Regulador de Temperatura

Este proyecto es un regulador de temperatura, en el momento en el que se superan los 19° (un limite impuesto de forma propia) se activan unos leds, al activarse estos leds se prende un ventilador de pc que se encarga de proporcionarle ventilacion y frio al sensor de temperatura, cuando baja de los 29° el ventilador se apaga.

Explicacion del codigo por partes:

Inclusion de Librerias:

```c
#include <stdio.h>
#include <math.h>
#include "pico/stdlib.h"
#include "mpu6050.h"

```

Stdio.h: Para imprimir en la consola

Math.h: Para hacer calculos matematicos

Pico/stdlib.h: Libreria estandar de raspbrry pi pico para entradas y salidas

Mpu6050.h: Nos permite trabajar con el sensor MPU6050

Pines y Constantes:

```c
#define SDA_GPIO  4
#define SCL_GPIO  5
#define LED_PIN_X 14 // Led rojo
#define LED_PIN_Y 15 // Led verde
#define FAN_PIN 13   // Ventilador
#define TEMP_LIMIT 29 // Límite de temperatura

```

SDA_GPIO y SCL_GPIO: Pines GPIO usados para el bus I2C
LED_PIN_X y LED_PIN_Y: Pines para los LEDs (rojo y verde).
FAN_PIN: Pin usado para controlar el ventilador
TEMP_LIMIT: Límite de temperatura para encender LEDs y activar el ventilador.

Funcion Main:

```c
int main(void)
{
    stdio_init_all();
    sleep_ms(1000);

    printf("Inicializando I2C...\n");

```

stdio_init_all: Inicializa la comunicación estándar para entrada/salida (necesaria para printf).

- sleep_ms(1000): Pausa de 1 segundo para asegurarse de que todo esté listo.
- printf: Imprime un mensaje en la consola para depuración.

```
20      // Inicialización de I2C
21      i2c_init(i2c0, 400000);
22      gpio_set_function(SDA_GPIO, GPIO_FUNC_I2C);
23      gpio_set_function(SCL_GPIO, GPIO_FUNC_I2C);
24      gpio_pull_up(SDA_GPIO);
25      gpio_pull_up(SCL_GPIO);
26
```

i2c_init: Configura el bus I2C en la instancia i2c0 con una velocidad de 400 kHz.
gpio_set_function: Asigna la función I2C a los pines SDA_GPIO y SCL_GPIO.
gpio_pull_up: Activa resistencias de pull-up en los pines I2C.

```
27      printf("Inicializando MPU6050...\n");
28
29      // Inicialización del MPU6050
30      mpu6050_init(i2c0, 0x68);
31      sleep_ms(2000);
32
33      printf("Who am I = 0x%2x\n", mpu6050_who_am_i());
34
```

mpu6050_init: Configura el sensor en el bus I2C con dirección 0x68.
sleep_ms(2000): Pausa de 2 segundos para estabilizar el sensor.
mpu6050_who_am_i: Devuelve un valor de identificación del sensor para verificar la comunicación.

```
35      // Configuración de los pines de salida
36      gpio_init(LED_PIN_X);
37      gpio_init(LED_PIN_Y);
38      gpio_init(FAN_PIN);
39      gpio_set_dir(LED_PIN_X, GPIO_OUT);
40      gpio_set_dir(LED_PIN_Y, GPIO_OUT);
41      gpio_set_dir(FAN_PIN, GPIO_OUT);
42
```

gpio_init: Inicializa los pines como GPIO.
gpio_set_dir: Configura los pines como salidas digitales.

Bucle Principal:

Lectura de datos del sensor

```
43      while (true) {
44          // Estructura para datos del sensor
45          mpu_accel_t accel = {0};
46          mpu_gyro_t gyro = {0};
47          int16_t temp = 0;
48
49          // Leer datos del MPU6050
50          mpu6050_read_accel(&accel);
51          mpu6050_read_gyro(&gyro);
52          mpu6050_read_temp(&temp);
53
```

mpu_accel_t y mpu_gyro_t: Estructuras para almacenar datos del acelerómetro y giroscopio.
mpu6050_read_accel, mpu6050_read_gyro, mpu6050_read_temp: Funciones que leen datos crudos del sensor.

Calculo de temperatura

```
53
54          // Convertir temperatura a un valor real (en decimales)
55          float temperature = temp / 340.0 + 36.53;
56
57          printf("Temperatura = %.2f\n", temperature);
58
```

La temperatura se convierte a grados Celsius usando la fórmula:
Temperatura = temp/340.0 + 36.53

Led y Ventilador

```
57          printf("Temperatura = %.2f\n", temperature);
58
59          // Verificar si la temperatura supera el límite y activar LEDs
60          bool led_x_on = temperature > TEMP_LIMIT;
61          bool led_y_on = temperature > TEMP_LIMIT;
62
63          gpio_put(LED_PIN_X, led_x_on);
64          gpio_put(LED_PIN_Y, led_y_on);
65
66          // Activar ventilador si cualquiera de los LEDs está encendido
67          if (led_x_on || led_y_on) {
68              gpio_put(FAN_PIN, 0); // Encender ventilador
69          } else {
70              gpio_put(FAN_PIN, 1); // Apagar ventilador
71          }
72
```

Control de LEDs: Se encienden si la temperatura supera el límite (TEMP_LIMIT).
Control del ventilador:
Encender (0) si al menos un LED está encendido.
Apagar (1) si ambos LEDs están apagados.

Pausa entre lecturas:

```
72
73          // Pausa de 100 ms entre lecturas
74          sleep_ms(100);
75      }
76  }
```