Universidad San Carlos de Guatemala
Facultad de Ingeniería
Escuela ciencias y Sistemas
Arquitectura de Computadores y Ensambladores 1

PRACTICA 1

Smart Home

No.	Nombre	Carnet
1	Kevin Manuel Veliz Galvez	201901441
2	Mario Alexander Ruano García	201902382
3	Byron Enrique Rumpich Sal	201907769
4	Iris Carolina Paz Guzman	202101728
5	Oscar David Padilla Vásquez	202103250

INTRODUCCIÓN

En este proyecto, se desarrollará un sistema de automatización del hogar utilizando la Raspberry Pi para convertir las viviendas de la residencial "Pinos Altos" en Smart Homes. Se emplearán tecnologías avanzadas como sensores de temperatura, detectores de fuego, botones de acceso y motores para simular el aire acondicionado, todos controlados mediante programación en Python.

El diseño y prueba inicial de los circuitos electrónicos se realizará en el software de simulación Proteus, asegurando la correcta funcionalidad antes de la implementación física. Los pines GPIO de la Raspberry Pi permitirán la comunicación directa con los sensores y actuadores, mientras que una pantalla LCD de 16x2 mostrará información relevante a los usuarios, como el estado de acceso y la temperatura del hogar.

El sistema integrará control manual y automático para dispositivos específicos, mejorando la comodidad y seguridad del hogar. El enfoque se centrará en mantener la seguridad del sistema de control de acceso y en automatizar inteligentemente los dispositivos del hogar, como el aire acondicionado, en función de las condiciones ambientales.
No se utilizarán controladores externos o hardware avanzado más allá de los componentes básicos disponibles.

OBJETIVOS

Objetivos Generales

- Se creó un sistema sencillo de domótica que integró varios dispositivos y sensores, mejorando la comodidad y seguridad de las casas en la residencial “Pinos Altos”.

- Se interactuó con la Raspberry Pi zero de manera de conocer su funcionamiento para la familiarización a usarla para automatizar cosas en casa.

Objetivos Específicos

- Se diseñó y simuló un circuito en Proteus que incluyó dispositivos de automatización del hogar, como botones de acceso, sensores de temperatura y detectores de fuego.

- Se programa la Raspberry Pi zero en Python para controlar estos dispositivos, asegurando que el sistema funcionara bien y cumpliera con las necesidades del hogar, como el control de acceso y la gestión del aire acondicionado.

DESCRIPCIÓN

Se diseñó un sistema de automatización del hogar utilizando la Raspberry Pi, integrando sensores de temperatura, detectores de fuego, botones de acceso y un motor para simular el aire acondicionado.

Primero, se creó y simuló el circuito en Proteus para asegurar su funcionalidad antes de la implementación física. Luego, se programó la Raspberry Pi en Python para controlar estos dispositivos, permitiendo tanto control manual como automático.

Se configuraron los pines GPIO para la comunicación con los sensores y actuadores, y una pantalla LCD de 16x2 se utilizó para mostrar información relevante, como el estado de acceso y la temperatura del hogar.

El enfoque se centró en mejorar la seguridad y comodidad de las viviendas, con un sistema de control de acceso seguro y automatización inteligente basada en las condiciones ambientales.

En las próximas fases, se continuará optimizando el sistema y se implementarán pruebas adicionales para asegurar su estabilidad y eficiencia.

Contenido

En el contexto del proyecto de automatización del hogar para la residencial "Pinos Altos", se emplean varios componentes clave que contribuyen al funcionamiento eficiente y seguro del sistema. A continuación, se presentan definiciones y explicaciones de los componentes utilizados, con un enfoque orientado a la arquitectura de computadoras y sistemas embebidos.

Sensor de Flama
Un sensor de flama es un dispositivo electrónico diseñado para detectar la presencia de fuego o llamas. Estos sensores generalmente utilizan fotodiodos o fototransistores para capturar la radiación infrarroja emitida por una llama. La señal detectada es convertida en una señal eléctrica que puede ser procesada por un microcontrolador o una placa de desarrollo como la Raspberry Pi. En el contexto del proyecto, el sensor de flama permite al sistema automatizado detectar condiciones de incendio, mejorando así la seguridad del hogar.

Arquitectura de Computadoras:
El sensor de flama envía señales digitales o analógicas a través de pines GPIO, que son gestionadas por el procesador de la Raspberry Pi. Estas señales se interpretan mediante el software para activar o desactivar otros dispositivos, como el buzzer o la luz de advertencia.

Buzzer
Un buzzer es un dispositivo que emite sonido cuando recibe una señal eléctrica. Existen dos tipos principales de buzzers: piezoeléctricos y electromagnéticos. En el contexto del proyecto, el buzzer piezoeléctrico se utiliza para emitir una alarma sonora cuando el sensor de flama detecta una llama. El buzzer convierte las señales eléctricas en ondas sonoras, proporcionando una indicación audible de una condición crítica.

Arquitectura de Computadoras: El buzzer se controla mediante los pines GPIO de la Raspberry Pi. Una señal lógica de alto o bajo enviada desde la Raspberry Pi activa o desactiva el buzzer, generando sonidos de alerta que se sincronizan con las señales de los sensores y otros componentes del sistema.

MCP3208

El MCP3208 es un convertidor analógico a digital (ADC) de 12 bits con 8 canales de entrada. Este componente es fundamental para la conversión de señales analógicas (como las del sensor de temperatura) en valores digitales que pueden ser procesados por la Raspberry Pi. Permite la lectura precisa de las señales analógicas desde varios sensores, facilitando la integración de datos analógicos en sistemas digitales.

Arquitectura de Computadoras: El MCP3208 se comunica con la Raspberry Pi a través del protocolo SPI (Serial Peripheral Interface). La Raspberry Pi envía comandos de lectura y recibe datos digitales del MCP3208, lo que permite el monitoreo de variables físicas como la temperatura del ambiente. Este ADC es crucial para la conversión de señales en el proceso de adquisición de datos en el sistema.

L293D

El *L293D es un controlador de motor de puente H que permite el control bidireccional de motores de corriente continua. Este componente es esencial para el manejo de actuadores como el motor que simula el aire acondicionado en el proyecto. El L293D puede manejar corrientes relativamente altas y proporciona protección contra sobrecargas y cortocircuitos.

Arquitectura de Computadoras: El L293D recibe señales de control desde los pines GPIO de la Raspberry Pi, que determinan la dirección y la velocidad del motor. El controlador de motor gestiona las señales eléctricas para dirigir el motor, permitiendo la automatización del control del aire acondicionado basado en las lecturas de temperatura. Esto demuestra cómo un controlador de motor se integra en un sistema embebido para realizar tareas físicas de manera eficiente.

Código Comentado

```python
import RPi.GPIO as GPIO
from time import sleep, time
import spidev

# Variables para el funcionamiento del controlador de la temperatura
spi = spidev.SpiDev()
spi.open(0, 0)

# Puerto de la temperatura CH0
channel_temp = 0
```

Se importan las bibliotecas necesarias y se configura el puerto SPI para la lectura de temperatura desde un sensor.

```python
# Función para leer el puerto
def ReadChannel(channel):
    adc = spi.xfer2([1, (8 + channel) << 4, 0])
    data = ((adc[1] & 3) << 8) + adc[2]
    return data

# Función para convertir el voltaje recibido a temperatura
def ConvertTemp(data, places):
    temp = ((data * 330) / float(1023))
    temp = round(temp, places)
    return temp
```

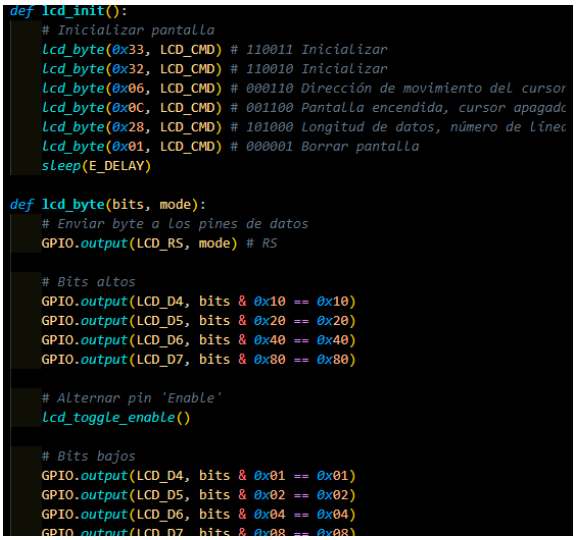
Estas funciones permiten leer los datos del sensor de temperatura y convertirlos en grados Celsius.

```python
# Configuración de los pines
GPIO.setmode(GPIO.BOARD)

# Pines para el LCD
LCD_RS = 15
LCD_E = 16
LCD_D4 = 18
LCD_D5 = 22
LCD_D6 = 29
LCD_D7 = 31

# Pines de luz
boton_pin = 32  # en donde está conectado el botón
led_pin = 33  # en donde está conectado el LED

# Pines para la contraseña
BTN_1 = 7   # SECCION B
BTN_2 = 11  # CARACTER 3
BTN_3 = 12  # GRUPO 3
BTN_4 = 13  # TECLA ENTER

# PIN PARA FUNCIONAMIENTO DEL MOTOR
MOTOR_1 = 36
MOTOR_2 = 35
```

Configuración de los pines GPIO para el LCD, botones, LEDs, motor y sensores.

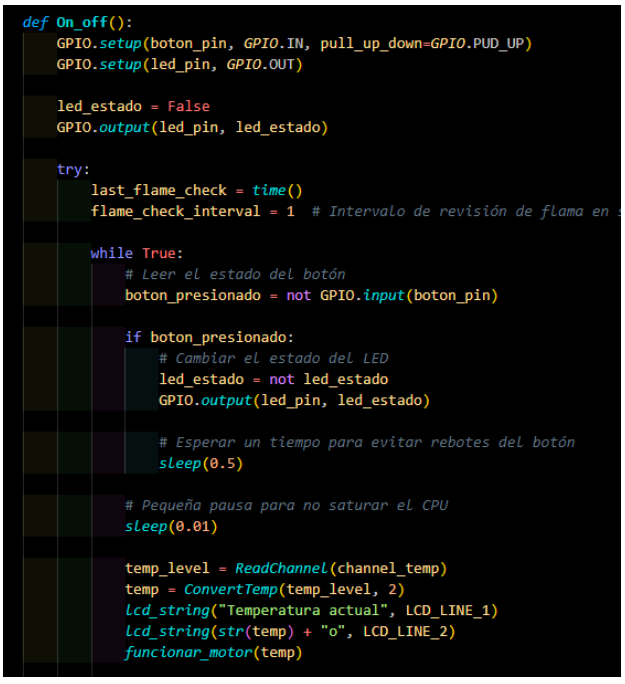
```python
def flame_detection():
    if GPIO.input(flame_Sensor):
        lcd_string("Flame Detected  ", LCD_LINE_1)
        GPIO.output(buzzer, True)
        GPIO.output(red_light, True)
    else:
        lcd_string("Flame Not Detected  ", LCD_LINE_1)
        GPIO.output(buzzer, False)
        GPIO.output(red_light, False)
```

Verifica si se detecta una flama y, en caso afirmativo, activa el buzzer y la luz roja. En caso contrario, los apaga.



```python
def lcd_init():
    # Inicializar pantalla
    lcd_byte(0x33, LCD_CMD) # 110011 Inicializar
    lcd_byte(0x32, LCD_CMD) # 110010 Inicializar
    lcd_byte(0x06, LCD_CMD) # 000110 Dirección de movimiento del cursor
    lcd_byte(0x0C, LCD_CMD) # 001100 Pantalla encendida, cursor apagado
    lcd_byte(0x28, LCD_CMD) # 101000 Longitud de datos, número de líneas
    lcd_byte(0x01, LCD_CMD) # 000001 Borrar pantalla
    sleep(E_DELAY)

def lcd_byte(bits, mode):
    # Enviar byte a los pines de datos
    GPIO.output(LCD_RS, mode) # RS

    # Bits altos
    GPIO.output(LCD_D4, bits & 0x10 == 0x10)
    GPIO.output(LCD_D5, bits & 0x20 == 0x20)
    GPIO.output(LCD_D6, bits & 0x40 == 0x40)
    GPIO.output(LCD_D7, bits & 0x80 == 0x80)

    # Alternar pin 'Enable'
    lcd_toggle_enable()

    # Bits bajos
    GPIO.output(LCD_D4, bits & 0x01 == 0x01)
    GPIO.output(LCD_D5, bits & 0x02 == 0x02)
    GPIO.output(LCD_D6, bits & 0x04 == 0x04)
    GPIO.output(LCD_D7, bits & 0x08 == 0x08)
```

Inicializa y controla la pantalla LCD para mostrar mensajes.



```python
def On_off():
    GPIO.setup(boton_pin, GPIO.IN, pull_up_down=GPIO.PUD_UP)
    GPIO.setup(led_pin, GPIO.OUT)

    led_estado = False
    GPIO.output(led_pin, led_estado)

    try:
        last_flame_check = time()
        flame_check_interval = 1  # Intervalo de revisión de flama en s

        while True:
            # Leer el estado del botón
            boton_presionado = not GPIO.input(boton_pin)

            if boton_presionado:
                # Cambiar el estado del LED
                led_estado = not led_estado
                GPIO.output(led_pin, led_estado)

                # Esperar un tiempo para evitar rebotes del botón
                sleep(0.5)

            # Pequeña pausa para no saturar el CPU
            sleep(0.01)

            temp_level = ReadChannel(channel_temp)
            temp = ConvertTemp(temp_level, 2)
            lcd_string("Temperatura actual", LCD_LINE_1)
            lcd_string(str(temp) + "o", LCD_LINE_2)
            funcionar_motor(temp)
```

Controla el estado del LED, lee la temperatura, actualiza la pantalla LCD, y verifica la presencia de flamas en intervalos regulares.

```python
def main():
    lcd_init()
    lcd_string("BIENVENIDO", LCD_LINE_1)
    lcd_string("INGRESE PATRON", LCD_LINE_2)

    try:
        patron = 'B03'
        patron_guardado = ''

        while True:
            if GPIO.input(BTN_1):
                print("Boton 1 presionado")
                patron_guardado += 'B'
                lcd_string(patron_guardado, LCD_LINE_2)
                sleep(0.5)

            if GPIO.input(BTN_2):
                print("Boton 2 presionado")
                patron_guardado += '0'
                lcd_string(patron_guardado, LCD_LINE_2)
                sleep(0.5)

            if GPIO.input(BTN_3):
                print("Boton 3 presionado")
                patron_guardado += '3'
                lcd_string(patron_guardado, LCD_LINE_2)
                sleep(0.5)

            if GPIO.input(BTN_4):
                print("Boton 4 presionado")
                lcd_string(patron_guardado, LCD_LINE_1)
                sleep(0.1)
                if patron_guardado == patron:
                    print("Patron correcto")
```

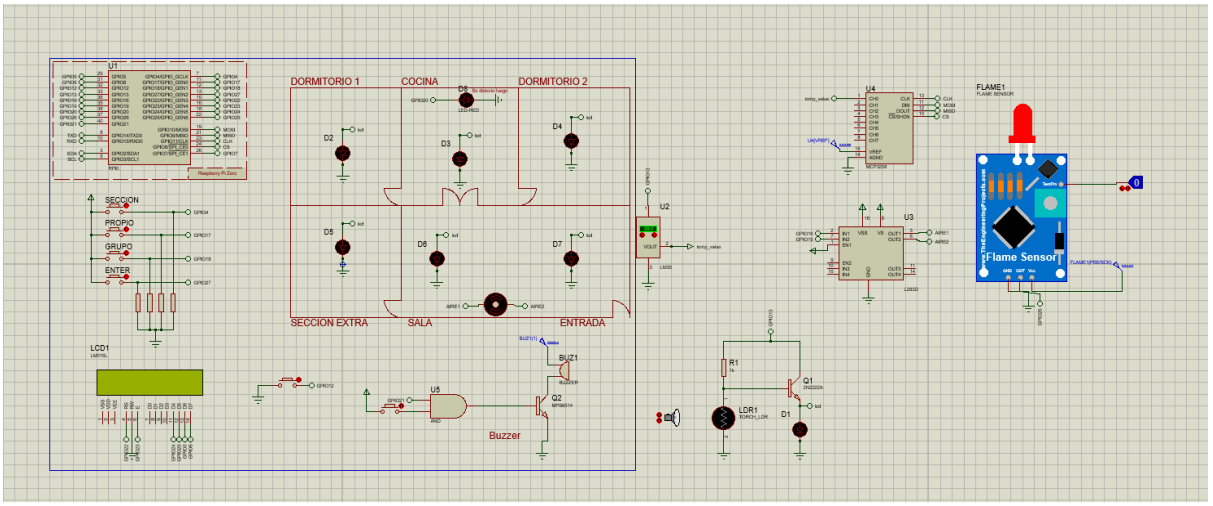
Permite al usuario ingresar una contraseña usando botones. Una vez ingresada correctamente, inicia la función `On_off`.

```python
def funcionar_motor(temp):
    if temp > 27:
        GPIO.output(MOTOR_1, GPIO.HIGH)
        GPIO.output(MOTOR_2, GPIO.LOW)
    else:
        GPIO.output(MOTOR_1, GPIO.LOW)
        GPIO.output(MOTOR_2, GPIO.LOW)
```

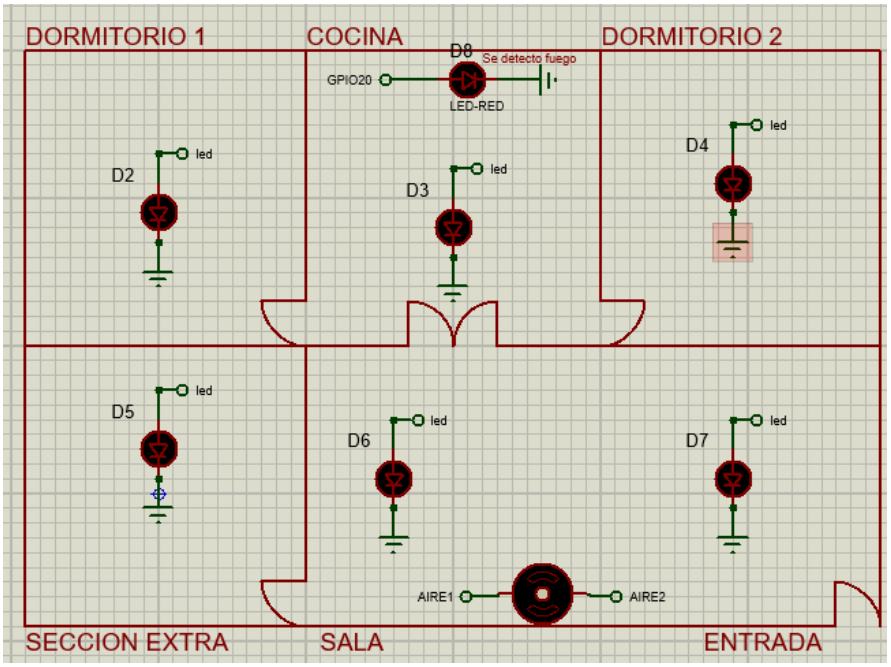
Controla el motor en función de la temperatura medida. Enciende el motor si la temperatura supera los 27 grados Celsius y lo apaga en caso contrario.

Proteus

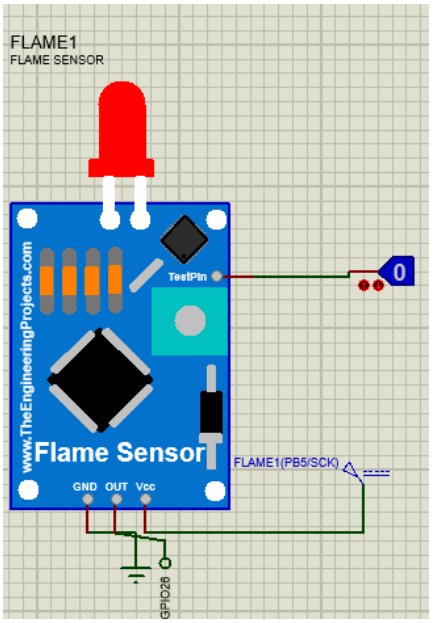
- Estructura General

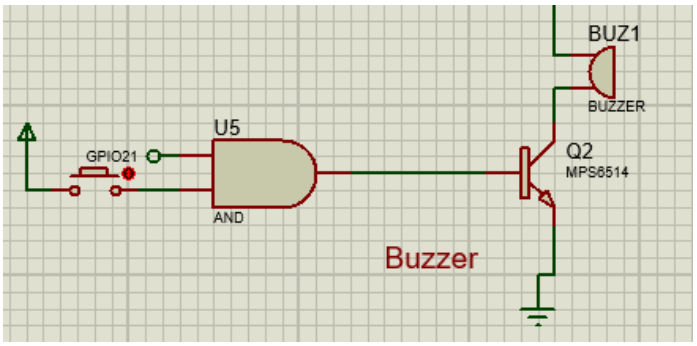


- Casa y Habitaciones

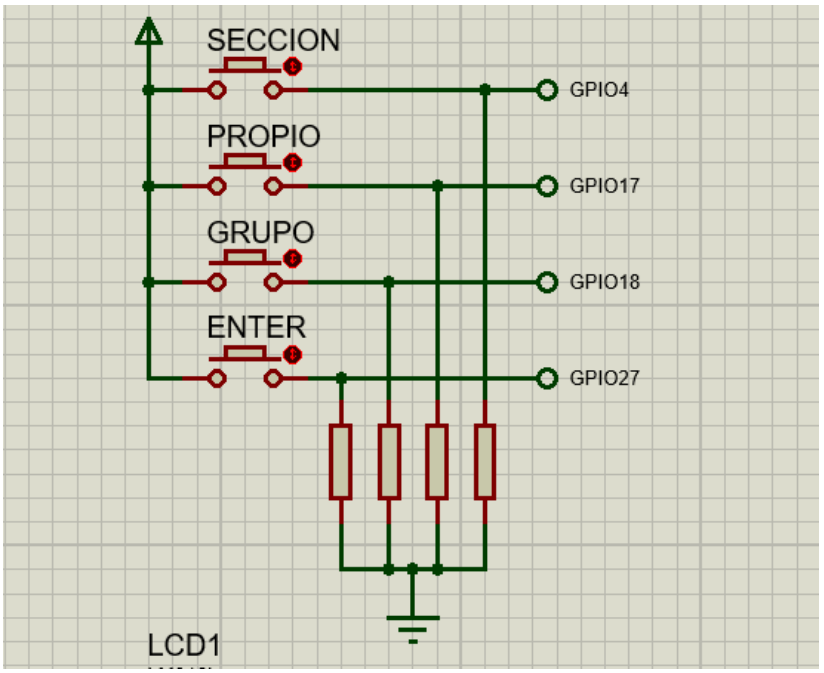


- Sensor de Flamas y Buzzer

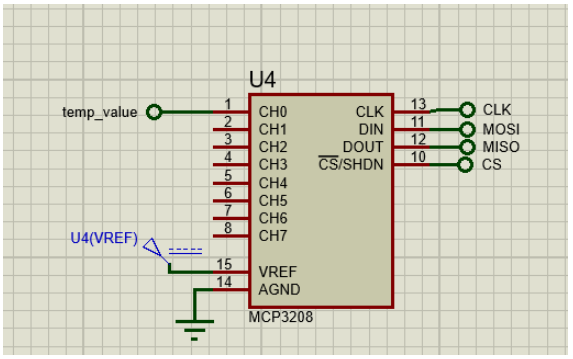


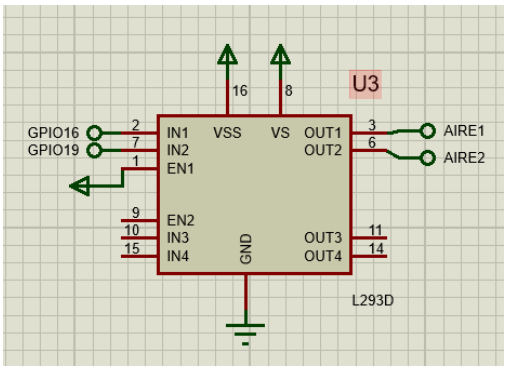


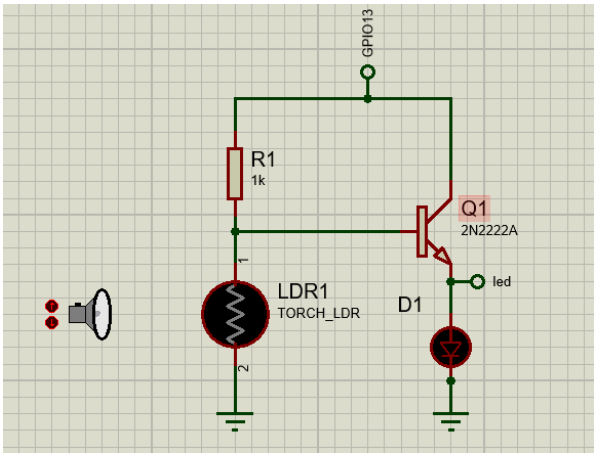
- Teclado y LCD

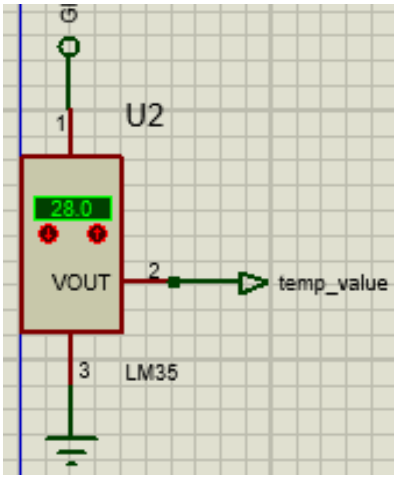


- MCP3208 Y L293D









Conclusiones

1. **Detección y Seguridad**: El sensor de flama es crucial para la detección temprana de incendios, mejorando la seguridad del hogar al enviar señales a la Raspberry Pi, que posteriormente activa alarmas sonoras mediante un buzzer. Esto asegura una respuesta rápida ante condiciones de peligro.
2. **Conversión de Señales y Procesamiento de Datos**: El MCP3208 es fundamental en la conversión de señales analógicas, como las provenientes de un sensor de temperatura, en datos digitales procesables por la Raspberry Pi. Esta capacidad de conversión es vital para integrar sensores analógicos en un sistema digital, permitiendo un monitoreo preciso de variables físicas.
3. **Control de Actuadores**: El L293D permite el control eficiente de motores de corriente continua, como los utilizados en sistemas de aire acondicionado, mediante la gestión de señales eléctricas desde la Raspberry Pi. Esto demuestra cómo los componentes de control, junto con la lógica de sistemas integrados, pueden automatizar tareas físicas basadas en datos de sensores.
4. **Arquitectura de Computadoras**: La interacción entre los diversos componentes y la Raspberry Pi a través de pines GPIO y protocolos como SPI, resalta la importancia de la arquitectura de computadoras en la implementación de sistemas embebidos para automatización. La gestión precisa de señales y la sincronización entre los componentes son clave para el funcionamiento integral del sistema automatizado.