# UNIVERSIDAD TECNOLOGICA NACIONAL
# FACULTAD REGIONAL SAN NICOLAS

**INGENIERIA EN ELECTRONICA**

# PROBLEMA DE INGENIERÍA

**TECNICAS DIGITALES III**

# Descargador automático de baterías


Integrantes**:**

- Pasqualetti Fermín
- Tomasi Darío Germán

Docentes**:**

- Profesor: Poblete Felipe
- Auxiliar: González Mariano


**AÑO 2010**

# INDICE

# OBJETIVOS DEL TRABAJO

El siguiente trabajo tiene como finalidad desarrollar un sistema que pueda conocer la capacidad de una batería o acumulador eléctrico.
Para conocer dicha capacidad se realizará un ensayo de descarga controlada (corriente constante), hasta obtener una tensión mínima en los bornes de la batería.
Un ensayo de estas características demanda demasiado tiempo, por lo tanto es de gran utilidad que el sistema sea automático.

# MATERIAS INTEGRADAS

- **Sistemas de Control I**

Con respecto a esta materia, se encuentra incluido el diseño de un PWM, PID, Anti windup y utilizamos los conocimientos de control de una planta (llamado así al circuito diseñado).

- **Técnicas Digitales II**

Con respecto a la materia antes nombrada, hemos utilizado los conocimientos adquiridos referidos al manejo y utilización del microcontrolador Motorola. Del cual detallaremos sus partes mas adelante.

- Electrónica Aplicada II

Con respecto a los conocimientos adquiridos en la materia mencionada podemos citar el diseño de la fuente de entrada y etapa de amplificación.

# POSIBLES APLICACIONES

• Está destinado al ensayo de baterías, para poder saber sobre la vida útil restante, su capacidad, voltaje entregado y obtener una curva de capacidad, la cual se puede comparar con la entregada por el fabricante para sacar conclusiones sobre la misma.

# BIBLIOGRAFÍA


• *Apuntes.*

1- Comunicación - MAX232 - Conversor TTL-RS232
Descripción por Eduardo J. Carletti

• *Sitios de Internet.*

1- CI MAX232 http://pdfserv.maxim-ic.com/en/ds/MAX220-MAX249.pdf
2- Microcontrolador Motorola MCHC908JK8
http://www.electrocomponentes.com/Ingenieria/Microcontroladores/Curso%20HC908/Parte_2_Capitulo%202.pdf
3- Puerto serie http://es.wikipedia.org/wiki/RS-232
4- Software para diseño de circuitos: Eagle - www.cadsoftusa.com (se encuentra tutorial en español)
5- Matlab http://www.mathworks.com/
6- LM324N http://www.datasheetcatalog.org/datasheets/208/62529_DS.pdf

# DESARROLLO

## *INTRODUCCIÓN*

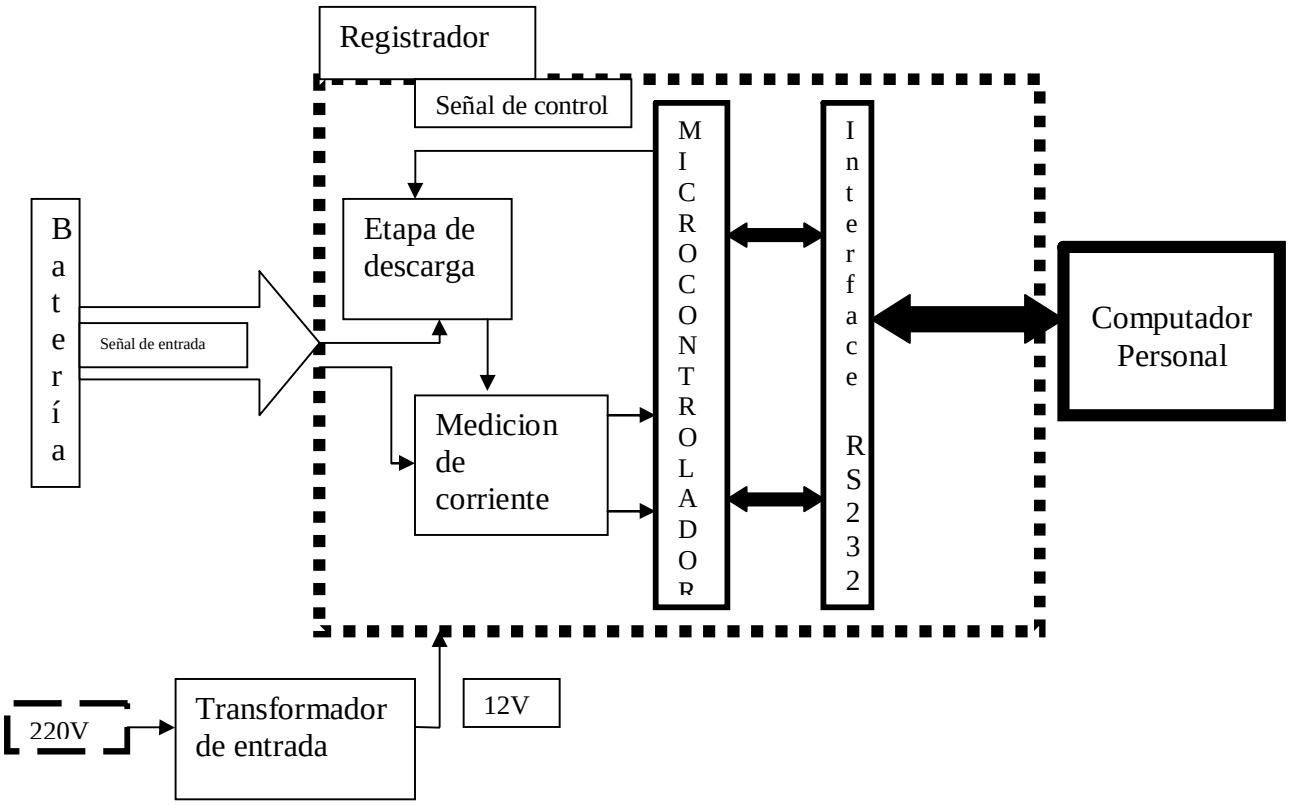



**Figura 1**

El diagrama de bloques anterior (figura 1) representa el diseño del Descargador de baterías realizado con los conocimentos adquiridos en las materias mencionadas mas arriba, además de incluir en este caso la comunicación con una PC, para así incorporar los conocimientos aportados por la materia Técnicas Digitales III.

Mediante estos, hemos logrado realizar una interfaz con una PC, para así poder visualizar y controlar mediante esta última, el funcionamiento del sistema completo, quedando de esta manera más práctico y sencillo de utilizar para cualquier usuario, aunque el mismo no posea conocimientos sobre la materia o bien no posea conocimientos sobre como realizar un ensayo de baterías. El sistema nos solicitara en pantalla los datos necesarios para poder comenzar con el ensayo. Una vez ingresados estos, comienza el ensayo y el usuario podrá visualizar en pantalla el resultado en tiempo real y obtener un resultado final para poder realizar un informe, sacar conclusiones sobre la batería, y demás datos importantes como comparar la curva obtenida con la entregada por el fabricante.

El sistema se origina como consecuencia de necesitar conocer la vida útil restante en nuestras baterías de apoyo, las cuales pueden ser utilizadas para alimentación de luces de emergencias, como también en UPS. En nuestro sistema como detallaremos mas adelante, contamos con diferentes etapas las cuales nos llevaran al resultado final, que es conocer la capacidad de la batería ensayada. En estas etapas, descriptas en detalle en Teoría de Funcionamiento, podemos ir adelantando que hemos usado etapas de potencia, diseñadas para trabajar con distintas tensiones, y en este punto podemos decir que

el sistema, consta con un transformador de entrada, el cual trabaja con una alimentación de 220V, y nos entrega 12V, ambas tensiones en corriente alterna. La salida del mismo es ingresada a una etapa rectificadora para así obtener los 12V de corriente continua para nuestro sistema. Con lo comentado podemos ver que nuestro sistema fácilmente se puede transformar en un equipo portable, ya que en realidad trabaja con 12V. Hasta aquí no hemos entrado en profundidad a nuestro diseño, ya que se pretende entregar un pantallazo general del funcionamiento.
Tenemos que partir para nuestro ensayo sabiendo que el mismo esta pensado para baterías de una tensión de 6V y hasta 5A/H. Para iniciar nuestro ensayo debemos establecer las condiciones iniciales, y esta es principalmente que la batería esta cargada completamente. Generalmente, como comentamos anteriormente, los lugares donde se utilizan estas baterías las mismas no van a estar descargadas por completo ya que seguramente poseen una carga de flotación, lo cual significa que poseen un sistema que evita que estas se descarguen.
Continuando con nuestro viaje hacia el interior del sistema, este también cuenta con un microprocesador Motorola, el cual es programado para poder efectuar distintas tareas, de todas ellas podemos destacar la utilización de su salida que es un PWM (modulador de ancho de pulso), que será quien nos permita manejar nuestra corriente de descarga, ya que como comentamos al principio se trata de un ensayo de descarga mediante corriente controlada. Esta corriente será disipada en un par de resistencias, las cuales están calculadas para soportar también la elevada temperatura que ellas deben disipar por el paso de la corriente a través de ellas.
Como sabemos hemos tenido que realizar una modificación a nuestra señal de 12V de corriente continua ya que las etapas amplificadoras, como nuestro microprocesador, trabajan con 5V de corriente continua.
Continuando con nuestra investigación, a partir de la cual nos surgió la idea de realizar el descargador de baterías, se nos presento el problema de que tenemos 2 tipos de baterías que son las más comunes, alcalinas o ácidas. Estos 2 tipos sobre los cuales se encuentra una reseña de su construcción y características mas abajo, nuestra investigación nos llevo a saber que los ensayos se deben realizar a una temperatura de trabajo para la cual la batería fue diseñada. Con estos nuevos datos realizamos las modificaciones necesarias en nuestra programación para poder en nuestra pantalla inicial solicitarle al operario que ingrese la temperatura a la cual va a realizar el ensayo y así nuestro sistema realiza una corrección por temperatura, para aproximar los resultados a la realidad.
Les comentaremos una breve descripción de la pantalla de inicio la cual podrán apreciar en el anexo. La misma cuenta con la posibilidad de ingresar: marca, tipo y número de la batería, si esta es alcalina o ácida, como así la capacidad y voltaje de la misma, luego los parámetros de descarga: corriente de descarga o porcentaje de corriente de descarga, luego podremos seleccionar si la tensión final es automática o manual, para lo cual nuestra programación contempla el porcentaje máximo de descarga al que se deben llevar las baterías para no dañarlas. Por último debemos ingresar la temperatura de trabajo, para así nuestro sistema pueda realizar la compensación necesaria.
A medida que avanzamos en nuestro desarrollo, comenzamos a pensar de qué manera sería más amigable la utilización del sistema, para lo cual recurrimos a la implementación del potente software MatLab. A partir de aquí es que se nos abrió un abanico de posibilidades. Optamos por desarrollar un software que nos permitiera ingresar los datos, como mencionamos antes en la pantalla principal. Esta idea nos llevo a realizar investigaciones sobre como comunicarnos con la PC a nuestro sistema. Surgió la problemática de las diferentes tensiones y lenguajes con los cuales trabajamos en nuestro sistema con el microprocesador Motorola y el lenguaje máquina. Para lo cual llegamos a la solución de implementar la comunicación mediante el puerto serie de nuestra PC. La comunicación con este puerto la realizamos con la implementación de un CIMAX232, el cual veremos su funcionamiento mas adelante como así también podremos ver su hoja de datos.
Una vez concretado nuestro objetivo, la construcción del sistema y la comunicación con la PC, llego el momento de ponerlo en funcionamiento y ensayar una batería. De aquí en adelante trabajamos en la obtención del resultado más semejante a la realidad. Esto pudimos hacerlo ya que nuestro sistema nos

arroja en tiempo real los datos necesarios de tensión y corriente instantánea, como también el tiempo transcurrido de ensayo. Todos estos datos nos van entregando una gráfica, la cual quedará en pantalla al finalizar el ensayo. Dijimos de realizarlo semejante a la realidad, ya que podemos valernos en algunos modelos de baterías de las gráficas entregadas por los fabricantes y datos de sus baterías, por este motivo es que podemos comparar los datos obtenidos con nuestro sistema con estos antes mencionados. El sistema además de mostrar el resultado final, nos ofrece la posibilidad de guardar el archivo, para disponer de este en el momento que creamos necesario.
Mas adelante veremos un ejemplo completo de ensayo y a la vez proporcionaremos una muestra de la hoja de datos de un fabricante, para así los lectores podrán ver la similitud de nuestros resultados. Estos ejemplos de ensayo completo se pueden ver en el anexo del presente informe.

## *RESEÑA SOBRE BATERÍAS*

### ¿Qué es una batería?

Una batería es un dispositivo electroquímico el cual almacena energía en forma química. Cuando se conecta a un circuito eléctrico, la energía química se transforma en energía eléctrica. Todas las baterías son similares en su construcción y están compuestas por un número de celdas electroquímicas. Cada una de estas celdas está compuesta de un electrodo positivo y otro negativo además de un separador. Cuando la batería se está descargando un cambio electroquímico se está produciendo entre los diferentes materiales en los dos electrodos. Los electrones son transportados entre el electrodo positivo y negativo vía un circuito externo (bombillas, motores de arranque etc.).

### Tecnología básica de la batería

Para ser capaz de comparar los distintos tipos de batería y comprender las diferencias es necesario tener una idea del funcionamiento básico de una batería.

Superficialmente la tecnología de la batería puede parecer simple. Sin embargo actualmente los procesos electroquímicos que se utilizan en una batería son bastante complejos. Existen diferentes factores que determinan el funcionamiento de una batería.

## EL PLOMO Y EL ÁCIDO

A pesar del gran esfuerzo realizado en investigación de los diferentes tipos de materiales las baterías de plomo ácido son las preferidas e insuperables por el amplio de aplicaciones que tienen. El plomo es abundante y no demasiado caro y es por esta razón por la cual es idóneo para la producción de baterías de buena calidad en grandes cantidades.

## ¿QUÉ ES UNA BATERÍA DE PLOMO ÁCIDO?

Las primeras baterías de plomo-ácido (acumuladores de plomo), fueron fabricadas a mediados del siglo XIX por Gastón Planté. Hoy en día todavía son uno de los tipos de baterías más comunes. Se descubrió que cuando el material de plomo se sumergía en una solución de ácido sulfúrico se producía un voltaje eléctrico el cual podía ser recargado.

Este tipo de baterías es único en cuanto que utiliza el plomo, material relativamente barato, tanto para la placa positiva como para la negativa.

El material activo de la placa positiva es óxido de plomo ($PbO_2$).

El de la placa negativa es plomo puro esponjoso y el electrolito está disuelto en (H2SO4).

Cuando hablamos de material activo en las baterías de ácido de plomo, nos referimos al óxido de plomo y al plomo esponjoso.

## DIVERSOS TIPOS DE BATERÍAS DE PLOMO ÁCIDO

La tecnología del plomo ácido puede variar según las diferentes necesidades existentes. Las baterías se clasifican en grupos según el uso que estas tengan y por su diseño. Las diferencias principales entre estos grupos se dan por la estructura y diseño de los electrodos (ó placas), el material activo y el electrolito.

Los tipos más comunes de baterías de plomo más comunes son:
- Baterías de tracción: para carretillas elevadoras, sillas de ruedas eléctricas y automóviles eléctricos.
- Baterías estacionarias: para fuentes de alimentación de emergencia y fuentes de alimentación ininterrumpida para usos de informática (UPS).
- Baterías de arranque: para arrancar automóviles y otros vehículos de motor diesel y gasolina.

Además de estos hay baterías especiales para otras áreas tales como control remoto, herramientas portátiles, motores de carretillas etc.

## DIFERENCIAS DE LA CONSTRUCCIÓN.

- Baterías de tracción. Las baterías de tracción están sujetas a una constante y relativamente pequeña descarga, durante largos periodos de tiempo, lo que supone un alto grado de descarga. Hay que procurar recargarlas, preferiblemente de 8 a 16 horas cada día antes de que se vuelvan a descargar.

Las baterías de tracción tienen electrodos muy gruesos con rejillas pesadas y un exceso de material activo.

- Baterías estacionarias. Las baterías estacionarias están constantemente siendo cargadas y se debe tener cuidado de evitar que se sequen. El electrolito y el material de la rejilla del electrodo están diseñados de forma que se minimice la corrosión.

- Baterías de arranque. Tienen que ser capaces de descargar el máximo de corriente posible en un corto espacio de tiempo manteniendo un alto voltaje. Tienen que ser capaces de aguantar muchas descargas incluso con cambios fuertes de temperatura. El peso, el diseño y la forma son también características determinantes.

Para poder cumplir su tarea principal que es arrancar un motor, se necesita mucha energía en un periodo corto de tiempo. Las baterías de arranque tienen generalmente una baja resistencia interna.

Esto puede lograrse con una gran área de superficie de electrodo, un pequeño espacio entre placas y unas conexiones "heavy-duty" (resistentes a duros servicios) entre celdas.

## *TEORÍA DE FUNCIONAMIENTO*

En nuestra introducción ofrecimos un adelanto del funcionamiento del equipo. En esta etapa, describiremos con mayor profundidad el funcionamiento de cada una de las etapas que compone nuestro sistema completo.
Partiendo de nuestra batería con carga, podremos conectarla a nuestro sistema para comenzar el ensayo. En este caso tendremos dos opciones para ejecutar el software, la primera, teniendo instalado MatLab en la PC podremos abrir este software y ejecutar nuestro ensayo. La segunda opción, la cual contempla que en una PC no se encuentre instalado el sistema MatLab, es realizar la ejecución del archivo .exe, previo a que como condición inicial para este caso debemos tener en la PC la carpeta de las librerías necesarias para el funcionamiento de este ejecutable.
Una vez que dimos inicio al software, y este nos pide los datos de la batería a ensayar, estamos en condiciones de poder observar la evolución del ensayo. Para un detalle mas profundo de su diseño es que realizamos la siguiente teoría de funcionamiento.
El funcionamiento del sistema se basa en la adquisición de datos tomados desde la planta (circuito completo), los cuales son registrados por el microcontrolador y enviados al PC mediante la transmisión de estos al puerto serie.
En el circuito podemos ver diversas etapas, las cuales son detalladas a continuación:

1- Etapa de entrada, a la cual llegan los 12V de corriente alterna provenientes del transformador de entrada, los cuales son rectificados y se pasan por un filtro, entra a un regulador de 12V y cuya salida entra a un regulador de 5V. En este caso colocamos un Led, el cual nos advierte que la etapa de fuente esta funcionando. De aquí es que necesitamos los 12V para alimentar los amplificadores operacionales y los 5V de corriente continua con los cuales alimentamos el microcontrolador y el CI MAX232.
2- Etapa de descarga, compuesta por un par de resistencias en paralelo y un transistor. Por las resistencias pasa la corriente que es uno de los datos necesarios para el control de descarga, la cual vuelve hacia la batería, cerrando su camino. Las resistencias mencionadas generan calor por el paso de la corriente a través de ellas.
3- Etapa de medición de corriente, en esta etapa se calcula la misma mediante los datos tomados desde los bornes de la batería y los datos en las resistencias de descarga, se calculada por medio de la medición de la tensión que cae sobre el par de resistencias antes mencionado cuyo valor es constante. Este proceso lo realizamos mediante dos operacionales incluidos en el CI LM324N (datasheet en sitios de Internet 6). Uno de ellos toma la tensión en batería y el otro la tensión que cae sobre la resistencia, podemos ver un par de potenciómetros los cuales se utilizan para regular la ganancia de la etapa.
4- Etapa del microcontrolador, en este caso utilizamos un microcontrolador Motorola MCHC908JK8 (datasheet en sitios de Internet 2), el cual programamos para poder realizar las cuentas y conversiones necesarias. Esta programación se puede ver en el anexo.
5- Etapa de comunicación, la cual consta de un CI MAX232 (datasheet en anexo y sitios de Internet 1) el cual nos permite la comunicación con el PC, tanto para el envío como para la recepción de datos.
6- Etapa de control, la salida del microcontrolador, utilizando el PWM y filtrando el mismo nos da el nivel medio del pulso que sale del PWM que es la corriente de control que llega a la base del transistor de descarga.

Los datos se obtienen al conectar la batería al sistema, la cual será descargada mediante la etapa descripta anteriormente drenando corriente de la misma. Este sistema, se inicia al recibir el pulso que da inicio al ensayo, proveniente del PC. Este dato es interpretado por el microcontrolador el cual inicia el ensayo y comienza el registro de los datos aportados por el circuito, los cuales envía con la tasa de transferencia especificada por la norma del puerto serie. El microcontrolador se encuentra configurado

con una previa realización del programa necesario para realizar las funciones correspondientes (ver programa en anexo), como ser la utilización del temporizador para realizar un PWM, cuya salida se conecta a la entrada positiva de un operacional, una vez filtrada. A la salida de este, se encuentra una configuración Darlington con una resistencia de 1.5 OHM en el emisor. En el colector se encuentra conectada la batería. Esto significa que la base del transistor se encuentra conectada a la salida del operacional y el negativo del operacional se encuentra conectado al emisor del transistor, esto se realiza para corregir mediante la realimentación el valor de la juntura igual a 0.7V, lo cual nos impediría obtener control entre 0V y 0.7V.
Refiriéndonos a la corriente de descarga, ésta debe ser controlada, lo que obtenemos colocando un operacional que tomando los datos de tensión en el par de resistencias, nos permite controlar la corriente de base del transistor del circuito de descarga. Entonces podemos decir que estamos manipulando la corriente mediante tensión.
Como queremos implementar este circuito para utilizarlo con una PC, es decir, queremos computarizar los resultados y el desarrollo del ensayo, debemos establecer la comunicación con el PC. Para esto debemos vincular los sistemas, para lo cual recurrimos a la comunicación mediante el puerto serie (Fig. 2)

Fig. 2 (ver anexo para mas información)

Para realizarlo es que ya hemos elegido nuestro microcontrolador, el cual posee un modulo de comunicación serie.
Como los datos utilizados por el microcontrolador están en valores TTL, es decir, que para su uso en el PC no son aptos, es por eso que se utiliza el CI MAX232 (Fig. 3),

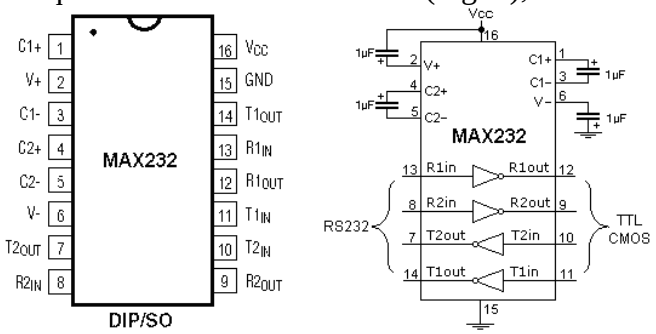
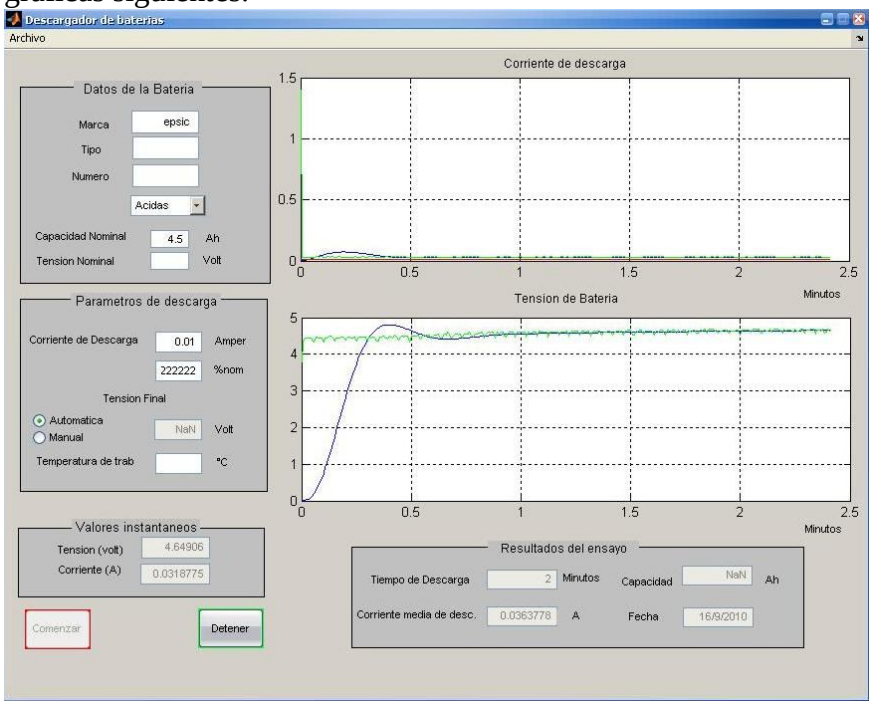


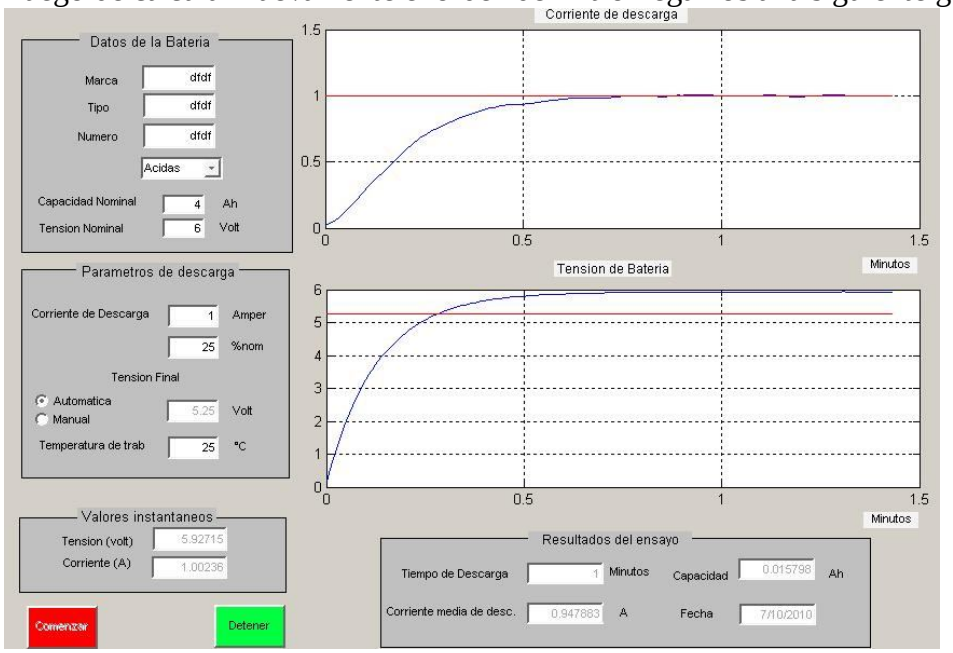

Fig. 3

el cual es capaz de convertir los valores TTL a lenguaje maquina que son los que puede manejar y utilizar el PC. La forma de trasmisión que utilizamos mediante el puerto serie es asincrónica y usamos la trasmisión de datos en el modo full duplex.
Con estos datos y utilizando una programación realizada con el software MATLAB (ver programa en anexo), con el cual podemos utilizar el puerto serie del PC y así registrar los datos enviados por la planta a través de la conversión antes mencionada, se puede realizar e ir visualizando en tiempo real la curva de ensayo, ya que tomamos datos aportados por el sistema y estos son bs que nos permiten enviar datos nuevamente al microcontrolador para poder variar el ancho al PWM antes mencionado, el

cual era el que emitía la señal de control, entonces como vemos, el control lo realizamos mediante el PC, a través del envío de datos por el puerto serie.
Para poder realizar la curva mencionada, se debe ejecutar el programa, para lo cual hemos creado un archivo .exe, el cual con las librerías adecuadas instaladas en cualquier PC, podemos utilizar para realizar los distintos ensayos, aunque en estas PCs no se encuentre instalado software alguno de los utilizados para la programación y puesta a punto del sistema.
Para comenzar con este, el programa, nos pide una serie de datos sobre la batería, como ser: tipo, capacidad en Ah(Ampère-hora), valor máximo de descarga, refiriéndose a este como el valor crítico al cual se quiere llevar la descarga de la batería, ya que estas, tienen un limite de descarga, bajo el cual la misma deja de ser útil, es decir que con el sistema se pueden realizar ensayos para medir la vida útil y el estado de la batería, como así también se pueden realizar ensayos destructivos.
Volviendo al inicio del programa, una vez iniciado el mismo, después de haber cargado los datos solicitados, este comenzará a mostrarnos en pantalla los resultados.
Los datos ingresados al iniciar el programa a medida que este se esta ejecutando podremos cambiarlos si es que necesitamos o queremos realizar el ensayo con una mayor o menor velocidad.
Cuando la batería llega al punto de máxima descarga, ingresado por nosotros o algún operario, el sistema da por terminado el ensayo, quedando en pantalla la gráfica de la curva obtenida, como así los valores de los parámetros por nosotros impuestos al iniciar el mismo. El programa nos permite generar un archivo para guardarlo y poder así consultarlo en cualquier momento que se lo solicite, conservando la información y curva obtenida.
El programa utilizado para realizar la interfaz gráfica, que es la que podemos observar en pantalla, el que nos solicita los datos de la batería, datos de máxima descarga, nos permite cambiar datos iniciales durante el funcionamiento es el MatLab. Con este potente software es que al ir ensayando el sistema pudimos observar que la evolución de la curva tenía un pequeño sobrevalor en su inicio hasta que lograba estabilizarse. Esto nos llevo a pensar en que lugar estaría el inconveniente, llegando a la conclusión de que podíamos estar utilizando un tipo de filtro digital el cual no era conveniente para nuestro caso. Como sabemos, disponemos de tres opciones de filtrado, sin filtro, filtrado ideal y filtrado excesivo. Nuestro filtro en primera instancia con los valores utilizados nos arrojaba las gráficas siguientes:

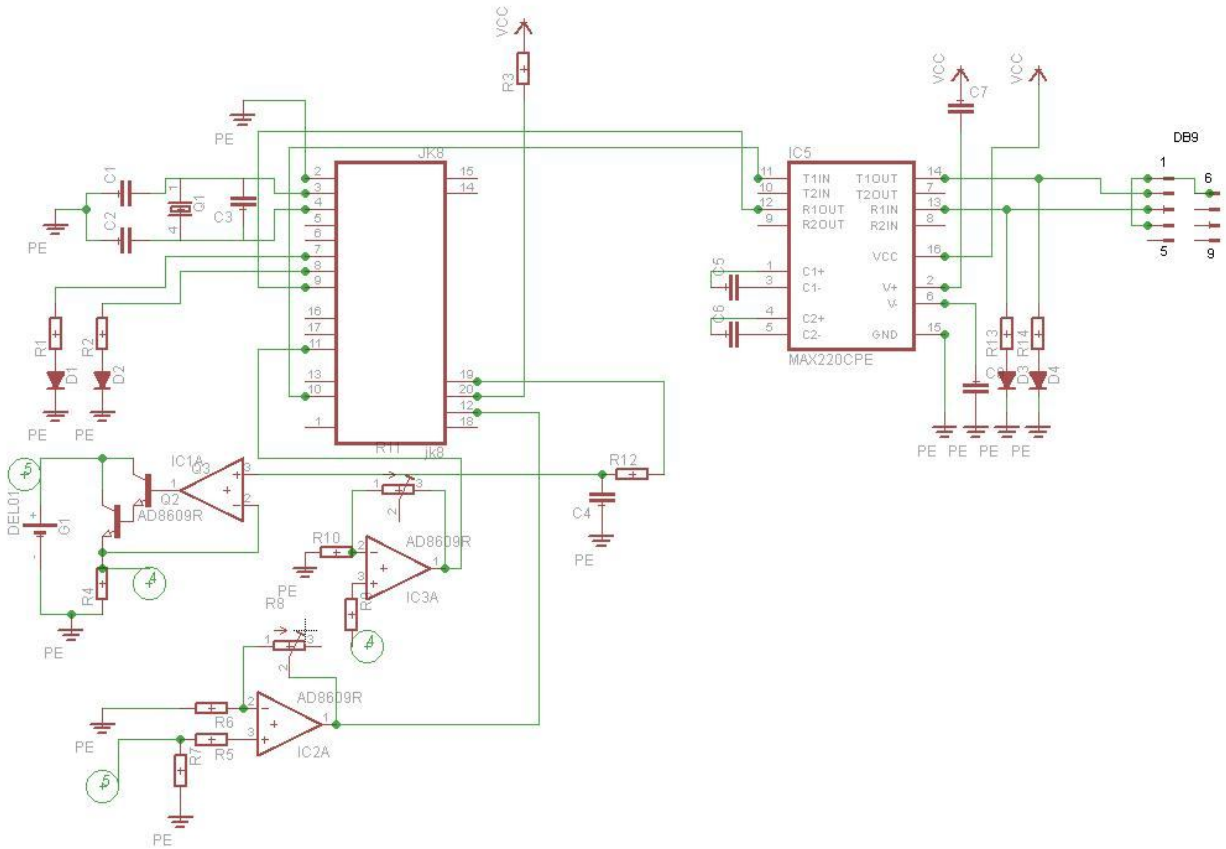


Donde podemos ver el efecto del sobrevalor, es decir, que en nuestro afán de filtrar la onda nos excedimos en el orden del filtro.
Luego de calcular nuevamente el orden del filtro llegamos a la siguiente gráfica:



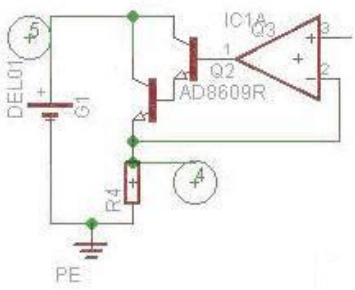

En la cual podemos ver que el sobrevalor no se encuentra, esto lo logramos modificando los parámetros de nuestro filtro digital, utilizando un filtro Butterworth (IIR) de primer orden y frecuencia de corte 0.02Hz.
Este software, además, lo hemos utilizado para manipular el puerto serie, para el envío y recepción de datos, como así también mediante este hemos podido desarrollar un ejecutable, mencionado mas arriba para realizar el ensayo en cualquier PC, instalando las librerías correctas de MatLab.
En conclusión, este potente software es el que da inicio al ensayo una vez que realizamos la carga de los datos solicitados, como así también es el que finaliza el mismo cuando se llega al límite de descarga impuesto por el usuario. Hemos también realizado la opción de guardar un archivo con los datos finales del ensayo, para así poder consultarlo en cualquier momento.

## *CIRCUITOS*

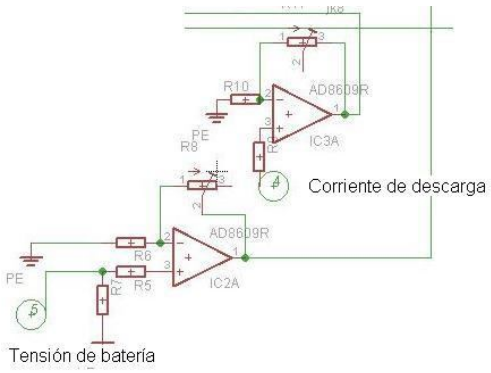



Circuito completo

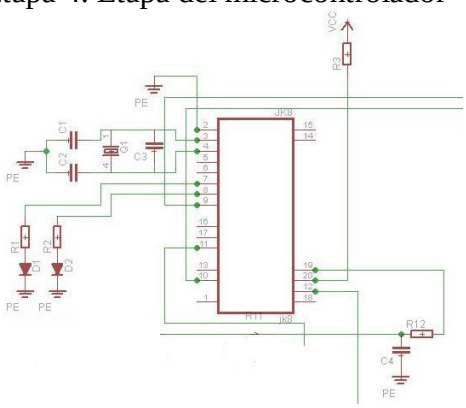
A continuación se identifican las etapas mencionadas en el circuito.
Etapa 2: Etapa de descarga



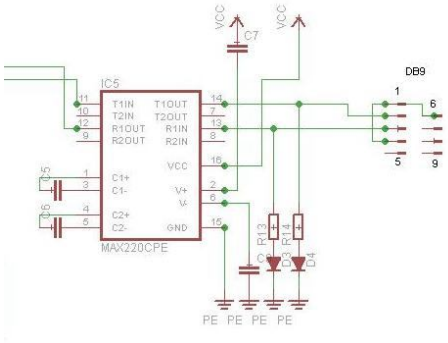

Etapa 3: Etapa de descarga



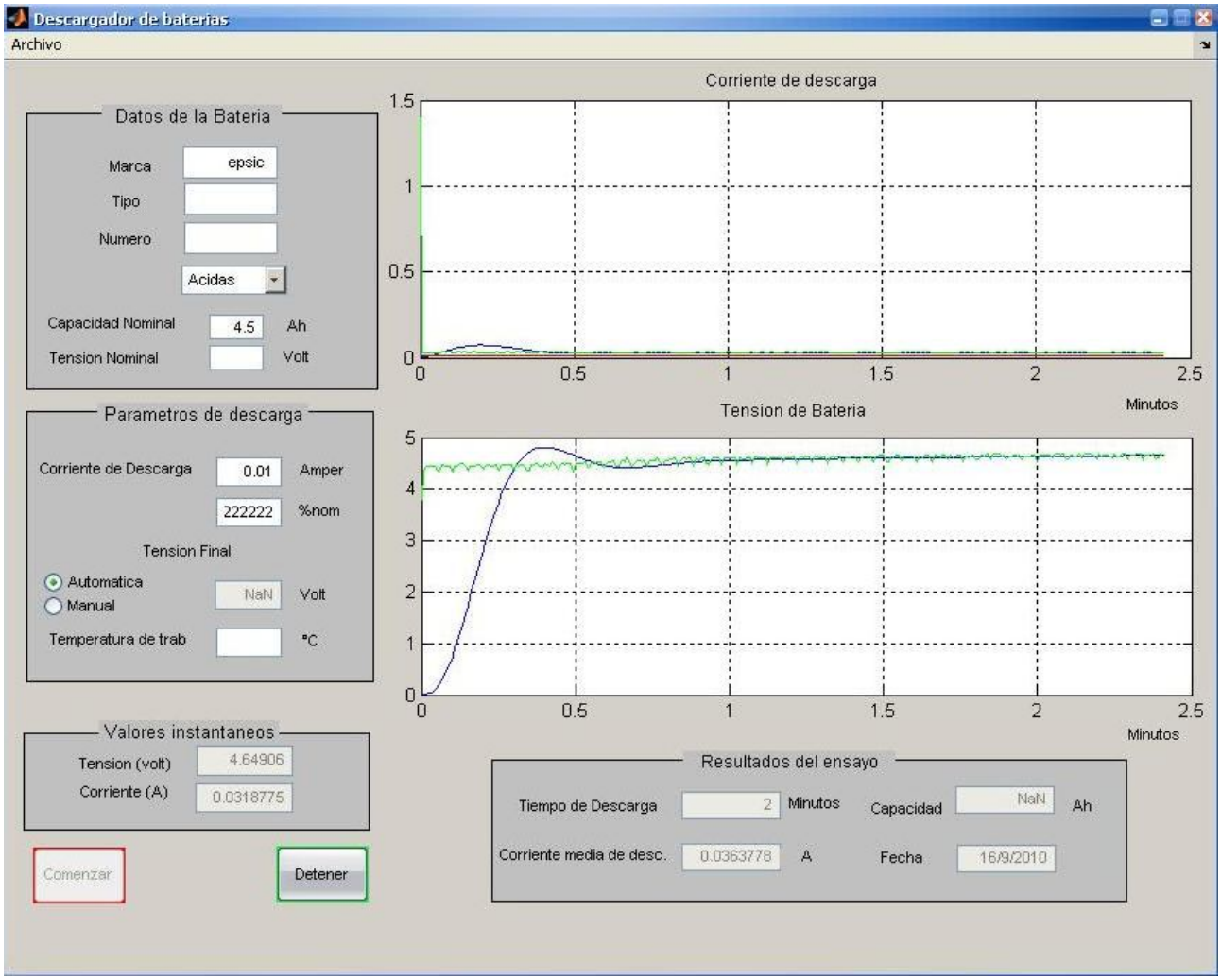

Etapa 4: Etapa del microcontrolador



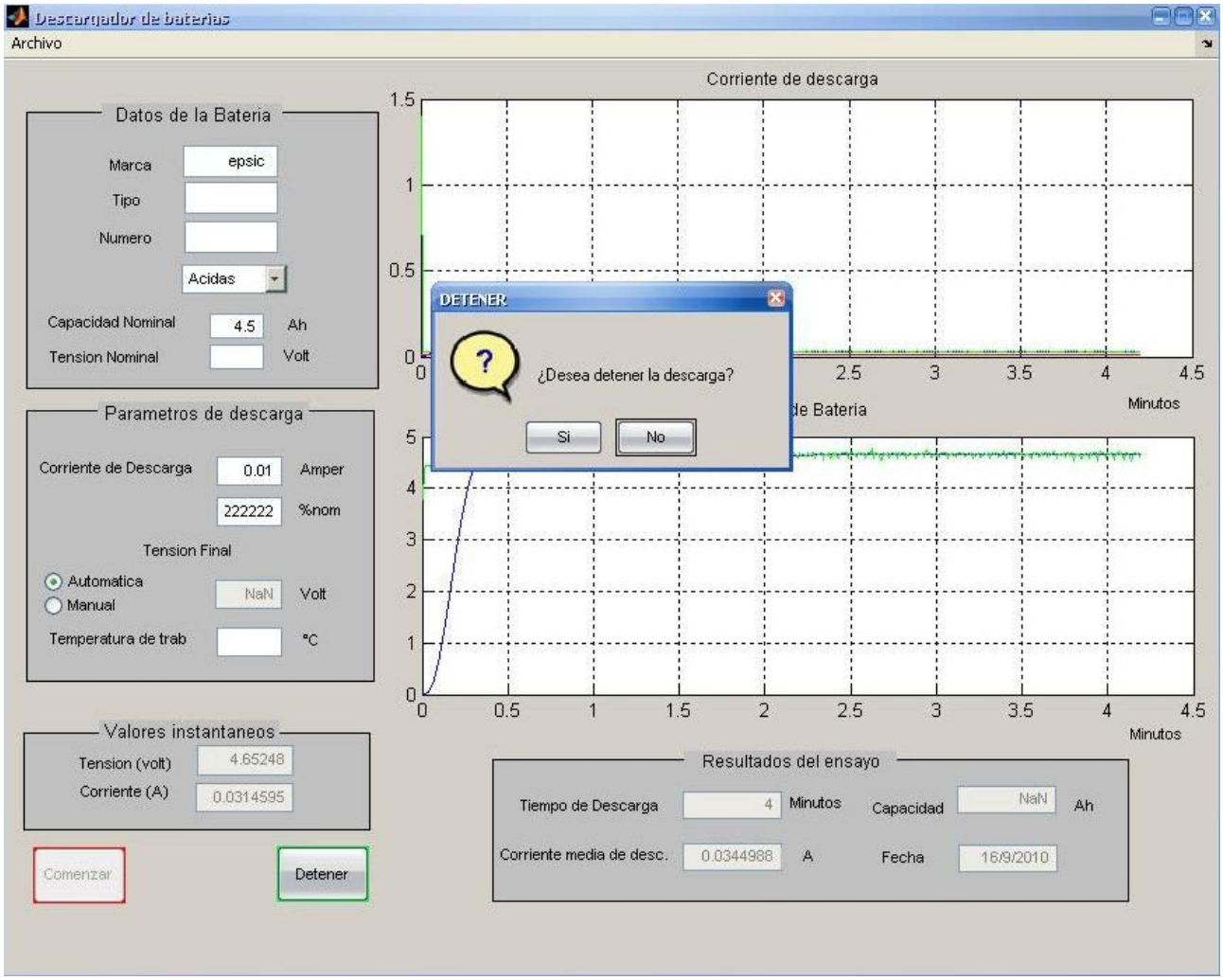

Etapa 5: Etapa de comunicación



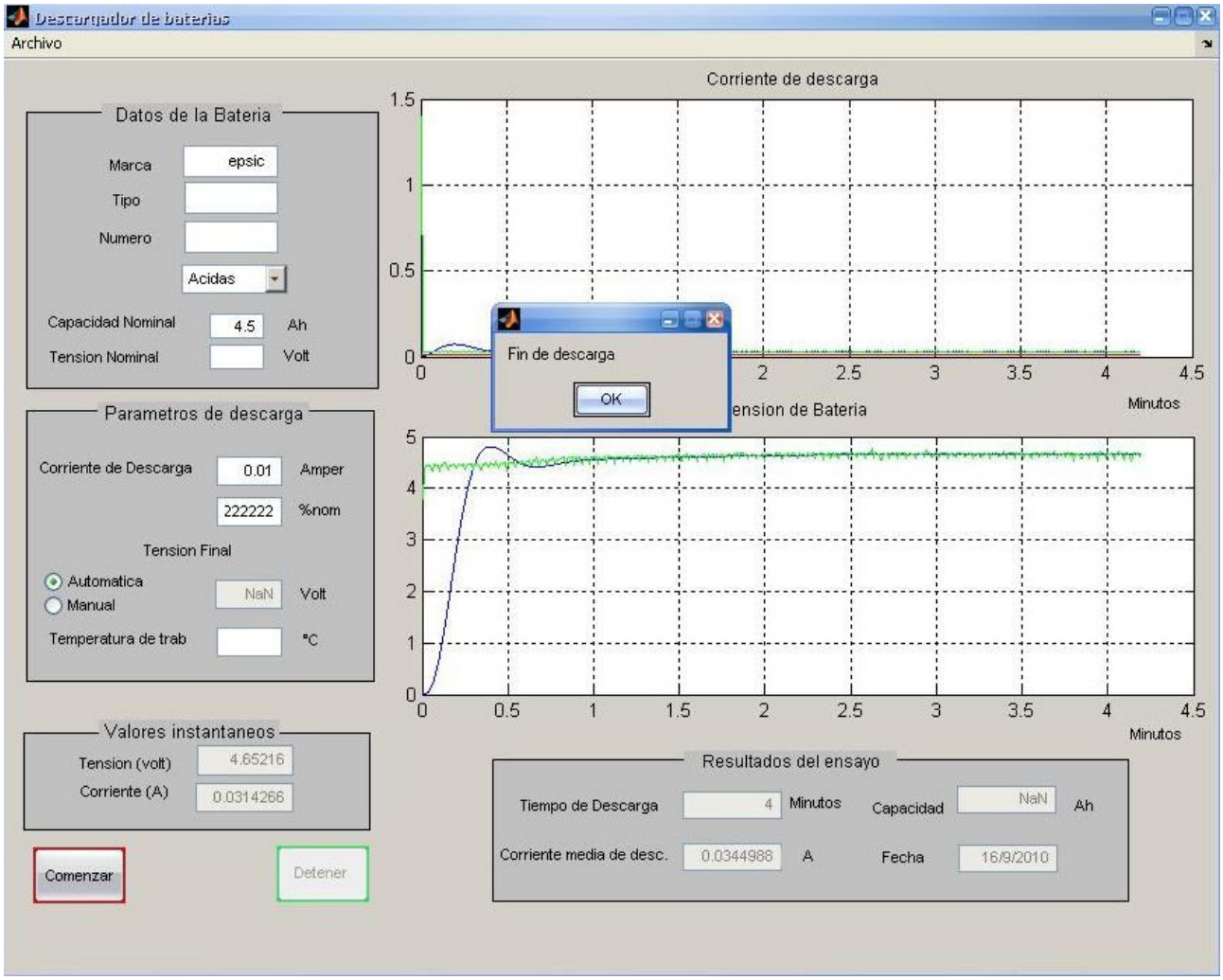
## *RESULTADOS DE LAS PRUEBAS*



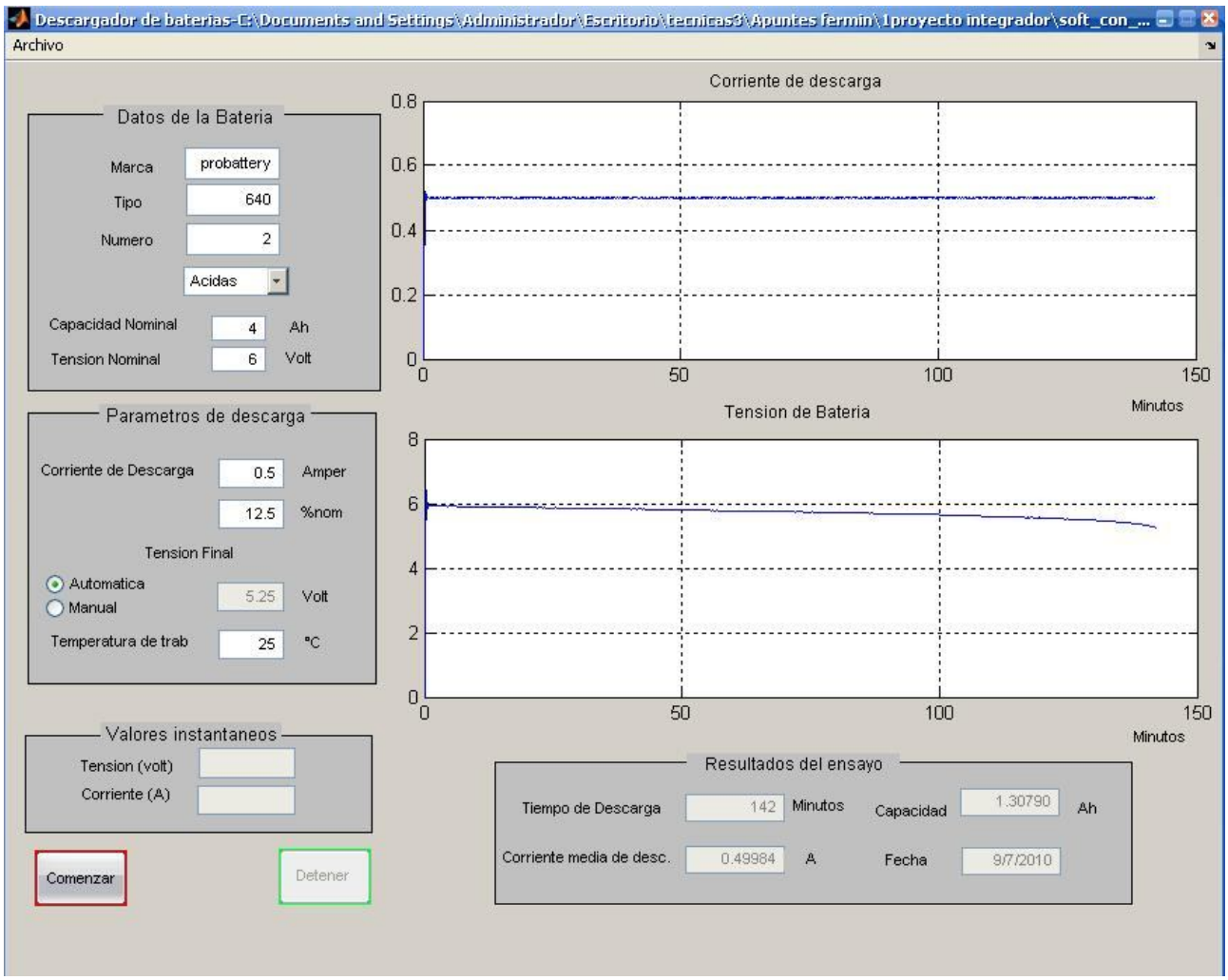

Aquí podemos ver el ensayo en sus comienzos con una corriente de descarga de 0.01A. Podemos observar los valores instantáneos de corriente y tensión.




En este caso observamos como podemos decidir si detener el ensayo en el momento deseado, el cual no se detiene hasta que se confirme, pudiendo continuar si esto se desea.




Aquí vemos un ejemplo de ensayo de descarga completado, el software desarrollado nos indica el final del mismo.




En este caso se realizo el ensayo según datos aportados por el fabricante de la batería, con respecto a la corriente de descarga a aplicar en el ensayo, obteniendo como resultado la curva anterior, la cual coincide con la hoja de datos del fabricante.

## *CONCLUSIONES*

Mediante el diseño del sistema pudimos obtener el conocimiento necesario para la comunicación con una PC y así poder enviar y recibir datos hacia y desde la misma.
El sistema completo, fue ensayado varias veces y comparados los resultados obtenidos a través del tiempo, con las gráficas entregadas por los fabricantes. Estas comparaciones nos arrojaron datos valiosos con respecto a la vida útil y capacidad de las baterías, como también pudimos comprobar un muy buen funcionamiento del sistema diseñado.
Obtuvimos una buena respuesta a los cambios de valores, mientras el sistema se encontraba en funcionamiento, actuando de manera precisa y correcta en la evolución de las curvas y una correcta generación del archivo con los datos del ensayo realizado.

# ANEXOS:

## *COMO REALIZAR UN ENSAYO CON EL DESCARGADOR DE BATERÍAS*

Para realizar el ensayo de descarga a una batería es esencial que la misma se encuentre totalmente cargada.
A continuación se detallaran los pasos a seguir.

1. conectar el equipo: cable de alimentación y cable de comunicación con la PC (puerto serie).
2. encender el equipo: se iluminaran cuatro Leds, tres rojos (testigo de fuente, Tx y Rx) y uno verde (equipo en stand by).
3. ejecutar el software mediante el archivo "pruebaguide.exe". Se observara la pantalla principal.
4. conectar la batería, a analizar, al equipo.
5. cargar los datos que solicita el software:
    - marca, tipo y numero de la batería
    - si esta es acida o alcalina
    - su capacidad y tensión nominal
    - la temperatura a la cual se efectuará el ensayo
    - la tensión a la que se debe llegar para finalizar el ensayo. Podrá ser automática (acidas 1.75vpc o alcalinas 1vpc) o manual (el usuario ingresa la tensión )
    - corriente de descarga. En ampere a porcentaje de la capacidad nominal.
6. Hacer clic en "comenzar" para iniciar la descarga. (se apaga el Led verde y se enciende uno rojo indicando "descarga en proceso").

Realizado estos pasos el instrumento permanecerá descargando la batería y visualizando, por medio de dos graficas, la evolución de la corriente de descarga y de la tensión en bornes. Esta última será de utilidad al final del ensayo.
También se puede ver la corriente y tensión instantáneas.
En el bloque "resultados de ensayo" de la pantalla principal se mostrara el tiempo transcurrido, la capacidad acumulada y la corriente media de descarga.
El ensayo terminara cuando la tensión en bornes de la batería alcance el valor mínimo ingresado.
Además el ensayo se puede detener en cualquier momento con el botón "Detener".

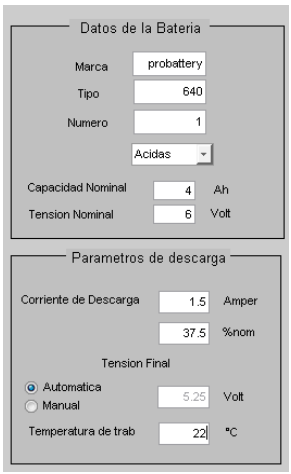
7. Una vez terminado, mediante el menú archivo-guardar se puede guardar un archivo con los datos obtenidos.
8. Los datos relevantes son la grafica de tensión de batería y el resultado final de la capacidad. Con estos se debe realizar una comparación con los datos aportados por el fabricante.

Veremos dos ejemplos de ensayos reales.
Ejemplo 1

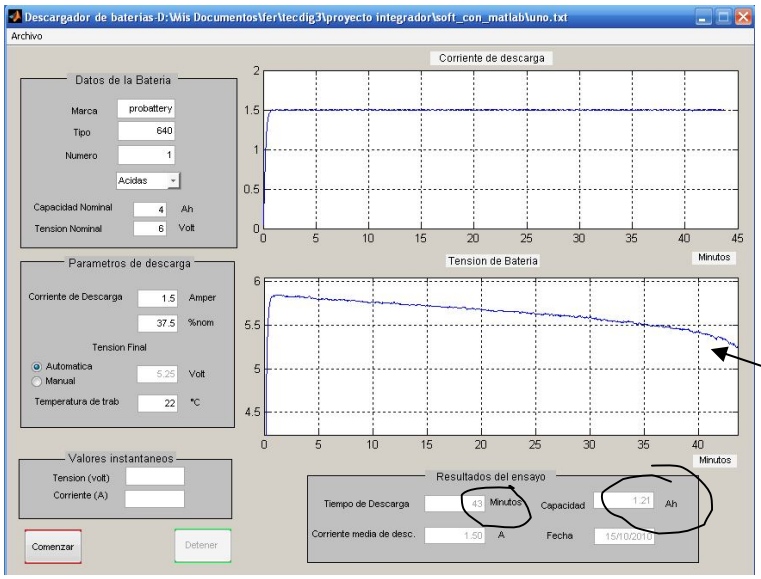
Los pasos 1, 2, 3 y 4 no los mostramos por su simplicidad.

El paso 5 consiste en cargar los datos

Marca: Probattery
Tipo: 640
Numero: 1
Batería: acida
Capacidad nominal: 4 Ah
Tensión nominal: 6 volt.
Tensión minima: Automática 5.25volt
Temperatura: 22 ºC
Corriente de descarga: 1.5 A




Paso 6, se hace clic en “Comenzar” y comienza el ensayo.
El proceso de descarga, dependiendo del estado de la batería, puede tardar varias horas.
Al finalizar se puede guardar los datos obtenidos



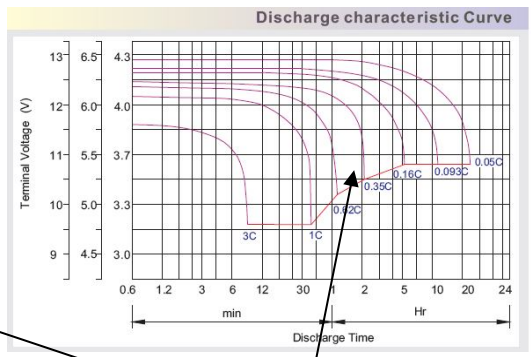



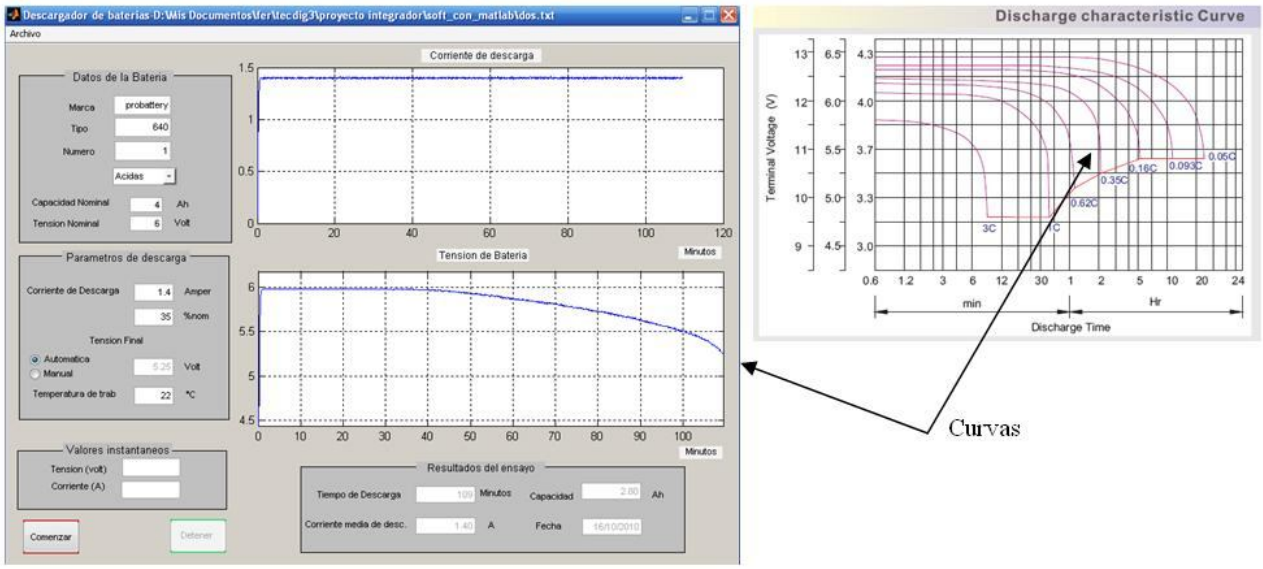

Figura1

curvas

Pantalla del Descargador de batería una vez finalizado el ensayo

La pantalla anterior muestra la capacidad acumulada (1.21ah de 4ah nom) y las curva de tensión que compararemos con la proporcionada por el fabricante para ver cuan alejada esta de lo ideal.
La curva que debemos observar, en la figura1, es la obtenida a 35% de la capacidad nominal (0.35C en la grafica.)
Vemos que el tiempo de descarga, en la grafica del fabricante, es de 2horas (120min) mientras que el tiempo de nuestra descarga es de 43 minutos.
La relación entre los valores nos revela que la batería bajo análisis tiene un 30% de la capacidad nominal.
Dependiendo de la aplicación de este acumulador se decidirá si debe ser reemplazado o no.

Ejemplo 2
El ensayo siguiente es realizado a una batería igual a la del ensayo anterior.
Por lo tanto los datos a cargar son los mismos a excepción de la corriente de descarga.
A esta batería la descargaremos drenándole una corriente correspondiente al 35% de su capacidad nominal (1.4A).
Los resultados




Comparando las curvas vemos que la batería ensayada conserva el 91% de la capacidad nominal.

# *LISTADOS DE PROGRAMAS*

## Software del microcontrolador

```
*********************************************************
*                     Proyecto para Técnicas Digitales 3  *
*                      Descargador de baterías            *
*********************************************************


$Include 'jl8regs.inc'

comienzoRam     EQU  $0060
comienzoRom     EQU  $dc00
Vectorreset     EQU  $FFFE
vector_receptor EQU  $ffE4
vector_timer      EQU  $fff2
vector_tch0 EQU  $fff4


copd            equ  0
coco            equ  7

 org comienzoRam


 org comienzoRom
configuraciones:
                bset copd,config1           ;deshabilita el contador de
Watchdog
                rsp
                clra
                clrx
                clrh

inicvariable:

                                            ;inicializa el conversor
                mov #%00000000,adclk        ;divide por 1 al reloj interno de
1megahz. 4megahz el cristal.
                                  ;inicializa el sci
            mov #%110000,scbr     ;divido el clock del bus 64*13 para obtener
1202baudios
            mov #%1000000,scc1          ;habilito el sci con 8 bit de datos,sin
paridad
            mov #%0100100,scc2          ;habilito el receptor

            mov #%00100000,tsc          ;inicializo el timer divido por 1 el
reloj del bus,desabilitola interrupcion
            mov #$00,tmodh              ;cargo  el valor 100 en el modulo
contador para que tarde 100useg, y generar una señal de 10khz
            mov #100t,tmodl
            mov #%00011010,tsc0         ;configuro el canal cero de tim PWM,
            clr ptb                     ;inicializa los puertos
            mov #%1100000,ddrb          ;configura los pines 5 y 6 del puerto B
como salida
            mov #%10000,ddrd

*************************************
*          Lazo principal            *
*************************************
```

```
inicio:           bclr 4,ptd
            bset 5,tsc           ;paro el timer
            mov #$00,tch0h             ;cargo 0 en el registro del canal para
generar un ciclo de trabajo de cero.
            mov #00t,tch0l
            mov #%0011111,adscr         ;apago el adc
            bclr 3,scc2          ;apago el transmisor
            bset 6,ptb           ;enciendo Led de ESPERA
            bclr 5,ptb           ;apago led de ON

            cli                  ;habilito las interrupciones
            wait                 ;espero la primera interrupcion
            bset 5,ptb           ;enciendo led de ON
            bclr 6,ptb           ;apago Led de ESPERA
            bclr 5,tsc           ;comienzo con el pwm


            mov #%0100100,adscr         ;enciendo el adc y lo
configuro(canal4,convercion continua,
                                        ;deshabilita la interrupción)
espero          brclr coco,adscr,espero    ;espero primera conversión para que
se estabilice el adc
                lda adr
nuevamuestra    bclr coco,adscr

                mov #%0100100,adscr         ;enciendo el adc y lo
configuro(canal4,convercion continua,
                                        ;desabilita la interrupcion)
                lda #1t                   ;paso al retardo 1ms
            jsr ret                    ;llamo subrutina de retardo
                bclr coco,adscr
espero1         brclr coco,adscr,espero1   ;segunda conversión
            lda adr
                bclr coco,adscr

            bset 3,scc2          ;habilito el transmisor
otro        brclr 7,scs1,otro    ;espero hasta que se pueda transmitir otro dato

            sta scdr             ;paso el dato de corriente del conversor al tx

                bclr coco,adscr
            mov #%0100011,adscr         ;enciendo el adc y lo
configuro(canal3,conversion continua,
                                        ;desabilita la interrupcion)

espero3         brclr coco,adscr,espero3   ;segunda conversion
            lda adr
                bclr coco,adscr

otro1       brclr 7,scs1,otro1         ;espero hasta que se pueda transmitir
otro dato
            sta scdr             ;paso el dato de tensión del conversor al tx
                                 ;espero 500 ms para enviar otras dos muestras
            lda #250t            ;paso al retardo 250ms
            jsr ret                    ;llamo subrrutina de retardo
                lda #250t              ;paso al retardo 250ms
            jsr ret                    ;llamo subrrutina de retardo

            jmp nuevamuestra

****************************************
```

```
*      rutina de interrupcion del sci rx *
****************************************

recepcion
              psha                      ;salvo A, X, y H
                pshx
                pshh
            lda scs1
            lda scdr           ;leo el dato recibido
            cbeqa #150t,inicio       ;si es el dato de fin de ensayo, termino de
adquirir
            sta tch0l          ;modifico el ciclo de trabajo


            pulh
            pulx
            pula
                rti                     ;retorno de interrupcion

****************************************
*           subrrutina de retardo 1mseg  *
****************************************

ret

vuelvo1     ldx #99t                  ;el bucle dura 1mseg
vuelvo      nop                       ;mediante el acumulador se pasa a la
subrrutina
            nop                       ;la cantidad de veces que debe repetir el
bucle
            nop                       ;consiguiendo retardos múltiplos de 1mseg
            nop
            nop
            nop
            decx
            bne vuelvo
            nop
            nop
            nop
            nop
            deca
            bne vuelvo1
            rts

****************************************
*        vectores de interrupcion        *
****************************************

 org vector_receptor

      dw recepcion            ;vector del receptor

 org vectorreset

        dw configuraciones    ; Reset Vector
        end
```

## Software en Matlab

```
function varargout = pruebaguide(varargin)

gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
                   'gui_Singleton',  gui_Singleton, ...
                   'gui_OpeningFcn', @descargador_OpeningFcn, ...
                   'gui_OutputFcn',  @pruebaguide_OutputFcn, ...
                   'gui_LayoutFcn',  [] , ...
                   'gui_Callback',   []);
if nargin & isstr(varargin{1})
    gui_State.gui_Callback = str2func(varargin{1});
end

if nargout
    [varargout{1:nargout}] = gui_mainfcn(gui_State, varargin{:});
else
    gui_mainfcn(gui_State, varargin{:});
end

function descargador_OpeningFcn(hObject, eventdata, handles, varargin)

handles.output = hObject;
global fecha
global temp
fecha=fix(clock);
ano=num2str(fecha([1]));
mes=num2str(fecha([2]));
dia=num2str(fecha([3]));
a='/';
fecha=strcat(dia,a,mes,a,ano);
set(handles.fecha,'string',fecha);

tensionactual=100;
tensionnominal=0;
tensionnominal=get(handles.tensionnom,'string');
tensionnominal=str2double(tensionnominal);
tipo=get(handles.acidalc,'value');
if tipo==1                                  %acidas
     tensionfinal=tensionnominal*1.75/2;

     guidata(hObject, handles);
                                          %alcalinas
else     tensionfinal=tensionnominal*1/1.25;
     guidata(hObject, handles);
end
  set(handles.tensionfinal,'string',tensionfinal)
  capnom=get(handles.capacidadnom,'string');
capnom=str2double(capnom);
sp=get(handles.setpoint,'string');
sp=str2double(sp);
spporc=(sp*100)/capnom;
set(handles.setp,'string',spporc);

axes(handles.ejecorriente);
grid on
axes(handles.ejetension);
grid on

guidata(hObject, handles);
```

```
function varargout = pruebaguide_OutputFcn(hObject, eventdata, handles)
varargout{1} = handles.output;

function comenzar_Callback(hObject, eventdata, handles)
 global fecha
[N,Wn]=buttord(0.02, 0.3, 0.5, 60);
[num,den]=butter(N,Wn);
global corrientefilt
global tensionfilt
global tensionfinal
global tensinactual
global tensionnominal
global T
global tiempo
global cap25
global corrmedia
global bandera
bandera=0;

puerto= serial('com1');
set(puerto,'BaudRate',1200);
set(puerto,'inputbuffersize',2);
set(puerto,'timeout',10);
fopen(puerto);

fwrite(puerto,'02');

    tiem=clock;
    while puerto.bytesavailable==0,
        duracion=etime(clock,tiem);
        if (duracion>2)
            fclose(puerto)
            errordlg('No se establece comunicacion')

            return
        end
    end
    set(handles.comenzar,'Enable','off')
    set(handles.detener,'Enable','on')
T=get(handles.temp,'string');
T=str2double(T);
tensionactual=100;
tensionnominal=0;
tensionnominal=get(handles.tensionnom,'string');
tensionnominal=str2double(tensionnominal);
tipo=get(handles.acidalc,'value');

tensionfinal=get(handles.tensionfinal,'string');
tensionfinal=str2double(tensionfinal);
K=2;
Ti=0.833;

Td=0;
Gsat=0.1;
i=0;

j=0;
antiwindup=0;
sumcorr=0;
sumatoria=0;
```

```
errorant=0;
error=0;
tiem=clock;

contador=0;
while bandera~=1
    duracion=etime(clock,tiem);
     duracion=duracion/60;
     duracionent=fix(duracion);
       i=i+1;

   %-------------------------------------
     [corractual,cantidad]=fread(puerto,1);
   %-------------------------------------

   if corractual==[]
       corractual=corrienteant;
   end

       corrienteant=corractual;
   tensionactual=fread(puerto,1);

     corractual=corractual*2.5/256;
     corriente([i])=corractual;
     %corriente media
     sumcorr=sumcorr+corractual;
     corrmedia=sumcorr/i;
     %capacidad a la temp de trabajo
     capacidadinst=corrmedia*duracionent/60;
     %referir medicion a 25ºC
     if duracionent>=60
         Y=0.006;
     end
     if duracionent<60
         Y=0.01;
     end

     cap25=capacidadinst/(1+Y*(T-25));
     set(handles.resultcapacidad,'string',cap25);
     set(handles.resultcorriente,'string',corrmedia);

     tensionactual=tensionactual*6/256;
     tension([i])=tensionactual;
     corrientefilt=filter(num,den,corriente);
     tensionfilt=filter(num,den,tension);

    if tensionactual<=tensionfinal
        contador=contador+1;
    else
        contador=0;
    end
    if contador>=5
        bandera=1;

    end

     setpoint=get(handles.setpoint,'string');
     setpoint=str2double(setpoint);
     axes(handles.ejecorriente);
     handles.ejex=0:1/120:1/120*j;
```

```
   setp([i])=setpoint;
   ten([i])=tensionfinal;

   plot(handles.ejex,corrientefilt);

   grid on
   hold on
   plot(handles.ejex,corriente,'green');
   hold on
   plot(handles.ejex,setp,'red');

   hold off
   axes(handles.ejetension);

   plot(handles.ejex,tensionfilt);
   grid on
   hold on
   plot(handles.ejex,tension,'green');
   hold on
   plot(handles.ejex,ten,'red');

   hold off
   j=j+1;

   pause(0.0001);
   %controlador
   %pid
   setpoint=get(handles.setpoint,'string');
   setpoint=str2double(setpoint);
   errorant=error;
   error=setpoint-corractual;
   %proporcional
   P=K*error;
   %integral
 sumatoria=error+sumatoria;
 I=(sumatoria/Ti)*K;
 %derivativo


 deltaT=0.5;
 D=((error-errorant)/deltaT)*Td*K;
 %las tres acciones
 control=P+I+D+antiwindup;
 %anti wind-up a 80%
if control>80
    antiwindup=80-control;
else antiwindup=0;
end
    if control<0.05
        antiwindup=0.05-control;
    else antiwindup=0;
    end
 antiwindup=antiwindup*Gsat;
 if control>99
     control=99;
 end

try
 fwrite(puerto,control);
catch
```

```
  end

   tiempo=fix(1/120*j);
   set(handles.resultiempo,'string',tiempo);

   set(handles.edita,'string',tensionfilt([i]));

   set(handles.edita1,'string',corrientefilt([i]));

end
fin=150;
fwrite(puerto,fin);

fclose(puerto);
msgbox('Fin de descarga')
set(handles.detener,'Enable','off')
set(handles.comenzar,'Enable','on')
guidata(hObject, handles);

function marca_Callback(hObject, eventdata, handles)
function tipo_Callback(hObject, eventdata, handles)
function acidalc_Callback(hObject, eventdata, handles)

tensionactual=100;
tensionnominal=0;
tensionnominal=get(handles.tensionnom,'string');
tensionnominal=str2double(tensionnominal);
tipo=get(handles.acidalc,'value');
if tipo==1                                  %acidas
     tensionfinal=tensionnominal*1.75/2;

     guidata(hObject, handles);
                                       %alcalinas
else   tensionfinal=tensionnominal*1/1.25;
     guidata(hObject, handles);
end
valor=get(handles.tfinalauto,'value');
if valor==1
        set(handles.tensionfinal,'string',tensionfinal)
end

guidata(hObject, handles);

function capacidadnom_Callback(hObject, eventdata, handles)
global capnom
global sp
capnom=get(handles.capacidadnom,'string');
capnom=str2double(capnom);
sp=get(handles.setpoint,'string');
sp=str2double(sp);
spporc=(sp*100)/capnom;
set(handles.setp,'string',spporc);
guidata(hObject, handles);

function setpoint_Callback(hObject, eventdata, handles)
global capnom
global sp
capnom=get(handles.capacidadnom,'string');
capnom=str2double(capnom);
sp=get(handles.setpoint,'string');
sp=str2double(sp);
```

```
spporc=(sp*100)/capnom;
set(handles.setp,'string',spporc);
guidata(hObject, handles);

function detener_Callback(hObject, eventdata, handles)
global bandera

opc=questdlg('¿Desea detener la descarga?','DETENER','Si','No','No');
if strcmp(opc,'No')
    bandera=0;
return
end
bandera=1;

function tensionnom_Callback(hObject, eventdata, handles)
tensionactual=100;
tensionnominal=0;
tensionnominal=get(handles.tensionnom,'string');
tensionnominal=str2double(tensionnominal);
tipo=get(handles.acidalc,'value');
if tipo==1                                  %acidas
     tensionfinal=tensionnominal*1.75/2;

     guidata(hObject, handles);
                                           %alcalinas
else     tensionfinal=tensionnominal*1/1.25;
     guidata(hObject, handles);
end
valor=get(handles.tfinalauto,'value');
if valor==1
        set(handles.tensionfinal,'string',tensionfinal)
end

guidata(hObject, handles);

function temp_Callback(hObject, eventdata, handles)
function resultcapacidad_Callback(hObject, eventdata, handles)
function fecha_Callback(hObject, eventdata, handles)
function resultcorriente_Callback(hObject, eventdata, handles)
function resultiempo_Callback(hObject, eventdata, handles)
function archivomenu_Callback(hObject, eventdata, handles)

function abrirmenu_Callback(hObject, eventdata, handles)
global tensionfilt
global corrientefilt
global fi
global fecha
global marca
global tipo
global numero
bandera=0;
tensionfilt=[0];
corrientefilt=[0];
[FileName Path]=uigetfile({'*.txt'},'Abrir documento');
if isequal(FileName,0)
return
else

[fi,texto]=fopen(fullfile(Path,FileName),'r');
completo=strcat('Descargador de baterias','-',fullfile(Path,FileName));
set(handles.pantalla,'name',completo)
```

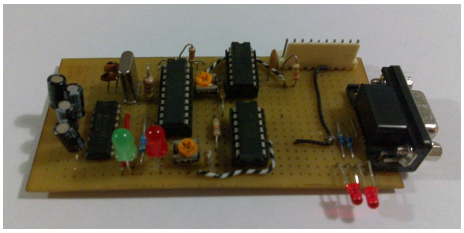
```
end
fecha=fscanf(fi,'%s',1);
marca=fscanf(fi,'%s',1);
tipo=fscanf(fi,'%s',1);
numero=fscanf(fi,'%s',1);
acidalc=fscanf(fi,'%d',1);
tensionnominal=fscanf(fi,'%f',1);
capnom=fscanf(fi,'%f',1);
sp=fscanf(fi,'%f',1);
T=fscanf(fi,'%f',1);
tiempo=fscanf(fi,'%f',1);
cap25=fscanf(fi,'%s',1);
corrmedia=fscanf(fi,'%s',1);

if acidalc==1                                   %acidas
     tensionfinal=tensionnominal*1.75/2;
     set(handles.acidalc,'value',1);
     guidata(hObject, handles);
     %end
 %if tipo==0                                        %alcalinas
else     tensionfinal=tensionnominal*1/1.25;
    set(handles.acidalc,'value',2);
     guidata(hObject, handles);
end
  set(handles.tensionfinal,'string',tensionfinal)

spporc=(sp*100)/capnom;
set(handles.setp,'string',spporc);

losdos=fscanf(fi,'%f');
g=length(losdos);
h=g/2;
for i=1:h

    tensionfilt([i])=losdos([i]);
    i=i+1;
end

for p=1:h
    corrientefilt([p])=losdos([i]);
    p=p+1;
    i=i+1;
end

axes(handles.ejetension);
xmin=0;
largo=length(tensionfilt);
xmax=1/120*(largo-1);
ymax=max(tensionfilt)+0.2;
ymin=0;
ejex=0:1/120:xmax;
%f=length(ejex);

plot(ejex,tensionfilt);
grid on
axes(handles.ejecorriente);

plot(ejex,corrientefilt);
grid on
%axis([xmin xmax ymin ymax]);
```

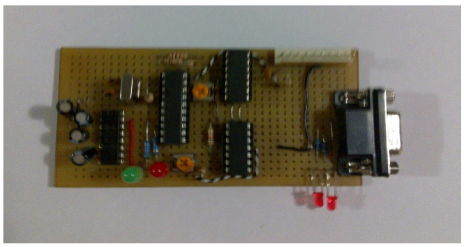
```
set(handles.marca,'string',marca)
set(handles.tipo,'string',tipo)
set(handles.numero,'string',numero)
%set(handles.acidalc,'value',acidalc)
set(handles.capacidadnom,'string',capnom)
set(handles.tensionnom,'string',tensionnominal)
set(handles.setpoint,'string',sp)
set(handles.temp,'string',T)
set(handles.resultiempo,'string',tiempo)
set(handles.resultcapacidad,'string',cap25)
set(handles.resultcorriente,'string',corrmedia)
set(handles.fecha,'string',fecha)

guidata(hObject, handles);

function cerrarmenu_Callback(hObject, eventdata, handles)
global tensionfilt
global corrientefilt
global fecha

global tensionnominal

global T
global tiempo
global cap25
global corrmedia

 marca=get(handles.marca,'string');
 tipo=get(handles.tipo,'string');
 numero=get(handles.numero,'string');
 acidoalc=get(handles.acidalc,'value');
 capnom=get(handles.capacidadnom,'string');
capnom=str2double(capnom);
sp=get(handles.setpoint,'string');
sp=str2double(sp);
[FileName Path]=uiputfile({'*.txt'},'Guardar documento');
if isequal(FileName,0);
return
else

[fi,texto] = fopen(fullfile(Path,[FileName]),'w');
fprintf(fi,'%s\t',fecha);
fprintf(fi,'%s\t',marca);
fprintf(fi,'%s\t',tipo);
fprintf(fi,'%s\t',numero);
fprintf(fi,'%d\t',acidoalc);
fprintf(fi,'%1.5f\t',tensionnominal);
fprintf(fi,'%1.5f\t',capnom);
fprintf(fi,'%1.5f\t',sp);
fprintf(fi,'%1.5f\t',T);
fprintf(fi,'%1.5f\t',tiempo);
fprintf(fi,'%1.5f\t',cap25);
fprintf(fi,'%1.5f\t',corrmedia);

fprintf(fi,'%1.5f\t',tensionfilt);

fprintf(fi,'%1.5f\t',corrientefilt);
fclose(fi);
end

function salirmenu_Callback(hObject, eventdata, handles)
```

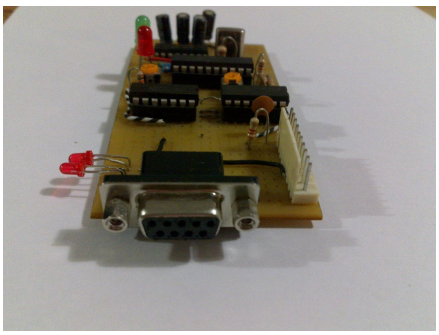
```
opc=questdlg('¿Desea salir del programa?','SALIR','Si','No','No');
if strcmp(opc,'No')
return;
end

clear,clc,close all

function reportemenu_Callback(hObject, eventdata, handles)

function generarmenu_Callback(hObject, eventdata, handles)

function tensionfinal_Callback(hObject, eventdata, handles)

function setp_Callback(hObject, eventdata, handles)

capnom=get(handles.capacidadnom,'string');
capnom=str2double(capnom);
sppor=get(handles.setp,'string');
sppor=str2double(sppor);
sp=sppor*capnom/100;
set(handles.setpoint,'string',sp)
guidata(hObject, handles);

function tensionactual_Callback(hObject, eventdata, handles)
function corrienteactual_Callback(hObject, eventdata, handles)
function numero_Callback(hObject, eventdata, handles)
function cerrar_Callback(hObject, eventdata, handles)

global fi
global fecha
set(handles.tipo,'string','');
set(handles.marca,'string','');
set(handles.numero,'string','');
set(handles.capacidadnom,'string','');
set(handles.tensionnom,'string','');
set(handles.tensionfinal,'string','');
set(handles.setpoint,'string','');
set(handles.temp,'string','');
set(handles.edita,'string','');
set(handles.edita1,'string','');
set(handles.resultiempo,'string','');
set(handles.resultcapacidad,'string','');
set(handles.resultcorriente,'string','');
set(handles.setp,'string','');
axes(handles.ejecorriente);

plot(0);
axis([0 1 0 1]);

axes(handles.ejetension);

plot(0);
axis([0 1 0 1])
completo='Descargador de baterias';
set(handles.pantalla,'name',completo);

fecha=fix(clock);
ano=num2str(fecha([1]));
mes=num2str(fecha([2]));
dia=num2str(fecha([3]));
a='/';
```

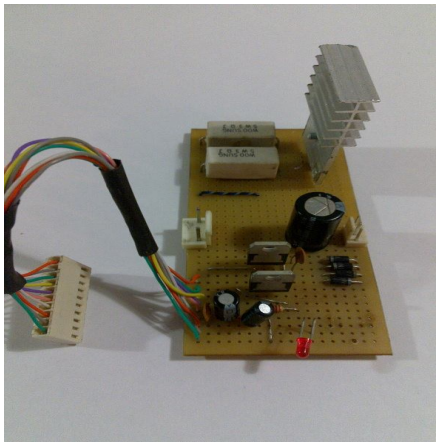
```
fecha=strcat(dia,a,mes,a,ano);
set(handles.fecha,'string',fecha);

fclose(fi);

% --- Executes on button press in tfinalmanual.
function tfinalmanual_Callback(hObject, eventdata, handles)
valor=get(handles.tfinalmanual,'value');
if valor==0
        set(handles.tfinalauto,'value',1);
       tensionactual=100;
tensionnominal=0;
tensionnominal=get(handles.tensionnom,'string');
tensionnominal=str2double(tensionnominal);
tipo=get(handles.acidalc,'value');
 if tipo==1                                  %acidas
     tensionfinal=tensionnominal*1.75/2;

     guidata(hObject, handles);
                                           %alcalinas
 else     tensionfinal=tensionnominal*1/1.25;
     guidata(hObject, handles);
 end
  set(handles.tensionfinal,'string',tensionfinal)
  set(handles.tensionfinal,'enable','off');

end
if valor==1
    set(handles.tfinalauto,'value',0);
set(handles.tensionfinal,'enable','on');
set(handles.tensionfinal,'string','');
end

function tfinalauto_Callback(hObject, eventdata, handles)
valor=get(handles.tfinalauto,'value');
if valor==0
        set(handles.tfinalmanual,'value',1);
        set(handles.tensionfinal,'string','');
end
if valor==1
    set(handles.tfinalmanual,'value',0);

    tensionactual=100;
tensionnominal=0;
tensionnominal=get(handles.tensionnom,'string');
tensionnominal=str2double(tensionnominal);
tipo=get(handles.acidalc,'value');
 if tipo==1                                  %acidas
     tensionfinal=tensionnominal*1.75/2;

     guidata(hObject, handles);
                                           %alcalinas
 else     tensionfinal=tensionnominal*1/1.25;
     guidata(hObject, handles);
 end
  set(handles.tensionfinal,'string',tensionfinal)
  set(handles.tensionfinal,'enable','off');
end

guidata(hObject, handles);
```

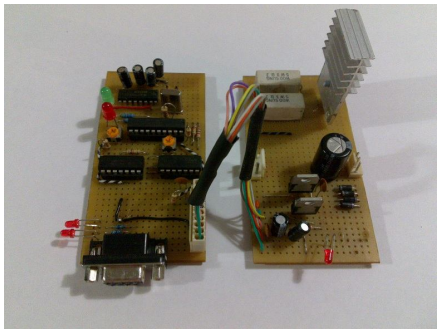
## *FOTOS DEL PROTOTIPO*





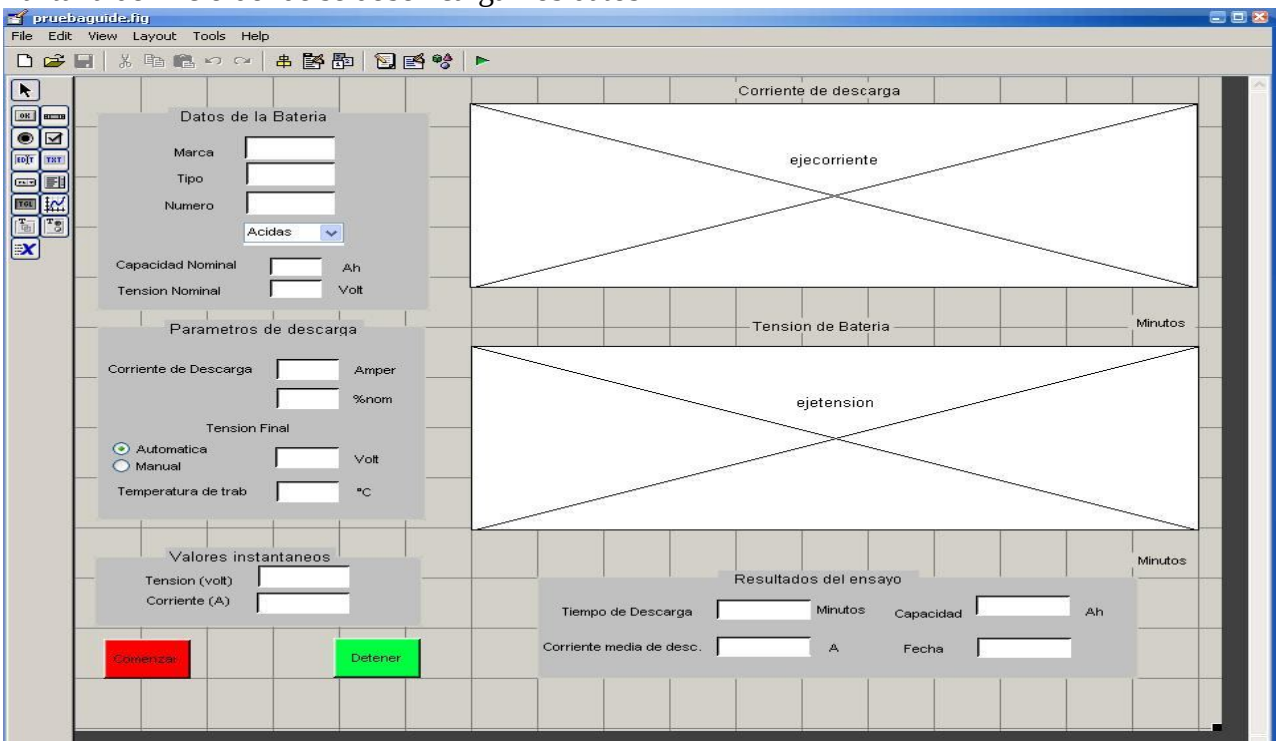






## *FOTOS DE PANTALLA*

Pantalla de inicio donde se deben cargar los datos



## *INFORMACIÓN ADICIONAL*

### Puerto serie

Un puerto serie o puerto serial es una interfaz de comunicaciones de datos digitales, frecuentemente utilizado por computadoras y periféricos, donde la información es transmitida bit a bit enviando un solo bit a la vez, en contraste con el puerto paralelo que envía varios bits simultáneamente.
RS-232 (Recommended Standard 232, también conocido como Electronic Industries Alliance RS-232C) es una interfaz que designa una norma para el intercambio serie de datos binarios entre un DTE (Equipo terminal de datos) y un DCE (*Data Communication Equipment*, Equipo de Comunicación de datos)
Construcción física

La interfaz RS-232 está diseñada para distancias cortas, de hasta 15 metros según la norma , y para velocidades de comunicación bajas, de no más de 20 Kilobits/segundo. A pesar de ello, muchas veces se utiliza a mayores velocidades con un resultado aceptable. La interfaz puede trabajar en comunicación asíncrona o síncrona y tipos de canal simplex, half duplex o full duplex. En un canal simplex los datos siempre viajarán en una dirección, por ejemplo desde DCE a DTE. En un canal half duplex, los datos pueden viajar en una u otra dirección, pero sólo durante un determinado periodo de tiempo; luego la línea debe ser conmutada antes que los datos puedan viajar en la otra dirección. En un canal full duplex, los datos pueden viajar en ambos sentidos simultáneamente. Las líneas de *handshaking* de la RS-232 se usan para resolver los problemas asociados con este modo de operación, tal como en qué dirección los datos deben viajar en un instante determinado.

Si un dispositivo de los que están conectados a una interfaz RS-232 procesa los datos a una velocidad menor de la que los recibe deben de conectarse las líneas *handshaking* que permiten realizar un control de flujo tal que al dispositivo más lento le de tiempo de procesar la información. Las líneas de "*hand shaking*" que permiten hacer este control de flujo son las líneas RTS y CTS. Los diseñadores del estándar no concibieron estas líneas para que funcionen de este modo, pero dada su utilidad en cada interfaz posterior se incluye este modo de uso

Los circuitos y sus definiciones

Las UART o U(S)ART (Transmisor y Receptor Síncrono Asíncrono Universal) se diseñaron para convertir las señales que maneja la CPU y transmitirlas al exterior. Las UART deben resolver problemas tales como la conversión de voltajes internos del DCE con respecto al DTE, gobernar las señales de control, y realizar la transformación desde el bus de datos de señales en paralelo a serie y viceversa. Debe ser robusta y deberá tolerar circuitos abiertos, cortocircuitos y escritura simultánea sobre un mismo pin, entre otras consideraciones. Es en la UART en donde se implementa la interfaz.

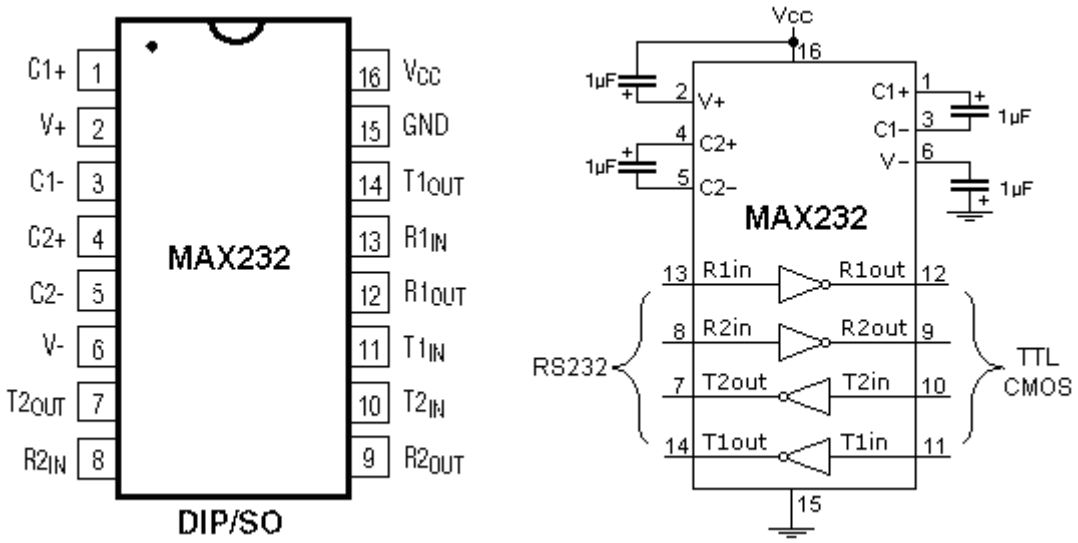
Generalmente cuando se requiere conectar un microcontrolador (con señales típicamente entre 3.3 y 5 V) con un puerto RS-232 estándar se utiliza un driver de línea, típicamente un MAX232 o compatible, el cual mediante dobladores de voltaje positivos y negativos permite obtener la señal bipolar (típicamente alrededor de +/- 6V) requerida por el estándar.

Las convenciones que se usan son las siguientes:

| Voltaje | Señal | Nivel Lógico | Control |
|---|---|---|---|
| *+3 a +15* | Espacio | 0 | On |
| *-3 a –15* | Marca | 1 | Off |

Los valores de tensión se invierten desde los valores lógicos. Por ejemplo, el valor lógico más positivo corresponde al voltaje más negativo. También un 0 lógico corresponde a la señal de valor verdadero o activada. Por ejemplo si la línea DTR está al valor 0 lógico, se encuentra en la gama de voltaje que va desde +3 a +15 V, entonces DTR está listo (*ready*).
MAX232 - Conversor TTL-RS232

El **MAX232** es un circuito integrado que convierte los niveles de las líneas de un puerto serie RS232 a niveles TTL y viceversa. Lo interesante es que sólo necesita una alimentación de 5V, ya que genera internamente algunas tensiones que son necesarias para el estándar RS232. Otros integrados que manejan las líneas RS232 requieren dos voltajes, +12V y -12V.




El MAX232 soluciona la conexión necesaria para lograr comunicación entre el puerto serie de una PC y cualquier otro circuito con funcionamiento en base a señales de nivel TTL/CMOS.

El circuito integrado posee dos conversores de nivel TTL a RS232 y otros dos que, a la inversa, convierten de RS232 a TTL.

Estos conversores son suficientes para manejar las cuatro señales más utilizadas del puerto serie del PC, que son TX, RX, RTS y CTS.

TX es la señal de transmisión de datos, RX es la de recepción, y RTS y CTS se utilizan para establecer el protocolo para el envío y recepción de los datos.

http://pdfserv.maxim-ic.com/en/ds/MAX220-MAX249.pdf