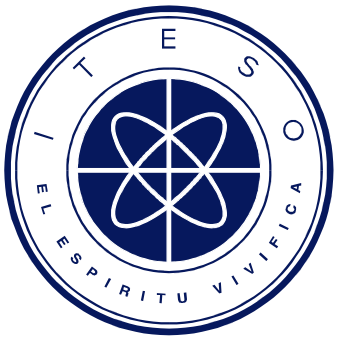
INSTITUTO TECNOLÓGICO Y DE ESTUDIOS SUPERIORES DE OCCIDENTE

Reconocimiento de validez oficial de estudios de nivel superior según acuerdo secretarial 15018, publicado en el Diario Oficial de la Federación el 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

Trabajo recepcional que para obtener el grado de

MAESTRO EN DISEÑO ELECTRÓNICO

Presenta: Carlos Alberto Musich Cuevas

Asesor: José Luis Chávez Hurtado

San Pedro Tlaquepaque, Jalisco. Agosto de 2017.

Contenido

Introducción

Este documento explica el diseño y desarrollo de los proyectos más relevantes de las materias de Sistemas Operativos Embebidos, Procesamiento Digital de Señales y Sistemas Embebidos respectivamente. Estos proyectos fueron escogidos por que representaron un mayor reto en su implementación y por ende un aprendizaje más significativo. Por otro lado, el área de concentración que se escogió fue la de Sistemas Embebidos debido a que es el área en la que me he desarrollado en el lado profesional y los conocimientos adquiridos no se han quedado sólo en teoría, sino que los he puesto en práctica en la parte laboral.

El primero consiste en la implementación de un sistema de control y monitoreo remoto utilizando como medio la red social Twitter. Se utiliza la tarjeta de desarrollo TWR-K60N512 de Freescale la cual se conecta al servidor de Twitter utilizando el paquete de desarrollo de TCP sobre IP que provee el sistema operativo en tiempo real MQX. La aplicación consiste en leer los mensajes de una cuenta de Twitter e interpretarlos, y si estos mensajes son comandos conocidos por el sistema, se ejecuta alguna acción. Así mismo la aplicación debe enviar un mensaje desde la misma cuenta de Twitter cada que se presiona un botón en la tarjeta o cada que el potenciómetro integrado a esta misma tarjeta cambia de valor.

El segundo consiste en la implementación de un ecualizador digital de 3 bandas con ganancia variable, desarrollado usado la tarjeta de TMDSDSK6713 de Spectrum Digital. El objetivo de este proyecto es aplicar el conocimiento en el diseño de filtros digitales mediante la técnica de ventaneo para filtros de respuesta al impulso finita (FIR) y su implementación en un sistema embebido para una aplicación de audio.

El tercer proyecto consiste en la implementación de una ecuación para obtener la aceleración de un vehículo aéreo no tripulado. Uno de los retos en este proyecto es obtener los datos involucrados en la ecuación desde medios externos, uno por SPI y el otro es una señal analógica que debe pasar por el ADC del microcontrolador. El microcontrolador que se utiliza en este proyecto es el MSP430G2231 el cual está muy limitado en memoria, por lo que el segundo reto es la optimización y el manejo de los recursos sin perder precisión al resolver la ecuación ya que ésta contiene funciones trigonométricas y utiliza datos en punto flotante.

1. Resumen de los proyectos realizados

1.1. Control Remoto Sobre Ethernet a través de Twitter

1.1.1 Introducción

Hoy en día las redes sociales son un tema muy popular y gran parte de la población está familiarizada con ellas, es por esto que en este proyecto se pretende tomar ventaja de su uso.

La facilidad, la conveniencia e incluso la necesidad de controlar y monitorear nuestra casa desde cualquier lugar en el que estemos es lo que da vida a esta propuesta que incluye la posibilidad de encender y apagar las luces de un edificio, abrir y cerrar puertas o ventanas además de tener el conocimiento de lo que está pasando en el inmueble mediante la lectura de sensores, todo a través de una cuenta personal de la red social Twitter.

Este proyecto consiste en el diseño e implementación de un cliente HTTP embebido en un microcontrolador de 32 bits con la finalidad de controlar y monitorear un sistema a través de Ethernet, el cual se conecta al servidor de Twitter que se utiliza como plataforma para el sistema de control y monitoreo.

1.1.2 Antecedentes

En la actualidad existen aplicaciones similares, dentro de la investigación que se realizó, las aplicaciones desarrolladas con las plataformas Arduino y Raspberry Pi son las más populares, sin embargo, hay significativas diferencias y ventajas en la forma en que está desarrollada esta aplicación. En principio las plataformas Arduino y Raspberry Pi fueron desarrolladas con el objetivo de estimular la enseñanza en ciencias de la ingeniería y computación, por lo que ninguna de las dos se consideran plataformas de desarrollo para productos comerciales. Hablando específicamente de Raspberry Pi, utiliza procesadores de Broadcom basados en ARM11 lo que implica un costo mucho más elevado, mayor consumo de potencia y la necesidad de un sistema operativo basado en Linux ya que su implementación está hecha en Python que es un lenguaje de alto nivel. Por otro lado, la implementación en Arduino es más similar a la de este proyecto ya que

Arduino se basa en un microcontrolador AVR de Atmel que se puede considerar dentro de la misma gama de los microcontroladores Kinetis, sin embargo, utiliza una librería específica para publicar mensajes en Twitter lo cual limita el alcance y la flexibilidad de la aplicación pues no tiene disponible ningún servicio genérico de TCP/IP a nivel de usuario.

Tomando lo anterior en cuenta se puede considerar que la integración de las herramientas y los servicios utilizados en este proyecto es una combinación optima y única ya que incluye el equilibrio entre precio, procesamiento y consumo de potencia que ofrece un microcontrolador Kinetis, más el alcance que ofrece el sistema operativo en tiempo real MQX con su librería de TCP/IP para manejar libremente la comunicación por Ethernet y la facilidad de uso de un tercer servicio llamado SuperTweet para poder encriptar la información y enviar un mensaje al servidor de Twitter.

1.1.3 Solución desarrollada

Para desarrollar el sistema se utilizó el microcontrolador K60N512 de la familia Kinetis de Freescale. Sobre este dispositivo se monta el Sistema Operativo en Tiempo Real MQX, el cual provee los servicios de manejador de tareas, mutex, controladores de GPIOs y ADC, funciones de fecha y hora, así como librerías de TCP sobre IP que fueron utilizadas en este proyecto.

La implementación de Software consta de 3 tareas, la tarea ‘Main’, ‘HTTP_Client_GET’ y ‘HTTP_Client_Post’.

La primera inicializa los puertos de propósito general e inicia el servicio de TCP/IP provisto por la librería RTCS de MQX, finalmente entra en un ciclo infinito donde se encarga de monitorear las lecturas de los sensores y los pines de entrada, así como reiniciar periódicamente la tarea ‘HTTP_Client_GET’, cuyo propósito es llevar a cabo la comunicación con el servidor de Twitter leyendo el último mensaje publicado por la cuenta de Twitter que fue creada específicamente para este proyecto, y analizándolo para ejecutar alguna acción si es que se reconoce como un comando válido. Por último, la tarea ‘HTTP_Client_Post’ se encarga de la comunicación con el servidor de Twitter con el fin de enviar mensajes desde el microcontrolador. Para llevar a cabo esta tarea se utiliza un tercer elemento que es el servicio de Super Tweet.

En el apéndice A se puede apreciar un diagrama de bloques, así como los diagramas de flujo de las tres tareas con su explicación detallada.

1.1.4 Análisis de Resultados

El cliente HTTP embebido en el microcontrolador de 32 bits Kinetis K60 de Freescale es capaz de leer los mensajes enviados por el usuario a la cuenta de Twitter "FSL_K60". Utilizando comandos predefinidos por el usuario la aplicación es capaz de interpretar los mensajes y ejecutar comandos como encender y apagar diodos luminosos o una bombilla de 60 Watts, así como controlar el sentido del giro de un motor de corriente directa de 12V interfazados con el microcontrolador a través de una etapa de potencia. También se cuenta con un potenciómetro y un botón para simular entradas asíncronas al sistema las cuales son notificadas al usuario a través de la misma cuenta de Twitter.

Para probar la solución implementada, la aplicación fue configurada para leer el último tweet cada 30 segundos y se enviaron todos los comandos reconocidos por el sistema repetidas veces. Después de varias sesiones de prueba y depurar algunos errores finalmente todos los comandos fueron ejecutados correctamente.

De igual manera se comprobó que el botón y el potenciómetro envían un mensaje cada vez que el botón es presionado o cada que el potenciómetro cambia su valor. En el apéndice A se puede apreciar una tabla con los comandos probados y los detalles de resultados.

1.1.5 Conclusiones

Este proyecto cumplió con el objetivo de aclarar cuál es la ventaja de usar un sistema operativo en tiempo real ya que separar la solución en diferentes tareas que pudieran ejecutarse simultáneamente lo hizo no solamente más fácil sino posible, siempre y cuando se comprendiera bien el flujo de las tareas y sus prioridades al momento de ejecutarse. Por otro lado, a pesar de las limitaciones de un sistema embebido contra las capacidades de una PC, el hecho de lograr conectar a Internet el sistema es un aprendizaje muy valioso ya que la tendencia es que todo debe estar conectado y ya que vivimos en un mundo globalizado esta tendencia de IoT marca una nueva generación de aplicaciones en las que todas deberán estar forzosamente conectadas a Internet.

1.2. Ecualizador Digital

1.2.1 Introducción

Este proyecto consiste en la implementación de un ecualizador digital que maneje 3 bandas de paso: pasa-bajas, pasa-bandas y pasa-altas, de ganancia variable, desarrollado usado la tarjeta de TMDSDSK6713 de Spectrum Digital. El objetivo de este proyecto es aplicar el conocimiento en el diseño de filtros digitales mediante la técnica de ventaneo para filtros de respuesta al impulso finita (FIR) y su implementación en un sistema embebido para una aplicación de audio.

1.2.2 Antecedentes

Los Ecualizadores son mecanismos que nos permiten modificar la capacidad de nivel de las frecuencias que estén en curso cambiando solo su amplitud. La diferencia fundamental entre los analógicos y los digitales es que mientras el comportamiento del sistema analógico radica en la topología y componentes utilizados en él, en el sistema digital el comportamiento queda definido exclusivamente por un algoritmo computacional.

Sus principales ventajas son la flexibilidad que poseen, puesto que pueden modificarse y actualizarse sin mayores inconvenientes, así como también son fácilmente reproducibles, puesto que no dependen de la tolerancia de los componentes discretos y/o integrados utilizados en circuitos analógicos, y son sumamente confiables gracias a que no envejecen ni sufren desvíos por causas ambientales.

1.2.3 Solución Desarrollada

La primera aproximación fue el desarrollo de los filtros en Matlab mediante la herramienta 'Filter Design & Analysis Tool' en los cuales se consideró una frecuencia de muestreo de 24KHz. El diseño de cada una de las bandas de paso del filtro fue hecho por separado. El filtro pasa bajas fue diseñado con una frecuencia de corte de 300Hz donde debe atenuar a 6dB, y consta de 61 coeficientes para que la fase sea lineal, sin embargo, la simulación mostró que la frecuencia de corte es hasta los 432 Hz debido a que el filtro requería más coeficientes para que la frecuencia de

corte fuese a los 300Hz. El filtro pasa banda tiene una frecuencia de paso a partir de los 300Hz hasta los 4.5 KHz, de modo que a partir de los ~300Hz la magnitud es mayor al 50%, volviendo a atenuarse a un 50% en 4.5KHz. El diseño de este filtro también tiene una fase lineal, lo cual se logra con un mínimo de 61 coeficientes. El filtro pasa-altas permite pasar frecuencias a partir de 4.5KHz, su implementación requirió solamente 20 coeficientes para poder tener una fase lineal debido a que la frecuencia de corte estaba más cerca a la frecuencia de muestreo, sin embargo, al tener 20 coeficientes, la pendiente de este filtro no era complementaria con el filtro pasa-bandas, lo cual resultaba en una ganancia inconstante del ecualizador por lo que se decidió utilizar 61 coeficientes al igual que los filtros anteriores.

En el apéndice B se pueden observar gráficas que muestran el comportamiento de los filtros con su explicación detallada.

El ecualizador consiste en la suma proporcional de cada una de las salidas de los filtros. El control para atenuar o amplificar una banda es la ganancia que se le da a la salida de cada filtro antes de la suma del filtro. El límite de una atenuación para cierta banda significa la multiplicación por cero, y la amplificación límite de la banda es la multiplicación por 2. En el apéndice B se muestra un diagrama a bloques del ecualizador.

Ya que las operaciones de punto flotante requieren una gran cantidad de procesamiento se decidió utilizar representación de punto fijo Q15 convirtiendo y normalizando los números en punto flotante. Con la finalidad de conocer cuál sería la variación en la respuesta del ecualizador al cambiar los coeficientes de los filtros de punto flotante a punto fijo se diseñó un script de Matlab el cual compara ambas aproximaciones y da a conocer el error generado por este cambio, sin embargo, no se detectó ninguna variación.

En el apéndice B se muestra el diagrama de flujo del ecualizador desde que se recibe la muestra hasta que se envía la muestra transformada, así como los coeficientes de los tres filtros usados para ganancias variables de 0 a 2 en pasos de 0.1.

1.2.4 Análisis de Resultados

Una vez implementado el algoritmo, el tiempo entre cada muestra no era suficiente para procesarla por lo que se cambió la representación de datos en punto flotante a punto fijo, y se usó una optimización del compilador generando código que cumpliera con el requerimiento anterior.

La frecuencia de muestreado es de 24K muestras por segundo, lo cual nos permite muestrear señales con un máximo de 12KHz sin generar aliasing.

Un área a explorar es la implementación de los filtros directamente en ensamblador para aumentar su procesamiento contra el aumento del nivel de optimización del compilador. Otro punto que no se consideró en la implementación del filtro pasa-altas fue el de recurrir a un filtro pasa-bandas con frecuencia de corte superior igual a 12KHz. Esto con la intención de evitar la introducción de señales que pudieran generar aliasing dada la restricción de 24KHz de muestreo.

Finalmente, otra área de desarrollo para un trabajo futuro es la del manejo de un bloque de muestras usando el módulo de DMA en vez de un procesamiento muestra por muestra, como se tiene actualmente.

1.2.5 Conclusiones

Con este proyecto se demostró que los conceptos vistos en la clase de procesamiento digital de señales tienen una aplicación en la construcción de filtros digitales. El objetivo de la construcción de un ecualizador se cumplió satisfactoriamente haciendo uso de diferentes conceptos de DSP que fueron implementados y transformados a una implementación en un sistema embebido. El documento final detalló los pasos necesarios para la generación de filtros digitales basados en requerimientos propuestos por el ecualizador en cuestión.

1.3. Implementación de Sistema para Calcular Aceleración de Vehículo Aéreo no Tripulado (Punto Fijo vs Punto Flotante)

1.3.1 Introducción

Este proyecto parte del supuesto de estar ayudando en el desarrollo de un vehículo aéreo no tripulado calculando su aceleración dada por la siguiente formula:

$$x = sensor\_1 - 3 \times \sin\left(\frac{sensor\_2}{16}\right) + \frac{(sensor\_1 + sensor\_2)}{2}$$

Donde el *sensor_1* provee una salida de 19 bits de datos con un rango de $30u_1$ a $120u_1$ la cual es enviada por SPI al microcontrolador y el sensor_2 es un dato de 8 bits que se obtiene de pasar por un ADC el voltaje diferencial que es directamente proporcional a la aceleración obtenida por un sensor teniendo para el valor de 0 = $0u_2$ y para 255 = $15.3u_2$

La unidad de la variable x es u_1 y se considera que $1u_2 = 0.23502u_1$.

1.3.2 Antecedentes

El desarrollo de aplicaciones en sistemas embebidos presenta grandes retos, ya que por un lado se enfocan en medir el tiempo, detectar y responder a eventos externos, lidiar con restricciones físicas, hacer el procesamiento de los datos y dar respuesta en tiempo real mientras que por el otro lado la capacidad de procesamiento y memoria son muy limitados comparados con los sistemas de propósito general. Es por esto que además de las optimizaciones del compilador, los desarrolladores pueden usar diferentes técnicas para reducir la carga de procesamiento de datos optimizando el manejo de recursos, principalmente memoria.

1.3.3 Solución Desarrollada

La solución desarrollada se implementó en 8 etapas las cuales se mencionan a continuación.

1) Dadas las especificaciones del problema se obtuvieron las ecuaciones para *sensor_1* y *sensor_2* las cuales se pueden apreciar en el apéndice C. De este modo nuestro sistema quedaba conformado por 4 ecuaciones, la primera para la variable *x*, la segunda para el *sensor_1*, la tercera para el *sensor_2* y la cuarta con la relación entre las unidades u_1 y u_2.
2) La ecuación para el *sensor_2* se dejó en términos de *u1* dando como resultado la ecuación 5 y se utilizó esta ecuación junto con la ecuación del *sensor_1* para sustituirlas en la ecuación original resultando una sexta ecuación.
3) Se utilizó Matlab para evaluar la ecuación 6 y encontrar una ecuación equivalente reducida dando lugar a la ecuación 7.
4) Se evaluaron diferentes formas de ejecutar la función seno ya que es la que utilizaría más recursos. Las primeras dos opciones que se evaluaron fueron el uso de tablas o de librerías, sin embargo, al evaluar la opción de la tabla se encontró que el rango de valores posibles para el argumento del Seno estaba muy limitado y los resultados de la función Seno dentro de este rango eran prácticamente el mismo valor del argumento por lo que se decidió dejar el argumento como variable y eliminar la función Seno. Este proceso para reducirse la ecuación a su mínima forma se realizó en 3 partes donde se obtuvieron las ecuaciones 8, 9 y 10.
5) Se modificó la ecuación 10 para poder utilizar representación Q7 de punto fijo, para esto las constantes usadas en la ecuación tuvieron que ser modificadas y/o truncadas.
6) Se evaluó el error en el peor de los casos utilizando punto fijo que fue 0.0017649903059862, el cual es menor a 0.5 que era lo que se requería en las especificaciones.
7) Se modificó la ecuación para utilizar punto flotante, en cuya representación se utilizó una mantisa de 16 bits por cuestiones de performance haciendo la transformación a números de punto flotante en tiempo de ejecución.
8) Se evaluó el error en la representación de punto flotante que fue de 0.012162990306**.** Este error pudo ser mucho menor con una mantisa de 23 bits sin embargo por cuestiones de performance se mantuvo de 16 bits ya que al final de cuentas el error es mucho menor a indicado en la especificación.

El desarrollo detallado de la implementación se encuentra en el apéndice C.

1.3.4 Análisis de Resultados

La ecuación se resolvió usando punto fijo y/o punto flotante y fue implementada en el microcontrolador MSP430G2231 enviando el dato de 19 bits desde otro microcontrolador simulando el *sensor_1* y usando un potenciómetro para simular el *sensor_2*.

Los resultados obtenidos no fueron los óptimos ya que hubo algunas variaciones contra los resultados teóricos, sin embargo, quedaron dentro del rango establecido por las especificaciones con suficiente margen de error.

También se hizo una evaluación de que sucedería si se utilizara un valor de 10 bits para el ADC y se observó que en esta implementación específicamente el error aumenta de forma considerable ya que se decidió eliminar la operación de Seno pues el resultado y el argumento eran prácticamente iguales, sin embargo, al utilizar 10 bits esta diferencia aumenta, aunque no llega a ser preocupante puesto que el error aún sigue estando lejos del 0.5 indicado en las especificaciones.

1.3.5 Conclusiones

Este proyecto me hizo reflexionar sobre qué tan complejo es para el compilador usar punto flotante y la indiferencia con la que lo usamos al programar. Aprendí que es mucho más fácil utilizar punto fijo, aunque el punto flotante promete mucha más precisión y que cuando se utilizan dispositivos con memoria muy limitada es imprescindible el uso de técnicas para optimizar la memoria como el uso de punto fijo.

Finalmente aprendí que la mayoría de las veces no nos preocupamos tanto por la memoria ya que siempre que programamos consideramos una memoria ilimitada en el caso de una PC o con memoria de sobra en el caso de un sistema embebido y esto podría ser la cusa de comportamientos inesperados en aplicaciones tanto embebidas como en sistemas operativos complejos.

2. Conclusiones

En este documento se logran concentrar los productos más significativos del aprendizaje adquirido en la Maestría en Diseño Electrónico donde el área de concentración fue sistemas embebidos.

Los conocimientos adquiridos con estos tres proyectos específicamente son de gran valor ya que no se quedan sólo como teoría, sino que se pueden aplicar perfectamente en proyectos reales cuando se trata de desarrollo de aplicaciones embebidas. Estos conocimientos proporcionan suficientes bases para introducirse en ramas específicas del mundo de los sistemas embebidos como los son los sistemas operativos en tiempo real o el procesamiento digital de señales, así como también proporcionan conocimiento más profundo en el desarrollo de aplicaciones embebidas en general y tener en cuenta factores que antes se ignoraban como es el caso de la optimización de recursos a la hora de desarrollar.

Durante el curso de la maestría además de desarrollar aplicaciones para consolidar los conocimientos también se trabajó de tal manera que se buscara seguir un ciclo de desarrollo real, así como fomentar el trabajo colaborativo.

Finalmente, después de haber desarrollado estos y otros proyectos a lo largo de la maestría me siento con muchas más herramientas para trabajar en el desarrollo de aplicación embebidas, con un panorama más amplio, y con capacidad de aprender cosas nuevas con mayor facilidad.

Apéndices

A. CONTROL REMOTO SOBRE ETHERNET A TRAVÉS DE TWITTER

Diagrama a Bloques (Diagrama Funcional)

La siguiente ilustración muestra la arquitectura de solución propuesta en este trabajo. El usuario puede conectarse desde cualquier dispositivo a su cuenta de Twitter. El bloque compuesto por el K60 con MQX embebido se encarga de leer e interpretar los mensajes enviados por el usuario. Utilizando comandos predefinidos el usuario puede controlar una bombilla de 60 Watts y un motor de corriente directa de 12V interfazados con el microcontrolador a través de una etapa de potencia. También se cuenta con un potenciómetro y un botón para obtener entradas asíncronas las cuales son notificadas al usuario a través de la misma cuenta de Twitter.

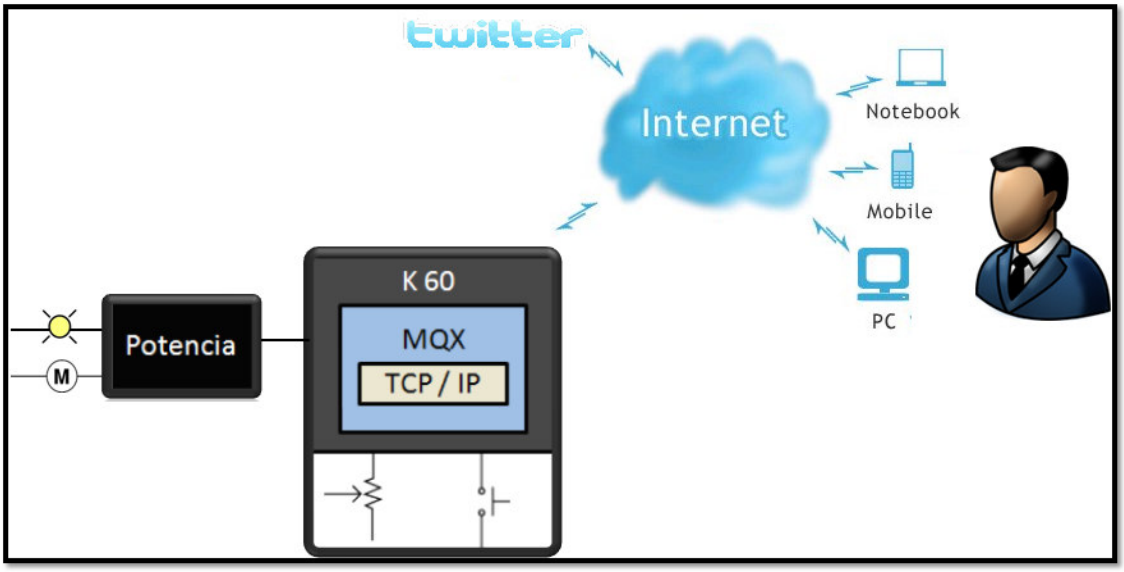


Diagrama de bloques

Arquitectura de Software

Tarea Main

Inicialmente, los pines de propósito general de entrada y salida son configurados para controlar 4 diodos luminosos en la tarjeta TWR-K60N512-KIT. El driver de LWGPIO es utilizado

para este fin. Estos diodos luminosos se encenderán y apagarán según los comandos leídos desde la cuenta de Twitter. Después de que los pines de propósito general de entrada y salida son inicializados la tarea Main llama a la función InitializeNetworking() para iniciar servicio de TCP/IP provisto por la librería RTCS de MQX. Este puede ser configurado para fijar una dirección de IP estática o utilizar DHCP para obtener una dirección IP dinámica.

Después de concluirse las inicializaciones, la tarea principal entra en un ciclo infinito donde se monitorean el convertidor analógico a digital y el botón, a su vez se lleva la cuenta del tiempo para iniciar periódicamente la tarea HTTP_Client_Get.

La siguiente figura muestra el flujo de la tarea principal.

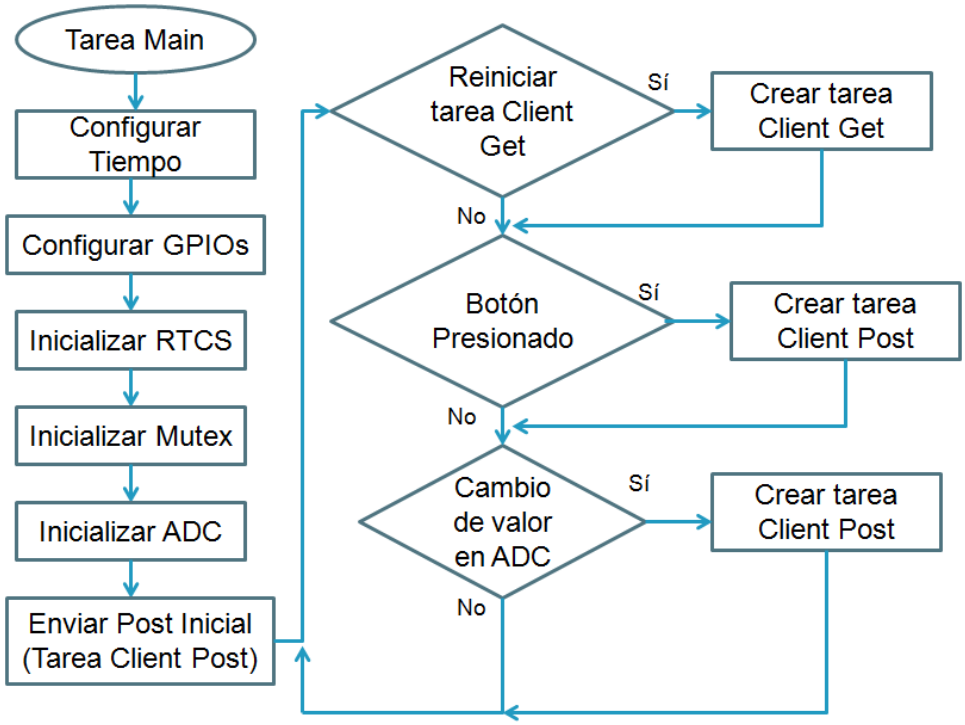


Diagrama de flujo Tarea Main

Cliente HTTP Get

El propósito de esta tarea es llevar a cabo la comunicación con el servidor de Twitter y leer los comandos a ser ejecutados. Para obtener los comandos de entrada la función httpclient() los lee desde el ultimo tweet publicado por una cuenta especifica de Twitter. El comando es después analizado y ejecutado según la implementación.

El tiempo para reiniciar la tarea es controlado por un valor que puede ser configurado por el usuario.

La siguiente figura muestra el diagrama de flujo de HTTP Client Get Task.

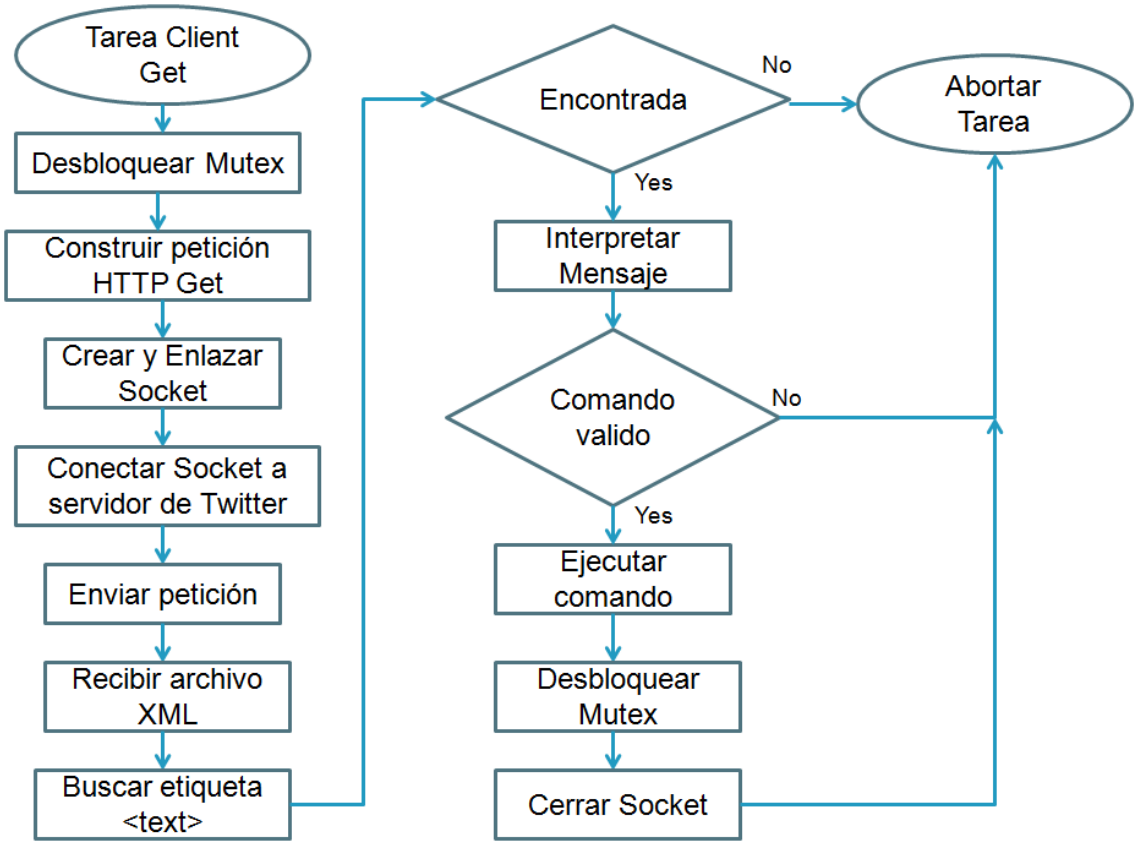


Diagrama de flujo Tarea Client Get

La tarea httpclient crea un socket TCP que utiliza el puerto 80 que a su vez pertenece al protocolo HTTP. Este socket se conecta a al servidor de Twitter. Para obtener el último tweet es necesario construir una petición de HTTP. Esta petición debe realizarse como lo indica la siguiente figura.

```
GET /users/fsl_mcu.xml HTTP/1.0\r\n
Host: twitter.com\r\n
User-Agent: HTMLGET 1.0\r\n\r\n
```

Petición HTTP Get

El protocolo HTTP se describe en el RFC 2616 (Request for Comments por sus siglas en inglés). A continuación, se describen cada una de las cadenas.

GET /users/FSL_K60.xml HTTP/1.0 – Se utiliza el método GET para obtener cualquier información que contenga el URL (Uniform Resource Locator por sus siglas en inglés). En este caso el URL devuelve el archivo FSL_K60.xml que corresponde a la cuenta de Twitter creada para este proyecto. También se envía la versión del protocolo HTTP. Este caso se usa la versión HTTP/1.0.

Host: twitter.com – La segunda cadena especifica el Host al que se hace la petición. Debe representar el URL del servidor al que se hace la solicitud.

User-Agent: HTMLGET 1.0 – La cadena User-Agent contiene la información del usuario. Se puede componer de múltiples palabras simbólicas a las que llamamos símbolo que el agente puede usar para identificarse. En este caso se utiliza un símbolo para identificar a la aplicación de control y monitoreo. Dicha cadena es "HTMLGET" versión "1.0".

El servicio de Twitter proporciona un archivo XML, el cual es analizado para obtener información de la cuenta de un usuario. Los tweets que se envían desde la cuenta de Twitter "FSL_K60" son utilizados para representar los comandos de entrada del microcontrolador. El archivo FSL_K60.xml consiste en una serie de datos que corresponden a la cuenta de Twitter "FSL_K60". La etiqueta <text> contiene el último tweet que el usuario envió al servicio de Twitter. Esta etiqueta puede encontrarse en el archivo XML.

Para obtener el archivo XML es necesario establecer una conexión TCP a través del puerto 80 con servidor HTTP de Twitter. La petición HTTP descrita en la figura 1 es enviada al servidor y el resultado es el archivo XML. El contenido del archivo es analizado, una vez que la etiqueta <text> es encontrada el análisis concluye y el contenido de esta etiqueta es interpretado y ejecutado en caso de que se identifique un comando valido.

El comando tiene una estructura especifica; este debe comenzar con un punto (.) que indica que es el inicio del comando. La primera palabra debe de ser "spark", esto le dice al analizador que el comando debe de ser evaluado. Otro punto (.) es necesario para separar el siguiente indicador. La aplicación implementa cuatro objetos; "led", "samplerate", "motor" y "light". El objeto "led" recibe dos parámetros. El primero indica el número de diodo luminoso a controlar, los valores pueden ir de 1 a 4. El segundo parámetro indica el nuevo estado del diodo luminoso,

solo dos valores son válidos; “on” y “off”. Estos valores le indican al microcontrolador si debe encender o apagar un diodo luminoso en específico.

La implementación del objeto “samplerate” se utiliza para modificar el tiempo de espera para que la aplicación vaya y cheque si hay nuevos mensajes en la cuenta de Twitter. Por default la aplicación inicia con 5 minutos como periodo.

También se implementaron los objetos “motor” y “light”. Para implementar estos comandos se editó el driver de LWGPIO para mandar tres señales de salida. Una que controle una luz de 60 Watts y las otras dos se utilizan para controlar el sentido del giro del motor de corriente directa los cuales están conectados con su respectiva etapa de potencia.

El objeto “motor” recibe como parámetros los comandos “left”, “right” y “off”. El objeto light solamente recibe “on” y “off”.

En la siguiente tabla se pueden observar algunos ejemplos:

.spark.led.N.on.	Enciende el diodo luminoso N del TWR-K60N512-kit, donde N es u numero entero del 1 al 4.
.spark.samplerate.N.	Cambia el periodo para reiniciar la tare HTTP Get al número de minutos indicado por N.
.spark.light.ACCION.	Envía la instrucción para encender o apagar una bombilla de 60W que es controlada a través de un pin de propósito general de entrada y salida, donde ACCION puede ser ‘on’ u ‘off’.
.spark.motor.DIRECCION.	Envía las instrucciones para girar el motor de corriente directa de 12V a la derecha o a la izquierda, donde DIRECCION puede ser ‘right’ o ‘left’.

Ejemplos de comandos

Como se muestra en la tabla los puntos (.) son los símbolos que se utilizan para identificar cada parte del comando. Es necesario que le comando inicie y termine con punto (.).

La lista de comandos puede ser tan extensa como la aplicación lo demande. Con esta implementación solo se pretende mostrar la funcionalidad de los comandos.

HTTP Client Post Task

El propósito de esta tarea es establecer la comunicación con el servidor de Twitter con el fin de enviar un mensaje desde el microcontrolador. Para llevar a cabo esta tarea se utiliza un tercer elemento que es el servicio de Super Tweet.

El servicio de SuperTweet.net ofrece una alternativa para los conjuntos de soluciones de TCP/IP que no cuentan con el mecanismo de Capa de Sockets Seguros (SSL por sus siglas en inglés). La autenticación básica de HTTP se utiliza para iniciar sesión en el servidor SuperTweet.net. Para empezar, es necesario firmarse en una sesión en Twitter para autorizar a la aplicación API MyAuth Proxy SuperTweet.net. el acceso a la cuenta de Twitter. Después elegir una nueva contraseña (diferente a la contraseña real para twitter) que las aplicaciones pueden utilizar con la API de http://api.supertweet.net. Después que estos pasos han sido completados es posible abrir comunicación entre MQX y Supertweet.net para que sea posible postear un tweet en la cuenta de Twitter.

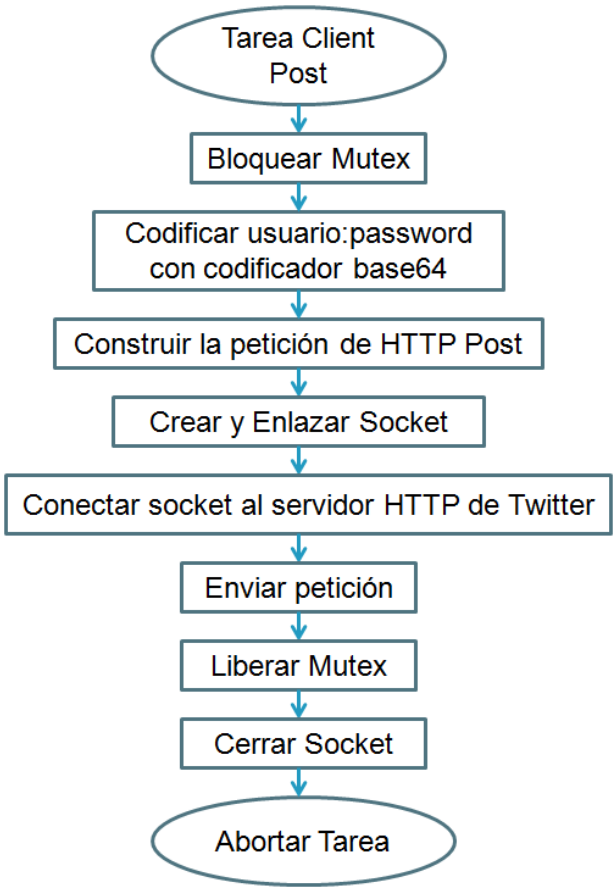


Diagram de flujo Tarea Client Post

La tarea httpclient crea un socket TCP que utiliza el puerto 80 el cual a su vez pertenece al protocolo HTTP. Este socket se conecta al servidor de SuperTweet. La API de SuperTweet.net proporciona un método que toma ventaja de la tecnología de autenticación OAuth de Twitter, sin el costo ni la complejidad de OAuth, en una simple aplicación en Twitter. Para publicar un tweet en una cuenta de Twitter, es necesario construir una petición HTTP y enviarla al servidor de SuperTweet. La solicitud debe de ser construida como se muestra en la siguiente figura.

```
POST /1/statuses/update.xml HTTP/1.1\r\n
Authorization: Basic ZnNsX21jdTp4c3cyMXFheg==\r\n"
User-Agent: HTMLPOST 1.0\r\n"
Host: api.supertweet.net\r\n"
Accept: */*\r\n"
Content-length: 35\r\n"
Content-Type: application/x-www-form-urlencoded\r\n\r\n"
status=Tweet from TWR-K60N512 board
```

Ilustración 1. Peticion HTTP Post

A continuación, se describen cada uno de los encabezados de la petición de HTTP mostrada en la figura 4 de acuerdo al RFC 2616.

POST /1/statuses/update.xml HTTP/1.1 – El método POST está diseñado para enviar un bloque de datos de manera uniforme. Un ejemplo puede ser el enviar de una forma llenada en algún sitio web para hacer un pago, o darse de alta para obtener algún servicio. En esta aplicación el método POST utiliza la API “/1/statuses/update.xml”. Esta API se utiliza para enviar un tweet desde la cuenta indicada.

Authorization: Basic ZnNsX21jdTp4c3cyMXFheg== -- El acceso por autenticación básica es un método de seguridad utilizado por los clientes HTTP en los cuales los usuarios proveen un nombre de usuario y una contraseña para poder hacer alguna petición desde alguna cuenta. El nombre de usuario y la contraseña se concatenan utilizando dos puntos (:). La cadena construida es codificada usando el algoritmo Base 64. En esta aplicación el usuario 'FSL_K60' y la contraseña 'xsw21qaz', son concatenados formando la cadena 'FSL_K60:xsw21qaz' la cual es codificada usando el algoritmo Base64. La cadena resultante de este proceso es 'ZnNsX21jdTp4c3cyMXFheg==’.

Usuario	Contraseña	Antes de Base64	Después de Base64
FSL_K60	xsw21qaz	FSL_K60:xsw21qaz	ZnNsX21jdTp4c3cyMXFheg==

Decodificación Base64

User-Agent: HTMLPOST 1.0 – El campo User-Agent contiene información acerca del agente de usuario (cliente) que origina la petición. Este campo puede contener varios símbolos y comentarios que identifican el agente. En este caso solamente un símbolo es utilizado; nombre "HTMLPOST" y versión "1.0".

Host: api.supertweet.net – Este campo especifica la Internet del recurso que se solicita. Este campo debe representar el nombre del servidor origen que está dado por el URL original. Esto le permite al servidor distinguir entre URLs internos y ambiguos de un servidor de múltiples conexiones.

Accept: */* -- El campo Accept se utiliza para especificar ciertos "media types" que son aceptados en la respuesta.

Content-length: 35 – El encabezado Content-Length indica el tamaño del mensaje que será enviado y está dado en número decimal.

Content-Type: application/x-www-form-urlencoded -- Indica el "media type" del contenido enviado.

status=Tweet from TWR-K60N512 board – Esta cadena es enviada a la API "/1/statuses/update.xml". Este texto representa un comando que indica el mensaje que será posteado en la cuenta de Twitter.

Una vez enviada la petición, Twitter regresa un archivo XML que incluye el código de la API ejecutada y el mensaje enviado. De esta manera se envía un mensaje a la cuenta de Twitter cada vez que el potenciómetro tiene una variación o cada vez que se presiona un botón de la tarjeta que en una aplicación real podríamos interpretar como notificaciones de lecturas de sensores.

Pruebas y Resultados

Para probar la solución implementada, se configuro para leer el último tweet cada 30 segundos y se enviaron todos los comandos reconocidos por el sistema repetidas veces. Después de varias sesiones de prueba y depurar algunos errores finalmente todos los comandos fueron ejecutados correctamente.

Lo comandos enviados fueron los siguientes:

Control de diodos luminosos	**Control de muestreo**	**Control de bombilla 60W**	**Control de motor de CD**
.spark.led.1.on.	.spark.samplerate.1.	.spark.light.on.	.spark.motor.left.
.spark.led.2.on.		.spark.light.off.	.spark.motor.right.
.spark.led.3.on.			
.spark.led.4.on.			

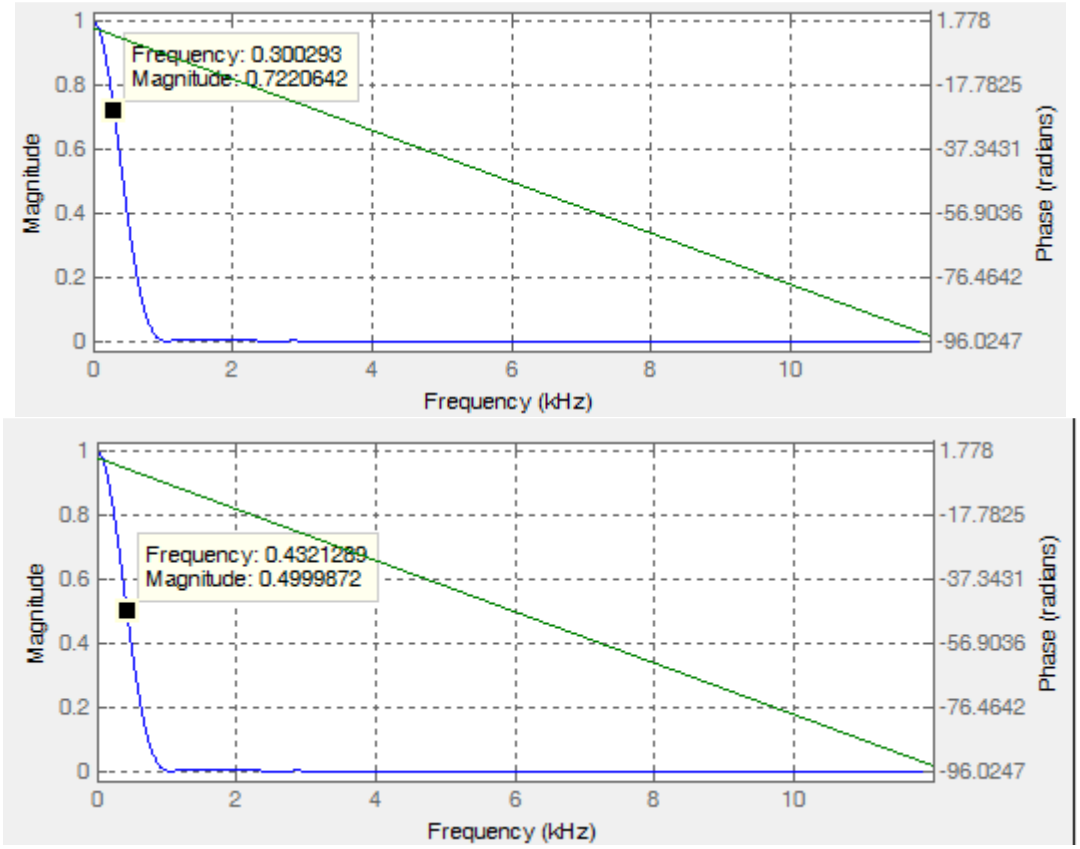
Pruebas

También se comprobó que el botón y el potenciómetro envían un tweet cada que el botón es presionado o cada que el potenciómetro cambia su valor.

B. ECUALIZADOR DIGITAL

Filtro Pasa-bajas

La siguiente figura muestra la representación del filtro pasa-bajas señalando la frecuencia de corte en magnitud y comportamiento de la fase.

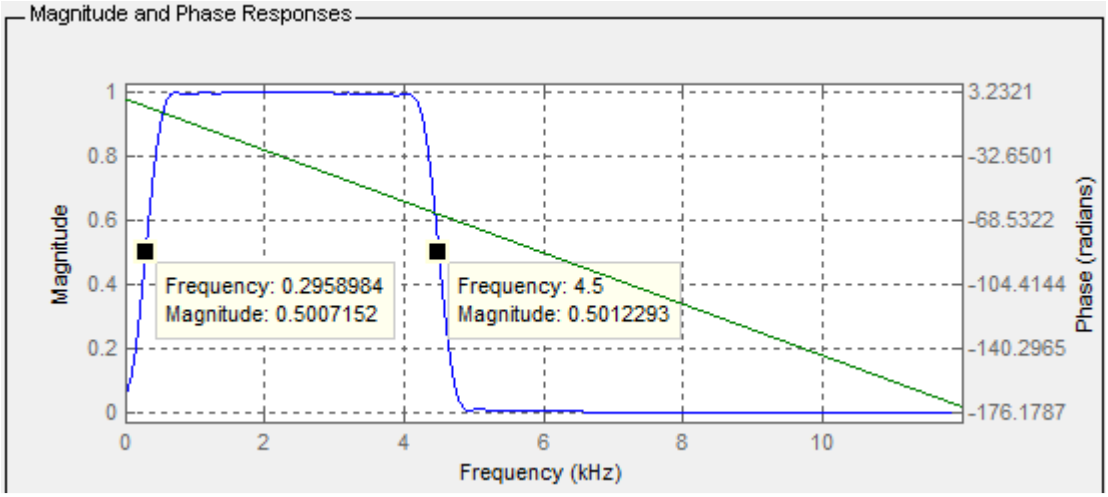


Filtro pasa-bajas

En la figura se puede apreciar que la fase de la salida del filtro es lineal, esto se logró diseñando un filtro de 61 coeficientes (con menos coeficientes la fase ya no era lineal). Se diseñó el filtro con una frecuencia de corte de 300Hz (donde debería atenuar a 6dB), sin embargo, se puede apreciar que la frecuencia de corte es hasta los 432 Hz, esto debido a que el filtro requería más coeficientes para que la frecuencia de corte fuese a los 300Hz, pero se decidió dejarla a los 432Hz para no aumentar el número de coeficientes y por lo tanto no aumentar el procesamiento en el DSP.

Filtro pasa-bandas

La figura 2 muestra la representación del filtro pasa-bandas señalando las frecuencias de corte y la magnitud.

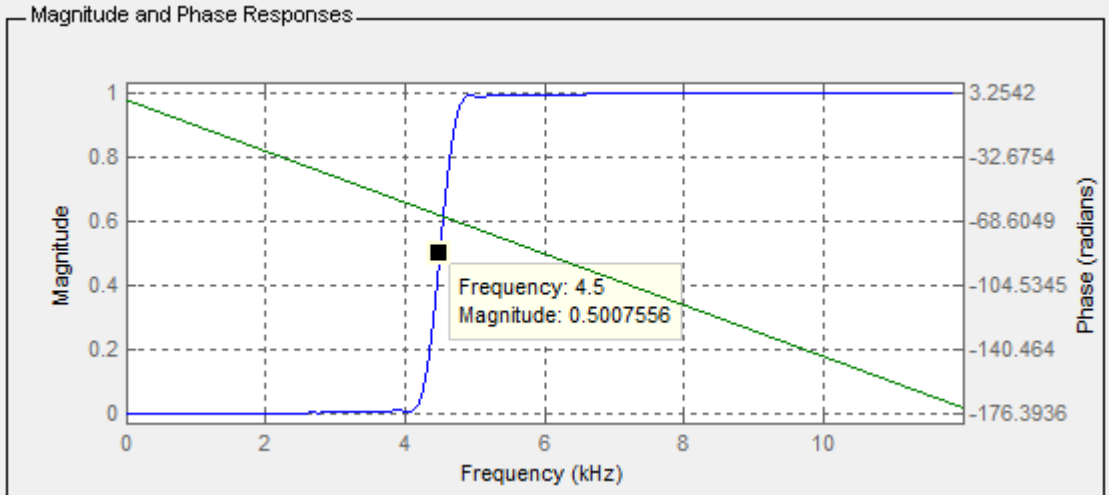


Filtro pasa-banda

Ahora se puede observar que el filtro tiene una frecuencia de paso a partir de los 300Hz hasta los 4.5 KHz. La gráfica muestra que a partir de los ~300Hz la magnitud es mayor al 50%, volviendo a atenuarse a un 50% en 4.5KHz. Nuevamente se verifica que el diseño de este filtro también tiene una fase lineal, lo cual nos garantiza que la magnitud no se verá afectada. El mínimo de coeficientes necesarios para tener una fase lineal es de 61.

Filtro pasa-altas

La figura 3 muestra la representación del filtro pasa-altas señalando las frecuencias de corte en magnitud y su fase.



Filtro pasa-altas

En esta ocasión se puede observar que el siguiente filtro deja pasar frecuencias a partir de 4.5KHz. Una vez más, la respuesta en fase también es lineal con lo cual garantizamos que la magnitud no se verá afectada en ciertas áreas. La implementación de este filtro requirió solamente 20 coeficientes para poder tener una fase lineal debido a que la frecuencia de corte estaba más cerca a la frecuencia de muestreo. Sin embargo, al tener 20 coeficientes, la pendiente de este filtro no era complementaria con el filtro pasa-bandas, con lo cual la ganancia del ecualizador no era constante, sino que tenía una irregularidad en la zona de transición entre filtros. De esta manera se decidió generar el número de coeficientes del filtro con el diseño anterior, es decir, 61 coeficientes.

Ecualizador

Diagrama de bloques

El siguiente diagrama a bloques muestra la interacción de los tres filtros en la integración del ecualizador.

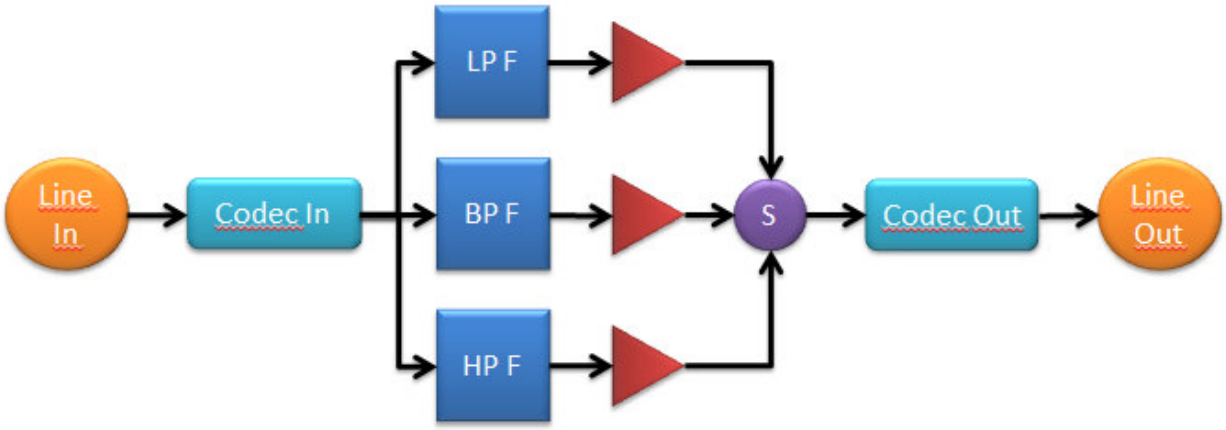
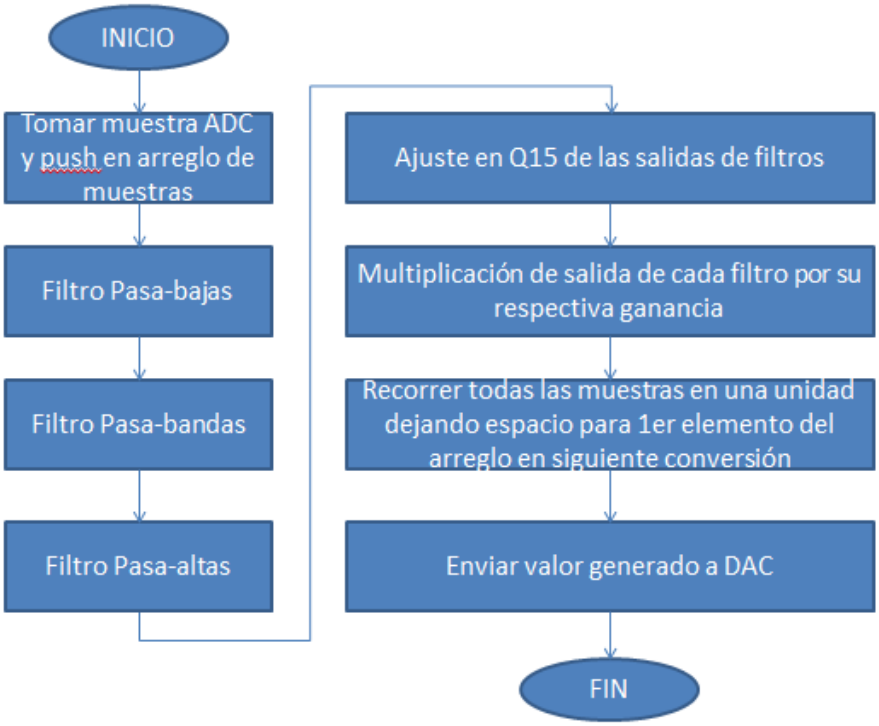


Diagrama de bloques de ecualizador

El filtro se diseñó para que la magnitud en cada frecuencia de corte únicamente aporte el 50%. Esto resulta en una respuesta del 100% por cada empalme de frecuencia de corte entre filtros contiguos. La excepción es el comportamiento del filtro pasa-bandas con las frecuencias mas bajas. La respuesta del filtro pasa bandas en la frecuencia 0 no empieza en amplitud 0, sino en un valor mayor debido a su pendiente provocada por el número de coeficientes usados. Esto resulta que durante la suma del ecualizador en esta banda de transición (banda pasa-bajas + pasa-altas) no resulte en una respuesta constante. Para compensar esta situación, la ganancia del pasa-bajas se ajusta a 66.5% para cada una de las muestras (este valor fue obtenido mediante prueba y error).

Diagrama de flujo

A continuación, se muestra el diagrama de flujo del ecualizador desde que se recibe la muestra hasta que se envía la muestra transformada.



Algoritmo

C. IMPLEMENTACIÓN DE SISTEMA PARA CALCULAR ACELERACIÓN DE VEHÍCULO AÉREO NO TRIPULADO (PUNTO FIJO VS PUNTO FLOTANTE)

Paso 1: ecuaciones propuestas

Partiendo de la ecuación **Eq. 1**:

$$x = sensor\_1 - 3 * \sin(\frac{sensor\_2}{16}) + (\frac{sensor\_1 + sensor\_2}{2}) \quad \textbf{[Eq. 1]}$$

Para el *sensor_1* (SPI), la siguiente ecuación se derive de las especificaciones:

$$sensor\_1 = (\frac{SPI}{2^{19} - 1} * 90) + 30 \text{ utilizando } u_1 \text{ como unidades } \textbf{[Eq. 2]}$$

Para el sensor_2 (ADC), se obtuvo la siguiente ecuación:

$$sensor\_2 = \frac{ADC}{2^8 - 1} * 15.3 \text{ utilizando como unidades } u_2 \textbf{ [Eq. 3]}$$

Del problema original tenemos que la siguiente igualdad es válida:

$1u2 = 0.23502\ u_1$ **[Eq. 4]**

Paso 2: resolviendo las ecuaciones

Eq 1 se muestra en términos de u_1. Después combinando **Eq 3** y **Eq 4** se obtiene la siguiente ecuación:

$$sensor\_2 = \frac{ADC}{2^8 - 1} * 15.3 * 0.23502 \text{ utilizando } u_1 \text{ como unidades [Eq 5]}$$

Ahora reemplazando **Eq. 2** y **Eq. 5** en **Eq. 1,** se obtiene la siguiente ecuación:

$$x = ((\frac{SPI}{2^{19} - 1} * 90) + 30) - 3\sin(\frac{\frac{ADC}{2^8 - 1} * 15.3 * 0.23502}{16}) + (\frac{((\frac{SPI}{2^{19} - 1} * 90) + 30) + (\frac{ADC}{2^8 - 1} * 15.3 * 0.23502)}{2})$$

[Eq. 6]

Eq. 6 está en términos de u_1.

Paso 3: simplificando la ecuación

Se buscó utilizar el inverso de los divisores para facilitar la ejecución de la aplicación, y haciendo algebra la ecuación se redujo a la siguiente forma.

$x = A * SPI + 45 - 3\sin(\frac{B * ADC}{16}) + \frac{B * ADC}{2}$ utilizando u_1 como unidades **[Eq. 7]**

Donde:

A = 0.000257492556557763
B = 0.0141012

Step 4: evaluando la función Seno

1) Quitando la función Seno

El valor del ADC debe estar entre 0 y 2^8-1 (255). Se evaluó el argumento del Seno $\frac{B * ADC}{16}$ y se identificó que en el rango en que opera el ADC el argumento del Sena es casi el mismo que el resultado de la función Seno. En el peor de los caos ocurre cuando el valor del ADC es 255 y el error para este caso es de **0.001887038** de modo que se considera no significativo.

2) Usando tablas

Al usar tablas solamente se consideran 2 sifras significativas para el argumento del Seno y el efecto de esta solución sería que todos los valores intermedios tendrían el mismo resultado.

En otras palabras:

Sin(0.070) = 0.070447599
Sin(0.079) = 0.070447599

Se evaluaron las dos aproximaciones y se concluyo que eliminar la función Seno introduce un error menor que utilizar tablas. Por otro lado la tabla utiliza mayor espacio en memoria y toma mas tiempo para hacer los cálculos. Así que considerando las limitantes del

microcontrolador (4KB of Flash and 128 bytes of RAM) se consideró que la mejor opción es eliminar la función Seno.

Después de eliminar la función Seno de la ecuación numero 7 la ecuación resultante queda como sigue:

$x = A * SPI + 45 - 3(\frac{B * ADC}{16}) + \frac{B * ADC}{2}$ utilizando u_1 como unidades **[Eq. 8]**

Y simplificando:

$x = A * SPI + 45 + \frac{5}{16}(B * ADC)$ utilizando u_1 como unidades **[Eq. 9]**

Finalmente resolviendo las contastes:

$x = (A * SPI) + 45 + (C * ADC)$ utilizando u_1 como unidades **[Eq. 10]**

Donde:

A = 0.000257492556557763

C = 0.004406625

Step 5: utilizando punto fijo

Ya que el tipo de dato que se utiliza es de 32 bits se convirtieron las constants a punto fijo y en hexadecimal:

A = 0xEA30285DADC3

C = 0x433D61

Aqui se muestra que la constant A no cabe en un dato de 32 bits por lo que es necesario truncarla.

Utilizando Q7 (7 dígitos para la representación fraccionaria)las constantes quedan de la siguiente forma:

$A_{embedded} = 2575x10^{-7}$
$C_{embedded} = 44066x10^{-7}$

El error introducido en la constant A es **0.000000007443442237**, y el error en C es **0.000000025**.

De este modo la ecuación 10 con valores de punto fijo quería de la siguiente forma:

$x = (A * SPI) + 450000000 + (C * ADC)$ utilizando u_1 como unidades **[Eq. 11]**

Donde:

A = 2575
C = 44066

Paso 6: cálculo de error usando punto fijo

Para el peor de los casos que es **255** (ADC, 2^8-1) y **524287** (SPI, 2^{19}-1), utilizando la ecuacón 7 (**Eq. 7)** da un resultado exacto de **181.1293504903060000** u1

Utilizando la ecuación 11 (**Eq. 11)**, el resultado es **1811275855**, que transformado de formato Q7 a decimal decimal ies **181.1275855**

La diferencia entre los resultados en el peor de los casos es **0.0017649903059862**, lo cual es menor a **0.5** de error.

Paso 7: utilizando punto flotante

Para la implementación en punto flotante se utilize una mantisa de 16 bits ya que si se utilazarn 23 bits durante la multi´plicacion se obtendria una mantisa de 46 bits, de tal modo que por cuestiones de performance se decidio utilizar una mantisa de 16 bits.

La fórmula para la solución en punto flotante es la siguiente:

$x = (A * float(SPI)) + float(45) + (C * float(ADC))$ utilizando u_1 como unidades **[Eq. 11]**

Donde:

//A=0.000257492556557763
A.sign = 0
A.exp = 116
A.mant = 0x0E00

//C=0.00440662
C.sign = 0
C.exp = 120
C.mant = 0x20CA

La operación float() indica que el cálculo para transformar a punto flotante esta hecho en tiempo de ejecución.

Paso 8: Cálculo del error usando punto flotante

Así como en la implementación de punto fijo, el error en el peor de los casos es cuando los valores del ADC y de SPI son los más altos. En este caso el valor real debería ser **181.1293504903060000** u1 y el resultado obtenido en punto flotante es:

Sign: 0 Exp: 134 Mantissa: 0x6A3C

Al convertirlo a decimal nos da un valor de **181.1171875** por lo que la diferencia con el valor real es de **0.012162990306** lo cual también es menor a **0.5** pero mayor al resultado en punto fijo y como conclusión, entre estas dos implementaciones la de punto fijo fue mejor.

Paso 9: Que pasaría si el ADC usara datos de 10 bits

Por la forma en que se implement la solución, en la **Eq.7**, la parte donde se evalua el Seno $\sin(\frac{B * ADC}{16})$, tiene un mayor error que con una variable de 8 bits, pero aparentemente no es relevante ya que el error final sigue estando lejor del 0.5.

ADC	Argumento	Seno	Differencia
1010	0.89013825	0.777158756	0.112979494
1011	0.891019575	0.777713076	0.113306499
1012	0.8919009	0.778266792	0.113634108
1013	0.892782225	0.778819904	0.113962321
1014	0.89366355	0.77937241	0.11429114
1015	0.894544875	0.779924311	0.114620564
1016	0.8954262	0.780475606	0.114950594
1017	0.896307525	0.781026295	0.11528123
1018	0.89718885	0.781576378	0.115612472
1019	0.898070175	0.782125853	0.115944322
1020	0.8989515	0.782674721	0.116276779
1021	0.899832825	0.783222981	0.116609844
1022	0.90071415	0.783770633	0.116943517
1023	0.901595475	0.784317675	0.1172778