|  |  |
| --- | --- |
| **Title:** | **Window Lifter**  **System Design** |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **History** | | | | |
| **Issue status**  (Index) | **Maturity/Date**  (draft/invalid/valid)  (dd-mmm-yyyy) | **Author**  Department | **Check/Release**  Department | **Description** |
| 1.0 | Draft  04-0ct-09 | Miguel Garcia | Miguel Garcia | Creation of the document |
| 2.0 | 21-Jul-15 | Misael AD | Diego Flores  David Rosales | Window lifter system design document |

**Table of Contents**

[1 Purpose 4](#_Toc425848552)

[2 Definitions and abbreviations 4](#_Toc425848553)

[3 Realization constraints and targets 5](#_Toc425848554)

[4 SW Conceptual design 6](#_Toc425848555)

[5 SW Component internal breakdown 7](#_Toc425848556)

[5.1 Functional Decomposition 8](#_Toc425848557)

[*5.2* Function *void main(void)* 9](#_Toc425848558)

[*5.3* Function void init\_ModesAndClock(void) 9](#_Toc425848559)

[*5.4* Function *void disableWatchdog(* 10](#_Toc425848560)

[*5.5* Function *void init\_OnBoardPushButtons(void)* 10](#_Toc425848561)

[*5.6* Function *void init\_LEDBar(void);* 10](#_Toc425848562)

[*5.7* Function *void SchM\_Init(&SchConfig);* 10](#_Toc425848563)

[*5.8* Function *void SchM\_Start(void)* 11](#_Toc425848564)

[6 Assumptions 11](#_Toc425848565)

# Purpose

Window Lifter system design document. This document shows the system decomposition to explain its functional parts. The purpose is to develop and implement an Electronic Window Control Module (EWCM) according to the AEP requirements defined in the window lifter requirements document

# Definitions and abbreviations

**Definitions**

|  |  |
| --- | --- |
|  |  |
|  |  |
|  |  |
|  |  |

**Abbreviations**

EWCM Electronic Window Control Module

GPI General Purpose Inputs

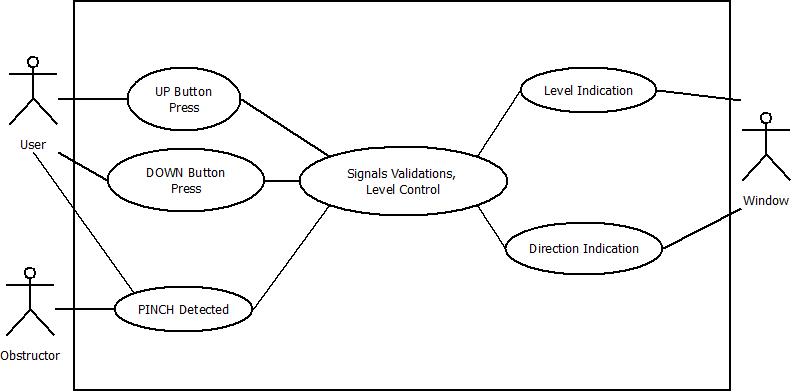
GPO General Purpose Outputs

SchM Scheduler module

**References**

|  |  |  |
| --- | --- | --- |
| **N°** | **Document name** |  |
| 1 | [Window lifter requirements](../Window%20lifter%20requirements.docx) |  |
| *2* | [*DSD State Machines*](DSD_STATE-MACHINES.docx) |  |
| 3 | [*Traceability matrix*](../1.0%20Requirements/Traceability%20Matrix%20Template.xlsm) |  |
| 4 | [Review document](../EWCM%20Review%20Document.docx) |  |
|  |  |  |

# Realization constraints and targets



Electronic Window Control Module

* **User**: Person that interacts with the window lifter inputs (push buttons)
* **Obstructer**: Object or person that interrupts the closing movement of the window
* **Window**: System actuator, for this case a 10-LED bar will be used. Two possible actions; UP movement and DOWN movement.

**Use cases**

* UP Button Press:

UP command input is activated (UP push button is pressed)

* DOWN Button Press:

DOWN command input is activated (DOWN push button is pressed)

* Signals validation & level control

System validates every input, de-bounces the signals, activates an operation mode according to the time, and modifies the window position.

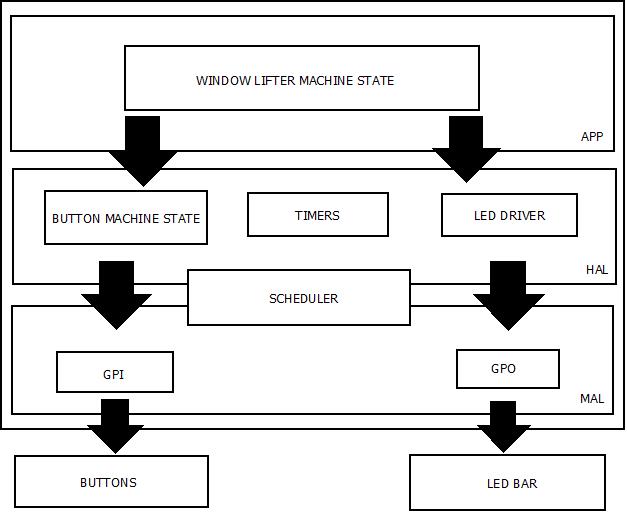
* Level indication:

Show the current window position or level; in this case a LED bar.

* Direction indication:

Show the movement’s direction UP or DOWN; using two LEDs.

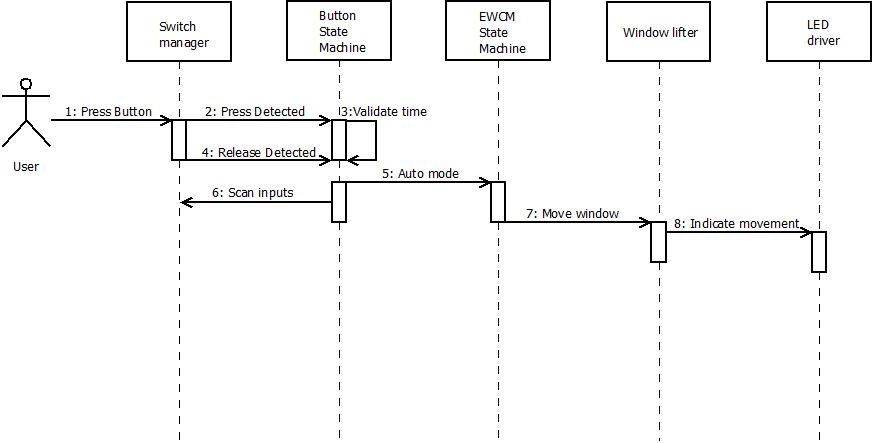
# SW Conceptual design

To get the system working it is required to use the GPIO in the MAL, because it will run on a scheduler the timers are implicit among the scheduler, the timer driver through the PIT sets the scheduler OS tick. The window lifter application gets the input values through the buttons machine state and controls the LEDs depending on the required state or function.

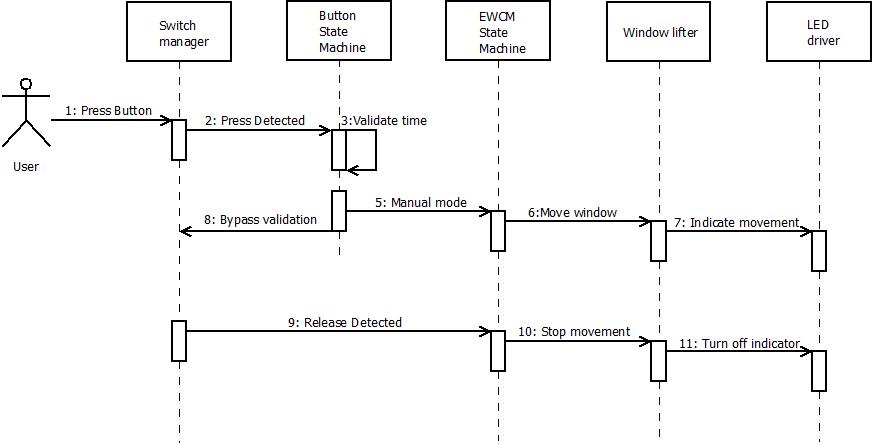
# SW Component internal breakdown

The system is composed of two state machines, one for the buttons and the other for the application functionality. System functionality defined by the three possible valid inputs (Auto mode or one-touch, manual mode and anti-pinch)

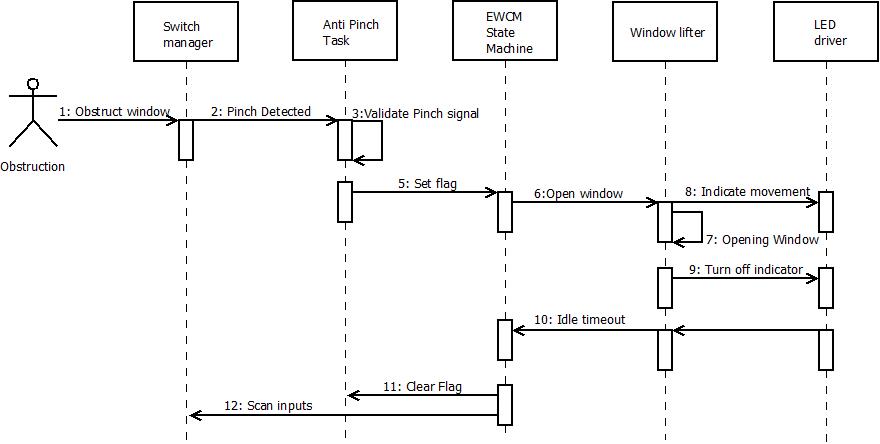
Automatic mode



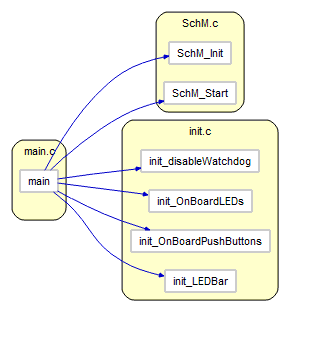
Manual mode



Anti-pinch mode



## Functional Decomposition

Microcontroller is initialized before handing over task control to the scheduler.  
  
  


## Function void main(void)

|  |  |
| --- | --- |
| **Description** | Initializes the microcontroller and gives control to the scheduler. |
| **Parameter** | *None* |
| **Return Value** | *None* |
| **Precondition** | *None* |
| **Post condition** | *Gives control to the scheduler.* |
| **Error Conditions** |  |

Main(void)

initModesAndClock();

disableWatchdog();

Init\_OnBoardPushButtons();

Init\_LEDBar();

SchM\_Init(&SchConfig);

SchM\_Start();

## Function void init\_ModesAndClock(void)

|  |  |
| --- | --- |
| **Description** | Initialize system clock mode. |
| **Parameter** | *None* |
| **Return Value** | *None* |
| **Precondition** |  |
| **Post condition** | *System Clock and working mode set* |
| **Error Conditions** |  |

## Function void disableWatchdog(

|  |  |
| --- | --- |
| **Description** | Disable the microcontroller watch dog timer |
| **Parameter** | *None* |
| **Precondition** | *None* |
| **Post condition** | *Watchdog timer turned off* |
| **Error Conditions** |  |

## Function void init\_OnBoardPushButtons(void)

|  |  |
| --- | --- |
| **Description** | Initialize Bolero’s on board push buttons |
| **Parameter** | *None* |
| **Precondition** | *None* |
| **Post condition** | *On board push buttons connected pins set as inputs* |
| **Error Conditions** |  |

## Function void init\_LEDBar(void);

|  |  |
| --- | --- |
| **Description** | Initialize pins to control external LED bar |
| **Parameter** | *None* |
| **Precondition** | *None* |
| **Post condition** | *Pins set as outputs to handle external LED bar* |
| **Error Conditions** |  |

## Function void SchM\_Init(&SchConfig);

|  |  |
| --- | --- |
| **Description** | Initializes scheduler with the tasks to execute |
| **Parameter** | *SchConfig table; contains every task detailed information* |
| **Precondition** | *None* |
| **Post condition** | *Scheduler initialized* |
| **Error Conditions** |  |

## Function void SchM\_Start(void)

|  |  |
| --- | --- |
| **Description** | Starts OS timer, enables interrupts |
| **Parameter** | *None* |
| **Precondition** | *None* |
| **Post condition** | *Scheduler takes control of the micro controller* |
| **Error Conditions** |  |

# Assumptions

1. System working in automatic mode (one-touch functionality: going UP or going DOWN) can be interrupted when a valid button press is detected and change direction. Functionality always obeys user command.
2. Window movement is always ruled by the 400ms time period. This to avoid that continuous valid button presses would move the window faster than allowed, preventing possible damage to the system.