FACULTAD DE INGENIERIA
Universidad de Buenos Aires

Laboratorio de Microcomputadoras - 86.09

Cofímetro

Profesor:			Ing. Guillermo Campiglio									
Cuatrimestre/Año:			2º/2019									
Turno de las clases prácticas			Miércoles 19-22 hs									
Jefe de trabajos prácticos:			Ricardo Arias									
Docente guía:			Ricardo Arias									
Autores			Seguimiento del proyecto									
Nombre	Apellido	Padron										
Manuel	Cabeza	101627										
Axel	Itzcovitz	100742										
Matías	Charrut	101137										

Observaciones:

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

Fecha de aprobación				Firma J.T.P

Coloquio	
Nota final	
Firma profesor	

Índice

1. Objetivos **1**

2. Descripción del proyecto **1**

3. Diagrama de bloques **1**

4. Circuito esquemático **1**
4.1. Medición de tensión 1
4.2. Medición de corriente 2

5. Listado de componentes **2**

6. Software **4**
6.1. Medición de tensión y corriente 6
6.2. Medición de desfasaje 6

7. Resultados **7**

8. Conclusiones **8**

9. Apéndice I (Circuito Completo) **8**

10. Apéndice II (Código) **18**

11. Apéndice III: Presupuesto del proyecto **35**

1. Objetivos

El objetivo del proyecto es presentar en un display la tensión, corriente y desfasaje de un dispositivo conectado a la linea.

2. Descripción del proyecto

El proyecto consiste en un bloque medidor, que se encarga de obtener la información de tensión y corriente del circuito conectado a la tensión de línea, así como de adaptar estos valores de tal forma de poder ser procesados por el microcontrolador. Con esta información recibida, este último irá recolectando los valores de tensión y corriente para obtener un valor RMS de cada uno, así como también analiza cada una de las señales por medio de los muestreos adaptados que se le ingresan para calcular el desfasaje que hay entre ambas señales, para así obtener el $\cos(\varphi)$ del dispositivo. Luego, se imprimen los valores obtenidos (tensión, corriente y $\cos(\varphi)$) utilizando un display LCD.

3. Diagrama de bloques

Figura 1: Diagrama de bloques.

El circuito funciona a partir de la tensión de linea para medir la corriente, tensión y el desfasaje. La medición de corriente se realiza mediante el sensor ASC-712 y la de tensión mediante un circuito realizado especialmente. El desfasaje se obtiene haciendo uso del comparador interno del microcontrolador y de un comparador externo. Estos tres parámetros son dispuestos en el display.

4. Circuito esquemático

El circuito completo se encuentra en el apéndice I. En esta sección se describe el circuito realizado para la medición de tensión y el sensor utilizado para la medición de corriente (ACS-712).

4.1. Medición de tensión

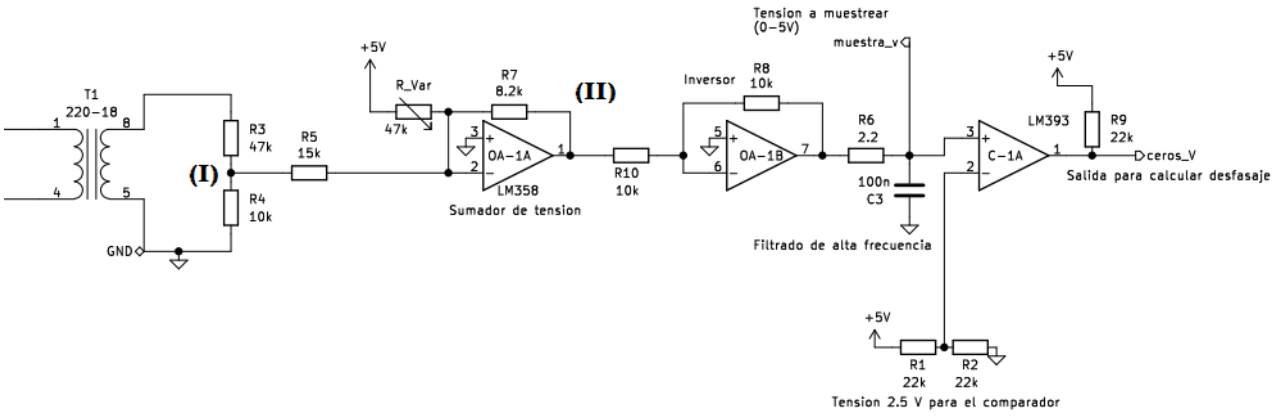


Figura 2: Esquema del circuito para medir tensión.

El circuito busca obtener una muestra de tensión proporcional a la de entrada que tome valores entre 0 y 5 V y una señal cuadrada que tome valores '1' cuando la entrada se encuentra en el semiciclo positivo y '0' cuando está en el semiciclo negativo. El mismo se conecta en paralelo con el dispositivo al cual se le va a medir la corriente.

Para esto se comienza con un transformador que baja el valor eficaz de la señal en un factor de aproximadamente 12.22. Esa señal pasa por un divisor resistivo para que la entrada al sumador oscile con un valor pico

de 2.5 V (nodo I). La otra entrada del sumador proviene de los 5 V. A partir de los valores de resistencias R5 - R7 se modifica el preset de 47 k para obtener en el nodo II una señal que oscila alrededor de 2,5 V con un valor pico de aproximadamente 2,5 V a una entrada de 220 V_{ef}. Esta señal pasa por un inversor para obtener la señal que se busca entre 0 y 5 V, esa pasa por un filtro RC de una alta frecuencia de corte para disminuir el ruido. A la salida del filtro se toma la señal *muestra_v* que va al ADC del microcontrolador. La misma va además al terminal inversor del comparador, que tiene su terminal positivo conectado a una señal continua de 2,5 V. Así a la salida del comparador se tiene la señal *ceros_v* que se utiliza para calcular el desfasaje. A la salida del comparador se utilizó una resistencia de Pull-Up.

Así se obtiene una señal en *muestra_v* cuyo valor eficaz se relaciona con el valor eficaz de la señal de entrada de forma lineal a través de la constante 91,3 (la obtención de la constante de proporcionalidad fue de forma experimental.).

4.2. Medición de corriente

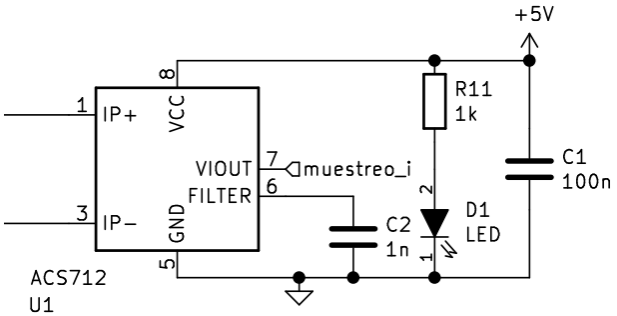


Figura 3: Esquema del circuito para medir corriente.

Para la medición de corriente se utilizó el circuito de la figura anterior conectado en serie con el vivo de la tensión de línea (ver Anexo I), el mismo asegura una variación lineal de tensión de salida según la corriente de entrada dada por la siguiente formula en el terminal *muestreo_i*, siempre que el valor de la corriente pico sea menor a 5A.

$$V_{OUT} = 2,5V + I\,0,2\frac{V}{A}$$

Como se alimenta con $V_{CC} = 5$ V, la señal de salida varía entre 0,5 y 5 V. Ésta se conecta con el terminal del ADC del microcontrolador.

5. Listado de componentes

- Display LCD 16x2
- Sensor de efecto Hall ACS712 ± 5A
- Comparador LM393
- Arduino UNO
- Amplificador Operacional LM358
- Transformador 220V-18V
- Tomacorriente 220V
- Enchufe 220V
- 2 Presets 100 KΩ

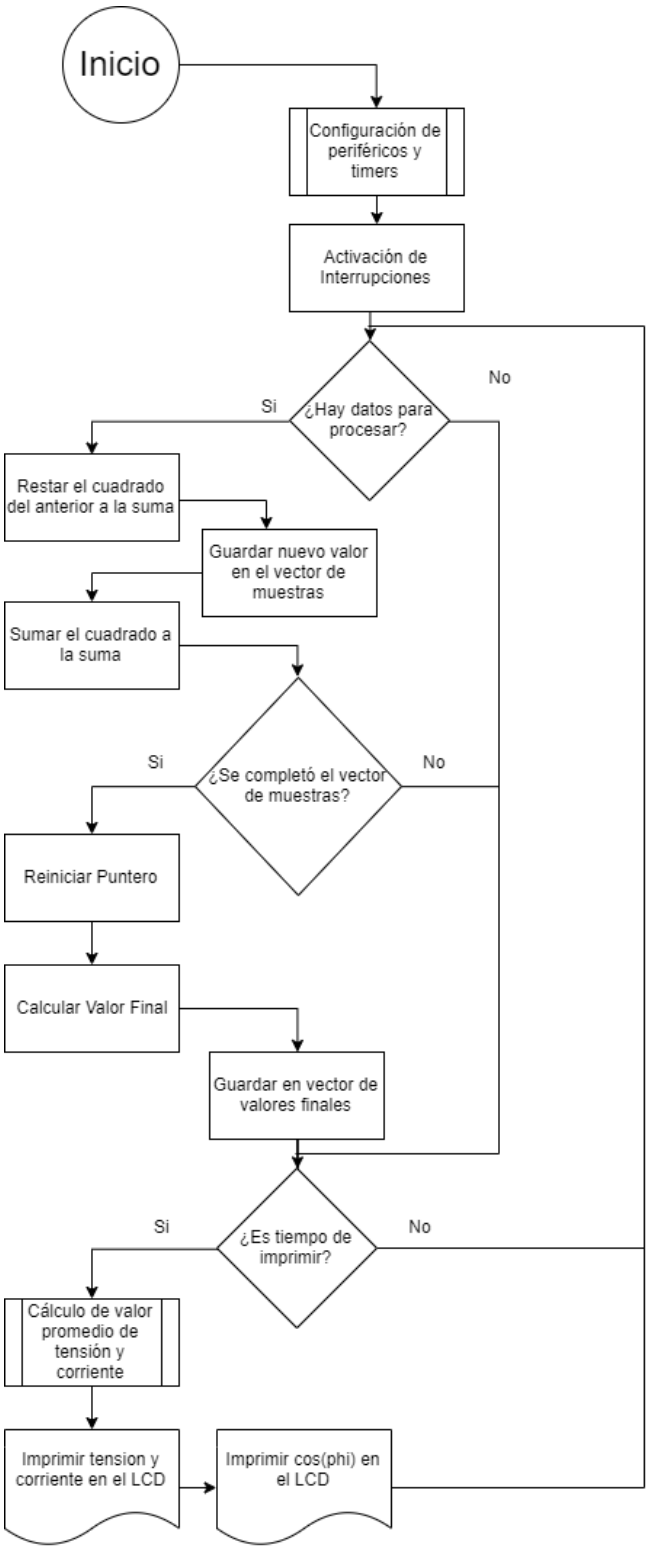


Figura 4: Diagrama de flujo del programa principal

6. Software

El programa principal comienza con la configuración del LCD, el ADC, el *timer0*, el *timer1* y las interrupciones. Luego, comienza el ciclo infinito del main, el cual verifica todo el tiempo si hay datos para procesar, además de vigilar si es momento de imprimir datos en el LCD. Son cuatro las distintas interrupciones del programa: el overflow del *timer0*, la cual enciende la conversión del ADC[1], la producida cuando el ADC termina de convertir, y las de cambio de estado en el pin D5 y en el comparador. Tanto la medición de tensión como la de corriente se realizan cada $39\,\mu s$, de manera tal que se muestreen 256 veces cada una en cada ciclo (50 Hz).

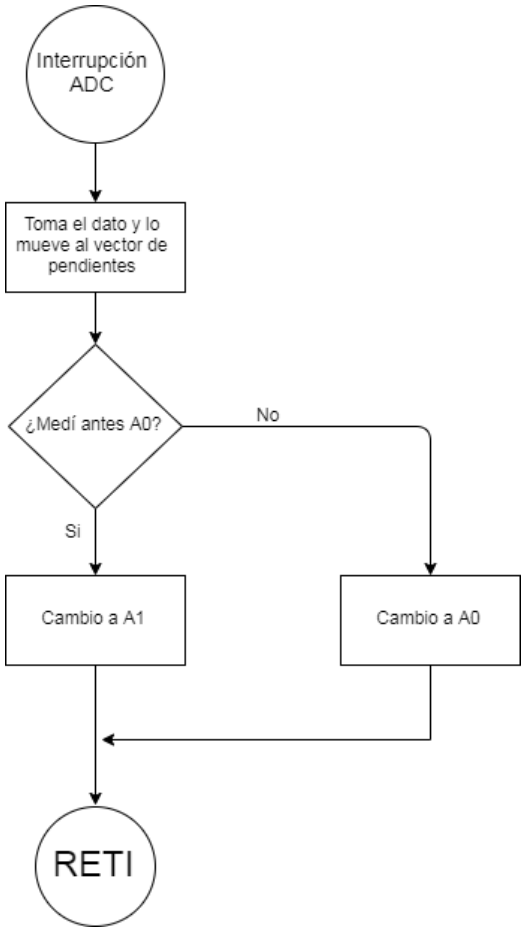
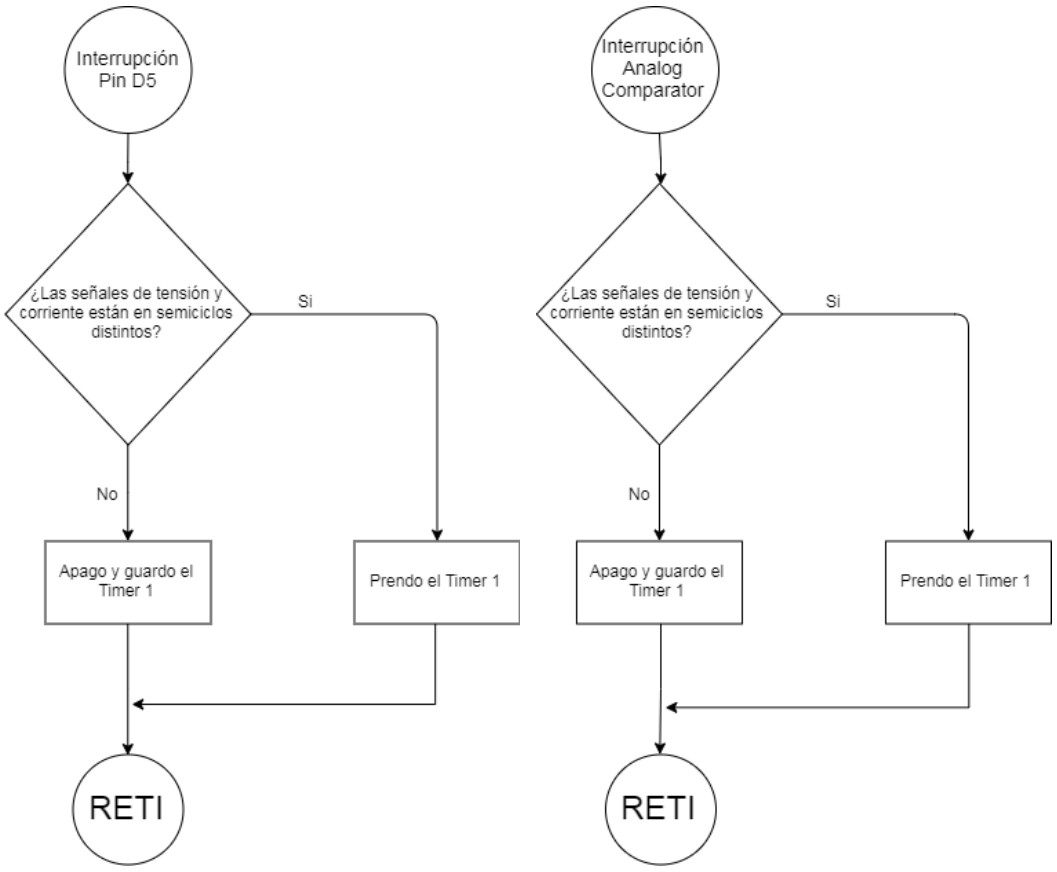


Figura 5: Diagrama de flujo de la interrupción por ADC

[1]El *timer0* también suma al contador que maneja la impresión, lo cual está ajustado para que imprima cada aproximadamente 1 segundo.

Figura 6: Diagrama de flujo de la interrupción por Timer 0



(a) Interrupción por el pin D5

(b) Interrupción por el comparador analógico

Figura 7: Diagramas de flujo de las interrupciones para el desfasaje

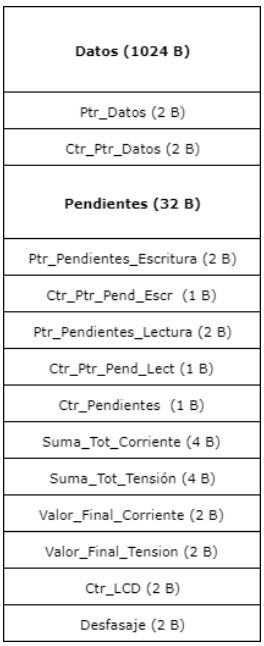


Figura 8: Mapa de Memoria RAM

6.1. Medición de tensión y corriente

La tensión se mide por el pin A0 y la corriente por el pin A1. El proceso de medición es similar en ambos casos. Dado que el ADC es de 10 bits, a partir de lo que se mida se obtendrá un valor entre 0 y 1023. Cada vez que se mida, la interrupción del ADC guardará el valor en un vector de pendientes (cuando se llena comienza a pisar valores) y suma a un contador. El programa principal vigila este contador y cuando detecta que hay un valor pendiente, lo procesa: calcula el cuadrado del valor anterior a éste (es decir, el valor que se guardó hace 256 muestras) y se lo resta a la suma total de los cuadrados. Después, guarda el valor nuevo en el vector, le calcula el cuadrado y se lo suma al total. Cuando detecta que se llenó el vector (es decir, que llegó a la posición 256) reinicia el puntero del vector (es decir, vuelve al principio) y calcula el valor final RMS de la tensión (o la corriente), de modo que

$$X_{RMS} = \sqrt{\frac{\sum_{i=1}^{256} x_i^2}{N}}$$

Este valor es guardado en un vector con los últimos 32 calculados para luego ser promediado al momento de imprimir, de modo que se pueda reducir el ruido introducido al momento de la medición. Para la impresión, este valor se usa para entrar en la tabla correspondiente (tensión o corriente), la cual lo convierte al valor real que va a ser impreso.

6.2. Medición de desfasaje

Para la medición del desfasaje, se busca obtener la diferencia de tiempo entre el cruce por cero de tensión y el cruce por cero de corriente. Para ello, primeramente se obtiene una señal cuadrada de cada una, tal que la misma valga '1' lógico para cuando se encuentra en el semiciclo positivo, y '0' lógico para cuando se encuentra en el semiciclo negativo. En el caso de la tensión, esta señal cuadrada valdrá '1' para los valores de tensión positivos, mientras que para la corriente, dado que la señal que sale del ACS712 se encuentra montada sobre una tensión continua de 2.5V, la señal valdrá '1' para valores de tensión mayores a esta continua sobre la que está montada. Para la obtención de la señal cuadrada de tensión, se utiliza el comparador LM393, el cual compara la señal de tensión con 2.5V, dado que la señal que obtenemos de la línea es amplificada para luego agregarle la continua de comparación, obteniendo la salida en el pin digital 5 (PD5), mientras que para la señal cuadrada de corriente se utiliza el comparador interno analógico del microcontrolador, realizando también la comparación con 2.5V, obteniendo la salida en el analog-comparator-output (ACO). Cada vez que cambia el pin 5 o la salida del comparador del microcontrolador salta una interrupción, la cual realiza una XOR entre los dos valores. Si los dos son un '1' o los dos son un '0', se apaga el *timer1* y se guarda su valor. Si los valores son diferentes, se reinicia el *timer1* y se prende. El *preescaler* se configuró de manera que sea posible medir 10ms, que es el máximo desfasaje posible que puede existir en una señal de línea. Para la impresión, este valor se usa para entrar en una tabla, la cual lo convierte al valor real que va a ser impreso.

7. Resultados

Una vez ensamblados hardware y software, los resultados iniciales fueron muy desalentadores, dado que no se conseguía estabilizar los valores a obtener, además de que los valores obtenidos no eran los correctos (según los valores de tensión de línea y corriente medidos con un multímetro). Así, se encontraron múltiples diferencias entre nuestra estimación de lo que sería el circuito con lo que finalmente era, como que la tensión continua del ACS712 no era exactamente 2.5V o que la frecuencia de la tensión de línea no coincidía con nuestra tensión de muestreo del microcontrolador. Por lo tanto, se debió realizar una calibración del circuito, para así ajustar nuestros cálculos iniciales a los valores reales de cada componente.

La calibración se realizó tomando muestras de los valores a la entrada y la salida del circuito y, dado que la relación entre los mismos varía como una raíz cuadrada al ser valores eficaces, se elevaron al cuadrado los resultados para levarla a una relación lineal y poder hacer cuadrados mínimos. Así se obtuvo una calibración final que lleva a una mejor aproximación de las mediciones que la original.

Para poder medir el $\cos(\varphi)$ se debió realizar un circuito aparte que genere un desfasaje entre dos señales, debido a que para medir un $\cos(\varphi)$ distinto de 1 se debía utilizar un instrumento altamente inductivo, como un motor, el cual resultaba impráctico para realizar la medición. Para el mismo se utilizó el circuito en la figura 9, con el cual utilizando un preset entre la entrada V_1 y el borne positivo del operacional se pudo regular un $\cos(\varphi)$ entre 0 y 1, para así verificar el correcto funcionamiento del medidor a lo largo de todos los valores.

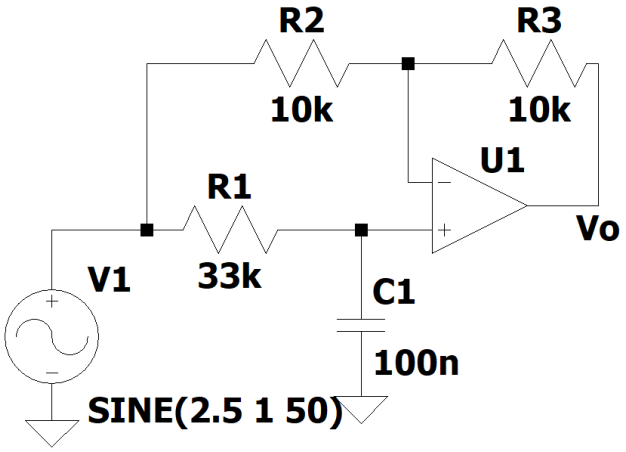


Figura 9: Circuito utilizado para la medición del $\cos(\varphi)$

Sobre la precisión final del las mediciones, en el caso de la tensión se debió mapear un valor eficaz a la entrada que iba desde 0 V hasta como máximo 250 V a una variación de valor eficaz a la salida de 0.56 V (como máximo). Debido a esto la incerteza final de las mediciones no se pudo reducir demasiado y de los 10 bits que se utilizaron para realizar la medición, es decir una tabla de 1024 posiciones, solo fueron utilizables 114 entradas.

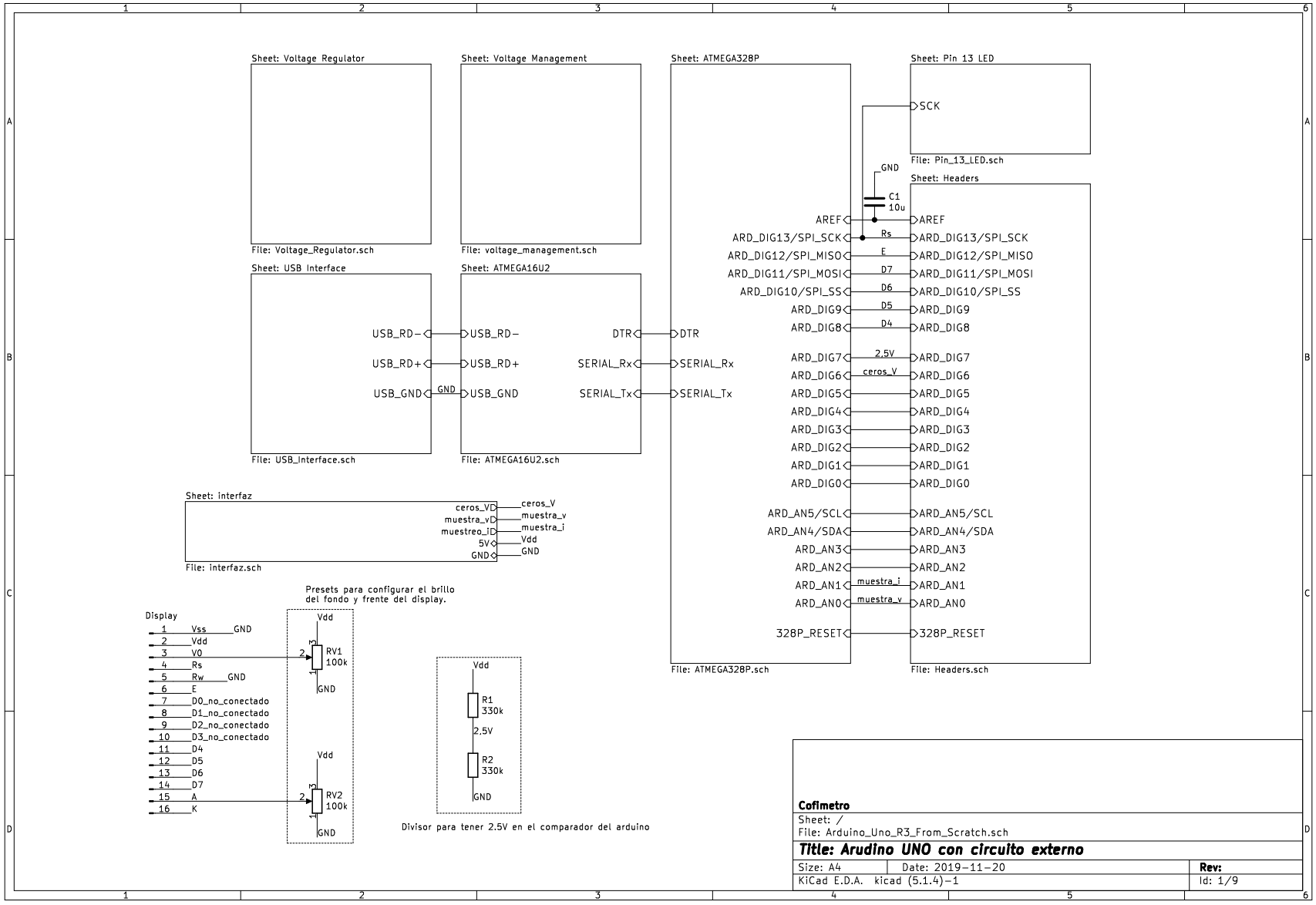
Algo análogo sucedió para la corriente, donde el valor eficaz máximo obtenible limitado por el sensor es de 3,57 A. A la salida también se obtiene una señal con una variación de valor eficaz menor a 0,56 V reduciendo el número efectivo de entradas a las tablas de mediciones y empeorando la incerteza de las mediciones.

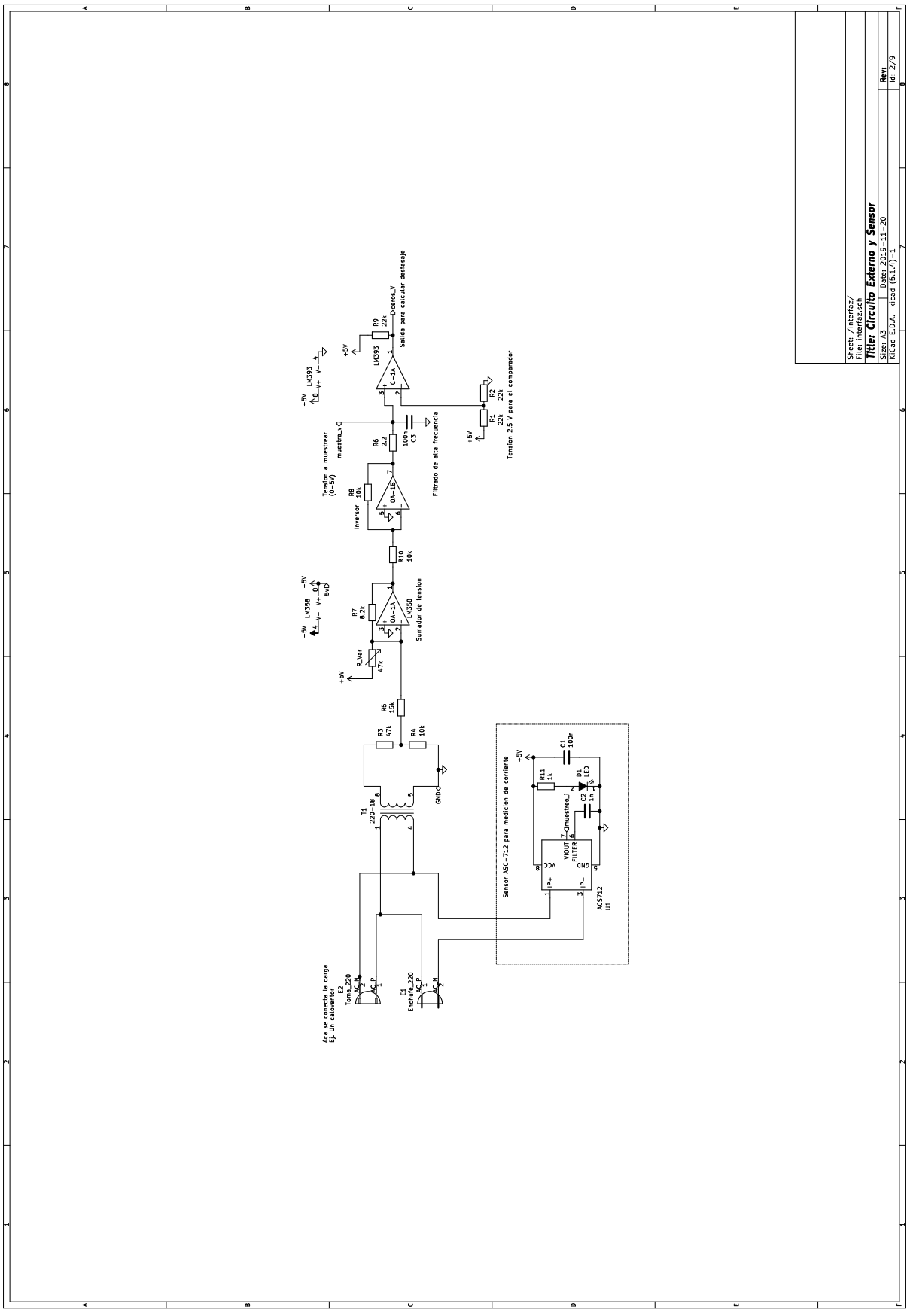
8. Conclusiones

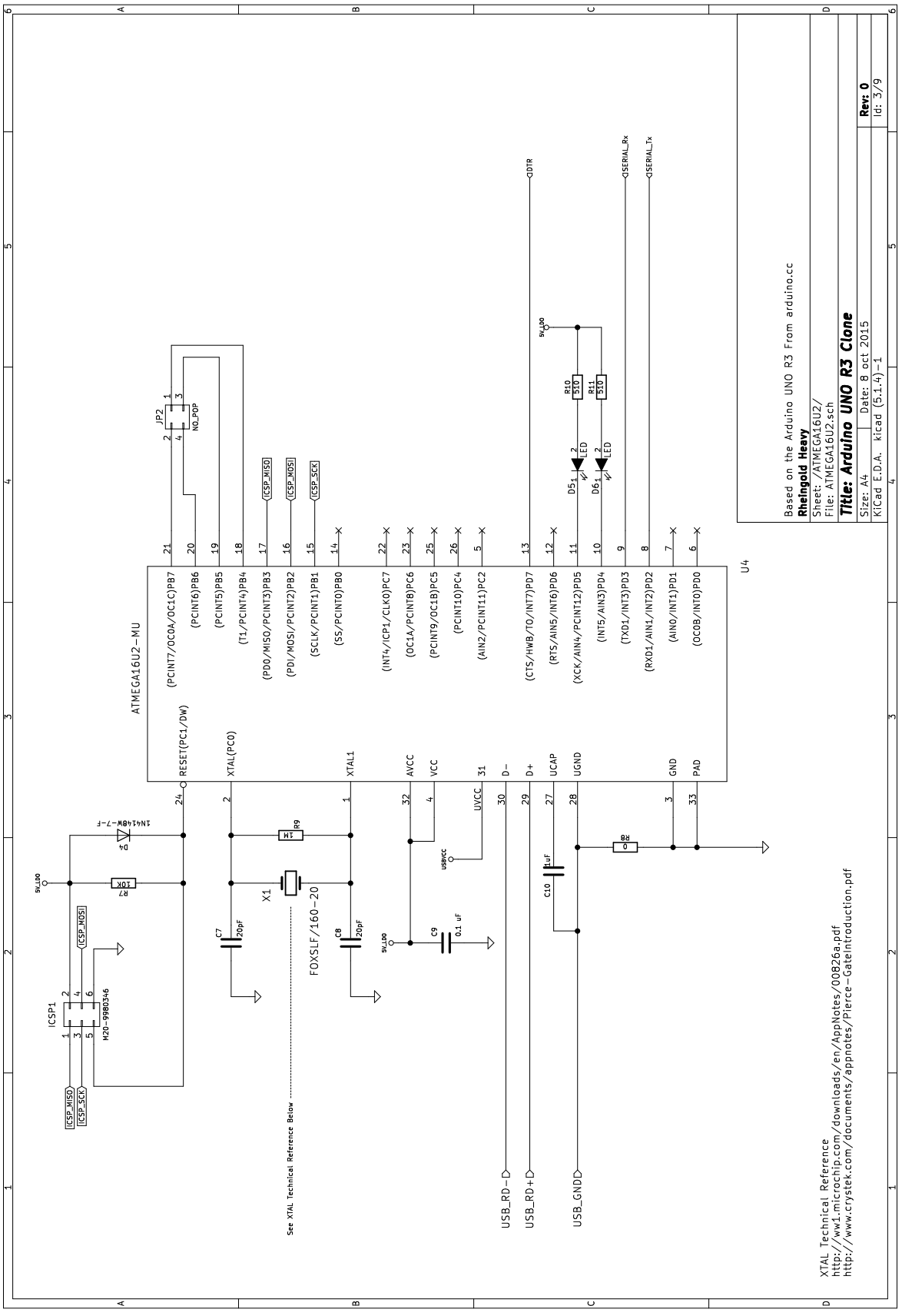
Una de las principales conclusiones que se obtuvieron a partir de la realización del proyecto es la importancia de la correcta obtención de los valores con los que se realizan los cálculos previos a las mediciones y la correcta caracterización de los periféricos, ya que por ejemplo, según los cálculos del amplificador para la tensión, la continua que agregaba era de 2,5 V, pero la que se agregaba realmente era de 2,4 V, y esto alteraba las mediciones. Estas diferencias entre la teoría y la práctica llevaron a que se obtuvieran serias diferencias entre los valores que debíamos obtener y los obtenidos, dada la sensibilidad que manejaba este circuito con respecto a estos valores.

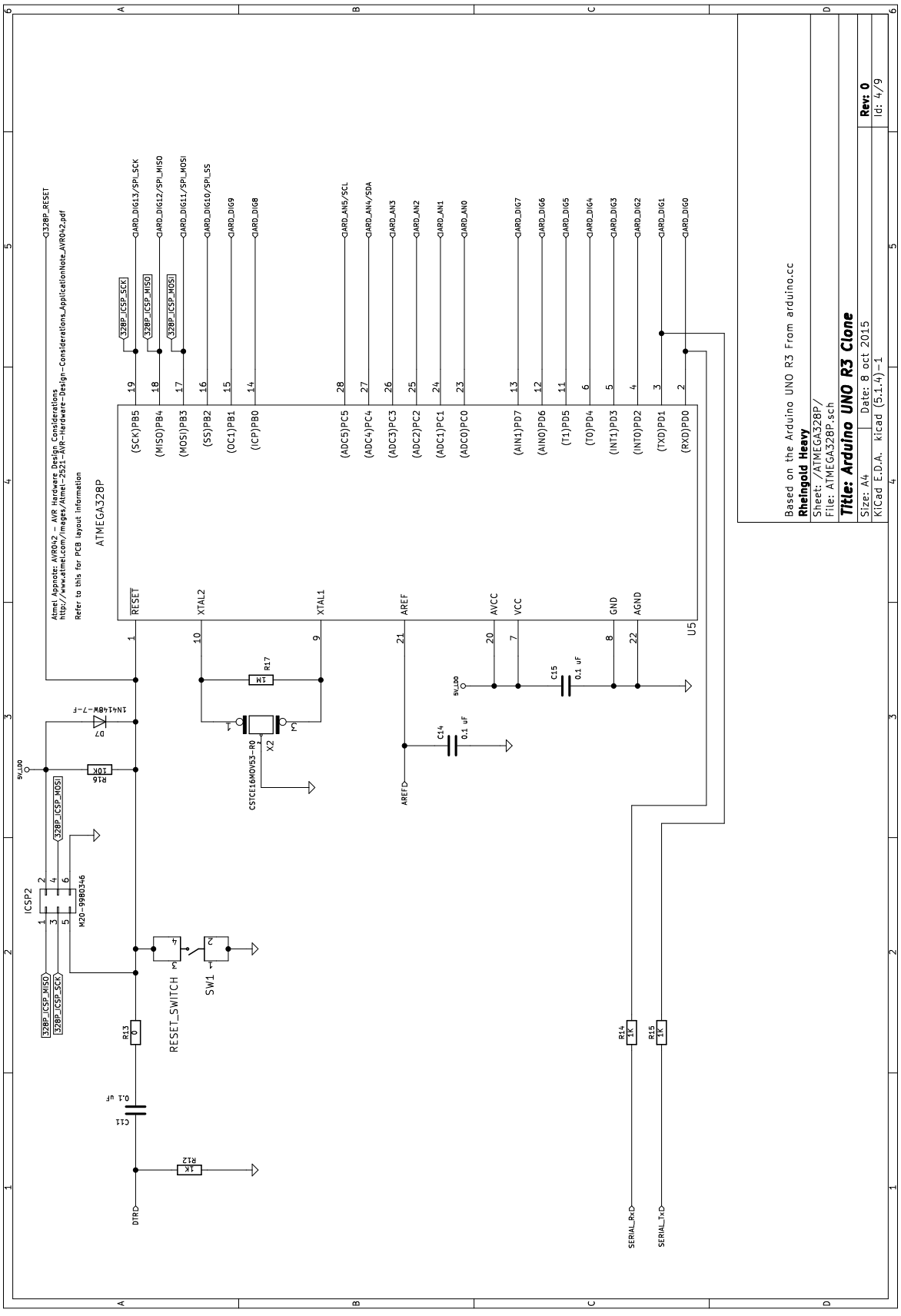
Por último, la experiencia de realizar un proyecto cuya función principal fuese la medición resultó un tanto conflictiva, debido a la cantidad de componentes que requiere el mismo, aumentando así la posibilidad de obtener mediciones erróneas o inestables ante la pequeña falla de alguno de estos, como un pequeño desafloje de los pines o los cables, un pequeño desajuste en la frecuencia de muestreo, o una ligera diferencia entre una constante teórica y real.

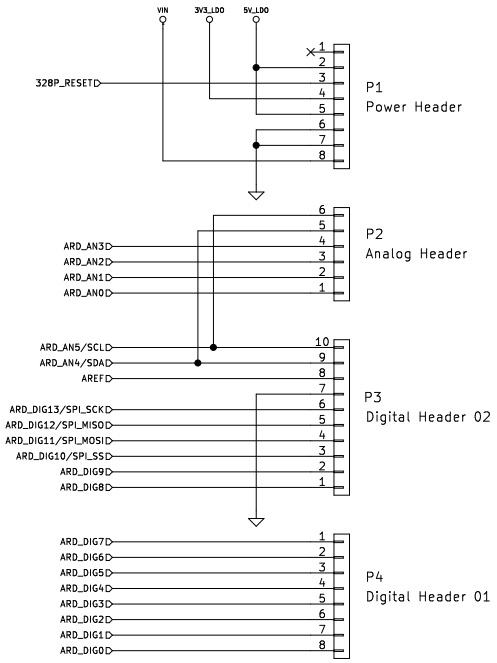
9. Apéndice I (Circuito Completo)


Sheet: Voltage Regulator
File: Voltage_Regulator.sch
Sheet: Voltage Management
File: voltage_management.sch
Sheet: ATMEGA328P
File: ATMEGA328P.sch
Sheet: Pin 13 LED
SCK
File: Pin_13_LED.sch
Sheet: Headers
File: Headers.sch
GND
C1
10u
AREF
ARD_DIG13/SPI_SCK
ARD_DIG12/SPI_MISO
ARD_DIG11/SPI_MOSI
ARD_DIG10/SPI_SS
ARD_DIG9
ARD_DIG8
ARD_DIG7
ARD_DIG6
ARD_DIG5
ARD_DIG4
ARD_DIG3
ARD_DIG2
ARD_DIG1
ARD_DIG0
ARD_AN5/SCL
ARD_AN4/SDA
ARD_AN3
ARD_AN2
ARD_AN1
ARD_AN0
328P_RESET
Rs
E
D7
D6
D5
D4
2.5V
ceros_V
muestra_i
muestra_v
Sheet: USB Interface
File: USB_Interface.sch
Sheet: ATMEGA16U2
File: ATMEGA16U2.sch
USB_RD-
USB_RD+
USB_GND
DTR
SERIAL_Rx
SERIAL_Tx
Sheet: interfaz
File: interfaz.sch
ceros_VD
muestra_vD
muestreo_ID
5V
Vdd
Presets para configurar el brillo
del fondo y frente del display.
Display
Vss
Vdd
V0
Rs
Rw
E
D0_no_conectado
D1_no_conectado
D2_no_conectado
D3_no_conectado
D4
D5
D6
D7
A
K
RV1
100k
RV2
100k
R1
330k
R2
330k
Divisor para tener 2.5V en el comparador del arduino
Cofimetro
Sheet: /
File: Arduino_Uno_R3_From_Scratch.sch
Title: Arudino UNO con circuito externo
Size: A4
Date: 2019-11-20
Rev:
KiCad E.D.A. kicad (5.1.4)-1
Id: 1/9


Sheet: /interfaz/
File: interfaz.sch
Title: Circuito Externo y Sensor
Size: A3
Date: 2019-11-20
Rev:
KiCad E.D.A. kicad (5.1.4)-1
Id: 2/9
Salida para calcular desfasaje
Tension a muestrear (0-5V)
Filtrado de alta frecuencia
Tension 2.5 V para el comparador
Inversor
Sumador de tension
Sensor ASC-712 para medicion de corriente
Aca se conecta la carga Ej: Un calovento
LM393
LM358
ACS712
Toma_220
Enchufe_220
GND


ATMEGA16U2-MU
U4
ICSP1
M20-9980346
JP2
NO_POP
X1
FOXSLF/160-20
C7
20pF
C8
20pF
C9
0.1 uF
C10
1uF
R7
10K
R8
0
R9
1M
R10
510
R11
510
D4
1N4148W-7-F
D5
LED
D6
LED
5V_USB
USBVCC
UVCC
RESET(PC1/DW)
XTAL(PC0)
XTAL1
AVCC
VCC
D-
D+
UCAP
UGND
GND
PAD
(PCINT7/OC0A/OC1C)PB7
(PCINT6)PB6
(PCINT5)PB5
(T1/PCINT4)PB4
(PDO/MISO/PCINT3)PB3
(PDI/MOSI/PCINT2)PB2
(SCLK/PCINT1)PB1
(SS/PCINT0)PB0
(INT4/ICP1/CLKO)PC7
(OC1A/PCINT8)PC6
(PCINT9/OC1B)PC5
(PCINT10)PC4
(AIN2/PCINT11)PC2
(CTS/HWB/TO/INT7)PD7
(RTS/AIN5/INT6)PD6
(XCK/AIN4/PCINT12)PD5
(INT5/AIN3)PD4
(TXD1/INT3)PD3
(RXD1/AIN1/INT2)PD2
(AIN0/INT1)PD1
(OC0B/INT0)PD0
ICSP_MISO
ICSP_MOSI
ICSP_SCK
DTR
SERIAL_Rx
SERIAL_Tx
USB_RD-
USB_RD+
USB_GND
See XTAL Technical Reference Below
Based on the Arduino UNO R3 From arduino.cc
Rheingold Heavy
Sheet: /ATMEGA16U2/
File: ATMEGA16U2.sch
Title: Arduino UNO R3 Clone
Size: A4
Date: 8 oct 2015
Rev: 0
KiCad E.D.A. kicad (5.1.4)-1
Id: 3/9
XTAL Technical Reference
http://ww1.microchip.com/downloads/en/AppNotes/00826a.pdf
http://www.crystek.com/documents/appnotes/Pierce-GateIntroduction.pdf


Based on the Arduino UNO R3 From arduino.cc
Rheingold Heavy
Sheet: /ATMEGA328P/
File: ATMEGA328P.sch
Title: Arduino UNO R3 Clone
Size: A4
Date: 8 oct 2015
Rev: 0
KiCad E.D.A. kicad (5.1.4)-1
Id: 4/9
ATMEGA328P
Atmel Appnote AVR042 - AVR Hardware Design Considerations
http://www.atmel.com/images/Atmel-2521-AVR-Hardware-Design-Considerations_ApplicationNote_AVR042.pdf
Refer to this for PCB layout information
328P_RESET
ARD_DIG13/SPI_SCK
ARD_DIG12/SPI_MISO
ARD_DIG11/SPI_MOSI
ARD_DIG10/SPI_SS
ARD_DIG9
ARD_DIG8
ARD_AN5/SCL
ARD_AN4/SDA
ARD_AN3
ARD_AN2
ARD_AN1
ARD_AN0
ARD_DIG7
ARD_DIG6
ARD_DIG5
ARD_DIG4
ARD_DIG3
ARD_DIG2
ARD_DIG1
ARD_DIG0
(SCK)PB5
(MISO)PB4
(MOSI)PB3
(SS)PB2
(OC1)PB1
(ICP)PB0
(ADC5)PC5
(ADC4)PC4
(ADC3)PC3
(ADC2)PC2
(ADC1)PC1
(ADC0)PC0
(AIN1)PD7
(AIN0)PD6
(T1)PD5
(T0)PD4
(INT1)PD3
(INT0)PD2
(TXD)PD1
(RXD)PD0
RESET
XTAL2
XTAL1
AREF
AVCC
VCC
GND
AGND
U5
328P_ICSP_SCK
328P_ICSP_MISO
328P_ICSP_MOSI
5V_UNO
R17
1M
X2
CSTCE16M0V53-R0
C14
0.1 uF
C15
0.1 uF
D7
1N4148W-7-F
R16
10K
ICSP2
M20-9980346
RESET_SWITCH
SW1
R13
0
C11
0.1 uF
R12
1K
DTRD
R14
1K
R15
1K
SERIAL_RxD
SERIAL_TxD



Based on the Arduino UNO R3 From arduino.cc

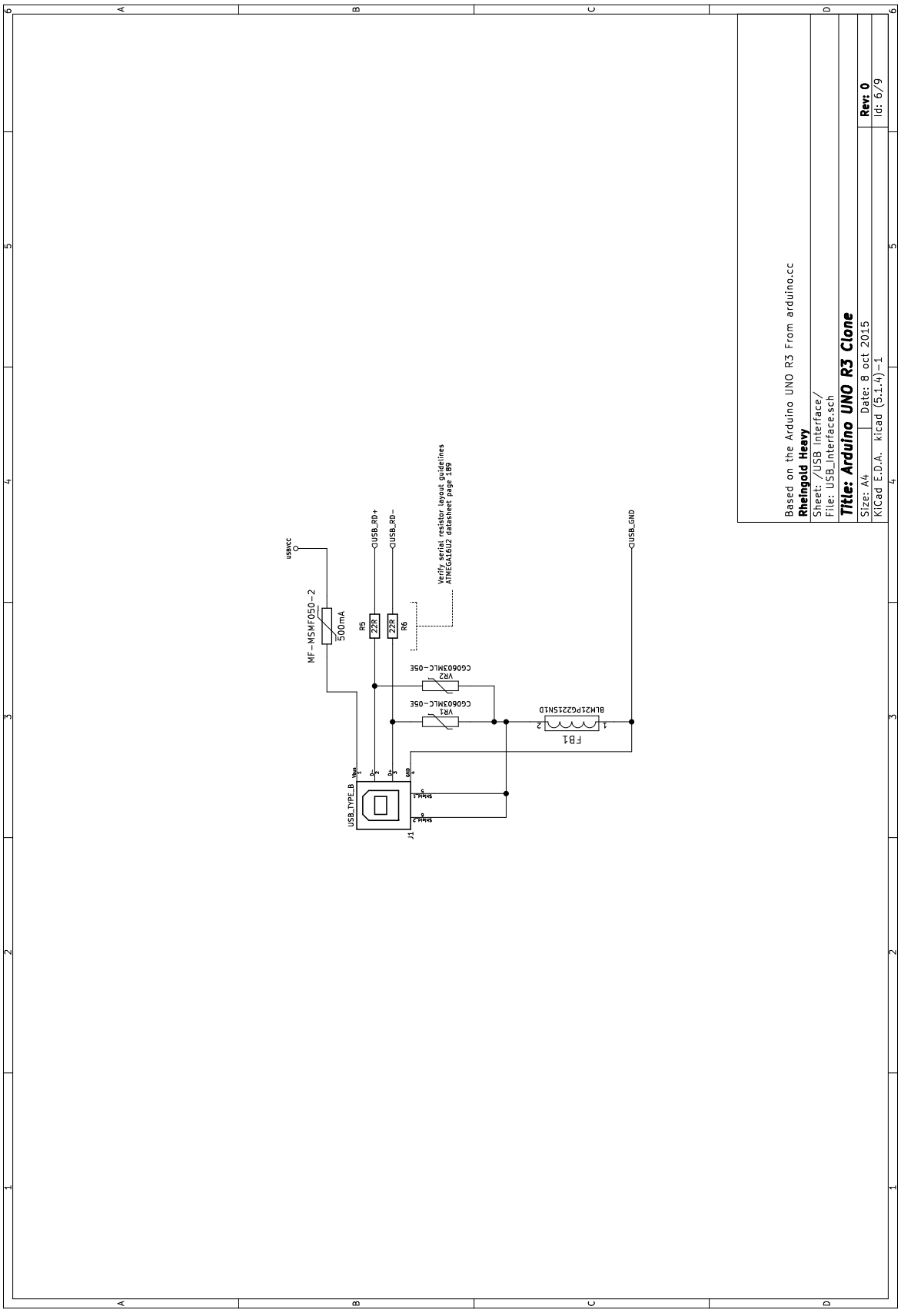
Rheingold Heavy

Sheet: /Headers/

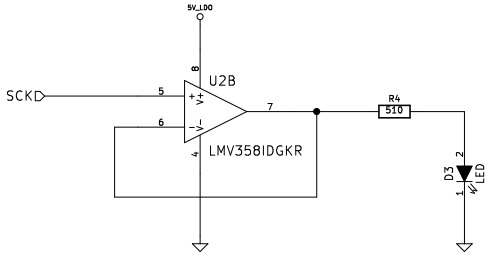
File: Headers.sch

Title: Arduino UNO R3 Clone

Size: A4	Date: 8 oct 2015	**Rev: 0**
KiCad E.D.A. kicad (5.1.4)-1		Id: 5/9


USBVCC
MF-MSMF050-2
500mA
USB_D0+
USB_D0-
R5
22R
R6
22R
Verify serial resistor layout guidelines
ATMEGA16U2 datasheet page 189
USB_GND
CG0603MLC-05E
VR2
CG0603MLC-05E
VR1
BLM21PG221SN1D
FB1
USB_TYPE_B
VBUS
D+
D-
GND
SHIELD1
SHIELD2
J1
Based on the Arduino UNO R3 From arduino.cc
Rheingold Heavy
Sheet: /USB Interface/
File: USB_Interface.sch
Title: Arduino UNO R3 Clone
Size: A4
Date: 8 oct 2015
Rev: 0
KiCad E.D.A. kicad (5.1.4)-1
Id: 6/9

SCK / PIN13 LED Buffer



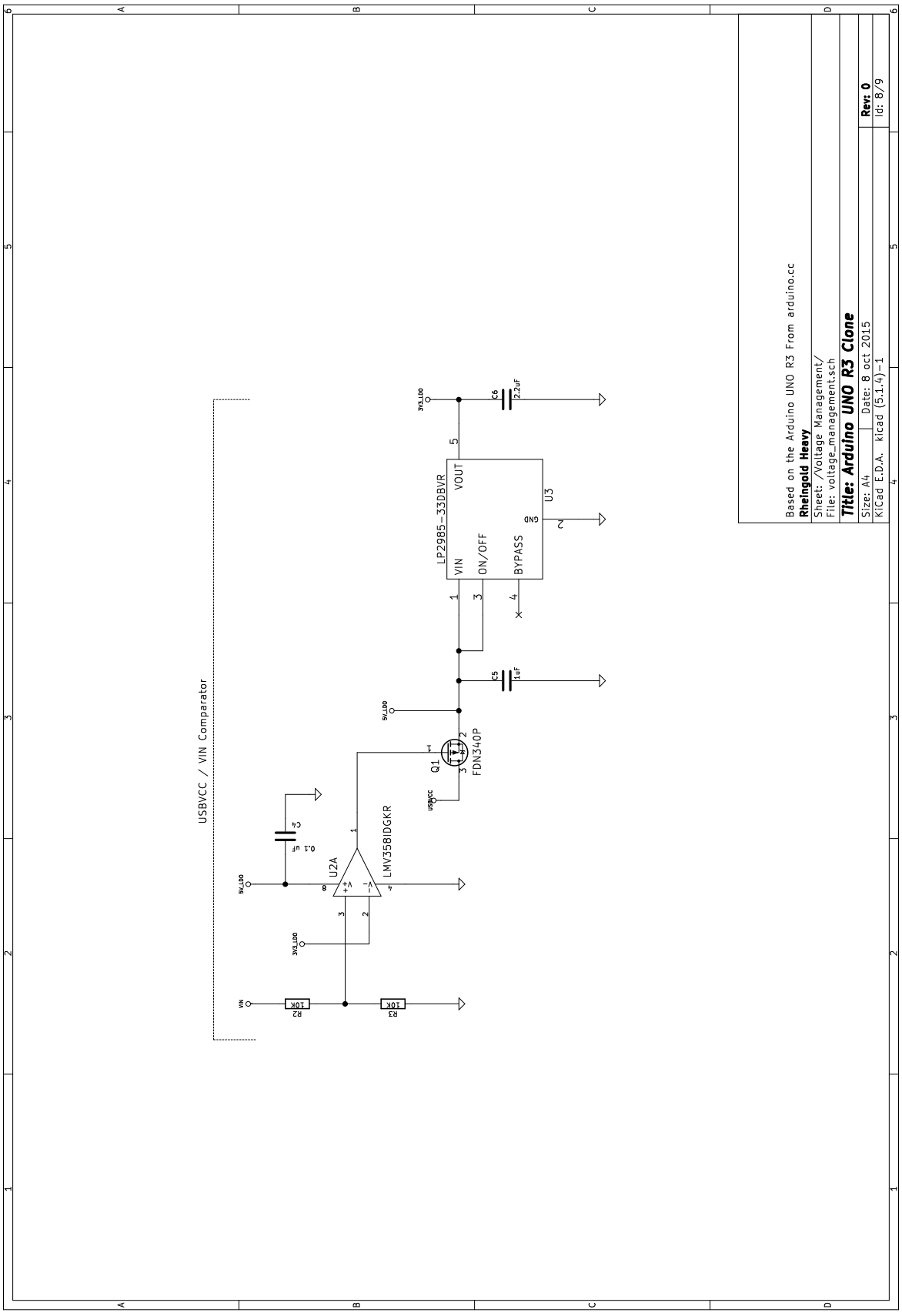
Based on the Arduino UNO R3 From arduino.cc

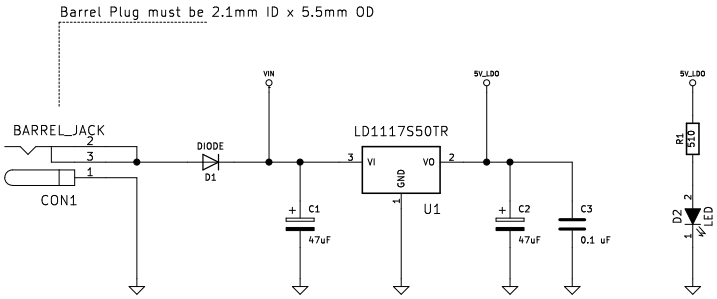
Rheingold Heavy

Sheet: /Pin 13 LED/
File: Pin_13_LED.sch

Title: Arduino UNO R3 Clone

Size: A4	Date: 8 oct 2015	**Rev: 0**
KiCad E.D.A. kicad (5.1.4)-1		Id: 7/9


USBVCC / VIN Comparator
LP2985-33DBVR
VIN
ON/OFF
BYPASS
VOUT
GND
U3
C6
2.2uF
C5
1uF
3V3_LDO
5V_LDO
Q1
FDN340P
USBVCC
U2A
LMV358IDGKR
C4
0.1 uF
R2
10K
R3
10K
VIN
Based on the Arduino UNO R3 From arduino.cc
Rheingold Heavy
Sheet: /Voltage Management/
File: voltage_management.sch
Title: Arduino UNO R3 Clone
Size: A4
Date: 8 oct 2015
KiCad E.D.A. kicad (5.1.4)-1
Rev: 0
Id: 8/9



All mention of the Arduino name and brand should be associated with them, not me.
Based on the Arduino UNO R3 From arduino.cc

Rheingold Heavy

Sheet: /Voltage Regulator/
File: Voltage_Regulator.sch

Title: Arduino UNO R3 Clone

Size: A4	Date: 8 oct 2015	**Rev: 0**
KiCad E.D.A. kicad (5.1.4)-1		Id: 9/9

10. Apéndice II (Código)

```
;file: main.asm
.include "m328pdef.inc"
;Frecuencia 16MHz

;Macros (tienen que estar en el main)
.MACRO WRITE_FROM_ROM ;macro para cargar desde la rom, se le pasa una posicion
    MOVE_TO
    MOV ZL, R23
    MOV ZH, R24
    LDI R21, 15 ; cargo el contador de linea
FOR_WRITE_ROM:
    LPM R20, Z+ ; cargo el caracter  desde la rom
    CPI R20, 0 ; veo si es fin de texto
    BREQ END_WRITE_ROM ; si es el final me voy de la macro
    CALL WRITE_4BITS_CHARACTER ;escribo
    CPI R21, 0 ;veo si termino la linea
    BREQ ENTER_WRITE_ROM ; si termino la linea me muevo a la otra
    DEC R21 ;decremento el contador de linea
    RJMP FOR_WRITE_ROM

ENTER_WRITE_ROM:
    MOVE_TO NEXT_LINE
    LDI R21, 15
    RJMP FOR_WRITE_ROM

END_WRITE_ROM:
    ;termina la macro
.ENDM

.MACRO MOVE_TO ;macro para mover el cursor del LCD
    MOV R20, R25 ;cargo la posicion a la que me quiero mover
    ORI R20, MOVE_DDRAM ;agrego el bit MSB = 1 para el comando
    CALL WRITE_4BITS_INSTRUCTION ;llamo a mandar instruccion
.ENDM


.DSEG
DATOS: .BYTE 1024 ;datos tanto de tension como de corriente, van intercalados
PTR_DATOS: .BYTE 2 ;puntero a la posicion actual de datos
CTR_PTR_DATOS: .BYTE 2 ;contador de en que posicion estoy en datos
PENDIENTES: .BYTE 32 ;vector de pendientes
PTR_PENDIENTES_ESCRITURA: .BYTE 2 ;puntero a la posicion actual de pendientes para escribir
CTR_PTR_PENDIENTES_ESCRITURA: .BYTE 1 ;contador de en que posicion estoy de pendientes para
    escribir
PTR_PENDIENTES_LECTURA: .BYTE 2 ;puntero a la posicion actual de pendientes para leer
CTR_PTR_PENDIENTES_LECTURA: .BYTE 1 ;contador de en que posicion estoy de pendientes para leer
CTR_PENDIENTES: .BYTE  1 ;contador de cuantos hay pendientes para procesar
SUMA_TOTAL_CORRIENTE: .BYTE 4 ;suma de cuadrados de corriente
SUMA_TOTAL_TENSION: .BYTE 4 ;suma de cuadrados de tension

VALOR_FINAL_TENSION: .BYTE 64 ;vector de valores finales de tension
VALOR_FINAL_CORRIENTE: .BYTE 64 ;vector de valores finales de corriente
CTR_PTR_VALOR_FINAL: .BYTE 1 ;contador para reseteo de puntero
PTR_VALOR_FINAL_TENSION: .BYTE 2 ;puntero de vector de tension
PTR_VALOR_FINAL_CORRIENTE: .BYTE 2 ;puntero de vector de corriente


CTR_LCD: .BYTE 2 ;contador de tiempo de refresco de lcd

DESFASAJE: .BYTE 1 ;valor de desfasaje

.CSEG
;vectores de interrupcion
.ORG 0x0000
    RJMP MAIN
.ORG PCI2addr
    RJMP HANDLER_XOR
.ORG OVF0addr
    RJMP HANDLER_TIMER0
.ORG ADCCaddr
    RJMP HANDLER_ADC
.ORG ACIaddr
    RJMP HANDLER_XOR

```

```
74 .ORG INT_VECTORS_SIZE
75
76
77 MAIN:
78     ;inicializo el stack
79     LDI R16, LOW(RAMEND)
80     OUT SPL, R16
81     LDI R16, HIGH(RAMEND)
82     OUT SPH, R16
83
84     CALL CONFIGURATION_LCD
85     CALL CONFIGURAR_ADC
86
87     ;borro toda la memoria ram
88     CALL INICIALIZAR_TODO
89     ;inicializo punteros
90     CALL INICIALIZAR_PUNTERO_ADC
91
92
93     CALL INICIALIZAR_SUMAS
94
95     CALL INICIALIZAR_PTR_DATOS
96
97     CALL INICIALIZAR_PUNTERO_PENDIENTES_LECTURA
98
99     CALL INICIALIZAR_PUNTERO_VALOR_FINAL
100
101    CLR R16
102    STS CTR_PENDIENTES, R16
103
104    ;configuro desfasaje
105    CALL CONFIGURAR_DESFASAJE
106
107    ;limpio lcd e imprimo el texto (V, I, cos(phi))
108    CALL CLEAN
109    CALL TEXTO_LCD
110
111    ;activo las interrupciones
112    SEI
113
114
115
116 HERE: ;ciclo infinito de verificar si hay pendientes para procesar e imprimir
117    LDS R16, CTR_PENDIENTES
118    CLR R0
119    CPSE R16, R0
120    CALL PROCESAR
121    LDS R16, CTR_LCD + 1
122    SBRC R16, 7
123    CALL IMPRIMIR_TODO
124    RJMP HERE
125
126
127 .include "adc.asm"
128 .include "imprimir.asm"
129 .include "procesar.asm"
130 .include "lcd.asm"
131 .include "cuadrado.asm"
132 .include "raiz_cuadrada.asm"
133 .include "desfasaje.asm"
134 .include "tabla_tension.asm"
135 .include "tabla_corriente.asm"
136 .include "tabla_cosfi.asm"
137
138
139 ;texto para imprimir.
140 TEX_TENSION: .DB "V=", '\0'
141 TEX_CORRIENTE: .DB "I=", '\0'
142 TEX_COSFI: .DB "cos(phi)=", '\0'
```

codigo/main.asm

```
[U+FFFD]ile: raiz_cuadrada.asm
2
3 .include "m328pdef.inc"
4
```

```
.DEF CUADRADO_4toBYTE = R2
.DEF CUADRADO_3erBYTE = R3
.DEF CUADRADO_2doBYTE = R4
.DEF CUADRADO_1erBYTE = R5
.DEF RESTO_4 = R6
.DEF RESTO_3 = R7
.DEF RESTO_2 = R8
.DEF RESTO_1 = R9
.DEF CONT_RL = R10
.DEF CONT_ITERACIONES = R11
.DEF DIVIS_4 = R12
.DEF DIVIS_3 = R13
.DEF DIVIS_2 = R14
.DEF DIVIS_1 = R15
.DEF TEMP = R16
.DEF INIT_CUADRADO_4toBYTE = R18
.DEF INIT_CUADRADO_3erBYTE = R19
.DEF INIT_CUADRADO_2doBYTE = R20
.DEF INIT_CUADRADO_1erBYTE = R21
.DEF RESUL_4 = R22
.DEF RESUL_3 = R23
.DEF RESUL_2 = R24
.DEF RESUL_1 = R25
.DEF DIVID_4 = R22
.DEF DIVID_3 = R23
.DEF DIVID_2 = R24
.DEF DIVID_1 = R25


.EQU CANT_ITERACIONES = 6
.EQU FALTA_RESTA_4toBYTE = 2
.EQU FALTA_RESTA_3erBYTE = 1
.EQU FALTA_RESTA_2doBYTE = 0


;se le pasan valores desde R5(LSB) hasta R2(MSB), y devuelve la raiz en R21(LSB) y en R20(MSB)
RAIZ_CUADRADA:
    ;se limpian todos los registros que se vayan a usar
    CLR RESTO_4
    CLR RESTO_3
    CLR RESTO_2
    CLR RESTO_1
    CLR CONT_RL
    CLR CONT_ITERACIONES
    CLR DIVIS_4
    CLR DIVIS_3
    CLR DIVIS_2
    CLR DIVIS_1
    CLR TEMP
    CLR INIT_CUADRADO_4toBYTE
    CLR INIT_CUADRADO_3erBYTE
    CLR INIT_CUADRADO_2doBYTE
    CLR INIT_CUADRADO_1erBYTE
    CLR RESUL_4
    CLR RESUL_3
    CLR RESUL_2
    CLR RESUL_1
    CLR DIVID_4
    CLR DIVID_3
    CLR DIVID_2
    CLR DIVID_1
    LDI TEMP,CANT_ITERACIONES
    MOV CONT_ITERACIONES,TEMP;CARGO LA CANTIDAD DE ITERACIONES

;si el numero que se ingresa para hacer la raiz cuadrada es 0, directamente se devuelve 0
;para evitar hacer divisiones por 0.
VERIFICAR_CERO:
    CLR TEMP
    CP CUADRADO_1erBYTE, TEMP
    BRNE INICIALIZAR_RAIZ
    CP CUADRADO_2doBYTE, TEMP
    BRNE INICIALIZAR_RAIZ
    CP CUADRADO_3erBYTE, TEMP
    BRNE INICIALIZAR_RAIZ
    CP CUADRADO_4toBYTE, TEMP
    BRNE INICIALIZAR_RAIZ
```

```
81      RET
82
83  ;INICIALIZO LA SEMILLA SEGUN EL TAMAÑO DEL NUMERO QUE ENTRA
84  INICIALIZAR_RAIZ:
85      CLR INIT_CUADRADO_4toBYTE
86      CLR INIT_CUADRADO_3erBYTE
87      LDI INIT_CUADRADO_2doBYTE,0xFF
88      LDI INIT_CUADRADO_1erBYTE,0xFF
89  VER_4toBYTE:
90      CLR TEMP
91      CP CUADRADO_4toBYTE,TEMP
92      BRNE VER_3erBYTE
93      ANDI INIT_CUADRADO_2doBYTE,0xF0
94  VER_3erBYTE:
95      CP CUADRADO_3erBYTE,TEMP
96      BRNE VER_2doBYTE
97      ANDI INIT_CUADRADO_2doBYTE,0x0F
98  VER_2doBYTE:
99      CP CUADRADO_2doBYTE,TEMP
100     BRNE VER_1erBYTE
101     ANDI INIT_CUADRADO_1erBYTE,0xF0
102 VER_1erBYTE:
103     CP CUADRADO_1erBYTE,TEMP
104     BRNE ITERACION_RAIZ
105     ANDI INIT_CUADRADO_1erBYTE,0x0F
106 ;[FIN]INICIALIZO LA SEMILLA SEGUN EL TAMAÑO DEL NUMERO QUE ENTRA
107
108
109 ;iteracion de newton-raphson
110 ITERACION_RAIZ:
111     CALL RAIZ_NR
112     DEC CONT_ITERACIONES
113     BRNE ITERACION_RAIZ
114 RET
115
116 ;ejecuta una iteracion de newton-raphson
117 RAIZ_NR:
118     ;a
119     MOVW DIVID_3:DIVID_4,CUADRADO_3erBYTE:CUADRADO_4toBYTE
120     MOVW DIVID_1:DIVID_2,CUADRADO_1erBYTE:CUADRADO_2doBYTE
121     ;[FIN]a
122     ;X(k)
123     MOVW DIVIS_3:DIVIS_4,INIT_CUADRADO_3erBYTE:INIT_CUADRADO_4toBYTE
124     MOVW DIVIS_1:DIVIS_2,INIT_CUADRADO_1erBYTE:INIT_CUADRADO_2doBYTE
125     ;[FIN]X(k)
126     CALL DIVISION;a/X(k)
127     ;X(k) + a/X(k)
128     ADD INIT_CUADRADO_1erBYTE,RESUL_1
129     ADC INIT_CUADRADO_2doBYTE,RESUL_2
130     ADC INIT_CUADRADO_3erBYTE,RESUL_3
131     ADC INIT_CUADRADO_4toBYTE,RESUL_4
132     ;[FIN]X(k) + a/X(k)
133     ;(X(k) + a/X(k))/2
134     LSR INIT_CUADRADO_4toBYTE
135     ROR INIT_CUADRADO_3erBYTE
136     ROR INIT_CUADRADO_2doBYTE
137     ROR INIT_CUADRADO_1erBYTE
138     ;[FIN](X(k) + a/X(k))/2
139 RET
140
141 ;divide mediante el metodo de rolido. recibe el dividendo desde R25(LSB) A R22(MSB) y el
        divisor
142 ;desde R15(LSB) a R12(MSB). Devuelve el resultado desde R25(LSB) a R22(MSB).
143 DIVISION:
144 ;LIMPIAR RESTO
145     CLR RESTO_4
146     CLR RESTO_3
147     CLR RESTO_2
148     SUB RESTO_1,RESTO_1
149 ;LIMPIAR RESTO
150     LDI TEMP,33;COMENZAR LOOP CONTADOR QUE VA A HACER TANTOS ROLIDOS COMO DIGITOS HAYA
151     MOV CONT_RL,TEMP;COMENZAR LOOP CONTADOR QUE VA A HACER TANTOS ROLIDOS COMO DIGITOS HAYA
152
153 ROL_DIVIDENDO:
154 ;SHIFTEAR DIVIDENDO
155     ROL DIVID_1
```

```
156     ROL DIVID_2
157     ROL DIVID_3
158     ROL DIVID_4
159 ;[FIN]SHIFTEAR DIVIDENDO
160
161     DEC CONT_RL;DECREMENTAR CONTADOR DE SHIFTEOS
162     BRNE ROL_RESTO;SI NO SE HICIERON TODOS LOS ROLIDOS, SE PROCEDE A CONTINUAR LA DIVISION
163     RJMP REDONDEO;SI SE HICIERON TODOS LOS ROLIDOS, SE DA POR TERMINADA LA DIVISION
164
165 ROL_RESTO:
166 ;SHIFTEAR RESTO
167     ROL RESTO_1
168     ROL RESTO_2
169     ROL RESTO_3
170     ROL RESTO_4
171 ;[FIN]SHIFTEAR RESTO
172
173 ;RESTO = RESTO - DIVISOR
174     SUB RESTO_1,DIVIS_1
175     SBC RESTO_2,DIVIS_2
176     SBC RESTO_3,DIVIS_3
177     SBC RESTO_4,DIVIS_4
178 ;RESTO = RESTO - DIVISOR
179
180     BRCC INC_RES;SI EL DIVISOR ES MENOR AL RESTO, SE MANTIENE EL RESTO COMO ESTABA Y SE
            INCREMENTA EL RESULTADO
181
182 ;SI EL DIVISOR ES MAYOR AL RESTO, SE LO RESTITUYE A COMO ESTABA ANTES DE LA RESTA
183     ADD RESTO_1,DIVIS_1
184     ADC RESTO_2,DIVIS_2
185     ADC RESTO_3,DIVIS_3
186     ADC RESTO_4,DIVIS_4
187     CLC;TAMBIEN SE RESTITUYE EL CARRY
188 ;SI EL DIVISOR ES MAYOR AL RESTO, SE LO RESTITUYE A COMO ESTABA ANTES DE LA RESTA
189     RJMP ROL_DIVIDENDO;SE VUELVE A ARRANCAR EL ROLIDO DEL DIVIDENDO
190
191 INC_RES:
192     SEC;SE PRENDE EL CARRY PARA SER INCLUIDO EN EL RESULTADO
193     RJMP ROL_DIVIDENDO;SE VUELVE A ARRANCAR EL ROLIDO DEL DIVIDENDO
194
195 ;se fija si el resto es mayor o igual a la mitad del divisor, y suma 1 si es asi .
196 REDONDEO:
197     LSR DIVIS_4
198     ROR DIVIS_3
199     ROR DIVIS_2
200     ROR DIVIS_1
201     SUB DIVIS_4,RESTO_4
202     BRCS SUM_1
203     BRNE RETIRADA
204     SUB DIVIS_3,RESTO_3
205     BRCS SUM_1
206     BRNE RETIRADA
207     SUB DIVIS_2,RESTO_2
208     BRCS SUM_1
209     BRNE RETIRADA
210     SUB DIVIS_1,RESTO_1
211     BRSH RETIRADA
212 SUM_1:
213     LDI TEMP,1
214     ADD DIVID_1,TEMP
215     CLR TEMP
216     ADC DIVID_2,TEMP
217     ADC DIVID_3,TEMP
218     ADC DIVID_4,TEMP
219 RETIRADA: RET
```

codigo/raiz_cuadrada.asm

```
1 ;file: procesar.asm
2
3 .include "m328pdef.inc"
4
5 .EQU CANT_PROM = 32
6
7 INICIALIZAR_TODO: ;inicializa el vector de datos en 0
8     LDI XL, LOW(DATOS)
```

```
9       LDI XH, HIGH(DATOS)
10      LDI R16, 1
11      LDI R17, 1  ;arranca en 1 el contador (LSB)
12      LDI R18, 0 ; MSB
13      LDI R19, 8
14      CLR R0
15  FOR_INICIALIZAR_TODO:
16      ST X+, R0
17      LDI R16, 1
18      ADD R17, R16
19      ADC R18, R0
20      CPSE R18, R19 ;cuento hasta 1024
21      RJMP FOR_INICIALIZAR_TODO
22      RET
23
24  INICIALIZAR_SUMAS: ;inicializo la suma de tension y corriente en 0
25      CLR R7
26      CLR R8
27      CLR R9
28      CLR R10
29      CALL GUARDAR_SUMA_TENSION
30      CALL GUARDAR_SUMA_CORRIENTE
31      RET
32
33
34  SUMAR_3_4_BYTES: ; SE LE PASA R7(LSB) R8 R9 Y R10 (MSB) Y R23(LSB) R24 Y R25 (MSB)
35      CLR R0
36      ADD R7, R23
37      ADC R8, R24
38      ADC R9, R25
39      ADC R10, R0
40      RET
41
42  RESTAR_4_3_BYTES: ;SE LE PASA R7(LSB) R8 R9 Y R10 (MSB) Y R23(LSB) R24 Y R25 (MSB)
43      CLR R0
44      SUB R7, R23
45      SBC R8, R24
46      SBC R9, R25
47      SBC R10, R0
48      RET
49
50  PROCESAR:
51      ;leo el pendiente
52      LDS XL, PTR_PENDIENTES_LECTURA
53      LDS XH, PTR_PENDIENTES_LECTURA + 1
54      LD R16, X+
55      LD R17, X+
56      STS PTR_PENDIENTES_LECTURA, XL
57      STS PTR_PENDIENTES_LECTURA + 1, XH
58
59      ;cargo la suma total (carga en r7(LSB) hasta r10 (MSB))
60      LDS R18, CTR_PTR_DATOS ; el contador arranca en cero entonces si el numero es par es
            tension y si es impar es corriente
61      SBRS R18, 0
62      CALL CARGAR_SUMA_TENSION
63      SBRC R18, 0
64      CALL CARGAR_SUMA_CORRIENTE
65
66      ;calculo el cuadrado del viejo y guardo el nuevo (y guardo el puntero de nuevo en ram)
67      LDS YL, PTR_DATOS
68      LDS YH, PTR_DATOS + 1
69      LD R21, Y
70      ST Y+, R16
71      LD R22, Y
72      ST Y+, R17
73      STS PTR_DATOS, YL
74      STS PTR_DATOS + 1, YH
75      CALL CUADRADO ;llamo a cuadrado con R21(L) y R22(M) y me devuelve R23(L) R24 Y R25 (M)
76
77      ;le resto a la suma total
78      CALL RESTAR_4_3_BYTES
79
80      ;calculo el cuadrado nuevo
81      MOV R21, R16
82      MOV R22, R17
83      CALL CUADRADO
```

```
84
85      ;sumo el cuadrado nuevo
86      CALL SUMAR_3_4_BYTES
87
88      ;guardo la suma de nuevo
89      SBRS R18, 0
90      CALL GUARDAR_SUMA_TENSION
91      SBRC R18, 0
92      CALL GUARDAR_SUMA_CORRIENTE
93
94
95      ;sumo al contador del puntero
96      LDS R18, CTR_PTR_DATOS
97      LDS R19, CTR_PTR_DATOS + 1 ; el LSB ya lo habia cargado anteriormente
98      LDI R16, 1
99      ADD R18, R16
100     CLR R0
101     ADC R19, R0
102
103
104     ;guardo el contador del puntero
105     STS CTR_PTR_DATOS, R18
106     STS CTR_PTR_DATOS + 1, R19
107
108     ;si llego a 512 reinicio
109     CPI R19, 2
110     BREQ BRANCH_INICIALIZAR_PUNTERO_DATOS
111
112 SEGUIR_PUNTERO_DATOS:
113     ;decrementa los pendientes.
114     LDS R19, CTR_PENDIENTES
115     DEC R19
116     STS CTR_PENDIENTES, R19
117
118     ;suma al contador del puntero y reinicia el puntero si se llego al final
119     LDS R19, CTR_PTR_PENDIENTES_LECTURA
120     DEC R19
121     STS CTR_PTR_PENDIENTES_LECTURA, R19
122     BREQ BRANCH_INICIALIZAR_PUNTERO_PENDIENTES_LECTURA
123 SEGUIR_INICIALIZAR_PUNTERO_PENDIENTES_LECTURA:
124     RET
125
126 BRANCH_INICIALIZAR_PUNTERO_DATOS:
127
128     ;llamo a la raiz
129     CALL CALCULAR_VALOR_FINAL_TENSION ;calculo valor final de tension
130     CALL CALCULAR_VALOR_FINAL_CORRIENTE ;calculo valor final de corriente
131     ;y reinicio puntero de vector de valores finales si es necesario (tension y corriente)
132     CALL INICIALIZAR_PTR_DATOS
133     RJMP SEGUIR_PUNTERO_DATOS
134
135 BRANCH_INICIALIZAR_PUNTERO_PENDIENTES_LECTURA:
136     CALL INICIALIZAR_PUNTERO_PENDIENTES_LECTURA
137     RJMP SEGUIR_INICIALIZAR_PUNTERO_PENDIENTES_LECTURA
138
139
140
141
142 INICIALIZAR_PUNTERO_VALOR_FINAL:
143     LDI R16, CANT_PROM ; cantidad de valores para realizar el promedio
144     STS CTR_PTR_VALOR_FINAL, R16
145     LDI XL, LOW(VALOR_FINAL_TENSION)
146     LDI XH, HIGH(VALOR_FINAL_TENSION)
147     STS PTR_VALOR_FINAL_TENSION, XL
148     STS PTR_VALOR_FINAL_TENSION + 1, XH
149     LDI XL, LOW(VALOR_FINAL_CORRIENTE)
150     LDI XH, HIGH(VALOR_FINAL_CORRIENTE)
151     STS PTR_VALOR_FINAL_CORRIENTE, XL
152     STS PTR_VALOR_FINAL_CORRIENTE + 1, XH
153     RET
154
155
156 INICIALIZAR_PTR_DATOS: ;cambia el puntero  al inicio de datos y reinicia el contador
157     LDI XL, LOW(DATOS)
158     LDI XH, HIGH(DATOS)
159     STS PTR_DATOS, XL
```

```
160     STS PTR_DATOS + 1, XH
161     LDI R18, 0
162     LDI R19, 0
163     STS CTR_PTR_DATOS, R18
164     STS CTR_PTR_DATOS + 1, R19
165     RET
166
167
168 INICIALIZAR_PUNTERO_PENDIENTES_LECTURA: ;carga  el puntero de los pendientes de lectura al
        principio y reinicia el contador
169     LDI XL, LOW(PENDIENTES)
170     LDI XH, HIGH(PENDIENTES)
171     STS PTR_PENDIENTES_LECTURA, XL
172     STS PTR_PENDIENTES_LECTURA + 1, XH
173     LDI R16, 16
174     STS CTR_PTR_PENDIENTES_LECTURA, R16
175     RET
176
177
178 CARGAR_SUMA_TENSION: ;carga la suma de tension a los registros R7, R8, R9 y R10
179     LDI XL, LOW(SUMA_TOTAL_TENSION)
180     LDI XH, HIGH(SUMA_TOTAL_TENSION)
181     LD R7, X+
182     LD R8, X+
183     LD R9, X+
184     LD R10, X+
185     RET
186
187 GUARDAR_SUMA_TENSION: ;guarda la suma de tension desde los registros R7, R8, R9 y R10
188     LDI XL, LOW(SUMA_TOTAL_TENSION)
189     LDI XH, HIGH(SUMA_TOTAL_TENSION)
190     ST X+, R7
191     ST X+, R8
192     ST X+, R9
193     ST X+, R10
194     RET
195
196 CARGAR_SUMA_CORRIENTE:;carga la suma de corriente a los registros R7, R8, R9 y R10
197     LDI XL, LOW(SUMA_TOTAL_CORRIENTE)
198     LDI XH, HIGH(SUMA_TOTAL_CORRIENTE)
199     LD R7, X+
200     LD R8, X+
201     LD R9, X+
202     LD R10, X+
203     RET
204
205 GUARDAR_SUMA_CORRIENTE: ;guarda la suma de corriente desde los registros R7, R8, R9 y R10
206     LDI XL, LOW(SUMA_TOTAL_CORRIENTE)
207     LDI XH, HIGH(SUMA_TOTAL_CORRIENTE)
208     ST X+, R7
209     ST X+, R8
210     ST X+, R9
211     ST X+, R10
212     RET
213
214
215 CALCULAR_VALOR_FINAL_TENSION: ;primero hace la raiz cuadrada y despues divide por 16 (shift
        derecho 4 veces)
216     LDI XL, LOW(SUMA_TOTAL_TENSION)
217     LDI XH, HIGH(SUMA_TOTAL_TENSION)
218     LD R5, X+
219     LD R4, X+
220     LD R3, X+
221     LD R2, X+
222     CALL RAIZ_CUADRADA; toma en R5(L) hasta R2 (M) (ojo! alreves que todo el programa)
223     ;devuelve desde R21(L) a R20(M)
224     LSR R20 ; /2
225     ROR R21 ; /2
226     LSR R20 ; /4
227     ROR R21 ; /4
228     LSR R20 ; /8
229     ROR R21 ; /8
230     LSR R20 ; /16
231     ROR R21 ; /16
232
233     LDS XL, PTR_VALOR_FINAL_TENSION
```

```
234     LDS XH, PTR_VALOR_FINAL_TENSION + 1
235     ST X+, R21
236     ST X+, R20
237     STS PTR_VALOR_FINAL_TENSION, XL
238     STS PTR_VALOR_FINAL_TENSION + 1, XH
239
240
241     RET
242
243 CALCULAR_VALOR_FINAL_CORRIENTE:
244     LDI XL, LOW(SUMA_TOTAL_CORRIENTE)
245     LDI XH, HIGH(SUMA_TOTAL_CORRIENTE)
246     LD R5, X+
247     LD R4, X+
248     LD R3, X+
249     LD R2, X+
250     CALL RAIZ_CUADRADA; toma en R5(L) hasta R2 (M) (ojo! alreves que todo el programa)
251     ;devuelve desde R21(L) a R20(M)
252     LSR R20 ; /2
253     ROR R21 ; /2
254     LSR R20 ; /4
255     ROR R21 ; /4
256     LSR R20 ; /8
257     ROR R21 ; /8
258     LSR R20 ; /16
259     ROR R21 ; /16
260
261     ;con promedio!
262     LDS XL, PTR_VALOR_FINAL_CORRIENTE
263     LDS XH, PTR_VALOR_FINAL_CORRIENTE + 1
264     ST X+, R21
265     ST X+, R20
266     STS PTR_VALOR_FINAL_CORRIENTE, XL
267     STS PTR_VALOR_FINAL_CORRIENTE + 1, XH
268
269     ;actualizo y chequeo contador
270     LDS R16, CTR_PTR_VALOR_FINAL
271     DEC R16
272     BREQ BRANCH_INICIALIZAR_PUNTERO_VALOR_FINAL
273     STS CTR_PTR_VALOR_FINAL, R16
274 SEGUIR_INICIALIZAR_PUNTERO_VALOR_FINAL:
275
276     RET
277
278 BRANCH_INICIALIZAR_PUNTERO_VALOR_FINAL:
279     CALL INICIALIZAR_PUNTERO_VALOR_FINAL
280     RJMP SEGUIR_INICIALIZAR_PUNTERO_VALOR_FINAL
281
282
283 PROMEDIO_TENSION:
284     LDI XL, LOW(VALOR_FINAL_TENSION)
285     LDI XH, HIGH(VALOR_FINAL_TENSION)
286     LDI R16, 31
287     LD R18, X+
288     LD R19, X+
289 FOR_PROMEDIO_TENSION:
290     LD R20, X+
291     LD R21, X+
292     ADD R18, R20
293     ADC R19, R21
294     DEC R16
295     BRNE FOR_PROMEDIO_TENSION
296     LSR R19 ; /2
297     ROR R18 ; /2
298     LSR R19 ; /4
299     ROR R18 ; /4
300     LSR R19 ; /8
301     ROR R18 ; /8
302     LSR R19 ; /16
303     ROR R18 ; /16
304     LSR R19 ; /32
305     ROR R18 ; /32
306     RET
307
308 PROMEDIO_CORRIENTE:
309     LDI XL, LOW(VALOR_FINAL_CORRIENTE)
```

```
    LDI XH, HIGH(VALOR_FINAL_CORRIENTE)
    LDI R16, CANT_PROM-1
    LD R18, X+
    LD R19, X+
FOR_PROMEDIO_CORRIENTE:
    LD R20, X+
    LD R21, X+
    ADD R18, R20
    ADC R19, R21
    DEC R16
    BRNE FOR_PROMEDIO_CORRIENTE
    LSR R19 ; /2
    ROR R18 ; /2
    LSR R19 ; /4
    ROR R18 ; /4
    LSR R19 ; /8
    ROR R18 ; /8
    LSR R19 ; /16
    ROR R18 ; /16
    LSR R19 ; /32
    ROR R18 ; /32
    RET
```

codigo/procesar.asm

```
;file: lcd.asm

.include "m328pdef.inc"

;FALTA: -hacer nueva funcion que permita escribir en la posicion que se quiera (usando MOVE_TO
    )

;pines
.EQU D4 = 0 ;LSB de comunicacion de datos (se usa en modo 4 bits)
.EQU D5 = 1 ;
.EQU D6 = 2 ;
.EQU D7 = 3 ;MSB de comunicacion de datos
.EQU ENABLE = 4 ;pin de enable
.EQU RS = 5 ;pin de RS


;comandos
.EQU SET_8BITS_LONG = 0x03 ;comando para configurar modo de 8 bits
.EQU SET_4BITS_LONG = 0x02 ;comando para configurar modo de 4 bits
.EQU FUNCTION_SET = 0x2C ;comando para configurar numero de lineas, etc
.EQU SET_DISPLAY = 0x0C ;comando para configurar el display
.EQU CLEAN_DISPLAY = 0x01 ;comando para limpiar el display
.EQU ENTRY_MODE = 0x06 ;comando para configurar cursor,etc
.EQU NEXT_LINE = 0x40 ;posicion ddram de la primera posicion de segunda linea
.EQU MOVE_DDRAM = 0x80 ;comando para cambiar posicion ddram
.EQU INICIO_LCD = 0x00 ;primera posicion en el lcd
.EQU FIN_V_LCD = 2; posicion para imprimir tension
.EQU FIN_I_LCD = 8 ;posicion para imprimir corriente
.EQU FIN_NUM_V_LCD = 6 ;posicion para imprimir texto de corriente
.EQU FIN_COSFI_LCD = 0x49; posicion para imprimir coseno phi

;mascaras
.EQU MASK_4LSB = 0x0F
.EQU MASK_MSB = 0x80




.MACRO DELAY ;macro para esperar
    CLR R18
    LDI R17, @0
FOR_1:
    DEC R17
    LDI R16, @1
FOR_2:
    DEC R16
    CPSE R16, R18
    RJMP FOR_2
    CPSE R17, R18
    RJMP FOR_1
```

```
51  .ENDM
52
53
54  .MACRO WRITE_FROM_RAM ;macro para cargar desde la ram, se le pasa una posicion
55      LDI ZL, LOW(@0)
56      LDI ZH, HIGH(@0)
57      LDI R21, 15 ; cargo el contador de linea
58  FOR_WRITE_RAM:
59      LPM R20, Z+ ; cargo el caracter  desde la ram
60      CPI R20, 0 ; veo si es fin de texto
61      BREQ END_WRITE_RAM ; si es el final me voy de la macro
62      CALL WRITE_4BITS_CHARACTER ;escribo
63      CPI R21, 0 [U+FFFD]h/o´[U+0000]a linea
64      BREQ ENTER_WRITE_RAM ; si termino la linea me muevo a la otra
65      DEC R21 ;decremento el contador de linea
66      RJMP FOR_WRITE_RAM
67
68  ENTER_WRITE_RAM:
69      MOVE_TO NEXT_LINE
70      LDI R21, 15
71      RJMP FOR_WRITE_RAM
72
73  END_WRITE_RAM:
74      ;termina la macro
75
76  .ENDM
77
78
79
80
81
82  CONFIGURATION_LCD:
83      ;configuro DDRB como salida
84      LDI R16, 0x3F
85      OUT DDRB, R16
86      LDI R16, 0X80
87      OUT DDRD, R16
88      ;limpio ENABLE y RS
89      CALL CLR_ENABLE
90      CALL CLR_RS
91      DELAY 255, 255 ; espero ~15ms
92      ;INICIO CONFIGURATION PARA MODO 4 BITS (SEGUN DATASHEET)
93      LDI R20, SET_8BITS_LONG
94      CALL WRITE_8BITS
95      DELAY 85, 255 ; espero ~5ms
96      LDI R20, SET_8BITS_LONG
97      CALL WRITE_8BITS
98      DELAY 1, 255 ; espero ~100us
99      LDI R20, SET_8BITS_LONG
100     CALL WRITE_8BITS
101     DELAY 255, 255
102     LDI R20, SET_4BITS_LONG
103     CALL WRITE_8BITS
104     DELAY 255, 255
105     ;FIN CONFIGURATION PARA MODO 4 BITS
106     ;configuro la funcion (numero de lineas, etc)
107     LDI R20, FUNCTION_SET
108     CALL WRITE_4BITS_INSTRUCTION
109     DELAY 2, 255
110     ;configuro el modo de entrada (incremento, decremento, desplazamiento, etc)
111     LDI R20, ENTRY_MODE
112     CALL WRITE_4BITS_INSTRUCTION
113     DELAY 2, 255
114     ;configuro el modo de display (parpadeo de cursor, encendido, apagado, etc)
115     LDI R20, SET_DISPLAY
116     CALL WRITE_4BITS_INSTRUCTION
117     DELAY 2, 255
118     ;limpio la pantalla
119     CALL CLEAN
120     RET
121
122 CLR_ENABLE: ;limpio la salida de enable
123     IN R16, PORTB
124     ANDI R16, ~(1<<ENABLE)
125     OUT PORTB, R16
126     RET
```

```
127
128 SET_ENABLE: ; seteo la salida de enable
129     IN R16, PORTB
130     ORI R16, (1<<ENABLE)
131     OUT PORTB, R16
132     RET
133
134 CLR_RS: ;limpio la salida de RS
135     IN R16, PORTB
136     ANDI R16, ~(1<<RS)
137     OUT PORTB, R16
138     RET
139
140 SET_RS: ; seteo la salida de RS
141     IN R16, PORTB
142     ORI R16, (1<<RS)
143     OUT PORTB, R16
144     RET
145
146
147
148 WRITE_8BITS:
149     ;recibe en R20 lo que se quiere escribir, solo se usa para configurar el LCD
150     CALL SET_ENABLE ; seteo enable
151     IN R16, PORTB ; cargo PORTB
152     ANDI R20, MASK_4LSB ; dejo solo los ultimos 4
153     ANDI R16, ~MASK_4LSB ;  limpio los ultimos de PORTB
154     OR R16, R20 ; cargo R20 en R16
155     OUT PORTB, R16 ; saco PORTB
156     CALL CLR_ENABLE ; limpio ENABLE
157     RET
158
159 WRITE_4BITS_INSTRUCTION:
160     ;recibe en R20 lo que se quiere escribir
161     CALL SET_ENABLE ;activo ENABLE
162     CALL CLR_RS ; RS = 0 (estoy mandando una instruccion)
163     IN R16, PORTB ; cargo PORTB
164     MOV R17, R20 ; copio R20 a R17 para no perderlo
165     SWAP R17 ; intercambio los nibbles ya que primero se manda el nibble alto
166     ANDI R16, ~MASK_4LSB ; limpio los ultimos de R16 (PORTB)
167     ANDI R17, MASK_4LSB ; dejo solo los ultimos de R17 para no pisar a PORTB
168     OR R16, R17; copio R17 a R16
169     OUT PORTB, R16 ;saco a PORTB
170     DELAY 1, 255 ; espero algunos microsegundos
171     CALL CLR_ENABLE ; ENABLE = 0, mando el dato
172     DELAY 1, 255 ; espero algunos microsegundos
173     ;lo mismo pero enviando el nibble mas bajo
174     CALL SET_ENABLE
175     MOV R17, R20
176     ANDI R16, ~MASK_4LSB
177     ANDI R17, MASK_4LSB
178     OR R16, R17
179     OUT PORTB, R16
180     DELAY 1, 255
181     CALL CLR_ENABLE
182     DELAY 1, 255
183     RET
184
185 WRITE_4BITS_CHARACTER:
186     ;recibe en R20 lo que se quiere escribir
187     CALL SET_ENABLE;desactivo enable
188     CALL SET_RS ;activo RS (escribo datos)
189     IN R16, PORTB
190     MOV R17, R20 ;copio R20 a R17
191     SWAP R17
192     ANDI R16, ~MASK_4LSB ; limpio los ultimos de R16
193     ANDI R17, MASK_4LSB ; dejo solo los ultimos de R17
194     OR R16, R17; copio R17 a R16
195     OUT PORTB, R16 ;saco a PORTB
196     DELAY 1, 255
197     CALL CLR_ENABLE ;activo enable
198     DELAY 1, 255
199     CALL SET_ENABLE ;desactivo enable
200     MOV R17, R20 ;copio de nuevo
201     ANDI R16, ~MASK_4LSB ; limpio los ultimos de R16
202     ANDI R17, MASK_4LSB ; dejo solo los ultimos de R17
```

```
203     OR R16, R17; copio R17 a R16
204     OUT PORTB, R16 ;saco a PORTB
205     DELAY 1, 255
206     CALL CLR_ENABLE ;activo enable
207     DELAY 1, 255
208     RET
209
210 ;limpia el lcd.
211 CLEAN:
212     LDI R20, CLEAN_DISPLAY ;cargo instruccion de limpiar
213     CALL WRITE_4BITS_INSTRUCTION ; llamo a escribir
214     DELAY 85, 255 ; espero
215     RET
216
217 ;funcion que imprime el texto base en el lcd
218 TEXTO_LCD:
219     LDI R23, LOW(TEX_TENSION<<1)
220     LDI R24, HIGH(TEX_TENSION<<1)
221     LDI R25, INICIO_LCD
222     WRITE_FROM_ROM
223     LDI R23, LOW(TEX_CORRIENTE<<1)
224     LDI R24, HIGH(TEX_CORRIENTE<<1)
225     LDI R25, FIN_NUM_V_LCD
226     WRITE_FROM_ROM
227     LDI R23, LOW(TEX_COSFI<<1)
228     LDI R24, HIGH(TEX_COSFI<<1)
229     LDI R25, NEXT_LINE
230     WRITE_FROM_ROM
231     RET
```

codigo/lcd.asm

```
1  ;file: imprimir.asm
2
3  IMPRIMIR_TODO:
4      PUSH R16
5      PUSH R17
6      PUSH R18
7      PUSH R19
8      PUSH R20
9      PUSH R21
10     PUSH R23
11     PUSH R24
12     PUSH R25
13     CLR R16
14     STS CTR_LCD, R16
15     STS CTR_LCD + 1, R16
16     CALL CONVERTIR_A_VALOR_E_IMPRIMIR_TENSION
17     CALL CONVERTIR_A_VALOR_E_IMPRIMIR_CORRIENTE
18     CALL CONVERTIR_IMPRIMIR_COFF
19     POP R25
20     POP R24
21     POP R23
22     POP R21
23     POP R20
24     POP R19
25     POP R18
26     POP R17
27     POP R16
28     RET
29
30 ;lee los valores de las ultimas tensiones medidas del vector, luego promedia
31 ;y despues busca en la tabla de tension e imprime con la macro WRITE_FROM_ROM
32 CONVERTIR_A_VALOR_E_IMPRIMIR_TENSION:
33     LDI R23, LOW(TABLA_TENSION<<1)
34     LDI R24, HIGH(TABLA_TENSION<<1)
35
36     CALL PROMEDIO_TENSION; CALCULA EL PROMEDIO DE LA TENSION Y DEVUELVE EN R23 (L) Y R24(M)
37
38     LSL R18
39     ROL R19
40     LSL R18
41     ROL R19
42
43     ADD R23, R18
44     ADC R24, R19
```

```
45      LDI R25, FIN_V_LCD
46      WRITE_FROM_ROM
47      RET
48
49  ;lee los valores de las ultimas corrientes medidas del vector, luego promedia
50  ;y despues busca en la tabla de corriente e imprime con la macro WRITE_FROM_ROM
51  CONVERTIR_A_VALOR_E_IMPRIMIR_CORRIENTE:
52      LDI R23, LOW(TABLA_CORRIENTE<<1)
53      LDI R24, HIGH(TABLA_CORRIENTE<<1)
54
55      CALL PROMEDIO_CORRIENTE
56
57      LDI R16, 6
58      MUL R18, R16
59      ADD R23, R0
60      ADC R24, R1
61      MUL R19, R16
62      ADD R24, R0
63      LDI R25, FIN_I_LCD
64      WRITE_FROM_ROM
65      RET
66
67  ;lee el valor de desfasaje de la ram y luego busca en la tabla e imprime mediante la macro
68  ;WRITE_FROM_ROM
69  CONVERTIR_IMPRIMIR_COFF:
70      PUSH R0
71      PUSH R1
72      LDI XL, LOW(DESFASAJE)
73      LDI XH, HIGH(DESFASAJE)
74      LDI R23, LOW(TABLA_COSFI<<1)
75      LDI R24, HIGH(TABLA_COSFI<<1)
76      LD R18, X
77      LDI R16, 6
78      MUL R18, R16
79      ADD R23, R0
80      ADC R24, R1
81      LDI R25, FIN_COSFI_LCD
82      WRITE_FROM_ROM
83
84      POP R1
85      POP R0
86      RET
```

codigo/imprimir.asm

```
1   ;file: desfasaje.asm
2
3   .include "m328pdef.inc"
4
5
6   .EQU MASK_LECTURA_TENSION = 0x04
7   .EQU PENDIENTE = 10
8   .EQU ORDENADA_ORIGEN = 25
9   .EQU CONFIG_ACSR = 0x08
10
11  .DEF TEMP2 = R17
12  .DEF TEMP1 = R16
13  .DEF TIEMPO_ARRIBA_L = R19
14
15
16  CONFIGURAR_DESFASAJE:
17
18      LDI TEMP2, CONFIG_ACSR
19      OUT ACSR, TEMP2
20
21  ;SETEO EL PUERTO D DE ENTRADA
22      LDI TEMP2,DDRD
23      ANDI TEMP2,MASK_LECTURA_TENSION
24      OUT DDRD,TEMP2
25  ;[FIN]SETEO EL PUERTO D DE ENTRADA
26
27  ;HABILITO LA INTERRUPCION POR PIN 5 DEL PUERTO D
28      LDS TEMP2,PCMSK2
29      ORI TEMP2,(1<<PCINT21)
30      STS PCMSK2,TEMP2
31  ;[FIN]HABILITO LA INTERRUPCION POR PIN 5 DEL PUERTO D
```

```
32
33 ;HABILITO LA INTERRUPCION POR PUERTO D
34       LDS TEMP2,PCICR
35       ORI TEMP2, (1<<PCIE2)
36       STS PCICR,TEMP2
37 ;[FIN]HABILITO LA INTERRUPCION POR PUERTO D
38
39       LDS TEMP2,TCCR1A;SETEO PARA QUE EL TIMER CUENTE LO MAS QUE PUEDA
40       ANDI TEMP2,0x00;SETEO PARA QUE EL TIMER CUENTE LO MAS QUE PUEDA
41       STS TCCR1A,TEMP2;SETEO PARA QUE EL TIMER CUENTE LO MAS QUE PUEDA
42
43
44 ;INICIALIZACION DE VARIABLES
45       CLR TIEMPO_ARRIBA_L
46
47
48 ;CONFIGURO EL INTERRUPT DEL TIMER
49       LDS TEMP2, TIMSK1
50       ORI TEMP2, 0
51       STS TIMSK1, TEMP2
52
53 RET
54 ;[FIN]INICIALIZACION DE VARIABLES
55
56 SET_TIMER1:
57
58       LDI TEMP2,0x02;PRESCALEO Y PRENDO EL TIMER
59       STS TCCR1B,TEMP2;PRESCALEO Y PRENDO EL TIMER
60 RET
61
62 UNSET_TIMER1:
63       LDI TEMP2,0x00;APAGO EL TIMER
64       STS TCCR1B,TEMP2;APAGO EL TIMER
65 RET
66
67 PRENDER_TIMER:
68       CLR TEMP2
69       STS TCNT1H, TEMP2
70       STS TCNT1L, TEMP2
71       NOP
72       NOP
73       NOP
74       NOP
75       NOP
76       NOP
77       CALL SET_TIMER1
78       RET
79
80
81
82 APAGAR_Y_GUARDAR_TIMER:
83       CALL UNSET_TIMER1 ; apago el timer
84       CALL GUARDAR_TIMER ; guardo el valor del timer
85       RET
86
87 GUARDAR_TIMER:
88       LDS TEMP2,TCNT1L
89       LDS TEMP1,TCNT1H
90       ;SHIFTEO  6 A LA DERECHA, PARA ARMAR UN BIT MEZCLA DEL BIT LOW Y HIGH.
91       LSR TEMP2
92       LSR TEMP2
93       LSR TEMP2
94       LSR TEMP2
95       LSR TEMP2
96       LSR TEMP2
97       ;SHIFTEO 2 A LA IZQUIERDA
98       LSL TEMP1
99       LSL TEMP1
100      OR TEMP2, TEMP1
101      CPI TEMP2, 15
102      BRLO SEGUIR_FILTRO
103      STS DESFASAJE, TEMP2
104 SEGUIR_FILTRO:
105
106 RET
107
```

```
HANDLER_XOR:
    PUSH TEMP1
    PUSH TEMP2
    PUSH TIEMPO_ARRIBA_L
    IN TEMP2, SREG
    PUSH TEMP2

    IN TEMP1, ACSR
    IN TEMP2, PIND
    EOR TEMP1, TEMP2
    SBRC TEMP1, 5
    CALL PRENDER_TIMER
    CALL APAGAR_Y_GUARDAR_TIMER

    POP TEMP2
    OUT SREG, TEMP2
    POP TIEMPO_ARRIBA_L
    POP TEMP2
    POP TEMP1

    RETI
```

codigo/desfasaje.asm

```
;file: cuadrado.asm

.include "m328pdef.inc"

.DEF aux_1 = R19


; Multiplica R21-R22 * R21-R22 y lo pone en R23-R24-R25
CUADRADO:
    PUSH aux_1
    PUSH R0
    PUSH R1
    CLR aux_1
    CLR R25
    MUL R21, R21
    MOV R23, R0
    MOV R24, R1

    MUL R22, R21
    ADD R24, R0
    ADC R25, aux_1
    ADD R24, R0
    ADC R25, aux_1
    ADD R25, R1
    ADD R25, R1

    MUL R22, R22
    ADD R25, R0


    POP R1
    POP R0
    POP aux_1
RET
```

codigo/cuadrado.asm

```
;file: adc.asm

;comandos ADC
.EQU ADMUX_CONFIG = 0x40
.EQU ADCSRA_CONFIG = 0x88
.EQU DIDR0_CONFIG = 0x3F



CONFIGURAR_ADC:

    ;configuro los pines de DDRC como entrada
    IN R16, DDRC
    LDI R16, 0x00
    OUT DDRC, R16
```

```
16
17      ;configuro el timer0 Y ADC
18
19      IN R16, TCCR0A
20      ANDI R16, 0x00
21      OUT TCCR0A,R16
22
23
24      LDI R16, 0x02
25      OUT TCCR0B, R16
26
27
28      LDI R16, 0x01
29      STS TIMSK0, R16
30
31      LDI R16, ADMUX_CONFIG
32      STS ADMUX, R16
33      LDI R16, ADCSRA_CONFIG
34      STS ADCSRA, R16
35      LDI R16, DIDR0_CONFIG
36      STS DIDR0, R16
37
38      RET
39
40
41  CLEAN_ADIF:
42      LDS R16, ADCSRA
43      ORI R16, (1<<ADIF)
44      STS ADCSRA, R16
45      RET
46
47  START_ADC:
48      LDS R16, ADCSRA
49      ORI R16, (1<<ADSC)
50      STS ADCSRA, R16
51      RET
52
53  HANDLER_TIMER0:
54      PUSH R16
55      IN R16, SREG
56      PUSH R16
57      PUSH R17
58      PUSH R18
59      PUSH R19
60      LDI R16, 181
61      OUT TCNT0, R16
62      CALL START_ADC
63      LDS R16, CTR_LCD + 1
64      SBRS R16, 7
65      CALL AUMENTAR_CONTADOR_LCD
66
67      POP R19
68      POP R18
69      POP R17
70      POP R16
71      OUT SREG, R16
72      POP R16
73      RETI
74
75  AUMENTAR_CONTADOR_LCD:
76      CLR R16
77      LDI R17, 1
78      LDS R18, CTR_LCD
79      LDS R19, CTR_LCD + 1
80      ADD R18, R17
81      ADC R19, R16
82      STS CTR_LCD, R18
83      STS CTR_LCD + 1, R19
84      RET
85
86
87  HANDLER_ADC:
88      PUSH R16 ;pusheo lo que uso
89      IN R16, SREG
90      PUSH R16
91      PUSH R17
```

```
92      PUSH R26
93      PUSH R27
94      LDS XL, PTR_PENDIENTES_ESCRITURA ;cargo el puntero que habia guardado antes
95      LDS XH, PTR_PENDIENTES_ESCRITURA + 1
96      LDS R16, ADCL ; guardo la parte baja del ADC
97      ST X+, R16 ; lo guardo en ram y sumo
98      LDS R17, ADCH ;guardo la parte alta del ADC
99      ST X+, R17 ; lo guardo en ram
100     STS PTR_PENDIENTES_ESCRITURA, XL ; vuelvo a guardar el puntero
101     STS PTR_PENDIENTES_ESCRITURA + 1, XH
102     LDS R17, CTR_PENDIENTES ; sumo al contador de pendientes para que se entere el programa
103     INC R17
104     STS CTR_PENDIENTES, R17
105     LDS R17, CTR_PTR_PENDIENTES_ESCRITURA ; sumo al contador para saber cuando reiniciar el
            puntero
106     DEC R17
107     STS CTR_PTR_PENDIENTES_ESCRITURA, R17 ; guardo el contador
108     BREQ BRANCH_INICIALIZAR_PUNTERO_ADC ; si es cero reinicio el puntero
109 SEGUIR_INICIALIZAR_PUNTERO_ADC:
110     LDS R16, ADMUX ;cargo el multiplexor del adc
111     LDI R17, 1
112     EOR R16, R17 ;hago el xor para intercambiar entre tension y corriente
113     STS ADMUX, R16
114     POP R27 ; popeo todo de nuevo
115     POP R26
116     POP R17
117     POP R16
118     OUT SREG, R16
119     POP R16
120
121     RETI
122
123
124 BRANCH_INICIALIZAR_PUNTERO_ADC:
125     CALL INICIALIZAR_PUNTERO_ADC
126     RJMP SEGUIR_INICIALIZAR_PUNTERO_ADC
127
128
129 INICIALIZAR_PUNTERO_ADC:
130     LDI XL, LOW(PENDIENTES) ;cargo el puntero de pendientes
131     LDI XH, HIGH(PENDIENTES)
132     STS PTR_PENDIENTES_ESCRITURA, XL ;guardo el puntero en mi espacio para el puntero
133     STS PTR_PENDIENTES_ESCRITURA + 1, XH
134     LDI R16, 16 ;reinicio el contador
135     STS CTR_PTR_PENDIENTES_ESCRITURA, R16; guardo el contador
136     RET
```

codigo/adc.asm

11. Apéndice III: Presupuesto del proyecto

- Display LCD 16x2: $230
- Sensor de efecto Hall ACS712 ± 5A: $220
- Comparador LM393: $15
- Arduino UNO: $500
- Amplificador Operacional LM358: $25
- Transformador 220V-18V: $1000
- Tomacorriente 220V: $30
- Enchufe 220V: $70
- 2 Presets 100 KΩ: $90

Total: $2180