// Inclusion de librerias

#include <stdio.h>

#include "pico/stdlib.h"

#include "hardware/adc.h"

#include "hardware/gpio.h"

#include <math.h>

// GPIO del 7 segmentos

#define SEG\_A   0

#define SEG\_B   1

#define SEG\_C   2

#define SEG\_D   3

#define SEG\_E   4

#define SEG\_F   5

#define SEG\_G   6

#define SEG\_DP  7

#define DIG\_1   8

#define DIG\_2   9

#define DIG\_3   10

// GPIO del ADC

#define ANALOG\_GPIO 26

// Canal del ADC

#define ANALOG\_CH   0

// Mascara para GPIO de segmentos

#define SEGMENTS  (1u << SEG\_A) | (1u << SEG\_B) | (1u << SEG\_C) | (1u << SEG\_D) | (1u << SEG\_E) | (1u << SEG\_F) | (1u << SEG\_G) | (1u << SEG\_DP)

// Mascara para cada digito

#define DIGIT\_1   (1 << DIG\_1)

#define DIGIT\_2   (1 << DIG\_2)

#define DIGIT\_3   (1 << DIG\_3)

// Mascara para GPIO de digitos

#define DIGITS    DIGIT\_1 | DIGIT\_2 | DIGIT\_3

// Tiempo de refresco para el siete segmentos

#define SLEEP\_MS 500

// Coeficiente de temperatura del termistor

#define BETA 8000

/\*

 \* @brief Display temperature value in 7 segment display

 \* @param temperature: value to display

 \*/

void display\_temp(float celsius) {

  // Array con valores de segmentos para cada numero (prende con cero el segmento)

  const uint8\_t digitos[10] = {

    ~0x3f, ~0x06, ~0x5b, ~0x4f, ~0x66,

    ~0x6d, ~0x7d, ~0x07, ~0x7f, ~0x67

  };

  // Variables para cada digito individualmente

  uint8\_t digit\_1, digit\_2, digit\_3;

  // Obtengo el valor de la decena

  digit\_1 = celsius / 10;

  // Prendo el digito 1

  gpio\_put\_masked(DIGITS, DIGIT\_1);

  // Dibujo el digito

  gpio\_put\_masked(SEGMENTS, digitos[digit\_1]);

  // Espero

  sleep\_ms(SLEEP\_MS);

  // Obtengo el valor de la unidad

  digit\_2 = celsius - digit\_1;

  // Prendo el digito 2

  gpio\_put\_masked(DIGITS, DIGIT\_2);

  // Dibujo el digito 2 con el punto

  gpio\_put\_masked(SEGMENTS, digitos[digit\_2] & ~(1 << SEG\_DP));

  // Espero

  sleep\_ms(SLEEP\_MS);

  // Obtengo el valor del primer decimal

  digit\_3 = (celsius -(digit\_1 + digit\_2)) \* 10;

  // Prendo el digito 3

  gpio\_put\_masked(DIGITS, DIGIT\_3);

  // Dibujo el digito

  gpio\_put\_masked(SEGMENTS, digitos[digit\_3]);

  // Espero

  sleep\_ms(SLEEP\_MS);

}

/\*

 \* @brief Programa principal

 \*/

int main() {

  // Variable para almacenar el resultado del ADC

  //uint16\_t adc\_reading = 0;

  uint16\_t res = 0;

  // Variable para guardar el valor de temperatura

  float celsius = 0.0;

  // Inicializacion de UART

  stdio\_init\_all();

  // Inicializacion de GPIO

  gpio\_init\_mask(SEGMENTS | DIGITS);

  // Configuro GPIO como salida

  gpio\_set\_dir\_out\_masked(SEGMENTS | DIGITS);

  // Valores iniciales de GPIO cero

  gpio\_put\_masked(SEGMENTS | DIGITS, 0x0);

  // Inicializo ADC

  adc\_init();

  // Inicializo GPIO26 como entrada analogica

  adc\_gpio\_init(26);

  // Selecciono canal analogico

  adc\_select\_input(0);

  // Enciendo todos los digitos

  gpio\_put\_masked(DIGITS, 0x0);

  // Bucle infinito

  while (true) {

    // Leer NTC

    int16\_t res = adc\_read();

    printf("res: %d\n", res);

    // Calculo temperatura

    float celsius = 1 / (log(1 / (4096. / res - 1)) / BETA + 1.0 / 298.15) - 273.15;

    // Puede verificarse el valor antes de mostrar en el display con el siguiente printf

    printf("Temperatura: %.2f\n", celsius);

    // Muestro temperatura

    display\_temp(celsius);

    // Espero

    sleep\_ms(SLEEP\_MS);

  }

}