**UNIVERSIDAD TECNOLÓGICA DE QUERÉTARO**



SOFTWARE DESIGN DOCUMENT FOR

THE DC MOTOR SPEED CONTROLLER

Embedded Software Course

Ugarte, Gabriela

Chavez, Jose Luis

31/10/2019

## RECORD OF CHANGES

\***A-** ADDED **M**-MODIFIED **D**-DELETED

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **VERSION NUMBER** | **DATE** | **NUMBER OF FIGURE, TABLE, OR PARAGRAPH** | **A \***  **M**  **D** | **TITLE OR BRIEF DESCRIPTION** | **Change Request Number** |
| 1 | 14/09/2019 |  | **A**-Adding Scope and Document Overview and Design subsections |  |  |
| 2 | 28/09/2019 |  |  | Initial Revision |  |
| 3 | 10/20/2019 |  | **A**-Adding all Diagram figures |  |  |
| 3 | 10/28/2019 |  |  | Final Revision |  |

Table of Contents

[RECORD OF CHANGES 2](#_Toc23372207)

[SCOPE 3](#_Toc23372208)

[Document Overview 3](#_Toc23372209)

[Design 3](#_Toc23372210)

[System Diagram 3](#_Toc23372211)

[Software Diagram 4](#_Toc23372212)

[ADC 4](#_Toc23372213)

[User Interface 5](#_Toc23372214)

[Controller 8](#_Toc23372215)

[Flow Diagram 9](#_Toc23372216)

[Control Diagram 10](#_Toc23372217)

[Call Graph 11](#_Toc23372218)

[Task Scheduler 12](#_Toc23372219)

# SCOPE

This Software Design Document (SDD) establishes the design for the software implementation for a DC Motor Speed Controller. The DC Motor Speed Controller is being developed under the direction of the Embedded Software Companies from Queretaro committee. Updates to this SDD will address future upgrades to the DC Motor Speed Controller.

# Document Overview

This SDD identifies all the system and software diagrams for the DC Motor Speed Controller software development.

# Design

## System Diagram

The System Diagram identifies the inputs and outputs of the system based on the project’s requirements (See Figure 1).

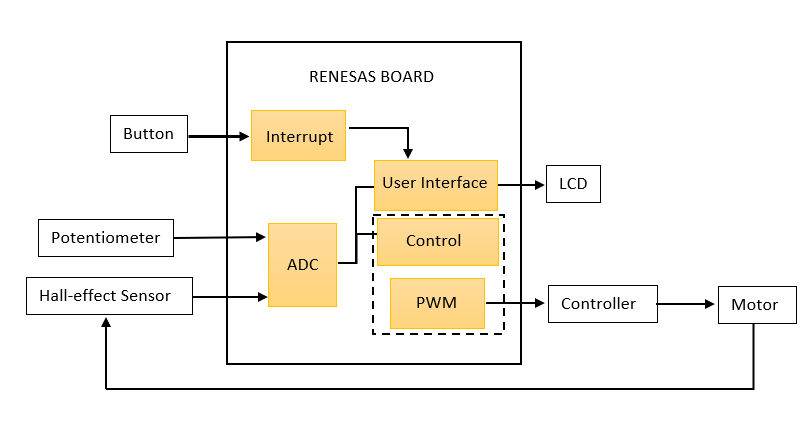


Figure 1. System Diagram

## 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 (See Figure 2).

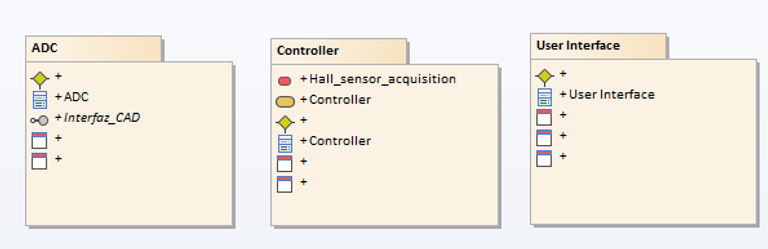


Figure 2. Software Diagram

### 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 (See Figure 3 and Figure 4).

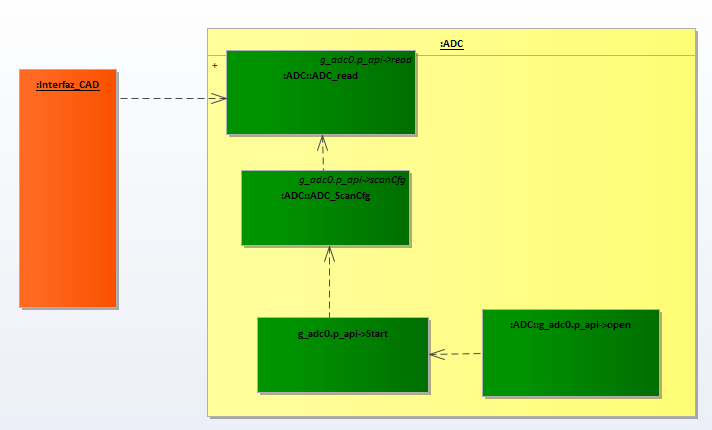


Figure . ADC Diagram



Figure 4. ADC Diagram

### 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 (See Figure 5).



Figure . User's Interface Diagram



Figure 6. User's Interface Diagram

### Controller

For the Controller module, the outputs from the ADC 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.

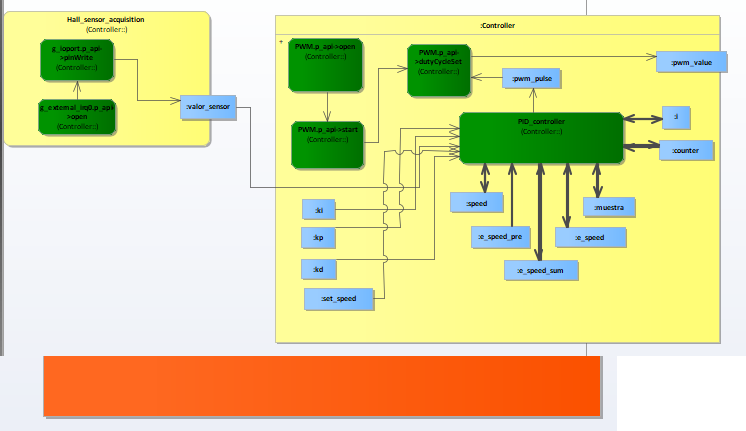


Figure . Controller's Diagram

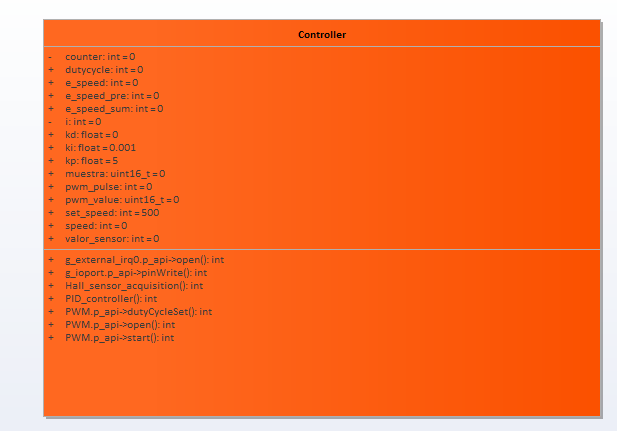


Figure 8. Controller's Diagram

## 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 (See Figure 9 and Figure 10).

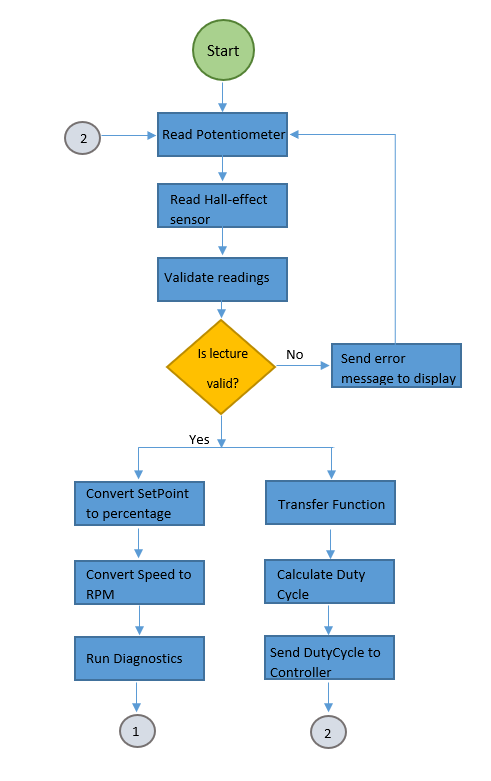


Figure . Flow Diagram Design

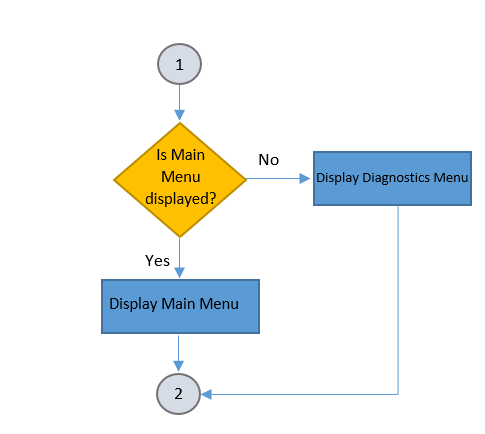


Figure . Flow Diagram Design

## Control Diagram

There are many types of controller that can be used to implement the elegant and effective output. One of them is by using a PI controller. PI stands for Proportional and Integral Controllers which are designed to eliminate the need for continuous operator attention thus provide automatic control to the system.

A proportional controller (Kp) will have the effect of reducing the rise time and will reduce, but never eliminate, the steady-state error. An integral control (Ki) will have the effect of eliminating the steady-state error, but it may make the transient response worse. Effects of each of controllers Kp and Ki on a closed-loop system are summarized in the table shown in Table 1.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Controller respond | Rise Time | Overshot | Setting Time | Error |
| Kp | Decrease | Increase | Small Change | Decrease |
| Ki | Decrease | Increase | Increase | Eliminate |

Table 1. Effects of Kp and Ki

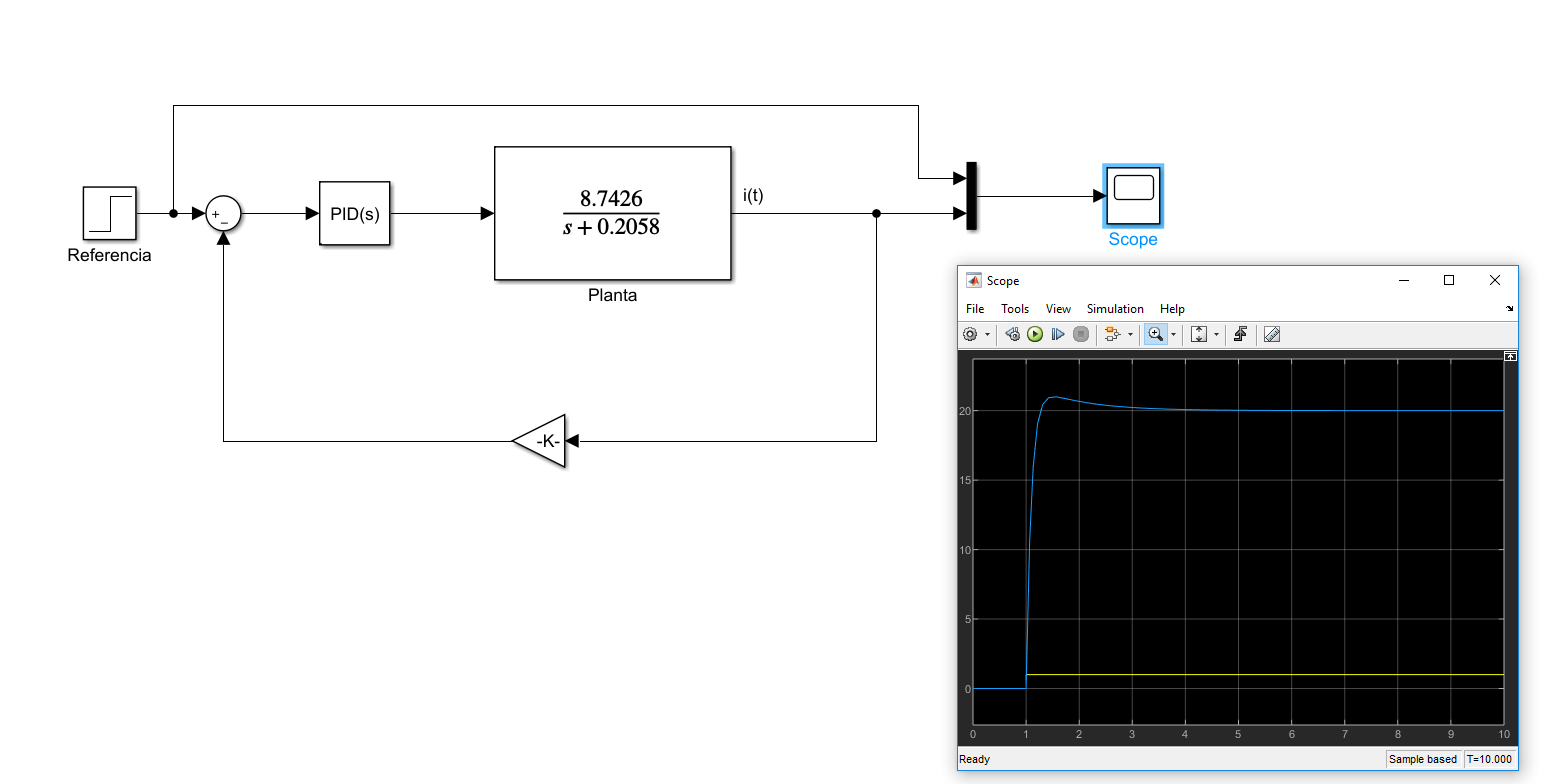
The controller system was modelled in MATLAB Simulink 2019 with a Kp value of 5 and a Ki value of 0.001 (See Figure 11).

Figure 11. PI Controller for the Project

## Call Graph

Figure 12 shows the call diagram with the function’s interaction within the code.

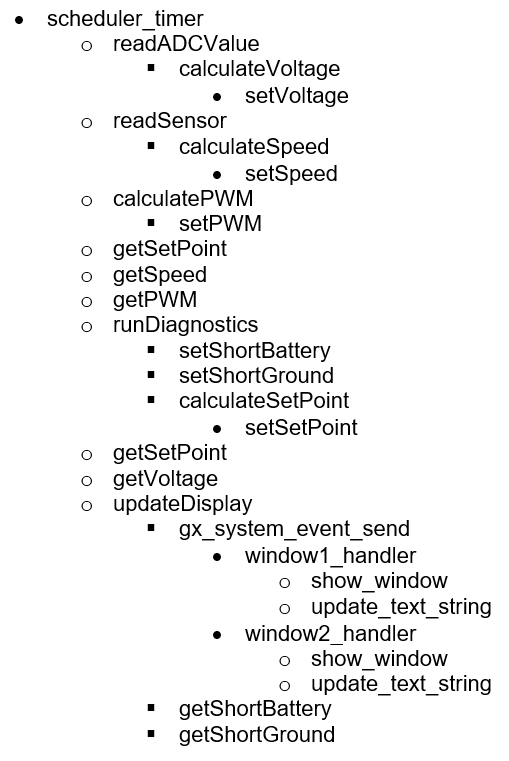


Figure 12. Function's call Diagram.

## 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 (See Figure 13).

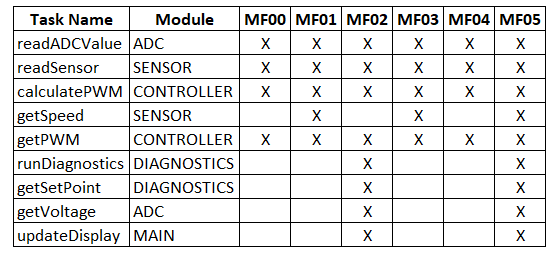


Figure . Task Scheduler