Universidad de San Carlos de Guatemala.
Facultad de ingeniería.
Escuela de ingeniería en ciencias y sistemas.
Área: ciencias de la computación
Curso: Arquitectura de computadores y ensambladores 2
Catedratico: Ing Jurgen Ramirez
Auxiliares: Samuel Pérez, Danny Cuxum

Proyecto Único - **Fase 1**

Control de Acceso Vehicular en Sistemas Inteligentes Implementando Análisis Meteorológico IoT

Integrantes del ***Grupo 1***:

- Jemima Solmaira Chavajay Quiejú - 201801521
- Evelio Marcos Josué Cruz Soliz - 202010040
- Santiago Julián Barrera Reyes - 201905884
- Elías Abraham Vasquez Soto - 201900131
- Giovanni Saul Concohá Cax - 202100229

https://github.com/Ggi0/ARQUI2B_2S2024_G1.git

Fecha de Entrega: 17/08/2024

Índice:

1. Introducción

El presente proyecto se enfoca en el desarrollo de un sistema avanzado de control de acceso vehicular que integra una estación meteorológica basada en Internet de las Cosas (IoT). Este sistema innovador busca elevar la eficiencia y seguridad en el control de acceso mediante la recopilación y análisis de datos ambientales en tiempo real.

Utilizando un microcontrolador Arduino como núcleo, el sistema incorpora una variedad de sensores para medir cuatro variables principales: temperatura, humedad, iluminación y concentración de CO2 en el aire. Además, se implementa un sensor de proximidad para detectar la presencia de vehículos u objetos, mejorando así la precisión del control de acceso.

La información recopilada se procesa y visualiza a través de: una pantalla LCD conectada al Arduino, que proporciona datos en tiempo real, y una aplicación desarrollada en Processing, que ofrece un dashboard más completo y detallado. Este enfoque dual permite tanto la visualización inmediata en el sitio como un análisis más profundo desde una estación de control.

Un aspecto clave del sistema es la implementación de una cola de mensajes basada en el protocolo MQTT, que garantiza la gestión eficiente de los datos generados. Esto no solo asegura el análisis completo de toda la información recopilada, sino que también sienta las bases para futuras expansiones del sistema.

A lo largo de este documento, se detallarán los componentes hardware y software utilizados, los procesos de implementación, y se analizará cómo este sistema contribuye a la optimización del control de acceso vehicular mediante la integración de datos meteorológicos. El proyecto se desarrollará en fases, siendo esta la primera, centrada en la implementación básica del hardware y software necesarios para la recopilación y visualización de datos.

2. Objetivos

Objetivo General:

Desarrollar un sistema inteligente de control de acceso vehicular, integrado con una estación meteorológica IoT, capaz de monitorear, registrar y analizar diversas variables ambientales en tiempo real, utilizando sensores y una plataforma de visualización centralizada.

Objetivos Específicos:

1. Diseñar e implementar un dispositivo que mida y registre variables meteorológicas como temperatura, humedad, iluminación y concentración de CO2, además de detectar proximidad a través de un sensor ultrasónico.
2. Desarrollar una solución que integre de manera eficiente los sensores, el microcontrolador Arduino y la plataforma de visualización, garantizando la correcta comunicación y procesamiento de los datos.
3. Implementar una plataforma centralizada basada en el framework IoT que gestione los datos recopilados y controle el sistema de acceso vehicular, utilizando un sistema de cola de mensajes y proporcionando una visualización accesible a través de una pantalla LCD e interfaz en Processing.

3 Marco Teórico

Concepto de IoT y su Importancia

El Internet de las Cosas (IoT, por sus siglas en inglés) es un concepto tecnológico que ha revolucionado la manera en que los dispositivos cotidianos interactúan con el entorno y los usuarios. IoT se refiere a la red colectiva de dispositivos conectados a Internet que recopilan y comparten datos en tiempo real. Estos dispositivos, equipados con sensores y capacidad de procesamiento, comunican la información recolectada a través de la nube o entre ellos mismos, posibilitando la automatización y toma de decisiones basada en inteligencia artificial (IA) o machine learning (ML).

La importancia del IoT radica en su capacidad para optimizar procesos, mejorar la eficiencia en diversas industrias y facilitar la vida diaria. Desde electrodomésticos inteligentes hasta complejos sistemas industriales, IoT se ha extendido a múltiples sectores como la salud, el transporte y la agricultura. En este proyecto, IoT desempeña un papel crucial al permitir la medición y registro de variables meteorológicas, así como el control automatizado de acceso vehicular, mejorando tanto la seguridad como la eficiencia operativa.

De igual modo, el Internet de las Cosas (IoT) se basa en una arquitectura de capas interconectadas que permiten la recopilación, procesamiento y análisis de datos desde dispositivos físicos hasta aplicaciones en la nube. Este stack tecnológico se compone de cinco capas principales:

1. Capa de Objetos (Things): Esta es la base del stack y representa los objetos físicos del mundo real que se conectan a internet. Incluye dispositivos cotidianos como electrodomésticos, vehículos, sensores ambientales y otros objetos que pueden recopilar o generar datos.
2. Capa de Hardware del Dispositivo: Comprende los componentes físicos integrados en los objetos IoT. Esto incluye sensores, actuadores, módulos de interfaz PLC, gateways y otros elementos de hardware que permiten la captura de datos y la interacción con el entorno.
3. Capa de Software del Dispositivo: Se refiere al software que se ejecuta en el procesador del dispositivo y controla su funcionalidad. Puede incluir sistemas operativos embebidos y programas escritos en lenguajes como Linux, C, Perl, Python o Qt, adaptados a las capacidades y requisitos específicos del dispositivo.
4. Capa de Comunicaciones: Esta capa define cómo los dispositivos se conectan a internet y transfieren datos. Incluye diversas tecnologías de conectividad como conexiones cableadas, Bluetooth, Wi-Fi, Zigbee, redes celulares (LTE-M/NB-IoT/2G/3G/4G/5G) y redes de área amplia de baja potencia (LoRaWAN).

5. Capa de Plataforma en la Nube: Es donde se capturan, procesan y almacenan los datos provenientes de los dispositivos IoT. Puede ser implementada en servidores físicos o en plataformas de nube como Amazon Web Services (AWS), Microsoft Azure o Google Cloud Platform (GCP).
6. Capa de Aplicaciones en la Nube: Representa la interfaz de usuario final para dashboards, configuraciones y gestión de dispositivos. Esta capa proporciona herramientas para la gestión de dispositivos, visualización de datos, análisis, configuración de alertas y alarmas, y aplicación de técnicas de aprendizaje automático.

Este stack tecnológico permite una integración fluida desde la captura de datos en el mundo físico hasta su análisis y presentación en aplicaciones sofisticadas, facilitando la creación de soluciones IoT completas y escalables.

Control de Acceso Vehicular

El control de acceso vehicular es un aspecto fundamental en la gestión de entradas y salidas en establecimientos como estacionamientos, centros logísticos, o incluso en el tránsito urbano. Tradicionalmente, estos sistemas se basaban en tecnologías simples, como tarjetas de identificación o barreras manuales. Sin embargo, con el avance de tecnologías como IoT, se han desarrollado soluciones más eficientes y seguras, capaces de gestionar el acceso vehicular de manera autónoma.

En este proyecto, se utiliza un sistema de IoT que integra sensores como el ultrasonido HC-SR04, junto con una pantalla LCD con interfaz I2C, para medir la proximidad de los vehículos y mostrar la información relevante en tiempo real. Esto permite una gestión más fluida y precisa del control de acceso, al mismo tiempo que se facilita la implementación de algoritmos de predicción de patrones, lo que mejora el flujo vehicular y reduce tiempos de espera.

Sistemas Meteorológicos y su Relación con IoT

La evolución de los sistemas meteorológicos ha sido impulsada por la tecnología IoT, permitiendo el despliegue de sensores conectados que recopilan datos ambientales en tiempo real desde diversas ubicaciones. Estos sensores monitorean variables clave como temperatura, humedad, CO2 y luminosidad, transmitiendo la información a plataformas centralizadas para su análisis y previsión de condiciones climáticas.

La integración de IoT en los sistemas meteorológicos facilita no solo un monitoreo eficiente del clima, sino también la implementación de sistemas predictivos que anticipan fenómenos meteorológicos. En este proyecto, se emplea sensor DHT11 para registrar datos ambientales y mostrarlos en una pantalla LCD, proporcionando una solución avanzada para la gestión de datos meteorológicos en tiempo real.

Este enfoque mejora la toma de decisiones en el control vehicular, ajustando el sistema según las condiciones ambientales para aumentar la seguridad y eficiencia. Los sistemas meteorológicos inteligentes, al ser accesibles y ofrecer monitoreo en tiempo real, superan las limitaciones de las estaciones tradicionales que suelen ser costosas y regionales.

EEPROM en Arduino:

La EEPROM (Electrically Erasable Programmable Read-Only Memory) es una memoria no volátil que permite a los microcontroladores de Arduino almacenar datos que persisten incluso después de apagar el dispositivo. Esta capacidad es esencial para aplicaciones donde es necesario conservar configuraciones o datos entre ciclos de encendido y apagado. Dependiendo del microcontrolador en la placa Arduino, la cantidad de EEPROM disponible varía: por ejemplo, el ATmega2560 ofrece 4 KB, mientras que el ATmega328P proporciona 1 KB. Para interactuar con la EEPROM, Arduino utiliza la librería `EEPROM.h`, que facilita operaciones como leer y escribir datos en la memoria.

Las funciones clave de esta librería incluyen `EEPROM.read()` para obtener el valor almacenado en una dirección específica de la EEPROM, y `EEPROM.write()` para escribir un valor en una dirección determinada. También existen funciones como `EEPROM.put()` y `EEPROM.get()` que permiten almacenar y recuperar valores complejos, como estructuras de datos o variables de punto flotante. Otra función importante es `EEPROM.update()`, que escribe en la memoria solo si el valor es diferente del que ya está almacenado, lo cual ayuda a prolongar la vida útil de la EEPROM al reducir la cantidad de escrituras innecesarias. Estas herramientas son fundamentales para aplicaciones que requieren la preservación de datos, como la configuración de dispositivos o el almacenamiento de registros históricos en sistemas embebidos.

Beneficios

1. Monitoreo en Tiempo Real: El sistema proporciona datos en tiempo real sobre variables meteorológicas clave como temperatura, humedad, iluminación, y niveles de CO2, permitiendo una respuesta rápida ante condiciones adversas.
2. Optimización de Recursos: Al integrar datos meteorológicos en el control de acceso vehicular, se pueden optimizar recursos como la energía al activar o desactivar componentes en función de las condiciones ambientales.
3. Mejora en la Toma de Decisiones: El análisis de datos meteorológicos recopilados permite predecir patrones climáticos, facilitando decisiones informadas que pueden mejorar la eficiencia y seguridad del sistema de acceso vehicular.

4. Escalabilidad y Flexibilidad: Al ser un sistema modular, permite la integración de nuevos sensores y tecnologías en el futuro, lo que lo hace adaptable a diferentes necesidades.

Impacto Ambiental

1. Reducción de la Huella de Carbono: El monitoreo de la calidad del aire mediante la medición de CO2 puede contribuir a estrategias de reducción de emisiones, mejorando la calidad del aire en el entorno del sistema.
2. Uso Eficiente de Energía: El sistema puede reducir el consumo energético al regular la iluminación y otros dispositivos electrónicos en función de las condiciones meteorológicas, lo que ayuda a disminuir el uso innecesario de energía.
3. Contribución a la Sostenibilidad: Al utilizar recursos de manera más eficiente y reducir las emisiones, el sistema contribuye a los objetivos de sostenibilidad ambiental.
4. Detección de Condiciones Peligrosas: Al monitorizar variables ambientales críticas, el sistema puede ayudar a detectar y mitigar condiciones potencialmente peligrosas para el medio ambiente, como la contaminación por CO2.

3.1 Descripción General

El proyecto "Control de acceso vehicular en sistemas inteligentes implementando análisis meteorológico IoT" representa una innovadora solución que fusiona la gestión de acceso vehicular con la monitorización ambiental en tiempo real. Este sistema utiliza tecnologías IoT para mejorar la seguridad y eficiencia en entornos de control de acceso, mientras recopila y analiza datos meteorológicos cruciales.

En el núcleo del sistema se encuentra un microcontrolador Arduino Mega, que actúa como cerebro de la operación, coordinando la recolección de datos de diversos sensores. Estos incluyen:

1. Sensores de temperatura y humedad (DHT11)
2. Sensor de movimiento ultrasónico (HC-SR04)
3. Sensor de iluminación (MÓDULO DE FOTORESISTENCIA DIGITAL)
4. Sensor de CO2 (MQ135)

Estos componentes trabajan en conjunto para proporcionar una visión integral del entorno, permitiendo no solo el control de acceso vehicular sino también el monitoreo de las condiciones ambientales.

La interfaz de usuario se implementa a través de una pantalla LCD conectada al Arduino, ofreciendo visualización en tiempo real de los datos recopilados. Adicionalmente, se utiliza Processing para crear un dashboard más completo y detallado, permitiendo una visualización gráfica avanzada de los datos en una computadora conectada.

El sistema incorpora botones físicos para la interacción del usuario, facilitando la navegación entre diferentes modos de visualización y el almacenamiento de datos históricos en la memoria EEPROM del Arduino.

La arquitectura del proyecto se alinea con el stack tecnológico IoT, abarcando las siguientes capas:

1. Capa de Dispositivos: Sensores y actuadores físicos.
2. Capa de Hardware: Arduino Mega como plataforma de control.
3. Capa de Software: Programación en Arduino IDE y Processing.

4. Capa de Comunicación: Conexión serial entre Arduino y computadora.
5. Capa de Aplicación: Dashboard en Processing para visualización y análisis de datos.

Este enfoque permite una integración fluida desde la captura de datos hasta su presentación y análisis, creando un sistema robusto y escalable.

El proyecto no solo mejora la seguridad en el control de acceso vehicular, sino que también proporciona valiosos insights sobre las condiciones ambientales del entorno. Esto puede ser utilizado para optimizar la gestión de instalaciones, mejorar la eficiencia energética y contribuir a la toma de decisiones basada en datos en diversos escenarios, desde campus universitarios hasta complejos industriales.

En resumen, este proyecto representa un avance significativo en la integración de tecnologías IoT con sistemas de control de acceso tradicionales, ofreciendo una solución versátil y altamente funcional que responde a las necesidades modernas de seguridad y gestión ambiental.

4. Descripción de la Solución

4.1 Hardware Utilizado

- **Microcontrolador**: Arduino mega

El **Arduino Mega** es una placa de desarrollo open-source que ofrece amplias capacidades para proyectos avanzados. Equipado con el microcontrolador **ATmega2560**, proporciona un conjunto robusto de características ideales para aplicaciones que demandan más recursos:

- **Entradas/Salidas Digitales**: 54 pines, 14 de los cuales soportan PWM.
- **Entradas Analógicas**: 16 pines para lectura de señales analógicas.
- **Puertos Seriales**: 4 UARTs para comunicación serial.
- **Memoria**: 256 KB de memoria Flash.
- **Velocidad del Reloj**: 16 MHz.

Conectividad incluye un puerto USB para programación y comunicación, un conector de alimentación para 7-12V DC, y un conector ICSP para programación en circuito. También cuenta con un botón de reset y dimensiones de 102 mm x 53 mm.

Compatibilidad: Acepta la mayoría de los shields diseñados para modelos anteriores como el Arduino Uno, lo que facilita la expansión del sistema.

Ventajas: Su capacidad extendida de pines y memoria, junto con versatilidad en comunicaciones, lo hace ideal para gestionar diversos sensores y actuadores en proyectos complejos como el "Control de acceso vehicular en sistemas inteligentes implementando análisis meteorológico IoT".

- **Sensores**:
    - **Temperatura y Humedad**: DHT11

      Descripción: El DHT11 es un sensor digital diseñado para medir la humedad relativa y la temperatura del ambiente. Es ampliamente utilizado en aplicaciones que requieren monitoreo ambiental debido a su bajo costo y facilidad de uso.

      Uso en Arduino: El sensor DHT11 se conecta a un pin digital de la placa Arduino, desde donde se puede leer tanto la temperatura como la humedad. Para facilitar la comunicación con el sensor, se recomienda utilizar la librería DHT, la cual permite acceder a los valores de temperatura y humedad de forma sencilla. El DHT11 es ideal para proyectos de monitoreo ambiental en interiores.

    - **Iluminación:** Fotocelda (LDR - Light Dependent Resistor)

Descripción: Una fotoresistencia o LDR es un sensor pasivo que cambia su resistencia en función de la cantidad de luz que incide sobre su superficie. Este cambio en la resistencia se traduce en una variación de voltaje, lo que permite medir la intensidad lumínica.

Uso en Arduino: La LDR se conecta a un pin analógico del Arduino, donde se mide la variación de voltaje resultante de los cambios en la iluminación. Este sensor es frecuentemente utilizado en proyectos de automatización, como sistemas de iluminación que se activan en condiciones de baja luz, proporcionando una solución eficiente para controlar dispositivos en función de la luz ambiental.

- **Calidad del Aire**: MQ135

Descripción: El MQ-135 es un sensor analógico que detecta la presencia de varios gases en el aire, incluyendo amoníaco, dióxido de carbono, y otros gases nocivos. Es utilizado en sistemas de monitoreo de calidad del aire para detectar concentraciones peligrosas de gases en ambientes cerrados.

Uso en Arduino: El MQ-135 se conecta a un pin analógico del Arduino para medir la concentración de gases en el aire. La señal analógica obtenida se procesa para determinar la calidad del aire en el entorno monitoreado. Este sensor es ideal para aplicaciones en las que se requiere monitorear la presencia de gases peligrosos y activar sistemas de ventilación o alarmas en caso de detección de niveles elevados.

- **Proximidad**: Sensor Ultrasonido HC-SR04

Descripción: El sensor ultrasónico HC-SR04 se utiliza para medir distancias en función del tiempo que tarda un pulso ultrasónico en viajar hacia un objeto y regresar. Este sensor es esencial en aplicaciones donde se requiere la detección precisa de objetos o la medición de distancias.

Uso en Arduino: El HC-SR04 se conecta a dos pines digitales del Arduino: uno para enviar el pulso ultrasónico (Trigger) y otro para recibir el eco de la señal (Echo). El tiempo transcurrido entre el envío y la recepción del pulso se utiliza para calcular la distancia al objeto. Este sensor es comúnmente implementado en sistemas de detección de obstáculos en robótica y automatización.

- **Plataforma de Visualización**: Pantalla LCD

La pantalla LCD utilizada en este proyecto es una unidad de 16x2 caracteres con retroiluminación azul, equipada con un módulo controlador compatible con el

chip HD44780. Esta pantalla permite la visualización clara de información y mensajes en dos líneas de 16 caracteres cada una.

Características Principales:

- **Interfaz I2C:** Facilita la conexión utilizando solo dos pines (SDA y SCL), reduciendo la complejidad del cableado y permitiendo la conexión de hasta 8 pantallas en un solo bus I2C. La dirección I2C por defecto suele ser 0x3F o 0x27, por lo que es crucial verificar y configurar correctamente la dirección en el código para asegurar el funcionamiento adecuado.
- **Consumo de Energía:** Bajo consumo de corriente, ideal para proyectos que requieren eficiencia energética.
- **Ángulo de Visión:** Amplio ángulo de visión, permitiendo una lectura clara desde diferentes ángulos.
- **Versatilidad:** Permite la implementación de menús de usuario, la visualización de datos de sensores y otras aplicaciones interactivas.

4.2 Software Utilizado

- **IDE**:
    - Arduino IDE

El **Arduino IDE** es una plataforma esencial para el desarrollo de proyectos con placas Arduino. Permite escribir, depurar y cargar el código (conocido como "sketches") en la placa. Su interfaz es intuitiva y está dividida en varias secciones clave: la barra de menús, la barra de botones, el editor de código, la barra de consola de mensajes y la barra de estado. Entre las principales funciones se incluyen la verificación y compilación del código, la carga del programa en la placa, y la apertura del monitor serial para la comunicación con el Arduino.

El IDE es compatible con sistemas operativos Windows, macOS y Linux. Para comenzar, se debe descargar e instalar el software desde el sitio web oficial de Arduino, seleccionar el archivo adecuado para el sistema operativo y seguir las instrucciones de instalación. Una vez instalado, se puede utilizar para crear y gestionar sketches, explorar ejemplos predefinidos y organizar proyectos en carpetas específicas.

    - Processing IDE

El **Processing IDE** es una herramienta de desarrollo diseñada para las artes electrónicas y el diseño visual, basada en el lenguaje Java. Su propósito principal es enseñar los fundamentos de la programación a no programadores mediante un enfoque visual. Processing simplifica el proceso de codificación con una interfaz gráfica que facilita la compilación y ejecución del código.

Cada sketch en Processing es una subclase de la clase PApplet, que implementa las principales características del lenguaje. Processing permite crear clases personalizadas dentro del sketch para manejar tipos de datos complejos, superando las limitaciones de los tipos de datos estándar. Además, Processing puede ser utilizado para visualizar datos y generar gráficos en proyectos que involucren Arduino y otros sistemas.

- **Librerías**:

EEPROM

- **Descripción:** La librería `EEPROM.h` permite leer y escribir en la memoria EEPROM de Arduino. La memoria EEPROM (Electrically Erasable Programmable Read-Only Memory) es útil para almacenar datos que deben conservarse entre reinicios del microcontrolador, como configuraciones o ajustes persistentes.

DHT

- **Descripción:** La librería `DHT.h` proporciona funciones para interactuar con sensores de temperatura y humedad DHT11 o DHT22. Facilita la lectura de datos de estos sensores, simplificando el proceso de integración y gestión de la información ambiental en los proyectos.

Wire

- **Descripción:** La librería `Wire.h` es fundamental para la comunicación I2C (Inter-Integrated Circuit) en Arduino. Permite al microcontrolador comunicarse con otros dispositivos I2C, como sensores y pantallas LCD, utilizando un protocolo de comunicación serial de dos cables.

LiquidCrystal_I2C

- **Descripción:** La librería `LiquidCrystal_I2C.h` se utiliza para manejar pantallas LCD que operan mediante comunicación I2C. Simplifica el control de pantallas LCD con interfaz I2C, permitiendo la visualización de información con una interfaz más sencilla y menos cables.

Processing Serial

- **Descripción:** La librería `processing.serial.*` permite a Processing establecer comunicación serial con el Arduino. Facilita la lectura y escritura de datos a través del puerto serial, lo cual es esencial para la integración de Processing con Arduino en proyectos que requieren visualización o control de datos en tiempo real.

5. Implementación del Sistema

5.1 Código de Arduino

El código implementado en Arduino es fundamental para el funcionamiento del sistema de control de acceso vehicular con análisis meteorológico IoT. A continuación, se detallan los aspectos más relevantes:

Inicialización y Configuración: El código comienza incluyendo las bibliotecas necesarias para el manejo de los sensores (DHT para temperatura y humedad) y la pantalla LCD (LiquidCrystal_I2C). Se definen los pines para los diversos sensores y se inicializan las variables globales para almacenar las lecturas.

En la función ***setup(),*** se inicializan los sensores, la comunicación serial, la pantalla LCD y se configuran las interrupciones para los botones de interacción.

Lectura de Sensores: La función sendSensorReadingsToProcessing() es crucial para la recopilación de datos. Esta función:

- Lee la temperatura y humedad del sensor DHT11.
- Obtiene los valores de luminosidad del sensor LDR.
- Captura los datos de calidad del aire del sensor MQ-135.
- Mide la distancia utilizando el sensor ultrasónico HC-SR04.

Cada lectura se imprime en el puerto serial con un formato específico (por ejemplo, "HUMEDAD/valor") para facilitar la comunicación con la aplicación de Processing.

Manejo de la Memoria EEPROM: El código implementa funciones para guardar y recuperar datos de la memoria EEPROM:

- saveDataToEEPROM(): Almacena las lecturas actuales de los sensores en la EEPROM.
- readLastDataFromEEPROM(): Lee y muestra el último conjunto de datos guardado.
- readAllDataFromEEPROM(): Lee y muestra todos los datos almacenados en la EEPROM.

Estas funciones permiten mantener un registro histórico de las mediciones, incluso cuando el dispositivo está apagado.

Interfaz de Usuario con LCD: La pantalla LCD se utiliza para mostrar información en tiempo real y opciones de menú:

- En el loop principal, se muestra un menú básico con opciones para el usuario.

- La función showSensorReadings() muestra las lecturas actuales de los sensores en la pantalla LCD.
- Las funciones de lectura de EEPROM también utilizan la LCD para mostrar los datos históricos.

Sistema de Interrupciones: Se implementan interrupciones para los botones de usuario, permitiendo una interacción fluida:

- Cada botón está asociado a una función específica (por ejemplo, mostrar datos actuales o guardar datos).
- Las interrupciones actualizan una bandera (lcdUpdateFlag) y un identificador de botón (buttonPressed), que luego se procesan en el loop principal.

Comunicación con Processing: El código envía regularmente datos a través del puerto serial, utilizando caracteres especiales ('$' y '#') como delimitadores para sincronizar la comunicación con la aplicación de Processing.

Este diseño modular y eficiente permite una recopilación constante de datos ambientales, su almacenamiento y visualización, formando la base del sistema de control de acceso vehicular con capacidades de análisis meteorológico.

5.2 Código de Processing

El código implementado en Processing es fundamental para la visualización y análisis de los datos recopilados por el sistema Arduino. A continuación, se detallan los aspectos más relevantes de las clases principales:

PanelPrincipal.pde

Este archivo contiene la estructura principal del programa. Sus características más relevantes incluyen:

- Importación de la biblioteca Serial para la comunicación con Arduino.
- Declaración de variables globales, incluyendo un diccionario FloatDict para almacenar los valores de los sensores.
- Inicialización de los paneles para cada sensor en el método setup().
- Implementación del bucle principal draw() que actualiza y dibuja los paneles de los sensores.
- Manejo de eventos seriales para la comunicación con Arduino.

Clase Ball: Esta clase representa partículas móviles utilizadas en las visualizaciones. Características principales:

- Constructor: Inicializa la posición y velocidad de la partícula.
- Método move(): Actualiza la posición de la partícula y maneja rebotes en los bordes del panel.

- Método display(): Dibuja la partícula en la pantalla.

Clase Humedad: Encargada de la visualización de los datos de humedad. Aspectos destacados:

- Utiliza un sistema de partículas para representar visualmente la humedad.
- Implementa un diseño responsive que se ajusta al tamaño de la ventana.
- Muestra una imagen (sombrilla) y texto con el porcentaje de humedad.

Clase Humo: Visualiza los datos de CO2 (representado como "humo"). Características:

- Utiliza la clase PanelSensor para crear una visualización dinámica.
- Actualiza la visualización cada 2 segundos para reflejar cambios en los datos.
- Muestra el porcentaje de CO2 con texto sobre un fondo azul.

Clase Luz: Encargada de la visualización de los datos de luminosidad. Aspectos clave:

- Dibuja una estrella giratoria cuyo color varía según el porcentaje de luz.
- Muestra una imagen de nube y texto con el porcentaje de luz.
- Implementa un diseño responsive similar a las otras clases.

Clase PanelOla: Utilizada para crear una visualización de onda, posiblemente para representar datos de manera dinámica. Características:

- Genera una onda sinusoidal cuya amplitud, velocidad y periodo se ajustan según los datos recibidos.
- Permite actualizar los parámetros de la onda en tiempo real.

PanelSensor

Define la clase PanelSensor, que representa un panel genérico para visualizar datos de sensores. Características principales:

- Manejo de la posición y dimensiones del panel.
- Implementación de un sistema de partículas (bolas) para efectos visuales.
- Métodos para actualizar y mostrar el panel.

Particle y ParticleSystem

Estos archivos implementan un sistema de partículas utilizado para efectos visuales en los paneles de sensores. Características clave:

- La clase Particle maneja el comportamiento individual de cada partícula.
- ParticleSystem gestiona un conjunto de partículas, controlando su creación, actualización y eliminación.

Proximidad

Implementa la clase Proximidad, que es un panel específico para visualizar datos del sensor de proximidad. Características destacadas:

- Utiliza la clase PanelOla para crear una representación visual dinámica.
- Actualiza periódicamente los parámetros del panel basándose en los datos del sensor.
- Dibuja información textual sobre el porcentaje de proximidad.

El sistema está diseñado de manera modular, permitiendo fácil expansión y modificación. Cada tipo de sensor tiene su propia clase de panel, heredando funcionalidades comunes de la clase PanelSensor.

Temperatura

El código está estructurado en la clase Temperatura, que encapsula toda la funcionalidad necesaria para visualizar y actualizar los datos de temperatura.

- Adaptabilidad: El panel se ajusta automáticamente según su posición en la cuadrícula (fila 1 o 2).
- Visualización intuitiva: Utiliza una combinación de elementos gráficos (imagen de termómetro y barra de temperatura) y texto para representar la información.
- Responsividad: El tamaño del texto y los elementos gráficos se ajustan según las dimensiones del panel.

6. Funcionalidad del Sistema

6.1 Medición de Variables Meteorológicas

- **Temperatura y Humedad**: El sistema utiliza un sensor DHT11 para medir tanto la temperatura como la humedad. Los datos se obtienen a través de la librería DHT, que facilita la lectura de estos valores. La temperatura se mide en grados Celsius y la humedad en porcentaje. Estos datos se muestran en la pantalla LCD y se envían al software de visualización en la computadora.
- **Iluminación**: Para medir la cantidad de luz, el sistema emplea un sensor LDR (Resistencia Dependiente de la Luz) conectado a un pin analógico. El valor leído representa la intensidad de la luz ambiente, donde valores más altos indican mayor luminosidad. Esta información se muestra en la pantalla LCD y se transmite al software de visualización.
- **Calidad del Aire**: La medición de CO2 se realiza mediante un sensor MQ-135 conectado a otro pin analógico. Este sensor proporciona una lectura que indica la concentración relativa de CO2 en el aire. Los valores más altos sugieren una mayor concentración de CO2. Al igual que con los otros sensores, esta información se muestra en la pantalla LCD y se envía al software de visualización.
- **Proximidad**: Para detectar la proximidad, el sistema utiliza un sensor ultrasónico HC-SR04. Este sensor emite ondas sonoras y mide el tiempo que tardan en regresar después de rebotar en un objeto. Con esta información, el sistema calcula la distancia en centímetros. La medición de proximidad se actualiza constantemente y se muestra junto con los otros datos en la pantalla LCD y en el software de visualización.

6.2 Interacción con Botones

En proyectos de Arduino, es común la necesidad de almacenar, visualizar y recuperar datos. Una configuración típica para lograr esto involucra el uso de una pantalla LCD para la visualización y la memoria EEPROM integrada para el almacenamiento no volátil de datos. Mediante la implementación de tres botones, se puede controlar la interacción con estos componentes de la siguiente manera:

- Botón 1: Guarda los datos actuales en la memoria EEPROM.
- Botón 2:Muestra los datos actuales en la pantalla LCD.
- Botón 3: Recupera y muestra en la LCD los últimos datos guardados en la EEPROM.

Botón 1: Guardar Datos en la EEPROM

Operación: Este botón, al ser presionado, indica al Arduino que debe almacenar los datos actuales en la memoria EEPROM para preservarlos incluso después de un reinicio o apagado.

Implementación: Similar a la detección del Botón 2, al activarse, se emplean funciones de la librería `EEPROM.h` para escribir los datos en direcciones específicas de la EEPROM. Es esencial manejar adecuadamente la posición de escritura para evitar sobrescribir datos no deseados y considerar la limitación de ciclos de escritura.

Botón 2: Mostrar Datos en la LCD

Operación: Al presionar este botón, el Arduino toma los datos actuales (por ejemplo, lecturas de sensores) y los muestra en la pantalla LCD.

Implementación: Se debe detectar la pulsación del botón utilizando una entrada digital configurada en un teclado de membrana de 4x1. Al activarse, se ejecuta una función que actualiza la pantalla LCD con los datos relevantes.

Botón 3: Recuperar Datos de la EEPROM

Operación: Al presionar este botón, el sistema recupera los últimos datos guardados en la EEPROM y los muestra en la pantalla LCD.

Implementación: Tras detectar la pulsación, se utilizan funciones de la librería `EEPROM.h` para leer los datos desde las direcciones previamente utilizadas para el almacenamiento. Luego, estos datos se procesan y se muestran en la LCD.

6.3 Problema de Bloqueo en la Interacción con Botones y Sensores

En muchos proyectos con Arduino, se presenta un problema común cuando se utilizan botones y sensores de manera concurrente. Este problema surge cuando el microcontrolador está ocupado realizando lecturas de sensores o ejecutando otros procesos, lo que puede impedir que se detecten las pulsaciones de los botones de manera inmediata o eficiente.

Por ejemplo, si un sensor está siendo leído continuamente o si hay un proceso de largo plazo (como una comunicación serie o un cálculo complejo), el microcontrolador puede no responder rápidamente a la interacción del usuario a través de los botones. Esto sucede porque, en un ciclo de ejecución normal, Arduino procesa las tareas secuencialmente, lo que significa que debe completar una tarea antes de pasar a la siguiente. Si la tarea de leer un sensor toma mucho tiempo,

cualquier pulsación de botón durante ese período podría ser ignorada o respondida con retraso, generando una experiencia de usuario insatisfactoria.

Este problema es especialmente relevante en proyectos donde la lectura de sensores es crítica y debe realizarse de manera continua, como en sistemas de monitoreo ambiental, donde se espera que los botones respondan instantáneamente para guardar, mostrar o recuperar datos

6.4 Memoria EEPROM en Arduino:

Pantalla LCD:

La pantalla LCD (Liquid Crystal Display) se utiliza para presentar información al usuario. En este contexto, muestra datos actuales o recuperados de la EEPROM según la interacción del usuario mediante los botones.

La EEPROM (Electrically Erasable Programmable Read-Only Memory) es una memoria no volátil integrada en las placas Arduino que permite almacenar datos incluso cuando la alimentación eléctrica se interrumpe. Características clave de la EEPROM en Arduino incluyen:

Persistencia de Datos: Los datos almacenados en la EEPROM permanecen intactos después de apagar el dispositivo.

Ciclos de Escritura Limitados: La EEPROM tiene un número finito de ciclos de escritura (generalmente alrededor de 100,000), por lo que se recomienda minimizar las operaciones de escritura para prolongar su vida útil.

Acceso Byte a Byte: La EEPROM se organiza en bytes, lo que permite leer y escribir datos en posiciones específicas.

7. Capas del Stack IoT Framework Implementadas

En esta fase inicial, nos enfocamos en las capas fundamentales del IoT Stack Framework para crear la base de nuestro sistema de control de acceso vehicular con análisis meteorológico. Las capas implementadas son:

1. Hardware:
    - Microcontrolador Arduino: Actúa como el cerebro del sistema, procesando los datos de los sensores.
    - Sensores:
        - DHT11: Mide temperatura y humedad del ambiente.
        - HC-SR04: Sensor ultrasónico para detectar proximidad.
        - Módulo de fotoresistencia digital: Mide la intensidad de luz ambiental.
        - MQ135: Detecta la concentración de CO2 en el aire.
    - Pantalla LCD con I2C: Permite la visualización de la información recopilada por los sensores, facilitando la interacción del usuario con el sistema.
    - Botones: Permiten la interacción del usuario con el sistema.
2. Software:
    - **Arduino IDE:** Se utiliza para programar el microcontrolador, permitiendo la lectura de los sensores, el procesamiento de datos, y la actualización de la pantalla LCD con I2C. El código .ino está diseñado para gestionar las operaciones básicas del sistema, como la adquisición de datos de los sensores y la interacción con los elementos de hardware.
    - **Processing IDE:** Se emplea para desarrollar una interfaz gráfica en la computadora que visualiza en tiempo real los datos obtenidos por el Arduino. El código en Processing está estructurado para recibir datos a través de la conexión serial y presentarlos visualmente, facilitando la interpretación de la información.
    - Dashboard en Processing para visualización y análisis de datos.
3. Comunicación:
    - Establece la comunicación entre el Arduino y la computadora, permitiendo la transferencia de datos en tiempo real.

En esta fase, no se implementan las capas de plataforma en la nube (Cloud Platform) ni aplicaciones en la nube (Cloud Applications), ya que el enfoque está en la recolección de datos locales y su visualización en tiempo real.

Esta implementación sienta las bases para futuras expansiones del sistema, permitiendo la recolección, procesamiento y visualización de datos meteorológicos y de proximidad, esenciales para el control de acceso vehicular inteligente.

8. Conclusiones

El desarrollo del sistema avanzado de control de acceso vehicular integrado con una estación meteorológica IoT ha demostrado ser una solución efectiva y robusta para optimizar tanto la seguridad como la eficiencia en la gestión de accesos vehiculares. Este proyecto, articulado en torno a la implementación de un microcontrolador Arduino y diversos sensores, ha permitido la captura precisa y en tiempo real de variables ambientales críticas como temperatura, humedad, iluminación y concentración de CO2, además de la detección de proximidad mediante un sensor ultrasónico.

La integración de una pantalla LCD con I2C y una plataforma de visualización desarrollada en Processing ha facilitado la presentación clara y accesible de los datos, tanto para la visualización inmediata en el sitio como para un análisis más detallado en una estación de control. El sistema también ha sido diseñado para incluir una cola de mensajes basada en el protocolo MQTT, lo que asegura una gestión eficiente y escalable de los datos, sentando las bases para futuras expansiones y mejoras del sistema.

En términos de software, la implementación de un código modular y eficiente en el Arduino IDE, complementado con la potente capacidad de visualización del Processing IDE, ha sido clave para el éxito del proyecto. La estructura del código ha permitido no solo la captura y procesamiento de datos en tiempo real, sino también su almacenamiento histórico en la memoria EEPROM, lo que proporciona una valiosa capacidad de recuperación de datos para análisis posteriores.

El enfoque por fases en el desarrollo de este proyecto ha permitido una implementación organizada y controlada, comenzando con la configuración básica del hardware y software necesarios para la recopilación y visualización de datos. Esta estrategia no solo ha garantizado la estabilidad y funcionalidad del sistema desde sus etapas iniciales, sino que también ha proporcionado una base sólida sobre la cual se pueden construir futuras capacidades, como la integración con plataformas en la nube para análisis de datos a gran escala y el desarrollo de aplicaciones de monitoreo remoto.

9. Anexos:

Diagrama de flujo del archivo principal de que maneja la lógica del proyecto:

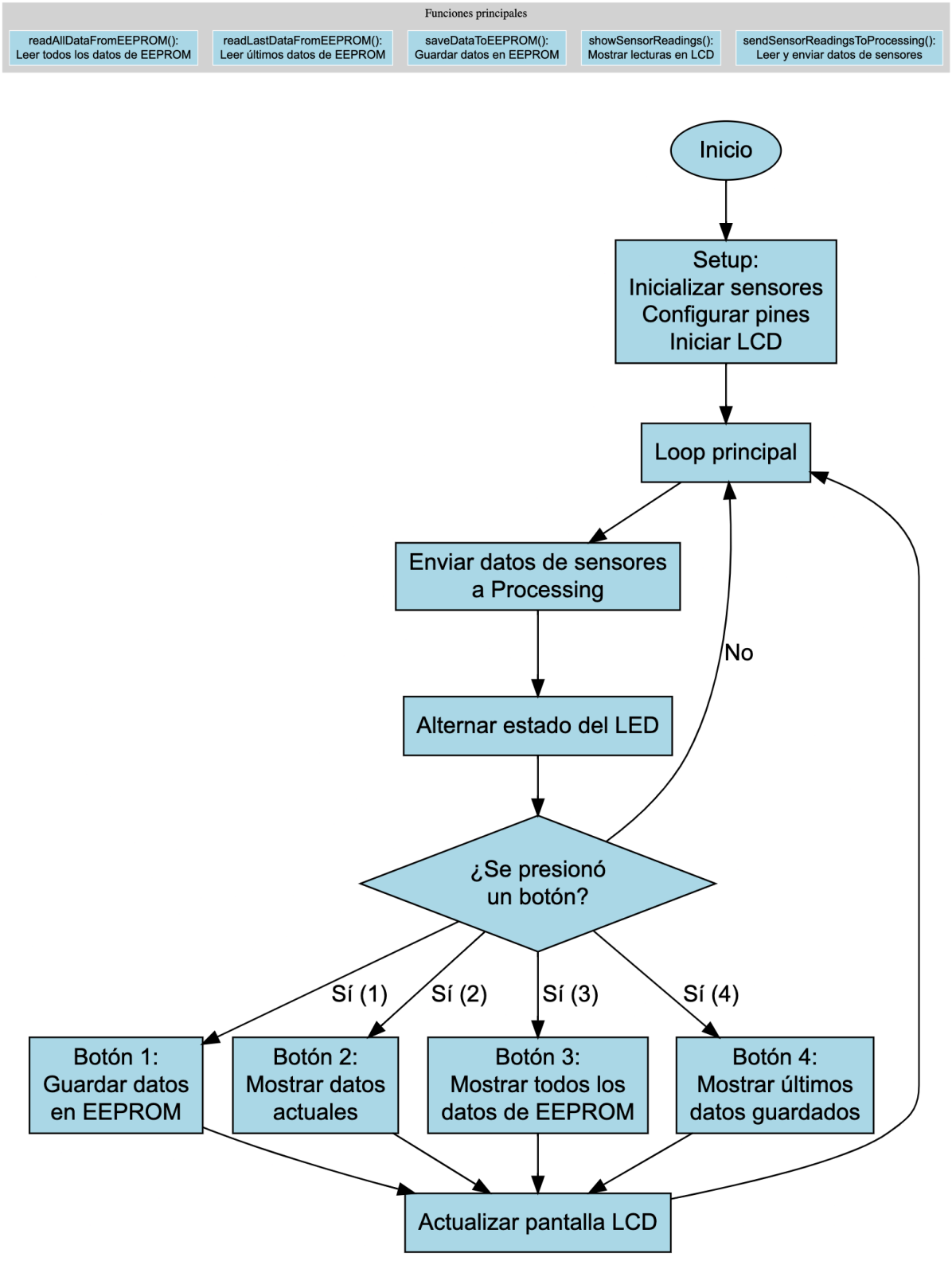

Figura 2.

Diagrama de clases:

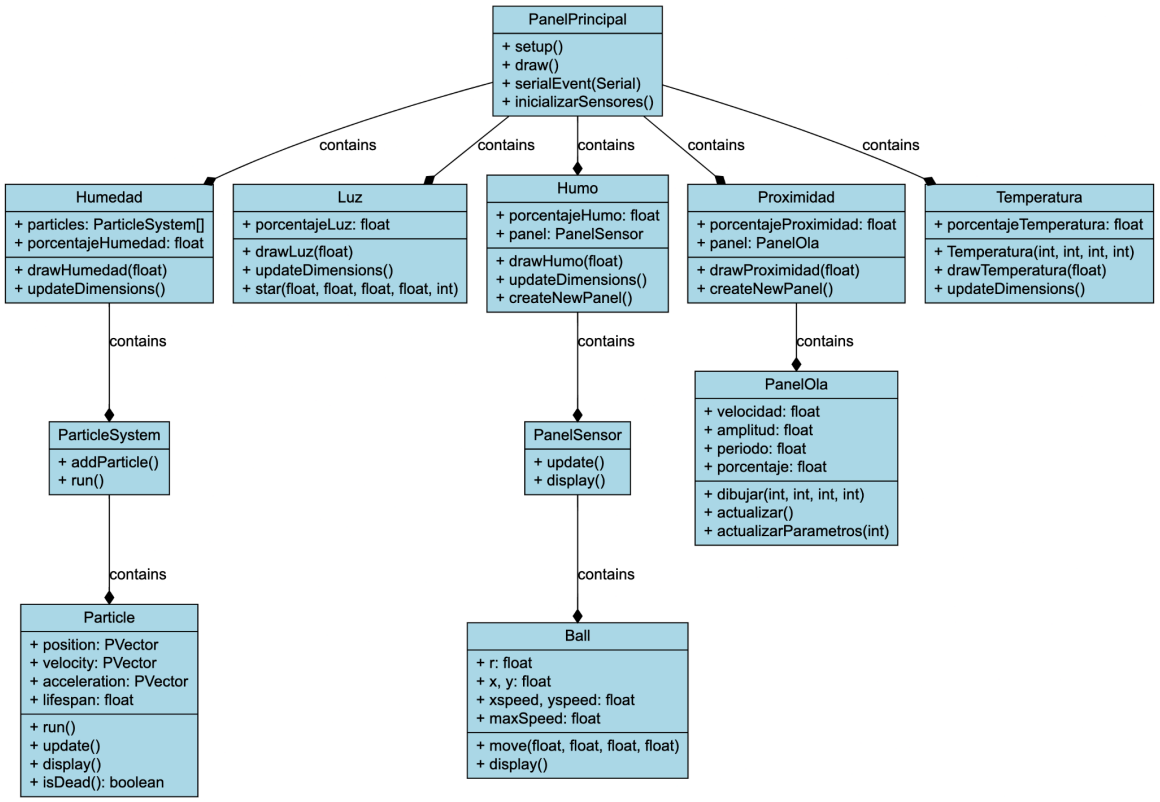

Figura 2.

Costos:

Descripción	Unidades	Precio Unidad	Total
Arduino Mega	1	0.00	0.00
Sensor de Temperatura y Humedad DHT1	1	34.00	34.00
Sensor de Calidad de aire MQ-135	1	39.00	39.00
Fotoresistencia analógicos Digitales	1	10.00	10.00
SENSOR ULTRASÓNICO HC-SR04	1	29.00	29.00
PANTALLA LCD	1	45.00	45.00
Jumpers 20cm	10	1.25	12.50
Cartón Chip	1	23.00	23.00
Silicon Caliente	6	2.00	12.00
Papel Cartulina	2	6.00	12.00
			216.50

Maqueta:

imagen 1.

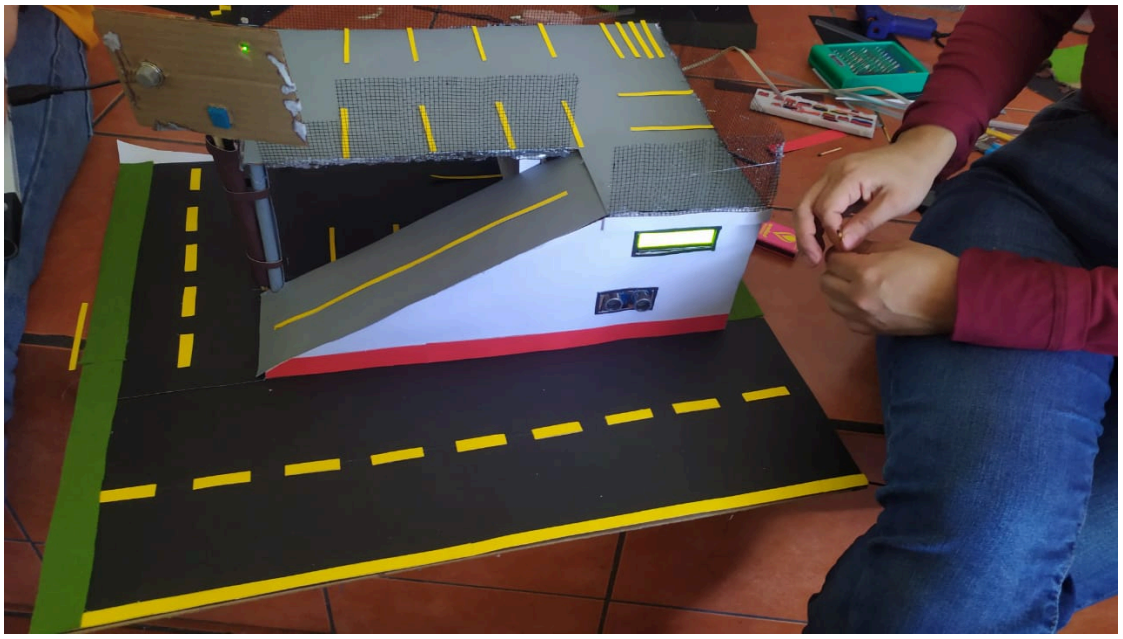

imagen 2.

Dashboard en Processing para visualización y análisis de datos:

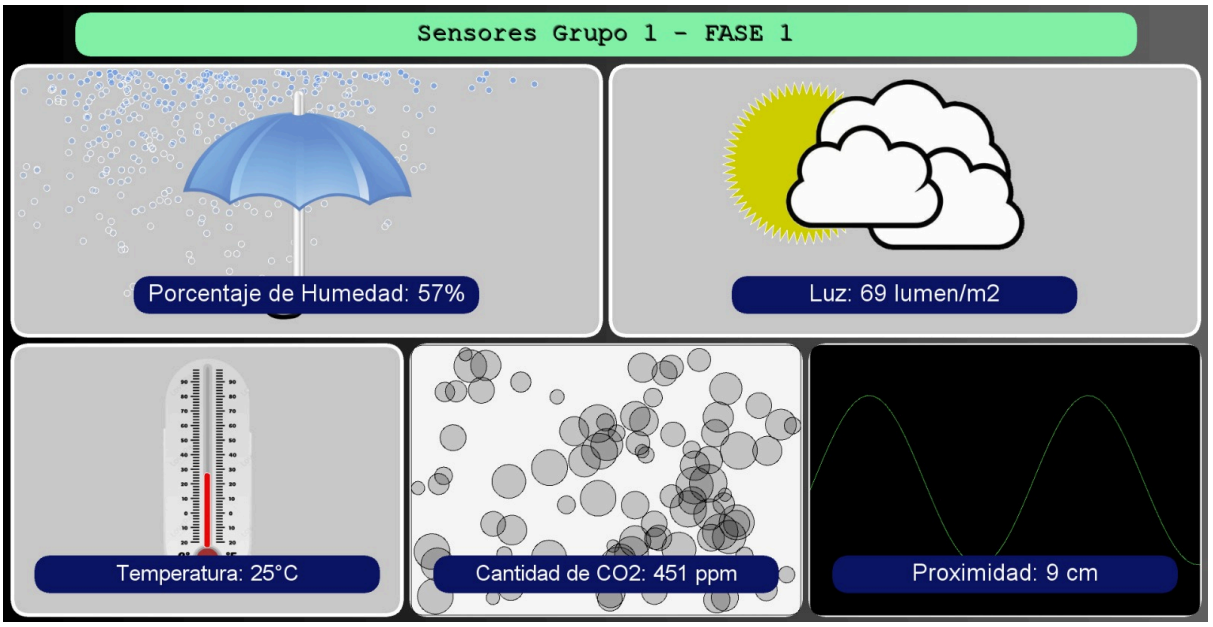

imagen 3

10. Referencias

Administrador. (2018, February 21). Arduino EEPROM y sus funciones. Retrieved August 17, 2024, from HeTPro-Tutoriales website: https://hetpro-store.com/TUTORIALES/arduino-eeprom/#:~:text=Arduino%20mega%202560.,cuando%20la%20tarjeta%20es%20apagada.

Cómo utilizar el DHT11 para medir la temperatura y humedad con Arduino. (2017, March 21). Retrieved August 17, 2024, from Programarfacil Arduino y Home Assistant website: https://programarfacil.com/blog/arduino-blog/sensor-dht11-temperatura-humedad-arduino/

Documentación de Processing.js | Khan Academy. (2023). Retrieved August 17, 2024, from Khan Academy website: https://es.khanacademy.org/computing/computer-programming/pjs-documentation

Instructables. (2018, December 5). Graphing With Processing. Retrieved August 17, 2024, from Instructables website: https://www.instructables.com/Graphing-With-Processing/

Memoria EEPROM de Arduino. (2021, December 14). Retrieved August 17, 2024, from Programarfacil Arduino y Home Assistant website: https://programarfacil.com/blog/arduino-blog/eeprom-arduino/

MQ-135 Detector de Calidad de Aire - UNIT Electronics. (2024, August 16). Retrieved August 17, 2024, from UNIT Electronics website: https://uelectronics.com/producto/mq-135-modulo-detector-de-calidad-de-aire/#:~:text=Sensor%20MQ%2D135&text=Una%20vez%20se%20calienta%20el,corriente%20a%20trav%C3%A9s%20de%20este.

Naylamp Mechatronics - Perú. (2022). Tutorial de Arduino y sensor ultrasónico HC-SR04. Retrieved August 17, 2024, from Naylamp Mechatronics - Perú website: https://naylampmechatronics.com/blog/10_tutorial-de-arduino-y-sensor-ultrasonico-hc-sr04.html

SYDLE. (2022, March 22). ¿Qué es Internet de las Cosas? Aprende todo sobre IoT. Retrieved August 17, 2024, from Blog SYDLE website: https://www.sydle.com/es/blog/internet-de-las-cosas-6239c79c3bbdd676577a1e76

Reference. (2024). Retrieved August 17, 2024, from Processing website: https://processing.org/reference/

Tecnología, D., les, J., & Juan. (n.d.). TRABAJO PRÁCTICO 1: COMPORTAMIENTO DE UNA LDR COMO SENSOR DE LUZ A) DISEÑO DE LA EXPERIENCIA 1. -OBJETIVOS. Retrieved from https://www.iesjorgejuan.es/sites/default/files/dp8/departamentos/tecnologia/materiales/sensor_LDR.pdf

Processing: Diseño de Interfaces. (n.d.). Retrieved from https://www.paulrosero-montalvo.com/gallery/secap6.2.pdf

¿Qué es IoT y cómo funciona? | SAP. (2021). Retrieved August 17, 2024, from SAP website: https://www.sap.com/latinamerica/products/artificial-intelligence/what-is-iot.html

¿Qué es IoT? - Explicación del Internet de las cosas - AWS. (2021). Retrieved August 17, 2024, from Amazon Web Services, Inc. website: https://aws.amazon.com/es/what-is/iot/#:~:text=con%20AWS%20IoT-,%C2%BFQu%C3%A9%20es%20el%20Internet%20de%20las%20cosas%20(IoT)%3F,como%20entre%20los%20propios%20dispositivos.

36- Trabajar con memoria EEPROM Arduino - **Electrodaddy**. (2021, February 27). Retrieved August 17, 2024, from **Electrodaddy** website: https://electrodaddy.com/trabajar-con-memoria-eeprom-arduino/

(2024). Retrieved August 17, 2024, from Arduino.cc website: https://docs.arduino.cc/learn/built-in-libraries/eeprom/