UNIVERSIDAD DE BUENOS AIRES

FACULTAD DE INGENIERÍA

MAESTRÍA EN SISTEMAS EMBEBIDOS

MEMORIA DEL TRABAJO FINAL

sAPI (simpleAPI): diseño e implementación de una biblioteca para sistematizar la programación de sistemas embebidos

Autor:
Esp. Ing. Eric Nicolás Pernia

Director:
MSc. Ing. Félix Gustavo E. Safar.

Jurados:
Dr. Ing. Pablo M. Gómez (UBA)
Mg. Ing. Pablo O. Ridolfi (UTN FRBA)
Ing. Juan Manuel Cruz (FIUBA,UTN-FRBA)

Este trabajo fue realizado en las Ciudad Autónoma de Buenos Aires, entre marzo de 2018 y diciembre de 2018.

Resumen

En esta memoria se presenta el diseño de una biblioteca para la programación de sistemas embebidos portable entre plataformas de hardware y lenguajes de programación. Se realizó una implementación de referencia en lenguaje C para las plataformas del Proyecto CIAA. Se realizó un banco de pruebas de hardware, junto a la utilización de testeo unitario e integración contínua para la validación.

Además de la biblioteca, se creó una herramienta de código abierto para desarrolladores que automatiza la implemetación de bibliotecas a partir de un modelo que las describe. De esta manera se facilita la futura ampliación de la biblioteca y su implementación en otras plataformas de hardware.

Agradecimientos

Agradecimientos personales. **[OPCIONAL]**

No olvidarse de agradecer al tutor.

No vale poner anti-agradecimientos (este trabajo fue posible a pesar de...)

Índice general

Índice de figuras

Índice de Tablas

Dedicado a... [OPCIONAL]

Capítulo 1

Introducción General

La idea de esta sección es presentar el tema de modo que cualquier persona que no conoce el tema pueda entender de qué se trata y por qué es importante realizar este trabajo y cuál es su impacto.

1.1. Contexto y justificación

Saraza... La idea de esta sección es presentar el tema de modo que cualquier persona que no conoce el tema pueda entender de qué se trata y por qué es importante realizar este trabajo y cuál es su impacto.

1.2. Motivación

Saraza...

1.2.1. Uso de mayúscula inicial para los título de secciones

Saraza...

1.2.2. Lenguajes de programación y Hardware en Sistemas Embebidos

Saraza...

1.2.3. Bibliotecas para microcontroladores ofrecidas por los fabricantes

Saraza...

1.2.4. Proyecto CIAA

Saraza...

1.3. Objetivos y alcance

En esta sección se definen los objetivos del presente Trabajo Final (sección 1.3.1) y el alcance (sección 1.3.2)

De esta forma, es posible programarlos sin necesidad de conocer detalles sobre la arquitectura subyacente. Este diseño promueve y permite la programación independiente del hardware, reduciendo la complejidad general del desarrollo de

sistemas embebidos. Esta biblioteca se utilizará para programar las diferentes plataformas que componen el Proyecto Computadora Industrial Abierta Argentina (CIAA).

1.3.1. Objetivos

El objetivo de este proyecto es diseñar e implementar una biblioteca de software para la programación de sistemas embebidos basados en microcontroladores con las siguientes características:

- Estar modelada independientemente de los lenguajes de programación.
- Definir una interfaz de programación de aplicaciones (API) sencilla que abstraiga los modos de uso más comunes de los periféricos típicos que hallados en cualquier microcontrolador del mercado.
- Ser totalmente portable entre diferentes arquitecturas de hardware sobre donde se ejecuta, manteniendo una API uniforme a lo largo de las mismas[1].

Dicha biblioteca se deberá implementar en lenguaje C para las plataformas del Proyecto CIAA.

1.3.2. Alcance

Este Trabajo Final incluye realización de:

- Diseño de la biblioteca. Archivos de descripción de la biblioteca mediante diferentes diagramas y código independiente del lenguaje de programación.
- Implementación en lenguaje C de la biblioteca para las plataformas de hardware:
  - CIAA-NXP.
  - EDU-CIAA-NXP.
  - CIAA-Z3R0.
  - PicoCIAA.
- Manual de instalación de las herramientas para utilizar la biblioteca con las plataformas de hardware citadas.
- Manual de referencia de la biblioteca.
- Ejemplos de utilización.

[1] Debe cumplir la función de capa de abstracción de hardware, o *Hardware Abstraction Layer* (HAL), en inglés

Capítulo 2

Introducción Específica

2.1. Plataformas del Proyecto CIAA

Saraza...

2.1.1. CIAA-NXP

Saraza...

2.1.2. EDU-CIAA-NXP

Saraza...

2.1.3. Pico-CIAA

Saraza...

2.1.4. CIAA-Z3R0

Saraza...

2.1.5. Arquitectura de Hardware

Saraza...

2.1.6. Bibliotecas disponibles

Saraza...

2.2. Requerimientos

Saraza...

2.2.1. Diseño de la biblioteca

Saraza...

2.2.2. Implementación

Saraza...

2.2.3. Documentación y difusión

Saraza...

2.3. Planificación

Diagrama de Gannt

Capítulo 3

Diseño

3.1. Análisis del software

La idea de esta sección es resaltar los problemas encontrados, los criterios utilizados y la justificación de las decisiones que se hayan tomado.

Se puede agregar código o pseudocódigo dentro de un entorno lstlisting con el siguiente código:

```
\begin{lstlisting}[caption= "un epígrafe descriptivo"]

las líneas de código irían aquí...

\end{lstlisting}
```

A modo de ejemplo:

ALGORITMO 3.1: Pseudocódigo del lazo principal de control.

```
#define MAX_SENSOR_NUMBER 3
#define MAX_ALARM_NUMBER  6
#define MAX_ACTUATOR_NUMBER 6

uint32_t sensorValue[MAX_SENSOR_NUMBER];
FunctionalState alarmControl[MAX_ALARM_NUMBER]; //ENABLE or DISABLE
state_t alarmState[MAX_ALARM_NUMBER];
state_t actuatorState[MAX_ACTUATOR_NUMBER];                    //ON or OF

void vControl() {

        initGlobalVariables();

        period = 500 ms;

        while(1) {

                ticks = xTaskGetTickCount();

                updateSensors();

                updateAlarms();
```

```
                controlActuators();

                vTaskDelayUntil(&ticks, period);
        }
}
```

3.2. Análisis del software

La idea de esta sección es resaltar los problemas encontrados, los criterios utilizados y la justificación de las decisiones que se hayan tomado.

Se puede agregar código o pseudocódigo dentro de un entorno lstlisting con el siguiente código:

```
\begin{lstlisting}[caption= "un epígrafe descriptivo"]

las líneas de código irían aquí...

\end{lstlisting}
```

A modo de ejemplo:

ALGORITMO 3.2: Pseudocódigo del lazo principal de control.

```
#define MAX_SENSOR_NUMBER 3
#define MAX_ALARM_NUMBER  6
#define MAX_ACTUATOR_NUMBER 6

uint32_t sensorValue[MAX_SENSOR_NUMBER];
FunctionalState alarmControl[MAX_ALARM_NUMBER]; //ENABLE or DISABLE
state_t alarmState[MAX_ALARM_NUMBER];
state_t actuatorState[MAX_ACTUATOR_NUMBER];                    //ON or OFF

void vControl() {

        initGlobalVariables();

        period = 500 ms;

        while(1) {

                ticks = xTaskGetTickCount();

                updateSensors();

                updateAlarms();

                controlActuators();

                vTaskDelayUntil(&ticks, period);
```

```
        }
}
```

3.3. Análisis del software

La idea de esta sección es resaltar los problemas encontrados, los criterios utilizados y la justificación de las decisiones que se hayan tomado.

Se puede agregar código o pseudocódigo dentro de un entorno lstlisting con el siguiente código:

```
\begin{lstlisting}[caption= "un epígrafe descriptivo"]

las líneas de código irían aquí...

\end{lstlisting}
```

A modo de ejemplo:

ALGORITMO 3.3: Pseudocódigo del lazo principal de control.

```
#define MAX_SENSOR_NUMBER 3
#define MAX_ALARM_NUMBER  6
#define MAX_ACTUATOR_NUMBER 6

uint32_t sensorValue[MAX_SENSOR_NUMBER];
FunctionalState alarmControl[MAX_ALARM_NUMBER]; //ENABLE or DISABLE
state_t alarmState[MAX_ALARM_NUMBER];
state_t actuatorState[MAX_ACTUATOR_NUMBER];                    //ON or OF

void vControl() {

        initGlobalVariables();

        period = 500 ms;

        while(1) {

                ticks = xTaskGetTickCount();

                updateSensors();

                updateAlarms();

                controlActuators();

                vTaskDelayUntil(&ticks, period);
        }
}
```

3.4. Análisis del software

La idea de esta sección es resaltar los problemas encontrados, los criterios utilizados y la justificación de las decisiones que se hayan tomado.

Se puede agregar código o pseudocódigo dentro de un entorno lstlisting con el siguiente código:

```
\begin{lstlisting}[caption= "un epígrafe descriptivo"]

las líneas de código irían aquí...

\end{lstlisting}
```

A modo de ejemplo:

ALGORITMO 3.4: Pseudocódigo del lazo principal de control.

```
#define MAX_SENSOR_NUMBER 3
#define MAX_ALARM_NUMBER  6
#define MAX_ACTUATOR_NUMBER 6

uint32_t sensorValue[MAX_SENSOR_NUMBER];
FunctionalState alarmControl[MAX_ALARM_NUMBER]; //ENABLE or DISABLE
state_t alarmState[MAX_ALARM_NUMBER];
state_t actuatorState[MAX_ACTUATOR_NUMBER];                    //ON or OFF

void vControl() {

        initGlobalVariables();

        period = 500 ms;

        while(1) {

                ticks = xTaskGetTickCount();

                updateSensors();

                updateAlarms();

                controlActuators();

                vTaskDelayUntil(&ticks, period);
        }
}
```

3.5. Análisis del software

La idea de esta sección es resaltar los problemas encontrados, los criterios utilizados y la justificación de las decisiones que se hayan tomado.

Se puede agregar código o pseudocódigo dentro de un entorno lstlisting con el siguiente código:

```
\begin{lstlisting}[caption= "un epígrafe descriptivo"]

las líneas de código irían aquí...

\end{lstlisting}
```

A modo de ejemplo:

ALGORITMO 3.5: Pseudocódigo del lazo principal de control.

```
#define MAX_SENSOR_NUMBER 3
#define MAX_ALARM_NUMBER  6
#define MAX_ACTUATOR_NUMBER 6

uint32_t sensorValue[MAX_SENSOR_NUMBER];
FunctionalState alarmControl[MAX_ALARM_NUMBER]; //ENABLE or DISABLE
state_t alarmState[MAX_ALARM_NUMBER];
state_t actuatorState[MAX_ACTUATOR_NUMBER];                    //ON or OF

void vControl() {

        initGlobalVariables();

        period = 500 ms;

        while(1) {

                ticks = xTaskGetTickCount();

                updateSensors();

                updateAlarms();

                controlActuators();

                vTaskDelayUntil(&ticks, period);
        }
}
```

3.6. Análisis del software

La idea de esta sección es resaltar los problemas encontrados, los criterios utilizados y la justificación de las decisiones que se hayan tomado.

Se puede agregar código o pseudocódigo dentro de un entorno lstlisting con el siguiente código:

```
\begin{lstlisting}[caption= "un epígrafe descriptivo"]
```

```
las líneas de código irían aquí...

\end{lstlisting}
```

A modo de ejemplo:

ALGORITMO 3.6: Pseudocódigo del lazo principal de control.

```
#define MAX_SENSOR_NUMBER 3
#define MAX_ALARM_NUMBER  6
#define MAX_ACTUATOR_NUMBER 6

uint32_t sensorValue[MAX_SENSOR_NUMBER];
FunctionalState alarmControl[MAX_ALARM_NUMBER]; //ENABLE or DISABLE
state_t alarmState[MAX_ALARM_NUMBER];
state_t actuatorState[MAX_ACTUATOR_NUMBER];                    //ON or OFF

void vControl() {

        initGlobalVariables();

        period = 500 ms;

        while(1) {

                ticks = xTaskGetTickCount();

                updateSensors();

                updateAlarms();

                controlActuators();

                vTaskDelayUntil(&ticks, period);
        }
}
```

3.7. Análisis del software

La idea de esta sección es resaltar los problemas encontrados, los criterios utilizados y la justificación de las decisiones que se hayan tomado.

Se puede agregar código o pseudocódigo dentro de un entorno lstlisting con el siguiente código:

```
\begin{lstlisting}[caption= "un epígrafe descriptivo"]

las líneas de código irían aquí...

\end{lstlisting}
```

A modo de ejemplo:

ALGORITMO 3.7: Pseudocódigo del lazo principal de control.

```
#define MAX_SENSOR_NUMBER 3
#define MAX_ALARM_NUMBER  6
#define MAX_ACTUATOR_NUMBER 6

uint32_t sensorValue[MAX_SENSOR_NUMBER];
FunctionalState alarmControl[MAX_ALARM_NUMBER]; //ENABLE or DISABLE
state_t alarmState[MAX_ALARM_NUMBER];
state_t actuatorState[MAX_ACTUATOR_NUMBER];                    //ON or OF

void vControl() {

        initGlobalVariables();

        period = 500 ms;

        while(1) {

                ticks = xTaskGetTickCount();

                updateSensors();

                updateAlarms();

                controlActuators();

                vTaskDelayUntil(&ticks, period);
        }
}
```

3.8. Análisis del software

La idea de esta sección es resaltar los problemas encontrados, los criterios utilizados y la justificación de las decisiones que se hayan tomado.

Se puede agregar código o pseudocódigo dentro de un entorno lstlisting con el siguiente código:

```
\begin{lstlisting}[caption= "un epígrafe descriptivo"]

las líneas de código irían aquí...

\end{lstlisting}
```

A modo de ejemplo:

ALGORITMO 3.8: Pseudocódigo del lazo principal de control.

```
#define MAX_SENSOR_NUMBER 3
```

```
#define MAX_ALARM_NUMBER  6
#define MAX_ACTUATOR_NUMBER 6

uint32_t sensorValue[MAX_SENSOR_NUMBER];
FunctionalState alarmControl[MAX_ALARM_NUMBER]; //ENABLE or DISABLE
state_t alarmState[MAX_ALARM_NUMBER];
state_t actuatorState[MAX_ACTUATOR_NUMBER];                    //ON or OFF

void vControl() {

        initGlobalVariables();

        period = 500 ms;

        while(1) {

                ticks = xTaskGetTickCount();

                updateSensors();

                updateAlarms();

                controlActuators();

                vTaskDelayUntil(&ticks, period);
        }
}
```

3.9. Análisis del software

La idea de esta sección es resaltar los problemas encontrados, los criterios utilizados y la justificación de las decisiones que se hayan tomado.

Se puede agregar código o pseudocódigo dentro de un entorno lstlisting con el siguiente código:

```
\begin{lstlisting}[caption= "un epígrafe descriptivo"]

las líneas de código irían aquí...

\end{lstlisting}
```

A modo de ejemplo:

ALGORITMO 3.9: Pseudocódigo del lazo principal de control.

```
#define MAX_SENSOR_NUMBER 3
#define MAX_ALARM_NUMBER  6
#define MAX_ACTUATOR_NUMBER 6

uint32_t sensorValue[MAX_SENSOR_NUMBER];
```

```
FunctionalState alarmControl[MAX_ALARM_NUMBER]; //ENABLE or DISABLE
state_t alarmState[MAX_ALARM_NUMBER];
state_t actuatorState[MAX_ACTUATOR_NUMBER];                    //ON or OF

void vControl() {

        initGlobalVariables();

        period = 500 ms;

        while(1) {

                ticks = xTaskGetTickCount();

                updateSensors();

                updateAlarms();

                controlActuators();

                vTaskDelayUntil(&ticks, period);
        }
}
```

3.10. Análisis del software

La idea de esta sección es resaltar los problemas encontrados, los criterios utilizados y la justificación de las decisiones que se hayan tomado.

Se puede agregar código o pseudocódigo dentro de un entorno lstlisting con el siguiente código:

```
\begin{lstlisting}[caption= "un epígrafe descriptivo"]

las líneas de código irían aquí...

\end{lstlisting}
```

A modo de ejemplo:

ALGORITMO 3.10: Pseudocódigo del lazo principal de control.

```
#define MAX_SENSOR_NUMBER 3
#define MAX_ALARM_NUMBER  6
#define MAX_ACTUATOR_NUMBER 6

uint32_t sensorValue[MAX_SENSOR_NUMBER];
FunctionalState alarmControl[MAX_ALARM_NUMBER]; //ENABLE or DISABLE
state_t alarmState[MAX_ALARM_NUMBER];
state_t actuatorState[MAX_ACTUATOR_NUMBER];                    //ON or OF
```

```
void vControl() {

        initGlobalVariables();

        period = 500 ms;

        while(1) {

                ticks = xTaskGetTickCount();

                updateSensors();

                updateAlarms();

                controlActuators();

                vTaskDelayUntil(&ticks, period);
        }
}
```

3.11. Análisis del software

La idea de esta sección es resaltar los problemas encontrados, los criterios utilizados y la justificación de las decisiones que se hayan tomado.

Se puede agregar código o pseudocódigo dentro de un entorno lstlisting con el siguiente código:

```
\begin{lstlisting}[caption= "un epígrafe descriptivo"]

las líneas de código irían aquí...

\end{lstlisting}
```

A modo de ejemplo:

ALGORITMO 3.11: Pseudocódigo del lazo principal de control.

```
#define MAX_SENSOR_NUMBER 3
#define MAX_ALARM_NUMBER  6
#define MAX_ACTUATOR_NUMBER 6

uint32_t sensorValue[MAX_SENSOR_NUMBER];
FunctionalState alarmControl[MAX_ALARM_NUMBER]; //ENABLE or DISABLE
state_t alarmState[MAX_ALARM_NUMBER];
state_t actuatorState[MAX_ACTUATOR_NUMBER];                    //ON or OFF

void vControl() {

        initGlobalVariables();
```

```
        period = 500 ms;

        while(1) {

                ticks = xTaskGetTickCount();

                updateSensors();

                updateAlarms();

                controlActuators();

                vTaskDelayUntil(&ticks, period);
        }
}
```

3.12. Análisis del software

La idea de esta sección es resaltar los problemas encontrados, los criterios utilizados y la justificación de las decisiones que se hayan tomado.

Se puede agregar código o pseudocódigo dentro de un entorno lstlisting con el siguiente código:

```
\begin{lstlisting}[caption= "un epígrafe descriptivo"]

las líneas de código irían aquí...

\end{lstlisting}
```

A modo de ejemplo:

ALGORITMO 3.12: Pseudocódigo del lazo principal de control.

```
#define MAX_SENSOR_NUMBER 3
#define MAX_ALARM_NUMBER  6
#define MAX_ACTUATOR_NUMBER 6

uint32_t sensorValue[MAX_SENSOR_NUMBER];
FunctionalState alarmControl[MAX_ALARM_NUMBER]; //ENABLE or DISABLE
state_t alarmState[MAX_ALARM_NUMBER];
state_t actuatorState[MAX_ACTUATOR_NUMBER];                    //ON or OF

void vControl() {

        initGlobalVariables();

        period = 500 ms;

        while(1) {
```

```
                ticks = xTaskGetTickCount();

                updateSensors();

                updateAlarms();

                controlActuators();

                vTaskDelayUntil(&ticks, period);
        }
}
```

3.13. Análisis del software

La idea de esta sección es resaltar los problemas encontrados, los criterios utilizados y la justificación de las decisiones que se hayan tomado.

Se puede agregar código o pseudocódigo dentro de un entorno lstlisting con el siguiente código:

```
\begin{lstlisting}[caption= "un epígrafe descriptivo"]

las líneas de código irían aquí...

\end{lstlisting}
```

A modo de ejemplo:

ALGORITMO 3.13: Pseudocódigo del lazo principal de control.

```
#define MAX_SENSOR_NUMBER 3
#define MAX_ALARM_NUMBER  6
#define MAX_ACTUATOR_NUMBER 6

uint32_t sensorValue[MAX_SENSOR_NUMBER];
FunctionalState alarmControl[MAX_ALARM_NUMBER]; //ENABLE or DISABLE
state_t alarmState[MAX_ALARM_NUMBER];
state_t actuatorState[MAX_ACTUATOR_NUMBER];                    //ON or OFF

void vControl() {

        initGlobalVariables();

        period = 500 ms;

        while(1) {

                ticks = xTaskGetTickCount();

                updateSensors();
```

```
                updateAlarms();

                controlActuators();

                vTaskDelayUntil(&ticks, period);
        }
}
```

3.14. Análisis del software

La idea de esta sección es resaltar los problemas encontrados, los criterios utilizados y la justificación de las decisiones que se hayan tomado.

Se puede agregar código o pseudocódigo dentro de un entorno lstlisting con el siguiente código:

```
\begin{lstlisting}[caption= "un epígrafe descriptivo"]

las líneas de código irían aquí...

\end{lstlisting}
```

A modo de ejemplo:

ALGORITMO 3.14: Pseudocódigo del lazo principal de control.

```
#define MAX_SENSOR_NUMBER 3
#define MAX_ALARM_NUMBER  6
#define MAX_ACTUATOR_NUMBER 6

uint32_t sensorValue[MAX_SENSOR_NUMBER];
FunctionalState alarmControl[MAX_ALARM_NUMBER]; //ENABLE or DISABLE
state_t alarmState[MAX_ALARM_NUMBER];
state_t actuatorState[MAX_ACTUATOR_NUMBER];                    //ON or OF

void vControl() {

        initGlobalVariables();

        period = 500 ms;

        while(1) {

                ticks = xTaskGetTickCount();

                updateSensors();

                updateAlarms();

                controlActuators();
```

```
                vTaskDelayUntil(&ticks , period);
        }
}
```

3.15. Análisis del software

La idea de esta sección es resaltar los problemas encontrados, los criterios utilizados y la justificación de las decisiones que se hayan tomado.

Se puede agregar código o pseudocódigo dentro de un entorno lstlisting con el siguiente código:

```
\begin{lstlisting}[caption= "un epígrafe descriptivo"]

las líneas de código irían aquí...

\end{lstlisting}
```

A modo de ejemplo:

ALGORITMO 3.15: Pseudocódigo del lazo principal de control.

```
#define MAX_SENSOR_NUMBER 3
#define MAX_ALARM_NUMBER  6
#define MAX_ACTUATOR_NUMBER 6

uint32_t sensorValue[MAX_SENSOR_NUMBER];
FunctionalState alarmControl[MAX_ALARM_NUMBER]; //ENABLE or DISABLE
state_t alarmState[MAX_ALARM_NUMBER];
state_t actuatorState[MAX_ACTUATOR_NUMBER];                    //ON or OFF

void vControl() {

        initGlobalVariables();

        period = 500 ms;

        while(1) {

                ticks = xTaskGetTickCount();

                updateSensors();

                updateAlarms();

                controlActuators();

                vTaskDelayUntil(&ticks , period);
        }
}
```

3.16. Análisis del software

La idea de esta sección es resaltar los problemas encontrados, los criterios utilizados y la justificación de las decisiones que se hayan tomado.

Se puede agregar código o pseudocódigo dentro de un entorno lstlisting con el siguiente código:

```
\begin{lstlisting}[caption= "un epígrafe descriptivo"]

las líneas de código irían aquí...

\end{lstlisting}
```

A modo de ejemplo:

ALGORITMO 3.16: Pseudocódigo del lazo principal de control.

```
#define MAX_SENSOR_NUMBER 3
#define MAX_ALARM_NUMBER  6
#define MAX_ACTUATOR_NUMBER 6

uint32_t sensorValue[MAX_SENSOR_NUMBER];
FunctionalState alarmControl[MAX_ALARM_NUMBER]; //ENABLE or DISABLE
state_t alarmState[MAX_ALARM_NUMBER];
state_t actuatorState[MAX_ACTUATOR_NUMBER];                    //ON or OF

void vControl() {

        initGlobalVariables();

        period = 500 ms;

        while(1) {

                ticks = xTaskGetTickCount();

                updateSensors();

                updateAlarms();

                controlActuators();

                vTaskDelayUntil(&ticks, period);
        }
}
```

Capítulo 4

Implementación

4.1. Análisis del software

La idea de esta sección es resaltar los problemas encontrados, los criterios utilizados y la justificación de las decisiones que se hayan tomado.

Se puede agregar código o pseudocódigo dentro de un entorno lstlisting con el siguiente código:

```
\begin{lstlisting}[caption= "un epígrafe descriptivo"]

las líneas de código irían aquí...

\end{lstlisting}
```

A modo de ejemplo:

ALGORITMO 4.1: Pseudocódigo del lazo principal de control.

```
#define MAX_SENSOR_NUMBER 3
#define MAX_ALARM_NUMBER  6
#define MAX_ACTUATOR_NUMBER 6

uint32_t sensorValue[MAX_SENSOR_NUMBER];
FunctionalState alarmControl[MAX_ALARM_NUMBER]; //ENABLE or DISABLE
state_t alarmState[MAX_ALARM_NUMBER];
state_t actuatorState[MAX_ACTUATOR_NUMBER];                    //ON or OF

void vControl() {

        initGlobalVariables();

        period = 500 ms;

        while(1) {

                ticks = xTaskGetTickCount();

                updateSensors();

                updateAlarms();
```

```
                controlActuators();

                vTaskDelayUntil(&ticks, period);
        }
}
```

4.2. Análisis del software

La idea de esta sección es resaltar los problemas encontrados, los criterios utilizados y la justificación de las decisiones que se hayan tomado.

Se puede agregar código o pseudocódigo dentro de un entorno lstlisting con el siguiente código:

```
\begin{lstlisting}[caption= "un epígrafe descriptivo"]

las líneas de código irían aquí...

\end{lstlisting}
```

A modo de ejemplo:

ALGORITMO 4.2: Pseudocódigo del lazo principal de control.

```
#define MAX_SENSOR_NUMBER 3
#define MAX_ALARM_NUMBER  6
#define MAX_ACTUATOR_NUMBER 6

uint32_t sensorValue[MAX_SENSOR_NUMBER];
FunctionalState alarmControl[MAX_ALARM_NUMBER]; //ENABLE or DISABLE
state_t alarmState[MAX_ALARM_NUMBER];
state_t actuatorState[MAX_ACTUATOR_NUMBER];                    //ON or OFF

void vControl() {

        initGlobalVariables();

        period = 500 ms;

        while(1) {

                ticks = xTaskGetTickCount();

                updateSensors();

                updateAlarms();

                controlActuators();

                vTaskDelayUntil(&ticks, period);
```

```
        }
}
```

4.3. Análisis del software

La idea de esta sección es resaltar los problemas encontrados, los criterios utilizados y la justificación de las decisiones que se hayan tomado.

Se puede agregar código o pseudocódigo dentro de un entorno lstlisting con el siguiente código:

```
\begin{lstlisting}[caption= "un epígrafe descriptivo"]

las líneas de código irían aquí...

\end{lstlisting}
```

A modo de ejemplo:

ALGORITMO 4.3: Pseudocódigo del lazo principal de control.

```
#define MAX_SENSOR_NUMBER 3
#define MAX_ALARM_NUMBER  6
#define MAX_ACTUATOR_NUMBER 6

uint32_t sensorValue[MAX_SENSOR_NUMBER];
FunctionalState alarmControl[MAX_ALARM_NUMBER]; //ENABLE or DISABLE
state_t alarmState[MAX_ALARM_NUMBER];
state_t actuatorState[MAX_ACTUATOR_NUMBER];                    //ON or OF

void vControl() {

        initGlobalVariables();

        period = 500 ms;

        while(1) {

                ticks = xTaskGetTickCount();

                updateSensors();

                updateAlarms();

                controlActuators();

                vTaskDelayUntil(&ticks, period);
        }
}
```

4.4. Análisis del software

La idea de esta sección es resaltar los problemas encontrados, los criterios utilizados y la justificación de las decisiones que se hayan tomado.

Se puede agregar código o pseudocódigo dentro de un entorno lstlisting con el siguiente código:

```
\begin{lstlisting}[caption= "un epígrafe descriptivo"]

las líneas de código irían aquí...

\end{lstlisting}
```

A modo de ejemplo:

ALGORITMO 4.4: Pseudocódigo del lazo principal de control.

```
#define MAX_SENSOR_NUMBER 3
#define MAX_ALARM_NUMBER  6
#define MAX_ACTUATOR_NUMBER 6

uint32_t sensorValue[MAX_SENSOR_NUMBER];
FunctionalState alarmControl[MAX_ALARM_NUMBER]; //ENABLE or DISABLE
state_t alarmState[MAX_ALARM_NUMBER];
state_t actuatorState[MAX_ACTUATOR_NUMBER];                    //ON or OFF

void vControl() {

        initGlobalVariables();

        period = 500 ms;

        while(1) {

                ticks = xTaskGetTickCount();

                updateSensors();

                updateAlarms();

                controlActuators();

                vTaskDelayUntil(&ticks, period);
        }
}
```

4.5. Análisis del software

La idea de esta sección es resaltar los problemas encontrados, los criterios utilizados y la justificación de las decisiones que se hayan tomado.

Se puede agregar código o pseudocódigo dentro de un entorno lstlisting con el siguiente código:

```
\begin{lstlisting}[caption= "un epígrafe descriptivo"]

las líneas de código irían aquí...

\end{lstlisting}
```

A modo de ejemplo:

ALGORITMO 4.5: Pseudocódigo del lazo principal de control.

```
#define MAX_SENSOR_NUMBER 3
#define MAX_ALARM_NUMBER  6
#define MAX_ACTUATOR_NUMBER 6

uint32_t sensorValue[MAX_SENSOR_NUMBER];
FunctionalState alarmControl[MAX_ALARM_NUMBER]; //ENABLE or DISABLE
state_t alarmState[MAX_ALARM_NUMBER];
state_t actuatorState[MAX_ACTUATOR_NUMBER];                    //ON or OF

void vControl() {

        initGlobalVariables();

        period = 500 ms;

        while(1) {

                ticks = xTaskGetTickCount();

                updateSensors();

                updateAlarms();

                controlActuators();

                vTaskDelayUntil(&ticks, period);
        }
}
```

4.6. Análisis del software

La idea de esta sección es resaltar los problemas encontrados, los criterios utilizados y la justificación de las decisiones que se hayan tomado.

Se puede agregar código o pseudocódigo dentro de un entorno lstlisting con el siguiente código:

```
\begin{lstlisting}[caption= "un epígrafe descriptivo"]
```

```
las líneas de código irían aquí...

\end{lstlisting}
```

A modo de ejemplo:

ALGORITMO 4.6: Pseudocódigo del lazo principal de control.

```
#define MAX_SENSOR_NUMBER 3
#define MAX_ALARM_NUMBER  6
#define MAX_ACTUATOR_NUMBER 6

uint32_t sensorValue[MAX_SENSOR_NUMBER];
FunctionalState alarmControl[MAX_ALARM_NUMBER]; //ENABLE or DISABLE
state_t alarmState[MAX_ALARM_NUMBER];
state_t actuatorState[MAX_ACTUATOR_NUMBER];                    //ON or OFF

void vControl() {

        initGlobalVariables();

        period = 500 ms;

        while(1) {

                ticks = xTaskGetTickCount();

                updateSensors();

                updateAlarms();

                controlActuators();

                vTaskDelayUntil(&ticks, period);
        }
}
```

4.7. Análisis del software

La idea de esta sección es resaltar los problemas encontrados, los criterios utilizados y la justificación de las decisiones que se hayan tomado.

Se puede agregar código o pseudocódigo dentro de un entorno lstlisting con el siguiente código:

```
\begin{lstlisting}[caption= "un epígrafe descriptivo"]

las líneas de código irían aquí...

\end{lstlisting}
```

A modo de ejemplo:

ALGORITMO 4.7: Pseudocódigo del lazo principal de control.

```
#define MAX_SENSOR_NUMBER 3
#define MAX_ALARM_NUMBER  6
#define MAX_ACTUATOR_NUMBER 6

uint32_t sensorValue[MAX_SENSOR_NUMBER];
FunctionalState alarmControl[MAX_ALARM_NUMBER]; //ENABLE or DISABLE
state_t alarmState[MAX_ALARM_NUMBER];
state_t actuatorState[MAX_ACTUATOR_NUMBER];                    //ON or OF

void vControl() {

        initGlobalVariables();

        period = 500 ms;

        while(1) {

                ticks = xTaskGetTickCount();

                updateSensors();

                updateAlarms();

                controlActuators();

                vTaskDelayUntil(&ticks, period);
        }
}
```

4.8. Análisis del software

La idea de esta sección es resaltar los problemas encontrados, los criterios utilizados y la justificación de las decisiones que se hayan tomado.

Se puede agregar código o pseudocódigo dentro de un entorno lstlisting con el siguiente código:

```
\begin{lstlisting}[caption= "un epígrafe descriptivo"]

las líneas de código irían aquí...

\end{lstlisting}
```

A modo de ejemplo:

ALGORITMO 4.8: Pseudocódigo del lazo principal de control.

```
#define MAX_SENSOR_NUMBER 3
```

```
#define MAX_ALARM_NUMBER  6
#define MAX_ACTUATOR_NUMBER 6

uint32_t sensorValue[MAX_SENSOR_NUMBER];
FunctionalState alarmControl[MAX_ALARM_NUMBER]; //ENABLE or DISABLE
state_t alarmState[MAX_ALARM_NUMBER];
state_t actuatorState[MAX_ACTUATOR_NUMBER];                    //ON or OFF

void vControl() {

        initGlobalVariables();

        period = 500 ms;

        while(1) {

                ticks = xTaskGetTickCount();

                updateSensors();

                updateAlarms();

                controlActuators();

                vTaskDelayUntil(&ticks, period);
        }
}
```

4.9. Análisis del software

La idea de esta sección es resaltar los problemas encontrados, los criterios utilizados y la justificación de las decisiones que se hayan tomado.

Se puede agregar código o pseudocódigo dentro de un entorno lstlisting con el siguiente código:

```
\begin{lstlisting}[caption= "un epígrafe descriptivo"]

las líneas de código irían aquí...

\end{lstlisting}
```

A modo de ejemplo:

ALGORITMO 4.9: Pseudocódigo del lazo principal de control.

```
#define MAX_SENSOR_NUMBER 3
#define MAX_ALARM_NUMBER  6
#define MAX_ACTUATOR_NUMBER 6

uint32_t sensorValue[MAX_SENSOR_NUMBER];
```

```
FunctionalState alarmControl[MAX_ALARM_NUMBER]; //ENABLE or DISABLE
state_t alarmState[MAX_ALARM_NUMBER];
state_t actuatorState[MAX_ACTUATOR_NUMBER];                    //ON or OF

void vControl() {

        initGlobalVariables();

        period = 500 ms;

        while(1) {

                ticks = xTaskGetTickCount();

                updateSensors();

                updateAlarms();

                controlActuators();

                vTaskDelayUntil(&ticks, period);
        }
}
```

Capítulo 5

Ensayos y Resultados

5.1. Pruebas funcionales del hardware

La idea de esta sección es explicar cómo se hicieron los ensayos, qué resultados se obtuvieron y analizarlos.

Capítulo 6

Conclusiones

6.1. Conclusiones generales

La idea de esta sección es resaltar cuáles son los principales aportes del trabajo realizado y cómo se podría continuar. Debe ser especialmente breve y concisa. Es buena idea usar un listado para enumerar los logros obtenidos.

6.2. Próximos pasos

Acá se indica cómo se podría continuar el trabajo más adelante.