

Implementación de internet de las cosas



Tecnológico
de Monterrey

Fecha de entrega: 18/10/2023

Alumnos

| | |
|-----------------------|-----------|
| Carlos Corzo Haua | A01782695 |
| Rafael Ison Shaooli | A01781837 |
| Patricio Tamayo Gómez | A01026184 |
| Maximiliano Gallardo | A01783128 |
| Carlos R. Maldonado | A01029426 |

Índice

| | |
|--|-----------|
| Abstract..... | 3 |
| Introducción..... | 3 |
| Materiales..... | 4 |
| Componentes Principales..... | 4 |
| ESP32..... | 4 |
| DHT-11..... | 6 |
| IR 08-H..... | 6 |
| MQ 135..... | 7 |
| MH-Real-Time Clock Module - 2..... | 9 |
| LCD 2x16..... | 10 |
| Características de las LCD:..... | 10 |
| ¿Cómo comunicarse con una pantalla LCD?..... | 11 |
| Componentes Secundarios..... | 11 |
| Protoboard..... | 11 |
| Jumpers M-M / M-H / H-H..... | 13 |
| Jumpers Macho-Macho:..... | 13 |
| Jumpers Macho-Hembra:..... | 13 |
| Jumpers Hembra-Hembra:..... | 14 |
| Desarrollo..... | 15 |
| Prototipo..... | 16 |
| Base de Datos en la Nube..... | 18 |
| Free my SQL..... | 18 |
| API..... | 19 |
| Arquitectura..... | 21 |
| Visualización de Datos..... | 22 |
| Lenguajes de Código Utilizados..... | 23 |
| Enlaces Externos..... | 24 |
| Resultados y Conclusión..... | 24 |
| Conclusiones Personales..... | 26 |
| Carlos Corzo..... | 26 |
| Carlos Rubén Maldonado Barreda..... | 27 |
| Maximiliando Gallardo..... | 28 |
| Patricio Tamayo Gómez..... | 29 |
| Rafael Ison Shaooli..... | 30 |
| Bibliografías..... | 31 |
| Anexo..... | 32 |
| Códigos..... | 32 |
| Arduino IDE..... | 32 |
| SQL-API..... | 35 |
| Dashboard..... | 37 |

Reporte Final Situación Problema

Abstract

El avance en la capacidad de procesamiento, la reducción de las dimensiones de dispositivos digitales y la optimización en el consumo de energía han habilitado la integración de sensores en diversos procesos, impulsando la transformación digital de la sociedad. La conectividad de datos asequible ha permitido la interconexión de estos dispositivos a internet, brindando información en tiempo real desde los sensores.

Introducción

El avance tecnológico ha permitido la integración de sensores en diversos procesos, impulsando la transformación digital de la sociedad. Este reporte aborda el desarrollo de un sistema de Internet de las Cosas (IoT) basado en microcontroladores conectados a internet con el propósito de adquirir datos a través de sensores y visualizarlos en una base de datos en la nube.

Las etapas del proyecto se distribuyen en cinco fases paralelas con las actividades de aprendizaje correspondientes. La primera etapa abarca la definición del proyecto y la evaluación de recursos y conocimientos necesarios. En la segunda etapa, se realiza la conexión y programación de sensores, diseñando circuitos y configurando el control de actuadores en función de los datos obtenidos. La tercera fase involucra el diseño de una base de datos y la conectividad con la interfaz de Arduino y Node MCU, asegurando la escritura y lectura en tiempo real de datos.

La cuarta etapa integra los sistemas individuales para pruebas iniciales y gestiona el análisis de datos y visualización de indicadores. La última fase implica la validación exhaustiva del sistema de IoT y la preparación de una presentación final.

Este proyecto ilustra la aplicación práctica de IoT en la recopilación de datos y control de dispositivos, destacando su relevancia en la era digital actual.

Materiales

La sección de materiales es un componente fundamental en el desarrollo del proyecto de Internet de las Cosas (IoT) basado en microcontroladores conectados a Internet para la adquisición de datos y el control de actuadores. Esta sección proporciona una visión completa de los recursos y herramientas esenciales que se utilizarán a lo largo de las diversas etapas del proyecto.

Los materiales desempeñan un papel crucial en la viabilidad y éxito de la implementación de un sistema de IoT, y su selección cuidadosa es fundamental para cumplir con los requerimientos específicos del proyecto. Los elementos a utilizar incluyen microcontroladores, sensores, componentes electrónicos, plataformas de desarrollo, software de programación y herramientas de conectividad.

En esta sección, exploraremos detalladamente los materiales que se emplearán para llevar a cabo cada una de las etapas del proyecto. Además, se describirá cómo estos materiales se integran para lograr la recopilación de datos en tiempo real, el control de dispositivos y la visualización de indicadores clave.

El conocimiento y comprensión de los materiales disponibles son esenciales para que los participantes puedan abordar eficazmente los desafíos planteados en las etapas subsiguientes del proyecto. Así, la sección de materiales sirve como la base sobre la cual se construirá y desarrollará un sistema de IoT efectivo y funcional por lo que presentamos los materiales en dos secciones, principales y secundarios.

Componentes Principales

ESP32

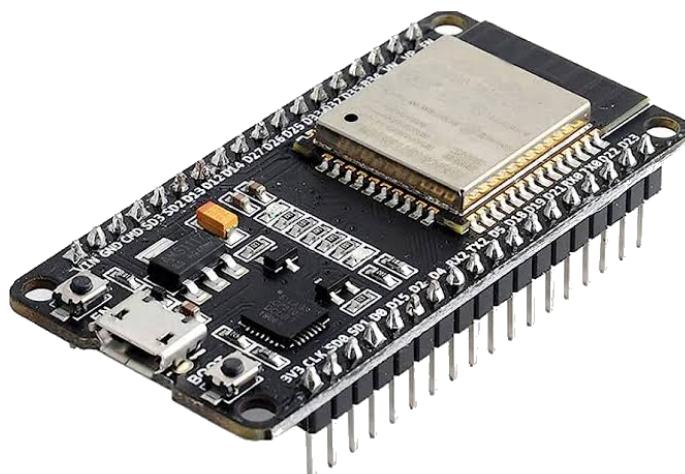


Figura 1. Microcontrolador dual mode ESP32

TC 1004B.401
Rubén Raya Delgado

- Microcontrolador de Doble Núcleo: El ESP32 cuenta con dos núcleos de procesamiento, lo que permite la ejecución simultánea de tareas, ideal para aplicaciones multitarea y de tiempo real.
- Wi-Fi Integrado: Proporciona conectividad Wi-Fi para la comunicación inalámbrica, lo que lo hace adecuado para aplicaciones IoT.
- Bluetooth: El ESP32 incluye soporte para Bluetooth clásico y Bluetooth de baja energía (BLE), lo que lo hace versátil para aplicaciones de comunicación inalámbrica.
- Conexión a Internet y Servidores Web: Puede conectarse a Internet y actuar como un servidor web, lo que facilita la creación de aplicaciones IoT.
- GPIO: Ofrece una amplia variedad de pines GPIO para interactuar con sensores, actuadores y dispositivos externos.
- Periféricos y Sensores Integrados: El ESP32 tiene una amplia gama de periféricos, como sensores de temperatura, acelerómetros, y más.
- Soporte para MicroSD: Puede utilizar tarjetas microSD para el almacenamiento de datos.
- Interfaz UART, SPI, I2C: Proporciona interfaces de comunicación serie y bus para conectarse a una variedad de dispositivos externos.
- Energéticamente Eficiente: Diseñado para ser eficiente en términos de consumo de energía, lo que lo hace adecuado para aplicaciones con restricciones de energía.
- Desarrollo en Arduino y Espressif IDF: Puede programarse con el IDE de Arduino y el Espressif IoT Development Framework (ESP-IDF).
- Seguridad: Ofrece características de seguridad, como el cifrado de datos, que son cruciales en aplicaciones IoT.
- Compatibilidad con OTA (Actualización por el Aire): Permite la actualización remota del firmware de manera inalámbrica, lo que es esencial en dispositivos IoT.
- Amplia Comunidad y Documentación: Cuenta con una comunidad activa de desarrolladores y una documentación sólida para ayudar en el desarrollo de proyectos.
- Versatilidad de Aplicaciones: Es adecuado para una amplia gama de aplicaciones, incluyendo automatización del hogar, dispositivos de seguimiento, monitoreo ambiental y más.

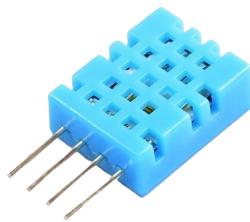
DHT-11

Figura 2. Sensor de humedad y temperatura DHT-11

El DHT-11 es un sensor digital de bajo costo utilizado para medir la temperatura y la humedad en el ambiente. Este sensor está diseñado para proporcionar mediciones precisas en un rango de temperatura de -20°C a 50°C y un rango de humedad relativa del 20% al 80%. El DHT-11 utiliza un elemento sensible a la humedad y un termistor para medir estas variables y luego convierte las mediciones en señales digitales que pueden ser fácilmente interpretadas por microcontroladores o sistemas electrónicos. Este sensor es comúnmente utilizado en proyectos de electrónica, domótica y aplicaciones relacionadas con el monitoreo y control de las condiciones ambientales.

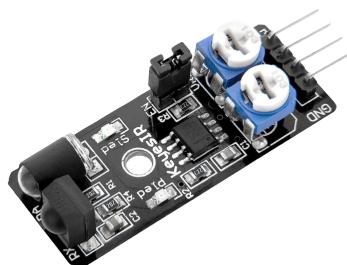
IR 08-H

Figura 3. Sensor de proximidad IR 08-H

El IR 08-H es un detector de obstáculos infrarrojo que identifica la presencia de un objeto al detectar la reflexión de la luz infrarroja que este emite. La elección de la luz infrarroja se debe a que no es visible para el ojo humano.

Estos sensores son relativamente simples en su diseño. Incluyen un LED emisor de luz infrarroja y un fotodiodo (como el tipo BPV10NF o similar) que recibe la luz reflejada por un posible obstáculo.

Los detectores de obstáculos generalmente vienen con una placa de medición estándar equipada con el comparador LM393, que convierte la lectura en un valor digital cuando se supera un umbral específico. Este umbral se puede ajustar a través de un potenciómetro en la placa.

Estos sensores son efectivos a cortas distancias, generalmente en el rango de 5 a 20 mm. Sin embargo, la cantidad de luz infrarroja recibida varía según el color, material, forma y posición del objeto, lo que limita su precisión para medir la distancia al obstáculo.

A pesar de esta limitación, se utilizan ampliamente para detectar obstáculos en pequeños vehículos o robots debido a su bajo costo. A menudo se colocan alrededor del perímetro para detectar obstáculos en múltiples direcciones.

Estos sensores también encuentran aplicaciones en la detección de la presencia de objetos en áreas específicas, determinar si una puerta está abierta o cerrada, o si una máquina ha alcanzado un punto específico en su recorrido.

El montaje es sencillo, ya que el módulo se alimenta a través de los pines Vcc y GND, que se conectan respectivamente a las salidas de 5V y GND en una placa Arduino.

MQ 135



Figura 4. Sensor de gases MQ-135

El sensor MQ-135 es un sensor de gas que se utiliza para detectar varios tipos de gases, incluyendo gases tóxicos como el monóxido de carbono, el dióxido de azufre, el amoníaco y otros gases dañinos para la salud. A continuación, te proporciono un resumen detallado de cómo funciona este sensor:

- Elemento Sensible: En el corazón del sensor MQ-135 hay un elemento sensible que consiste en un óxido metálico semiconductor. La composición de este óxido varía según el modelo del sensor y se selecciona específicamente para ser sensible a ciertos gases.
- Calentamiento: Para que el sensor funcione correctamente, necesita estar a una temperatura constante. Por lo tanto, incorpora un elemento calefactor que eleva su temperatura a un nivel óptimo. Este calentamiento constante es esencial para la estabilidad y precisión del sensor.
- Detección de Gases: Cuando el sensor está encendido y funcionando, el óxido metálico sensible reacciona químicamente con los gases presentes en el entorno. La

reacción con un gas en particular cambia la resistencia eléctrica del elemento sensible. La magnitud de este cambio en la resistencia está relacionada con la concentración del gas en el aire.

- Medición de Resistencia: El sensor MQ-135 mide continuamente la resistencia del elemento sensible. Cuando la concentración de un gas objetivo en el ambiente aumenta, la resistencia del sensor también aumenta. Este cambio en la resistencia se convierte en una señal eléctrica que puede ser procesada y evaluada.
- Calibración: Para interpretar con precisión la concentración de gas, el sensor MQ-135 requiere una calibración específica para el tipo de gas que se está midiendo. Esto significa que debes conocer y establecer una relación entre la resistencia medida y la concentración del gas de interés. Esta calibración es fundamental para obtener mediciones precisas.
- Salida Analógica: El sensor MQ-135 generalmente proporciona una señal analógica, que puede ser directamente relacionada con la concentración del gas detectado. Esta señal puede ser conectada a un microcontrolador, como Arduino, para su procesamiento y visualización.
- Límite de Detección: Cabe mencionar que el MQ-135 tiene un límite de detección específico para cada gas. No es igualmente sensible a todos los gases, por lo que es importante conocer las características del sensor y su sensibilidad a diferentes gases.

En resumen, el sensor MQ-135 funciona midiendo el cambio en la resistencia eléctrica de un elemento sensible cuando reacciona con gases específicos en el ambiente. La interpretación precisa de esta resistencia requiere una calibración adecuada, y la salida del sensor se utiliza para determinar la concentración del gas detectado en el aire.

MH-Real-Time Clock Module - 2



Figura 5. Real time clock module

El módulo MH-Real-Time Clock - 2 se basa en el integrado DS1302, que contiene un reloj/calendario en tiempo real y 31 bytes de RAM estática. Este modelo integrado es similar al DS1307, pero con algunas diferencias:

- El DS1302 utiliza una interfaz SPI, mientras que el DS1307 usa una interfaz I2C.
- El DS1302 puede cargar la batería, a diferencia del DS1307.
- El DS1307 tiene una salida de onda cuadrada programable.

El DS1302 es un chip de bajo consumo diseñado para mantener un registro preciso del tiempo, la fecha y otros datos relacionados. Se comunica con el microprocesador a través de una interfaz serial y proporciona información sobre segundos, minutos, horas, día, fecha, mes y año. Para la interfaz, se requieren solo tres cables además de la alimentación: CE (RST), I/O (línea de datos) y SCLK (reloj serial). Los datos pueden transferirse hacia y desde el reloj/RAM de a 1 byte por vez o enviar hasta 31 bytes a la vez.

El DS1302 está diseñado para operar con un consumo de energía muy bajo, almacenando la información del reloj y los datos con menos de 1 μ W. Tiene pines de alimentación duales, uno principal (Pin #1 y #4) y otro para la batería de respaldo (Pin #8 y #4), que utiliza una batería no recargable de 260mAh. El tiempo de retención teórico de los datos es de más de 10 años. En una aplicación típica, la alimentación principal proviene de Arduino, y la de respaldo de una batería tipo CR2032.

En el diagrama del circuito del Módulo de Reloj en Tiempo Real DS1302, VCC se configura para aceptar la alimentación principal, que suele ser de 3.3V pero también puede ser de 5V. No se recomienda aplicar más de 7V, ya que podría dañar el módulo. VBAT se utiliza como energía de respaldo, suministrada por la batería CR2032 de 3.3V.

El cristal de cuarzo se coloca entre los pines #2 y #3. Los pines #5, #6 y #7 se usan para la comunicación de datos entre el módulo y el microcontrolador, y se conectan al pin CE, también conocido como RST.

Antes de cargar el programa, debes instalar la biblioteca adecuada, que se puede descargar desde el sitio web de Henning Karlsen, DS1302. El programa de Arduino verifica el funcionamiento del DS1302 y muestra la fecha y la hora actual en el monitor serie o en una pantalla de 16x2 con interfaz I2C. En la primera ejecución, debes configurar la fecha y la hora, pero una vez hecho, esta información se almacenará en la memoria del módulo respaldada por la batería de respaldo.

LCD 2x16



Figura 6. Display LCD 16x2

Las pantallas LCD, o "Liquid Crystal Displays" (Pantallas de Cristal Líquido), son ampliamente utilizadas en la industria, proyectos escolares y aplicaciones comerciales debido a su atractivo visual y versatilidad. Aunque a menudo asociamos las siglas "LCD" con las pantallas de televisión o las pantallas de computadoras, en realidad, las pantallas LCD se utilizan en una amplia variedad de dispositivos, desde relojes y calculadoras hasta electrodomésticos e impresoras.

Características de las LCD:

1. Tamaño: El tamaño de una pantalla LCD se mide generalmente en pulgadas a lo largo de su diagonal. Por ejemplo, una pantalla "LCD 16x2" significa que puede mostrar simultáneamente 16 caracteres en dos filas de 8 caracteres cada una.
2. Resolución: La resolución se expresa en términos de la cantidad de píxeles en la pantalla, generalmente en términos de ancho x alto. Por ejemplo, las pantallas HD tienen una resolución de 1920x1080. La elección de la resolución adecuada depende de la aplicación y la necesidad de detalle.
3. Brillo: La luminosidad de la pantalla es crucial, ya que algunas aplicaciones requieren una pantalla más brillante para una mejor visibilidad. La mayoría de las pantallas LCD cuentan con una luz de fondo y la capacidad de ajustar su brillo.
4. Iluminación (CCFL y LED): La iluminación de las pantallas LCD puede ser proporcionada por lámparas fluorescentes CCFL o luces LED. Las luces LED son más eficientes y tienen una vida útil más larga en comparación con las lámparas CCFL.
5. Contraste: El contraste es la relación entre la parte más brillante y la más oscura de la pantalla, lo que afecta la calidad de la imagen.
6. Ángulo de visión: Es el rango de ángulo desde el cual un usuario puede ver la pantalla sin perder calidad. Un buen ángulo de visión es esencial para asegurar que la información se vea claramente desde varios ángulos.

7. Número de caracteres: Las pantallas LCD vienen en varios tamaños, lo que afecta la cantidad de caracteres que pueden mostrarse. Algunos tamaños estándar incluyen 16x2, 20x4 y 8x2.

¿Cómo comunicarse con una pantalla LCD?

Comunicación en paralelo: Este método implica una comunicación directa con Arduino o el microcontrolador. Los datos se envían a la pantalla a través de un conjunto de pines (generalmente de 0 a 7 o de 0 a 4) en un proceso de envío rápido.

Componentes Secundarios

Protoboard

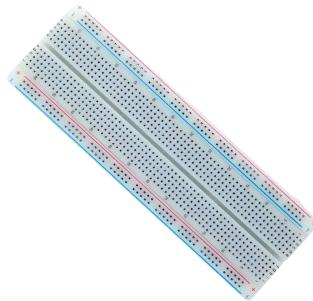


Figura 7. Protoboard

La protoboard, o "tableta protoboard", es una herramienta fundamental en electrónica que se utiliza para crear y ensayar circuitos electrónicos de manera temporal y sin necesidad de soldadura. A continuación, se detalla su funcionamiento:

- Material Aislante: La protoboard está hecha de un material aislante, generalmente plástico, que evita que los componentes y conexiones eléctricas entren en contacto directo con superficies conductoras, lo que previene cortocircuitos y asegura la seguridad de los componentes electrónicos.
- Orificios Conducidos: En la parte superior de la protoboard, se encuentran una serie de orificios organizados en filas y columnas. Estos orificios están conectados internamente por tiras de metal conductor que siguen un patrón específico. Estas tiras actúan como conexiones eléctricas entre los orificios.
- Inserción de Componentes: Los componentes electrónicos, como resistencias, transistores, diodos, cables y otros elementos, se pueden insertar directamente en los orificios de la protoboard. Cuando se inserta un componente en un orificio, sus terminales hacen contacto con las tiras conductoras internas, lo que establece una conexión eléctrica.

- Montaje sin Soldadura: La ventaja principal de una protoboard es que permite montar circuitos electrónicos sin la necesidad de soldar los componentes. Esto facilita la creación de prototipos y pruebas rápidas, ya que los componentes se pueden reutilizar en diferentes configuraciones.
- Conexiones Flexibles: Los cables de conexión se utilizan para conectar los componentes entre sí. Estos cables se insertan en los orificios y se conectan a través de las tiras conductoras internas, permitiendo una conexión flexible y reconfigurable de los componentes.
- Facilidad de Modificación: La protoboard permite realizar modificaciones y correcciones en el circuito de manera sencilla. Si es necesario cambiar la ubicación de un componente o realizar ajustes en la conexión, se pueden hacer ajustes de manera inmediata sin dañar los componentes.
- Experimentación y Prototipos: La protoboard es una herramienta valiosa para la experimentación y desarrollo de prototipos en electrónica. Los diseñadores y estudiantes pueden probar y evaluar diferentes configuraciones de circuitos antes de realizar una implementación permanente.

En resumen, una protoboard es una herramienta esencial para la creación y prueba de circuitos electrónicos temporales. Permite la inserción de componentes y la conexión de forma sencilla y flexible sin soldadura, lo que la convierte en una elección popular para la experimentación y prototipos en el campo de la electrónica.

Jumpers M-M / M-H / H-H

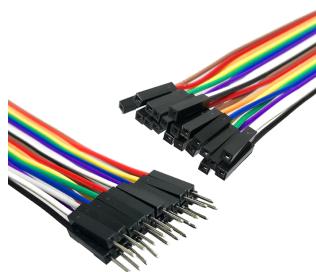


Figura 8. Jumpers

Los jumpers Macho-Macho, Macho-Hembra y Hembra-Hembra son componentes esenciales en proyectos de Arduino, protoboard y sensores, ya que facilitan la conexión de dispositivos electrónicos. A continuación, describo y explico el funcionamiento de cada tipo de cable:

Jumpers Macho-Macho:

- Descripción: Estos cables tienen un conector macho en ambos extremos. El conector macho tiene pinos o clavijas metálicas que encajan en los conectores hembra de los dispositivos, como sensores, placas de desarrollo Arduino o Raspberry Pi.
- Funcionamiento: Los cables macho-macho se utilizan para conectar componentes que tienen conectores hembra, como sensores y placas de desarrollo. Los pinos macho en un extremo se insertan en los conectores hembra del dispositivo de destino, estableciendo una conexión eléctrica y de datos. Esto permite la transferencia de señales, energía y datos entre los dispositivos conectados.

Jumpers Macho-Hembra:

- Descripción: Estos cables tienen un conector macho en un extremo y un conector hembra en el otro extremo. El conector macho se utiliza para conectar a dispositivos con conectores hembra, mientras que el conector hembra se utiliza para conectar a dispositivos con conectores macho.
- Funcionamiento: Los cables macho-hembra actúan como una extensión o adaptador. El conector macho se conecta a un dispositivo con conector hembra, como un sensor, y el conector hembra se conecta a otro dispositivo con conector macho, como una placa Arduino. Esto permite alargar la distancia entre los componentes o adaptar conectores de diferentes tipos, lo que resulta útil para organizar y extender las conexiones en un proyecto.

Jumpers Hembra-Hembra:

- Descripción: Estos cables tienen un conector hembra en ambos extremos. Se utilizan para conectar dispositivos que tienen conectores hembra, permitiendo una extensión o conexión entre ellos.
- Funcionamiento: Los cables hembra-hembra son útiles cuando se necesita una extensión directa de una conexión entre dispositivos con conectores hembra. Por ejemplo, se pueden utilizar para conectar dos sensores o dos placas de desarrollo que tienen conectores hembra sin la necesidad de adaptadores adicionales.

En resumen, los cables Macho-Macho se utilizan para conectar dispositivos con conectores hembra, los cables Macho-Hembra actúan como adaptadores o extensiones entre dispositivos con diferentes conectores, y los cables Hembra-Hembra permiten la extensión de conexiones entre dispositivos con conectores hembra. Estos cables son esenciales en

proyectos de Arduino, protoboard y sensores para establecer conexiones eléctricas y de datos de manera conveniente y eficiente.

Desarrollo

La sección de desarrollo del proyecto es el núcleo del proceso en el cual se transforman los conceptos teóricos y las etapas iniciales en un sistema de Internet de las Cosas (IoT) totalmente funcional. Este informe aborda detalladamente cómo se diseñó e implementó cada aspecto del proyecto, desde el prototipo inicial hasta la arquitectura completa del sistema de IoT, incluyendo la base de datos en la nube, la interfaz de programación de aplicaciones (API), la visualización de datos y su interconexión.

El desarrollo del proyecto se inicia con la construcción del prototipo, una versión inicial del sistema de IoT. En esta etapa, se exploran y prueban diferentes configuraciones de hardware y software para garantizar que el diseño sea viable y cumpla con los requisitos. Se describirá en detalle la selección de componentes, la estructura del prototipo y cómo evolucionó a lo largo del proyecto.

La base de datos en la nube es un componente central para almacenar y gestionar los datos recopilados por los sensores y actuadores. En esta sección, se detallará la elección de la plataforma de nube, la estructura de la base de datos, y cómo se configuró para admitir la recopilación y el almacenamiento eficientes de datos en tiempo real.

La API se encarga de permitir la comunicación y el intercambio de datos entre los microcontroladores y la base de datos en la nube. Se examinará cómo se desarrollaron y configuraron las API para habilitar la transmisión de datos bidireccional, lo que permite a los dispositivos enviar datos a la nube y recibir comandos de control.

La visualización de datos desempeña un papel crucial en la utilidad de cualquier sistema de IoT. En esta sección, se explicará cómo se diseñaron las interfaces de usuario para mostrar información en tiempo real y cómo se generaron al menos cinco indicadores distintos (mediante 4 sensores) para proporcionar una comprensión clara de los datos recopilados.

La arquitectura general del sistema se considera un elemento crítico en el desarrollo del proyecto. Se analizará cómo se establecieron las conexiones entre los microcontroladores, la nube y la interfaz de usuario, y cómo se garantizó la seguridad y la eficiencia en la transmisión de datos.

A lo largo de esta sección de desarrollo, se abordarán los desafíos, éxitos y lecciones aprendidas en cada aspecto del proyecto, proporcionando una visión completa del proceso de creación de un sistema de IoT funcional y adaptable.

Prototipo

La sección del prototipo de simulación representa un hito esencial en el desarrollo de nuestro proyecto de Internet de las Cosas (IoT). En esta fase final, se ha creado una versión a escala que encapsula los conceptos, componentes y funcionalidades clave del sistema completo. Esta sección profundizará en los detalles del prototipo, su diseño, implementación y su relevancia en el contexto del proyecto.

El prototipo ha sido concebido como un modelo reducido del sistema final de IoT. Se ha prestado especial atención a la selección de componentes, la configuración de sensores y actuadores, y la elección de los microcontroladores que imitan las capacidades de los dispositivos finales. El diseño del prototipo establece las bases para evaluar la viabilidad técnica y funcional del proyecto.

En esta sección, se presentarán los aspectos técnicos de la implementación del prototipo. Se describirá cómo se programaron los microcontroladores utilizando código Arduino (C++) para capturar datos de los sensores y controlar los actuadores. Además, se explicará cómo se logró la comunicación entre los microcontroladores y la transmisión de datos a través de la red.

El prototipo de simulación es esencial para evaluar la eficacia de nuestro enfoque en un entorno controlado. Permite la identificación temprana de posibles desafíos y la optimización de los componentes y la lógica del sistema antes de su implementación a mayor escala. También sirve como una herramienta de demostración y capacitación para los equipos involucrados en el proyecto.

El prototipo se ha desarrollado con objetivos específicos en mente, como la validación de conceptos, la identificación de posibles mejoras en el diseño, y la prueba de la interacción entre sensores y la base de datos en la nube. Este enfoque en la simulación a escala permite abordar desafíos potenciales de manera proactiva.

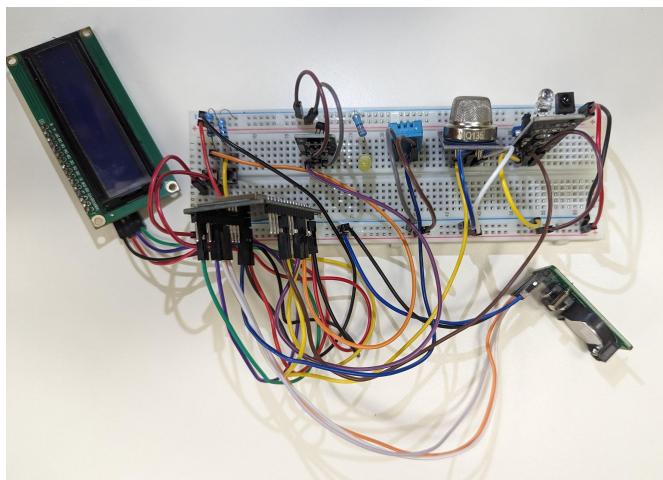


Figura 9. Prototipo (circuito) físico vista superior

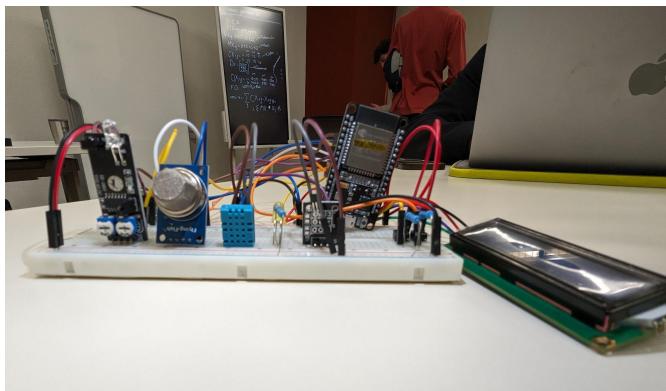


Figura 10. Prototipo (circuitu) físico vista lateral

En las ilustraciones superiores, se puede visualizar el prototipo final de esta situación problema. Este, incorpora varios componentes electrónicos clave, cada uno desempeñando un papel esencial en la funcionalidad general del sistema. Aquí describiremos en detalle estos componentes y cómo se conectan entre sí:

En primer lugar, el sensor de temperatura y humedad, se pretende que esté monitoreando el ambiente para medir la temperatura y la humedad, registrando datos cruciales para garantizar la frescura y la conservación adecuada de los productos almacenados en el interior del refrigerador. El sensor de proximidad utiliza infrarrojos para detectar objetos cercanos y su distancia, lo que permite registrar cuándo se abre o cierra la puerta del refrigerador, siendo esencial para el control de acceso y la eficiencia energética. El sensor de gas está diseñado para monitorear la calidad del aire, identificando y alertando sobre la presencia de gases potencialmente peligrosos en el entorno, registrando esta información para garantizar un ambiente seguro y saludable. Además, el módulo que proporciona una referencia precisa de fecha y hora, es esencial para el registro temporal de eventos y datos recopilados en el sistema de IoT. Estos componentes electrónicos desempeñan un papel central en la funcionalidad del sistema al medir y registrar datos críticos para el control y supervisión efectiva del refrigerador y su contenido.

Estos componentes electrónicos no solo miden, sino que también registran datos cruciales que constituyen la base de nuestro sistema de IoT, permitiendo un control preciso y una supervisión efectiva del refrigerador y su contenido.

El prototipo utiliza un microcontrolador de doble núcleo ESP32, que actúa como el cerebro del sistema IoT. Las conexiones de los sensores se realizan directamente en la protoboard, conectando los pines de voltaje y tierra, así como las señales al microcontrolador. Por otro lado, la pantalla y el módulo se conectan al microcontrolador, permitiendo la visualización de datos y el registro temporal de eventos. Para lograr una funcionalidad completa, el sistema

utiliza botones que se conectan al microcontrolador, permitiendo la interacción del usuario y actuando como un multiplexor en el código, lo que facilita la visualización de distintos valores de sensores en la pantalla.

Este prototipo, nos ayuda a simular la como sería la solución del problema en la vida real, este es el encargado de recopilar la información y mandarla a diferentes espacios (API, SQL, DASH) donde se interpretará el funcionamiento del mismo. El análisis detallado del prototipo de simulación es crucial para comprender su importancia en el contexto del proyecto de IoT y sienta las bases para la implementación exitosa de la versión completa del sistema. Este prototipo representa un paso crítico hacia la creación de un sistema funcional y adaptable.

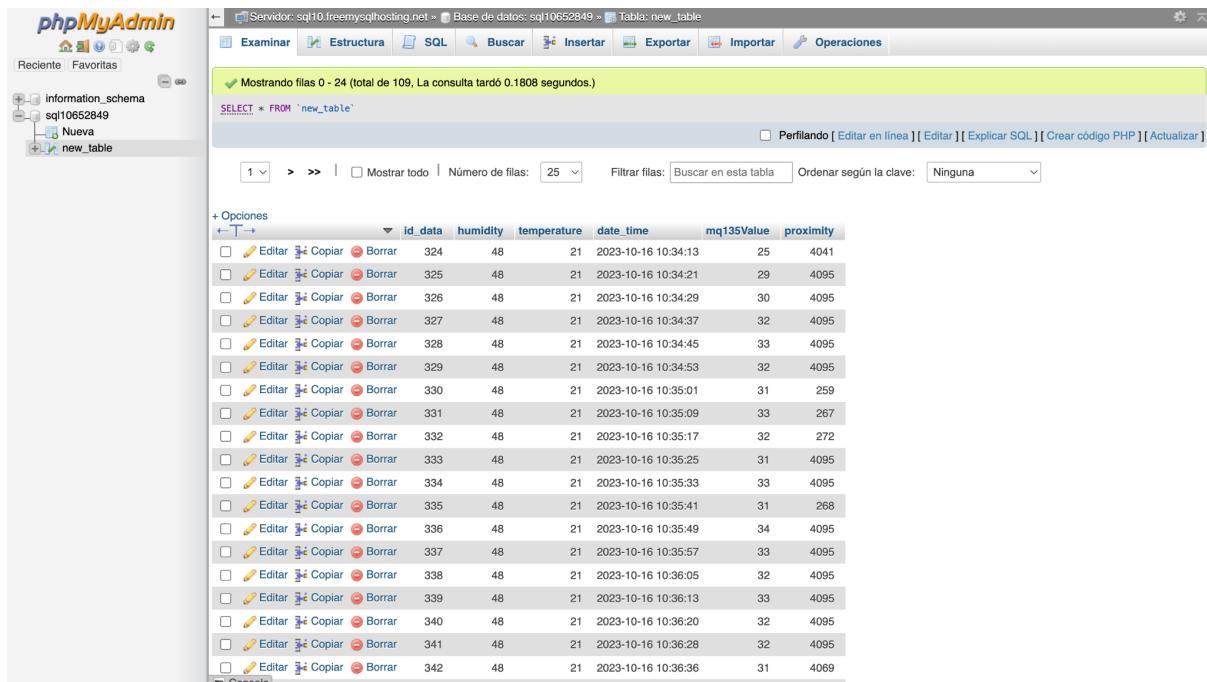
Base de Datos en la Nube

Free my SQL

MySQL es un sistema de gestión de bases de datos de código abierto que hemos utilizado. Es ampliamente conocido por su confiabilidad, escalabilidad y facilidad de uso. Algunas características clave que hemos experimentado incluyen:

1. Gratuito y de código abierto: MySQL es una opción gratuita y de código abierto que hemos aprovechado en numerosos proyectos, lo que ayuda a reducir los costos de licencia.
2. Rendimiento: Hemos notado que MySQL es rápido y eficiente en el manejo de consultas y transacciones, lo que lo hace ideal para aplicaciones que requieren un alto rendimiento.
3. Escalabilidad: Ha sido útil en situaciones en las que nuestras aplicaciones necesitaban crecer, ya que MySQL se adapta bien a las necesidades cambiantes.
4. Soporte para múltiples plataformas: Lo hemos utilizado en una variedad de sistemas operativos, lo que facilita su integración en diferentes entornos.
5. Replicación y alta disponibilidad: La capacidad de configurar replicación en MySQL ha sido esencial para garantizar la alta disponibilidad y la recuperación de desastres en algunos de nuestros proyectos.
6. Lenguaje SQL estándar: MySQL utiliza el estándar SQL para consultas y administración de datos, lo que simplifica el trabajo con él.
7. Amplia comunidad y soporte: La comunidad activa que respalda MySQL ha sido valiosa para obtener ayuda, tutoriales y recursos en línea.
8. Herramientas de administración: Hemos utilizado herramientas como MySQL Workbench para administrar nuestras bases de datos de manera más eficiente.

MySQL ha sido una opción confiable y versátil que hemos utilizado en nuestro trabajo. Su gratuidad, rendimiento y escalabilidad lo convierten en una elección sólida para una amplia gama de aplicaciones y entornos.



| | + Opciones | id_data | humidity | temperature | date_time | mq135Value | proximity |
|--------------------------|--|-------------------------|--------------------------|-----------------------------|---------------------------|----------------------------|---------------------------|
| <input type="checkbox"/> | Editar Copiar Borrar | 324 | 48 | 21 | 2023-10-16 10:34:13 | 25 | 4041 |
| <input type="checkbox"/> | Editar Copiar Borrar | 325 | 48 | 21 | 2023-10-16 10:34:21 | 29 | 4095 |
| <input type="checkbox"/> | Editar Copiar Borrar | 326 | 48 | 21 | 2023-10-16 10:34:29 | 30 | 4095 |
| <input type="checkbox"/> | Editar Copiar Borrar | 327 | 48 | 21 | 2023-10-16 10:34:37 | 32 | 4095 |
| <input type="checkbox"/> | Editar Copiar Borrar | 328 | 48 | 21 | 2023-10-16 10:34:45 | 33 | 4095 |
| <input type="checkbox"/> | Editar Copiar Borrar | 329 | 48 | 21 | 2023-10-16 10:34:53 | 32 | 4095 |
| <input type="checkbox"/> | Editar Copiar Borrar | 330 | 48 | 21 | 2023-10-16 10:35:01 | 31 | 259 |
| <input type="checkbox"/> | Editar Copiar Borrar | 331 | 48 | 21 | 2023-10-16 10:35:09 | 33 | 267 |
| <input type="checkbox"/> | Editar Copiar Borrar | 332 | 48 | 21 | 2023-10-16 10:35:17 | 32 | 272 |
| <input type="checkbox"/> | Editar Copiar Borrar | 333 | 48 | 21 | 2023-10-16 10:35:25 | 31 | 4095 |
| <input type="checkbox"/> | Editar Copiar Borrar | 334 | 48 | 21 | 2023-10-16 10:35:33 | 33 | 4095 |
| <input type="checkbox"/> | Editar Copiar Borrar | 335 | 48 | 21 | 2023-10-16 10:35:41 | 31 | 268 |
| <input type="checkbox"/> | Editar Copiar Borrar | 336 | 48 | 21 | 2023-10-16 10:35:49 | 34 | 4095 |
| <input type="checkbox"/> | Editar Copiar Borrar | 337 | 48 | 21 | 2023-10-16 10:35:57 | 33 | 4095 |
| <input type="checkbox"/> | Editar Copiar Borrar | 338 | 48 | 21 | 2023-10-16 10:36:05 | 32 | 4095 |
| <input type="checkbox"/> | Editar Copiar Borrar | 339 | 48 | 21 | 2023-10-16 10:36:13 | 33 | 4095 |
| <input type="checkbox"/> | Editar Copiar Borrar | 340 | 48 | 21 | 2023-10-16 10:36:20 | 32 | 4095 |
| <input type="checkbox"/> | Editar Copiar Borrar | 341 | 48 | 21 | 2023-10-16 10:36:28 | 32 | 4095 |
| <input type="checkbox"/> | Editar Copiar Borrar | 342 | 48 | 21 | 2023-10-16 10:36:36 | 31 | 4069 |

Figura 11. MySQL Online con Datosos

API

El código de api es esencialmente un programa de servidor web que hemos diseñado para recibir datos de sensores y almacenarlos en una base de datos MySQL. Tenemos un dispositivo como un microcontrolador ESP32 con varios sensores que recopilan información, como la temperatura, humedad, valor de MQ135 y proximidad. Este código actúa como una especie de intermediario que recibe estos datos y los guarda de manera organizada para su posterior uso.

Hemos utilizado Flask, que es un framework de Python para crear aplicaciones web. En nuestro código, hemos creado una instancia de una aplicación Flask llamada "app." Flask se encarga de manejar las solicitudes HTTP entrantes y las rutas que definimos en nuestro programa.

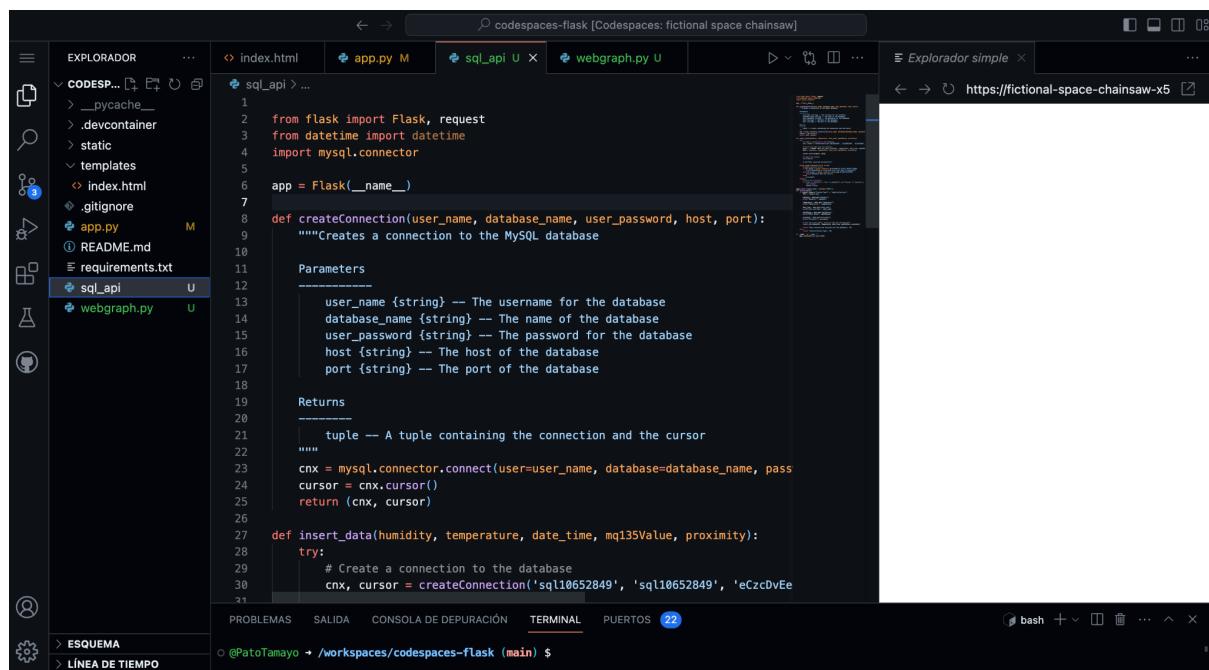
Tenemos dos funciones clave en nuestro código: `createConnection` y `insert_data`. La función `createConnection` se encarga de establecer una conexión con una base de datos MySQL. Para lograr esto, toma información como el nombre de usuario, la contraseña, el nombre de la base de datos, la dirección del servidor y el puerto de conexión. Esta función

retorna una conexión a la base de datos y un "cursor" que utilizamos para interactuar con la base de datos.

Por otro lado, la función `insert_data` toma datos de sensores (como humedad, temperatura, fecha y hora, valor MQ135 y proximidad) y los inserta en una tabla específica en la base de datos MySQL. Si la inserción de datos es exitosa, confirmamos y cerramos la conexión y el cursor. En caso de errores, como problemas de autenticación o problemas con la base de datos, los manejamos apropiadamente.

Hemos definido una ruta en la aplicación Flask: `/sensor_data`. Cuando recibimos una solicitud POST en esta ruta, verificamos si los datos tienen el formato correcto (en este caso, esperamos que estén en formato JSON). Luego, procesamos los datos, como la humedad, temperatura y otros valores de los sensores, y utilizamos la función `insert_data` para guardarlos en la base de datos. Si todo va bien, respondemos con un mensaje de confirmación.

Finalmente, cuando ejecutamos el programa , Flask se inicia y comienza a escuchar en un puerto (en este caso, el puerto 5000). Esto significa que el programa está listo para recibir datos de los sensores a través de solicitudes HTTP y guardarlos de manera ordenada en la base de datos. En resumen, este código es una parte crucial para la recopilación y almacenamiento de datos de sensores en un proyecto más amplio que hemos desarrollado.



The screenshot shows a Codespace environment with the following components:

- EXPLORADOR** sidebar: Shows files like index.html, app.py, sql_api, webgraph.py, README.md, requirements.txt, and .gitignore.
- Code Editor**: Displays Python code for `sql_api`. It includes a docstring for `createConnection` and a function `insert_data`.
- Terminal**: Shows a bash prompt at `@PatoTamayo ~ /workspaces/codespaces-flask (main) $`.
- Browser Tab**: Shows a browser window at `https://fictional-space-chainsaw-x5`.

```

1  from flask import Flask, request
2  from datetime import datetime
3  import mysql.connector
4
5  app = Flask(__name__)
6
7  def createConnection(user_name, database_name, user_password, host, port):
8      """Creates a connection to the MySQL database
9
10     Parameters
11         user_name {string} -- The username for the database
12         database_name {string} -- The name of the database
13         user_password {string} -- The password for the database
14         host {string} -- The host of the database
15         port {string} -- The port of the database
16
17     Returns
18
19         tuple -- A tuple containing the connection and the cursor
20
21     """
22
23     cnx = mysql.connector.connect(user=user_name, database=database_name, pass
24     cursor = cnx.cursor()
25     return (cnx, cursor)
26
27 def insert_data(humidity, temperature, date_time, mq135Value, proximity):
28     try:
29         # Create a connection to the database
30         cnx, cursor = createConnection('sql10652849', 'sql10652849', 'eCzcDvEe
31
32
33
34
35
36
37
38
39
3

```

Figura 12. Codespace

Arquitectura

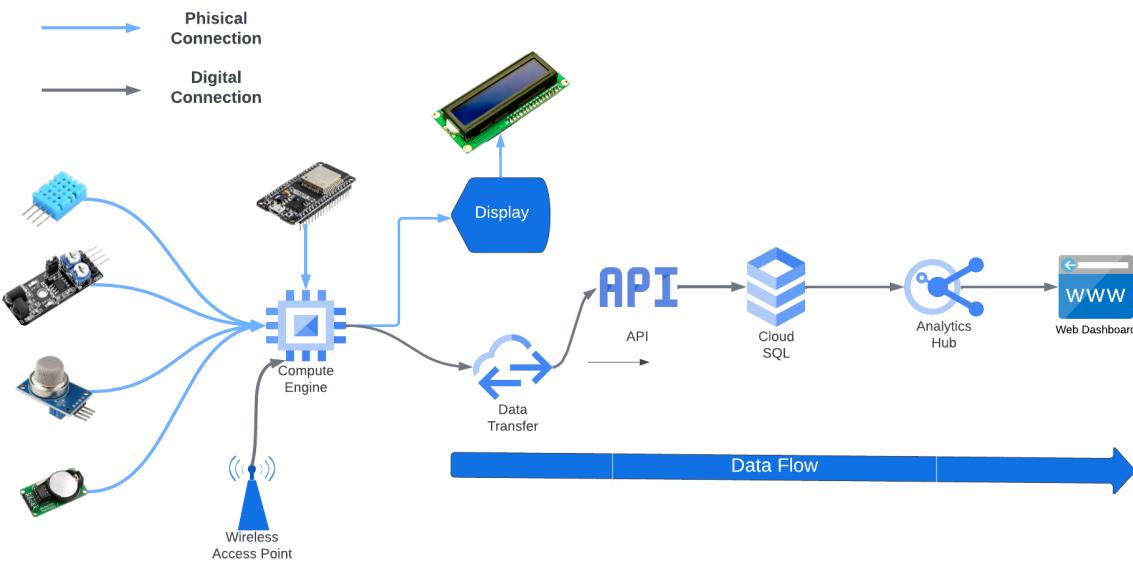


Figura 13. Diagrama de arquitectura general

Adentrándonos un poco más en las partes específicas de la arquitectura, encontramos dos partes relevantes dentro del sistema. En primer lugar tenemos la conexión entre el código Arduino local y el servidor de Codespaces. En primer lugar, se busca transmitir los datos de los sensores, recabados por el microcontrolador ESP32, directamente al servidor. Esto requiere de un API, como se mencionó anteriormente, la cual de manera más detallada es una interfaz que permite la comunicación entre dos programas.

Dentro del código de arduino, encontramos puntos clave que se deben resaltar. En primer lugar, se utiliza la librería WiFi.h para crear la conexión entre el código e internet, permitiendo que la API lleve a cabo la comunicación de datos. Por otro lado, encontramos el API Endpoint, que permite guardar la dirección del URL a donde se enviarán los datos. Mediante un método POST, los datos seguirán la ruta definida en Codespaces (mediante la instancia de la aplicación Flask). En el código del servidor, mismo que estableció la creación del framework Flask donde se permitiría crear una ruta para la recepción de datos. Esta ruta cuenta con los métodos get y POST, utilizados para recibir y recabar los datos enviados mediante el código de Arduino. Finalmente, los datos se envían en formato JSON, debido a la compatibilidad con codespaces de GitHub al igual que la forma en que están diseñados para documentos.

Finalmente, al tener la ruta por la cual se reciben los datos, es importante recordar que estos mismos se registraron y enviaron en formato JSON. Debido a esto, es importante que

la transferencia del servidor de Codespaces, permite inicializar la función de envío de datos a la base de datos. El envío de datos se lleva a cabo mediante una conexión de la biblioteca mysql.connector, donde se deben validar las credenciales apropiadas de la base de datos correspondientes. El envío de datos se lleva a cabo con un query, permitiendo que los datos sean ingresados correctamente. Se utilizan los métodos cursos y cnx para ejecutar y confirmar los cambios de los valores ingresados. De esta forma, permitimos que el arduino logré enviar correctamente mediante un terciario, los datos a un registro de MySQL para su correcto almacenamiento de manera no local.

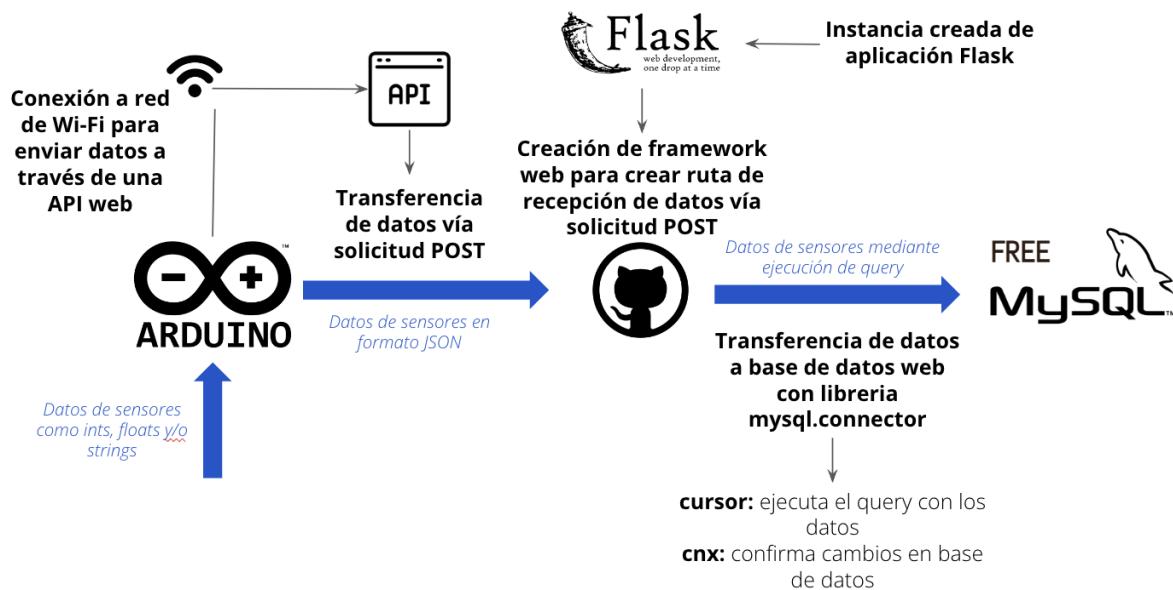


Figura 14. Diagrama de arquitectura (Transferencia de datos vía Wi-Fi y API)

Visualización de Datos

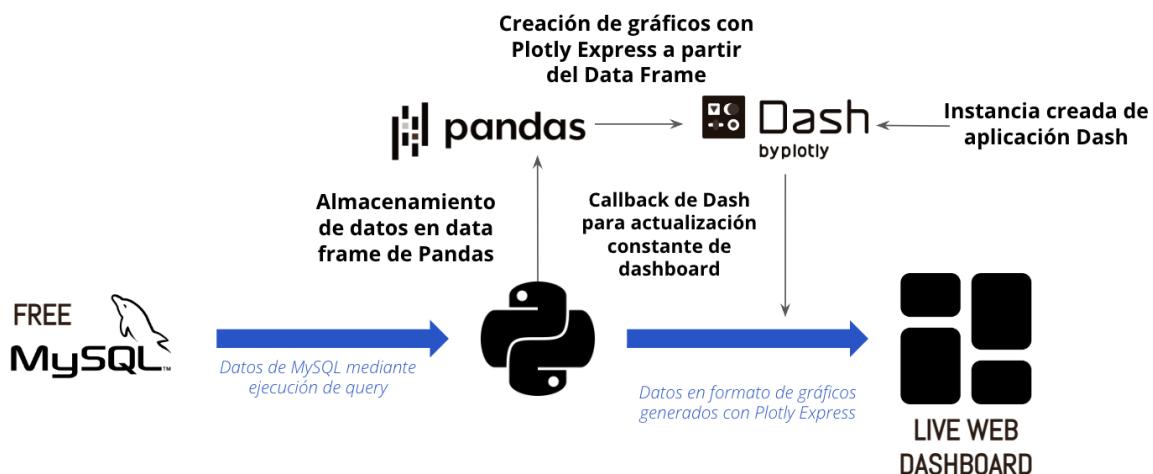


Figura 15. Diagrama de arquitectura (Visualización de Datos en Dashboard Live)

La segunda parte de la arquitectura involucra la recolección de los datos de la base de MySQL con el fin de poder visualizarla de manera efectiva. El tener una base de datos a la cual se le agreguen tuplas con datos nuevos y distintos atributos cada 5 segundos (debido al delay que tiene el código de lectura de sensores), no permite que alguien fácilmente pueda notar las fluctuaciones de los valores y lectura de los sensores, aplicados a un contexto real, no permitiría tener un alcance y visibilidad correctos como se pretende en un sistema.

Dado esto, la parte de visualización de datos es clave para permitir que el sistema funcione de manera adecuada. Como bien sabemos, uno de los principios fundamentales de IoT es la recopilación de datos para su correcto análisis. De esta forma, se busca tener un código adicional, donde se puedan obtener los datos de la base de datos en el servidor gratis de MySQL y poder desplegar estos de forma intuitiva. Esto se logra recuperando los datos mediante, de nuevo, una conexión a la base de datos que habilite la creación de una consulta para seleccionar los datos. A partir de esta consulta, debemos recuperar estos datos e insertarlos (con la ayuda de la biblioteca Pandas) en un DataFrame que permita la obtención de estos mismos y traducción a gráficas visuales.

Teniendo el data frame creado y vinculado con la selección de datos, se procede a utilizar la librería Dash. Se crea una instancia de la misma, y con esto, se permite inicializar una página web que recibe los datos del data frame, y crear un gráfico que sea pertinente a las variables. Finalmente, existe una instancia de la aplicación dash que se ejecuta con un evento específico, el cual, en este caso, se refiere al cambio de número de intervalos. Dicho callback tiene dos salidas, el gráfico de los valores de humedad, temperatura y gas al igual que un segundo relacionado a la proximidad. El callback permite la actualización de gráficos debido a la función encontrada en su cuerpo y la creación de gráficos a partir de la selección de datos de otra función incluida en el callback. Utilizando Plotly Express, ambos gráficos son creados, referidos y actualizados al dashboard online.

Lenguajes de Código Utilizados

La sección de "Lenguajes de Código Utilizados" es un componente esencial de este informe, ya que ofrece una visión detallada de las implementaciones técnicas que respaldan la operatividad de nuestro sistema de Internet de las Cosas (IoT). En esta sección, se explorarán y analizarán los códigos utilizados en el proyecto, destacando el uso de tres lenguajes de programación fundamentales: Arduino (C++), Python y HTML.

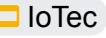
- Código de [Arduino \(C++\)](#):
 - El código de Arduino es la columna vertebral de nuestro sistema, ya que controla el funcionamiento de los microcontroladores. A lo largo de esta sección, se presentarán los fragmentos de código escritos en C++ para Arduino, que abordan la interacción con sensores y actuadores, la transmisión de datos a la nube y la recepción de comandos de control. Exploraremos las funciones y estructuras clave que hacen posible la comunicación entre el hardware y el software.
- Código de [Python](#):
 - Python se utiliza para desarrollar las API y las operaciones de gestión de datos en la nube. En esta sección, se examinarán las secuencias de comandos escritas en Python que facilitan la comunicación entre los

microcontroladores y la base de datos en la nube. Se destacará la lógica subyacente que permite la transferencia eficiente de información en tiempo real.

- Código de [HTML](#):
 - El código HTML se utiliza para crear las interfaces de usuario y la visualización de datos. Se explorarán los fragmentos de código HTML que dan forma a las pantallas de visualización de datos en tiempo real. Se describirán las estructuras, etiquetas y estilos utilizados para crear una experiencia de usuario intuitiva y efectiva.

A lo largo de esta sección, se desglosarán los códigos utilizados en cada uno de estos lenguajes de programación, se analizarán sus funciones específicas y se resaltarán las sinergias entre ellos para lograr la funcionalidad integral del sistema de IoT. Este análisis técnico es esencial para comprender el núcleo del proyecto y su viabilidad técnica.

Enlaces Externos

Enlace a presentación final click en el siguiente tab → 

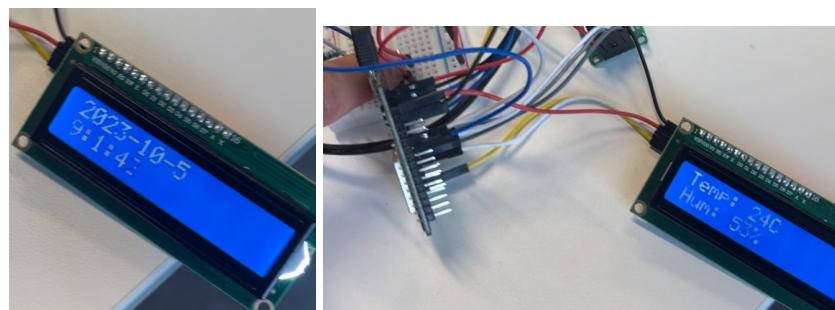
Link a repositorio de SP https://github.com/maxgallardo0/SP_IoT

Link a Codespaces de SP:<https://github.com/PatoTamayo/SP>

Link a Youtube: [Click aquí](#)

Resultados y Conclusión

A partir de una demostración hecha en clase, pudimos visualizar los resultados esperados. En estos, pudimos encontrar 3 aspectos esenciales que nos hicieron encontrar la validez del trabajo realizado en las últimas 5 semanas. En primera instancia se tienen los resultados visualizados en el display LCD. Estos, muestran un funcionamiento correcto de las conexiones del circuito físico, ya que a través de un circuito combinacional de tipo multiplexor, se logra determinar que tipo de lectura se quiere visualizar. Dicho resultado no puede ser obtenido sin antes contemplar las conexiones que cada sensor requiere, las conexiones generales de la LCD junto con el ESP32 y las conexiones básicas de tierra, voltaje, resistencias y push buttons.



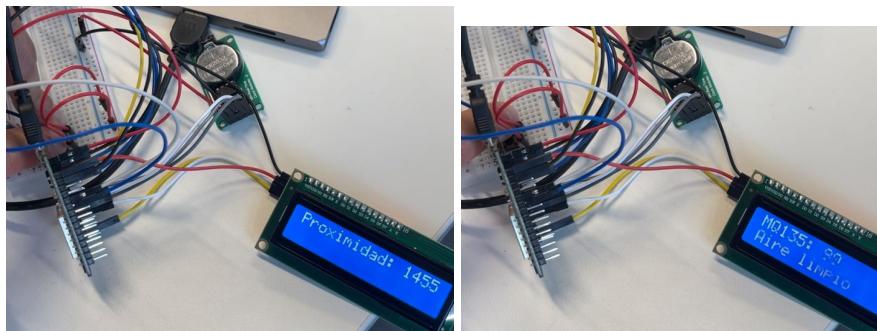
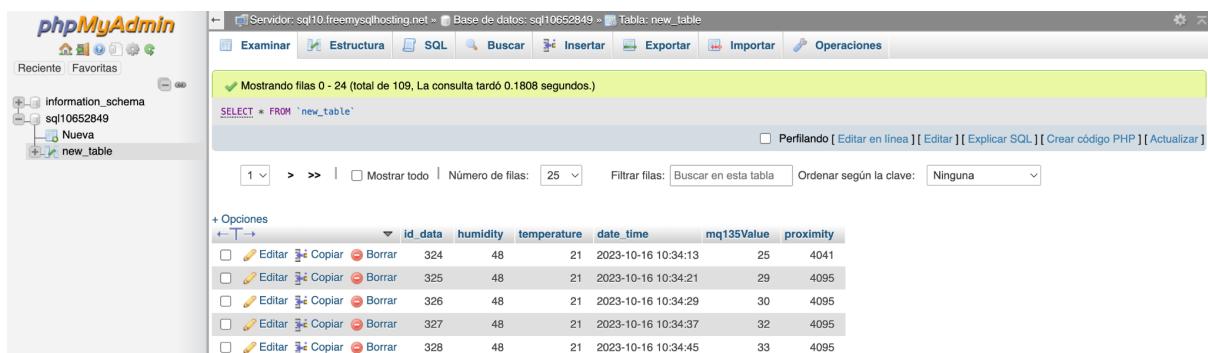


Figura 16. LCD Display con las 4 lecturas de sensores

Por otro lado, otro resultado que demuestra la conclusión correcta del proyecto, es la conexión entre código en python e instancias creadas en Codespaces, para el traslado de dichos datos a MySQL. En la base de datos global, logramos registrar las diferentes lecturas. En este caso, dicho registro sirvió para su posterior visualización pero el hecho de tener el conocimiento y aprendizaje de inserción a bases de datos, nos permite poder tener aplicaciones nuevas a esta herramienta como análisis profundo de datos en programas como MiniTab.



| | <input type="checkbox"/> Editar | <input type="checkbox"/> Copiar | <input type="checkbox"/> Borrar | id_data | humidity | temperature | date_time | mq135Value | proximity |
|--------------------------|---------------------------------|---------------------------------|---------------------------------|---------|----------|-------------|---------------------|------------|-----------|
| <input type="checkbox"/> | <input type="checkbox"/> Editar | <input type="checkbox"/> Copiar | <input type="checkbox"/> Borrar | 324 | 48 | 21 | 2023-10-16 10:34:13 | 25 | 4041 |
| <input type="checkbox"/> | <input type="checkbox"/> Editar | <input type="checkbox"/> Copiar | <input type="checkbox"/> Borrar | 325 | 48 | 21 | 2023-10-16 10:34:21 | 29 | 4095 |
| <input type="checkbox"/> | <input type="checkbox"/> Editar | <input type="checkbox"/> Copiar | <input type="checkbox"/> Borrar | 326 | 48 | 21 | 2023-10-16 10:34:29 | 30 | 4095 |
| <input type="checkbox"/> | <input type="checkbox"/> Editar | <input type="checkbox"/> Copiar | <input type="checkbox"/> Borrar | 327 | 48 | 21 | 2023-10-16 10:34:37 | 32 | 4095 |
| <input type="checkbox"/> | <input type="checkbox"/> Editar | <input type="checkbox"/> Copiar | <input type="checkbox"/> Borrar | 328 | 48 | 21 | 2023-10-16 10:34:45 | 33 | 4095 |

Figura 17. MySQL online con datos insertados

Finalmente, el proyecto concluye con lo que un cliente o usuario esperaría. No buscamos que el usuario lea una pantalla LCD o utilice una base de datos para conocer las lecturas. Esto, aunque nos brinda las bases y herramientas de los datos, es contraproducente para la solución que se tenía pensada. Dado esto, el hecho de poder visualizarlo de una manera más efectiva y amigable permite concluir con el argumento más importante del proyecto. El IoT tiene infinitas aplicaciones, pero nosotros logramos apreciar una de las más claves, una visualización de datos apropiada. En la figura siguiente podemos observar un dashboard sobre las fluctuaciones en datos. Dicho dashboard, permite que un usuario común, ya sea un individuo en casa o una planta de producción, pueda tener en tiempo real, lecturas relevantes sobre sus sistemas de refrigeración. El tener acceso a este tipo de datos, que tienen un sin fin de trabajo detrás, nos dio a entender la importancia de involucrar tantos aspectos para eficientizar procesos que podrían ser muy complicados. En vez de tener 4 lecturas esparcidas por diferentes plataformas, o incluso en un mismo dispositivo, este sistema de IoT permite tener un resultado efectivo del comportamiento real de nuestro refrigerador simulado.

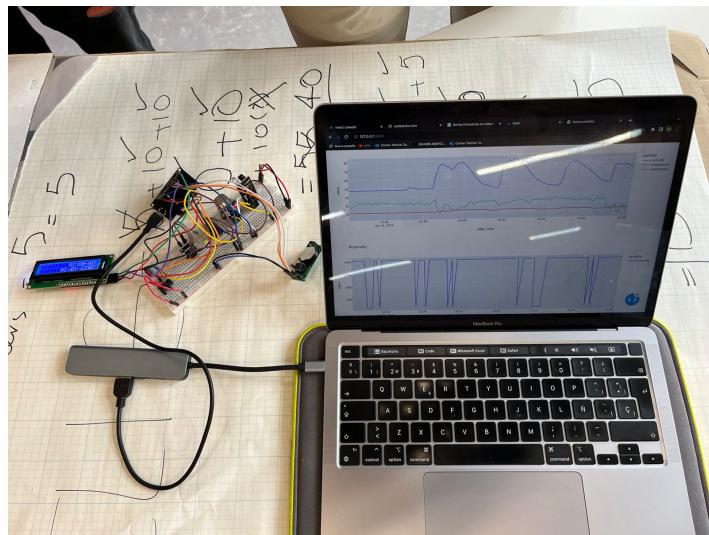


Figura 18. Dashboard con lecturas de sensores de circuito físico

Conclusiones Personales

Carlos Corzo

El reporte destaca el impacto de la transformación digital en la sociedad, con un enfoque particular en la Internet de las Cosas (IoT). El avance tecnológico nos ha permitido la integración de sensores en diversos procesos para la situación problema, lo que tiene un gran potencial para mejorar la eficiencia y la toma de decisiones en diferentes áreas, desde la gestión de la calidad del aire hasta el control de la temperatura en un refrigerador. Esto subraya la importancia de la innovación y la tecnología en la actualidad.

El proyecto de IoT se divide en varias fases, desde la definición del proyecto hasta la validación. Esto refleja la importancia de una metodología estructurada y un enfoque paso a paso para abordar proyectos tecnológicos complejos como la SCRUM. La planificación y la gestión adecuadas son fundamentales para el éxito por lo que la utilización de GitHub fue una herramienta esencial en el desarrollo y control.

La elección cuidadosa de los materiales es la base para el éxito del proyecto de IoT. La inclusión de microcontroladores, sensores y otros componentes electrónicos da base a comprender y seleccionar los recursos adecuados para el trabajo. Esto también resalta la importancia de conocer las herramientas disponibles para abordar los desafíos técnicos.

El reporte muestra la importancia de la conectividad y la comunicación en un sistema IoT. La capacidad de transmitir datos desde los sensores hasta una base de datos en la nube es crucial para la recopilación y el análisis de datos en tiempo real. Esto resalta la importancia de las API y la programación en la gestión de datos.

La visualización de datos desempeña un papel fundamental en la utilidad del sistema. Los datos deben presentarse de manera efectiva para que los usuarios o empresas puedan comprender y utilizar la información recopilada. Esto nos da paso a mencionar la relevancia de las interfaces de usuario y las herramientas de visualización de datos como la LCD que utilizamos y el DASH.

Este proyecto de IoT es el resultado de la colaboración entre el equipo con conocimientos en programación, electrónica y gestión de bases de datos. Esto destaca la importancia de trabajar en equipo y aprovechar diversas habilidades para abordar proyectos tecnológicos complejos.

Carlos Rubén Maldonado Barreda

La situación problema fue una que se sintió como un reto total. Desde el inicio, notamos como esta misma incorporaba aspectos de distintas partes del IoT. Pudimos indagar en la parte electrónica, de software y de arquitectura de sistemas con el fin de lograr tener un dispositivo que emule una solución a un problema real. De forma complementaria, encontramos también que la parte de desarrollo de proyectos en ocasiones favoreció una mejor colaboración.

En resumen, lo que se buscó fue hacer un acercamiento a 3 esferas principales: conexión de circuitos físicos, programación con distintos lenguajes (incluyendo SQL) y conexión entre dispositivos y plataformas. Esta combinación es fundamental para el establecimiento de un sistema de IoT. En general, el aprendizaje fue modular, permitiendo que conforme se avanzará, se aportarán nuevos conocimientos del sistema y funcionalidades antes no exploradas, de lo que se podía hacer. De forma muy superficial, nos permitió entender de qué manera el mundo físico, el mundo de software y las conexiones entre estos funcionan.

Es importante mencionar los puntos de éxito que el proyecto tuvo. En primera instancia, creemos que el hecho de haber incluido un sensor adicional fue un punto crítico y de mucha dificultad, pero que favoreció la adecuación del sistema a un entorno real. La forma en que se pensó el sensor de proximidad fue mediante la apertura de una puerta del refrigerador sugerido, lo que embonaba directamente con la visualización de datos que se pretende compartir. Dicho sensor nos reto, debido a la complejidad de su conexión y sus características y parámetros para lecturas. Por otro lado, creemos que fue valioso incluir un dashboard que se actualizará de manera automática. Esto no solo permite tener una solución más coherente (nadie quiere ver datos pasados si no lo que actualmente está pasando). Finalmente, el hecho de entender correctamente los pasos y la forma en que el sistema está compuesto, sobre todo por la parte de arquitectura, es algo que fomenta el

correcto aprendizaje. Como tal, las múltiples conexiones y transferencia de datos no se visualizaron de manera individual, si no que el reto nos permitió contemplarlas como un grupo o colectivo. A parte de esto, la colaboración con mis compañeros impulsó que se apreciará el sistema en general y no por partes.

Algunas áreas de oportunidad que cabe la pena destacar son; el orden en la conexión de los circuitos físicos, una explicación más profunda de ciertos parámetros de código como los métodos en las rutas para poder replicarlos a futuro y un diseño del refrigerador físico, para demostrar correctamente el funcionamiento de los distintos sensores.

Maximiliando Gallardo

Durante el desarrollo de esta situación problemática, se nos brindó la oportunidad de adentrarnos en un proyecto de Internet de las Cosas (IoT) lleno de desafíos y emociones. A lo largo de este proceso, exploramos a fondo el mundo de los sensores, las conexiones, las bases de datos y la programación, construyendo gradualmente nuestro conocimiento y habilidades en estas áreas fundamentales.

El proceso de desarrollo ha sido un viaje apasionante. Comenzamos con la creación de un prototipo, que sirvió como punto de partida para experimentar con diversas configuraciones de hardware y software. La base de datos en la nube emergió como una pieza fundamental al permitirnos almacenar y gestionar los datos generados por los sensores y actuadores. La API se convirtió en el vehículo que facilitó la comunicación efectiva entre estos elementos, permitiéndonos la transmisión de datos en tiempo real.

La visualización de datos en interfaces de usuario, como Dash, se erigió como una herramienta crucial para nuestra comprensión de los datos recopilados. La representación gráfica nos brindó una visión clara y significativa del comportamiento de los sensores, lo que contribuyó a una mejor toma de decisiones y análisis.

A lo largo de este proceso, no faltaron desafíos que pusieron a prueba nuestra paciencia y habilidades de resolución de problemas. Pero cada obstáculo superado fue una lección valiosa en nuestra travesía de aprendizaje. A menudo, descubrimos que los errores aparentemente insignificantes, como la selección del pin incorrecto, podían tener un impacto significativo en el funcionamiento del sistema. Estos momentos de enfrentar dificultades nos obligaron a profundizar en nuestra comprensión y a buscar soluciones efectivas.

Este proyecto ha sido un auténtico reto, pero también un testimonio de nuestro compromiso, perseverancia y espíritu de equipo. Nunca renunciamos y trabajamos juntos para superar esta compleja situación problema. Hoy, estamos orgullosos de lo que hemos logrado y

emocionados por las lecciones que nos ha brindado. Este viaje en el mundo de la IoT ha sido una experiencia inmensamente enriquecedora que ha ampliado nuestra comprensión y nos ha preparado para afrontar futuros desafíos con confianza y determinación.

Patricio Tamayo Gómez

La situación problema que abordamos en este proyecto representó un desafío fascinante desde el principio. De manera notable, combinó elementos de la electrónica, la programación y la arquitectura de sistemas, permitiéndonos explorar y comprender en profundidad cada uno de estos aspectos esenciales de la Internet de las Cosas (IoT). A medida que avanzábamos en el proyecto, quedó claro que estábamos inmersos en un proceso de aprendizaje modular, en el cual cada fase nos proporcionaba nuevos conocimientos y nos llevaba a explorar funcionalidades previamente desconocidas.

En esencia, nuestro enfoque se centró en tres esferas principales: la conexión de circuitos físicos, la programación con diversos lenguajes (incluyendo SQL) y la integración de dispositivos y plataformas. Comprendimos que esta combinación es fundamental para establecer un sistema de IoT eficaz. Nos permitió comprender de manera más profunda cómo interactúan el mundo físico y el mundo del software, así como las conexiones esenciales que los unen.

Es importante destacar los logros notables que alcanzamos en el proyecto. En primer lugar, la inclusión de un sensor adicional representó un desafío crítico, pero al mismo tiempo, una oportunidad de adaptar nuestro sistema a un entorno real. El sensor de proximidad, diseñado para detectar la apertura de una puerta de un refrigerador, se alineó perfectamente con la visualización de datos que buscábamos lograr. A pesar de su complejidad, abordar este sensor nos permitió adquirir un conocimiento más profundo de su conexión y sus parámetros de lectura.

Además, la implementación de un dashboard que se actualiza automáticamente fue un paso valioso. Esto garantiza que nuestra solución sea coherente y refleje en tiempo real lo que está sucediendo en el entorno. Por último, comprender la arquitectura general del sistema, con sus múltiples conexiones y transferencias de datos, nos llevó a apreciar la complejidad del sistema como un todo cohesionado. En lugar de ver las partes de manera aislada, logramos comprender cómo funcionan juntas en armonía.

Sin embargo, también identificamos áreas de oportunidad importantes. La organización de la conexión de los circuitos físicos requiere una mayor atención y orden. Además, una explicación más profunda de ciertos parámetros de código, como los métodos en las rutas,

sería beneficiosa para futuros proyectos. Además, consideramos que un diseño físico del refrigerador habría sido valioso para demostrar de manera más efectiva el funcionamiento de los distintos sensores.

En resumen, este proyecto de IoT fue una experiencia enriquecedora que nos permitió abordar una situación problema compleja y aprender en profundidad sobre las diversas disciplinas involucradas. Los desafíos que enfrentamos nos llevaron a lograr un entendimiento más completo del sistema, y la colaboración con nuestros compañeros fomentó una apreciación holística de la solución en su conjunto. Esta experiencia nos ha preparado para futuros proyectos y desafíos en el emocionante mundo de la Internet de las Cosas.

Rafael Ison Shaooli

En esta situación problema, hemos explorado un proyecto de Internet de las Cosas (IoT). Durante todas las clases fuimos aprendiendo más cosas las cuales pudimos integrar para la situación final. Fue de gran importancia el entendimiento de los tipos de sensores y las conexiones, así como las bases de datos, SQL y el funcionamiento de Arduino.

El proceso de desarrollo comienza con la creación de un prototipo, que actúa como punto de partida para experimentar con diversas configuraciones de hardware y software. La base de datos en la nube desempeña un papel muy importante al permitirnos almacenar y gestionar los datos recopilados por sensores y actuadores, mientras que la API facilita la comunicación eficaz. La visualización de datos en interfaces de usuario (Dash) proporciona una comprensión clara de los datos recopilados, y una visión gráfica del comportamiento de los sensores.

A lo largo de este proceso, enfrentamos muchos desafíos, y aprendimos lecciones valiosas que enriquecieron nuestra comprensión de cómo crear un sistema de IoT funcional y adaptable, fue realmente un reto, ya que poco a poco nos íbamos dando cuenta de cómo era que realmente funcionaban las cosas, luego teníamos errores y simplemente eran por una cosa pequeña, como por ejemplo el pin equivocado (no sabíamos que hay unos pines que tiene un comportamiento especial). Este proyecto fue realmente de gran aprendizaje y fue un muy buen trabajo en equipo, nunca nos rendimos y pudimos solucionar esta gran situación problema, aprendiendo y estando orgullosos de lo que logramos.

Bibliografías

Sensor de Temperatura y Humedad Relativa DHT11. (s. f.). Naylamp Mechatronics - Perú.

<https://naylampmechatronics.com/sensores-temperatura-y-humedad/57-sensor-de-temperatura-y-humedad-relativa-dht11.html>

ESP32 series datasheet. (2023). En

https://www.espressif.com/sites/default/files/documentation/esp32_datasheet_en.pdf (N.o ESP32-D0WD-V3).

Introducción a las LCD. (s. f.). https://www.5hertz.com/index.php?route=tutoriales/tutorial&tutorial_id=9

Llamas, L. (2016). Detector de obstáculos con sensor infrarrojo y arduino. Luis Llamas.

<https://www.luisllamas.es/detectar-obstaculos-con-sensor-infrarrojo-y-arduino/>

Sensor de Temperatura y Humedad Relativa DHT11. (s. f.). Naylamp Mechatronics - Perú.

<https://naylampmechatronics.com/sensores-temperatura-y-humedad/57-sensor-de-temperatura-y-humedad-relativa-dht11.html>

Sensor MQ-135 Gas Calidad de aire. (s. f.). Naylamp Mechatronics - Perú.

<https://naylampmechatronics.com/sensores-gas/73-sensor-mq-135-gas-calidad-aire.html>

Tableta Protoboard. (2021, 1 marzo). Portal Académico del CCH.

<https://portalacademico.cch.unam.mx/cibernetica/implementacion-de-circuitos-logicos/tableta-protoboard>

Using the MH-Real-Time Clock Modules-2. (s. f.).

<https://fritzing.org/projects/using-the-mh-real-time-clock-modules-2>

Anexo

Códigos

Arduino IDE

```
C/C++  
  
#include <Wire.h>  
#include <LiquidCrystal_I2C.h>  
#include <DFRobot_DHT11.h>  
#include <Ds1302.h>  
#include <WiFi.h>  
#include <HTTPClient.h>  
#include <ArduinoJson.h>  
  
  
#define DHT11_PIN 15  
#define IR_SENSOR_PIN 34  
#define MQ135_PIN 35  
#define PIN_ENA 19  
#define PIN_CLK 5  
#define PIN_DAT 18  
#define BUTTON1_PIN 14  
#define BUTTON2_PIN 27  
  
DFRobot_DHT11 dht11;  
Ds1302 rtc(PIN_ENA, PIN_CLK, PIN_DAT);  
LiquidCrystal_I2C lcd(0x27, 16, 2);  
  
const char* ssid = "Tec-Contingencia";  
const char* password = "";  
const           char*           apiEndpoint           =  
"https://fictional-space-chainsaw-x5wpgvx5q9jr3pgv7-5000.app.github.dev/sensor_data";  
  
int currentView = 0;  
  
void setupWifi() {  
    Serial.begin(9600);  
    Serial.print("Connecting to WiFi");  
    WiFi.begin(ssid, password);  
    while (WiFi.status() != WL_CONNECTED) {  
        delay(100);  
        Serial.print(".");  
    }  
    Serial.print(" Connected: ");  
    Serial.println(WiFi.localIP());  
}  
  
void setup() {  
    Serial.begin(9600);  
    pinMode(IR_SENSOR_PIN, INPUT);  
    pinMode(BUTTON1_PIN, INPUT);  
    pinMode(BUTTON2_PIN, INPUT);  
    lcd.init();  
    lcd.backlight();  
    rtc.init();  
}
```

```

Ds1302::DateTime dt = {
    .year = 23,
    .month = Ds1302::MONTH_OCT,
    .day = 12,
    .hour = 10,
    .minute = 0,
    .second = 0,
    .dow = Ds1302::DOW_THU
};
rtc.setDateTime(&dt);
setupWifi();
}

void sendData(float temperature, float humidity, int mq135Value, int proximity) {
    Serial.print("Sending data to API: ");

    HttpClient http;
    http.begin(apiEndpoint);
    http.addHeader("Content-Type", "application/json");

    StaticJsonDocument<300> doc;

    doc["temperature"] = temperature;
    doc["humidity"] = humidity;
    doc["mq135Value"] = mq135Value;
    doc["proximity"] = proximity;

    Ds1302::DateTime now;
    rtc.getDateTime(&now);
    String datetime = "20" + String(now.year) + "-" +
        ((now.month < 10) ? "0" : "") + String(now.month) + "-" +
        ((now.day < 10) ? "0" : "") + String(now.day) + " " +
        ((now.hour < 10) ? "0" : "") + String(now.hour) + ":" +
        ((now.minute < 10) ? "0" : "") + String(now.minute) + ":" +
        ((now.second < 10) ? "0" : "") + String(now.second);
    doc["date_time"] = datetime;

    String json;
    serializeJson(doc, json);
    Serial.println("Sending JSON: " + json);

    int httpResponseCode = http.POST(json);
    if (httpResponseCode > 0) {
        Serial.print("HTTP Response code: ");
        Serial.println(httpResponseCode);
        String responseString = http.getString();
        Serial.println("Received response: " + responseString);
    } else {
        Serial.print("Error code: ");
        Serial.println(httpResponseCode);
    }
    http.end();
}

void loop() {

```

```
dht11.read(DHT11_PIN);
float temperature = dht11.temperature;
float humidity = dht11.humidity;
int mq135Value = analogRead(MQ135_PIN) / 10;
int proximity = analogRead(IR_SENSOR_PIN);

Serial.print("Temperatura: ");
Serial.println(temperature);
Serial.print("Humedad: ");
Serial.println(humidity);
Serial.print("MQ135: ");
Serial.println(mq135Value);
Serial.print("Proximity: ");
Serial.println(proximity);

sendData(temperature, humidity, mq135Value, proximity);

int button1State = digitalRead(BUTTON1_PIN);
int button2State = digitalRead(BUTTON2_PIN);

if (button1State == LOW && button2State == HIGH) {
    currentView = (currentView + 1) % 4;
    delay(5000);
}

lcd.clear();

switch (currentView) {
    case 0: {
        Ds1302::DateTime now;
        rtc.getDateTime(&now);
        lcd.setCursor(0, 0);
        lcd.print("20");
        lcd.print(now.year);
        lcd.print("-");
        lcd.print(now.month);
        lcd.print("-");
        lcd.print(now.day);
        lcd.setCursor(0, 1);
        lcd.print(now.hour);
        lcd.print(":");
        lcd.print(now.minute);
        lcd.print(":");
        lcd.print(now.second);
        break;
    }
    case 1: {
        lcd.setCursor(0, 0);
        lcd.print("Temp: ");
        lcd.print(temperature);
        lcd.print("C");
        lcd.setCursor(0, 1);
        lcd.print("Hum: ");
        lcd.print(humidity);
    }
}
```

```
lcd.print("%");
break;
}
case 2: {
lcd.setCursor(0, 0);
lcd.print("MQ135: ");
lcd.print(mq135Value);
lcd.setCursor(0, 1);
if (mq135Value < 100) {
lcd.print("Aire limpio");
} else if (mq135Value < 200) {
lcd.print("Niveles normales");
} else {
lcd.print("Aire contaminado");
}
break;
}
case 3: {
lcd.setCursor(0, 0);
lcd.print("Proximidad: ");
lcd.print(proximity);
break;
}
}

delay(5000);
}
```

SQL-API

Python

```
from flask import Flask, request
from datetime import datetime
import mysql.connector

app = Flask(__name__)

def createConnection(user_name, database_name, user_password, host, port):
    """Creates a connection to the MySQL database

Parameters
-----
user_name {string} -- The username for the database
database_name {string} -- The name of the database
user_password {string} -- The password for the database
host {string} -- The host of the database
port {string} -- The port of the database

Returns
-----
```

```

        tuple -- A tuple containing the connection and the cursor
"""

    cnx    = mysql.connector.connect(user=user_name,    database=database_name,
password=user_password, host=host, port=port)
    cursor = cnx.cursor()
    return (cnx, cursor)

def insert_data(humidity, temperature, date_time, mq135Value, proximity):
    try:
        # Create a connection to the database
        cnx, cursor = createConnection('sql10652849', 'sql10652849', 'eCzcDvEeph',
'sql10.freemysqlhosting.net', '3306')

        # Insert the data into the database
        query = ("INSERT INTO new_table (humidity, temperature, date_time, mq135Value,
proximity) VALUES (%s, %s, %s, %s, %s)")
        data = (humidity, temperature, date_time, mq135Value, proximity)

        cursor.execute(query, data)

        # Commit the changes
        cnx.commit()

        print("Data inserted successfully")

    except mysql.connector.Error as err:
        # Handle possible errors
        if err.errno == mysql.connector.errorcode.ER_ACCESS_DENIED_ERROR:
            print("Something is wrong with your user name or password")
        elif err.errno == mysql.connector.errorcode.ER_BAD_DB_ERROR:
            print("Database does not exist")
        else:
            print(err)
    finally:
        # Close the connection
        if ('cnx' in locals() or 'cnx' in globals()) and ('cursor' in locals() or 'cursor' in
globals()):
            cnx.close()
            cursor.close()

@app.route("/sensor_data", methods=["POST"])
def receive_data():
    if request.headers["Content-Type"] == "application/json":
        data = request.json

        humidity = data.get("humidity")
        print("Humidity:", humidity)

        temperature = data.get("temperature")
        print("Temperature:", temperature)

        date_time = data.get("date_time")
        print("Date and Time:", date_time)

        mq135Value = data.get("mq135Value")

```

```
print("MQ135 Value:", mq135Value)

proximity = data.get("proximity")
print("Proximity:", proximity)

# Call the insert_data() function with the received data
insert_data(humidity, temperature, date_time, mq135Value, proximity)

return "Data received and inserted into the database", 200
else:
    return "Invalid Content-Type", 400

if __name__ == "__main__":
    app.run(debug=True, port=5000)
```

Dashboard

Python

```
from dash import Dash, html, dcc
import plotly.express as px
import pandas as pd
from dash.dependencies import Input, Output
import mysql.connector

# Función para crear una conexión a la base de datos
def createConnection(user_name, database_name, user_password, host, port):
    cnx = mysql.connector.connect(user=user_name, database=database_name,
password=user_password, host=host, port=port)
    cursor = cnx.cursor()
    return (cnx, cursor)

# Función para seleccionar datos de la base de datos
def select_data():
    try:
        # Crear una conexión a la base de datos
        cnx, cursor = createConnection('sql10652849', 'sql10652849', 'eCzcDvEeph',
'sql10.freemysqlhosting.net', '3306')

        # Consulta SQL para obtener datos (ajusta la consulta según tu esquema de base de
datos)
        query = "SELECT * FROM new_table"

        cursor.execute(query)

        # Obtener los datos
        data = cursor.fetchall()

        # Cerrar la conexión
        cursor.close()
        cnx.close()
```

```

    return data

except mysql.connector.Error as err:
    if err.errno == mysql.connector.errorcode.ER_ACCESS_DENIED_ERROR:
        print("Algo está mal con tu nombre de usuario o contraseña")
    elif err.errno == mysql.connector.errorcode.ER_BAD_DB_ERROR:
        print("La base de datos no existe")
    else:
        print(err)

data = pd.DataFrame(select_data(), columns=["id_data", "humidity", "temperature",
"date_time", "mq135Value", "proximity"])

# Crear una instancia de la aplicación Dash
app = Dash(__name__)

# Definir el diseño de la aplicación web
app.layout = html.Div([
    html.H1("Temperature and Humidity", style={'text-align': 'center'}),
    dcc.Graph(id='temperature-humidity-plot', figure=px.line(data, x='date_time',
y=[ "humidity", "temperature", "mq135Value"], title="Temperature and humidity are closely
related in outdoor applications")),
    dcc.Graph(id='proximity-plot', figure=px.line(data, x='date_time', y=[ "proximity"],
title="Things are far away")),
    dcc.Interval(
        id='interval-component',
        interval=5 * 1000, # Intervalo de actualización en milisegundos (5 segundos)
        n_intervals=0 # Inicialización del contador de intervalos
    )
])

@app.callback(
    [Output('temperature-humidity-plot', 'figure'),
     Output('proximity-plot', 'figure')],
    Input('interval-component', 'n_intervals')
)
def update_graphs(n):
    data = pd.DataFrame(select_data(), columns=["id_data", "humidity", "temperature",
"date_time", "mq135Value", "proximity"])

    fig1 = px.line(data, x='date_time', y=[ "humidity", "temperature", "mq135Value"],
title="Temperature and humidity")
    fig2 = px.line(data, x='date_time', y=[ "proximity"], title="Proximity")

    return fig1, fig2

if __name__ == '__main__':
    app.run_server(debug=True)

```