Software requirements

Editors:

* Dan Deac
* Bogdan Cupsa
* Roxana Petran

Contents

[1. INTRODUCTION 1](#_Toc1562187374)

[2. COMPONENTS 2](#_Toc1187717274)

[2.1. ADC Module 2](#_Toc298037314)

[2.2 PWM Module 2](#_Toc2113446660)

[2.3 Scheduler Module 2](#_Toc591862690)

[2.4 Ignition Module 2](#_Toc645857945)

[2.5 Mileage Module 2](#_Toc1121119239)

[2.6 Wipers module 2](#_Toc1389331654)

[2.7 Lights module 3](#_Toc642975040)

[2.8 Climate control module 3](#_Toc1344537527)

[2.9 Air Conditioning 4](#_Toc1380652156)

[3. FUNCTIONALITIES 5](#_Toc2038070200)

# 1. INTRODUCTION

The purpose of this project is to simulate a system with real sensors. In real time the data will be read from the ADC channels and written to the PWM channels. Instead of doing this, we will use some files for the inputs and outputs.

The data is read and written to the corresponding files of the modules. If we want to read data, we will read from the 4 ADC channels. The same goes with the PWM module, but in this case, we will write the values taken from the ADC to the 2 PWM channels after we apply some logic on them.

The scheduler is used to automate the tasks and perform them at a given time. In this case we can read the ADC pins every 10 milliseconds, for example. The timers are used to check how much time left remained until another task should start.

While the engine is running and the car is moving, the mileage gets incremented.

The data is read from the channels and placed into the corresponding well-defined structures.

# 2. COMPONENTS

## 2.1. ADC Module

2.1.1. In the sleep state, the ADC pins are not read.

2.1.2. In the wake-up state, the initialization of the ADC channels is being done.

2.1.3. The ADC channels can be activated or deactivated.

2.1.4. In the ADC module we can read from the corresponding file, set values to the four channels and return the values from the specified channel.

## 2.2 PWM Module

2.2.1. The PWM channels can be activated or deactivated.

2.2.2. There are two PWM channels through which we send data.

2.2.3. In the PWM module we can set values to the two channels, return the values from the specified channel and write in the corresponding file.

## 2.3 Scheduler Module

2.3.1. The synchronization is done by scheduler.

2.3.2. The scheduler task can take from 1ms to 10ms and to 100ms.

2.4 Ignition Module

2.4.1. We read ignition value from the corresponding file and set the ignition status flag accordingly.

2.4.2. We should provide the user with a command in order to modify the current state.

## 2.5 Mileage Module

2.5.1. We can read the initial value of the mileage from the corresponding file.

2.5.2. The mileage values are incremented if we are in the KL30 state.

## 2.6 Wipers module

2.6.1. There are three possible states: WIPERS\_OFF, WIPERS\_ON, WIPERS\_AUTO.

2.6.2. The WIPERS\_OFF state is the default state. We reach this state by shutting down the wipers manually.

2.6.3. In the WIPERS\_ON state, the wipers are turned on manually.

2.6.4. In the WIPERS\_AUTO state, we read a value from ADC, based on which the speed of the wipers is automatically controlled.

2.6.5. In each of the cases mentioned above, the current status of the wipers, whether they work or not, is sent to the PWM channel.

## 2.7 Lights module

2.7.1. There are three possible states: LIGHTS\_OFF, LIGHTS\_ON, LIGHTS\_AUTO.

2.7.2. The LIGHTS\_OFF state is the default state. We reach this state by shutting down the lights manually.

2.7.3. In the LIGHTS\_ON state, the lights are turned on manually.

2.7.4. In the LIGHTS\_AUTO state, we read a value from ADC, based on which the speed of the lights is automatically controlled.

2.7.5. In both LIGHTS\_AUTO and LIGHTS\_ON states we can modify the intensity of the lights (High or Low beam). This can be done manually or automatically by reading and processing a value from ADC, reaching the following three states: HB\_OFF (low beam), HB\_ON (high beam), HB\_AUTO (automatically by reading from ADC).

2.7.6. In each of the cases mentioned above, the current status of the lights, whether they work or not, is sent to the PWM channel.

## 2.8 Climate control module

2.8.1 There are three possible states: AUTO\_ON, AUTO\_OFF, AUTO\_AC.

2.8.2 The AUTO \_OFF state is the default state. We reach this state by shutting down the AUTO module manually, by setting the AUTO bit in climateButtons.txt to 0.

2.8.3 If AC is on and then is also turned on, the module will take control of the climate control, and will either let AC on (if cooling is necessary) or turn it off.

2.8.4 In the AUTO\_ON state, the climate control module takes information from temperature sensors in the car, through the ADC module.

2.8.5 The sensors read are the cabin temperature, outside temperature, and air quality from outside. These are on ADC channels 5,6,7 in the order mentioned above.

2.8.6 In AUTO\_ON state, the module constantly makes sure that the temperature requested by the user coincides with cabin temperature.

2.8.7 In AUTO\_ON, the cooling fan gauge is ignored, and the air flow will be controlled by the module, unless the user modifies the fan gauge. If this happens the AUTO module will be turned off and the AC will be turned on consequently.

2.8.8 In AUTO\_ON state, temperature in the cabin is controlled by mixing cold air with the hot air coming from the heating core. It will try to maintain the temperature the driver sets, by controlling the speed of the fan, and a valve which is the blend door, deciding how much hot air gets mixed with the cool air from the AC.

2.8.9 When heating is needed, AC will be turned off, and the amount of heated air blowing in the cabin will be controlled by the blend door. If the temperature is cold in the cabin, and the user chooses a temperature over a certain threshold, the blend door valve will open and allow as much heated air to flow in the cabin as possible.

2.8.10 Climate control state is controlled by the **climate\_state** module.

2.8.11 There is a recirculation button, which if pressed, closes the valve which allows air from the cabin filter (exterior) to enter the cabin and recirculates the air in the car.

2.8.12 When on AUTO\_ON, the quality of the air in the car is compared with the one outside. If there is a car in front that emits smoke, the valve with intake will be closed, and air will be automatically recirculated. If the air is fine outside, it will open the valve and allow fresh air in.

2.8.13 If the temperature in the cabin is way colder than the requested temperature (the car just started, and it's probably winter) the car will close the air intake valve (from outside) and recirculate the air, so it warms the interior faster.

## 2.9 Air Conditioning

2.9.1 There are 2 possible states: AC\_ON\_OFF = **true**, AC\_ON\_OFF = **false**, controlled by the value (1- on, 0-off) of the AC bit in climateButtons.txt.

2.9.2. The AC\_ON\_OFF is set to **false** (0) at the beginning. We reach this state by writing 0 to the AC bit in the buttons file.

2.9.3 The AC\_ON\_OFF = **true** state can be reached if it is turned on by the AC bit being 1, or the AUTO mode decides it needs to cool down the cabin and starts the AC.

2.9.4 The AC\_ON\_OFF = **true** state can be also reached when “turning” the fan gauge, represented by changing the numeric value in the climateButtons.txt file (corresponding to the fan gauge), even if the AUTO module is on. This action will override the AUTO bit to 0.

2.9.5. In the AC\_ON\_OFF = **true** state, the AC module is turned on. Depending on the speed chosen by fan gauge, it will spin the cooling fan, accordingly, blowing only cold air into the cabin.

2.9.6. The AC will start the cooling mechanism which always works at full power, its either on, or off, not in between. Changing the temperature of the air blowing in the cabin is done in the AUTO module by a blend door, which mixes in the cool air from the AC radiator with the heat from the heating mechanism.

2.10 Rear and Front windshield demist function.

2.10.1 When either of the buttons are pressed – the value in the climateButtons.txt of the corresponding bits is 1, then this functionality will start the AC and blow air on the windshield. The reason is also to use the cooling mechanism to dry the air and drop humidity.

2.10.2 When rear button is pressed (bit is 1 in climateButtons.txt file), it will turn on the heating mechanism using the resistance in the glass, AC will not get turned on.

# 3. FUNCTIONALITIES

These are the functionalities of the project:

3.1 The system has the following states:

In the SLEEP state nothing happens, and the ADC pins are not read.

In the WAKE\_UP state the ADC module is initialized.

In the RUN state both the ADC and PWM modules are initialized, the synchronization being done by a scheduler.

3.2. The ADC module can be activated or deactivated.

3.3. There are 7 ADC Channels where we set or return the values that are read from the files.

3.4. The PWM module ca be activated or deactivated.

3.5. There are 6 PWM channels witch we send data to.

3.6. The PWM module will be used for displaying the outputs.

3.7. The synchronization is done by scheduler.

3.8. The scheduler tasks can take 1, 10 and 100 milliseconds.

3.9. The mileage has a previous value that is read from the corresponding file and then gets incremented while we are in the running state.

3.10. The initial ignition value is read from the ignition file and the ignition status flag is modified accordingly.

3.11. We change the system's Klemme state to KL0 and KL30 according to the ignition status flag.

3.12. In the terminal, the user interface asks for a command and then the KL state is modified. After each Kl state is done, we are asked again to provide a command: 'A' / 'a' for KL0, 'B'/'b' for KL15, 'C'/'c' for KL30.

3.13. In the case of KL30\_RUN STATE, the state is running for 6 iterations and then we are asked again if we want to stay in the running state or change it.

3.14. We can read the values from ADC channel 1 only if we are in KL15 and KL30 states. In KL0 state we can't perform this action as ADC is disabled.

3.15. While in KL30, all the critical systems in the car have power and this includes the lights, but not the wipers and the climate module.

3.16. While in KL15, all the modules have power. The user can set one by one the values starting from the lights, then wipers and then climate modules. This is done by asking the user the state in which we want the modules to be in, for wipers and lights.

3.17 In KL15 state, we are able to start the climate module. Given that there are a lot of buttons to tweak, this is done by changing values in the climateButtons.txt file. To be able to better see the results, the climate module is put in a polling loop, after it ran once. This is done so the module can easily showcase the functionality without running the program again and having to select lights, wipers and so on every time.

3.18 In KL15, while in the climate module, the polling functionality can be stopped by pressing ‘n’ or ‘N’ when asked either to continue or not. In case another button is pressed (i.e ‘y’, ‘enter’ etc.), the module will run the module again, reading the newly changed data (if the user modified it) from the climateButtons.txt and the ADC module file with the sensor data.

3.19. After the climate module is stopped by pressing ‘n’ or ‘N’, the program will finish the climate module and ask again which Klemme state is desired by the user.

3.20. At the beginning of each Klemme state function, all the PWM channels that should not get power are turned off and their value is set to 0. Example, if KL15 state was active, and lights were on, when moving on to KL30, all the other channels will be 0 except the lights channel. So, the lights will be kept on, as they are the only module in this project that should run during KL30 state.

