Instituto Politécnico Nacional

Escuela Superior de Cómputo

Practica 2.- LECTURA SERIAL DE UNA ENTRADA ANALÓGICA.

Sistemas en chip
Profesora: Ana Luz Barrales Lopez

Luciano Hernández Jonathan
Rodriguez Morales Iñaki

16 de octubre de 2024

1 Introducción

El manejo de entradas analógicas es una capacidad fundamental en los sistemas embebidos, ya que la mayoría de los sensores en el mundo físico generan señales analógicas que deben ser interpretadas y procesadas digitalmente. En los SoC, como los microcontroladores, la conversión de estas señales es gestionada por un Convertidor Analógico-Digital (ADC), que transforma las variaciones continuas en una señal discreta comprensible para el procesador. Esta funcionalidad es crucial en aplicaciones que requieren la monitorización de parámetros físicos como temperatura, luz, presión o, en este caso, la posición de un potenciómetro, permitiendo una interacción eficiente entre el entorno físico y el sistema digital.

Los SoC que incorporan la capacidad de manejar entradas analógicas son esenciales en el desarrollo de soluciones para dispositivos inteligentes, sistemas de control y automatización, proporcionando una base sólida para la integración de sensores en una amplia gama de proyectos de ingeniería.

1.1 Objetivos

- **Manejar una entrada analógica.** Aprender a leer señales analógicas provenientes de un dispositivo como el potenciómetro, utilizando el convertidor analógico-digital (ADC) del Arduino.
- **Mostrar los datos obtenidos en una salida digital (display)** Esto implica mapear los valores analógicos y mostrarlos de manera clara en el display utilizando pines digitales del Arduino.
- **Conocer los estados de un potenciómetro.** Este objetivo busca entender cómo un potenciómetro varía su resistencia en función de su posición, generando diferentes valores analógicos.
- **Conocer las funciones `AnalogRead()` y `SerialPrint()`.** Se pretende comprender e implementar la función `AnalogRead()` para capturar valores analógicos y `SerialPrint()` para enviar datos al monitor serial.

1.2 Materiales

Los materiales utilizados en la práctica son los siguientes:

- Arduino Uno
- Potenciómetro de 10k ohms
- 2 displays de 7 segmentos (cátodo común)
- 14 resistencias de 330 ohms
- Protoboard
- Cables de conexión
- Arduino IDE

2 Desarrollo de la Práctica

2.1 Explicación de las conexiones

La conexión del circuito consiste en conectar el potenciómetro al Arduino de la siguiente manera: el pin central del potenciómetro va al pin A0 del Arduino (entrada analógica), mientras que los otros dos pines del potenciómetro van a VCC (5V) y GND, respectivamente. Los dos displays de 7 segmentos se conectan a pines digitales del Arduino (D2-D13), utilizando resistencias limitadoras para proteger los LEDs de los displays. Estos pines digitales enviarán la señal digital a los displays de 7 segmentos.

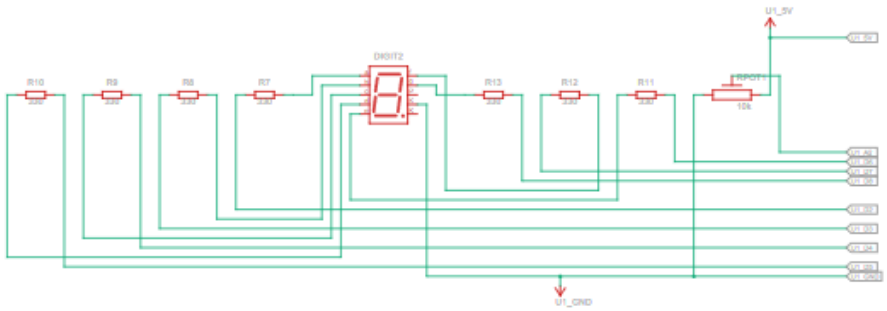

Figure 1: Diagrama del circuito: entrada analógica a salida digital.

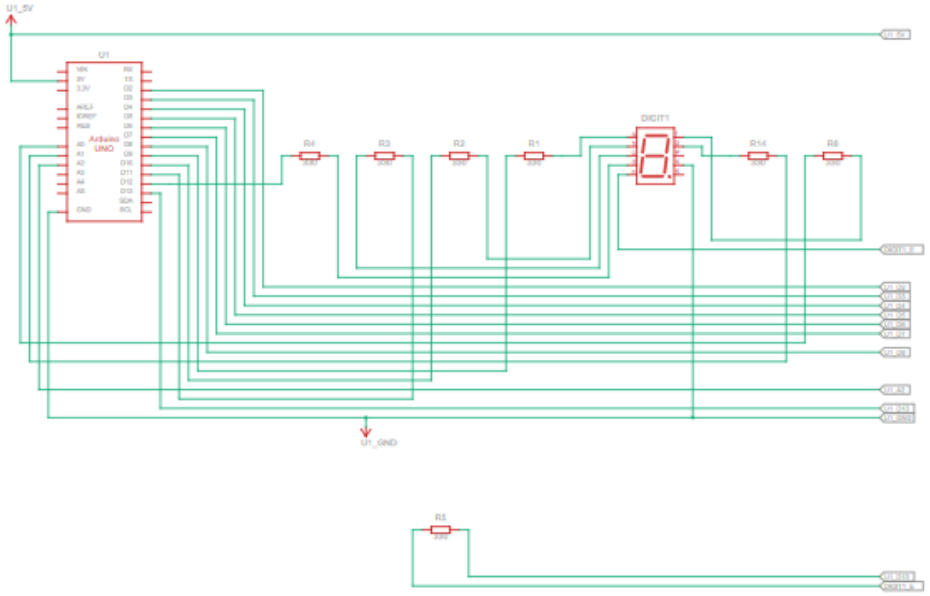

Figure 2: Diagrama del circuito: entrada analógica a salida digital.

2.2 Desarrollo del código

2.2.1 Configuración de Pines

El siguiente bloque de código define los pines digitales utilizados para controlar los displays de 7 segmentos. Los pines están asignados a las posiciones de los segmentos para

formar los números:

```
// Pines para los segmentos del primer display (unidades)
int segA_unidades = 2;
int segB_unidades = 3;
int segC_unidades = 4;
int segD_unidades = 5;
int segE_unidades = 6;
int segF_unidades = 7;
int segG_unidades = 8;

// Pines para los segmentos del segundo display (decenas)
int segA_decenas = 9;
int segB_decenas = 10;
int segC_decenas = 11;
int segD_decenas = 12;
int segE_decenas = 13;
int segF_decenas = A0;  // Usando A0 como pin digital
int segG_decenas = A1;  // Usando A1 como pin digital

int potPin = A2;
```

2.2.2 Array que Contiene los Segmentos para los Números 0-9

Este bloque define un array bidimensional que representa los segmentos de cada número del 0 al 9. Cada sub-array tiene siete valores, donde cada valor corresponde a un segmento (A-G) del display. Un valor de 1 enciende el segmento, mientras que 0 lo apaga.

```
// Array que contiene los segmentos para los n meros 0-9
int numeros[10][7] = {
  {1, 1, 1, 1, 1, 1, 0}, // 0
  {0, 1, 1, 0, 0, 0, 0}, // 1
  {1, 1, 0, 1, 1, 0, 1}, // 2
  {1, 1, 1, 1, 0, 0, 1}, // 3
  {0, 1, 1, 0, 0, 1, 1}, // 4
  {1, 0, 1, 1, 0, 1, 1}, // 5
  {1, 0, 1, 1, 1, 1, 1}, // 6
  {1, 1, 1, 0, 0, 0, 0}, // 7
  {1, 1, 1, 1, 1, 1, 1}, // 8
  {1, 1, 1, 0, 0, 1, 1}  // 9
};
```

2.2.3 Inicialización de los Pines

En la función setup(), se configuran todos los pines de los segmentos de los displays como salidas. Esto es esencial para que el Arduino pueda enviar señales a los pines y controlar los segmentos del display.

```
// Inicializaci n de los pines
void setup() {
  // Configurar pines de los segmentos de unidades como salidas
  pinMode(segA_unidades, OUTPUT);
  pinMode(segB_unidades, OUTPUT);
  pinMode(segC_unidades, OUTPUT);
  pinMode(segD_unidades, OUTPUT);
  pinMode(segE_unidades, OUTPUT);
  pinMode(segF_unidades, OUTPUT);
```

```
  pinMode(segG_unidades, OUTPUT);

  // Configurar pines de los segmentos de decenas como salidas
  pinMode(segA_decenas, OUTPUT);
  pinMode(segB_decenas, OUTPUT);
  pinMode(segC_decenas, OUTPUT);
  pinMode(segD_decenas, OUTPUT);
  pinMode(segE_decenas, OUTPUT);
  pinMode(segF_decenas, OUTPUT);
  pinMode(segG_decenas, OUTPUT);
}
```

2.2.4 Función para Mostrar un Número en el Displays

Este es el bucle principal del programa que se ejecuta repetidamente. Primero, lee el valor del potenciómetro usando analogRead(), que devuelve un valor entre 0 y 1023. Luego, se utiliza la función map() para escalar ese valor a un rango entre 0 y 10. Se calculan las decenas y las unidades a partir del valor escalado, y se llaman a las funciones mostrarUnidades() y mostrarDecenas() para visualizar el número en los displays. Finalmente, se incluye un pequeño retraso de 10 ms antes de la próxima iteración del bucle.

```
// Funci n para mostrar un n mero en el display de unidades
void mostrarUnidades(int numero) {
  digitalWrite(segA_unidades, numeros[numero][0]);
  digitalWrite(segB_unidades, numeros[numero][1]);
  digitalWrite(segC_unidades, numeros[numero][2]);
  digitalWrite(segD_unidades, numeros[numero][3]);
  digitalWrite(segE_unidades, numeros[numero][4]);
  digitalWrite(segF_unidades, numeros[numero][5]);
  digitalWrite(segG_unidades, numeros[numero][6]);
}

// Funci n para mostrar un n mero en el display de decenas
void mostrarDecenas(int numero) {
  digitalWrite(segA_decenas, numeros[numero][0]);
  digitalWrite(segB_decenas, numeros[numero][1]);
  digitalWrite(segC_decenas, numeros[numero][2]);
  digitalWrite(segD_decenas, numeros[numero][3]);
  digitalWrite(segE_decenas, numeros[numero][4]);
  digitalWrite(segF_decenas, numeros[numero][5]);
  digitalWrite(segG_decenas, numeros[numero][6]);
}
```

2.2.5 Bucle principal

La función `mostrarDigito()` activa los segmentos correctos en función del dígito a mostrar. Cada número del 0 al 9 tiene una combinación específica de segmentos encendidos:

```
void loop() {
  // Leer el valor del potenci metro (0-1023)
  int valorPot = analogRead(potPin);

  // Mapear el valor del potenci metro de 0-1023 a 0-10 (n mero
    m ximo 10)
```

```
6   int valorEscalado = map(valorPot, 0, 1023, 0, 10);
7
8   // Calcular decenas y unidades
9   int decenas = valorEscalado / 10;
10  int unidades = valorEscalado % 10;
11
12  // Mostrar las unidades en el primer display
13  mostrarUnidades(unidades);
14
15  // Mostrar las decenas en el segundo display
16  mostrarDecenas(decenas);
17
18  delay(10);
19 }
```

3 Display de 7 segmentos

Para entender mejor la configuración y código del proyecto es importante conocer como está compuesto un display de 7 segmentos, el cual se puede observar a continuación.

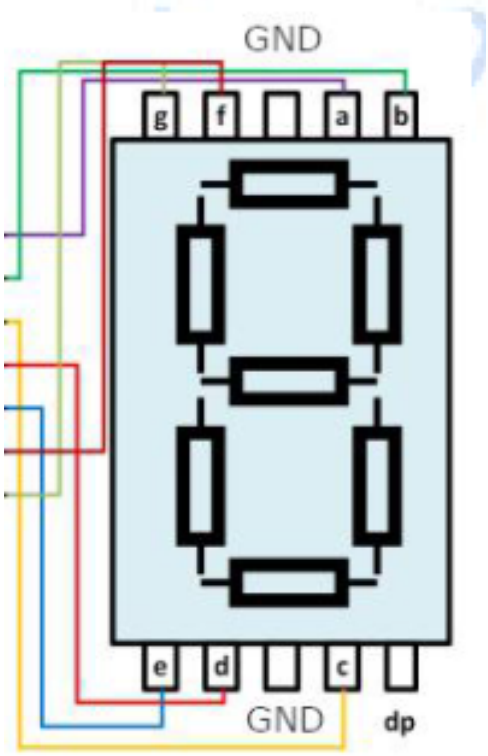

Figure 3: Datasheet display de 7 segmentos.

Por consecuente, la configuración para controlar los segmentos del display se sigue de acuerdo al codigo previo, tomando en cuenta cuales secciones son las que deseamos que esten encendidas de acuerdo al caso.

3.1 Simulación

La simulación del circuito fue realizada en la plataforma **Tinkercad**, un entorno de simulación en línea que permite construir y ejecutar circuitos electrónicos de manera virtual. En el que previo a implementarse con componentes reales, se llevo a cabo en esta plataforma.

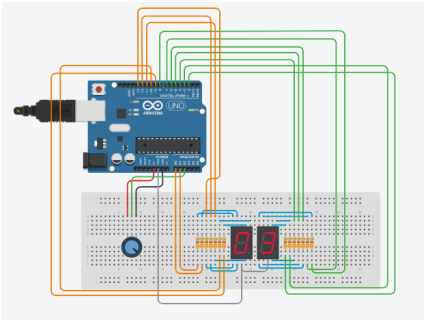

Figure 4: Funcionamiento del cricuito

Primera imagen donde se puede observar el funcionamiento del circuito de manera correcta en la simulación.

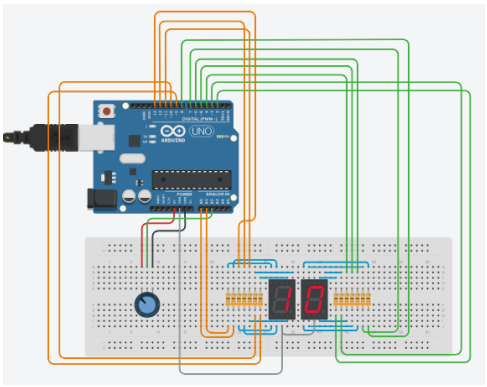

Figure 5: Funcionamiento del cricuito

Segunda imagen donde se puede observar el funcionamiento del circuito de manera correcta en la simulación con el potenciometro al maximo.

3.2 Implementación

Una vez implementado la conexion y desarrollo del codigo dentro de la simulación, se elaboró el circuito de manera practica con componentes reales para observar su funcionamiento, a continuación, se muestran las imagenes del citcuito en funcionamiento conectado a aun arduino uno.

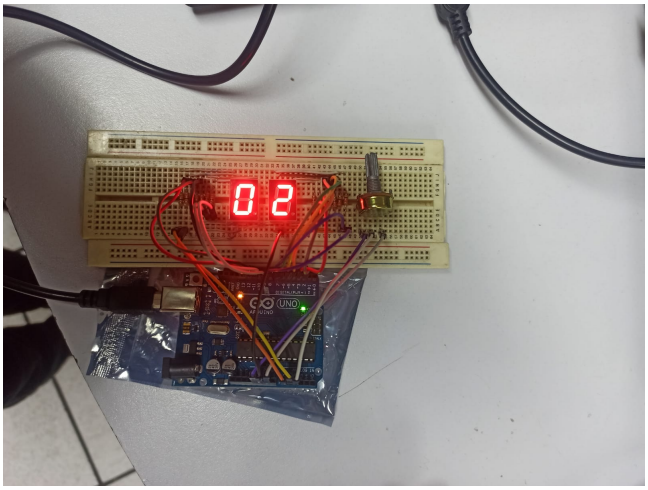

Figure 6: Funcionamiento del cricuito

En esta primera imagen se ve como el potenciometro se encuentra cerca de su recorrido inicial, por lo que el display de 7 segmentos muestra un número 2.

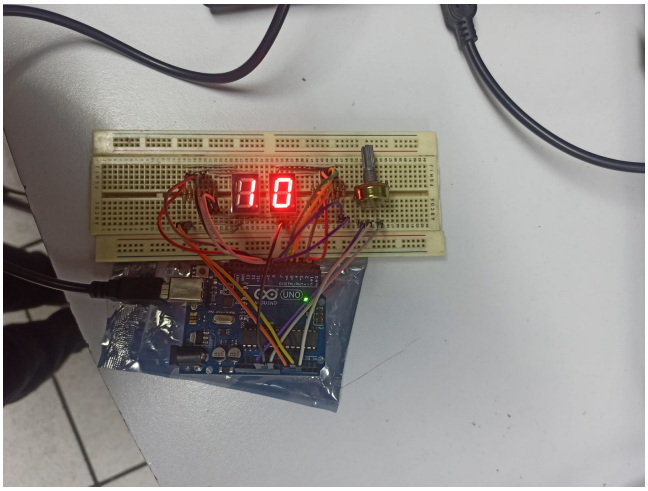

Figure 7: Funcionamiento del cricuito

En esta segunda figura se puede observar como al llegar al limite máximo del potenciometro, el primer display se coloca en 0, y el siplay de las decenas se pone en 1, formando el numero 10 tal y como se encuentra programado.

Después de lo anterior, queda documentado que la practica de entrada y lectura de una señal analógica pudo completarse de manera exitosa, obteniendo los resultados esperado de esta práctica.

4 Conclusiones

- **Morales Rodriguez Iñaki** Durante la práctica, logramos con éxito el manejo de una entrada analógica utilizando un potenciómetro, lo que permitió una lectura precisa de su posición y la conversión de la señal analógica a un valor digital. Este objetivo fue de suma importancia para entender el funcionamiento del Convertidor Analógico-Digital (ADC) de un Arduino. La capacidad de medir variaciones en

la resistencia del potenciómetro y traducir esas mediciones en un valor digital fue importante para llevar a cabo la practica. Este proyecto no solo nos ayudó a mejorar la comprensión teórica, sino que también práctica en el que implementamos nuestros conocimientos.

- **Luciano Hernández Jonathan** Durante la elaboracion de esta practica logramos cumplir el objetivo de mostrar los datos obtenidos en un display de 7 segmentos, la cual tuvo su grado de complejidad, pero que pudimos llevar a cabo con éxito. Al implementar la visualización en tiempo real, se facilitó la interpretación de los datos, lo que demostró la efectividad del diseño del circuito y el control de las salidas digitales del Arduino. Además el uso de un simulador nos ayudó enormemente a evitar tener errores al implementarlo en un circuito real, pues al haberse desarrollado en primera instancia en una simulación, el margen de error es minimo al realizar las conexiones en un entorno real.

5 Referencias

1 A. Hernández, "Entradas Analógicas y Convertidor Analógico-Digital (ADC) en Arduino," Control Automático Educación, 17-Oct-2023. [Enlace: https://control automaticoeducacion.com/sistemas-embebidos/arduino/entradas-analogicas-adc/].

- Código Electrónica. "Señal Digital a Display de 7 Segmentos."https://tinyurl.com/4n92v39d, Accedido el 17 de octubre de 2024.

- G. M.1-5611CURG. "Microcontroladores." AG Electrónica, https://agelectronica.lat/pdfs/textos/G/GM1-5611CURG.PDF. Accedido el 17 de octubre de 2024.