Fiuba
Facultad de Ingeniería
Universidad de Buenos Aires

Laboratorio de Microprocesadores - 86.07

Trabajo Práctico Nº8:

Puerto serie

Profesor:			Ing. Guillermo Campiglio									
Cuatrimestre/Año:			1º/2020									
Turno de las clases prácticas			Miércoles									
Jefe de trabajos prácticos:			Ing. Pedro Ignacio Martos									
Docentes guía:			Ing. Fabricio Baglivo, Ing. Fernando Pucci									
Fecha de presentación:			19/08/2020									
Alumno			Seguimiento del proyecto									
Nombre	Apellido	Padrón										
Maximiliano Daniel	Reigada	100565										

Observaciones:

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

Fecha de aprobación			Firma J.T.P

Índice

1. Objetivo

El presente trabajo práctico tiene como objetivo progresar con el manejo de registros del USART de los que dispone el microcontrolador ATmega328P, a fin de establecer una comunicación bidireccional serie entre éste y una computadora de escritorio.

2. Descripción

Mediante la implementación de una placa de microcontrolador de código abierto basado en el microchip ATmega328P, denominada Arduino Uno, se busca desarrollar un programa cuya función consta esencialmente en dos partes:

1. Al encender microcontrolador, el programa transmite el texto

   *** Hola Labo de Micro ***

   Escriba 1, 2, 3 o 4 para controlar los LEDs

   y se muestra en el terminal serie.

2. Si en el terminal serie se aprieta la tecla n (con $n = 1,\ 2,\ 3$ o 4), entonces se enciende/apaga el LED n (función de toggle).

Para este caso se usa el USART o Transmisor-Receptor Universal Sincrónico/Asíncrono del que dispone el microcontrolador. Como se quiere que la comunicación sea asincrónica, sin bit de paridad, con un bit de paro y con un tamaño de datos de 8 bits, se carga en el registro `UCSR0C` el valor correspondiente para que esto suceda. Por otra parte, para establecer la velocidad en 9600 *bps*, se cargan los registros `UBRR0H/L` con los valores indicados en la tabla dispuesta en la hoja de datos del ATmega328P para un oscilador de $16MHz$.

Luego de enviar carácter por carácter del mensaje inicial, se procede a leer los datos que se reciben por el puerto serie, y en caso de que alguno sea igual a 1, 2, 3 o 4, se alterna el estado lógico en que se encuentre el pin del puerto B conectado al LED correspondiente al número seleccionado.

3. Diagrama de conexiones en bloques

A continuación, puede verse el diagrama de conexiones en bloques correspondiente a este trabajo práctico:

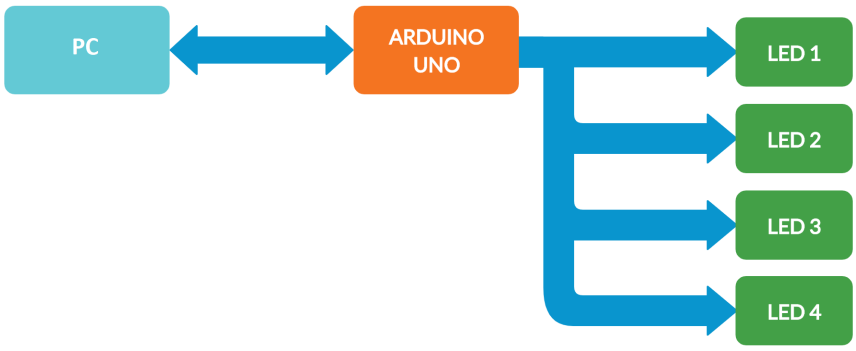

Figura 1: Diagrama de conexiones en bloques.

4. Circuito esquemático

Se procede a realizar el esquemático del circuito a implementar. Cabe destacar que, si bien en la realidad los componentes se encuentran conectado al Arduino Uno y este a su vez a una fuente de tensión externa, resulta más útil representar en el esquema solo los componentes electrónicos esenciales del sistema, conectados a referencias de tensión de un determinado valor:

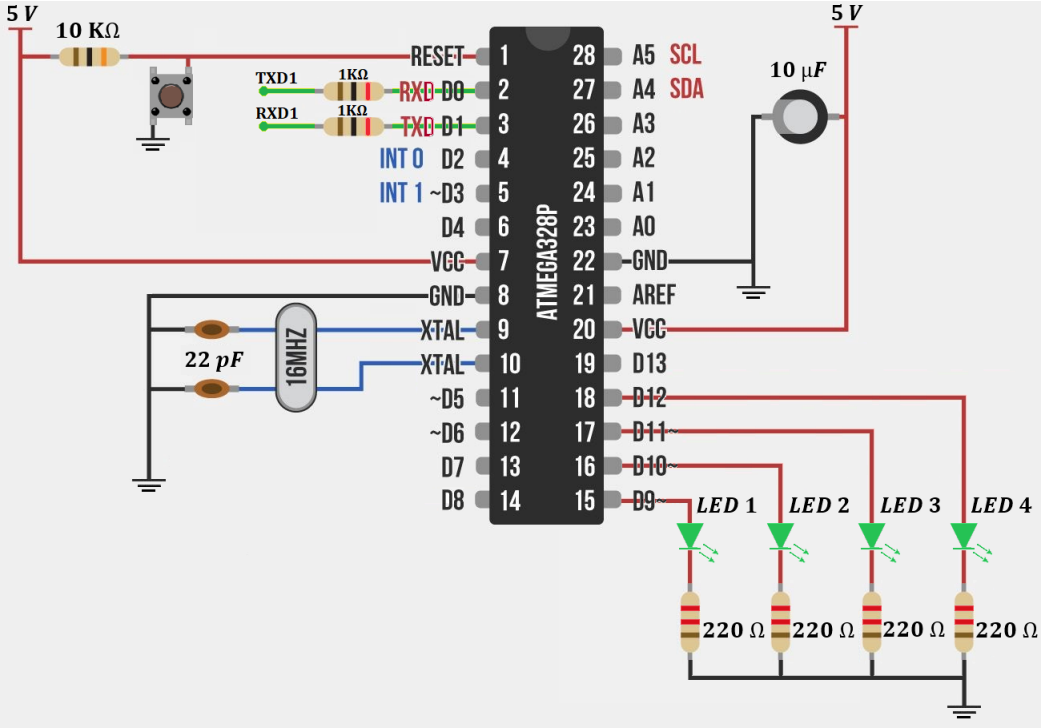

Figura 2: Circuito esquemático.

Es necesario también aclarar para este caso, que en el Arduino Uno los pines TXD y RXD de los que dispone el Atmega328p se encuentran conectados por medio de un resistor de $1K\Omega$ con los pines RXD1 y TXD1 pertenecientes al Atmega16u2, el cual se encarga de establecer la comunicación vía USB con la computadora.

5. Listado de componentes

En el desarrollo de este trabajo práctico se utilizan los siguientes componentes:

- 4 LEDs verdes de 5mm
- 4 Resistores de 220 Ω
- 5 Cables de conexión
- 1 Protoboard
- 1 Arduino Uno (con microcontrolador ATmega328P incluido)

6. Diagrama de flujo del Software

A continuación, se explicitan los diagramas de flujo correspondientes al código desarrollado:

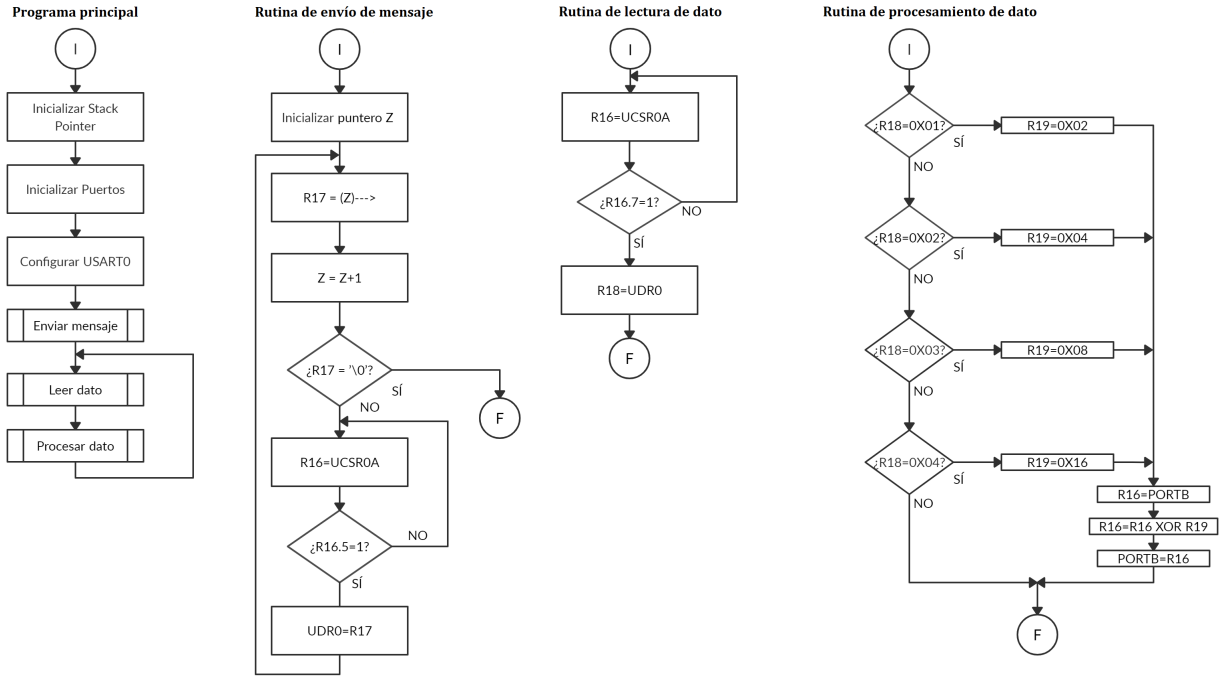

Figura 3: Diagramas de flujo.

7. Código de programa

A continuación, puede verse el código desarrollado para este proyecto:

```
;***************************************************************************
; Código correspondiente al ejercicio planteado en el Trabajo Práctico 8.
;
; Alumno: Reigada Maximiliano Daniel
; Padrón: 100565
;***************************************************************************

.INCLUDE "m328pdef.inc"

.DEF AUX=R16
.DEF CARACTER=R17
.DEF DATO_IN=R18
.DEF MASK_PIN=R19

.CSEG
.ORG 0X0000
      RJMP  config

.ORG INT_VECTORS_SIZE

config:
      LDI    AUX, HIGH(RAMEND)              ;Inicializo el SP al final de la RAM.
      OUT    SPH, AUX
      LDI    AUX, LOW(RAMEND)
      OUT    SPL, AUX

      LDI    AUX, 0XFE                      ;Declaro PORTD0 como entrada y el resto
      OUT    DDRD, AUX                      ;como salidas.

      LDI    AUX, 0XFF                      ;Declaro al puerto B como salida.
      OUT    DDRB, AUX

      LDI    AUX, 0X00                      ;Configuro el Baud Rate del USART0 en 9600 bps,
      STS    UBRR0H, AUX                    ;cargando en UBRR0 el valor 103 por tabla
      LDI    AUX, 0X67                      ;de datasheet.
      STS    UBRR0L, AUX

      LDI    AUX, (1<<UCSZ01) | (1<<UCSZ00) ;Configuro el tamaño de los datos en 8N1.
      STS    UCSR0C, AUX

      LDI    AUX, (1<<RXEN0) | (1<<TXEN0)   ;Habilito la recepción y transmisión de datos
      STS    UCSR0B, AUX                    ;por puerto serie.

main:
      RCALL  enviar_mensaje                 ;Envío el mensaje inicial por el puerto serie.

leer_dato:
      LDS    AUX, UCSR0A                    ;Entro en loop hasta que se reciba un dato.
      SBRS   AUX, RXC0
      RJMP   leer_dato

      LDS    DATO_IN, UDR0                  ;Cuando se recibe un dato, lo cargo en DATO_IN.
      RCALL  procesar_dato                  ;Proceso el dato recibido.

      RJMP   leer_dato                      ;Salto a leer el próximo dato.

.ORG 0X500
MENSAJE_PROG: .DB "*** Hola Labo de Micro ***", '\n','\n',"Escriba 1, 2, 3 o 4 para controlar
      los LEDs", '\0'

enviar_mensaje:
      LDI    ZL, LOW(MENSAJE_PROG<<1)       ;Apunto al primer elemento de la tabla en
      LDI    ZH, HIGH(MENSAJE_PROG<<1)      ;memoria del programa con el puntero Z.

cargar_caracter:
      LPM    CARACTER, Z+
      CPI    CARACTER, '\0'                 ;En caso de que el carácter sea 0, salto a
      BREQ   fin_envio                      ;finalizar la transmisión del mensaje.
```

```
68
69  enviar_caracter:
70        LDS     AUX, UCSR0A
71        SBRS    AUX, UDRE0                  ;Entro en loop hasta que UDR0 este vacío
72        RJMP    enviar_caracter             ;y se pueda enviar el próximo carácter.
73
74        STS     UDR0, CARACTER              ;Envío el carácter por el puerto serie.
75        RJMP    cargar_caracter
76
77  fin_envio:
78        RET
79
80
81  procesar_dato:
82        CPI     DATO_IN, 0X01
83        BREQ    seleccionar_LED1
84
85        CPI     DATO_IN, 0X02
86        BREQ    seleccionar_LED2
87
88        CPI     DATO_IN, 0X03
89        BREQ    seleccionar_LED3
90
91        CPI     DATO_IN, 0X04
92        BREQ    seleccionar_LED4
93
94  fin_procesamiento:
95        RET
96
97  seleccionar_LED1:
98        LDI     MASK_PIN, (1<<PORTB1)       ;Como DATO_IN=1 ----> MASK_PIN=0b00000010.
99        RJMP    alternar_LED
100
101 seleccionar_LED2:
102       LDI     MASK_PIN, (1<<PORTB2)       ;Como DATO_IN=2 ----> MASK_PIN=0b00000100.
103       RJMP    alternar_LED
104
105 seleccionar_LED3:
106       LDI     MASK_PIN, (1<<PORTB3)       ;Como DATO_IN=3 ----> MASK_PIN=0b00001000.
107       RJMP    alternar_LED
108
109 seleccionar_LED4:
110       LDI     MASK_PIN, (1<<PORTB4)       ;Como DATO_IN=4 ----> MASK_PIN=0b00010000.
111       RJMP    alternar_LED
112
113 alternar_LED:
114       IN      AUX, PORTB
115       EOR     AUX, MASK_PIN               ;Aplico una XOR entre MASK_PIN y el valor en PORTB
116       OUT     PORTB, AUX                  ;para solo alternar el valor del pin seleccionado.
117       RJMP    fin_procesamiento
```

8. Resultados

Luego de desarrollar el código descripto en las secciones anteriores y armar el circuito especificado, se logra emitir el mensaje inicial desde el microcontrolador hacia la computadora por medio del puerto serie, para poder visualizarlo a través del terminal del Atmel Studio como puede apreciarse en la siguiente captura:

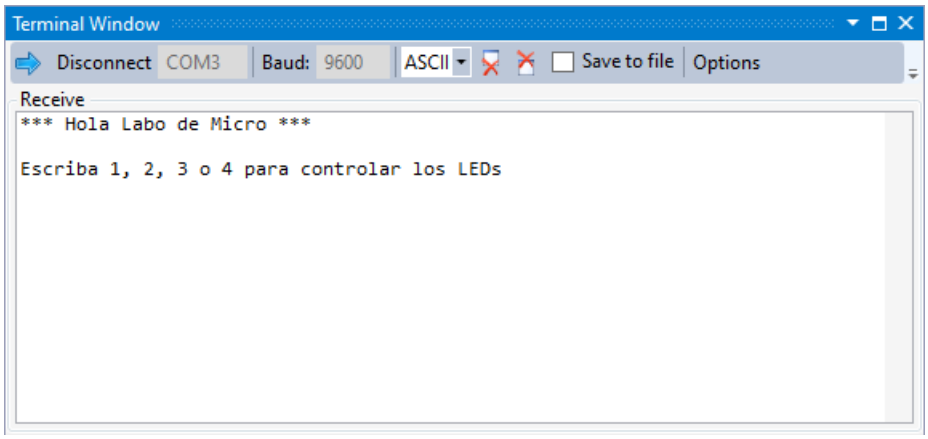

Figura 4: Captura del mensaje inicial en terminal.

Luego de este mensaje, se corrobora que escribiendo y enviando desde la terminal cualquiera de los números presentes en éste, se enciende/apaga el LED que le corresponde a cada uno. En caso de que se envíe algún valor que no sea compatible, el programa simplemente lo ignora.

Finalmente, se procede a realizar un listado de los componentes utilizados y sus costos:

Componentes	Costos
Leds verde de 5mm	$28,00
Resistor de 220 Ω	$22,00
Cables de conexión	$10,00
Protoboard	$232,00
Arduino Uno	$720,00
TOTAL	$1012,00

9. Conclusiones

Tras haber realizado todos los pasos pedidos en el enunciado de este trabajo práctico, resta destacar las conclusiones que la experiencia ha aportado.

A partir de este trabajo se logra afianzar el manejo de un protocolo para comunicaciones duales como lo es es USART, estableciendo en este caso un modo asincrónico 8N1 a una velocidad determinada. Son estas últimas características las que dan la posibilidad a futuro de implementar distintos modos según el caso en que se busque aplicar esta herramienta, debido al previo análisis de configuración sobre los registros con los que se cuenta.

Por otro lado, en esta aplicación no se utilizaron ninguna de las interrupciones de las que dispone el USART0, de modo que para la recepción de datos se revisa constantemente si el microcontrolador recibió alguno hasta que esto suceda, por medio de un bucle presente en el programa principal. En caso de que se quiera implementar la interrupción asociada a la recepción de datos, este último bucle no sería necesario, e inclusive se podría disminuir el consumo del dispositivo poniéndolo en modo sleep mientras no se reciba un dato o se lo esté procesando.