Instituto Tecnológico y de Estudios Superiores de Occidente

Reconocimiento de validez oficial de estudios de nivel superior según acuerdo secretarial 15018, publicado en el Diario Oficial de la Federación del 29 de noviembre de 1976.

Departamento de Electrónica, Sistemas e Informática

Maestría en Diseño Electrónico

REPORTE DE FORMACIÓN COMPLEMENTARIA EN ÁREA DE CONCENTRACIÓN EN SISTEMAS EMBEBIDOS Y TELECOMUNICACIONES

TRABAJO RECEPCIONAL que para obtener el **GRADO** de
MAESTRO EN DISEÑO ELECTRÓNICO

Presenta: **ADOLFO HERNÁNDEZ PADILLA**
Asesor **DR. OMAR HUMBERTO LONGORIA GÁNDARA**

Tlaquepaque, Jalisco. 07 de enero del 2021.

Contenido

1. Introducción

Durante el estudio de la Maestría en Diseño Electrónico hubo varias áreas de interés, pero la que de mi mayor interés fue el área de sistemas embebidos. En el presente reporte se darán a conocer los proyectos que fueron más relevantes. Observé también que en esta área existe un amplio rango de aprendizaje que me servirá durante la carrera profesional. Por lo cual con esto he reforzado, aprendido y practicado mis conocimientos en esta área. En el transcurso de mis estudios tuve el placer de disfrutar adquirir muchas experiencias nuevas de otras áreas que reforzaron también con grandes resultados mis conocimientos generales en el diseño electrónico. Tomé tres materias con más relevancia para mi desarrollo profesional que fueron las siguientes.

- **Ingeniería de software en ambientes embebidos.**
    - Proyecto: Arquitectura de software de un módulo BCM.
- **Sistemas Embebidos Avanzados.**
    - Proyecto: DMA Feature: Analog Signal Sampling Input.
- **Métodos de simulación de circuito electrónicos.**
    - Simulación de un circuito electrónico de una electromiografía.

2. Resumen de Proyectos Realizados

A continuación, se presenta una vista general de los proyectos Arquitectura de software de un módulo BCM, DMA (Direct Memory Access) Feature: Analog Signal Sampling Input y Simulación de un circuito electrónico de una electromiografía. Los reportes serán incluidos en el área de apéndices.

2.1. Proyecto 1 (Arquitectura de software de un módulo BCM)

En este proyecto se presenta una arquitectura de software para un módulo automotriz, llamado por sus siglas en inglés como BCM (Body Control Module). Para esta implementación se utilizó una arquitectura de software con un estándar de automotriz llamado AUTOSAR, (Automotive Open System Architecture) muy comúnmente empleado en toda la rama automotriz.

2.1.1 Introducción

Los sistemas electrónicos automotrices cada vez son más complejos y, por lo tanto, requieren ser mejores y muchos más eficientes cada día. Para lograrlo se hace uso de herramientas para hacer implementaciones mucho más robustas, eficientes, escalables y duraderas. Por lo cual se han creado varios estándares en la industria. Este estándar nos ayuda a hacer una implementación óptima de los recursos y las capas del sistema, entre otras virtudes que nos otorga el estándar.

2.1.2 Antecedentes

En sus inicios los sistemas automotrices fueron muy sencillos y proporcionaron un soporte mínimo, pero presentaban muchas fallas, sus implementaciones eran muy diferentes todos ellos, lo que provocaba una casi nula estandarización. La arquitectura de software no existía y no había un estándar en el que los fabricantes se basarán para la creación de los módulos. Esto llevó a la formación de estándares que nos ayudaron a establecer una forma de organizar los datos que el automóvil monitorea, genera y controla, con la finalidad de tener el mejor funcionamiento de los recursos del sistema. Debido a lo anterior, se vio la necesidad de crear arquitecturas de software que describieran el comportamiento de cada módulo, cómo se interconectaban entre sí y que tuviesen características muy específicas en implementación, mantenimiento y escalabilidad.

2.1.3 Solución Desarrollada

A partir de una lista de requerimientos del sistema BCM (Véase en la Tabla 1.0) se inició la arquitectura del software.

	Requerimientos del sistema	Requerimientos de SW/HW
TR1	**El sistema tendrá 2 terminales para interactuar con el interruptor de luz y los relés: iluminación interior y exterior.**	El pin de entrada MC debe estar en un nivel alto cuando el interruptor de luz está en la posición de encendido.
		El pin de entrada MC debe estar en un nivel bajo cuando el interruptor de luz está en la posición de apagado.
		La luz interior se apaga tan pronto como se enciende el interruptor de encendido.
		Si se abren todas las puertas, la lámpara interior se enciende 20 minutos y después se retrasa.
		La salida del relé de los faros delanteros está activada cuando el interruptor de los faros delanteros está encendido.
		Si el interruptor de la luz trasera está encendido, solo el relé de la luz trasera debe de está encendido.
		Tan pronto como se cierra la puerta, la luz interior se enciende 30 segundos y se atenúa en 2 segundos y después se apagará.
TR2	**Debe de usarse MCU del NXP MPC5606**	Los valores predeterminados de los registros deben ser 0 cuando la MCU está energizada.
		Las salidas digitales deben ser (0) es bajo y (1) alto.
		Las salidas analógicas deben ser bajas (0-3v) y altas (3.1-5v)
		Las entradas digitales deben ser (0) es bajo y (1) alto.
		Las entradas analógicas deben ser bajas (0-3v) y altas (3.1-5v)
		Las salidas analógicas/digitales deben ser bajas (0-3v) y altas (3.1-5v)
		Los valores de los registros deben inicializarse con valores de la ROM.
TR3	**El sistema tendrá 4 terminales analógicas para sensores de entrada: asistencia al conductor, nivel de combustible, pedal de freno.**	El rango de inicialización debe ser de 5 a 12 voltios.
TR4	**Todos los actuadores deben estar aislados del MCU.**	Todos los actuadores deben activarse de 0 a 5 voltios dentro de un voltaje de salida de 12 voltios.
TR5	**El protocolo que se utilizará será CAN 2.0 es común para aplicaciones automotrices.**	El mensaje de la lata debe tener el formato que se muestra en la tabla 5.
		RX ID must be 0x695 software mode and 0x726 application mode.
		TX ID debe ser el modo de software 0x696 y el modo de aplicación 0x72E.
		La velocidad en baudios debe ser de 500 K.
		El tipo de controlador debe ser de alta velocidad.

Tabla 1.0. Ejemplo de tabla de requerimientos del sistema de BMC.

La arquitectura del módulo se hizo por capas y módulos como lo sugiere AUTOSAR véase en la Figura 1.0. Cada capa tiene sus módulos internos que están conectados entre sí.

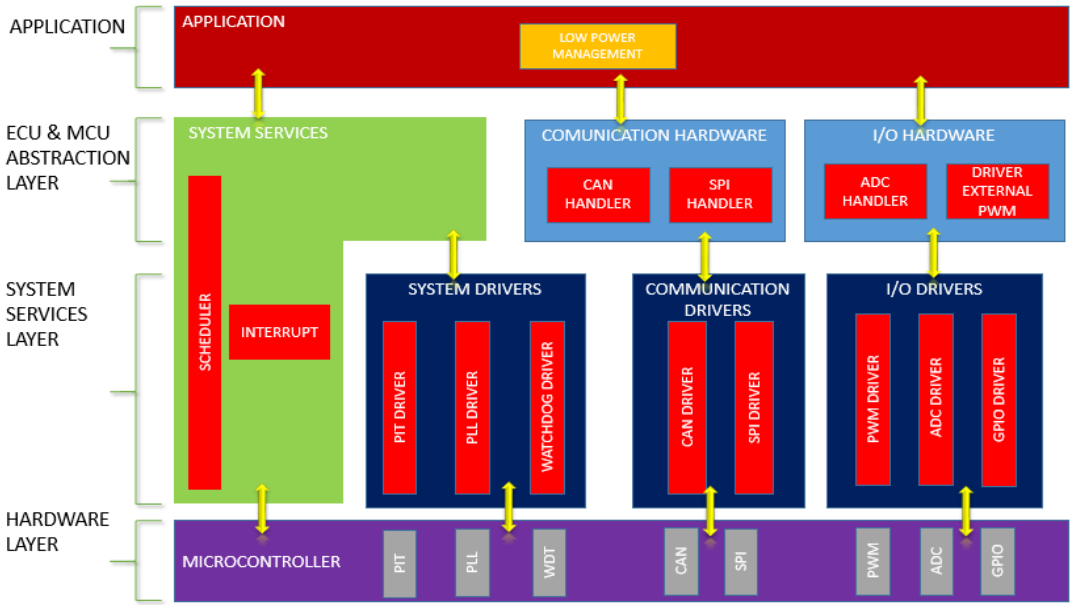


Figura 1.0. Estructura de capas del módulo BCM en AUTOSAR

También se utilizó el diagrama a bloques del módulo BCM, donde se obtuvo una mejor relación de la implementación con la arquitectura (Véase la Figura 1.1). Además, se utilizaron las entradas y salidas para la enumeración de los módulos.

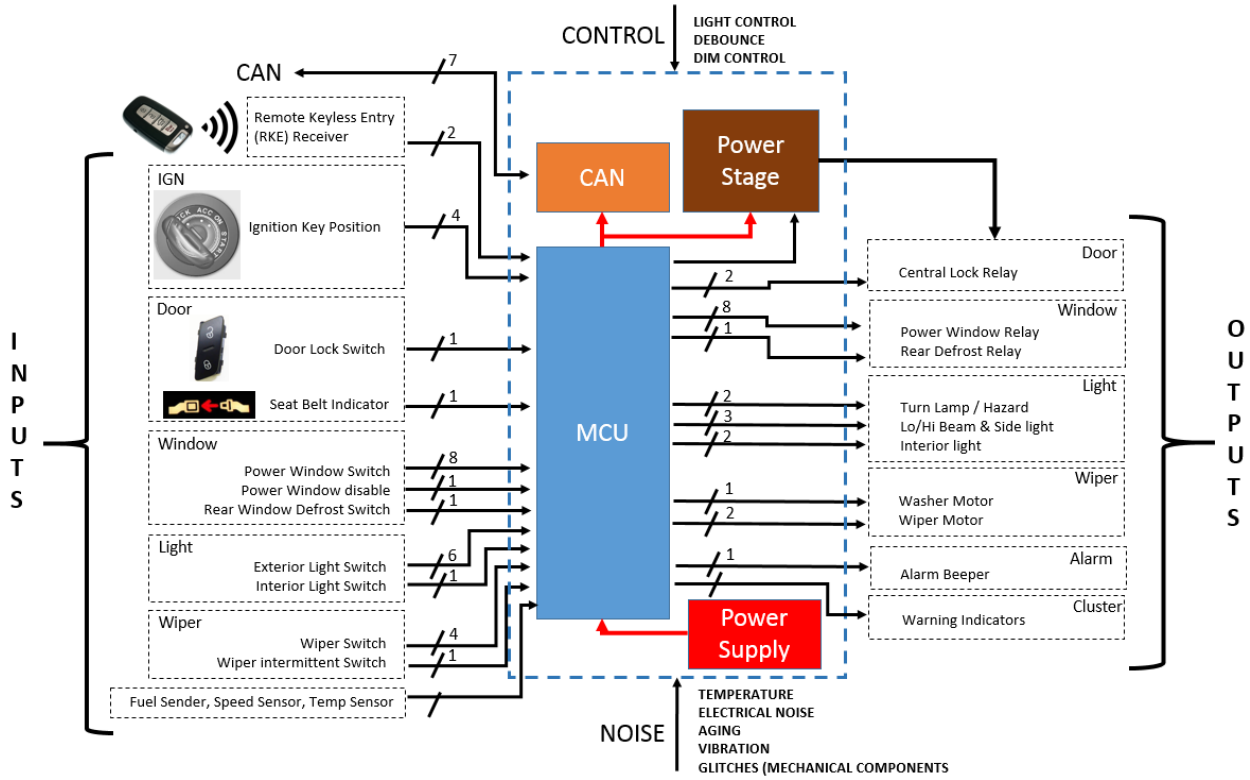


Figura 1.1. Diagrama a bloques del módulo BCM.

2.1.4 Análisis de resultados

Como resultado se obtuvo una implementación de arquitectura de sistema de software con el estándar AUTOSAR. La implementación fue por etapas como se indica en el estándar, siguiendo protocolos, lineamientos y estructuras de conexión entre cada módulo, respetando sus líneas de conexiones. Hubo algunas partes que no se pudieron estandarizar, ya que no todos los módulos existentes se encuentran soportados en el estándar, con esto se facilitará la codificación de programa (librerías, clases, estructuras, variables y entre otros aspectos) a fin de obtener los beneficios del estándar y para contemplar los objetivos finales del proyecto. Todo esto fue realizado con mucho cuidado siguiendo la especificación de cada bloque y con los requerimientos que el módulo BMC necesita.

2.1.5 Conclusiones.

Como resultado se diseñó la arquitectura de un módulo automotriz BMC, que ayudará, en un próximo paso al diseño del software. Cabe resaltar que el software no fue diseñado en esta etapa ya que la clase se enfocó a hacer la arquitectura solamente. Al proporcionar esta arquitectura a un ingeniero de software, se le facilitará la realización del código al momento de su implementación, ya que viene explícitamente cómo hacerlo, y que nombre y variables se deben de utilizar para cada cosa durante el desarrollo. Hablando sobre el trabajo que se realizó, fue difícil al principio ya que se tuvo que aprender que acerca del módulo BMC y aparte los requerimientos que el cliente final necesito. También otro reto fue entender todas las reglas que AUTOSAR comprende y poder usarlas en la implementación de la arquitectura. Para terminar, se realizó una arquitectura del módulo BMC contemplando los requerimientos iniciales.

2.2. Proyecto 2 (DMA Feature: Analog Signal Sampling Input)

En este proyecto se realizó la implementación de un módulo DAC (Digital to Analog Converter) + DMA (Direct Memory Access) para el diseño de un bloque de un electrocardiograma. Los datos serán ingresados al DAC del microcontrolador por medio de un disparador que es activado por un PWM (Pulse Width Modulation) interno. Se utilizará la memoria DMA sin pasar por RAM o ROM y así serán procesados los datos de una manera más rápida.

2.2.1 Introducción

La electrónica ha apoyado al avance en la medicina moderna a tal grado de que el promedio de vida de la humanidad ha subido, y los equipos electrónicos han ayudado a tener un mejor manejo y monitoreo de órganos internos del cuerpo humano, en específico el corazón. El electrocardiograma es una herramienta que ayuda a los doctores a diagnosticar con más precisión del comportamiento del corazón. De aquí que este dispositivo se consideró como parte del proyecto de la materia de sistemas embebidos avanzados.

2.2.2 Antecedentes

El avance en el monitoreo del corazón con equipos electrónicos como el electrocardiograma ha sido paulatinamente veloz en fases y con varios procesos; para lograrlo, las ramas de la electrónica y la medicina tuvieron que evolucionar. La medicina necesitaba comprender con precisión los estímulos del corazón, cómo obtenerlos, decodificarlos y saber qué significaban. Los microcontroladores son encargados de leer los impulsos, procesarlos basándose en lo que la medicina investigó (reglas de comportamiento) y generar resultados v para poder facilitar un diagnóstico rápido y preciso. Los estímulos del cuerpo humano fueron interpretados por impulsos eléctricos, lo que la electrónica puedo controlar y manejar para poder entender que significaban. Para almacenar los impulsos eléctricos fue necesario guardarlos en memorias electrónicas para después manejarlos. Al paso del tiempo se fue mejorando el proceso de almacenamiento con memorias internas mucho más cercanas a la unidad de procesamiento y evitando el uso de memorias externas. Aquí es donde observamos esta nueva característica de los microcontroladores que en nuestro caso es DAC-DMA.

2.2.3 Solución Desarrollada

El propósito de este proyecto es un manejo rápido de los datos entre memoria y unidad de procesamiento, por lo que se utiliza una característica especial con la que cuenta nuestro microprocesador (DAC-DMA). Ya que, en otro modo de implementación, el proceso se realiza con buses más lejanos con una latencia a memoria mucho mayor. En lo que consiste la solución es tomar muestras de un sensor real y registrarlas en una hoja de cálculo, que es leído por el microcontrolador por medio serial y almacenado en la memoria interna, y que proporcionan una lectura como si fuese tomadas directamente de los sensores de entrada, el módulo DAC-DMA maneja los datos para ser procesados de una manera mucho más rápida y sin necesidad de utilizar algún otro bus de datos. Se configurará un periférico del microcontrolador llamado AFEC (Analog Front-End Controllers), que es la base para poder trabajar con el DMA interno, el que toca la memoria de datos internos que es donde se está almacenados las muestras tomadas del electrocardiograma. La configuración de AFEC utilizará una función de **autoscan** y una utilería del microcontrolador que realiza el disparo (trigger) que activará al generador de un módulo PWM interno que periódicamente nos estará entregado los datos para procesarlos y dejarlos en la memoria interna para su lectura.

2.2.4 Análisis de resultados

Se pudo configurar correctamente el DAC+DMA, las funciones de AFEC y se usaron tres tipos de configuración de tiempo de captura. Para la captura de los datos se utilizó una señal PWM como disparador, cada uno de sus flancos de subida sirvió para detecta cada dato con un ajuste de tiempo determinado. Las configuraciones fueron 50us, 62us y 50ms para apreciar las diferentes variaciones con respecto al tiempo de captura. Para comprobar el funcionamiento prácticamente, se conectaron dos tarjetas experimentales SAM V71 (véase en la Fig. 2.0) en la que se trabajó; la primera tarjeta sirvió para generar los datos a procesar y la segunda tarjeta los capturó y envió a memoria entregando como resultado final el proceso completo del electrocardiograma.

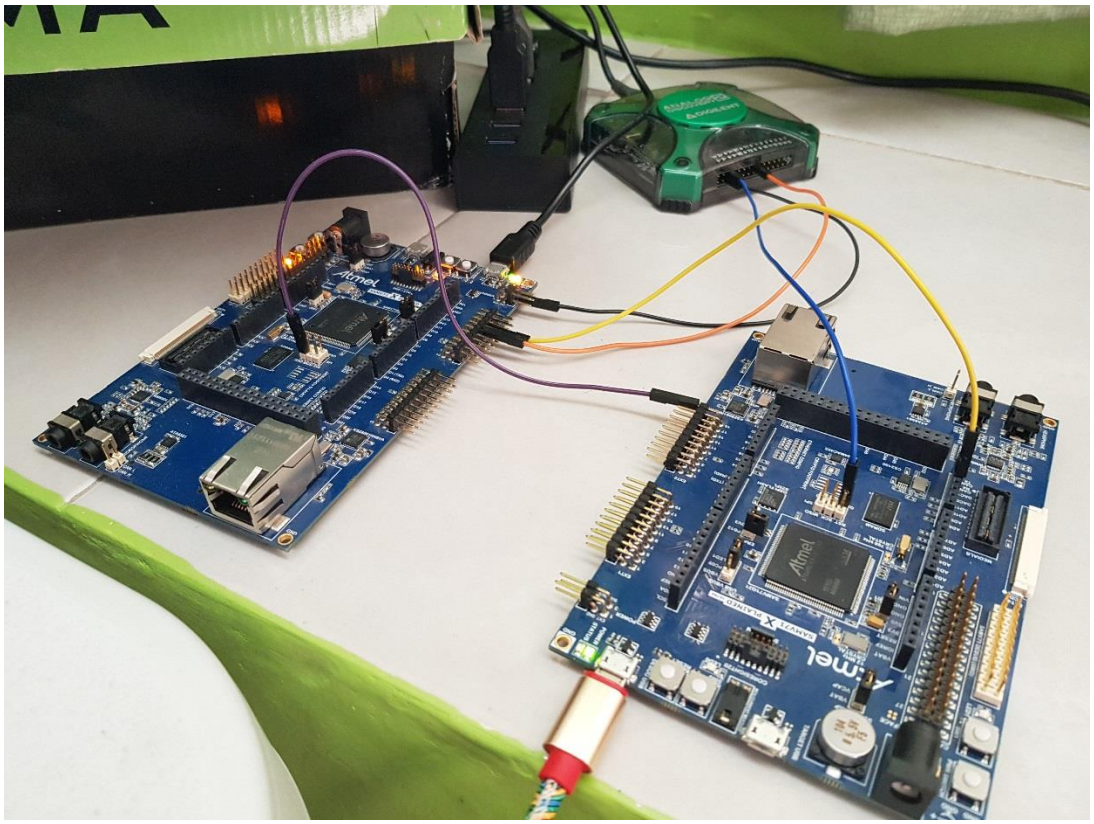


Figura. 2.0. Conexión de dos tarjetas SAM V71. Para la solución de DAC+DMA

2.2.5 Conclusiones.

En este proyecto se utilizaron varios de los temas vistos durante la materia de sistemas embebidos, desde entender la estructura interna del microprocesador y hasta como se programa cada una de sus funciones. Se implemento funciones especial como la transformada rápida de Fourier (FFT) para procesamiento de datos. La interacción con el hardware es de suma importancia a lo largo de este curco, ya que se desarrolló las habilidades necesarias para conocer los periféricos de la tarjeta que se utilizó. En este proyectó no se utilizó el concepto de “Basic Software”, ya que implementar este controlador no fue sencillo. Además de que AUTOSAR no tiene una arquitectura definida para DMA.

2.3. Proyecto 3 (Simulación de circuito electrónico para un electro miógrafo)

El objetivo de este proyecto es realizar una simulación de un filtro Sallen-Key como parte de un circuito de un electromiógrafo, utilizado en uno de los principales productos médicos existentes que mide la actividad eléctrica muscular.

2.3.1 Introducción

Un electromiógrafo presenta un problema desde el punto de vista de la electrónica, las señales de entrada para este equipo son muy ruidosa, además es costosa su implementación. Este equipo requiere del uso de un par de electrodos para la adquisición de las señales. El desarrollo de este proyecto se enfocará en la simulación de un filtro 8to orden Butterworth con topología Sallen-Key y un amplificador de instrumentación, partiendo de señales previamente obtenidas de uno electrodo real. El circuito conforma parte de la solución de una unidad protésica mecánica para solucionar una amputación transfemoral, y que ya cuente con una extremidad inferior. Se diseño y desarrolló el análisis del circuito electrónico con el uso de electrodos para la lectura de la actividad eléctrica producida por los músculos esqueléticos que participan en el movimiento de flexión y extensión de la rodilla del extremo cicatrizado del miembro amputado (muñón).

2.3.2 Antecedentes

Los electromiógrafos son muy comunes usados en nuestros días, su inicio fue hace tiempo aproximadamente en los años 1800, en ese entonces se percataron que los músculos vivos generan una pequeña corriente eléctrica al momento de moverse y que pueden ser medibles. Al paso del tiempo se crearon varios métodos más eficientes y fueron desarrollándose para poder medir con más precisión las señales del cuerpo. Hablando de la electrónica se han creado nuevos métodos de simulación para probar los circuitos electrónicos; al simular los circuitos podemos evitar fallas en su fabricación y así ahorrarnos gastos innecesarios de tiempo y dinero.

2.3.3 Solución Desarrollada

Iniciamos con el circuito previamente diseñado del filtro 8to orden de tipo Butterworth con topología Sallen-Key de ganancia unitaria para atenuar las frecuencias que se encuentran sobre los 500Hz y el amplificador de instrumentación. Los valores de voltaje que se manejaron fueron tomados directamente de la superficie de la piel, el rango en que se trabajo fue alrededor entre 10mV a 15mV a una frecuencia de 500Hz, es la que mejor responde el músculo esquelético que es donde la prótesis esta trabaja. En este caso el circuito servirá para la lectura de señales musculares del muslo humano. Se uso un amplificador INA128 (Véase en la Fig. 3.0) con una ganancia de 500 y un nivel de voltaje de salida de 5 voltios para trabajar con una tarjeta Arduino.

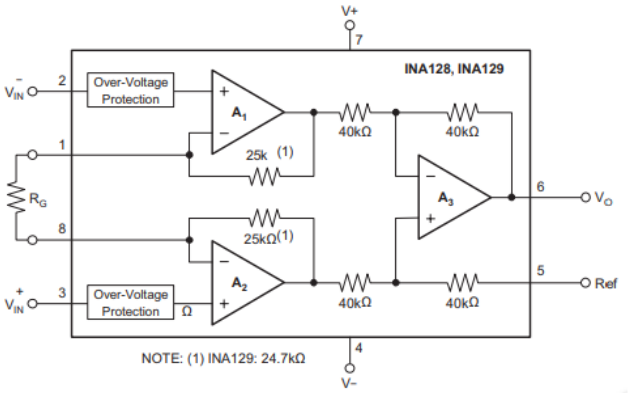


Figura 3.0. Bloque interno del 1NA148 amplificador de instrumentación de baja potencia.

Además, un filtro pasa bajas de 8vo orden tipo Butterworth (Véase en la Fig. 3.1) con topología Sallen-Key (Filtro electrónico activo valioso por su simplicidad) de ganancia unitaria para atenuar las frecuencias que se encuentran sobre los 500Hz.

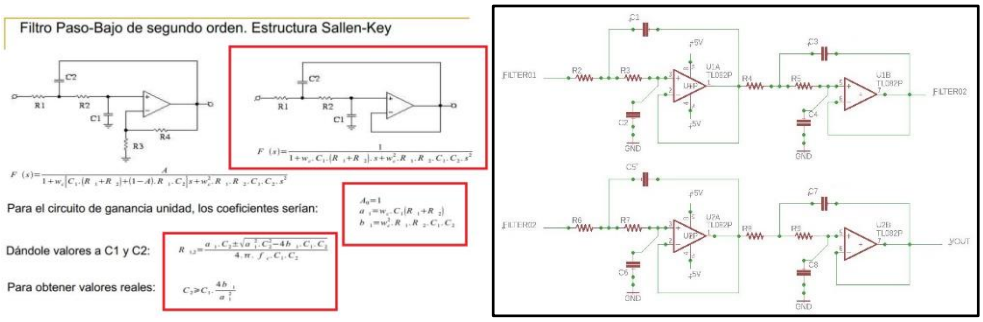


Figura. 3.1. (a)Ecuaciones y diseño básico para la formación de este filtro. (b) Esquemático del filtro.

Para el análisis de estos circuitos se utilizó WSPICE, primero fue el amplificador de instrumentación que se usó un modelo simulado directamente del circuito integrados con matrícula 1NA148. Se observa como la señal la de salida (Véase en la Fig. 3.2) sigue a la entrada negativa pero cuando esta con respecto a la señal positiva 180° desfasada, existe una pequeña ganancia adicional.

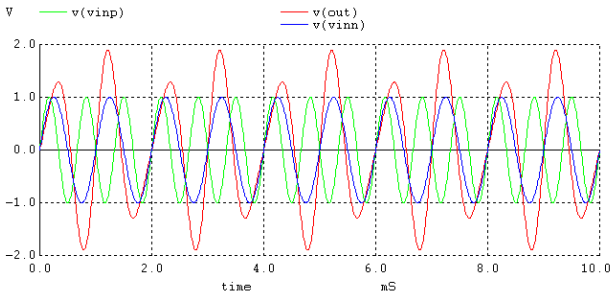


Figura 3.2. Simulación con WINSPICE las señales de entrada y salida del amplificador diferencial.

Después se simuló el filtro que también se utilizó WSPICE. Obteniendo una respuesta a la frecuencia adecuada para la utilización de nuestro dispositivo. (Véase en el Fig. 3.3)

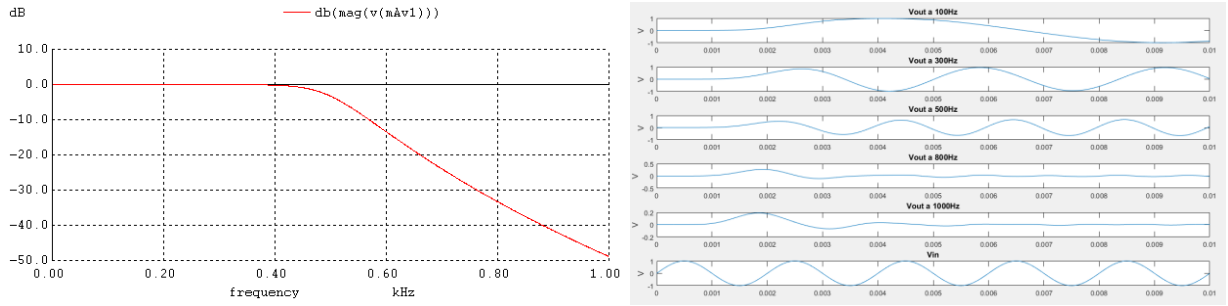


Figura. 3.3 (a)Simulación WSPICE de la respuesta 500Hz del filtro a -3db. (b) Relación de señales de salida a diferentes frecuencias del filtro.

2.3.4 Análisis de resultados

Se observó en la simulación final que el diseño funciona adecuadamente como se espera. Además, para tener un estudio más completo y encontrar posibles problemas con la manufactura y las tolerancias de materiales se realizó un Análisis de Monte Carlos (Véase en la Fig. 3.3). Se hizo el análisis con los valores reales y tomando en cuenta las toleraciones máximas y mínimas de los valores los componentes, para así determinar cuanta variación pudiéramos soportar en este punto de la fabricación y con la toleración de los componentes.

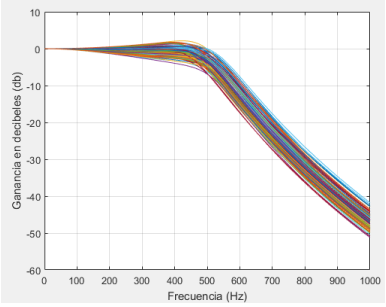


Figura. 3.3 Simulación WSPICE de la respuesta del filtro a -3db usando análisis de Monte Carlo para toleración de componentes.

2.3.5 Conclusiones.

Este proyecto pasó por varias etapas. La información sobre el desarrollo del filtro se extrajo de la tesis (Desarrollo e Implementación del prototipo de rótula 'Torres' para la automatización de una prótesis de extremidad inferior). La implementación en el simulador WinSpice me limitó debido a que se tuvo una versión sin registro; esta me limitó al número de nodos que pude conectar entre sí y que el circuito no se pudo simular completo en una sola simulación. Por lo tanto, se dividió el circuito en dos partes, primero el amplificador de instrumentación y segundo el filtro de octavo orden. Para simular el componente 1NA148 se descargó su librería directamente del proveedor.

En el Análisis de Monte Carlo se observó que con pequeños cambios en los valores los componentes se afectaba la respuesta del circuito de una manera drástica, por lo que fue necesario poner atención en los valores indicados, aquí la importancia de este análisis. Otro reto fue el controlador entre Matlab y WinSpice, que se diseñó desde cero; aquí se tuvo muchos problemas, uno de ellos fue la ruta de donde correr el programa de WinSpice que requería que el folder de Matlab coincidirá con las carpetas de la dirección de instalación del WinSpice; otro problema fueron las versiones de WinSpice, porque en la versión 1.40 el nombre de una subrutina de Matlab no debería de ser mayor de seis caracteres tarde mucho tiempo en resolver esto.

3. Conclusiones Generales

Mi experiencia de realizar mi maestría en el ITESO fue muy agradable y productiva de manera profesional y social. La maestría en diseño electrónico en ITESO fue algo complicada, ya que te exige mucho tiempo y compromiso, tanto en la realización de tarea, proyectos y horas de estudios adicionales aunado a el trabajo laboral. Me gustó que la maestría se enfoque mucho más a lo práctico, ya que me ayudó a reforzar mis conocimientos teóricos visto previamente en la ingeniería, que me ayudaron a tener un mejor enfoque en mis conocimientos básicos y un mayor resultado de mi aprendizaje. En el área de sistema embebidos entendí los conceptos de un estándar como AUTOSAR, la diferencia entre programar leguajes C y C para embebidos; también el porqué de la diferencia entre estos y cómo es que afecta en el hardware. En el aspecto del diseño electrónico de PCB observé la importancia de los pequeños detalles, en el que aprendí que hasta un pequeño mal cálculo puede provocar cambios en las características finales del diseño. También comprendí lo importancia de tener un modelado eléctrico preciso para poder tener los mejores resultados al momento de mandar a fabricar y esto con ayuda de simuladores electrónicos.

4. Referencias.

Apéndice A “Body Control Module Software Architecture (AUTOSAR)”
Apéndice B “DMA Feature: Analog Signal Sampling Input”
Apéndice C “Métodos de simulación de circuito electrónicos”

5. Apéndices

5.1. Apéndice A

Instituto Tecnológico y de Estudios Superiores de Occidente

Ingeniería de Software

BCM

Body Control Module

Software Architecture

Adolfo Hernández Padilla
Héctor Aguirre Diaz
Sarahi Partida Ochoa
Edgar Oropeza

Guadalajara, Jalisco. Noviembre, 2016

Índex

Introduction

The current document describes the software architecture of the BCM (Body Control Module). For its implementation, the AUTOSAR (Automotive Open System Architecture) model will be used as the main architecture reference; however, it will not be fully compliant with AUTOSAR since only some layers will be used.

Software Architecture

The software architecture model uses the system requirements, the software requirements and the layers suggested by the AUTOSAR model to generate the architecture.
For the current implementation of the software architecture the following layers will be included:

- **Application Layer**
- **ECU and MCU Abstraction Layer**
- **System services Layer**
- **Hardware Layer**

Each of these layers contain different blocks for the implementation as shown in the diagram of the *Figure 1*. In some cases, the blocks can expand into two different layers: e.g. System Services Module.

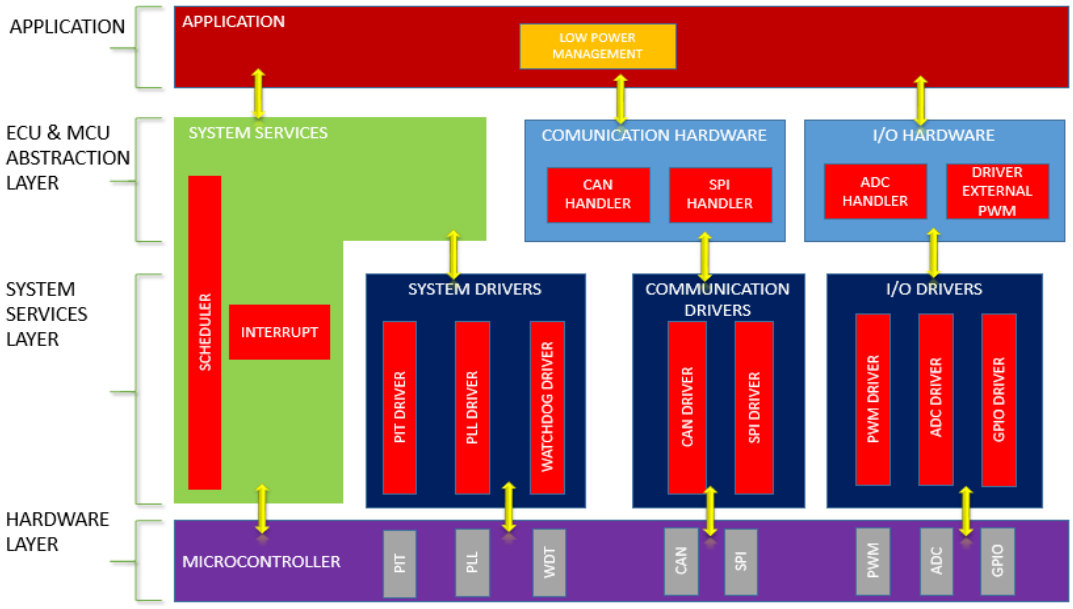


Figure 1: BCM Software Architecture

Application Layer (APL)

This layer will oversee the low power management, which can switch the operation mode of the BCM to or from low consumption mode.

When the system detects a low voltage condition, the BCM turns off the board and interrupts the CAN communication after sending a message to the cluster about the battery status.

ECU & MCU Abstraction Layer

The MCU layer contains software modules with direct access to the microcontroller internal peripherals and memory mapped external devices. Their main function is to make higher software layers independent of microcontroller.

The ECU layer provides an API for access to peripherals and devices regardless of their location (microcontroller internal/external), the number of existing devices of that type, the hardware realization of the same devices and their connection to the microcontroller. These interfaces contain the functionality to abstract the hardware realization of any specific device and they do not change the content of the data.

On the same line, the handlers are specific interfaces that control the concurrent, multiple, and asynchronous access of one or multiple clients to one or more drivers without changing the content of the data.

System Services (OS)

This block includes basic hardware configuration consisting of an Interrupt Handler module and an OS scheduler module. The named modules have direct access to the microcontroller internal peripherals and memory mapped external devices. Their main function is to make higher software layers independent of microcontroller.

OS Scheduler: This module is responsible for managing the CPU resources' usage, some of the characteristics include: To provide a priority-based scheduling for the CPU tasks, configuration and escalation based on statistic data and sensitive to real-time performance reasoning.

Interrupt Handler: The module responsible to handle the system interrupts. These are classified in three categories: Periodic Interrupts, Communication Interrupts, and I/O Interrupts

Communication Hardware Abstraction

This group of modules interfaces the communication controllers and the ECU hardware layout. For all communication systems, a specific communication hardware abstraction is required. Its task is to provide equal mechanism to access a bus channel regardless of its location (on-chip/ on-board).

I/O Hardware Abstraction

This group of modules interface the peripheral I/O devices (on-chip or onboard) and the ECU hardware layout (excluding sensors and actuators). The different I/O devices are accessed via an I/O signal interface. Its task is to represent I/O signals as they are connected to the ECU hardware (e.g. current, voltage and frequency measurements).

Input/output drivers

The BCM will interact with sensors and actuators through the drivers for Analog and Digital Signals. Some of the digital signals will be used as luminosity signals (LEDs) to indicate status of the system.

System drivers

These drivers allow the execution of the scheduler, take over the execution flow, verify the correct operation of the system (through the watchdog), verify the integrity of the memory and provide the OS tick for the scheduler time base.

Communication drivers

These drivers will be responsible for handling the communication Framework, the CAN and SPI interfaces and the I/O and Network management.

Access to the hardware is routed through the hardware layer to prevent direct access to the microcontroller registers from higher-level software. The hardware layer ensures a standard interface to the different components of the ECU Software Layer. Its main function is to manage the MCU peripherals and provide hardware values to the ECU modules in the layers above. The module will do this through a notification structure to support the distribution of commands, responses, and information to the different processes. The peripherals handled by the hardware layer are:

Digital I/O (DIO)
Analog/Digital Converter (ADC)
Pulse Width (De)Modulator (PWM, PWD)
Capture Compare Unit (CCU)
Watchdog Timer (WDT)
Serial Peripheral Interface (SPI)
Controller Area Network Bus (CAN)

Software Design

The BCM software design represents an architectural structure with interfaces that simplify the connections between the different modules and the external environment.

The life cycle of the BCM development is described next:

1. **Requirements:** Identify the problem and classify the requirements into functional and non-functional.
2. **Model of problem:** Analyze and build the model of the problem by taking the AUTOSAR reference as the base line.
3. **Postulate a design solution:** Organize the elements and modules to build the software architecture.

4. **Validate Solution:** Generate the V-Model to validate the design.
5. **Refine Design Solution:** Iterate through the software design cycle to implement corrections and improvements to the model.
6. **Implement Solution:** Programming the software design in C language using Atmel IDE.

Scheduler

The OS scheduler uses a real time periodic interrupt implemented by hardware to generate the heartbeat of the operative system; this heartbeat is known as the OS tick.

The flow of interrupt service routine (ISR) of the timer interrupt is described in the *Figure 1*. The timer interrupt must be set before the configuration of the PLL.

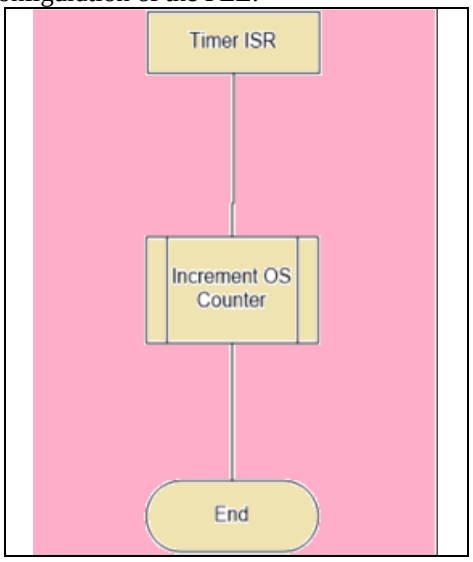


Figure 1: ISR of the periodic timer interrupt.

The periodic operation of the BCM is described in the task tables. A brief description, the periodicity and the priority of the tasks to be executed are described in the task tables. A fully cooperative scheduling algorithm is used to schedule the execution of all task in the system. The current implementation offers flexibility to configure different modes of operation for the BCM. The modes of operation can be switched to *normal mode* and *low voltage mode* and are represented in two different tables. *Table 1* shows the task tables definition.

Table 1: Task table definitions

Task Name	*Normal mode Functions*	*Low power mode Functions*	*Rate [mS]*
Task2MS	*Debounce Key Status* *Debounce Brake Pedal*		*2*
Task4MS	*Get Battery Level* *Get Brake Pedal Status*	*Debounce Key Status* *Debounce Brake Pedal Get Brake Pedal Status*	*4*
Task8MS	*Set PWM Duty Cycle* *Send Data to Cluster* *Get Lights Signal Status* *Get Stalk Status*	*Send Data to Cluster* *Get Lights Signal Status* *Get Stalk Status*	*8*
Task16MS	*Process Received CAN Frames*	*Process Received CAN Frames*	*16*
Task32MS	*Transmit Stalk Status* *Transmit Lights Status*	*Set PWM Duty Cycle* *Transmit Stalk Status*	*32*

Transmit Brake Pedal Status	*Transmit Lights Status* *Transmit Brake Pedal Status*

After hardware configuration and prior to the execution of the tasks, the real time scheduler needs to find the active task table, validate that the table is the correct one and store the address of the valid table for the Kernel. This procedure is represented in the flow diagram of the *Figure 2*.

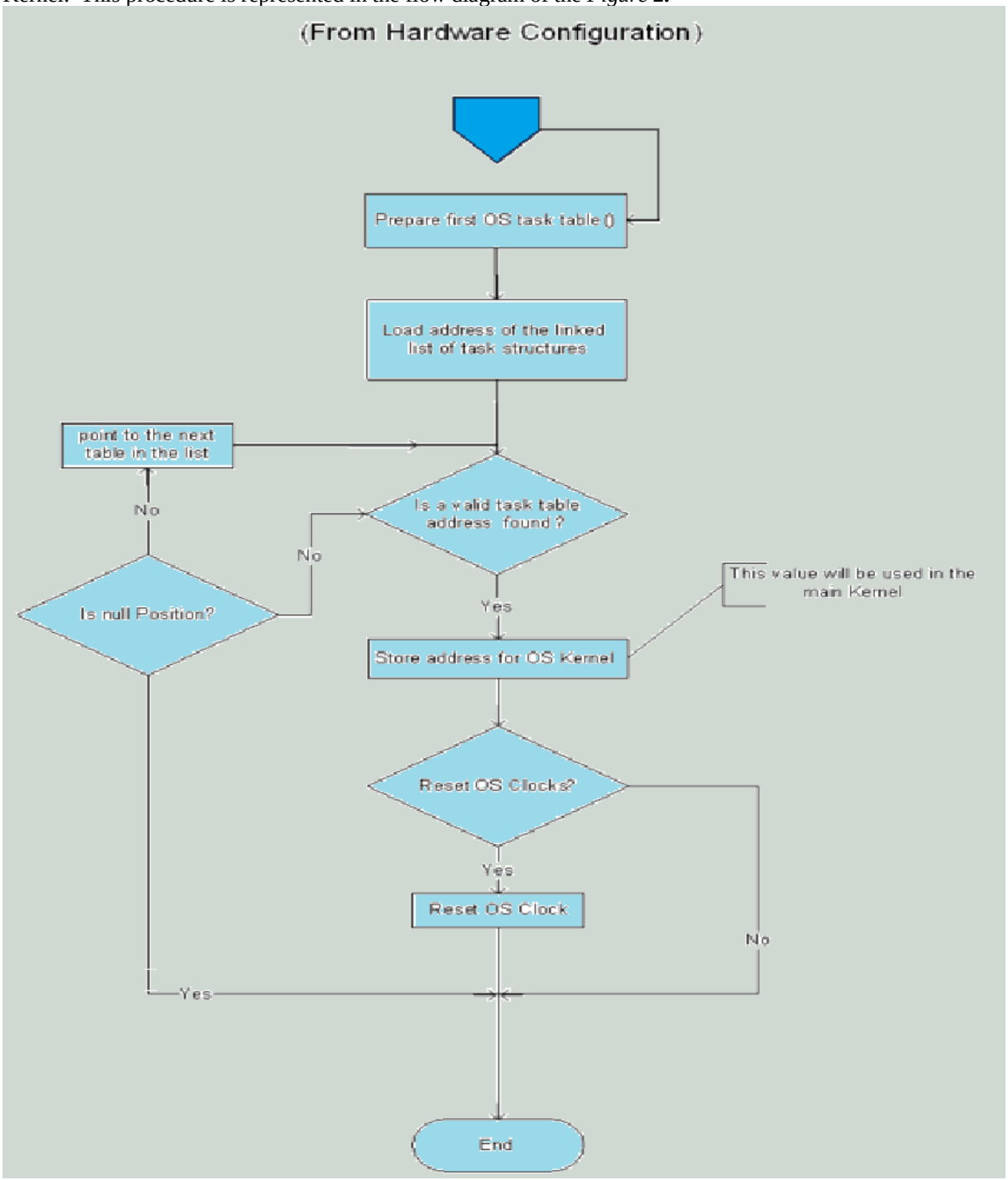


Figure 2: Flow Diagram of the preparation of the first OS task table.

The decision of which task will be executed will be performed by the kernel of the scheduler. The logic implementation of this module is illustrated in *Figure 3.*

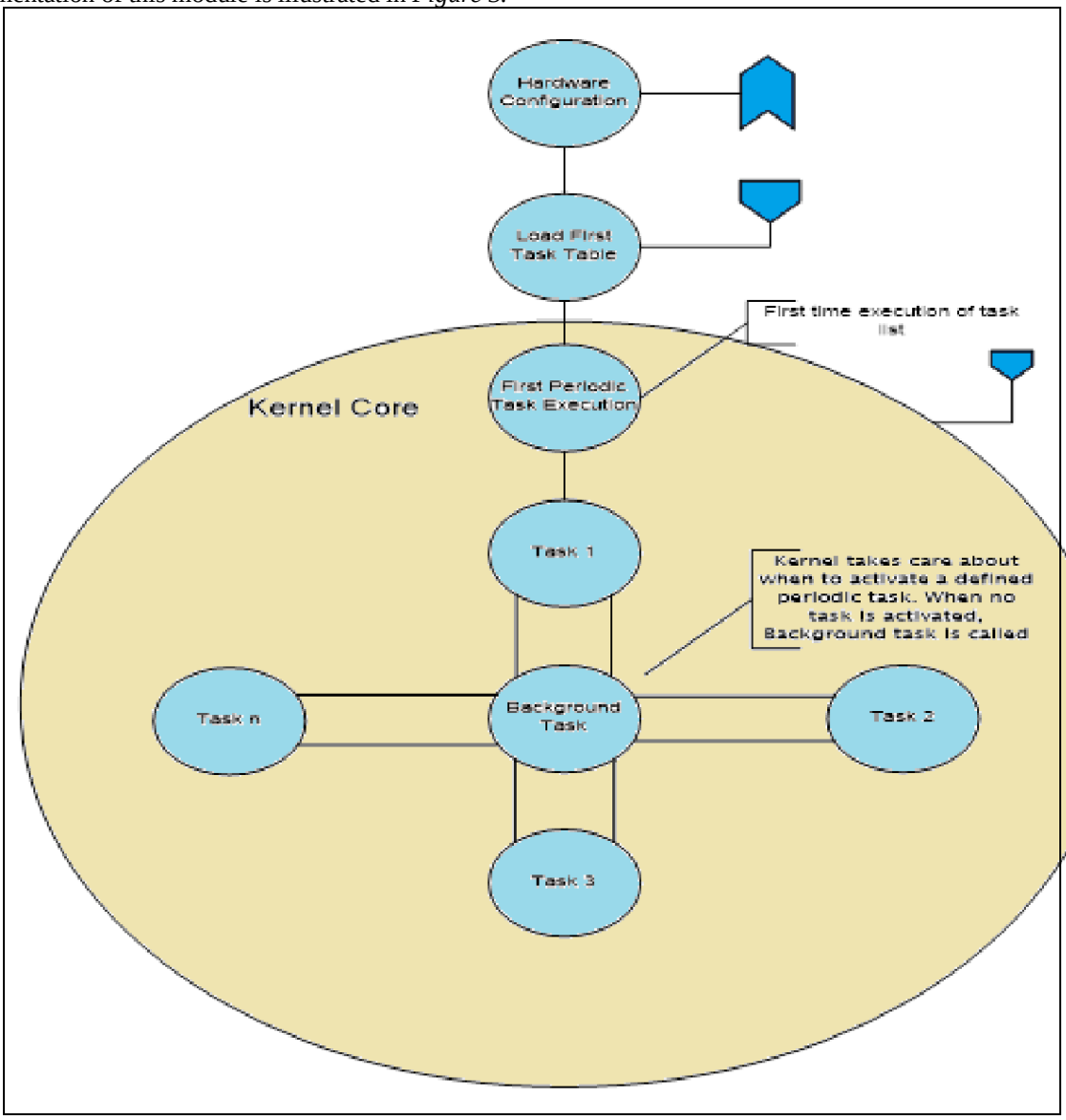


Figure 3: OS kernel block diagram.

CAN driver.

CAN Bus is a vehicle bus standard designed to allow microcontrollers and devices to communicate with each other a vehicle without a host computer. It is also a message-based- protocol, designed specifically for automotive applications. The CAN bus driver allows communication between BCM and Cluster for sending and receiving messages, *Figure 4.*

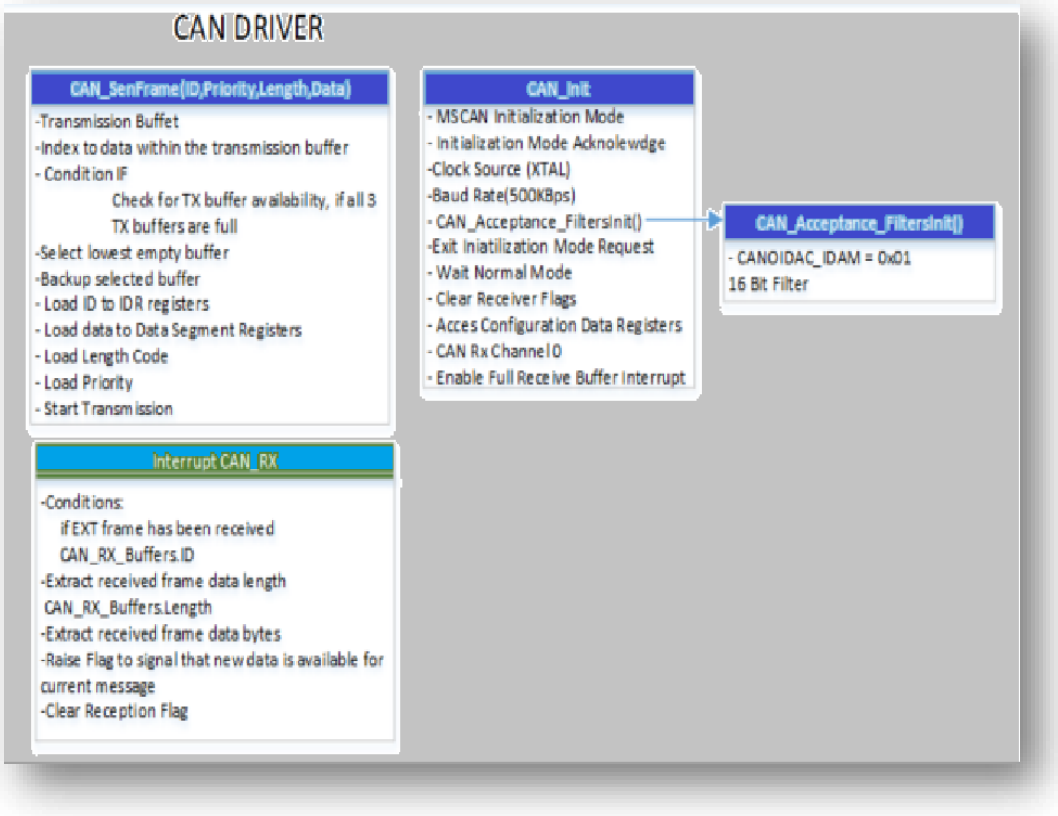


Figure 4: CAN block diagram.

PWM driver.

The PWM driver is used to control the intensity of the light (both external and internal). The BCM system has PWM signals that are connected into the input of the power switch, this driver has four functions that are shown in *Figure 5.*

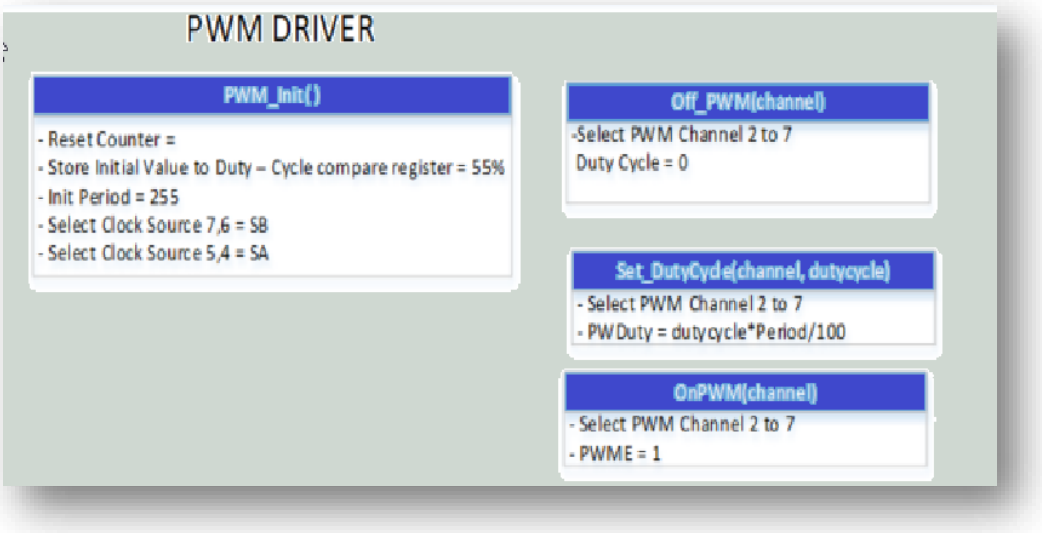


Figure 5: PWM block diagram.

ADC driver.

Most of the input sensors to the BCM are analog, the signals need to be digitalized by using an ADC port. The ADC can be configured to 10- or 12-bit resolution. For the current implementation, the ADC will be configured to 12 bits interrupt mode conversion. The sensors that are connected to the ADC ports are: temperature, humidity, speed, voltage battery and proximity. The functions contained in the ADC driver are shown in *Figure 6.*

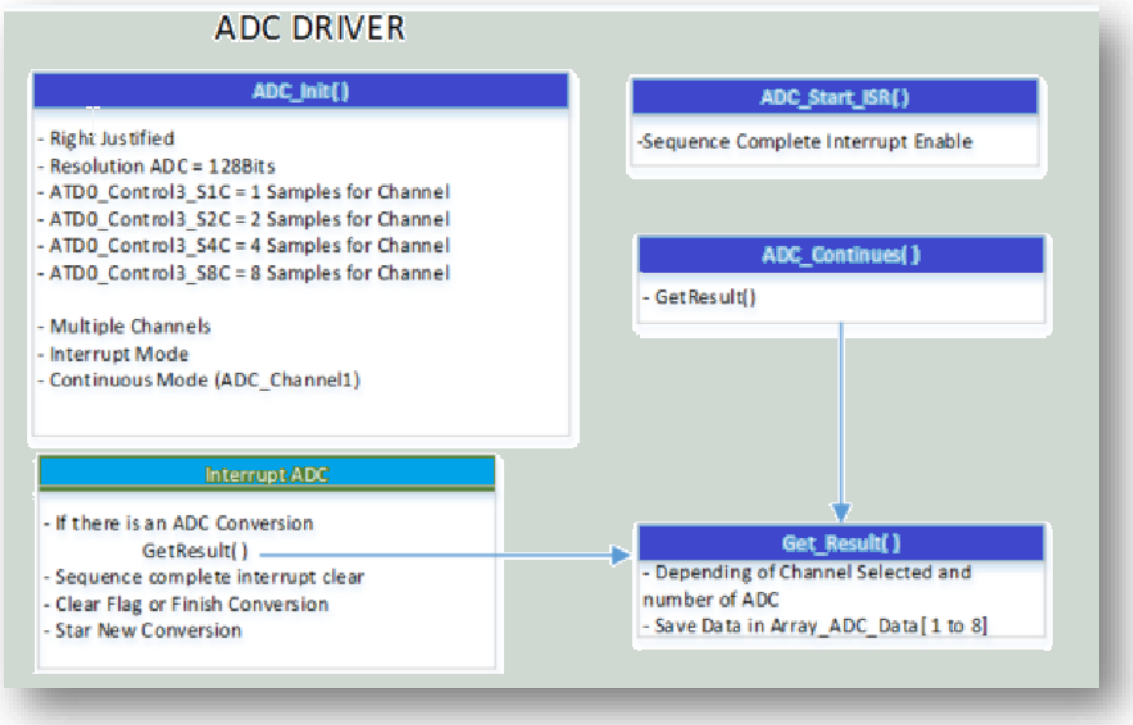


Figure 6: ADC block diagram.

GPIO driver.

The GPIO port will be used to indicate the status in the system performance. The BCM will report if the CAN bus lost the communication or if it is working in low power mode by using a LED connected to a GPIO port. The function to use a port are shown in *Figure 7*.

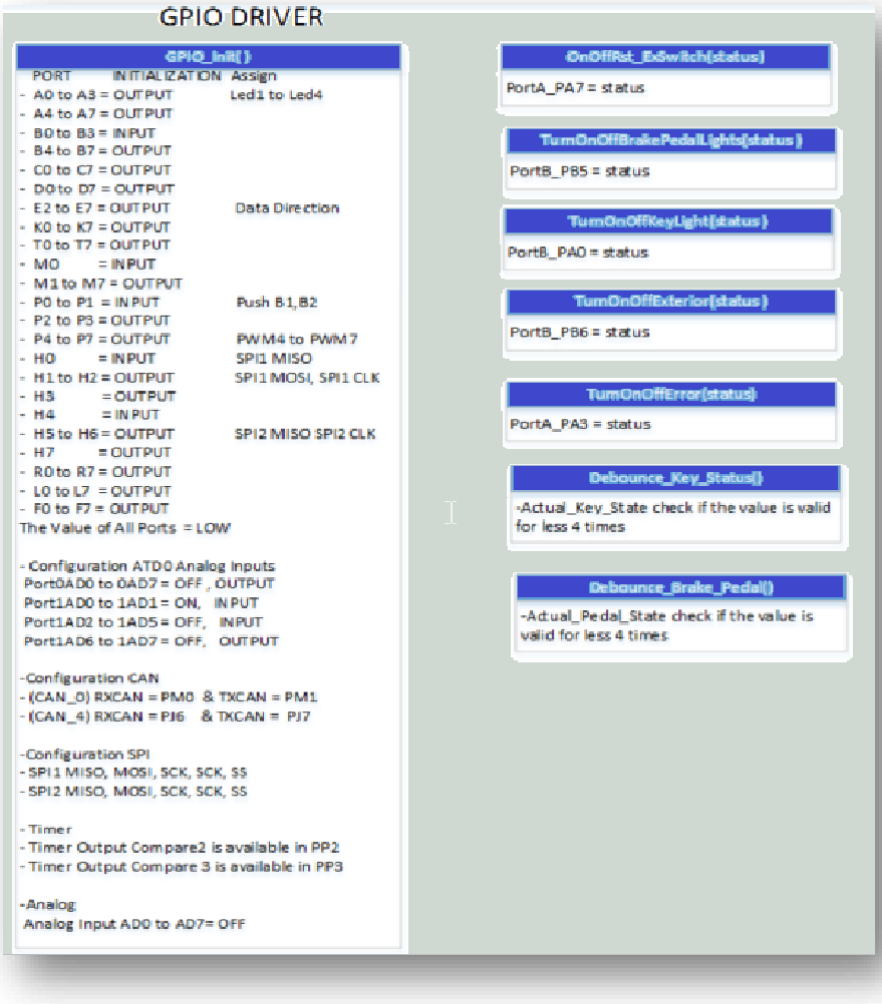


Figure 7: GPIO block diagram.

PIT Driver.

The PIT driver will assist the BCM scheduler to execute the periodic tasks needed by the system. The functions are listed in *Figure 8*.

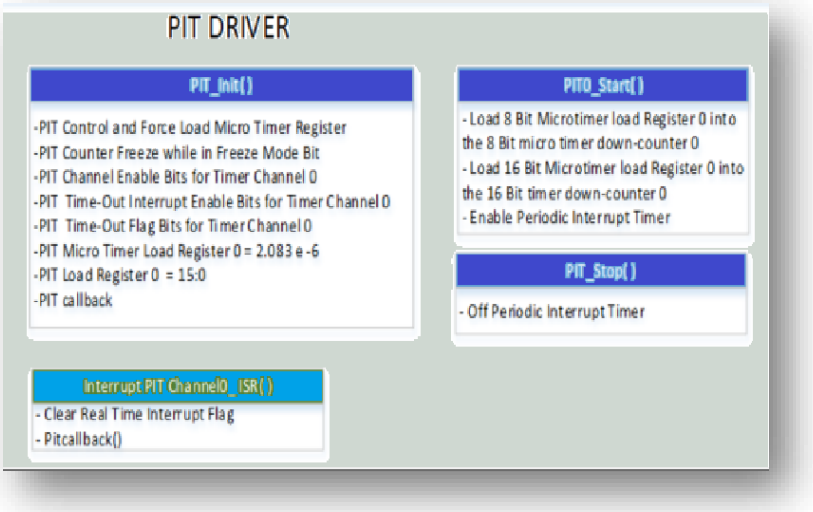


Figure 8: PIT block diagram.

Watchdog driver.

The purpose of the watchdog is to provide the system with a way to recover in the case of incorrect code execution or other events that may cause uncontrolled operation of the MCU. Typically, a watchdog is a continuously running timer that expirers or rolls over at a predetermined time interval. This interval is usually determined by the system clock frequency and a watchdog timeout value that is set by the application.

The application must perform some specific action before the timer expires, which causes a reset of the watchdog timer, and a restart of the timeout count. The WD must have the same time value that the task with the less execution time, in this case is 2ms

CAN Handler.

The CAN Handler contains some functions that permit to obtain the signals of different sensors sending by cluster and send messages with the system status, *Figure 9* has these functions.

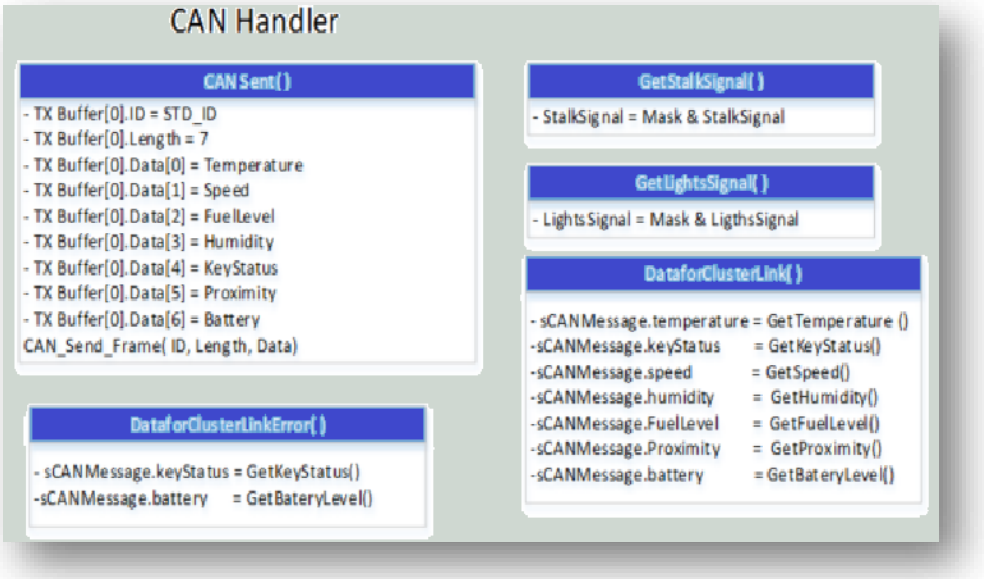


Figure 9: CAN Handler Block Diagram.

ADC Handler.

It permits to get the values for Proximity, Humidity, Temperature and Battery Level sensors, the *Figure 10* shows the functions that are using in this driver.

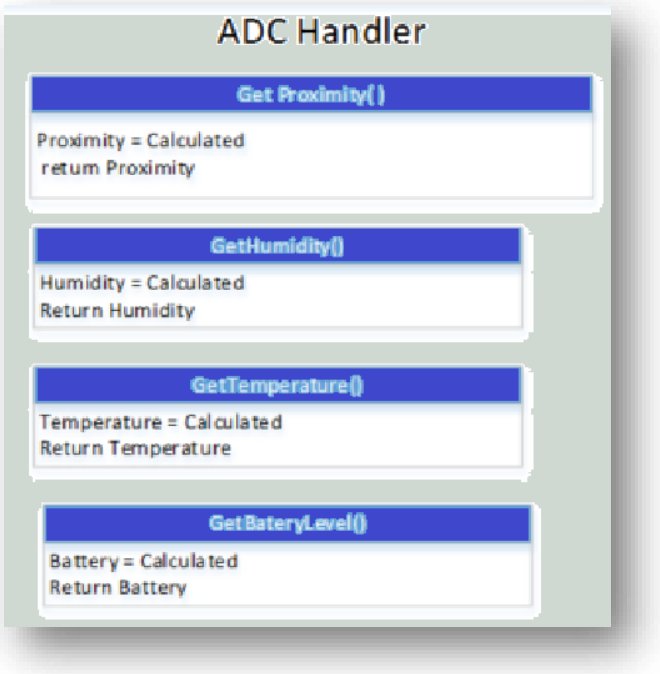


Figure 10: ADC handler block diagram.

The following diagrams show the algorithm to perform a conversion from a digital value to a number "readable" by the final user, for instance degrees, RH%, km/h, these values are then sent to the cluster display. These functions run in a low priority task.

Temperature: The temperature sensor has a resolution of 10mV per degree (Celsius). The ADC resolution is 12 bits (max count = 4095), and the CONSTANT is set to 500.

When the temperature sensor reads 30C, it produces a 300mV DC level at its output. When passed through the ADC this value is converted to 250. By applying the following formula, the final value to be displayed in the cluster reads 30C.

$$Cent = \frac{ADCValue * CONSTCENT}{RESOLUTION} = \frac{250 * 500}{4095} = \frac{1250000}{4095} = 30.52°C$$

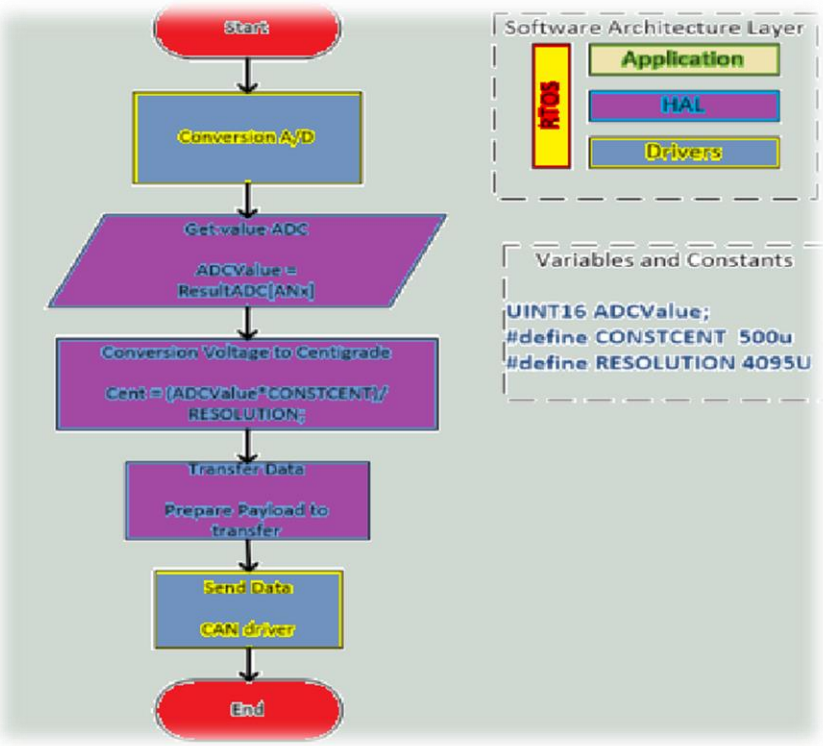


Figure 11: Converter ADC Digital Value to Centigrade.

Humidity: The humidity range (10 %RH to 90 %RH) is translated to a voltage range (1 to 2.9V) by the humidity sensor. The following formula is used to calculate the final value to be displayed in the cluster, when considering the maximum range of the sensor and a given constant.

$$HUM = \frac{ADCValue * MAX\_PORC}{MAX\_RANGE} = \frac{1666 * 90}{2497} = 60.04\ \%RH$$

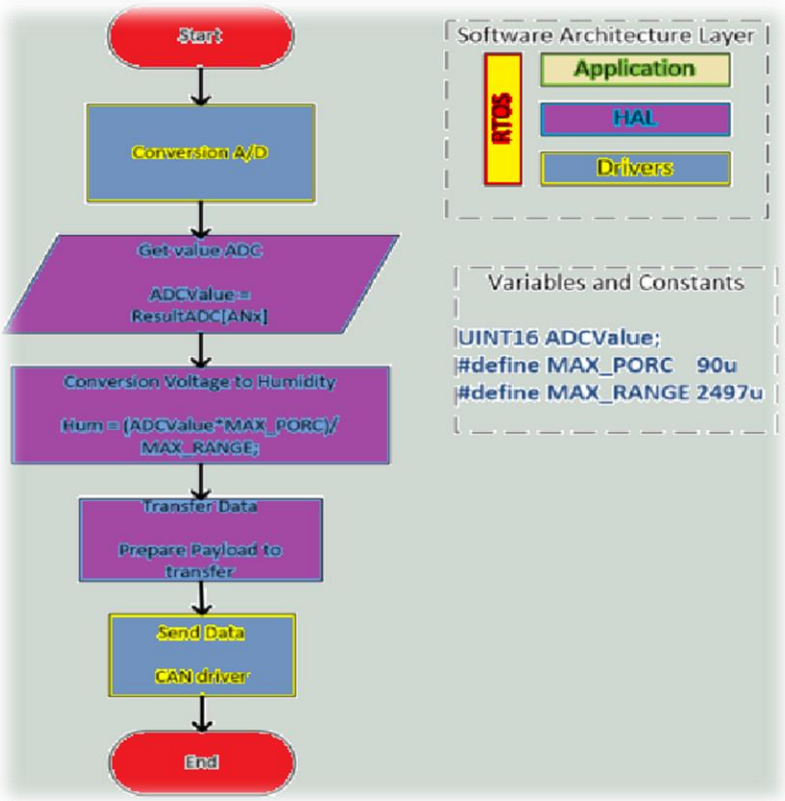


Figure 12: Converter ADC Digital Value to Humidity

Battery Voltage: The CONSTVOLT constant is obtained by dividing the ADC resolution by the max battery voltage (12V). If we assume a 4.5V reading in the output, the battery voltage is 11V.

$$Voltage = \frac{ADCValue}{CONTSVOLT} = \frac{3750}{341} = 10.99v.$$

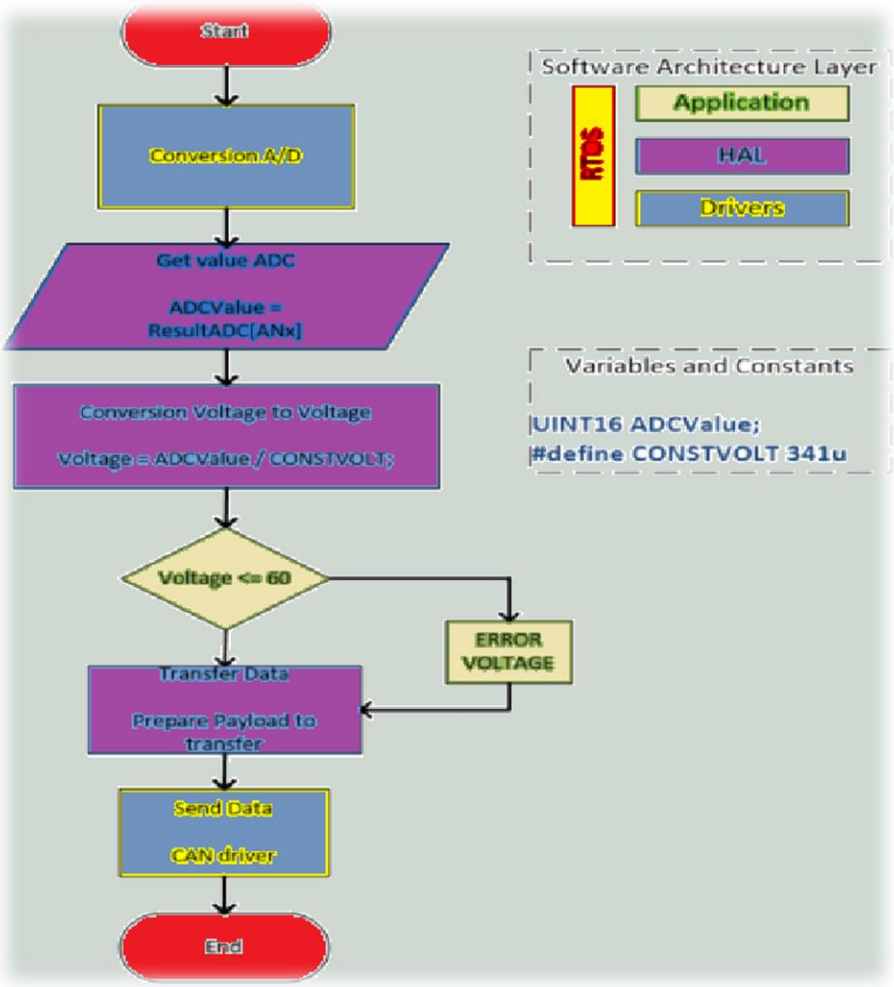


Figure 13: Converter ADC Value to Voltage.

Proximity: The proximity sensor resolution is given by:

$$Res_{Sensor} = \frac{5}{512} = 9.8mv = 1\ in\ or\ 2.54cm$$

And the ADC resolution is:

$$Res\_ADC = \frac{5}{4095} = 1.22mv$$

If Res_Sensor is divided by Res_ADC the result is 8, which means, each 8 units is equal to 2.54cm. The RESOLUTION_IN_CM constant is Res_Sensor multiplied by 100, at the end of the operation the result is divided by 100 to get the value in centimeters.

$$Dist = \frac{\left(\frac{ADCValue}{PRS\_TO\_ADC\_UNIT}\right) * RESOLUTION\_IN\_CM}{GO\_TO\_CM} = \frac{\left(\frac{1280}{8}\right) * 254}{100} = 398.14cm$$

When an object is present at 4m distance (400 cm), the output voltage is 1.53 or 1280 counts of the ADC.

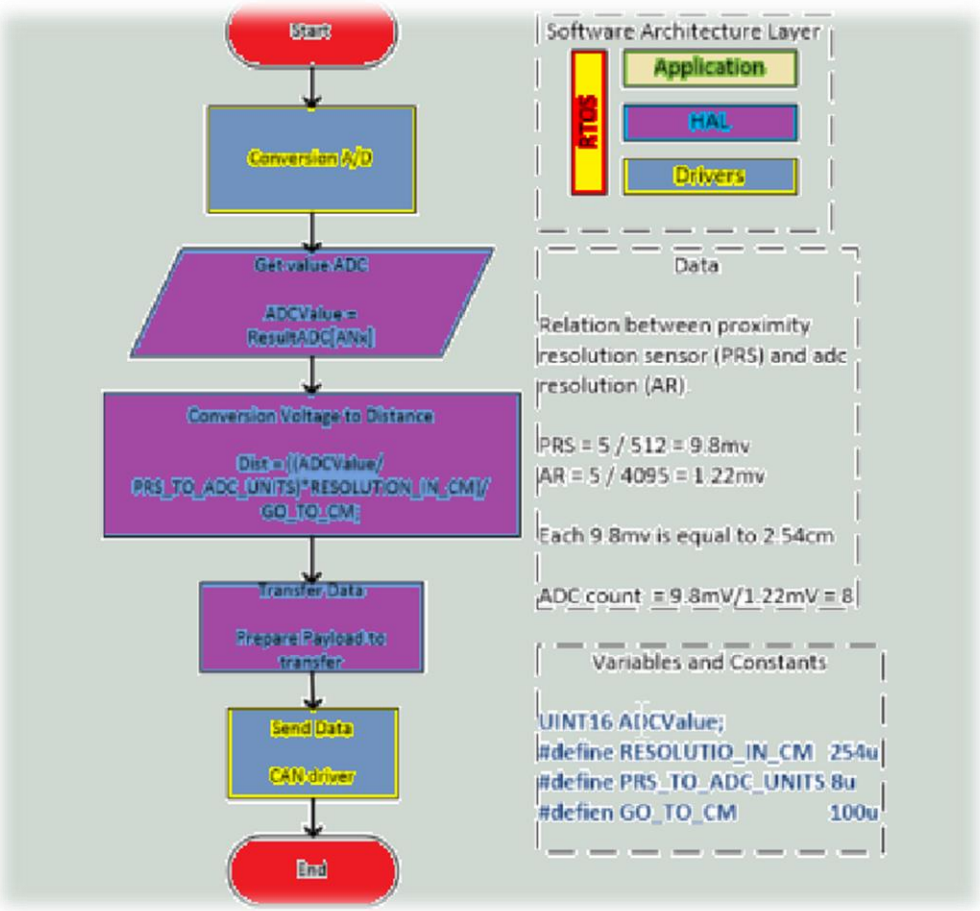


Figure 14: Converter ADC to D

5.2. **Apéndice B**

DMA Feature: Analog Signal Sampling Input

Sistemas Embebidos

Carlos Adrián Lugo Meneses
Email: xe723388@iteso.mx

Diego Peralta Reynoso
Email: xe723456@iteso.mx

Adolfo Hernández Padilla
Email: md701988@iteso.mx

[Seleccione la fecha]

ITESO
www.sistemasembebidos.iteso.mx/alumnos

I T E S O A. C., Universidad Jesuita de Guadalajara

Periférico Sur Miguel Gómez Morín #8585, Tlaquepaque, Jalisco, México
Technical Report Number: ESE-O2014-002

Abstract: *Este documento explica paso por paso cómo el equipo implementó la solución al problema dado en clase para la configuración de AFEC con DMA con disparo por medio de un timer para un evento periódico. Este proyecto será la será parte del proyecto final.*
Keywords: *Timer, Señales Analógicas, AFEC, DAC, Software Specification, DMA, linked lists, IO, hardware interaction, trigger, muestreo.*

Contenido

Requerimientos

Usando los ejemplos base del DAC + DMA realizar un proyecto que cumpla con los siguientes objetivos:

Table 1 Requerimientos

#	Requerimientos
1	El reloj y configuración del periférico AFEC en base a SAMP_PER -> dispara automáticamente la transferencia de DMA de fuente (periférico) a destino (memoria) cuando el resultado esté listo.
2	Configuración de AFEC en "autoscan". El disparo de transferencia de fuente (periférico/memoria) a destino (memoria) deberá ser en base a una señal de PWM configurada con un periodo definido de acuerdo con SAMP_PER.

Implementación

Con la finalidad de cumplir con los requerimientos propuestos para esta tarea, la estrategia es disparar el periférico AFEC mediante un timer. Una vez el dato ha sido satisfactoriamente convertido, procedemos a copiarlo desde el periférico a la memoria mediante el periférico DAM.

La otra opción es configurar el AFEC para que este leyendo de forma continua y el DMA dispararlo mediante un timer de manera periódica, sin embargo, esta estrategia no fue de nuestro agrado puesto que corremos el riesgo de disparar el DMA cuando el ADC este escribiendo en registro, pudiendo producir una corrupción de datos.

Nuestra opción permite que los periféricos funcionen sinérgicamente y solo cuando es necesario.

Un requisito de la tarea es usar punto fijo. Parea esto, hemos decidido usar **U16_10** y como unidad base 1 microsegundo, esto nos permite contar sin problemas desde 1 microsegundo hasta 50000 (figura 1) con una variable de 16 bits.

En la tarea se añade el archivo Tiempo_calculos.xlsx, el cuál especifica todos los posibles valores que puede tomar nuestro contador.

La implementación del timer, así como la habilitación de este como fuente externa en modo "trigger" ha sido tomada de nuestra tarea anterior acerca del módulo DAC.

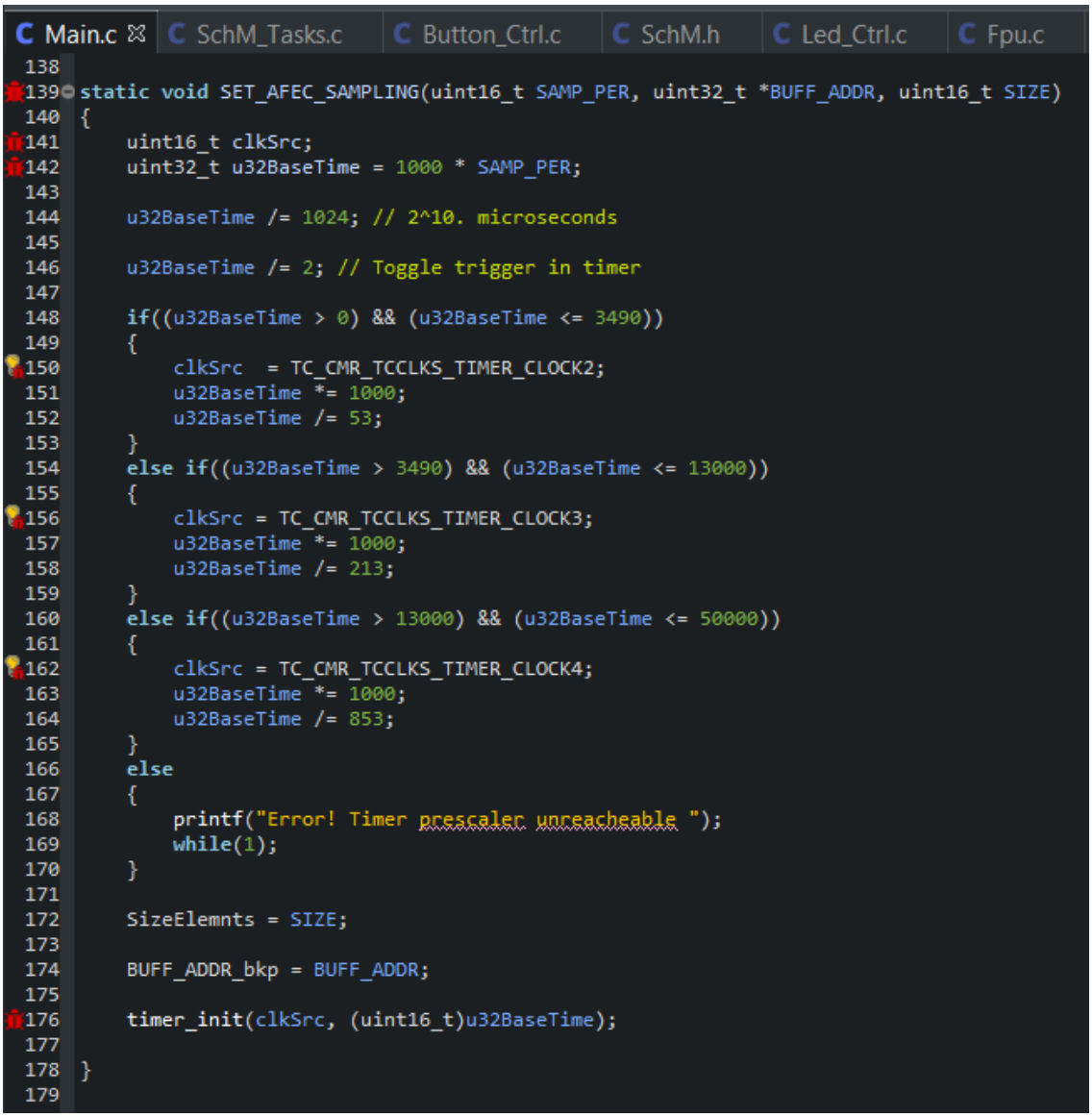


Figura 1. Implementación de la función Set_AFEC_SAMPLING.

Hemos creado las tres configuraciones propuestas en la tarea (figura 1). Respecto a la configuración 3, el tamaño del buffer lo hemos modificado de 16 a 512 bytes, debido a que generaba problemas dentro de la implementación de la FFT.

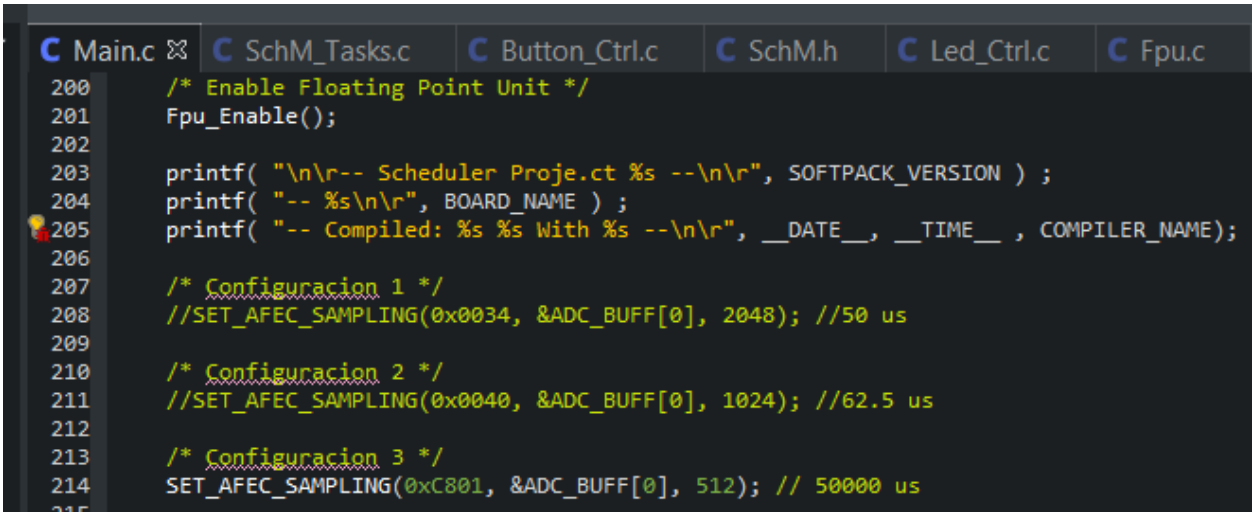


Figura 2. Configuraciones de muestreo.

Pruebas y resultados

En las imágenes 3, 4 y 5 se muestra el pin de trigger del AFEC para las 3 configuraciones. El módulo AFEC solo se dispara con flancos de subida.

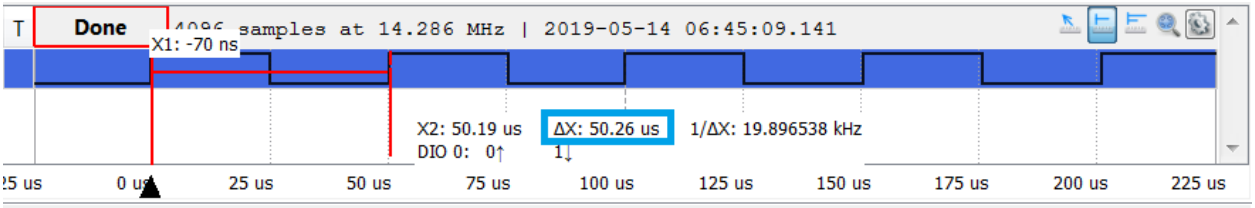


Figura 3. Tiempo de muestreo para la configuración 1.

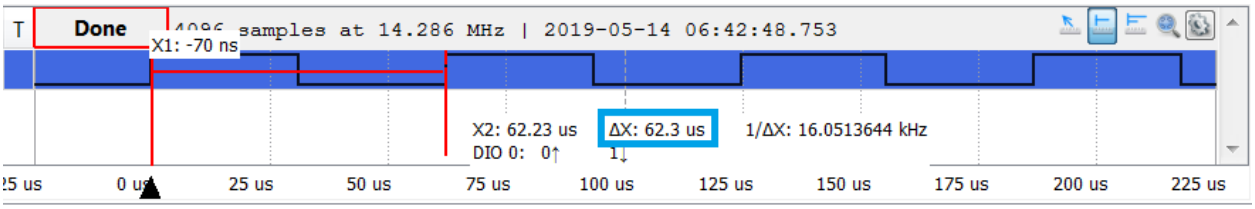


Figura 4. Tiempo de muestreo para la configuración 2.

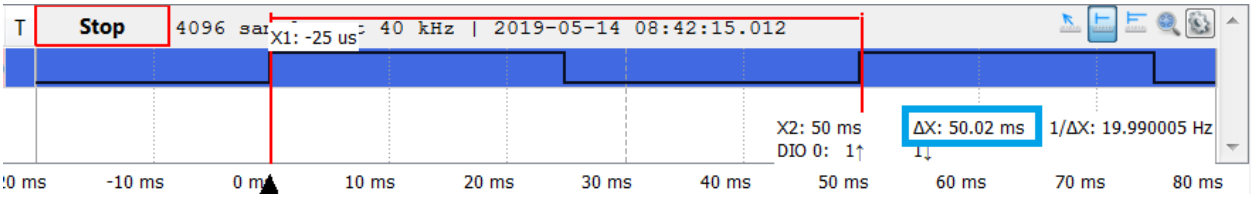


Figura 5. Tiempo de muestreo para la configuración 3.

Con objetivo de prueba, hemos conectado la tarea anterior del DAC con esta referente al AFEC. La primera es la señal de salida de la segunda (figura 5).

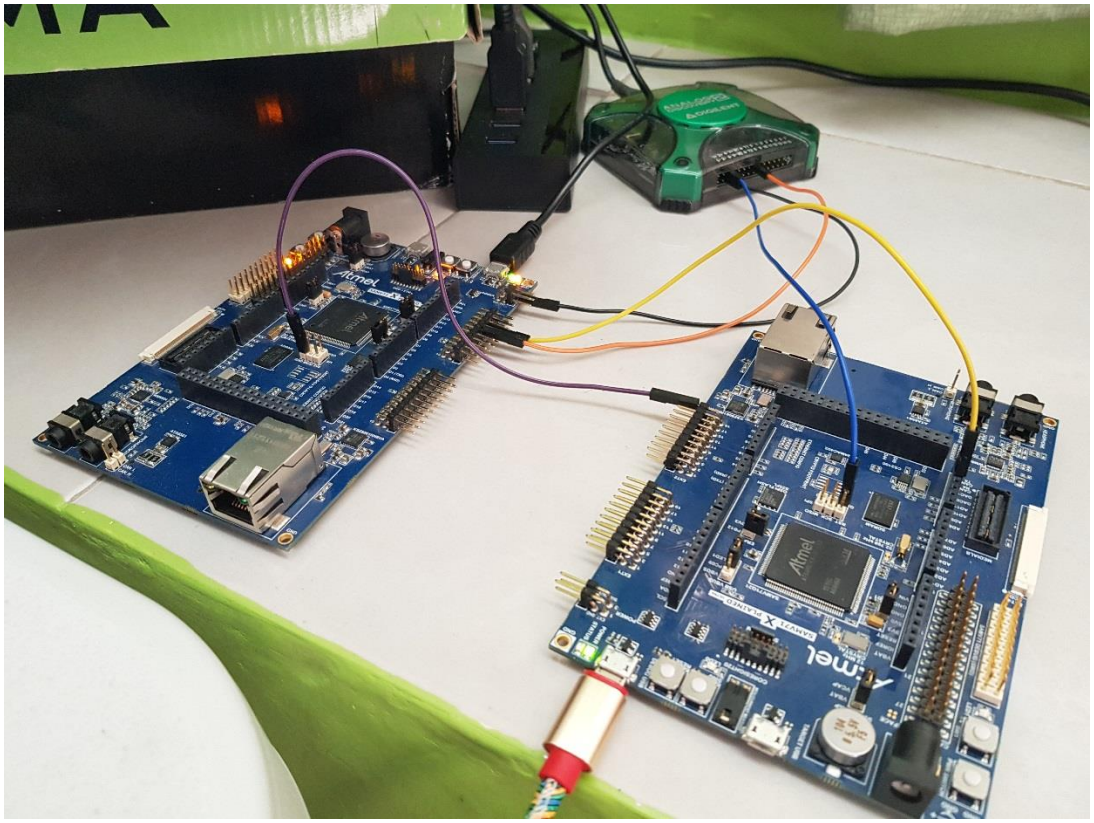


Figura 6. Conexión de 2 tarjetas SAM V71. Una genera una señal de salida que la otra toma como entrada.

La señal de entrada del AFEC es como la figura 6.

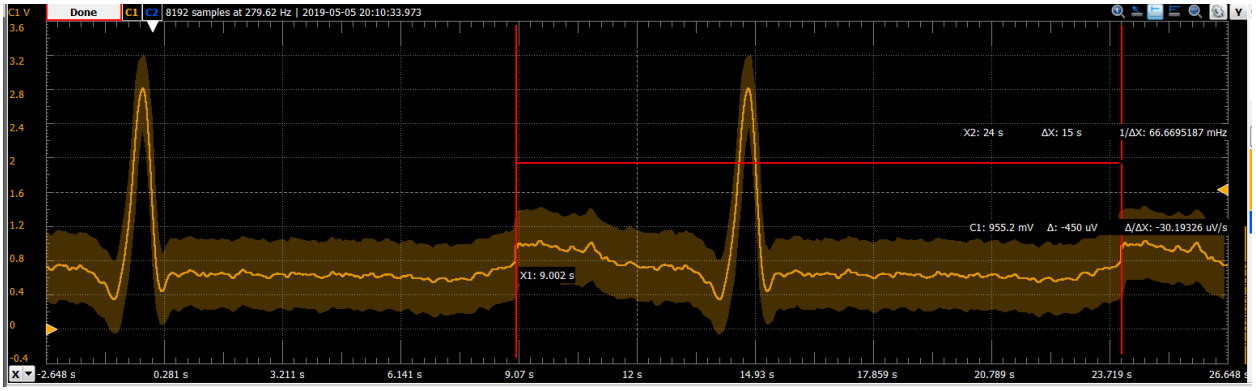


Figura 7. Señal de entrada al módulo AFEC.

En el archivo Graficas.xlsx se puede ver el resultado para las 3 configuraciones. Hemos extraído los arreglos del WinIdea para hacer estas gráficas, sin embargo, tenemos que decir que en este proyecto en específico en algunas ocasiones los valores de las variables mostradas por la ventana “Variables” eran incorrectos, notamos este comportamiento en más de 1 computadora. Debido a esto, los resultados podrían tener cierta cantidad de error.

En estas graficas son mostradas tanto la señal de entrada de la FFT, así como su señal de salida.

4. Conclusión

Definitivamente esta tarea fue la más retadora de todas. Esta engloba mucho de las anteriores y sin duda con lo que más tuvimos problema fue con la implementación del punto binario.

Vimos como el punto binario nos ayuda a hacer operación sin necesidad de usar punto flotante.

Otro gran reto fue la FFT. A pesar de verlo en clase, podemos decir que importa mucho tener un background al respecto puesto si no, se corre el riesgo de no entender al 100% esta tarea.

A pesar de que esta tarea no tuvo una arquitectura definida, tratamos de separar las funcionalidades en funciones para tener una mejor organización.

5. Referencias

1. Atmel (2016). SAM V71 Datasheet. Extraído de: http://ww1.microchip.com/downloads/en/DeviceDoc/Atmel-44003-32-bit-Cortex-M7-Microcontroller-SAM-V71Q-SAM-V71N-SAM-V71J_Datasheet.pdf, 5 de mayo de 2019.
2. Atmel-44046A-Using-AFE-in-SAM-SAM-V71-V70-E70-S70_Application Note-07/2015.
3. Projector de DAC + DAM Optimized. (adjunto a este)

5.3. **Apéndice C**

Métodos de simulación de circuito electrónicos

Autor: Adolfo Hernández Padilla
Periodo: 15/01/2018 al 14/04/2018.
Maestro: José Ernesto Rayas Sánchez, PhD.

"Simulación de circuito electrónico para un electro miógrafo"

Breve descripción.

El avance de la medicina en estas últimas décadas ha sido creciendo de una forma exponencial esto aumento debido al aumento de las investigaciones científicas y en su gran parte es debido al avance en cierta ciencia de la física que llamamos electrónica. La electrónica ha aportado un gran avance al respecto con tecnología la cual ha ayudado al aumento de equipos electrónicos que han enriquecido mucho al área médica.

Introducción.

Un equipo médico muy utilizado es un electro miógrafo el cual detecta señales nerviosas, musculares de un cuerpo humano. Pero presenta un problema desde el

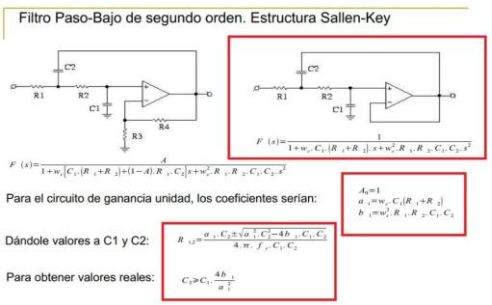
punto de vista de la electrónica, este es el ruido eléctrico, como también lo es el ruido de alta y baja frecuencia, además en muchos casos es costoso el uso de placas para la adquisición de datos. El desarrollo de este tema de investigación se enfocará a la adquisición de señales electromiografías para realizar la automatización de una unidad protésica mecánica de pierna de personas con amputación transfemoral que ya cuentan con su unidad protésica mecánica de extremidad inferior. Diseñar y desarrollar circuito electrónico donde se haga uso de electrodos de superficie para la lectura de la actividad eléctrica producida por los músculos esqueléticos que participan en el movimiento de flexión y extensión de la rodilla del extremo cicatrizado del miembro amputado (muñón).

Conceptualización del proyecto.

Como requerimos el diseño de este circuito para este electro miógrafo. Los valores en voltaje que manejan las lecturas electromiografías en la superficie de la piel se encuentran alrededor de los 10mV y 15mV, a una frecuencia de 500Hz para los músculos esqueléticos, en este caso el circuito servirá para la lectura de señales musculares (contracciones musculares) del grupo muscular en el muslo humano. Se utiliza un amplificador INA128 con una ganancia de 500, esto para poder trabajar los valores de salida (5v) en una tarjeta Arduino, (o Intel Edison) además un filtro pasa bajas de 8vo orden tipo Butterworth con topología SallenKey de ganancia unitaria para atenuar las frecuencias que se encuentran sobre los 500Hz. En los circuitos se homologa la tierra para no tener diferencia de potenciales. Estos circuitos son alimentados con un convertidor dcdc y las entradas al amplificador son tres señales que son V+, V-, y Vref. Según ciertos artículos se encuentra un ruido dentro de los 0 a 150Hz, pero utilizando otros diseños prototipo (con otros valores de componentes) pudimos apreciar que muchos de estos ruidos se reducen al tener el circuito bien aterrizado al cuerpo del paciente y utilizando además una batería (así se evita el ruido de los 60 Hz). Los capacitores del circuito son cerámicos through-hole, los resistores también son through-hole de 1/4 de watt. Los dos diseños son técnicamente los mismos, a única diferencia entre ellos es que un tiene un convertidos dcdc con posibilidad de alimentar otra tarjeta y otra tarjeta sin convertidor dcdc para tomar alimentación de manera externa.

Valores de los componentes

A continuación, se presentará los valores calculados del filtro pasa bajas 8vo orden, coeficientes filtro Butterworth 8vo orden en conjunto con las ecuaciones a la cual representa este filtro.



Ecuaciones y diseño básico para la formación de este filtro.

Valores del filtro de octavo orden.

C11 = 2.2e-6	**C21 =** 2.2e-6	**C31 =** 4.7e-7	**C41 =** 1.0e-7
C12 = 4.7e-6	**C22 =** 3.3e-6	**C32 =** 4.7e-6	**C42 =** 3.3e-6
C12Nom = 2.2870e-6	**C22Nom =** 3.1824e-6	**C32Nom =** 1.5228e-006	**C42Nom =** 2.6272e-006
R11b = 40.227	**R21b =** 97.587	**R31b =** 66.902	**R41b =** 340.60
R12b = 243.59	**R22b =** 143.01	**R32b =** 685.60	**R42b =** 901.44

Simulación en WSPICE.

Se Realizo la simulación de un circuito para un censado muscular para prótesis de pierna que se mencionó anterior mente, se basó en la técnica básico de un electro miógrafo.

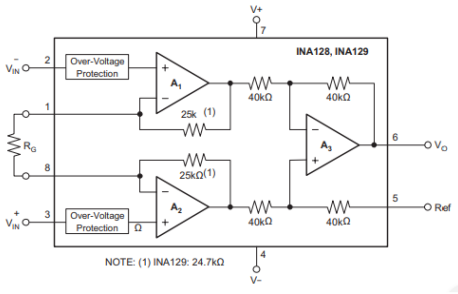
Este circuito está conformado por dos partes primero el primero básicamente es un amplificador de instrumentación y con su amplificador diferencial embebidos en un circuito integrados con matrícula 1NA148, llamado amplificador instrumental de baja potencia de precisión. El cual tiene una ganancia que está regida por la siguiente ecuación.

$$G = 1 + \frac{50K\Omega}{Rg};$$

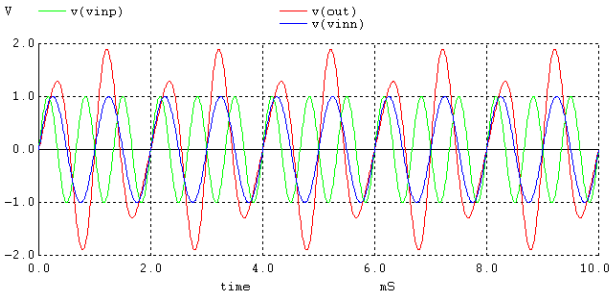
Para cuestión de este circuito se seleccionó la ganancia de 51 veces dando como valor de Rg de 1KΩ.

Amplificador de instrumentación.

Se muestra el circuito interno de este IC dando valores a sus nodos para poder realizar el análisis de wspice correspondiente para su simulación del circuito.



Bloques interno del 1NA148 amplificador de instrumentacion de baja potencia.

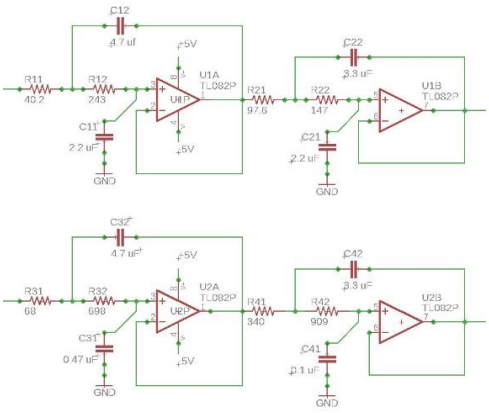


Cuando la señal positiva tiene una frecuencia mayor a la negativa se observa esta señal de salida.

Código en apéndice A.

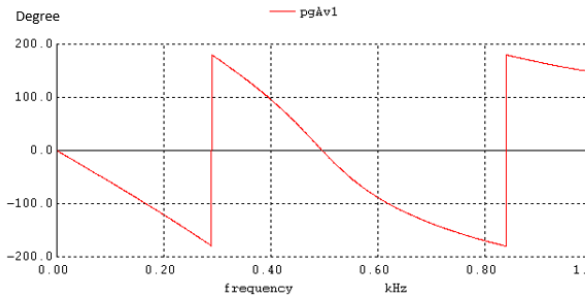
Filtro de Octavo Orden.

Se realiza la simulación del filtro de octavo orden, presentando el circuito con su nodos y datos para realizar la simulación en wspice.

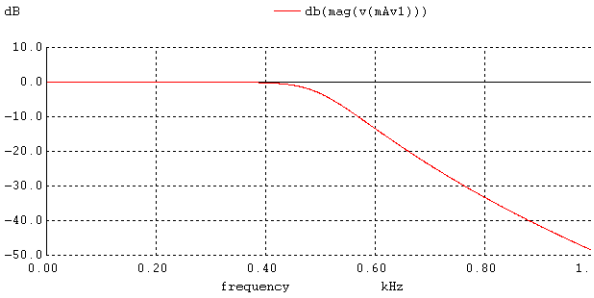


Este es el circuito completo que analizar

Código en apéndice B.

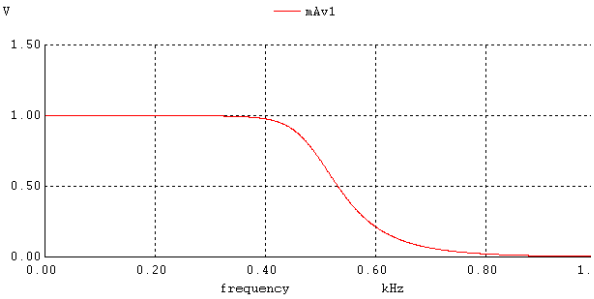


Respuesta de la ganancia del circuito en grados.



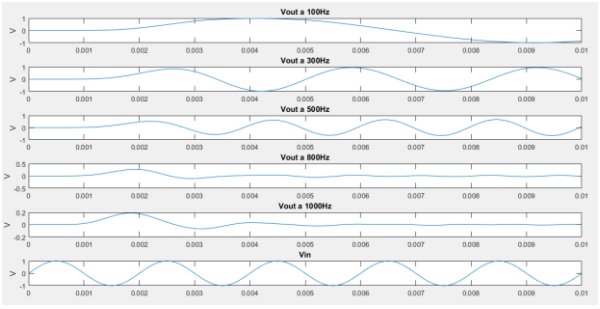
Respuesta de la ganancia del circuito en decibeles.

Aquí observaremos a que frecuencia de corte el circuito está funcionando a los -3decibeles. Es esta grafica se observa claramente que la frecuencia a -3 decibeles es de 495.4Hz. este tiene una variación de lo esperado de 0.92% por debajo de lo deseado es menor al 1% de variación de la señal de corte esperada.



Respuesta de la ganancia del circuito en magnitud.

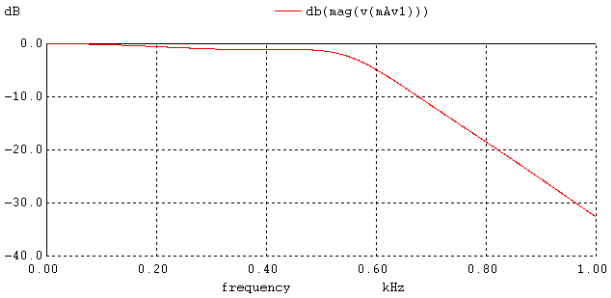
Si realizamos un análisis transitorio con 5 frecuencias diferentes. (100, 300, 500, 800 y 1000Hz) La respuesta al circuito a estas tres frecuencias fue la siguiente (Vout).



Se presenta la señal del circuito a respuesta transitoria a un chequeo de 10ms de la señal de salida. Este es la resulta en transitorios se observa que el circuito se estabiliza alrededor de entre 6 y 5 milisegundos.

Análisis con componentes ideales.

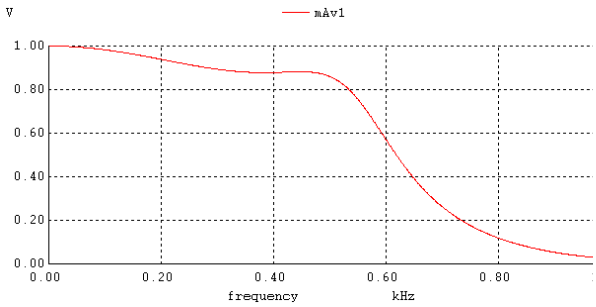
Analizaremos todo el filtro analógico Butterworth de octavo orden. (Operaciones Ideales.) El circuito será el mismo del inciso uno solamente se utilizará el operacional ideal.



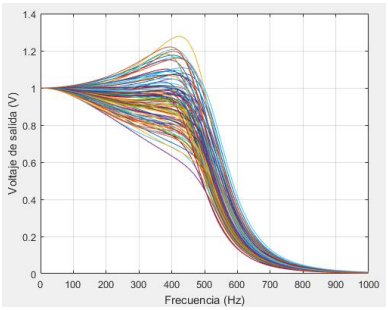
Respuesta del circuito en decibeles.

Código en apéndice B.

Aquí se observa que el circuito trabajara a una frecuencia de corte de 564Hz. Si usara operacionales ideales y componentes calculados.



Respuesta de la ganancia del circuito en magnitud.



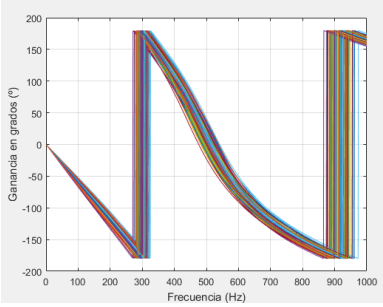
Respuesta de la ganancia del circuito en magnitud (Vout)

Análisis de Monte Carlos.

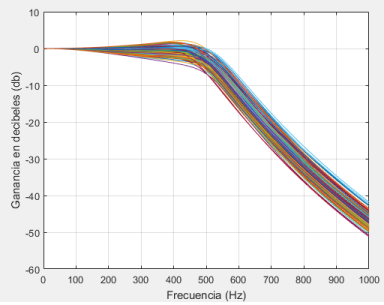
Se Realizará un análisis de Monte-Carlos con valores reales. Para esto obtuvieron los valores tolerancias de todos los componentes del circuito.

Primer Filtro	Segundo Filtro	Tercer Filtro	Cuarto Filtro
C11 = ± 20%,	C11 = ± 20%,	C11 = ± 10%,	C11 = ± 10%,
C12 = ± 10%,	C12 = ± 10%,	C12 = ± 10%,	C12 = ± 10%,
R11 = ± 1%,	R11 = ± 0.1%,	R11 = ± 1%,	R11 = ± 1%,
R12 = ± 1%,	R12 = ± 1%,	R12 = ± 1%,	R12 = ± 1%,

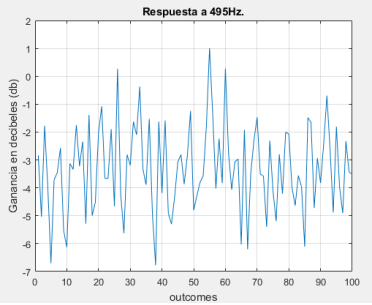
A continuación, realizamos la graficas con 100 outcomes del análisis de Monte-Carlos.



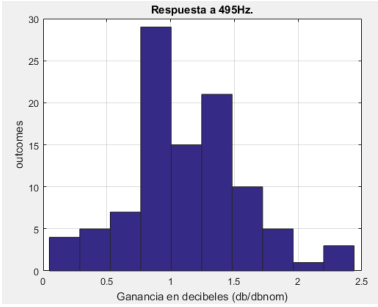
Respuesta de la ganancia del circuito en grados.



Respuesta del circuito en decibeles.

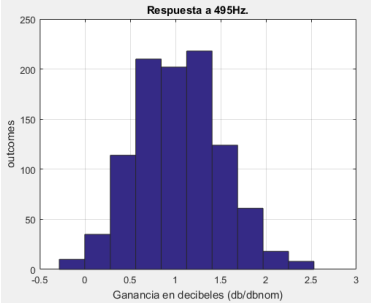


Respuesta de las señales en decibeles cuando toca las señales la respuesta de corte.

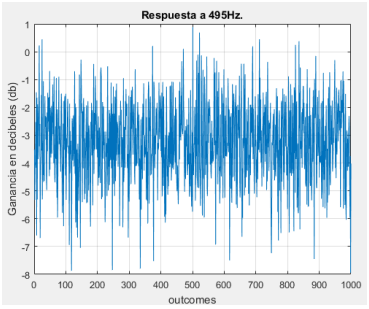


Este es histograma de esta repuesta. Con 100outcomes.

Ahora realizaremos el ejercicio a 1000 outcomes.



Histograma de la respuesta a decibeles a 1000 outcomes.



Respuesta de las señales en decibeles cuando toca las señales la respuesta de corte a 1000 outcomes.

Conclusiones

Se observo que el circuito que se simulo en este proyecto se pasó por varias etapas primero se desarrolló teóricamente el filtro esto no fue tan difícil ya que existía los datos ya obtenidos por la tesis en la cual se extrajo la información, pero lo que si fue un reto fue la implementación en wspice debido a que se tiene una versión si registro esta no limita al número de nodos que pudiéramos conectar y por lo cual el circuito no se pudo simular completamente. Por lo tanto, se divide el circuito en dos partes la parte del amplificador de instrumentación y la parte del filtro de octavo orden. En la parte del circuito de amplificador de instrumentación se descargó la Liberia directamente del proveedor pero el detalle fue que para utilizarla necesitamos la versión complete del wspice la cual no se tenía y esto provoca que no pudieras probar directamente la Liberia del fabricante debido a que no se cuenta con la licencia completa, debido a esto, se realizó la equivalencia en bloque de este IC y es como asís e pudo simular dando los resultados anterior mente mostrados en la parte superior. Bueno pasando a lo que se observó en el segundo circuito el filtro de octavo orden no represento mucho dificulta, pero así no trabajo ya que tenía muchas conexione y pudiéramos perdernos fácilmente pero aquí entra la parte de que si se pudieras hacer de forma gráfica se facilitaría mucho mejor la topología del circuito. Pero en fin este filtro se observó que con hacer pequeños cambios de valores de sus componentes afectábamos de una manera muy drástica la respuesta de este por lo cual es muy importante poner mucha atención con los valores correctos y que esto esté en su posición adecuada en el filtro completo. Aquí una vez realizado bien la conexión no se presentó ningún problema más que esto la simulación y el manejo de grafico fue ya dominado. El reto se vio en el driver ya que se tuvo que crear un arreglo donde se almacenaran todos las variables que se tenían en el código que se va a variar para su driver en Matlab aquí se tuvieron mucho problema que se fueron resolviendo como se iban presentándose, por ejemplo el problema de la ruta de donde correr el programa de wspice se aprendió que el folder de Matlab tenía que coincidir con las carpetas de dirección de instalación del wspice para correr el circuito el cual se tardó mucho en entender y solucionar la falla. parte que sería muy bueno mejorar esta parte en una segunda edición del controlador del wspice en Matlab. Otro problema se observó que en la versión 1.6 el nombre de una subrutina en wspice no afecta el nombre si es más de 6 caracteres, pero en la versión de 1.40 si afecta mucho y como es la que se utilizó para este practica si provoco grandes problemas a utilizarla. Generaba un error con el código **Internal error: txfree () Bad block en signature.** Este error me dio por que la versión de spice que utilizamos soporta un límite de palabras para poner nombre a una subrutina. Como comentario final fue una clase muy dinámica y me gustó mucho todos los temas impartidos y me fue de mucha ayuda para aprender y

fortalecer más mis conocimientos en electrónica y en si en

el análisis de circuitos.

Bibliografía:

[1] http://www.pardell.es/electromiografo.html
[2] https://desi.iteso.mx/erayas/cad.htm
[3] http://www.ti.com/lit/ds/symlink/ina129.pdf
[4] https://robologs.net/2016/02/11/emg-con-arduino-y-e-health-sensor-platform-parte-i-leer-los-electrodos/
[5] Tesis, "**Desarrollo e Implementación del prototipo de rotula 'Torres' para la automatización de una prótesis de extremidad inferior**"; Autor: Derek Torres Martínez. Universidad de Guadalajara, Dra. Ma. Del Roció Maciel Arellano.

Apéndice A.

Código en wspice. Del circuito.

```
################################################################################
Proyecto de método de simulación de circuito electrónicos
*simulación de un circuito EMG.
*------------------
*Adolfo Hernández
*------------------
.LIB library_models.txt
```

```
.LIB INA128.LIB

*descripción del circuito.
*declaración de fuentes de voltaje
* Descripción SIN.
* SIN (Voff Vamp Freq Td Df)
* Voff = Offset
* Vamp = voltaje de amplitud de la señal
* Freq = Frecuencia de la señal.
* Td = Tiempo de retardo.
* Df = Factor de amortiguamiento.
*Vs  in    0      DC  0.1V AC 1V SIN(0 1 100)
V++ vcc   0          DC  5V
V--       vee        0          DC -5V
Vs+ vinp  0      DC  1V AC 1V SIN(0 1 1000)
Vs- vinn  0      DC  1V AC 1V SIN(0 2 1500)
**********************************************************************************
*declaración de semiconductores.
* (REV N/A)     SUPPLY VOLTAGE: +/-15V
* CONNECTIONS:   NON-INVERTING INPUT
*              | INVERTING INPUT
*              | | POSITIVE POWER SUPPLY
*              | | | NEGATIVE POWER SUPPLY
*              | | | | OUTPUT
*              | | | | |
*.SUBCKT UA741    1 2 3 4 5
* CONNECTIONS:     NON-INVERTING INPUT
*                | INVERTING INPUT
*                | | OUTPUT
*                | | |
*                | | |
*                | | |
*.SUBCKT OpampIdeal  P N OUT
*----------------------
*OPERATIONAL AMPLIFIERS
*----------------------
* OPERATIONAL AMPLIFIER. SUBCIRCUIT
* QUASI-IDEAL
* CONNECTIONS:     NON-INVERTING INPUT
*                | INVERTING INPUT
*                | | OUTPUT
*                | | |
*                | | |
*                | | |
*.SUBCKT OpampIdeal  P N OUT
**********************************************************************************
```

```
*las terminales sustrato y surtido la cortocircuitamos.
*XOpA1  b  c   vcc   vee   c      UA741
*XOpA2  e  f   vcc   vee   f      UA741
*XOpA3  h  i   vcc   vee   i      UA741
*XOpA4  k  out vcc   vee   out    UA741
*XOpAmp1  b  c   c      OpampIdeal
*XOpAmp2  e  f   f      OpampIdeal
*XOpAmp3  h  i   i      OpampIdeal
*XOpAmp4  k  out out    OpampIdeal

XINA1  vinn  a1   vcc   vee   a3   UA741
XINA2  vinp  a2   vcc   vee   a4   UA741
XINA3  a6    a5   vcc   vee   a7   UA741

*declaración de componentes discretos.
*INA128 resistencia de ganancia
R1 a1 a3 24.7K
R2 a2 a4 24.7K
R3 a3 a5 40K
R4 a4 a6 40K
R5 a5 out 40K
R6 a6 0 40K
R7 out 0 100MEG
RG a1 a2 5

*OpAmp1
*R11    in  a   40.2
*R12    a   b   243
*C11    b   0   2.2uF
*C12    a   c   4.7uF
*R11    in  a   40.227
*R12    a   b   243.59
*C11    b   0   2.2870uF
*C12    a   c   4.7uF
*OpAmp2
*R21    c   d   97.6
*R22    d   e   147
*C21    e   0   2.2uF
*C22    d   f   3.3uF
*R21    c   d   97.587
*R22    d   e   147.01
*C21    e   0   2.2uF
*C22    d   f   3.31824uF
*OpAmp3
*R31    f   g   68
*R32    g   h   698
*C31    h   0   0.47uF
*C32    g   i   4.7uF
```

```
*R31    f   g   66.902
*R32    g   h   685.60
*C31    h   0   0.15228uF
*C32    g   i   4.7uF
*OpAmp4
*R41    i   j   340
*R42    j   k   909
*C41    k   0   0.1uF
*C42    j   out 3.3uF
*R41    i   j   340.60
*R42    j   k   901.44
*C41    k   0   0.1uF
*C42    j   out 2.6272uF
*Realizamos el análisis.

*Control
.control
   destroy all
   AC LIN 1000 1Hz 1KHz
   *Av1    = v(out)/v(in)
   *mAv1   = mag(Av1)
   *pAv1   = phase(Av1)
   *pgAv1  = (pAv1*180)/pi
   *write emg_no_dcdc_wspice_w_comm.csv v(out) v(in) Av1 pAv1 pgAv1
   plot vinp
   *plot Av1
   *plot mAv1
   *plot vdb(v(out))
   *plot pAv1
   *plot pgAv1
   *Description TRANS
   *TRANS <TSTEP> <TSTOP> <TSTART>
   *Por default inicia en tiempo 0 si no coloca tiempo inicial.
   *TRAN 10us 10ms
   *plot v(out) v(vinp) v(vinn)
   *write emg_no_dcdc_wspice_w_comm.csv v(out) v(in)
   *DC Vs 1mV 15mA 0.1mA
   *plot v(out)
   *OP
   *write op_results.csv all
   .endc
.end
################################################################################
```

Apéndice B.

Código en Matlab. Del circuito.

DRIVER

```
##########################################################################################
% ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
% Adolfo Hernández Padilla       May 06, 2018 ITESO 701988
% ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
%         Driving emg_no_dcdc_wspice_w_comm.cir from Matlab

function [y_outputs] = emg_no_dcdc_wspice_w_comm_driver(x)
   % forma de uso.
   % definición el vector "x"
   % x(1)= Voltaje de o0ffset de la señal entrada en volts.
   % x(2)= Amplitud de la señal cuadrada en votls.
   % x(3)= Frecuencia de la señal de entrada en Hz
   % x(4)= Voltaje de alimentación OpAmp en volts. (Para positivo y negativo mismo valor)
   % x(5)= R11 con valor default de 40.2ohms. (Valor en ohms)
   % x(6)= R12 con valor default de 243ohms. (Valor en ohms)
   % x(7)= C11 con valor default de 2.2uF. (valor de microfaradios)
   % x(8)= C12 con valor default de 4.7uF. (valor de microfaradios)
   % x(9)= R21 con valor default de 97.6ohms. (Valor en ohms)
   % x(10)= R22 con valor default de 147ohms. (Valor en ohms)
   % x(11)= C21 con valor default de 2.2uF. (valor de microfaradios)
   % x(12)= C22 con valor default de 3.3uF. (valor de microfaradios)
   % x(13)= R31 con valor default de 68ohms. (Valor en ohms)
   % x(14)= R32 con valor default de 698ohms. (Valor en ohms)
   % x(15)= C31 con valor default de 0.47uF. (valor de microfaradios)
   % x(16)= C32 con valor default de 4.7uF. (valor de microfaradios)
   % x(17)= R41 con valor default de 340ohms. (Valor en ohms)
   % x(18)= R42 con valor default de 909ohms. (Valor en ohms)
   % x(19)= C41 con valor default de 0.1uF. (valor de microfaradios)
   % x(20)= C42 con valor default de 3.3uF. (valor de microfaradios)
   % x(21)= TSETP fijo default 10us.
   % x(22)= TSETP fijo default 10ms.
   % x(23)= "N": Numero de muestras por división en análisis transitorio. (para cálculo de tstep
automático, TSTEP=TSTOP/N)
   % x(24)= "A": Bandera de cálculo de tiempo de simulación automático.
   %... dependiendo la frecuencia "Freq x(3)" y pasos de muestreo "N x(32)".
   %... Si este valor es "0" se probará con tiempo por default.
   %... Si este valor es "1" se realizará el cálculo automático,
   %... cualquier otro valor será por default.
   % x(25)= "OP": Bandera de selección de amplificador operación ideal o real.
   %... Si este valor es "0" se probará con valores ideales.
   %... Si este valor es "1" se probará con valores reales.
   %... cualquier otro valor será valores reales.
   % x(26)= Indicar el nombre(Matricula) del amplificador operación a utilizar según su subrutina.
ej:"UA741"
   % x(27)= Bandera de selector de simulación. Existen la siguiente opción.
```

```
%... Si este valor es "0" se realizará una simulación AC (Análisis de corriente alterna).
%... Si este valor es "1" se realizará una simulación OP (Punto de operación).
%... Si este valor es "2" se realizará una simulación TRAN (Análisis transitorio).
%... Si este valor es "3" se realizará una simulación DC (Análisis voltaje directo).
%... cualquier otro valor no se realizará un análisis.
% x(28)= Cantidad de punto para análisis de AC.
% x(29)= Frecuencia inicial de análisis de AC en Hz.
% x(30)= Frecuencia final de análisis de AC en Hz.
% x(31)= Tipo de análisis de análisis de AC.
%... Si este valor es "0" se realizará una simulación AC de forma décadas DEC
%... cualquier otro valor simulación AC de forma Lineal LIN.
% Define SPICE Script, ss
ss{1} = 'Projecto emg_no_dcdc_wspice_w_comm';
ss{2} = '.LIB library_models.txt';
ss{3} = ['Vs  in     0    DC  1V AC 1V SIN(' num2str(x(1)) 'V ' num2str(x(2)) 'V ' num2str(x(3)) 'Hz)'];
if x(25) == 0
   ss{4} = ['*No apply Vcc'];
   ss{5} = ['*No apply Vee'];
elseif x(25) == 1
   ss{4} = ['V++ vcc       0                DC       ' num2str(x(4)) 'V'];
   ss{5} = ['V--vee                0                DC       -' num2str(x(4)) 'V'];
else
   ss{4} = ['V++ vcc       0                DC       ' num2str(x(4)) 'V'];
   ss{5} = ['V--vee                0                DC       -' num2str(x(4)) 'V'];
end
if x(25) == 0
   ss{6} = 'XOP1  b  c   c      OpampIdeal';
   ss{7} = 'XOP2  e  f   f      OpampIdeal';
   ss{8} = 'XOP3  h  i   i      OpampIdeal';
   ss{9} = 'XOP4  k  out out    OpampIdeal';
elseif x(25) == 1
   ss{6} = 'XOP1  b  c   vcc   vee   c      UA741';
   ss{7} = 'XOP2  e  f   vcc   vee   f      UA741';
   ss{8} = 'XOP3  h  i   vcc   vee   i      UA741';
   ss{9} = 'XOP4  k  out vcc   vee   out    UA741';
else
   ss{6} = 'XOP1  b  c   vcc   vee   c      UA741';
   ss{7} = 'XOP2  e  f   vcc   vee   f      UA741';
   ss{8} = 'XOP3  h  i   vcc   vee   i      UA741';
   ss{9} = 'XOP4  k  out vcc   vee   out    UA741';
end
ss{10} = ['R11    in   a   ' num2str(x(5)) 'Ohms'];
ss{11} = ['R12    a    b   ' num2str(x(6)) 'Ohms'];
ss{12} = ['C11    b    0   ' num2str(x(7)) 'uF'];
ss{13} = ['C12    a    c   ' num2str(x(8)) 'uF'];
ss{14} = ['R21    c    d   ' num2str(x(9)) 'Ohms'];
```

```
ss{15} = ['R22    d    e   ' num2str(x(10)) 'Ohms'];
ss{16} = ['C21    e    0   ' num2str(x(11)) 'uF'];
ss{17} = ['C22    d    f   ' num2str(x(12)) 'uF'];
ss{18} = ['R31    f    g   ' num2str(x(13)) 'Ohms'];
ss{19} = ['R32    g    h   ' num2str(x(14)) 'Ohms'];
ss{20} = ['C31    h    0   ' num2str(x(15)) 'uF'];
ss{21} = ['C32    g    i   ' num2str(x(16)) 'uF'];
ss{22} = ['R41    i    j   ' num2str(x(17)) 'Ohms'];
ss{23} = ['R42    j    k   ' num2str(x(18)) 'Ohms'];
ss{24} = ['C41    k    0   ' num2str(x(19)) 'uF'];
ss{25} = ['C42    j    out ' num2str(x(20)) 'uF'];
ss{26} = '.control';
ss{27} = 'destroy all';
% (Analisis).
%... Si este valor es  (Analisis voltaje directo).
if x(27) == 0 %"0" se realizará una simulación AC
   if x(31) == 0 %Analisis de Decadas DEC
      ss{28} = ['AC DEC ' num2str(x(28)) ' ' num2str(x(29)) 'Hz ' num2str(x(30)) 'Hz'];
      ss{29} = 'Av     = v(out)/v(in)';
      ss{30} = 'dbAv   = vdb(Av)';
      ss{31} = 'pAv    = phase(Av)';
      ss{32} = 'pgAv   = (pAv*180)/pi';
      ss{33} = 'write ac_analisys.csv v(out) v(in) Av dbAv pgAv';
   else %análisis de décadas LIN
      ss{28} = ['AC LIN ' num2str(x(28)) ' ' num2str(x(29)) 'Hz ' num2str(x(30)) 'Hz'];
      ss{29} = 'Av     = v(out)/v(in)';
      ss{30} = 'dbAv   = vdb(Av)';
      ss{31} = 'magAv    = mag(Av)';
      ss{32} = 'pgAv   = (phase(Av)*180)/pi';
      ss{33} = 'write ac_analisys.csv v(out) v(in) Av dbAv pgAv magAv';
   end
elseif x(27) == 1 %"1" se realizará una simulación OP.
   ss{28} = 'OP';
   ss{29} = '';
   ss{30} = '';
   ss{31} = '';
   ss{32} = '';
   ss{33} = 'write op_analisys.csv v(out) v(in)';
elseif x(27) == 2 %"2" se realizará una simulación TRAN.
   if x(24) == 0
      ss{28} = 'TRAN 0.00001s 0.01s';
      ss{29} = '';
      ss{30} = '';
      ss{31} = '';
      ss{32} = '';
      ss{33} = 'write tran_analisys.csv v(out) v(in)';
   elseif x(24) == 1
      TSTOP=(1/x(3))*10;
```

```
        TSTEP=TSTOP/x(23);
        % el valor natural 10 reseprenta la veces que se muestra la salida con respecto a la frecuencia de
entrada.
        ss{28} = ['TRAN ' num2str(TSTEP) 's ' num2str(TSTOP) 's'];
        ss{29} = '';
        ss{30} = '';
        ss{31} = '';
        ss{32} = '';
        ss{33} = 'write tran_analisys.csv v(out) v(in)';
      else
        ss{28} = 'TRAN 0.00001s 0.01s';
        ss{29} = '';
        ss{30} = '';
        ss{31} = '';
        ss{32} = '';
        ss{33} = 'write tran_analisys.csv v(out) v(in)';
      end
    elseif x(27) == 3 %"3" se realizará una simulación DC.
      ss{28} = 'DC Vs 1mV 15mA 0.1mA';
      ss{29} = '';
      ss{30} = '';
      ss{31} = '';
      ss{32} = '';
      ss{33} = 'write dc_analisys.csv v(out) v(in) v(s)';
    else
      ss{28} = '*None activity simulation';
      ss{29} = '';
      ss{30} = '';
      ss{31} = '';
      ss{32} = '';
      ss{33} = '*None ';
    end
    ss{34} = 'quit';
    ss{35} = '.endc';
    ss{36} = '.end';
    % ----- Save SPICE Script as a Circuit File in Matlab Working Directory
    CircuitFileName = 'emg_no_dcdc_wspice_w_comm.cir';
    ckt_file = str2mat(ss);
    [rows,columns] = size(ckt_file);
    % ----- File identifier opened.
    fid = fopen(CircuitFileName,'w+');
    for i = 1:rows
      fprintf(fid, '%s', ckt_file(i,:));
      fprintf(fid, '%s\r\n', '');
    end
    % ----- File identifier closed.
```

```
  fclose(fid);
  % ----- Run WinSpice Circuit File
  ExecFile = 'C:\wspice3\wspice3 ';
  [status,cmdout]=system([ExecFile CircuitFileName]);
  %status
  % ----- Read WinSpice Output Files
  if x(27) == 0 %"0" se realizara una simulación AC
     Resp       = csvread('ac_analisys.csv',1,0);
     Freq       = Resp(:,1);
     Vout       = Resp(:,3);
     Vin        = Resp(:,5);
     AvMag      = Resp(:,7);
     AvPas      = Resp(:,8);
     Avdb       = Resp(:,9);
     Avdeg      = Resp(:,11);
     Avmag1     = Resp(:,13);
     y_outputs   = [Freq Vout Vin AvMag AvPas Avdb Avdeg Avmag1];
  elseif x(27) == 1 %"1" se realizará una simulación OP.
     %ss{29} = 'write op_analisys.csv v(out) v(in)';
  elseif x(27) == 2 %"2" se realizará una simulación TRAN.
     Resp       = csvread('tran_analisys.csv',1,0);
     Time       = Resp(:,1);
     Vout       = Resp(:,2);
     Vin        = Resp(:,3);
     y_outputs   = [Time Vout Vin];
  elseif x(27) == 3 %"3" se realizará una simulación DC.
     %ss{29} = 'write dc_analisys.csv v(out) v(in) v(s)';
  else
     %ss{29} = '*None ';
  end
  % Erase WinSpice Output Files
  if x(27) == 0 %"0" se realizara una simulación AC
     delete ac_analisys.csv;
  elseif x(27) == 1 %"1" se realizará una simulación OP.
     delete op_analisys.csv;
  elseif x(27) == 2 %"2" se realizará una simulación TRAN.
     delete tran_analisys.csv;
  elseif x(27) == 3 %"3" se realizará una simulación DC.
     delete dc_analisys.csv;
  else
     delete ac_analisys.csv;
     delete op_analisys.csv;
     delete tran_analisys.csv;
     delete dc_analisys.csv;
  end
##########################################################################################
```

TESTER

```
################################################################################
% ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
% Adolfo Hernandez Padilla       May 06, 2018 ITESO 701988
% ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
%         Tester driving emg_no_dcdc_wspice_w_comm.cir from Matlab
% Asignación de valores de las variables del driver.
x=zeros(1,31);
% x(1)= Voltaje de o0ffset de la señal entrada en volts.
x(1)=0;
% x(2)= Amplitud de la señal cuadrada en votls.
x(2)=1;
% x(3)= Frecuencia de la señal de entrada en Hz
x(3)=1000;
% x(4)= Voltaje de alimentación OpAmp en volts. (Para positivo y negativo mismo valor)
x(4)=5;
% x(5)= R11 con valor default de 40.2ohms. (Valor en ohms)
x(5)=40.2;
% x(6)= R12 con valor default de 243ohms. (Valor en ohms)
x(6)=243;
% x(7)= C11 con valor default de 2.2uF. (valor de microfaradios)
x(7)=2.2;
% x(8)= C12 con valor default de 4.7uF. (valor de microfaradios)
x(8)=4.7;
% x(9)= R21 con valor default de 97.6ohms. (Valor en ohms)
x(9)=97.6;
% x(10)= R22 con valor default de 147ohms. (Valor en ohms)
x(10)=147;
% x(11)= C21 con valor default de 2.2uF. (valor de microfaradios)
x(11)=2.2;
% x(12)= C22 con valor default de 3.3uF. (valor de microfaradios)
x(12)=2.2;
% x(13)= R31 con valor default de 68ohms. (Valor en ohms)
x(13)=68;
% x(14)= R32 con valor default de 698ohms. (Valor en ohms)
x(14)=698;
% x(15)= C31 con valor default de 0.47uF. (valor de microfaradios)
x(15)=0.47;
% x(16)= C32 con valor default de 4.7uF. (valor de microfaradios)
x(16)=4.7;
% x(17)= R41 con valor default de 340ohms. (Valor en ohms)
x(17)=340;
% x(18)= R42 con valor default de 909ohms. (Valor en ohms)
x(18)=909;
% x(19)= C41 con valor default de 0.1uF. (valor de microfaradios)
x(19)=0.1;
% x(20)= C42 con valor default de 3.3uF. (valor de microfaradios)
```

```
x(20)=3.3;
% x(21)= TSETP fijo default 10us.
x(21)=0.00001;
% x(22)= TSETP fijo default 10ms.
x(22)=0.01;
% x(23)= "N": Numero de muestras por división en análisis transitorio. (para cálculo de tstep automático,
TSTEP=TSTOP/N)
x(23)=500;
% x(24)= "A": Bandera de cálculo de tiempo de simulación automático.
%... dependiendo la frecuencia "Freq x(3)" y pasos de muestreo "N x(32)".
%... Si este valor es "0" se probará con tiempo por default.
%... Si este valor es "1" se realizará el cálculo automático,
%... cualquier otro valor será por default.
x(24)=0;
% x(25)= "OP": Bandera de selección de amplificador operacional ideal o real.
%... Si este valor es "0" se probará con valores ideales.
%... Si este valor es "1" se probará con valores reales.
%... cualquier otro valor serán reales.
x(25)=1;
% x(26)= Indicar el nombre(Matricula) del amplificador operacional a utilizar según su subrutina.
ej:"UA741"
x(26)=0; % No es utilizable porque es un string.
% x(27)= Bandera de selector de simulación. Existen la siguiente opción.
%... Si este valor es "0" se realizará una simulación AC (análisis de corriente alterna).
%... Si este valor es "1" se realizará una simulación OP (Punto de operación).
%... Si este valor es "2" se realizará una simulación TRAN (análisis transitorio).
%... Si este valor es "3" se realizará una simulación DC (análisis voltaje directo).
%... cualquier otro valor no se realizará un análisis.
x(27)=0;
% x(28)= Cantidad de punto para análisis de AC.
x(28)=1000;
% x(29)= Frecuencia inicial de análisis de AC en Hz.
x(29)=1;
% x(30)= Frecuencia final de análisis de AC en Hz.
x(30)=1000;
% x(31)= Tipo de análisis de análisis de AC.
%... Si este valor es "0" se realizará una simulación AC en décadas DEC
%... cualquier otro valor simulación AC de forma Lineal LIN.
x(31)=1;
% Mandar a Llamar a la función.
%[f,Av,Rest] = emg_no_dcdc_wspice_w_comm_driver(x);
if x(27) == 0 %"0" se realizará una simulación AC
   [y_outputs] = emg_no_dcdc_wspice_w_comm_driver(x);
   FreqNom     = y_outputs(:,1);
   VoutNom     = y_outputs(:,2);
   VinNom      = y_outputs(:,3);
   AvMagNom    = y_outputs(:,4);
   AvPhaNom    = y_outputs(:,5);
```

```
AvdbNom     = y_outputs(:,6);
AvdegNom    = y_outputs(:,7);
Avmag1Nom   = y_outputs(:,8);
% análisis de Monte Carlos.
% Definimos los outcomes y tolerancias a trabajar.
N=100;
% Damos el porcentaje de tolerancia de los componentes.
%OpAmp1
t_R11=0.01; % 1% de Tolerancia.
t_R12=0.01; % 1% de Tolerancia.
t_C11=0.20; % 20% de Tolerancia.
t_C12=0.10; % 10% de Tolerancia.
%OpAmp2
t_R21=0.001;% 0.1% de Tolerancia.
t_R22=0.1;  % 1% de Tolerancia.
t_C21=0.20; % 20% de Tolerancia.
t_C22=0.10; % 10% de Tolerancia.
%OpAmp3
t_R31=0.01; % 1% de Tolerancia.
t_R32=0.01; % 1% de Tolerancia.
t_C31=0.10; % 10% de Tolerancia.
t_C32=0.10; % 10% de Tolerancia.
%OpAmp4
t_R41=0.01; % 1% de Tolerancia.
t_R42=0.01; % 1% de Tolerancia.
t_C41=0.10; % 10% de Tolerancia.
t_C42=0.10; % 10% de Tolerancia.
% Creamos el arreglo de tolerancias.
tau=[t_R11   t_R12   t_C11   t_C12   t_R21   t_R22   t_C21   t_C22   t_R31   t_R32   t_C31   t_C32
      t_R41   t_R42   t_C41   t_C42];
%Generamos los outcomes aleatorios.
Y=zeros(N,length(x));
for j=1:N
   Y(j,:)=x;
   Y(j,4)=x(4)*((1+tau(1)*(2*rand-1)));
   Y(j,5)=x(5)*((1+tau(2)*(2*rand-1)));
   Y(j,6)=x(6)*((1+tau(3)*(2*rand-1)));
   Y(j,7)=x(7)*((1+tau(4)*(2*rand-1)));
   Y(j,8)=x(8)*((1+tau(5)*(2*rand-1)));
   Y(j,9)=x(9)*((1+tau(6)*(2*rand-1)));
   Y(j,10)=x(10)*((1+tau(7)*(2*rand-1)));
   Y(j,11)=x(11)*((1+tau(8)*(2*rand-1)));
   Y(j,12)=x(12)*((1+tau(9)*(2*rand-1)));
   Y(j,13)=x(13)*((1+tau(10)*(2*rand-1)));
   Y(j,14)=x(14)*((1+tau(11)*(2*rand-1)));
   Y(j,15)=x(15)*((1+tau(12)*(2*rand-1)));
```

```
   Y(j,16)=x(16)*((1+tau(13)*(2*rand-1)));
   Y(j,17)=x(17)*((1+tau(14)*(2*rand-1)));
   Y(j,18)=x(18)*((1+tau(15)*(2*rand-1)));
   Y(j,19)=x(19)*((1+tau(16)*(2*rand-1)));
end
%Calculamos la respuesta a los valores de outcomes generados.
%FreqNom     = y_outputs(:,1);
%VoutNom     = y_outputs(:,2);
%VinNom      = y_outputs(:,3);
%AvMagNom    = y_outputs(:,4);
%AvPhaNom    = y_outputs(:,5);
%AvdbNom     = y_outputs(:,6);
%AvdegNom    = y_outputs(:,7);
%Avmag1Nom   = y_outputs(:,8);
%Inicializamos los arreglos iniciales.
Vout_rand   = zeros(N,length(FreqNom));
Av_rand     = zeros(N,length(FreqNom));
AvMag_rand  = zeros(N,length(FreqNom));
AvPas_rand  = zeros(N,length(FreqNom));
Avdb_rand   = zeros(N,length(FreqNom));
Avdeg_rand  = zeros(N,length(FreqNom));
AvMag1_rand = zeros(N,length(FreqNom));
Av_rand_min = zeros(N,1);
Av_rand_his = zeros(N,1);
for j=1:N
   [y_outputs] = emg_no_dcdc_wspice_w_comm_driver(Y(j,:));
   % Señales básicas.
   Freq       = y_outputs(:,1);
   Vin        = y_outputs(:,3);
   % Se□s salidas.
   Vout_rand(j,:)   = y_outputs(:,2);
   AvMag_rand(j,:)  = y_outputs(:,4);
   AvPas_rand(j,:)  = y_outputs(:,5);
   Av_rand(j,:)     = AvMag_rand(j,:)+(AvPas_rand(j,:)*sqrt(-1));
   Avdb_rand(j,:)   = y_outputs(:,6);
   Avdeg_rand(j,:)  = y_outputs(:,7);
   AvMag1_rand(j,:)  = y_outputs(:,8);
   %disp (['N: ' num2str(j) ]);
   for z=1:length(FreqNom)
      if (z == 495)
         Av_rand_min(j,:) = Avdb_rand(j,z);
         Av_rand_his(j,:) = Avdb_rand(j,z)/AvdbNom(z)
         %disp "Entro";
      end
   end
end
figure
plot(Freq,AvMag1_rand)
```

```
    xlabel('Frecuencia (Hz)');
    ylabel('Voltaje de salida (V)');
    grid on
    figure
    plot(Freq,Av_rand)
    xlabel('Frecuencia (Hz)');
    ylabel('Ganancia(V/V)');
    grid on
    figure
    plot(Freq,Avdb_rand)
    xlabel('Frecuencia (Hz)');
    ylabel('Ganancia en decibeles (db)');
    grid on
    figure
    plot(Freq,Avdeg_rand)
    xlabel('Frecuencia (Hz)');
    ylabel('Ganancia en grados (ک');
    grid on

    figure
    plot(Av_rand_min)
    title('Respuesta a 495Hz.')
    xlabel('outcomes');
    ylabel('Ganancia en decibeles (db)');
    grid on

    figure
    hist(Av_rand_his,10)
    title('Respuesta a 495Hz.')
    xlabel('Ganancia en decibeles (db/dbnom)');
    ylabel('outcomes');
    grid on
elseif x(27) == 1 %"1" se realizará una simulación OP.
    %ss{29} = 'write op_analisys.csv v(out) v(in)';
elseif x(27) == 2 %"2" se realizará una simulación TRAN.
    [y_outputs] = emg_no_dcdc_wspice_w_comm_driver(x);
    Time       = y_outputs(:,1);
    Vout       = y_outputs(:,2);
    Vin        = y_outputs(:,3);
    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    figure % new figure
    ax1 = subplot(6,1,1); % top subplot
    ax2 = subplot(6,1,2); % bottom subplot
    ax3 = subplot(6,1,3); % bottom subplot
    ax4 = subplot(6,1,4); % bottom subplot
    ax5 = subplot(6,1,5); % bottom subplot
```

```
ax6 = subplot(6,1,6); % bottom subplot
% Create a 2-D line plot in each axes by referring to the axes
% handles. Add a title and _y_-axis label to each axes by passing the axes
% handles to the |title| and |ylabel| functions.
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% x(3)= Frecuencia de la se□de entrada en Hz
x(3)=100;
[y_outputs_1] = emg_no_dcdc_wspice_w_comm_driver(x);
Time_1       = y_outputs_1(:,1);
Vout_1       = y_outputs_1(:,2);
Vin_1        = y_outputs_1(:,3);
% x(3)= Frecuencia de la señal de entrada en Hz
x(3)=300;
[y_outputs_2] = emg_no_dcdc_wspice_w_comm_driver(x);
Time_2       = y_outputs_2(:,1);
Vout_2       = y_outputs_2(:,2);
Vin_2        = y_outputs_2(:,3);
% x(3)= Frecuencia de la señal de entrada en Hz
x(3)=500;
[y_outputs_3] = emg_no_dcdc_wspice_w_comm_driver(x);
Time_3       = y_outputs_3(:,1);
Vout_3       = y_outputs_3(:,2);
Vin_3        = y_outputs_3(:,3);
% x(3)= Frecuencia de la se□de entrada en Hz
x(3)=800;
[y_outputs_4] = emg_no_dcdc_wspice_w_comm_driver(x);
Time_4       = y_outputs_4(:,1);
Vout_4       = y_outputs_4(:,2);
Vin_4        = y_outputs_4(:,3);
% x(3)= Frecuencia de la señal de entrada en Hz
x(3)=1000;
[y_outputs_5] = emg_no_dcdc_wspice_w_comm_driver(x);
Time_5       = y_outputs_5(:,1);
Vout_5       = y_outputs_5(:,2);
Vin_5        = y_outputs_5(:,3);
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
plot(ax1,Time_1,Vout_1)
title(ax1,'Vout a 100Hz')
ylabel(ax1,'V')
plot(ax2,Time_2,Vout_2)
title(ax2,'Vout a 300Hz')
ylabel(ax2,'V')
plot(ax3,Time_3,Vout_3)
title(ax3,'Vout a 500Hz')
ylabel(ax3,'V')
plot(ax4,Time_4,Vout_4)
title(ax4,'Vout a 800Hz')
ylabel(ax4,'V')
```

```
   plot(ax5,Time_5,Vout_5)
   title(ax5,'Vout a 1000Hz')
   ylabel(ax5,'V')
   plot(ax6,Time,Vin)
   title(ax6,'Vin')
   ylabel(ax6,'V')
   %%%%%%%%%%%%%%%%%%%%%%%%%%%%%
   figure
   plot(Time,Vout,'-b',Time,Vin,'-r')
   xlabel('Tiempo (s)');
   ylabel('Voltaje de salida (V)');
   grid on
elseif x(27) == 3 %"3" se realizará una simulación DC.
   %ss{29} = 'write dc_analisys.csv v(out) v(in) v(s)';
else
   %ss{29} = '*None ';
end
##########################################################################################
```