Lab 2: Output e Input Básicos (GPIOs)

Laboratorio Nivel 2

1. Objetivo

El presente laboratorio tiene por propósito el introducirlos al funcionamiento básico de sus microcontroladores, a través del manejo de entradas y salidas básicas (*General Purpose Inputs and Outputs* o GPIOs). Se aprenderá cómo escribir códigos en un computador, tanto en Assembly como en C, para posteriormente cargarlos y ejecutarlos en un microcontrolador. A su vez se sugieren las herramientas disponibles para la realización de esta actividad.

A modo de concretar este objetivo, deberán comprender el cómo operan estos puertos a través de una lectura a conciencia de los respectivos *datasheets*. Con el objetivo de llegar a una actividad en donde se probarán estas actividades.

Esta experiencia estará separada en 3 partes, de las que solo se revisarán las últimas 2.

1. La primera actividad, es para familizarizarse con las herramientas disponibles del curso.
2. La segunda actividad, es para ver las semejanzas de la realización de código en Assembly y en C.
3. La tercera actividad, complementa la segunda, al incorporar el uso de botones y debouncing.

2. Descarga e instalación de plataforma de desarrollo

2.1. Windows - macOS - Linux

En este caso se hará uso del programa Code Composer Studio (CCS versión 8 o superior), el cual se encuentra disponible en la página de Texas Instruments[1]. Una vez descargado el instalador correspondiente a su sistema operativo, deben ejecutar el instalador correspondiente.

Al iniciar la instalación deberán aceptar los términos y condiciones y seleccionar la familia de procesadores, en este caso la opción corresponde a **"MSP430 ultra-low power MCUs"**. **Es importante que la ruta en la que se instale CCS no debe contener espacios ni tildes**. Cuando el programa termine de instalarse y se inicie deberán crear un nuevo proyecto

[1] A agosto de 2020, se está en versión 10 http://www.ti.com/tool/CCSTUDIO

y seleccionar la carpeta de destino (nuevamente, en una ruta que no contenga espacios ni tildes). Recomendamos que esta carpeta sea creada en la carpeta clonada del repositorio privado en el que ustedes trabajarán.

Cuando inicien un proyecto, deberán seleccionar el modelo del microcontrolador, el cual corresponde al **MSP430F5529**. Una vez terminen de escribir y guardar su código deberán presionar "Build" y luego "Flash". Una buena práctica para la detección de errores y depuración del código es probarlo con la opción "Debug", la cual permite monitorear el comportamiento de su programa línea a línea.

Nota: Por buena praxis, al usar el modo "Flash" se recomienda presionar el botón de reset luego de cargar el código a su placa. Esto debido a que en algunos casos el nuevo código empezará a ejecutarse en el microcontrolador después de resetearse.

2.2. Alternativa macOS

Para proceder a la instalación del compilador y su respectivo toolchain requerimos instalar en primer lugar el gestor de paquetes Homebrew mediante el comando:

```
/usr/bin/ruby -e "$(curl -fsSL https://raw.githubusercontent.com/Homebrew/install/master/install)"
```

Luego aceptamos e instalamos el complemento de Xcode mediante el siguiente comando (omitir si ya se ha hecho):

```
xcode-select --install
```

Este proceso requiere de varias librerías las cuales se instalarán mediante Homebrew de la siguiente forma:

```
brew tap sampsyo/mspgcc

brew install msp430-libc

brew install mspdebug

brew install git

git clone https://github.com/sampsyo/homebrew-mspgcc.git

```

```
cd homebrew-mspgcc

./addlinks.sh
```

Finalmente se debe descargar los drivers de las placas msp430 mediante el link https://github.com/loky32/msp430_OSX.git.

Movemos el archivo `ibmsp430.so` a la ubicación `/usr/local/lib/`. Con esto concluye la instalación y ahora puedes compilar y subir a la placa msp430. Probamos el compilador mediante el siguiente comando:

```
msp430-gcc -Wall -Wextra --std=gnu99 -mmcu=msp430f5529 -Os -o nombre.elf nombre.c
msp430-objcopy --output-target=elf32-msp430 nombre.elf nombre.bin
```

Finalmente subimos el archivo compilado a la placa mediante el comando:

```
mspdebug tilib "prog nombre.bin"
```

2.3. Alternativa Linux

Para la debida instalación de el compilador de terminal se requiere de la descarga directa del programa en el link: http://www.ti.com/tool/msp430-gcc-opensource, donde deben seleccionar MSP430-GCC-OPENSOURCE $->$ Get Software, y luego seleccionar el programa correspondiente a su sistema operativo, en este caso se usará

- `"msp430-gcc-full-linux-x64-installer-6.1.0.0.run"`

Una vez descargado en la terminal ejecutar:

```
chmod +x msp430-gcc-full-linux-x64-installer-6.1.0.0.run
sudo ./msp430-gcc-full-linux-x64-installer-6.1.0.0.run
```

Eso abrirá una ventana con el instalador. Seguir los pasos de instalación, el cual instalará todo en una ubicación similar a: `/home/user/ti/gcc`

Bastaría agregar al *bash profile* ubicado en normalmente en el archivo: `/home/user/.bashrc` (Ubuntu 18.04, podría tener otro nombre en un sistema operativo distinto) la siguiente linea:

- `export PATH="$PATH:/home/USER/ti/gcc/bin"`

Ya debería estar todo listo para poder pasar tus programas a la tarjeta de desarrollo, se recomienda la creación de una función agregando al archivo anterior las lineas:

```
function mspcompile () {
msp430-gcc -Wall -Wextra --std=gnu99 -mmcu=msp430f5529 -Os -o out_msp.elf "$@"
msp430-objcopy --output-target=elf32-msp430 out_msp.elf out_msp.bin
mspdebug tilib "prog out_msp.bin"
rm out_msp.bin
rm out_msp.elf
}
```

3. Descripción de la actividad

3.1. *Task 1*: Blink Led!

Cabe señalar que esta actividad será guiada en clases de laboratorio.

3.1.1. C

En la carpeta `MSP430F5529` se encuentra el archivo `hello_world_msp.c`. Este debe subirse a la placa siguiendo los pasos más arriba. Luego, un LED de esta parpadeará.

Recomendamos analizar este código y entender el procedimiento que está sucediendo. Para ello el uso del modo debug será útil.

3.1.2. Assembly

En la carpeta `MSP430F5529` se encuentra el archivo `hello_world_msp.asm`. Este debe subirse a la placa, con la diferencia que a la hora de crear el proyecto en CCS, se debe seleccionar la opción "Empty Assembly only proyect". Luego, un LED de esta parpadeará.

Recomendamos analizar este código y entender el procedimiento que está sucediendo. Para ello el uso del modo debug será útil.

3.2. *Task 2*: OUTPUTs, Colores RGB

Para la siguiente actividad, deberá conectar el Booster Pack MKII a su microcontrolador, tenga precaución con la orientación ya que esto podría dañar gravemente la tarjeta. La orientación es la siguiente:

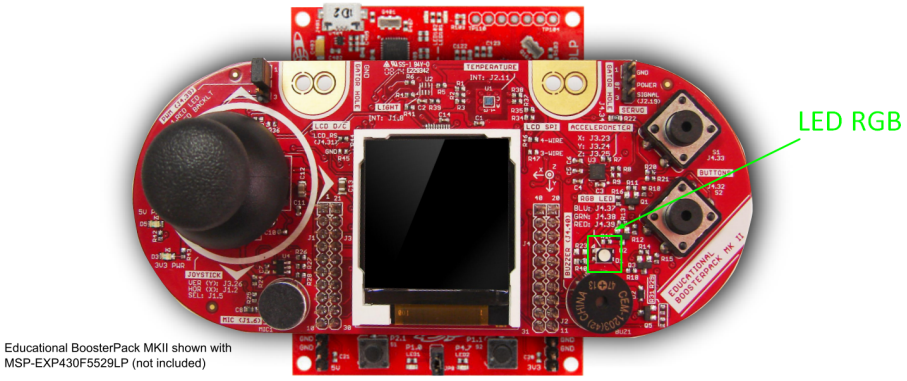

Figura 1: Orientación y led RGB. Fuente: BOOSTXL-EDUMKII Quick Start Guide

Como observará en la figura 1, existe un led RGB que será el que se utilizará en este laboratorio. Este permite realizar distintos colores al ir mezclando las intensidades de los canales Rojo, Verde y Azul, siendo la base de un modelo de color aditivo. [2] Sin embargo, si solo se utilizan los colores iniciales, sin variación de intensidad, es posible desplegar la siguiente lista de colores, que también es posible verlo en la figura 2.

[2]Modelo de color RGB: https://www.hisour.com/es/rgb-color-model-24867/

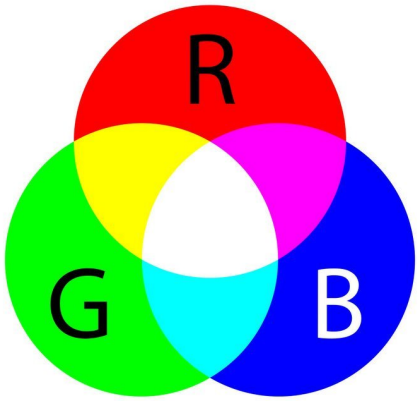

Figura 2: Modelo RGB

- Rojo
- Magenta (Rojo + Azul)
- Azul
- Cyan (Azul + Verde)
- Verde
- Amarillo (Verde + Rojo)
- Blanco (Rojo + Verde + Azul)

3.2.1. Assembly (25 %)

Teniendo esta información, la actividad a realizar consistirá en crear una secuencia que itere sobre estos colores, comenzando con el rojo. Para comenzar esta actividad es necesario determinar cuáles son los puertos asociados a este led y determinar si están conectados en ánodo o cátodo común. La información necesaria está disponible en los datasheets del Booster Pack MKII y del microcontrolador

Debe cumplir con lo siguiente:

1. El delay entre cada color debe estar cercano al medio segundo (aproximado, no preciso).
2. Luego del color blanco, debe volver al color rojo.
3. Continuar indefinidamente.

3.2.2. C (25 %)

Mismas consideraciones que Assembly, pero hecho en C. La idea de esta actividad es que pueda identificar las diferencias existentes entre Assembly y C.

3.3. Task 3: Seleccionador de secuencia RGB. (50 %)

En esta sección se busca introducir al alumno a la lectura de entradas digitales y cómo procesar estas para realizar alguna acción, **utilizando el lenguaje C**. Esto se realizará mediante la configuración de tres botones:

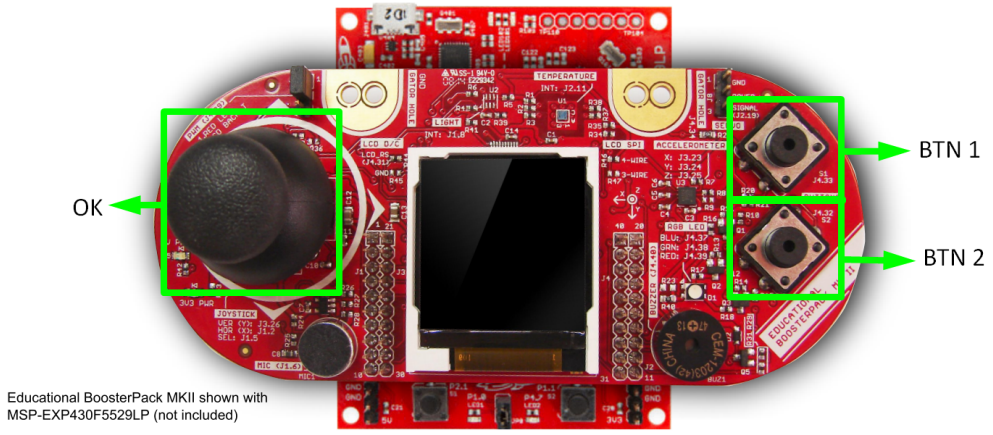

Figura 3: Botones a utilizar

La actividad consistirá en que a partir de los colores disponibles en el LED RGB (los 7 ya mencionados en la lista), se puedan seleccionar 4 (que pueden ser iguales o distintos) para que luego sean desplegados en una secuencia que se repita 4 veces. Para elegir los colores, se realizará el siguiente procedimiento:

Selección de colores

Si consideramos la secuencia de colores mostrada en la figura 2 de forma cíclica, cada vez que se presione el BTN1, indicado en la figura 3 seguiremos un secuencia en el sentido de las manecillas del reloj :

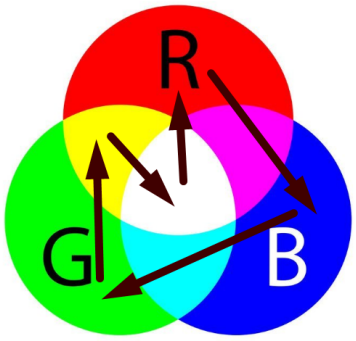

Figura 4: Secuencia horaria con BTN1

En cambio si presionamos el BTN2, seguiremos una secuencia en sentido contrario a las manecillas del reloj:

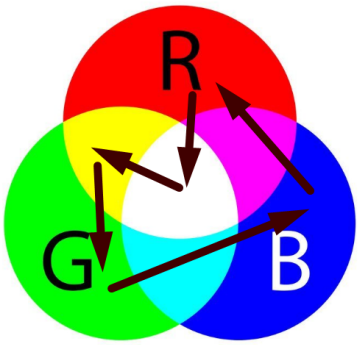

Figura 5: Secuencia antihoraria con BTN2

Cada vez que se presiona el botón, se deberá avanzar solo 1 color y deberá visualizarse automáticamente en el led RGB. Cuando se llegue al color seleccionado, se deberá presionar un botón OK, que corresponde al botón del Joystick, tal como está indicado en la figura 3.

Led Status

Para que el usuario sepa en qué posición de este arreglo se encuentra actualmente, se utilizarán leds de status. Estos corresponderán a los leds que trae por defecto el Launchpad del MSP430F5529 y que se destacan a continuación:

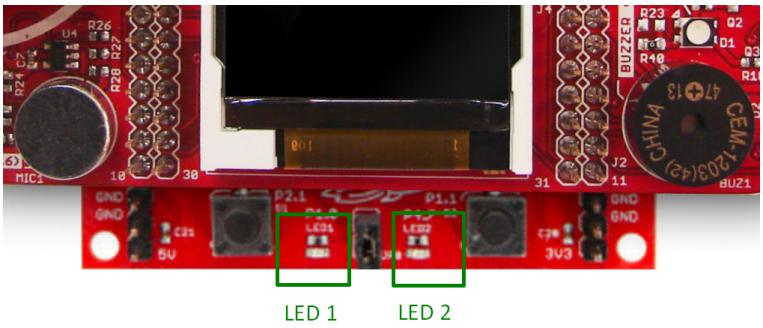

Figura 6: Led de status

Estos actuarán como una numeración binaria, que dependiendo de si están encendidos o apagados, indicarán una posición. Esto se detalla en la tabla 1:

Posición / Iteración	LED 1	LED 2
1	OFF	OFF
2	OFF	ON
3	ON	OFF
4	ON	ON

Tabla 1: Tabla de led de status

Visualización arreglo

Al terminar de escoger el último color de este arreglo, se debe mostrar la secuencia almacenada 4 veces. Para saber qué número de iteración se está mostrando, nuevamente se utilizarán los leds de estatus indicados en la tabla 1. Con la diferencia que ahora cambiarán solamente al concluir cada ciclo.

Al momento de terminar la visualización de los colores, se deberá volver al estado inicial y se tiene que permitir seleccionar una nueva secuencia de colores siguiendo los mismos pasos ya mencionados.

4. Consideraciones y recomendaciones

A continuación se listarán una serie de consideraciones y recomendaciones que se deben tomar en cuenta a la hora de la realización del Task 3.

4.1. Consideraciones

4.1.1. Debouncing por software

Para el caso de los botones 1 y 2 experimentará el efecto de *bounce*, por lo que deberá diseñar mediante *software* una rutina que se encargue del *debouncing* de dicho botón. En el caso del botón del botón del Joystick, este efecto podría ser menor, debido a que contiene un debouncing por *hardware*. Son embargo, de requerirlo, deberá incorporar también debouncing en este botón.

Lo que se pide en esta actividad es que primero configure los botones 1 y 2 del Booster Pack MKII para que, cuando sea presionado, aumente el ciclo y cambie el color. Este proceso debe ser cíclico, por lo que al llegar al último color se debe volver al inicial. Como complemento de estos botones, se deberá configurar el botón del Joytick para que, cuando

tenga una lectura de que fue presionado, actúe como un "Ok".

Su programa debe contemplar que mientras alguno de los tres botones se encuentre presionado, presionar otro no tenga efecto.

IMPORTANTE. El cambio solo debe ocurrir en el instante en que el botón es presionado, no antes, no después, no después de un pequeño delay, no cuando es liberado ni tampoco cuando se mantiene presionado.

En la figura 7, se muestra lo que sucede cuando se presiona un botón [3] muestra lo que ocurre cuando un botón es presionado.

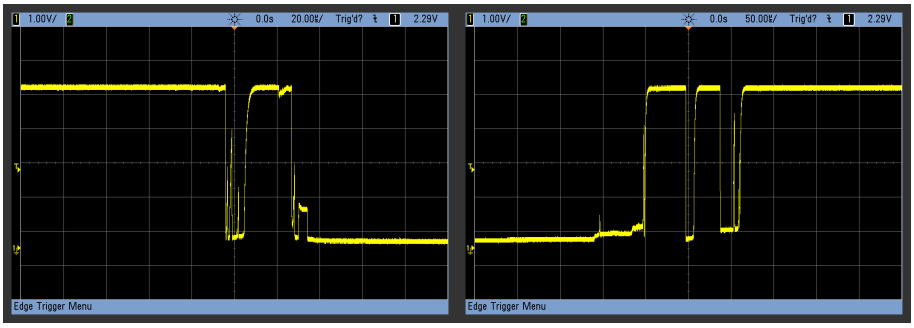

Figura 7: Bouncing de un botón

Por otra parte, la figura 8 muestra la lectura que se espera:

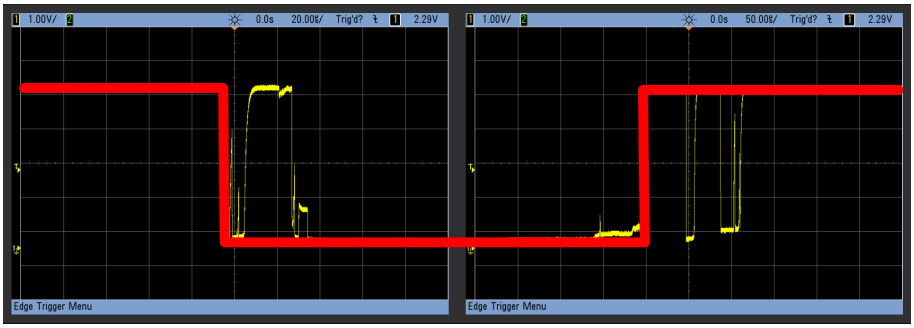

Figura 8: Lectura ideal esperada

[3] Fuente imagen: Hackaday https://hackaday.com/2015/12/09/embed-with-elliot-debounce-your-noisy-buttons-part-i/

4.1.2. Otras consideraciones

- Es importante notar que al estar configurando pines como entradas, es necesario utilizar resistencias de *pull-up*. Estas vienen en general dentro de la tarjeta o se pueden utilizar resistencias externas, pero para este laboratorio es un requerimiento que usen las internas.
- Deben definir si es que utilizarán resistencias de *pull-up* o *pull-down* y si es que es posible realizar esto o no.
- El proceso de debouncing puede ser realizado mediantes interrupciones de pin e interrupciones de timers. Sin embargo en este laboratorio no se evaluarán, por lo que se puede utilizar *delays*.
- El cambio de color solo debe ocurrir al momento de presionar el botón. Si este se mantiene presionado, no se deben cambiar los colores indefinidamente. Si se suelta, no debe ocurrir nada.

4.2. Recomendaciones

Este Task, al deber realizar múltiples consideraciones e incorporar distintos puertos, no es recomendable intentar realizarlo completamente la primera vez. Para ello sugerimos lo siguiente:

- Comenzar identificando los puertos a utilizar y dejarlos anotado como comentario en una parte del código. Ayudará a la memoria y recordar cuál es cuál.
- Realizar el *debouncing* de cada botón por separado. Para esto un buen código de inicio sería encender y apagar un led, cada vez que el botón se presiona. Esto ayudará a corroborar que si se mantiene presionado el botón, no hay cambios; y si se suelta, tampoco los habrán.
- El modo *debug* de Code Composer Studio es muy útil para monitorear cada línea del código. Además puede ayudar a identificar que el cambio se produce cuando se presiona el botón y no después de un pequeño delay.
- Revisar y estudiar el Datasheet. Contiene información relevante para la realización del laboratorio.
- Utilizar máquinas de estado.

5. Lectura recomendada

- Páginas iniciales del Datasheet del MSP430F5529
- Capitulo 12: Digital I/O Module del MSP430x5xx and MSP430x6xx Family User's Guide.
- Secciones 5 y 6 del MSP430F552x, MSP430F551x Mixed-Signal Microcontrollers datasheet.
- 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`

6. Sesión de preguntas

6.1. 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.

6.2. Preguntas

Responde satisfactoriamente preguntas aleatorias que abordan los siguientes temas:

- Especificaciones generales del MSP430F5529. Esta información puede encontrarse al comienzo del *datasheet*.
- Especificaciones generales del compilador/programa que está utilizando (Debe entender su funcionamiento).
- Especificaciones generales del MSP430F5529 en lo que respecta a inputs y outpus. Tales como existencia de resistencias internas, voltajes y corrientes máximas admitidas.
- Conceptos iniciales del tiempo del procesador y el significado o generación de delays().
- Conceptos relacionados con Assembly en el MSP430.
- Configuración de pull-ups y pull-down si aplica.
- Comprender **cada línea** de su programa.
- Entendimiento del concepto de debouncing por software y su diferencia por hardware.

7. Fecha y forma de de entrega

La entrega se realizará en 2 partes.

- Miércoles 16 de septiembre a las 13:00 hrs. Task 2. Solo entrega de código, sin sesión de preguntas.
- Lunes 5 de octubre a las 9:59 hrs. Task 3. Durante este día se realizará la sesión de preguntas en los horarios por publicar.

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.

Conectarse en el horario asignado para la ronda de *preguntas y respuestas*
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. [4][5]
- 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.

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

8. Bonus Assembly

Para quienes estén interesados en profundizar un poco más en Assembly, existirá un bonus de 1 décima en el promedio final de los laboratorios (equivalentes a aproximadamente 8 décimas en este LAB) por la realización de una versión simplificada del Task 3 en este lenguaje. Las consideraciones son las siguientes:

- Realizar el Task 3 en Assembly no reemplaza realizar el Task 3 en C. Por lo que de realizar un bonus, se deben entregar ambos proyectos.
- Se considerará realizar una versión simplificada de este Task:
  - Se puede simplificar a solo utilizar 2 botones, pudiendo eliminar el BTN1 o BTN2.
  - Se puede prescindir de leds de status.
- Existirá un punto intermedio de este bonus correspondiente a 0,5 décimas en caso de que no esté realizado de forma correcta el bonus. (Se recuerda que el promedio final es calculado con 2 décimales).