Departamento de Electrónica
Laboratorio de Microprocesadores - 86.07

Trabajo Práctico Obligatorio 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								
Docente guía:			Ing. Fabricio Baglivo, Ing. Fernando Pucci								
Autor			Seguimiento del proyecto								
Maximiliano	Porta	98800									

Observaciones:

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

Fecha de aprobación				Firma J.T.P

20 de agosto de 2020

Índice

1. Objetivo

Consiste en aprender la comunicación bidireccional entre el Atmega328p y una computadora, enviando un texto que diga "Hola labo de Micros, pulse 1,2,3, o 4 para encender los ledsz recibir datos específicos de la computadora, recibe números del 1 al 4 y en función de esos números prende o apaga los leds.

2. Descripción

2.1. Introducción

Para el desarrollo de este trabajo de laboratorio se utilizo el diagrama de la Figura **??**.Se observo en la hoja de datos del microcontrolador atmega328p la corriente máxima de entrada/salida ($I = 20mA$), usando las leyes de ohm se obtiene la ecuación 1,con una Resistencia de 220Ω y la tensión para un diodo led verde, se obtuvo que la corriente de salida total es $I_{Out} = 10{,}9mA$, por lo cual es menor a la máxima permitida por pin del Atmega328p.

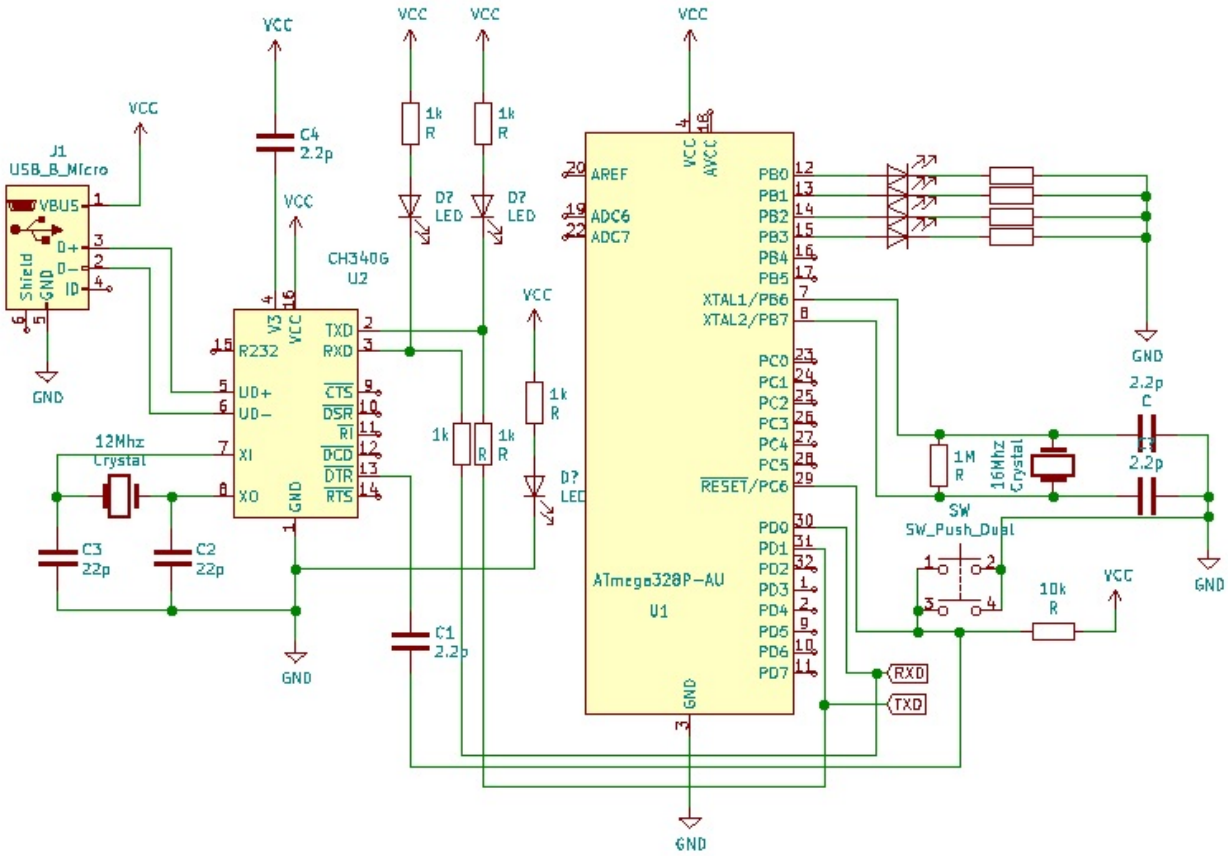

Figura 1: Diagrama de conexión

$$I_{Led} = \frac{V_{CC} - V_{Led}}{R_{Led}} \tag{1}$$

2.2. Comunicación Serial a través de la USART

La USART incluye recursos independientes para transmisión y recepción, esto posibilita una operación full-duplex, es decir, es posible enviar y recibir datos en forma simultánea. La terminal destinada para la transmisión de datos es TXD y para la recepción es RXD. Estas terminales corresponden con PD1 (TXD) y PD0 (RXD).

La comunicación puede ser síncrona o asíncrona, la diferencia entre estos modos se ilustra en la figura 2. En una comunicación síncrona, el emisor envía la señal de reloj, además de los datos, de esta forma el receptor va obteniendo cada bit en un flanco de subida o bajada de la señal de reloj, según se haya configurado. En una comunicación asíncrona el emisor únicamente envía los datos, no se envía señal de reloj.

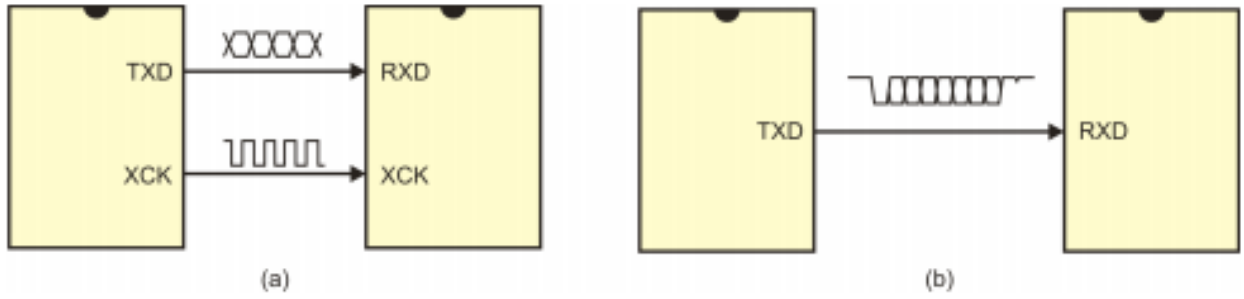

Figura 2: Comunicación serial (a) síncrona y (b) asíncrona

Para la transferencia de datos, para el transmisor y el receptor, se deben definir los siguientes parámetros:

Velocidad de transmisión (Baudrate): Se refiere a la cantidad de bits que se transmiten en un segundo, se mide en bits/segundo, usualmente referido como bauds o simplemente bps.

Número de bits de datos: Cada dato se integra por un número pre-definido de bits, pueden ser 5, 6, 7, 8 ó hasta 9 bits.

Bit de paridad: Es un bit de seguridad que acompaña a cada dato, se ajusta automáticamente en alto o bajo para complementar un número par de 1's (paridad par) o un número impar de 1's (paridad impar)

Número de bits de paro: Los bits de paro sirven para separar 2 datos consecutivos. En este caso, se debe definir si se utiliza 1 ó 2 bits de paro.

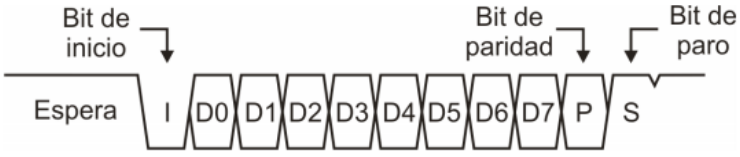

Figura 3: Trama serie de datos

La USART se compone de 3 bloques principales que son los registros UCSRA, UCSRB y UCSRC, con estos registros se realiza la configuración y se conoce el estado de la USART (UCSR, USART Configuration and State Register). Los bloques de transmisión y recepción son independientes entre sí. El bloque generador de reloj proporciona las señales de sincronización a los otros 2 bloques.

El Generador de Baud Rate incluye un contador descendente cuyo valor máximo es tomado del registro UBRR, el contador recarga su valor máximo cuando llega a 0 o cuando se escribe en UBRR[L]. Una señal interna es conmutada cada vez que el contador alcanza un 0, por

ello, la frecuencia base para CLK1 es la frecuencia del oscilador dividida entre UBRR + 1. Dependiendo del modo de operación de la USART, esta señal interna es dividida entre 2, 8 ó 16. La USART soporta 4 modos de operación que son, Normal asíncrono,Asíncrono a doble velocidad, Síncrono como maestro y Síncrono como esclavo.

La figura 4 muestra la organización del bloque para la transmisión, donde se observa que los datos a ser transmitidos deben ser escritos en el Registro de Datos de la USART (UDR, USART Data Register). Con UDR realmente se hace referencia a dos registros ubicados en la misma dirección, un registro sólo de escritura para transmitir datos y un registro sólo de lectura para recibirlos.

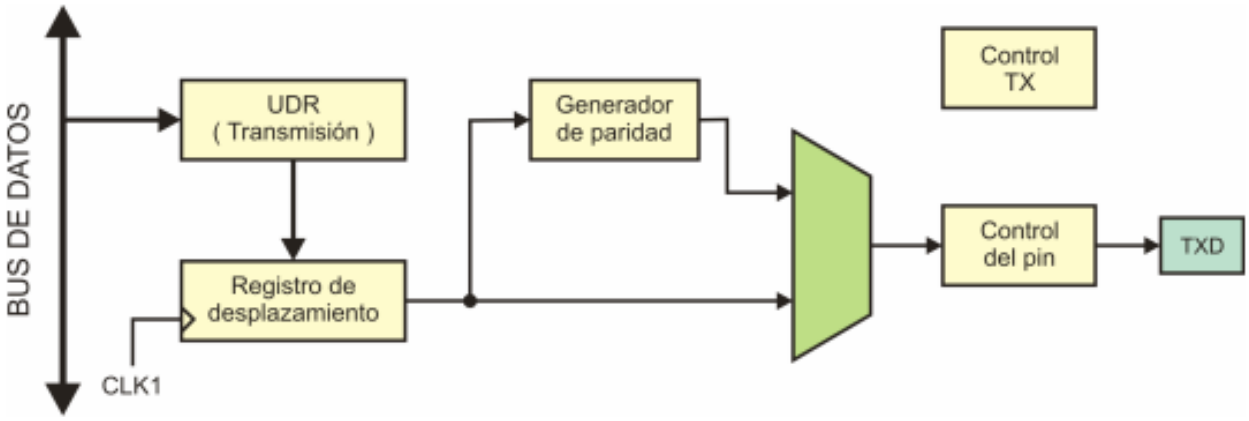

Figura 4: Bloque para la transmisión de datos

En la figura 5se muestra la organización del bloque para la recepción de datos. También contiene un Registro de Datos de la USART (UDR), aunque en este caso es un registro sólo de lectura empleado para recibir datos seriales por medio de la USART. La recepción serial se basa en un registro de desplazamiento que realiza la conversión de serie a paralelo. El receptor debe habilitarse para que en cualquier momento pueda recibir un dato, la habilitación se realiza con la puesta en alto del bit RXE (bit 4 del registro UCSRB)

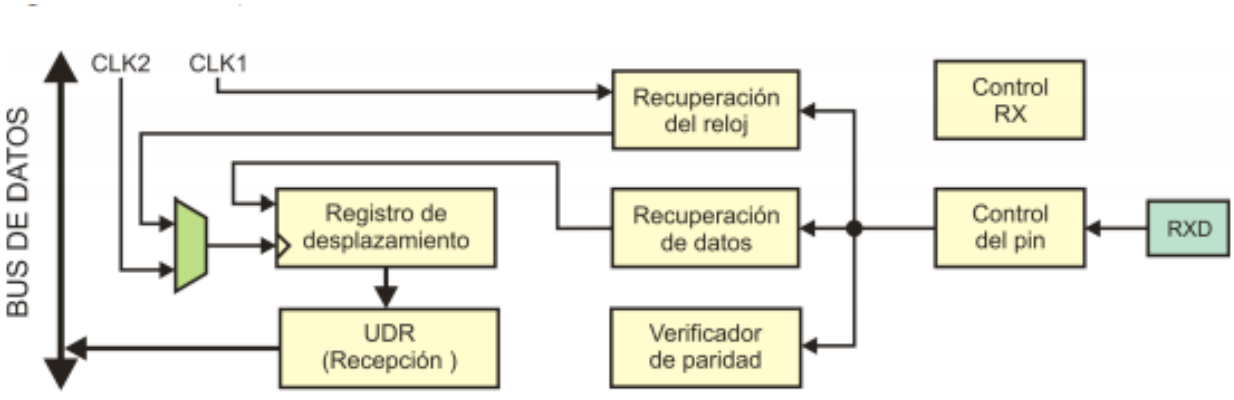

Figura 5: Bloque para la recepción de datos

2.3. Diagrama de Conexión del Circuito Físico

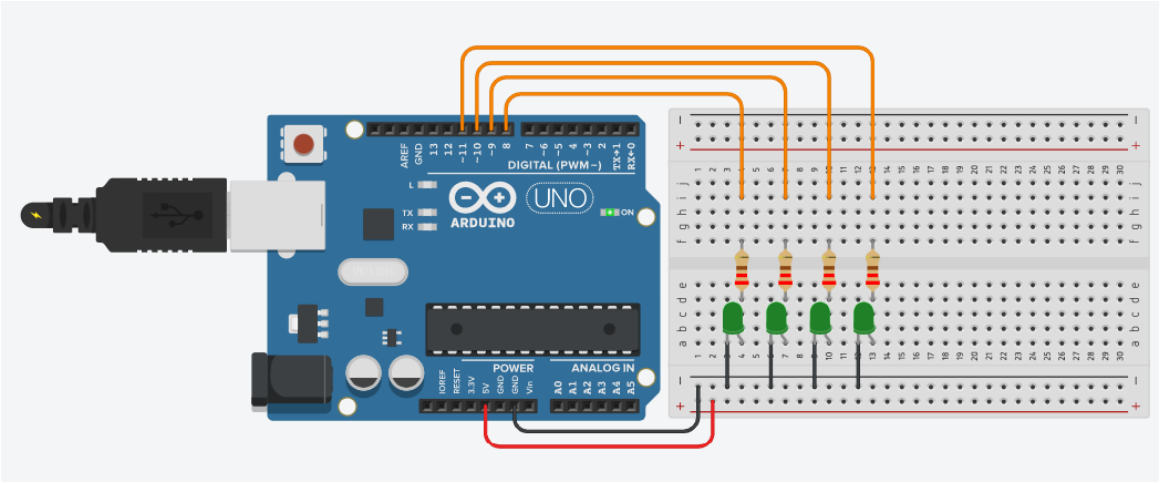

Figura 6: Diagrama del Circuito Físico

2.4. Esquema en Bloques

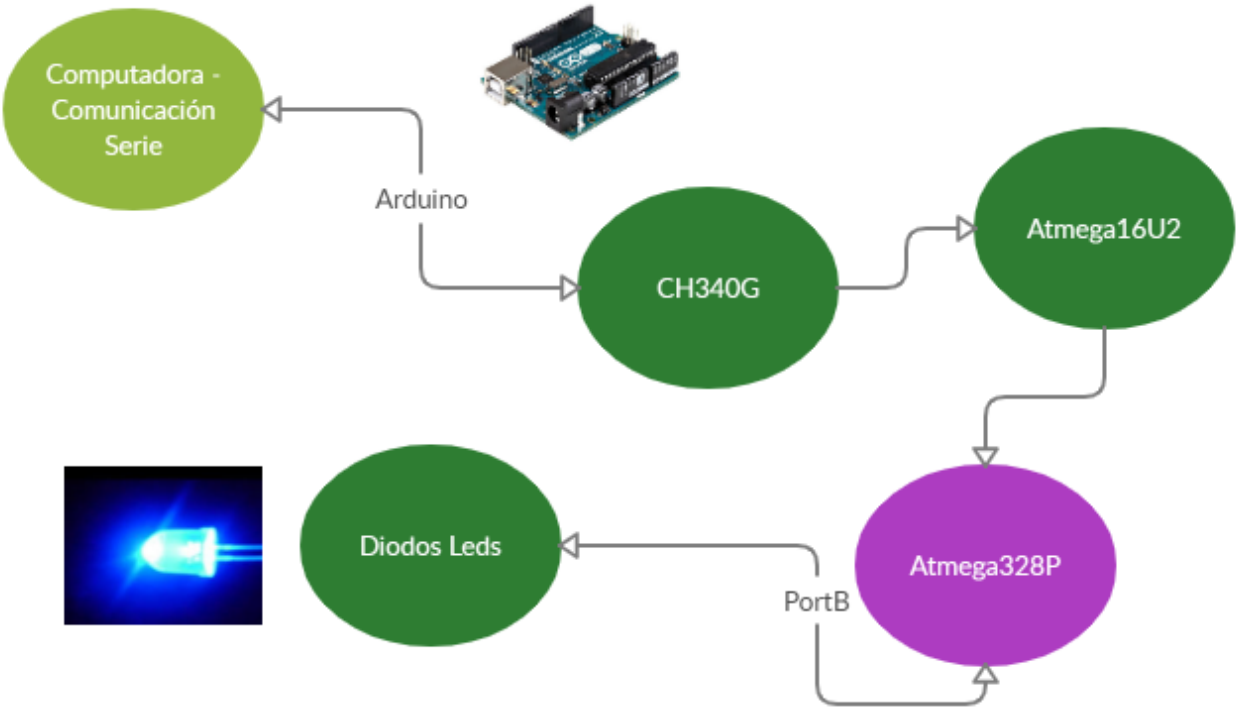

Figura 7: Diagrama en bloques

2.5. Materiales Utilizados

- x4(10 Ars)Diodo Led verde de 3mm (Alta Luminosidad).
- x4(10 Ars)Resistor de 220Ω de carbón 1/4 Watt.
- 1x(725 Ars)Placa Arduino Uno con USB.
- 1x(96.5 Ars)Placa Experimental.
- 1x(10 Ars)Cables de conexión.
- Costo del Proyecto : 880 Ars

2.6. Diagrama de Flujo

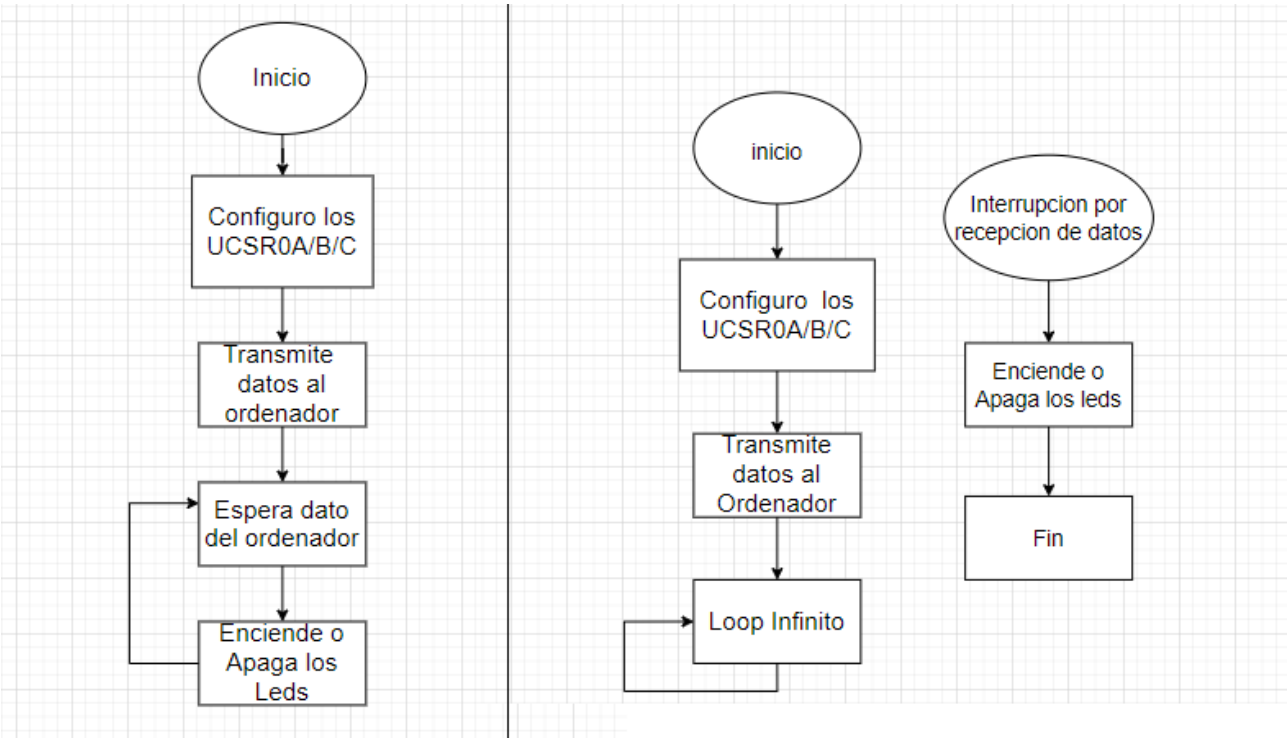

Figura 8: Diagrama de flujo sin interrupción (izq) con interrupción (der)

2.7. Lógica del Programa

Primero realizo todas las configuraciones de los puertos, configuro los UCSR0A/B/C para activar la recepción y la transmisión de datos, con 8 bits de datos, 1 bit de stop, sin bits de paridad y en modo asincrónico, primero transmito los datos, para que sea mas cómodo los cargo en un vector en ROM y los voy leyendo desde ahí, cuando termina de transmitir los datos, se pasa al bucle de recepción donde ahí se queda esperando hasta recibir un dato de la computadora, cuando recibe un dato valido, realiza la función programada y luego vuelve al loop de recepción, si recibe cualquier otro dato (no valido), simplemente se ignora y vuelve al loop de recepción de datos.

2.8. Links de funcionamiento del circuito

Comunicacion Serie:`https://youtu.be/8gWrPvlLOR0`

3. Código

3.1. Código sin interrupción de recepción de datos

```
.INCLUDE "M328PDEF.INC"

.MACRO LED
        SBIC PORTB,@0           ;MIRO EL ESTADO DEL LED
        JMP APAGO
        SBI PORTB,@0
        JMP LOOP
APAGO:
        CBI PORTB,@0
        JMP LOOP
.ENDM

.CSEG
.ORG 0X0000
        JMP MAIN

.ORG INT_VECTORS_SIZE

MAIN:
        LDI R16,0X00
        OUT DDRD,R16                    ;PUERTO D COMO ENTRAD
        LDI R16,0XFF
        OUT DDRB,R16                    ;PUERTO B COMO SALIDA

        LDI R16,1<<RXEN0|1<<TXEN0
        STS     UCSR0B,R16
        LDI R16,1<<UCSZ01|1<<UCSZ00|0<<UMSEL00
        STS UCSR0C,R16  ;8-BIT,SIN PARIDAD,1 BIT DE STOP,ASINCRONICO
        LDI R16,0X00
        STS UBRR0H,R16  ;9600 BAUD RATE-->UBRR 0X68
        LDI R16,0X68
        STS UBRR0L,R16
        LDI R16,0X00
        CALL PUNTEROS

TR:
        LPM R18,Z+
        CPI R18,0
        BREQ LOOP

        CALL TRANSMITIR
        JMP TR

LOOP:
        LDS R17,UCSR0A
        SBRS R17,RXC0           ;VEO SI SE RECIBIO UN DATO
```

```
        JMP LOOP
        LDS R17,UDR0            ;PASO EL DATO AL REG
        CPI R17,'1'
        BREQ ES_UNO
        CPI R17,'2'
        BREQ ES_DOS
        CPI R17,'3'
        BREQ ES_TRES
        CPI R17,'4'
        BREQ ES_CUATRO
        JMP LOOP
;SI LLEGO ACA NO RECIBIO UN DATO VALIDO, VUELVE A LEER

ES_UNO:
        LED 0
ES_DOS:
        LED 1
ES_TRES:
        LED 2
ES_CUATRO:
        LED 3

TRANSMITIR:
        LDS R17,UCSR0A
        SBRS R17,UDRE0
        JMP TRANSMITIR
        STS UDR0,R18            ;TRANSMITO EL DATO
        RET

PUNTEROS:       ;INICIALIZA LOS PUNTEROS EN ROM
        LDI ZL,LOW(TABLA_ROM<<1)
        LDI ZH,HIGH(TABLA_ROM<<1)
        RET

TABLA_ROM: .db "*** Hola Labo de Micro ***", 10, 13,"Escriba 1, 2, 3
o 4 para controlar los LEDs", 10, 13, 0
```

3.2. Código con interrupción de recepción de datos

```
.INCLUDE "M328PDEF.INC"

.MACRO LED
        SBIC PORTB,@0           ;MIRO EL ESTADO DEL LED
        JMP APAGO
        SBI PORTB,@0
        RETI
APAGO:
        CBI PORTB,@0
        RETI
.ENDM
```

```
.CSEG
.ORG 0X0000
        JMP MAIN

.ORG URXCaddr               ;0x0024
        JMP RECIBE

.ORG INT_VECTORS_SIZE

MAIN:
        LDI R16,0X00
        OUT DDRD,R16                    ;PUERTO D COMO ENTRAD
        LDI R16,0XFF
        OUT DDRB,R16                    ;PUERTO B COMO SALIDA

        LDI R16,1<<RXEN0|1<<TXEN0|1<<RXCIE0
        STS     UCSR0B,R16      ;HABILITO INTERRUPCIONES
        LDI R16,1<<UCSZ01|1<<UCSZ00|0<<UMSEL00
        STS UCSR0C,R16  ;8-BIT,SIN PARIDAD,1 BIT DE STOP,ASINCRONICO
        LDI R16,0X00
        STS UBRR0H,R16  ;9600 BAUD RATE-->UBRR 0X68
        LDI R16,0X68
        STS UBRR0L,R16
        LDI R16,0X00
        CALL PUNTEROS

        SEI

TR:
        LPM R18,Z+
        CPI R18,0
        BREQ LOOP_PRINCIPAL
        CALL TRANSMITIR
        JMP TR

LOOP_PRINCIPAL:
        NOP
        JMP LOOP_PRINCIPAL

RECIBE:
        LDS R17,UDR0                ;PASO EL DATO AL REG
        CPI R17,'1'
        BREQ ES_UNO
        CPI R17,'2'
        BREQ ES_DOS
        CPI R17,'3'
        BREQ ES_TRES
        CPI R17,'4'
        BREQ ES_CUATRO
        RETI
```

```
ES_UNO:
        LED 0
ES_DOS:
        LED 1
ES_TRES:
        LED 2
ES_CUATRO:
        LED 3


TRANSMITIR:
        LDS R17,UCSR0A
        SBRS R17,UDRE0
        JMP TRANSMITIR
        STS UDR0,R18                ;TRANSMITO EL DATO
        RET

PUNTEROS:       ;INICIALIZA LOS PUNTEROS EN ROM
        LDI ZL,LOW(TABLA_ROM<<1)
        LDI ZH,HIGH(TABLA_ROM<<1)
        RET

TABLA_ROM: .db "*** Hola Labo de Micro ***", 10, 13,"Escriba 1, 2, 3
o 4 para controlar los LEDs", 10, 13, 0
```

4. Resultados y Conclusiones

4.1. Resultados

Se logro transmitir y recibir datos correctamente se puede ver el funcionamiento del mismo en la sección 2.8 a través de un vídeo en YouTube.

En el enunciado preguntaba si hacia falta configurar un delay por la conexión usb, pero no hizo falta del mismo por que empieza a transmitir cuando se activa la conexión serie del Arduino IDE, así que no se pierde la transmisión de datos.

4.2. Conclusiones

Aprendí a leer bien la hoja de datos del Atmega328p para poder configurar los registros UCSR0A/B/C para que la comunicación serie funcione como esperaba, también aprendí a calcular el baudrate, ya que la primera vez no me funcionaba por que no lo había configurado a través de la UBRR0H/L. Para la comunicación serie se utilizo el arduino ide.

Como se puede observar cambiando un poco el código se logra evitar que el microcontrolador este siempre leyendo, si se recibió un dato, activando la interrupción de recepción de datos.