# Construcción de un prototipo de bajo costo para medición de señales y niveles de contaminación acústica

Miguel Giovanny Molina Villacís
Ximena Carolina Acaro Chacón
María Fernanda Molina Miranda
Pietro Corapi
Luis Arturo Espín Pazmiño
Joselyn Stefania Quimis Suárez

Savez editorial


# Construcción de un prototipo de bajo costo para medición de señales y niveles de contaminación acústica

# Construcción de un prototipo de bajo costo para medición de señales y niveles de contaminación acústica

Miguel Giovanny Molina Villacís
Ximena Carolina Acaro Chacón
María Fernanda Molina Miranda
Pietro Corapi
Luis Arturo Espín Pazmiño
Joselyn Stefania Quimis Suárez

Savez editorial

Miguel Giovanny Molina Villacís
Ximena Carolina Acaro Chacón
María Fernanda Molina Miranda
Pietro Corapi
Luis Arturo Espín Pazmiño
Joselyn Stefania Quimis Suárez

Construcción de un prototipo de
bajo costo para medición de señales
y niveles de contaminación acústica

ISBN: 978-9942-603-28-9

Savez editorial
**Título:**

Construcción de un prototipo de
bajo costo para medición de señales
y niveles de contaminación acústica

Primera Edición: Diciembre 2021

**ISBN:** **978-9942-603-28-9**
Obra revisada previamente por la modalidad doble par ciego, en caso de requerir información sobre el proceso comunicarse al correo electrónico
editor@savezeditorial.com

## DEDICATORIA

El presente libro se lo dedica a la comunidad académica que busca siempre trabajar en proyectos innovadores utilizando conocimientos de Tecnologías de la Información

AGRADECIMIENTO

Agradecemos a Dios y a nuestras familias por su apoyo incondicional en el desarrollo académico basado en la Investigación y a todas las personas que han aportado con su conocimiento para la realización de este libro

# ÍNDICE GENERAL

## ABREVIATURAS

| | |
|---|---|
| OMS | Organización Mundial de la Salud |
| UE | Unión Europea |
| dB | Decibelios |
| SoC | System on Chip |
| HTTPS | Protocolo de Transferencia de Hipertexto |
| WSS | Seguridad del Servidor Web |
| GPIO | Entrada / Salida de uso general |
| MQTT | Transporte de Telemetría de cola de Mensajes |
| SGBD | Servidor de Gestion de Base de Datos |
| SPA | Aplicación de Página Única |
| UTC | Tiempo Universal Coordinado |
| UIT | Union International de Telecomunicaciones |

SIMBOLOGÍA

ÍNDICE DE CUADROS

ÍNDICE DE GRÁFICOS

# CAPÍTULO I

## INTRODUCCIÓN

El ambiente de ruidos o vibraciones, generan incomodidad y malestar en las vidas cotidianas de las personas, la perpetración de la modernidad ligada a la estructura social, ha confrontado grandes efectos ambientales, salud y calidad de vida. Se pueden encontrar estudios que analizan y demuestran una clara relación entre los niveles altos de ruido y enfermedades en la población. Aunque varias Organizaciones como la OMS y la UE han impulsado para el control y reducción del ruido ambiental, aún existe una clara falta de atención por parte del sector político y las administraciones responsables de establecer medidas.

Millones de personas conviven con altos niveles de ruido, especialmente en las grandes ciudades con ruidos que sobrepasan los 65 dB, el umbral establecido por la OMS, varios estudios revelan que la exposición de altos niveles de ruido puede provocar alteraciones del sueño, pérdida auditiva, problemas como estrés, ansiedad, problemas cardiovasculares y problemas de aprendizaje. Las fuentes que producen ruidos dentro de la vida cotidiana, son las industrias, los medios de transportes y las grandes vías de comunicación.

Según las estadísticas de la OMS, más del 5% de la población mundial representado con 466 millones de personas padece algún problema auditivo, donde muchas veces son por causas adquiridas y no congénitas, como la exposición al ruido excesivo, por ejemplo, entornos laborales en los que se trabaja con maquinaria ruidosa o se producen explosiones, uso de aparatos de audio personales durante tiempos prolongados o bares discotecas y acontecimientos deportivos.
Este fenómeno afecta a estudiantes, que se encuentran en centros de Educación.

# CAPÍTULO II

# MARCO TEÓRICO

## ANTECEDENTES

El crecimiento de las grandes ciudades y el incremento de las actividades que se desarrollan en los núcleos urbanos han ocasionado una contaminación acústica afectando el descanso, estudio, actividades laborales o daño a las personas y el ambiente. Según (Verde, 2017)España es uno de los países más ruidosos, por lo que la causa se debe a la mala planificación y gestión para evitar este tipo de contaminación que es peligrosa al vivir expuesto a un nivel de ruido superior a las condiciones normales. (p.110) por lo que es importante concientizar a las personas a tomar medidas para disminuir la contaminación acústica.

En la actualidad existen tecnologías para medir el ruido dentro del aula como Betzol semáforo con control de ruido, Too Noisy Pro muestra de manera gráfica y divertida el nivel de ruido de fondo del aula o el espacio en que se encuentre, Noise Monitor- My Class Rules monitoriza el nivel de ruido del aula en decibeles, los centros de educación de España han instalados como precaución medidores en forma de aplicaciones, ya que apunta entre 30 y 40% de los estudiantes padecen algún grado de pérdida auditiva. (Ferreyra et al., 2019, p.244)

Para el desarrollo de este trabajo, ha habido varios estudios en lo que respecta a la contaminación acústica en la cual se puede verificar los diferentes problemas que causa en el área de educación y las soluciones en la cual el uso de la tecnología es primordial para brindar mejor calidad de aprendizaje y enseñanza.

Prototipo experimental para la monitorización de ruido en tiempo real en entornos urbanos propuso la utilización de redes de sensores inalámbricos para la elaboración de los mapas de ruido. Las redes de sensores inalámbricas están formadas por un número importante de nodos sensores, dispositivos de bajo coste con capacidad restringida de proceso de información y comunicación, y alimentados normalmente por baterías. (Friedrich et al., 2017, p.199)

Monitoreo de ruidos en sitios cerrados, en este proyecto se implementó un sistema de medición de emisores de ruido desde un sensor con el fin

de aplicarlo a la biblioteca. Para ello, se emplea un sensor de ruido, los cuales son calibrados y se pasa así a realizar las mediciones correspondientes. Luego de la toma de información de los sensores y mediante comunicación física con una interfaz de usuario, el dispositivo envió los datos correspondientes, los cuales posteriormente se analizan. (Leyva Perez, 2015, p.97)

Sistema de medición acústica usando NODEMCU ESP8266 para determinar el ruido en la av. Víctor Larco 14 Trujillo, 2018, se construyó un dispositivo físico moderno y de bajo costo, en versión de prototipo en referencia a los existentes para dichas mediciones. El hardware incluyo partes orientadas de la nueva tecnología IoT denominado NodeMcu que incluye chip Wifi y otras herramientas micrófono, protoboard, una pantalla alfanumérica LCD, adaptador LCD a I2C y tres LED´S multicolores. Para su funcionamiento se utilizó internet vía Wifi y una batería portátil, compartiendo datos desde de un teléfono celular. (Otiniano López, 2018, p.144)

Implementación de un prototipo de medición de ruido ambiental con geoposicionamiento, el sistema para la realización de las mediciones está por: micrófono, tarjeta de adquisición y acondicionamiento de datos, una tarjeta electrónica STM32f429, que consiste en un micro controlador CortexM4, y un GPS. La implementación y construcción del prototipo y su posterior determino que el diseño utilizado es viable económica y técnicamente, con medidas precisas para cumplir con su aplicación de medidor de ruido ambiental, además de la estación de monitoreo y plataforma usada. (Vélez González, 2016, p.56)

Diseño e implementación de un prototipo de medición acústica remoto, el proceso comprendió el diseño inicial del sistema completo, que consta de la unidad de medición, una estación de recepción y recolección de datos. El diseño y construcción del prototipo tiene una precisión aceptable para su aplicación de sonómetro de reconocimiento, estación de monitoreo y la plataforma usada permitirá mejoras y expansiones a la funcionalidad. (Bodenhorts & Carlos, 2014, p.255)

## FUNDAMENTACIÓN TEÓRICA

Dentro de esta sección se aplicará lo más conveniente para el diseño del prototipo de señales de contaminación acústica

## ¿QUE ES ARDUINO IDE?

Entorno de desarrollo y programación de código abierto para placas de Arduino, esta plataforma permite crear diferentes tipos de microcontroladores de una sola placa a los que los programadores puede darles diferentes tipos.

Se define con dos conceptos libres Hardware y Software

**Hardware libre:** son dispositivos cuyas especificaciones y diagramas son de accesos públicos, de manera que los programadores puedan replicarlas. Con el fin de que cualquier persona o empresa pueda crear sus propias placas

**Software libre**: son programas informáticos cuyo código es accesible para cualquiera para utilizarlo y modificarlo. (Pedrera, 2017, p.122)

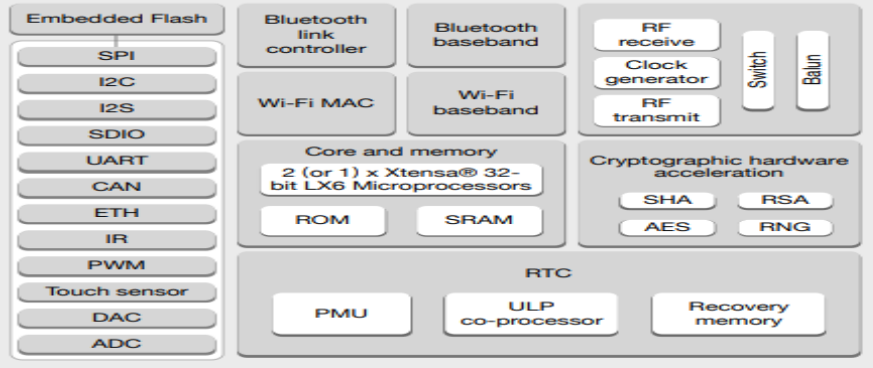
## ¿QUÉ ES NODE MCU ESP 32?

ESP 32 es un microcontrolador con un modelo de wifi y bluetooth integrado, lo cual es un sistema bajo costo y consumo en un chip SoC, por lo que se lo conoce como un funcionamiento dual o doble núcleo de 32 bits a 160 Mhz, es decir, que se puede dedicar uno al programa principal y otro a la comunicación continua, por otro lado, incluyen capacidades de cifrado acelerado por hardware, lo que lo hace seguro al utilizar comunicaciones seguras como HTTPS y WWS. Diseñado para dispositivos móviles; tanto para aplicaciones de electrónica y las aplicaciones IOT, logrando un consumo de energía bajo a través de funciones de ahorro de energía.

Desarrollo de aplicaciones para IOT con el módulo ESP32, en este proyecto se realizó la medición de datos ambientales y la humedad con el sensor BME 280, en el que se visualiza en diferentes plataformas: desde un servidor web, desde ThingSpeak, que se trata de una base de datos orientada a IoT, desde Adafruit.io, que se trata de un MQTT-Broker que

mediante un panel de control permite interactuar con el módulo ESP32, y desde la plataforma IFTTT, que mediante la conexión al servicio de Twitter, permitirá visualizar los datos desde dicho servicio. (Benito Herranz, 2019, p.79)

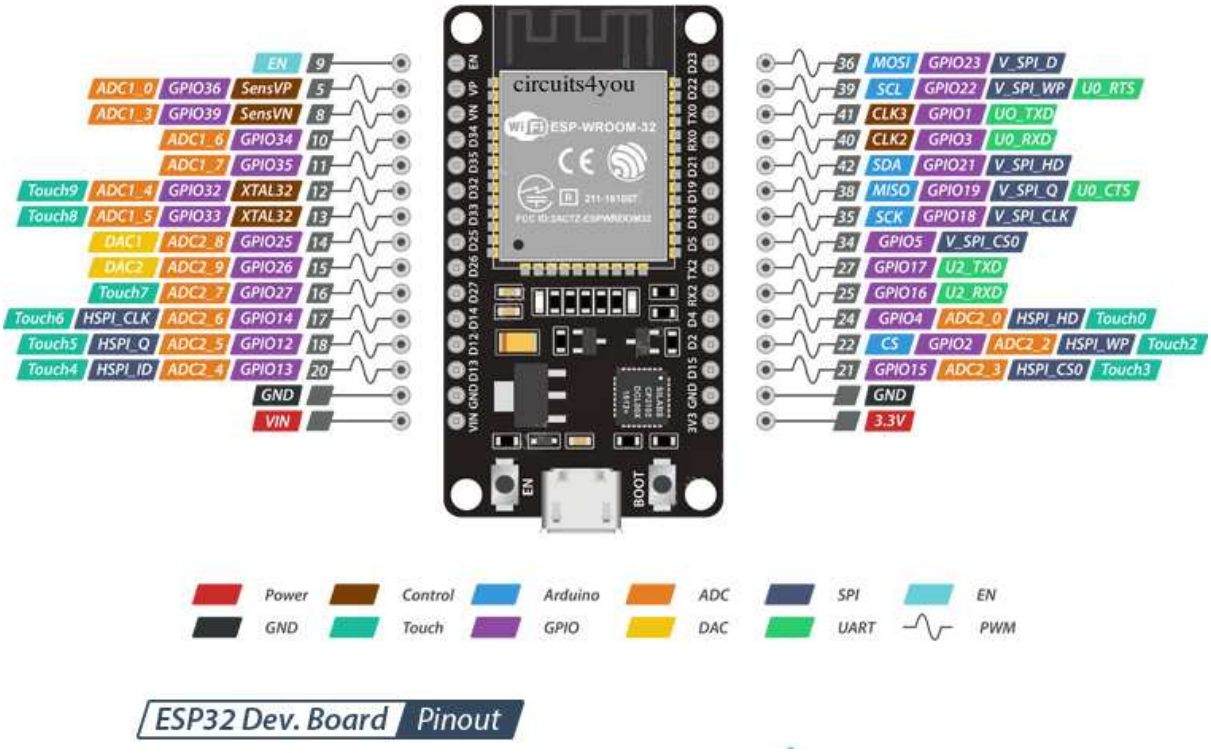
*Gráfico 1* Características técnicas del ESP-32




**Elaborado:** Juan Carlos Macho
**Fuente:** soloarduino.blogspot.com

Este módulo permite comunicar con diversos dispositivos con los protocolos como SPI, I2C, CAN, ETHERNET, etc. A demás se puede conectar con diversos dispositivos bluetooth, también amplificadores de bajo consumo, tiene una interfaz para la tarjeta sd, entre otros.

*Gráfico 2* Diagrama ESP 32 WROOM-32




**Elaborado:** Jacob Shroeder
**Fuente:** www.ioxhop.com

Dentro de la figura 2. Se muestran los pines, en el cual están organizados para uso de lo que requiera el proyecto

**Cuadro 1** *Significado de pines ESP32*

| Pines | Significado |
|---|---|
| GPIO | pines de entrada y salida general |
| ADC | conversor analógico digital |
| UO_TXD y UO_RXD | pines de conexión cruzada |
| SDA | Para enviar y recibir datos |
| Touch | Pin táctil |
| UART | Característica de hardware que maneja la comunicación los requisitos de tiempo y el encuadre de datos |

| | |
|---|---|
| DAC | Convertidor Analógico Digital |
| SPI | Controla los periféricos de la ESP32 |
| GND | Conexión tierra |
| 3.3v | Voltaje ESP 32 |
| EN | Reinicio |

**Elaborado:** Joselyn Quimis Suárez
**Fuente:** Microcontrollerslab.com,2015.

## ¿QUE SON SENSORES?

Los sensores son dispositivos electrónicos capaz de detectar variables físicas tales como temperatura, iluminación, movimientos y presión y convertirlas en variaciones de magnitud eléctrica o magnética. También son conocidos como transductores, por lo que son componentes fundamentales de los sistemas modernos de adquisición de datos (AKA DAQ O DAS). Existen varios tipos de sensores que se han inventado para medir fenómenos y dependiendo del tipo de sensor, su salida eléctrica puede ser un voltaje, corriente, resistencia o salidas digitales, por lo que generan una serie de bytes de datos escalados o no escalados, mientras la salida de sensores analógicos que está conectada a la entrada de un acondicionador de señal. (Serna, Ros, & Rico, 2010, p.278)

## DIFERENCIA ENTRE RUIDO Y SONIDO

Ruido es un tipo de sonido que produce sensaciones desagradables y causa trastornos en la salud de los humanos. Sonido es la vibración mecánica de las moléculas de un gas, de un líquido, o de un sólido que se propaga en forma de ondas y lo puede percibir el oído humano. (Domínguez Ruiz, 2015, p.300)

### Tipos de Ruido

Existen diferentes tipos de ruido, que varían según su característica, otra forma de clasificación de los sonidos distingue entre ruido blanco, rosa, marrón e industrial:

### Ruido Continuo

Se presenta cuando el nivel de presión sonora es prácticamente constante durante el periodo de observación (a lo largo de la jornada de trabajo). Este tipo de ruido es típico de las industrias como la textil y un taller de herramientas automáticas, donde el nivel de ruido no varía significativamente durante todo el día de trabajo. (Alonso Díaz, 2014, p.188)

### Ruido Intermitente

Es cuando se producen caídas bruscas hasta el nivel ambiental de forma intermitente, volviéndose a alcanzar el nivel superior. El nivel superior debe mantenerse durante más de un segundo antes de producirse una nueva caída. Ruido característico de plantas de fundición, aserraderos, industria metal mecánica etc. (Robledo, 2014, p.115)

### Ruido de Impacto

Se caracteriza por una elevación brusca de ruido en un tiempo inferior a 35 milisegundos y una duración total de menos de 500 milisegundos. Ejemplos explosiones, maquinas compactadoras. (Mendez Ortiz & Ruiz Escobar, 2019, p.265)

## Característica del ruido

- Es el contaminante más barato.
- Es fácil de producir y necesita muy poca energía para ser emitido.
- Es complejo de medir y cuantificar.
- No deja residuos, no tiene un efecto acumulativo en el medio, pero si puede tener un efecto acumulativo en el hombre.
- Tiene un radio de acción mucho menor que otros contaminantes.

- No se traslada a través de los sistemas naturales.
- Se percibe solo por un sentido: el Oído, lo cual hace subestimar su efecto; (esto no sucede con el agua, por ejemplo, donde la contaminación se puede percibir por su aspecto, olor y sabor). (Basco Prado, Fariñas Rodríguez, & Hidalgo Blanco, 2010, p.275)

## ¿CUÁLES SON LOS DIFERENTES NIVELES DE RUIDO QUE EXISTEN?

Los niveles de ruido se miden en decibelios según su intensidad y nivel de potencia. El valor 0 dB equivale al umbral de audición del ser humano, aunque este puede variar entre las personas, se considera de forma genérica, el valor mínimo de audición. (Sanjuanero, 2012, p.122)

**Cuadro 2** *Niveles de Ruido*

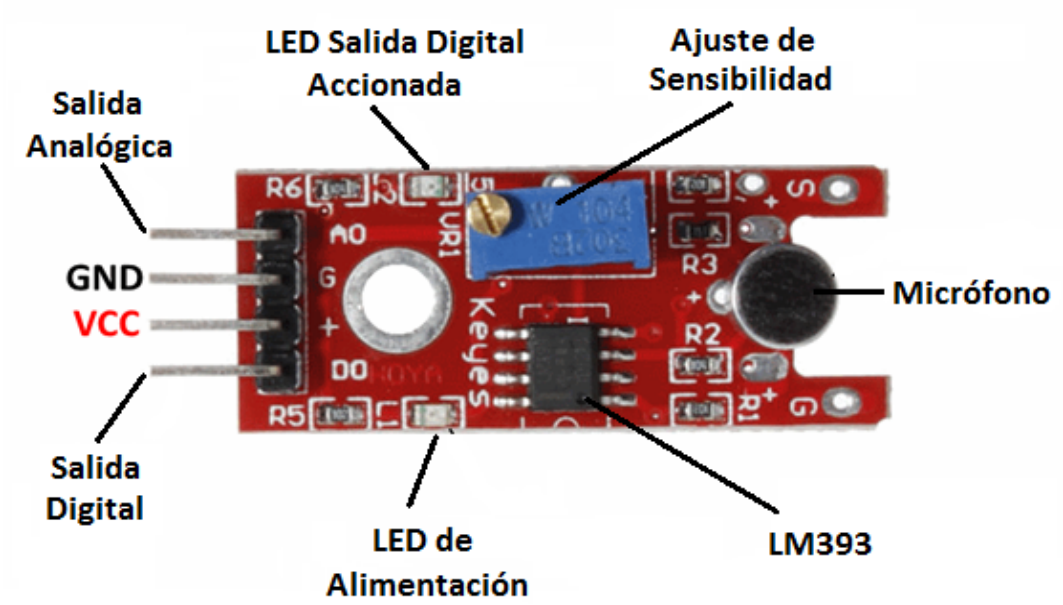
| Nivel | Descripción |
|---|---|
| 0 | Nivel mínimo de audición |
| 10-30 | Nivel de ruido bajo equivalente a una conversación tranquila. |
| 30-50 | Nivel de ruido bajo equivalente a una conversación normal. |
| 55 | Nivel de confort acústico establecido en Ecuador |
| 65 | Nivel máximo permitido de tolerancia acústica establecido por la OMS. |
| 65-75 | Ruido molesto equivalente a una calle con tráfico, televisión alta |
| 75-100 | Inicio de daños en el oído que produce sensaciones molestas y nerviosismo. |
| 100-120 | Riesgo de sordera |
| 120 | Umbral de dolor acústico |
| 140 | Nivel máximo que el oído humano puede soportar. |

**Elaborado:** Joselyn Quimis Suárez
**Fuente:** Sanjuanero,lineaverdeceutatrece.com,2012.

## ¿QUE SON SENSORES DE RUIDO?

El funcionamiento principal consiste en la detección de ondas sonoras y acústicas, donde las oscilaciones de la presión del aire, son convertidas en ondas mecánicas. Se representa en una tarjeta de pequeñas dimensiones para *aplicaciones* en las que se necesitan medir la intensidad del sonido o ejecutar alguna *instrucción* al detectar un ruido.(Mejía Saca, 2018, p.75)

*Gráfico 3* Diagrama Sensor de Sonido KY-037




**Elaborado:** Juan Carlos Macho
**Fuente:** Prometec.net/sensor-sonido-led-s4a

A continuación, se muestran el significado de cada una de las partes del sensor de sonido, en el cual se puede visualizar en la tabla 3.

**Cuadro 3** *Partes del Sensor de Sonido*

| PARTES | SIGNIFICADO |
|---|---|
| DO | Es una salida digital que actúa de modo comparador. Si el sonido captado por el micrófono supera un determinado nivel se pone HIGH. |
| AO | Es una salida analógica que da un valor entre 0 y 1023 en función del sonido |
| GND y VCC | Ajuste tierra y voltaje. En el centro tenemos la conexión a 5V y a GND |
| LED Salida Digital Accionada | Si se conecta bien el sensor, se |

| | |
|---|---|
| | deberá iluminar el LED de alimentación, mientras que el LED de salida digital accionada puede o no estar encendido |
| Ajuste de Sensibilidad | El ajuste de sensibilidad del micrófono se hace mediante un potenciómetro, en el cual hay que girar para con un destornillador plano. |
| Micrófono | Capta el sonido |
| LM393 | Es un circuito integrado que contiene dos unidades iguales, diseñado para ser utilizado como comparador de voltaje de presión |
| LED de Alimentación | Contiene dos LED´S, uno que indica si hay alimentación en el sensor y otro que se ilumina si DO esta HIGH |

**Elaborado:** Joselyn Quimis Suárez
**Fuente:** Prometec, prometec.net,2017

### ¿QUE ES ADS 1115?

Conversor analógico digital externo ideal, el cual se usa cuando se requiere más resolución o de más pines analógicos, que contiene 16 bits en el que puede resultar útil para ciertos proyectos, cuyo objetivo es ser procesada por un microcontrolador para diversos propósitos. (Guillen-Mendoza, Ramos Martín, & Santana Rodríguez, 2016, p.123)

A continuación, se muestra el siguiente cuadro 4.

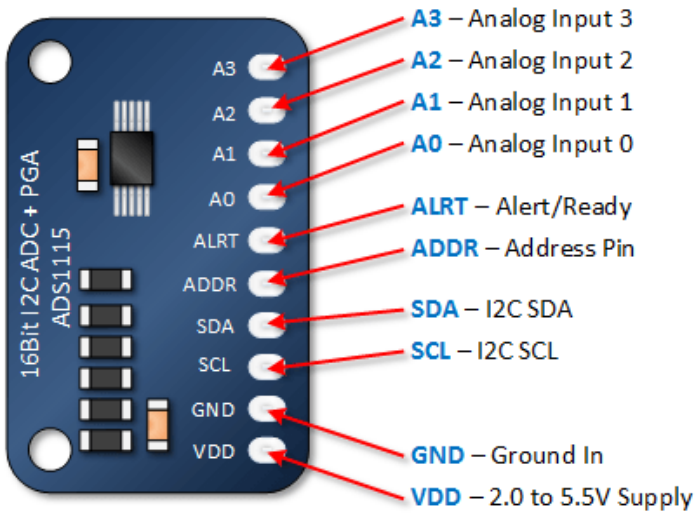
Cuadro 4 *ADS 1115*

| Característica | Valor |
|---|---|
| Voltaje de operación | de 2V a 5,5V |
| Consumo de corriente | 150 µA (modo continuo) |
| Velocidad de muestreo programable | de 8 Hz a 860 Hz |
| Resolución | 16-bit |
| Canales | 4 canales de entrada o 2 diferenciales |
| Interfaz de comunicación | I2C (4 direcciones) |

**Elaborado:** Guellen-Mendoza, Ramos Martín & Santana Rodríguez
**Fuente:** cdn-shop.adafruit.com/datasheets/ads1115.pdf,2016

En el gráfico 4, se muestra el uso de esta herramienta ya que permitirá ampliar la resolución, permitiendo captar más ruido y su identificación de cada uno del pin que componen esta placa.

**Gráfico 4** Diagrama ADS 1115




**Elaboración:** Joselyn Quimis Suarez
**Fuente:** robu.in/product/ads1115-16-bit-adc-4-channel-programmable-gain-amplifier/

**Cuadro 5** *Características ADS1115*

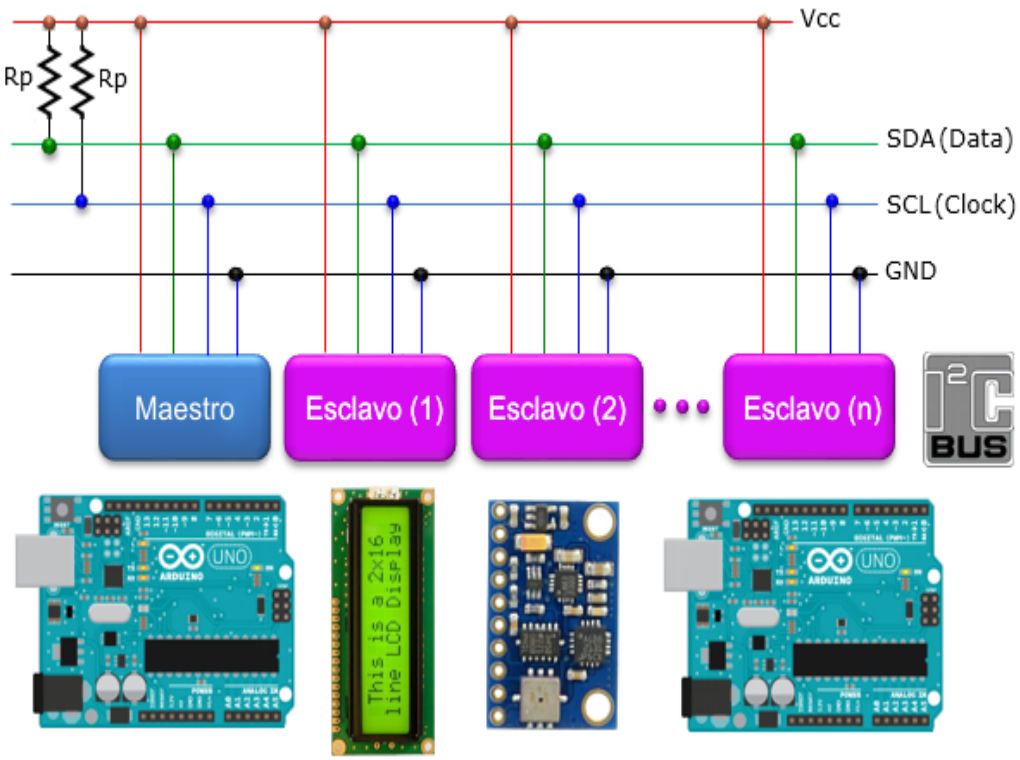
| PIN | CARACTERISTICAS |
|---|---|
| VDD Y GND | Son los pines de alimentación |
| A0-A3 | Resolución de 15 bits |
| ALRT/READY | Hay dos usos para el pin ALERTA / Listo. La primera es la alerta de "umbral del comparador" (el modo predeterminado). La segunda es la alerta de "lectura de ADC lista". Tiene que configurar algunos registros para operar el modo ADC ready ALERT. |
| ADDR | El pin ADDR determina la dirección I2C del dispositivo, El modo por defecto es conectar el pin ADDR a GND, lo que da lugar a la dirección 0x48, 0X49 VDD Y 0X4A SDA. |
| SDA Y SCL | Pines de reloj y de datos I2C (protocolo de comunicación serial utilizado en multitud de sensores y actuadores). |

**Elaborado:** Joselyn Quimis Suárez
**Fuente:** Luis del Valle, 2018

## ¿QUE ES I2C?

Circuito integrado interno es un bus con múltiples maestros, por lo que pueden conectarse varios chips al mismo bus y actuar como maestros, iniciando la transferencia de datos. Este estándar facilita la comunicación entre microcontroladores y dispositivos de inteligencia, para esto se requiere de dos líneas de señal y un común, permitiendo el intercambio de información entre varios dispositivos a una velocidad aceptable de 100kbits por segundos, aunque hay casos especiales en los que el reloj llega hasta 3.4 MHz. (Quiñonez, Lizarraga, Peraza, & Zatarain, 2019, p.245)

**Gráfico 5** *Esquema interno ADS 1115*




**Elaboración:** Vladimir Alves
**Fuente:** .vladcontrol.com.br

## MAESTRO

Se encarga de controlar el cable de reloj, llamada SCL serial clock, este se encarga de iniciar y para la comunicación. La información binaria serial se envía solo por la línea de datos seriales llamada SDA Serial Data, por lo tanto, dos maestros no pueden hacer uso del mismo puerto. Por otro lado, puede funcionar de dos maneras como, maestro-transistor y maestro receptor. (Luján Cuenca, 2017, p.187)

### Función principal de maestro

- Iniciar la comunicación – S
- Enviar 7 bits de dirección – ADDR
- Generar 1 bit de Lectura ó Escritura – R/W
- Enviar 8 bits de dirección de memoria
- Transmitir 8 bits de datos –

- Confirmar la recepción de datos – ACK – ACKnowledged
- Generar confirmación de No-recepción, NACK – No-ACKnowledged
- Finalizar la comunicación.

### Maestro Transistor y Esclavo Receptor

Se usa este modo cuando se desea configurar un registro de esclavo I2C

### Maestro Receptor y Esclavo transmisor

Se usa cuando se quiere leer información del sensor I2C. (Oberg et al., 2011, p.261)

## ESCLAVO

Suele ser un sensor, el cual suministrara la información de interés de MAESTRO, puede actuar de dos formas como esclavo transmisor o esclavo receptor. Un esclavo no puede generar a la señal SCL. Sus funciones son las siguientes.

Enviar información en paquetes de 8 bits

Enviar confirmaciones de recepción llamadas ACK. (Mankar, Darode, Trivedi, Kanoje, & Shahare, 2014, p.125)

## ¿QUE ES MODULO GPS?

Es un módulo receptor que utiliza la comunicación USART, la cual significa Universal Syncronous/Asyncronous Receiver Transmitter ó Transmisor-Receptor Síncrono/Asíncrono Universal, es un protocolo que emplea las comunicaciones duales, por lo tanto, es la capacidad de recibir y transmitir simultáneamente los datos, ya que se caracteriza por transmitirlos de manera serial, significando que solo un bit es transferido por el canal de tiempo. Esta comunicación permite comunicarse con el microcontrolador y terminal de pc, por otro lado, recibe información como la altitud, longitud, latitud, hora UTC principal estándar del tiempo, etc de los satélites en forma de cadena NMEA, lo cual la cadena corresponde a la salida serial de un GPS que soporta el estándar de comunicación NMEA, lo que permite comunicarse unos a otros. En este estándar cada valor está

separado por una coma, como ejemplo ‘4807.038,N' y '01131.000,E' que corresponden a la latitud y longitud. Esta cadena necesita ser analizada para extraer la información que queremos usar. (Ruiz, 2013, p.183)

### Características del módulo GPS NEO-6M

- Tiene una antena externa y una EEPROM incorporada.
- Interfaz: RS232 TTL, envío de TXD y recepción de señales RXD a un nivel de compatibilidad TTL/COMS. El nivel TTL es 0-5 V y no tiene alimentación externa gracias al uso interno del “circuito de carga único “RS-232” .
- Fuente de alimentación: 3V a 5V
- Velocidad de transmisión predeterminada: 9600 bps
- Funciona con frases NMEA estándar. (Zabala, Cuenca, León, & Cabrera, 2018, p.62)

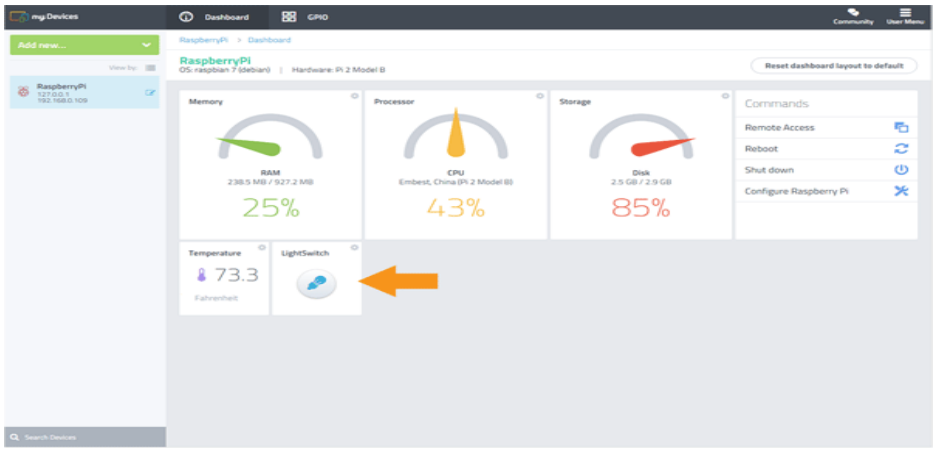
### ¿QUE ES IFTTT?

Sus siglas significan “If This, Then That” y permit crear y programar acciones entre diferentes aplicaciones dentro de su página, en la cual conecta diferentes servicios online para automatizar todo tipo de tareas conocidas como Applets o recetas, cuya función es intermediario entre sus más de 350 servidores asociados para poder combinar acciones y automatismo desde redes sociales hasta sistemas domóticos y estrategias de correo electrónico.

Integración de robot social en sistema domótico, el objetivo de este proyecto es dar a conocer un nuevo robot, Aisoy, y principalmente y lo más atractivo, su integración en domótica. Para la aplicación de la domótica se ha adquirido un kit de bombillas LED Belkin inteligentes, capaces de ser conectadas a la plataforma IFTTT. (Racero Valcárcel, 2016, p.111)

## ¿QUE ES CAYENNE MYDEVICES?

**Gráfico 6** *Plataforma Cayenne*




**Elaborado:** Programación Diseño Autorizado y Control
**Fuente:** pdacontroles.com

Es una plataforma que se emplea frecuentemente en el prototipo de los diferentes dispositivos que son IoT, permite que sea fácil de programar, ya que es intuitivo. Se pueden crear paneles de control de una forma sencilla arrastrando y soltando widgets para controlar los dispositivos conectados al IoT que puede ser el ESP-32 o domóticos a través del protocolo MQTT. Para el uso del sensor se usa una librería predefinida, ya que necesarias para su control por medio del microcontrolador. (Vélez, 2020, p.77)

## ¿QUE ES PROTOCOLO MQTT?

Es un protocolo de mensajería liviano cliente/servidor, en el cual puede usarse sobre TCP/IP. Utiliza unas metodologías de eventos y mensajes basándose en publicación y suscripción, lo cual hace de intermediario que es responsable de distribuir mensajes a los clientes. Este protocolo MQTT ideado por IBM también es conocido como Machine to Machine, por lo que se ha vuelto tan popular para la comunicación entre dispositivos de IoT.

## USANDO MQTT CON CAYENNE

MQTT es el transporte y la API preferidos para enviar datos a Cayenne Cloud o para dispositivos que reciben comandos de MQTT es el transporte y la API preferidos para enviar datos a Cayenne Cloud o para dispositivos que reciben comandos de utilizando Cayenne Cloud. Cayenne MQTT es sencillo y fácil de usar, ofreciendo varias formas diferentes de conectar sus datos a Cayenne. (Moreno Cerdà, 2018, p.456)

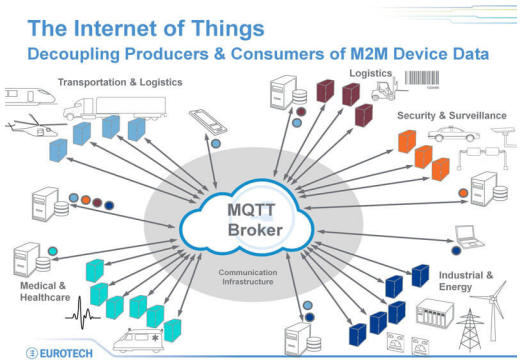
## SEGURIDAD EN MQTT

El protocolo MQTT dispone de distintas medidas de seguridad para proteger las comunicaciones. Incluye transporte SSL/TLS, autenticación por usuario y contraseña o mediante certificados. (Lee, Kim, Hong, & Ju, 2013, p.315)

## ARQUITECTURA MQTT

La arquitectura MQTT tiene una topología estrella, con un nodo central que hace de servidor o bróker. El broker es el encargado de gestionar la red y de transmitir los mensajes, para mantener activo el canal, los clientes mandan periódicamente un paquete (PINGREQ) y esperan la respuesta del broker (PINGRESP). La comunicación puede ser cifrada entre otras muchas opciones. (Borja Vega, 2020, p.205)

*Gráfico 7* Arquitectura MQTT




**Elaborado**: Jefferson Crespo M.
**Fuente**: aprendiendoarduino.com

## DEFINICIONES CONCEPTUALES

### RUIDO BLANCO

Ruido blanco es un tipo de señal de carácter aleatorio, que no presenta correlación estadística entre sus valores en dos tiempos diferentes, presenta todas las frecuencias y su potencia es constante, ejemplo de esto podría ser el sonido de una aspiradora en funcionamiento o un secador de cabello. (dos Santos & Delfino, 2019, p.168)

### RUIDO ROSA

Se usa habitualmente en acústica para ecualizar el sonido en salas de audición, esta caracterizado por una densidad espectral inversamente proporcional a la frecuencia, por lo que se visualiza como un ruido de nivel constante en todas las bandas de octava. Por otro lado, se ha comprobado la mejora de la sensación de bienestar, la productividad del ámbito laboral y la mejora del sueño como ejemplo el ruido de una lluvia suave o los latidos del corazón. (Gutiérrez Jaramillo, 2015, p.25)

### RUIDO MARRON

El ruido marrón es conocido como ruido rojo o brown por su creador Robert Brown y por la similitud de la amplitud de las frecuencias del sonido con partículas del movimiento browniano, está compuesto principalmente por frecuencias medias y graves, mientras que el ruido blanco las frecuencias son más altas. En el ruido marrón la energía decrece 6dB por octava, porque las frecuencias son bajas, similar al sonido de agua del mar o al batir las olas, por lo que es considerado al igual que el ruido rosa se considera agradable. (Varela Neila, 2016, p.164)

### RUIDO INDUSTRIAL

Es producido por las actividades humanas de este sector, el ruido industrial no solo es un riesgo laboral, sino que también pueden causar trastornos y molestias a la población cercana. (Mahedero Biot, 2020, p.95)

### ONDAS SONORAS

Una onda sonora es una onda expansiva que puede ser percibida por el oído humano, estas pueden propagarse por distintos medios, los cuales

pueden ser sólidos, líquidos y gaseosos. Las ondas sonoras están sujetas a las leyes físicas como la reflexión, flexión, refracción y absorción, la velocidad depende del medio en que se propagan las ondas, en el aire la velocidad es de 330 metros por segundo mientras que en materiales solidos ascendente a varios miles de metros por segundos. Las ondas de sonido no se pueden propagar en el vacío, ya que necesitan de un medio para hacerlo.(Carbonel Martínez, 2014, p.133)

## ONDAS ACUSTICAS

Es una onda longitudinal asociada con el sonido, por lo que si se propaga en un medio elástico y contino genera una variación de presión o densidad, transmitiéndose en forma de onda esférica. Las variaciones de presión, humedad o temperatura del medio, producen el desplazamiento de las moléculas que lo forman. Cada molécula transmite la vibración provocando un movimiento en cadena, el movimiento de las moléculas del medio produce en el oído humano una sensación descrita como sonido. (Montilla Pérez, 2018, p.48)

## ESPECTRO DE FRECUENCIA

Se caracteriza por la distribución de amplitudes para cada frecuencia de un fenómeno ondulatorio, como son los colores, las notas musicales, las ondas electromagnéticas de radio o TV e incluso la rotación regular de la Tierra. El oído descompone el sonido recibido en sus componentes frecuenciales, es decir las ondas senoidales según el teorema de Fourier, conforman ese sonido, en los cuales pueden extenderse a sonidos aperiódicos, que pueden ser tan simples como los sonidos de una campana o tan complejos como el ruido blanco similar al que capta una emisora de FM en ausencia de señal. El espectro es importante porque permite una descripción de las ondas sonoras que están íntimamente vinculada con el efecto de diferentes dispositivos y modificadores físicos del sonido. **(López Molinero, 2018, p.278)**

# CAPÍTULO III

## CONSTRUCCIÓN DEL PROTOTIPO

Para el diseño del prototipo de señales de contaminación acústica se requiere de un sensor de sonido para captar el ruido y poder medirlo, trabajando junto con el microcontrolador ESP32, en el cual se programa cada una de las herramientas que van conectadas al microcontrolador y GPS para saber la ubicación de donde provenga el ruido, por otra parte, se utilizará una plataforma digital llamada Cayenne, en la cual permite monitorear y controlar el ESP 32. Este prototipo para que pueda funcionar necesita estar conectado a wifi, para mostrar la información de requerida, de igual manera se puede adaptar a cualquier red donde se encuentre.

La plataforma también se puede ver en cualquier dispositivo móvil, en la cual tiene un sistema de notificación que permite saber en qué sitio de la ciudad hay altos niveles de dB

### SOFTWARE

*Cuadro 6* Arduino IDE

| CARACTERISTICAS | ESPECIFICACIONES |
|---|---|
| **Versión** | 1.8.13 |
| **Sistema Operativo** | Windows XP, vista, Windows 7, Windows 8, 10 |

**Fuente:** Datos de investigación
**Elaboración:** Joselyn Quimis Suarez

*Cuadro 7* Cayenne

| CARACTERISTICAS | ESPECIFICACIONES |
|---|---|
| Navegador | Chrome, Mozilla y Microsoft Edge |
| Sistema Operativos | iOS y Android |
| Placas y Hardware | Rasberry Pi, Arduino, ESP 8266 y dispositivos LoRaWan |
| Comunicación | Placas con Bluethooth y Wifi |

**Fuente:** Datos de investigación
**Elaboración:** Joselyn Quimis Suarez

Para el diseño de este prototipo se usará una plataforma digital open source

- **Etapas de la metodología del sistema**

En el diseño del prototipo se usa la metodología en cascada.

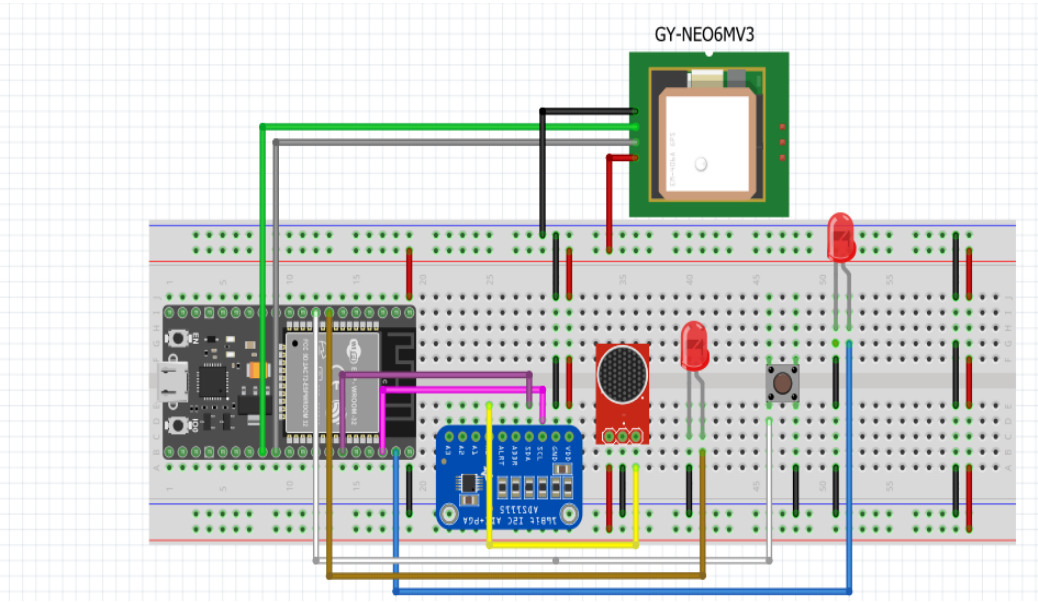
### 1.1.1. Etapa análisis de requisitos del software:

En esta etapa se buscó un software libre.

### 1.1.2. Etapa de diseño del sistema:

El diseño del prototipo requiere algunas herramientas para la programación y configuración del mismo como el Arduino IDE y la plataforma Cayenne. Por otra parte, los componentes para armar el prototipo que se utilizaron fue el sensor de ruido, ESP32, microcontrolador modulo GPS.

*Gráfico 8* Diseño del Prototipo

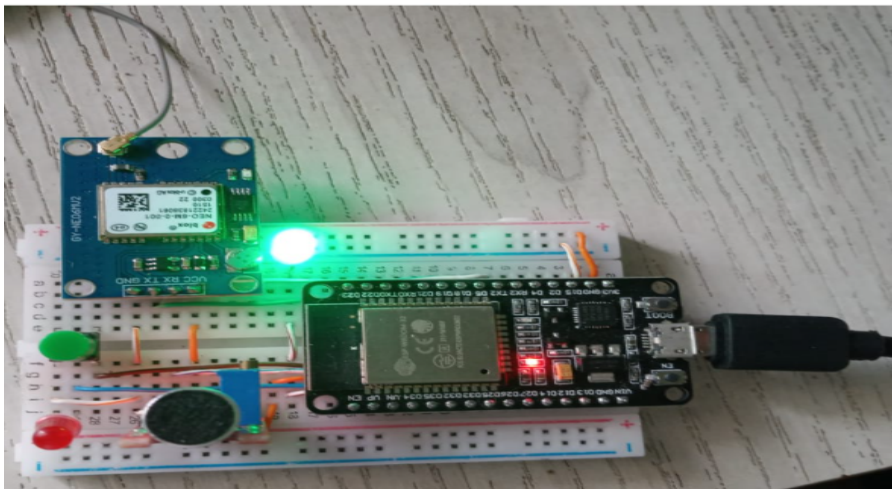



**Elaboración:** Joselyn Quimis Suarez
**Fuente:** Frizing placa de prototipos

Para uso del microcontrolador ESP-32 se conectaron sus respectivos voltajes de corriente directa y GND tierra, como se puede visualizar en la fig. 12. Todos estos conjuntos de componentes estarán conectados por la ESP32, trabajando de acuerdo con el esquema de cada uno de los siguientes componentes.

- **Conexión modulo GPS**. - Para conectar el módulo GPS con la ESP 32 se usan los pines 16 y 17. El pin GPIO 16 que es Rx se conectara con el pin Tx del GPS, mientras el pin GPIO 17 equivale Tx se conectara con el pin Rx del GPS, realizando una comunicación cruzada como también se planteara en el diseño físico y su respectiva programación. Los cables negro y rojo van a ir conectados al voltaje + y GND tierra.
- **ADS 1115.-** Es un conversor analógico digital externo que podemos conectar a un procesador como Arduino, para medir señales lógicas, su comunicación es IDS, la cual obtiene los datos del sensor de ruido usando los pines SDA 16 Y SCL 17, conectándolo con los pines del microcontrolador GPIO 21 y 22. Se utilizo, debido a que la ESP 32 no ve los valores menores a 0.1v, haciendo valer como voltaje 0
- **Conexión sensor de ruido. –** se definió los pines GPIO 34 en el cual se conectó al pin AUD del sensor para uso del micrófono y sus respectivos voltajes.
- **Botón. –** se realiza la conexión de voltaje tierra y el pin GPIO 33 Touch 8 para uso del botón por lo cual se plasmará en el desarrollo del prototipo. Como se puede apreciar en la fig 13.
- **Led Red. –** utilizara el pin GPIO 32 en el cual se conectará en el ánodo del led, mientras que el cátodo se conectara el voltaje tierra o GND, para uso de conexión de red.
- **Led Ruido. –** el cátodo de este led se conectará a negativo, mientras el ánodo se conectará con el pin GIO 35, para uso de alarma.

**Gráfico 9** Diseño del prototipo armado



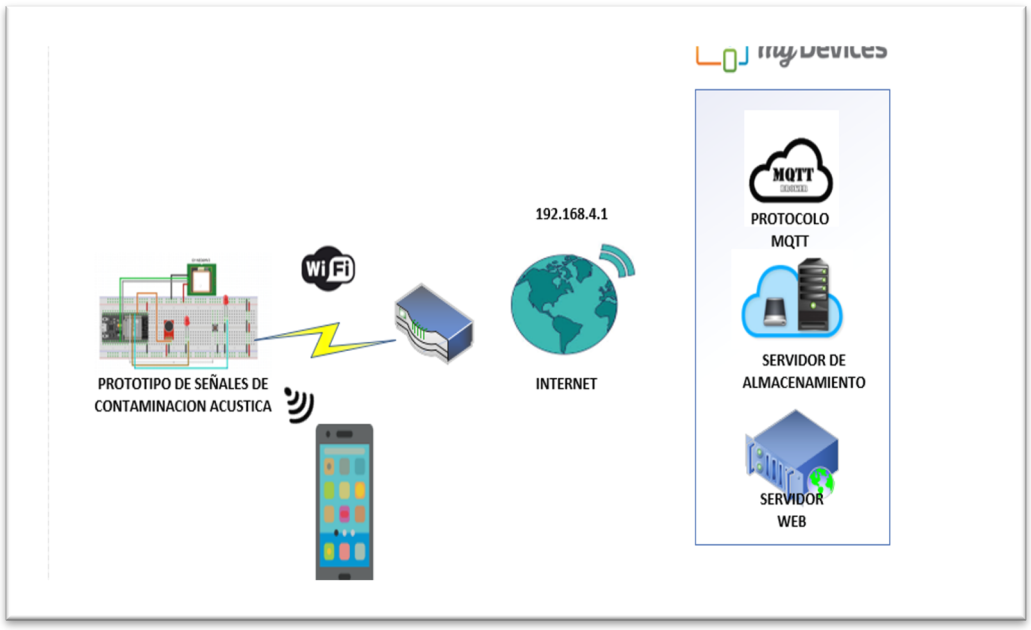
**Elaboración:** Joselyn Quimis Suarez
**Fuente:** Datos de Investigación

En la fig.13 se muestra el prototipo ya armado de acuerdo con el diseño antes planteado con sus respectivos componentes y visualizando su función, como se puede apreciar el led de red se encuentra encendido porque esta conecta a la red Wifi, mientras que el led de ruido solo se prendera si hay ruido en el ambiente.

### 1.1.3. Arquitectura del sistema

En este diagrama se muestra todos los procesos que se van a ejecutar según el código desarrollado en Arduino IDE. A continuación, se muestra el siguiente gráfico 16.

*Gráfico 10* Arquitectura de Red




**Elaboración:** Joselyn Quimis Suarez
**Fuente:** Datos de Investigación

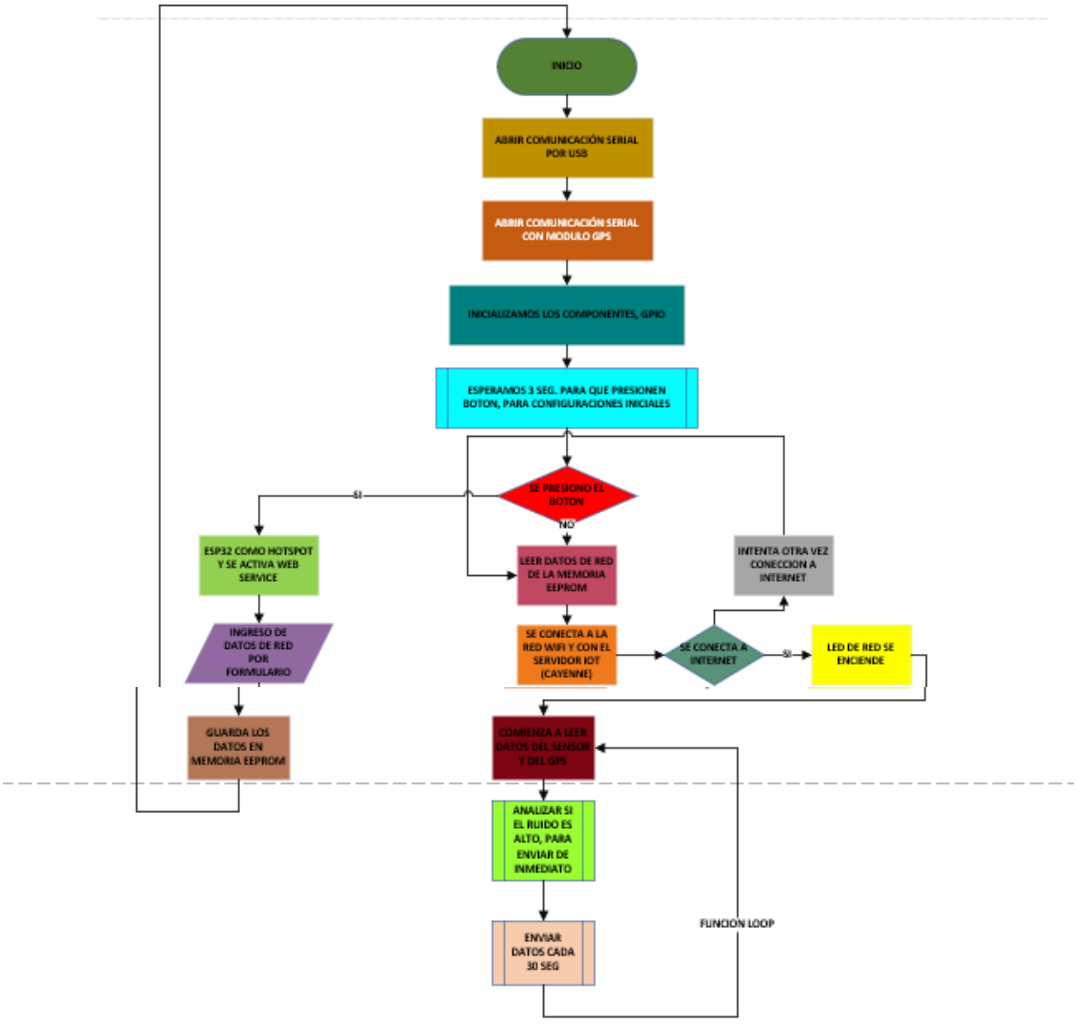
En este diagrama se puede visualizar todas las herramientas usadas a lo largo del proyecto, lo cual explica la conexión del prototipo de señales de contaminación acústica, ya que se pone como modo hac hoc con la red Wifi creada llamada "proyecto joselyn", en la cual se puede verificar en un dispositivo móvil, todo este proceso se realiza después de haber reiniciado la ESP-32 con el botón EN y la conexión del monitor serial ubicado en Arduino IDE. Una vez realizado se colocará la dirección ip 192.168.4.1 del web server, en el cual, se procederá a llenar el formulario con los datos de la red, ya conectado enviará los datos de sensor y GPS a la plataforma cayenne y el monitor serial.

### 1.1.4. Etapa de diseño del programa:

#### 1.1.4.1. Diagrama de programación Prototipo de Señales de Contaminación Acústica

Este diagrama indica cada uno de los procesos que se irán ejecutando, de acuerdo al código. Dichas funciones se estarán explicando en la siguiente etapa de Codificación.

*Gráfico 11* Diagrama de programación




**Elaboración:** Joselyn Quimis Suarez
**Fuente:** Datos de Investigación

## 2. DEMOSTRAR UN SISTEMA DE NOTIFICACIÓN, PARA SABER EL SECTOR EN QUE SE ENCUENTRA UBICADO LA INSTITUCIÓN INDICANDO LOS ALTOS NIVELES DE RUIDOS.

**2.1.1. Etapa Codificación:** en esta fase se desarrolla el programa con sus respectivas configuraciones, en las cuales se irán detallando.

### 2.1.2. Instalación de herramientas

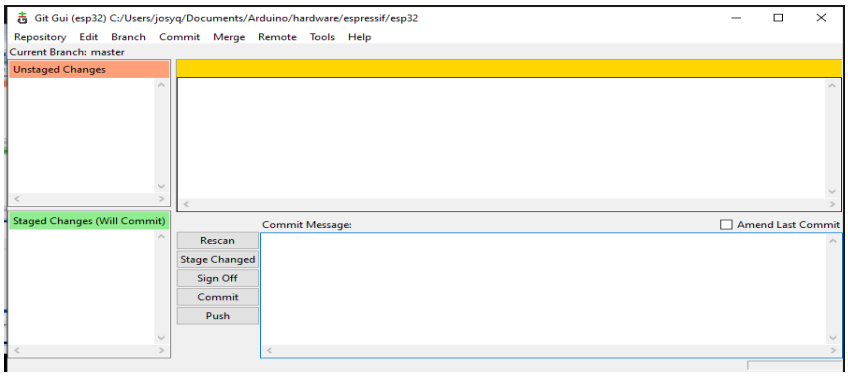
#### 2.1.2.1. Instalación Arduino IDE

En este proyecto se usó la herramienta de Arduino IDE, lo cual nos permite programar la ESP32, para esto se instaló también varias librerías para Cayenne para uso de la plataforma, GPS y el soporte para ESP32.

#### 2.1.2.2. Instalación Soporte ESP32

Para la instalación del soporte para ESP32, se instaló el programa Git 2.30.0 ya que, sirve para clonar directorios, en la cual se procedió a realizar el proceso una vez ya instalado con Git Gui la interfaz de este programa. Siguiendo los pasos de un tutorial se clona un repositorio existente, por lo que aparece una ventana indicando que esta completado el proceso y está instalada el soporte ESP32 que podremos verificar en la carpeta de Arduino. Como se puede apreciar en los gráficos 18 y 19.

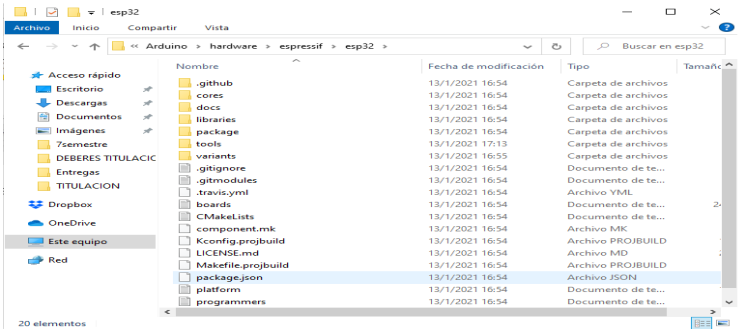
*Gráfico 12* Instalación del soporte para ESP 32




**Elaboración:** Joselyn Quimis Suarez

**Fuente:** GIT 2.30.0

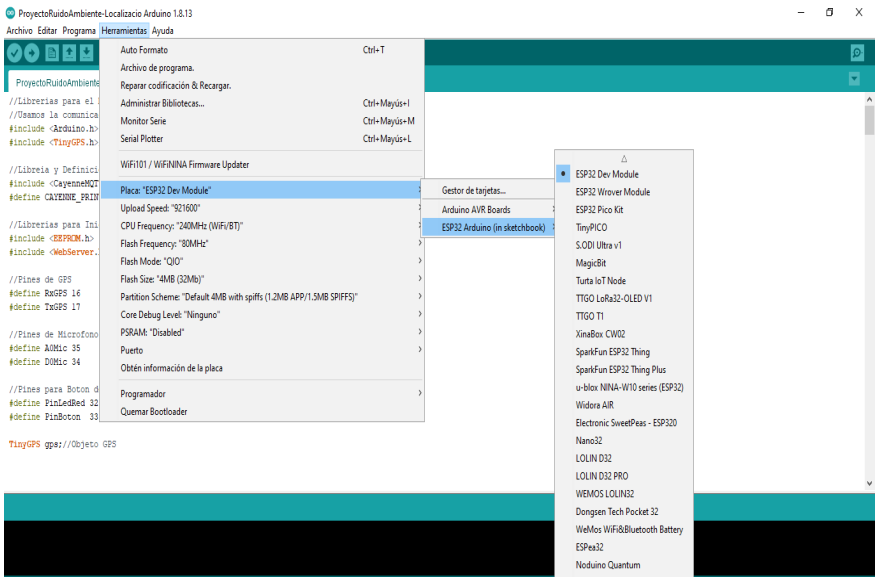
*Gráfico 13* Soporte para ESP 32




**Elaboración:** Joselyn Quimis Suarez
**Fuente:** Datos de Investigación

Una vez estando en la carpeta de Arduino, se verifica que también se encuentre ESP32 ya instalado. Como se puede visualizar en el gráfico 20.

*Gráfico 14* Arduino y Placa ESP 32

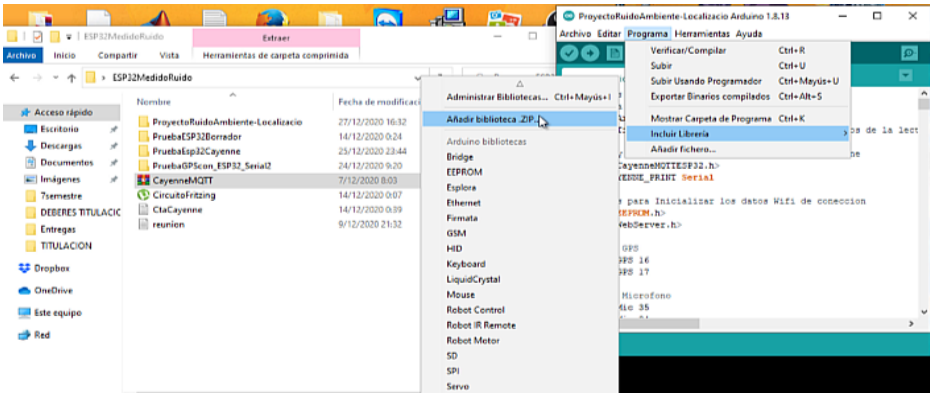



**Elaboración:** Joselyn Quimis Suarez
**Fuente**: Arduino IDE

### 2.1.2.3. Instalación Librería Cayenne

Para la instalación de la librería cayenne también se realizó el proceso de descarga y una vez realizado este proceso se incluye la librería a Arduino IDE con un archivo zip. Como se puede visualizar en el gráfico 21

*Gráfico 15* Librería Cayenne



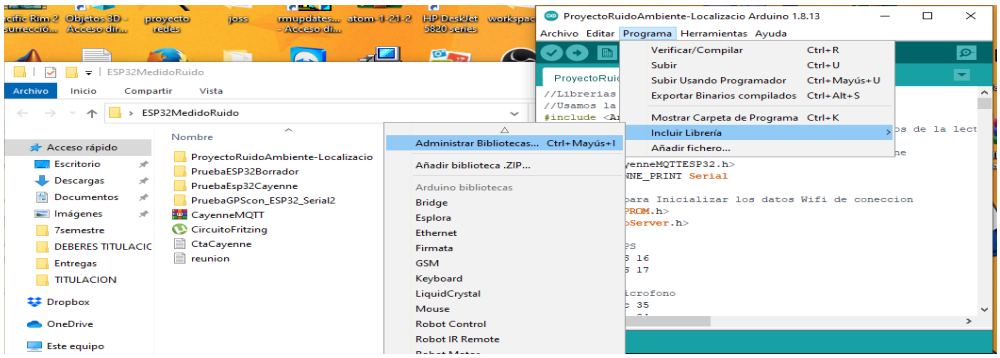

**Elaboración:** Joselyn Quimis Suarez

**Fuente:** Arduino IDE

### 2.1.2.4. Instalación Librería GPS

La instalación de esta librería es el mismo proceso que la anterior con la diferencia que no agrega a un archivo zip sino seleccionando la opción administrador bibliotecas. Como se puede visualizar en el gráfico 22

*Gráfico 16* Librería GPS




**Fuente:** Arduino IDE

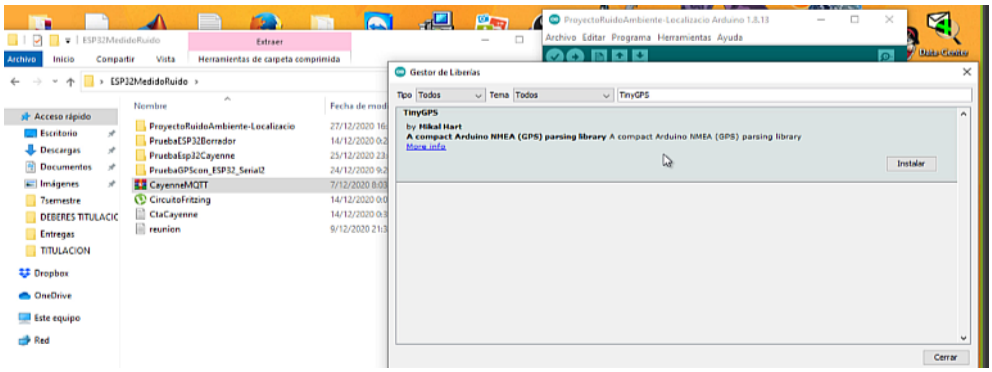
**Elaboración:** Joselyn Quimis Suarez

Una vez seleccionado esta opción se busca la librería para el uso del módulo GPS llamada TinyGPS una vez agregado se utilizará para el desarrollo del programa. Como se puede visualizar en los siguientes gráficos 23 y 24.

*Gráfico 17* Librería TinyGPS


**Elaboración:** Joselyn Quimis Suarez
**Fuente:** Arduino IDE

*Gráfico 18* TinyGPS

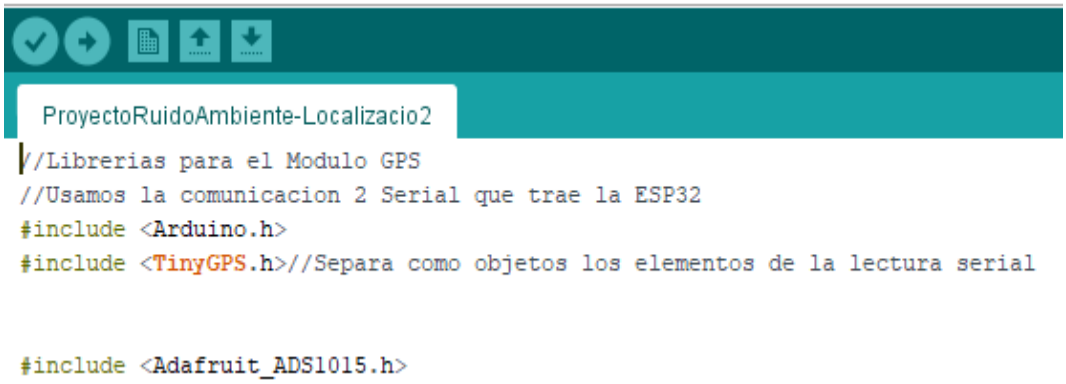



**Elaboración**: Joselyn Quimis Suarez
**Fuente:** Arduino IDE

### 2.1.2.5. Instalación de Librería ADS 1115

Se realiza el proceso anterior, de instalación de librería, en cual se instala Adafruit_ADS1015 para el uso del conversor ADS 1115, por lo que esta incluye la función map, que también será utilizada en el proyecto. A continuación, se muestra la siguiente figura 26.

*Gráfico 19* Librería Adafruit_ADS1015




**Elaboración:** Joselyn Quimis Suarez
**Fuente:** Arduino IDE

### 2.1.2.6. Instalación de la librería Webhooks

Para el proceso de instalación de las librerías de Http Client se realiza lo anterior mencionado junto con otras librerías, en la cual se van a utilizar para poder conectarse con la plataforma IFTTT. A continuación, se muestra en el siguiente gráfico 27

*Gráfico 20* Librerías HTTP Client

```
//Librerias de Http Client
#include <WiFi.h>
#include <ArduinoJson.h>
#include <HTTPClient.h>
#include <WiFiClient.h>
```

**Elaboración:** Joselyn Quimis Suarez
**Fuente:** Arduino IDE

## 2.1.3. Etapa Programación

### 2.1.3.1. INICIO - Declaración de variables pines

Como se muestra en el gráfico 27, se declaran los pines de los componentes que se van a utilizar, en el cual se explicó en el diseño anterior como van conectadas con la ESP-32.

*Gráfico 21* Definición de variables GPS


**Elaboración:** Joselyn Quimis Suarez
**Fuente:** Arduino IDE

### 2.1.3.2. Abrir comunicación serial por usb y gps-voip setup()

Dentro de esta función se ejecutan dos procesos en cascada que son serial y serial 2. En serial la comunicación será cruzada, ya que en la placa de ESP32 se encuentra los pines GPIO 3 Rx y 1 Tx permitiendo que la ESP32 se conecte con la computadora, por medio del USB, declarando con una velocidad de 115200, en la cual comunica con el monitor serial, mientras serial 2, abre la comunicación serial para uso del módulo GPS con una velocidad de 9600, en la cual también incluye, puerto serial (8) bits de datos, (n)ninguno y (1) bit de para, por utimo, sus pines declarados RxGps y TxGps. Todo esto colocado en una función, que es la primera en ejecutarse dentro del programa. Como se puede visualizar en el siguiente gráfico 28

*Gráfico 22* Función Principal

```
void setup()
{
  Serial.begin(115200);//Comunicacion con Monitor Serial de IDE
  Serial.println("Proyecto Joselyn");
  delay(2000);
  Serial.println("Iniciando comunicacion con GPS");
  Serial2.begin(9600, SERIAL_8N1, RxGPS, TxGPS);// serial2 pines 16 y 17
  Serial.println(TinyGPS::library_version());
```

**Elaboración:** Joselyn Quimis Suarez
**Fuente:** Arduino IDE

### 2.1.3.3. Inicialización de componentes

Se inician las entradas y salidas de los leds de ruido y red, mientras en el pin botón define una entrada input_pullup, lo cual se usa la resistencia interna de pull up para detectar la pulsación de un botón. Esto significa que cuando lee una entrada, su rango varia de 0 a 4095. Como se puede visualizar en el siguiente gráfico 29.

*Gráfico 23* Componentes

```
//Definicion como entrada y salida digital de pines de
//boton y led de red
pinMode(LedRuido, OUTPUT);
pinMode(PinLedRed, OUTPUT);
pinMode(PinBoton, INPUT_PULLUP);
```

**Fuente:** Arduino IDE
**Elaboración:** Joselyn Quimis Suarez

### 2.1.3.4. Tiempo de espera 3 segundos

En este subproceso, se esperará 3 para entrar en la validación que sería la configuración de la red, en la cual tiene dos validaciones para ejecutar este proceso. A continuación, se muestra el siguiente gráfico 30.

*Gráfico 24* Tiempo de espera configuración

```
EEPROM.begin(512);
/*Ahora va las rutinas para el programa*/
//En caso de que se presione el boton en los 3 Seg al encender el equipo
//Entrara en un Modo de configuracion Inicial
bool EstConfInicial = false;
delay(3000);
//Esta condicion hara poner la ESP en modo AdHoc y se creara un servidor WEB
//Con el formulario HTML q se declaro en la parte de arriba
//
if (digitalRead(PinBoton) == 0) { //Si se lee una entrada en alta La bandera EstConfInicial cambia a alto y nos pone en la espera para recibir datos via ESP
  EstConfInicial = true;
}
if (EstConfInicial == true) {

  modoconf();
```

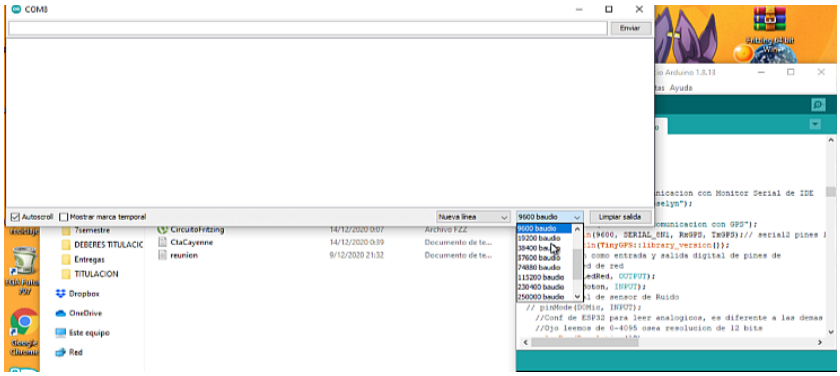
**Elaboración:** Joselyn Quimis Suarez
**Fuente:** Arduino IDE

### 2.1.3.5. Presionar el botón si

En el caso de presionar el botón se ejecutarán los siguientes procesos. Pero antes de este paso se procederá ingresar en el monitor serial y se colocará

la velocidad serial para comunicarse con la computadora como se puede apreciar en el siguiente gráfico 31.

*Gráfico 25* Monitor serial



**Elaboración:** Joselyn Quimis Suarez
**Fuente:** Datos de Investigación

### 2.1.3.6. ESP modo hostpot

Entrando a la condición de if, en el caso de que, si presione el botón, la variable EstConfInicial pasara de false a true, cuando haya un cero, ya que el circuito físico el botón se encuentra en tierra, al presionar dicho botón se envía un cero lógico lo que se denomina en una entrada pullup. Como se puede apreciar en la siguiente figura. 33

*Gráfico 26* Modo ADHOC

```
bool EstConfInicial = false;
delay(3000);
//Esta condicion hara poner la ESP en modo AdHoc y se creara un servidor WEB
//Con el formulario HTML q se declaro en la parte de arriba
//
if (digitalRead(PinBoton) == 0) { //Si se lee una entrada en alta La bandera EstConfInicial cambia a alto y nos pone en la espera para recibir datos via ESP
  EstConfInicial = true;
}
if (EstConfInicial == true) {

  modoconf();
  delay(100);
```

**Elaboración:** Joselyn Quimis Suarez
**Fuente:** Arduino IDE

Una vez realizado esta condición entra a la función modoconf(), se ejecutará el código para poder prender y apagar el led de red, con la función digitalWrite que permite escribir los valores lógicos digitales en un pin de salida de una tarjera de Arduino, sumando con el código donde guarda la información del SSID y contraseña de la red Wifi. Como se puede visualizar en el siguiente grafico 33.

*Gráfico 27* Led de red encendido

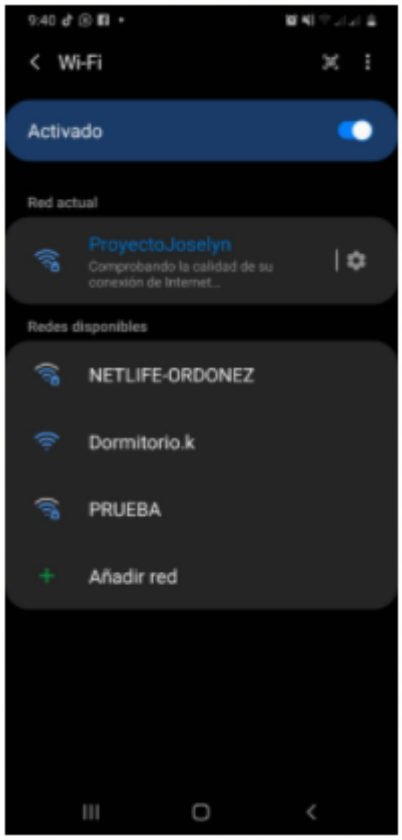
```
void modoconf() {

  delay(100);
  digitalWrite(PinLedRed, HIGH);
  delay(100);
  digitalWrite(PinLedRed, LOW);
  delay(100);
  digitalWrite(PinLedRed, HIGH);
  delay(100);
  digitalWrite(PinLedRed, LOW);
  delay(100);
  digitalWrite(PinLedRed, HIGH);
  delay(100);
  digitalWrite(PinLedRed, LOW);
  delay(100);
  digitalWrite(PinLedRed, HIGH);
  delay(100);
  digitalWrite(PinLedRed, LOW);

  WiFi.softAP(ssidConf, passConf);
  IPAddress myIP = WiFi.softAPIP();
  Serial.print("IP del acces point: ");
  Serial.println(myIP);
  Serial.println("WebServer iniciado...");

  server.on("/", paginaconf); //esta es la pagina de configuracion
```

**Elaboración:** Joselyn Quimis Suarez
**Fuente:** Arduino IDE

### 2.1.3.7. ESP como hostpot y web server

Este proceso permite que la ESP32 actúe como punto de acceso, conectándose a una red llamada "proyecto joselyn" y permite guardar la red de wifi, entrando a una dirección ip que es 192.168.4.1. A continuación se muestra en el siguiente gráfico 34.

*Gráfico 28* Red Wifi




**Elaboración:** Joselyn Quimis Suarez
**Fuente:** Samsumg SmartPhone

### 2.1.3.8. Ingreso de datos de red por formulario-formulario de inicio

Este formulario se desplegará al momento de pulsar el botón de la ESP32 llamado EN o reset, en el cual se creará una red WiFi, en la red 192.164.4.1, en el cual se escribirá los datos de la red SSID y Password de la red disponible. En la cual se muestra en las siguientes figuras 36 y 37.

*Gráfico 29* Web Server Formulario


**Elaboración:** Joselyn Quimis Suarez
**Fuente:** Chrome

Como se muestra en el gráfico 36, se realiza el proceso de llenar el formulario con los datos de la red Wifi que este en alcance, una vez conectándose al internet, el monitor serial indicara los valores del sensor y GPS, a su vez se envían en la plataforma Cayenne y red social Facebook

*Gráfico 30* Formulario de Inicio


**Fuente**: Arduino IDE
**Elaboración:** Joselyn Quimis Suarez

Esta codificada con código HTML, al presionar el botón **submit** que es guardar se aplica el método Get entra a una validación y se guarda la información en una memoria EPROM, ya que es una memoria no volátil.

### 2.1.3.9. Guardar los datos EEPROM

En este proceso se ejecuta las funciones guardar_conf() y grabar(), en la cual guardar la información ingresada de la red, en el cual, la línea de código serial.println(server.arg(ssid)) recibe los valores enviadas por la función get del botón guardar dentro del formulario web. Como se puede visualizar en el siguiente gráfico 37.

*Gráfico 31* Guardar datos EEPROM

```
//--------------------GUARDAR CONFIGURACION-----------------------
void guardar_conf() {
  Serial.println("Los datos recogidos son>>");
  Serial.println(server.arg("ssid"));//Recibimos los valores que envia por GET el formulario web
  grabar(0, server.arg("ssid"));

  Serial.println(server.arg("wifiPassword"));
  grabar(50, server.arg("wifiPassword"));
  mensaje = "Configuracion Guardada...";
  paginaconf();
  delay(100);
  ESP.restart(); //Funcion de reinicio de programa
}

//---------------Función para grabar en la EEPROM-----------------
void grabar(int addr, String a) {
  int tamano = a.length();
  char inchar[50];
  a.toCharArray(inchar, tamano + 1);
  for (int i = 0; i < tamano; i++) {
    EEPROM.write(addr + i, inchar[i]);
  }
  for (int i = tamano; i < 50; i++) {
    EEPROM.write(addr + i, 255);
  }
  EEPROM.commit();
}
```

**Elaboración:** Joselyn Quimis Suarez
**Fuente**: Arduino IDE

### 2.1.3.10. En caso de no presionar botón-leer datos de red de la memoria EEPROM

En esta función se leerá la información ingresada, que esta guardada en la memoria EEPROM de acuerdo con la línea de código usada EEPROM.read(i), significa que la i está dentro de un ciclo for, en la cual esta

variable contiene la información y será leída por la memoria EEPROM.read. cómo se puede visualizar en el siguiente gráfico 38.

*Gráfico 32* Leer datos EEPROM

```
//-----------------Función para leer la EEPROM-----------------------
String leer(int addr) {
  byte lectura;
  String strlectura;
  for (int i = addr; i < addr + 50; i++) {
    lectura = EEPROM.read(i);
    if (lectura != 255) {
      strlectura += (char)lectura;
    }
  }
  return strlectura;
}
```

**Elaboración:** Joselyn Quimis Suarez
**Fuente:** Arduino IDE

### 2.1.3.11. Se conecta a la red WiFi y cayenne

Una vez leyendo el proceso anterior, podrá conectarse a la red wifi y también a la plataforma Cayenne, el programa va hacer que se encienda el led red y en el caso de que se desconecte como no está guardado va ir de nuevo a la función de leer la memoria EEPROM hasta que pueda conectarse. A continuación, se muestra en el siguiente gráfico 39.

*Gráfico 33* Conexión red WiFi y Cayenne

```
} else {
  leer(0).toCharArray(ssid, 50);//String SSID convertido a CharArray
  leer(50).toCharArray(wifiPassword, 50);//string CLAVE convertido a CharArray
  Serial.println("Los datos son>>>>" + leer(0) + leer(50));
  Serial.println(ssid);
  Serial.println(wifiPassword);
}
//Si todo va bien con la coneccion al Wifi, se conectara a el Broker MQTT
//de Cayenne.El led se encendera indicando que se conecto a internet
Cayenne.begin(username, password, clientID, ssid, wifiPassword);
bool con = Cayenne.isConect();
if (con) {
  Serial.println("Me conecte a NET");
  digitalWrite(PinLedRed, HIGH);
  leerUbicacion();
  Serial.println("Sali de leer Ubicacion");
} else {
  Serial.println("NO pude conectar a NET");
  digitalWrite(PinLedRed, LOW);
}
```

**Elaboración:** Joselyn Quimis Suarez
**Fuente:** Datos de Investigación

### 2.1.3.12. Void loop ()

En esta función bucle permite ejecutar las funciones cayenne() y check sensor un numero infinito de veces y es la segunda función en ejecutarse dentro del programa. A continuación, se muestra en el siguiente gráfico 40.

*Gráfico 34* Función Void Loop

```
void loop() {
  //int ruidoS = analogRead(D0Mic);
  //  Serial.print("Sensor ruido va> ");
  // Serial.println(ruidoS);

  // adc0 = ads.readADC_SingleEnded(0);
  // Voltage = (adc0 * 0.1875) / 1000;

  //Serial.print("El voltaj es> ");
  //Serial.println(Voltage);
  //delay(1000);

  Cayenne.loop();
  checkSensor();
  delay(200);

}
```

**Elaboración:** Joselyn Quimis Suarez
**Fuente:** Datos de Investigación

### 2.1.3.13. Comienza a leer datos del sensor y GPS-función checksensor

Esta función permite analizar si hay un ruido antes que se envíen los mensajes a Cayenne, ya que esta plataforma envía cada 20 segundos, por lo que, si en ese tiempo hay un ruido alto, checkSensor recupera por que realiza el cálculo cada 250 milisegundos, permitiendo que también pueda medir el nivel de ruido según su voltaje y avisando el nivel de ruido a la plataforma Cayenne. Como se muestra en el siguiente gráfico 41.

*Gráfico 35* Función CheckSensor

```
void checkSensor()
{
  String msjto="";

  unsigned long currentMillis = millis();
  // revisa cada 1/4 de segundo
  if (currentMillis - previousMillis >= 250) {
    adc0 = ads.readADC_SingleEnded(0);
    Voltage = (adc0 * 0.1875) / 1000;

    Serial.print("El voltaj es> ");
    Serial.println(Voltage);
    ruidoS = fmap(Voltage, 0.03, 1.45,  47.0,  86.0);
    Serial.print("Sensor ruido va> ");
    Serial.println(ruidoS);
    if (ruidoS > 65) {
      digitalWrite(LedRuido, HIGH);
      delay(250);
     leerUbicacion();
      Serial.print("Se va a enviar datos");
      Cayenne.virtualWrite(0, ruidoS, "snr", "db");
      Cayenne.virtualWrite(1, flat);
      Cayenne.virtualWrite(2, flon);
      Cayenne.virtualWrite(3, altitud);
      msjto= SolicitudWebHook(urlWebHook);
    } else digitalWrite(LedRuido, LOW);
    previousMillis = currentMillis;
  }
}
```

**Elaboración:** Joselyn Quimis Suarez
**Fuente:** Arduino IDE

Descripción del código dentro de la función checkSensor. A continuación, se muestra en el siguiente cuadro 18.

**Cuadro 8** *Código checkSensor*

| Código | Descripción |
|---|---|
| ADS1115 | Las líneas de código que pertenecen a esta placa que están codificadas dentro de esta función son adc0=ads.readADC_singleEnder (0) es una función para leer el sensor que está definida por la librería del conversor ADS1115 y el |

| | |
|---|---|
| | código Voltaje =(ads0*0.1875)/1000, esta línea permite realizar el cálculo del voltaje en el sensor. |
| Fmap | Realiza un mapeo de un valor mínimo y valor máximo a un rango dado, por lo que la línea de código ruidoS = fmap(Voltage, 0.03, 1.45, 47.0, 86.0); permite dar el nivel de ruido según el voltaje. |
| If(ruidos>65)<br>digitalWrite (LedRuido, HIGH) | Cuando se cumpla esta condición se prendera el led indicando ruido, ya que está sobrepasando el nivel normal de ruido que es 65 dB |
| leerUbicacion(); | Lee la ubicación es una función del módulo GPS, permitiendo que pueda actualizar los datos y leerlos, ya que estas serán enviadas a Cayenne y la red social Facebook |
| Cayenne,virtualWrite () | Se envían los datos como formatos parámetros, en el cuál irá el nivel de ruido, y las variables del GPS lóngitud, Latitud, elevación. |
| msjto= SolicitudWebHook(urlWebHook); | Mensaje de aceptación del servidor de IFTTT |

Fuente: Datos de Investigación
Elaboración: Joselyn Quimis Suarez

### 2.1.3.14. Función leer ubicación

En esta función se ejecutará todo el proceso del GPS, que es útil para el proyecto, su comunicación es serial para que pueda conectarse con el microcontrolador ESP-32, por lo que, todo este proceso permitirá dar la ubicación actual, en que se encuentra el prototipo y a su vez ayudará que se pueda visualizar en la plataforma Cayenne
. A continuación, se muestra en el siguiente gráfico 42.

*Gráfico 36* Función LeerUbicacion

```
void leerUbicacion() {
  bool newData = false;
  unsigned long chars;
  unsigned short sentences, failed;

  // For one second we parse GPS data and report some key values
  for (unsigned long start = millis(); millis() - start < 1000;)
  {
    while (Serial2.available())
    {
      char c = Serial2.read();
      // Serial.write(c); // uncomment this line if you want to see the GPS data flowing
      if (gps.encode(c)) // Did a new valid sentence come in?
        newData = true;
    }
  }

  if (newData)
  {
    // unsigned long age;

    gps.f_get_position(&flat, &flon);
    Serial.print("LAT=");
    Serial.print(flat == TinyGPS::GPS_INVALID_F_ANGLE ? 0.0 : flat, 6);
    Serial.print(" LON=");
    Serial.print(flon == TinyGPS::GPS_INVALID_F_ANGLE ? 0.0 : flon, 6);
    Serial.print(" ALTITUDE=");
```

**Elaboración:** Joselyn Quimis Suarez

**Fuente**: Arduino IDE

Descripción del código dentro de la función LeerUbicacion, a continuación, se muestra en el siguiente cuadro 19.

**Cuadro 9** *Código Función LeerUbicacion*

| Código | Descripción |
|---|---|
| while (Serial2.available())<br>char c = Serial2.read(); | Mientras esté disponible serial2 que es la conexión de GPS y la ESP-32, se leerá la serial 2 y se guardará en una variable c . |
| if (gps.encode(c))<br>newData = true; | En esta línea se encuentra una condición en la cual indica la posición y velocidad angular del GPS. Por lo que si se cumple esta condición existe un nuevo dato, permitiendo que pueda ubicarse en el sector que se encuentra |
| if (newData) | Se abre otra condición en la cual se afirma un nuevo dato y se hará el siguiente proceso |

| | |
|---|---|
| gps.f_get_position(&flat, &flon); | Esta función permite que pueda posicionar los valores de longitud y latitud. |
| Serial.print("LAT="); Serial.print(flat == TinyGPS::GPS_INVALID_F_ANGLE ? 0.0 : flat, 6); Serial.print(" LON="); Serial.print(flon == TinyGPS::GPS_INVALID_F_ANGLE ? 0.0 : flon, 6); Serial.print(" ALTITUDE="); Serial.print(gps.f_altitude() == TinyGPS::GPS_INVALID_F_ALTITUDE ? 0 : gps.f_altitude()); altitud = gps.f_altitude(); | Este proceso se ejecuta para validar todas las varias de latitud y longitud, llamando a la biblioteca TinyGPS. Mientras que la altitud usa su propia función |
| gps.stats(&chars, &sentences, &failed); Serial.print(" CHARS="); Serial.print(chars); Serial.print(" SENTENCES="); Serial.print(sentences); Serial.print(" CSUM ERR="); Serial.println(failed); | Caso contrario de que el GPS no envíe señal al satélite aparece estas variables chars,sentences, csum err |

**Elaboración:** Joselyn Quimis Suarez
**Fuente:** Datos de Investigación

### 2.1.3.15. Función solicitudwebhook

Es una función string que va a recibir como para metro la url "http://maker.ifttt.com/trigger/cayenne/with/key/bhS5lQg9JFaiqpuquCuxHu" definido por webhook en IFTTT, permitiendo comunicar con el servidor de esta plataforma y a su vez en Facebook. Por lo que aparece la ubicación dentro de la red social indicando un alto nivel de ruido. A continuación, se muestra en los siguientes gráficos 43 y 44.

*Gráfico 37* Función SolicitudWebHook

```
String SolicitudWebHook(String UrlReq) {
  /*
  float ruidoS = 0;
  float flat, flon, altitud;
  */
  //json a enviar
  //{"value1":"66","value2":-2.227724620217205,"value3":-79.93263052814042}
  String payload;
  String dt;
  DynamicJsonDocument doc(1024);
  doc["value1"] = ruidoS;
  doc["value2"] = flat;
  doc["value3"] = flon;
  serializeJson(doc, dt);
  serializeJson(doc, Serial);


  Serial.print("[HTTP] begin...\n");
  if (http.begin(client, UrlReq)) {  // HTTP
    Serial.print("[HTTP] POST...\n");
    // start connection and send HTTP header
    http.addHeader("Content-Type", "application/json");
    //http.addHeader("Content-LENGTH", String(dt.length()));
    int CodHttp = http.POST(dt);
    Serial.println(CodHttp);
```

**Elaboración:** Joselyn Quimis Suarez
**Fuente:** Arduino IDE

*Gráfico 38* Solicitud HTTP

```
if (http.begin(client, UrlReq)) {  // HTTP
  Serial.print("[HTTP] POST...\n");
  // start connection and send HTTP header
  http.addHeader("Content-Type", "application/json");
  //http.addHeader("Content-LENGTH", String(dt.length()));
  int CodHttp = http.POST(dt);
  Serial.println(CodHttp);

  // httpCode will be negative on error
  if (CodHttp == 200) {
    // HTTP header has been send and Server response header has been handled

    Serial.printf("[HTTP] POST... code: %d\n", CodHttp);
    payload = http.getString();
    Serial.println(payload);

  } else {
    Serial.printf("[HTTP] POST... Fallo, error: %s\n", http.errorToString(CodHttp).c_str());
    Serial.printf("[HTTP] NO conecion\n");
    payload = "";

  }
  http.end();
 else {
  Serial.printf("[HTTP] NO conecion\n");
```

**Elaboración:** Joselyn Quimis Suarez
**Fuente:** Arduino IDE

Descripción del código dentro de la función SolicitudWebHook. A continuación, se muestra en el siguiente cuadro 20.

Cuadro 10 *Código SolicitudWebHook*

| Código | Descripción |
|---|---|
| Json | Es un formato para trabajar en diferentes Frimeworks. |
| Variables-Json<br>{"value1":"66","value2":-2.227724620217205,"value3":-79.93263052814042} | Estas variables se van usar, ya que es recomendado por IFTTT para que pueda comunicarse con esta plataforma y a su vez presente el grafico del mapa en Facebook. |
| DynamicJsonDocument doc(1024) | Almacena un documento Json en la memoria junto con la capacidad especificadas. |
| Doc[“value 1”]=ruidos<br>Doc[“value 2”]=float<br>Doc[“value 3”]=flon | Las variables Json se evalúan junto con las variables de GPS y las variables de checkSensor. |
| serializeJson(doc, dt);<br>serializeJson(doc, Serial); | Propiedad de un objeto Json que convierte en bites de un lado y del otro lado mantiene su versión original. Se pueden serializar el JsonDocument mediante una sobrecarga de métodos serializeJson(), por lo que envía los datos doc y deeserializar un fichero Json a un objeto mediante una de las sobrecargas del método deserializeJson() que devuelve un valor indicando si la conversión se ha realizado correctamente. |

Fuente: Datos de Investigación
Elaboración: Joselyn Quimis Suarez

Solicitud de servidor dentro de la función SolicitudWebHook. Como se puede apreciar en el siguiente cuadro 21.

**Cuadro 11** *Solicitud función SolicitudWebHook*

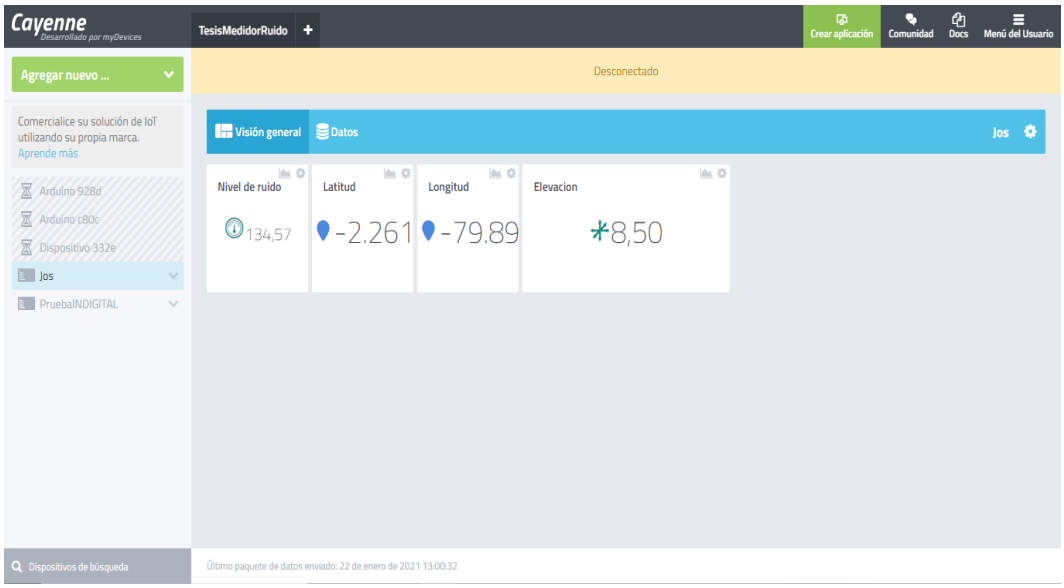
| Código | Descripción |
|---|---|
| if (HTTP. Begin(client, UrlReq)) | Se inicia Http haciendo el uso de la biblioteca ya declarada, si cumple la condición que es cliente, se envía la url. |
| http.addHeader("Content-Type", "application/json"); | Se añade una cabecera y se envía un body Json |
| int CodHttp = http.POST(dt); | Esta línea significa la respuesta que da el servidor |
| if (CodHttp == 200) | Se realiza una validación con el puerto 200 que es de IFTTT, dado el caso seguirá ejecutándose el programa |
| payload = http.getString(); | En esta línea se lee la carga con el método get. |
| Serial.printf("[HTTP] POST... Fallo, error: %s\n", http.errorToString(CodHttp).c_str()); | Caso contrario el programa cierra conexión |

**Elaboración**: Joselyn Quimis Suarez
**Fuente:** Datos de Investigación

### 2.1.3.16. Plataforma cayenne

En esta plataforma se mostrará de forma más didáctica creando paneles de los valores de ruido y los datos de GPS. A continuación, se muestra en el siguiente gráfico 45.

*Gráfico 39* Plataforma Cayenne Proyecto




**Elaboración:** Joselyn Quimis Suarez
**Fuente:** Cayenne My Device

### 2.1.3.17. Plataforma IFTTT

Esta plataforma va a permitir que se puede visualizar de manera más didáctica en la red social Facebook, ya que esta plataforma es amigable y es usada para automatización de dispositivos, casas, etc. A continuación, se muestra en el siguiente gráfico 46.

*Gráfico 40* Creación de Applets


**Elaboración:** Joselyn Quimis Suarez
**Fuente:** IFTTT

En esta gráfica 47. se muestra la creación de APPLETS o receta en la cual se configuro de acuerdo a su guía en IFTTT por lo que el código también se encuentra de igual manera.

*Gráfico 41* Configuración de Applets


**Elaboración:** Joselyn Quimis Suarez
**Fuente:** IFTTT

Para que escuche y reciba se configuro el webhooks, la cual tiene por nombre Cayenne y se activa para que envié datos en Facebook. A continuación, se muestra en el siguiente grafico 48.

*Gráfico 42* Link para mostrar mapa

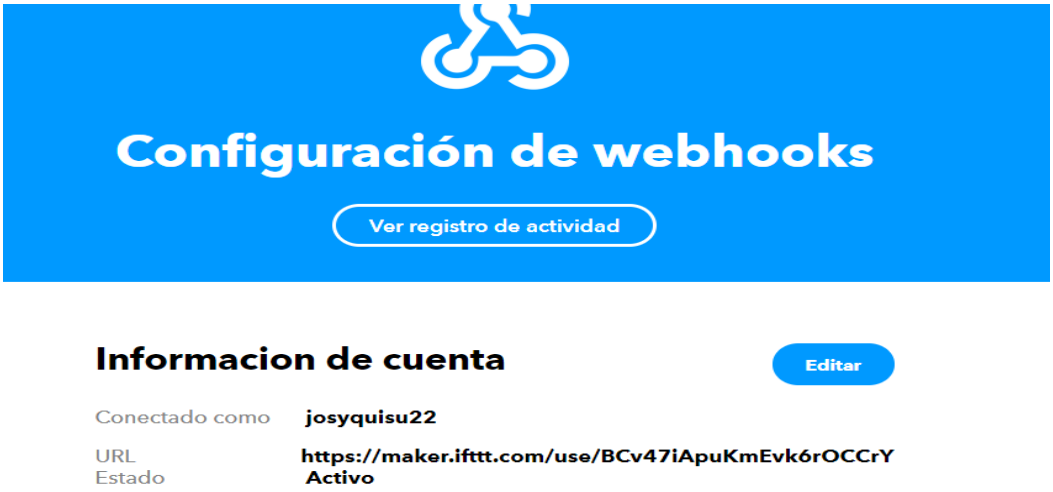

**Elaboración:** Joselyn Quimis Suarez
**Fuente:** IFTTT

En esta figura se muestra el link utilizado en la codificación, el cual pertenece a IFTTT y este permite mostrar el mapa en ubicación actual.

### 2.1.3.18. WEBHOOKS

Dentro de la plataforma IFTTT se realizan la configuración de webhooks, el cual nos permite comunicar e intercambiar mensajes con la ESP-32 o Arduino, este servicio web espera una petición o llamada a una URL concreta. A continuación, se muestra en el siguiente gráfico 49.

*Gráfico 43* Configuración de Webhooks




**Elaboración:** Joselyn Quimis Suarez

Como se puede visualizar en el gráfico 49, esta enlazado con la cuenta personal y el link es la clave que permite comunicarse y enviar datos de la ESP 32 a IFTTT.

Como se muestra en este grafico 51, se puede visualizar el mapa y el mensaje de nivel de ruido enviado por el prototipo, de esta forma hace que el proyecto sea más dinámico y fácil de administrar.

### 2.1.4. Etapa pruebas y error

Se realizo las respectivas pruebas tanto código como el prototipo físico para el correcto funcionamiento de este y de todos sus componentes, llegando a la conclusión, de que el sistema se encuentra óptimo.

## 3. COMPONENTES DE MEDICIÓN DE SEÑALES DE RUIDO.

### 3.1.1. Etapa validación y verificación del producto de software:

En este proceso se realizó las respectivas mediciones con el prototipo experimental de contaminación de señales acústicas en los diferentes sectores de la ciudad.

#### 3.1.1.1. Medición de contaminación acústica

Luego de haber terminado de construir el prototipo se lo colocó en área de prueba obteniendo las siguientes mediciones.
, como se muestra en el siguiente cuadro 22

**Cuadro 12** *Medición de contaminación acústica*

| Hora | Lunes | Martes | Miércoles | Jueves | Viernes | Promedio de ruido |
|---|---|---|---|---|---|---|
| 12h00 | 71,22dB | 128.29dB | 76,13dB | 70,65dB | 80.29dB | 85.31dB |
| 15h00 | 72.85dB | 70.86dB | 80,36dB | 84,61dB | 71.44dB | 76 dB |
| 17h00 | 78,03dB | 129.48dB | 79,61dB | 132,39dB | 78.10dB | 99.52dB |
| 19h00 | 70.22dB | 107.70dB | 71,22dB | 90,20dB | 129.82 | 93.83dB |

**Elaboración:** Joselyn Quimis Suarez
**Fuente:** Datos de investigación

Dentro de este cuadro, se puede realizar el análisis en el cual se muestran horas de mayor ruido en el día, que perjudican a los usuarios . Con la evaluación de

#### 3.1.1.2. MANTENIMIENTO:

El prototipo experimental, podrá en el futuro, realizar mejoras si fuera

## CAPÍTULO IV

### FUNCIONAMIENTO DEL PROTOTIPO

Los resultados del prototipo muestran un uso eficiente de funcionamiento del sistema .

A continuación se muestran las mediciones realizadas en tiempo real con el prototipo construído.

| Hora | Lunes | Martes | Miércoles | Jueves | Viernes | Promedio de ruido |
|---|---|---|---|---|---|---|
| 12h00 | 71,22dB | 128.29dB | 76,13dB | 70,65dB | 80.29dB | 85.31dB |
| 15h00 | 72.85dB | 70.86dB | 80,36dB | 84,61dB | 71.44dB | 76 dB |
| 17h00 | 78,03dB | 129.48dB | 79,61dB | 132,39dB | 78.10dB | 99.52dB |
| 19h00 | 70.22dB | 107.70dB | 71,22dB | 90,20dB | 129.82 | 93.83dB |
| Servicio | | | Aceptación | | | |

Fuente: Datos de Investigación

Elaboración: Joselyn Quimis Suarez

El prototipo es funcional y permite medir niveles de ruido en un entorno específico.

.

El prototipo permite encontrar la ubicación en que se genera el ruido, mostrando al usuario final la información en la plataforma Cayenne. Los componentes implementados tienen alto grado de confiabilidad .

El microcontrolador ESP32 debe complementarse con el conversor ADS 1115 para que ayudara a leer los voltajes mínimos importantes en la captación de ruidos.

En el módulo GPS también se recomienda tener cerca para su rápida conexión o un lugar donde no haya obstáculos que bloquee el cielo y pueda afectar la conexión del GPS o ningún factor ambiental que afecta la conexión del GPS al satélite.

# REFERENCIAS


Alonso Díaz, J. A. (2014). Resultados de la aplicación del protocolo de ruido en trabajadores expuestos a un nivel de ruido continuo diario equivalente igual o superior a 85 decibelios (A). *Medicina y Seguridad del Trabajo*, *60*(234), 9-23.

AMBIENTAL, C. G. D. P. (2017). Ministerio del Ambiente. *Ordenamento do Território e*.

Basco Prado, L., Fariñas Rodríguez, S., & Hidalgo Blanco, M. Á. (2010). Características del sueño de los pacientes en una unidad de cuidados intensivos. *Revista Cubana de Enfermería*, *26*(2), 0-0.

Benito Herranz, Á. (2019). Desarrollo de aplicaciones para IoT con el módulo ESP32.

Bodenhorts, M., & Carlos, J. (2014). *Diseño e implementación de un prototipo de medición acústica remoto*. Quito: Universidad de las Américas, 2014.,

Borja Vega, E. R. (2020). Diseño de una arquitectura usando el protocolo Message Queue Telemetry Transport (MQTT) sobre plataformas de bajo coste, para monitorización de procesos industriales.

Carbonel Martínez, A. Diseño e implementación de un middleware CoAP-MQTT-HTTP para la mejora de la interoperabilidad de los protocolos de aplicación en redes IoT.

de la Salud, A. M. (2017). *Prevención de la sordera y la pérdida de audición: informe de la Secretaría*. Retrieved from

DEL ECUADOR, C. C. (2019). DEL ECUADOR. *Recuperado el*, *15*.

Domínguez Ruiz, A. L. M. (2015). El poder vinculante del sonido: La construcción de la identidad y la diferencia en el espacio sonoro. *Alteridades*, *25*(50), 95-104.

dos Santos, O. L., & Delfino, L. R. (2019). UM PROJETO DE SALA DE AULA INTELIGENTE PARA A FAESA COM O USO DA INTERNET DAS COISAS E MQTT. *Revista Científica FAESA*, *15*(2 Especial), 121-142.

Estrada-Rodríguez, C., & Ramírez, I. M. (2010). Impacto del ruido ambiental en estudiantes de educación primaria de la Ciudad de México. *Revista Latinoamericana de Medicina Conductual/Latin American Journal of Behavioral Medicine*, *1*(1), 57-68.

Ferreyra, S. P., Cravero, G. A., Longoni, H. C., López, J. F., Parada, M. F., & Díaz, M. S. (2019). Calidad Acústica de Aulas Universitarias: Análisis y Evaluación de Parámetros Acústicos de Recintos. *Revista Tecnología y Ciencia*(35), 41-61.

Friedrich, G. R., Giron, P., Reggiani, G., Azurro, A., Coppo, R., Sequeira, M., . . . Cofre, L. (2017). *Procesamiento de señales vibro-acústicas: análisis de casos de estudio, modelación, prototipado y experimentación.* Paper presented at the XIX Workshop de Investigadores en Ciencias de la Computación (WICC 2017, ITBA, Buenos Aires).

Gavilánez Mejía, L. A. (2017). *Derechos de autor y limitaciones de la ley de propiedad intelectual en el Ecuador.* Universidad Técnica de Ambato, Facultad de Jurisprudencia y Ciencias …,

Guillen-Mendoza, C., Ramos Martín, A., & Santana Rodríguez, J. J. (2016). Diseño experimental de un regulador PID de bajo costo para complementar el aprendizaje en control de procesos.

Gutiérrez Jaramillo, M. M. (2015). Ruido Rosa: consolidando mi identidad vocal y artística.

Lee, S., Kim, H., Hong, D.-k., & Ju, H. (2013). *Correlation analysis of MQTT loss and delay according to QoS level.* Paper presented at the The International Conference on Information Networking 2013 (ICOIN).

Leyva Perez, O. A. Monitoreo de ruido en sitios cerrados.

López Molinero, J. (2018). Implementación del protocolo MQTT-S sobre IEEE 802.15. 4e en plataformas OpenMOTE.

Luján Cuenca, V. (2017). Sistema de sensores integrado en equipación de Bomberos.

Mahedero Biot, F. (2020). *Desarrollo de una aplicación IoT para el envío de imágenes mediante el protocolo MQTT.*

Mankar, J., Darode, C., Trivedi, K., Kanoje, M., & Shahare, P. (2014). Review of I2C protocol. *International Journal of Research in Advent Technology, 2*(1).

Mejía Saca, D. G. (2018). *Implementación, calibración y evaluación de un sistema de medición de ruido con un sensor de sonido que permita la transmisión inalámbrica de datos.* Universidad del Azuay,

Mendez Ortiz, L. F., & Ruiz Escobar, M. (2019). *Diseño de guías de trabajo para prácticas de laboratorio para la asignatura seguridad y salud en el trabajo.* Universidad de Ibagué. Facultad de Ingeniería. Ingeniería Industrial,

Montilla Pérez, A. (2018). Pasarela sobre Raspeberry Pi entre intermediario MQTT y plataforma DBaas.

Moreno Cerdà, F. (2018). *Demostrador arquitectura publish/subscribe con MQTT.* Universitat Politècnica de Catalunya,

Novillo-Ortiz, D., D'Agostino, M., & Becerra-Posada, F. (2016). El rol de la OPS/OMS en el desarrollo de capacidad en eSalud en las Américas: análisis del período 2011-2015. *Revista Panamericana de Salud Pública, 40*, 85-89.

Oberg, J., Hu, W., Irturk, A., Tiwari, M., Sherwood, T., & Kastner, R. (2011). *Information flow isolation in I2C and USB.* Paper presented at the 2011 48th ACM/EDAC/IEEE Design Automation Conference (DAC).

Otiniano López, M. F. (2018). Sistema de Medición Acústica usando NODEMCU ESP8266 para Determinar el Nivel de Ruido en Av. Víctor Larco cuadra 14 Trujillo 2018.

Pedrera, A. C. (2017). *Arduino para Principiantes: 2ª Edición*: IT Campus Academy.

Quiñonez, Y., Lizarraga, C., Peraza, J., & Zatarain, O. (2019). Sistema inteligente para el monitoreo automatizado del transporte público en tiempo real. *RISTI-Revista Ibérica de Sistemas e Tecnologias de Informação*(31), 94-105.

Racero Valcárcel, A. R. (2016). Integración de robot social en sistema domótico.

Robledo, F. H. (2014). *Riesgos físicos I: ruido, vibraciones y presiones anormales*: Ecoe Ediciones.

Ruiz, G. I. (2013). Aplicaciones del sistema de información geo referenciado en el Ecuador. *Yachana Revista Científica, 2*(2).

Sanjuanero, A. N. (2012). *Evaluación de los niveles de ruido en una unidad de cuidados intensivos neonatales.* Universidad Autónoma de Madrid,

Serna, A., Ros, F., & Rico, J. (2010). *Guía práctica de sensores*: Creaciones Copyright SL.

Toribio, L. A., Aranguren, D. C., Ruiz, D. M., & Maqueda, M. J. R. (2011). Ruido ambiental: seguridad y salud. *Tecnología y desarrollo, 9*, 31.

Trigás Gallego, M. (2012). Metodologia scrum.

Varela Neila, E. (2016). Método de desarrollo en serie mediante Montecarlo: comparación con aproximaciones markovianas efectivas.

Vélez, D. P. (2020). *Diseño de un dispositivo wearable para el monitoreo de la oxigenación y ritmo cardiaco.* Paper presented at the Memorias del Congreso Nacional de Ingeniería Biomédica.

Vélez González, E. A. (2016). *Implementación de un prototipo de medición de ruido con geoposicionamiento.* Quito: Universidad de las Américas, 2016,

Verde, L. (2017). Contaminación Acústica. *Obtenido de Contaminación Acústica: http://www. lineaverdeceutatrace. com/lv/consejos-ambientales/contaminacionacustica/contaminacion-acustica. pdf*.

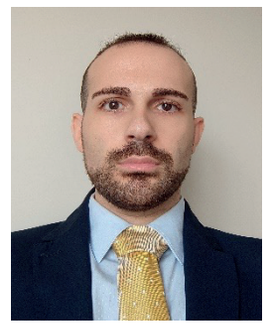
Zabala, M., Cuenca, L., León, J., & Cabrera, F. (2018). Arquitectura de acoplamiento entre INS/GPS para navegación precisa en trayectorias establecidas. *Maskay, 8*(1), 20-26.

**MIGUEL GIOVANNY MOLINA VILLACÍS**
https://orcid.org/0000-0002-7080-2354
UNIVERSIDAD DE GUAYAQUIL
Ingeniero en Electrónica y Telecomunicaciones, graduado en la Escuela Superior Politécnica del Litoral, Máster en Telecomunicaciones de la Escuela Superior Politécnica del Litoral, Máster en Administración de Empresas de la Universidad Técnica Federico Santa María de Chile. Director de Proyecto de Investigación de la Universidad de Guayaquil . Actualmente, se desempeña como docente de la Universidad de Guayaquil, Facultad de Ciencias Matemáticas y Físicas. Investigador y autor de artículos científicos de impacto mundial y regional.

**XIMENA CAROLINA ACARO CHACÓN**
https://orcid.org/0000-0001-7092-7248
UNIVERSIDAD DE GUAYAQUIL
Ingeniera en Electrónica y Telecomunicaciones, Máster Universitario en Ingeniería Electrónica. Actualmente, se desempeña como docente de la Universidad de Guayaquil, Facultad de Ciencias Matemáticas y Físicas. Investigadora y autora de artículos científicos de impacto mundial y regional.

**MARIA FERNANDA MOLINA MIRANDA.**
ORCID: https://orcid.org/0000-0002-4237-4364,
CORREO INSTITUCIONAL: maria.molinam@ug.edu.ec
Ingeniera en Telemática graduada de ESPOL (2011), Máster Universitario en Ingeniería en redes y servicios telemáticos de la Universidad Politécnica de Madrid (2015) y desde el 2020 doctorando en la Universidad de Granada en el área de TIC. Desde 2016 se ha desempeñado como Docente, directora de tesis e investigadora en la Universidad de Guayaquil, Facultad de Ciencias Matemáticas y Físicas, carreras Ingeniería en networking y telecomunicaciones y Tecnologías de la Información.
Ha realizado publicaciones en congresos y revistas nacionales con temas relacionados a Internet de las Cosas.

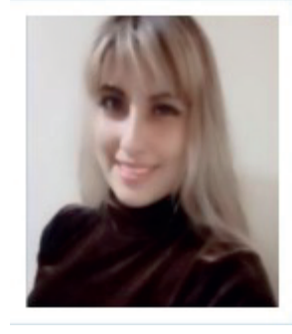
**PIETRO CORAPI**
https://orcid.org/0000-0001-8626-0134
UNIVERSIDAD DE GUAYAQUIL
Máster en Ingeniería Civil, graduado en la Università della Calabria (Italia), se desempeña como docente de grado y postgrado en Carrera de Ingeniería Civil, Facultad de Ciencias Matemáticas y Físicas de la Universidad de Guayaquil. Investigador y autor de artículos científicos de impacto mundial y regional. Miembro del International Association for Hydro-Environment Engineering and Research. Ingeniero consultor en el área de Hidráulica y Sanitaria.



**LUIS ARTURO ESPÍN PAZMIÑO.**
ORCID: https://orcid.org/0000-0002-1663-2489,
CORREO INSTITUCIONAL: luis.espinp@ug.edu.ec

Magister en Administración de Empresas con mención en Telecomunicaciones, Ingeniero en Telecomunicaciones con mención en Gestión Empresarial, Diploma Superior en Diseño Curricular por Competencias, actualmente docente titular en la Universidad de Guayaquil en las carreras de Tecnologías de la Información y en Networking y Telecomunicaciones.

**JOSELYN STEFANIA QUIMIS SUÁREZ.**
Ingeniera en Networking y Telecomunicaciones, graduada en la Facultad de Ciencias Matemáticas y Física en la Universidad de Guayaquil , investigadora en el área de Tecnologías de la Información, experta en redes, lenguajes de programación y sistemas operativos




ISBN: 978-9942-603-28-9