# Programación avanzada de sistemas embebidos en RTOS y Linux Embebido

**Trabajo Práctico: Manejo de interrupciones con FreeRTOS**

Dr. Julio Dondo - Mg. Romina Molina

## Plataforma de hardware

**Aclaración para la placa de desarrollo ZYBO:** para que Vivado reconozca la plataforma como Board, copiar la carpeta zybo-z7-10 en el directorio de instalación de Vivado:

/opt/.../Vivado/2016.4/data/boards/board_files/zynq/

Hacer uso de la terminal en modo superusuario y del comando cp.

### Configuración del proyecto

. Abrir el programa **Vivado** de Xilinx.

. Hacer click en **Create New Project**. En la ventana emergente hacer click en **Next** para iniciar el proceso de creación y configuración del nuevo proyecto.

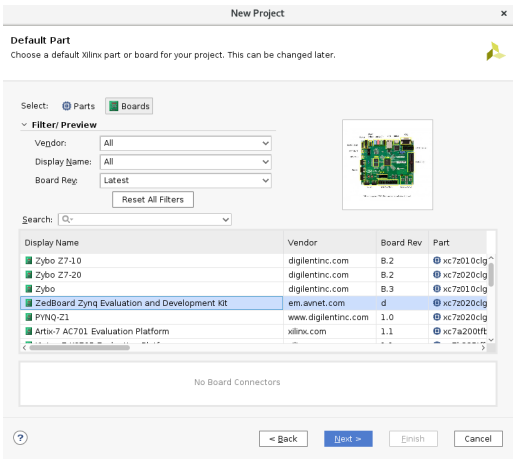
. Colocar el nombre del proyecto y seleccionar el directorio de ubicación del mismo. Hacer click en **Next**.

. En la ventana siguiente seleccionar la opción **RTL Project** en Project Type Form y hacer click en **Next**.

. En la ventana Add Sources hacer click en **Next**.

. En la ventana para agregar Constraints hacer click en **Next**.

. En Default Part, hacer click en **Boards** y seleccionar la plataforma correspondiente (Zedboard o Zybo). Hacer click en **Next**.

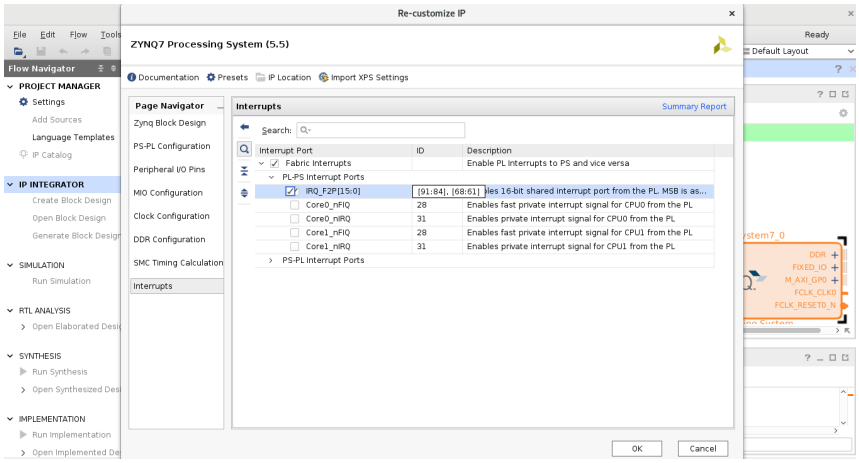



. Hacer click en **Finish** para crear el proyecto.

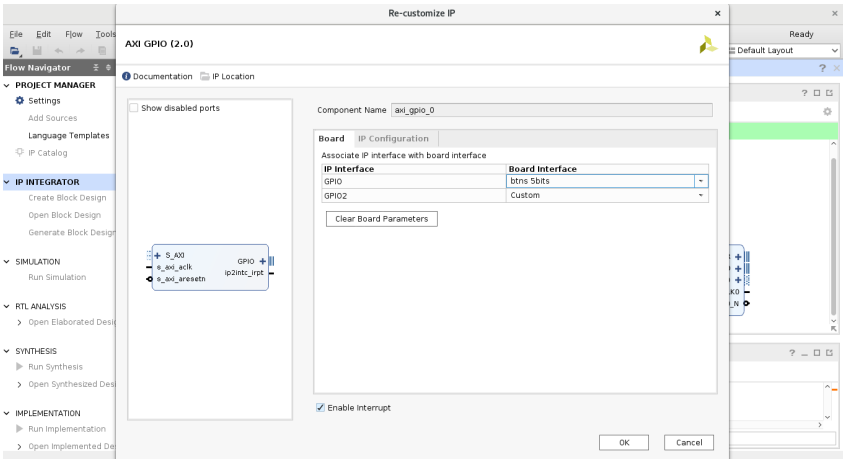
## Creación de la plataforma de hardware

1. En el panel izquierdo (Flow Navigator) hacer click en **Create Block Design**. En la ventana emergente hacer click en **Ok**.

2. Agregar el IP **ZYNQ7 Processing System**. Para configurarlo hacer doble click en el bloque IP, ir a **Interrupts**, habilitar **Fabric Interrupts**. Expandir **PL-PS interrupt Ports**, tildar la opción **IRQ_F2P[15:0]**. Hacer click en **Ok**.




3. Agregar el IP **AXI GPIO** para los botones: hacer doble click en el bloque IP. En Board, Board Interface elegir **btns 5bits** para GPIO, tildar la opción **Enable Interrupt**. Hacer click en **OK**.

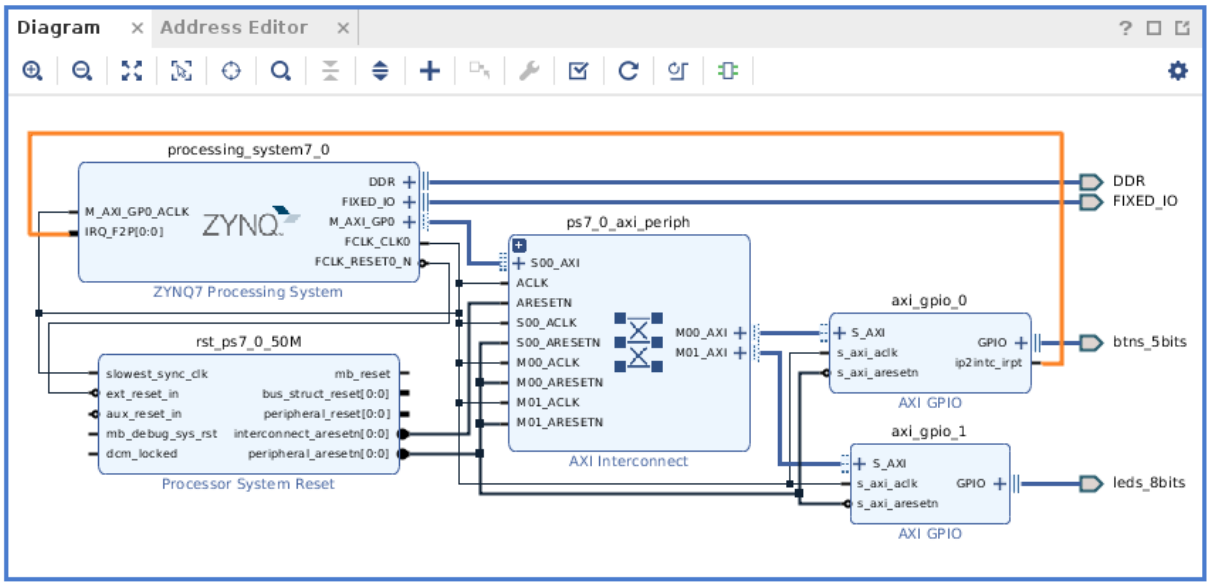



4. Agregar el IP **AXI GPIO** para los leds. Para configurarlo hacer doble click en el bloque IP. En Board, Board Interface elegir **leds 8bits** para GPIO. Hacer click en **OK**.

5. Hacer click en **Run Block Automation**. Destildar la opción **Apply board preset**. Click en **Ok**.

6. Hacer click en **Run Connection Automation**. En la ventana emergente tildar la opción **All Automation**. Click en **Ok**.

7. Conectar de forma manual la salida **ip2intc_irpt** del bloque GPIO correspondiente a los botones a la entrada de interrupción **IRQ_F2P[0:0]** del sistema de procesamiento.




8. Seguidamente, crear **HDL Wrapper**.

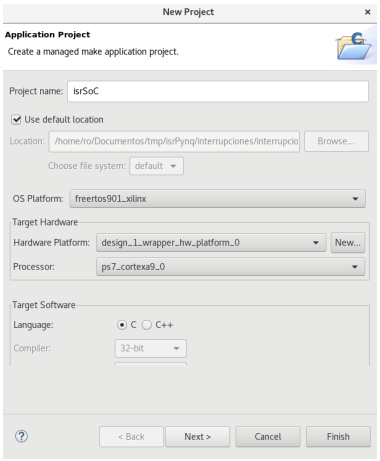
9. Ejecutar la síntesis, implementación y finalmente generar el bitstream.

10. Exportar el hardware generado: ir a File → Export → Export Hardware. Tildar la opción **Include Bitsream**.

11. Ir a File → Launch SDK.

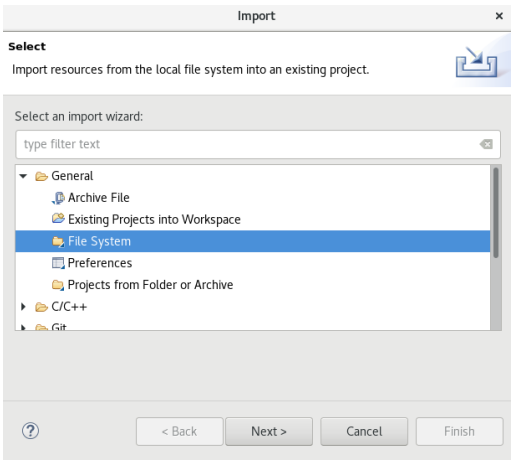
## Creación de proyecto en SDK

1. En SDK, crear un nuevo proyecto. En la opción **OS Platform** elegir FreeRTOS en la versión correspondiente. Hacer click en **Next**.




2. En la ventana emergente de templates, seleccionar **Empty Application**. Hacer click en **Finish**.

3. Una vez creado el nuevo proyecto, importar el archivo **simpleApp.c**, el cual se utilizará de base para el resto de los ejercicios. Para esto, dentro del proyecto creado en SDK, hacer click derecho sobre la carpeta **src** y luego click en **Import...**. En la ventana emergente, desplegar la opción **General** y elegir **File System**. Hacer click en **Next**. Seguidamente, elegir el directorio donde se encuentra el archivo simpleApp.c y hacer click en **Aceptar**. Seleccionar el archivo y hacer click en **Finish**.



## Manejo de Semáforo Binario y Mutex

4. **Semáforo Binario**
   (a) Crear el objeto **xSemaphoreHandle semHandler**
   (b) Inicializar **semHandler** para hacer uso del semáforo binario.
   (c) Generar una tarea **taskLED** que incremente una variable **contador** y que muestre el valor de la misma por los leds.
   (d) Generar una tarea **taskPRINT** que muestre por la terminal el valor de la variable **contador**.
   (e) En la tarea **taskLED**, cuando el valor llegue a 8, hacer uso del semáforo binario para que la tarea **taskPRINT** muestre el valor de la variable.

5. **Mutex**
   (a) Inicializar la variable **semHandler** para hacer uso de mutex.
   (b) Modificar la tarea **taskLED** de manera tal de que muestre el valor de la variable **contador** por la terminal.
   (c) Generar una tarea para la lectura de los botones **taskBTN**. Esta tarea debe mostrar por la terminal el valor del botón cuando es presionado.
   (d) Hacer uso de **Mutex** para la gestión del recurso compartido.

# Interrupciones externas

1. Generar un nuevo proyecto en SDK e importar el archivo isrBtn_FreeRTOS.c.

2. Generar una tarea **taskLED** que incremente una variable **contador** y que muestre el valor de la misma por los leds.

3. Generar una tarea **taskISRHandler**, la cual será la encargada de mostrar el valor de 1 por los leds cuando un botón es presionado.

4. Hacer uso del semáforo binario para la sincronización entre **GpioIsr** y **taskISRHandler**: La función **GpioIsr** debe leer el estado de los botones y hacer uso de **xSemaphoreGiveFromISR()**. La función **taskISRHandler** debe verificar el estado del semáforo mediante **xSemaphoreTake()** para mostrar el valor de 1 por los leds cuando ocurre la interrupción por evento de botón.

5. Dentro de la función **GpioIsr**, comentar cuál es la función de **xHigherPriorityTaskWoken** y **portYIELD_FROM_ISR(xHigherPriorityTaskWoken)**.