|  |  |
| --- | --- |
| **Title:** | **WINLIFT**  **SW Component < 1.0 >** |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **History** | | | | |
| **Issue status**  (Index) | **Maturity/Date**  (draft/invalid/valid)  (dd-mmm-yyyy) | **Author**  Department | **Check/Release**  Department | **Description** |
| 7 | Draft  2-Nov-15 | Oscar Miranda  B.S. | Oscar Miranda  B.S. | References of requirements 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 |
|  |  |  |
|  |  |  |
|  |  |  |

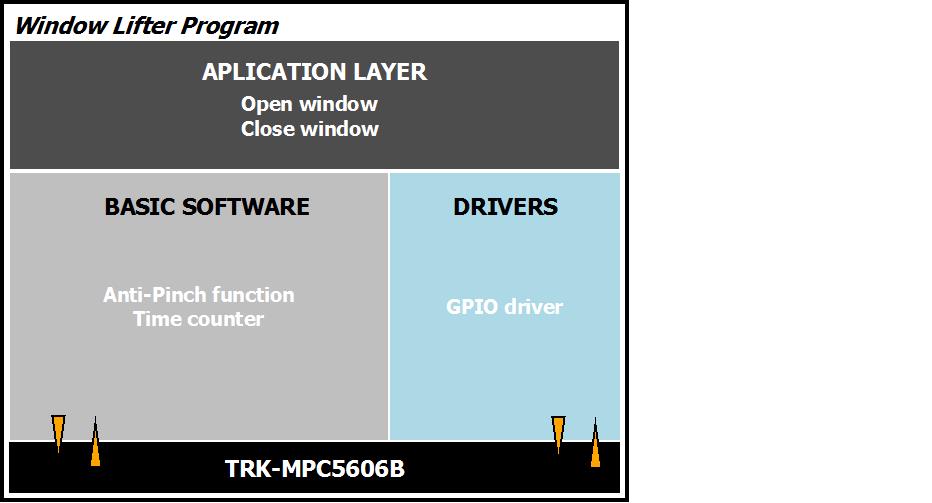
# Realization constraints and targets

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. **Req. 2.1**
* It will have a function that control the closure of the window. **Req. 2.2**
* 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. **Req. 4.3**
* When anti-pinch is active, a sub-function will disable all inputs for 5 seconds. **Req. 4.6**
* When opening or closing the window there will be an indicator LED indicating the process in progress. **Req. 2.8**
* There will be a function that counts how much time a button have been pressed. **Req. 3 and Req. 3.1**

# 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.

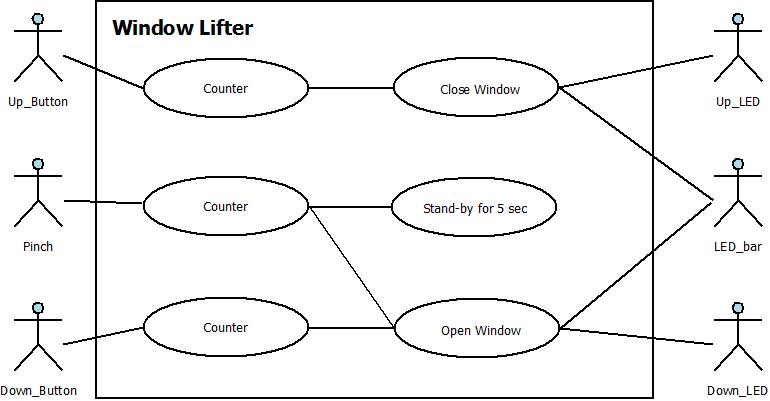


# SW Component internal breakdown

## Diagrams

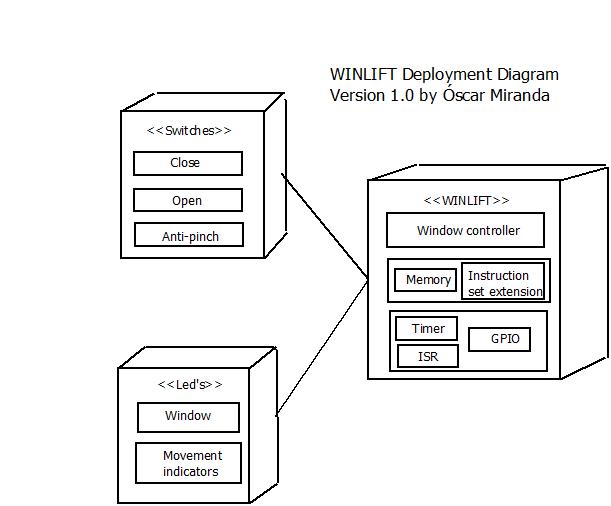
### Use Case Diagram

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.



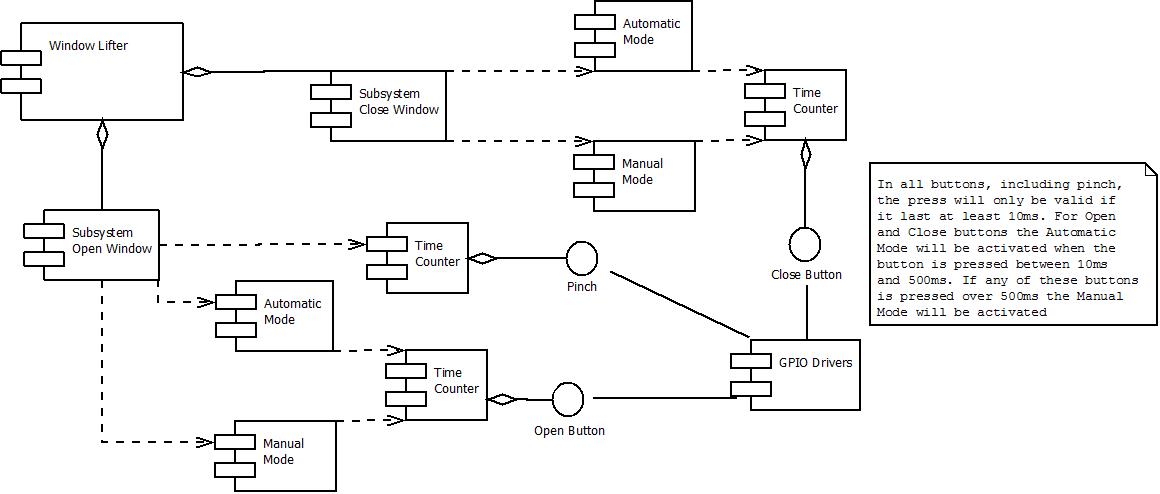
### Deployment Diagram

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



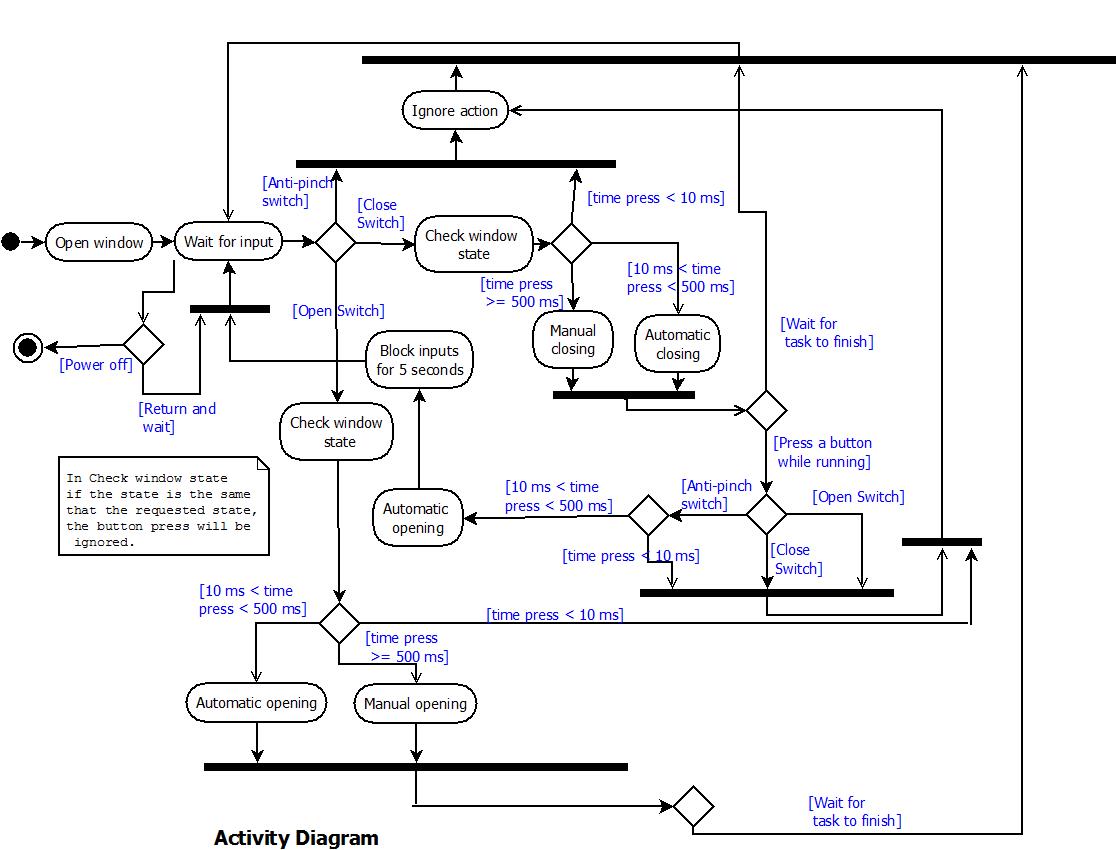
### Component Diagram

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

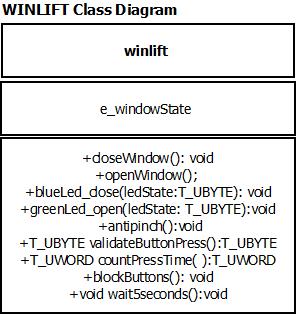


### Activity Diagram

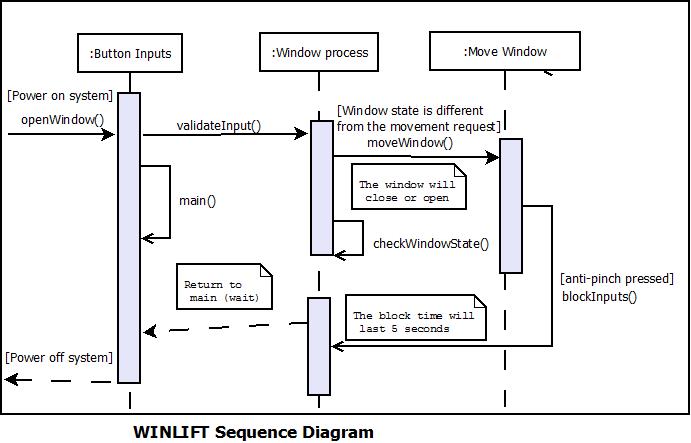
The following flowchart defines all the default and other possible states of the window, the operations like open or close window, the anti-pinch functionality, the validation of a button and the end of the program flow that is when the system is reset, or turned off.



### Class Diagram

The class represents the API of WINLIFT. It has 9 methods and 1 attribute.

### 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

## ***void closeWindow( )***

|  |  |
| --- | --- |
| **Description** | Lifts the window.  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. |
| **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** | *Not defined* |

***Req. 2.6***

## ***void openWindow( )***

|  |  |
| --- | --- |
| **Description** | Lowers the window.  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. |
| **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** | *Not defined* |

***Req. 2.6***

## void blueLed\_close(*T\_UBYTE ledState* )

|  |  |
| --- | --- |
| **Description** | Turns on/off the blue Led: if ledState == 0 turns on the led, if  ledState != 0 turns off the led |
| **Parameter 1** <input| output| inout> | *T\_UBYTE ledState* |
| **Precondition** | closeWindow() called |
| **Post condition** | *Turns on/off blueLed and the state of the window is defined* |
| **Error Conditions** | *Not defined* |

***Req. 2.8***

## void greenLed\_open(T\_UBYTE ledState )

|  |  |
| --- | --- |
| **Description** | Turns on/off the green Led: if ledState == 0 turns on the led, if  ledState != 0 turns off the led |
| **Parameter 1** <input| output| inout> | *T\_UBYTE ledState* |
| **Precondition** | openWindow() called |
| **Post condition** | *Turns on/off green Led and the state of the window is defined* |
| **Error Conditions** | *Not defined* |

***Req. 2.8***

## ***void antipinch()***

|  |  |
| --- | --- |
| **Description** | Stops and lowers the window. This function will call the openWindow() task, and disable all the inputs during 5 seconds. |
| **Return Value** | *There is no return value* |
| **Precondition** | Only open button was pressed |
| **Post condition** | *Block inputs during 5 seconds* |
| **Error Conditions** | *Not defined* |

***Req. 4.3, 4.4, 4.6***

## T\_UBYTE validateButtonPress( )

|  |  |
| --- | --- |
| **Description** | It validates the time press button (time press > 10 ms)  And choose a manual or automatic action |
| **Return Value** | *There is no return value* |
| **Precondition** | A button was pressed |
| **Post condition** | *Call antiPinch(), openWindow() or closeWindow()* |
| **Error Conditions** | *Not defined* |

***Req. 3.1***

## T\_ULONG countPressTime( )

|  |  |
| --- | --- |
| **Description** | It counts the time of a button press  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. |
| **Parameter 1** <input| output| inout> | *No parameters* |
| **Return Value** | *The elapsed time since a button was pressed until it is released* |
| **Precondition** | close, open or antipinch button must have been pressed |
| **Post condition** | *Timer is cleared* |
| **Error Conditions** | *Not defined* |
|  |  |

***Req. 4.6***

## ***void blockButtons()***

|  |  |
| --- | --- |
| **Description** | This function disable all the buttons for 5 seconds while the window is opening. When the window is totally open, all the inputs are enable. |
| **Return Value** | *There is no return value* |
| **Precondition** | A validation of anti-pinch was performed |
| **Post condition** | *Enable all the inputs* |
| **Error Conditions** | *Not defined* |

***Req. 4.6***

## ***void wait5seconds()***

|  |  |
| --- | --- |
| **Description** | A delay of 5 seconds |
| **Return Value** | *There is no return value* |
| **Precondition** | BlockButtons was invoked. |
| **Post condition** | *the system is waiting for another input* |
| **Error Conditions** | *Not defined* |

***Req. 4.6***