centrar">Entretenimiento</h2>

        </div>

        <div class="card__image-holder">

                                  <img         class="card__image"
src="https://artecconsultants.com/wp-
content/uploads/2020/12/bolas-de-discoteca-1024x682.jpg"
alt="discoteca" width="100%" />
        </div>



      </div>

      <div class="card">

        <div class="card-title">

          <h2 class="centrar">Entretenimiento</h2>

        </div>

        <div class="card__image-holder">

                                  <img         class="card__image"
src="https://artecconsultants.com/wp-
content/uploads/2020/12/bolas-de-discoteca-1024x682.jpg"
alt="discoteca" width="100%" />
        </div>


```

```
      </div>

      <div class="card">

        <div class="card-title">

          <h2 class="centrar">Entretenimiento</h2>

        </div>

        <div class="card__image-holder">

                                  <img        class="card__image"
src="https://artecconsultants.com/wp-
content/uploads/2020/12/bolas-de-discoteca-1024x682.jpg"
alt="discoteca" width="100%" />
        </div>



      </div>

      <div class="card">

        <div class="card-title">

          <h2 class="centrar">Entretenimiento</h2>

        </div>

        <div class="card__image-holder">

                                  <img        class="card__image"
src="https://artecconsultants.com/wp-
content/uploads/2020/12/bolas-de-discoteca-1024x682.jpg"
alt="discoteca" width="100%" />
        </div>


```

```
      </div>

      <div class="card">

        <div class="card-title">

          <h2 class="centrar">Entretenimiento</h2>

        </div>

        <div class="card__image-holder">

                                  <img        class="card__image"
src="https://artecconsultants.com/wp-
content/uploads/2020/12/bolas-de-discoteca-1024x682.jpg"
alt="discoteca" width="100%" />
        </div>


      </div>

    </div>

    </body>

    </html>
```

```
1       )=====";
1
3
```

Posteriormente a que el usuario escoja una opción en el menú anterior, el sitio lo redirigirá hacia la página, en la cual se mostraran los valores censados por el DHT11 y el usuario podrá mediante un botón elegir si desea encender o apagar el led, como se muestra en la Figura 72.

Figura *72*

Pantalla de control de aparatos del hogar

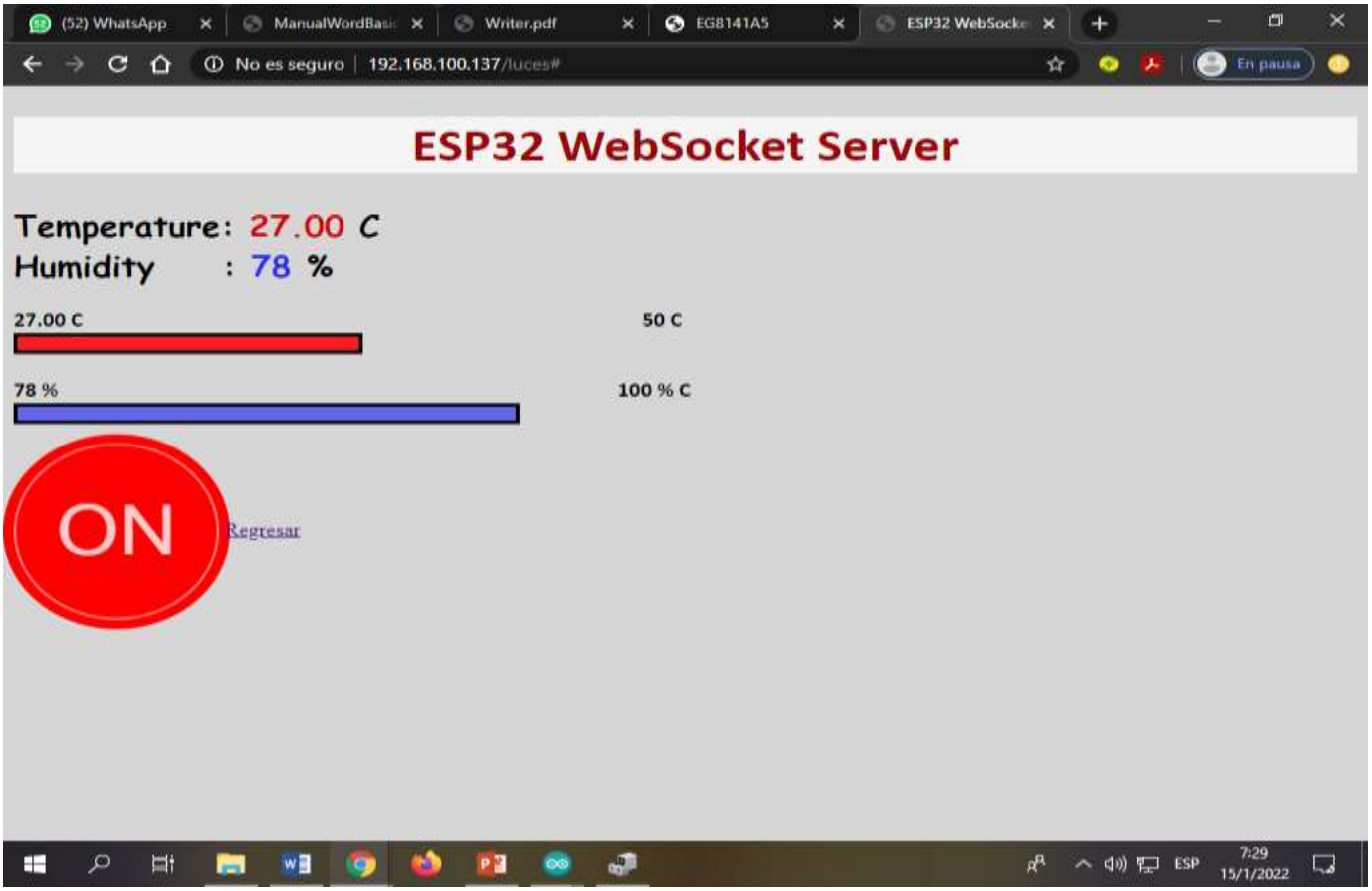


Tabla 27

Código de la página de control de aparatos del hogar

	Página de control de aparatos del hogar
1	//----------------------------
2	const char Mainweb[] PROGMEM =
3	R"=====(
4	<!DOCTYPE HTML>
5	<html>
6	<title>ESP32 WebSocket Server</title>
7	<!----------------------------CSS---------------------------->
8	<link rel="stylesheet" type="text/css" href="https://emerkaskadldak.000webhostapp.com/grid_folder/index.css">
9	<!----------------------------HTML---------------------------->
10	<body>
11	ESP32 WebSocket Server
12	<h2>
13	Temperature: 0 C
14	Humidity : 0 %
15	</h2>
16	<h3>
17	0 C
18	50 C
19	

```
            <div id="dynRectangle1"></div><br>
                                <span    id="Humvalue2">0</span>     %
            
  

            
   100 % C
            <div id="dynRectangle2"></div>

        </h3>

        <a href="#" id="btn" ONCLICK='fanONOFF()'> </a>

        <a href="./" >Regresar</a>

    </body>

    <!----------------------JavaScript------------------------>

    <script>

      InitWebSocket()

      function InitWebSocket()

      {

                                  websock             =             new
    WebSocket('ws://'+window.location.hostname+':81/');
        websock.onmessage=function(evt){

           JSONobj = JSON.parse(evt.data);

                    document.getElementById('btn').innerHTML    =
    JSONobj.LEDonoff;
              if(JSONobj.LEDonoff == 'ON'){

```

```
        document.getElementById('btn').style.background='#FF0000';

        document.getElementById('btn').style["boxShadow"] = "0px 0px 0px 8px #FF0000";
                }

                else{

        document.getElementById('btn').style.background='#111111';

        document.getElementById('btn').style["boxShadow"] = "0px 0px 0px 8px #111111";
                }

                 document.getElementById('Tempvalue').innerHTML   = JSONobj.TEMP;
                document.getElementById('Tempvalue2').innerHTML   = JSONobj.TEMP;
               var temp = parseInt(JSONobj.TEMP * 9.5);

                document.getElementById("dynRectangle1").style.width = temp+"px";
                 document.getElementById('Humvalue').innerHTML   = JSONobj.HUM;
                 document.getElementById('Humvalue2').innerHTML   = JSONobj.HUM;
               var hum = parseInt(JSONobj.HUM * 4.8);

                document.getElementById("dynRectangle2").style.width = hum+"px";
                    document.getElementById('FANbtn').innerHTML   = JSONobj.LEDonoff;
            }

          }

          //-----------------------------------------------------
```

```
6
5         function fanONOFF(){
7
5           FANbtn = 'LEDonoff=ON';
8
5           if(document.getElementById('btn').innerHTML == 'ON')  {
9
6             FANbtn = 'LEDonoff=OFF';
0
6           }
1
6           websock.send(FANbtn);
2
6         }
3
6       </script>
4
6       </html>
5
6       )=====";
6
```

Aplicación de WebSocket con el Framework VUE.JS

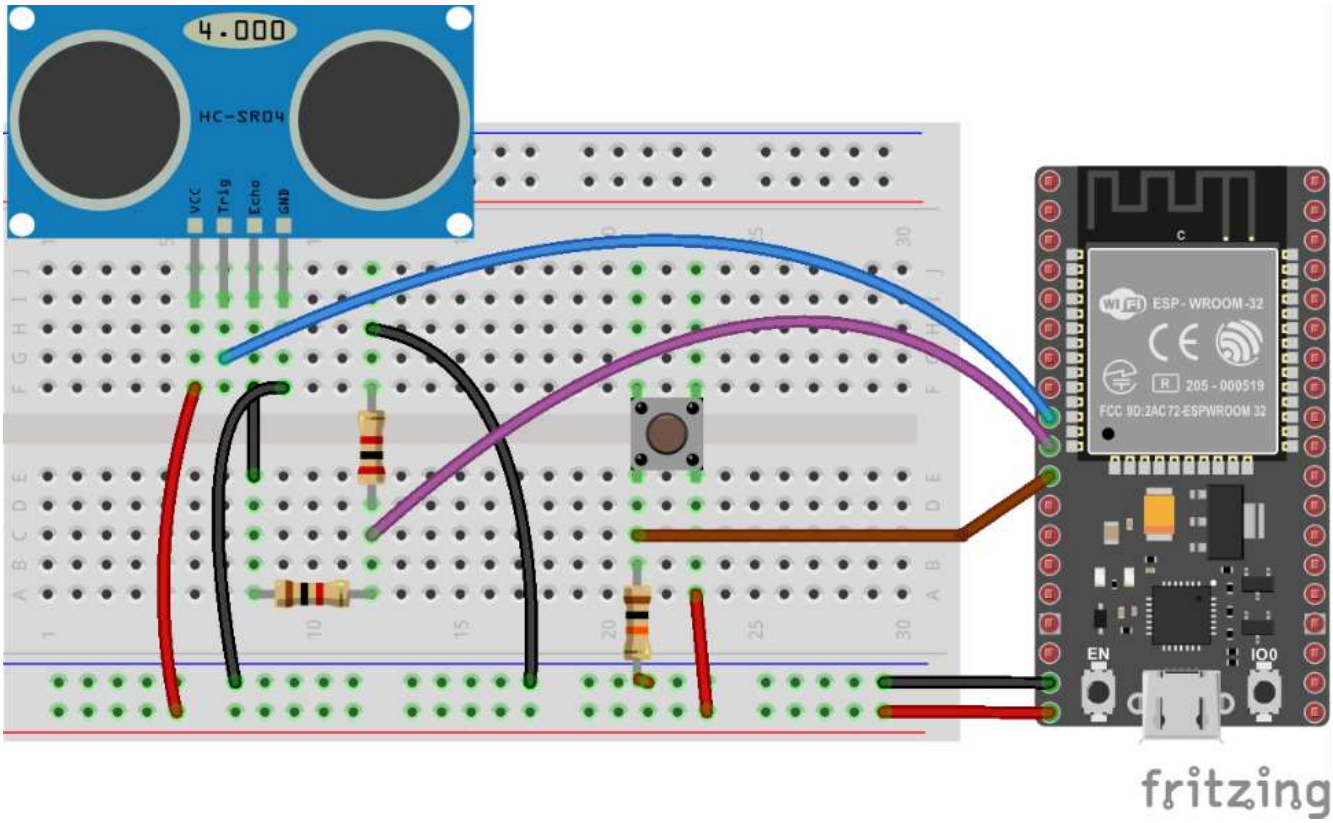
En el siguiente proyecto consiste en poder medir la distancia a través de un sensor ultrasónico conectado a un esp32 que enviara mensajes de la distancia censada por medio de websockets a nuestro cliente, además conectamos un pulsador físico para encender y apagar el led integrado que cuenta nuestro esp32.

Tal y como se observa en la figura 73, podemos encontrar el esquema de conexión del sensor HC-SR04 a nuestro esp32 cabe resaltar que el sensor se le conecto un divisor de voltaje debido que el pin de comunicación del sensor trabaja a 5v y el esp32 solo funciona a 3.3v, esto podría quemar nuestra placa y es por eso que es necesario agregar un divisor de voltaje que consiste en conectar dos resistencias en serie, de esta manera podemos obtener el voltaje equivalente a 3.3v con la cual trabaja el esp32.

A demás se encuentra conectado un pulsador que está en modo PULL-DOWN que nos ayudara en nuestra practica el encender y apagar el led integrado del esp32.

Figura 73

Esquema de Conexión del Sensor HC-SR04 al ESP32.



Desarrollo del Backend

Tabla 28

Instalaciones necesarias en el Backend

Librería	***Enlace de Descarga***	***Descripción***
ESPAsyncWebServer	https://github.com/me-no-dev/ESPAsyncWebServer	"Librería Open Source podemos crear un servidor asíncrono, es decir, que es capaz de atender a varios clientes de forma simultánea." [a]
ESPAsyncTCP	https://github.com/me-no-	Esta es una biblioteca TCP

	dev/ESPAsyncTCP	totalmente asíncrona, destinada a permitir un entorno de red multiconexión sin problemas que es la base para la implementación de la librería ESPAsyncWebServer.

Nota. La tabla muestra aquellas librerías que ayudaran a montar el web Server y hacer uso de websockets.

Esta tabla ha sido adaptada de “Cómo hacer un servidor asíncrono en ESP8266 o ESP32”, por Luis Lamas, 2022 (https://www.luisllamas.es/como-hacer-un-servidor-asincrono-en-esp8266/).

Para que el código sea más fácil y cómodo de leer se lo ha divido en varios ficheros según su funcionalidad, en el fichero principal tenemos el siguiente contenido como se logra apreciar en la tabla 29.

Tabla 29

Sketch Principal sobre el Código en Arduino

	ESP32_SENSORULTRASONICO.ino
1	//Librerias necesarias
2	#include <**WiFi**.h>
3	#include <ESPAsyncWebServer.h>
4	#include <SPIFFS.h>
5	
6	//Incluimos los demás ficheros de nuestra aplicacion
7	#include "config.h"
8	#include "WebSockets.hpp"
9	#include "Server.hpp"
10	#include "ESP32_Utils.hpp"
11	#include "ESP32_Utils_AWS.hpp"
12	

```
13  void setup() {
14    //Inicializa el led integrado como salida
15    pinMode(LED_BUILTIN, OUTPUT);
16    // inicializa el puerto seria a 115200 baudios
17    Serial.begin (115200);
18    //Inicializa el SPIFFS para el manejo de archivos
19    SPIFFS.begin();
20    //Nos permite conectar a una red WIFI
21    ConnectWiFi_STA();
22    //Inicializa el servidor
23    InitServer();
24    //Inicializa el InitWebSockets
25    InitWebSockets();
26    pinMode(Pecho, INPUT); // define el pin 6 como entrada(echo)
27   pinMode(Ptrig, OUTPUT); //pin 7 como salida  (triger)
28  }
```

```
29
30  void loop(){
31    //lee el estado del conmutador en una variable local
32    buttonState = digitalRead(buttonPin);
33   //Si el interruptor es presionado enciendo o apaga el led
34    if (buttonState == HIGH) {
35      digitalWrite(LED_BUILTIN, HIGH);
36    } else {
37      digitalWrite(LED_BUILTIN, LOW);
38    }
39
40    digitalWrite(Ptrig, LOW);
42    delayMicroseconds(2);
42    digitalWrite(Ptrig, HIGH);
43    delayMicroseconds(10); //genera el pulso por 10ms
44    digitalWrite(Ptrig, LOW);
```

```
45
46        duracion = pulseIn(Pecho, HIGH);
47       // calcula la distancia en centimetros
48        distancia = (duracion/2) / 29;
49        //Se imprime en el monitor serial la distancia sensada
50        Serial.print(distancia);
51        Serial.println("cm");
52        // cada 400ms se imprime la distancia
53        ws.textAll(String(distancia));
54        delay(400);
55      }
```

Por otro lado, creamos un fichero 'config.h' que contendrá las inicializaciones de variables globales que será usado en el setup y loop de la aplicación.

Tabla 30

Código del Archivo config.ino

	config.ino
1	//Pin para la lectura del interruptor
2	const int buttonPin = 17;

```
3   //Variable usada para guardar el estado del pulsador
4   int buttonState = 0;
5
6   //Credenciales para conectar a la red WIFI
7   const char* ssid = "COLOCAR_SSID";
8   const char* password = "COLOCAR_PASSWORD";
9
10  //Pines para el echo y trigger del sensor
11  #define Pecho 18
12  #define Ptrig 19
13
14  long duracion, distancia;
```

También contamos con un fichero '*ESP32_UTILS.hpp'* en la cual contiene un método que sirve para poder conectarnos a una red WIFI, una vez que se logre conectar podemos obtener la IP asignada que se la utiliza para poder manejar la conexión de websocket y al colocar la IP en un navegador nuestro ESP32 servirá el sitio web de la aplicación.

A continuación, en la tabla 31 se muestra el código que se necesita para poder conectarnos a una red WIFI.

Tabla 31

Código del Archivo ESP32_UTILS.hpp

ESP32_UTILS.hpp

```
1   void ConnectWiFi_STA(){
2     Serial.println("");
3   //Se estable modo station
4     WiFi.mode(WIFI_STA);
5   //Se usa el nombre de red y contraseña para intentar conectarnos
6   WiFi.begin(ssid, password);
7   //Esperar a que nos conectemos
8   while (WiFi.status() != WL_CONNECTED)  {
9       delay(100);
```

```
        Serial.print('.');
    }

    //Se muestra la red y la direccion IP asignada
    Serial.println("");
    Serial.print("Iniciado STA:\t");
    Serial.println(ssid);
    Serial.print("IP address:\t");
    Serial.println(WiFi.localIP());
    }
```

Analizando ahora el fichero ‘Server.hpp’, en este está definido unos ruteos, al acceder a la ruta raíz ‘/’ se servirá el contenido web que se haya guardado en la memoria SPIFFS.

SPIFFS (SPI Flash File System) Es un sistema de archivos diseñado para funcionar en memorias flash conectadas por SPI en dispositivos embebidos y con escasa cantidad de RAM, como el ESP8266 y el ESP32. (LLamas, 2019), todos estos ficheros habitualmente se los guarda dentro de una carpeta llamada data.

Si no existe alguna ruta devolvemos un error de 404 not found al cliente.

Tabla 32

Código del Archivo Server.hpp

Server.hpp

```
1  AsyncWebServer server(80);
2
3  void InitServer(){
4  //cargamos la página de inicio
5  server.serveStatic("/", SPIFFS, "/").setDefaultFile("index.html");
6  //Este método se acciona cuando no encuentra una pagina
7  server.onNotFound([](AsyncWebServerRequest *request) {
8  request->send(404, "text/plain", "Not found");
9  });
10 //Inicializamos el server
11 server.begin();
12 Serial.println("HTTP server started");
13 }
```

En las siguientes instrucciones de código consiste en la configuración del Async Websocket que nos permite mantener una conexión entre el servidor y el cliente con una alta velocidad de refresco.

Básicamente, recibimos los eventos de Async Websocket y, cuando se recibe un paquete entero, lanzamos la función 'ProcessRequest()' que se encuentra en un fichero aparte para gestionar aquellos mensajes que realiza el cliente.

Tabla 33

Código del Archivo CONFIG_WEBSOCKET.hpp

	CONFIG_WEBSOCKET.hpp
1	void onWsEvent(AsyncWebSocket * server, AsyncWebSocketClient * client, AwsEventType type, void * arg, uint8_t *data, **size_t** len){
2	//Cliente Conectado
3	if(type == WS_EVT_CONNECT){
4	client->printf("Hello Client %u :)", client->id());
5	client->ping();
6	} else if(type == WS_EVT_DISCONNECT){
7	//Cliente Desconectado
8	**Serial**.printf("ws[%s][%u] disconnect: %u\n", server->url(), client->id());
9	} else if(type == WS_EVT_ERROR){
10	//Se recibió un error desde el otro extremo
11	**Serial**.printf("ws[%s][%u] error(%u): %s\n", server->url(), client->id(), *((uint16_t*)arg), (char*)data);
12	} else if(type == WS_EVT_PONG){
13	//Se recibió el mensaje pong (en respuesta a una solicitud de ping)
14	**Serial**.printf("ws[%s][%u] pong[%u]: %s\n", server->url(), client->id(), len, (len)?(char*)data:"");
15	} else if(type == WS_EVT_DATA){

```
16  //Paquetes de datos
17    AwsFrameInfo * info = (AwsFrameInfo*)arg;
18    String msg = "";
19   if(info->final && info->index == 0 && info->len == len){
20    if(info->opcode == WS_TEXT){
21    for(size_t i=0; i < info->len; i++) {
22        msg += (char) data[i];
23   }
24  }else{
25    char buff[3];
26    for(size_t i=0; i < info->len; i++) {
27    sprintf(buff, "%02x ", (uint8_t) data[i]);
28    msg += buff ;
29   }
30  }
31
```

```
32      if(info->opcode == WS_TEXT)
33        ProcessRequest(client, msg);
34
35      }else{
36      //El mensaje se compone de varios marco y se divide en varios paquetes
37      if(info->opcode == WS_TEXT){
38        for(size_t i=0; i < len; i++) {
39      msg += (char) data[i];
40        }
41      } else {
42        char buff[3];
43        for(size_t i=0; i < len; i++) {
44          sprintf(buff, "%02x ", (uint8_t) data[i]);
45      msg += buff ;
46        }
47      }
```

```
48      Serial.printf("%s\n",msg.c_str());
49
50      if((info->index + len) == info->len){
51
52        if(info->final){
53
54      if(info->message_opcode == WS_TEXT)
55
56      //Se invoca a un metodo para procesar el mensaje recibido en un fichero aparte.
57
58       ProcessRequest(client, msg);
59
60          }
61
62         }
63
64         }
65
66       }
67
68      }
69
70      //Inicializar el servicio del websocket
71
72      void InitWebSockets(){
73
74        ws.onEvent(onWsEvent);
75
76        server.addHandler(&ws);
77
```

```
    Serial.println("WebSocket server started");
}
```

Finalmente tenemos el código fichero 'Websockets.hpp' que se muestra en la Tabla 34, contiene el método 'ProcessRequest' que hemos definimos en el archivo anterior, dicho método nos ayudar a gestionar los mensajes del cliente, en este ejemplo no recibiremos un mensaje por parte del cliente solo el servidor comunicara al cliente notificando la distancia censada en cm, pero si se llega a necesitar recibir mensajes podemos realizar alguna acción dentro de dicho método.

Tabla 34

Código del Archivo CONFIG_WEBSOCKET.hpp

Websockets.hpp

```
//Crea un objeto ws que se conectara mediante el path '/ws'
AsyncWebSocket ws("/ws");

//El siguiente método nos sirve para recibir las peticiones que realiza el cliente por medio de websockets
void ProcessRequest(AsyncWebSocketClient *client, String request){
  //Hacer algo…
}
```

Desarrollo del Frontend

Para el desarrollo del sitio web se hace uso del framework VUEJS, para poder usarlo se lo realizara mediante el uso de su CDN, además se utiliza la librería Vuetify que nos ayudara con los aspectos gráficos de la interfaz de usuario igualmente utilizamos su CDN en el proyecto.

Como se logra apreciar en la Figura 74 al colocar la IP que se nos asignó al momento de conectarnos a una red wifi en el navegador, nuestro backend nos servirá la interfaz en donde mostramos la distancia en cm que se recibe por parte del backend mediante la tecnología websockets.

Además, se encuentra una barra de progreso lineal que acorde a la distancia que este cambiara su porcentaje y color, si la distancia es menor a los 10cm se coloca de color rojo, si es mayor a los 10cm y menor a los 30cm se coloca de color amarillo y si es mayor a 30cm este se tornara de color verde, esta acción se la realiza de manera reactiva gracias al uso framework vuejs.

En la configuración del fichero 'index.html' se incluye los cdn para usar VUEJS, VUETIFY y una librería llamada VueNativeSock que nos permitirá mantener una conexión de manera más sencilla con el webockets, adicional se define los componentes de vuetify 'v-card' que es aquel donde se muestra la distancia y el componente 'v-progress-linear' que es la barra de progreso en la cual la usamos para aplicar una pequeña animación de cargar según la distancia en que se encuentre midiendo nuestro sensor.

Figura 74

Ejecución de la Aplicación del Lado del Cliente

Tabla 35

Código del Archivo principal index.html

index.html

```
1  <!DOCTYPE html>
2  <html>
3  <head>
4  <link
   ref="https://fonts.googleapis.com/css?family=Roboto:100,300,400,500,700,900"
   rel="stylesheet">
```

```
5   <link
    href="https://cdn.jsdelivr.net/npm/@mdi/font@6.x/css/materialdesignicons.min.css
    " rel="stylesheet">
6   <link              href="https://cdn.jsdelivr.net/npm/vuetify@2.x/dist/vuetify.min.css"
    rel="stylesheet">
7   <meta name="viewport" content="width=device-width, initial-scale=1, maximum-
    scale=1, user-scalable=no, minimal-ui">
8   </head>
9   <body>
10    <div id="app">
11          <v-app>
12   <v-row class="d-block">
13              <v-col md="6" offset-md="3" class="mt-5">
14     <v-card color="#385F73" dark class="text-center">
15      <v-card-title class="text-h5 text-center d-flex justify-center">
16       DISTANCIA
17      </v-card-title>
18
19     <v-card-subtitle class="mt-3 text-h4">
20      {{ distancia }}cm
21     </v-card-subtitle>
```

```
1
2
2
2      </v-card>
3
2       </v-col>
4
2       <v-col md="6" offset-md="3" class="mt-4">
5
2         <v-progress-linear
6
2                   :color="estado"
7
2                   height="16"
8
2                   :value="valor"
9
3                   striped>
0
3             </v-progress-linear>
1
3       </v-col>
2
3     </v-row>
3
3       </v-app>
4
3     </div>
5
3      <script                          src="https://cdn.jsdelivr.net/npm/vue-native-websocket-
6     forkgz@2.0.10/dist/build.min.js"></script>
3     <script type="text/javascript" src="./vendor/vue.min.js"></script>
```

```
7
3    <script src="https://cdn.jsdelivr.net/npm/vuetify@2.x/dist/vuetify.js"></script>
8
3    <script type="text/javascript" src="./js/app.js"></script>
9
4
0
4     </body>
1
4    </html>
2
```

En el fichero 'app.js' se define la conexión del websocket haciendo uso de la librería vueNativeSock, al montarse el componente de vuejs se define las variables reactivas como distancia, estado, valor que son utilizada para mostrarse dentro del componente y se aplica una sencilla lógica para tratar estos datos según la distancia recibida tal como se ve en el contenido de a continuación.

Tabla 36

Código del Archivo principal App.js

App.js

```
1    Vue.use(VueNativeSock.default, 'ws://TU_IP_ASIGNADA/ws')
2
3    new Vue({
4      el: '#app',
5      vuetify: new Vuetify(),
       data: function(){
6    //Definimos las variables a usar
7      return{
8          distancia: 0,
9          estado: '',
1          valor: 0
```

```
0
            }
1
1
        },
1
2
    //Esta accion se ejecuta una vez que se monte el componente
1
3
        mounted() {
1
4
         //onmessage escucha los mensaje del websocket
1
5
            this.$socket.onmessage = ({data}) => {
1
6
             this.distancia = data //Se guarda la distancia
1
7
        /*
1
8
          Se aplica una regla de 3 para obtener la proporcionalidad
1
8         del 100% la cual se usa para la barra de progreso
        */
2
0
             let conversion = parseInt((parseInt(data) * 100) / 30);
2
1
          this.valor = conversion;
2
2
              //Se aplica un color de fondo segun la distancia recibida
2
3
              if(conversion > 62){
2
4
                  this.estado = 'green lighten-1';
2
5
              }else if(conversion <= 62 && conversion >= 35){
2
```

```
26
27                 this.estado = 'yellow lighten-1';
28             }else if(conversion < 35){
29                 this.estado = 'red lighten-1';
30             }
31         }
32     }
33 })
```

Conectar un ESP32 con una Aplicación en Vuejs con VueNative Websocket.

En este ejemplo práctico vamos a realizar una comunicación de nuestro cliente con nuestra tarjeta ESP32 mediante websockets.

Según (Manzanares, 2021) “Websocket es una tecnología desarrollada por el W3C (World Wide Web Consortium) que permite establecer una comunicación bidireccional entre cliente y servidor proporcionando canales a través de una única conexión TCP.”, mediante el uso de esta tecnología podemos tener una comunicación más interactiva y ágil entre nuestro cliente y servidor.

Figura 75

Ejecución de la Aplicación del Lado del Cliente

El objetivo en la realización del lado del cliente es mantener una comunicación con el servidor, es decir en recibir mensajes por parte del servidor y que el cliente pueda emitir mensajes hacia el servidor, para el desarrollo del frontend se hace uso del framework Vuejs, en la interfaz como se aprecia en la figura 98 se muestra el tiempo en milisegundos que nos envía nuestro servidor ESP32 mediante websocket, para lograr enviar un mensaje por parte del cliente la aplicación web cuenta con un botón que enviara un mensaje para alternar el estado de encendido o apagado del GPIO 2 que es el pin del led integrado que cuenta el ESP32, de esta manera logramos mantener una comunicación bidireccional con el cliente y el servidor.

Tabla 37

Instalaciones de librerías necesarias para el backend.

Librerias	**Enlace de Descarga**	**Descripción**
ESPAsyncWebServer	https://github.com/me-no-dev/ESPAsyncWebServer	"Librería Open Source podemos crear un servidor asíncrono, es decir, que es capaz de atender a varios clientes de forma simultánea." a
ESPAsyncTCP	https://github.com/me-no-dev/ESPAsyncTCP	Esta es una biblioteca TCP totalmente asíncrona,

	destinada a permitir un entorno de red multiconexión sin problemas que es la base para la implementación de la librería ESPAsyncWebServer.

Nota. La tabla muestra aquellas librerías que ayudaran a montar el web Server y hacer uso de websockets.

Esta tabla ha sido adaptada de “Cómo hacer un servidor asíncrono en ESP8266 o ESP32”, por Luis Lamas, 2022 (https://www.luisllamas.es/como-hacer-un-servidor-asincrono-en-esp8266/).

En el fichero ‘.ino’ como se parecía en la tabla 38 primeramente incluiremos la librería WiFi que lo usamos para conectarnos hacia alguna red, seguido incluimos la biblioteca ‘ESPAsyncWebServer’ que es necesario descargarlo en conjunto con ‘AsyncTCP’ para su correcto funcionamiento, esta librería trae incluido soporte para trabajar con websockets y servicios web, además tenemos separado la lógica del programa en 4 ficheros por funcionalidades para que el código esté más ordenado y reusable.

Tabla 38

Código del fichero principal iot-esp32.ino

iot-esp32.ino

```
1  #include <WiFi.h>
2  #include <ESPAsyncWebServer.h>
3
4  #include "config.h"
5  #include "Websocket.hpp"
6  #include "WIFI.hpp"
7  #include "WebSocketsConfig.hpp"
8
9  void setup() {
10   pinMode(LED_BUILTIN, OUTPUT);
```

```
11
12      Serial.begin(9600);
13
14      conectarWifi();
15
16       InitWebSockets();
17      }
18
19     void loop() {
20
21        ws.textAll(String(millis()));
22        delay(500);
23     }
```

Setup

Dentro del setup le especificamos a través del método pinMode que nuestro led integrado (LED_BUILTIN) se comporte como salida, después inicializamos nuestro monitor serial e incluimos los métodos de inicialización conectarWifi() e InitWebSockets() como se logra apreciar en la tabla 39.

Tabla 39

Código del Bloque Setup del Sketch Principal.

```
1    pinMode(LED_BUILTIN, OUTPUT);
2    Serial.begin(9600);
3    conectarWifi();
4    InitWebSockets();
```

Loop

En la sección del loop, mediante el objeto ws que es una instancia del Websocket se envía un texto a todos los clientes el tiempo que transcurre en milisegundos, esto se hace de manera indefinida cada 500 milisegundos, el código quedaría de la siguiente manera como se aprecia en la tabla 40.

Tabla 40

Código del Bloque Loop del Sketch Principal.

```
1    ws.textAll(String(millis()));
2    delay(500);
```

En la tabla 41 se crea las variables que contiene las credenciales de la red a la que se desea conectar.

Tabla 41

Código del Fichero 'config.h'

```
1    const char* ssid = "REEMPLAZAR_CON_SU_SSID";
2    const char* password = "REEMPLAZAR_CON_SU_PASSWORD";
```

En la configuración como se muestra en la tabla 42 contiene una función que permite conectarnos a una red wifi, esta hará uso de las credenciales que creamos anteriormente, una vez que se logre conectar se imprimirá través del monitor serial el SSID a la que nos encontramos conectado y la ip

que se nos asignó, esta ip es muy importante conocer cuál es dado que la usaremos más adelante en nuestra aplicación de vue para conectarnos mediante el websockets

Tabla 42

Código del Fichero 'WIFI.hpp'

WIFI.hpp

```
1   void conectarWifi(){
2     WiFi.mode(WIFI_STA);
3     Serial.println("Conectando...");
4     WiFi.begin(ssid, password);
5
6   //Esperar a que nos conectemos
7   while(WiFi.status() != WL_CONNECTED){
8       delay(100);
9       Serial.print(".");
10  }
11
12    Serial.println();
13
14    Serial.print("Conectado a:\t");
15
16    Serial.println(WiFi.SSID());
17
18    Serial.print("IP address:\t");
19
20    Serial.println(WiFi.localIP());
21
22  }
23
```

En la tabla 43 se muestra la configuración necesaria para el uso del asyncwebsocket, en el método onWsEvent que se encuentra en la línea 1 contiene una serie de condiciones if en la cual representa las siguientes acciones:

- WS_EVT_CONNECT cuando un cliente se ha conectado al websocket;
- WS_EVT_DISCONNECT cuando un cliente se ha desconectado del websocket;
- WS_EVT_PONG en respuesta a una solicitud de ping;
- WS_EVT_ERROR cuando se recibe un error del cliente.
- WS_EVT_DATA cuando se recibe un paquete de datos del cliente;

En la condición final del WS_EVT_DATA esta lanzara un método llamado “ProcessRequest” que se encuentra en otro fichero, como se muestra en la Tabla 43, en ella se gestionara aquellos mensajes nos envié un cliente, seguido existe un método denominado InitWebSockets que lo que va hacer es inicializar nuestro servidor servidor y el websockets.

Tabla 43

Código del Fichero ‘WIFI.hpp’

WebSocketsConfig.hpp

```
1  void onWsEvent(AsyncWebSocket * server, AsyncWebSocketClient * client,
   AwsEventType type, void * arg, uint8_t *data, size_t len){
2  //Cliente Conectado
3  if(type == WS_EVT_CONNECT){
4      client->printf("Hello Client %u :)", client->id());
5      client->ping();
6  } else if(type == WS_EVT_DISCONNECT){
7  //Cliente Desconectado
8  Serial.printf("ws[%s][%u] disconnect: %u\n", server->url(), client->id());
9  } else if(type == WS_EVT_ERROR){
10 //Se recibió un error desde el otro extremo
11 Serial.printf("ws[%s][%u] error(%u): %s\n", server->url(), client->id(),
   *((uint16_t*)arg), (char*)data);
12 } else if(type == WS_EVT_PONG){
```

```
    //Se recibió el mensaje pong (en respuesta a una solicitud de ping tal vez)

    Serial.printf("ws[%s][%u] pong[%u]: %s\n", server->url(), client->id(), len, (len)?(char*)data:"");
    } else if(type == WS_EVT_DATA){

    //Paquetes de datos

      AwsFrameInfo * info = (AwsFrameInfo*)arg;

      String msg = "";

    if(info->final && info->index == 0 && info->len == len){

      if(info->opcode == WS_TEXT){

      for(size_t i=0; i < info->len; i++) {

          msg += (char) data[i];

      }

    }else{

      char buff[3];

      for(size_t i=0; i < info->len; i++) {

      sprintf(buff, "%02x ", (uint8_t) data[i]);

      msg += buff ;
```

```
29      }
30  }
31
32  if(info->opcode == WS_TEXT)
33    ProcessRequest(client, msg);
34
35  }else{
36  //El mensaje se compone de varios marco y se divide en varios paquetes
37  if(info->opcode == WS_TEXT){
38    for(size_t i=0; i < len; i++) {
39  msg += (char) data[i];
40    }
41  } else {
42    char buff[3];
43    for(size_t i=0; i < len; i++) {
44      sprintf(buff, "%02x ", (uint8_t) data[i]);
```

```
45      msg += buff ;
46        }
47      }
48      Serial.printf("%s\n",msg.c_str());
49
50      if((info->index + len) == info->len){
51        if(info->final){
52      if(info->message_opcode == WS_TEXT)
53      //Se invoca a un metodo para procesar el mensaje recibido en un fichero aparte.
54       ProcessRequest(client, msg);
55          }
56        }
57       }
58      }
59      //Inicializar el servicio del websocket
60
```

```
61  void InitWebSockets(){
61
62    ws.onEvent(onWsEvent);
62
63    server.addHandler(&ws);
63
64    Serial.println("WebSocket server started");
64
65  }
65
```

Finalmente, en la tabla 44 contiene el código del último fichero del backend, atreves del método que se encuentra a partir de la línea 10 se encarga de recibir aquellos mensajes que realiza nuestro cliente, en este ejemplo sencillo el cliente nos envía un dato en la cual se la pasamos a un método como parámetro para que nos encienda o apague un led dependiendo si es true o falso, además creamos un objeto ws que se conectara mediante el path '/ws' y nuestro server estará escuchando mediante el puerto 80.

Tabla 44

Código del Fichero 'WebSocketsConfig.hpp'

WebSocketsConfig.hpp

```
1   AsyncWebSocket ws("/ws");
2
3     AsyncWebServer server(80);
4
5     void ledOnOff(String request){
6     bool estado = request == "true" ? true : false;
7     digitalWrite(LED_BUILTIN, estado);
8    }
9
1   void ProcessRequest(AsyncWebSocketClient *client, String request){
```

```
0
1       ledOnOff(request);
1
1     }
2
```

Desarrollo del Frontend

Tabla 45

Instalaciones necesarias para el frontend.

	Enlace de Descarga	***Descripción***
NodeJs	https://nodejs.org/es/	"Node.js®, Node.js, es un entorno en tiempo de ejecución multiplataforma para la capa del servidor (en el lado del servidor) basado en JavaScript."
Vue CLI	https://cli.vuejs.org/	"Es un conjunto de herramientas para la creación rápida de prototipos de las aplicaciones Vue."

Nota. La tabla muestra aquellos paquetes necesarios para lograr trabajar con el framework vuejs.

Esta tabla ha sido adaptada de "Aumente el flujo de trabajo de Vue.js con Vue CLI 3", por Luis Chiabrera, 2018 (https://code.tutsplus.com/es/tutorials/boost-your-vuejs-workflow-with-vue-cli-3--cms-32232).

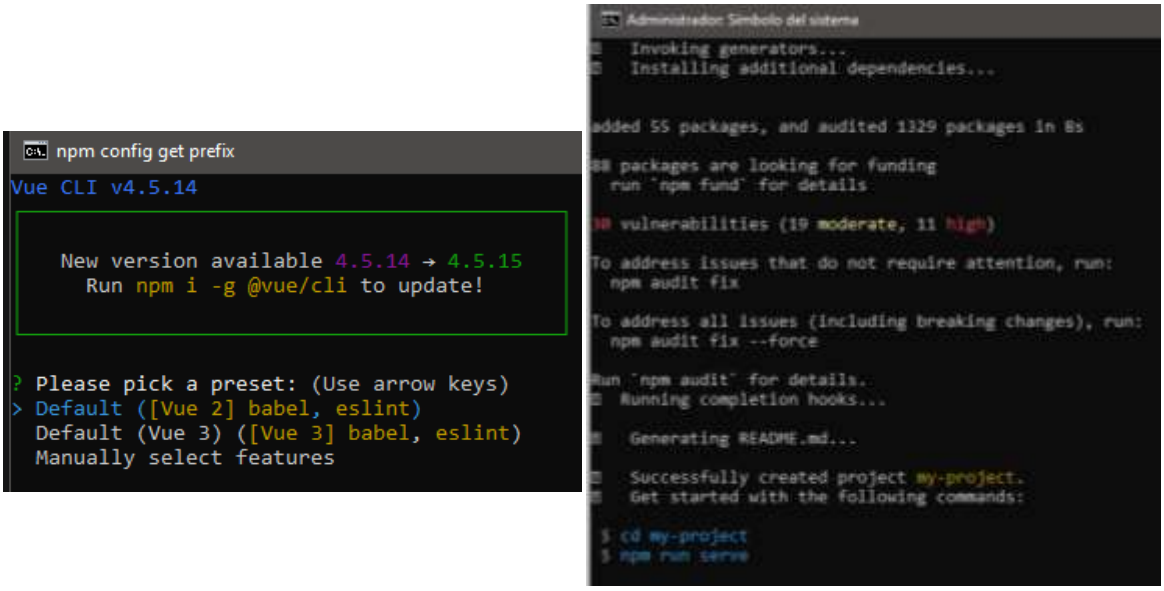
Como se mencionó antes vamos hacer uso de vuejs para nuestro frontend, primeramente, es necesario que tenga instalado Nodejs y seguido instalarse vue-cli que a través de comandos este nos generara el scaffolding de base de la aplicación vuejs.

Vuejs es un framework de JavaScript el cual fue desarrollado por Evan You, quien trabajaba para Google, realizando prototipos para otros proyectos, pero luego decidió diseñar una opción más comprensible que satisfaga las necesidades al momento de hacer prototipos. Según (José) en su artículo Vuejs "es un f**ramework progresivo**, es decir, es un Framework que sirve para consumir interfaz del usuario". (Moncayo, 2017)

Para crear el proyecto abrimos una terminal y digitamos el siguiente comando: ***vue create my-project*** se nos desplegara el siguiente menú cómo se logra apreciar en la Figura 76 y escogemos la instalación por default con la versión 2 de vue:

Figura 76

Ejecución de la Aplicación del Lado del Client



Una vez que finalice podemos abrir nuestro proyecto en algún entorno de desarrollo de preferencia e instalamos el siguiente paquete ‘npm i vue-native-websocket-forkgz’, este paquete nos ayudara a conectarnos mediante websockets.

Ya en nuestro archivo **src/main.js** de nuestro proyecto vue agregamos las siguientes líneas de código:

```
import VueNativeSock from 'vue-native-websocket-forkgz'
Vue.use(VueNativeSock, 'ws://colocar_tu_ip_asignada/ws')
```

Esta configuración nos permitirá hacer la conexión de nuestro websocket es solo necesario reemplazarle con la ip que nos asignado cuando nos conectamos a la red wifi.

Ahora realizaremos unos cambios en el componente que viene por defecto en nuestro proyecto src/components/HelloWorld.vue, y la dejamos cómo se logra apreciar en la tabla 46 en este componente se muestra un título que dirá Milisegundo que es un dato que se recibe como props, después se reflejara en una etiqueta h2 el tiempo de milisegundo que es dato que recibimos de nuestro esp32 cada 500 milisegundos mediante websockets.

Además, existen 2 botones para el encendido y apagado del led integrado de la placa esp32, al dar click estos accionaran un método que recibe el estado del led y por último se muestra un mensaje del estado en que se encuentra el led encendido/apagado.

Tabla 46

Código del Componente HelloWorld.vue

HelloWorld.vue

```
1.  <template>
2.    <div>
3.      <h1>{{ msg }}</h1>
4.
5.      <h2>{{ milis }}</h2>
6.
7.    <button v-if="!estado" type="button" class="btn btn-success mt-4"
      @click="ledOnOff(true)">
8.          Encender Led
9.    </button>
10.
11.   <button v-else type="button" @click="ledOnOff(false)"
12.    class="btn btn-danger mt-4">
13.      Apagar Led
14.   </button>
15.
16.   <br>
17.
```

```
.
18      <h5 class="mt-2">
.
19       El led esta {{ estado ? 'Encendido' : 'Apagado' }}
.
20      </h5>
.
21
.
22      </div>
.
23    </template>
.
```

Dentro de la etiqueta script, en la propiedad props como se muestra en la tabla 47 en la línea 4 se recibe un dato msg que usamos para colocarlo como título en nuestra aplicación, también creamos 2 variables uno denominado milis que es aquel que contendrá los milisegundos que nos envía el esp32 y otro llamado estado que es usado para conocer si el led se encuentra encendido o apagado.

En la propiedad mounted definimos el método. onmessage que es aquel que recibe aquellos mensajes de nuestro servidor, una vez que recibimos el dato actualizamos nuestra variable milis y por últimos se encuentra un método llamado **ledOnOff** que es aquel que envía un mensaje a nuestro servidor para cambiar el estado de nuestro led.

Tabla 47

Código JavaScript del Componente HelloWorld.vue

HelloWorld.vue – JavaScript

```
1   <script>
.
2   export default {
.
3     props: {
```

```
.
    msg: String
.
  },
.
  data(){
.
    return{
.
      milis: 321,
.
      estado: false
.
    }
10.
  },
11.
    mounted(){
12.
    this.$socket.onmessage = (info) => {
13.
      this.milis = info.data;
14.
    }
15.
  },
16.
  methods: {
17.
    ledOnOff(estado){
18.
      this.estado = estado;
1
```

```
19.
20.            this.$socket.send(estado);
21.          }
22.        }
23.      }
24.      </script>
```

Aplicación WebSockets con autentificación

En la siguiente sesión se desarrollará un proyecto que consiste que a través de una interfaz web se pueda visualizar los datos enviados en formato JSON de varios sensores y los estados de 3 leds, la aplicación contara con un login para validar su sesión y si los datos ingresados son correctos pasara a un panel administrativo en donde podrá gestionar a los usuarios, controlar los leds y ver los datos de lectura de los sensores, cabe resaltar que para el funcionamiento de la aplicacion no es necesario contar con internet, solo es necesario conectarse a la intranet del hogar y los protocolos de comunicación a usar será por medio de Websocket y peticiones Ajax permitiéndonos obtener los datos sin necesidad de recarga el sitio web.

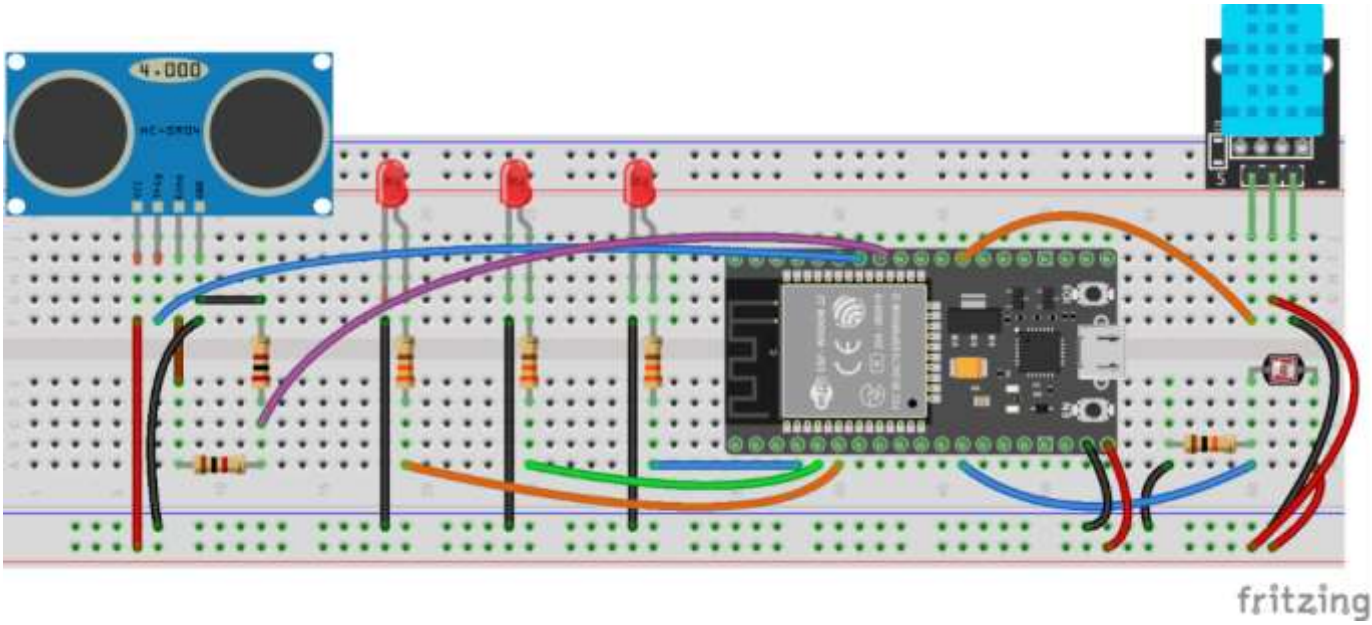
Tal y como se logra apreciar en la Figura 77 se muestra el esquema de conexión de los componentes del proyecto, los materiales usados para este ejemplo son:

- Módulo de temperatura y humedad dht11
- Sensor ultrasonido HC-SR04
- Sensor de luz LDR.
- 3 diodos led
- Jumpers
- Esp32 Doit Devkit v1 Board
- Protoboard

- Resistencias 3 de 330 Ω, 1 de 10kΩ, 1 de 1k kΩ y 1 de 2kΩ

Figura 77

Esquema de Conexión de los sensores y leds al ESP32



Desarrollo Del Backend

Primeramente, se explicará las configuraciones necesarias que se debe hacer en el backend ósea a nuestra placa esp32, es necesario haber descargado aquella librería que se mencionan en la tabla 48 que son librerías que no vienen por defecto al instalar el entorno de desarrollo del Arduino Ide, estas librerías nos permitirán realizar varias funciones que se explicarán posteriormente.

Tabla 48

Instalaciones necesarias en el Backend

Librería	***Enlace de Descarga***	***Descripción***
ESPAsyncWebServer	https://github.com/me-no-dev/ESPAsyncWebServer	"Es una librería Open Source podemos crear un servidor asíncrono, es decir, que es capaz de atender a varios clientes de forma simultánea." [a]
ESPAsyncTCP	https://github.com/me-no-	Esta es una biblioteca TCP totalmente asíncrona, destinada a

	dev/ESPAsyncTCP	permitir un entorno de red multiconexión sin problemas que es la base para la implementación de la librería ESPAsyncWebServer.
DHT sensor library	https://github.com/adafruit/DHT-sensor-library	"Permite obtener los valores de temperatura y humedad de los sensores dth." [b]
ArduinoJson	https://github.com/bblanchon/ArduinoJson	"Arduino Json que incorpora funciones para serializar y deserializar objetos de forma sencilla."[c]
SPIFFS	https://github.com/me-no-dev/arduino-esp32fs-plugin	"Nos brinda un sistema de archivos para con ciertas condiciones en la cual podemos acceder para leer o escribir ficheros." [d]
AsyncElegantOTA	https://github.com/ayushsharma82/AsyncElegantOTA	"Una biblioteca de interfaz de usuario que proporciona elementos interactivos para sus actualizaciones inalámbricas" [e]

Nota. [a] (LLAMAS, 2019). [b](Alzate, 2019). [C](Damián, 2020). [d](GARCIA, 2019). e (Sharma, s.f.).

En la tabla 49 que se muestra a continuación se lograr ver el sketch del fichero principal de código en Arduino, en las líneas del 1 al 6 consiste en incluir las librerías que se descargaron previamente, después en las líneas del 8 al 12 se incluyen demás ficheros de configuración, que se encuentran dividido del skept principal con el fin de que la lectura del código sea más entendible y facil de leer.

Posteriormente sigue la sesión del setup que la función que realiza es el inicialiar varias librerías como el webServer, los sensores, el monitor serial, conectarnos a una red wifi y los 3 leds.

En ell bloque del void loop desde la línea de código del 35 hasta 43 es para obtener la distancia usando el censor de ultranosido, después desde la línea 42 hasta 48 nos permite obtener los datos de temperatura y humedad que capta nuestro censor dht11, laen la línea 50 nos ayuda obtener la cantidad de luz que recibe el sensor ldr si se supera a un valor de 1000 en la variable “valorLDR” contendrá un texto que diga “Dia” caso contrario sea “Noche”.

Finalmente las ultima líneas de código consiste en serializar los datos de los sensores y estados del led y enviarlos por medio de websockes a nuestro cliente cada 5 segundos.

Tabla 49

Sketch Principal del Código en Arduino

	IOT.ino
1	#include <ArduinoJson.h>
2	#include <**WiFi**.h>
3	#include <ESPAsyncWebServer.h>
4	#include <SPIFFS.h>
5	#include <**AsyncElegantOTA**.h>
6	#include "DHT.h"
7	
8	#include "config.h"
9	#include "WebSockets.hpp"
10	#include "Server.hpp"
11	#include "ESP32_Utils.hpp"
12	#include "ESP32_Utils_AWS.hpp"
14	void setup() {
15	dht.begin();
16	
17	**Serial**.begin (115200); // inicializa el puerto seria a 115200 baudios

```
18
19      SPIFFS.begin();
20
21      ConnectWiFi_STA();
22
23      InitServer();
24      InitWebSockets();
25      pinMode(Pecho, INPUT);  // define el pin 6 como entrada (echo)
26      pinMode(Ptrig, OUTPUT); // define el pin 7 como salida  (triger)
27
28      //Pines de los leds
29      pinMode(pinFoco1, OUTPUT);
30      pinMode(pinFoco2, OUTPUT);
31      pinMode(pinFoco3, OUTPUT);
32    }
33
34    void loop(){
35      digitalWrite(Ptrig, LOW);
36      delayMicroseconds(2);
37      digitalWrite(Ptrig, HIGH);
38      delayMicroseconds(10);
39      digitalWrite(Ptrig, LOW);
40
41      //Obtener valor de distancia
42      duracion = pulseIn(Pecho, HIGH);
43      distancia = (duracion/2) / 29; // calcula la distancia en centimetros
44
45      //Obtener datos de temperatura y humedad
46      temperatura = dht.readTemperature();
47      humedad = dht.readHumidity();
48      dato_indice = dht.computeHeatIndex(temperatura, humedad, false);
49
```

```
50      valorLDR = (analogRead(analogPin) > 1000) ? "Dia" : "Noche";
51
52      if(!newMessage){
53        serializar();
54        ws.textAll(output);
55        output="";
56      }
57
58      delay(5000);
59    }
```

Por otro lado, creamos un fichero ‘config.h’ que contendrá las inicializaciones de las variables globales y los pines que se harán uso en el esp32 para las conecciones de los leds y sensores tal y como se logra apreciar en la tabla 50.

Tabla 50

Codigo del fichero config.ino

config.ino

```
1   const char* ssid = "TU_SSID";
2   const char* password = "TU_PASSWORD";
3
4   IPAddress ip(192, 168, 1, 200);
5    IPAddress gateway(192, 168, 1, 1);
6    IPAddress subnet(255, 255, 255, 0);
7
8    float temperatura, humedad, dato_indice;
9
10   const int analogPin = 33; //pin del sensor luminocidad
11
12   String valorLDR;
13
```

```
12
13  //Variable usada para conocer cuando el cliente envia un mensaje por websocket
14  bool newMessage = false;
15
16  //output contendra los datos una vez serializado
17  String output;
18
19  byte pinLed1 = 15, pinLed2 = 4, pinLed3 = 5;
20  bool estadoLed1, estadoLed2, estadoLed3 = false;
21
22  #define Pecho 18
23  #define Ptrig 19
24  long duracion, distancia;
25  DHT dht(27, DHT11);
```

Para conectarnos a una red wifi en la tabla 51 nos muestra la configuración necesaria para realizar esta acción en la línea 6 usando el método config de la librería WIFI establece una ip estatica que resulta más practico el poder colocar una ip deseada.

Tabla 51

Código del fichero ESP32_UTILS.hpp

ESP32_UTILS.hpp	
1	void ConnectWiFi_STA(){
2	Serial.println("");
3	WiFi.mode(WIFI_STA);
4	WiFi.begin(ssid, password);
5	
6	WiFi.config(ip, gateway, subnet);
7	
8	while (WiFi.status() != WL_CONNECTED) {
9	delay(100);
10	Serial.print('.');
11	}
12	
13	Serial.println("");
14	Serial.print("Iniciado STA:\t");
15	Serial.println(ssid);
16	Serial.print("IP address:\t");
17	Serial.println(WiFi.localIP());
18	}

En las siguientes instrucciones de código consiste en preparar la configuración del Async Websocket que es aquella librería que permite mantener una conexión bidireccional entre el servidor y el cliente con una alta velocidad de refresco.

Básicamente, recibimos los eventos de Async Websocket y, cuando se recibe un paquete entero, lanzamos la función 'ProcessRequest()' que se encuentra en un fichero aparte para gestionar aquellos mensajes que realiza el cliente.

Tabla 52

Código del fichero CONFIG_WEBSOCKET.hpp

CONFIG_WEBSOCKET.hpp	
1	void onWsEvent(AsyncWebSocket * server, AsyncWebSocketClient * client, AwsEventType type, void * arg, uint8_t *data, **size_t** len){
2	//Cliente Conectado
3	if(type == WS_EVT_CONNECT){
4	client->printf("Hello Client %u :)", client->id());
5	client->ping();
6	} else if(type == WS_EVT_DISCONNECT){
7	//Cliente Desconectado
8	**Serial**.printf("ws[%s][%u] disconnect: %u\n", server->url(), client->id());
9	} else if(type == WS_EVT_ERROR){
10	//Se recibió un error desde el otro extremo
11	**Serial**.printf("ws[%s][%u] error(%u): %s\n", server->url(), client->id(), *((uint16_t*)arg), (char*)data);
12	} else if(type == WS_EVT_PONG){
13	//Se recibió el mensaje pong (en respuesta a una solicitud de ping)
14	**Serial**.printf("ws[%s][%u] pong[%u]: %s\n", server->url(), client->id(), len, (len)?(char*)data:"");
15	} else if(type == WS_EVT_DATA){

```
15
16      //Paquetes de datos
17
18        AwsFrameInfo * info = (AwsFrameInfo*)arg;
19
20        String msg = "";
21
22       if(info->final && info->index == 0 && info->len == len){
23
24        if(info->opcode == WS_TEXT){
25
26        for(size_t i=0; i < info->len; i++) {
27
28            msg += (char) data[i];
29
30       }
31
32      }else{
33
34        char buff[3];
35
36        for(size_t i=0; i < info->len; i++) {
37
38        sprintf(buff, "%02x ", (uint8_t) data[i]);
39
40        msg += buff ;
41
42       }
43
44      }
45
46
47
```

```
1
3 if(info->opcode == WS_TEXT)
2
3    ProcessRequest(client, msg);
3
3
4
3 }else{
5
3 //El mensaje se compone de varios marco y se divide en varios paquetes
6
3 if(info->opcode == WS_TEXT){
7
3    for(size_t i=0; i < len; i++) {
8
3 msg += (char) data[i];
9
4    }
0
4 } else {
2
4    char buff[3];
2
4    for(size_t i=0; i < len; i++) {
3
4       sprintf(buff, "%02x ", (uint8_t) data[i]);
4
4 msg += buff ;
5
4    }
6
4 }
```

```
47
48      Serial.printf("%s\n",msg.c_str());
49
50
51      if((info->index + len) == info->len){
52
53        if(info->final){
54
55      if(info->message_opcode == WS_TEXT)
56
57      //Se invoca a un metodo para procesar el mensaje recibido en un fichero aparte.
58
59       ProcessRequest(client, msg);
60
61          }
62
63        }
64
65       }
66
67      }
68
69      //Inicializar el servicio del websocket
70
71      void InitWebSockets(){
72
73        ws.onEvent(onWsEvent);
74
75        server.addHandler(&ws);
```

```
3
6
4
6
5
```

```
    Serial.println("WebSocket server started");
}
```

Continuando toca analizar el código del fichero “server.hpp” que es lo que muestra en la tabla 53, principalmente la acción que realiza es en crear un api rest en donde asociamos los distintos verbos y acciones para intercambiar información en formato json a las peticiones http que realice algún cliente.

Al realizar una petición GET a la ruta raíz “/” se servirá el archivo “login.html” que se encuentra almacenado en la memoria del esp32 para que los usuarios logren iniciar sesión con su respectiva cuenta, dichas cuentas de usuarios se encontraran almacenado en un archivo denominado “usuarios.json” que hace la función de una pequeña base de datos, con el fin de evitar hacer uso de internet y utilizar algún servicio de base de datos externo.

También en la línea 50 de este fichero se encuentra implementado la librería AsyncElegantOTA, que nos brinca una funcionalidad muy interesante que es el de poder cargar nuestro firmware o los archivos del SPIFFS de manera inalámbrica en lugar del habitual puerto serie.

Tabla 53

Código del fichero Server.hpp

Server.hpp

```
1  AsyncWebServer server(80);
2
3  void crudUsers(uint8_t *usuario, size_t len){
4  //Abrimos el archivo que contiene los usuarios
5   File f = SPIFFS.open("/db/usuarios.json", "w");
6   for(size_t i = 0; i < len; i++){
7    f.write(usuario[i]);  //Sobreescribimos el fichero usuario.json
8   }
9   f.close();
```

```
}

void InitServer(){

 server.serveStatic("/", SPIFFS, "/").setDefaultFile("login.html");

//METODO PARA ELIMINAR UN USUARIO

 server.on("/user", HTTP_DELETE, [](AsyncWebServerRequest *request){},NULL,
[](AsyncWebServerRequest *request, uint8_t *data, size_t len, size_t index, size_t total){
  crudUsers(data, len);

  request->send(200, "application/json", "ok");

});

//METODO PARA OBTENER USUARIOS

 server.on("/user", HTTP_GET, [](AsyncWebServerRequest *request) {

  //Abrimos el archivo que contiene los usuarios

  File f = SPIFFS.open("/db/usuarios.json", "r");

  String users;

  while(f.available()) {
```

```
26        //Se lee line por linea y lo cancatenamos en la variable users
27          users += f.readString();
28       }
29        //Se retorna el objeto json al cliente
30        request->send(200, "application/json", users);
31        f.close();
32      });
33    //METODO PARA CREAR USUARIO
34    server.on("/user", HTTP_POST, [](AsyncWebServerRequest *request){},
      NULL,
35    [](AsyncWebServerRequest *request, uint8_t *data, size_t len, size_t index,
      size_t total){
36      crudUsers(data, len);
37      request->send(200, "application/json", "ok");
38    });
39    //METODO PARA EDITAR USUARIO
40    server.on("/user", HTTP_PUT, [](AsyncWebServerRequest *request){}, NULL,
42    [](AsyncWebServerRequest *request, uint8_t *data, size_t len, size_t index,
      size_t total){
```

```
42          crudUsers(data, len);
43
44          request->send(200, "application/json", "ok");
45
46      });
47
48      //Si no existe una ruta se envia un texto not found
49
50      server.onNotFound([](AsyncWebServerRequest *request) { //Este metodo se
        acciona cuando no encuentra una pagina
51          request->send(400, "text/plain", "Not found");
52
53      });
54
55
56          AsyncElegantOTA.begin(&server, "Fernando", "54321"); // Start ElegantOTA
57
58          server.begin();
59
60          Serial.println("HTTP server started");
61
62      }
63
```

Finalmente contamos con el ultimo fichero llamado “Websocket.hpp” que se aprecia en la tabla 54, se encarga de recibir y enviar mensajes por websocket, estos datos se los convierte en formato json usando el método serializar tal y como se muestra en las líneas del 3 al 14, si algún cliente envia un mensaje para cambiar el estado de algún led se accionara un método llamado “encenderApagarLeds()”, una vez que realice la acción se le devuelve el estado actual en que se encuentre los leds y los datos de los sensores.

Tabla 54

Código del fichero Server.hpp

	Websocket.hpp
1	AsyncWebSocket ws("/ws");
2	
3	void serializar(){
4	StaticJsonDocument<200> doc;
5	doc["distancia"] = String(distancia);
6	doc["temperatura"] = temperatura;
7	doc["humedad"] = humedad;
8	doc["dato_indice"] = dato_indice;
9	doc["luminocidad"] = valorLDR;
10	doc["estadoLed1"] = estadoLed1;
11	doc["estadoLed2"] = estadoLed2;
12	doc["estadoLed3"] = estadoLed3;
13	serializeJson(doc, output);
14	}
15	
16	void encenderApagarLeds(String leds){
17	StaticJsonDocument<96> doc;
18	
19	DeserializationError error = deserializeJson(doc, leds);

```
19
20
21     if (error) {
22       Serial.print("deserializeJson() failed: ");
23       Serial.println(error.c_str());
24       return;
25     }
26
27     bool led1 = doc["led1"];
28     bool led2 = doc["led2"];
29     bool led3 = doc["led3"];
30
31     estadoLed1 = led1;
32     estadoLed2 = led2;
33     estadoLed3 = led3;
34
35     digitalWrite(pinLed1, led1);
```

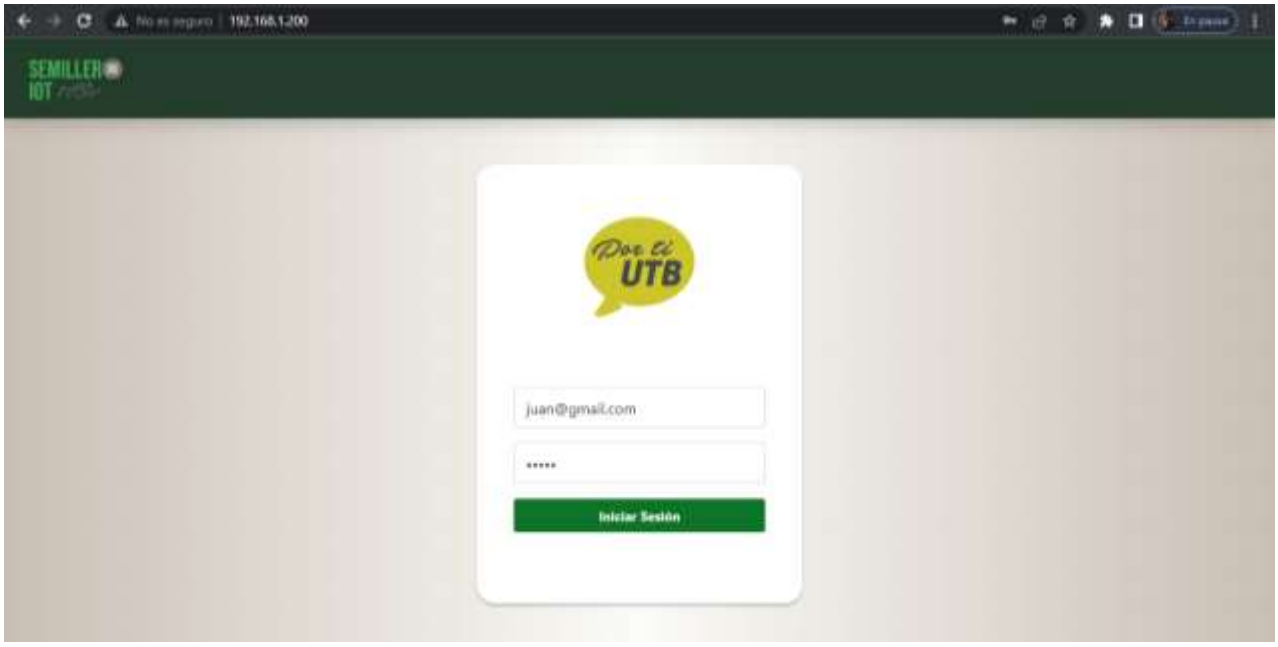
```
35
36        digitalWrite(pinLed2, led2);
37        digitalWrite(pinLed3, led3);
38
39        serializar();
40        ws.textAll(output);  //Enviamos datos al cliente mediante websockets
41        output="";  //Limpiamos variable
42        newMessage = false;
43      }
44    //El siguiente método nos sirve para recibir las peticiones que realiza el cliente por medio de websockets
45    void ProcessRequest(AsyncWebSocketClient *client, String request){
46      newMessage = true;
47      if(newMessage) encenderApagarLeds(request);
48    }
```

Desarrollo Del Frontend

En la parte del cliente se desarrollan varias vistas que nos ayudan a realizar una serie de acciones, primeramente, es necesario que un usuario llegue a iniciar sesión tal como le logra apreciar en la Figura 78, para luego pasar a la parte administrativa de la aplicación web.

Figura 78

Vista de inicio de sesión.



Para validar el inicio de sesión se realiza peticiones htpp al api rest que se implementó en nuestro esp32 anteriormente, en la tabla 8 se muestra que se realiza una petición de tipo get hacia la ruta '/user' para obtener todos los usuarios que se encuentran almacenados en la memoria del esp32 para luego proceder a realizar la validación con las credenciales ingresadas, si los datos son correctos se redirige al usuario a las vista 'leds.html', caso contrario se le muestra una alerta que el correo o contraseña esta incorrecto.

Tabla 55

Código del fichero Server.hpp

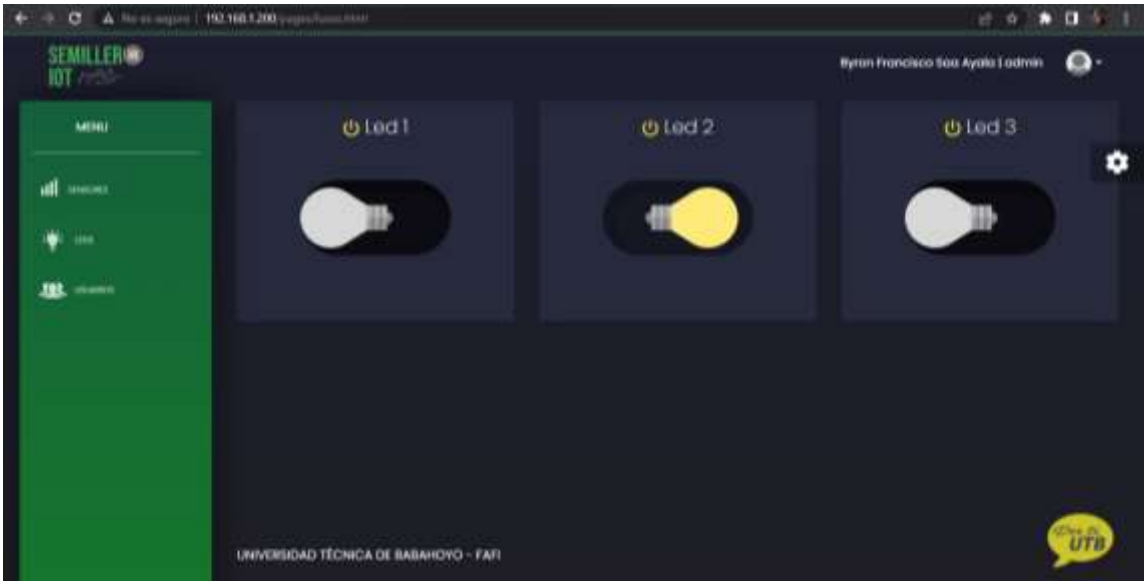
	Login.js
1	const formLogin = document.**querySelector**("#formLogin");
2	const inputEmail = document.**querySelector**("#inputEmail");
3	const inputPassword = document.**querySelector**("#inputPassword");
4	const toastLiveExample = document.**getElementById**('liveToast')
5	
6	formLogin.**addEventListener**('submit', function(e){
7	e.**preventDefault**();

```
8
9       fetch('/user')
10        .then(data => data.json())
11        .then(usuarios => {
12          const resultado = usuarios.find(
13            usuario => usuario.email == inputEmail.value
14          );
15          if(!resultado){
16            return Toast.fire({
17              icon: 'error',
18              title: 'Error, este usuario no existe'
19            })
20          }
21
22          if(resultado.password != btoa(inputPassword.value)){
23            return Toast.fire({
24              icon: 'warning',
25              title: 'La contraseña esta incorrecta'
26            })
27          }
28          sessionStorage.setItem('user', JSON.stringify(resultado));
29          window.location.href = `/pages/usuarios.html`
30        })
31    })
```

Luego de que algún usuario haya iniciado sesión se lo redirige a la vista que se aprecia en la Figura 79, en dicha figura se logra ver como luce la interfaz administrativa, en esta vista se cuenta con 3 switch y nos permite conocer los estados en que se encuentran las luces y controlar el encendido y apagado de los 3 leds mediante el uso de websockets.

Figura 79

Vista para el control de los leds.



En la siguiente tabla numero 56 contiene la configuración para lograr controlar los estados de los leds, en la línea de 1 al 3 se selecciona los elementos con los identificadores del switch para aplicar el evento ‘change’ y llamar al método ‘ledOnOff’ cada vez que exista algún cambio, el método lo que realiza es enviar un mensaje por medio de la coneccion del websocket que se realiza en la línea 5, con un objeto json de los estados de los leds.

A partir de la línea de código del 35 al 47 se logra escuchar los mensajes que nos realiza nuestro backend, se desestructura setea los switches de los leds con el estado actual de las luces.

Tabla 56

Código del fichero luces.js

	luces.js
1	`let led1 = document.querySelector('#led1');`
2	`let led2 = document.querySelector('#led2');`
3	`let led3 = document.querySelector('#led3');`
4	
5	`var connection = new WebSocket('ws://' + location.hostname + '/ws', ['arduino']);`
6	

```
7   function ledOnOff(){
8     let leds = {
9     "led1": led1.checked,
10    "led2": led2.checked,
11
12    "led3": led3.checked
13
14    }
15
16
17  led1.addEventListener('change', () => {
18
19    ledOnOff();
20
21  })
22
23  led2.addEventListener('change', () => {
24
25   ledOnOff();
26
27  })
28
```

```
4
25  led3.addEventListener('change', () => {
26   ledOnOff();
27  })
28
29  //Se ejecuta cuando exita conexion con websocket
30  connection.onopen = function () {
31    console.log('Connected: ');
32  };
33
34  //Se ejecuta cuando exita error de conexion con websocket
35  connection.onerror = function (error) {
36   console.log('WebSocket Error ', error);
37  };
38  //Se ejecuta se reciba mensajes por medio de websocket
39  connection.onmessage = function (e) {
40   const { estadoLed1,
```

```
                estadoLed2,
                estadoLed3 } = JSON.parse(e.data);

        led1.checked = estadoLed1;
        led2.checked = estadoLed2;
        led3.checked = estadoLed3;
    };
    //Se ejecuta cuando se cierra la conexion con websocket
    connection.onclose = function () {
      console.log('WebSocket connection closed');
    };
```

Al dar click en la opción del menú ‘sensores’ se mostrará la interfaz que se aprecia en la Figura 80, en este módulo se refleja los datos de lecturas de los sensores como la temperatura, humedad, la distancia y el sensor de luminosidad que recibimos cada 5 segundos mediante el protocolo websockets.

Figura 80

Vista del panel de lectura de sensores.

En las siguientes instrucciones que se encuentra en la tabla 57, es muy similar al fichero anterior, en el bloque de código que se aprecia en las líneas del 38 al 52, se reciben de igual manera los mensajes que envía el servidor y se muestran los datos en las etiquetas html y en los objetos de ciertos gadgets que representa de manera más interactivas los datos de temperatura y humedad.

Tabla 57

Código del fichero luces.js

	sensores.js
1	let txtDistancia = document.**querySelector**('#count-box');
2	let txtLuminocidad = document.**querySelector**('.txtLuminocidad');
3	
4	var connection = new **WebSocket**('ws://' + location.hostname + '/ws', ['arduino']);
5	
6	**$**('.gauge').**each**(function (index, item) {
7	let params = {
8	initialValue: 0,
9	higherValue: 100,
10	title: `Temperatura`,

```
    subtitle: '0 ºC' };

    let gauge = new GaugeChart(item, params);

    gauge.init();

});

var g1 = new JustGage({

    id: 'g1',

    value: 45,

    min: 0,

    max: 100,

    symbol: '%',

    counter: true

});

function mostrarTemperatura(valor){
```

```
        $('.gauge').each(function (index, item) {

            let gauge = $(item).dxCircularGauge('instance');

            let randomNum = valor;

            let gaugeElement = $(gauge._$element[0]);



            gaugeElement.find('.dxg-title text').last().html(`${randomNum} ºC`);

            gauge.value(randomNum);

        });

    }


    //Se ejecuta se reciba mensajes por medio de websocket

    connection.onmessage = function (e) {

        //Desestructurar objeto json

        const { distancia,

                temperatura,

                humedad,
```

```
            luminocidad } = JSON.parse(e.data);

        //Mostrar luminosidad

        txtLuminocidad.innerHTML  = `${luminocidad}`


        mostrarTemperatura(Math.trunc(temperatura))

        //Mostrar el valor de Humedad

        g1.refresh(parseInt(humedad));

        //Mostrar la distancia

        txtDistancia.innerHTML  = `${distancia} cm`

    };
```

Finalmente, en la Figura 81 se aprecia el ultimo modulo que es para el control de usuario, en esta vista se podrá ver, crear, editar o eliminar usuarios que podrán iniciar sesión en la aplicación, cada usuario cuenta con un rol en específico sea ‘Administrador’ o ‘Invitado’, solo el usuario administrador cuenta con el permiso de poder gestionar el módulo de usuarios, mientras a los usuarios de rol invitado solo podrán hacer uso de los demás módulos.

Figura 81

Vista del panel de lectura de sensores

En la última tabla se encuentra el código JavaScript del módulo del usuario, en este archivo principalmente realiza peticiones htpp a la Api Rest de nuestro backend y cuenta con los métodos para consultar, crear, editar o eliminar a algún usuario

Tabla 58

Código del fichero usuarios.js

usuarios.js	
1	const **cargarUsuarios** = () => {
2	**fetch**('/user')
3	.**then**(response => response.**json**())
4	.**then**(data => {
5	listUsers = data;
6	let filas = ''
7	data.**forEach**((e, index) => {
8	filas += `
9	<tr>
10	<td>${ index += 1 }</td>
11	<td>${ e.usuario }</td>
12	<td>${ e.email }</td>

```
2
1                <td>${ e.rol }</td>
3
1                <td>
4
1                 <button class="btn btn-primary btn-sm btn-action"
5
1     onclick="editarUsuario('${e.id}','${e.usuario}', '${e.email}', '${e.rol}')">
6
1                   Editar
7
1                 </button>
8
1                 <button class="btn btn-danger btn-sm btn-action"
9
2                   onclick="eliminarUsuario('${e.id}')">
0
2                   Eliminar
1
2                 </button>
2
2                </td>
3
2              </tr>`});
4
2            document.querySelector(".tr-users").innerHTML = filas
5
2          })
6
2        }
7
2
```

```
28
29  const eliminarUsuario = async (id) => {
30
31   const result = listUsers.filter(usuario => usuario.id != id);
32
33
34   const data  = await fetch('/user', {
35
36     method: "DELETE",
37
38     body: JSON.stringify(result),
39
40     headers: { "Content-type":  "application/json"}
41
42   }).then(r => r.text());
43
44
45   cargarUsuarios();
46
47  }
48
49
50  const crearUsuario = async () => {
51
52  listUsers.unshift({
53
54   id: uuid.v4(),
55
56   email: email.value,
```

```
4
4       password: btoa(password.value),
5
4       usuario: user.value,
6
4       rol: rol.value
7
4     });
8
4
9
5     const data  = await fetch("/user", {
0
5       method: "POST",
1
5       body: JSON.stringify(listUsers),
2
5       headers: { "Content-type": "application/json"}
3
5     }).then(r => r.text());
4
5
5
5       cargarUsuarios();
6
5       alert("usuario creado");
7
5     }
8
5
9
6     const editarUsuario = async () => {
```

```
                                                                    
    const indice = listUsers.findIndex(x => x.id === userId.value);
                                                                    
    listUsers[indice].email    = email.value;
                                                                    
    listUsers[indice].usuario = user.value;
                                                                    
  if(updatePassword) listUsers[indice].password = btoa(password.value);
                                                                    
  listUsers[indice].rol      = rol.value;
                                                                    
                                                                    
    const data  = await fetch("/user", {
                                                                    
      method: "PUT",
                                                                    
      body: JSON.stringify(listUsers),
                                                                    
      headers: { "Content-type": "application/json" }
                                                                    
    }).then(r => r.text());
                                                                    
                                                                    
  cargarUsuarios();
                                                                    
    alert("Usuario editado");
                                                                    
  }
                                                                    
```

CAPÍTULO V: Protocolos de comunicación para IoT

El IoT le permite solucionar los problemas de su negocio usando sus propios datos. La Internet de las cosas no va únicamente de dispositivos conectados, sino de la información que recopilan esos dispositivos y las eficaces conclusiones inmediatas que se pueden obtener con esa información. Estas conclusiones se pueden usar para transformar su negocio y reducir los costos con mejoras como la reducción del desperdicio de materiales, la optimización de los procesos operativos y mecánicos, o la expansión a nuevas líneas de negocio que solo son posibles con datos confiables en tiempo real. Cree una ventaja competitiva real usando IoT para convertir sus datos en información y esa información en acciones (Microsoft Azure, 2020).

¿Por qué un protocolo para iot?

Cada protocolo de IoT de la arquitectura del sistema de IoT permite la comunicación de dispositivo a dispositivo, de dispositivo a puerta de enlace, de puerta de enlace a centro de datos o de puerta de enlace a nube, así como la comunicación entre centros de datos (Microsoft Azure, 2020).

Protocolos de comunicación para el internet de las cosas (IoT)

Nivel de aplicación

El nivel de aplicación actúa como interfaz entre el usuario y el dispositivo con un protocolo de IoT determinado.

- **Advanced Message Queuing Protocol (AMQP)**
  Nivel de software que crea interoperabilidad entre el middleware de mensajería. Ayuda a que una gran variedad de aplicaciones y sistemas funcionen juntos, lo que permite crear una mensajería normalizada a escala industrial.

- **Protocolo de aplicación restringida (CoAP)**
  Protocolo de red y ancho de banda restringidos diseñado para que dispositivos con capacidad limitada puedan conectarse en la comunicación entre máquinas. CoAP es también un protocolo de transferencia de documentos que se ejecuta a través del Protocolo de datagramas de usuario (UDP).
- **Servicio de distribución de datos (DDS)**

Protocolo de comunicación punto a punto versátil que hace de todo, desde ejecutar pequeños dispositivos hasta conectar redes de alto rendimiento. DDS optimiza la implementación, aumenta la confiabilidad y reduce la complejidad.

- **Message Queue Telemetry Transport (MQTT)**
  Protocolo de mensajería diseñado para la comunicación ligera entre equipos que se usa principalmente para las conexiones de poco ancho de banda con ubicaciones remotas. MQTT utiliza un patrón de publicación-suscripción y es ideal para dispositivos pequeños que requieren un uso eficiente del ancho de banda y de la batería.

Nivel de transporte

En cualquier protocolo de IoT, el nivel de transporte habilita y protege la comunicación de los datos mientras viajan entre niveles.

- **Protocolo de control de transmisión (TCP)**
  Protocolo dominante en la mayor parte de la conectividad con Internet. Ofrece comunicación entre hosts, para lo que divide grandes conjuntos de datos en paquetes individuales que reenvía y vuelve a ensamblar según sea necesario.
- **Protocolo de datagramas de usuario (UDP)**
  Protocolo de comunicaciones que permite la comunicación entre procesos y se ejecuta sobre IP. UDP mejora la velocidad de transferencia de datos a través de TCP y es ideal para las aplicaciones que requieren transmisiones de datos sin pérdida.

Nivel de red

El nivel de red de un protocolo de IoT permite la comunicación entre los dispositivos individuales y el enrutador.

- **IP**
  Muchos protocolos de IoT utilizan IPv4, mientras que las ejecuciones más recientes usan IPv6. Esta actualización reciente de IP redirige el tráfico a través de Internet e identifica y localiza dispositivos en la red.
- **6LoWPAN**
  Este protocolo de IoT funciona mejor con dispositivos de baja potencia que tienen una capacidad de procesamiento limitada.

Nivel de vínculo de datos

El nivel de datos es la parte del protocolo de IoT que transfiere los datos dentro de la arquitectura del sistema e identifica y corrige los errores que encuentra en el nivel físico.

- **IEEE 802.15.4**
  Estándar de radio para una conexión inalámbrica de bajo consumo. Se usa con Zigbee, 6LoWPAN y otros estándares para crear redes inalámbricas insertadas.
- **LPWAN**
  Las redes de área extensa de baja potencia (LPWAN) permiten la comunicación a distancias desde 500 metros hasta más de 10 km en algunos lugares. LoRaWAN es un ejemplo de red LPWAN optimizada para un consumo bajo de energía.

Nivel físico

El nivel físico es el canal de comunicación entre los dispositivos de un entorno específico.

- **Bluetooth Low Energy (BLE)**
  BLE reduce drásticamente el consumo de energía y el costo, y mantiene una distancia de conectividad similar a la del Bluetooth clásico. BLE funciona de forma nativa en todos los sistemas operativos móviles y se está convirtiendo rápidamente en el favorito para la electrónica de consumo por su bajo costo y la larga duración de la batería.
- **Ethernet**
  Esta conexión por cable es una opción menos costosa que proporciona conectividad rápida para datos con una latencia baja.
- **Evolución a largo plazo (LTE)**
  Estándar de comunicación inalámbrica de banda ancha para dispositivos móviles y terminales de datos. LTE aumenta la capacidad y la velocidad de las redes inalámbricas y admite secuencias de difusión y multidifusión.
- **Transmisión de datos en proximidad (NFC)**
  Conjunto de protocolos de comunicación que utilizan campos electromagnéticos y permiten que dos dispositivos se comuniquen si están a una distancia no superior a cuatro centímetros. Los dispositivos habilitados para NFC funcionan como tarjetas de identidad y suelen utilizarse para pagos móviles, vales y tarjetas inteligentes sin contacto.
- **Power Line Communication (PLC)**

Tecnología de comunicación que permite enviar y recibir datos a través de los cables eléctricos. Permite alimentar y controlar un dispositivo IoT a través del mismo cable.

- **Identificación por radiofrecuencia (RFID)**

  El protocolo RFID utiliza campos electromagnéticos para hacer un seguimiento de las etiquetas electrónicas no alimentadas de otro modo. El hardware compatible proporciona energía y se comunica con estas etiquetas para leer su información con fines de identificación y autenticación.

- **Wi-Fi/802.11**

  Wi-Fi/802.11 es la conexión habitual en hogares y oficinas. Aunque es una opción económica, puede que no se ajuste a todos los escenarios por su alcance limitado y el consumo energético ininterrumpido.

- **Z-Wave**

  Red en malla que usa ondas de radio de baja potencia para la comunicación de dispositivo a dispositivo.

- **Zigbee**

  Especificación basada en IEEE 802.15.4 para un conjunto de protocolos de comunicación de alto nivel que se usan para crear redes de área personal con un radio digital pequeño de baja potencia.

Creación de Aplicación Web Utilizando Firebase con Esp32 y Esp8266

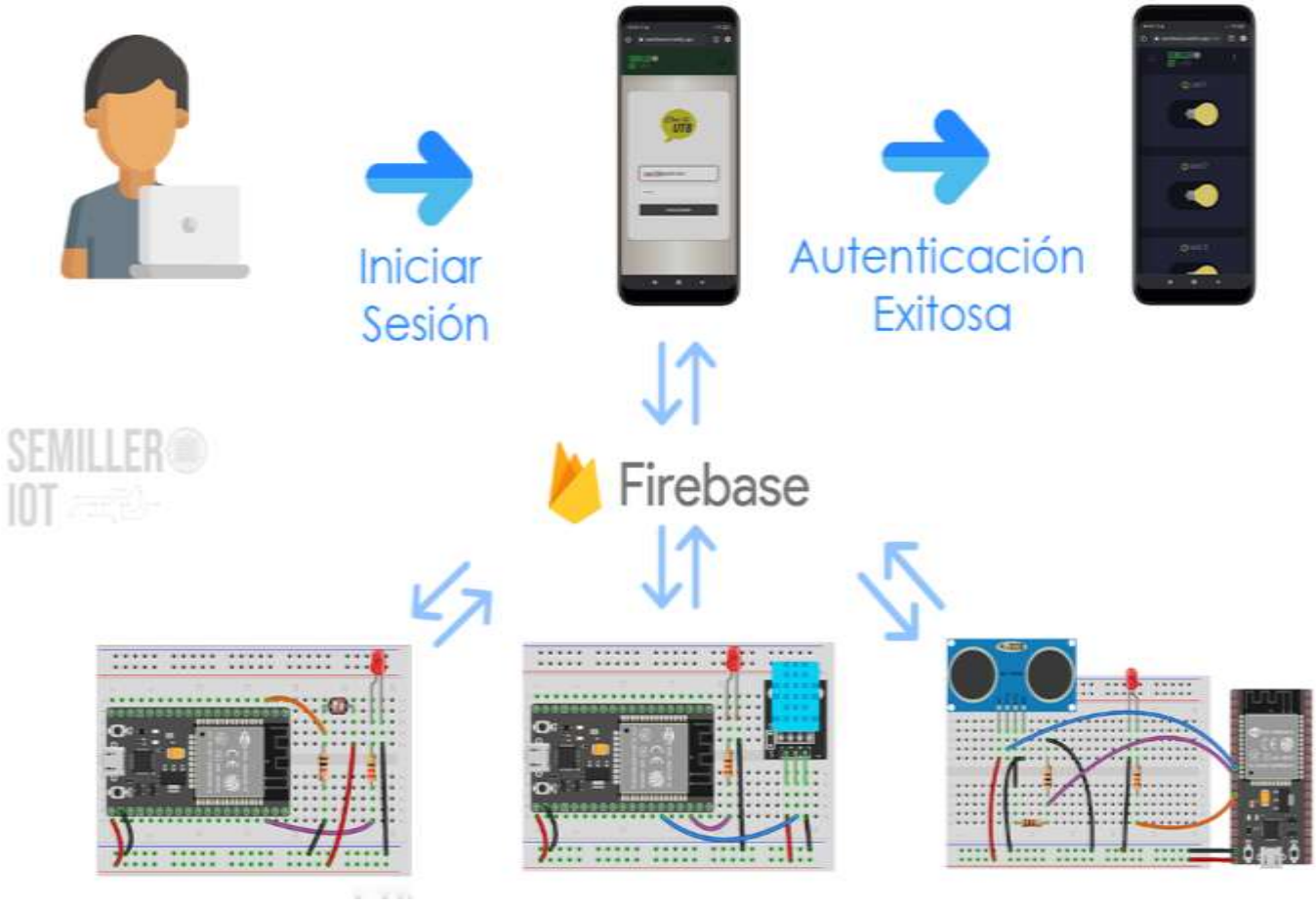
En la siguiente práctica se realizará una aplicación web que nos permitirá leer los datos de varios sensores como el sensor de distancia, sensor de luminosidad y un sensor de luz, además se podrá controlar el encendido y apagado de los leds que tendrán conectados cada placa como el ESP32 y el ESP8266, ambas tarjetas harán uso de la base de datos Realtime de Firebase, que nos ayudará mantener una comunicación en tiempo real con nuestro cliente desde cualquier sitio haciendo uso de internet.

En la Figura 82 se puede apreciar un esquema gráfico en la cual detalla el funcionamiento del proyecto, para hacer uso de la aplicación web es necesario primeramente iniciar sesión mediante un correo y una contraseña, si la autenticación es exitosa lo llevará al panel administrativo en donde se podrá visualizar datos de sensores, y controlar las salidas de los leds que cuenta cada placa, la aplicación web y las tarjetas contarán con un oyente de Firebase que estará pendiente ante cualquier cambio que realice el cliente o el backend para mostrar los datos de los sensores que se haya actualizado o cumplir con una petición de encender o apagar algún led que requiera el cliente.

Entre los materiales a usar serán los mismo usados en la práctica antes vista con el tema “Aplicación WebSockets con autentificación”, con la diferencia que en este ejercicio se utilizara 3 placas, 1 ESP32 y 2 ESP8266 cada uno conectados en un mini protoboard y contando con un tipo de sensor más un led, tal y como se logra apreciar en la Figura 82.

Figura 82

Esquema del funcionamiento del proyecto haciendo uso de Firebase.

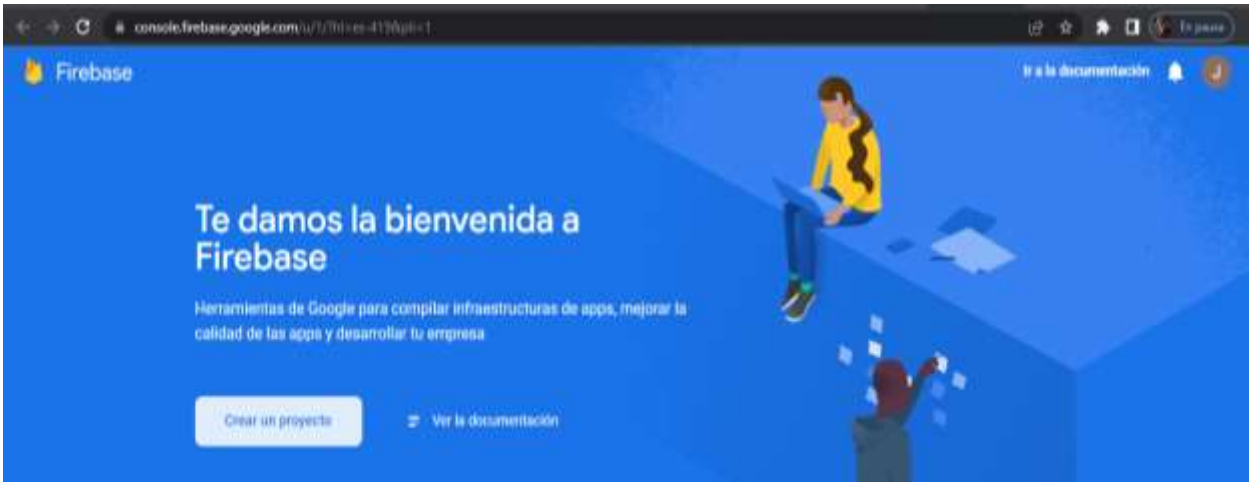


Crear Nuevo Proyecto en Firebase

Se Comenzará explicando la configuración necesaria que se debe realizar en Firebase para posteriormente poder hacer uso de este servicio, para ello es necesario acudir a su sitio oficial que es el siguiente: https://firebase.google.com/, se debe crear una cuenta y después lo enviará a la siguiente interfaz.

Figura 83

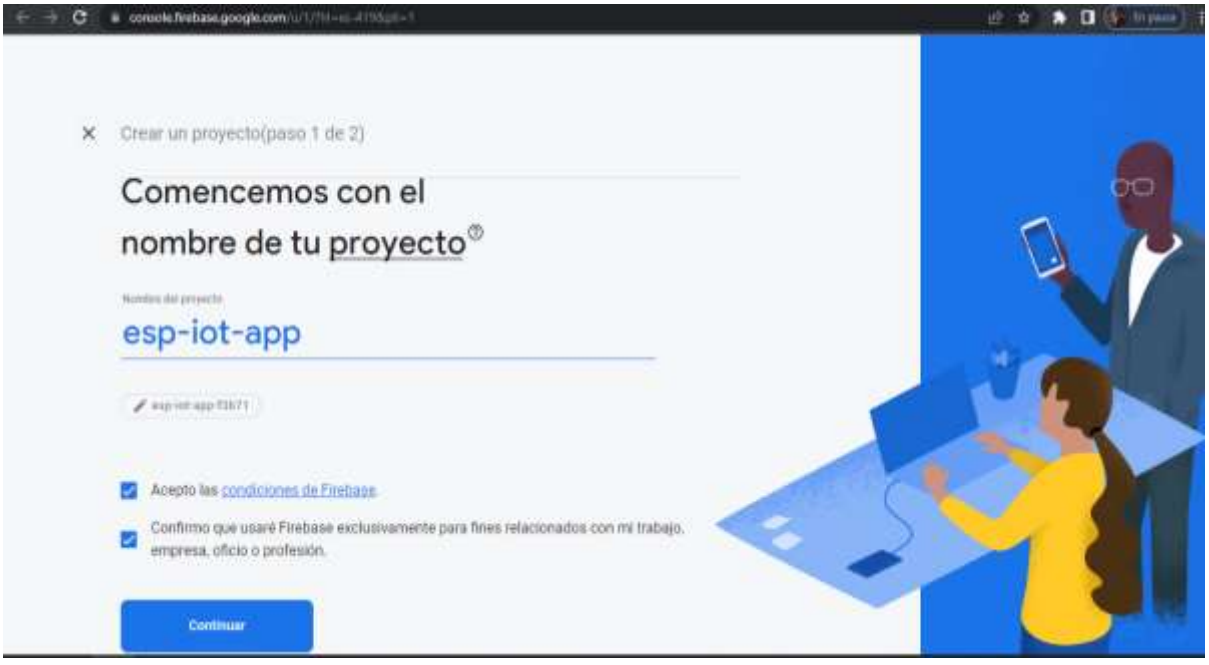
Ventana de bienvenida de Firebase



Una vez dado clic en crear proyecto se mostrará una ventana como se ve en la Figura 84, en la cual definimos un nombre del proyecto y aceptamos los términos y condiciones de uso.

Figura 84

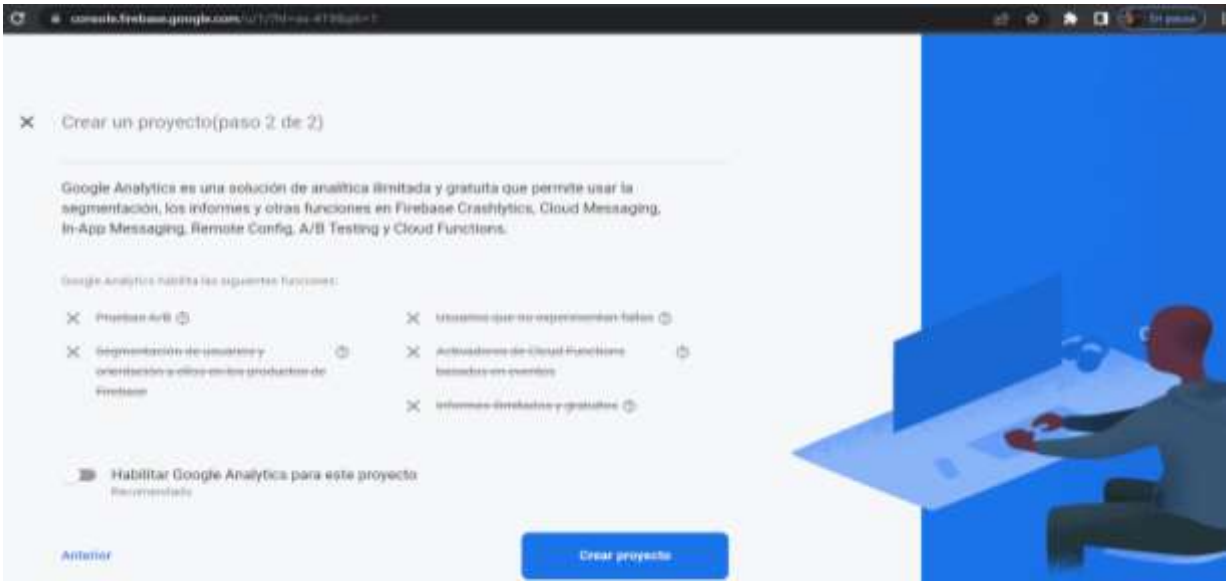
Ventana de creación de proyecto en Firebase



Seguido se muestra una opción para hacer uso Google Analytics y lo deshabilitamos porque no es necesario en este proyecto, se tomará unos segundos en crearse el proyecto, una vez listo daremos clic en continuar.

Figura 85

Ventana con opción de activar servicio de Google Analytics en Firebase.



Una vez finalizado la creación del proyecto en Firebase se mostrará la interfaz que se ve en la Figura 86, el siguiente paso es habilitar el método de autenticación, para ello damos clic en la opción Authentication seguido elegimos por Correo electrónico/contraseña y finalmente chequeamos habilitar por ese método.

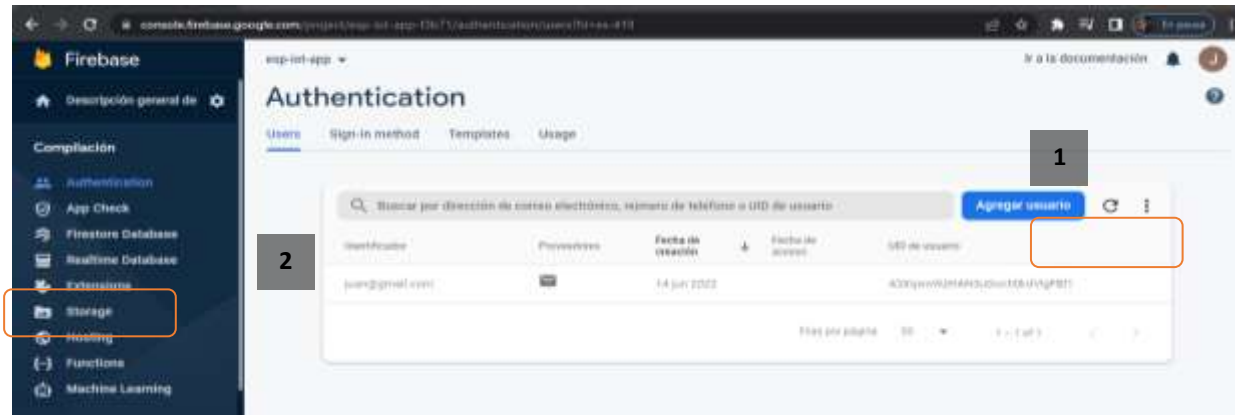
Figura 86

Página de consola del proyecto creado.

Agregamos un usuario de prueba para realizar las prácticas de autenticación por el método de Correo electrónico/contraseña, como siguiente paso ahora es momento de crear la base de datos Realtime que nos ofrece como servicio Firebase, para ello daremos clic en Realtime Database.

Figura 87

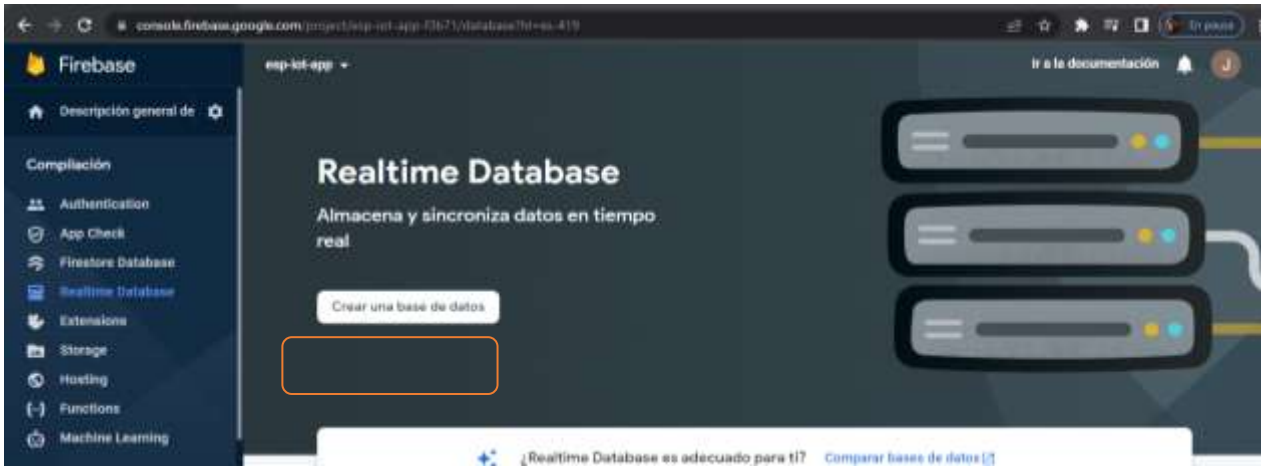
Listado de usuarios para autenticación por Correo Electrónico/Contraseña



Para completar la creación de la base de datos damos clic en crear como se aprecia en la Figura 88, y elegimos las opciones que vienen por defecto, esto tomará unos segundos en completarse la creación.

Figura 88

Ventana para crear un Realtime Database de Firebase.



Una vez creada la base datos obtendrá una URL única como se destaca en la Figura 89, este nos servirá más tarde para que la aplicación web, el ESP32 o el ESP8266 interactúen con la base de

datos, el tipo de información que almacena este tipo de base de datos es en formato JSON, por lo que a continuación se explicara los campos necesarios que se tendrá para el funcionamiento del proyecto.

Figura 89

Ventana para crear un Realtime Database de Firebase.

En la Figura 90 que se ve a continuación, se muestra la realización del objeto JSON con la estructura de cómo guardaremos los datos del proyecto, unos de los objetivos es que el proyecto se pueda usar en diferentes casas es por ello que el objeto JSON contendrá diferentes nodos como: *Casa1*, *Casa2* y así hasta el número de casa que se desee, de tal manera que si un usuario pertenece a la Casa1 solo podrá leer y editar el nodo que le corresponde, dentro de cada nodo contiene los datos de todos los componentes a usar como las salidas digitales de los leds y datos de los sensores como la distancia, humedad, luminosidad y temperatura.

Figura 90

Formato del objeto Json a usar en la base de datos.

```
{
  "Casa1": {
  "outputs": {
   "digital": {
   "led1": false,
    "led2": true,
    "led3": false
        }
      },
    "sensor": {
   "distancia": 34,
   "humedad": 52.5,
"luminosidad": "Dia",
  "temperatura": 75
        }
      },
    "Casa2": {
        ...
      }
    }
```

Algo importante que debemos obtener es la clave de autorización para poder conectarnos desde nuestras tarjetas, para ello damos clic en configuración del proyecto como se muestra en el paso de la Figura 91 y después se elige la opción de cuentas de servicios.

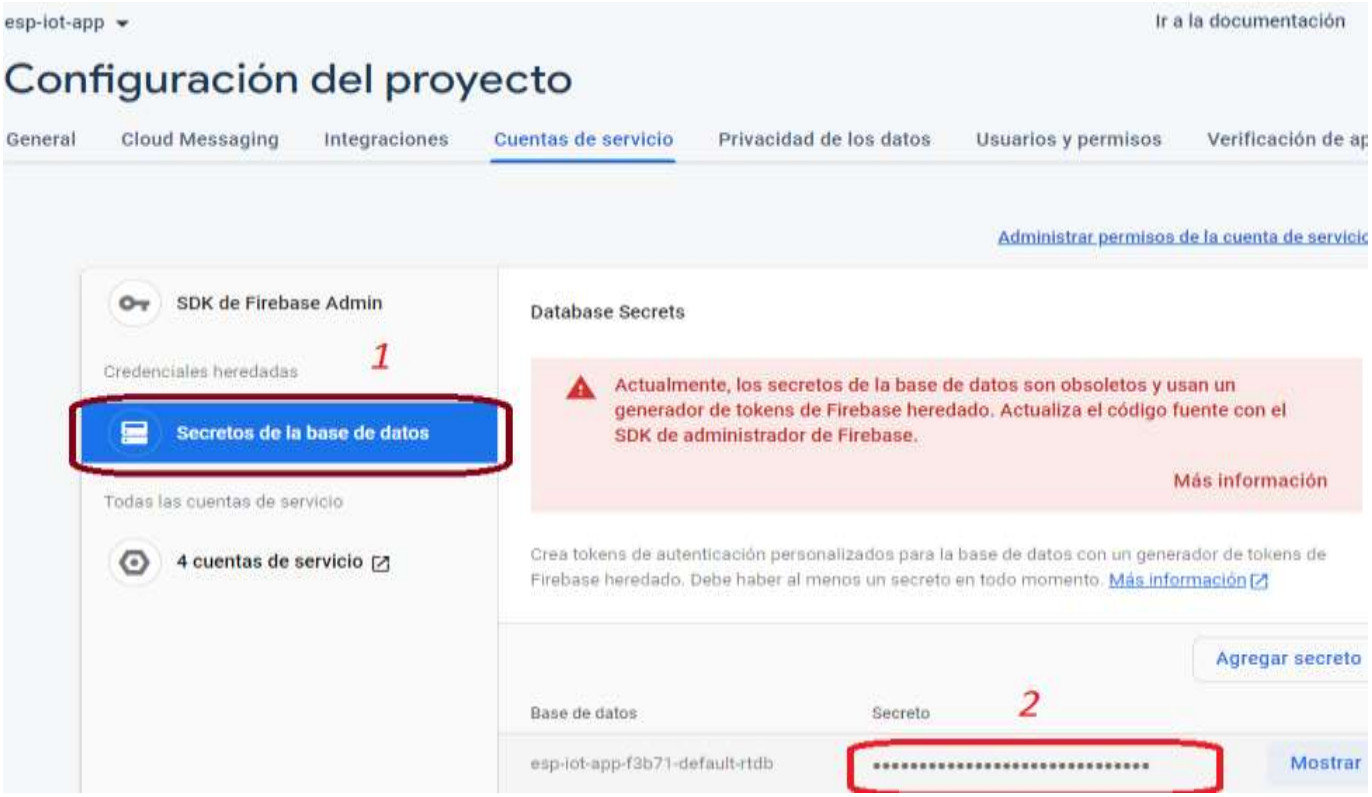
Figura 91

Obtener clave secreta del proyecto de Firebase.

Esto nos llevará a la siguiente ventana como se aprecia en la Figura 92, aquí primeramente seleccionamos servicios de la base de datos y se nos carga el api key como se resalta como punto 2 en la figura, esta clave la copiamos y guardamos ya que se utilizará al momento de programar la tarjeta para el uso del servicio de Firebase.

Figura 92

Ventana para obtener la clave secreta de la base de datos.



Interactuando con la Base de Datos Realtime de Firebase usando el Esp32 y Esp8266.

En esta sección se explicará la configuración a realizar a las 3 placas para poder leer y editar registros de nuestra base de datos creada previamente, en las próximas tablas 1 y 2 que se mostrarán a continuación serán las mismas líneas de códigos que contendrán en la programación de cada placa, es por eso que solo se colocará explicando una vez y se hará referencia en cada en los sketches de programación en Arduino.

Dentro de la tabla 59 a partir de las líneas del 2 al 7 se declaran constantes en la cual se debe reemplazar con las credenciales de la red WIFI a conectar y las credenciales del proyecto Firebase, luego en la línea 10 se usa el objeto firebaseData de la librería de Firebase, este objeto nos ayudará a leer y actualizar algún dato, además en la línea 11 se usa el objeto stream que es un listener, este verifica si existe algún cambio en la base de datos y podamos hacer alguna acción al respecto.

Cada tarjeta contará con un temporizador para que cada 5 minutos actualice el dato de lectura de su sensor en la base de datos de Firebase, esta configuración se declara en las líneas 14 y 16 de la tabla 59, finalmente en la línea 19 se apunta en el nodo de la casa en que se trabajara.

Tabla 59

Fichero con variables de configuración y credenciales.

	config.h
1	// Credenciales de la red WIFI
2	#define WIFI_SSID "TU_SSID_WIFI"
3	#define WIFI_PASSWORD "TU_PASSWORD"
4	
5	//Credenciales del Proyecto Firebase
6	#define FIREBASE_HOST "COLOCAR "
7	#define FIREBASE_AUTH "COLOCAR TU TOKEN SECRETO"
8	
9	//Firebase Data object
10	FirebaseData firebaseData;
11	
12	FirebaseData stream;
13	
14	
15	//Almacena la ultima vez que hubo actualización
16	
17	unsigned long previousMillis = 0;
18	
19	//El intervalo para la actualización es de 5mint
20	
21	const long interval = 300000; //valor en milisegundo
22	
23	

```
18  //Nodo de la casa en especifico
19
20  String nodo = "/Casa1";
21
```

En la tabla 60, contiene un sketch ya familiar y que hemos usado mucho en prácticas anteriores, que consiste en un método para poder conectarnos hacia una red WIFI, esto nos ayudará mucho para poder hacer uso de internet y utilizar los servicio de Firebase.

Tabla 60

Fichero con función para conectar a una red WIFI.

connect_WIFI.hpp

```
1   void ConnectWiFi_STA(){
2   Serial.println("");
3   //conectarse a red WIFI en modo estación
4   WiFi.mode(WIFI_STA);
5   WiFi.begin(WIFI_SSID, WIFI_PASSWORD);
6
7   while (WiFi.status() != WL_CONNECTED){
8   delay(100);
9   Serial.print('.');
10  }
11
12
13  Serial.println("");
14
15  Serial.println("Direcciòn IP:\t");
16
17  Serial.println(WIFI.localIp());
18
```

En la tabla 60 se puede apreciar el sketch principal de la primera placa que es un ESP32, aquí se incluyen unas librerías como la de WIFI, también la librería ArduinoJson debido que los datos que recibimos de Firebase es en formato JSON y es necesario deserializar el mensaje que se reciba y la última librería incluida es FirebaseESP32 que nos ayudará interactuar con la base de datos de Firebase, además se incluyen los ficheros config.h y WifiConnect.hpp que contiene unas variables de configuración y un método para conectarnos a la red WIFI que ya se explicaron previamente en la tabla 59 y 60.

En la sección de setup se inicializan los pines del sensor de distancia dht11, un led, se hace una llamada al método de conexión a la red WIFI e inicializa la librería de Firebase.

Ahora en la sección de código del void loop, se inicia obteniendo el tiempo transcurrido en milisegundo, esto no servirá para controlar que cada 5 minutos se realice una petición para actualizar el dato de lectura del sensor de distancia en Firebase esto se hace a partir de las líneas del 51 al 64, en las demás instrucciones de código consiste en verificar si existe algún cambio en la base de datos debido alguna petición que realice algún cliente y recibir dicho mensaje para realizar la acción requerida.

Tabla 61

Sketch principal de la placa ESP32.

	Esp32-sensor-distancia.ino
1	#include <**WiFi**.h>
2	#include <ArduinoJson.h>
3	#include "FirebaseESP32.h"
4	
5	#include "config.h"
6	#include "WifiConnect.hpp"
7	
8	byte pinLed1 = 21;
9	bool estadoLed1 = false;
10	#define Pecho 18
11	#define Ptrig 19
12	long duracion, distancia;

```
13
14  void setup(){
15   pinMode(pinLed1, OUTPUT);
16   pinMode(Pecho, INPUT);  // define el pin 18 como entrada (echo)
17   pinMode(Ptrig, OUTPUT); // define el pin 19 como salida
     (trigger)
18
19   ConnectWiFi_STA();
20
21   Firebase.begin(FIREBASE_HOST, FIREBASE_AUTH);
22   //suscribirse a los cambios de datos en un nodo definido.
23   if (!Firebase.beginStream(firebaseData, nodo)){
24     Serial.println(firebaseData.errorReason());
25   }
26   Firebase.reconnectWiFi(true);
27  }
28
29  void loop(){
30   //Obtiene el tiempo transcurrido en milisegundo
31   unsigned long currentMillis = millis();
32
33   //Devuelve true cuando hay nuevos datos de transmisiòn
     disponible
34   if (firebaseData.streamAvailable()){
35     FirebaseJson *json = firebaseData.to<FirebaseJson *>();
36     String mensaje = json->raw();
37    //Comprobamos si hubo cambio en la salida digitales
38    if(firebaseData.dataPath() == "/outputs/digital"){
39       StaticJsonDocument<32> doc;
40       deserializeJson(doc, mensaje);
41       //comprobamos si hubo cambio en el led1
42       int index = mensaje.indexOf("led1");
```

```
43              if(index>=0) estadoLed1 = doc["led1"];
44            }else if(firebaseData.dataPath() == "/"){
45              StaticJsonDocument<256> doc;
46              deserializeJson(doc, mensaje);
47              JsonObject outputs_digital = doc["outputs"]["digital"];
48              estadoLed1 = outputs_digital["led1"];
49            }
50          }
51          //cada 5mint actualizamos el valor del sensor de distancia
52          if (currentMillis - previousMillis >= interval) {
53            digitalWrite(Ptrig, LOW);
54            delayMicroseconds(2);
55            digitalWrite(Ptrig, HIGH);
56            delayMicroseconds(10);
57            digitalWrite(Ptrig, LOW);
58
59            //Obtener valor de distancia
60            duracion = pulseIn(Pecho, HIGH);
61            distancia = (duracion/2) / 29; // calcula la distancia en cm
62            Firebase.setInt(firebaseData, nodo + "/sensor/distancia",
          distancia);
63
64            previousMillis = currentMillis;
65          }
66          //Actualizamos el estado del led
67          digitalWrite(pinLed1, estadoLed1);
68          }
```

De aquí en adelante la mayor parte de código es similar al código que se analizó en la tabla 60 sobre la programación de la placa ESP32, debido que el funcionamiento es igual, en la tabla 61 en las primeras líneas se incluye las mismas librerías, pero unas basadas precisamente para la placa

ESP8266, dentro de la sección del void setup se inicializan los pines como el del sensor de luminosidad, el pin del led e igualmente se inicializa la librería de firebase.

Por otro lado, en la sección del void loop no existe diferencia alguna a la configuración anterior, se escucha aquellos cambios que se realice en firebase y cada 5 minutos se actualiza el dato del sensor en este caso el de la luminosidad.

Tabla 62

Sketch principal de la placa ESP8266 con sensor de luminosidad.

	ESP8266_sensor_luminosidad.ino
1	#include <ArduinoJson.h>
2	#include <ESP8266WiFi.h>
3	#include <FirebaseESP8266.h>
4	
5	#include "config.h"
6	#include "Connect_Wifi.hpp"
7	
8	byte pinLed2 = 0;
9	bool estadoLed2 = false;
10	
11	//pin del sensor luminosidad
12	
13	const int analogPin = 4;
14	
15	String valorLDR;
16	
17	
18	void setup(){
19	
20	pinMode(pinLed2, OUTPUT);
21	

```
    ConnectWiFi_STA();

    //Inicialice la biblioteca con la configuración de Firebase.
    Firebase.begin(FIREBASE_HOST, FIREBASE_AUTH);

    //suscribirse a los cambios de datos en un nodo definido.
    if (!Firebase.beginStream(firebaseData, nodo)){
      Serial.println(firebaseData.errorReason());
    }
    Firebase.reconnectWiFi(true);
  }

  void loop(){
    //Obtiene el tiempo transcurrido en milisegundo
    unsigned long currentMillis = millis();
```

```
    //Devuelve true cuando hay nuevos datos de transmisiòn disponible

    if (firebaseData.streamAvailable()){

      FirebaseJson *json = firebaseData.to<FirebaseJson *>();

      String mensaje = json->raw();



     //Comprobamos si hubo cambio en la salida digitales

     if(firebaseData.dataPath() == "/outputs/digital"){

        StaticJsonDocument<32> doc;

        deserializeJson(doc, mensaje);

        //comprobamos si hubo cambio en el led2

        int index = mensaje.indexOf("led2");

        if(index>=0) estadoLed2 = doc["led2"];



     }else if(firebaseData.dataPath() == /"){

        StaticJsonDocument<256> doc;

        deserializeJson(doc, mensaje);
```

```
            JsonObject outputs_digital = doc["outputs"]["digital"];

            estadoLed2 = outputs_digital["led2"];

        }

    }

    //cada 5mint actualizamos el valor del sensor en Firebase

    if (currentMillis - previousMillis >= interval) {

      valorLDR = (analogRead(analogPin) > 1000) ? "Dia" : "Noche";

     Firebase.setString(firebaseData, nodo + "/sensor/luminosidad",
    valorLDR);

      previousMillis = currentMillis;

    }

    digitalWrite(pinLed2, estadoLed2);

  }
```

Finalmente en la programación de la última tarjeta ESP8266 sigue la misma lógica de las tarjetas anteriores, en este caso se usa un sensor de temperatura y humedad dht11, por lo que el mayor cambio será implementar la lógica para hacer uso de este sensor, para ello se incluye su librería

‘DHT.h’ y hacemos uso de sus métodos readTemperature y readHumidity como se ejemplifica en las líneas 53 y 54 de la tabla 62, Tanto en la sección del setup y void loop se implementan la misma configuración ante vista.

Tabla 63

Sketch principal de la placa ESP8266 con sensor de temperatura y humedad.

	ESP8266_sensor_temperatura_humedad.ino
1	#include "DHT.h"
2	#include <ArduinoJson.h>
3	#include <ESP8266WiFi.h>
4	#include <FirebaseESP8266.h>
5	
6	#include "config.h"
7	#include "Connect_Wifi.hpp"
8	
9	byte pinLed3 = 5;
10	bool estadoLed3 = false;
11	
12	
13	float temperatura, humedad;
14	
15	DHT dht(2, DHT11);
16	
17	
18	void setup(){
19	
20	pinMode(pinLed3, OUTPUT);
21	
22	ConnectWiFi_STA();
23	

```
  //Inicializamos el sensor dht11

  dht.begin();

  //Inicialice la biblioteca con la configuración de Firebase.

  Firebase.begin(FIREBASE_HOST, FIREBASE_AUTH);

  if (!Firebase.beginStream(firebaseData, nodo)){

    Serial.println(firebaseData.errorReason());

  }

  Firebase.reconnectWiFi(true);

}


void loop(){

  //Obtiene el tiempo transcurrido en milisegundo

  unsigned long currentMillis = millis();


  //Devuelve true cuando hay nuevos datos de transmisiòn disponible

  if (firebaseData.streamAvailable()){
```

```
    FirebaseJson *json = firebaseData.to<FirebaseJson *>();

    String mensaje = json->raw();



  //Comprobamos si hubo cambio en la salida digitales

  if(firebaseData.dataPath() == "/outputs/digital"){

     StaticJsonDocument<32> doc;

     deserializeJson(doc, mensaje);

     //comprobamos si hubo cambio en el led3

     int index = mensaje.indexOf("led3");

     if(index>=0) estadoLed3 = doc["led3"];

  }else if(firebaseData.dataPath() == /"){

     StaticJsonDocument<256> doc;

     deserializeJson(doc, mensaje);

     JsonObject outputs_digital = doc["outputs"]["digital"];

     estadoLed3 = outputs_digital["led3"];

  }
```

```
    }

    //cada 5mint actualizamos el valor del sensor dht11 en Firebase

    if (currentMillis - previousMillis >= interval) {

      temperatura = dht.readTemperature(); //Leemos la temperatura

      humedad = dht.readHumidity();   //Leemos la humedad

      Firebase.setInt(firebaseData, nodo + "/sensor/humedad", humedad);

      Firebase.setInt(firebaseData, nodo + "/sensor/temperatura",
    temperatura);


      previousMillis = currentMillis;

    }

    //Actualizamos el estado del led

    digitalWrite(pinLed3, estadoLed3);

    }
```

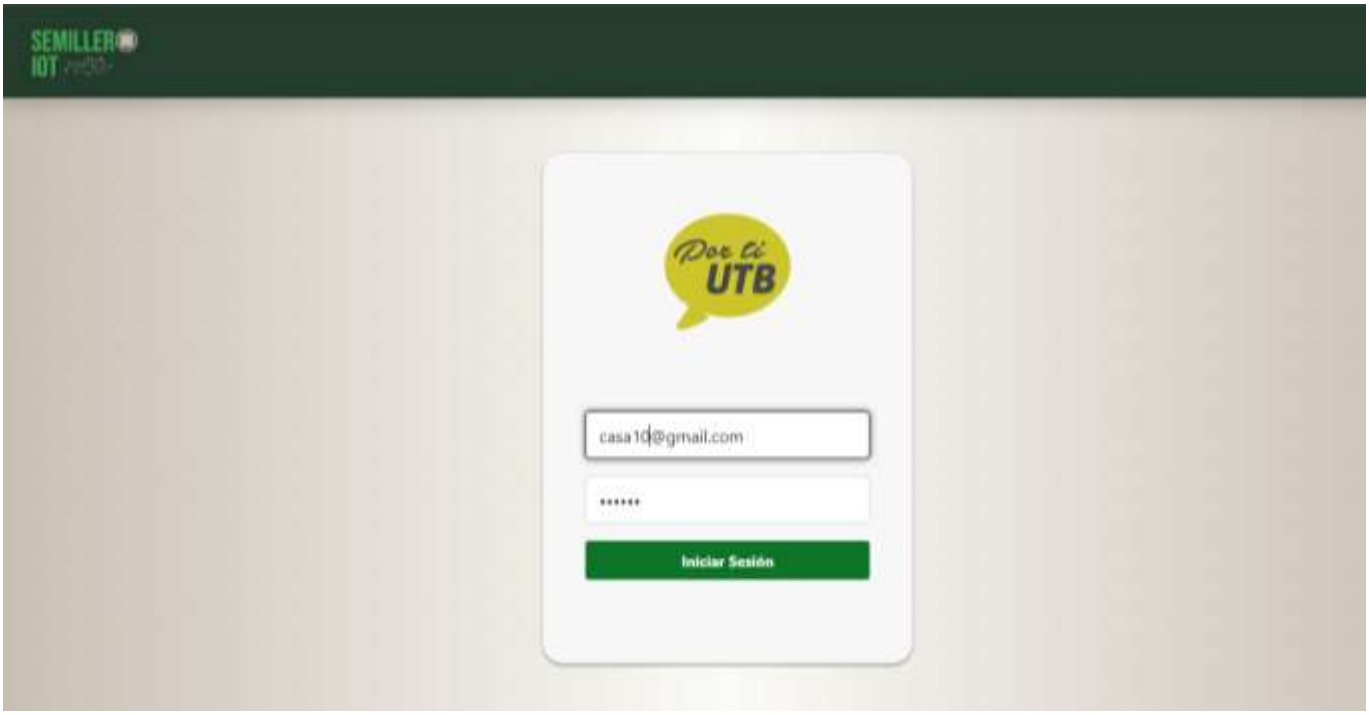
Creación de la Aplicación Web haciendo uso de Firebase

Ahora se comenzará a desarrollar la aplicación web para que cualquier cliente pueda controlar las placas que se programaron previamente, la aplicación web se la construirá haciendo uso del framework de JavaScript Vuejs para realizar nuestras interfaces, como primer paso todo usuario debe

iniciar sesión, la interfaz para autenticarse se aprecia en la Figura 93, en donde deberá colocar un correo y una contraseña para su validación.

Figura 93

Página de inicio de Sesión



Para realizar la validación de inicio de sesión se lo hará haciendo peticiones http al Api Rest de Firebase mediante la librería axios, para instalar esta librería ejecutamos el comando *npm install axios* en nuestro proyecto, en un archivo de JavaScript realizamos la configuración que se aprecia en la tabla 64, esto nos ayudará para aplicar una URL base del api rest y además es necesario colocarle como parámetro el apiKey del proyecto de Firebase, esta clave la podemos conseguir en la configuración del proyecto.

Tabla 64

Configuración de axios para la autenticación de usuarios.

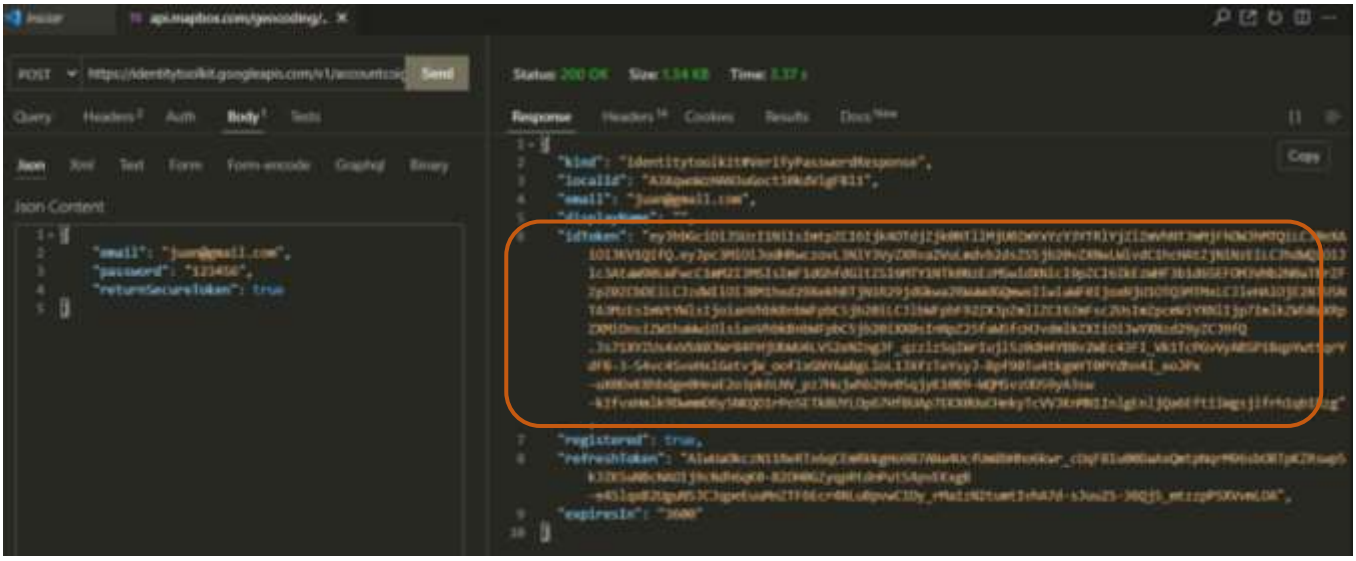
	authApi.js
1	import axios from 'axios'
2	
3	const authApi = axios.create({

```
4     baseURL: 'https://identitytoolkit.googleapis.com/v1/accounts',
5     params: {
6       key: 'AIzaSyAjmIbqpCYTxaYMNxCTID9ZSiLbKHe1nkQ'
7     }
8   })
9   export default authApi
```

Una característica que deben contar los usuarios es estar asignados a alguna casa en específica, de tal manera que al iniciar sesión lo enviará al panel administrativo mostrando los datos del nodo respectivo. Para conseguir esta funcionalidad se lo hará seteando la propiedad displayName del usuario de la siguiente manera: “nombres_usuario | Casa#”, de este modo con JavaScript podemos separar esta cadena y obtener el nombre del usuario y el número de la casa que pertenece.

Figura 94

Petición HTTP de autenticación de usuario.



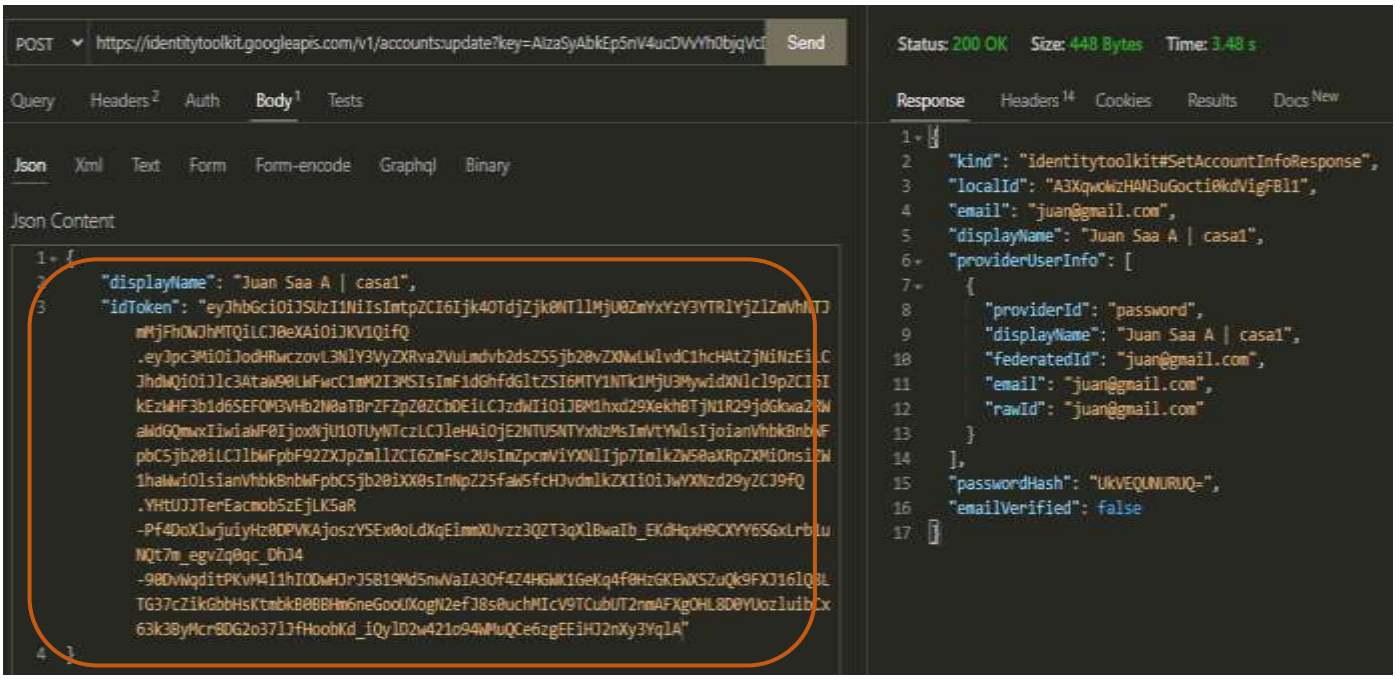
Primeramente, para cambiar el displayName se necesita el idToken que nos devuelve Firebase al momento de autenticarse como se resalta en el ejemplo realizado en la Figura 94, en esa figura se realiza la prueba de inicio de sesión realizando una petición al api rest de Firebase enviando los campos, email, password y returnSecureToken hacia la siguiente ruta: https://identitytoolkit.googleapis.com/v1/accounts:signInWithPassword?key=COLOCAR_KEY

Si los datos enviados son correctos recibimos como respuesta un objeto JSON con varios campos, el cual el que necesitamos y guardamos será el idToken.

Una vez que ya contamos con el idToken solo falta realizar una petición más enviando los campos de displayName y idToken hacia la siguiente ruta: *https://identitytoolkit.googleapis.com/v1/accounts:update?key=XXXX*

Figura 95

Petición HTTP para setear el displayName de algún usuario.



Revisando la tabla 65 se aprecia el componente de autenticación, en las primeras líneas se encuentra el template con los elementos HTML principalmente el formulario que es donde el usuario podrá ingresar sus datos, los inputs poseen el atributo de vuejs llamado v-model que se usa para crear bindings con el objeto user que retorna el script en la línea 44.

Al realizarse el envío del formulario se acciona el método submit que se encuentra en la línea 45, este método es asíncrono debido que debe esperar la respuesta de validación por parte de Firebase, si la respuesta es positiva recibiremos varios datos como el idToken, refreshToken y displayName, estos datos se lo guarda en el localStorage para su posterior uso, en el campo de displayName se extrae el número del hogar al que pertenece el usuario con el fin de trabajar en el nodo del objeto JSON que le corresponde.

Tabla 65

Componente de login con su template y script.

	login.vue
1	<template>
2	
3	<img id="profile-img" class="profile-img-card"
4	src="/img/porti_utb.png"
5	/>
6	
7	
8	<form @submit.prevent="submit" class="form-signin">
9	
10	
11	<input type="email" v-model="user.email"
12	placeholder="Correo electronico"
13	required autocomplete="off" class="form-control" />
14	
15	<input type="password" v-model="user.password"
16	placeholder="Contraseña" class="form-control mt-3"
17	required />
18	
19	<button

```
                class="btn btn-lg btn-primary btn-block btn-signin mt-3"
                type="submit"
            >
                Iniciar Sesión
            </button>
        </form>


    </div>
</template>


<script>
import { ref } from 'vue';
import Swal from 'sweetalert2';
import authApi from '@/api/authApi'
export default {
    setup(){
```

```
const user = ref({
  email: '',
  password: '',
  returnSecureToken: true
})

return {
  user,
  submit: async () => {
    try {
      const { data: { idToken, refreshToken, displayName } } = await authApi.post(':signInWithPassword', user.value)
      //Obtener numero de hogar del usuario
      let hogar = displayName.split('|')[1].trim();

      localStorage.setItem('idToken', idToken)
```

```
                localStorage.setItem('refresh', refreshToken)

                localStorage.setItem('hogar', hogar)

                //redireccionamos al panel administrativo

                window.location.href = '/dashboard';

            } catch (error) {

                Swal.fire('Error', error.response.data.error.message, 'error')

            }

        }

    }

};

</script>
```

Una vez que el usuario haya iniciado sesión se necesita interactuar con Firebase para leer datos o realizar alguna modificación que se necesite, para ello instalamos en nuestro proyecto la librería de firebase con el siguiente comando: *npm install firebase* y creamos un archivo JavaScript y colocamos la configuración que tenemos en la tabla 8 reemplazando con los datos de nuestro proyecto, con esto podemos interactuar de mejor manera con nuestra base de datos.

Tabla 66

Configuración de la librería Firebase

firebase.js

```
1   import { initializeApp } from "firebase/app";
2   import { getDatabase, ref } from "firebase/database";
3   //Reemplazar con la configuración de su proyecto
4   const firebaseConfig = {
5     apiKey: "AIzaSyAjmIbqpCYXXXXXXXXXXXX",
6     authDomain: "esp-iot-app-df933.firebaseapp.com",
7     databaseURL: "https://esp-iot-app-df93-XXXXXXXXX",
8     projectId: "esp-iot-app-df933",
9     storageBucket: "esp-iot-app-df933.appspot.com",
1     messagingSenderId: "407XXXXXXXXX",
0
1     appId: "1:407506270625:web:XXXXXXXXXXX"
1
1   };
2
1   //Inicializa Firebase
3
1   initializeApp(firebaseConfig);
4
1   //Obtener # de hogar del usuario autenticado
5
1   const casa = localStorage.getItem('hogar');
6
1   const db = getDatabase();
7
1   //Se apunta al nodo que se va a leer o editar los datos
8
1   const starCountRef = ref(db, `${casa}/`);
```

```
9
2
0
2       export default starCountRef;
1
```

Si el usuario inicio sesión exitosamente se lo enviara automáticamente a la página con la interfaz que se visualiza en la Figura 96, en esta vista se nos muestra la lectura de todos los sensores que usan nuestras tarjetas, cabe resaltar que los datos de lectura corresponden al número del hogar del cual es el usuario.

Figura 96

Vista de monitoreo de los sensores de las placas ESP32 y ESP8266

Ahora se analizará la lógica usada en la vista de sensores, esto lo encontramos en la tabla 67 que se muestra a continuación, en las primeras líneas del 1 al 7 se hace varias importaciones como librerías de animación para representar con un mejor diseño los datos de los sensores, se importa también el archivo de configuración de Firebase realizado anteriormente y se usa vuex para manejar nuestro gestor de estado globales.

En el bloque de código del setup podemos notar los métodos montarChartHumedad y montarChartTemperature estos hacen uso de las librerías de diseño y representar un gráfico con el valor tomado de la base de datos de Firebase, el método que realiza la consulta en el nodo de la casa del usuario se encuentra a partir de la línea 51 llamado firebase, esta función se ejecuta al cargar el componente y es un oyente en el cual también se acciona cuando exista algún cambio en el nodo de la casa que se esté autenticado, de esta manera podemos representar en tiempo real alguna actualización de lectura de los sensores de las tarjetas ESP32 o el ESP8266.

Tabla 67

Código JavaScript del componente sensores.vue

	sensores.vue
1	import { ref, computed, onMounted } from 'vue';
2	import { useStore } from 'vuex'
3	import starCountRef from '@/utils/firebase';
4	import { onValue } from "firebase/database";
5	import JustGage from "justgage";
6	import * as echarts from "echarts/core";
7	import "echarts-liquidfill";
8	
9	export default {
10	setup(){
11	const store = useStore()
12	
13	const liquid = ref(null);
14	
15	let valueTemperature = ref(0)
16	
17	let valueHumedad = ref(0)
18	
19	let chartLiquid = null;
20	

```
let gauge = null;


onMounted(() => {
  chartLiquid = echarts.init(liquid.value);

  montarChartTemperature(0);
  gauge.destroy()
  montarChartTemperature(0);

  firebase();
});

const montarChartHumedad = () => {
  chartLiquid.setOption({
    series: [{
      type: "liquidFill", data: [valueHumedad.value / 100]
```

```
32          }]
33
34        });
35
36      }
37
38
39      const montarChartTemperature = (temperatura) => {
40
41        gauge = new JustGage({
42
43          id: 'gauge',
44
45          value: temperatura,
46
47          min: 0,
48
49          max: 100,
50
51          symbol: '°',
52
53          pointer: true,
54
55          gaugeWidthScale: 0.6,
56
57          counter: true,
58
59          labelFontColor: '#ffffff',
60
61          valueFontColor: '#ffffff',
62
```

```
48          });
49
50      }
51
52
53      const firebase = () => {
54
55          onValue(starCountRef, (snapshot) => {
56
57
58              const data = snapshot.val();
59
60              store.commit('auth/updateIotData', data)
61
62
63
64  const humedad = computed (() => store.state.auth.sensor.humedad)
65
66  const temperatura = computed (() => store.state.auth.sensor.temperatura)
67  const distancia = computed (() => store.state.auth.sensor.distancia)
68
69  const luminosidad = computed (() => store.state.auth.sensor.luminosidad)
70
71
72              document.getElementById('count-box').innerHTML = `${distancia.value} cm`
73              document.getElementById('txtLuminosidad').innerHTML = `${luminosidad.value}`
```

```
64
                valueHumedad.value = humedad.value
65
                montarChartHumedad();
66
67
                gauge.refresh(temperatura.value);
68
              })
69
            }
70
71
            return{
72
              liquid, valueTemperature, valueHumedad
73
            }
74
          }
75
        }
76
```

Otra vista a la que podemos acceder es para el control de las luces led que tenemos conectados a nuestras tarjetas, la interfaz se apreciar en la Figura 97, al dar clic para encender o apagar algún led se realiza la actualización del estado del led a Firebase y la tarjeta a la cual esté conectado dicho led detectara automáticamente ese cambio y encenderá o apagara el led según el nuevo estado.

Figura 97

Vista de monitoreo de las luces leds de las placas ESP32 y ESP8266

Podemos notar en la tabla 10 el script del componente luces.vue, en las primeras líneas se realiza unas importaciones de librerías necesarias, en la sección del setup se declara principalmente 3 constantes que estarán asociados a los inputs que representan a los leds, al dar clic en algún input se llama al método updateStatusLed que se encuentra a partir de la línea 30 y recibe como parámetro que led ha dado clic cambiando su estado y realizando una petición HTTP al Api Rest de Firebase seteando el estado del led al nodo de la casa correspondiente.

Tabla 68

Código JavaScript del componente luces.vue

	luces.vue
1	import { computed, ref } from 'vue';
2	import { useStore } from 'vuex'
3	import starCountRef from '@/utils/firebase';

```
import { onValue } from "firebase/database";
import dashboardApi from '@/api/dashboardApi'

export default {
  setup(){
    const store = useStore()
    const led1 = ref(false);

    const led2 = ref(false);

    const led3 = ref(false);



    onValue(starCountRef, (snapshot) => {

      const data = snapshot.val();

        store.commit('auth/updateIotData', data)

const fireled1 = computed (() => store.state.auth.outputs.digital.led1)

const fireled2 = computed (() => store.state.auth.outputs.digital.led2)

const fireled3 = computed (() => store.state.auth.outputs.digital.led3)

        led1.value = fireled1.value

        led2.value = fireled2.value

        led3.value = fireled3.value

```

```
})


return{

  led1,

  led2,

  led3,


  updateStatusLed: async (element) => {

    let config = {};

    switch (element) {

      case 'led1':

        led1.value = !led1.value

        config = { led1:  led1.value}

        break;

      case 'led2':

        led2.value = !led2.value

```

```
            config = { led2:  led2.value}

            break;

          case 'led3':

            led3.value = !led3.value

            config = { led3:  led3.value}

            break;

        }

        const casa = store.state.auth.hogar

        await dashboardApi.patch(`/${casa}/outputs/digital.json`, config)

      }

    }

  }
```

Bibliografía

Alzate, O. F. (20 de Octubre de 2019). *codigo electronica*. Obtenido de codigo electronica.: http://www.codigoelectronica.com/blog/libreria-dht-arduino#:~:text=Retorna-,DHT%20sensor,en%20cualquier%20proyecto%20con%20arduino.

aosong. (s.f.). Obtenido de aosong: http://www.aosong.com/en/products-32.html

Arduino, P. c. (s.f.). *Proyectos con arduino*. Obtenido de https://proyectosconarduino.com/curso/variables-globales-en-arduino/

Arduino.cl. (s.f.). *Arduino*. Obtenido de Arduino: https://arduino.cl/que-es-arduino/

Arduino.cl. (s.f.). *Ingenieria MCI*. Obtenido de https://arduino.cl/introduccion-a-los-tipos-de-dato-con-arduino/

B. Global. (s.f.). *Bosch Sensortec*. Obtenido de Bosch Sensortec: https://www.bosch-sensortec.com/products/environmental-sensors/pressure-sensors/bmp388/

Bosch Sensortec. (s.f.). Obtenido de Bosch Sensortec: https://www.bosch-sensortec.com/products/environmental-sensors/gas-sensors/bme680/

codificada, C. (23 de Agosto de 2019). *Creatividad codificada*. Obtenido de https://creatividadcodificada.com/arduino/tipos-de-datos-y-operadores-en-arduino/

Custodio, E., & Wilfredo, C. (2016). *Universidad Ricardo Palma.* Obtenido de https://www.urp.edu.pe/pdf/id/5792/n/simulacion-e-instalacion-domotica-en-casas-para-el-control-de-seguridad-e-ilumacion

Damián, J. (25 de Enero de 2020). *Electro Geek.* Obtenido de https://www.electrogeekshop.com/como-usar-ficheros-json-en-arduino-con-arduino-json/

datasheetspdf. (2013). Obtenido de datasheetspdf: https://datasheetspdf.com/pdf/792210/ABCPROYECTOS/DHT11/1

Educative. (s.f.). *Educative*. Obtenido de https://www.educative.io/edpresso/what-is-a-cpp-struct

GARCIA, V. (23 de 08 de 2019). *HISPAVILA.COM*. Obtenido de https://www.hispavila.com/entendiendo-el-archivo-spiffs/

Geek factory. (24 de 11 de 2021). *Geek factory*. Obtenido de Geek factory: https://www.geekfactory.mx/tienda/sensores/bmp180-sensor-de-presion-atmosferica/

Geek Factory. (07 de 02 de 2022). *Geek Factory*. Obtenido de Geek Factory: https://www.geekfactory.mx/tienda/sensores/hc-sr04-sensor-de-distancia-ultrasonico/

Gerchev, I. (29 de Noviembre de 2018). *code.tutsplus*. Obtenido de https://code.tutsplus.com/es/tutorials/boost-your-vuejs-workflow-with-vue-cli-3--cms-32232

Herramientas tecnologicas profesionales. (s.f.). *HETPRO*. Obtenido de https://hetpro-store.com/TUTORIALES/arduino-string/

HeTPro. (s.f.). *HeTPro*. Obtenido de HeTPro: https://hetpro-store.com/mpu6050/

LLAMAS, L. (4 de Septiembre de 2019). *Luis Llamas Ingenieria, Informatica y diseño*. Obtenido de https://www.luisllamas.es/como-hacer-un-servidor-asincrono-en-esp8266/

LLamas, L. (19 de Agosto de 2019). *LuisLLamas*. Obtenido de https://www.luisllamas.es/como-usar-el-spiffs-del-esp8266-con-el-arduino-ide/

MACTRONICA. (2022). *MACTRONICA*. Obtenido de MACTRONICA: https://www.mactronica.com.co/sensor-de-movimiento-pir-mh-sr602-sr602

Manzanares, M. (22 de Febrero de 2021). *inlab.fib.upc*. Obtenido de inlab.fib.upc: https://inlab.fib.upc.edu/es/blog/uso-de-websockets-en-una-aplicacion-web

Microsoft Azure. (2020). *Microsoft* . Obtenido de Microsoft .

Moncayo, J. M. (26 de Octubre de 2017). *openwebinars*. Obtenido de openwebinars: https://openwebinars.net/blog/que-es-vuejs/

Naylamp Mechatronics. (s.f.). Obtenido de Naylamp Mechatronics: https://naylampmechatronics.com/sensores-posicion-inerciales-gps/357-sensor-de-presion-temperatura-y-humedad-bme280.html

Naylamp Mechatronics . (s.f.). *Naylamp Mechatronics - Perú*. Obtenido de Naylamp Mechatronics - Perú: https://naylampmechatronics.com/conversores-dc-dc/85-convertidor-voltaje-dc-dc-step-down-2a-mp1584en.html

naylampmechatronics. (s.f.). Obtenido de naylampmechatronics: https://naylampmechatronics.com/blog/46_tutorial-sensor-digital-de-temperatura-ds18b20.html

Rivera, D. (16 de 05 de 2020). *Pleets Blog*. Obtenido de https://blog.pleets.org/article/introducci%C3%B3n-al-lenguaje-c++

Sharma, A. (s.f.). *https://www.arduino.cc/*. Obtenido de https://www.arduino.cc/reference/en/libraries/asyncelegantota/

Simões, C. (27 de Julio de 2021). *itdo*. Obtenido de itdo: https://www.itdo.com/blog/que-es-node-js-y-para-que-sirve/

UNIT Electronics. (03 de Marzo de 2022). Obtenido de https://uelectronics.com/producto/sensor-magnetico-mc-38/

UNIT Electronics. (3 de marzo de 2022). *UNIT Electronics*. Obtenido de UNIT Electronics: https://uelectronics.com/producto/hlk-5m-ac-dc-convertidor-5w/

wordpress. (16 de Noviembre de 2016). *Aprendiendo Arduino.* Obtenido de https://aprendiendoarduino.wordpress.com/2016/11/16/funciones-definidas-por-usuario-2/

UNIVERSIDAD TÉCNICA DE BABAHOYO

ISBN: 978-9942-606-19-8