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

Laboratorio de Microprocesadores (86.07)

TRABAJO PRÁCTICO OBLIGATORIO 7

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

Índice

1. Objetivos

El trabajo práctico consiste en generar un programa capaz de variar la intensidad de luz de un LED por modulación del ancho de pulso utilizando el Timer 0 bajo su funcionamiento en modo PWM. 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:

```
.device ATmega328P
.cseg
.org 0x0000

.EQU PIN_BTN1=2
.EQU PIN_BTN2=3
.EQU PIN_LED=6

.EQU PASO = 10

.DEF TEMP = R16
.DEF COUNTER = R17

main:

        LDI R16,HIGH(RAMEND)
        STS SPH,R16
        LDI R16, LOW(RAMEND)
        STS SPL, R16

        /*Configuracion de puertos*/
    CBI DDRD, PIN_BTN1
        CBI DDRD, PIN_BTN2
        SBI DDRD, PIN_LED

        /*Configuro el timer0*/
        LDI R16,(1<<WGM01)|(1<<WGM00)|(1<<COM0A1) ;Non-inverted COM0A && WGM2:0=011= Mode 3
        OUT TCCR0A,R16
        LDI R17,(1<<CS00);        ;Fast PWM, modo 3. No prescaling
        OUT TCCR0B,R17

        /*Leo el valor de OCR0A*/
        IN COUNTER, OCR0A
        LDI TEMP, PASO

LOOP:
    SBIC PIND, PIN_BTN1
    RJMP INC_COUNTER
        SBIC PIND, PIN_BTN2
        RJMP DEC_COUNTER
        RJMP LOOP
```

```
INC_COUNTER:
        CALL ANTIBOUNCE_BTN1
        ADD COUNTER,TEMP
        OUT OCR0A, COUNTER
        RJMP LOOP

DEC_COUNTER:
        CALL ANTIBOUNCE_BTN2
        SUB COUNTER, TEMP
        OUT OCR0A, COUNTER
        RJMP LOOP

RESET_COUNTER:
        LDI COUNTER, 0x00
        CALL DELAY
        RJMP LOOP

ANTIBOUNCE_BTN1:
        CALL DELAY
        SBIS PIND, PIN_BTN1
        RJMP LOOP
        RET

ANTIBOUNCE_BTN2:
        CALL DELAY
        SBIS PIND, PIN_BTN2
        RJMP LOOP
        RET

DELAY:
; Delay 600 000 cycles
; 37.5ms at 16.0 MHz
    LDI  R20, 60
L1: LDI  R21, 40
L2:     LDI  R22, 250
L3:     DEC  R22
    BRNE L3             ; Repetir 250 veces
    DEC  R21
    BRNE L2             ; Repetir L3 unas 40 veces      (Ciclos = 250*40 = 10 000)
        DEC  R20
        BRNE L1         ; Repetir L2 unas 80 veces      (Ciclos = 10 000*60 = 600 000)
        RET
```

Se utilizo la directiva *.DEVICE* para identificar el tipo de microcontrolador utilizado. Luego con la directiva *.EQU* se definieron las constantes que van a ser utilizadas en el código y con la directiva *.DEF* se definieron nuevos nombres significativos para los registros utilizados.

La primera tarea del *Main* es inicializar el Stack. Luego se configura el puerto D tal que los pines 2 y 3 corresponden a los pulsadores de aumento y decremento de intensidad respectivamente, y por ultimo el pin 6 como salida en donde sera conectado el LED ya que este pin corresponde al registro OCR0A del timer 0.
Se configura el timer 0 como "Fast PWM.[en] modo 3 al introducir los valores WGM2:0=011 en el registro TCCR0A. Con este modo se configura el valor de TOP en 0xFF, se actualiza el valor de OCR0A como BOTTOM y se produce un overflow al llegar al valor máximo que sera el TOP. Esto nos permite variar el ancho de modulación de banda al variar el valor de OCR0A produciendo así que el pulso varié al ser mas estrecho el limite entre BOTTOM y TOP. A medida que el valor de OCR0A se acerque al máximo, el pulso generado tendrá un valor promedio de señal, o "duty cycle", mayor y esto se vera reflejado en una mayor intensidad lumínica del LED. Además se configuro la salida por el registro OCR0A en modo "non-invertedçon el valor COM0A1:0 = 10, y se eligió no variar la frecuencia del clock al no introducir "preesclaingçon el valor CS00:1 = 01 del registro TCCR0B.
El programa actualiza el valor de OCR0A según el valor del "PASOïngresado al inicio del mismo. El loop principal verifica el estado de los pulsadores para conocer a cual subrutina debe ingresar. Si se activa el "PIN_BTN1"

se ingresa a "$INC\_COUNTER$" en donde se actualiza el valor de OCR0A sumando el PASO determinado. Al contrario, si se activa el "PIN_BTN2" se ingresa a "$DEC\_COUNTER$" en donde se actualiza el valor de OCR0A restando el PASO determinado.
Es importante destacar que para cada estado alto que se detecta se procede 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 37,5ms dado que el mismo permitía obtener una cantidad de pulsos cercano al valor de 255 con paso unitario.

$$\frac{ciclos}{frecuecia} = \frac{600000}{16MHz} = \frac{250 * 4 * 80}{16MHz} = 37{,}5ms \tag{1}$$

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.

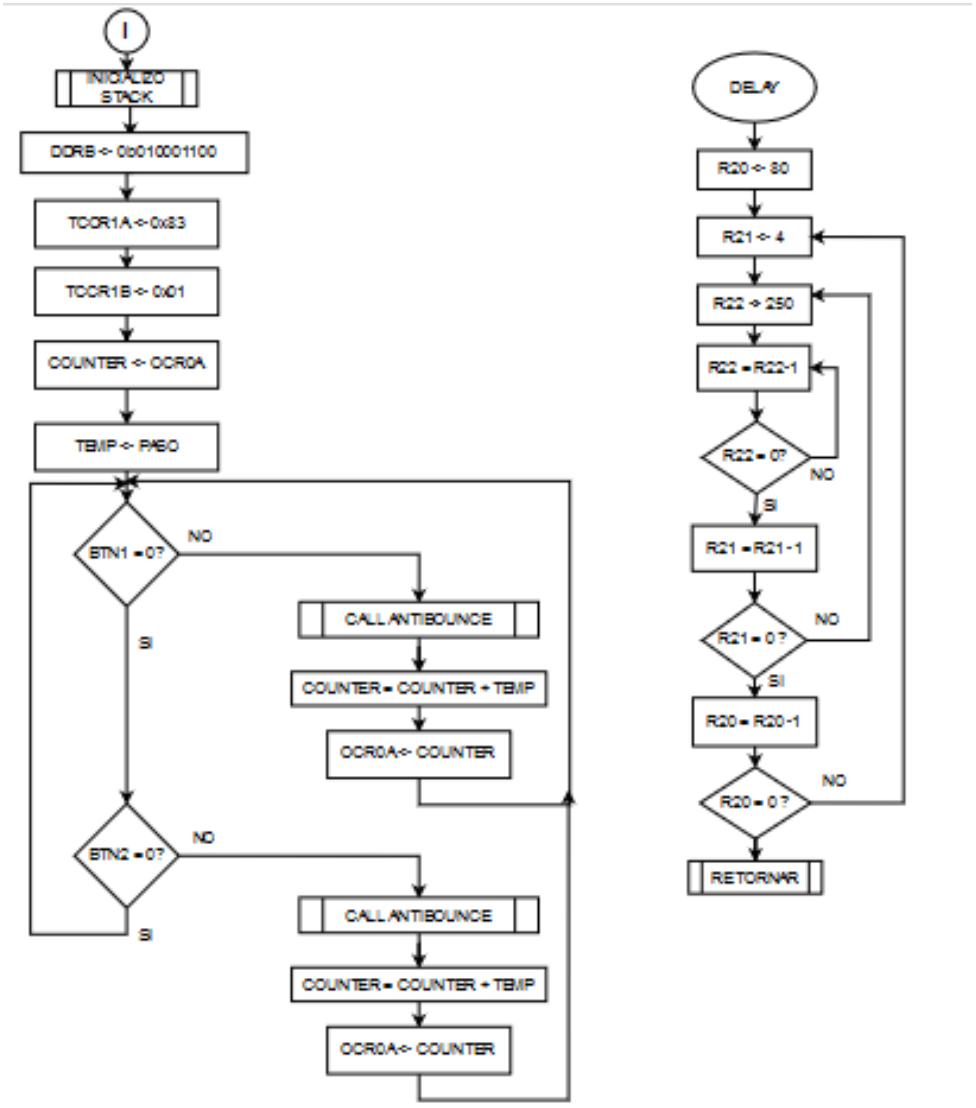

Figura 1: Diagrama de flujo del loop principal.

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: 1 diodo led, dos pulsadores, dos resistencia de $10K\Omega$ y 1 resistencias de 220Ω. Para la alimentación y trasmisión del código se utilizo el puerto USB de salida incorporado a 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.

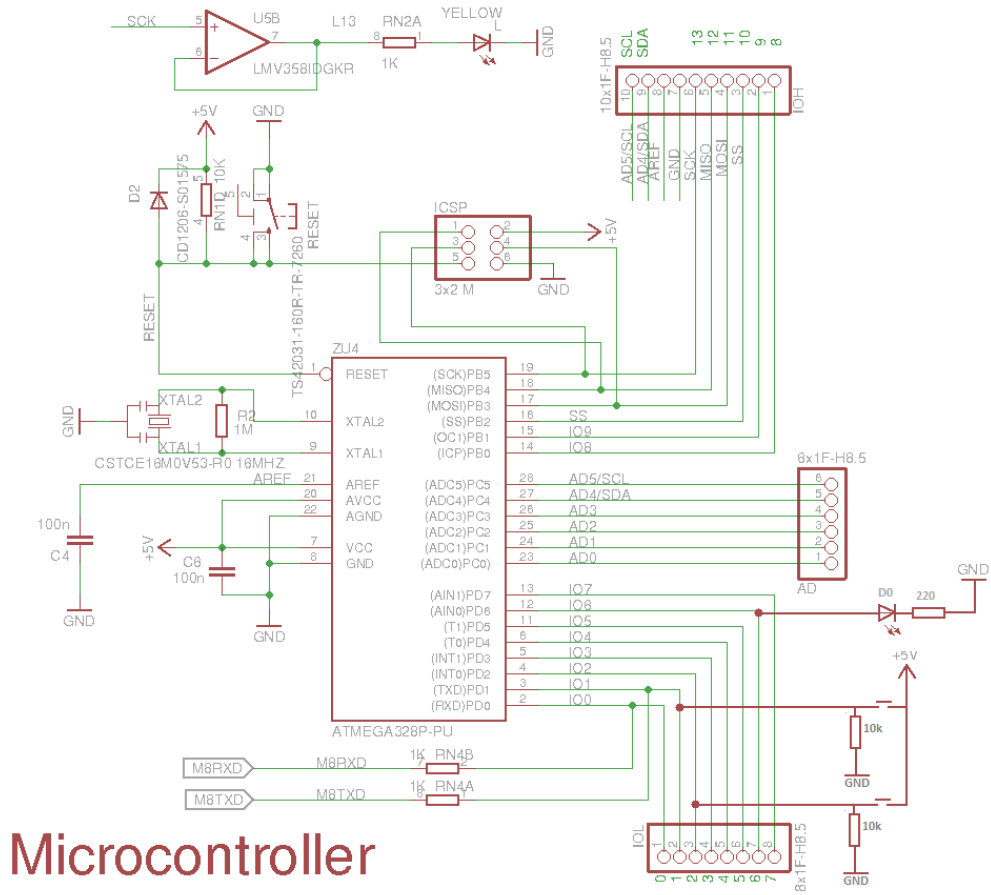

Figura 2: Diagrama de conexión.

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.

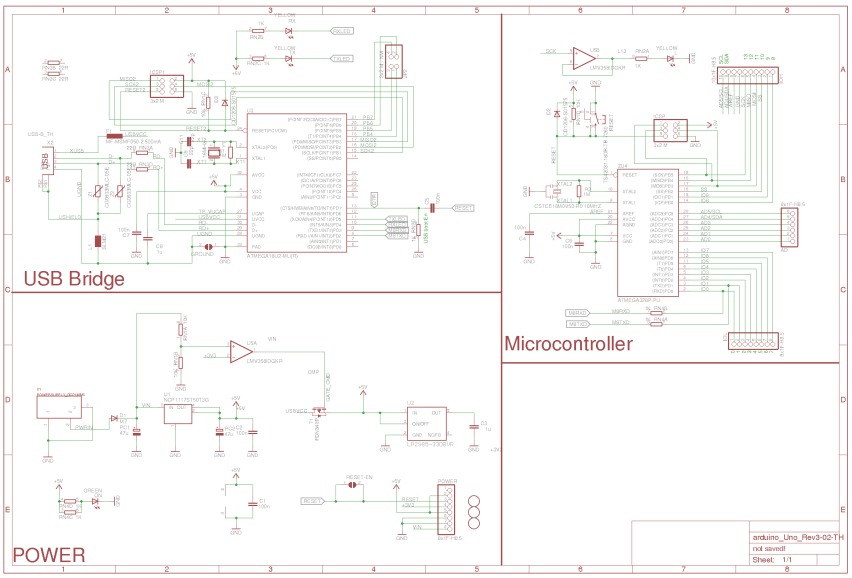

Figura 3: Diagrama de conexión de la placa.

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 1: Precios de componentes.

3. Conclusiones

Para este trabajo practico se introdujo el uso de timers como una herramienta útil para generar diferentes señales de tren de pulsos con duty cycle variable según el deseo del usuario. Esta herramienta es útil para trabajar con motores y distintos periféricos que utilicen corriente continua, y necesiten un control por modulación de ancho de pulso. Por ejemplo, podemos extrapolar el brillo del LED como la velocidad de un motor de continua. Además, se utilizo una rutina de pooling para chequear la variación de estados de los pulsadores y una rutina para manejar el rebote de ambos pulsadores.

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