Sistemas Empotrados y Ubicuos - Appunti

Francesco Lorenzoni

Febrero 2025

Contents

Part I

Introduction to SEU

Chapter 1

Introduction

Es importante utilizar el poliformat. La idea del proyecto es realizar un sensor con un Cortex AM4 sobra una placa de desarollo con algunos sensor de temperatura y humedad (y otras cosas). Entonces es un proyecto de IoT.

Estudiaremos cómo tratar las entradas analógicas y digitales, cómo organizar los datos y procerarlos, cómo enviarlos a un servidor, cómo recibirlos y visualizarlos.

La teoria si tiene el viernes online sincrona.

Se programa en C.

FreeRTOS no tiene procesos, tiene ilos (£Tasks? £Es lo mismo de MCPS?).

El objectivo es hacer el proyecto.

El proyecto es diviso en 5 partes y 3 entragables:

1. Entragable I - Práctica 1,2,3 (20%) - 13/03/2025
2. Entragable II - Práctica 4,5 (20%) - 10/04/2025
3. Entragable III - Proyecto Final (60%) - 05/06/2025

La fecha de examen de teoría es 06/06/2025.

F411RE es la placa de desarollo.

Chapter 2

Conceptos Generales

2.1 Sistemas Empotrados y Ubicuos

Ubicuo significa que está en todas partes, y empotrado significa que está dentro de algo; entonces un sistema empotrado y ubicuo es un sistema que está en todas partes y dentro de algo.

Tal vez se puede utilizar como sinonimo de empotrado la palabra “embebido”.

La computación ubicua implica que cada persona interactúa con cientos de dispositivos sin ser consciente. Esto se opone al uso de un ordenador personal, donde el usuario es consciente de que está utilizando directamente un ordenador.

2.1.1 Computación en la nube

En la nube hay dispositivos fisicos y virtuales también. L’IoT utiliza la nube para almacenar y procesar datos, porque tipicamente los dispositivos IoT no tienen suficiente capacidad de procesamiento. Una de las cosas más importantes de los IoT es la eficiencia energética, porque los dispositivos IoT están alimentados por baterías, entonces es importante que los dispositivos IoT consuman poca energía.

Chapter 3

FreeRTOS - Tasks

3.1 Crear una Tarea

Listing 3.1: Definir la funcion de tarea

```
void vATaskFunction( void *pvParameters )
{
    for( ;; )
    {
        -- Task application code here. --
    }

    /* Tasks must not attempt to return from their implementing
    function or otherwise exit.  In newer FreeRTOS port
    attempting to do so will result in an configASSERT() being
    called if it is defined.  If it is necessary for a task to
    exit then have the task call vTaskDelete( NULL ) to ensure
    its exit is clean. */
    vTaskDelete( NULL );
}
```

`xTaskCreate()` es el componente fundamental de un sistema multitarea. Probablemente la más compleja de todas las funciones del API.

```
portBASE_TYPE  xTaskCreate(
   pdTASK_CODE           pvTaskCode,
   const signed char     *const pcName,
   unsigned short        usStackDepth,
   void                * pvParameters,
   unsigned portBASE_TYPE uxPriority,
   xTaskHandle           *pxCreatedTask
   );
```

- ◇ `pvTaskCode`
  Puntero a la función que implementa la tarea (simplemente el nombre de la función)
- ◇ `pcName`
  Un nombre descriptivo para la tarea. FreeRTOS no lo usa, pero ayuda a la depuración.
- ◇ `uStackDepth` es el tamaño de la pila en numero de palabras, no en bytes. Por ejemplo, la pila de un Cortex-M3 tiene un ancho de palabra de 32-bits. `pvParameters` El valor asignado a pvParameters será el valor pasado a la función de tarea a través del parámetro con el mismo nombre `uxPriority` Define la prioridad con la que se ejecutará la tarea. Las prioridades pueden asignarse desde 0, (la menor prioridad), a (`configMAX_PRIORITIES-1`), que es la mayor. Varias tareas pueden tener la misma prioridad `pxCreatedTask` Devuelve un identificador para la tarea creada, para poder hacer referencia a ella en futuras llamadas del API P.ej., cambiar la prioridad o eliminar la tarea Se puede pasar NULL si no se va a usar Valor de retorno, dos posibilidades: pdTRUE : la tarea se ha creado con éxito. `errCOULD_NOT_ALLOCATE_REQUIRED_MEMORY` tarea no creada por insuficiente memoria dinámica (heap) para asignar las estructuras de datos y pila necesarios.

3.2