Facultad de Ingeniería
Universidad de Buenos Aires

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			Seguimiento del proyecto									
Mariano Federico	Guglieri	99573										

Observaciones:

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

Fecha de aprobación				Firma J.T.P

Coloquio	
Nota final	
Firma profesor	

Índice

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.

3. Diagrama en bloques

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

Figura 1: Diagrama de conexiones en bloque

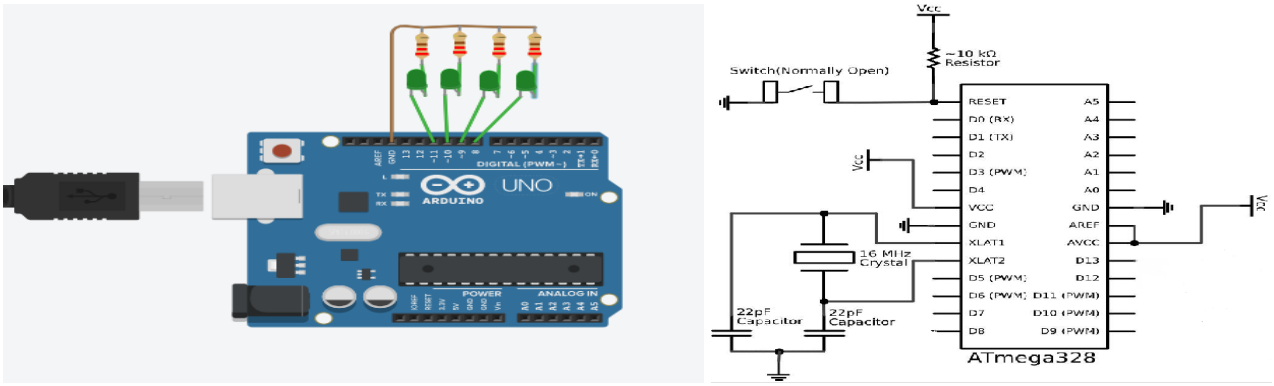
El programa se descarga de la PC al micro. Una vez que se conecta el atmega328p a la computadora, cuando se abra la terminal, éste enviará un saludo junto a la instrucción: *"Escriba 1,2,3 0 4 para controlar los LEDs"*. Según la directiva que se reciba desde el teclado, se encenderá o se apagará el led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

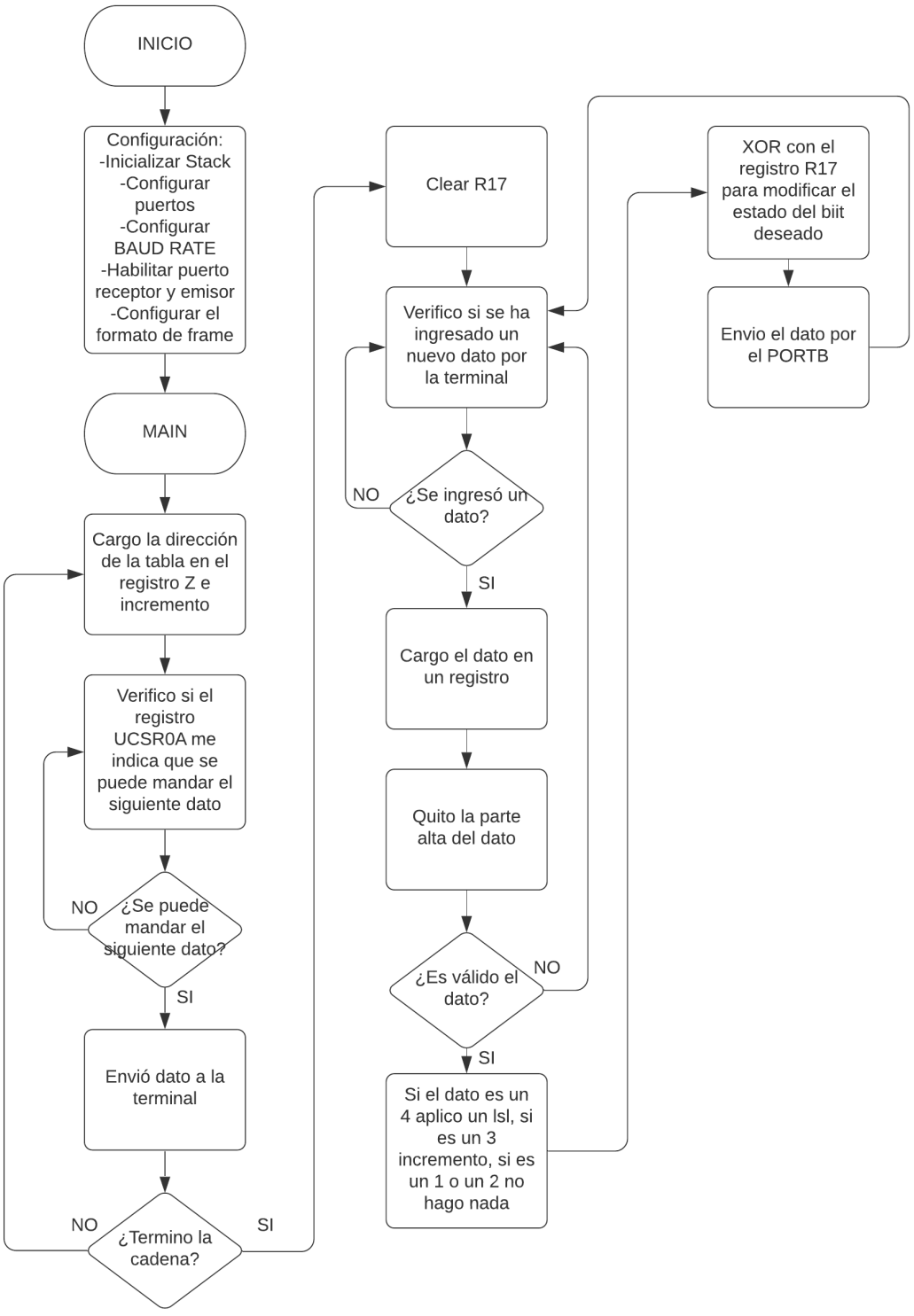


Figura 3: Diagrama de flujos

6. Códigos

A continuación se presenta el código:

```
.include "m328pdef.inc"

.cseg
.org 0x0000
   jmp      configuracion
.org INT_VECTORS_SIZE

configuracion:
   ;inicializo el stack
   ldi      R16, HIGH(RAMEND)
   out      SPH, R16
   ldi      R16, LOW(RAMEND)
   out      SPL, R16

   ;declaro el PORTB como salida
   ldi      R16,0xff
   out      DDRB,R16

   ;declaro los pines del PORTD
   ldi      R16, 0xfe
   out      DDRD, R16

   ;configuro BAUD RATE
   ldi      R16, 0x00
   sts      UBRR0H, R16
   ldi      R16, 103
   sts      UBRR0L, R16

   ;Habilito puerto receptor y emisor
   ldi      R16, (1<<RXEN0) | (1<<TXEN0)
   sts      UCSR0B, R16

   ;Frame format: 8bit data, 1 stop bit, no parity
   ldi      R16, (1<<UCSZ01) | (1<<UCSZ00)
   sts      UCSR0C, R16

   sei

main:
                ldi      ZH, HIGH(secuencia1 <<1)      ;En esta seccin mando la cadena que se leer en la terminal.
                ldi      ZL, LOW (secuencia1 <<1)      ;El  c digo  implementado es  p r cticamente  al realizado en el TP3
                ldi      R17, 68

   siguiente:        lpm      R16, Z+
   USART_Transmit:      lds      R18, UCSR0A               ;Me fijo cuando el bit UDRE0 del registro UCSR0A se pone en 1
                sbrs  R18, UDRE0                ;Si se pone en 1, est  listo para mandar el siguiente dato
                rjmp  USART_Transmit
                sts      UDR0, R16
                dec      R17
                brne  siguiente
   ;----------------*****************************************------------------------------
                clr      R17
   USART_Receive:
                lds      R18, UCSR0A               ;Loop que espera a recibir data
                sbrs  R18, RXC0                 ;cuando se recibe data, el bit RXC0 se activa saliendo del loop
                rjmp  USART_Receive
                lds      R16, UDR0

                andi  R16, 0x0f                 ;Mascara para descartar los bits que no me sirven

                cpi      R16, 1                    ;Todas las entradas que no sean 1,2,3 o 4 no deberian
                brlo  USART_Receive             ;hacer nada
                cpi      R16, 5
                brcc  USART_Receive

                sbrc  R16,2                   ;Si la entrada es un 4
                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
    estado bajo a alto
74                  out     PORTB, R17                ;o de alto o bajo. Saco por el puerto B el estado
75
76   rjmp  USART_Receive
77
78 secuencia1:
79 .DB '*','*','*',' ','H','o','l','a',' ','L','a','b','o',' ','d','e',' ','M','i','c','r','o',' ','*','*'
    ,'*','\n','E','s','c','r','i','b','a',' ','1',',','2',',','3',' ','o',' ','4',' ','p','a','r','a','
     ','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.