



UNIVERSIDAD
DE ATACAMA

FACULTAD TECNOLÓGICA

TECNÓLOGO EN INSTRUMENTACIÓN Y AUTOMATIZACIÓN INDUSTRIAL

**MEDICIÓN DE LA CONDICIÓN DEL AIRE CON SENSOR DE MATERIAL
PARTICULADO CONECTADO A MICROCONTROLADOR ESP32**

Ignacio Nicolas Gamboa Rivera

Copiapó, Chile 2023



UNIVERSIDAD
DE ATACAMA

FACULTAD TECNOLÓGICA

TECNÓLOGO EN INSTRUMENTACIÓN Y AUTOMATIZACIÓN INDUSTRIAL

**MEDICIÓN DE LA CONDICIÓN DEL AIRE CON SENSOR DE MATERIAL
PARTICULADO CONECTADO A MICROCONTROLADOR ESP32**

Trabajo de titulación presentado en conformidad a los requisitos para obtener el título de
Tecnólogo en Instrumentación y Automatización Industrial.

Profesor guía: Ricardo Quiroz Concha

Copiapó, Chile 2023

RESUMEN

En un contexto de creciente urbanización e industrialización, la contaminación atmosférica, particularmente la generada por la industria minera, se ha convertido en un desafío crítico para la salud pública y el medio ambiente. Este estudio propuso abordar este problema mediante el desarrollo e implementación de un sistema de monitoreo de material particulado (PM2.5). Se empleó el microcontrolador LilyGo ESP32 modelo T-Beam V1.1 y el sensor GP2Y1010AU0F-V3, para detectar partículas en el aire.

La metodología abarcó desde el análisis y selección de componentes hasta el diseño detallado del circuito electrónico mediante software AutoCAD. Se construyó un prototipo físico, que fue sometido a pruebas para verificar su precisión y confiabilidad. La comparación de resultados antes y después de la implementación permitió identificar mejoras y ajustes necesarios.

Este enfoque integral busca no solo abordar la problemática específica de la contaminación minera en la zona urbana, sino también sentar las bases para futuras aplicaciones en la gestión de la calidad del aire en diversos entornos.

PALABRAS CLAVES

SENSOR - MICROCONTROLADOR

ABSTRACT

In a context of increasing urbanization and industrialization, air pollution, particularly that generated by the mining industry, has become a critical challenge for public health and the environment. This study proposed to address this problem by developing and implementing a particulate matter (PM2.5) monitoring system. The LilyGo ESP32 microcontroller model T-Beam V1.1 and the GP2Y1010AU0F-V3 sensor were used to detect particles in the air.

The methodology ranged from the analysis and selection of components to the detailed design of the electronic circuit using AutoCAD software. A physical prototype was built and tested for accuracy and reliability. The comparison of results before and after implementation made it possible to identify improvements and necessary adjustments.

This comprehensive approach seeks not only to address the specific problem of mining pollution in the urban area, but also to lay the foundations for future applications in air quality management in various environments.

KEYWORDS

SENSOR – MICROCONTROLLER

INDICE

CAPÍTULO I	1
1 INTRODUCCIÓN	1
1.1 OBJETIVO GENERAL	2
1.2 OBJETIVOS ESPECÍFICOS	2
1.3 PLANTEAMIENTO DEL PROBLEMA	2
CAPÍTULO II	3
2 MARCO TEÓRICO	3
2.1 MATERIAL PARTICULADO	3
2.1.1 ¿QUÉ ES EL MP, Y CÓMO LLEGA AL AIRE?	3
2.1.2 FUENTES DE MP	4
2.1.3 EFECTOS DEL MATERIAL PARTICULADO (PM) SOBRE LA SALUD Y EL MEDIOAMBIENTE	5
2.2.1 TRANSDUCTOR	9
2.2.1.1 COMPARATIVA DE SENSORES	10
2.2.1.3 SENSOR DE MATERIAL PARTICULADO GP2Y1010AU0F	14
2.2.2 MICROCONTROLADOR	16
2.2.2.1 ¿QUÉ ES UN MICROCONTROLADOR?	16
2.2.2.2 COMPARACIONES ENTRE MICROCONTROLADORES	17
2.2.2.3 PLACA DE DESARROLLO ESP32	21
2.3 ARDUINO SOFTWARE	23
2.4 CHAT GPT	27
CAPÍTULO III	29
3 METODOLOGÍA	29
3.1 INTRODUCCIÓN	29

3.2 CONFIGURACIÓN INICIAL	29
3.2.1 RECOPIACIÓN DE MATERIALES NECESARIOS	29
3.2.2 CREACIÓN DE LA CUENTA THINKSPEAK:	30
3.3 CONEXIÓN DE LOS COMPONENTES	33
3.4 PROGRAMACIÓN DE LA PLACA	34
CAPÍTULO IV	48
4 MEJORAS	48
4.1 ADICIÓN DE ALERTA ACÚSTICA PARA CONDICIONES DE CONTAMINACIÓN:	49
4.2 ADICIÓN DEL SISTEMA AUXILIAR DE ALIMENTACIÓN PARA REFORZAR LA AUTONOMÍA DEL SISTEMA DE MEDICIÓN DE MP.	55
4.3 OTRAS OPCIONES CONSIDERADAS PARA MEJORAS FUTURAS ...	57
BIBLIOGRAFÍA	59
ANEXO	61

CAPÍTULO I

1 INTRODUCCIÓN

En un mundo cada vez más conectado y tecnológicamente avanzado, los microcontroladores y los sensores pueden surgir como herramientas vitales a la hora de abordar la contaminación ambiental y el impacto que genera en la salud humana. Estas pequeñas pero poderosas innovaciones tecnológicas ayudan a que comprendamos y nos concienticemos en la calidad del aire y los agentes contaminantes en nuestro entorno.

La creciente urbanización, la industrialización y el aumento en la movilidad, han dado lugar a niveles alarmantes de contaminación atmosférica en muchas regiones del mundo. Los agentes contaminantes como las partículas finas, los compuestos orgánicos volátiles y los gases nocivos se han convertido en una amenaza persistente para la salud de las personas y el bienestar del planeta. La exposición prolongada a estas sustancias tóxicas ha sido vinculada a una amplia gama de problemas de salud, desde enfermedades respiratorias y cardiovasculares hasta el aumento del riesgo de cáncer y trastornos neurodegenerativos.

En este contexto, los microcontroladores y los sensores asumen un papel esencial al permitirnos monitorear la concentración de contaminantes en el aire. Estas tecnologías no solo nos impulsan a medir la contaminación, sino que también abren la puerta a una comprensión más profunda de sus efectos perjudiciales en la salud humana.

Los microcontroladores son unidades de procesamiento altamente eficientes que pueden ejecutar tareas específicas y controlar dispositivos electrónicos. Cuando se combinan con sensores, pueden colaborar a detectar la presencia de agentes contaminantes y recopilar datos en tiempo real.

1.1 OBJETIVO GENERAL

Proponer un sistema de monitoreo de material particulado (PM2.5) mediante la aplicación de componentes electrónicos, con el fin de proporcionar información en tiempo real sobre la condición del aire en un entorno específico.

1.2 OBJETIVOS ESPECÍFICOS

1. Analizar las características de los componentes a utilizar y seleccionar los más idóneos mediante comparación de estos mismos
2. Levantamiento de circuito electrónico y prototipo físico mediante la aplicación de software AutoCad.
3. Desarrollar circuito físico para demostración del principio de funcionamiento y verificación de su precisión y confiabilidad.
4. Comparación, mejoras antes y después de la solución propuesta.

1.3 PLANTEAMIENTO DEL PROBLEMA

La industria minera, una fuente vital de recursos naturales y riqueza económica, es un pilar fundamental para el desarrollo de muchas regiones y países en la actualidad. Sin embargo, este sector también enfrenta importantes desafíos ambientales y de salud pública, especialmente en lo que respecta a la contaminación que genera y su impacto en las áreas urbanas cercanas.

El problema central que se aborda en este contexto es el creciente y perjudicial impacto de la contaminación generada por la industria minera en las ciudades. A medida que la demanda mundial de minerales y metales continúa aumentando, las operaciones mineras se expanden y se ubican más cerca de zonas urbanas, lo que aumenta la exposición de las poblaciones locales a diversos agentes contaminantes. El principal aspecto a tratar es la emisión de partículas finas y material particulado que son generados por la extracción, el transporte y el procesamiento de minerales y que se liberan en el aire. Estas partículas pueden ser transportadas por el viento hacia áreas urbanas, lo que resulta en una mala calidad del aire. La exposición crónica a PM2.5 se ha asociado con una serie de problemas de salud, incluyendo enfermedades respiratorias y cardiovasculares.

CAPÍTULO II

2 MARCO TEÓRICO

El material particulado (MP) se refiere a partículas sólidas o líquidas suspendidas en el aire, que pueden ser perjudiciales para la salud y el medio ambiente. Las partículas finas, como PM2.5 y PM10, son especialmente preocupantes debido a su capacidad para ingresar profundamente en nuestros cuerpos. La exposición a MP a largo plazo está relacionada con enfermedades graves. Para monitorear las condiciones del aire, se utilizará el sensor GP2Y1010AU0F-V3, que detecta partículas utilizando luz infrarroja. Los datos recopilados pueden procesarse con un microcontrolador ESP32 y enviarse por Wi-Fi o Bluetooth. Este sistema de detección puede aplicarse para evaluar la calidad del aire en áreas urbanas, evaluar medidas de control de la contaminación y proteger la salud pública.

2.1 MATERIAL PARTICULADO

2.1.1 ¿Qué es el MP, y cómo llega al aire? (Epa, s.f.)

MP es la abreviatura de material particulado (también llamado contaminación por partículas), el término para una mezcla de partículas sólidas y gotas líquidas que se encuentran en el aire. Algunas partículas, como el polvo, la suciedad, el hollín, o el humo, son lo suficientemente grandes y oscuras como para verlas a simple vista. Otras son tan pequeñas que solo pueden detectarse mediante el uso de un microscopio electrónico.

La contaminación por partículas incluye:

- PM10: partículas inhalables que tienen diámetros de, por lo general, 10 micrómetros y menores; y
- PM2,5: partículas inhalables finas que tienen diámetros de, por lo general, 2,5 micrómetros y menores.

¿Qué son 2,5 micrómetros? Tome como referencia un solo cabello de su cabeza. En promedio el cabello humano mide, aproximadamente, 70 micrómetros de diámetro, lo que lo hace 30 veces más grande que la partícula fina más grande.

Figura N° 1 Ejemplo de cabello con respecto a PM



Fuente: Página de la EPA

En Chile los límites de Concentración para Material Particulado Respirable MP2.5 se encuentran en el artículo 3 del decreto 12 “ESTABLECE NORMA PRIMARIA DE CALIDAD AMBIENTAL PARA MATERIAL PARTICULADO FINO RESPIRABLE MP 2,5”:

(congreso, 2011) Artículo 3°. *La norma primaria de calidad del aire para material particulado fino es veinte microgramos por metro cúbico (20 μg/m³), como concentración anual, y cincuenta microgramos por metro cúbico (50 μg/m³), como concentración de 24 horas.*

2.1.2 Fuentes de MP (Epa, s.f.)

Estas partículas vienen en muchos tamaños y formas, y pueden estar conformadas por cientos de diferentes químicos.

Algunas se emiten directamente desde una fuente, como obras en construcción, caminos sin asfaltar, campos, chimeneas o incendios.

La mayoría de las partículas se forman en la atmósfera como resultado de reacciones complejas de químicos, como el dióxido de azufre y los óxidos de nitrógeno, que son contaminantes emitidos por centrales eléctricas, **industrias** y automóviles.

La mina de Chuquicamata de Chile, ha sido un punto de preocupación importante debido a la emisión de material particulado y su significativo impacto ambiental. Siendo una de las minas de cobre más grandes y antiguas del mundo, su operación ha dejado una

profunda huella en el entorno natural, generando partículas finas de material particulado, como el PM2.5 y el PM10. La extracción de minerales en Chuquicamata ha llevado a problemas graves, como la degradación del suelo, la contaminación del agua y la emisión de partículas tóxicas en el aire. Estas partículas suspendidas no solo afectan a las comunidades locales, sino que también tienen un impacto global al contribuir al cambio climático y la mala calidad del aire.

2.1.3 Efectos del material particulado (PM) sobre la salud y el medioambiente

Efectos sobre la salud (EPA, s.f.)

El tamaño de las partículas se encuentra directamente vinculado con el potencial para provocar problemas de salud. Las partículas pequeñas de menos de 10 micrómetros de diámetro suponen los mayores problemas, debido a que pueden llegar a la profundidad de los pulmones, y algunas hasta pueden alcanzar el torrente sanguíneo.

La exposición a estas partículas puede afectar tanto a los pulmones como al corazón. Múltiples estudios científicos vincularon la exposición a la contaminación por partículas a una variedad de problemas, que incluye:

- muerte prematura en personas con enfermedades cardíacas o pulmonares
- infartos de miocardio no mortales
- latidos irregulares
- asma agravada
- función pulmonar reducida
- síntomas respiratorios aumentados, como irritación en las vías respiratorias, tos o dificultad para respirar.

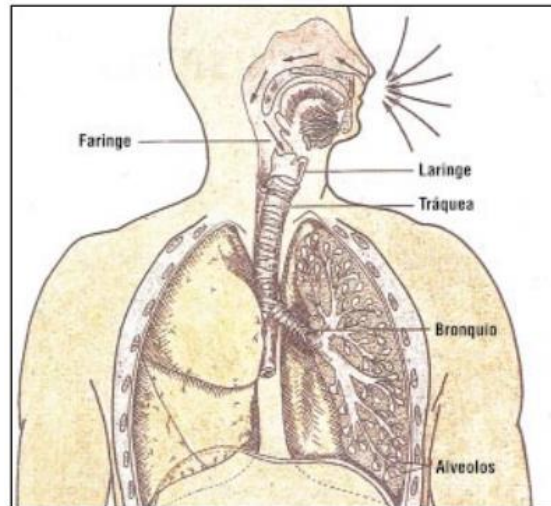
La exposición a la contaminación por partículas tiende a afectar en su mayoría a personas con enfermedades cardíacas o pulmonares, niños y adultos mayores.

El material particulado, en particular el polvo de sílice, está estrechamente relacionado con la silicosis.

¿Qué es la silicosis? (ACHS, s.f.)

Es una enfermedad del pulmón, producida por la permanente aspiración de polvo de sílice libre cristalizada. El polvo de sílice produce alteraciones fibrosas en los pulmones, disminuyendo la capacidad respiratoria porque el pulmón pierde elasticidad. Los alvéolos pulmonares se endurecen al ser reemplazados por callosidades fibróticas, y el enfermo se hace más vulnerable a la tuberculosis.

Figura N° 2 Representación del sistema pulmonar



Fuente: ACHS

La silicosis es una enfermedad irreversible, quien la contrae no tiene posibilidad de sanar. Sólo es posible detener su avance cuando es detectada tempranamente. En las etapas iniciales no hay síntoma alguno, pero en la forma más avanzada la persona se cansa fácilmente ante cualquier esfuerzo. No hay dolor ni otras manifestaciones específicas. Se diagnostica a través de exámenes radiológicos.

¿Qué es la Sílice?

Se llama sílice al Dióxido de Silicio (SiO_2), que se encuentra en la naturaleza en formas muy diversas de cuarzo, como un depósito cristalino. Es uno de los elementos más abundantes en la corteza terrestre. Está presente en casi todos los yacimientos minerales (cobre, hierro, carbón, etc. y en muchas industrias como fundiciones, cerámica, fábrica de vidrios, ladrillos refractarios, limpiado abrasivo, etc.).

El principal agente productor de silicosis es la SILICE LIBRE CRISTALIZADA, que aparece en mayor o menor cantidad en menas y rocas. Su concentración es mayor en trabajos de perforación, carguío de minerales, corrida de buitras, molienda en chancadores y otros, y se debe a que estos procesos generan una gran cantidad de polvo, en el cual se encuentra este elemento nocivo para la salud.

Figura N° 3 Cristal de Cuarzo

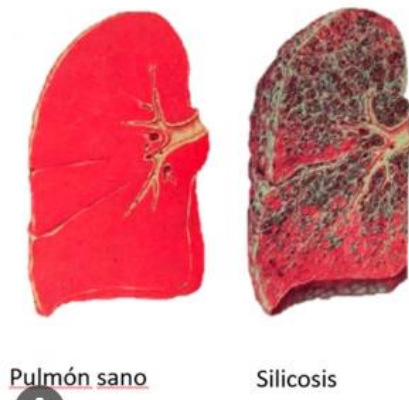


Fuente: ACHS

Cantidad de polvo en el ambiente

La silicosis se contrae por el polvillo de sílice que penetra en los pulmones, cuando la persona respira. A mayor cantidad de polvo en el ambiente y mientras más tiempo permanezcamos en él, más cantidad de sílice entrará a los pulmones.

Figura N° 4 Pulmón sano y Pulmón con silicosis



Fuente: ACHS

Efectos sobre el medioambiente

Deterioro en la visibilidad: Las partículas finas (PM2.5) son la causa principal de visibilidad reducida (bruma).

Figura N° 5 Plano de Ciudad de Calama



Fuente: Google imágenes

Figura N° 6 Fundición de Chuquicamata



Fuente: Google Imágenes

2.2 INTRODUCCION A LOS COMPONENTES Y PROGRAMAS QUE SE UTILIZARÁN EN EL PROYECTO

2.2.1 TRANSDUCTOR (Pallás Areny, 2003)

Se denomina transductor, en general, a todo dispositivo que convierte una señal de una forma física en una señal correspondiente, pero de otra forma física distinta. Es, por tanto, un dispositivo que convierte un tipo de energía en otro. Esto significa que la señal de entrada es siempre una energía o potencia, pero al medir, uno de los componentes de la señal suele ser tan pequeña que puede despreciarse, y se interpreta que se mide solo el otro componente.

Al medir una fuerza, por ejemplo, se supone que el desplazamiento del transductor es despreciable, es decir, que no se “carga” al sistema, ya que de lo contrario podría suceder que éste fuera incapaz de aportar la energía necesaria para el desplazamiento. Pero en la transducción siempre se extrae una cierta energía del sistema donde se mide, por lo que es importante garantizar que esto no lo perturba.

Dado que hay seis tipos de señales: mecánicas, térmicas, magnéticas, eléctricas, ópticas y moleculares (químicas), cualquier dispositivo que convierta una señal de un tipo en una señal de otro tipo debería considerarse un transductor, y la señal de salida podría ser de cualquier forma física “útil”. En la práctica, no obstante, se consideran transductores por

autonomasia (excelencia) aquellos que ofrecen una señal de salida eléctrica. Ello se debe al interés de este tipo de señales en la mayoría de procesos de medida.

Un sensor es un dispositivo que, a partir de la energía del medio donde se mide, da una señal de salida transducible que es función de la variable medida.

2.2.1.1 COMPARATIVA DE SENSORES

GP2Y1010AU0F-V3

Rango de Medición: $0.6 \mu\text{g}/\text{m}^3$ - $1 \text{ mg}/\text{m}^3$

Principio de Funcionamiento: Detección de infrarrojos

Precisión: Moderada

Tamaño de Partículas: $\geq 0.8 \mu\text{m}$

Salida: Analógica

Precio aproximado: \$8.000

Notas: Ampliamente utilizado para detectar partículas grandes en el aire.

Figura N° 7 Sensor GP2Y1010AU0F



Fuente: Google imágenes

SDS011

Rango de Medición: $0.3 \mu\text{g}/\text{m}^3$ - $10 \text{ mg}/\text{m}^3$

Principio de Funcionamiento: Difusión láser

Precisión: Buena

Tamaño de Partículas: 0.3 μm - 10 μm

Salida: Digital

Precio aproximado: \$30.000

Notas: Sensor ampliamente utilizado, alta precisión y fácil de usar.

Figura N° 8 Sensor SDS011



Fuente: Google imágenes

PMS5003

Rango de Medición: 1 $\mu\text{g}/\text{m}^3$ - 1 mg/m^3

Principio de Funcionamiento: Difusión láser

Precisión: Buena

Tamaño de Partículas: 0.3 μm – 10 μm

Salida: Digital

Precio aproximado: \$20.000

Notas: Sensor compacto y preciso con una amplia gama de aplicaciones.

Figura N° 9 Sensor PMS5003



Fuente: Google imágenes

Plantower PMS7003

Rango de Medición: 1 $\mu\text{g}/\text{m}^3$ - 1 mg/m^3

Principio de Funcionamiento: Difusión láser

Precisión: Buena

Tamaño de Partículas: 0.3 μm – 10 μm

Salida: Digital

Precio aproximado: \$20.000

Notas: Similar al PMS5003, pero con algunas características adicionales.

Figura N° 10 Sensor PMS7003



Fuente: Google imágenes

Honeywell HPMA115S0-XXX

Rango de Medición: 0.3 $\mu\text{g}/\text{m}^3$ - 10 mg/m^3

Principio de Funcionamiento: Difusión láser

Precisión: Excelente

Tamaño de Partículas: 0.3 μm - 10 μm

Salida: Digital

Precio aproximado: \$60.000

Notas: Ofrece una alta precisión y una interfaz UART para la comunicación.

Figura N° 11 Sensor HPM115S0-XXX



Fuente: Google imágenes

2.2.1.3 SENSOR DE MATERIAL PARTICULADO GP2Y1010AU0F (Gidahatar, s.f.)

El GP2Y1010AU0F es un sensor óptico de calidad de aire diseñado para medir la densidad de partículas. Es capaz de medir la luz reflectada del polvo en el aire, siendo eficaz para detectar partículas finas como el humo de cigarrillo, también es usado comúnmente en sistema purificadores de aire, tiene bajo consumo energético (20 mA máximo, promedio 11 mA) y la salida del sensor es analógica proporcional a la medida de densidad de polvo.

A continuación, se presentan algunas características claves e información sobre el sensor GP2Y1010AU0F:

1. Principio de funcionamiento: El sensor opera según el principio de dispersión de la luz. Contiene una fuente de luz LED infrarroja y un fotodetector. A medida que las partículas pasan por el área de detección, dispersan la luz de la LED, y el fotodetector mide la intensidad de la luz dispersada. El sensor luego correlaciona esta intensidad de luz con la concentración de partículas.

2. Señal de salida: El sensor proporciona una señal de salida de voltaje analógico que es proporcional a la concentración de partículas en el aire. Este voltaje de salida se puede leer mediante un microcontrolador u otro dispositivo electrónico para su procesamiento adicional.

3. **Rango de medición:** El sensor GP2Y1010AU0F puede medir típicamente concentraciones de partículas en el rango de 0.1 a 10 micrómetros. Este rango cubre partículas finas que suelen estar asociadas con la contaminación del aire en interiores y que pueden afectar la salud humana.

4. **Aplicación:** Este sensor se utiliza a menudo en purificadores de aire, monitores de calidad del aire y sistemas de monitoreo ambiental para indicar los niveles de contaminación del aire. Puede ayudar a los usuarios a comprender la calidad del aire y tomar medidas apropiadas para mitigar posibles riesgos para la salud.

5. **Limitaciones:** Si bien el GP2Y1010AU0F proporciona una solución rentable para la detección de partículas, es importante tener en cuenta que es posible que no proporcione mediciones altamente precisas. Es más adecuado para *evaluaciones cualitativas o relativas de concentraciones* de partículas en lugar de una cuantificación precisa.

Características y Especificaciones:

Dimensiones: El sensor suele ser compacto y consta de una carcasa con un orificio de entrada para las partículas y una ranura para la salida de luz.

Alimentación: Generalmente opera con un voltaje de alimentación de alrededor de 5V.

Salida Analógica: Proporciona una salida analógica en forma de voltaje que varía según la concentración de partículas en el aire.

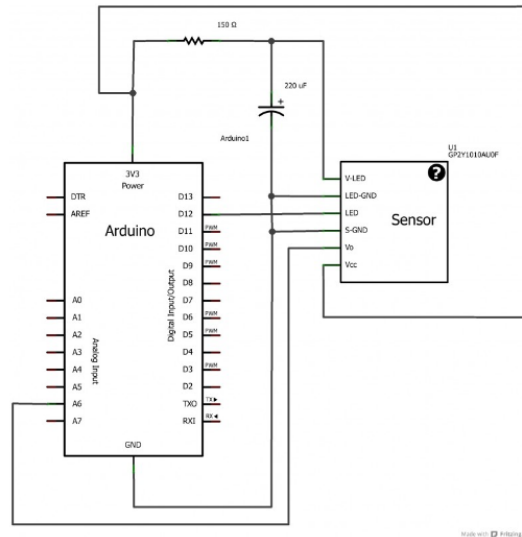
Tiempo de Respuesta: El sensor puede tener un tiempo de respuesta de unos pocos segundos, pero este valor puede variar según el diseño y las condiciones de operación.

Condiciones Ambientales:

Temperatura de Operación: El sensor suele funcionar en un rango de temperaturas específico, generalmente entre -10°C y 65°C. Operar fuera de este rango podría afectar su precisión.

Humedad Relativa: La humedad relativa del entorno también puede influir en las mediciones del sensor. En condiciones de alta humedad, las partículas podrían agruparse y afectar la medición.

Figura N° 12 Esquema del sensor a un microcontrolador



Fuente: Google Imágenes

2.2.2 MICROCONTROLADOR

2.2.2.1 ¿QUÉ ES UN MICROCONTROLADOR? (Wikipedia, Microcontrolador, s.f.)

Un microcontrolador (abreviado μC , UC o mCU) es un circuito integrado programable, capaz de ejecutar las órdenes grabadas en su memoria. Está compuesto de varios bloques funcionales que cumplen una tarea específica. Un microcontrolador incluye en su interior las tres principales unidades funcionales de una computadora: unidad central de procesamiento, memoria y periféricos de entrada/salida.

Algunos microcontroladores pueden utilizar palabras de cuatro bits y funcionan a velocidad de reloj con frecuencias tan bajas como 4 kHz, con un consumo de baja potencia (mW o microwatts). Por lo general, tendrá la capacidad de mantenerse a la espera de un evento como pulsar un botón o de otra interrupción; así, el consumo de energía durante el estado de reposo (reloj de la CPU y los periféricos de la mayoría) puede ser solo de nanowatts, lo que hace que muchos de ellos sean muy adecuados para aplicaciones con batería de larga duración. Otros microcontroladores pueden servir para roles de rendimiento crítico, donde sea necesario actuar más como un procesador digital de señal (DSP), con velocidades de reloj y consumo de energía más altos.

A continuación, se presentan algunas características clave de los microcontroladores:

1. **CPU Incorporada:** Todos los microcontroladores tienen una unidad de procesamiento central (CPU) que ejecuta instrucciones y realiza operaciones de cálculo.
2. **Memoria:** Los microcontroladores tienen memoria incorporada para almacenar programas (memoria de programa o flash) y datos (memoria RAM) necesarios para su funcionamiento.
3. **Puertos de Entrada/Salida (E/S):** Los microcontroladores están equipados con puertos de E/S que permiten la conexión a sensores, actuadores y otros dispositivos externos para interactuar con el entorno.
4. **Temporizadores y Contadores:** Suelen incluir temporizadores y contadores que son esenciales para controlar eventos y medir el tiempo.
5. **Periféricos Específicos:** Pueden incluir periféricos específicos según la aplicación, como convertidores analógico-digital (ADC), convertidores digital-analógico (DAC), interfaces de comunicación (UART, SPI, I2C), entre otros.
6. **Bajo Consumo de Energía:** Muchos microcontroladores están diseñados para ser eficientes en cuanto al consumo de energía, lo que los hace ideales para aplicaciones con baterías o alimentación limitada.

2.2.2.2 COMPARACIONES ENTRE MICROCONTROLADORES

Lilygo T-Beam V1.1 LoRa32:

Microcontrolador: ESP32 (doble núcleo).

Velocidad de Procesador: Hasta 240 MHz.

Conectividad: Wi-Fi, Bluetooth, LoRa, GPS.

Alimentación: Batería LiPo o USB.

Precio aproximado: \$35.000

Notas: Placa con conectividad avanzada, ideal para proyectos y rastreo con GPS.

Figura N° 13 Placa Lilygo T-Beam V1.1



Fuente: Google imágenes

Arduino Uno:

Microcontrolador: ATmega328P.

Velocidad de Procesador: 16 MHz.

Conectividad: USB, UART, I2C, SPI.

Alimentación: 5V a través de USB o 9V.

Precio aproximado: \$20.000

Notas: Placa clásica para proyectos de electrónica con conexiones cableadas y sin conectividad inalámbrica.

Figura N° 14 Placa Arduino Uno



Fuente: Google imágenes

Arduino Leonardo:

Microcontrolador: ATmega32U4.

Velocidad de Procesador: 16 MHz.

Conectividad: USB, UART, I2C, SPI.

Alimentación: 5V a través de USB.

Precio aproximado: \$13.000

Notas: Similar al Arduino Uno, pero con la capacidad adicional de emular un dispositivo USB.

Figura N° 15 Placa Arduino Leonardo



Fuente: Google imágenes

ESP8266 NodeMCU:

Microcontrolador: ESP8266.

Velocidad de Procesador: Hasta 80 MHz.

Conectividad: Wi-Fi.

Alimentación: 3.3V a través de USB.

Precio aproximado: \$5.000

Notas: Placa compacta y económica con conectividad Wi-Fi, ideal para proyectos.

Figura N° 16 Placa ESP8266 NodeMCU



Fuente: Google imágenes

2.2.2.3 PLACA DE DESARROLLO ESP32 (randomnerdtutorials, s.f.)

El ESP32 es una serie de microcontroladores System on a Chip (SoC) de bajo costo y bajo consumo desarrollados por Espressif que incluyen capacidades inalámbricas Wi-Fi y Bluetooth y procesador de doble núcleo. Esta placa tiene las siguientes características:

- **Bajo consumo:** El ESP32 consume muy poca energía en comparación con otros microcontroladores, y admite estados de modo de bajo consumo como la suspensión profunda para ahorrar energía;
- **Capacidades Wi-Fi:** El ESP32 puede conectarse fácilmente a una red Wi-Fi para conectarse a Internet (modo estación), o crear su propia red inalámbrica Wi-Fi (modo punto de acceso) para que otros dispositivos puedan conectarse a ella, puede tener múltiples dispositivos que se comuniquen entre sí utilizando sus capacidades Wi-Fi;
- **Bluetooth:** El ESP32 admite Bluetooth clásico y Bluetooth de baja energía, lo que es útil para una amplia variedad de aplicaciones.
- **Doble núcleo:** La mayoría de los ESP32 son de doble núcleo, vienen con 2 microprocesadores Xtensa LX32 de 6 bits: núcleo 0 y núcleo 1.
- **Interfaz periférica enriquecida de entrada/salida:** El ESP32 admite una amplia variedad de entradas (leer datos) y de salida (para enviar comandos/señales) como táctil capacitiva, ADC, DAC, UART, SPI, I2C, PWM y mucho más.
- **Compatible con el "lenguaje de programación" Arduino:** Aquellos que ya están familiarizados con la programación de la placa Arduino, se puede programar el ESP32 en el estilo Arduino.

La placa LilyGO ESP32 T-Beam V1.1 con protocolo LoRa32 es una solución de desarrollo compacta y poderosa que combina el versátil microcontrolador ESP32 con la tecnología LoRa (Long Range). Diseñada para proyectos que requieren comunicación de largo alcance y bajo consumo de energía, esta placa ofrece una amplia gama de características y funcionalidades.

Equipada con un módulo LoRa SX1276, permite la comunicación inalámbrica en distancias considerables, lo que la hace ideal para aplicaciones como monitoreo ambiental, rastreo de activos, agricultura inteligente y más. Además, el módulo GPS incorporado brinda capacidades de geolocalización precisas, lo que añade un elemento adicional de funcionalidad a sus proyectos.

La LilyGO ESP32 T-Beam V1.1 es compatible con la plataforma de desarrollo Arduino, lo que facilita la programación y desarrollo de aplicaciones. Ya sea que esté interesado en implementar soluciones de monitoreo remoto, seguimiento en tiempo real u otros proyectos que requieran comunicación de largo alcance, esta placa ofrece las herramientas necesarias para llevar a cabo planes emocionantes y efectivos.

Características de la placa LilyGO ESP32 T-Beam V1.1 con Protocolo LoRa32:

Microcontrolador ESP32: Equipada con el microcontrolador ESP32 de Espressif, que incluye un procesador de doble núcleo de 32 bits, Wi-Fi y Bluetooth integrados.

Módulo LoRa SX1276: Permite la comunicación de largo alcance a través de la tecnología LoRa en las bandas ISM.

Módulo GPS: Ofrece capacidades de geolocalización precisas para rastreo y seguimiento.

Conexión para Batería: Dispone de un conector para batería recargable, ideal para aplicaciones de bajo consumo energético y móviles.

Conexión USB: Proporciona una interfaz USB para programación y alimentación de la placa.

Antena Incluida: Viene con una antena incorporada, lo que facilita la configuración y la comunicación de largo alcance.

Compatibilidad con LoRaWAN: Adecuada para aplicaciones que utilizan el protocolo LoRaWAN para comunicaciones de largo alcance y bajo consumo de energía.

Conexión UART: Proporciona una interfaz UART para la comunicación y programación de la placa.

2.3 ARDUINO SOFTWARE (Arduino, ¿Qué es Arduino?, s.f.)

Arduino es una plataforma electrónica de código abierto basada en hardware y software fáciles de usar. Puede indicarle a su placa qué hacer enviando un conjunto de instrucciones al microcontrolador de la placa. Para ello se utiliza el lenguaje de programación Arduino (basado en Wiring), y el Software Arduino (IDE), basado en Processing.

A lo largo de los años, Arduino ha sido el cerebro de miles de proyectos, desde objetos cotidianos hasta complejos instrumentos científicos. Una comunidad mundial de creadores (estudiantes, aficionados, artistas, programadores y profesionales) se ha reunido en torno a esta plataforma de código abierto, sus contribuciones se han sumado a una increíble cantidad de conocimientos accesibles que pueden ser de gran ayuda tanto para principiantes como para expertos.

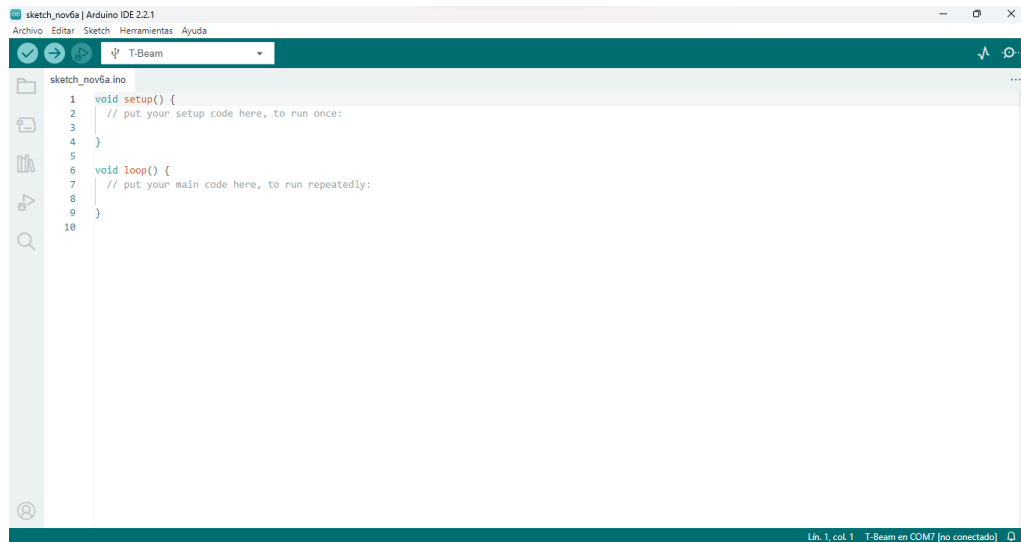
Arduino nació en el Instituto de Diseño de Interacción de Ivrea como una herramienta fácil para la creación rápida de prototipos, dirigida a estudiantes sin conocimientos de electrónica y programación. Tan pronto como llegó a una comunidad más amplia, la placa Arduino comenzó a cambiar para adaptarse a las nuevas necesidades y desafíos, diferenciando su oferta desde simples placas de 8 bits hasta productos para aplicaciones IoT, wearables, impresión 3D y entornos integrados.

¿Por qué Arduino?

- **Multiplataforma:** el software Arduino (IDE) se ejecuta en los sistemas operativos Windows, Macintosh, OSX y Linux. La mayoría de los sistemas de microcontroladores se limitan a Windows.
- **Entorno de programación simple y claro:** el software Arduino (IDE) es fácil de usar para principiantes, pero lo suficientemente flexible como para que los usuarios avanzados también lo aprovechen. Para los profesores, se basa convenientemente en el entorno de programación Processing, por lo que los estudiantes que aprendan a programar en ese entorno estarán familiarizados con el funcionamiento del IDE de Arduino.
- **Software de código abierto y extensible:** el software Arduino se publica como herramientas de código abierto, disponibles para su extensión por parte de programadores experimentados. El lenguaje se puede expandir a través de

bibliotecas C++, y las personas que quieran entender los detalles técnicos pueden dar el salto de Arduino al lenguaje de programación AVR C en el que se basa. Del mismo modo, puede agregar código AVR-C directamente a sus programas Arduino si lo desea.

Figura N° 17 Inicio del programa Arduino IDE



Fuente: Pantallazo

El propósito principal del Arduino IDE es simplificar el proceso de desarrollo de proyectos electrónicos, eliminando la necesidad de comprender complejas herramientas de programación y proporcionando un entorno accesible incluso para principiantes. Permite a los desarrolladores, estudiantes y entusiastas de la electrónica:

- Crear, editar y depurar código de manera eficiente.
- Utilizar bibliotecas predefinidas que simplifican la programación de hardware específico de Arduino.
- Compilar el código para verificar la ausencia de errores y preparar el programa para su carga en la placa.
- Cargar el programa en la placa, a través de un cable USB, lo que permite la interacción del hardware con el código.

El IDE incluye una amplia variedad de bibliotecas predefinidas que facilitan la programación de hardware específico de Arduino. Estas bibliotecas proporcionan

funciones y métodos listos para usar para controlar componentes comunes, como sensores, actuadores y pantallas, lo que ahorra tiempo y esfuerzo a los desarrolladores.

Además, ofrece herramientas básicas de depuración que ayudan a identificar y solucionar problemas en el código. Los mensajes de error y advertencia proporcionados durante la compilación pueden guiar a los programadores a corregir problemas en su código.

Una de las características distintivas es su capacidad para cargar programas en las placas a través de un cable USB de manera sencilla. Esto permite que los proyectos se implementen y prueben con facilidad en el hardware real.

Está disponible para varios sistemas operativos, incluyendo Windows, macOS y Linux, lo que lo hace accesible para una amplia audiencia. Los usuarios pueden trabajar en sus sistemas operativos preferidos sin restricciones.

El proceso de desarrollo con el Arduino IDE generalmente sigue estos pasos:

- **Abrir el Arduino IDE:** Inicie el Arduino IDE en su computadora. Dependiendo del sistema operativo, puede encontrar el programa en el menú de aplicaciones o en la barra de tareas.
- **Crear un Nuevo Proyecto:** Cree un nuevo proyecto seleccionando "Archivo" > "Nuevo" en el IDE. Esto abrirá una nueva ventana de código en blanco.
- **Escribir y Editar el Código:** En la ventana de código, escriba su programa utilizando el lenguaje de programación C/C++. El Arduino IDE proporciona sugerencias de autocompletado y resalta la sintaxis para facilitar la escritura del código.
- **Verificar Errores y Compilar el Programa:** Antes de cargar el programa en la placa Arduino, verifique que no haya errores en el código. Para hacerlo, haga clic en el botón de verificación (ícono de palomita) en la parte superior de la ventana. El IDE le notificará si encuentra algún error en su código.
- **Cargar el Programa en la Placa Arduino:** Conecte la placa Arduino a su computadora mediante un cable USB. Seleccione el tipo de placa y el puerto en el menú "Herramientas" del IDE. Haga clic en el botón de carga (ícono de flecha) en

la parte superior de la ventana. El IDE compilará su programa y lo cargará en la placa Arduino.

2.3 THINKSPEAK (Mathworks, s.f.)

ThingSpeak es un servicio de plataforma IoT (internet de las cosas) analytics que le permite agregar, visualizar y analizar flujos de datos en vivo en la nube. Puede enviar datos a ThingSpeak™ desde sus dispositivos, crear visualizaciones instantáneas de datos en vivo y enviar alertas utilizando servicios web como Twitter® y Twilio®. Con análisis MATLAB® dentro de ThingSpeak, puede escribir y ejecutar código MATLAB para realizar preprocesamiento, visualizaciones y análisis. ThingSpeak permite a los ingenieros y científicos crear prototipos y construir sistemas IoT sin configurar servidores ni desarrollar software web.

Las características principales de ThingSpeak incluyen:

- **Recopilación de datos:** ThingSpeak permite a los usuarios enviar datos desde sensores y dispositivos conectados a través de HTTP, MQTT u otros protocolos de comunicación.
- **Almacenamiento de datos:** Los datos enviados a ThingSpeak se almacenan en la nube, lo que facilita su acceso y gestión desde cualquier lugar con conexión a Internet.
- **Visualización de datos:** La plataforma ofrece herramientas para crear gráficos y visualizaciones de los datos recopilados, lo que facilita la interpretación de la información.
- **Análisis de datos:** ThingSpeak permite realizar análisis de datos en tiempo real y aplicar algoritmos personalizados para procesar la información recopilada.
- **Automatización:** Los usuarios pueden configurar reglas y acciones automatizadas en función de los datos recopilados, como el envío de notificaciones o el control de dispositivos.

ThingSpeak es una plataforma versátil que se utiliza en una amplia variedad de aplicaciones, como monitoreo del clima, seguimiento de la calidad del aire, control de dispositivos domésticos inteligentes, gestión de sistemas de riego, y mucho más.

2.4 CHAT GPT (Openai, 2024)

¿Cómo funciona ChatGPT?

ChatGPT está optimizado a partir de GPT-3.5, un modelo de lenguaje entrenado para producir texto. ChatGPT se optimizó para el diálogo mediante el uso de Reinforcement Learning with Human Feedback (RLHF), un método que utiliza demostraciones humanas y comparaciones de preferencias para guiar al modelo hacia el comportamiento deseado.

¿Por qué la IA parece tan real y realista?

Estos modelos se entrenaron con grandes cantidades de datos de Internet escritos por humanos, incluidas conversaciones, por lo que las respuestas que proporciona pueden sonar similares a las humanas. Es importante tener en cuenta que esto es un resultado directo del diseño del sistema (es decir, maximizar la similitud entre los resultados y el conjunto de datos con el que se entrenaron los modelos) y que dichos resultados pueden ser inexactos, falsos y, en ocasiones, engañosos.

¿Puedo confiar en que la IA me dice la verdad?

ChatGPT no está conectado a Internet y, en ocasiones, puede producir respuestas incorrectas. Tiene un conocimiento limitado del mundo y los eventos posteriores a 2021 y también puede producir ocasionalmente instrucciones dañinas o contenido sesgado.

Ejemplo:

Figura N° 18 Ejemplo de cómo funciona ChatGPT

The image shows a screenshot of the ChatGPT web interface. At the top, there are two model selection buttons: "GPT-3.5" (selected) and "GPT-4". Below this is the "ChatGPT" logo. There are four suggestion buttons: "Give me ideas about how to plan my New Years resolutions", "Come up with concepts for a retro-style arcade game", "Suggest some names for my cafe-by-day, bar-by-night business", and "Recommend a dish to bring to a potluck".

The main chat area shows a user message: "Hola chatgpt, me podrías realizar una pequeña programación de un led prendiendo y apagando cada 5 segundos". A green send button is to the right. Below the message is a disclaimer: "ChatGPT can make mistakes. Consider checking important information."

The AI's response starts with: "Claro, puedo proporcionarte un ejemplo simple de código para controlar un LED y hacer que se encienda y apague cada 5 segundos utilizando un lenguaje de programación como Arduino. Asumiré que tienes un LED conectado a un pin digital en tu placa Arduino. Aquí tienes un ejemplo de código en Arduino:"

The code is shown in a dark-themed editor with a "Copy code" button. The code is as follows:

```
cpp
// Definimos el número de pin al que está conectado el LED
const int ledPin = 13;

void setup() {
  // Configuramos el pin del LED como salida
  pinMode(ledPin, OUTPUT);
}

void loop() {
  // Encendemos el LED
  digitalWrite(ledPin, HIGH);
  delay(5000); // Esperamos 5 segundos (5000 milisegundos)

  // Apagamos el LED
  digitalWrite(ledPin, LOW);
  delay(5000); // Esperamos otros 5 segundos
}
```

Below the code is a "Regenerate" button and a "Send a message" input field. Another disclaimer is present: "ChatGPT can make mistakes. Consider checking important information."

The AI continues with an explanation: "En este código, primero se define el número de pin al que está conectado el LED (en este caso, el pin 13). En el método `setup()`, se configura ese pin como salida. Luego, en el método `loop()`, encendemos el LED, esperamos 5 segundos con `delay(5000)` y luego lo apagamos, nuevamente esperando 5 segundos. Asegúrate de tener una placa Arduino y un LED conectados correctamente antes de cargar este programa en tu Arduino. Puedes ajustar el número de pin o el tiempo de espera si es necesario."

At the bottom, there are buttons for "Was this response better or worse?", "Better", "Worse", "Same", and "Regenerate", along with another "Send a message" input field and a final disclaimer: "ChatGPT can make mistakes. Consider checking important information."

CAPÍTULO III

3 METODOLOGÍA

3.1 INTRODUCCIÓN

El propósito de este proyecto fue crear un sistema de monitorización que permitió recopilar datos en tiempo real sobre la concentración de material particulado en el ambiente. La placa LilyGO ESP32 T-Beam se encargó de la recopilación de datos y su posterior envío a la plataforma en la nube ThingSpeak, donde los datos se almacenaron y visualizaron de manera accesible. Este enfoque proporcionó una herramienta efectiva para la evaluación continua de la calidad del aire en una ubicación específica (domicilio).

A lo largo del informe, se detallaron los pasos necesarios para lograr con éxito la monitorización de material particulado, desde la configuración inicial hasta la programación de la placa y la posterior transferencia de datos a ThingSpeak. Además, se destacaron las pruebas, ajustes y desafíos encontrados durante el proceso, así como las conclusiones y recomendaciones resultantes de este esfuerzo.

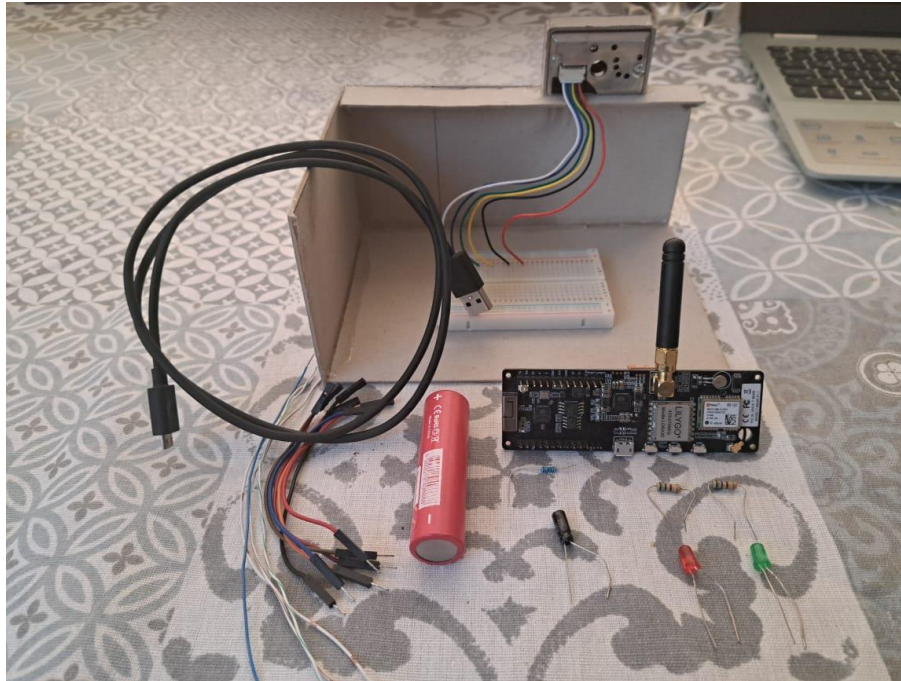
3.2 Configuración inicial

3.2.1 Recopilación de materiales necesarios

Se aseguró de tener todos los componentes necesarios para el proyecto. Los elementos claves fueron:

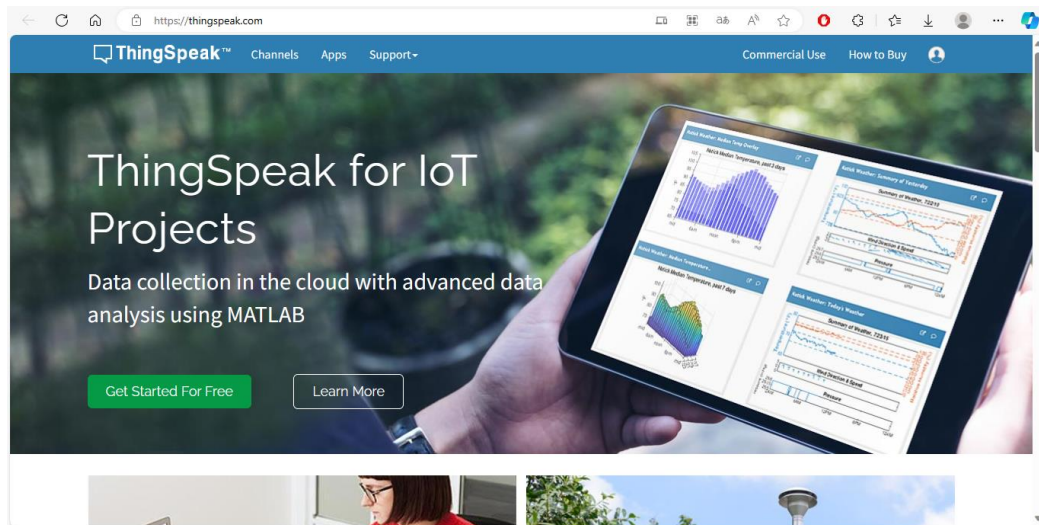
- Placa LilyGO ESP32 T-Beam V1.1.
- Sensor GP2Y1010AU0F-V3.
- Fuente de alimentación adecuada para la placa ESP32, en este caso, se optó por una batería recargable.
- Cables y conectores necesarios para conectar la placa y el sensor.
- Computadora con software de programación, como Arduino IDE.
- Red Wi-Fi y una cuenta en ThingSpeak.

Figura N° 19 Materiales para proyecto

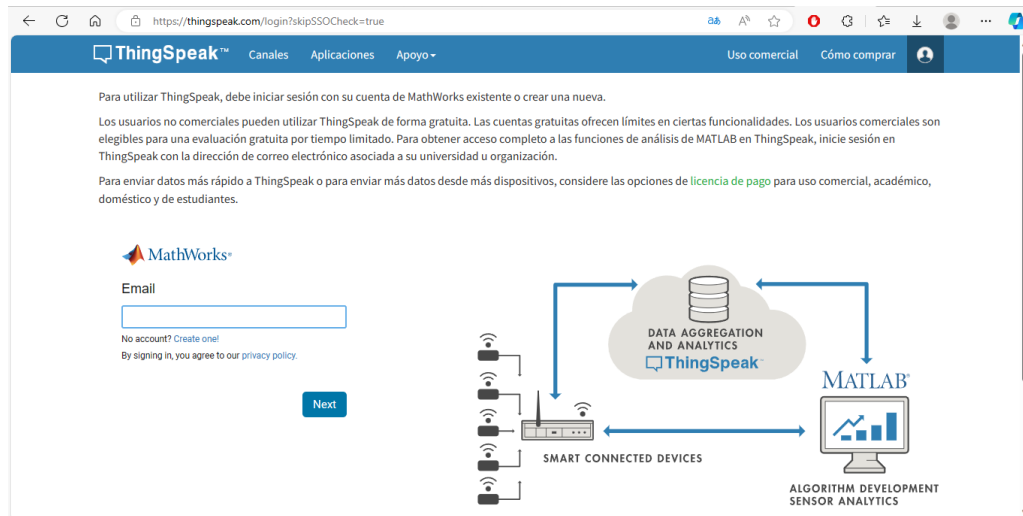


3.2.2 Creación de la cuenta THINKSPEAK:

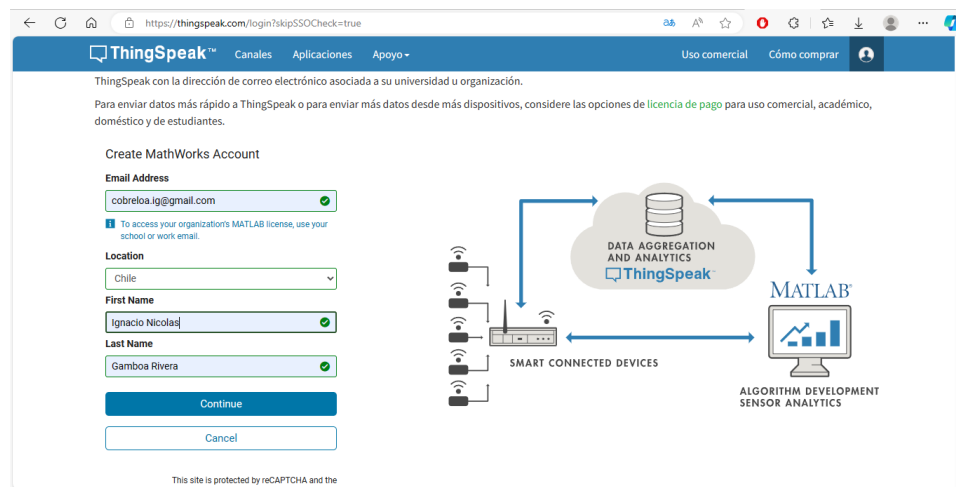
1.- Primero se ingresó a la página <https://thingspeak.com/>



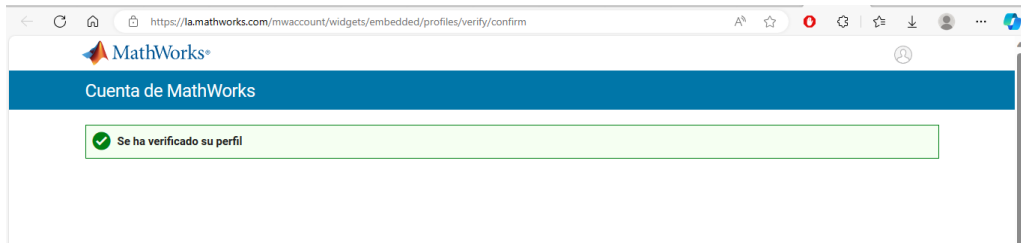
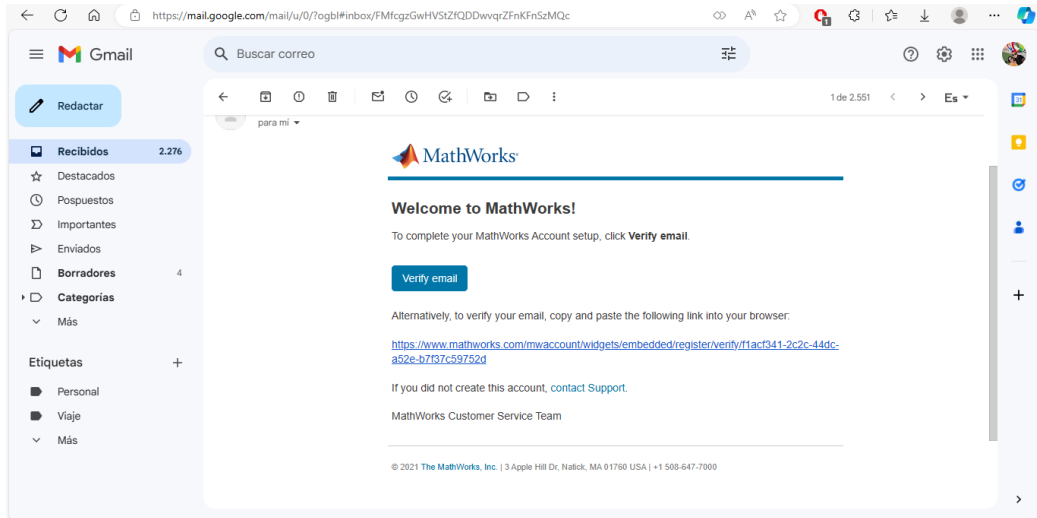
La página estaba en inglés, pero hoy en día estas páginas se pueden traducir de manera automática. En la esquina superior derecha me dirigí al icono de persona (al lado de “how to buy”) y apareció lo siguiente:



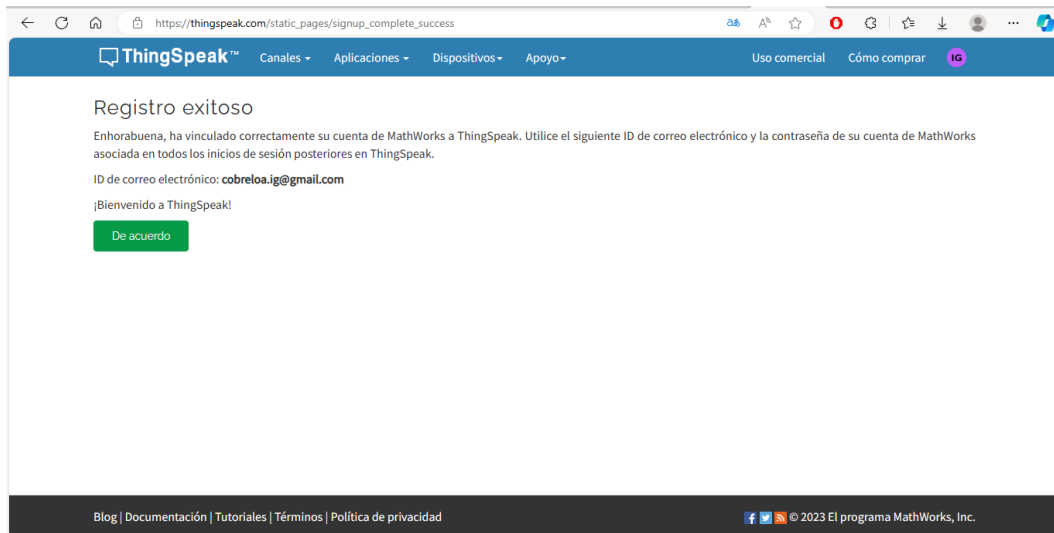
¡Para la creación de la cuenta se registró y se realizó apretando la opción “Create one!”.



Lo siguiente fue rellenar los datos y continuar con el proceso el cual después se consideró verificar la cuenta introduciendo el correo que se ingresó.



Luego se continuó y se creó la contraseña (pide caracteres especiales para una mayor seguridad) y se procedió a finalizar el proceso.



Con eso quedó la cuenta lista para la recopilación de datos que se explicará más adelante.

3.3 Conexión de los componentes

Figura N° 20 Diagrama de conexión del sensor y microcontrolador

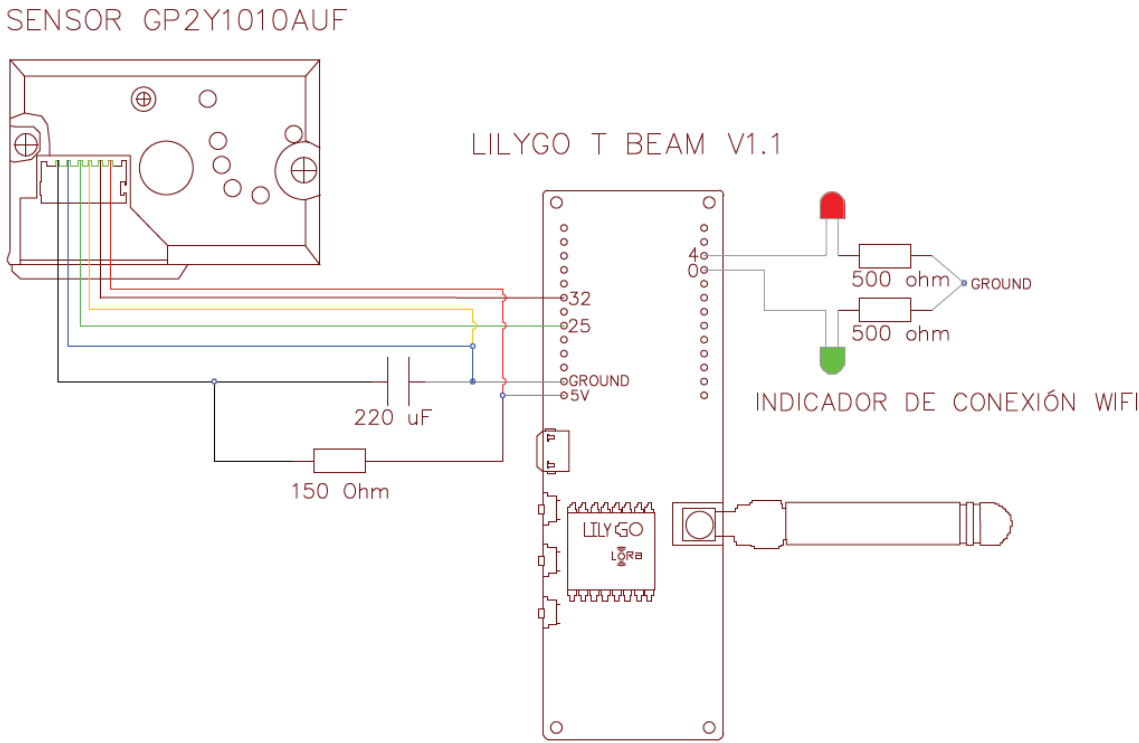
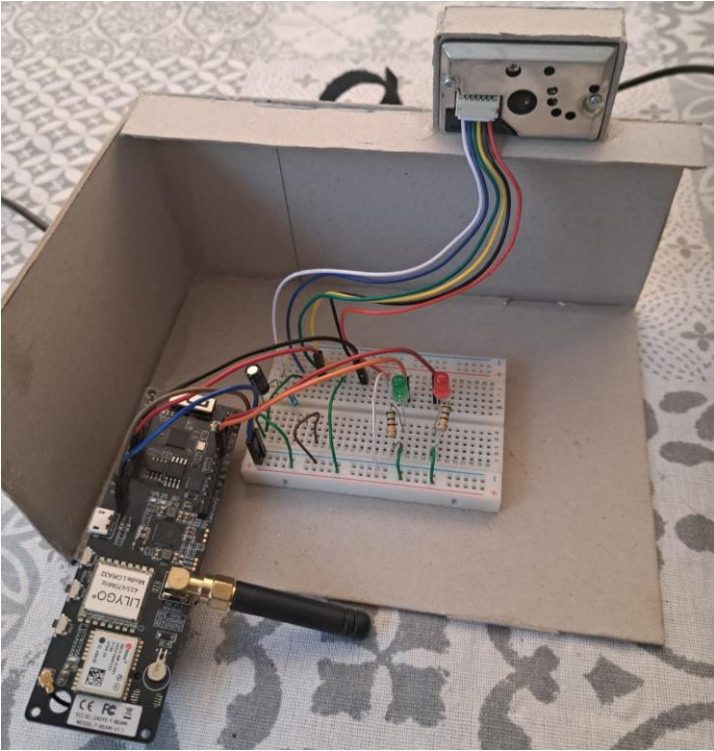


Figura N° 21 Conexionado físico



La resistencia, $R1=150\Omega$ y el condensador, $C1=220\mu F$ son necesarios para conducir la unidad de pulsos LED del GP2Y1010AU0F.

Para haber realizado esta conexión se asistió de la página <https://gidahatari.com/ih-es/tutorial-del-instalacion-y-puesta-en-marcha-de-un-sensor-de-polvo-en-arduino> en la cual se dio una guía de como conectar cada pin en la placa (en este caso del sensor a una protoboard y de la placa a esta misma).

3.4 Programación de la placa

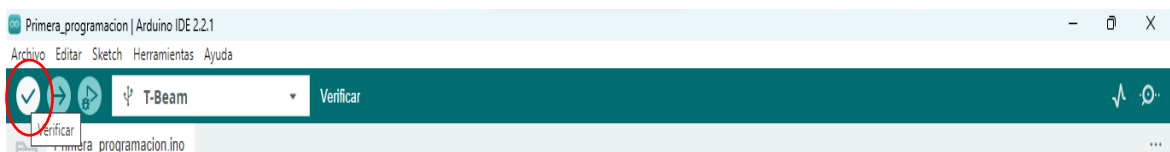
Ya para esta parte, se encaminó en la lógica que tenía la página anteriormente mencionada en la que me dio la guía de programar el sensor, mediante la aplicación **Arduino IDE** y que luego se esquematizó en la placa. La programación fue la siguiente:

Figura N° 22 Programación base

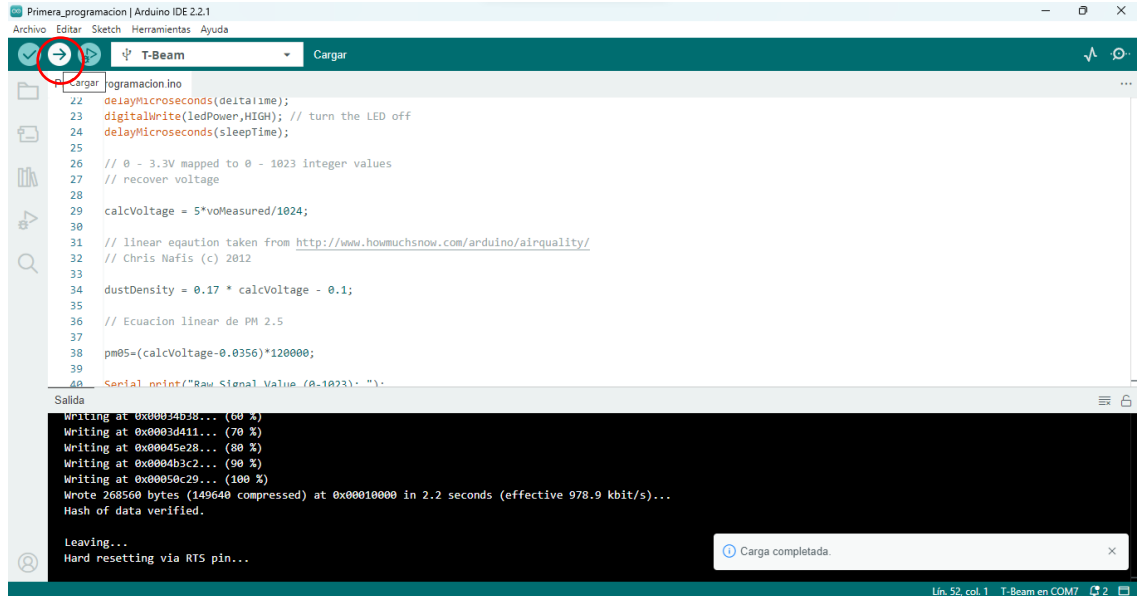


```
Primera_programacion | Arduino IDE 2.2.1
Archivo Editar Sketch Herramientas Ayuda
T-Beam
Primera_programacion.ino
1 int measurePin = 32;
2 int ledPower = 25;
3 int samplingTime = 280;
4 int deltaTime = 40;
5 int sleepTime = 9680;
6
7 float voMeasured = 0;
8 float calcVoltage = 0;
9 float dustDensity = 0;
10 float pm05=0;
11
12 void setup(){
13   Serial.begin(9600);
14   pinMode(ledPower,OUTPUT);
15 }
16
17 void loop(){
18   digitalWrite(ledPower,LOW); // power on the LED
19   delayMicroseconds(samplingTime);
20   voMeasured = analogRead(measurePin); // read the dust value
21
22   delayMicroseconds(deltaTime);
23   digitalWrite(ledPower,HIGH); // turn the LED off
24   delayMicroseconds(sleepTime);
25
26   // 0 - 3.3V mapped to 0 - 1023 integer values
27   // recover voltage
28
29
30   --
31   calcVoltage = 5*voMeasured/1024;
32
33   // linear equation taken from http://www.howmuchsnow.com/arduino/airquality/
34   // Chris Nafis (c) 2012
35
36   dustDensity = 0.17 * calcVoltage - 0.1;
37
38   // Ecuacion linear de PM 2.5
39   |
40   pm05=(calcVoltage-0.0356)*120000;
41
42   Serial.print("Raw Signal Value (0-1023): ");
43   Serial.print(voMeasured);
44   Serial.print(" - Voltage: ");
45   Serial.print(calcVoltage);
46   Serial.print(" - Dust Density(mg/m3): ");
47   Serial.println(dustDensity);
48   Serial.print(" - PM 0.5(particulas/0.01 pie3): ");
49   Serial.println(pm05);
50   delay(1000);
51
52 }
```

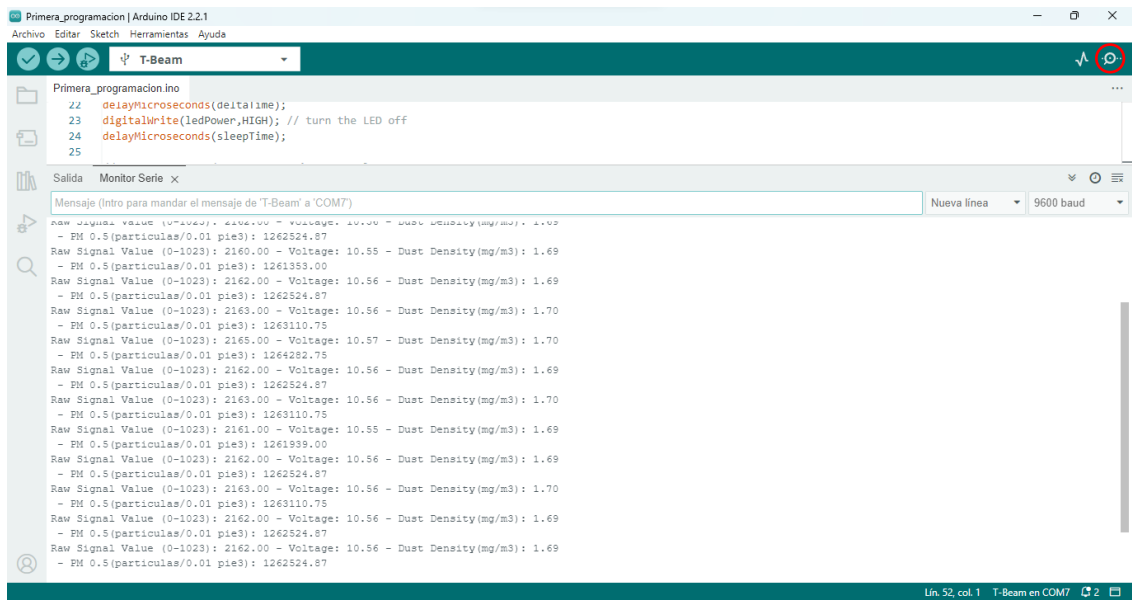
Para verificar que esta programación estuviera de forma correcta se debía de verificar en el icono de “tick” que se encuentra en la parte superior izquierda.



Una vez chequeada la programación se procedió a cargar el código para la placa.



Una vez que cargó la programación, los resultados se podían ver en el monitor serial que se encuentra en la parte superior derecha en la que su icono es una “lupa”.



Primera programación en Arduino IDE

Ya que se poseía la base de la programación, faltaba conectar la placa al Wifi y que, de esta manera, se pudiera enviar los datos a Thinkspeak, por lo cual se reforzó de la inteligencia artificial **Chat GPT**, para desarrollar el código del proyecto. Para ello, se

transcribió la misma programación y se solicitó que dejara tal cual el código, y que agregara las funciones de wifi (petición especial en la que se pudiera conectar la placa Wifi mediante la función autoconnect), y que estos puedan llegar a la página de Thingspeak para recopilar los datos y mostrar en forma de gráfico.

EL resultado que me generó Chat GPT se muestra en las siguientes imágenes:

Figura N° 23 Programación final con ayuda de ChatGPT



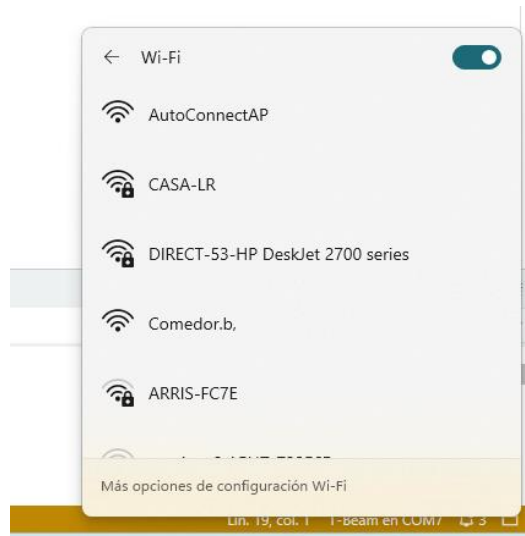
```
Programa_final | Arduino IDE 2.2.1
Archivo Editar Sketch Herramientas Ayuda
T-Beam
Programa_final.ino
1 #include <WiFiManager.h>
2 #include <ThingSpeak.h>
3
4 const char* server = "api.thingspeak.com";
5 const unsigned long channelID = 2334847; // Reemplaza TU_ID_CANAL con tu ID de canal ThingSpeak
6 const char* apiKey = "WEM7PH00TAG1E45R"; // Reemplaza TU_API_KEY con tu clave de API de ThingSpeak
7
8 WiFiClient client;
9
10 int measurePin = 32;
11 int ledPower = 25;
12 int samplingTime = 280;
13 int deltaTime = 40;
14 int sleepTime = 9680;
15 float voMeasured = 0;
16 float calcVoltage = 0;
17 float dustDensity = 0;
18 float pm05 = 0;
19
20 unsigned long previousSerialMillis = 0;
21 const unsigned long serialInterval = 5000; // Intervalo de muestra en el monitor serial (5 segundos)
22
23 unsigned long previousThingSpeakMillis = 0;
24 const unsigned long thingSpeakInterval = 30000; // Intervalo de envío a ThingSpeak (30 segundos)
25
26 const int redLed = 4; // Pin del LED rojo
27 const int greenLed = 0; // Pin del LED verde
28
29 unsigned long previousWiFiRetryMillis = 0;
30 const unsigned long WiFiRetryInterval = 30000; // Intentar reconexión cada 30 segundos
31
32 void setup() {
33   Serial.begin(9600);
34   pinMode(ledPower, OUTPUT);
35   pinMode(redLed, OUTPUT); // Configura el pin del LED rojo como salida
36   pinMode(greenLed, OUTPUT); // Configura el pin del LED verde como salida
37
38   // Inicializar WiFiManager
39   WiFiManager wifiManager;
40   wifiManager.autoConnect("AutoConnectAP"); // Crea un punto de acceso WiFi para configuración
41
42   ThingSpeak.begin(client); // Inicializar ThingSpeak
43 }
44
45 void loop() {
46   unsigned long currentMillis = millis();
47
48   digitalWrite(ledPower, LOW);
49   delayMicroseconds(samplingTime);
50   voMeasured = analogRead(measurePin);
51   delayMicroseconds(deltaTime);
52   digitalWrite(ledPower, HIGH);
53   delayMicroseconds(sleepTime);
54
55   calcVoltage = 5 * voMeasured / 1024;
```

```
56 dustDensity = 0.17 * calcVoltage - 0.1;
57 pm05 = (calcVoltage - 0.0356) * 120000;
58
59 if (currentMillis - previousSerialMillis >= serialInterval) {
60     previousSerialMillis = currentMillis;
61     Serial.print("Raw Signal Value (0-1023): ");
62     Serial.print(voMeasured);
63     Serial.print(" - Voltage: ");
64     Serial.print(calcVoltage);
65     Serial.print(" - Dust Density(mg/m3): ");
66     if (dustDensity < 0) {
67         Serial.println(0); // Si el valor es negativo, muestra 0 en su lugar
68     } else {
69         Serial.println(dustDensity);
70     }
71     Serial.print(" - PM 0.5(particulas/0.01 pie3): ");
72     Serial.println(pm05);
73 }
74
75 // Verifica el estado de la conexión WiFi y controla los LEDs
76 if (WiFi.status() != WL_CONNECTED) {
77     // WiFi desconectado, enciende el LED rojo y apaga el LED verde
78     digitalWrite(redLed, HIGH);
79     digitalWrite(greenLed, LOW);
80
81     // Intenta reconectar cada cierto tiempo (por ejemplo, cada 30 segundos)
82     if (millis() - previousWiFiRetryMillis >= WiFiRetryInterval) {
83
84         previousWiFiRetryMillis = millis();
85         WiFi.disconnect();
86         WiFi.begin(); // Inicia el proceso de conexión WiFi sin especificar SSID ni contraseña
87     } else {
88         // WiFi conectado, enciende el LED verde y apaga el LED rojo
89
90         digitalWrite(greenLed, HIGH);
91         digitalWrite(redLed, LOW);
92
93         // Envía datos a ThingSpeak
94         if (currentMillis - previousThingSpeakMillis >= thingSpeakInterval) {
95             previousThingSpeakMillis = currentMillis;
96             ThingSpeak.setField(1, dustDensity); // Configura el valor del campo 1
97             ThingSpeak.setField(2, pm05); // Configura el valor del campo 2
98             int updateStatus = ThingSpeak.writeFields(channelID, apiKey); // Enviar datos a ThingSpeak
99
100             if (updateStatus == 200) {
101                 Serial.println("Data sent to ThingSpeak");
102             } else {
103                 Serial.println("Error sending data to ThingSpeak");
104             }
105         }
106     }
107 }
```

Monitor Serie x Salida

Mensaje (Intro para mandar el mensaje de 'T-Beam' a 'COM7')

Para conectar la placa al wifi se debía buscar en el computador, la selección de Wifi y escoger la que tiene nombre “AutoconnectAp”.



Se conectó al Wifi y se dispuso la página la cual se requerían los siguientes pasos:

Se Seleccionaba configurar Wifi para luego ver que red se conseguía conectar.

WiFiManager

AutoConnectAP

Configure WiFi

Info

Exit

Update

Not connected to VTR-0791115
AP not found

Elegida la red se colocaba la contraseña y se oprimía el botón “Save” y quedó conectado a la red Wifi.

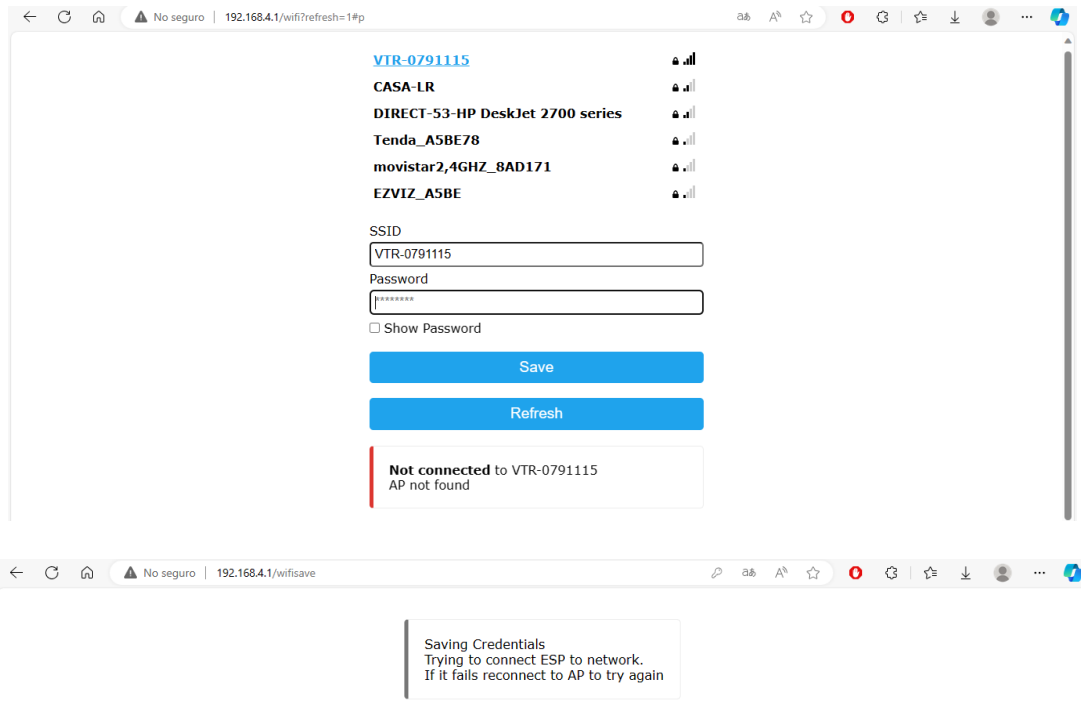


Figura N° 24 Placa sin ninguna red Wifi

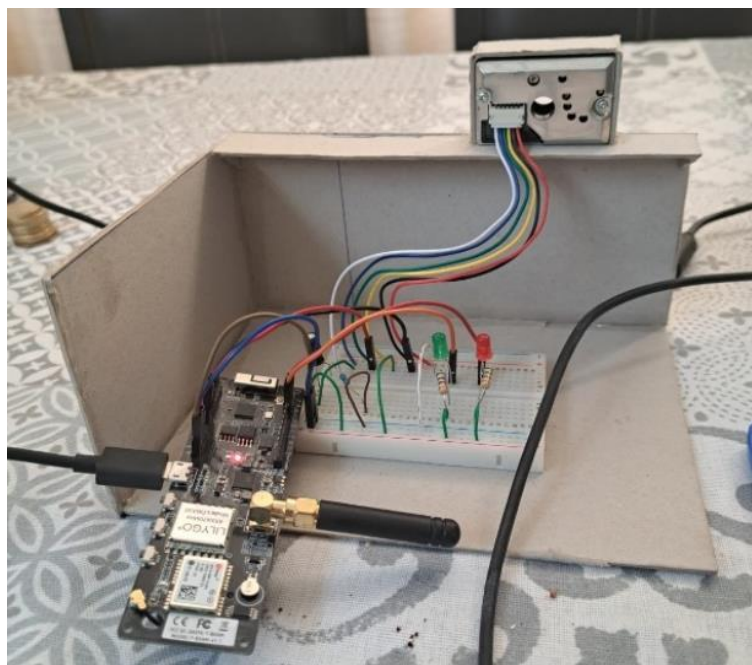


Figura N° 25 Placa conectada a red Wifi

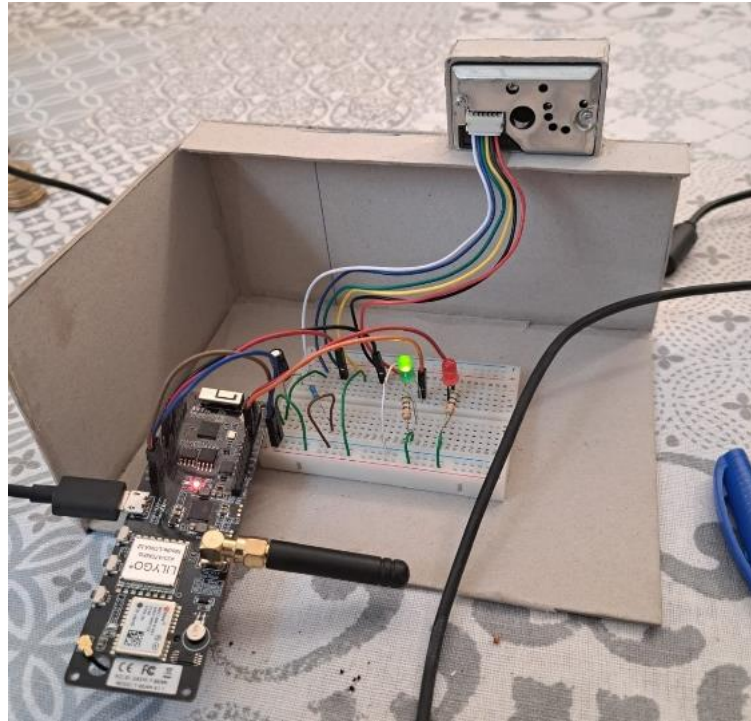
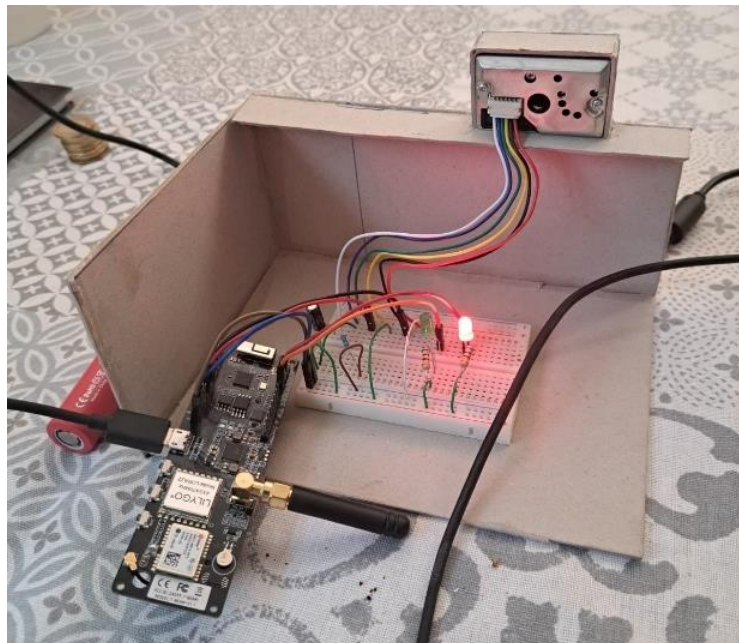
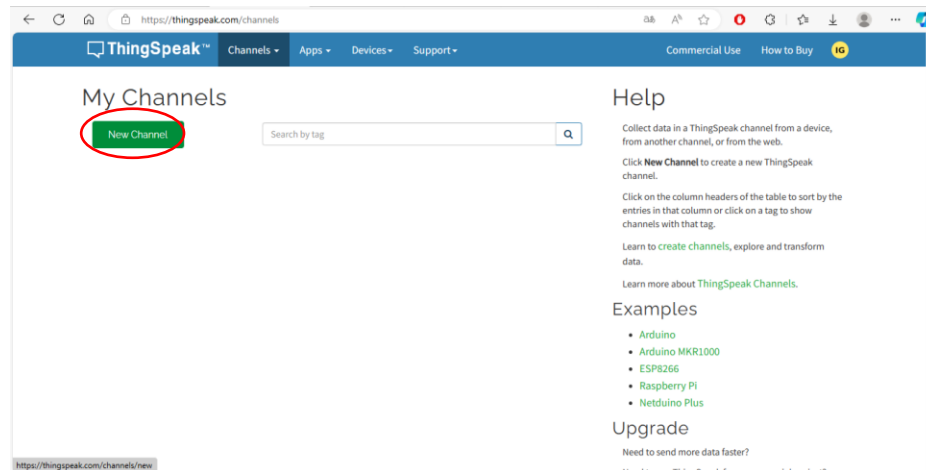
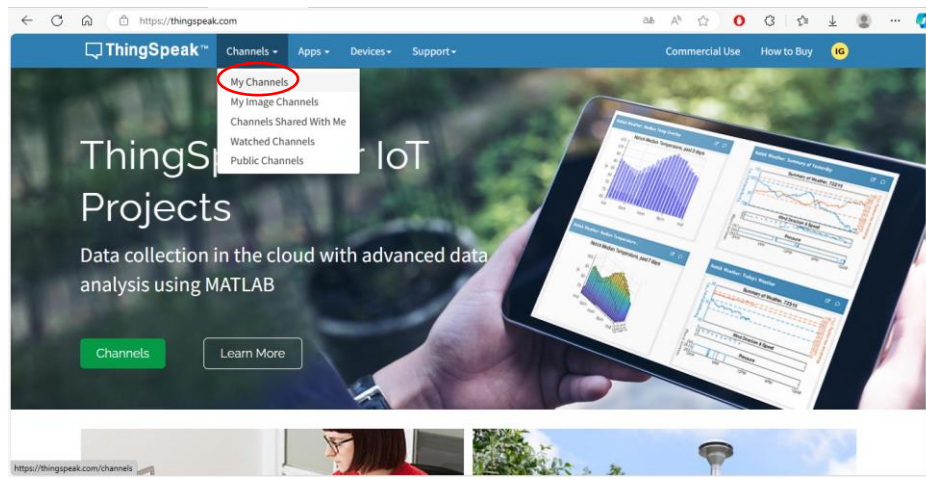


Figura N° 26 Placa cuando red Wifi esta desconectada

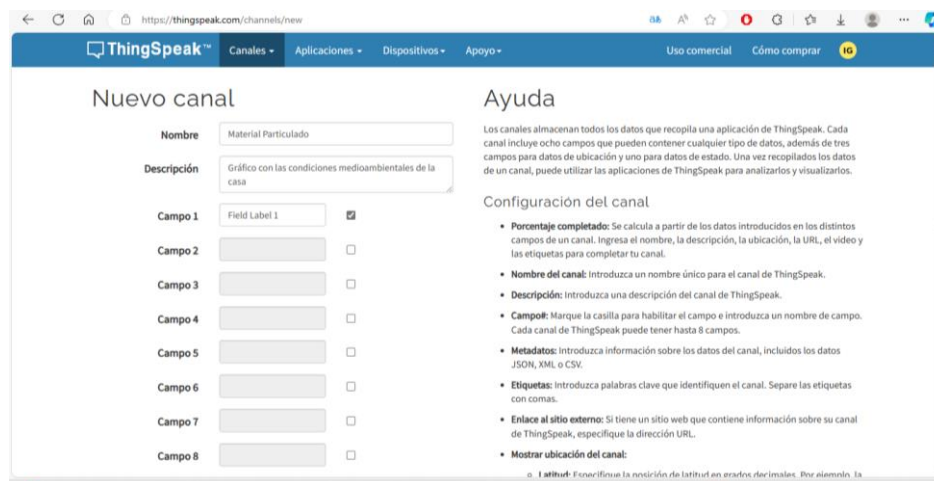


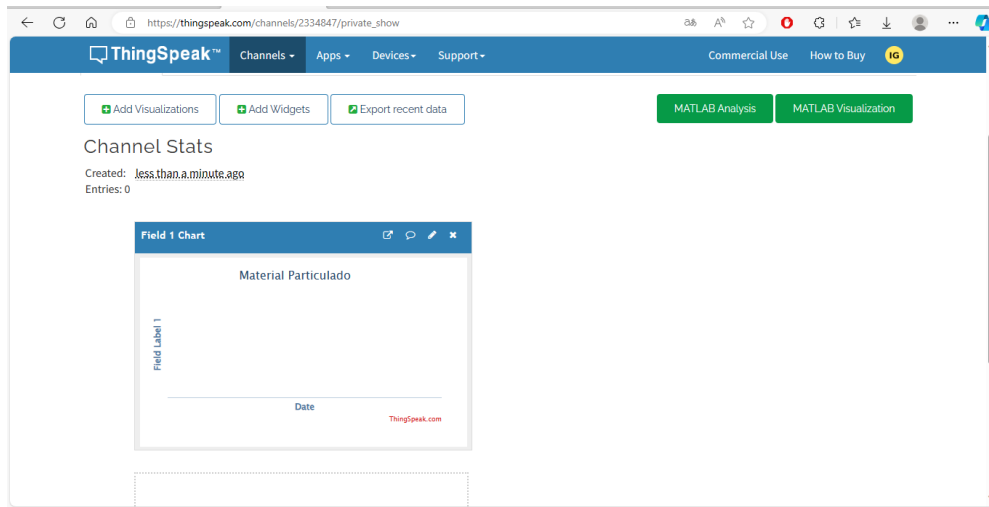
Para enviar los datos de la placa a la página Thinkspeak, se consideraba saber el ID del canal en la que se hizo el gráfico y la clave, que se podía generar de manera aleatoria. Esto se hizo de la siguiente forma:

Lo primero se iniciaba sesión (la cual se registró como se muestra en capítulos anteriores) para posteriormente crear un canal.

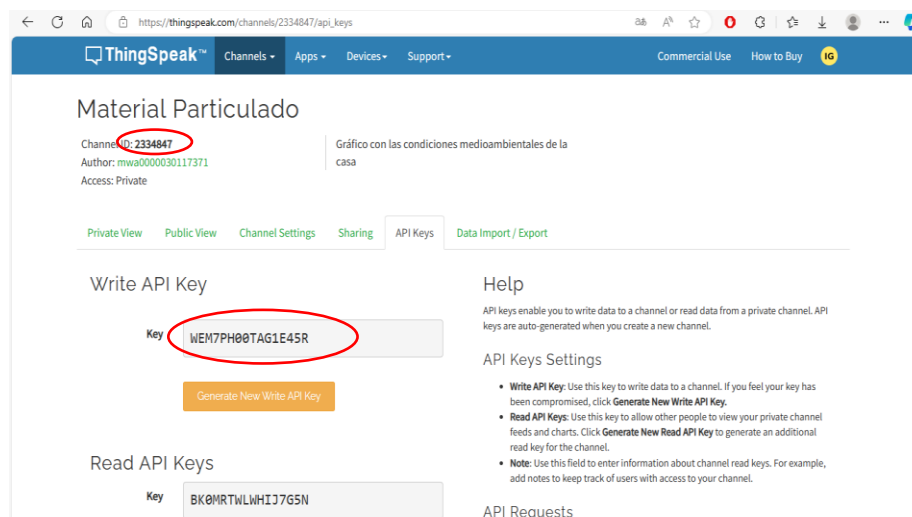


Se hizo el registro de los datos para generar el gráfico.





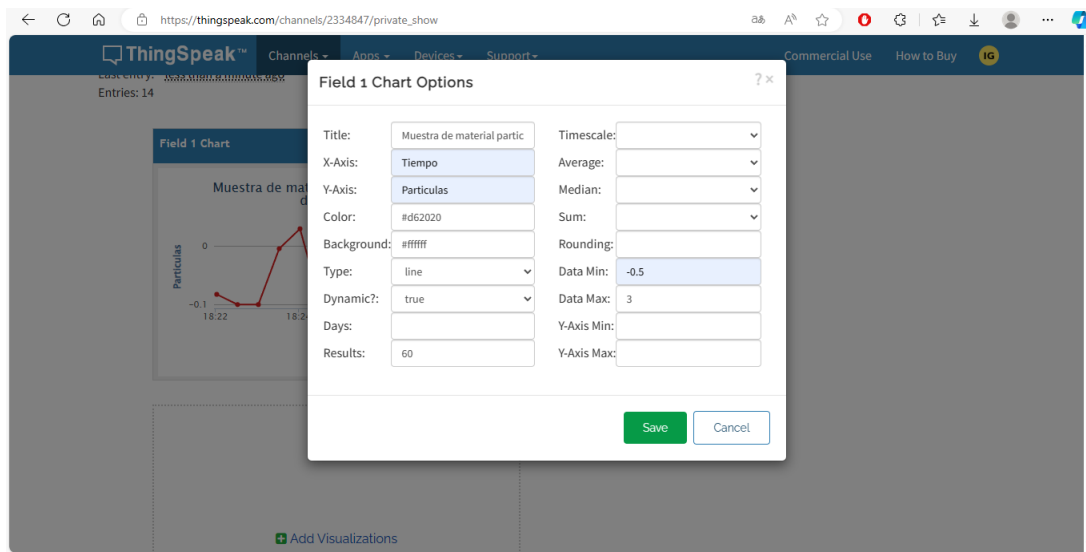
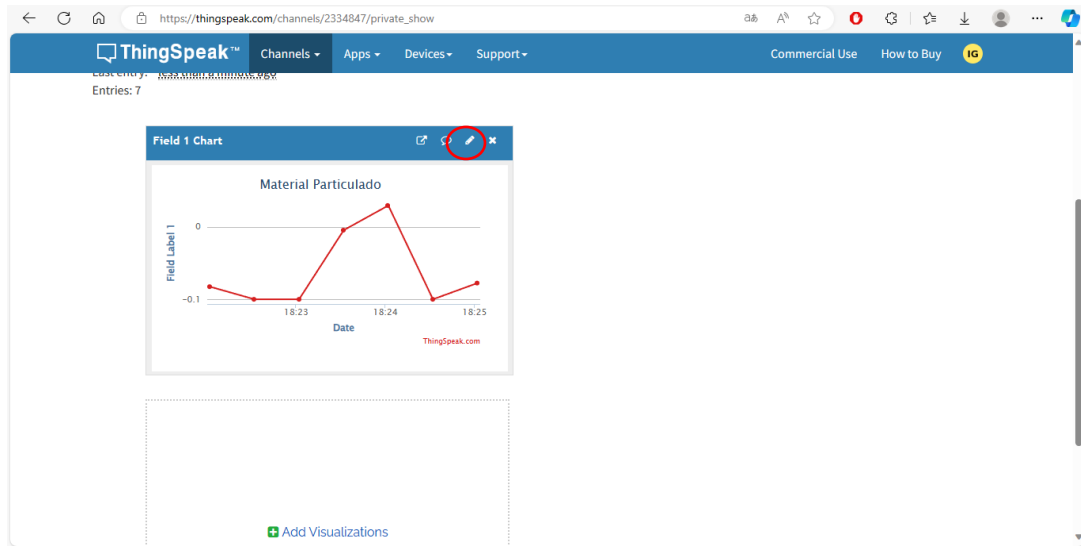
Ya con el gráfico hecho solo faltó colocar los datos del canal y dejarlos por escrito en la programación.



Estos son los datos que se colocaron en la programación:

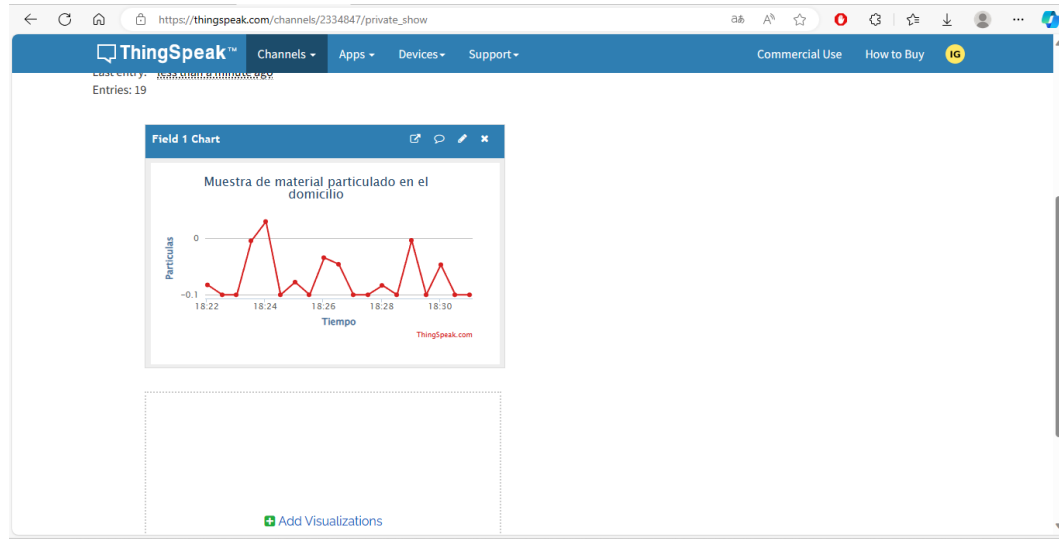
```
Programa_final | Arduino IDE 2.2.1
Archivo Editar Sketch Herramientas Ayuda
T-Beam
Programa_final.ino
1 #include <WiFiManager.h>
2 #include <ThingSpeak.h>
3
4 const char* server = "api.thingspeak.com";
5 const unsigned long channelID = 2334847; // Reemplaza TU_ID_CANAL con tu ID de canal ThingSpeak
6 const char* apiKey = "WEM7PH00TAG1E45R"; // Reemplaza TU_API_KEY con tu clave de API de ThingSpeak
7
8 WiFiClient client;
9
10 int measurePin = 32;
11
```

Por último, se realizó las modificaciones al gráfico para que se vieran los datos que se estaban recopilando (que en este caso fue el material particulado (Eje Y) y el tiempo (Eje X)). Esto se hizo en el gráfico, en la parte superior, se debió presionar el icono de lápiz:

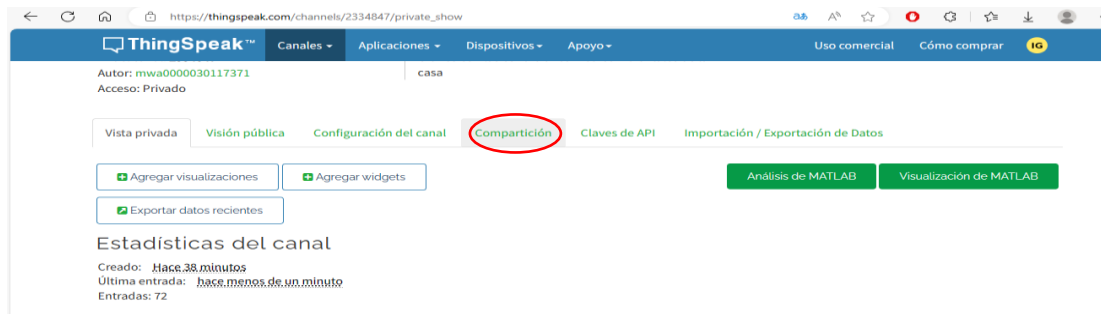


Y el gráfico quedó de la siguiente manera:

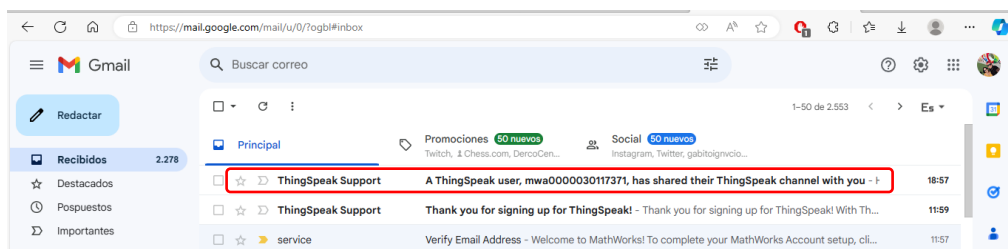
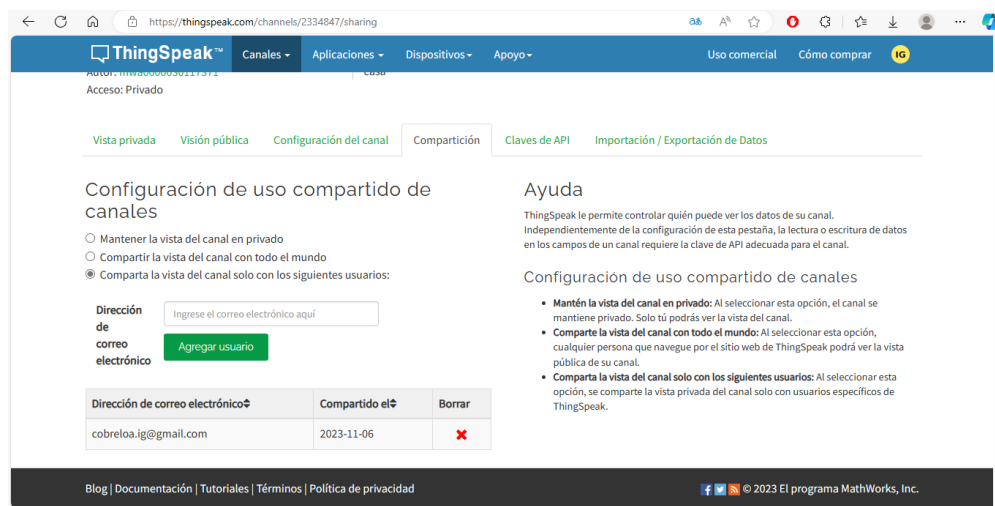
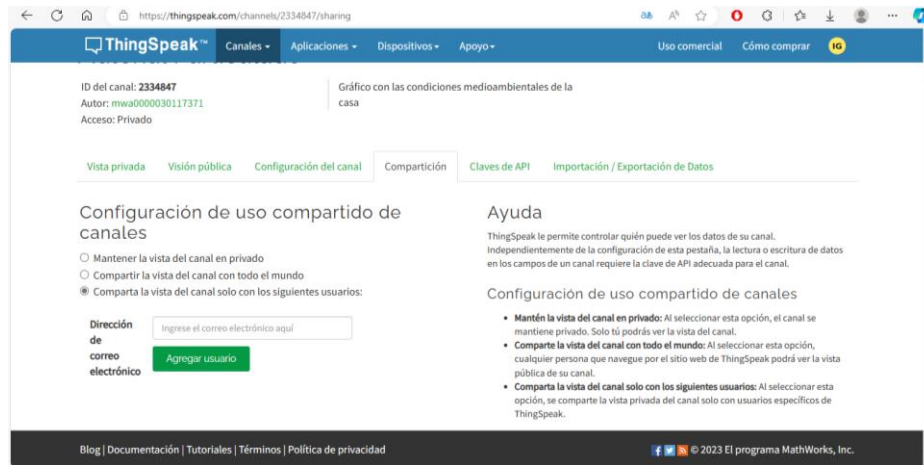
Figura N° 27 Gráfico final de Thingspeak



Para compartir este grafico con otras personas que pretendían saber el estado del domicilio con respecto a si está contaminado o no, se concibió de la siguiente manera:



La manera de compartir el gráfico en esta página, solo se pudo realizar mediante correo y no por otras apps como por ejemplo Whatsapp. Lo otro es que a la persona que se le envió el gráfico, debía de tener también una cuenta de Thingspeak.



Esto dio a entender que la persona, ya obtenía la opción de ver el gráfico, por lo que debía de iniciar sesión y realizar los siguientes pasos:

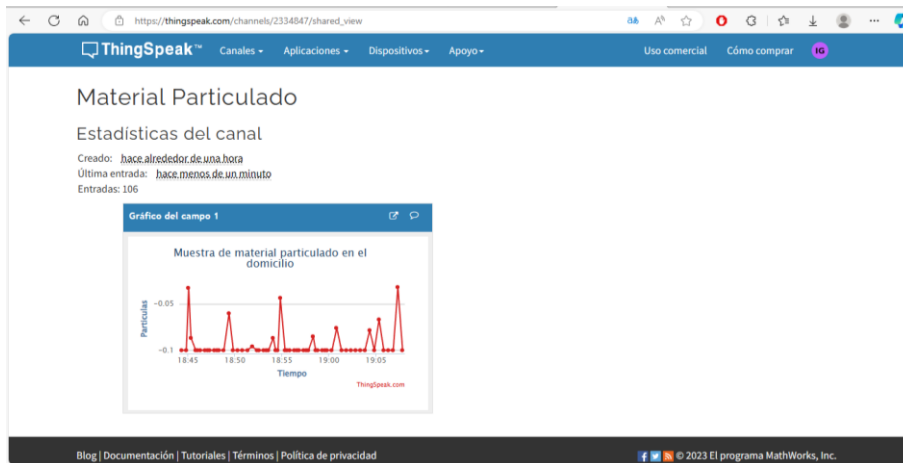
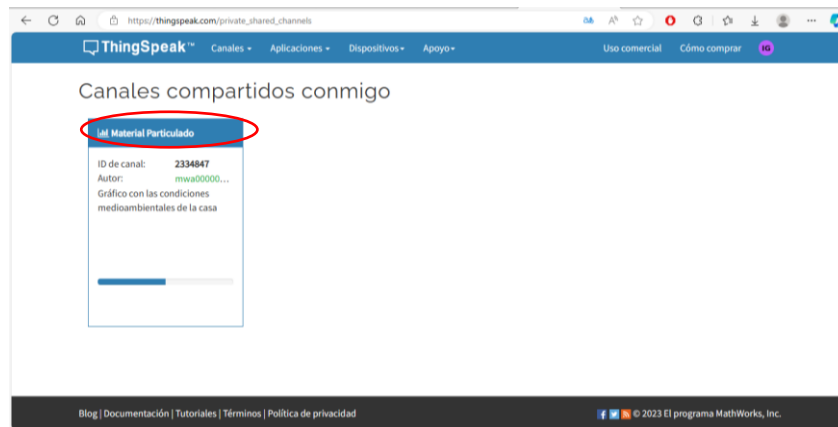
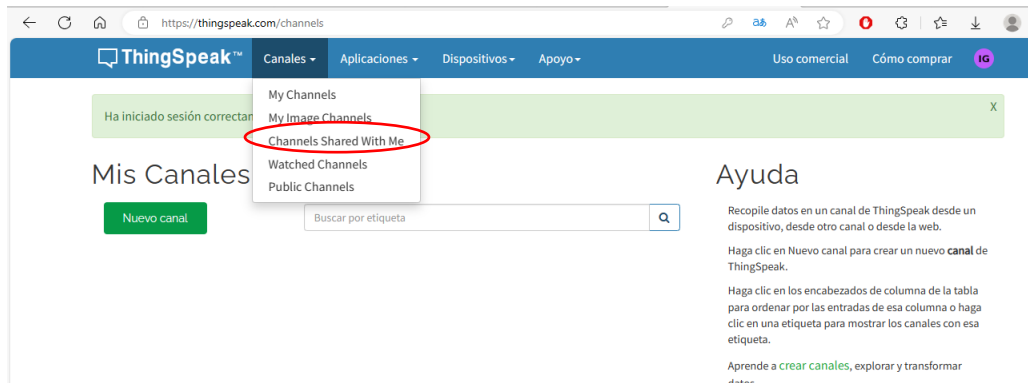


Gráfico compartido

La gran ventaja además que se otorgó con esta plataforma fue el ver estos datos en lugares lejanos como por ejemplo en la siguiente imagen:

Figura N° 28 Referencia de localización en celular



En la imagen se vio que me encontraba fuera de la casa (Mi casa se representa con el icono corazón) y que igual pude ver los datos que se recopilaron en la placa a pesar de la distancia. Además el gráfico indica que no hay mayor cantidad de material particulado, por lo que en casa no hay problema de contaminación.

CAPÍTULO IV

4 MEJORAS

En el transcurso del proyecto, se logró exitosamente la implementación del sistema de recolección. Los avances permitieron mediciones, que inciden en la condición del aire, desde la concentración de contaminantes.

Sin embargo, un aspecto esencial que se identificó durante el desarrollo fue la ausencia de un mecanismo efectivo para alertar sobre condiciones críticas de contaminación atmosférica. A pesar de contar con las evaluaciones, carecía de una función que notificara

de manera inmediata y audible la presencia de altos niveles de contaminación, limitando la utilidad práctica del sistema.

Esta carencia fue reconocida como una oportunidad para mejorar y elevar la eficacia del proyecto. La necesidad de una alerta que complementara las mediciones existentes se volvió evidente para garantizar no solo la recopilación de datos, sino también la capacidad de informar y prevenir a los usuarios sobre condiciones adversas en la calidad del aire.

Además, para reforzar la autonomía del sistema, se implementó una mejora adicional mediante la integración de un circuito auxiliar alimentado por un panel solar. Este circuito garantiza una fuente de energía sostenible y continua, lo que mantiene la placa energizada independientemente de las condiciones climáticas y aumenta la confiabilidad y disponibilidad del sistema.

4.1 Adición de Alerta Acústica para Condiciones de contaminación:

Una mejora significativa que se integró al sistema de medición de calidad de aire fue la inclusión de una alerta acústica. Esta mejora se implementó para proporcionar una notificación inmediata cuando los niveles de contaminación atmosférica alcanzan un umbral predeterminado. Al no poder tener un mecanismo de referencia con respecto a cuanto debía ser el umbral para que notificara si el ambiente está contaminado, dejé el valor máximo, el cual el sensor valor máximo mide 10 micrómetros. La adición de una bocina o dispositivo de alerta sonora permitió una rápida identificación de condiciones de aire denso o altamente contaminado, ofreciendo una solución proactiva para alertar a los usuarios o a las personas cercanas al entorno de medición.

Detalles de la Implementación:

Funcionalidad: La alerta se activa cuando los sensores detectan niveles de contaminación atmosférica que superan los límites establecidos como seguros.

Interfaz con el Sistema Principal: La bocina se encuentra conectada al microcontrolador principal y se activa mediante un protocolo predefinido en el código, permitiendo una sincronización efectiva con los datos recopilados por el sistema de medición.

Beneficios: Proporcionó una respuesta inmediata y audible ante condiciones ambientales adversas, alertando a las personas cercanas para que tomen las medidas necesarias.

Impacto en la Aplicación Práctica: Esta adición mejoró significativamente la utilidad del sistema, convirtiéndolo en una herramienta más completa y proactiva para la detección de condiciones de calidad de aire adversas.

Buzzer (Wikipedia, Zumbador, s.f.): Un zumbador (en inglés buzzer) es un transductor electroacústico que produce un sonido o zumbido continuo o intermitente de un mismo tono (generalmente agudo). Sirve como mecanismo de señalización o aviso y se utiliza en múltiples sistemas, como en automóviles o en electrodomésticos, incluidos los despertadores.

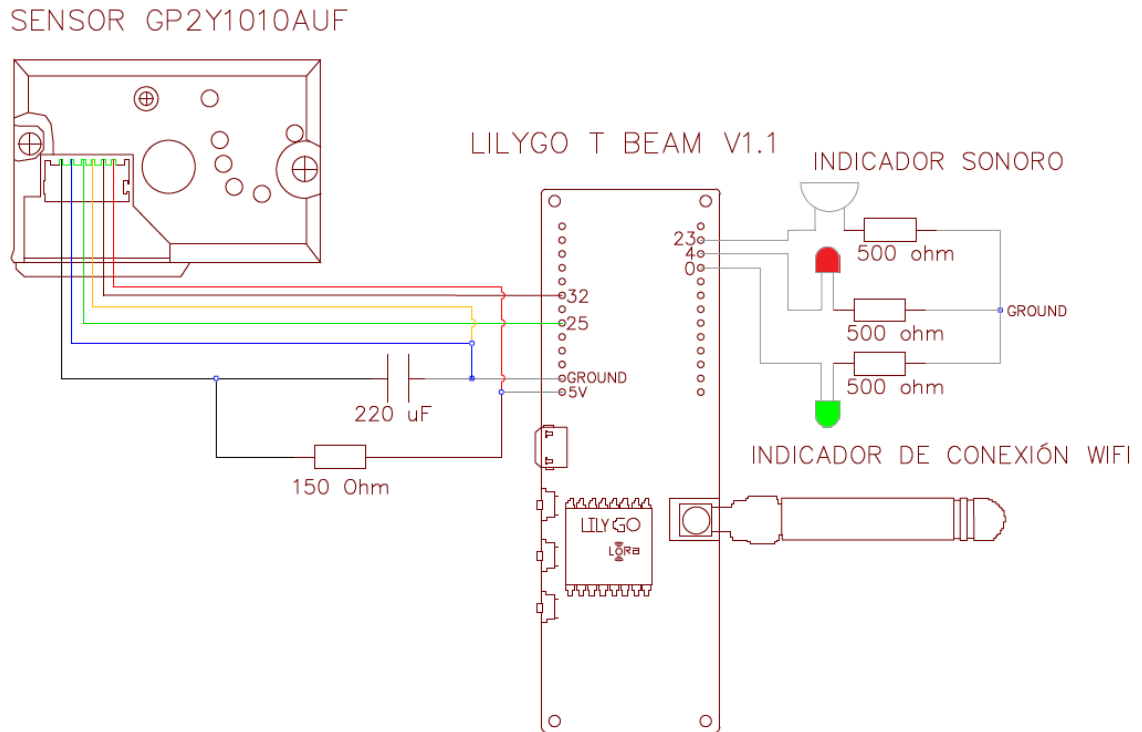
Figura N° 29 Buzzer



Fuente: Google

Lo siguiente fue que se adicionó este elemento dentro del circuito:

Figura N° 30 Diagrama de conexión del sensor y microcontrolador con Buzzer



Lo siguiente fue agregar a la programación el buzzer y con ayuda de chatGPT agregar la condición de si estaba dentro de parámetros altos, notificar mediante el indicador sonoro que hay contaminación en el ambiente siempre que este dentro de un rango de tiempo (pasado 5 segundos).

```
Programa_final.ino
1  #include <WiFiManager.h>
2  #include <ThingSpeak.h>
3
4  const char* server = "api.thingspeak.com";
5  const unsigned long channelID = 2334847; // Reemplaza TU_ID_CANAL con tu ID de canal ThingSpeak
6  const char* apiKey = "WEM7PH00TAG1E45R"; // Reemplaza TU_API_KEY con tu clave de API de ThingSpeak
7
8  WiFiClient client;
9
10 int measurePin = 32;
11 int ledPower = 25;
12 int samplingTime = 280;
13 int deltaTime = 40;
14 int sleepTime = 9680;
15 float voMeasured = 0;
16 float calcVoltage = 0;
17 float dustDensity = 0;
18 float pm05 = 0;
19
20 unsigned long previousSerialMillis = 0;
21 const unsigned long serialInterval = 15000; // Intervalo de muestra en el monitor serial (15 segundos)
22
23 unsigned long previousThingSpeakMillis = 0;
24 const unsigned long thingSpeakInterval = 15000; // Intervalo de envío a ThingSpeak (15 segundos)
25
26 const int redLed = 4; // Pin del LED rojo
27 const int greenLed = 0; // Pin del LED verde
28 const int buzzerPin = 23; // Pin del buzzer
29
30 unsigned long highDensityStartTime = 0; // Variable para almacenar el tiempo en el que la densidad supera el umbral
31 bool buzzerSounding = false; // Variable para controlar si el buzzer está sonando
32
33 unsigned long previousWiFiRetryMillis = 0;
34 const unsigned long WiFiRetryInterval = 30000; // Intentar reconexión cada 30 segundos
35
36 void setup() {
37   Serial.begin(9600);
38   pinMode(ledPower, OUTPUT);
39   pinMode(redLed, OUTPUT); // Configura el pin del LED rojo como salida
40   pinMode(greenLed, OUTPUT); // Configura el pin del LED verde como salida
41   pinMode(buzzerPin, OUTPUT); // Configura el pin del buzzer como salida
42
43   // Inicializar WiFiManager
44   WiFiManager wifiManager;
45   wifiManager.autoConnect("AutoConnectAP"); // Crea un punto de acceso WiFi para configuración
46
47   ThingSpeak.begin(client); // Inicializar ThingSpeak
48 }
49
50 void loop() {
51   unsigned long currentMillis = millis();
52
53   digitalWrite(ledPower, LOW);
54   delayMicroseconds(samplingTime);
55   voMeasured = analogRead(measurePin);
56   delayMicroseconds(deltaTime);
```

```

57 digitalWrite(ledPower, HIGH);
58 delayMicroseconds(sleepTime);
59
60 calcVoltage = 5 * voMeasured / 1024;
61 dustDensity = 0.17 * calcVoltage - 0.1;
62 pm05 = (calcVoltage - 0.0356) * 120000;
63
64 if (dustDensity > 1.2) {
65     if (!buzzerSounding) {
66         if (highDensityStartTime == 0) {
67             highDensityStartTime = millis();
68         }
69         if (millis() - highDensityStartTime >= 10000) {
70             digitalWrite(buzzerPin, HIGH);
71             buzzerSounding = true;
72         }
73     }
74 } else {
75     digitalWrite(buzzerPin, LOW);
76     buzzerSounding = false;
77     highDensityStartTime = 0;
78 }
79
80 if (currentMillis - previousSerialMillis >= serialInterval) {
81     previousSerialMillis = currentMillis;
82     Serial.print("Raw Signal Value (0-1023): ");
83     Serial.print(voMeasured);
84     Serial.print(" - Voltage: ");
85
86     Serial.print(calcVoltage);
87     Serial.print(" - Dust Density(mg/m3): ");
88     if (dustDensity < 0) {
89         dustDensity = 0;
90     }
91     Serial.println(dustDensity);
92     Serial.print(" - PM 0.5(particulas/0.01 pie3): ");
93     if (pm05 < 0) {
94         pm05 = 0;
95     }
96     Serial.println(pm05);
97 }
98
99 if (WiFi.status() != WL_CONNECTED) {
100     digitalWrite(redLed, HIGH);
101     digitalWrite(greenLed, LOW);
102
103     if (millis() - previousWiFiRetryMillis >= WiFiRetryInterval) {
104         previousWiFiRetryMillis = millis();
105         WiFi.disconnect();
106         WiFi.begin();
107     }
108 } else {
109     digitalWrite(greenLed, HIGH);
110     digitalWrite(redLed, LOW);
111
112     if (currentMillis - previousThingSpeakMillis >= thingSpeakInterval) {
113         previousThingSpeakMillis = currentMillis;

```

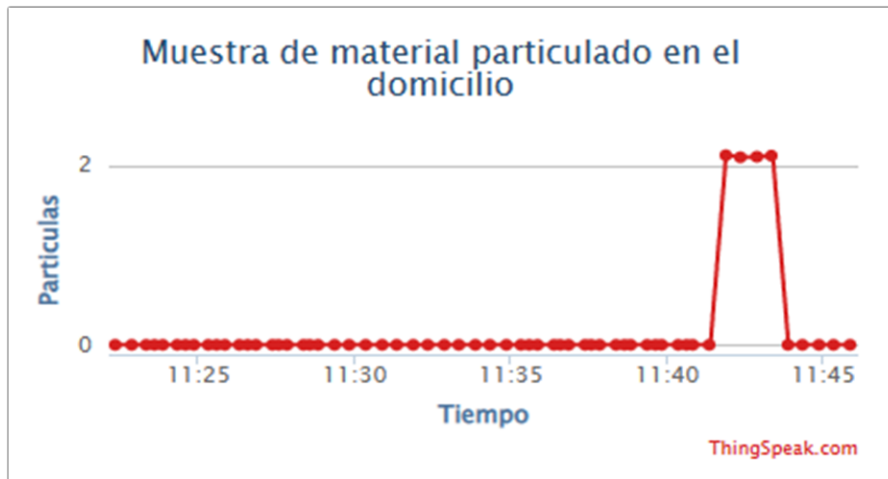
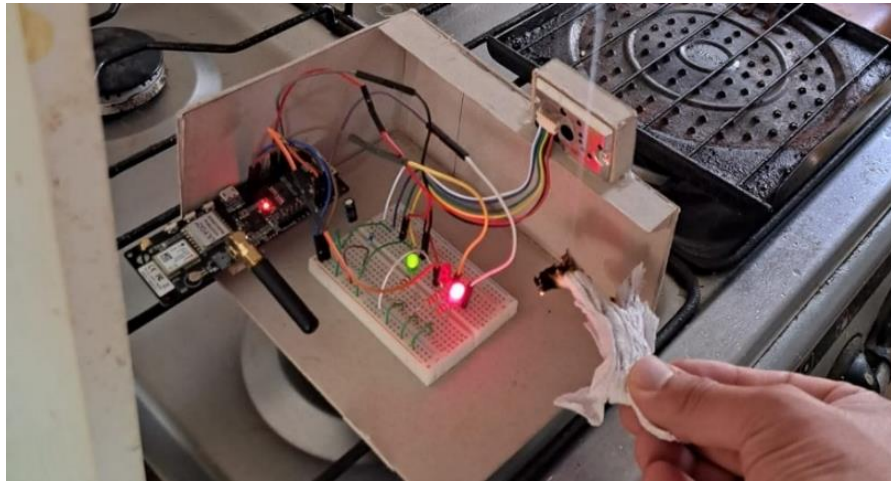
```

113
114         if (dustDensity < 0) {
115             dustDensity = 0;
116         }
117         if (pm05 < 0) {
118             pm05 = 0;
119         }
120
121         ThingSpeak.setField(1, dustDensity);
122         ThingSpeak.setField(2, pm05);
123
124         int updateStatus = ThingSpeak.writeFields(channelID, apiKey);
125
126         if (updateStatus == 200) {
127             Serial.println("Data sent to ThingSpeak");
128         } else {
129             Serial.println("Error sending data to ThingSpeak");
130         }
131     }
132 }
133 }

```

Quedando como conclusión lo siguiente:

Figura N° 31 Demostración de funcionamiento de material particulado



Como se observa en la Figura N° 31 imagen superior, se realizó una simulación de material particulado (humo) mediante la utilización de papel quemado. El sensor detectó la presencia de MP y este emitió la señal al led rojo el cual representó al buzzer, que, por cuestión de principio de funcionamiento, no se captó la emisión sonora de este componente.

En la misma figura, pero imagen inferior, se observó la cantidad de MP generado en un tiempo determinado, y en donde determinado tiempo estuvo funcionando el buzzer (representado por led rojo).

4.2 Adición del sistema auxiliar de alimentación para reforzar la autonomía del sistema de medición de MP.

Otra mejora al sistema de medición de calidad de aire, fue un circuito auxiliar alimentado por panel solar, para potenciar su autonomía energética. Este circuito fue diseñado para mantener el sistema energizado de manera sostenible y continua, independientemente de la red eléctrica.

Detalles de la Integración del Circuito Alimentado por Panel Solar:

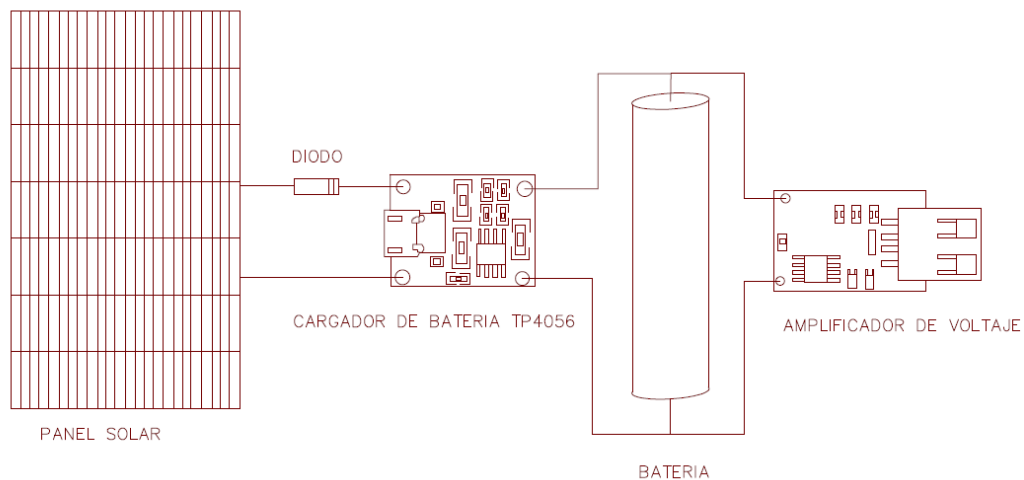
Funcionalidad: El panel solar provee una fuente de energía sostenible al sistema, asegurando su funcionamiento incluso en condiciones climáticas variables.

Conexión con el Sistema Principal: El panel solar está conectado a un controlador y protector de carga y a una batería de respaldo, garantizando una alimentación constante y estable.

Beneficios: Asegura la disponibilidad continua de energía para el sistema, mejorando su fiabilidad y capacidad operativa.

Impacto en la Aplicación Práctica: Esta integración mejoró la confiabilidad y la independencia del sistema, asegurando su funcionamiento ininterrumpido y aumentando su utilidad en condiciones ambientales diversas.

Figura N° 32 Diagrama de sistema auxiliar de alimentación



Fuente: https://www.youtube.com/watch?v=4rVHhKAHcz4&t=237s&ab_channel=AgriculturaElectronica

El sistema de alimentación solar para el microcontrolador se compone de varios elementos esenciales que aseguran un suministro estable y confiable de energía:

Panel Solar: El panel solar, como fuente principal de energía, captura la luz solar y la convierte en energía eléctrica. Esta energía es dirigida al circuito para su almacenamiento y uso posterior.

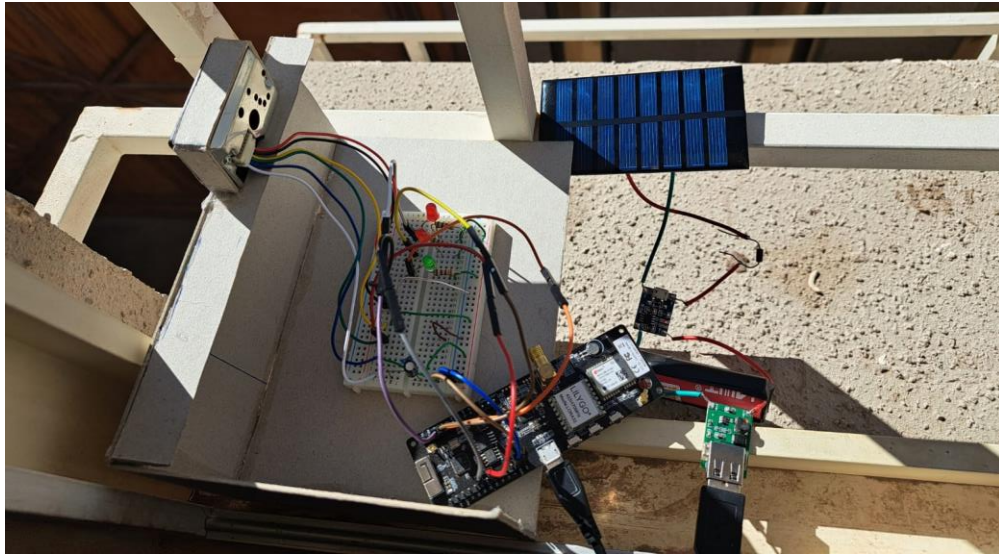
Cargador TP4056: Este componente juega un papel crítico al regular la carga de la batería. El TP4056 estabiliza la tensión y protege la batería de sobrecargas. Sus LED indican el estado de carga: el LED rojo señala la carga en curso, mientras que el LED azul indica una batería completamente cargada.

Batería de Litio: Se utiliza una batería de litio de 3.7V como un banco de energía. Esta batería almacena la electricidad generada por el panel solar para su uso cuando no haya luz solar disponible.

Voltage Booster: El booster de voltaje es esencial para estabilizar la tensión de salida y mantenerla constante a 5V DC, requeridos por el microcontrolador. Su función principal es elevar la tensión de la batería de litio a un nivel adecuado para alimentar el microcontrolador de manera eficiente y constante.

Diodo de Protección: El diodo se utiliza para evitar la descarga inversa de la batería de litio a través del panel solar cuando no hay luz.

Figura N° 33 Sistema auxiliar de alimentación mediante panel solar



4.3 Otras Opciones Consideradas para Mejoras Futuras

A pesar de la implementación exitosa de la alerta acústica, existen varias posibilidades de mejora que podrían ser consideradas para futuras interacciones con el proyecto:

Integración de Alertas Visuales: Explorar la inclusión de indicadores visuales, como luces LED o una pantalla visual, para complementar la alerta acústica y proporcionar una notificación más amplia y accesible.

Conectividad Remota y Notificaciones en Tiempo Real: Investigar la viabilidad de integrar capacidades de conectividad remota para enviar notificaciones en tiempo real a dispositivos móviles o a una plataforma en la nube, permitiendo una supervisión continua y remota de la calidad del aire.

Mejora en la Precisión de las Mediciones: Considerar la calibración adicional de los sensores o la exploración de sensores más precisos para mejorar la exactitud de las mediciones y la detección temprana de condiciones de aire de baja calidad.

Estas opciones representan posibles direcciones para futuras mejoras en el sistema, ampliando su funcionalidad y precisión para abordar aún más desafíos en la medición y monitoreo de la calidad del aire.

CONCLUSIONES

En la culminación de este proyecto de implementación de un sistema de monitoreo de calidad del aire, se ha logrado con éxito la integración de tecnologías clave, destacando la combinación de la placa LilyGO ESP32 T-Beam V1.1, el sensor GP2Y1010AU0F-V3 y la plataforma ThingSpeak. Este enfoque ha resultado ser altamente efectivo para la recopilación, almacenamiento y visualización de datos en tiempo real, ofreciendo una concientización al usuario, por la calidad del aire y su impacto en la salud humana y el medio ambiente.

A lo largo del desarrollo de este proyecto, se identificó una limitación significativa que ha sido abordada con éxito. La ausencia de un mecanismo eficaz de alerta para condiciones críticas de contaminación atmosférica se resolvió mediante la implementación de una alerta acústica. La adición de un timbre proporciona notificaciones inmediatas y audibles cuando los niveles de contaminación que exceden los umbrales seguros predefinidos, mejorando considerablemente la utilidad práctica del sistema y permitiendo una respuesta proactiva ante condiciones adversas de calidad del aire.

La integración de un sistema de alimentación adicional al proyecto principal es una ventaja que puede satisfacer notablemente la necesidad del consumidor al evitarle el contacto directo con los componentes para un posible cambio de batería, ya que, la alimentación principal de los componentes estará siempre respaldada por el sistema de alimentación auxiliar siempre y cuando este reciba la irradiación solar por medio del módulo solar incorporado y así entregar un suministro constante de energía.

La consideración de mejoras futuras, como la integración de alertas visuales, subraya el compromiso continuo con la evolución y perfeccionamiento del sistema. Estas mejoras potenciales sugieren una dirección para expandir la funcionalidad y la precisión del sistema, abordando desafíos adicionales en la medición y monitoreo de la calidad del aire. En última instancia, este proyecto no solo ha proporcionado una solución para la monitorización ambiental, sino que también establece una base sólida para futuras innovaciones y desarrollos en el ámbito de la calidad del aire.

BIBLIOGRAFÍA

- ACHS. (s.f.). *Silicosis*. Obtenido de https://www.achs.cl/docs/librariesprovider2/empresa/centro-de-fichas/trabajadores/silicosis.pdf?sfvrsn=5d12126c_0
- Arduino. (s.f.). *¿Qué es Arduino?* Obtenido de <https://www.arduino.cc/en/Guide/Introduction>
- Arduino. (s.f.). *Guia de referencia de Arduino*. Obtenido de <https://www.arduino.cc/reference/es/>
- congreso, B. d. (2011). *Decreto 12*. Obtenido de <https://www.bcn.cl/leychile/navegar?idNorma=1025202>
- Electronica, A. (s.f.). *Youtube*. Obtenido de https://www.youtube.com/watch?v=4rVHhKAHcz4&t=237s&ab_channel=AgriculturaElectronica
- Epa. (s.f.). *Conceptos básicos sobre el material particulado*. Obtenido de <https://espanol.epa.gov/espanol/conceptos-basicos-sobre-el-material-particulado-pm-por-sus-siglas-en-ingles>
- EPA. (s.f.). *Efectos del material particulado (PM) sobre la salud y el medioambiente*. Obtenido de <https://espanol.epa.gov/espanol/efectos-del-material-particulado-pm-sobre-la-salud-y-el-medioambiente#:~:text=Efectos%20sobre%20la%20salud&text=s%3%ADntomas%20respiratorios%20aumentados%2C%20como%20irritaci%C3%B3n,pulmonares%2C%20ni%C3%Blas%20y%20adultos>
- Gidahatar. (s.f.). Obtenido de <https://gidahatari.com/ih-es/tutorial-del-instalacion-y-puesta-en-marcha-de-un-sensor-de-polvo-en-arduino>
- Mathworks. (s.f.). *Thinkspeak*. Obtenido de <https://la.mathworks.com/help/thingspeak/mechatronicstore>
- mechatronicstore. (s.f.). *Controlador tp4056*. Obtenido de <https://www.mechatronicstore.cl/controlador-tp4056-para-bat-litio-5v-1a-mini-usb/>

Openai, N. (2024). *ChatGPT*. Obtenido de <https://help.openai.com/en/articles/6783457-what-is-chatgpt>

Pallás Areny, R. (2003). *Google académico*. Obtenido de Sensores y acondicionadores de señal:

https://books.google.com.pe/books?hl=es&lr=&id=Eevyk28_fVkC&oi=fnd&pg=PR11&dq=sensor&ots=JZfK00Epdf&sig=e1Wen8VlePHzJH4Cns2FkBMb22E#v=onepage&q=sensor&f=false

randomnerdtutorials. (s.f.). *ESP32*. Obtenido de <https://randomnerdtutorials.com/getting-started-with-esp32/>

Wikipedia. (s.f.). *Microcontrolador*. Obtenido de

[https://es.wikipedia.org/wiki/Microcontrolador#:~:text=Un%20microcontrolador%20\(abreviado%20%C2%B5C%2C%20UC,que%20cumplen%20una%20tarea%20espec%C3%ADfica.](https://es.wikipedia.org/wiki/Microcontrolador#:~:text=Un%20microcontrolador%20(abreviado%20%C2%B5C%2C%20UC,que%20cumplen%20una%20tarea%20espec%C3%ADfica.)

Wikipedia. (s.f.). *Zumbador*. Obtenido de

[https://es.wikipedia.org/wiki/Zumbador#:~:text=Un%20zumbador%20\(en%20ingl%C3%A9s%20buzzer,en%20electrodom%C3%A9sticos%20incluidos%20los%20despertadores.](https://es.wikipedia.org/wiki/Zumbador#:~:text=Un%20zumbador%20(en%20ingl%C3%A9s%20buzzer,en%20electrodom%C3%A9sticos%20incluidos%20los%20despertadores.)

ANEXO

1.- EXPLICACIÓN DE LOS CÓDIGOS: (Arduino, Guía de referencia de Arduino, s.f.)

1. void setup ():

El setup () es una función especial en el lenguaje de programación Arduino. Esta función se ejecuta una sola vez al inicio del programa, justo después de que la placa Arduino se inicia o se reinicia. Su propósito principal es configurar y preparar el entorno para que el programa funcione correctamente. Aquí es donde se suelen realizar tareas como:

- Inicializar pines (configuración como entrada o salida).
- Establecer velocidades de comunicación (por ejemplo, la velocidad de comunicación serie).
- Realizar configuraciones iniciales, como encender un LED de estado.

El setup () es esencial para preparar el hardware y el software antes de que el programa principal comience a funcionar en el bucle principal.

2. void loop ():

El loop () es otra función especial en Arduino y es el núcleo del programa. Una vez que el setup () ha completado su ejecución, el Arduino entra en un bucle infinito en el que ejecuta continuamente el código dentro de la función loop (). En otras palabras, el código dentro de loop () se ejecuta una y otra vez en un ciclo sin fin mientras la placa Arduino esté encendida.

Aquí es donde se coloca la lógica principal del programa, como:

- Lectura de sensores.
- Control de actuadores.
- Toma de decisiones en función de la entrada del entorno.
- Comunicación con otros dispositivos o sistemas.

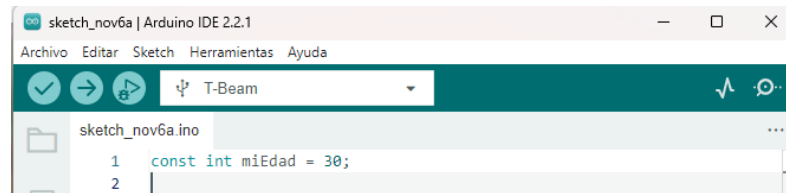
Cualquier acción que deba repetirse continuamente, como la interacción en tiempo real con el entorno, se programa dentro de loop (). Esta función es la que permite que el programa Arduino responda de manera dinámica a su entorno y realice las tareas deseadas.

3. const:

La palabra clave `const` se utiliza para declarar una constante en programación, es decir, un valor que no puede ser modificado una vez que se le asigna un valor inicial.

Las constantes son útiles para definir valores que no deben cambiar durante la ejecución del programa, como valores fijos como pi (π) o umbrales de comparación.

Por ejemplo, puedes declarar una constante como esta:



```
sketch_nov6a.ino
1  const int miEdad = 30;
2
```

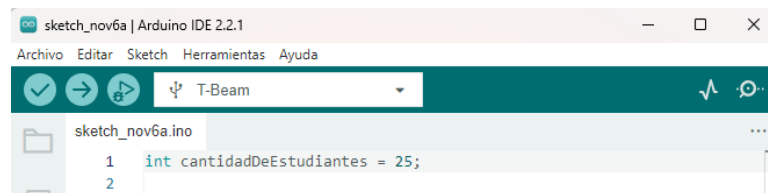
Esto declara una constante llamada `miEdad` con un valor de 30, y este valor no puede cambiarse en el transcurso del programa.

4. int:

`int` es un tipo de dato que se utiliza para representar números enteros, es decir, números sin parte decimal.

Los valores `int` pueden ser positivos o negativos y se utilizan comúnmente para contar, indexar elementos en arreglos o realizar cálculos aritméticos enteros.

Por ejemplo, puedes declarar una variable `int` de la siguiente manera:



```
sketch_nov6a.ino
1  int cantidadDeEstudiantes = 25;
2
```

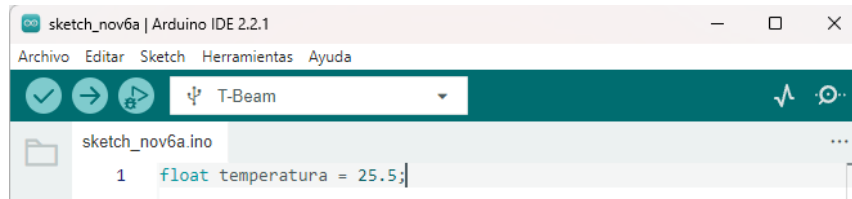
Esto declara una variable llamada `cantidadDeEstudiantes` con un valor de 25, que es un número entero.

5. float:

`float` es un tipo de dato que se utiliza para representar números en coma flotante, es decir, números que pueden tener parte decimal.

Los valores float se utilizan para representar números reales y se usan comúnmente en cálculos que involucran valores con decimales, como mediciones o cálculos matemáticos que requieren precisión.

Por ejemplo, se puede declarar una variable float de la siguiente manera:



The screenshot shows the Arduino IDE interface. The window title is "sketch_nov6a | Arduino IDE 2.2.1". The menu bar includes "Archivo", "Editar", "Sketch", "Herramientas", and "Ayuda". The toolbar contains icons for check, run, and upload, along with a dropdown menu set to "T-Beam". The main editor area shows the file "sketch_nov6a.ino" with the following code on line 1: `float temperatura = 25.5;`

Esto declara una variable llamada temperatura con un valor de 25.5, que es un número en coma flotante.

6. Serial.begin ():

Es una función utilizada en programación Arduino para configurar la comunicación serie (UART o USART) entre la placa Arduino y una computadora o dispositivo externo.

Se utiliza para inicializar la velocidad de transmisión de datos entre el Arduino y el dispositivo con el que se está comunicando. La velocidad se especifica en baudios (bps, bits por segundo).

Por lo general, esta función se llama en la función setup () para configurar la comunicación serie antes de entrar en el bucle principal (loop ()).

Ejemplo:



The screenshot shows the Arduino IDE interface. The window title is "sketch_nov6a | Arduino IDE 2.2.1". The menu bar includes "Archivo", "Editar", "Sketch", "Herramientas", and "Ayuda". The toolbar contains icons for check, run, and upload, along with a dropdown menu set to "T-Beam". The main editor area shows the file "sketch_nov6a.ino" with the following code:

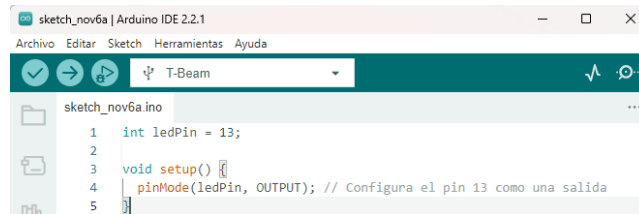
```
1 void setup() {
2   Serial.begin(9600); // Inicializa la comunicación serie a 9600 baudios
3 }
```

7. pinMode():

Es una función que se utiliza para configurar un pin en la placa Arduino como una entrada o una salida. Esto es esencial para definir cómo se utilizará un pin en un proyecto.

Se utiliza para establecer si un pin se utilizará para recibir señales de entrada (como desde sensores) o para enviar señales de salida (como para controlar un LED o un motor).

Ejemplo:



```
sketch_nov6a.ino
1 int ledPin = 13;
2
3 void setup() {
4   pinMode(ledPin, OUTPUT); // Configura el pin 13 como una salida
5 }
```

8. digitalWrite():

Es una función que se utiliza para cambiar el estado de un pin digital en una placa Arduino. Puede establecer un pin en estado alto (HIGH, que generalmente es 5V) o en estado bajo (LOW, que generalmente es 0V).

Se utiliza principalmente para controlar dispositivos conectados a los pines digitales, como LED, relés y otros actuadores.

Ejemplo:



```
sketch_nov6a.ino
1 int ledPin = 13;
2
3 void setup() {
4   pinMode(ledPin, OUTPUT); // Configura el pin 13 como una salida
5 }
6
7 void loop() {
8   digitalWrite(ledPin, HIGH); // Enciende el LED conectado al pin 13
9   delay(1000); // Espera 1 segundo
10  digitalWrite(ledPin, LOW); // Apaga el LED
11  delay(1000);
12 }
```

9. Serial.print():

Es una función utilizada para enviar datos a través de la comunicación serie, generalmente para mostrar información en el monitor serie de la computadora o para enviar datos a otros dispositivos serie.

Se puede utilizar Serial.print() para mostrar valores de variables, mensajes de estado o cualquier otra información relevante en el monitor serie.

Ejemplo:



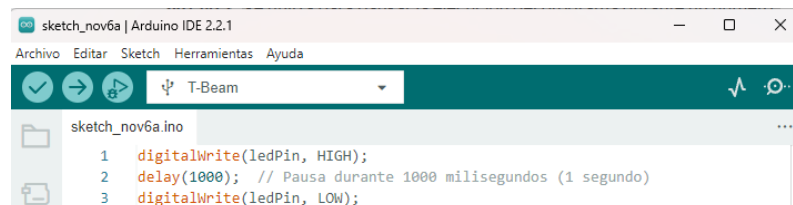
```
sketch_nov6a.ino
1  int sensorValue = 512;
2
3  void setup() {
4    Serial.begin(9600); // Inicializa la comunicación serie a 9600 baudios
5  }
6
7  void loop() {
8    Serial.print("Valor del sensor: ");
9    Serial.print(sensorValue); // Muestra el valor del sensor
10   Serial.println(); // Agrega un salto de línea para una mejor legibilidad
11   delay(1000);
12 }
```

10. delay(millisecods):

Se utiliza para pausar la ejecución del programa durante un número específico de milisegundos. Milliseconds es el tiempo de retraso en milisegundos.

Es útil para crear intervalos de tiempo en tu programa, como esperar antes de realizar una acción o crear parpadeos con LEDs.

Por ejemplo, para hacer un parpadeo de un segundo en un LED:



```
sketch_nov6a.ino
1  digitalWrite(ledPin, HIGH);
2  delay(1000); // Pausa durante 1000 milisegundos (1 segundo)
3  digitalWrite(ledPin, LOW);
```

11. unsigned long:

Las variables largas sin signo son variables de tamaño extendido para el almacenamiento de números y almacenan 32 bits (4 bytes). A diferencia de las posiciones largas estándar, las posiciones largas sin signo no almacenarán números negativos, lo que hace que su rango sea de 0 a 4.294.967.295 ($2^{32} - 1$).

Por ejemplo

```

sketch_jan21a.ino
1 unsigned long time;
2
3 void setup()
4 {
5   Serial.begin(9600);
6 }
7
8 void loop()
9 {
10  Serial.print("Time: ");
11  time = millis();
12  //prints time since program started
13  Serial.println(time);
14  // wait a second so as not to send massive amounts of data
15  delay(1000);
16 }

```

12. If:

La sentencia comprueba si hay una condición y ejecuta la sentencia de procedimiento o el conjunto de sentencias si la condición es 'verdadera'.if

Los corchetes se pueden omitir después de una instrucción if. Si se hace esto, la siguiente línea (definida por el punto y coma) se convierte en la única instrucción condicional. Por ejemplo:

```

sketch_jan21a.ino
1 if (x > 120) digitalWrite(LEDpin, HIGH);
2
3 if (x > 120)
4 digitalWrite(LEDpin, HIGH);
5
6 if (x > 120){ digitalWrite(LEDpin, HIGH); }
7
8 if (x > 120){
9   digitalWrite(LEDpin1, HIGH);
10  digitalWrite(LEDpin2, HIGH);
11 } // all are correct

```

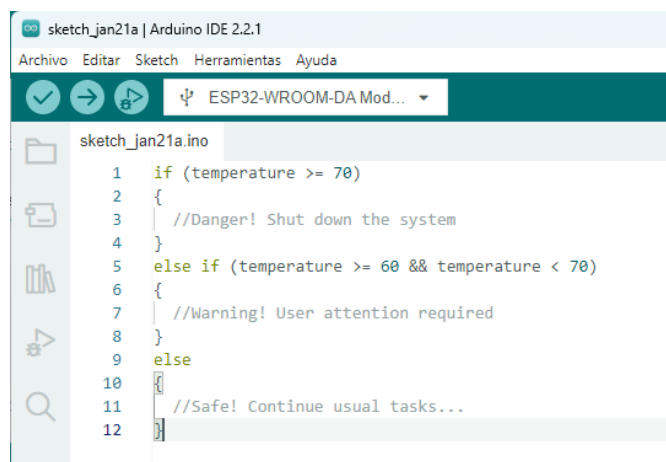
13. Else

Esto permite un mayor control sobre el flujo de código que la instrucción if básica, al permitir que se agrupen varias pruebas. Se ejecutará una cláusula (si existe) si la condición de la instrucción da como resultado. Pueden realizar otra prueba, de modo que se puedan ejecutar varias pruebas mutuamente excluyentes al mismo tiempo.if...elseelseiffalseelseif

Cada prueba pasará a la siguiente hasta que se encuentre una prueba verdadera. Cuando se encuentra una prueba verdadera, se ejecuta su bloque de código asociado y el programa salta a la línea que sigue a toda la construcción if/else. Si ninguna prueba resulta ser verdadera, se ejecuta el bloque predeterminado, si hay uno, y establece el comportamiento predeterminado.else

Tenga en cuenta que un bloque se puede usar con o sin un bloque de terminación y viceversa. Se permite un número ilimitado de estas sucursales.else ifelse if

Por ejemplo:



```
sketch_jan21a.ino
1  if (temperature >= 70)
2  {
3    //Danger! Shut down the system
4  }
5  else if (temperature >= 60 && temperature < 70)
6  {
7    //Warning! User attention required
8  }
9  else
10 {
11 //Safe! Continue usual tasks...
12 }
```

14. #include

#include se utiliza para incluir bibliotecas externas en el boceto. Esto le da al programador acceso a un gran grupo de bibliotecas C estándar (grupos de funciones prefabricadas), y también a bibliotecas escritas especialmente para Arduino.

Tenga en cuenta que, al igual que #define, no tiene terminador de punto y coma, y el compilador producirá mensajes de error crípticos si agrega uno.