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

# Laboratorio de Microprocesadores (86.07)

TRABAJO PRÁCTICO OBLIGATORIO 6

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

# Índice

# 1. Objetivos

El trabajo práctico consiste en generar un programa que sea capaz de hacer parpadear un LED con diferentes tipos de retardo generados por el timer numero 1 interno. 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 MASK_CS =  0b11111000

.EQU POS_BTN1 = 2
.EQU POS_BTN2 = 3
.EQU POS_LED = 3

.ORG 0
        RJMP MAIN

.ORG OVF1addr
        RJMP OVF_ISR

MAIN:
        LDI R16,LOW(RAMEND)             ;Inicializacion del stack
        OUT SPL,R16
        LDI R16, HIGH(RAMEND)
        OUT SPH, R16

        CALL SET_PORTS                  ; Configuro puertos
        CALL SET_TIMER1                 ; Configuro el timer

LOOP:
        SBIS PIND,POS_BTN1                      ; Si BTN1 esta en 0 saltea lo siguiente
        RJMP BTN1_LOW
        SBIC PIND,POS_BTN2
        RJMP BLINK_1024         ; BTN2 = 1
        RJMP BLINK_256          ; BTN2 = 0

BTN1_LOW:
        SBIC PIND,POS_BTN2      ;  Si BTN2 = 0 saltea lo siguiente
        RJMP BLINK_64           ;  BTN2 = 1
        RJMP BLINK_FIJO         ;  BTN2 = 0

BLINK_FIJO:     ; BTN1 = 0 && BTN2 = 0
        LDS R16,TCCR1B
        ANDI R16, MASK_CS
        ORI R16,(0<<CS12)|(0<<CS11)|(0<<CS10)
        STS TCCR1B,R16                          ;Timer apagado
        SBI PORTB,POS_LED
        RJMP LOOP
```

```
BLINK_64:          ; BTN1 = 0 && BTN2 = 1
        CALL ANTIBOUNCE_BTN2
        LDS R16, TCCR1B
        ANDI R16, MASK_CS
        ORI R16, (0<<CS12)|(1<<CS11)|(1<<CS10); clk/64
        STS TCCR1B, R16          ;CS12:CS10 =>011. WGM13:WGM10 => 0000 (Modo normal)
        RJMP LOOP

BLINK_256:         ; BTN1 = 1 && BTN2 = 0
        CALL ANTIBOUNCE_BTN1
        LDS R16,TCCR1B
        ANDI R16,MASK_CS
        ORI R16, (1<<CS12)|(0<<CS11)|(0<<CS10) ; clk/256
        STS TCCR1B,R16           ;CS12:CS10 =>100. WGM13:WGM10 => 0000 (Modo normal)
        RJMP LOOP

BLINK_1024:        ; BTN1 = 1 && BTN2 = 1
        CALL ANTIBOUNCE_BTN1
        CALL ANTIBOUNCE_BTN2

        LDS R16,TCCR1B
        ANDI R16, MASK_CS
        ORI R16, (1<<CS12)|(0<<CS11)|(1<<CS10) ; clk/1024
        STS TCCR1B,R16           ;CS12:CS10 =>101. WGM13:WGM10 => 0000 (Modo normal)
        RJMP LOOP

SET_PORTS:
        SBI DDRB,POS_LED                    ;PB3 como salida
        CBI DDRD,POS_BTN1
        CBI DDRD,POS_BTN2                   ;PD2:PD3 como entrada
        RET

SET_TIMER1:
        LDI R16,0
        STS TCCR1A,R16
        STS TCCR1B,R16           ;WGM13:WGM10 => 0000 (Modo normal)
        LDI R16,(1<<TOIE1 )
        STS TIMSK1,R16           ;Se habilita la interrupci[U+FFFD]n por overflow del timer1
        SEI                                 ;Se habilita la interrupci[U+FFFD]n global
        RET

ANTIBOUNCE_BTN1:
        CALL ANTIBOUNCE
        SBIS PIND,POS_BTN1
        RJMP LOOP
        RET

ANTIBOUNCE_BTN2:
        CALL ANTIBOUNCE
        SBIS PIND,POS_BTN2
        RJMP LOOP
        RET

OVF_ISR:                            ;ISR para cambiar estado del LED
        CALL SET_TIMER1
        SBIS PORTB, POS_LED             ;Si el led esta prendido, se apaga
        RJMP TURN_ON                ;Si el led esta apagado, se prende
        RJMP TURN_OFF
```

```
TURN_ON:
        SBI PORTB, POS_LED
        RETI
TURN_OFF:
        CBI PORTB,POS_LED
        RETI

ANTIBOUNCE:
; Delay 80 000 cycles
; 5ms at 16.0 MHz
        LDI  R20, 80
L1:     LDI  R21, 4
L2:     LDI  R22, 250
L3:     DEC  R22
        BRNE L3          ; Repetir 250 veces
        DEC  R21
        BRNE L2          ; Repetir L3 unas 4 veces(Ciclos = 250*4=1000)
            DEC  R20
        BRNE L1          ; Repetir L2 unas 80 veces(Ciclos = 1000*80=80000)
            RET
```

Se utilizo la instrucción *.INCLUDE* para identificar el tipo de microcontrolador utilizado. Luego con la instrucción *.EQU* se definieron las constantes que van a ser utilizadas en el código. Estas son la mascara con la que se setean luego los bit del timer, y las posiciones de los puertos utilizadas.

La primera tarea del *Main* es inicializar el Stack. Se configuran los puertos con la rutina $SET\_PORTS$. El puerto B como salida, en donde se conecta un LED en el bit 3, y el puerto D como entrada donde se conectaron pulsadores en los bits 2 y 3.
Se configura el timer 1 del microcontrolador con la rutina $SET\_TIMER1$. Se ingresa un 0x00 en los registros TCCR1A y TCCR1B para lograr que los bits WGM13:WGM10 configuran al timer en modo normal de operación. Con este modo el timer incrementa su valor con cada ciclo de reloj contando desde 0x0000 hasta 0xFFFF (el timer 1 es de 16 bits) usando los registras TCNTL y TCNTH. Al llegar al valor máximo se produce un overflow que activa la interrupción. Esta interrupción se habilita con el bit TOIE1 en el registro TIMSK. Por ultimo se habilitan las interrupciones de forma global con la instrucción *SEI*.
Se ingresa a un LOOP cuya función es verificar el estado de los pulsadores. En base a estos estados define el tipo de retardo utilizado para generar el parpadeo del LED. Cada retardo esta determinado por la proporción de escala de clock. Esta se determina según el valor de los bits CS12:CS10 en el registro TCCR1B. Por lo tanto, cada subrutina tendrá una frecuencia distinta según la escala adoptada y se vera afectado el tiempo en el cual se produce el overflow para cada caso. Dado que se habilita la interrupción por overflow, podemos verificar que el tiempo transcurrido para que esto suceda como el cociente entre que el valor inicial y final del contador y la frecuencia del clock en cada caso.

| Pulsador 1 | Pulsador 2 | Escala | Clock | Periodo | Tiempo transcurrido |
|---|---|---|---|---|---|
| 0 | 0 | - | 16 MHz | 62.5us | 4.096ms |
| 0 | 1 | 64 | 250 kHz | 4us | 262ms |
| 1 | 0 | 256 | 62.5 kHz | 16us | 1,05s |
| 1 | 1 | 1024 | 15,63 kHz | 64us | 4,20s |

Tabla 1: Frecuencia de clock para cada estado de pulsador.

Luego se desarrolla la rutina de interrupción $OVF\_ISR$ que se encarga de cambiar el estado del LED.
Es importante destacar que para cada estado alto que se detectaba se procedió a realizar una rutina $ANTIBOUNCE$ para la verificación del estado. Para esto se produce un retardo, y luego la confirmación del estado del pulsador en cuestión. El objetivo es evitar una falla en la lectura del pulsador por el efecto rebote que pudiera causar el pulsador. Para este caso se utilizo un retardo menor de $5ms$.

$$\frac{ciclos}{frecuecia} = \frac{80000}{16MHz} = \frac{250 * 4 * 80}{16MHz} = 5ms \tag{1}$$

### 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.

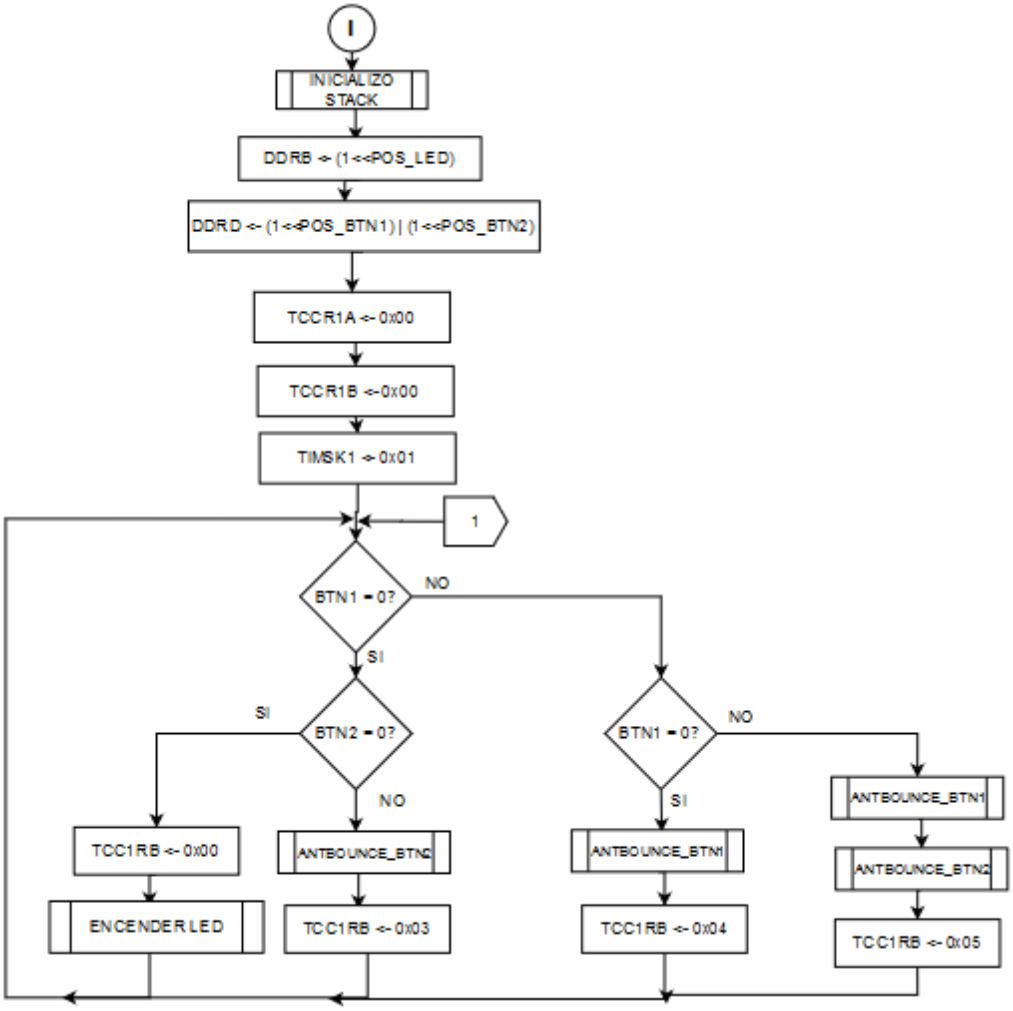


Figura 1: Diagrama de flujo del loop principal.

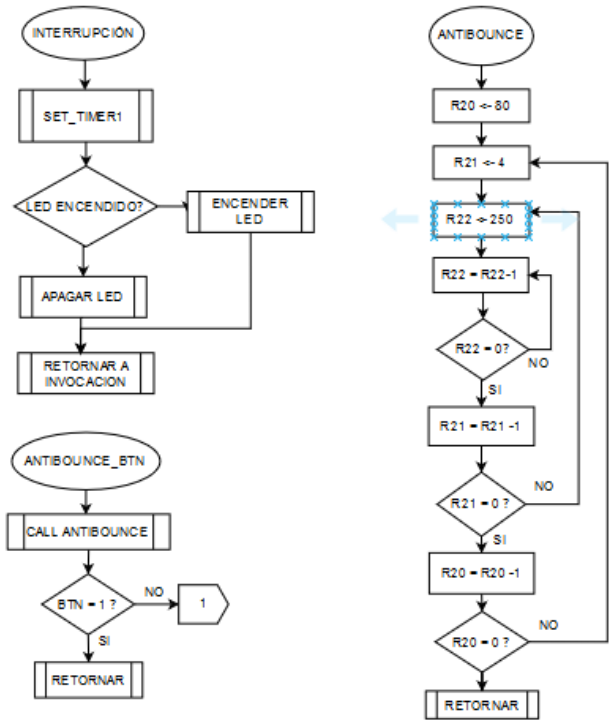


Figura 2: Diagrama de flujo de funciones.

## 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, 2 diodos led, un pulsador, una resistencia de $10K\Omega$ y 2 resistencias de $220\Omega$. Para la alimentación y trasmisión del código se utilizo el puerto USB que viene con 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.

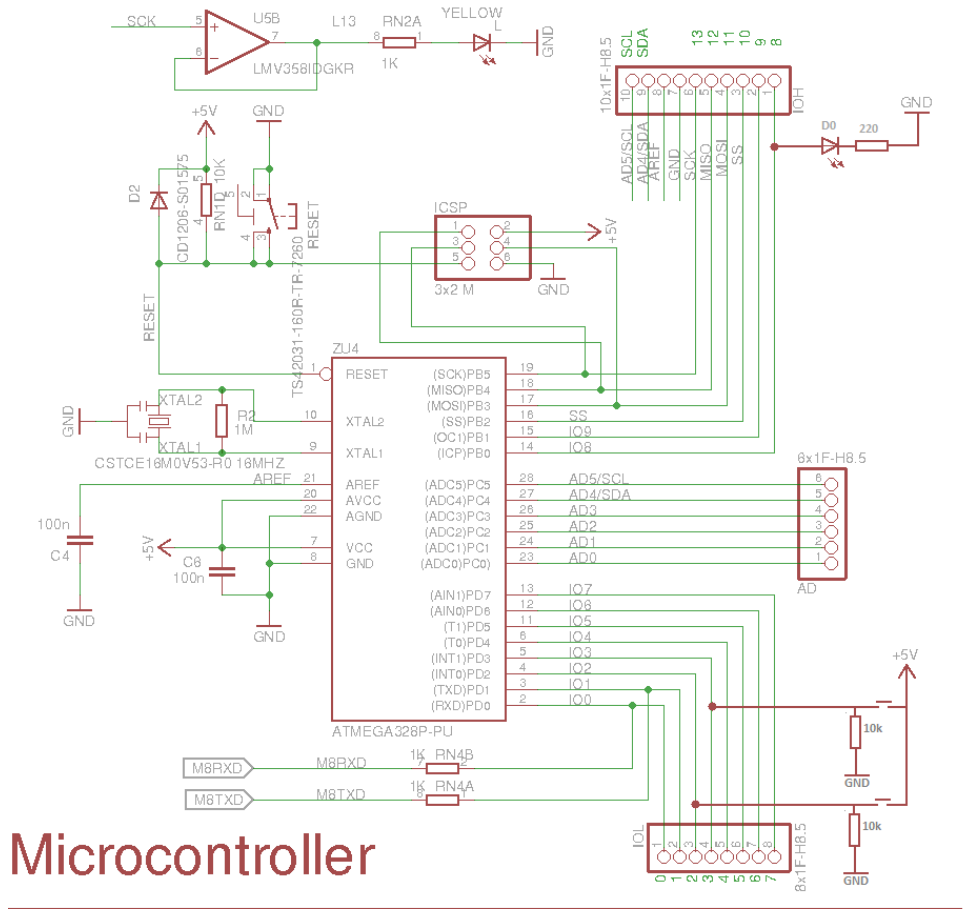


Figura 3: Diagrama de conexión.

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.

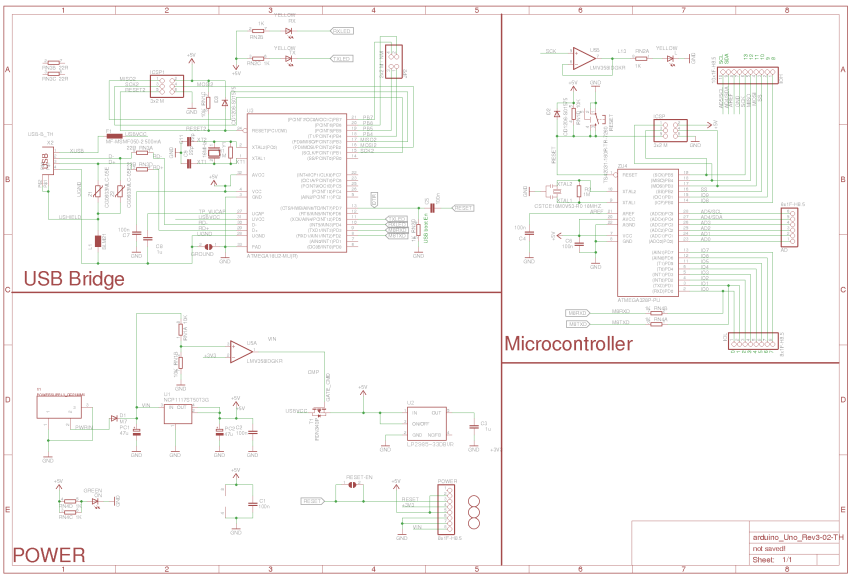


Figura 4: Diagrama de conexión de la placa.

### 2.2.1. Listado de componentes

Para este trabajo practico fueron utilizados los siguientes elementos. Se estima un coste total de 1300$.

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

Tabla 2: Precios de componentes.

# 3. Conclusiones

Para este trabajo practico se introdujo el uso de timers como una herramienta útil para generar diferentes tipos de retardos variando la frecuencia de clock usada por el microcontrolador. Además, se utilizo una rutina de pooling para chequear la variación de estados de los pulsadores y una interrupción causada por el overflow causado en cada timer configurados en modo normal. Este control constante en los estados del pulsador causaron una modificación en el parpadeo del LED pudiendo comprobar la implementación física del programa.

# 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