Proyecto final “Sensing Assistant”

Dan Arturo,Zhu Alfredo, Cruz Miguel, Acosta Fernando.
A01650672, A01651980, A01656527, A01333721
Instituto Tecnológico y de Estudios Superiores de de Monterrey

Abstract
El desarrollo de nuevas tecnologías en el área de la domótica ha traído consigo el surgimiento de nuevas aplicaciones de los sistemas embebidos. En el presente documento se detalla el desarrollo de un sistema inteligente de monitoreo de parámetros del hogar como lo son la temperatura y presencia. Esto a través de la implementación de un sistema en tiempo real (FreeRTOS), el uso de sensores analógicos (ADC), USART e interfaz gráfica en LCD que junto al uso de audio (TWI - TPA 6130) permiten el monitoreo correcto del hogar.

I. Nomenclatura

Interfaz de dos hilos, *TWI*.
Conversor Analógico a Digital, *ADC*.
Pantalla de Cristal Líquido, *LCD*.
Sistema Operativo en Tiempo Real, *RTOS*.
Acceso Directo a Memoria, *DMA*
Transmisor-Receptor Síncrono/Asíncrono Universal, *USART*.
Digital Segura, *SD*.

II. Introducción

Los sistemas embebidos se han convertido en una pieza vital, además de las computadoras de uso general. Un sistema embebido es una pequeña unidad de procesamiento que forma parte de un sistema, dispositivo o máquina más grande. Incluye tanto hardware como software y su propósito es controlar el dispositivo y permitir que un usuario interactúe con él. Suelen tener un número limitado y específicos de tareas que realizan. Es una combinación de programa y equipo, que puede ser programable o no, los sistemas embebidos suelen ser no programables y la misma se suele realizar por el fabricante, a menudo es posible actualizar el software del sistema integrado, por ejemplo los relojes medidores de actividad física; aunque existen sistemas embebidos programables con fines educativos y de prueba. Algunos ejemplos de sistemas embebidos son:

- Sistemas de calefacción.
- Sistemas de control de motor en vehículos.
- Electrodomésticos.
- Relojes digitales.
- Calculadoras electrónicas.
- Sistemas GPS
- Medidores de actividad física.

Dependiendo del propósito del sistema embebido, se utilizan generalmente dispositivos de entrada como sensores, botones e interruptores para producir una salida. [1]

Algunas características de los sistemas embebidos son:

- Ejecución continua.
- Alta accesibilidad y confiabilidad.
- Creación en torno a un trabajo continuo.
- Cumplen con una actividad simple y específica.
- Asociados con periféricos para interconectar información y generar gadgets.
- Ofrecen una alta calidad y resistencia.
- Interfaz de usuario poco compleja.
- Memoria restringida.

Los sistemas embebidos tienen muchas ventajas, como su fácil creación para aplicaciones de cierto nivel complejo, pueden tener interconexiones de periféricos, poseen una velocidad constante y alta,

confiabilidad elevada, versátil por su pequeño tamaño, debido a su número limitado de funciones propicia que sean más baratos de diseñar y construir, requieren menos energía siendo algunos capaces de trabajar con baterías y se pueden construir con procesadores más baratos y menos potentes.

Así como estos sistemas tienen ventajas, también tienen desventajas, por ejemplo, si uno de estos sistemas es creado sin que se pueda modificar su programación, implica que no podrá sufrir alguna alteración o mejora de *Software*. Al ser creados con una función específica, si se desea adaptar para algún otro propósito, no será posible.

Los sistemas embebidos que son capaces de ser programados, pueden implementar un kernel de sistema operativo en tiempo real, conocido como RTOS. Existen una gran variedad de versiones de RTOS, una de las más comunes que se puede utilizar es el FreeRTOS gracias a que es de fuente abierta.

FreeRTOS es un programador de subprocesos. Sin embargo, no tiene memoria virtual, sistemas de archivos ni modelo de seguridad. Este sistema operativo en tiempo real ofrece bastantes ventajas, la primera de ellas es su simplicidad, ya que no se necesita más que un archivo *.c* para ponerlo a correr en el microcontrolador. Si se necesita escribir o modificar el código del kernel para resolver un problema, no es difícil de entender este código. El pequeño tamaño de FreeRTOS va en conjunto con su simplicidad, debido a que en microprocesadores la memoria RAM y su almacenamiento flash son limitados, FreeRTOS permite ejecutar acciones complejas con un pequeño tamaño de programa ahorrando algunos bytes de memoria. [2]

También, FreeRTOS es “omnipresente”, es decir, se ejecuta prácticamente en cualquier microcontrolador pequeño y moderno, esto es bueno, ya que cuando se lleva un producto al mercado, le brinda cierto grado de aislamiento de hardware. “Free” significa que es completamente de código abierto, debido a que Amazon adquirió FreeRTOS en 2017 y se ofrece gratuitamente.

Finalmente, una ventaja es su operatividad en tiempo real, lo que significa que el diseñador del sistema puede garantizar que el procesador atenderá los eventos críticos en el tiempo dentro de los tiempos de respuesta óptimos o necesitados. A pesar de tener esta gran ventaja, los microprocesadores se han vuelto mucho más potentes, por lo que la cantidad de aplicaciones que necesitan una programación estricta en tiempo van disminuyendo.

III. Desarrollo

Se comenzó por analizar y generar diferentes ideas que representaran un reto y que formasen parte de las necesidades actuales del mercado, en cuanto utilidad y tecnología. Es por ello que se decidió desarrollar un prototipo de sistema embebido para aplicaciones de domótica u hogares inteligentes, usando como base la tarjeta de desarrollo EVK1105, la cual nos permitió incluir elementos relevantes para la aplicación como FreeRTOS, TWI (Audio - TPA6130+ABDAC), USART, ADC y LCD.

Figura 1: Tarjeta de desarrollo de Atmel, EVK1105.

El proyecto requiere de la adquisición de 3 sensores analógicos. Sin embargo, la tarjeta de desarrollo de ATMEL EVK1105 no cuenta con los pines de canales suficientes de ADC expuestos para poder utilizarlos. Se pudo identificar que los puntos de prueba TP36(GPIO 23) y TP51(GPIO 21) corresponden a los canales 2 y 0 de ADC. Considerando esto, se propuso utilizar un multiplexor analógico 74HC4067 de 16 entradas con 4 pines de selección. Dado que solo se requieren al menos 2 pines para poder multiplexar 3 canales, se aterrizan a tierra los dos pines de los bits más significativos a tierra y se conectan los otros dos pines de selección a los GPIOs 62 y 63. Las entradas de los 3 sensores analógicos se conectan a las entradas del multiplexor analógico. Se debe de tomar en consideración la correcta colocación de los puentes jumper que se encuentran arriba de la pantalla para que el audio pueda salir de manera correcta. Para este proyecto, se utilizó el amplificador TP6130 y el ABDAC*(Audio Bit Stream Digital to Analog Converter)* integrado en la tarjeta EVK1105. Las bocinas utilizadas en este proyecto e conectan en la entrada tipo Jack de audio.

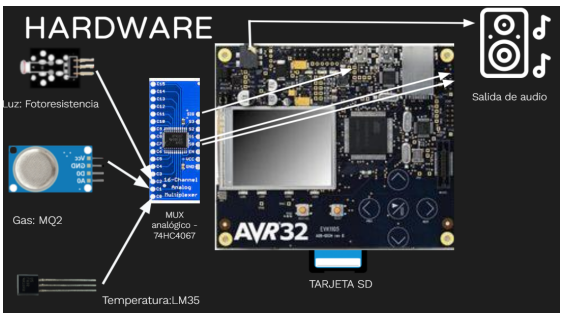

Figura 2:Diagrama de conexión de hardware.

Una vez definido el proyecto, su alcance y los elementos a utilizar se pasó a la etapa de diseño del prototipo, por lo que se bosquejaron las tareas a realizar dentro del sistema RTOS el cual permitiría un sistema mucho más eficiente y con tiempos de respuesta acordes a las necesidades del proyecto. En este caso, se optó por implementar siete tareas las cuales se dividieron en ADC junto con USART para captura y transmisión de datos, TWI para producir audio (3 tareas) , LCD para mostrar información relevante de las mediciones y SD para almacenar la información (2 tareas). Si bien, la implementación de estas tareas representó una mejora significativa en comparación con un sistema que no usase RTOS, se debió considerar un sistema de “semáforos” para los casos en los que se usasen el mismo recurso, ya que permite evitar que las tareas usen el mismo recurso en el mismo tiempo. Este es el caso del TWI - Audio que tiene tres tareas y la SD que tiene dos tareas. El semáforo funciona como una manera de esperar a que termine una tarea para poder realizar la otra y que los recursos no se usen de forma simultánea, evitando así errores al momento de realizar las tareas.

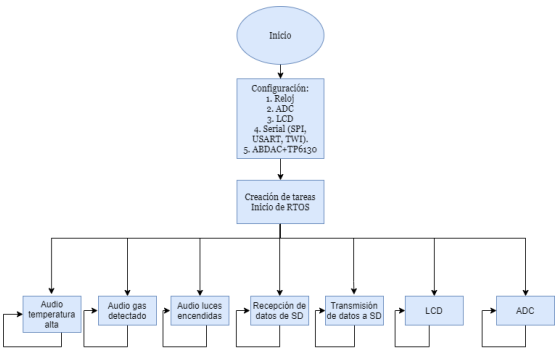

Figura 3:Diagrama tareas e inicialización FREE RTOS

Para este proyecto que se basa en FreeRTOS, es importante considerar los estados de transición para entender el funcionamiento del código. Se encuentran 4 estados principales en los que se puede encontrar una tarea. Cuando la CPU del microcontrolador ejecuta una tarea, este se encuentra en estado de ejecución y mientras pasa esto, las demás tareas se encuentran en los otros estados. Todas las tareas que están en el estado listo para ejecución entrarán al estado de ejecución cuando les toque. Hay un estado de espera de las tareas el cual espera a que un evento ocurra, las tareas saldrán de dicho estado después de que un evento específico pueda transicionar al estado de espera, ejecución o incluso puede estar inactivo. El estado inactivo es una tarea terminada y no participa en la transición al menos que se vuelva a crear.

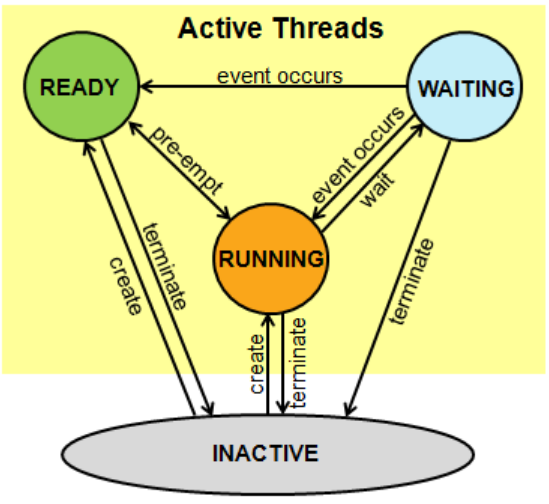

Figura 4:Diagrama RTOS

Se implementa un contador para seleccionar los canales del multiplexor y de esta forma seleccionar el sensor que se desea leer. Se utilizaron las colas de RTOS para transmitir la información a diferentes tareas, en este caso se transmite la información para que la tarea de la LCD y de la tarea de la SD la puedan utilizar. Aunque podría ser posible separar el ADC de USART para que trabajen en dos tareas diferentes, la transmisión de USART se implementó en la misma tarea que el ADC con el fin de tener una sencillez para depurar el código y detectar fuentes de errores inmediatas debido al FreeRTOS. Una vez creadas las tareas y comprobar que funcionasen de forma correcta, enviando una respuesta a la computadora por medio de USART, se procedió a probar el audio del sistema.

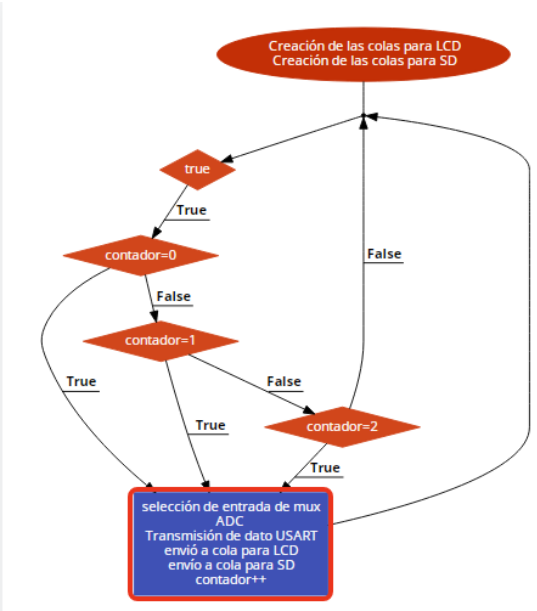

Figura 5:Diagrama ADC + USART

Se diseñó la interfaz gráfica a mostrar en la LCD. Se buscó que esta mostrase el cambio de temperatura registrado por un sensor LM35 y modificase el color de una barra lateral de acuerdo a la temperatura de la habitación. Así mismo, se incluyó el símbolo de una casa, la cual cambiaría de

color de acuerdo a las mediciones de un sensor de gas MQ2, lo cual mostraría si hay presencia de gas dentro de la habitación, para de esa manera reducir los riesgos que esto presentarían. La pantalla LCD, al ser un dispositivo de salida, debe de proporcionar una respuesta visible al usuario que le permita conocer el estado de todos los sensores utilizados, es por esto que se crea una tarea específica para recibir los datos de los sensores y compararse con un valor específico que actuará como un umbral, en el que dependiendo del sensor y del valor numérico que se le asignó al umbral, se otorgará una respuesta visible en la pantalla, por ejemplo para el sensor de temperatura, el cual medirá esta característica, se compara con el valor establecido con el usuario y en caso de ser mayor al umbral, se dibujará una figura que represente que se sobrepasó el límite de la temperatura y se reanudará la tarea de audio de temperatura, la cuál hará sonar una alerta audible de este fenómeno; en caso contrario se dibuja una figura la cual muestra que los valores son correctos y suspende la tarea de audio de temperatura, ya que aún no se requiere usar. Así mismo, la misma tarea de LCD permite reanudar o suspender las tareas de audio dependiendo de las condiciones de los datos que recibe a partir de la tarea de ADC.

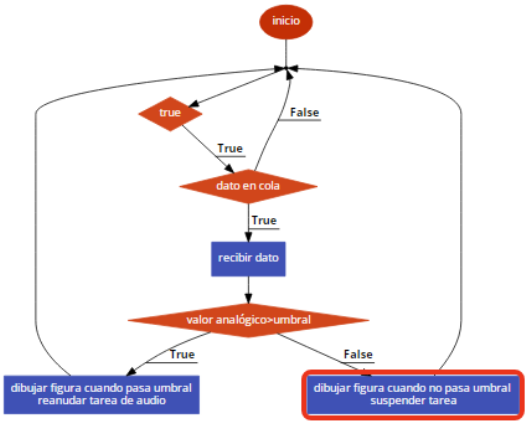

Figura 6: Diagrama LCD

La reproducción de audio resultó muy retadora al implementarlo con RTOS, ya que se debía conocer bien su funcionamiento, así como el de DMA para poder modificar parámetros dentro de las funciones de audio dadas por el compilador. Uno de los grandes problemas presentados por el uso de audio fue averiguar cómo cambiar la reproducción de los audios y cómo compilarlo en la plataforma de programación de ATMEL 7. Lo que se hizo principalmente fue tomar archivos de los headers de twi y del tp6130 de las versiones de ATMEL 7 y sustituirlas por los mismos headers que generan en las versiones anteriores. El audio que se quiso reproducir fue generado en Audacity con 11025Hz de frecuencia y con codificación de 8 bits PCM. La baja resolución de audio permite que podamos guardar más archivos de audio en la EVK1105 a costo de la calidad de audio. Para pasar el audio a un header y poder leerlo, se utilizó un código de MATLAB para leer el audio y pasarlo a un formato de 8 bits hexadecimal. En el caso de la implementación de TWI - AUDIO, se ocuparon semáforos para realizar cada tarea de forma que ninguna ocurriera de forma simultánea, ya que se usan los mismos recursos. La idea detrás del concepto de semáforo es el uso de un estilo de bandera que nos permita identificar cuando una está siendo realizada y soltar esa tarea o liberarla una vez que termine, dando paso así a la siguiente tarea. En este diagrama, se observa cómo una vez comprobado que se puede realizar la tarea se procede a cargar los archivos de audio, se reproducen y se cargan los datos de DMA, para finalmente soltar o liberar la tarea, dando pauta así a que la próxima tarea de audio puede ser realizada,

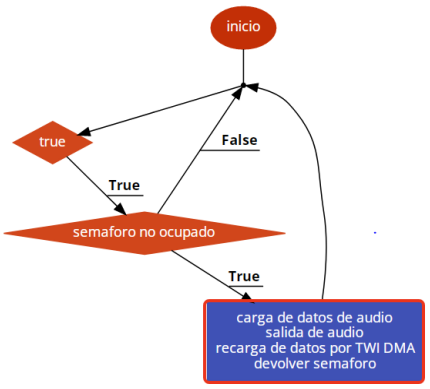

Figura 7: Diagrama TWI - AUDIO

Para este proyecto, la SD se utilizó para almacenar los datos medidos de los sensores analógicos. Hay dos tareas de SD, una que es para escribir y otra para leer en los sectores. Debido a que estas dos tareas ocupan el mismo recurso, se implementó una condición de semáforo para que las dos tareas no entrarán en conflicto con el mismo recurso. Para leer y escribir en la SD, solo se utilizó las funciones de la tarjeta SD. Para escribir en cualquier sector, lo que se hace es abrir el sector, escribir el dato y luego cerrar el sector. Para leer los datos de la SD, está acondicionando por una condición de la presión de la tecla central de la EVK1105 para poder mostrar los datos en la pantalla. Además, antes de leer el sector, se escribe información del mismo dato a recibir debido al buffer circular.

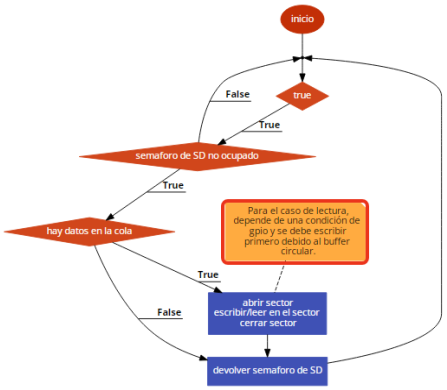

Figura 8: Diagrama SD

IV. Resultados

Los resultados obtenidos al finalizar el desarrollo y pruebas del prototipo del proyecto nos arrojaron que las respuestas del sistema a cambios de temperatura, luz y presencia de gas en la habitación de prueba ocurrieron en los tiempos óptimos para el prototipo, ya que gracias al FREERTOS, algunos procesos ocurrían de forma optimizada lo que permitió disminuir de forma considerable los tiempos de respuesta.

Del mismo modo, las respuestas del sistema mostradas a través de la LCD y el sistema de audio (TP6130+ABDAC) sucedieron de forma correcta de acuerdo a los parámetros establecidos dentro del programa. Por un lado, para el caso en el que la temperatura fuese fría la barra lateral de nuestra interfaz de usuario se volvió azul, mientras que en caso contrario esta se tornaba roja a la vez que mostraba el valor numérico de la temperatura a un lado de la barra. Así mismo, se detonaba una respuesta de audio que expresaba que la temperatura se encontraba caliente según las mediciones del sensor. Por otro lado, si el sensor de presencia de gas MQ2 notaba la presencia de gas, este mandaba una respuesta de audio al usuario a la vez que mostraba que la habitación está ocupada por gas, a través de la interfaz gráfica de la LCD. Así mismo, existió una respuesta de audio y cambio en el símbolo de foco en la LCD al registrarse un cambio notable en la luz, lo cual significa un cambio de día a noche.

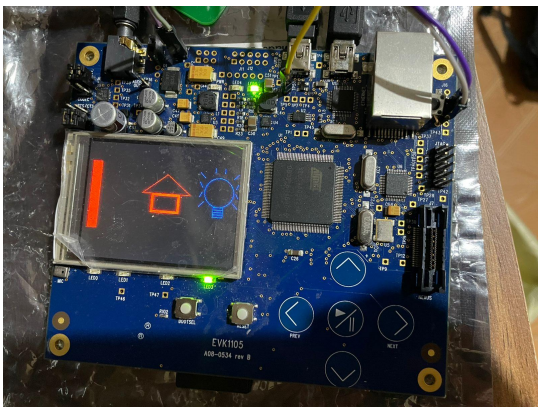

Figura 9: Interfaz gráfica LCD

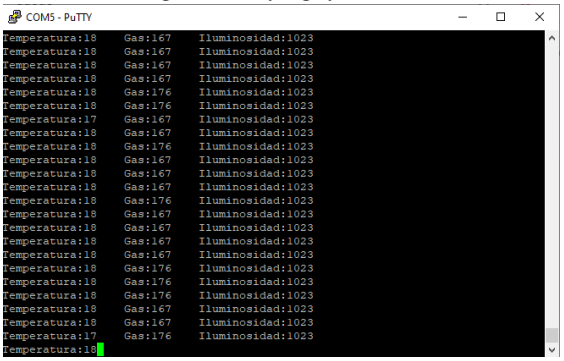

Figura 10: Datos recibidos por USART

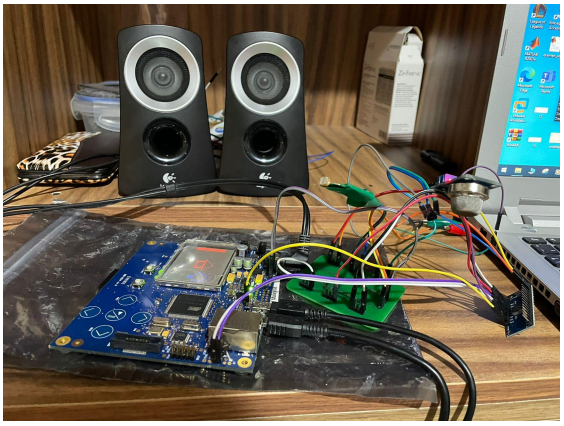

Figura 11: Prototipo vista 1

Figura 12: Prototipo vista 2

V. Conclusiones

Alfredo Zhu: Se logró desarrollar un sistema de monitoreo de parámetros del hogar basado en la tarjeta de desarrollo EVK1105 y FREERTOS, el cual nos permitió medir y dar una respuesta auditiva y visual a cambios en distintos parámetros de una habitación. Considero que desarrollar este proyecto representó un buen reto para el equipo, ya que se debió investigar e implementar elementos los cuales sólo se vieron de forma teórica, como lo son FREERTOS y TWI. Así mismo, logramos sobrellevar los problemas que surgieron durante del desarrollo del mismo e implementar elementos que históricamente no habían podido ser utilizadas en la versión actual de Atmel Studio. En mi opinión, desarrollar proyectos que te permitan un mejor entendimiento de la materia; así como, conocer e implementar tecnologías que se utilicen en el mundo actual es de suma importancia para cualquier ingeniero. Me siento satisfecho del trabajo realizado y la labor colaborativa de todo el equipo.

Miguel Cruz: Como se mencionó en el documento, los sistemas embebidos toman mayor fuerza día con día, y sus aplicaciones van creciendo cada vez más. Es difícil imaginar un mundo sin sistemas embebidos y nosotros como ingenieros que tienen fuertes conocimientos en el tema, debemos estar a la vanguardia del diseño e implementación de sistemas de este tipo. Proyectos de este tipo son completamente enriquecedores para nosotros, ya que nos acercan un poco a lo que es el diseño en grandes compañías dedicadas a eso. A pesar de que fue relativamente poco tiempo, considero que el equipo supo hacerle frente a los problemas relacionados con software como la mencionada incompatibilidad de audio y RTOS con versiones recientes de Atmel Studio/Microchip Studio, así como con problemas de lógica de la tarjeta de

desarrollo, el funcionamiento fue el adecuado y los conocimientos adquiridos a lo largo de todo el curso, se vieron reforzados gracias a esta actividad.

Arturo Dan: En conclusión, puedo decir que este proyecto representó un verdadero reto para todo el equipo, ya que pesé a que al inició creímos que existirían restricciones considerables como el hecho de que históricamente el audio solo funcionaba en ATMEL 5 y FREERTOS lo habíamos probado solo en ATMEL 7 o microchip studio. Logramos implementar el audio en ATMEL 7 e integrarlo con el sistema FREERTOS. Así mismo, considero que logramos disminuir considerablemente la curva de aprendizaje al utilizar recursos que no habíamos llevado a la práctica como audio y ADC, e inclusive un tema nuevo como lo fue FREERTOS. En mi opinión, este proyecto en particular me resultó muy útil para reafirmar conocimientos de la materia a la vez que me adentraba más en nuevos temas del mundo de los sistemas embebidos. Estoy satisfecho con el trabajo realizado y me siento agradecido de haber podido aprender mucho durante el desarrollo de este proyecto.

Fernando Acosta: Se desarrolló un sistema inteligente que corre sobre la tarjeta de desarrollo EVK 1105 para el monitoreo de las variables de: Presencia, gas y temperatura. Gracias a la integración de la pantalla LCD de esta tarjeta pudimos dibujar en esta 3 íconos que representan la retroalimentación hacia el usuario para poder comprender de manera gráfica cómo es que se están comportando las variables monitoreadas. Con relación a la implementación de audio, puedo asegurar que fue una labor ardua debido a que debimos mudar todo ese código de ATMEL 5 a la versión 7 de este software para lograr la integración con el FreeRTOS. Con relación a esto último, me percaté que tal cual fue visto en la parte teórica del curso en primer parcial, se utilizan semáforos para poder realizar las tareas en RTOS, en nuestro caso solo se puede correr una tarea a la vez por restricciones propias de la EVK, hubiera sido interesante quizá tener un segundo núcleo para incursionar más en la programación multitarea. Por último, es importante señalar que para la integración completa de nuestro proyecto, hicimos uso de todos los módulos que vimos en el segundo parcial. En términos generales puedo concluir que sin duda fue un reto de buen nivel lograr la integración de todos los bloques que conforman nuestro proyecto y me deja satisfecho poder ver nuestros conocimientos aterrizados en una implementación con la combinación de Hardware y Software.

VI. Referencias

[1] Devices, P. Embedded Systems. In *Mobile HCI*.

[2] Barry, R. (2008). FreeRTOS. *Internet, Oct*.