

LABORATORIO DE PRINCIPIOS DE MECATRÓNICA

18 de febrero de 2022

Práctica #2

Arduino

Grupo: 1

L002

Estudiante:

■ García Cortez

Alejandro

■ Hermida Flores

Manuel Joaquín

■ Chicatti Avendaño

Josué Doménico

Profesor:

Benito Granados–Rojas

Índice

1. Introducción	2
2. Experimentos y Simulaciones	2
2.1. Convertidor analógico digital	2
2.2. Liquid-Crystal Display (LCD)	3
2.3. Servomotor	4
3. Conclusiones	6
4. Enlaces externos	6



1. Introducción

Durante el primer laboratorio aprendimos que era un controlador y manejamos de manera introductoria un ambiente de desarrollo, IDE, para Arduino. Esta segunda práctica complementa lo aprendido al hacer uso de las librerías presentes en Arduino, las cuales nos añadirán funcionalidad. Estas funciones nos permitirán trabajar con más elementos en nuestra tarjeta o externos a ella, en este caso, sensores y actuadores.

Esta práctica consiste de tres ejercicios que manejan distintos componentes. Estos componentes pueden funcionar para adquirir señales o desplegar información. En otras palabras, pueden ser entradas o salidas.

La primera parte consiste en el uso del potenciómetro y el convertidor analógico-digital para leer los valores del primero. Posteriormente, se usa un LCD para desplegar texto de distintas formas. Finalmente, se analiza el comportamiento de un servo-motor para integrar su movimiento con los valores dados por un potenciómetro.

2. Experimentos y Simulaciones

2.1. Convertidor analógico digital

El primer ejercicio consistió en el uso del convertidor analógico digital junto a un potenciómetro y un LED. La conexión de estos elementos se realizó como indicaba el diagrama de la práctica. Los elementos dentro de nuestra protoboard se pueden apreciar en la Figura 1.

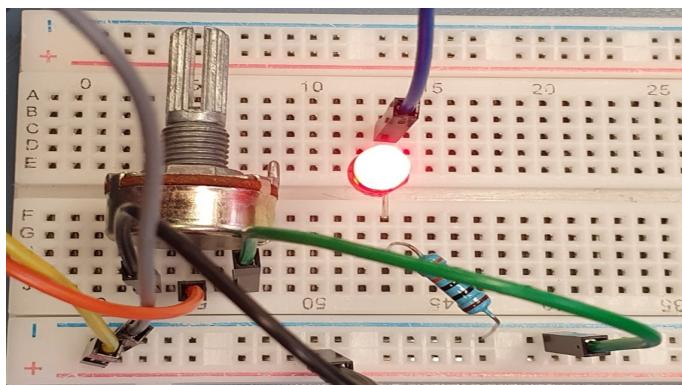


Figura 1: Conexión en protoboard

En la primera parte se analizó un código proporcionado dentro de la práctica. Se observó que el código aprovecha la conexión del potenciómetro en un pin analógico y lee su valor mediante la función `analogRead()`. Finalmente, imprime esos valores en el monitor serial. El código se muestra a continuación:

El código fue modificado para desplegar el voltaje correspondiente con la posición del potenciómetro. La modificación incluyó una regla de tres al saber los rangos de

voltaje (0 a 5V) y del valor leído (0 a 1023). El nuevo código se aprecia a continuación:

```
int analogPin = A0;
int val = 0;

void setup() {
    Serial.begin(9600); // setup serial
}

void loop() {
    val = analogRead(analogPin); // read the input pin
    Serial.print ("Conversion analogico-digital: ");
    aux = val*5/1023;
    Serial.println(aux);
    delay(500);
}
```

Para hacer que el LED prendiera solo si el voltaje que entraba a A0 era mayor a 3V se añadió un bloque condicional if-else. Dentro de la condición del verdadero se usó digitalWrite en *HIGH*, mientras que en lo falso, se usó *LOW*.

El último ejercicio requirió hacer que el LED encendiera proporcionalmente al voltaje en el puerto A0. Se volvió a usar como base el código de la segunda parte a la cual se le añadió analogWrite, pero usando el valor leído (val) entre 4 como segundo argumento. La división fue necesario, ya que la función recibe un número entre 0 y 255. De esa manera, el ciclo de trabajo de la señal PWM que manda la función analogWrite fue mapeado con los valores del potenciómetro.

2.2. Liquid-Crystal Display (LCD)

Para empezar a hacer este ejercicio debemos inicializar los pines de la interfaz del LCD. Para esto debemos incluir la librería *LiquidCrystal.h* que tiene todos los métodos para operar el LCD y hacer todas las partes de esta parte de la práctica.

Para configurar el LCD debimos conectar los pines VSS, VCC, RS, RW, E, D4, D5, D6 y D7 al Arduino. Además, tuvimos que conectar el pin VEE al potenciómetro para controlar el contraste del fondo.

Primero, usamos el método constructor para crear una instancia de *LiquidCrystal* llamada lcd. En los parámetros debemos indicar en qué pines del Arduino conectamos los pines del LCD. Luego, tenemos que configurar cuantas filas y columnas tendrá esta instancia. Cada vez que queramos imprimir sobre una línea, debemos colocar el cursor en la fila y columna en la que se empezará a escribir. Para hacer el texto parpadear simplemente debemos dar un delay después del método print y

luego usar el método clear con su respectivo delay.

```
#include <LiquidCrystal.h> // include the library code:  
LiquidCrystal lcd(12, 11, 5, 4, 3, 2); // initialize the interface pins  
  
void setup() {  
    lcd.begin(16, 2); // set up the LCD's number of columns and rows:  
}  
  
void loop() {  
    lcd.setCursor(1, 0);  
    lcd.print("Principios de"); // Print a message to the LCD.  
    lcd.setCursor(2, 1);  
    lcd.print("Mecatronica");  
    delay(750);  
    lcd.clear();  
    delay(250);  
}
```

En la segunda parte del ejercicio debimos escribir código análogo al anterior pero añadiendo el método siguiente, el cual iba moviendo el texto hacia la izquierda:

```
lcd.scrollDisplayLeft();
```

2.3. Servomotor

En la última parte de la práctica se revisó la librería de *Servo*. Esta librería nos permite controlar un servomotor, los cuales son un tipo de actuador rotacional que se caracterizan por permitir el control preciso de la posición del actuador.

El primer ejercicio de esta parte consistió en un simple programa que iba aumentando la posición del servomotor en un paso de tamaño. Los servomotores aceptan posiciones entre 0 y 180. Este programa, al llegar a la posición 180, comenzaba a disminuir las posiciones hasta llegar a la 0 y repetía.

El segundo y tercer ejercicio consistieron en la integración de un convertidor analógico digital, del LCD y de un servomotor en una misma actividad. Se tenía que controlar la posición del servomotor con un potenciómetro al mismo tiempo que se mostraban los valores, tanto de voltaje leido como de ángulo, en el LCD. Para lograr esto se utilizó el siguiente código.

```
#include <Servo.h>  
#include <LiquidCrystal.h>  
  
LiquidCrystal lcd(12, 11, 5, 4, 3, 2);
```

```
Servo myservo;  
  
int analogPin = A0;  
double val = 0;  
double vol = 0;  
double pos = 0;  
  
void setup() {  
    myservo.attach(9);  
    Serial.begin(9600);  
    lcd.begin(16, 2);  
}  
  
void loop() {  
    val = analogRead(analogPin);  
    vol = val*5.0/1023.0;  
    pos = map(val,0,1023,0,180);  
    myservo.write(pos);  
  
    delay(50);  
    lcd.setCursor(0, 0);  
    lcd.print("Vol");  
    lcd.setCursor(8, 0);  
    lcd.print(String(vol));  
    lcd.setCursor(0, 1);  
    lcd.print("Pos");  
    lcd.setCursor(8, 1);  
    lcd.print(pos);  
}
```

El código comienza importando las librerías de Servo y de LiquidCrystal, las cuales nos permiten controlar el servomotor y el LCD que usaremos. Después declaramos los objetos con los que se controlaran el servomotor y el LCD, así como otras variables. El código funciona leyendo constantemente el valor de potenciómetro, ubi-

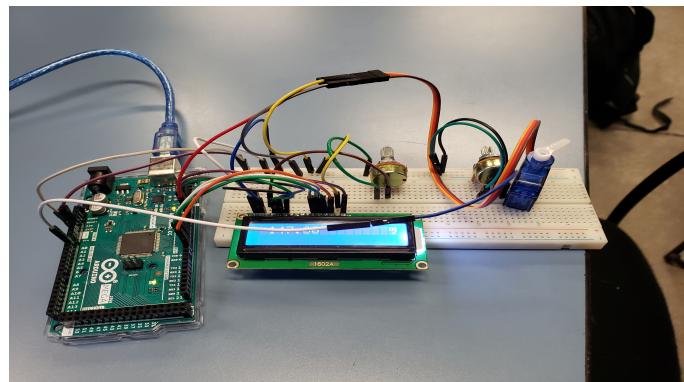


Figura 2: Implementación Servomotor, LCD y Potenciómetro

cado en el puerto A0. Utilizando esta valor lo escalamos, de forma que obtenemos un voltaje y posición de acuerdo a la lectura. La posición es mandada al servo a través de la instrucción `myservo.write(pos)`. Por otro lado, imprimimos en el LCD las etiqueta y valores de `pos` y `vol` auxiliandonos de `lcd.setCursor()` y de `lcd.print()`. Termina el código y se vuelve a ejecutar.

3. Conclusiones

El concluir esta práctica podemos decir que cumplimos que todos sus objetivos. Manejamos el monitor serial para observar los valores del potenciómetro y desplegamos texto de distintas formas en el LCD. También usamos los periféricos ADC y PWM para adquirir o desplegar información.

Una parte esencial de programar en Arduino es aprender a usar la documentación de todas las librerías que ya están incorporadas en el IDE. Al informarnos de los métodos ya incluídos en las librerías podemos hacer código más eficiente y con menos errores. Además, nos ahorramos repetir código, trabajo y tiempo que ya fue realizado antes.

Un ejemplo de esto son las librerías usadas en esta práctica. Sin embargo, debemos ser cuidadosos y entender la documentación. Por ejemplo, en el primer y tercer ejercicio optamos por usar nuestra propia implementación de la función `map` puesto que el tipo de datos que usaba la función ya implementada era inadecuada para obtener los resultados que buscábamos.

4. Enlaces externos

<https://github.com/ManoHF/lab-mecatronica>