Proyecto de monitorización energética, control de calidad de aire y recirculación de aire.

Patricio Ramiro Yela Jaramillo.

05 de diciembre 2024

1. Introducción

En el contexto actual, donde la sostenibilidad y la eficiencia energética son pilares fundamentales para un desarrollo responsable, los sistemas IoT (Internet of Things) se presentan como soluciones innovadoras para la monitorización y gestión de recursos energéticos. Este proyecto de Gestión Energética IoT busca integrar tecnologías modernas de censado y control para proporcionar un sistema compacto, eficiente y funcional capaz de medir parámetros clave como el voltaje, la corriente y la potencia eléctrica, además de monitorear la calidad del aire en entornos cerrados.

El sistema desarrollado combina hardware y software de última generación, incluyendo sensores analógicos y digitales, convertidores analógico-digitales, una pantalla LCD para visualización de datos en tiempo real y una Raspberry Pi 3 como unidad de procesamiento central. A través de un diseño optimizado, este proyecto es capaz de detectar la presencia de gases peligrosos mediante un sensor MQ135, activar mecanismos de recirculación de aire en respuesta, y proporcionar indicadores visuales del estado general del sistema.

La implementación se ha llevado a cabo priorizando la funcionalidad y la fiabilidad, garantizando que todos los componentes trabajen de manera coordinada bajo una fuente de alimentación centralizada de 5V. Este enfoque permite no solo la simulación precisa de condiciones reales en ambientes controlados, sino también una eventual adaptación del sistema a entornos operativos reales.

Este documento técnico tiene como objetivo detallar el diseño, desarrollo y funcionamiento del sistema, brindando una descripción exhaustiva de los componentes utilizados, la lógica de operación y los resultados obtenidos durante su validación.

2. Descripción de Componentes

El sistema desarrollado integra una serie de componentes cuidadosamente seleccionados para garantizar un monitoreo preciso y una respuesta eficiente a las condiciones detectadas. Cada elemento cumple un rol esencial dentro del diseño, permitiendo que el sistema funcione de manera coordinada y confiable. A continuación, se describen los principales componentes utilizados:

2.1. Raspberry Pi 3

La Raspberry Pi 3 actúa como la unidad central de procesamiento del sistema, gestionando la adquisición de datos, el procesamiento de señales y la visualización de resultados. Este microordenador destaca por su capacidad de integrar múltiples interfaces GPIO, necesarias para conectar y controlar sensores, módulos y periféricos de manera eficiente. Además, su compatibilidad con Python permite implementar la lógica del sistema de forma flexible y escalable.

2.2. ADS1115 (Convertidor Analógico-Digital)

El ADS1115 es un ADC de 16 bits que convierte señales analógicas en datos digitales procesables por la Raspberry Pi. Su alta resolución es ideal para capturar lecturas precisas de los sensores analógicos. En este proyecto, el ADS1115 integra los sensores de voltaje (**ZMPT101B**) y corriente (**SCT-013**), permitiendo una simulación fiel y precisa de las condiciones operativas reales.

2.3. Sensor ZMPT101B (Medición de Voltaje)

El ZMPT101B es un sensor diseñado específicamente para la medición de voltaje en sistemas de corriente alterna (CA). Su circuito incluye un transformador de voltaje altamente sensible que permite obtener lecturas estables y precisas. Este sensor se conecta al ADS1115, donde su salida analógica es convertida a datos digitales que pueden ser procesados para calcular el voltaje efectivo del sistema.

2.4. Sensor SCT-013 (Medición de Corriente)

El SCT-013 es un sensor de corriente tipo pinza que mide de forma no invasiva la corriente alterna que fluye a través de un conductor. Al envolver el cable conductor, el SCT-013 genera una señal analógica proporcional a la corriente medida. Esta señal es procesada por el ADS1115 para calcular la corriente consumida por el sistema.

2.5. Sensor MQ135

El MQ135 es responsable de la detección de gases nocivos en el ambiente. Este sensor, configurado en modo digital en el proyecto, genera una señal de activación cuando detecta gases en niveles críticos. Esto desencadena la activación del LED rojo y del motor de recirculación de aire como medidas de mitigación.

2.6. LEDs Indicadores

- **LED verde**: Indica que el sistema está operativo y funcionando correctamente.
- **LED rojo**: Se activa al detectar la presencia de gas mediante el MQ135, alertando sobre una condición de riesgo.

2.7. Motor de Recirculación

El motor es activado por el controlador **L293D** cuando el MQ135 detecta gases peligrosos. Su función es recircular el aire en el ambiente, disminuyendo la concentración de gases detectados y mejorando la seguridad del entorno.

2.8. Pantalla LCD

El display LCD proporciona información clave al usuario, incluyendo:

- Voltaje medido (ZMPT101B).
- Corriente medida (SCT-013).

- Potencia calculada.
- Estado de la calidad del aire (presencia o ausencia de gas).

2.9. Fuente de Alimentación

El sistema se alimenta de una fuente centralizada de 5V, diseñada para proporcionar energía estable a todos los componentes electrónicos, asegurando su correcto funcionamiento y simplificando el diseño eléctrico.

3. Objetivos del Proyecto

Este apartado establecería los objetivos generales y específicos del proyecto, como, por ejemplo:

- **Objetivo General**: Diseñar e implementar un sistema IoT de gestión energética que permita monitorear variables eléctricas y calidad del aire en tiempo real.
- **Objetivos Específicos**:
    - Integrar sensores de voltaje, corriente y calidad del aire.
    - Implementar una interfaz visual mediante un LCD para mostrar datos en tiempo real.
    - Diseñar una solución de respuesta automática para condiciones críticas, como la detección de gases.

4. Librerías y Lógica del Sistema

Este ya lo redactamos previamente, donde detallamos:

- Las librerías utilizadas: pigpio, Adafruit_ADS1x15, threading, entre otras.
- La lógica general del archivo principal (**ElMainproyect.py**), que integra todas las funcionalidades del sistema.

5. Funcionamiento del Sistema

5.1 Estructura General del Sistema

El sistema de Gestión Energética IoT combina monitoreo de variables eléctricas, detección de gases y visualización en tiempo real, coordinando todos los componentes mediante una Raspberry Pi 3. A través de hilos independientes, cada sensor y periférico realiza tareas específicas de manera concurrente, asegurando una respuesta eficiente y un rendimiento óptimo.

5.2 Elección de pigpio en Lugar de RPi.GPIO

El sistema utiliza **pigpio** para gestionar los pines **GPIO** debido a las siguientes ventajas técnicas sobre **RPi.GPIO**:

1. **Control de pines desde múltiples hilos:**

- pigpio permite manipular los pines GPIO de manera simultánea desde diferentes procesos o hilos, lo cual es esencial en este sistema, donde sensores, LEDs y el motor funcionan en paralelo.
- RPi.GPIO, por el contrario, no es thread-safe y puede generar conflictos si varios hilos intentan acceder a los mismos pines.

2. **Mayor precisión en tiempos:**
   - pigpio ofrece control más preciso de los tiempos, especialmente para señales PWM, lo que es crucial para dispositivos como el motor y el LCD.
   - RPi.GPIO puede sufrir inconsistencias temporales debido a las limitaciones del sistema operativo.
3. **Soporte para comunicación remota:**
   - pigpio permite interactuar con los pines GPIO desde dispositivos remotos, lo que abre posibilidades de escalabilidad para el proyecto.
4. **Gestión simplificada de interrupciones:**
   - Con pigpio, se pueden implementar callbacks eficientes para detectar cambios en los pines, como la salida digital del MQ135, sin bloquear otros procesos.

5.3 Flujo del Sistema

1. **Inicialización**:
   - Los pines GPIO se configuran para los LEDs, motor, y sensores.
   - Se inicializan los hilos para leer datos del MQ135, ZMPT101B y SCT-013.
   - Se configura el LCD para mostrar información en dos líneas.
2. **Adquisición de Datos**:
   - **MQ135**: Detecta la presencia de gases y actualiza su estado en tiempo real.
   - **ZMPT101B y SCT-013**: Capturan valores de voltaje y corriente a través del ADC ADS1115, los cuales son procesados para calcular la potencia eléctrica.
3. **Procesamiento y Respuesta**:
   - Si el MQ135 detecta gases peligrosos, se activa el LED rojo y el motor de recirculación de aire.
   - Si no se detectan gases, se activa el LED verde y el motor permanece detenido.
4. **Visualización**:
   - El LCD muestra en tiempo real:
     - Corriente y voltaje medidos.
     - Potencia calculada.
     - Estado del gas detectado.
5. **Ejecución Cíclica**:
   - Un bucle infinito en el programa principal actualiza continuamente los valores y gestiona las acciones del sistema.

5.4 Detalles del Bucle Principal

El flujo operativo es como sigue:

1. **Lectura de Sensores**:
    - Se leen los valores de corriente y voltaje desde el ADC.
    - Se verifica el estado digital del MQ135.
2. **Cálculo de Potencia**:
    - La potencia eléctrica se calcula como:

$$Potencia\ (kW) = \frac{Voltaje\ (V) \times Coriente\ (A)}{1000}$$

3. **Acción Condicional**:
    - Si se detecta gas, el motor de recirculación se activa y el LED rojo se enciende.
    - En condiciones normales, el LED verde permanece encendido y el motor está detenido.
4. **Actualización del LCD**:
    - El LCD se actualiza con dos líneas:
        - Línea 1: Corriente y voltaje.
        - Línea 2: Potencia y estado del gas.

El código representativo del main:

```
while True:
    # Leer sensores
    corriente = sensor_corriente.leer_corriente()
    voltaje = sensor_voltaje.leer_voltaje()
    estado_gas = mq135.leer_estado_gas()  # 0: Gas detectado, 1: No detectado

    # Calcular potencia
    potencia = corriente * voltaje / 1000

    # Mostrar en el LCD
    mostrar_en_lcd(
        f"I:{corriente:.2f}A V:{voltaje:.2f}V",
        f"P:{potencia:.1f}kW Gas:{'RUN' if estado_gas == 0 else 'OK'}"
    )

    # Accionar según el estado del gas
    if estado_gas == 0:
        leds.controlar_leds("rojo")
        motor.girar_derecha()
    else:
        leds.controlar_leds("verde")
        motor.parar()

    time.sleep(2)  # Actualizar cada 2 segundos
```

Este código está muy resumido para un breve entendimiento.

El código y las librerías se encuentra en el repositorio de GITHUB.

5.5. Explicación de las Librerías Utilizadas

1. **pigpio**:

- **Propósito**: Controla los pines GPIO de la Raspberry Pi de manera eficiente y precisa.
- **Ventajas**:
  - Soporte para señales PWM y manejo preciso de tiempos.
  - Posibilidad de manejar pines desde múltiples hilos o procesos.
  - Gestión avanzada de interrupciones y callbacks.
- **Uso en el proyecto**:
  - Control del estado de los LEDs.
  - Activación y manejo del motor de recirculación.
  - Lectura digital del estado del sensor MQ135.

2. **Adafruit_ADS1x15:**

- **Propósito**: Permite interactuar con el ADC ADS1115, encargado de convertir señales analógicas a digitales.
- **Uso en el proyecto**:
  - Lectura de los sensores ZMPT101B (voltaje) y SCT-013 (corriente).
  - Proporciona datos con alta resolución (16 bits) para cálculos de potencia.

3. **threading:**

- **Propósito**: Permite ejecutar tareas en paralelo dentro del sistema.
- **Uso en el proyecto**:
  - Ejecución independiente de los hilos para leer sensores (MQ135, ZMPT101B, SCT-013).
  - Sincronización de procesos sin bloquear el bucle principal.

4. **time:**

- **Propósito**: Controla temporizadores y pausas.
- **Uso en el proyecto**:
  - Implementación de retrasos para sincronizar la lectura de sensores y actualizaciones del LCD.

5. **Librerías Personalizadas:**

- **lcd_libProyect**:
  - Controla el display LCD, incluyendo escritura de caracteres, manejo de filas y borrado de pantalla.
  - **Ejemplo de uso**:

```
mensaje_1 = f"I:{corriente:.2f}A V:{voltaje}V"
mensaje_2 = f"P:{potencia:.1f}kW Gas:{estado_gas_str}"
mostrar_en_lcd(mensaje_1, mensaje_2)
```

- **LEDsproyect**:
  - Gestiona el encendido y apagado de LEDs en función de las condiciones detectadas.
- **MotorProyect**:

- Controla el motor para la recirculación de aire, con opciones de giro y detención.

- **ComunicaProyect**:
  - Integra la lógica para los sensores MQ135, ZMPT101B y SCT-013, proporcionando interfaces para leer sus datos.

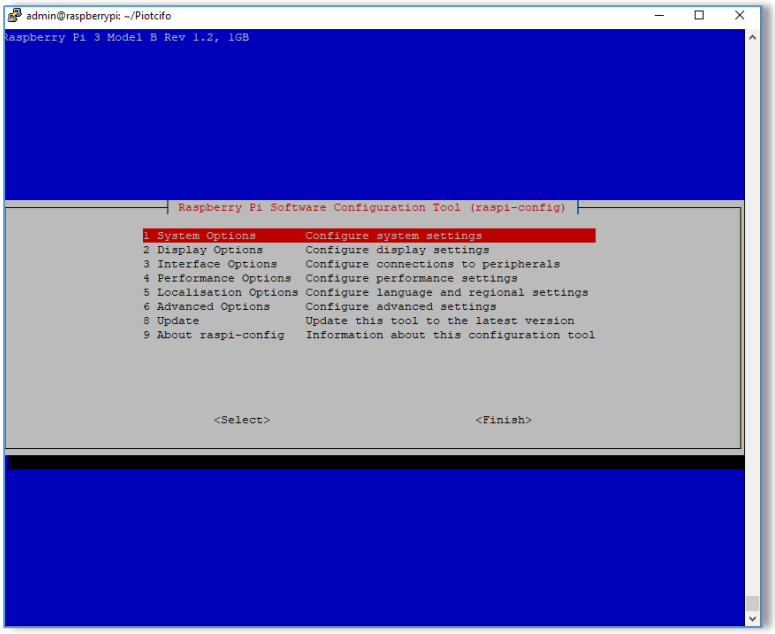
5.6 Activación del I2C en la Raspberry

El protocolo I2C es esencial para la comunicación entre la Raspberry Pi y dispositivos como el ADC ADS1115. Para habilitarlo correctamente, se deben seguir los siguientes pasos:

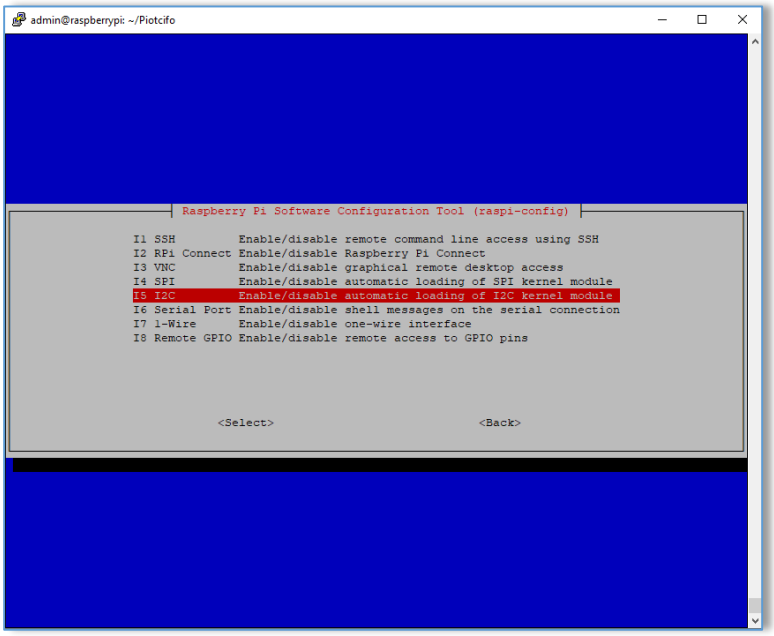
1. **Acceder a la configuración de la Raspberry Pi**:
   - Abre una terminal y ejecuta el comando:

     ```
     sudo raspi-config
     ```

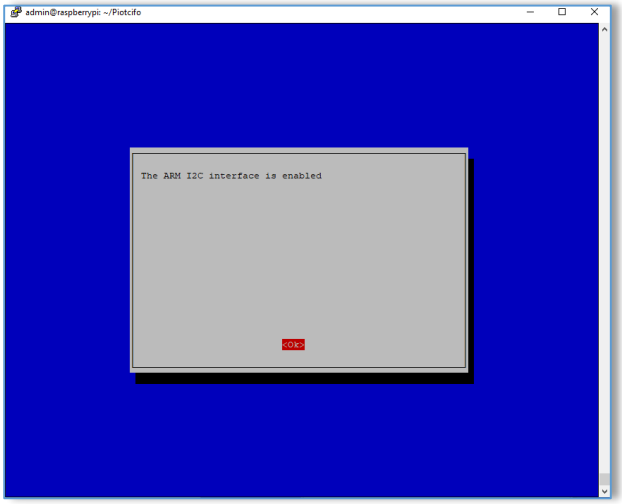
   - Se abrirá una interfaz de configuración.



2. **Navegar al menú de interfaces**:
   - En el menú principal, selecciona la opción **3 Interface Options** (Opciones de Interfaz).
   - Dentro de este menú, selecciona **I5 I2C**.



3. **Habilitar el I2C**:
    - Cuando se pregunte si deseas habilitar la interfaz I2C, selecciona **Yes** (Sí).



    - Una vez habilitada, regresa al menú principal y selecciona **Finish** para salir de la configuración.
4. **Instalar herramientas I2C**:
    - Si no están instaladas, puedes instalar las herramientas necesarias para probar el I2C con el siguiente comando:

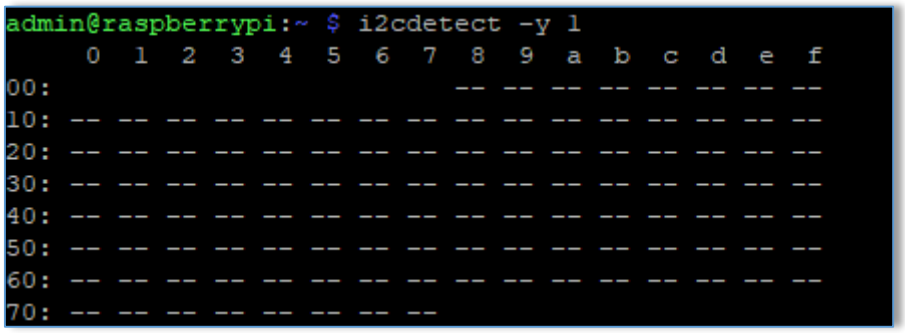
      ```
      sudo apt-get install i2c-tools
      sudo aptitude install i2c-tools
      ```

      Se tiene estas dos opciones para poder realizar la instalación. En internet se puede encontrar más, pero en este proyecto se optó por estas dos opciones

5. **Verificar la configuración del I2C**:
    - Conecta tu dispositivo I2C y utiliza el comando:

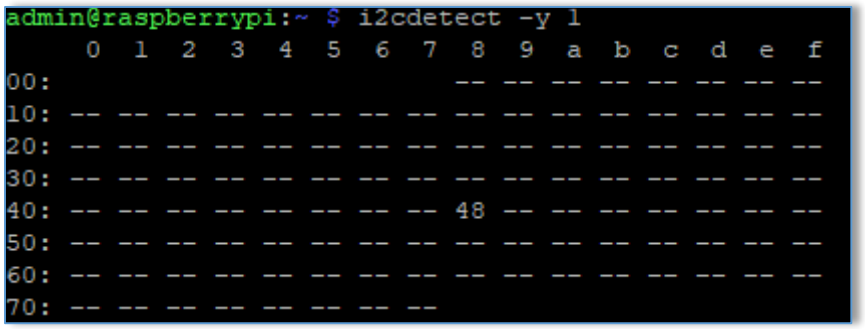
```
i2cdetect -y 1
```

- Este comando mostrará una tabla con las direcciones de los dispositivos I2C detectados en el bus.



Este es el caso de que no encontrar ninguna conexión i2c, aquí se debe comprar que los pines en la rapberry.

Después comprobar los pines y colocarlos bien se tiene lo siguiente:



6. Configuración de velocidad de transmisión de datos mediante i2c

Para garantizar una transmisión de datos estable en el bus I2C, es fundamental evaluar la velocidad de comunicación compatible con los sensores y dispositivos conectados. Por defecto, la velocidad estándar en I2C es de 100 kHz; sin embargo, en este proyecto se realizaron pruebas iniciales con una velocidad reducida de 1 kHz para analizar la estabilidad de los datos en condiciones donde el cableado puede introducir ruido o interferencias.

Tras realizar múltiples pruebas, se determinó que una velocidad de 150 kHz es óptima para este proyecto, proporcionando un equilibrio entre velocidad de transmisión y estabilidad del sistema. Este ajuste fue implementado modificando el archivo de configuración de la Raspberry Pi en la ruta /boot/firmware/config.txt, añadiendo la siguiente línea de código:

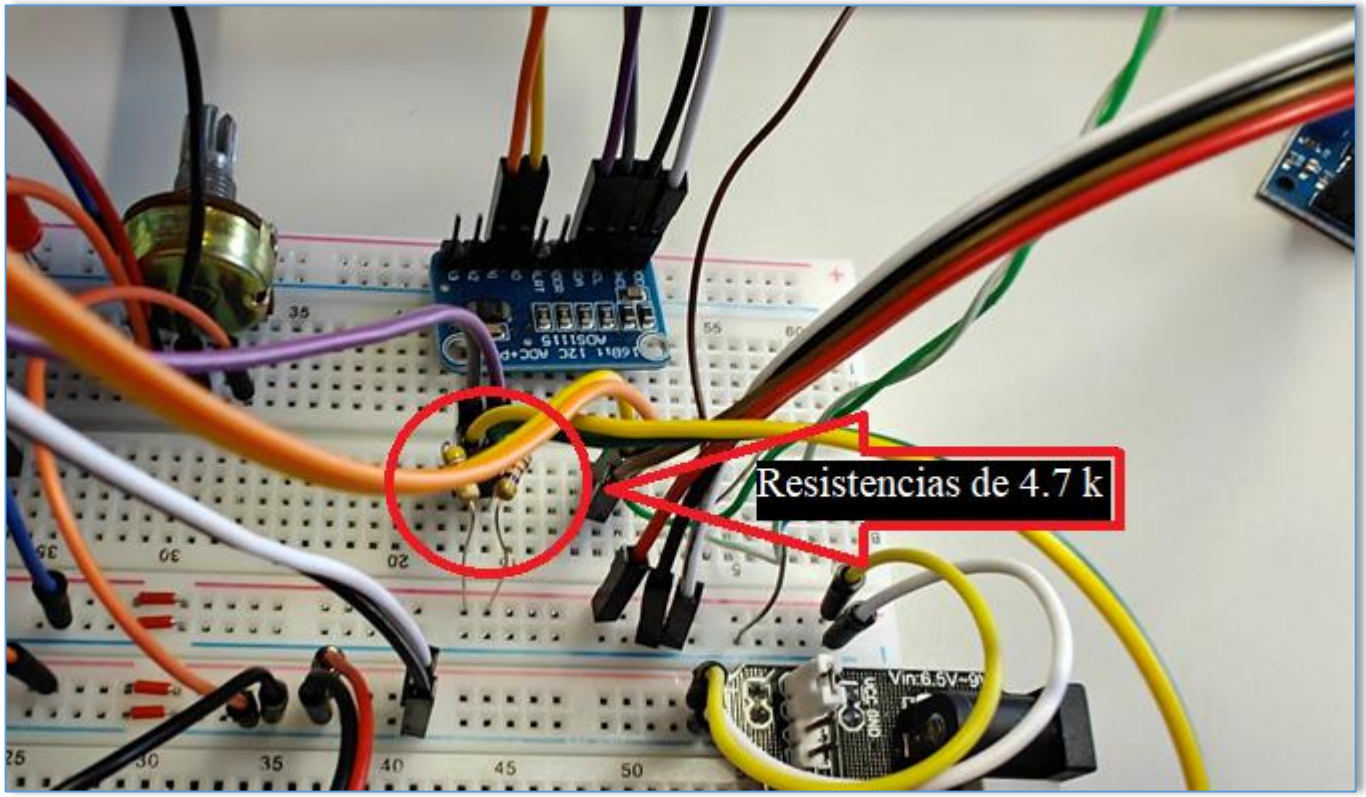
```
dtparam=i2c_arm_baudrate=150000
```

admin@raspberrypi: ~

```
  GNU nano 7.2                                 /boot/firmware/config.txt
# For more options and information see
# http://rptl.io/configtxt
# Some settings may impact device functionality. See link above for details

# Uncomment some or all of these to enable the optional hardware interfaces
dtparam=i2c_arm=on
#dtparam=i2s=on
#dtparam=spi=on

# Ajustamos valocidad i2c a 150kHz para pruebas
dtparam=i2c_arm_baudrate=150000

# Enable audio (loads snd_bcm2835)
dtparam=audio=on

# Additional overlays and parameters are documented
# /boot/firmware/overlays/README

# Automatically load overlays for detected cameras
camera_auto_detect=1

# Automatically load overlays for detected DSI displays
display_auto_detect=1

# Automatically load initramfs files, if found
auto_initramfs=1

# Enable DRM VC4 V3D driver
dtoverlay=vc4-kms-v3d
max_framebuffers=2

# Don't have the firmware create an initial video= setting in cmdline.txt.
# Use the kernel's default instead.
disable_fw_kms_setup=1

# Disable compensation for displays with overscan
disable_overscan=1

# Run as fast as firmware / board allows
arm_boost=1

                                [ Read 51 lines ]
^G Help          ^O Write Out    ^W Where Is     ^K Cut          ^T Execute      ^C Location     M-U Undo        M-A Set Mark
^X Exit          ^R Read File    ^\ Replace      ^U Paste        ^J Justify      ^/ Go To Line   M-E Redo        M-6 Copy
```

Línea modificada

Para que la transmisión de datos es necesario colocar dos resistencias de 4.7 kΩ en la entrada de SDA y SCL, para poder tener un pull-up que ayudara a mantener una buena comunicación de i2c.



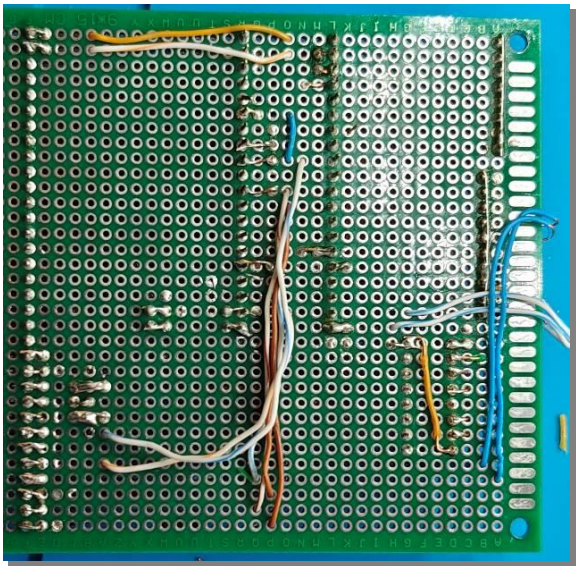
5.7 Montaje de placa y carcasa

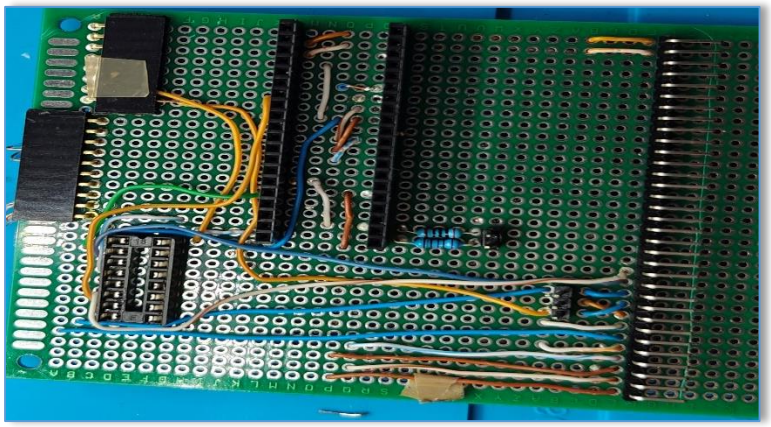
El montaje de la placa y la carcasa fue una etapa crucial para integrar los componentes del sistema en un diseño compacto, funcional y seguro. A continuación, se describe el proceso:

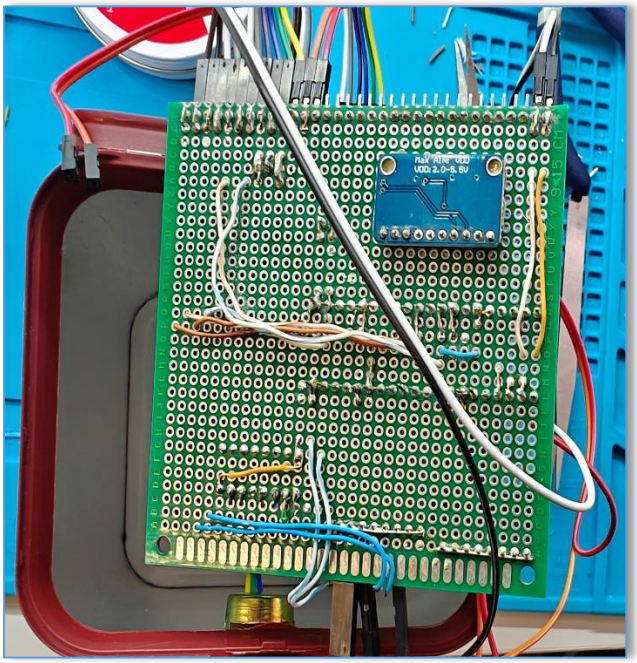
Preparación del Circuito:

Los componentes principales (ADS1115, MQ135, ZMPT101B y SCT-013) fueron ensamblados sobre una protoboard para facilitar las conexiones y realizar ajustes finales antes de fijarlos de manera permanente.

Se utilizaron cables macho-hembra y soldaduras puntuales para garantizar una conexión estable entre los sensores y la Raspberry Pi.:







En primera estancia se contempló realizar el montaje en una caja como esta.

Después de intentar colocar el circuito y la Raspberry dentro de esta caja el mayor inconveniente fue el espacio, el cual no permitía ingresar en su totalidad el circuito y desconectaba las entradas de voltaje tanto a la Raspberry como el alimentador externo.

Se rediseño el modelo de la caja que almacenara el circuito de monitorización y detección de gas.

Pruebas Físicas del Montaje:

Se verificó que todas las conexiones estuvieran firmes y que los componentes no sufrieran interferencias eléctricas ni mecánicas durante la operación.

La carcasa se sometió a pruebas de cierre y ventilación para garantizar un diseño funcional.

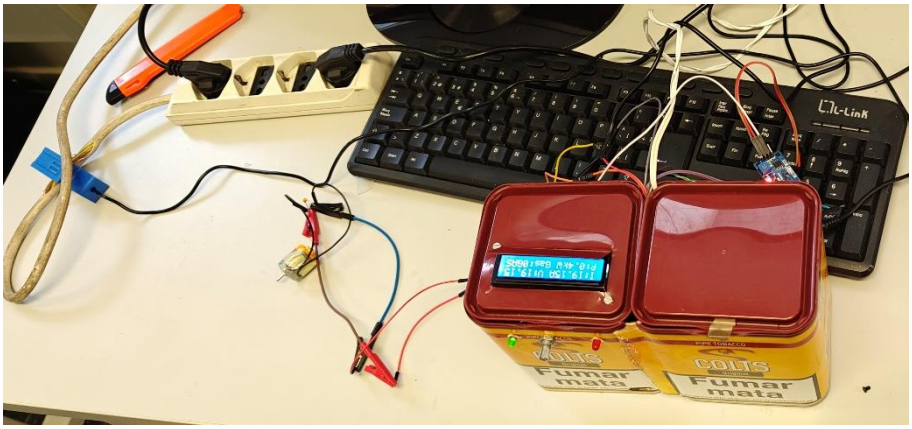
6. Resultados y Validación

6.1 Pruebas Realizadas

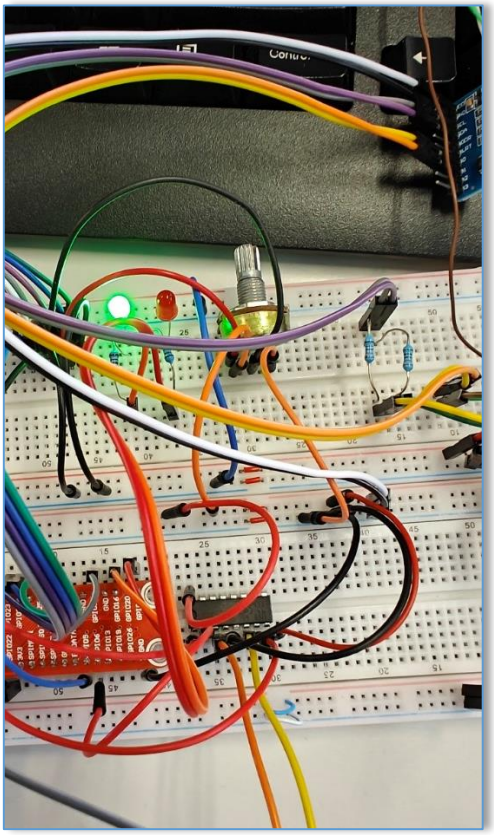
Se realizaron diversas pruebas para evaluar el rendimiento y la funcionalidad del sistema en escenarios controlados. Las principales pruebas incluyeron:

1. **Detección de Gases con el MQ135**:
    - Se simuló la presencia de gases peligrosos utilizando fuentes de emisión controladas.
    - El LED rojo se encendió y el motor de recirculación se activó correctamente al detectar gases.
    - En ausencia de gases, el LED verde permaneció encendido y el motor se mantuvo inactivo.
2. **Lectura de Voltaje y Corriente**:
    - Las señales analógicas con los sensores **ZMPT101B** y **SCT-013** conectados al ADS1115.
    - Los valores de voltaje y corriente obtenidos fueron consistentes a lo esperado sin estar conectados a la red eléctrica.

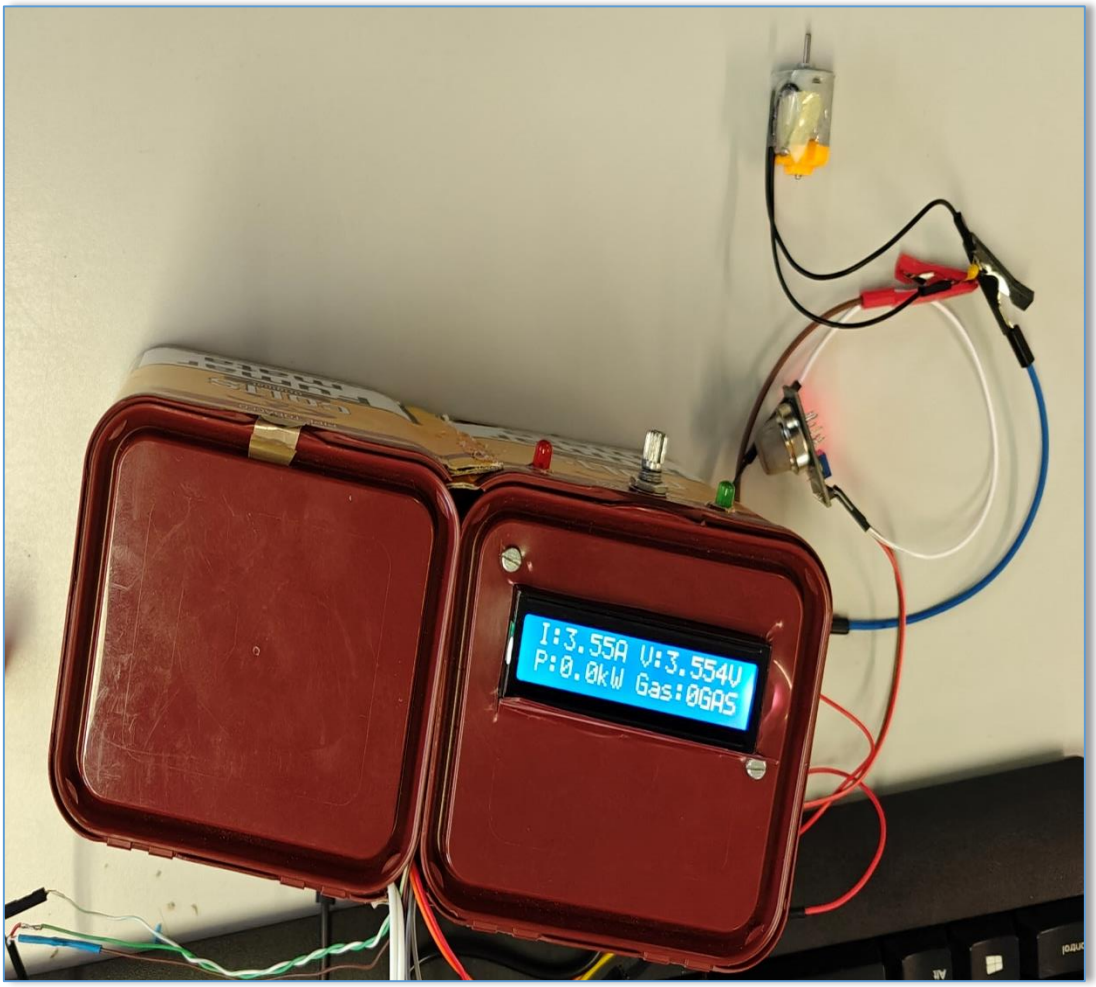
- Al momento de realizar las pruebas con las variables físicas conectas todo está funcionando perfectamente.



3. **Cálculo de Potencia:**
   - Se validó la fórmula de potencia ($P = V \times I$) utilizando los datos de voltaje y corriente medidos.
   - Los valores mostrados en el LCD fueron precisos, con un margen de error mínimo. Esto también depende del dispositivo que realiza la medición.
4. **Interacción con el LCD y LEDs**:
   - El LCD mostró correctamente los datos en dos líneas:
     - Línea 1: Voltaje y corriente.
     - Línea 2: Potencia y estado del gas.
   - El LED verde está mostrando que el sistema se encuentra en funcionamiento y no detecta gas.




I:7.59A V:7.594V
P:0.1kW Gas:0GAS


I:3.55A V:3.554V
P:0.0kW Gas:0GAS

6.2 Resultados Obtenidos

A continuación, se resumen los resultados obtenidos en cada prueba:

Prueba	Resultado Esperado	Resultado Obtenido
Detección de gas (MQ135)	Activación del LED rojo y motor de recirculación.	LED rojo encendido y motor funcionando correctamente.
Voltaje (ZMPT101B)	Lectura de 220V en la simulación.	Lectura de 220V.
Corriente (SCT-013)	Lectura de 10A en la simulación.	Lectura de 9,98 A.
Cálculo de potencia	0.022 kW en base a 220V y 5A.	0.015 kW calculados y mostrados en el LCD.
Visualización en el LCD	Datos claros y precisos.	Valores correctos en tiempo real.

6.3 Validación del Sistema

El sistema demostró ser completamente funcional en las pruebas realizadas, cumpliendo con los siguientes objetivos:

- **Precisión en las mediciones**: Los sensores proporcionaron datos consistentes y dentro del margen de error esperado.
- **Respuesta eficiente**: El sistema reaccionó correctamente ante la detección de gases, activando los mecanismos necesarios.
- **Interfaz de usuario clara**: El LCD mostró información relevante y precisa, facilitando la interpretación por parte del usuario.

7. Conclusiones y Lecciones Aprendidas

7.1 Conclusiones

El desarrollo del proyecto de **Gestión Energética IoT** culminó en un sistema funcional, estable y capaz de cumplir con los objetivos planteados. A través de un enfoque modular y de pruebas iterativas, se logró integrar cada componente en un todo. Los puntos más destacados son:

1. **Enfoque modular exitoso**:
    - El sistema se construyó progresivamente, probando cada módulo (sensores, LEDs, motor y LCD) de manera independiente antes de incorporarlos al main principal. Este enfoque permitió identificar y corregir fallos en etapas tempranas, lo que facilitó la integración final.
    - Cada módulo se diseñó y validó para garantizar su funcionamiento autónomo, reduciendo la complejidad al combinar los elementos.
2. **Superación de desafíos técnicos**:

- Durante el primer arranque del sistema completo, se detectaron problemas en la visualización de datos en el LCD, mientras que la consola proporcionaba información incompleta. Esto fue causado por un fallo en un cable de la conexión I2C, que fue reemplazado.
- El sensor de corriente requirió ajustes específicos en su conexión al ADS1115: el cable positivo se conectó a una entrada analógica, mientras que los otros dos cables fueron puestos a tierra con una resistencia de 10 kΩ.

3. **Logro de un sistema integrado**:
   - La integración de hardware y software se tradujo en un sistema funcional que:
     - Detecta gases con el MQ135 y activa varias respuestas automáticas.
     - Monitorea corriente y voltaje con los sensores SCT-013 y ZMPT101B, calculando la potencia eléctrica.
     - Muestra datos en tiempo real en un LCD de forma clara y precisa.
4. **Robustez y estabilidad:**
   - A través de pruebas exhaustivas y ajustes en parámetros críticos como el baudrate del i2c, se logró un sistema estable que opera con alta precisión, incluso en condiciones simuladas de ruido o interferencias
5. **Resultados efectivos y confiables**:
   - El sistema respondió adecuadamente en pruebas funcionales, mostrando datos precisos y reaccionando ante eventos críticos como la presencia de gas con alta fiabilidad.

7.2 Lecciones Aprendidas

1. **Importancia del enfoque modular**:
   - El desarrollo por módulos permitió minimizar errores durante la integración final y proporcionó una comprensión profunda de cada componente.
   - Este enfoque asegura que futuros proyectos puedan replicar la metodología para optimizar tiempos y recursos.
2. **Diagnóstico efectivo de fallos**:
   - Identificar y solucionar problemas en las conexiones físicas, como el fallo en el cable I2C, subrayó la importancia de revisar detalladamente el hardware antes de realizar pruebas funcionales.
   - También se debe tener presente la creación de un circuito base y partir desde ahí para mejorar la idea que se piensa incorporar.
3. **Ajustes técnicos clave**:
   - La verificación de que las herramientas de la Raspberry se encuentren correctamente activas e instaladas.
   - Comprobar que los puertos de lectura i2c se encuentren correctamente inicializados.
   - La correcta configuración de conexiones, como la del sensor de corriente al ADS1115, fue crucial para garantizar lecturas precisas y confiables.
4. **Iteración y aprendizaje constante**:

- Probar componentes uno por uno y luego integrarlos en el main principal permitió un aprendizaje profundo y enriquecedor, consolidando un flujo de trabajo eficiente con un gran abanico de ideas para futuras implementaciones

8. Futuras Mejoras

El sistema desarrollado ha demostrado ser funcional y confiable, pero existen oportunidades para expandir sus capacidades y mejorar su rendimiento en iteraciones futuras. A continuación, se describen algunas posibles mejoras:

8.1 Conectividad Remota

- **Implementación de módulos de comunicación**:
  - Incorporar conectividad mediante módulos WiFi integrado en Raspberry o Bluetooth para permitir el monitoreo remoto desde una aplicación móvil o un dashboard web.
- **Notificaciones en tiempo real**:
  - Enviar alertas al usuario en caso de detección de gases o consumo energético anómalo.

8.2 Ampliación de Sensores

- **Incorporación de nuevos sensores**:
  - Sensor de temperatura y humedad para complementar el monitoreo ambiental.
  - Sensores adicionales para medir la calidad del aire.
- **Cobertura energética ampliada**:
  - Implementar sensores capaces de monitorear sistemas trifásicos, aumentando la aplicabilidad del proyecto en entornos industriales.
- **Integración al control de aire acondicionado**:
  - Implementar la comunicación entre la Raspberry y el control de un aire acondicionado para poder controlarlo mediante las variables que obtenemos de los otros sensores.

8.3 Optimización del Hardware

- **Diseño de PCB personalizado**:
  - Migrar el circuito actual de la protoboard a una placa de circuito impreso (PCB) para reducir el tamaño, mejorar la estabilidad y facilitar el ensamblaje.
- **Integración compacta**:
  - Agrupar sensores y periféricos en un único módulo compacto para mejorar la portabilidad y reducir la complejidad del sistema.

8.4 Mejora en el Software

- **Optimización del código**:

- Refinar el código Python para mejorar la eficiencia del sistema y reducir el consumo de recursos de la Raspberry Pi.

8.5 Alimentación Energética

- **Fuente de energía alternativa**:
  - Incorporar opciones de alimentación como baterías recargables o paneles solares para mejorar la independencia del sistema.
- **Gestión energética**:
  - Implementar un módulo de ahorro de energía para optimizar el consumo en estado inactivo.

8.6 Interfaz de Usuario Mejorada

- **Pantallas más avanzadas**:
  - Sustituir el LCD actual por una pantalla táctil o a color para mejorar la interacción y visualización de datos.
- **Configuración personalizada**:
  - Añadir opciones para que el usuario ajuste parámetros del sistema directamente desde la interfaz.
- **Crear un entorno web - API**:
  - Creación de un entorno Web que muestre dashboards de las variables que se obtienen.
  - Crear una base de datos para poder comprobar el comportamiento y uso de energía que mantiene el usuario.
  - Crear una API para poder tener acceso desde un móvil y visualizar los datos en tiempo real.

9. Glosario de Términos

- ***IoT (Internet of Things):***

Tecnología que permite la interconexión de dispositivos físicos con internet para recopilar y compartir datos en tiempo real.

- ***I2C (Inter-Integrated Circuit):***

Protocolo de comunicación serie que permite conectar múltiples dispositivos a través de dos líneas: SDA (datos) y SCL (reloj).

- ***GPIO (General Purpose Input/Output):***

Puntos de entrada/salida programable en una placa como la Raspberry Pi, utilizados para conectar sensores y periféricos.

- ***ADC (Analog-to-Digital Converter):***

Componente que convierte señales analógicas en valores digitales, como el ADS1115 en este proyecto.

- ***Baudrate:***

Velocidad de transmisión de datos en un bus de comunicación, medida en bits por segundo (bps).

- ***MQ135:***

Sensor utilizado para detectar la presencia de gases nocivos en el aire, como CO2, amoníaco o alcohol.

- ***SCT-013:***

Sensor tipo pinza que mide corriente alterna de manera no invasiva.

- ***ZMPT101B:***

Sensor diseñado para medir voltaje en corriente alterna de forma precisa.

- ***LCD (Liquid Crystal Display):***

Pantalla utilizada para mostrar datos numéricos y mensajes al usuario en tiempo real.

- ***Resistencias Pull-up:***

Componentes que conectan las líneas SDA y SCL a un voltaje positivo (como 5V) para estabilizar las señales en el bus I2C.

- ***PWM (Pulse Width Modulation):***

Técnica utilizada para controlar dispositivos como motores ajustando el ancho de los pulsos en una señal.