Universidad de Buenos Aires
Facultad De Ingeniería
Año 2021 – 1.er Cuatrimestre

Trabajo Práctico Integrador.

8607 Laboratorio de microprocesadores - Curso 3

Alumno		Padrón		Correo electrónico
Scarramberg, Álvaro	—	103370	—	ascarramberg@fi.uba.ar
Pistone, Sofía	—	102456	—	spistone@fi.uba.ar

Índice

1. Objetivo del proyecto

Desarrollar un programa el cual permite analizar, mediante un gráfico, la carga y descarga de un circuito RC el cual es alimentado mediante una señal generada y verificada por el mismo microcontrolador que realiza las mediciones del circuito.

2. Descripción del proyecto

El trabajo consiste en el uso del conversor analógico digital del microcontrolador Atmega328p para la lectura de un circuito RC alimentado por una señal cuadrada de 60Hz para su posterior transmisión mediante el modulo USART a una computadora, de forma que se pueda obtener el gráfico temporal de la tensión sobre este circuito.

La señal con la que se alimenta al RC es generada por el mismo microcontrolador, el cual a su vez debe comprobar si la frecuencia y el Duty Cycle de dicha señal son efectivamente los esperados y luego mandar un mensaje a la computadora con la confirmación.

3. Circuito esquemático

Usando Proteus se obtuvo el esquemático del circuito utilizado. El resultado se muestra en la siguiente figura:

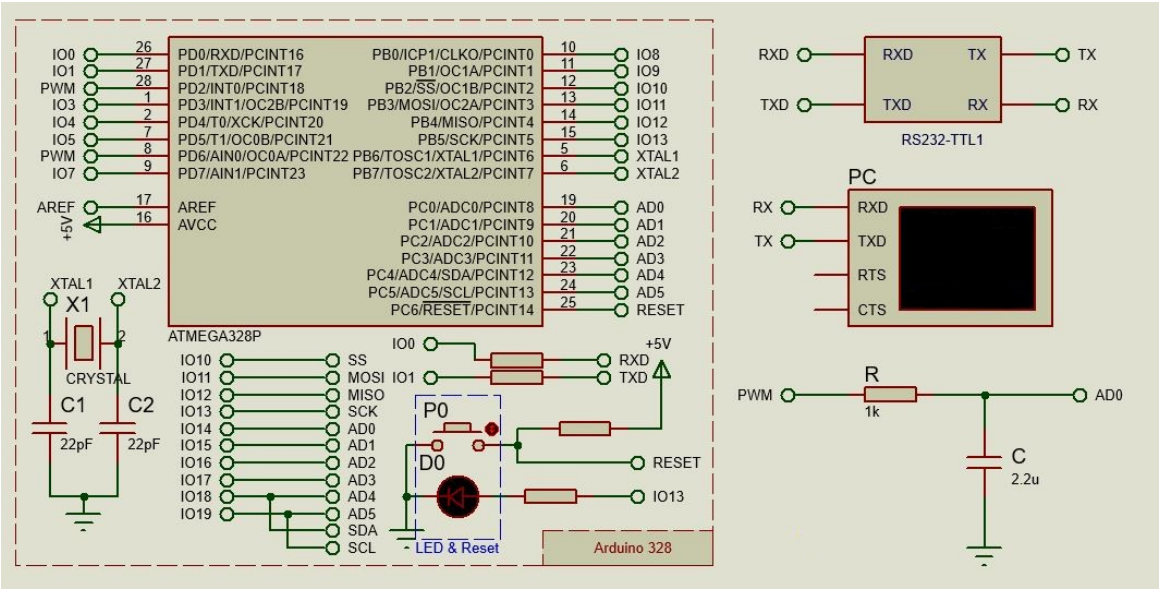

Figura 1: Esquemático del circuito en Proteus.

Se seleccionaron los valores de R y C de manera que en la duración del semi-periodo de la señal cuadrada que alimenta al circuito transcurran mas de 3 constantes de tiempo del RC. Para un capacitor de 2,2uF se calculo el valor del resistor:

$$\tau = R \cdot C \qquad \frac{T}{2} = 8{,}33ms > 3 \cdot \tau = 3 \cdot R \cdot C \qquad \Rightarrow \qquad R < \frac{8{,}33ms}{3 \cdot C} = 1{,}26k\Omega$$

Dado el valor del resistor se ve que los parámetros de los elementos seleccionados cumplen la condición requerida para la constante de tiempo del circuito RC.

4. Listado de componentes y tabla de gastos

Del mismo esquemático se obtuvo la lista de componentes y sus precios asociados:

5 Resistors			
Quantity	References	Value	Unit Cost
4	R,R1-R3	1k	$2.72
1	R4	10k	$3.50
Sub-totals:			$14.38
1 Integrated Circuits			
Quantity	References	Value	Unit Cost
1	U1	ATMEGA328P	$430.00
Sub-totals:			$430.00
1 Diodes			
Quantity	References	Value	Unit Cost
1	D0	LED-RED	$3.50
Sub-totals:			$3.50
4 Miscellaneous			
Quantity	References	Value	Unit Cost
1	P0	Pulsador	$13.50
1	PC		
1	RS232 TTL-1		$325.50
1	X1	CRYSTAL	$47.49
Sub-totals:			$386.49
3 Capacitors			
Quantity	References	Value	Unit Cost
1	C	2.2u	$11.60
2	C1-C2	22pF	$3.81
Sub-totals:			$19.22
Totals:			$853.59

Figura 2: Lista de componentes y precios.

5. Diagrama de flujo del programa

El programa basa su funcionamiento principalmente en el uso de interrupciones por lo que el programa principal simplemente realiza las diversas configuraciones necesarias para el funcionamiento de los timers, interrupciones externas, el conversor digital y el modulo de comunicación serial. Se muestra a continuación el diagrama de flujo correspondiente a esta parte del programa:

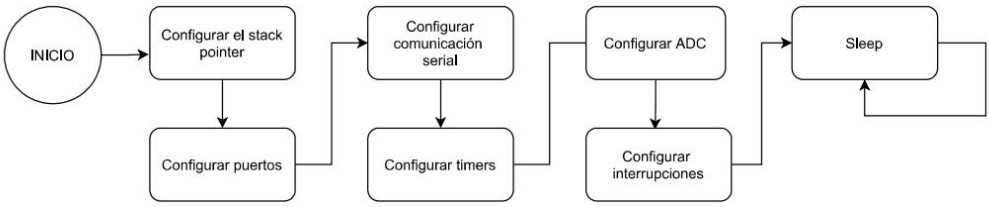

Figura 3: Diagrama de flujo del programa general de la primera parte.

Para alimentar al circuito RC se necesita generar una señal cuadrada de 60Hz y un duty cycle del 50 % utilizando al timer 0. Para lograr esto se decidió configurarlo en modo FAST PWM con su pin de salida OC0A en modo toggle de forma que por cada coincidencia en la comparación del timer con OCR0A se tenga un semi-periodo de la señal de salida y por lo tanto que se pueda configurar fácilmente la duración del mismo, además de esta forma se asegura que el duty cycle de la señal generada es exactamente del 50 % (Según lo que indica la hoja de datos).

Teniendo en cuenta que el clock utilizado es de 16MHz, para un prescaler de 1024 y un valor de OCR0A de 129 se puede calcular que la frecuencia de la señal obtenida es:

$$F_{PWM} = \frac{F_{Clk}}{2 \cdot (O_{CR0A} + 1) \cdot Prescaler} = 60{,}096Hz \quad \Rightarrow \quad Error = 0{,}16\,\%$$

Luego de finalizar todas las configuraciones iniciales el programa debe medir el periodo y el ciclo de trabajo de la señal generada para verificar que cumple efectivamente las condiciones requeridas. Esto se logra haciendo uso de la interrupción externa 0 en modo pin change y conectada al pin de salida de la señal PWM, en conjunto con la interrupción por captura del Timer 1.

Para la implementación de la lógica que permite hacer esto se utilizaron a 'T' y 'C' como flags, los cuales, se debe aclarar, están inicialmente en estado apagado.

La lógica implementada en el programa ante activaciones de la interrupción externa 0 se describe en el siguiente diagrama de flujo:

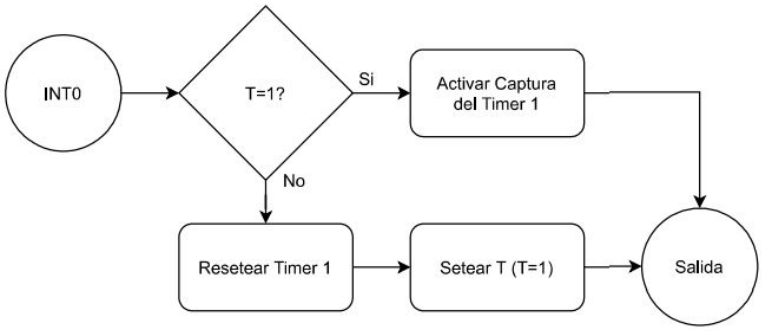

Figura 4: Diagrama de flujo correspondiente a la interrupción externa 0.

Se ve que en primer lugar se debe asegurar que se comience a contar a partir de un punto adecuado y por eso es que ante el primer flanco se reinicia al timer 1 de forma que al detectar los siguientes flancos y activar la interrupción por captura del timer, este va a contener un valor que refleja realmente la duración del semi-periodo de la señal.

En la primera activación de la interrupción por captura se tiene asegurado que el contenido del timer 1 corresponde a la medición del tiempo de duración de uno de los semi-periodos de la señal cuadrada, pero esto no es suficiente para poder determinar los parámetros de la señal por lo que resulta necesario almacenar en algún registro los valores medidos y realizar la medición de la duración del siguiente semi-periodo de la señal, para lograr esto se reinicia al timer 1 y se espera a la siguiente activación de la interrupción externa e inmediatamente después la segunda activación de la interrupción por captura del timer 1.

Ya llegado a este punto se tienen las mediciones de la duración de ambos semi-periodos de la señal cuadrada y se puede determinar si cumple las condiciones requeridas, para ver esto simplemente se debe comprobar que ambos valores medidos sean iguales o que al menos la diferencia entre ambos sea despreciable, para verificar el duty cycle y que la duración correspondiente a estos corresponda a un rango de tolerancia para el cual se considera que la señal medida es de 60Hz (En el programa implementado se admitió una tolerancia del 5 %).

Todo este proceso que se realiza en la interrupción por captura del timer 1 se resume en el siguiente diagrama de flujo:

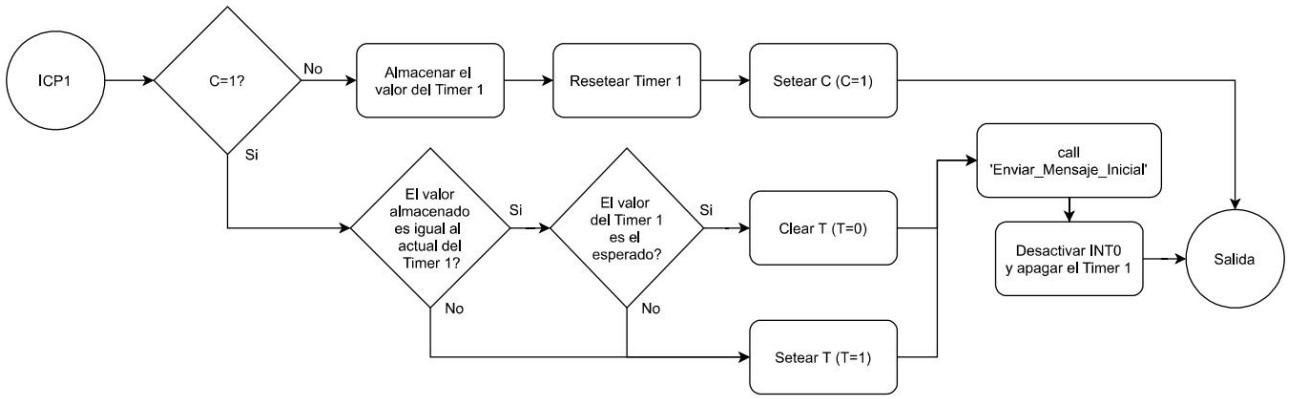

Figura 5: Diagrama de flujo correspondiente a la interrupción por captura del Timer 1.

Se ve que luego de determinar si la señal es la esperada o no la interrupción utiliza a T como flag para indicar el resultado de esta verificación y llama a la función 'Enviar_Mensaje_Inicial' la cual se encarga de enviar el mensaje adecuado según el resultado de la verificación y al finalizar activa por primera vez al conversor analógico digital.

En el caso de que el pin correspondiente a la interrupción externa no este conectado a la salida de la señal PWM es posible que el programa se quede trabado esperando a un cambio de flanco que probablemente nunca ocurra o simplemente tarde tanto que su duración no se pueda almacenar en el timer 1, sea cual sea el caso, es necesario evitar que esto suceda y se pueda avanzar a la parte del programa que realiza las conversiones de tensión y envía los resultados ya que en caso de que sea necesario esto puede resultar útil para encontrar errores en el circuito y se pueda determinar que es realmente lo que esta midiendo tanto el ADC como la interrupción externa 0.

Para lograr esto simplemente se utiliza la interrupción por overflow del timer 1. Cuando se produce esta interrupción se sabe claramente que la señal medida no puede ser de ninguna manera la que se estaba esperando generar por lo que simplemente se setea como flag a 'T' y se llama a la función 'Enviar_Mensaje_Inicial' de forma que se envíe el mensaje de verificación incorrecta y se continúe con el funcionamiento del resto del programa:

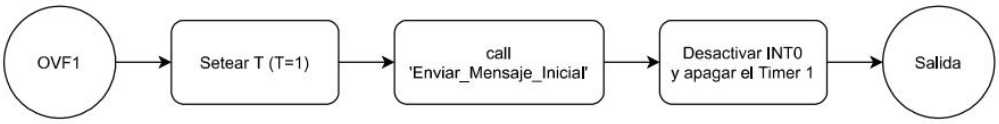

Figura 6: Diagrama de flujo correspondiente a la interrupción por overflow del Timer 1.

Luego de que se finaliza el envío del mensaje de verificación el programa realiza una primera conversión de la tensión analógica a digital, la cual va a provocar que el ADC comience a activarse continuamente ya que se configuró a este módulo en el modo de disparo automático. Al ADC se lo configuró con un prescaler de 128 por lo que su frecuencia de conversión se puede obtener de la siguiente forma:

$$F_{ADC} = \frac{F_{clk}}{13 \cdot prescaler} = 9{,}62kHz$$

Para realizar el envío de los datos se habilitó la interrupción por conversión completa del ADC de forma que al finalizar cada conversión se inicie el envío del resultado obtenido.

Al momento en el que se tiene la activación de esta interrupción es necesario que el anterior dato medido ya se haya terminado de enviar, es por eso que el Baudrate utilizado fue de 115,2kb/s tal que, para el formato '8N1' utilizado, la frecuencia de transmisión de la información sea de 11,52kHz de forma que esta frecuencia sea mayor a la de conversión del ADC y no haya ningún problema al hacer el envío de los datos.

Para el valor de Baudrate seleccionado la hoja de datos indica que el valor real de la frecuencia de transmisión va a resultar un 3.5 % menor al esperado, pero para los fines de este trabajo los errores que se puedan generar a causa de esto no van a resultar apreciables.

El conversor analógico digital devuelve resultados con una resolución de 10 bits, pero a la computadora solo se le envían los 8 bits mas significativos debido a que para los fines de este trabajo la resolución que se obtiene en este caso resulta suficiente para obtener resultados aceptables al construir el gráfico de la tensión medida. Hacer esto permite simplificar notablemente la lógica del programa implementado. La tensión de alimentación es de 5V por lo que para 8 bits de representación se ve que la resolución de las mediciones va a ser de 20mV aproximadamente.

Teniendo en cuenta que cada semi-periodo de la señal medida dura aproximadamente 8,3ms y que para la frecuencia seleccionada para el ADC el tiempo entre cada muestreo es de 104us, se puede calcular que la cantidad de muestras que se miden en cada semi-periodo son aproximadamente 80 y por lo tanto en cada periodo son 160.

Para la recepción de los datos en la computadora se utilizó un script de MATLAB que permite leer los datos recibidos por el puerto serie seleccionado y con estos generar un gráfico de la tensión del RC en función del tiempo, lo cual se puede lograr ya que se conoce la tasa de muestreo con la que se obtuvieron los datos.

6. Resultados

Se observa un correcto funcionamiento del circuito con el programa implementado, este se probó en un modelo físico el cual permitió, con ayuda de MATLAB, obtener el siguiente gráfico el cual representa la carga y descarga del RC. Se nota que no se supera los límites 0V-5V y que no llega a cargarse por completo, lo cual es esperable debido a que lo mismo sucedería en aproximadamente 5τ, lo cual es mayor a la señal generada.

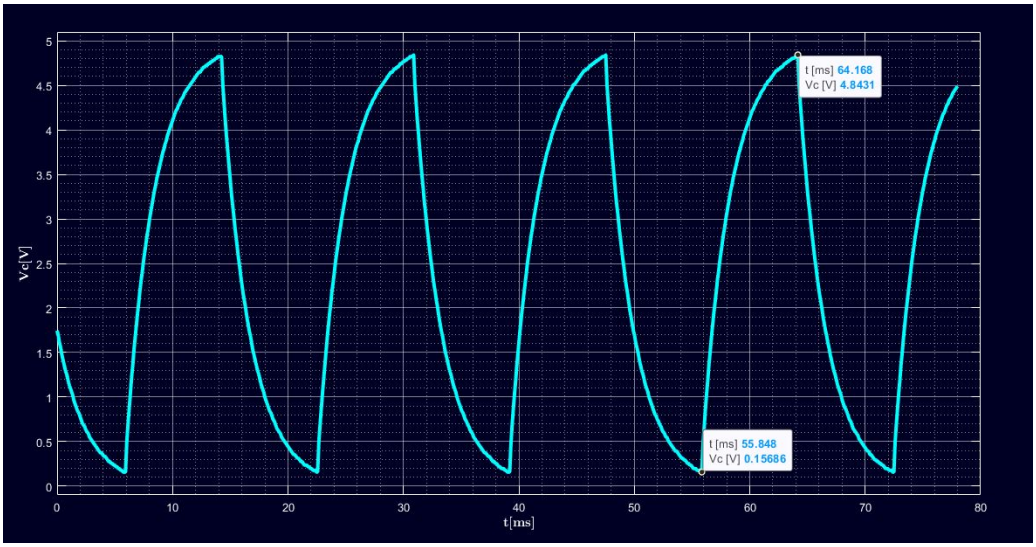

Figura 7: Resultado de la medición de la tensión sobre el circuito RC.

7. Conclusión

Siguiendo las pautas pedidas, se pudo implementar satisfactoriamente el programa y se pudo corroborar su correcto funcionamiento como se puede ver en la sección de resultados. A lo largo de este proyecto se logró muestrear y construir la carga y descarga de un capacitor. Se pudo aprender tanto del funcionamiento del conversor analógico (ADC) como profundizar los conocimientos de las distintas configuraciones de Timers y sus diferentes aplicaciones.

A lo largo del desarrollo de este trabajo se presentaron algunas dificultades, como lograr que en la parte donde se comprueba que el Duty Cycle funcione a la par, sin interrumpir, a la función que envía los datos muestreados. Se notó la importancia de realizar de manera previa los cálculos pertinentes, de modo que se pueda asegurar que ninguna funcionalidad compromete a la otra.

8. Código utilizados

8.1. Código del programa

Se muestra a continuación el código de assembly del trabajo:

```
;
; Laboratorio de microprocesadores[8607] - Curso 3
;
; Trabajo Práctico Integrador
;
; Objetivo : Desarrollar un programa el cual permite
; analizar, mediante un grafico, la carga y descarga
; de un RC el cual es alimentado mediante una señal
; generada y verificada por el mimsmo micro.
;
; Autores:
;        Sofía Pistone - 102456
;        Álvaro Scarramberg - 103370
;
; Fecha de entrega: 23-07-2021
;*********************************************************

;.include "m328pdef.inc"

; Defino constantes
.equ baud=8 ;Para tener un baud rate de 115200 con un
              ;clock de 16MHz se usa este valor

; Definiciones de registros
.def temp=R16
.def contador=R20

; Macros
.macro setStack
        ldi r16, low(@0)
        out SPL, r16
        ldi r16, high(@0)
        out SPH, r16
.endmacro

.dseg
.org SRAM_START

SREG_RAM: .byte 1

.cseg
.org 0x0000
        rjmp        main
.org INT0addr
        rjmp        Handler_Int_Ext0
.org ICP1addr
        rjmp        Handler_Int_T1_Capt
.org OVF1addr
        rjmp        Handler_Int_OVF1
.org UTXCaddr
```

```
        rjmp        Handler_TXC
.org ADCCaddr
        rjmp        Handler_Int_ADCC

.org INT_VECTORS_SIZE

main:
        setStack RAMEND
        rcall configurar_puertos
        rcall configurar_serial
        rcall configurar_timers
        rcall configurar_ADC
        rcall configurar_int0

        ; Se almacena a SREG en
        ; RAM de forma auxiliar
        in temp, SREG
        sts SREG_RAM, temp

        sei

main_loop:
        sleep
        jmp main_loop

;***********************************************************
configurar_puertos:

        ;PB0 como puerto de salida para
        ;poder activar la captura del T1

        in temp, DDRB
        ori temp, (1<<DDB0)         ;salidas
        out DDRB, temp

        ;PC0 entrada al ADC

        in temp, DDRC
        andi temp, ~(1 << DDC0)     ;entradas
        out DDRC, temp

        ;PD2 entrada a int0
        ;PD6 salida del pwm

        in temp, DDRD
        andi temp, ~(1 << DDD2)     ;entradas
        ori temp,   (1 << DDD6)     ;salidas
        out DDRD, temp

        ret

;***********************************************************************
configurar_serial:
        ldi r16, LOW(baud)
        ldi r17, HIGH(baud)
```

```
        sts UBRR0L, r16         ; Cargar el prescaler
        sts UBRR0H, r17         ; a UBRR0

        ; 8N1: 8 bits de datos
        ; sin paridad y 1 bit de stop

        ; configurar modo de transmisión: Asincrónico-->UMSEL=00
        ; Paridad: desactivada-->UPM0=00
        ; stop bits: 1 bit-->USBS0=0
        ; Cantidad de bits del mensaje: 8-->UCSZ0=011

        ldi R16, (0<<UMSEL00)|(0<<UPM00)|(0<<USBS0)|(3<<UCSZ00)
        sts UCSR0C, R16

        ; Habilitar transmitter
        ldi R16, (0<<RXEN0) | (1<<TXEN0) | (0<<UCSZ02)
        sts UCSR0B, R16

        ; Baudrate = 115200 bit/s --> FT= 11.52kByte/s
        ; Un byte tarda en enviarse: TT=86.8us

        ; Habilito la interrupción por Tr. completa
        lds R17, UCSR0B
        ori R17, (1<<TXCIE0)
        sts UCSR0B, R17

        ret

;***********************************************************************
configurar_timers:

;-----------------------------------------------------------------------
; TIMER 1 - Modo Fast PWM ---> Genera la señal PWM
;-----------------------------------------------------------------------

        ; Se configura al timer 0 en modo Fast PWM
        ; Con TOP=OCR0A
                ; WGM0 = 111

        ; Para obtener el DC del 50%
        ; OC0A --> Toggle on compare match
                ; COM0A = 01

        ; Cada 2 comparaciones se tiene un periodo
        ; de la señal de salida por lo tanto:

        ; FPWM =  Fclk / ( 2 * (OCR0A+1) * prescaler )

        ; OCR0A = 129 - Prescaler=1024 --> FPWM = 60.096 Hz  ERROR = 0.16%

        in temp, TCCR0A
        andi temp, ~(1<<COM0A1)
        ori temp, (1<<WGM01)|(1<<WGM00)|(1<<COM0A0)
        out TCCR0A, temp
```

```
        in temp, TCCR0B
        andi temp, ~(1<<CS01)
        ori temp, (1<<CS02)|(1<<CS00)|(1<<WGM02)
        out TCCR0B, temp

        ldi temp, 129
        out OCR0A, temp

;-----------------------------------------------------------------------
; TIMER 1 - Modo Captura
;-----------------------------------------------------------------------
        clr temp; contador inicializado en 0
        sts TCNT1H, temp
        sts TCNT1L, temp

; Configuro modo normal (0) (WGM1=0000)
        ldi temp, (0 << WGM11 ) |(0 << WGM10 )
        sts TCCR1A , temp

; Prescaler 1024 - Rising edge
        ldi temp, (0<<WGM13)|(0<<WGM12)|(1<<CS02)|(0<<CS01)|(1<<CS00)|(1<<ICES1)
        sts TCCR1B, temp

; Habilito interrupción por captura
        ldi temp, (1 << ICIE1 )
        sts TIMSK1, temp

;        ldi temp, 0x01
;        clr r17
;        sts OCR1AH , temp
;        sts OCR1AL , r17

; Habilito la interrupción por overflow

        lds temp, TIMSK1
        ori temp, (1<<TOIE1)
        sts TIMSK1, temp

        ret


;*********************************************************
configurar_ADC:

        ; Se necesita que el tiempo que se tarda
        ; por conversión sea mayor al tiempo que
        ; se tarda en enviar el resultado

        ; TC>TT = 86.8us FC<FT = 11.52kHz

        ; FC = FADC/13 = FClk/(13*prescaler)
        ; Prescaler = 128 --> FC = 9.62kHz < FT --> ok!

        ; ADLAR = 1
        ; Se usa VCC como REF--> REFS=01
```

```
        ; Se selecciona el canal 0 --> MUX=0b000
        lds temp, ADMUX
        andi temp, ~((1<<REFS1)|(1<<MUX2)|(1<<MUX1)|(1<<MUX0))
        ori temp, (1<<REFS0)|(1<<ADLAR)
        sts ADMUX, temp

        ; Desactivo los puertos del ADC que no
        ; se utilizan para minimizar el consumo
        lds temp, DIDR0
        andi temp, ~ (1<<ADC0D)
        ori temp, 0b00111110
        sts DIDR0, temp

        ; Prescaler=128:                    DPS=111
        ; Se habilita al ADC:               ADEN=1
        ; Se habilita su interr.:           ADIE=1
        ; Se borra el flag:                 ADIF=0
        ; Se habilita el disparo automático: ADATE=1
        lds temp, ADCSRA
        andi temp, ~((1<<ADIF))
        ori temp, (1<<ADEN)|(1<<ADATE)|(1<<ADIE)|(1<<ADPS2)|(1<<ADPS0)|(1<<ADPS1)
        sts ADCSRA, temp

        ret

;*********************************************************************
configurar_int0 :         ; Se configura por Pin change
        ldi temp, (0 << ISC01 )|(1 << ISC00 )
        sts EICRA , temp

        ldi temp, (1 << INT0 )
        out EIMSK, temp

        ret

;*********************************************************
; Rutina que se encarga de comenzar el envío del mensaje
; inicial que indica el resultado de la verificación de
; la señal medida por int0
;
; Se utiliza a T como flag para decidir cual de los
; mensajes se debe enviar
;
;*********************************************************
Enviar_Mensaje_Inicial:

        ; Se verifica con el Flag cual
        ; mensaje se debe enviar
        brts Msj_Frecuencia_Incorrecta

Msj_Frecuencia_Correcta:
        ldi ZH, high(VerificacionCorrecta<<1)
        ldi ZL, low(VerificacionCorrecta<<1)
        rjmp Enviar_Primer_Caracter
```

```
Msj_Frecuencia_Incorrecta:
        ldi ZH, high(VerificacionIncorrecta<<1)
        ldi ZL, low(VerificacionIncorrecta<<1)

Enviar_Primer_Caracter:
        lpm temp, Z+
        sts UDR0, temp

Enviar_Mensaje_Inicial_End:
        ret

;*********************************************************
; Rutina que se encarga de comenzar las conversiones del
; ADC y el envio de los resultados
;
;*********************************************************
ComenzarTransmision:

        ; Realizar la primera conversión del ADC
        lds temp, ADCSRA
        ori temp, (1<<ADSC)
        sts ADCSRA, temp

        ret

;*************************************************************************
; Rutina que se encarga de comenzar el conteo y
; activar la interrupcion por captura
;
; Se reinicia T1 ante el primer flanco asc. o desc.
; detectado para comenzar a contar el primer semi-periodo
;
; Uso T como flag para saber si ya comenzó el conteo
;
;*************************************************************************
Handler_Int_Ext0:
        push temp
        in temp, SREG
        push temp
        lds temp, SREG_RAM
        out SREG, temp

        brtc comenzar_conteo

; Activar Captura del timer 1
        sbi PINB ,0
        rjmp fin_int

comenzar_conteo:
        clr temp
        sts TCNT1H , temp
        sts TCNT1L , temp
        SET

fin_int:
```

```
        in temp, SREG
        sts SREG_RAM, temp
        pop temp
        out SREG, temp
        pop temp
        reti

;***************************************************************************
; Rutina que se encarga de verificar que el el Duty - Cycle y F sean correctos.
;
; Mido 2 semiperiodos, si ambos son iguales entonces el duty cycle es
; correcto y si ademas el tiempo de duración de estos pertenece a cierto
; rango se comprueba que la frecuencia también es adecuada.
;
; Uso C como flag para indicar cual de los semi-periodos se está midiendo
;
;***************************************************************************
Handler_Int_T1_Capt:
; guardo registro de estado
        push temp
        in temp, SREG
        push temp
        lds temp, SREG_RAM
        out SREG, temp

; apagar PB0 ( ICP1 )
        sbi PINB ,0

        brcs Almaceno_Semi_Periodo_2

Almaceno_Semi_Periodo_1:
; Leer ICR1L e ICR1H
        lds r24 , ICR1L
        lds r25 , ICR1H
        SEC
        clr temp
        sts TCNT1H , temp
        sts TCNT1L , temp
        rjmp fin_Int_T1_Capt

Almaceno_Semi_Periodo_2:
        lds r22 , ICR1L
        lds r23 , ICR1H

        ; Compruebo la frecuencia con
        ; una tolerancia del 5%

        ; El semiperiodo debe ser de 8,33ms
        ; --> T/2 e [7.92ms ; 8.75ms ]

verificar_frecuencia:
        ; menor a 8,75ms = > ICR1L < 137 si es mayor hay un error
        cpi r24, 137
        brsh error
        ; mayor a 7.92ms = > ICR1L > 123 si es menor o igual hay un error
```

```
        cpi r24, 123
        brlo error

verificar_DC0:
        ; Compruebo si ambos semi-periodos son iguales
        cp R25, R23
        brne error

        ; Se admite una cierta tolerancia: |R24-R22|<7
        ; Para los valores esperados es de ~5%

        cp R24, R22
        breq correcto
        brlo verificar_DC2

verificar_DC1:
        sub R24, R22
        cpi R24, 7
        brsh error
        rjmp correcto

verificar_DC2:
        sub R22, R24
        cpi R22, 7
        brsh error

correcto:
        clt
        rjmp Enviar_Verificacion

error:
        set

Enviar_Verificacion:
        rcall Enviar_Mensaje_Inicial

        ; Apago int0
        ldi temp, (0 << INT0 )
        out EIMSK, temp

        ; Apago timer1
        ldi contador, (0 << CS02 ) |(0 << CS01 ) |(0 << CS00 )
        sts TCCR1B , contador
        clr temp
        sts TCNT1H , temp
        sts TCNT1L , temp

fin_Int_T1_Capt:
; Recupero registro de estado
        in temp, SREG
        sts SREG_RAM, temp
        pop temp
        out SREG, temp
        pop temp
        reti
```

```
;*********************************************************************
; Handler de la int. por Overflow del timer 1
;
; Si el timer 1 llega a overflow la señal medida tiene un periodo
; demasiado grande por lo que se asume que se tiene un error
;
;*********************************************************************
Handler_Int_OVF1:
        push R16
        push R17
        in R16, SREG
        push R16

        ; Indico error con T
        SET

        ; Envío el mensaje de verificación
        rcall        Enviar_Mensaje_Inicial

        ; Apago int0
        ldi temp, (0 << INT0 )
        out EIMSK, temp

        ; Apago al timer 1
        ldi contador, (0 << CS02 ) |(0 << CS01 ) |(0 << CS00 )
        sts TCCR1B, contador
        clr temp
        sts TCNT1H, temp
        sts TCNT1L, temp


Handler_Int_OVF1_End:
        pop R16
        out SREG, R16
        pop R17
        pop R16

        reti

;*********************************************************************
; Handler de la int. por transmisión completa
;
; Se encarga de realizar el envio del mensaje de verificación
;
;*********************************************************************
Handler_TXC:
        push R16
        push R17
        in R16, SREG
        push R16

        lpm R16, Z+         ; Cargar caracter de la tabla
        cpi R16, 0         ; Revisar si se llegó al final del string
```

```
        breq Fin_del_mensaje_inicial

        sts UDR0, r16 ; Enviar el caracter
        rjmp Handler_TXC_End

Fin_del_mensaje_inicial:
    ; Desactivo la interrupción por transmisión completa
        lds R17, UCSR0B
        andi R17, ~(1<<TXCIE0)
        sts UCSR0B, R17
        rcall ComenzarTransmision

Handler_TXC_End:
        pop R16
        out SREG, R16
        pop R17
        pop R16

        reti

;*****************************************************************
; Handler de la int. por conversión completa del ADC
;
; Al terminar una conversión simplemente se envía el resultado
;
;*****************************************************************
Handler_Int_ADCC:
        push R17
        push temp
        in temp, SREG
        push temp

        ; Enviar la parte alta del ADC
        lds temp, ADCH
        sts        UDR0, temp

Handler_Int_ADCC_End:
        pop temp
        out SREG, temp
        pop temp
        pop R17

        reti

;*********************************************************

VerificacionCorrecta: .db "** El duty cycle es de 50\% y la frecuencia de la señal PWM es de 6

VerificacionIncorrecta: .db "** La señal medida no es la esperada **",0
```

8.2. Script de MATLAB

Se muestra a continuación el código del script de MATLAB utilizado:

```
%%
% Leer datos del puerto serie
clear all;
clc

t=1:10000;
begin=9250;
T=128*13/(16*10^3);
puerto='COM6';

delete(instrfind({'Port'},{puerto}));
puerto_serial=serial(puerto);
puerto_serial.InputBufferSize=10001;
puerto_serial.BaudRate=115200;
puerto_serial.timeout=30;
warning('off','MATLAB:serial:fscanf:unsuccesfulRead');

fopen(puerto_serial);

y=fread(puerto_serial, length(t), 'uint8');

%%
% Generar el gráfico de la tensión del RC

xcyan = (1/255)*[0,255,255];%Color de línea%
dblue = (1/255)*[0,0,35];%Color de fondo%
plot([(begin:10000)-begin]*T,y(begin:10000)*5/255,'color', xcyan , 'linewidth', 3)
ylim([-0.1,5.1])

grid on
grid minor
box on
%legend([""]);
xlabel('$\textbf{t[ms]}$', 'interpreter', 'latex')
ylabel('$\textbf{Vc[V]}$', 'interpreter', 'latex')
ax = gca;
set(gca, 'color', dblue);
set(gcf, 'color', dblue);
ax.Title.Color = 'white';
ax.XLabel.Color = 'white';
ax.YLabel.Color = 'white';
ax.XColor = 'white';
ax.YColor = 'white';
ax.GridAlpha = 0.5;
ax.GridColor = 'white';
ax.MinorGridColor = 'white';
ax.MinorGridAlpha = 0.5;

hold off

%%
% Extraer el mensaje inicial
```

```
z=char(y);
x=transpose(z(1:100));
g=extractBetween(x,"** ", " **");
disp(g);

%%
% Cerrar el puerto y borrar las variables
fclose(puerto_serial);
delete(puerto_serial);
clear all;
```