
Universidad de Buenos Aires

Facultad de Ingeniería

2do Cuatrimestre 2020

[86.07] Laboratorio de Microprocesadores

# Trabajo Práctico Nº 2

**Alumno:** Agustín Miguel Flouret

**Padrón:** 102298

**Turno:** Martes

**Docentes:**

- Gerardo Stola
- Guido Salaya
- Fernando Cofman

**Fecha de entrega:** 3 de noviembre de 2020

# Índice

# 1. Objetivo del proyecto

En este proyecto se buscará avanzar con el manejo de puertos mediante el uso de interrupciones externas. También se analizarán las características DC del microcontrolador y la corriente entregada en este proyecto, mediante el uso de hojas de datos.

# 2. Descripción del proyecto

Para realizar el proyecto se utilizará la placa Arduino UNO, que incluye un microcontrolador ATmega328P. La programación del microcontrolador se realizará a través del Arduino, utilizando el software AVRDUDE. El lenguaje de programación que se usará es el Assembly de AVR, y se ensamblará con Atmel Studio. El hardware externo se detallará en la siguiente sección.

Las tareas a realizar son las siguientes:

- En primer lugar se realizará un programa que permita prender un LED a la vez, en un arreglo de 6 LEDs, desplazando el LED prendido de izquierda a derecha y de derecha a izquierda.
- Luego, se implementarán las dos interrupciones externas, a ser accionadas mediante pulsadores. La primera rutina de interrupción consiste en apagar los LEDs centrales y hacer parpadear los de los extremos, a una frecuencia de 1 Hz.
- La segunda interrupción hará que los LEDs reflejen el contenido de un contador binario de 0 a 63.

# 3. Listado de componentes y gastos

- 1 Arduino UNO - $1009
- 1 protoboard - $300
- 3 LED rojos - $30
- 3 LED verdes - $30
- 2 pulsadores - $54
- 6 resistores 220Ω - $42
- 2 resistores $1k\Omega$ - $14
- **Gasto total: $1479**

# 4. Circuito esquemático de Arduino UNO

En la figura 1 se muestra el circuito esquemático completo de la placa Arduino UNO que se usará en este trabajo. Para poder observar con mayor detalle las conexiones con otros componentes, en las secciones que siguen se mostrará solo la parte inferior derecha del esquemático en forma ampliada.

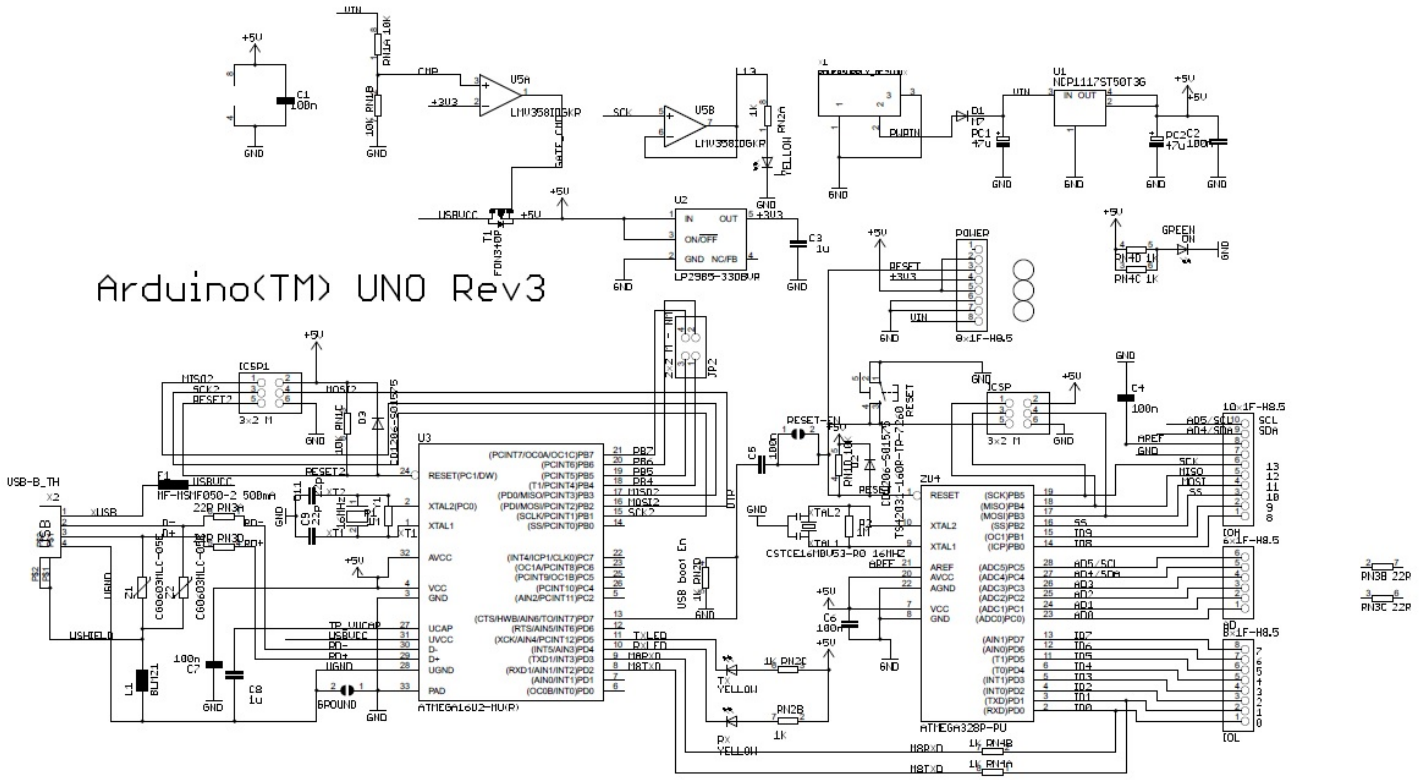


Figura 1: Circuito esquemático de la placa Arduino UNO.

# 5. Desarrollo del proyecto

Se conectaron 6 LEDs, con resistores de 220Ω en serie, a los pines del puerto C del 0 al 5. Para activar las interrupciones se conectaron dos pulsadores, con resistores de pull down externos, a los pines PD2 y PD3, correspondientes a las interrupciones externas INT0 e INT1 respectivamente. A continuación se muestra el diagrama de conexiones en bloques y el circuito esquemático del proyecto.

## 5.1. Diagrama de conexiones en bloques

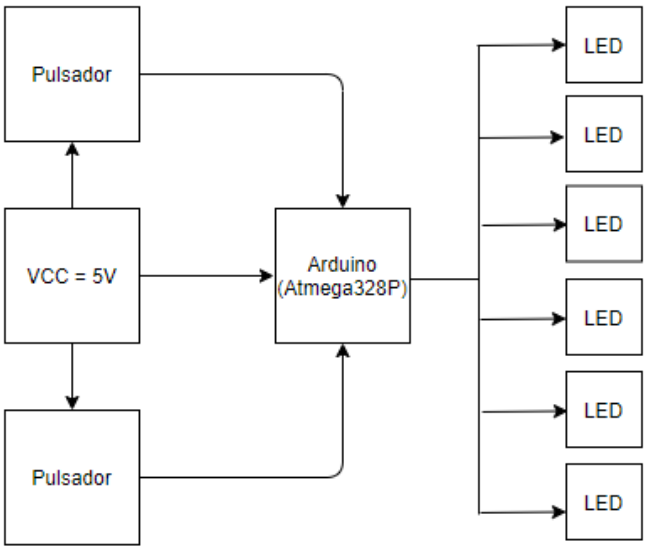


Figura 2: Diagrama de bloques del proyecto.

## 5.2. Circuito esquemático

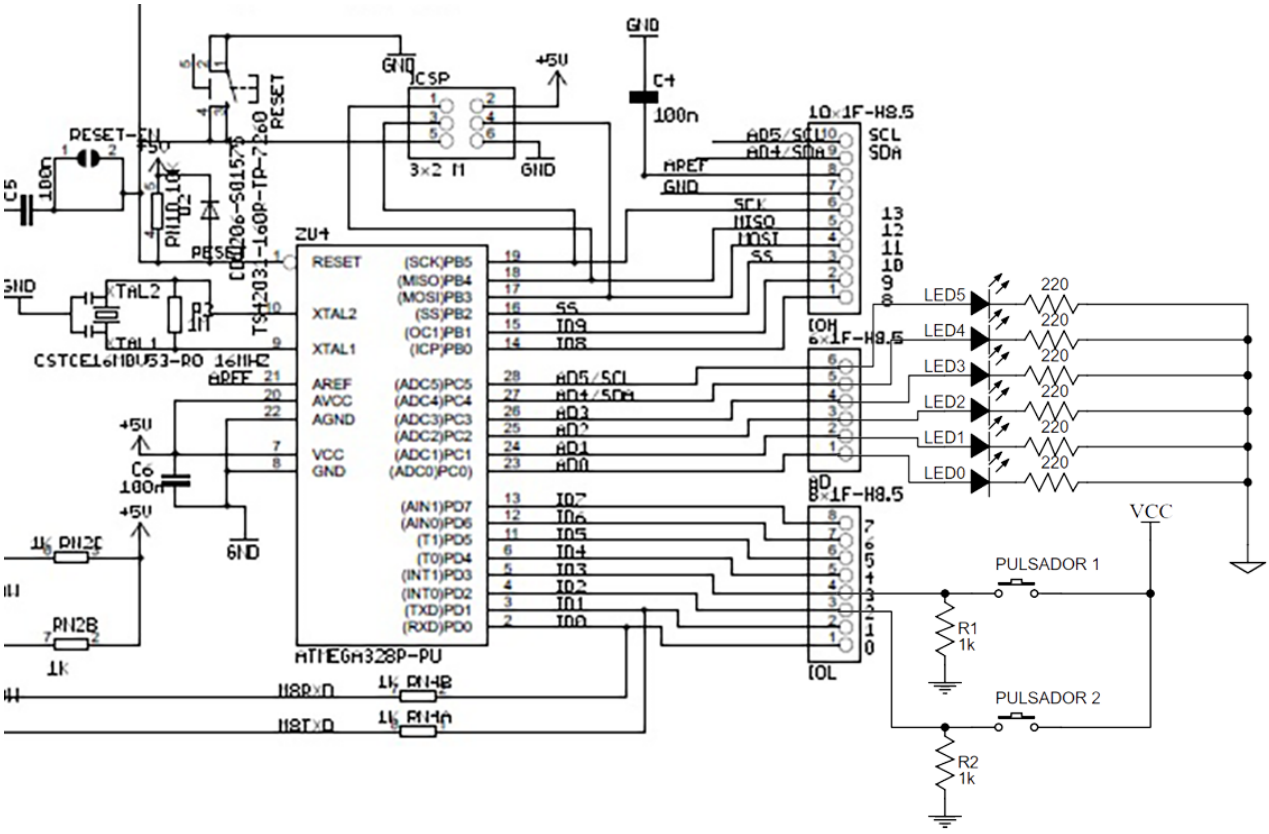


Figura 3: Circuito esquemático del proyecto.

## 5.3. Análisis de tensiones y corrientes

De la hoja de datos del microcontrolador [1] se obtuvo que la corriente de salida en estado lógico alto ($I_{OH}$) de cada pin de entrada/salida debe ser menor que 40 mA. En particular, la suma de las corrientes $I_{OH}$ de los pines del puerto C debe ser menor que 150 mA.

En cuanto a los LEDs, se obtuvo de sus hojas de datos que los valores típicos de tensión de polarización en directa son $V_F = 1{,}65V$ para el LED rojo [2] y $V_F = 2{,}1V$ para el LED verde [3]. Mediante la ecuación $I = \frac{V_{CC} - V_F}{R}$ se puede calcular la corriente que entrega cada pin para polarizar un LED, teniendo en cuenta que se utilizaron resistores de $220\Omega$ y que la máxima tensión posible que podría tener cada pin es 5V. Para el LED rojo, esto resulta $I = 15mA$, y para el LED verde $I = 13mA$. Como tenemos 3 LEDs verdes y 3 rojos, la suma de las corrientes es $I = 84mA$, lo cual es menor que el valor total máximo que puede entregar el puerto C.

## 5.4. Software

Al iniciar el programa se inicializa el stack y se configura el puerto C (pines 0 a 5) como salida, donde están conectados los 6 leds. En el registro EICRA se configuran las interrupciones. En este caso se eligió utilizar interrupciones activadas por flanco ascendente.

Al igual que en el TP1, entre cada encendido y apagado del LED se utilizó la rutina de retardo provista por la cátedra. Este retardo es de 8.000.000 ciclos, lo cual constituye, a una frecuencia de clock de 16 MHz, un retardo de medio segundo. Con este delay se obtiene la frecuencia requerida de parpadeo en la interrupción 0 (1 Hz).

En las interrupciones se guarda en el stack el estado actual del registro de estados y del puerto C y luego se desapilan, para que cuando termine de ejecutarse la rutina de interrupción se vuelva al estado anterior, tanto en los leds como en el registro de estados.

En las interrupciones también se limpia el bit correspondiente en el registro EIFR, con tal de que, si se detecta más de un flanco ascendente (por efecto rebote, por ejemplo), no se vuelva a ejecutar la misma interrupción. Si se

[1] https://ww1.microchip.com/downloads/en/DeviceDoc/Atmel-7810-Automotive-Microcontrollers-ATmega328P_Datasheet.pdf
[2] https://www.alliedelec.com/m/d/6355b8aba0b01578df0bb7b871ceefd7.pdf
[3] http://www.farnell.com/datasheets/1671521.pdf

presiona primero un pulsador y después otro, en primer lugar se ejecuta la interrupción asociada al pulsador que se presionó primero, y, al finalizar la ejecución, se ejecuta la otra. Si se presionan los dos al mismo tiempo, se ejecuta la que tiene más prioridad (INT0). Si se quisiera que una interrupción se ejecute dentro de otra, habría que setear manualmente el flag I dentro de dicha interrupción, ya que al ejecutarse una rutina de interrupción se deshabilita automáticamente este flag.

### 5.4.1. Diagrama de flujo

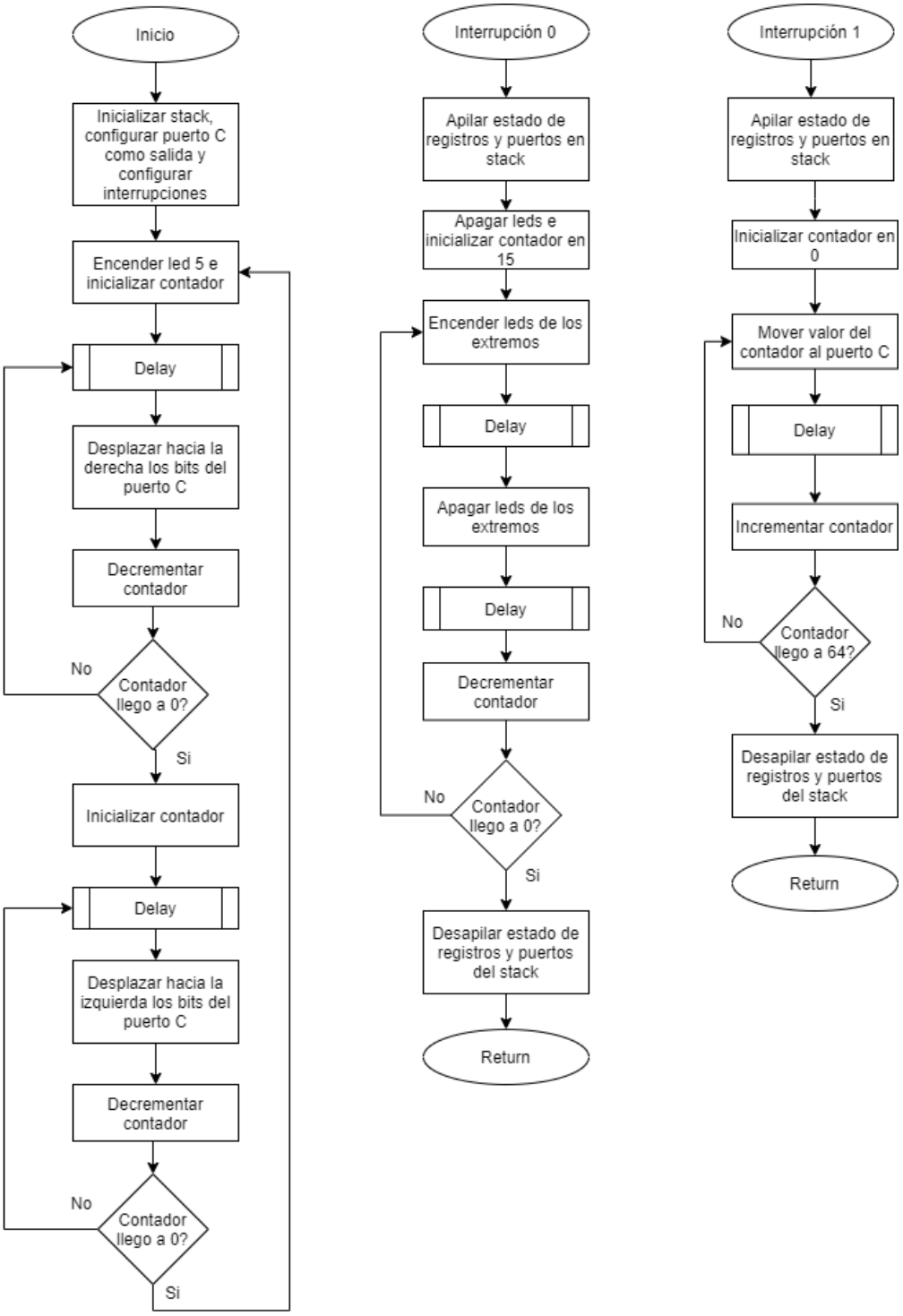


Figura 4: Diagrama de flujo del proyecto.

### 5.4.2. Código del programa

```
.include "m328pdef.inc"

.cseg
.org 0
        jmp inicio

.org 2
        jmp interrupcion0

.org 4
        jmp interrupcion1

inicio:
        ldi r16, high(RAMEND)
        out SPH, r16
        ldi r16, low(RAMEND)
        out SPL, r16        ; Inicializo stack
        ldi r16, 0x3f
        out DDRC, r16       ; Configuro los pines C0 a C5 como salidas
        ldi r16, 0x0f
        sts EICRA, r16      ; Configuro INT0 e INT1 como interrupciones por flanco ascendente
        ldi r16, 0x03
        out EIMSK, r16      ; Habilito INT0 e INT1
        sei                 ; Seteo flag I


desplazamiento_leds:
        sbi PORTC, PC5      ; Enciendo el led del pin C5
        ldi r17, 5
derecha:                ; Desplazo hacia la derecha los bits del PORTC
        call delay
        in r21, PORTC
        lsr r21
        out PORTC, r21
        dec r17
        brne derecha

        ldi r17, 5
izquierda:              ; Desplazo hacia la izquierda los bits del PORTC
        call delay
        in r21, PORTC
        lsl r21
        out PORTC, r21
        dec r17
        brne izquierda
        jmp desplazamiento_leds




delay:                  ; Delay de 8000000 ciclos (0.5s)
        ldi r18, 41
        ldi r19, 150
        ldi r20, 128
L1: dec  r20
    brne L1
    dec  r19
    brne L1
```

```
59      dec  r18
60      brne L1
61          ret
62
63
64  /*
65  Interrupcion INT0: los LEDs de los extremos parpadean
66  a una frecuencia de 1 Hz, y los centrales se apagan
67  */
68  interrupcion0:
69          push r16        ; Apilo en el stack el registro r16
70          in r16, sreg
71          push r16        ; Apilo en el stack el registro de estados
72          in r16, PORTC
73          push r16        ; Apilo en el stack el estado del puerto C
74          ldi r16, 0
75          out PORTC, r16  ; Apago todos los leds
76          ldi r16, 15
77  parpadear:
78          sbi PORTC, PC0
79          sbi PORTC, PC5  ; Enciendo los leds de los extremos
80          call delay
81          cbi PORTC, PC0
82          cbi PORTC, PC5  ; Apago los leds de los extremos
83          call delay
84          dec r16
85          brne parpadear
86          sbi EIFR, 0     ; Limpio el bit de flag de INT0
87          pop r16
88          out PORTC, r16  ; Desapilo el estado del puerto C
89          pop r16
90          out sreg, r16   ; Desapilo el registro de estados
91          pop r16         ; Desapilo el registro 16
92          reti
93
94
95  /*
96  Interrupcion INT1: los LEDs reflejan el contenido del
97  registro r16, que incrementa desde 0 hasta 63
98  */
99  interrupcion1:
100         push r16        ; Apilo en el stack el registro r16
101         in r16, sreg
102         push r16        ; Apilo en el stack el registro de estados
103         in r16, PORTC
104         push r16        ; Apilo en el stack el estado del puerto C
105         ldi r16, 0
106 contador:
107         out portc, r16  ; Actualizo el puerto C con el contenido de r16
108         call delay
109         inc r16
110         cpi r16, 64
111         brne contador
112         nop
113         sbi eifr, 1     ; Limpio el bit de flag de INT0
114         pop r16
115         out PORTC, r16  ; Desapilo el estado del puerto C
116         pop r16
117         out sreg, r16   ; Desapilo el registro de estados
118         pop r16         ; Desapilo el registro 16
```

```
119             reti
```

# 6. Resultados

Se logró obtener los resultados esperados: encender cada led, desplazando el encendido de izquierda a derecha y viceversa, y configurar correctamente las interrupciones para poder ejecutar las rutinas de parpadeo de los leds extremos y del contador.

# 7. Conclusiones

En este trabajo se logró implementar con éxito lo requerido. Se observó la ventaja de utilizar interrupciones para detectar un cambio de estado en una entrada, en lugar de realizar comparaciones constantemente. Se observó que es importante tener en cuenta el rebote del pulsador en el código del programa, ya que puede provocar comportamientos como la activación no deseada de una interrupción. También se analizaron las corrientes que entrega el microcontrolador para encender los leds. Se puede concluir que siempre hay que tener en cuenta la corriente máxima que puede entregar un puerto, para no sobrecargarlo. Por esta razón, es importante analizar las hojas de datos del microcontrolador y de los componentes que se utilizan en el circuito.