



Laboratorio de Microprocesadores - 86.07

Trabajo Práctico Obligatorio N°8

| | | | | | | | | | | | | |
|-------------------------------|----------|---------|--|--|--|--|--|--|--|--|--|--|
| 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 | | | | | | | | | | |
| Ivan Eric | Rubin | 100 577 | | | | | | | | | | |

Observaciones:

.....

.....

.....

.....

.....

.....

.....

.....

.....

| | | | | |
|---------------------|--|--|--|-------------|
| Fecha de aprobación | | | | Firma J.T.P |
| | | | | |

| | |
|----------------|--|
| Coloquio | |
| Nota final | |
| Firma profesor | |

1. Objetivo del trabajo

El objetivo de este trabajo será generar una conexión bidireccional entre el microprocesador y la computadora. Se deberá poder controlar un arreglo de LEDs desde la PC y también enviar un mensaje desde el ATmega328P que se muestre en el terminal serie.

2. Descripción del trabajo

Utilizando la terminal de Atmel Studio se visualizará el mensaje "*** Hola Labo de Micro *** Escriba 1, 2, 3 o 4 para controlar los LEDs". También enviando los números 1, 2, 3 o 4 se controlarán los LEDs conectados al microprocesador. Por ejemplo si se envía un 1, el LED1 cambiará de estado (Si se encuentra apagado se encenderá y viceversa).

Para esto se utilizará el Transmisor-Receptor Universal Sincrónico/Asincrónico (USART) del AT-Mega328P. Se configurará para que haya una velocidad de transmisión de 9600 bps y que haya una comunicación 8N1 (8 bits de datos, sin paridad y 1 bit de stop).

3. Diagrama de conexiones en bloques

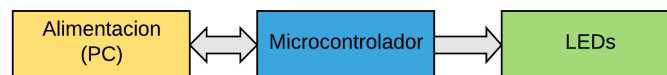


Figura 3.1: Diagrama de bloques.

4. Circuito Esquemático

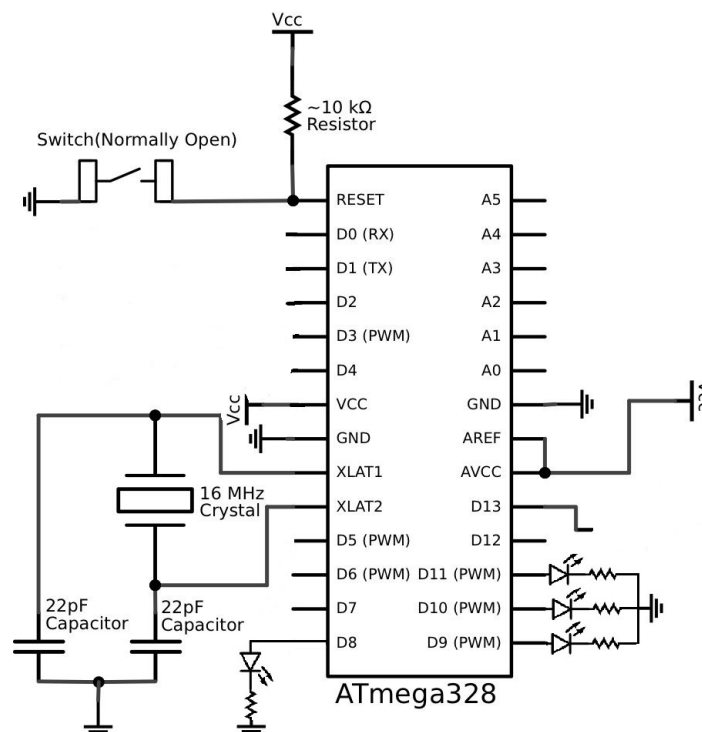


Figura 4.1: Circuito Esquemático.

5. Listado de componentes

- Arduino UNO + Conector PC (864\$).
- Microprocesador ATMEGA 328P (Integrado en Arduino).
- Resistencia de $220\Omega \pm 1\%$ (4x20\$).
- Diodo Emisor de Luz (LED) (4 x 40\$).
- Cables Macho-Macho (40 cables x 100\$).
- Protoboard (200\$).

6. Diagrama de flujo del software

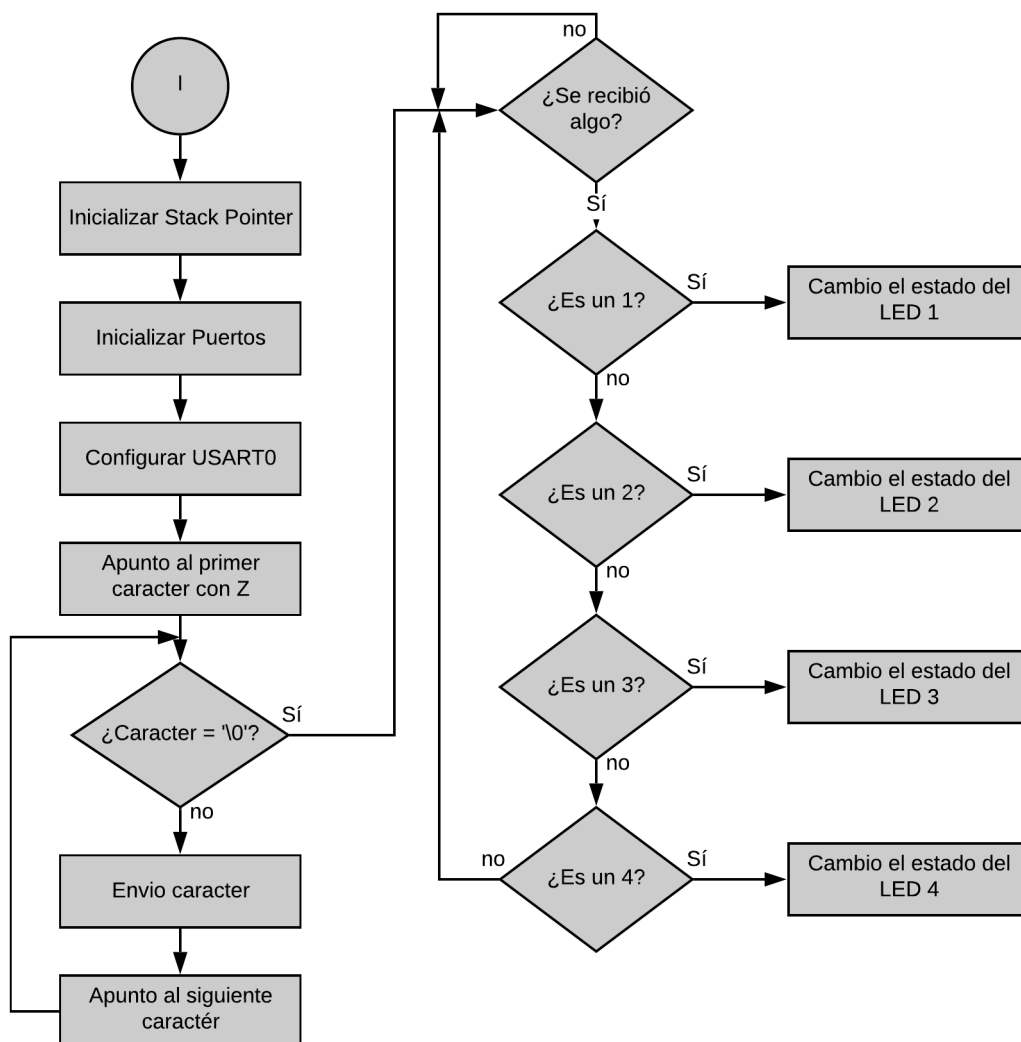


Figura 6.1: Diagrama de flujo.

7. Código de programa

```

1  /*
2  *
3  *   Trabajo Práctico Obligatorio 8
4  *   Autor: Ivan Eric Rubin
5  *
6  */
7  .include "m328pdef.inc"
8
9  .def dummyr=R16
10 .def char=R17
11 .def LED_number=R18
12 .def LED_pos=R19
13
14 .equ MSG_END='\0'
15
16 .cseg
17
18 .org 0x0000
19     jmp     configuracion
20
21 .org INT_VECTORS_SIZE
22
23 configuracion:
24     ; Inicializo el Stack Pointer.
25     ldi     dummyr, HIGH(RAMEND)
26     out     SPH, dummyr
27     ldi     dummyr, LOW(RAMEND)
28     out     SPL, dummyr
29
30     ; Puerto B como salida.
31     ldi     dummyr, 0xFF
32     out     DDRB, dummyr
33
34     ; Puerto D como salida salvo el pin 0.
35     ldi     dummyr, 0xFE
36     out     DDRD, dummyr
37
38     ; Configuro el BAUD RATE en 9600 bps.
39     ldi     dummyr, 0x00
40     sts     UBRROH, dummyr
41     ldi     dummyr, 103
42     sts     UBRROL, dummyr
43
44     ; Habilito recepcion y emisión de datos.
45     ldi     dummyr, (1<<RXEN0) | (1<<TXEN0)
46     sts     UCSROB, dummyr
47
48     ; 8 bits de datos, sin paridad, 1 bit de stop (8N1).
49     ldi     dummyr, (1<<UCSZ01) | (1<<UCSZ00)
50     sts     UCSROC, dummyr
51
52 main:
53     ; Envio el mensaje
54     ; Cargo el mensaje en el registro Z.
55     ldi     ZH, HIGH(INIT_MSG <<1)
56     ldi     ZL, LOW(INIT_MSG <<1)
57
58     ; Envío caracter por caracter.
59     set_char:

```

```

60     lpm      char, Z+
61     cpi      char, MSG_END
62     breq     msg_ended           ; Si el caracter es '\0', terminó.
63
64     ; Mando el caracter por el puerto serie.
65     send_puerto_serie:
66     lds      dummyr, UCSROA
67     sbrs     dummyr, UDRE0       ; Si UDRE0=1 mando el siguiente caracter.
68     rjmp     send_puerto_serie
69
70     sts      UDR0, char          ; Envío.
71     rjmp     set_char           ; Cargo siguiente caracter.
72
73     msg_ended:
74     rjmp     get_data
75
76     ; Leo datos y controlo LEDs.
77     get_data:
78     lds      dummyr, UCSROA
79     sbrs     dummyr, RXC0        ; Si se recibio algo lo proceso.
80     rjmp     get_data
81
82     ; Guardo el numero de LED recibido y lo guardo.
83     lds      LED_number, UDR0
84
85     andi     LED_number, 0x0F
86
87     ; Si el dato es 1, 2, 3 o 4 enciendo el led correspondiente
88     cpi      LED_number, 1
89     breq     LED1
90
91     cpi      LED_number, 2
92     breq     LED2
93
94     cpi      LED_number, 3
95     breq     LED3
96
97     cpi      LED_number, 4
98     breq     LED4
99
100    rjmp     get_data
101
102    LED1:
103    ldi      LED_pos, (1<<PORTB0)
104    rjmp     on_off
105
106    LED2:
107    ldi      LED_pos, (1<<PORTB1)
108    rjmp     on_off
109
110    LED3:
111    ldi      LED_pos, (1<<PORTB2)
112    rjmp     on_off
113
114    LED4:
115    ldi      LED_pos, (1<<PORTB3)
116    rjmp     on_off
117
118    ; Enciendo o apago el LED correspondiente
119    on_off:
120    in       dummyr, PORTB

```

```
121     eor        dummyr, LED_pos
122     out        PORTB, dummyr
123     rjmp       get_data
124
125 INIT_MSG:
126     .db "*** Hola Labo de Micro ***", '\n', '\n', "Escriba 1, 2, 3 o 4 para controlar
    los LEDs", '\0'
```

8. Resultados

Se logró lo pedido sin mayores dificultades. El control de los LEDs funcionó correctamente con el código implementado

9. Conclusiones

Se destaca de este trabajo la utilidad de la comunicación serial entre el microprocesador y la computadora. Puede ser de utilidad para poder visualizar texto y ver si las cosas están funcionando correctamente. Cabe mencionar que se podría haber configurado el USART0 de muchas maneras que serían de gran utilidad en otros casos.