INSTITUTO TECNOLÓGICO DE CHIHUAHUA

DIVISIÓN DE ESTUDIOS DE POSGRADO E INVESTIGACIÓN

“AUTO-GENERADOR DE CÓDIGO PARA SISTEMAS EMBEBIDOS CON APLICACIONES RESIDENCIALES”

TESIS

PARA OBTENER EL GRADO DE

MAESTRO EN INGENIERÍA MECATRÓNICA

PRESENTA:

JOSÉ MIGUEL DÍAZ ARRIAGA

DIRECTORA DE LA TESIS:

MSC. CLAUDIA PRIETO RESÉNDIZ

CO-ASESOR:

MSEE. SALVADOR ALMANZA GARCÍA

CO-ASESOR:

DR. ROGELIO ENRIQUE BARAY ARANA

SEP
SECRETARÍA DE EDUCACIÓN PÚBLICA

TECNOLÓGICO NACIONAL DE MÉXICO

CHIHUAHUA, CHIH, MEXICO, JUNIO 2019

Agradecimiento a la institución

Le agradezco al Instituto Tecnológico de Chihuahua por ser una parte central en mi formación técnica, la cual me ha abierto muchas puertas en la vida.

Agradecimientos

Agradezco primero que nada a mi esposa Lavinia Rangel por su apoyo constante en todos mis proyectos. A mi familia por la ayuda y presión positiva. A Claudia Prieto y Salvador Almanza por su constante tutoría. Y finalmente a Resideo por permitir un espacio para la experimentación y formación de su personal.

Resumen

Los sistemas embebidos son prácticamente ubicuos en esta época y es difícil imaginar una vida sin ellos, especialmente dentro de nuestros hogares. Una alta demanda por todos estos aparatos ha creado una gran competencia para proveerlos al mejor precio en el tiempo indicado con las mejores características. La tendencia de que los productos residenciales tengan cada vez mas características avanzadas tiene como consecuencia de que el software embebido dentro de ellos se vuelva cada vez mas complejo, lo que lleva a tiempos de desarrollo y validación más largos. Mientras se diversifican los tipos de productos residenciales avanzados, se vuelve mas difícil el mantenimiento en campo de todos los distintos modelos que se van creando.

Para ayudar con esta problemática se está proponiendo un *framework* flexible para sistemas embebidos, el cual sea independiente de la arquitectura de microcontroladores o microprocesadores utilizados, junto a un auto-generador de código para la configuración amigable al usuario, sencilla y rápida. Esto permitirá tener una base de software común entre proyectos distintos, lo que acorta la curva de aprendizaje, el tiempo de desarrollo, y la validación de módulos.

En este documento se detalla la arquitectura del *framework* propuesto, y como logra una independencia de la arquitectura de hardware y compilador utilizado. Se muestra la arquitectura del auto-generador que configura el *framework*, y finalmente se presenta un prueba de concepto para demostrar las capacidades y flexibilidad del sistema trabajando en su totalidad.

Abstract

Embedded systems are practically ubiquitous in this day and age, and it's hard to imagine a life without them, especially inside our homes. A high demand for these devices has created a great competition to provide them at the best price at the right time, with the best features. The trend in which residential products have more and more advanced features has as consequence that the embedded software inside them must be more and more complex, which leads to longer development and testing times. As the types of advanced residential products keep diversifying, the field maintenance of all the different models that are being created grows more difficult.

To help with this problem a flexible framework for embedded systems is being proposed, which is microcontroller or microprocessor architecture independent, along with a code auto-generator for a user friendly, simple, and fast framework configuration. This will allow to have a common software base between different projects, which shortens the learning curve, development time, and module validation.

This document details the proposed framework architecture, and how it manages a used hardware architecture and compiler independence. The architecture of the auto-generator, which configures the framework is shown, and finally, a proof of concept test to demonstrate the capabilities and flexibility of the whole system working is presented.

Índice general

Índice de figuras

1 MARCO TEÓRICO

1.1 Antecedentes

1.1.1 Productos residenciales

En este documento se va a manejar el termino de producto residencial para los dispositivos electrónicos y electrodomésticos que se encuentran usualmente en un hogar y que no cumplen con tareas especializadas como de seguridad o del ámbito médico. Esto incluye aparatos como termostatos, lavadoras o televisiones. Se estima que el mercado completo de electrónica de consumo en el 2013 fue de más de 1 billón de dólares y que este numero aumentará a 1.5 billones para el 2022 [1].

1.1.2 Sistemas embebidos

Un sistema embebido se define como un sistema computacional aplicado distinto a otros tipos de computadoras como una computadora personal (*PC*) [2]. Algunas características típicas de los sistemas embebidos incluyen:

- Tienen un propósito específico, esta es la característica principal que los diferencia de las computadoras personales
- Cuentan con limitaciones en hardware y/o software
- Usualmente tienen requerimientos de confiabilidad mas estrictos que una computadora personal

Cabe mencionar que la definición de un sistema embebido no está fija y varía dependiendo de los autores.

El sistema de guía inercial de la nave espacial para alunizar de las misiones Apollo en la década de 1960 es comúnmente considerado como el primer sistema embebido [3]. Gracias a la invención de los microcontroladores y microprocesadores en la década de 1970 y su reducción en costos de producción los sistemas embebidos se volvieron lo suficientemente económicos como para ser incluidos en múltiples electrodomésticos. Hoy en día sus aplicaciones van desde complejos sistemas espaciales, militares y médicos hasta aplicaciones residenciales de confort y entretenimiento.

En 2008 se tenían aproximadamente 30 microprocesadores / microcontroladores por cada 1 persona en países desarrollados, y a partir de ahí ese numero ha ido aumentando entre 10 %y 20 % cada año [4]. Una de las aplicaciones destacables de los microcontroladores y microprocesadores en la actualidad son los automóviles, donde los modelos básicos contienen alrededor de 30 microprocesadores mientras que en los modelos de lujo este numero asciende hasta más de 100 [5].

1.1.3 Framework

En software un marco de trabajo conocido como *framework* por su nombre en inglés, como se le mencionará de aquí en adelante, es una plataforma donde se desarrollan aplicaciones de software [6]. Sirve como la base para la programación y puede contener clases y funciones predefinidas para interactuar con distintas partes del programa. Facilitan el desarrollo de software al proveer elementos predefinidos sobre los cuales trabajar [7]. Un ejemplo de *framework* para microcontroladores es el *“Advanced Software Framework (ASF)”* desarrollado por Microchip para sus familias de microcontroladores AVR y SAM [8].

1.1.4 Desarrollo de Software en Resideo

El centro de diseño de Honeywell Chihuahua nació en el 2006 con 2 ingenieros como parte del centro de producción "*Chihuahua Manufacturing Operation*" ubicado en la Ciudad de Chihuahua, Chihuahua. A finales de octubre del 2018 se separó de Honeywell la parte de la compañía que se dedica a productos de confort y seguridad residencial nombrándose Resideo [9]. El centro de diseño de Chihuahua pertenece a Resideo, siendo uno de los centros de ingeniería más importantes dentro de la compañía a nivel mundial.

Para mediados de 2019 cuenta con mas de 120 empleados en un edificio propio y se han desarrollado proyectos de ahorros con un valor de mas de 30 millones de dolares, así como el desarrollo de nuevos productos estrella actualmente en el mercado, como el termostato *T9 Smart Thermostat* [10].

Las actividades hasta la fecha incluyen proyectos de reducción de costo y soporte para productos ya existentes en campo. Los productos en los que se trabajan tienen una gran diversidad de de arquitecturas de hardware y software, desde microcontroladores de 4 bits para productos básicos hasta microprocesadores de 32 bits que manejan gráficos en pantallas táctiles a color y conexiones inalámbricas, como WiFi y Redlink [11]. La gran diversidad de productos lleva a que cada proyecto tenga una curva de aprendizaje fuerte ya que todos fueron desarrollados en distintos periodos de tiempo, por distintas personas en diversas partes del mundo, y gracias a que Honeywell fue adquiriendo distintas compañías en su historia, también existen productos que fueron desarrollados fuera de Honeywell/Resideo.

Actualmente está creciendo el desarrollo de productos nuevos para el mercado principalmente de Norteamérica y Europa, el cual requiere de desarrollo de software completo a diferencia de solo modificaciones a productos actuales. Uno de los

requisitos claves en del desarrollo de estos nuevos productos es el tiempo para poder ser colocado en el mercado.

1.1.5 Auto-generación de código

Un auto-generador de código es un programa que produce código en un lenguaje de manera automática a partir de un esquema expresado diferente [12]. Entre los generadores de código más comunes están los compiladores que transforman código en lenguaje de alto nivel a lenguajes de bajo nivel como ensamblador o incluso lenguaje maquina. En este documento se hará referencia a los generadores de código fuente, que operan en un lenguaje mas elevado que los compiladores. Estos generadores de código fuente permiten desarrollar software a partir de modelos preestablecidos y el código resultante es menos propenso a errores que si se escribiera manualmente [13].

Ventajas de utilizar un generador de código:

- Reducción del tiempo de desarrollo y prueba.
- Reducción de mantenimiento del código generado.
- Se elimina la necesidad de que el desarrollador modifique el *framework* ya definido.
- Estandarización de código entre productos.

Desventajas:

- La arquitectura y el *framework* del software a desarrollar deben de estar previamente bien definidos para poder ser útil, y una modificación para un proyecto específico puede afectar a otros proyectos.

- Se generan mas archivos y líneas de código que si se escribiera el código a mano.
- En ciertos casos se puede volver mas complejo el desarrollar alrededor de las limitaciones de un *framework* definido que no puede ser modificado.
- Un framework general puede dejar sin soporte a características específicas deseables de ciertos microcontroladores.

Entre los ejemplos de generadores de código fuente existentes en el mercado actual están el Aplillet de Renesas [14], el cual permite generar y configurar los controladores de periféricos a partir de una interfaz gráfica para una gama de microcontroladores de sus familias. Otro ejemplo es el Atmel Start de Microchip [15], el cual tiene su interfaz con el desarrollador a partir de una pagina web permitiendo siempre tener acceso a la versión mas actualizada del producto. Estos generadores están limitados en el soporte a microcontroladores de la misma compañía por razones obvias, y a veces también están diseñados para un entorno de desarrollo o compilador en específico.

El diferenciador principal de de este *framework* y auto-generador de código propuestos es la capacidad de soportar microcontroladores de distintos fabricantes y arquitecturas, siendo agnóstico al entorno de desarrollo o compilador que se esté usando.

1.1.6 Desarrollo de software embebido

En la actualidad los sistemas embebidos son una parte constante de nuestra vida diaria, desde electrodomésticos y sistemas de entretenimiento en nuestros hogares, hasta sistemas críticos de seguridad en la industria. El mercado global competitivo moderno favorece a los productos con mas características que llegan primero a los consumidores, por lo que un desarrollo rápido y confiable se vuelve

indispensable para una compañía. El tiempo típico de desarrollo de un producto nuevo puede tomar entre 45 y 75 semanas para productos de media y alta complejidad [16].

Actualmente dada la complejidad de los sistemas embebidos modernos, la mayor inversión en su diseño ha ido cambiando de hardware a software. Esto resalta la importancia de tener metodologías y herramientas que reduzcan el tiempo de desarrollo y mejoren la calidad en el producto. Una de estas herramientas es la de auto-generadores de código, estos permiten crear una base de desarrollo a través de una interfaz gráfica simple e intutiva.

Un *framework* de software para sistemas embebidos junto con un auto-generador de código que pueda configurarse de manera gráfica puede representar una gran ayuda al tiempo de desarrollo y pruebas, así como reducir la curva de aprendizaje para trabajar en distintos productos que hayan sido desarrollados con este *framework*. Otra de las ventajas es la potencial reducción de errores generados por escribir código de manera manual.

El objetivo principal de utilizar auto-generadores de código es mejorar la velocidad de desarrollo por medio de estandarización de arquitecturas de software en productos. Últimamente se ha observado un cambio de los esfuerzos de desarrollo de nuevos productos, en 2008 aproximadamente el 20 % de los ingenieros totales involucrados en desarrollo de un producto estaban en el desarrollo y prueba de software, para el 2013 este numero se incrementó a un 60 % [17]. De igual manera la importancia del software embebido en la calidad de los productos ha ido creciendo. Un ejemplo de esto son los marca-pasos, entre los años 1990 y 2000 en Estados Unidos se tuvieron reportes de aproximadamente 500,000 errores en estos dispositivos, de los cuales el 40 % estaban relacionados con el firmware [18].

2 HERRAMIENTAS DE DESARROLLO

A continuación se describen los parámetros de selección usados para elegir las herramientas de desarrollo usadas para generar el proyecto de tesis.

2.1 Lenguajes de programación

El primer paso para el desarrollo fue seleccionar el lenguaje de programación adecuado, para esto se tomaron los siguientes requerimientos:

1. Ser un lenguaje maduro con un grupo de usuarios grande, asegurando así un fácil acceso a documentación y soporte.
2. Tener soporte para manejo de archivos de texto plano.
3. Tener soporte para expresiones regulares.
4. Tener soporte para diseñar interfaces gráficas.
5. Ser orientado a objetos, ya que esto proporciona mayor facilidad para el manejo de archivos.
6. Poder ser compilado y que sus programas puedan ser ejecutados en un sistema operativo Windows (Al menos las versiones 7 y 10).
7. Como característica adicional no indispensable, poder ser compilado y que sus programas puedan ser ejecutados en un sistema operativo Linux (Al menos kernel 3.x en adelante).

Para el primer punto se tomaron como referencia la guía de lenguajes de programación más populares del 2017 según la revista *Spectrum* de IEEE [19], tomando en cuenta los lenguajes tipo *“enterprise”* y el índice de popularidad de lenguajes de programación *PYPL [20]*.

En base a los requerimientos establecidos se lograron seleccionar los siguientes lenguajes con sus respectivas características.

2.1.1 JAVA

JAVA fue creado en 1995 y se ha convertido en uno de los lenguajes de programación para PC y web más populares al poder ser ejecutados en múltiples plataformas sin la necesidad de ser compilado para cada una de ellas [21] gracias al uso de una máquina virtual de JAVA instalada en la plataforma. Es un lenguaje libre desarrollado inicialmente por la empresa Sun Microsystems y actualmente es propiedad de la empresa Oracle.

Características notables:

- Funcionamiento “Escribe una vez y ejecuta en todas partes”, esto significa que el mismo programa puede ser ejecutado en diversas plataformas sin necesidad de ser compilado para cada una de ellas.
- Sistemas operativos oficialmente soportados por Java 8 [22]:
  - Windows: Vista, 7, 8.x y 10
  - Linux: Ubuntu 14.x en adelante, RedHat 7.x en adelante
  - Mac OSX: 1.8 en adelante
- Soporte para expresiones regulares.
- Existen diversos *frameworks* disponibles para diseñar una interfaz gráfica, algunos de ellos se pueden integrar directamente en el entorno de desarrollo.

2.1.2 Python

Python fue creado en 1991, rápidamente se ha popularizado para desarrollar aplicaciones en múltiples plataformas. Es un lenguaje interpretado, por lo que el mismo código se ejecuta en distintas plataformas sin tener que compilar para cada una de ellas [23].

Características notables:

- El código de Python puede ser ejecutado en diferentes plataformas usando interpretadores.
- Sistemas operativos oficialmente soportados [24]:
  - Windows
  - Linux
  - Mac OSX
- Soporte para expresiones regulares.
- Existen diversos *frameworks* disponibles para diseñar interfaces gráficas, pero la mayoría son específicos a un sistema operativo.
- Recientemente se ha posicionado como uno de los lenguajes de programación más populares [19].

2.1.3 C#

C# fue creado por la empresa Microsoft en 2000 derivado del lenguaje C dentro del *framework* .NET [25].

Características notables:

- Limitado únicamente al sistema operativo Windows. Una implementación de código abierto (Mono) es compatible con Windows, Linux y Mac OSX, pero cuenta con un soporte más limitado [26].
- Soporte para expresiones regulares.
- Lenguaje moderno, cuenta con soporte de Microsoft, una compañía grande.
- Diseño de interfaz gráfica sencillo utilizando Visual Studio.

2.2 Entorno de desarrollo

Un Entorno de Desarrollo Integrado, o *IDE* (por sus siglas en inglés) como se le conocerá de aquí en adelante, es una aplicación de software que facilita el desarrollo de software. Usualmente consta de un editor de texto para el código, herramientas para facilitar el proceso de compilación y herramientas para depurar el software desarrollado [27].

Para seleccionar el entorno de desarrollo se tomaron en cuenta los siguientes requerimientos:

1. Ser suficientemente popular para tener una base de usuarios grande y contar con soporte.
2. Capacidad para crear interfaces gráficas de una manera sencilla.
3. Una licencia tipo FOSS (Software Libre y Gratuito, por sus siglas en inglés) es preferible para evitar una inversión monetaria.
4. Soporte para el sistema operativo Windows necesario, el soporte para Linux es deseado, pero no indispensable.

Para el primer punto se tomó como referencia el índice de popularidad de *IDEs* PYPL [20]. Teniendo estos requerimientos los *IDEs* seleccionados a evaluar fueron:

2.2.1 Eclipse

IDE de tipo software libre con soporte para múltiples lenguajes y plataformas, más comúnmente usado para desarrollar en Java. Existen múltiples complementos llamados *plug-ins* que aumentan sus características y capacidades para desarrollo [28].

La fundación Eclipse proporciona también un Kit de Desarrollo de Software, conocido como SDK por sus siglas en inglés, el cual permite tomar este *IDE* para diversas aplicaciones. Esto ha logrado que existan distintos *IDEs* para aplicaciones específicas basados en Eclipse [29].

Lenguajes relevantes soportados:

- Java
- Python
- C++

Características relevantes:

- Utiliza la licencia pública de Eclipse que lo identifica como software libre [30].
- Multiplataforma, soporta los siguientes sistemas operativos:
  - Windows
  - Linux
  - Mac OS
- Soporta la posibilidad de diseñar interfaces gráficas de manera gráfica utilizando WindowBuilder.

2.2.2 NetBeans

IDE creado por la empresa Oracle, quien desarrolla JAVA. Principalmente está enfocado para desarrollar aplicaciones en JAVA con interfaz gráfica [31]

Lenguajes relevantes soportados:

- Java
- C++

Características relevantes:

- Utiliza la licencia GPL v2, lo que lo identifica como software libre [32].
- Se ejecuta en Java, por lo que es compatible con todos los sistemas operativos soportados por ese lenguaje.
- Incluye la capacidad para diseñar interfaces gráficas de manera gráfica para Java.

2.2.3 Visual Studio

Lenguajes relevantes soportados:

- Python
- C#
- Visual Basic .NET

Características relevantes:

- Utiliza una licencia propietaria, ver abajo para más detalles.
- Solo es soportado en Windows

- Incluye la capacidad para diseñar interfaces gráficas de manera gráfica para C#.

Este es software propietario, existen distintas versiones del *IDE* incluyendo una versión *freeware* "*Community edition*" que restringe su uso para desarrollar aplicaciones de código abierto o de uso no comercial cuando se trabaja dentro de una empresa [33]. Ya que este proyecto está en parte apoyado por la empresa Resideo, según los términos y condiciones se requeriría adquirir una licencia comercial, cuyo precio comienza en $1,119.00 USD por año [34].

2.3 Control de versión

Contar con un sistema de control de versión es indispensable para cualquier trabajo complejo. Permite tener un respaldo remoto del trabajo, así como llevar un registro de los cambios que se hacen. Esto permite también un trabajo colaborativo de forma más sencilla [35].

Además de un sistema de control de versión fue necesario seleccionar un servicio que provea un servidor remoto en el cual resida el repositorio.

2.3.1 Subversión

Sistema de control de versión creado por Apache Foundation, utiliza una licencia de software libre [36].

Servicios de servidores:

- RiouxSVN [37]
- Perforce [38]
- XP-dev [39]

2.3.2 Git

Sistema de control de versión moderno creado por Linus Torvalds, utiliza un licencia de software libre [40].

Servicios de servidores:

- GitHub[41]
- BitBucket [42]

2.4 Formatos para archivos de configuración

Se identificaron 3 distintos archivos de configuración necesarios para el auto-generador. Estos se describirán a detalle en la sección 4.2, en resumen estas son las necesidades:

1. Información del microcontrolador. Incluye el número de pins y sus características. Necesita ser un archivo externo al auto-generador para evitar tener que generar un nuevo programa por cada microcontrolador que se agregue.
2. Información del proyecto. Incluye las ligas a los otros archivos de configuración así como los datos del proyecto, como el nombre y el microcontrolador seleccionado.
3. Configuración de microcontrolador. Incluye la configuración seleccionada de los diferentes periféricos del microcontrolador.

Se identificaron las siguientes características como deseables para el formato de los archivos de configuración:

1. Formato de texto plano, para poder ser revisado o modificado en cualquier editor de texto.

2. Formato estándar para asegurar documentación y soporte.

Basado en estas características se identificaron los siguientes formatos.

2.4.1 XML

Formato creado en 1996, significa "*Extensible Markup Language*", y su énfasis principal es el de ser simple y general [43].

Características notables:

- Similar a HTML, usa símbolos de markup para describir los contenidos de la página o archivos [44].
- Puede ser desplegado como HTML en navegadores de Internet para una fácil visualización [45].
- Una de las metas de diseño de XML incluyen "Será fácil escribir programas que procesen documentos XML" [46].

2.4.2 JSON

Formato creado en 1999, significa "*JavaScript Object Notation*", y su énfasis principal es el de ser un formato de datos independiente de el lenguaje de programación utilizado [47].

Características notables:

- Más fácil de escribir que XML.
- Un archivo equivalente de XML sería más grande.
- JSON puede ser analizado por una función estándar de JavaScript.
- Se utiliza principalmente para transmitir datos entre un servidor y aplicaciones web, como una alternativa a XML [48].

2.4.3 YAML

Formato creado en 2001, significa "*YAML Ain't Markup Language*", y su énfasis principal es el de poder ser leído fácilmente por un humano [49].

Características notables:

- Es más sencillo de leer que XML o JSON, pero más difícil de generar y analizar por software [50].

2.5 Selección de herramientas

Después del análisis, se decidió por utilizar las siguientes herramientas:

Lenguaje de programación: JAVA. Los factores principales que marcaron la decisión fueron el costo nulo, amplio soporte, portabilidad, y la capacidad de desarrollar una interfaz gráfica multi-plataforma de manera nativa. Python fue una alternativa muy cercana, siendo descartado al final por no tener un editor de interfaz gráfica sencillo multi-plataforma.

Entorno de desarrollo: Eclipse. El amplio uso de Eclipse como IDE para múltiples lenguajes hace la curva de aprendizaje baja, junto con la capacidad de ser ejecutado en diversos sistemas operativos, colocó a Eclipse muy por encima de los demás IDEs.

Control de versión: Git. Se optó por la opción más popular [51] por su amplio soporte. Se Utilizó un servidor privado de Bitbucket [42] principalmente porque a la fecha de inicio del proyecto era la opción más robusta que ofrecía repositorios gratuitos privados.

Formatos para archivos de configuración: XML. Se decidió usar la opción más usada en programas comerciales por la familiaridad de los desarrolladores.

3 FRAMEWORK

Para poder llegar al objetivo de tener un código de aplicación que sea transportable entre diferentes arquitecturas de microcontroladores, es necesario tener una base a la cual la aplicación pueda usar sin necesidad de saber en que microcontrolador está trabajando. A continuación se describe el *framework* para sistemas embebidos que se está proponiendo.

3.1 Arquitectura general

La Figura 3.1 muestra la arquitectura general de un programa genérico utilizando el *framework* propuesto, el cual está compuesto por los elementos dentro del cuadro con línea punteada. A continuación se describen los elementos mostrados.

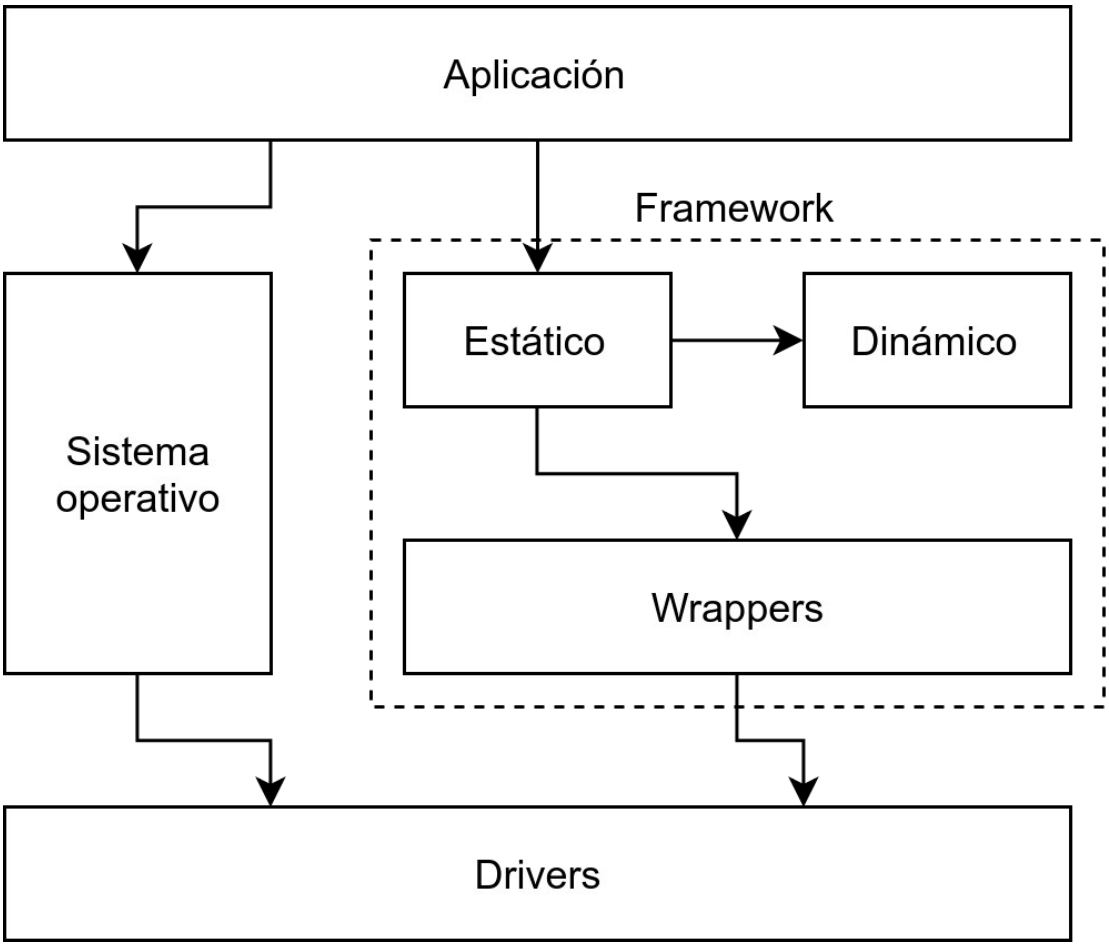

Figura 3.1 Arquitectura de programa usando el framework propuesto

3.1.1 Aplicación

Para fines de simplificación, se muestra la aplicación como un solo elemento, pero este puede constar de múltiples elementos, como capas de abstracción, o los distintos módulos necesarios para la aplicación que no hagan uso directo del hardware. Usualmente en productos de complejidad media o alta, la aplicación es por mucho la parte del software donde se concentra mayor complejidad, al contener el comportamiento del dispositivo.

El objetivo principal es que la aplicación no sea dependiente de un microcontrolador específico. Al no hacer uso directo de los periféricos del microcontrolador, la aplicación puede ser transferida a otro microcontrolador sin cambios.

3.1.2 Sistema operativo

El sistema operativo se muestra en el diagrama, pero es totalmente opcional, dependiendo de la aplicación. El sistema operativo puede hacer uso directo de los periféricos del microcontrolador, por lo que su configuración es específica para cada microcontrolador.

El sistema operativo puede ser desde un programador de tareas simple hecho a la medida para el proyecto, hasta un sistema operativo en tiempo real complejo, como FreeRTOS [52].

3.1.3 Controladores

Los controladores (*drivers*) son específicos para cada periférico del microcontrolador, por lo que no son compatibles entre 2 microcontroladores de diferentes fabricantes o familias.

3.1.4 Framework

En la sección 3.2 se describen con detalle los elementos que forman parte del *framework* propuesto.

3.2 Arquitectura de Framework

La Figura 3.2 muestra la arquitectura genérica de un módulo del *framework*, por módulo se entiende el control de un periférico específico del microcontrolador, como

GPIO [53], ADC, SPI, etc. Para mostrar ejemplos, se usará el módulo de GPIO al ser el más común y uno de los más simples.

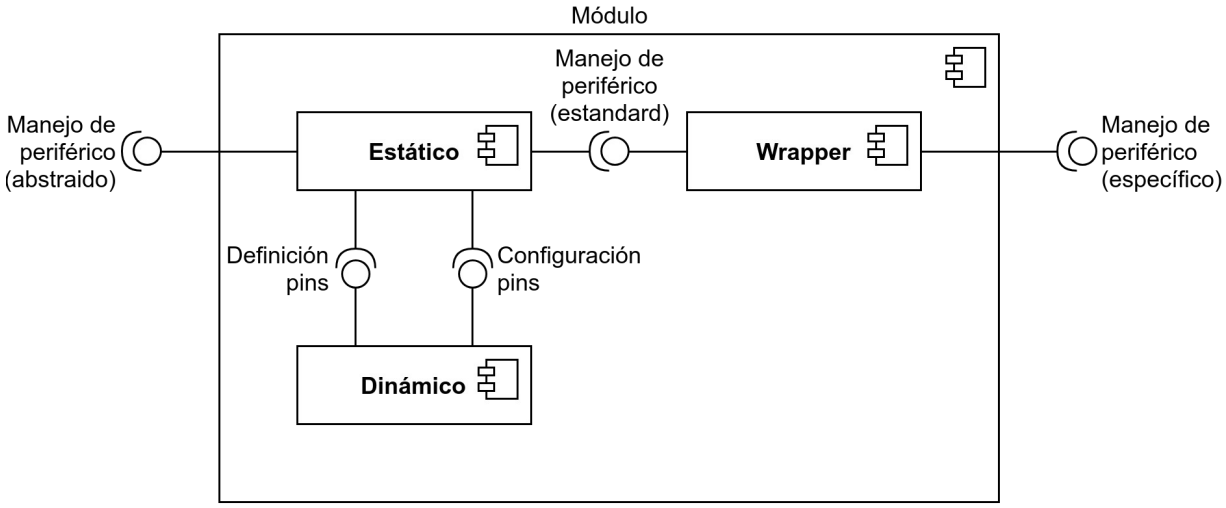

Figura 3.2 Arquitectura de un módulo dentro del framework

El propósito de un módulo es abstraer el manejo de periféricos del microcontrolador de un nivel alto, desacoplado del microcontrolador a un nivel bajo, específico al microcontrolador. Para lograr este propósito el módulo tiene como requerimiento de interfaz el controlador específico del microcontrolador para manejar el periférico en cuestión.

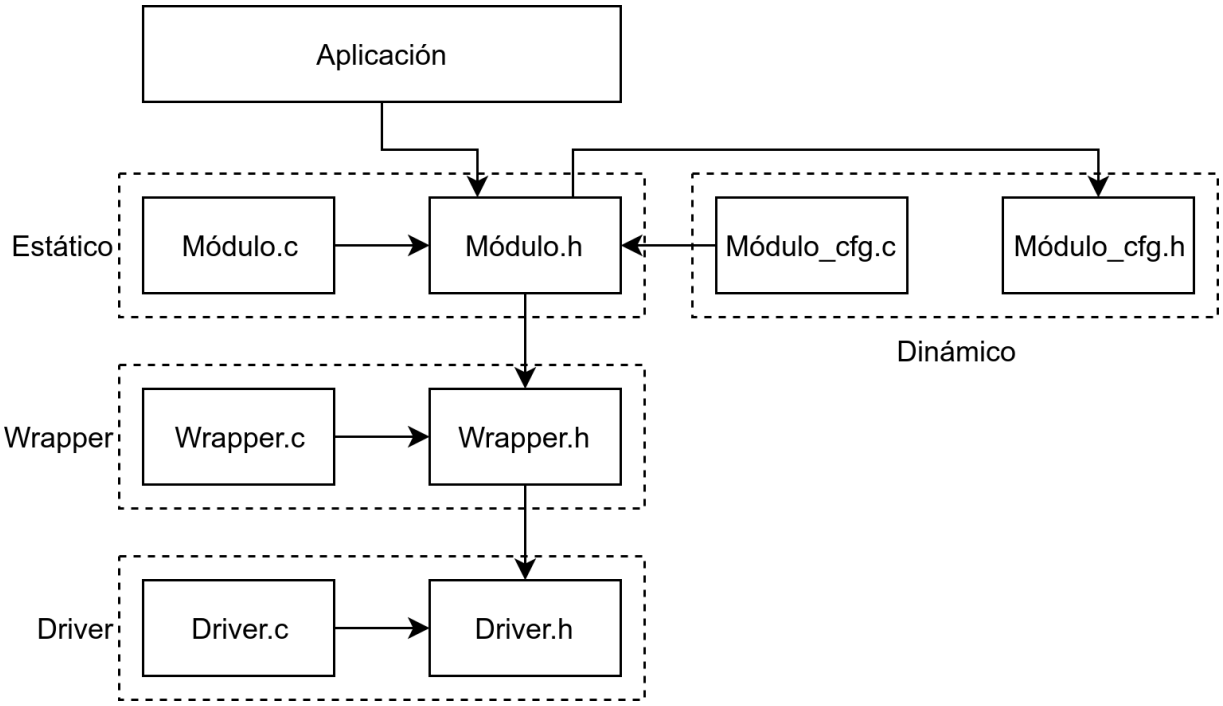

Figura 3.3 Relación de archivos de un módulo

La Figura 3.3 muestra la relación de archivos del *framework* propuesto para un módulo genérico. Las líneas punteadas marcan distintos grupos de archivos, a los cuales se les referirá como “elementos”.

La estructura de los archivos del *framework* en el proyecto es la siguiente:

- Framework
  - common
    - frameworkCommon.h
    - frameworkIncludes.h
  - drivers
    - {módulo}
      - cfg
        - {módulo}_cfg.c
        - {módulo}_cfg.h
      - inc
        - {módulo}.h
      - src
        - {módulo}.c

A continuación se describen los elementos que conforman un módulo.

3.2.1 Elemento estático

Este elemento está compuesto de un conjunto de funciones genéricas para cualquier microcontrolador. El objetivo es que la aplicación u otros módulos de software puedan utilizar este módulo directamente sin necesitar saber cual microcontrolador se está utilizando, de esta manera se asegura que la aplicación

pueda ser migrada a otro microcontrolador sin modificaciones. Este módulo hace uso directo de los *wrappers* descritos en la sección 3.2.3 por medio de funciones con nombres y parámetros genéricos traducidos de los controladores específicos del microcontrolador.

Ejemplo de archivos que componen el elemento estático en un módulo GPIO:

- gpio.c
- gpio.h

```
void            Gpio_Init(const Gpio_cfg_t* cfgPtr);
void            Gpio_WritePort(Gpio_portId_t port, Gpio_data_t value);
void            Gpio_SetPin(Gpio_portId_t port, Gpio_pinId_t pin,
Gpio_pinState_t state);
Gpio_data_t     Gpio_ReadPort(Gpio_portId_t port);
Gpio_pinState_t Gpio_GetPin(Gpio_portId_t port, Gpio_pinId_t pin);
```

Figura 3.4 Ejemplo de funciones para un módulo estático GPIO

En la Figura 3.4 se pueden observar varias características de las funciones del elemento estático que lo hacen genérico para cualquier microcontrolador.

```
typedef struct
{
    Gpio_portId_t   Port;
    Gpio_pin_t      Pin;
    Gpio_mode_t     Mode;
    Gpio_alt_t      Alternate;
    Gpio_pull_t     Pull;
    Gpio_speed_t    Speed;
    Gpio_pinState_t InitOutValue;
} Gpio_cfg_t;
```

Figura 3.5 Ejemplo de definición de estructura de configuración para un pin GPIO

La función de inicialización tiene como parámetro de entrada un apuntador que lleva a un arreglo de estructuras que contienen toda la información necesaria para

establecer los estados iniciales de los pins, pero al llamar esta función no se requiere conocimiento de cuales pins se están estableciendo, ni siquiera saber la cantidad de pins que se van a configurar. La sección 3.2.2 describe como se define este arreglo de estructuras.

La estructura de configuración contiene toda la información necesaria para inicializar un pin GPIO sin necesidad de saber que arquitectura de microcontrolador se está usando.

Para poder manejar los puertos y pins del microcontrolador se utilizan tipos de datos personalizados, esto permite que estas funciones puedan ser utilizadas en distintas arquitecturas de microcontrolador, como 8, 16 o 32 bits. En la sección 3.2.2 se describirá como se definen estos tipos de datos en el elemento dinámico.

3.2.2 Elemento dinámico

Este elemento está compuesto de estructuras o arreglos de estructuras que indican el estado inicial del periférico a configurar, así como de un conjunto de definiciones particulares del microcontrolador a utilizar. Esto permite que el elemento estático pueda permanecer igual y el elemento dinámico sea el único que se configure por el usuario.

Ejemplo de archivos que componen el elemento estático en un módulo GPIO:

- gpio_cfg.c
- gpgio_cfg.h

```
const GPIO_cfg_t GPIO_cfg[GPIO_ELEMENTS_MAX] = {
    {
        LED_1_PORT,
        LED_1_PIN,
        LED_1_MODE,
        LED_1_ALT,
        LED_1_PULL,
        LED_1_SPEED,
        LED_1_INIT_OUT
    },
    {
        LED_2_PORT,
        LED_2_PIN,
        LED_2_MODE,
        LED_2_ALT,
        LED_2_PULL,
        LED_2_SPEED,
        LED_2_INIT_OUT
    },
    {
        SW_1_PORT,
        SW_1_PIN,
        SW_1_MODE,
        SW_1_ALT,
        SW_1_PULL,
        SW_1_SPEED,
        SW_1_INIT_OUT
    }
}
```

Figura 3.6 Ejemplo de arreglo de estructura dentro de un elemento dinámico en un módulo GPIO

La Figura 3.6 muestra la configuración inicial para un microcontrolador ficticio de 4 pins GPIO, donde se incluyen de manera amigable al desarrollador. El tipo de datos para la estructura fue establecido en el elemento dinámico, esto se puede observar en la Figura 3.5

```
/**
 * @brief GPIO pin IDs
 */
typedef enum
{
    PIN_0 = 0,
    PIN_1,
    PIN_2,
    PIN_3,
    PIN_4,
    PIN_5,
    PIN_6,
    PIN_7,
    PIN_8,
    PIN_9,
    PIN_10,
    PIN_11,
    PIN_12,
    PIN_13,
    PIN_14,
    PIN_15,
    PIN_MAX
} Gpio_pinId_t;

/**
 * @brief GPIO port data type
 */
typedef GPIO_TypeDef Gpio_port_t;

/**
 * @brief GPIO pin data type
 */
typedef uint32_t Gpio_pin_t;
```

Figura 3.7 Ejemplo de definición de tipos de configuración específicos

La Figura 3.7 muestra un ejemplo de definición de tipos de datos específicos del microcontrolador. Estas definiciones son necesarias para la estructura definida en la Figura 3.5

3.2.3 Wrapper

Los *wrappers* sirven como traductores entre el módulo estático del *framework* y los controladores específicos del microcontrolador. Estos deben ser específicos al microcontrolador, o al menos a la familia a la que pertenecen.

3.2.4 Archivos de cabecera generales

En la carpeta general de common se incluyen 2 archivos de cabecera (*headers*), los cuales son utilizados por todos los módulos del *framework*. Ambos archivos son generados por el auto-generador descrito en el capítulo 4

frameworkCommon.h incluye las librerías estándares necesarias y macros útiles para todos los módulos del *framework*, además de información de la versión del *framework*.

f**rameworkIncludes.h** contiene la inclusión de todos los módulos que se estén utilizando en el proyecto.

3.3 Módulos

A continuación se va a describir el elemento estático del módulo, el elemento dinámico será descrito en la sección 6.1.1.

3.3.1 GPIO

Las siglas GPIO en inglés (*General Purpose Input/Output*) se traducen como “Entrada/Salida de Propósito General”. Es una señal digital flexible controlada por software [53]. Es el tipo de función más común que puede tener un pin de un microcontrolador, y una de las más simples, por esta razón se eligió como el primer módulo a ejercitar.

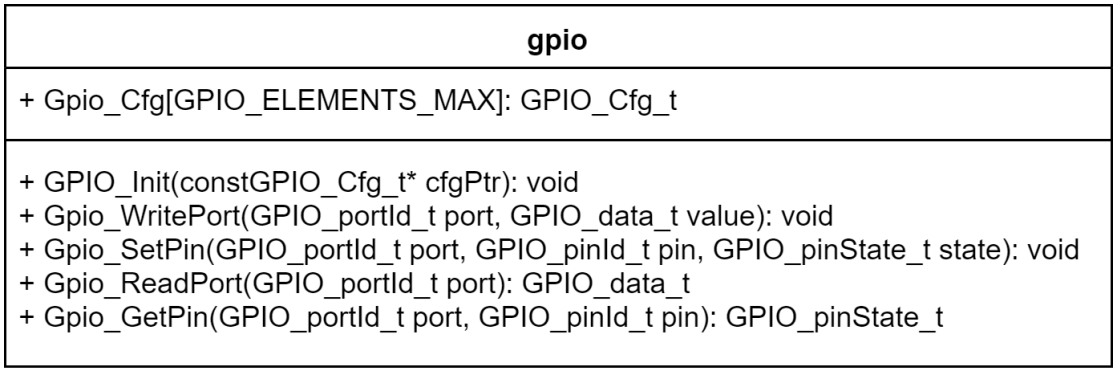

Figura 3.8 Clase gpio

3.3.1.1 gpio.h

Este archivo contiene definiciones generales necesarias para el módulo GPIO.

```
typedef struct
{
    GPIO_portId_t    Port;
    GPIO_Pin_t       Pin;
    GPIO_Mode_t      Mode;
    GPIO_Alternate_t Alternate;
    GPIO_Pull_t      Pull;
    GPIO_Speed_t     Speed;
} GPIO_Cfg_t;
```

Figura 3.9 Definición de estructura GPIO_Cfg_t

La definición principal es la de la estructura de configuración `GPIO_Cfg_t` de un pin GPIO mostrada en la Figura 3.9. Los elementos de la estructura representan las propiedades del pin que se esté configurando, y sus valores, las distintas opciones de configuración.

A continuación se describen los elementos de la estructura que deben de configurarse para cada pin. Se marcan con “Opcional” las propiedades que pueden no existir para todos los píns en ciertos microcontroladores.

- **Port**: Puerto del microcontrolador donde se encuentra el pin.
- **Pin**: Número del pin dentro del puerto.
- **Mode**: Modo de operación del pin, generalmente entrada o salida.
- **Alternate**: (Opcional) Función alterna del pin.
- **Pull**: (Opcional) Modo de resistencia de polarización
- **Speed**: (Opcional) Velocidad del pin.

El tipo de dato para los elementos de la estructura depende de la arquitectura de cada microcontrolador, por lo que la definición de los tipos de datos deberá estar dentro del *wrapper* para el microcontrolador.

En el elemento dinámico todos los elementos deben de estar configurados a algún valor, en caso de no tener una configuración disponible para ese pin se deberán configurar con un valor que lo indique. Este valor depende de los controladores que se estén utilizando en un microcontrolador específico, por lo que la definición de este valor reside en el *wrapper*. Las configuraciones posibles para estas características son descritas en la sección 6.1.1.

3.3.1.2 gpio.c

La Figura 3.8 muestra las funciones públicas del módulo gpio que están disponibles para que la aplicación pueda hacer uso de los pins GPIO.

La función `GPIO_Init(const GPIO_Cfg_t* cfgPtr)` inicializa cada pin GPIO del microcontrolador utilizando la información de las estructuras tipo

`GPIO_Cfg_t` descritas en la sección 3.3.1.1. Se encarga de configurar el modo, resistencia de polarización, velocidad, y de aplicar, también el estado de salida inicial del pin. Toma como argumento de entrada un apuntador al arreglo de estructuras y no regresa un valor.

La función `Gpio_WritePort(GPIO_portId_t port, GPIO_data_t value)` escribe un valor numérico a un puerto del microcontrolador para establecer una salida en los pins. Toma como argumentos de entrada el puerto y el valor a escribir y no regresa un valor.

La función `Gpio_SetPin(GPIO_portId_t port, GPIO_pinId_t pin, GPIO_pinState_t state)` permite establecer un estado binario a un pin en específico. Toma como argumentos de entrada el puerto, el pin y el valor a establecer, y no regresa un valor.

La función `Gpio_ReadPort(GPIO_portId_t port)` permite saber el estado binario de un puerto completo del microcontrolador. Toma como argumento de entrada el puerto, y regresa un valor numérico correspondiente al arreglo del total de los valores de los pins en el puerto.

La función `Gpio_GetPin(GPIO_portId_t port, GPIO_pinId_t pin)` permite saber el estado binario de un pin específico del microcontrolador. Toma como argumentos de entrada el puerto y el pin, y regresa el valor correspondiente al estado del pin.

Todas estas funciones toman como argumentos valores con tipos de datos que están definidos en los *wrappers* de cada microcontrolador. Estos datos son procesados y redirigidos a los controladores específicos a través de los *wrappers*.

4 AUTO-GENERADOR

El propósito del auto-generador es crear los elementos dinámicos de los módulos a partir de una interfaz gráfica amigable al usuario. Esto se logra por medio de plantillas como base para los archivos de código fuente, y archivos de configuración donde se indican las propiedades de los microcontroladores, así como las propiedades de los proyectos.

El alcance de este proyecto de tesis se limitará a la configuración del módulo GPIO probada en 2 microcontroladores: ST STM32F334R8 [54] y Texas Instruments MSP430FR6989 [55]. La sección 7.2 describe las características de estos microcontroladores.

4.1 Plantillas

Para poder generar los archivos fuente de los elementos dinámicos, se usan plantillas con un formato predefinido. Así el generador no necesita crear los archivos fuente desde cero, solo requiere modificar las plantillas, esto permite actualizar el generador y las plantillas por separado, cuando sea necesario hacer modificaciones al framework, o al generador.

```
/**
 * @file frameworkCommon.h
 * @author Miguel Diaz
 * @brief framework common header
 */

/************************
 * Guard                *
 ************************/
#ifndef _FRAMEWORK_COMMON_H_
#define _FRAMEWORK_COMMON_H_

/************************
 * Includes             *
 ************************/
FWK_COMMON_INCLUDES

/************************
 * Public Macros        *
 ************************/
#define FRAMEWORK_VERSION_MAJOR 0
#define FRAMEWORK_VERSION_MINOR 1
#define FRAMEWORK_VERSION_PATCH 0

/************************
 * Public Defines       *
 ************************/

// Bit/Byte related definitions
#define SHIFT_8_BITS (8)
#define LOW_BYTE_IN_16B_MASK (0x00FF)
#define HIGH_BYTE_IN_16B_MASK (0xFF00)

FWK_GPIO_COMMON_DEFINITIONS

#endif /* _FRAMEWORK_COMMON_H_ */
```

Figura 4.1 Plantilla para frameworkCommon.h

Para poder modificar las plantillas, es necesario indicar por medio de textos clave, en que secciones del archivo se puede escribir el código generado.

```
/**
 * @file frameworkCommon.h
 * @author Miguel Diaz
 * @brief framework common header
 */

/************************
 * Guard                *
 ************************/
#ifndef _FRAMEWORK_COMMON_H_
#define _FRAMEWORK_COMMON_H_

/************************
 * Includes             *
 ************************/
// ################## Kamino generator v1.0.0: Generated code! ################
// ######## Do NOT modify code between this header and the footer below #######
// Framework Common header files:
#include <stdlib.h>
#include <stdbool.h>
// ################## Kamino generator v1.0.0: Generated code! ################
// ######## Do NOT modify code between this footer and the header above #######

/************************
 * Public Macros        *
 ************************/
#define FRAMEWORK_VERSION_MAJOR 0
#define FRAMEWORK_VERSION_MINOR 1
#define FRAMEWORK_VERSION_PATCH 0

/************************
 * Public Defines       *
 ************************/

// Bit/Byte related definitions
#define SHIFT_8_BITS (8)
#define LOW_BYTE_IN_16B_MASK (0x00FF)
#define HIGH_BYTE_IN_16B_MASK (0xFF00)

// ################## Kamino generator v1.0.0: Generated code! ################
// ######## Do NOT modify code between this header and the footer below #######
// ST STM32F334R8 Common definitions:

// ################## Kamino generator v1.0.0: Generated code! ################
// ######## Do NOT modify code between this footer and the header above #######

#endif /* _FRAMEWORK_COMMON_H_ */
```

Figura 4.2 frameworkCommon.h con código generado

La Figura 4.1 muestra la plantilla para el archivo de encabezado **frameworkCommon.h**, ahí se pueden observar los textos clave `FWK_COMMON_INCLUDES` y `FWK_GPIO_COMMON_DEFINITIONS`. Estos textos serán

reemplazados por el generador por los archivos de cabecera requeridos por el proyecto, y definiciones específicas para el módulo GPIO respectivamente.

La Figura 4.2 muestra un ejemplo de un archivo **frameworkCommon.h** con código generado para un proyecto basado en un microcontrolador STM32F334R8. Aquí se puede observar que los textos clave fueron reemplazados por código del generador. En este caso específico las librerías comunes que fueron requeridas son **stdlib.h** y **stdbool.h**, y ninguna definición común para el módulo de GPIO fue necesaria. El ejemplo muestra también que al reemplazar los textos clave se incluye en el archivo un comentario indicando que parte del código es generada y no debería ser modificada, así como el nombre y versión del generador usado.

4.2 Archivos de configuración

Para poder almacenar las capacidades del microcontrolador, así como las configuraciones del proyecto, se utilizan archivos de configuración tipo XML.

4.2.1 Capacidades del microcontrolador

El archivo de capacidades del microcontrolador contiene la siguiente información necesaria por el generador, así como los marcadores (*tags*) XML utilizados para identificar el campo correspondiente.

Fabricante: <manufacturer> La compañía fabricante del microcontrolador

Familia: <family> La familia del microcontrolador

Modelo: <model> El número de parte completo del microcontrolador

Archivos de cabecera comunes del *framework*: <include_common> Los archivos de cabecera necesarios para todo el *framework*.

Archivos de cabecera específicos del módulo: <include_{módulo}> Los archivos de cabecera necesarios solo para un módulo.

Definiciones comunes del *framework*: <cfg_def_common> Definiciones necesarias para todo el *framework.*

Definiciones específicas del módulo: <cfg_def_{módulo}> Definiciones necesarias solo para un módulo.

Pin: <pin> Pin del microcontrolador, contiene las siguientes características:

- **Número:** <number> Número del pin en el empaquetado del microcontrolador.
- **Nombre:** <name> Nombre del pin, de acuerdo al fabricante.
- **Función principal:** Función principal del pin, puede tener las siguientes opciones:
  - **Alimentación:** <vcc> Voltaje de alimentación.
  - **Referencia:** <gnd> Referencia de voltaje.
  - **Reset:** <reset> Reset.
  - **Miscelaneo:** <misc> Otra función especial aun no soportada.
  - **GPIO:** <gpio> Puerto de entrada/salida general.
- **Puerto:** <port> Puerto del pin GPIO.
- **Pin de puerto:** <portPin> Número del pin dentro del puerto de acuerdo a la librería de controladores usada.

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<!DOCTYPE xml>
<microcontroller>
	<manufacturer>ST</manufacturer>
	<family>STM32</family>
	<model>STM32F334R8</model>
	<includes>
		<include_common>stdlib.h</include_common>
		<include_common>stdbool.h</include_common>
		<include_gpio>stm32f3xx_hal.h</include_gpio>
	</includes>
	<configuration_definitions>
		<!-- GPIO definitions -->
		<cfg_def_gpio>PULL_NOT_AVAILABLE GPIO_NOPULL</cfg_def_gpio>
		<cfg_def_gpio>SPEED_FAST GPIO_SPEED_HIGH</cfg_def_gpio>
		<cfg_def_gpio>GPIO_ALT_NONE ((uint8_t)0xFF)</cfg_def_gpio>
		<cfg_def_gpio>MODE_INPUT GPIO_MODE_INPUT</cfg_def_gpio>
	</configuration_definitions>
	<pins>
		<pin>
			<!-- Backup power supply -->
			<number>1</number>
			<name>VBAT</name>
			<vcc>VBAT</vcc>
		</pin>
		<pin>
			<number>2</number>
			<name>PC13</name>
			<gpio>GPIO</gpio>
			<port>PORT_C</port>
			<portPin>PIN_13</portPin>
		</pin>
		<pin>
			<number>3</number>
			<name>PC14</name>
			<gpio>GPIO</gpio>
			<port>PORT_C</port>
			<portPin>PIN_14</portPin>
		</pin>
	</pins>
</microcontroller>
```

Figura 4.3 Fragmento de archivo de capacidades para un microcontrolador STM32F334R8

La Figura 4.3 muestra un ejemplo de un fragmento de un archivo de capacidades para un microcontrolador STM32F334R8. Aquí se pueden apreciar las distintas características indicadas por los marcadores de XML.

El anexo Error: Reference source not found incluye los archivos de capacidades para los microcontroladores utilizados como prueba de concepto.

4.2.2 Propiedades de proyecto

El archivo de configuración de proyecto contiene la siguiente información del proyecto con sus respectivos marcadores XML

Nombre: <name> Nombre del proyecto

Archivo de configuración: <ConfigFile> Archivo que contiene la configuración de los pins del microcontrolador, descrito en la sección 4.2.3

Archivo de microcontrolador: <ucFile> Archivo que contiene toda la información del microcontrolador a usar, descrito en la sección 4.2.1

Carpeta del *framework*: <frameworkFolder> carpeta donde se colocarán los archivos de código fuente generados.

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<!DOCTYPE xml>
<CodeGeneratorProject>
        <name>Test_Project</name>
        <ConfigFile>testProjectConf.xml</ConfigFile>
        <ucFile>STM32F334R8.xml</ucFile>
        <frameworkFolder>framework</frameworkFolder>
</CodeGeneratorProject>
```

Figura 4.4 Ejemplo de propiedades de proyecto

La Figura 4.4 muestra un ejemplo de archivo de propiedades de proyecto.

4.2.3 Configuración del proyecto

El archivo de configuración del proyecto contiene la siguiente información sobre la configuración seleccionada por el usuario de los periféricos del microcontrolador con sus respectivos marcadores XML. La siguiente descripción incluye la configuración del módulo GPIO.

Pin: <pin> Pin del microcontrolador, contiene la información de configuración seleccionada:

- **Nombre:** <name> Nombre del pin.
- **Puerto:** <port> Puerto GPIO al que pertenece el pin.
- **Selección:** <selected> Indica si el pin está seleccionado para ser configurado, puede tener los siguientes valores:
  - **Seleccionado:** YES
  - **No seleccionado:** NOT
- **Modo:** <Mode> Modo de configuración del pin, puede tener los siguientes valores:
  - **Entrada:** MODE_INPUT
  - **Salida:** MODE_INPUT
  - **Función alterna:** MODE_INPUT
  - **Entrada analógica:** MODE_ANALOG
- **Tipo de salida:** <OutType> En caso de aplicar, tipo de salida del pin, puede tener los siguientes valores:
  - **Salida en contrafase:** OTYPE_PUSH_PULL

- **Drenaje abierto:** OTYPE_OPEN_DRAIN
- **No disponible:** OTYPE_NOT_AVAILABLE

- **Nivel de salida:** <OutLevel> En caso de aplicar, nivel de salida del pin, puede tener los siguientes valores:
  - **Nivel bajo:** LOW
  - **Nivel alto:** HIGH
- **Resistencia de polarización:** <Pull> En caso de aplicar, tipo de polarización en la resistencia del pin, puede tener los siguientes valores:
  - **Pull-Up:** PULL_UP
  - **Pull-Down:** PULL_DOWN
  - **No disponible:** PULL_NOT_AVAILABLE
- **Velocidad:** <Speed> En caso de aplicar, la velocidad a la que trabajará el pin, puede tener los siguientes valores:
  - **Rápido:** SPEED_FAST
  - **Medio:** SPEED_MEDIUM
  - **Alto:** SPEED_HIGH
  - **No disponible:** SPEED_NOT_AVAILABLE
- **Nombre clave:** <codeName> El nombre clave que el usuario le puede dar al pin para identificarlo fácilmente.

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<Microcontroller_Configuration>
	<pin>
		PC13
		<name>PC13</name>
		<port>PORT_C</port>
		<selected>YES</selected>
		<Mode>MODE_INPUT</Mode>
		<OutType>OTYPE_PUSH_PULL</OutType>
		<OutLevel>LOW</OutLevel>
		<Pull>PULL_UP</Pull>
		<Speed>SPEED_FAST</Speed>
		<codeName>USER_BUTTON</codeName>
	</pin>
	<pin>
		PC14
		<name>PC14</name>
		<port>PORT_C</port>
		<selected>NOT</selected>
		<Mode>MODE_INPUT</Mode>
		<OutType>OTYPE_PUSH_PULL</OutType>
		<OutLevel>LOW</OutLevel>
		<Pull>PULL_NOT_AVAILABLE</Pull>
		<Speed>SPEED_FAST</Speed>
		<codeName>PC14</codeName>
	</pin>
</Microcontroller_Configuration>
```

Figura 4.5 Ejemplo de configuración del proyecto

La Figura 4.5 muestra un ejemplo del archivo de configuración para un proyecto. Aquí se pueden apreciar los nombres claves que el usuario eligió para identificar los pins de manera útil.

4.3 Arquitectura general

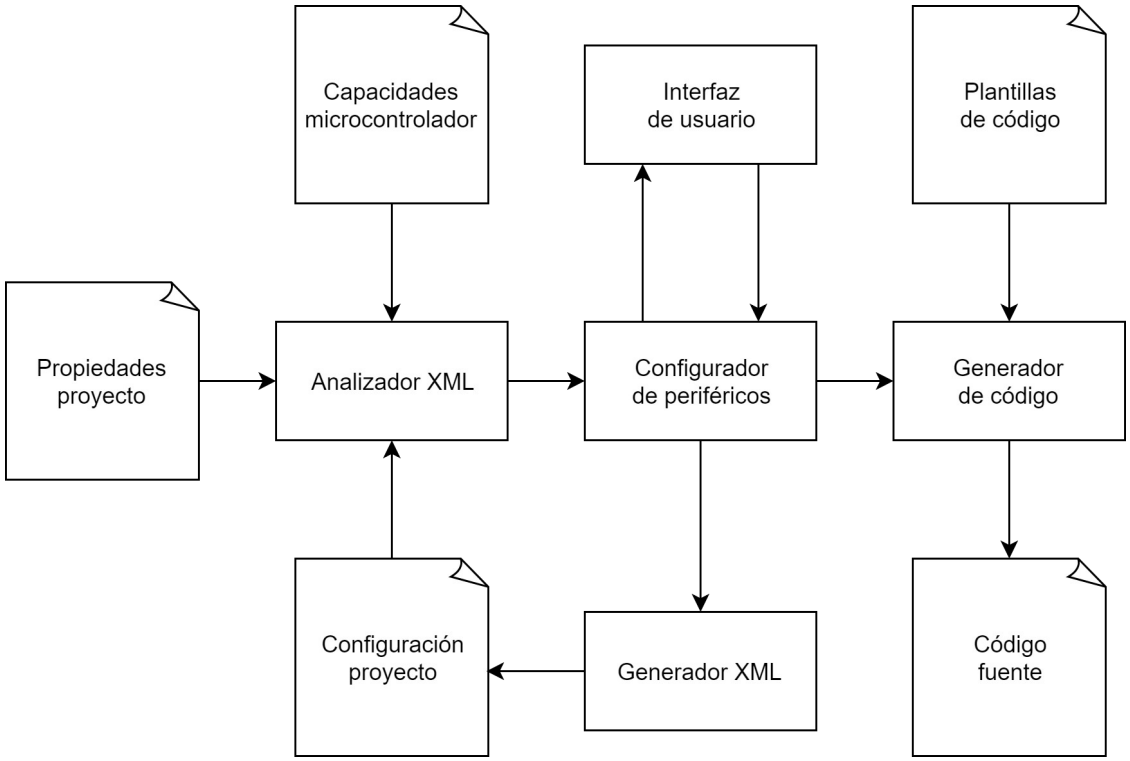

Figura 4.6 Arquitectura general del auto-generador

La Figura 4.6 muestra la arquitectura general del auto-generador. Incluye los archivos de configuración descritos en la sección 4.2, las plantillas descritas en la sección 4.1, y el código generado que será descrito en la sección 6.

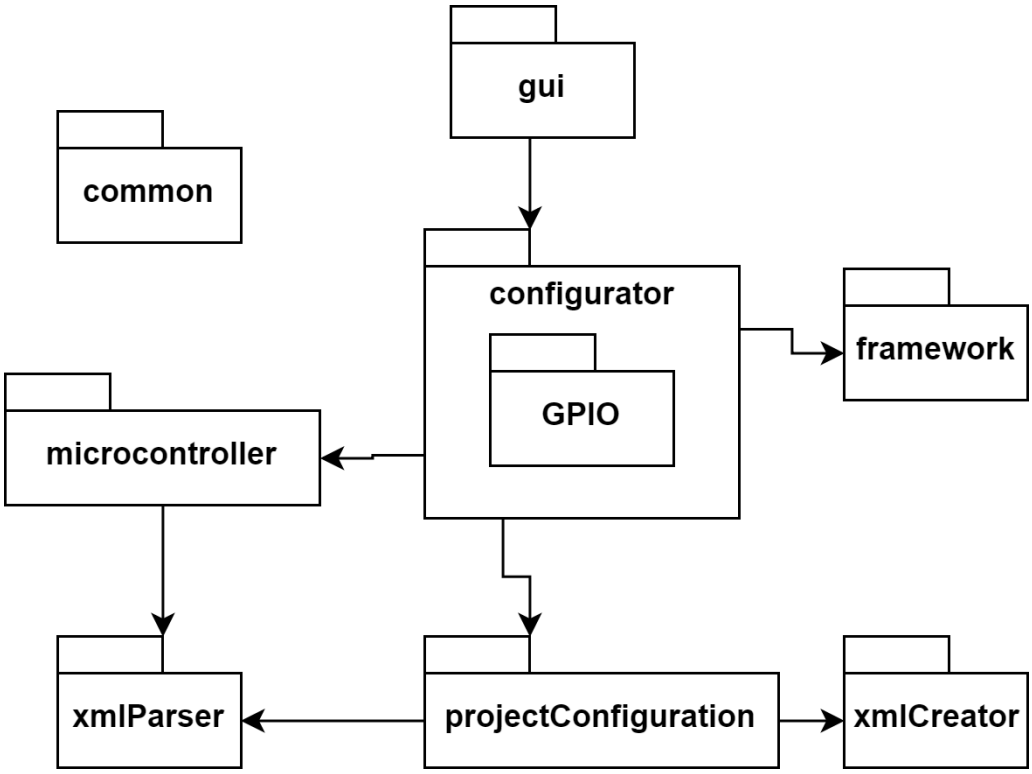

Figura 4.7 Diagrama de paquetes

La Figura 4.7 muestra la arquitectura del auto-generador desde el punto de vista de los paquetes y sus relaciones. El paquete common es dependencia de todos los demás, por lo que no se muestra explícitamente esta relación.

4.3.1 Analizador XML

La función del analizador (*parser*) XML es tomar la información dentro de archivos de configuración en formato XML y pasarla a objetos de JAVA que puedan interactuar dentro del auto-generador. El paquete utilizado para esto es **xmlParser**, la única clase que contiene es **XmlOpener**, y puede verse en la Figura 4.8.

XmlOpener
- XmlDoc: Document
+ XmlOpener(): Constructor + OpenFile(File inFile): ErrorCode + getParsedDoc(): Document + getElementInfoFromDoc(Document doc, String elementName): String + getElementInfo(Element element, String elementName): String

Figura 4.8 Clase XmlOpener dentro del paquete xmlParser

La clase XmlOpener se encarga de abrir los archivos de configuración o capacidades, validar que su formato (no su contenido) sea correcto, y extraer los elementos requeridos. Este paquete es utilizado por los paquetes de **microcontroller** y **projectConfiguration** para extraer la información del microcontrolador y de la configuración del proyecto respectivamente.

4.3.2 Generador XML

ConfXmlWriter
- XmlDoc: Document - RootElement: Element - Element[]: PinElement - UCConf: Microcontroller
+ ConfXmlWriter(Microcontroller uC): Constructor - configurePins(): void + addPin(PinConf pin, int pinNum): void - addPinChild(String elName, String elInfo, int pinNum): void + writeXml(String fileName): ErrorCode

Figura 4.9 Clase ConfXmlWriter dentro del paquete xmlCreator

La clase ConfXmlWriter se encarga de crear los documentos de configuración del proyecto. Este paquete es usado por el paquete **configurator**.

4.3.3 Configurador de periféricos

Para poder lograr la configuración de los periféricos se hace uso de 2 paquetes: **microcontroller** (Figura 4.10), y **configurator** (Figura 4.11).

4.3.3.1 Microcontroller

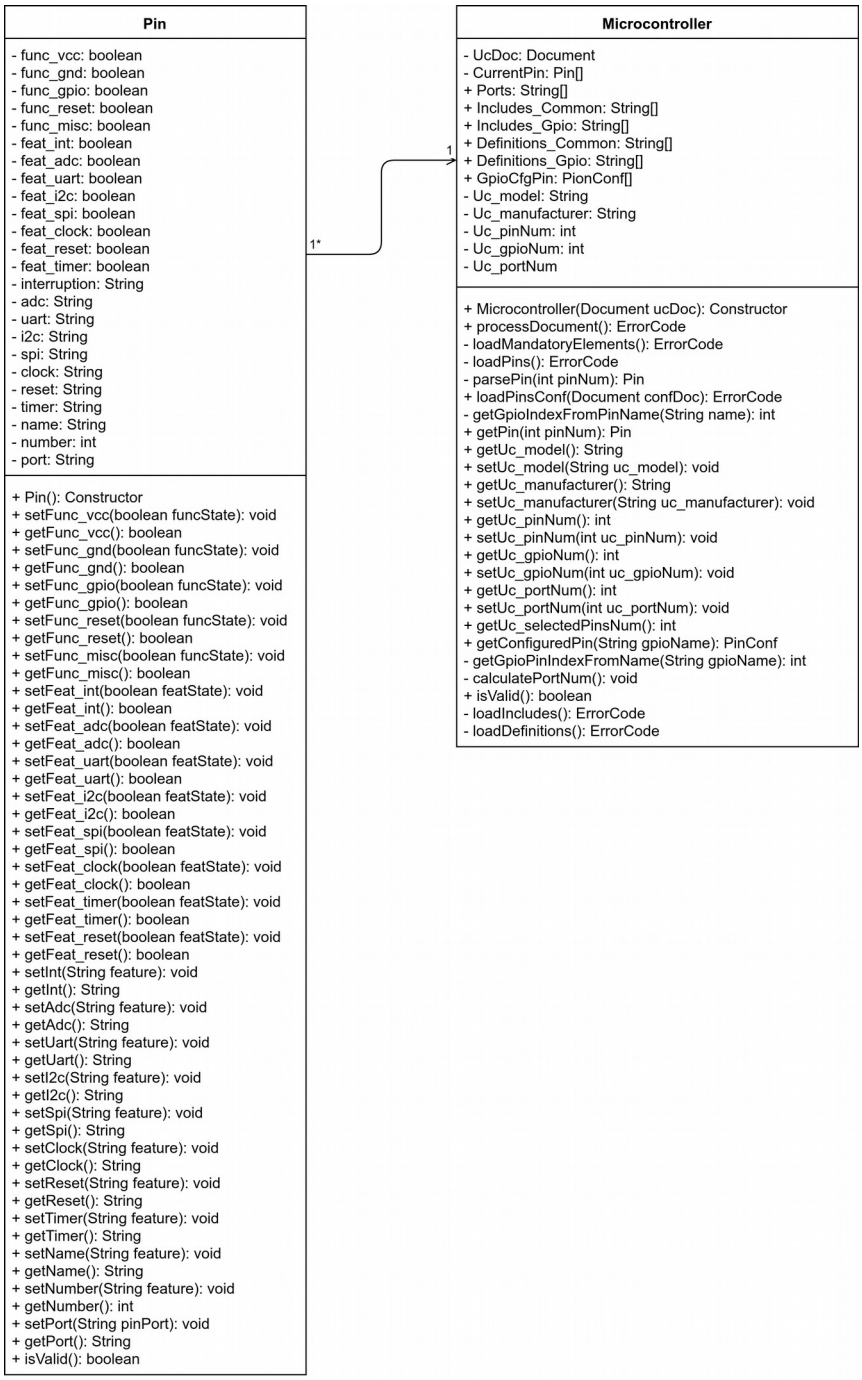

Figura 4.10 Clases en el paquete microcontroller

El paquete **microcontroller** tiene 2 clases: **Pin** y **Microcontroller**. La clase **Pin** tiene toda la información sobre las funciones y características que puede tener un pin, esta información es extraída del archivo de capacidades del microcontrolador.

Funciones:

- **Vcc**: Voltaje de alimentación.
- **Gnd:** Voltaje de referencia.
- **Reset:** Reset.
- **GPIO:** Entrada o salida de propósito general.
- **Misceláneo:** Función actualmente no soportada por el generador.

Características actualmente en el alcance del generador:

- **Nombre:** Nombre del pin según el fabricante.
- **Número:** Número del pin dentro del empaquetado.
- **Puerto:** Puerto al que pertenece el pin GPIO

Características actualmente fuera del alcance del generador:

- **Interrupción:** Interrupción de un pin.
- **ADC:** Convertidor analógico a digital.
- **UART:** Receptor-transmisor asíncrono universal [56].
- **I^2C:** Comunicación inter-circuitos integrados [57].
- **SPI:** Interfaz serial periférica [58].
- **Clock:** Conexión con el oscilador o cristal externo.
- **Reset:** Reset general del microcontrolador.

- **Timer:** Capacidad de mostrar un contador interno en un pin.

Todos estas funciones y características están contenidas en campos privados dentro de la clase, el establecimiento y acceso a esta información se hace a través de métodos públicos, para poder validar que se establezca correctamente un pin. Por ejemplo, el método `setFunc_gpio(boolean funcState)` no solo se encarga de establecer que el pin tiene la función de GPIO, también deshabilita las demás funciones no compatibles, o en caso de deshabilitar la función de GPIO, también deshabilita las características relacionadas, como UART, I^2C, etc.

Además de los métodos para establecer y acceder a las funciones y características del pin, existe un método para validar que el pin esté configurado correctamente. Este método revisa que el pin tenga un nombre, un número, y solamente una función. En caso de no cumplir con cualquiera de estas condiciones, se establece como un pin inválido, generando un código de error que se propaga por el programa.

La clase Microcontroller utiliza una instancia de la clase **Pin** para cada pin del microcontrolador, validando que la información contenida en el archivo sea correcta. Esta clase es la que es usada por los demás paquetes. Además de información de los pins del microcontrolador, la clase **Microcontroller** contiene lo siguiente.

Características del microcontrolador:

- **Fabricante:** Compañía que fabrica el microcontrolador.
- **Familia:** Grupo al que pertenece el microcontrolador.
- **Modelo:** Número de parte completo.
- **Número de pins:** Número total de pins.
- **Número de GPIOs:** Número total de pins que tienen la función de GPIO.

- **Número de puertos:** Número total de puertos GPIO.

Información necesaria para la compilación:

- **Archivos a incluir comunes:** Archivos de cabecera necesarios para la compilación de todo el *framework*.

- **Archivos a incluir de GPIO:** Archivos de cabecera necesarios para la compilación del módulo GPIO, mayormente librerías utilizadas por los *wrappers*.

- **Definiciones comunes:** Definiciones necesarias para la compilación de todo el *framework*.

- **Definiciones de GPIO:** Definiciones necesarias para la compilación del módulo GPIO.

Características para configuración:

    - **Configuración de un pin GPIO:** Configuración de un pin GPIO, descrito a detalle en la sección 4.3.3.2

Por último, la clase **Microcontroller** incluye un método donde se valida el microcontrolador completo. Revisando que tenga la información necesaria, como fabricante, familia y modelo, también se valida que tenga al menos un pin tipo GPIO, un pin de alimentación, y un pin de referencia para que pueda configurarse correctamente.

4.3.3.2 Configurator

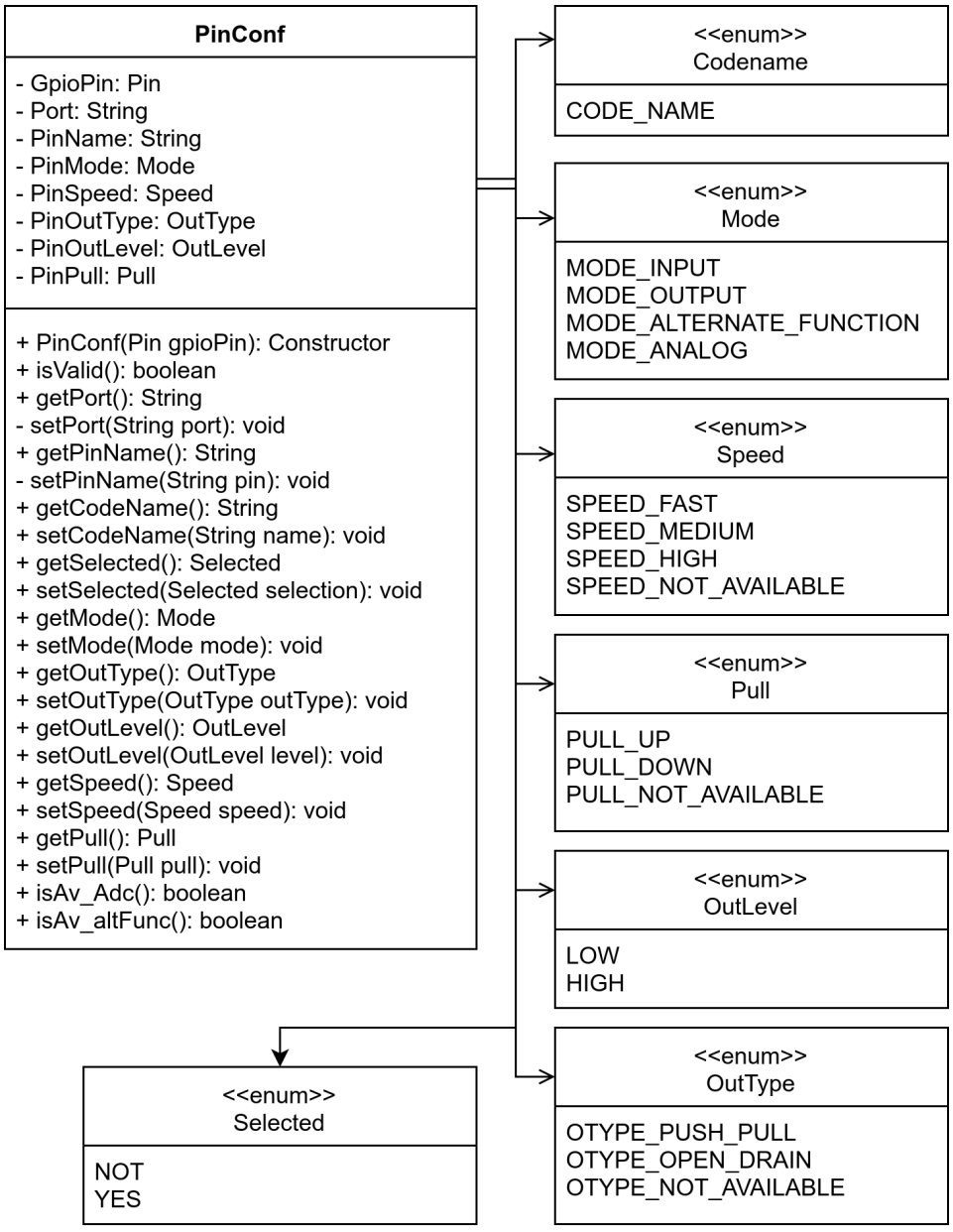

Figura 4.11 Clases dentro del paquete configurator

El paquete **Configurator** tiene una clase principal, la cual hace uso de un subpaquete, **Configurator.GPIO**. Este subpaquete contiene enumeraciones con las configuraciones posibles para un pin GPIO.

Enumeraciones para un pin GPIO:

- **Nombre clave:** Nombre con el que el usuario puede identificar al pin dentro del código en forma de una cadena de caracteres.
- **Selección:** Indica si el pin ha sido seleccionado para configuración.
  - **No seleccionado:** NOT
  - **Seleccionado:** YES
- **Modo:** modo de operación del pin
  - **Entrada:** MODE_INPUT – Entrada
  - **Salida:** MODE_OUTPUT – Salida
  - **Función alterna:** MODE_ALTERNATE_FUNCTION – Función alterna, como comunicación serial.
  - **Análogo:** MODE_ANALOG – Entrada analógica
- **Nivel de salida:** Nivel del pin en caso de que haya sido configurado como salida.
  - **Bajo:** LOW
  - **Alto:** HIGH
- **Tipo de salida:** Configuración del pín en salida.
  - **Salida en contrafase (*Push-Pull*):** OTYPE_PUSH_PULL
  - **Drenaje abierto (*Open drain*):** OTYPE_OPEN_DRAIN

- **Resistencia de polarización:**
  - **PullUp:** PULL_UP
  - **PullDown:** PULL_DOWN
  - **No disponible:** PULL_NOT_AVAILABLE
- **Velocidad del pin:**
  - **Rápido:** SPEED_FAST
  - **Medio:** SPEED_MEDIUM
  - **Alto:** SPEED_HIGH
  - **No disponible:** SPEED_NOT_AVAILABLE

4.3.4 Generador de código

4.3.4.1 framework

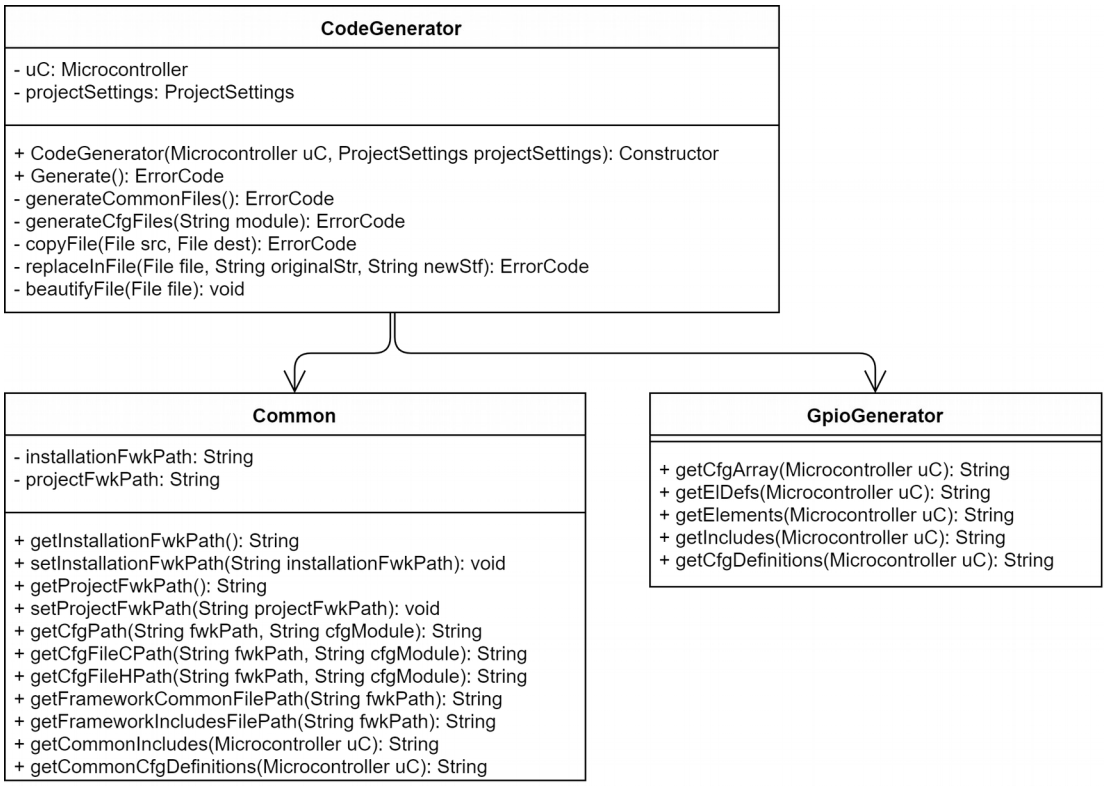

Figura 4.12 Clases dentro del paquete framework

El paquete ***framework*** es el que se encarga de la generación del código fuente, contiene las clases **Common**, **CodeGenerator** y **GpioGenerator**. La generación de código es descrita a detalle en la sección 6.

La clase **Common** contiene información común para todos los módulos:

- Ubicación de la carpeta que contiene plantillas para código.

- Ubicación de la carpeta que contiene el *framework* para el proyecto, donde se va a colocar el código generado.
- Ubicación de las plantillas de los elementos dinámicos.
- Ubicación de los archivos de código de elementos dinámicos.
- Ubicación de los archivos de cabecera del *framework*.
- Archivos de cabecera necesarios para la compilación del *framework*.
- Archivos de cabecera necesarios para la compilación de un módulo.

La clase **GpioGenerator** crea el código generado para reemplazar el texto clave en las plantillas. Los detalles de la generación son descritos en la sección 6.1. Genera la siguiente información:

- Arreglo de configuración de pins.
- Definiciones de elementos, o características de configuración de los pins.
- Archivos de cabecera necesarios para compilar el módulo GPIO. Básicamente son los requerimientos para los *wrappers*.
- Definiciones necesarias para compilar el módulo. Permiten interactuar con los *wrappers*.

La clase **CodeGenerator** se encarga de coordinar la generación de código de los múltiples módulos que se espera tenga el proyecto, actualmente el único habilitado es el módulo GPIO. El único método relevante para los demás paquetes es el de generar código.

4.3.5 Interfaz de usuario

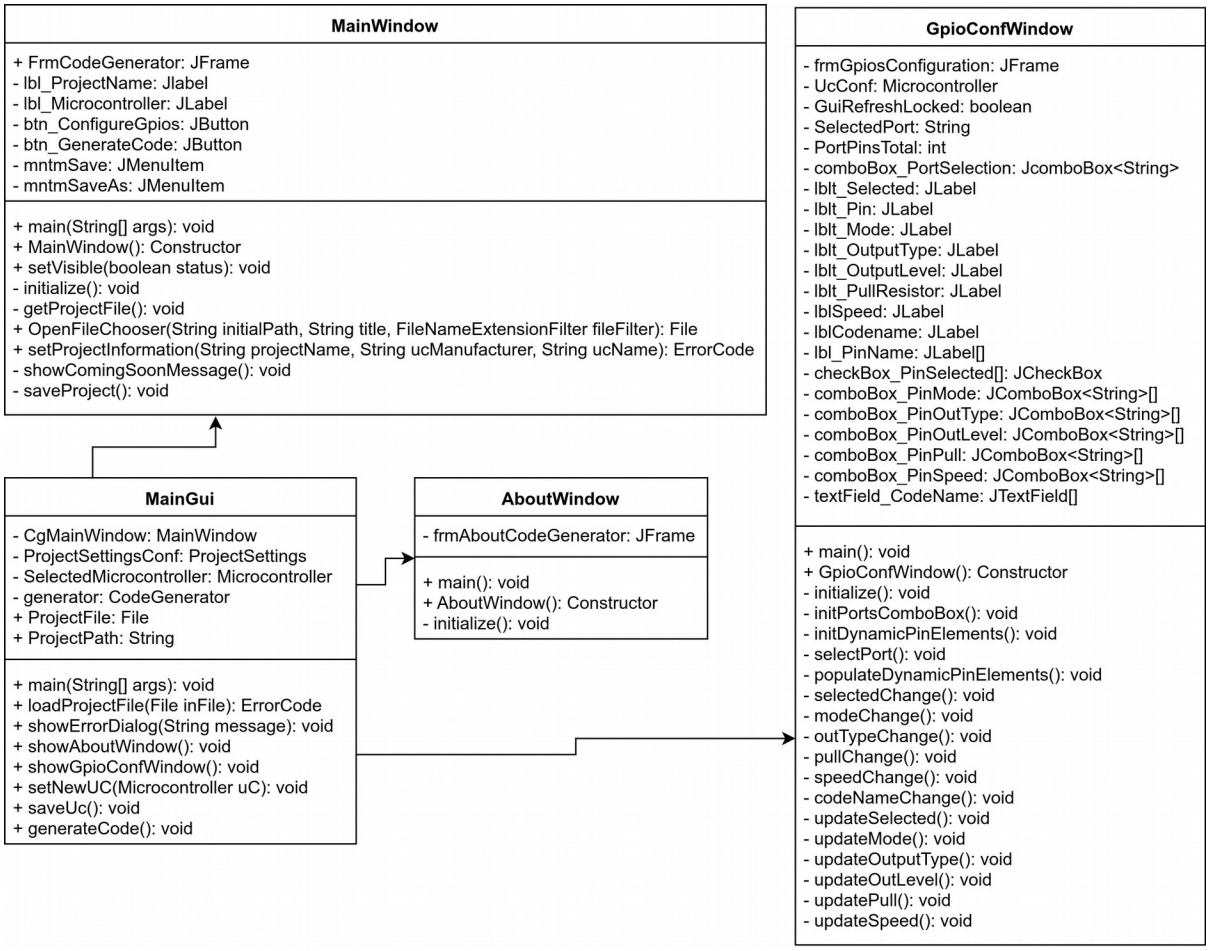

Figura 4.13 Clases dentro del paquete gui

El paquete **gui** contiene las clases que conforman la interfaz gráfica. Esta interfaz será detallada con capturas de pantalla en la sección 5.

4.3.5.1 MainGui

La clase **MainGui** es el punto de entrada de todo el programa (clase **main**). Al iniciar, se manda llamar a las demás clases de la interfaz gráfica, y coordina cuando aparece o desaparece cada ventana.

4.3.5.2 MainWindow

Ventana principal del programa, aquí se selecciona el proyecto con el que se va a trabajar y se controla la generación de código, así como el guardado del proyecto.

4.3.5.3 AboutWindow

Ventana de información diversa del generador, como versión y autor.

4.3.5.4 GpioConfWindow

Ventana para la configuración de pins GPIO.

4.3.6 Common

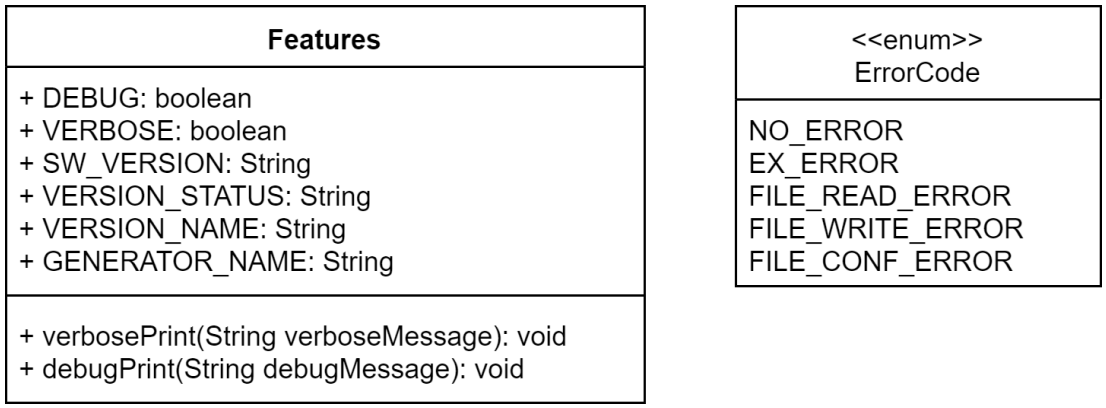

Figura 4.14 Clases dentro del paquete common

El paquete **common** contiene una clase y una enumeración que son utilizadas por el resto del programa para funcionamiento general.

4.3.6.1 Features

Campos comunes:

- **Depuración:** DEBUG – Bandera que indica si se está ejecutando una versión de depuración del programa. Esto sirve para ejecutar tareas que ayudan al desarrollo, pero no son necesarias para la versión final.
- **Verboso:** VERBOSE – Bandera que indica si el programa va a mostrar información explícita a través de la terminal del sistema.
- **Versión de software:** SW_VERSION – Indica la versión del programa.
- **Estado de versión:** VERSION_STATUS – Indica el estado del programa siendo ejecutado, como versión alfa, beta, etc.
- **Nombre de versión:** Cada versión tiene un nombre clave para distinguirla. Actualmente las versiones van en orden alfabético utilizando nombres de planetas dentro del universo ficticio de las películas que pertenecen a la serie "La guerra de las galaxias" [59].
- **Impresión verbosa:** El método `verbosePrint()` permite imprimir mensajes en la consola del sistema para que el usuario o desarrollador pueda saber el estado y progreso del programa. Cada mensaje va marcado con un # de sufijo que indica que es un mensaje informativo. Esta función solo tiene efecto cuando la bandera VERB0SE está establecida.
- **Impresión de depuración:** El método `debugPrint()` permite imprimir mensajes a la consola del sistema para que el desarrollador obtenga información para el desarrollo y depuración del programa, pero que no es de

interés o utilidad para el usuario final. Cada mensaje va marcado con un #$ de sufijo que indica que es un mensaje de depuración. Esta función solo tiene efecto cuando la bandera DEBUG está establecida.

4.3.6.2 ErrorCode

La enumeración **ErrorCode** contiene una lista de posibles estados de error en los que puede entrar el programa. Son utilizados para poder confirmar que una operación fu exitosa, o reaccionar a cierto error para fallar de manera planeada, así como indicar al desarrollador cual fue el tipo de fallo.

- **Sin error:** `NO_ERROR` – Indica que la operación fue terminada de manera exitosa.
- **Error de ejecución:** `EX_ERROR` – Código para un error genérico, o sin un grupo especifico.
- **Error de lectura de archivo:** `FILE_READ_ERROR` – Error al leer un archivo, puede ser porque el archivo no existe, o tiene un formato incorrecto.
- **Error de escritura de archivo:** `FILE_WRITE_ERROR` – Error al escribir un archivo, puede ser porque la ubicación no existe, o por cuestión de permisos en la carpeta.
- **Error de configuración de archivo:** `FILE_CONF_ERROR` – Error en la configuración de un archivo.

5 INTERFAZ GRÁFICA

Para que el programa sea amigable con el usuario, es necesario tener una interfaz gráfica que permita la configuración del microcontrolador de manera sencilla y explícita.

5.1 Swing

Swing es un conjunto de herramientas para interfaces gráficas de usuario en JAVA que incluye un conjunto de artilugios (*widgets*) [60]. Está basado en lo que provee el paquete de AWT, pero provee componentes ligeros que son independientes de la plataforma donde son ejecutados [61], esto permite que el mismo código sea compatible en Windows y Linux. Swing incluye controles como botones, barras de herramientas, campos de texto, etc.

A continuación se muestran capturas de pantalla de la interfaz gráfica. Aunque el programa fue probado en Windows 10 y Linux, todas las capturas mostradas fueron ejecutadas en el sistema operativo Linux corriendo el entorno gráfico Gnome [62] versión 3.32.2 utilizando el tema oficial *Adwaita* a menos de que se especifique lo contrario.

5.2 Ventana principal

Figura 5.1 Ventana principal

La ventana principal ofrece la interacción inicial con el usuario. Permite seleccionar el archivo de proyecto a utilizar, y provee acceso a la ventana de configuración y de información, así como del botón que comienza la generación del código fuente. Aquí se despliega la información del proyecto actual, incluyendo el microcontrolador usado. Se propone que en esta ventana se agreguen botones para acceder a las ventanas de configuración de todos los módulos habilitados, por el momento se muestra el acceso a la ventana de configuración de GPIO.

5.3 Ventana de información

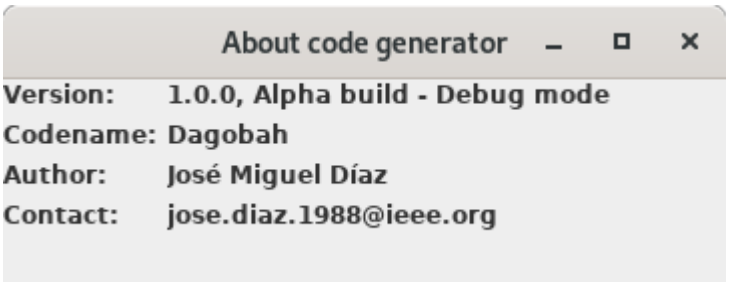

Figura 5.2 Ventana de información

La ventana de información despliega información sobre el generador, como la versión y su estado, nombre clave y si se está ejecutando una versión de depuración. Además muestra la información del autor y como contactarlo.

5.4 Ventanas de configuración

Se propone tener una ventana de configuración para cada módulo del *framework* que esté soportado, para el alcance de este proyecto de tesis, esto se limita al módulo GPIO.

5.4.1 Configuración GPIO

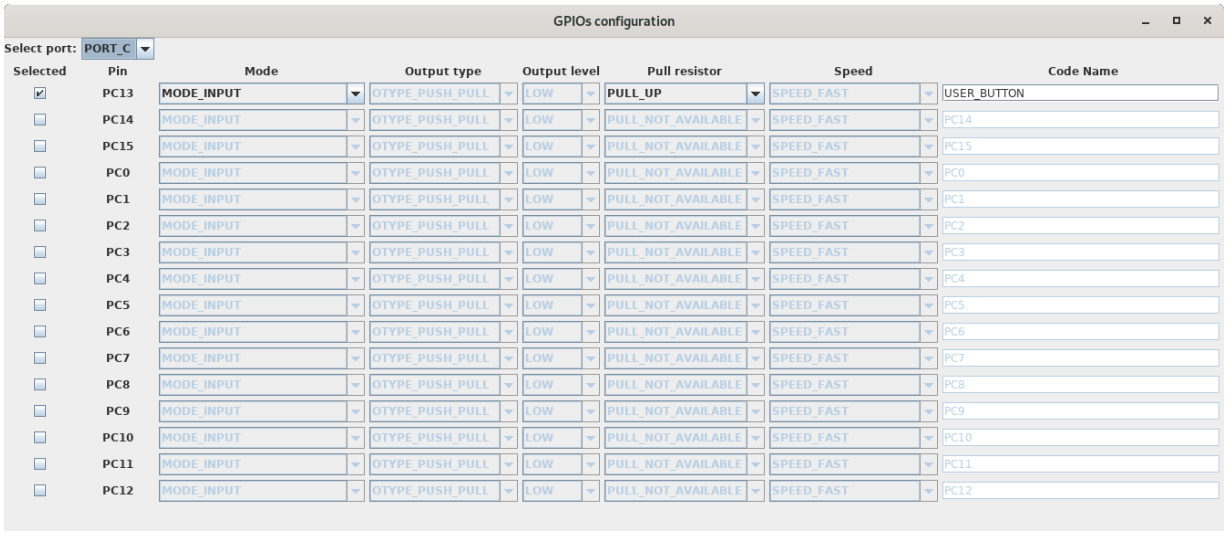

Figura 5.3 Ventana de configuración de GPIO

La ventana de configuración de GPIO muestra los pins del microcontrolador agrupados de acuerdo al puerto al que pertenecen. Los campos que permite modificar están descritos en la sección 4.3.3.2. En la parte superior izquierda se muestra un listado de los puertos disponibles en el microcontrolador para seleccionar. Los elementos de esta ventana se generan de manera dinámica dependiendo del número de pins en el puerto seleccionado.

Una caja de selección permite al usuario elegir cuales pins se van a incluir en la configuración, solo se va a generar código para los pins seleccionados. Una etiqueta muestra el nombre del pin. Las características de modo, tipo de salida, nivel de salida, resistencia de polarización, y velocidad se despliegan en menús seleccionables. Se habilitan y deshabilitan automáticamente dependiendo de la configuración de otras características como se detalla a continuación.

Características a configurar:

- **Selección**
  - Siempre habilitado
- **Modo**
  - Habilitado cuando Selected es verdadero
  - Deshabilitado cuando Selected es falso
- **Tipo de salida**
  - Habilitada cuando Mode es MODE_OUTPUT y Selected es verdadero
  - Deshabilitada cuando Mode es MODE_INPUT, MODE_ALTERNATE_FUNCTION, o MODE_ANALOG, o cuando Selected es falso
- **Nivel de salida**
  - Habilitada cuando Mode es MODE_OUTPUT y Selected es verdadero
  - Deshabilitada cuando Mode es MODE_INPUT o Selected es falso
- **Resistencia de polarización**
  - Habilitada cuando Mode es MODE_INPUT y Selected es verdadero
  - Deshabilitada cuando Mode es MODE_OUTPUT, MODE_ALTERNATE_FUNCTION, o MODE_ANALOG, o Selected es falso
- **Velocidad**
  - Habilitada cuando Mode es MODE_OUTPUT y Selected es verdadero
  - Deshabilitada cuando Mode es MODE_INPUT, MODE_ALTERNATE_FUNCTION, o MODE_ANALOG, o Selected es falso

- **Nombre clave**
  - Habilitado cuando Selected es verdadero
  - Deshabilitado cuando Selected es falso

Todas estas características son guardadas en el archivo de configuración del proyecto, aunque el pin esté deshabilitado.

5.5 Ventanas de selección de archivos

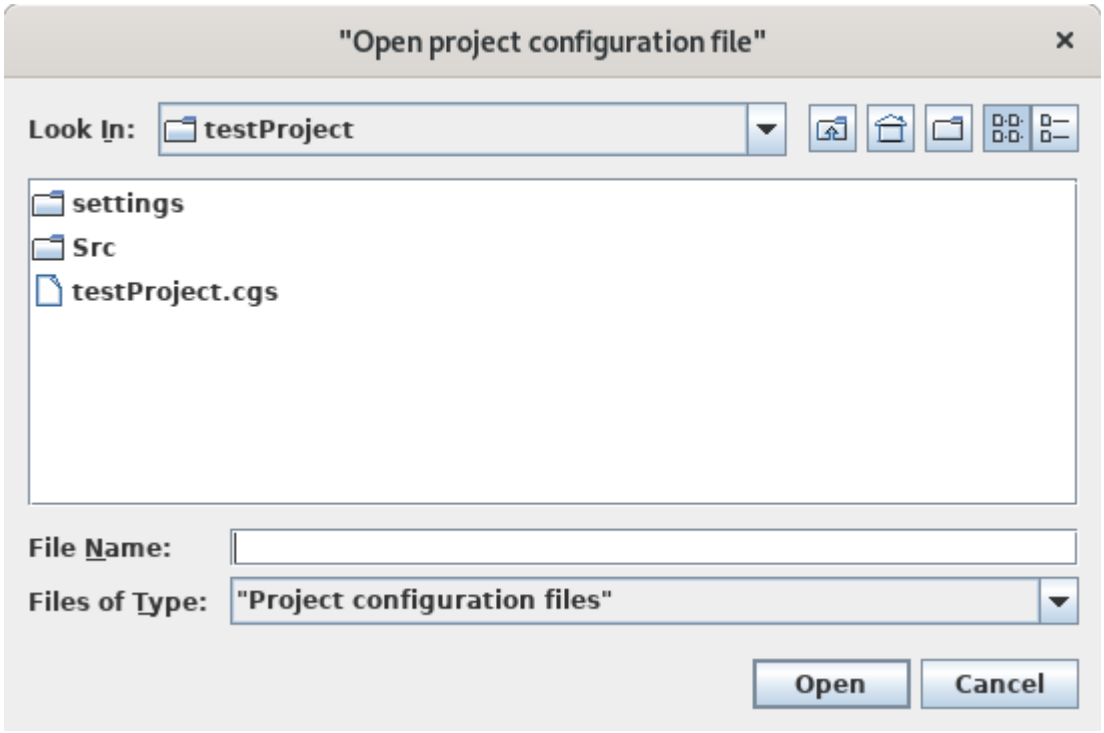

Figura 5.4 Ventana de selección de archivos en Linux con Gnome 3.32.2

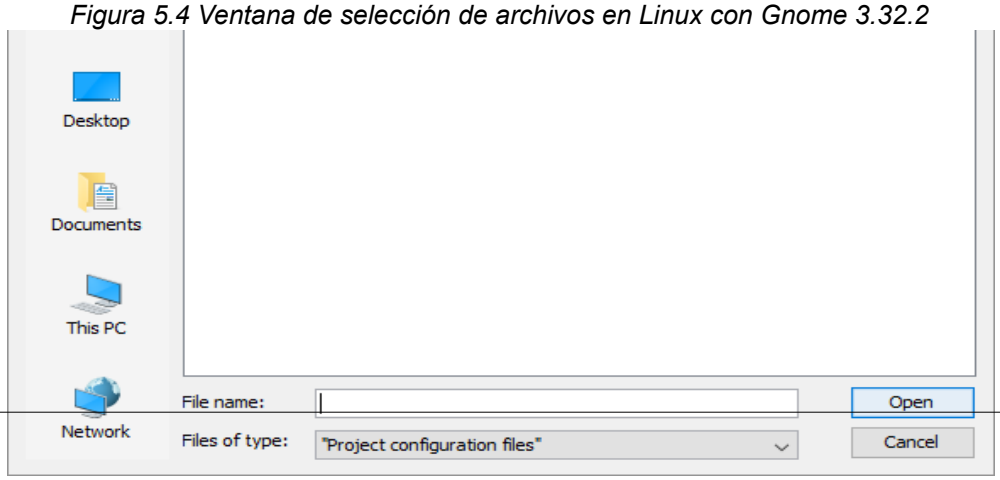

Figura 5.5 Ventana de selección de archivo en Windows 10

Para la selección de archivos se utiliza la herramienta de swing *JfileChooser*. Esta herramienta utiliza el selector de archivos del sistema, por lo que la ventana es diferente en cada sistema operativo. Esto es visible comparando la Figura 5.4 y la Figura 5.5, en donde se observa la misma ventana en los sistemas operativos Linux y Windows respectivamente.

6 GENERACIÓN DE CÓDIGO

En la sección 4.1 se describió el modelo de plantillas que se utiliza para la generación de código. Los pasos principales para obtener el código son los siguientes:

1. Copiar plantillas desde la ubicación de instalación a la ubicación del *framework* dentro del proyecto.
2. Generar el código de los elementos dinámicos dentro de cada módulo.
3. Reemplazar los textos clave en las plantillas de los elementos dinámicos de cada módulo con el código generado.
4. Generar el código de los archivos de cabecera del proyecto.
5. Reemplazar los textos clave en las plantillas de los archivos de cabecera del proyecto

6.1 Configuración de módulos

La Figura 3.2 muestra la relación entre el elemento dinámico y el elemento estático, y la Figura 3.3 muestra la relación entre los archivos dentro del módulo. Al solo generar los elementos dinámicos de los módulos nos aseguramos que la interacción entre los elementos estáticos sea igual para una aplicación específica sin importar sobre que plataforma de hardware está corriendo.

6.1.1 Módulo GPIO

El propósito del elemento dinámico del módulo de GPIO es proveer un listado de los pins disponibles en el microcontrolador, así como de su configuración inicial.

Además se provee un nombre clave para cada pin que puede ser utilizado por el usuario para identificar pins por su función sin necesidad de conocer a que pin físico está conectado en el microcontrolador, lo que permite una migración de aplicación entre distintas plataformas de hardware.

6.1.1.1 gpio_cfg.h

En el anexo 10.2.4 se ve la plantilla usada de base para el archivo de configuración **gpio_cfg.h**. Este archivo contiene definiciones necesarias para el uso del *framework*, y la mayor parte de su contenido es generado.

La primer definición es de un enumerador generado llamado `Gpio_elementsType`, los elementos contenidos son todos los pins con capacidad general de entrada salida a configurar del microcontrolador. Los nombres de cada elemento corresponden al nombre clave indicado por el usuario en la ventana de configuración. La Figura 6.1 muestra un ejemplo de este código generado.

```
typedef enum
{
    // ################## Kamino generator v1.0.0: Generated code! ################
    // ######## Do NOT modify code between this header and the footer below #######
    USER_BUTTON,
    USER_LED,
    // ################## Kamino generator v1.0.0: Generated code! ################
    // ######## Do NOT modify code between this footer and the header above #######
    GPIO_ELEMENTS_MAX
} Gpio_elementsType;
```

Figura 6.1 Enumerador de elementos GPIO

En la sección de definiciones públicas se encuentran primero definiciones de palabras claves para uso del *framework* propuesto. Este código es generado a partir del archivo de capacidades del microcontrolador.

Por último se tienen las definiciones de las configuraciones de los pins usadas en el arreglo de estructuras descrito en la sección 6.1.1.2. El objetivo de usar definiciones de preprocesador es para que el usuario tenga acceso explícito a la configuración de cada pin expresada con el nombre clave. Estas definiciones son generadas a partir de la información indicada en la ventana de configuración y contenida en el archivo de configuración. Figura 6.2 muestra un ejemplo de definiciones de configuraciones generadas.

```
// ################## Kamino generator v1.0.0: Generated code! ################
// ######## Do NOT modify code between this header and the footer below #######
// USER_BUTTON
#define USER_BUTTON_PORT PORT_C
#define USER_BUTTON_PIN PIN_13
#define USER_BUTTON_MODE MODE_INPUT
#define USER_BUTTON_ALT GPIO_ALT_NONE
#define USER_BUTTON_PULL PULL_UP
#define USER_BUTTON_SPEED SPEED_FAST
#define USER_BUTTON_INIT_OUT LOW

// USER_LED
#define USER_LED_PORT PORT_A
#define USER_LED_PIN PIN_12
#define USER_LED_MODE MODE_OUTPUT
#define USER_LED_ALT GPIO_ALT_NONE
#define USER_LED_PULL PULL_NOT_AVAILABLE
#define USER_LED_SPEED SPEED_FAST
#define USER_LED_INIT_OUT LOW

// ################## Kamino generator v1.0.0: Generated code! ################
// ######## Do NOT modify code between this footer and the header above #######
```

Figura 6.2 Definiciones de características para pins GPIO

6.1.1.2 gpio_cfg.c

En el anexo 10.2.3 se ve la plantilla usada de base para el archivo de configuración **gpio_cfg.c**. Este archivo contiene el arreglo de estructuras de configuración y la mayor parte de su contenido es generado.

Contiene la definición de un arreglo de estructuras tipo `GPIO_Cfg_t` llamado `Gpio_Cfg`. El número de elementos corresponde al número de pins tipo GPIO a

configurar del microcontrolador. La Figura 3.9 muestra los elementos de una estructura tipo `GPIO_Cfg_t`, esta definición se encuentra en el archivo de cabecera gpio.h. La Figura 6.3 muestra un ejemplo de este arreglo de estructuras.

```
const Gpio_cfg_t Gpio_Cfg[GPIO_ELEMENTS_MAX] = {
// ################## Kamino generator v1.0.0: Generated code! ################
// ######## Do NOT modify code between this header and the footer below #######
    {
        USER_BUTTON_PORT,
        USER_BUTTON_PIN,
        USER_BUTTON_MODE,
        USER_BUTTON_ALT,
        USER_BUTTON_PULL,
        USER_BUTTON_SPEED,
        USER_BUTTON_INIT_OUT
    },
    {
        USER_LED_PORT,
        USER_LED_PIN,
        USER_LED_MODE,
        USER_LED_ALT,
        USER_LED_PULL,
        USER_LED_SPEED,
        USER_LED_INIT_OUT
    }
// ################## Kamino generator v1.0.0: Generated code! ################
// ######## Do NOT modify code between this footer and the header above #######
```

Figura 6.3 Arreglo de estructuras de configuración

El arreglo de estructuras contiene las definiciones indicadas en la sección 6.1.1.1 para poder indicar de manera más amigable al usuario a que pin corresponde cada estructura.

6.2 Configuración de archivos de cabecera de proyecto

6.2.1 frameworkCommon.h

En el anexo 10.2.1 se ve la plantilla usada de base para el archivo de cabecera **frameworkCommon.h**. Este archivo contiene definiciones necesarias para todo el proyecto y parte de su contenido es generado.

En la sección de Inclusiones se incluyen las librerías que son necesarias por el microcontrolador del proyecto. Este código es generado a partir de la información en el archivo de capacidades del microcontrolador.

La siguiente sección generada es la de definiciones que son necesarias por el microcontrolador del proyecto. Este código es generado a partir de la información en el archivo de capacidades del microcontrolador.

6.2.2 frameworkIncludes.h

En el anexo 10.2.2 se ve la plantilla usada de base para el archivo de cabecera **frameworkIncludes.h**. Este archivo contiene la inclusión de las librerías necesarias para el proyecto y la mayor parte de su contenido es generado.

En la sección de inclusiones se incluyen las librerías de los módulos del framework que van a ser usados en el proyecto, para evitar incluir innecesariamente módulos que no estén configurados o soportados. Este código es generado a partir de información del proyecto indicando cuales módulos están siendo configurados.

7 VALIDACIÓN

7.1 Prueba de concepto

El objetivo de la prueba de concepto es demostrar la viabilidad de tener un programa con una aplicación que pueda ser ejecutada en al menos 2 arquitecturas de microcontroladores diferentes sin modificación alguna. Esta aplicación hace uso del framework propuesto, el cual es configurado utilizando el generador de código para 2 microcontroladores distintos.

Se eligieron 2 microcontroladores de fabricantes distintos que no compartan la misma arquitectura para poder demostrar la flexibilidad del *framework*, y el generador. Estos microcontroladores son usados dentro de tablillas de evaluación para un manejo más sencillo. La siguiente tabla muestra una comparación rápida entre las 2 tablillas usadas, la sección 7.2 incluye una descripción más detallada.

Tablilla de evaluación	**NUCLEO-F334R8**	**MSP-EXP430FR6989**
Fabricante	ST	Texas Instruments
Microcontrolador	STM32F334R8	MSP430FR6989
Arquitectura	ARM Cortex-M4 32bits	MSP430 16 bits
Reloj principal	72 MHz	16 MHz
RAM	16KB	2KB
Memoria Flash interna	64KB	128KB
Pins	64	100
Pins GPIOs	51	83
Entradas para usuario	1 botón	2 botones
Salidas para usuario	1 LED	2 LEDs 1 pantalla de segmentos

Ambas tablillas de evaluación incluyen su depurador integrado, y pueden ser programadas y depuradas utilizando software gratuito descrito con más detalle en la sección 7.3. Además, ambas tablillas incluyen interfaces con el usuario para comprobar el comportamiento de la aplicación, este comportamiento es descrito en la sección 7.4.

7.2 Microcontroladores

Se seleccionaron 2 microcontroladores populares con relativo bajo costo de 2 fabricantes comerciales. Ambos microcontroladores pertenecen a familias que se utilizan ampliamente en el sector comercial (Como Resideo), así como en el de entusiastas.

7.2.1 STM32F334R8

La familia STM32F3 del fabricante ST contiene microcontroladores de 32 bits basados en la arquitectura ARM Cortex-M4 [63]. El modelo STM32F334R8 es un microcontrolador de propósito general y de bajo costo [54, p. 32]. Para la prueba de concepto se trabaja en una tarjeta de desarrollo modelo NUCLEO-F334R8 [64].

7.2.2 MSP430FR6989

La familia MSP430 del fabricante Texas Instruments contiene microcontroladores de 16 bits basado en una arquitectura RISC. El modelo MSP430FR6989 es un microcontrolador de propósito general y de bajo consumo de corriente [55]. Para la prueba de concepto se trabaja en una tarjeta de desarrollo modelo MSP-EXP430FR6989 [65].

7.3 IDEs

Para la compilación del software generado, programación de las tablillas de evaluación, y depuración se seleccionaron los *IDEs* recomendados por los fabricantes de los microcontroladores. Ambos *IDEs* contienen todas las herramientas necesarias para el desarrollo de software, son gratuitos, y pueden ser ejecutados de manera nativa en los sistemas operativos Linux y Windows. Para esta prueba de concepto ambos *IDEs* son ejecutados en una computadora con sistema operativo Arch Linux [66] usando el kernel 5.1.4-arch1-1-ARCH, el entorno de escritorio es Gnome 3.32.2. La versión de JAVA usada para el generador de código es OpenJDK 10.0.2 de 64b.

7.3.1 Atollic TrueSTUDIO

IDE basado en Eclipse, soporta microcontroladores de la familia STM32. Contiene un compilador, ensamblador, enlazador, y depurador basados en las herramientas de código abierto GCC [67], y GDB [68]. En este *IDE* se pueden generar programas basados en ejemplos provistos por el fabricante, los cuales contienen código de configuración para el reloj interno y otros periféricos necesarios para el funcionamiento del microcontrolador [69]. Para esta prueba se utiliza la versión 9.3.0 para Linux.

7.3.2 Code Composer Studio

IDE basado en Eclipse, soporta una amplia gama de microcontroladores de diversas familias del fabricante Texas Instruments, entre ellas la MSP430. Contiene un compilador, ensamblador, enlazador, y depurador basados en las herramientas de código abierto GCC [67], y GDB [68]. En este *IDE* se pueden generar programas basados en ejemplos provistos por el fabricante, los cuales contienen código de

configuración para el reloj interno y otros periféricos necesarios para el funcionamiento del microcontrolador [70]. Para esta prueba se utiliza la versión 9.0.1.00004 para Linux.

7.4 Aplicación

Como prueba de concepto se decidió utilizar una aplicación sencilla que aprovechara las interfaces en común disponibles en ambas tablillas de evaluación. Tomando como característica mínima común, se utiliza un botón de usuario y un LED de usuario. El comportamiento del programa es simple: Al presionar el botón de usuario se enciende el LED de usuario, de la misma manera, al soltar el botón, se apaga el LED. En el anexo 10.4.1 se muestra el código fuente de la aplicación que se utilizó exactamente igual en los programas para ambas tablillas de evaluación.

Como configuración de los pins GPIO se establecieron los pins en las siguientes configuraciones:

- Botón de usuario: Entrada sin resistencia de polarización.
- LED de usuario: Salida en nivel inicial bajo.
- Otros pins no utilizados: Sin configurar.

Los anexos 10.3.1 y 10.3.2 muestran los archivos de configuración utilizados para los microcontroladores STM32F334R8 y MSP430FR6989 respectivamente.

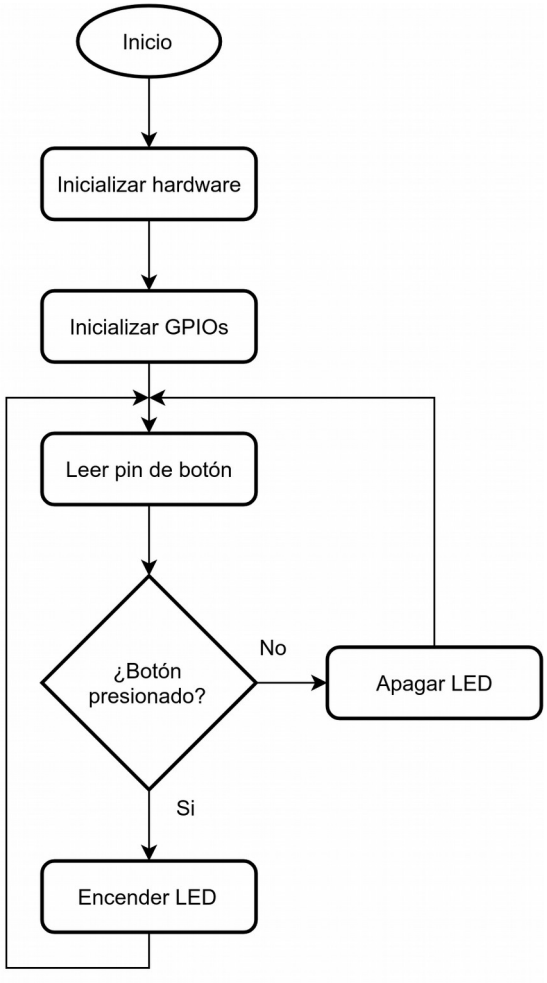

Figura 7.1 Diagrama de flujo de prueba de concepto

La Figura 7.1 muestra el diagrama de flujo para el programa utilizado como prueba de concepto.

Para tener una base del programa se utilizaron ejemplos del fabricante con configuración básica de hardware inicial. El módulo de configuración de relojes queda fuera del alcance de este proyecto.

7.5 Consideraciones

Por cuestiones de tiempo, ciertas funciones se hicieron de forma manual, o se acotó su alcance con respecto al diseño del proyecto planeado a futuro. Los archivos generados se copiaron de manera manual a su carpeta de destino, y falta la interfaz gráfica para especificar una ubicación.

Las capacidades de ambos microcontroladores son muy diferentes, ambos ejecutan código con velocidades de reloj distintas, además de tener conjuntos de instrucciones con arquitecturas particulares. Por esta razón, se eligió un comportamiento de aplicación simple que no sea afectado a un nivel perceptible por un humano.

Los circuitos de ambas tablillas son distintos, por lo que la definición de los niveles físicos de voltaje requeridos para encender el LED de usuario, y representativos de la posición del botón de usuario fueron definidos fuera del archivo de aplicación, normalmente esto sería definido en la capa de abstracción de hardware, en este caso, por razones de simplicidad se definió en el archivo de cabecera main.h. la Figura 7.2 y la Figura 7.3 muestran estas definiciones para ambas tablillas.

```
 // Initialize clocks and HAL
#define HWInit()           \
    MCU_Init(&Mcu_SysCfg); \
    HAL_Init();

#define BUTTON_PRESSED LOW
#define BUTTON_NOT_PRESSED HIGH

#define LED_ON HIGH
#define LED_OFF LOW
```

Figura 7.2 Definiciones de HAL para NUCLEO-F334R8

```
// Stop watchdog timer and disable the
GPIO power-on default high-impedance mode
#define HWInit()              \
    WDTCTL = WDTPW | WDTHOLD; \
    PM5CTL0 &= ~LOCKLPM5;

#define BUTTON_PRESSED LOW
#define BUTTON_NOT_PRESSED HIGH

#define LED_ON HIGH
#define LED_OFF LOW
```

Figura 7.3 Definiciones de HAL para MSP-EXP430FR6989

Otra consideración es la inicialización básica del hardware, dadas las limitaciones de este proyecto, la configuración de módulos como relojes quedan fuera de su alcance, principalmente por su alta complejidad y por ser bastante específicos a los microcontroladores. Por esta razón se definió de manera genérica un macro de preprocesador `HWInit()` para la inicialización de hardware, y se definió su comportamiento de manera particular para cada microcontrolador, estas definiciones se pueden ver en la Figura 7.2 y la Figura 7.3.

Para acortar el tiempo de desarrollo, se hizo uso de librerías para manejo de periféricos provistas por los fabricantes. Para el STM32F334R8 se utilizó la librería *STM32F3xx_HAL_Driver [71]* versión 1.10.0 provista por ST. Para el MSP430FR6989 se utilizó la librería *driverlib MSP430FR5xx_6xx [72]* versión 2.91.11.01 provista por Texas Instruments. Estas librerías proveen los controladores para periféricos de bajo nivel y son utilizadas por los *wrappers* que fueron escritos como traductores para el elemento estático del módulo GPIO.

7.6 Parámetros de prueba

Para poder considerar que la prueba de concepto es exitosa, se establecieron parámetros de falla, si al menos uno de estos se cumple, se considera que la prueba falló.

- El generador de código no puede ser ejecutado, o tiene una falla crítica que provoca el paro de su ejecución.
- El código no puede ser generado exitosamente a partir de los archivos de capacidades.
- La configuración seleccionada no pudo ser almacenada correctamente en un archivo de configuración.
- Al abrir el archivo de configuración previamente generado, la configuración desplegada es diferente a la indicada por el usuario.
- El *framework* muestra errores de compilación o enlazamiento dentro del proyecto.
- El código generado muestra errores de compilación o enlazamiento dentro del proyecto.
- El archivo binario creado no puede ser cargado y depurado en el microcontrolador.
- El comportamiento de la aplicación ejecutada en un microcontrolador no es el esperado.
- El comportamiento de la aplicación en 2 microcontroladores distintos es diferente de manera aparente al usuario.

- Se requiere hacer cambios en la capa de aplicación para conseguir el resultado esperado.

8 CONCLUSIONES

8.1 Documentación

Este proyecto de tesis deja establecida las arquitecturas generales del *framework* propuesto y del generador de código, aunque el alcance para ambas herramientas se limita por el momento al módulo GPIO en 2 microcontroladores, este documento establece la base para expandir el proyecto a otros microcontroladores y otros módulos para control de periféricos.

El código del *framework* se encuentra almacenado en un repositorio con control de versión tipo Git en un servidor del servicio BitBucket [42]. Este repositorio incluye además documentación en una serie de páginas estilo *Wiki [73]* donde se indican las herramientas necesarias para el desarrollo, y la arquitectura general. Esto con el propósito de que el proyecto pueda ser fácilmente retomado y expandido.

El código del generador también se encuentra almacenado en un repositorio de BitBucket. Su documentación es generada automáticamente desde el código fuente por el programa Doxygen [74]. Esta documentación es generada en un archivo PDF y en una serie de páginas html interactivas que pueden ser visualizadas en cualquier explorador web.

8.2 Resultados

Se llevaron a cabo las siguientes operaciones para comprobar los parámetros descritos en la sección 7.6:

1 Se generaron los proyectos base para cada microcontrolador con las librerías de controladores en sus respectivos IDEs.

1.1 Se crearon las definiciones de configuraciones base mínima para el hardware en sus respectivos.

1.2 Se importó el *framework* a cada proyecto.

1.3 Se crearon los *wrappers* para interactuar con cada librería de controladores.

2 Se crearon proyectos para el generador utilizando los archivos de capacidades de cada microcontrolador.

2.1 Se configuró un pin como botón de usuario y otro como LED de usuario.

2.2 Se generó el código para cada microcontrolador.

2.3 Se guardó el archivo de configuración de proyecto y se volvió a abrir para asegurarse de que no se pierda avance.

3 Se agregó el código generado en cada proyecto.

3.1 Se compiló cada proyecto revisando que no existieran errores ni alertas.

4 Se probó la aplicación en cada tarjeta de desarrollo.

4.1 Se utilizó el depurador integrado para cargar el programa y comprobar su comportamiento.

4.2 Se revisó de manera física que el comportamiento fuera el esperado, y que no existiera una diferencia perceptible entre ambas tablillas de evaluación.

Se logró el objetivo de poder tener una aplicación igual línea por línea de código siendo ejecutada en 2 microcontroladores de arquitecturas completamente diferentes, utilizando 2 IDEs distintos con sus compiladores propios. Los elementos estáticos del *framework* también fueron idénticos en ambos proyectos, lo único que tuvo que ser desarrollado a mano fueron los *wrappers* para interactuar con las librerías de controladores. Los elementos dinámicos del *framework* fueron generados en su totalidad.

Con los resultados de esta prueba de concepto se concluyó que la premisa del proyecto es posible, y se prevé que sea escalable para soportar más familias de microcontroladores en distintos compiladores, y en un futuro soportar además otros periféricos de hardware.

8.3 Pasos siguientes

Estos son los pasos siguientes que se han identificado para poder continuar con el proyecto hasta tener un producto que pueda ser utilizado de forma entusiasta y profesional que por razones de tiempo no fue posible completar en esta fase del proyecto.

- Expandir los periféricos soportados tanto por el *framework*, como por el generador de código.
    - Se considera que los periféricos más sencillos son los de comunicación serial, como UART, I^2C, SPI, etc.
    - Se considera que el módulo más retador es el de los relojes del microcontrolador, al ser altamente específico a la arquitectura utilizada.

- Expandir la selección de microcontroladores soportados creando nuevos archivos de capacidades y *wrappers*, preferiblemente microcontroladores de arquitecturas distintas para seguir comprobando el concepto.
- Implementar las características mínimas del generador para hacer la experiencia con el usuario más amigable.
    - Selector de microcontrolador de acuerdo a los archivos XML existentes en la carpeta de instalación.
    - Configurador gráfico de características del proyecto.
        - Microcontrolador.
        - Ubicación de carpeta de proyecto.
        - Nombre del proyecto.
    - Copiado automático de capas de controladores y *wrappers* a la carpeta destino del código generado según el microcontrolador seleccionado.

9 BIBLIOGRAFÍA

[1] PRNewswire, “Transparency Market Research”, *Global Consumer Electronics Market to Expand at 4.0% CAGR by 2022 Owing to Growth of Online Retail Industry: Transparency Market Research*, feb-2016. [En línea]. Disponible en: http://www.prnewswire.com/news-releases/global-consumer-electronics-market-to-expand-at-40-cagr-by-2022-owing-to-growth-of-online-retail-industry-transparency-market-research-567217721.html. [Consultado: 01-dic-2017].

[2] T. Noergaard, *Embedded Systems Architecture*, 2a ed. Newnes, 2005.

[3] K. Hoganson, *Concepts In Computing*. Jones & Bartlett Learning, 2007.

[4] C. Ebert y C. Jones, “Embedded software: Facts, figures and future”. IEEE Computer Society, 2009.

[5] J. Motavalli, “The Dozens of Computers That Make Modern Cars Go (and Stop)”, feb-2010. [En línea]. Disponible en: http://www.nytimes.com/2010/02/05/technology/05electronics.html.

[6] P. Christensson, “Framework Definition”, mar-2013. [En línea]. Disponible en: https://techterms.com/definition/framework.

[7] M. Baker, “What is a Software Framework? And why should you like ’em?”, may-2009. [En línea]. Disponible en: http://info.cimetrix.com/blog/bid/22339/What-is-a-Software-Framework-And-why-should-you-like-em.

[8] Microchip, “Advanced Software Framework (ASF)”, jun-2017. [En línea]. Disponible en: http://www.microchip.com/development-tools/atmel-studio-7/advanced-software-framework-%28asf%29. [Consultado: 01-dic-2017].

[9] J. Cornell, “Honeywell Completes Spin-Off Of Resideo Technologies”, *Forbes*. [En línea]. Disponible en: https://www.forbes.com/sites/joecornell/2018/11/02/honeywell-completes-spin-off-of-resideo-technologies/. [Consultado: 18-may-2019].

[10] “T9 Smart Thermostat | Honeywell Home”. [En línea]. Disponible en: https://t9.honeywellhome.com/. [Consultado: 25-may-2019].

[11] “RedLINK Platform — Honeywell Home | Forward Thinking”. [En línea]. Disponible en: https://forwardthinking.honeywellhome.com/redlink. [Consultado: 25-may-2019].

[12] M. R. Nigro, J. A. Castro, y F. H. Torres, “Generación Automática de Código a Partir de Máquinas de Estado Finito”, *Comput. Sist. Vol 14*, jun. 2011.

[13] S. L. Jim, “From UML diagrams to behavioural source code”, *Cent. Voor Wiskd. En Inform.*, sep. 2006.

[14] Renesas, “AP4, Applilet”, jun-2017. [En línea]. Disponible en: https://www.renesas.com/en-us/products/software-tools/tools/code-generator/ap4-application-leading-tool-applilet.html. [Consultado: 01-dic-2017].

[15] Microchip, “Atmel Start”, jun-2017. [En línea]. Disponible en: http://start.atmel.com/. [Consultado: 01-dic-2017].

[16] Warley Design Solutions, “How long does product development take?”, jun-2017. [En línea]. Disponible en: http://www.warleydesign.co.uk/how-long-does-product-development-take/.

BIBLIOGRAFÍA

[17] H. Bauer y O. Burkacky, *When software meets hardware: Excellence in embedded-software development*. 2013.

[18] W. Maisel, M. Sweeney, W. Stevenson, K. Ellison, y L. Epstein, “Recalls and safety Alerts Involving Pacemakers and Implantable Cardioverter-Defibrillator Generatos”, *American Medical Association*, 2001.

[19] S. Cass, “The 2017 Top Programming Languages”, *IEEE Spectrum: Technology, Engineering, and Science News*, 18-jul-2017. [En línea]. Disponible en: https://spectrum.ieee.org/computing/software/the-2017-top-programming-languages. [Consultado: 06-dic-2017].

[20] “PYPL PopularitY of Programming Language index”. [En línea]. Disponible en: http://pypl.github.io/PYPL.html. [Consultado: 06-dic-2017].

[21] A. Binstock, “OracleVoice: Java’s 20 Years Of Innovation”, *Forbes*. [En línea]. Disponible en: https://www.forbes.com/sites/oracle/2015/05/20/javas-20-years-of-innovation/. [Consultado: 06-dic-2017].

[22] “What are the system requirements for Java?” [En línea]. Disponible en: https://www.java.com/en/download/help/sysreq.xml. [Consultado: 01-dic-2017].

[23] “Python (programming language)”, *Wikipedia*. 04-dic-2017.

[24] “Python Operating Systems List”, *Python.org*. [En línea]. Disponible en: https://www.python.org/downloads/operating-systems/. [Consultado: 01-dic-2017].

[25] “C Sharp (programming language)”, *Wikipedia*. 08-dic-2017.

[26] “Supported Platforms | Mono”. [En línea]. Disponible en: http://www.mono-project.com/docs/about-mono/supported-platforms/. [Consultado: 01-dic-2017].

[27] “Integrated development environment”, *Wikipedia*. 21-nov-2017.

[28] “List of Eclipse-based software - Wikipedia”. [En línea]. Disponible en: https://en.wikipedia.org/wiki/List_of_Eclipse-based_software. [Consultado: 01-dic-2017].

[29] “Eclipse desktop & web IDEs”. [En línea]. Disponible en: https://eclipse.org/ide/. [Consultado: 01-dic-2017].

[30] “Eclipse Public License - Version 1.0”, *Eclipse Public License - v 1.0*. [En línea]. Disponible en: http://www.eclipse.org/legal/epl-v10.html. [Consultado: 01-dic-2017].

[31] “NetBeans IDE - Overview”. [En línea]. Disponible en: https://netbeans.org/features/index.html. [Consultado: 08-dic-2017].

[32] “gnu.org”. [En línea]. Disponible en: https://www.gnu.org/licenses/old-licenses/gpl-2.0.en.html. [Consultado: 01-dic-2017].

[33] K. Lewallen, “MICROSOFT VISUAL STUDIO COMMUNITY 2017”, *Visual Studio*, 13-ene-2017. .

[34] “Pricing and Purchasing Options | Visual Studio”. [En línea]. Disponible en: https://www.visualstudio.com/vs/pricing/. [Consultado: 01-dic-2017].

[35] Atlassian, “What is version control | Atlassian Git Tutorial”, *Atlassian*. [En línea]. Disponible en: https://www.atlassian.com/git/tutorials/what-is-version-control. [Consultado: 21-mar-2018].

[36] “Apache Licenses”. [En línea]. Disponible en: https://www.apache.org/licenses/. [Consultado: 21-mar-2018].

BIBLIOGRAFÍA

[37] "RiouxSVN — Free, Private Subversion Hosting". [En línea]. Disponible en: https://riouxsvn.com/. [Consultado: 21-mar-2018].

[38] "Helix TeamHub Code Repository & Code Review Tools | Perforce". [En línea]. Disponible en: https://www.perforce.com/products/helix-teamhub. [Consultado: 21-mar-2018].

[39] "XP-Dev.com - Enterprise grade Subversion, Git, Mercurial & Trac Hosting". [En línea]. Disponible en: https://xp-dev.com/. [Consultado: 21-mar-2018].

[40] "Git". [En línea]. Disponible en: https://git-scm.com/. [Consultado: 25-may-2019].

[41] "Build software better, together", *GitHub*. [En línea]. Disponible en: https://github.com. [Consultado: 25-may-2019].

[42] "Bitbucket - Features | Atlassian". [En línea]. Disponible en: https://www.atlassian.com/software/bitbucket/features. [Consultado: 21-mar-2018].

[43] "XML", *Wikipedia*. 29-abr-2019.

[44] "What is XML? What Opens a XML? File Format List from WhatIs.com". [En línea]. Disponible en: http://whatis.techtarget.com/fileformat/XML-eXtensible-markup-language. [Consultado: 01-dic-2017].

[45] "What's the Difference Between JSON, XML, and YAML?", *Electronic Design*, 28-abr-2015. [En línea]. Disponible en: http://www.electronicdesign.com/dev-tools/whats-difference-between-json-xml-and-yaml. [Consultado: 01-dic-2017].

[46] "Extensible Markup Language (XML) 1.0 (Fifth Edition)". [En línea]. Disponible en: https://www.w3.org/TR/REC-xml/#sec-origin-goals. [Consultado: 01-dic-2017].

[47] "JSON", *Wikipedia*. 07-may-2019.

[48] "What is JSON?", *Squarespace*. [En línea]. Disponible en: https://developers.squarespace.com/. [Consultado: 01-dic-2017].

[49] "YAML", *Wikipedia*. 16-may-2019.

[50] "YAML Ain't Markup Language (YAML™) Version 1.2". [En línea]. Disponible en: http://yaml.org/spec/1.2/spec.html#id2759572. [Consultado: 01-dic-2017].

[51] "Compare Repositories - Open Hub". [En línea]. Disponible en: https://www.openhub.net/repositories/compare. [Consultado: 30-may-2019].

[52] "FreeRTOS - Market leading RTOS (Real Time Operating System) for embedded systems with Internet of Things extensions". [En línea]. Disponible en: https://www.freertos.org/. [Consultado: 30-may-2019].

[53] "General Purpose Input/Output (GPIO) Introduction — The Linux Kernel documentation", *General Purpose Input/Output (GPIO)*. [En línea]. Disponible en: https://www.kernel.org/doc/html/latest/driver-api/gpio/intro.html. [Consultado: 07-may-2019].

[54] "STM32F334R8 - Mainstream Mixed signals MCUs ARM Cortex-M4 core with DSP and FPU, 64 Kbytes Flash, 72 MHz CPU, CCM, 12-bit ADC 5 MSPS, comparators, op-amp, hr timer - STMicroelectronics". [En línea]. Disponible en: https://www.st.com/en/microcontrollers-microprocessors/stm32f334r8.html. [Consultado: 21-mar-2019].

[55] "MSP430FR6989 MSP430FR6989 16 MHz ULP Microcontroller - 128 KB FRAM, 2KB SRAM, 83 IO, ADC12, LCD, AES, Scan IF | TI.com". [En línea]. Disponible en: http://www.ti.com/product/MSP430FR6989. [Consultado: 21-mar-2019].

BIBLIOGRAFÍA

[56] U. Nanda y S. K. Pattnaik, “Universal Asynchronous Receiver and Transmitter (UART)”, en *2016 3rd International Conference on Advanced Computing and Communication Systems (ICACCS)*, 2016, vol. 01, pp. 1–5.

[57] NXP Semiconductors, “I2C-bus specification and user manual”. 04-abr-2014.

[58] Motorola, Inc., “SPI Block Guide V03.06”. 04-feb-2003.

[59] “List of planets”, *Wookieepedia*. [En línea]. Disponible en: https://starwars.fandom.com/wiki/List_of_planets. [Consultado: 25-mar-2019].

[60] “What is Java Swing? - Definition from Techopedia”, *Techopedia.com*. [En línea]. Disponible en: https://www.techopedia.com/definition/26102/java-swing. [Consultado: 25-mar-2019].

[61] “Java Swing Tutorial - javatpoint”, *www.javatpoint.com*. [En línea]. Disponible en: https://www.javatpoint.com/java-swing. [Consultado: 25-mar-2019].

[62] “GNOME 3 – GNOME”. .

[63] “Microcontrollers - STM32 Arm Cortex MCUs - STMicroelectronics”, *STM32 32-bit Arm Cortex MCUs*. [En línea]. Disponible en: https://www.st.com/en/microcontrollers-microprocessors/stm32-32-bit-arm-cortex-mcus.html. [Consultado: 09-may-2019].

[64] “NUCLEO-F334R8 - STM32 Nucleo-64 development board with STM32F334R8 MCU, supports Arduino and ST morpho connectivity - STMicroelectronics”. [En línea]. Disponible en: https://www.st.com/en/evaluation-tools/nucleo-f334r8.html. [Consultado: 13-may-2019].

[65] “MSP-EXP430FR6989 MSP430FR6989 LaunchPad Development Kit | TI.com”. [En línea]. Disponible en: http://www.ti.com/tool/msp-exp430fr6989. [Consultado: 13-may-2019].

[66] “Arch Linux - ArchWiki”. [En línea]. Disponible en: https://wiki.archlinux.org/index.php/Arch_Linux. [Consultado: 18-may-2019].

[67] “GCC, the GNU Compiler Collection - GNU Project - Free Software Foundation (FSF)”. [En línea]. Disponible en: https://gcc.gnu.org/. [Consultado: 14-may-2019].

[68] “GDB: The GNU Project Debugger”. [En línea]. Disponible en: https://www.gnu.org/software/gdb/. [Consultado: 14-may-2019].

[69] “TrueSTUDIO - Atollic - ST”. [En línea]. Disponible en: https://atollic.com/truestudio/. [Consultado: 13-may-2019].

[70] “CCSTUDIO Code Composer Studio (CCS) Integrated Development Environment (IDE) | TI.com”. [En línea]. Disponible en: http://www.ti.com/tool/CCSTUDIO. [Consultado: 13-may-2019].

[71] ST, “Description of STM32F3xx HAL drivers”. ST, feb-2015.

[72] Texas Instruments, “MSP430 DriverLib for MSP430FR5xx6xx Devices”. Texas Instruments, may-2017.

[73] “Wiki”, *Wikipedia*. 16-may-2019.

[74] “Doxygen: Main Page”. [En línea]. Disponible en: http://www.doxygen.nl/. [Consultado: 16-may-2019].

10 ANEXOS

10.1 Framework

10.1.1 gpio.c

```
/**
 * @file gpio.c
 * @author Miguel Diaz
 * @brief GPIO framework module
 */

/************************
 * Includes             *
 ************************/
#include "gpio.h"

/************************
 * Type Definitions     *
 ************************/

/************************
 * Macros               *
 ************************/

/************************
 * Defines              *
 ************************/

/************************
 * Constants            *
 ************************/

/************************
 * Calibrations         *
 ************************/

/************************
 * Global Variables     *
 ************************/

/************************
 * File Scope Variables *
 ************************/

/************************
 * Function Prototypes  *
 ************************/

/************************
 * Function Definitions *
 ************************/
```

ANEXOS

```
/**
 * @brief Initialize GPIO pin
 * @param cfgPtr pin configuration structure
 */
void Gpio_Init(const Gpio_cfg_t* cfgPtr)
{
    if (NULL != cfgPtr)
    {
        for (uint8_t gpioNum = 0; gpioNum < GPIO_ELEMENTS_MAX; gpioNum++)
        {
            GpioWrapper_setMode(cfgPtr[gpioNum].Port, cfgPtr[gpioNum].Pin,
cfgPtr[gpioNum].Mode);
            GpioWrapper_setPull(cfgPtr[gpioNum].Port, cfgPtr[gpioNum].Pin,
cfgPtr[gpioNum].Pull);
            GpioWrapper_setSpeed(cfgPtr[gpioNum].Port, cfgPtr[gpioNum].Pin,
cfgPtr[gpioNum].Speed);

            if (cfgPtr[gpioNum].Mode == MODE_OUTPUT)
            {
                Gpio_SetPin(cfgPtr[gpioNum].Port, cfgPtr[gpioNum].Pin,
cfgPtr[gpioNum].InitOutValue);
            }
        }
    }
}

/**
 * @brief Write value to port
 * @param port Port to be written
 * @param value Value to be written to port
 */
void Gpio_WritePort(Gpio_portId_t port, Gpio_data_t value)
{
    GpioWrapper_WritePort(port, value);
}

/**
 * @brief Write value to GPIO pin
 * @param port Pin's port
 * @param pin Pin to be written
 * @param state Value to be written to pin
 */
void Gpio_SetPin(Gpio_portId_t port, Gpio_pinId_t pin, Gpio_pinState_t state)
{
    GpioWrapper_SetPin(port, pin, state);
}

/**
 * @brief Read port's value
 * @param port Pin's port
 * @return Port's value
 */
Gpio_data_t Gpio_ReadPort(Gpio_portId_t port)
{
    return GpioWrapper_ReadPort(port);
}

/**
 * @brief Read GPIO pin's value
```

```
 * @param port Pin's port
 * @param pin Pin to be read
 * @return Pin's value
 */
Gpio_pinState_t Gpio_GetPin(Gpio_portId_t port, Gpio_pinId_t pin)
{
    return GpioWrapper_GetPin(port, pin);
}
```

10.1.2 gpio.h

```
/**
 * @file gpio.h
 * @author Miguel Diaz
 * @brief GPIO framework module header
 */

/************************
 * Guard                *
 ************************/
#ifndef _GPIO_H_
#define _GPIO_H_

/************************
 * Includes             *
 ************************/
#include "frameworkCommon.h"
#include "gpio_cfg.h"
#include "gpio_wrapper.h"

/************************
 * Public Types         *
 ************************/

/**
 * @brief GPIO configuration structure
 */
typedef struct
{
    Gpio_portId_t   Port;
    Gpio_pin_t      Pin;
    Gpio_mode_t     Mode;
    Gpio_alt_t      Alternate;
    Gpio_pull_t     Pull;
    Gpio_speed_t    Speed;
    Gpio_pinState_t InitOutValue;
} Gpio_cfg_t;

/************************
 * Public Macros        *
 ************************/

/************************
 * Public Defines       *
 ************************/

/************************
 * Public Constants     *
 ************************/
```

```
/************************
 * Public Calibrations  *
 ************************/

/************************
 * Public Variables     *
 ************************/
extern const Gpio_cfg_t Gpio_Cfg[GPIO_ELEMENTS_MAX];

/************************
 * Public Functions     *
 ************************/
void            Gpio_Init(const Gpio_cfg_t* cfgPtr);
void            Gpio_WritePort(Gpio_portId_t port, Gpio_data_t value);
void            Gpio_SetPin(Gpio_portId_t port, Gpio_pinId_t pin, Gpio_pinState_t state);
Gpio_data_t     Gpio_ReadPort(Gpio_portId_t port);
Gpio_pinState_t Gpio_GetPin(Gpio_portId_t port, Gpio_pinId_t pin);

#endif /* _GPIO_H_ */
```

10.2 Plantillas

10.2.1 frameworkCommon.h

```
/**
 * @file frameworkCommon.h
 * @author Miguel Diaz
 * @brief framework common header
 */

/************************
 * Guard                *
 ************************/
#ifndef _FRAMEWORK_COMMON_H_
#define _FRAMEWORK_COMMON_H_

/************************
 * Includes             *
 ************************/
FWK_COMMON_INCLUDES

/************************
 * Public Macros        *
 ************************/
#define FRAMEWORK_VERSION_MAJOR 0
#define FRAMEWORK_VERSION_MINOR 1
#define FRAMEWORK_VERSION_PATCH 0

/************************
 * Public Defines       *
 ************************/

// Bit/Byte related definitions
#define SHIFT_8_BITS (8)
#define LOW_BYTE_IN_16B_MASK (0x00FF)
#define HIGH_BYTE_IN_16B_MASK (0xFF00)
```

```
FWK_GPIO_COMMON_DEFINITIONS

#endif /* _FRAMEWORK_COMMON_H_ */
```

10.2.2 FrameworkIncludes.h

```
/**
 * @file frameworkIncludes.h
 * @author Miguel Diaz
 * @brief Framework main header
 *
 * This header includes all framework modules to be used in the project
 */

/************************
 * Guard                *
 ************************/
#ifndef _FRAMEWORK_INCLUDES_H_
#define _FRAMEWORK_INCLUDES_H_

/************************
 * Includes             *
 ************************/
// Include framework modules depending on what you are using:
FWK_MODULES_INCLUDES

/************************
 * Public Types         *
 ************************/

/************************
 * Public Macros        *
 ************************/

/************************
 * Public Defines       *
 ************************/

/************************
 * Public Constants     *
 ************************/

/************************
 * Public Calibrations  *
 ************************/

/************************
 * Public Variables     *
 ************************/

/************************
 * Public Functions     *
 ************************/

#endif /* _FRAMEWORK_INCLUDES_H_ */
```

10.2.3 gpio_cfg.c

```
/**
 * @file gpio_cfg.c
 * @author Miguel Diaz
 * @brief GPIO gramework configuration
 */

/************************
 * Includes              *
 ************************/
#include "gpio.h"

/************************
 * Type Definitions     *
 ************************/

/************************
 * Macros                *
 ************************/

/************************
 * Defines               *
 ************************/

/************************
 * Constants             *
 ************************/

/************************
 * Calibrations          *
 ************************/

/************************
 * Global Variables      *
 ************************/

/**
 * @brief GPIO initial configuration structures array
 */
const Gpio_cfg_t Gpio_Cfg[GPIO_ELEMENTS_MAX] = {FWK_GPIO_CFG_ARRAY};
/************************
 * File Scope Variables *
 ************************/

/************************
 * Function Prototypes  *
 ************************/

/************************
 * Function Definitions *
 ************************/
```

10.2.4 gpio_cfg.h

```
/**
 * @file gpio_cfg.c
 * @author Miguel Diaz
```

```
 * @brief GPIO gramework configuration
 */

/************************
 * Includes             *
 ************************/
#include "gpio.h"

/************************
 * Type Definitions     *
 ************************/

/************************
 * Macros               *
 ************************/

/************************
 * Defines              *
 ************************/

/************************
 * Constants            *
 ************************/

/************************
 * Calibrations         *
 ************************/

/************************
 * Global Variables     *
 ************************/

/**
 * @brief GPIO initial configuration structures array
 */
const Gpio_cfg_t Gpio_Cfg[GPIO_ELEMENTS_MAX] = {FWK_GPIO_CFG_ARRAY};
/************************
 * File Scope Variables *
 ************************/

/************************
 * Function Prototypes  *
 ************************/

/************************
 * Function Definitions *
 ************************/
```

10.3 Archivos de configuración de proyecto

10.3.1 STM32F334R8

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<Microcontroller_Configuration>
    <pin>PC13
        <name>PC13</name>
        <port>PORT_C</port>
```

```
    <selected>YES</selected>
    <Mode>MODE_INPUT</Mode>
    <OutType>OTYPE_PUSH_PULL</OutType>
    <OutLevel>LOW</OutLevel>
    <Pull>PULL_UP</Pull>
    <Speed>SPEED_FAST</Speed>
    <codeName>USER_BUTTON</codeName>
</pin>
<pin>PC14
    <name>PC14</name>
    <port>PORT_C</port>
    <selected>NOT</selected>
    <Mode>MODE_INPUT</Mode>
    <OutType>OTYPE_PUSH_PULL</OutType>
    <OutLevel>LOW</OutLevel>
    <Pull>PULL_NOT_AVAILABLE</Pull>
    <Speed>SPEED_FAST</Speed>
    <codeName>PC14</codeName>
</pin>
<pin>PC15
    <name>PC15</name>
    <port>PORT_C</port>
    <selected>NOT</selected>
    <Mode>MODE_INPUT</Mode>
    <OutType>OTYPE_PUSH_PULL</OutType>
    <OutLevel>LOW</OutLevel>
    <Pull>PULL_NOT_AVAILABLE</Pull>
    <Speed>SPEED_FAST</Speed>
    <codeName>PC15</codeName>
</pin>
<pin>PF0
    <name>PF0</name>
    <port>PORT_F</port>
    <selected>NOT</selected>
    <Mode>MODE_INPUT</Mode>
    <OutType>OTYPE_PUSH_PULL</OutType>
    <OutLevel>LOW</OutLevel>
    <Pull>PULL_NOT_AVAILABLE</Pull>
    <Speed>SPEED_FAST</Speed>
    <codeName>PF0</codeName>
</pin>
<pin>PF1
    <name>PF1</name>
    <port>PORT_F</port>
    <selected>NOT</selected>
    <Mode>MODE_INPUT</Mode>
    <OutType>OTYPE_PUSH_PULL</OutType>
    <OutLevel>LOW</OutLevel>
    <Pull>PULL_NOT_AVAILABLE</Pull>
    <Speed>SPEED_FAST</Speed>
    <codeName>PF1</codeName>
</pin>
<pin>PC0
    <name>PC0</name>
    <port>PORT_C</port>
    <selected>NOT</selected>
    <Mode>MODE_INPUT</Mode>
    <OutType>OTYPE_PUSH_PULL</OutType>
    <OutLevel>LOW</OutLevel>
    <Pull>PULL_NOT_AVAILABLE</Pull>
```

```
    <Speed>SPEED_FAST</Speed>
    <codeName>PC0</codeName>
</pin>
<pin>PC1
    <name>PC1</name>
    <port>PORT_C</port>
    <selected>NOT</selected>
    <Mode>MODE_INPUT</Mode>
    <OutType>OTYPE_PUSH_PULL</OutType>
    <OutLevel>LOW</OutLevel>
    <Pull>PULL_NOT_AVAILABLE</Pull>
    <Speed>SPEED_FAST</Speed>
    <codeName>PC1</codeName>
</pin>
<pin>PC2
    <name>PC2</name>
    <port>PORT_C</port>
    <selected>NOT</selected>
    <Mode>MODE_INPUT</Mode>
    <OutType>OTYPE_PUSH_PULL</OutType>
    <OutLevel>LOW</OutLevel>
    <Pull>PULL_NOT_AVAILABLE</Pull>
    <Speed>SPEED_FAST</Speed>
    <codeName>PC2</codeName>
</pin>
<pin>PC3
    <name>PC3</name>
    <port>PORT_C</port>
    <selected>NOT</selected>
    <Mode>MODE_INPUT</Mode>
    <OutType>OTYPE_PUSH_PULL</OutType>
    <OutLevel>LOW</OutLevel>
    <Pull>PULL_NOT_AVAILABLE</Pull>
    <Speed>SPEED_FAST</Speed>
    <codeName>PC3</codeName>
</pin>
<pin>PA0
    <name>PA0</name>
    <port>PORT_A</port>
    <selected>NOT</selected>
    <Mode>MODE_INPUT</Mode>
    <OutType>OTYPE_PUSH_PULL</OutType>
    <OutLevel>LOW</OutLevel>
    <Pull>PULL_NOT_AVAILABLE</Pull>
    <Speed>SPEED_FAST</Speed>
    <codeName>PA0</codeName>
</pin>
<pin>PA1
    <name>PA1</name>
    <port>PORT_A</port>
    <selected>NOT</selected>
    <Mode>MODE_INPUT</Mode>
    <OutType>OTYPE_PUSH_PULL</OutType>
    <OutLevel>LOW</OutLevel>
    <Pull>PULL_NOT_AVAILABLE</Pull>
    <Speed>SPEED_FAST</Speed>
    <codeName>PA1</codeName>
</pin>
<pin>PA2
    <name>PA2</name>
```

```
    <port>PORT_A</port>
    <selected>NOT</selected>
    <Mode>MODE_INPUT</Mode>
    <OutType>OTYPE_PUSH_PULL</OutType>
    <OutLevel>LOW</OutLevel>
    <Pull>PULL_NOT_AVAILABLE</Pull>
    <Speed>SPEED_FAST</Speed>
    <codeName>PA2</codeName>
</pin>
<pin>PA3
    <name>PA3</name>
    <port>PORT_A</port>
    <selected>NOT</selected>
    <Mode>MODE_INPUT</Mode>
    <OutType>OTYPE_PUSH_PULL</OutType>
    <OutLevel>LOW</OutLevel>
    <Pull>PULL_NOT_AVAILABLE</Pull>
    <Speed>SPEED_FAST</Speed>
    <codeName>PA3</codeName>
</pin>
<pin>PA4
    <name>PA4</name>
    <port>PORT_A</port>
    <selected>NOT</selected>
    <Mode>MODE_INPUT</Mode>
    <OutType>OTYPE_PUSH_PULL</OutType>
    <OutLevel>LOW</OutLevel>
    <Pull>PULL_NOT_AVAILABLE</Pull>
    <Speed>SPEED_FAST</Speed>
    <codeName>PA4</codeName>
</pin>
<pin>PA5
    <name>PA5</name>
    <port>PORT_A</port>
    <selected>NOT</selected>
    <Mode>MODE_INPUT</Mode>
    <OutType>OTYPE_PUSH_PULL</OutType>
    <OutLevel>LOW</OutLevel>
    <Pull>PULL_NOT_AVAILABLE</Pull>
    <Speed>SPEED_FAST</Speed>
    <codeName>PA5</codeName>
</pin>
<pin>PA6
    <name>PA6</name>
    <port>PORT_A</port>
    <selected>NOT</selected>
    <Mode>MODE_INPUT</Mode>
    <OutType>OTYPE_PUSH_PULL</OutType>
    <OutLevel>LOW</OutLevel>
    <Pull>PULL_NOT_AVAILABLE</Pull>
    <Speed>SPEED_FAST</Speed>
    <codeName>PA6</codeName>
</pin>
<pin>PA7
    <name>PA7</name>
    <port>PORT_A</port>
    <selected>NOT</selected>
    <Mode>MODE_INPUT</Mode>
    <OutType>OTYPE_PUSH_PULL</OutType>
    <OutLevel>LOW</OutLevel>
```

```
    <Pull>PULL_NOT_AVAILABLE</Pull>
    <Speed>SPEED_FAST</Speed>
    <codeName>PA7</codeName>
</pin>
<pin>PC4
    <name>PC4</name>
    <port>PORT_C</port>
    <selected>NOT</selected>
    <Mode>MODE_INPUT</Mode>
    <OutType>OTYPE_PUSH_PULL</OutType>
    <OutLevel>LOW</OutLevel>
    <Pull>PULL_NOT_AVAILABLE</Pull>
    <Speed>SPEED_FAST</Speed>
    <codeName>PC4</codeName>
</pin>
<pin>PC5
    <name>PC5</name>
    <port>PORT_C</port>
    <selected>NOT</selected>
    <Mode>MODE_INPUT</Mode>
    <OutType>OTYPE_PUSH_PULL</OutType>
    <OutLevel>LOW</OutLevel>
    <Pull>PULL_NOT_AVAILABLE</Pull>
    <Speed>SPEED_FAST</Speed>
    <codeName>PC5</codeName>
</pin>
<pin>PB0
    <name>PB0</name>
    <port>PORT_B</port>
    <selected>NOT</selected>
    <Mode>MODE_INPUT</Mode>
    <OutType>OTYPE_PUSH_PULL</OutType>
    <OutLevel>LOW</OutLevel>
    <Pull>PULL_NOT_AVAILABLE</Pull>
    <Speed>SPEED_FAST</Speed>
    <codeName>PB0</codeName>
</pin>
<pin>PB1
    <name>PB1</name>
    <port>PORT_B</port>
    <selected>NOT</selected>
    <Mode>MODE_INPUT</Mode>
    <OutType>OTYPE_PUSH_PULL</OutType>
    <OutLevel>LOW</OutLevel>
    <Pull>PULL_NOT_AVAILABLE</Pull>
    <Speed>SPEED_FAST</Speed>
    <codeName>PB1</codeName>
</pin>
<pin>PB2
    <name>PB2</name>
    <port>PORT_B</port>
    <selected>NOT</selected>
    <Mode>MODE_INPUT</Mode>
    <OutType>OTYPE_PUSH_PULL</OutType>
    <OutLevel>LOW</OutLevel>
    <Pull>PULL_NOT_AVAILABLE</Pull>
    <Speed>SPEED_FAST</Speed>
    <codeName>PB2</codeName>
</pin>
<pin>PB10
```

```
    <name>PB10</name>
    <port>PORT_B</port>
    <selected>NOT</selected>
    <Mode>MODE_INPUT</Mode>
    <OutType>OTYPE_PUSH_PULL</OutType>
    <OutLevel>LOW</OutLevel>
    <Pull>PULL_NOT_AVAILABLE</Pull>
    <Speed>SPEED_FAST</Speed>
    <codeName>PB10</codeName>
</pin>
<pin>PB11
    <name>PB11</name>
    <port>PORT_B</port>
    <selected>NOT</selected>
    <Mode>MODE_INPUT</Mode>
    <OutType>OTYPE_PUSH_PULL</OutType>
    <OutLevel>LOW</OutLevel>
    <Pull>PULL_NOT_AVAILABLE</Pull>
    <Speed>SPEED_FAST</Speed>
    <codeName>PB11</codeName>
</pin>
<pin>PB12
    <name>PB12</name>
    <port>PORT_B</port>
    <selected>NOT</selected>
    <Mode>MODE_INPUT</Mode>
    <OutType>OTYPE_PUSH_PULL</OutType>
    <OutLevel>LOW</OutLevel>
    <Pull>PULL_NOT_AVAILABLE</Pull>
    <Speed>SPEED_FAST</Speed>
    <codeName>PB12</codeName>
</pin>
<pin>PB13
    <name>PB13</name>
    <port>PORT_B</port>
    <selected>NOT</selected>
    <Mode>MODE_INPUT</Mode>
    <OutType>OTYPE_PUSH_PULL</OutType>
    <OutLevel>LOW</OutLevel>
    <Pull>PULL_NOT_AVAILABLE</Pull>
    <Speed>SPEED_FAST</Speed>
    <codeName>PB13</codeName>
</pin>
<pin>PB14
    <name>PB14</name>
    <port>PORT_B</port>
    <selected>NOT</selected>
    <Mode>MODE_INPUT</Mode>
    <OutType>OTYPE_PUSH_PULL</OutType>
    <OutLevel>LOW</OutLevel>
    <Pull>PULL_NOT_AVAILABLE</Pull>
    <Speed>SPEED_FAST</Speed>
    <codeName>PB14</codeName>
</pin>
<pin>PB15
    <name>PB15</name>
    <port>PORT_B</port>
    <selected>NOT</selected>
    <Mode>MODE_INPUT</Mode>
    <OutType>OTYPE_PUSH_PULL</OutType>
```

```
    <OutLevel>LOW</OutLevel>
    <Pull>PULL_NOT_AVAILABLE</Pull>
    <Speed>SPEED_FAST</Speed>
    <codeName>PB15</codeName>
</pin>
<pin>PC6
    <name>PC6</name>
    <port>PORT_C</port>
    <selected>NOT</selected>
    <Mode>MODE_INPUT</Mode>
    <OutType>OTYPE_PUSH_PULL</OutType>
    <OutLevel>LOW</OutLevel>
    <Pull>PULL_NOT_AVAILABLE</Pull>
    <Speed>SPEED_FAST</Speed>
    <codeName>PC6</codeName>
</pin>
<pin>PC7
    <name>PC7</name>
    <port>PORT_C</port>
    <selected>NOT</selected>
    <Mode>MODE_INPUT</Mode>
    <OutType>OTYPE_PUSH_PULL</OutType>
    <OutLevel>LOW</OutLevel>
    <Pull>PULL_NOT_AVAILABLE</Pull>
    <Speed>SPEED_FAST</Speed>
    <codeName>PC7</codeName>
</pin>
<pin>PC8
    <name>PC8</name>
    <port>PORT_C</port>
    <selected>NOT</selected>
    <Mode>MODE_INPUT</Mode>
    <OutType>OTYPE_PUSH_PULL</OutType>
    <OutLevel>LOW</OutLevel>
    <Pull>PULL_NOT_AVAILABLE</Pull>
    <Speed>SPEED_FAST</Speed>
    <codeName>PC8</codeName>
</pin>
<pin>PC9
    <name>PC9</name>
    <port>PORT_C</port>
    <selected>NOT</selected>
    <Mode>MODE_INPUT</Mode>
    <OutType>OTYPE_PUSH_PULL</OutType>
    <OutLevel>LOW</OutLevel>
    <Pull>PULL_NOT_AVAILABLE</Pull>
    <Speed>SPEED_FAST</Speed>
    <codeName>PC9</codeName>
</pin>
<pin>PA8
    <name>PA8</name>
    <port>PORT_A</port>
    <selected>NOT</selected>
    <Mode>MODE_INPUT</Mode>
    <OutType>OTYPE_PUSH_PULL</OutType>
    <OutLevel>LOW</OutLevel>
    <Pull>PULL_NOT_AVAILABLE</Pull>
    <Speed>SPEED_FAST</Speed>
    <codeName>PA8</codeName>
</pin>
```

ANEXOS

```
<pin>PA9
    <name>PA9</name>
    <port>PORT_A</port>
    <selected>NOT</selected>
    <Mode>MODE_INPUT</Mode>
    <OutType>OTYPE_PUSH_PULL</OutType>
    <OutLevel>LOW</OutLevel>
    <Pull>PULL_NOT_AVAILABLE</Pull>
    <Speed>SPEED_FAST</Speed>
    <codeName>PA9</codeName>
</pin>
<pin>PA10
    <name>PA10</name>
    <port>PORT_A</port>
    <selected>NOT</selected>
    <Mode>MODE_INPUT</Mode>
    <OutType>OTYPE_PUSH_PULL</OutType>
    <OutLevel>LOW</OutLevel>
    <Pull>PULL_NOT_AVAILABLE</Pull>
    <Speed>SPEED_FAST</Speed>
    <codeName>PA10</codeName>
</pin>
<pin>PA11
    <name>PA11</name>
    <port>PORT_A</port>
    <selected>NOT</selected>
    <Mode>MODE_INPUT</Mode>
    <OutType>OTYPE_PUSH_PULL</OutType>
    <OutLevel>LOW</OutLevel>
    <Pull>PULL_NOT_AVAILABLE</Pull>
    <Speed>SPEED_FAST</Speed>
    <codeName>PA11</codeName>
</pin>
<pin>PA12
    <name>PA12</name>
    <port>PORT_A</port>
    <selected>YES</selected>
    <Mode>MODE_OUTPUT</Mode>
    <OutType>OTYPE_PUSH_PULL</OutType>
    <OutLevel>LOW</OutLevel>
    <Pull>PULL_NOT_AVAILABLE</Pull>
    <Speed>SPEED_FAST</Speed>
    <codeName>USER_LED</codeName>
</pin>
<pin>PA13
    <name>PA13</name>
    <port>PORT_A</port>
    <selected>NOT</selected>
    <Mode>MODE_INPUT</Mode>
    <OutType>OTYPE_PUSH_PULL</OutType>
    <OutLevel>LOW</OutLevel>
    <Pull>PULL_NOT_AVAILABLE</Pull>
    <Speed>SPEED_FAST</Speed>
    <codeName>PA13</codeName>
</pin>
<pin>PA14
    <name>PA14</name>
    <port>PORT_A</port>
    <selected>NOT</selected>
    <Mode>MODE_INPUT</Mode>
```

```
    <OutType>OTYPE_PUSH_PULL</OutType>
    <OutLevel>LOW</OutLevel>
    <Pull>PULL_NOT_AVAILABLE</Pull>
    <Speed>SPEED_FAST</Speed>
    <codeName>PA14</codeName>
</pin>
<pin>PA15
    <name>PA15</name>
    <port>PORT_A</port>
    <selected>NOT</selected>
    <Mode>MODE_INPUT</Mode>
    <OutType>OTYPE_PUSH_PULL</OutType>
    <OutLevel>LOW</OutLevel>
    <Pull>PULL_NOT_AVAILABLE</Pull>
    <Speed>SPEED_FAST</Speed>
    <codeName>PA15</codeName>
</pin>
<pin>PC10
    <name>PC10</name>
    <port>PORT_C</port>
    <selected>NOT</selected>
    <Mode>MODE_INPUT</Mode>
    <OutType>OTYPE_PUSH_PULL</OutType>
    <OutLevel>LOW</OutLevel>
    <Pull>PULL_NOT_AVAILABLE</Pull>
    <Speed>SPEED_FAST</Speed>
    <codeName>PC10</codeName>
</pin>
<pin>PC11
    <name>PC11</name>
    <port>PORT_C</port>
    <selected>NOT</selected>
    <Mode>MODE_INPUT</Mode>
    <OutType>OTYPE_PUSH_PULL</OutType>
    <OutLevel>LOW</OutLevel>
    <Pull>PULL_NOT_AVAILABLE</Pull>
    <Speed>SPEED_FAST</Speed>
    <codeName>PC11</codeName>
</pin>
<pin>PC12
    <name>PC12</name>
    <port>PORT_C</port>
    <selected>NOT</selected>
    <Mode>MODE_INPUT</Mode>
    <OutType>OTYPE_PUSH_PULL</OutType>
    <OutLevel>LOW</OutLevel>
    <Pull>PULL_NOT_AVAILABLE</Pull>
    <Speed>SPEED_FAST</Speed>
    <codeName>PC12</codeName>
</pin>
<pin>PD2
    <name>PD2</name>
    <port>PORT_D</port>
    <selected>NOT</selected>
    <Mode>MODE_INPUT</Mode>
    <OutType>OTYPE_PUSH_PULL</OutType>
    <OutLevel>LOW</OutLevel>
    <Pull>PULL_NOT_AVAILABLE</Pull>
    <Speed>SPEED_FAST</Speed>
    <codeName>PD2</codeName>
```

```
</pin>
<pin>PB3
    <name>PB3</name>
    <port>PORT_B</port>
    <selected>NOT</selected>
    <Mode>MODE_INPUT</Mode>
    <OutType>OTYPE_PUSH_PULL</OutType>
    <OutLevel>LOW</OutLevel>
    <Pull>PULL_NOT_AVAILABLE</Pull>
    <Speed>SPEED_FAST</Speed>
    <codeName>PB3</codeName>
</pin>
<pin>PB4
    <name>PB4</name>
    <port>PORT_B</port>
    <selected>NOT</selected>
    <Mode>MODE_INPUT</Mode>
    <OutType>OTYPE_PUSH_PULL</OutType>
    <OutLevel>LOW</OutLevel>
    <Pull>PULL_NOT_AVAILABLE</Pull>
    <Speed>SPEED_FAST</Speed>
    <codeName>PB4</codeName>
</pin>
<pin>PB5
    <name>PB5</name>
    <port>PORT_B</port>
    <selected>NOT</selected>
    <Mode>MODE_INPUT</Mode>
    <OutType>OTYPE_PUSH_PULL</OutType>
    <OutLevel>LOW</OutLevel>
    <Pull>PULL_NOT_AVAILABLE</Pull>
    <Speed>SPEED_FAST</Speed>
    <codeName>PB5</codeName>
</pin>
<pin>PB6
    <name>PB6</name>
    <port>PORT_B</port>
    <selected>NOT</selected>
    <Mode>MODE_INPUT</Mode>
    <OutType>OTYPE_PUSH_PULL</OutType>
    <OutLevel>LOW</OutLevel>
    <Pull>PULL_NOT_AVAILABLE</Pull>
    <Speed>SPEED_FAST</Speed>
    <codeName>PB6</codeName>
</pin>
<pin>PB7
    <name>PB7</name>
    <port>PORT_B</port>
    <selected>NOT</selected>
    <Mode>MODE_INPUT</Mode>
    <OutType>OTYPE_PUSH_PULL</OutType>
    <OutLevel>LOW</OutLevel>
    <Pull>PULL_NOT_AVAILABLE</Pull>
    <Speed>SPEED_FAST</Speed>
    <codeName>PB7</codeName>
</pin>
<pin>PB8
    <name>PB8</name>
    <port>PORT_B</port>
    <selected>NOT</selected>
```

```
        <Mode>MODE_INPUT</Mode>
        <OutType>OTYPE_PUSH_PULL</OutType>
        <OutLevel>LOW</OutLevel>
        <Pull>PULL_NOT_AVAILABLE</Pull>
        <Speed>SPEED_FAST</Speed>
        <codeName>PB8</codeName>
    </pin>
    <pin>PB9
        <name>PB9</name>
        <port>PORT_B</port>
        <selected>NOT</selected>
        <Mode>MODE_INPUT</Mode>
        <OutType>OTYPE_PUSH_PULL</OutType>
        <OutLevel>LOW</OutLevel>
        <Pull>PULL_NOT_AVAILABLE</Pull>
        <Speed>SPEED_FAST</Speed>
        <codeName>PB9</codeName>
    </pin>
</Microcontroller_Configuration>
```

10.3.2 MSP430FR6989

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<Microcontroller_Configuration>
       <pin>
              P4_3
              <name>P4_3</name>
              <port>GPIO_PORT_P4</port>
              <selected>NOT</selected>
              <Mode>MODE_INPUT</Mode>
              <OutType>OTYPE_PUSH_PULL</OutType>
              <OutLevel>LOW</OutLevel>
              <Pull>PULL_NOT_AVAILABLE</Pull>
              <Speed>SPEED_FAST</Speed>
              <codeName>P4_3</codeName>
       </pin>
       <pin>
              P1_4
              <name>P1_4</name>
              <port>GPIO_PORT_P1</port>
              <selected>NOT</selected>
              <Mode>MODE_INPUT</Mode>
              <OutType>OTYPE_PUSH_PULL</OutType>
              <OutLevel>LOW</OutLevel>
              <Pull>PULL_NOT_AVAILABLE</Pull>
              <Speed>SPEED_FAST</Speed>
              <codeName>P1_4</codeName>
       </pin>
       <pin>
              P1_5
              <name>P1_5</name>
              <port>GPIO_PORT_P1</port>
              <selected>NOT</selected>
              <Mode>MODE_INPUT</Mode>
              <OutType>OTYPE_PUSH_PULL</OutType>
              <OutLevel>LOW</OutLevel>
              <Pull>PULL_NOT_AVAILABLE</Pull>
              <Speed>SPEED_FAST</Speed>
              <codeName>P1_5</codeName>
```

ANEXOS

```
</pin>
<pin>
        P1_6
        <name>P1_6</name>
        <port>GPIO_PORT_P1</port>
        <selected>NOT</selected>
        <Mode>MODE_INPUT</Mode>
        <OutType>OTYPE_PUSH_PULL</OutType>
        <OutLevel>LOW</OutLevel>
        <Pull>PULL_NOT_AVAILABLE</Pull>
        <Speed>SPEED_FAST</Speed>
        <codeName>P1_6</codeName>
</pin>
<pin>
        P1_7
        <name>P1_7</name>
        <port>GPIO_PORT_P1</port>
        <selected>NOT</selected>
        <Mode>MODE_INPUT</Mode>
        <OutType>OTYPE_PUSH_PULL</OutType>
        <OutLevel>LOW</OutLevel>
        <Pull>PULL_NOT_AVAILABLE</Pull>
        <Speed>SPEED_FAST</Speed>
        <codeName>P1_7</codeName>
</pin>
<pin>
        P6_0
        <name>P6_0</name>
        <port>GPIO_PORT_P6</port>
        <selected>NOT</selected>
        <Mode>MODE_INPUT</Mode>
        <OutType>OTYPE_PUSH_PULL</OutType>
        <OutLevel>LOW</OutLevel>
        <Pull>PULL_NOT_AVAILABLE</Pull>
        <Speed>SPEED_FAST</Speed>
        <codeName>P6_0</codeName>
</pin>
<pin>
        P6_1
        <name>P6_1</name>
        <port>GPIO_PORT_P6</port>
        <selected>NOT</selected>
        <Mode>MODE_INPUT</Mode>
        <OutType>OTYPE_PUSH_PULL</OutType>
        <OutLevel>LOW</OutLevel>
        <Pull>PULL_NOT_AVAILABLE</Pull>
        <Speed>SPEED_FAST</Speed>
        <codeName>P6_1</codeName>
</pin>
<pin>
        P6_2
        <name>P6_2</name>
        <port>GPIO_PORT_P6</port>
        <selected>NOT</selected>
        <Mode>MODE_INPUT</Mode>
        <OutType>OTYPE_PUSH_PULL</OutType>
        <OutLevel>LOW</OutLevel>
        <Pull>PULL_NOT_AVAILABLE</Pull>
        <Speed>SPEED_FAST</Speed>
        <codeName>P6_2</codeName>
```

```
</pin>
<pin>
        P6_3
        <name>P6_3</name>
        <port>GPIO_PORT_P6</port>
        <selected>NOT</selected>
        <Mode>MODE_INPUT</Mode>
        <OutType>OTYPE_PUSH_PULL</OutType>
        <OutLevel>LOW</OutLevel>
        <Pull>PULL_NOT_AVAILABLE</Pull>
        <Speed>SPEED_FAST</Speed>
        <codeName>P6_3</codeName>
</pin>
<pin>
        P6_4
        <name>P6_4</name>
        <port>GPIO_PORT_P6</port>
        <selected>NOT</selected>
        <Mode>MODE_INPUT</Mode>
        <OutType>OTYPE_PUSH_PULL</OutType>
        <OutLevel>LOW</OutLevel>
        <Pull>PULL_NOT_AVAILABLE</Pull>
        <Speed>SPEED_FAST</Speed>
        <codeName>P6_4</codeName>
</pin>
<pin>
        P6_5
        <name>P6_5</name>
        <port>GPIO_PORT_P6</port>
        <selected>NOT</selected>
        <Mode>MODE_INPUT</Mode>
        <OutType>OTYPE_PUSH_PULL</OutType>
        <OutLevel>LOW</OutLevel>
        <Pull>PULL_NOT_AVAILABLE</Pull>
        <Speed>SPEED_FAST</Speed>
        <codeName>P6_5</codeName>
</pin>
<pin>
        P6_6
        <name>P6_6</name>
        <port>GPIO_PORT_P6</port>
        <selected>NOT</selected>
        <Mode>MODE_INPUT</Mode>
        <OutType>OTYPE_PUSH_PULL</OutType>
        <OutLevel>LOW</OutLevel>
        <Pull>PULL_NOT_AVAILABLE</Pull>
        <Speed>SPEED_FAST</Speed>
        <codeName>P6_6</codeName>
</pin>
<pin>
        P2_4
        <name>P2_4</name>
        <port>GPIO_PORT_P2</port>
        <selected>NOT</selected>
        <Mode>MODE_INPUT</Mode>
        <OutType>OTYPE_PUSH_PULL</OutType>
        <OutLevel>LOW</OutLevel>
        <Pull>PULL_NOT_AVAILABLE</Pull>
        <Speed>SPEED_FAST</Speed>
        <codeName>P2_4</codeName>
```

ANEXOS

```
</pin>
<pin>
        P2_5
        <name>P2_5</name>
        <port>GPIO_PORT_P2</port>
        <selected>NOT</selected>
        <Mode>MODE_INPUT</Mode>
        <OutType>OTYPE_PUSH_PULL</OutType>
        <OutLevel>LOW</OutLevel>
        <Pull>PULL_NOT_AVAILABLE</Pull>
        <Speed>SPEED_FAST</Speed>
        <codeName>P2_5</codeName>
</pin>
<pin>
        P2_6
        <name>P2_6</name>
        <port>GPIO_PORT_P2</port>
        <selected>NOT</selected>
        <Mode>MODE_INPUT</Mode>
        <OutType>OTYPE_PUSH_PULL</OutType>
        <OutLevel>LOW</OutLevel>
        <Pull>PULL_NOT_AVAILABLE</Pull>
        <Speed>SPEED_FAST</Speed>
        <codeName>P2_6</codeName>
</pin>
<pin>
        P2_7
        <name>P2_7</name>
        <port>GPIO_PORT_P2</port>
        <selected>NOT</selected>
        <Mode>MODE_INPUT</Mode>
        <OutType>OTYPE_PUSH_PULL</OutType>
        <OutLevel>LOW</OutLevel>
        <Pull>PULL_NOT_AVAILABLE</Pull>
        <Speed>SPEED_FAST</Speed>
        <codeName>P2_7</codeName>
</pin>
<pin>
        P10_2
        <name>P10_2</name>
        <port>GPIO_PORT_P10</port>
        <selected>NOT</selected>
        <Mode>MODE_INPUT</Mode>
        <OutType>OTYPE_PUSH_PULL</OutType>
        <OutLevel>LOW</OutLevel>
        <Pull>PULL_NOT_AVAILABLE</Pull>
        <Speed>SPEED_FAST</Speed>
        <codeName>P10_2</codeName>
</pin>
<pin>
        P5_0
        <name>P5_0</name>
        <port>GPIO_PORT_P5</port>
        <selected>NOT</selected>
        <Mode>MODE_INPUT</Mode>
        <OutType>OTYPE_PUSH_PULL</OutType>
        <OutLevel>LOW</OutLevel>
        <Pull>PULL_NOT_AVAILABLE</Pull>
        <Speed>SPEED_FAST</Speed>
        <codeName>P5_0</codeName>
```

```
</pin>
<pin>
        P5_1
        <name>P5_1</name>
        <port>GPIO_PORT_P5</port>
        <selected>NOT</selected>
        <Mode>MODE_INPUT</Mode>
        <OutType>OTYPE_PUSH_PULL</OutType>
        <OutLevel>LOW</OutLevel>
        <Pull>PULL_NOT_AVAILABLE</Pull>
        <Speed>SPEED_FAST</Speed>
        <codeName>P5_1</codeName>
</pin>
<pin>
        P5_2
        <name>P5_2</name>
        <port>GPIO_PORT_P5</port>
        <selected>NOT</selected>
        <Mode>MODE_INPUT</Mode>
        <OutType>OTYPE_PUSH_PULL</OutType>
        <OutLevel>LOW</OutLevel>
        <Pull>PULL_NOT_AVAILABLE</Pull>
        <Speed>SPEED_FAST</Speed>
        <codeName>P5_2</codeName>
</pin>
<pin>
        P5_3
        <name>P5_3</name>
        <port>GPIO_PORT_P5</port>
        <selected>NOT</selected>
        <Mode>MODE_INPUT</Mode>
        <OutType>OTYPE_PUSH_PULL</OutType>
        <OutLevel>LOW</OutLevel>
        <Pull>PULL_NOT_AVAILABLE</Pull>
        <Speed>SPEED_FAST</Speed>
        <codeName>P5_3</codeName>
</pin>
<pin>
        P3_0
        <name>P3_0</name>
        <port>GPIO_PORT_P3</port>
        <selected>NOT</selected>
        <Mode>MODE_INPUT</Mode>
        <OutType>OTYPE_PUSH_PULL</OutType>
        <OutLevel>LOW</OutLevel>
        <Pull>PULL_NOT_AVAILABLE</Pull>
        <Speed>SPEED_FAST</Speed>
        <codeName>P3_0</codeName>
</pin>
<pin>
        P3_1
        <name>P3_1</name>
        <port>GPIO_PORT_P3</port>
        <selected>NOT</selected>
        <Mode>MODE_INPUT</Mode>
        <OutType>OTYPE_PUSH_PULL</OutType>
        <OutLevel>LOW</OutLevel>
        <Pull>PULL_NOT_AVAILABLE</Pull>
        <Speed>SPEED_FAST</Speed>
        <codeName>P3_1</codeName>
```

ANEXOS

```
</pin>
<pin>
        P3_2
        <name>P3_2</name>
        <port>GPIO_PORT_P3</port>
        <selected>NOT</selected>
        <Mode>MODE_INPUT</Mode>
        <OutType>OTYPE_PUSH_PULL</OutType>
        <OutLevel>LOW</OutLevel>
        <Pull>PULL_NOT_AVAILABLE</Pull>
        <Speed>SPEED_FAST</Speed>
        <codeName>P3_2</codeName>
</pin>
<pin>
        PJ_0
        <name>PJ_0</name>
        <port>GPIO_PORT_PJ</port>
        <selected>NOT</selected>
        <Mode>MODE_INPUT</Mode>
        <OutType>OTYPE_PUSH_PULL</OutType>
        <OutLevel>LOW</OutLevel>
        <Pull>PULL_NOT_AVAILABLE</Pull>
        <Speed>SPEED_FAST</Speed>
        <codeName>PJ_0</codeName>
</pin>
<pin>
        PJ_1
        <name>PJ_1</name>
        <port>GPIO_PORT_PJ</port>
        <selected>NOT</selected>
        <Mode>MODE_INPUT</Mode>
        <OutType>OTYPE_PUSH_PULL</OutType>
        <OutLevel>LOW</OutLevel>
        <Pull>PULL_NOT_AVAILABLE</Pull>
        <Speed>SPEED_FAST</Speed>
        <codeName>PJ_1</codeName>
</pin>
<pin>
        PJ_2
        <name>PJ_2</name>
        <port>GPIO_PORT_PJ</port>
        <selected>NOT</selected>
        <Mode>MODE_INPUT</Mode>
        <OutType>OTYPE_PUSH_PULL</OutType>
        <OutLevel>LOW</OutLevel>
        <Pull>PULL_NOT_AVAILABLE</Pull>
        <Speed>SPEED_FAST</Speed>
        <codeName>PJ_2</codeName>
</pin>
<pin>
        PJ_3
        <name>PJ_3</name>
        <port>GPIO_PORT_PJ</port>
        <selected>NOT</selected>
        <Mode>MODE_INPUT</Mode>
        <OutType>OTYPE_PUSH_PULL</OutType>
        <OutLevel>LOW</OutLevel>
        <Pull>PULL_NOT_AVAILABLE</Pull>
        <Speed>SPEED_FAST</Speed>
        <codeName>PJ_3</codeName>
```

ANEXOS

```
</pin>
<pin>
        P6_7
        <name>P6_7</name>
        <port>GPIO_PORT_P6</port>
        <selected>NOT</selected>
        <Mode>MODE_INPUT</Mode>
        <OutType>OTYPE_PUSH_PULL</OutType>
        <OutLevel>LOW</OutLevel>
        <Pull>PULL_NOT_AVAILABLE</Pull>
        <Speed>SPEED_FAST</Speed>
        <codeName>P6_7</codeName>
</pin>
<pin>
        P7_5
        <name>P7_5</name>
        <port>GPIO_PORT_P7</port>
        <selected>NOT</selected>
        <Mode>MODE_INPUT</Mode>
        <OutType>OTYPE_PUSH_PULL</OutType>
        <OutLevel>LOW</OutLevel>
        <Pull>PULL_NOT_AVAILABLE</Pull>
        <Speed>SPEED_FAST</Speed>
        <codeName>P7_5</codeName>
</pin>
<pin>
        P7_6
        <name>P7_6</name>
        <port>GPIO_PORT_P7</port>
        <selected>NOT</selected>
        <Mode>MODE_INPUT</Mode>
        <OutType>OTYPE_PUSH_PULL</OutType>
        <OutLevel>LOW</OutLevel>
        <Pull>PULL_NOT_AVAILABLE</Pull>
        <Speed>SPEED_FAST</Speed>
        <codeName>P7_6</codeName>
</pin>
<pin>
        P10_1
        <name>P10_1</name>
        <port>GPIO_PORT_P10</port>
        <selected>NOT</selected>
        <Mode>MODE_INPUT</Mode>
        <OutType>OTYPE_PUSH_PULL</OutType>
        <OutLevel>LOW</OutLevel>
        <Pull>PULL_NOT_AVAILABLE</Pull>
        <Speed>SPEED_FAST</Speed>
        <codeName>P10_1</codeName>
</pin>
<pin>
        P7_7
        <name>P7_7</name>
        <port>GPIO_PORT_P7</port>
        <selected>NOT</selected>
        <Mode>MODE_INPUT</Mode>
        <OutType>OTYPE_PUSH_PULL</OutType>
        <OutLevel>LOW</OutLevel>
        <Pull>PULL_NOT_AVAILABLE</Pull>
        <Speed>SPEED_FAST</Speed>
        <codeName>P7_7</codeName>
```

```
</pin>
<pin>
        P3_3
        <name>P3_3</name>
        <port>GPIO_PORT_P3</port>
        <selected>NOT</selected>
        <Mode>MODE_INPUT</Mode>
        <OutType>OTYPE_PUSH_PULL</OutType>
        <OutLevel>LOW</OutLevel>
        <Pull>PULL_NOT_AVAILABLE</Pull>
        <Speed>SPEED_FAST</Speed>
        <codeName>P3_3</codeName>
</pin>
<pin>
        P3_4
        <name>P3_4</name>
        <port>GPIO_PORT_P3</port>
        <selected>NOT</selected>
        <Mode>MODE_INPUT</Mode>
        <OutType>OTYPE_PUSH_PULL</OutType>
        <OutLevel>LOW</OutLevel>
        <Pull>PULL_NOT_AVAILABLE</Pull>
        <Speed>SPEED_FAST</Speed>
        <codeName>P3_4</codeName>
</pin>
<pin>
        P3_5
        <name>P3_5</name>
        <port>GPIO_PORT_P3</port>
        <selected>NOT</selected>
        <Mode>MODE_INPUT</Mode>
        <OutType>OTYPE_PUSH_PULL</OutType>
        <OutLevel>LOW</OutLevel>
        <Pull>PULL_NOT_AVAILABLE</Pull>
        <Speed>SPEED_FAST</Speed>
        <codeName>P3_5</codeName>
</pin>
<pin>
        P3_6
        <name>P3_6</name>
        <port>GPIO_PORT_P3</port>
        <selected>NOT</selected>
        <Mode>MODE_INPUT</Mode>
        <OutType>OTYPE_PUSH_PULL</OutType>
        <OutLevel>LOW</OutLevel>
        <Pull>PULL_NOT_AVAILABLE</Pull>
        <Speed>SPEED_FAST</Speed>
        <codeName>P3_6</codeName>
</pin>
<pin>
        P3_7
        <name>P3_7</name>
        <port>GPIO_PORT_P3</port>
        <selected>NOT</selected>
        <Mode>MODE_INPUT</Mode>
        <OutType>OTYPE_PUSH_PULL</OutType>
        <OutLevel>LOW</OutLevel>
        <Pull>PULL_NOT_AVAILABLE</Pull>
        <Speed>SPEED_FAST</Speed>
        <codeName>P3_7</codeName>
```

ANEXOS

```
</pin>
<pin>
        P8_0
        <name>P8_0</name>
        <port>GPIO_PORT_P8</port>
        <selected>NOT</selected>
        <Mode>MODE_INPUT</Mode>
        <OutType>OTYPE_PUSH_PULL</OutType>
        <OutLevel>LOW</OutLevel>
        <Pull>PULL_NOT_AVAILABLE</Pull>
        <Speed>SPEED_FAST</Speed>
        <codeName>P8_0</codeName>
</pin>
<pin>
        P8_1
        <name>P8_1</name>
        <port>GPIO_PORT_P8</port>
        <selected>NOT</selected>
        <Mode>MODE_INPUT</Mode>
        <OutType>OTYPE_PUSH_PULL</OutType>
        <OutLevel>LOW</OutLevel>
        <Pull>PULL_NOT_AVAILABLE</Pull>
        <Speed>SPEED_FAST</Speed>
        <codeName>P8_1</codeName>
</pin>
<pin>
        P8_2
        <name>P8_2</name>
        <port>GPIO_PORT_P8</port>
        <selected>NOT</selected>
        <Mode>MODE_INPUT</Mode>
        <OutType>OTYPE_PUSH_PULL</OutType>
        <OutLevel>LOW</OutLevel>
        <Pull>PULL_NOT_AVAILABLE</Pull>
        <Speed>SPEED_FAST</Speed>
        <codeName>P8_2</codeName>
</pin>
<pin>
        P8_3
        <name>P8_3</name>
        <port>GPIO_PORT_P8</port>
        <selected>NOT</selected>
        <Mode>MODE_INPUT</Mode>
        <OutType>OTYPE_PUSH_PULL</OutType>
        <OutLevel>LOW</OutLevel>
        <Pull>PULL_NOT_AVAILABLE</Pull>
        <Speed>SPEED_FAST</Speed>
        <codeName>P8_3</codeName>
</pin>
<pin>
        P2_3
        <name>P2_3</name>
        <port>GPIO_PORT_P2</port>
        <selected>NOT</selected>
        <Mode>MODE_INPUT</Mode>
        <OutType>OTYPE_PUSH_PULL</OutType>
        <OutLevel>LOW</OutLevel>
        <Pull>PULL_NOT_AVAILABLE</Pull>
        <Speed>SPEED_FAST</Speed>
        <codeName>P2_3</codeName>
```

ANEXOS

```
</pin>
<pin>
        P2_2
        <name>P2_2</name>
        <port>GPIO_PORT_P2</port>
        <selected>NOT</selected>
        <Mode>MODE_INPUT</Mode>
        <OutType>OTYPE_PUSH_PULL</OutType>
        <OutLevel>LOW</OutLevel>
        <Pull>PULL_NOT_AVAILABLE</Pull>
        <Speed>SPEED_FAST</Speed>
        <codeName>P2_2</codeName>
</pin>
<pin>
        P2_1
        <name>P2_1</name>
        <port>GPIO_PORT_P2</port>
        <selected>NOT</selected>
        <Mode>MODE_INPUT</Mode>
        <OutType>OTYPE_PUSH_PULL</OutType>
        <OutLevel>LOW</OutLevel>
        <Pull>PULL_NOT_AVAILABLE</Pull>
        <Speed>SPEED_FAST</Speed>
        <codeName>P2_1</codeName>
</pin>
<pin>
        P2_0
        <name>P2_0</name>
        <port>GPIO_PORT_P2</port>
        <selected>NOT</selected>
        <Mode>MODE_INPUT</Mode>
        <OutType>OTYPE_PUSH_PULL</OutType>
        <OutLevel>LOW</OutLevel>
        <Pull>PULL_NOT_AVAILABLE</Pull>
        <Speed>SPEED_FAST</Speed>
        <codeName>P2_0</codeName>
</pin>
<pin>
        P7_0
        <name>P7_0</name>
        <port>GPIO_PORT_P7</port>
        <selected>NOT</selected>
        <Mode>MODE_INPUT</Mode>
        <OutType>OTYPE_PUSH_PULL</OutType>
        <OutLevel>LOW</OutLevel>
        <Pull>PULL_NOT_AVAILABLE</Pull>
        <Speed>SPEED_FAST</Speed>
        <codeName>P7_0</codeName>
</pin>
<pin>
        P7_1
        <name>P7_1</name>
        <port>GPIO_PORT_P7</port>
        <selected>NOT</selected>
        <Mode>MODE_INPUT</Mode>
        <OutType>OTYPE_PUSH_PULL</OutType>
        <OutLevel>LOW</OutLevel>
        <Pull>PULL_NOT_AVAILABLE</Pull>
        <Speed>SPEED_FAST</Speed>
        <codeName>P7_1</codeName>
```

```
</pin>
<pin>
        P7_2
        <name>P7_2</name>
        <port>GPIO_PORT_P7</port>
        <selected>NOT</selected>
        <Mode>MODE_INPUT</Mode>
        <OutType>OTYPE_PUSH_PULL</OutType>
        <OutLevel>LOW</OutLevel>
        <Pull>PULL_NOT_AVAILABLE</Pull>
        <Speed>SPEED_FAST</Speed>
        <codeName>P7_2</codeName>
</pin>
<pin>
        P7_3
        <name>P7_3</name>
        <port>GPIO_PORT_P7</port>
        <selected>NOT</selected>
        <Mode>MODE_INPUT</Mode>
        <OutType>OTYPE_PUSH_PULL</OutType>
        <OutLevel>LOW</OutLevel>
        <Pull>PULL_NOT_AVAILABLE</Pull>
        <Speed>SPEED_FAST</Speed>
        <codeName>P7_3</codeName>
</pin>
<pin>
        P7_4
        <name>P7_4</name>
        <port>GPIO_PORT_P7</port>
        <selected>NOT</selected>
        <Mode>MODE_INPUT</Mode>
        <OutType>OTYPE_PUSH_PULL</OutType>
        <OutLevel>LOW</OutLevel>
        <Pull>PULL_NOT_AVAILABLE</Pull>
        <Speed>SPEED_FAST</Speed>
        <codeName>P7_4</codeName>
</pin>
<pin>
        P8_4
        <name>P8_4</name>
        <port>GPIO_PORT_P8</port>
        <selected>NOT</selected>
        <Mode>MODE_INPUT</Mode>
        <OutType>OTYPE_PUSH_PULL</OutType>
        <OutLevel>LOW</OutLevel>
        <Pull>PULL_NOT_AVAILABLE</Pull>
        <Speed>SPEED_FAST</Speed>
        <codeName>P8_4</codeName>
</pin>
<pin>
        P8_5
        <name>P8_5</name>
        <port>GPIO_PORT_P8</port>
        <selected>NOT</selected>
        <Mode>MODE_INPUT</Mode>
        <OutType>OTYPE_PUSH_PULL</OutType>
        <OutLevel>LOW</OutLevel>
        <Pull>PULL_NOT_AVAILABLE</Pull>
        <Speed>SPEED_FAST</Speed>
        <codeName>P8_5</codeName>
```

ANEXOS

```
</pin>
<pin>
        P8_6
        <name>P8_6</name>
        <port>GPIO_PORT_P8</port>
        <selected>NOT</selected>
        <Mode>MODE_INPUT</Mode>
        <OutType>OTYPE_PUSH_PULL</OutType>
        <OutLevel>LOW</OutLevel>
        <Pull>PULL_NOT_AVAILABLE</Pull>
        <Speed>SPEED_FAST</Speed>
        <codeName>P8_6</codeName>
</pin>
<pin>
        P8_7
        <name>P8_7</name>
        <port>GPIO_PORT_P8</port>
        <selected>NOT</selected>
        <Mode>MODE_INPUT</Mode>
        <OutType>OTYPE_PUSH_PULL</OutType>
        <OutLevel>LOW</OutLevel>
        <Pull>PULL_NOT_AVAILABLE</Pull>
        <Speed>SPEED_FAST</Speed>
        <codeName>P8_7</codeName>
</pin>
<pin>
        P1_3
        <name>P1_3</name>
        <port>GPIO_PORT_P1</port>
        <selected>NOT</selected>
        <Mode>MODE_INPUT</Mode>
        <OutType>OTYPE_PUSH_PULL</OutType>
        <OutLevel>LOW</OutLevel>
        <Pull>PULL_NOT_AVAILABLE</Pull>
        <Speed>SPEED_FAST</Speed>
        <codeName>P1_3</codeName>
</pin>
<pin>
        P1_2
        <name>P1_2</name>
        <port>GPIO_PORT_P1</port>
        <selected>NOT</selected>
        <Mode>MODE_INPUT</Mode>
        <OutType>OTYPE_PUSH_PULL</OutType>
        <OutLevel>LOW</OutLevel>
        <Pull>PULL_NOT_AVAILABLE</Pull>
        <Speed>SPEED_FAST</Speed>
        <codeName>P1_2</codeName>
</pin>
<pin>
        P1_1
        <name>P1_1</name>
        <port>GPIO_PORT_P1</port>
        <selected>YES</selected>
        <Mode>MODE_INPUT</Mode>
        <OutType>OTYPE_PUSH_PULL</OutType>
        <OutLevel>LOW</OutLevel>
        <Pull>PULL_UP</Pull>
        <Speed>SPEED_FAST</Speed>
        <codeName>USER_BUTTON</codeName>
```

ANEXOS

```
</pin>
<pin>
        P1_0
        <name>P1_0</name>
        <port>GPIO_PORT_P1</port>
        <selected>YES</selected>
        <Mode>MODE_OUTPUT</Mode>
        <OutType>OTYPE_PUSH_PULL</OutType>
        <OutLevel>LOW</OutLevel>
        <Pull>PULL_NOT_AVAILABLE</Pull>
        <Speed>SPEED_FAST</Speed>
        <codeName>USER_LED</codeName>
</pin>
<pin>
        P9_0
        <name>P9_0</name>
        <port>GPIO_PORT_P9</port>
        <selected>NOT</selected>
        <Mode>MODE_INPUT</Mode>
        <OutType>OTYPE_PUSH_PULL</OutType>
        <OutLevel>LOW</OutLevel>
        <Pull>PULL_NOT_AVAILABLE</Pull>
        <Speed>SPEED_FAST</Speed>
        <codeName>P9_0</codeName>
</pin>
<pin>
        P9_1
        <name>P9_1</name>
        <port>GPIO_PORT_P9</port>
        <selected>NOT</selected>
        <Mode>MODE_INPUT</Mode>
        <OutType>OTYPE_PUSH_PULL</OutType>
        <OutLevel>LOW</OutLevel>
        <Pull>PULL_NOT_AVAILABLE</Pull>
        <Speed>SPEED_FAST</Speed>
        <codeName>P9_1</codeName>
</pin>
<pin>
        P9_2
        <name>P9_2</name>
        <port>GPIO_PORT_P9</port>
        <selected>NOT</selected>
        <Mode>MODE_INPUT</Mode>
        <OutType>OTYPE_PUSH_PULL</OutType>
        <OutLevel>LOW</OutLevel>
        <Pull>PULL_NOT_AVAILABLE</Pull>
        <Speed>SPEED_FAST</Speed>
        <codeName>P9_2</codeName>
</pin>
<pin>
        P9_3
        <name>P9_3</name>
        <port>GPIO_PORT_P9</port>
        <selected>NOT</selected>
        <Mode>MODE_INPUT</Mode>
        <OutType>OTYPE_PUSH_PULL</OutType>
        <OutLevel>LOW</OutLevel>
        <Pull>PULL_NOT_AVAILABLE</Pull>
        <Speed>SPEED_FAST</Speed>
        <codeName>P9_3</codeName>
```

```
</pin>
<pin>
        P9_4
        <name>P9_4</name>
        <port>GPIO_PORT_P9</port>
        <selected>NOT</selected>
        <Mode>MODE_INPUT</Mode>
        <OutType>OTYPE_PUSH_PULL</OutType>
        <OutLevel>LOW</OutLevel>
        <Pull>PULL_NOT_AVAILABLE</Pull>
        <Speed>SPEED_FAST</Speed>
        <codeName>P9_4</codeName>
</pin>
<pin>
        P9_5
        <name>P9_5</name>
        <port>GPIO_PORT_P9</port>
        <selected>NOT</selected>
        <Mode>MODE_INPUT</Mode>
        <OutType>OTYPE_PUSH_PULL</OutType>
        <OutLevel>LOW</OutLevel>
        <Pull>PULL_NOT_AVAILABLE</Pull>
        <Speed>SPEED_FAST</Speed>
        <codeName>P9_5</codeName>
</pin>
<pin>
        P9_6
        <name>P9_6</name>
        <port>GPIO_PORT_P9</port>
        <selected>NOT</selected>
        <Mode>MODE_INPUT</Mode>
        <OutType>OTYPE_PUSH_PULL</OutType>
        <OutLevel>LOW</OutLevel>
        <Pull>PULL_NOT_AVAILABLE</Pull>
        <Speed>SPEED_FAST</Speed>
        <codeName>P9_6</codeName>
</pin>
<pin>
        P9_7
        <name>P9_7</name>
        <port>GPIO_PORT_P9</port>
        <selected>NOT</selected>
        <Mode>MODE_INPUT</Mode>
        <OutType>OTYPE_PUSH_PULL</OutType>
        <OutLevel>LOW</OutLevel>
        <Pull>PULL_NOT_AVAILABLE</Pull>
        <Speed>SPEED_FAST</Speed>
        <codeName>P9_7</codeName>
</pin>
<pin>
        PJ_7
        <name>PJ_7</name>
        <port>GPIO_PORT_PJ</port>
        <selected>NOT</selected>
        <Mode>MODE_INPUT</Mode>
        <OutType>OTYPE_PUSH_PULL</OutType>
        <OutLevel>LOW</OutLevel>
        <Pull>PULL_NOT_AVAILABLE</Pull>
        <Speed>SPEED_FAST</Speed>
        <codeName>PJ_7</codeName>
```

ANEXOS

```
</pin>
<pin>
        PJ_6
        <name>PJ_6</name>
        <port>GPIO_PORT_PJ</port>
        <selected>NOT</selected>
        <Mode>MODE_INPUT</Mode>
        <OutType>OTYPE_PUSH_PULL</OutType>
        <OutLevel>LOW</OutLevel>
        <Pull>PULL_NOT_AVAILABLE</Pull>
        <Speed>SPEED_FAST</Speed>
        <codeName>PJ_6</codeName>
</pin>
<pin>
        PJ_4
        <name>PJ_4</name>
        <port>GPIO_PORT_PJ</port>
        <selected>NOT</selected>
        <Mode>MODE_INPUT</Mode>
        <OutType>OTYPE_PUSH_PULL</OutType>
        <OutLevel>LOW</OutLevel>
        <Pull>PULL_NOT_AVAILABLE</Pull>
        <Speed>SPEED_FAST</Speed>
        <codeName>PJ_4</codeName>
</pin>
<pin>
        PJ_5
        <name>PJ_5</name>
        <port>GPIO_PORT_PJ</port>
        <selected>NOT</selected>
        <Mode>MODE_INPUT</Mode>
        <OutType>OTYPE_PUSH_PULL</OutType>
        <OutLevel>LOW</OutLevel>
        <Pull>PULL_NOT_AVAILABLE</Pull>
        <Speed>SPEED_FAST</Speed>
        <codeName>PJ_5</codeName>
</pin>
<pin>
        P5_4
        <name>P5_4</name>
        <port>GPIO_PORT_P5</port>
        <selected>NOT</selected>
        <Mode>MODE_INPUT</Mode>
        <OutType>OTYPE_PUSH_PULL</OutType>
        <OutLevel>LOW</OutLevel>
        <Pull>PULL_NOT_AVAILABLE</Pull>
        <Speed>SPEED_FAST</Speed>
        <codeName>P5_4</codeName>
</pin>
<pin>
        P5_5
        <name>P5_5</name>
        <port>GPIO_PORT_P5</port>
        <selected>NOT</selected>
        <Mode>MODE_INPUT</Mode>
        <OutType>OTYPE_PUSH_PULL</OutType>
        <OutLevel>LOW</OutLevel>
        <Pull>PULL_NOT_AVAILABLE</Pull>
        <Speed>SPEED_FAST</Speed>
        <codeName>P5_5</codeName>
```

ANEXOS

```
</pin>
<pin>
        P5_6
        <name>P5_6</name>
        <port>GPIO_PORT_P5</port>
        <selected>NOT</selected>
        <Mode>MODE_INPUT</Mode>
        <OutType>OTYPE_PUSH_PULL</OutType>
        <OutLevel>LOW</OutLevel>
        <Pull>PULL_NOT_AVAILABLE</Pull>
        <Speed>SPEED_FAST</Speed>
        <codeName>P5_6</codeName>
</pin>
<pin>
        P5_7
        <name>P5_7</name>
        <port>GPIO_PORT_P5</port>
        <selected>NOT</selected>
        <Mode>MODE_INPUT</Mode>
        <OutType>OTYPE_PUSH_PULL</OutType>
        <OutLevel>LOW</OutLevel>
        <Pull>PULL_NOT_AVAILABLE</Pull>
        <Speed>SPEED_FAST</Speed>
        <codeName>P5_7</codeName>
</pin>
<pin>
        P4_4
        <name>P4_4</name>
        <port>GPIO_PORT_P4</port>
        <selected>NOT</selected>
        <Mode>MODE_INPUT</Mode>
        <OutType>OTYPE_PUSH_PULL</OutType>
        <OutLevel>LOW</OutLevel>
        <Pull>PULL_NOT_AVAILABLE</Pull>
        <Speed>SPEED_FAST</Speed>
        <codeName>P4_4</codeName>
</pin>
<pin>
        P4_5
        <name>P4_5</name>
        <port>GPIO_PORT_P4</port>
        <selected>NOT</selected>
        <Mode>MODE_INPUT</Mode>
        <OutType>OTYPE_PUSH_PULL</OutType>
        <OutLevel>LOW</OutLevel>
        <Pull>PULL_NOT_AVAILABLE</Pull>
        <Speed>SPEED_FAST</Speed>
        <codeName>P4_5</codeName>
</pin>
<pin>
        P4_6
        <name>P4_6</name>
        <port>GPIO_PORT_P4</port>
        <selected>NOT</selected>
        <Mode>MODE_INPUT</Mode>
        <OutType>OTYPE_PUSH_PULL</OutType>
        <OutLevel>LOW</OutLevel>
        <Pull>PULL_NOT_AVAILABLE</Pull>
        <Speed>SPEED_FAST</Speed>
        <codeName>P4_6</codeName>
```

```
</pin>
<pin>
        P4_7
        <name>P4_7</name>
        <port>GPIO_PORT_P9</port>
        <selected>NOT</selected>
        <Mode>MODE_INPUT</Mode>
        <OutType>OTYPE_PUSH_PULL</OutType>
        <OutLevel>LOW</OutLevel>
        <Pull>PULL_NOT_AVAILABLE</Pull>
        <Speed>SPEED_FAST</Speed>
        <codeName>P4_7</codeName>
</pin>
<pin>
        P10_0
        <name>P10_0</name>
        <port>GPIO_PORT_P10</port>
        <selected>NOT</selected>
        <Mode>MODE_INPUT</Mode>
        <OutType>OTYPE_PUSH_PULL</OutType>
        <OutLevel>LOW</OutLevel>
        <Pull>PULL_NOT_AVAILABLE</Pull>
        <Speed>SPEED_FAST</Speed>
        <codeName>P10_0</codeName>
</pin>
<pin>
        P4_0
        <name>P4_0</name>
        <port>GPIO_PORT_P4</port>
        <selected>NOT</selected>
        <Mode>MODE_INPUT</Mode>
        <OutType>OTYPE_PUSH_PULL</OutType>
        <OutLevel>LOW</OutLevel>
        <Pull>PULL_NOT_AVAILABLE</Pull>
        <Speed>SPEED_FAST</Speed>
        <codeName>P4_0</codeName>
</pin>
<pin>
        P4_1
        <name>P4_1</name>
        <port>GPIO_PORT_P4</port>
        <selected>NOT</selected>
        <Mode>MODE_INPUT</Mode>
        <OutType>OTYPE_PUSH_PULL</OutType>
        <OutLevel>LOW</OutLevel>
        <Pull>PULL_NOT_AVAILABLE</Pull>
        <Speed>SPEED_FAST</Speed>
        <codeName>P4_1</codeName>
</pin>
<pin>
        P4_2
        <name>P4_2</name>
        <port>GPIO_PORT_P4</port>
        <selected>NOT</selected>
        <Mode>MODE_INPUT</Mode>
        <OutType>OTYPE_PUSH_PULL</OutType>
        <OutLevel>LOW</OutLevel>
        <Pull>PULL_NOT_AVAILABLE</Pull>
        <Speed>SPEED_FAST</Speed>
        <codeName>P4_2</codeName>
```

```
      </pin>
</Microcontroller_Configuration>
```

10.4 Prueba de concepto

10.4.1 main.c

```
/*****************************************************************************
*. Copyright 2017 Miguel Diaz, All Rights Reserved
*.=============================================================================
*. Template C Source File
*.=============================================================================
*. FILE: main.c
*.
*. DESCRIPTION:
*. This file
*.
*. SOFTWARE AND DOCUMENTATION ARE PROVIDED "AS IS" FOR THE PURPOSES OF THE
*. PROVIDED TRAINING CLASS AND IT IS NOT INTENDED AS PRODUCTION CODE AND WITHOUT
*. WARRANTY OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING WITHOUT LIMITATION,
*. ANY WARRANTY OF MERCHANTABILITY, TITLE, NON-INFRINGEMENT AND FITNESS FOR A
*. PARTICULAR PURPOSE. IN NO EVENT SHALL SALVADOR ALMANZA OR ITS LICENSORS BE LIABLE OR
*. OBLIGATED UNDER CONTRACT, NEGLIGENCE, STRICT LIABILITY, CONTRIBUTION, BREACH
*. OF WARRANTY, OR OTHER LEGAL EQUITABLE THEORY ANY DIRECT OR INDIRECT DAMAGES
*. OR EXPENSES INCLUDING BUT NOT LIMITED TO ANY INCIDENTAL, SPECIAL, INDIRECT,
*. PUNITIVE OR CONSEQUENTIAL DAMAGES, LOST PROFITS OR LOST DATA, COST OF
*. PROCUREMENT OF SUBSTITUTE GOODS, TECHNOLOGY, SERVICES, OR ANY CLAIMS BY THIRD
*. PARTIES (INCLUDING BUT NOT LIMITED TO ANY DEFENSE THEREOF), OR OTHER SIMILAR
*. COSTS.
*.
\*****************************************************************************/
/************************
 * Includes             *
 ************************/
#include "frameworkIncludes.h"
#include "main.h"

/************************
 * Type Definitions     *
 ************************/

/************************
 * Macros               *
 ************************/

/************************
 * Defines              *
 ************************/

/************************
 * Constants            *
 ************************/

/************************
 * Calibrations         *
 ************************/
```

ANEXOS

```
/***********************
 * Global Variables     *
 ***********************/

/***********************
 * File Scope Variables *
 ***********************/

/***********************
 * Function Prototypes  *
 ***********************/

/***********************
 * Function Definitions *
 ***********************/

/*****************************************************************
*. Name: main
*.==============================================================
*. Description:
*. C entry point, executes the systems initialization and application
*.
\*****************************************************************/
int main(void)
{
    Gpio_pinState_t ledState = LED_OFF;

    // Initialize basic hardware functionality
    HWInit();

    // Initialize GPIOs
    Gpio_Init(Gpio_Cfg);

    while (true)
    {
        if (Gpio_GetPin(USER_BUTTON_PORT, USER_BUTTON_PIN) == BUTTON_PRESSED)
        {
            ledState = LED_ON;
        }
        else
        {
            ledState = LED_OFF;
        }

        Gpio_SetPin(USER_LED_PORT, USER_LED_PIN, ledState);
    }
}
```