Fiuba
Facultad de Ingeniería
Universidad de Buenos Aires

Laboratorio de Microprocesadores (86.07)

Proyecto:

Trabajo Practico Nº 3

Puerto Serie

Profesor:	Ing. Guillermo Campiglio
Cuatrimestre / Año:	1 er cuatrimestre 2021
Turno de clases prácticas:	Miércoles
Jefe de Trabajos Prácticos:	Ing. Pedro Ignacio Martos
Docente guía:	Ing. Fabricio Baglivo

Autores			**Seguimiento del proyecto**								
Nombre	**Apellido**	**Padrón**									
Gonzalo	Puy	99784									

Observaciones:

Fecha de aprobación		

Firma J.T.P

COLOQUIO	
Nota Final	
Firma Profesor	

Índice

1. Introducción

En este trabajo se buscará establecer comunicación bidireccional entre un microcontrolador AVR de 8 bits y una computadora de escritorio.

2. Desarrollo

2.1. Banco de mediciones

Para el trabajo se utilizara el siguiente banco de mediciones, compuesto por

- Placa de desarrollo Arduino "UNO" y su respectivo cable para conectar la placa a la PC.
- El microcontrolador a usar, es el que viene integrado en la placa Arduino: `Atmega328P`.
- Protoboard
- Cables macho-macho para conexión del protoboard y de la placa Arduino.
- 4 LEDs (2 de color Rojo y 2 de color amarillo).
- 4 resistencias de 220 Ω.

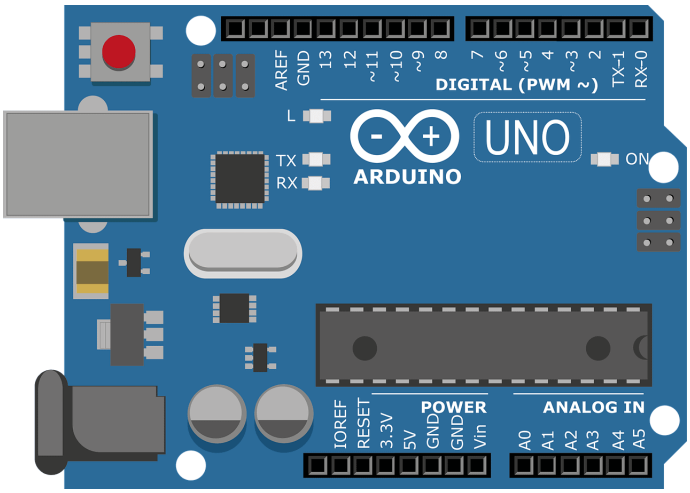

Figura 1: Placa de desarrollo Arduino UNO.

Las conexiones usadas se muestran en las siguientes figuras

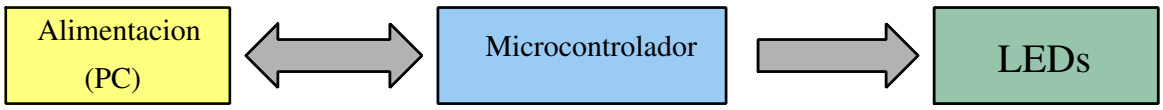

Figura 2: Diagrama de bloques.

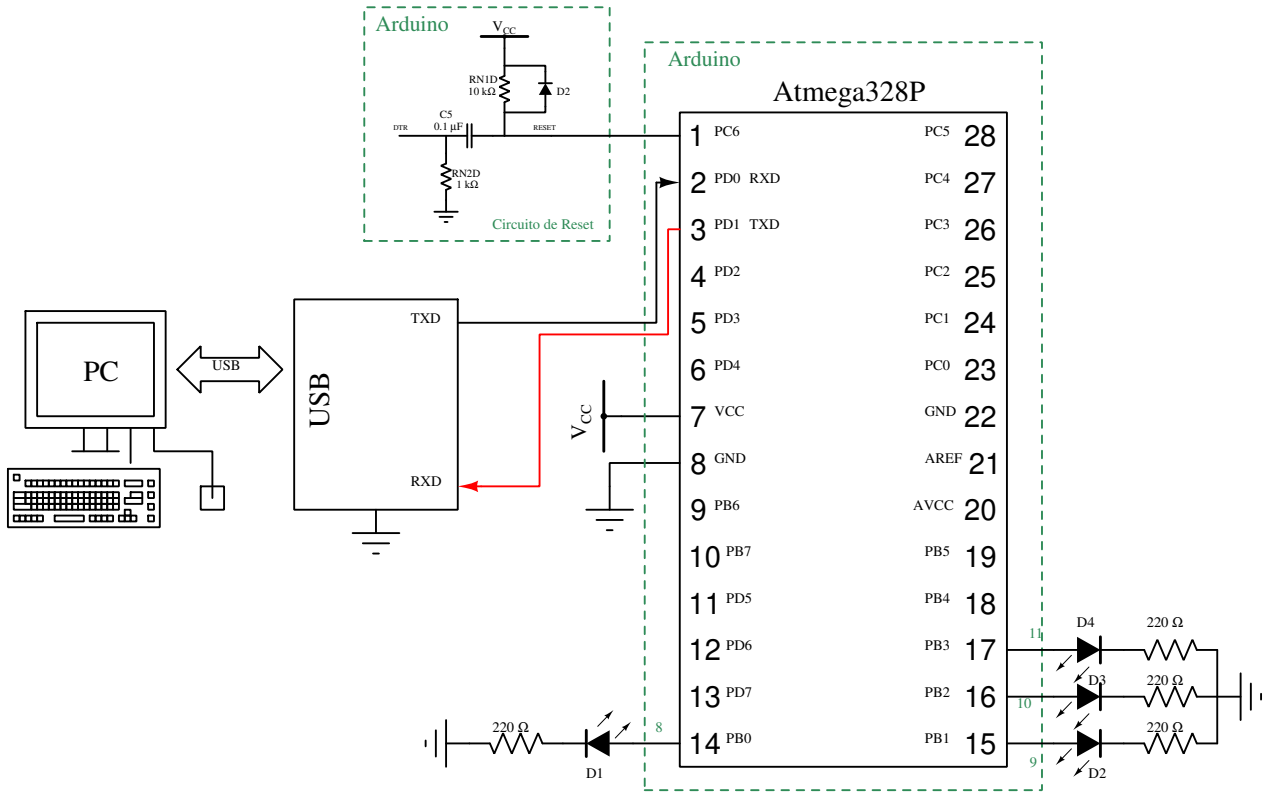

Figura 3: Conexión utilizada para el trabajo.

2.2. Programa a implementar

Al encender el microcontrolador, el programa transmitirá el texto

```
*** Hola Labo de Micro ***
Escriba 1,2,3 o 4 para controlar los
LEDs
```

El cual sera mostrado en el terminal serie (PC). Si en este terminal serie se presiona la tecla '1', entonces se enciende/apaga el LED 1. Si se presiona la tecla '2', ocurre lo propio con el LED 2 y así para los 4 LEDs.

Para lograr lo deseado se utilizará el Transmisor-Receptor Universal Sincrónico/Asincrónico (USART) del microcontrolador ATMega328P, configurado de la siguiente manera

- Velocidad de 9600 bps o *baudrate*. Esto significa que al dar la orden de transmitir o recibir un carácter, se sabe que los dígitos binarios se transmitirán a razón de 1/9600 segundos.
- Se adoptará el mecánico mas común de sincronización, es decir, comunicación asincrónica. Esto significa que no se tendrá una señal de *clock* que acompañe los datos, sino una estructura de datos adicional que enmarca la información y permite sincronizar transmisor con receptor.
- Se configurará un largo de caracteres de 8 bits de datos, sin paridad y con 1 bit de *stop*.

A continuación se muestra el diagrama de flujo del programa, con el que se explica de forma simplificada el funcionamiento del programa.

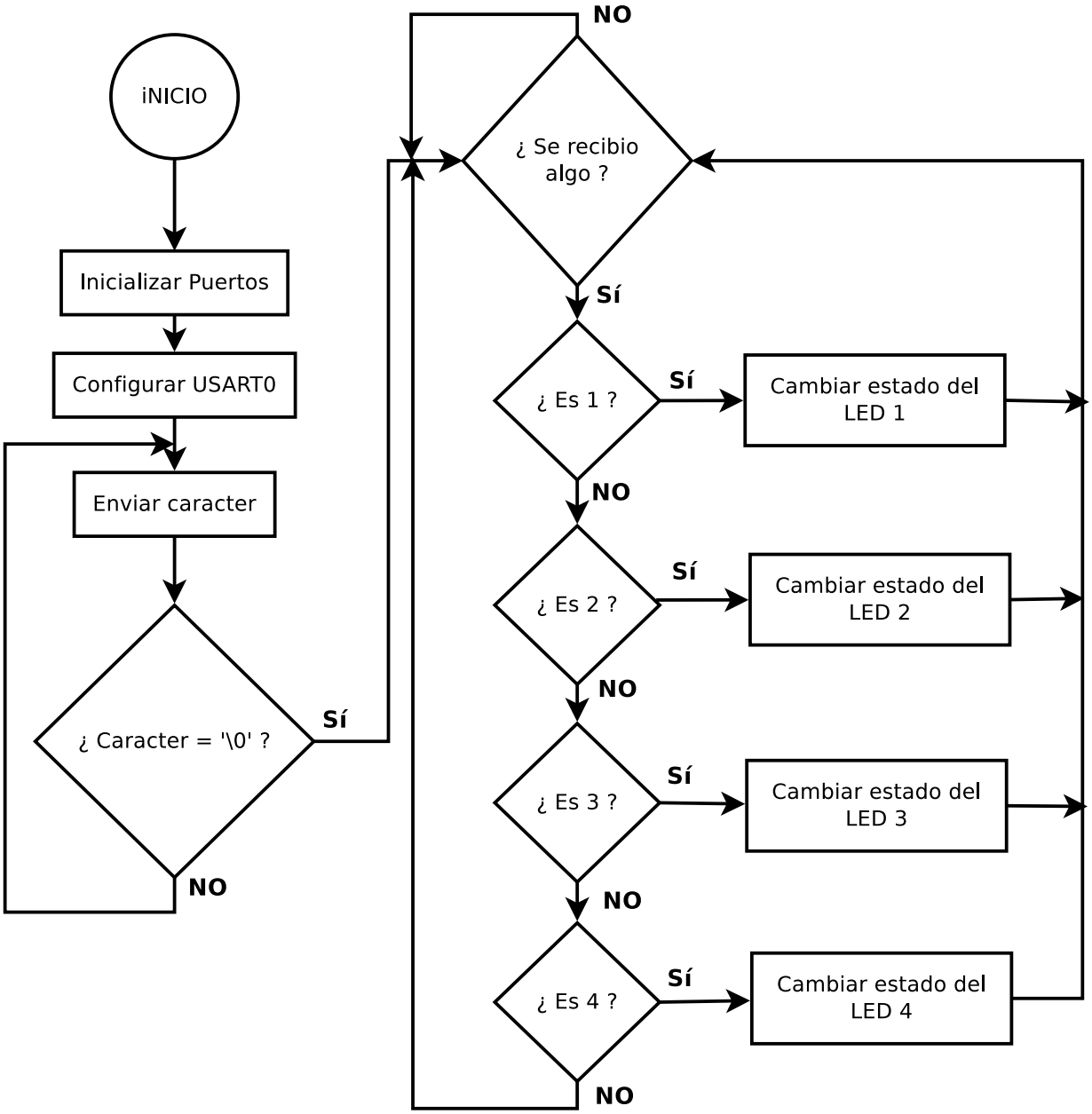

Figura 4: Diagrama de flujo.

2.3. Resultados

Se logró lo pedido sin mayores dificultades. El control de los LEDs funcionó correctamente con el código implementado. A continuación, se agrega el enlace a un vídeo en el cual se muestra la puesta a prueba del programa.

2.4. Código implementado

Para finalizar, se adjunta el código implementado para el programa propuesto en este trabajo.

```
;
; TP3 - Puerto Serie
;
; Created: 3/7/2021 20:57:45
; Autor : Puy Gonzalo
; Padron : 99784
;

.include "m328pdef.inc"
.include "MACROS.inc"
.include "DEFINES.inc"

.CSEG
.ORG 0x0000
        JMP     CONFIG

.ORG INT_VECTORS_SIZE

CONFIG:
        initSP
        initPORTS
        initUSART

MAIN:
        CALL    DISPLAY_MSG
RECIBIR:
        CALL    GET_DATA
        CALL    SET_LEDs
        JMP     RECIBIR


.INCLUDE "ENVIARYRECIBIR.inc"
.INCLUDE "LEDs.inc"



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

Listing 1: Código del programa

```
;
; Defines.inc
;
; Este archivo contiene .EQU y .DEF necesarios para el codigo
;
; Created: 5/6/2021 01:43:15
; Alumno : Puy Gonzalo
; Padron : 99784

.EQU    MSG_END = '\0'
.EQU    BAUDRATEH = 0X00
.EQU    BAUDRATEL = 0X67

.DEF    dummyreg = r16
.DEF    Caracter = r17
.DEF    Dato_R = r18
.DEF    Pin_Mask = r19
```

```
 ;
 ; Macros.inc
 ;
 ; Este archivo contiene Macros utiles para el progama.
 ;
 ;  Created: 5/6/2021 01:43:15
 ;  Autor  : Puy Gonzalo
 ;  Padron : 99784


 ; Esta macro inicializa el puntero Z a una posicion en ROM
; Uso:      initZ   <Etiqueta>
.MACRO  initZ
            LDI     ZH,HIGH(@0<<1)
            LDI     ZL,LOW(@0<<1)
.ENDMACRO


; Esta macro inicializa el stack pointer
; Uso:      initSP
.MACRO  initSP
            LDI dummyreg,LOW(RAMEND)
            OUT spl,dummyreg
            LDI dummyreg,HIGH(RAMEND)
            OUT sph,dummyreg
.ENDMACRO

; Esta macro inicializa los puertos correspondientes
; Uso:      initPORTS
.MACRO  initPORTS
; Puertos PB0-4 como salida
        LDI dummyreg,0x0F
        OUT DDRB,dummyreg
```

```
34 ; Puerto D como salida salvo PD0
35         LDI dummyreg,0xFE
36         OUT DDRD,dummyreg
37 .ENDMACRO
38
39
40
41 ; Esta macro configura USART0
42 ;
43 .MACRO  initUSART
44 ;Seteo Baud Rate en 96000 bps
45         LDI dummyreg,BAUDRATEH
46         STS UBRR0H,dummyreg
47         LDI dummyreg,BAUDRATEL
48         STS UBRR0L,dummyreg
49 ;Activo Transmisor y receptor.
50         LDI dummyreg, (1<<RXEN0) | (1<<TXEN0)
51         STS UCSR0B,dummyreg
52 ; Asincronico, Datos 8N1 (8-bits de datos, sin paridad y 1 bit de
       stop).
53         LDI dummyreg,6
54         STS UCSR0C,dummyreg
55 .ENDMACRO
```

```
1  ;
2  ;  ENVIARYRECIBIR.inc
3  ;
4  ; Created: 3/7/2021 15:04:46
5  ;  Alumno: Puy Gonzalo
6  ; Padron : 99784
7  ;
8
9  ; Mensaje incial
10 DISPLAY_MSG:
11         initZ   INIT_MSG
12 SET_CHAR:
13         LPM     Caracter,Z+
14         CPI     Caracter,MSG_END
15         BREQ    RETORNO
16 SEND:
17         LDS     dummyreg,UCSR0A
18         SBRS    dummyreg, UDRE0
19         JMP     SEND
20
21         STS     UDR0,Caracter
22         JMP     SET_CHAR
23 RETORNO:
24         RET
25
26 ; Recibir datos
27 GET_DATA:
```

```
28          LDS     dummyreg, UCSR0A
29          SBRS    dummyreg, RXC0
30          JMP     GET_DATA
31
32          LDS     Dato_R,UDR0
33          RET
```

```
1   ;
2   ; LEDs.inc
3   ;
4   ; Created: 3/7/2021 15:07:04
5   ; Alumno: Puy Gonzalo
6   ; Padron : 99784
7   ;
8
9    SET_LEDs:
10          CPI     Dato_R,1
11          BREQ    LED_1
12
13          CPI     Dato_R,2
14          BREQ    LED_2
15
16          CPI     Dato_R,3
17          BREQ    LED_3
18
19          CPI     Dato_R,4
20          BREQ    LED_4
21
22          RET
23
24  LED_1:
25          LDI     Pin_Mask,0b00000001
26          JMP     on_off
27
28  LED_2:
29          LDI     Pin_Mask,0b00000010
30          JMP     on_off
31
32  LED_3:
33          LDI     Pin_Mask,0b00000100
34          JMP     on_off
35
36  LED_4:
37          LDI     Pin_Mask,0b00001000
38          JMP     on_off
39
40  on_off:
41          IN      dummyreg,PORTB
42          EOR     dummyreg,Pin_Mask
43          OUT     PORTB,dummyreg
44          RET
```

2.5. Planteo extra

Como actividad adicional se propuso en el enunciado del trabajo practico la implementación de un programa con la misma funcionalidad pero utilizando la técnica de interrupciones para el caso de la recepción de los datos por puerto serie.

Se implementó dicho programa logrando un funcionamiento correcto del mismo. El Diagrama de flujo del programa y el código utilizado para este se adjunta a continuación.

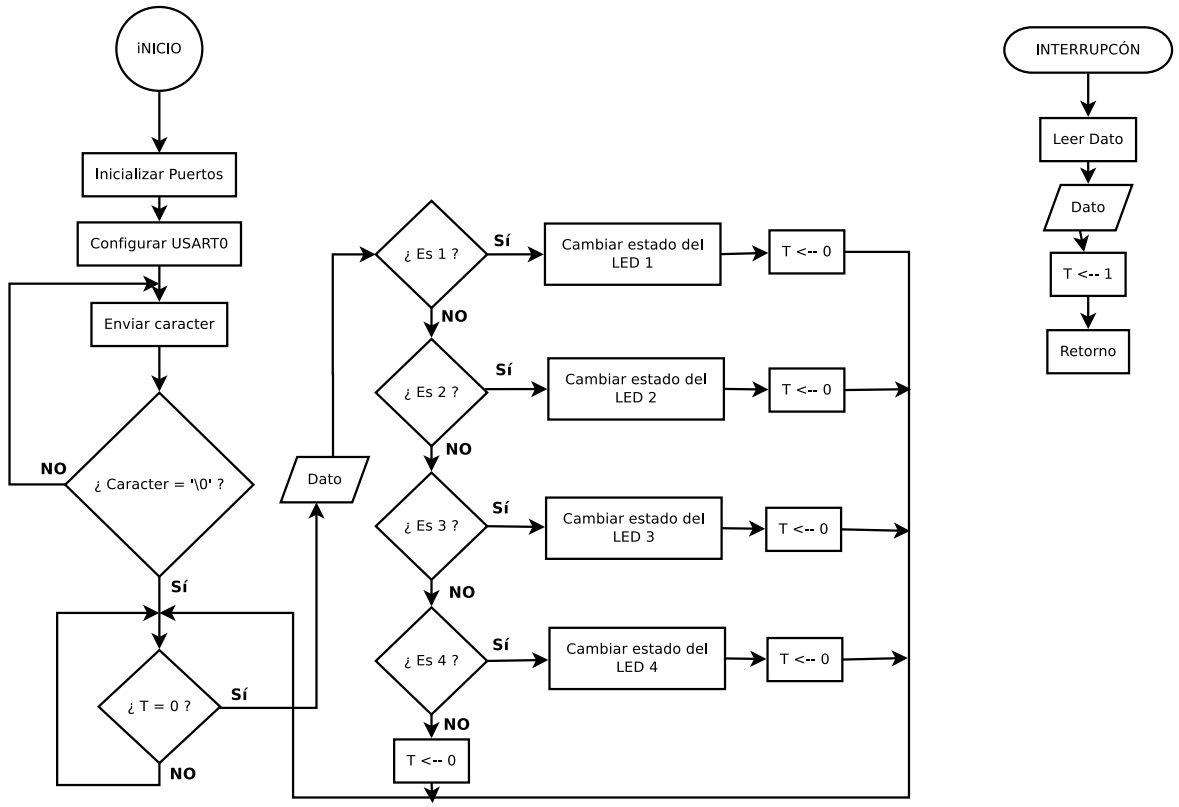

Figura 5: Diagrama de flujo.

```
;
; TP3 - Puerto Serie
; Actividad adicional dada por enunciado.
;
; Created: 3/7/2021 20:57:45
; Autor : Puy Gonzalo
; Padron : 99784
;

.include "m328pdef.inc"
.include "MACROS2.inc"
.include "DEFINES2.inc"

.CSEG
.ORG 0x0000
        JMP     CONFIG
.ORG URXCaddr
        JMP     isr_rxComplete

```

```
20 .ORG INT_VECTORS_SIZE
21
22 CONFIG:
23         initSP
24         initPORTS
25         initUSART
26
27 MAIN:
28         CALL    DISPLAY_MSG
29 LOOP:
30         BRTC    NO_DATA
31         CALL    SET_LEDs
32 NO_DATA:
33         JMP     LOOP
34
35 .INCLUDE "ENVIAR2.inc"
36 .INCLUDE "LEDs2.inc"
37 .INCLUDE "INTERRUPCION.inc"
38
39
40
41 INIT_MSG: .DB "*** Hola Labo de Micro ***", '\n', '\n', "Escriba
      1,2,3 o 4 para controlar los LEDs",'\0'
```

Listing 2: Código del programa

```
1  ;
2  ; Macros2.inc
3  ;
4  ; Este archivo contiene Macros utiles para el progama.
5  ;
6  ;  Created: 5/6/2021 01:43:15
7  ;  Autor  : Puy Gonzalo
8  ;   Padron : 99784
9
10
11  ; Esta macro inicializa el puntero Z a una posicion en ROM
12 ; Uso:      initZ   <Etiqueta>
13 .MACRO  initZ
14             LDI     ZH,HIGH(@0<<1)
15             LDI     ZL,LOW(@0<<1)
16 .ENDMACRO
17
18
19 ; Esta macro inicializa el stack pointer
20 ; Uso:      initSP
21 .MACRO  initSP
22             LDI dummyreg,LOW(RAMEND)
23             OUT spl,dummyreg
24             LDI dummyreg,HIGH(RAMEND)
25             OUT sph,dummyreg
26 .ENDMACRO
```

```
27
28 ; Esta macro inicializa los puertos correspondientes
29 ; Uso:      initPORTS
30 .MACRO  initPORTS
31 ; Puerto PB0-4 como salida
32         LDI dummyreg,0x0F
33         OUT DDRB,dummyreg
34 ;Puerto D como salida salvo PD0
35         LDI dummyreg,0xFE
36         OUT DDRD,dummyreg
37 .ENDMACRO
38
39
40
41 ; Esta macro configura USART0
42 ;
43 .MACRO  initUSART
44         ;Seteo Baud Rate en 96000 bps
45         LDI dummyreg,BAUDRATEH
46         STS UBRR0H,dummyreg
47         LDI dummyreg,BAUDRATEL
48         STS UBRR0L,dummyreg
49 ;Activo Transmisor, receptor e interrupcion para el receptor.
50         LDI dummyreg, (1<<RXCIE0) | (1<<RXEN0) | (1<<TXEN0)
51         STS UCSR0B,dummyreg
52 ; Asincronico, Datos 8N1 (8-bits de datos, sin paridad y 1 bit de
       stop).
53         LDI dummyreg,6
54         STS UCSR0C,dummyreg
55 ; Activo interrupciones globales
56         SEI
57 .ENDMACRO
```

```
1  ;
2  ; Define2.inc
3  ;
4  ; Este archivo contiene .EQU y .DEF necesarios para el codigo
5  ;
6  ; Created: 5/6/2021 01:43:15
7  ; Alumno : Puy Gonzalo
8  ; Padron : 99784
9
10 .EQU     MSG_END = '\0'
11 .EQU     BAUDRATEH = 0X00
12 .EQU     BAUDRATEL = 0X67
13
14 .DEF     dummyreg = r16
15 .DEF     Caracter = r17
16 .DEF     Dato_R = r18
17 .DEF     Pin_Mask = r19
```

```
;
;ENVIAR2.inc
;
; Created: 5/6/2021 01:43:15
; Alumno : Puy Gonzalo
; Padron : 99784

; Mensaje incial
DISPLAY_MSG:
        initZ   INIT_MSG
SET_CHAR:
        LPM     Caracter,Z+
        CPI     Caracter,MSG_END
        BREQ    RETORNO
SEND:
        LDS     dummyreg,UCSR0A
        SBRS    dummyreg, UDRE0
        JMP     SEND

        STS     UDR0,Caracter
        JMP     SET_CHAR
RETORNO:
        RET
```

```
;
; LEDs2.inc
;
; Created: 5/6/2021 01:43:15
; Alumno : Puy Gonzalo
; Padron : 99784

SET_LEDs:
        CPI     Dato_R,1
        BREQ    LED_1

        CPI     Dato_R,2
        BREQ    LED_2

        CPI     Dato_R,3
        BREQ    LED_3

        CPI     Dato_R,4
        BREQ    LED_4

        CLT     ;Hago Clear del Bit T de SREG
        CLR     Dato_R
        RET

LED_1:
        LDI     Pin_Mask,0b00000001
        JMP     on_off
```

```
28
29 LED_2:
30          LDI     Pin_Mask,0b00000010
31          JMP     on_off
32
33 LED_3:
34          LDI     Pin_Mask,0b00000100
35          JMP     on_off
36
37 LED_4:
38          LDI     Pin_Mask,0b00001000
39          JMP     on_off
40
41 on_off:
42          IN      dummyreg,PORTB
43          EOR     dummyreg,Pin_Mask
44          OUT     PORTB,dummyreg
45          CLT     ;Hago Clear del Bit T de SREG
46          CLR     Dato_R
47          RET
```

```
1  ;
2  ; INTERRUPCION.inc
3  ;
4  ; Created: 5/6/2021 01:43:15
5  ; Alumno : Puy Gonzalo
6  ; Padron : 99784
7
8  ; Rutina de Interrupcion
9  ; Guardo el dato recibido en un registro
10 ; Luego seteo el bit T de SREG en 1.
11
12  isr_rxComplete:
13         LDS     Dato_R,UDR0
14         SET
15         RETI
```

3. Conclusiones

Para concluir, se puede destacar la utilidad de este trabajo a la hora de aprender sobre la comunicación serial que posee el microcontrolador. Además, se presentó un panorama básico y general sobre los protocolos de comunicación de datos.