



# FACULTAD DE INGENIERIA

Universidad de Buenos Aires

Laboratorio de Microprocesadores - 86.07

## Trabajo Práctico N°1: Parpadeo de un LED

|                               |       |       |                           |  |  |  |  |  |  |  |  |  |
|-------------------------------|-------|-------|---------------------------|--|--|--|--|--|--|--|--|--|
| Profesor:                     |       |       | Ing. Guillermo Campiglio  |  |  |  |  |  |  |  |  |  |
| Cuatrimestre/Año:             |       |       | 1°/2020                   |  |  |  |  |  |  |  |  |  |
| Turno de las clases prácticas |       |       | Miércoles                 |  |  |  |  |  |  |  |  |  |
| Jefe de trabajos prácticos:   |       |       | Ing. Pedro Ignacio Martos |  |  |  |  |  |  |  |  |  |
| Docente guía:                 |       |       | -                         |  |  |  |  |  |  |  |  |  |
|                               |       |       |                           |  |  |  |  |  |  |  |  |  |
| Autor                         |       |       | Seguimiento del proyecto  |  |  |  |  |  |  |  |  |  |
| Francisco                     | Rossi | 99540 |                           |  |  |  |  |  |  |  |  |  |

### Observaciones:

.....

.....

.....

.....

.....

.....

.....

.....

|                     |  |  |             |  |  |
|---------------------|--|--|-------------|--|--|
| Fecha de aprobación |  |  | Firma J.T.P |  |  |
|                     |  |  |             |  |  |

|                |  |
|----------------|--|
| Coloquio       |  |
| Nota final     |  |
| Firma profesor |  |

# Índice

|  |          |
|--|----------|
| <b>1. Introducción</b>                             | <b>2</b> |
| 1.1. Objetivo . . . . .                            | 2        |
| 1.2. Descripción . . . . .                         | 2        |
| <b>2. Materiales</b>                               | <b>2</b> |
| <b>3. Diagrama en Bloques</b>                      | <b>2</b> |
| <b>4. Esquemático</b>                              | <b>3</b> |
| <b>5. Diagrama de flujo</b>                        | <b>4</b> |
| <b>6. Códigos</b>                                  | <b>5</b> |
| 6.1. a) Utilizando todo el puerto. . . . .         | 5        |
| 6.2. b) Utilizando solo un bit del puerto. . . . . | 6        |
| 6.3. Frecuencia de oscilación del LED . . . . .    | 7        |
| <b>7. Resultados</b>                               | <b>8</b> |
| <b>8. Conclusiones</b>                             | <b>8</b> |

## 1. Introducción

En el siguiente informe se explica el diseño de dos programas escritos en lenguaje Assembler para hacer parpadear un LED conectado al pin *PB0* de un *ATMEGA328p* de dos maneras diferentes:

- Utilizando todo el puerto.
- Utilizando solo un bit del puerto.

### 1.1. Objetivo

El objetivo es controlar un pin de entrada/salida del microcontrolador.

### 1.2. Descripción

Se conecta un LED con un resistor en serie al pin *PB0*, para controlar el encendido y apagado del mismo mediante las instrucciones de Assembler correspondientes a los casos (a) y (b), en los cuales se controlara en un caso todo el puerto y en el otro un único pin respectivamente. También se realiza el ajuste de la frecuencia de oscilación a partir de pequeños cambios en el código.

## 2. Materiales

Se utilizaron los siguiente materiales para el proyecto:

- 1 LED Blanco
- 1 Resistor de  $220\Omega$
- 1 Microcontrolador ATmega328p (Utilizando el integrado con el Arduino Uno)

## 3. Diagrama en Bloques

En la **Fig. 1** se muestra un diagrama en bloques del circuito.

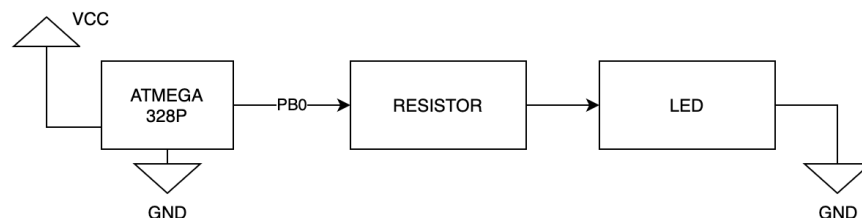


Figura 1: Diagrama en bloques.

## 4. Esquemático

En las **Fig. 3** y **Fig. 2** se muestra como se conectó el arduino con el LED y el resistor de 220  $\Omega$ .

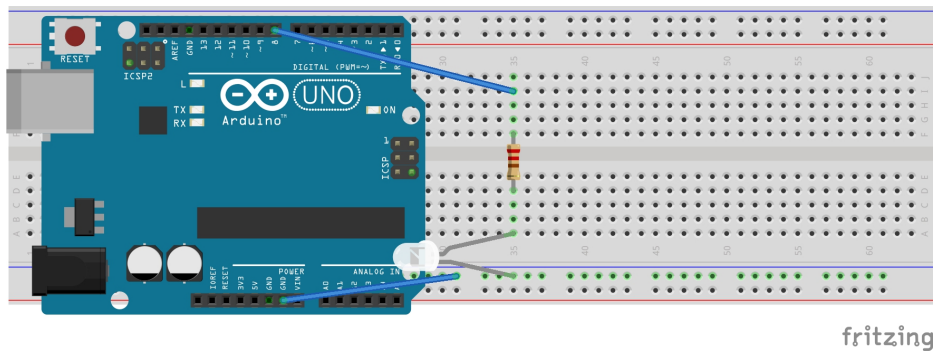


Figura 2: Conexión entre arduino y LED.

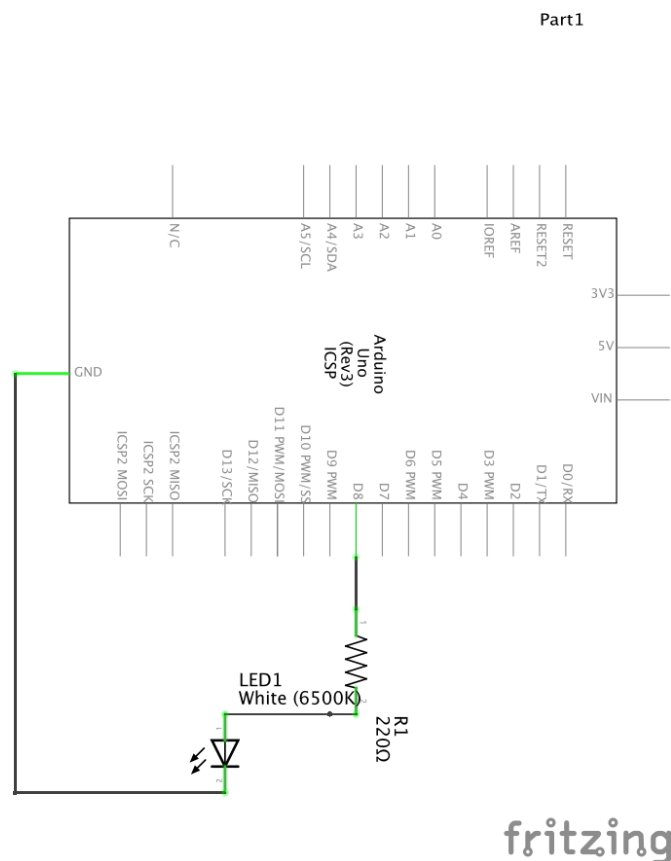


Figura 3: Esquemático del circuito implementado.

## 5. Diagrama de flujo

En la **Fig. 4** se muestra el diagrama de flujo del programa (b), el caso (a) es idéntico pero  $DDRB = 0xFF$  y LED on representa a todo el puerto B.

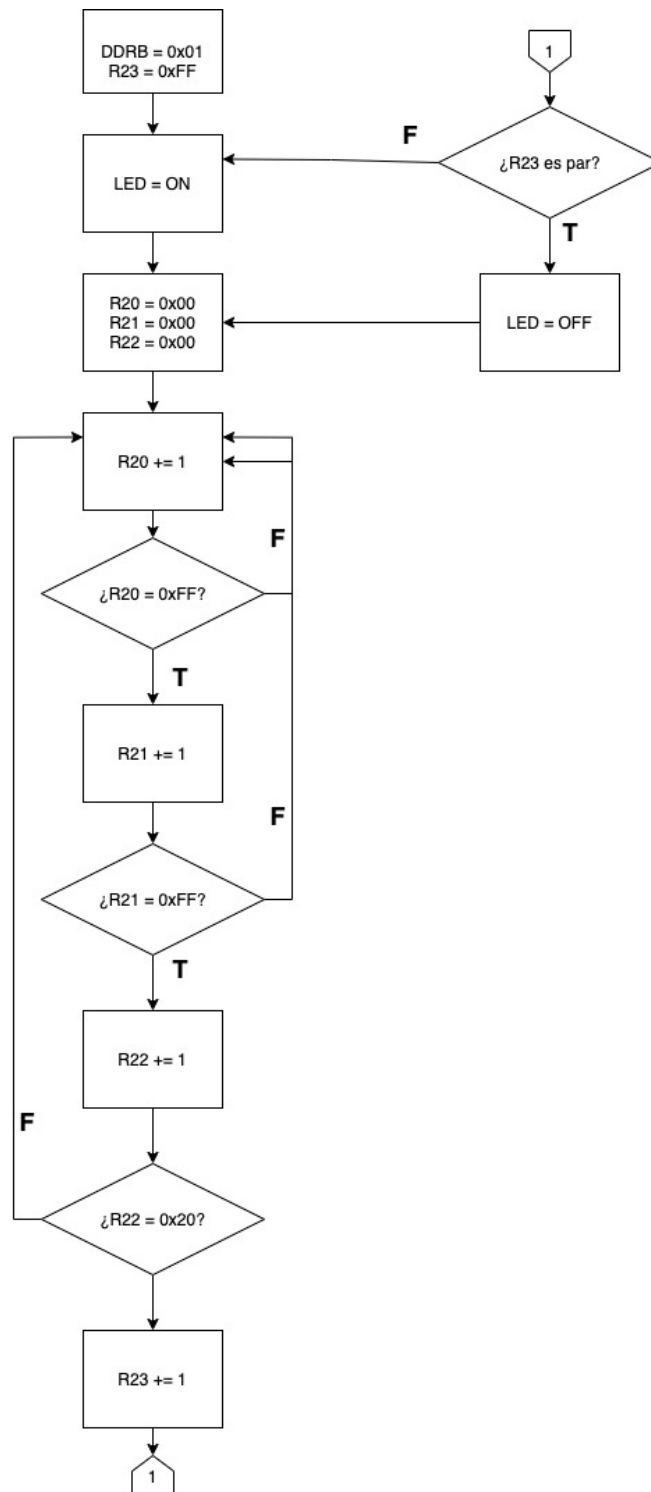


Figura 4: Diagrama de flujo.

## 6. Códigos

Ambos códigos comparten la mayoría de sus líneas, la diferencia entre ambos es que en (a) se configura todo el *PortB* como salida y se hace conmutar el mismo todo como uno, en este caso podemos conectar el LED a cualquier *PIN* del puerto y este parpadearía. En cambio, en (b) se configura únicamente el *PB0* como salida y se hace conmutar el mismo, por lo tanto, solo conmutaría este *PIN*.

### 6.1. a) Utilizando todo el puerto.

```

1  .include "m328pdef.inc"
2
3  .cseg
4  .org 0x0000
5      jmp main
6
7  .org INT.VECTORS_SIZE
8
9
10 main:
11
12     ldi    r20,0xff
13     out    DDRB,r20      ; Configuro puerto B como salida
14
15     ser    r25            ; valores 0xFF y 0x00 para luego conmutar el puerto B
16     clr    r26
17
18     ldi    r23,0xff
19 on:    out    PORTB,r25    ; enciendo TODO el PUERTO B.
20
21
22 delay:
23     ldi    r20,0x00
24     ldi    r21,0x00
25     ldi    r22,0x00
26
27 cycle: inc    r20
28     cpi    r20,0xff
29     brlo   cycle
30     ldi    r20,0x00
31     inc    r21
32     cpi    r21,0xff
33     brlo   cycle
34     ldi    r21,0x00
35     inc    r22
36     cpi    r22,0x20      ; con este valor es facil variar de forma apreciable la frecuencia
37                             ; ya que es un multiplicador de todos los incrementos anteriores
38     brlo   cycle
39     inc    r23
40
41     mov    r24, r23
42     andi   r24, 0x01      ; el resultado es 0 o 1, y define si r23 es o no par.
43     breq   off           ; Si es el valor de r23 es par entonces que apague el led,
44                             ; si no que lo prenda, así cada vez que incrementa r23 tiene
45                             ; un resultado diferente, hace 1 ciclo con LED on y otro con LED off
46     RJMP   on            ; reinicio el ciclo
47
48 off:    out    PORTB,r26    ; apago el PUERTO B entero
49     RJMP   delay

```

## 6.2. b) Utilizando solo un bit del puerto.

```

1  ..include "m328pdef.inc"
2
3  .equ lsb = 0x01
4
5  .cseg
6  .org 0x0000
7      jmp main
8
9  .org INT_VECTORS_SIZE
10
11 main:
12
13     ldi    r20,lsb
14     out    DDRB,r20    ; Configuro el PB,0 como salida. No todo el puerto.
15
16     ldi    r23,0xff    ; esto es para incializar r23 en un valor que al incrementarlo
17                        ;sea par y cumplir la condicion en el primer breq que debe ser True
18 on:     sbi    PORTB,0    ;enciendo el LED en PB0.
19
20
21 delay:
22     ldi    r20,0x00
23     ldi    r21,0x00
24     ldi    r22,0x00
25
26 cycle:  inc    r20
27     cpi    r20,0xff
28     brlo   cycle
29     ldi    r20,0x00
30     inc    r21
31     cpi    r21,0xff
32     brlo   cycle
33     ldi    r21,0x00
34     inc    r22
35     cpi    r22,0x20    ; con este valor es facil variar de forma apreciable la frecuencia
36                        ;ya que es un multiplicador de todos los incrementos anteriores
37     brlo   cycle
38     inc    r23
39
40     mov    r24, r23
41     andi   r24, 0x01 ; el resultado es 0 o 1, y define si r23 es o no par.
42     breq   off        ; Si es el valor de r23 es par entonces que apague el led,
43                        ;si no que lo prenda, asi cada vez que incrementa r23 tiene
44                        ;un resultado diferente, hace 1 ciclo con LED on y otro con LED off
45     RJMP   on          ; reinicio el ciclo
46
47 off:     cbi    PORTB,0    ;apago el LED
48     RJMP   delay        ; reinicio el ciclo
    
```

El ciclo para realizar los retardos esta basado en incrementos de registros, dependiendo del valor de R23 se determina si el LED se debe apagar o prender. Y se repite el ciclo infinitamente. En la siguiente sección se explicará en profundidad la relación entre el incremento de registro y los tiempos de conmutación del PIN o el Puerto. El código fue pensado para lograr un tiempo en alto y en bajo similares es decir, que el *Duty Cycle* sea lo más cercano posible al 50 %.

### 6.3. Frecuencia de oscilación del LED

En ambos casos, queremos hacer que la oscilación sea apreciable a simple vista, es decir setear el tiempo de encendido y apagado del LED, lo que es lo mismo que el tiempo de encendido y apagado del *PB0*.

La frecuencia por default del clock del *Arduino UNO* es  $f_{ck} = 16 \text{ MHz}$ , de manera que sabiendo los ciclos de maquina que hay entre encendido y apagado  $ciclos_{on}$ ,  $ciclos_{off}$  podemos obtener los tiempos de encendido y apagado como  $t_{on} = \frac{ciclos_{on}}{f_{ck}}$ ,  $t_{off} = \frac{ciclos_{off}}{f_{ck}}$  y con esto podemos obtener la frecuencia de oscilación del LED como  $f = \frac{1}{t_{on} + t_{off}}$ .

Luego, para calcular la cantidad de ciclos en alto y en bajo debemos analizar como vamos a realizar el ciclo en el código, en este caso se realizó incrementando registro de manera que cada registro tiene un *peso*, al igual que el sistema numérico de manera que cuando R20 sea incrementado 255 veces el registro R21 se incrementara una vez y se volverá a repetir el incremento entre 0 y 255 del valor de R20, lo mismo sucederá con R22 y R21 hasta el valor máximo de R22, en este caso se eligió  $R22_{MAX} = 0x20 \underbrace{=}_{\text{a decimal}} 32$ .

De manera que la cantidad de ciclos totales se podrá calcular como la cantidad de ciclos que se toma en incrementar el registro R20 ( $CC_{R20}$ ) sumado a la cantidad de ciclos que se toma en incrementar el registro R21 ( $CC_{R21}$ ) sumado a la cantidad de ciclos que se toma en incrementar el registro R22 ( $CC_{R22}$ ) sumado a los ciclos extras.

Teniendo en cuenta la **Tabla. 1** con la cantidad de ciclos que toma cada instrucción.

Tabla 1: Ciclos de maquina por instrucción

| Instrucción | Ciclos |
|-------------|--------|
| SBI         | 2      |
| CBI         | 2      |
| LDI         | 1      |
| INC         | 1      |
| CPI         | 1      |
| BRLO        | 1      |
| MOV         | 1      |
| ANDI        | 1      |
| BREQ_FALSE  | 1      |
| BREQ_TRUE   | 2      |
| BRLO_FALSE  | 1      |
| BRLO_TRUE   | 2      |
| RJMP        | 2      |

Podemos representar lo dicho anteriormente de manera matemática como:

$$CC_{R20} = \left[ 255 \cdot [cy(INC) + cy(CPI) + cy(BRLO_T)] - cy(BRLO_T) + cy(BRLO_F) \right] \cdot 255 \cdot R22_{MAX}$$

$$CC_{R21} = \left[ 255 \cdot [cy(INC) + cy(CPI) + cy(BRLO_T) + cy(LDI)] - cy(BRLO_T) + cy(BRLO_F) \right] \cdot R22_{MAX}$$

$$CC_{R22} = [R22_{MAX} \cdot (cy(INC) + cy(CPI) + cy(BRLO_T))] - cy(BRLO_T) + cy(BRLO_F)$$



Obteniendo luego la cantidad de ciclos en los cuales el LED estará encendido y apagado, ordenados por el orden de aparición:

$$cy_{on} = 3cy(LDI) + CC_{R21} + CC_{R22} + cy(MOV) + cy(ANDI) + cy(BREQ_T) + cy(CBI) + CC_{R20} = 8355976$$

$$cy_{off} = 3cy(LDI) + CC_{R20} + CC_{R21} + CC_{R22} + cy(MOV) + cy(ANDI) + cy(BREQ_F) + cy(RJMP) + cy(SBI) = 8355977$$

Luego utilizando la expresión  $t_x = \frac{ciclos_x}{f - ck}$  se obtienen los valores de  $t_{on}$  y  $t_{off}$ .

Los valores obtenidos se muestran en la **Tabla. 2**.

Tabla 2: Tiempos de encendido y apagado.

|     | Ciclos  | Tiempo (s) |
|-----|---------|------------|
| ON  | 8355976 | 0,52225    |
| OFF | 8355977 | 0,52225    |

Con estos resultados obtenemos la frecuencia de oscilación del LED y el *Duty Cycle*:

$$Duty\ Cycle = 50\ \%$$

$$f = 0,95740\ Hz$$

## 7. Resultados

Se logró diseñar dos programas para el microcontrolador ATMEGA328p que hagan oscilar un LED colocado en el puerto B, con *Duty Cycle*  $\approx 50\ \%$  y con frecuencia adaptable cambiando el valor de  $R22_{MAX}$  de dos maneras diferentes, una utilizando todo el puerto y otra un único PIN.

## 8. Conclusiones

Se logró controlar el pin de entrada y salida para hacer titilar el LED tanto individualmente (b), así como todo el puerto junto (b), podemos encontrar la posible utilidad de los dos casos, por ejemplo en el caso (a) podemos sincronizar todo el puerto para que conmute al mismo tiempo en caso, por ejemplo, de querer conmutar múltiples LEDs de manera sincronizada sin que la corriente máxima que puede ser obtenida del pin sea alcanzada. En el otro caso es de utilidad cuando en los otros pines hay otros periféricos conectados y no deseamos alterar sus valores.