PROYECTO FINAL- ELECTRÓNICA DIGITAL I

Presentado por:

Juan Pablo López Bolívar - *julopezbo@unal.edu.co*

Laura Valentina López Vergara - *laulopezve@unal.edu.co*

Daniel Castillo Silva - *dcastillosi@unal.edu.co*

Profesor:

Johan Sebastian Eslava

jseslavag@unal.edu.co

Jueves 6 de Marzo

Universidad Nacional de Colombia

Facultad de Ingeniería

2024-2S

CONTENIDO

1. TÍTULO

Generación de Energía y Monitoreo en Tiempo Real mediante Pisos Piezoeléctricos Implementados en una FPGA.

2. INTRODUCCIÓN

El crecimiento poblacional, el desarrollo tecnológico y las crecientes demandas energéticas han impulsado la necesidad de explorar fuentes de energía sostenibles que complementen o reemplacen las tradicionales [1], [2]. Las limitaciones de los combustibles fósiles y su impacto ambiental han motivado la búsqueda de soluciones innovadoras que permitan una generación energética más limpia y eficiente [3], [4]. En este contexto, la piezoelectricidad surge como una alternativa prometedora, al permitir la conversión de energía mecánica, generada por el movimiento humano, en energía eléctrica [5].

Este proyecto se centra en el diseño e implementación de un sistema de generación de energía basado en sensores piezoeléctricos, instalados en superficies de tránsito peatonal. La energía generada será almacenada temporalmente en condensadores para estabilizar el voltaje, el cual luego será enviado a un convertidor analógico-digital (ADC) capacitivo ADS1015. Este ADC se encargará de transformar los voltajes en valores binarios [6], los cuales serán procesados por una FPGA para su análisis y gestión. Posteriormente, los datos recopilados sobre la cantidad de pasos y la energía producida se transmitirán en tiempo real a una pantalla mediante un módulo Wi-Fi, facilitando su monitoreo.

Esta iniciativa no solo busca optimizar la recolección de energía cinética desaprovechada, sino también demostrar el potencial de la integración de tecnologías digitales en aplicaciones de energía renovable.

2. OBJETIVOS:

Objetivo General:

Desarrollar un **prototipo funcional** de un sistema de captación de energía mediante sensores piezoeléctricos, capaz de medir en **tiempo real** la cantidad de energía generada por el paso de las personas y el voltaje almacenado en condensadores que simulan una batería, con el fin de demostrar su funcionamiento en espacios de alto tránsito.

Objetivos Específicos:

1. **Implementar un sistema de captación de energía** basado en un arreglo de sensores piezoeléctricos conectados en serie y paralelo, optimizando su eficiencia de conversión energética.

2. **Diseñar y programar un sistema de adquisición de datos** en una FPGA BlackIce 40, encargado de leer los valores de voltaje a través de un ADC capacitivo ADS1015 mediante comunicación I2C y UART.
3. **Desarrollar una máquina de estados** en Verilog para gestionar la adquisición y transmisión de datos de los sensores hacia un ESP32-S2 Mini a través del protocolo SPI.
4. **Configurar la transmisión de datos en tiempo real** desde el ESP32 hacia Arduino Cloud, permitiendo la visualización remota del voltaje y la energía generada.
5. **Optimizar la interfaz de visualización** para proporcionar una representación clara y accesible de los datos captados, facilitando su interpretación por posibles usuarios o clientes.

3. ARQUITECTURA

1. Parte análoga:

- **Diagrama de bloques:**

A continuación, se presenta la representación gráfica del diagrama de bloques:

Figura 1. Diagrama de bloques.

- **Diseño e implementación del circuito:**

Se optó por implementar un arreglo combinado en serie y paralelo de los piezoeléctricos, con el objetivo de obtener simultáneamente valores significativos de voltaje y corriente, optimizando así la eficiencia en la generación de energía.

Para ello, la implementación de los siguientes materiales fue fundamental:

- Catorce piezoeléctricos
- Una lámina de acrílico
- Cables
- Estaño
- Silicona
- Tabla de madera
- Condensadores (10 uF - 50V)
- Diodos 1N4148

Inicialmente, se realizó la distribución de los piezoeléctricos sobre una tabla de madera con el objetivo de garantizar que la presión generada por cada pisada se distribuyera de manera

uniforme entre ellos. Para asegurar este aspecto, se utilizó una lámina de acrílico junto con gomas de silicona adhesiva, lo que permitió optimizar la transmisión de la fuerza sobre cada dispositivo.

La salida generada por los piezoeléctricos se configuró en paralelo, permitiendo su uso como fuente de entrada en el arreglo circuital. Este arreglo está diseñado para almacenar el voltaje y la energía generada mediante tres condensadores conectados en paralelo, lo que imita el funcionamiento de una batería al incrementar la capacidad de almacenamiento sin alterar el nivel de voltaje.

4. SIMULACIÓN Y MONTAJE INICIAL DE CIRCUITO DE PIEZOELÉCTRICOS Y CIRCUITO ANALÓGICO

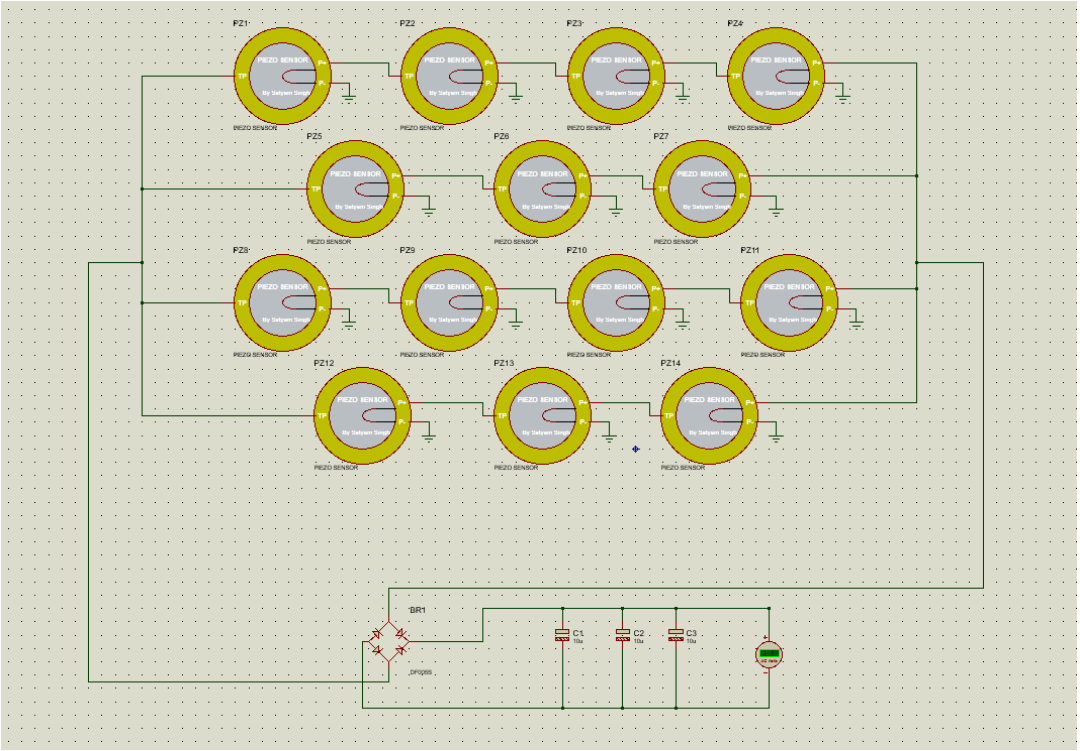

Figura 2. Implementación en Proteus.

La simulación en Proteus tiene como objetivo evaluar el comportamiento del circuito diseñado para la captación y almacenamiento de energía generada por discos piezoeléctricos. Se analizan la disposición de los componentes, la rectificación de la señal y la eficiencia del almacenamiento de energía en los condensadores.

4.1. Disposición en la simulación

Generación de Energía – Arreglo de Piezoeléctricos: En la parte superior de la simulación, se observa un conjunto de 14 discos piezoeléctricos organizados en un esquema serie/paralelo. Esta disposición busca equilibrar voltaje y corriente para mejorar la eficiencia del sistema. Cada disco piezoeléctrico actúa como una fuente de voltaje alterno que responde a la presión aplicada.

Rectificación de la Señal – Puente de Diodos: La energía generada por los piezoeléctricos se dirige hacia un puente de diodos 1N4148, encargado de convertir la señal alterna (AC) en continua (DC). Este puente de diodos permite una rectificación de onda completa, utilizando ambas mitades de la señal alterna y mejorando la eficiencia energética.

Filtrado y Almacenamiento – Banco de Condensadores: A la salida del puente de diodos, se conecta un banco de condensadores en paralelo compuesto por tres condensadores de 10 µF - 50V. La función principal de estos condensadores es suavizar la señal rectificada y almacenar la energía generada. La conexión en paralelo aumenta la capacidad total de almacenamiento y ayuda a reducir la fluctuación del voltaje.

4.2. Resultado de simulación en proteus de “una pisada”

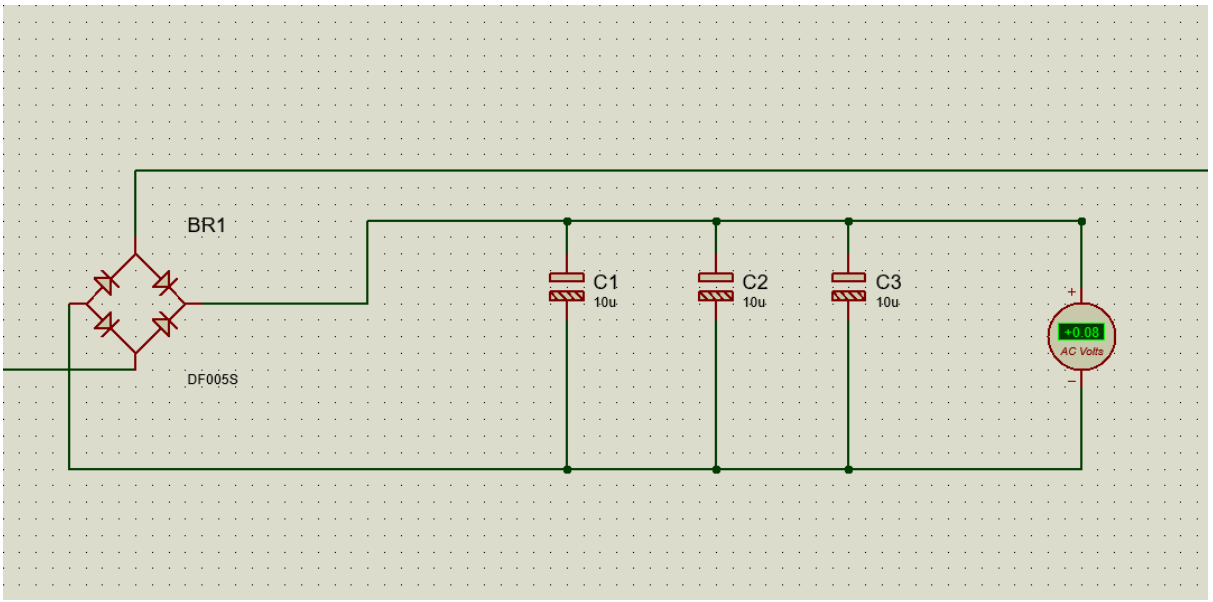

Figura 3. Simulación de una pisada.

Medición de la Salida: En la simulación se ha colocado un voltímetro a la salida del banco de condensadores. La lectura del voltímetro muestra un valor de 0.08V AC, tras únicamente una señal de entrada interpretada como una pisada.

4.3. Resultados Observados en la Simulación:

- El circuito logra rectificar la señal alterna generada por los piezoeléctricos.
- Los condensadores permiten almacenar la energía y estabilizar la salida.
- Se observa una pequeña oscilación en la salida, posiblemente debido a la carga dinámica de los piezoeléctricos o la falta de un filtrado adicional.
- La configuración serie/paralelo en los piezoeléctricos parece adecuada para equilibrar voltaje y corriente.

A continuación se adjuntan imágenes de la implementación física del proyecto.

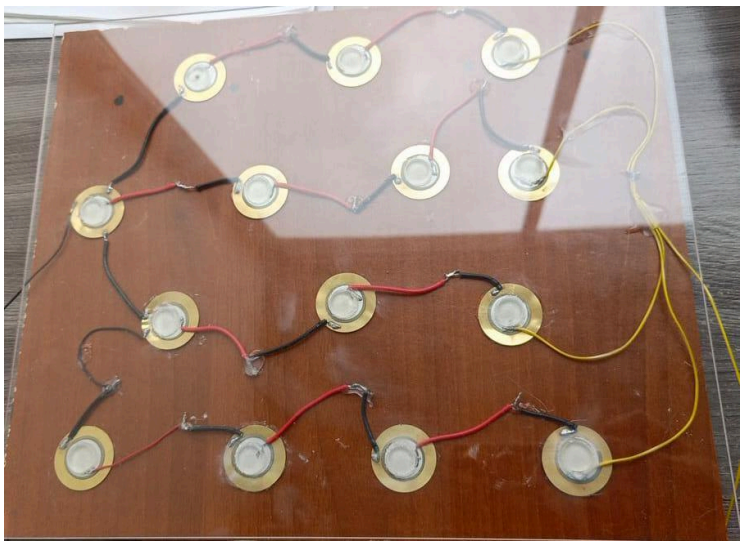

Figura 4. Implementación física circuito piezoeléctricos sobre tabla de madera.

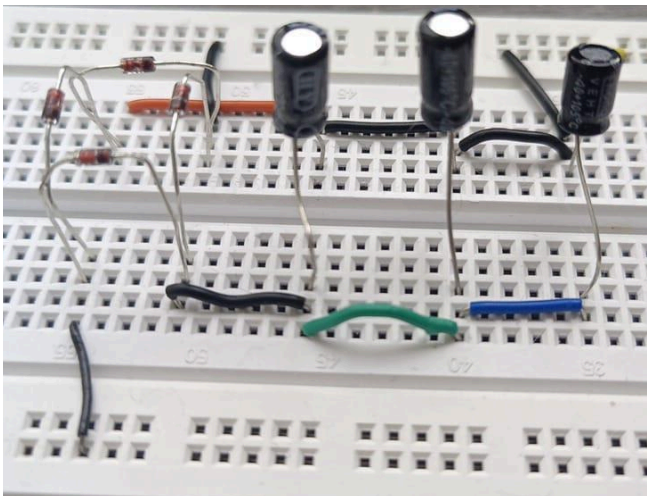

Figura 5. Circuito análogo - rectificador de onda y condensadores en paralelo.

El circuito de los piezoeléctricos se conecta con el circuito análogo para que éste rectifique la onda de corriente alterna. La tensión recae sobre los condensadores para posteriormente ser medida y pasar por el ADC para poder ser procesado en valores binarios por la FPGA.

4.4. Descripción Hardware:

5. FPGA

La FPGA en este proyecto tiene la función de procesar los datos digitales que recibe del ADC ADS1015. Los valores binarios que llegan a la FPGA son procesados internamente y transformados en arreglos de vectores. Estos vectores representan los datos de manera organizada y eficiente, lo que permite realizar cálculos y análisis, como la cantidad de energía generada.

El proceso exacto de cómo se realiza esta transformación está pendiente de detalles adicionales, pero en términos generales, la FPGA utiliza bloques lógicos programables para manejar estos bits y manipularlos en arreglos que se pueden utilizar para el procesamiento posterior. Esto facilita el análisis y la gestión de los datos de manera efectiva en tiempo real.

En cuanto a los recursos utilizados en la FPGA, al hace urso de la función make log-syn, obtenemos que:

6. PROTOCOLO I²C: ADC ADS1015

El ADC sirve para convertir la señal analógica generada por los piezoeléctricos en datos digitales que puedan ser procesados por la FPGA. Este ADC tiene un convertidor de 12 bits, que permite convertir el voltaje analógico a una representación digital con alta resolución.

Tras que la señal de salida ha sido rectificada y suavizada mediante los condensadores, estabilizando el voltaje para una medición precisa, la conversión a digital con ADS1015 da lugar. Una vez que la señal es estabilizada y rectificada, se envía al convertidor analógico-digital (ADC) ADS1015. Este ADC opera a través del protocolo de comunicación I²C, lo que permite su fácil integración con dispositivos como la FPGA, ya que utiliza solo dos líneas de comunicación: SDA (data) y SCL (clock). El ADS1015 convierte la señal analógica de voltaje en valores digitales que representan la intensidad del voltaje en formato binario, con una resolución de 12 bits.

6.1. Diseño del protocolo I2C

Para diseñar el protocolo I2C se consideró la información suministrada por la hoja de datos del ADS1015. En la figura 6 se puede observar el comportamiento correspondiente a las señales SCL y SDA cuando se desea realizar una operación de escritura y lectura.

El protocolo I2C parte de tener ambas señales en un valor lógico 1. Cuando se desea iniciar una nueva comunicación se coloca la señal SDA en un valor lógico 0, lo que pone en funcionamiento la señal de reloj SCL. Tras colocar SDA en 0, enviamos la dirección del esclavo con el que nos queremos comunicar, seguido de un bit para indicar si vamos a realizar una operación de escritura (R/W = 0) o de lectura (R/W = 1), y si la dirección es correcta, el dispositivo nos responderá con un bit de confirmación ACK = 0.

- Operación de escritura: Si R/W es cero (0), a continuación enviamos la dirección del registro al cual queremos apuntar. El registro con dirección 8b’00000001 se denomina registro de configuración y aquel con dirección 8b’00000000 se le conoce como registro de conversión. El registro de configuración nos permite modificar la funcionalidad del conversor. Por su parte, el registro de conversión almacena el valor de la señal analógica en formato digital.
    - Registro de configuración: Podemos configurar el ADS1015 enviando dos bytes de información, de acuerdo con los parámetros y las posibilidades presentadas en su hoja de datos. Así como con el registro de apuntador, enviamos los primeros 8 bits, a los que el esclavo responderá con un 0, luego

enviamos el segundo conjunto de bits, y finalizamos la transmisión colocando la señal SDA en 0.

- Registro apuntador: Solo podemos acceder a este registro cuando queremos realizar una operación de escritura. A partir del valor que le asignemos, en una futura operación de lectura podremos recuperar un cierto conjunto de datos. En otras palabras, podemos leer tanto el registro de configuración como el registro de conversión.

- Operación de lectura: Si R/W es uno (1), habilitamos la señal SDA como entrada para recibir datos a través de ella. Posteriormente recibimos 8 bits desde el esclavo, luego enviamos un bit de confirmación ACK = 0, recibimos los siguientes 8 bits, y finalizamos la comunicación con otro bit de confirmación y colocando la señal SDA en 0.

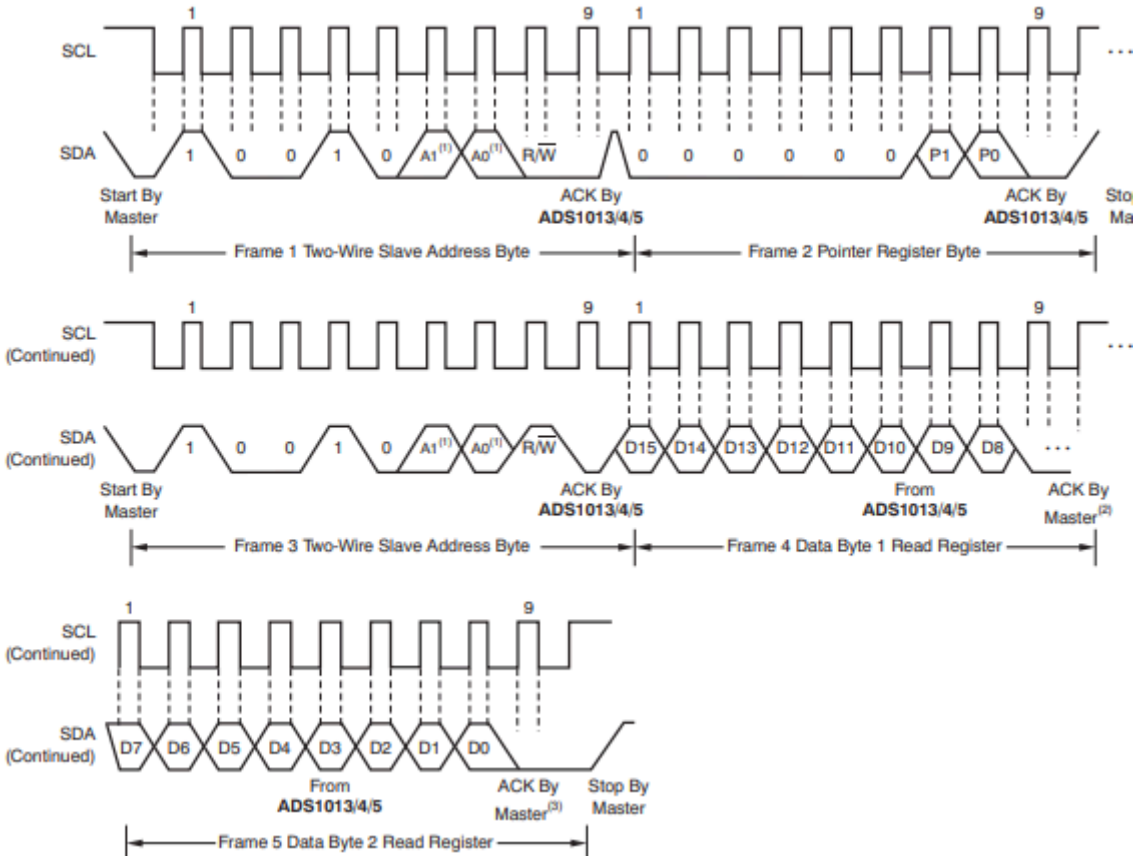

Figura 6. Comportamiento de señales SDA y SCL del protocolo I2C.

Otros aspectos importantes para considerar:

- La señal SCL tiene una frecuencia de 0.1 MHz; valor que se encuentra dentro del rango indicado en la hoja de datos del ADS1015.
- La señal SCL permanece en 1 mientras se transmite un dato. Los tiempos mínimos de estado en alto y bajo se pueden observar en la hoja de datos.
- Cada transmisión de datos finaliza colocando la señal SCL en 1, seguido de la señal SDA en 1.

6.2. Máquina de Estados en el Módulo I2C:

A continuación, se presenta la explicación de la máquina de estados realizada para ejecutar el protocolo de comunicación I²C.

El código usa una máquina de estados finitos (FSM) para manejar la comunicación I2C. Esta FSM se define con el registro S, que puede tomar los siguientes valores:

- Estado 0: Inicio de transmisión
- Estado 1: Escritura de datos
- Estado 2: Lectura de datos
- Estado 3: Final de transmisión

La FSM cambia de estado según condiciones específicas, y cada estado ejecuta ciertas acciones.

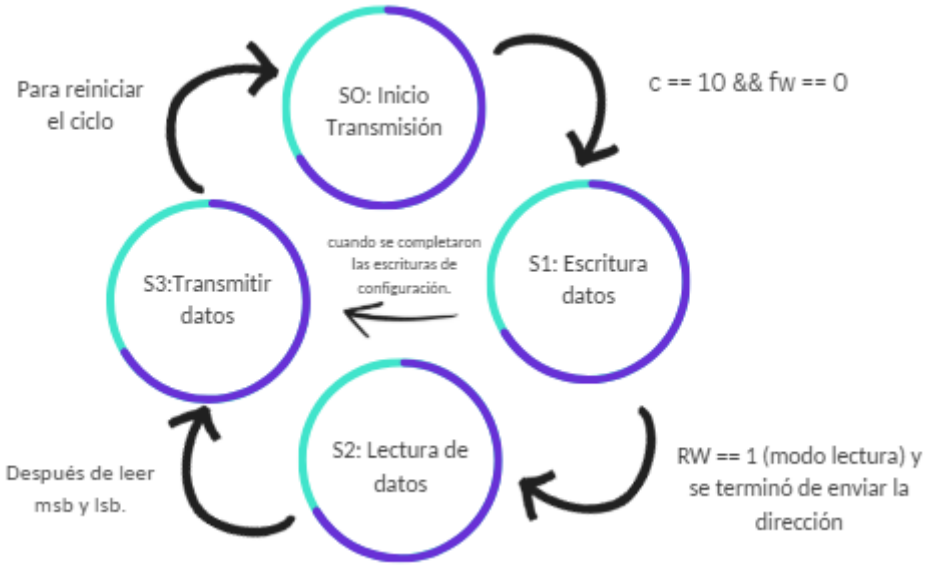

Figura 7. Máquina de estados I2C.

A continuación, se presenta la explicación del código de la máquina de estados realizada en verilog:

Estado 0: Inicio de transmisión

Este estado prepara la línea SDA para comenzar la transmisión.

```
C/C++
case(S)
```

```
    0: begin    // Inicio de transmisión
        if (SCL_en == 0) begin
            SDA_en <= 1;
            SDA_out <= 1;
        end

        if (c == 10 && fw == 0)begin
            SDA_en <= 1;
              SDA_out <= 0; // Se genera la condición de
START (SDA pasa de 1 a 0 mientras SCL sigue en 1)
            S <= 1; // Se pasa al estado de escritura
        end
    end
```

¿Qué está pasando aquí?

1. Se asegura de que `SDA` esté en alto (`SDA_out = 1`).
2. Cuando el contador `c` llega a 10, `SDA_out` cambia a 0, lo que indica el inicio de la comunicación I2C.
3. Se pasa al Estado 1 para comenzar la escritura de datos.

Estado 1: Escritura

Aquí se envían los datos al esclavo I2C. Se usa otra variable llamada `sw` para dividir este estado en 4 sub-estados:

C/C++
```
1: begin    // Estado de escritura
```

```
    case (sw)
        2'b00: begin // Envío de dirección del esclavo
            bc <= 1;  // Se activa la transmisión de bits
            if (cf <= 8) begin
                SDA_en  <= 1;
                  SDA_out <= dir_msj[8 - cf]; // Se envía bit por bit la dirección del esclavo
            end
             else if (cf == 9) begin // Después del 9º bit (ACK)
                  SDA_en  <= 0; // Liberamos SDA para que el esclavo pueda responder
                if (RW == 1)
                        S   <= 2;  // Si se está en modo lectura, pasar a estado 2
                else
                      sw <= 2'b01; // Si se está en modo escritura, pasar al siguiente subestado
            end
        end
```

¿Qué está pasando aquí?

1. Se envía la dirección del dispositivo esclavo en `dir_msj` (7 bits de dirección + 1 bit de lectura/escritura).
2. Se cuenta `cf` hasta 9 (8 bits + 1 bit ACK).
3. Si `RW == 1`, se cambia al estado 2 (lectura).

4. Si `RW == 0`, se pasa a `sw = 01` para enviar más datos.

Sub-estado 2'b01: Envío del Registro Apuntador

```
C/C++
        2'b01: begin // Envío del registro de apuntador
            if (cf <= 8) begin
                SDA_en  <= 1;
                 SDA_out <= reg_ap[8 - cf]; // Se envía el
byte de registro
            end
            else if (cf == 9) begin // Se espera el ACK
                SDA_en  <= 0;
                if (ap == 1'b1)
                            sw <= 2'b10; // Si `ap == 1`,
continuar con más datos
                else begin
                    fw <= 1;
                     S <= 3; // Si no hay más datos, ir al
estado final
                end
            end
        end
```

¿Qué está pasando aquí?

1. Se envía un byte (`reg_ap`) que indica qué registro se quiere leer/escribir en el esclavo.
2. Si `ap == 1`, se pasa a `sw = 10` para enviar más datos.

3. Si `ap == 0`, se finaliza la comunicación (estado `S = 3`).

Sub-estado 2'b10 y 2'b11: Envío de Configuración

```C/C++
        2'b10: begin // Envío de configuración 1
            if (cf <= 8) begin
                SDA_en  <= 1;
                  SDA_out <= conf1[8 - cf]; // Se envía el
primer byte de configuración
            end
            else if (cf == 9) begin
                SDA_en  <= 0;
                sw <= 2'b11; // Ir al siguiente dato
            end
        end

        2'b11: begin // Envío de configuración 2
            if (cf <= 8) begin
                SDA_en  <= 1;
                  SDA_out <= conf2[8 - cf]; // Se envía el
segundo byte de configuración
            end
            else if (cf == 9) begin
                SDA_en  <= 0;
```

```
                fw <= 1;
                S <= 3; // Ir al estado final
            end
        end
```

¿Qué está pasando aquí?

- Se envían los datos `conf1` y `conf2` al esclavo.
- Luego se va al estado 3 para finalizar.

Estado 2: Lectura de Datos

Si `RW == 1`, se entra en este estado. Se lee la información enviada por el esclavo.

```C/C++
2: begin    // Estado de lectura
    if (sw == 0) begin
        if (cf <= 8)
            msb[8 - cf] <= SDA_in; // Se almacena el byte alto
        else if (cf == 9) begin
            SDA_en <= 1;
            SDA_out <= 0; // Se envía ACK
            sw <= 1;
        end
    end
    else begin
```

```
        if (cf <= 8) begin
            SDA_en <= 0;
             lsb[8 - cf] <= SDA_in; // Se almacena el byte
bajo
        end
        else if (cf == 9) begin
            SDA_en <= 1;
            SDA_out <= 0;
            S <= 3; // Ir al estado final
        end
    end
end
```

¿Qué está pasando aquí?

1. Se lee el primer byte (`msb`).
2. Luego se envía un ACK (`SDA_out = 0`).
3. Se lee el segundo byte (`lsb`).
4. Se pasa al estado 3 para finalizar.

Estado 3: Final de transmisión

```C/C++
3: begin    // Final de transmisión
    bc <= 0;
    S  <= 1'b0;
    sw <= 2'b00;
```

```
end
```

¿Qué está pasando aquí?

- Se limpian las variables (`bc`, `S`, `sw`).
- La FSM vuelve al estado 0 para empezar otra transmisión.

De igual manera, se puede observar el funcionamiento de la señal SCL (azul) y SDA (amarilla), en donde:

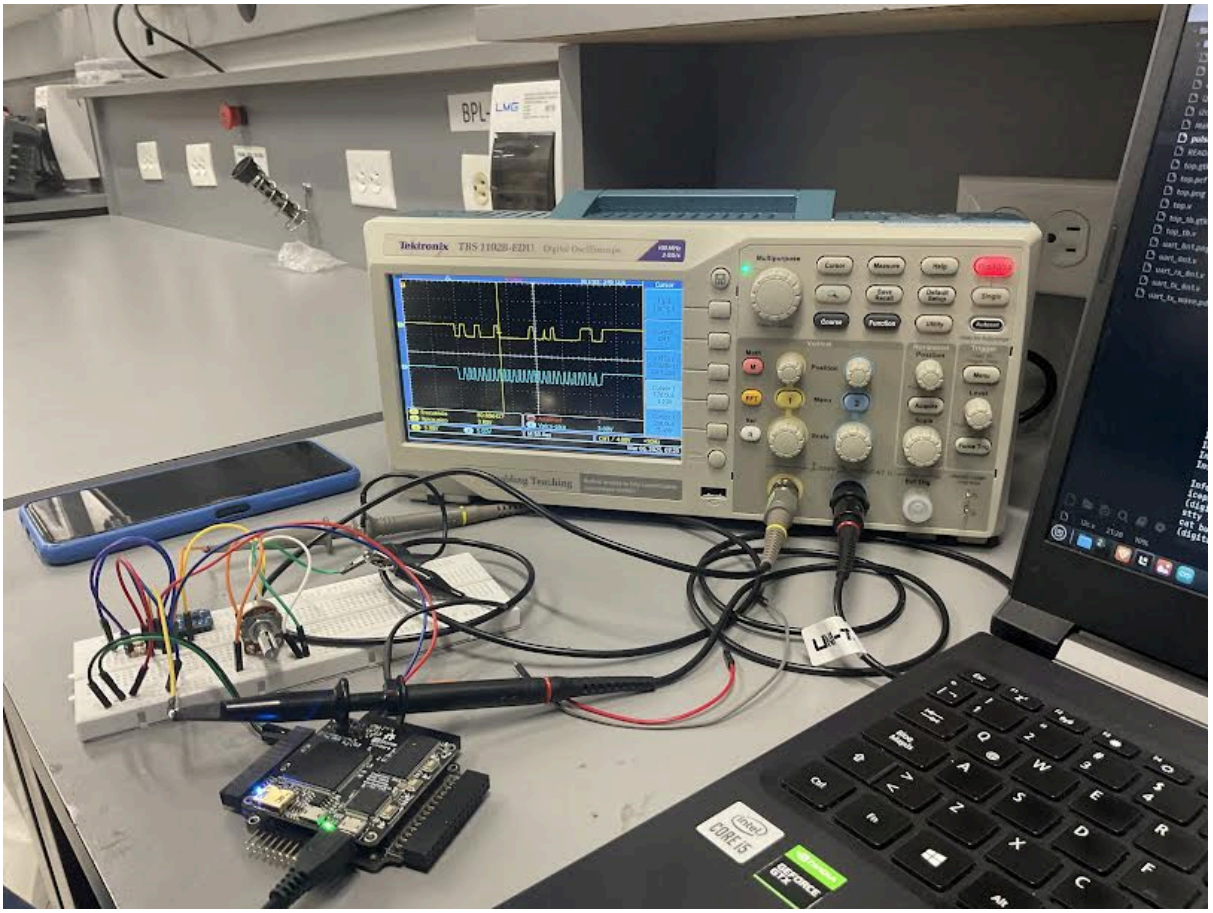

Figura 8. Comprobación del I2C con el osciloscopio.

7. DISEÑO Y MONTAJE DE PCB

La implementación analógica del circuito se diseñó con el objetivo de generar una PCB con el fin de cumplir con la rúbrica en cuanto a la presentación del proyecto, para eso se hizo uso del software KiCad, el cual permitió establecer este aspecto.

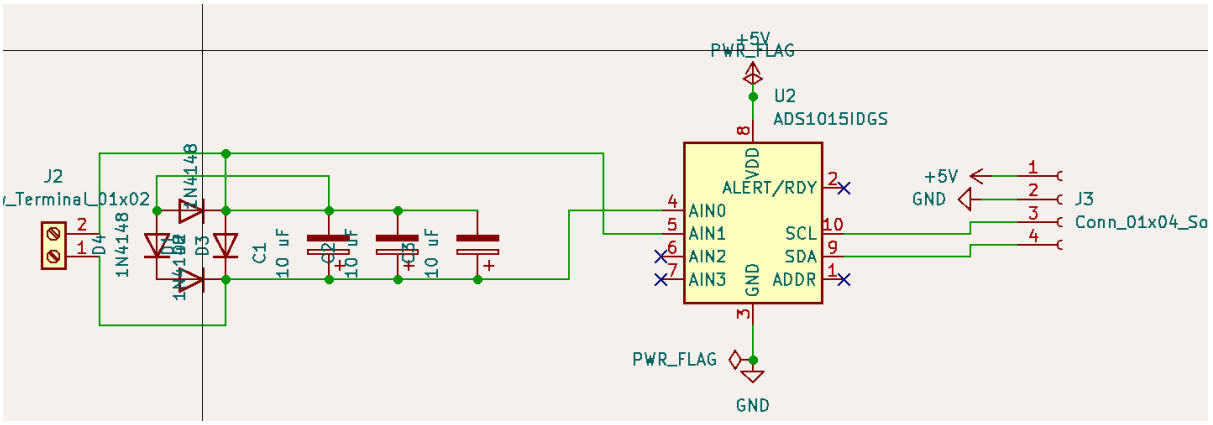

Figura 6. PCB en KiCad.

Se implementó la entrada de los piezoeléctricos como una bornera de 1x2, luego el puente rectificador de diodos con referencia al 1N4148 y los tres condensadores electrolíticos en paralelo. Finalmente, KiCad permite la implementación del conversor ADS1015, se relacionaron las entradas, las salidas, los puertos de energización y salida a la FPGA.

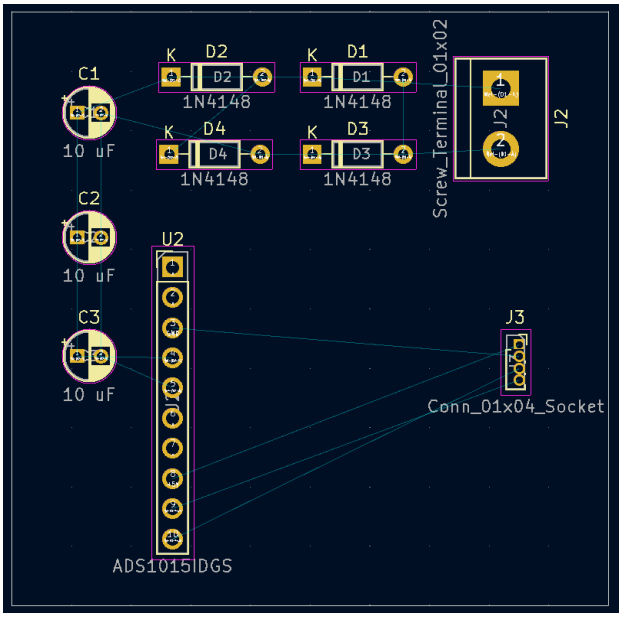

Figura 7. Implementación PCB.

Finalmente este es el diseño propuesto de la pcb, con un tamaño aproximado de 5x5 cm, sin embargo, se está a la espera de la implementación del ESP32 el cual nos ayudará a la transmisión y visualización por medio de WiFi, se espera adjuntar este dispositivo al diseño de la PCB propuesta.

8. MODULO ESP-32

Una vez que los datos son procesados por la FPGA, está los transmite a un módulo ESP-32 a través de un cable UART (Universal Asynchronous Receiver-Transmitter). El UART es un protocolo de comunicación que permite la transferencia de datos entre la FPGA y el ESP-32

de manera serial. Este protocolo es ideal para la comunicación entre dispositivos a través de un canal de datos simple, utilizando dos líneas principales: una para transmitir (TX) y otra para recibir (RX).

El ESP-32 actúa como un intermediario entre la FPGA y la nube de Arduino Cloud. Este módulo tiene capacidades de conectividad Wi-Fi, lo que le permite enviar los datos procesados de la FPGA de manera inalámbrica a la nube. A través de la red Wi-Fi, el ESP-32 transmite los datos en tiempo real, facilitando su visualización y monitoreo desde cualquier dispositivo conectado a Internet.

Una vez que los datos llegan a la Arduino Cloud, pueden ser almacenados, procesados y visualizados de manera remota. La nube permite acceder a los datos de manera centralizada y proporciona herramientas para analizar el rendimiento del sistema de generación de energía en tiempo real. Esto es especialmente útil para monitorear parámetros como la cantidad de energía generada, además de proporcionar un historial de los datos recolectados.

Este flujo de datos desde la FPGA hacia la nube a través del ESP-32 asegura que la información esté siempre disponible para su análisis y permite una gestión eficiente de los recursos generados por los sensores piezoeléctricos. Además, al usar Arduino Cloud, se habilitan opciones de integración con otros dispositivos y plataformas de IoT, lo que amplía las posibilidades de expansión y control del sistema.

9. ARDUINO CLOUD

Arduino Cloud es una plataforma web que permite gestionar y monitorear proyectos de Internet de las Cosas (IoT) de manera sencilla y eficiente. A través de esta plataforma, los dispositivos conectados a la red, como el ESP-32 en este proyecto, pueden transmitir datos de manera remota, los cuales se pueden visualizar, almacenar y analizar en tiempo real desde cualquier dispositivo con acceso a Internet.

Una de las características más destacadas de Arduino Cloud es su integración con la Arduino IoT Cloud que permite a los usuarios crear Dashboard personalizados, que proporcionan una interfaz visual para controlar y monitorear los datos de los dispositivos conectados. En el contexto de tu proyecto, una vez que los datos procesados por la FPGA son enviados al ESP-32, este los transmite a la nube, donde pueden ser visualizados en tiempo real a través de estos Dashboards. Esto facilita el acceso remoto a la información generada, como la cantidad de pasos o la energía recolectada.

Además de su capacidad para monitorear en tiempo real, Arduino Cloud permite configurar alertas y notificaciones, lo que podría ser útil para señalar eventos específicos, como un incremento inusual en la energía generada o problemas técnicos en el sistema. También facilita la gestión de dispositivos de manera centralizada, lo que permite realizar actualizaciones y ajustes de forma remota sin necesidad de intervención física.

Arduino Cloud también ofrece funcionalidades para almacenar datos históricos, permitiendo un análisis a largo plazo del rendimiento del sistema, lo cual es valioso para evaluar la

eficiencia del sistema de generación de energía piezoeléctrico y tomar decisiones informadas sobre futuras mejoras o ajustes. Esta integración de datos en la nube optimiza la operatividad del sistema, ya que los usuarios pueden tener acceso constante y remoto a la información, contribuyendo a una gestión más eficiente y a una mejor toma de decisiones.

Este código en **Arduino** se encarga de recibir datos por el **puerto serial**, procesarlos y calcular un valor de voltaje a partir de un número digital recibido. A continuación, te explico cada parte paso a paso:

9.1. Definición de Variables

```C/C++
uint8_t estado, msb, lsb;

int mensaje, v;

int16_t num_dig;
```

- `estado`: almacena los datos recibidos por el **puerto serial**.
- `msb` y `lsb`: guardan el byte más significativo (**Most Significant Byte - MSB**) y el byte menos significativo (**Least Significant Byte - LSB**) de un número de 12 bits.
- `mensaje`: sirve como un **indicador de estado** para procesar los datos en diferentes etapas.
- `v`: almacena el voltaje calculado.
- `num_dig`: contiene el número digital de 12 bits obtenido de `msb` y `lsb`.

9.2. Función de Complemento a 2

```C/C++
int16_t complemento2(uint16_t num){
  if (num & 0x800)
    return num | 0xF000;
  else
    return num;
}
```

- Esta función convierte un número de **12 bits con signo** (que usa complemento a 2) a un número de **16 bits con signo**.
- Si el bit más significativo (**bit 11**) es **1** (`num & 0x800`), significa que el número es negativo y se extiende el signo con `0xF000` para que mantenga su valor negativo en 16 bits.
- Si el número es positivo, simplemente lo devuelve sin cambios.

Ejemplo:

- `num = 0x7FF (0111 1111 1111) → positivo, no cambia.`
- `num = 0x800 (1000 0000 0000) → negativo, se convierte en 0xF800 (1111 1000 0000 0000).`

9.3. Configuración Inicial (`setup()`)

```C/C++
void setup() {

  Serial.begin(9600);



  msb = 0;

  lsb = 0;

  num_dig = 0;

}
```

- **Inicia la comunicación serial** a **9600 baudios**.
- Inicializa `msb`, `lsb` y `num_dig` en **0**.

9.4. Bucle Principal (`loop()`)

```C/C++
void loop() {

  if (Serial.available()){

    estado = Serial.read();
```

```
  }
```

- **Si hay datos en el puerto serial**, los lee y los almacena en `estado`.

9.4.1. Procesamiento del Mensaje

```C/C++
  if (estado == 253) {
    mensaje = 1;
  } else if (mensaje == 1) {
    msb = estado;
    mensaje = 2;
  } else if (mensaje == 2) {
    lsb = estado;
    mensaje = 3;
  }
```

- **El protocolo de comunicación usa un byte de inicio con el valor `253`.**
- Si recibe `253`, **se prepara para recibir los datos** (`mensaje = 1`).
- Luego, recibe el `MSB` y lo almacena en `msb`, **pasando al estado 2**.
- Después, recibe el `LSB` y lo almacena en `lsb`, **pasando al estado 3**.

9.5. Conversión de Datos

```C/C++
  if (mensaje == 3) {
     num_dig = (msb << 4) | (lsb >> 4);  // Se corrigió el desplazamiento
    mensaje = 0;  // Reiniciar para esperar nuevos datos
```

```
}
```

- Una vez que se tienen `msb` y `lsb`, se reconstruye el número de **12 bits**:
  - Se desplaza `msb` **4 bits a la izquierda** (`msb << 4`).
  - Se desplaza `lsb` **4 bits a la derecha** (`lsb >> 4`).
  - Se combinan usando `|` (OR bit a bit) para formar el número completo.

Ejemplo:

- `msb = 0x12 (0001 0010)`, `lsb = 0x34 (0011 0100)`.
- `num_dig = (0x12 << 4) | (0x34 >> 4) = (0010 0100 0000) | (0000 0011) = 0010 0100 0011`.

9.6. Conversión a Complemento a 2

C/C++

```
int16_t num = complemento2(num_dig);

Serial.println(num);
```

- Se convierte el número de 12 bits en un entero con **signo de 16 bits** usando `complemento2(num_dig)`.
- Se imprime el número resultante en el monitor serial.

9.7. Cálculo del Voltaje

C/C++

```
v = 5*(num_dig - (-274))/(1200 - (-274));
```

- **Interpola el número digital `num_dig` a un voltaje**:
  - **Cuando `num_dig = -274`, el voltaje es 0.7V**.
  - **Cuando `num_dig = 1200`, el voltaje es 5V**.
  - Se usa la **ecuación de la recta** entre estos dos puntos para calcular el voltaje `v`

10. RESULTADOS FINALES

DESCRIPCIÓN DE LA SOLUCIÓN REALIZADA

La solución desarrollada en este proyecto consistió en la implementación de un sistema de captación de energía basado en sensores piezoeléctricos, con la capacidad de medir en tiempo real el voltaje generado y almacenado en condensadores que simulan una batería. Para ello, se diseñó una arquitectura que integra una FPGA BlackIce 40, encargada de la adquisición de datos a través de un ADC capacitivo ADS1015, y un ESP32-S2 Mini, responsable de la transmisión de información a Arduino Cloud para su visualización remota.

Inicialmente, en la solución propuesta, se contemplaba no solo la captación de energía, sino también la medición del número de pasos realizados por el usuario. Sin embargo, por cuestiones de tiempo y optimización, esta funcionalidad fue descartada, ya que requería la integración de un circuito adicional en el puente rectificador para detectar los pulsos generados por cada paso. Esta modificación habría añadido complejidad al diseño y al procesamiento de datos, comprometiendo el desarrollo dentro del plazo establecido.

A pesar de esta diferencia con la solución inicial, el objetivo principal del proyecto se mantuvo inalterado: demostrar la viabilidad de captar energía piezoeléctrica de los pasos y visualizar en tiempo real el voltaje generado. La arquitectura implementada permite un monitoreo eficiente y sienta las bases para futuras mejoras, como la integración del conteo de pasos en versiones posteriores del sistema.

Solución y alcance del proyecto:

- **Solución del proyecto:**

  Este proyecto propone la implementación de pisos piezoeléctricos para la generación de energía mediante el aprovechamiento del movimiento humano, especialmente al caminar, en espacios públicos de alta afluencia como estaciones de transmilenio y aeropuertos. La energía generada por los sensores piezoeléctricos será convertida en electricidad y utilizada para alimentar sistemas de **iluminación** y **señalización** en estas áreas. Mediante una FPGA, se procesarán en tiempo real datos sobre la energía generada, los cuales serán enviados a dispositivos móviles, pantallas o computadoras para su monitoreo remoto. Esta solución innovadora no solo contribuye a la optimización del uso de energía, sino que también ofrece un sistema autónomo y sostenible para la gestión energética en espacios públicos.

- **Alcance del Proyecto:**

  El proyecto tiene como objetivo ofrecer una solución ecológica y eficiente al aprovechar el movimiento humano como fuente de energía renovable, alineándose con los **Objetivos de Desarrollo Sostenible (ODS)** de las Naciones Unidas, especialmente con los siguientes:

Fuente: Objetivos de desarrollo sostenible. [8]

1. **ODS 7 (Energía asequible y no contaminante):** Generación de energía renovable mediante el aprovechamiento del movimiento humano.
2. **ODS 9 (Industria, innovación e infraestructura):** Implementación de soluciones tecnológicas innovadoras y sostenibles en la infraestructura urbana.
3. **ODS 11 (Ciudades y comunidades sostenibles):** Mejora de la eficiencia energética en espacios públicos, contribuyendo a la sostenibilidad de las ciudades.
4. **ODS 12 (Producción y consumo responsables):** Promoción de tecnologías que optimizan el uso de los recursos y reducen el desperdicio energético.

A continuación, se detallan los principales componentes y funcionalidades del proyecto:

a. **Generación de Energía Piezoeléctrica:** El sistema se basará en sensores piezoeléctricos integrados en los pisos, que convertirán la energía cinética del caminar de los usuarios en electricidad, la cual será utilizada para alimentar luces y señales en estaciones de buses, aeropuertos y otras áreas públicas.
b. **Monitoreo en Tiempo Real:** La FPGA procesará los datos de energía generada, lo que permitirá un control eficiente y en tiempo real sobre el rendimiento del sistema. Los datos recopilados se enviarán a dispositivos externos (celulares, computadoras, pantallas) para su visualización y análisis.
c. **Iluminación y Señalización:** La energía generada alimentará sistemas de **iluminación** y **señalización** (pantallas informativas y direccionales) en espacios públicos, mejorando la visibilidad y la eficiencia energética en estos entornos de alta demanda.
d. **Implementación en FPGA:** El diseño y la lógica del sistema serán implementados en una FPGA, lo que permitirá procesar y gestionar la información de manera eficiente, integrando la recogida de datos y el control de energía en un solo sistema compacto y flexible.

El proyecto permitirá reducir la dependencia de fuentes energéticas tradicionales y minimizar las emisiones de carbono, contribuyendo de manera significativa al cumplimiento de metas internacionales y nacionales sobre la reducción de gases de efecto invernadero y el impulso de energías limpias.

11. CONSUMO FPGA

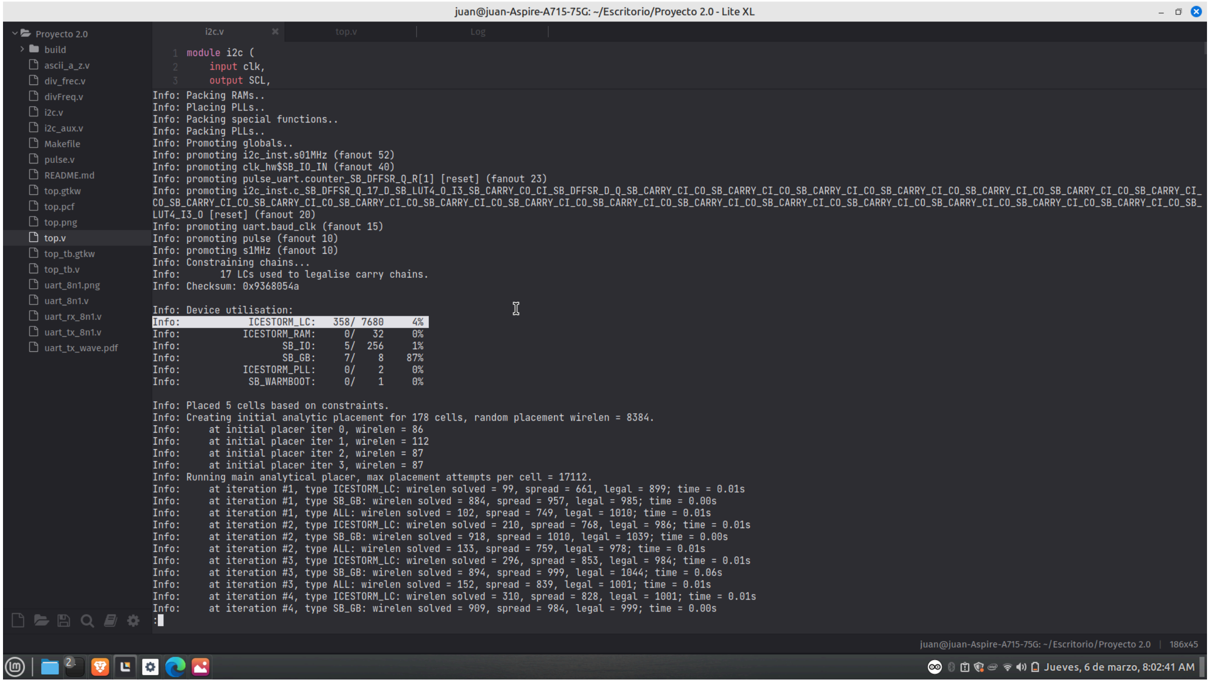

Figura 8. Consumo FPGA.

En la implementación del sistema en la FPGA iCE40, se observó un consumo eficiente de recursos. El diseño ocupa 358 de 7680 celdas lógicas (LUTs), lo que representa un 4% del total disponible. Además, no se hace uso de la memoria RAM interna (0 de 32 bloques, 0%).

Por otro lado, se observa un uso completo del único PLL disponible (1 de 1, 100%), lo cual indica que el sistema depende de este recurso para la gestión de frecuencia. También se emplea un 87% de los buffers globales de señal (SB_GB: 8 de 8), lo que sugiere un alto uso de redes de distribución de reloj o señales críticas en el diseño.

Estos valores reflejan una utilización moderada de los recursos lógicos, permitiendo futuras expansiones, aunque el uso del PLL y los buffers globales podría requerir optimización en versiones futuras del proyecto.

12. PRESUPUESTO

A continuaciòn, se muestra el presupuesto completo en una sola tabla, organizada de derecha a izquierda e incluyendo todos los costos:

Total (COP)	Precio Unitario (COP)	Cantidad	Ítem
$ 4.000	$ 2.000	2	Silicona
$ 15.600	$15.600	1	Lámina de Acrílico
$ 20.000	$ 1.000	20	Piezoeléctricos
$ 4.000	$ 400	10	Resortes
-	-	14	Tapas
$ 19.500	$ 1.393	14	Goma silicona adhesiva
$ 25.000	$25.000	1	PCB
-	-	-	Cables
$ 15.000	$15.000	1	Hoja de segueta
-	-	-	Superbonder
$ 90.100	-	-	**Subtotal Materiales**
$ 20.000	$ 10.000/hora	2	Mano de obra (ensamblaje)
$ 800.000	-	1	Desarrollo de código (1 mes)
$ 5.000	-	1	Distribución y empaques
$ 105.100	-	-	**Costo Total de Producción por Unidad**
$ 500.000 $10.000.000	-	-	**Precio de Venta según Escala**
$50.500.000	-	-	**Ingresos Proyectados a 6 Meses**
$ 1.366.300	-	-	**Costo Total de Producción (13 sistemas)**
$59.133.700	-	-	**Ganancia Bruta Estimada**

El sistema de generación de energía basado en sensores piezoeléctricos ofrece una solución innovadora para aprovechar el tránsito de personas en espacios de alto flujo, permitiendo visualizar en tiempo real la cantidad de pasos y la energía generada. Este producto está

dirigido a estaciones de metro, organizadores de eventos y centros comerciales, donde puede contribuir a la sostenibilidad y optimización del consumo energético.

La estrategia de ventas contempla distintos modelos según el cliente. Para pequeñas instalaciones de prueba o empresas interesadas en explorar la tecnología, se propone un sistema individual con un precio de venta entre $250.000 y $500.000 COP. Para instalaciones más amplias en estaciones del metro de Bogotá o eventos de gran magnitud, se plantea un sistema modular con un rango de $3.000.000 a $10.000.000 COP, dependiendo del área cubierta, la cantidad de sensores y la integración con plataformas de monitoreo. Además, se ofrece un modelo de servicio con contrato de mantenimiento y monitoreo, cuyo costo mensual puede variar entre $500.000 y $1.500.000 COP, garantizando la operatividad del sistema, la actualización del software de visualización y el análisis de datos.

En un escenario conservador, donde se vendan tres sistemas medianos a estaciones de metro por $5.000.000 COP cada uno, cinco instalaciones modulares en eventos por $3.000.000 COP cada una, cinco sistemas pequeños en centros comerciales por $500.000 COP cada uno, y se concreten tres contratos de mantenimiento de $1.000.000 COP mensuales durante seis meses, los ingresos proyectados a medio año alcanzarían aproximadamente $50.500.000 COP. Para maximizar el impacto comercial, es fundamental desarrollar prototipos funcionales con material audiovisual que evidencie su efectividad, establecer contactos con empresas de transporte, eventos y retail, y diseñar una plataforma de monitoreo visual para agregar valor al producto. Asimismo, se recomienda explorar alianzas con entidades gubernamentales y organizaciones interesadas en energías renovables para obtener financiamiento y respaldo institucional.

11.1. Cálculo del Costo de Producción

El punto de partida es el costo de fabricación del sistema. Según la lista de materiales, el costo por unidad es de $80.100 COP, sin incluir costos indirectos. Adicionalmente, se han considerado:

- Mano de obra: Si ensamblar cada unidad toma 2 horas y el pago promedio es $10.000 COP/hora, se suma $20.000 COP por unidad. La realización del código tomó aproximadamente 1 mes, por esa razón se considera un precio de $800.000 COP.
- Distribución y empaques: $5.000 COP por unidad para envíos y embalaje.
- Software y monitoreo en tiempo real: No tiene un costo directo por unidad, pero se agrega valor mediante el servicio de datos.
- Costo total estimado por unidad: $105.100 COP.

11.2. Definición de Precio de Venta

El precio de venta no se basa solo en los costos, sino en el valor agregado y el mercado objetivo. Para esto, se han usado los siguientes criterios:

- Margen de ganancia: Un producto tecnológico con diferenciación suele aplicar un margen del 50% al 200% sobre costos.
- Benchmarking: Tecnologías similares de generación de energía y análisis de datos en el mercado pueden costar desde $500.000 hasta varios millones de pesos, dependiendo de la escala.
- Escalabilidad: Clientes como estaciones de metro y organizadores de eventos están dispuestos a pagar más por un sistema modular y adaptable.

11.3. Precios de referencia por tipo de cliente:

- Sistemas pequeños: $500.000 COP (para pequeñas empresas o pruebas piloto).
- Sistemas medianos: $5.000.000 COP (para estaciones de metro y eventos de tamaño medio).
- Sistemas grandes: $10.000.000 COP (para eventos masivos y proyectos gubernamentales).
- Servicio de mantenimiento y monitoreo: $1.000.000 COP/mes (para garantizar el funcionamiento, datos en tiempo real y soporte técnico).

Rentabilidad y Margen de Ganancia

Para estimar la rentabilidad, se considera:

- Costo total de producción: Si en seis meses se venden 13 sistemas (3 grandes, 5 medianos, 5 pequeños), el costo de fabricación sería aproximadamente $105.100 × 13 = $1.366.300 COP.
- Ganancia bruta estimada: $50.500.000 - $1.366.300 = $49.133.700 COP (sin contar gastos adicionales de logística, impuestos y márketing).
- Margen de ganancia sobre costos directos: Aproximadamente 97%.

11.4. Factores a Considerar en la Venta

1. Diferenciación tecnológica: El sistema no solo genera energía, sino que visualiza en tiempo real el impacto, lo que lo hace atractivo para empresas con interés en sostenibilidad.
2. Estrategia de ventas B2B: Se prioriza vender a entidades grandes (gobiernos, transporte público, eventos) en lugar de consumidores individuales.
3. Costos de implementación a gran escala: Aunque la fabricación es económica, la instalación en espacios públicos requiere alianzas estratégicas.

El precio de venta está definido estratégicamente para maximizar la rentabilidad sin ser inaccesible para los clientes objetivo. Si se logran ventas en el rango proyectado, el negocio puede generar ingresos significativos con costos de producción relativamente bajos. Esta estrategia también permite escalar el modelo y explorar nuevas fuentes de ingresos a través de contratos de mantenimiento y ampliaciones modulares.

13. CONCLUSIONES

- A lo largo del desarrollo de este proyecto, se logró diseñar e implementar un prototipo funcional de un sistema de captación de energía basado en sensores piezoeléctricos, capaz de medir en tiempo real la energía generada por el paso de personas y el voltaje almacenado en condensadores que simulan una batería.
- La utilización de una FPGA BlackIce 40 junto con un ADC capacitivo ADS1015 permitió la adquisición precisa de los valores de voltaje mediante comunicación I2C y UART, garantizando un procesamiento eficiente de la información. Asimismo, la implementación de una máquina de estados en Verilog facilitó la gestión y transmisión de los datos desde los sensores piezoeléctricos hasta el ESP32-S2 Mini a través del protocolo SPI, asegurando un flujo de información confiable y sin interrupciones.
- Por otro lado, la integración con Arduino Cloud permitió la visualización remota y en tiempo real de los datos obtenidos, proporcionando una plataforma accesible para monitorear el desempeño del sistema. Esta funcionalidad abre la posibilidad de adaptar el prototipo a diferentes aplicaciones en entornos de alto tránsito, como estaciones de transporte, eventos masivos y espacios urbanos, donde la captación de energía cinética podría representar una fuente complementaria de generación energética.
- En conclusión, el proyecto demuestra la viabilidad de la captación de energía piezoeléctrica como una tecnología emergente para el aprovechamiento de la energía del movimiento humano. Su implementación a mayor escala podría contribuir al desarrollo de sistemas energéticos sostenibles y eficientes en el ámbito urbano.

14. TRABAJOS FUTUROS

Este proyecto ha demostrado la viabilidad de la captación de energía piezoeléctrica para la medición y visualización en tiempo real del voltaje generado por el paso de personas. Sin embargo, existen diversas oportunidades para mejorar y expandir esta tecnología en futuras investigaciones y desarrollos.

1. Optimización del arreglo piezoeléctrico: Se pueden explorar diferentes configuraciones de conexión en serie y paralelo para maximizar la eficiencia en la captación de energía, así como el uso de nuevos materiales piezoeléctricos con mayor coeficiente de conversión.

2. Implementación de un sistema de almacenamiento eficiente: Se podría integrar un banco de supercondensadores o baterías que permita acumular la energía generada y regular su salida para alimentar dispositivos electrónicos de baja potencia.

3. Mejoras en la adquisición y procesamiento de datos: La implementación de técnicas de filtrado digital y algoritmos de optimización en la FPGA permitiría reducir ruidos

en la medición y mejorar la precisión de los valores obtenidos.

4. Ampliación del sistema de comunicación y visualización: Se pueden explorar otras plataformas IoT además de Arduino Cloud, como MQTT, Node-RED o aplicaciones móviles personalizadas, para hacer más accesible la información a usuarios finales y gestores energéticos.

5. Pruebas en entornos reales: La instalación del sistema en lugares de alto tránsito, como estaciones de metro, centros comerciales o eventos masivos, permitiría evaluar su desempeño en condiciones reales y validar su potencial de implementación a mayor escala.

6. Desarrollo de un modelo escalable: Se podría diseñar un sistema modular que permita conectar múltiples arreglos piezoeléctricos, facilitando su implementación en infraestructuras urbanas y promoviendo su integración en proyectos de energía sostenible.

En general, este proyecto sienta las bases para el desarrollo de sistemas de captación de energía cinética con aplicaciones en movilidad urbana, eficiencia energética y sostenibilidad. Su evolución podría contribuir al diseño de soluciones innovadoras para el aprovechamiento de la energía en entornos de alto tránsito.

15. BIBLIOGRAFÍA

[1] Van Ruijven, B.J., De Cian, E. and Sue Wing, I., (2019). Amplification of future energy demand growth due to climate change. *Nature communications*, *10*(1), p.2762.

[2] Scheffran, J., Felkers, M., & Froese, R. (2020). Economic growth and the global energy demand. *Green energy to sustainability: strategies for global industries*, 1-44.

[3] Soeder, D. J., & Soeder, D. J. (2021). Fossil fuels and climate change. *Fracking and the Environment: A scientific assessment of the environmental risks from hydraulic fracturing and fossil fuels*, 155-185.

[4]Panwar, N. L., Kaushik, S. C., & Kothari, S. (2011). Role of renewable energy sources in environmental protection: A review. *Renewable and sustainable energy reviews*, *15*(3), 1513-1524.

[5] Departamento Nacional de Planeación. (2022). *Plan Nacional de Desarrollo 2022-2026*. Recuperado de https://www.dnp.gov.co/plan-nacional-desarrollo/pnd-2022-2026

[6] DataSheet ADS1015. Recuperado de https://www.alldatasheet.com/view.jsp?Searchword=ads1015%20datasheet&msclkid=b37e63

[7] United Nations. (2015). *Objetivos de desarrollo sostenible*. Naciones Unidas. Recuperado de https://www.un.org/sustainabledevelopment/es/objetivos-de-desarrollo-sostenible/