Cocktail Machine

Desarrollo de una máquina dispensadora de cócteles automática controlada por Raspberry Pi

Curso: Creación de prototipos de IoT con Raspberry

Profesor: Joan Masdemont

4/12/2024

Luís Miguel Costa Ferreira

Memoria del Proyecto: Sistema Automatizado de Preparación de Cócteles

Introducción

¿Por qué crear una coctelera automatizada?

El proyecto 'Cocktail Machine' busca automatizar la preparación de cócteles utilizando tecnología moderna, combinando Raspberry Pi con sensores, actuadores y software en Python. Este sistema tiene como objetivo ofrecer una experiencia fácil de usar, permitiendo a cualquier persona disfrutar de cócteles precisos sin necesidad de conocimientos avanzados en mixología.

Descripción del Producto

La máquina utiliza una Raspberry Pi como controlador principal, un LCD para la interfaz de usuario, un encoder rotativo para seleccionar recetas y bombas peristálticas para dispensar los ingredientes. El sistema puede preparar varias recetas almacenadas en el programa y es capaz de manejar ingredientes con diferentes tasas de flujo.

Objetivos

General: Diseñar y construir un sistema automatizado para preparar cócteles.

Específicos:

1. Automatización del proceso: Controlar bombas para dispensar líquidos con precisión.
2. Interfaz de usuario: Crear una navegación intuitiva para seleccionar recetas.
3. Versatilidad: Implementar recetas personalizadas y facilitar la limpieza automática del sistema.
4. Modularidad: Diseñar un sistema fácil de mantener y ampliar.

Materiales y Hardware

- Raspberry Pi
- Pantalla LCD 16x2
- Encoder rotativo
- Bombas peristálticas (1 grande y 2 pequeñas)
- Placas controladoras L293D
- Fuentes de alimentación

Las bombas están configuradas para manejar diferentes tasas de flujo:

Bomba pequeña: ~2.3 mL/seg

Bomba grande: ~7.1 mL/seg

Especificaciones Técnicas del Sistema

1. Especificaciones de las Bombas

Bomba	Corriente Media (A)	Corriente en Picos (A)
Grande	0,48	0,6
Pequeña	0,17	0,3

Las bombas tienen diferentes demandas de corriente dependiendo de su tamaño. Esto debe considerarse al elegir una fuente de alimentación adecuada.

2. Detalles del L293D

Característica	Detalles
Corriente máxima por canal	Hasta 0.6A continuo, 1.2A en picos breves
Número de canales	2 (puede controlar dos motores por separado)
Voltaje de operación	De 4.5V a 36V
Protección interna	Incluye diodos de protección contra corrientes inversas
Caída de tensión	Aproximadamente 1.2V por canal

El L293D es un controlador de motores que permite manejar bombas mediante señales GPIO. Cada canal soporta un motor y está protegido contra corrientes inversas.

3. Flujo de las Bombas

Bomba	Flujo Aprox. (mL/min)
Pequeña	100
Grande	200

Se usará 1 bomba grande para el ingrediente más abundante y 2 pequeñas para los otros ingredientes. Un controlador L293D se conectará a la bomba grande. Otro controlará 2 bombas pequeñas.

4. Potencia Necesaria

Componente	Consumo de Corriente	Potencia (W)
Bomba grande (12V)	0,48 A	5,76
Bombas pequeñas (12V) (x2)	2 x 0,17 A	4,08
Raspberry Pi 3 (5V)	2,5 A	12,5
LCD 16x2 (5V)	0,02 A (retroiluminado)	0,1
Potencia combinada	-	**22.4 W**

Potencia total mínima: 22.4W (Versión con 6 bombas: 32.28W)
Para seguridad y margen de operación, se recomienda usar una fuente que suministre 5V y 12V y al menos 5A (25W) si ambos dispositivos se alimentan desde la misma fuente.

5. Pruebas de Bombas

Bomba	Tiempo para 500mL	Flujo (mL/min)	Flujo (mL/seg)
Pequeña	3 min 40 seg	137	2.3
Grande	1 min 10 seg	427	7.1

Durante las pruebas, se determinó que las bombas tienen tasas de flujo específicas que garantizan precisión en la dispensación de líquidos.

6. Conexión de los Pines

Controlador	Pines de Control	Estado
L293D	IN1 e IN2	LOW y HIGH

Los pinos IN1 y IN2 del controlador serán conectados a LOW y HIGH respectivamente, para controlar las bombas con 1 pino en vez de 3 pinos.

Explicación del Código

El programa principal está diseñado en Python y tiene varios componentes clave que se explican a continuación:

1. Configuración Inicial

- Configuración de GPIO para manejar el encoder, LCD y las bombas peristálticas.
- Inicialización de las clases LCD y MotorDC, responsables del control del display y las bombas, respectivamente.

2. Control del LCD

La clase LCD muestra mensajes relevantes durante el proceso, como el nombre de la receta y la cantidad de cada ingrediente. Incluye métodos para manejar la escritura de caracteres y la navegación entre líneas.

```
def escriu_frase(self, frase):
    """Escribe una frase en el LCD, dividiendo en dos líneas si es necesario."""
    # Divide la frase en líneas y escribe en el LCD
```

3. Control de Bombas

- Las bombas se controlan mediante la clase MotorDC, que utiliza PWM para manejar el tiempo y la intensidad de dispensación.
- Se calcula el tiempo necesario para dispensar una cantidad específica de líquido basada en el flujo.

```
def calcular_tiempo(ml, flujo):
    return ml / flujo
```

4. Selección de Recetas

Los usuarios seleccionan recetas utilizando un encoder rotativo. La función `navigate_menu` actualiza la receta seleccionada basándose en la dirección de giro del encoder.

```
def navigate_menu():
    while True:
        clk_state = GPIO.input(CLK)
        if clk_state != clk_last_state:
            position = (position + 1) % len(drink_recipes)
            lcd.escriu_frase(drink_recipes[position]["name"])
        clk_last_state = clk_state
```

5. Preparación de Cócteles

La función `preparar_bebida` controla el proceso de preparación, activando las bombas según las cantidades especificadas en la receta.

```python
def preparar_bebida(lcd, bebida):
    for ingrediente, cantidad in bebida["ingredients"].items():
        if ingrediente in bombas:
            tiempo = calcular_tiempo(cantidad, flujo)
            bombas[ingrediente].dispensar(tiempo)
```

Resultados y Pruebas

1. Pruebas de flujo: Las bombas dispensaron líquidos con un margen de error mínimo tras la calibración.
2. Usabilidad: La interfaz permite navegar fácilmente por las recetas y visualizar el progreso en el LCD.
3. Consistencia: Las recetas se prepararon con medidas exactas y tiempos controlados.

Imágenes del Proyecto

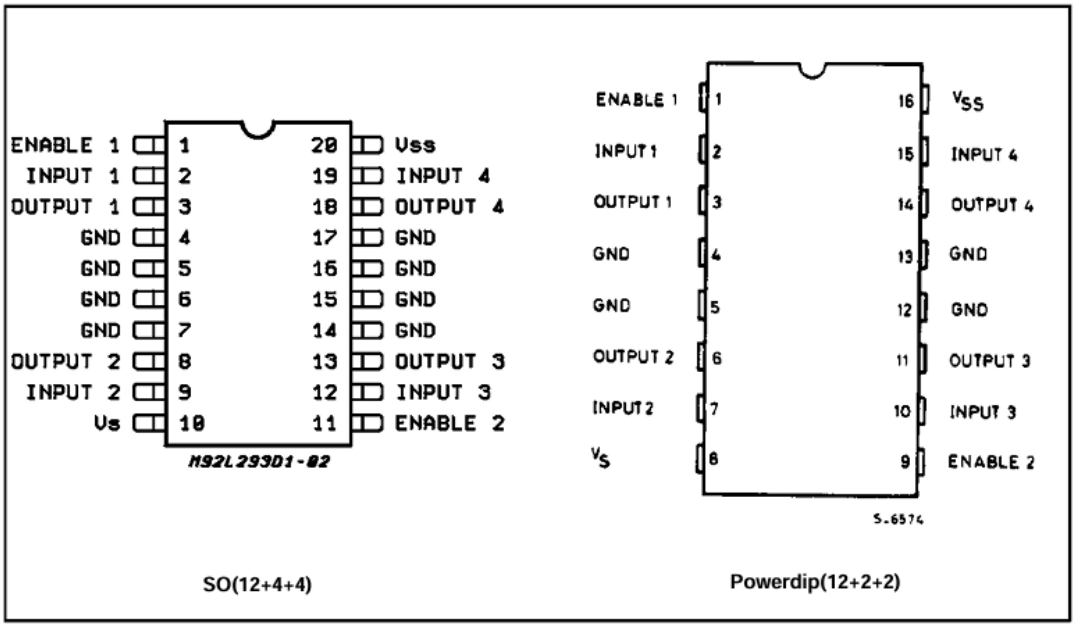
A continuación se presentan algunas imágenes tomadas durante el desarrollo del proyecto para ilustrar el montaje y la implementación del sistema automatizado.

L293D - L293DD

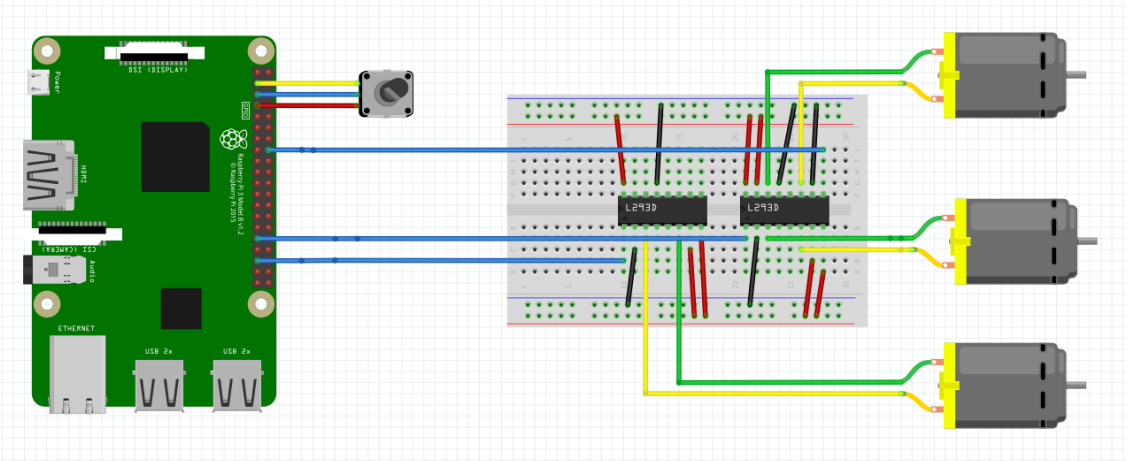
ABSOLUTE MAXIMUM RATINGS

Symbol	Parameter	Value	Unit
V_S	Supply Voltage	36	V
V_{SS}	Logic Supply Voltage	36	V
V_i	Input Voltage	7	V
V_{en}	Enable Voltage	7	V
I_o	Peak Output Current (100 µs non repetitive)	1.2	A
P_{tot}	Total Power Dissipation at T_{pins} = 90 °C	4	W
T_{stg}, T_j	Storage and Junction Temperature	– 40 to 150	°C

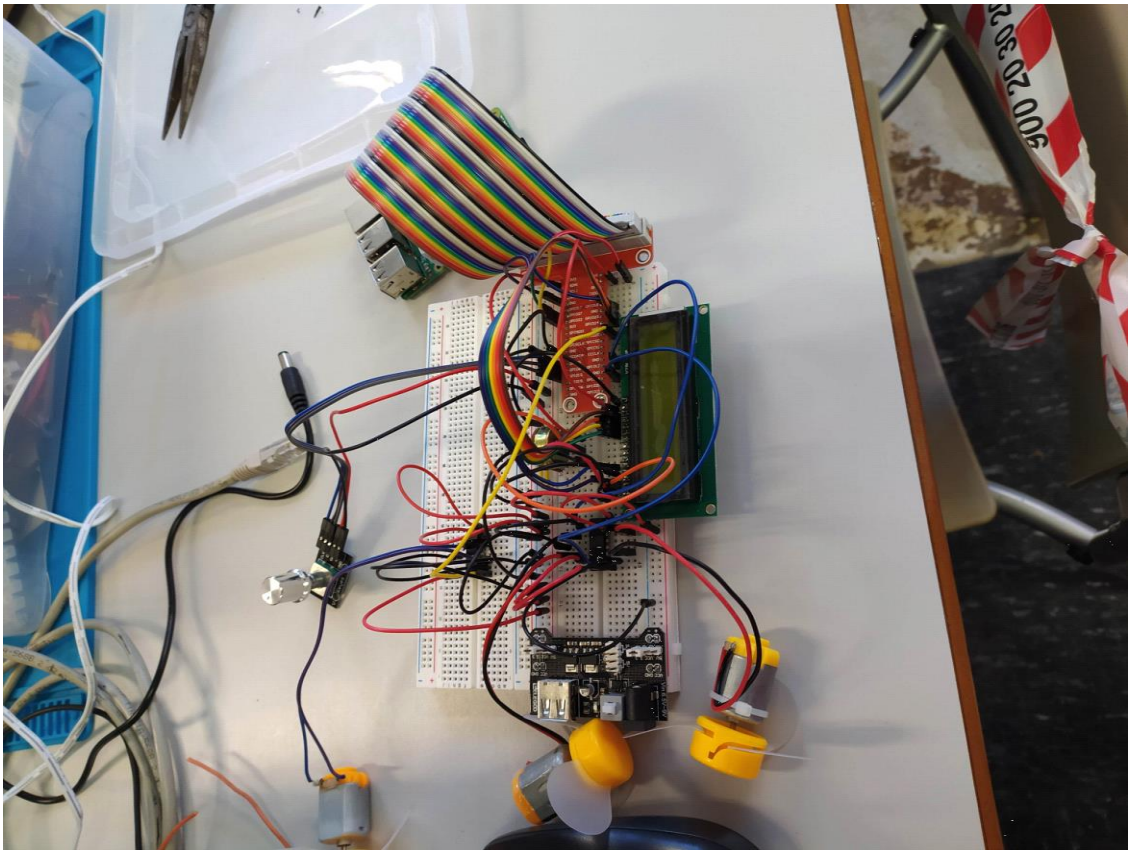
PIN CONNECTIONS (Top view)



PinOut del L293D



Esquema en Fritzing de la ProtoBoard

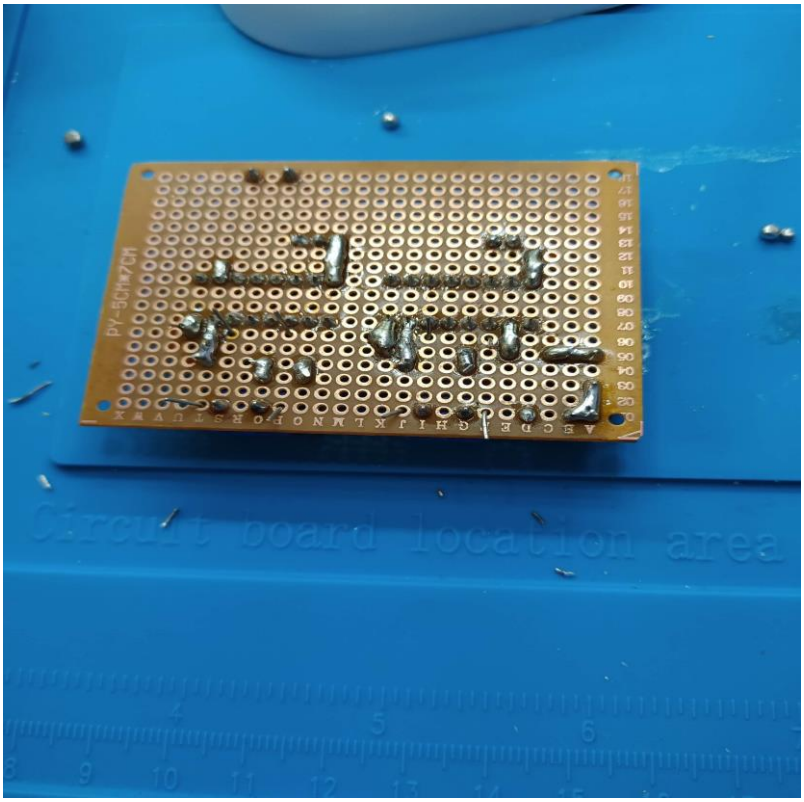


Montaje con motores DC antes de conectar las bombas y la fonte de alimentación

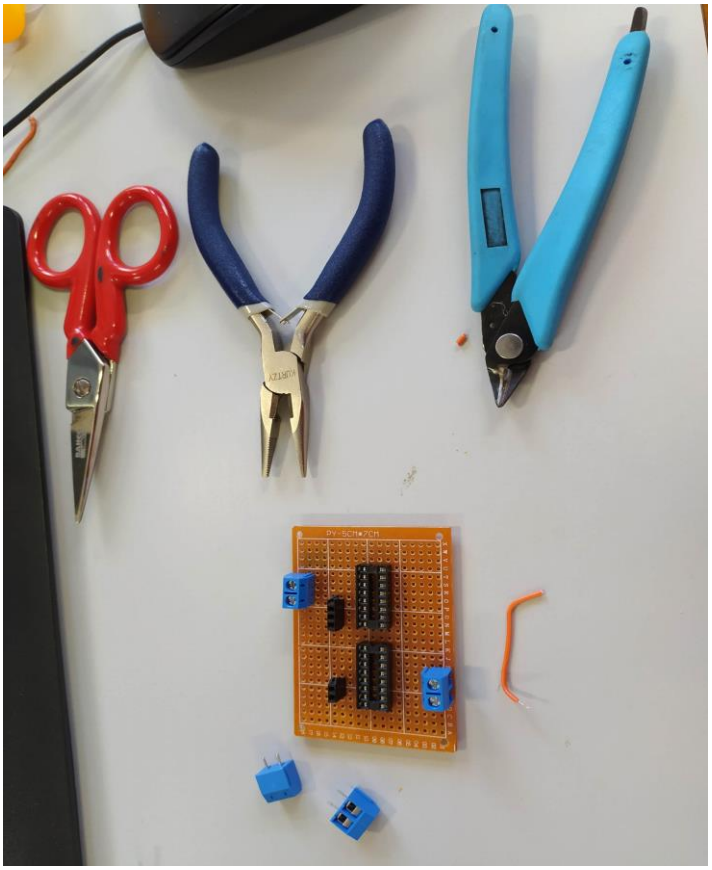
Soldadura de las bombas con cable de 0,75mm^2 de sección.

Instalación de las bombas, se procuró aislar el circuito hidraulico del circuito eléctrico.

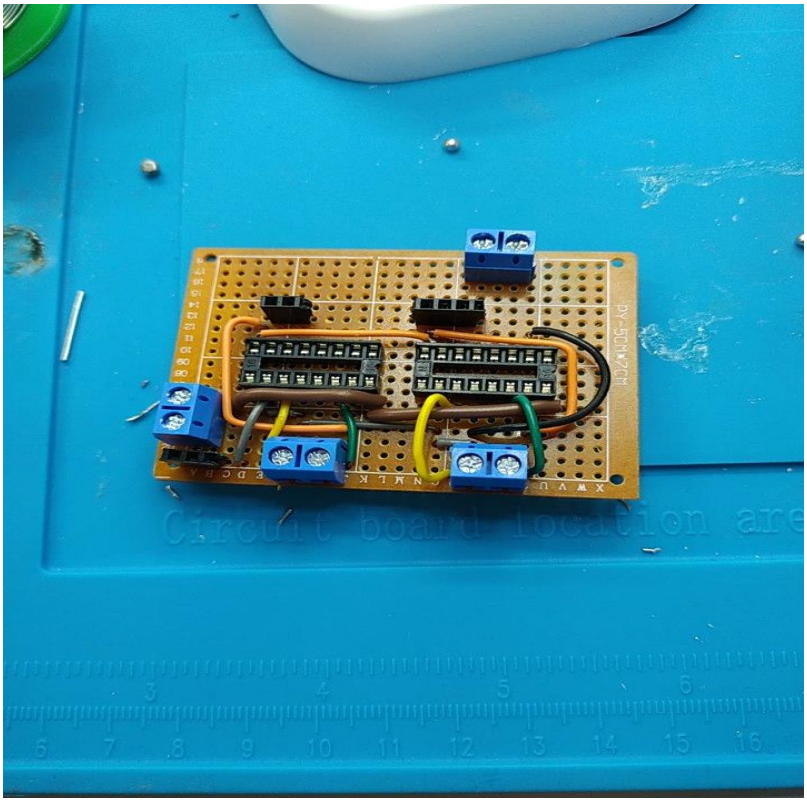
Instalación de las bombas. Se procuró aislar el circuito hidráulico del circuito eléctrico.



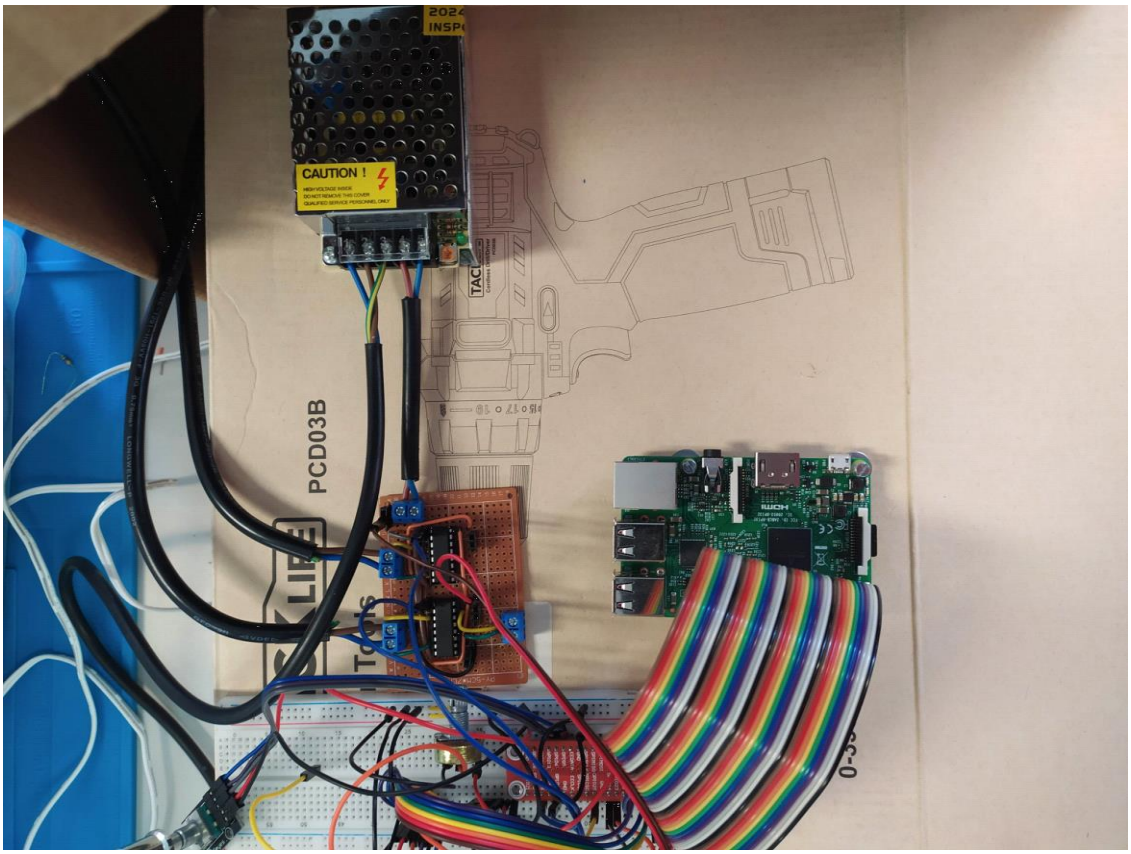
Soldadura de la placa de controlo de las bombas.



Se usaran borneras. La corriente de las bombas llega a ser superior a la soportada pelos cables Dupond



El cable marrón es la alimentación proveniente de la fuente de alimentación (0,75mm^2 de sección)



Montaje final antes de cerrar la máquina

Conclusión

El proyecto 'Cocktail Machine' demuestra cómo la tecnología puede simplificar procesos cotidianos. Se logró un sistema funcional y modular que combina hardware y software de manera eficiente. A futuro, se podrían integrar más recetas, modos de limpieza automática y un sistema de control remoto mediante una aplicación móvil.

Anexo: Código Fuente

```
import RPi.GPIO as GPIO
import time

# Configuración de pines para encoder y LCD
CLK = 5
DT = 6
SW = 13

# Configuración de pines para el LCD
RS = 4
E = 18
D4 = 27
D5 = 22
D6 = 23
D7 = 24

# Configuración de GPIO
GPIO.setmode(GPIO.BCM)
GPIO.setup(CLK, GPIO.IN, pull_up_down=GPIO.PUD_UP)
GPIO.setup(DT, GPIO.IN, pull_up_down=GPIO.PUD_UP)
GPIO.setup(SW, GPIO.IN, pull_up_down=GPIO.PUD_UP)

# Flujo de las bombas
FLUJO_PEQ = 1.78 # mL/seg
FLUJO_GRA = 6.75 #mL/seg
FLUJO_ORANGE = 5.3  # mL/seg

# Clase LCD (manteniendo la original)
class LCD:
    def __init__(self, rs, e, d4, d5, d6, d7, pausa=0.02):
        # Configura els pins i pausa
        self.RS = rs
        self.E = e
        self.D4 = d4
        self.D5 = d5
        self.D6 = d6
        self.D7 = d7
        self.PAUSA = pausa

        # Configura els pins GPIO
        GPIO.setmode(GPIO.BCM)
        GPIO.setup([self.RS, self.E, self.D4, self.D5, self.D6, self.D7], GPIO.OUT)
        GPIO.output([self.RS, self.E, self.D4, self.D5, self.D6, self.D7], 0)

    def char2bin(self, char):
        """Converteix un caràcter en una representació binària de 8 bits i la retorna com una tupla d'enters."""
        strink = bin(ord(char))[2:]
        strink = '0' * (8 - len(strink)) + strink
        resultat = ''
        for bit in strink:
            resultat = bit + resultat
        res = resultat[4:] + resultat[:4]
        tupla = tuple([int(element) for element in res])
        return tupla

    def modecomandament(self, valor):
        """Configura el mode de comandament del display, ajustant el pin RS segons si és instrucció o dada."""
        GPIO.output(self.RS, GPIO.HIGH if not valor else GPIO.LOW)
```

```
    GPIO.output(self.E, GPIO.LOW)

def escriu_a_fila_u(self):
    """Mou el cursor a l'inici de la primera fila del display."""
    self.modecomandament(True)
    self.escriu4bits(0, 0, 0, 0)
    self.escriu4bits(0, 0, 0, 0)

def escriu_a_fila_dos(self):
    """Mou el cursor a l'inici de la segona fila del display."""
    self.modecomandament(True)
    self.escriu4bits(0, 0, 1, 1)
    self.escriu4bits(0, 0, 0, 0)

def escriu4bits(self, b1, b2, b3, b4):
    """Envia 4 bits al display a través dels pins D4-D7."""
    GPIO.output(self.D4, b1)
    GPIO.output(self.D5, b2)
    GPIO.output(self.D6, b3)
    GPIO.output(self.D7, b4)
    time.sleep(self.PAUSA)
    GPIO.output(self.E, GPIO.HIGH)
    GPIO.output(self.E, GPIO.LOW)
    time.sleep(self.PAUSA)

def esborra_la_pantalla(self):
    """Envia la instrucció per esborrar tot el display."""
    self.modecomandament(True)
    time.sleep(self.PAUSA)
    self.escriu4bits(0, 0, 0, 0)
    self.escriu4bits(1, 0, 0, 0)

def envia_dades_al_display(self, dada):
    """Envia un caràcter (en forma de tupla de 8 bits) al display per a ser mostrat."""
    self.modecomandament(False)
    self.escriu4bits(dada[0], dada[1], dada[2], dada[3])
    self.escriu4bits(dada[4], dada[5], dada[6], dada[7])

def detencio_pantalla(self):
    """Posa el display en mode de detenció o pausa."""
    self.modecomandament(True)
    self.escriu4bits(0, 0, 0, 0)
    self.escriu4bits(0, 0, 1, 1)

def inicia_pantalla(self):
    """Configura el display per iniciar-lo, establint-lo amb dues files i esborrant-lo."""
    self.modecomandament(True)
    for _ in range(3):
        self.escriu4bits(1, 1, 0, 0)
    for _ in range(2):
        self.escriu4bits(0, 1, 0, 0)
    self.escriu4bits(1, 0, 1, 1)
    self.escriu4bits(0, 0, 0, 0)
    self.escriu4bits(1, 1, 1, 1)
    self.esborra_la_pantalla()

def cleanup(self):
    """Neteja la configuració de GPIO."""
    GPIO.cleanup()
```

```
    def escriu_frase(self, frase):
        """
        Escriu una frase completa al LCD.
        Divideix la frase en dues línies si és més llarga que l'amplada del LCD (16 caràcters).
        """
        self.esborra_la_pantalla()

        linea_1 = frase[:16]
        linea_2 = frase[16:32]

        self.escriu_a_fila_u()
        for char in linea_1:
            self.envia_dades_al_display(self.char2bin(char))

        if linea_2.strip():
            self.escriu_a_fila_dos()
            for char in linea_2:
                self.envia_dades_al_display(self.char2bin(char))

# Clase MotorDC para controlar las bombas
class MotorDC:
    def __init__(self, ena, in1, in2):
        self.ENA = ena
        self.IN1 = in1
        self.IN2 = in2
        GPIO.setup([self.ENA, self.IN1, self.IN2], GPIO.OUT)
        self.pwm = GPIO.PWM(self.ENA, 100)  # Frecuencia de 100 Hz
        self.pwm.start(0)

    def dispensar(self, tiempo, dutycycle=100):
        GPIO.output(self.IN1, GPIO.LOW)
        GPIO.output(self.IN2, GPIO.HIGH)
        self.pwm.ChangeDutyCycle(dutycycle)
        time.sleep(tiempo)
        self.pwm.ChangeDutyCycle(0)

    def detener(self):
        self.pwm.ChangeDutyCycle(0)

    def cleanup(self):
        """Detura el PWM i neteja la configuració GPIO del motor."""
        self.pwm.stop()
        del self.pwm  # Elimina l'objecte PWM per evitar errors en el destructor
        GPIO.output([self.ENA, self.IN1, self.IN2], GPIO.LOW)

# Configuración de bombas
bombas = {
    "Rum": MotorDC(16, 20, 21),  # Pines GPIO para la bomba de "Rum"
    "Coke": MotorDC(12, 19, 26),  # Pines GPIO para la bomba de "Coke"
    "Orange Juice": MotorDC(25, 17, 21),  # Pines GPIO para la bomba de "Orange Juice"

}

# Configuración de recetas
drink_recipes = [
    {"name": "Rum & Coke", "ingredients": {"Rum": 50, "Coke": 150}},
    {"name": "Gin & Tonic", "ingredients": {"Gin": 50, "Tonic": 150}},
    {"name": "Screwdriver", "ingredients": {"Vodka": 50, "Orange Juice": 150}},
    {"name": "Long Island", "ingredients": {"Gin": 15, "Rum": 15, "Vodka": 15, "Tequila": 15, "Coke": 100, "Orange
Juice": 15}},
```

```
    {"name": "Margarita", "ingredients": {"Tequila": 50, "mmix": 150}},
    {"name": "Gin & Juice", "ingredients": {"Gin": 50, "Orange Juice": 150}},
    {"name": "Tequila Sunrise", "ingredients": {"Tequila": 50, "Orange Juice": 150}},
]

# Función para calcular tiempo de dispensación
def calcular_tiempo(ml, ingrediente):
    if ingrediente == "Rum":
        flujo = FLUJO_PEQ
    elif ingrediente == "Coke":
        flujo = FLUJO_GRA
    elif ingrediente == "Orange Juice":
        flujo = FLUJO_ORANGE
    else:
        raise ValueError(f"Flujo desconocido para {ingrediente}")
    return ml / flujo


# Función para activar las bombas y preparar la bebida
def preparar_bebida(lcd, bebida):
    lcd.escriu_frase(f"Preparando {bebida['name']}")
    for ingrediente, cantidad in bebida["ingredients"].items():
        if ingrediente in bombas:
            tiempo = calcular_tiempo(cantidad, ingrediente)
            lcd.escriu_frase(f"{ingrediente}: {cantidad}ml")
            bombas[ingrediente].dispensar(tiempo)
        else:
            lcd.escriu_frase(f"{ingrediente}: No disponible")
        time.sleep(2)  # Pausa entre ingredientes
    lcd.escriu_frase(f"{bebida['name']} lista!")


def cleanup_tot():
    """Atura motors, neteja LCD i GPIO."""
    for motor in bombas.values():
        motor.detener()
        motor.cleanup()  # Atura i neteja cada motor
    lcd.esborra_la_pantalla()
    lcd.detencio_pantalla()
    time.sleep(1)  # Esperar un poco para asegurarnos de que el LCD se limpia correctamente
    #lcd.cleanup()  # Neteja el LCD
    #GPIO.cleanup()  # Neteja els GPIO



def iniciar_lcd():
    """Inicialitza el LCD de manera segura per evitar errors de display."""
    lcd = LCD(RS, E, D4, D5, D6, D7)
    lcd.inicia_pantalla()  # Inicialitza el display a un estat net i configurat
    return lcd

# Navegación del menú

def navigate_menu():
    clk_last_state = GPIO.input(CLK)
    position = 0

    try:
        while True:
            clk_state = GPIO.input(CLK)
```

```
            if clk_state != clk_last_state:
                if GPIO.input(DT) != clk_state:
                    position = (position + 1) % len(drink_recipes)
                else:
                    position = (position - 1) % len(drink_recipes)
                lcd.escriu_frase(drink_recipes[position]["name"])
            clk_last_state = clk_state

            if GPIO.input(SW) == GPIO.LOW:
                bebida = drink_recipes[position]
                preparar_bebida(lcd, bebida)
                break  # Salimos después de preparar
            time.sleep(0.1)

    except KeyboardInterrupt:
        print("Programa interrumpido.")
        cleanup_tot()  # Llamamos a la función de limpieza para detener todo y limpiar el LCD

# Ejecución del programa
if __name__ == "__main__":
    lcd = iniciar_lcd()  # Usamos la función de inicialización limpia
    lcd.escriu_frase("Elige tu bebida")
    navigate_menu()
```