



Laboratorio de Microprocesadores (86.07)

Proyecto: Trabajo Practico Integrador

Profesor:	Ing. Guillermo Campiglio
Cuatrimestre / Año:	1 ^{er} cuatrimestre 2021
Turno de clases prácticas:	Miércoles
Jefe de Trabajos Prácticos:	Ing. Pedro Ignacio Martos
Docente guía:	Ing. Fabricio Baglivo

Autores			Seguimiento del proyecto									
Nombre	Apellido	Padrón										
Gonzalo	Puy	99784										

Observaciones:

Fecha de aprobación		

Firma J.T.P

COLOQUIO	
Nota Final	
Firma Profesor	

Índice

1. Introducción	2
2. Desarrollo	3
2.1. Banco de mediciones	3
2.2. Programa a implementar	4
2.2.1. Generación de la señal mediante PWM	4
2.2.2. Verificación de la Frecuencia de la señal	5
2.2.3. ADC	5
2.2.4. Diagrama de flujo	6
2.3. Simulación	7
2.4. Código del programa	8
3. Resultados	15
4. Conclusiones	16

1. Introducción

En el presente trabajo se buscará cerrar conceptos vistos a lo largo de la materia y además, ver el funcionamiento del Conversor Analógico Digital (ADC) que tienen los microprocesadores de la familia AVR.

Para lograr estos objetivos, se realizará un programa cuya finalidad será la de graficar la carga y descarga de un capacitor.

2. Desarrollo

2.1. Banco de mediciones

Para llevar a cabo el trabajo se utilizó el siguiente banco de mediciones:

- Placa de desarrollo Arduino “UNO” y su respectivo cable para conectar la placa a la PC.
- El microcontrolador a usar, es el que viene integrado en la placa Arduino: Atmega328P.
- Protoboard
- Cables macho-macho para conexión del protoboard y de la placa Arduino.
- 1 resistor de 150 k Ω .
- 1 capacitor de 12 nF.

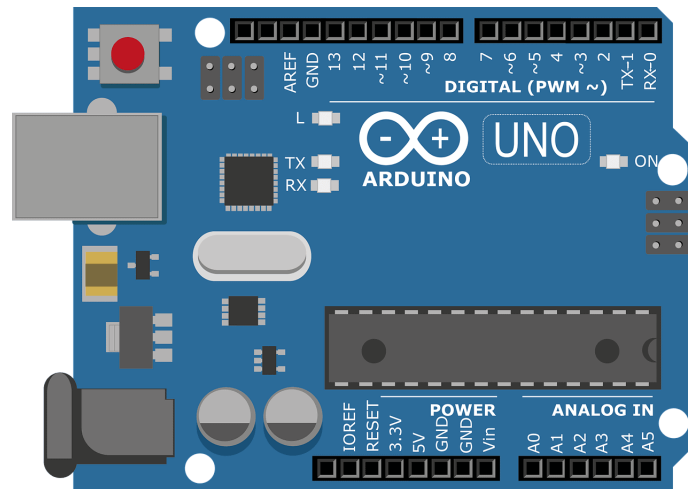


Figura 1: Placa de desarrollo Arduino UNO.

El diagrama de bloques simplificados del programa es el siguiente

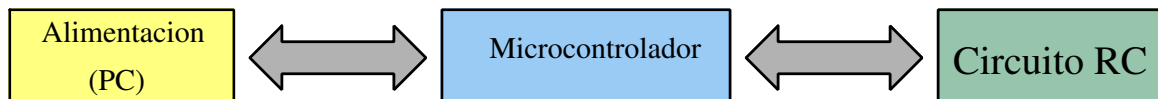


Figura 2: Diagrama de bloques.

La conexión utilizada es la siguiente

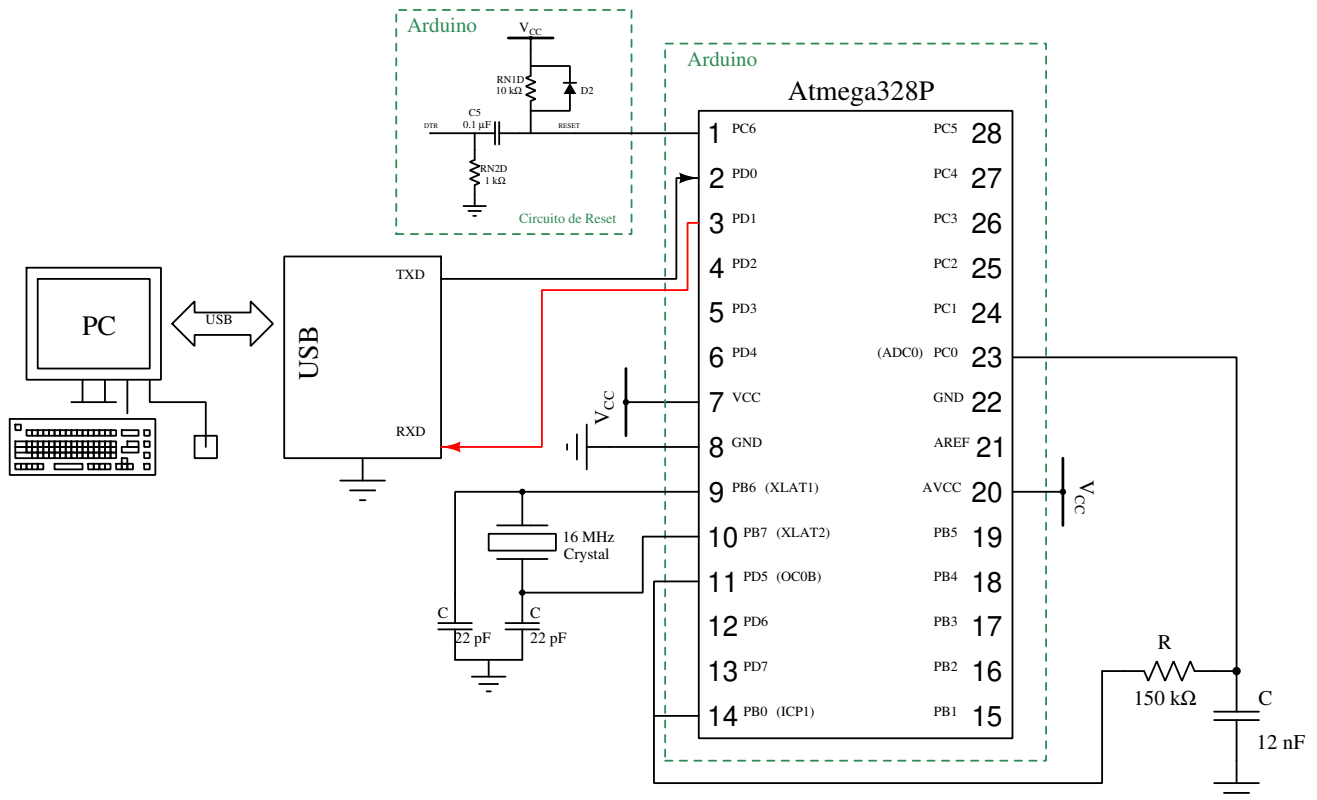


Figura 3: Caption

Es necesario aclarar que no se usó la misma conexión dada por el enunciado del trabajo. En este, el circuito RC se encontraba conectado al PIN 0C1A. Esto significa que la señal cuadrada generada con el PWM era mediante el Timer 1, esto no tiene sentido si se utiliza el modo captura del mismo Timer 1 para obtener la frecuencia de la señal generada. Por esto se decidió generar la señal cuadrada con el Timer 0.

2.2. Programa a implementar

Este programa, generará una señal cuadrada con PWM del Timer 0, luego se medirá la frecuencia de la señal mediante el modo captura del Timer 1. Finalmente se obtendrá con el ADC valores de tensión sobre el capacitor (V_C) para luego transmitirlos vía serial a la PC y graficarlos.

Tanto el valor de la frecuencia de la señal como los valores de tensión serán recibidos en la PC mediante un *script* de Python para luego ser procesados.

2.2.1. Generación de la señal mediante PWM

En el enunciado se pide obtener una señal mediante FAST-PWM de 50 Hz y un Duty Cycle (Ciclo de trabajo) del 50 %.

Obtener una frecuencia de 50 Hz con el Timer 0 no es posible, ya que este es un Timer de 8 bits. Por lo que se buscó alguna configuración que de una frecuencia lo mas cercana posible a la pedida.

Por lo tanto, se utilizó el modo 3 del Timer 0 (FAST-PWM no inversor) con el prescaler de 1024. Además, se tomó como salida de la señal el PIN 0C0B, dejando 0C0A desactivado. En consecuencia, la frecuencia de la onda generada queda dada según la fórmula

$$f = \frac{f_{oscilador}}{256 \cdot N} = \frac{16 \text{ MHz}}{256 \cdot 1024} = 61,035 \text{ Hz}$$

Notar, que esta frecuencia resultó bastante cercana a la pedida por el enunciado.

Para encontrar el valor que se debe colocar en el registro OCR0B para obtener un ciclo de trabajo del 50 % se utilizó la siguiente fórmula

$$\text{Duty Cycle} = \frac{\text{OCR0B} + 1}{256} \cdot 100$$

Reemplazando Duty Cycle por 50, se obtuvo un valor de 127.

2.2.2. Verificación de la Frecuencia de la señal

Para obtener el valor de la frecuencia generada, se utilizo el modo captura del Timer 1, el cual se configuro en modo normal, 0C1A y 0C1B desactivados, con un prescaler de 8. Esto significa que el timer tendrá una frecuencia de $f = 16 \text{ MHz}/8 = 2 \text{ MHz}$

Mediante software, se realizó el calculo del periodo de la señal generada con el PWM del Timer 0. Ese valor se transmitió a la PC donde se hicieron los cálculos finales mediante el *script* de Python.

2.2.3. ADC

Por ultimo, el ADC se configuro para que la referencia de voltaje sea AV_{CC} . Este PIN tiene una conexión interna en la placa Arduino por la cual se tiene que $AV_{CC} = V_{CC} = 5 \text{ V}$.

También se seteo el bit ADLAR del registro ADMUX en 1, por lo que al terminar al conversión, los resultados en los registros ADCH y ADCL estarán dados como se muestra en la siguiente figura

24.9.3.2 ADLAR = 1

Bit	15	14	13	12	11	10	9	8	
(0x79)	ADC9	ADC8	ADC7	ADC6	ADC5	ADC4	ADC3	ADC2	ADCH
(0x78)	ADC1	ADC0	—	—	—	—	—	—	ADCL
	7	6	5	4	3	2	1	0	
Read/Write	R	R	R	R	R	R	R	R	
	R	R	R	R	R	R	R	R	
Initial Value	0	0	0	0	0	0	0	0	
	0	0	0	0	0	0	0	0	

Figura 4: Caption

De esta forma, se podrá descartar el registro ADCL que tiene los bits menos significativos de la conversión los cuales suelen ser ruido.

Entonces como la conversión es de 8 bits, se pueden obtener los valores digitales que van a obtenerse con el ADC. Estos están dados por la fórmula

$$D_{out} = \frac{V_{in}}{\text{step size}}$$

Donde $\text{step size} = 5/256 = 19,53 \text{ mV}$. Reacomodando la fórmula anterior se obtendrán los valores de las tensiones que serán transmitidas a la PC para luego ser graficadas. Esta cuenta se realizará mediante el *script* de Python.

Como se busca que la conversión sea lo mas precisa posible, se utilizó el mayor prescaler posible (128).

Para terminar con la configuración del ADC, se seteo como PIN de entrada a ADC0 y se configuró el conversor en el modo *Auto Trigger-Free running mode*, con lo cual el conversor realizará una conversión tras otra.

2.2.4. Diagrama de flujo

Finalmente se muestra a continuación el diagrama de flujo del programa, el cual indica de forma resumida el funcionamiento del programa.

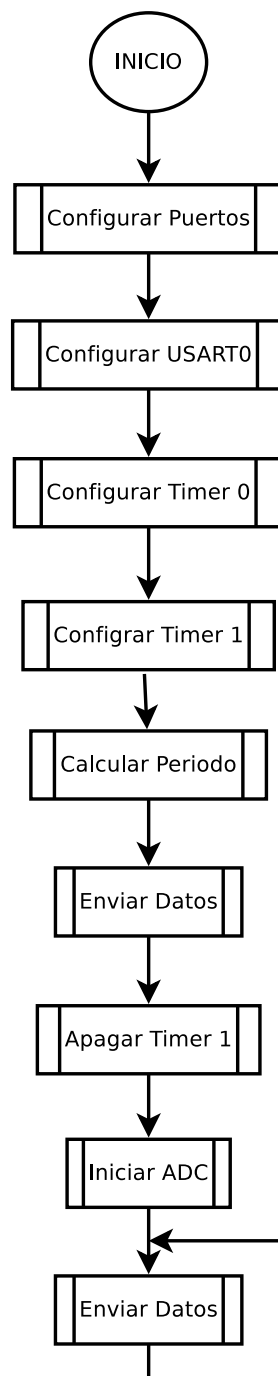


Figura 5: Diagrama de flujo del programa.

2.3. Simulación

Para obtener una idea general de lo que se debería ver con el gráfico generado por el programa, se realizó una simulación en *LTSpice*. Mediante el siguiente esquema y código en Python se muestran los resultados de la simulación.

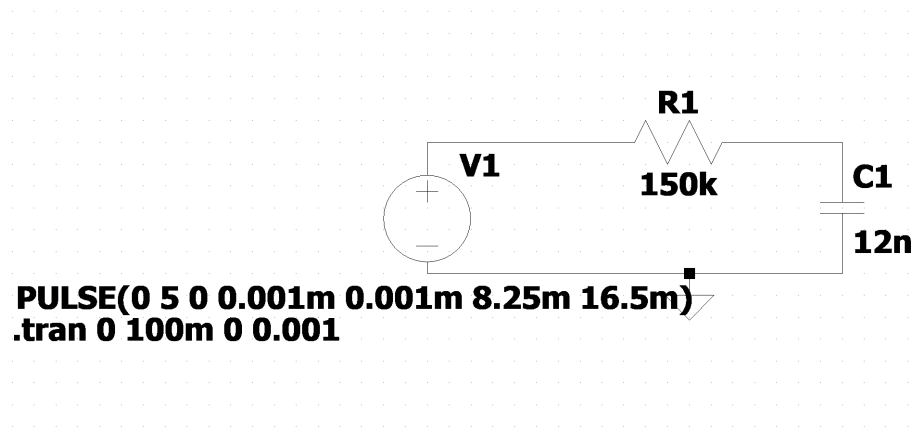


Figura 6: Esquema usado en Spice.

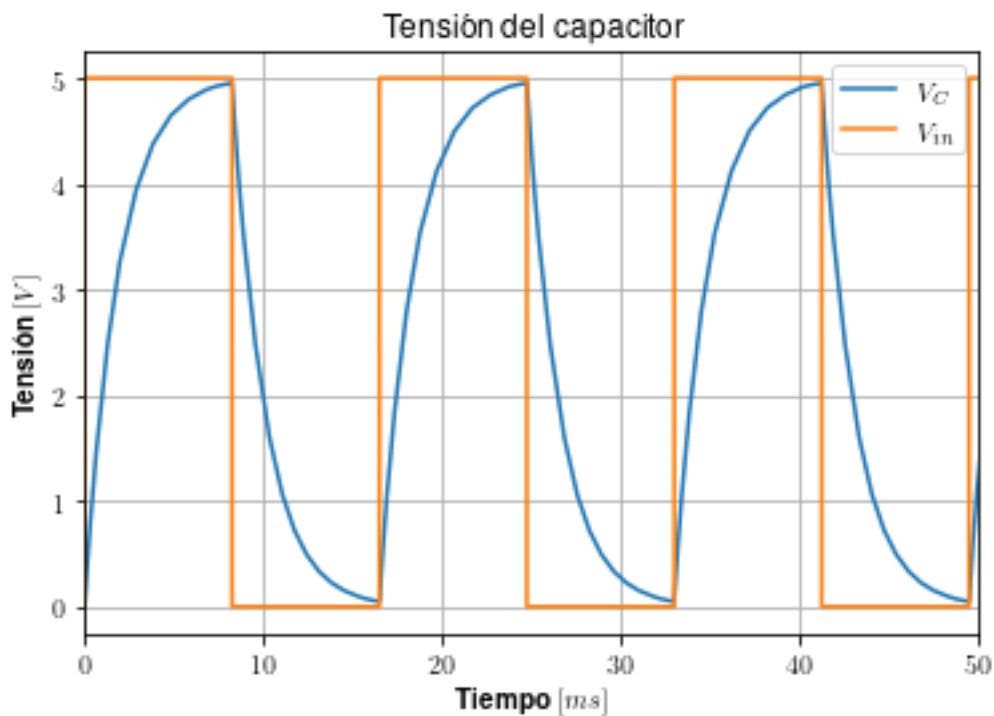


Figura 7: Resultado de la simulación

El código utilizado para generar la figura 7, se muestra a continuación

```
1 # Imports
2 import pandas as pd
3 import matplotlib.pyplot as plt
4
5 # Para poder renderizar Latex
```



```

6 plt.rcParams.update({
7     "text.usetex": True,
8     "font.family": "sans-serif",
9     "font.sans-serif": ["Helvetica"]})
10
11
12
13 data = pd.read_csv(r'TP_Integrador_Simulacion.txt',
14                   sep='\t',
15                   skiprows=1,
16                   names=["time", "Vin", "Vc"])
17
18 t = data.time * 10e2
19 vc = data.Vc
20 vin = data.Vin
21
22
23 plt.figure()
24 plt.plot(t,vc,'-',label=r'$V_C$')
25 plt.plot(t,vin,'-',label=r'$V_{in}$')
26 plt.title(r'Tension del capacitor')
27 plt.xlabel(r'\textbf{Tiempo}  $[ms]$')
28 plt.ylabel(r'\textbf{Tension}  $[V]$')
29 plt.xlim([0,50])
30 plt.legend()
31 plt.grid()
32 plt.show()

```

2.4. Código del programa

A continuación se muestra el código utilizado para la realización del programa

```

1 ;
2 ; TP INTEGRADOR
3 ;
4 ; Autor : Puy Gonzalo
5 ; Padron : 99784
6 ;
7
8 .include "m328pdef.inc"
9 .include "MACROS.inc"
10 .include "DEFINES.inc"
11
12 .DSEG
13 .ORG SRAM_START
14 PERIODO: .BYTE SIZE
15
16 .CSEG
17 .ORG 0x0000
18     JMP     CONFIG
19
20 .ORG INT_VECTORS_SIZE
21
22 CONFIG:

```

```

23      initSP
24      initPORTS
25      initUSART
26      initTimer0
27      initTimer1
28
29 MAIN:
30      CALL    CALCULAR_PERIODO
31      turn_off_Timer1
32      initADC
33 ADC_CONVERT:
34      CALL    ADC_SEND
35      JMP     ADC_CONVERT
36
37
38 .INCLUDE "PERIODO.inc"
39 .INCLUDE "SEND_ADC.INC"

```

Listing 1: main.asm

```

1  ;
2  ; Defines.inc
3  ;
4  ; Autor : Puy Gonzalo
5  ; Padron : 99784
6  ;
7  ; Este archivo contiene .DEF y .EQU utiles para el programa.
8
9  .EQU    BAUDRATEH = 0X00
10 .EQU    BAUDRATEL = 0X67
11 .EQU    SIZE = 2
12 .EQU    MAX_CONTADOR_PERIODO = 3
13 .EQU    OCROB_VALUE = 0x7F
14 .EQU    ADMUX_MASK = 0b01100000
15 .EQU    ADCSRA_MASK = 0b11100111
16 .EQU    ADCSRB_MASK = 0b00000000
17
18
19 .DEF     dummyreg = r16
20 .DEF     contador = r17
21 .DEF     auxL = r18
22 .DEF     auxH = r19
23 .DEF     ADC_DATA = r20
24 .DEF     periodo_L = r21
25 .DEF     periodo_H = r22

```

Listing 2: DEFINES.inc

```

1  ;
2  ; MACROS.inc
3  ;
4  ; Autor : Puy Gonzalo

```

```

5 ; Padron : 99784
6 ;
7 ; Este archivo contiene macros utiles para el programa.
8
9 ; Esta macro inicializa el stack pointer
10 ; Uso:      initSP
11 .MACRO      initSP
12             LDI dummyreg,LOW(RAMEND)
13             OUT spl,dummyreg
14             LDI dummyreg,HIGH(RAMEND)
15             OUT sph,dummyreg
16 .ENDMACRO
17
18 ; Esta macro inicializa el puntero X a una posicion en RAM
19 ; Uso:      initX  <Etiqueta>
20 .MACRO      initX
21             LDI      XH,HIGH(@0)
22             LDI      XL,LOW(@0)
23 .ENDMACRO
24
25 ; ::::::::::: PUERTOS :::::::::::
26
27 ; Esta macro inicializa los puertos correspondientes
28 ; Uso:      initPORTS
29 .MACRO      initPORTS
30 ; Puerto PD5 como salida (OC0B).
31 ; Puerto PB0 como entrada (ICP1).
32 ; Pin PC0 (ADC0) como entrada.
33             LDI dummyreg, (0<<PB0)
34             OUT DDRB,dummyreg
35             LDI dummyreg, (1<<PB0)
36             OUT PORTB,dummyreg
37             LDI dummyreg, (1<<PD5)
38             OUT DDRD, dummyreg
39             LDI dummyreg, (0<<PC0)
40             OUT DDRC,dummyreg
41 .ENDMACRO
42
43 ; ::::::::::: TIMERS :::::::::::
44
45 ; Estas macros configuran el Timer/Counter 1 y el Timer/Counter 0
46
47 ; Timer0: Modo Fast PWM no-invertido (Modo 3) Con prescaler de
48           1024
49 ; Onda generada: f = 50 Hz, Duty Cycle = 50%
50
51 ; Timer1: Modo normal, OC1A y OC1B desactivados. Seteo input
52           capture
53
54 ; Configuracion Timer0
55 ;

```

```

55 .MACRO    initTimer0
56         LDI dummyreg, (0<<COM0A1) | (0<<COM0A0) | (1<<COM0B1) |
          (0<<COM0B0) | (1<<WGM01) | (1<<WGM00)
57         OUT TCCR0A,dummyreg
58         LDI dummyreg, (0<<WGM02) | (1<<CS02) | (0<<CS01) | (1<<
          CS00)
59         OUT TCCR0B,dummyreg
60         LDI dummyreg, OCR0B_VALUE
61         OUT OCR0B,dummyreg
62 .ENDMACRO
63
64 ; Esta macro apaga el Timer0
65 ; Uso: turn_off_Timer0
66
67 .MACRO    turn_off_Timer0
68         LDI      dummyreg,0
69         OUT      TCCR0B,dummyreg
70 .ENDMACRO
71
72 ; Configuracion Timer1
73 ;
74
75 .MACRO    initTimer1
76         LDI dummyreg, (0<<COM1A1) | (0<<COM1A0) | (0<<COM1B1) |
          (0<<COM1B0) | (0<<WGM11) | (0<<WGM10)
77         STS TCCR1A,dummyreg
78         LDI dummyreg,(0<<ICNC1) | (1<<ICES1) | (0<<WGM13) | (0<<
          WGM12) | (0<<CS12) | (1<<CS11) | (0<<CS10)
79         STS TCCR1B,dummyreg
80 .ENDMACRO
81
82 ; Esta macro apaga el Timer1
83 ;
84
85 .MACRO    turn_off_Timer1
86         LDI      dummyreg,0
87         STS      TCCR1B,dummyreg
88 .ENDMACRO
89
90 ; :::::::::: Puerto Serie ::::::::::
91
92 ; Esta macro configura USART0
93 ; Uso: initUSART
94
95 .MACRO    initUSART
96 ;Seteo Baud Rate en 96000 bps
97         LDI dummyreg,BAUDRATEH
98         STS UBRROH,dummyreg
99         LDI dummyreg,BAUDRATEL
100        STS UBRROL,dummyreg
101 ;Activo Transmisor y receptor.
102        LDI dummyreg, (1<<RXEN0) | (1<<TXEN0)

```

```

103         STS UCSROB,dummyreg
104 ; Asincronico, Datos 8N1 (8-bits de datos, sin paridad y 1 bit de
105         stop).
106         LDI dummyreg,6
107         STS UCSROC,dummyreg
108 .ENDMACRO
109 ; ::::::::::::::: ADC :::::::::::::::
110
111 ;Estas macros configuran todo lo correspondiente al ADC
112
113 .MACRO    initADC
114         LDI dummyreg, ADMUX_MASK
115         STS ADMUX,dummyreg
116         LDI dummyreg, ADCSRA_MASK
117         STS ADCSRA,dummyreg
118         LDI dummyreg, ADCSRB_MASK
119         STS ADCSRB, dummyreg
120 .ENDMACRO
121
122 .MACRO    turn_off_ADC
123         LDI dummyreg, 0
124         STS ADCSRA,dummyreg
125 .ENDMACRO

```

Listing 3: MACROS.inc

```

1 ;
2 ; PERIODO.inc
3 ;
4 ; Autor : Puy Gonzalo
5 ; Padron : 99784
6 ;
7 ;
8
9 CALCULAR_PERIODO:
10         CLR      contador
11         initX    PERIODO
12 LOOP_P1:
13         IN        dummyreg,TIFR1
14         SBRS      dummyreg,ICF1
15         JMP       LOOP_P1
16         LDS       auxL,ICR1L
17         LDS       auxH,ICR1H
18         SBI       TIFR1,ICF1      ;Hago clear de ICF1
19 LOOP_P2:
20         IN        dummyreg,TIFR1
21         SBRS      dummyreg,ICF1
22         JMP       LOOP_P2
23         SBI       TIFR1,ICF1
24         LDS       periodo_L,ICR1L ; Guardo en periodo_L el NUEVO
        valor LOW

```

```

25         SUB    periodo_L,auxL
26         ST     X+,periodo_L      ; En PERIODO queda: periodo_L:
           periodo_H
27         LDS    periodo_H,ICR1H ; Guardo en periodo_H el NUEVO
           valor HIGH
28         SBC    periodo_H,auxH
29         ST     X+,periodo_H
30         initX   PERIODO
31
32 SET_NUM:
33         LD      r20,X+
34         INC     contador
35         CPI     contador,MAX_CONTADOR_PERIODO
36         BREQ    RETORNO_CALCULAR_PERIODO
37
38 SEND:
39         LDS     dummyreg, UCSROA
40         SBRS    dummyreg, UDREO
41         JMP     SEND
42
43         STS     UDR0,r20
44         JMP     SET_NUM
45
46 RETORNO_CALCULAR_PERIODO:
47         RET

```

Listing 4: PERIODO.inc

```

1  ;
2  ; SEND_ADC.inc
3  ;
4  ; Autor : Puy Gonzalo
5  ; Padron : 99784
6  ;
7  ;
8
9  ADC_SEND:
10         LDS     dummyreg, UCSROA
11         SBRS    dummyreg, UDREO
12         JMP     ADC_SEND
13
14         LDS     ADC_DATA, ADCH
15         STS     UDR0,r20
16         RET

```

Listing 5: SEND_ADC.inc

El *script* de Python utilizado fue el siguiente

```

1  # Imports
2  import serial
3  import matplotlib.pyplot as plt

```

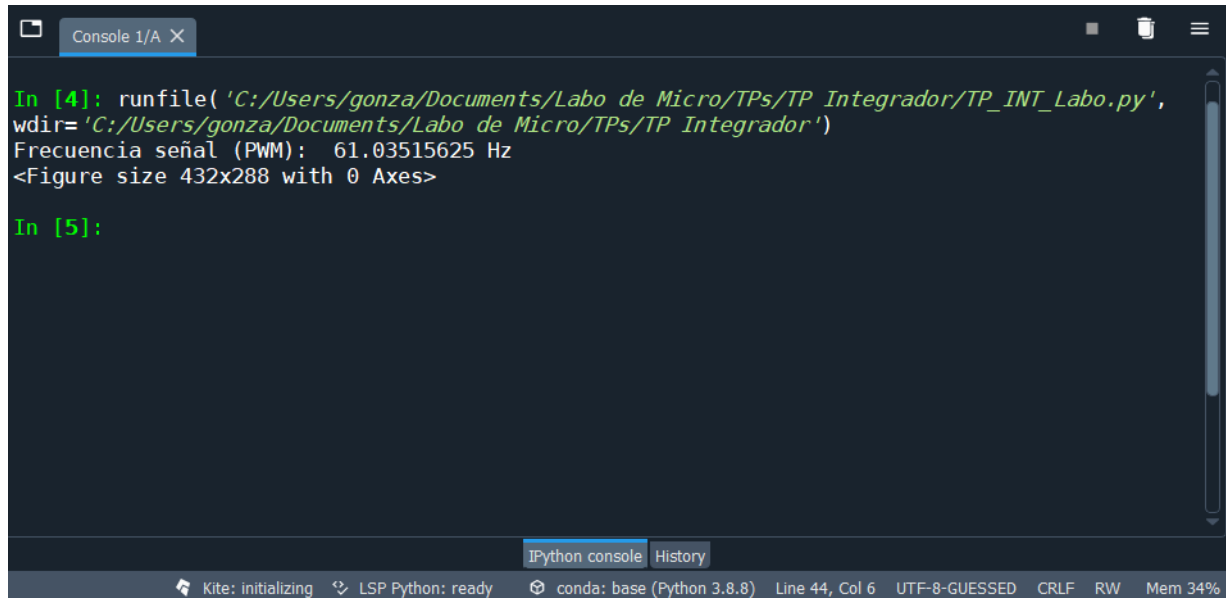
```

4 import numpy as np
5
6 # Declaracion de variables usadas
7 size_Periodo = 2
8 size_ADC = 1
9 f_osc = 16e6
10 N = 8 #Prescaler (Timer1)
11 f_timer = f_osc/N
12 T_timer = 1/f_timer
13 step_size = 19.53e-3
14
15
16 # Abro puerto
17 serial_port = serial.Serial(port = "COM4",
18 baudrate = 9600,
19 parity=serial.PARITY_NONE ,
20 bytesize=serial.EIGHTBITS,
21 stopbits=serial.STOPBITS_ONE,
22 timeout=2
23 )
24
25
26 # Recibo dato y calculo la frecuencia
27 data = serial_port.read(size_Periodo)
28 data =int.from_bytes(data, 'little',signed=False)
29
30
31
32 T = data*T_timer
33 f = 1/T
34
35 print("Frecuencia (PWM): ",f, "Hz")
36
37 # Tension del capacitor (ADC)
38 plt.close('all')
39 plt.figure()
40 plt.ion()
41 plt.show()
42
43 adc_data = np.array([])
44 i = 0
45
46 while i < 60:
47     a = serial_port.read(size_ADC)
48     a = int.from_bytes(a, 'big',signed=False)
49     a = float(a)
50     adc_data = np.append(adc_data, a)
51     plt.cla()
52     plt.plot(adc_data*step_size)
53     plt.title('Tension del capacitor')
54     plt.ylabel('Tension [V]')
55     plt.xlabel('Tiempo')
56     plt.grid()
57     i = i+1
58
59 # Cierro puerto
60 serial_port.close()

```

3. Resultados

Para mostrar los resultados obtenidos se presenta una captura de pantalla de la corrida del programa y luego, el gráfico generado por el *script* de Python



```

In [4]: runfile('C:/Users/gonza/Documents/Labo de Micro/TPs/TP Integrador/TP_INT_Labo.py',
wdir='C:/Users/gonza/Documents/Labo de Micro/TPs/TP Integrador')
Frecuencia señal (PWM): 61.03515625 Hz
<Figure size 432x288 with 0 Axes>

In [5]:
  
```

Figura 8: Corrida del *script* utilizado

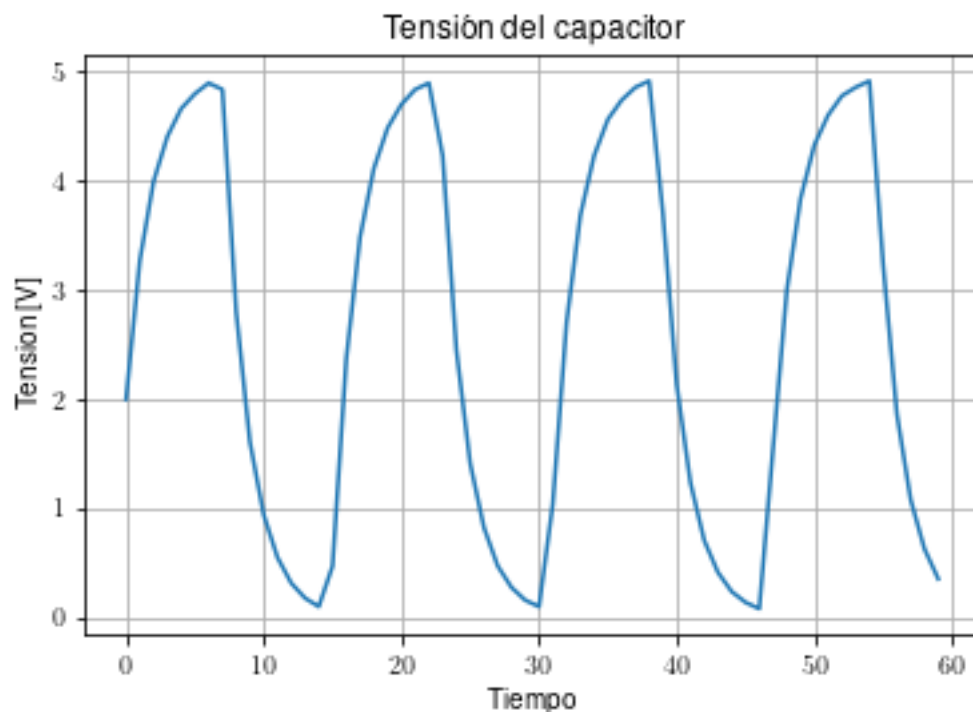


Figura 9: Gráfico obtenido con los valores dados por el conversor

Como se ve en la figura 8, el calculo de la frecuencia de la señal generada resultó correcto.

Por otro lado, en la figura 9 se puede ver una muy buena conversión. Si bien al no contar con prescalers mas grandes la conversión no es del todo precisa, se considera suficiente.

4. Conclusiones

Finalmente se puede concluir que se logró entender el funcionamiento del conversor analógico digital que ofrecen los microprocesadores de la familia AVR y las diferentes configuraciones que estos tienen.

Por otro lado, se pudo observar otra función de los Timers/Counters que no se había visto en trabajos anteriores de la asignatura, como fue el caso del modo de captura. Este modo resulta útil, no solo para el cálculo de periodos de señales externas sino también para registrar el tiempo de arribo de un evento externo.

Otro aspecto importante a destacar, es la poderosa función de esta familia de microcontroladores de generar señales.