Lab 04, ADC y UART

Laboratorio Nivel 3

1. Introducción

Este laboratorio hace uso de una de las características relevantes de los microcontroladores: la Comunicación Serial. Esta funcionalidad permite transmitir y recibir distintos tipos de contenidos entre diferentes dispositivos, permitiendo un mayor nivel de control que usando simplemente GPIOs o señales analógicas. En esta oportunidad se trabajará con el protocolo de comunicación USART o *Universal Synchronous and Asynchronous Receiver and Transmitter* para realizar la comunicación con un computador.

Cabe indicar, que cuando se realiza la comunicación entre dos dispositivos, es común esperar que esta sea de carácter bidireccional, por lo que se espera que sea capaz de transmitir como de recibir. Para lo que existen diversos métodos, como lo es consultar constantemente el estado de variables (*polling-base communication*) o de esperar tiempos muertos hasta cumplir una condición (*busy-waiting*). Estos métodos resultan muy poco eficientes, pues mantienen completamente ocupado al microcontrolador en la transmisión o recepción, impidiendo poder utilizar ese tiempo en otras actividades.

Este tiempo en que no hay comunicación, puede ser utilizado para la adquisición de información. En laboratorios anteriores se trabajó con inputs mediante GPIOs usando o no interrupciones. Para este laboratorio además, se utilizará el Conversor Ánalogo Digital (ADC) incorporado en el microcontrolador para adquirir información y a partir de esta, ejecutar acciones.

Esta información, junto con la capacidad de comunicación, son insuficientes si solo se envían datos sin sentido. Por lo que entra en juego la capacidad de contar con un Protocolo de Comunicación que permita identificar un mensaje recibido o crear uno para ser enviado, con el objetivo de tomar las decisiones pertinentes acerca de algún proceso o de información a enviar.

Junto con lo anterior, se incorpora el concepto de interrupciones para la utilización de Timers y para la detección de entradas externas mediante puertos digitales.

El objetivo de este laboratorio, es lograr la implementación de un pequeño sistema, que a partir de instrucciones y datos leídos, sea capaz de controlar un programa alojado en un computador.

2. Actividad

Esta actividad consistirá en el creación de un pequeño mando de juego, con el que se controlará una interfaz gráfica entregada.

Los distintos Task, son subdivisiones de la actividad final, que consistirá en la implementación del sistema completo. Tienen como objetivo guiar el desarrollo y solo se revisará el último.

2.1. Booster Pack

A modo de definir un lenguaje común para este laboratorio, se utilizarán los siguientes elementos.

2.1.1. Analog Digital Converter

El Booster Pack, contiene distintos elementos que se encuentran conectados a entradas que leen datos digitales, para este laboratorio se hará uso de 4:

- Joystick Sentido Y
- Joystick Sentido X
- Acelerómetro Sentido X
- Acelerómetro Sentido Y

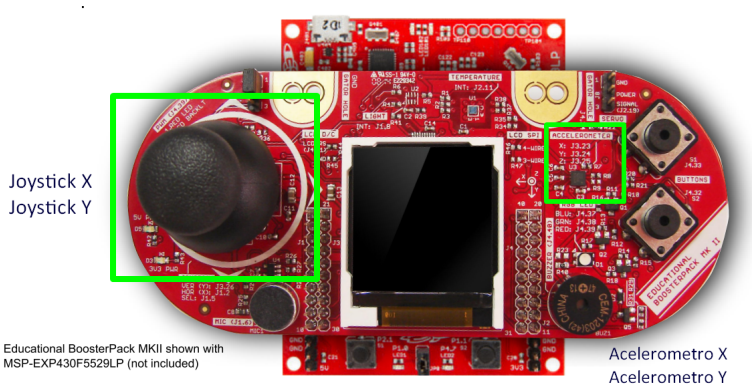

Figura 1: Componentes con conexión ADC

2.2. Buzzer

Recordando lo realizado en el laboratorio anterior, se utilizara el buzzer para la generación de efectos sonoros:

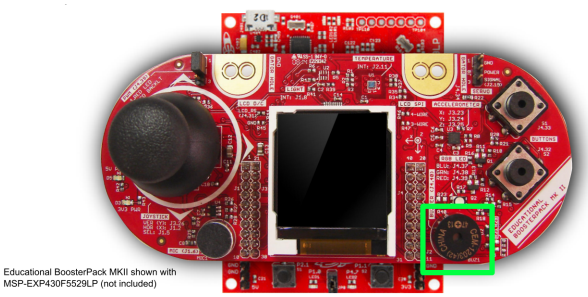

Figura 2: Buzzer

2.3. *Task 1*: Estudio de niveles del ADC mediante UART

Esta primera actividad, tiene por objetivo familiarizarse con el módulo ADC, de forma de conocer cómo adquirir la información y cómo enviarla por UART. Para ello, deberá implementar una biblioteca específica para UART. Esta biblioteca deberá ser instanciada en el programa principal y debe ser capaz de lo siguiente:

- Poder configurarse con distintos Baudrates, al menos:
  - 9600
  - 19200
  - 57600
  - 115200
- Poder configurar distintas paridades
- Enviar un caracter
- Enviar un string de largo arbitrario

Con esta función, deberá crear un código que tenga la capacidad de leer el valor de un canal ADC, crear un identificador para el mismo y enviar el valor medido en 4 dígitos. Un mensaje de ejemplo sería: `X0522`. Que indicaría que se leyó la variable X, el valor 0522. Utilice el siguiente configuración:

- Acc_x: `X0123`
- Acc_y: `Y0123`
- Joys_x: `A0123`
- Joys_y: `B0123`

Con esto, identifique y anote cuales son los límites reales del Joystick y del acelerómetro. Con estos valores deberá realizar una linealización entre 0-255 para posteriormente solo enviar estos valores por UART. (Ej: : `X123`)

2.3.1. Recomendaciones

Para llevar a cabo esta actividad, se le hará entrega de un código base que deberá rellenar. Para llevar a cabo esta labor, necesitará de un programa terminal como minicom, gtkterm, CuteCom o **RealTerm** (los ayudantes recomiendan este). Es de su responsabilidad la elección, instalación y familiarización con el programa que usted escoja.

2.4. *Task 2*: Recepción de instrucciones por UART

En esta actividad (que puede ser realizada en un código distinto al Task 1), se procederá a aumentar las capacidades de la comunicación UART. El objetivo será enviar instrucciones sencillas al MSP, esto será realizado de dos formas:

2.4.1. Recepción mediante `receive_char()`

Complete la función `receive_char()` y retorne el valor leído con la función `transmit_char()`. Esto es un test que permite chequear que lo que recibe el microcontrolador es exactamente lo que envía. En función de esto, cree un código sencillo que sea capaz de recibir los siguientes comandos:

- `R`: Enciende solo el led rojo del RGB
- `G`: Enciende solo el led Verde del RGB
- `B`: Enciende solo el led azul del RGB

Recomendación: El led RGB tiene una potencia bastante alta, por lo que puede producir dolores de cabeza el mirarlos fijamente y además de producir un consumo de corriente innecesario. Es por eso, que en este laboratorio se deberá bajar la intensidad mediante una PWM para evitar esto. Un duty cycle del 25 % debería ser suficiente. El no hacerlo, generará un descuento de 3 décimas en el laboratorio.

2.4.2. Recepción con interrupciones

Una vez comprendido el cómo funcionan la recepción de carácteres en el MSP, se mejorará el sistema para no tener que esperar (y de esta forma, detener el sistema) mientras realiza alguna otra acción. Para esto, deberá crear un código que dependiendo de la instrucción enviada, reproduzca un sonido con el Buzzer a una duración dada.

- `C`: Reproduce por aprox 250 ms nota `#define N_A4 440`
- `W`: Reproduce por aprox 250 ms nota `#define N_A5 880`
- `F`: Reproduce por aprox 250 ms nota `#define N_A6 1760`

Incorpore además a esta interrupción, las instrucciones realizadas en el punto anterior con los LEDs.

2.5. *Task 3*: Controlador de interfaz gráfica

En este Task, se deberá unir lo realizado por separado en los Task anteriores, para así construir un mando que sea capaz de controlar e interactuar con una interfaz entregada.

2.5.1. Interfaz

La interfaz fue construida en Python (testeado en la versión 3.6) y hace uso de las librerías pygame, serial, matplotlib y numpy. Lo siguiente es una captura de ella:

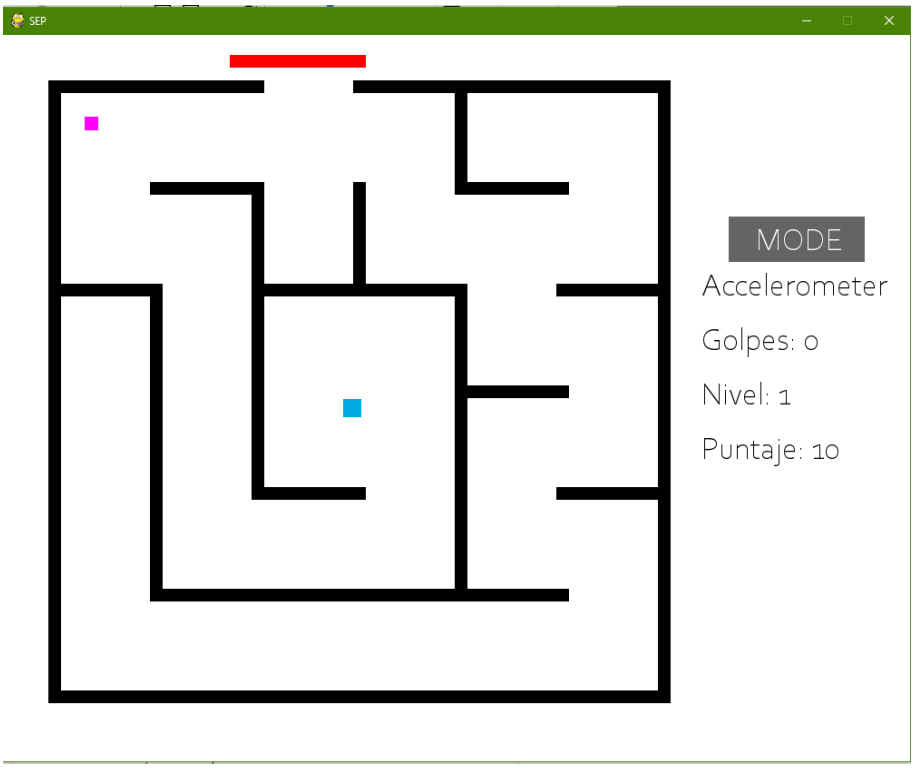

Figura 3: Laberinto

El objetivo es controlar el cuadrado azul, comenzando en el centro del laberinto para llegar a la salida, señalada con el rectángulo rojo. Para esto deberá:

- Evitar tocar paredes. Cada vez que sucede, su puntaje se verá disminuido en 1 pt. Además, se emitirá un pitido en el MSP.

- Recolectar la fruta, (indicado con un cuadro color Magenta) este aparece aleatoriamente en las esquinas del laberinto.
- Una vez recolectada la fruta, será posible pasar de nivel.
- El objetivo es superar todos los niveles, en el menor tiempo posible y con el mayor puntaje.

2.5.2. Modos de juego

Se tendrán 2 modos de juegos disponibles, Acelerómetro y Joystick. Por defecto se comenzará en modo Joystick. Por lo que se deberá enviar el valor medido mediante UART siguiendo la estructura `X023Y127`. Es decir:

- Enviar identificador `X`
- Enviar el valor medido horizontal en un rango de 0 a 255, utilizando 3 dígitos.
- Enviar identificador `Y`
- Enviar el valor medido vertical en un rango de 0 a 255, utilizando 3 dígitos.

Cabe señalar que la interfaz no reconoce si los datos provienen del Joystick o del acelerómetro, por lo que ambos mantendrán la misma estructura. Al presionar el botón MODE, se podrá cambiar de modo, por lo que el microcontrolador deberá proceder a enviar los datos correspondientes a ese modo.

- Si está en Acelerómetro, se cambia a Joystick y enviará la letra `N`.
- Si está en Joystick, se cambia a Acelerómetro y enviará la letra `M`.

Cada vez que el cuadro azul toca un borde enviará los siguientes comandos, de modo que se reproduzca un pidido con las frecuencias indicadas en el Task 2.

- Al tocar con una pared, enviará la letra `C` indicando una colisión.
- Al tocar con la fruta, enviará la letra `F` indicando que tocó la fruta.
- Al tocar con la barra roja y si es que ya se comió la fruta, enviará la letra `W`, que indica que se ganó el nivel.

Además, dependiendo de la cantidad de golpes, se enviarán los siguientes comandos.

- Si la cantidad de golpes es igual a 1, enviará la letra `G`, indicando que se debe encender el led verde a modo de aviso.
- Si la cantidad de golpes es igual a 10, enviará la letra `B`, indicando que se debe encender el led azul a modo de aviso.
- Si la cantidad de golpes es igual a 20, enviará la letra `R`, indicando que se debe encender el led rojo a modo de aviso.

2.6. Recomendaciones y sugerencias

- No sobrecargue de instrucciones a las interrupciones. Las interrupciones deben ser acciones pequeñas que cambien estados del programa, no se espera que todo su programa esté en las interrupciones.
- El modo *debug* de Code Composer Studio es muy útil para monitorear cada línea del código. Se pueden seleccionar breakpoints para monitorear las interrupciones y llegar a esas líneas cuando se ejecuta una acción.
- Revisar y estudiar el Datasheet. Contiene información relevante para la realización del laboratorio.
- Utilizar máquinas de estado. Al igual que el laboratorio anterior, será muy necesario.
- Para hacer funcionar la interfaz, deberá indicar cual es el puerto a utilizar, esto se ejecuta de la siguiente forma:

```
-py game.py -b 57600 -p COM3
```

La instrucción -b indica el baudrate a utilizar y la instrucción -p el puerto conectado. Debe editarlo acorde a su computador, en caso de ser usuario de Windows, el proceso debería ser automático.

3. Lectura recomendada

- Páginas iniciales del Datasheet del MSP430F5529
- Capitulo 28: ADC12_A del MSP430x5xx and MSP430x6xx Family User's Guide.
- Capitulo 36: Universal Serial Communication Interface – UART Mode del MSP430x5xx and MSP430x6xx Family User's Guide.
- *Buzzer* `https://www.cuidevices.com/product/audio/buzzers/audio-transducers/cem-1203(42)`
- MSP430x5xx and MSP430x6xx Family User's Guide.
- MSP430F552x, MSP430F551x Mixed-Signal Microcontrollers datasheet.
- BOOSTXL-EDUMKI User Guide `https://www.ti.com/lit/ug/slau599a/slau599a.pdf`

4. Sesión de preguntas

4.1. Consideraciones generales

- Debe cumplir con el horario asignado y solamente tendrá aproximadamente 10 min para realizar la presentación. Se pide encarecidamente respetar esta regla.
- Cualquier consulta sobre los criterios de evaluación de cada laboratorio debe ser realizada en las issues, donde estará disponible para que sea revisada por todos los alumnos.
- Puede utilizar cualquier recurso que tenga disponible. Ya sea un block de notas, mostrar brevemente alguna parte de su código, tener una presentación de apoyo, el Datasheet, etc. Queda a criterio del ayudante corrector la solicitud de mostrar este apoyo.
- IMPORTANTE: se prohibirá el uso de funciones de Energia.nu para la programación de las tarjetas de desarrollo, esto por la simplicidad que involucra, por lo que deberán mostrar que entienden qué están realizando al momento de programar.

4.2. Preguntas

Responde satisfactoriamente preguntas aleatorias que abordan los siguientes temas:

- Especificaciones generales del MSP430F5529 en lo que respecta al uso del modulo de ADC.
- Especificaciones generales del MSP430F5529 en lo que respecta a *timers*. Tales como frecuencias admitidas, elección de modos, cantidad de contadores, cantidad de *timers*, etc.
- Funcionamiento del sistema del protocolo UART en el MSP, tanto a nivel general como a nivel específico.
- Comprender **cada línea** de su programa.
- Entendimiento de *Headers Files* o archivos de extensión .h. ¿Cuál es el beneficio que tiene la creación de bibliotecas y cómo ayuda a la estructuración de código ?
- ¿Qué significan las directivas #IFNDEF #DEFINE #ENDIF ?

5. Fecha y forma de de entrega

La entrega se debe realizar el día Miércoles 4 de noviembre las 9:59 hrs. Durante este día se realizará la sesión de preguntas en los horarios por publicar. Conectarse en el horario asignado para la ronda de *preguntas y respuestas*.

Deberá subir su **código completo y ordenado** a su repositorio de GitHub junto con un archivo Readme.md que indique las fuentes utilizadas y una lista de lo que no fue implementado en su código. De forma que el ayudante corrector lo tenga claro. Además debe indicar sistema operativo usado y/o si utilizó un compilador online.

El código entregado debe cumplir con lo siguiente, el no cumplimiento de estos puntos podría incurrir en un descuento:

- Programa compilable y sin errores. Recomendamos subir el proyecto completo generado en Code Composer Studio.
- Código ordenado y debidamente comentado. No se exigirá ningún estándar en particular, pero se debe mantener una estructura que permita una lectura fácil. [1] [2]
- Recursos utilizados se deben citar en el README.md de su repositorio.
- Su código debe tener un encabezado como el mostrado en el ejemplo de "Hello World!" del LAB 01.

[1] En https://developer.gnome.org/programming-guidelines/stable/c-coding-style.html.en pueden encontrar ejemplos de cómo mantener un buen formato de código.

[2] En https://codebeautify.org/c-formatter-beautifier pueden reformatear un código.