Práctica 11
Lectura de Temperatura con ADC en Raspberry Pi Pico

José Josef Medellín Reséndez
Carrera: Mecatrónica
Tercer semestre

19 de abril de 2025

Introducción

Esta práctica fue para ejecutarse en una Raspberry Pi Pico con el propósito de tomar lecturas del sensor de temperatura usando un convertidor analógico a digital (ADC) en el microcontrolador. Para hacer esto, creé un archivo C y un archivo CMakeLists.txt para definir el Pico SDK. Para así gestionar varias implementaciones a lo largo del proceso de desarrollo, como configuración de pines analógicos, conversión de datos a temperatura y transferencia de resultados vía USB.

Desarrollo

Descripción del código en C

Este código está diseñado para medir la temperatura utilizando el conversor analógico-digital (ADC) de la `Raspberry Pi Pico`, leyendo una señal desde un sensor conectado al pin 26 (canal 0 del ADC). La lectura se realiza de forma continua y los resultados se muestran por consola en formato `CSV`, es decir, temperatura y voltaje separados por coma.

Primero, se inicializan los periféricos necesarios: se habilita la entrada/salida estándar para permitir el uso de `printf`, se inicializa el módulo `ADC`, se configura el pin correspondiente para lectura analógica y se selecciona el canal del sensor. Luego, dentro de un bucle infinito, se realiza la lectura del valor analógico captado por el sensor, que es un número entero entre 0 y 4095 (debido a que el ADC de la Pico tiene una resolución de 12 bits).

Ese valor se convierte a voltaje considerando una referencia de 3.3V, y posteriormente se transforma en temperatura, bajo la suposición de que el sensor entrega 10 mV por cada grado Celsius (lo cual es típico en sensores como el `LM35`).

Cada medio segundo, el programa imprime la temperatura y el voltaje correspondientes, y se asegura de que los datos se envíen inmediatamente al puerto serie mediante `fflush(stdout)`. Esto es especialmente útil cuando los datos se grafican en tiempo real desde una computadora.

En resumen, este programa convierte las lecturas de un sensor analógico de temperatura en datos digitales claros y legibles a través del monitor serie, siendo ideal para registros, monitoreos o visualizaciones en tiempo real.

Código fuente - ADC.c

Listing 1: Código en C para lectura de temperatura

```c
#include <stdio.h>
#include "pico/stdlib.h"
#include "hardware/adc.h"

#define TEMP_SENSOR_PIN 26
#define ADC_MAX_VALUE 4096.0f
#define VOLTAGE_REF 3.3f

int main() {
    stdio_init_all();
    adc_init();
    adc_gpio_init(TEMP_SENSOR_PIN);
    adc_select_input(0);

    while (1) {
        uint16_t valor_adc = adc_read();
        float voltaje = (valor_adc * VOLTAGE_REF) / ADC_MAX_VALUE;
        float temperatura = voltaje * 100.0f;

        printf("%.2f,%.2f\n", temperatura, voltaje);  // Enviar datos separados por coma
        fflush(stdout);  // Asegura que los datos se impriman en tiempo real

        sleep_ms(500);
    }
    return 0;
}
```

CMakeLists.txt

Este archivo contiene la configuración de compilación del proyecto utilizando `CMake`, y está diseñado para ejecutarse en la `Raspberry Pi Pico`. En él se establece el nombre del proyecto como `Practica_11`, y se especifica que se requiere al menos la versión 3.13 de `CMake` para su correcta compilación.

A continuación, se incorpora e inicializa el SDK oficial de la `Raspberry Pi Pico`, lo cual permite el uso de funciones específicas del hardware como el manejo de pines, temporizadores, lectura de sensores y otros periféricos.

El archivo principal del proyecto es `ADC.c`, y se define que debe compilarse como un ejecutable con el nombre `ADC`. Para permitir el uso de funciones básicas y el control del conversor analógico-digital (ADC), se enlazan las bibliotecas `pico_stdlib` y `hardware_adc`, ambas incluidas en el SDK.

También se configura la salida estándar para redirigir los mensajes enviados con `printf` a través del puerto USB, lo cual es útil para monitorear los datos desde una computadora. La salida UART se desactiva, ya que no se utilizará en este proyecto.

Finalmente, se indica que se deben generar los archivos binarios necesarios para cargar el programa en la `Raspberry Pi Pico`, incluyendo el archivo `.uf2`, que puede ser transferido directamente a la placa cuando está en modo de programación. En resumen, este archivo `CMake` automatiza todo el proceso de compilación, enlazado y generación de binarios para asegurar que el proyecto funcione correctamente con el ADC del microcontrolador.

Código fuente - CMakeLists.txt

Listing 2: Archivo CMakeLists.txt para el proyecto

```
1
2     cmake_minimum_required ( VERSION 3.13)
3
4     # Nombre del proyecto
5     project ( Practica_11 )
6
7     # Habilitar la Raspberry Pi Pico SDK
8     include ( pico_sdk_import.cmake )
9     pico_sdk_init ()
10
11    # Agregar el archivo fuente
12    add_executable ( ADC
13    ADC.c
14    )
15
16    # Agregar las bibliotecas estandar de la Pico
17    target_link_libraries ( ADC
18    pico_stdlib
19    hardware_adc )
20
21    # Habilitar la salida USB para el puerto serie
22    pico_enable_stdio_usb ( ADC 1)
23    pico_enable_stdio_uart ( ADC 0)
24
25    # Crear el archivo binario UF2
26    pico_add_extra_outputs ( ADC )
```

Conclusión

El método de control lo implementé utilizando un ADC Raspberry Pi Pico con un sensor de temperatura, para garantizar precisión y eficiencia continuas. El código C hace todo lo necesario para inicializar y configurar el sensor, y CMakeLists.txt le ayuda a compilar un ejecutable que incluye la tabla. Esto me ayudo a la comprensión sobre la importancia de la operación analógica, el entorno del microcontrolador y la codificación y configuración para que el sistema funcione correctamente.

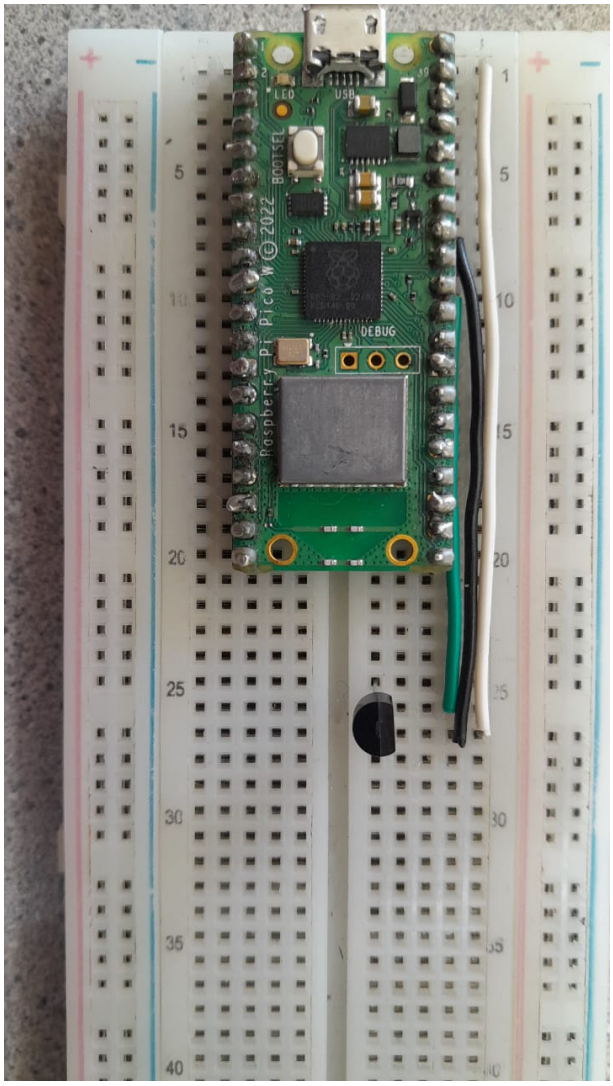

Figura 1: Raspberry conectado al sensor