Laboratorio de Microprocesadores - 86.07

Trabajo Práctico Obligatorio N°5

Uso del ADC

Profesor:			Ing. Guillermo Campiglio								
Cuatrimestre/Año:			1°/2020								
Turno de las clases prácticas			Miércoles								
Jefe de trabajos prácticos:			Ing. Pedro Ignacio Martos								
Docente guía:			Ing. Fabricio Baglivo, Ing. Fernando Pucci								
Autores			Seguimiento del proyecto								
Nombre	Apellido	Padrón									
Santiago	López	100566									

Observaciones:

.....
.....
.....
.....
.....
.....
.....
.....
.....

Fecha de aprobación				Firma J.T.P

Coloquio	
Nota final	
Firma profesor	

Índice

1. Objetivo

Representar una señal analógica de forma digital, mediante el uso del conversor analógico digital integrado en el ATmega328p.

2. Descripción

Se recibe una señal analógica entre 0V y VCC, la cual varía de acuerdo a un potenciómetro logarítmico manejado a mano. Esta señal es leída por el conversor analógico digital integrado en el microcontrolador, y sale con una definición de 10 bits, los cuales se procesan para llegar a una señal de 6 bits de defición, reprensentada mediante 6 LEDs escribiendo de forma binaria valores entre 0 y 63.

Para enviar la señal a los LEDs en el momento adecuado se utilizó la interrupción integrada del ADC, la cual se dispara en cuanto la conversión de la señal está lista. Luego de sacar el valor digitalizado se vuelve a leer la señal de entrada utilizando la configuración de auto-trigger.

3. Diagrama de conexiones en bloque

Las conexiones siguieron el esquema de la Figura 1.

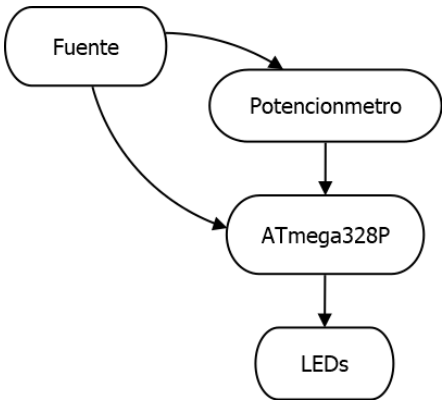

Figura 1: Diagrama de conexiones en bloque.

4. Esquemático

La Figura 2 muestra las conexiones eléctricas efectuadas en el práctico. Los componentes utilizados se encuentran en la sección siguiente. En caso de querer utilizarse la resistencia de *pull-up* interna del micro, se conectaría el puslador entre el pin de la interrupción y tierra, seteando el micro para detectar flancos descendentes en lugar de flancos ascendentes. De esta forma se ahorra el uso del resistor externo de 10kΩ.

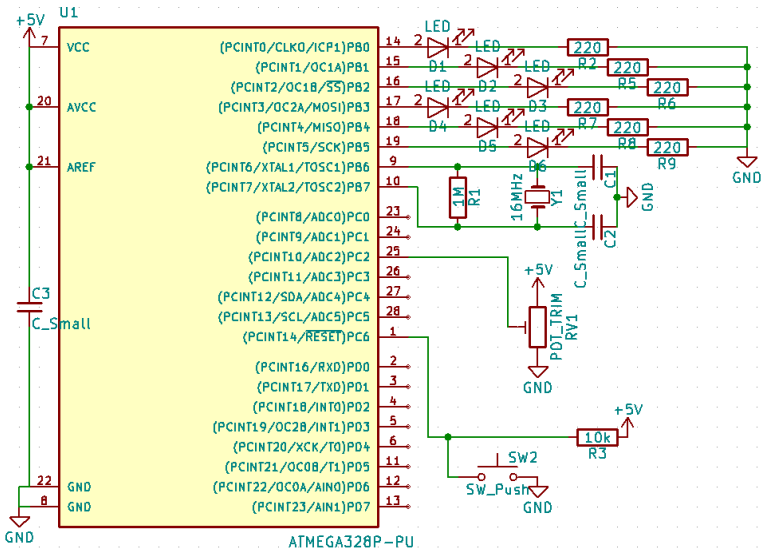

Figura 2: Circuito esquemático.

5. Listado de componentes

Los componentes utilizados fueron los listados a continuación:

- Placa de desarrollo Arduino UNO
- Resistores de 220Ω
- LEDs de color rojo
- Potenciometro logarítmico
- Protoboard
- Cables unipolares

6. Diagrama de flujo

En la Figura 3 se presentan los pasos a seguir por el programa. La primera rutina lleva los setups de los puertos y del stack del micro, mientras que la segunda rutina representa la interrupción del producida por el final de la conversión analógico-digital.

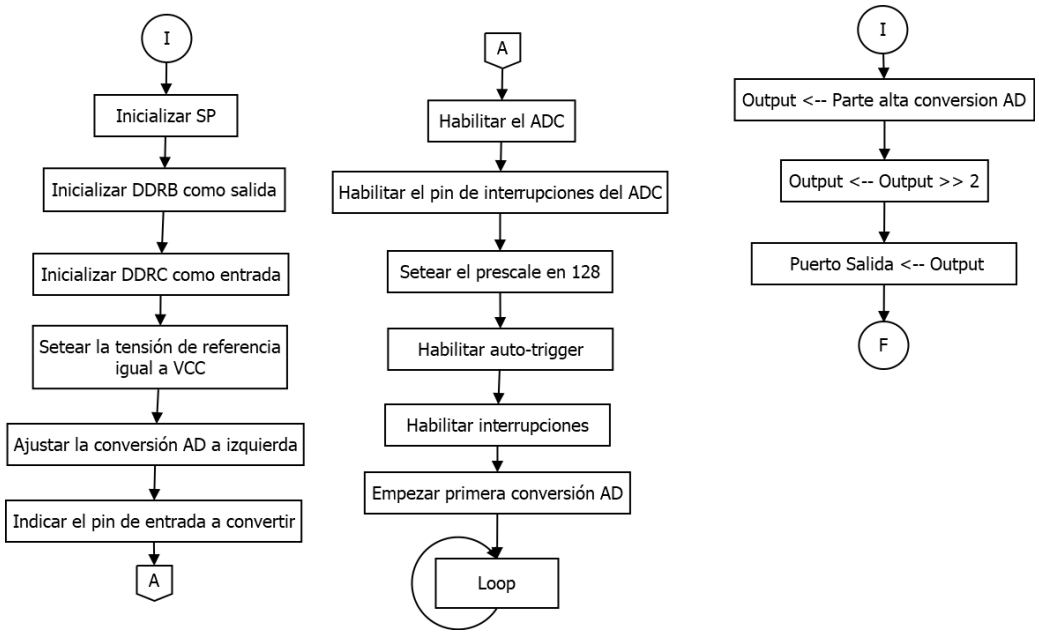

Figura 3: Diagrama de flujo.

7. Código fuente

El programa consiste en el setup de los registros, y luego tan solo esperar a las interrupciones del ADC, para indicar el valor convertido a través del puerto B.

Se decidió utilizar las interrupciones del ADC para no tener que implementar un contador para saber cuando una conversión estuviera hecha. De esta forma el código se volvió mucho más compacto, ya que el llamado a las interrupciones se hace de forma automática, como ya se experimentó en el trabajo práctico previo.

Para expresar la salida en 6 bits, en lugar de los 10 bits que utiliza el ADC, se ajustó el conversor a izquierda, con el fin de descartar fácilmente los dos bits menos significativos, y luego de copiar los 8 bits más significativos a un registro, se hizo un corrimiento de 2 bits a derecha, con tal de que los bits perdidos fueran los más cambiantes, y los menos representativos del valor leído.

```
 1: .include "m328pdef.inc"
 2:
 3: .dseg
 4:
 5: .def conf = r16
 6: .def output = r17
 7: .def aux = r18
 8: .equ led_port = PORTB
 9: .equ led_reg = DDRB
10:
11: .macro init_ports
12:     ldi conf, 0xff
13:     out DDRB, conf
14:     clr conf
15:     out PORTB, conf
16:     out DDRC, conf
17: .endmacro
18:
19: .macro init_adc
20: // Vref = VCC, ajusto la conversion a izq
21: // pongo el mux en el ADC2
22:     ldi conf, 0x62
23:     sts ADMUX, conf
24: // adc enable, interrupt enable, prescale 128 por usar clock de 16MHz
25:     ldi conf, 0xAF
26:     sts ADCSRA, conf
27: // free running mode
28:     clr conf
29:     sts ADCSRB, conf
30: .endmacro
31:
32:
33: .macro init_sp
34:     ldi conf, low(RAMEND)
35:     out spl, conf
36:     ldi conf, high(RAMEND)
37:     out sph, conf
38: .endmacro
39:
40: .cseg
41: .org 0x0000
42:     jmp main
43: .org 0x002A // no econtre esta etiqueta
44:     jmp adc_isr
45:
46:
47: .org INT_VECTORS_SIZE
48: main:
49:
50:     init_sp
51:     init_ports
52:     init_adc
53:     sei
54: // 1era conversion
55:     lds conf, ADCSRA
56:     ori conf, 0x40
57:     sts ADCSRA, conf
58:
59: here:
60:     nop
61:     nop
62:     rjmp here
63:
64: adc_isr:
65:     lds output, ADCH
66:     lsr output
67:     lsr output
68:     out led_port, output
69: reti
```

8. Costos

A continuación se presenta un listado de los costos de los componentes utilizados en el práctico.

- Arduino UNO - $850
- Resistores - $50
- LEDs rojo - $120
- Potenciometro - $50
- Protoboard - $240
- Cables unipolares - $150

Sumando un costo total de $1460.

9. Conclusiones

El conversor analógico digital resulta de fácil uso en conjunto a las interrupciones, evitando la implementación de un timer, ya que se decidió por muestrear lo más rápido posible la señal entrante.

El uso de 6 de los 10 bits de resolución del ADC permitió abaratar los costos del práctico, debido a la menor cantidad de componentes utilizados. Además, dadas pequeñas variaciones en la señal de entrada, el circuito respondía de la forma esperada, demostrando así que no se hubiera detectado un cambio significativo en el valor de salida por el uso de más bits.