PROTOCOLOS EN S.E. PRACTICA 1 – R. OLIVA /CESE 2018

VERSION 04-07-2018

1.1 ENUNCIADO

TRABAJO PRACTICO 1 / PRSE - CESE2018

Utilizando los ejemplos provistos se desea guardar un log de datos de mediciones tomadas por el ADC, en un único archivo. Además incluir en cada muestra un time-stamp de cuando la misma fue obtenida utilizando el periférico RTC. El archivo obtenido deberá contener líneas con el siguiente formato:

CH1;CH2;CH3;YYYY/MM/DD_hh:mm:ss;

Por ejemplo:

Archivo “Muestras.txt”

155;1;24;2018/07/10_20:05:36;

154;0;425;2018/07/10_20:05:37;

1.2 RESULTADOS

Al Miercoles 4-7-2018: Creado TP1_SD (lee ADC, RTC y envía a archivo SD las lecturas + Timestamp). La solución utiliza los elementos ya vistos en la Clase I y los elementos de la sAPI. Incorporamos el uso de la función stdioSprintf() para escribir strings con formato específico.

I) Se agrega lo siguiente como función auxiliar utilizando stdioSprintf(), que soporta el formato requerido por el práctico para

```
/*
* printDateAndTime()
* Devolver fecha y hora en formato "YYYY/DD/MM_HH:MM:SS"
*/
void printDateAndTime( rtc_t * rtc, char *sbuf ){

   stdioSprintf(sbuf, "%04d/%02d/%02d_%02d:%02d:%02d",
                (int) (rtc->year),
                (int) (rtc->month),
                (int) (rtc->mday),
                (int) (rtc->hour),
                (int) (rtc->min),
                (int) (rtc->sec)
   );

}
```

II) CONFIGURACION: El programa combina los ejemplos de RTC y SD, además del de lectura del ADC. Se han agregado comentarios a los puntos relevantes:

```
// ---------- CONFIGURACIONES ------------------------------
   // Inicializar y configurar la plataforma
   boardConfig();

   // (a1) SPI configuration
   spiConfig( SPI0 );

   // (a2) Inicializar el conteo de Ticks con resolucion de 10ms,
   // con tickHook diskTickHook
   tickConfig( 10 );
```

```
tickCallbackSet( diskTickHook, NULL );

/* Inicializar UART_USB a 115200 baudios */
uartConfig( UART_USB, 115200 );

/* (b)Inicializar ADC  */
adcConfig( ADC_ENABLE ); /* ADC */

uartWriteString( UART_USB, "\r\n TP1_SD Ensayo 1");

rtc.year = 2018;
rtc.month = 7;
rtc.mday = 3;
rtc.wday = 1;
rtc.hour = 13;
rtc.min = 17;
rtc.sec= 0;

/* Inicializar RTC */
uartWriteString( UART_USB, "\r\n Inicializo RTC..");
/* (c) Inicializacion fija RTC
val = rtcConfig( &rtc );

delayConfig( &delay1s, 1000 );

delay(2000); // El RTC tarda en setear la hora, por eso el delay

for( i=0; i<10; i++ ){
   /* Leer fecha y hora */
   val = rtcRead( &rtc );
   /* Mostrar fecha y hora en formato "DD/MM/YYYY, HH:MM:SS" */
   showDateAndTime( &rtc );
   delay(1000);
}

/* (d) Leo la Entrada Analogica AI0 - ADC0 CH1 */
uartWriteString( UART_USB, "\n\r ADC Canales ADCBuf:\r\n");
muestra1 = adcRead( CH1 );
muestra2 = adcRead( CH2 );
muestra3 = adcRead( CH3 );
stdioSprintf(ADCbuf, "%03d;%03d;%03d;", muestra1,muestra2,muestra3);
stdioPrintf(UART_USB, "%s \n\r", ADCbuf);

// ------ PROGRAMA QUE ESCRIBE EN LA SD -------

// (e) Give a work area to the default drive
uartWriteString( UART_USB, "\r\n SD_fmount() ..");
if( f_mount( &fs, "", 0 ) != FR_OK ){
   // If this fails, it means that the function could
   // not register a file system object.
         uartWriteString( UART_USB, "\r\n Error en SD_fmount..");
   // Check whether the SD card is correctly connected
} else {
             uartWriteString( UART_USB, "\r\n SD_fmount OK!.");
}
```

Comentarios del código:

(a1), (a2) – Inicialización del Puerto SPI, y enlace a la rutina de temporización disk_timerproc() requerida por la FAT-FFS de ELM/Chan. Esto permite funcionar correctamente en el acceso a las tarjetas SD con ese sistema de archivos.
NOTA: Lo que puede agregarse (pero no estoy seguro cómo para la CIAA), para que el FFS incorpore timestamps en el trabajo de modificación y creación de archivos, es un enlace a las rutinas de acceso RTC (se requieren rutinas separadas de hora y fecha). En el caso de una placa CL2 de 8 bits, esto lo realizamos de la siguiente manera:

```
// 15.01.2018 Pointers to rtc functions (now defined in TI-CL2_12a.h,.c) unchanged..
/* init the pointer to the RTC function used for reading time */
prtc_get_time=(void (*)(unsigned char *,unsigned char *,unsigned char *)) rtc_get_time;
/* init the pointer to the RTC function used for reading date */
prtc_get_date=(void (*)(unsigned char *,unsigned char *,unsigned int *)) rtc_get_date;
```

(b), – Inicialización solamente del ADC vía sAPI
(c), – Asignación de valores al RTC e inicialización. Habría que buscar una forma a través del análisis del valor de retorno val, para ver si el RTC opera correctamente / batería correctamente instalada, y que la inicialización sea una única vez. En el caso del CL2 una de las versiones permite resetear RTC con una tecla presionada, y después reingresar valores de usuario vía otra rutina:

```
//  RTC Read..
RTC_result = rtc_get_timeNdate(&RTCHour, &RTCMin, &RTCSec, &RTCDay, &RTCMonth, &RTCYear);
// clock battery dead.. reset to something sane
// Set to error 16.2.09
// Back to TWI_SUCCESS 17.2.09
// Force initialize if KBD_LEFT_ARROW pressed.. 4.1.2012
// RTC_result = 187;
if((RTC_result != TWI_SUCCESS)|(KBD_LEFT_ARROW == 0))
     {
     printf("RTC-Now initialized!\n\r");
     set_LCD_cur(0,0);
     disp_cstr("RTes RTC Init.. ");
         // Probable requirement of Variables, not constants in rtc_set_time/date..
         // Verify 19.11.2017
                 RTCHour = 22;
         RTCMin  = 41;
         RTCSec  = 0;
         rtc_set_time(RTCHour,RTCMin,RTCSec);
         delay_ms(1000);
         RTCDay  = 19;
         RTCMonth = 11;
         RTCYear  = 2017;
         rtc_set_date(RTCDay,RTCMonth,RTCYear);
                 delay_ms(1000);
     }
 else
    {
    printf("RTC-OK!\n\r");
    set_LCD_cur(0,0);
    disp_cstr("RTC OK read..   ");
 }
```

(d), Lee entradas analógicas como enteros, imprime a un string y luego ese string al puerto serie.
(e), Via la función f_mount() de ELM-Chan, prepara la unidad lógica que se asigna a los archivos de la SD.

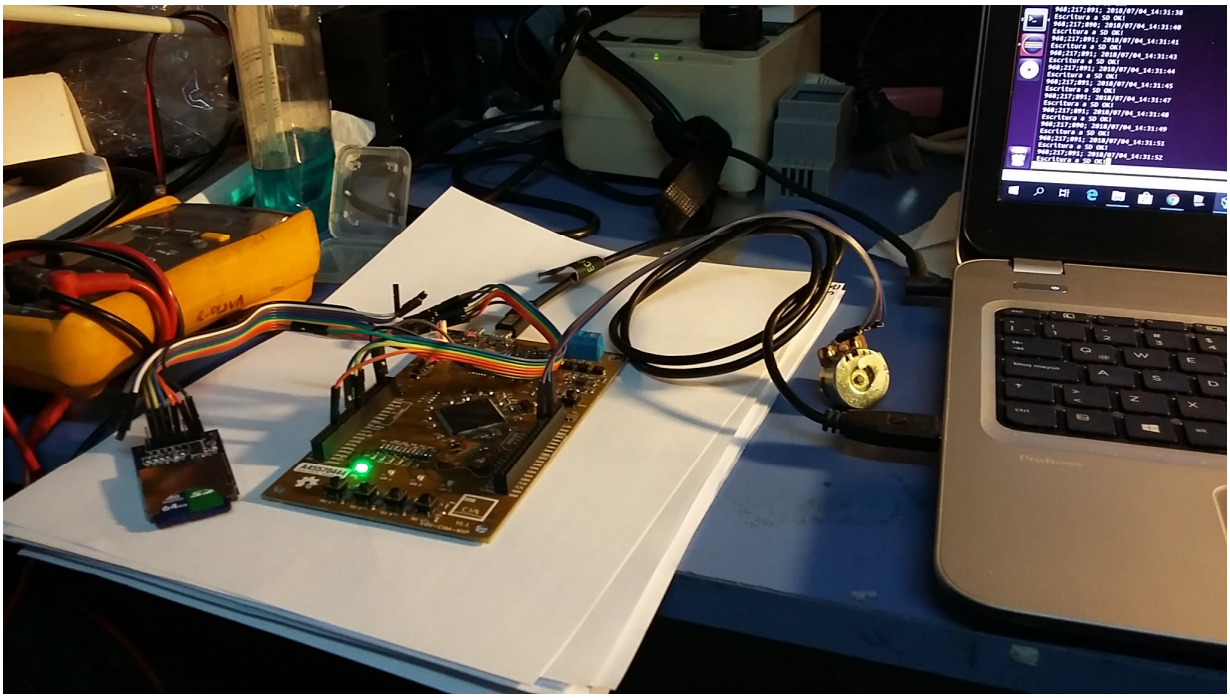
III) LAZO PRINCIPAL: El programa utiliza elementos de los ejemplos de RTC, SD y ADC. Se ha agregado el acceso a SD a un archivo de nombre fijo FILENAME = muetras.txt . Se utiliza un delay no bloqueante para cada 1 segundo llamar a la secuencia: - Lectura RTC a un string RTCbuf, lectura de ADC (3 canales) e impresión a un string ADCbuf, apertura de muestras.txt como WR y APPEND de muestras.txt, escritura de los strings obtenidos con f_write() [Se requiere revisar Nº bytes que escribe..], cierre de archivo y verificación de bytes escritos. LED verde si todo ok, LED Rojo si error.

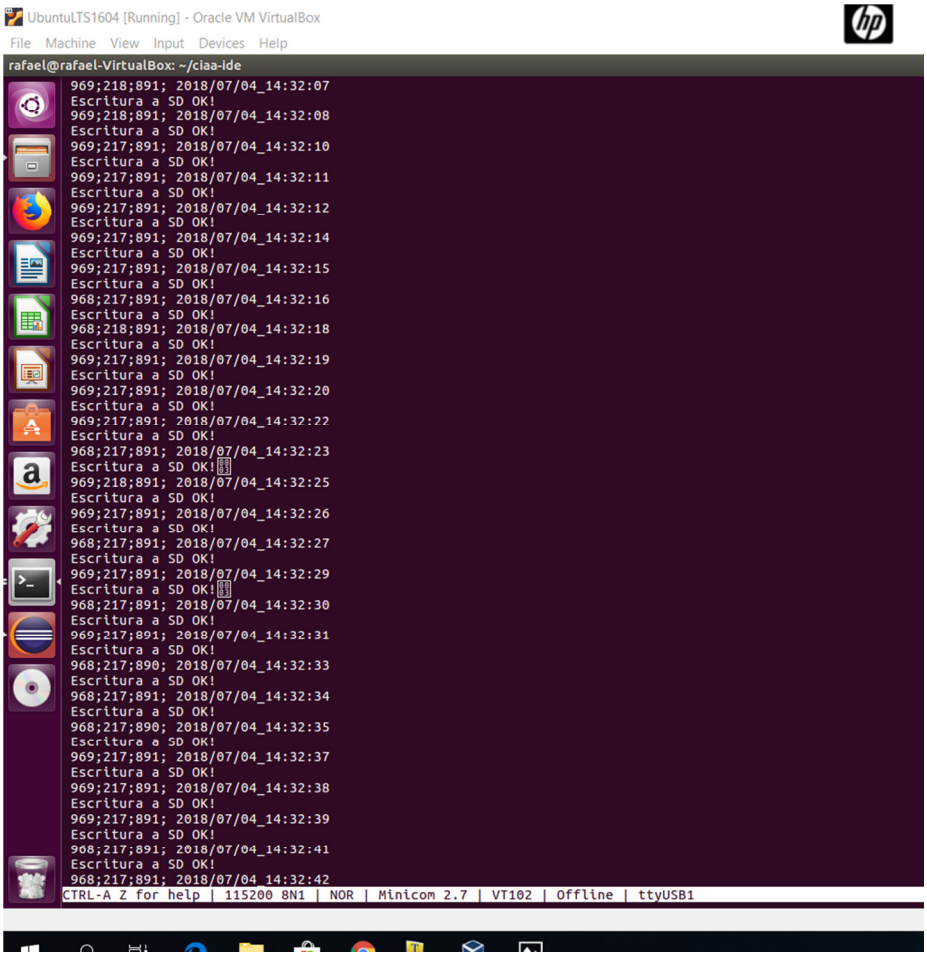
```
//Infinite loop

uartWriteString( UART_USB, "\n\r ADC Ch1; Ch2; Ch3; YYYY/DD/MM_HH:MM:SS\r\n");
while(TRUE) {

      /* delayRead retorna TRUE cuando se cumple el tiempo de retardo */
      if ( delayRead( &delay1s ) ){

        val = rtcRead( &rtc );
        printDateAndTime( &rtc , RTCbuf );
        muestra1 = adcRead( CH1 );
        muestra2 = adcRead( CH2 );
        muestra3 = adcRead( CH3 );
        stdioSprintf(ADCbuf, "%03d;%03d;%03d;", muestra1,muestra2,muestra3);
        stdioPrintf(UART_USB, "\n\r %s ", ADCbuf);
        stdioPrintf(UART_USB, "%s", RTCbuf);

                      if( f_open( &fp, FILENAME, FA_WRITE | FA_OPEN_APPEND ) == FR_OK ){
                         f_write( &fp, ADCbuf, sizeof(ADCbuf), &nbytes );
                         f_write( &fp, RTCbuf, sizeof(RTCbuf), &nbytes );
                         f_write( &fp, "\n\r", 3, &nbytes);
                         f_close(&fp);

                         if( nbytes >= 3 ){
                                // Turn ON LEDG if the write operation was successful
                                uartWriteString( UART_USB, "\r\n Escritura a SD OK!");
                                gpioWrite( LEDG, ON );
                             }
                       } else{
                         // Turn ON LEDR if the write operation was fail
                         gpioWrite( LEDR, ON );
```

```
                                        uartWriteString( UART_USB, "\r\n Error en SD Write!");
                                    }
                }  // End if fopen()
        } // End if delay...
//}       // End while(1)
```

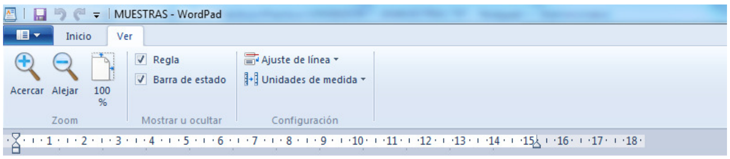
1.3 FOTO Y SALIDAS

1.3.1 Ensayo 4-7-2018: Enviado zip TP1_SDb, salidas en Minicom registradas en pantalla. Archivo muestras.txt



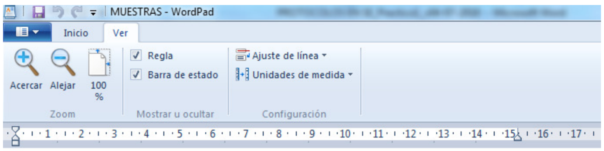


1.3.2 Muestras.txt – En Wordpad se puede ver: (potenciómetro en canal 2, valor 609)



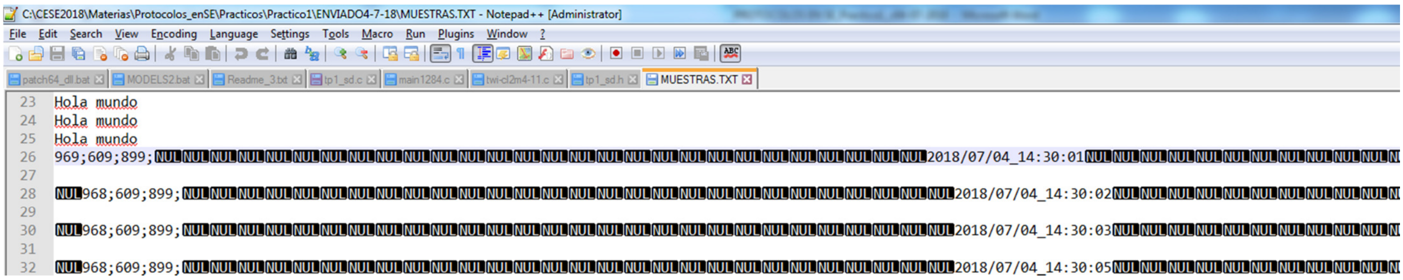

```
Hola mundo
Hola mundo
Hola mundo
Hola mundo
Hola mundo
Hola mundo
969;609;899;                    2018/07/04_14:30:01

 968;609;899;                    2018/07/04_14:30:02

 968;609;899;                    2018/07/04_14:30:03

 968;609;899;                    2018/07/04_14:30:05

 969;609;899;                    2018/07/04_14:30:06

 969;609;899;                    2018/07/04_14:30:07

 968;609;899;                    2018/07/04_14:30:08

 969;609;900;                    2018/07/04_14:30:10

 969;609;900;                    2018/07/04_14:30:11

 968;609;899;                    2018/07/04_14:30:12

 968;609;900;                    2018/07/04_14:30:14

 968;609;900;                    2018/07/04_14:30:15

 968;609;899;                    2018/07/04_14:30:16

 968;609;899;                    2018/07/04_14:30:17
```

1.3.3 Potenciómetro en canal 2, valor 217 (AL FINAL del archivo).




```
 969;217;891;                    2018/07/04_14:32:19

 969;217;891;                    2018/07/04_14:32:20

 969;217;891;                    2018/07/04_14:32:22

 968;217;891;                    2018/07/04_14:32:23

 969;218;891;                    2018/07/04_14:32:25

 969;217;891;                    2018/07/04_14:32:26

 968;217;891;                    2018/07/04_14:32:27

 969;217;891;                    2018/07/04_14:32:29

 968;217;891;                    2018/07/04_14:32:30

 969;217;891;                    2018/07/04_14:32:31

 968;217;890;                    2018/07/04_14:32:33

 968;217;891;                    2018/07/04_14:32:34

 968;217;890;                    2018/07/04_14:32:35

 969;217;891;                    2018/07/04_14:32:37

 969;217;891;                    2018/07/04_14:32:38

 969;217;891;                    2018/07/04_14:32:39

 968;217;891;                    2018/07/04_14:32:41
```

NOTA: Observando con NP++, se ven los caracteres extra que agrega f_write():




```
Hola mundo
Hola mundo
Hola mundo
969;609;899;NULNULNULNULNULNULNULNULNULNULNULNULNULNULNULNULNULNULNULNULNULNULNULNULNULNULNULNUL2018/07/04_14:30:01NULNULNULNULNULNULNULNULNULNULNULN

NUL968;609;899;NULNULNULNULNULNULNULNULNULNULNULNULNULNULNULNULNULNULNULNULNULNULNULNULNULNULNULNUL2018/07/04_14:30:02NULNULNULNULNULNULNULNULNULNULN

NUL968;609;899;NULNULNULNULNULNULNULNULNULNULNULNULNULNULNULNULNULNULNULNULNULNULNULNULNULNULNULNUL2018/07/04_14:30:03NULNULNULNULNULNULNULNULNULNULN

NUL968;609;899;NULNULNULNULNULNULNULNULNULNULNULNULNULNULNULNULNULNULNULNULNULNULNULNULNULNULNULNUL2018/07/04_14:30:05NULNULNULNULNULNULNULNULNULNULN
```