Universidad de Buenos Aires
Facultad de Ingeniería
Año 2020 - 1^{ero} cuatrimestre

Laboratorio de Microprocesadores (86.07)

TRABAJO PRÁCTICO OBLIGATORIO 5

ESTUDIANTE:
Pintos Gaston 99711
gmpintos@fi.uba.ar

Índice

1. Objetivos

El trabajo práctico consiste en generar un programa que sea capaz utilizar el conversor analógico digital interno del microprocesador. Con este programa se visualiza a través de 6 LED conectados al puerto B el valor binario tomado del ADC al ser variada la señal con un potenciómetro. Se utilizo el microcontrolador *atmega 328P* de la placa *Arduino*. A lo largo del desarrollo se presentaran las herramientas utilizadas para la realización del proyecto.

2. Desarrollo:

2.1. Herramientas de Software:

Para el siguiente proyecto se utilizo la herramienta *AtmelStudio* para desarrollar el código en lenguaje *Assembly*. La misma herramienta permite simular el código escrito con lo cual fue posible realizar pruebas a lo largo del desarrollo. Además se utilizo la herramienta *AVRdude* para poder montar el programa en el microcontrolador.

2.1.1. Código de Montaje:

Se utilizo el siguiente código para cumplir la tarea asignada:

```
.DEVICE ATmega328P

; Vector de Main
.ORG 0
            RJMP Main

Main:
;====== Inicializ Stack====
    LDI R16,low(RAMEND)
    OUT SPL,R16
    LDI R16, high(RAMEND)
    OUT SPH, R16

;===== Seteo puertos ============
    LDI R16, 0xFF
    OUT DDRB, R16    ;Puerto D como salida
    LDI R16, 0x00
    OUT DDRC, R16    ;Puerto C como entrada

;==== Configuro ADC ========
    LDI R16, (1<<ADEN)|(1<<ADPS2)|(1<<ADPS2)|(1<<ADPS1)|(1<<ADPS0)
    STS ADCSRA, R16
    LDI r16, (0<<REFS1)|(1<<REFS0)|(1<<ADLAR)|(0<<MUX3)
        |(0<<MUX2)|(0<<MUX1)|(0<<MUX0)
    STS ADMUX, R16  ;Entrada - justificado a derecha

READ_ADC:
    LDS R16, ADCSRA ;leo el registro
    SBR R16, (1<<ADSC)       ;Pongo un 1 en el bit ADSC
    STS ADCSRA, R16 ;Comienza la conversion

KEEP_POLING:
    LDS R16, ADCSRA
    SBRS R16,ADIF            ;Verifico final de conversion
    RJMP KEEP_POLING

    SBR R16, ADIF
    STS ADCSRA, R16    ;Sobre escribo ADIF con 1

    LDS     R17, ADCH          ;Leo los registros convertidos
```

```
    LSR R17
    LSR R17
    OUT     PORTB, R17
    RJMP READ_ADC       ; Vuelve a leer
```

Se utilizo la instrucción *.DEVICE* para identificar el tipo de microcontrolador utilizado. Con la directiva *.ORG* se seteo el *stack pointer* al inicio de la memoria flash donde se ubican las directivas del programa.

La primera tarea del *Main* es inicializar el Stack. Se configuran el puerto B como salida, en donde seran conectados los 6 LED, y el puerto C como entrada ya que contiene la entrada al conversor analógico-digital.
Para configurar el conversor, se tuvieron en cuenta el registro ADCSRA y ADMUX.
Dentro del registro ADCSRA se activan el bit ADEN para iniciar la conversion y los bits ADPS 2:0 que configuran la escala de conversión. Con esto se obtiene una frecuencia de:

$$\frac{16MHz}{128} = 125KHz \tag{1}$$

Con lo cual se obtiene una frecuencia de muestreo menor al limite de $200KHz$.
En el registro ADMUX, se activaron: el bit ADLAR para obtener los datos de los registros ADCH y ADCL justificados a derecha, el bit REFS1 para obtener una tension de referencia igual a V_{cc}, los bits MUX0:3 fueron puestos en 0 para utilizar el pin correspondiente a ADC0 como entrada al conversor analógico.
La primera tarea del programa es activar la conversión de la señal activando el bit ADSC correspondiente al registro ADCSRA. Luego se ingresa a un loop llamado $KEEP\_POLING$ encargado de verificar el final de la conversión según el valor en el bit ADIF del registro ADCSRA. Una vez confirmada la conversión, se sobre escribe el bit ADIF con un 1 para poder borrar el fLAG. Después se lee el registro ADCH, se rotan los bits 2 veces ya que contamos con solo 6 LED en vez de 8 LED, y se imprime el resultado por el puerto B.

2.2. Uso de interrupciones:

Análogo al desarrollo de código en la sección anterior, se utilizaron interrupciones para invocar al conversor ADC. Esto hace un uso mas eficiente de las funciones del microcontrolador al evitar métodos de polling.

```
.DEVICE ATmega328P

; Vector de Main
.ORG 0
    RJMP Main

.ORG ADCCaddr
    RJMP ADC_INT_HANDLER

Main:
;====== Inicializ Stack====
    LDI R16,low(RAMEND)
    OUT SPL,R16
    LDI R16, high(RAMEND)
    OUT SPH, R16

;===== Seteo puertos ==========
    LDI R16, 0xFF
    OUT DDRB, R16    ;Puerto B como salida
    LDI R16, 0x00
    OUT DDRC, R16    ;Puerto C como entrada

;====== Habilito interrupciones =======
    SEI

;==== Configuro ADC =======
    LDI R16, (1<<ADEN)|(1<<ADIE)|(1<<ADPS2)
    |(1<<ADPS2)|(1<<ADPS1)|(1<<ADPS0)
    STS ADCSRA, R16          ;Habilito ADC y selecciono ck/128
    LDI r16, (0<<REFS1)|(1<<REFS0)|(1<<ADLAR)
    |(0<<MUX3)|(0<<MUX2)|(0<<MUX1)|(0<<MUX0)
```

```
        STS ADMUX, R16          ;Entrada - justificado a derecha

        LDS R16, ADCSRA         ;leo el registro
        SBR R16, (1<<ADSC)      ;Pongo un 1 en el bit ADSC
        STS ADCSRA, R16         ;Comienza la conversion

    LOOP:
        RJMP LOOP

    ADC_INT_HANDLER:
        LDS     R17, ADCH               ;Leo los registros convertidos
        LSR R17
        LSR R17
        OUT     PORTB, R17

        LDS R16, ADCSRA         ;leo el registro
        SBR R16, (1<<ADSC)  ;Pongo un 1 en el bit ADSC
        STS ADCSRA, R16         ;Comienza la conversion
        RETI
```

El desarrollo del código es similar al punto anterior. EN diferencia al mismo, se encuentra que en el registro ADSCRA se activa el bit ADIE que es el encargado de activar las interrupciones para el conversor. Además se activan las interrupciones externas con la instrucción SEI y la rutina de lectura, desplazamiento e impresión de valores se realiza al generarse la interrupción.

2.2.1. Montaje del código:

Paecuencia dera el montaje del código con la herramienta *AVR DUDE* a través del *ATMEL STUDIO* se utilizaron los siguientes comandos:
"C:9-Laboratorio de Microcomputadoras-6.3-mingw32(1).exe"-C
"C:9-Laboratorio de Microcomputadoras-6.3-mingw32(1).conf" -v
-p atmega328p -c arduino -P COM3 -D
-U flash:w:"$(MSBuildProjectDirectory)$(Configuration)$(OutputFileName).hex":i

Con esta herramienta fue posible que el código escrito fuera procesado por el microcontrolador para poder ejecutar las tareas programadas.

2.2.2. Diagramas de flujo:

Para una comprensión de manera gráfica se presenta a continuación un diagrama de flujo del código previamente explicado.

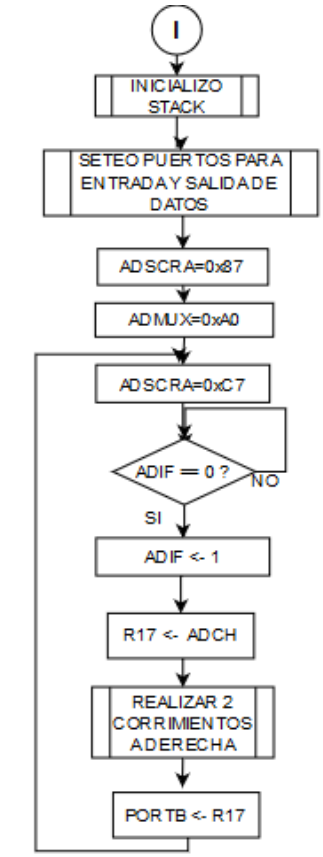

Figura 1: Diagrama de flujo de código utilizando poling.

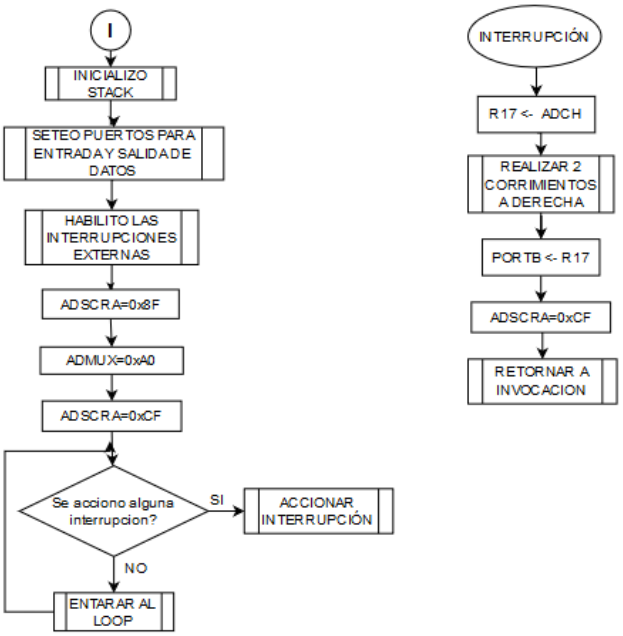

Figura 2: Diagrama de flujo de código utilizando interrupciones.

2.3. Herramientas de Hardware:

Para realizar el proyecto se utilizaron el microcontrolador de la placa *Arduino Uno*, un protoboard donde se montaron todos los elementos, 2 diodos led, un pulsador, una resistencia de $10K\Omega$ y 2 resistencias de 220Ω. Para la alimentación y trasmisión del código se utilizo el puerto USB que viene con la placa. Para interconexión entre la placa y el protoboard se utilizaron cables macho-macho. Se puede visualizar la conexión de los elementos en la siguiente imagen.

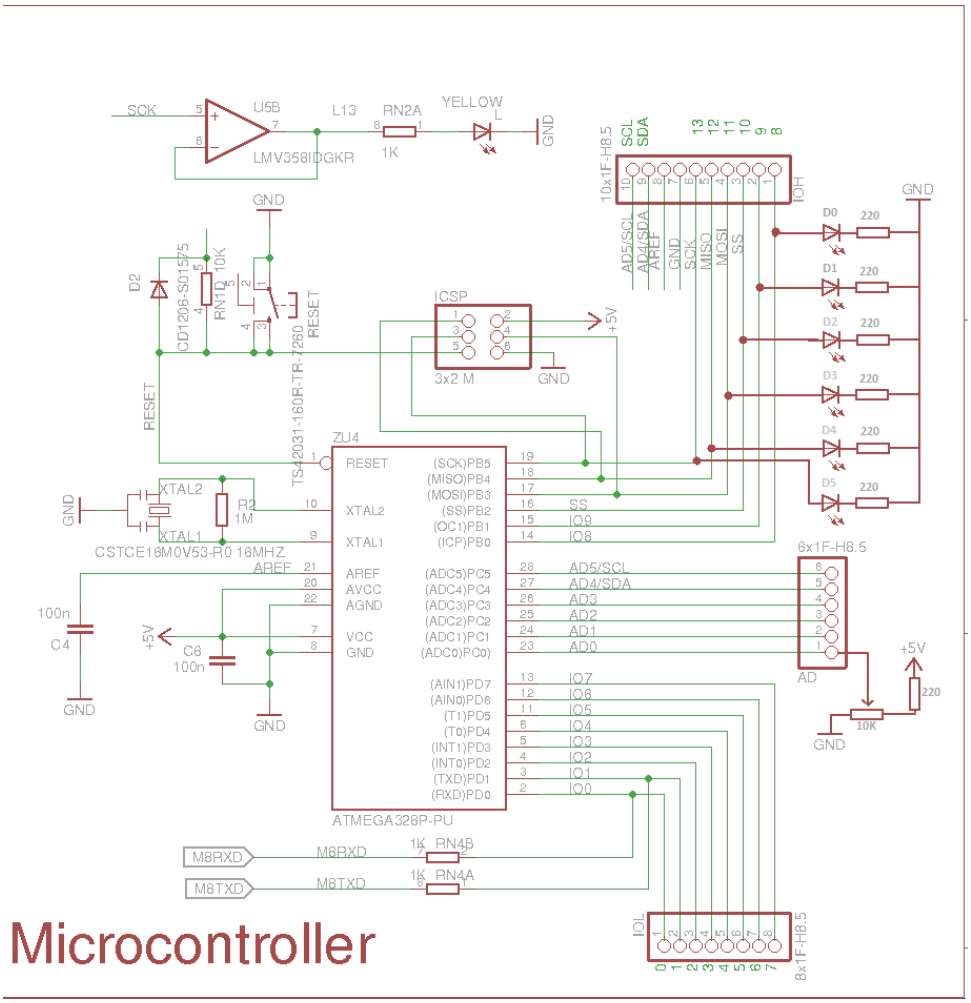

Figura 3: Diagrama de conexión.

La placa *Arduino Uno* cuenta con un microcontrolador *ATmega328p* . El ATmega328 proporciona comunicación serie UART TTL (5V), que está disponible en los pines digitales 0 (RX) y 1 (TX), Dentro de sus conexiones cuenta con 14 pines digitales de entrada/salida. Tiene una memoria flash de 32KB, una SRAM de 2KB y una EEPROM de 1KB. La velocidad del reloj es de 16MHz.

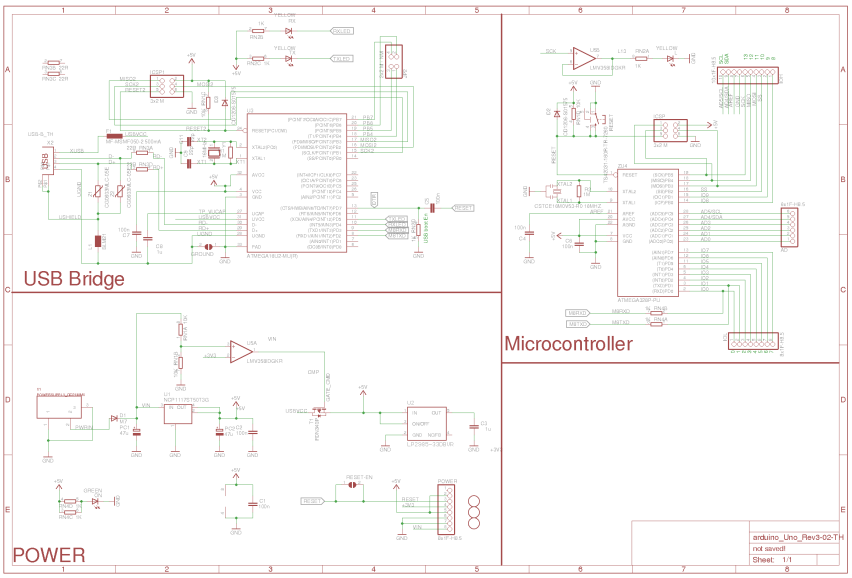

Figura 4: Diagrama de conexión de la placa.

2.3.1. Listado de componentes

Para este trabajo practico fueron utilizados los siguientes elementos. Se estima un coste total de 1800$.

Artículo	Precio
Arduino Uno	$800 - 1500$
Led	$20 - 30$
Resistencia	$5 - 10$
Potenciometro	$20 - 40$

Tabla 1: Precios de componentes.

3. Conclusiones

Para este trabajo practico se introdujo el uso del conversor analógico digital como una herramienta eficiente y fundamental ante las realización de diversas tares por parte del microcontrolador. Se configuraron utilizando los registros ADSCRA y ADMUX para poder configura el conversor interno. Fueron desarrolladas dos estrategias distintas como son el poling y las interrupciones para lograr el mismo objetivo pero con distinto funcionamiento interno. El encendido correcto de la secuencia de LED ante una variación de la señal causada por el potenciómetro pudo verificar el cumplimiento de la consigna.

4. Bibliografía

- AVR Microcontroller and Embedded Systems: Using Assembly and C, 1stEdition, Muhammad Ali Mazidi, Pearson
- "Digital Design, Principles and Practices", 3rdEdition, J.Wakerly, Prentice-Hall
- Set de instrucciones AVR de 8 bits
- Execution cycle of the AVR architecture
- Guía de Trabajos Prácticos
- ATmega328P Datasheet