Universidad de Buenos Aires

Facultad de Ingeniería

2do Cuatrimestre 2020

[86.07] Laboratorio de Microprocesadores

Trabajo Práctico Nº 1

Alumno: Agustín Miguel Flouret

Padrón: 102298

Turno: Martes

Docentes:

- Gerardo Stola
- Guido Salaya
- Fernando Cofman

Fecha de entrega: 27 de octubre de 2020

Índice

1. Objetivo del proyecto

En este proyecto se buscará familiarizarse con un microcontrolador AVR y su programación en Assembly; en particular, la utilización de los registros de los puertos y de las resistencias *pull up* internas, con el objetivo de controlar el parpadeo de un LED.

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 haga parpadear un LED conectado en el pin PD2, utilizando la rutina de retardo.
- Luego, se modificará el programa para que el LED se prenda cuando se presiona el pulsador 1 y quede parpadeando hasta que se apague cuando se presiona el pulsador 2. El LED está conectado al pin PD2 y los pulsadores 1 y 2 a los pines PB0 y PD7 respectivamente.
- Por último, se modificará el circuito y el programa para usar la resistencia de *pull up* interna de los ports al conectar un pulsador. En este caso, no será necesario utilizar resistores de *pull down* externos como en el caso anterior.

3. Listado de componentes y gastos

- 1 Arduino UNO - $1009
- 1 protoboard - $300
- 1 LED rojo - $10
- 2 pulsadores - $54
- 1 resistor 220Ω - $7
- 2 resistores $10k\Omega$ - $14
- **Gasto total: $1394**

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.

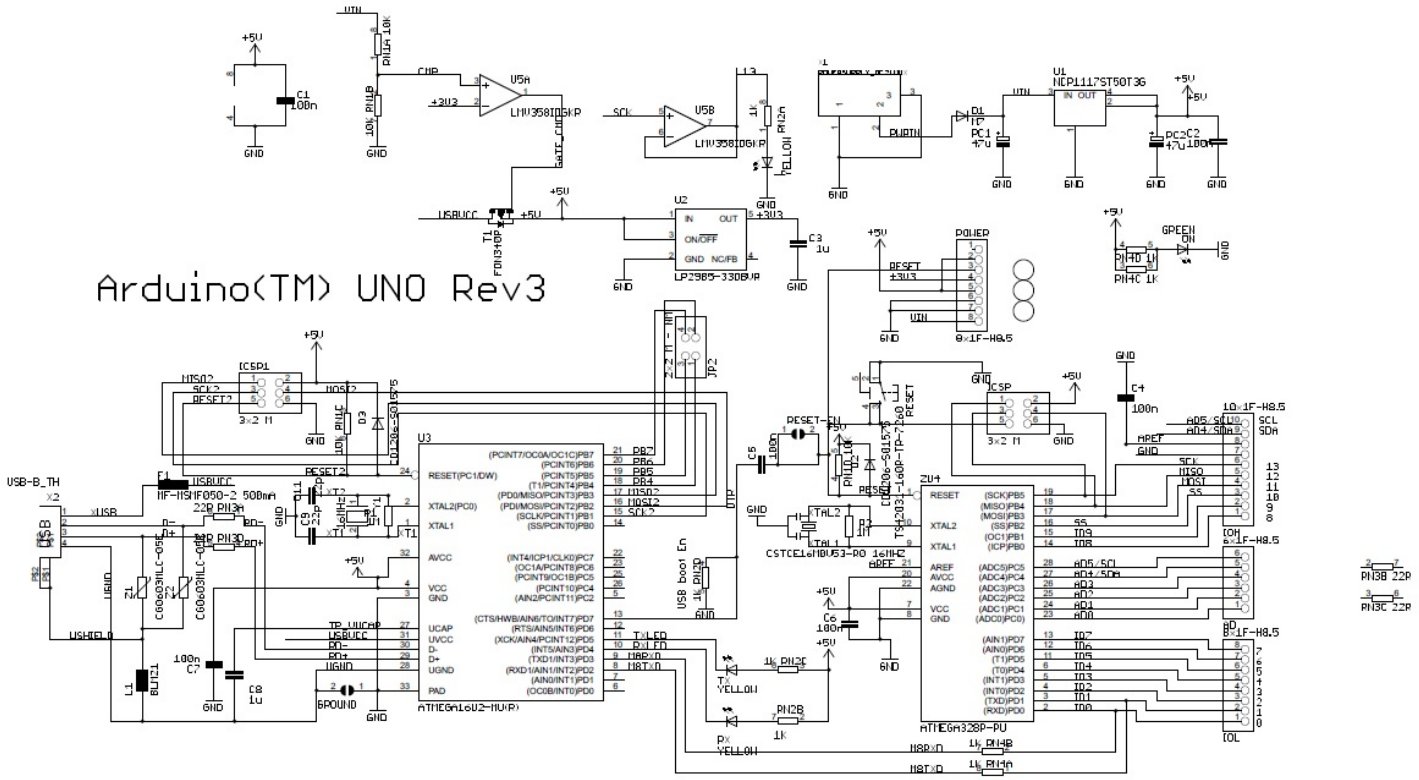

Figura 1: Circuito esquemático de la placa Arduino UNO.

5. Desarrollo del proyecto

5.1. Caso 1

En el primer caso se realizó un programa para hacer parpadear el LED. Para esto, se conectó el Arduino a una tensión de 5V mediante la entrada USB, y luego se conectó el LED al puerto PD2, correspondiente al pin 2 del Arduino. A continuación se muestra el diagrama de conexiones en bloques y el circuito esquemático de esta etapa del proyecto.

5.1.1. Diagrama de conexiones en bloques

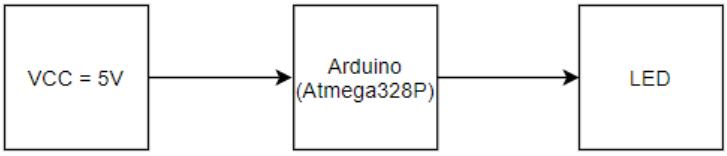

Figura 2: Diagrama de bloques del caso 1.

5.1.2. Circuito esquemático

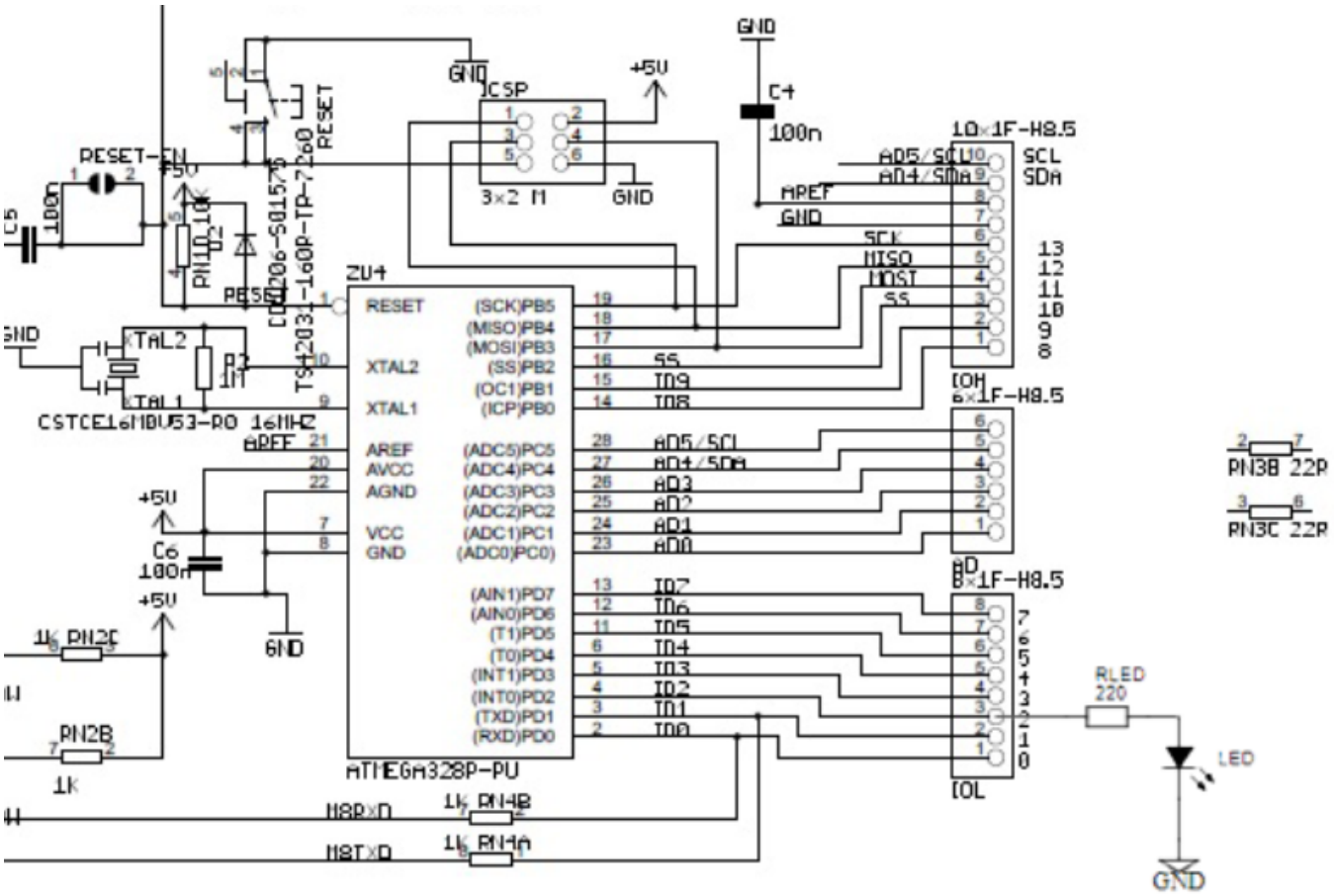

Figura 3: Circuito esquematico del caso 1.

5.1.3. Diagrama de flujo

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.

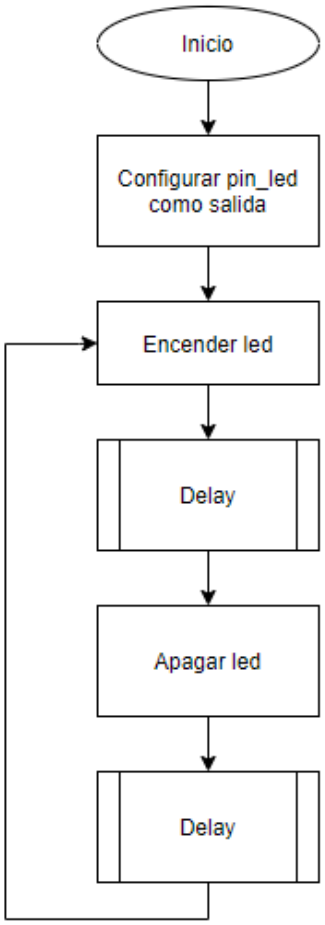

Figura 4: Diagrama de flujo del caso 1.

5.1.4. Código del programa

```
.include "m328pdef.inc"

.equ pin_led = PD2              ; Defino como constante el pin donde se conecta el led (PD2)

.cseg
.org 0


inicio:
        sbi DDRD, pin_led           ; Configuro el pin del led como salida
parpadear:
        sbi PORTD, pin_led          ; Enciendo el led
        call delay                  ; Delay 0.5s
        cbi PORTD, pin_led          ; Apago el led
        call delay                  ; Delay 0.5s
        jmp parpadear


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
        dec r18
        brne l1
        ret
```

5.2. Caso 2

En el segundo caso se armó un circuito con pulsadores, realizando las conexiones como indica el diagrama de bloques de la figura 5. Para los pulsadores se utilizaron resistores de *pull down* de $10k\Omega$. Al presionar uno de los pulsadores, el LED comienza a parpadear. Mientras está parpadeando, si se presiona el otro pulsador, el LED dejará de parpadear.

5.2.1. Diagrama de conexiones en bloques

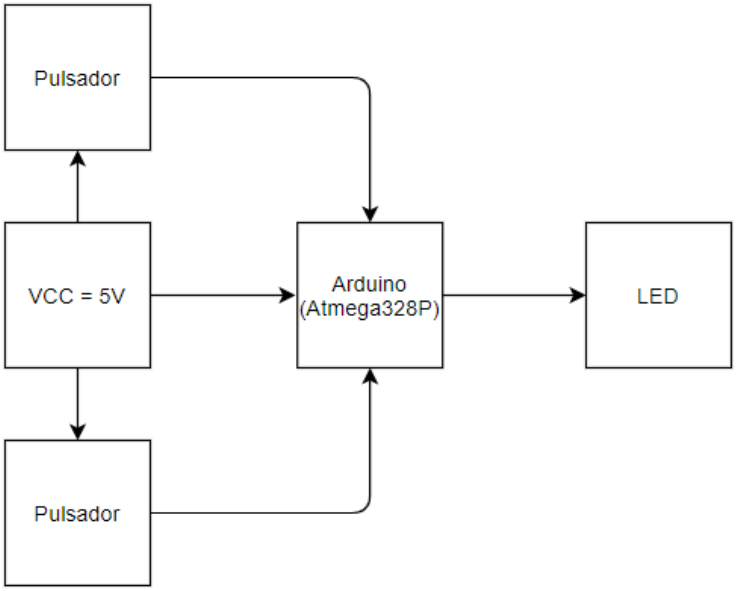

Figura 5: Diagrama de bloques del caso 2.

5.2.2. Circuito esquemático

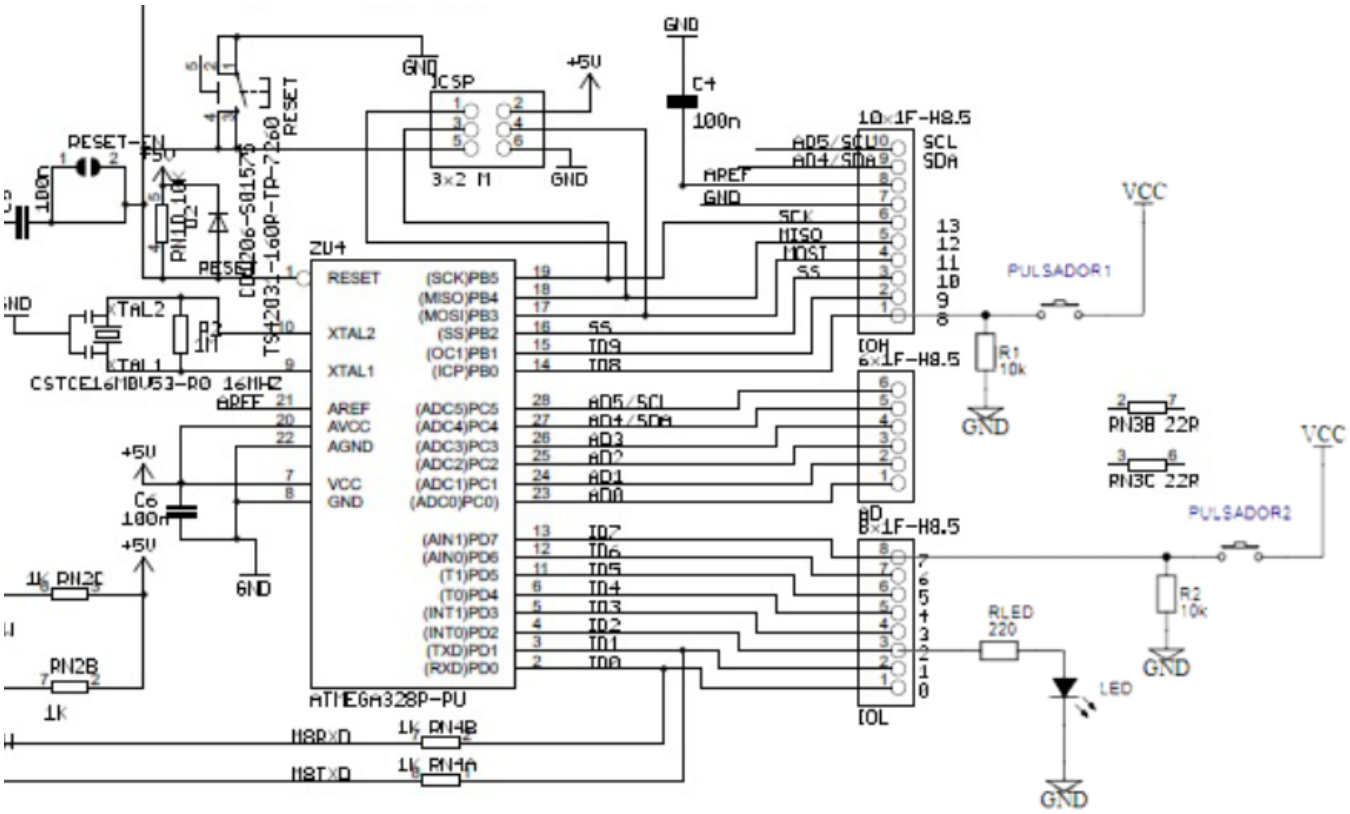

Figura 6: Circuito esquematico del caso 2.

5.2.3. Diagrama de flujo

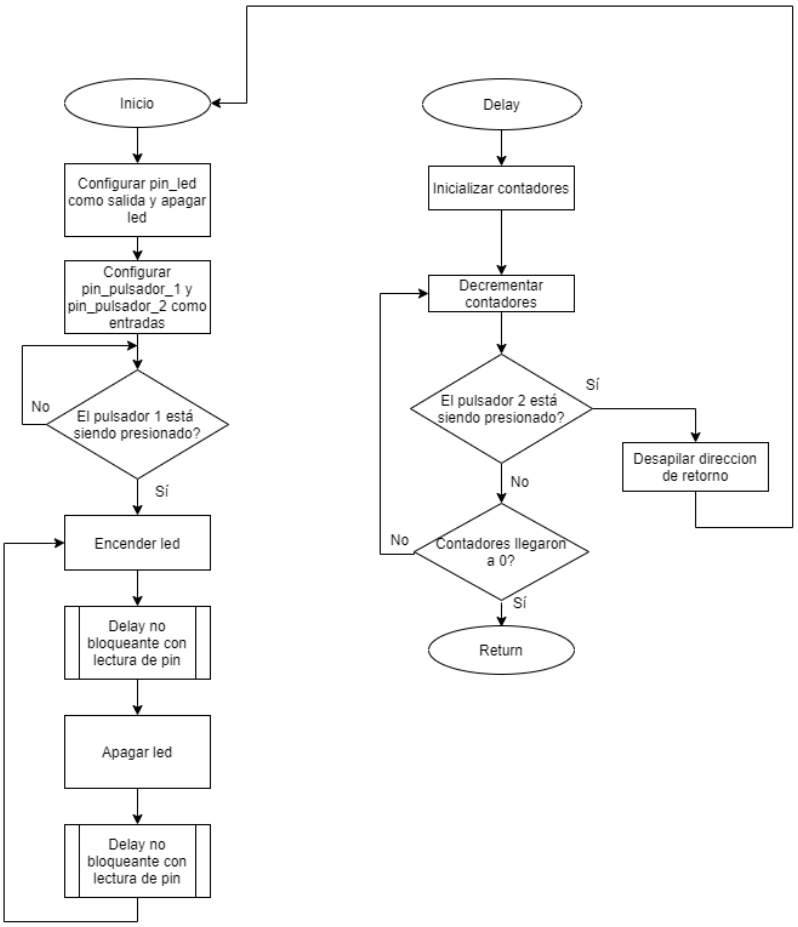

Figura 7: Diagrama de flujo del caso 2.

Como se puede ver en el diagrama de flujo y en el código del programa, dentro de la rutina de delay se chequea si el pulsador 2 está presionado. Particularmente, dentro del loop interior, correspondiente al registro R20. Esto se debe a que, durante la ejecución del programa, la mayor cantidad de tiempo transcurre dentro de este bucle, por lo que es más probable detectar un cambio de estado lógico en el pin del pulsador en esta sección del programa.

Como consecuencia de esto el tiempo de retardo es más largo, ya que se ejecutan más instrucciones en el mismo.

5.2.4. Código del programa

```
.include "m328pdef.inc"

; Defino las constantes correspondientes a los pines que voy a utilizar
.equ pin_led = PD2
.equ pin_pulsador_1 = PB0
.equ pin_pulsador_2 = PD7

.cseg
.org        0


inicio:
        cbi PORTD, pin_led              ; Inicializo el pin del led con un 0 logico
        sbi DDRD, pin_led               ; Configuro el pin del led como salida
        cbi DDRB, pin_pulsador_1        ; Configuro los pines de los pulsadores como entradas
        cbi DDRD, pin_pulsador_2
l0:        sbis PINB, pin_pulsador_1       ; Mientras el pulsador 1 no se presione, seguira en el loop
```

```
18          jmp l0
19  parpadear:
20          sbi PORTD, pin_led
21          call delay
22          cbi PORTD, pin_led
23          call delay
24          jmp parpadear
25
26
27  delay:
28          ldi r18, 41
29          ldi r19, 150
30          ldi r20, 128
31  l1: dec r20
32          sbic PIND, pin_pulsador_2                ;  Mientras el pulsador 2 no se presione, continuara e
33          jmp apagar_led
34          brne l1
35          dec r19
36          brne l1
37          dec r18
38          brne l1
39          ret
40  apagar_led:
41          pop r21
42          pop r21
43          jmp inicio
44
```

5.3. Caso 3

El tercer caso sigue la misma lógica del anterior. La diferencia está en que no es necesario utilizar resistencias externas de $10k\Omega$, ya que se habilitan las resistencias de *pull up* internas en los puertos. Entonces, los pulsadores se conectan a tierra y no a la fuente. Además, como al presionar los pulsadores se impone un cero lógico en las entradas, fue necesario intercambiar las instrucciones de los saltos condicionales: `sbic` por `sbis` y viceversa.

5.3.1. Diagrama de conexiones en bloques

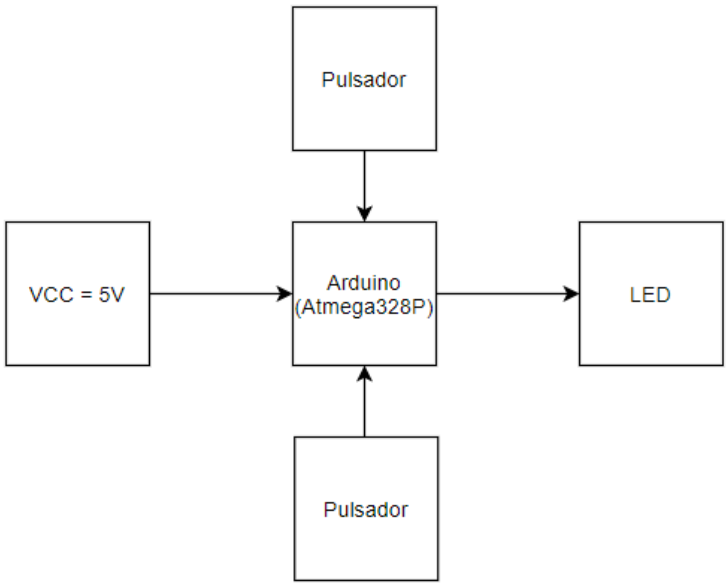

Figura 8: Diagrama de bloques del caso 3.

5.3.2. Circuito esquemático

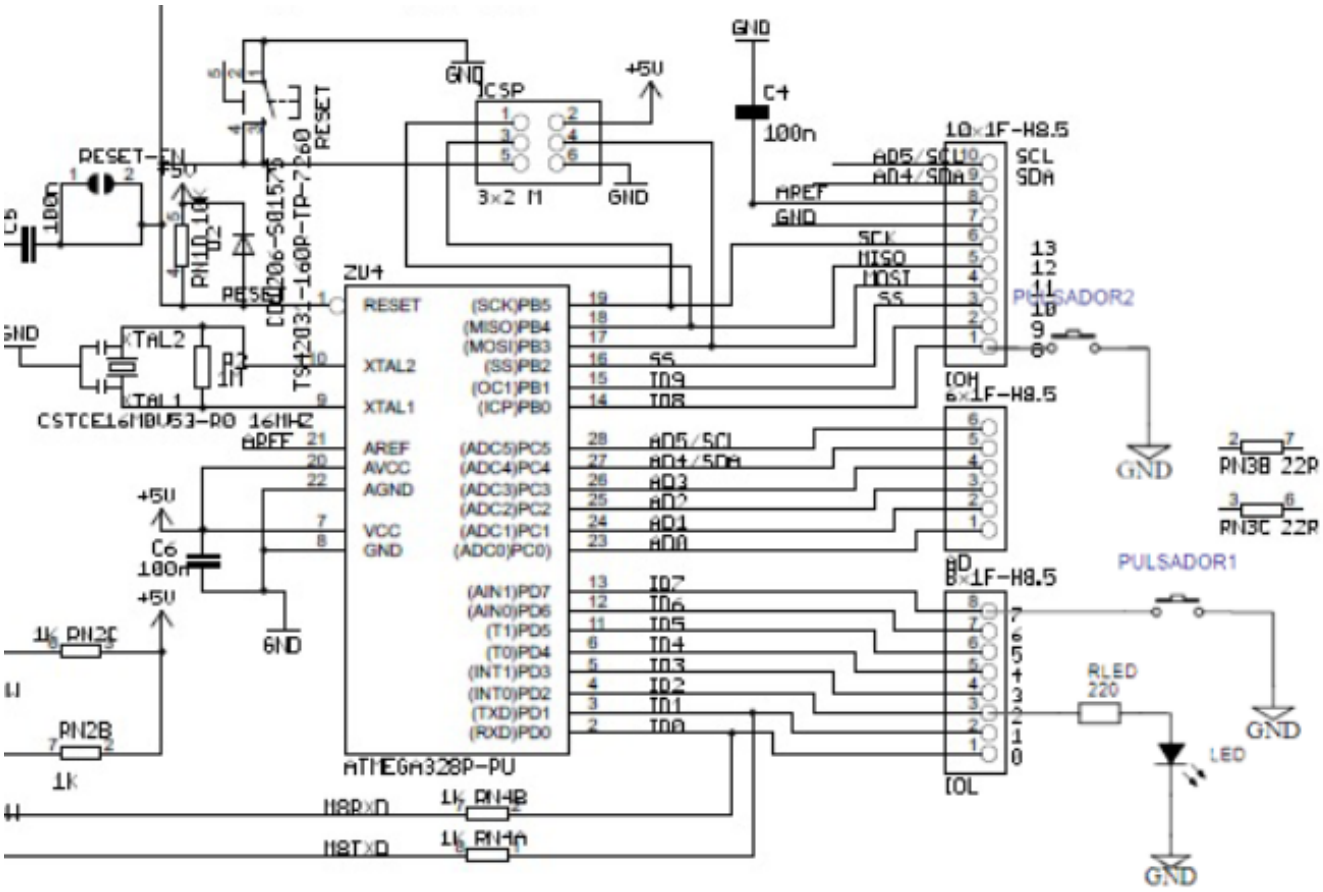

Figura 9: Circuito esquematico del caso 3.

5.3.3. Diagrama de flujo

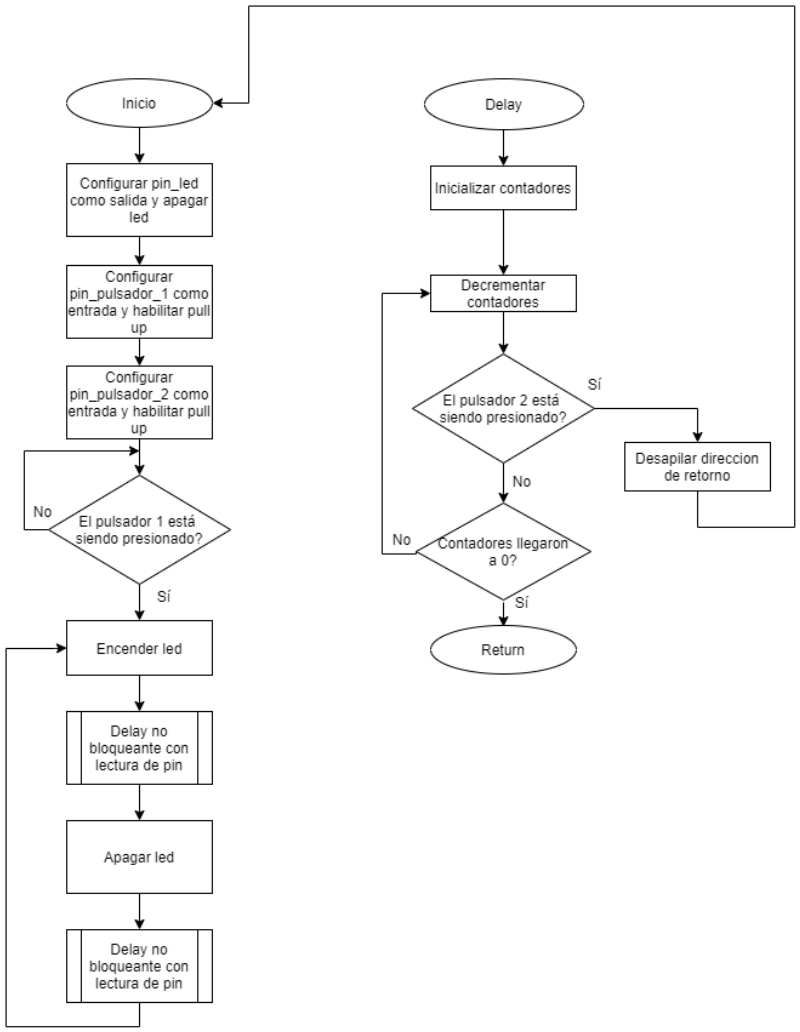

Figura 10: Diagrama de flujo del caso 3.

5.3.4. Código del programa

```
.include "m328pdef.inc"

; Defino las constantes correspondientes a los pines que voy a utilizar
.equ pin_led = PD2
.equ pin_pulsador_1 = PB0
.equ pin_pulsador_2 = PD7

.cseg
.org        0


inicio:
        cbi PORTD, pin_led          ; Inicializo el pin del led con un 0 logico
        sbi DDRD, pin_led           ; Configuro el pin como salida

        sbi PORTB, pin_pulsador_1   ; Activo resistencia de pull up interna del pin correspondiente al
        cbi DDRB, pin_pulsador_1    ; Configuro el pin como entrada

        sbi PORTD, pin_pulsador_2   ; Activo resistencia de pull up interna del pin correspondiente al
        cbi DDRD, pin_pulsador_2    ; Configuro el pin como entrada
```

```
21
22  l0:        sbic PINB, pin_pulsador_1   ; Mientras el pulsador 1 no se presione, seguira en el loop. Si
23          jmp l0
24
25  parpadear:
26          sbi PORTD, 2
27          call delay
28          cbi PORTD, 2
29          call delay
30          jmp parpadear
31
32
33  delay:
34          ldi r18, 41
35          ldi r19, 150
36          ldi r20, 128
37  l1: dec r20
38          sbis PIND, 7    ; Mientras el pulsador 2 no se presione, continuara el delay. Si se presiona, s
39          jmp apagar_led
40          brne l1
41          dec r19
42          brne l1
43          dec r18
44          brne l1
45          ret
46  apagar_led:
47          pop r21
48          pop r21
49          jmp inicio
50
51
```

6. Resultados

En todos los casos se logró obtener los resultados esperados: hacer parpadear el LED, controlar el parpadeo con los botones y hacer lo mismo utilizando las resistencias de *pull up* internas.

7. Conclusiones

En este trabajo sirvió como introducción a la programación con Assembly y al manejo de puertos del microcontrolador. Además, se pudo observar la ventaja de utilizar las resistencias de *pull up* internas: se ahorran dos resistores en el circuito, y no hace falta conectar los pulsadores a la fuente. Sin embargo, hay que tener en cuenta que los valores lógicos que se obtienen al presionar los pulsadores son distintos que cuando se usan resistencias de *pull down* . Por lo tanto, hay que realizar los cambios correspondientes en el código del programa.