ESCUELA POLITÉCNICA NACIONAL

FACULTAD DE INGENIERIA ELÉCTRICA Y ELECTRÓNICA

APLICACIONES CON SISTEMAS EMBEBIDOS

IMPLEMENTACIÓN DE UN PROTOTIPO PARA LA DETECCIÓN DEL NIVEL DE RUIDO E ILUMINACIÓN DE UN AMBIENTE UTILIZANDO SISTEMAS EMBEBIDOS

TRABAJO DE INTEGRACIÓN CURRICULAR PRESENTADO COMO REQUISITO PARA LA OBTENCIÓN DEL TÍTULO DE INGENIERO EN TELECOMUNICACIONES

MICHAEL ALEXANDER PRADO CAZA
michael.prado@epn.edu.ec

DIRECTOR: SORAYA LUCÍA SINCHE MAITA, PhD.
soraya.sinche@epn.edu.ec

DMQ, abril 2023

CERTIFICACIONES

Yo, Michael Alexander Prado Caza declaro que el trabajo de integración curricular aquí descrito es de mi autoría; que no ha sido previamente presentado para ningún grado o calificación profesional; y, que he consultado las referencias bibliográficas que se incluyen en este documento.

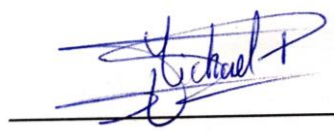
Michael Alexander Prado Caza

Certifico que el presente trabajo de integración curricular fue desarrollado por Michael Alexander Prado Caza, bajo mi supervisión.

Soraya Lucia Sinche Maita, PhD.

DECLARACIÓN DE AUTORÍA

A través de la presente declaración, afirmamos que el trabajo de integración curricular aquí descrito, así como el (los) producto(s) resultante(s) del mismo, son públicos y estarán a disposición de la comunidad a través del repositorio institucional de la Escuela Politécnica Nacional; sin embargo, la titularidad de los derechos patrimoniales nos corresponde a los autores que hemos contribuido en el desarrollo del presente trabajo; observando para el efecto las disposiciones establecidas por el órgano competente en propiedad intelectual, la normativa interna y demás normas.



Michael Alexander Prado Caza

Soraya Lucía Sinche Maita, PhD.

DEDICATORIA

A lo largo de mi existencia han estado presentes mujeres que me han convertido en el hombre que soy ahora, quiero dedicar mi esfuerzo a:

Mi Michita, Mercedes Chimbolema, quien siempre estuvo para mí en cada etapa de mi vida, es el ser humano más noble, valiente, y de infinitas buenas cualidades.

Mi madre, Angélica Caza, quien siempre lucho para sacar a nuestra familia adelante y le debo todo, espero demostrarle cuanto la amo y seguir mejorando cada día.

Mi tía, Cecilia Caza, quien me ha brindado sus cuidados, consejos y protección desde que nací y se ha convertido en otra madre para mí.

Allyson Olaya, quien me ha amado, acompañado y apoyado en mi última etapa universitaria, y me ha hecho entender que la salud mental es importante.

AGRADECIMIENTO

Agradezco a Dios por todo y en especial por bendecirme con la hermosa familia que tengo, en los días malos y buenos siempre me he sentido protegido por su inmenso amor.

A la Escuela Politécnica Nacional, quien me ha brindado las mejores amistades que he podido encontrar, además, de conocimiento y desarrollo personal.

A la PhD. Soraya Sinche, quien siempre me brindo apoyo incondicional para terminar el presente trabajo con éxito.

A Bryan Enriquez, quien me enseño lo que es la verdadera amistad y siempre está mi lado en cada locura que se nos ocurre, ahora siento que es parte de mi familia.

ÍNDICE DE CONTENIDO

.....

ÍNDICE DE FIGURAS

ÍNDICE DE TABLAS

ÍNDICE DE ECUACIONES

ÍNDICE DE CÓDIGOS

RESUMEN

En la actualidad, el desarrollo tecnológico, industrias, tráfico de autos, construcción, entre otros, generan altos niveles de ruido, los cuales pueden llegar a ser peligrosos. Conocer el tipo de ambiente sonoro y lumínico es importante para prevenir riesgos en la salud de las personas y para realizar las actividades diarias de la mejor forma posible.

Por este motivo el presente trabajo de integración curricular tiene como finalidad, implementar un prototipo basado es sistemas embebidos, que detecte el nivel de ruido e iluminación; de esta manera, cualquier usuario con acceso a un dispositivo móvil podrá realizar mediciones del nivel de ruido e iluminación, monitorear estos datos en tiempo real y conocer que niveles son perjudiciales. Además, realizando estas mediciones se pueden crear bases de datos para estudios posteriores que permitan analizar estos datos y correlacionarlos con parámetros como el rendimiento académico de estudiantes o desempeño laboral de personas.

Este documento cuenta con tres capítulos, en el Capítulo 1 se presentan los conceptos relacionados con sistemas embebidos, sensores, ruido e iluminación. El Capítulo 2 incluye el diseño e implementación de un prototipo que mide el nivel de ruido e iluminación. Finalmente, en el Capítulo 3 se realizan pruebas del prototipo en diferentes escenarios y se comparan los datos obtenidos con mediciones realizadas por aplicaciones móviles que miden el nivel de ruido e iluminación., así como las conclusiones y recomendaciones.

PALABRAS CLAVE: Detección de Ruido, Iluminación, Sistemas Embebidos, Sensores.

ABSTRACT

Nowadays, technological development, industries, car traffic, construction, among others, generate high noise levels, which can become dangerous. Knowing the type of sound and light environment is important to prevent risks to people's health and to perform daily activities in the best possible way.

For this reason, the purpose of this curricular integration work is to implement a prototype based on embedded systems, which detects the level of noise and lighting; in this way, any user with access to a mobile device can measure the level of noise and lighting, monitor these data in real time and know which levels are harmful. In addition, these measurements can be used to create databases for subsequent studies to analyze these data and correlate them with parameters such as people work performance or students' academic performance.

This document has three chapters. Chapter 1 presents the concepts related to embedded systems, sensors, noise and lighting. Chapter 2 includes the design and implementation of a prototype that measures the level of noise and lighting. Finally, Chapter 3 tests the prototype in different scenarios and compares the data obtained with measurements made by mobile applications that measure noise level and illumination, as well as conclusions and recommendations.

KEYWORDS: Noise Detection, Lighting, Embedded Systems, Sensors.

1. DESCRIPCIÓN DEL COMPONENTE DESARROLLADO

Hoy en día el ambiente que rodea a las personas dentro de sus casas, oficinas y aulas, puede presentar niveles de ruido e iluminación que causan molestias, incomodidad y afectan a las personas [1]. Exponerse a una luminosidad inadecuada o a altos niveles de ruido puede causar fatiga ocular, cansancio, dolor de cabeza, estrés, entre otros [2].

La pérdida del sentido de la audición, conocida también como hipoacusia, se presenta cada vez de forma más temprana en las personas y se encuentra con frecuencia relacionada al aumento de enfermedades crónicas no transmisibles, las cuales están asociadas a altas exposiciones de nivel de ruido. La Organización Mundial de la Salud (OMS) estima que alrededor de 360 millones de personas en el planeta, presentan algún grado de discapacidad auditiva ocasionada por el ruido [1]. Los estudios relacionados con las actividades de recreación en la juventud muestran una seria afectación en la audición, esto es debido a extensos tiempos de exposición a altos niveles de ruido, como consecuencia hay disminución del rendimiento académico o problemas con los estudios durante la etapa de la juventud [1].

En áreas educativas se debe tener una adecuada iluminación dependiendo de las actividades que se realice. Por ejemplo, lugares donde se realizan labores prácticas como laboratorios, talleres, salas de cómputo, entre otros, deben tener un nivel de iluminación diferente a donde se dictan solamente clases teóricas [2].

Por otra parte, el conjunto de datos obtenidos a través de sensores, hacen posible efectuar estudios en donde se analice el desempeño de las actividades que realizan los usuarios que están expuestos a los factores de ruido y luz. Además, se previene problemas que afecten la salud y bienestar de las personas.

El presente trabajo se enfoca en la detección de los niveles de ruido y luz en un ambiente *indoor*, utilizando sensores y sistemas embebidos, que permitan recopilar información y obtener datos en tiempo real, para posteriormente ser analizados, con el objetivo de establecer sectores donde el ruido y la iluminación pueden llegar a ser un problema.

Al obtener los datos de los niveles de ruido e iluminación, se informará a los usuarios que estén expuestos a estos fenómenos a través de un terminal móvil, si se encuentran en un ambiente idóneo o existe algún tipo de riesgo o daño a la salud.

1.1 Objetivo General

- Implementar un prototipo para la detección del nivel de ruido e iluminación de un ambiente utilizando sistemas embebidos.

1.2 Objetivos Específicos

- Estudiar los fundamentos de sistemas embebidos y sensores de ruido e iluminación.
- Determinar los niveles de ruido e iluminación a los que pueden estar expuestos las personas.
- Desarrollar un prototipo de prueba para la detección de los niveles de ruido e iluminación de un ambiente *indoor*.
- Analizar los resultados obtenidos de las pruebas de funcionamiento del prototipo.

1.3 Alcance

Se presenta un estudio de los conceptos fundamentales de sistemas embebidos y sensores de ruido e iluminación, elaborando resúmenes, mapas conceptuales, tablas, gráficos etc.

Se investigan cuáles son los niveles de ruido e iluminación que afectan a las personas, recopilando información de los últimos cinco años con fuentes de búsqueda confiables como revistas de salud, medicina y publicaciones de la OMS.

Como parte del diseño, se define los componentes a ser utilizados en el prototipo de prueba, donde se analizan los diferentes sensores que existen en el mercado para determinar cuáles son los más adecuados para el prototipo.

Se implementa un prototipo de prueba que permite recopilar y monitorear datos de iluminación y ruido en tiempo real en ambiente *indoor*. Adicionalmente, se genera un *dataset* con los datos obtenidos para analizar esta información. Se implementa un protocolo de pruebas para verificar el funcionamiento del prototipo y se realiza ajustes de ser necesario.

1.4 Marco Teórico

En esta sección se presentan los fundamentos teóricos que son esenciales para entender y llevar a cabo el presente trabajo de integración curricular.

1.4.1 Sistemas Embebidos

Los sistemas embebidos son un conjunto de software y hardware computacional que tienen como objetivo cumplir con una tarea específica, ésta se lleva a cabo usualmente en tiempo real. La característica principal de este tipo de sistemas es realizar una única función dentro de sistemas más complejos; por ejemplo, el sistema interno de un celular.

El hardware de los sistemas embebidos se basa en microprocesadores o microcontroladores, mientras que el software por lo general es simple y requiere poca memoria. Tradicionalmente se puede observar que los sistemas embebidos están interconectados a sensores y/o actuadores [4].

1.4.1.1 Características de los Sistemas Embebidos

Los sistemas embebidos presentan varias características, entre las más importantes se tiene: la memoria, debido a que es un recurso limitado y el CPU (*Central Processing Unit*), que es la parte inteligente del sistema. A continuación, se presenta un listado de estas características.

a) Presenta cuatro tipos de CPU: DSP (*Digital Signal Processor*), un DSC (*Digital Signal Controller*), un microprocesador o un microcontrolador
b) Presenta tres tipos de memoria: SRAM (*Static Random Access Memory*), EEPROM (*Electrically Erasable Programmable Read-Only Memory*) y memoria flash.
c) Cumple un objetivo en concreto.
d) Por lo general son programables.
e) Recolectan datos o información en tiempo real.
f) La unión de varios sistemas embebidos forma sistemas más complejos.
g) Presentan un bajo consumo de potencia.
h) Su precio es económico en el mercado.

1.4.1.2 Clasificación de los Sistemas Embebidos

Se puede clasificar a los sistemas embebidos dependiendo si están constituido por un DSP (*Digital Signal Processor*), un DSC (*Digital Signal Controller*), un microprocesador o un

microcontrolador, ya que estos elementos son la parte inteligente del sistema llamada CPU [5]. En la Tabla 1.1 se observa de la clasificación de los tipos de sistemas embebidos.

Tabla 1.1 Tabla de la clasificación de los tipos de sistemas embebidos que existen.

CPU	Descripción	Ejemplo
DSP	Sistema embebido que procesa una señal digital. Se usa por lo general para el procesamiento de audio y video.	Texas Instruments: C2000-C5000-C6000
DSC	Presentan características de un DSP y un microcontrolador. Se usan en detección de fenómenos físicos con sensores, control de motores, entre otros.	Microchip: dsPIC33F
Microprocesador	Circuito integrado de alta complejidad, el cual puede ejecutar cálculos aritméticos o lógicos. Se usa generalmente en computadores.	AMD: Ryzen 5 3600.
Microcontrolador	Circuito integrado digital programable para realizar diversas tareas. Se lo utiliza en juegos, calculadoras, alarmas, pantallas, entre otros.	Arduino: ATmega2560

1.4.1.3 Tipos de memoria de los Sistemas Embebidos

La memoria que incluyen los sistemas embebidos es un recurso que se debe gestionar de forma adecuada y eficiente. A continuación, se detallan los tres tipos de memoria presentes en estos sistemas.

a) Memoria SRAM: memoria donde se almacenan los datos usados en el transcurso de las operaciones. Por lo general se tratan como un banco de registros y memoria volátil, se debe controlar su uso para prevenir que se agote.

b) Memoria EEPROM: memoria no volátil que se usa para que los datos, como variables o programas del microcontrolador, queden guardados cuando el sistema embebido se apaga o se realiza un *reset*. Presenta un número limitado de lecturas y escrituras, la escritura se realiza byte a byte, lo que hace que sea más lenta que la SRAM.

c) Memoria Flash: memoria del programa, se puede ejecutar un programa desde esta memoria sin modificar los datos, si se requiere realizar esta tarea se debe copiar los datos a la memoria SRAM y ahí editarlos. Este tipo de memoria tiene una vida

útil de aproximadamente cien mil ciclos de escritura, lo que hace que tenga una duración de varios años.

1.4.2 Sensores

Un sensor es un dispositivo electrónico que permite captar una señal que pudo originarse por algún fenómeno físico, electromagnético, sonoro, entre otros. El uso principal que se da a estos dispositivos es recolectar información de parámetros físicos y ambientales. La característica más relevante es su bajo consumo de energía, debido a que están equipados con gran autonomía, que hace posible que se implementen en entornos poco accesibles [4].

En el presente trabajo se va a medir el nivel de ruido e iluminación por lo que se van a definir estos sensores, así como su funcionamiento.

1.4.2.1 Sensor de Ruido

Es un módulo electrónico que transforma ondas acústicas en señales eléctricas. El funcionamiento de los detectores de ruido se basa en los cambios de capacitancia generados por la vibración de las ondas sonoras. La vibración del aire hace que el diafragma del micrófono se mueva provocando una variación de capacitancia en el condensador acoplado [6].

1.4.2.2 Sensor de Iluminación

Es un dispositivo electrónico que convierte la energía de la luz en radiaciones electromagnéticas o en fotones; además percibe los cambios de la iluminación en un lugar específico [7].

1.4.3 Conceptos Generales de Ruido e Iluminación

En esta sección se definen los fenómenos que se van a detectar mediante la implementación del prototipo basado en sistemas embebidos y sensores.

1.4.3.1 Ruido

Es un fenómeno físico que se produce por las vibraciones sonoras en el medio ambiente, se caracteriza por ser una señal no deseada, que carece de algún tipo de armonía [3]. Para el sentido auditivo resulta molesto y puede afectar a la salud de las personas si se encuentran expuestas a niveles altos de ruido.

En la Tabla 1.2 se puede observar los diferentes tipos de ruido que existen.

Tabla 1.2 Tipos de ruido.

Ruido	Característica	Ejemplo
Continúo	Se da cuando el nivel de presión sonora es constante en un intervalo de tiempo de observación [8].	Industrias, talleres de herramientas.
Intermitente	Se produce cuando se alcanzan un nivel de ruido ambiental alto durante al menos un segundo, pero el nivel disminuye abruptamente llegando a ser muy bajo [8].	Aserraderos, plantas de fundición, mecánica.
De impacto	Se produce debido a algún impacto o explosión y genera muy altos niveles de ruido por milisegundos.	Bombas, explosiones, accidentes.

1.4.3.2 Decibel y su relación con el ruido

El nivel de ruido tiene como unidad de medida el decibel (dB). El decibel (dB) se define como la décima parte del Bel, razón de potencia o intensidad que satisface la Ecuación 1.1 [8]:

$$\log(R) = \frac{1\ dB}{10}$$

Ecuación 1.1 Fórmula para transformar la razón de energía, potencia o intensidad a dB

Donde R, es la razón de energía, potencia o intensidad.

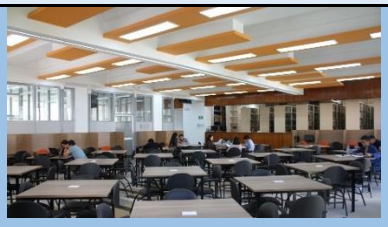
Existen algunas variantes de esta unidad que se usan para realizar curvas relacionadas con el ruido y estas son [8]:

- Curva A (dBA): mide la respuesta del oído, con respecto a un sonido de baja intensidad. Se usa para definir el nivel de contaminación acústica y la amenaza que existe.
- Curva B (dBB): mide la respuesta del oído, con respecto a un sonido de intensidad media.
- Curva C (dBC): mide la respuesta del oído, con respecto a un sonido de intensidad alta.
- Curva D (dBD): para analizar el nivel de ruido que generan los aviones.

1.4.3.3 Niveles de Ruido

Por lo general, el nivel de ruido dependiendo de la fuente sonora varía de 0 hasta 180 dB. Existen cinco tipos de ambientes sonoros, que tienen un nivel de ruido determinado y se detallan en la Tabla 1.3.

Tabla 1.3 Ambientes y niveles de ruido [9].

Tipo de ambiente sonoro	Acción	Nivel de ruido (dB)	Imagen
Ambiente silencioso	Silencio	0	
	Pisada	10	
	Viento en los árboles.	20	
Ambiente poco ruidoso	Conversación baja	30	
	Biblioteca	40	
	Oficina tranquila	50 **Nivel propuesto por la OMS**	
Ambiente ruidoso	Conversación normal	60	
	Tráfico en la ciudad	80	
	Aspiradora	90	
Ambiente molesto	Motocicleta	100	
Ambiente insoportable	Concierto	120	
	Martillo neumático	130	
	Avión despegando	150	
	Explosión	180	

1.4.3.4 Contaminación Acústica

La contaminación acústica es la presencia o generación de niveles de ruido que están por encima del tolerable para las personas y provocan molestias, riesgos y daños a la salud.

Debido al avance de la tecnología, la industria, la movilización, entre otros, las fuentes de contaminación acústica son cada vez mayores. Según un estudio de la OMS el tráfico urbano es el principal responsable del 80% de este tipo de contaminación [10].

1.4.3.5 Consecuencias de niveles de ruido altos en la salud de las personas

La salud siempre debe ser prioridad, por esta razón la Asociación Médica Mundial (AMM), declara que los niveles de ruido altos son una amenaza relevante para la salud pública. Las consecuencias están relacionadas con la audición, el sistema nervioso vegetativo, la psiquis, la comunicación oral, el sueño y el rendimiento. Además, se presume que el ruido puede estar relacionado con las enfermedades en las cuales el estrés está presente, un ejemplo serían las enfermedades cardiovasculares [11].

La pérdida auditiva conocida en términos médicos como hipoacusia, es la lesión que más relevancia tiene al estar expuestos a exposición prolongada de altos niveles de ruido y según su intensidad se clasifica de la siguiente forma [1]:

- Hipoacusia ligera (pérdida entre 20-40 dB).
- Hipoacusia moderada (pérdida entre 41-60 dB).
- Hipoacusia grave (pérdida entre 61-80 dB).
- Hipoacusia profunda (pérdida mayor a 81 dB).

Existen cinco factores que determinan el riesgo de hipoacusia y se presentan en la Tabla 1.4.

Tabla 1.4. Factores que determinan el riesgo de hipoacusia.

#	Factor	Descripción
1	Intensidad	Cuanto más alto sea el nivel de ruido, mayor será el daño que provoque.
2	Tipo de ruido	Depende si es continuo, intermitente, fluctuante o de impacto. El ruido continuo se soporta más que el discontinuo [8].
3	Tiempo de exposición al ruido	Se refiere a las horas por día u horas por semana de exposición.
4	Edad	El sentido de la audición se va perdiendo con la edad, sin importar si hay exposición al ruido o no.
5	Susceptibilidad Individual	Cada persona reacciona de forma diferente dependiendo de sus propias condiciones.

1.4.3.6 Iluminación

En términos generales, se define a la iluminación como la acción de proporcionar luz en un ambiente determinado. A lo largo de la historia se ha buscado resolver el problema de la ausencia de luz, por este motivo se han creado distintos tipos de fuentes de luz. Por otra parte, los sistemas de iluminación deben ser adecuados para el tipo de trabajo o acción que realicen las personas [12].

1.4.3.7 Flujo Luminoso

Se define como la cantidad de energía luminosa que emite una fuente de luz en todas las direcciones. La unidad de medida se conoce como lumen (Lm) y su símbolo está representado por la letra griega (Φ). Los dispositivos más comunes que radian flujo luminoso son los focos o bombillos [12].

1.4.3.8 Intensidad Luminosa

Se define como la relación entre el flujo luminoso (Φ) y el ángulo solido en una dirección determinada. Se representa con el símbolo (I) y su unidad de medida es la candela (cd). La intensidad luminosa satisface la Ecuación 1.2 [12].

$$I = \frac{\Phi}{W}$$

Ecuación 1.2 Cálculo de la intensidad luminosa.

Donde:

I, es la Intensidad luminosa en (Cd).

Φ, es el flujo luminoso dentro del ángulo sólido en (Lm).

W, es el ángulo sólido en estereorradianes (sr).

1.4.3.9 Nivel de Iluminación o Iluminancia

Se define como la relación entre el flujo luminoso presente sobre una parte de la superficie que comprende el punto por el área de ese elemento. Se representa con la letra E, en la Figura 1.1 se observa su unidad de medida. La iluminancia satisface la Ecuación 1.3 [12].

$$E = \frac{\Phi}{S}$$

Ecuación 1.3 Cálculo del nivel de iluminación.

Donde:

E, nivel de iluminación en luxes (Lx).

Φ, flujo luminoso que índice sobre una superficie en lúmenes (Lm).

S, superficie en m^2.

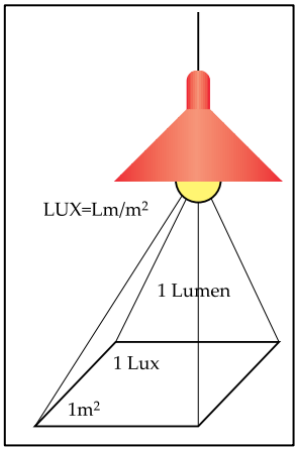


Figura 1.1 Unidad de medida del nivel de iluminación (LUX) [12].

Existen niveles de iluminación que se recomiendan según el lugar que se esté iluminando y estos se pueden observar en la Tabla 1.5.

Tabla 1.5. Clasificación de un ambiente lumínico y nivel de iluminación [13].

Áreas y clases de local	Mínimo (LUX)	Óptimo (LUX)	Máximo (LUX)
Viviendas			
Dormitorios	100	150	200
Áreas de aseo	100	150	200
Áreas de estar	200	300	500
Cocinas	100	150	200
Áreas de estudio	300	500	750
Zonas generales de edificios			
Áreas de circulación y pasillos	50	100	150
Centros docentes			
Aulas y laboratorios	300	400	500
Bibliotecas y salas de estudio	300	500	750
Oficinas			
Oficinas, salas de proceso y conferencias	450	500	750
Oficinas grandes	500	750	1000

1.4.3.10 Contaminación Lumínica

Se define a este tipo de contaminación como la perturbación de los niveles naturales de luz en el ambiente exterior provocado por emisiones de luz artificiales. La causa principal de este fenómeno se debe al incremento de la población y expansión masiva de las áreas urbanas, la industria e implementación tecnológicas de nuevas formas de iluminación [14]. Debido a un sistema de iluminación mal diseñado o con un nivel de iluminación inadecuado se originan los siguientes fenómenos:

a) Reflexión de la luz hacia el cielo: se crea por la radicación de luz artificial sobre un área que presenta propiedades que favorecen a la reflexión de la luz incidente sobre dicho elemento y ésta alcance la atmósfera como luz residual.
b) Dispersión de la luz al cielo: se crea por la interrelación de la luz con las partículas existentes en la atmósfera, emitiendo luz en varias direcciones.
c) Intrusión Lumínica: es la radiación de flujos luminosos que irrumpen dentro de zonas en las que no deben estar, pueden provocar molestias o daños.
d) Deslumbramiento: es una forma de contaminación lumínica que consiste en la radiación de flujos luminosos que obstruyen o limitan la visión.

1.4.3.11 Consecuencias de una deficiente iluminación en la salud de las personas

Una ineficiente iluminación puede provocar varios problemas en la salud de las personas, cada lugar en el que las personas realizan sus tareas diarias, debe tener un adecuado nivel de iluminación [2]. Las consecuencias más relevantes de este problema son [2]:

a) Fatiga ocular: enfermedad que se origina por el uso intenso y prolongado del sentido de la vista.
b) Accidentes laborales: provocados por una iluminación inadecuada en el lugar de trabajo.
c) Trastornos oculares: Dolor e inflamación en los párpados, fatiga visual, pesadez, lagrimeo, enrojecimiento, irritación, visión alterada
d) Dolores de cabeza: se recomienda que un doctor revise el problema para verificar si es la iluminación la que lo provoca.
e) Fatiga: falta de energía o agotamiento, cuando la persona está agotada por estrés o insomnio, la fatiga puede durar todo el día.
f) Efectos anímicos: disminución de la concentración, productividad, atención y desánimo.

2. METODOLOGÍA

En el presente trabajo de integración curricular se establece una metodología experimental, debido a que se va a desarrollar un prototipo para la detección del nivel de ruido e iluminación. Al obtener los datos, estos pueden estar sujetos a fallas o errores. En la Figura 2.1 se detalla cada una de las fases.

En primer lugar, se lleva a cabo la fase de exploración del mercado, en la que se estudian los dispositivos que están en el mercado y con esta información se eligen los más adecuados.

A continuación, se implementa la fase de diseño del prototipo, en la cual se establecen los elementos que se van a utilizar tales como el sistema embebido, sensor de ruido, sensor de iluminación, entre otros. Luego, se va a realizar la fase de implementación del prototipo interconectando los sensores y el sistema embebido; además, con ayuda de la programación en el IDE (Entorno de Desarrollo Integrado) de Arduino se va obtener los datos de los niveles de ruido e iluminación en dB y luxes, respectivamente.

Finalmente, se lleva a cabo la fase de análisis de las mediciones obtenidas, en la cual, con base a los resultados obtenidos en las pruebas de funcionamiento, se calcula el error que existe en el prototipo, contrastando los datos obtenidos con el prototipo con mediciones del nivel de ruido e iluminación obtenidas usando una aplicación móvil o dispositivos electrónicos. Además, se realiza un análisis de los datos mediante distintos tipos de gráficas.

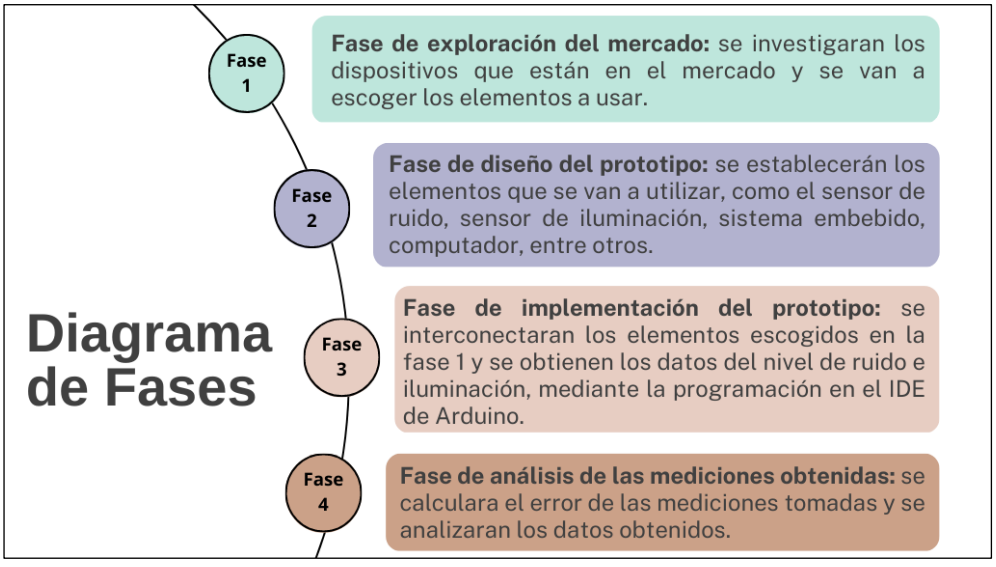


Figura 2.1 Diagrama de fases de la implementación del prototipo de medición del nivel de ruido e iluminación.

2.1 Fase de Exploración del Mercado

En esta fase se va a explorar en el mercado las características más relevantes de los dispositivos que se requieren para la implementación del prototipo de detección del nivel de ruido e iluminación.

2.1.1 Tipos de sistemas embebidos disponibles en el mercado

Hon en día, existe gran variedad de sistemas embebidos en el mercado, los más usados son las placas de Arduino, NodeMCU, entre otros. Estos dispositivos son los que permiten realizar la adquisición de datos a través de los sensores; además son la parte inteligente de todo el sistema IoT (*Internet of things*). En la Tabla 2.1, se presentan las características de cada elemento.

Tabla 2.1. Tipos de sistema embebido en el mercado [15], [16] y [17].

Dispositivo	Arduino Uno	Arduino Mega 2560	ESP 32
Imagen			
Microcontrolador / Microprocesador	ATMega328P	ATmega2560	Dual core Xtensa LX6 de 32 bits
Frecuencia(velocidad de reloj)	16 MHz	16 MHz	160 MHz y 240 MHz
Memoria	- 32 KB Flash - 2KB RAM - 1KB EEPROM	- 256 KB (8KB usados por el bootloader) - SRAM: 8KB - EEPROM: 4KB	- 520 KB de RAM - 448 KB de ROM - SRAM de 16 KB en RTC
Pines	- Pin in/out: 14 pines digitales (6 PWM) - Pines analógicos: 6 - Puerto serial: 1	- Pin in/out: 54 pines (14 PWM) - 16 pines analógicos de entrada - 4 puertos seriales	- 38 pines totales - 16 pines (ADC) - 3 SPI interfaces - 3 UART interfaces - 2 I2C interfaces - 2 pines (DAC) - 2 I2S interfaces - 10 Capacitive sensing GPIOs
Interfaces de comunicación	USB (Tipo B-2.0)	USB (Tipo B-2.0) – SERIE	- USB (Tipo A) - 802.11 b / g / n - 802.11 n (2.4 GHz), - Bluetooth v4.2 BR / EDR y BLE
Consumo Energético	46 mA	40 y 50 mA	50 mA
Dimensiones	68.6×53.4 mm	101.52×53.3 mm	51 x 23 x 8 mm
Precio (incluido IVA)	20 USD	40 USD	15 USD

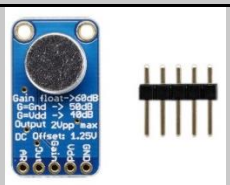
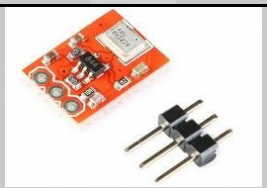
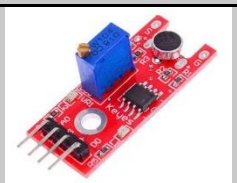
De la Tabla 2.1, se evidencia que el dispositivo que más se ajusta a los requerimientos del prototipo es el ESP32, debido a que presenta las siguientes ventajas.

- Es el dispositivo más económico de los elementos estudiados, presenta mejor procesamiento y memoria que los demás dispositivos.
- Tiene el módulo WiFi integrado, por lo que es idóneo para aplicaciones de IoT.
- La placa ESP32, es la que menor dimensiones presenta, por lo que es adecuada para colocarla en una placa de circuito impreso (PCB).
- Utiliza el mismo software de código abierto de arduino IDE.

2.1.2 Sensores de ruido disponibles en el mercado

El sensor de ruido es un dispositivo que va a convertir las ondas acústicas en señales eléctricas o de voltaje, debido a que son de bajo consumo, en el mercado por lo general resultan ser económicos. A continuación, se presenta en la Tabla 2.2 con las características de tres sensores de ruido.

Tabla 2.2. Sensores de ruido disponibles en el mercado [18], [19] y [20].

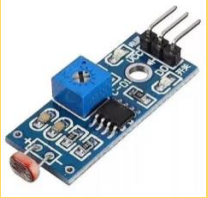
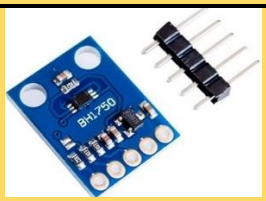
Sensor de ruido	Adafruit AGC Electret Microphone Amplifier MAX9814	ADMP 401 MEMS Micrófono Omnidireccional Breakout módulo Board	KY-038 Microphone sound sensor module
Imagen			
Rango de frecuencia	20 Hz - 20 KHz	100Hz -15kHz	100 Hz– 10 KHz
Alimentación	2,7 – 5,5V.	1,5 a 3,3V	5V
Ajustes de ganancia	40dB, 50dB, 60dB	67 dB	-
Sensibilidad	-42±3dB (0dB=1V/Pa.1KHz)	−42 dBV	– 46 ± 2,0 (0dB = 1V/Pa a 1KHz)
Consumo de Corriente	<3 mA	<250 uA	< 1mA
Salida analógica	1	1	1
Costo (incluido IVA)	8 USD	32 USD	3 USD

Después de observar la Tabla 2.2, se ha decidió trabajar con el sensor MAX9814, debido a que este sensor cuenta con circuito de amplificación de sonido, un micrófono con *low-noise* y tiene control automático de ganancia (AGC), dependiendo de la conexión de sus pines. Todo esto hace posible que los ruidos cercanos o lejanos sean detectados por el dispositivo de forma eficiente. Finalmente, el precio del sensor es económico y su tamaño es factible para colocarlo en una placa PCB.

2.1.3 Sensores de iluminación disponibles en el mercado

Existen varios tipos de sensores de iluminación, el más conocido es el módulo LDR, que se basa en una fotorresistencia, que cambia su valor dependiendo la intensidad de luz que reciba.

Tabla 2.3. Sensores de iluminación disponibles en el mercado [21] y [22].

Sensor de iluminación	**Módulo sensor LDR**	**Módulo Sensor Intensidad Luz i2c BH1750 Gy-302 Pic**
Imagen		
Voltaje de Operación	3.3V - 5V DC	3V – 5V
Conexión de cables	VCC, GND, DO, AO	VCC ,GND, SCL, SDA, ADDR
Tipo de salida	Digital y analógica	Nivel de iluminación (luxes)
Número de pines	3	5
Rango de medición	-	1-65535 lux
Costo (incluido IVA)	3 USD	5 USD

En la Tabla 2.3, se evidencia que el sensor BH1750 presenta mayores prestaciones debido a que mide el nivel de iluminación en luxes, mientras que para conseguir este resultado con el sensor LDR, se debe usar formulas y programación extra; además, el costo de los sensores resulta ser bajo y no hay diferencia representativa.

2.2 Fase de Diseño del Prototipo

En esta fase se define todos los elementos que son necesarios en el diseño del prototipo, y cuál va a ser el proceso que se lleva a cabo para que los datos obtenidos por los sensores lleguen hasta el usuario final.

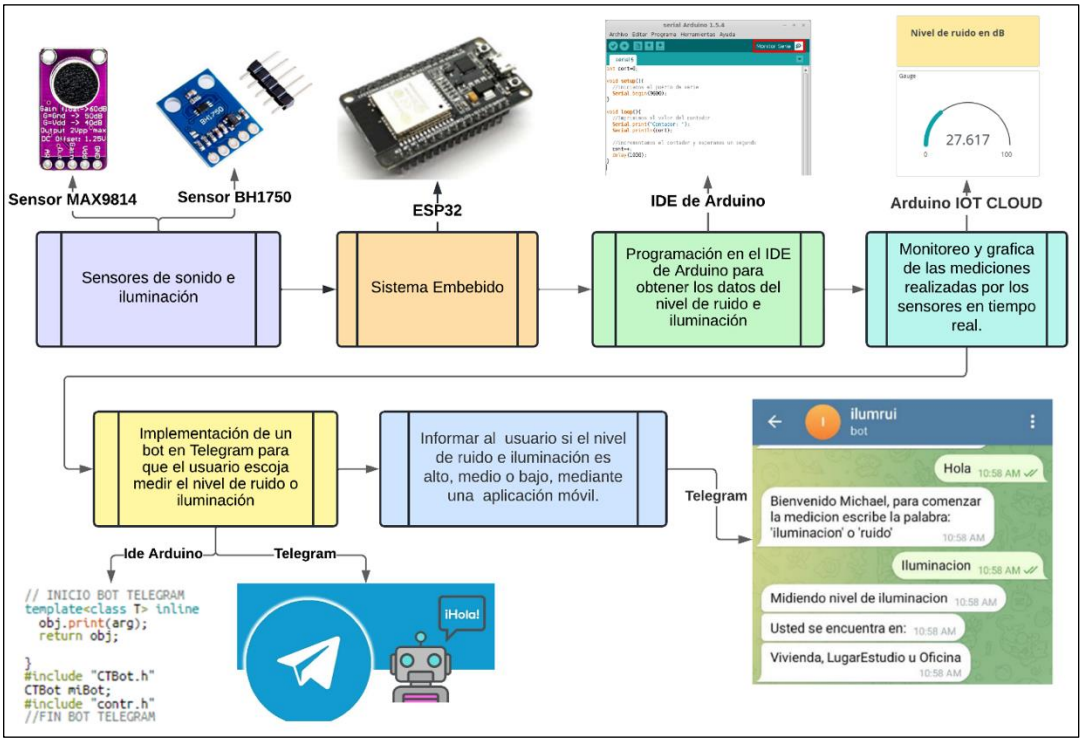


Figura 2.2 Diagrama de bloque del proceso de diseño del prototipo.

El diagrama de bloque de la Figura 2.2 muestra cómo se interconecta cada bloque que forma parte del diseño del prototipo. A continuación, se detalla lo que se realiza en cada etapa.

2.2.1 Sensores de sonido e iluminación

En esta etapa se utiliza el módulo MAX9814 para obtener el nivel de ruido y el sensor BH1750 para medir el nivel de iluminación. Estos elementos se conectan a la placa ESP32, la cual es el sistema embebido que se usa en todo el sistema.

2.2.1.1 Sensor de Sonido MAX9814

Convierte ondas sonoras a una señal eléctrica analógica en valores enteros entre 0-1023, de este modo se puede convertir esta señal a voltaje y nivel de ruido. El sensor tiene distintos valores de precisión y ganancia dependiendo la conexión de sus pines, esto se observa en la Tabla 2.4.

Tabla 2.4 Ajustes de precisión (AR) y ganancia (*Gain*) del sensor MAX9814.

Pin	Sin conexión	VCC	GND
Gain	60 dB	50 dB	40 dB
AR	1:4000	1:2000	1:500

A continuación, se observa en la Tabla 2.5 y Figura 2.3 como se debe conectar el ESP32 con el sensor.

Tabla 2.5 Conexión entre el ESP32 y el sensor MAX9814.

ESP32	MAX9814
V5	VCC
GND	GND
Out (analógico)	G34
Gain	-
AR	-

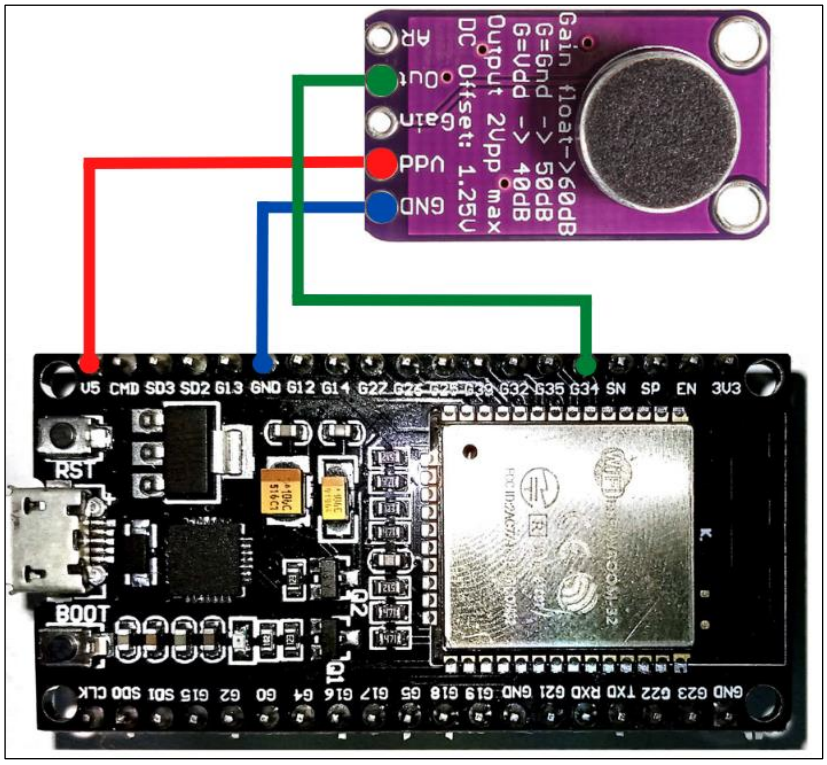


Figura 2.3 Conexión entre el ESP32 y el sensor MAX9814

2.2.1.2 Sensor de nivel de iluminación BH1750

Es un sensor con salida digital, que detecta el nivel de iluminación o la iluminancia en luxes [$lumem/m^2$], presenta un rango de medición configurable de 1 a 65535 lx. El sensor contiene un regulador interno de 3.3V, que permite que se alimente de un puerto de 5V.

Es posible realizar conexiones con la mayoría de microcontroladores, debido a que tiene una interfaz de comunicación I2C (*Inter-Integrated Circuit*). Además, cuenta con un pin para establecer la dirección, que se lo puede conectar a tierra (GND) o por lo general se lo deja

libre. En la Figura 2.4 y la Tabla 2.6 se puede observar los pines a los que se conectan los elementos.

Tabla 2.6 Conexión entre el ESP32 y el sensor KY-038.

ESP32	BH1750
3V3	VCC
GND	GND
SCL	G22
SDA	G21
ADDR	-

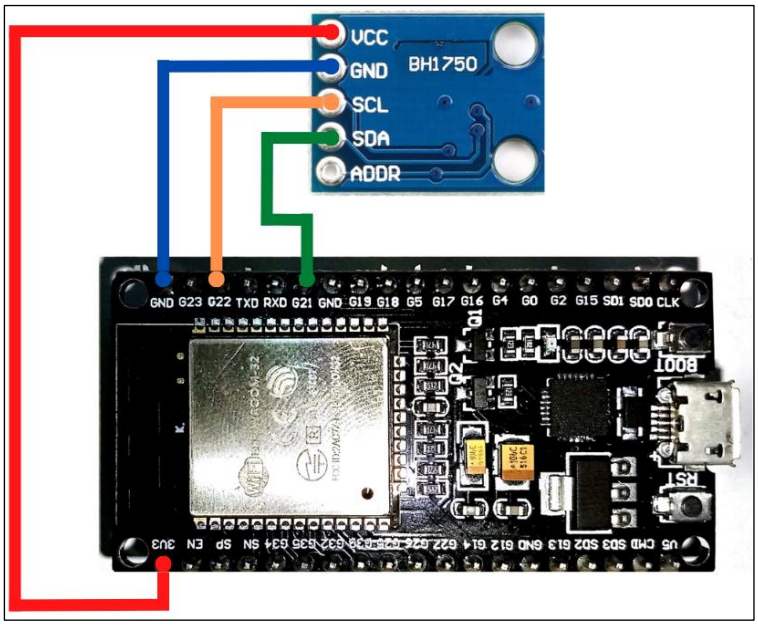


Figura 2.4 Conexión del Sensor BH1750 con la placa ESP32.

2.2.2 Sistema Embebido ESP32

Es una placa electrónica que contiene varios elementos que conforman todo el sistema embebido. Se caracteriza por tener un microprocesador de 32 bits, una memoria RAM de 520 KB, 38 pines y su mayor ventaja es que tiene conexión WiFi, por lo que se puede usar en sistemas IoT. Presenta mayores prestaciones que algunas placas de Arduino; además tiene un precio económico y su disponibilidad en el mercado es alta. Para alimentar la placa se realiza una conexión mediante un cable microUSB, como se observa en la Figura 2.5.

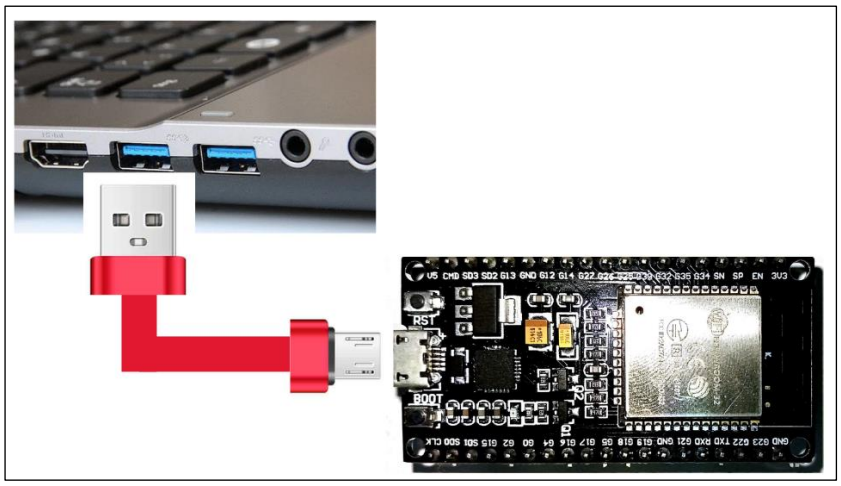


Figura 2.5 Conexión entre la placa ESP32 y la PC.

2.2.3 Programación en el IDE de Arduino para obtener los datos del nivel de ruido e iluminación

La placa ESP32 se puede programar con el IDE de Arduino instalando las librerías y drivers adecuados. En esta sección se va a mostrar y explicar cómo se realizó la programación para obtener el nivel de ruido e iluminación.

2.2.3.1 Programación en el IDE de arduino para obtener los datos del nivel de iluminación

Para obtener los datos del nivel de iluminación en luxes se instala la librería del sensor BH1750 en el gestor de bibliotecas.

Figura 2.6 Librería del sensor BH1750 [23].

A continuación, se presenta el código 2.1, el cual se toman las medidas del nivel de iluminación para realizar pruebas de funcionamiento y verificar que las conexiones de los pines estén correctas.

Código 2.1 Medidor del nivel de iluminación.

```
#include <BH1750.h> //libreria del sensor BH1750
#include <Wire.h>
BH1750 lightMeter; //se crea el objeto lightMeter

void setup() {
  Serial.begin(9600); //velocidad de transmision
  Wire.begin(); // se inicializa la libreria Wire y conecta Arduino al bus
  lightMeter.begin(); //se inicializa el sensor

  Serial.println(F("PRUEBA SENSOR DE NIVEL DE ILUMINACION - BH1750"));
//imprime en el monitor serie
}

void loop() {
  float ilum = lightMeter.readLightLevel(); //se inicializa la variable
ilum (aqui se guarda la medicion del sensor)
  Serial.print("Nivel de iluminacion:");
  Serial.print(ilum);
  Serial.println(" lx");
  delay(1000); //pausa de 1 segundo
}
```

En la Figura 2.7 se observan los datos del nivel de iluminación que se adquieren, y en la Figura 2.8 el nivel de iluminación máximo que se obtuvo con una lámpara.

```
X/3 PRUEBA SENSOR DE NIVEL DE ILUMINACION - BH1750
Nivel de iluminacion:2488.33 lx
Nivel de iluminacion:2503.33 lx
Nivel de iluminacion:2503.33 lx
Nivel de iluminacion:2500.00 lx
Nivel de iluminacion:2500.00 lx
Nivel de iluminacion:2580.83 lx
Nivel de iluminacion:27907.50 lx
Nivel de iluminacion:27403.33 lx
Nivel de iluminacion:48555.83 lx
Nivel de iluminacion:49790.00 lx
Nivel de iluminacion:47625.00 lx
Nivel de iluminacion:54612.50 lx
```

Figura 2.7 Datos del nivel de iluminación vistos en el monitor serie de Arduino.

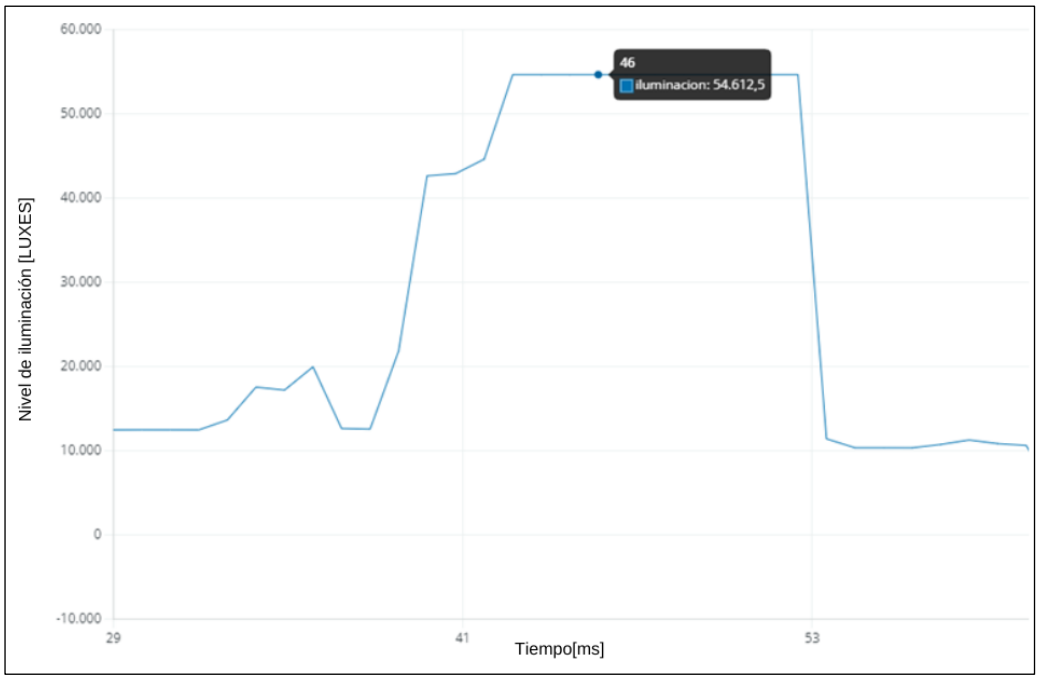


Figura 2.8 Nivel de iluminación visto en el Plotter Serie del IDE de Arduino.

2.2.3.2 Programación en el IDE de Arduino para obtener los datos del nivel de ruido

El sensor de MAX9814 no requiere la instalación de ninguna librería del IDE de Arduino, solamente necesita leer el valor del pin Out, con la función *analogRead*. En el código 2.2, se muestra la programación para obtener los datos del sensor y verificar que la conexión de todos los pines es correcta y en la Figura 2.9 se observa los resultados al ejecutar el código

Código 2.2 Medidor del voltaje ADC del sensor MAX9814.

```
int sound_analog = 34; //se declara el pin 34 del ESP32, el cual esta
conectado el pin Out del sensor MAX9814

void setup(){
  Serial.begin(9600); //velocidad de transmision
}

void loop(){
  int val_analog = analogRead(sound_analog); //se declara la variable
val_analog que guarda el dato del sensor
  Serial.println("El valor medido por el sensor es:");
  Serial.println(val_analog); //se imprime en el monitor serie
  delay(1000); //pausa de 1 segundo
}
```

```
El valor medido por el sensor es:
1345
El valor medido por el sensor es:
2799
El valor medido por el sensor es:
2800
El valor medido por el sensor es:
2799
```

Figura 2.9 Datos del sensor vistos en el monitor serie de arduino.

En la Figura 2.10 se observa que el valor máximo que se obtiene con el sensor es 2800.

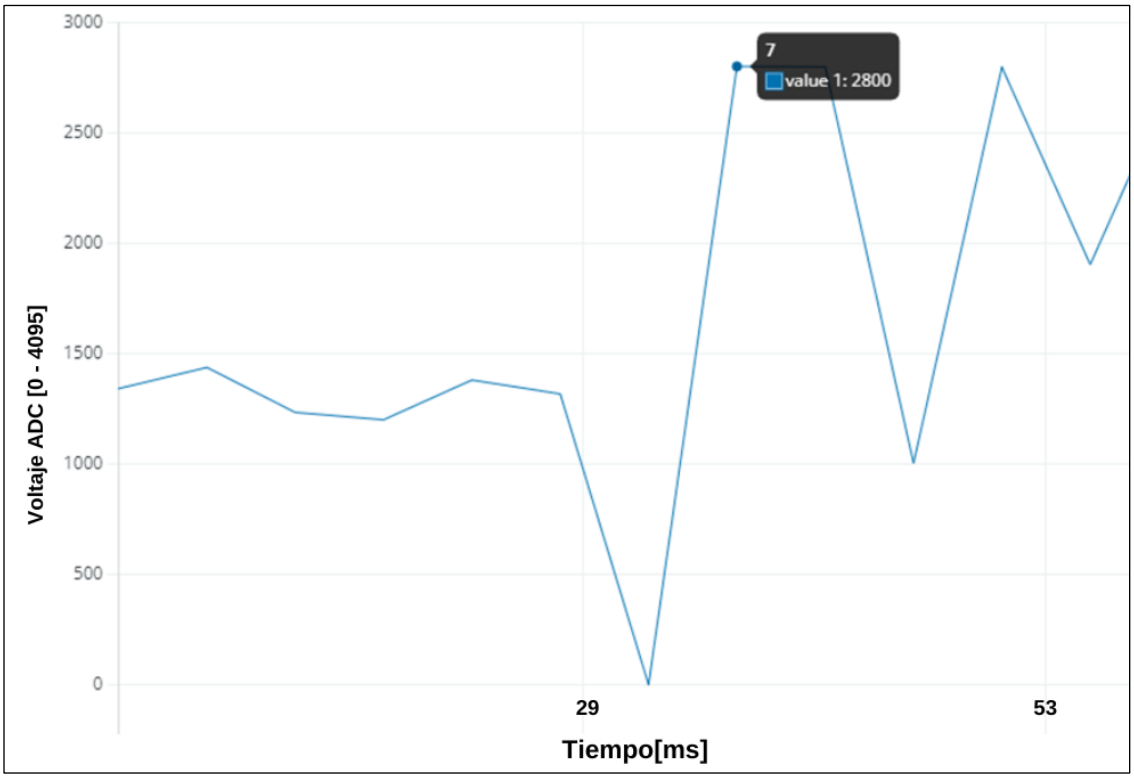


Figura 2.10 Valor máximo del sensor visto en el Plotter Serie del IDE de Arduino.

Al realizar pruebas con el sensor MAX9814, se observó que el pin *Out*, envía como salida valores enteros de 0 a 2800, este valor se debe transformar a voltaje con la Ecuación 2.1.

$$V_s = \frac{V_{ADC} * Out}{Out_{max}} = \frac{2.256[V] * Out}{2800}$$

Ecuación 2.1 Cálculo el voltaje de la señal de salida [24].

Donde:

V_s, es el voltaje de la señal de salida.

V_{ADC}, es el voltaje ADC máximo, en este caso si $Out_{max} = 2800$, entonces $V_{ADC} = 2.256$ [V]. Ya que la placa ESP32, asigna 3.3 [V] al valor 4095.

Out, son los valores enteros que envía el pin *Out* y varia de 0 a 2800.

Ahora se puede obtener el voltaje de la señal de salida en dB con la Ecuación 2.2.

$$V_{dB} = 20 \log_{10}\left(\frac{Vs}{V_0}\right)$$

Ecuación 2.2 Cálculo del voltaje de la señal de salida en dB.

Donde:

V_s, es el voltaje de la señal de salida.

V_0, es el voltaje de referencia igual a 1 voltio.

Finalmente, para hallar el valor del nivel de ruido en dB, se utiliza la Ecuación 2.3.

$$r_{dB} = V_{dB} + 94 + S_{MAX9814} - G_{MAX9814}$$

Ecuación 2.3 Cálculo del nivel de ruido en dB.

Donde:

V_{dB}, voltaje de la señal de salida en decibelios.

94, es $94\ dB_{SPL}$ equivalente a 1 Pascal (Pa). SPL es (*Sound Pressure Level)*, este término se usa para referirse al nivel de ruido en inglés.

$S_{MAX9814}$, sensibilidad del sensor MAX9814.

$G_{MAX9814}$, ganancia del sensor MAX9814.

Cuando se detecta sonido lo primero que se debe analizar es que la señal obtenida varíe rápidamente con el tiempo y además presenta ruido. Por este motivo, el valor exacto de una medición no es correcto, y se debe realizar las mediciones en un determinado intervalo de tiempo.

Para realizar este proceso se establecen ventanas temporales, es decir, un intervalo de tiempo en el cual se realiza la medición y se calcula el valor máximo y mínimo medido dentro de la ventana. Finalmente, al restar estos valores se obtiene la amplitud pico-pico de la señal. El código 2.3 permite medir el nivel de ruido y se basó en la referencia [24].

Código 2.3 Medidor del nivel de ruido en dB

```
#include <math.h>
int sensorPIN = 34;       //es el pin34 del ESP32
const int S_MAX = -42;    // Sensibilidad del microfono dB
const int G_MAX = 60;     // ganancia del sensor dB
const int Ventana = 50;   //toma MUESTRAS por 50 ms -> 20Hz
unsigned int Out_ADC=0;   // se declara la variable del voltaje ADC
void setup(){
  Serial.begin(9600);
}
void loop() {
   unsigned long startMillis= millis();  // Inicio de la ventana
   unsigned int ADCpp = 0;          // valor del voltaje pico-pico ADC
   unsigned int signalMax = 0;
   unsigned int signalMin = 2800;  // 12 bit ADC = 2^12=4095 pero el sensor
solo da hasta 2800
   // se recoge los datos durante la ventana de tiempo
   while (millis() - startMillis < Ventana){
      Out_ADC = analogRead(sensorPIN);
      if (Out_ADC <2800){ // ver si tiene un nuevo maximo
         if (Out_ADC > signalMax){
            signalMax = Out_ADC;  // guarda solo los niveles máximos
         }
         else if (Out_ADC < signalMin){
            signalMin = Out_ADC;  // guardar sólo los niveles minimos
         }
      }
   }
    ADCpp= signalMax - signalMin; // max - min = amplitud pico-pico ADC
    double Vs,V_dB,r_dB;
    Vs = ((ADCpp*2.256)/2800); // 2.256 es el voltaje ADC que envia placa
ESP32 2800->2.256
    V_dB = 20*log10(Vs/0.007943); //  0.007943-> -42dB sensibilidad del
micrófono
    //Calculo del nivel de ruido en dB:
    r_dB = V_dB + 94 + S_MAX - G_MAX;  //94 dBspl =1Pa
    // Mostrar los resultados en el monitor serie
    Serial.print("ADC: " + String(ADCpp) );
    Serial.print("\t");
    Serial.print("voltaje: " + String(Vs) );
    Serial.print("\t");                        // prints a tab
    Serial.print("VdB: " + String(V_dB) );
    Serial.print("\t");
    Serial.println("NivelRuido: " + String(r_dB) );
}
```

```
ADC: 326        voltaje: 0.26   VdB: 30.39      NivelRuido(dB): 22.39
ADC: 330        voltaje: 0.27   VdB: 30.49      NivelRuido(dB): 22.49
ADC: 424        voltaje: 0.34   VdB: 32.67      NivelRuido(dB): 24.67
ADC: 385        voltaje: 0.31   VdB: 31.83      NivelRuido(dB): 23.83
ADC: 411        voltaje: 0.33   VdB: 32.40      NivelRuido(dB): 24.40
ADC: 375        voltaje: 0.30   VdB: 31.60      NivelRuido(dB): 23.60
ADC: 344        voltaje: 0.28   VdB: 30.86      NivelRuido(dB): 22.86
ADC: 387        voltaje: 0.31   VdB: 31.88      NivelRuido(dB): 23.88
ADC: 475        voltaje: 0.38   VdB: 33.66      NivelRuido(dB): 25.66
ADC: 521        voltaje: 0.42   VdB: 34.46      NivelRuido(dB): 26.46
ADC: 563        voltaje: 0.45   VdB: 35.13      NivelRuido(dB): 27.13
ADC: 590        voltaje: 0.48   VdB: 35.54      NivelRuido(dB): 27.54
```

Figura 2.11 Voltaje ADC, voltaje de la señal de salida, voltaje de la señal de salida en dB y nivel de ruido en dB visto en el monitor serie del IDE de arduino.

2.2.4 Monitoreo de las mediciones realizadas por los sensores en tiempo real

En la actualidad existen varias herramientas que ayudan a monitorear y graficar los datos de los sensores, de esta forma se puede acceder a esta información desde una laptop o un celular. En esta etapa se utiliza Arduino IoT Cloud, la cual es una plataforma en línea que ayuda a la elaboración, implementación y monitoreo de proyectos IoT.

2.2.4.1 Relación entre Sistemas Embebidos e Internet de las cosas (IoT)

El Internet de las Cosas interconecta un elemento, prototipo, electrodoméstico, sensor, entre otros, a Internet con el objetivo de brindarle al aparato inteligencia y de esta manera se pueda relacionar con el entorno que lo rodea, dependiendo de la programación y el objetivo que cumpla el elemento se tendrá diferentes tipos de sistemas IoT.

Los sistemas embebidos dan soporte a los sistemas IoT, ya que es la parte inteligente y la que realiza la adquisición de datos a través de sensores. En la Tabla 2.7 se observa como está constituida la arquitectura IoT.

2.2.4.2 Uso de Arduino IoT Cloud para monitorear datos de sensores

Arduino IoT Cloud ayuda a los usuarios del IDE de Arduino, a interconectar dispositivos o sensores a Internet. De esta manera se puede programar y conectar distintas placas como el ESP32, Arduino Mega, entre otros a esta plataforma.

La versión gratuita permite crear solamente dos elementos o cosas, pero se puede crear todas las variables, dispositivos y *dashboards* que se requieran para poder monitorear y graficar los datos medidos por los sensores.

Tabla 2.7 Capas de la arquitectura IoT [25].

Capa	Definición	Ejemplos
Detección	Está constituida por el conjunto de dispositivos que realizan determinadas tareas, tales como mediciones de magnitudes físicas.	Sensor de luz, humedad, ruido, electrodomésticos.
Intercambio de Datos	Establece la conexión entre los aparatos o sensores a internet mediante tecnologías inalámbricas.	WiFi, LoRa,Zigbee, Bluetooth.
Análisis y Procesamiento	Recopila y analiza los datos enviados por los aparatos o sensores con el objetivo de ser procesados o monitoreados en tiempo real.	Arduino IoT Cloud, Node RED, ThingsBoard.
Interfaz de usuario o Aplicación	Realiza de forma remota la administración, monitoreo, gestión y visualización de los dispositivos.	IoT Cloude Remote, Node-RED Client Editor.

A continuación, se presentan los pasos para la creación de una nueva cosa o elemento.

- **Paso 1:** Crear una cuenta e iniciar sesión en la plataforma de IoT Cloud.

En la Figura 2.12 se presenta la página de inicio para crear una cuenta e inicial sesión.

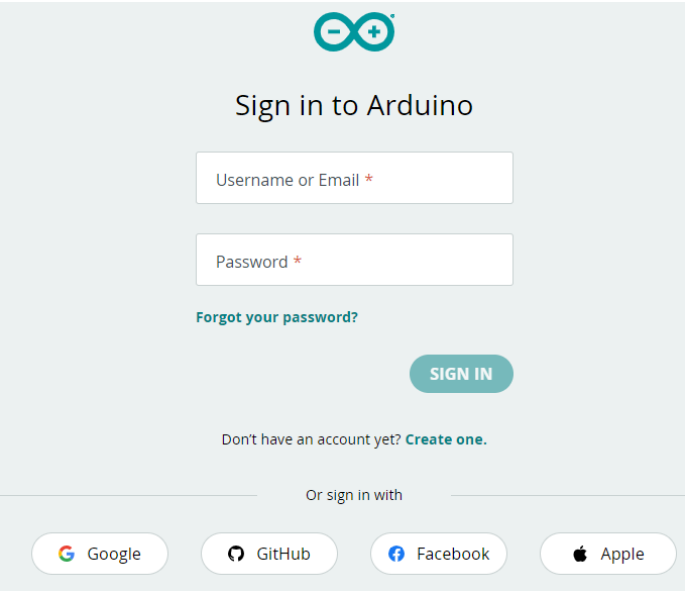


Figura 2.12 Página de inicio de sesión de la plataforma IoT Cloud.

- **Paso 2:** Crear un nuevo dispositivo

En la pestaña *Devices,* en *ADD* se crea un nuevo dispositivo. Existen varias placas que se puede añadir, en este trabajo se usa la placa ESP32 Dev Module, y se le debe asignar un nombre y guardar el *device ID* y *Secret Key* del dispositivo, Se puede descargar esta información (Figura 2.13).

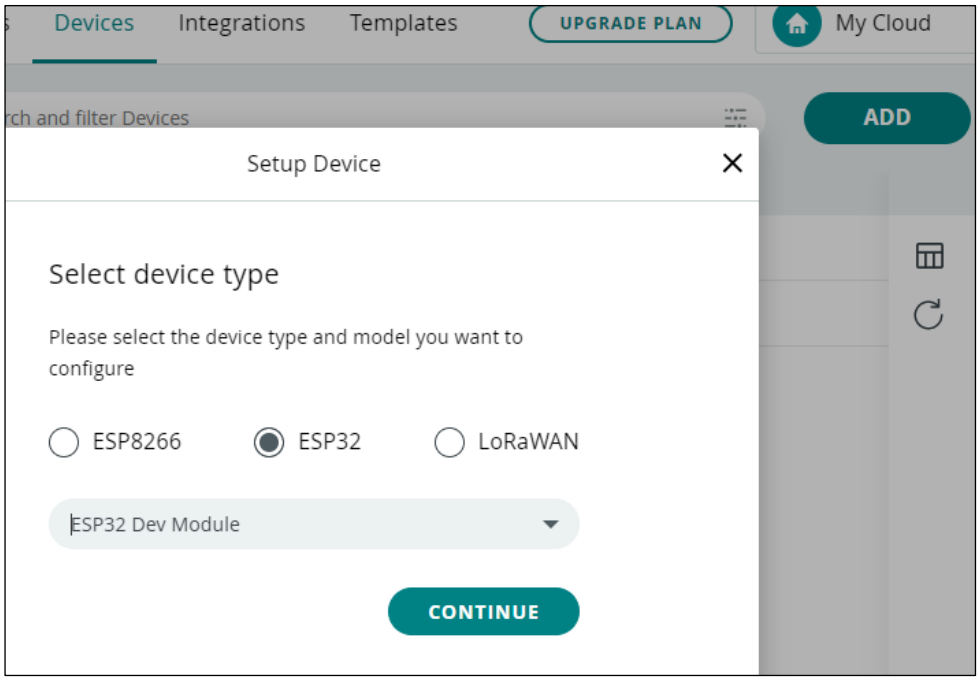


Figura 2.13 Agregar un nuevo dispositivo en la plataforma IoT Cloud.

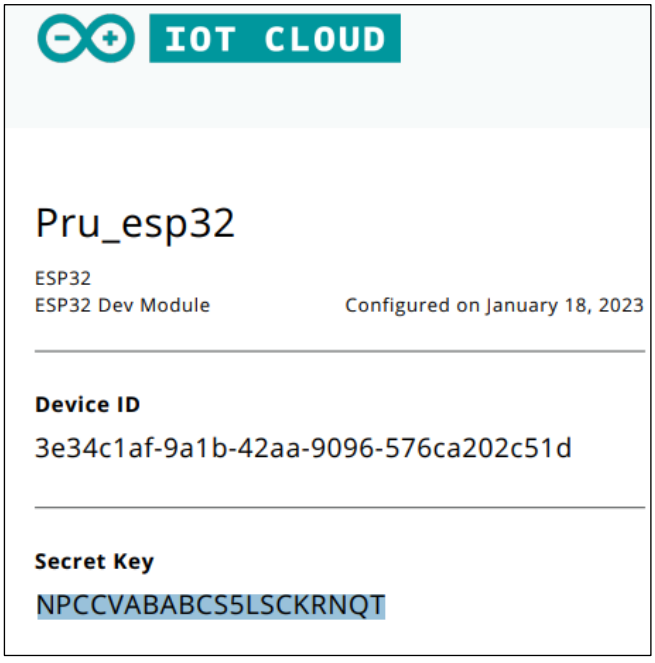


Figura 2.14 Guardado del ID y la clave secreta del dispositivo como un archivo PDF.

- **Paso 3:** Crear un elemento o cosa

En *ADD* de la pestaña *Things,* se crea un elemento o cosa nueva. En esta sección se puede declarar variables, las cuales luego se pueden graficar en el *Dashboard.*

En la opción *Associated Device* se debe seleccionar el dispositivo que se creó en el paso 2 y en *Network* se debe colocar la información de la red WiFi a la que se va a conectar y se coloca la *Secret Key.* En la Figura 2.14 y Figura 2.15 se observa estas opciones en la plataforma IoT Cloud.

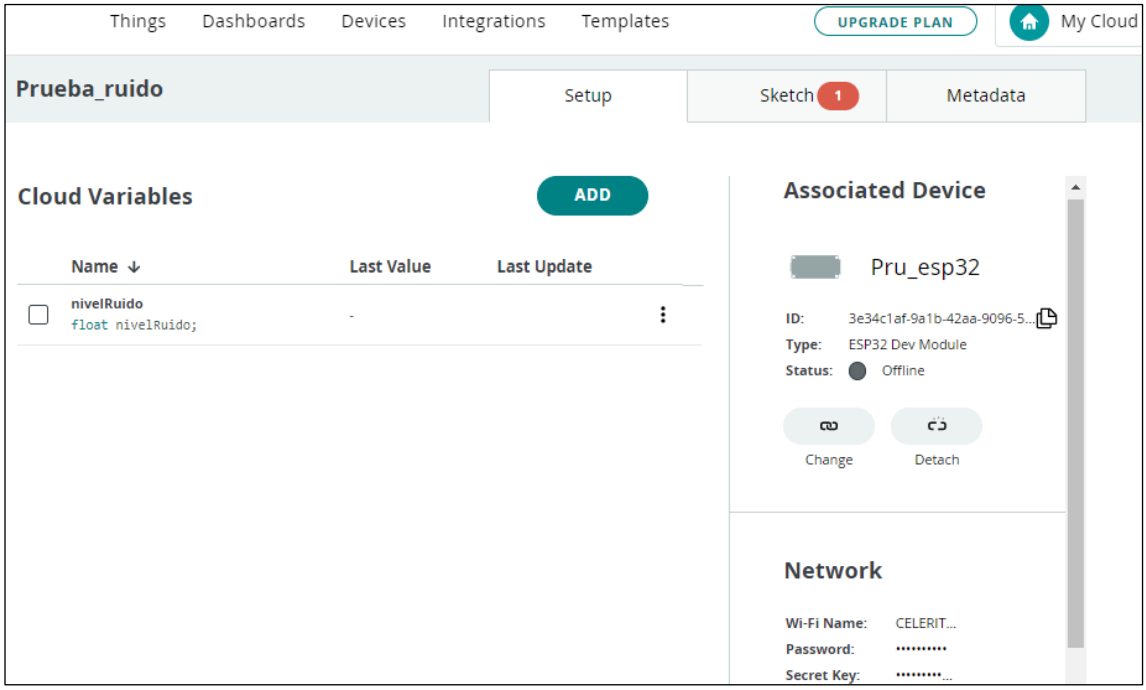


Figura 2.15 Pestaña *Things* de la plataforma IoT Cloud.

- **Paso 4:** Instalar complementos para usar Full Editor en la plataforma IoT Cloud

En la pestaña *Sketch,* se selecciona *Open full editor,* la primera vez que se abra, se debe instalar un complemento que indica la plataforma.

Luego de la instalación se puede incluir las librerías que requieran los sensores, definir los puertos, programar el código, seleccionar la placa y el puerto de comunicaciones para finalmente subir el programa a la placa ESP32 y conectarse por WiFi a la plataforma IoT Cloud. En la Figura 2.16 se muestra la interfaz gráfica del Sketch en la plataforma IoT Cloud.

Figura 2.16 Editor de Sketch de la plataforma IoT Cloud.

2.2.5 Implementación de un *bot* en Telegram para que el usuario escoja medir el nivel de ruido o iluminación

Telegram es una aplicación de mensajeria que esta orientada a brindar velocidad y seguridad a sus usuarios. Cuenta con la opción para crear un bot y establecer comunicación entre el bot, IoT Cloud y el usuario de telegram. En primer lugar se debe buscar *BotFather,* y enviar el mensaje */newbot* como se muestra en la Figura 2.17.

Figura 2.17 Creación de un bot en Telegram.

Luego de colocar el nombre al *bot*, se recibe un mensaje con el *token*, esto es importante, ya que se requiere esta información para que cualquier usuario pueda enviar mensajes al *bot*.

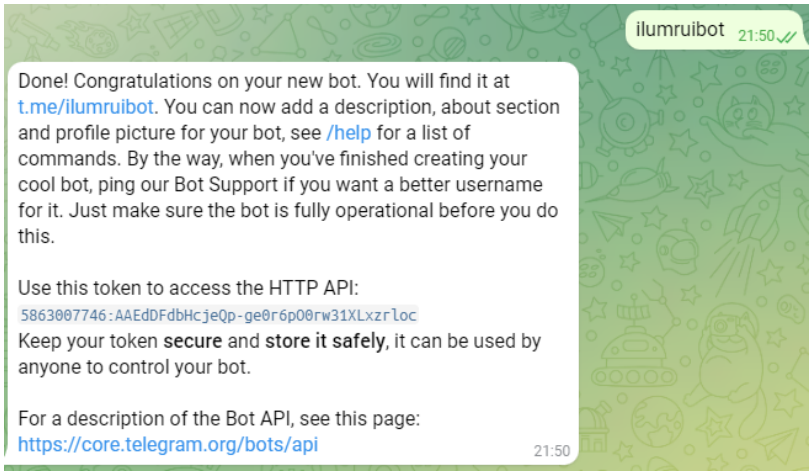


Figura 2.18 Información del token en el *bot* en Telegram.

El código para poder realizar las mediciones del nivel de ruido e iluminación se encuentra en el Anexo I.

Figura 2.19 Medición del nivel de iluminación con Telegram.

La última etapa del diagrama de bloque (Informar al usuario si el nivel de ruido e iluminación es alto, medio o bajo, mediante una aplicación móvil) se va a detallar en el capítulo 2.3 Fase de implementación del prototipo.

2.3 Fase de implementación del Prototipo

En esta sección se va a detallar el diseño de la placa PCB (*Printed Circuit Board*), en el cual se integra la placa ESP32, el sensor MAX9814 y el sensor BH1750. Para realizar este proceso se va a utilizar la plataforma en line EasyEDA, que permite diseñar y fabricar circuitos [26]. Además, se va a programar y mostrar como el usuario desde Telegram puede realizar las mediciones de ruido o iluminación, recibir los mensajes si se encuentra en un ambiente adecuado y monitorear los datos desde el *Dashboard* del IoT Cloud.

2.3.1 Diseño y elaboración de la Placa PCB en EasyEDA

EasyEDA es un software en línea gratuito, que no necesita instalación, y esta implementada en la nube, su principal función es el diseño, simulación y construcción del PCB. Para poder utilizar esta herramienta se debe crear una cuenta o también permite registrarse con una cuenta de *Gmail*. Una de sus principales características es que se puede crear y compartir elementos entre los usuarios de la plataforma, por lo que se puede acceder casi a cualquier dispositivo electrónico que se requiera. A continuación, se detallan los pasos que hay que seguir para el diseño del circuito.

- **Paso 1:** Crear un nuevo proyecto

Cuando se genera un nuevo proyecto, se crea automáticamente un *Sheet,* en donde se puede buscar los elementos en *Commonly Library* o *Library*. Luego se debe colocar los dispositivos a usarse y realizar las conexiones como se mostró en la sección 2.2.1. En la Figura 2.20 se muestra el circuito con los elementos.

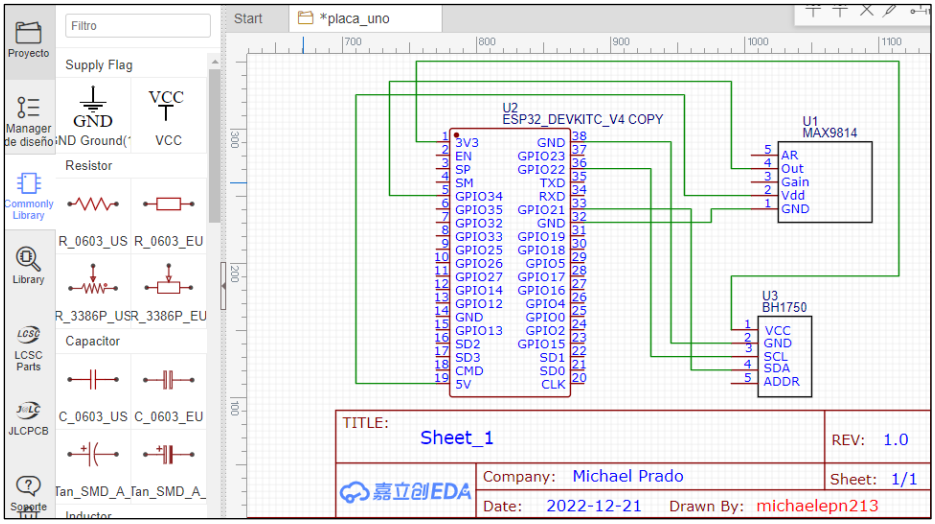


Figura 2.20 Diseño del circuito en la plataforma EasyEDA.

- **Paso 2:** Generar esquemático PCB

Cuando todas las conexiones estén realizadas, se ingresa “Diseño” y luego en “Convertir esquemático a PCB”. Se crea automáticamente la pestaña para realizar el circuito PCB y aparece la ventana de la Figura 2.21, en la cual se escoge las dimensiones de la placa.

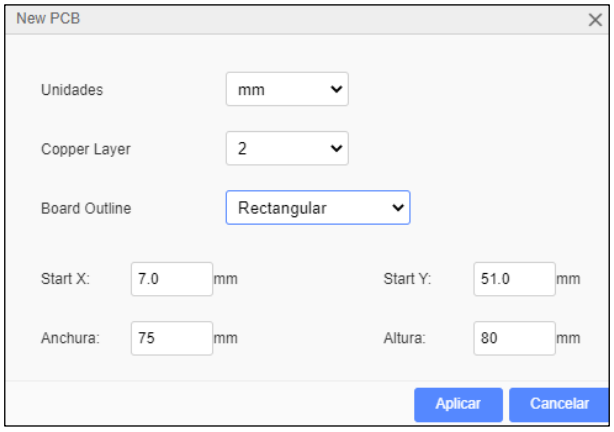


Figura 2.21 Ventana para colocar las dimensiones de la placa PCB en EasyEDA.

- **Paso 3:** Establecer las dimensiones de los caminos del circuito PCB

Se colocan los elementos en posiciones adecuadas para evitar que los caminos se topen. Luego en la pestaña “Ruteo” y en “Auto ruteo”, se abre una nueva ventana y se da clic en “Diseñar regla”, aquí se puede configurar las dimensiones que van a tener los caminos. Esto se muestra en la Figura 2.22.

Figura 2.22 Diseño de las reglas de los caminos de la placa PCB en EasyEDA.

- **Paso 4:** Establecer los caminos del circuito en capa inferior

Luego de configurar las dimensiones se regresa a la ventana “Auto Router Config” y se selecciona “Capa inferior”, finalmente damos clic en ejecutar en la ventana que se observa en la Figura 2.23.

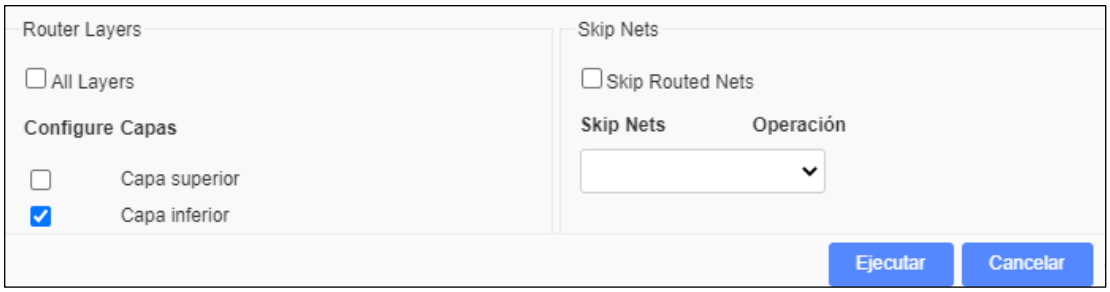


Figura 2.23 Auto Router Config en EasyEDA.

- **Paso 5:** Diseño final del circuito PCB del prototipo

Se obtiene el circuito PCB, se puede agregar texto, imágenes o cambiar las dimensiones de los caminos o puertos para evitar problemas al momento de quemar la placa. Cuando todo esté listo, se da clic en “Archivo – Exportar – PDF” . En la Figura 2.24 se observa el archivo PDF que ya se puede usar para realizar la placa PCB.

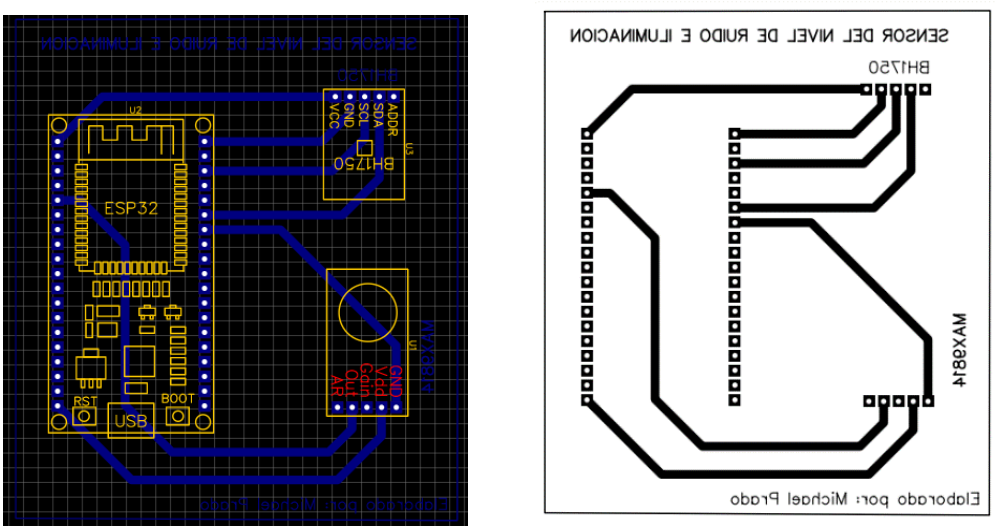


Figura 2.24 Circuito PCB terminado en archivo PDF.

2.3.2 Fabricación del circuito PCB

Para elaborar la placa se utiliza el método del planchado, el cual consiste en imprimir el circuito en formato PDF en una hoja de papel transfer, luego se coloca la impresión sobre el FR4 (Fibra de vidrio para circuitos impresos PCB). Se aplica calor por aproximadamente 30 segundos y se retira el papel transfer. Luego se coloca la placa en cloruro férrico para eliminar la parte de cobre. En la Figura 2.25 se presenta el resultado del proceso.

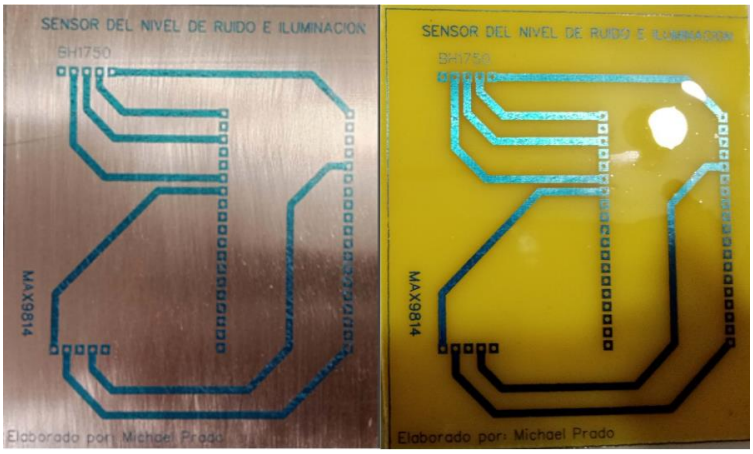


Figura 2.25 Proceso de quemado del circuito PCB.

Finalmente, se debe limpiar el circuito, se coloca unos segundos la placa en un líquido de estaño-plata para fortalecer los caminos, se realiza la perforación de los pines y se suelda los *headers* hembra a los puertos de la placa PCB. En la Figura 2.26 se observa la placa terminada.

Figura 2.26 Capa superior e inferior de placa PCB terminada.

2.3.3 Programación en IoT Cloud para realizar las mediciones

Se realizó la programación en IoT Cloud para que un usuario de Telegram envíe un mensaje al bot “ilumrui”, este *bot* responderá de forma automática con las opciones para medir el nivel de ruido o iluminación.

Finalmente, se informa al usuario si el nivel de ruido e iluminación es adecuado y se le indica qué valor tiene en el entorno que lo rodea, también se puede monitorear y visualizar los datos obtenidos en el *Dashboard de IoT Remote*. El código al ser extenso se lo puede observar en el Anexo I. En la Figura 2.27 se muestra los resultados del programa.

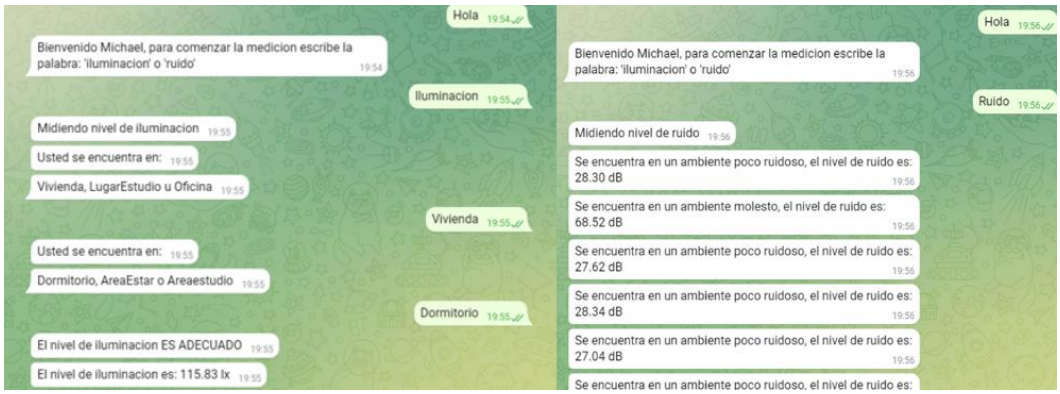


Figura 2.27 Medición del nivel de iluminación y ruido desde Telegram.

En la Figura 2.28 se muestra el monitoreo en tiempo real del nivel de ruido e iluminación, en la plataforma IoT Remote.

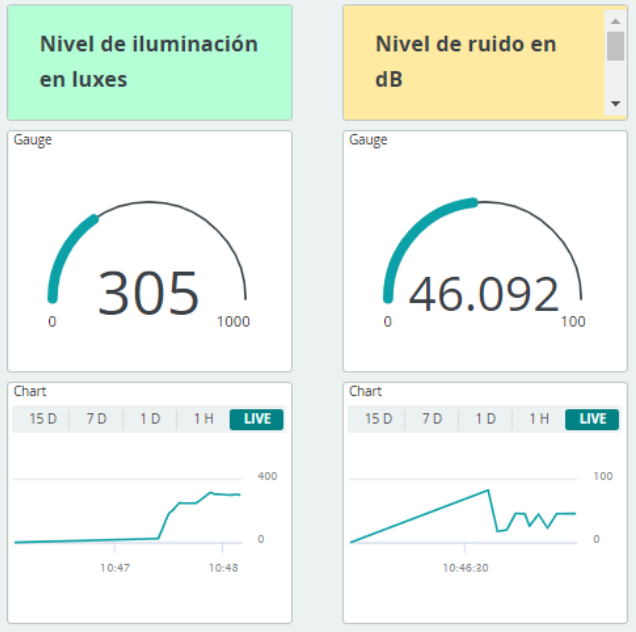


Figura 2.28 Monitoreo del nivel de iluminación y ruido en IoT Remote.

2.3.4 Diseño y fabricación de la caja protectora del prototipo

Para el diseño de la caja protectora se utilizó un software en línea “jeromeleary.com”, que permite generar las dimensiones automáticamente; además, guarda el archivo en formato vectorial del tipo DXF (*Drawing Exchange Format*). Este formato se usa para realizar diseños en Autocad u otro software que permita realizar diseño gráfico computacional. En la Figura 2.29 se muestra la interfaz gráfica de “jeromeleary.com”.

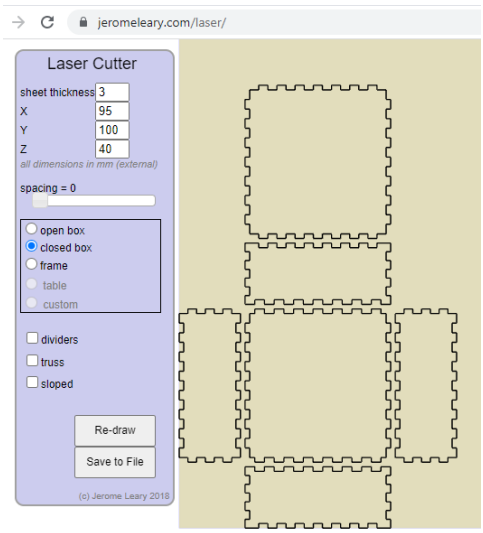


Figura 2.29 Diseño de la caja protectora en el software en línea “jeromeleary.com”.

En la Figura 2.30 se utiliza el software CorelDRAW para abrir el archivo DXF, este programa se usa para el diseño gráfico computacional, luego se realiza los agujeros por los cuales van a salir los sensores MAX9814 y BH1750. Finalmente, se exporta el archivo en PDF, se realiza el corte laser en acrílico de 3mm y se ensamblan las piezas como se muestra en la Figura 2.31.

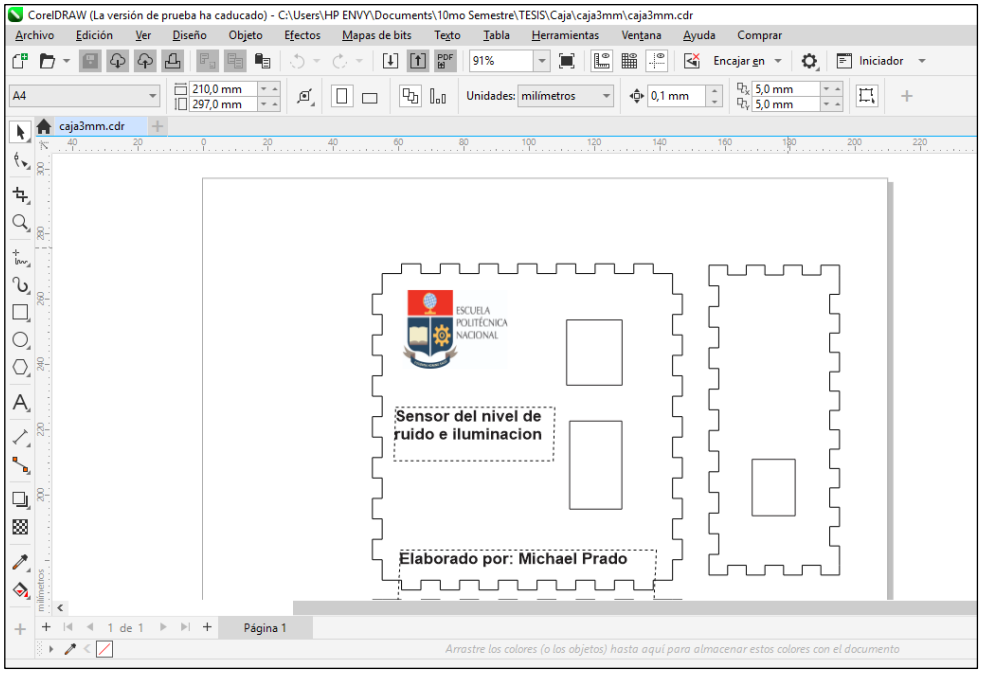


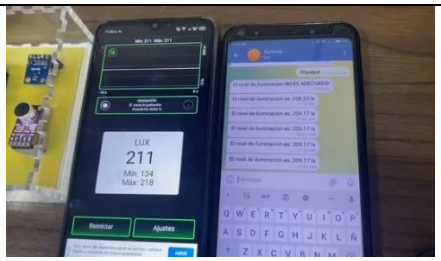
Figura 2.30 Diseño de la caja protectora en el software “CorelDRAW”.

Figura 2.31 Caja protectora ensamblada.

2.4 Fase de análisis de las mediciones obtenidas

En esta etapa del diagrama de bloque, se realizan distintas pruebas para calcular el error que existe en las mediciones realizadas, contrastando los datos del nivel de ruido e iluminación del prototipo con aplicaciones móviles. A continuación, en la Tabla 2.8 se muestra los escenarios y las evidencias de las mediciones realizadas.

Tabla 2.8 Escenarios de las mediciones realizadas

Escenario	Ubicación	Imágenes
Vivienda	Dormitorio	
Biblioteca	Biblioteca de la Facultad de Ingeniería Eléctrica y Electrónica.	
Oficinas	Oficinas del club de robótica	
Aula	Laboratorio de Comunicación Digital	

Los resultados y análisis de las mediciones se van a desarrollar en el Capítulo 3. Además, en el Anexo III, se puede observar el video de las mediciones realizadas en los distintos escenarios.

3. PRUEBAS, RESULTADOS, CONCLUSIONES Y RECOMENDACIONES

En el presente capitulo se desarrollan pruebas de la medición del nivel de ruido e iluminación en cuatro escenarios *indoor*. Se realiza un monitoreo en tiempo real y se contrastan los valores con mediciones tomadas con aplicaciones móviles que midan el nivel de ruido e iluminación, de esta manera se procede a calcular el error que existe en las mediciones hechas con el prototipo.

3.1 Pruebas

Las pruebas se realizan en cuatro escenarios (vivienda, oficinas, biblioteca y aulas de clase), las medidas se efectúan en la mañana, tarde y noche, de esta forma se pretende conocer el nivel de ruido e iluminación al que están expuestos los usuarios a lo largo del día. Las aplicaciones que se van a utilizar son Decibel X y Luxómetro, las cuales se detallan a continuación.

3.1.1 Decibel X

Es una aplicación móvil que sirve para medir el nivel de ruido en dB, su ventaja es que se puede configurar la calibración y de esta forma conseguir niveles de ruido más altos o bajos. Si se accede a la membresía premium, se puede guardar y exportar los datos que se mida en un periodo de tiempo en formato WAV, CSV o HTML. La interfaz gráfica se muestra en la Figura 3.1.

Figura 3.1 Interfaz gráfica de la aplicación Decibel X [27].

3.1.2 Luxómetro

Es una aplicación móvil que mide la iluminancia o el nivel de iluminación en luxes, como el sensor de cada dispositivo móvil es diferente, la aplicación cuenta con la opción de calibración para obtener valores más exactos. Es gratuito y permite grabar y exportar los datos que se miden en formato texto o CSV. En los dispositivos móviles el sensor de iluminación se encuentra en la pantalla. La interfaz gráfica se muestra en la Figura 3.2.

Figura 3.2 Interfaz gráfica de la aplicación Luxómetro [28].

3.1.3 Mediciones del nivel de Ruido e Iluminación

Se realizan 10 mediciones cada segundo del nivel de ruido e iluminación, este valor se escogió luego de realizar pruebas.

En Tabla 3.1 se observa que a partir de la medida 5, el nivel de iluminación se estabiliza. Además, como la señal de ruido varía muy rápido con el tiempo, se debe tomar medidas durante un período de tiempo, si se busca establecer el tipo de ambiente sonoro en el que se encuentra. En el Anexo II se puede observar el *dataset* creado a partir de las pruebas.

Tabla 3.1 Mediciones de la iluminancia con el prototipo y la aplicación Luxómetro.

N° medida	Prototipo [lux]	Luxómetro [lux]	Error [%]
1	125,83	144	12,62
2	126,67	137	7,54
3	124,17	127	2,23
4	121,67	123	1,08
5	120,83	122	0,96
6	120,83	122	0,96
7	120	121	0,83
8	120	121	0,83
9	120	119	0,84
10	120	120	0,00

3.1.3.1 Mediciones del nivel de ruido en la vivienda

Se toman 10 medidas en un dormitorio, el horario en el que se tomó las medidas fue la noche. En la Tabla 3.2 se muestra el valor medido con el prototipo, la aplicación móvil y el error que existe. Para que las mediciones sean acertadas se debe colocar el micrófono del celular y del sensor MAX9814, en la misma posición y distancia de la fuente de ruido.

Tabla 3.2 Mediciones del nivel de ruido con el prototipo y la aplicación Decibel X.

	N° medida	Prototipo [dB]	Decibel X [dB]	Error [%]
Ambiente poco ruidoso	1	23,74	24,6	3,50
	2	25,89	24,4	6,11
	3	28,46	24,7	15,22
	4	25,86	28	7,64
	5	23,58	25	5,68
	6	22,96	24,7	7,04
	7	24,12	24,7	2,35
	8	25,28	24,3	4,03
	9	19,8	25	20,80
	10	24,21	24,4	0,78
			Error promedio	7,32

Se observa que en promedio existe un error de 7,32%, esto quiere decir que las mediciones realizadas son acertadas como se observa en la Figura 3.3, el único valor atípico que se hallo es la medida 9, esto se debe a que la señal de ruido varía muy rápido con el tiempo.

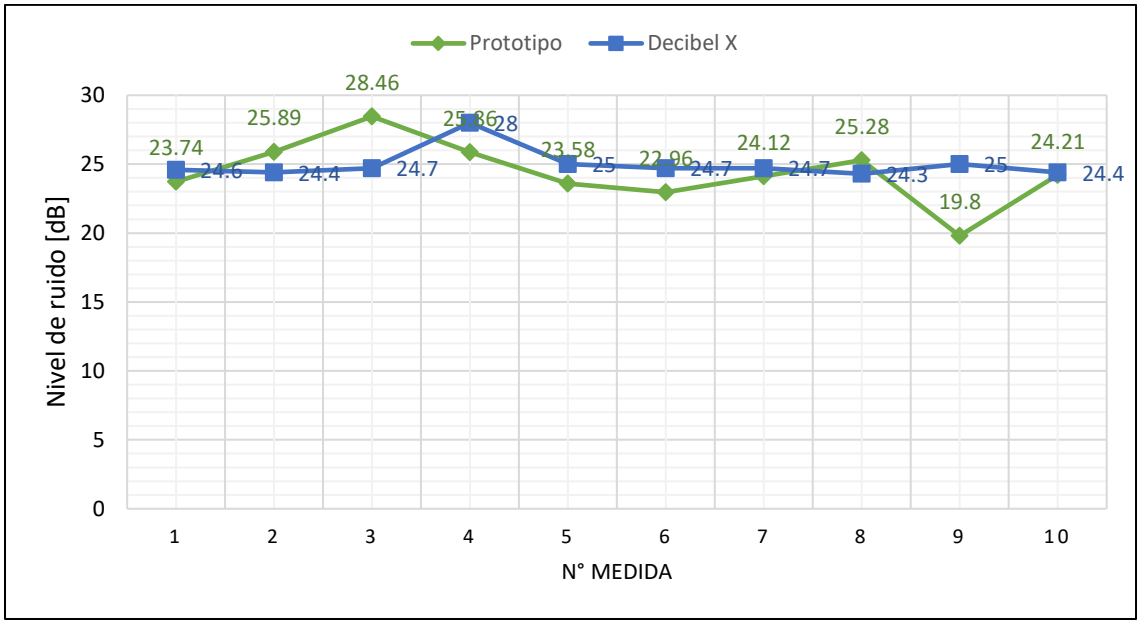


Figura 3.3 Mediciones del nivel de ruido con el prototipo y Decibel X en la vivienda.

3.1.3.2 Mediciones del nivel de iluminación en la vivienda

Para este escenario se realizan en el dormitorio 10 mediciones del nivel de iluminación en la tarde, donde la fuente de luz es el sol. Como se observa en Tabla 1.5 el nivel de iluminación adecuado es de 100 a 200 luxes en este escenario.

Tabla 3.3 Mediciones de iluminancia con el prototipo y la aplicación Luxómetro.

	N° medida	Prototipo [lux]	Luxómetro [lux]	Error [%]
Vivienda dormitorio	1	125,83	144	12,62
	2	126,67	137	7,54
	3	124,17	127	2,23
	4	121,67	123	1,08
	5	120,83	122	0,96
	6	120,83	122	0,96
	7	120	121	0,83
	8	120	121	0,83
	9	120	119	0,84
	10	120	120	0,00
			Error promedio	2,79

En la Tabla 3.3 se muestra que el error promedio es bajo e igual a 2,79%. En la Figura 3.4 se observa que mientras avanzó el tiempo, las medidas cada vez fueron más acertadas.

Se debe tomar en cuenta que el sensor (BH1750) y la pantalla del celular, deben estar en la misma posición y ángulo para recibir la luz.

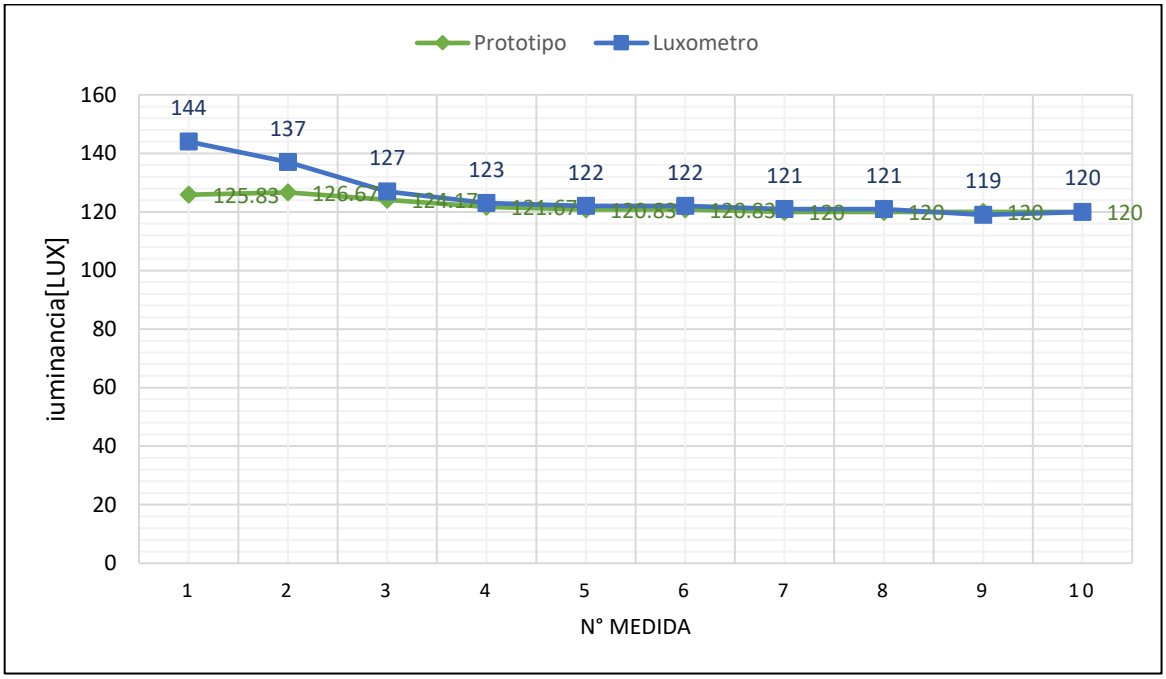


Figura 3.4 Mediciones de la iluminancia con el prototipo y Luxómetro en la vivienda.

3.1.3.3 Mediciones del nivel de ruido en biblioteca

Las mediciones se realizaron en la biblioteca de la Faculta de Ingeniería Eléctrica y Electrónica (FIEE), aproximadamente a las 11:00 am.

Tabla 3.4 Mediciones del nivel de ruido con el prototipo y la aplicación Decibel X.

	N° medida	Prototipo [dB]	Decibel X [dB]	Error [%]
Ambiente ruidoso	1	62,19	56,9	9,30
	2	60,81	58,4	4,13
	3	61,6	60,2	2,33
	4	44,97	48,1	6,51
	5	60,17	59,9	0,45
	6	41,87	52,5	20,25
	7	61,55	58,6	5,03
	8	60,62	60,3	0,53
	9	42,61	44	3,16
	10	60,98	58,8	3,71
			Error promedio	5,54

Debido a la afluencia de estudiantes se puede observar en Tabla 3.4 que la mayoría de valores del nivel de ruido están por encima de los 60 dB, por lo que se encuentran en un ambiente ruidoso. En la Figura 3.5 se muestra que los datos obtenidos con el prototipo y la aplicación son similares, a excepción de la medida 6, que presenta un mayor error.

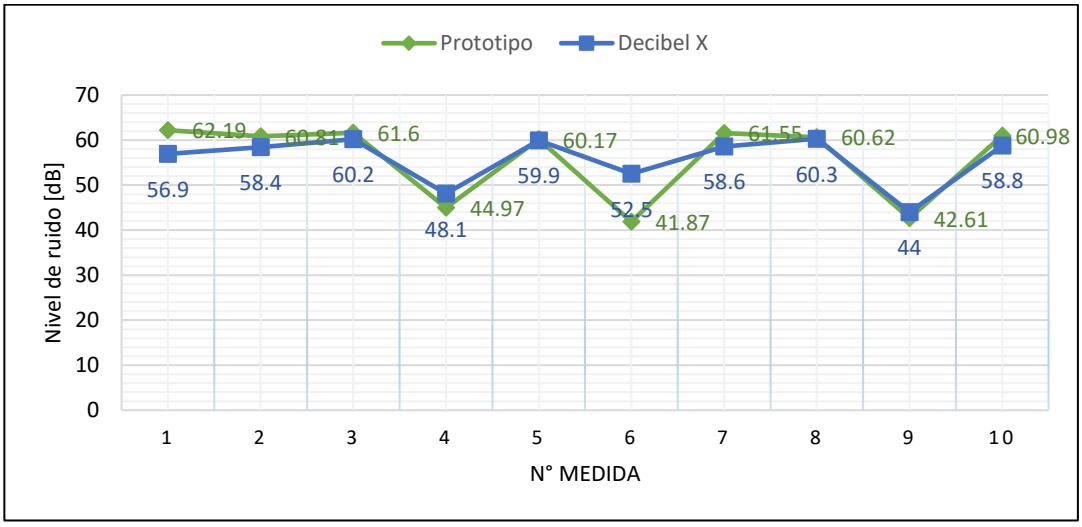


Figura 3.5 Mediciones del nivel de ruido con el prototipo y Decibel X en la biblioteca de la FIEE.

3.1.3.4 Mediciones del nivel de iluminación en biblioteca

Las mediciones se realizaron en la biblioteca de la Facultad de Ingeniería Eléctrica y Electrónica, durante la mañana, aproximadamente a las 11:00 am, por lo que la fuente de luz era el sol.

Tabla 3.5 Mediciones de iluminancia con el prototipo y la aplicación Luxómetro

	N° medida	Prototipo [lux]	Luxómetro [lux]	Error [%]
Vivienda dormitorio	1	609,17	608	0,19
	2	601,67	600	0,28
	3	603,33	601	0,39
	4	605,83	603	0,47
	5	608,33	606	0,38
	6	610	608	0,33
	7	611,67	611	0,11
	8	612,5	612	0,08
	9	612,5	611	0,25
	10	612,5	614	0,24
			Error promedio	0,27

En la Tabla 3.5 se puede observar que el nivel de iluminación estuvo alrededor de los 600 luxes, por lo que la iluminancia es adecuada para este escenario. En la Figura 3.6 se observa que los datos obtenidos son muy similares y que existió una pequeña caída del nivel de iluminación, pero de tan solo 10 luxes al inicio de las mediciones.

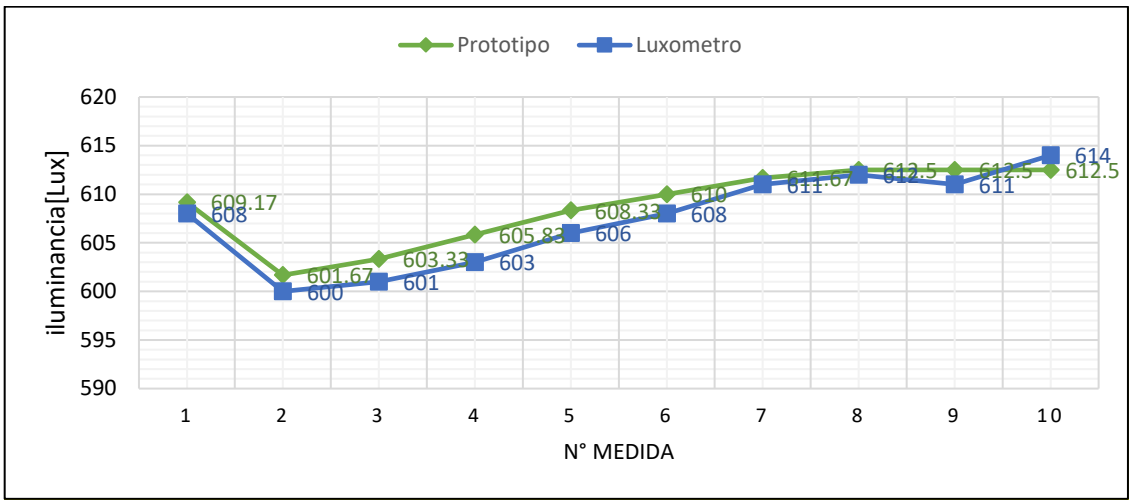


Figura 3.6 Mediciones de la iluminancia con el prototipo y Luxómetro en la biblioteca de la FIEE.

3.1.3.5 Mediciones del nivel de ruido en oficinas

El nivel de ruido en oficinas tranquilas y en una conversación normal, debe ser 50 y 60 dB respectivamente, según la Tabla 1.3. En este escenario se realizaron 10 medidas en las oficinas del club de robótica.

Tabla 3.6 Mediciones del nivel de ruido con el prototipo y la aplicación Decibel X.

	N° medida	Prototipo [dB]	Decibel X [dB]	Error [%]
Ambiente ruidoso	1	61,58	58,1	5,99
	2	60,73	57	6,54
	3	61,77	58	6,50
	4	61,93	59,4	4,26
	5	62,04	58,2	6,60
	6	61,69	63,7	3,16
	7	85,73	64,3	33,33
	8	62,76	61	2,89
	9	61,48	58,3	5,45
	10	62,12	60	3,53
			Error promedio	7,82

En la Tabla 3.6 y la Figura 3.7 se observa que el único valor atípico, se produjo en la medida 7, en este caso hay que tomar en cuenta que el valor que mide el sensor y envía a Telegram, presenta más retraso que la aplicación Decibel X, lo que puede provocar que algún dato presente alto error.

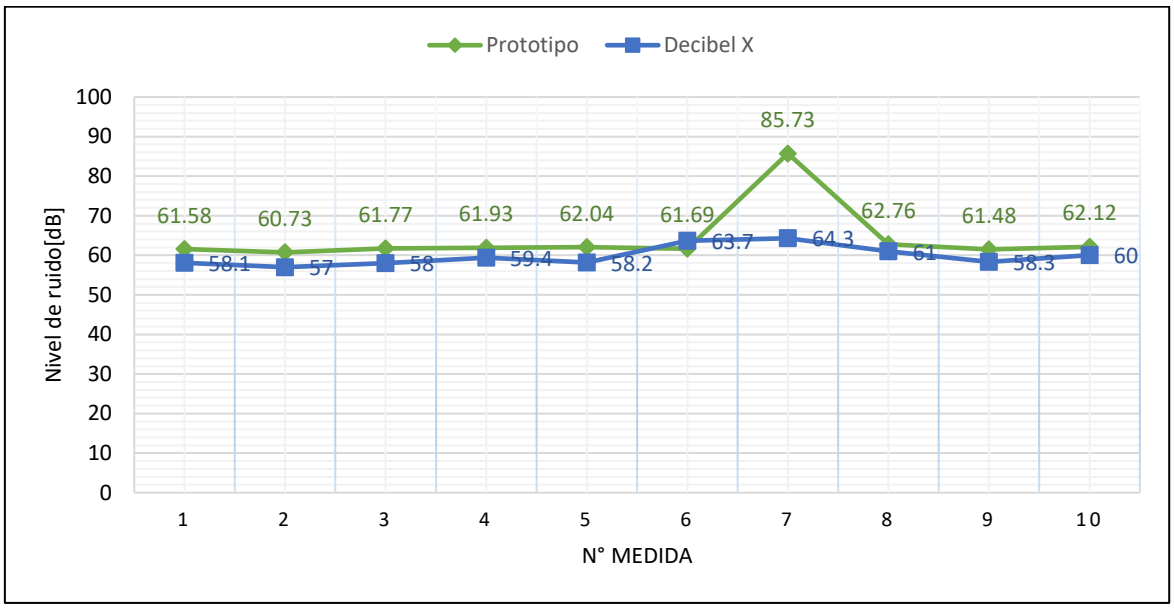


Figura 3.7 Grafico de las mediciones realizadas con el prototipo y Decibel X.

3.1.3.6 Mediciones del nivel de iluminación en oficinas

Se realizó las mediciones en las oficinas del club de robótica al mediodía, a esta hora existía luz solar, pero por la arquitectura de la oficina no llega tanto los rayos del sol y la fuente principal de luz es artificial.

Tabla 3.7 Mediciones de iluminancia con el prototipo y la aplicación Luxómetro.

	N° medida	Prototipo [dB]	Luxómetro [dB]	Error [%]
Oficina pequeña	1	208,33	211	1,27
	2	209,17	211	0,87
	3	209,17	211	0,87
	4	209,17	211	0,87
	5	209,17	211	0,87
	6	208,33	211	1,27
	7	209,17	211	0,87
	8	209,17	211	0,87
	9	209,17	211	0,87
	10	209,17	211	0,87
			Error promedio	0,95

En la Tabla 3.7 se puede observar que los valores obtenidos del nivel de iluminación esta entre 208 y 211 luxes, por lo que al momento de las mediciones no existe un adecuado ambiente lumínico. En la Figura 3.8 se observa que la iluminancia con la aplicación Luxómetro fue lineal, mientras que con el prototipo solo la medida 1 y 6 cambiaron.

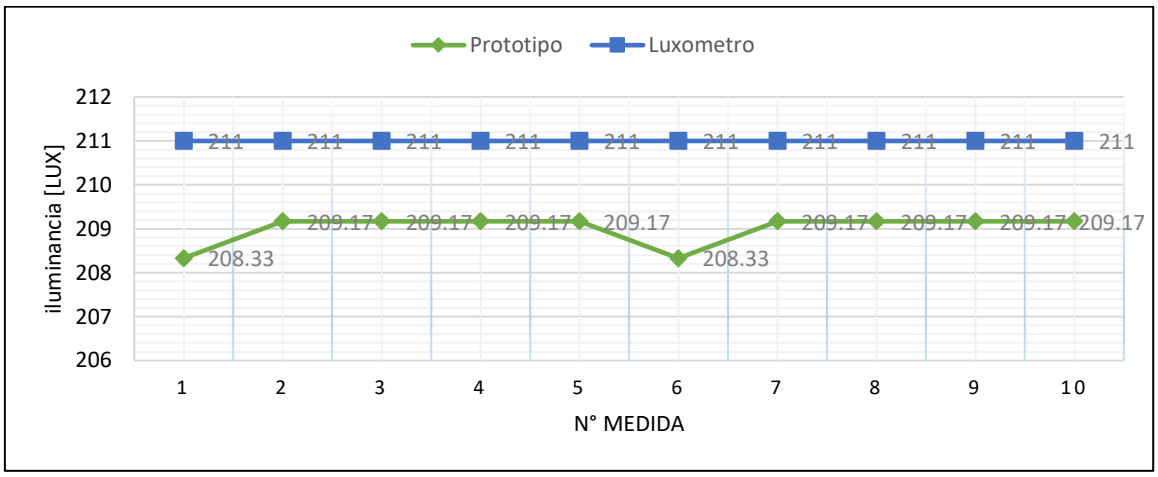


Figura 3.8 Mediciones de la iluminancia con el prototipo y Luxómetro en oficina pequeña.

3.1.3.7 Mediciones del nivel de ruido en aulas de clase

Las mediciones se realizaron en el laboratorio de Comunicación Digital de la Facultad de Ingeniería Eléctrica y Electrónica, en el horario de 2 a 4 pm durante el desarrollo de prácticas de laboratorio. En este escenario se tiene que los estudiantes estaban conversando en voz baja, pero en pocos momentos existió silencio.

Tabla 3.8 Mediciones del nivel de ruido con el prototipo y la aplicación Decibel X.

	N° medida	Prototipo [dB]	Decibel X [dB]	Error [%]
Ambiente ruidoso	1	46,67	43,8	6,55
	2	47,52	43,1	10,26
	3	45,74	44	3,95
	4	44,65	48,2	7,37
	5	23,27	24,6	5,41
	6	46,95	46,8	0,32
	7	23,35	24,9	6,22
	8	45,67	43	6,21
	9	47,71	44,7	6,73
	10	45,48	43,6	4,31
			Error promedio	5,73

En la Tabla 3.8 se muestra que los datos obtenidos cuando existe variación del nivel de ruido, son muy cercanos. En la Figura 3.9 se observa que el nivel de ruido disminuyo dos veces y el prototipo capto esta señal correctamente.

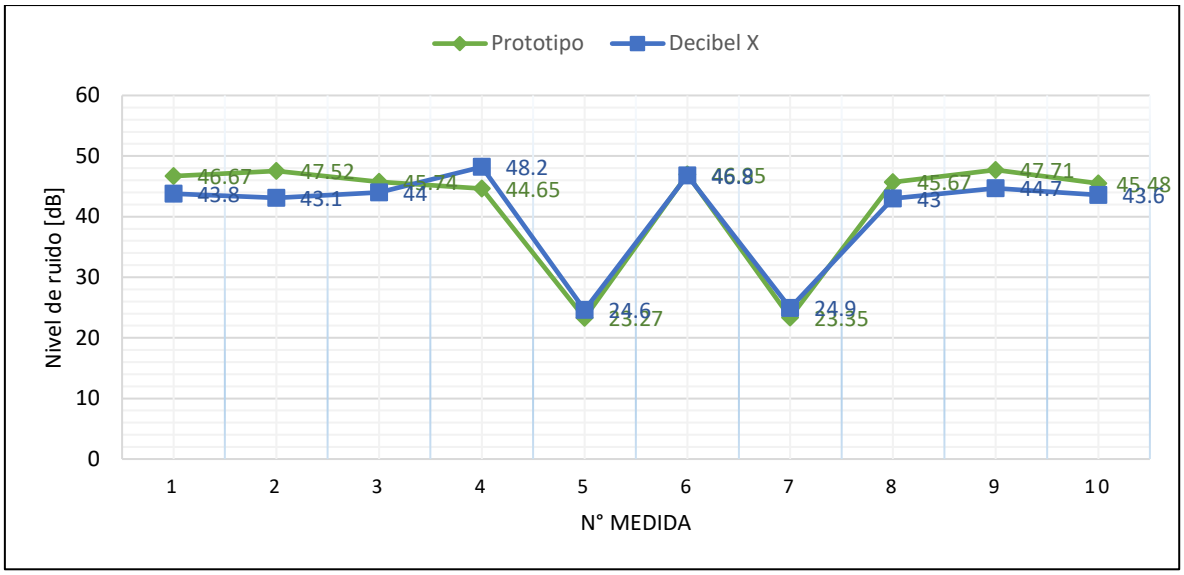


Figura 3.9 Grafico de las mediciones realizadas con el prototipo y Decibel X.

3.1.3.8 Mediciones del nivel de iluminación en aulas de clase

Las pruebas se realizaron en el laboratorio de Comunicación Digital de la Facultad de Ingeniería Eléctrica y Electrónica, aproximadamente a las 14:00 horas, durante el desarrollo de prácticas de laboratorio, la fuente de luz fue el sol pero también existió fuentes artificiales.

Tabla 3.9 Mediciones de iluminancia con el prototipo y la aplicación Luxómetro.

	N° medida	Prototipo [dB]	Luxómetro[Lux]	Error [%]
Aula	1	218,33	215	1,55
	2	218,33	215	1,55
	3	217,5	215	1,16
	4	216,67	215	0,78
	5	217,5	217	0,23
	6	217,5	217	0,23
	7	218,33	215	1,55
	8	220	215	2,33
	9	219,17	215	1,94
	10	217,5	215	1,16
			Error promedio	1,25

En la Tabla 3.9 se muestra que las mediciones tomadas por el prototipo tienen mayor variación y que son más reales que la aplicación Luxómetro, la cual solo muestra dos variaciones en la iluminancia. En la Figura 3.10 se observa que el nivel de iluminación fue casi el mismo en las medidas 5 y 6.

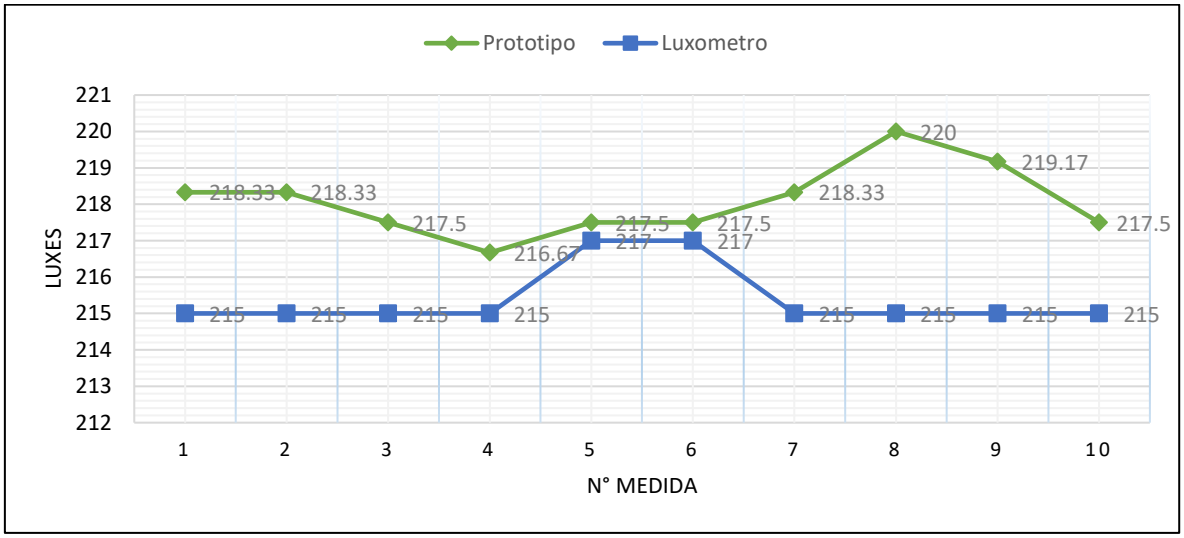


Figura 3.10 Mediciones de la iluminancia con el prototipo y Luxómetro en aula.

3.2 Conclusiones

La placa ESP32 ha demostrado por su costo y características, ser la elección correcta para realizar el prototipo de detección de ruido e iluminación, ya que los dos sensores utilizados para realizar las mediciones, transmitieron los datos a la placa sin problemas. Además, existen varias herramientas y fuentes en Internet que proporcionan información acerca de la configuración de la misma. La conexión de la placa a Internet por medio de WiFi es su principal ventaja.

El sensor de sonido MAX9814 captó el ruido satisfactoriamente en las pruebas realizadas en los distintos escenarios. Además, al contar con fases de amplificación y ganancia, lo hacen adecuado para realizar mediciones del nivel de ruido. Para obtener un rango amplio en las mediciones del nivel de ruido se incluyó un factor de calibración, este se obtuvo mediante pruebas realizadas con la aplicación móvil Decibel X.

El sensor BH1750 mide el nivel de iluminación en luxes, por lo que la programación resulta más sencilla que al usar otros sensores, tiene un amplio rango de medición, por lo que tiene la capacidad de medir una alta iluminancia. Realizando pruebas se observó que en ambientes *indoor* no se presentan niveles altos de iluminación.

Con ayuda de Arduino IoT Cloud, se puede monitorear en tiempo real los datos de cualquier tipo de sensor, esta herramienta presenta distintas opciones para visualizar las mediciones y su principal ventaja es que se puede crear las variables y *dashboards* que se requieran.

El prototipo implementado permite al usuario por medio de la aplicación de mensajería Telegram, que pueda con un mensaje al *bot* (ilumrui), realizar las mediciones del nivel de ruido e iluminación y conocer si el ambiente en el que se encuentra es adecuado o no, dependiendo de los datos obtenidos por los sensores. Además, para comodidad del usuario se creó el estado alerta, en el cual el prototipo está midiendo durante un periodo de tiempo los niveles de ruido e iluminación y notifica al usuario cuando estos han sobrepasado niveles altos que pueden ser perjudiciales para la salud de las personas.

Para realizar las pruebas de funcionamiento, el prototipo debe estar conectado a una red WiFi, al realizar las mediciones en tiempo real, puede existir algún retraso en el envío del mensaje dependiendo de la velocidad de acceso a Internet y la señal de la red a la que se está conectando,

Al realizar las pruebas de funcionamiento y contrastar las mediciones de los sensores con las aplicaciones móviles, se encontró que el sensor BH1750, mide de mejor forma el nivel de iluminación que la aplicación Luxómetro, esto debido a que el sensor de luz del celular, tiene el objetivo de detectar luz en el ambiente y de esta forma subir o bajar el brillo, más no medir la iluminancia. Mientras que la aplicación Decibel X, detecta de mejor forma el sonido que el sensor MAX9814, esto se debe a que los *smathphones* en la actualidad tienen micrófonos de excelente calidad que permiten que las aplicaciones midan el nivel de ruido con gran precisión.

3.3 Recomendaciones

En el mercado existe gran variedad de sensores de luz y sonido, antes de escoger un de ellos es importante definir cuál es el propósito del prototipo, de esta forma se podrá seleccionar el sensor adecuado, ya que se ha realizado pruebas con el sensor de sonido KY-038, pero se evidenció que sólo detecta el sonido que este cercano al micrófono. Por otra parte, el módulo LDR, que detecta luz, no es adecuado para medir el nivel de iluminación debido a que se debe usar una fórmula para transformar el voltaje ADC a luxes y esta fórmula presenta limitaciones cuando el nivel de iluminación es alto.

La placa ESP32 tiene varios pines en los que se puede conectar los sensores, se recomienda escoger el pin dependiendo de la función que cumplan, ya que algunos pines son usados al momento de conectarse a Internet, y si estos también se conectan a los

sensores, no va a existir comunicación serial y el monitor serie del IDE de Arduino no va a mostrar nada.

Para realizar las mediciones de los datos con Telegram, en la programación se requiere el token del bot, la clave y el nombre de la red WiFi, estos datos deben estar en otra pestaña del IDE de Arduino y ser llamados en el código fuente con (#include "nombreArchivo"), esto se realiza por motivos de seguridad, ya que si alguien conoce estos datos puede realizar modificaciones al *bot*.

Para mejorar la detección de sonido se puede cambiar el sensor MAX9814 por uno que cuente con un micrófono de mejor calidad y sensibilidad. En la fase de exploración del mercado se evidenció que en el mercado Ecuatoriano es difícil encontrar este tipo de sensores.

4. REFERENCIAS BIBLIOGRÁFICAS

[1] Hernández O, Hernández-Montero G, López E. “Ruido y salud - Revista Cubana de Medicina Militar”. Noviembre, 2019. Accedido: 1 de noviembre de 2022. [En línea]. Disponible en: http://www.revmedmilitar.sld.cu/index.php/mil/article/view/431

[2] Álvarez, C. “Identificación de riesgos del nivel de iluminación de aulas, talleres y laboratorios de la Facultad de Mecánica – ESPOCH bajo normas vigentes”. Escuela Superior Politécnica de Chimborazo. Junio, 2015. Accedido: 1 de noviembre de 2022. [En línea]. Disponible en: http://dspace.espoch.edu.ec/handle/123456789/4180

[3] Vásconez, C.” Sistema de monitoreo de nivel de ruido ambiental para el casco central de la ciudad de Ambato”. Noviembre, 2018. Accedido: 1 de noviembre de 2022 [En línea]. Disponible en: https://repositorio.uta.edu.ec/bitstream/123456789/28939/1/Tesis_t1495ec.pdf

[4] A. Álvarez, J. Patiño. “Plataforma de detección de sonidos y generación de servicios en el hogar basado en enfoques de IoT como ayuda a las personas con discapacidad auditiva para afianzar su proceso de autonomía y vivencia social”. 2020. [En línea]. Disponible en: http://repository.unipiloto.edu.co/bitstream/handle/20.500.12277/9922/DocumentoProyectodeGrado.pdf?sequence=1&isAllowed=y

[5] B. Ubeda. “Apuntes de: Sistemas embebidos”. 2009. Accedido: 9 de noviembre de 2022. [En línea]. Disponible en: https://www.um.es/documents/4874468/19345367/ssee-t01.pdf/4ea71f56-2950-4c3f-acbe-e7699e490f4e

[6] Hobbydu. “Sensor de sonido LM393 de micrófono compatible con Arduino”. 2022. Accedido: 22 de agosto de 2022. [En línea]. Disponible en: https://hobbydu.com/modulos-sonido-para-arduino/sensor-de-sonido-50.html

[7] Industrias GSL. “Sensor de Luz”. Julio, 2021. Accedido: 22 de agosto de 2022. [En línea]. Disponible en: https://industriasgsl.com/blogs/automatizacion/sensor_de_luz

[8] Escuela Colombiana de Ingeniería Julio Garavito “Laboratorio de Condiciones de Trabajo - RUIDO”. Facultad de Ingeniería Industrial. Enero de 2011. Accedido: 24 de octubre de 2022. [En línea]. Disponible en: https://escuelaing.s3.amazonaws.com/staging/documents/7863_ruido.pdf

[9] H. Arqué. “NIVELES DE RUIDO SUPERIORES A LA TOLERANCIA”. Octubre, 2017. Accedido: 14 de noviembre de 2022. [En línea]. Disponible en: https://www.salesa.es/es/noticias/niveles-de-ruido-superiores-a-la-tolerancia/_noticia:110/

[10] L. Z. Narváez, M. T. Rentería, M. F. Díaz, M. Sarria-Paja and J. Ochoa, "Estimating noise pollution caused by vehicular traffic in an institution of higher education in the city of Cali," 2019. Accedido: 1 de noviembre de 2022 [En línea]. Disponible en: https://ieeexplore.ieee.org/document/8730265

[11] Asociación Médica Mundial. “Declaración de la AMM sobre la contaminación acústica”. Septiembre, 2022. Accedido: 1 de noviembre de 2022. [En línea]. Disponible en: https://www.wma.net/es/policies-post/declaracion-de-la-amm-sobre-la-contaminacion-acustica/#:~:text=La%20Organizaci%C3%B3n%20Mundial%20de%20la,el%20sue%C3%B1o%20y%20el%20rendimiento

[12] Instituto Nacional de Seguridad e Higiene en el Trabajo (INSHT). “Iluminación en el puesto de trabajo. Criterios para la evaluación y acondicionamiento de los puestos”. Diciembre, 2015. Accedido: 28 de diciembre de 2022. [En línea]. Disponible en: https://www.insst.es/documents/94886/96076/Iluminacion+en+el+puesto+de+trabajo/9f9299b8-ec3c-449e-81af-2f178848fd0a

[13] GoLed by Ozado. “¿Qué cantidad de luxes es la apropiada para iluminar un determinado ambiente ?”. Diciembre, 2020. Accedido: 28 de diciembre de 2022. [En linea]. Disponible en: https://goledperu.com/que-cantidad-de-luxes-es-la-apropiada-para-iluminar-un-determinado-ambiente/

[14] Á. Boehmwald. “Contaminación Lumínica”. Mayo, 2019. Accedido: 3 de noviembre de 2022. [En línea]. Disponible en: https://planesynormas.mma.gob.cl/archivos/2020/proyectos/Contaminacion_Luminica__Boehmwald_R._.pdf

[15] J. Guerra. “Arduino Mega 2560 el hermano mayor de Arduino UNO”. 2020. Accedido: 30 de noviembre de 2022. [En línea]. Disponible en: https://programarfacil.com/blog/arduino-blog/arduino-mega-2560/#comment-5149903283

[16] R. Alonso. “Raspberry Pi vs Arduino, ¿en qué se diferencian y para qué se usan?”. Marzo, 2022. Accedido: 30 de noviembre de 2022. [En línea]. Disponible en: https://hardzone.es/reportajes/comparativas/raspberry-pi-vs-arduino/

[17] Espressif Systems. “ESP32 Series Datasheet”. 2022. Accedido: 28 de noviembre de 2022. [En línea]. Disponible en: https://www.espressif.com/sites/default/files/documentation/esp32_datasheet_en.pdf

[18] Analog Devices, “Microphone Amplifier with AGC and Low-Noise Microphone Bias”. febrero de 2020. Accedido: 30 de noviembre de 2022. [En línea]. Disponible en: https://datasheets.maximintegrated.com/en/ds/MAX9814.pdf

[19] Analog Devices, “Omnidirectional Microphone with Bottom Port and Analog Output ADMP401”. 2012. Accedido: 30 de noviembre de 2022. [En línea]. Disponible en: https://www.analog.com/media/en/technical-documentation/obsolete-data-sheets/ADMP401.pdf

[20] Electrostore. “Módulo micrófono con sensor de sonido ky-038”. 2019. Accedido: 28 de diciembre de 2022. [En línea]. Disponible en: https://grupoelectrostore.com/shop/sensores/sonido/modulo-microfono-con-sensor-de-sonido-ky-038/

[21] Carrod Electrónica. “Módulo Sensor de Luz LDR por Fotoresistencia”. 2014. Accedido: 28 de diciembre de 2022. [En línea]. Disponible en: https://www.carrod.mx/products/tapete-para-raton-liso-antiderrapante

[22] Art of Circuits. “GY-302 BH1750 16-Bit Serial Output Light LUX Sensor Module”. 2022. Accedido: 28 de diciembre de 2022. [En línea]. Disponible en: https://artofcircuits.com/product/gy-302-bh1750-16-bit-serial-output-light-lux-sensor-module

[23] Naylamp Mechatronics. “Tutorial módulo sensor de luz BH1750”. Noviembre de 2016. Accedido: 30 de noviembre de 2022. [En línea]. Disponible en: https://naylampmechatronics.com/blog/44_tutorial-modulo-sensor-de-luz-bh1750.html

[24] C. Toapanta. “Sistema de monitoreo de factores ambientales externos en unidades educativas céntricas del cantón Píllaro basado en tecnología Lora”. Agosto de 2021. Accedido: 3 de enero de 2023. [En línea]. Disponible en: https://repositorio.uta.edu.ec/bitstream/123456789/33713/1/t1882ec.pdf

[25] Hua-Dong Ma. “Internet of Things: Objectives and Scientific Challenges”. Septiembre de 2011. Accedido: 30 de noviembre de 2022. [En línea]. Disponible en: https://jcst.ict.ac.cn/EN/Y2011/V26/I6/919

[26] EasyEDA. “The Next Generation of PCB Design Tool”. 2023. Accedido: 30 de enero de 2023. [En línea]. Disponible en: https://easyeda.com/

[27] SkyPaw Co. “Decibel X: dBA Sonómetro Pro”. 2023. Accedido: 30 de enero de 2023. [En línea]. Disponible en: https://play.google.com/store/apps/details?id=com.skypaw.decibel&hl=es_EC&gl=US

[28] Crunchy ByteBox. “Luxómetro”. 2019. Accedido: 30 de enero de 2023. [En línea]. Disponible en: https://play.google.com/store/apps/details?id=crunchybytebox.lightmeter&hl=es_EC&gl=US

5 ANEXOS

ANEXO I. Código completo para medir el nivel de ruido e iluminación desde Telegram y monitorear los datos con Arduino IoT Cloud.

ANEXO II. *Dataset* de los datos medidos en las pruebas.

ANEXO III. Video de las mediciones realizadas en las pruebas.

ANEXO I. Código completo para medir el nivel de ruido e iluminación desde Telegram y monitorear los datos con Arduino IoT Cloud

Código I.1 Código completo para medir el nivel de ruido e iluminación desde Telegram y monitorear los datos con Arduino IoT Cloud

```
#include "thingProperties.h"

//INICIO ILUM
#include <BH1750.h>
#include <Wire.h>
BH1750 lightMeter;
//FIN ILUM

//INICIO RUIDO
#include <math.h>
int sensorPIN = 34;
//FIN RUIDO

// INICIO BOT TELEGRAM
template<class T> inline Print &operator <<(Print &obj, T arg) {
  obj.print(arg);
  return obj;
}
#include "CTBot.h"
CTBot miBot;
#include "contr.h"//archivo que contiene la contraseña, nombre de red y
el token del bot de Telegram
//FIN BOT TELEGRAM
```

```
void setup() {
  Serial.begin(9600); //Vtx
  //INICIO ILUM
  Wire.begin();
  lightMeter.begin();
  //FIN ILUM

  //IOT ARDUINO
  initProperties();
  ArduinoCloud.begin(ArduinoIoTPreferredConnection);
  setDebugMessageLevel(2);
  ArduinoCloud.printDebugInfo();
  //FIN IOT ARDUINO

  //INICIO BOT TELEGRAM
  Serial.println("_____________________________");
  Serial.println("Iniciando Bot de Telegram");
  miBot.wifiConnect(ssid, password);
  miBot.setTelegramToken(token);
  if (miBot.testConnection()) {
    Serial.println("\n Conectado");
    Serial.println("_____________________________");
  }
  else {
    Serial.println("\n No se pudo conectar el bot! Revisar la red
WiFi!");
    Serial.println("_____________________________");
  }
  //FIN BOT TELEGRAM
}
```

```
void loop() {
  //_____________________________________________COMIENZA EL CODIGO
LOOP_____________________________________________//
  TBMessage msg;
  ArduinoCloud.update();

  //Si se recibe un mensaje

  if (CTBotMessageText == miBot.getNewMessage(msg)) {
    Serial << "Mensaje: " << msg.sender.firstName << " - " <<  msg.text
<< "\n";
    //Se escoge entre iluminacion o ruido
    if (msg.text.equalsIgnoreCase("ilumination")) {
      miBot.sendMessage(msg.sender.id, "Midiendo nivel de
iluminacion");
      miBot.sendMessage(msg.sender.id, "Usted se encuentra en: \n
Vivienda \n LugarEstudio \n Oficina");
    }
    else if ((msg.text.equalsIgnoreCase("Vivienda"))) {
      miBot.sendMessage(msg.sender.id, "Usted se encuentra en:\n
Dormitorio \n AreaEstar \n Areaestudio");
    }
    else if (msg.text.equalsIgnoreCase("dormitorio")) {
      ilum = lightMeter.readLightLevel();
      if (ilum > 100 && ilum < 200) {
        miBot.sendMessage(msg.sender.id, "El nivel de iluminacion ES
ADECUADO");
        //LAZO FOR para medir durante 10 segundos
        for (int cont = 0; cont < 10; cont++) {
          ArduinoCloud.update();
          ilum = lightMeter.readLightLevel();
          char smsIlum[100];
          snprintf(smsIlum, 100, "El nivel de iluminacion es: %.2f lx",
ilum);
          Serial.println(smsIlum);
          miBot.sendMessage(msg.sender.id, smsIlum);
          delay(250);
        }
```

```
    miBot.sendMessage(msg.sender.id, "Se ha terminado de realizar 10
mediciones");
      } else {
        miBot.sendMessage(msg.sender.id, "El nivel de iluminacion NO ES
ADECUADO");
        for (int cont = 0; cont < 10; cont++) {
          ArduinoCloud.update();
          ilum = lightMeter.readLightLevel();
          char smsIlum[100];
          snprintf(smsIlum, 100, "El nivel de iluminacion es: %.2f lx",
ilum);
          Serial.println(smsIlum);
          miBot.sendMessage(msg.sender.id, smsIlum);
          delay(250);
        }
        miBot.sendMessage(msg.sender.id, "Se ha terminado de realizar
10 mediciones");
      }

    }

    else if (msg.text.equalsIgnoreCase("Areaestar")) {
      ilum = lightMeter.readLightLevel();
      if (ilum > 200 && ilum < 500) {
        miBot.sendMessage(msg.sender.id, "El nivel de iluminacion ES
ADECUADO ");
        //LAZO FOR para medir durante 10 segundos
        for (int cont = 0; cont < 10; cont++) {
          ArduinoCloud.update();
          ilum = lightMeter.readLightLevel();
          char smsIlum[100];
          snprintf(smsIlum, 100, "El nivel de iluminacion es: %.2f lx",
ilum);
          Serial.println(smsIlum);
          miBot.sendMessage(msg.sender.id, smsIlum);
          delay(250);
        }
        miBot.sendMessage(msg.sender.id, "Se ha terminado de realizar
10 mediciones");
```

```
 } else {
        miBot.sendMessage(msg.sender.id, "El nivel de iluminacion NO ES
ADECUADO  ");
        for (int cont = 0; cont < 10; cont++) {
          ArduinoCloud.update();
          ilum = lightMeter.readLightLevel();
          char smsIlum[100];
          snprintf(smsIlum, 100, "El nivel de iluminacion es: %.2f lx",
ilum);
          Serial.println(smsIlum);
          miBot.sendMessage(msg.sender.id, smsIlum);
          delay(250);
        }
        miBot.sendMessage(msg.sender.id, "Se ha terminado de realizar
10 mediciones");
      }

    }
    else if (msg.text.equalsIgnoreCase("Areaestudio")) {
      ilum = lightMeter.readLightLevel();
      if (ilum > 300 && ilum < 750) {
        miBot.sendMessage(msg.sender.id, "El nivel de iluminacion ES
ADECUADO :)");
        //LAZO FOR para medir durante 10 segundos
        for (int cont = 0; cont < 10; cont++) {
          ArduinoCloud.update();
          ilum = lightMeter.readLightLevel();
          char smsIlum[100];
          snprintf(smsIlum, 100, "El nivel de iluminacion es: %.2f lx",
ilum);
          Serial.println(smsIlum);
          miBot.sendMessage(msg.sender.id, smsIlum);
          delay(250);
        }
        miBot.sendMessage(msg.sender.id, "Se ha terminado de realizar
10 mediciones");
      } else {
```

```
       miBot.sendMessage(msg.sender.id, "El nivel de iluminacion NO ES
ADECUADO! ");
        for (int cont = 0; cont < 10; cont++) {
          ArduinoCloud.update();
          ilum = lightMeter.readLightLevel();
          char smsIlum[100];
          snprintf(smsIlum, 100, "El nivel de iluminacion es: %.2f lx",
ilum);
          Serial.println(smsIlum);
          miBot.sendMessage(msg.sender.id, smsIlum);
          delay(250);
        }
        miBot.sendMessage(msg.sender.id, "Se ha terminado de realizar
10 mediciones");
      }

    }
    else if ((msg.text.equalsIgnoreCase("lugarestudio"))) {
      miBot.sendMessage(msg.sender.id, "Usted se encuentra en: \n aula
\n biblioteca");

    }

    else if (msg.text.equalsIgnoreCase("Aula")) {
      ilum = lightMeter.readLightLevel();
      if (ilum > 300 && ilum < 500) {
        miBot.sendMessage(msg.sender.id, "El nivel de iluminacion ES
ADECUADO :)");
        //LAZO FOR para medir durante 10 segundos
        for (int cont = 0; cont < 10; cont++) {
          ArduinoCloud.update();
          ilum = lightMeter.readLightLevel();
          char smsIlum[100];
          snprintf(smsIlum, 100, "El nivel de iluminacion es: %.2f lx",
ilum);
          Serial.println(smsIlum);
          miBot.sendMessage(msg.sender.id, smsIlum);
          delay(250);
        }
        miBot.sendMessage(msg.sender.id, "Se ha terminado de realizar
10 mediciones");
```

```
     } else {
        miBot.sendMessage(msg.sender.id, "El nivel de iluminacion NO ES
ADECUADO! ");
        for (int cont = 0; cont < 10; cont++) {
          ArduinoCloud.update();
          ilum = lightMeter.readLightLevel();
          char smsIlum[100];
          snprintf(smsIlum, 100, "El nivel de iluminacion es: %.2f lx",
ilum);
          Serial.println(smsIlum);
          miBot.sendMessage(msg.sender.id, smsIlum);
          delay(250);
        }
        miBot.sendMessage(msg.sender.id, "Se ha terminado de realizar
10 mediciones");
      }

    }
    else if (msg.text.equalsIgnoreCase("biblioteca")) {
      ilum = lightMeter.readLightLevel();
      if (ilum > 300 && ilum < 750) {
        miBot.sendMessage(msg.sender.id, "El nivel de iluminacion ES
ADECUADO :)");
        //LAZO FOR para medir durante 10 segundos
        for (int cont = 0; cont < 10; cont++) {
          ArduinoCloud.update();
          ilum = lightMeter.readLightLevel();
          char smsIlum[100];
          snprintf(smsIlum, 100, "El nivel de iluminacion es: %.2f lx",
ilum);
          Serial.println(smsIlum);
          miBot.sendMessage(msg.sender.id, smsIlum);
          delay(250);
        }
        miBot.sendMessage(msg.sender.id, "Se ha terminado de realizar
10 mediciones");
      } else {
```

```
       miBot.sendMessage(msg.sender.id, "El nivel de iluminacion NO ES
ADECUADO! ");
        for (int cont = 0; cont < 10; cont++) {
          ArduinoCloud.update();
          ilum = lightMeter.readLightLevel();
          char smsIlum[100];
          snprintf(smsIlum, 100, "El nivel de iluminacion es: %.2f lx",
ilum);
          Serial.println(smsIlum);
          miBot.sendMessage(msg.sender.id, smsIlum);
          delay(250);
        }
        miBot.sendMessage(msg.sender.id, "Se ha terminado de realizar
10 mediciones");
      }

    }
    else if ((msg.text.equalsIgnoreCase("oficina"))) {
      miBot.sendMessage(msg.sender.id, "Usted se encuentra en: \n
Ofipeque \n Ofigrande");
    }
    else if (msg.text.equalsIgnoreCase("ofipeque")) {
      ilum = lightMeter.readLightLevel();
      if (ilum > 450 && ilum < 750) {
        miBot.sendMessage(msg.sender.id, "El nivel de iluminacion ES
ADECUADO :)");
        //LAZO FOR para medir durante 10 segundos
        for (int cont = 0; cont < 10; cont++) {
          ArduinoCloud.update();
          ilum = lightMeter.readLightLevel();
          char smsIlum[100];
          snprintf(smsIlum, 100, "El nivel de iluminacion es: %.2f lx",
ilum);
          Serial.println(smsIlum);
          miBot.sendMessage(msg.sender.id, smsIlum);
          delay(250);
        }
```

```
       miBot.sendMessage(msg.sender.id, "Se ha terminado de realizar 10
mediciones");
      } else {
        miBot.sendMessage(msg.sender.id, "El nivel de iluminacion NO ES
ADECUADO! ");
        for (int cont = 0; cont < 10; cont++) {
          ArduinoCloud.update();
          ilum = lightMeter.readLightLevel();
          char smsIlum[100];
          snprintf(smsIlum, 100, "El nivel de iluminacion es: %.2f lx",
ilum);
          Serial.println(smsIlum);
          miBot.sendMessage(msg.sender.id, smsIlum);
          delay(250);
        }
        miBot.sendMessage(msg.sender.id, "Se ha terminado de realizar
10 mediciones");
      }

    }
    else if (msg.text.equalsIgnoreCase("ofigrande")) {
      ilum = lightMeter.readLightLevel();
      if (ilum > 500 && ilum < 1000) {
        miBot.sendMessage(msg.sender.id, "El nivel de iluminacion ES
ADECUADO :)");
        //LAZO FOR para medir durante 10 segundos
        for (int cont = 0; cont < 10; cont++) {
          ArduinoCloud.update();
          ilum = lightMeter.readLightLevel();
          char smsIlum[100];
          snprintf(smsIlum, 100, "El nivel de iluminacion es: %.2f lx",
ilum);
          Serial.println(smsIlum);
          miBot.sendMessage(msg.sender.id, smsIlum);
          delay(250);
        }
        miBot.sendMessage(msg.sender.id, "Se ha terminado de realizar
10 mediciones");
```

```
     } else {
        miBot.sendMessage(msg.sender.id, "El nivel de iluminacion NO ES
ADECUADO! ");
        for (int cont = 0; cont < 10; cont++) {
          ArduinoCloud.update();
          ilum = lightMeter.readLightLevel();
          char smsIlum[100];
          snprintf(smsIlum, 100, "El nivel de iluminacion es: %.2f lx",
ilum);
          Serial.println(smsIlum);
          miBot.sendMessage(msg.sender.id, smsIlum);
          delay(250);
        }
        miBot.sendMessage(msg.sender.id, "Se ha terminado de realizar
10 mediciones");
      }

    }
    //RUIDO
    else if (msg.text.equalsIgnoreCase("ruido")) {
      Serial.println("DETECTOR DEL NIVEL DE RUIDO");
      ArduinoCloud.update();
      miBot.sendMessage(msg.sender.id, "Midiendo nivel de ruido");
      //Empieza el lazo for que toma 10 medidas
      for (int cont = 0; cont < 10; cont++) {
        ArduinoCloud.update();
        // Declaracion de las varibles del sensor MAX9814
        int S_MAX = -42;   // Sensibilidad del microfono dB
        int G_MAX = 60;    // ganancia del sensor dB
        int Ventana = 50;  //toma MUESTRAS por 50ms
        int Out_ADC = 0; // se declara la variable del voltaje ADC
        long startMillis = millis(); // Inicio de la ventana
        int ADCpp = 0;         // valor del voltaje pico-pico ADC
        int signalMax = 0;
        int signalMin = 2800;  // 12 bit ADC = 2^12=4095 pero el sensor
solo envia hasta 2800
        // se recoge los datos durante la ventana de tiempo
```

```
       while (millis() - startMillis < Ventana) {
          Out_ADC = analogRead(sensorPIN);
          if (Out_ADC < 2800) {
            // ver si tiene un nuevo maximo
            if (Out_ADC > signalMax) {
              signalMax = Out_ADC;  // guarda solo los niveles máximos
            }
            else if (Out_ADC < signalMin) {
              signalMin = Out_ADC;  // guardar sólo los niveles minimos
            }
          }
        }
        ADCpp = signalMax - signalMin; // max - min = amplitud pico-
pico ADC
        double Vs, V_dB;
        //double r_dB;
        Vs = ((ADCpp * 2.256) / 2800); // los 2.256 es el voltaje ADC
que envia placa ESP32 - 2800->2.256V
        V_dB = 20 * log10(Vs / 0.007943); //  0.007943-> -42dB es la
sensibilidad del microfono
        r_dB = V_dB + 94 + S_MAX - G_MAX;  // calculo del nivel de
ruido
        // IF que divide el ambiente sonoro
        if (ADCpp == 0) {
          Serial.println("Tomo el valor de 0, algo esta mal!");
        } else if (ADCpp > 1 && ADCpp < 700) {
          char smsRui[100];
          snprintf(smsRui, 100, "Se encuentra en un ambiente poco
ruidoso, el nivel de ruido es: %.2f dB", r_dB);
          Serial.println(ADCpp);
          Serial.println(r_dB);
          miBot.sendMessage(msg.sender.id, smsRui);
          delay(100);
        } else if (ADCpp > 700 && ADCpp < 1400) {
          r_dB = r_dB + 10.0;
          char smsRui[100];
          snprintf(smsRui, 100, "Se encuentra en un ambiente ruidoso,
el nivel de ruido es: %.2f dB", r_dB);
          Serial.println(ADCpp);
          Serial.println(r_dB);
          miBot.sendMessage(msg.sender.id, smsRui);
          delay(100);
```

```
       } else if (ADCpp > 1400 && ADCpp < 2100) {
          r_dB = r_dB + 20.0;
          char smsRui[100];
          snprintf(smsRui, 100, "Se encuentra en un ambiente ruidoso,
el nivel de ruido es: %.2f dB", r_dB);
          Serial.println(ADCpp);
          Serial.println(r_dB);
          miBot.sendMessage(msg.sender.id, smsRui);
          delay(100);
        } else if (ADCpp > 2100 && ADCpp < 2790) {
          r_dB = r_dB + 45.0;
          char smsRui[100];
          snprintf(smsRui, 100, "Se encuentra en un ambiente ruidoso,
el nivel de ruido es: %.2f dB", r_dB);
          Serial.println(ADCpp);
          Serial.println(r_dB);
          miBot.sendMessage(msg.sender.id, smsRui);
          delay(100);
        } else {
          r_dB = r_dB + 50.0;
          char smsRui[100];
          snprintf(smsRui, 100, "Se encuentra en un ambiente molesto,
el nivel de ruido es: %.2f dB", r_dB);
          Serial.println(r_dB);
          miBot.sendMessage(msg.sender.id, smsRui);
          delay(100);
        }
      } //Termina el lazo for que toma 10 medidas de ruido
      miBot.sendMessage(msg.sender.id, "Se ha terminado de realizar 10
mediciones");
      }else if(msg.text.equalsIgnoreCase("Alerta")) {
        miBot.sendMessage(msg.sender.id, "Se esta tomando mediciones.\n
Se enviara un mensaje en caso de que el nivel de ruido o iluminacion
aumente drasticamente ");
        //_____________ALERTA_____________________________________________
        for (int cont = 0; cont < 300; cont++) { //300-> 5 minutos
          ArduinoCloud.update();
          //Si sube drasticamente los niveles de iluminacion
          ilum = lightMeter.readLightLevel();
          if(ilum>1000.0){
            char smsIlum[100];
            snprintf(smsIlum, 100, "El nivel de iluminacion incremento
drasticamente: %.2f lx", ilum);
            Serial.println(smsIlum);
            miBot.sendMessage(msg.sender.id, smsIlum);
          }
```

```
          //RUIDO ALERTA
             int S_MAX = -42;   // Sensibilidad del microfono dB
             int G_MAX = 60;    // ganancia del sensor dB
             int Ventana = 50;  //toma MUESTRAS por 50ms
             int Out_ADC = 0; // se declara la variable del voltaje ADC
             long startMillis = millis(); // Inicio de la ventana
             int ADCpp = 0;         // valor del voltaje pico-pico ADC
             int signalMax = 0;
             int signalMin = 2800;  // 12 bit ADC = 2^12=4095 pero el
sensor solo envia hasta 2800
             // se recoge los datos durante la ventana de tiempo
             while (millis() - startMillis < Ventana) {
               Out_ADC = analogRead(sensorPIN);
               if (Out_ADC < 2800) {
                 // ver si tiene un nuevo maximo
                 if (Out_ADC > signalMax) {
                   signalMax = Out_ADC;  // guarda solo los niveles
máximos
                 }
                 else if (Out_ADC < signalMin) {
                   signalMin = Out_ADC;  // guardar sólo los niveles
minimos
                 }
               }
             }
             ADCpp = signalMax - signalMin; // max - min = amplitud
pico-pico ADC
             double Vs, V_dB;
             //double r_dB;
             Vs = ((ADCpp * 2.256) / 2800); // los 2.256 es el voltaje
ADC que envia placa ESP32 - 2800->2.256V
             V_dB = 20 * log10(Vs / 0.007943); //  0.007943-> -42dB es
la sensibilidad del microfono
             r_dB = V_dB + 94 + S_MAX - G_MAX;  // calculo del nivel de
ruido
             if (ADCpp > 2500 && ADCpp < 2805) {
               r_dB = r_dB + 50.0;
               char smsRui[100];
               snprintf(smsRui, 100, "Se encuentra en un ambiente
MOLESTO, el nivel de ruido es: %.2f dB", r_dB);
               Serial.println(r_dB);
               miBot.sendMessage(msg.sender.id, smsRui);
               delay(100);
             }
             delay(500);
         }
```

```
    }
     else {
       //mensaje
       miBot.sendMessage(msg.sender.id, "Bienvenido " +
msg.sender.firstName + ", para comenzar la medicion escribe la palabra:
\n Ilumination \n Ruido \n Alerta");
     }
  }

}//FIN DEL
LOOP
```

ANEXO II. *Dataset* de los datos medidos en las pruebas

Se creó un repositorio en Github, donde se puede encontrar el *dataset* en formato Excel y CSV.

Link: https://github.com/MichaelPrado27/Dataset-Ruido-Iluminacion.git

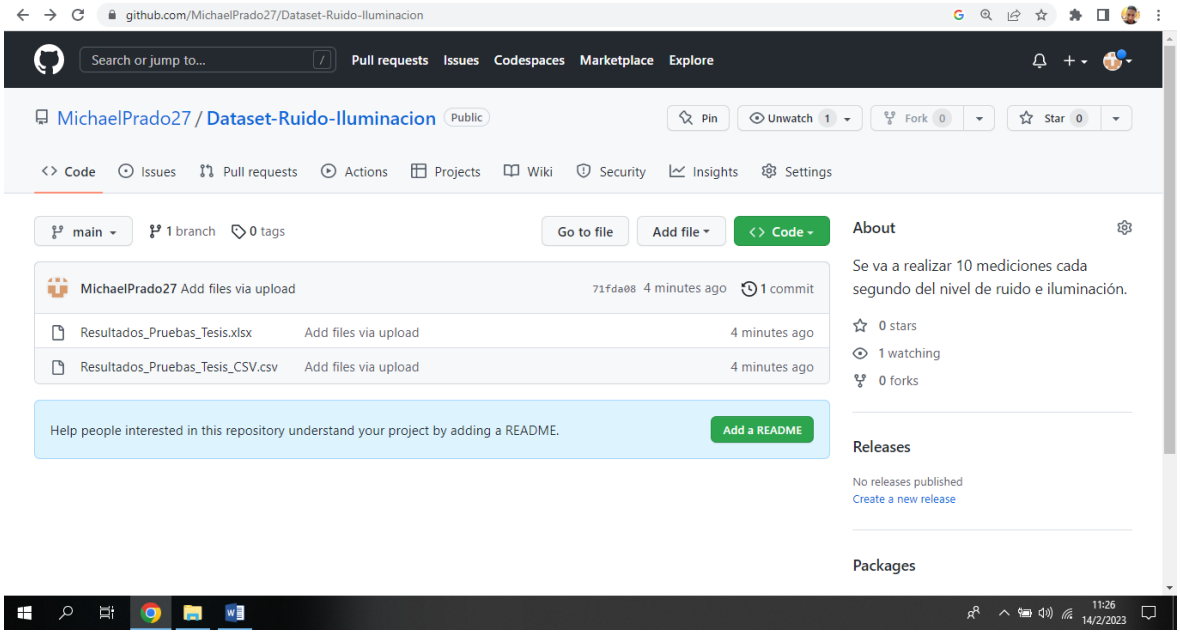


Figura II.1 *Dataset* en formato Excel y CSV en Github

ANEXO III. Video de las mediciones realizadas en las pruebas

Link: https://www.youtube.com/watch?v=4PTj29iS1IA

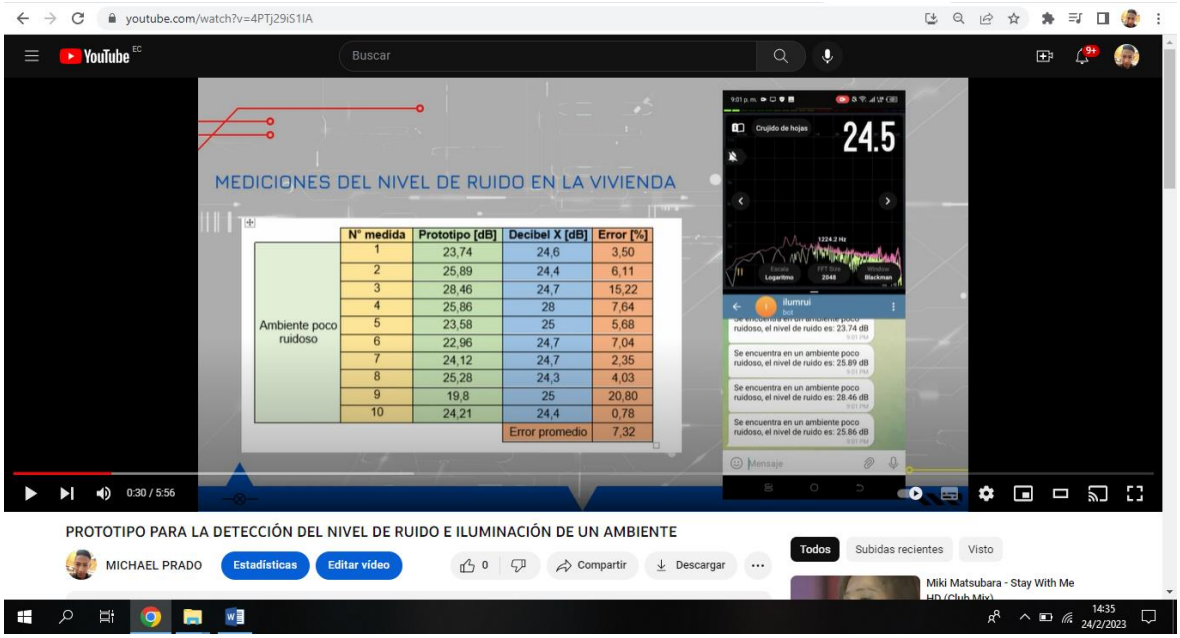


Figura III.1 Video en YouTube de las mediciones realizadas en las pruebas