FACULTAD
DE INGENIERIA
Universidad de Buenos Aires

Laboratorio de Microprocesadores - 86.07

Puerto Serie

Profesor:			Ing. Guillermo Campiglio								
Cuatrimestre/Año:			1º/2020								
Turno de las clases prácticas			Miercoles 19 hs								
Jefe de trabajos prácticos:			Pedro Ignacio Martos								
Docente guía:			Pedro Martos, Fabricio Baglivo, Fernando Pucci								
Autores			Seguimiento del proyecto								
Nombre	Apellido	Padrón									
Leonel	Mendoza	101153									

Observaciones:

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

Fecha de aprobación				Firma J.T.P

Coloquio	
Nota final	
Firma profesor	

1. Objetivo

El objetivo de este trabajo es comunicar bidireccionalmente al microcontrolador con una PC de escritorio mediante Puerto Serie. Usar los registros del USART del microcontrolador, y generar interrupciones mediante transmision y recepción.

2. Descripción

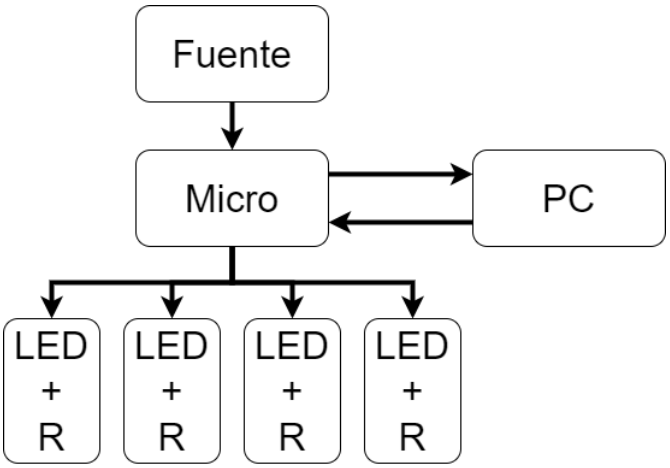
1) Al encender el microcontrolador, el programa deberá transmitir el texto:

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

El texto de arriba deberá mostrarse en el terminal serie

2) Si en el terminal serie se aprieta la tecla '1', entonces se enciende/apaga el LED 1 (toggle). Si se aprieta la tecla '2', ocurre lo mismo con el LED 2 y así lo mismo para los cuatro LEDs.

3. Diagrama en bloques



4. Esquemático

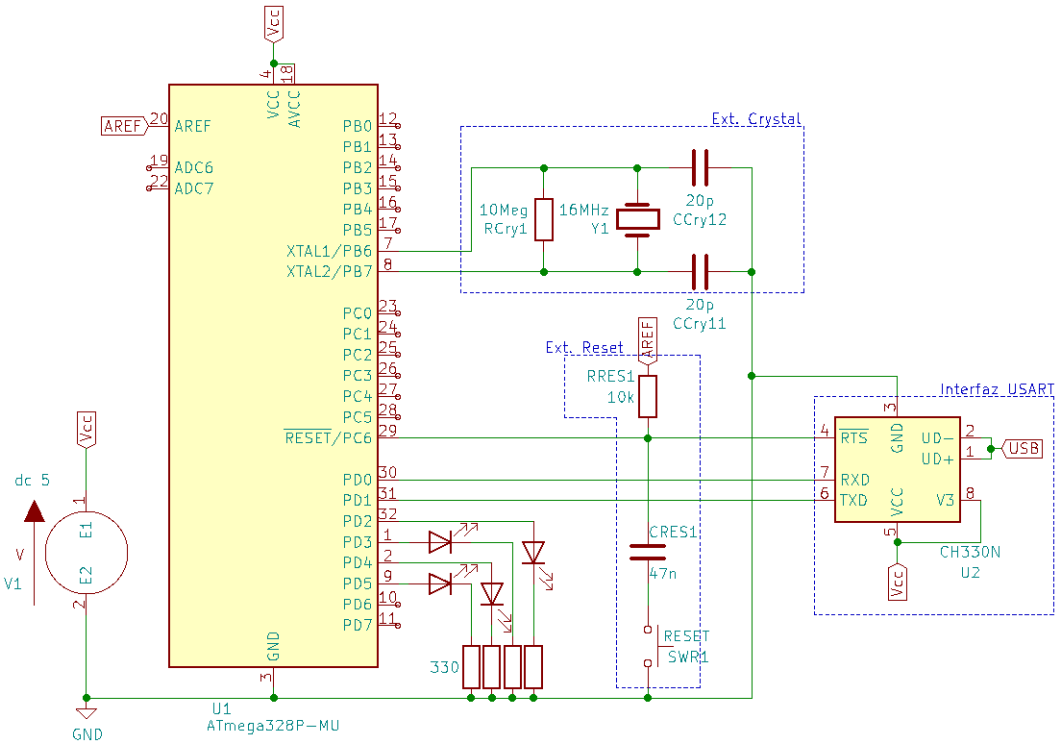
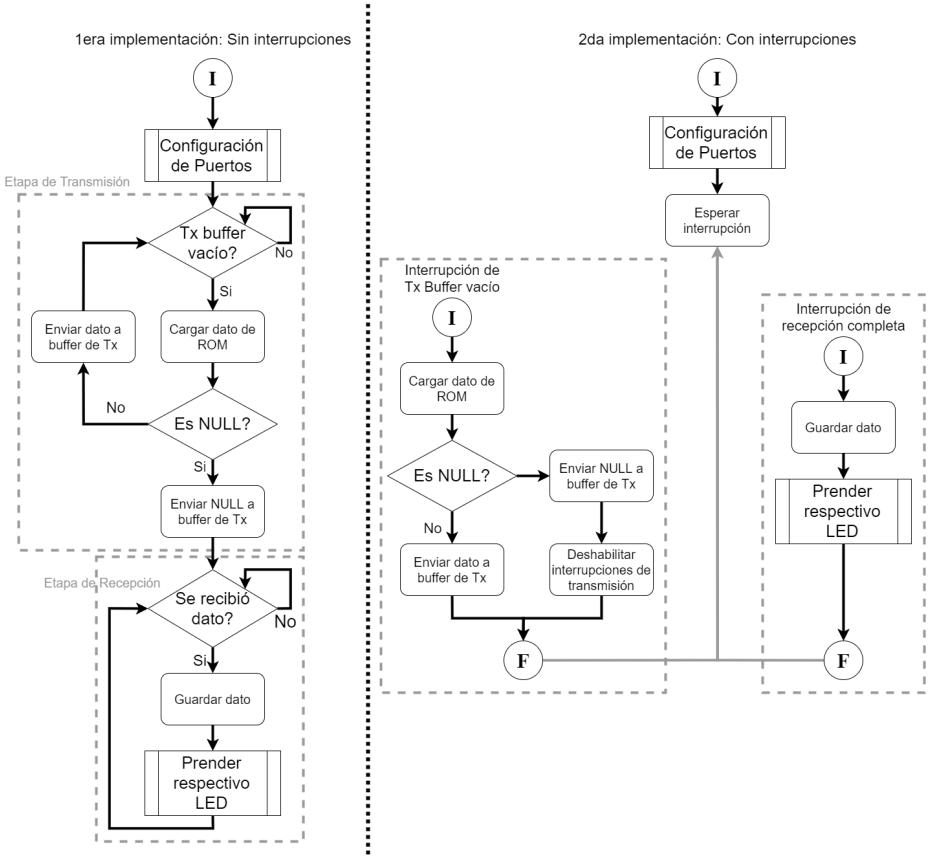


Figura 1: Esquemático del circuito

5. Listado de componentes

- Microcontrolador *ATmega328p* y programador USBasp (Arduino UNO) [AR$ 950]
- 4x LED [AR$ 40]
- 3x Resistencia (330 Ω) [AR$ 12]
- 2x Pulsador (10 $k\Omega$) [AR$ 30]

6. Diagrama de Flujo



7. Codigo de programa

7.1. Sin uso de interrupciones

```
.include "m328pdef.inc"

; * * * * * * * * * *
;   START MACROS  ;
; * * * * * * * * * *

.MACRO SET_SP ;[auxGPR]
    LDI @0, low(RAMEND)
    OUT SPL, @0
    LDI @0, high(RAMEND)
    OUT SPH, @0
.ENDM

.MACRO SET_X ;[LABEL to data memory]
    LDI XL, low(@0)
    LDI XH, high(@0)
.ENDM

.MACRO SET_Y ;[LABEL to data memory]
    LDI YL, low(@0)
    LDI YH, high(@0)
```

```
.ENDM

.MACRO SET_Z ;[LABEL to prog memory]
    LDI ZL, low(@0 << 1)
    LDI ZH, high(@0 << 1)
.ENDM

; * * * * * * * * * *
;   END MACROS    ;
; * * * * * * * * * *



.DEF aux = R16
.DEF rdata = R17
.DEF regdata = R18

.EQU LED1 = PD2
.EQU LED2 = PD3
.EQU LED3 = PD4
.EQU LED4 = PD5

.CSEG

    .ORG 0X0000                 ; En esta direccion escribo la instruccion JMP conf
    JMP conf

    .ORG INT_VECTORS_SIZE       ; Direccion donde escribir el codigo

conf:
    SET_SP aux

    LDI aux, 0b00111110     ; PD0 entrada (Rx), PD1 salida (Tx), PD2/3/4/5 salida (LEDs)
    OUT DDRD, aux

    LDI aux, 0x00           ; 9600 BAUD @ 16Mhz ? UBRR0 = 103 (tabla ATmega32 datasheet)
    STS UBRR0H, aux
    LDI aux, 103
    STS UBRR0L, aux

    LDI aux, (1 << UCSZ01 | 1 << UCSZ00)    ; format ? 8bit data + 1bit stop
    STS UCSR0C, aux

    LDI aux, (1<<RXEN0 | 1<<TXEN0)
    STS UCSR0B, aux
main:
    SET_Z   MSJ
    CALL    transmit
wait:
    LDS     aux, UCSR0A ; espero recibir datos
    SBRS    aux, RXC0
    RJMP    wait
    LDS     rdata, UDR0
    CALL    toggle_led
    RJMP    wait

transmit:
    LDS     aux, UCSR0A
    SBRS    aux, UDRE0
```

```
    RJMP    transmit    ; cuando el buffer de tx esta vacio, mando otro dato
    LPM     aux, Z+
    CPI     aux, 0
    BREQ    end_transmit    ; si no hay mas datos apago la interrupcion
    STS     UDR0, aux
    RJMP    transmit
end_transmit:
    STS     UDR0, aux ; transmito un ultimo null
    RET


toggle_led:     ; prendo el led requerido
    IN      regdata, PORTD      ; cargo el estado del puerto
    CPI     rdata, '1'
    BREQ    led1_toggle
    CPI     rdata, '2'
    BREQ    led2_toggle
    CPI     rdata, '3'
    BREQ    led3_toggle
    CPI     rdata, '4'
    BREQ    led4_toggle
    JMP     endrx
led_on:
    EOR     regdata, aux
    OUT     PORTD, regdata
endrx:
    RET
led1_toggle:
    LDI     aux, 0b00000100
    JMP     led_on
led2_toggle:
    LDI     aux, 0b00001000
    JMP     led_on
led3_toggle:
    LDI     aux, 0b00010000
    JMP     led_on
led4_toggle:
    LDI     aux, 0b00100000
    JMP     led_on

.ORG    0x500   MSJ: .db ' ', '*', '*', '*', 'H', 'o', 'l', 'a', ' ', 'L', 'a', 'b', 'o', ' '
```

7.2. Con uso de interrupciones

```
.include "m328pdef.inc"

; * * * * * * * * * *
;   START MACROS  ;
; * * * * * * * * * *

.MACRO SET_SP ;[auxGPR]
    LDI @0, low(RAMEND)
    OUT SPL, @0
    LDI @0, high(RAMEND)
    OUT SPH, @0
.ENDM

.MACRO SET_X ;[LABEL to data memory]
    LDI XL, low(@0)
    LDI XH, high(@0)
```

```
.ENDM

.MACRO SET_Y ;[LABEL to data memory]
    LDI YL, low(@0)
    LDI YH, high(@0)
.ENDM

.MACRO SET_Z ;[LABEL to prog memory]
    LDI ZL, low(@0 << 1)
    LDI ZH, high(@0 << 1)
.ENDM

; * * * * * * * * * *
;   END MACROS    ;
; * * * * * * * * * *

.DEF aux = R16
.DEF rdata = R17
.DEF regdata = R18

.EQU LED1 = PD2
.EQU LED2 = PD3
.EQU LED3 = PD4
.EQU LED4 = PD5

.CSEG

    .ORG 0X0000                 ; En esta direccion escribo la instruccion JMP conf
    JMP conf

    .ORG UDREaddr               ; Interrupcion de buffer vacio de tx
    JMP isr_txempty

    .ORG UTXCaddr               ; Interrupcion de fin de transmision
    JMP isr_txdone

    .ORG URXCaddr               ; Interrupcion de fin de recepcion
    JMP isr_rxdone

    .ORG INT_VECTORS_SIZE       ; Direccion donde escribir el codigo

conf:
    SET_SP aux

    LDI aux, 0b00111110     ; PD0 entrada (Rx), PD1 salida (Tx), PD2/3/4/5 salida (LEDs)
    OUT DDRD, aux
    LDI aux, 0b00100000     ; PB5 como salida para comunicar estados de lectura y escritura
    OUT DDRB, aux

    LDI aux, 0x00           ; 9600 BAUD @ 16Mhz ? UBRR0 = 103 (tabla ATmega32 datasheet)
    STS UBRR0H, aux
    LDI aux, 103
    STS UBRR0L, aux

    LDI aux, (1 << UCSZ01 | 1 << UCSZ00)    ; format ? 8bit data + 1bit stop
    STS UCSR0C, aux

    LDI aux, (0<<RXCIE0 | 1<<TXCIE0 | 1<<UDRIE0 | 0<<RXEN0 | 1<<TXEN0)
    ; habilito las interrupciones de transmision y tx buffer vacio
```

```
    STS UCSR0B, aux

    SEI
main:
    SET_Z MSJ
hold:
    NOP
    RJMP    hold

isr_txempty:    ; cuando el buffer de tx esta vacio, mando otro dato
    LPM     aux, Z+
    CPI     aux, 0
    BREQ    end_tx      ; si no hay mas datos apago la interrupcion
    STS     UDR0, aux
reti_txempty:
    RETI
end_tx:
    STS     UDR0, aux ; transmito un ultimo null
    LDI     aux, (1<<RXCIE0 | 0<<TXCIE0 | 0<<UDRIE0 | 1<<RXEN0 | 0<<TXEN0)
    STS     UCSR0B, aux ; deshabilito transmision, habilito recepcion e int recep completa
    RJMP    reti_txempty

isr_txdone:
    RETI

isr_rxdone:     ; recibo la orden para prender algun led
    LDS     rdata, UDR0
    IN      regdata, PORTD      ; cargo el estado del puerto
    CPI     rdata, '1'
    BREQ    led1_toggle
    CPI     rdata, '2'
    BREQ    led2_toggle
    CPI     rdata, '3'
    BREQ    led3_toggle
    CPI     rdata, '4'
    BREQ    led4_toggle
    JMP     endrx
led_on:
    EOR     regdata, aux
    OUT     PORTD, regdata
endrx:
    RETI
led1_toggle:
    LDI     aux, 0b00000100
    JMP     led_on
led2_toggle:
    LDI     aux, 0b00001000
    JMP     led_on
led3_toggle:
    LDI     aux, 0b00010000
    JMP     led_on
led4_toggle:
    LDI     aux, 0b00100000
    JMP     led_on

.ORG    0x500   MSJ: .db ' ', '*', '*', '*', 'H', 'o', 'l', 'a', ' ', 'L', 'a', 'b', 'o', ' '
```

8. Resultados

Se logro establecer comunicación bidireccional con una PC mediante ambos métodos.

9. Conclusiones

Mediante la configuración de los registros de USART, se estableció comunicación bidireccional con una PC y se recibio (en la PC) el mensaje del microcontrolador, tambien se logró enviar órdenes para *togglear* los LEDs.

Tuvo que hacerse uso de PowerShell en Windows 10 para poder acceder al puerto y leer/escribir del/al microcontrolador correctamente, segun el siguiente procedimiento:

```
#   Listar puertos disponibles, por nombre:

[System.IO.Ports.SerialPort]::getportnames()

#   Generar un objeto de la clase puertos:

$port= new-Object System.IO.Ports.SerialPort COMn,9600,None,8,one

#   Abrir el puerto:

$port.open()

#   Leer puerto:

$port.ReadLine()

#   Escribir puerto:

$port.Write()

#   Cerrar puerto:

$port.close()
```