**UNIVERSIDAD TECNOLÓGICA DE QUERÉTARO**

**CESEQ**



**Diplomado en Software Embebido**

Software design for a DC motor speed controller

DOCUMENT: Software Development Plan #CESEQ001

Scrum Master/Developer/Testing Engineer: Cerecero, Jenny

Master/Developer/Testing Engineer: Quiroz Diana

Approver: Avila, Francisco

Date (20190809): 20190405

# Log

|  |  |  |  |
| --- | --- | --- | --- |
| Version | Date (20190809) | Description | Reviewer |
| 0.0.0 | 20190724 | First release | Cerecero Jenny |
| 0.0.1 | 20190725 | SDP Update | Quiroz Diana |
| 0.0.2 | 20190725 | SDP Update | Quiroz Diana |
| 0.0.3 | 20190726 | SDP Update | Cerecero Jenny |
| 0.0.4 | 20190729 | Adding STD | Cerecero Jenny |
| 0.0.5 | 20190730 | Updates to SDP | Quiroz Diana |
| 0.0.6 | 20190805 | Updates | Quiroz Diana |
| 0.0.7 | 20190809 | Final Updates | Jenny Cerecero |

# Index

Table of Contents

[1. Log 2](#_Toc16263256)

[2. Index 3](#_Toc16263257)

[Table of Contents 3](#_Toc16263258)

[3. Project Scope 5](#_Toc16263259)

[4. Deliverables 5](#_Toc16263260)

[4.1 Documents 5](#_Toc16263261)

[4.2 Code 7](#_Toc16263262)

[5. Development methodology 7](#_Toc16263263)

[Scrum Board 7](#_Toc16263264)

[Positions 8](#_Toc16263265)

[Schedule 8](#_Toc16263266)

[6. Estimates 9](#_Toc16263267)

[7. Planning 11](#_Toc16263268)

[8. Solving Problem Strategy 13](#_Toc16263269)

[9. Design 13](#_Toc16263270)

[System Diagram 13](#_Toc16263271)

[Software Diagram 14](#_Toc16263272)

[ADC 14](#_Toc16263273)

[User Interface 14](#_Toc16263274)

[Controller 15](#_Toc16263275)

[Flow Diagram 15](#_Toc16263276)

[Control Diagram 16](#_Toc16263277)

[Call Graph 16](#_Toc16263278)

[Task Scheduler 17](#_Toc16263279)

[9.1. Standards 18](#_Toc16263280)

[9.2. Naming conventions 19](#_Toc16263281)

[10. Testing 20](#_Toc16263282)

[10.1. Verification strategy (black box test) 20](#_Toc16263283)

[10.2. White box strategy 20](#_Toc16263284)

[11. Release 20](#_Toc16263285)

[11.1. Integration Tests Strategy 21](#_Toc16263286)

[11.2. Validation Testing / Functional Testing 21](#_Toc16263287)

[11.3. Flash and RAM measurement 21](#_Toc16263288)

[Flash 21](#_Toc16263289)

[RAM 22](#_Toc16263290)

# Project Scope

Description of the full project can be observed in the following document:

*<PROJECT\_PATH>/1) Requirements/stakeholder/20190501c Requisitos del proyecto integrador CESEQ.docx*

The project was completed with changes to the original scope specified in the aforementioned document. Those changes are listed below:

* Set Point adjustment through UART port: This project did not consider the adjustment of the Set Point using this communication protocol. Set Point is adjusted exclusively through a potentiometer.

Additionally, for those modules where the project description provided several options to proceed, the selections made by this team are listed below:

* Control Algorithm: The close-loop algorithm used for this project is of the PID type.
* Time management for processes: A scheduler was used to execute the system’s tasks.

The requirements for this project were defined in two documents:

1. *<PROJECT\_PATH>/1) Requirements/Requisitos\_ProyectoIntegrador\_DSE.pdf*
2. *<PROJECT\_PATH>/1) Requirements/Requisitos\_SW\_ProyectoIntegrador\_DSE.pdf*

The first document contains the System Requirements, while the second document contains the Software Requirements. These documents were generated with the help of an additional software, *ReqView v2.6.2*, which helped us to enumerate the requirements while they were being generated.

It should also be noted that the changes to the original scope and selections mentioned above were done in agreement with the customer and are reflected in the requirement documents.

# Deliverables

## 4.1 Documents

The following table contains the list of documents that will be delivered for this project. Please note that the document location is relative to the path where this document is placed (*<PROJECT\_PATH>*).

|  |  |  |
| --- | --- | --- |
| **Document Name** | **Short Description** | **Location** |
| Diagrama\_EntradasSalidas.pdf | System diagram for the project (Inputs/Outputs) |  |
| Diagrama\_Flujo.pdf | Flow chart for the code |  |
| DSE\_Gantt | Gantt Project Planning |  |
| DSE\_SW\_PI\_140619\_001 | Software diagram |  |
| Matriz\_Riesgos.xlsx | Risk Management File |  |
| Minutas.docx | Document containing dates, topics and resolution for meetings with tutor. |  |
| Requisitos\_ProyectoIntegrador\_DSE.pdf | System Requirements |  |
| Requisitos\_SW\_ProyectoIntegrador\_DSE.pdf | Software Requirements |  |
| Statement Of Work.docx | Document containing details about how risks, reviews and changes were handled. |  |
| Software Design Document | Contains system and software diagrams. |  |
| Software Standards | Contains details about the software standard used for this project. |  |
| Naming Conventions | Contains details about the naming conventions used inside the code. |  |
| Blackbox Test Baseline | Details Blackbox strategy. |  |
| Whitebox Test Baseline | Details Whitebox strategy. |  |
| Integration Test Baseline | Details Integration testing strategy. |  |
| Validation Test Baseline | Details Validation strategy. |  |
| ThroughputFlashRAM Test Procedure | Details test procedures and results for the throughput, Flash and RAM measurements. |  |
| Blackbox Test Results | Blackbox Test Results |  |
| Whitebox Test Results | Whitebox Test Results |  |
| Integration Test Results | Integration Test Results |  |
| Validation Test Results | Validation Test Results |  |
| Auditoria de calidad V1.0.docx | Format for the Project’s evaluation |  |

## 4.2 Code

The complete RENESAS project can be found in the following zip file:

*<PROJECT\_PATH>/3) Design/DSE\_Project.zip*

Although several files are contained inside the RENESAS project, only those manually modified are listed in the table below. Please note that the files location is relative to the zip file *(<PROJECT\_PATH>/3) Design/DSE\_Project.zip*).

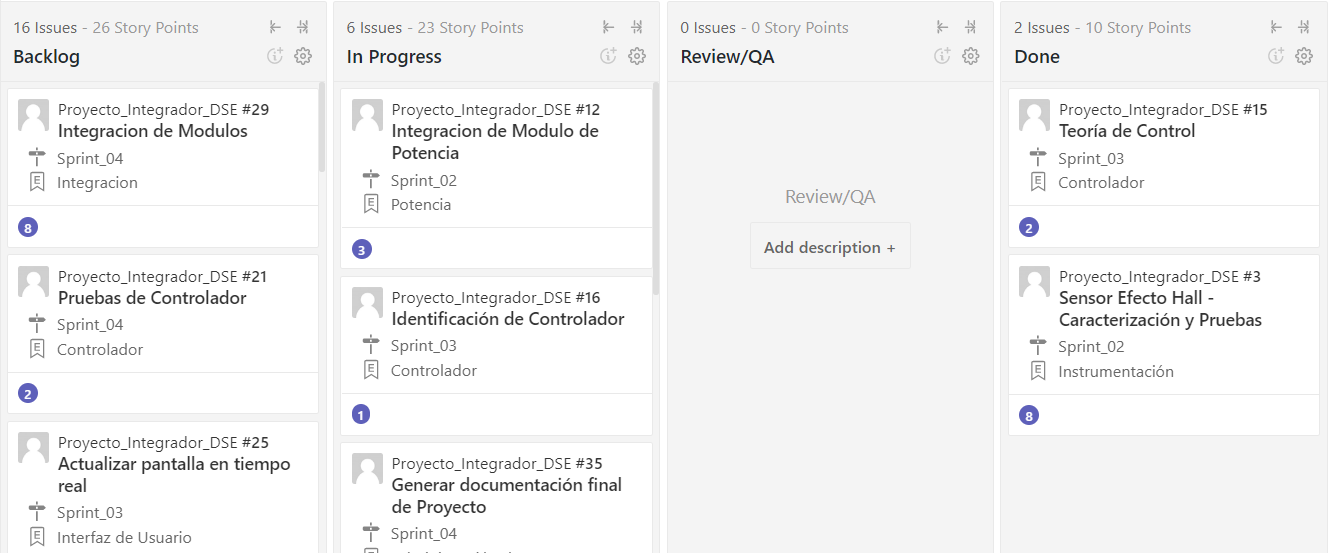
|  |  |  |
| --- | --- | --- |
| **File Name** | **Short Description** | **Location** |
| guiapp\_event\_handlers.c | Event handlers for actions related to the LCD display | src/guiapp\_event\_handlers.c |
| main\_thread\_entry.c | Main function. Timers, Interruptions, PWM are initialized. | src/main\_thread\_entry.c |
| common.h | Name definitions used throughout the RENESAS project. | src/common.h |
| adc.c | Reading of potenciometer signal and converting it and sampling the readings. | src/adc.c |
| adc.h | Definition of functions used for the adc implementation. | src/adc.h |
| controller.c | Calculate and get PWM; PID controller implementation: sintonized variables (kp, ki and kd), calculation of error. | src/controller.c |
| controller.h | Definition of functions used for the controller implementation. | src/controller.h |
| diagnostics.c | Get Setpoint, short to battery, short to ground. | src/diagnostics.c |
| diagnostics.h | Definition of functions used for the diagnostics implementation. | src/diagnostics.h |
| sensor.c | Signal processing of the hall effect sensor. Get current speed | src/sensor.c |
| sensor.h | Definition of functions used for the sensor implementation. | src/sensor.h |

# Development methodology

The methodology used for this project was SCRUM, the details of the implementation of the methodology can be seen below.

## Scrum Board

The Scrum Board was managed through ZenHub, the agile project management within GitHub. Here, all the activities defined for the project were monitored and their progress was tracked through their status in the board.



## Positions

Following the SCRUM methodology, the following roles were defined:

* **Product Owner**: Francisco Ávila (Tutor)
* **Scrum Master**: Cerecero Jenny, Quiroz Diana
* **Developers**: Cerecero Jenny, Quiroz Diana.

Since the team only consists of two team members, the role of *Scrum Master* was exchanged between team members with each sprint. Something similar was applied to the role of *Developer*, where both team members needed to complete the activities defined at the beginning of each Sprint.

## Schedule

Together with the Product Owner, it was defined that each Sprint would last two weeks. At the end of each Sprint we would schedule a Sprint Review where the progress on the stories defined for that sprint would be reviewed. Depending on the current progress and the comments from the Product Owner, we adjusted the activities planned for the next Sprint. This was repeated throughout the development of the project.

Additionally, there was a meeting every week (whenever possible) between the Developer, Scrum Master and the Product Owner to review progress and overall doubts about the project. These meetings helped to make sure the team was progressing in the right direction and allowed us to identify and define actions for possible risks that could impact the project.

The topics discussed in these weekly meetings were documented in a Word file, which can be found in the following path:

*<PROJECT\_PATH>/2) Planning/Minutas.docx*

# Estimates

Before making the time estimates for the activities required to complete the project, an analysis was done to identify the facts, assumptions and risks involved; this with the intention of making time estimates closer to reality, organize activities in such way that dependencies do not interfere with deliveries, and identify potential risks at early stages of the planning to be able to mitigate them with the minimum impact possible.

The facts considered for the project are the following:

* There is a workstation available with the following components:
  + PC
  + Internet connectivity
  + Oscilloscope
  + Function Generator
  + DC Power Supply
  + Multimeter
  + SK-S7G2 RENESAS Board
* The team will have a plant available with the following components:
  + Potentiometer
  + H-Bridge Motor Driver
* The following software will be available for the team to use:
  + Matlab
  + E2 Studio IDE
  + Git Hub
* Human Resources:
  + Product Owner (Tutor)
  + 2 Team Members acting as SCRUM Master and Developer

The assumptions made for the project are the following:

* Hardware is available and in good conditions.
* Software is available and has the necessary functionality to aid with the project’s deliverables.
* Laboratory will be available to work on the project on Fridays from 02:00pm to 06:00pm.
* Team members will be available to work on the project on Fridays from 02:00pm to 06:00pm.
* Project requirements will not change from those defined at the beginning of the course.

The risks identified for the project are the following:

* Unable to work with RENESAS Board outside of UTEQ
* Inaccurate planning due to lack of knowledge on project’s complexity.
* Uncalibrated sensor
* Unstable controller
* Damaged components
* Change in requirements
* Team members’ unavailability to work on project outside of Fridays

These risks were monitored throughout the development of the project using a risk matrix. This file can be found in the following path:

*<PROJECT\_PATH>/2) Planning/Matriz\_Riesgos.xlsx*

Considering all the previous points, it was possible to make estimates for the activities that needed to be done in order to complete the project. Using the SCRUM methodology, the epics and stories shown in the table below were defined. The “Estimate” column reflects the latest estimate done for the stories; these points, which represent hours, were adjusted during each Sprint.

|  |  |  |
| --- | --- | --- |
| **Activity** | **Type** | **Estimate** |
| Requirements Analysis | Epic |  |
| Read Requirements | Story | 2 |
| Requirements Definition | Story | 6 |
| Instrumentation | Epic |  |
| Hall Effect Sensor Tests | Story | 5 |
| Sensor Characterization | Story | 2 |
| Potentiometer Tests | Story | 2 |
| Potentiometer Characterization | Story | 1 |
| Display Tests | Story | 6 |
| Power | Epic |  |
| H-Driver Tests | Story | 3 |
| H-Driver Characterization | Story | 3 |
| Motor Tests | Story | 2 |
| Power Module Integration | Story | 3 |
| Controller | Epic |  |
| Control Theory Overview | Story | 3 |
| Identify Plant | Story | 2 |
| Obtain Plant’s transfer function | Story | 3 |
| Simulate plant | Story | 2 |
| Develop control algorithm | Story | 2 |
| Implement controller | Story | 5 |
| Test controller | Story | 3 |
| User Interface | Epic |  |
| Set display | Story | 3 |
| Test display | Story | 2 |
| Diagnostics | Epic |  |
| Short to Battery | Story | 3 |
| Short to Ground | Story | 3 |
| Memory corruption | Story | 2 |
| Button in short circuit | Story | 2 |
| Integration | Epic |  |
| Integration of modules | Story | 5 |
| Hardware Tests | Epic |  |
| Software Tests | Epic |  |
| Black Box Tests | Story | 6 |
| White Box Tests | Story | 6 |
| Additional Tests | Story | 2 |
| Project Management | Epic |  |
| Develop SDP | Story | 5 |
| Develop SVP | Story | 5 |
| Develop Architecture document | Story | 3 |
| Generate additional documentation | Story | 8 |

These epics and stories were saved in a GitHub repository, where the tracking was being monitored in order to analyze the progress done throughout the development of the project.

# Planning

The epics were assigned due dates in order to keep track of their progress and their estimated date of completion. This information was then placed in a Gantt chart, which was presented to the customer so he could also evaluate the progress of the project. The Gantt chart with the final plan and dates can be found in the following path:

*<PROJECT\_PATH>/2) Planning/DSE\_Gantt.pod*

The stories defined in the previous section were divided between the team members in order to be completed. The team member who was responsible of doing the necessary activities to complete the task was assigned the role of *developer*, while the other team member was assigned the role of *reviewer/tester*, who made sure the “Definition of Done” was fulfilled for that particular story.

The table below shows the developer assigned to the story as well as its definition of done, which would be used by the reviewer/tester as a parameter to validate that the story was actually complete.

|  |  |  |  |
| --- | --- | --- | --- |
| **Activity** | **Type** | **Developer** | **Definition of Done** |
| Requirements Analysis | Epic |  |  |
| Read Requirements | Story | Both | Read UTEQ’s document about project. |
| Requirements Definition | Story | Both | Upload requirements document to Github. |
| Instrumentation | Epic |  |  |
| Hall Effect Sensor Tests | Story |  | Observe in oscilloscope an error lower than 10% in measurements. |
| Sensor Characterization | Story |  |
| Potentiometer Tests | Story |  | Observe in multimeter an error lower than 10% in measurements. |
| Potentiometer Characterization | Story |  |
| Display Tests | Story |  | Upload project with customized display to Github. |
| Power | Epic |  |  |
| H-Driver Tests | Story |  | Pass (whitebox/blackbox/ integration/validation) tests |
| H-Driver Characterization | Story |  |
| Motor Tests | Story |  | Pass (whitebox/blackbox/ integration/validation) tests |
| Power Module Integration | Story |  |
| Controller | Epic |  |  |
| Control Theory Overview | Story |  | Identify possible controller types to use. |
| Identify Plant | Story |  | Generate control diagram. |
| Obtain Plant’s transfer function | Story |  | Document transfer function in requirements. |
| Simulate plant | Story |  | Generate plots in Matlab. |
| Develop control algorithm | Story |  | Upload code to Github. |
| Implement controller | Story |  | Integrate controller code. |
| Test controller | Story |  | Pass (whitebox/blackbox/ integration/validation) tests |
| User Interface | Epic |  |  |
| Set display | Story |  | Configure display information according to requirements. |
| Test display | Story |  | Integrate display code. |
| Diagnostics | Epic |  |  |
| Short to Battery | Story |  | Pass (whitebox/blackbox/ integration/validation) tests |
| Short to Ground | Story |  | Pass (whitebox/blackbox/ integration/validation) tests |
| Memory corruption | Story |  | Pass (whitebox/blackbox/ integration/validation) tests |
| Button in short circuit | Story |  | Pass (whitebox/blackbox/ integration/validation) tests |
| Integration | Epic |  |  |
| Integration of modules | Story |  | Pass (whitebox/blackbox/ integration/validation) tests |
| Hardware Tests | Epic |  |  |
| Software Tests | Epic |  |  |
| Black Box Tests | Story |  | Generate Black Box Results document. |
| White Box Tests | Story |  | Generate White Box Results document. |
| Additional Tests | Story |  | Generate Additional Tests document. |
| Project Management | Epic |  |  |
| Develop SDP | Story |  | Complete SDP has been uploaded to Github. |
| Develop SVP | Story |  | Complete SVP has been uploaded to Github. |
| Develop Architecture document | Story |  | Architecture document has been uploaded to Github. |
| Generate additional documentation | Story |  | Additional documentation has been uploaded to Github following folder structure. |

# Solving Problem Strategy

For the Problem Solving Strategy to be used, FMEA was discussed with the customer to be applied only for the driver that contains the sensor as this feature was presenting failures during the development and integration phases of the project, more specifically during the signal processing of the hall effect sensor and during the PID controller sintonization.

The intention of this section is describe a mitigation plan to avoid the failure to happen and a solution in case the failure is presented again.

1. Mitigation plan: Constant monitoring of the amount of current flowing through the system, this will be done by visual inspection at the voltage source.
2. Solution: Talk to another team and agree to exchange the driver while they are not using it.

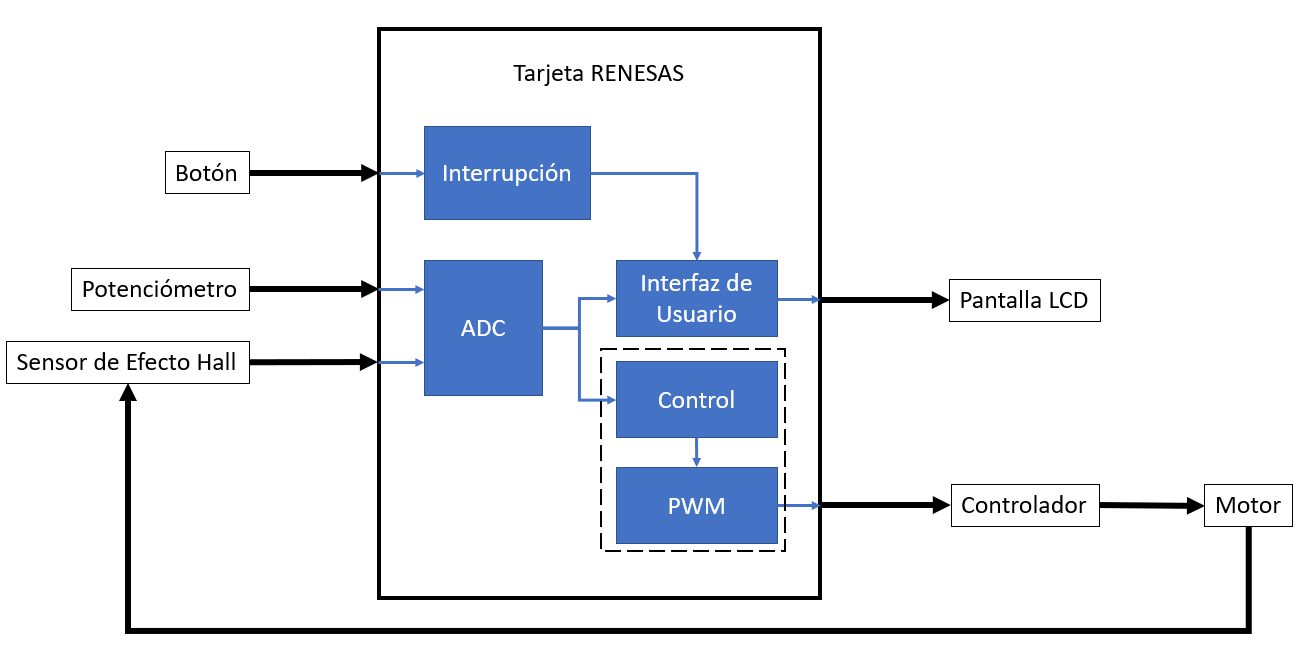
NOTE: The other risks were already identified during the planning phase of the project and can be found in the following document:

*<PROJECT\_PATH>/2) Planning/Matriz\_Riesgos.xlsx*

# Design

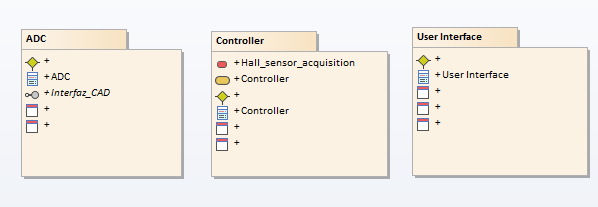
## System Diagram

The System Diagram below identifies the inputs and outputs of the system based on the project’s requirements (image in Spanish).



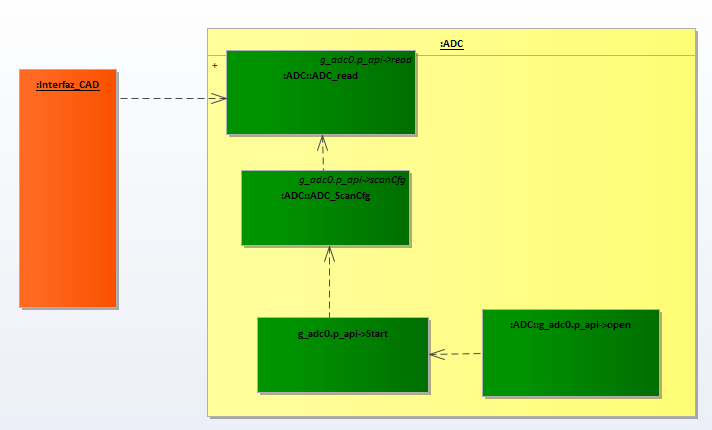
## Software Diagram

To develop the Software Diagram, the *Enterprise Architect v13.5.1351* software was used to identify the modules that would have to be implemented, including their functions, inputs and outputs.



### ADC

For the ADC module, two inputs were considered, one coming from the potentiometer and another one from the Hall Effect Sensor. This module contains three functions, one to read the incoming ADC value, another one to validate the reading and finally one to store the value in its corresponding variable, which could be either POT\_VAR or HFS\_VAR, the outputs of the module.

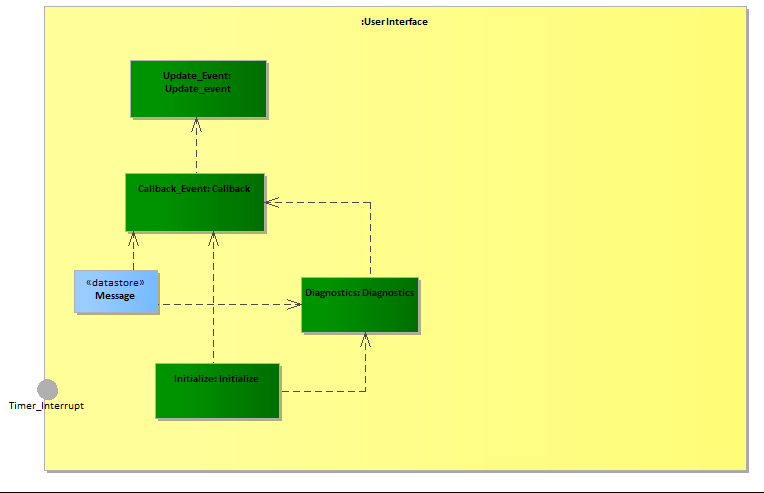




### User Interface

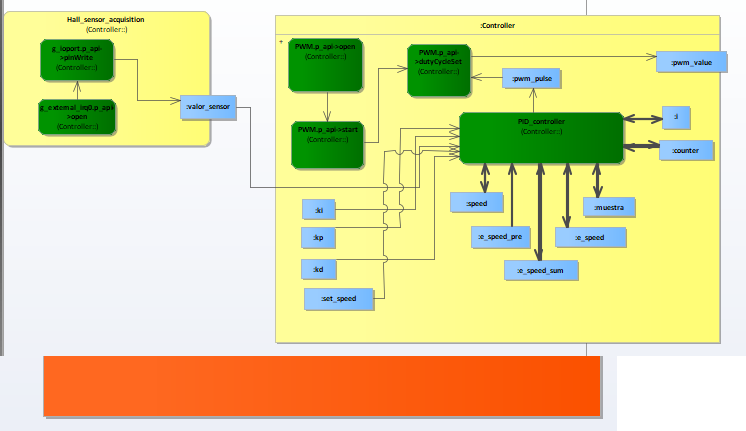
For the User Interface module, its inputs will be the outputs from the ADC module together with the button status that will change the menu shown in the display and the information from the diagnostics. This module will read the inputs and, depending on the menu being displayed, will either convert the ADC and Sensor’s value to the set point and speed respectively or will determine if the monitored values lead to a short to battery or short to ground. Finally, one last function will be in charge of displaying the appropriate message on the display.

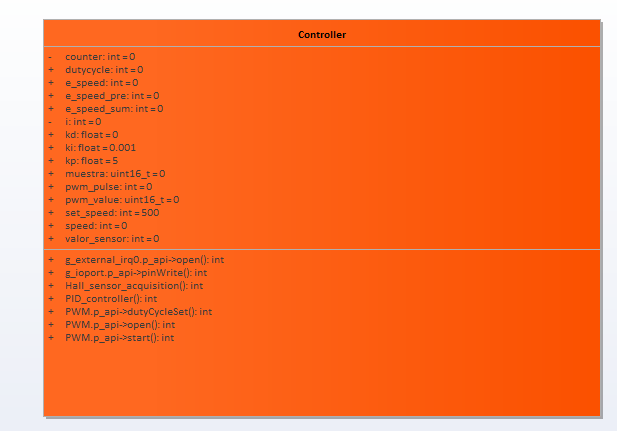




### Controller

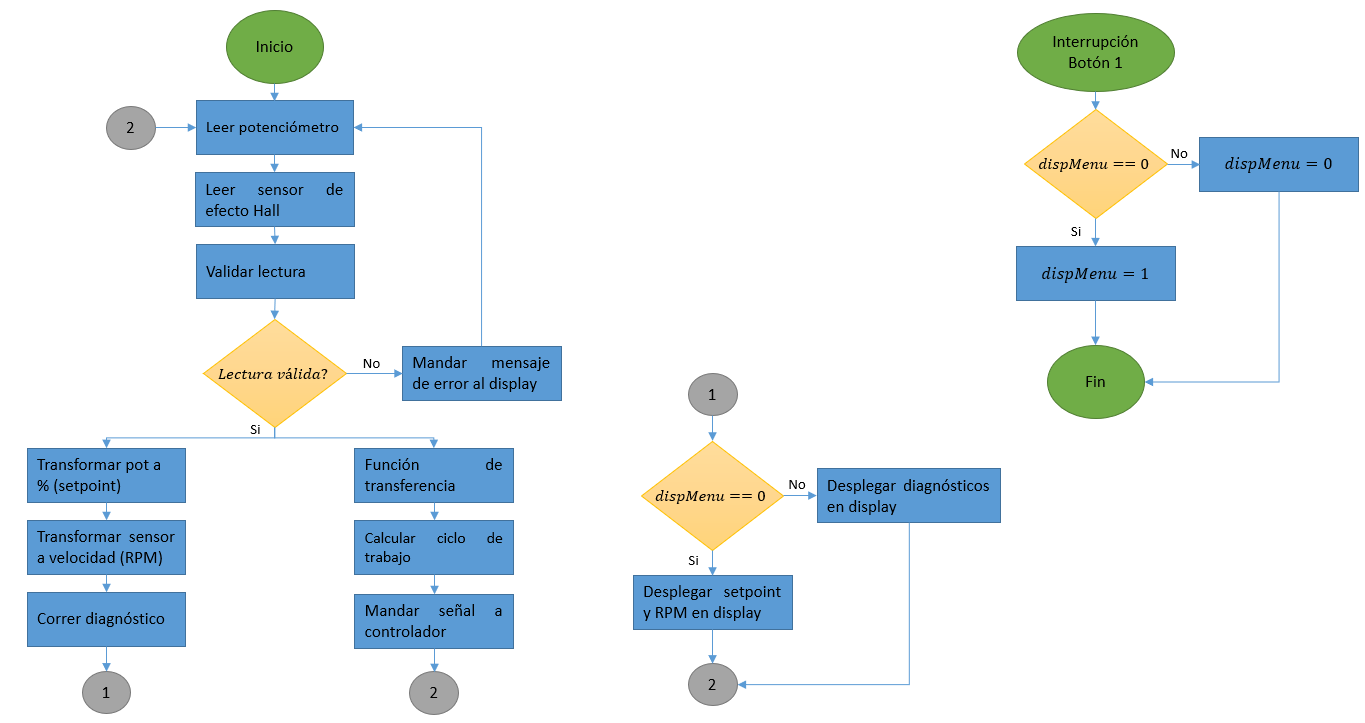
For the Controller module, the outputs from the ADC (which have already been validated) will be the inputs. These signals will be read, then they will be used for the control algorithm in order to define the new speed that must be set for the motor and finally that speed will be converted to a duty cycle that will be assigned to the RENESAS output set as PWM.



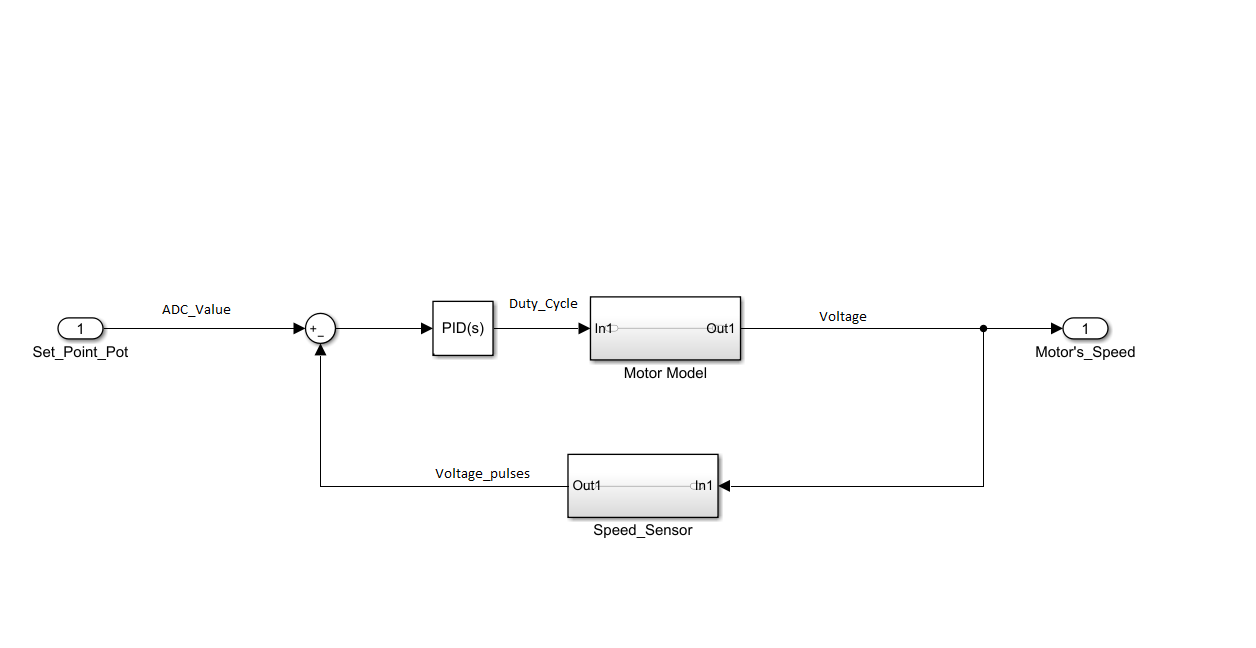


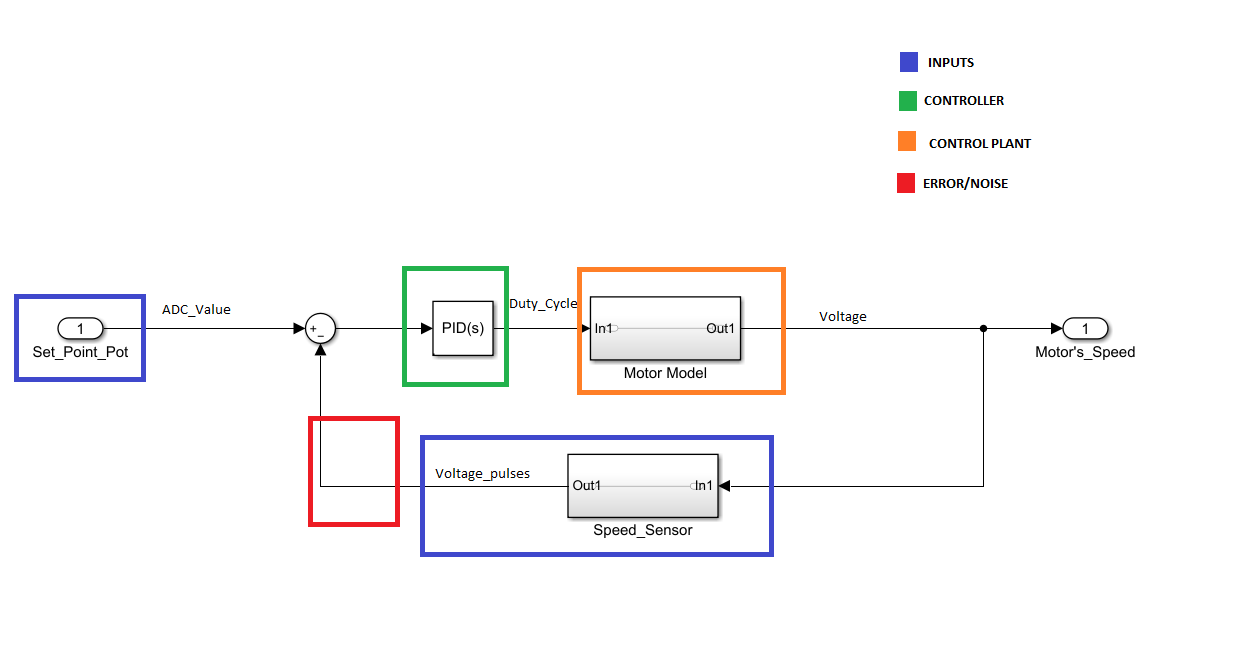
## Flow Diagram

The flow diagram below shows what the program will do throughout its execution. First, it will read and validate information coming from the potentiometer and the Hall Effect sensor, then it will do several tasks: calculate the set point, calculate the actual speed, run the diagnostics, run the controller algorithm, and calculate the new duty cycle; then it will send this signal to the power driver. A digital button configured on the display will trigger an interruption that will toggle between the two available menus, either the main menu (showing speed, duty cycle, etc.) or the diagnostics menu.



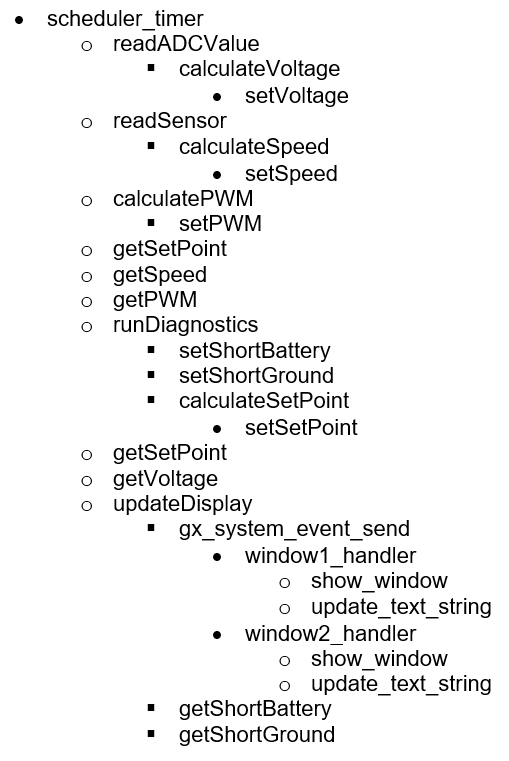
## Control Diagram





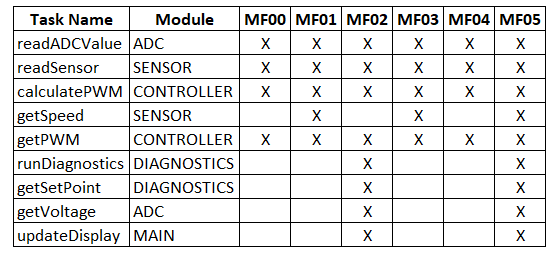
## Call Graph

The call diagram below shows the function’s interaction within the code:



## Task Scheduler

In the table below is found a description of the task scheduler, which consists of six minor frames. It is observed which tasks are executed in each frame and the module they belong to.

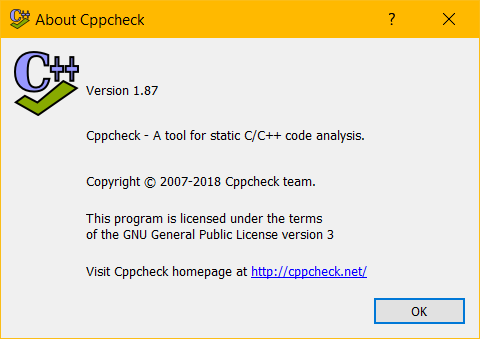


## Standards

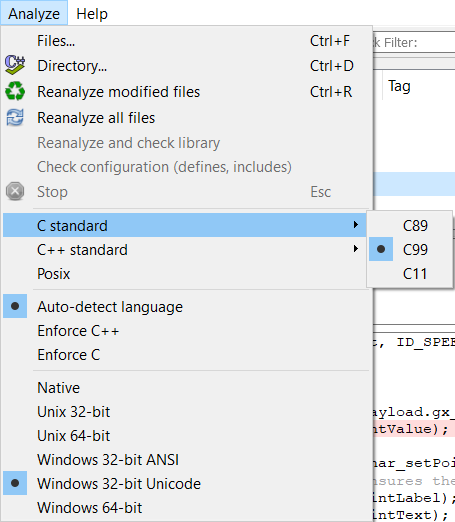
The coding standard used for this project was **C99**, formally known as ISO/IEC 9899:2018. There is no direct link to view this standard, since the ANSI WEBSTORE charges for the ability to download a PDF copy of the document. Instead, the link to the download page is shown below:

<https://webstore.ansi.org/Standards/ISO/ISOIEC98992018>

In order to validate that the standard was followed correctly, an additional software tool was used, *Cppcheck v1.87*:



This program allows users to analyze C code files with a specific standard from its catalogue; in this case, C99 was selected:



All errors or warnings shown by the tool were corrected before delivering the final version of the code.

## Naming conventions

The naming convention shall comply with the following format:

**Local and global variables:** The tags for local and global variables shall contain the data type in lowercase at the beginning then, separated by an underscore is followed the variable name where the first letter of the first word shall be lowercase and the first letter of the second word shall be uppercase as showed in the following format: datatype\_variableName

**Functions:** The tags for functions shall contain the description of the function, in other words, the name shall describe what the function is doing; where first letter of the first word shall be lowercase and the first letter of the second word shall be uppercase as showed in the following format: functionDescription

**Code files:** The code files shall be named in lowercase, in case of having two or more words they shall be separated by an underscore: controller.h, hal\_entry.c

**All Documents:** The name of the documents is related to what it contains, the words are separated by an underscore and the first letter of each word shall be uppercase: Software\_Development\_Plan.docx

# Testing

## Verification strategy (black box test)

Black box testing will be carried out to verify system level requirements and high level requirements. Black Bock tests will be created to perform normal and robustness verification of one or more requirements.

Black box testing will be performed on the Renesas target. Each test will consist on a procedure with steps to follow in order to perform the verification among with expected results to obtain, as well as a trace to the requirements tested.

Test results will be considered as ‘FAIL’ when the actual results of the performed test do not match the ones expected and as ‘PASS’ when they do. The failure of a test can imply one of the following: the requirements are not correct, the code is not correct, the test needs to be modified in order to fully cover the requirement or an additional test is needed.

The back box tests are found in the following path:

*<PROJECT\_PATH>/4) Verification/Black\_Box\_Tests*

## White box strategy

White box testing will be performed to ensure that the software components integrate correctly. The goal of this activity is to verify that the software interactions occur as expected by the software design and requirements and that the couples between components are behaving appropriately.

This activity will verify high or system level requirements which verification cannot be achieved through black box testing. As well as Black Box verification strategy, White Box tests will consist of a procedure of steps to follow and expected results. The same PASS and FAIL criteria as mentioned in section 10.1 is applied to this activity.

The white box tests are found in the following path:

*<PROJECT\_PATH>/4) Verification/White\_Box\_Tests*

# Release

The firmware version number shall start with DSE\_G2 followed by the date (YYYYMMDD) then, the hardware version in three digits (XXX) and finally the Software version in three digits (XXX).

DSE\_G2/Date/Hw version/Sw version

DSE\_G2/YYYYMMDD/XXX/XXX

## Integration Tests Strategy

The Integration Strategy for this project consist in an ascendant integration where each module is integrated once it is developed and tested individually. It starts when the ADC and the Display configurations are done. Then the sensor and motor are integrated and at the end the controlled is incorporated to the system.

Once the hole system is integrated, it shall run during an hour to validate the integration.

The integration was considered as a milestone in the Gantt, and it is located in the following path:

*<PROJECT\_PATH>/2) Planning/Gantt*

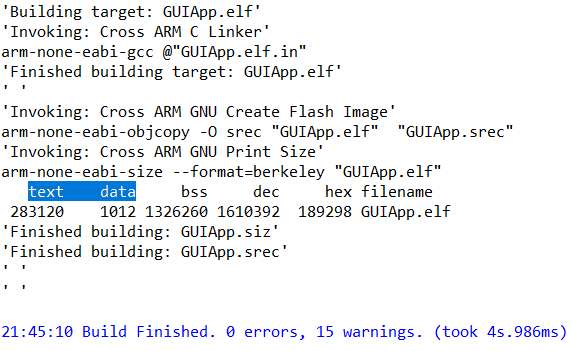
## Validation Testing / Functional Testing

The black box and white box testing cover this section. Refer to sections 10.1 and 10.2 of this documents.

## Flash and RAM measurement

### Flash

Flash memory usage is calculated by RENESAS while building a project and is displayed in the Console. This value can be obtained by adding the *text* and *data* values.



Thus, the final Flash memory usage for this project was **277.47 KB**.

### RAM

Similar to the Flash memory, RAM usage is calculated by RENESAS while building a project and is displayed in the Console. In this case, the usage is obtained by adding the *bss* and *data* values.