



Laboratorio de Microprocesadores - 86.07

Trabajo Práctico Obligatorio N°1

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	
Mariano Federico	Guglieri 99573
Seguimiento del proyecto	

Observaciones:

.....

.....

.....

.....

.....

.....

.....

.....

Fecha de aprobación		Firma J.T.P

Coloquio	
Nota final	
Firma profesor	

Índice

1. Objetivos	2
2. Desarrollo	2
2.1. Configuración del USART0	2
3. Diagrama en bloques	2
4. Esquemático y listado de componentes	3
5. Diagrama de flujos	3
6. Códigos	5
7. Resultados	6
8. Conclusiones	6

1. Objetivos

El objetivo de este trabajo consiste en establecer una comunicación bidireccional serie entre microcontrolador y una computadora. Se implementará un programa que envíe un texto desde el atmega328p a la terminal del Atmel Studio. Además, se deberá poder encender y apagar 4 LEDs que se encuentran conectados a los pines del atmega328p mediante el ingreso de datos en la terminal de la computadora.

2. Desarrollo

Para la realización del trabajo se utilizó la plataforma de desarrollo de Atmel, Atmel Studio versión 7.0, en donde se implementó el Software. Para la parte física se requirió de un Arduino UNO, el cual sirvió como programador para el integrado ATMEGA328P, además de suplirlo con la energía necesaria para su funcionamiento. Se utilizaron cuatro resistores junto a cuatro leds de color verde conectados al puerto b y cinco cables para unir el circuito.

La primera parte del código está dedicada a la configuración básica de los puertos, la inicialización del Stack Pointer y de los registros del USART. El programa principal consta de dos partes. La primera envía el mensaje que se quiere enviar y la otra se encarga de recibir el mensaje que se ingrese por el teclado. Se pudo haber hecho dos call para seccionar el código pero se decidió no hacerlo ya que no era muy largo.

2.1. Configuración del USART0

La inicialización del USART consta de configurar el baud rate de transmisión, el formato de sincronización y la habilitación de la transmisión y recepción.

Como se transmitió a velocidad simple y el cristal del Arduino es de 16MHz, se buscó en las tablas correspondientes qué valor se debía cargar en el registro UBRR0 para que trabaje con un Baud Rate de 9600.

Luego, sólo se requirió habilitar el puerto de recepción y transmisión, ya que no se utilizaron interrupciones asociadas al USART0. Se implemento Polling.

El formato de bit se definió para que trabaje con datos de 8bits, con un solo bit de parada y sin bit de paridad. Toda esta configuración se detalla en el archivo TP8.asm.

Cabe aclarar, que el PIND0 se declaró como entrada y los demás pines del PORTD como salida. Esto se debe a que este pin es el pin de Rx.

3. Diagrama en bloques

El diagrama de conexiones en bloque del trabajo se presenta a continuación:

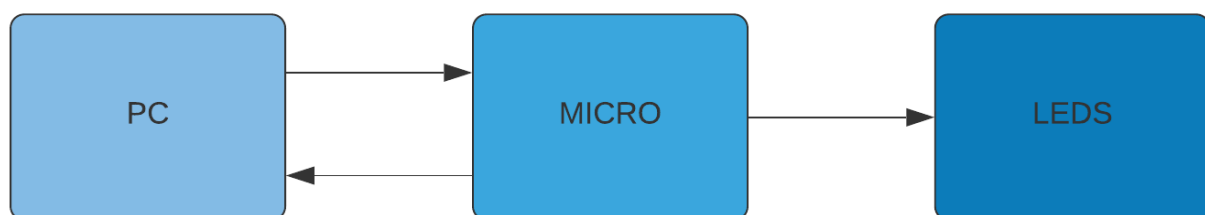


Figura 1: Diagrama de conexiones en bloque

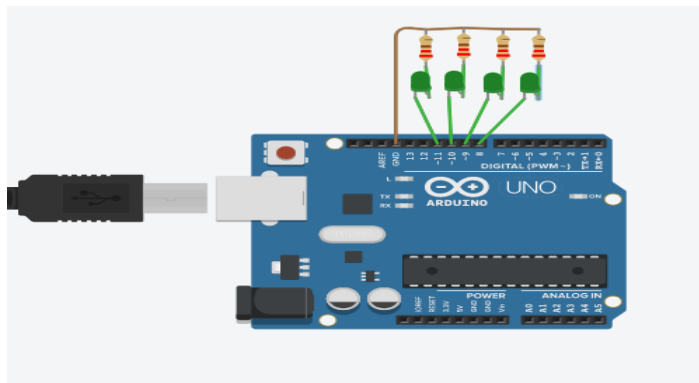
El programa se descarga de la PC al micro. Una vez que se conecta el atmega328p a la computadora, cuando se abra la terminal, éste enviará un saludo junto a la instrucción: *"Escriba 1,2,3 0 4 para controlar los LEDs"*. Según la directiva que se reciba desde el teclado, se encenderá o se apagará el led 1,2,3 o 4.

4. Esquemático y listado de componentes

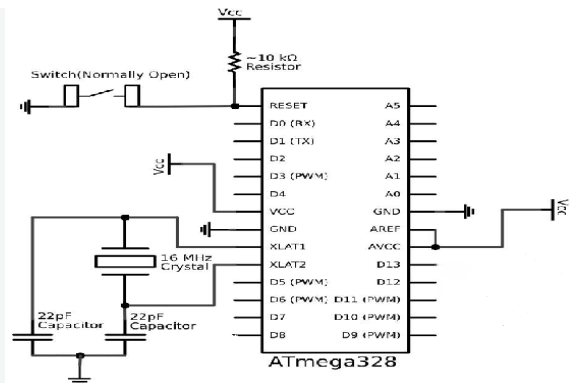
Para el circuito se utilizaron los siguientes componentes:

- 4 resistores de 220Ω
- 4 leds de color verde
- 5 cables para protoboard
- 1 placa arduino con atmega328p
- 1 protoboard

La plataforma arduino le agrega al microcontrolador un cristal de 16Mhz y una resistencia de $10k\Omega$, conectados como en la figura 2.b.



(a) Esquemático del circuito implementado



(b) Circuito con cristal de 16MHz

Figura 2: Esquemático del circuito

En total se habrá gastado unos 50 pesos para el circuito. Sin contar la plataforma arduino.

5. Diagrama de flujos

A continuación se presenta el diagrama de flujo. El primero consiste en el programa principal más la configuración previa. La configuración de puertos y del USART se puso todo en un mismo bloque ya que lo primero se vio en los trabajos anteriores y la configuración del USART se explicó más arriba. La sección del programa principal es prácticamente idéntica a la implementación del código del trabajo práctico 3, el cual manda una cadena en la memoria FLASH al PORTB. La diferencia radica en que esta cadena es enviada a la terminal de la computadora y se manda sólo una vez. La segunda parte del main está constantemente verificando si se ha ingresado un dato desde la terminal (Polling). Se verifica además, si el dato ingresado es válido. De no serlo, se vuelve a preguntar si se ha ingresado un dato. Si el dato es válido, es enviado al PORTB, con la salvedad, que si el dato es un 3 o un 4, por cómo se hizo la implementación, éste deberá ser modificado antes de enviarlo al PORTB, para que se pueda encender el LED correspondiente.

Cabe aclarar que en los trabajos prácticos anteriores se intentó que la lógica sea, en lo posible, mantenible y adaptable. En este caso se priorizó que el código sea compacto. Es decir, poder agregar un quinto led que cumpla lo especificado no será tan trivial.

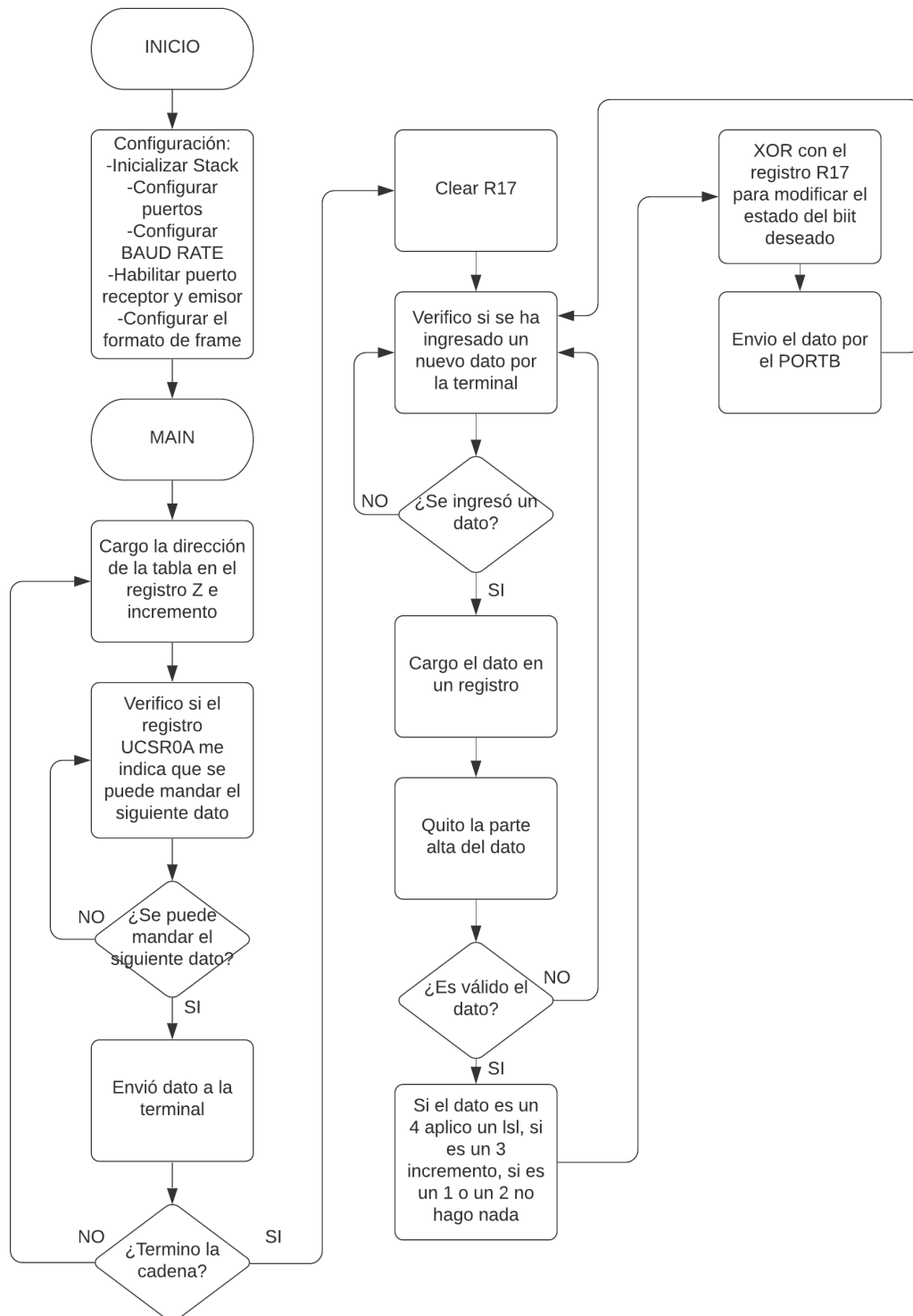


Figura 3: Diagrama de flujos

6. Códigos

A continuación se presenta el código:

```

1 .include "m328pdef.inc"
2
3 .cseg
4 .org 0x0000
5     jmp     configuracion
6 .org INT_VECTORS_SIZE
7
8 configuracion:
9     ;inicializo el stack
10    ldi     R16, HIGH(RAMEND)
11    out     SPH, R16
12    ldi     R16, LOW(RAMEND)
13    out     SPL, R16
14
15    ;declaro el PORTB como salida
16    ldi     R16, 0xff
17    out     DDRB, R16
18
19    ;declaro los pines del PORTD
20    ldi     R16, 0xfe
21    out     DDRD, R16
22
23    ;configuro BAUD RATE
24    ldi     R16, 0x00
25    sts     UBRR0H, R16
26    ldi     R16, 103
27    sts     UBRR0L, R16
28
29    ;Habilito puerto receptor y emisor
30    ldi     R16, (1<<RXEN0) | (1<<TXEN0)
31    sts     UCSROB, R16
32
33    ;Frame format: 8bit data, 1 stop bit, no parity
34    ldi     R16, (1<<UCSZ01) | (1<<UCSZ00)
35    sts     UCSROC, R16
36
37    sei
38
39 main:
40    ldi     ZH, HIGH(sequencia1 <<1)    ;En esta seccion mando la cadena que se leer
41    en la terminal.
42    ldi     ZL, LOW (sequencia1 <<1)    ;El cdigo implementado es prcticamente al
43    realizado en el TP3
44    ldi     R17, 68
45
46 siguiente:    lpm     R16, Z+
47 USART_Transmit:    lds     R18, UCSROA    ;Me fijo cuando el bit UDRE0 del registro
48 UCSROA se pone en 1
49 sbrs     R18, UDRE0    ;Si se pone en 1, est listo para mandar el
50 siguiente dato
51 rjmp     USART_Transmit
52 sts     UDRO, R16
53 dec     R17
54 brne     siguiente
55 ;-----*****-----
56 clr     R17
57 USART_Receive:
58 lds     R18, UCSROA    ;Loop que espera a recibir data
59 sbrs     R18, RXC0    ;cuando se recibe data, el bit RXC0 se activa
60 saliendo del loop
61 rjmp     USART_Receive
62 lds     R16, UDRO
63 andi     R16, 0x0f    ;Mascara para descartar los bits que no me sirven
64 cpi     R16, 1    ;Todas las entradas que no sean 1,2,3 o 4 no
65 deberian
66 brlo     USART_Receive    ;hacer nada
67 cpi     R16, 5
68 brcc     USART_Receive
69 sbrs     R16, 2    ;Si la entrada es un 4
70 lsl     R16

```

```

68
69         cpi        R16,3                ;Si la entrada es un 3
70         brne      no_es_un_3
71         inc        R16
72
73         no_es_un_3:  eor        R17, R16    ;xor entre el registro r17 y r16 para cambiar de
74         estado bajo a alto
75         out        PORTB, R17            ;o de alto o bajo. Saco por el puerto B el estado
76
77         rjmp       USART_Receive
78
79 secuencial:
80 .DB  '*', '*', '*', ' ', 'H', 'o', 'l', 'a', ' ', 'L', 'a', 'b', 'o', ' ', 'd', 'e', ' ', 'M', 'i', 'c', 'r', 'o', ' ', '*', '*'
81     , '*', '\n', 'E', 's', 'c', 'r', 'i', 'b', 'a', ' ', '1', ' ', '2', ' ', '3', ' ', 'o', ' ', '4', ' ', 'p', 'a', 'r', 'a', ' ',
82     , 'c', 'o', 'n', 't', 'r', 'o', 'l', 'a', 'r', ' ', 'l', 'o', 's', ' ', 'L', 'E', 'D', 's'

```

7. Resultados

Se observó lo que se esperaba sin mayor inconveniente. El enunciado aconsejaba agregar un delay en el inicio pero no hizo falta.

8. Conclusiones

En este trabajo se pudo estudiar la característica del atmega328p que es la comunicación bidireccional serie. Si bien sólo se configuró el USART de una manera específica, se observaron los diferentes modos de trabajo que poseen los registros de recepción y transmisión. Como la posibilidad de trabajar de manera sincrónica o asincrónica, doble o simple presición, enviar bits de paridad o no, etc. Además, se hace mención a la importancia de setear correctamente el baud rate ya que, de no tener el mismo baud rate que la terminal, los mensajes recibidos y transmitidos no podrán leerse correctamente.