Universidad de Buenos Aires
Facultad de Ingeniería
Año 2020 - 1^{ero} cuatrimestre

Laboratorio de Microprocesadores (86.07)

TRABAJO PRÁCTICO OBLIGATORIO 8

ESTUDIANTE:
Pintos Gaston 99711
gmpintos@fi.uba.ar

Índice

1. Objetivos

El trabajo práctico consiste en generar un programa capaz de establecer una comunicación serie bilateral asincronica entre el microcontrolador y una computadora de escritorio a través del transmisor USART incorporador. Se utilizo el microcontrolador *atmega 328P* de la placa *Arduino*. A lo largo del desarrollo se presentaran las herramientas utilizadas para la realización del proyecto.

2. Desarrollo:

2.1. Herramientas de Software:

Para el siguiente proyecto se utilizo la herramienta *AtmelStudio* para desarrollar el código en lenguaje *Assembly*. La misma herramienta permite simular el código escrito con lo cual fue posible realizar pruebas a lo largo del desarrollo. Además se utilizo la herramienta *AVRdude* para poder montar el programa en el microcontrolador.

2.1.1. Código de Montaje:

Se utilizo el siguiente código para cumplir la tarea asignada:

```
.INCLUDE "m328Pdef.inc"

.EQU END_MSJ = 0X5C      ;Codigo ASCCI para caracter '\'

.EQU POS_LED1 = 1
.EQU POS_LED2 = 5
.EQU POS_LED3 = 3
.EQU POS_LED4 = 4

.EQU PUERTO_SALIDA = PORTB
.EQU DDR_SALIDA = DDRB

.ORG 0X100
TABLA_ROM_MSJ: .DB 0b00001010, "*** Hola Labo de Micro ***", 0b00001010
TABLA_ROM_OPT: .DB 0b00001010, "Escriba 1, 2, 3 o 4 para controlar los LEDs \ "

.ORG 0x00
        RJMP MAIN

MAIN:
        LDI R16, HIGH(RAMEND)
        OUT SPH, R16
        LDI R16, LOW(RAMEND)
        OUT SPL, R16

        LDI R16, 0xFF
        OUT DDR_SALIDA, R16

        LDI R16,(1<<RXEN0)|(1<<TXEN0)|(1<<UCSZ02) ;Se habilita transmision y recepcion
        STS UCSR0B, R16
        LDI R16,(1<<UCSZ01)|(1<UCSZ00) ;Parity deshabilitado, 9 bits,
        STS UCSR0C, R16
        LDI R16, 103     ; Baud Rate = 9600
        STS UBRR0L, R16

        LDI ZH, HIGH(TABLA_ROM_MSJ <<1) ;Apunto hacia la tabla con MENSAJE
        LDI ZL, LOW(TABLA_ROM_MSJ <<1)
```

```
LOOP:
        LPM R17, Z+             ;Se almacena el valor de la tabla ROM en R17
        CPI R17, END_MSJ        ;Se compara con el caracter final
        BREQ END_MESSAGE        ;Si termino la lectura se manda el siguiente mensaje

TRANSMIT:
        LDS R16,UCSR0A
        SBRS R16,UDRE0          ;Esta el buffer listo para transmitir?
        RJMP TRANSMIT           ;Si no lo esta, se espera
        STS UDR0,R17
        RJMP LOOP

END_MESSAGE:

READ:
        LDS R16,UCSR0A
        SBRS R16,RXC0       ;Verifico si se recibio algun mensaje
        RJMP READ
OPTIONS:
        LDS R18,UDR0        ;Leo el dato recibido y comparo
        CPI R18, 0x31
        BREQ ENCENDER_UNO
        CPI R18, 0x32
        BREQ ENCENDER_DOS
        CPI R18, 0x33
        BREQ ENCENDER_TRES
        CPI R18, 0x34
        BREQ ENCENDER_CUATRO
        RJMP READ

ENCENDER_UNO:
        SBIS PUERTO_SALIDA, POS_LED1
        RJMP ENCENDER1
        RJMP APAGAR1
ENCENDER1:
        SBI PUERTO_SALIDA, POS_LED1
        CALL DELAY
        RJMP READ
APAGAR1:
        CBI PUERTO_SALIDA,POS_LED1
        CALL DELAY
        RJMP READ


ENCENDER_DOS:
        SBIS PUERTO_SALIDA, POS_LED2
        RJMP ENCENDER2
        RJMP APAGAR2
ENCENDER2:
        SBI PUERTO_SALIDA, POS_LED2
        CALL DELAY
        RJMP READ
APAGAR2:
        CBI PUERTO_SALIDA,POS_LED2
        CALL DELAY
        RJMP READ

ENCENDER_TRES:
        SBIS PUERTO_SALIDA, POS_LED3
        RJMP ENCENDER3
```

```
        RJMP APAGAR3
ENCENDER3:
        SBI PUERTO_SALIDA, POS_LED3
        CALL DELAY
        RJMP READ
APAGAR3:
        CBI PUERTO_SALIDA,POS_LED3
        CALL DELAY
        RJMP READ

ENCENDER_CUATRO:
        SBIS PUERTO_SALIDA, POS_LED4
        RJMP ENCENDER4
        RJMP APAGAR4
ENCENDER4:
        SBI PUERTO_SALIDA, POS_LED4
        CALL DELAY
        RJMP READ
APAGAR4:
        CBI PUERTO_SALIDA,POS_LED4
        CALL DELAY
        RJMP READ

DELAY:
; Delay 60 000 cycles
; 3.75ms at 16.0 MHz
   LDI  R20, 60
L1:LDI  R21, 4
L2:LDI  R22, 250
L3:DEC  R22
   BRNE L3
   DEC  R21
   BRNE L2
   DEC  R20
   BRNE L1
   RET
```

Se utilizo la directiva *.DEVICE* para identificar el tipo de microcontrolador utilizado. Luego con la directiva *.EQU* se definieron las constantes que van a ser utilizadas en el código y con la directiva *.DB* se introducen los mensajes que serán transmitidos a la terminal, los cuales comunican los números que se esperan recibir para encender los LED.

La primera tarea del *Main* es inicializar el Stack. Luego se configura el puerto B definido por la constante $PORT\_SALIDA$ en el cual serán conectados los 4 LED de salida.
Se configura el puerto serie USART0 interno del microcontrolador. Se encendieron los bits RXEN0, TXNE0 y UCZ02 del byte UCSR0B para habilitar la trasmisión y recepción de datos por el puerto serie. Luego se encienden los bits UCZ01 y UCZ00, que junto con el bit UCZO2, determinan la cantidad de bits de información a trasmitir. En este caso, el valor introducido UCZ0:2=111 configuran 9 bits de información. El valor fue elegido dado que al visualizar la terminal con 8 bits de información los caracteres no se trasmitían de manera correcta. En el byte UBRR0L se introduce el valor decimal 103 para establecer el baudrate en 9600 según la siguiente ecuación:

$$BAUD = \frac{f_{OSC}}{16(UBRROL+)} = \frac{16MHz}{16(103+1)} = 9615{,}38 \quad (1)$$

El programa comienza leyendo y trasmitiendo la tabla registrada en memoria con la utilización de un puntero. Para trasmitir el mensaje se utiliza el registro UDR0 en donde se guarda cada carácter a medida que se lee el mensaje. Una vez caracter de fin de mensaje, se procede a la sección de recepción. Para este caso, se verifica el valor del bit RXC0 del byte UCSR0A encargado de comunicar si se recibe algún dato. Este dato recibido se utiliza para reconocer cual LED debería ser encendido según la comparación entre el dato recibido, y el valor introducido en código ASCII. Según cada numero ingresado se enciende el LED correspondiente, o bien se apaga si se encuentra encendido, al ejecutar la función $ENCENDER\_N$ donde N sera el numero de LED.

Para este caso se utilizo un retardo de 3,75ms para las funciones encargadas de encender y apagar el LED.

$$\frac{ciclos}{frecuecia} = \frac{60000}{16MHz} = \frac{250 * 4 * 60}{16MHz} = 3{,}75ms \tag{2}$$

2.1.2. Montaje del código:

Paecuencia dera el montaje del código con la herramienta *AVR DUDE* a través del *ATMEL STUDIO* se utilizaron los siguientes comandos:
"C:9-Laboratorio de Microcomputadoras-6.3-mingw32(1).exe"-C
"C:9-Laboratorio de Microcomputadoras-6.3-mingw32(1).conf" -v
-p atmega328p -c arduino -P COM3 -D
-U flash:w:"$(MSBuildProjectDirectory)$(Configuration)$(OutputFileName).hex":i

Con esta herramienta fue posible que el código escrito fuera procesado por el microcontrolador para poder ejecutar las tareas programadas.

2.1.3. Diagramas de flujo:

Para una comprensión de manera gráfica se presenta a continuación un diagrama de flujo del código previamente explicado.

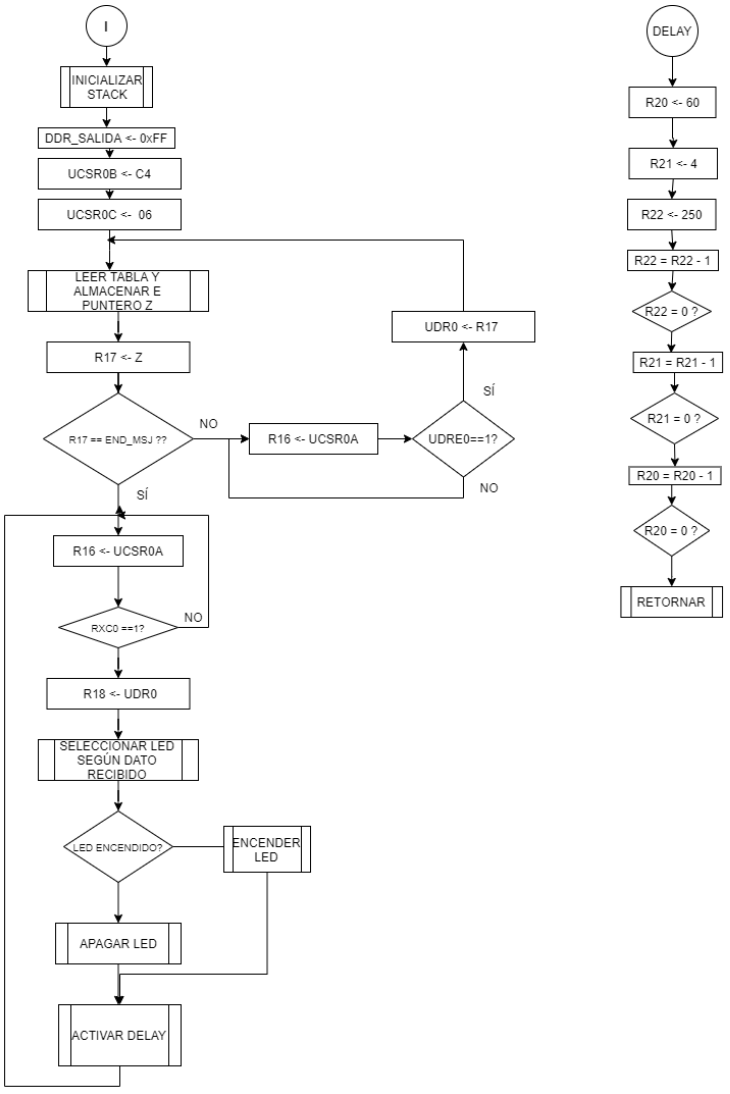

Figura 1: Diagrama de flujo del loop principal.

2.2. Herramientas de Hardware:

Para realizar el proyecto se utilizaron el microcontrolador de la placa *Arduino Uno*, un protoboard donde se montaron todos los elementos: 4 diodo led junto con 4 resistencias de 220Ω. Para la alimentación y trasmisión del código se utilizo el puerto USB de salida incorporado a la placa. Para interconexión entre la placa y el protoboard se utilizaron cables macho-macho. Se puede visualizar la conexión de los elementos en la siguiente imagen.

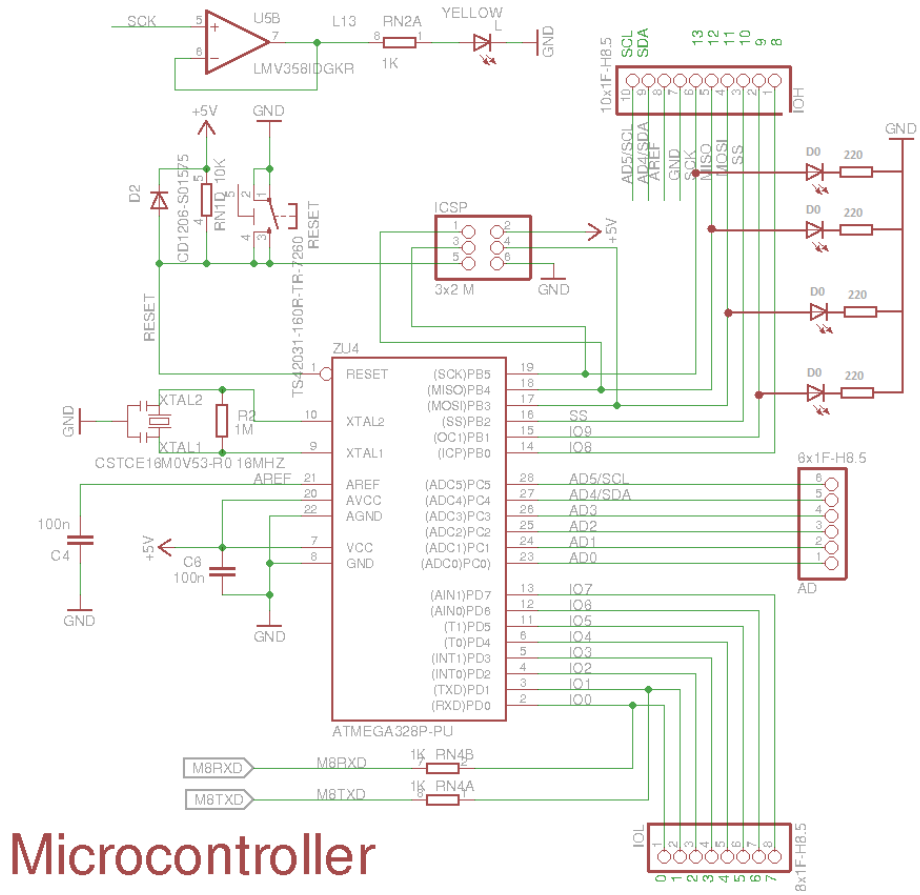

Figura 2: Diagrama de conexión.

La placa *Arduino Uno* cuenta con un microcontrolador *ATmega328p* . El ATmega328 proporciona comunicación serie UART TTL (5V), que está disponible en los pines digitales 0 (RX) y 1 (TX). Dentro de sus conexiones cuenta con 14 pines digitales de entrada/salida. Tiene una memoria flash de 32KB, una SRAM de 2KB y una EEPROM de 1KB. La velocidad del reloj es de 16MHz.

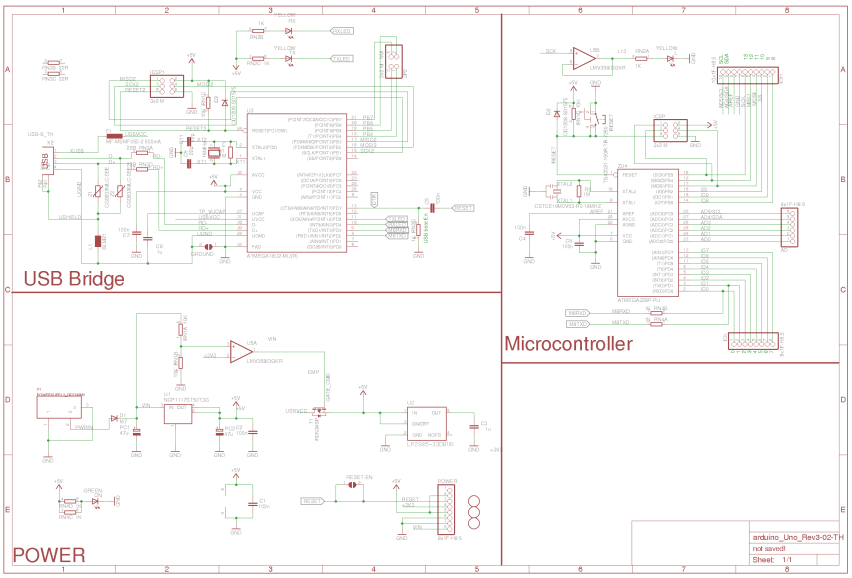

Figura 3: Diagrama de conexión de la placa.

2.2.1. Listado de componentes

Para este trabajo practico fueron utilizados los siguientes elementos. Se estima un coste total de 1400$.

Artículo	Precio
Arduino Uno	$800 - 1500$$
Led	$20 - 30$$
Resistencia	$5 - 10$$

Tabla 1: Precios de componentes.

3. Conclusiones

Para este trabajo practico se introdujo la comunicación serie bidireccional entre la terminal y el microcontrolador. Para establecer esta comunicación se utilizo el dispositivo USART integrado en el microcontrolador encargado de las comunicación por puerto serie. Se pudo establecer una comunicación asincronica con la computadora por parte del programa que a través del pooling corrobora la trasmisión y recepción de datos. Esta herramienta sirve para establecer la comunicación entre distintos dispositivos con el microcontrolador como hemos comprobado con este trabajo.

4. Bibliografía

- AVR Microcontroller and Embedded Systems: Using Assembly and C, 1stEdition, Muhammad Ali Mazidi, Pearson
- "Digital Design, Principles and Practices", 3rdEdition, J.Wakerly, Prentice-Hall
- Set de instrucciones AVR de 8 bits
- Execution cycle of the AVR architecture
- Guía de Trabajos Prácticos
- ATmega328P Datasheet