CARPETA TÉCNICA

7° 1° Aviónica *Comisión "C"*

IMPA TRQ E.E.S.T N°7 2025

Victoria Baza – Mauricio Blasco – Raúl Broncano – Ignacio García
Isidro Stabile – Santiago Tejeda

Integrantes

Victoria Josefina Baza

DNI: 47750221

Tel.: 11 6246-0953

Mail: victoriajosefinabaza@gmail.com

Fecha de Nacimiento: 10/01/2007

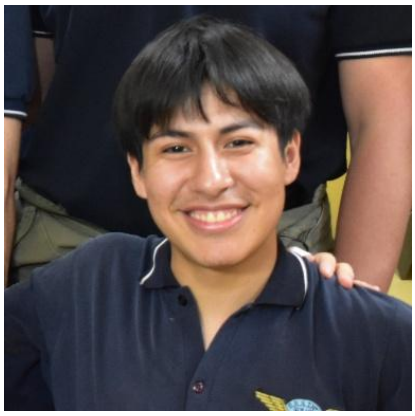
Mauricio Blasco

DNI: 47877887

Tel.: 11 3649-1857

Mail: germauricio975@gmail.com

Fecha de Nacimiento: 22/05/2007



Víctor Raúl Broncano Ramos

DNI: 48159076

Tel.: 11 6679-4686

Mail: raul455896@gmail.com

Fecha de Nacimiento: 29/01/2007

Ignacio García Louzan

DNI: 47349722

Tel.: 11 6155-1737

Mail: nachogarcialouzan@gmail.com

Fecha de Nacimiento: 19/07/2006

Isidro Joaquín Stabile

DNI: 47484606

Tel.: 11 3170-7976

Mail: isidrost@hotmail.com

Fecha de Nacimiento: 01/08/2006

Santiago Alejandro Tejeda

DNI: 47739884

Tel.: 11 2695-8907

Mail: santiagotejeda36@hotmail.com

Fecha de Nacimiento: 16/01/2007

Docentes tutores

- Arguello, Gustavo
- Bianco, Carlos
- Carlassara, Fabrizio
- Medina, Sergio
- Palmieri, Diego

Fecha de Inicio

- 7 de Abril de 2025

Esfuerzo del proyecto

- 31 Semanas de Trabajo
- 220 Horas de Trabajo en total

Redes y enlaces de interés

- Página Web
  https://pleper-impa.github.io/web/
- Instagram
  https://www.instagram.com/pleper_25/
- GitHub
  https://github.com/impatrq/Pleper
- Gmail
  pleper.infoimpa@gmail.com

Agradecimientos

Les agradecemos a todos los que colaboraron con nuestro proyecto. Pleper.

Índice

1. Introducción

PLEPER, por sus siglas “Placa de Energía Piezoeléctrica enovable” es una baldosa que permite que las personas que pasen por encima de ella generen una pequeña cantidad de energía eléctrica, permitiéndoles así colaborar con la generación de una energía autosustentable que tiene como fin almacenar suficiente carga para su aplicación en casos de emergencia, asegurando así que haya un medio de proveer luz en un corte de electricidad por un tiempo suficiente para ponerse en resguardo.

El sistema se basa en una placa que explota el fenómeno de la piezoelectricidad, aprovechando los pulsos generados por los sensores cerámicos al ser deformados mecánicamente. Su diseño de baldosa es especialmente útil para poder implementarlo en vías públicas donde hay un alto tránsito y se puede aprovechar mejor la generación de energía eléctrica.

Aspiramos a lograr demostrar que es posible desarrollar formas alternativas de generar energía, concientizando a las personas mediante el uso de la misma placa y haciéndolos participes del cambio.

1.1. Resumen del proyecto

El proyecto se centra en el desarrollo de una baldosa con distintas capas la cual, al presionarla, generará un pequeño pulso eléctrico que se busca aprovechar mediante el almacenamiento de esta energía. El sistema está constituido principalmente por los siguientes componentes: una baldosa de madera como base, distintas bases de caucho que sostienen 32 sensores piezoeléctricos, cuatro resortes en cada esquina, topes de goma entre los sensores piezoeléctricos y la placa de madera que esta encima, una baldosa de caucho como superficie para pisar, un módulo de almacenamiento con 4 capacitores de 10uF, un microcontrolador y una luz de emergencia.

El funcionamiento del sistema es el siguiente: el usuario caminará por encima de la baldosa como si fuera una placa más del piso, generando presión mecánica sobre esta y luego permitiendo que vuelva a su estado original. En esta deformación mecánica, los sensores piezoeléctricos generaran un pequeño pulso que será rectificado y almacenado en varios capacitores conectadas en paralelo.

Una vez varios usuarios, o el mismo haciendo distintas pasadas, pisen la placa, esta energía almacenada será distribuida por un microcontrolador, el cual la utilizará para alimentar un cartel de emergencia, simulando una situación donde la energía eléctrica se fuera en una estación de Subte, un lugar muy concurrido que, al pasar muchas personas, lograría almacenar una buena carga para alimentar el cartel de emergencia, pudiéndose aprovechar.

1.2. Motivación

Como estudiantes de séptimo año, buscamos desarrollar un proyecto con el fin de cumplir con el requerimiento horario de practicas profesionalizantes. Nuestra intención inicial fue desarrollar algo relacionado a energías alternativas, en vista del creciente interés por el desarrollo de energías renovables con el fin de apaciguar las consecuencias experimentadas por el cambio climático. Bajo este marco, y en búsqueda de alternativas no tan exploradas aún, decidimos tomar como base de nuestro proyecto el efecto piezoeléctrico, no tan visibilizado. Creemos que este proyecto tiene el potencial de concientizar a más personas sobre la necesidad de buscar alternativas no tan destructivas a nuestro ecosistema, visibilizando un fenómeno que, si bien se ha explorado en algunas partes del mundo, poca gente conoce su existencia y potencial.

1.3. Estado del arte

1.3.1. Pavegen

Uno de los referentes que analizamos para la conceptualización del proyecto fue Pavegen, una empresa británica dedicada al desarrollo de baldosas inteligentes capaces de transformar la energía de las pisadas humanas en electricidad utilizable. Estas baldosas funcionan mediante un sistema de inducción electromagnética y movimiento mecánico, generando pequeñas cantidades de energía cada vez que una persona las pisa.

El objetivo principal de esta tecnología es aprovechar el tránsito peatonal en espacios públicos para alimentar sistemas de bajo consumo, como luminarias LED, pantallas interactivas o sensores de monitoreo. Además, Pavegen incorpora herramientas de recolección de datos que permiten medir la cantidad de pasos y el flujo de personas en un lugar determinado, contribuyendo así al desarrollo de entornos urbanos más inteligentes y sostenibles.

Este ejemplo fue uno de los primeros que hallamos al investigar sobre aportes a la utilización de la energía piezoeléctrica, nos permitió reflexionar sobre la viabilidad de utilizar la energía cinética como fuente renovable aplicada a la vida cotidiana. Si bien nuestro proyecto no busca replicar la complejidad de Pavegen, nos resultó relevante como antecedente que demuestra cómo la energía generada por el movimiento humano puede almacenarse y aprovecharse de manera innovadora.

1.3.2. Subtes en Tokio

Otro caso que analizamos fue el de los sistemas de transporte en Tokio, donde se han implementado tecnologías piezoeléctricas en estaciones de metro con el objetivo de generar energía a partir del tránsito masivo de pasajeros. Estas instalaciones utilizan baldosas especiales que convierten la presión de las pisadas en electricidad, la cual luego se emplea para alimentar iluminación de bajo consumo, paneles informativos y otros dispositivos dentro de las estaciones.

El contexto japonés resulta particularmente interesante debido al alto caudal de personas que circula diariamente por el metro de Tokio, lo que permite obtener un volumen considerable de energía renovable a partir de una acción cotidiana como caminar. Además, estas iniciativas se enmarcan dentro de las políticas de sostenibilidad y eficiencia energética que caracterizan a muchas ciudades japonesas, donde la innovación tecnológica se aplica directamente a la vida urbana.

Este ejemplo nos permitió comprender cómo la energía piezoeléctrica puede ser aplicada a gran escala en espacios públicos con alto flujo de usuarios. Si bien nuestro proyecto no busca alcanzar esa magnitud, consideramos valiosa esta experiencia como inspiración para adaptar la misma lógica en un prototipo más reducido y demostrativo.

1.4. Objetivos

El objetivo de nuestro proyecto es el de lograr generar energía utilizando como fuente principal proveedora la deformación mecánica que generan las pisadas y ser capaces de almacenarla y aprovecharla para alimentar un LED que sirva como luminaria de emergencia, pudiendo generar energía de una forma no convencional en casos de emergencia.

1.5. ¿Quiénes Somos?

Somos el grupo PLEPER (Placa de Energía Piezoelectrica Renovable), conformado por seis estudiantes de la especialidad de Aviónica del séptimo año, primera división, comisión C, de la Escuela Técnica N°7 IMPA “Taller Regional Quilmes”.

1.6. Desarrollo del Proyecto

A continuación, se presenta un diagrama en bloques que implicó la columna vertebral de nuestro desarrollo, centrándonos en cumplir lo presente aquí pero

haciendo las modificaciones necesarias al encontrar una oposición a la idea original:

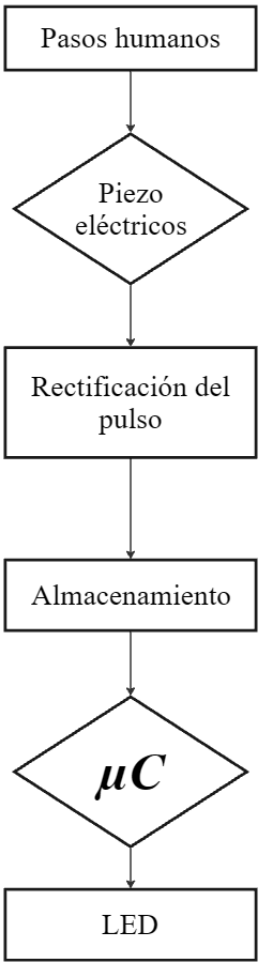


Figura 1: diagrama de flujo representando el eje de desarrollo propuesto

2. Estructura

2.1. Diagrama General de la Estructura

Se utilizo el programa AutoCAD 2025 para el diseño de piezas 3D como los pilares y los topes de los piezoeléctricos.

2.1.1. Diseño de la baldosa

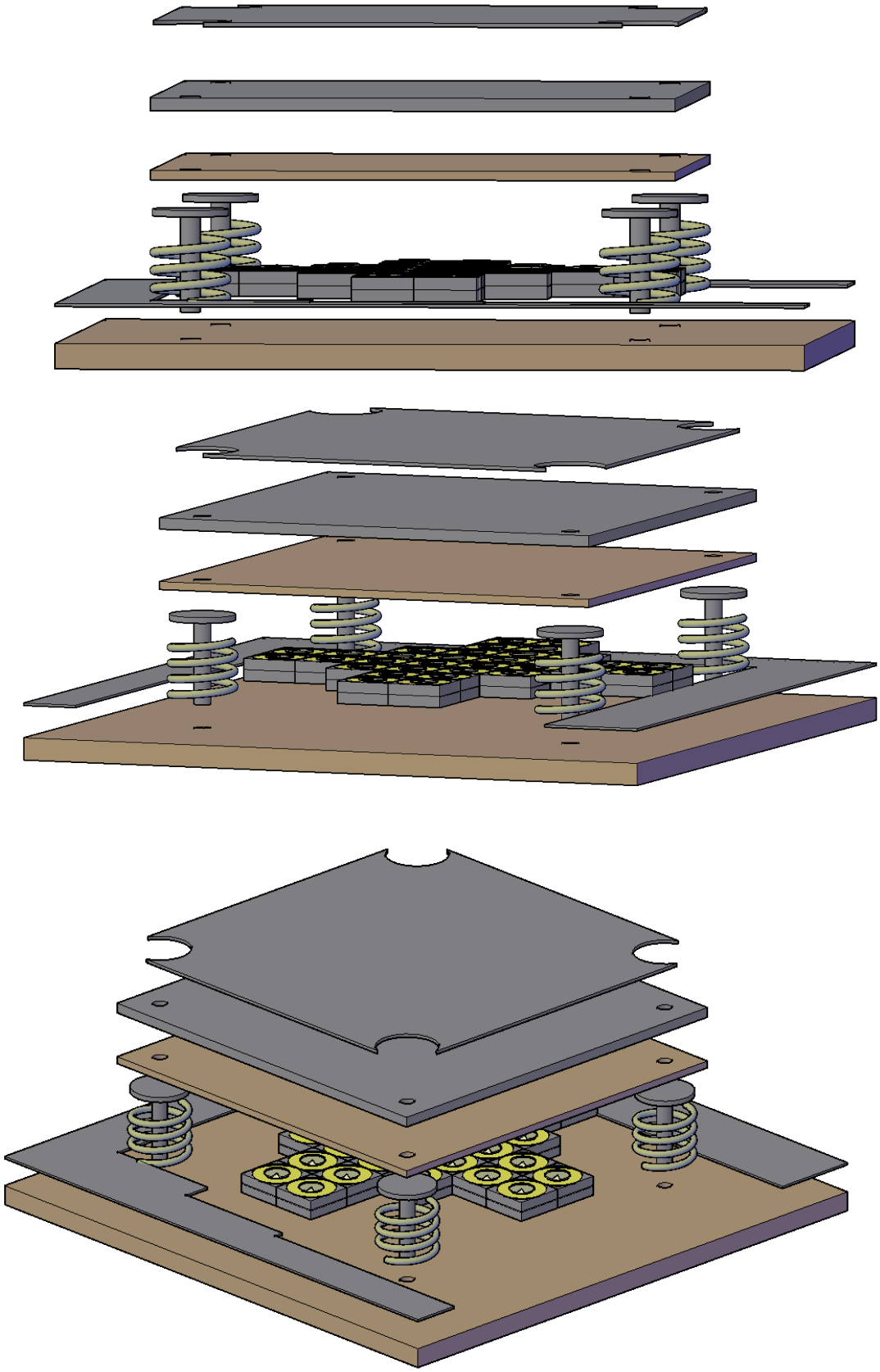
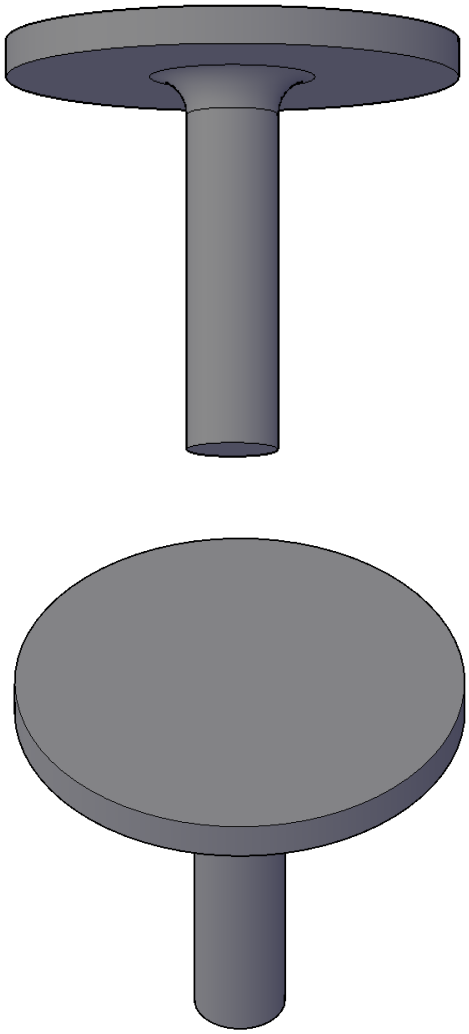


Figura 2, 3 y 4: Vistas del diseño de la estructura en AutoCAD

2.1.2. Diseño de los pilares



Figuras 5 y 6: Vistas de los pilares diseñados en AutoCAD

2.2. Descripción y Diseños de Partes Estructurales

2.2.1. Base

Consiste en una tabla elaborada con madera terciada de 7 capas, con un área de 42x42 cm y agujereada en las cuatro esquinas con orificios de 13mm de diámetro y 5mm de profundidad para la colocación de los pilares 3D. Posee también una alfombra decorativa sobre ella. En el medio de la base se distribuyeron los piezoeléctricos para aprovechar al máximo la tensión generada.

Figura 7: Base de madera utilizada

2.2.2. Pilares 3D

Para mantener la baldosa firme, se añadieron pilares en cada esquina. Fueron diseñados en AutoCAD e impresos en 3D. Se forman por una base circular de 12mm de diámetro y 65mm de altura, y sobre ella un círculo de 60mm de diámetro y 5mm de altura. La conexión entre ambas partes fue redondeada para repartir eficientemente el peso. Estos pilares fueron pegados con silicona a los agujeros de la base.

Figura 8: Pilaress impresos en 3D

2.2.3. Resortes

Para que la baldosa regrese a su posición original tras ser pisada, se utilizaron cuatro resortes en las esquinas. Poseen una altura de 4,75 cm, un radio exterior de 5,93 cm y 3,5 vueltas. Para mantenerlos en posición, se pegó una estructura circular de 46mm (diámetro interno) y 51mm (diámetro exterior).

Figura 9: Resortes utilizados

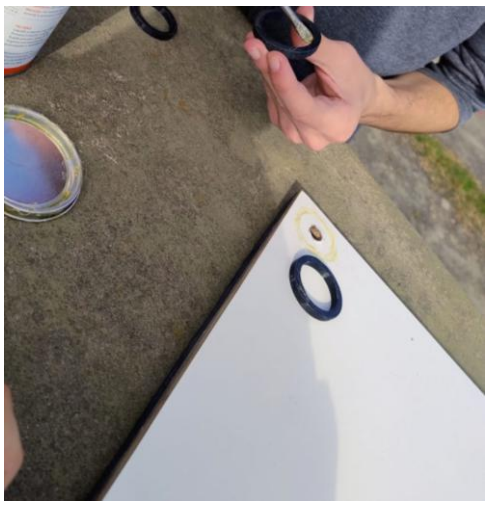


Figura 10: Anillos para mantener los resortes

2.2.4. Ensamblaje Piezoeléctrico

2.2.4.1. Bases de Piezoeléctricos:

Se requería una base que aumente la altura en la que se ubican los piezoeléctricos, además de permitirlos deformarse y amortiguar el impacto por lo que no podía ser echa de un material rígido. Tras analizar detenidamente su forma decidimos realizarlos a base de caucho, el cual repartimos sobre las medidas hechas anteriormente en la base de madera. Recortamos el caucho en rectángulos con una base de 5x5 con una altura de 2cm y los pegamos a la base utilizando adhesivo de contacto.

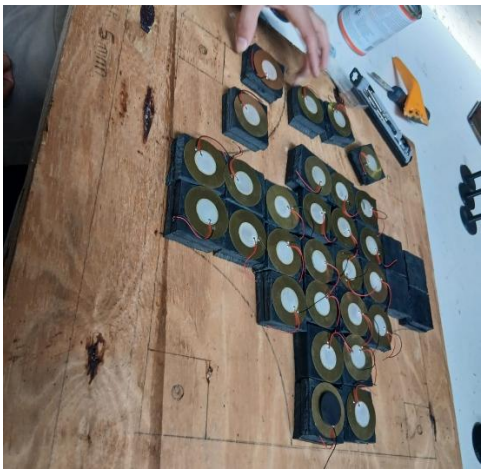


Figura 11: Bases de piezoeléctricos con ellos

2.2.4.2. Sensores Piezoeléctricos:

Se colocaron sobre las bases de caucho en grupos de cuatro conectados en paralelo. Cada piezoeléctrico posee un diámetro de 5cm (negativo) y 2,5cm (positivo). Se pegaron con adhesivo de contacto.

Figura 12: Sensor piezoeléctrico

2.2.4.3. Topes:

Para aumentar la eficiencia y reducir el desgaste de las soldaduras, se colocaron topes cúbicos. Estos topes fueron elaborados en AutoCAD e impresos en 3D.

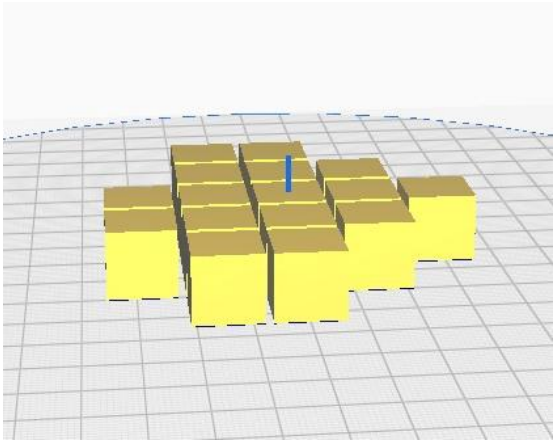


Figura 13: Diseño 3D de los topes cúbicos

Figura 14: Topes cúbicos

2.3.5. Tapa (Superficie de Pisada)

2.3.5.1. Madera:

Base de 42x42 cm con cuatro agujeros de 13mm en las esquinas por donde pasan los pilares 3D.

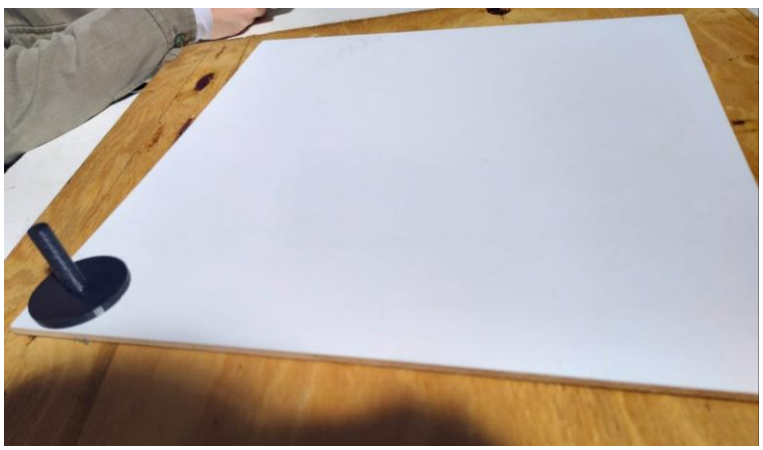


Figura 15: Madera de base para la tapa

2.3.5.2. Caucho:

La capa de caucho mide 42 cm x 42 cm, y posee un agujero en cada esquina con un diámetro de 13mm. Esta capa está colocada para disminuir el contacto del pie con la madera y aumentar su vida útil. Sobre este se ubica una alfombra antideslizante que evita que se ensucie de forma tan rápida la baldosa y a su vez facilita la limpieza de esta.

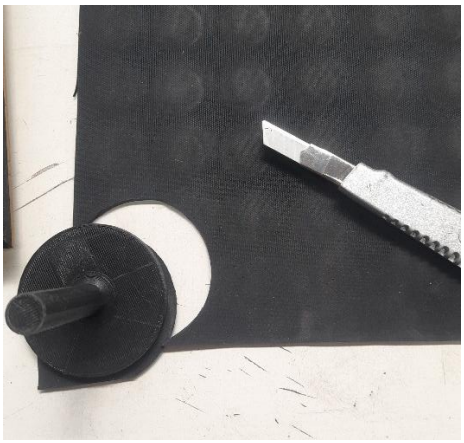


Figura 16: Caucho de alfombra para la tapa

2.2.6. Gabinete de display

La caja de madera del sistema fue diseñada para proteger y alojar los componentes electrónicos del entorno del display de manera robusta, ordenada y estética. Su principal función es servir como gabinete contenedor del sistema completo, asegurando una correcta organización del circuito, estabilidad mecánica y seguridad durante su uso o exposición.

Es una caja de 20 cm x 40 cm con 15 cm de altura, está elaborada con madera de un grosor de 2cm. En la parte frontal se expande 24cm hacia arriba para poder atornillar el cartel de emergencia, además posee una ranura a 11 cm de los costados a una altura de 7,5 cm, la cual tiene un ancho de 18 cm y una altura de 3 cm. En la parte trasera se encuentra un agujero en la que se colocaría la puerta que permitiría abrir y cerrar la caja. Por ultimo en la parte inferior se ubica un círculo de 4 cm de diámetro para poder conectar el tubo PVC.

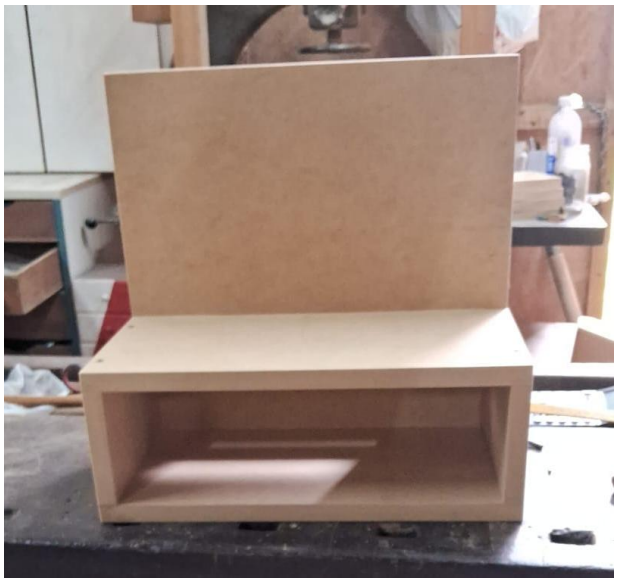


Figura 17: Gabinete de madera

2.2.7. Tubo PVC

Se agregó un tubo PVC para que el display se encuentre a la altura de la vista de las personas y no resulte incomodo, el cual tiene 4 cm de diámetro para mejorar su rigidez y una altura de 1,3 metros acoplado a la base de la baldosa por medio de dos soportes en L que lo sostienen por medio de los tornillos. Posee un corte en la parte inferior de 1 cm x 1 cm que permite pasar los cables por su interior. Decidimos pintar este tubo de color negro para mejorar la estética de este.

Figura 18: Tubo PVC

3. Electrónica

3.1. Software utilizado para el desarrollo de esquemáticos y PCB

Principalmente se utilizó KiCad 9.0 tanto para diseñar los esquemáticos como para los diseños PCB. Se utilizó el visualizador de modelos 3D integrado para poder ver cada placa desarrollada.

3.2. Esquemático de cada bloque

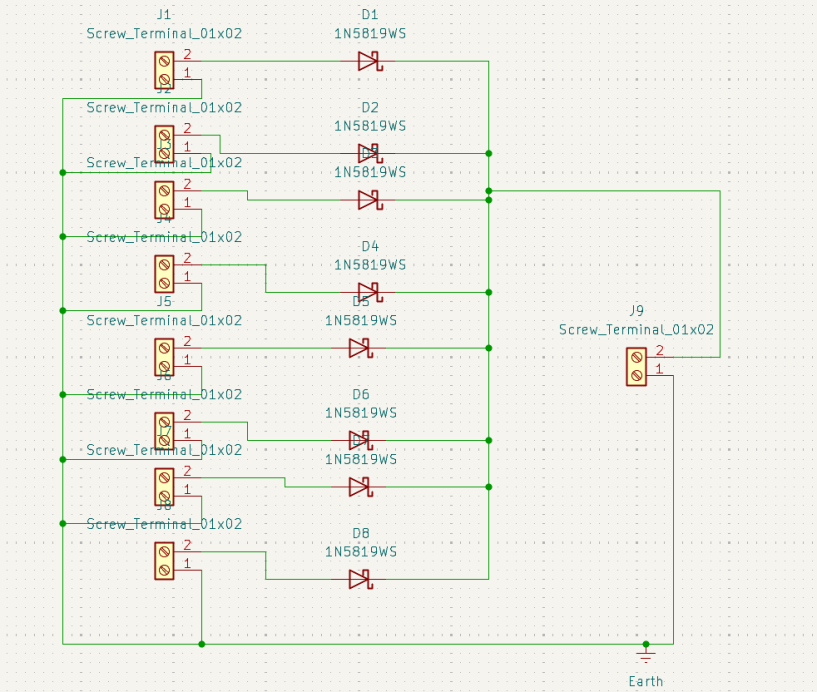


Figura 19: Esquemático de la “Placa Rectificadora”

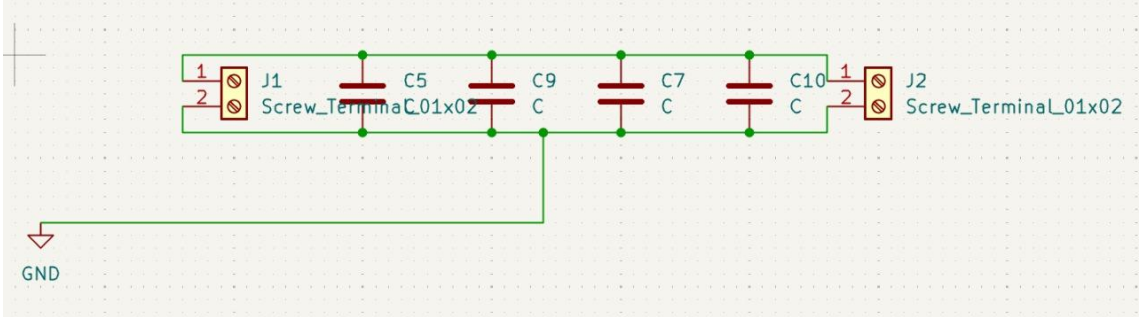


Figura 20: Esquemático del “Banco de Capacitores”

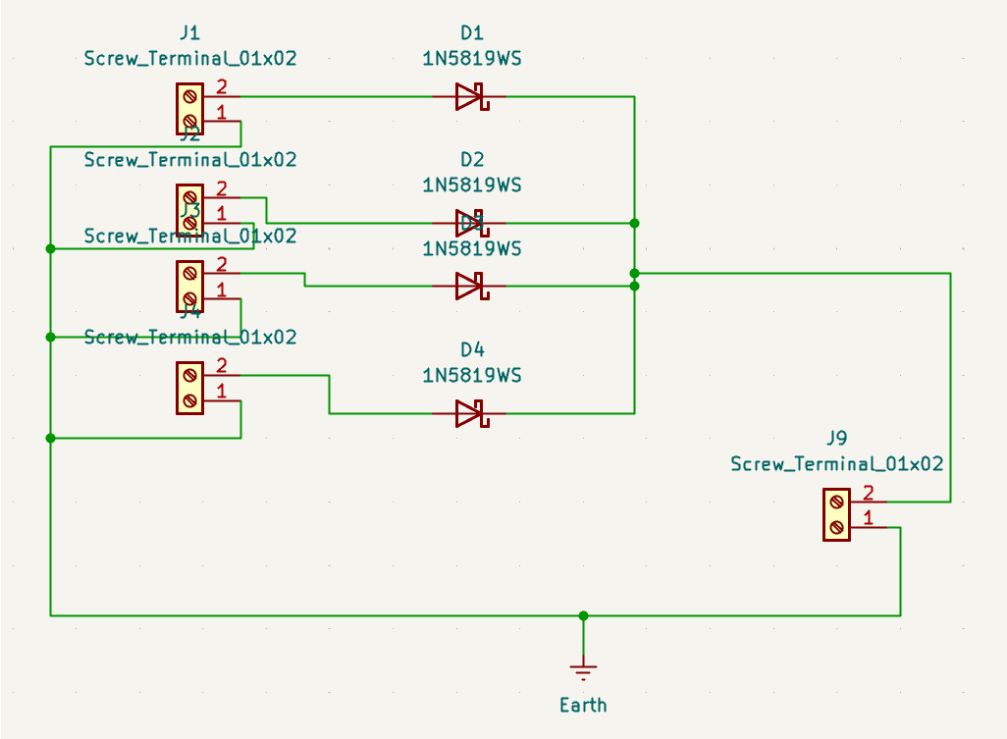


Figura 21: Esquemático de la “Segunda Placa Rectificadora”

3.3. PCB de cada bloque

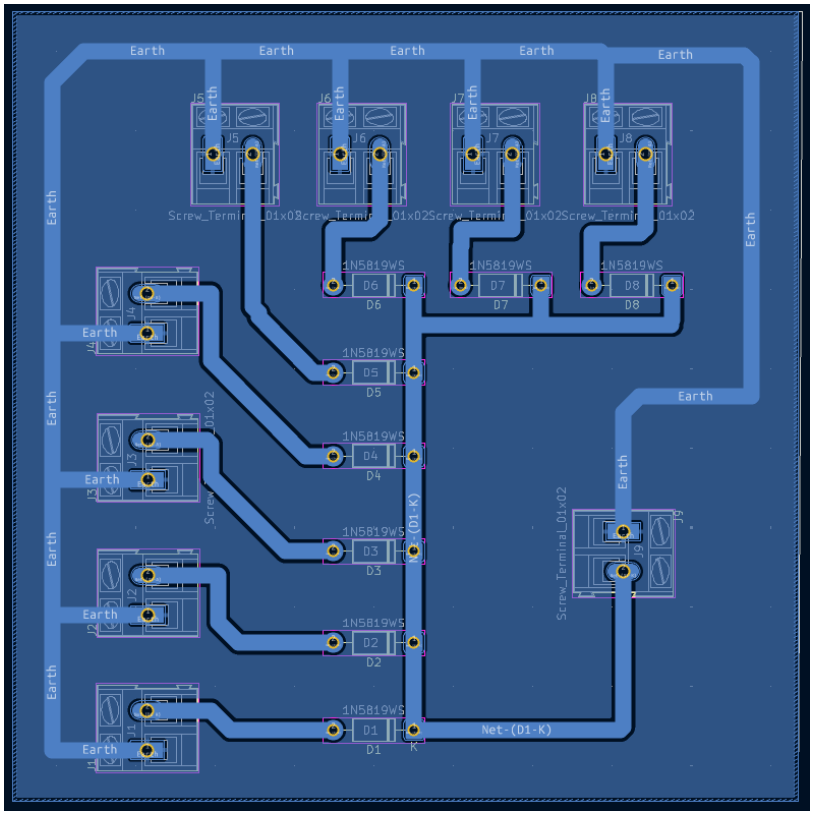


Figura 22: PCB de la “Placa Rectificadora”

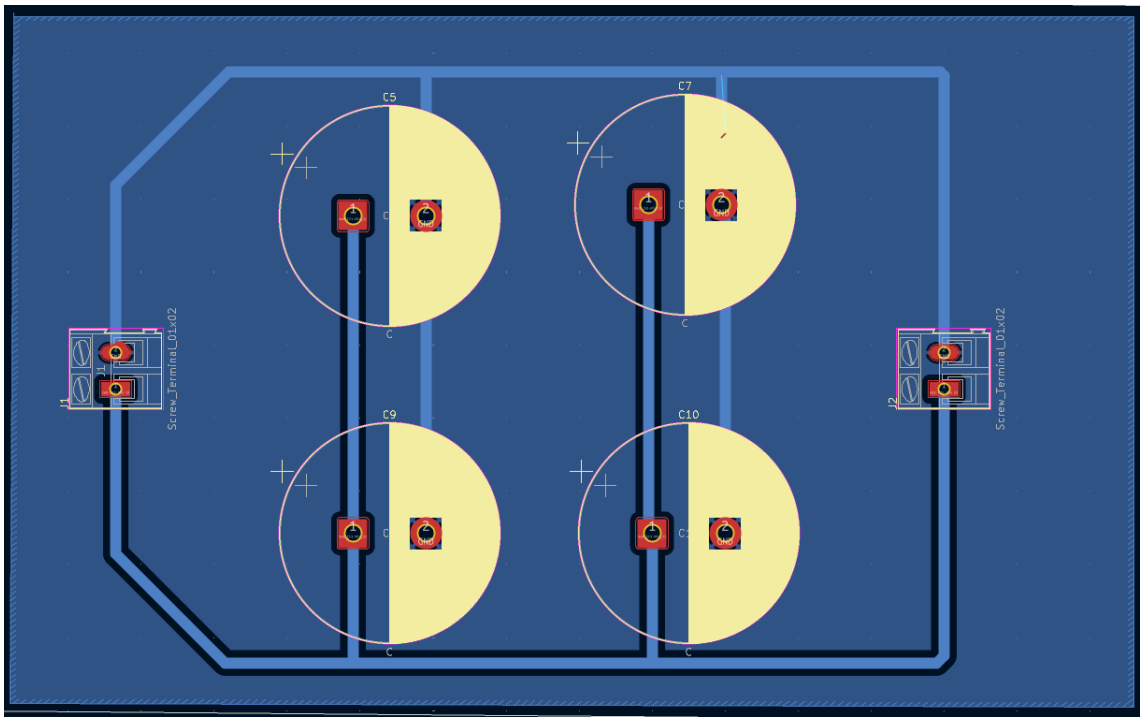


Figura 23: PCB del “Banco de Capacitores”

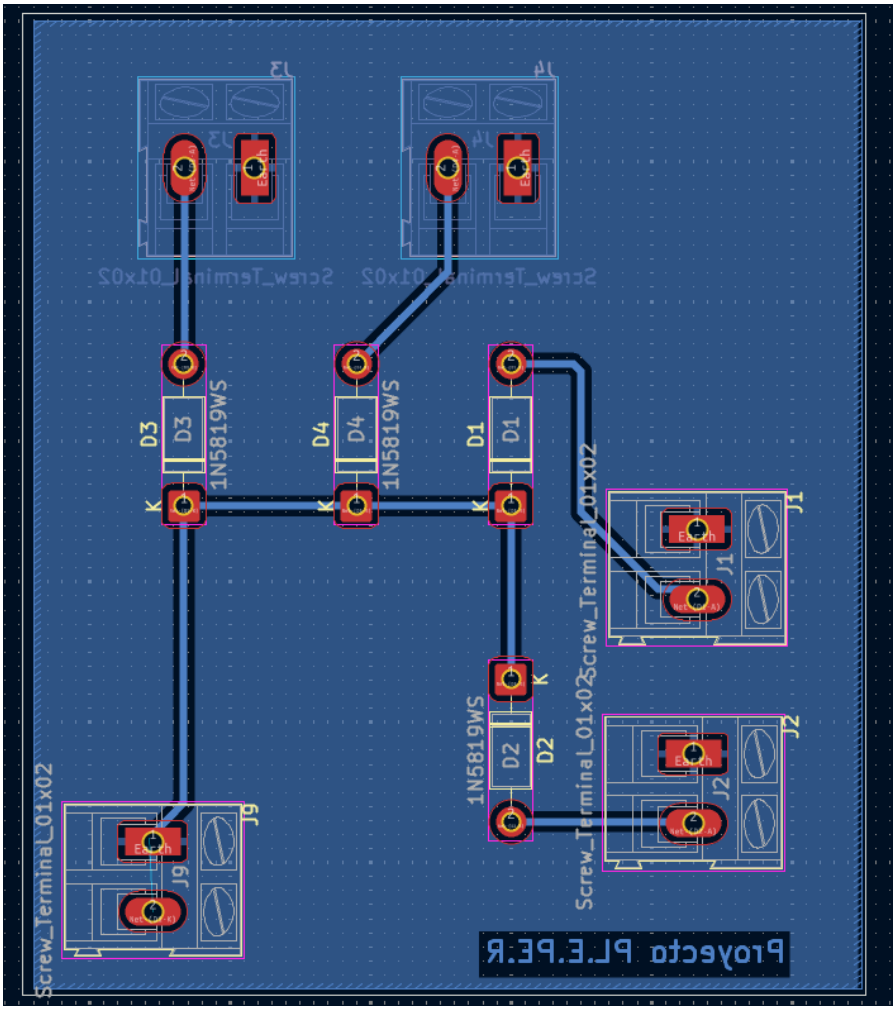


Figura 24: PCB de la “Segunda Placa Rectificadora”

3.4. Modelo 3D de cada PCB

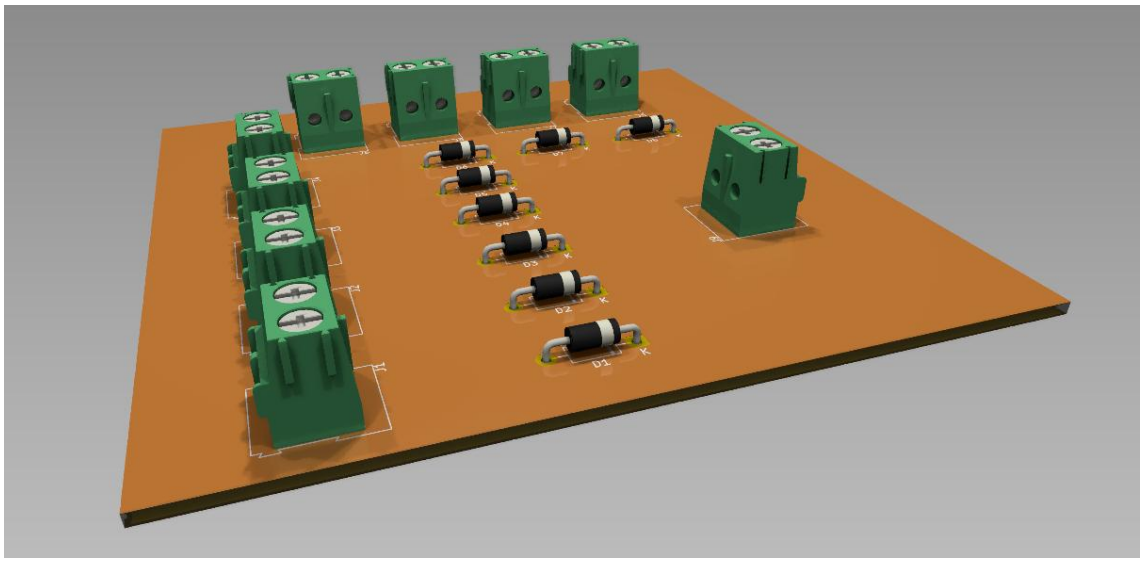


Figura 25: Modelo 3D de la “Placa Rectificadora”

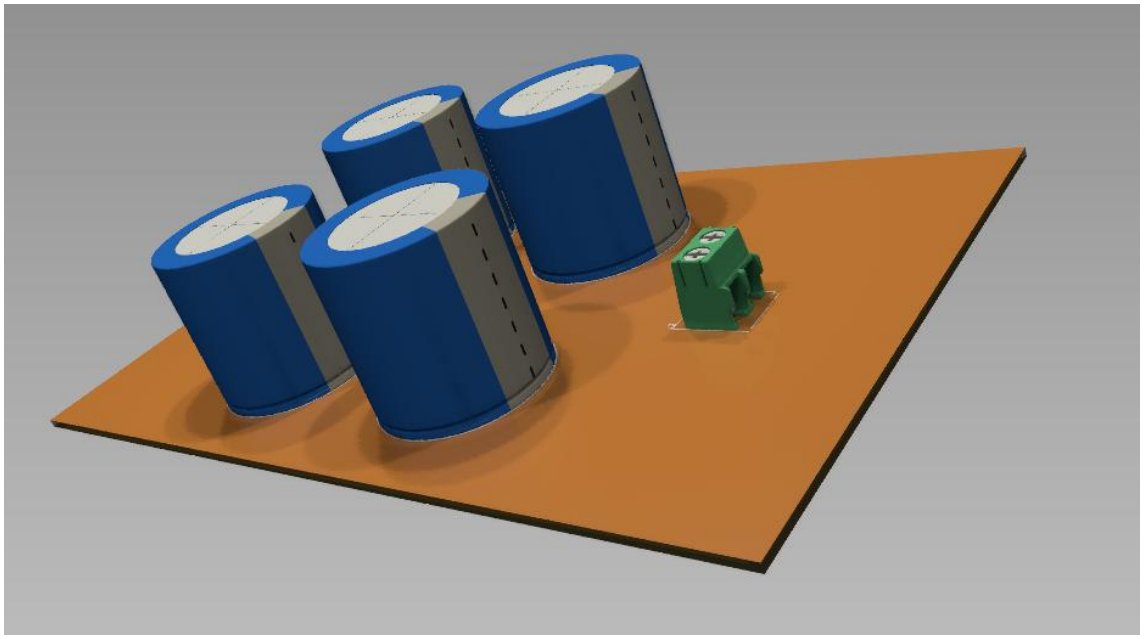


Figura 26: Modelo 3D del “Banco de Capacitores”

4. Hardware

4.1. Circuito del Divisor de Tensión/Raspberry Pi Pico y Conexiones

4.1.1. Objetivo

El circuito diseñado para el proyecto tiene como función detectar los picos de tensión generados por una baldosa piezoeléctrica al ser presionada. Cada vez que alguien pisa la superficie, el material piezoeléctrico produce una diferencia de potencial momentánea. Esta energía eléctrica, aunque breve y de baja corriente, se manifiesta como un pulso de tensión que se puede medir. La idea central es registrar esos picos y mostrarlos en un display LCD en tiempo real, permitiendo observar de manera visual la magnitud de la tensión producida por cada pisada.

Para lograrlo, se utiliza una Raspberry Pi Pico W como unidad principal de medición y procesamiento. La señal de la baldosa pasa primero por una etapa rectificadora que convierte la tensión alterna generada en una señal continua. Luego, esta señal se reduce mediante un divisor de tensión antes de ingresar al convertidor analógico–digital (ADC) del microcontrolador. Finalmente, la Pico procesa el valor obtenido y lo muestra en el display LCD a través de la comunicación I2C.

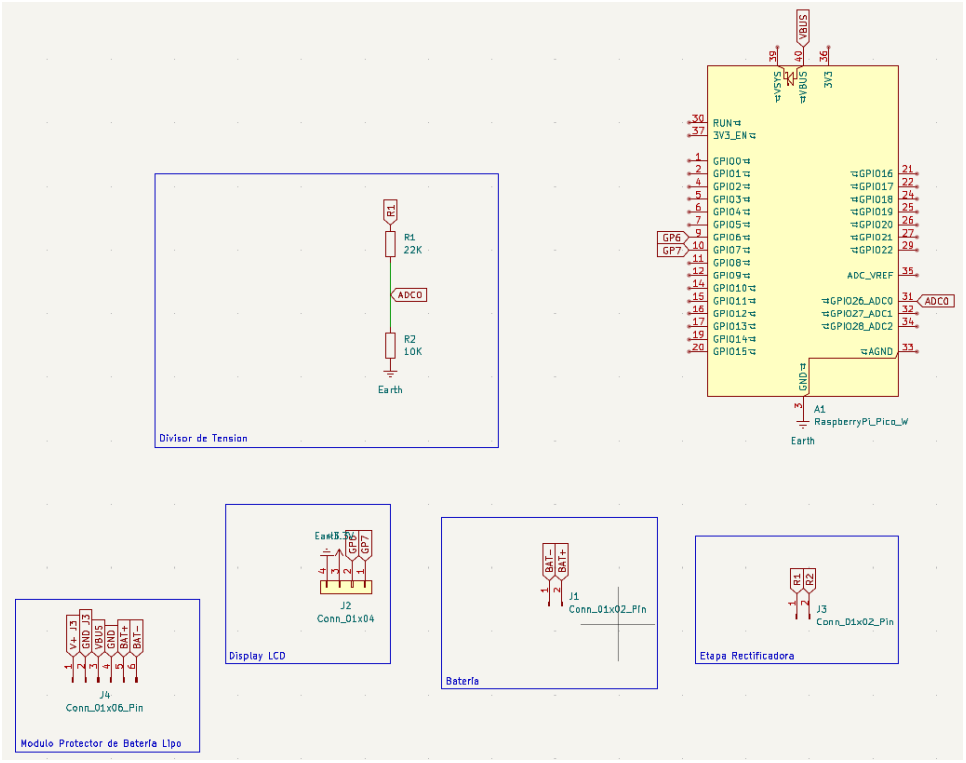


Figura 28: Esquema de componentes para el circuito de la Raspberry Pi Pico

4.2. Etapa Piezoeléctrica y Rectificadora

La baldosa piezoeléctrica actúa como generador eléctrico cuando se le aplica presión. Su estructura interna convierte la energía mecánica del impacto o pisada en energía eléctrica, generando una señal de corriente alterna. Dicha señal es irregular y dura solo el tiempo que la baldosa permanece bajo presión. Para que esta señal pueda ser medida por la Raspberry, primero debe ser rectificada, ya que el ADC solo puede trabajar con tensiones positivas de corriente continua.

La etapa rectificadora, representada en el esquema mediante el conector J3, se encarga de convertir la señal alterna en una señal pulsante continúa utilizando un puente de diodos. De esta forma, se eliminan los semiciclos negativos y se obtiene una salida positiva proporcional a la fuerza ejercida sobre la baldosa. Esta etapa es fundamental porque estabiliza la polaridad de la señal y protege los componentes electrónicos posteriores frente a inversiones de tensión.

Figura 29: Piezoeléctricos Distribuidos sobre la baldosa

4.3. Divisor de Tensión

La señal rectificada que proviene de la baldosa puede alcanzar valores que superan los 3,3 V máximos que el ADC de la Raspberry puede medir de forma segura. Para evitar daños, se utiliza un divisor resistivo formado por dos resistencias en serie, R1 de 22 kΩ y R2 de 10 kΩ. Este circuito reduce la tensión de entrada manteniendo la proporción entre ambas resistencias. El punto intermedio entre ellas se conecta directamente al pin ADC0 del microcontrolador.

Gracias a este divisor, los picos de tensión de la baldosa son escalados a un rango compatible con el ADC, garantizando una medición segura y precisa. Por ejemplo, un pico de 10 V a la entrada del divisor se traduce aproximadamente en 3,1 V en la entrada del ADC. En algunos casos, puede añadirse un pequeño capacitor en paralelo con la resistencia inferior para filtrar ruido y obtener una

lectura más estable. En conjunto, este bloque cumple un rol de protección y adaptación de señal dentro del sistema.

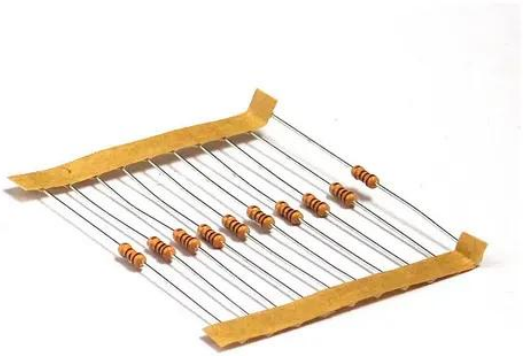


Figura 30: Conjunto de resistencias para formar un divisor de tensión

4.4. Raspberry Pi Pico W

La Raspberry Pi Pico W es el microcontrolador encargado de procesar la información proveniente del divisor de tensión. Su convertidor analógico–digital convierte los pulsos eléctricos en valores numéricos que el programa interpreta en tiempo real. Al detectar un pico de tensión, la Pico calcula su magnitud y la envía inmediatamente al display LCD.

El microcontrolador se comunica con el LCD mediante la interfaz I2C, utilizando los pines GPIO6 y GPIO7 para las líneas de datos (SDA) y reloj (SCL). El pin GPIO26, correspondiente al canal ADC0, se usa para la medición analógica de los pulsos. La Pico también se encarga de la alimentación de la pantalla y de mantener la referencia de 3,3 V para el ADC, garantizando lecturas consistentes. De este modo, la placa centraliza la adquisición, procesamiento y visualización de los datos en un único módulo compacto y de bajo consumo.

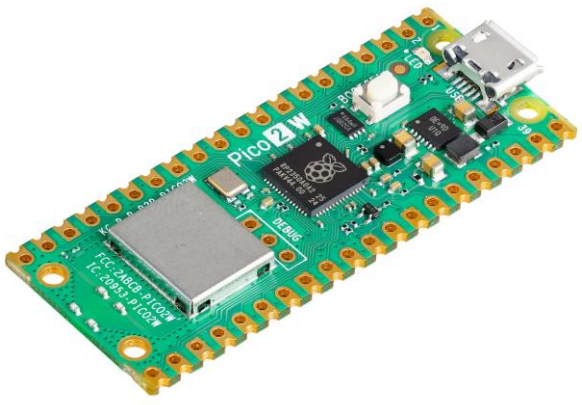


Figura 31: Microcontrolador Raspberry Pi Pico

4.5. Display LCD

El display LCD 16x2 cumple la función de mostrar la tensión detectada en cada pisada. Está conectado a la Raspberry mediante un módulo adaptador I2C, lo que reduce significativamente la cantidad de pines necesarios para la comunicación. Gracias a esta interfaz, el microcontrolador puede enviar comandos y datos con solo dos líneas, lo que simplifica el cableado y mejora la estabilidad del sistema.

Durante el funcionamiento, el display muestra en tiempo real el valor de tensión correspondiente al último pico detectado. Esto permite observar visualmente la respuesta eléctrica de la baldosa piezoeléctrica y comparar la intensidad de las pisadas. Su alimentación puede provenir directamente de la Pico, y el módulo I2C incluido en la parte posterior del LCD se encarga de traducir las señales digitales enviadas por la Raspberry.

Figuras 32 y 33: Display LCD 16x2

4.6. Alimentación y Protección

Para la alimentación del microcontrolador se utilizó un Power Bank de Royalcell, ya que el microcontrolador requería una alimentación constante de 5V de continua para poder funcionar correctamente.

Figura 34: Cargador portatil

5. Software

El programa desarrollado para la Raspberry Pi Pico W tiene como propósito medir los picos de tensión generados por la baldosa piezoeléctrica, procesarlos digitalmente y mostrarlos en tiempo real en un display LCD con comunicación I2C. El código fue escrito en lenguaje C y utiliza las bibliotecas del SDK oficial de Raspberry Pi Pico, junto con una librería propia denominada lcd.c que gestiona las funciones de visualización.

El programa realiza un ciclo continuo de adquisición, conversión y visualización de datos. Cada vez que la baldosa genera un pulso eléctrico, la señal pasa por el divisor resistivo, es convertida a una lectura digital mediante el ADC del microcontrolador y finalmente se muestra en el LCD en forma de valor instantáneo de tensión.

5.1. Configuración de Bibliotecas y Definiciones Iniciales

En la primera sección del código se incluyen las librerías necesarias para el funcionamiento del programa. Se utilizan las cabeceras hardware/gpio.h, hardware/i2c.h y hardware/adc.h para acceder a las funciones de bajo nivel que

controlan los periféricos de la Pico, mientras que pico/stdlib.h brinda funciones generales del sistema. La inclusión de "lcd.h" corresponde a la librería desarrollada para controlar el display LCD mediante el bus I2C.

Luego se definen los pines de conexión: GPIO6 y GPIO7 se asignan como líneas de datos (SDA) y reloj (SCL) del bus I2C, mientras que GPIO26 se establece como la entrada analógica ADC0. También se definen dos constantes fundamentales: ADC_MAX_VAL, que representa el valor máximo que puede entregar el conversor analógico–digital (4095, correspondiente a 12 bits), y VREF, que define la tensión de referencia del ADC (3.3 V).

A continuación se declaran las variables del divisor de tensión (R1 y R2), cuyos valores se usan para convertir la lectura del ADC en un voltaje real. Estas variables son configurables, lo que permite adaptar el código a diferentes divisores sin modificar la lógica principal.

5.2. Función calcular_voltaje_real()

Esta función cumple el papel de adquirir el valor analógico desde el pin ADC0 y convertirlo en una tensión equivalente al valor real presente en la entrada del divisor resistivo. Primero, mediante adc_select_input(0) se selecciona el canal ADC0 de la Pico. Luego se ejecuta adc_read(), que devuelve un valor entero entre 0 y 4095 proporcional a la tensión medida.

El siguiente paso consiste en convertir ese número a un voltaje, aplicando la relación entre el valor digital y la referencia de 3.3 V. Finalmente, el resultado se corrige considerando la relación del divisor de tensión, multiplicando por el factor $(R1 + R2)/R2$. De esta manera se obtiene el voltaje real generado por la baldosa piezoeléctrica, expresado en volts. Esta función encapsula todo el proceso de medición, simplificando el cuerpo principal del programa.

5.3. Función Principal *main()*

La función main() es el núcleo del programa y se encarga de inicializar los periféricos, configurar los módulos de hardware y ejecutar el ciclo de medición continua. Primero se invoca stdio_init_all() para iniciar las funciones básicas del sistema. Luego se configura el bus I2C: se inicializa el módulo i2c1 con una frecuencia de 100 kHz y se asignan los pines GPIO6 y GPIO7 a la función I2C. También se activan resistencias de pull-up internas en ambas líneas para garantizar la correcta comunicación con el módulo LCD.

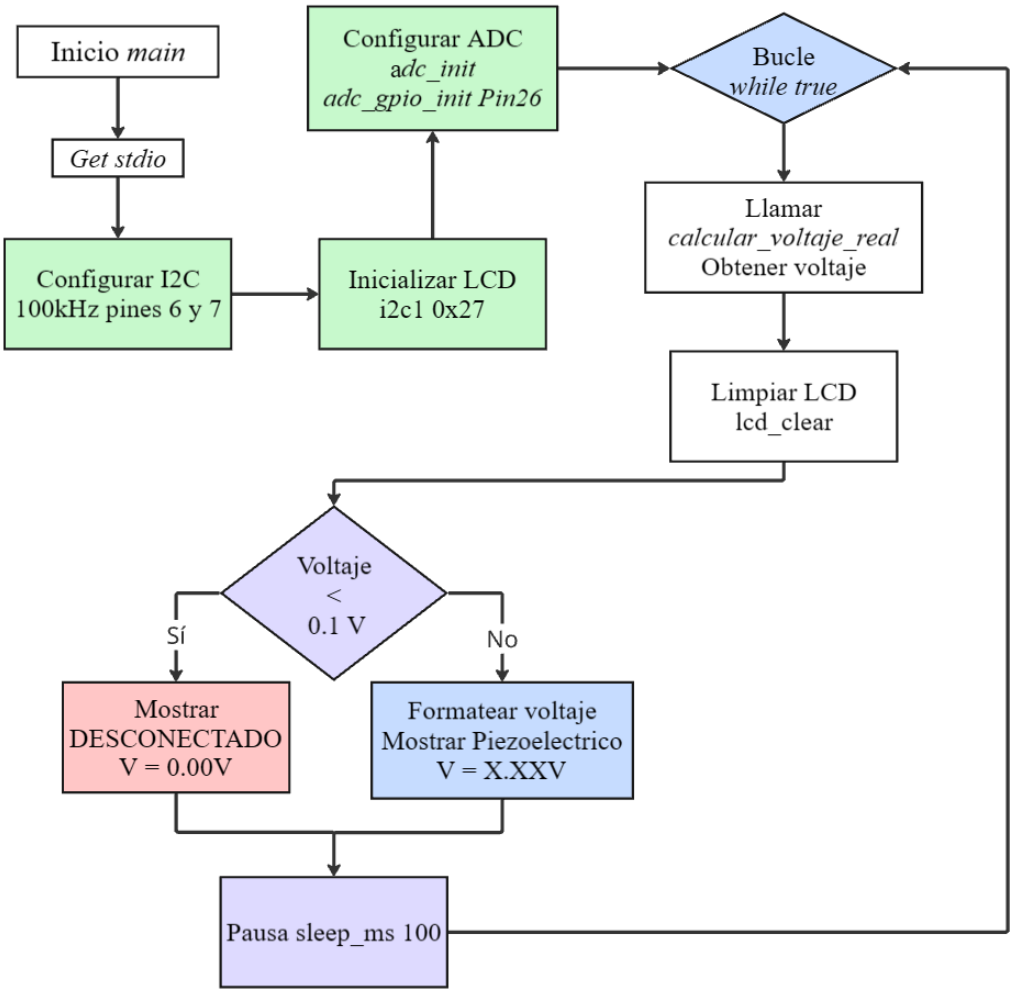


Figura 35: Diagrama en bloques de la función main.c

Una vez configurado el bus, se llama a lcd_init(i2c1, 0x27) para inicializar el display en la dirección I2C 0x27, que es la más común en los módulos basados en PCF8574. Paralelamente, se configura el sistema de adquisición analógica: se inicializa el ADC y se habilita el pin GPIO26 como entrada analógica mediante adc_gpio_init(ADC_PIN).

Con la inicialización completa, el programa entra en un bucle while(true) que se ejecuta indefinidamente. Dentro de este bucle, se llama a calcular_voltaje_real() para obtener la lectura actual del sensor piezoeléctrico.

Luego se limpia la pantalla con lcd_clear() y se actualiza el contenido del display según el valor leído. Si la tensión medida es menor a 0.1 V, el programa interpreta que la baldosa no está siendo presionada o que el sensor está desconectado, mostrando en pantalla el mensaje “DESCONECTADO” junto con “V = 0.00V”. En caso contrario, se formatea el valor medido en una cadena de texto con dos decimales mediante snprintf(), y se imprime en la segunda línea del LCD precedido por el texto “Piezoelectrico:”.

Finalmente, se incluye una pausa de 100 milisegundos con sleep_ms(100) antes de repetir el ciclo, evitando parpadeos excesivos en la pantalla y permitiendo una actualización fluida y legible de los valores.

5.4. Librerías lcd.c y lcd.h

El conjunto de archivos lcd.c y lcd.h conforma una librería auxiliar encargada de manejar toda la comunicación con el display LCD mediante el bus I2C. En su implementación se encuentran funciones para enviar comandos, escribir caracteres o cadenas de texto y posicionar el cursor en pantalla.

La inicialización del display se realiza en modo de 4 bits, lo que permite enviar datos en dos mitades (nibbles) utilizando menos líneas. Además, se definen funciones de bajo nivel como i2c_write_byte() y lcd_toggle_enable() que garantizan la sincronización correcta entre la Pico y el controlador del display (habitualmente un PCF8574).

Durante la inicialización, también se cargan en la memoria del LCD caracteres personalizados, aunque en esta versión del código no se utilizan, ya que el objetivo principal es mostrar valores numéricos en formato de texto. La separación de esta librería respecto al archivo principal mejora la legibilidad del código y permite reutilizarla en otros proyectos sin modificaciones.

5.5. Archivo CMakeLists.txt

El archivo CMakeLists.txt actúa como el configurador del entorno de compilación. En él se indican los parámetros necesarios para construir el proyecto con el SDK de Raspberry Pi Pico. Se especifica el estándar del lenguaje, la versión del SDK, la placa objetivo (pico2_w) y las bibliotecas que deben vincularse al programa.

También se incluyen las dependencias de hardware necesarias (hardware_i2c y hardware_adc), y se agrega el subdirectorio de la librería LCD. Este archivo es interpretado por CMake para generar los scripts de compilación y enlazado que permiten crear el archivo ejecutable final que se carga en la placa.

5.6. Explicación del código por línea

5.6.1. pleper.c

```
#include "hardware/gpio.h"
```

Incluye el encabezado del SDK para controlar GPIO (configurar pines, funciones de pin, etc.).

```
#include "hardware/i2c.h"
```

Incluye el módulo de I2C del SDK (para inicializar el bus, fijar velocidad y mandar bytes).

```
#include "hardware/adc.h"
```

Incluye el módulo de ADC (convertidor analógico–digital) del RP2040.

```
#include "lcd.h"
```

Incluye la librería propia para manejar el LCD I2C (prototipos de funciones como lcd_init, lcd_string, etc.).

```
#include <stdio.h>
```

Incluye funciones estándar de C como snprintf para formatear cadenas.

```
#include "pico/stdlib.h"
```

Incluye utilidades generales del SDK de Pico (temporización como sleep_ms, inicializaciones, etc.).

```
#define GPIO_SDA 6
```

Define el pin GPIO6 para la línea SDA del bus I2C.

```
#define GPIO_SCL 7
```

Define el pin GPIO7 para la línea SCL del bus I2C.

```
#define ADC_PIN 26 // ADC0
```

Define GPIO26 como pin de entrada analógica; corresponde a ADC0.

```
#define ADC_MAX_VAL 4095.0f
```

Constante con el valor máximo del ADC (12 bits → 0..4095) en punto flotante.

```
#define VREF 3.3f
```

Define la tensión de referencia del ADC (3.3 V) usada para convertir conteos a voltios.

```
float R1 = 30000.0f; // ohms (arriba)
```

Variable con el valor de la resistencia superior del divisor (R1). Se usa para deshacer la atenuación.

```
float R2 = 10000.0f; // ohms (abajo)
```

Variable con el valor de la resistencia inferior del divisor (R2).

```
float calcular_voltaje_real() {
```

Declara la función que lee el ADC y devuelve el voltaje real antes del divisor.

```
adc_select_input(0);
```

Selecciona el canal ADC0 (GPIO26). Asegura que las lecturas provengan del canal correcto.

```
uint16_t raw = adc_read();
```

Ejecuta una conversión ADC y guarda el conteo (0..4095).

```
float v_adc = (raw / ADC_MAX_VAL) * VREF;
```

Convierte el conteo a voltios en el pin ADC: proporción del máximo por la referencia.

```
float v_real = v_adc * ((R1 + R2) / R2);
```

Deshace el divisor: multiplica por el factor ((R1+R2)/R2) para obtener el voltaje real en la salida del rectificador.

```
return v_real;
```

Devuelve el voltaje real (en V).

```
}
```

Cierre de la función calcular_voltaje_real.

```
int main() {
```

Punto de entrada del programa.

```
stdio_init_all();
```

Inicializa el subsystem stdio (útil si en el futuro se usa USB/UART para debug).

```
i2c_init(i2c1, 100000);
```

Inicializa el controlador I2C1 a 100 kHz (velocidad estándar).

```
gpio_set_function(GPIO_SDA, GPIO_FUNC_I2C);
```

Configura GPIO6 en función I2C para SDA.

```
gpio_set_function(GPIO_SCL, GPIO_FUNC_I2C);
```

Configura GPIO7 en función I2C para SCL.

```
gpio_pull_up(GPIO_SDA);
```

Activa pull-up interno en SDA (de todos modos suelen requerirse pull-ups externos en I2C).

```
gpio_pull_up(GPIO_SCL);
```

Activa pull-up interno en SCL.

```
lcd_init(i2c1, 0x27);
```

Inicializa el LCD usando el bus i2c1 y la dirección 0x27 (común en PCF8574).

```
adc_init();
```

Inicializa el ADC del RP2040.

```
adc_gpio_init(ADC_PIN);
```

Habilita el GPIO26 como entrada analógica para el ADC.

```
while (true) {
```

Comienza el bucle infinito de adquisición y visualización.

```
float v = calcular_voltaje_real();
```

Obtiene el voltaje real actual (después de compensar el divisor).

```
lcd_clear();
```

Limpia el LCD antes de escribir la nueva lectura (garantiza que no queden caracteres viejos).

```
if (v < 0.10f) {
```

Comprueba si la tensión es muy baja (umbral de 0.10 V); se interpreta como sin pisada o desconectado.

```
lcd_set_cursor(0, 0);
```

Posiciona el cursor en línea 0, columna 0.

```
lcd_string("DESCONECTADO");
```

Muestra el estado “DESCONECTADO” (sin pulso válido).

```
lcd_set_cursor(1, 0);
```

Posiciona el cursor en línea 1, columna 0.

```
lcd_string("V = 0.00V");
```

Escribe una lectura fija de 0.00 V para claridad visual.

```
}
```

Fin del bloque if.

```
else {
```

Caso contrario: hay tensión válida a mostrar.

```
char buffer[16];
```

Reserva un buffer para formatear el texto de la segunda línea.

```
lcd_set_cursor(0, 0);
```

Posiciona cursor en línea 0, columna 0.

```
lcd_string("Piezoelectrico:");
```

Escribe el título de la medición en la primera línea.

```
snprintf(buffer, sizeof(buffer), "V= %.2fV  ", v);
```

Formatea el valor en volts con 2 decimales. Los espacios extra al final barren residuos en la línea.

```
lcd_set_cursor(1, 0);
```

Posiciona el cursor en línea 1, columna 0.

```
lcd_string(buffer);
```

Escribe el valor de tensión recién formateado.

```
    }
```

Cierra el else.

```
sleep_ms(100);
```

Pausa 100 ms para una tasa de actualización ≈ 10 Hz y evitar parpadeos.

```
  }
```

Cierra el bucle infinito.

```
}
```

Fin de main.

```c
#include "hardware/gpio.h"
#include "hardware/i2c.h"
#include "hardware/adc.h"
#include "lcd.h"
#include <stdio.h>
#include "pico/stdlib.h"

#define GPIO_SDA 6
#define GPIO_SCL 7
#define ADC_PIN 26 // ADC0

#define ADC_MAX_VAL 4095.0f
#define VREF 3.3f

// Ajustá estos valores cuando definamos el divisor
float R1 = 30000.0f; // ohms (arriba)
float R2 = 10000.0f; // ohms (abajo)

float calcular_voltaje_real() {
    adc_select_input(0);
    uint16_t raw = adc_read();

    float v_adc = (raw / ADC_MAX_VAL) * VREF;
    float v_real = v_adc * ((R1 + R2) / R2);

    return v_real;
}

int main() {
    stdio_init_all();

    // LCD I2C
    i2c_init(i2c1, 100000);
    gpio_set_function(GPIO_SDA, GPIO_FUNC_I2C);
    gpio_set_function(GPIO_SCL, GPIO_FUNC_I2C);
    gpio_pull_up(GPIO_SDA);
    gpio_pull_up(GPIO_SCL);
```

Figura 36: Captura de código de pleper.c

5.6.2. CMakeLists.txt

 # Generated Cmake Pico project file

Comentario informativo generado por la plantilla/IDE.

cmake_minimum_required(VERSION 3.13)

Indica la versión mínima de CMake requerida para procesar este proyecto.

set(CMAKE_C_STANDARD 11)

Fija el estándar de C a C11.

set(CMAKE_CXX_STANDARD 17)

Fija el estándar de C++ a C++17 (aunque no se use C++ en este proyecto, queda definido).

set(CMAKE_EXPORT_COMPILE_COMMANDS ON)

Genera compile_commands.json (útil para herramientas de análisis/IDE).

if(WIN32)

set(USERHOME $ENV{USERPROFILE})

```
else()
  set(USERHOME $ENV{HOME})
endif()
```

Detecta el home del usuario de forma portable (Windows vs. Unix-like).

```
set(sdkVersion 2.2.0)
set(toolchainVersion 14_2_Rel1)
set(picotoolVersion 2.2.0-a4)
```

Variables usadas por la extensión de VS Code para sincronizar versiones.

```
set(picoVscode ${USERHOME}/.pico-sdk/cmake/pico-vscode.cmake)
if (EXISTS ${picoVscode})
  include(${picoVscode})
endif()
```

Incluye utilidades de pico-vscode si están presentes (integración IDE).

```
set(PICO_BOARD pico2_w CACHE STRING "Board type")
```

Define la placa objetivo como pico2_w y lo guarda en caché CMake.

```
include(pico_sdk_import.cmake)
```

Importa el SDK de Pico (ruta establecida por variables/entorno).

```
project(pleper C CXX ASM)
```

Declara el proyecto pleper y los lenguajes usados (C, C++, ASM).

```
pico_sdk_init()
```

Inicializa variables y rutas del Pico SDK para el build.

```
add_executable(pleper pleper.c )
```

Crea el ejecutable pleper con la fuente principal pleper.c.

```
pico_set_program_name(pleper "pleper")
```

Asigna el nombre de programa (metadatos) al binario.

```
pico_set_program_version(pleper "0.1")
```

Define la versión del programa.

```
pico_enable_stdio_uart(pleper 0)
```

Desactiva la salida UART para stdio (0 = off).

```
pico_enable_stdio_usb(pleper 0)
```

Desactiva la salida USB para stdio. (Se usa solo LCD en este proyecto.)

```
target_link_libraries(pleper
    pico_stdlib)
```

Enlaza contra la librería estándar del Pico SDK.

```
target_include_directories(pleper PRIVATE
    ${CMAKE_CURRENT_LIST_DIR}
)
```

Agrega el directorio actual a los includes del compilador para este target.

```
target_link_libraries(pleper
    hardware_i2c
    hardware_adc
    )
```

Enlaza las librerías de hardware I2C y hardware ADC requeridas.

```
pico_add_extra_outputs(pleper)
```

Genera artefactos extra: UF2, ELF, bin, mapa, etc.

```
add_subdirectory(lcd)
```

Incluye el subproyecto de la librería LCD ubicado en el directorio lcd/.

```
target_link_libraries(${PROJECT_NAME} lcd)
```

Enlaza el ejecutable principal con la librería lcd creada en lcd/.

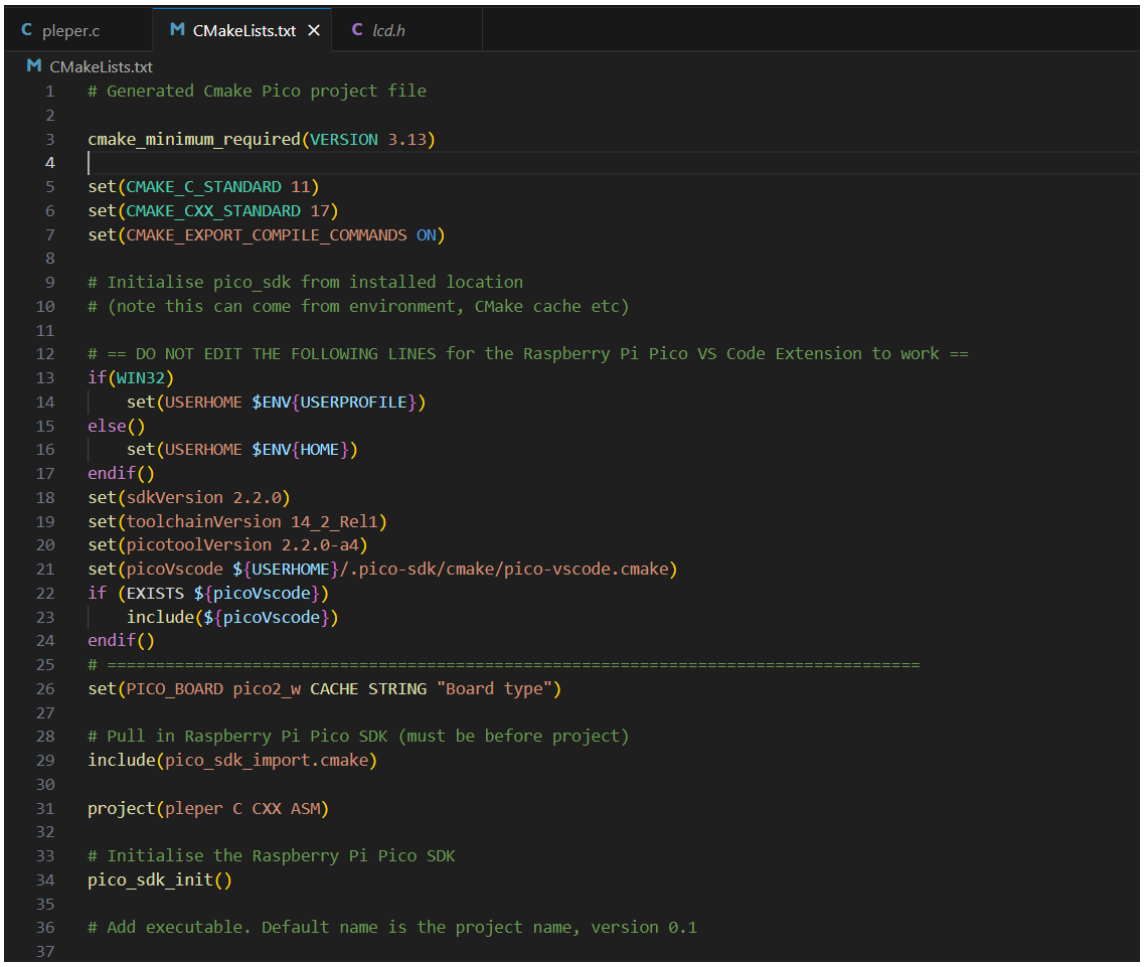


Figura 37: Captura de código de CMakeLists.txt

5.6.3. lcd.c

#include "lcd.h"

Incluye los prototipos y constantes del driver LCD.

*static i2c_inst_t *lcd_i2c;*

Puntero estático al controlador I2C que usará el LCD (i2c0 o i2c1).

static uint8_t addr;

Guarda la dirección I2C de 7 bits del módulo LCD (p.ej., 0x27).

static uint8_t barra_completa[8] =
{0x1F,0x1F,0x1F,0x1F,0x1F,0x1F,0x1F,0x1F};

static uint8_t barra_vacia[8] =
{0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00};

Define dos caracteres personalizados (CGRAM) para barra llena/vacía (no se usan en esta versión, pero quedan disponibles).

```
void lcd_create_char(uint8_t location, uint8_t charmap[]) {
  location &= 0x7;
  lcd_send_byte(LCD_SETCGRAMADDR | (location << 3), LCD_COMMAND);
  for (int i = 0; i < 8; i++) {
    lcd_send_byte(charmap[i], LCD_CHARACTER);
  }
}
```

Crea un carácter personalizado en la CGRAM del LCD; location 0..7. Envía dirección base y los 8 bytes de la matriz de puntos.

```
void i2c_write_byte(uint8_t val) {
  i2c_write_blocking(lcd_i2c, addr, &val, 1, false);
}
```

Envía un byte por I2C (bloqueante) a la dirección del LCD/expansor.

```
void lcd_toggle_enable(uint8_t val) {
#define DELAY_US 600
  sleep_us(DELAY_US);
  i2c_write_byte(val | LCD_ENABLE_BIT);
  sleep_us(DELAY_US);
  i2c_write_byte(val & ~LCD_ENABLE_BIT);
  sleep_us(DELAY_US);
}
```

Genera el pulso de Enable (E) requerido por el LCD paralelo. Se agrega delay para cumplir tiempos mínimos.

```
void lcd_send_byte(uint8_t val, int mode) {
  uint8_t high = mode | (val & 0xF0) | LCD_BACKLIGHT;
  uint8_t low = mode | ((val << 4) & 0xF0) | LCD_BACKLIGHT;
  i2c_write_byte(high);
  lcd_toggle_enable(high);
```

```
  i2c_write_byte(low);
  lcd_toggle_enable(low);
}
```

Envía un byte al LCD en 4 bits (alto y bajo). mode selecciona comando o dato. Mantiene el backlight encendido y conmuta E en cada nibble.

```
void lcd_clear(void) {
  lcd_send_byte(LCD_CLEARDISPLAY, LCD_COMMAND);
}
```

Manda el comando de limpiar la DDRAM y volver el cursor al inicio.

```
void lcd_set_cursor(int line, int position) {
  int line_offsets[] = { 0x00, 0x40, 0x14, 0x54 };
  int val = 0x80 + line_offsets[line] + position;
  lcd_send_byte(val, LCD_COMMAND);
}
```

Posiciona el cursor en la línea (0..3 según mapeo interno) y columna indicada; el valor final se envía como comando SETDDRAMADDR.

```
void lcd_char(char val) {
  lcd_send_byte(val, LCD_CHARACTER);
}
```

Escribe un carácter (dato) en la posición actual.

```
void lcd_string(const char *s) {
  while (*s) {
    lcd_char(*s++);
  }
}
```

Escribe una cadena completa carácter por carácter.

```
void lcd_init(i2c_inst_t *i2c, uint8_t address) {
  lcd_i2c = i2c;
```

```
    addr = address;

    lcd_send_byte(0x03, LCD_COMMAND);

    lcd_send_byte(0x03, LCD_COMMAND);

    lcd_send_byte(0x03, LCD_COMMAND);

    lcd_send_byte(0x02, LCD_COMMAND);

    lcd_send_byte(LCD_ENTRYMODESET | LCD_ENTRYLEFT,
LCD_COMMAND);

    lcd_send_byte(LCD_FUNCTIONSET | LCD_2LINE, LCD_COMMAND);

    lcd_send_byte(LCD_DISPLAYCONTROL | LCD_DISPLAYON,
LCD_COMMAND);

    lcd_create_char(0, barra_completa);

    lcd_create_char(1, barra_vacia);

    lcd_clear();

}
```

Inicializa el modo 4 bits del LCD (secuencia estándar 0x03 x3 y 0x02). Configura dirección de entrada, número de líneas, display ON, y carga caracteres personalizados 0 y 1. Limpia la pantalla al final.

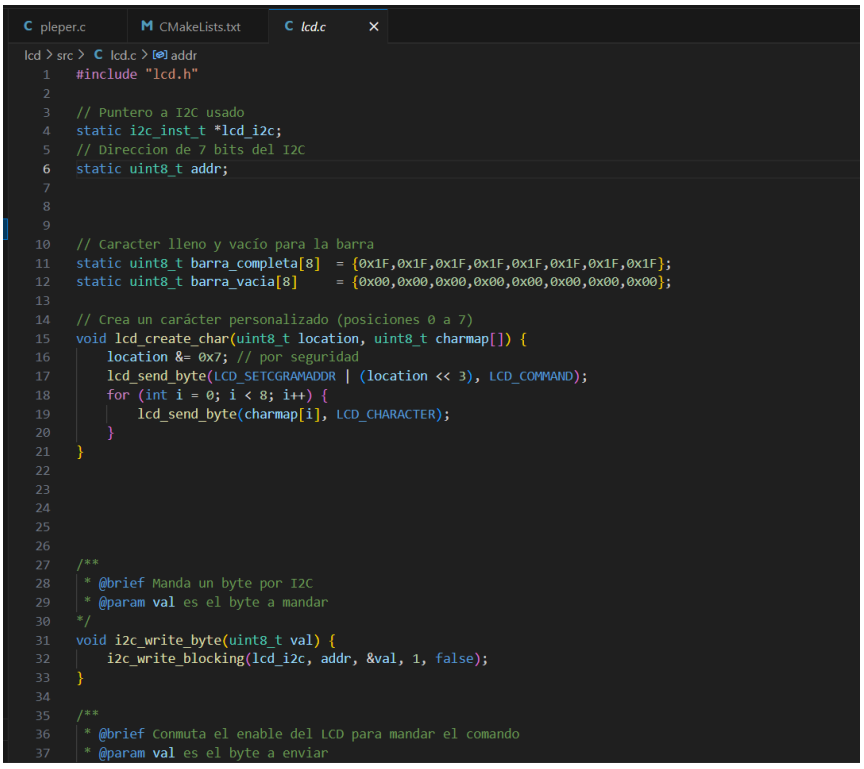


Figura 38: Captura de código de lcd.c

5.6.4. lcd.h

```
#include <stdio.h>

#include "pico/stdlib.h"

#include "hardware/i2c.h"
```

Incluye dependencias estándar, temporización del Pico y tipos/funciones de I2C necesarios por el header.

```
#define LCD_CLEARDISPLAY 0x01

#define LCD_RETURNHOME 0x02

#define LCD_ENTRYMODESET 0x04

#define LCD_DISPLAYCONTROL 0x08

#define LCD_CURSORSHIFT 0x10

#define LCD_FUNCTIONSET 0x20

#define LCD_SETCGRAMADDR 0x40

#define LCD_SETDDRAMADDR 0x80
```

Constantes de comandos base del controlador de LCD compatibles con HD44780.

```
#define LCD_ENTRYSHIFTINCREMENT 0x01

#define LCD_ENTRYLEFT 0x02
```

Flags del modo de entrada: desplazamiento automático e ingreso izquierda-a-derecha.

```
#define LCD_BLINKON 0x01

#define LCD_CURSORON 0x02

#define LCD_DISPLAYON 0x04
```

Flags de control de display: parpadeo, cursor visible y encendido del display.

```
#define LCD_MOVERIGHT 0x04

#define LCD_DISPLAYMOVE 0x08
```

Flags para desplazar el display/cursor a derecha.

```
#define LCD_5x10DOTS 0x04
```

```
#define LCD_2LINE 0x08
```

```
#define LCD_8BITMODE 0x10
```

Flags de seteo de función: tamaño de la matriz de puntos, líneas y modo 8 bits (aquí operamos en 4 bits, por eso no se usa ese flag).

```
#define LCD_BACKLIGHT 0x08
```

Bit que controla el backlight a través del expansor I2C (PCF8574 mapea ese bit a la línea de luz).

```
#define LCD_ENABLE_BIT 0x04
```

Bit que se usa para conmutar E (Enable) del LCD.

```
#define LCD_CHARACTER  1
```

```
#define LCD_COMMAND    0
```

Constantes para indicar a lcd_send_byte si enviamos dato (carácter) o comando.

```
#define MAX_LINES      2
```

```
#define MAX_CHARS      16
```

Dimensiones lógicas del LCD 16x2 usadas como referencia.

```
void i2c_write_byte(uint8_t val);
```

```
void lcd_toggle_enable(uint8_t val);
```

```
void lcd_send_byte(uint8_t val, int mode);
```

```
void lcd_clear(void);
```

```
void lcd_set_cursor(int line, int position);
```

```
void lcd_char(char val);
```

```
void lcd_string(const char *s);
```

```
void lcd_init(i2c_inst_t *i2c, uint8_t address);
```

Prototipos de todas las funciones públicas del driver LCD.

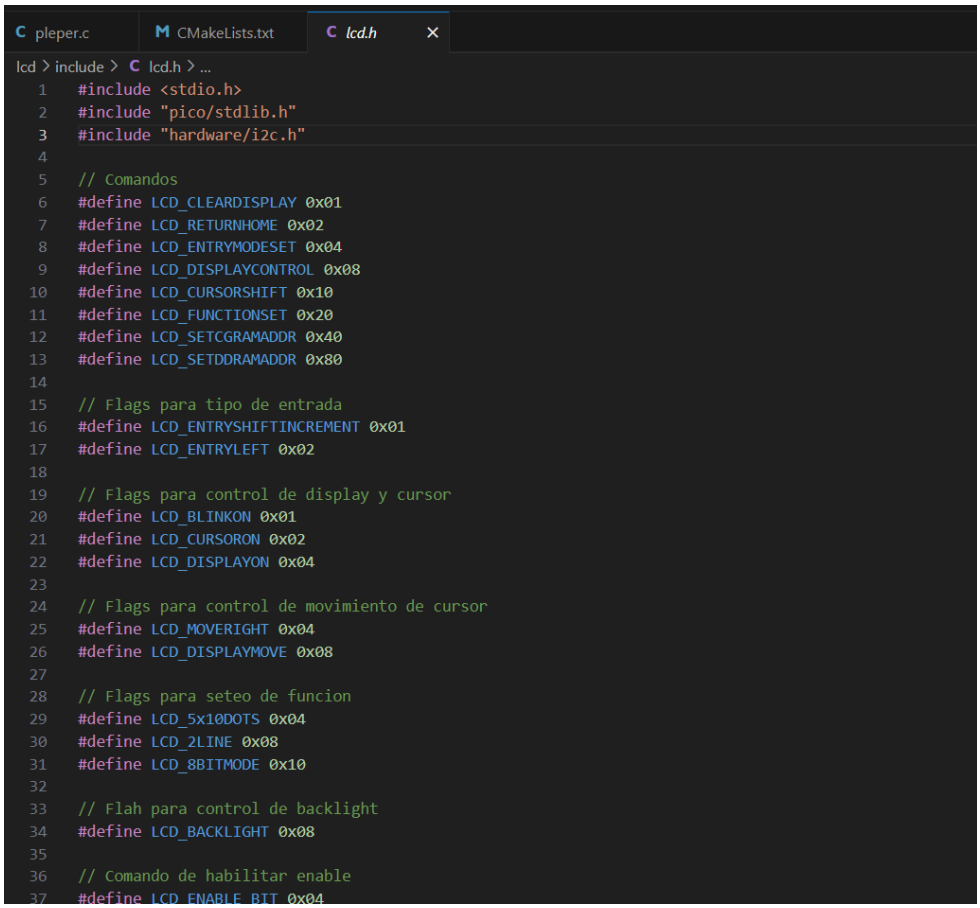


Figura 39: Captura de código de lcd.h

6. Anexo

6.1. Biblografía

Para el desarrollo de nuestro proyecto se han considerado las siguientes referencias:

- Análisis comparativo del nivel de energía eléctrica producida por baldosas piezoeléctricas en función de su forma geométrica
- Aplicaciones del efecto piezoeléctrico para la generación de energía
- Articulo cientifico Proyecto de grado piezoelectricos en perfil vial piloto en Bogotá
- Baldosa Piezoeléctrica para alimentar sistemas de iluminación de bajo consumo energético
- Desarrollo de un prototipo de baldosa generadora de energía eléctrica a partir de la piezoelectricidad y almacenamiento de la energía producida

- Diseño de un prototipo para generación energética mediante tecnología piezoeléctrica
- Diseño e Implementación de un Prototipo Portable de Generación Piezoeléctrica
- Diseño en implementación de un generador piezoeléctrico baldosa, para alimentar un sistema de iluminación de baja potencia
- FOOTSTEP POWER GENERATION USING PIEZOELECTRIC PLATE
- Generación de Energía Eléctrica Limpia mediante Baldosas Piezoeléctricas
- Materiales Piezoeléctricos Calculo y Simulación del Circuito Equivalente de un Cristal de Cuarzo
- Piezoelectric Effect Smart roads in green energy
- Piezoelectric Footstep Power Generation
- Proyecto de Diseño de un Sistema de Baldosas Piezoeléctricas en un Gran Centro Comercial
- Transductores Piezoeléctricos aplicados a la Generación de Energía (ESTUDIO EXPERIMENTAL)