

UNIVERSIDAD CENTRAL DEL ECUADOR

FACULTAD DE INGENIERÍA Y CIENCIAS APLICADAS

SISTEMAS DE INFORMACIÓN

INFRAESTRUCTURA DE TI II



Título:

GEMELO DIGITAL

2024-2025

GEMELO DIGITAL

GRUPO 5:

Luis Andres Aucancela Morocho

Kenny Israel Cisneros Calderon

Davinson Mayer Diaz Tapia

Petter Alejandro Rodríguez Proaño

Juan Ariel Tulcanaza Paucar

Carrera Sistemas de Información, Facultad de Ingeniería y Ciencias Aplicadas

Ing. Robert Enríquez

2024 - 2025

ÍNDICE DE CONTENIDO

Practica – Gemelo Digital.....	¡Error! Marcador no definido.
OBJETIVOS	¡Error! Marcador no definido.
Objetivo general:.....	¡Error! Marcador no definido.
Objetivos específicos:	¡Error! Marcador no definido.
ALCANCE	¡Error! Marcador no definido.
LIMITACIONES	¡Error! Marcador no definido.
INTRODUCCION	¡Error! Marcador no definido.
MARCO TEÓRICO	¡Error! Marcador no definido.
1. Predicción y optimización del rendimiento	¡Error! Marcador no definido.
2. Estrategia de emisiones.....	¡Error! Marcador no definido.
3. Ahorro de tiempo y costos de desarrollo.....	¡Error! Marcador no definido.
4. Modelo de super red tres capas	¡Error! Marcador no definido.
5. Construcción modelo super red de datos	¡Error! Marcador no definido.
SERVERS	¡Error! Marcador no definido.
DESARROLLO	¡Error! Marcador no definido.
Configuración Arduinos.....	¡Error! Marcador no definido.
Configuración MQTT	¡Error! Marcador no definido.
Configuración Unity	¡Error! Marcador no definido.
CONCLUSIONES	¡Error! Marcador no definido.
ANEXOS	¡Error! Marcador no definido.
BIBLIOGRAFIA	¡Error! Marcador no definido.

ÍNDICE DE ILUSTRACIONES

Ilustración 1: Interfaz Putty	¡Error! Marcador no definido.
Ilustración 2: Icono V380 cámara	¡Error! Marcador no definido.
Ilustración 3: Interfaz 3CX	¡Error! Marcador no definido.
Ilustración 4: Interfaz Virtual Box	¡Error! Marcador no definido.
Ilustración 5: Interfaz Asterisk	¡Error! Marcador no definido.
Ilustración 6: Interfaz CUCM (Communication Manager Cisco)	¡Error! Marcador no definido.
Ilustración 7: Cableado Router y Switch	¡Error! Marcador no definido.
Ilustración 8: Cableado Access Point	¡Error! Marcador no definido.
Ilustración 9: Cableado teléfonos Cisco y ATA D-Link	¡Error! Marcador no definido.
Ilustración 10: Proceso de configuración de equipos	¡Error! Marcador no definido.
Ilustración 11: Configuración de Vlans en Switch	¡Error! Marcador no definido.
Ilustración 12: Configuración de DHCP y encapsulamiento en Router	¡Error! Marcador no definido.

Gemelo Digital en el Desarrollo de Motores de Combustión Interna (IC)

OBJETIVOS

OBJETIVO GENERAL

Desarrollar un gemelo digital para un motor de combustión interna que incorpore sensores de temperatura y proximidad, un modelo virtual en Unity y un servidor MQTT para la transmisión en tiempo real de datos, con el propósito de optimizar el monitoreo, la simulación y el análisis del rendimiento del motor.

OBJETIVOS ESPECÍFICOS

- Desarrollar e integrar los sensores adecuados para medir la temperatura y la proximidad en el motor de combustión interna, Implementando el hardware necesario para la recolección de datos en el motor
- Diseñar y desarrollar un modelo 3D del motor de combustión interna en Unity, con la funcionalidad para recibir y visualizar datos de los sensores en el modelo virtual
- Configurar un servidor MQTT para la transmisión y recepción eficiente y robusta de datos en tiempo real, que facilite la comunicación entre los sensores y el modelo virtual

ALCANCE

La integración y sincronización de los componentes garantizará que el modelo virtual refleje con precisión las condiciones del motor. Además, se desarrollarán herramientas de monitoreo y análisis dentro del gemelo digital, permitiendo la interpretación de datos y generación de informes sobre el rendimiento y estado del motor. Se realizarán pruebas exhaustivas para verificar la precisión y fiabilidad del sistema, ajustando y mejorando según los resultados obtenidos. Este proyecto no incluirá la optimización física del motor ni la integración con otros sistemas externos, y las mejoras posteriores

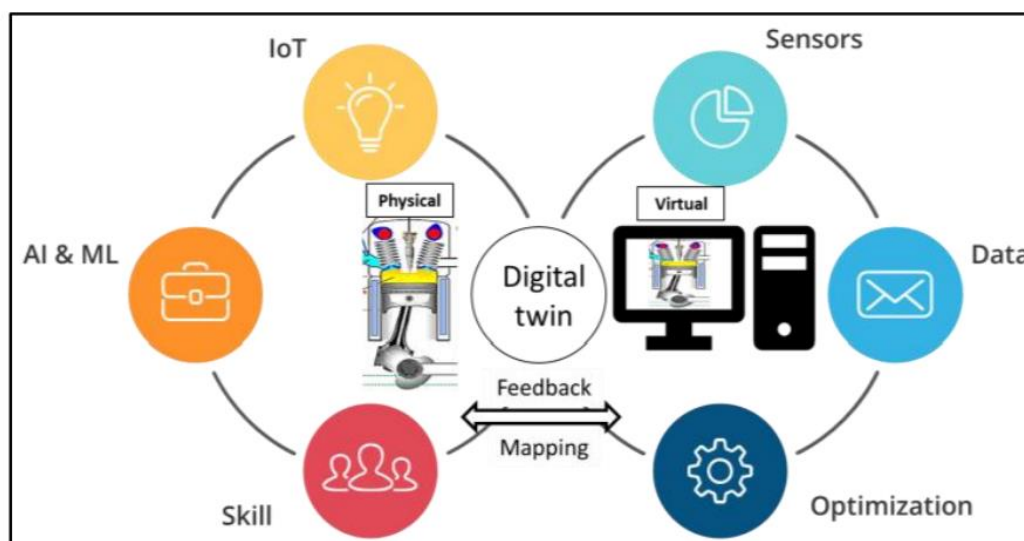
LIMITACIONES

La calidad de los datos transmitidos por estos sensores es crucial para el funcionamiento correcto del gemelo digital. Si los sensores no proporcionan datos precisos o presentan fallos en su funcionamiento, la representación virtual del motor en Unity y las decisiones basadas en estos datos pueden no ser exactas. Además, cualquier retraso o pérdida de datos en la transmisión a través del servidor MQTT puede afectar la sincronización en tiempo real entre el motor físico y su gemelo digital

INTRODUCCION

Los investigadores pueden explorar y refinar diseños en un entorno virtual antes de construir un prototipo físico utilizando gemelos digitales. Esta característica acelera la fase de diseño al permitir iteraciones rápidas y modificaciones basadas en los resultados de las simulaciones. Los investigadores pueden experimentar con diversos ajustes, configuraciones y materiales para optimizar la eficiencia y el rendimiento (Sørensen et al., 2022; Y. Wu et al., 2022).

el modelo de gemelo digital de cinco dimensiones contiene tres tipos de elementos: entidad física, modelo virtual y sistema de servicio, cada uno de los cuales es un nodo de diferente naturaleza. Los datos, almacenados en el pool de recursos, son un conjunto dinámico debido a la adición continua de nuevos datos. En el proceso de producción y operación de las entidades físicas, existe una relación de coordinación entre ellos. Existe una relación de control entre los sistemas de servicios, debido a su función en secuencia. Las entidades físicas son simuladas por modelos virtuales, mientras que el modelo virtual funciona a través de una serie de sistemas de servicios



Cuadro 1: Esquema de un gemelo digital Motor de combustión interna

MARCO TEÓRICO

Predicción y optimización del rendimiento: Al combinar modelado basado en física con datos de sensores en tiempo real, los gemelos digitales proporcionan una imagen del rendimiento del motor bajo diferentes condiciones operativas. Esto permite a los investigadores simular y optimizar parámetros de rendimiento como la eficiencia del combustible, la potencia y las emisiones. Las pruebas iterativas en un entorno virtual identifican las opciones de diseño que ofrecen el mejor compromiso entre rendimiento, eficiencia e impacto ambiental (Xu et al., 2021; Söderäng et al., 2022).

Estrategia de emisiones: En una era de leyes estrictas sobre emisiones, los gemelos digitales pueden jugar un papel importante en la creación y mejora de técnicas de control de emisiones. Los diseñadores pueden probar el rendimiento de diversas estrategias de reducción de emisiones simulando la combustión y controlando las emisiones en una réplica virtual. Esto permite diseñar motores que cumplan o superen los límites de emisiones mientras se mantienen lo más eficientes posible (Xu et al., 2022; Zhao et al., 2021).

Ahorro de tiempo y costos de desarrollo: Integrar gemelos digitales reduce sustancialmente el tiempo de desarrollo de motores IC. Al optimizar diseños en la práctica, se requieren menos prototipos físicos, optimizando recursos

El núcleo del gemelo digital es el modelo y los datos. El modelo incluye Entidad Física (PE), Modelo Virtual (VM), Sistema de Servicios (Ss), Datos del Gemelo Digital (DD) y Conexión (CN). PE, VM y Ss pueden interactuar entre sí. La información entre PE, VE y Ss se transmite a DD a través de un flujo de datos. DD impulsa PE, VE y Ss



MODELO DE SUPER-RED DE TRES CAPAS

Capa de Datos:

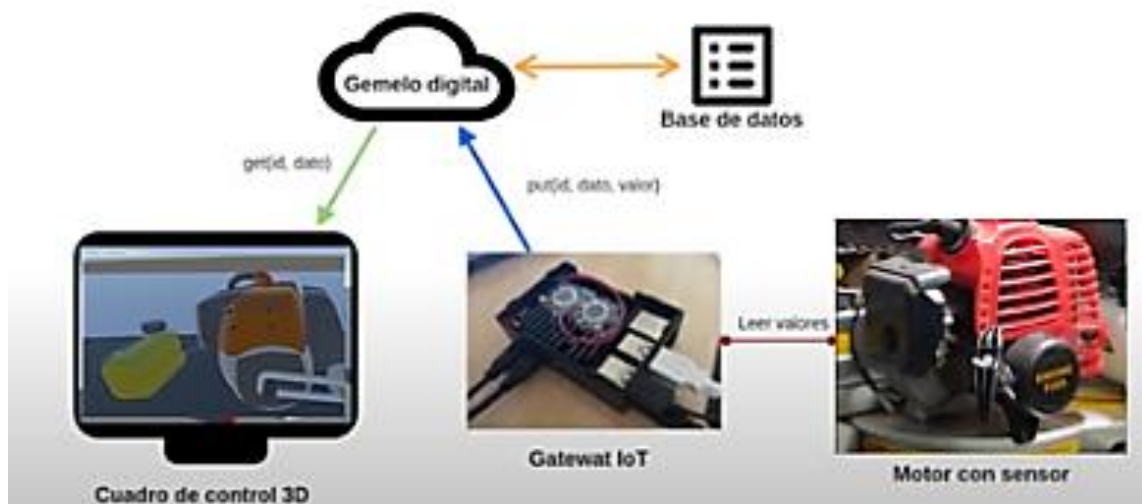
- **Recopilación de Datos:** Esta capa se encarga de la recopilación de datos de diversas fuentes, incluyendo sensores, sistemas de gestión de datos y bases de datos externas.
- **Almacenamiento de Datos:** Los datos recopilados se almacenan en una base de datos centralizada que permite un acceso rápido y eficiente para el análisis y la simulación.

Capa de Red:

- **Integración de Datos:** Los datos de diferentes fuentes se integran en esta capa, asegurando que estén sincronizados y disponibles para su análisis en tiempo real.
- **Comunicación:** La capa de red facilita la comunicación entre diferentes componentes del sistema, permitiendo el intercambio de datos y la colaboración en tiempo real.

Capa de Modelo:

- **Modelado y Simulación:** En esta capa se desarrollan y ejecutan modelos digitales que simulan el comportamiento de los productos mecánicos en diversas condiciones operativas.
- **Análisis de Datos:** Los datos recopilados se analizan utilizando técnicas avanzadas de análisis y algoritmos de aprendizaje automático para identificar patrones y predecir posibles fallos.



CONSTRUCCIÓN DEL MODELO DE SUPER-RED DE DATOS

1) Capa física de datos (SN_{PE})

Sensores:

- Sensor de Temperatura (TempSensor)
- Sensor de Proximidad (ProxSensor)

Nodos:

$$N_{PE} = \{\text{TempSensor}, \text{ProxSensor}\}$$

Aristas:

$$E_{PE-PE} = \{\text{eTemp-Prox}\}$$

Modelo de Red:

$$SN_{PE} = (\{p1, p2\}, \{(p1, p2)\})$$

2) Capa virtual de datos (SN_{VE})

Modelo Virtual Unity:

- Modelo Virtual de Temperatura (TempModel)
- Modelo Virtual de Proximidad (ProxModel)

$$N_{VE} = \{\text{TempModel}, \text{ProxModel}\}$$

$$\text{Aristas: } E_{VE-VE} = \{\text{eTempModel-ProxModel}\}$$

Modelo de Red Virtual:

$$\begin{aligned} SN_{VE} &= (N_{VE}, E_{VE-VE}) \\ SN_{VE} &= (\{v1, v2\}, \{(v1, v2)\}) \end{aligned}$$

3) Sistema de servicio de datos (SN_{SS})

Sistemas de Servicios:

- Servicio de Monitoreo de Temperatura (TempService)
- Servicio de Monitoreo de Proximidad (ProxService)

$$N_S = \{\text{TempService}, \text{ProxService}\}$$

$$\text{Aristas: } E_{S-S} = \{\text{eTempService-ProxService}\}$$

Modelo de Red de Servicio:

$$\begin{aligned} SN_{SS} &= (N_S, E_S) \\ SN_{SS} &= (\{s1, s2\}, \{(s1, s2)\}) \end{aligned}$$

4) Mapeo entre SN_{PE} y SN_{VE}

Cada entidad física tiene una relación de mapeo uno a uno con un modelo virtual.

$$M_{PE-VE} = \{(TempSensor, TempModel), (ProxSensor, ProxModel)\}$$

$$M_{PE-VE} = \{(P_j, VE_k) | j \in SN_{PE}, k \in SN_{VE}\}$$

$$M_{PE-VE} = \{(p1, v1), (p2, v2)\}$$

5) Mapeo entre $SN_{PE}(SN_{VE})$ y SN_{SS}

Los datos recopilados por cada entidad física corresponden a los modos funcionales de uno o varios sistemas de servicio.

$$S(PE_i) = \{S_i | S_i \in S, \varphi(PE_i, S_i) = 1\}$$

En este caso:

$$S(TempSensor) = \{TempService\}$$

$$S(ProxSensor) = \{ProxService\}$$

6) Mapeo entre $SN_{VE}(SN_{PE})$ y SN_{SS}

Los modelos virtuales se mapean a los sistemas de servicio correspondientes.

$$S(TempModel) = \{TempService\}$$

$$S(ProxModel) = \{ProxService\}$$

7) Representación Formal de la Super-Red

La super-red de datos se expresa como:

$$SN = (PE, VE, S, M_{PE-VE}, M_{PE-SS}, M_{VE-SS})$$

8) Fórmulas y Ecuaciones

Sensores y Datos Físicos

- Temperatura: $T(t)$
- Proximidad: $P(t)$


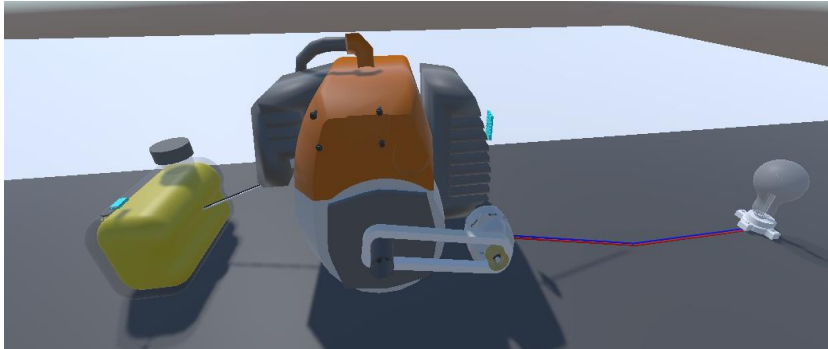
Modelo Virtual

- Actualización de Temperatura: $TempModel(t) = f(T(t))$
- Actualización de Proximidad $ProxModel(t) = f(P(t))$

Servicio de Monitoreo

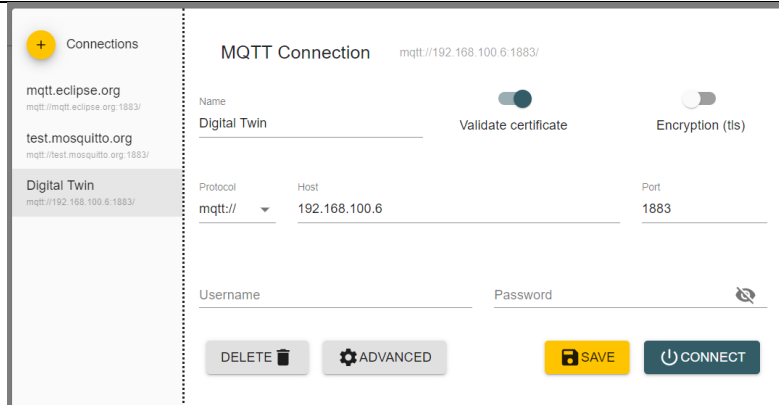
- Servicio de Temperatura: $TempService(T(t))$
- Servicio de Proximidad: $TempService(P(t))$

Herramientas para la implementación del gemelo digital

Servidor	Interfaz
<p>Arduinio</p> <p>Hardware: Sensores de temperatura (como el DHT11 o DHT22) y proximidad (como el HC-SR04).</p> <p>Software: Arduino IDE para programar y leer los datos de los sensores</p> <p>Serial Communication: Para enviar datos del Arduino a una computadora.</p> <p>WiFi/Bluetooth Modules: Como el ESP8266 o el ESP32, para transmitir datos de forma inalámbrica</p>	
<p>Unity</p> <p>Unity Engine: Para crear el modelo digital del motor y la simulación.</p> <p>C# Scripts: Para integrar y procesar los datos en Unity.</p> <p>Unity MQTT Client: Para recibir datos en tiempo real de los sensores Arduino</p>	

MQTT

MQTT Mosquitto: Un protocolo ligero de mensajería para la comunicación entre dispositivos y aplicaciones



PROTEUS

Simulación mixta:

Conocido por su capacidad de simular tanto circuitos analógicos como digitales de manera integrada. Esto incluye la simulación de microcontroladores, carga programas reales y ver cómo interactúan con otros componentes del circuito

Desarrollo de firmware:

Proteus permite probar el código del firmware en microcontroladores simulados dentro de un entorno de circuito completo

Compatibilidad con microcontroladores:

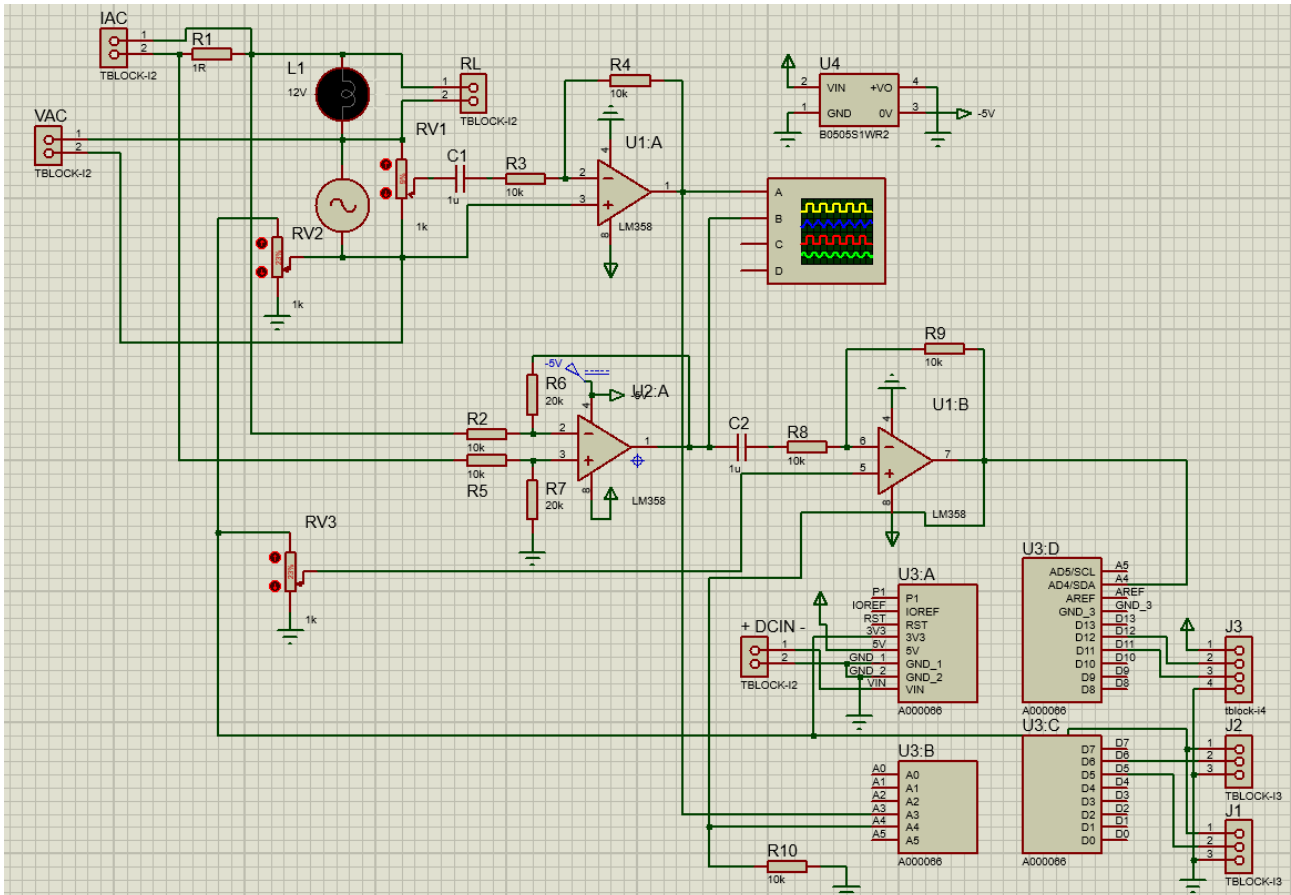
Soporta una amplia gama de microcontroladores como PIC, AVR, Arduino, ARM, entre otros



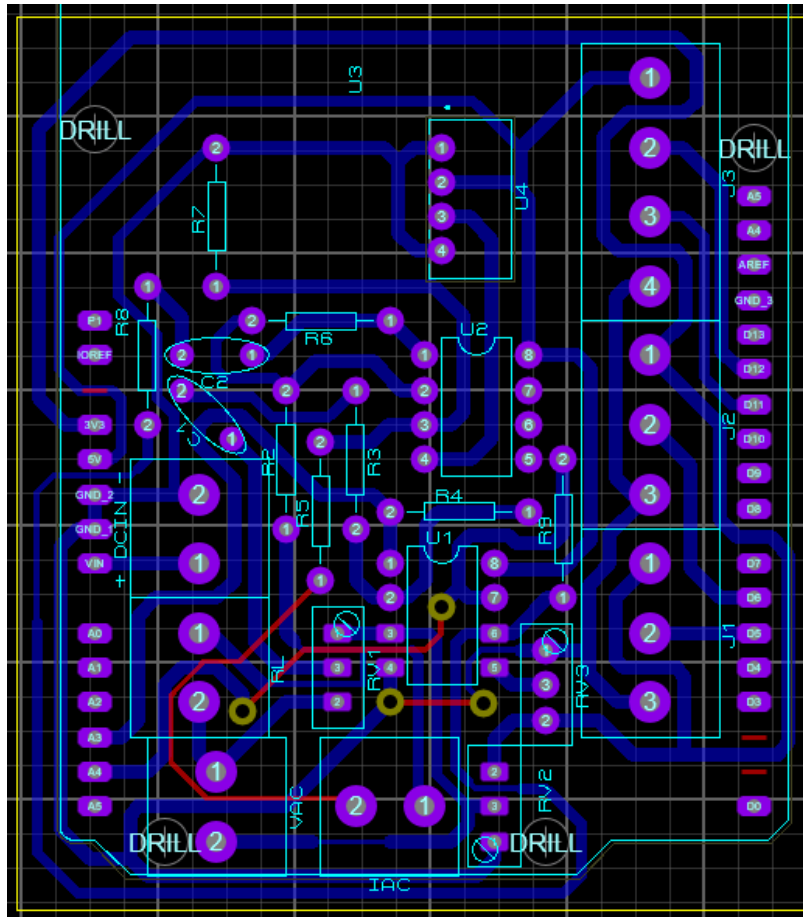
Desarrollo

Adaptación en Proteus

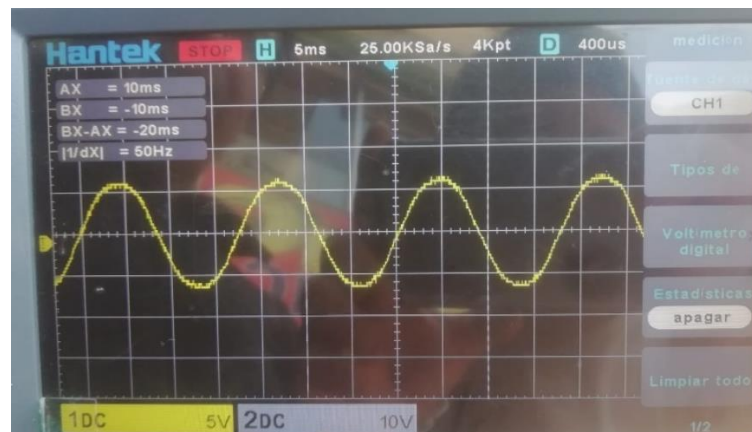
Sistema de control que integra varios componentes, como un relé, amplificadores operacionales (op-amps), un transformador, y una unidad de microcontrolador (MCU).



- VAC: Corriente alterna (VAC)
- (RL): Relé
- (L1): Bobina
- Resistencias (R1, R3, R4, etc.)
- Potenciómetros (RV1, RV2, RV3)
- Condensadores (C1, C2)
- Amplificadores Operacionales (LM358) U1 y U1:B , U2:A , U4
- Microcontrolador U3 (A, B, C, D) , pines A0, A1, A2, etc , salidas digitales (D2, D3, D4, etc.)
- Conexiones de E/S , conectores J2 y J3
- Osciloscopio (U5)4



- **Pistas y Pads:** Las pistas en azul indican las conexiones eléctricas entre los componentes, y los pads (círculos morados) son los puntos de conexión donde se soldarán los componentes.
- **Conectores y Terminales:** Los conectores identificados como +DCIN, IAC, VAC, y otros que se ven en el esquemático, están claramente definidos en el PCB para la entrada y salida de energía, así como para las señales de control.
- **Rutas de Señales:** Las pistas que interconectan los componentes pasivos y activos (como resistencias, condensadores, op-amps, y el microcontrolador) están distribuidas eficientemente para minimizar la longitud de las conexiones y reducir posibles interferencias o ruido en las señales.
- **Alimentación:** La distribución de la alimentación (+5V, GND) está bien organizada, con rutas claras hacia los componentes que requieren energía



Convertir una señal analógica, muestreada y convertida a una señal digital mediante un convertidor ADC, a una señal en unidades de voltaje.

El valor RMS (Root Mean Square) es particularmente útil para señales senoidales porque proporciona una medida del valor efectivo o de potencia de la señal, que es más representativa del trabajo que la señal puede realizar en una carga resistiva. Para una señal senoidal pura, el valor RMS es conocido y tiene una relación directa con el valor pico de la señal

$$V_{rms} = \frac{V_{pico}}{\sqrt{2}}$$

$$3.3 \text{ V}$$

$$ADC \Rightarrow 12 \text{ bits} \Rightarrow 0 - 4095$$

$$\frac{8.8 \text{ v}}{4095} = R$$

$$\frac{(ADC * R)^2}{\sum \left(\frac{1}{\# \text{ Muestras}} \right)}$$

$$= \sqrt{} \Rightarrow RMS$$

CALCULAR LA ENERGÍA DE ENTRADA**Fórmula:**

$$\text{Energía de entrada (J)} = \text{Masa de combustible (kg)} \times \text{Poder calorífico (J/kg)}$$

Obtener la masa de combustible consumido:

- Medir el consumo de gasolina en un tiempo determinado (en litros).
- Convertir el volumen de gasolina a masa utilizando la densidad de la gasolina (aproximadamente 0.74 kg/l para gasolina).

$$\text{Masa de combustible (kg)} = \text{Volumen de combustible (l)} \times 0.74 \text{ kg/l}$$

Obtener el poder calorífico de la gasolina:

- El poder calorífico de la gasolina es aproximadamente 44,000,000 J/kg.

Calcular la energía de entrada:

- Energía de entrada (J) = Masa de combustible (kg) x 44, 000, 000J/kg

Fórmula:

$$\text{Potencia de entrada (W)} = \frac{\text{Energía de entrada (J)}}{\text{Tiempo (s)}}$$

- Consumo de combustible: **0.5 litros/hora**
- Masa de combustible: **0.37 kg**
- Energía de entrada: **16.200,000J**
- Tiempo de funcionamiento: **3600 s**

$$\text{Potencia de entrada: } 4322.22 \text{ W}$$

POTENCIA DE ENTRADA

$$P_E = M_c \times PC$$

$$P_E = \text{Potencia de entrada}$$

$$M_c = \text{Masa Combustible}$$

$$PC = \text{Poder Calorífico}$$

POTENCIA DE SALIDA




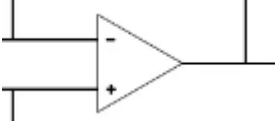

$$P_s = I_s * R_s$$

DESPERDICIO

$$P_D = \frac{P_E}{P_S}$$

Configuración de Arduino

Programa los sensores de temperatura y proximidad para leer y enviar datos, utilizando módulos de comunicación (WiFi/Bluetooth) para enviar los datos a un servidor o computadora

SENSOR	CONFIGURACIÓN
<p>DHT11</p> 	<pre> DHT.read(16); temperatura=DHT.temperature; Serial.println(temperatura); sprintf(datos1, "%d",temperatura); digitalWrite(19,1); </pre>
<p>ULTRASONIC HC-SR04</p> 	<pre> distancia_t=pulseIn(23,1); distancia_t=distancia_t*331/2000; distancia=map(distancia_t,0,150,0,100); sprintf(datos2, "%d",distancia); </pre>
<p>Rs (shunt)</p>  <p>AMPLIFICADOR DIFERENCIAL</p> 	<pre> it=0; for(a=1;a<10000;a++) { i=analogRead(36); i=i-2047; i=i*0.48/1281.13; i=i*i; it=it+i; } i=sqrt(it); //if(i<250) i=0; corriente=i; sprintf(datos3, "%d",corriente); </pre>
<p>AMPLIFICADOR OPERACIONAL</p> 	<pre> vt=0; for(a=1;a<10000;a++) { v=analogRead(34); v=v-2047; v=v*12.25/1281.13; v=v*v; vt=vt+v; } v=sqrt(vt); if(v<250) v=0; voltaje=v; sprintf(datos4, "%d",voltaje); </pre>

ENVIO DE DATOS

```

const char* ssid      = "MOVISTAR CALDERON 2.4G";
const char* password = "JDGC2010";

unsigned long distancia_t;
int a=0;
float vt=0;
float v=0;

float it=0;
float i=0;

int temperatura=0;
int distancia=0;
int corriente=0;
int voltaje=0;

char *server = "192.168.100.234";//"broker.emqx.io";
int port = 1883;

char datos1[5];
char datos2[5];
char datos3[5];
char datos4[5];

String results = "";

```

```

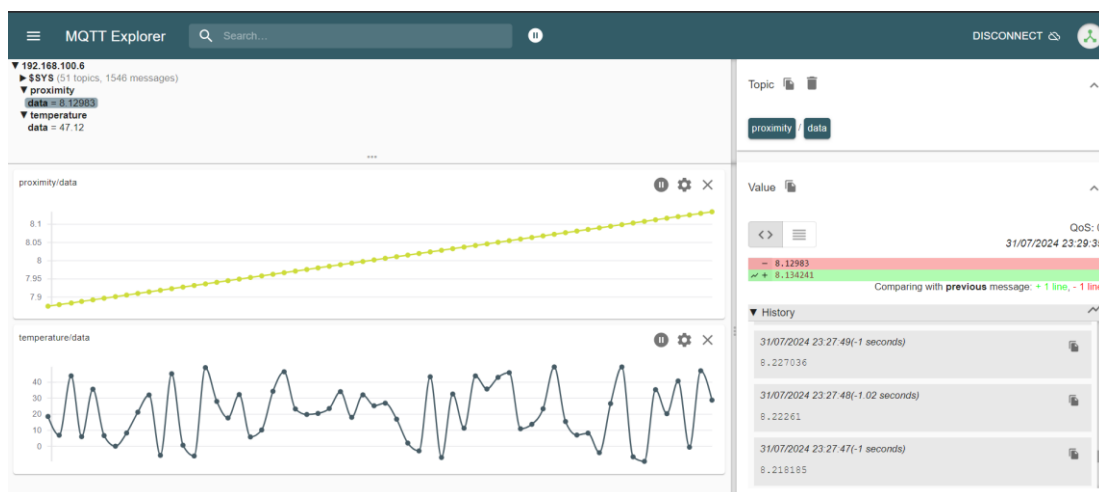
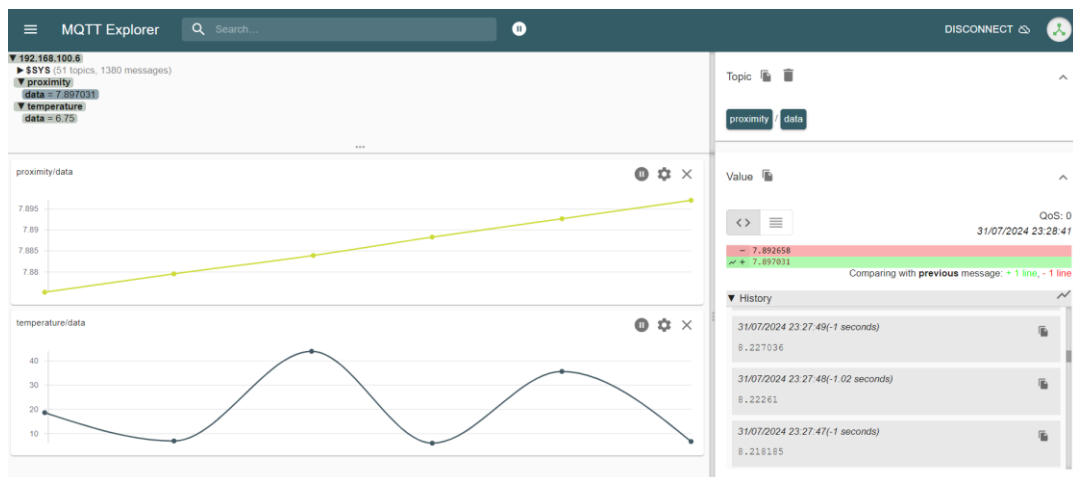
mqttClient.publish("sensor/temperatura",datos1);
delay(10);
mqttClient.publish("sensor/distancia",datos2);
delay(10);
mqttClient.publish("sensor/corriente",datos3);
delay(10);
mqttClient.publish("sensor/voltaje",datos4);
delay(1000);
tiempo_retardo = millis();

```

Plataforma de Monitoreo (Mosquitto MQTT):

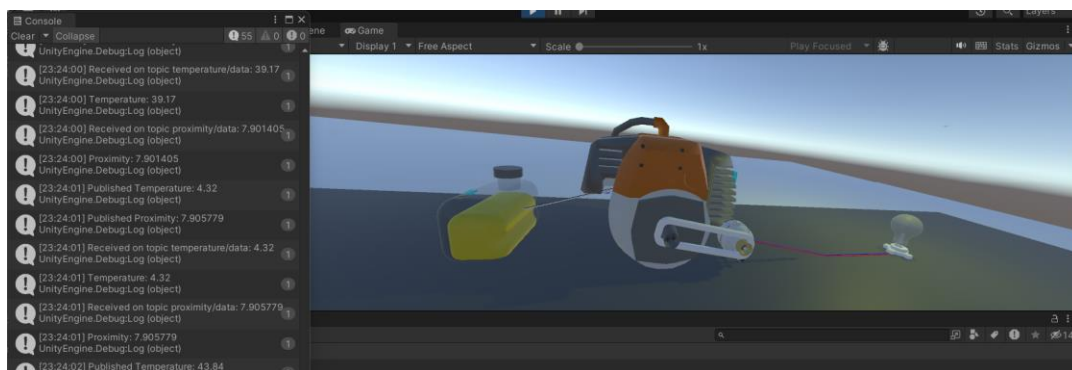
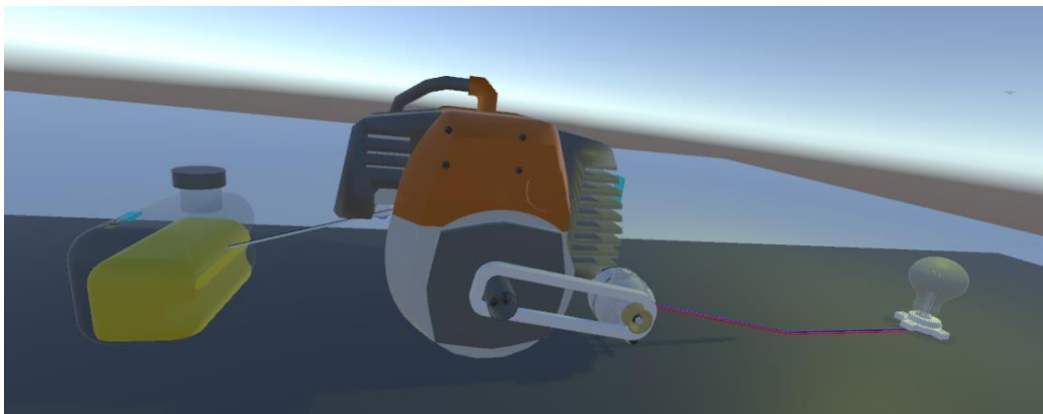
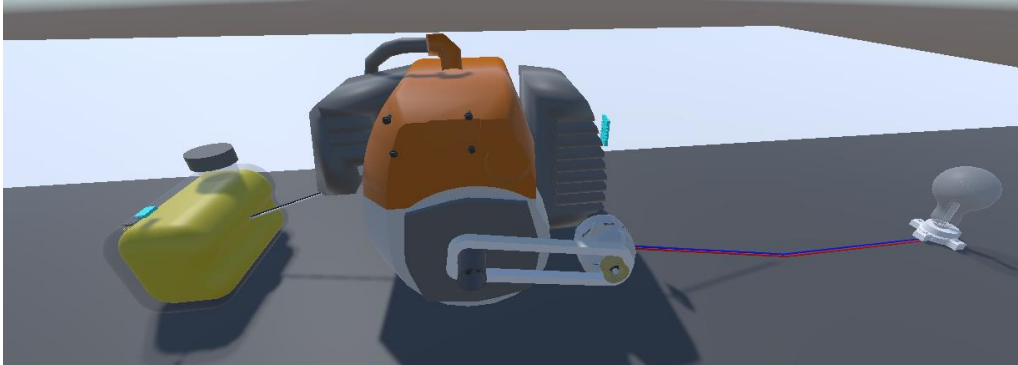
Configura un servidor MQTT para recibir datos de los sensores Arduino.

The screenshot shows the Mosquitto MQTT web interface. On the left, there's a sidebar with a list of connections: 'mqtt.eclipse.org', 'test.mosquitto.org', and 'Digital Twin' (selected). The main area is titled 'MQTT Connection' and shows the connection string 'mqtt://192.168.100.6:1883/'. Below this, there are fields for 'Name' (Digital Twin), 'Protocol' (mqtt://), 'Host' (192.168.100.6), and 'Port' (1883). There are also checkboxes for 'Validate certificate' and 'Encryption (tls)'. At the bottom, there are fields for 'Username' and 'Password', and buttons for 'DELETE', 'ADVANCED', 'SAVE', and 'CONNECT'.



Integración con Unity:

- Desarrolla el modelo digital del motor en Unity.
- Implementa scripts en C# para conectar Unity al servidor MQTT y recibir datos en tiempo real.
- Sincroniza los datos de los sensores con la simulación en Unity



CONCLUSIONES

- **Mejorar el Monitoreo en Tiempo Real:** La integración de sensores de temperatura y proximidad con un servidor MQTT y un modelo virtual en Unity ha permitido un monitoreo detallado y en tiempo real del motor de combustión interna. Esto facilita la detección temprana de posibles fallos y mejora la capacidad de respuesta ante problemas operativos.
- **Aumentar la Precisión en el Análisis de Datos:** La capacidad de visualizar y analizar los datos recolectados de los sensores en un entorno virtual ha proporcionado una comprensión más profunda del comportamiento y rendimiento del motor. Las herramientas de análisis implementadas dentro del gemelo digital permiten interpretar los datos de manera más efectiva, generando informes precisos que pueden ser utilizados para optimizar el mantenimiento y operación del motor.
- **Facilitar la Simulación y Entrenamiento:** El modelo virtual del motor en Unity no solo sirve para el monitoreo, sino también como una herramienta de simulación y entrenamiento. Los operadores y técnicos pueden interactuar con el gemelo digital para aprender sobre el funcionamiento del motor, practicar procedimientos de mantenimiento y evaluar escenarios hipotéticos sin riesgo para el equipo físico.
- **Evidenciar la Importancia de la Precisión de los Sensores:** Durante el desarrollo del proyecto, se destacó la importancia de la selección y calibración adecuada de los sensores de temperatura y proximidad. La precisión y fiabilidad de los datos recolectados son esenciales para el correcto funcionamiento del gemelo digital y para tomar decisiones informadas basadas en estos datos que se han recopilado a lo largo de todo el proyecto con la finalidad.

ANEXOS

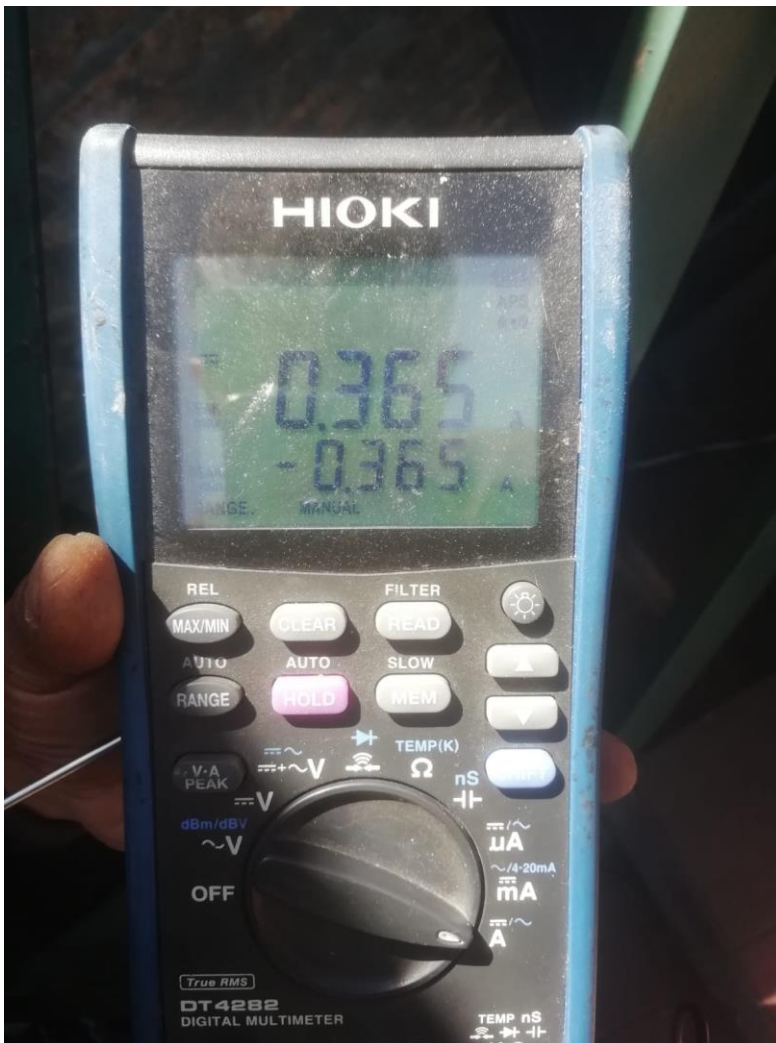
```
Simbolo del sistema x + v
c:\Program Files\mosquitto>mosquitto_sub -d -t prueba_gemelodigital
Client null sending CONNECT
Client null received CONNACK (0)
Client null sending SUBSCRIBE (Mid: 1, Topic: prueba_gemelodigital, QoS: 0,
Options: 0x00)
Client null received SUBACK
Subscribed (mid: 1): 0
Client null received PUBLISH (d0, q0, r0, m0, 'prueba_gemelodigital', ... (1
9 bytes))
hola gemelo digital
Client null sending PINGREQ
Client null received PINGRESP

Simbolo del sistema x + v
Microsoft Windows [Versión 10.0.22631.3880]
(c) Microsoft Corporation. Todos los derechos reservados.

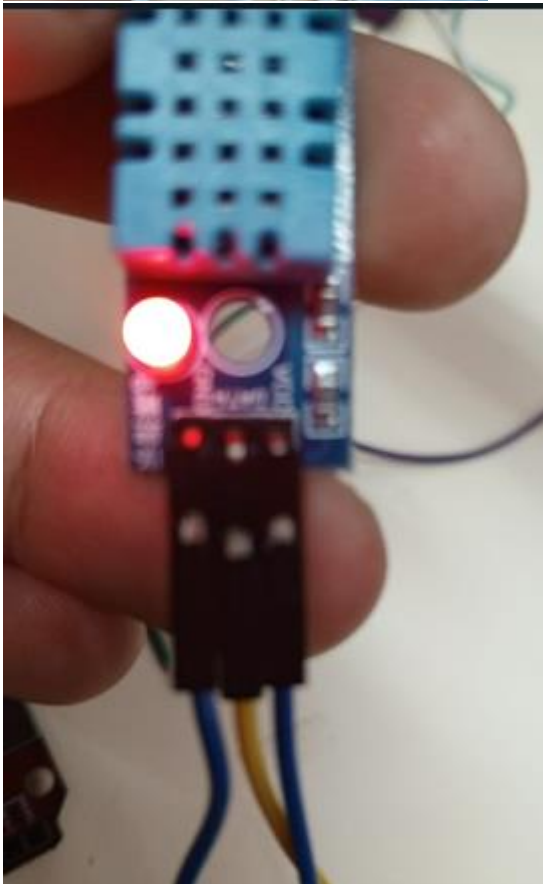
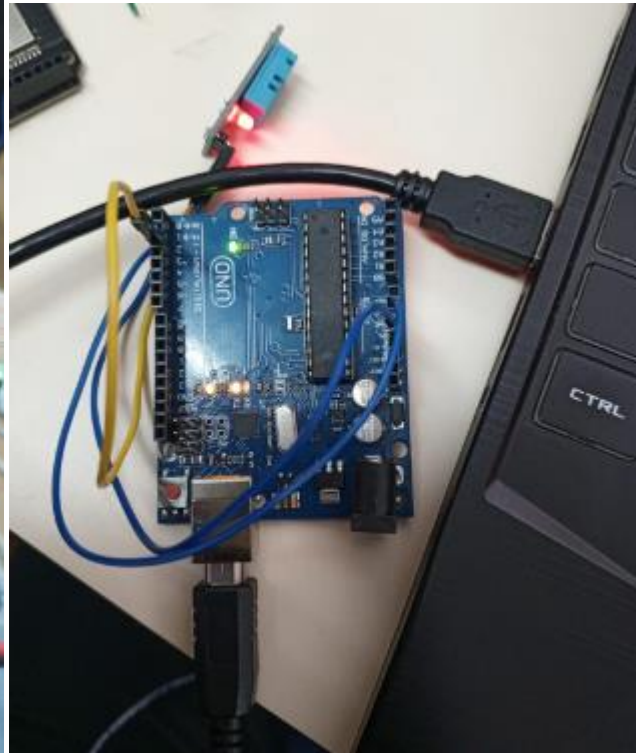
C:\Users\LENOVO.USER>
C:\Users\LENOVO.USER>cd c:/program files/mosquitto

c:\Program Files\mosquitto>mosquitto_pub -h 192.168.100.6 -t prueba_gemelo
dital -m "hola gemelo digital"

c:\Program Files\mosquitto>
```







Bibliografía

<https://journal.cbiores.id/index.php/jese/article/view/5/5>

https://www.researchgate.net/publication/359102696_Maintenance_Digital_Twin_using_vibration_data

<https://www.nature.com/articles/s41598-021-04545-5>

<file:///C:/Users/LENOVO.USER/Downloads/s41598-021-04545-5.pdf>

<https://www.youtube.com/watch?v=4V-wLR35HO0>