quick2wire-ingeniería Documentation

Publicación 1

Diego Martín

09 de July de 2015

Índice general

quick2wire-ingeniería

En el presente apartado se recogen los diferentes aspectos de análisis y diseño de la biblioteca

El presente producto **software** no realiza una interacción directa con un usuario humano, sino que en su lugar es integrada en otras aplicaciones que aprovechan su funcionalidad. Esta particular característica requiere la realización de una etapa de análisis atípica.

CAPÍTULO 1

Dominio del problema

Actualmente la mayoría de *APIs* que posibilitan la manipulación del puerto GPIO presente en la Raspberry Pi requieren que el programa que aprovecha esta funcinalidad disponga de privilegios de superusuario. Esta restricción compromete la seguridad en el sistema a crear, que contará con un número de usuarios muy elevado y cuyas interacciones con el sistema no están controladas. Permitir el acceso al rol de superusuario a estos individuos supone un riesgo inaceptable (ejemplos son un error en un ejecutable creado por un usuario, que podría dañar el sistema de forma significativa o la posibilidad, si bien reducida, de un ataque intencionado).

Se han analizado diferentes alternativas de terceros que satisfagan los requisitos de la biblioteca, entre las que figuran RPi.GPIO y WiringPi, entre otras. La única alternativa viable es la biblioteca de quick2wire. Sin embargo, esta biblioteca únicamente cuenta con una versión en Python. La mayoría de las aplicaciones que aprovecharán el puerto GPIO serán programadas en C o C++, por lo que la integración del código Python no se considera una alternativa eficaz.

CAPÍTULO 2

Objetivos

2.1 OBJ1: Control de los pines GPIO por usuarios no privilegiados

- **Versión**: 1
- **Autores**: Requisito marcado por la implementación original
- **Fuentes**: Implementación original
- **Descripción**: La biblioteca y el resto de utilidades deberán proveer acceso a este dispositivo de forma que el acceso no exiga contar con privilegios de administración del sistema (usuario *root* en sistemas *NIX).
- **Importancia**: Muy alta
- **Urgencia**: Alta
- **Estado**: Alta
- **Estabilidad**: Estable

2.2 OBJ2 Compatibilidad con C y C++

- **Versión**: 1
- **Autores**: Diego Martín
- **Fuentes**: Fase de análisis preliminar
- **Descripción**: La biblioteca deberá ser integrable en código fuente escrito en C, utilizando para ello puntos de acceso a la biblioteca que no dependan de C++.
- **Importancia**: Media
- **Urgencia**: Media
- **Estado**: Completo
- **Estabilidad**: Estable
- **Comentarios**

2.3 OBJ3 Gestión de permisos y propiedad

- **Versión**: 1.5
- **Autores**: Requisito marcado por la implementación original
- **Fuentes**: Implementación original
- **Descripción**: Una vez que un usuario controle un pin la biblioteca debe garantizar la exclusividad de uso del mismo.
- **Importancia**: Muy alta
- **Urgencia**: Alta
- **Estado**: Alta
- **Estabilidad**: Estable
- **Comentarios**

2.4 OBJ4 Independencia del *hardware*

- **Versión**: 1.5
- **Autores**: Requisito marcado por la implementación original
- **Fuentes**: Implementación original
- **Descripción**: La biblioteca debe ser compatible con todas las versiones de la placa **Raspberry Pi** , detectando el modelo sobre el que la biblioteca se encuentra en tiempo de ejecución sin intervención del programador.
- **Importancia**: Alta
- **Urgencia**: Alta
- **Estado**: Completo
- **Estabilidad**: Estable
- **Comentarios**

CAPÍTULO 3

Actores

El sistema no interactúa con ningún usuario humano de forma directa, por lo que se reduce la importancia de las entidades humanas en la identificación de actores.

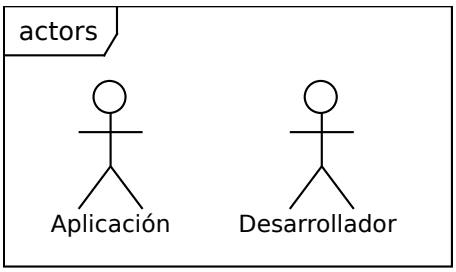
3.1 Actor 1: Aplicación

- **Versión**: 1
- **Autores**: Diego Martín
- **Fuentes**: Implementación original
- **Descripción**: Utiliza la API para manipular el GPIO según el comportamiento que el programador haya especificado.

3.2 Actor 2: Desarrollador

- **Versión**: 1
- **Autores**: Diego Martín
- **Fuentes**: Implementación original
- **Descripción**: Los desarrolladores de aplicaciones interactuarán con la API de forma indirecta y con la herramienta `gpio-admin` de forma directa.

3.3 Diagrama de actores del sistema



CAPÍTULO 4

Requisitos de información

4.1 IRQ1 Información sobre los usuarios

- **Versión**: 1
- **Autores**: Requisito marcado por la implementación original
- **Fuentes**: Implementación original
- **Objetivos asociados**: OBJ-1, OBJ-3
- **Requisitos asociados**: RF-2, RF-3, RF-4, RF-5, RF-6, RF-7, RF-8
- **Descripción**: La utilidad verifica a través de los diferentes mecanismos de gestión de permisos del sistema los privilegios de cada usuario.
- **Datos específicos**: UID, grupos de usuario
- **Tiempo de vida**: Tiempo de reserva de un Pin
- **Ocurrencias simultáneas**: Tantas como pines se reserven concurrentemente
- **Importancia**: Alta
- **Urgencia**: Alta
- **Estado**: Completo
- **Estabilidad**: Estable

CAPÍTULO 5

Requisitos funcionales

5.1 RF-1: Comunicación con la biblioteca

- **Versión**: 1
- **Autores**: Requisito marcado por la implementación original
- **Fuentes**: Implementación original
- **Objetivos asociados**: OBJ-1, OBJ-3
- **Requisitos asociados**: NFR-2
- **Descripción**: Siguiendo el mismo modelo de trabajo de la implementación original, la realización de las diferentes operaciones se realizará mediante llamadas al comando `gpio-admin`.
- **Precondición**
- **Secuencia normal**:
  1. La implementación de la biblioteca solicita la realización de una tarea al comando, especificando los parámetros de la aplicación a través de los parámetros de entrada propios de una llamada al sistema (recogidos en el array `argv`).
  2. La biblioteca determina la identidad del usuario mediante el UID del proceso.
  3. Se realiza la operación solicitada y emite un mensaje de confirmación o fallo.
- **Poscondición**: La operación es realizada.
- **Excepciones**:
  - En el caso de que la biblioteca retorne un error, comienza el caso de uso **RF-7**
- **Rendimiento**: Alto
- **Frecuencia**: Alta
- **Importancia**: Media
- **Urgencia**: Alta
- **Estado**: Completo
- **Estabilidad**: Estable

5.2 RF-2: "Registro" de un pin

- **Versión**: 1
- **Autores**: Requisito marcado por la implementación original
- **Fuentes**: Implementación original
- **Objetivos asociados**: OBJ-3
- **Requisitos asociados**: IRQ-1
- **Descripción**: La biblioteca considera que un pin debe ser "propiedad" de un usuario antes de poder utilizar el mismo, y a petición de un usuario verificará que el pin solicitado está libre u ocupado, reservándolo en el primer caso.
- **Precondición**
- **Secuencia normal**:
  1. La aplicación solicita el registro de un pin para su uso a la biblioteca. Comienza el caso de uso RF-1.
  2. Se analiza el código de retorno de la llamada al sistema. En caso de que el pin se encuentre reservado, el caso de uso finaliza y el usuario es notificado. En caso de que el pin no se encuentre reservado, la biblioteca indica que el usuario solicitante es el nuevo propietario.
  3. La biblioteca notifica al usuario de que la reserva ha sido satisfactoria.
- **Poscondición**: El usuario obtiene acceso al pin.
- **Excepciones**: No se plantean excepciones
- **Rendimiento**: Alto
- **Frecuencia**: Generalmente un usuario realizará una reserva en cada ejecución.
- **Importancia**: Alta
- **Urgencia**: Alta
- **Estado**: Completo
- **Estabilidad**: Estable

5.3 RF-3: "Liberación" de un pin

- **Versión**: 1
- **Autores**: Requisito marcado por la implementación original
- **Fuentes**: Implementación original
- **Objetivos asociados**: OBJ-1, OBJ-3
- **Requisitos asociados**: IRQ-1
- **Descripción**: De la misma forma que un pin es reservado debe ser liberado al finalizar la ejecución, de forma que posteriores usuarios puedan utilizar el mismo.
- **Precondición**: El pin debe estar reservado.
- **Secuencia normal**:
  1. La aplicación solicita la liberación del pin al comando `gpio-admin`. Comienza el caso de uso *RF-1*

2. La aplicación es notificado del resultado de la operación.

- **Poscondición**: El pin es liberado.
- **Excepciones**: En caso de que se solicite la liberación de un pin no reservado, se notifica a la aplicación y el caso de uso finaliza.
- **Rendimiento**: Alto
- **Frecuencia**: Media
- **Importancia**: Alta
- **Urgencia**: Alta
- **Estado**: Completo
- **Estabilidad**: Estable

5.4 RF-4: Modificación de la "dirección" del pin

- **Versión**: 1
- **Autores**: Requisito marcado por la implementación original
- **Fuentes**: Implementación original
- **Objetivos asociados**: OBJ-1, OBJ-3
- **Requisitos asociados**: IRQ-1
- **Descripción**: La dirección del pin indica si se utilizará para recibir o enviar información. Todos los pines pueden operar en ambos modos.
- **Precondición**: El pin debe estar reservado previamente.
- **Secuencia normal**:

  1. La aplicación solicita modificar la dirección del pin al comando `gpio-admin`. Comienza el caso de uso *RF-1*.

- **Poscondición**: La dirección del pin es modificada.
- **Excepciones**: En caso de que se solicite la modificación de un pin no reservado, se notifica a la aplicación y el caso de uso finaliza.
- **Rendimiento**
- **Frecuencia**: Se estima que durante cada ejecución se modificará la dirección del pin únicamente durante la fase de inicialización del mismo.
- **Importancia**: Alta
- **Urgencia**: Alta
- **Estado**: Completo
- **Estabilidad**: Estable

5.5 RF-5: Modificación del valor de un pin

- **Versión**: 1

- **Autores**: Requisito marcado por la implementación original
- **Fuentes**: Implementación original
- **Objetivos asociados**: OBJ-1, OBJ-3
- **Requisitos asociados**: IRQ-1
- **Descripción**: En caso de que la dirección del pin sea de salida, el valor del pin podrá ser modificado (los valores de tensión admitidos con 0 y 5 voltios).
- **Precondición**: El pin debe estar reservado previamente.
- **Secuencia normal**:
  1. La aplicación solicita modificar el valor del pin al comando `gpio-admin`. Comienza el caso de uso *RF-1*.
- **Poscondición**: El valor del pin es modificado.
- **Excepciones**: En caso de que se solicite la modificación de un pin no reservado, se notifica a la aplicación y el caso de uso finaliza.
- **Rendimiento**:
- **Frecuencia**: El número de modificaciones que se realizarán en un programa se estima muy elevado, por lo que el rendimiento deberá ser alto.
- **Importancia**: Alta
- **Urgencia**: Alta
- **Estado**: Completo
- **Estabilidad**: Estable

5.6 RF-6: Lectura del valor de un pin

- **Versión**: 1
- **Autores**: Requisito marcado por la implementación original
- **Fuentes**: Implementación original
- **Objetivos asociados**: OBJ-1, OBJ-3
- **Requisitos asociados**: IRQ-1
- **Descripción**: En caso de que la dirección del pin sea de entrada, el valor (tensión) del pin podrá ser consultado (los valores de trabajo son 0 y 5 voltios).
- **Precondición**: El pin debe estar reservado previamente.
- **Secuencia normal**:
  1. La aplicación solicita el valor del pin al comando `gpio-admin`. Comienza el caso de uso *RF-1*.
  2. Se retorna el valor del pin a la aplicación.
- **Poscondición**: La aplicación cuenta con el valor pin.
- **Excepciones**: En caso de que se solicite la lectura de un pin no reservado, se notifica a la aplicación y el caso de uso finaliza.
- **Rendimiento**:

- **Frecuencia**: El número de consultas que se realizarán en un programa se estima muy elevado, por lo que el rendimiento deberá ser alto.
- **Importancia**: Alta
- **Urgencia**: Alta
- **Estado**: No realizado
- **Estabilidad**: Estable

5.7 RF-7: Gestión de error

- **Versión**: 1
- **Autores**: Diego Martín
- **Fuentes**: Análisis preliminar
- **Objetivos asociados**: OBJ-1, OBJ-3
- **Requisitos asociados**: NFR-3
- **Descripción**: En el caso de que la biblioteca o la herramienta `gpio-admin` retornen un código de error no identificado, comienza este caso de uso.
- **Precondición**: Se debe dar un error en la biblioteca o en `gpio-admin`
- **Secuencia normal**:
  1. La biblioteca intenta identificar el error que se ha dado.
  2. En caso de poder determinarlo envía un mensaje descriptivo a la aplicación. En caso contrario utilizará un mensaje genérico.
- **Poscondición**: La aplicación es informada del error.
- **Excepciones**: No se plantean excepciones
- **Rendimiento**: Medio
- **Frecuencia**: Baja
- **Importancia**: Media
- **Urgencia**: Media
- **Estado**: Completo
- **Estabilidad**: Estable

5.8 RF-8: Interacción directa con `gpio-admin`

- **Versión**: 1
- **Autores**: Diego Martín
- **Fuentes**: Análisis preliminar
- **Objetivos asociados**: OBJ-1
- **Requisitos asociados**: IRQ-1

- **Descripción**: La herramienta `gpio-admin` puede ser utilizada de forma directa por el usuario, de la misma forma que la biblioteca interactúa con ella.
- **Precondición**: El usuario debe haber iniciado sesión en el sistema.
- **Secuencia normal**:
  1. El usuario ejecuta la herramienta, utilizando los parámetros de la línea de comandos para indicar la acción a realizar.
  2. La biblioteca ejecuta la acción y muestra los resultados de éxito o error.
- **Poscondición**: La operación a ejecutar se ha llevado a cabo.
- **Excepciones**: En caos de que se dé algún tipo de error, la biblioteca mostrará un mensaje informativo.
- **Rendimiento**: Alto
- **Frecuencia**: Alta
- **Importancia**: Alta
- **Urgencia**: Media
- **Estado**: Completo
- **Estabilidad**: Estable

5.9 Vista de casos de uso

5.9.1 Vista de paquetes de casos de uso

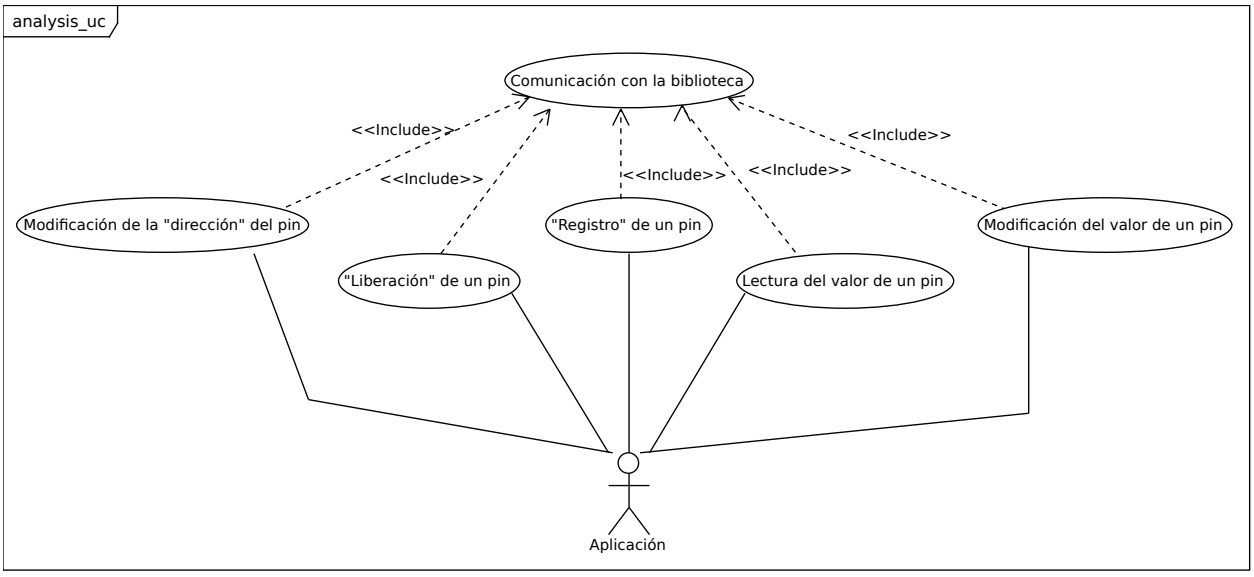


Figura 5.1: Vista del paquete de casos de uso. Se considera que todos los requisitos funcionales son realizados por el actor Aplicación (**ACT1**), que es implementado por el desarrollador.

Vista de paquetes de casos de uso

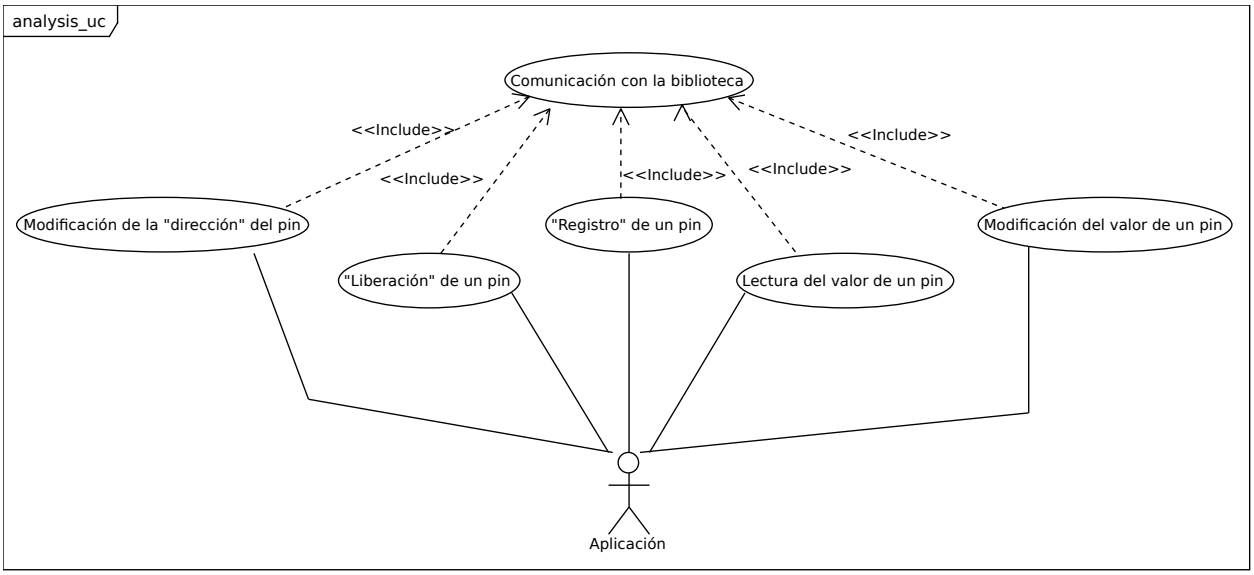


Figura 6.1: Vista del paquete de casos de uso. Se considera que todos los requisitos funcionales son realizados por el actor Aplicación (**ACT1**), que es implementado por el desarrollador.

CAPÍTULO 7

Requisitos no funcionales

7.1 NFR1: CMake

- **Versión**: 1
- **Autores**: Diego Martín
- **Fuentes**: Análisis preliminar
- **Objetivos asociados**: OBJ-2
- **Descripción**: Se utilizará CMake para la gestión de los diferentes procesos de compilación e instalación del *software*. Además se deberá crear un módulo de CMake que permita localizar la biblioteca creada al ser invocado.
- **Importancia**: Media
- **Urgencia**: Baja
- **Estado**: Completo
- **Estabilidad**: Alta

7.2 NFR2: Biblioteca compartida

- **Versión**: 1
- **Autores**: Diego Martín
- **Fuentes**: Análisis preliminar
- **Objetivos asociados**: OBJ-2
- **Descripción**: La inclusión de la biblioteca en el sistema debe realizarse en tiempo de ejecución mediante el su compilación como biblioteca compartida.
- **Importancia**: Baja
- **Urgencia**: Baja
- **Estado**: Completo
- **Estabilidad**: Alta

7.3 NFR3: Similitud

- **Versión**: 1
- **Autores**: Diego Martín
- **Fuentes**: Análisis preliminar
- **Descripción**: A fin de facilitar el trabajo con la API a desarrolladores conocedores de la versión en Python, se conservarán las convenciones de nombres y secuencias de operación que esta utiliza, sin que esto impida aprovechar características de C++ no presentes en Python que se consideren beneficiosas.
- **Importancia**: Media
- **Urgencia**: Media
- **Estado**: Completo
- **Estabilidad**: Estable

7.4 NFR4: Documentación

- **Versión**: 1
- **Autores**: Diego Martín
- **Fuentes**: Análisis preliminar
- **Descripción**: La API deberá incluir una documentación fácil de comprender por cualquier desarrollador con conocimientos de C++.
- **Importancia**: Media
- **Urgencia**: Media
- **Estado**: Completo
- **Estabilidad**: Estable

Vista estática del producto

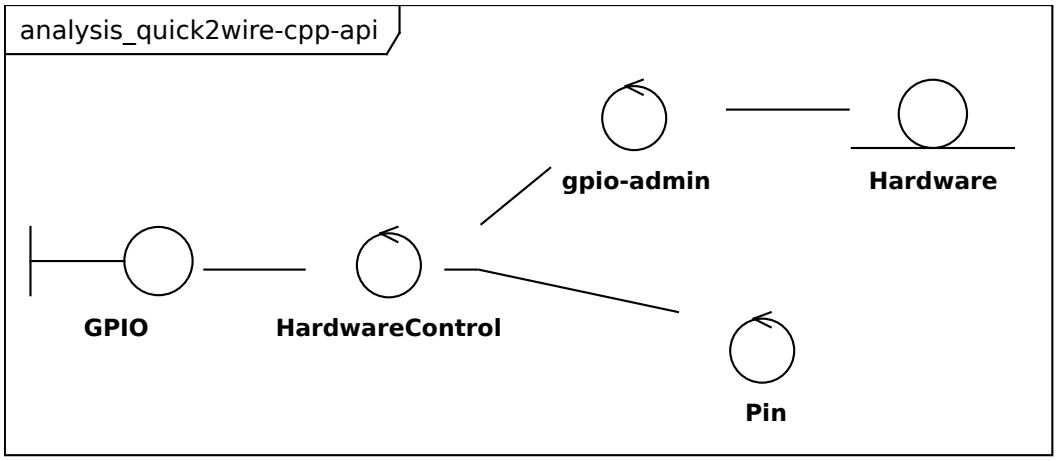


Figura 8.1: La vista estática recoge una clase de interfaz que será aprovechada por las diferentes aplicaciones. Dicha interfaz permitirá el acceso a una clase de control que creará los diferentes objetos de control pin y controlará la comunicación con el comando (interpretado como clase de control) `gpio-admin`, que será el encargado de realizar la comunicación con el *hardware*.

CAPÍTULO 9

Vista de interacción

9.1 Interacción con un pin

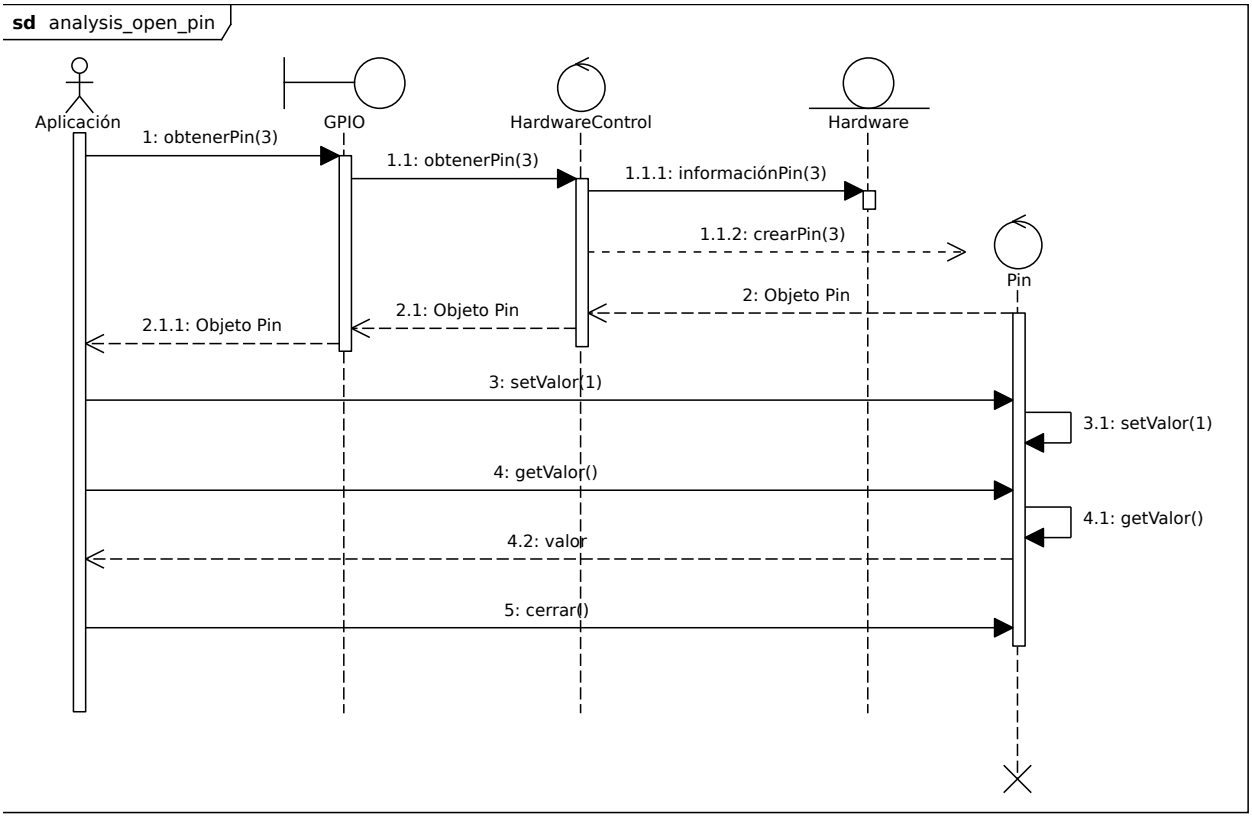


Figura 9.1: El diagrama representa un escenario completo de interacción con un pin, desde su reserva hasta la liberación de este.

Vista de máquina de estados

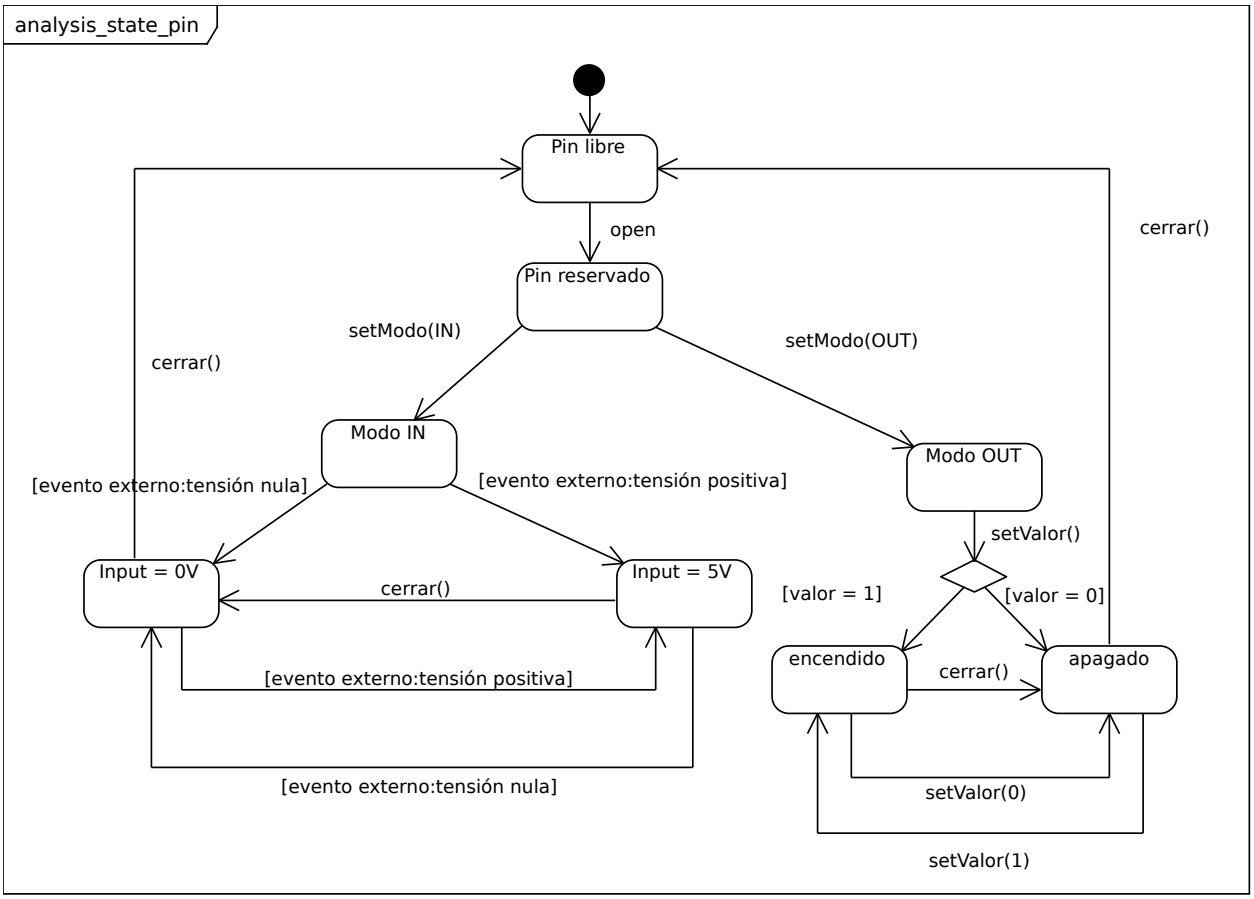


Figura 10.1: Representación de todos los estados en los que un pin puede encontrarse. En el caso del modo IN, los eventos que provocan un cambio de estado son provocados por el *hardware* al que el pin se encuentra conectado (por ejemplo, un sensor).

CAPÍTULO 11

Fase de diseño

11.1 Ámbito

La API se integrará con diferentes aplicaciones mediante su compilación como biblioteca compartida. Este archivo se almacenará en uno de los directorios de búsqueda de bibliotecas del sistema, del mismo modo que los archivos de cabecera serán situados en uno de los directorios de inclusión. La gestión de esta dependencia se ultima con la utilización de un módulo de búsqueda de CMake, por lo que se recomendará su uso (de forma opcional) a los diferentes usuarios.

Los usuarios con privilegios de acceso a la API serán aquellos miembros del grupo `gpio`, del cual son miembros todos los desarrolladores de la organización en la que la biblioteca se utilizará.

11.2 Diseño de datos

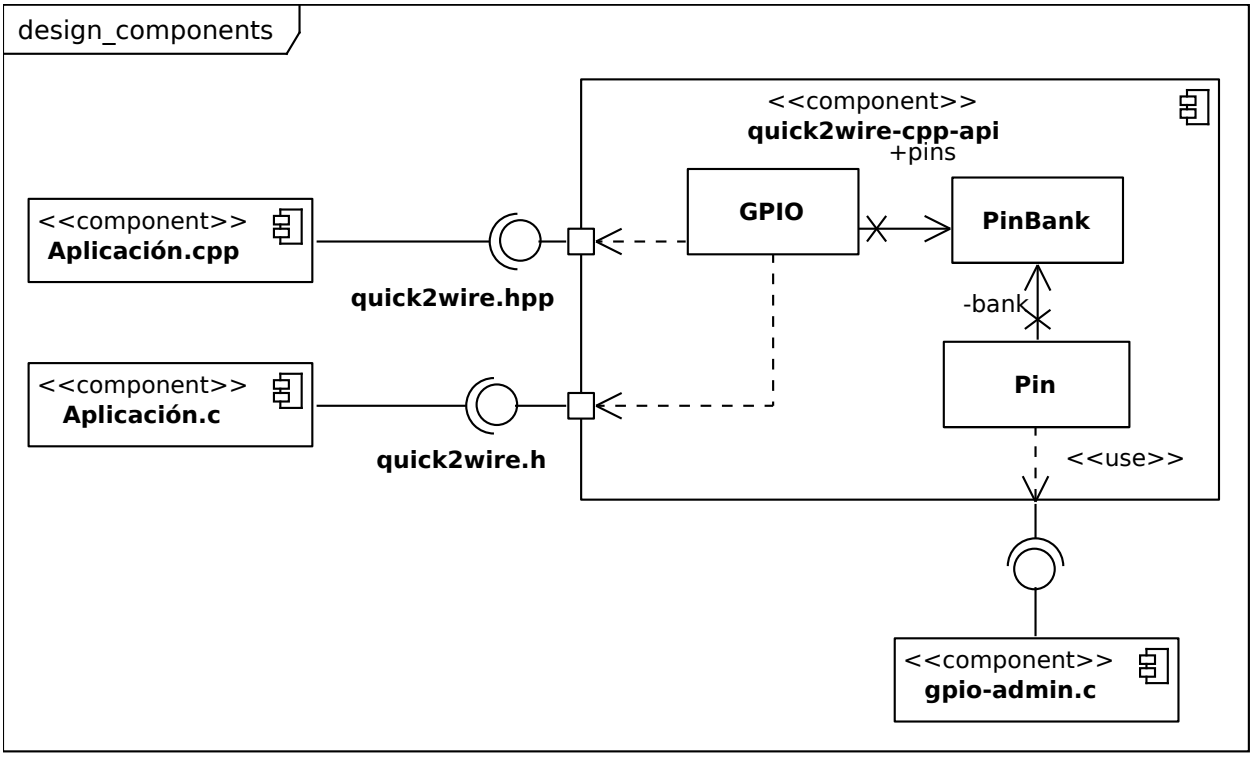
11.2.1 Objetos de datos

Únicamente serán gestionados los objetos pin como contenedores de información, que permitirán consultar la información del pin al que apuntan.

11.2.2 Estructuras de archivo

Se consultarán los grupos de los que un usuario es miembro, generalmente recogidos en el fichero `/etc/passwd`.

11.3 Diseño arquitectónico



11.4 Despliegue

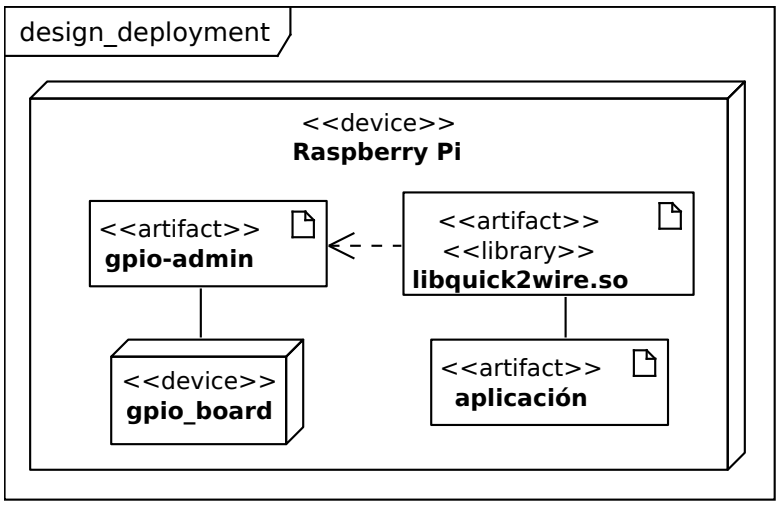


Figura 11.1: Representación de los diferentes componentes y sus interacciones. Nótese que el hardware GPIO se representa como componente del dispositivo Raspberry Pi

11.5 Vista estática del producto

11.5.1 API

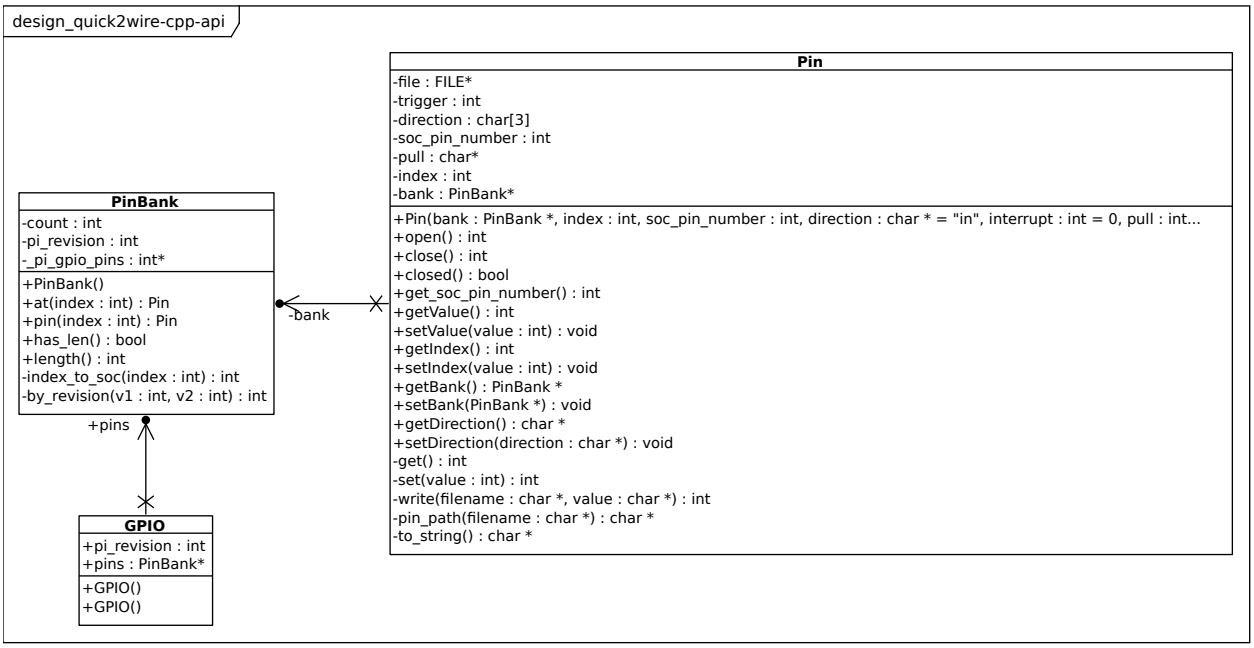


Figura 11.2: La API utiliza como punto de entrada la clase `GPIO`, que gestionará un objeto `PinBank`, el cual se encargará de la gestión de los diferentes pines.

11.5.2 gpio-admin

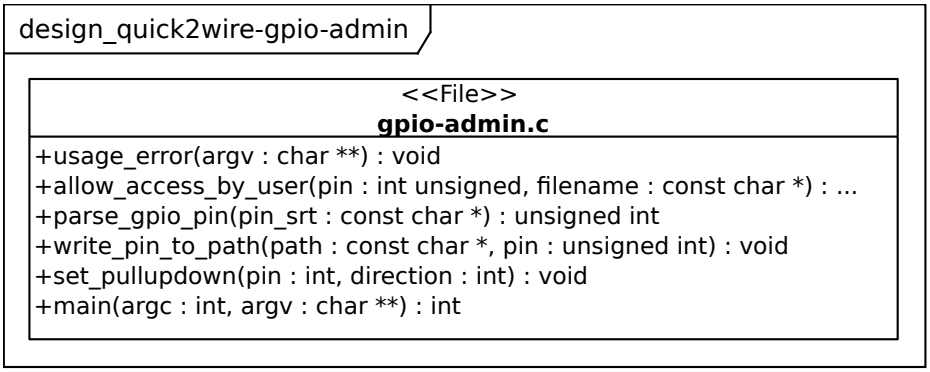


Figura 11.3: El ejecutable `gpio-admin` no se implementa siguiendo el paradigma de orientación a objetos. Se representa en el diagrama mediante una clase con el estereotipo `<<file>>`.

11.6 Diseño procedimental

En este apartado se referencian los algoritmos implementados en el sistema que son considerados de mayor relevancia.

Detección de la versión de la placa, para adaptar el número de pines ofrecidos al valor físico.

```
#include "board_revision.hpp"

#define CPUINFO_PATH "/proc/cpuinfo"

#define PI_VERSION_1 1
#define PI_VERSION_2 2
int revision(){
        std::ifstream fcpuinfo(CPUINFO_PATH);
        if(fcpuinfo.fail()){
                return 3;
        }
        std::string aux_str;
        std::string revision_str("Revision");
        while(std::getline(fcpuinfo, aux_str)){
                if(revision_str.compare(0, revision_str.length(), aux_str.substr(0, revision_str.leng
                        if((aux_str.compare(aux_str.length() - 2, aux_str.length() -1, "2") == 0) ||
                                return 1;
                        }
                        return 2;
                }
        }
        return 0;
}
```

Apertura de un pin mediante una llamada al comando `gpio-admin`.

```
int gpio_admin(char * subcommand, int pin, char* pull){

        char command[MAX_LEN];

        snprintf(command, MAX_LEN, "gpio-admin %s %d\n", subcommand, pin);//, pull == NULL ? "" : pu
        FILE* f = popen(command, "r");

        return pclose(f);
}
int Pin::open(){
         gpio_admin("export", this->soc_pin_number, this->pull);
         if(NULL == (this->file = fopen(this->pin_path("value"), "r+"))){
                 quick2wire_errno = PIN_ERR;
                 return 1;
        }

         return this->write("direction", this->direction);

        /*if self._direction == In:
        self._write("edge", self._interrupt if self._interrupt is not None else "none")
    */
}
```

Modificación del valor de un pin.

```
int Pin::set(int value){

        if(this->closed()){
                perror("The device is closed");
                return -1;
        }
        if(strcmp(OUT, "out") != 0){
```

```
                perror("The direction is not \"out\"");
                return -2;
        }

        fseek(this->file, 0, SEEK_SET);

        char buff[3];
        snprintf(buff, 3, "%d", value);
        fputs(buff, this->file);
        fflush(this->file);

        return value;
}
```

11.7 Pruebas

Se han realizado pruebas de concepto para el uso de la biblioteca (ver la sección "Examples" de la documentación técnica). Además de pruebas unitarias manuales y pruebas de integración con aplicaciones ya creadas o programadas explícitamente con el fin de probar la biblioteca, integrando la misma con la biblioteca MPI.

CAPÍTULO 12

Gestión del proyecto

La gestión del proyecto comienza en la iteración número 10 (issue #101 en Redmine), realizando una etapa de análisis previa para determinar la viabilidad de diferentes alternativas. Todas las tareas de desarrollo terminan en dicha iteración, añadiéndose varias tareas en la iteración número 13 (issue #148 en Redmine) relacionadas con nuevas características u operaciones de soporte.

CAPÍTULO 13

Indices and tables

- genindex
- modindex
- search