Practica 11: Medición de temperatura y visualización gráfica

Matematicas por Computadora

Alumno: Itzel Estrada Arcos (368980)
Docente: Ing. Jesus Padron

Lunes 7 de abril 2025

0.1 Introduction.

En esta práctica, desarrollamos un proyecto básico para el microcontrolador Raspberry Pi Pico enfocado en la lectura de un sensor de temperatura mediante el uso del conversor analógico-digital. Para ello, creamos una carpeta de trabajo llamada “Práctica-11”, en la que incluimos todos los archivos esenciales. Utilizando Pico Visual Studio como entorno de desarrollo, implementamos y configuramos el código necesario tanto a nivel de hardware como de software para obtener datos del sensor, procesarlos y transmitirlos por el puerto USB.

0.2 Desarrollo de la practica.

0.2.1 Proceso paso a paso.

Creamos una nueva carpeta llamada ”Práctica-11”. A la que agregaremos los archivos importantes de la aplicación, estos archivos incluyen .vscode, pico-sdkimport.cmake, CMakeLists.txt y un archivo de código fuente en C. Este último se llamará ADC.c.

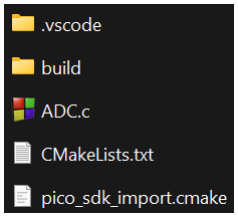

Figure 1: Archivos necesarios

Abrimos Pico Visual Studio y se cargará la carpeta de ejercicios.

Codigo en C.

Para la inclusión de librerías.
El programa lo comenzamos incluyendo librerías necesarias para trabajar con el microcontrolador Raspberry Pi Pico. Estas permiten usar funciones estándar de entrada/salida, inicializar periféricos y acceder al conversor analógico-digital.

Para la definición de constantes.
Definimos el pin al que está conectado el sensor de temperatura `(pin 26)`, el valor máximo que puede entregar el ADC `(4096 en una conversión de 12 bits)`, y el voltaje de referencia usado por el ADC `(3.3 voltios)`.

Para iniciar.

Dentro de la función principal, habilitamos la comunicación estándar y configuramos el ADC. También preparamos el pin 26 para lectura analógica y seleccionamos el canal correspondiente al pin 26 `(canal 0 del ADC)`.

Para el bucle principal.
El código entra en un bucle infinito en el que hace lo siguiente:
o Lee el valor digital del ADC, que representa el voltaje medido.
o Convierte ese valor digital a voltaje real usando una regla de tres basada en el valor máximo del ADC y el voltaje de referencia.
o Transforma el voltaje en temperatura. Se asume que el sensor entrega 10 mV por cada grado Celsius, por lo que se multiplica por 100.
o Imprime la temperatura y el voltaje separados por una coma.
o Fuerza la salida de datos inmediatamente para evitar retrasos por almacenamiento en buffer.
o Espera 500 milisegundos antes de hacer la siguiente lectura, lo que da aproximadamente dos lecturas por segundo.

Para finalizar.
Aunque el programa nunca sale del bucle, incluimos un return 0 por convención, indicando que el programa terminó sin errores.

CMakeList

Para establecer la versión mínima de `CMake`.
Establecemos la versión mínima requerida de CMake `(3.13)`. Esto nos asegura compatibilidad con las funciones que se usaremos más adelante.
Definimos el nombre del proyecto como `Practica_11`.

Para importar el SDK de Raspberry Pi Pico.
Incluimos un archivo especial `(pico_sdk_import.cmake)` que nos permite usar las herramientas y librerías del SDK oficial de Raspberry Pi Pico.

Para iniciar el SDK.
Iniciamos el SDK de la Pico para poder usar sus funciones, como entrada/salida, control del ADC, y otras funcionalidades específicas del microcontrolador.

Para agregar el archivo fuente.
Indicamos que el programa que se va a compilar se llama ADC y que su código está en el archivo ADC.c. Esto nos genera un ejecutable con ese nombre.

Para la vinculación de bibliotecas necesarias.
Enlazamos el ejecutable con las bibliotecas necesarias:
o `pico_stdlib:` nos proporciona funciones básicas como printf, sleep-ms, etc.
o `hardware_adc:` permite controlar el conversor analógico-digital del microcontrolador.

Para la configuración de la salida serial.
Habilitamos la salida de datos por USB y desactivamos la salida por UART. Esto no permite que los datos printf aparezcan por el puerto USB, como si fuera una terminal serial.

Para generar del archivo UF2.
Especificamos que se deben generar archivos binarios adicionales, incluyendo el archivo .uf2 que es el que arrastraremos directamente al Pico para cargar el programa.

0.3 Conclusion.

Al finalizar esta práctica, logramos construir exitosamente un programa funcional que permite al Raspberry Pi Pico leer datos analógicos desde un sensor de temperatura, convertirlos a una señal digital, calcular la temperatura correspondiente y enviarla en tiempo real por el puerto USB. Esta actividad nos permitió afianzar tanto el uso de herramientas de desarrollo como la interacción directa con periféricos mediante programación en C.

Debido a que la entrega del reporte se atraso, tuve que usar la Pico para otras practicas y desafortunadamente no tome fotos de los resultados al igual que mis compañeros cercanos a los que les pedí apoyo. Sin embargo, recuerdo que el profesor logro tomar video de mis resultados.

0.4 Anexos

ADC.c > main()

```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;
}
```

Figure 2: Codigo en C

CMakeLists.txt

```
cmake_minimum_required ( VERSION 3.13)

# Nombre del proyecto
project ( Practica_11 )

# Habilitar la Raspberry Pi Pico SDK
include ( pico_sdk_import.cmake )
pico_sdk_init ()

# Agregar el archivo fuente
add_executable ( ADC
ADC.c
)

# Agregar las bibliotecas estandar de la Pico
target_link_libraries ( ADC
pico_stdlib
hardware_adc )

# Habilitar la salida USB para el puerto serie
pico_enable_stdio_usb ( ADC 1)
pico_enable_stdio_uart ( ADC 0)

# Crear el archivo binario UF2
pico_add_extra_outputs ( ADC )
```

Figure 3: CMakeList