Universidad de la República
Facultad de Ingeniería

IMPETOM B - Inyección, adquisición y tratamiento inicial de señales para tomografía torácica por impedancia eléctrica en placa de pruebas Omap-L137

Memoria de proyecto presentada a la Facultad de Ingeniería de la Universidad de la República por

Nicolás Alfaro, Fernanda Martinucci, Martín Arregui

en cumplimiento parcial de los requerimientos para la obtención del título de Ingeniero Electricista.

Tutor

Eduardo Santos ..... Universidad de la República
Franco Simini ..... Universidad de la República

Tribunal

Juan Cardelino ..... Universidad de la República
Juan Pablo Oliver ..... Universidad de la República
Washintong Olivera ..... Universidad de la República

Montevideo
31/07/2015

IMPETOM B - Inyección, adquisición y tratamiento inicial de señales para tomografía torácica por impedancia eléctrica en placa de pruebas Omap-L137, Nicolás Alfaro, Fernanda Martinucci, Martín Arregui.

Esta tesis fue preparada en LaTeX usando la clase iietesis (v1.1).
Contiene un total de 127 páginas.
Compilada el sábado 12 diciembre, 2015.
http://iie.fing.edu.uy/

"El verdadero progreso es el que pone la tecnología al alcance de todos."

Henry Ford.

Esta página ha sido intencionalmente dejada en blanco.

Tabla de contenidos

Esta página ha sido intencionalmente dejada en blanco.

Capítulo 1

Introducción

Mucho se ha avanzado en el último tiempo en tomografía, utilizando diferentes tecnologías, por transmisión de ultrasonido, de rayos X, magnética. Algunas de ellas invasivas y otras muy costosas. Surge entonces, desde hace ya varias décadas, un nuevo método llamado EIT por sus siglas en inglés (Tomografía de Impedancia Eléctrica).

La EIT fue introducida por Kiber Barber y Brian Brown [2] en los años 80 y desde entonces ha visto un crecimiento notable utilizándose para diferentes propósitos tales como monitoreo de funciones cerebrales o la obtención de imágenes mamarias.

La inyección de corrientes de alta frecuencia (50 KHz) en amplitudes no percibidas por el organismo humano (del orden de 2 mA) y posterior medida de voltajes cutáneos, permite estimar la distribución de la conductividad eléctrica al recorrer puntos fijos del tórax del paciente. Las matrices de valores superficiales son procesadas para obtener imágenes de cortes tomográficos.

En particular, el tratamiento del edema de pulmón, se basa entre otros parámetros en la estimación de la cantidad del volumen alveolar ocupado por líquido, y este método, conocido como Tomografía de Impedancia Eléctrica, permite plantear el desarrollo de un equipo de bajo costo, no invasivo y de aplicación prolongada que muestre a lo largo de los días la "mancha" de líquido en los pulmones y su evolución.

Desde 1995 el Núcleo de Ingeniería Biomédica (nib) ha desarrollado sucesivamente circuitos, programas de reconstrucción y prototipos completos de EIT, bajo el nombre de IMPETOM (IMPEdancia TOMografía), con resultados de pruebas en voluntarios sanos y fantomas.

1.1. Objetivo del proyecto global IMPETOM

IMPETOM pretende la implementación de un sistema completo e integrado de EIT para la monitorización del edema de pulmón.

1.2. Objetivo del proyecto IMPETOM-B

El objetivo de *IMPETOM-B*, es la reinigeniería de un tomógrafo de uso clínico que prevé la utilización de un "Evaluation Board" (*OMAP-L137 EVM*) incorporando la tecnología de doble núcleo DSP+GPP, que aporta un enfoque nuevo al diseño de la circuitería. En este sentido, se explora la generación de señal y la posibilidad de realizar la medida de voltajes resultantes en la piel del paciente, mediante el códec de audio de la plaqueta y los canales de entrada y salida de la misma, aprovechando la capacidad de demodulación en el DSP. El resultado esperado del funcionamiento es obtener una matriz de datos según norma IMPETOM.

Capítulo 2

Estado del arte

Una técnica cada vez es más empleada para caracterizar diversos materiales es la espectroscopia de impedancia eléctrica, que estudia el comportamiento de la impedancia eléctrica del material en el dominio de la frecuencia. Al igual que otros cientos de usos, algunas aplicaciones biomédicas (como es el caso de *IMPETOM-B*), también hacen uso de técnicas de medición de propiedades eléctricas de los materiales mediante estímulos de corriente.

Como parte de esto, es imprescindible poner al lector en conocimiento acerca de diversos elementos que se debe tener presentes para contextualizar los aspectos constructivos más importantes de la EIT, así como también la necesidad particular de la misma.

Dentro de este capítulo, se empieza por la descripción del problema al cual está orientado el desarrollo (Edema de Pulmón), para luego continuar con una breve reseña de Bioimpedancia (impedancia del cuerpo humano), y finalmente terminar con los principales conceptos a tener presentes a la hora de hablar de un prototipo funcional que realice tomografías de impedancia eléctrica.

2.1. Edema de pulmón

El Edema de pulmón es la acumulación anormal de liquido en el pulmón, que impide la normal oxigenación de la sangre, ocasionando falta de oxigeno en los tejidos del cuerpo, además de una disminución en la ventilación. Generalmente las causas son insuficiencias cardíacas. A medida que el corazón deja de funcionar, la presión en las venas que van al pulmón comienza a elevarse y el líquido es impelido hacia los alvéolos. Este líquido, se comporta como una barrera que interrumpe el movimiento normal del oxigeno a través de los pulmones, provocando dificultad para respirar. También puede ser causado por una lesión directa en el pulmón, como la causada por gas venenoso (común en los incendios productores de grandes humos) o infección severa. El edema pulmonar puede ser una complicación de un ataque cardíaco, filtración o estrechamiento de las válvulas cardíacas o cualquier

enfermedad cardíaca que ocasione, ya sea debilitamiento o rigidez del músculo cardíaco.

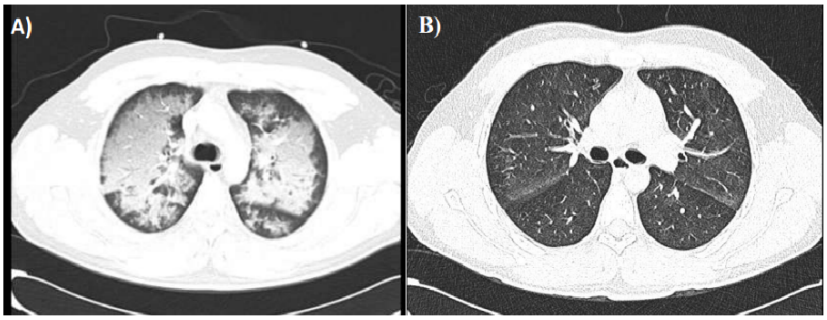


Figura 2.1: A al izquierda (A), imagen tomográfica de pulmón con edema. A la derecha, imagen tomográfica de pulmón sin edema (B). [3]

Los medicamentos que se pueden recetar abarcan diuréticos, los cuales eliminan el exceso de líquido del cuerpo, medicamentos para fortalecer el miocardio, controlar el ritmo cardíaco o aliviar la presión sobre el corazón. De todas maneras, el tratamiento del edema pulmonar casi siempre es en una sala de urgencias de un hospital, debido a que también suele ser una complicación común de la ventilación mecánica. Con los métodos tradicionales actuales, es necesario movilizar el paciente hasta una sala de rayos X, para observar el avance del edema en el pulmón, lo que significa un alto riesgo.

Es por esto último, que una unidad que realice un monitoreo de la evolución del estado de los pulmones ante este problema, sin que sea invasivo, o de alto riesgo (movilizamiento del paciente de un CTI), tiene gran potencial para contribuir con la calidad de atención en hospitales. [6]

2.2. Bioimpedancia

Dado que el objetivo detrás de IMPETOM es la medida de la impedancia del cuerpo humano (mas particularmente en el tórax, alrededor de los pulmones), es necesario disponer también de este concepto.
Se sabe que la impedancia del cuerpo humano es función de diversos factores tales como la tensión aplicada, la frecuencia, la duración del paso de la corriente, la temperatura, el grado de humedad de la piel, la presión del contacto, la dureza de la epidermis, entre otros. También depende de las diferentes partes del cuerpo en la que se mide tal como, la piel, los músculos, la sangre, etc. En una primera aproximación, todas estas características pueden ser modeladas conjuntamente con una impedancia compuesta por elementos resistivos y capacitivos.
Durante el paso de la electricidad, la impedancia de nuestro cuerpo se comporta como una suma de tres impedancias en serie: impedancia de la piel en la zona de entrada, la impedancia interna del cuerpo y la impedancia de la piel en la zona de salida.

Estudios previos [4], demostraron que es posible modelar la impedancia del cuerpo humano de la siguiente manera:

$$Z = R_{\text{ínf}} + \frac{R_0 - R_{\text{ínf}}}{1 + (jw\tau_Z)^{\alpha}} \tag{2.1}$$

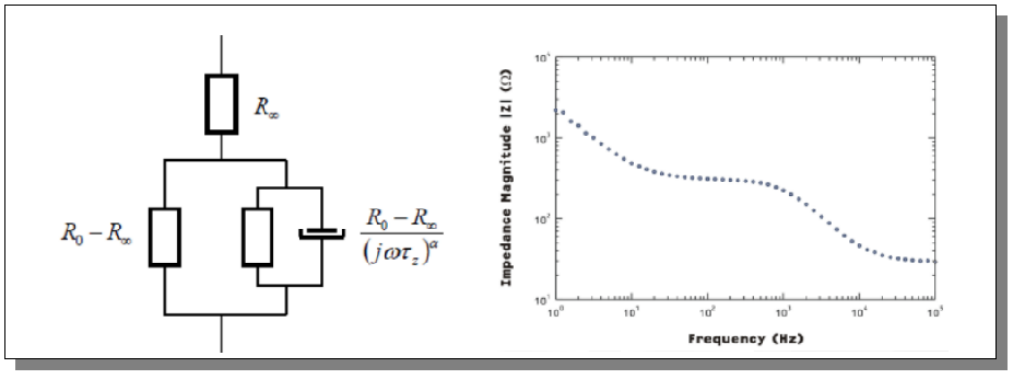


Figura 2.2: Modelo de impedancia del cuerpo humano. [4]

.

2.3. Historia del proyecto IMPETOM

El Núcleo de Ingeniería Biomédica (nib) de las facultades de Medicina e Ingeniería, y el hospital de Clínicas, ha desarrollado desde mediados de los 90' sucesivos circuitos, programas de reconstrucción y prototipos completos de EIT bajo el nombre de IMPETOM (IMPEdancia TOMográfica). Desde entonces y en el marco de ese desarrollo, se han impulsado importantes proyectos, entre los que se destacan *IMPETOM-I* [8], *IMPETOM-C* [10] e *IMPETOM* [7].

Como una breve reseña diremos que *IMPETOM-I*, implementó el software dedicado a la reconstrucción de un corte tomográfico a partir de una matriz de impedancias relevada sobre el paciente. Su presentación, es la de un programa de software ejecutable sobre un PC que despliega en la pantalla una imagen resultante de aplicar un algoritmo determinado (elegible entre varios algoritmos de reconstrucción) sobre la matriz de impedancias relevada sobre el tórax del paciente, que recibe como dato de entrada en un formato adecuado (formato *IMPETOM*). 2.3

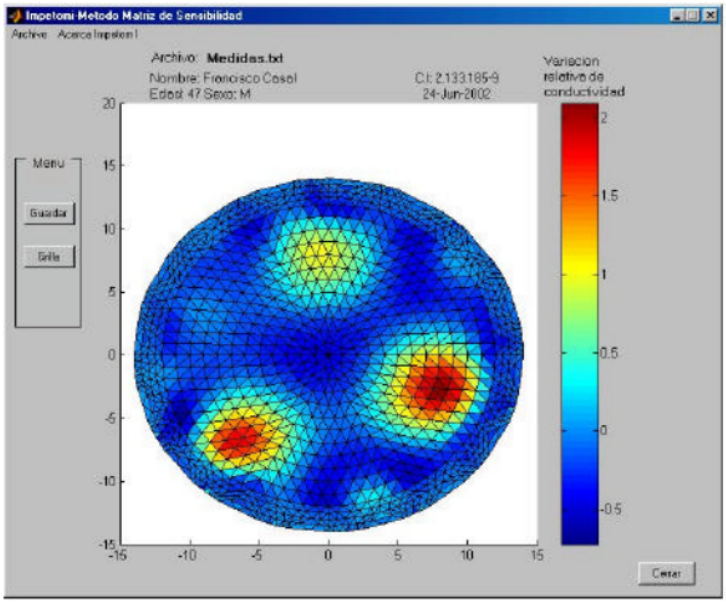


Figura 2.3: Reconstrucción de imagen de *IMPETOM-I* a partir de matriz de datos [8]

.

El segundo proyecto (aunque simultáneos en tiempo con *IMPETOM-I*), *IMPETOM-C*, consiste en la electrónica necesaria para medir las impedancias corporales mediante la inyección de las corrientes, a través de los electrodos dispuestos sobre el paciente, y la medición de los voltajes resultantes 2.4.

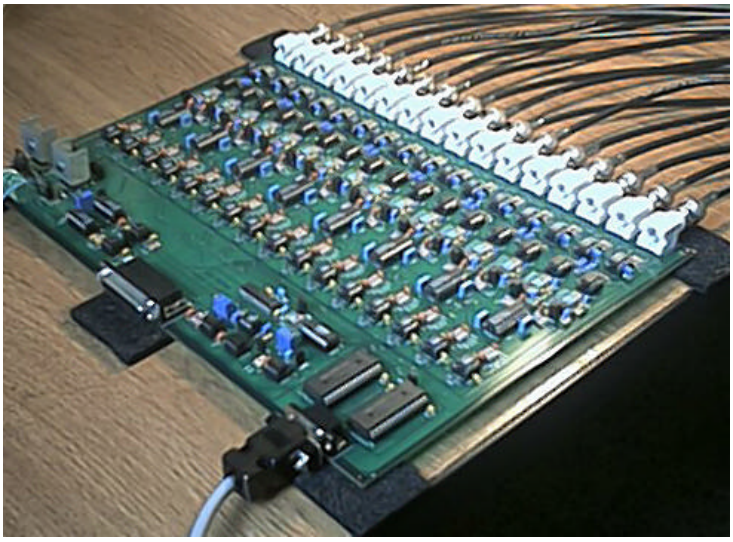


Figura 2.4: Hardware *IMPETOM-C*. Se pueden observar las entradas paralelas para la conexión de electrodos (16). [10]

El proyecto *IMPETOM* incorpora los dos desarrollos anteriores y los unifica en un prototipo utilizable para la investigación, con posibilidades (basadas en futu-

ros desarrollos) de comercialización, seguro para su utilización en seres humanos, modular (ya que permite la incorporación de otros algoritmos de reconstrucción), de manejo intuitivo y amigable, y compacto 2.5.

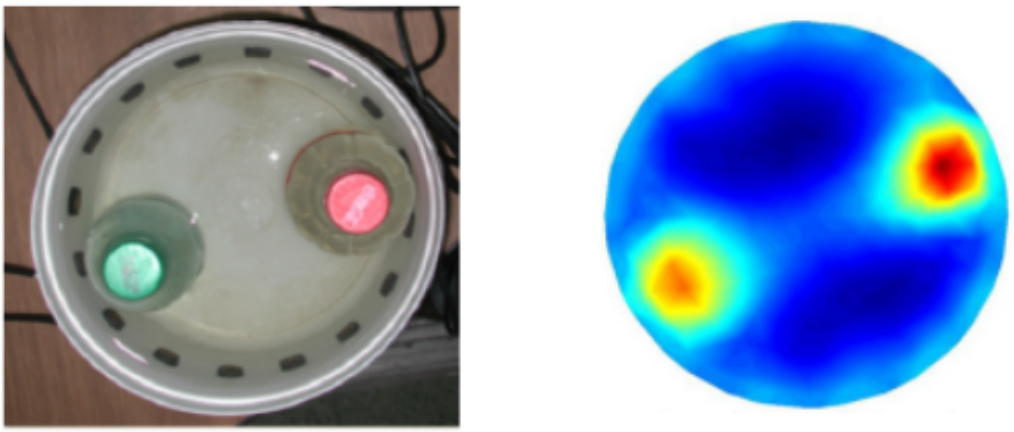


Figura 2.5: Reconstrucción de imagen tomográfica de *IMPETOM* unificando *IMPETOM-I* e *IMPETOM-C*. [7]

2.4. IMPETOM-B dentro de IMPETOM

Para ubicar nuestro proyecto dentro de la globalidad de el conjunto de proyectos IMPETOM obsérvese el diagrama de bloques de la figura 2.6 que corresponde al de un equipo completo de EIT.

IMPETOM-B concentra los bloques encargados de recolectar y procesar los datos de voltaje (bloques Synchronic Demodulation, Sinusoidal signal generation, Control and Storage), no así los bloques de reconstrucción de imagen, fuente de corriente, multiplexado de electrodos y los electrodos propiamente dicho.

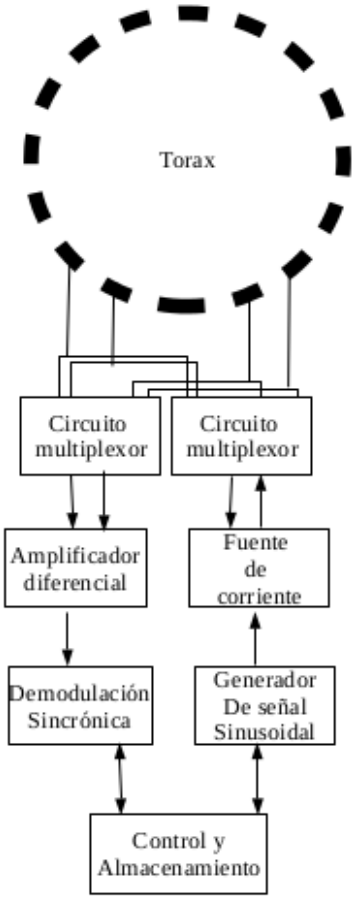


Figura 2.6: Diagrama de bloques de equipo completo de EIT

2.5. Introducción al Funcionamiento

Dado el modelo de impedancia corporal (resistivo/capacitivo) descripto en la sección 2.2, la misma se puede caracterizar mediante la inyección de una corriente conocida de valor constante y de frecuencia única a través de un electrodo y retirarla por otro, mientras que las medidas de los voltajes son realizadas entre los restantes electrodos (esta técnica es llamada de *corrientes constantes*).

Existes otras técnicas, llamada de *corriente óptima* [14], que inyectan corrientes de diferentes patrones simultáneamente (por ejemplo frecuencias), lo cual proporciona información complementaria (sobre todo en lo referente a la componente reactiva de la mencionada impedancia), mientras que la medida de voltaje es realizada en los mismos electrodos que inyectan corriente o en un grupo independiente de éstos.

En el caso de *IMPETOM*, se tiene para cada medida, dos pares de electrodos, uno para la inyección de corriente en la región (par de corriente), y un segundo par para medir el voltaje (par de voltaje). Esta configuración se denomina tetra-

polar o de 4 electrodos [14].

Aunque varios autores han investigado acerca de la forma óptima de inyectar las corrientes a través de los electrodos, Brown [14] llevó a cabo la estrategia mencionada aplicando la corriente y midiendo el voltaje entre pares de electrodos adyacentes. Usando N electrodos puestos alrededor de un objeto, pueden obtenerse *N-1* medidas de voltaje independientes, y la última medida (medida N) puede encontrarse por superposición.

Para el caso de IMPETOM, la conexión al paciente se realiza a través de 16 electrodos por los cuales se inyecta la señal de corriente, en un par de electrodos adyacentes, y se mide los voltajes diferenciales resultantes en los 14 electrodos restantes (figura 2.7)

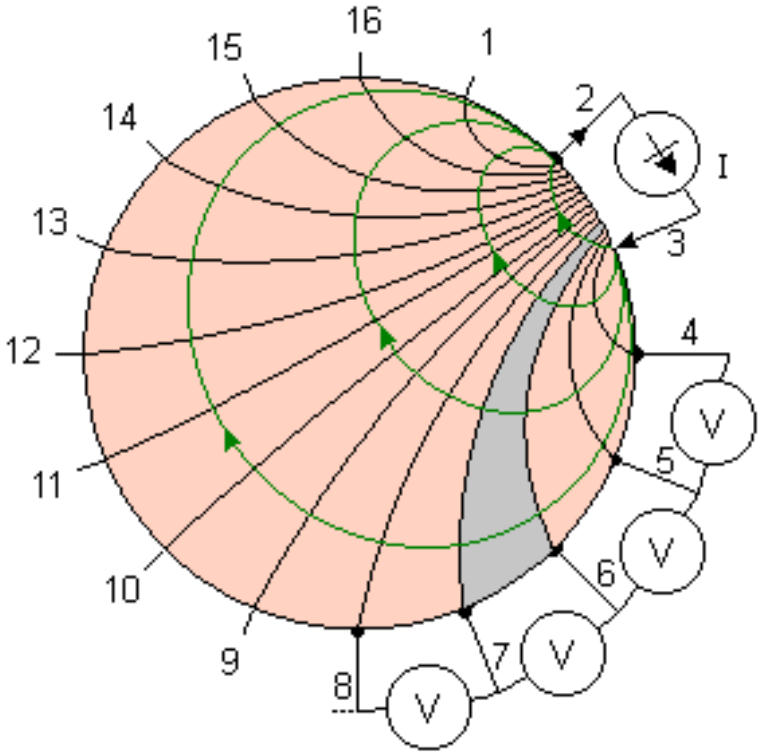


Figura 2.7: Diagrama de disposición de Electrodos. Método de electrodos adyacentes. [14]

Cada caída de tensión es obtenida de los sucesivos pares de electrodos de medida em1-em2, em2-em3, etc. hasta completar las 16 medidas. Finalmente, se pasa al siguiente par de electrodos de inyección obteniendo un nuevo perfil para luego repetir todo el proceso (figura 2.8).

Como otra alternativa a la arquitectura de electrodos adyacentes, un formato entrelazado (interleaved) fue adoptado por algunos autores, en el que electrodos especializados de inyección de corriente y electrodos de medida de voltaje son colo-

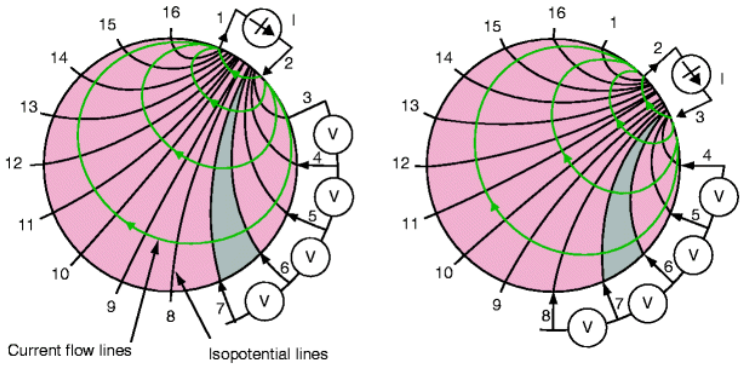


Figura 2.8: Método de electrodo adyacente, rotación de electrodos [12]

.

cados en forma alternada alrededor del objeto. Avis y Barber [14], han investigado otras dos configuraciones en que los pares de electrodos de corriente se colocan con separaciones angulares de 90° (configuración cruzada), y 180° (configuración polar).
Una vez obtenidas las medidas de voltaje, es necesario realizar un pre-procesamiento de los datos para obtener la información relevante de la impedancia, o sea, la información que permita reconstruir el mapa de conductividad corporal (más adelante se verá que se trata de la parte real de la misma). Luego de este primer procesamiento de datos, los resultados se reciben en otro módulo mediante la algoritmia necesaria y en un archivo de texto plano (*archivo.txt* en formato IMPETOM). Relevando los conjuntos de medidas con el mismo juego de electrodos y en las mismas posiciones en el cuerpo, el algoritmo produce imágenes diferenciales consistentes de la región a ser reconstruida.

Según el poder de calculo del hardware donde corra el programa de reconstrucción de imágenes, la velocidad y precisión con que se recolecten y manejen los datos obtenidos de los electrodos, es posible desplegar la información en tiempo real o simplemente un diagnóstico de la evolución obtenida.

Cualquiera sea el resultado final, el objetivo es el mismo, brindar al especialista un informe visual (ver figura 2.5) que le permita monitorear la evolución del paciente sin transgredir aún mas su salud con los beneficios que esto conlleva en la calidad del tratamiento.

Capítulo 3

Hardware *IMPETOM-B*

Anteriores proyectos desarrollados de *IMPETOM*, implementan el hardware de manera integral, ensamblando cada componente discreto. Este tipo de solución tiene importantes ventajas desde el punto de vista de los costos, pero en contrapartida, hace que las dificultad del diseño sean importantes, además de que se obtiene un sistema donde los ruidos eléctricos y las imperfecciones de los componentes discretos, dominan las medidas obtenidas, haciendo difícil obtener una reconstrucción y por ende, una imagen razonable.
Para atacar este problema, es que se decide tomar el camino de utilizar DSPs según se deduce en la documentación [13].

Estos sistemas tienen por objetivo la evaluación o iniciación del ingeniero en el uso de cierto modelo de DSP y las herramientas desarrolladas, muchas veces con fines didácticos. La ventaja de utilizar estas tarjetas es la de tener un sistema completo que contiene el DSP, memoria, demoduladores que incluyen los conversores A/D y amplificadores de ganancia programable (PGAs), comunicación con la computadora, puertos para uso general, el circuito para programar el DSP; todo esto ya probado y dispuesto de la forma más óptima pronto para obtener resultados desde el momento [13].

OPCIÓN	VENTAJAS	DESVENTAJAS
Componentes discretos sin DSP	• Costo	• Ruidos eléctricos • Imperfecciones en los componentes
Tarjeta de evaluación	• Sistema completo • Armado de forma óptima	• Sin flexibilidad para elegir componentes • Costo
Componentes discretos con DSP	• Sistema adaptado a nuestro problema • Costo menos a un Starter kit	• Difícil que todos los componentes trabajes de forma armónica • Difícil construcción y armado de la placa

Tabla 3.1: Tabla de ventajas y desventajas según el tipo de implementación del sistema. [13]

El proyecto *IMPETOM-B*, se desarrolla partiendo de la base de que ya se tiene disponible el hardware donde se implementará el sistema, es decir, se cuenta con una placa de evaluación, en este caso, la *EVM OMAP-L137* de la compañía *Texas Instruments*, y consiste en una plataforma independiente que permite desarrollar y evaluar variadas aplicaciones para el procesador OMAP-L137 de la misma empresa. La figura 3.1 muestra el diagrama de bloques de la placa.

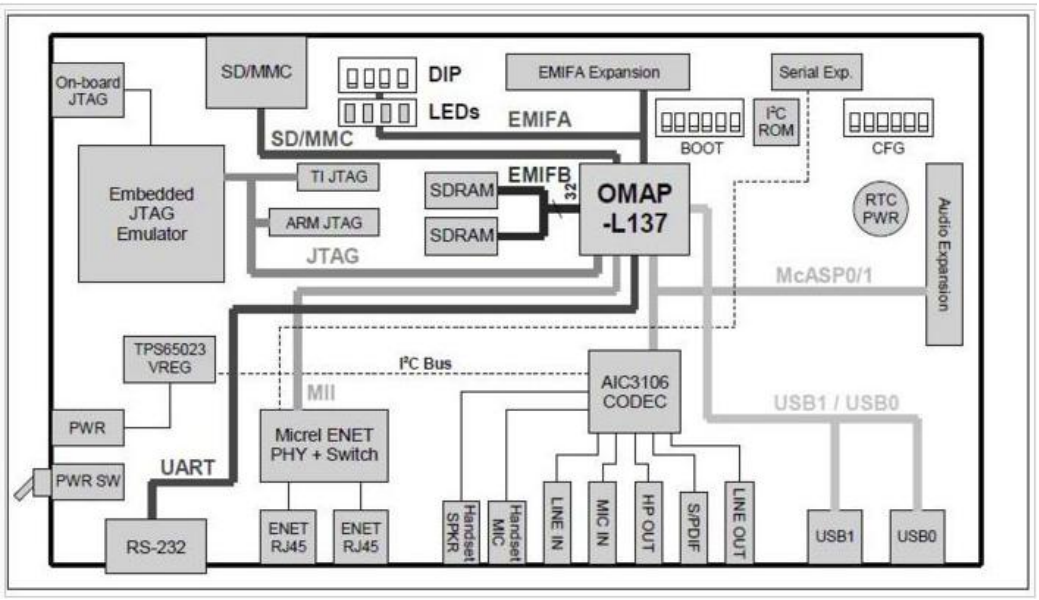


Figura 3.1: Diagrama de bloques de la *EVM OMAP-L137*. [23]

Se puede observar en la figura 3.2 un diagrama pormenorizado de los elementos a utilizar en el proyecto donde se agrega a la placa EVMOMAPL137 una fuente de corriente sencilla junto a un fantoma de pruebas.

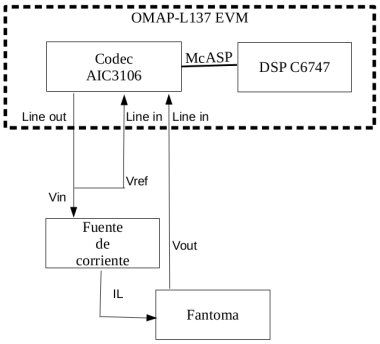


Figura 3.2: Diagrama Pormenorizado del proyecto IMPETOM-B

3.1. Principales Características

Esta placa tiene incorporado un sistema multi-core system on-chip (SoC) que permite trabajar simultáneamente con dos o mas chips de igual o diferente arquitectura. En el caso del modelo integrado al *OMAP-L137*, se tiene un sistema de arquitectura heterogénea (un DSP (*Digital Signal Processor*) y un procesador con arquitectura ARM (*Advanced RISC (Reduced Instruction Set Computing) Machine)*) que permiten realizar múltiples tareas paralelas, especificas para cada core (procesamiento de señales en DSP y administración para el ARM), aumentando así la velocidad de procesamiento.
Además cuenta con una serie de periféricos, y entradas y salidas para diversos propósitos. Los mas importantes para este proyecto son el codec de audio AIC3106, McASP (Multichannel Audio Serial Port), y entradas y salidas estéreo Line IN y Line OUT.

3.2. Procesador OMAP-L137

Como ya se mencionó, el procesador OMAP-L137 contiene dos núcleos primarios, un procesador ARM926EJ-S para realizar tareas de control general del sistema, pudiendo operar hasta los 300 MHz, y un procesador de punto flotante C6747 VLIW DSP para manejar eficientemente la comunicación y tareas de procesamiento de datos.

El ARM926EJ-S es un núcleo de procesamiento de tipo RISC (por su siglas en ingles de *Reduced Instruction Set Computing*, Conjunto Reducido de Instrucciones de Computadora) de 32 bits que trabaja con instrucciones de 32 o 16 bits, y procesos de 32, 16 o 8 bits de datos. Este core utiliza programación paralela por lo que todas las partes del procesador y el sistema de memoria pueden operar continuamente. Además cuenta con un coprocesador 15 (*CP15*), módulos de protección, y Unidades de Administración de Memoria (MMUs por sus siglas en ingles de *Memory Management Units*) para datos y programas. Por último, el núcleo ARM cuenta con 8KB de RAM (en formato de tabla de vectores) y 64KB de ROM.

El núcleo DSP del OMAP-L137 usa arquitectura basada en caches de dos niveles. El nivel 1 se sub-divide a su vez en dos partes, el *cache de programa Nivel 1* (L1P) y el *cache de datos de Nivel 1* (L1D). El primero es un cache directamente mapeado en 32KB, mientras que el (L1D) es un cache asociado en conjunto de dos vías, también de 32KB. El cache de nivel 2, es en realidad un cache de programa nivel 2 (L2P) que consisten en un espacio de memoria de 256KB compartidos entre programa y datos. Además, este fragmento de memoria puede ser accedido por el ARM, algo sumamente necesario para la comunicación y trasmisión de los datos entre el DSP y el ARM.

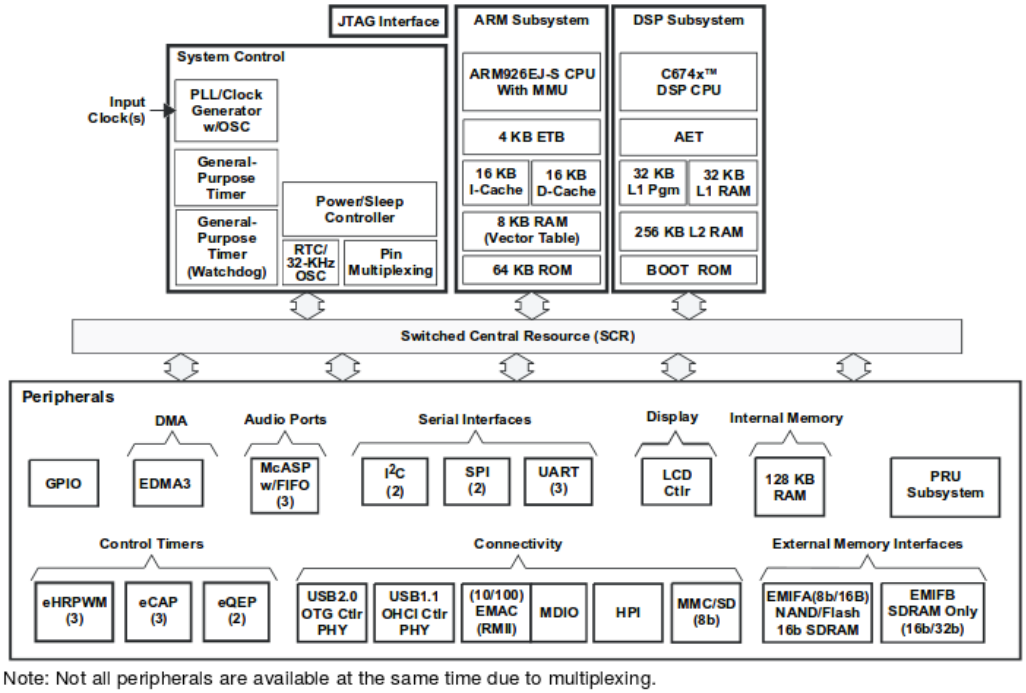


Figura 3.3: Diagrama de bloques del sistema de evaluación *EVM OMAP-L137*. [23]

3.3. Codec de Audio AIC3106

Se trata de un codec estéreo *TLV320AIC3106*, el cual en líneas generales, permite al DSP transmitir y recibir audio analógico. Muestrea las señales analógicas en la entrada de *line_in* y la convierte a datos digitales para que pueda ser procesado por el DSP. Luego que el DSP procesa los datos, éste nuevamente utiliza el codec para reconstruir la señal analógica a ser enviada por el puerto de salida de audio. El codec se comunica mediante dos buses: el bus I2C, el cual es usado para el control de la interfaz del codec, mientras que McASP controla el flujo de audio. La interfaz de la señal de audio se realiza mediante conectores jacks de 3.5mm, que corresponden a las entradas y salidas. Un generador interno de frecuencia de muestreo subdivide el reloj del sistema para generar frecuencias de audio comunes, la cual se setea mediante un registro interno del mismo.

Más en detalle, se trata de un (codificador-decodificador) de audio estéreo de baja potencia con amplificadores de ganancia ajustable, así como múltiples inputs y outputs programables en configuración single-ended o diferencial. Además contiene control digital de pre-amplificadores estéreo, control automático de ganancia (AGC) con capacidad de mezclar entre múltiples entradas analógicas, así como filtros programables para eliminar ruido. En la figura 3.4 se muestra una conexión típica del *AIC3106*.

Algunas de las características mencionadas, son necesarias para la operativa de la nueva implementación de *IMPETOM-B*. Por ejemplo, las salidas analógicas

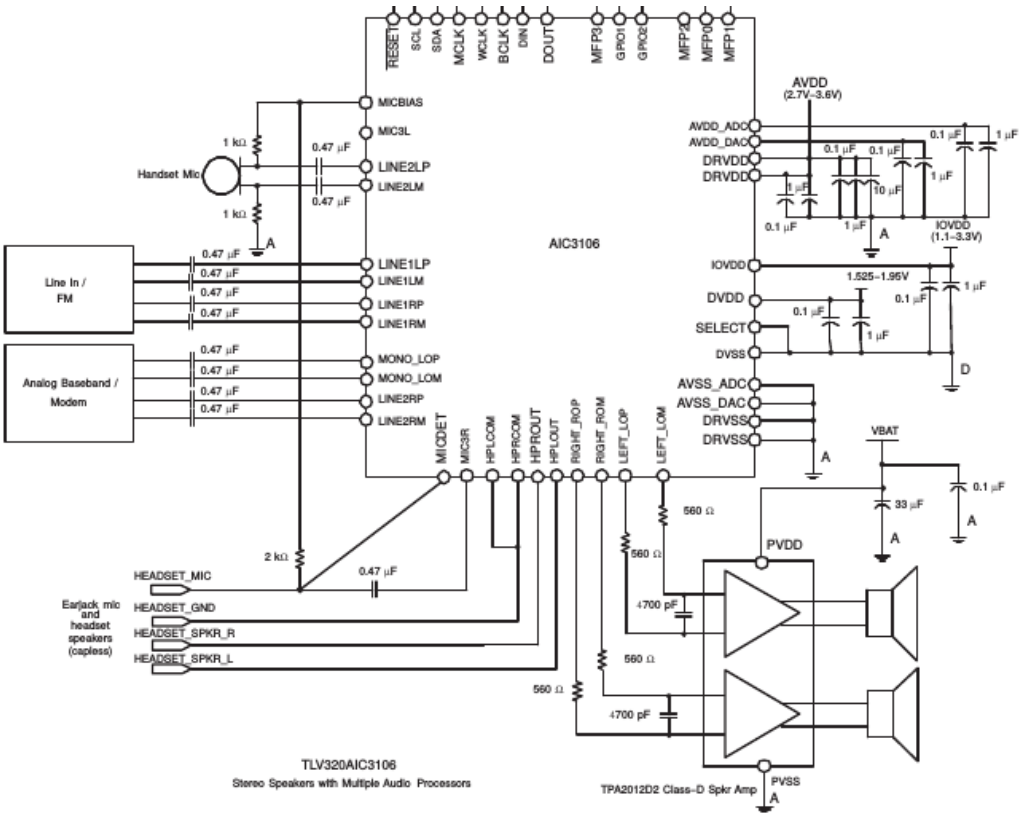


Figura 3.4: Esquemático de conexión interna del codec de audio *AIC3106*. [22]

pueden proveer la señal necesaria para controlar la fuente de corriente, y múltiples entradas para muestrear la señal proveniente de los electrodos.

El DAC de audio estéreo soporta velocidades de muestreo que van desde los $8kHz$ hasta los $48kHz$, frecuencia que en principio son suficientes para muestrear señales de hasta $24kHz$ (frecuencia aproximada a la que se inyectará la corriente, $16KHz$ - 5).

Como entradas, el *AIC3106* incluye diez pines de audio analógico (los más importantes para el proyecto son: *LINE2LP*, *LINE2LM*; luego se tiene *LINE2RP*, *LINE2RM*, *LINE1RP*, *LINE1RM*, *LINE1LP*, *LINE1RM*, *MONO_LOP*, *MONO_LOM*), configurables hasta en cuatro pares diferenciales, más un par de una sola terminal. Cada una de las terminales se conecta a amplificadores operacionales diferenciales (uno por ADC/canal PGA) con la posibilidad de activar sólo un conjunto de interruptores por amplificador operacional a la vez. Luego, como salida se tienen las señales DAC_L/R (salidas del DAC de audio estéreo), que pueden ser dirigidos por control de registro basado en los requisitos del sistema.

En la placa disponible se han implementado dos configuraciones de salida estéreo, *line_out* y *headphones*, y dos configuraciones de entrada, *line_in* (estéreo) y *mic*

(mono), con conectores para jacks de 3.5mm.

Registros de Control Fundamentales

Para utilizar el codec de audio como herramienta adquisidora de voltajes, a través de sus canales de grabación, y como DDS (*Direct Digital Syntetizer*) a través de sus canales de salida, es necesario conocer los registros que permiten su manipulación. Se analizan entonces, dichos registros partiendo siempre de la configuración base realizada en el software test (*aic3106_test*) incluido con el kit de la placa.

Configuración Base

En el capítulo *Software test del codec*, se utiliza la configuración de la rutina *aic3106_loop_linein()* como base a partir de la cual se confecciona la función *Grabar()* de nuestro proyecto. La configuración efectuada en esta rutina del software test de aic3106 incluido en el kit de software de la EVM OMAPL137 es la siguiente:

- Frecuencia de muestreo (registros del 3 al 7). Se desea trabajar a la máxima frecuencia posible y con el mayor número de canales disponibles, esto es $48kHz$.

- Modo de Comunicación con el Periférico McASP (registros 8, 9 y 10). Se configura el codec como esclavo, además la transferencia se realizará bajo el protocolo I2S con palabras de 16 bits. No se le agrega offset a los datos.

- Conexión y Ganancia de los Canales. Se opta por utilizar los canales izquierda y derecha de la entrada *line_in* para realizar las medidas, y el canal *line_out* como salida para ser utilizado de DDS. Los registros que realizan las conexiones internas en el codec, así como también, los que configuran ganancias de las etapas entrada o salida son: 15, 16, 19, 22, 27, 30, 37, 38, 43, 44, 51, 58, 64, 65, 72, 82, 86, 92 y 93.

- Configuración del Reloj del Sistema (registro 102)

Recapitulando lo visto, el codec de audio incluido en la placa de evaluación, es capaz de producir señales analógicas de hasta $24kHz$, en principio suficientes para controlar la fuente de corriente que inyecta la señal testigo. Cuenta con un canal de entrada mono titulado *mic_in* y otro estéreo llamado *line_in* con los que se puede adquirir las señales proveniente de los electrodos, optandose por grabar a través del canal estéreo, ya que permite realizar la adquisición más rápidamente, tomando de a dos valores por lectura. La ganancia de los amplificadores a la entrada es variable y puede ser configurada mediante un registro, permitiendo modificaciones de forma dinámica que ayudan a la adaptación de las señales a adquirir. Además, el codec con la configuración implementada, adquiere datos con 16 bits de largo, lo que resulta en un rango de entrada de 65536 niveles.

3.4. Multichannel Audio Serial Port *McASP*

El Multichannel Audio Serial Port (*McASP*), es un periférico de propósito general que funciona como puerto serial, óptimo para las aplicaciones de multicanal de audio, realizando operaciones de transmisión y recepción de tramas de datos a través de hasta 16 serializadores (canales) que pueden ser habilitados individualmente, y operar de forma sincronizada o completamente independiente con distintos master clocks, usando incluso diferentes formatos de trasmisión/recepción. Es de gran utilidad para flujos por TDM (*time-division multiplexed*), protocolos Inter-IC Sound (*I2S*) e interfaz de trasmisión audio digital (*DIT*) entre componentes [23] [17].

A continuación se muestra una descripción general del funcionamiento del McASP, sus componentes principales, registros de configuración y control utilizados en el programa a implementar. Este capítulo no pretende ser una descripción en profundidad del hardware, ya que el capítulo 26 de [23] se dedica a eso.

Entre las características más destacables del McASP se encuentran las siguientes:

- Módulos independientes de trasmisión y recepción, con generadores programables de reloj independientes. Esta flexibilidad permite por ejemplo, recibir datos a $48kHz$ y trasmitir a $96kHz$.
- Hasta 16 pines asignables de dato serial.
- Conexión directa con conversores analógico-digital y digital-analógico, códec de audio, interfaz de recepción de audio digital (DIR), y otros.
- Amplia variedad de Inter-IC Sound (I2S) y formatos similares de bit-stream.
- Control de errores y recuperación:

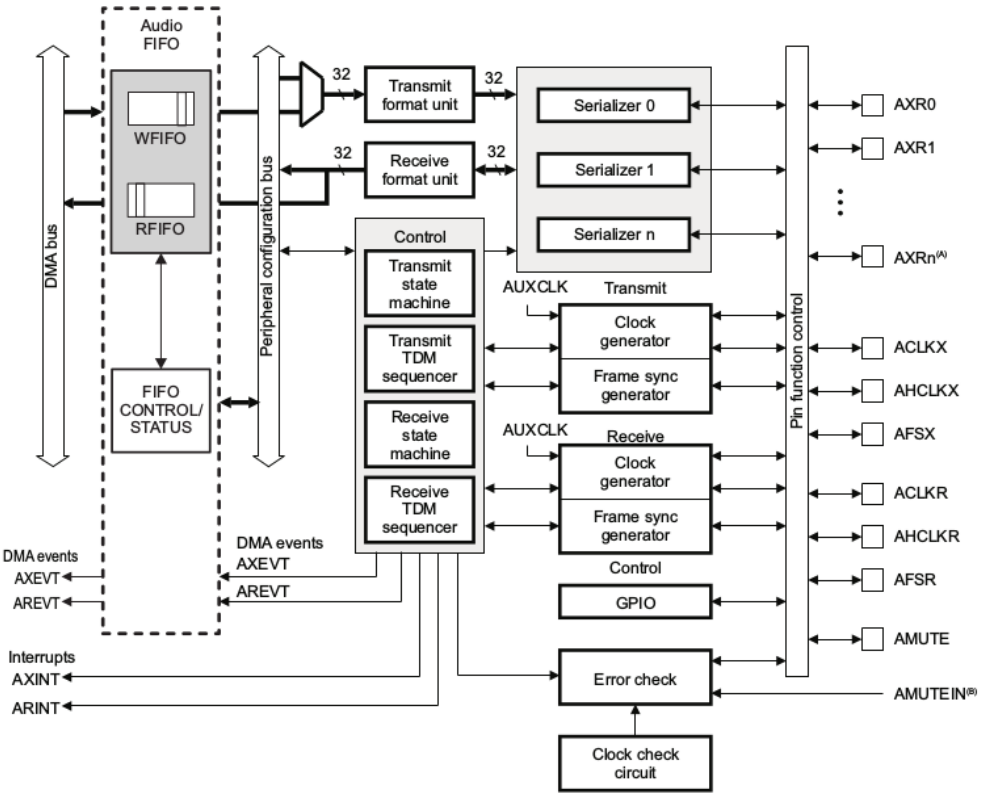
  Transmitir insuficiencia de datos y excesos del receptor, debido por ejemplo, a que el sistema no cumple con requisitos de tiempo.

  Sincronismo de trama tardía o temprana en modo TDM.

  Reloj de alta frecuencia fuera de rango, tanto para trasmisión como recepción.

En la figura 3.5 se observan los siguientes pines del puerto *McASP*:

```
AXR[n]--hasta 16 pines de dato
AHCLKX -- master clock de alta frecuencia de trasmisión
ACLKX -- bit clock de trasmisión.
AFSX -- señal de sincronismo de trama de trasmisión
AHCLKR -- master clock de alta frecuencia de recepcion
ACLKR -- bit clock de recepcion
AFSR -- señal de sincronismo de trama de recepción
AMUTE -- mute output.
```



A McASP0 has up to 16 serial data pins, $n = 15$; McASP1 has up to 12 serial data pins, $n = 11$; McASP2 has up to 4 serial data pins, $n = 3$.

B One of the DSP's external pins, see your device-specific data manual.

Figura 3.5: Diagrama de bloques del puerto *McASP*. [23]

Funcionamiento General

Cada *serializer* consta de un registro de corrimiento (*shift register* llamado *XRSR*), un buffer de datos (*XRBUF*), un registro de control (*SRCTL*), y la lógica necesaria para la alineación de las diferentes opciones disponibles. Para la recepción por ejemplo, los datos ingresan a los *serializer* provenientes del codec de audio por una serie de pines llamados *AXR[n]*, seteables (según los que se necesiten) mediante los registros de control *SRCTL[n]* correspondiente a cada pin. Luego, ingresan al registro *XRSR* que se va completando a medida que se reciben estos datos. Cuando se tiene el dato completo, se guarda en el registro *RBUF* para finalmente pasar por tres etapas dentro de la *Unidad de Formato*. La figura 3.6 muestra el esquemático del flujo de los datos, los registros y buffers que se configuran y utilizan.

Entre los formatos de comunicación soportados por el McASP (TDM, I2S y DIT), sólo a fin de estudiar lo esencial para los propósitos de *IMPETOM-B*, se detalla el proceso TDM, multi-canales de transmisión y recepción en formato TDM sincrónico (*Time Division Multiplexed* sincrónico).

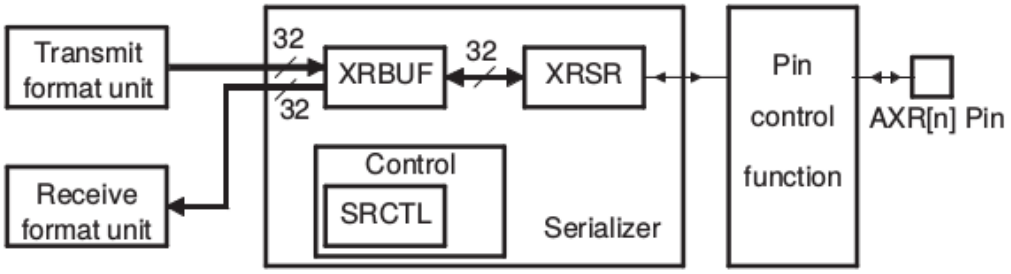


Figura 3.6: Diagrama descriptivo del flujo de bits en los *serializadores* del puerto *McASP*, tanto en recepción, como en transmisión. En recepción por ejemplo, los bits provenientes del codec de audio llegan mediante los pines *AXR[n]* al registro *XRSR*. Cuando este registro esta completo, el dato se pasa al registro *RBUF*. Por último los datos son procesados en la unidad de formato (3.8) antes de ser leídos por el DSP
. [17]

El formato de transmisión TDM consiste en tres componentes: el *clock*, los datos, y el *frame sync*. En una transferencia, todos los bits de datos (AXR[n]) están sincronizados con el clock serial (ACLKX or ACLKR) y agrupados en palabras, que a su vez se agrupan luego en slots (cada uno de estos slots hacen referencia a canales). De igual modo, una frame consiste en múltiples slots (o canales), y cada una de estas frames son definidas por la señal *frame sync* (ACLKX or ACLKR) (figura 3.7).

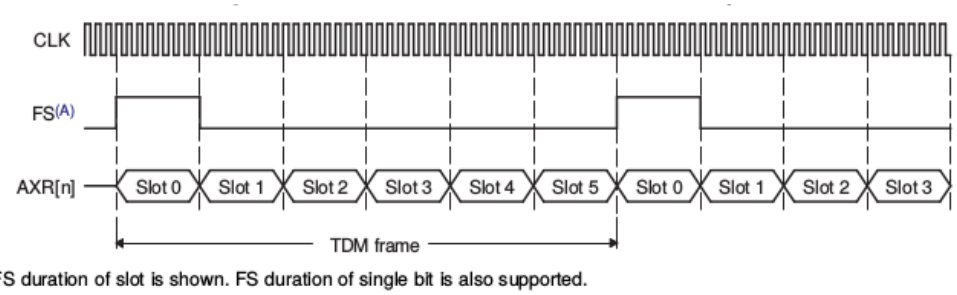


Figura 3.7: Transmisión de datos en formato *TDM sincrónico* y 6 canales o slots. Cada slot soporta una palabra de datos (hasta 16 bits), que se agrupan a su vez en *TDM frames* según la cantidad de canales disponibles. [17]

El McASP puede manejar hasta dos unidades de formato simultáneamente, una para transmitir y una para recibir, que pueden incluso variar entre si. Estas unidades de formato seteables, reasignan automáticamente los bits de datos dentro de cada palabras al formato natural del DSP (representación Q13), y al formato requerido por los dispositivos exteriores (formato I2S por ejemlpo).

- Mascara de bit y bits de relleno.
- Alineación de datos dentro de la palabra.
- MSB o LSB first.

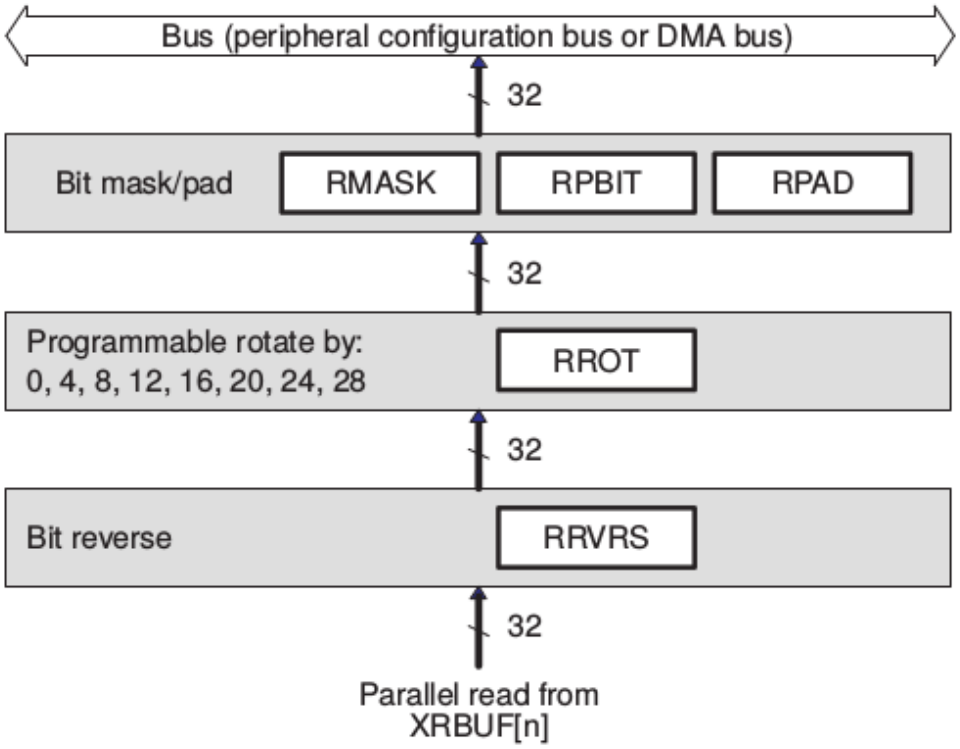


Figura 3.8: Diagrama de flujo, descriptivo del procesamiento de palabra de datos según la unidad de formato en recepción. [17]

Así entonces, el dato queda listo para ser leído por el DSP en el formato correcto. La transmisión se realiza de manera inversa, el dato pasa primero por la unidad de formato y luego es desplazado hacia afuera mediante los *serializer*.

Capítulo 4

Circuito Auxiliar

Cuando se dice circuitería auxiliar, se refiere a la circuitería complementaria necesaria para alcanzar los propósitos de *IMPETOM-B*, o sea, fuente de corriente, conexión y multiplexación de electrodos y circuito de prueba *RC* que simula característica de la impedancia del cuerpo humano (fantoma).

4.1. Fuente de Corriente

IMPETOM-B hace uso de técnicas de medición de propiedades eléctricas de los materiales mediante estímulos de corriente, y por eso es necesario contar con una fuente de corriente que tenga una alta impedancia de salida y que presente una apropiada respuesta en frecuencia junto con una estabilidad ante posibles oscilaciones.

Una de las opciones mas utilizadas son las fuentes de corriente controladas por voltaje (FCCV, señal de entrada corresponde a voltaje, y la señal de salida es una corriente). Las FCCV pueden clasificarse en dos categorías: aquellas qué tienen una alta impedancia de salida debido a las propiedades físicas de un dispositivo, y las que tienen una alta impedancia de salida debido a un sistema realimentado en lazo cerrado.

Producto de que la construcción de una fuente de corriente que cumpla con todas las características deseadas para la implementación de EIT excede los alcances de este proyecto, y es por si solo un proyecto independiente, en principio se apunta a recorrer el camino más simple y que cumple los mínimos requerimientos según la documentación consultada [14] [11]. En este sentido, la primer implementación de fuente de corriente es el modelo *Howland*.

4.1.1. Fuente de corriente tipo Howland

Existen varias topologías de fuentes de corriente tales como: la fuente de corriente tipo Howland, amplificador inversor de transconductancia, amplificador no inversor de transconductancia, fuente de corriente basada en espejo de corriente, etc. Si bien, las fuentes que presentan un mejor desempeño de todas las anteriores son la fuente Howland y el espejo de corriente, quien muestra una ventaja constructiva sustancial, es la fuente Howland, ya que utiliza sólo un amplificador operacional.

La fuente de corriente Howland usa un amplificador operacional que utiliza la realimentación inversora y no inversora (también conocidas como negativa y positiva respectivamente) para obtener una corriente independiente de la carga, en este caso, de la impedancia corporal. La siguiente figura 4.1 muestra el esquemático de la fuente en cuestión.

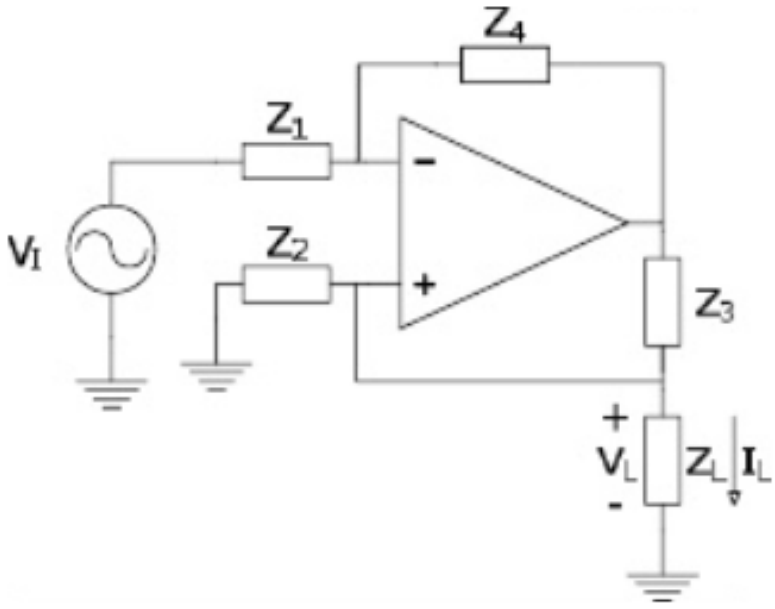


Figura 4.1: Esquemático de fuente de corriente tipo Howland. [10]

La corriente de salida I_L, en función de la tensión de entrada V_{in} es:

$$I_L = V_+(\frac{Z_4}{Z_1 Z_3} - \frac{1}{Z_2}) - V_{in}(\frac{Z_4}{Z_1 Z_3}) \tag{4.1}$$

Donde se distingue la componente de offset:

$$V_+(\frac{Z_4}{Z_1 Z_3} - \frac{1}{Z_2}) \tag{4.2}$$

Mas tarde se mostrará como este punto representa una importante limitante a la hora de la precisión deseada en las medidas.

Si se equilibran las resistencias haciendo que:

$$\frac{Z_C}{Z_B} = \frac{Z_D}{Z_A} \tag{4.3}$$

Se logra que la corriente de la fuente *Howland* queda independiente de la impedancia de la carga, la resistencia de salida tiene el valor infinito y el offset generado queda en 0.

$$I_L = \frac{-1}{Z_2} V_{in} \tag{4.4}$$

$$I_{offset} = V_+ (\frac{Z_4}{Z_1 Z_3} - \frac{1}{Z_2}) = 0 \tag{4.5}$$

$$Z_{salida} = \frac{V_+}{I_L} = (\frac{Z_4}{Z_1 Z_3} - \frac{1}{Z_2})^{-1} \tag{4.6}$$

Por supuesto, toda la deducción anterior es mera teoría. En la práctica, el circuito de la fuente de corriente *Howland*, proporciona una impedancia de salida infinita y un offset de 0, si y sólo sí, las impedancias indicadas están bien equilibradas. Por tanto, se depende fuertemente de la precisión de los valores de éstas, siendo imposible lograr un apareamiento perfecto.

Además de todo lo anterior, la realidad marca que el circuito funciona bien para corrientes DC, viendo que los problemas aparecen a frecuencias por encima de esta, de algunos KHz, justo en el rango de trabajo necesario. En este punto, las capacidades parásitas de las resistencias y capacidades de fuga, empiezan a tener preponderancia y la impedancia de salida del circuito se degrada a medida que la frecuencia de la señal de corriente aumenta.
Acompañando los parámetros anteriores, existen otros que marcan la robustez del sistema. Dado que estos exceden el alcance de este proyecto no se estudiarán en este documento, aunque se enumeran a continuación:

- La ganancia de la fuente
- El ancho de banda del amplificador operacional
- Slew Rate (SR)
- Corriente de Polarización de Entrada (I_{bias})

Los puntos anteriores son de singular importancia a la hora de percibir los efectos del uso sobre los pacientes. Recuerde que *IMPETOM-B* pretende ser usado en pacientes de CTI de manera frecuente, por lo que debe cumplir con parámetros de seguridad en todo momento.

De todas formas, se decide probar además una segunda opción, implementando una fuente de corriente más inmune a los desapariamientos de resistencia y al ruido.

4.1.2. Fuente de corriente con amplificador *AD844*

"El amplificador operacional con realimentación de corriente (current-feedback amplifier: CFA) AD844 de Analog Devices, puede ser usado como una fuente de corriente controlada por voltaje. Este integrado combina gran ancho de banda y muy alto slew-rate con una excelente performance de continua. Su arquitectura de realimentación de corriente resulta en una muy buena performance en AC, alta linealidad y una respuesta al impulso excepcionalmente limpia. El voltaje de offset y las corrientes de bias en las entradas son ajustadas mediante láser (laser trimmed) durante la construcción del integrado, para minimizar los errores de continua" [10].

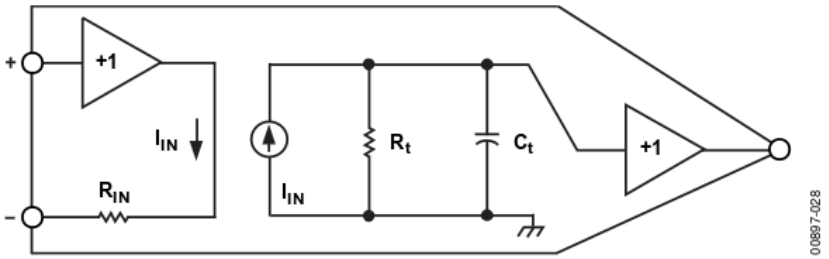


Figura 4.2: Esquemático amplificador AD844. [1]

Desde el principio se hacen notar las diferencias con la fuente *Howland*, mientras que en esta última se tiene un pobre desempeño para frecuencias por encima de la DC y valores de offset importantes, la nueva implementación menciona: *"gran ancho de banda y muy alto slew-rate con una excelente performance de continua. Su arquitectura de realimentación de corriente resulta en una muy buena performance en AC"*.

Yendo al detalle constructivo del amplificador, el mismo cuenta con un nodo interno accesible y de alta impedancia (Z en la figura 4.3) para la compensación externa de frecuencia. El voltaje aplicado a la entrada no inversora de alta impedancia, es copiado hacia la entrada inversora de baja impedancia, mientras que la corriente de salida del nodo Z (I), es reproducida por los espejos de corriente.

$$I = \frac{V_i}{R} \tag{4.7}$$

"El uso de este circuito integrado soluciona los problemas originados en la asimetría de los espejos y la inestabilidad térmica, pero aparecen otros errores debido a la precisión en la ganancia de la primer etapa y su resistencia de salida no nula R_{ob}" [10]. [1]

La siguiente figura 4.4 muestra un modelo de las fuentes de error en continua y de ruido para el *AD844*. La corriente de bias de la entrada inversora I_{BN}, circula a través de la resistencia de realimentación R_1, mientras que la I_{BP} de la entrada no

[1] De acuerdo al fabricante, el voltaje de la entrada no inversora debe mantenerse entre –1V y +1V para mejores resultados [1].

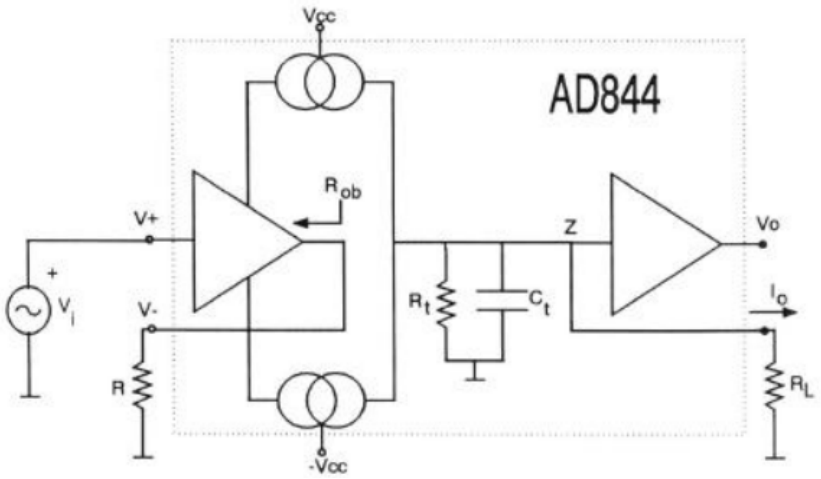


Figura 4.3: Esquemático de Fuente de corriente con amplificador AD844. [10]

inversora, circula por la resistencia R_P. El voltaje resultante (sumado al voltaje de offset:V_{OS}), aparece en la entrada inversora, teniendo un error total de continua, V_0 en la salida según la siguiente ecuación:

$$V_O = (I_{BP}R_P + V_{OS} + I_{BN}R_{in})(1 + \frac{R_1}{R_2})I_{BN}R_{in} \tag{4.8}$$

Se puede apreciar que el voltaje DC, es proporcional a la resistencia de realimentación R_1. Las fuentes de ruido pueden ser modeladas en una forma similar a las corrientes de bias DC, donde cada fuente de ruido es I_{nn} , I_{np} , V_n , y el ruido inducido en la salida V_{ON} resulta en:

$$V_{ON}(rms) = \sqrt{((I_{np}R_P)^2 + V_n^2 + (1 + \frac{R_1}{R_2})^2(I_{nn}R_1)} \tag{4.9}$$

Resulta entonces, que es posible mantener los valores de voltaje de offset y de las corrientes de bias dentro de rangos aceptables, ajustando debidamente los valores de los parámetros I_{np}, I_{nn}, R_P, V_n, R_1, R_2, etc (recuerde que estos son ajustadas mediante láser (laser trimmed) durante la construcción del integrado).

4.2. Fantoma de Pruebas

El propósito del fantoma es simular las características de la impedancia del cuerpo humano. Para esto, recordando lo visto en el capítulo *Estado del Arte*2, donde se indica que la impedancia del cuerpo humano puede modelarse en un circuito eléctrico como una suma de tres impedancias en serie: impedancia de la piel en la zona de entrada, la impedancia interna del cuerpo y la impedancia de la piel en la zona de salida. Estas impedancia a su vez, se representan como un capacitor a la entrada y a la salida (contacto del electrodo con la piel) y una impedancia compuesta por una resistencia y una capacitancia por los distintos tejidos del interior del tórax. En resumen, se tiene un circuito *RC*. 2.2

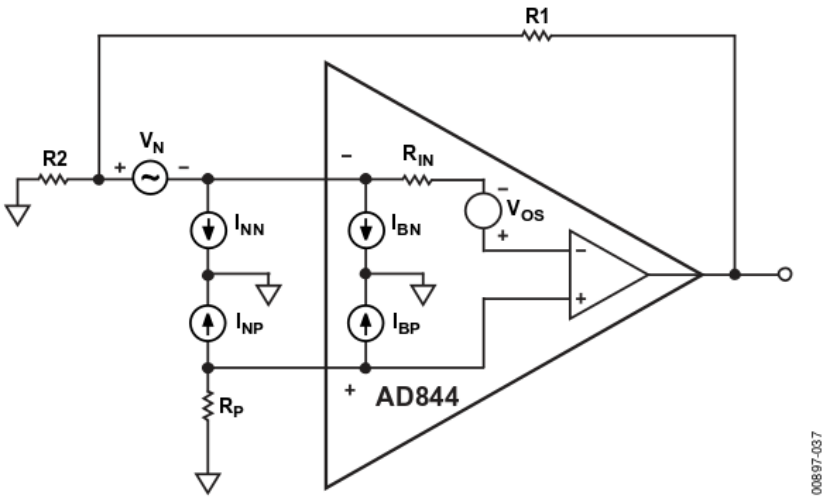


Figura 4.4: Esquemático de modelo de fuentes de error en continua y ruido para el AD844. [10]

Detrás de la esencia de lo que es en realidad el fantoma, su propósito va más allá y es facilitar el relevamiento que permita verificar qué tan fehacientes son las medidas de voltaje obtenidas. La medida de voltaje debe ser lo mas exacta posible, esto quiere decir, que se debe obtener el valor exacto del módulo y defasaje entre la señal de entrada y salida. En este sentido, el fantoma simula características controlables (valores variables de R y C), para desde allí, deducir las características del voltaje teórico medido. Desde el punto de vista matemático, a la entrada analógica debe llegar una señal del estilo

$$V_{in} = A\rho cos(w_c t + \theta) \tag{4.10}$$

donde A es la ganancia de la señal emitida por la salida analógica del codec de audio (impuesta como señal de sincronismo como se verá mas adelante 5), y ρ el módulo de la impedancia total del cuerpo humano. Sabiendo el valor de R y C configurados, se deduce si el valor del voltaje y defasaje obtenidos siguen el comportamiento esperado, y permite realizar diferentes pruebas para dictaminar que tan precisa es la medida en cada una de las situaciones propuestas (frecuencia de corriente inyectada, resistencia del cuerpo humano, capacidad del circuito humano).

4.3. Circuito de Conexión y Multiplexado de Electrodos

Para entrar en detalle, debe regresar al punto donde se describe el funcionamiento general de *IMPETOM*2. Recuerde que para cada medida se tienen dos pares de electrodos, uno para la inyección de corriente en la región (par de corriente), y otro para medir el voltaje en la frontera (par de voltaje), configuración denominada tetrapolar o de 4 electrodos. En total, la conexión al paciente se realiza a través de 16 electrodos, por los cuales se inyecta la señal de corriente por un

par adyacente y se miden las diferencias de voltaje resultantes en los 14 electrodos restantes (13 medidas) 2.8, repitiendo el proceso hasta completar la inyección de corriente para todos los pares de electrodos.

Para realizar el control del circuito multiplexador, se cuenta con puertos de comunicación serie (UART o USB), un puerto paralelo (*EMIF-A*), y puertos de comunicación *RJ-45*. Debido a la dificultad y poco tiempo disponible, así como también que se tenía un conocimiento previo, se optó por realizar el control mediante el puerto UART.

A partir de esto, se debió diseñar un circuito secuencial que responda a un solo pulsos para cambiar los electrodos de medida e inyección de corriente. Dado que el recorrido de los electrodos es efectivamente secuencias y en un mismo sentido, tomando las precauciones del caso es posible contemplar todas las variantes del multiplexado. El circuito cuenta con 2 registros de tipo contador de 4 bits (16 posiciones), uno para los electrodos inyectores de corriente y otro para los electrodos medidores de voltaje. Tenga presente que en ciertas ocasiones, los electrodos medidores de voltaje e inyectores de corriente pueden ser lo mismos, pero tomando esta precaución es que realiza el conexionado.

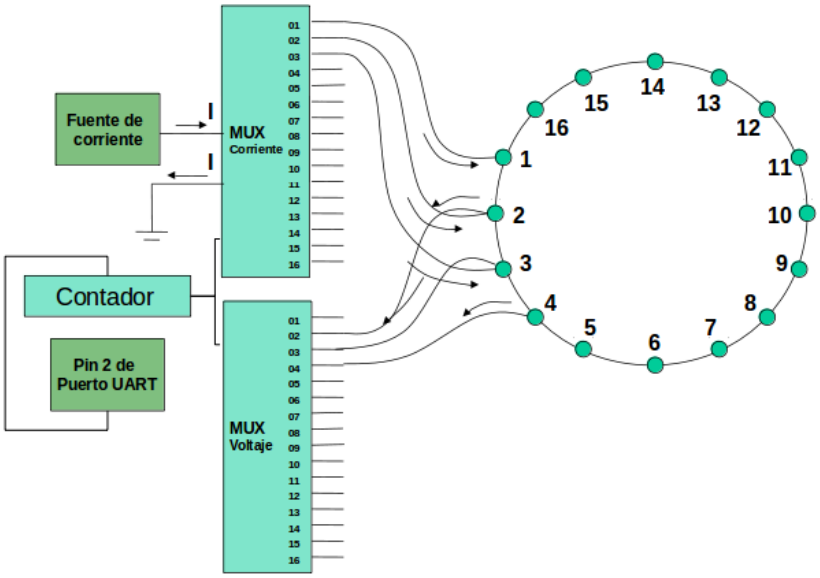


Figura 4.5: Esquemático de conexionado de multiplexación. El circuito cuenta con 2 registros de tipo contador de 4 bits (16 posiciones), para el control de los MUX de los electrodos inyectores de corriente y el MUX de los electrodos medidores de voltaje.

No obstante, este prototipo de circuitos de multiplexado que implementaran una comunicación serie mediante puerto UART, queda al margen del proyecto producto de su complejidad.

Esta página ha sido intencionalmente dejada en blanco.

Capítulo 5

Tratamiento de Datos

El principal objetivo de la precisa y correcta medición de voltajes, es poder enviar información suficiente como para obtener una reconstrucción, y por ende, una imagen, bien detallada y fehaciente, que permita un correcto diagnóstico.

El método de reconstrucción de imágenes utilizado por el sistema EIT *IMPETOM*, se basa en percibir los cambios en la resistividad, derivados a su vez, de cambio en los voltajes de los electrodos. Se puede decir que cada uno de estos voltajes es en sí, una condición de borde de un problema condicionado que representa la impedancia o el potencial en cierto camino de corriente. Es por ello que se pasa al sistema de reconstrucción de imágenes, una matriz de voltajes. De todas maneras, compartir únicamente las medidas de voltajes de los electrodos no da ninguna información de la verdadera impedancia corporal, ni siquiera algo representativo. Es oportuno recordar que a las frecuencias que se quiere trabajar (menores a $50KHz$), la señal de corriente no es inmune a la composición de los diferentes tejidos y fluidos, por lo que se introduce un desfasaje en la señal medida respecto a la inyectada, y la medida del módulo del voltaje por si sola, no cuenta con esta información de fase. Se desprende entonces, que es necesario un procesamiento extra de la señal obtenida, y por ende, de los datos recolectados antes de entregarlos en el formato especificado. En lo que sigue, se realiza el estudio que revele la información buscada.

5.1. Análisis Teórico

Por el contrario de lo que se pude pensar, la reconstrucción de imágenes a partir de cambios en la resistividad, derivados de un cambio en los voltajes, torna al problema mejor condicionado. Esto es debido a muchos de los errores sistemáticos (tales como defasaje introducido por el muestreo, no idealidad de fuente de corriente, ganancias de los distintos canales, etc) se cancelan parcialmente a partir de la diferencia de dos conjuntos de voltajes.
Las técnicas mas utilizadas para la reconstrucción de las imágenes, están determinadas por operadores lineales invariables, que pueden representarse como una

multiplicación de un vector de medidas de voltaje por una matriz que representa el operador de la reconstrucción [13].

Simplemente a modo de poner en conocimiento al lector del desarrollo realizado hasta ahora en el tema, cabe destacar que la implementación del sistema de reconstrucción de imágenes *IMPETOM-I* [8]), implementa tres métodos de reconstrucción:

- Retroproyección
- Matriz de Sensibilidad
- Newton-Raphson

Los dos primeros son técnicas de un solo paso, es decir, reconstrucción basada en la linealización del problema inverso obteniendo una solución que corresponde a la variación de conductividades respecto a una distribución de referencia. A pesar que estos métodos de un solo paso generan una imagen utilizando una única operación matemática, y por consiguiente son métodos de reconstrucción rápidos (para generar imágenes dinámicas), la simplicidad de la formulación, hace que la búsqueda de los coeficientes de la matriz que representa el operador no sea fácil. En cambio, el método de Newton-Raphson es iterativo. Utiliza operaciones repetitivas que en cada paso mejoran la imagen. Intenta una reconstrucción de la distribución de conductividades absoluta teniendo en cuenta la ausencia de linealidad del problema [13].

De lo adelantado en la introducción, los valores obtenidos de los electrodos representan a groso modo la impedancia que se quiere medir, pero no cuenta con la información necesaria para identificarla completamente. Para el caso del proyecto *IMPETOM-B*, interesa particularmente obtener tan sólo la parte real de la impedancia corporal puesto que es información suficiente para obtener una buena reconstrucción [11] [12].

Intuitivamente surge la interrogante: ¿Cómo obtener la parte real de una impedancia medida simplemente de una señal sinusoidal recibida? Respondiendo a esto se construye el siguiente análisis:

Suponga se tiene la impedancia corporal, compuesta de una parte resistiva y otra capacitiva, que se puede modelar como una impedancia Z de módulo ρ_z y ángulo θ.

Como se mencionó en los primeros capítulos, la manera de revelar la información de esta impedancia es inyectando una señal de corriente alterna I, de valor constante y frecuencia conocida, caracterizada por la ecuación

$$I = Acos(w_c t) \tag{5.1}$$

Cuando dicha corriente atraviesa la impedancia corporal Z, resulta una señal de voltaje cuya expresión es similar a la siguiente:

$$V = \rho_z A cos(w_c t + \theta) \tag{5.2}$$

Este voltaje tiene incorporado la información del módulo y fase de la impedancia, implícita en amplitud y defasaje respecto a la señal original. Para obtener sólo las componentes que nos interesan, se recurre a las propiedades de los receptores sincrónicos y por consiguiente, la multiplicación de los cosenos. Entonces se multiplica el voltaje recibido de los electrodos por la señal de un oscilador local

$$Vosc = Bcos(w_c t) \tag{5.3}$$

llegando a la expresión:

$$V.Vosc = \rho_z A cos(w_c t + \theta) B cos(w_c t) \tag{5.4}$$

Aprovechando algunas propiedades trigonométricas, la ecuación anterior resulta en:

$$V.Vosc = \rho_z \frac{A \times B}{2} [cos(\theta) + cos(2w_c t + \theta)] \tag{5.5}$$

Como se puede ver, la nueva expresión de la señal tiene dos componentes bien distintas. La primera es un valor constante correspondiente a una señal de continua, que es a su vez, proporcional al módulo de la impedancia, más precisamente, la proyección de esta sobre el eje real $\rho_z cos(\theta)$ a menos de una constante $\frac{A \times B}{2}$. Dicho de otra manera, este valor es en realidad la parte real de la impedancia Z a menos de una constante. El segundo fragmento de la expresión representa una componente de alterna a alta frecuencia (el doble de la original). Esta última componente puede ser eliminada fácilmente si se pasa la señal por un filtro LPF de frecuencia de corte menor que $2w_c$.

En el caso de *IMPETOM-B* se implementó el filtro *IIR* (de Respuesta al Impuso Infinito). Como su nombre lo indica, se trata de un tipo de filtro digital en el que si la entrada x es una señal impulso, la salida y tendrá un número infinito de términos no nulos, es decir, nunca vuelve al reposo. Además, la salida depende de las entradas actuales, pasadas, y de las salidas en instantes anteriores (esto último se logra a través del uso de realimentación de la salida).

$$y_n = b_0 x_n + b_1 x_{n-1} + \cdots + b_N x_{n-N} - a_1 y_{n-1} - a_2 y_{n-2} - \cdots - a_M y_{n-M} \tag{5.6}$$

Donde los a y b son los coeficientes del filtro, y el orden, el máximo entre los valores de M y N, siendo M y N términos que determinan la cantidad de polos y ceros en la función de transferencia.

Esta implementación característica del filtro IIR, le permite tener varias ventajas sobres otros diseños, entre ellas, ir procesando los datos a medida que van llegando (no es necesario llenar una variable o buffer para luego procesar los datos), si los datos recolectados son suficientes, el valor de salida del filtro tiende al valor correspondiente sin importar que la entrada vuelva a ser nula (respuesta al impulso infinita), y pueden cumplir las mismas exigencias que otros filtros pero con menos orden de filtro. Estas tres características representan una ventaja inigualable a la hora de implementar el filtro, pues representa una menor carga computacional [9].

En líneas generales, el sistema debe implementar una transferencia del estilo siguiente: 5.1

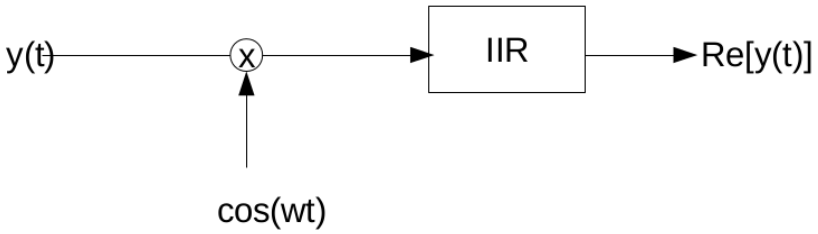


Figura 5.1: Diagramas de bloques del sistema implementado en software una vez ingresan los datos y(t).

El resultado final de estas multiplicaciones y el filtrado, es efectivamente sólo el valor de la componente real de la impedancia medida de los electrodos, a menos de una ganancia o atenuación:

$$V.Vosc_{filtrada} = \rho_z cos(\theta)\frac{A \times B}{2} \tag{5.7}$$

De esta manera, la medida de voltaje obtenida representa la parte real de la impedancia medida entre los electrodos, lo que permite dar más información para obtener una buena reconstrucción.

5.2. Digitalización

Como se mencionó en la descripción de la *EVM Evaluation Board* (capítulo *EMOmap-L137*), la misma cuenta con una frecuencia de muestreo máxima de ($48KHz$) para cada canal (izquierdo y derecho por ser estero). Según el teorema de muestreo, para poder reconstruir la señal correctamente con la f_s indicada, se deben tener señales de hasta $24KHz$ como máximo, lo que ya representa una limitación bien importante.

Si recuerda lo visto en capítulos anteriores, donde se mostraba el proceso matemático que llevaba a la obtención de la parte real de la impedancia corporal, se

vio que la señal de voltaje percibida en los electrodos debe ser multiplicada por una señal de similares características, o sea, una sinusoide de igual frecuencia y fase. Si la señal de entrada a los electrodos es de la forma $K_1 cos(w_c t + \phi)$ y la señal de sincronismo es $K_2 cos(w_c t + \theta)$, naturalmente de cuentas se llega a:

$$V_{electrodo}.V_{sincronismo} = K_1 cos(w_c t + \phi) K_2 cos(w_c t + \theta) \tag{5.8}$$

El resultado de esta multiplicación (aplicando otras entidades matemáticas y trigonometría) es:

$$V_{electrodo}.V_{sincronismo} = K_1 K_2 [\frac{cos(\theta - \phi) + cos(2w_c t + \theta + \phi)}{2}] \tag{5.9}$$

En este punto se tiene una señal con componentes de frecuencia $2f_c$. De no tener las precauciones ya mencionadas, puede que todavía se este por encima de la frecuencia límite si la señal de control emitida estuviese encima de los $12KHz$ ($2 \times 12KHz$=$24KHz$). De todas maneras, la implementación de la rutina de adquisición y procesamiento de muestras, no necesita de la reconstrucción de la señal en ningún momento, ya que trabaja sólo con la tabla de muestras adquiridas. Esto permite alcanzar idealmente los $24KHz$ aunque se establece una frecuencia de trabajo de $16KHz$.
Las complicaciones con la señal inyectada y capturada no terminan con lo anterior. Debido a que no se dispone de una señal de reloj analógica interna de acceso fácil y menos aún, controlable digitalmente desde alguno de los dispositivos de la placa, se debe disponer de una representación digital de la señal, en este caso de la sinusoide. Esto quiere decir que en realidad se dispone de una tabla de valores que representan puntos de la sinusoide, y que luego el DAC se encarga de unir de forma adecuada.

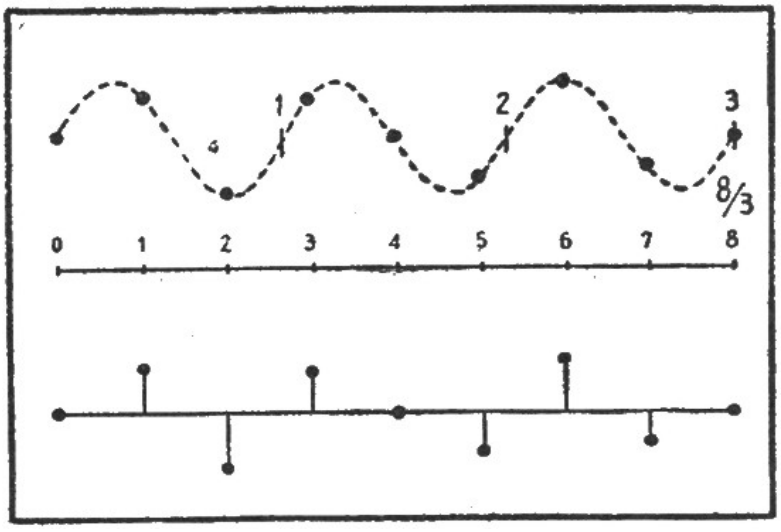


Figura 5.2: Representación Digital de Sinusoide

Puesto que el DSP trabaja a una velocidad varios ordenes mayor a lo que lo hacen los diferentes periféricos, como por ejemplo DACs y codec de audio, la limitante en la velocidad con que se fijan los valores de una muestra en la salida analógica correspondiente, proviene de estos últimos elementos. Ya se mencionó que la frecuencia de muestreo es $f_s = 48KHz$, entonces los diferentes periféricos de la placa pueden poner como máximo 48 muestras por milisegundo. En otras palabras, esto hace que si se desean señales de frecuencia de $1KHz$, se necesitarán 48 valores (se muestrea 48 veces en 1 milisegundo, $\frac{48}{1ms} = 48KHz = f_s$), para representar señales de $2KHz$, se necesitan entonces 48 valores que representen 2 períodos de la señal (se muestrea 24 veces en 0,5 milisegundos), y así sucesivamente, hasta llegar a representar 48 períodos de reloj en 1 milisegundo, o lo que es lo mismo $\frac{1}{(1/48)ms} = 48KHz = f_s$, lo que implicaría representar un período mediante una muestra, transformándose en un problema a la hora de reconstruir la señal. Nuevamente llegamos a los límites indicados en el teorema de muestreo, mostrando que se debe tener al menos dos muestras por período $\frac{2}{(1/24)ms} = 48KHz = f_s$ para que la señal sea representable, o sea, una f_c máxima de $24KHz$.

Luego de obtenidos las muestras que representen fehacientemente la señal, es hora de recuperar la información del módulo y fase de la impedancia, o en este caso, la parte real de la impedancia. Para obtenerla, se recurre a las propiedades de los receptores sincrónicos y por consiguiente, la multiplicación de los cosenos como se mencionó al principio de este capítulo. Para hacerlo posible se multiplican las muestras de las señales recibidas de los electrodos con la señal del oscilador local que describe una sinusoide de la forma 5.3, generando así nuevos valores de muestras que describen la señal de la siguiente forma:

$$x_n = Kcos(w_c t_n + \theta)cos(w_c t_n) \tag{5.10}$$

Debiendo ser K una representación proporcional de $\rho_z AB$. El resultado anterior conserva las mismas características descriptas en la teoría de la multiplicación de cosenos, por lo que también vale:

$$x_n = Kcos(w_c t_n + \theta)cos(w_c t_n) = \frac{K}{2}[cos(\theta) + cos(2w_c t_n + \theta)] \tag{5.11}$$

Contando ya con la anterior representación digital de la señal, el segundo paso es recuperar la información que nos interesa, haciéndola pasar por el filtro IIR, al igual que lo indicado en el análisis teórico. El diagrama de implementación del filtro se muestra en la siguiente figura 5.3 y representa la ecuación teórica 5.6.

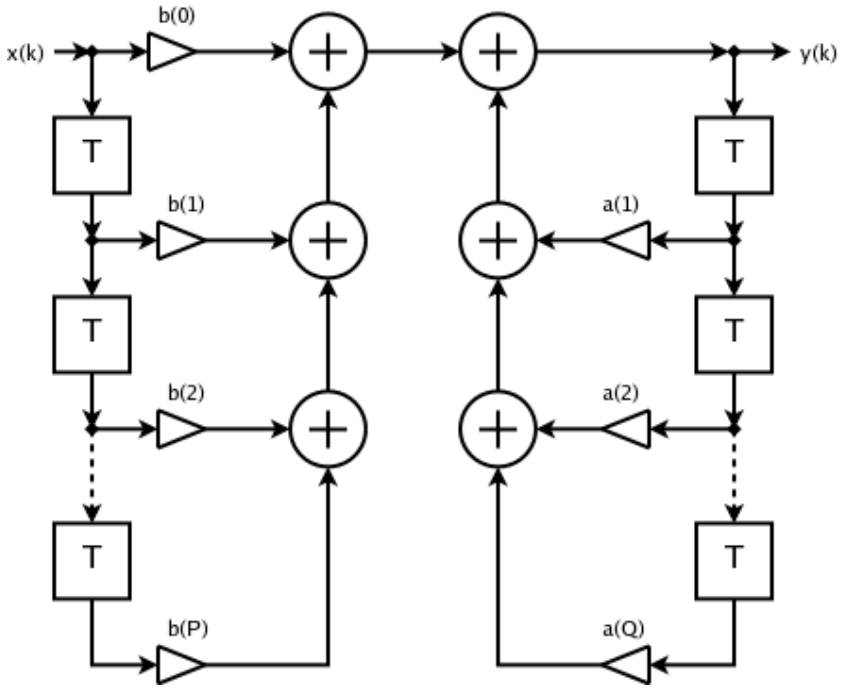


Figura 5.3: Estructura computacional del filtro IIR [9]

Finalmente, el resultado de la salida del filtro tiende al valor de la componente real de la impedancia, a menos de una constante 5.4.

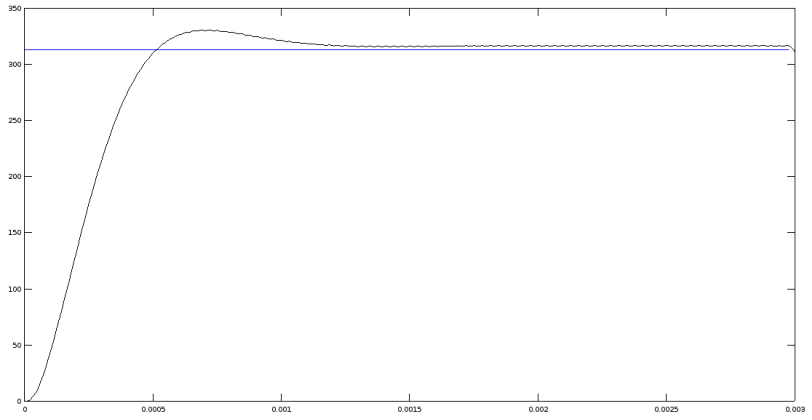


Figura 5.4: Representación de la salida del filtro de recepción IIR. Se observa como tiende a un valor constante, en teoría el valor de la parte real de la impedancia medida a menos de una constante.

Esta página ha sido intencionalmente dejada en blanco.

Capítulo 6

Software *IMPETOM-B*

Si el lector ha seguido cuidadosamente el correr de los capítulos, sabrá que ya se ha reparado en conceptos acerca de funcionamiento mínimo requerido, particularmente referidos al procesamientos de las señales dentro del núcleo DSP. La presente sección profundiza el estudio para obtener un software integral que logre el uso y la comunicación entre los núcleos DSP y GPP.

6.1. Conceptos Generales

Como ya se ha mencionado repetidas veces capítulos atrás, en líneas generales el problema se reduce a la medida reiterada de la impedancia corporal distribuida, excitada por la fuente de corriente entre dos electrodos, mientras se realiza la medida en otro juego de electrodos hasta completar las 16 medidas. Finalmente, se pasa al siguiente par de electrodos de inyección obteniendo un nuevo perfil para repetir todo el proceso. Una vez obtenidas las medidas de voltaje, se procesan los datos para obtener la información relevante y se envían a otro módulo para generar finalmente un archivo de texto plano (*archivo.txt* y formato IMPETOM).

Para lograr el máximo aprovechamiento de recursos que la placa puede brindar, es imprescindible de la utilización de forma simultanea de ambos núcleos, por tanto, cada núcleo debe encargarse de tareas para los cuales fueron concebidos. En este sentido, el desarrollo se encamina para que el encargado principal del procesamiento de las señales que provienen de los electrodos alrededor del tórax, el manejo y generación de la señal de control de la fuente de corriente, y el control digital sobre los multiplexores deba ser el DSP (*Digital Signal Processor*), mientras que el GPP (*General Purpouse Processor*) se encarga de iniciar y administrar los procesos que interactúan en todo el sistema además de la generación del archivo en texto plano **.txt*.

Definida las principales características de cada una de las partes involucradas, se puede continuar con la descripción del funcionamiento del software, de forma de mantener la estructura propia del sistema, realizándolo separadamente según el núcleo al que se aplica. Se describe entonces en primer lugar lo competente con

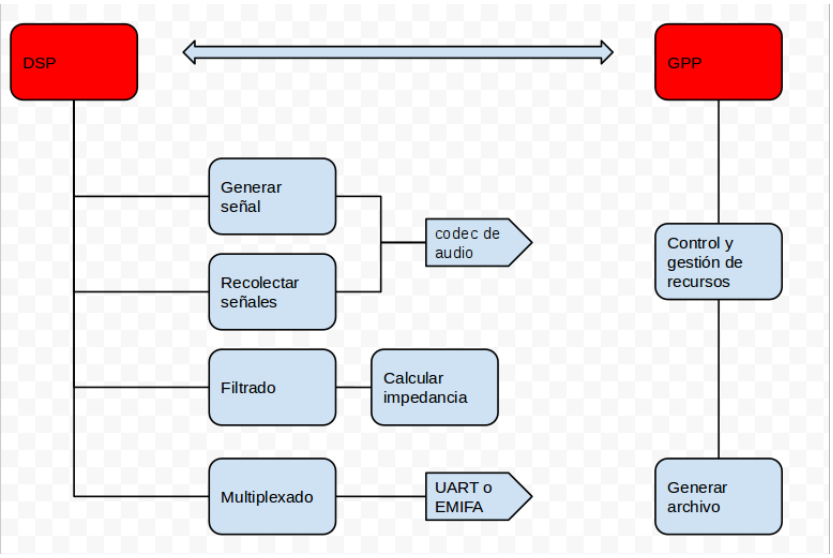


Figura 6.1: Diagrama de tareas en núcleos DSP y GPP.

el núcleo DSP para luego continuar con el GPP.

6.2. DSP *Digital Signal Processor*

Comprender el funcionamiento lógico que implementa el DSP no es tarea sencilla dado que no sólo debe implementar la generación y medida de señales, sino que también debe inicializar todos los periféricos que el DSP utiliza para la comunicación (librerías, entradas, salidas, Codec, etc).

Efectivamente, es esto último lo primero en hacer la rutina en DSP, crea instancias de los diferentes periféricos a utilizar (Codec de audio y McASP), los inicializa y setea de acuerdo con los objetivos particulares (ganancias, modo de transmisión, trama y bits comprendidos en la comunicación y sincronismo, puertos entrada y salida, etc). Una vez dispuesto todo para la correcta operación de los diferentes elementos que el DSP administra, se pasa a la secuencia de generación de señal de control para la fuente de corriente.

La señal de control para la fuente de corriente se crea a través una tabla con valores que toma una sinusoide, que luego es reconstruida utilizando las propiedades del DAC del codec de audio. Recuerde que la cantidad de valores de la tabla depende de la frecuencia con que se quiere trabajar5. Si la frecuencia de muestreo del codec está fija en $48KHz$ y en el caso de IMPETOM-B, se quiere trabajar con señales de $16KHz$ (frecuencia suficiente para que el paciente no perciba molestias), en principio, trabajando con 3 muestras por período resulta en estos $16KHz$ $(48/3 = 16)$. Recurriendo al teorema de muestreo, es posible reconstruir señales con frecuencias máximas de 24 KHz ($f_s > 2W$).

Especificada la señal de control a emitir, el siguiente paso es poner los datos a la salida analógica. La forma de realizar esto es utilizando *serializers*, los encargados del manejo de entrada y salida de datos del McASP 3.6.

Finalmente, la forma de comunicarse con el exterior es deslizar los bits de datos hacia dentro o afuera del sistema en el registro *RXBUF* correspondiente. En el caso del sistema para *IMPETOM-B*, el registro configurado como receptor es el *RBUF_0* mientras que el transmisor es el *XBUF_5*. Además se necesita esperar tanto a que el *Codec de Audio* esté pronto para recibir (momento en que se transmite), así como también a que se tengan datos disponibles a ser leídos en el canal (momento en que se recibe), los datos son enviados y recibidos en dos variables de la rutina.

Hasta aquí sólo se ha resuelto el problema de enviar una señal analógica para el control de la fuente de corriente, y recibir una señal, también analógica, proveniente de los electrodos, pero como se mencionó antes, la señal ya es recibida en una variable de programa, por lo que ya se está en condiciones de trabajar con ella.
En este punto es prudente comentar acerca del formato en que el McASP intercambia los datos con los periféricos, léase codec de audio, DACs y ADCs. En el capítulo dirigido al codec, se indica que soporta multi-canales de transmisión y recepción en formato TDM sincrónico (*Time Division Multiplexed* sincrónico). Esto quiere decir que los bits de datos son agrupados en palabras y luego en slots, donde cada uno de estos slots hacen referencia a canales 3.7.
Esto significa que a pesar de que la física de implementación del codec y de los ADCs permiten capturar muestras de manera instantáneas, son enviadas al núcleo DSP en instantes diferentes. *IMPETOM-B* se piensa de manera de utilizar dos canales, uno para recibir la señal desde los electrodos y otro para relevar la señal que controla la fuente de corriente, y que luego se utiliza como señal de referencia del oscilador local (ecuación 5.3). Entonces, las muestras de los dos canales son enviadas en slots consecutivos, generando un desfasaje de una muestra. Es imprescindible entonces tener este punto en mente a la hora de procesar los datos y actuar en consecuencia, programando para eliminar este desfasaje en la propia rutina. Más adelante en este documento en el capítulo *Resultados*, se muestra a partir de pruebas que efectivamente esto ocurre.
El siguiente paso es entonces, el procesamiento de los datos recopilados.
Según lo visto en el capítulo *Tratamiento de Datos*, se debe asegurar una correcta medida de voltaje desde los electrodos, extrayendo la información que luego permitirá la correcta generación de imágenes de corte tomográfico. Esta información tiene que ver la parte real de la impedancia como se dijo, y se obtiene a partir de un batido de frecuencia con una señal de igual frecuencia que la inyectada, o sea, multiplicación muestra a muestra de la representación digital de la señal recibida de los electrodos $Acos(w_c t_n + \theta)$, con la señal local que describe una sinusoide de la forma 5.3 (extraída de la señal que controla la fuente de corriente). Luego la

aplicación de un filtro pasa-bajos (filtro *IIR*) y finalmente el valor deseado. En síntesis, el programa se implementa según el sistema dado en 5.1

Puesto que la reconstrucción de la imagen se realiza en a base un número determinado de medidas, todo el procedimiento anterior, desde la recolección de los datos hasta la obtención del valor de la impedancia, se repite 16 veces a lo largo de los 16 electrodos, 16 períodos.

Para ir intercambiando los electrodos de corriente y los de medida, se implementa una lógica de multiplexado siguiendo un patrón bien específico ya mencionado en capítulos anteriores. Simplemente recordando lo señalado se tiene que:

- Todas las medidas se realizan en un único sentido.
- La inyección de corriente se realiza a través de dos electrodos adyacentes.
- La medida de voltaje se realiza entre dos electrodos adyacentes.

Puesto que la multiplexación de los electrodos debe ser implementada por circuitería externa y controlada digitalmente desde el software, las características de la placa, la hacen dificultosa. Se estudió trabajar con el puerto *UART* y la salida digital serie que este posee, más un sistema externo que cambia de manera secuencial los 16 electrodos. Esta no es la implementación más óptima, pero el desarrollo trasciende las fronteras de este proyecto. En consecuencia, se implementa una lógica sencilla con el objetivo de realizar pruebas de funcionamiento y desempeño.

De funcionar todo correctamente, se consiguen entonces las medidas de voltaje que deben ser presentadas en un formato determinado (formato *IMPETOM*) en un archivo de texto plano **.txt*. Si bien el mismo DSP es capaz de correr una subrutina que cree este archivo, tal tarea también puede ser realizada en el GPP dejando recursos disponibles en el DSP para lo que realmente es potente, el procesamiento de señales. Entonces, el DSP, en lugar de crear el archivo a medida que reporta un valor, lo envía al GPP, o mejor dicho, el DSP guarda la variable en un lugar de memoria especificado y compartido con el GPP para luego enviar un*NOTIFY* avisando que tiene datos disponibles para leer. Ahora es el GPP quien se encarga de construir el archivo.

Finalmente, una vez recolectadas todas las medidas necesarias, el DSP cierra los periféricos, registros y buses utilizados.

6.3. GPP *General Purpouse Processor*

El núcleo GPP es el encargado de administrar todo el sistema, es quien bootea todo el programa desde la memoria no volátil, carga en el DSP el programa que corre para el procesamiento de las diferentes señales, administra los recursos de procesamiento (que sólo serán del DSP, pues es el único núcleo que requiere de

recursos) y realiza tareas de baja prioridad en tieme-slots que el sistema esté trabajando con recursos disponibles.

Siguiendo un orden cronológico, lo primero que el GPP realiza entonces es cargar desde memoria no volátil todo el programa, incluyendo OS y rutinas que corren en GPP, así como también los programas que serán luego de DSP. Una vez iniciados corriendo el *Sistema Operativo*, ya se está en posición de enviar el fragmento de programa que corre en DSP, mediante funciones del paquete de software *DSP-Link* que pueden convocarse fácilmente. La siguiente imagen ilustra el proceso de comunicación entre el DSP y el GPP que se mencionó 6.2.

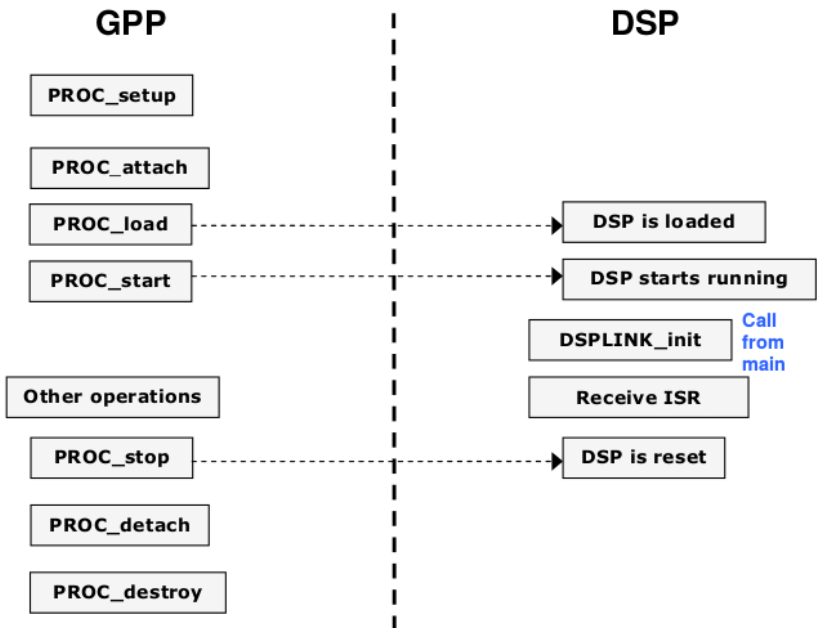


Figura 6.2: Mensajeria involucrada en el inicio de sistema y carga de rutinas en DSP [21]

Una vez se tengan cargados todos los programas y rutinas en su ambiente correspondiente, cada quien empieza con su parte dependiendo del lugar en el scheduler que el OS (en GPP) le de a cada proceso. Así mismo, cada vez que culmine una iteración de medida en el DSP como se mostró antes, éste último envía al GPP la notificación de que existen datos disponibles para su procesamiento. Una subrutina (también parte de *NOTIFY*) corren en el GPP y está censando por el aviso. Si lo hay, procede a leer los datos desde memoria y cargarlos en texto plano uno debajo del otro, y retornar un mensaje de *ACK*. El funcionamiento que se muestra en la siguiente imagen es similar al indicado en palabras.

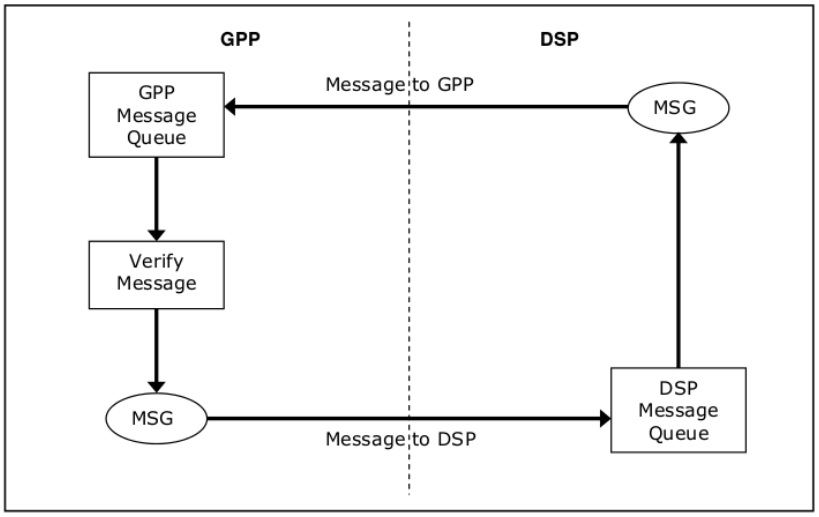


Figura 6.3: Flujo de trabajo para la comunicación entre DSP y GPP [21]

Esta subrutina que corren en el GPP se ejecuta una y otra vez hasta completar las medidas necesarias, creando un archivo **.txt* de 16 columnas con 16 valores cada una, obteniendo entonces el archivo en formato *IMPETOM*, similar a lo siguiente:

0,00000000 1,56001989 ...
0,00000000 0,00000000 ...
1,54458006 0,00000000 ...
1,49876550 1,53950988 ...
1,33448498 1,50723576 ...
1,19887399 1,38189918 ...
1,09887467 1,29799013 ...
0,86511900 1,08127454 ...
0,80065666 0,99865130 ...
0,87334110 0,90603512 ...
1,00098122 0,93534222 ...
1,18741221 1,09228120 ...
1,28875900 1,14102219 ...
1,36335409 1,38007590 ...
1,45099948 1,40033549 ...
1,55844644 1,59212845 ...

Capítulo 7

Resultados

Finalmente, ya con los principales problemas y desafíos de diseño resueltos, hemos de encaminar las pruebas de validación.

En este capítulo, se ha de poner especial énfasis en observar el desempeño en la medida de voltaje y la precisión obtenida haciendo uso del circuito fantoma.

7.1. Relevamiento de fuente de corriente - Primera parte

En una primera instancia se realizo una fuente de corriente de tipo Howland a partir de un amplificador operacional OPA2227 y 4 resistores de tolerancia al 5 %. Se controlo dicha fuente de corriente con un generador de señales y se conecto la fuente a un circuito fantoma R-C con $C = 47nF$ y R, que de ahora en mas llamaremos R-fantoma, variable. La capacitancia C ha sido seleccionada de manera tal que la impedancia resultante del paralelo, al variar R-fantoma en el rango del preset disponible de $10K\Omega$, abarque una variación de desfase lo mayor posible, así de esta manera, se podrá evaluar que tan bien el sistema calcula la parte real de la impedancia.

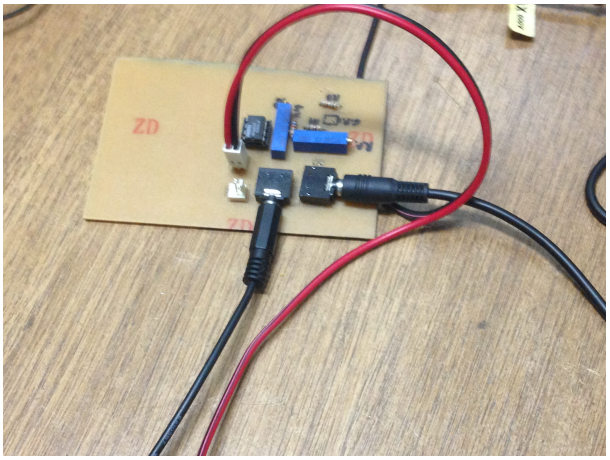


Figura 7.1: Circuito de pruebas. Foto tomada en laboratorio del NIB.

En la figura 7.1 se observa una foto del circuito implementado, el cual cuenta con una fuente de corriente de tipo Howland conectada a una carga RC paralelo a ser usado como fantoma. Las salidas y entradas al circuito se implementaron a través de conectores de tipo jacks de 3.5mm.

A partir de dicha configuración realizada en el laboratorio se relevó el desempeño de la fuente de corriente mediante medidas de voltaje en bornes del circuito R-C con el osciloscopio. Luego se conectó el circuito a los canales de audio de la placa EVMOMAP-L137 con el software IMPETOM-B y se realizó el relevamiento análogo.

Los resultados obtenidos se pueden observar en las tablas 7.1 7.2 y figuras 7.2 7.3 7.4 7.5 7.6.

Tabla Ensayo de fuente de corriente

Relevamiento de fantoma con osciloscopio						
R-fantoma (ohms)	C-fantoma (F)	Frecuencia (Hz)	Vin (V)	Vout (V)	Desfasaje (s)	Parte real estimada con osciloscopio (ohms)
150	4,7E-8	1,6E+4	2	0,14	5,0E-6	107,3
200	4,7E-8	1,6E+4	2	0,17	7,0E-6	113,9
300	4,7E-8	1,6E+4	2	0,18	8,0E-6	111,1
500	4,7E-8	1,6E+4	2	0,22	1,1E-5	67,0
700	4,7E-8	1,6E+4	2	0,23	1,2E-5	57,1
1000	4,7E-8	1,6E+4	2	0,23	1,3E-5	50,7
5000	4,7E-8	1,6E+4	2	0,24	1,5E-5	12,8

Tabla 7.1: Relevamiento de fuente de corriente tipo Howland conectada a carga RC con $C = 47nF$ y R variando en el rango de 150Ω a 5000Ω.

En la tabla 7.1 se observa una columna correspondiente al valor seteado en R-fantoma en cada caso. Los valores seleccionados de: frecuencia de la señal inyectada, C-fantoma y voltaje de entrada se mantuvieron constantes. El valor relevado de Voltaje de salida y desfase entre señales y por último, a partir de estos datos relevados, se calculo la parte real de la carga. Este ultimo dato puede ser comparado con el valor teórico calculado a partir de los parámetros del circuito y de esta manera evaluar el desempeño de la fuente.

En la tabla 7.2 se compara con los datos obtenidos en osciloscopio y los datos teóricos de parte real de la carga. Las columnas corresponden a R-fantoma que es el valor seteado en la resistencia de carga. Parte real teórica, es la parte real de la carga calculada a partir de los valores nominales de las componentes del circuito (R,C y W). La columna Parte real estimada en placa, corresponde al valor de parte real que calcula y entrega el software IMPETOM-B, mientras que la columna parte real estimada en osciloscopio corresponde a la parte real calculada a partir de las medidas que se relevaron con el osciloscopio. Luego se observa en las subsiguientes columnas, las diferencias entre estos datos, a modo de poder realizar comparaciones.

Tabla comparativa entre datos obtenidos por IMPETOM-B y datos obtenidos por osciloscopio

Relevamiento de fantoma con placa y diferencia entre medidas						
R-fantoma (ohms)	Parte real teórica (ohms)	Parte real estimada en Placa (ohms)	Parte real estimada con osciloscopio (ohms)	Diferencia porcentual entre parte real estimada con osciloscopio respecto parte real teórica	Diferencia porcentual entre parte real estimada en placa respecto parte real teórica	Diferencia porcentual entre parte real estimada en placa respecto parte real estimada con osciloscopio
150	99,8	89,0	107,3	7,4%	-10,8%	-17,0%
200	105,7	104,5	113,9	7,8%	-1,1%	-8,3%
300	99,7	106,5	111,1	11,5%	6,8%	-4,1%
500	76,0	83,8	87,2	14,7%	10,3%	-3,8%
700	58,6	72,9	72,5	23,6%	24,3%	0,5%
1000	42,9	53,2	54,9	28,1%	24,1%	-3,1%
5000	8,9	15,9	13,3	49,2%	77,7%	19,1%
promedio de apartamientos					18,8%	-2,4%

Tabla 7.2: Datos obtenidos en ensayo con placa de IMPETOM-B.

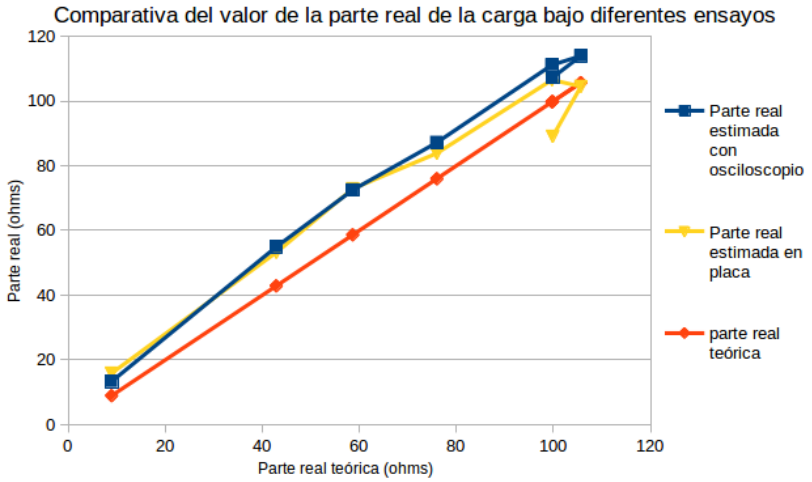


Figura 7.2: Comparativo de la parte real estimada de la carga a partir de los datos obtenidos en osciloscopio, y los valores teóricos de la parte real en función de los valores teóricos de la parte real.

En la gráfica de la figura 7.2 se puede observar como los datos obtenidos en la placa (amarillo) prácticamente se superponen a los obtenidos en osciloscopio (azul). Por otra parte los datos teóricos (naranja) se apartan levemente, el apartamiento, se puede observar, es prácticamente una constante a lo largo de todo el rango.

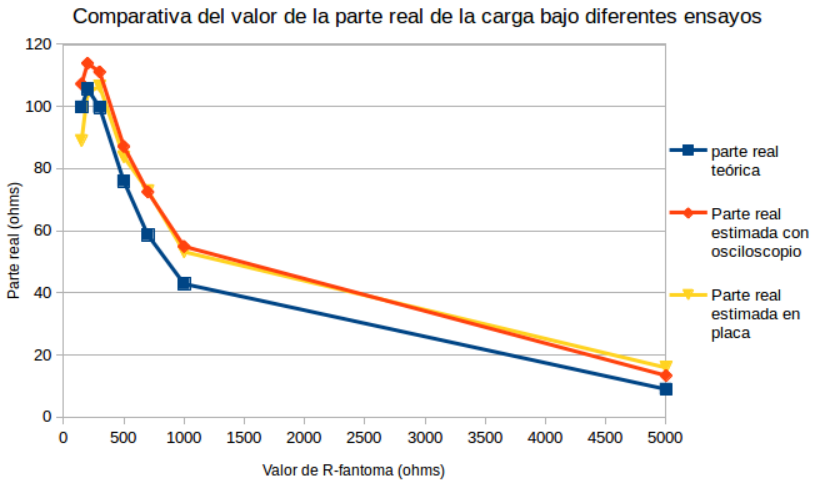


Figura 7.3: Comparativo de la parte real estimada de la carga a partir de los datos obtenidos en placa, datos obtenidos en osciloscopio, y los valores teóricos de la parte real en función de R-fantoma.

El gráfico de la figura 7.3 muestra, en este caso, los resultados obtenidos en función del valor de la resistencia R-fantoma, de esta manera se puede observar que el comportamiento en todos los casos es el correspondiente al de la carga paralelo RC, por lo que se puede concluir que la placa responde correctamente a los cambios de impedancia tanto en amplitud como en fase.

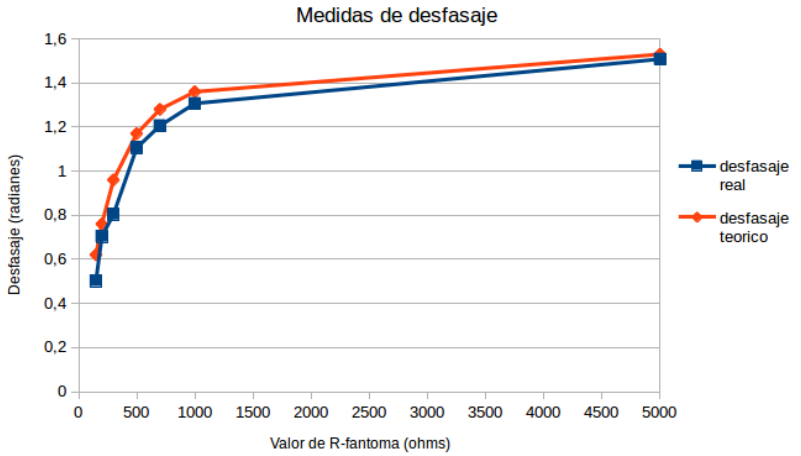


Figura 7.4: Comparativo del desfase introducido por la carga relevado a partir de los datos obtenidos en osciloscopio, y los valores teóricos de desfase en función de R-fantoma.

En el caso de la figura 7.4 lo que se observa es el desempeño del circuito de pruebas (fuente de corriente + carga paralelo RC), en cuanto al desfase introducido por las diferentes impedancias al variar R. No se gráfica en este caso el desfase visto

por la placa dado que el software IMPETOM-B solo entrega los valores de parte real de la carga.

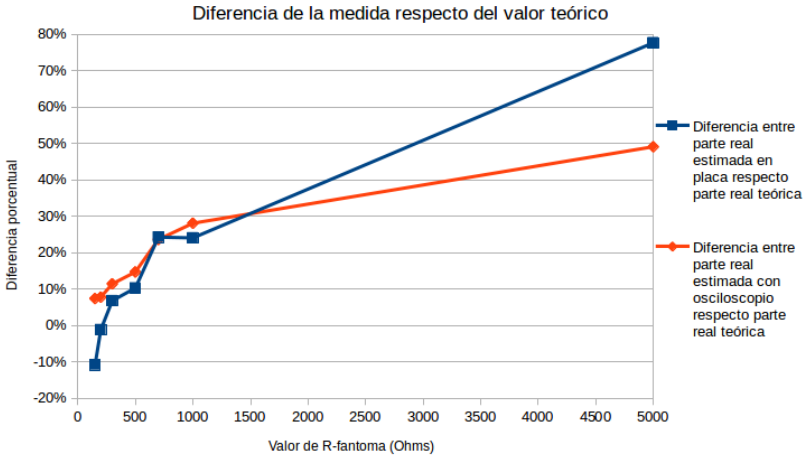


Figura 7.5: Comparativo de apartamiento de datos obtenidos en placa y datos obtenidos en osciloscopio en función del valor de R-fantoma.

La gráfica de la figura 7.5 permite analizar el apartamiento del dato que entregan ambos equipos, (Osciloscopio y placa IMPETOM-B) del dato teórico en función del valor de la resistencia R-fantoma. Es valido aclarar que, si bien las diferencias respecto del valor teórico alcanzan el 80 %, esto no significa que se este cometiendo dicho error en la medida ya que el valor teórico es simplemente un valor de referencia y la realidad de los hechos debe ser considerada a partir de los datos arrojados por el osciloscopio. No obstante el gráfico permite analizar el comportamiento de ambos instrumentos al aumentar R-fantoma el cual sigue la misma tendencia. Al observar el extremo en 5000Ω, la diferencia entre las gráficas es de un 30 %, pero tampoco este valor representa el error, ya que como se dijo anteriormente el porcentaje se ha calculado en base a un valor de referencia teórico.

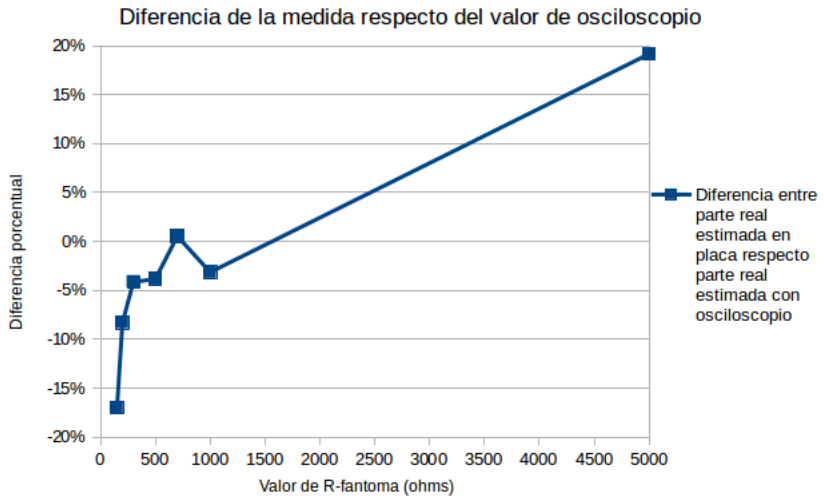


Figura 7.6: Apartamiento porcentual entre el valor de parte real obtenido a partir de datos de osciloscopio y datos de placa en función del valor de la resistencia de fantoma.

En el caso de la figura 7.6 se analiza el apartamiento de los datos de parte real de la carga arrojados por la placa, del valor de parte real calculado a partir del relevamiento en el osciloscopio. Los apartamientos llegan a ser de un 20 % del dato relevado en osciloscopio.

Continuando con las pruebas, y buscando obtener un relevamiento mas eficiente de la placa IMPETOM-B, se opto por mejorar el entorno de pruebas para una segunda instancia de relevamiento mas detallado.

7.2. Relevamiento de fuente de corriente - Segunda parte

Se implementó una fuente de corriente a partir de un integrado especifico (AD844) que elimina el problema de desapareo entre resistencias de diseño. Además, se interpuso un circuito seguidor entre la placa EVMOMAPL137 y la fuente de corriente de manera tal de poder eliminar posibles errores en la medida causadas por el acoplamiento entre etapas.

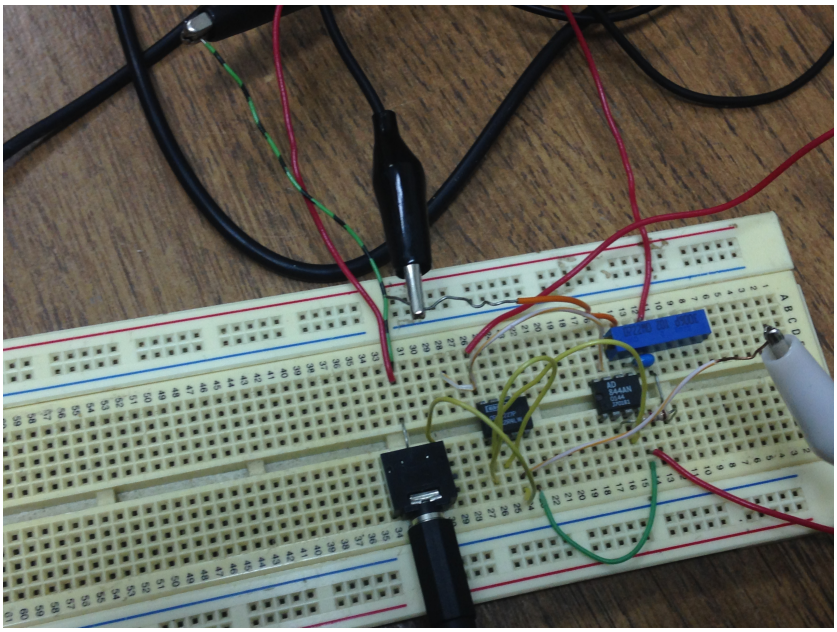


Figura 7.7: Circuito de pruebas. Foto tomada en laboratorio del NIB.

El circuito de la figura 7.7 se implemento sobre protoboard y cuenta con una fuente de corriente confeccionada a partir del integrado AD844, una etapa seguidora a la entrada del circuito y una carga RC paralelo a ser usada como fantoma.

Nuevamente, a partir de la configuración descripta anteriormente, se relevó el desempeño de la fuente de corriente mediante medidas de voltaje en bornes del circuito R-C con el osciloscopio.

Luego se conectó el circuito a los canales de audio de la placa EVMOMAP-L137 con el software IMPETOM-B y se realizó el relevamiento análogo.

Los resultados obtenidos se pueden observar en las tablas 7.3 7.4 y figuras 7.8 7.9 7.10.

Tabla ensayo de fuente de corriente

Relevamiento de fantoma con osciloscopio					
Rfantoma (ohms)	Vin (V)	Vout (V)	Desfasaje obtenido (s)	Desfasaje obtenido (rad)	Desfasaje teórico (rad)
350	0,89	0,097	9,80E-6	0,99	1,03
400	0,90	0,101	9,80E-6	0,99	1,08
450	0,90	0,105	1,06E-5	1,07	1,13
500	0,90	0,106	1,10E-5	1,11	1,17
550	0,90	0,109	1,16E-5	1,17	1,20
600	0,90	0,111	1,20E-5	1,21	1,23

Tabla 7.3: Relevamiento de fuente de corriente conectada a carga RC con $C = 47nF$ y R variando en el rango de 350Ω a 600Ω.

En la tabla 7.3 se observa una columna correspondiente al valor seteado en R-fantoma en cada caso, los valores seleccionados de: frecuencia de la señal inyectada, C-fantoma y voltaje de entrada que se mantuvieron constantes. El valor relevado de Voltaje de salida y desfase en segundos y en radianes. Por último se comparó con el desfase teórico que debería introducir la carga, calculado a partir de los parámetros del circuito.

Tabla comparativa entre datos obtenidos por IMPETOM-B y datos obtenidos por osciloscopio

RELEVAMIENTO DE DATOS EN PLACA IMPETOM-B					
R-FANTOMA (Ω)	Parte real teórica (Ω)	Parte real estimada en placa (Ω)	Parte real corregida en placa (Ω)	Diferencia porcentual entre parte real corregida en placa respecto a parte real teórica	Diferencia porcentual entre parte real estimada en placa respecto a parte real teórica
350	93,7	108,3	92,6	-1,23%	15,6%
360	92,5	107,8	92,0	-0,54%	16,6%
370	91,2	106,9	91,0	-0,28%	17,2%
380	90,0	105,9	89,8	-0,19%	17,7%
390	88,7	105,5	89,4	0,76%	18,9%
400	87,5	104,5	88,2	0,82%	19,4%
410	86,3	103,3	86,8	0,67%	19,7%
420	85,1	102,6	86,1	1,22%	20,6%
430	83,9	102,2	85,7	2,16%	21,9%
440	82,7	100,4	83,7	1,21%	21,5%
450	81,5	98,8	81,8	0,38%	21,2%
460	80,4	98,1	81,1	0,87%	22,1%
470	79,2	97,8	80,8	1,92%	23,5%
480	78,1	96,4	79,1	1,22%	23,3%
490	77,0	95,3	77,9	1,11%	23,7%
500	76,0	94,6	77,1	1,43%	24,5%
510	74,9	93,9	76,3	1,82%	25,3%
520	73,9	91,5	73,6	-0,41%	23,8%
530	72,9	90,8	72,8	-0,14%	24,5%
540	71,9	89,5	71,4	-0,75%	24,5%
550	70,9	88,7	70,4	-0,74%	25,0%
560	70,0	88,3	70,0	0,08%	26,2%
570	69,1	86,8	68,4	-1,01%	25,7%
580	68,2	86,1	67,6	-0,85%	26,4%
590	67,3	85,9	67,3	0,04%	27,7%
600	66,4	84,69	66,0	-0,66%	27,6%

Tabla 7.4: Datos obtenidos en ensayo con placa de IMPETOM-B, en la tabla se compara con los datos teóricos de parte real de la carga.

La primer columna de la tabla 7.4 corresponde al valor seteado en R-fantoma, luego Parte real teórica, corresponde al valor que toma la parte real de la carga

calculada a partir de los valores nominales de las componentes. La columna de parte real estimada en placa, corresponde al valor que calcula la placa IMPETOM-B. La columna parte real corregida, corresponde a una corrección lineal de los datos arrojados por la placa respecto de los datos teóricos. Es valida la aclaración de que si bien para el calculo de la parte real corregida, se realizó un ajuste por mínimos cuadrados entre los datos teóricos y los datos arrojados por la placa con 5 valores representativos de la muestra, este ajuste no agrega información sobre los datos obtenidos en placa y no será parte del software IMPETOM-B realizar dicha corrección. Los datos corregidos son calculados simplemente a los efectos de realizar el análisis comparativo entre los datos arrojados por la placa y los datos teóricos.

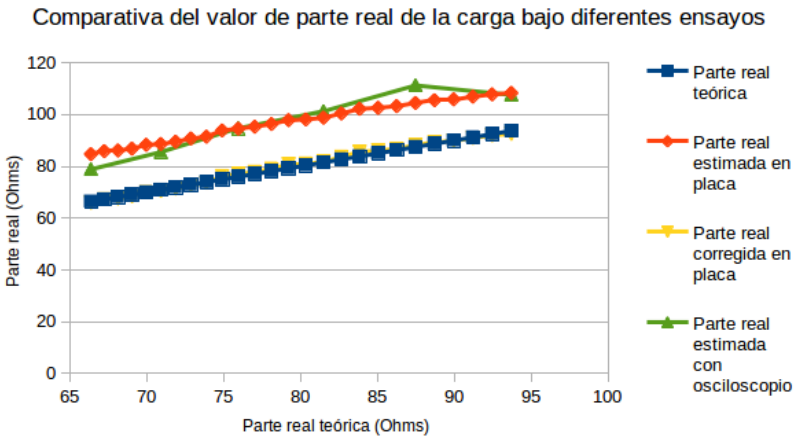


Figura 7.8: Comparativo de la parte real estimada de la carga a partir de los datos obtenidos en osciloscopio, valores teóricos, valores de placa, y valores de placa corregidos en función de los valores teóricos de la parte real.

En la gráfica de la figura 7.8 se observan las similitudes entre los datos obtenidos en la placa y el osciloscopio, y los datos de placa corregidos mediante mínimos cuadrados sobre 5 valores representativos y el valor teórico de parte real.

En la gráfica de la figura 7.9 se observan las similitudes entre los datos obtenidos en la placa y el osciloscopio, y los datos de placa corregidos mediante mínimos cuadrados sobre 5 valores representativos y el valor teórico de parte real.

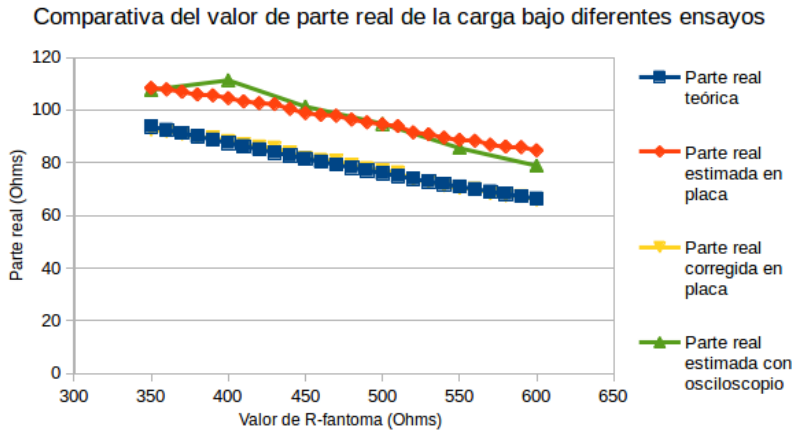


Figura 7.9: Comparativo de la parte real estimada de la carga a partir de los datos obtenidos en placa, datos obtenidos en osciloscopio, y los valores teóricos de la parte real en función de R-fantoma.

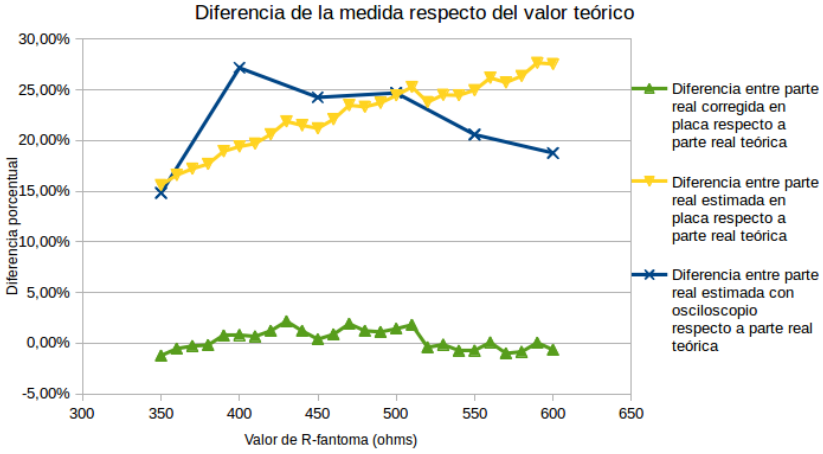


Figura 7.10: Comparativo de apartamientos entre valor teórico de parte real y datos obtenidos en placa, corregidos en placa y datos obtenidos en osciloscopio, en función de R-fantoma.

Analizando los datos obtenidos en ambos ensayos, se puede observar que a pesar de que los valores que arroja la placa tenga apartamientos que van del 15 % al 28 % de los valores nominales, en este caso, las diferencias entre estos datos se mantienen aparentemente constantes por lo que se asume, dichos apartamientos son provocados debido a errores sistemáticos en la medida. A los efectos de comprobar el alcance de la hipótesis propuesta sobre errores sistemáticos, se plantea un ajuste lineal a partir de 5 valores representativos.

Luego del ajuste, los apartamientos se reducen a un promedio de 0.34 % alcanzando un máximo de 2.16 % en uno de los datos. Se realizo un estudio de dispersión en la medida de la placa IMPETOM-B tomando un valor para R-fantoma de referencia (560Ω) y sometiéndolo repetidas veces al proceso de medición. Los

resultados obtenidos se pueden observar en la tabla 7.5 y figuras 7.11 7.12.

Tabla Análisis de dispersión en la medida de la placa

Resultados de ensayos sobre una misma carga (R-fantoma = 560 ohms)					
Ensayo	Parte real teórica (Ω)	Parte real estimada en placa (Ω)	Parte real corregida en placa (Ω)	Diferencia porcentual entre parte real corregida en placa respecto a parte rel teórica	Diferencia porcentual entre parte estimada en placa respecto a parte rel teórica
1	70,0	88,3	70,0	-0,08%	26,2%
2	70,0	88,4	70,1	-0,13%	26,2%
3	70,0	87,9	69,5	0,64%	25,6%
4	70,0	88,5	70,2	-0,33%	26,4%
5	70,0	87,8	69,4	0,83%	25,4%

Tabla 7.5: Valores obtenidos en placa a partir de sucesivos ensayos sobre una misma carga.

Se observa en la tabla 7.5, la primer columna corresponde al número de ensayo y la segunda columna al valor teórico de parte real de la carga. Luego la columna parte real estimada en placa corresponde al valor que arroja la placa IMPETOM-B mientras que parte real corregida en placa corresponde al ajuste lineal de los datos mediante procedimiento de mínimos cuadrados con 5 valores representativos de parte real respecto de los valores teóricos de los mismos. Las subsiguientes columnas, muestran la diferencia entre la parte real corregida y la parte teórica, y la parte real calculada en la placa y la parte real teórica respectivamente.

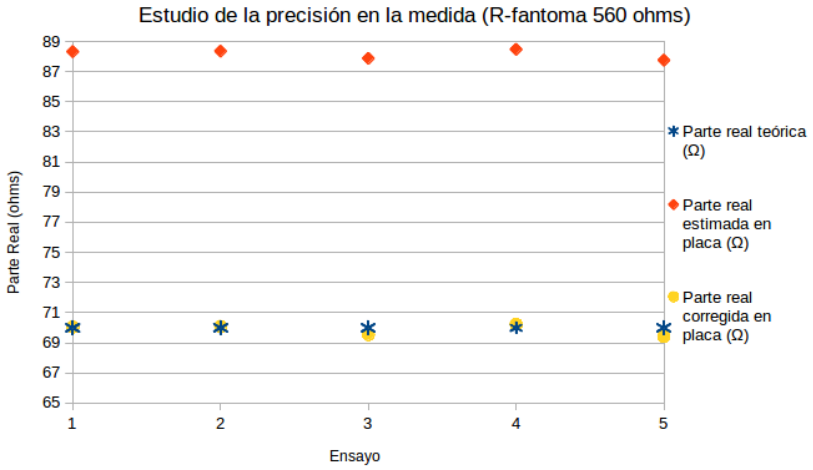


Figura 7.11: Sucesivos ensayos de medida de parte real de una misma carga.

Se observan, en la figura 7.11, parte real estimada en placa y parte real corregida en placa. Se observan los datos corregidos (amarillo) y los datos teóricos (azul) prácticamente coincidentes en la gráfica, llegando a existir apartamientos de menos de un 1 % en el peor caso. En naranja, los datos que entrega la placa sin corregir. Se puede observar que la diferencia entre los datos corregidos y sin corregir es prácticamente una constante

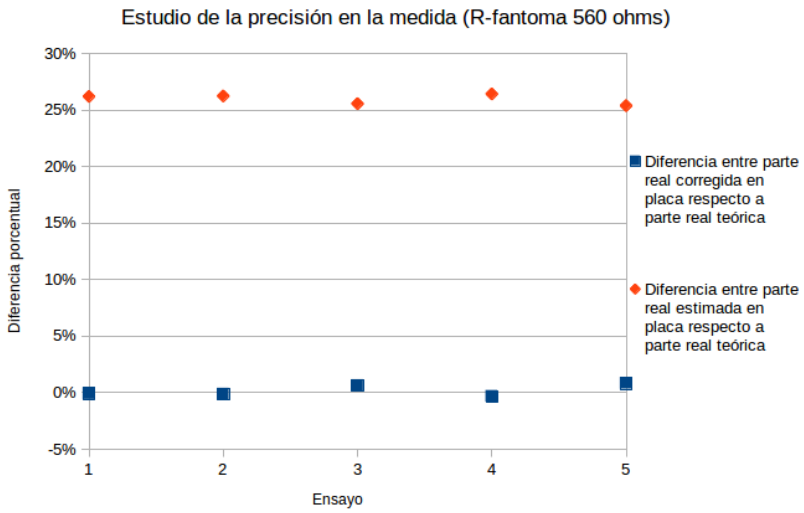


Figura 7.12: Diferencias respecto del valor teórico en sucesivos ensayos de medida de parte real de una misma carga.

Se observan, en la figura 7.12, diferencias de parte real estimada en placa y parte real corregida en placa. En naranja, la diferencia entre la parte real que entrega la placa IMPETOM-B y el valor de parte real teórico. En azul, la diferencia entre la parte real que entrega la placa IMPETOM-B corregida y el valor de parte real teórico.

Se observa que es posible lograr una configuración de sistema que tome medidas y logre calcular, con apartamientos menores a un 2 %, los valores de parte real de la impedancia.

Continuando con los ensayos se procedió a retirar el capacitor de la carga fantoma de manera tal de poder relevar el desempeño del sistema frente a la toma de valores resistivos en un rango mayor al relevado hasta el momento. Dichos valores en el entorno de los 560 Ohm, se asemejan mas a la resistividad del cuerpo humano a través del tórax. Los resultados obtenidos se pueden observar en la tabla 7.6 y figuras 7.13 7.14.

En la tabla 7.6 se compara con los datos teóricos de parte real de la carga. En la primer columna se tabula el valor seteado de R-fantoma y en la segunda la parte real estimada en la placa IMPETOM-B. Luego en la tercer columna se observa el valor de parte real corregida mediante mínimos cuadrados sobre 5 valores representativos de parte real obtenida y los correspondientes valores teóricos de parte real. En las 2 ultimas columnas se tabulan las diferencias entre la parte real corregida y la parte real teórica, y la diferencia entre la parte real sin corregir y la parte real teórica.

Tabla datos resistivos puros

R-fantoma (ohms)	Parte real estimada en placa (ohms)	Parte real corregida en placa (ohms)	Diferencia porcentual entre parte real corregida en placa respecto a parte real teórica	Diferencia porcentual entre parte real estimada en placa respecto a parte real teórica
350	322	352	-0,5%	-8,0%
360	334	364	-1,2%	-7,3%
370	341	372	-0,5%	-7,9%
380	349	381	-0,2%	-8,1%
390	359	392	-0,5%	-7,8%
400	366	400	0,1%	-8,4%
410	374	408	0,6%	-8,9%
420	385	419	0,1%	-8,4%
430	390	426	1,0%	-9,2%
440	402	438	0,4%	-8,7%
450	412	449	0,3%	-8,5%
460	421	459	0,3%	-8,5%
470	432	471	-0,2%	-8,1%
480	438	478	0,5%	-8,7%
490	452	493	-0,6%	-7,7%
500	457	497	0,5%	-8,7%
510	466	508	0,4%	-8,5%
520	472	514	1,1%	-9,2%
530	481	524	1,1%	-9,2%
540	495	539	0,2%	-8,4%
550	507	553	-0,5%	-7,7%
560	514	559	0,1%	-8,3%
570	521	568	0,4%	-8,5%
580	528	574	1,0%	-9,1%
590	531	579	1,9%	-9,9%
600	543,77	591,99	1,3%	-9,4%

Tabla 7.6: Datos obtenidos en ensayo con placa de IMPETOM-B.

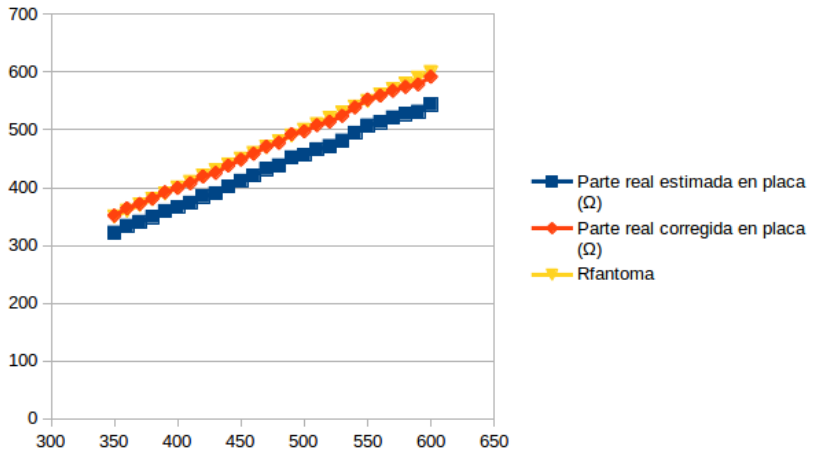


Figura 7.13: Valores de resistencia medida.

En la figura 7.13 se comparan resistencia calculada en placa, corregida en placa y valor de R-fantoma. En azul se gráfica la parte real obtenida por la placa IMPETOM-B, en naranja la parte real corregida a los datos teóricos mediante mínimos cuadrados sobre 5 valores representativos. Finalmente en amarillo se muestra el valor R-fantoma.

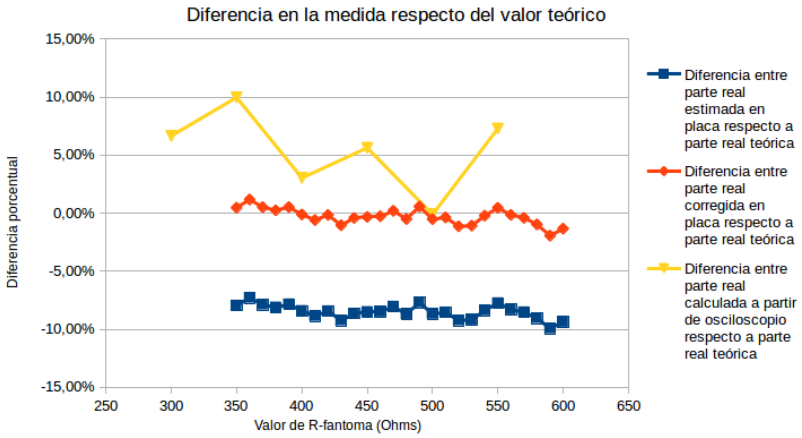


Figura 7.14: Diferencias respecto del valor teórico en ensayos de medida de parte real.

En la figura 7.14 se observan diferencias entre la parte real teórica y, la parte real estimada en placa (azul), parte real corregida en placa (naranja) y parte real obtenida a partir de datos de osciloscopio (amarillo).

Se puede observar que los resultados obtenidos en el rango de 350Ω a 600Ω son consistentes con las pruebas realizadas anteriormente de menor rango de parte real de la carga.

7.3. Conclusiones sobre el relevamiento

- Si bien se realizaron calibraciones que corrigen los datos mediante una relación lineal respecto del valor teórico, esto solo fue a modo de comparar los datos obtenidos tanto en osciloscopio como en la placa IMPETOM-B con un valor de referencia. Es el valor obtenido en osciloscopio el que se debe tomar como valor de referencia y, es frente a este valor que se deben observar y corregir los apartamientos del valor obtenido en la placa IMPETOM-B.

- Luego de observar el análisis completo, se puede concluir que el comportamiento de la placa IMPETOM-B para los rangos de voltaje manejados, tiene similares características al comportamiento del osciloscopio.

- En la primera etapa de relevamiento, la fuente de corriente y circuito fantoma fueron implementados mediante circuito impreso sobre una placa de pertinax, mientras que para la segunda etapa el circuito fue confeccionado en protoboard, no obstante se observaron mejoras en los datos recabados. Esto permite concluir que el desempeño de la placa IMPETOM-B seguramente este por encima del desempeño relevado en este proyecto. El relevamiento de desempeño de la placa IMPETOM-B es ampliamente mejorable en cuanto a, circuitos auxiliares que permitan medidas y cálculos mas precisos, así como también procedimientos y análisis de desempeño de instrumentos de medida que, por falta de tiempo, no fue posible realizar en este proyecto y quedará para trabajos futuros.

- Luego de tomar las medidas, al realizar la corrección lineal que elimine los errores sistemáticos en la adquisición, se obtuvieron resultados que se apartan en el peor caso en un 2.16 % del valor teórico de parte real siendo el promedio de apartamientos un 0.34 % .

- Se pudo comprobar que la corrección de desfase introducida de 1 muestra, analizada en el capitulo *Software IMPETOM-B*, fue suficiente para la obtención de datos coherentes al análisis teórico.

- Finalmente, si bien a lo largo de este análisis, se han realizado exhaustivas comparaciones a partir de un valor al que se ha denominado “Valor corregido”, dicha corrección es solo a fines de facilitar la comparación cualitativa entre los diferentes datos del problema y, en ningún momento, significa una mejora a los datos obtenidos sin corregir. Sumar y multiplicar por una constante no agrega información extra sobre los datos y es un proceso totalmente prescindible a los efectos de obtener una medida representativa de la parte real de la impedancia.

Esta página ha sido intencionalmente dejada en blanco.

Capítulo 8

Conclusiones y Trabajos Futuros

8.1. Conclusiones

De los resultados obtenidos, es posible decir que el proyecto *IMPETOM-B* logra lo siguiente:

- Implementar un sistema digital, capaz de controlar una fuente de corriente mediante una señal de voltaje.
- Concluir que el codec de audio *AIC3106* de la placa *EVM OMAP-L137*, es apta para cumplir con los objetivos del proyecto, es decir, generar y recibir señales en uno o varios canales, con la suficiente precisión.
- Cumplir con el objetivo principal del proyecto de generar un archivo en norma *IMPETOM* con los voltajes adquiridos por la placa.

En síntesis, *IMPETOM-B* incorpora al desarrollo de la EIT, un sistema EVM ARM+DSP para el control de una fuente de corriente, la recepción y manejo de datos recibidos de manera analógica, aprovechando las capacidades del codec de audio y las de procesamiento del núcleo DSP, para finalmente generar archivos en formato especifico a ser levantados en rutinas de reconstrucción de imágenes.
IMPETOM-B obtiene datos desde electrodos y los procesa, dejándolos listos para aplicarles la reconstrucción tomográfica, haciendo de interfaz entre circuitería externa de electrodos, y el software de reconstrucción.
Reviendo los objetivos marcados en la documentación del proyecto (*"Primarios: obtener en un archivo en norma IMPETOM con los valores de voltajes en los electrodos, y secundarios: realizar el control de la fuente de corriente con el códec de audio de la placa y realizar la adquisición de voltajes mediante los canales de entrada al códec de audio de la placa"*), se ha cumplido con los mismos.

8.2. Trabajos Futuros

Aún falta mucho camino por recorrer antes de obtener una versión funcional que integre todo el desarrollo en un solo sistema *IMPETOM*. Si bien se ha logrado obtener excelentes resultados a la hora relevar las señales desde electrodos y posterior procesamiento (desempeño de núcleo DSP y codec de audio), se mostró que la fuente de corriente implementada en este proyecto, tuvo un rendimiento debajo de lo esperado. Es esencial entonces, desarrollar una fuente de corriente estable que cumpla las más altas exigencia en desempeño y criterios seguridad, y es en sí mismo, todo un proyecto por separado.
Otro punto a mejorar es el multiplexado de electrodos. A pesar de que se estudió y trabajó el tema, no se implementa, debido a los tiempos disponibles y a estar por fuera del alcance de nuestro proyecto. No obstante, es sabido que la placa dispuesta para este proyecto (*EVM OMAP-L137*) cuenta con puertos de comunicación serie (UART o USB), un puerto pensado para expansión de memoria (*EMIF-A*), así como también puertos de comunicación *RJ-45*. Cualquiera de ellos, es candidato para la implementación del multiplexado.
Finalmente, otro punto en el desarrollo, es implementar el software de reconstrucción de imágenes en el propio sistema, utilizando para ello el núcleo DSP para la reconstrucción gestionada por ARM donde actualmente corre una plataforma linux.
El relevamiento de desempeño de sistema es mejorable, y también quedará para trabajos futuros, realizar un circuito para pruebas de alta precisión que permita evaluar el alcance real de *IMPETOM-B*.

Capítulo 9

Tiempos y Costos

En cierta medida, el proyecto de fin de carrera no es sólo la aplicación práctica de los conocimientos y habilidades adquiridos a lo largo de la carrera para el desarrollo, invención o estudio en el campo de interés, sino que también marca para los estudiantes, una primera aproximación a la actividad económica. Es sabido que la buena planificación del proyecto, ahorra al final del camino cientos de problemas y mucho dinero, permitiendo administrar esfuerzos y recursos de manera eficiente a lo largo del mismo. En este sentido, en el presente capítulo se exponen los tiempos y costos involucrados en el proyecto *IMPETOM-B* para finalmente realizar una comparativa de que tan apartada de la realidad estuvo la planificación del proyecto.

9.1. Tiempos

Existen dos grandes lecturas para la interpretación del tiempo dedicado en el proyecto, el primero refiere al tiempo dedicado por tarea o actividad, mientras que la segunda lectura, apunta a la dedicación total en horas semanales. En base a la primer lectura se tiene lo siguiente: 9.2

Desde el inicio del proyecto, uno de los objetivos internos del grupo era ser constantes a la hora de la dedicación a la tarea, evitando así calamidades sobre el final de los plazos y teniendo margen para otras actividades curriculares en el período del proyecto (léase parciales, exámenes y descansos). Se puede observar que durante el transcurso de la actividad, con excepción de los períodos de parciales, exámenes y vacaciones, se ha logrado estar casi en 15 horas semanales percápita.

En el gráfico 9.2 es de mencionar lo siguiente:

- Cuando se dicen *Pruebas*, se incluyen tanto las pruebas de performance y desempeño de la EVM que dieron lugar a la ventana de trabajo observada en el capítulo de *Resultados*, pruebas de configuración de la placa, de recolección y emisión de datos, y también de reconstrucciones y simulaciones realizadas en Matlab.

- Cuando se dice *Estudios Previos*, no sólo se abarca la historia del proyecto, la de EIT y diferentes componentes del nuevo sistema (Evaluation Board,

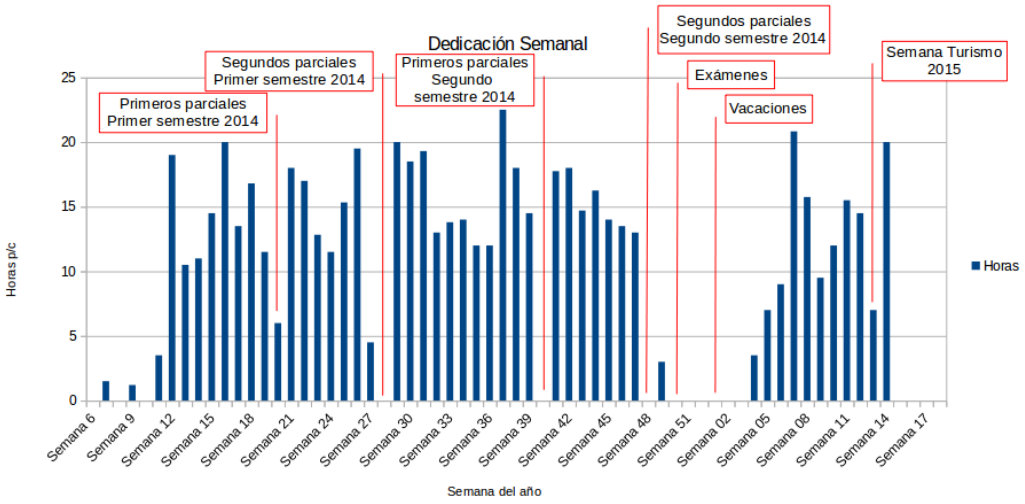


Figura 9.1: Gráfica de la dedicación semanal percapita en horas durante proyecto.

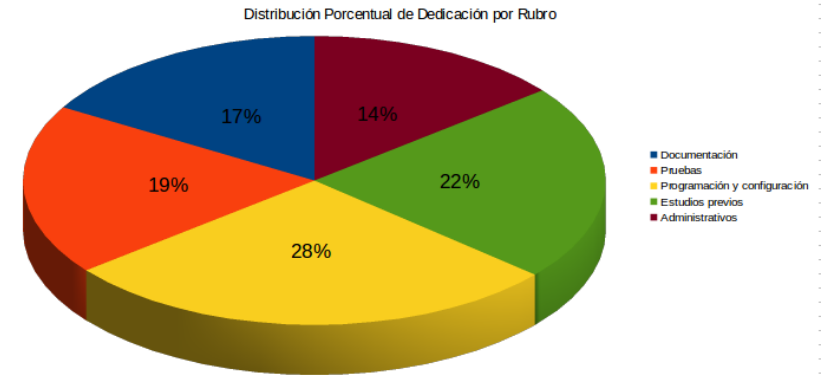


Figura 9.2: Gráfica de dedicación porcentual por Rubro durante proyecto.

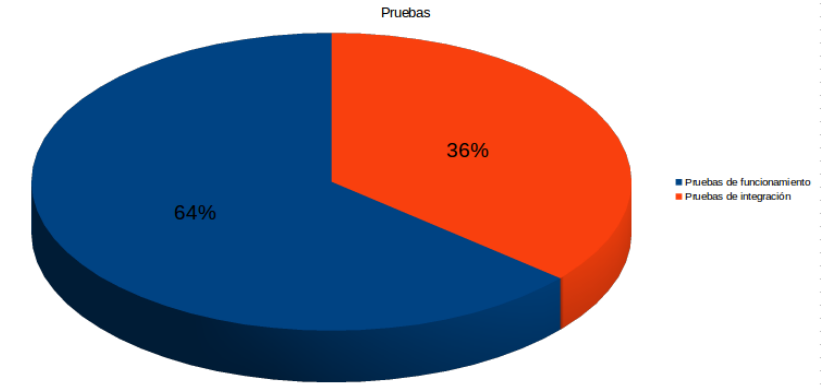


Figura 9.3: Distribución de tareas durante proyecto: Pruebas.

Codec de Audio, puertos disponibles), sino que también, aquel estudio que surgió a medida que transcurría el proyecto, como puede ser: investigación de reconstrucción de señal, consultas a *Textas Instruments*, papers referentes a reconstrucción de la parte real de la impedancia corporal, etc.

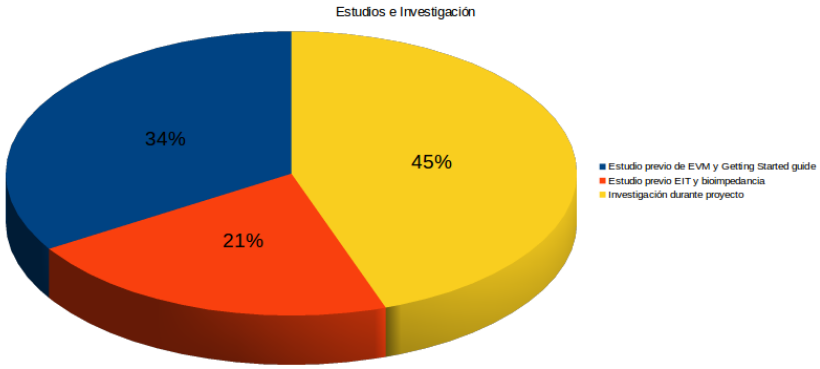


Figura 9.4: Distribución de tareas durante proyecto: Estudios.

- Al decir *Programación y Configuración* se indican, no solo las tareas referidas a la configuración de la placa propiamente dicha, sino que también a la puesta a punto de la misma y los host PC que utilizan las diferentes herramientas incluidas en el *Starter Kit*.

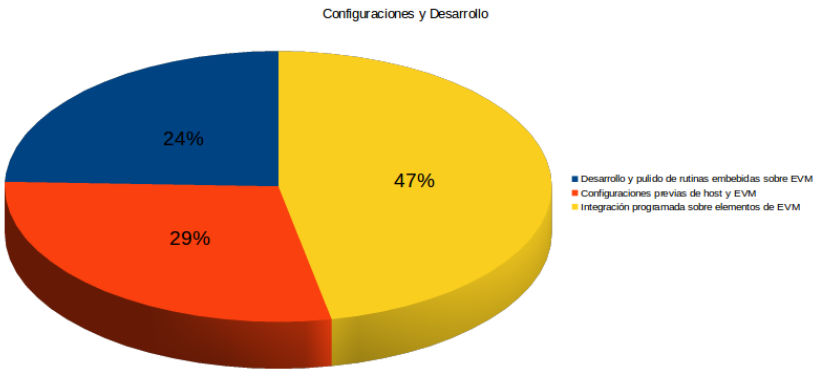


Figura 9.5: Distribución de tareas durante proyecto: Configuración y Desarrollo

- *Documentación*: Plasmar en un documento todo lo referido al proyecto así como también la configuración de herramientas para ello (Latex, Mendeley, bibliografía, etc).
- Las tareas *Administrativas* hacen referencias a reuniones, clases de proyecto y preparación de entregables (tareas, hitos, etc), y puestas a punto comunes dentro del grupo.

En ninguno de los anteriores items, se incluyen las horas de dedicación personales, que van desde la puesta a punto de las terminales de trabajo hasta elaboración

de rutinas de prueba en matlab y estudios para distintos propósitos del proyecto. Se estima que esta cantidad de horas se eleva a unas 50 para cada uno de los integrantes.

9.2. Planificación *vs* Real

En los hitos presentados durante el período del proyecto, se mostró el avance del mismo comparado con la planificación realizada al inicio 9.6.

ID	Nombre	Fecha de inicio	Fecha de fin	Duración	Antecesores	Recursos
0	ESTUDIO DEPROYECTOS ANTERIORES Y NOCIONES BASICASPREVIAS	17/03/14	11/04/14	20		
0.1	BIOLOGIA DEL CUERPO HUMANO	17/03/14	17/03/14	1		Fernanda
0.2	BIOIMPEDANCIA	17/03/14	17/03/14	1		Martín
0.3	NORMAS DE SEGURIDAD	17/03/14	17/03/14	1		Nicolás
0.4	PROYECTOS ANTERIORES	18/03/14	24/03/14	5		Martín Fernanda Nicolás
0.5	EIT	25/03/14	11/04/14	14		Martín Fernanda Nicolás
0.6	DOCUMENTACION 1	14/04/14	18/04/14	5	0	Martín Fernanda Nicolás
1	ESTUDIAR DSP- EVALUATION BOARD DISPONIBLE	21/04/14	27/05/14	27		
1.1	ARQUITECTURA Y RANGO DE FUNCIONAMIENTO	21/04/14	25/04/14	5		Martín Fernanda Nicolás
1.2	HERRAMIENTAS DE SOFTWARE ASOCIADAS A LA PLACA	28/04/14	23/05/14	20		
1.2.1	LENGUAJE PROGRAMACION	28/04/14	23/05/14	20		Fernanda Nicolás
1.2.2	ENTORNO DE PROGRAMACION Y COMPILACION	28/04/14	9/05/14	10		Martín
1.2.3	S.O. DE LA PLACA	12/05/14	23/05/14	10	1.1	Martín
1.3	PUESTA EN COMUN DEL GRUPO	26/05/14	26/05/14	1	1.2	Martín Fernanda Nicolás
1.4	HOLA MUNDO	27/05/14	27/05/14	1	1.3	Martín Fernanda Nicolás
1.5	DOCUMENTACION 2	28/05/14	3/06/14	5	1	Martín Fernanda Nicolás
2	EVALUAR EL ALCANCE Y POSIBILIDADES DEL CODEC DE AUDIO	4/06/14	17/06/14	10	1.1	
2.1	ESTUDIO DE LOS REQUERIMIENTOS DEL PROBLEMA	4/06/14	9/06/14	4		Martín Fernanda
2.2	ESTUDIAR EN PROFUNDIDAD EL CODEC	4/06/14	17/06/14	10		
2.2.1	ARQUITECTURA Y RANGO DE OPERACION	4/06/14	9/06/14	4		Nicolás
2.2.2	PUESTA EN COMUN	10/06/14	10/06/14	1	2.2.1	Martín Fernanda Nicolás
2.2.3	FUNCIONAMIENTO Y NO IDEALIDADES	11/06/14	17/06/14	5		Martín Fernanda Nicolás
2.3	CONCLUSION Y DOCUMENTACION 3	18/06/14	24/06/14	5	2	Martín Fernanda Nicolás
3	Buffer 1	25/06/14	8/07/14	10		Martín Fernanda Nicolás
4	IMPLEMENTAR PROGRAMACION DEL DSP	9/07/14	26/08/14	35	1 – 2	
4.1	IMPLEMENTAR Y TESTEAR FUNCIONES	9/07/14	19/08/14	30		
4.1.1	GENERACION DE SEÑALES	9/07/14	19/08/14	30		Martín Nicolás
4.1.2	RECOLECCION DE DATOS	9/07/14	19/08/14	30		Fernanda Nicolás
4.1.3	CONTROL	9/07/14	19/08/14	30		Martín Fernanda
4.2	UNIR Y TESTEAR SISTEMA	20/08/14	26/08/14	5	4.1	Martín Fernanda Nicolás
4.3	DOCUMENTACION 4	27/08/14	2/09/14	5	4	Martín Fernanda Nicolás
5	Buffer 2	3/09/14	23/09/14	15		Martín Fernanda Nicolás
6	IMPLEMENTAR FUENTE DE CORRIENTE	24/09/14	14/10/14	15		
6.1	ESTUDIO DE DIVERSAS POSIBILIDADES Y SUS COMPONENTES	24/09/14	30/09/14	5		Martín Fernanda Nicolás
6.2	NO IDEALIDADES	1/10/14	3/10/14	3		Martín Fernanda Nicolás
6.3	ELECCION Y CALIBRACIONDE LA FUENTE DE CORRIENTE	6/10/14	14/10/14	7		Martín Fernanda Nicolás
7	ESTUDIO Y ELECCION DE ELEMENTOS DE CIRCUITERIA EXTERNA	15/10/14	21/10/14	5		Martín Fernanda Nicolás
8	UNIR PARTES	22/10/14	4/11/14	10	6 – 4	Martín Fernanda Nicolás
9	TESTEO	5/11/14	25/11/14	15	8	Martín Fernanda Nicolás
10	DOCUMENTACION 5	26/11/14	2/12/14	5	9	Martín Fernanda Nicolás
11	Buffer 3	3/12/14	16/12/14	10		Martín Fernanda Nicolás

Figura 9.6: EDT, planificación de tareas del proyecto *IMPETOM-B*

Recordando lo visto en cada uno de ellos, durante el primero se tuvo la necesidad de replanificar las tareas y su duración, puesto las dificultades encontradas, tareas surgidas y la propia inexperiencia de planificación. Al final de cuentas, luego del hito 1, la planificación resulta en lo siguiente tabla: 9.1

Ya en el segundo hito, y con un poco más de tiempo disponible, las tareas y su duración fueron más ajustadas a lo planificado, aunque se agregó mas tiempo para pruebas y documentación. 9.2

Tal como se vaticinó en las clases de proyecto, el mantenerse dentro de los tiempos planificados de cada tarea, ayuda a administrar los recursos disponibles y enfocar los esfuerzos en determinadas áreas de interés o necesidad, pero sin perder de vista los plazos de entrega propios de cualquier proyecto. En el caso de *IMPETOM-B*, ayudó a dar idea del itinerario que se debía seguir y cuan lejos se estaba en cada punto de alcanzar el objetivo final, permitiendo ajustar a demanda las horas dedicadas y recursos.

ESTUDIO DEPROYECTOS ANTERIORES Y NOCIONES BASICASPREVIAS	17/03/14	11/04/14
BIOLOGIA DEL CUERPO HUMANO	17/03/14	17/03/14
BIOIMPEDANCIA	17/03/14	17/03/14
NORMAS DE SEGURIDAD	17/03/14	17/03/14
PROYECTOS ANTERIORES	18/03/14	24/03/14
EIT	25/03/14	11/04/14
DOCUMENTACION 1	14/04/14	18/04/14
ESTUDIAR DSP- EVALUATION BOARD DISPONIBLE	21/04/14	27/05/14
ARQUITECTURA Y RANGO DE FUNCIONAMIENTO	21/04/14	25/04/14
HERRAMIENTAS DE SOFTWARE ASOCIADAS A LA PLACA	28/04/14	23/05/14
LENGUAJE PROGRAMACION	28/04/14	23/05/14
ENTORNO DE PROGRAMACION Y COMPILACION	28/04/14	09/05/14
S.O. DE LA PLACA	12/05/14	23/05/14
PUESTA EN COMUN DEL GRUPO	26/05/14	26/05/14
HOLA MUNDO	27/05/14	27/05/14
DOCUMENTACION 2	28/05/14	03/06/14
EVALUAR EL ALCANCE Y POSIBILIDADES DEL CODEC DE AUDIO	04/06/14	21/08/14
ESTUDIO DE LOS REQUERIMIENTOS DEL PROBLEMA	04/06/14	15/07/14
ESTUDIAR EN PROFUNDIDAD EL CODEC	16/07/14	21/08/14
Protocolo MCASP	16/07/14	04/08/14
Registros del codec aic3106	16/07/14	04/08/14
Software test de codec en DSP	16/07/14	04/08/14
PUESTA EN COMUN	05/08/14	21/08/14
IMPLEMENTAR PROGRAMACION DEL DSP	26/08/14	30/09/14
Estudio e implementación de DSPLINK	26/08/14	05/09/14
Preparación para Hito 1	09/09/14	12/09/14
Hito 1	15/09/14	15/09/14
Implementar Software IMPETOM-B	15/09/14	23/09/14
Software IMPETOM-B en DSP	15/09/14	23/09/14
Software IMPETOM-B en GPP	15/09/14	23/09/14
UNIR Y TESTEAR SISTEMA	24/09/14	30/09/14
DOCUMENTACION 3	01/10/14	07/10/14
IMPLEMENTAR FUENTE DE CORRIENTE	08/10/14	28/10/14
ESTUDIO DE DIVERSAS POSIBILIDADES Y SUS COMPONENTES	08/10/14	14/10/14
NO IDEALIDADES	15/10/14	17/10/14
ELECCION Y CALIBRACIONDE LA FUENTE DE CORRIENTE	20/10/14	28/10/14
ESTUDIO Y ELECCION DE ELEMENTOS DE CIRCUITERIA EXTERNA	29/10/14	04/11/14
UNIR PARTES	05/11/14	18/11/14
TESTEO	19/11/14	09/12/14
DOCUMENTACION 4	10/12/14	16/12/14

Tabla 9.1: WBS post Hito 1 (al 15 de setiembre de 2014).

Nombre	Fecha de inicio	Fecha de fin
ESTUDIO DE PROYECTOS ANTERIORES Y NOCIONES BASICAS PREVIAS	17/03/14	11/04/14
BIOLOGIA DEL CUERPO HUMANO	17/03/14	17/03/14
BIOIMPEDANCIA	17/03/14	17/03/14
NORMAS DE SEGURIDAD	17/03/14	17/03/14
PROYECTOS ANTERIORES	18/03/14	24/03/14
EIT	25/03/14	11/04/14
DOCUMENTACION 1	14/04/14	18/04/14
ESTUDIAR DSP - EVALUATION BOARD DISPONIBLE	21/04/14	27/05/14
ARQUITECTURA Y RANGO DE FUNCIONAMIENTO	21/04/14	25/04/14
HERRAMIENTAS DE SOFTWARE ASOCIADAS A LA PLACA	28/04/14	23/05/14
LENGUAJE PROGRAMACION	28/04/14	23/05/14
ENTORNO DE PROGRAMACION Y COMPILACION	28/04/14	9/05/14
S.O. DE LA PLACA	12/05/14	23/05/14
PUESTA EN COMUN DEL GRUPO	26/05/14	26/05/14
HOLA MUNDO	27/05/14	27/05/14
DOCUMENTACION 2	28/05/14	3/06/14
EVALUAR EL ALCANCE Y POSIBILIDADES DEL CODEC DE AUDIO	4/06/14	21/08/14
ESTUDIO DE LOS REQUERIMIENTOS DEL PROBLEMA	4/06/14	15/07/14
ESTUDIAR EN PROFUNDIDAD EL CODEC	16/07/14	21/08/14
Protocolo MCASP	16/07/14	4/08/14
Registros del codec aic3106	16/07/14	4/08/14
Software test de codec en DSP	16/07/14	4/08/14
PUESTA EN COMUN	5/08/14	21/08/14
IMPLEMENTAR PROGRAMACION DEL DSP	26/08/14	10/10/14
Estudio e implementación de DSPLINK	26/08/14	5/09/14
Preparación para Hito 1	9/09/14	12/09/14
Hito 1	15/09/14	15/09/14
Implementar Software IMPETOM-B	22/09/14	9/10/14
Software IMPETOM-B en DSP	22/09/14	9/10/14
Software IMPETOM-B en GPP	22/09/14	9/10/14
UNIR Y TESTEAR SISTEMA	10/10/14	10/10/14
DOCUMENTACION 3	13/10/14	22/10/14
Estudio de circuito de multiplexado	23/10/14	21/11/14
Estudio de tratamiento de la señal en DSP	23/10/14	21/11/14
DOCUMENTACION 4	2/12/14	8/12/14
Estudio para optimizar en el consumo de recursos	20/01/15	26/01/15
Testeo y corrección de errores	27/01/15	25/02/15
Fuente de corriente y circuito de prueba	12/02/15	18/02/15
Pruebas de desempeño	27/02/15	10/03/15
Preparación para Hito 2	26/02/15	26/02/15
Hito 2	3/03/15	3/03/15
DOCUMENTACION 4	11/03/15	31/03/15

Tabla 9.2: WBS post Hito 2 (al 3 de marzo de 2015.)

9.3. Costos

9.3.1. Costo de Desarrollo

El costo de desarrollo que aquí se presenta, es en esencia, el costo de dedicación de los recursos humanos integrantes del proyecto. En el siguiente cuadro se enumeran las horas de dedicación por tarea o rubro y el costo generado por ello, fijando la hora-hombre en U$S 25 9.3.

Tarea	Dedicación p/c (Horas)	Costo (U$S)
Estudio previo de EVM y Getting Started guide	48,85	1221,25
Estudio previo EIT y bioimpedancia	31	775
Investigación durante proyecto	64,39	1609,75
Pruebas de funcionamiento	79,88	1997
Pruebas de integración	45,32	1133
Desarrollo y pulido de rutinas embebidas sobre EVM	44,78	1119,5
Configuraciones previas de host y EVM	52,5	1312,5
Integración programada sobre elementos de EVM	85,85	2146,25
Documentación	108,16	2704
Administrativos	93,46	2336,5
Total	**654,19**	**16354,75**

Tabla 9.3: Costos de Desarrollo de *IMPETOM-B* durante proyecto, disgregado según dedicación en horas por rubro

En definitiva, el costo de desarrollo total del proyecto, sumando los aportes de los tres integrantes asciende a aproximadamente U$S 16.354,75 x 3 = U$S 49.064,25. Nuevamente se deja fuera del cálculo las horas extras dedicadas por cada uno de los integrantes (50 horas @ U$S 25 la hora, representan U$S 1.250 por integrante extras, o U$S 3.750 en total).

La discriminación de tareas realizadas en la tabla 9.3, permite evidenciar posibles áreas de ahorro del proyecto, como lo pueden ser las *Configuraciones previas de host y EVM* o *Administrativas*.

9.3.2. Compras

Dado que para el proyecto en cuestión ya se tenia disponible la placa *EVM OMAP-L137*, nos permite llegar a muy buenos costos de compra finales. Sin embargo, y para tener una visión mas global del proyecto, se incluyen los costos de la placa.

9.3.3. Posibilidades de Comercialización

Antes de explorar las posibilidades de comercialización, recuerde que el proyecto *IMPETOM-B* es parte de un proyecto de desarrollo más global de la tecnología EIT, donde en el mejor de los casos, se está a poco más de mitad de camino de

Insumos	Costo (U$S)
Circuitería Externa de Multiplexación	75
Circuitería Externa de Fuente de Corriente***	150
Placa EVM OMAP-L137 de *Texas Instruments*	394
envío*	64,4
impuestos (hasta un 60% del valor de etiquetado del bien)**	365,04
Total	**1048,44**

Tabla 9.4: Costos de Insumos de proyecto *IMPEOTM-B*

lograr una implementación de un sistema funcional compacto de EIT. Aún falta integrar una etapa de reconstrucción de imágenes [8], la circuitería externa vinculada a la multiplexación entre los electrodos, construcción de una fuente de corriente estable y de precisión, etc. Aclarado esto, es posible proyectar las posibilidades de desarrollo del sistema completo.

Lo primero a observar son los insumos. La placa donde se realiza la recolección y preparación de datos para ser enviada al sistema de reconstrucción de imágenes (*IMPETOM-B*, es actualmente una placa de evaluación de *Texas Instruments*), y es por si sola, un sistema de evaluación completo que posee funcionalidades, puertos y periféricos que son totalmente prescindibles en este proyecto de EIT. Ejemplificando esto, se destacan los puertos *RJ-45*, puertos *USB*, Codecs de Video, luces leds, entre otros. Si se realizan pedidos en cantidades (diluyendo además los costos de envíos) y especificando los componentes necesarios (Multi-Core ARM+DSP, Codec de Audio, puertos Line-in left y right, etc), incluyendo la circuitería de multiplexado y fuente de corriente, es posible obtener un mejor precio por unidad que los actuales U$S 733,08, más costos de circuitería externa, pudiendo alcanzar valores aproximados a U$S 350,00. Existen otras alternativas más económicas a la EVM de *Texas Instruments*, como lo son algunas placas *Arduino*, cuyo costo apenas puede superar los U$S 200,00 en el mercado local, y que además cuentan con la posibilidad de desarrollar sobre una plataforma basada enteramente en sistemas *Open Source*, reduciendo a cero los costos por licencias. Igualmente, para esta última alternativa se debe re-estudiar los alcances y posibilidades como se realizó para este proyecto.

Mucho del desarrollo y algoritmia del sistema de reconstrucción de imágenes ya se encuentra resuelto en el proyecto *IMPETOM-I* [8], aunque sería una buena oportunidad de migrar hacia el mundo *Open Source*, similar a lo que se mencionó en el anterior párrafo.
[1]

Por otro lado, y en base a los costos generados en este y otros proyectos *IMPE-*

[1] * Suponiendo la compra de insumos especiales en el exterior, como por ejemplo operacionales AD844

Insumos	Costo (U$S)
Placa de *Texas Instruments* especialmente diseñada que incluye fuente de corriente y circuito multiplexador	350
envío*	64,4
impuestos (hasta un 60% del valor de etiquetado del b	248,64
Total	**663,04**

Tabla 9.5: Costos en insumos para fabricación de 1 *IMPEOTM-B*

TOM [8] [10] [7], la principal inversión radica en las horas hombres invertidas en investigación y desarrollo de las diferentes etapas. A groso modo, y suponiendo similar tiempo de dedicación para el resto de los proyectos, diremos que el costo de horas que resta agregar al costo total, puede ascender a otros U$S 100.000,00.

Sumando el resto de los proyectos previos, que sentaron las bases de este desarrollo, los costos finales no son nada menores 9.6.

Proyectos	Costo (U$S)	Costo al día de hoy (U$S) ****
Impetom-C @ 2002	80443	218774,69
Impetom-I @ 2002	80778	219685,77
Impetom @ 2005 *****	50000	135981,19
Impetom-48 @ 2007	21534	58564,38
Impetom-B @ 2015	49898,69	49898,69
Proyectos de implementación de fuente de corriente y circuito multiplexador	100000	100000
Total	**382653,69**	**782904,71**

Tabla 9.6: Costos de proyectos *IMPETOM*, previos y futuros a *IMPETOM-B*, incluyéndolo.

[2] [3]

Luego del desarrollo, y ya con los procesos pulidos y terciarizados (es posible pedir a la compañía de construcción de placas que integre la etapa de multiplexado y generación de señal de corriente), la configuración y ensamblado de cada una de las unidades consta de 3 etapas bien metódicas:

- Compra de insumos
- Carga de configuración: etapa recolectora de datos y software de reconstrucción.
- Integración y pruebas

Sólo basados de la experiencia recolectada, es posible crear tal vez 2 sistemas EIT por semana con la ayuda de 2 estudiantes (U$S 10 la hora contando deducciones) dedicados 4 horas diarias. La ecuación mensual resulta entonces U$S 10,00

[2] **** Retroactivo a 8 % anual

[3] ***** No se tiene demasiado detalle en la documentación

x 20 horas x 4 = U$S 800,00 por estudiante. Luego multiplicando por los 2 estudiantes, la suma asciende a U$S 800,00 x 2 = U$S 1.600,00, traducido a U$S 200,00 por placa construida en el mes.

Insumos	Costo (U$S)
Placa de *Texas Instruments* especialmente diseñada que incluye fuente de corriente y circuito multiplexador	2800
envío*	75,9
impuestos (hasta un 60% del valor de etiquetado del bien)**	1725,54
Mano de Obra de configuración y ensamblado	1600
Total	**6201,44**
Costo por unidad	**775,18**

Tabla 9.7: Costos de insumos de construcción al por mayor

En definitiva, se tiene dos columnas de costos principales, una el desarrollo del sistema y la otra, el costo por unidad construida. Tomando en cuenta sólo la segunda de las perspectivas, es posible tener un Tomógrafo por Impedancia Eléctrica de fabricación nacional por U$S 775,18 aproximadamente.

Actualmente, existen sistemas basados en EIT ya comerciales, uno de ellos es el *PulmoVista* patentado por la empresa alemana *Draeger* [5], que brinda similares funciones a lo pretendido por *IMPETOM*, pero cuyo valor asciende a los U$S 100.000,00 la unidad. Sin hacer un gran análisis de marketing y mercado para establecer el precio final de venta de *IMPETOM*, es fácil apreciar que existe una muy buena ventana de oportunidad. Considerando la cantidad de camas de CTI disponibles en Uruguay (aproximadamente unas 676 a octubre del 2013), siendo optimistas, es posible alcanzar un 85 % del total de estas camas. Suponiendo simplemente a modo de referencia un precio de venta accesible de U$S 5.000,00, representaría un ingreso de U$S 2.873.000,00, suma que como puede ver, amortiza además todo el desarrollo previo.

De todas maneras, existen otros aspectos no menos importantes a la hora de establecer una empresa, como lo son horas de capacitación a médicos, desarrollos posteriores, mantenimiento y consultas post-venta y un siempre prudente fondo jurídico.

9.4. Recapitulo

Es posible separar el análisis anterior en dos pilares, el primero referido a la planificación realizada en un principio y que tan ajustada fue a lo largo del período, y el segundo pilar, apuntado a los costos del proyecto, tanto particular a este *IMPETOM-B*, como al resto de los proyectos y en su conjunto.

Referidos al primero de los pilares indicados, si bien al principio se estuvo algo desfasados con la planificación, propio de la inexperiencia en el tema y los problemas surgidos a medida que transcurría el proyecto, luego se ha ajustado relativamente bien a los tiempos finales manejados.

Para lo segundo, en resumen y desde una proyección optimista, es posible alcanzar una versión funcional y 100 % comerciable de un sistema EIT de desarrollo, ensamblado y configuración nacional a relativamente corto plazo (3 años de trabajo), de costo de fabricación no superior a los U$S 800,00 y de inversión amortizable posiblemente sólo con el mercado local. Es sin lugar a dudas, una ventana de oportunidad para desarrollar y explotar un sistema revolucionario, no invasivo y de muy bajo costo comparado con otras alternativas del mercado.

Esta página ha sido intencionalmente dejada en blanco.

Apéndice A

Configuración Inicial del Hardware

Para trabajar con la *DSP evaluation board OMAPL137/TMS320 (EVM)* disponible, es necesario comprender en primer lugar los elementos de software básicos que la componen y hacen a su funcionamiento así como también su instalación. Para realizar esta tarea se cuenta con el manual incluido en la instalación del *PSP* (*Platform Software Package* [19]) realizada en la PC de trabajo.

Poner en funcionamiento la EVM en cuestión requiere la instalación del software básico brindado por el fabricante. La placa cuenta con dos procesadores, un *DSP* (*Digital Signal Processor*) y un procesador con arquitectura *ARM* (*Advanced RISC (Reduced Instruction Set Computer) Machine*). Estos procesadores necesitan los llamados *ubl's* (*user boot loader*) que se encargan de inicializar parámetros del sistema (direcciones de puertos, periféricos, buffers y buses, espacios de memoria, etc). En primer lugar esta el *ubl* que corresponde al DSP (*dsp-spi-ais.bin*), luego para el ARM se tiene el(*ubl-spi.bin*), y por último, el sistema debe correr el software *u-boot* (*universal bootloader*) (*u-boot.bin*) mediante el cual será posible configurar y cargar el sistema operativo en la placa.

Además de lo anterior, existen diferentes configuraciones que varían dependiendo del tipo de memoria a utilizar. Se ha seleccionado para este proyecto la *SpiFlash0*, disponible en el hardware, para cargar los *ubl's* y el software *u-boot* así como también para el kernel del sistema operativo. Para el file system se elige trabajar en la *SD Card*.

Según lo mostrado, y antes de instalar cualquier aplicación, se debe reparar en los requerimientos mínimos del sistema que posibilitan el correcto desarrollo, así como también, cuáles de las herramientas disponibles en el *Sterted Kit* son inevitablemente requeridas.

A.1. Programación del OMAP-L137 EVM

A.1.1. Requerimientos de Sistema

La completa instalación de todos los componentes *PSP* (*Plataform Support Packege*) requiere tanto un host Microsoft Windows como un Linux. La máquina Microsoft Windows es requerida para correr el CCStudio v3.3, necesario para construir los User Boot Loader (UBL), los flash writers así como también puede ser usado para grabar las imágenes de boot (UBL, u-boot) dentro de la flash usando los flash writers. El host Linux es necesario para los siguientes propósitos:

- Compilar el U-Boot y Linux kernel.
- El hosting del servidor TFTP requerido para descargar a la placa las imágenes del kernel y del file system usando U-Boot.
- Hosting del servidor NFS para bootear la EVM como usuario root del file system.
- Compilar la herramienta DSP-Link descripta más adelante en este mismo capítulo.

Es posible usar cualquiera de los host disponibles, tanto el Microsoft Windows como el Linux para correr un emulador de terminal (HyperTerminal) y conectarse al EVM usando el puerto serial.

A.1.2. Requerimientos de Host para Softwares

Viendo los requerimientos desde le punto de vista del software necesario, se tiene lo siguiente:

- CCstudio 3.3.38 corre sobre Microsoft Windows.
- MontaVista Pro 5.0 para ARM v5t sobre Linux
- Aplicación serial para consola terminal que puede correr tanto sobre Microsoft Windows como sobre Linux.
- Servidores TFTP y NFS (generalmente sobre host linux).

El procedimiento de instalación recomendado [19] sugiere la instalación del paquete *PSP* en un host Linux y luego copiar la porción usada en Microsoft Windows (proyectos de CCStudio para construir los UBLs y los flash writers). De esta manera, para usuarios OMAP-L137 se deben incluir los siguientes programas del paquete:

- *OMAPL137_arm_setuplinux_1_00_00_11.bin*: Instalador para estaciones Linux del software *Starter Kit* para OMAP-L137 ARM. Este software de desarrollo básico incluye a su vez:

  - Paquetes Soporte de Plataforma Linux (PSP) que incluyen utilidades board-flashing que corren en ARM y driver de Linux.
  - Utilidades del DSP/BIOS.
  - Codec Engine.
  - Utilidades Board-Flashing que corren sobre el Core DSP.
  - Dsplinker.
  - Driver de bajo nivel (Low Level Driver LLD) EDMA (correspondientes al DSP).
  - Componentes de Framework incluyendo DMAN3 (no utilizado en este proyecto).
  - Utilidades Linux incluyendo CMEM (no utilizado en este proyecto).
  - Paquete de algoritmos xDais (no utilizado en este proyecto).

- *LPTB-02.03.00.02-beta.bin*: Instalador para el banco de prueba de preformance de los driver de linux (no utilizado en este proyecto).

- *OMAPL137_dsp_setupwin32_1_00_00_11.exe*: Instalador en terminal Windows encargado de instalar el software *Starter Kit* para OMAP-L137 DSP, que además incluye otros componentes básicos:

  - Utilidades de DSP/BIOS, Codec Engine, utilidades Board-flashing del DSP, Dsplink, EDMA LLD, componentes Framework, utilidades para Linux, y paquetes de algoritmos xDais. Estos paquetes son idénticos a los contenidos en el *OMAPL137_arm_setuplinux_1_00_00_11.bin*.
  - PSP DSP/BIOS (Platform Support Package) que incluye drivers DSP/BIOS y ejemplos.
  - File System de tiempo real sobre DSP/BIOS.

- *setupwin32_ndk-2_0_0_0.exe*: Otro instalador para un PC Windows que instalará además una versión de evaluación del software Network Development Kit (NDK) que correrá sobre el DSP OMAP-L137.

- *bios_setuplinux_5_33_05.bin/bios_setupwin32_5_33_05.exe*: Más instaladores de DSP/BIOS para estaciones Linux o Windows respectivamente.

- *xdctools_setuplinux_3_10_05_61.bin/xdctools_setupwin32_3_10_05_61.exe*: Se trata de instaladores de herramientas XDC según la estación que corresponda. Las herramientas XDC son necesarias para construir varios sofwares, incluyendo los Codec Engine, componentes de Framework, Dsplink, etc.

- *ti_cgt_c6000_6.1.9_setup_linux_x86.bin/TI_CGT_C6000_6.1.9_setup.exe*: Ambos binarios instalan, donde corresponda, las herramientas de generación de código TI, necesarias por si no se tienen las últimas versiones de los generadores de código (también se pueden descargar desde la página de Texas Instruments).

- *BIOSUSB_01_00_00_Setup.exe*: El paquete BIOSUSB (basado en JUNGO USB stack) provee la capacidad de incorporar funcionalidades de host o dispositivos a través de USB por la plataforma de dispositivo *embeddedC6747/OMAPL137* (no usadas en este proyecto).
- *mvl_5_0_0801921_demo_sys_setuplinux.bin*: En este caso se trata de un instalador para un host Linux que instala una versión de demostración de herramientas de sistemas del MontaVista Professional Edition v5.0 y un file system.

NOTA: TI recomienda escoger entre los ARM Linux drivers o los DSP BIOS Drivers para ejecutar pero no ambos al mismo tiempo.

- Por último, se debe instalar el CCStudio (Windows workstation).

Aquí se tienen dispuestas todas la herramientas para el óptimo desarrollo de nuestra tarea sobre le placa. Lo siguiente es empezar a transferir archivos y configuraciones a la EVM.

A.2. User Boot Loader (UBL) y Universal BootLoader (u-boot)

En esta sección se detalla como cargar el software de booteo (*ubl*), tanto del ARM como de el DSP, en la memoria SpiFlash0 de la EVM así como también el software *u-boot*.

Es necesario aclarar que el archivo *u-boot.bin* adjunto con la versión I de la EVM, con la que se cuenta para el proyecto, contiene un error de fábrica que no permite detectar la memoria SpiFlash0. Por lo tanto, resulta necesario descargar la última versión del archivo en cuestión [15].

Debe descargarse también desde dicha fuente el paquete *dsp_spi_flash_writer* donde se podrá encontrar el archivo *ccsv3.3/debug/spiflash_writer_arm.out*. Este archivo es el programa que permitirá escribir en la memoria SpiFlash0 los archivos *dsp-spi-ais.bin* (ubl del DSP) y *ubl-spi.bin* (ubl del ARM), que pueden obtenerse en la carpeta de instalación del *PSP* (*Platform Software Package*). Además permitirá cargar también el archivo *u-boot.bin* el cual, como se explicó anteriormente, debe ser descargado en su versión mas reciente.

NOTA: Los archivos descriptos anteriormente se pueden encontrar en la carpeta de instalación del PSP bajo el siguiente path: *REL_LSP_02_20_#_#/PSP_02_20_#_#/bin/*

Una vez que se cuenta con los archivos necesarios, se debe conectar la placa mediante puerto USB (*Embed USB*) con el switch *sw2* en 1111x [19] y ejecutar el software CSSv3.3 configurado adecuadamente como se detalla en la sección correspondiente de este mismo informe.

En la siguiente figura A.1 se puede observar una ventana de CSSv3.3 con la EVM conectada y lista para su configuración.

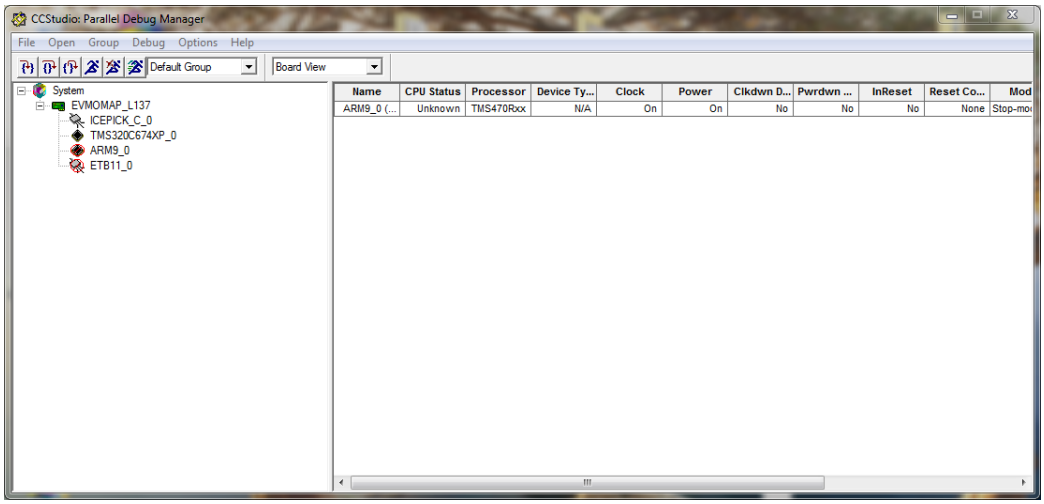


Figura A.1: Vista de CCSv3.3 con EVM conectada. Para conectar la misma, luego de abrir el CCS, ir a *open*, *open TMS320C673X...* o *open ARM9...* según el núcleo donde se quiera trabajar

En primer lugar se conecta el DSP (*TMS320..*), luego el GPP (*ARM9*) y por último se procede a abrir el diálogo con el GPP (click con boton derecho del mouse sobre el GPP *ARM9_0*). Esta última acción permite que se cargue en GPP el programa *spiflash_writer_arm.out* el que permitirá ejecutar los comandos que cargan los archivos *.bin* en la memoria SpiFlash0, dicha ventana se puede observar en la figura A.2.

Una vez en este punto, es fácil cargar los archivos necesarios a la SpiFlash0. A continuación se detalla la carga de cada uno de los archivos individualmente.

Cargando UBL del DSP

Ejecutando el comando *dspais* en el cuadro de diálogo de la herramienta A.5, ésta inmediatamente pide el path del archivo *dsp-spi-ais.bin* (*ubl* del DSP). Luego de unos segundos, se terminara de cargar el archivo *.bin* en la memoria SpiFLash0 y la herramienta despliega el mensaje "*Files Matched*.ªl final de la carga.

Cargando UBL del ARM

Nuevamente en el cuadro de diálogoA.5, ejecutando el comando *armubl*, la herramienta pedirá el path del archivo *ubl-spi.bin* (ubl del ARM). Luego de unos segundos el software terminara de cargar el archivo *.bin* en la memoria SPIFLash0 y la consola expide el siguiente mensaje al final de la carga: "*Files Matched*".

Cargando U-BOOT.

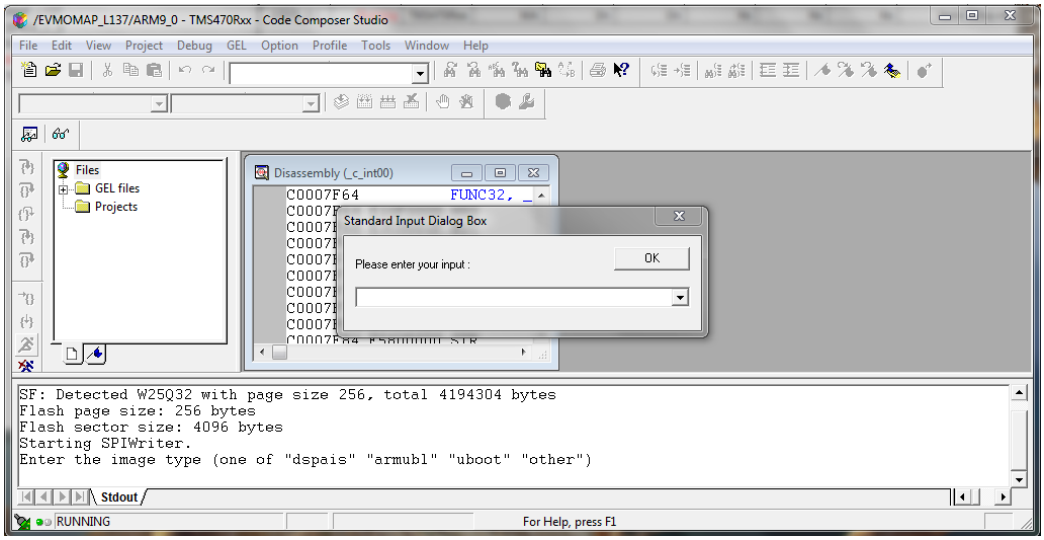


Figura A.2: Vista de CCSv3.3 ejecutando *spiflash_writer_arm.out* sobre el ARM. Ventana de diálogo para cargar *UBLs* de DSP y ARM, y *u-boot*.

Finalmente, ejecutando el comando *uboot* dentro del cuadro de dialogoA.5, la herramienta pide el path del archivo *u-boot.bin*. Una vez ingresada la ruta y luego de unos segundos, se termina de cargar el archivo en la memoria SpiFLash0 mostrando en consola el mensaje *Files Matched*.

A.3. Configuración U-BOOT - Kernel S.O.

Una vez instalado en memoria SPIFlash0 el software de booteo del sistema, se conecta la EVM a un *PC host* mediante el cable UART R232 y el switch *sw2* en la configuración 01010x (boot desde SPI0). Con ayuda del software *putty* (o similar) es posible tener acceso a la plataforma donde mas tarde se instalará el Sistema Operativo de la EVM.

NOTA: La documentación [19] especifica que la configuración de *putty* para acceder a la placa mediante es la siguiente:

```
bps: 115200
Data bits: 8
Parity: No
Flow control: No
Stop bits: 1
```

Ya con la configuraciones y archivos dispuestos sobre la placa, y luego de un reset de la misma, es posible observar el correcto booteo de la EVM con su correspondiente linea de comandos u-boot.

Load del Kernel de Linux

En este punto se tiene la EVM con la mínima configuración que permite su funcionamiento. Para poder continuar el trabajo es necesario cargar el kernel del sistema operativo que corre en la sistema.
En este proyecto se utiliza la versión incluida en el kit de *MontaVista Linux* cuyo kernel se encuentra bajo el path: *REL_LSP_02_20_#_#/PSP_02_20_#_#/bin/* en la carpeta de instalación del *PSP*. Se trata del archivo llamado *uImage*.

NOTA: Como nota importante, para realizar exitosamente los pasos que a continuación se enumeran, es necesario contar con un servidor *tftp* bajo la misma red local que la placa. El objetivo de esto es brindar una manera de transferir el archivo *uImage* a la EVM. Una forma de obtener dicho servidor se describe en la sección referente a la configuración del software en PC.

Para cargar el archivo *uImage* (kernel de linux) en la SPIFlash0 y configurar el File System en la SD card, se deben ejecutar una serie de comandos secuenciales. Los mismos deben ser ingresados sobre el ambiente u-boot del sistema en el orden indicado a continuación.

En primer lugar, y con la placa conectada a la red local mediante un cable UTP cualquiera a través de uno de sus puestos *RJ-45* disponibles, se debe obtener una IP de esta red. Para ello se ejecuta:

```
U-Boot>setenv autoload no
U-Boot>dhcp
```

Como se mencionó anteriormente, es necesario configurar de alguna manera la dirección de donde obtener el archivo *uImage*. Si como se dijo anteriormente, se configura un servidor *tftp* dentro de la red local con la IP 192.168.1.44, se debe ingresar la dirección como se muestra:

```
U-Boot> setenv serverip 192.168.1.44
```

Luego se setea el nombre del archivo kernel de linux disponible:

```
U-Boot> setenv bootfile uImage
```

Ahora se selecciona la memoria SPIFlash0, destino dentro del sistema EVM:

```
U-Boot>sf probe 0
```

A estas alturas, se esta en condiciones de iniciar la descarga del archivo a la EVM, por lo que se procede a ello mediante el siguiente comando (en este también se detalla la dirección de memoria RAM donde empieza a descargar el archivo uImage):

```
U-Boot>tftp 0xc0700000 uImage
```

Finalizada la descarga, se tiene el archivo en memoria RAM. Es necesario entonces limpiar un espacio de memoria flash donde guardar el uImage. Se ejecuta el comando:

```
U-Boot>sf erase 0x60000 0x220000
```

Una vez hecho espacio, se esta en condiciones de transferir el archivo *uImage* de RAM al espacio de memoria correspondiente de SPIFlash0, lo cual es posible a través de la orden siguiente:

```
U-Boot>sf write 0xc0700000 0x60000 0x220000
```

Luego se setean los parámetros que *u-boot* usa para bootear desde un sistema de archivos en la *SD Card* (*root=/dev/mmsblk0p1*):

```
U-Boot>setenv bootargs 'console=ttyS2,115200n8 noinitrd rw ip=dhcp root=/dev/mmsblk0p1 rooti
U-Boot>setenv bootcmd 'sf probe 0;sf read 0xc0700000 0x60000 0x220000; bootm 0xc0700000'
```

Finalmente, se guardan las variables de entorno modificadas:

```
U-Boot>saveenv
```

Ejecutando el comando *boot* en la consola, o si se resetea la placa, se debe observar el booteo automático del S.O linux previamente instalado cargado a la *SD Card*.

File System en SD Card

Crear el file system en la SD card no representa gran dificultad y puede hacerse sin problemas siguiendo los pasos descriptos en la Getting Started Guide for OMAP-L137 [24]

A.4. Herramientas de Programación

Luego de salvados los pasos anteriores, se han sentados las bases que posibilitan el desarrollo de programas a un nivel de programación mas alto, en este caso en el lenguaje C.Esta sección, se acerca al lector a las herramientas de desarrollo y programación de la placa OMAP-L137 EVM. Pero antes de ello debemos recordar algunos conceptos acerca de EVM, en particular que se está frente a un sistema Multi-Core. El diseño Multi-Core System on-chip (SoC), permite trabajar simultáneamente con dos o mas chips de igual o diferente arquitectura. En el primero de los casos, la principal ventaja radica en tener el mismo conjunto de instrucciones facilitando el trabajo al repartir carga, mientras que en el segundo caso, el sistema de arquitectura heterogénea permite realizar múltiples tareas paralelas aumentando la velocidad de procesamiento con tareas especificas para cada core. En nuestro caso nos concentraremos en la arquitectura multi-core (ARM+DSP) que es la implementada en el SoC OMAP-L137.

La solución que se provee (*EVM OMAP-L137*), combina el poder del DSP con las capacidad de propósito general de la arquitectura ARM en un solo procesador integrado DSP+ARM. Este procesador integrado combina networking, administración de archivos y features de interfaces de usuario (sistema de control y procesos de aplicaciones) del procesador ARM al correr sistemas operativos (OS) tales como Linux™, Microsoft®, Windows®, Embedded CE y Android™, con el poder del tiempo real y procesamiento intensivo de señales del Core DSP.

Adicionalmente, los diversos periféricos del chip proveen conectividad (EMAC and USB 2.0), interfaces de memoria de alta velocidad (DDR2 y/o DDR3) y una interfaz bus estandar (PCI Express 2.0, I2C, UART, etc.).

Aclarados los puntos anteriores, es intuitivo el hecho de que los usuarios OMAP-L137 son capaces de desarrollar programas tanto para el núcleo ARM como para el DSP, por lo que se necesitarán diferentes plataformas de desarrollo. De todas maneras, se centra el estudio en el desarrollo de la aplicación en DSP, puesto que es este núcleo el encargado del procesamiento principal del sistema IMPETOM (ver siguiente figura A.3).

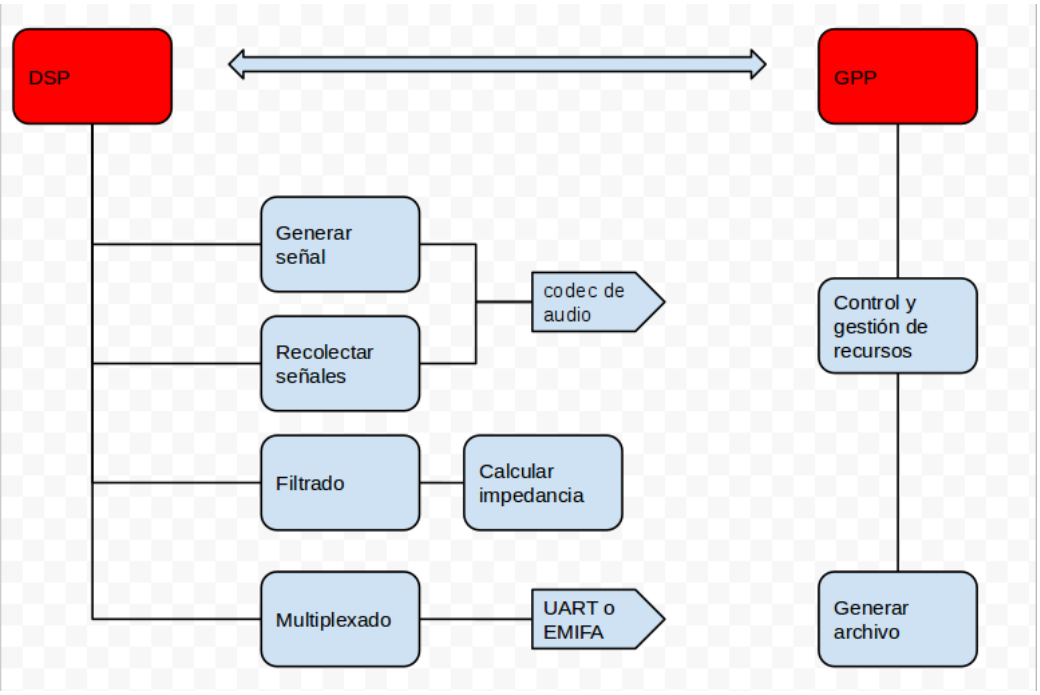


Figura A.3: Diagrama de tareas. Núcleos GPP y DSP

La herramienta predilecta de desarrollo sugerida en la documentación [19] es el CCStudio, que posibilita tanto la programación como el cargar las diferentes rutinas dentro del núcleo DSP. A continuación se ilustra A.4 en diagrama de bloques, el conglomerado de bloques que forman el CCStudio versión 3.3.

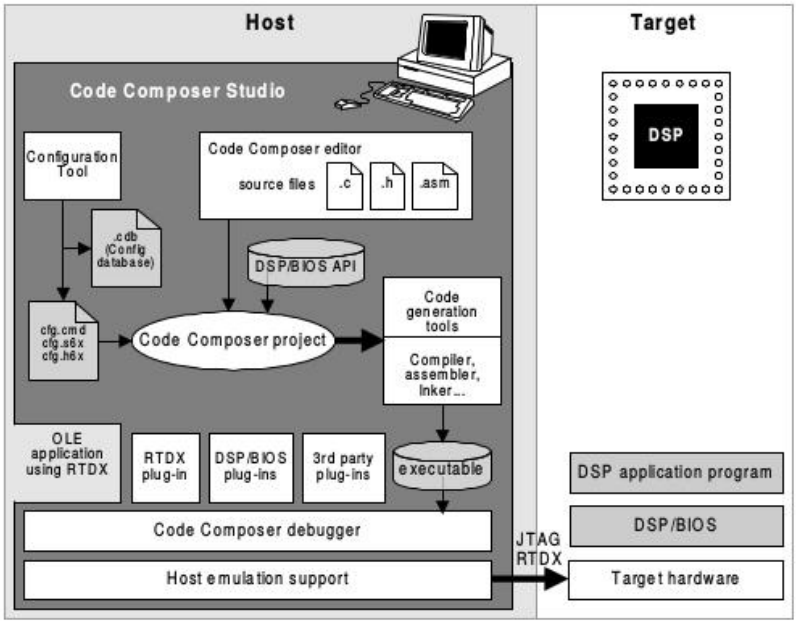


Figura A.4: Diagrama de Bloques CCStudio [16]

A.4.1. Code Compresor Studio

El CCStudio (Code Compresor Studio) es una herramienta mejorada y veloz para el desarrollo de aplicaciones y programas con ayuda de módulos de análisis de señales y pruebas en tiempo real, utilizado para crear aplicaciones al agregar archivos a un proyecto. Estos archivos son usados para construir aplicaciones y pueden incluir otros archivos, tales como archivos fuente en código C, archivos fuente en assembler, archivos de objeto, librerías, archivos Linker Command y archivos incluidos. La anterior figura A.4 muestra también como este programa acopla las diferentes plataformas de desarrollo y archivos mencionados. El conjunto de los módulos mostrados, recorre un flujo de desarrollo como el que se describe a continuación A.5, junto con lista que describe cada uno de los pasos.

- El compilador acepta código fuente en C y produce el código assembler.
- El ensamblador traduce el archivo código en lenguaje assembler a un archivo objeto en lenguaje de máquina. Este archivo orientado a la maquina es basado en Common Object File Format (COFF).
- El optimizador de assembler acepta que se escriba en un código lineal sin considerar la estructura de hilos o la asignación de registros. Este asigna los registros y transforma el código lineal en uno paralelo altamente eficiente.
- Linker combina archivos de objeto en un único módulo ejecutable. Así como crea este módulo, también relocaliza y resuelve referencias externas.

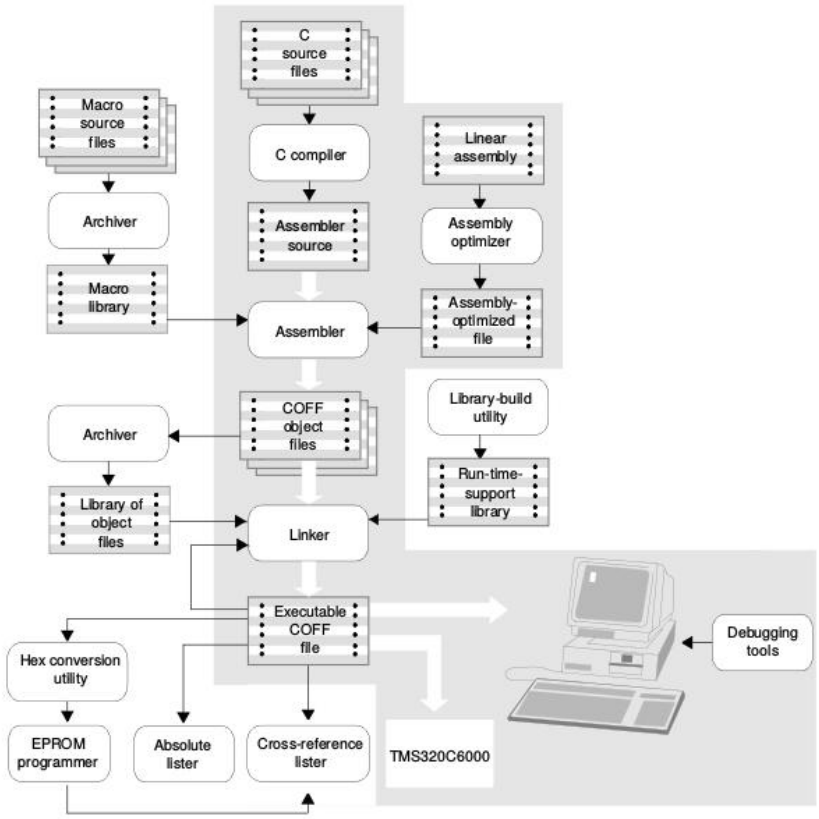


Figura A.5: Diagrama de flujo de iteraciones CCStudio [16]

- El archivador acepta y colecta un grupo de archivos en un sólo archivo llamado librería. Es posible usar la utilería de construcción de librerías para construir nuestra propia librería customizada en tiempo real.
- La librería de soporte en tiempo real contiene funciones de standar ANSI, utilidades de compilador, funciones de aritmética de punto flotante y funciones de I/O
- Los convertidores de hex convierten los objetos COFF en objetos de formato ASCII-hex, Intel, Motorola-S o Tektronix.
- La lista de referencia cruzada usa los archivos de objeto para crear símbolos de referencia cruzada.

NOTA: El camino de desarrollo para programas en lenguaje C está sombreada. El resto son funciones periféricas que mejora el proceso de desarrollo. Por mas información acerca del manejo del CCStudio consultar [16].

Lamentablemente no alcanza sólo con instalar la herramienta, sino que también es necesario realizar una mínima configuración de esta para poder conectarse a la placa e interpretar los comandos introducidos. Para ello se deben seguir los siguientes pasos:

- Conectarse a CCStudio Setup (antes haber instalado controladores para la conexión a través de USB), y conectar el EVM (DSP + ARM) A.1.

- Una vez conectado, hacer click derecho en el dispositivo que indica el DSP (TMS320...) y seleccionar *Open*. Se despliega un nuevo cuadro, correspondiente al CCStudio v3.3 donde se cargaran luego los programas.

- En el menú, ir a la parte de configuración y cargar los archivos de que interpretan los comandos en cada uno de los chips, en este caso al estar conectados al DSP se debe cargar el evmomapl137_arm.gel y evmomapl137_dsp.gel

NOTA: Los archivos con terminación gel (General Extension Language) corresponden a un estandar que es usado por el CCS debugger.

- Realizar idéntico procedimiento a los items 2 y 3 para el ARM.

Recién a estas alturas, y finalizado este último setup, se tienen a disposición todas la herramientas operativas.

A.4.2. Entorno de Programación

Como se mencionó en la sección *Herramientas de Programación*, el lenguaje fuente utilizado como intérprete es C y para programar el OMAP-L137 es necesario del CCStudio. El Code Composer Studio permite de una manera muy sencilla integrar diferentes programas (Código fuente en C, Asembler, librerías, etc) en un único objeto a cargar en el chip, un archivo project (pjt)Ṅo se ahonda acerca de como programar en C, sólo se indica el procedimiento.

- Codigo C El lenguaje utilizado para desarrollar aplicaciones y programas es el C. Para ello el CCStudio provee un entorno gráfico que permite escribir todas las instrucciones.

- Archivo .cmd Este archivo es un intérprete de las instrucciones del programa en C, que traduce las mismas a lenguaje de máquina (Asembler).

- Archivo .lib Una vez cargados todos los archivos en el mismo "Project", se procede a ensamblar (opción *build*) todos sus componentes para luego cargarlos al chip. $Project, BuilAll$. Si la operación transcurrió sin inconvenientes se muestra una ventana de logs donde indica errores y advertencias.

Luego de pasar por todos estos pasos, se tiene el archivo final pronto para cargar en la placa. El proceder es tan sencillo como sigue: Project, Load Project. Inmediatamente se despliega una ventana donde muestra le evolución del proceso y con ello el resultado del programa, un string "*hola mundo*", un sonido, etc.

A.5. DSPLINK

Dada la importancia de este elemento, es necesario dedicarle al menos un sección diferencial. El DSP/BIOS LINK o DSPLINK es el software base utilizado para la comunicación entre procesadores. Proporciona una API genérica que permite la conexión entre el GPP y DSP desde las aplicaciones. Esta facilidad elimina la necesidad de desarrollar tal relación desde cero y permite centrarse solo en el desarrollo de aplicaciones [20].

Se describe a continuación como instalar y correr ejemplos para OMAP-L137 utilizando DSPLink

A.5.1. Procedimiento de construcción del DSPLINK

Configuración del entorno de compilación
Se deben configurar las variables de entorno necesarias:

- DSPLINK - Raíz de instalación del DSPLINK
- PATH - Añade la ruta para incluir scripts proporcionados en la instalación.

Archivo de Distribución del Sistema Operativo

Algunos valores de configuración dentro de los archivo de distribución se deben ajustar al entorno de creación del usuario. Los archivos a modificar son:

- *$DSPLINK/make/Linux/davinci_mvlpro5.0.mk*
- *$DSPLINK/make/DspBios/c64xxp_5.xx_linux.mk*

Dentro de estos archivos los valores que pueden necesitar ser modificados son:

- BASE_BUILDOS
- BASE_INSTALL
- BASE_SABIOS
- BASE_CGTOOLS

Archivo de Configuración de Herramientas del Sistema

El DSPLINK make configura las herramientas y llamadas del sistema mediante un archivo especifico llamado systools.mk,el cual se encuentra en *DSPLINK/make/¡$(GPPOS)—$(DSI*

Dentro de estos archivos los valores que pueden necesitar ser modificados son:

- BASE_PERL

Configuración del DSPLINK para OMAP-L137

Este comando configura varios parámetros como la plataforma y GPP OS entre otros. Para nuestro sistema, el comando a ejecutar es el siguiente:

```
perl $DSPLINK/config/bin/dsplinkcfg.pl --platform=OMAPL1XX --nodsp=1 --dspcfg_0=OMAPL1XXGEMS
```

Luego se deben realizar dos pasos mas antes de que la configuración este completa. Los cuales preparan los paquetes XDC de las *XDC-tools*.

```
cd $DSPLINK/dsp
$XDC_INSTALL_DIR/xdc clean
$XDC_INSTALL_DIR/xdc .interfaces
cd $DSPLINK/gpp
$(XDC_INSTALL_DIR)/xdc clean
$(XDC_INSTALL_DIR)/xdc .interfaces
```

Se debe configurar Linux para OMA-PL137 con herramientas uclibc, en el comando se usa la configuración para el DA8xx ya que su plataforma es similar al OMAP-L137

```
make ARCH=arm CROSS_COMPILE=arm_v5t_le-distclean
postamk ARCH=arm CROSS_COMPILE=arm_v5t_le- da830_omapl137_defconfig
```

Terminado esto se puede compliar de manera correcta los archivos fuente y ejemplos, ubicándose en la carpeta donde están los mismos y ejecutando el comando

```
make - s [debug | release]
```

A.6. Recapitulo

Transcurrido el pasado capítulo, es posible tener una idea de la amplia variedad de elementos y aplicaciones necesarios para la configuración y desarrollo del sistemas embebido en la Evaluation Board (EVM) OMAP-L137. Lo anterior pretende establecer los paquetes de software y configuraciones mínimas necesarias para el correcto desarrollo de la tarea, sin necesidad de recurrir a la documentación provista por el fabricante, la cual es dispersa, de difícil acceso (sistemas propietarios), poco intuitiva y poco detallada (características que nos presentaron una barrera importante a la hora del desarrollar el proyecto). Haciendo un recapítulo de la sección, y cumplidos todos los requerimientos de sistema y software necesarios, los pasos a seguir son los siguientes:

- Cargar los User Boot Loader (UBL) en cada núcleo (ARM y DSP) y Universal BootLoader (u-boot), esenciales para luego cargar las rutinas, O.S., y demás elementos necesarios (DSP-Link por ejemplo).
- Cargar el Kernel de Linux en el núcleo ARM.
- *Sd Card* con Sistema Operativo
- Configurar la herramienta de programación CCStudio para poder acceder al núcleo DSP.
- Configurar el DSP-Link para permitir la comunicación entre los núcleos.

Esta página ha sido intencionalmente dejada en blanco.

Apéndice B

Software *IMPETOM-B*

Incorporar a la implementación ambos núcleos requiere extras que permitan la comunicación, no basta con tener el software específico en cada uno, sino que también debe existir elementos que hagan posible la transferencia de datos, y para ello, la comunicación entre los dos núcleos se implementa a través de un fragmento de memoria compartida específica.

Si bien existen tres formas de diferentes de implementar esta comunicación (denominados buffer de comunicación estático de mínimo controlB.1, Buffer de comunicación dinámico de mínimo controlB.2 y múltiples BuffersB.3), centraremos el estudio en aquel cuya implementación es más sencilla y de mínimo control para las necesidades de *IMPETOM-B*. Dado que los datos intercambiados, incluido el programa a enviar al DSP, tienen un tamaño fijo, basta con configurar el sistema con un buffer de comunicación estático de mínimo control (ver siguiente figura B.1).

De todas maneras, disponer de memoria compartida tampoco es suficiente para la comunicación entre núcleos, se necesitan de aplicaciones y funciones que manejen el intercambio y la cola de mensajes desde y hacia los lados, permitiendo implementar el protocolo de comunicación. Entonces existen un conjunto de subprogramas configurables en ambos lados del sistema (GPP y DSP). Este conjunto consta tan solo de dos procesos *main*: una aplicación Linux que carga y inicia el DSP usando la llamada de sistema *PROC*, así como intercambiar mensajes y datos mediante la cola de mensajes (enviados vía función *MSGQ*) que se implementa en la región de memoria compartida (configurable a través de la función *POOL*). Y una aplicación DSP/BIOS, que comienza la ejecución luego de ser cargada por la aplicación Linux, con la tarea *TSK* que recibe y envía de regreso los mensajes usando también estructuras *MSGQ* y *POOL*.

Durante el siguiente punto se enumerara y describe más en detalle cada una de estos procesos residentes tanto en el DSP como en el GPP.

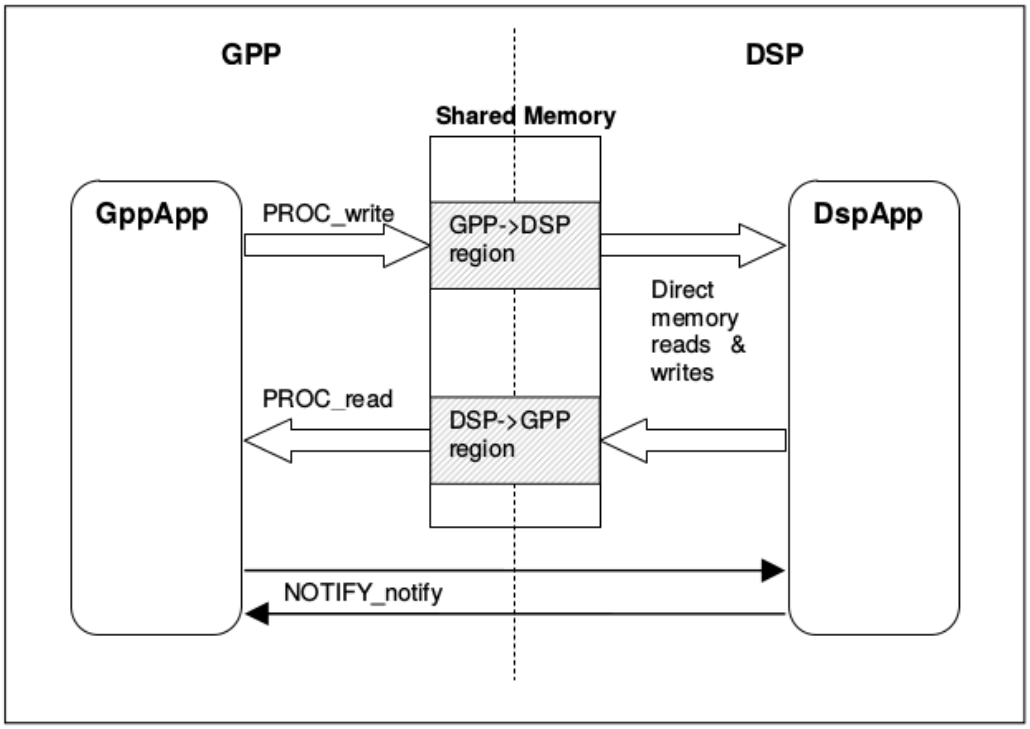


Figura B.1: Comunicación entre DSP y GPP (en memoria compartida) con configuración de memoria mínimo control. Extraído de documentación [18]

B.1. GPP

Dentro del core GPP (ARM) se puede encontrar un conjunto de archivos que componen a la aplicación que se ejecuta en él bajo el sistema operativo Montavista Linux la cual resulta esencial para la gestión y el control del software de procesamiento de señales que corre en DSP y posibilita la independencia del sistema de la PC. Para la realización de estas aplicaciones se partió del software ejemplo descrito en la documentación del dsplink *user guide* [18] y en particular en la aplicación *message*.

Del lado del GPP entonces, la aplicación ejemplo consta de 3 archivos fundamentales:

- *main.c* realiza la interfaz con Linux y simplemente realiza la comprobación de errores inicial de los parámetros de la línea de comandos, este es un archivo que no será necesario modificarlo ya que simplemente es el punto de entrada.

- *system_os.c* contiene una capa de abstracción entre el código y el sistema operativo Linux. De esta forma que puede ser fácilmente portado a otros sistemas operativos (Windows CE, etc), dado que este es un archivo de interfaz no será necesario modificarlo para la confección de nuestro sistema.

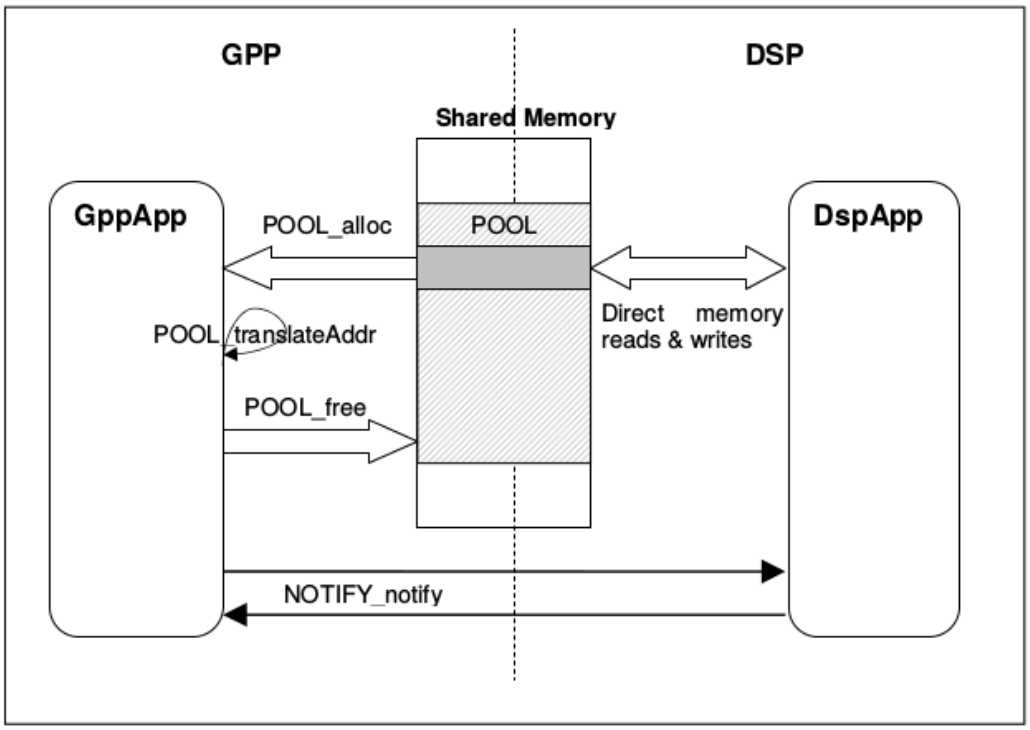


Figura B.2: Comunicación entre DSP y GPP (en memoria compartida) implementando buffer dinámico. Extraído de documentación [18]

- *message.c* contiene la mayor parte de la aplicación y tiene la siguiente estructura:

  message_main(): Punto de entrada de la aplicación. Contiene la secuencia de llamadas a funciones de este módulo, así como la verificación de datos.

  message_Create(): La función de asignación y de arranque, la parte más compleja del código. Configura el DSP a través PROC_setup() y PROC_attach(). Crea la piscina de memoria utilizando POOL_open() y abre la cola de mensajes de control entre los dos núcleos. Esta función es genérica para cualquier aplicación que requiera comunicación entre DSP y GPP y a nuestros propósitos no necesita ser modificada.

  Función *message_Execute()*: El cuerpo principal de la ejecución. Es la función encargada de realizar el control de lo que se hace con los mensajes, que enviar y que hacer con lo que recibe desde el DSP. Esta función es la que modificaremos para adaptar el software a nuestras necesidades.

  message_Delete (): Cierra el transporte a distancia y se detiene la ejecución en DSP. También cierra el MSGQ y elimina el objeto PROC de la memoria. Esta secuencia es importante, ya que debe suceder en orden inverso a message_Create() para liberar correctamente la memoria.

Como se mencionó al principio, esta implementación del lado del GPP tiene su imagen del lado del DSP para asegurar la correcta comunicación.

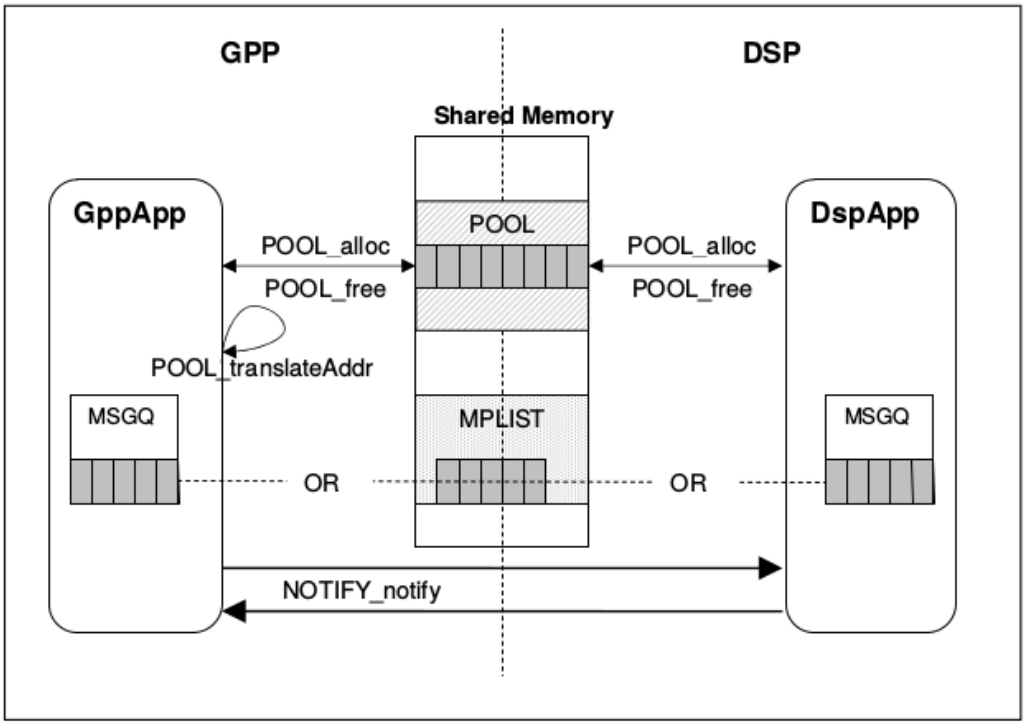


Figura B.3: Comunicación entre DSP y GPP (en memoria compartida) implementando configuración de buffers dinámicos. Extraído de documentación [18]

B.2. DSP

Del lado del DSP se tiene los siguientes tres archivos fuentes [18]:

- *main.c* contiene las llamadas de las funciones del DSP que inicializan el DSP/Link (*DSPLINK_init()*), el BSL (*EVMOMAPL137_init()*) y crea la aplicación tarea (*tskMessage*) que contiene la siguiente secuencia de aplicaciones.

- *helloDSP_config.c* contiene el pool de memoria y la configuración de las colas de *Messahe*. La configuración de este archivo debe ser la misma que la implementada por el archivo *helloDSP.c* para el core ARM.

- *tskMessage.c* contiene la mayor parte de las aplicaciones:

  La función *TSKMESSAGE_create()*, que asigna y empieza el programa. Asignar memoria para el mecanismo de transporte físico del DSPLink vía *MEM_calloc()* y empezar el semáforo de cola de *message* usando *SEM_new()*. La secuencia es similar a lo que ocurre de lado del ARM, se crea la cola de mensaje (*MSGQ_open()*) y setea el manejador de mensajes de errores (*MSGQ_setErrorHandler()*). Por último, se espera por la primer señal de sync desde el ARM a través de *MSGQ_locate()*.

TSKMESSAGE_execute() es el cuerpo de ejecución main. Envía el primer mensaje de vuelta al ARM (*MSG_put()*) y corre en un loop al mismo tiempo que pasa por las aplicaciones. En este loop corren varias aplicaciones y se escribe al buffer de cola de mensajes. Al final del loop, se regresa un el último mensaje al ARM.

Función *TSKMESSAGE_delete()* que desvincula el programa liberando la cola de mensajes (para evitar que prosigan las comunicaciones) y la cierra. Por último libera la memoria usada para el transporte y devuelve a la aplicación *main* la tarea *(tskMessage()*. Esta tarea finaliza y retorna al loop idle del DSP/BIOS.

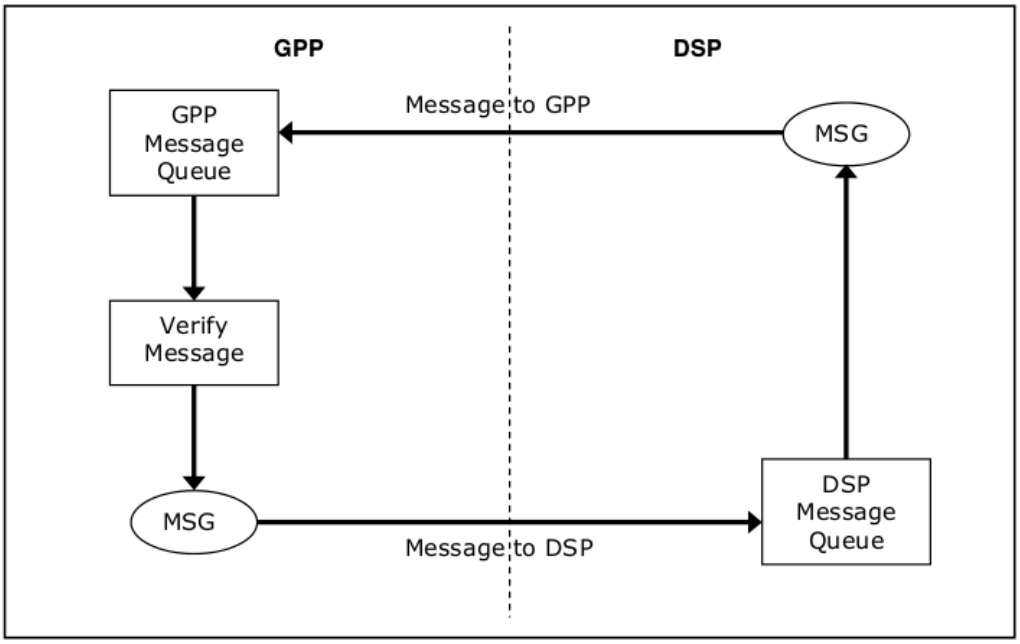


Figura B.4: Esquemático de flujo de mensajes entre GPP y DSP. Ejemplo de aplicación MESSAGE. Extraído de la documentación [18]

Esta página ha sido intencionalmente dejada en blanco.

Apéndice C

Configuración de Registros del McASP

Como ya se mencionó antes, el McASP es un periférico de propósito general que funciona como puerto de audio serial, que realiza operaciones de transmisión y recepción de tramas de datos a través de hasta 16 "canales". Cada uno de estos "canales" se denomina *serializer* y constan de un registro de corrimiento (*shift register* llamado *XRSR*), un buffer de datos (*XRBUF*), un registro de control (*SRCTL*), y la lógica necesaria para la alineación de las diferentes opciones disponibles. Si bien los *serializer* son fundamentales en lo que a la comunicación se refiere, para utilizar el McASP es necesario conocer también el resto de los registros que permiten su configuración. A continuación se detallan dichos registros partiendo de la configuración base realizada en el software test (*aic3106 test*) incluido con el kit de la placa.

C.1. Configuración base

En el capitulo 5 se detalla la utilización de la función *aic3106 loop linein()* como base, y ya que la comunicación del códec y el DSP se realiza mediante un puerto McASP, también se utiliza dicha función como base para el seteo de los registros del McASP de la rutina *Grabar()* del proyecto *IMPETOM-B*.

Inicialización

Para inicializar el McASP tanto en trasmisión como recepción, se debe seguir un orden en el seteo de los registros.

1- *Reseteo Global*

En primer lugar, se resetea el McASP mediante los registros control global (*Global Control GBLCTL*), control global de recepción (*Global Reception Control RGBLCTL*), y control global de trasmisión (*Global Transmission Control XGBLCTL*).

```
Registro GBLCTL  -> 0 // Reset
Registro RGBLCTL -> 0 // Reset RX
```

```
Registro XGBLCTL -> 0 // Reset TX
```

2- *Recepción*

En segundo lugar, se setean los registros para la recepción mediante los cuales se configura: mascara de datos, formato del dato, configuración del reloj de alta frecuencia, reloj de bit, señal de control, número de slots activos y habilitación de interrupciones.

```
Registro RMASK  -> 0xffffffff; // No padding used
Registro RFMT  -> 0x00008078; // MSB 16bit, 1-delay, no pad, CFGBus
Registro AFSRCTL  -> 0x00000112; // 2TDM, 1bit Rising, INTERNAL FS, word
Registro ACLKRCTL -> 0x000000AF; // Rising INTERNAL CLK,(from tx side)
Registro AHCLKRCTL -> 0x00000000; // INT CLK (from tx side)
Registro RTDM  -> 0x00000003; // Slots 0,1
Registro RINTCTL  -> 0x00000000; // Not used
Registro RCLKCHK -> 0x00FF0008; // 255-MAX 0-MIN, div-by-256
```

3- *Trasmisión*

La transmisión se setea de manera similar a la recepción, teniendo la posibilidad en el registro de configuración del reloj de trasmisión, poder setear el sincronismo o no entre transmisión y recepción.

```
Registro XMASK -> 0xffffffff; // No padding used
Registro XFMT     -> 0x00008078; // MSB 16bit, 1-delay, no pad, CFGBus
Registro AFSXCTL  -> 0x00000112; // 2TDM, 1bit Rising edge INTERNAL FS, word
Registro ACLKXCTL -> 0x000000AF; // Synchronous, Rising INTERNAL CLK, div-by-16
Registro AHCLKXCTL -> 0x00000000; // EXT CLK
Registro XTDM      -> 0x00000003; // Slots 0,1
Registro XINTCTL   -> 0x00000000; // Not used
Registro XCLKCHK   -> 0x00FF0008; // 255-MAX 0-MIN, div-by-256
```

4- *Serializadores*

Se configuran cuales serializadores son salidas o entradas de datos.

```
 Registro SRCTL5 -> 0x000D;     // MCASP1.AXR1[5] --> DIN
 Registro SRCTL0  -> 0x000E;     // MCASP1.AXR1[0] <-- DOUT
```

Los registros SRCTL[n] son de gran importancia ya que en estos se indica cuando los registros de trasmisión ($XBUF$) y recepción ($RBUF$) están prontos para ser llenados o leídos respectivamente. De no leerse o completarse estos registros en un tiempo adecuado puede producirse una sobreescritura del dato en el buffer de recepción y/o una omisión de datos a trasmitir, perdiéndose información en la comunicación.

Con los siguientes registros se configura para determinar si los pines son de propósito McASP o GPIO, y si son entrada o salida.

```
Registro PFUNC  -> 0;          // All MCASPs
Registro PDIR -> 0x14000020; // All inputs except AXR0[5], ACLKX1, AFSX1
```

Por último se resetean las interrupciones.

```
Registro XSTAT  -> 0x0000ffff;        // Clear all
Registro RSTAT  -> 0x0000ffff;        // Clear all
```

C.1.1. Recapitulo

El McASP es capaz de realizar una comunicación controlada y confiable entre el códec de audio y el DSP, mediante la configuración de una comunicación en formato TDM, 2 slots, sincronizada y de 16 bits de dato tanto para entrada como para salida.

Esta página ha sido intencionalmente dejada en blanco.

Apéndice D

Configuración de Registros del Codec Audio

Recordando lo mencionado en capítulos centrales, el periférico *AIC3106* [22] es un codec (codificador-decodificador) de audio estéreo de baja potencia, con amplificadores de ganancia ajustable, múltiples inputs y outputs programables, control digital de pre-amplificadores estéreo, control automático de ganancia (AGC) con capacidad de mezclar entre múltiples entradas analógicas y filtros programables para eliminar el ruido.

Para utilizar los canales de entrada analógica del codec de audio como herramienta adquisidora de voltajes, es necesario conocer los registros que permiten su manipulación. En este capitulo se analizan dichos registros fundamentales partiendo siempre de la configuración base realizada en el software test (*aic3106_test*) incluido con el kit de la placa.

Configuración base

En el anexo *Software Test en DSP*, se concluye la utilización de la rutina *aic3106_loop_linein()* como base a partir de la cual se confecciona la función *Grabar()* del proyecto. La configuración efectuada en esta rutina (incluido en el kit de software de la EVM OMAPL137) es la siguiente:

Frecuencia de muestreo

Para trabajar a la máxima frecuencia posible y con el mayor número de canales disponibles, es necesario configurar la velocidad de muestreo a 48kHz. Los registros involucrados aquí son el 3, 4, 5, 6 y 7.

```
Registros de configuracion del PLL
 Registro 3  -> 0x22 [PLL=OFF][Q=4][P=2]
 Registro 4  -> 0x20 [J=8]
 Registro 5  -> 0x6E [D=7075]
```

```
 Registro 6  -> 0x23 [D=7075]

Codec Datapath Setup
 Registro 7  -> 0x0A [FS=48kHz][LeftDAC=LEFT][RightDAC=RIGHT]
```

Modo de Comunicación con el Periférico McASP

Se configura el codec como esclavo, además la transferencia se realizará bajo el protocolo I2S con palabras de 16 bits. No se le agrega offset a los datos.

```
Audio Interface Control A, B y C
 Registro 8  -> 0x00 [BCLK=Slave][MCLK=Slave]
 Registro 9  -> 0x00 [I2S mode][16 bit]
 Registro 10 -> 0x00 [Data offset=0]
```

Conexión y Ganancia de los Canales

Se opta por utilizar los canales izquierda y derecha de la entrada *line_in* para realizar las medidas, y el canal *headphone* como salida para ser utilizado de DDS. Los registros que realizan las conexiones internas en el codec, así como también, los que configuran ganancias de las etapas entrada o salida deja se muestran seguidamente.

```
Left and Right ADC PGA Gain
 Registro 15 -> 0x00 [Mute=OFF]
 Registro 16 -> 0x00 [Mute=OFF]

LINE1L to Left ADC and LINE1R to Right ADC
 Registro 19 -> 0x04 [SingleEnd][Gain=0dB][Power=ON][SoftStep=OncePerFS]
 Registro 22 -> 0x04 [SingleEnd][Gain=0dB][Power=ON][SoftStep=OncePerFS]

Left AGC B and Right AGC B
 Registro 27 -> 0x00 [OFF]
 Registro 30 -> 0x00 [OFF]

DAC Power & Output Dvr
 Registro 37 -> 0xE0 [LeftDAC=ON][RightDAC=ON][HPLCOM=SingleEnd]

High Power Output Dvr
 Registro 38 -> 0x10 [HPRCOM=SingleEnd][ShortCircuit=OFF]

Left DAC Digital Volume and Right DAC Digital Volume
 Registro 43 -> 0x00 [Mute=OFF][Gain=0dB]
 Registro 44 -> 0x00 [Mute=OFF][Gain=0dB]

DAC_L1 to HPLOUT Volume
```

```
 Registro 47 -> 0x80 [Routed]

HPLOUT and HPLCOM Output
 Registro 51 -> 0x09 [Mute=OFF][Power=ON]
 Registro 58 -> 0x00 []

DAC_R1 to HPROUT Volume
 Registro 64 -> 0x80 [Routed]

HPROUT and HPRCOM Output
 Registro 65 -> 0x09 [Mute=OFF][Power=ON]
 Registro 72 -> 0x00 []

DAC_L1 to LEFT_LOP/M Volume
 Registro 82 -> 0x80 [Routed]

LINE2R to LEFT_LOP/M Volume
 Registro 86 -> 0x09 []

DAC_R1 to RIGHT_LOP/M Volume
 Registro 92 -> 0x80 [Routed]

RIGHT_LOP/M Output
 Registro 93 -> 0x09 [Mute=OFF][Power=ON]
```

Configuración del Reloj del Sistema

Finalmente, se configura el clock del sistema con los siguientes registros:

```
GPIO Control Register B
 Registro 101-> 0x01 [CODEC_CLKIN = CLKDIV_OUT]

Clock Generation Control
 Registro 102-> 0x00 [PLLCLK_IN and CLKDIV_IN use MCLK]
```

Esta página ha sido intencionalmente dejada en blanco.

Apéndice E

Software test en DSP - Digital Signal Processor

Como se mencionó en uno de los capítulos centrales, el *OMAP-L137* combina dos tipos de Cores, el GPP con arquitectura tipo ARM encargado de tareas de propósito general y el Core DSP cuya principal función es el procesamiento intensivo de señales en tiempo real. Se desprende de aquí entonces, que el encargado principal del procesamiento de las señales que provienen de los electrodos alrededor del tórax, el manejo y generación de la señal de control de la fuente de corriente, debe ser el DSP. Sin embargo, comprender el funcionamiento lógico que implementa el DSP no es tarea sencilla dado que no sólo debe implementar la generación y medida de señales, sino que también debe inicializar todos los periféricos que el DSP utiliza para la comunicación (librerías, entradas, salidas, Codec, etc). El anexo describe los primeros pasos para la comprensión del funcionamiento del DSP, a través del estudio de los test que provee el kit de instalación y que son utilizados para chequear del funcionamiento general de esta.

E.1. Generalidades

Los casos que se describen a continuación, corresponden a tests para le codec de audio *AIC3106* (estos pueden ser localizados en el directorio de test del software kit que incluye el ccStudio en *../boards/EVOMAPL137/DSP/tests*). La estructura de las rutinas de prueba de cada test, siguen la misma lógica de programación: inicialización, implementación de funciones y liberación de recursos. Esto es producto a que se utilizan los mismos periféricos, bus y protocolos. A continuación, se divide el estudio de los test en estas 3 parte.

E.2. Tests de Codec *AIC3106*

Además del obvio uso del codec de audio en las pruebas, también se deben configurar otros elementos necesarios para establecer y coordinar la comunicación con el exterior, como lo son el McASP (MultiChanel Audio Serial Port) y buffers.

No se ahonda demasiado en el funcionamiento de cada uno de estos elementos pero si se pretende una aproximación y comprensión de la mecánica de los test, enumerando sólo aquello que se debe tener en cuenta para un correcto funcionamiento.

Inicialización

Este punto es fundamental a la hora de activa y preparar lo necesario para el correcto funcionamiento de los elementos. Dado que las funciones utilizadas por el test del Codec tienen similar estructura, se realiza una descripción general de la inicialización de las funciones *loop_micIn, loop_lineOut, loop_lineIn* y *headPhone*.

Lo primero que se inicializa es el objeto DSP mediante *EVMOMAPL137_init()*.

El sistema inicializa el Codec a través de la función de sistema *EVMOMAPL137_AIC3106_rset(**#reg,valor**)* provista desde las librerías correspondientes. **#reg** puede ser directamente el número de registro o una variable donde se pase el mismo, y **valor**, el seteo del registro. Por ejemplo *EVMOMAPL137_AIC3106_rset(**AIC3106_RESET,0x80**)* o *EVMOMAPL137_AIC3106_rset(**7,0x6E**)*.

Los registros mínimos necesarios a ser seteados son los siguientes:

- Resetar el Codec.
- Configuración de PLLs.
- Setup de ruta de los datos.
- Seteo de las interfaces de control.
- Ganancias de ADCs y DACs.
- Interconexión de rutas.
- AGCs izquierdo y derecho.
- Control de registros para GPIO.
- Clock.

Lo primero a configurar en este punto es el nuevo objeto McASP mediante *mcasp=&MCASP_MODULE_1*. Luego de obtenido esto, ya se está en condiciones de configurar cada uno de los elementos correspondientes.

Para el seteos de inicialización del periférico McASP, se procede de distinta forma a lo visto para el Codec, ya que sus elementos son inicializados de forma directa a través de *mcasp-¿regs-¿nombre_registro = valor*, donde **nombre_registro** es el nombre del registro propiamente dicho que se quiere setear y al que se el quiere pasar el nuevo **valor**. El orden de inicialización se enumera como sigue:

- Reset del registro de control global.
- Reseteo de la configuración para transmisión (TX) y recepción (RX)
- Selección de pins para datos de entrada y salida (MCASP1.AXR1[5] como DataIN y MCASP1.AXR1[0] como DataOUT por ejemplo).

Es importante señalar además que alguno de los seteos (Codec o McASP) deben realizarse post cumplir condiciones. Por ejemplo, en algunos casos debe esperarse a que el bus o registro estén preparado antes de ser configurados *while(mcasp-¿regs-¿XGBLCTL AND GBLCTL_XHCLKRST ON)* set otro registro.

Implementación

Una vez inicializados los elementos, se tienen los periféricos preparados para su utilización. Se describen a partir de aquí, los cuerpo de las funciones que hace posible la implementación de los tests

loop_micIn

El principal objetivo de esta rutina es emitir un tono a través de la salida lineOut para luego recuperarlo en la entrada de micrófono. Luego del tramo de inicialización, la rutina *loop_micIn* espera a que el buffer de escritura este pronto para recibir datos a través de un *while*, hasta que el bit 4 del registro *SRCTL5* esté en 1.

Una vez se haya cumplido esta condición, la rutina prosigue hacia un loop que dura aproximadamente 1 segundo (1000 ms), donde se repite la secuencia de programa siguiente:

- Se espera a que el registro del McASP esté pronto para recibir datos (aguarda a que el bit 5 del registro *SRCTL5* esté en 1).
- Escribe en el buffer *XBUF5* una de las 48 muestras correspondientes a una representación de una señal sinusoidal de 1 KHz. Según la configuración de inicialización, el buffer *XBUF5* proveerá la ruta hacia la salida analógica *lineOut*. En este punto es importante reparar en el hecho de mencionar -*epresentación de 48 muestras de una señal sinusoidal...”*, punto que tomara especial interés más adelante en el documento.
- Se aguarda nuevamente, en esta oportunidad por *XBUF0* que indica qué el sistema está pronto para leer (bit 5 del registro *SRCTL0* esté en 1).
- Cumplida la condición anterior, el DSP puede finalmente recuperar la una muestra de la señal desde *XBUF0* para guardarla en una variable de la rutina.
- Se inicia el proceso nuevamente.

Observe que la secuencia anterior se realiza por muestra, incrementando una variable que permite ir cambiándola a medida que avanza el programa, y como se mencionó en algún punto, dura 1000 iteraciones. Para este caso específico de la señal representada de 1 KHz, 1000 muestras representan 1 segundo.

Luego de finalizada toda la secuencia de escritura y lectura, la rutina *loop_micIn* pasa a la parte de cierre de los diferentes componentes de sistema.

loop_lineIn

Este test implementa una rutina prácticamente igual a la que se tiene para *loop_micIn*, con la única excepción de que agrega la posibilidad de emitir y recibir la señal analógica por dos canales, derecho e izquierdo (en estéreo), *lineOut* y *lineIn* respectivamente. Nuevamente se espera a que el registro de salida indique que el sistema está pronto para emitir la señal, se escribe en buffer *XBUF5* la muestra correspondiente. Luego, se vuelve a esperar a que el sistema este listo para recibir datos, y cumplida la condición, lee de *XBUF0* para guardarla en una variable. La implementación de la rutina es exactamente igual a la vista antes, sólo cambia un fragmento de la configuración inicial (de la inicialización), que habilitan los nuevos canales. Para ello, se configuran las ruta y ganancias a través de los registros del Codec entre el número 17 y 25, para que en lugar de recibir las muestras desde *micIn* (que es una entrada mono), se haga desde *lineIn* que tiene la posibilidad de recibir por *lineIn Rigth* y *lineIn Left* (derecha izquierda respectivamente).

Otro detalle importante de la configuración, es que las muestras se toman desde un mismo buffer (*XBUF0*), para luego ser guardadas de manera alternada en dos variables distintas, *derecha* e *izquierda*. Esto es debido al mecanismo TDM (Time Division Multiplexing) que utiliza el McASP para canalizar los datos, y que se estudiará en la sección referente a McASP.

Luego se procede al cierre y reset de los elemento utilizados.

tone_lineOut

La rutina *tone_lineOut* simplemente intenta emitir una señal (tono) tanto por el canal izquierdo así por como el derecho, aprovechando la característica que provee la salida analógica *lineOut*. Para realizarlo, también implementa la misma lógica que lo mostrado antes. Se espera hasta tener habilitado el sistema para escribir la muestra en buffer. Luego, dado que también se utiliza el método de transmisión TDM, el sistema colocará alternadamente las muestras en uno u otro canal a pesar de utilizar el mismo buffer.
El cambio de path y ganancias hacia la salida correspondiente, se realiza con el seteo adecuado de los registros 47, 51, 58, 64, 65, 72, 82, 86, 92 y 93 del McASP.

tone_headPhone

El test para emitir un tono a través de la salida analógica *headPhone* repite, nuevamente, las mismas instrucciones que el test utilizado para emitir el tono a través de *lineOut*, y de la misma forma que antes, lo único que se necesita para cambiar la ruta hacia la salida analógica deseada radica en la reconfiguración adecuada de los registros correspondiente.

Cierre de Sistema

El correcto cierre o reset de los periféricos y protocolos utilizados, permite reutilizar los mismos sin necesidad de un reset completo del sistema (de la placa), y es un fragmento de programa que consta de no más de 8 líneas.
Para el caso del McASP, se resetea el protocolo y se vuelve a la página 0 de memoria, mientras que para el Codec de Audio, se vuelven a 0 todos los registros de control *SRCTL_i* y *GBLCTL*.

E.3. Recapitulo

Se han visto como cada uno de los test disponibles permiten emitir señales analógicas y digitales utilizando las salidas *lineOut* o *headPhone*, así como también grabar en variable temporal, un fragmento de señal recibida desde las entradas analógicas del codec de audio *lineIn* o *micIn*. En este sentido, las características que *IMPETOM-B* determinan las siguientes necesidades:

- Canal analógico de salida para emitir una señal de control hacia la fuente de corriente.
- Canales analógicos de entrada para recibir señales desde los 16 electrodos.
- Salida digital para el control de multiplexores y enrutar debidamente, tanto la salida de la fuente de corriente hacia el electrodo indicado, como el electrodo donde se toman las muestras.

En síntesis, es necesario múltiples salidas y múltiples entradas. La configuración para obtener múltiples entradas puede ser tomada desde el test *loop_lineIn* ya que *micIn* provee una salida mono. Para obtener múltiples entradas, la configuración puede ser tomada tanto desde *loop_lineOut* como *headPhone* ya que ambas utilizan canales analógicos en estéreo (derecha e izquierda, dos canales en cada caso), aunque también es provista por la función *loop_lineIn*.

A partir de esto, se tiene entonces un primer prototipo de rutina para implementar las funciones de *IMPETOM-B*. Se puede reciclar la función test *loop_lineIn* que ya provee lo necesario para recibir la información analógica (voltajes) desde las entradas derecha e izquierda y emitir la señal analógica de control de la fuente de corriente a través de *lineOut* (derecha y/o izquierda). Lo único a agregar debe ser la salida de una señal digital para el control del circuito multiplexador.

Esta página ha sido intencionalmente dejada en blanco.

Apéndice F

Circuito auxiliar

F.1. Software Test del Puerto UART

De similar forma que lo mostrado para los test del Codec de audio, el programa para realizar el test del puerto UART también cumple con esa cierta estructura, salvo que en esta oportunidad en lugar de inicialización, implementación y liberación de recursos se tiene tan solo apertura del puerto e implementación del test.

Lo que si se comparte con los test anteriores es el procedimiento de creación de los elementos. En este sentido lo primero es crear el objeto DSP mediante *EVMOMAPL137_init()* dentro de la función *main* al igual que antes y ya con el elemento DSP, se crea el objeto UART, dentro de la función que implementa el test mediante *UART_Handle uart_i* (observe que puede tener tantos objetos *UART_i* como puertos se tengan).

Antes de continuar con la implementación del test, es necesario algunas consideraciones particulares, aunque no muy detalladas, del protocolo utilizado por el puerto UART. Se trata de una trasmisión de datos de forma serie y donde se pueden configurar las velocidades entre 9600 y 115200 baudios.

Dicho esto, lo que prosigue a la creación del elemento *UART_i*, es abrir el mismo a través de la función de sistema *EVMOMAPL137_UART_open(n, velocidad)* donde *n* es el ID del elemento UART y *velocidad* a la que se quiere transmitir.

En este punto ya se está en condiciones de operar con el elemento, se inicia entonces la secuencia de implementación del test del puerto UART. La implementación es serial, en concordancia con el funcionamiento del puerto por la siguiente secuencia se realiza por bit:

- Se espera a que el puerto esté pronto para transmitir los datos, lo que se realiza mediante la función proveída por el sistema *EVMOMAPL137_UART_xmtReady(uart_i)*. Mientras no se cumpla esta condición, se reduce un contador llamado *timeout* cuya función es limitar la espera por el puerto a tiempos aceptables. Si por

algún motivo no se llegase a cumplir esta condición, *timeout* expira y fuerza a la función a devolver el error correspondiente.

- Una vez confirmado que el puerto está listo para transmitir los datos, se ingresan los mismos al tx mediante otra función de sistema *EVMOMAPL137_UART_putChar(uart0, uart_tx[i])*. Esta función selecciona el bit i de la variable (en este caso un buffer uart_tx) para ser colocado en el puerto *uart0*.
- Se resetea el reloj *timeout*.
- Se vuelve a esperar a que el puerto este en condiciones de recibir datos y para ello se censa el mismo con *EVMOMAPL137_UART_rcvReady(uart0)*. De igual manera que visto para esperar a transmitir, recibir utiliza el mismo contador *timeout* para controlar el tiempo de espera, forzando a la salida y el aviso correspondiente en caso de que el mismo expire.
- Una vez se haya pasado exitosamente la espera anterior, se lee el dato correspondiente *EVMOMAPL137_UART_getChar(uart0, &uart_rx[i])*.
- Se repite la secuencia con el bit i+1.

Nuevamente se aclara que esta secuencia se repite de forma de lograr transmisión y recepción de a bits de forma serial.

Finalmente se comprueba que lo enviado concuerde con lo recibido y con ello se finaliza el test. Otra importante diferencia que aquí se aprecia es que no se necesita liberación del recurso UART, simplemente se sale del test.

Referencias

[1] Analog-Devices, "AD844 - 60 MHz 2000 V/us Monolithic Op Amp," 2015. [Online]. Available: http://www.analog.com/media/en/technical-documentation/data-sheets/AD844.pdf

[2] D. C. Barber and B. H. Brown, "Applied potential tomography," *J. Phys. E.*, vol. 17, no. 9, p. 723, 1984. [Online]. Available: http://stacks.iop.org/0022-3735/17/i=9/a=002

[3] R. Carrillo, Y. Ortiz, and C. Peña, "Edema agudo pulmonar postextubación secundario a laringoespasmo," *Rev. Mex. Anestesiol.*, 2012. [Online]. Available: http://www.medigraphic.com/pdfs/rma/cma-2012/cma123g.pdf

[4] K. S. Cole and R. H. Cole, "Dispersion and absorption in dielectrics. I. Alternating current characteristics." *J. Chem. Phys., vol. 9, no. 4, p. 341*, 1941.

[5] Draeger, "PulmoVista500 - Draeger. Monitoreo de ventilacion mecanica," *Haciendo visible la Vent.*, 2010. [Online]. Available: http://campaigns.draeger.com/pulmovista500/es/

[6] A. Falcon, C. Rodriguez, F. Molina, E. Pereira, and N. Diaz, "Edema agudo de pulmón," *Rev. las Ciencias la Salud Cienfuegos*, 2006. [Online]. Available: http://www.sld.cu/galerias/pdf/sitios/urgencia/7_edema_agudo_pulmon.pdf

[7] S. Gonzalez, A. Liguori, and F. Simini, "IMPETOM Tomógrafo de Impedancia Eléctrica." Universidad de la República - Montevideo, Uruguay, p. 109, 2005.

[8] R. Hartmam, J. Lobo, M. Ruétalo, and F. Simini, "IMPETOM-I Tomógrafo de Impedancia Eléctrica." Universidad de la República - Montevideo, Uruguay, p. 199, 2002.

[9] A. Oppenheim and R. Schafer, "Discrete-Time Signal Processing,3/E," *Massachusetts Inst. Technol. - Hewlett packard Lab.*, 2009.

[10] A. Rodríguez, A. Ferreira, and F. Simini, "IMPETOM-C Tomógrafo de Impedancia Eléctrica." Universidad de la República - Montevideo, Uruguay, p. 159, 2002.

[11] E. Santos, "Equipamiento para tomografía por impedancia eléctrica," in *Semin. Ing. Biomédica*, F. Simini, Ed. Montevideo, Uruguay: Universidad de la Republica, 2014.

[12] E. Santos and F. Simini, "Electrical Impedance Tomography for pulmonary oedema extent monitoring: review and updated design," *J. Phys. Conf. Ser.*, 2012.

[13] ——, "Alternativas de proyecto tendientes a un tomógrafo por impedancia eléctrica para la presentación compacta del edema de pulmón a partir de cortes toracicos en tiempo real," M.S thesis, Universidad de la República - Montevideo, Uruguay, p. 255, 2014.

[14] G. Saulnier, "EIT instrumentation," in *Electr. impedance Tomogr. methods, Hist. Appl.*, D. Holder, Ed. London: Institute of Physics, 2005, ch. 2, p. 67.

[15] Spectrum-Digital, "Spectrum Digital Support - EVMOMAPL137," 2008. [Online]. Available: http://support.spectrumdigital.com/

[16] Texas-Instruments, "TMS320C6000 Code Composer Studio Tutorial," p. 126, 2000. [Online]. Available: http://www.ti.com/lit/ug/spru301c/spru301c.pdf

[17] ——, *TMS320C6000 DSP Multichannel Audio Serial Port (McASP)*, Included in the EVM-OMAPL137 instalation kit, 2008. [Online]. Available: http://www.ti.com/tool/TMDSOSKL137

[18] ——, *DSPLINK - User Guide*, Included in the EVM-OMAPL137 instalation kit, 2009. [Online]. Available: http://www.ti.com/tool/TMDSOSKL137

[19] ——, *OMAP-L137 EVM PSP User's Guide*, Included in the EVM-OMAPL137 instalation kit, 2009. [Online]. Available: http://www.ti.com/tool/TMDSOSKL137

[20] ——, *DSP/BIOSTM LINK*, 2010. [Online]. Available: http://processors.wiki.ti.com/index.php/Category:DSPLink

[21] ——, *DSP/BIOS^{TM} LINK PROGRAMMER'S GUIDE LNK 161 USR Version 1.61.03*, Included in the EVM-OMAPL137 instalation kit, 2010. [Online]. Available: http://www.ti.com/tool/TMDSOSKL137

[22] ——, *Low-Power Stereo Audio Codec for portable audio/telephony TVL320AIC3106 Datasheet*, Included in the EVM-OMAPL137 instalation kit, 2012. [Online]. Available: http://www.ti.com/tool/TMDSOSKL137

[23] ——, *OMAP-L137 C6000 DSP+ARM Processor Technical Reference Manual*, Included in the EVM-OMAPL137 instalation kit, 2013. [Online]. Available: http://www.ti.com/tool/TMDSOSKL137

[24] ——, "Getting Started Guide for OMAP-L137," 2014. [Online]. Available: http://processors.wiki.ti.com/index.php/Getting_Started_Guide_for_OMAP-L137

Índice de tablas

Esta página ha sido intencionalmente dejada en blanco.

Índice de figuras

Esta es la última página.
Compilado el sábado 12 diciembre, 2015.
http://iie.fing.edu.uy/