



Laboratorio de Microcomputadoras - 86.07

## Capacímetro

Profesor:	Ing. Guillermo Campiglio								
Cuatrimestre/Año:	1º/2016								
Turno de las clases prácticas	miercoles 19 hs								
Jefe de trabajos prácticos:	Ing. Ricardo Arias								
Docente guía:	Gavinowich Gabriel Hugo								
Autores		Seguimiento del proyecto							
Nombre	Apellido	Padrón							
Marcelo	Sanchez	87685							

### Observaciones:

.....  
.....  
.....  
.....  
.....  
.....  
.....  
.....  
.....  
.....  
.....

Fecha de aprobación		Firma J.T.P

Coloquio
Nota final
Firma profesor

# Índice

<b>1. Introducción (Proyecto original)</b>	<b>3</b>
<b>2. Objetivos propuestos</b>	<b>4</b>
<b>3. Desarrollo</b>	<b>4</b>
3.1. Diagrama en bloques . . . . .	5
3.2. Diagrama de flujo . . . . .	6
<b>4. Descripción del Hardware</b>	<b>6</b>
4.1. Circuito analógico . . . . .	7
4.2. Microcontrolador . . . . .	9
<b>5. Descripción del Software</b>	<b>9</b>
<b>6. Objetivos logrados</b>	<b>10</b>
<b>7. Capacímetro (Proyecto definitivo)</b>	<b>11</b>
<b>8. Conclusiones y posibles mejoras</b>	<b>12</b>
<b>9. Fotos</b>	<b>14</b>
<b>10. Código</b>	<b>17</b>
<b>11. Bibliografía y recursos</b>	<b>29</b>
<b>12. Apéndice</b>	<b>29</b>

## 1. Introducción (Proyecto original)

### **Localizador de pares de cobre simétricos.**

En el mundo de las telecomunicaciones, la telefonía fija es uno de los servicios mas contratado por la comunidad. Para responder a esta demanda, las empresas despliegan una red de miles de kilómetros de cables de cobre. Una de las consecuencias directas de este despliegue, es que se requiere de un mantenimiento constante de la red.

En este contexto, se busca diseñar un circuito que permita detectar fallas de continuidad en pares de cobre simétricos, para facilitar el trabajo del personal técnico. El mismo debe cumplir con especificaciones básicas tales como: ser transportable, poseer una manipulación y lectura sencilla de resultados, buena velocidad de respuesta y larga durabilidad de la batería para poder cubrir una jornada completa de trabajo.

## 2. Objetivos propuestos

El objetivo principal de este proyecto es poder medir la longitud del par telefónico en un rango que va desde 0 a 1020 metros aproximadamente. Y de ser posible, autocalibrar los parámetros para medir las longitudes mas cortas, que pueden ir desde los 0 a 200 metros aproximadamente. Se desea que el microcontrolador pueda discriminar estos rangos para autoescalar la medición y modificar los parámetros necesarios para una mayor precisión.

## 3. Desarrollo

Para cumplir con el objetivo, se parte de las características físicas del cable. Sabiendo que, por construcción, presenta una capacidad conocida por kilómetro, se diseñará un circuito capaz de medir capacidad, y finalmente se la convertirá en la distancia real del cable. Los resultados se presentarán en una pantalla LCD para su fácil visualización. Como se trata de capacidades pequeñas ( $52 \text{ nF/Km}$ ), se utilizará una técnica que combina un circuito analógico y la capacidad de procesamiento del microcontrolador. El circuito analógico, funciona básicamente con una onda cuadrada como señal de entrada (proporcionada por el  $\mu\text{C}$ ), la que carga y descarga el cable (su capacidad asociada) en forma sucesiva. Y a través de un circuito configurado como integrador, testea las corrientes de carga del mismo, sumándolas en el tiempo. EL resultado es un nivel de tensión a la salida para cada valor de capacidad.

El microcontrolador toma el nivel de tensión y lo convierte en un dato digital gracias al módulo ADC incorporado. Luego es procesado para poder ser mostrado en una pantalla LCD. Ver figura 1.

### 3.1. Diagrama en bloques

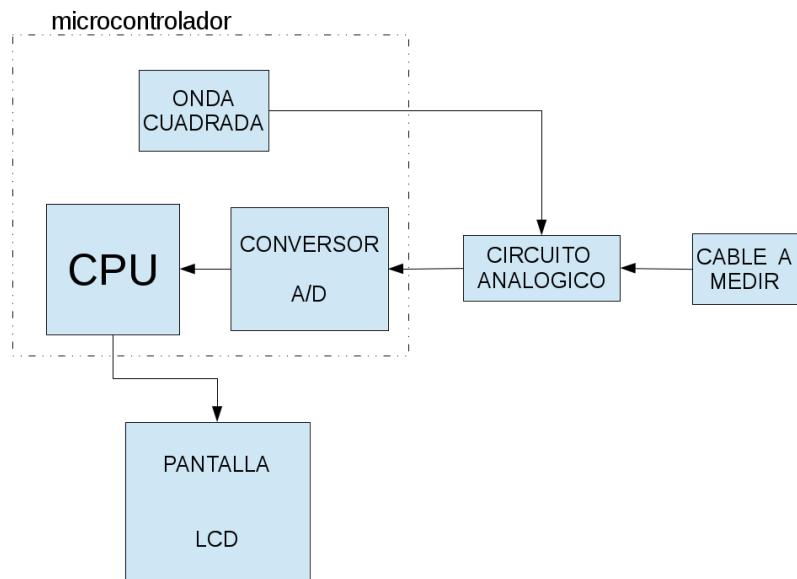


Figura 1: Diagrama en bloques.

### 3.2. Diagrama de flujo

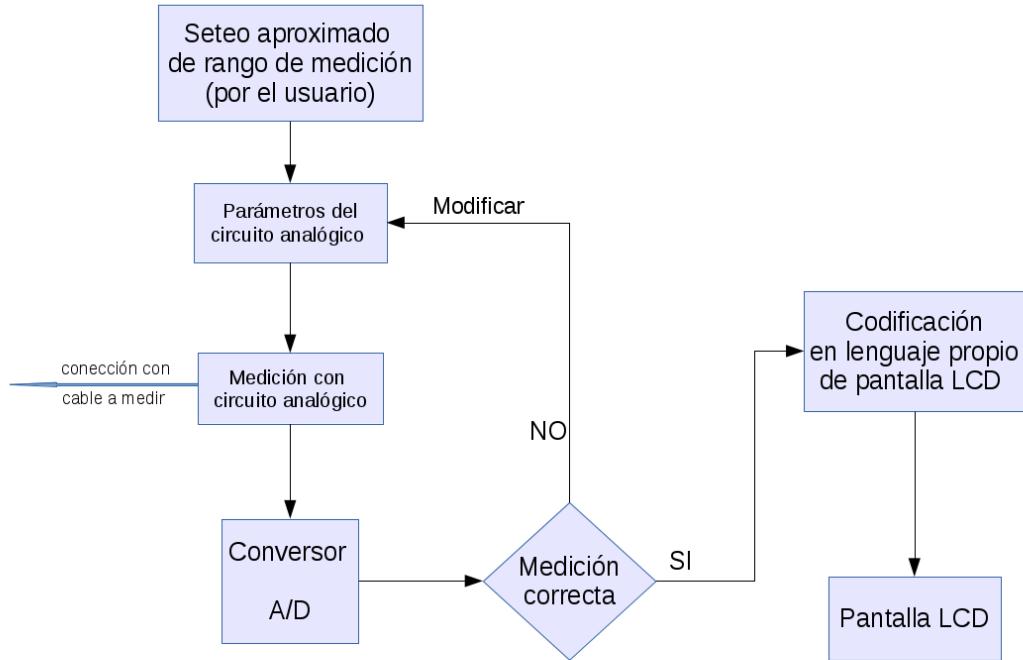


Figura 2: Diagrama de flujo.

## 4. Descripción del Hardware

Se explica a continuación el funcionamiento de cada parte del circuito.

### 4.1. Circuito analógico

En el circuito de la figura 3, la corriente de carga fluye a través de  $R_1$ , y se filtra y se amplifica por el amplificador  $U_2A$  diferencial. La descarga se realiza a través de  $D_2$ . La corriente media en  $R_1$  es proporcional a la capacitancia.

Para la salida de 5V, si  $AV$  es la ganancia del amplificador, la capacidad máxima a medir es:

$$C = 5V / (5,74V * AV * R_1 * 455,000Hz) \quad (1)$$

En este caso,  $Av = 10$ , por lo que  $C = 128pF$ . Para valores inferiores, simplemente aumentar el valor de  $R_8$ .

Para el proyecto se adaptaron los valores para conseguir una tensión por cada longitud medida, para un rango de 100 a 1020 mts. Y a partir de una hoja de datos de pares telefónicos mostrados en la figura , se decidió modelar el circuito como una resistencia en serie con una capacidad asociada por kilómetro. Con este modelo se obtuvieron los siguientes valores de componentes:

- $D_1 = D_2 = 1N4148$

Es fundamental tener diodos rápidos en esta parte, sino la proporción de corriente inversa al conmutar la señal lo suma el amplificador diferencial. Disminuyendo la precision de la medición.

- $R_1 = R_2 = 10\Omega$

Como el cable tendrá una resistencia en serie, el cual se sumará a  $R_1$ , el resultado para la longitud de 1020 mts será de  $53nF$  y  $91\Omega + R_1 \approx 101\Omega$ . Para estos valores se adoptó un periodo de  $160\mu F$  para la onda cuadrada.

- $C_1 = C_2 = 220nF$

Mediante simulación estos valores son los que resultaron con menor ripple a la salida del circuito.

- $R_4 = R_5 = 100k\Omega$  ; Los valores son los presentes en el original.

- $R_3 = R_6 = 1M\Omega$

- $R_7 = 100\Omega$

- $C_1 = C_2 = 220nF$

- $C_3 = 10nF$

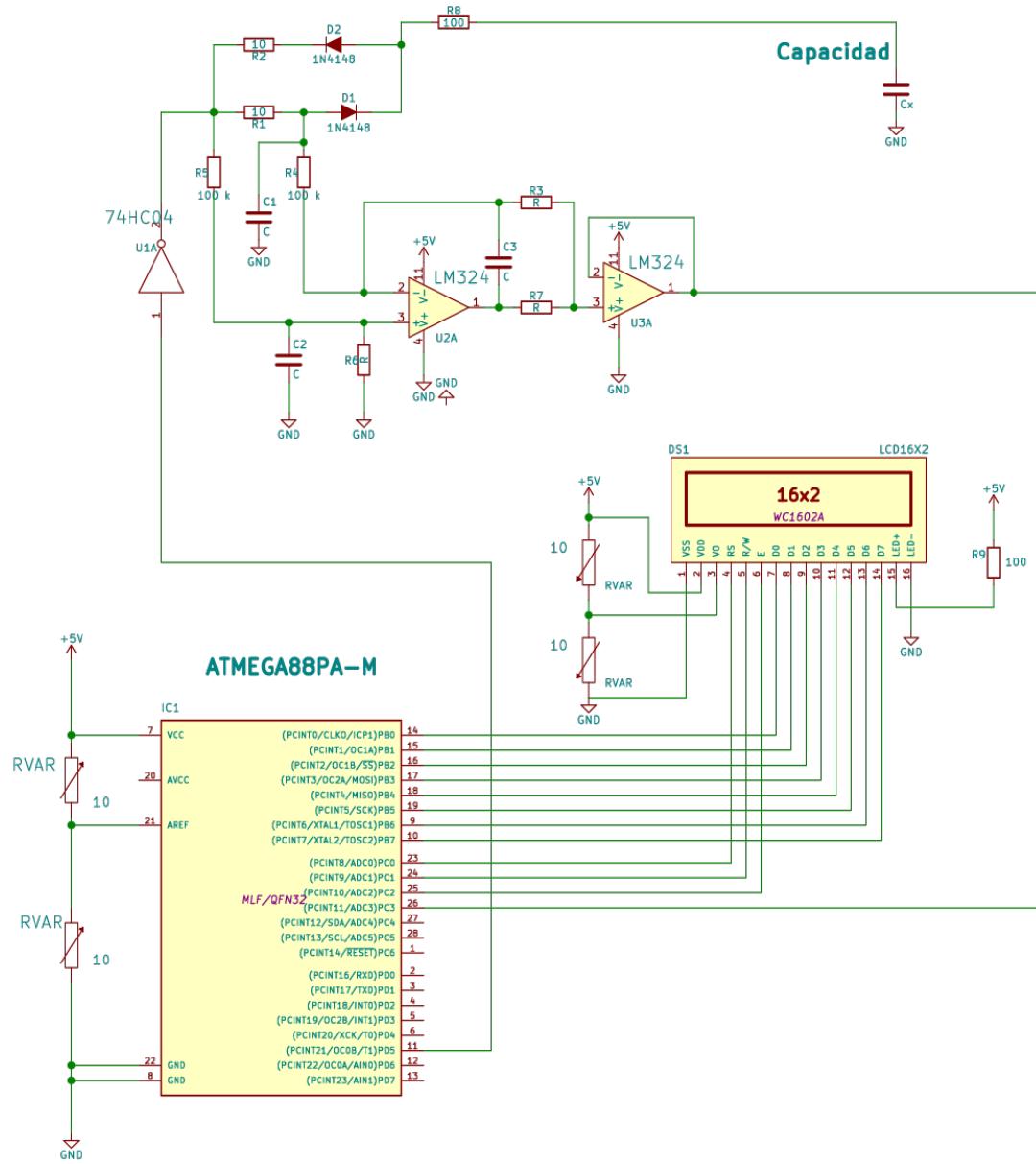


Figura 3: Esquemático capacitímetro

## 4.2. Microcontrolador

Se utilizó el  $\mu C$  Atmega88PA. La función del mismo se puede dividir en 4 partes (ver figura 1):

1. Onda cuadrada.

Genera una onda cuadrada utilizando el Timer O del  $\mu C$ , para el funcionamiento del circuito analógico.

2. Módulo ADC.

Para convertir la tensión proporcionada por el capacímetro analógico antes descrito, y poder procesarlo.

3. Procesamiento.

Se utiliza principalmente un promediador para eliminar los ruidos en la entrada del ADC. Se explica con detalle en la descripción del Software.

4. Pantalla LCD.

Para una lectura clara los resultados se muestran en una pantalla LCD de 16 X 2, utilizando los puertos C y D del  $\mu C$ .

## 5. Descripción del Software

El programa funciona siguiendo el diagrama de flujo que se muestra en la figura 4

El promedio se hace 2 veces y se comparan para poder discernir entre una tensión transitoria de una permanente en el pin del ADC, ya que el circuito analógico tiene un tiempo necesario para establecer esta tensión. Con éste método el programa es independiente de los tiempos necesarios por cada medición para lograr el régimen permanente al que se quiere llegar.

Luego tiene otras etapas de decisión como por ejemplo para determinar si la longitud es mayor a 1020 , que sería el rango máximo de diseño. Como también verificar si es menor a 200 metros para ajustar el Período de la onda cuadrada y así tener mayor precisión en este rango.

Se incorporan además subrutinas de Delays para los tiempos de espera del LCD, de escritura con frases guardadas en la memoria ROM y otras como el conversor del ADC a código ASCII para enviar a la pantalla.

En el diagrama se ve que el dispositivo está constantemente midiendo. Con ésta configuración se puede ver si la medición es correcta o varía en el tiempo, para verificar el método utilizado. Como también es útil para cambiar de "par", (si se

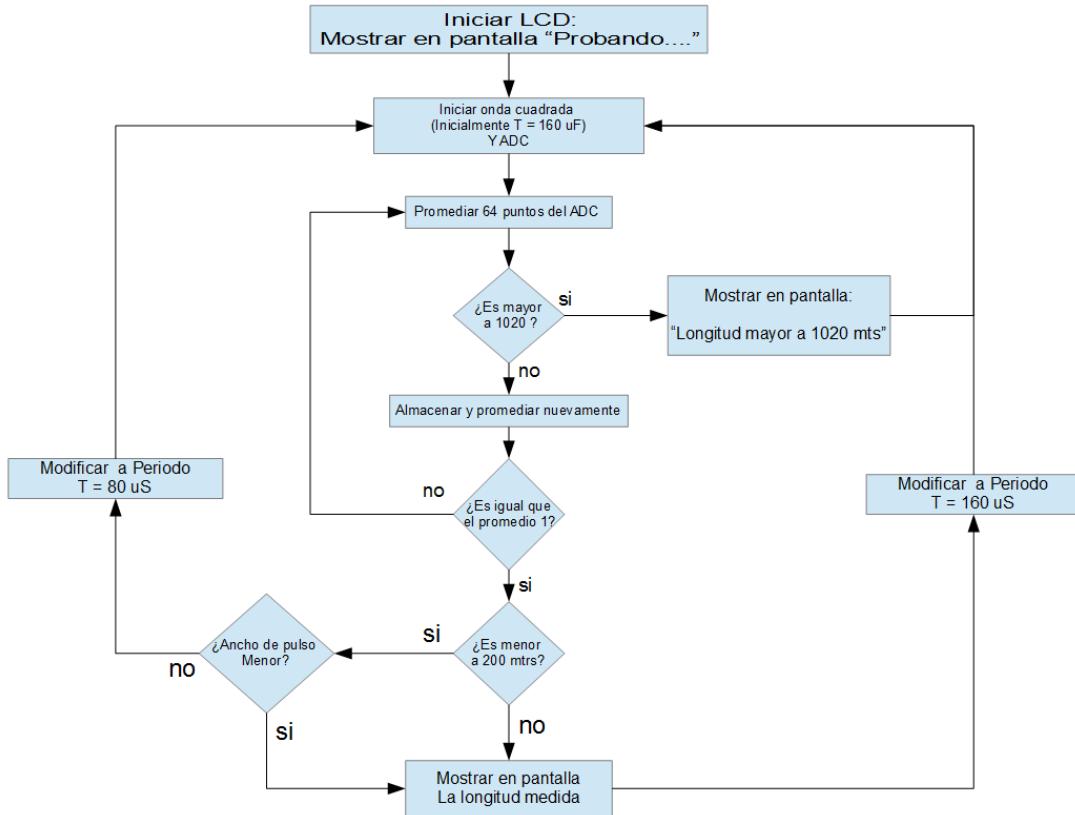


Figura 4: Diagrama de flujo del código.

trabaja con cables multipares), para comparar longitudes de una manera ágil. A ésta forma se la suele denominar "Medición en tiempo real".

## 6. Objetivos logrados

Utilizando el modelo teórico descrito en la sección 4 se simuló y probó el circuito reemplazando el cable por componentes concentrados (una resistencia y un capacitor). Los resultados fueron satisfactorios. Como se pude ver en la figura 5 , las tensiones se modifican linealmente con la distancia teórica.

Luego se procedió a medir en forma práctica, en un cable real. El resultado del mismo no fue satisfactorio, ya que la medición no arrojó resultados precisos comparándola con las mediciones teóricas del modelo y verificando el resultado con el localizador de fallas "Dynantel 965MC". La medición obtenida se comportó de manera aleatoria. Este resultado, podría entenderse observando que en el momento de las pruebas, se decidió agregar al circuito una resistencia en serie con el cable,

lo que provocó inmediatamente la estabilidad de la medición, pero sin arrojar resultados correctos aún calibrando el dispositivo. Lo que muestra una discrepancia con el modelo utilizado. Al notar este inconveniente, se decidió avanzar el proyecto en un **medidor de capacidad**, ya que los parámetros pueden ajustarse de manera más precisa para lograr resultados correctos.

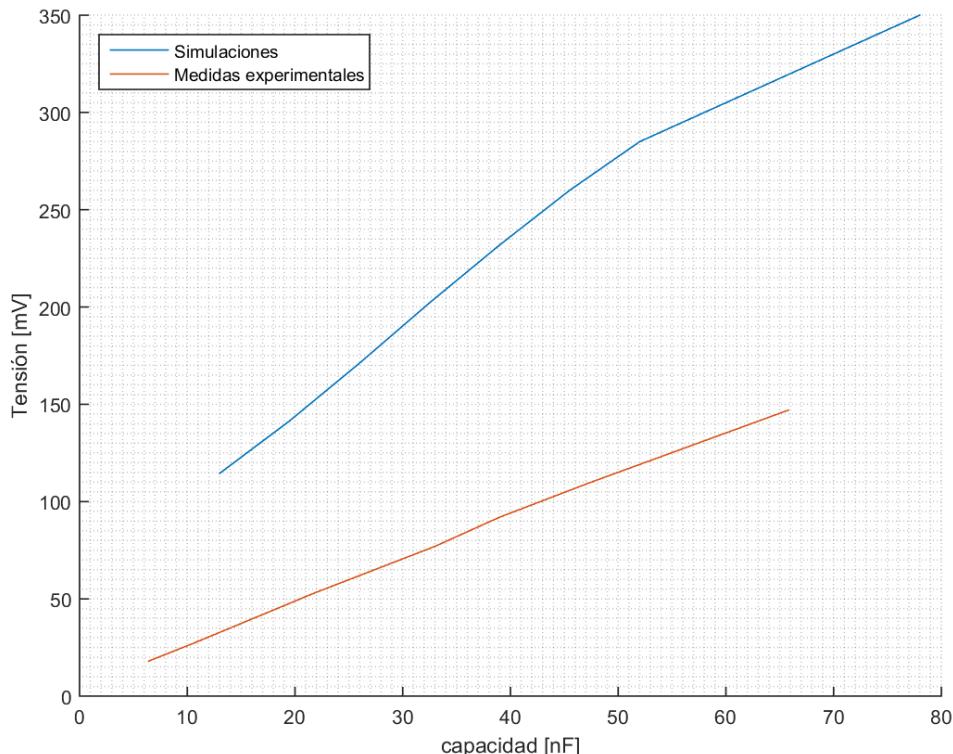


Figura 5: Grafico Tensión-Capacidad medida ( $T = 160$  ms).

## 7. Capacímetro (Proyecto definitivo)

Avanzando entonces el proyecto como un medidor de capacitancia, se modificaron algunos parámetros. Básicamente en vez de mostrar en pantalla la longitud se muestra la capacidad medida y para no modificar el circuito se trabajó en el rango de capacidades que se esperaba medir con los pares de cobre, quedando los rangos de 1 nF a 50 nF. Para ello se agregó una resistencia al circuito original en serie con la capacidad a medir de  $80\Omega$ . El diagrama del código quedó definido entonces por la figura 6

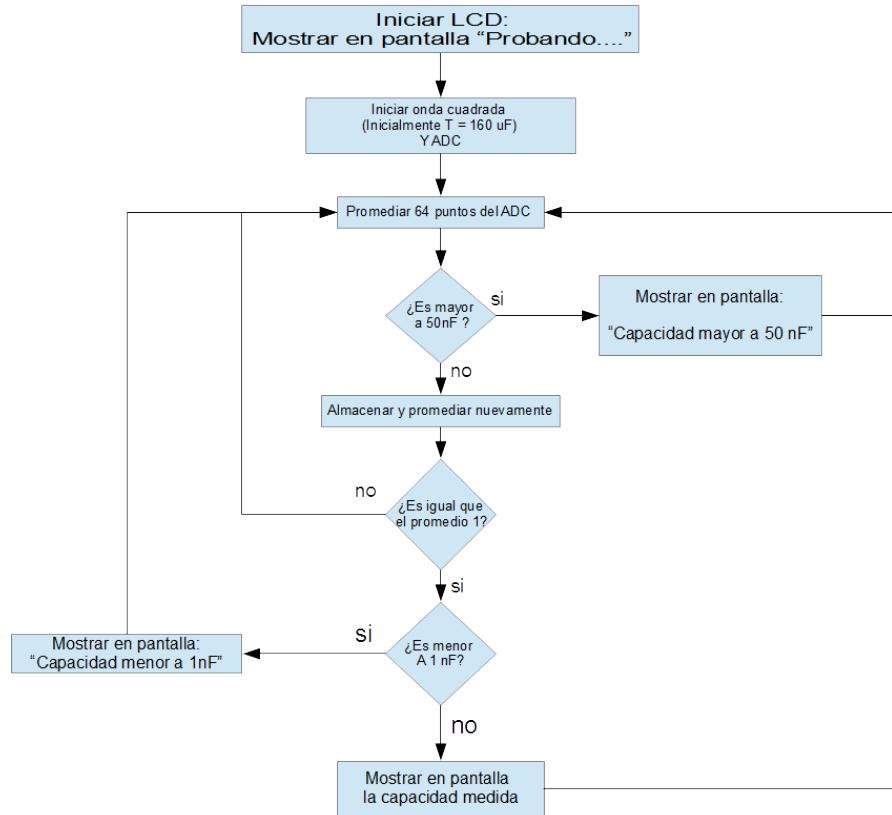


Figura 6: Diagrama de flujo del código definitivo.

## 8. Conclusiones y posibles mejoras

El diseño original del proyecto se basó en un circuito analógico que mide capacidades pequeñas. Aprovechando esta sensibilidad se adaptó el mismo al modelo de par telefónico descrito en la sección 4. Lamentablemente el cable no concordaba con el modelo propuesto. Resultando en una medición incorrecta con el método propuesto.

Se concluye con esta experiencia que se debe investigar y determinar las limitaciones del modelo a utilizar en cualquier fenómeno físico. En particular con este proyecto, los inconvenientes surgieron al simplificar una línea de transmisión en solo 2 componentes concentrados.

Una posible mejora podría obtenerse sin modificar la estructura del diseño, reemplazando la resistencia  $R_1$  por un valor cercano a  $100\Omega$ . Luego medir experimentalmente distintas longitudes, graficar los puntos longitud - tensión ADC. Y con ello obtener una función para incorporar al micro y ajustarlo correctamente para mostrar finalmente la verdadera medida.

Con respecto al Capacímetro, los resultados fueron satisfactorios. La curva Capacidad-Promedio que se muestra en la figura 7, se obtuvo de las mediciones experimentales, tomando como referencia las mediciones realizadas con el Multímetro *BK PRECISION 2712*. En el mismo se ve el comportamiento lineal de la medición. En la próxima sección se muestran algunas fotos del proyecto terminado.

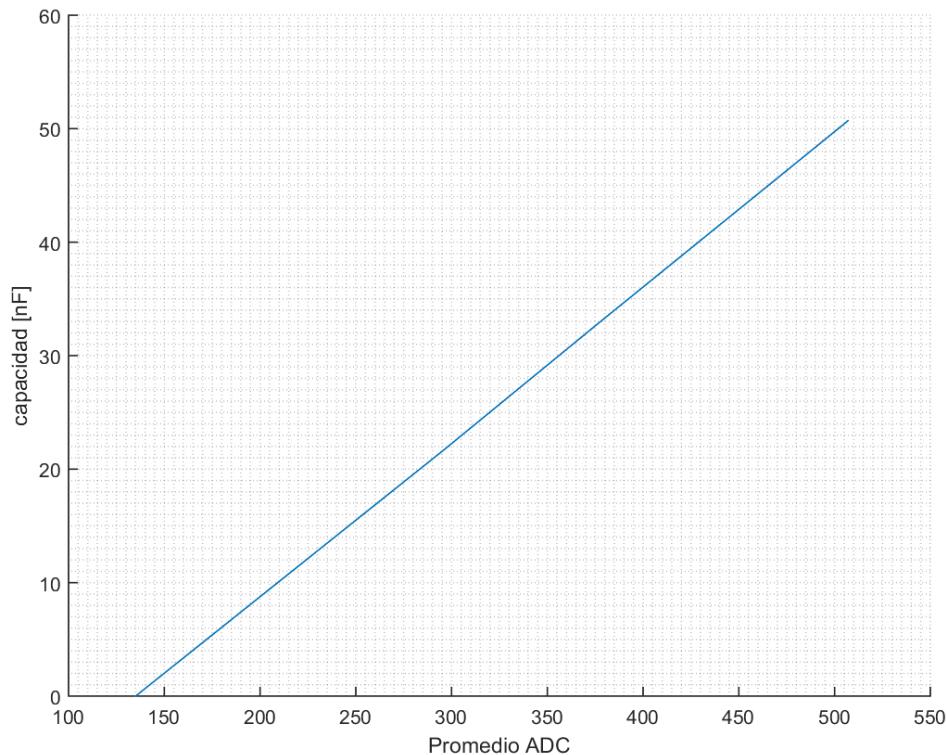


Figura 7: Curva experimental promedio ADC vs Capacidad medida.

## 9. Fotos

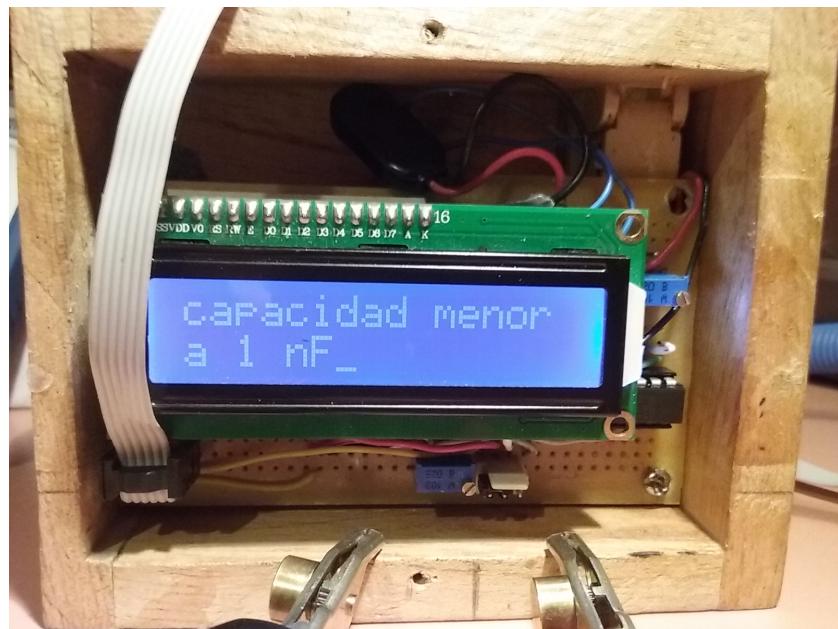


Figura 8: Foto 1. Capacímetro desconectado



Figura 9: Foto 2. Midiendo un capacitor de poliéster de 3 nF



Figura 10: Foto 3. Midiendo un capacitor de poliéster de 22 nF



Figura 11: Foto 4. Midiendo un capacitor de poliéster de 33 nF



Figura 12: Foto 5. Midiendo un capacitor de poliéster de 47 nF



Figura 13: Foto 6. Midiendo un capacitor de poliéster de 68 nF

## 10. Código

```

/*
 * Capacimetro.asm
 *
 * Created: 29/06/2016 06:43:24 p.m.
 * Author: MarceloFernando
 */

; .include "m88PAdef.inc"           ;comento ésta linea porque
; Atmel Studio 6 incluye la librería
;
; Declaro constantes para identificar los puertos utilizados
; por el LCD
;
.equ      lcd_dprt = PORTB ;puerto de datos del lcd
.equ      lcd_ddr = DDRB ;puerto ddr del lcd
.equ      lcd_dpin = PINB ;puerto pin del lcd
.equ      lcd_cprt = PORTC ;puerto de comandos del lcd
.equ      lcd_cddr = DDRC ;puero de comandos ddr del lcd
.equ      lcd_cpin = PINC ;puerto de comandos pin del lcd
.equ      rs = 0            ;registro rs del lcd
.equ      rw = 1            ;registro r/w del lcd
.equ      en = 2            ;registro en del lcd

;Otras constantes
;
.equ      offset = 184
.equ      minimo = (1 * 10 + offset)*64/85 ;defino el mínimo

;
;Defino unas variables que voy a utilizar para promediar
;y convertir los valores leídos del adc.
;
.def      u_mil = r18
.def      centena = r19
.def      decena = r20
.def      unidad = r21

.def      prom_high = r22
.def      prom_low = r23

;Se abre un segmento de datos para definir variables
.dseg
;.org    0x100
adc_1 :     .byte 128          ;vector para guardar 64 valores
                           ;de 2 bytes

;Segmento de código
.cseg
rjmp main

```

```

.org int_vectors_size ; comienzo después de las interrupciones.

main:
    ldi r16, low(ramend) ; Inicializo el
    out spl, r16           ; Stack Pointer
    ldi r16, high(ramend)
    out sph, r16

    rcall ini_lcd          ; inicio el LCD

    ldi zl, low(MJE0<<1) ; "Iniciando ----"
    ldi zh, high(MJE0<<1) ; en pantalla
    rcall enviar_frase
    rcall delay_5seg

    ldi r18, 80             ; Comienzo la onda cuadrada con un
medicion :
    rcall ini_onda_cuad

    rcall ini_ADC

    ldi r16, 0X01           ; Borro la pantalla
    rcall enviar_com

    ldi zl, low(MJE1<<1) ; envío "Probando...." a la pantalla
    ldi zh, high(MJE1<<1)
    rcall enviar_frase
    rcall delay_5seg

medir : rcall grabar_y_promediar
; el promedio me devuelve los valores en prom_high prom_low

        cpi prom_high, 2 ; si el valor es mayor a 50nF =
; 500 en el adc envio un mje en pantalla
        brcc mayor_a_50n
        cpi prom_high, 1
        brcs comparacion_2
        cpi prom_low, 244
        brcc mayor_a_50n
        rjmp rango_ok

comparacion_2 :
        cpi prom_low, minimo ; en esta parte comparo
        brcs longitud_menor_1n

rango_ok :
        rcall escalar

        rcall convertir_a_longitud ; la subrutina toma los valores

```

;El resultado queda guardado en las variables u\_mil centena decena ;unidad

```

ldi r16,0X01           ; Borro la pantalla
rcall enviar_com

ldi zl, low(MJE5<<1)    ; envio mje "Longitud = " a lcd
ldi zh, high(MJE5<<1)
rcall enviar_frase

; si r20 > 1022 -----> mostrar MJE8.
; o si r20 < 20 -----> cambiar ancho de pulso a T=160ms
; sino convertir volts medidos en distancia del cable y

ldi r16,0XC0           ; Comienzo en la segunda linea
rcall enviar_com

mov r16, u_mil          ; muestro en pantalla el
mov r16, centena
rcall enviar_dato
mov r16, decena
rcall enviar_dato
ldi r16,'.'
rcall enviar_dato
mov r16, unidad
rcall enviar_dato

ldi zl, low(MJE6<<1)    ; envío " nF" a la pantalla
ldi zh, high(MJE6<<1)
rcall enviar_frase

rjmp medir

```

mayor\_a\_50n :

```

ldi r16,0X01           ; Borro la pantalla
rcall enviar_com
ldi zl, low(MJE8<<1)    ; envío "mayor a 50 nF" a la
rcall enviar_frase
rjmp medir

```

longitud\_menor\_1n :

```

ldi r16,0X01           ; Borro la pantalla
rcall enviar_com
ldi zl, low(MJE9<<1)    ; envío "menor a 1 nF" a la
ldi zh, high(MJE9<<1)    ; pantalla
rcall enviar_frase
rjmp medir

```

#####

```

;Rutinas para iniciar el LCD
#####
;-----;
;-----;

ini_lcd :
    ldi r21 , 0xFF
    out lcd_dddr , r21      ; configuro los puertos de datos como sa
    out lcd_cddr , r21      ; configuro los puertos de comando
    cbi lcd_cprt , en       ; en = 0 para luego grabar
    rcall delay_10ms         ; Espero a que se prenda el LCD

    ldi r16 ,0X38            ; Inicio al LCD en 2 lineas , matriz

    ;Prendo el Display , Prendo el cursor
    ldi r16 ,0X0E
    rcall enviar_com

    ;Borro la pantalla
    ldi r16 ,0X01
    rcall enviar_com

    ;Corro el cursor hacia la derecha
    ldi r16 ,0X06
    rcall enviar_com
    ret

;-----;
;rutinas para enviar datos o comandos al LCD
;-----;

enviar_frase :
    ;ldi r16 ,0X01           ; Borro la pantalla
    ;rcall enviar_com
    /*ldi r16 ,0X06           ; Corro el cursor hacia la derecha
    rcall enviar_com*/
    ldi r17 , 0

ciclo_subr1 :
    inc r17
    cpi r17 , 17
    breq linea_2
    lpm r16 , z+
    cpi r16 ,0
    breq salir_subr1
    rcall enviar_dato
    rjmp ciclo_subr1

linea_2 :
    ldi r16 ,0XC0            ; Comienzo en la segunda linea
    rcall enviar_com
    rcall ciclo_subr1

salir_subr1 :
    ret

enviar_com:
    cbi lcd_cprt , rs        ; rs=0 para mandar comandos
    cbi lcd_cprt , rw        ; rw=0 para escribir en el LCD

```

```

        sbi lcd_cprt, en           ;E=1
        out lcd_dpdt, r16
        rcall sdelay
        cbi lcd_cpdt, en          ;E=0 es el flanco descendente
        rcall delay_2ms
        ret
    
```

## enviar\_dato:

```

        sbi lcd_cpdt, rs          ;rs=1 para mandar datos
        cbi lcd_cpdt, rw          ;rw=0 para escribir en el LCD
        sbi lcd_cpdt, en          ;E=1
        out lcd_dpdt, r16
        rcall sdelay
        cbi lcd_cpdt, en          ;E=0 ;Es el flanco descendente (High)
        ;Low)
        rcall delay_2ms
        ret
    
```

```

#####
; Delays
#####
    
```

## delay\_40us :

```

        push r16                  ; [2] + rcall [3] = 5
        ldi r16, 18               ; [1] --> ciclos = 6
    
```

## ciclo\_40us :

```

        dec r16                   ; [1]*18 --> ciclos = 24
        brne ciclo_40us ; [1/2]*18 --> ciclos = 33
        pop r16                  ; [2] --> ciclos = 35
        nop                      ; [1] --> ciclos = 36
        ret                      ; [4] --> ciclos = 40, para un clk de 1 Mhz ;
        ; 40us
    
```

## sdelay: nop

```

        nop
        ret
    
```

## delay\_100us:

```

        push r16                  ; [2] + rcall [3] = 5
        ldi r16, 58               ; [1] --> ciclos = 6
    
```

## ciclo\_100us :

```

        dec r16                   ; [1]*58 --> ciclos = 64
        brne ciclo_100us ; [1/2]*58 --> ciclos = 93
        pop r16                  ; [2] --> ciclos = 95
        nop                      ; [1] --> ciclos = 96
        ret                      ; [4] --> ciclos = 100, con un
        ; = 100us
    
```

## delay\_2ms :

```

        push r16          ; [2] + [3] del rcall = [5]
        push r17          ; [2] —> ciclos = 7
        ldi r17,24        ; [1] —> ciclos = 8
ciclo1_2ms :
        ldi r16,22        ; [1]*24*33 —>ciclos = 800
ciclo2_2ms :
        dec r16          ; [1]*22
        brne ciclo2_2ms ; [1/2]*22 = subtotal = 33
        dec r17          ; [1]*24*33 —>ciclos = 1592
        brne ciclo1_2ms ; [1/2]*24*33 —>ciclos = 1988
        nop
        nop
        nop
        nop          ; [4] —>ciclos = 1992
        pop r17          ; [2] —>ciclos = 1994
        pop r16          ; [2] —>ciclos = 1996
        ret          ; [4] —>ciclos = 2000, para un
        ; delay = 2,00 ms

delay_10ms :
        push r16
        push r17
        ldi r17,22
ciclo1_10ms :
        ldi r16,121
ciclo2_10ms :
        dec r16
        brne ciclo2_10ms
        dec r17
        brne ciclo1_10ms
        nop
        nop
        pop r17
        pop r16
        ret

delay_100ms :
        ldi r16, 20      ; [1]
ciclo_100ms :
        rcall delay_2ms ; 2ms*20
        dec r16          ; 2ms*20
        brne ciclo_100ms ; 2ms*10 —> aprox delay (20+20)
        ;= 100 ms
        ret

delay_5seg :
        ldi r17, 25

ciclo_dy5s :
        rcall delay_100ms
        dec r17
        brne ciclo_dy5s
        ret

```

```

#####
;Rutina para iniciar onda cuadrada
#####
;Genera una onda cuadrada con periodo T = 2*r18 .

ini_onda_cuad :
    ldi r16, 0x00           ; configuro el PORTD.6 como salida
    ori r16, (1<<PD6)
    out DDRD, r16
    ldi r17, 0x00
    out PORTD, r17          ; comienzo la señal con 0

    ; para generar ciclos de 40 us
    out OCR0A, r18          ; configuro el período con el valor
    ; guardado en r18

    ; modo CTC(toogle) + fast PWM
    in r17, TCCR0A
    ori r17, ((1<<COM0A0)|(1<<WGM01)|(1<<WGM00))
    andi r17, ~(1<<COM0A1)
    out TCCR0A, r17          ; TCCR0A = 0b01xxxx11

    ; modo fast PWM, sin preescaler
    in r17, TCCR0B
    ori r17, ((1<<WGM02)|(1<<CS00))
    andi r17, ~(((1<<CS02)|(1<<CS01)))
    out TCCR0B, r17          ; TCCR0B = xxxx1010

    ret

#####
;Rutina para iniciar el conversor ADC en modo Free Runing, sin habilitar ,int
#####

ini_ADC :           ;ADMUX – ADC Multiplexer Selection Register
;Ref externa , registros guardados a derecha , selecciono ADC3 =
;pin26 del micro
    lds r16, ADMUX
    ori r16, (1<<MUX1|1<<MUX0)
    andi r16, ~(((1<<REFS1)|(1<<REFS0)|(1<<ADLAR)|(1<<MUX3)|(1<<MUL))
    sts ADMUX, r16           ; ADMUX = 00000011

    ;ADCSRA – ADC Control and Status Register A
    ;ADC on , conversión on , Trigger on , Interrupción ADC off ,
    lds r17, ADCSRA
    ori r17, ((1<<ADEN)|(1<<ADSC)|(1<<ADATE)|(1<<ADPS2)|(1<<ADPS1))
    andi r17, ~(((1<<ADIF)|(1<<ADIE)))
    sts ADCSRA, r17          ; ADCSRA = 11100000

    ;ADCSR – ADC Control and Status Register B
    ;Configuro el ADC en modo "Free Runing" , aunque ya este por
    ; default
    ldi r16, 0x00

```

```

sts adcsrb , r16 ; adcsrb = 00000000

; DIDR0 – Digital Input Disable Register 0
; desactivo los Buffers de los pines ADC que no utilizo para
; ahorrar energía , dejo solo el ADC4
; lds r17 , didr0
; ori r17 , ((1<<ADC5D)|(1<<ADC3D)|(1<<ADC2D)|(1<<ADC1D)
; |(1<<ADC0D))
; andi r17 , ~(1<<ADC4D)
; sts didr0 , r17

ret

/*#####
Rutina para guardar las tensiones obtenidas por el ADC y promediarlas
#####*/
;La rutina graba 64 tensiones en la variable adc_1 obtenidas del adc ,
;el mismo tiene que estar configurado como "Free Runing".
;Luego las promedia y vuelve a repetir el proceso. Se comparan
;entonces los 2 promedios obtenidos. Si son iguales sale de la
;subrutina y devuelve el resultado en los registros definidos como
;prom_high y prom_low.

grabar_y_promediar :
    ldi r20 , 0x00
    ldi r21 , 0x00
comenzar_muestreo :
    ldi xl , low(adc_1) ; Apunto la variable x e y al vector
    ; adc_1
    ldi xh , high(adc_1)
    ldi r16 , 64

grabar_adc1 :
    lds r17 , ADCL ; guardo los valores obtenidos por el
    ; ADC en vector
    lds r18 , ADCH
    st x+, r17
    st x+, r18

    rcall delay_2ms ; espero 4 ms para volver a guardar
    rcall delay_2ms ; un valor
    dec r16
    brne grabar_adc1

promedio_1 :
    ldi xl , low(adc_1) ; Apunto la variable x al vector
    ; adc_1
    ldi xh , high(adc_1) ; para sumar los datos almacenados
    ldi r22 , 0x00 ; inicializo las variables
    ldi r23 , 0x00 ; q voy a utilizar para sumar los valores
    ldi r16 , 64 ; sumo los 64 valores guardados en vector

```

suma :

```
ld r17, x+           ; r17 = ADCL
ld r18, x+           ; r18 = ADCH
add r23, r17          ; r22 r23 = val_1ADC + val_2ADC +
; .....+ val_32_ADC
adc r22, r18
dec r16
```

division :

```
lsr r22
; queda guardado en r22 r23

cpi r22, 3
; sigo con el 2º promedio
cpi r23, 251
brcc salir           ; si el valor
; 1020 = 1200 mts salgo del promedio
```

promedio\_2 :

```
cp r20, r22          ; comparo el valor almacenado con el
; obtenido con el adc
brne copiar_dato    ; si no es igual copio el
; dato en r20 r21 y vuelo a muestrear
cp r21, r23
breq salir           ; si son iguales salgo de la subrutina
```

copiar\_dato :

```
mov r20, r22
mov r21, r23
rjmp comenzar_muestreo ; si los valores obtenidos
; no son iguales promediar otra vez
```

salir : ret ; al salir los valores promediados quedan
; almacenados en r22 r23

; Factor de escala

;Como la entrada del ADC tiene un offset debo escalar la lectura para
;que me arroje el resultado correcto
;El calculo es el siguiente: Cx = m \* adc - offset. Donde Cx
;es mi capacidad incognita ym es la pendiente
;y adc es la lectura que tomo del pin. m = 1,362 offset = 164.

escalar :

```
push r16
push r17
ldi r16, 87           ; multiplico por 87/64 aprox 1,36
mul prom_low, r16      ; multiplico por 85 lo que está
; guardado en prom_low
mov r17, r1
mov prom_low, r0
```

```

mul prom_high, r16
mov prom_high, r0
add prom_high, r17

ldi r16, 6
divido_x64 :
    lsr prom_high
    ror prom_low
    brne divido_x64

    cpi prom_low, offset + 1           ; si la resta es negati
    brcs resta_2                     ; salto a la resta 2

    subi prom_low, offset
    rjmp salir_escalar

resta_2 :
    subi prom_high, 1
    ldi r17, 256 - offset
    add prom_low, r17

salir_escalar :
    pop r17
    pop r16

    ret

#####
;Rutina para convertir los datos del adc en numeros decimales ascii
;para representar en el lcd.
#####
;Esta rutina recibe valores guardados en las variables prom_high
;prom_low y las convierte en decimales en codigo ascii.
;Los resultados quedan almacenados en u_mil centena decena unidad.

convertir_a_longitud :
    ldi u_mil, '0'                   ; inicio las variables en '0'
    ; escrito en ascii
    ldi centena, '0'
    ldi decena, '0'
    ldi unidad, '0'

    cpi prom_high, 3                ; comparo con el registro de pro
    ; para saber si es menor que mil
    brne menor_a_mil
    cpi prom_low, 232
    brcs menor_a_mil

    ldi u_mil, '1'                 ; si es mayor a mil grabo 1 y 0 en
    ; u_mil y centena respec.

```

```

ldi centena , '0'
subi prom_low, 232 ;y le resto 232 al adc_low

rjmp sumar_decenas ;como la medicion es hasta 20
;no hace falta sumar las centenas

menor_a_mil :
    sbrs prom_high, 0
    rjmp bit_1 ;salto si hay un 0 en
;del registro alto de promedio

    ldi r16, 2 ;sumo 256 si el bit está en 1
    add centena, r16
    ldi r16, 5
    add decena, r16
    ldi r16, 6
    add unidad, r16

bit_1 : sbrs prom_high, 1
        rjmp sumar_centenas ;salto si hay un 0 en el bit
;registro alto de promedio

        ldi r16, 5 ;sumo 512 si el bit está en 1
        add centena, r16
        ldi r16, 1
        add decena, r16
        ldi r16, 2
        add unidad, r16

sumar_centenas :
    cpi prom_low, 100
    brccs sumar_decenas ;si el resultado en el registr
;bajo es menor a 100 (b = 1) sigo con las decenas

    subi prom_low, 100 ;sino le resto 100 al registr
;incremento la centena y vuelvo a comparar con 100
    inc centena
    rjmp sumar_centenas

sumar_decenas :
    cpi prom_low, 10
    brccs sumar_unidades ;si el resultado en el registr
;bajo es menor a 10 sigo con las unidades

    subi prom_low, 10 ;sino le resto 10 al registr
;incremento la decena y vuelvo a comparar con 10
    inc decena
    rjmp sumar_decenas

sumar_unidades :
    add unidad, prom_low
    cpi unidad, '0' + 10 ;comparo la unidad con el nro

```

```

; 10 en ascii
brcs verificar_decena ; si el resultado es negativo
;(C = 1) verifco las decenas

subi unidad , 10 ; sino le resto los 10 sobrantes y
;le sumo uno a la decena
inc decena

verificar_decena :
    cpi decena , '0'+ 10 ;comparo la decena con el nro
    ; 10 en ascii
    brcs salir_conversor ; si el resultado es negativo (C

    subi decena , 10 ; sino le resto los 10 sobrantes y
    ;le sumo uno a la centena
    inc centena

salir_conversor : ; al salir los resultados quedan almacenados en
                    ; u_mil centena decena unidad que son re
    ret ; registros r18 r19 r20 r21.

;

; Constantes en memoria Flash para utilizar en pantalla LCD
;
MJE0 : .db "Iniciando...",0
MJE1 : .db "Probando.....",0
MJE2 : .db "MENU PRINCIPAL",0
MJE3 : .db "Probar linea",0
MJE4 : .db "Medir capacidad",0
MJE5 : .db "capacidad = ",0
MJE6 : .db " nF",0
MJE7 : .db " pF",0
MJE8 : .db "capacidad mayor a 50 nF",0
MJE9 : .db "capacidad menor a 1 nF",0

```

## 11. Bibliografía y recursos

<http://www.edn.com/design/test-and-measurement/4423158/Precision-capacitance-meter>.

<http://www.condumex.com.mx/ES/telecomunicaciones/Productos20telecomunicaciones/Cables%20te>

## 12. Apéndice

Se utilizaron las ecuaciones 2 y 3 para transformar la lectura del ADC en unidades de capacidad, basándose en la curva de la figura 7.

$$m = \frac{507}{507 - 137} = 1,362 \quad (2)$$

En el programa se reemplazó  $m$  por  $\frac{87}{64}$

$$b = -\frac{87}{64} * 135 \approx -184 \quad (3)$$

Por eso se utilizó  $offset = -184$