Universidad de Costa Rica

Escuela de Ingeniería Eléctrica

Laboratorio de programación en sistemas incrustados.

B22245 - Daniel Díaz Molina

B05568 - Jorge Rojas Vargas

Informe de Laboratorio 1:

Luz de noche accionada por voz

# Introducción

## Problema a resolver

Creación del prototipo de un sistema de luz activada por voz, este prototipo es capaz de demostrar la operación de las tres versiones del producto: La versión de 5 W, la de 10 W y la de 15 W. El estado inicial de encendido o apagado de las luces LED del sistema está definido por el nivel de luminosidad en la habitación donde se encuentra instalado el sistema.

La lámpara deberá encenderse de forma automática si el nivel de luminosidad está por debajo del límite día/noche y además exista en la habitación alguna fuente de ruido que sea de un nivel de volumen mayor al 10% del nivel de volumen de audio promedio de los últimos 5 segundos. Asimismo, será posible cambiar de estado usando únicamente un botón pulsador.

## Solución propuesta.

El prototipo de desarrolló haciendo uso de la placa TI MSP432 y el Educational BoosterPack MKII. Gracias a los requerimientos de diseño fue posible conocer que es necesario utilizar los LEDs RGB, el micrófono, el convertidor analógico/digital, el sensor de luz, los botones pulsadores de estos dispositivos y el módulo Timer32 con el fin de llevar a cabo la implementación del prototipo.

# Desarrollo

## Implementación

Se optó por dividir las diferentes funcionalidades del sistema para luego coordinar su funcionamiento mediante la creación de una máquina de estados. Para poder saber si había un aumento en el nivel de ruido promedio de los últimos 5 segundos, se utilizó un buffer circular que iba almacenando los promedios de cada 432 mediciones del micrófono. Si el promedio de los valores correspondiente al último segundo es 10% mayor a los promedios de los últimos 5.0112 segundos, una variable booleana indica que existe un valor alto de ruido. Como frecuencia de muestreo se decidió utilizar un valor de 8.620kHz, el cual está cerca de los 8kHz a los que se muestrea el servicio de telefonía convencional.

Para la utilización del sensor de luminosidad, se decidió utilizar la capa de abstracción de dicho sensor utilizado en los ejemplos de software de la TI MSP432. Esto facilitó su implementación e inicialización.

### Funcionamiento de la máquina de estados

Al correr la máquina de estados, inicialmente habrá un parpadeo de LED que indica que el sistema entró en operación, seguidamente se evalúa el valor actual de luz percibida por el sensor de luz (estado **RESET**). Dependiendo de este valor apaga (estado **LIGHT\_OFF**) o enciende el LED (estado **LIGHT\_ON**) durante el tiempo establecido. Si el botón se presiona durante LIGHT\_ON, el estado se pasa a LIGHT\_OFF y viceversa, además de que si durante LIGHT\_OFF hay una entrada de ruido en el micrófono y el nivel de luminosidad es bajo, se pasará de estado a LIGHT\_ON.

En la transición al estado **LIGHT\_ON** se activa un conteo del Timer32, para que la luz se apague automáticamente aproximadamente 10 segundos después.

Si al presionar el botón se da transición al estado **LIGHT\_OFF** se vacía el buffer circular, de modo que no se puede volver a encender la luz por el ruido hasta 5 segundos después (si el buffer no está lleno siempre se retorna un nivel de audio bajo). Esta fue una decisión de diseño que se tomó, pues muchas veces la acción de presionar el botón causa suficiente ruido como para que la luz se vuelva a encender inmediatamente. Dado que el botón parece ser un mecanismo de *override* para el comportamiento normal de esperar que la luz se apague, este comportamiento de volver a encenderse nos pareció indeseado.

## Problemas encontrados

1. Se tuvieron dificultades a la hora de crear la función encargada de encender el LED cuando se detectaran picos de volumen en el nivel de audio obtenido por el micrófono, pues inicialmente no se tenía una respuesta tan inmediata a como se deseaba. Esto fue corregido tras un proceso de corrección de errores y depuración.
2. Inicialmente hubo cierta dificultad por encontrar la manera más sencilla de hacerle debounce al botón pulsador. Existen varios enfoques pero el elegido fue el de cambiar el flanco que inicializa la interrupción del Puerto 1 dependiendo de si el botón está pulsado o sin pulsar. Además se desactivan las interrupciones por un periodo de 10 ms luego de presionar el botón, para esto se usó el Timer32.
3. Para obtener el nivel promedio del valor de volumen de audio, inicialmente se hacía uso de 2 buffers circulares: Uno para almacenar los valores de las mediciones del micrófono, y otro para almacenar valores promedio de las mediciones del primer buffer, por su implementación estos usan memoria dinámica. Esto consumía demasiada memoria, en la configuración se indicaba que el tamaño del Heap es 1 KB pero para usar los 2 buffers se requerían aproximadamente 1.3 KB por lo que se optó por sustituir el primer buffer con una suma consecutiva de valores muestreados, hasta llegar a 432 muestras. Estas muestras se promedian y se almacenan en el buffer de valores promedio. Así se obtuno la misma función utilizando menos memoria (400 B).

## Posibles mejoras

* Una de las cosas que notamos al probar el dispositivo es la suceptibilidad a ciertos parámetros que no están explícitamente específicados. Por ejemplo el nivel de luz, el umbral a partil del cual se considera día y noche. Un *feature* interesante sería la posibilidad de cambiar este umbral con el dispositivo funcionando, por ejemplo obteniendolo de la lectura de un potenciómetro.
* Entre los requerimientos del sistema se indicaba que una de las condiciones de encendido era que el nivel de audio del último segundo fuera 10% mayor que el de los útlimos 5. Sin embargo, en la práctica parece ser una diferencia muy pequeña, y casi cualquier ruido activa esta condición. De hecho, se tomó la libertad de cambiar el diseño y aumentarlo a un 20%, aún así consideramos que el micrófono sigue siendo bastante sensible al ruido. Habría que considerar donde se va a encontrar el micrófono y hacer más pruebas, para saber si sería conveniente incrementar aún más este porcentaje.

# Bibliografía

* *MSP-EXP432P401R SimpleLink™ MSP432P401R high-precision ADC LaunchPad™ Development Kit | TI.com*. (2018). *Ti.com*. Obtenido 21 abril 2018, from <http://www.ti.com/tool/MSP-EXP432P401R>
* *MSP432 Technical Reference Manual*. (2018). *Ti.com*. Obtenido 21 Abril 2018, from <http://www.ti.com/lit/ug/slau356h/slau356h.pdf>
* StackExchange. (2018). *Why is telephone audio sampled at 8 kHz?*. *Signal Processing Stack Exchange*. Retrieved 21 Abril 2018, from https://dsp.stackexchange.com/questions/22107/why-is-telephone-audio-sampled-at-8-khz