**1.1**

**Introducción a biblioteca sAPI:**

Comenzar a programar en la placa EDU-CIAA puede tornarse complicado en un principio, ya que si no se está familiarizado con RTOS, cargar, compilar y depurar un programa en osek-os resultara complejo al comienzo.

Para facilitar el manejo de la placa EDU-CIAA a aquellas personas que la utilizaran por primera vez, se puede hacer uso de una librería llamada sAPI, que los adentrara de una forma más amigable a la programación de la placa.

Esta biblioteca implementa una API simple para la programación de microcontroladores.

Una API es una interfaz de programación de aplicaciones, es el conjunto de [subrutinas](https://es.wikipedia.org/wiki/Subrutina), funciones y procedimientos (o [métodos](https://es.wikipedia.org/wiki/M%C3%A9todo_(inform%C3%A1tica)), en la [programación orientada a objetos](https://es.wikipedia.org/wiki/Programaci%C3%B3n_orientada_a_objetos)) que ofrece cierta [biblioteca](https://es.wikipedia.org/wiki/Biblioteca_(programaci%C3%B3n)) para ser utilizado por otro [software](https://es.wikipedia.org/wiki/Software) como una capa de abstracción en la [programación](https://es.wikipedia.org/wiki/Programaci%C3%B3n), generalmente (aunque no necesariamente) entre los niveles o capas inferiores y los superiores del software. Son usadas generalmente en las [bibliotecas](https://es.wikipedia.org/wiki/Biblioteca_(inform%C3%A1tica)) de programación.

La motivación para el desarrollo de la biblioteca sAPI surge de la necesidad de manejar los periféricos directamente desde una máquina virtual de Java para el desarrollo de Java sobre la CIAA y corresponde a la parte de bajo nivel de las clases de periféricos en Java que básicamente bindea a funciones escritas en C.

Luego se extendió la misma para facilitar el uso de la EDU-CIAA-NXP a personas no expertas en la arquitectura del LPC4337 facilitando el uso de esta plataforma.

La idea es tener periféricos abstractos y lo más genéricos posibles. Que sea bien independiente de la arquitectura y en lo posible que las funciones sean todas del tipo:

* moduloConfig();
* moduloRead();
* moduloWrite();

Utiliza nombres sencillos al estilo de la biblioteca Wiring, pero en lugar del concepto de pin utiliza el concepto periférico, para lograr una API unificada, sin importar el número de pines que tenga un cierto periférico.

**Módulos incluidos**

* Tipos de datos.
* Mapa de periféricos.
* Plataforma.
* Tick.
* Retardo.
* E/S Digital.
* E/S Analógica.
* Uart.

**Plataformas**

Actualmente disponible para las plataformas:

* EDU-CIAA-NXP (microcontrolador NXP LPC4337).
* CIAA-NXP (microcontrolador NXP LPC4337).

**Capas de software:**

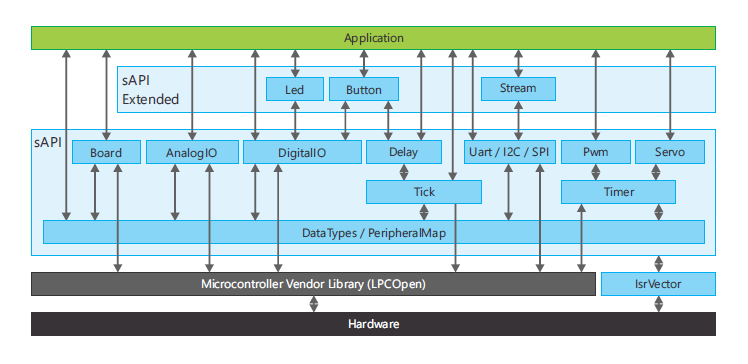


Figura 1.1: Modelo en capas y módulos.