**UNIVERSIDAD TECNOLÓGICA DE QUERÉTARO**

**CESEQ**



**Diplomado en Software Embebido**

Proyecto Integrador: Control de velocidad de motor CD

<File Name: Software Development Plan/SWRA\_YYYYMMDD.docx/etc>

DOCUMENT: Software Development

Document No. #CESEQ\_SDP\_001

Scrum Master: Solís, Gabriel

Developer. Pintor, Antonio

Date (YYYYMMDD): 20190724

Version: 1.0.0.

Project Version: 1.0.1.

# Log

Document Version

|  |  |  |  |
| --- | --- | --- | --- |
| Version | Date (yyyymmdd) | Description | Reviewer |
| 1.0.0. | 20190405 | First release | Pérez, Adbeel |
|  |  |  |  |

Project Document Version

|  |  |  |  |
| --- | --- | --- | --- |
| Version | Date (yyyymmdd) | Description | Reviewer |
| 1.0.1. | 20190723 | Modified “6. Estimates” | Solis, Gabriel |
| 1.0.2. | 20190724 | Modified “5”, “6” | Solis, Gabriel |
| 1.0.3 | 20190729 | Modified “6. Estimates” | Solis, Gabriel |
| 1.0.4. | 20190803 | Modified “5”, “6”, “10.3” | Solis, Gabriel |
| 1.0.5. | 20190805 | Added “3. Project scope”, “8. Solving Problem Strategy, “9. Design”, “9.2 Naming conventions” | Pintor, Antonio |
| 1.0.6. | 20190806 | Updated “3, 8 and 9” | Pintor, Antonio |

# Index

Table of Contents

[1. Log 2](#_Toc8215461)

[2. Index 3](#_Toc8215462)

[3. Project Scope 4](#_Toc8215463)

[4. Deliverables 4](#_Toc8215464)

[5. Development methodology 4](#_Toc8215465)

[6. Estimates 4](#_Toc8215466)

[7. Planning 5](#_Toc8215467)

[8. Solving Problem Strategy 6](#_Toc8215468)

[9. Design 6](#_Toc8215469)

[9.1. Standards 6](#_Toc8215470)

[9.2. Naming conventions 6](#_Toc8215471)

[10. Testing 7](#_Toc8215472)

[10.1. Verification strategy (black box test) 7](#_Toc8215473)

[10.2. White box strategy 7](#_Toc8215474)

[10.3. Cyclomatic Complexity Redundance index 7](#_Toc8215475)

[11. Release 8](#_Toc8215476)

[11.1. Software Development Folder 8](#_Toc8215477)

[11.2. Integration Tests Strategy 8](#_Toc8215478)

[11.3. Validation Testing / Functional Testing 8](#_Toc8215479)

[11.4. Throughput and Flash and RAM measurement 9](#_Toc8215480)

[12. Results 9](#_Toc8215481)

[13. Lessons Learned 9](#_Toc8215482)

# Project Scope

El alcance de este proyecto es aplicar durante el desarrollo todas las metodologías, estrategias, herramientas, diseño del software y métodos de prueba vistas durante el diplomado de software embebido.

**Definición del problema**.

Controlar la velocidad de un motor de corriente directa mediante la aplicación de una señal cuadrada que varía en su ancho de pulso y cuya frecuencia de trabajo **debe** ser constante.

La frecuencia de trabajo **debe** estar en un rango de *f* = 100 Hz a *f* = 1 KHz (esta frecuencia puede ser modificada para obtener dentro del rango para mejorar la señal de retroalimentación del motor).

Una vez seleccionada la frecuencia de trabajo (señal del sensor de efecto hall con menor ruido), esta **debe** ser fija, variando únicamente el “*duty cycle*”.

Mediante el uso de un sensor de efecto hall acoplado al rotor del motor se **debe** medir la velocidad del motor el cual proveerá una serie de pulsos cada que se complete una vuelta completa, esta medición.

Así, a mayor velocidad del motor, mayor será el número de pulsos leídos y mientras menor sea la velocidad, menor será el número de pulsos.

El voltaje de alimentación de la tarjeta de potencia debe ser de 12 Volts.

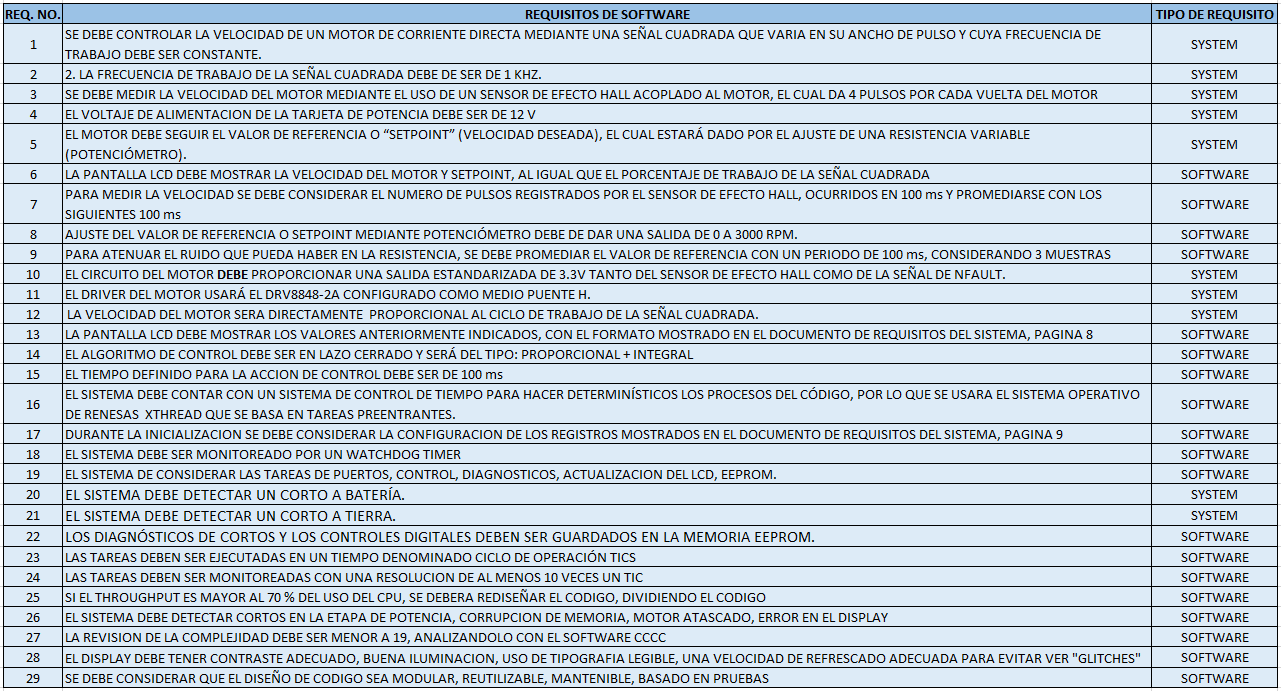
El motor **debe** seguir el valor de referencia o “SetPoint” (velocidad deseada), el cual estará dado por una entrada de la tarjeta de control.

La pantalla LCD o interfaz gráfica **debe** mostrar la velocidad del motor y SetPoint (ambos en RPM’s); así como el porcentaje de trabajo de la señal cuadrada.

Para más detalles del proyecto consulte el siguiente documento:

<PATH DEL PROYECTO>\1) Requirements\stakeholder

**Requisitos del Proyecto.**



Este documento de Requisitos del Proyecto también puede ser encontrado en el siguiente archivo.

<PROJECT\_PATH>\ 1) Requirements\3. SWRA\_20190405.xlsx

# Deliverables

En esta sección se menciona la lista de los entregables tanto de documentación como de código.

Estos documentos son encontrados en las siguientes carpetas para su revisión:

1) Requirements

* 3. SWRA\_20190405.xlsx

2) Planning

* 7. Planning\_20190405.xlsx
* 8. FMEA\_20190405.xlsx

3) Design

* 9. SoftwareDesignDocument\_20190405.docx
* 9.1. SoftwareStandards\_20190405.docx
* 9.2. NamingConventions \_20190405.docx

4) Verification

* 10.1. BlackboxTest\_baseline.docx
* 10.2. WhiteboxTest\_baseline.docx
* 10.3. CCR\_baseline.docx
* 11.1. IntegrationTesting\_baseline.docx
* 11.2. ValidationTesting\_baseline.docx
* 11.3. ThroughputRAMFlash\_procedure.docx

4) Verification\Results

* 10.1. BlackboxTest\_20190720.docx
* 10.2. WhiteboxTest\_20190720.docx
* 10.3. CCR\_20190720.docx
* 11.1. IntegrationTesting\_20190720.docx
* 11.2. ValidationTesting\_20190720.docx

# Development methodology

Durante el proyecto se utilizó para llevar el orden de tareas la metodología Agile Scrum, usando como herramienta la extensión diseñada para agile de GitHub, llamada ZenHub, en ella se realizó la creación de tickets, épicas y sprints.

La duración de los sprints para la programación fue de pocos días, dado que la herramienta fue utilizada para llevar los cambios de software que hicimos, y dado que no comprendíamos el funcionamiento de la herramienta Git, comenzamos a usarla cuando el código estaba cercano a ser la versión final.

Posiciones: Scrum Master: Gabriel Solís

Developers: José Antonio Pintor

Product owner: Marcos Samuel Peña

In case the team select SCRUM Methodology, it **SHALL** specify the controls like:

* Scrum board,
* Length of the sprint.
* Schedule of the Meetings.
* Positions: Scrum masters, product owner and developers.
* Planning board.

# Estimates

**HECHOS.**

Tarjeta de desarrollo:

Se dispone de una tarjeta Renesas Synergy SK-S7G2, con su cable de datos USB para la conexión con la PC, transmisión de datos, alimentación eléctrica y el debuggeo de programas.

Planta de control:

Se dispone de una planta de control automático, la cual consta de un motor de DC, con aspas plásticas integradas como carga del motor, un sensor de efecto Hall integrado, una tarjeta de potencia y cables para su conexión con la tarjeta de control.

Computadora de escritorio:

Se cuenta con una PC Hewlett Packard, con potencia de procesamiento más que suficiente para ejecutar el software con el que trabajamos para desarrollar el software de control de velocidad requerido por el diplomado.

Dispositivos de medición y suministro de energía:

Se cuenta con dispositivos electrónicos que ayudaron a realizar las mediciones eléctricas necesarias para llevar a cabo el proyecto integrador. El equipo electrónico consta de un multímetro digital, una fuente de potencia con múltiples canales, un generador de funciones, y un osciloscopio para medir las señales eléctricas provenientes de las salidas del microcontrolador.

**SUPOSICIONES.**

Hardware dañado:

El caso de que el hardware estuviera dañado nos habría afectado de gran manera puesto que recibimos la planta de control de manera tardía y una demora más de tiempo por esta razón habría sido de gran impacto para el proyecto.

El tiempo de uso del laboratorio fue escaso las primeras veces que comenzamos a utilizarlo, ya que no contábamos con la tarjeta, la planta de control, o los cables de los dispositivos de medición, por lo tanto, no podíamos llevar a cabo actividades relacionadas a construir el proyecto. Fue semanas después que pudimos tener acceso completo al equipo al igual que trabajar en el laboratorio las tardes entre semana.

El equipo de trabajo, constituido por los dos desarrolladores anteriormente mencionados tuvieron bastante tiempo para trabajar en el proyecto, puesto que al trabajar en la misma empresa tienen comunicación todos los días laborales y pueden compartir ideas y soluciones para los problemas presentados.

El hardware que fue prestado se encontró en buen estado en general, a excepción de los cables de conexión de la tarjeta de potencia, ya que estos estaban rotos, sin terminal de conexión o incorrectamente ordenados según la hoja de datos de la tarjeta de potencia.

Software nuevo:

El ambiente de desarrollo de Renesas e2 studio es nuevo para el equipo de trabajo y ya que cuenta con una gran cantidad de herramientas de compilación y debuggeo, será de cierta dificultad para ellos utilizarlas todas, sin embargo, las más comunes son sencillas de ocupar y muy intuitivas para el usuario.

El hardware no presenta demasiada dificultad de comprender puesto que se cuenta con una hoja de datos de conexión de la tarjeta de potencia, lo cual hará que su comprensión sea mejor.

Algunos de los riesgos que presenta el proyecto se refieren a la perdida de clases y por ende de hora de labor en el proyecto debido a actividades laborales fuera de la ciudad. El aprender a utilizar el software de control de versiones es algo nuevo para ambos integrantes del equipo, esto podría provocar confusión en uso o el surgimiento de problemas inesperados.

**ESTIMADOS DE ACTIVIDADES.**

Recursos humanos:

Dado que seguimos una estrategia de desarrollo basada en Agile Scrum, el product owner del proyecto integrador fue el tutor de nuestro equipo el Ing. Marcos Samuel Peña, el rol de Scrum Master estuvo a cargo del ing. Gabriel Solís, quien también fungió como desarrollador al igual que el ing. Antonio Pintor.

Se contó con dos desarrolladores para el proyecto integrador aquí presentado.

Software:

Los desarrolladores cuentan con experiencia en la programación estructurada, sin embargo, no tienen conocimiento de las sentencias de código utilizadas en concreto con este microcontrolador, por lo tanto, deberán invertir tiempo en la lectura, investigación y análisis de ejemplos (notas de aplicación) de las funciones que requiere el microcontrolador para funcionar.

El uso del software de Renesas fue de utilidad a la hora de realizar la programación de la tarjeta de control, las herramientas que ofrece nos permitieron llevar a cabo el proyecto por medio de threads para realizar las tareas de control y del despliegue de información a través de la pantalla LCD con facilidad.

Documentación.

Entre los desarrolladores a cargo de este proyecto se repartieron tareas referentes a la documentación, aunque hubo tareas que debían desarrollarse por ambos ingenieros, dado que requerían de conocimientos acerca de cada parte de código que cada uno desarrolló.

**ESTIMADOS DE CÓDIGO.**

En cuanto a los módulos que se necesitan programar para el proyecto integrador, se encuentran el uso de un ADC, para la entrada de señal del potenciómetro que se usará para ajustar el valor de referencia del control, varios módulos de temporizadores, para llevar la cuenta de tiempo usado entre los flancos de subida y bajada del sensor de efecto hall, esto para obtener la frecuencia y velocidad del motor, un módulo de señal PWM, un módulo de interrupción que servirá de igual forma para la señal del sensor de velocidad, para la comunicación con la pantalla LCD se ocupara un módulo SPI y el procesamiento de gráficos estará a cargo del módulo GUIX.

* It **SHALL** have a breakdown of all task and activities that are needed and analyzed their dependency between them, some good examples to estimate are:
  + **Activities etimated**
    - **Create and update documents** (design planning verification and so on). Consider the time to create and update documents (SDP, schedule, control code, meetings and peer reviews).
    - **Create, update and execute Verification** **Plan** (white and black test, cyclomatic complexity index calculation, Integration testing, throughput, RAM and FLASH measurement, C99, C11 or other standard evaluation).

# Planning

Se realizó un diagrama de Grantt con las actividades mas importantes del proyecto para realizar, algunas llevaron mas tiempo de lo planeado debido a dificultades con la investigación del código o la búsqueda de como implementarlo, o a la hora de hacer pruebas cada vez que implementábamos código.

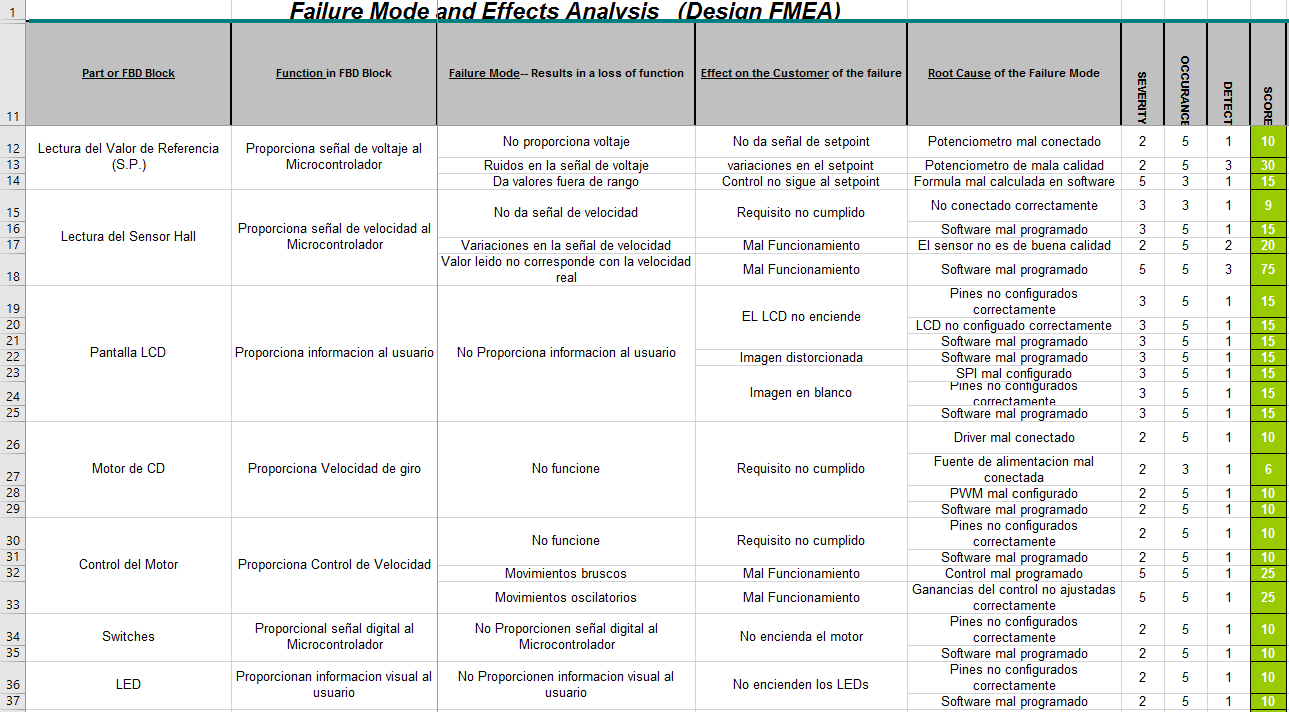
El documento se puede encontrar en la ruta siguiente, dentro de la carpeta principal del proyecto integrador.

PATH\3) Planning\7. Planning\_20190405.xlsx

# Solving Problem Strategy

En esta sección se presenta el análisis de Modos de Falla y Efectos (FMEA) por sus siglas en inglés, dicho documento puede ser encontrado en el siguiente archivo:

<PROJECT\_PATH>\2) Planning\8. DFMEA\_20190405.xlsx

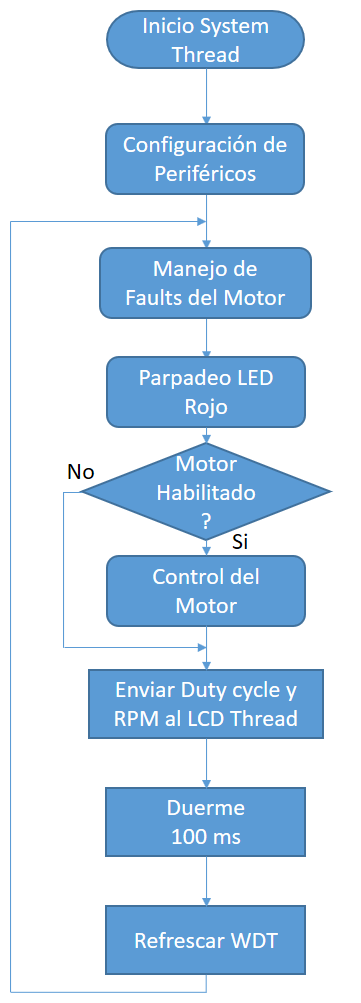


* In case an error be detected during the development stage, this section SHALL contain a mitigation plan including the 5 whys methodology for hw, sw and document issues.

# Design

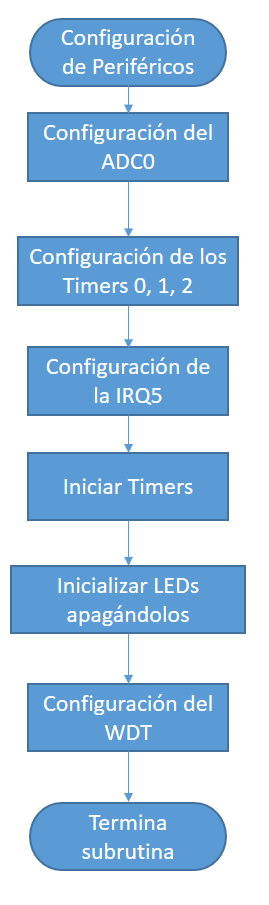
En esta sección se muestran los diagramas de Flujo del diseño del software, así como un diagrama de estados del encendido y apagado del Motor de DC, también se muestra el diagrama a bloques del hardware que visualiza las conexiones entre la tarjeta de Renesas, el módulo de potencia del Motor, el motor, el potenciómetro, el LCD, los switches y los LEDs.

El primero que se muestra es el flujo del programa principal del System\_Thread (Figura 9.1), que es el que contiene la parte de lectura del potenciómetro, lectura del sensor Hall, lectura de los switches, el control del motor, la salida del PWM y la actualización de LEDs entre otros.



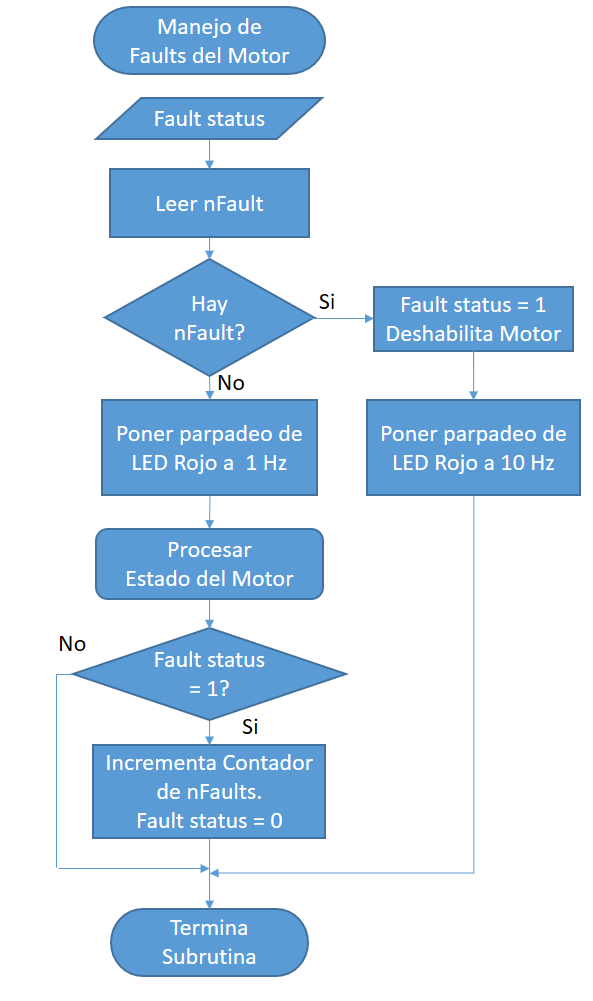
**Figura 9.1**

La Figura 9.2 muestra la configuración del ADC, de los Timers, la interrupción externa para detectar los pulsos del sensor Hall, la configuración del Watchdog timer e inicialización de los LEDs en apagados.



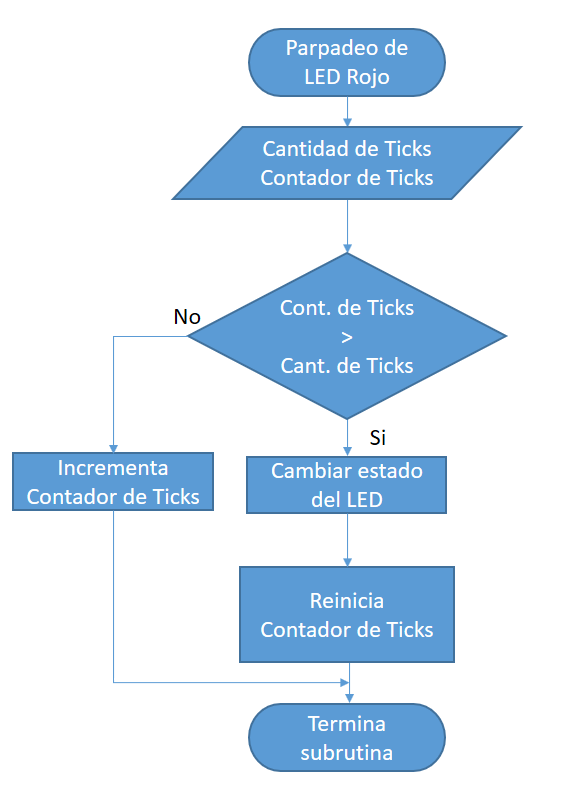
**Figura 9.2**

La Figura 9.3 muestra la subrutina del manejo del nFault del Motor, si hay un fault se apaga inmediatamente el motor y el LED rojo destella a una frecuencia de 10 Hz, cuando el motor regresa a su estado normal, el destello del LED rojo cambia su modo normal de 1 Hz y se ejecuta de nuevo la subrutina del estado del Motor la cual se encarga de monitorear los switches para encender o apagar el motor.



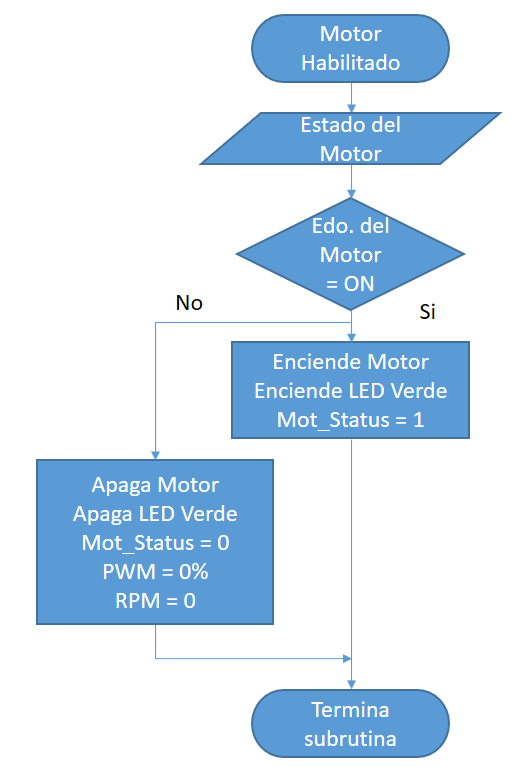
**Figura 9.3**

La figura 9.4 muestra la subrutina para el destello o parpadeo del LED Rojo, que está en función del valor que tenga la variable global “cantidad de Ticks” la cual el LED cambiara de valor cada que el contador de Ticks sea igual a la cantidad de Ticks.



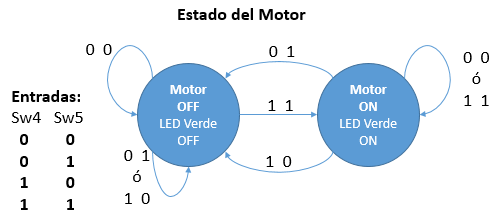
**Figura 9.4**

La figura 9.5 muestra la subrutina que enciende o paga el motor dependiendo del valor que tenga la variable global “Estado del Motor”, si esta es igual a uno, el motor se enciende, así como el LED verde, en caso de que el valor sea igual a cero tanto el motor como el LED verde se apagan y los valores que se envían al LCD también son iguales a cero.



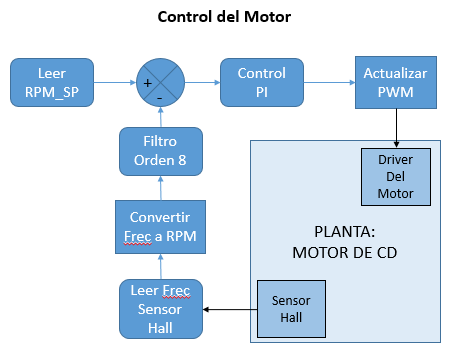
**Figura 9.5**

La figura 9.6 muestra un diagrama de estados que de acuerdo a los valores que tengan las entradas de los switches Sw4 y Sw5 será el estado del Motor: Encendido a apagado.



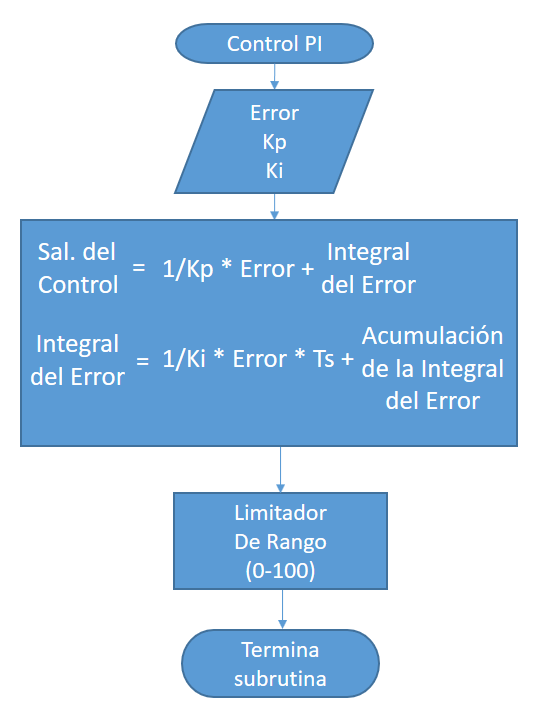
**Figura 9.6**

La figura 9.7 muestra el control del Motor el cual tiene como entradas el valor de setpoint que viene del potenciómetro leído por el ADC y del valor obtenido por el sensor Hall, el control seleccionado fue un control PI, la parte derivativa de un control PID no se está usando ya que este no se recomienda cuando hay variaciones o ruido en la señal que proviene de la planta ya que la parte derivativa amplifica estas variaciones. La salida del control PI va a un PWM ya que la velocidad será directamente proporcional al porcentaje de trabajo de la señal cuadrada que estará entre el rango de 0 a 100%, y la frecuencia del PWM (canal 2) se mantiene constante a 1 KHz.



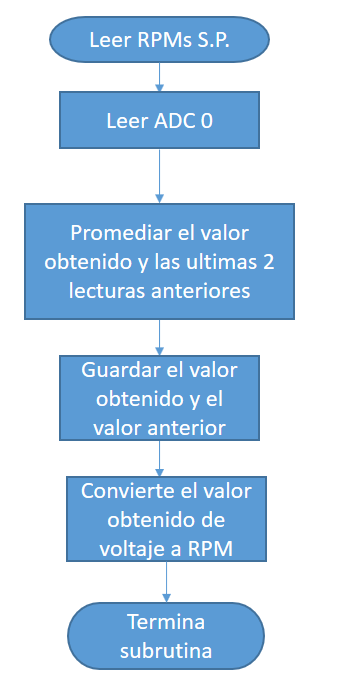
**Figura 9.7**

Figura 9.8 muestra el algoritmo del PI, el control PID es de los más comúnmente usados ya que para este tipo de controles no es necesario conocer el modelo de la planta, simplemente se hace un ajuste o sintonización de las ganancias aplicando algún método por ejemplo el de Ziegler y Nichols, cuyo método se explicará más adelante.



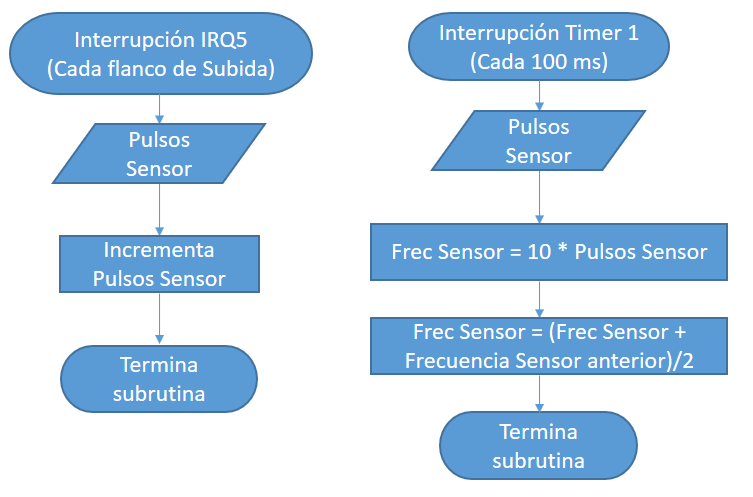
**Figura 9.8**

La figura 9.9 muestra la subrutina para leer el valor del set point en RPMs, este se realiza a través de un potenciómetro conectado al ADC canal 0, como se observa en la figura este obtiene el promedio de las 3 últimas lecturas obtenidas del potenciómetro para disminuir las variaciones, y el valor resultante es convertido a RPM.



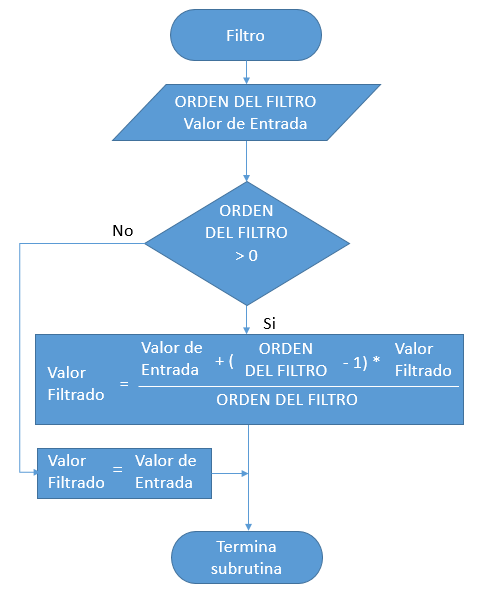
**Figura 9.9**

La figura 9.10 muestra las dos interrupciones que son utilizadas para leer el valor del sensor en RPM, la interrupción externa IRQ5 está configurada para detectar flancos de subida de la señal cuadrada proveniente del sensor, cada que ocurre la variable Pulsos Sensor es incrementada en uno, y la interrupción del timer 1 se configuró para que cada 100 ms haya un servicio de interrupción en donde se obtendrán la cantidad de pulsos leidos en este periodo, luego este valor es convertido a frecuencia y finalmente se obtiene el promedio de los dos últimos valores de frecuencia obtenidos.



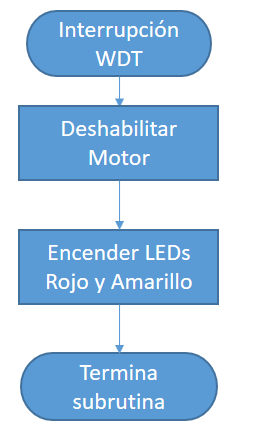
**Figura 9.10**

La figura 9.11 muestra la implementación de un filtro de orden 8, ya que en las pruebas obtenidas de la lectura del sensor se observaron variaciones aún con el promedio de las dos últimas muestras de la frecuencia.



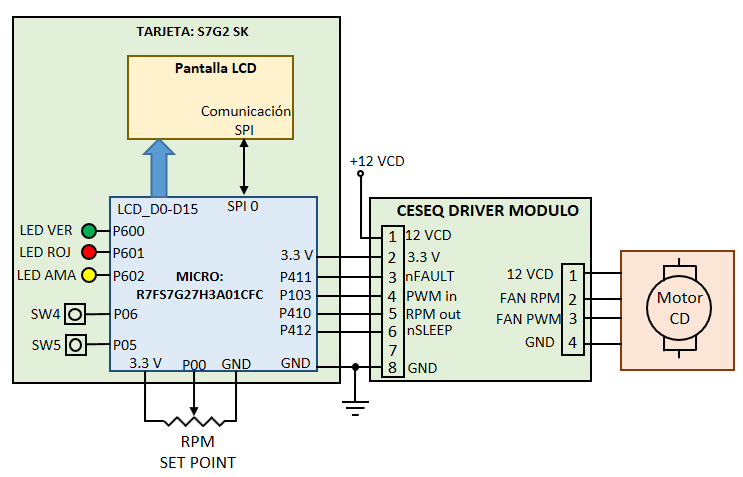
**Figura 9.11**

Finalmente, en la figura 9.12 se muestra la interrupción del watchdog timer, la cual cuando no se refresca el WDT debido a un estancamiento o bloqueo en el código, esta será llamada y apagará inmediatamente el motor y encenderá los LED Rojo y Amarillo.

****

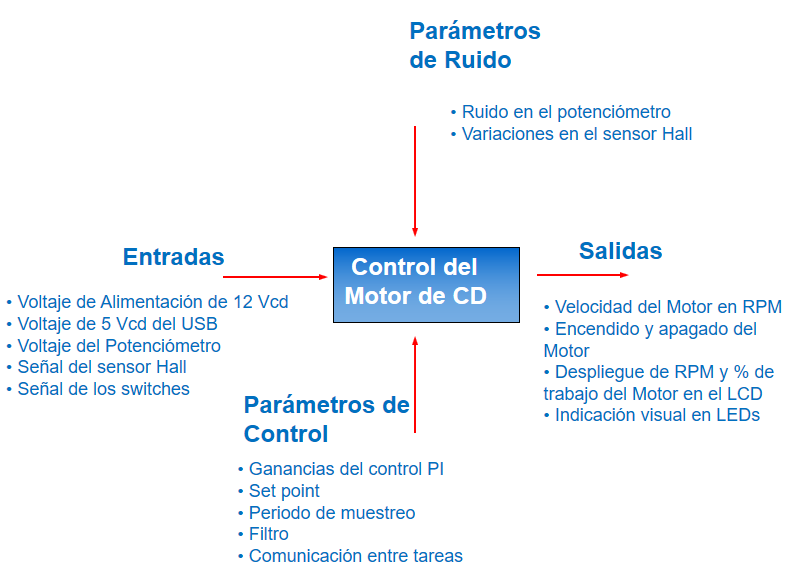
**Figura 9.12**

La figura 9.13 muestra un diagrama de bloques de las conexiones entre la tarjeta de Renesas, el módulo de potencia, el Motor, el potenciómetro, el LCD, los switches y los LEDs.



**Figura 9.13**

Adicionalmente en la figura 9.14 muestra el diagrama de control donde se define las entradas, las salidas, el ruido y las señales de control.



**Figura 9.14**

This section MUST be contained in this document or in a different document linked to this section, the new document SHALL be contained at:

<PROJECT\_PATH>\3) Design\9. SoftwareDesignDocument\_20190405.docx

Sections 9.1. and 9.2. MUST be contained in this document or MUST be divided into different documents. With the naming defined in every section.

## Standards

El software de desarrollo e2 studio de Renesas, tiene incluida una librería de lenguaje basado en la norma C99, la cual evalua el código buscando que cumpla con tal norma.

La herramienta usada para aplicar este estándar fue el software de entorno de desarrollo Renesas e2 studio.

In case C89-C90, C11 or other standard be used, it SHALL be specified in this section and additionally add the link to the standard used.

Additionally, the tool used to evaluate the standard SHALL be defined here if apply.

This section MUST be contained in this document or in a different document linked to this section, the new document SHALL be contained at:

<PROJECT\_PATH>\3) Design\ 9.1. SoftwareStandards\_20190405.docx

## Naming conventions

El propósito del presente documento es presentar el estándar que se debe de utilizar para codificar un programa en lenguaje C, independientemente de la plataforma que se trate se deben de seguir ciertas reglas al momento de desarrollar software, el objetivo de estas es que el programa sea claro y explícito.

**Convención para los nombres de variables**

Cada palabra de una variable debe empezar con Mayúscula después del prefijo del tipo de variable.

El uso del guión bajo podrá ser empleado para separar las palabras, por ejemplo:

u16Frec\_PWM

## Prefijo para el identificador de alcance de una variable.

El identificador o nombre de la variable deberá tener un prefijo que determine su alcance. El tipo de alcance podrá ser global o local. El alcance por default de una variable es **Global**. Este identificador deberá ser el primer carácter en cualquier nombre que se le vaya asignar a una variable.

|  |  |  |
| --- | --- | --- |
| **Identificador de alcance** | **Prefijo** | **Ejemplo** |
| Alcance local | L minúscula | uint8\_t u8Mot\_status |
|  | Ninguno | uint16\_t u16RPM |

## Prefijos para el nombre de una variable.

Los identificadores (nombre) de las variables deberán ser predefinidos con caracteres que representen el tipo de variable. Esto es representado en la siguiente tabla. A diferencia de la definición del tipo de variable que es con mayúscula, el nombre de la variable empieza con el tipo de variable en minúscula. Esto es:

int16\_t tipo de variable en mayúsculas.

i16Ctrl\_Out Nombre de la variable, empieza con el tipo, pero en minúsculas.

| Caracteres empleados | Tipo Nombre Variable | Comentario |
| --- | --- | --- |
| u8 | uint8\_t u8Pulses; | unsigned char |
| i8 | int8\_t i8Input\_Value; | signed char |
| u16 | uint16\_t u16Frec\_Sensor; | unsigned short |
| i16 | int16\_t i16Integr\_Error; | signed short |
| u32 | uint32\_t u32Counts; | unsigned long |
| i32 | int32\_t i32Control\_Out; | signed long |
| f32 | float f32Value\_Filtered; | Float |
| f64 | double f64Variable; | Double |
| Array…………..a | uint16\_t au16Send\_Data[2] | uint16\_t es usado para determinar la longitud del arreglo. |
| Struct………….s | struct ssp\_err\_t sError; | Define sError como una estructura del tipo ssp\_err\_t |
| Register………Regn | #define Reg8TxD\_Data u0tb | Define el registro datos para Tx de n=8 bits |
| Pointer……………p | uint8\_t \* pu8Window; | Generalmente, p es combinado con otro carácter de prefijo. (En este ejemplo ‘pu8’ denota un puntero a un tipo uint8\_t.) |

## Convención para los nombres de las funciones.

El identificador de la función deberá ser **FN**. Después del prefijo FN debe de seguir un guión bajo, y posteriormente el tipo de dato que regresa la función.

El nombre de una función deberá consistir de un pronombre que describa el sujeto de la función y un verbo que describa la acción que realizará la función. Por ejemplo para la función que verifica si el dispositivo I2C está listo deberá ser nombrada.

uint16\_t FN\_u16PI\_Control(int16\_t li16Error)

## Convención para los nombres de las Subrutinas.

De forma similar a las funciones, se deberán de seguir las siguientes reglas para la definición de subrutinas.

El identificador de las subrutinas deberá ser **SR**.

Después del prefijo SR debe de seguir un guión bajo. Evitar separar el nombre de la subrutina con el guion bajo. Ejemplo:

void SR\_Motor\_Control(void)

A diferencia de las funciones una subrutina no regresa valor alguno, pero si puede contener parámetros en el llamado de esta como en una función.

## Convención general para nombres

Se deberá tener en cuenta al momento de nombrar ya sean variables o funciones, para que el nombre tenga la mayor claridad posible. Siguiendo los siguientes puntos:

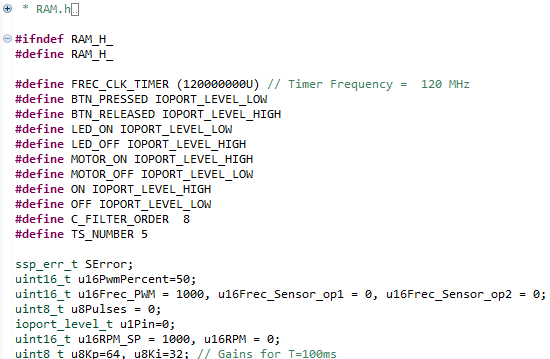
* Los nombres deberán ser lo suficientemente explícitos sin sobrecargar el uso de verbos. Por ejemplo, u32Message\_Queue es más significativo que u32Queue.
* El ahorrarse algunos caracteres y usar abreviaturas se debe de evitar, esto se reflejará al momento de leer y entender el código, lo cual tiene mayor prioridad que el ahorro de caracteres en el nombramiento de variables y funciones.

## 

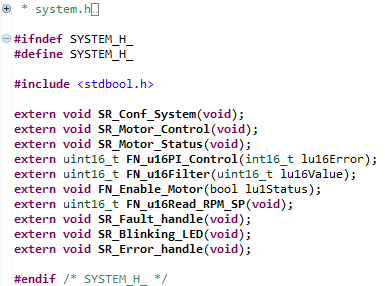
## Estructuración del programa

En esta sección se presenta la estructura general que debe de seguir cualquier programa implementado en lenguaje C en un microcontrolador.

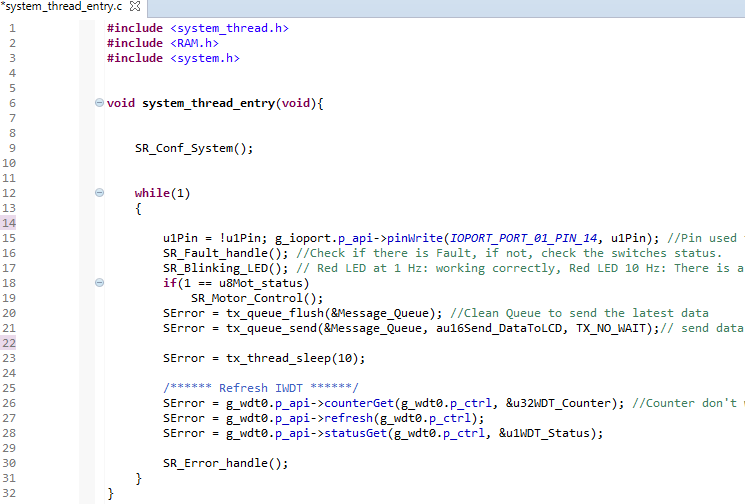
Se incluyen los .h específicos del micro, las constantes y variables Globales en el encabezado RAM.h y además este RAM.h se usara en todos los .c



En los .h de cada .c se ponen como extern sus funciones y subrutinas para que los demás .c puedan hacer referencias a ellas y para que el propio .c las tenga como prototipo.



En cada .c se incluyen solo las referencias de los .h que contengan las funciones y subrutinas que necesitan



This section MUST be contained in this document or in a different document linked to this section, the new document SHALL be contained at:

<PROJECT\_PATH>\3) Design\9.2. NamingConventions\_20190405.docx

In code comments, It SHALL contain the requirements which is implemented with the code described.

# Testing

## Verification strategy (black box test)

This section SHALL be contained at:

<PROJECT\_PATH>\4) Verification\10.1. BlackboxTest\_baseline.docx

…and its results SHALL be located with the date as suffix, as following is indicated:

<PROJECT\_PATH>\4) Verification\Results\10.1. BlackboxTest\_20190405.docx

Every time a module or feature is implemented, it SHALL contain their tests section and SHALL be contained with the reference to the requirement number in order to have traceability.

## White box strategy

It SHALL define the software which is going to be used, for instance: gtest, junit, sunit, etc.

A document baseline SHALL be created as a reference for all the project implementation. This document SHALL be located at:

<PROJECT\_PATH>\4) Verification\10.2. WhiteboxTest\_baseline.docx

…and its result SHALL be located at:

<PROJECT\_PATH>\4) Verification\Results\10.2. WhiteboxTest\_20190405.docx

Every time a module or feature is implemented, every test case SHALL contain a reference to the requirement number in order to have traceability.

## Cyclomatic Complexity Redundance index

Esta seccion esta contenida en la siguiente dirección

<PROJECT\_PATH>\4) Verification\ 10.3. CCRI\_20190405.docx

Los resultados de esta sección estan contenidos en la siguiente dirección

<PROJECT\_PATH>\4) Verification\Results\10.3. CCRI\_20190405.docx

# Release

Firmware version number SHALL be defined in this section, and the strategy used for that, an example MUST be:

Naming convention for delivered work products like: code and documents shall be defined in this section, the name shall be kept for those documents that SDP describes.

Date/Hw version/Sw version

20190405/001/ 001

The code shall be controlled in GITHUB and path shall be defined here.

## Software Development Folder

The path for software development folder shall be defined in this section and be contained and controlled at GITHUB previous to the final release.

## Integration Tests Strategy

This section SHALL be contained in the planning and reflected in the schedule.

IT **SHALL** be defined a document baseline as a reference for all the project implementation. This document **SHALL** be located at:

<PROJECT\_PATH>\4) Verification\11.1. IntegrationTesting\_baseline.docx

…and its RESULT SHALL be located at:

<PROJECT\_PATH>\4) Verification\Results\11.1. IntegrationTesting\_20190405.docx

Every time a module or feature is implemented, every test case SHALL contain a reference to the requirement number in order to have traceability.

This test MUST contain the plant connected or not.

## Validation Testing / Functional Testing

This section SHALL be contained in the planning and reflected in the schedule.

IT **SHALL** be defined a document baseline as a reference for all the project implementation. This document **SHALL** be located at:

<PROJECT\_PATH>\4) Verification\11.2. ValidationTesting\_baseline.docx

…and its RESULT SHALL be located at:

<PROJECT\_PATH>\4) Verification\Results\11.2. ValidationTesting\_20190405.docx

Every time a module or feature is implemented, every test case SHALL contain a reference to the requirement number in order to have traceability.

This test SHALL contain the plant connected.

## Throughput and Flash and RAM measurement

This section SHALL be contained in the planning and reflected in the schedule.

It SHALL define the RAM, Flash and Throughtput measurements strategy at:

<PROJECT\_PATH>\4) Verification\ 11.3. ThroughputRAMFlash\_procedure

# Results

All pictures, videos or miscellaneous SHALL be posted at:

<PROJECT\_PATH>\5) Results

# Lessons Learned

All comments, feedback or others SHALL be documented in this section.