

Universidad Central Del Ecuador



Tema:

Generación de Energía eléctrica mediante baldosas piezoelectricas

Integrantes:

Wilmer Andrango, Fabian Cueva, Bryan Quishpe, Jimmy Quimba, Cesar Cueva

Grupo:

4

Asignatura:

Infraestructura de TI II

Desarrollo.....	4
Materiales	4
Entradas/Salidas (GPIO):	5
Alimentación:	6
Procedimiento.....	14
Configuración de los discos piezoeléctricos.....	14
Gemelo digital	17
Manual del modelo físico y digital.....	26
2. OBJETIVOS	25
2.1 Objetivo General	27
2.2 Objetivos Específicos	27
3. MATERIALES	27
4. HERRAMIENTAS	28
5. PROCEDIMIENTO.....	28
5.1 Preparación de la Base	28
5.2 Instalación de los Discos Piezoeléctricos	28
5.3 Conexión protoboard.....	29
5.4 Conexión ESP32	30
5.5 Conexión INA219 I2C	30
5.6 Almacenamiento.....	31
5.7 Ensamblaje Final	31
6. Pruebas y validación modelo físico	31

7. Seguridad modelo físico	32
8. Programación de Arduino	32
9. Conexión a Base de Datos	33
Configuración de la base de datos	41
Creación de la conexión	42
Comprobación de la conexión	42
Obtención y depuración de los datos	42
Validación de los datos	43
Inserción de los datos en la base de datos	43
10. Implementación del Gemelo Digital en Unity	42
Diagrama eléctrico	53
Resultados del experimento	55
Conclusiones	56
Recomendaciones	56
Bibliografía	58
Anexos.....	60

Introducción

El avance de la Industria 4.0 e IoT (Internet de las Cosas) ha impulsado la búsqueda de fuentes de energía eficientes, sostenibles y autónomas para alimentar dispositivos inteligentes y sistemas interconectados. En este contexto, la energía **piezoeléctrica** se presenta como una alternativa innovadora capaz de convertir vibraciones mecánicas en *energía eléctrica*, lo que la convierte en una solución viable para aplicaciones tecnológicas.

Este proyecto es una propuesta para la Universidad Central del Ecuador (UCE), institución que busca invertir en energías alternativas y avanzar hacia el concepto de universidad inteligente. La iniciativa plantea el uso de materiales piezoeléctricos para recolectar y almacenar energía en el campus universitario, con el fin de alimentar sensores, dispositivos IoT y sistemas automatizados. Esto no solo contribuirá a la reducción del consumo energético convencional, sino que también promoverá un entorno más eficiente y sostenible.

El objetivo principal de este estudio es demostrar la viabilidad de la energía piezoeléctrica dentro del ecosistema, evaluando su integración con tecnologías emergentes para la creación de infraestructuras energéticamente autónomas.

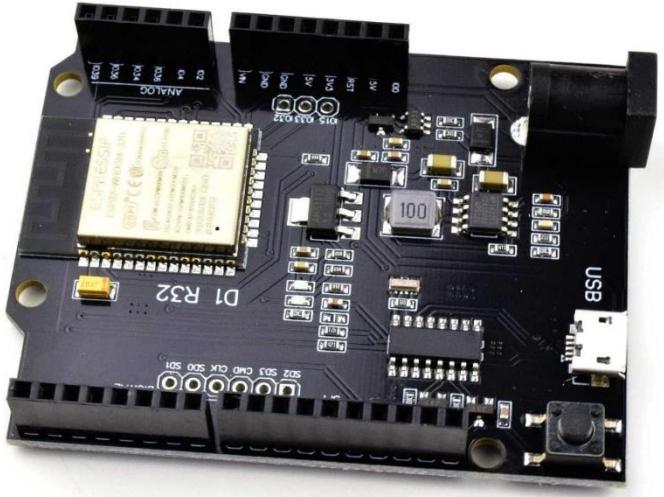
Este análisis busca demostrar el potencial de la energía piezoeléctrica, su aplicabilidad en entornos académicos y su impacto en el desarrollo de la UCE como una universidad referente en innovación y sostenibilidad. Además, el proyecto incluye el diseño y prueba de un prototipo con recolección de datos en tiempo real y su correspondiente modelo digital.

Desarrollo

Materiales

1. ESPDUINO_32

Es una placa de desarrollo basada en el microcontrolador ESP32, que combina las funcionalidades del popular ESP32 con la facilidad de uso y el formato de Arduino. Está diseñada para ser compatible con los shields (placas de expansión) de Arduino, lo que permite a los desarrolladores aprovechar las capacidades avanzadas del ESP32, como conectividad Wi-Fi y Bluetooth[1].



Características:

Microcontrolador ESP32:

- Procesador dual-core de 32 bits (Xtensa LX6).
- Frecuencia de reloj de hasta 240 MHz.
- Conectividad Wi-Fi (802.11 b/g/n) y Bluetooth (clásico y BLE).
- Memoria Flash integrada (generalmente 4 MB).
- Memoria RAM de 520 KB.

Entradas/Salidas (GPIO):

- Múltiples pines GPIO digitales.
- Entradas analógicas (ADC).
- Salidas PWM.

- Interfaces de comunicación como UART, I2C, SPI.

Alimentación:

- Voltaje de operación de 3.3V.
- Puede ser alimentado a través del conector USB o mediante el jack de alimentación de Arduino.

Usos comunes:

- Proyectos de IoT (Internet de las Cosas) debido a su conectividad Wi-Fi y Bluetooth.
- Automatización del hogar.
- Robótica.
- Prototipado rápido de dispositivos electrónicos.

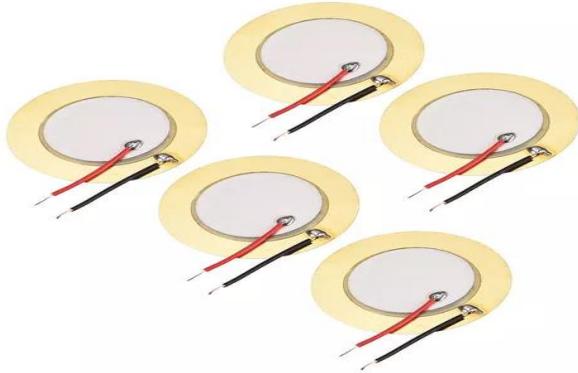
2. DISCOS PIEZOELÉCTRICOS

Un sensor piezoeléctrico es un dispositivo eléctrico que mide la aceleración, la presión o la fuerza y convierte esa información en una señal eléctrica. En numerosos sectores, estos sensores se utilizan principalmente para el control de procesos, el control de calidad y la investigación y el desarrollo. Los usos de este sensor incluyen equipos aeroespaciales, médicos y nucleares, así como su uso como sensor de presión en los paneles táctiles de los teléfonos móviles[2].

Varían en tamaño, desde pequeños discos de unos pocos milímetros de diámetro hasta tamaños más grandes, adaptándose a diversas aplicaciones.

Los sistemas de transducción piezoeléctricos funcionan mediante la polarización eléctrica de ciertos materiales al ser deformados por una fuerza. Esta polarización genera un campo eléctrico que permite la conversión de energía mecánica en energía eléctrica. Además, al aplicar un campo eléctrico, los materiales se deforman y vibran, produciendo ondas acústicas. Estas ondas interactúan con su entorno, permitiendo medir propiedades del medio a

través del campo eléctrico del sensor. Los sensores piezoelectrómicos pueden detectar variaciones en presión, temperatura, masa, densidad o viscosidad de fluidos [2].



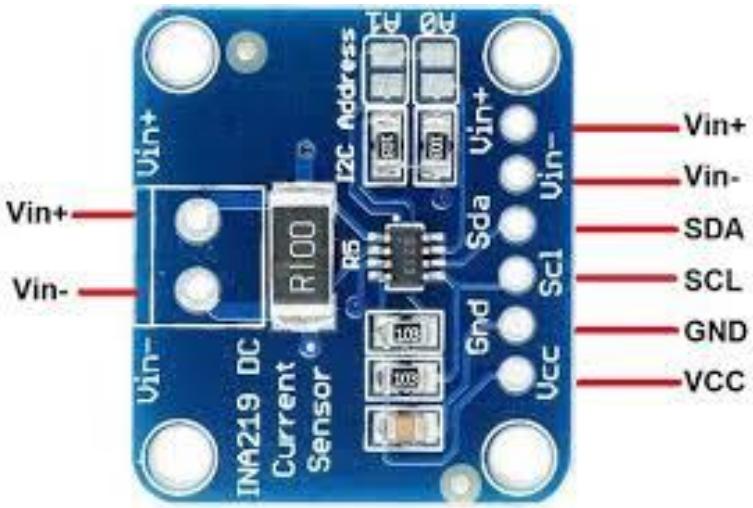
3. SENSOR INA219

El INA219 es un sensor fabricado por Texas Instruments que permite realizar medidas de tensión, intensidad y potencia en circuitos electrónicos.

El INA219 puede medir tensiones de 0V a 26V, pudiendo elegir entre dos escalas de 16V y 32V, con una precisión máxima de $\pm 0.5\%$.

La medición de corriente del módulo INA219 es ajustable por software, permitiendo rangos desde $\pm 3.2\text{A}$ con precisión de 0.8mA hasta $\pm 400\text{mA}$ con 0.1mA de precisión.

Compatible con alimentaciones de 3V3 y 5V, utiliza comunicación I2C, facilitando conexiones con procesadores como Arduino para medir intensidad, tensión y potencia eléctrica simultáneamente [3].



4. PROTOBOARD

Una placa de pruebas, o protoboard, es un tablero con orificios conectados eléctricamente que permite insertar elementos y cables para crear circuitos electrónicos. Fabricada con materiales aislantes y conductores, facilita el prototipado y la prueba de circuitos antes de su producción comercial. Se conoce también como "breadboard" o "placa protoboard", y su principal función es permitir la experimentación y modificación sencilla de proyectos y diseños electrónicos[4].

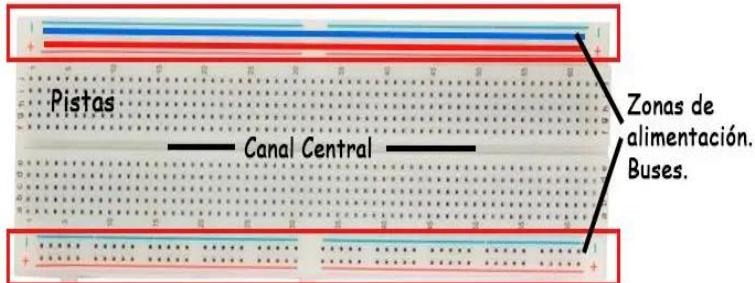
Básicamente un protoboard se divide en tres zonas:

Buses: se encuentran situados a ambos lados de la placa. Se representan mediante las líneas rojas (buses positivos o de voltaje) y azules (buses negativos o de tierra); no existe conexión física entre ellas. La fuente de alimentación se conecta, en general, en este lugar.

Canal central: se encuentra ubicado en la zona central, separando las zonas de conexión superior de las inferiores y se utiliza para la interconexión de los circuitos integrados, manteniendo aislados los pines de ambos lados del circuito integrado.

Pistas: los demás orificios de la protoboard son parte de las pistas. Las pistas se encuentran distribuidas en filas, las cuales están señaladas con números, y las columnas están señaladas con letras.

Las protoboards tienen un código de localización para sus orificios, donde las filas se enumeran con letras y las columnas con números.



5. CABLES JUMPER MACHO-HEMBRA Y MACHO-MACHO

Un jumper o saltador es un elemento que permite cerrar el circuito eléctrico del que forma parte dos conexiones.

Macho a macho (M/M): Ambos extremos del cable cuentan con conectores macho, ideales para conectar dos dispositivos con pines hembra. Son frecuentemente utilizados en protoboards para interconectar componentes [5].

Macho a hembra (M/F): Un extremo tiene un conector macho y el otro un conector hembra. Esta configuración es útil para conectar dispositivos con pines macho a aquellos con pines hembra, como sensores o módulos [6].

6.DIODO LED BLANCO

Es un componente electrónico que emite luz blanca cuando se le aplica una corriente eléctrica. Los LEDs (diodos emisores de luz) son semiconductores que, al ser energizados, liberan energía en forma de luz. En el caso de los LEDs blancos, esta luz se obtiene combinando diferentes longitudes de onda, lo que resulta en una emisión de luz blanca [7].

Características:

- Color: Blanco

- Voltaje: 3.2-3.4.
- Corriente: 25 mA.
- Longitud de onda: – 625-630nm.
- Intensidad lumínica: 14000-16000 mcd.



7. CABLE PUENTE

Un cable de conexión es un conductor eléctrico diseñado para establecer enlaces entre diferentes componentes o dispositivos en un circuito electrónico. Estos cables permiten la transmisión de señales eléctricas, datos o energía, facilitando la interconexión y comunicación entre elementos del sistema [8] .



8. DIODOS RECTIFICADORES 1N4007

Un diodo rectificador es un tipo de diodo diseñado específicamente para convertir corriente alterna (CA) en corriente continua (CC). Este proceso, conocido como rectificación,

es fundamental en diversas aplicaciones electrónicas y eléctricas. Los diodos rectificadores tienen dos terminales: el ánodo y el cátodo. Cuando se aplica una diferencia de potencial positiva en el ánodo y negativa en el cátodo, el diodo permite que la corriente fluya a través de él. Si se invierte la polaridad, el diodo bloquea el flujo de corriente, actuando como un interruptor unidireccional [9].



9. CAPACITOR ELECTROLÍTICO 10 uF 16 V.

Un capacitor electrolítico es un tipo de condensador que usa un líquido iónico conductor como una de sus placas. Son valiosos en circuitos eléctricos con relativa alta corriente y baja frecuencia. Este es especialmente el caso en los filtros de alimentadores de corriente, donde se usan para almacenar la carga, y moderar la tensión eléctrica de salida y las fluctuaciones de corriente en la salida rectificada. También son muy usados en los circuitos que deben conducir corriente continua pero no corriente alterna. Este es un capacitor que tiene una capacitancia de $47\mu\text{f}$ y soporta un voltaje máximo de 10 V [10].

10. BATERIAS RECARGABLES 18650 88MAH 3.7V A 4.2 V

Son celdas de iones de litio de forma cilíndrica, comúnmente utilizadas en aplicaciones que requieren una fuente de energía portátil y de alta capacidad.

Con una capacidad de 8800mAh, estas baterías pueden suministrar una corriente de 8800 miliamperios-hora, lo que indica una mayor duración de funcionamiento en comparación con baterías de menor capacidad.

Operan en un rango de voltaje de 3.7V (voltaje nominal) a 4.2V (voltaje máximo de carga), adecuado para dispositivos que requieren una fuente de alimentación estable y confiable [11].



11. PORTA PILAS X2-18650 -3.7 V

Es un soporte diseñado para alojar dos baterías recargables de tipo 18650, que son celdas cilíndricas de iones de litio con un voltaje nominal de 3.7V. Esta porta pilas permite conectar las dos baterías en serie, sumando sus voltajes y proporcionando una salida de 7.4V, adecuada para dispositivos que requieren un voltaje más alto [12].



Baldosa piezoeléctrica en Londres

En Reino Unido y Europa se han llevado a cabo casi 30 proyectos de Pavegen, tanto permanentes como temporales. Desde hace dos años cuatro de estas baldosas, colocadas en la Simon Langton Grammar School para chicos, cerca de Canterbury, obtienen energía de las pisadas de sus 1.100 estudiantes para mantener la iluminación del pasillo. Igualmente, han sido de utilidad en festivales de música para cargar teléfonos móviles y encender luces de tecnología LED.

Kemball-Cook desarrolló la idea de las baldosas durante su etapa universitaria al analizar el uso de energía solar y eólica en áreas urbanas. En 2009, su proyecto ganó atención mediática, lo que le llevó a fundar su empresa. Recibió un capital inicial de 800.000 dólares de familiares y amigos, y posteriormente logró más financiamiento de Ángeles de Negocios en Londres[12] .

Las baldosas están diseñadas para reducir al mínimo la huella de carbono. El revestimiento superior está hecho de goma reciclada de neumáticos y aproximadamente el 80% de los polímeros utilizados para el resto de los componentes puede ser reciclado. Un paso genera de media 7 vatios de electricidad, aunque depende del peso de la persona, y cada paso empuja 5 milímetros hacia abajo la goma, diferencia «imperceptible para los peatones», según Kemball-Cook [14].

El panel V3 genera hasta 5W de potencia mediante las pisadas de transeúntes, transformando energía cinética en electricidad para iluminar farolas en la misma calle [15] .

Las baldosas son impermeables al agua, lo que les permite resistir lluvia, nieve y hielo; y los ensayos realizados han evidenciado que podrían persistir por un mínimo de cinco años, aunque Kemball-Cook sostiene que lo más recomendable sería que perduraran 20.

Procedimiento

Configuración de los discos piezoeléctricos

- ✓ Se realizó una conexión una conexión mixta de los discos piezoeléctricos, para ello se soldaron dos grupos de tres discos piezoeléctricos en paralelo con los polos negativos y positivos. Luego se realizó una conexión en serie con los polos restantes de cada grupo para hacerlo un solo circuito.
- ✓ En el protoboard se conectaron las dos salidas de la conexión de los discos para que puedan conectarse con los diodos de corriente, dado que los discos producen corriente alterna AC, es necesario utilizar un circuito rectificador de onda completa para transformar la corriente alterna a corriente continua. Se utilizaron 4 diodos rectificadores.
- ✓ Se conectaron dos cables que sirven como puentes para la conexión hacia el condensador, este condensador se encarga de almacenar y estabilizar la energía.

- ✓ Luego se realizó una conexión desde el condensador hasta el sensor INA219 que se encarga de medir el voltaje y corriente para obtener la potencia en vatios generados.
 - ✓ La medición se la realizó para un diodo led blanco que se ilumina cuando se presiona las baldosas produciendo energía eléctrica a partir del fenómeno de la piezoelectricidad.
 - ✓ Para el envío de datos del sensor fue necesario utilizar la placa ESP32, la cual se conectó a una porta pilas de 7.4 V. La placa ESP32 se la programó en Arduino IDE. A continuación, se presenta el código:

```

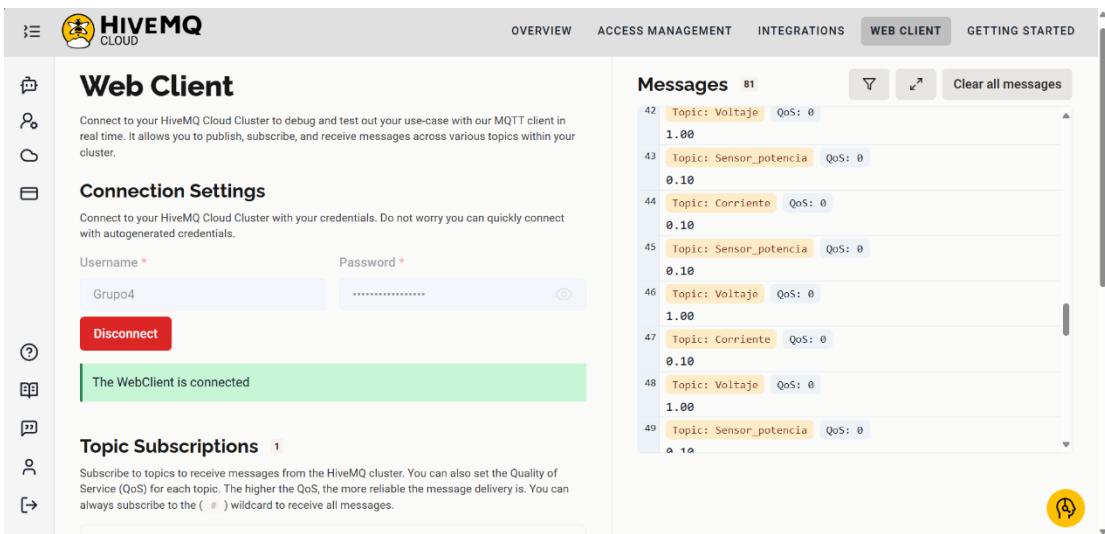
1 #include <default_INA219.h>
2 #include <Wire.h>
3 #define SDA 4
4 #define SCL 22
5 #include <DF1.h>
6 #include <PubSubClient.h>
7 #include <WiFiClientSecure.h> // Para conexiones seguras (MQTT con TLS)
8 // DEFINICIONES DE VARIABLES
9 float shuntvoltage = 0;
10 float busvoltage = 0;
11 float current_mA = 0;
12 float loadvoltage = 0; //
13 float power_mW = 0;
14 float potencia=0; // POTENCIA EN VATIOS
15 float energia=0; // POTENCIA EN VATIOS HORA
16 uint8_t address;
17 uint8_t error;
18
19 // Configuración WiFi
20 const char* ssid = "Redmi10";
21 const char* password = "4snaqgtdczzb83c";
22
23 // Configuración MQTT
24 const char* mqtt_server = "f0e40bb5a35e4e56a00edc9235d53ca7.s1.eu.hivemq.cloud"; // replace with your broker url
25 const char* mqtt_username = "Grupe4";
26 const char* mqtt_password = "Piezoelectric321";
27 const int mqtt_port = 8883;
28 //TOPICOS
29 const char* sensor1_topic="sensor_potencia";
30 const char* command1_topic="POTENCIA";
31 const char* sensor_voltage="Volts";
32 const char* command2_topic="VOLTAJE";
33 const char* sensor_current="Corriente";
34 const char* command3_topic="CORRIENTE";
35 //Conexion del wifi
36 WiFiClientSecure espClient;
37 PubSubClient client(espClient);
38
39 unsigned long lastMsg = 0;
40 #define MSG_BUFFER_SIZE (50)
41 char msg[MSG_BUFFER_SIZE];

```

```
BALDOZA_PIEZOELECTRICA.ino
...
84     ina219.begin();
85     // conexión a la red
86     WiFi.mode(WIFI_STA);
87     WiFi.begin(ssid, password);
88
89     while (WiFi.status() != WL_CONNECTED) {
90         delay(500);
91         Serial.print(".");
92     }
93
94     randomSeed(micros());
95     Serial.println("WiFi conectado\nDirección IP: ");
96     Serial.println(WiFi.localIP());
97
98     while (!Serial) delay(1);
99     // ENVIO DEL COMENZADOO DE CONEXION
100    espClient.setCallback(callback);
101    client.setServer(mqtt_server, mqtt_port);
102    client.setCallback(callback);
103
104 }
105
106 void loop() {
107     // Intentar la conexión
108     if (!client.connected()) reconnect();
109     client.loop();
110
111     //CALIBRACION DEL MODULO DEL SENSOR
112     ina219.setCalibration_10V_400mA();
113     // VARIABLES A OBTENER LAS MEDICIONES
114     shuntvoltage = ina219.getShuntVoltage_mV();
115     busvoltage = ina219.getBusVoltage_V();
116     current_mA = -0.996*ina219.getCurrent_mA();
117     power_mW = ina219.getPower_mW();
118     loadvoltage = busvoltage + (shuntvoltage / 1000);
119     Serial.println(loadvoltage,2);
120
121     Serial.println("V");
122
123     Serial.println(current_mA,2);
124 }
```

```
BALDOZA_PIEZOELECTRICA.ino
...
126     Serial.println("mW");
127     potencia=loadvoltage*current_mA;
128
129     Serial.println(potencia/1000,3);
130
131     Serial.println(" mW");
132     energia = energia + (potencia / 3600);
133
134     Serial.println(energia / 1000);
135
136     Serial.println("\n");
137     // publicación del mensajes en los Topicos
138     publishMessage(sensor1_topic, String(potencia),true);
139     publishMessage(sensor_voltaje, String(loadvoltage),true);
140     publishMessage(sensor_corriente, String(current_mA),true);
141
142
143     delay(1000); // Refresca cada segundo
144 }
145 // -----
146 //-----Function-----
147
148 void reconnect() {
149     // Loop until we're reconnected
150     while (!client.connected()) {
151         // Print error message while connecting...
152         String clientId = "ESP8266-32-"; // Create a random client ID
153         clientId += String(random(0xffff), HEX);
154         // Attempt to connect
155         if (client.connect(clientId.c_str(), mqtt_username, mqtt_password)) {
156             Serial.println("connected");
157
158             client.subscribe(command1_topic); // subscribe the topics here
159             client.subscribe(command2_topic); // subscribe the topics here
160             client.subscribe(command3_topic); // subscribe the topics here
161
162         } else {
163             Serial.print("failed, rc=");
164             Serial.print(client.state());
165             Serial.println(" try again in 5 seconds");
166             delay(5000);
167         }
168     }
169 }
170
171 //-----
172 // This void is called every time we have a message from the broker
173
174 void callback(char* topic, byte* payload, unsigned int length) {
175     String incomingMessage = "";
176     for (int i = 0; i < length; i++) incomingMessage+= (char)payload[i];
177     Serial.println("Mensaje ya llegó ["+String(topic)+" "+incomingMessage];
178     // check for other commands
179     /* else if ( strcmp(topic,command2_topic) == 0){
180     | if (incomingMessage.equals("1")) { } // do something else
181     |
182     }*/
183
184
185 //-----
186 void publishMessage(const char* topic, String payload , boolean retained){
187     if (client.publish(topic, payload.c_str(), true))
188         Serial.println("Message published ["+String(topic)+": "+payload);
189 }
```

- ✓ La integración con la tecnología IoT permite hacer un análisis de los datos recolectados que servirán para mejorar el modelo de baldosa y su aplicación en entornos reales como la Universidad Central. A continuación, se muestra una imagen sobre el broker Hive MQ utilizado para el registro de datos:



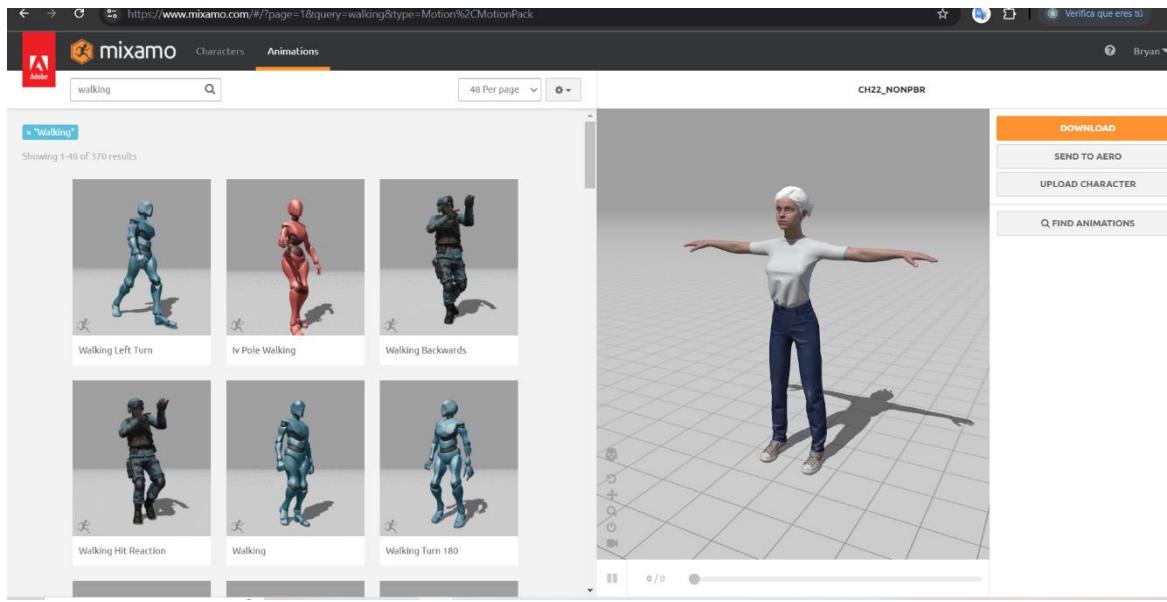
Gemelo digital

El modelo digital desarrollado para este proyecto es una representación virtual de la entrada principal de la UCE, el cual simula el sistema de generación de energía piezoeléctrica a través del paso de personas. Este modelo tiene como propósito ilustrar cómo la energía generada por las pisadas en diferentes áreas de la universidad puede ser utilizada para alimentar pequeños dispositivos, como luminarias, al mismo tiempo que contribuye a la concienciación sobre las energías renovables y la sostenibilidad.

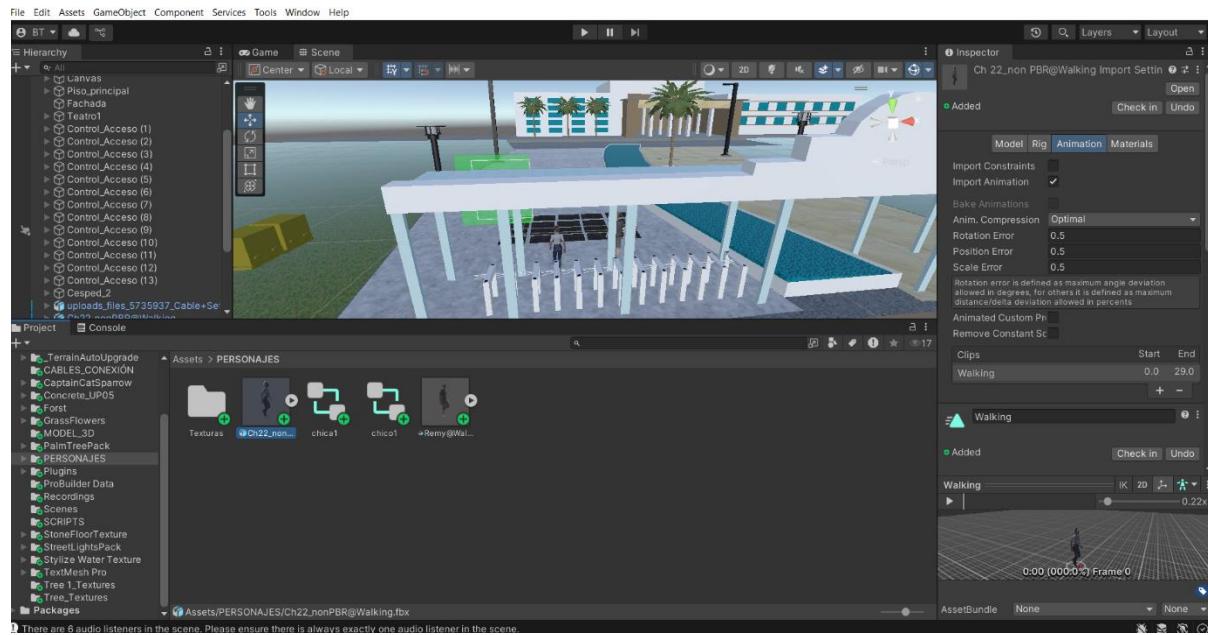
La creación del gemelo digital se realizó utilizando Unity, una plataforma útil para simulaciones en 3D. Unity nos permitió generar un entorno interactivo y visualmente representativo de los espacios de la universidad. En este modelo, se simula el tránsito de personas sobre las baldosas piezoeléctricas, activando el sistema de generación de energía. Además, el gemelo digital refleja en tiempo real la energía generada en función del número de personas caminando y la cantidad de pasos dados, proporcionando una visualización clara del funcionamiento del sistema.

Animación:

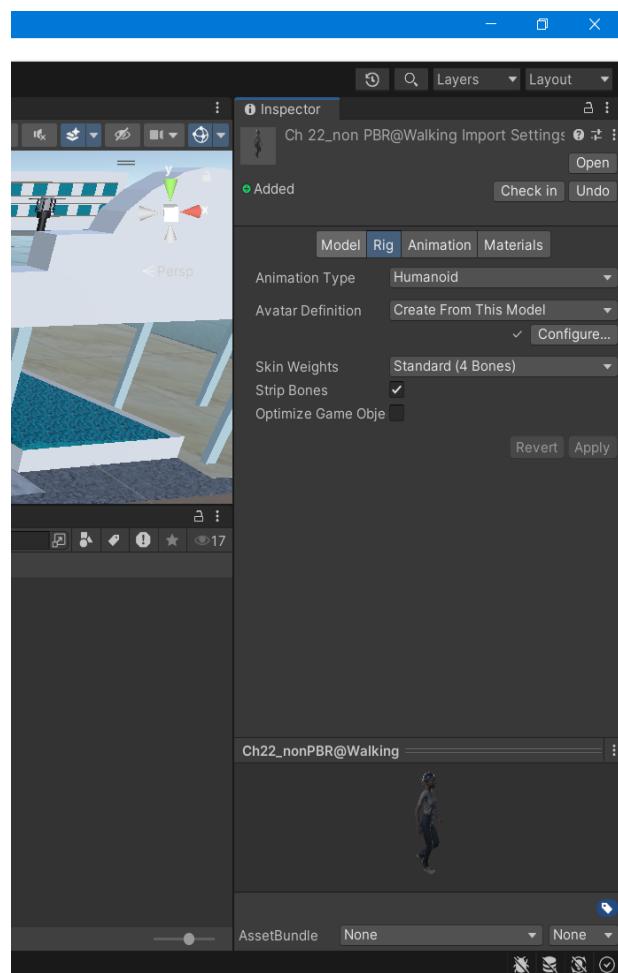
Para la animación de las personas usamos los paquetes de recursos (servicios) dreamtech y mixamo.com



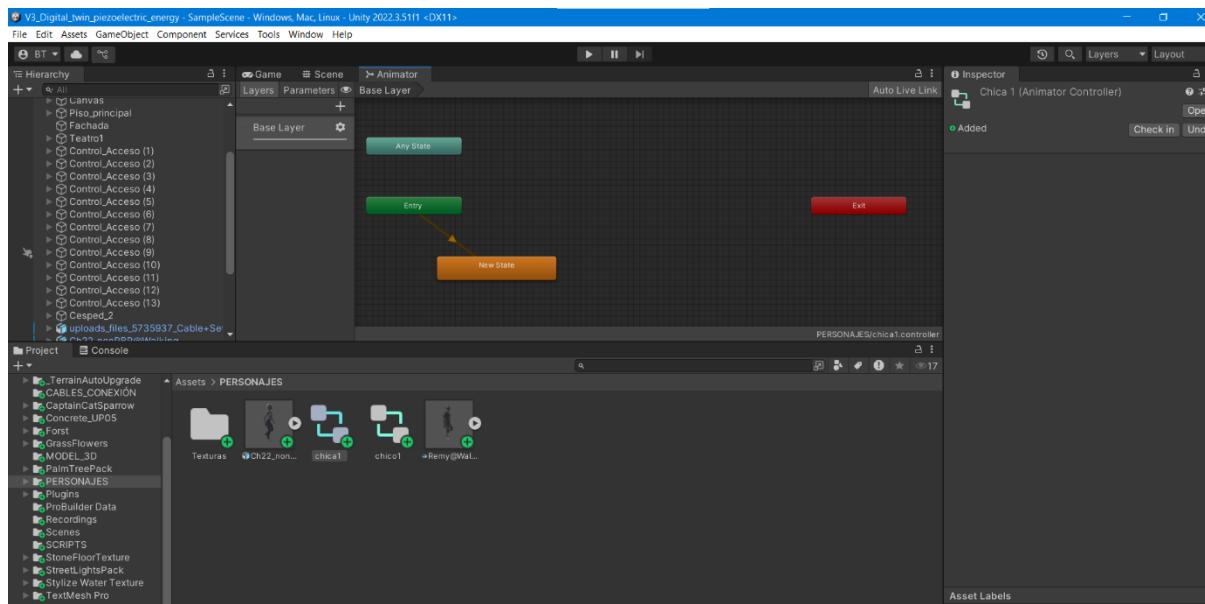
Una vez que descargamos el modelo de nuestra preferencia lo importamos en nuestro proyecto. Aquí podemos editar las texturas, posición, etc.



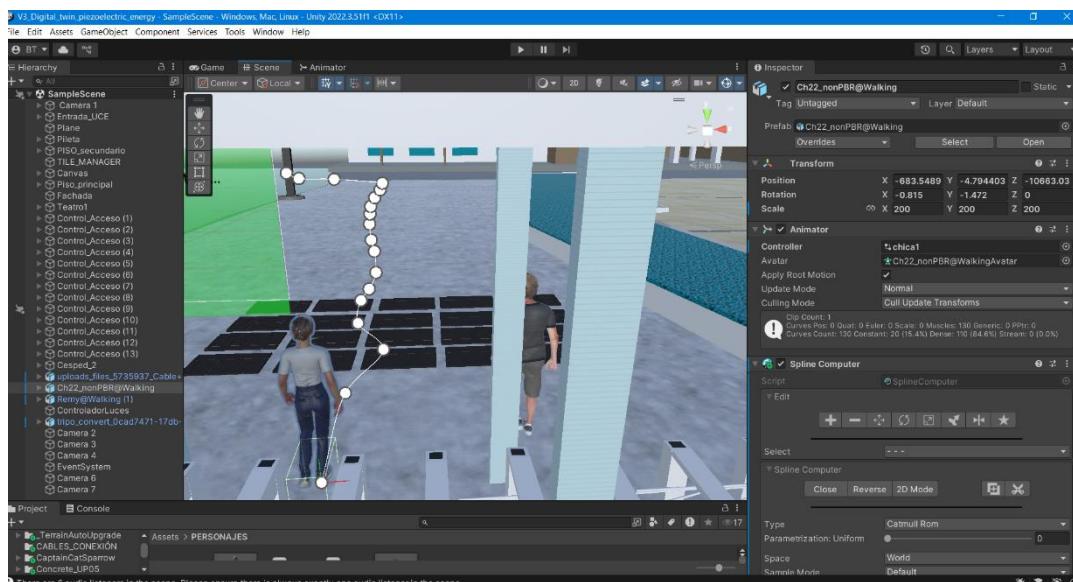
El siguiente paso es acceder a la opción “Rig”, configurar el tipo de animación como “Humanoid”. Luego en “Animation” habilitamos la opción “Loop Time”.

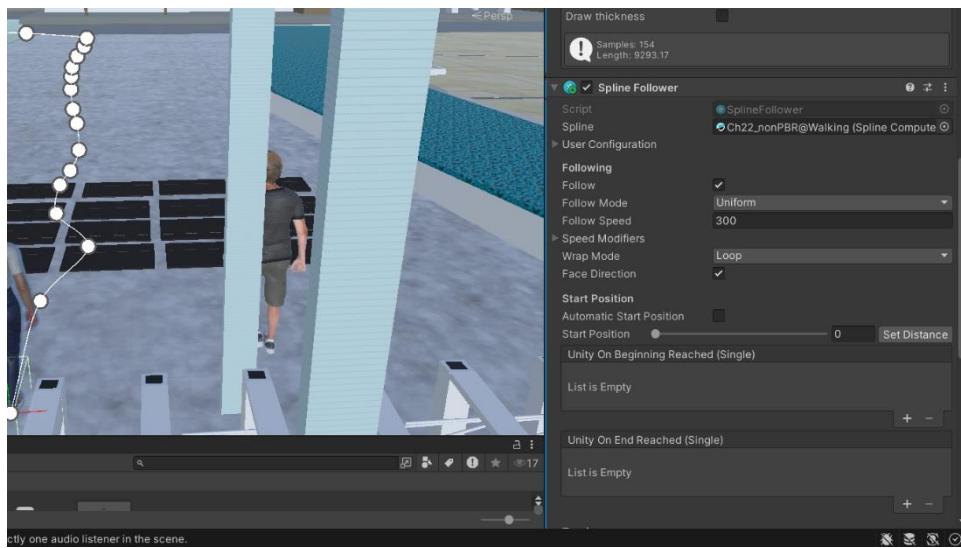


Luego creamos una animación en “create animation”, “Animation controller” y asignamos y configuramos la animación de nuestro modelo.



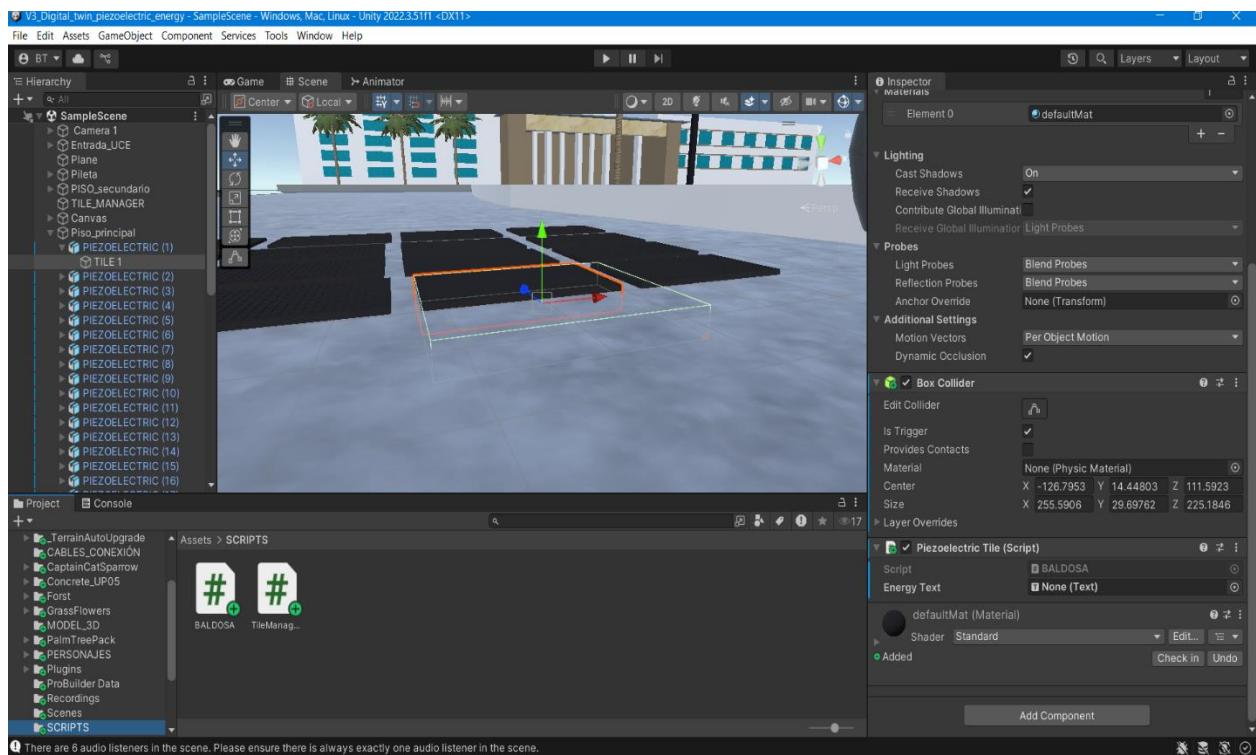
En “Add Component” si elegimos “Spline Computer” podemos trazar la ruta de nuestro modelo de forma visual. “Spline Follower” nos permite editar la velocidad del modelo, posición inicial y su tipo de movimiento. Esto se realizó gracias a la librería DreamTeck, el cual es una herramienta de productividad que ayuda a crear y manipular geometría y trayectorias de manera eficiente, comúnmente usada en juegos que requieren movimientos complejos, como juegos de carreras o simuladores de vuelo [13].





Baldosa:

El script en Unity simula el funcionamiento de una baldosa piezoeléctrica, generando energía a partir de las pisadas de las personas. Cada vez que un personaje entra en contacto con la baldosa, se genera un voltaje aleatorio entre 0.99V y 1V por cada disco piezoeléctrico, y la potencia generada se calcula según la fórmula de potencia eléctrica $P=V\times I$, donde V es el voltaje e (I) la corriente. En este caso, la corriente se calcula dividiendo la potencia por el voltaje total generado por los discos piezoeléctricos dispuestos en serie. La potencia generada en cada paso es mostrada en tiempo real en la interfaz de usuario mediante un texto que se actualiza constantemente. Además, cuando no hay colisiones, el voltaje, corriente y potencia disminuyen gradualmente a través de un factor de decaimiento para simular la pérdida de energía en ausencia de actividad, asegurando que los valores se mantengan realistas. Este comportamiento es monitoreado y mostrado en el UI, proporcionando datos actualizados sobre el voltaje, la corriente y la potencia generada por cada paso.



El script en Unity gestiona varias baldosas piezoeléctricas, calculando la potencia total, corriente y voltaje generados por todas las baldosas activadas. En cada frame, se recorre la lista de baldosas y se suman los valores de potencia, corriente y voltaje de cada una utilizando los métodos GetCurrentPower(), GetCurrentCurrent() y GetCurrentVoltage(), respectivamente. Estos valores se acumulan para obtener el total generado por todas las baldosas activadas. Finalmente, los resultados se muestran en tiempo real en la interfaz de usuario a través de un Text que actualiza la potencia, corriente y voltaje totales generados, proporcionando un resumen claro de la energía generada por el sistema de baldosas piezoeléctricas. Este script permite monitorear de manera eficiente el rendimiento total del sistema piezoeléctrico en tiempo real, facilitando la visualización de los datos generados por cada paso y el comportamiento general del sistema.

Código BALDOSA:

```

using UnityEngine;
using UnityEngine.UI; // Para actualizar la UI

public class PiezoelectricTile : MonoBehaviour
{
    // Datos de energía por paso
    private const float powerPerStep = 2f; // Potencia instantánea generada por cada paso (en vatios)
    private const int numberOFdiscs = 6; // Número de discos piezoelectríficos en serie

    // Variables para el voltaje, corriente y potencia actual
    private float currentVoltage = 0f;
    private float currentCurrent = 0f;
    private float currentPower = 0f;

    // Factor de decaimiento para reducir el voltaje, corriente y potencia cuando no hay colisiones
    private const float decayFactor = 0.99f; // Reduce el voltaje, corriente y potencia en un 1% por frame
    private const float minVoltage = 0.01f; // Valor mínimo para considerar que no hay generación

    // Referencia al texto de UI donde se mostrará la energía generada
    public Text energyText;

    void Update()
    {
        // Si no hay colisiones, reducir gradualmente el voltaje, corriente y potencia
        if (currentVoltage > minVoltage)
        {
            currentVoltage *= decayFactor;
            currentCurrent *= decayFactor;
            currentPower *= decayFactor;
        }
        else
        {
            // Si el voltaje es muy bajo, considerarlo como 0
            currentVoltage = 0f;
            currentCurrent = 0f;
            currentPower = 0f;
        }

        // Actualizar la UI en cada frame
        DisplayEnergy();
    }

    void OnTriggerEnter(Collider other)
    {
        // Comprobar si el objeto que entra en contacto con la baldosa es un personaje
        if (other.CompareTag("Person"))
        {
            // Generar un voltaje aleatorio entre 0.99V y 1V por cada paso
            float voltagePerDisc = Random.Range(0.99f, 1.0f); // Voltaje aleatorio entre 0.99 y 1
            currentVoltage = voltagePerDisc * numberOFdiscs; // Voltaje total en serie

            // Calcular la corriente (P = V * I, entonces I = P / V)
            currentCurrent = powerPerStep / currentVoltage;
        }
    }

    void DisplayEnergy()
    {
        // Muestra los datos de voltaje, corrientes y potencia en el UI
        energyText.text = "Voltaje generado: " + currentVoltage.ToString("F2") + " \n" +
                          "Corriente generada: " + currentCurrent.ToString("F3") + " \n" +
                          "Potencia generada: " + currentPower.ToString("F2") + " W";
    }
}

```

```

using UnityEngine;
using UnityEngine.UI; // Para actualizar la UI

public class PiezoelectricTile : MonoBehaviour
{
    // Datos de energía por paso
    private const float powerPerStep = 2f;
    private const int numberOFdiscs = 6;
    private float currentVoltage = 0f;
    private float currentCurrent = 0f;
    private float currentPower = 0f;

    void Update()
    {
        // Actualizar la UI en cada frame
        DisplayEnergy();
    }

    void OnTriggerEnter(Collider other)
    {
        // Comprobar si el objeto que entra en contacto con la baldosa es un personaje
        if (other.CompareTag("Person"))
        {
            // Generar un voltaje aleatorio entre 0.99V y 1V por cada paso
            float voltagePerDisc = Random.Range(0.99f, 1.0f); // Voltaje aleatorio entre 0.99 y 1
            currentVoltage = voltagePerDisc * numberOFdiscs; // Voltaje total en serie

            // Calcular la corriente (P = V * I, entonces I = P / V)
            currentCurrent = powerPerStep / currentVoltage;

            // Calcular la potencia (P = V * I)
            currentPower = currentVoltage * currentCurrent;
        }
    }

    void DisplayEnergy()
    {
        // Muestra los datos de voltaje, corrientes y potencia en el UI
        energyText.text = "Voltaje generado: " + currentVoltage.ToString("F2") + " \n" +
                          "Corriente generada: " + currentCurrent.ToString("F3") + " \n" +
                          "Potencia generada: " + currentPower.ToString("F2") + " W";
    }

    // Método para obtener la potencia generada por esta baldosa
    public float GetCurrentPower()
    {
        return currentPower;
    }

    public float GetCurrentCurrent()
    {
        return currentCurrent;
    }

    public float GetCurrentVoltage()
    {
        return currentVoltage;
    }
}

```

Script para el manejo de varias baldosas:

The screenshot shows the Unity Editor interface with the code editor open. The script file is named 'TileManager.cs' and is located in the 'BALDOSA.cs' folder. The code implements a MonoBehaviour that tracks a list of piezoelectric tiles and calculates their total power, current, and voltage. It also updates a UI text component with these values.

```

1  using UnityEngine;
2  using UnityEngine.UI;
3  using System.Collections.Generic;
4
5  public class TileManager : MonoBehaviour
6  {
7      // Lista para almacenar todos las baldosas piezoelectricas
8      public List<PiezoelectricTile> tiles;
9
10     // Texto UI para mostrar la potencia, corriente y voltaje totales generados
11     public Text totalEnergyText;
12
13     void Update()
14     {
15         float totalPower = 0f;
16         float totalCurrent = 0f;
17         float totalVoltage = 0f;
18
19         // Sumar la potencia, corriente y voltaje generados por cada baldosa
20         foreach (PiezoelectricTile tile in tiles)
21         {
22             totalPower += tile.GetCurrentPower();
23             totalCurrent += tile.GetCurrentCurrent(); // Sumar la corriente generada por cada baldosa
24             totalVoltage += tile.GetCurrentVoltage(); // Sumar el voltaje generado por cada baldosa
25         }
26
27         // Mostrar la potencia, corriente y voltaje totales en la UI
28         totalEnergyText.text = "Potencia total generada: " + totalPower.ToString("F2") + "\n" +
29                             "Corriente total generada: " + totalCurrent.ToString("F3") + "\n" +
30                             "Voltaje total generado: " + totalVoltage.ToString("F2") + " V";
31     }
32 }

```

Este modelo simula baldosas piezoelectricas que generan energía cuando son pisadas.

Los scripts que controlan este comportamiento se encargan de calcular la energía generada (voltaje, corriente y potencia) por cada baldosa y mostrar estos valores en la interfaz de usuario (UI).

1. Script: PiezoelectricTile (Baldosa Piezoelectrífica)

Este script controla el comportamiento de una baldosa piezoelectrica individual cuando un personaje la pisa. Se encarga de generar los valores de energía y actualizarlos constantemente.

Componentes principales:

- **Variables de energía:**
- **powerPerStep:** Define cuánta energía (en vatios) se genera cada vez que alguien pisa la baldosa. En este caso, se generan 2 vatios por paso.

- **Variables de estado actual:**

currentVoltage, currentCurrent, currentPower: Estas variables almacenan el voltaje, corriente y potencia generados en cada momento.

- **Comportamiento en el tiempo (Update):**

Cuando no hay interacción (nadie pisa la baldosa), la energía generada disminuye gradualmente hasta llegar a cero. Esto se controla mediante un factor de decaimiento (decayFactor), que reduce los valores de energía en un 1% por cada frame.

- **Interacción con el personaje (OnTriggerEnter):**

Cuando un objeto con la etiqueta "Person" entra en contacto con la baldosa, se genera un voltaje aleatorio entre 0.99V y 1V por cada disco piezoeléctrico. El voltaje total se calcula multiplicando este valor por el número de discos.

Luego, se calcula la corriente generada utilizando la fórmula de potencia:

Potencia=Voltaje×Corriente. A partir de esta fórmula, la corriente se calcula como $I=P/V$

- **Actualización de la interfaz (DisplayEnergy):**

Los valores de voltaje, corriente y potencia generados se muestran en tiempo real en la interfaz de usuario mediante el texto energyText.

Update: Reduce gradualmente los valores de energía cuando la baldosa no es pisada.

OnTriggerEnter: Genera energía cuando un personaje pisa la baldosa.

DisplayEnergy: Muestra los valores generados en la UI.

Métodos de obtención: GetCurrentPower(), GetCurrentCurrent(), GetCurrentVoltage() permiten acceder a los valores actuales de potencia, corriente y voltaje.

2. Script: TileManager (Gestor de Baldosas)

Este script se encarga de gestionar varias baldosas piezoeléctricas y sumar la energía generada por todas ellas. La energía total también se muestra en la interfaz de usuario.

Componentes principales:

- **Lista de baldosas (tiles):**

Este script mantiene una lista de todas las baldosas piezoeléctricas en la escena. Cada baldosa está controlada por su propio script (PiezoelectricTile).

- **Comportamiento en el tiempo (Update):**

Cada frame, el script suma la potencia, corriente y voltaje generados por todas las baldosas en la lista.

Los valores totales se muestran en un texto de la UI (totalEnergyText).

Update: Recorre todas las baldosas y suma los valores de energía generados (potencia, corriente y voltaje), actualizando la UI en tiempo real.

Manual del modelo físico y digital

La baldosa piezoeléctrica es un dispositivo capaz de generar energía eléctrica a partir de la presión mecánica aplicada sobre su superficie. Este manual proporciona una guía

detallada para la construcción de un prototipo, incluyendo materiales, herramientas, procedimiento, conexión a la base de datos y demás recomendaciones de seguridad.

2.1 Objetivo General

Construir un prototipo funcional de baldosa piezoeléctrica capaz de generar energía a partir de la presión ejercida por una persona.

2.2 Objetivos Específicos

- Seleccionar y adquirir los materiales adecuados para la fabricación.
- Ensamblar los componentes piezoeléctricos en una estructura funcional.
- Evaluar el desempeño de la baldosa en términos de generación eléctrica.
- Establecer una conexión con una base de datos.
- Documentar el proceso para futuras mejoras y replicación del diseño.

3. MATERIALES

- Cartón prensado A4 (1)
- Madera plana (1)
- Espuma flex de 20 cm x 30 cm (1)
- 35 mm discos piezoeléctricos (6)
- Arduino esp32 (1)
- Mini Proto board (1)
- Cables de conexión (1m)
- Cables puente (paquete) macho a macho y hembra
- Capacitor de 10uf (1)
- Resistencia de 100Kohm (1)
- Resistencia de 10K ohm (1)
- Diodos 1N4007 rectificador (5)
- Transistor BC547 PNP (1)

- Barras de silicona (2)
- Cinta led (1)
- Led rojo (1)
- Portapilas 18650 2x 3.7v (1)
- Pilas 18650 3.7V (2)
- Modulo INA219I2C (1 sensor)
- Cinta Aislante

4. HERRAMIENTAS

- Soldador y estaño
- Cautín
- Multímetro digital
- Pinzas y tijeras
- Regla y marcador
- Tapas huecas de botellas (6)

5. PROCEDIMIENTO

5.1 Preparación de la Base

1. Cortar la base de cartón con las dimensiones deseadas.
2. Perforar orificios si es necesario para el paso del cableado.

5.2 Instalación de los Discos Piezoeléctricos

1. Ubicar los discos piezoeléctricos en una matriz uniforme.
2. Fijar los discos sobre tapas huecas de botellas utilizando silicón.

3. Conectar los discos en un circuito mixto (serie y paralelo) según el diseño del circuito.
(Revisar Proteus)
4. Soldar los cables a los terminales de cada disco. (Utilizar soldador y estaño)

5.3 Conexión protoboard

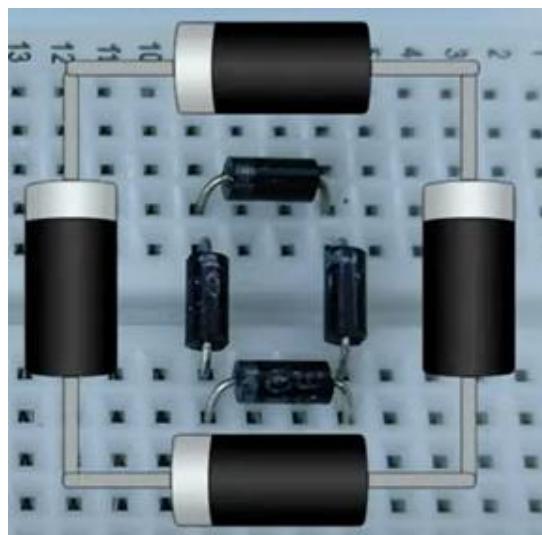
DIODOS 1N4007

Insertar en los puntos **e6** y **f6**

Insertar en los puntos **g6** y **g9**

Insertar en los puntos **d6** y **d9**

Insertar en los puntos **e9** y **f9**



CABLE TIMBRE

Insertar en los puntos **j6** y **j14**

Insertar en los puntos **b9** y **b14**

Insertar en los puntos **e23** y **f23**

Insertar en los puntos **i14** y **i23**

Insertar en los puntos **j23** y punto de línea negativo

CAPACITOR

Insertar en los puntos **g14** (negativo) y **d14** (positivo)

RESISTENCIAS

100k ohm insertar en los puntos **a14 y a19**

10k ohm insertar en los puntos **b19 y b23**

10k ohm insertar en los puntos **e24 y f24**

TRANSISTOR

COLLECTOR A23

BASE A24

EMISOR A25

DISCOS PIEZOELECTRICOS

NEGATIVO: EN **H9**

POSITIVO: EN **B6**

5.4 Conexión ESP32

CABLE MACHO A MACHO

Insertar en los puntos **c19 y gpio36 o gpio39 (esp32)**

Insertar en los puntos **g24 (base transistor) y pin gpio 13,14,16 o 17 cualquiera sirve (esp32)**

Insertar en el punto **negativo** – de protoboard y **gnd (esp32)**

Insertar en el punto **positivo** + de protoboard y **vin (esp32)**

5.5 Conexión INA219 I2C

Cable hembra a macho

Insertar en **gnd (display) y punto negativo (protoboard) (este a su vez está conectado al gnd del esp32 por un cable macho a macho)**

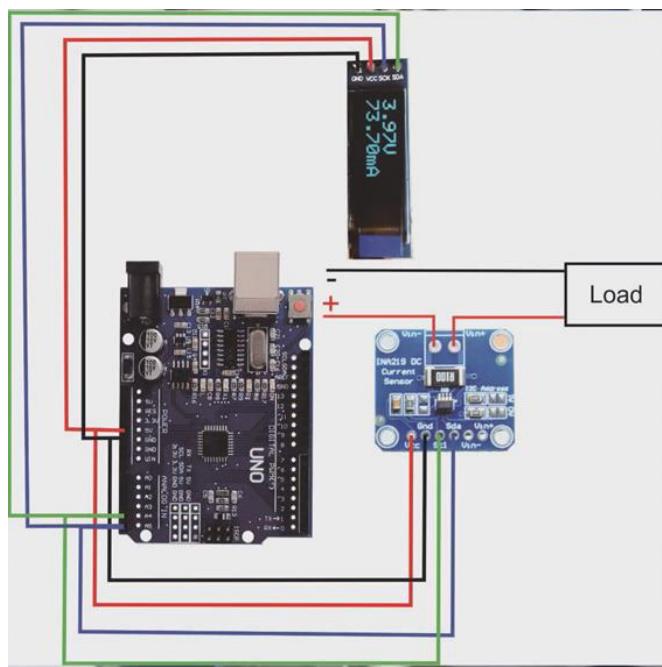
Insertar en **vcc (display) y pin 5v (esp32)**

Insertar en **sda (display) y pin gpio 21 (sda) (esp32 para datos)**

Insertar en **scl (display) y pin gpio 22 (scl) (esp32 para el reloj i2c)**

Insertar en **VIN + y la carga (LED rojo) a conectar (en serie)**

Insertar en vin - y la fuente de voltaje que va hacia la carga. (En serie)



5.6 Almacenamiento

1. Acoplar la batería recargable para almacenar la energía para el arduino.

5.7 Ensamblaje Final

1. Colocar una capa de protección sobre los discos.
2. Instalar la cubierta superior de acrílico o madera.
3. Asegurar la estructura con adhesivo aislante.

6. Pruebas y validación modelo físico

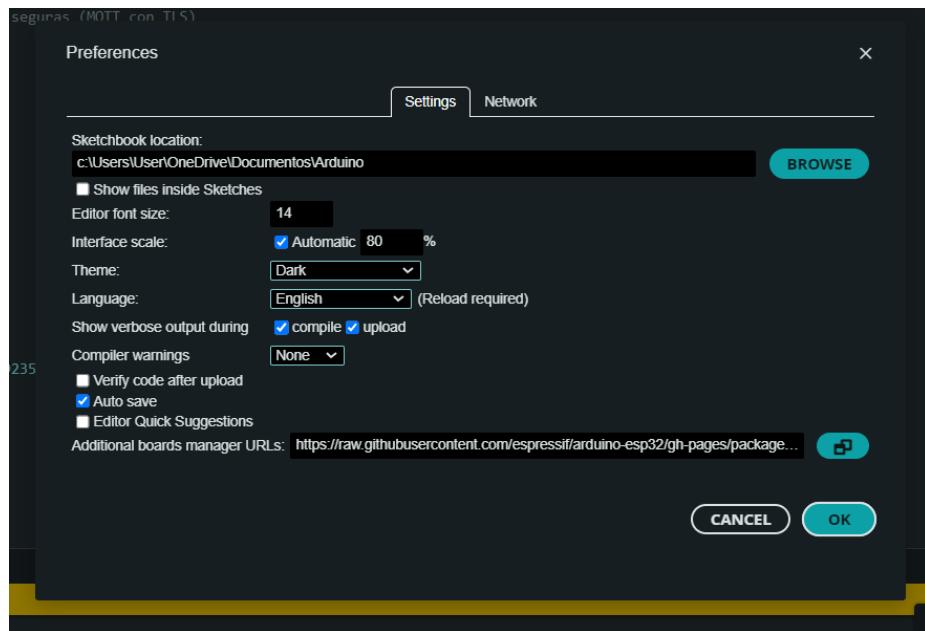
1. Utilizar un multímetro para medir el voltaje y corriente generados.
2. Aplicar presión manualmente y registrar los valores obtenidos.
3. Evaluar la eficiencia de almacenamiento en la batería.
4. Realizar ajustes si es necesario para optimizar el rendimiento.

7. Seguridad modelo físico

- Evitar cortocircuitos al soldar componentes.
- Aislard adecuadamente los cables para prevenir descargas.
- Usar protecciones adecuadas como guantes y gafas de seguridad.

8. Programación de Arduino

1. Configurar el entorno de desarrollo de Arduino IDE.
2. Agregar la url: https://raw.githubusercontent.com/espressif/arduino-esp32/gh-pages/package_esp32_index.json en la sección de preferencias para configurar la placa esp32 en el edito Arduino IDE.



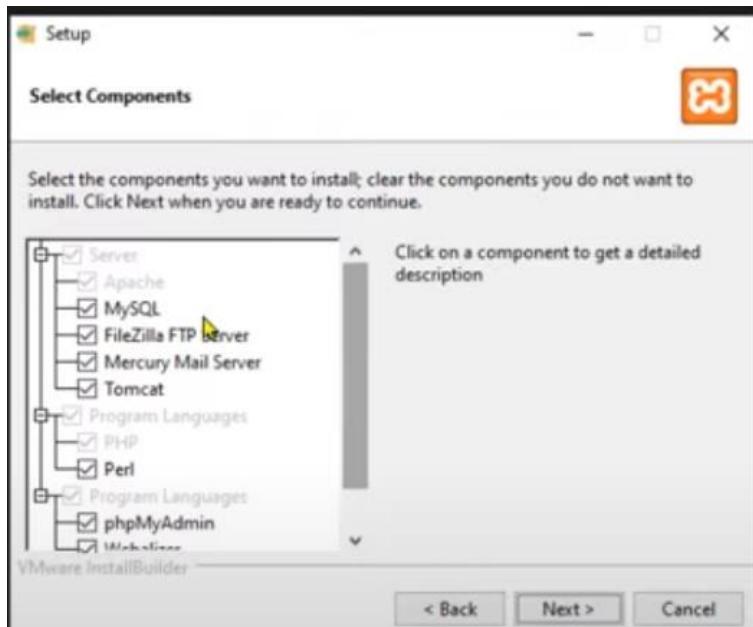
3. Instalar bibliotecas necesarias: Adafruit INA219 , WiFi, Wire.h, PubSubClient.h y WiFiClientSecure.h

```
#include <Adafruit_INA219.h>
#include <Wire.h>
#define SDA 21
#define SCL 22
#include <WiFi.h>
#include <PubSubClient.h>
#include <WiFiClientSecure.h> // Para conexiones seguras (MQTT con TLS)
```

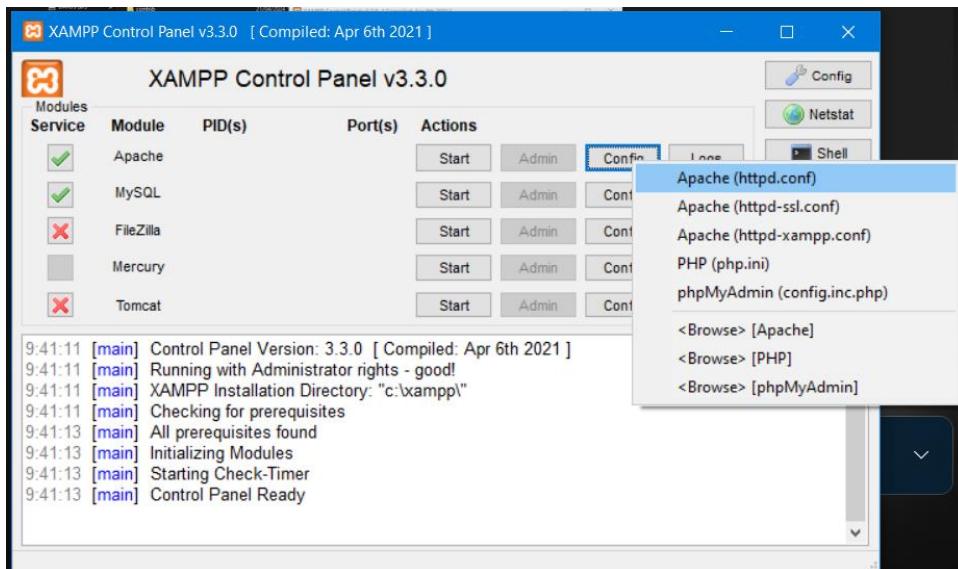
4. Conectar los sensores piezoelectricos a la placa Arduino.
5. Conectar el ESP32 al pc con un cable serial, seleccionando el puerto COM y la placa ESP32.
6. Configurar el broker Hive MQ y crear el usuario y tópicos
7. Escribir un código en C++ para leer los valores de voltaje generados y enviarlos al broker Hive MQ.
8. Implementar un código de transmisión de datos usando WiFi con ESP8266 o ESP32.
9. Probar la transmisión de datos enviando valores a la consola serial.

9. Conexión a Base de Datos

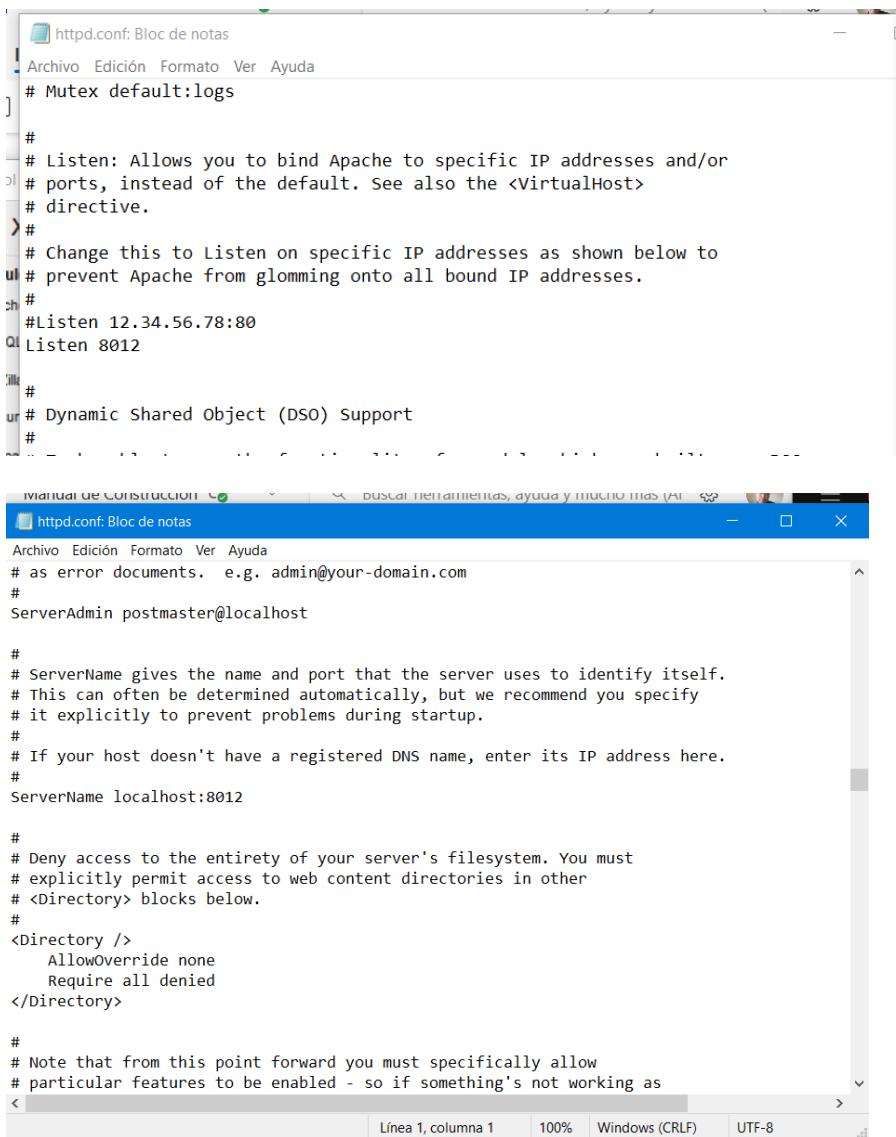
1. Crear y configurar una base de datos por medio de XAMPP, que es una distribución de software libre que proporciona un entorno de servidor local para desarrolladores web. Para configurar XAMPP descargamos el archivo [xampp-windows-x64-8.2.4-0-VS16](#) en nuestro equipo. Durante la descarga, debemos verificar que los componentes este seleccionado phpMyAdmin.



2. Como recomendación, configuraremos XAMPP con el puerto 8012 , debido a que puede existir conflictos con otras aplicaciones si dejamos por defecto con el puerto 80. En Apache > Config > Apache(httpd.conf) :



3. Cambiamos al puerto 8012 en las siguientes partes del archivo.



The image shows two separate windows of the Windows Notepad application. Both windows have the title 'httpd.conf: Bloc de notas'.

Top Window Content:

```

Mutex default:logs
#
# Listen: Allows you to bind Apache to specific IP addresses and/or
# ports, instead of the default. See also the <VirtualHost>
# directive.
#
# Change this to Listen on specific IP addresses as shown below to
# prevent Apache from glomming onto all bound IP addresses.
#
#Listen 12.34.56.78:80
Listen 8012
#
# Dynamic Shared Object (DSO) Support
#

```

Bottom Window Content:

```

# as error documents. e.g. admin@your-domain.com
#
ServerAdmin postmaster@localhost

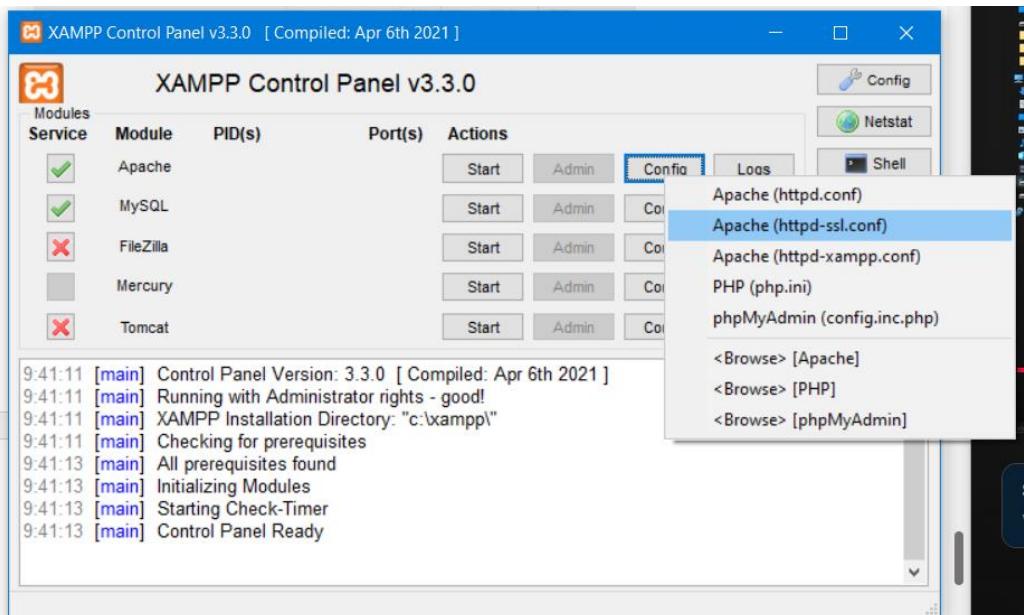
#
# ServerName gives the name and port that the server uses to identify itself.
# This can often be determined automatically, but we recommend you specify
# it explicitly to prevent problems during startup.
#
# If your host doesn't have a registered DNS name, enter its IP address here.
#
ServerName localhost:8012

#
# Deny access to the entirety of your server's filesystem. You must
# explicitly permit access to web content directories in other
# <Directory> blocks below.
#
<Directory />
    AllowOverride none
    Require all denied
</Directory>

#
# Note that from this point forward you must specifically allow
# particular features to be enabled - so if something's not working as
# intended, make sure it is listed here.

```

4. Ahora también configuremos en el archivo Apache (httpd-ssl.conf). Cambiamos en las partes del archivo que se muestran a continuación al puerto 1443:

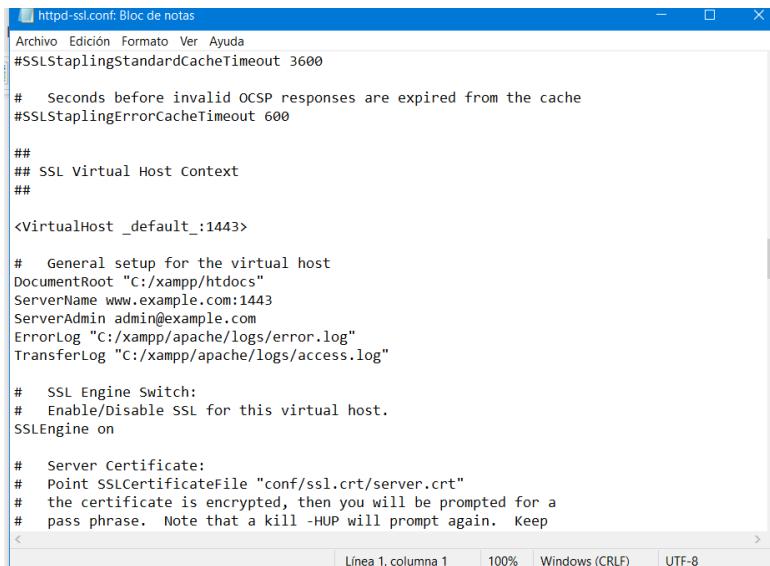


```
httpd-ssl.conf: Bloc de notas
Archivo Edición Formato Ver Ayuda
# platforms additionally provide a /dev/urandom device which doesn't
# block. So, if available, use this one instead. Read the mod_ssl User
# Manual for more details.
#
#SSLRandomSeed startup file:/dev/random 512
#SSLRandomSeed startup file:/dev/urandom 512
#SSLRandomSeed connect file:/dev/random 512
#SSLRandomSeed connect file:/dev/urandom 512

#
# When we also provide SSL we have to listen to the
# standard HTTP port (see above) and to the HTTPS port
#
Listen 1443

##
## SSL Global Context
##
## All SSL configuration in this context applies both to
## the main server and all SSL-enabled virtual hosts.
##

# SSL Cipher Suite:
# List the ciphers that the client is permitted to negotiate,
# and that httpd will negotiate as the client of a proxied server.
```



```

httpd-ssl.conf: Bloc de notas
Archivo Edición Formato Ver Ayuda
#SSLStaplingStandardCacheTimeout 3600
# Seconds before invalid OCSP responses are expired from the cache
#SSLStaplingErrorCacheTimeout 600

## 
## SSL Virtual Host Context
## 

<VirtualHost _default_:1443>

# General setup for the virtual host
DocumentRoot "C:/xampp/htdocs"
ServerName www.example.com:1443
ServerAdmin admin@example.com
ErrorLog "C:/xampp/apache/logs/error.log"
TransferLog "C:/xampp/apache/logs/access.log"

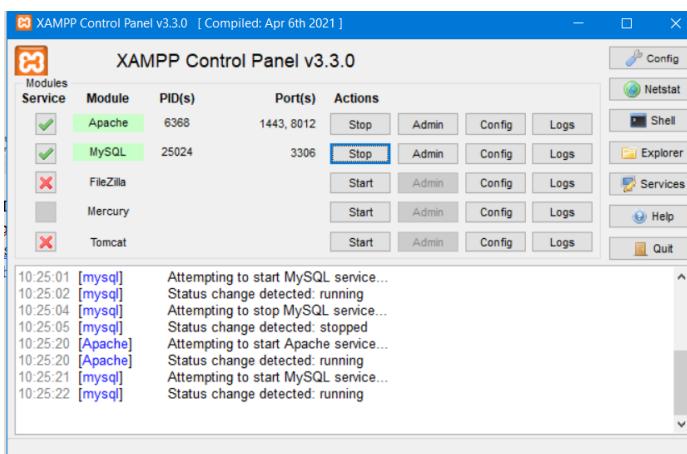
# SSL Engine Switch:
# Enable/Disable SSL for this virtual host.
SSLEngine on

# Server Certificate:
# Point SSLCertificateFile "conf/ssl.crt/server.crt"
# the certificate is encrypted, then you will be prompted for a
# pass phrase. Note that a kill -HUP will prompt again. Keep
<

```

Línea 1, columna 1 | 100% | Windows (CRLF) | UTF-8

5. Ahora vamos a iniciar Apache y MySQL .



6. En nuestro navegador ingresamos “localhost:8012”,y se nos muestra lo siguiente:



Welcome to XAMPP for Windows 8.2.4

You have successfully installed XAMPP on this system! Now you can start using Apache, MariaDB, PHP and other components. You can find more info in the FAQs section or check the HOW-TO Guides for getting started with PHP applications.

XAMPP is meant only for development purposes. It has certain configuration settings that make it easy to develop locally but that are insecure if you want to have your installation accessible to others.

Start the XAMPP Control Panel to check the server status.

Community

XAMPP has been around for more than 10 years – there is a huge community behind it. You can get involved by joining our Forums, liking us on Facebook, or following our exploits on Twitter.



7. Ingresamos a PhpMyAdmin , el cual es un sistema web creado con el lenguaje de programación php , que nos sirve para administrar bases de datos MySQL.

The screenshot shows the phpMyAdmin interface with the following sections visible:

- Configuraciones generales:** Shows the connection configuration for the server (utf8mb4_unicode_ci).
- Configuraciones de apariencia:** Shows language (Español - Spanish) and theme (pnahome) settings.
- Servidor de base de datos:** Displays server details including IP (127.0.0.1), port (TCP/IP), MySQL version (10.4.28-MariaDB), and character set (UTF-8 Unicode).
- Servidor web:** Displays server details including Apache (2.4.56), PHP (8.2.4), and MySQL (8.0.28).
- phpMyAdmin:** Displays version information (5.2.1) and links to documentation, support, and licensing.
- Notificación:** A message at the bottom indicates an older version is available (5.2.2).
- Bottom navigation:** Includes links for Favorites, Options, History, and Clear.

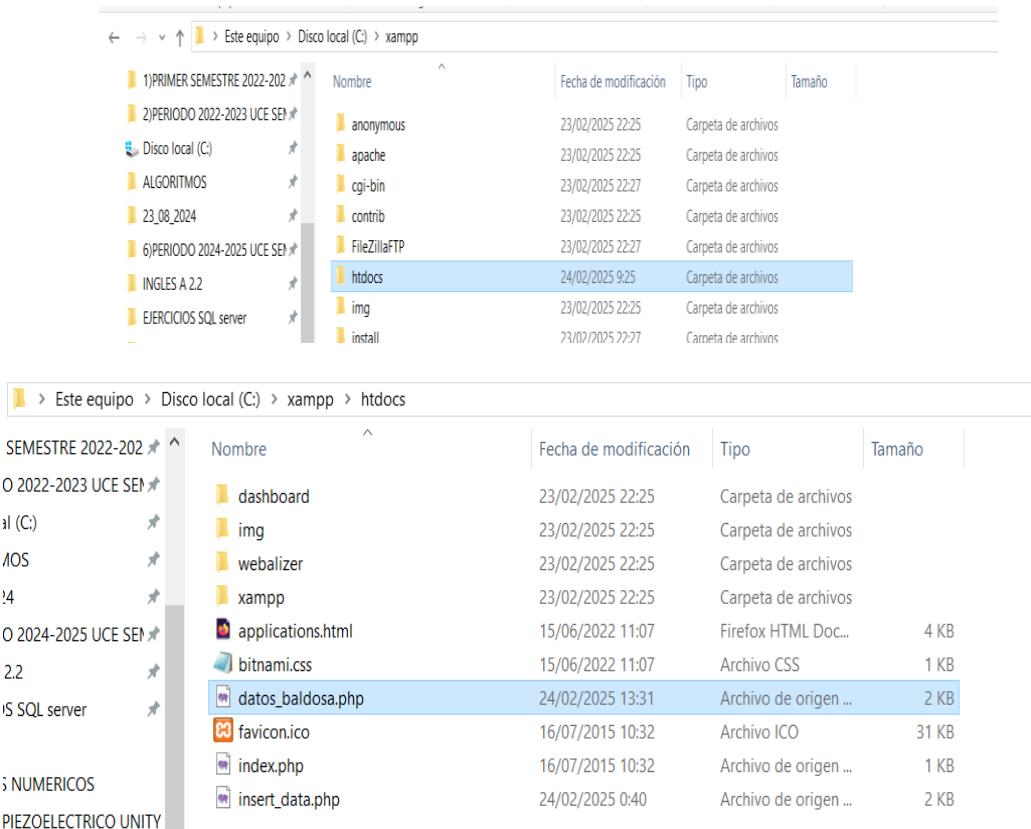
8. Creamos una base de datos con el nombre datos_baldosa.

9. Creamos una tabla para almacenar la información de voltaje , corriente, y potencia.

10. Se nos abre una ventana donde debemos asignar los atributos de la tabla, con sus campos respectivos. En este caso se asignó un id como identificador, (corriente, voltaje, potencia) asignamos con el tipo de dato FLOAT y por último fecha con el tipo de dato TIMESTAMP, que se usa para almacenar una combinación de fecha y hora.

Nombre	Tipo	Longitud/Valores	Predeterminado	Colejamiento	Atributos	Nodo A_J	Comentarios	Virtualidad	Mover columna	Tipo de medio
id	INT	11	Ninguno							
corriente	FLOAT		Ninguno							
voltaje	FLOAT		Ninguno							
potencia	FLOAT		Ninguno							
fecha	TIMESTAMP		CURRENT_TIMESTAMP							

11. EL siguiente paso es ubicarnos en la carpeta `htdocs`. En ella crearemos un archivo `php` para colocar el código que permite enviar a la base de datos los datos del modelo digital.



12. Script `php` para enviar la información a la base de datos desde el modelo digital en Unity.

```
<?php
// Configuración de la base de datos
$servername = "localhost";
$username = "root"; // Usuario por defecto en XAMPP
$password = ""; // Contraseña por defecto en XAMPP (vacía)
$dbname = "datos_baldosa"; // Nombre de la base de datos

// Crear la conexión
$conn = new mysqli($servername, $username, $password, $dbname);
```

```

// Comprobar la conexión
if ($conn->connect_error) {
    die("Conexión fallida: " . $conn->connect_error);
} else {
    echo "Conexión exitosa a la base de datos.<br>";
}

// Verificar si los datos están presentes en $_POST
if (isset($_POST['potencia'], $_POST['corriente'], $_POST['voltaje']))
{
    // Obtener los valores enviados y asegurarse de que sean cadenas de
    // texto (VARCHAR)
    $potencia = (string)$_POST['potencia'];
    $corriente = (string)$_POST['corriente'];
    $voltaje = (string)$_POST['voltaje'];

    // Mostrar los datos recibidos para depuración
    echo "Datos recibidos: Potencia = $potencia, Corriente =
$corriente, Voltaje = $voltaje<br>";

    // Verificar que los valores no sean NULL (permitir valor "0.00")
    if ($potencia !== null && $corriente !== null && $voltaje !== null)
    {
        // Insertar los datos como VARCHAR
        $sql = "INSERT INTO sensor_data (corriente, voltaje, potencia)
                VALUES ('$corriente', '$voltaje', '$potencia')";

        if ($conn->query($sql) === TRUE) {
            echo "Datos insertados correctamente en la base de
            datos.<br>";
        } else {
            echo "Error al insertar datos: " . $sql . "<br>" . $conn-
            >error;
        }
    } else {
        echo "Los datos no están completos.<br>";
    }
} else {
    echo "Faltan datos en la solicitud.<br>";
}

// Cerrar la conexión
$conn->close();
?>

```

Este script PHP:

- Recibe datos de potencia, corriente y voltaje enviados mediante una solicitud POST.
- Conecta a una base de datos MySQL.
- Inserta los datos recibidos en una tabla llamada sensor_data.
- Muestra mensajes de éxito o error en la salida para depuración.

10. Implementación del Gemelo Digital en Unity

1. Configurar Unity e instalar las bibliotecas necesarias.
2. Crear un modelo 3D de la baldosa piezoeléctrica.
3. Implementar un script en C# para conectar Unity a la base de datos y para conectar Unity al bróker Hive MQ.

Script DATOS.cs :

Este código es un script de Unity (TileManagerData) que se encarga de gestionar la comunicación con un broker MQTT para recibir datos de sensores (potencia, voltaje y corriente) y mostrarlos en la interfaz de usuario (UI). Además, envía estos datos a un servidor PHP mediante una solicitud HTTP POST.

```
using UnityEngine;
using UnityEngine.UI;
using MQTTnet; // Namespace principal
using MQTTnet.Client; // Namespace para el cliente MQTT
//using MQTTnet.Client.Options; // Namespace para las opciones del cliente
using System.Text;
using System.Threading.Tasks;
using System;
using UnityEngine.Networking;
using System.Collections;
```

```

using System.Collections.Generic;
using System.Data.SqlTypes;
using System.Globalization;

public class TileManagerData : MonoBehaviour
{
    // Referencias a los elementos de UI
    public Text potenciaText;
    public Text voltajeText;
    public Text corrienteText;

    // Configuraci n MQTT
    private IMqttClient mqttClient;
    public string brokerAddress =
    "f0e40bb5a35e4e56a00edc9235d53ca7.s1.eu.hivemq.cloud";
    public string username = "Grupo4";
    public string password = "Piezoelectric321/";

    public string topicPotencia = "Sensor_potencia";
    public string topicVoltaje = "Voltaje";
    public string topicCorriente = "Corriente";

    // Variables para almacenar los datos recibidos
    private string potenciaMsg = "";
    private string voltajeMsg = "";
    private string corrienteMsg = "";

    //URL base de datos

    private string phpUrl = "http://localhost:8012/datos\_baldosa.php";
    // Cambia esta URL por la ruta de tu script PHP

    async void Start()
    {
        InvokeRepeating("DatosObtenidos", 0.5f, 1f); // Llamar a la
        simulaci n de todos los datos cada 1 segundos
        // Crear el cliente MQTT
        var factory = new MqttFactory();
        mqttClient = factory.CreateMqttClient();

        // Configurar opciones del cliente para HiveMQ
        var options = new MqttClientOptionsBuilder()
            .WithTcpServer(brokerAddress, 8883) // Puerto para MQTT
            sobre TLS/SSL
    }
}

```

```

        //.WithTls() //

Habilita TLS/SSL
    .WithTlsOptions(new MqttClientTlsOptions
    {
        UseTls = true
    })
    .WithCredentials(username, password) // Autenticaci?n
    .WithCleanSession()
    .Build();

    // Conectar al broker HiveMQ
    await ConnectToBroker(options);
}

async Task ConnectToBroker(MqttClientOptions options)
{
    try
    {
        // Conectar al broker
        var connectResult = await mqttClient.ConnectAsync(options);

        if (connectResult.ResultCode ==
MqttClientConnectResultCode.Success)
        {
            Debug.Log($"Conectado a {brokerAddress} en el puerto
8883...");

            // Suscribirse a los t?picos
            await mqttClient.SubscribeAsync(new
MqttTopicFilterBuilder().WithTopic(topicPotencia).Build());
            await mqttClient.SubscribeAsync(new
MqttTopicFilterBuilder().WithTopic(topicVoltaje).Build());
            await mqttClient.SubscribeAsync(new
MqttTopicFilterBuilder().WithTopic(topicCorriente).Build());

            Debug.Log($"Suscrito a los t?picos: {topicPotencia},
{topicVoltaje}, {topicCorriente}");

            // Manejar mensajes recibidos
            mqttClient.ApplicationMessageReceivedAsync += async e=>
            {
                string payload =
Encoding.UTF8.GetString(e.ApplicationMessage.PayloadSegment);
                Debug.Log($"Mensaje recibido en
{e.ApplicationMessage.Topic}: {payload}");
            };
        }
    }
}

```

```

        // Actualizar las variables seg n el t pico
recibido
        if (e.ApplicationMessage.Topic == topicPotencia)
        {
            potenciaMsg = $"Potencia: {payload} W";
        }
        else if (e.ApplicationMessage.Topic ==
topicVoltaje)
        {
            voltajeMsg = $"Voltaje: {payload} V";
        }
        else if (e.ApplicationMessage.Topic ==
topicCorriente)
        {
            corrienteMsg = $"Corriente: {payload} A";
        }
        // Devolver una tarea completada
        await Task.CompletedTask;
    };
}
else
{
    Debug.LogError($"Error al conectar:
{connectResult.ResultCode}");
}
}
catch (Exception ex)
{
    Debug.LogError($"Excepc n al conectar: {ex.Message}");
}
}

void Update()
{
    // Actualizar la UI en el hilo principal
    if (potenciaText != null)
        potenciaText.text = potenciaMsg;

    if (voltajeText != null)
        voltajeText.text = voltajeMsg;

    if (corrienteText != null)
        corrienteText.text = corrienteMsg;
}

async void OnDestroy()
{

```

```

        if (mqttClient != null && mqttClient.IsConnected)
        {
            await mqttClient.DisconnectAsync();
            Debug.Log("Desconectado del broker HiveMQ");
        }
    }

    // Método para obtener la potencia generada por esta baldosa
    public float GetCurrentPower()
    {
        // Extraer el valor numérico del texto y convertirlo a float
        if (float.TryParse(potenciaMsg.Replace("Potencia: ",
        "").Replace(" W", ""), out float potencia))
        {
            return potencia;
        }
        else
        {
            Debug.LogError("No se pudo convertir la potencia a
float.");
            return 0.0f; // Valor por defecto en caso de error
        }
    }

    public float GetCurrentCurrent()
    {
        // Extraer el valor numérico del texto y convertirlo a float
        if (float.TryParse(corrienteMsg.Replace("Corriente: ",
        "").Replace(" A", ""), out float corriente))
        {
            return corriente;
        }
        else
        {
            Debug.LogError("No se pudo convertir la corriente a
float.");
            return 0.0f; // Valor por defecto en caso de error
        }
    }

    public float GetCurrentVoltage()
    {
        // Extraer el valor numérico del texto y convertirlo a float
        if (float.TryParse(voltajeMsg.Replace("Voltaje: ",
        "").Replace(" V", ""), out float voltaje))
        {
            return voltaje;
        }
    }
}

```

```

        }
    else
    {
        Debug.LogError("No se pudo convertir el voltaje a float.");
        return 0.0f; // Valor por defecto en caso de error
    }
}

// parte de PHP

void DatosObtenidos()
{
    float potencia = GetCurrentPower() ;
    float corriente = GetCurrentCurrent() ;
    float voltaje = GetCurrentVoltage() ;

    string potenciaStr =
potencia.ToString(CultureInfo.InvariantCulture);
    string corrienteStr =
corriente.ToString(CultureInfo.InvariantCulture);
    string voltajeStr =
voltaje.ToString(CultureInfo.InvariantCulture);

    StartCoroutine(EnviarDatosAPHP(corrienteStr, voltajeStr,
potenciaStr));
}

// Enviar los datos a PHP mediante una solicitud POST
private IEnumerator EnviarDatosAPHP(string corriente, string
voltaje, string potencia)
{
    // Crear un diccionario para los parámetros a enviar
    var postData = new Dictionary<string, string>
    {

        { "corriente", corriente },
        { "voltaje", voltaje },
        { "potencia", potencia },
    };

    // Crear la solicitud POST con UnityWebRequest
    using (UnityWebRequest www = UnityWebRequest.Post(phiUrl,
postData))
    {
        yield return www.SendWebRequest();
    }
}

```

```
        if (www.result == UnityWebRequest.Result.Success)
        {
            Debug.Log($"Datos enviados correctamente: Corriente = {corriente}, Voltaje = {voltaje}, Potencia = {potencia}");
            Debug.Log($"Respuesta del servidor: {www.downloadHandler.text}");
        }
        else
        {
            Debug.LogError($"Error al enviar datos: {www.error}");
        }
    }
}
```

El Script BALDOSA.cs:

- Detecta cuando un personaje pisa una baldosa piezoeléctrica.
 - Obtiene los datos de voltaje, corriente y potencia desde TileManagerData (que recibe datos MQTT).
 - Calcula la energía generada multiplicando los valores por el número de baldosas pisadas.
 - Proporciona métodos para acceder a los datos de energía generada y al contador de baldosas pisadas.

```
using UnityEngine;

public class PiezoelectricTile : MonoBehaviour
{
    // Referencia al script que gestiona los datos de la baldosa
(TileManagerData)
    public TileManagerData tileManagerData;

    // Variables para el voltaje, corriente y potencia actual
    private float currentVoltage = 0f;
    private float currentCurrent = 0f;
    private float currentPower = 0f;

    // Corregir el nombre de la variable estática
    private static int tileSteppedOn = 0;

    void OnTriggerEnter(Collider other)
```

```

    {
        // Comprobar si el objeto que entra en contacto con la baldosa
        es un personaje
        if (other.CompareTag("Person"))
        {
            // Incrementar el contador de baldosas pisadas
            tileSteppedOn++;

            // Solo capturar los datos de voltaje, corriente y potencia
            // la primera vez que se pisa
            if (tileSteppedOn == 1)
            {
                if (tileManagerData.GetCurrentVoltage() > 0 &&
                    tileManagerData.GetCurrentCurrent() > 0 && tileManagerData.GetCurrentPower() >
                    0)
                {
                    // Obtener los datos desde el script
                    TileManagerData
                        currentVoltage =
                    tileManagerData.GetCurrentVoltage();
                        currentCurrent =
                    tileManagerData.GetCurrentCurrent();
                        currentPower = tileManagerData.GetCurrentPower();

                    // Mostrar los datos en el log de Unity para
                    // verificar
                    Debug.Log($"Voltaje: {currentVoltage}V, Corriente:
                    {currentCurrent}A, Potencia: {currentPower}W");
                }
            }
        }
    }

    // Método para obtener la potencia generada por esta baldosa
    public float GetCurrentPower()
    {
        return currentPower * tileSteppedOn; // Multiplicar por el
        // número de baldosas pisadas
    }

    public float GetCurrentCurrent()
    {
        return currentCurrent * tileSteppedOn; // Multiplicar por el
        // número de baldosas pisadas
    }

    public float GetCurrentVoltage()
}

```

```

    {
        return currentVoltage * tileSteppedOn; // Multiplicar por el
número de baldosas pisadas
    }

    // Método para obtener el número total de baldosas pisadas
public static int GetTotalTilesSteppedOn()
{
    return tileSteppedOn;
}
}

```

Script Manejo_Baldosas.cs:

- Recopila la energía (potencia, corriente y voltaje) generada por todas las baldosas piezoeléctricas en la lista tiles.
- Suma estos valores para calcular la energía total generada.
- Muestra los resultados en la interfaz de usuario (UI) en tiempo real.

```

using UnityEngine;
using UnityEngine.UI;
using System.Collections.Generic;

public class ManejoBaldosas : MonoBehaviour
{
    // Lista para almacenar todas las baldosas piezoeléctricas
    public List<PiezoelectricTile> tiles;

    // Texto UI para mostrar la potencia, corriente y voltaje totales
generados
    public Text totalEnergyText;

    void Update()
    {
        float totalPower = 0.0f;
        float totalCurrent = 0.0f;
        float totalVoltage = 0.0f;
    }
}

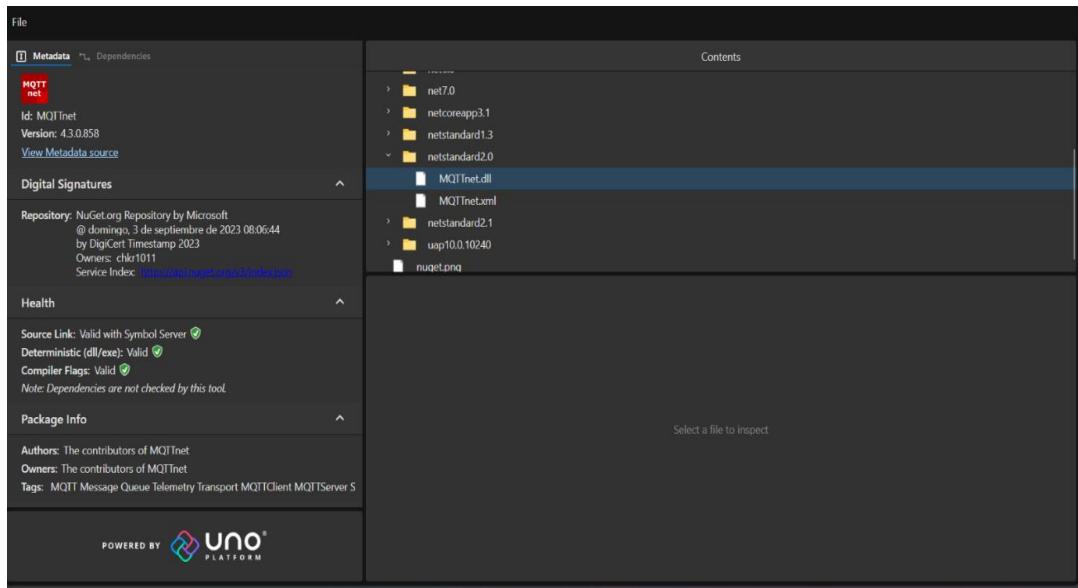
```

```
// Sumar la potencia, corriente y voltaje generados por cada
baldosa
foreach (PiezoelectricTile tile in tiles)
{
    totalPower += tile.GetCurrentPower();
    totalCurrent += tile.GetCurrentCurrent(); // Sumar la
corriente generada por cada baldosa
    totalVoltage += tile.GetCurrentVoltage(); // Sumar el
voltaje generado por cada baldosa
}

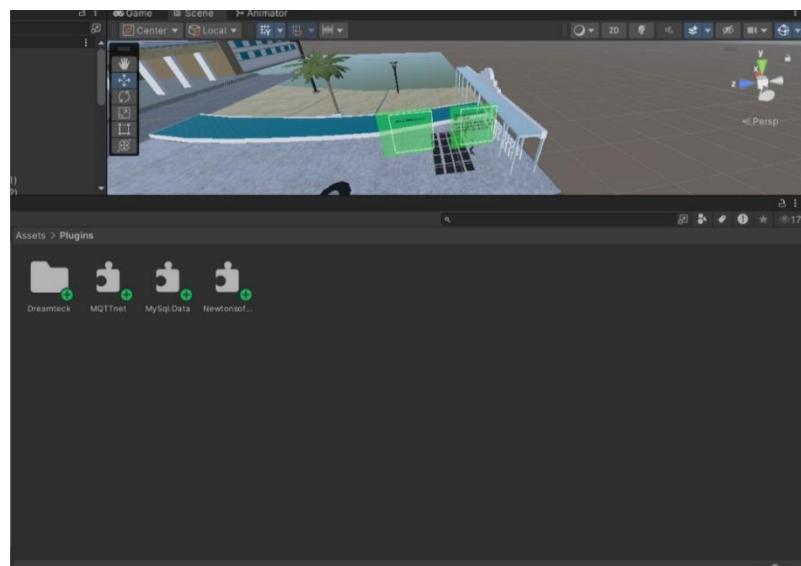
// Mostrar la potencia, corriente y voltaje totales en la UI
int totalTilesSteppedOn =
PiezoelectricTile.GetTotalTilesSteppedOn();
    totalEnergyText.text = "Potencia total generada: " +
totalPower.ToString("F2") + " W\n" +
                    "Corriente total generada: " +
totalCurrent.ToString("F2") + " A\n" +
                    "Voltaje total generado: " +
totalVoltage.ToString("F2") + " V\n" +
                    "Total de baldosas pisadas: " +
totalTilesSteppedOn;
}
}
```

4. Se descargan la librería MQTTnet que permite conectar al broker y de esta manera transmitir los datos al gemelo digital. El enlace de descarga es el siguiente:

<https://www.nuget.org/packages/MQTTnet/4.3.0.858>



5. El archivo .dll de la librería mencionada se la incluye en la carpeta de plugin en Unity.



6. Realizar pruebas de funcionamiento y ajuste de la visualización.

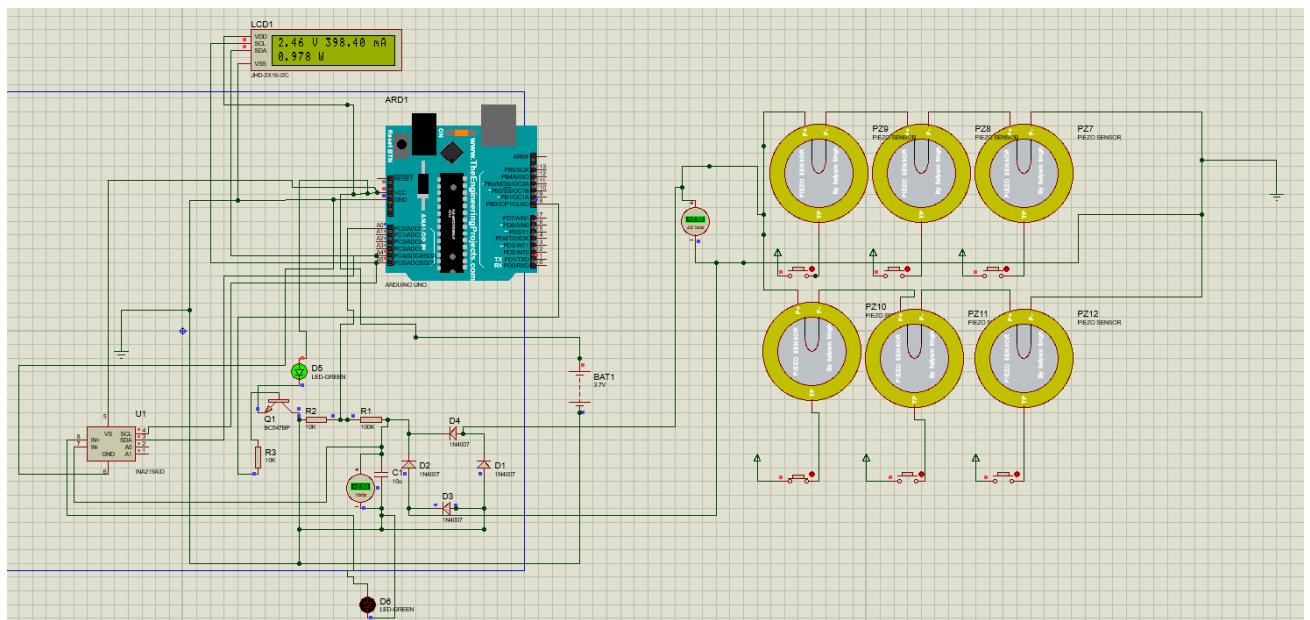
Video presentación del proyecto

[Enlace del video](#)

<https://uceedu->

my.sharepoint.com/:v/g/personal/facueva_uce_edu_ec/Efv9UTuK7HZOvEOXxIPCEt8B7nlEv25-4gifC1aKZeFAFA?e=A0QVgq

Diagrama eléctrico



El circuito diseñado en Proteus consta de varios componentes clave para el correcto funcionamiento del sistema:

1. Generación de Energía Piezoeléctrica:

- Se emplean seis discos piezoeléctricos de 35 mm conectados en un circuito mixto (serie y paralelo) para optimizar la generación de corriente y voltaje.
- Estos discos generan energía eléctrica en respuesta a una fuerza mecánica aplicada sobre ellos.

2. Rectificación de la Señal:

- Dado que la energía generada por los discos piezoelectricos es de corriente alterna (CA), se utilizan diodos rectificadores 1N4007 para convertirla en corriente continua (CC).
- Se implementa un rectificador de onda completa para mejorar la eficiencia en la conversión de la energía.

3. Almacenamiento de Energía:

- Un capacitor electrolítico de $10\mu F$ y 16V es empleado para almacenar la energía rectificada y proporcionar un flujo más estable de corriente cuando cesa la presión sobre los discos.

4. Medición de Voltaje y Corriente:

- Se incorpora el sensor INA219 en serie para medir la corriente y voltaje generados.
- Este sensor se comunica con el ESP32 a través de un bus I2C.

5. Procesamiento y Transmisión de Datos:

- La placa ESP32 recibe las mediciones del INA219 y transmite los datos mediante el protocolo MQTT a una plataforma en la nube, como HiveMQ.
- Esto permite visualizar en tiempo real los valores de corriente y voltaje generados.

6. Activación del LED:

- La energía generada se utiliza para encender un LED rojo, lo que demuestra de manera visual la conversión de energía mecánica en energía eléctrica.

Resultados del experimento

El sistema piezoeléctrico desarrollado logró generar en promedio entre 0.3V y 1V por pulsación. La simulación en Proteus permitió optimizar el circuito, reduciendo pérdidas de energía antes de su implementación física.

La ventaja del sistema destaca en que es una fuente de energía renovable, ya que aprovecha la energía cinética humana para proporcionar una fuente continua y sostenible.

El prototipo real, integrado con Arduino ESP32; se logró transmitir los datos recolectados por el sensor INA219 en tiempo real con un lapso de 1 segundo entre reporte y reporte. El servidor MQTT mostró parámetros clave como voltaje y potencia generada, facilitando la supervisión y graficación del sistema. En pruebas prolongadas, el sistema mantuvo una estabilidad. A nivel teórico, aunque la generación energética no es suficiente para alimentar grandes infraestructuras, se demostró su viabilidad para dispositivos de bajo consumo, como sensores IoT.

La desventaja del modelo es que requiere de una fuerte inversión inicial, mantenimiento periódico y monitoreo para garantizar su rendimiento óptimo. Además, su potencia de salida es limitada, y depende en gran medida del tráfico peatonal,

El gemelo digital en Unity permitió simular la entrada principal de la UCE, mostrando que un tráfico promedio de *50* personas por hora generaría hasta *32.39V* útiles (cada una). Aunque los datos no eran exactos, la simulación podría ayudar a optimizar la ubicación de los sensores antes de la implementación física.

El proyecto demostró que la energía piezoeléctrica puede integrarse en entornos universitarios inteligentes, reduciendo la dependencia de fuentes convencionales y promoviendo la sostenibilidad.

Conclusiones

El experimento comprobó que el sistema piezoeléctrico genera entre 0.3V y 1V por pulsación, lo cual, junto con la optimización realizada en la simulación (Proteus), valida su capacidad para convertir energía mecánica en energía eléctrica de forma sostenible.

El prototipo real, que utiliza un Arduino ESP32 y el sensor INA219, demostró una transmisión de datos en tiempo real (cada 1 segundo) y mantuvo estabilidad en pruebas prolongadas, lo que lo hace viable para alimentar dispositivos de bajo consumo, como sensores IoT, en entornos inteligentes.

La simulación mediante un gemelo digital en Unity, aplicada a la entrada de la UCE, mostró que un tráfico peatonal moderado (50 personas por hora) puede generar suficiente energía para aplicaciones de bajo consumo. Esto sugiere que, aunque la potencia es limitada, el sistema puede integrarse efectivamente en infraestructuras universitarias para mejorar la sostenibilidad.

Recomendaciones

Fomentar la conexión entre la energía piezoeléctrica, IoT y otras fuentes renovables (como paneles solares) para crear soluciones energéticas integrales que optimicen la gestión de recursos y fortalezcan la sostenibilidad de la infraestructura.

Iniciar la implementación del sistema para alimentar dispositivos de baja demanda energética (por ejemplo, sensores, iluminación LED en áreas estratégicas) en la UCE. Esto

permitirá evaluar y ajustar el rendimiento antes de considerar aplicaciones de mayor escala o integración con otras fuentes de energía renovable.

Establecer protocolos de monitoreo y mantenimiento periódico del sistema, utilizando tecnologías de IoT y gemelos digitales (como el modelo en Unity) para optimizar la ubicación de sensores y mejorar el rendimiento. Esto asegurará que el sistema se mantenga operativo y se ajuste dinámicamente a las variaciones en el tráfico peatonal, garantizando una operación óptima y sostenible.

Por otro lado, la implementación de una batería sería fundamental para almacenar la energía generada por las baldosas piezoeléctricas, permitiendo un uso más eficiente y continuo de la energía. La batería actuaría como un buffer, almacenando la energía durante los períodos de alta actividad (cuando hay muchas pisadas) y liberándola cuando sea necesario, por ejemplo, para cargar dispositivos móviles o alimentar sistemas de iluminación. Se recomienda utilizar una batería de 12V ya que es compatible con los voltajes generados por los sensores piezoeléctricos después de la rectificación. Además, la inclusión de un sistema de monitoreo con un microcontrolador, como el Arduino Uno, permitiría supervisar el estado de la batería y optimizar su carga y descarga. Esto no solo mejoraría la eficiencia del sistema, sino que también permitiría notificar al usuario sobre el estado de la batería, como cuando está baja o completamente cargada, lo que es especialmente útil en aplicaciones prácticas como estaciones de carga públicas o sistemas de iluminación autónomos.

Bibliografía

- [1] J . A. R. Morales, “Esp32 características y pines,” *PASIÓN ELECTRÓNICA*, Feb. 12, 2022. https://pasionelectronica.com/esp32-caracteristicas-y-pines/?utm_source=chatgpt.com
- [2] C. De Sousa and L. Manganiello, “Estado del Arte: Aplicaciones de los sensores piezoelectricos en la detección de elementos contaminantes en alimentos,” 2018. <https://www.redalyc.org/journal/707/70757670014/html/>
- [3] L. Llamas, “Medir tensión, intensidad y potencia con Arduino y INA219,” *Luis Llamas*, May 10, 2020. https://www.luisllamas.es/medir-tension-intensidad-y-potencia-con-arduino-y-ina219/?utm_source=chatgpt.com
- [4] E. O. Bit, “Protoboard - Placa de pruebas para Arduino,” *El Octavo Bit*, May 26, 2020. <https://eloctavobit.com/modulos-sensores/protoboard-placa-de-pruebas>
- [5] “Cables jumper macho-hembra • Factor Evolución,” *Factor Evolución*, Oct. 24, 2018. <https://www.factor.mx/portal/base-de-conocimiento/cables-jumper-macho-hembra/>
- [6] “Cables jumper macho-macho • Factor Evolución,” *Factor Evolución*, Oct. 24, 2018. <https://www.factor.mx/portal/base-de-conocimiento/cables-jumper-macho-macho/>
- [7] AV Electronics, “Diodo LED alto brillo 5mm - AV Electronics,” *AV Electronics*, Jan. 23, 2025. <https://avelectronics.cc/producto/diodo-led-alto-brillo-5mm/>
- [8] “Cable para protoboard (metro),” *Teslatronica Sumador S.A.S. - NIT: 901640333-1*. <https://sumador.com/products/cable-para-protoboard>
- [9] Redeweb and Redeweb, “Qué es un Diodo Rectificador: Funcionamiento y Tipos,” *Revista Española De Electrónica / Todas Las Noticias De Electrónica Actualizadas a Diario*, Oct. 08, 2024. <https://www.redeweb.com/actualidad/que-es-un-diodo-rectificador/>
- [10] MEGATRONICA, “Condensador electrolítico 10UF 16V - MEGATRONICA,” *MEGATRONICA*, Feb. 23, 2024. https://megatronica.cc/producto/condensador-electrolitico-10uf-16v/?srsltid=AfmBOorNZ5nvffIXfHQQDJTKYccWtIKD3oVaCHgdRchXZ8HGCxf_Rwux
- [11] “Bateria 18650 4.2V 8800mAh | La Bobina de Tesla,” *La Bobina De Tesla*. <https://www.labobinadetesla.com/product-page/bateria-18650-4-2v-8800mah>

[12] “PORTA PILAS PARA X2 BATERÍAS MODELO 18650 PARA PCB -5,”
SSDIELECT ELECTRONICA SAS. <https://ssdiselect.com/socket-para-baterias/3840-holder-18650-pcb-x2.html>

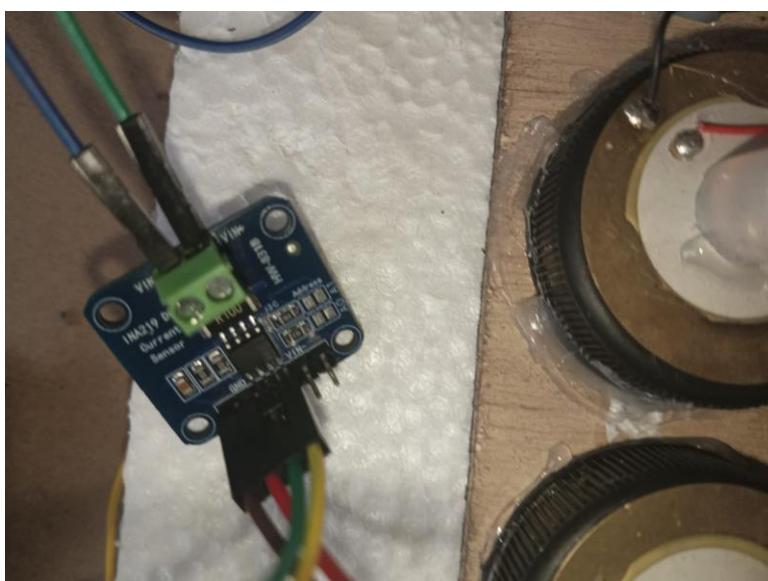
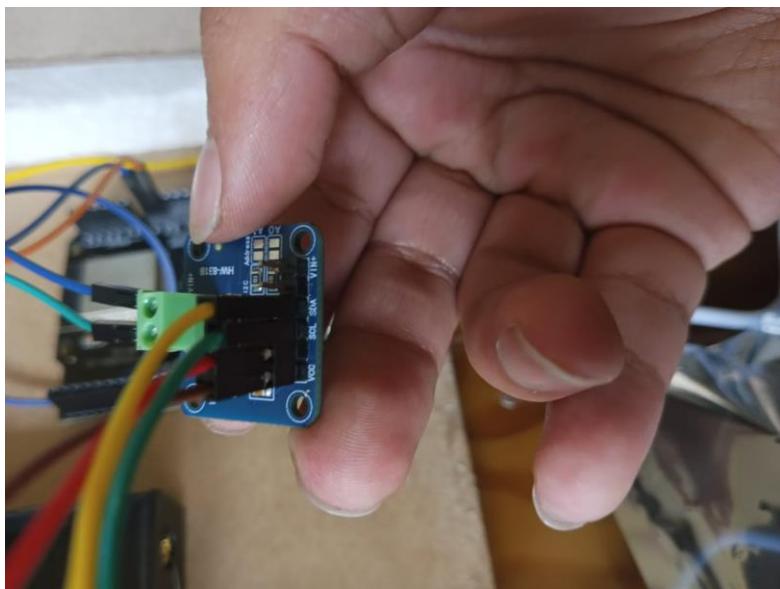
[13] “DreamTeck SPLines | Utilities Tools | Unity Asset Store,” *Unity Asset Store*, Aug. 06, 2021. <https://assetstore.unity.com/packages/tools/utilities/dreamteck-splines-61926>

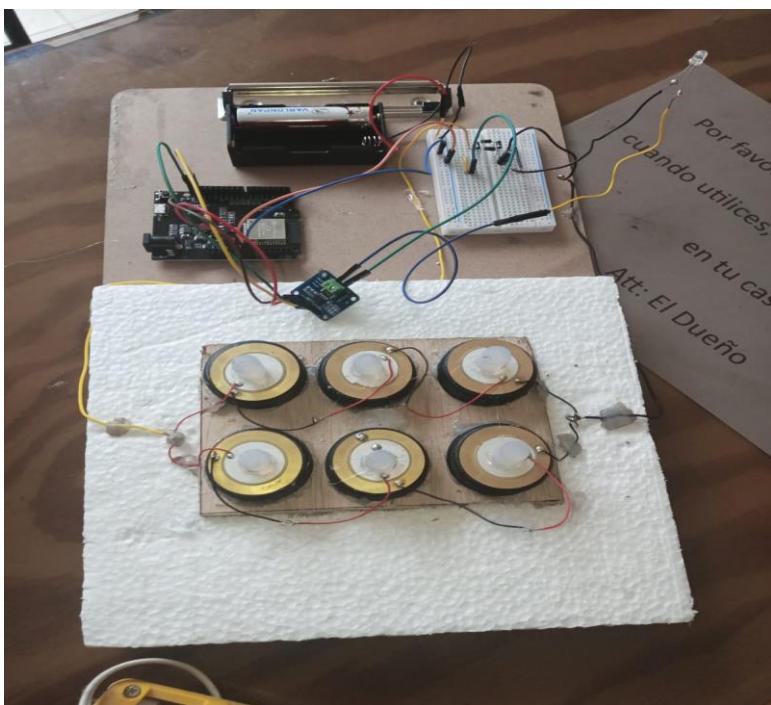
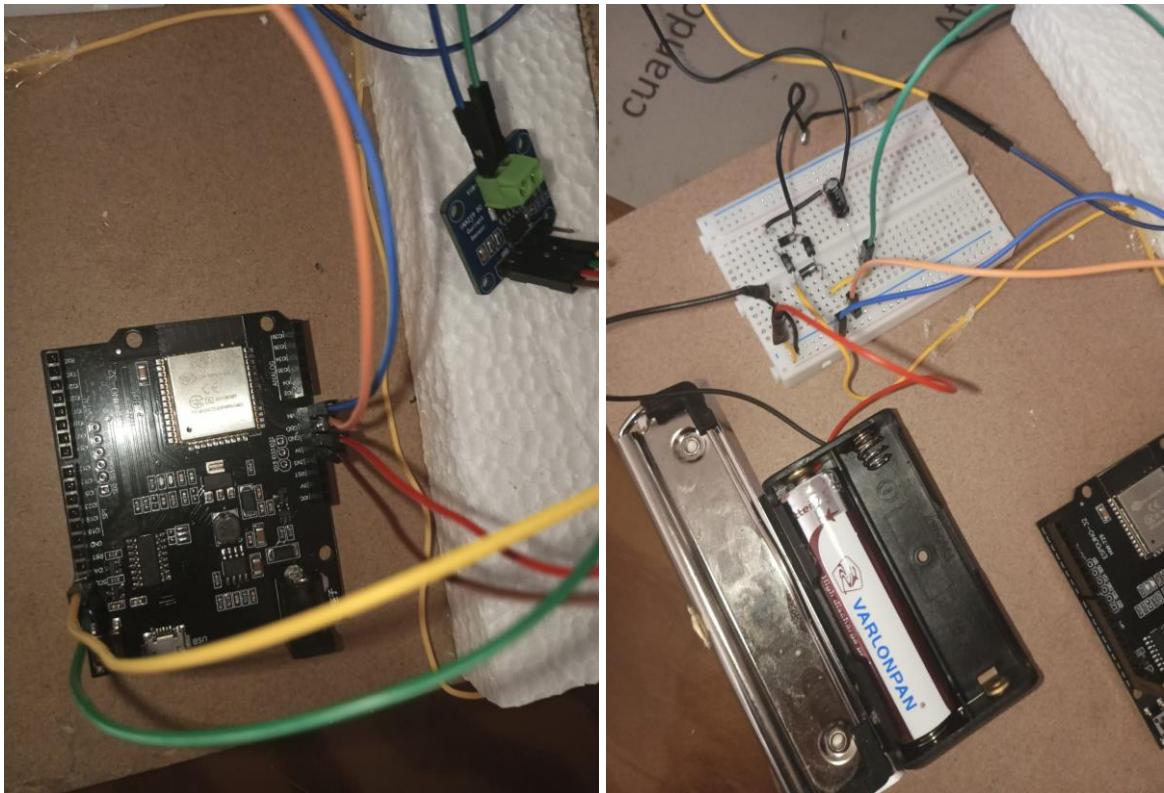
[14] L. & T. F. Images, “Unas baldosas generan energía con nuestras pisadas,” *National Geographic*, Nov. 09, 2017. <https://www.nationalgeographic.es/medio-ambiente/unas-baldosas-generan-energia-con-nuestras-pisadas>

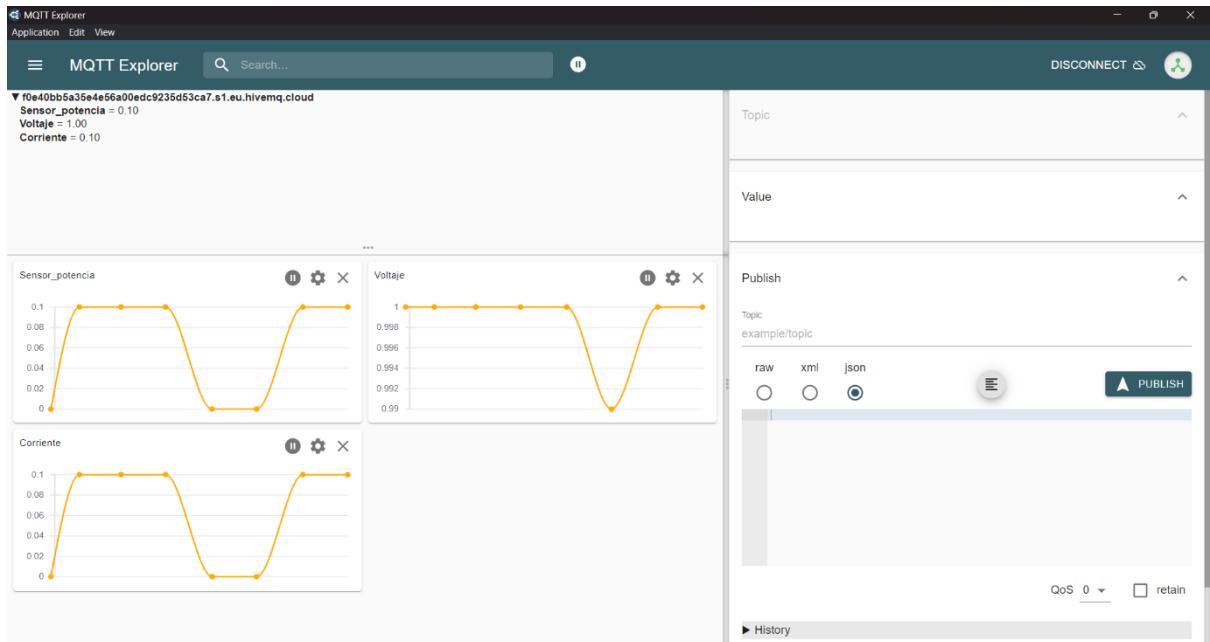
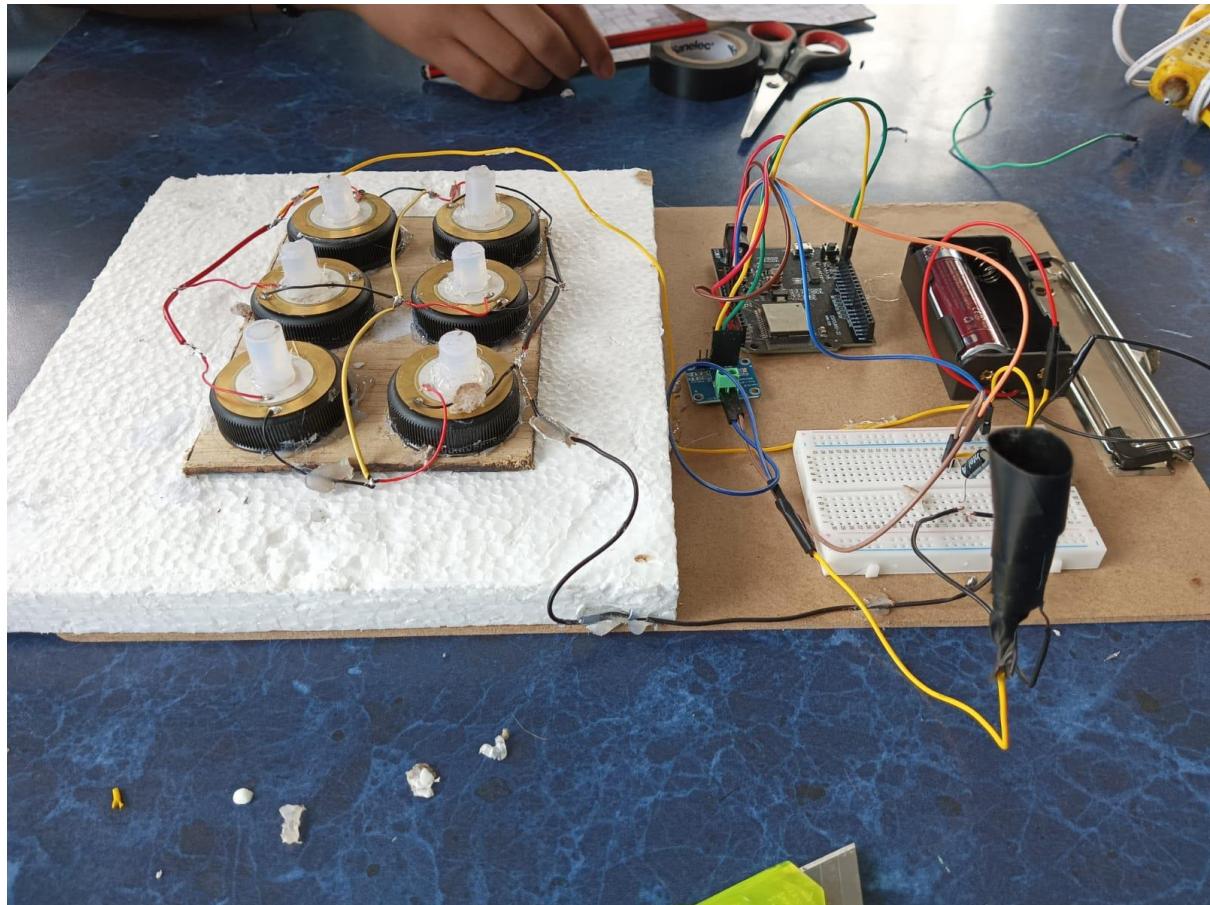
[15] “Pavegen. Diseño urbano inteligente y energías limpias,” *Arquitectura*, Apr. 12, 2022. <https://arquitecturayempresa.es/noticia/pavegen-diseno-urbano-inteligente-y-energias-limpias>

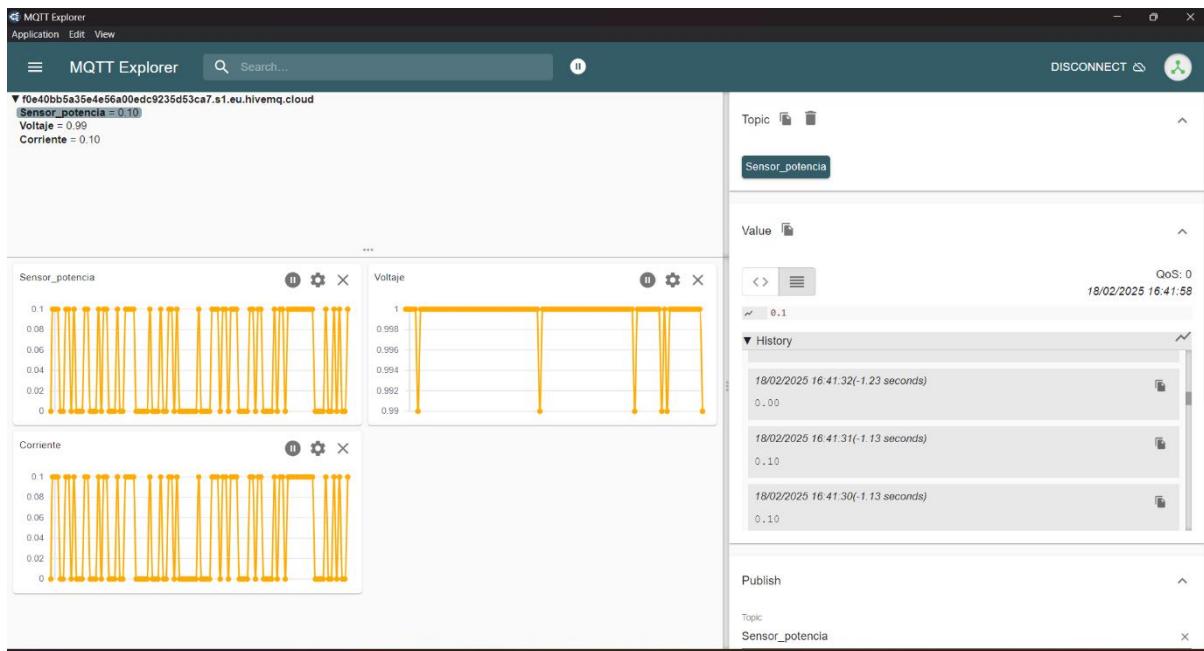
Link archivo del modelo digital Unity y base de datos:
[archivos unity datos](#)

Anexos









The screenshot shows the HiveMQ Cloud Web Client interface. At the top, there's a navigation bar with 'OVERVIEW', 'ACCESS MANAGEMENT', 'INTEGRATIONS', 'WEB CLIENT' (which is highlighted in blue), and 'GETTING STARTED'.

The left sidebar contains several icons and sections:

- Web Client**: A brief introduction to the client.
- Connection Settings**: A form for entering a username ('Username *') and password ('Password *'). The 'Username' field contains 'Grupo4' and the 'Password' field contains '*****'. A red 'Disconnect' button is below the fields.
- Topic Subscriptions**: A section for managing topic subscriptions.

The main content area includes a 'Messages' section on the right showing a list of received messages:

	Topic	QoS
42	Voltaje	0
	1.00	
43	Sensor_potencia	0
	0.10	
44	Corriente	0
	0.10	
45	Sensor_potencia	0
	0.10	
46	Voltaje	0
	1.00	
47	Corriente	0
	0.10	
48	Voltaje	0
	1.00	
49	Sensor_potencia	0
	0.10	

A green message bar at the bottom of the sidebar says 'The WebClient is connected'.

