Window lifter with scheduler

Title: **SW Component < Window Lifter Sch 1.0 >**

History				
Issue status (Index)	**Maturity/Date** (draft/invalid/valid) (dd-mmm-yyyy)	**Author** Department	**Check/Release** Department	**Description**
1.0	Draft 16-nov-15	Francisco Quirarte	Francisco Quirarte y David Díaz	Creation of the document.

History				
Issue status (Index)	**Maturity/Date** (draft/invalid/valid) (dd-mmm-yyyy)	**Author** Department	**Check/Release** Department	**Description**
2.0	Draft 17-nov-15	David Díaz	Francisco Quirarte	Added: Purpose, Definitions and abbreviations.

History				
Issue status (Index)	**Maturity/Date** (draft/invalid/valid) (dd-mmm-yyyy)	**Author** Department	**Check/Release** Department	**Description**
3.0	Draft 18-Nov-15	Francisco Quirarte y David Díaz	Francisco Quirarte	Added: Realization constraints and targets.

History				
Issue status (Index)	**Maturity/Date** (draft/invalid/valid) (dd-mmm-yyyy)	**Author** Department	**Check/Release** Department	**Description**
4.0	Draft 19- Nov -15	Francisco Quirarte y David Díaz	Oswaldo Garcia	Added: SW Conceptual design.

History				
Issue status (Index)	**Maturity/Date** (draft/invalid/valid) (dd-mmm-yyyy)	**Author** Department	**Check/Release** Department	**Description**
5.0	Draft 19-Nov-15	Francisco Quirarte y David Díaz	Oswaldo Garcia	Added: SW Component internal breakdown. Added: General corrections.

History				
Issue status (Index)	**Maturity/Date** (draft/invalid/valid) (dd-mmm-yyyy)	**Author** Department	**Check/Release** Department	**Description**
6.0	Draft 22-nov-15	Francisco Quirarte y David Díaz	Isamar Gálvez	Review of SW Component internal breakdown and added general corrections.

History				
Issue status (Index)	**Maturity/Date** (draft/invalid/valid) (dd-mmm-yyyy)	**Author** Department	**Check/Release** Department	**Description**
7.0	Release 23-Nov-15	Francisco Quirarte y David Díaz	Oswaldo Garcia e Isamar Gálvez	Review of SW Conceptual design and added general corrections.

Table of Contents

1 Purpose

This document has been created to explain how the window lifter works. The desired Window Lifter has to control the movement of a car's window. It has too different movements (Up/Down) and an Anti-Pinch system, which allows controlling the way the windows behave when it finds an obstacle.

Window lifter is a module that controlees the movement of a window. The module works with two switches that indicate the direction of the movement.

The window will be emulated using a 10 led bar. The time between each transition shall be 400 msec. Each window movement has to be indicated trough a led color. Depending on movement each led has to be turn on: **UP-BLUE** and **DOWN-GREEN**.

In order to consider a validate button press; the button has to be pressed at least 10 msec. The module has to be able to detect fail button press. In case the button is pressed less than 10 msec or a button combination it will to be considered as invalid.

The module will have an antipinch function that will be emulated with a push button. This signal just can be considered as valid when the movement is UP. If this signal is valid then the module has to stop the UP Movement and then DOWN the window until the window get totally OPEN. After window is totally OPEN the module has to ignore during 5 seconds all button press. After this time the module has to recognize every button press.

All the system is implemented in tasks on a scheduler main program.

2 Definitions and abbreviations

Definitions

Acronym	Definition
msec	Mili second
Window lifter	It is a system that controls the movement (up and down) of the car window.
antipinch	It is a system that stop the movement of the window when it detects an object or a body part while the window is going up.
count	Counter
Sw	Switch
validpress	Validate the press of the button

Abbreviations

Acronym	Definition
STM	System Timer Module
GPIO	General Purpose Input Output
CRT	Conversion Timing registers
IPM	Input Period Measurement
CIR	Channel Interrupt Register
SIUL	System Integration Unit Lite
STM CMP	STM Compare Register

References

N°	Document name
1	*Traceability Matrix Template.xls*
2	MPC5604B/C Microcontroller Reference Manual.pdf
3	Test_template.xls
4	http://www.freescale.com/products/power-architecture-processors/mpc5xxx-5xxx-32-bit-mcus/mpc56xx-mcus/mpc5606b-startertrak-development-kit:TRK-MPC5606B
5	http://www.freescale.com/products/power-architecture-processors/mpc5xxx-5xxx-32-bit-mcus/mpc56xx-mcus/ultra-reliable-mpc56xb-mcu-for-automotive-industrial-general-purpose:MPC560xB#pspFeatures
6	SW_C_Code_Review_Template.docx
7	Window-lifter-requirements.docx

3 Realization constraints and targets

System must ensure reliability on window behavior and security with the anti-pinch function, push buttons functions shall be much delimited in any unusual situation from ambiguous scenarios.
The constrains to accomplish this targets are the use of microcontroller freescale TRK-MPC5606B, emulation just for 10 led bar and anti-pinch function managed only by push button with scheduler control.

The system will be running in the MPC 5606B. With the following specifications:

- MPC5606B MCU in a 144LQFP package.
- On-board JTAG connection via open source OSBDM circuit using the MPC9S08JM MCU
- MCZ3390S5EK system basis chip with advanced power management and integrated CAN transceiver.
- CAN and LIN interface.
- Analog interface with potentiometer.
- High-efficiency LEDs.
- SCI serial communication interface.

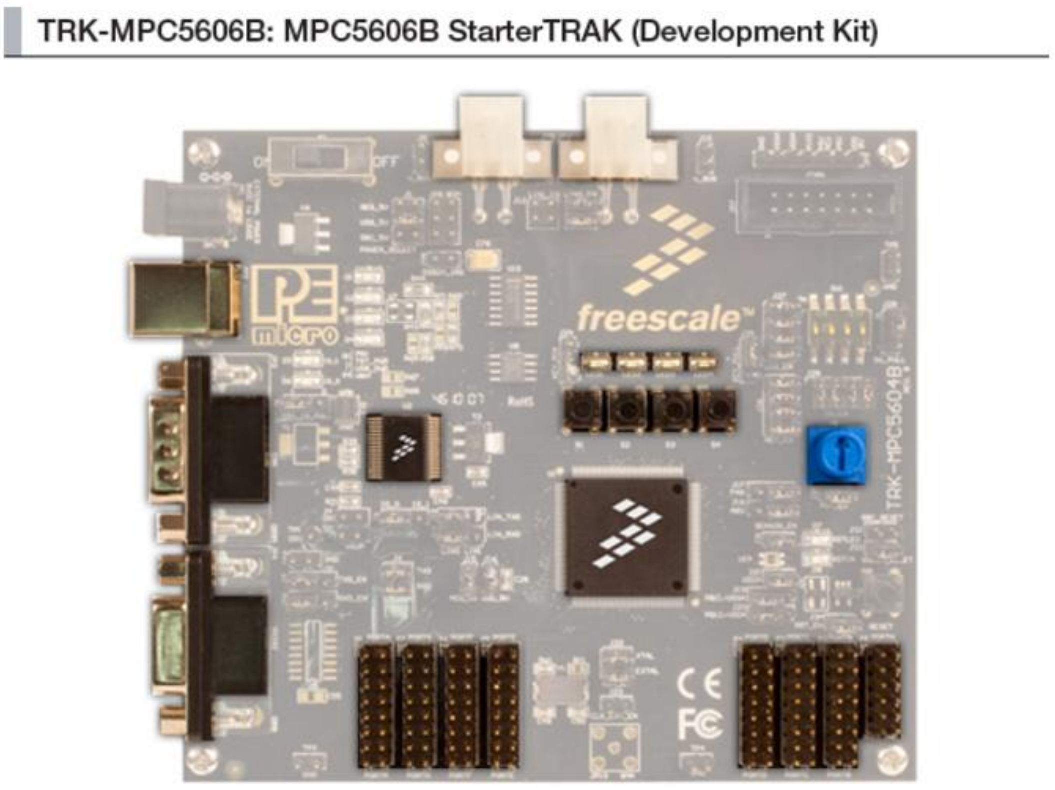

Figure 1: MPC5606B board.

- Operating Frequency (Max): 64 MHz.
- Total DMA Channels 16.
- Internal Flash (KB): 512
- GPIOs: 149.
- EEPROM: 64 KB DataFlash®
- RAM: Up to 96 KB
- Timer: 16 bits up to 64 channels
- ADC:
- 10 bits up to 36 channels
- 12 bits up to 16 channels
- Up to six CAN
- Up to six SPI
- Up to 10 LINFlex

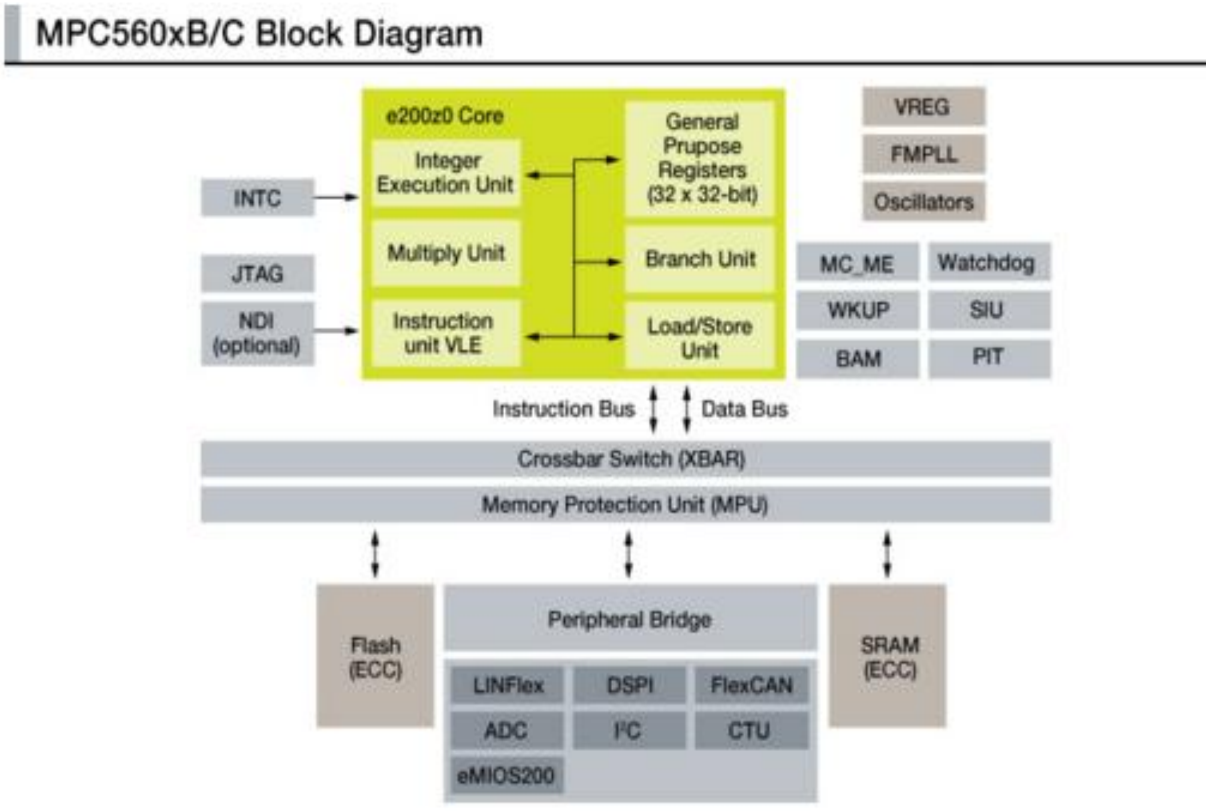

Figure 2: MPC56X B/C Boolero Arquitecture Family

4 SW Conceptual design

4.1 Deployment Diagram (A)

Are simple block diagrams, showing the physical configurations of software and hardware items. Often combined with component diagrams to illustrate which parts of the software run on which controller.

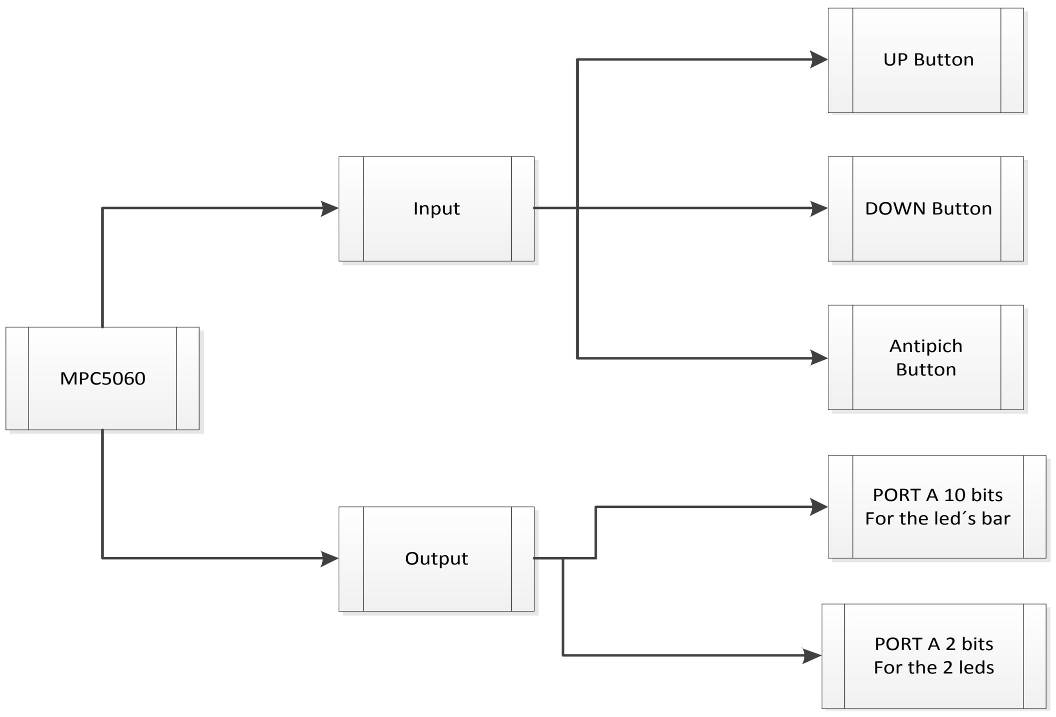
UP Button
Input
DOWN Button
MPC5060
Antipich Button
PORT A 10 bits For the led´s bar
Output
PORT A 2 bits For the 2 leds

Figure 3: Deployment Diagram. Represents the physical configurations of software and hardware items.

4.2 Use Case Diagram

Diagram (a)

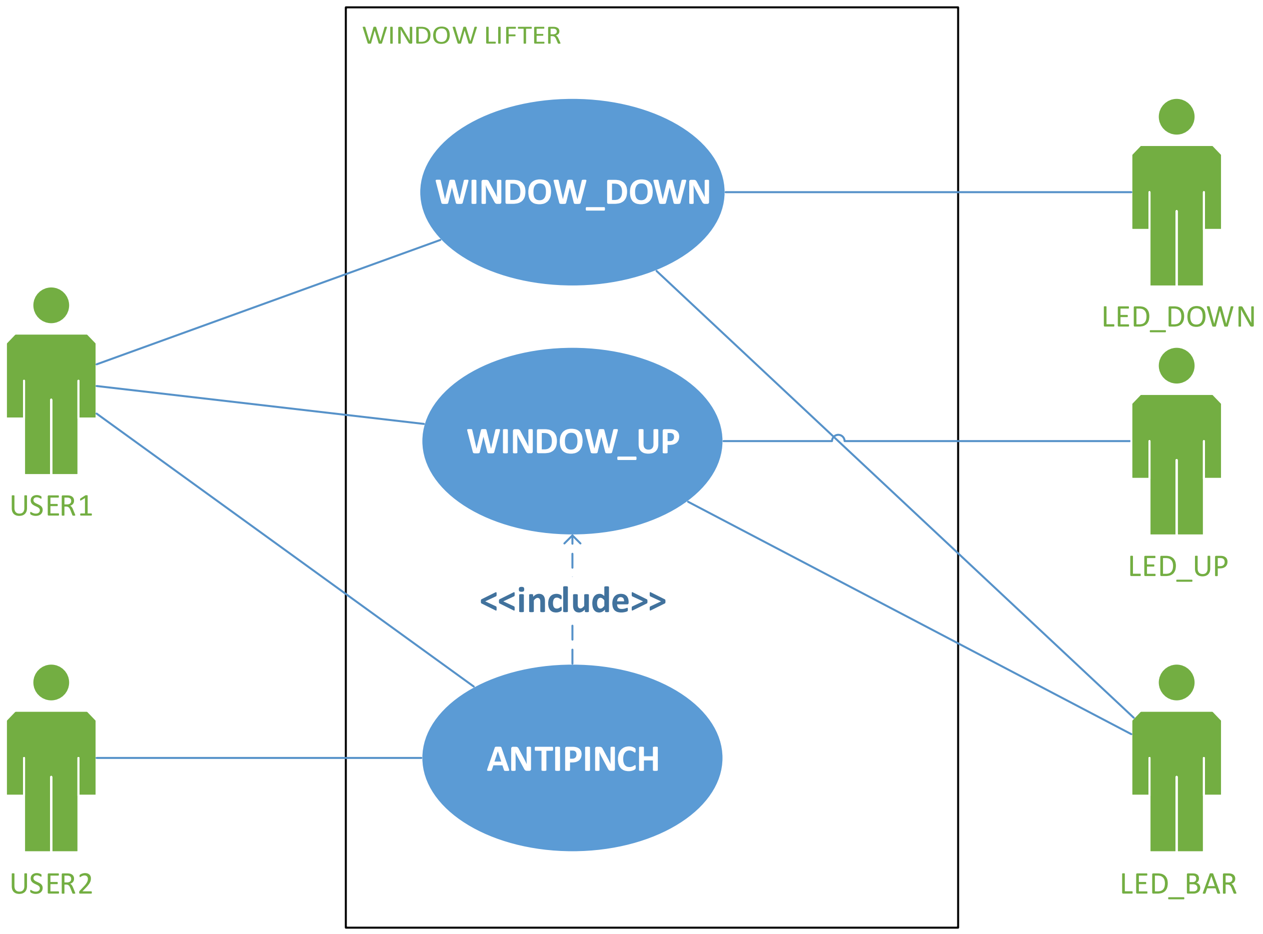

Figure 4: Use Case Diagram (a). We can see the buttons used to control the movement, the functions included in the system and the Window Lifter Application.

Diagram (b)

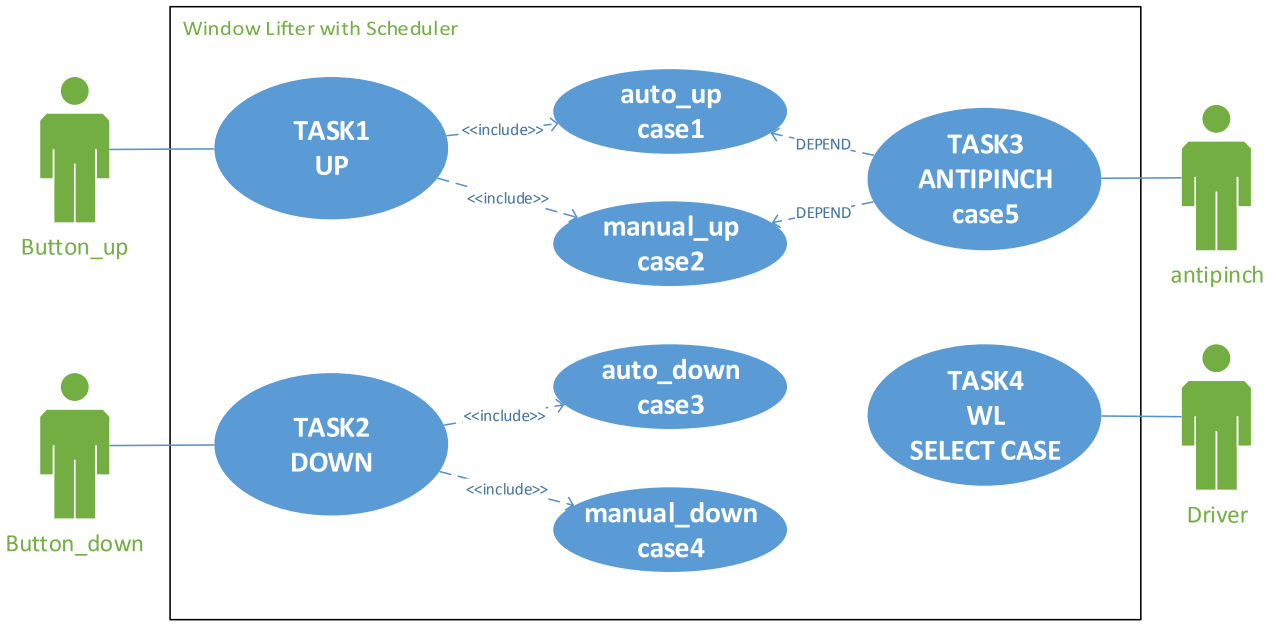

Figure 5: Use Case Diagram (b). In this diagram we can see another way, that represents the basic interaction of human and Window Lifter system.

4.3 Component Diagram (A/D)

Diagram (a)

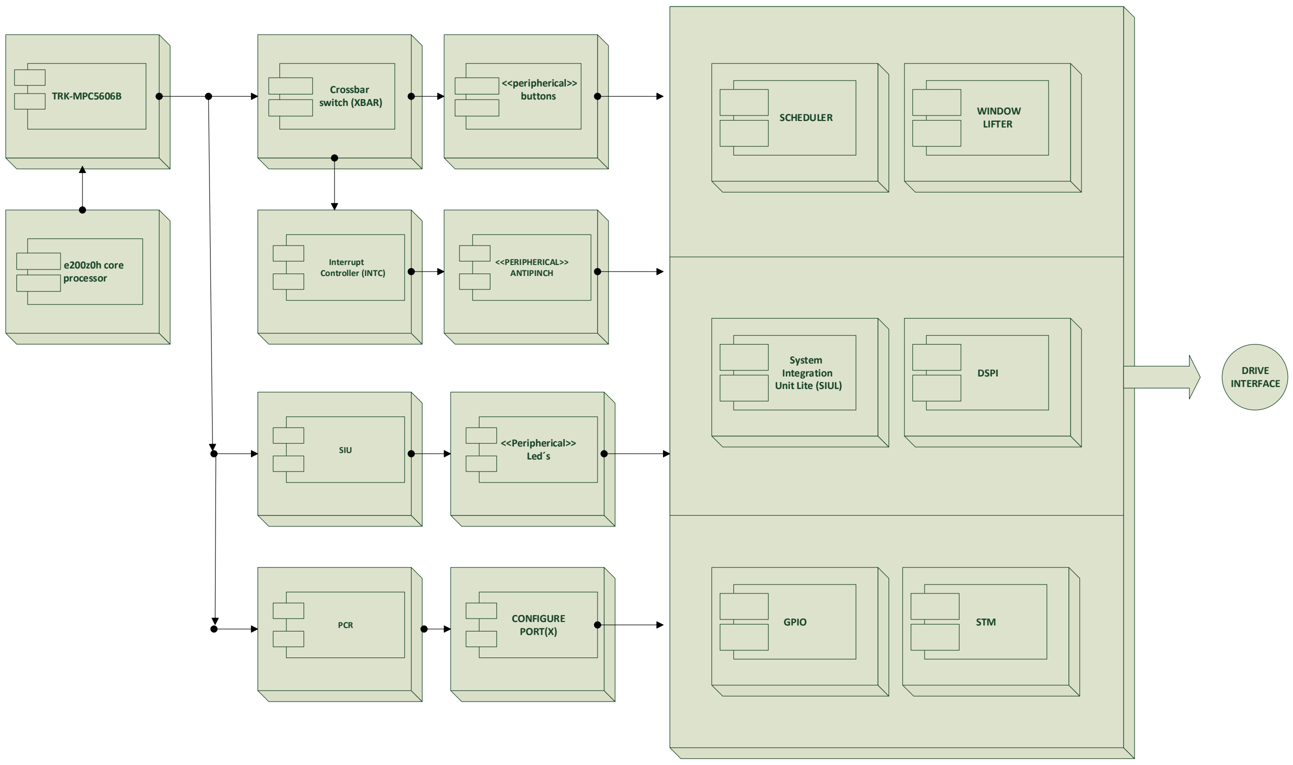

Figure 6: Component Diagram (a). Shows the structure of components, i.e. modular, deployable and replaceable parts of a system, subsystem or module that encapsulate implementation and expose a set of interfaces.

Diagram (b)

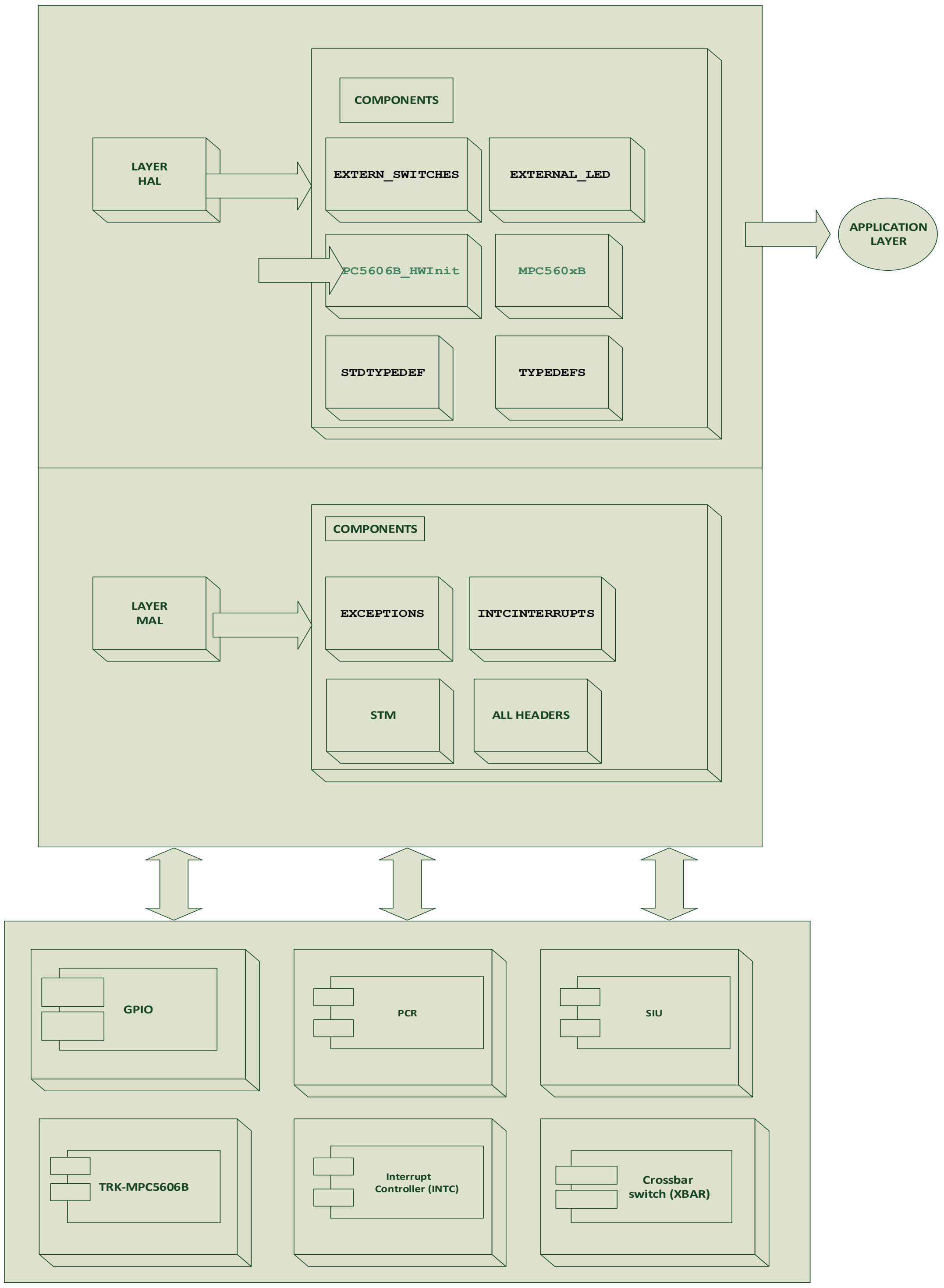

Figure 7: Component Diagram (b). Components can be organized in a hierarchical manner, i.e. a big subsystem contains a set of smaller subsystems. The smallest component in such a structure is equal to a module in the functional world.

4.4 Activity Diagram (R)

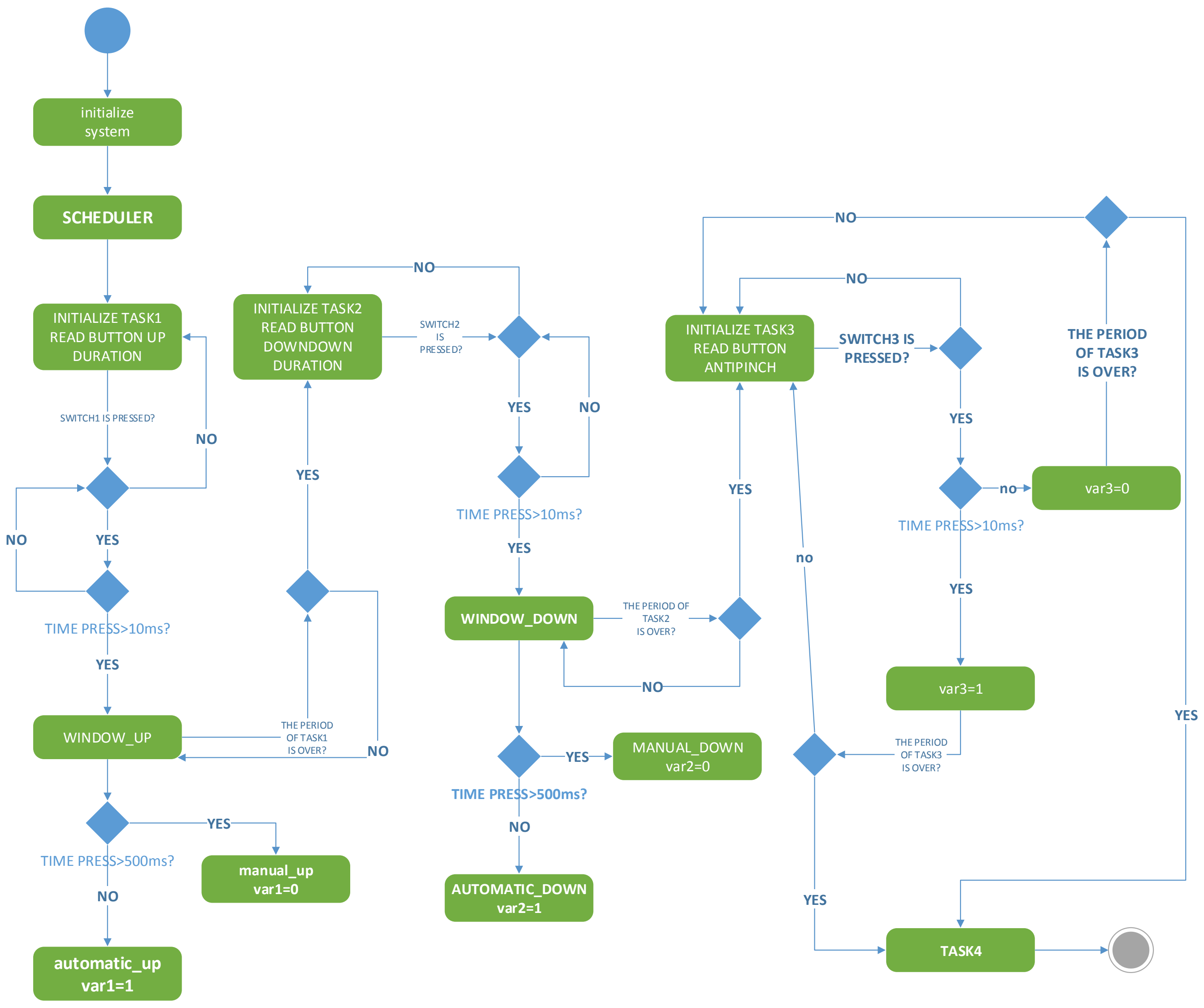

Figure 8: Activity Diagram. Describes the behavior of the Window Lifter program**.**

4.5 CLASS

Diagram (a)

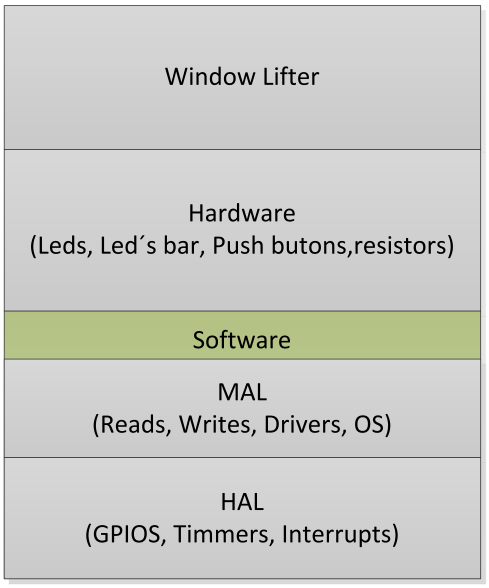
Window Lifter
Hardware
(Leds, Led´s bar, Push butons,resistors)
Software
MAL
(Reads, Writes, Drivers, OS)
HAL
(GPIOS, Timmers, Interrupts)

Figure 9: Represents the different classes of the software

Diagram (b)

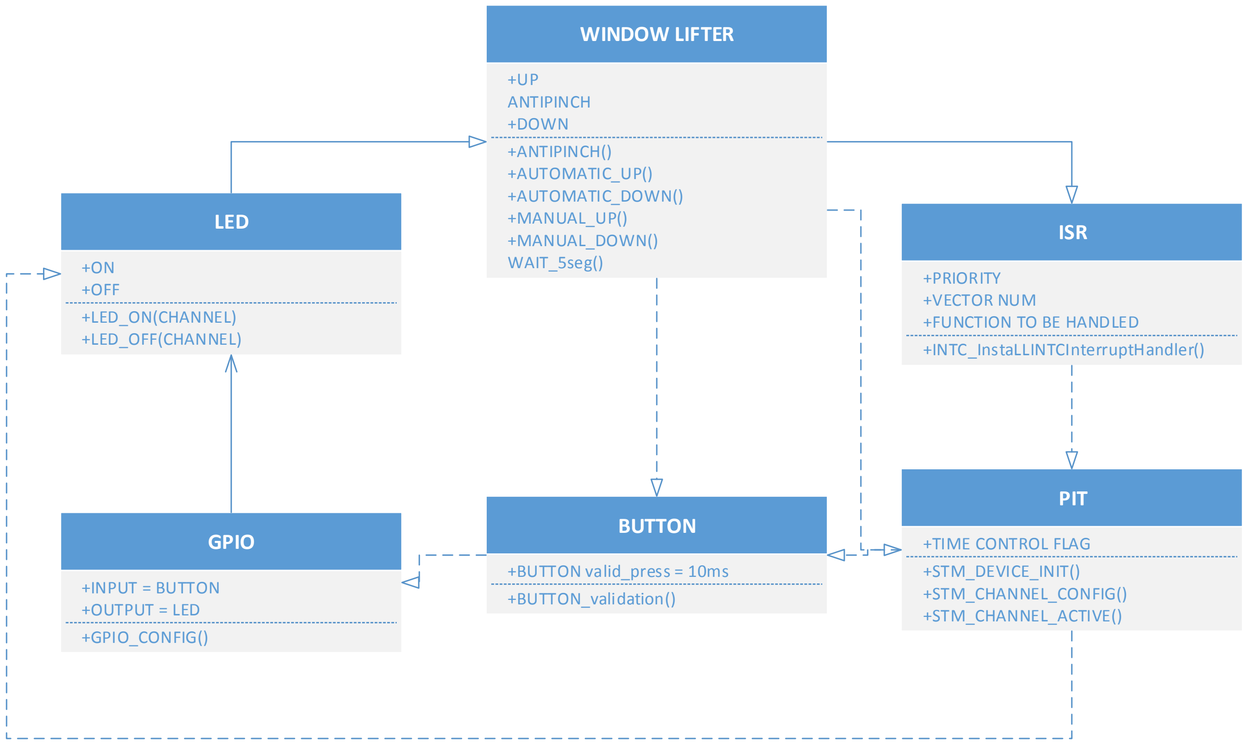

Figure 10: Represents the different classes of the software.

4.6 SEQUENCE

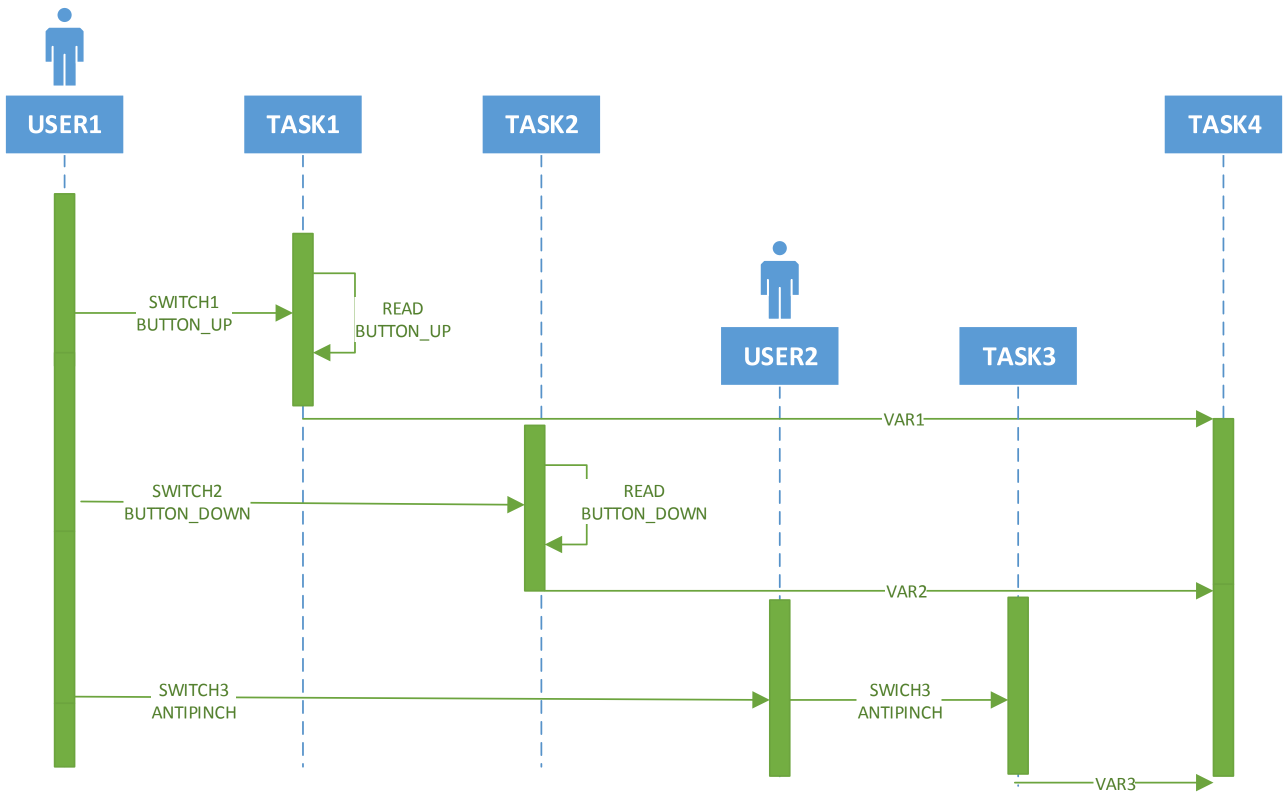

Figure 11: Sequence. Pattern of interaction among the objects, arranged in chronological order lifelines.

4.7 State machine diagram

Describes how the application shall works and the flow between states

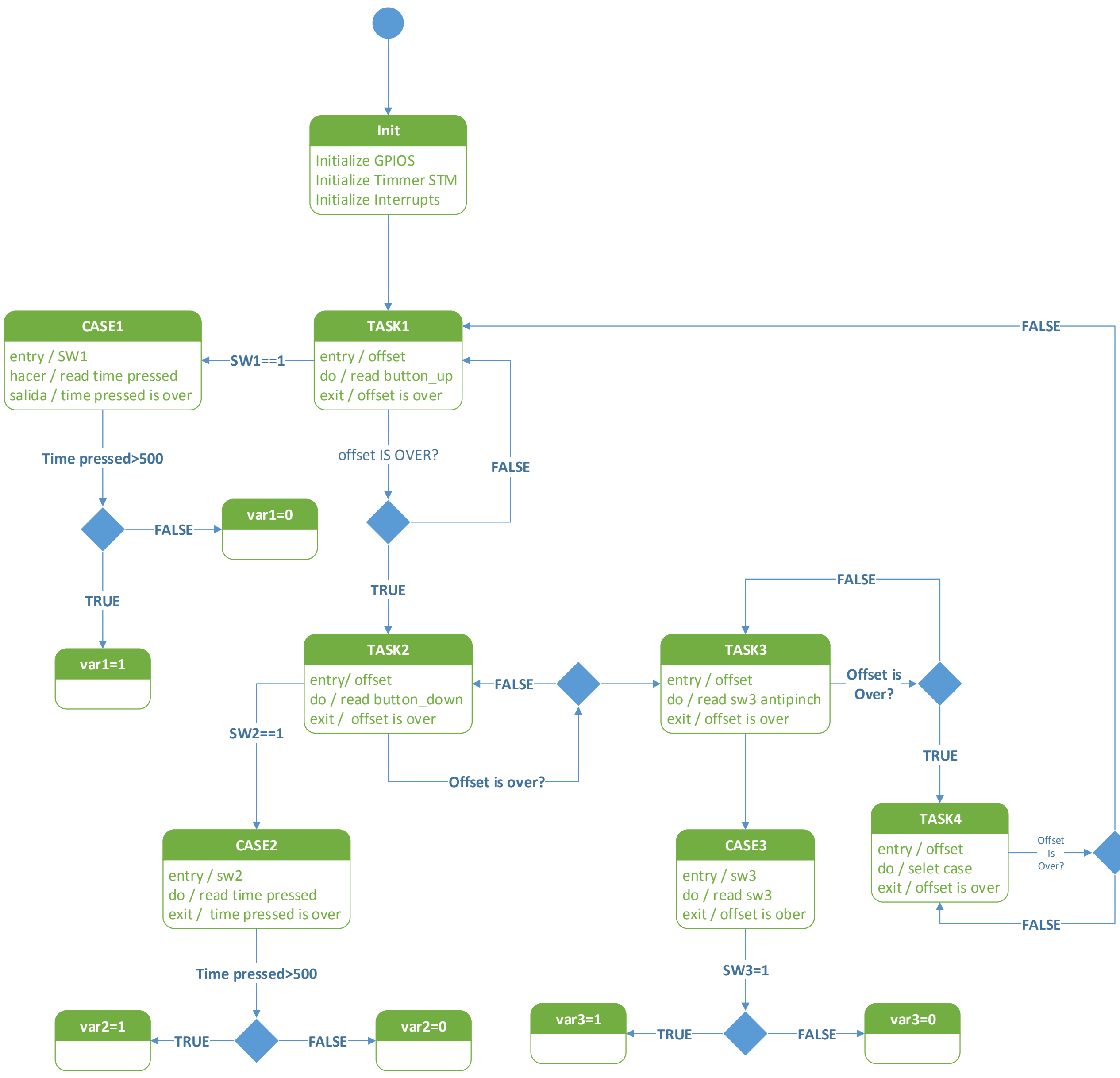

Figure 12: State machine diagram

5 SW Component internal breakdown

Based on the above concept of the design, SW components will be managed as the following subdivided files:

GPIO: Enable and set outputs for led and inputs for push buttons

Button: Capture button signal as up or down when a time condition is accomplished, as result is the initial state from the inputs

Window lifter: When an input is captured and the state is defined by the button file, this is evaluated with the state machine contained into the window lifter file, also here is where the states are stablished with the conditions based on customer requirements

- A header which contains all the software to perform the main system was used the name of this header is DRIVERS.h.
- I will use a header called config_timer.h which contains initialization of the timer STM of the microcontroller.
- I will use a header called Pb.h which contains initialization of PLL and reading the buttons and the frequency at which they will be working.
- I will use a header called Port_config.h which contains initialization of the Configure the portA and Switches.
- I will use a header called stdtypedef.h which contains Public type header file for the coreHAL.
- It was created three global variables which are: interrupt,validar_Pb and count_Pb.
- The variable interrupt habe the task of turn on the leds with steps of 400ms.
- The variable validar_Pb habe the task of enable the function “Pb_up_auto” this function enable the requirement of turn on the ten leds in the led bar simulating the sequence up in automatic.
- The variable count_Pb habe the task of increase a counter to know the position of the time flag in 10ms of stm timer.
- The function “initModesAndClock” habe the task of initialization yhe frecuence of the microcontroller.
- Inside of the function “Principal_function” we have the function “Pb_up_auto”, “Pb_Down”, “interruption” and “condición para pb_up_auto”.

5.1 Functional Decomposition

- **Read button Up duration**: this requirement have the task decide among which the function is activated function up one touch or the function function up manual.
- **Function Up one touch**: this function has the task of turn on the leds by steps of 400ms in automatic.
- **Function up manual**: this function has the task of turn on the led bar while the push button up is pressed for more of 500ms.
- **Antipinch signal**: this function turn off the led's on the led bar if the antipinch button is pressed the led bar simulating the led bar turn off one by one in steps of 400ms.
- **Read button down duration**: this requirement has the task to decide whether if the function is activated function read button down duration.
- **Read button down duration**: this requirement have the task decide among which the function is activated function down one touch or function down manual.
- **Button down one touch**: this function have the task of turn off the leds by steps of 400ms in automatic.
- **Function down manual**: this function has the task of turn off the led bar while the push button up is pressed for more of 500ms.

5.2 File name descriptions

File Name	Description
Tasks.c	Provides the timed task definitions
Tasks.h	Export the timed task interfaces to the scheduler configuration file
Exceptions.c	Setup of IVPR to point to the EXCEPTION HANDLERS memory area defined in the linker command file.
Exceptions.h	Export the Exception handlers to scheduler module
Timer.c	Configurate the STM timer initialization and configurate the STM flag in 1 ms
Timer.h	Export the STM flag and rise the tickflag each 1 millisecond
Leds.c	Configured the PORTA as outputs and led1 to led4 on board
Leds.h	Export the configuration of porta and led1 to led4 on board
Typedesfs.h	This file defines all of the data types for the Motorola header file
Stdtypedef.h	Public type header file for the coreHAL
INIT.c	Initialise PLL before turning it on
INIT.h	Export the pll configuration
IntcInterrupts.c	Contains an implementations of generic interrupt controller handling routines for the MPC56xx and PX MCU families
IntcInterrupts.h	Export the interface of interrupt controller handing
Kernel.h	Contain all headers implemented in the system

5.3 Api specification

Name:	**Taskmasktype**			
Range:	tickflag	Mask_1ms	OxFA00	Mask required for 1 ms task
	task1	Mask_1ms		Mask required for 1 ms task
	task2	Mask_3ms		Mask required for 3 ms task
	Task3	Mask_5ms		Mask required for 5 ms task
	task4	Mask_17ms		Mask required for 17 ms task
Description	The mask values to generate the task offset			

5.4 Schmodule_configType

Name:	Scheduler	
Type:	u8	
Range:	n.a.	Structure to hold the module's configuration set. The Contents of this data structure are implementation specific
Description:	Counter tasks trought the tickflag	

5.5 Enum S_Task

Name:	**S_TASKS**	
Type:	*structure*	
Range:	*Implementation specific structure*	*Structure to hold the module's configuration set. The contents of this data structure are implementation specific.*
Description:	*Configuration of members: period, offset and function of the tasks*	

5.6 Int ModesAndClock

All functions shall be exported in SchModule.h file and defined in SchModule.c file.

Name:	**initModesAndClock**	
syntax	initModesAndClock();	
Parameters(in/out)	Function type	handlerFn
Parameter2(in/out)	Port(N)	unsigned short vectorNum Configuration of required port of the board
Description:	Initialisation of modes and clocks configuration of the micro-controller	

5.7 INTC_InstallINTCInterruptHandler

Name:	**INTC_InstallINTCInterruptHandler**	
syntax	Void INTC_InstallINTCInterruptHandler(function, port,1)	
Parameters(in/out)	Function type	handlerFn
Parameter2(in/out)	Port(N)	unsigned short vectorNum Configuration of required port of the board

Parameter3(in/out)		unsigned char psrPriority
Description:	Contains an implementation of generic interrupt controller handling routines for the MPC56xx and PX MCU families.	

5.8 Tick Flag

Service name:	Tickflag
Syntax:	Void tickflag(Void)
Parameters(in/out)	None
Parameters(in/out)	None
Returns value:	None
Description:	Callback function periodically called from the timer module providing the tick reference

5.9 STM_config_clock

Service name:	**STM_config_clock**
Syntax:	Void **STM_config_clock** (Void)
Parameters(in/out)	None
Parameters(in/out)	None
Returns value:	None
Description:	Configure the STM flag at 1ms and initialize the counter

5.10 Enum Tareas

Name:	**Tareas**	
Type:	Enum	
Range:	Implementation specific enum	enum to hold the module's configuration set. The contents of this data structure are implementation specific.
Location:	Tasks.h	
Description:	add the number of tasks	

5.11 Function void valid_press (void)

Description	*In this function is detected which button is pressed, and the time it is pressed. This make the system go to the desired state (UP/DOWN)*
Parameter 1	*Button UP or Button DOWN pressed*
Return Value	*State*
Precondition	*Buttons must be pressed at least 10 ms*
Post condition	
Error Conditions	

5.12 Automatic_up(void)

Description	This function contains the automatic up movement performed by the window
Parameter 1	***void***
Return Value	***void***
Precondition	Press some valid button.
Post condition	
Error Conditions	

Dynamic Behavior

In this function, the window moves automatically up. The system enters this function when the function Button_Validation(void) detects a valid button press and meets the requirements for automatic mode.

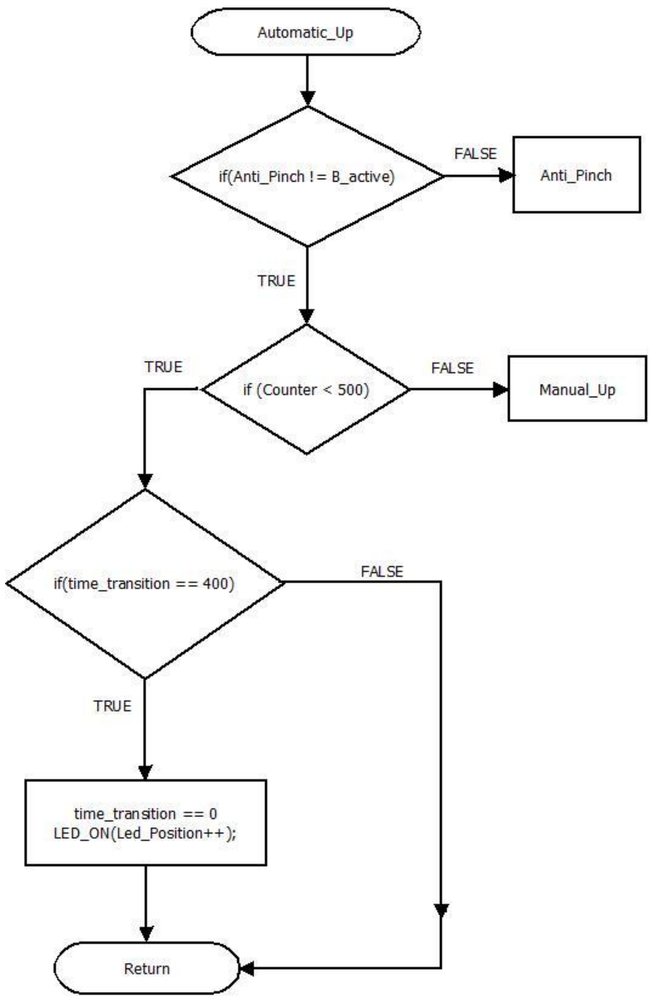

Figure 13: Diagram that represents the behavior of the *Button_Validation();* function.

5.13 Automatic_down(void)

Description	This function contains the automatic down movement performed by the window
Parameter 1	***void***
Return Value	***void***
Precondition	Press some valid button.
Post condition	
Error Conditions	

Dynamic Behavior

In this function, the window moves automatically down. The system enters this function when the function valid_press(void) detects a valid button press and meets the requirements for automatic mode.

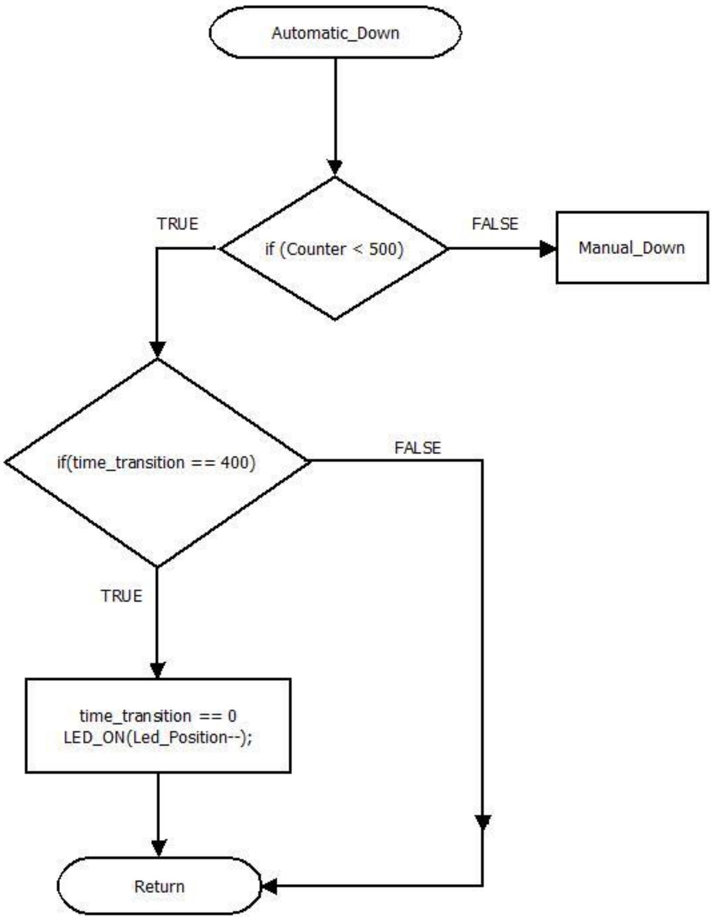

Figure 14: Diagram that represents the behavior of the *Automatic_down();* function.

5.14 Manual_up(void)

Description	This function contains the manual up movement performed by the window
Parameter 1	***void***
Return Value	***void***
Precondition	Press some valid button.
Post condition	
Error Conditions	

Dynamic Behavior

In this function, the window moves manually up. The system enters this function when the function Button_Validation(void) detects a valid button press and meets the requirements needed for manual mode.

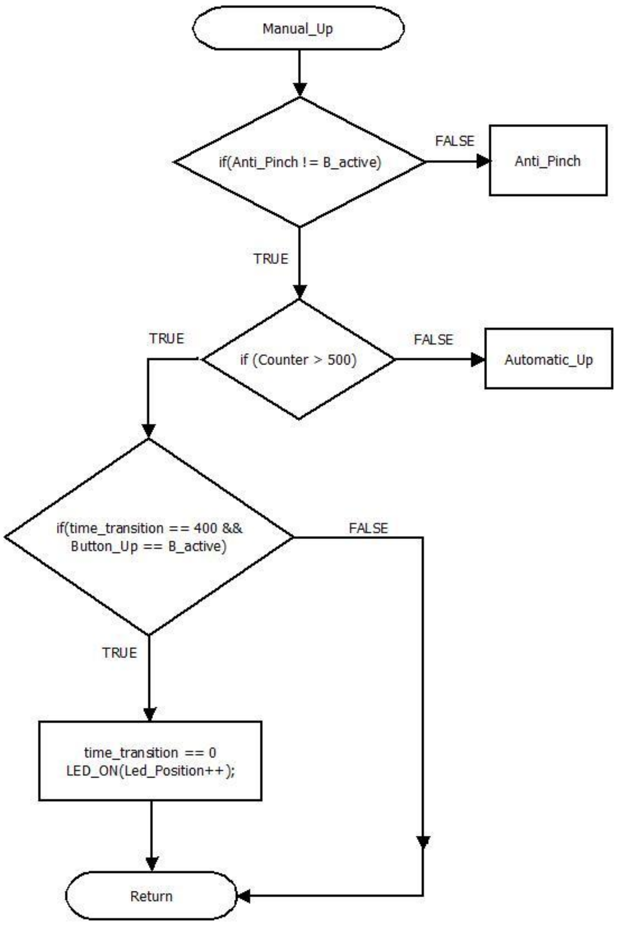

Figure 15: Diagram that represents the behavior of the *Manual_up();* function.

5.15 Manual_down(void)

Description	This function contains the manual down movement performed by the window
Parameter 1	***void***
Return Value	***void***
Precondition	Press some valid button.
Post condition	
Error Conditions	

Dynamic Behavior

In this function, the window moves manually down. The system enters this function when the function Button_Validation(void) detects a valid button press and meets the requirements needed for manual mode.

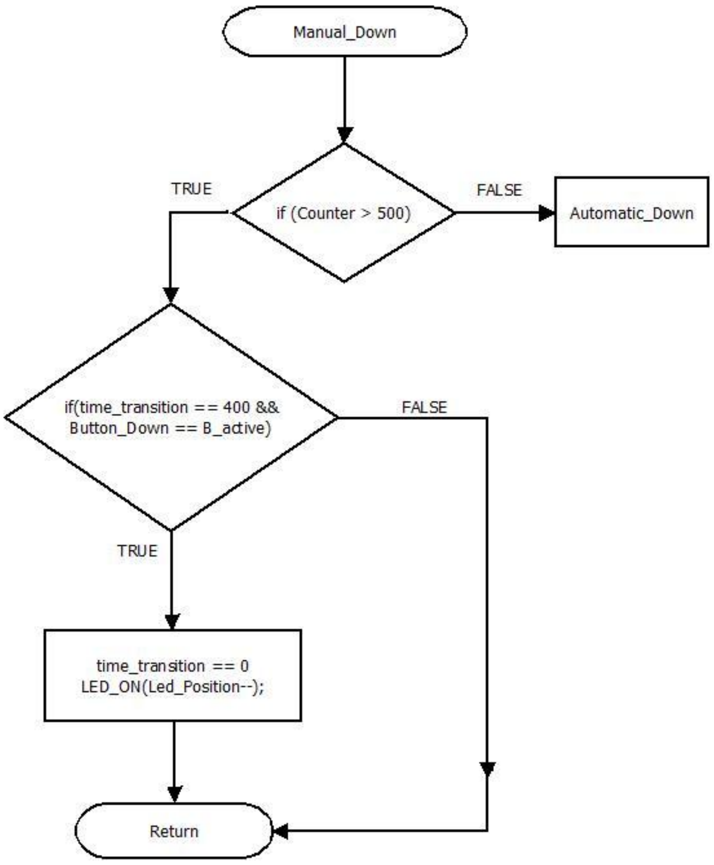

Figure 16: Diagram that represents the behavior of the Manual_up(); function.

5.16 AntiPinch(void)

Description	This function cheks if there's a button pressed representing an obstacle for the window.
Parameter 1	***Button_Up pressed***
Return Value	***Void***
Precondition	Press some valid button.
Post condition	
Error Conditions	

Dynamic Behavior

This function is a safety function. It is responsible of opening the window when an obstacle is detected.

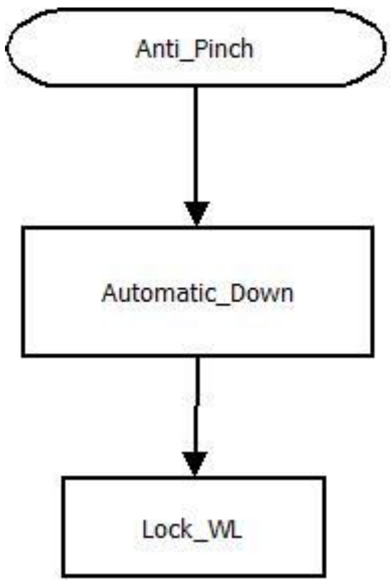

Figure 17: Diagram that represents the behavior of the AntiPinch(); function.

5.17 Lock_WL(void)

Description	This function blocks the system for 5 seconds
Parameter 1	***Anti_Pinch () active***
Return Value	***void***
Precondition	Anti_Pinch detected
Post condition	
Error Conditions	

Dynamic Behavior

This function blocks all the system for five seconds, after the antipinch function was executed it.