Simulador de desfibrilador automático implantable utilizando un microcontrolador MSP430g2553 y Matlab.

Juan José Guzmán Cruz, Aldo René Salcedo García, Brenda Karen Tapia Pineda
Departamento de Ingeniería Biomédica, Universidad Autónoma Metropolitana, Iztapalapa, México

Resumen— ***Se realizó un sistema retroalimentado para detectar la onda R de un ECG que generara un pulso de reactivación. La señal de ECG se proporcionó mediante Matlab a partir del puerto auxiliar para audífonos, se utilizó un microcontrolador MSP430g2553 de Texas Instrument para detectar la onda R y generar un pulso cuando esta onda no estuviese presente, este pulso se capturo con Matlab por el auxiliar para micrófono y sirvió de condición para reenviar la señal de ECG.*** *Palabras clave*— ECG, Matlab, MSP430.

I. Introducción

El siguiente trabajo tiene la finalidad de describir las actividades realizadas en este proyecto, en el cual se tomó como punto de acción proponer una solución para el tratamiento del paro cardiaco a partir del uso de un desfibrilador interno con función automática.

¿Qué es un desfibrilador?

Un desfibrilador es un dispositivo que administra una descarga eléctrica al corazón a través de la pared torácica. Sus sensores integrados analizan el ritmo cardiaco del paciente durante unos 10 segundos, detectan el estado del paciente e indica si es necesario suministrar una descarga eléctrica. Después de producirse el shock, el desfibrilador vuelve a analizar al paciente y aconsejará una nueva descarga en el caso de ser necesaria. [3]

Desfibrilador interno

La técnica que utiliza el desfibrilador automático implantable (DAI), es que la energía se administra desde el endocardio, lo cual precisa mucha menor cantidad de energía ya que el generador está implantado en el tejido subcutáneo con cables-electrodos generalmente situados en las cavidades cardíacas derechas. Se colocan en pacientes con especial riesgo de presentar una fibrilación ventricular. [4]

II. Metodología

1) Planteamiento del problema e implementación. Para resolver el proyecto planteado, lo primero fue determinar lo que íbamos a hacer (como se muestra en la figura 1), posteriormente, tuvimos que separar las actividades a realizar para poder agilizarlas, sin embargo, se comentaban en equipo los problemas que iban surgiendo y se realizaba retroalimentación de ideas para poder darles solución a dichos problemas.

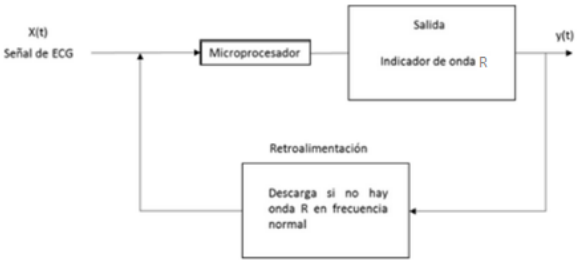

Figura 1.Diagrama a bloques del proyecto a realizar.

Empezamos por generar un plan de trabajo como se observa en la figura siguiente:

		Nombre	Duración	Inicio	Terminado
1		**Aldo Salcedo**	6 days	**10/03/14 08:00 AM**	**17/03/14 05:00 PM**
2		Muestreo continuo de la señal usando un ciclo for	1 day	10/03/14 08:00 AM	10/03/14 05:00 PM
3		Comprobar valores almacenados en el ADC10MEM	1 day	11/03/14 08:00 AM	11/03/14 05:00 PM
4		Relación de ADC10MEM con el voltaje observado en el oscilloscopio	1 day	12/03/14 08:00 AM	12/03/14 05:00 PM
5		Obtener archivos de señales de ECG con patologías	1 day	13/03/14 08:00 AM	13/03/14 05:00 PM
6		Utilizar los archivos de patologías	2 days	14/03/14 08:00 AM	17/03/14 05:00 PM
7		**Brenda Tapia**	6 days	**10/03/14 08:00 AM**	**17/03/14 05:00 PM**
8		Hacer diseño de circuito de conexiones	2 days	10/03/14 08:00 AM	11/03/14 05:00 PM
9		Adquisición de componentes electrónicos	1 day	12/03/14 08:00 AM	12/03/14 05:00 PM
10		Analizar las señales de los archivos y determinar la señal de ECG	2 days	13/03/14 08:00 AM	14/03/14 05:00 PM
11		Comprobar amplitudes y frecuencias	1 day	17/03/14 08:00 AM	17/03/14 05:00 PM
12		**Juan José Guzman**	6 days	**10/03/14 08:00 AM**	**17/03/14 05:00 PM**
13		Usar matlab para reproducir un archivo que contenga los datos de ECG	1 day	10/03/14 08:00 AM	10/03/14 05:00 PM
14		Revisar con el oscilloscopio las características de la señal	1 day	12/03/14 08:00 AM	12/03/14 05:00 PM
15		Obtener la señal externa	1 day	13/03/14 08:00 AM	13/03/14 05:00 PM
16		Segunda etapa de programación: detección de arritmias para emitir banderas	1 day	14/03/14 08:00 AM	14/03/14 05:00 PM
17		Enviar pulso cuando hay bandera y seguir monitoreando	1 day	17/03/14 08:00 AM	17/03/14 05:00 PM
18		**Todos**	7 days	**18/03/14 08:00 AM**	**26/03/14 05:00 PM**
19		Acoplamiento y correcciones	2 days	18/03/14 08:00 AM	19/03/14 05:00 PM
20		Redacción del reporte	2 days	20/03/14 08:00 AM	21/03/14 05:00 PM
21		Presentación	1 day	24/03/14 08:00 AM	24/03/14 05:00 PM
22		Entrega de proyecto terminal	2 days	25/03/14 08:00 AM	26/03/14 05:00 PM

Figura 2. Calendario de plan de trabajo.

Utilizamos un diagrama de Gantt para hacer más eficientes los tiempos de problemas a resolver, como se observa a continuación:

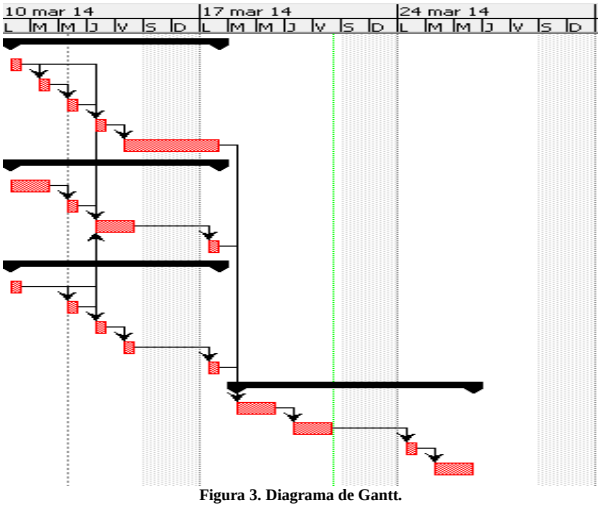

Figura 3. Diagrama de Gantt.

Utilizando el programa Matlab, reprodujimos una señal de ECG sin patologías, analógica, con una frecuencia de 1Hz aproximadamente, y 2 volts de amplitud. Esta señal salía por el jack para audífonos de la computadora, usamos un cable auxiliar macho-macho para poder leer la señal y conectarla al pin P1.4 (el cual fue asignado como entrada) por medio del caimán rojo y el caimán verde lo conectamos a la tierra de la tarjeta MSP430.

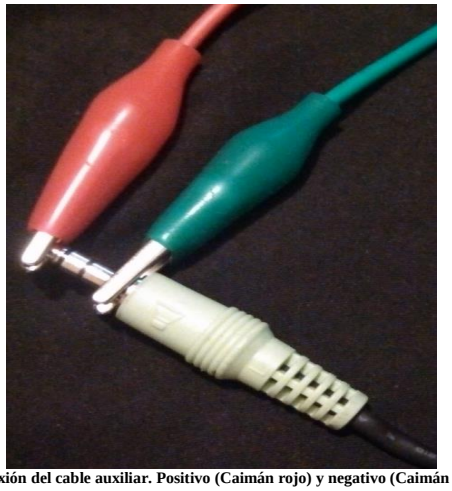

Figura 4. Conexión del cable auxiliar. Positivo (Caimán rojo) y negativo (Caimán verde).

Una vez obtenida la señal de ECG, utilizamos la tarjeta MSP430 para muestrear y detectar los puntos más altos del ECG, que corresponden a la onda R del complejo QRS. El valor máximo de la onda R supera la capacidad del ADC10 para la representación de voltajes y dado que ninguna otra onda está cercana a ese valor, decidimos fijar el umbral para detectar la onda R a 1022. Cuando una señal alcanza este valor se enciende el led verde y después de 100 µs el led se apaga.

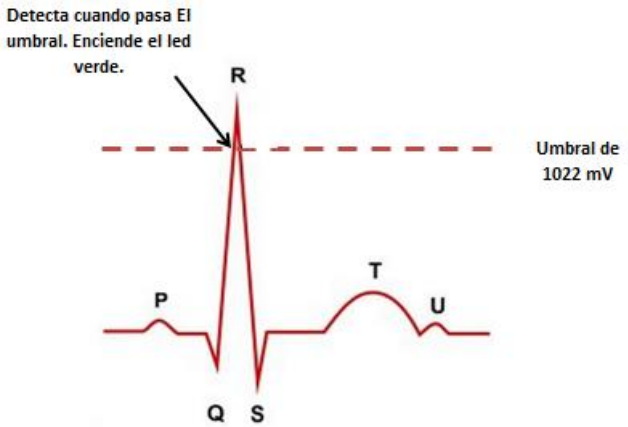

Figura 5. Detección de umbral y encendido de led verde.

La etapa de retroalimentación sucede cuando pasan tres segundos sin detectar la presencia de una onda R. Cuando esto ocurre se enciende el led rojo (P1.0) y este pulso es enviado a Matlab para que reinicie el envío de la señal de ECG, lo cual interpretamos como el impulso eléctrico para la reanimación del músculo cardiaco.

El pulso generado al momento de prender el led rojo lo tomamos e ingresamos por la entrada de micrófono de la computadora, que mediante un código en matlab (función wavrecord *ver anexo*), lo detecta y reinicia el programa en Matlab.

III. RESULTADOS

La señal de ECG que se obtiene a partir de Matlab, es analógica. Usando un cable auxiliar y un osciloscopio pudimos apreciar la forma de la onda que Matlab nos entrega, la cual se muestra en la siguiente figura:

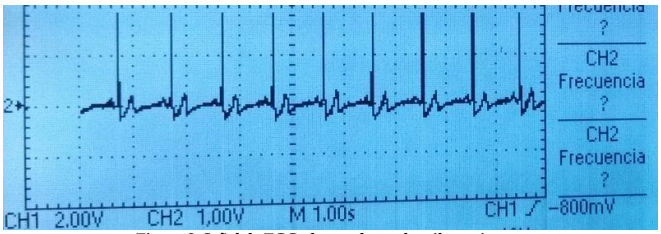

Figura 6. Señal de ECG observada en el osciloscopio.

Al muestrear la señal de manera continua mediante un ciclo *while* infinito, en cada umbral detectado de la señal analógica se emite un pulso, éste también se puede observar en el led verde de la MSP430, o incluso escuchar con una bocina si se conecta al puerto P1.6.

La siguiente imagen muestra que el código implementado funciona de la forma requerida:

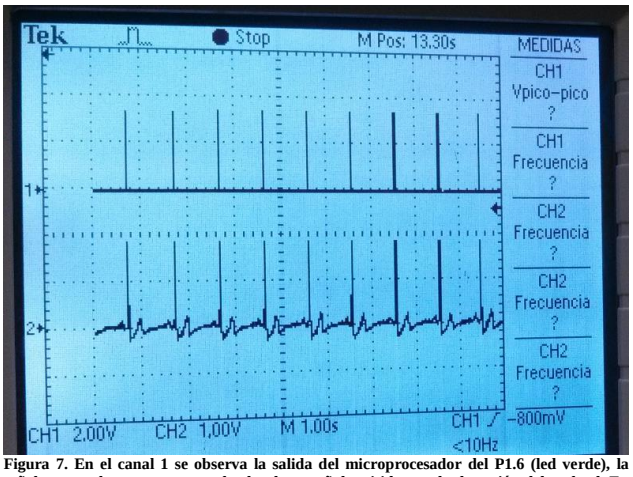

Figura 7. En el canal 1 se observa la salida del microprocesador del P1.6 (led verde), la señal que se observa corresponde al pulso o señal emitida tras la detección del umbral. En el canal 2 se observa la señal de ECG enviada desde Matlab, al MSP430 en el P1.4.

La señal que observamos a la salida del P1.6 del microprocesador, es una señal con una amplitud de aproximadamente 2.6 V y frecuencia de 1Hz.

En la figura siguiente se observa en el canal 1 la señal del led verde (detección de onda R), y en el canal 2 la señal del led rojo (onda R no detectada después de 2 segundos). En la pantalla de osciloscopio se observa un tiempo de 2.56 segundos para generar el pulso de retroalimentación, muy cercano a los 3 segundos de no detección de pulsos que pretendíamos obtener.

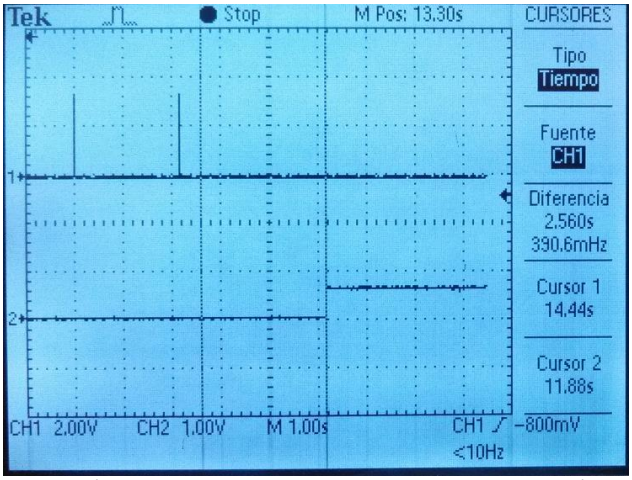

Figura 8. Señal del ECG observada en el osciloscopio. Canal 1, salida de la señal del microprocesador para el led verde (P1.6). Canal 2, salida de la señal del microprocesador para el led rojo (P1.0).

La siguiente figura nos muestra la señal graficada en Matlab para detectar en qué momento se da el pulso de retroalimentación (led rojo). La figura 9b muestra un momento en el que supera el valor de 0.9 y solamente ocurre cuando el led rojo genera ese pulso, por lo tanto tomamos dicho valor como nuestro parámetro para la retroalimentación. Si no se supera ese parámetro, no se genera la retroalimentación, figura 9a.

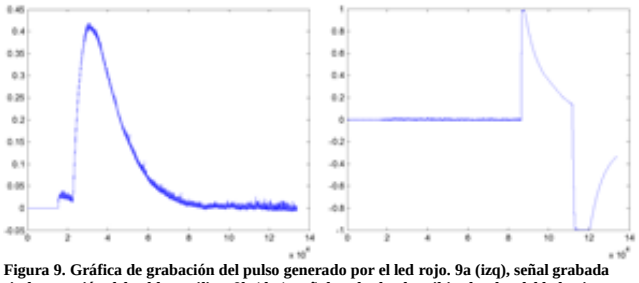

Figura 9. Gráfica de grabación del pulso generado por el led rojo. 9a (izq), señal grabada sin la conexión del cable auxiliar. 9b (der), señal grabada al recibir el pulso del led rojo.

El proyecto resuelve los problemas planteados. En la siguiente figura se observa el material utilizado:

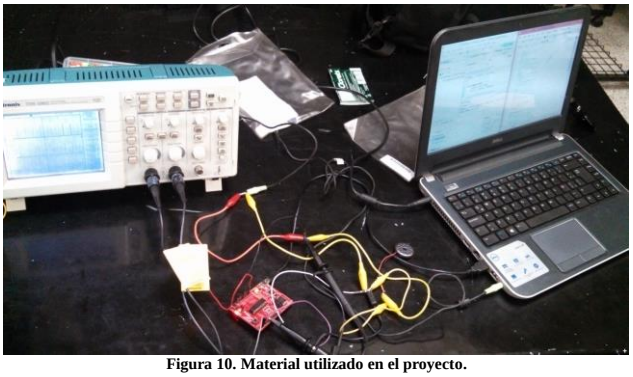

Figura 10. Material utilizado en el proyecto.

IV. OBSERVACIONES

Para poder obtener una señal analógica en Matlab, requeríamos de archivos que contuvieran datos muestreados de varias señales de ECG, al menos un registro de un ECG de una persona sana y otro u otros con patologías en frecuencia cardiaca o de amplitud en el complejo QRS, por lo que consultamos la página physionet.org. Una vez obtenidos dichos registros de las señales de ECG, las graficamos en Matlab y obtuvimos lo siguiente:

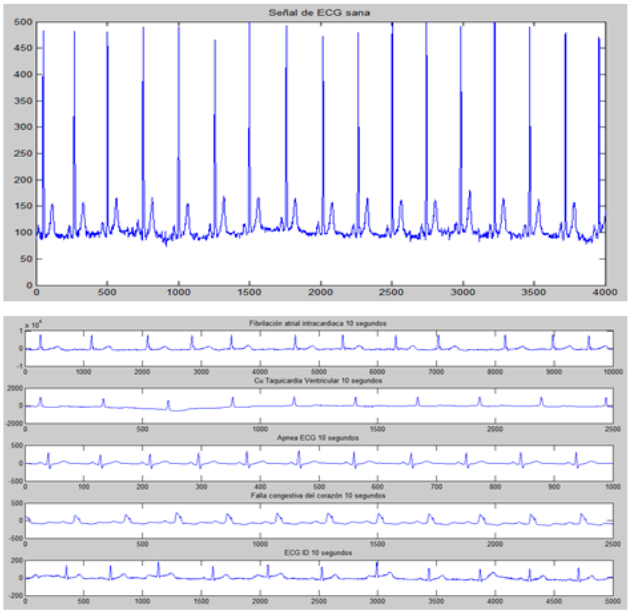

Figura 11. a) Señal de ECG sano. b) Señales de ECG con patologías.

Ya que teníamos los registros de las señales, tuvimos que decidir por donde sacar dichas señales de manera analógica, por lo que pensamos en algún adaptador que nos permitiera obtenerlas, pero después de hacer investigación, descubrimos que se podían obtener por el puerto de audio.

Empezamos por observar la señal de ECG de una persona sana, con la intención de comparar dicha señal,

contra las señales de ECG con patologías, sin embargo, al no contar con el tiempo suficiente para hacer las pruebas en el proyecto, tuvimos que optar por quedarnos con la comparación de la señal sana contra una isoeléctrica (generada el momento de que se detiene la señal). Por primera instancia, pensamos en usar el comando *winsound* de tal forma que se fueran almacenando los datos y los fuéramos obteniendo de manera analógica (figura 12). Pero decidimos utilizar por facilidad el comando *soundsc*, descrito en el anexo del código.

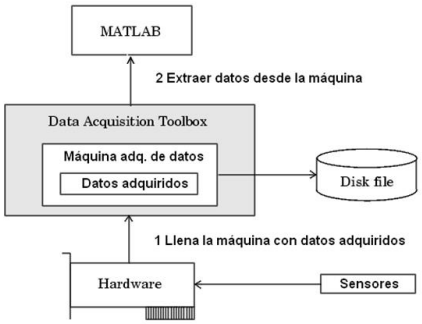

Figura 12. Almacenamiento de datos para obtener una señal analógica.

La señal que obtuvimos en el osciloscopio, se observa en los resultados.

Ya con la señal analógica, creímos que bastaría con detectar el punto R, del complejo QRS de un ECG, para ahorrar tiempo, esto sería, convertir todo el ECG en pulsos los cuales representarían únicamente la onda R, de esta manera, solo requeríamos conocer los parámetros de frecuencia y la amplitud para realizar el programa en el CCS. Pretendíamos hacer esta parte diseñando un circuito RC que fuera capaz de proporcionarnos dicho pulso y que se mantuviera el tiempo necesario para que el microprocesador lo detectara, pero el diseño y las pruebas del circuito requerían más tiempo, así que decidimos aplicar varias veces la señal del ECG directamente a la MSP430 al canal asignado para la conversión analógico-digital. Al realizar las pruebas, variando el volumen de la computadora pudimos apreciar que la amplitud de la señal de salida variaba y con esto logramos detectar la onda R debido a que sobresale del resto de las ondas de ECG, también logramos apreciar que algunas de estas ondas no eran detectadas, para resolver esto último, nos hacía falta ajustar el voltaje de comparación de la tarjeta y conocer el tiempo necesario para que se pudiera detectar todos las ondas R. Lo solucionamos colocando un delay, debido que al no tener delay, el microprocesador no tenía el tiempo necesario para detectar el pico, además de que la duración del pulso emitido depende de este delay por lo que fue necesario ajustarlo.

En lo referente a la retroalimentación, esperábamos realizarla programando de tal forma que en caso de que detectara un cero en la señal de ECG, se prendiera el led rojo, el cual serviría como indicación para que nuestro programa en Matlab volviera a ejecutarse y generara otra vez la señal de ECG de inicio pero, como la señal de ECG tiene componentes en cero, se prendía el led rojo continuamente y evitaba que fuera una solución. Al no ser una solución, decidimos amplificar la señal de ECG antes de meterla al microprocesador de tal forma que no hubiera valores en cero en ella. Usamos una amplificación de 5 veces como se muestra:

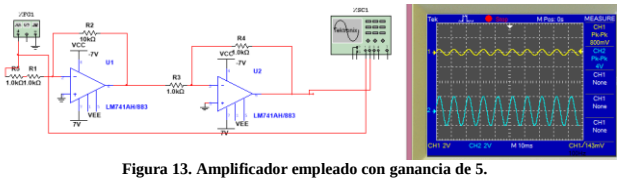

Figura 13. Amplificador empleado con ganancia de 5.

Al observar que incluso al amplificar la señal, seguía existiendo un voltaje que no permitía usar la condición de cero, tuvimos que replantear el problema y pensar otra forma para resolverlo.

Pensamos en hacer la condición de tiempo de que en caso de que el voltaje fuera mayor a 1022 (escala del ADC10) seguiría prendiendo el led verde, el cual sería t. y el punto en el que detecta otra vez el umbral y apaga el led verde sería t+1, por otro lado, el tiempo de separación entre la detección de pulso y pulso de la señal ya muestreado, lo llamaríamos p. Pretendíamos hacer que cuando t+1=p, se prendiera el led rojo, que es lo que indicaría una irregularidad en caso de detectar una señal de ECG con patologías, pero al momento de intentar implementarlo, los tiempos eran muy difíciles de ajustar y además el código intentado era muy complejo y no lográbamos que hiciera nada.

Ante el fracaso con los métodos anteriores, nos vimos en la necesidad de buscar otro método, el cual consistió en realizar un contador. En caso de no detectar un pulso en un lapso máximo de 3 segundos mandaría una señal para que el led rojo prendiera. Éste último fue el que resultó más efectivo, por lo que fue el que utilizamos, como se muestra en los resultados.
Al tener la programación del microprocesador, requeríamos que el pulso que obtenemos del led rojo regresara a Matlab, por lo que se nos ocurrió que con una grabación podríamos volver a meter la señal. Y si la señal de audio era de salida, requeríamos una de entrada, por lo que utilizamos la de micrófono.

El problema que esperábamos resolver era que la detección en Matlab fuera en tiempo real, y poderla manipular, por lo que pretendíamos utilizar un capacitor para retrasar la señal a la salida del microprocesador y permitir que diera el tiempo suficiente de grabación para

lograrlo, sin embargo, ningún capacitor era lo suficientemente pequeño y todos absorbían por completo la señal, así que buscamos otra manera.

Pretendíamos utilizar un contador de década (circuito CD4017) como se muestra:

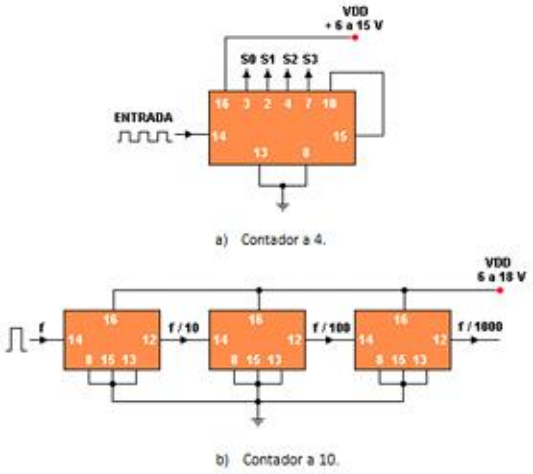

Figura 14. Circuito de contador analógico. a) Contador a 4. b) Contador a 10.

Esto tampoco sirvió para resolver el problema, ya que para una señal senoidal y para una cuadrada si funcionaba, pero para la señal de ECG no.

Al buscar una nueva solución, decidimos graficar la señal de entrada en Matlab que obteníamos al momento de que el led rojo prendía, para determinar si había algún cambio que pudiéramos detectar y utilizar para nuestra retroalimentación y así fue (como se muestra en los resultados). Sin embargo, al realizar la detección del pulso de entrada proveniente del led rojo del microprocesador, al probarlo con otra computadora, ese parámetro puede llegar a cambiar y habría que hacer una calibración para que la señal de retroalimentación volviera a funcionar, en este caso, ajustar en el código para que funcionara cuando el valor fuera de -0.9, ya que voltea la señal, como se muestra en la figura siguiente:

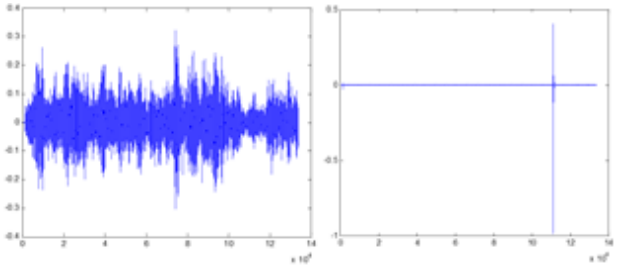

Figura 15. Gráfica de grabación del pulso generado por el led rojo, computadora 2. 14a (izq), señal grabada sin la conexión del cable auxiliar. 14b (der), señal grabada al recibir el pulso del led rojo.

Finalmente, no hubo problema al momento de programar Matlab para que repitiera el ciclo, ya que teníamos un parámetro que nos permitía detectar el pulso de entrada, metimos todo el código a un ciclo *while* para facilitar la retroalimentación.

V. Discusión

Al momento de ir avanzando en el proyecto, nos fuimos encontrando con diferentes dificultades, las cuales abordamos en equipo y solucionamos de tal manera que pudiéramos avanzar en el proyecto y retrasarnos lo menos posible. Usualmente los inconvenientes, son imprevistos y no son abordados hasta que uno se topa con ellos.

Debido a que la amplitud de la señal del ECG depende del volumen de la computadora, éste parámetro debe ajustarse para cada computadora. Intentamos hacer que Matlab emitiera la señal de ECG y que al mismo tiempo estuviera monitoreando si había una señal, en este caso la que emite el led rojo, para reiniciar la reproducción del archivo, pero Matlab no puede hacer ambas cosas, por ello no se puede realizar el monitoreo en tiempo real.

VI. TRABAJO A FUTURO

En caso de haber tenido más tiempo, este proyecto realmente pudo haber sido usado como una especie de holter, ya que con Matlab se pueden utilizar diferentes señales e incluso combinarse, por lo que la detección pudo haber sido utilizada y comparada con muchas muestras, al ser detectada por el microprocesador, y al ajustar parámetros, el led rojo en lugar de funcionar como una descarga, pudo haber funcionado como un marcador que determinara alguna patología.

Otra aplicación que puede usarse es como un interfaz de matlab a la MSP430 o viceversa, al saber cómo interconectar las señales, se pueden hacer otros tipos de estudios, no necesariamente de ECG, sino también de EEG, EMG, etc. Incluso podría utilizarse como reconocimiento de voz en matlab y muestrearse en el MSP430 para tomarlo como un indicador.

VII. CONCLUSIÓN

Code Composer Studio y la MSP430, fueron herramientas que nos permitieron realizar algunos de los objetivos planteados en nuestro proyecto, es indispensable conocer los alcances de estos dos elementos para no limitar el proyecto que se desea desarrollar.

Cuando se realiza un proyecto y se requiere usar un dispositivo como elemento principal, no se debe pensar en

dicho dispositivo como única herramienta de desarrollo, se debe hacer uso de todos los recursos que nos permitan hacer más eficiente el resultado al cual queremos llegar. Por ello decidimos emplear Matlab, como una herramienta que nos permitió acercarnos más a una situación real pues usamos un ECG que se obtuvo de una persona, lo cual nos hace reflexionar más sobre el deber del Ingeniero Biomédico y de las responsabilidades que adquiere con la sociedad.

Actualmente no se cuenta con un equipo de monitoreo constante que sea capaz de revertir un paro cardiaco. Creemos que un equipo que tenga la capacidad de reanimar al corazón de manera automática reduciría el riesgo de daños cerebrales por falta de oxigenación en el paciente.

Usar más de una computadora para hacer el monitoreo en tiempo real no es eficiente, lo ideal es que se pueda hacer con una sola y se requiere un programa que permita realizar la reproducción del archivo y al mismo tiempo monitorear las señales de la MSP430.

BIBLIOGRAFÍA

[1] Texas Instruments. MSP430x2xx User's Guide. Literature. Number: SLAU144J. December 2004–Revised July 2013.

[2] Davies. John, "MSP430 Microcontroller Basics Burlington, MA: Newnes, 2008.

[3] ¿Qué es un desfibrilador? Sanimedic group. Fecha de consulta 26 mar 2014. http://www.sanimedic.com/es/productos/sanicardio/ique-es-un-desfibrilador.html

[4] Desfibrilación y cardioversión eléctrica. Wikipedia. Última modificación 19 mar 2014. Fecha de consulta 26 mar 2014. http://es.wikipedia.org/wiki/Desfibrilaci%C3%B3n_y_cardioversi%C3%B3n_el%C3%A9ctrica

ANEXO 1
Código usado en Matlab

```
clear all                %Limpiar todo
clc
esp=1;                   %espera
while esp >= 0.9           %Parámetro dado por nuestra gráfica de grabación

load('senal_ecg.mat');     % cargamos la señal de ECG

x=ECG(1,:);                  % Definimos x como una señal de ECG
Fs=0:0.004:40-0.004;         % Definimos nuestra frecuencia de muestreo, dado que la
señal de entrada original estaba muestreada a 250 Hz y tenía 10000 puntos, deducimos
que tomaba muestras cada 0.004 segundos hasta la totalidad de la señal
que era de 40 segundos

Fa=0:0.0001:40-0.0001;       % Definimos una frecuencia de muestreo artificial, es
decir que toma muestras a un intervalo de tiempo menor de lo que se muestreó la
señal originalmente, en este caso el intervalo es de 0.0001 segundos.

yyi=spline(Fs,x,Fa);         % Dado que la señal original de ECG está muestreada a
una Fs y estamos simulando que tomó muestras a un menor intervalo de tiempo, les
asignamos a esas muestras los valores más cercanos muestreados realmente, esto es
para que la señal se vea más clara


Fay=20000;                   % Utilizando el teorema de Nyquist muestreamos al doble
de la frecuencia máxima
soundsc(yyi,Fay);            % Convertimos la señal digital de ECG, muestreada con
la nueva frecuencia de muestreo Fa, a una señal analógica que saldrá por el puerto
de audio de la computadora.

Fs =44600;
y = wavrecord(3*Fs,Fs,1);

i=0;
for(i=1:length(y))
    if y(i) >= 0.9;      %Si hay un valor mayor a 0.9
        esp=y(i)         %Reinicia el ciclo a esp
        break            % para
    else
        esp=0;           % En otro caso, no reinicies
    end
end

plot(y)                  %Graficar la señal de grabación
fprintf(1,'Fin')         %Mostrar en pantalla el texto fin para saber que terminó

end
```

ANEXO 2

Código en Code Composer Studio para muestrear la señal, detectar la onda R y generar la señal para retroalimentación

```
#include <msp430g2553.h>

int v;

int tempOut ()
{
   long t=0;
   long n=100000;
   for (t=0;t<n;++t){

      ADC10CTL0 |= ENC + ADC10SC;
      while(ADC10CTL1 & BUSY);
         v=ADC10MEM;
         ADC10CTL0&=~ENC;

      if (v>1022){
         P1OUT |= BIT6;
         P1OUT &= ~BIT0;
          __delay_cycles(100);
          P1OUT &= ~BIT6;
          t=0;
      }
      else if(v<1022 && t==20000){
            P1OUT |= BIT0;
            _delay_cycles(100);
      }
   }
}

void main(void) {
    volatile int temp;
   WDTCTL = WDTPW | WDTHOLD;

   P1DIR |= 0x41;
   P1OUT &= ~0X41;

   temp= 0;

    ADC10CTL0=SREF_1 + REFON + ADC10ON + ADC10SHT_3;
    ADC10CTL1=INCH_4 + ADC10DIV_3;

    while (1){
                     temp=tempOut ();
      }
}
```

ANEXO 3

Código en Code Composer Studio para conocer el número de iteraciones en un ciclo for necesarias, para repetir un proceso durante un segundo.
Utilizando el reloj de 1MHz son necesarias 100,000 iteraciones en un ciclo for para perdurar por un segundo, mientras que el delay utiliza 1,000,000

```
#include <msp430g2553.h>

/*
 * main.c
 */

int tempOut (){
   long t=0;
   long n=100000;
   for (t=0;t<n;++t){
      P1OUT |= BIT0;
}

    P1OUT &= ~BIT0;
    _delay_cycles(1000000);
    tempOut ();
}

int main(void) {
    WDTCTL = WDTPW | WDTHOLD;  // Stop watchdog timer

    P1DIR |= BIT0;
    P1OUT &= ~BIT0;

    tempOut ();
}
```