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

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **History** | | | | |
| **Issue status**  (Index) | **Maturity/Date**  (draft/invalid/valid)  (dd-mmm-yyyy) | **Author**  Department | **Check/Release**  Department | **Description** |
| 10 | valid  2-Nov-15 | Óscar Francisco  B.S. | Óscar Francisco  B.S. | Description of the deployment, activity and class diagram modified and expanded. Class diagram was modified because of the name of the class was wrong and the elements of the e\_wState were missed. |

**Table of Contents**

[1 Purpose 3](#_Toc434224514)

[2 Definitions and abbreviations 3](#_Toc434224515)

[3 Realization constraints and targets 3](#_Toc434224516)

[4 SW Conceptual design 4](#_Toc434224517)

[5 SW Component internal breakdown 5](#_Toc434224518)

[5.1 Diagrams 5](#_Toc434224519)

[5.1.1 Use Case Diagram 5](#_Toc434224520)

[5.1.2 Deployment Diagram 6](#_Toc434224521)

[5.1.3 Component Diagram 7](#_Toc434224522)

[5.1.4 Activity Diagram 8](#_Toc434224523)

[5.1.5 Class Diagram 9](#_Toc434224524)

[5.1.6 Sequence Diagram 9](#_Toc434224525)

[5.2 Functional Decomposition 10](#_Toc434224526)

[*5.3* *void WL\_WinMUp(void)* 10](#_Toc434224527)

[*5.4* *void WL\_WinMDw(void)* 10](#_Toc434224528)

[5.5 *void WL\_WinAUp(void)* 10](#_Toc434224529)

[*5.6* *void WL\_WinADw(void)* 11](#_Toc434224530)

[5.7 *void WL\_A\_Pinch (void)* 11](#_Toc434224531)

[5.8 *void WL\_CheckValid()* 11](#_Toc434224532)

[*5.9* *void blockButtons()* 11](#_Toc434224533)

[*5.10* *void wait5seconds()* 12](#_Toc434224534)

[*5.11* *int WL\_CheckAutoManualUp(void)* 12](#_Toc434224535)

[*5.12* *int WL\_CheckAutoManualDw(void)* 12](#_Toc434224536)

# 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 |
| 2 | Test\_Plan | 2 |
|  |  |  |
|  |  |  |

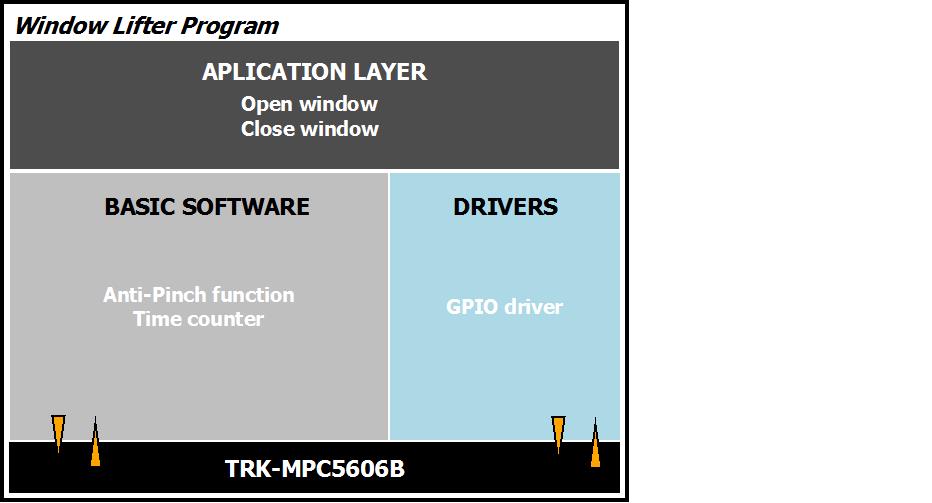
# Realization constraints and targets

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 and validate the press. **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.

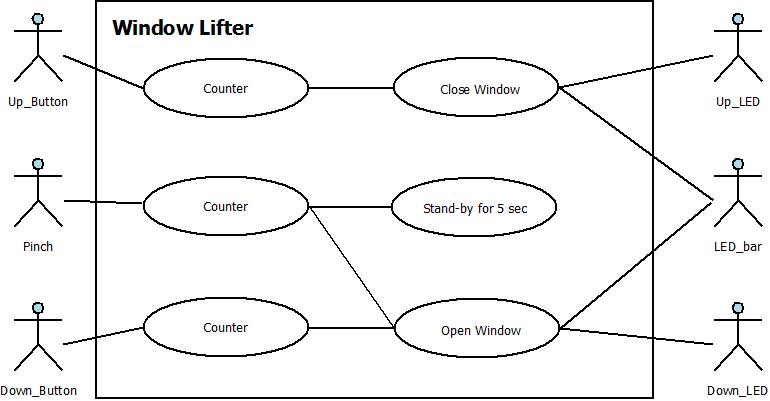


# SW Component internal breakdown

## Diagrams

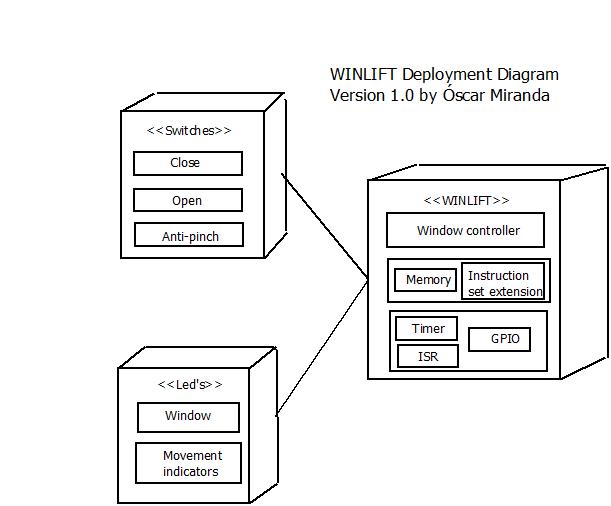
### Use Case Diagram

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.



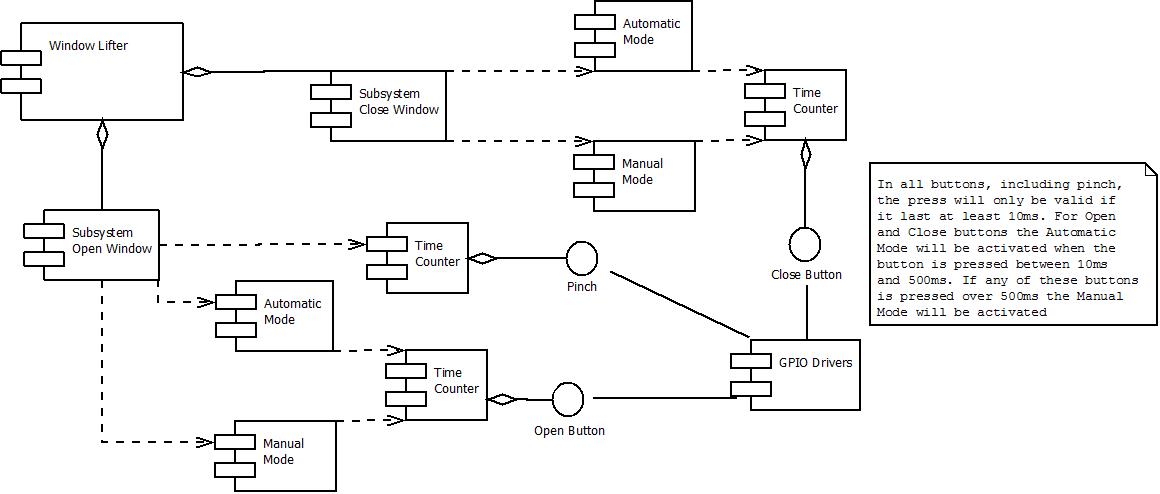
### Deployment Diagram

The following deployment diagram represents the inputs that are divided in three: Close, Open and anti-pinch; the outputs that are the LED simulation of the window, and the indicator LEDS, that display whether the window is lifting or lowering; the processing module that corresponds to the API, the HAL and the MCAL, in these modules the timings, the inputs and ouputs of the microcontroller and the interrupts are defined.



### Component Diagram

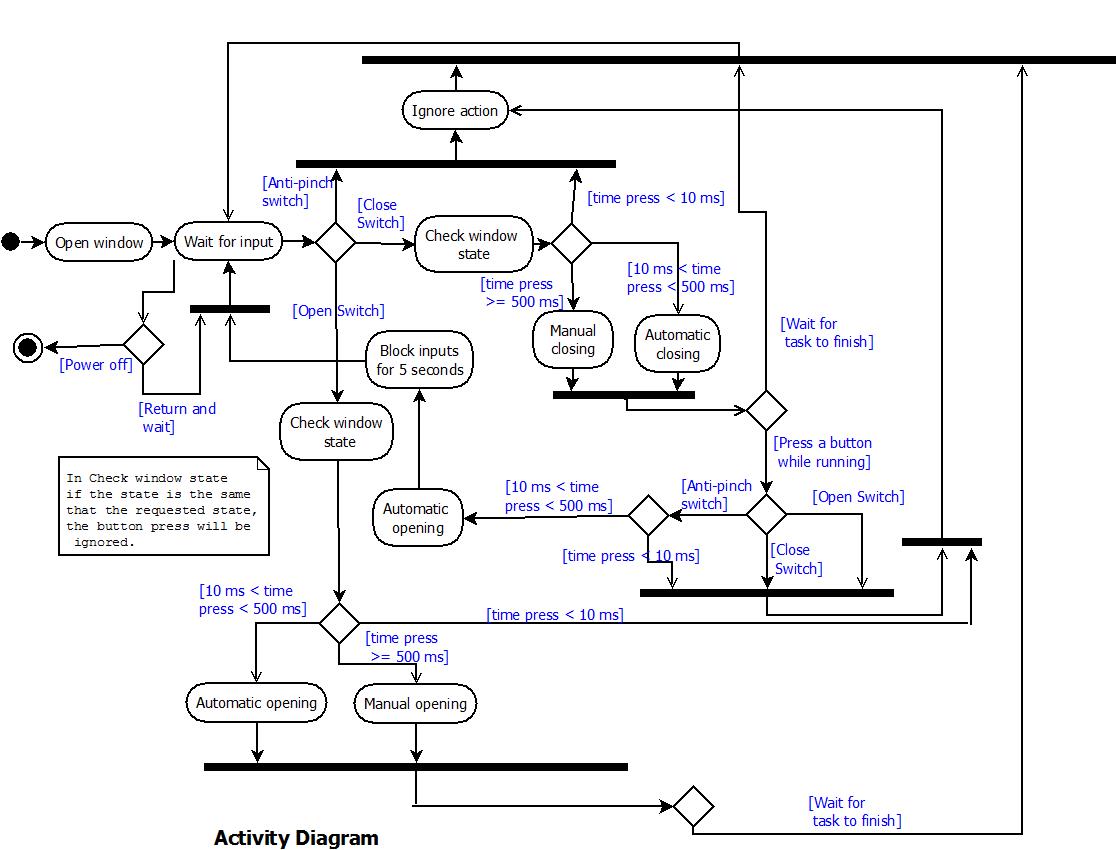
The following Component Diagram describes the structure and relations between the sub-systems comprehended in the Window Lifter system.



### Activity Diagram

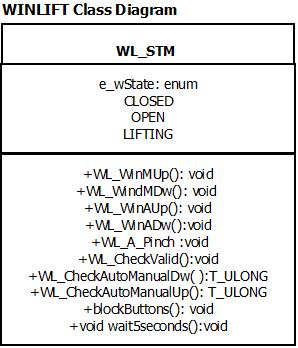
The following flowchart defines all the default and other possible states of the window and requests of the processor, the operations like open or close window, the anti-pinch functionality, the validation of a button and the end of the program flow that is when the system is reset, or turned off.

Roughly, the process of the program begins with a default state, that is open, then the system have to wait for an input. If a button is pressed, first of all, the current state of the window must be checked to avoid errors, for example, only the anti pinch functionality have to be activated if the state of the window is Lifting. Depending on the button press time, the program will evaluate whether to select a manual closing, an automatic closing, or goes to wait. If the anti pinch button is pressed, all the inputs must be disable for 5 seconds and then goes wait. To exit of wait state, there are two options, either to turn off the system or enter an input.

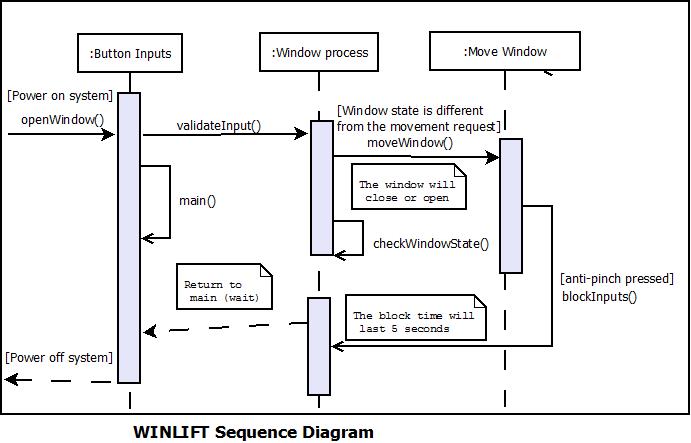


### Class Diagram

The class represents the API of WINLIFT. It has 10 methods and 1 attribute called e\_wState that is an enum and refers to the state of the window and also has three values: CLOSE, OPEN and LIFTING. For more details of the functions go to section 5.2.



### 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 WL\_WinMUp(void)***

|  |  |
| --- | --- |
| **Description** | Lifts the window.  This function will simulate the manual closure of the window by turning on, in a down-to-up transition of 400 ms, the LED bar. The function also turn on (when active) a blue LED indicator. |
| **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* |
| **Requirement** | Req 2.6, 3.3 |

## ***void WL\_WinMDw(void)***

|  |  |
| --- | --- |
| **Description** | Lowers the window.  This function will simulate the manual opening of the window by turning on, in a up-to-down transition of 400 ms, the LED bar. The function also turn on (when active) a blue LED indicator. |
| **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* |
| **Requirement** | Req 2.6, 3.3 |

## ***void WL\_WinAUp(void)***

|  |  |
| --- | --- |
| **Description** | Lifts the window.  This function will simulate the automatic closure of the window by turning on, in a down-to-up transition of 400 ms, the LED bar. The function also turn on (when active) a blue LED indicator. |
| **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* |
| **Requirement** | Req 2.6, 3.3 |

## ***void WL\_WinADw(void)***

|  |  |
| --- | --- |
| **Description** | Lowers the window.  This function will simulate the automatic opening of the window by turning on, in a up-to-down transition of 400 ms, the LED bar. The function also turn on (when active) a blue LED indicator. |
| **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* |
| **Requirement** | Req 2.6, 3.3 |

## ***void WL\_A\_Pinch (void)***

|  |  |
| --- | --- |
| **Description** | Stops and lowers the window. This function will be activated by a external interruption. After executed will disable all the inputs for 5 seconds. |
| **Return Value** | *There is no return value* |
| **Precondition** | Anti pinch simulation active. Window close or closing. |
| **Post condition** | *Block all inputs during 5 seconds* |
| **Error Conditions** | *Not defined* |
| **Requirement** | Req 4.3, 4.4, 4.6 |

## void WL\_CheckValid()

|  |  |
| --- | --- |
| **Description** | It counts the time of a button press to determine if it was a valid 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. |
| **Return Value** | *If it’s a valid button press* |
| **Precondition** | close, open or anti-pinch button must have been pressed |
| **Post condition** | *Timer is cleared* |
| **Error Conditions** | *Not defined* |
| **Requirement** | Req 3.1 |

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

|  |  |
| --- | --- |
| **Description** | This function disable all the buttons and call the wait5seconds function. After the window is totally open, then all the inputs are re-enabled. |
| **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* |
| **Requirement** | Req 4.6 |

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

|  |  |
| --- | --- |
| **Description** | This function count 5 seconds. |
| **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* |
| **Requirement** | Req 4.6 |

## ***T\_ULONG WL\_CheckAutoManualUp(void)***

|  |  |
| --- | --- |
| **Description** | Counts the time that a button has been pressed to determine if it will be an automatic or manual rutine for the closure. |
| **Return Value** | *Automatic or manual confirmation* |
| **Precondition** | Close button pressed |
| **Post condition** | *the system is waiting for another input* |
| **Error Conditions** | *Not defined* |
| **Requirement** | Req 3.3 |

## ***T\_ULONG WL\_CheckAutoManualDw(void)***

|  |  |
| --- | --- |
| **Description** | Counts the time that a button has been pressed to determine if it will be an automatic or manual routine for the opening. |
| **Return Value** | *Automatic or manual confirmation* |
| **Precondition** | Open button pressed |
| **Post condition** | *the system is waiting for another input* |
| **Error Conditions** | *Not defined* |
| **Requirement** | Req 3.3 |