



CESE – SOTR I

Clase 6: Simulacro de Examen



Ing. Juan Manuel Cruz

jcruz@fi.uba.ar



Gerente de Ingeniería - Cia. Hasar SAIC

Profesor Asociado Ordinario - Técnicas Digitales II UTN FRBA

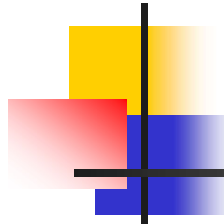
Profesor Adjunto Regular - Sistemas Embebidos FIUBA

ACSE - Laboratorio de Sistemas Embebidos FIUBA



Asociación Civil para la Investigación,
Promoción y Desarrollo de los
Sistemas Electrónicos Embebidos

2024 – ABR – 06

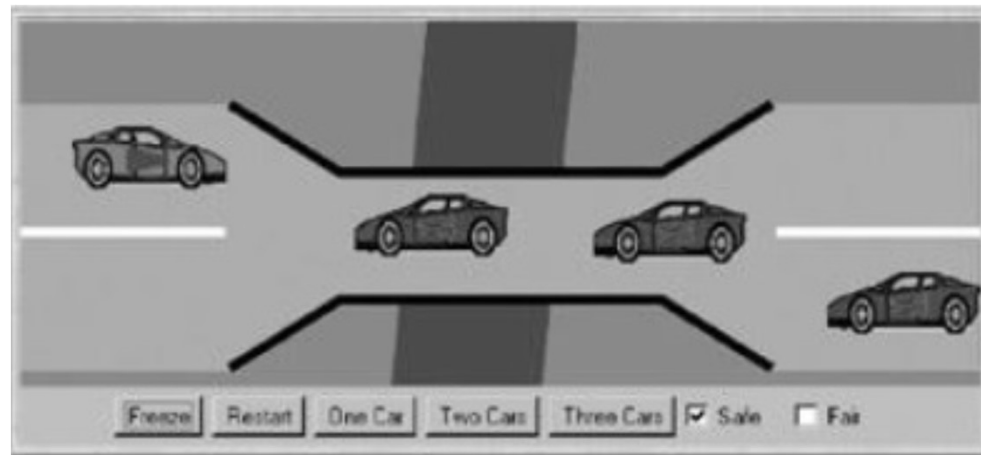


Temario

- Simulacro de examen
- Enunciado
- Implementación

Enunciado

- **Implementar** un **Sistema de Control de Acceso** que **monitorea & controla** el **ingreso/egreso** de **vehículos** (sólo un punto de ingreso y sólo un punto de egreso) a un **punte estrecho** de **capacidad** limitada (**sólo** un **vehículo**)

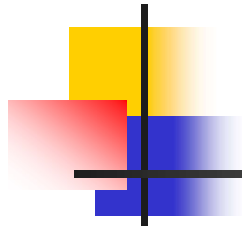


- Se trata de **una ruta de doble sentido** que llega a un **punte estrecho** por el que **cabe un solo vehículo**



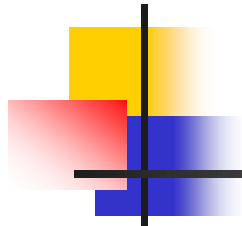
Enunciado

- Para la implementación se sugieren **dos tareas** (una para cada punto de **ingreso/egreso**), a saber:
 - `void task_a(void *parameters);` // Monitoreo y Control ...
 - `void task_b(void *parameters);` // Monitoreo y Control ...
- Observaciones:
 - Al **punto estrecho** los vehículos, **ingresan/egresan** por **orden** de llegada y **uno** a la vez
 - La **capacidad** del **punto estrecho** está **limitada** a **sólo un** vehículo
 - Cada uno de los puntos de **ingreso/egreso** al **punto estrecho** cuenta con un **semáforo vial** a controlar



Enunciado

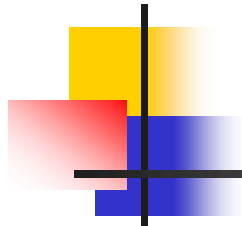
- A falta de hardware para estimular a las tareas, contamos con una tercera tarea de test (que periódicamente recupera un **estímulo** de un **array** y lo envía las otras **tareas**):
- `void task_test(void *parameters);`
 - **Estímulos** => *Entry_A, Exit_A, Entry_B, Exit_B*
- Project => [freertos_app_example_001.zip](#) (foder: **app**)



Enunciado

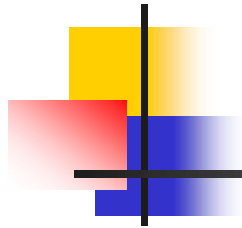
IMPORTANTE: Se recomienda implementar con "**semáforos binarios** y **mutex**" y seguir los siguientes pasos:

- Tarea de **generación** de **estímulos** **task_test()**
 - Resolver la generación de **estímulos** p/probar el funcionamiento de **task_a()** y **task_b()**
- Tarea de monitoreo/control de punto de **ingreso/egreso** **task_a()**
 - Resolver la **sincronización** con los **estímulos generados** por **task_test()**
 - Resolver el **acceso** a **recursos compartidos** con la tarea de monitoreo/control de **ingreso/egreso** **task_b()**



Enunciado

- Tarea de monitoreo/control de punto de **ingreso/egreso**
task_b()
 - Resolver la **sincronización** con los **estímulos generados** por **task_test()**
 - Resolver el **acceso** a **recursos compartidos** con la tarea de monitoreo/control de **ingreso/egreso** **task_a()**
- Indique **modificaciones** necesarias para gobernar los **semáforos viales** (**Rojo**, **Verde**) en las entradas al **punto estrecho**
- **Subir al Campus**: La carpeta **app** (comprimida en un archivo del tipo **".zip** o **".rar** ", nombrar: **freertos_app_example_001-Apellidos_Nombres.zip**



Implementación

- Tarea de generación de **estímulos** **task_test()**
 - Resolver la generación de **estímulos** p/probar el funcionamiento de **task_a()** y **task_b()**
 - **void task_test (void *parameters);**
 - **Estímulos** => **Entry_A, Exit_A, Entry_B, Exit_B**
- ¿Qué recursos provee **FreeRTOS** para sincronizar tarea entre sí o tareas con interrupciones?
 - **Semáforos** (Binarios, Contadores, Mutex)
 - **Colas**



Implementación

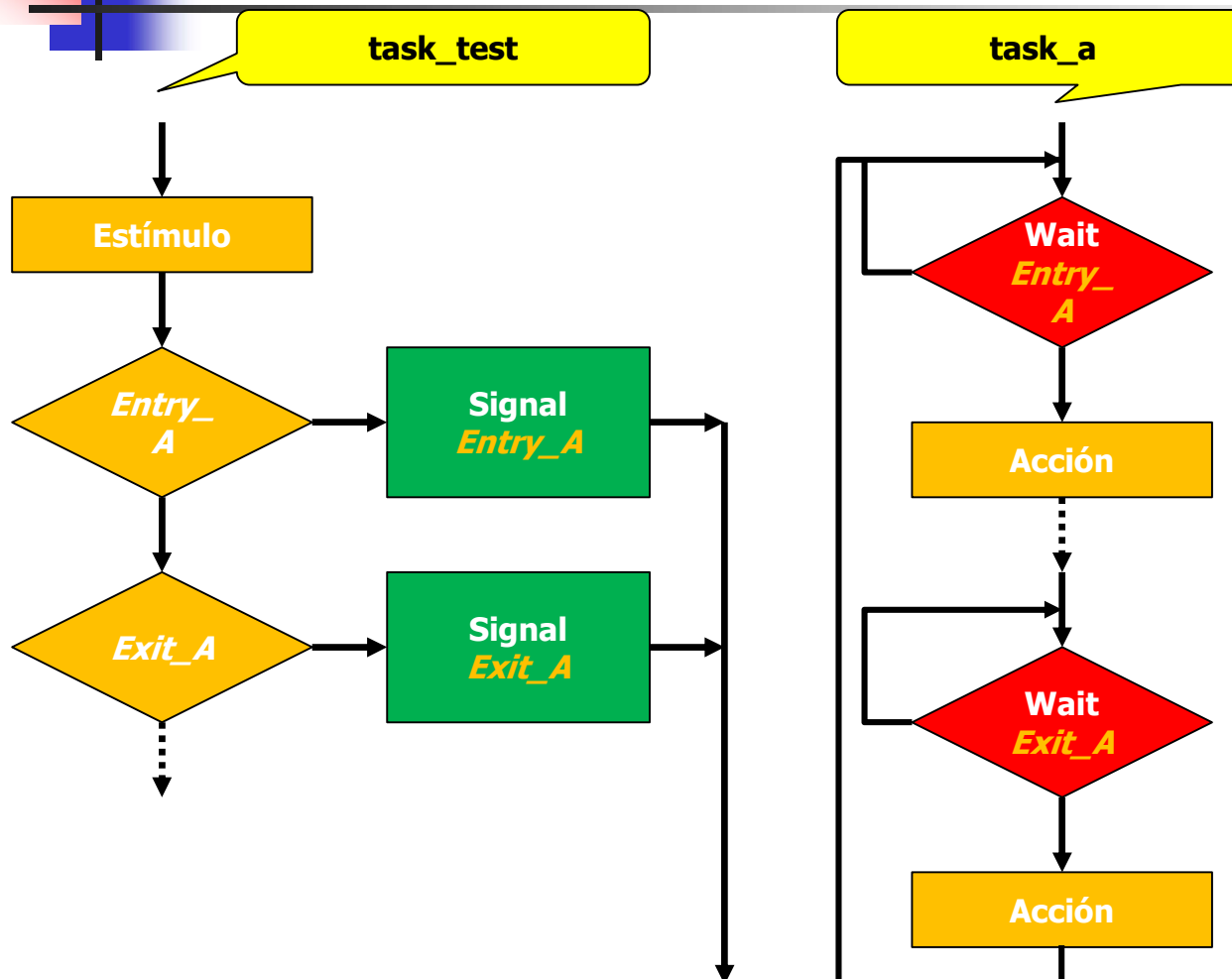
- ¿Qué **recurso** de FreeRTOS elegimos para **sincronizar** dichas **tareas**, **cuántos** necesitamos y qué **primitivas** usamos?
 - **Semáforo Binario**, **4** (cuatro), uno para cada **estímulo**
 - SemaphoreHandle_t **xSemaphoreCreateBinary**(void)
 - <https://www.freertos.org/xSemaphoreCreateBinary.html>
 - **xSemaphoreGive**(SemaphoreHandle_t xSemaphore)
 - <https://www.freertos.org/a00123.html>
 - **xSemaphoreTake**(SemaphoreHandle_t xSemaphore, TickType_t xTicksToWait)
 - <https://www.freertos.org/a00122.html>



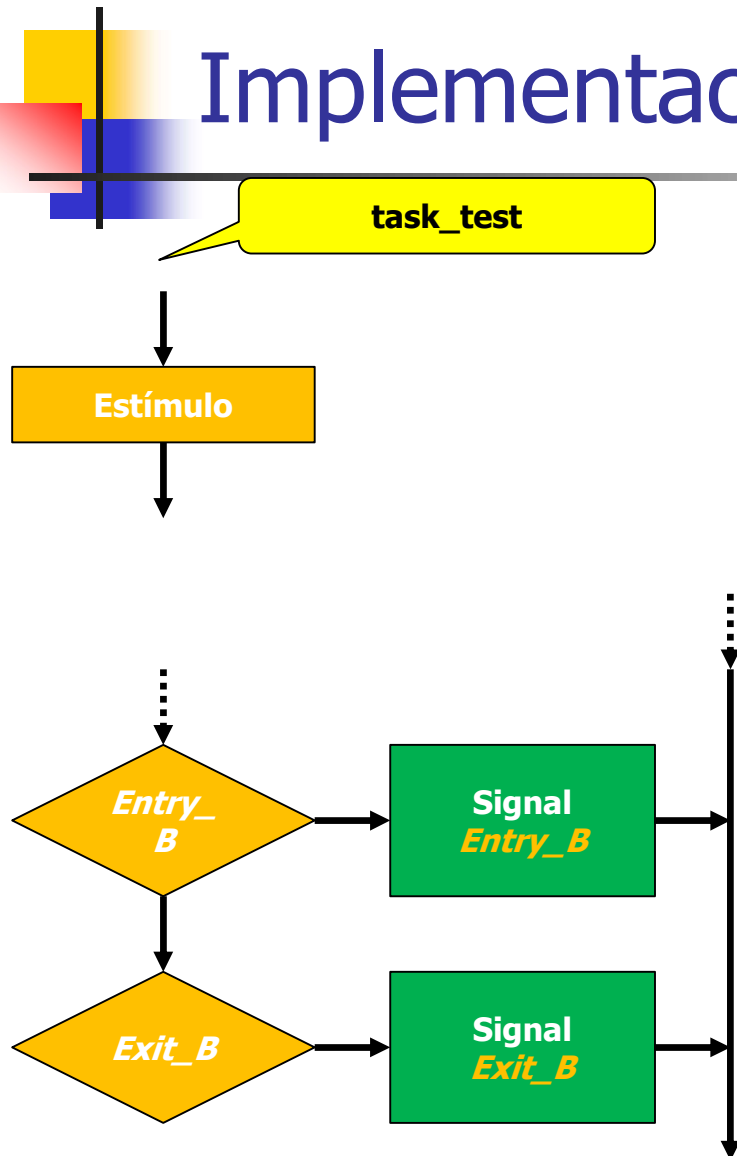
Implementación

- ¿En qué partes del programa usamos c/u de las **primitivas** y **porqué**?
 - SemaphoreHandle_t **xSemaphoreCreateBinary**(void)
 - **app_init**(void)
 - **xSemaphoreGive**(SemaphoreHandle_t xSemaphore)
 - void **task_test** (void *parameters);
 - **xSemaphoreTake**(SemaphoreHandle_t xSemaphore, TickType_t xTicksToWait)
 - void **task_a**(void *parameters);
 - void **task_b**(void *parameters);

Implementación

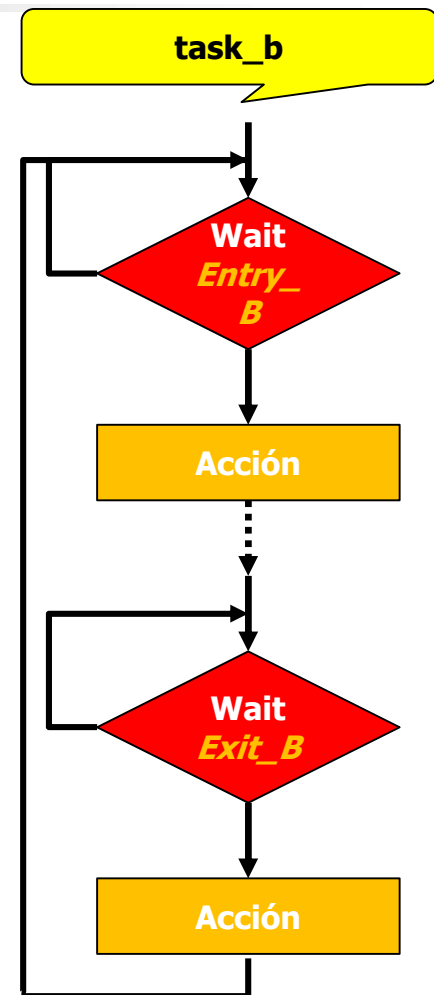


Implementación



2024 – ABR – 06

Ing. Juan Manuel Cruz



12



Implementación

Example usage:

```
SemaphoreHandle_t xSemaphore;  
  
void vATask( void * pvParameters )  
{  
    /* Attempt to create a semaphore. */  
    xSemaphore = xSemaphoreCreateBinary();  
  
    if( xSemaphore == NULL )  
    {  
        /* There was insufficient FreeRTOS heap available for the semaphore to  
        be created successfully. */  
    }  
    else  
    {  
        /* The semaphore can now be used. Its handle is stored in the  
        xSemaphore variable. Calling xSemaphoreTake() on the semaphore here  
        will fail until the semaphore has first been given. */  
    }  
}
```



Implementación

```
/* A task that uses the semaphore. */
void vAnotherTask( void * pvParameters )
{
    /* ... Do other things. */

    if( xSemaphore != NULL )
    {
        /* See if we can obtain the semaphore. If the semaphore is not
        available wait 10 ticks to see if it becomes free. */
        if( xSemaphoreTake( xSemaphore, ( TickType_t ) 10 ) == pdTRUE )
        {
            /* We were able to obtain the semaphore and can now access the
            shared resource. */

            /* ... */

            /* We have finished accessing the shared resource. Release the
            semaphore. */
            xSemaphoreGive( xSemaphore );
        }
        else
        {
            /* We could not obtain the semaphore and can therefore not access
            the shared resource safely. */
        }
    }
}
```



Implementación

- ¿Qué **recurso** de FreeRTOS elegimos para permitir que **un solo vehículo** pase por el **punto estrecho**, cuántos necesitamos y qué primitivas usamos?
 - **Semáforo Mutex, 1** (UN), uno para el **punto estrecho**
 - SemaphoreHandle_t **xSemaphoreCreateMutex(void)**
 - <https://www.freertos.org/CreateMutex.html>
 - **xSemaphoreGive**(SemaphoreHandle_t xSemaphore)
 - <https://www.freertos.org/a00123.html>
 - **xSemaphoreTake**(SemaphoreHandle_t xSemaphore, TickType_t xTicksToWait)
 - <https://www.freertos.org/a00122.html>



Implementación

- ¿En qué partes del programa usamos c/u de las **primitivas** y **porqué**?
 - SemaphoreHandle_t **xSemaphoreCreateMutex**(void)
 - **app_init**(void)
 - **xSemaphoreGive**(SemaphoreHandle_t xSemaphore)
 - void **task_a**(void *parameters);
 - void **task_b**(void *parameters);
 - **xSemaphoreTake**(SemaphoreHandle_t xSemaphore, TickType_t xTicksToWait)
 - Idem anterior



Implementación

Example usage:

```
SemaphoreHandle_t xSemaphore;  
  
void vATask( void * pvParameters )  
{  
    /* Create a mutex type semaphore. */  
    xSemaphore = xSemaphoreCreateMutex();  
  
    if( xSemaphore != NULL )  
    {  
        /* The semaphore was created successfully and  
        can be used. */  
    }  
}
```

Implementación

