|  |  |
| --- | --- |
| **Title:** | **WINLIFT**  **SW Component < XXXXXXX >** |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **History** | | | | |
| **Issue status**  (Index) | **Maturity/Date**  (draft/invalid/valid)  (dd-mmm-yyyy) | **Author**  Department | **Check/Release**  Department | **Description** |
| 3 | Draft  31-0ct-15 | Oscar Miranda  B.S. | Oscar Miranda  B.S. | Modifications were made in chapter 5:Subtitles, Activity and sequence diagrams added. |

**Table of Contents**

[1 Purpose 3](file:///C:\Users\x\Documents\Workspace_Local\Winlift_Repo\trunk\Setup\Doc\V-Cycle%20Process\2.0%20Design\Software%20Design.docx#_Toc433796033)

[2 Definitions and abbreviations 3](file:///C:\Users\x\Documents\Workspace_Local\Winlift_Repo\trunk\Setup\Doc\V-Cycle%20Process\2.0%20Design\Software%20Design.docx#_Toc433796034)

[3 Realization constraints and targets 3](file:///C:\Users\x\Documents\Workspace_Local\Winlift_Repo\trunk\Setup\Doc\V-Cycle%20Process\2.0%20Design\Software%20Design.docx#_Toc433796035)

[4 SW Conceptual design 4](file:///C:\Users\x\Documents\Workspace_Local\Winlift_Repo\trunk\Setup\Doc\V-Cycle%20Process\2.0%20Design\Software%20Design.docx#_Toc433796036)

[5 SW Component internal breakdown 5](file:///C:\Users\x\Documents\Workspace_Local\Winlift_Repo\trunk\Setup\Doc\V-Cycle%20Process\2.0%20Design\Software%20Design.docx#_Toc433796037)

[5.1 Functional Decomposition 5](file:///C:\Users\x\Documents\Workspace_Local\Winlift_Repo\trunk\Setup\Doc\V-Cycle%20Process\2.0%20Design\Software%20Design.docx#_Toc433796038)

[*5.2* *Function void Close( )* 6](file:///C:\Users\x\Documents\Workspace_Local\Winlift_Repo\trunk\Setup\Doc\V-Cycle%20Process\2.0%20Design\Software%20Design.docx#_Toc433796039)

[*5.3* *Function void Open( )* 6](file:///C:\Users\x\Documents\Workspace_Local\Winlift_Repo\trunk\Setup\Doc\V-Cycle%20Process\2.0%20Design\Software%20Design.docx#_Toc433796040)

[5.4 *Function* void blueLed\_Close( ) 7](file:///C:\Users\x\Documents\Workspace_Local\Winlift_Repo\trunk\Setup\Doc\V-Cycle%20Process\2.0%20Design\Software%20Design.docx#_Toc433796041)

[5.5 *Function* void greenLed\_Open( ) 7](file:///C:\Users\x\Documents\Workspace_Local\Winlift_Repo\trunk\Setup\Doc\V-Cycle%20Process\2.0%20Design\Software%20Design.docx#_Toc433796042)

[5.6 *Void Antipinch()* 7](file:///C:\Users\x\Documents\Workspace_Local\Winlift_Repo\trunk\Setup\Doc\V-Cycle%20Process\2.0%20Design\Software%20Design.docx#_Toc433796043)

[5.7 *Function* void countPressingTime(unsigned int counterTime ) 7](file:///C:\Users\x\Documents\Workspace_Local\Winlift_Repo\trunk\Setup\Doc\V-Cycle%20Process\2.0%20Design\Software%20Design.docx#_Toc433796044)

# Purpose

The purpose of this project is develop a software, for an embedded system, that controls a car window movement, with anti-pitch security function. The software will be implemented in a MPC5606B Freescale development board.

# Definitions and abbreviations

**Definitions**

|  |  |
| --- | --- |
| WINLIFT  GPIO  ISR  STM | Name of the project, which means Window Lifter  General purpose inputs and outputs  Interrupt Service Routine  System Timer Module |
| API | May refer to Application Programming Interface or Application Layer |
| HAL | Hardware Abstraction Layer |
| MCAL | Microcontroller Abstraction Layer |
|  |  |

**Abbreviations**

Only SW Component specific abbreviations.

**References**

|  |  |  |
| --- | --- | --- |
| **N°** | **Document name** | **Reference** |
| 1 | Traceability Matrix Template | 1 |
|  |  |  |
|  |  |  |
|  |  |  |

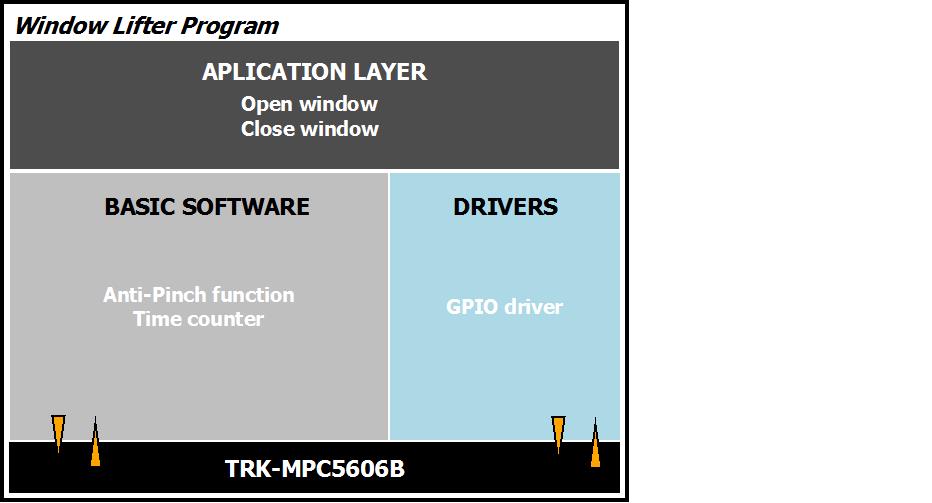
# Realization constraints and targets

The project has several functionalities to control the window, which includes the following ones:

* It will have a function that control the opening of the window.
* It will have a function that control the closure of the window.
* It will have an anti-pinch functionality, declared as interruption, which will stop the closure of the window and will open it. This functionality is for security purposes.
* When anti-pinch is active, a sub-function will disable all inputs for 5 seconds.
* When opening or closing the window there will be an indicator LED indicating the process in progress.
* There will be a function that counts how much time a button have been pressed.

# SW Conceptual design

The next diagram represents the inputs and outputs of the WINLIFT’s conceptual design and the general tasks that must be performed.

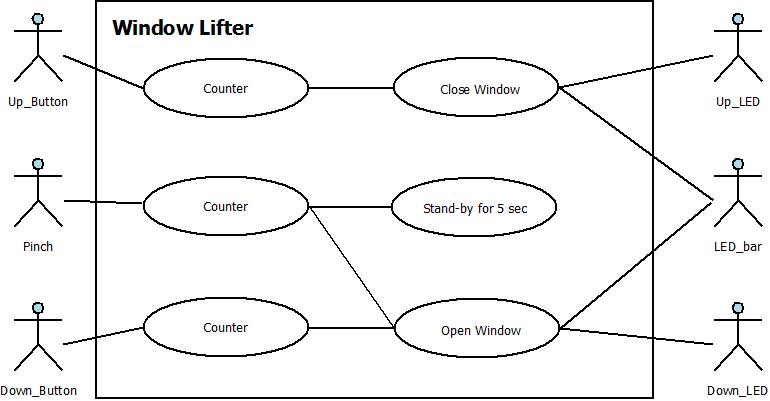


# SW Component internal breakdown

## Diagrams

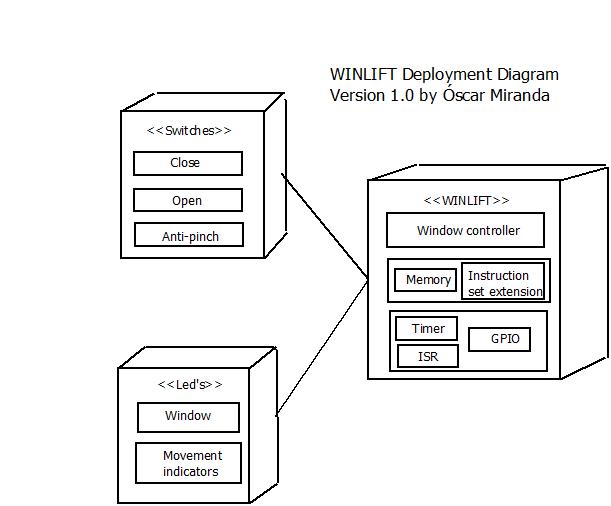
### Use Case Diagram

The following Use Case diagram describes the general interaction between the main actor and the function that will be added to the program. It describes the over-all behavior of the window lift system.



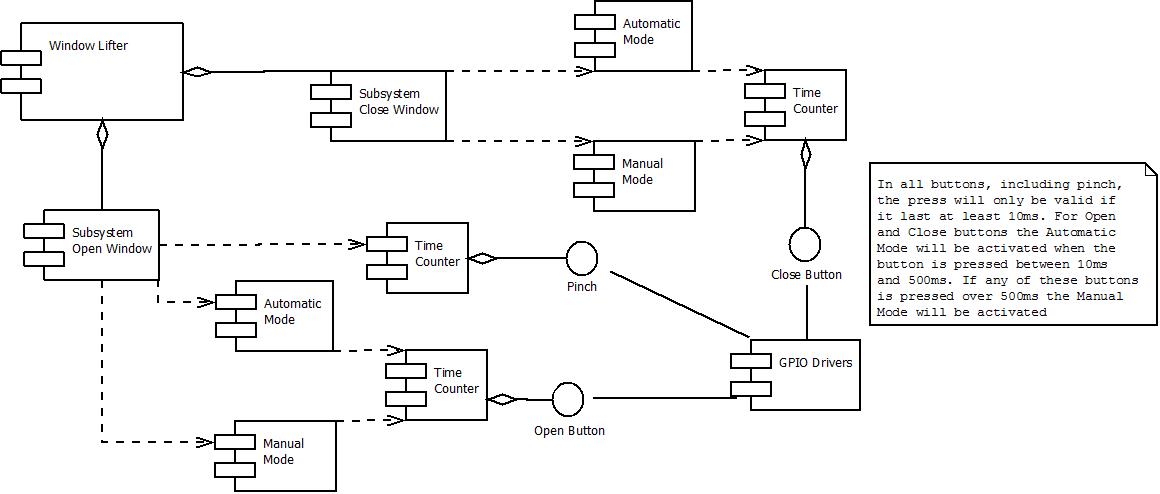
### Deployment Diagram

The following deployment diagram represents the inputs, outputs, the processing module that corresponds to the API, the HAL and the MCAL.

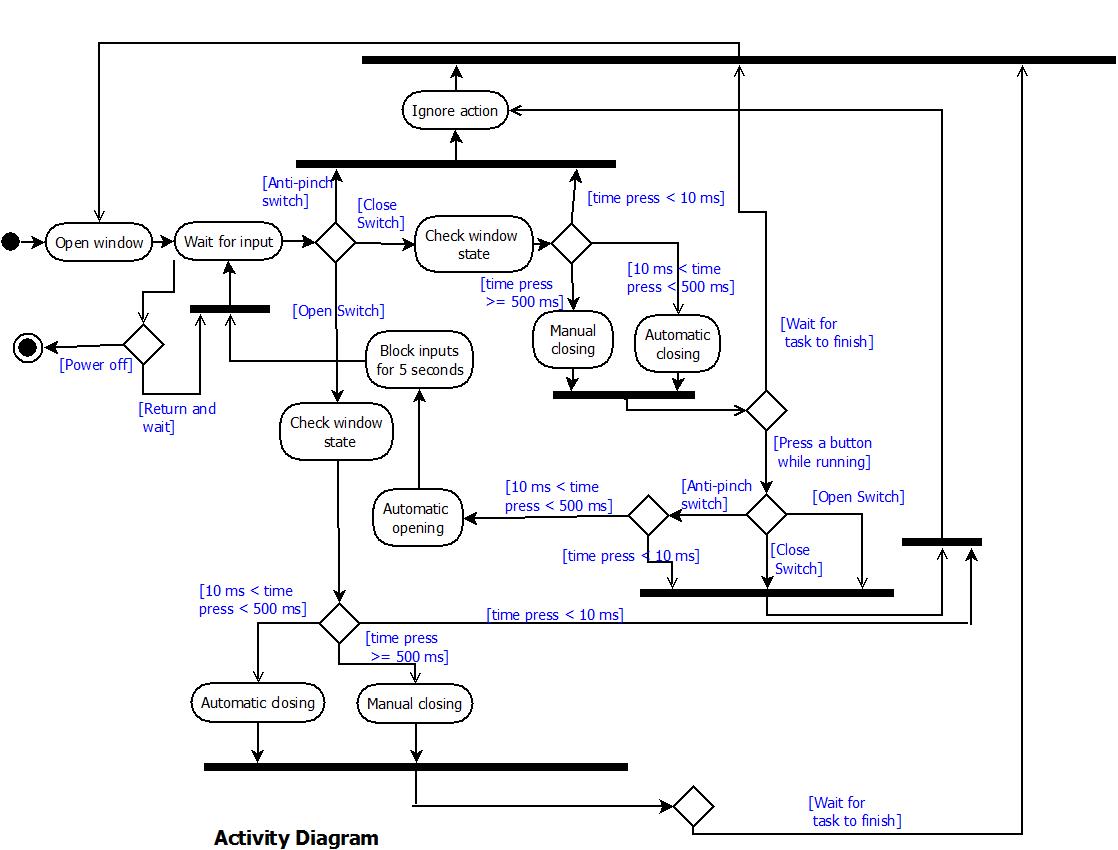


### Component Diagram

The following Component Diagram describes the structure and relations between the sub-systems comprehended in the Window Lifter system.



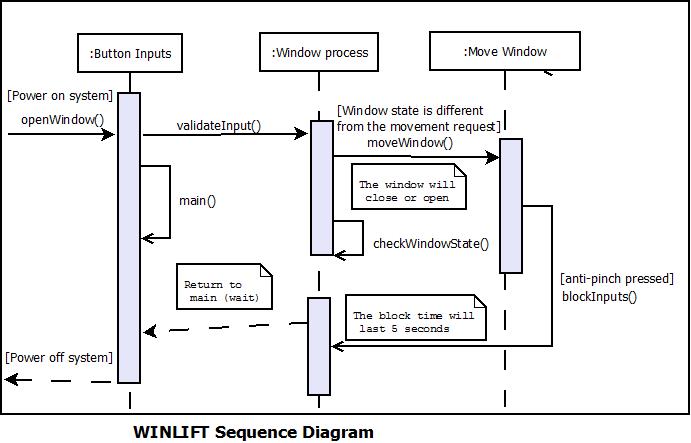
### Activity Diagram

The following flowchart defines all the default and other possible states of the window, the operations like open or close window, the anti-pinch functionality, the validation of a button and the end of the program flow that is when the system is reset, or turned off.

### Class Diagram

### Sequence Diagram

This diagram represents a general flow of the WINLIFT software. Note that the function moveWindow() could be a openWindow() or a closeWindow() function, and while the system is running, the most of the time, it will wait for an input, until is turn off. The initial or default state is window open.



## Functional Decomposition

Overview of functions and their dependencies shown by a Static Function Tree  
  
  


**Function Description and Dynamic Behavior**

Provide detailed static and dynamic description of all functions of the SW Component.   
Functions which are defined in other SW Components shall only be referenced in the external interface description!  
The signature description shall be done inside the function header in the source code.  
  
For each function, the following section should be copied

## *Function void closeWindow( )*

|  |  |
| --- | --- |
| **Description** | Lifts the window |
| **Return Value** | *There is no return value* |
| **Precondition** | Only can be called when the when up button is pressed |
| **Post condition** | *Leds’ transition down-to-up executes* |
| **Error Conditions** |  |

This function will simulate and opening of the window by turning on, in an down-to-up transition of 400 ms, the leds. This function will call the function blueLed\_open() once to turn on the blue led, and when the window is totally open, the blueLed\_open() function must be call a final time to turn off the blue led.

## *Function void openWindow( )*

|  |  |
| --- | --- |
| **Description** | Lowers the window |
| **Return Value** | *There is no return value* |
| **Precondition** | Only can be called when the down button is pressed |
| **Post condition** | *Leds’ transition up-to-down executes* |
| **Error Conditions** |  |

This function will simulate and opening of the window by turning on, in an up-to-down transition of 400 ms, the leds. This function will call the function greenLed\_open() once to turn on the green led, and when the window is totally open, the greenLed\_open() function must be call a final time to turn off the blue led.

## *Function* void blueLed\_close( )

|  |  |
| --- | --- |
| **Description** | Turns on the blue Led |
| **Return Value** | *There is no return value* |
| **Precondition** | Antipinch functionality is disabled |
| **Post condition** | *Turns on/off Led* |
| **Error Conditions** |  |

## *Function* void greenLed\_open( )

|  |  |
| --- | --- |
| **Description** | Turns on the green Led |
| **Return Value** | *There is no return value* |
| **Precondition** | Antipinch functionality is disabled |
| **Post condition** | *Turns on/off green Led* |
| **Error Conditions** |  |

## *Function void antipinch()*

|  |  |
| --- | --- |
| **Description** | Stops and lowers the window |
| **Return Value** | *There is no return value* |
| **Precondition** | Only up button was pressed |
| **Post condition** | *Block inputs during 5 seconds* |
| **Error Conditions** |  |

This function will call the openWindow() task, and disable all the inputs during 5 seconds.

## Function T\_UWORD countPressTime( )

|  |  |
| --- | --- |
| **Description** | It counts the time of a button press |
| **Parameter 1** <input| output| inout> | *No parameters* |
| **Return Value** | *The elapsed time since a button was pressed until it is released* |
| **Precondition** | Up, down or antipinch button must have been pressed |
| **Post condition** | Call up(), down, or antipinch functions |
| **Error Conditions** |  |

This function will count the time of a button press, the register of the button will be read and then using the timer module it’ll start to count the time until the button is released.