



(6609) Laboratorio de Microcomputadoras

Proyecto:
(*tp5 ADC*)

Profesor:	Ing. Guillermo Campiglio
Cuatrimestre / Año:	1ro/2020
Turno de clases prácticas:	Miércoles
Jefe de Trabajos Prácticos:	Pedro Martos
Docente guía:	

Autores			Seguimiento del proyecto									
Nombre	Apellido	Padrón										
Cristian	Simonelli	87879										

Observaciones:

Fecha de aprobación

Firma J.T.P.

COLOQUIO	
Nota final	
Firma Profesor	

Objetivo:	2
Desarrollo.	2
ADC avr.	2
Registros	2
ADCSRA – ADC Control and Status Register A	2
ADMUX – ADC Multiplexer Selection Register	3
ADCL and ADCH – The ADC Data Register	3
ADCSRB – ADC Control and Status Register B	3
Listado de componentes:	4
Diagrama en bloques:	5
Circuito esquemático:	5
Diagrama de flujos:	6
Código:	7
Resultado:	8

Objetivo:

Convertir una entrada analogica en un valor digital.

Desarrollo.

Se utiliza un conversor analogico digital para leer el valor del potenciómetro, el cual se convierte a 6 bits para encender 6 leds.

ADC avr.

Adc de 10bits por aproximación sucesivas.

En este trabajo práctico se utiliza la modalidad de conversión libre, con interrupción. Esto es, el adc se convierte constantemente, cuando termina se dispara la rutina de interrupción.

Registros

ADCSRA – ADC Control and Status Register A

Bit (0x7A)	7	6	5	4	3	2	1	0	
	ADEN	ADSC	ADATE	ADIF	ADIE	ADPS2	ADPS1	ADPS0	ADCSRA
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

ADPS2	ADPS1	ADPS0	Division Factor
0	0	0	2
0	0	1	2
0	1	0	4
0	1	1	8
1	0	0	16
1	0	1	32
1	1	0	64
1	1	1	128

ADEN: Se activa la conversión.

ADPS2 , ADPS1, ADPS0 = 1 Se habilita la división por 128.

ADMUX – ADC Multiplexer Selection Register

Bit (0x7C)	7	6	5	4	3	2	1	0	
	REFS1	REFS0	ADLAR	–	MUX3	MUX2	MUX1	MUX0	ADMUX
Read/Write	R/W	R/W	R/W	R	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

ADLAR: Se ajusta a izquierda el dato

ADCL and ADCH – The ADC Data Register

23.9.3.2 ADLAR = 1

Bit	15	14	13	12	11	10	9	8	
(0x79)	ADC9	ADC8	ADC7	ADC6	ADC5	ADC4	ADC3	ADC2	ADCH
(0x78)	ADC1	ADC0	–	–	–	–	–	–	ADCL
	7	6	5	4	3	2	1	0	
Read/Write	R	R	R	R	R	R	R	R	
	R	R	R	R	R	R	R	R	
Initial Value	0	0	0	0	0	0	0	0	
	0	0	0	0	0	0	0	0	

REFS0. Referencia vcc.

Salida ADC0

MUX3..0	Single Ended Input
0000	ADC0
0001	ADC1
0010	ADC2
0011	ADC3
0100	ADC4
0101	ADC5
0110	ADC6
0111	ADC7
1000	ADC8 ⁽¹⁾
1001	(reserved)
1010	(reserved)
1011	(reserved)
1100	(reserved)
1101	(reserved)
1110	1.1V (V _{BG})
1111	0V (GND)

ADCSRB – ADC Control and Status Register B

Bit	7	6	5	4	3	2	1	0	
(0x7B)	–	ACME	–	–	–	ADTS2	ADTS1	ADTS0	ADCSRB
Read/Write	R	R/W	R	R	R	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

ADTS2	ADTS1	ADTS0	Trigger Source
0	0	0	Free running mode
0	0	1	Analog comparator
0	1	0	External interrupt request 0
0	1	1	Timer/Counter0 compare match A
1	0	0	Timer/Counter0 overflow
1	0	1	Timer/Counter1 compare match B
1	1	0	Timer/Counter1 overflow
1	1	1	Timer/Counter1 capture event

El programa es muy simple.

Cada vez que se dispara la interrupción del conversor de adc, se carga el dato convertido en el puerto B, en dicho puerto hay 6 leds conectados desde pb0 hasta pb5. Son los 6 bits de conversión.

Listado de componentes:

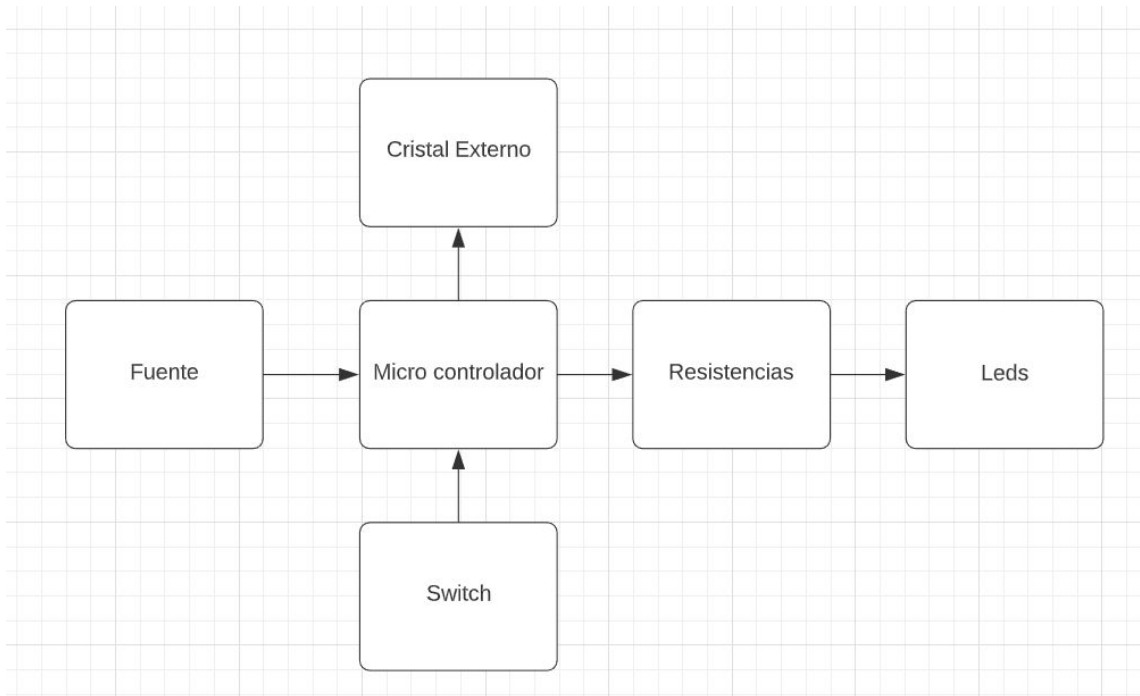
Placa arduino UNO Atmega 328p \$659 aprox 10 usd.

6 led (pack de 10) \$70.

1 potenciómetro 10k \$60

6 resistencias 220 ohm 1/8w 1% \$50.

Diagrama en bloques:



Circuito esquemático:

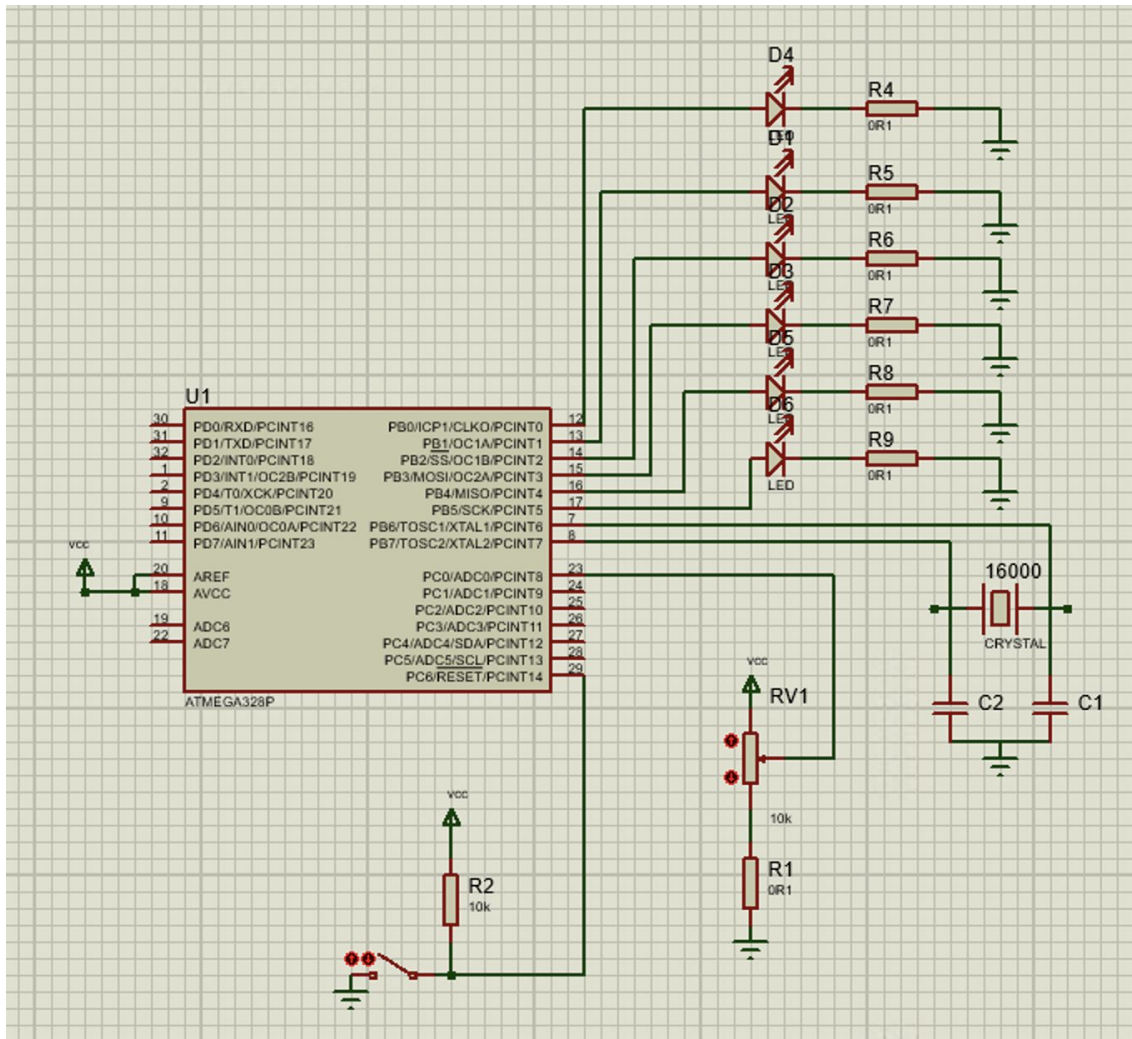
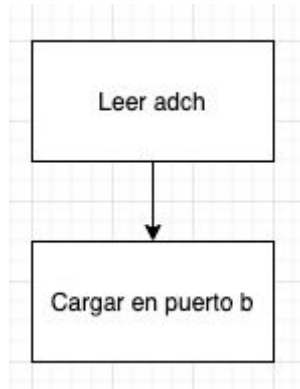


Diagrama de flujos:



Código:

```
.include "m328pdef.inc" ; Valid definitions
to 238p

.equ init_mask = 0x40 ; pin connected to
the input switch
.equ end_mask = 0x01 ; pin connected to
the input switch

.org 0x000 ; The next
instruction has to be written to add 0x0000

        rjmp      main ; Relative jump to
main

.org INT_VECTORS_SIZE ; inter vector

main:

        ldi        r20, HIGH(RAMEND) ; Load r20 with the
last ram address higher byte
        out         sph, r20 ; Load higher byte in
sp with r20
        ldi        r20, LOW(RAMEND) ; Load r20 with the
last ram address lower byte
        out         spl, r20 ; Load lower byte in
sp with r20

        ldi        r20, 0x3f ; Set port d.0 d.1
d.2 d.3 d.4 d.5 as output
        out         DDRD, r20 ;

        ldi        r20, 0xfb ; Port c.2 as input
        out         DDRC, r20 ;

        ; Set channel 2,
ADLAR (data in adch), Vcc as reference
        ldi        r20, (1 << ADLAR | 1 << MUX1 | 1 << REFS0)
```

```

        sts        ADMUX, r20
                                ; ADEN adc enable
                                ; ADSC start
conversion
                                ; adpsx = 111
division factor in 128
        ldi        r20, ( 1 << ADEN | 1 << ADSC | 1 << ADPS2 | 1
<< ADPS2 | 1 << ADPS2)
        sts        ADCSRA, r20

loop:
        call       conversion_start    ; Starts conversion
        call       conversion_wait     ; Waits the
conversion to complete
        call       adc_read            ; let value in r16
        out        PORTD, r16          ; writes this value
in portd

        jmp loop

conversion_wait:
        lds        r16, ADCSRA         ; polls ADIF until
conversion is complete
        sbrs       r16, 4              ;
        jmp        conversion_wait     ;
        ret

conversion_start:
        ldi        r16, (1 << ADSC)    ; Trigger conversion
        lds        r17, ADCSRA         ;
        or         r17, r16            ;
        sts        ADCSRA, r17         ;
        ret

adc_read:
        lds        r16, ADCH           ; reads adc value
        lsr        r16                 ; 63/255 aprox 4
        lsr        r16                 ; divide by 4
        ret

```

versión con interrupciones.

```

/
.include "m328pdef.inc"                ; Valid definitions to 238p

.org 0x000                             ; The next instruction has to be
written to add 0x0000

                rjmp     config         ; Relative jump to main

.org 0x002A
                rjmp     adc_isr

.org INT_VECTORS_SIZE                  ; inter vector

config:

                ldi      r20, HIGH(RAMEND) ; Load r20 with the last ram address
higher byte
                out      sph, r20         ; Load higher byte in sp with r20
                ldi      r20, LOW(RAMEND) ; Load r20 with the last ram address
lower byte
                out      spl, r20        ; Load lower byte in sp with r20

                ldi      r20, 0x3f
                out      PORTB, r20      ; Set the b port pin
                out      DDRB, r20       ; Set the same port pin as output and
input all others

                ldi      r22, 0xFE
                out      DDRC, r22

                ldi      r22, 0xaf
                sts      ADCSRA, r22     ; start conversion
                ldi      r22, 0x60
                sts      ADMUX, r22

                sei

main:
                jmp      main

adc_isr:
                lds      r16, ADCH
                out      PORTB, r16
                reti
```

Resultado:

Se logra convertir el valor de un potenciómetro de a la entrada del microcontrolador en un valor binario.