Universidad de Buenos Aires

Facultad de Ingeniería

Trabajo Práctico N° 4:
Puerto serie

ALUMNO

Mundani Vegega, Ezequiel **emundani@fi.uba.ar** **102312**

ASIGNATURA

86.07 - Laboratorio de Microprocesadores

7 de diciembre de 2022

Introducción y objetivos

En este trabajo se busca programar un microcontrolador ATMega328P en Assembler y hacer su circuito para poder encender y apagar cuatro LEDs desde la terminal de una computadora. Además se busca que el microcontrolador envíe un mensaje a la computadora.

Lista de materiales

- 4 resistores de 220 Ω
- 4 LEDs
- 1 microcontrolador ATmega328P
- 5 cables macho-macho para protoboard
- 1 protoboard
- 1 fuente de alimentación de 5V

Parte A

El software fue desarrollado con Microchip Studio, para luego ser ensamblado y subido al microprocesador con AVRDude. El programa consiste de un único archivo "main.asm".

El flujo del programa es el siguiente: primero se configuran los pines del puerto D, el 0 como entrada por ser el RX, el 1 como salida por ser TX y el 4, 5, 6 y 7 como salida dado a estos estarán conectados los pines.

Luego se configuran los tres status registers de la USART. Los parámetros importantes a establecer fueron habilitar TX y RX, establecer el tamaño de carácter en ocho y establecer el baude rate en 9600.

Se envía el mensaje "*** Hola Labo de Micro *** Escriba 1, 2, 3 o 4 para controlar los LEDs", cambiando de línea luego de la primera frase utilizando los caracteres de salto de linea de windows (\r y \n). Además se puso el carácter de fin de string de C al final (\0) para poder identificar correctamente cuándo termina el string. Dicho mensaje fue almacenado en la memoria del programa y se envía carácter por carácter utilizando el puntero Z (el \0 no se envía).

Finalmente se espera medio segundo y todo este proceso se repite indefinidamente. El esperar medio segundo es para no inundar la consola de mensajes. Todo el flujo dicho se puede observar en el siguiente diagrama:

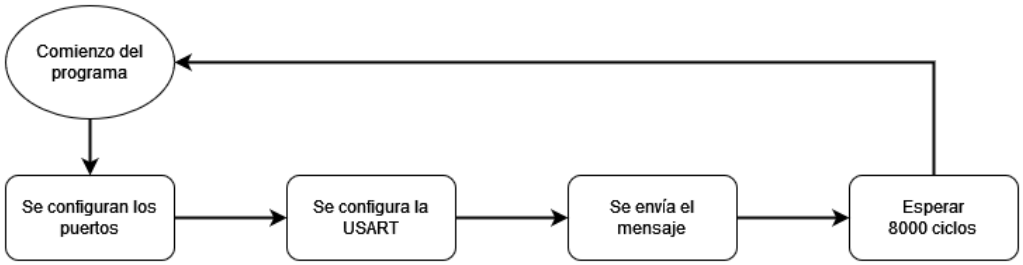

Figura 1: Diagrama del flujo del programa A.

Recepción y envío por medio de la USART

Se utilizó comunicación asincrónica, sin bit de paridad, utilizando un bit de stop de largo 1, con un baude rate de 9600 bds.

Se utilizó un registro para almacenar el último carácter recibido o el valor que se quiere transmitir. Primero se espera a que el buffer de la USART esté disponible para leer o escribir y luego se almacena en este el valor del registro (o se almacena en el registro desde el buffer, si es una recepción).

Para probar el envío y recepción se utilizó el software hercules. Desde este se estableció la conexión serie y se pudo leer el mensaje. También se probó utilizando la terminal de Arduino pero no resultó tan cómoda.

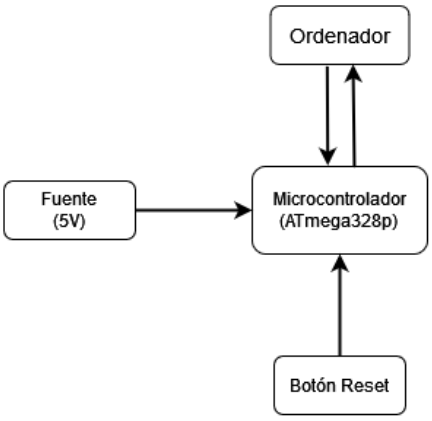

Figura 2: Diagrama en bloques del circuito A.

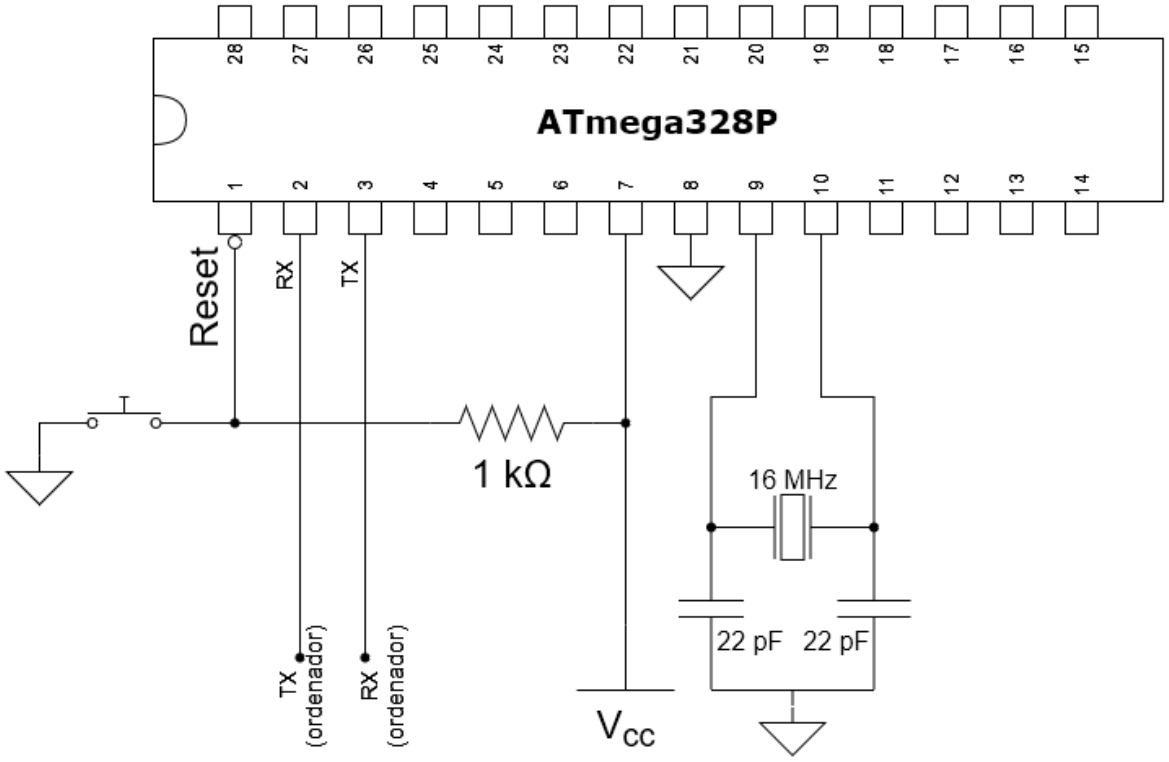

Figura 3: Diagrama esquemático del hardware de la parte A.

Parte B

La diferencia entre esta parte y la anterior es que en esta luego de enviarse el mensaje a la terminal, se está a la espera de recibir un 1, 2, 3 o 4 para encender el LED correspondiente.

Para esto se configuraron los últimos cuatro pines puerto D como salida (a estos pines se conectarán los LEDs). Una vez que se recibió el carácter, básicamente se realiza un if-else para cada LED y cambia el estado del seleccionado. En caso de recibir un carácter no definido no se hace nada.

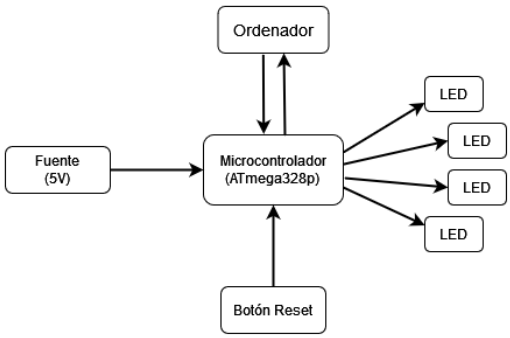

Figura 4: Diagrama en bloques del circuito B.

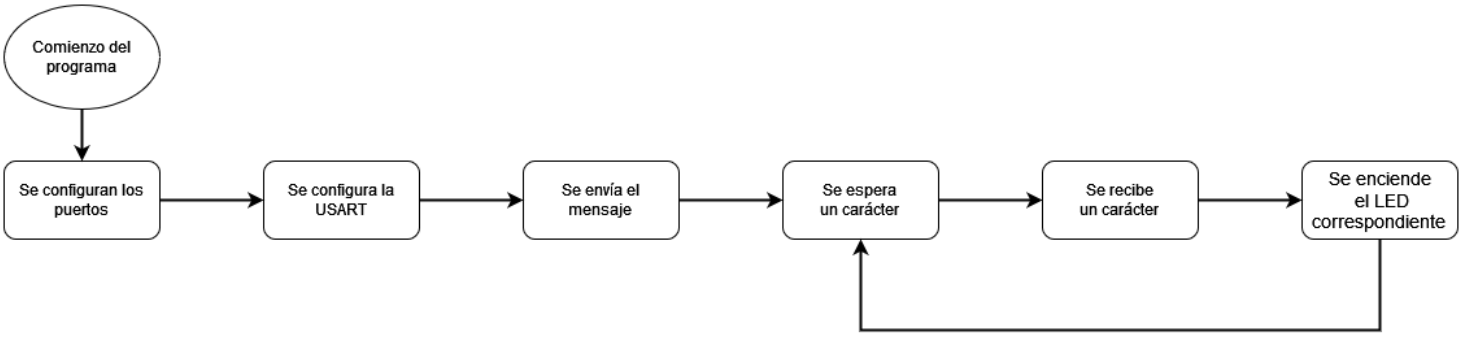

Figura 5: Diagrama del flujo del programa B.

Para el correcto funcionamiento, los dispositivos se deben conectar de la siguiente manera:

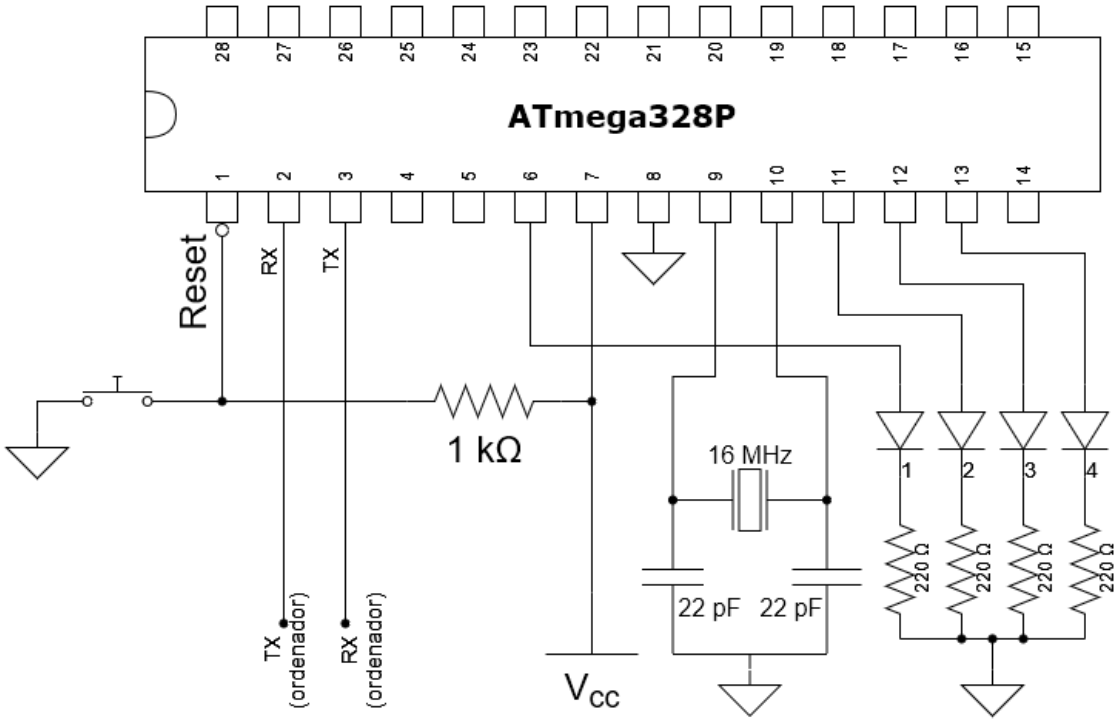

Figura 6: Diagrama esquemático del hardware.

Conclusiones

- Al momento de trabajar con comunicación asincrónica es importante que tanto receptor como emisor estén trabajando a la misma frecuencia y ambos sepan el largo de los mensajes.
- Es muy útil tener estandarizado un carácter para establecer el final de un string, como lo es el \0.
- En caso de querer subir un nuevo programa al microcontrolador no hay que estar usando ese puerto para comunicación serie.