MBSD Assignment #3 A.Y. 2024/25

# Purposes

* Perform some parts of the Functional and Technical Safety Concept analysis, according to ISO26262, of a “one pedal controller” for a car.
* Implement some of the safety concepts in the Simulink model of the controller developed in Laboratory #2.
* Perform unit and integration tests on the implemented safety-related functionalities.

It is available an example of a Functional Safety Concept for the item Front Light Manager (FLM).

The deliverable, composed of

* the report (the following pages of this document)
* the Simulink models on where the safety concepts have been implemented
* all the needed files to replicate the software testing results

has to be provided as a .ZIP file (do not use other compression formats) up to **May 14th at 23:59.** It shall also contain a brief report explaining the design of the controller using the following template.

It is sufficient that only one of the group members uploads it.

**IMPORTANT:**

For making the Functional Safety Concept coherent between the groups of the course, consider as ASIL C all the safety goals related to unintended acceleration (those leading to an increase of the vehicle’s speed modulus) and as ASIL B the warnings to the driver and the unintended deceleration (those leading to a decrease of the vehicle’s speed modulus).

# Model-Based Software Design, A.Y. 2022/23

# Laboratory 3 Report

## Components of the working group (max 2 people)

* Federico Bena, 328197
* Michele Lauriola, 332679

Functional Safety Concept

One pedal

# Functional safety architecture

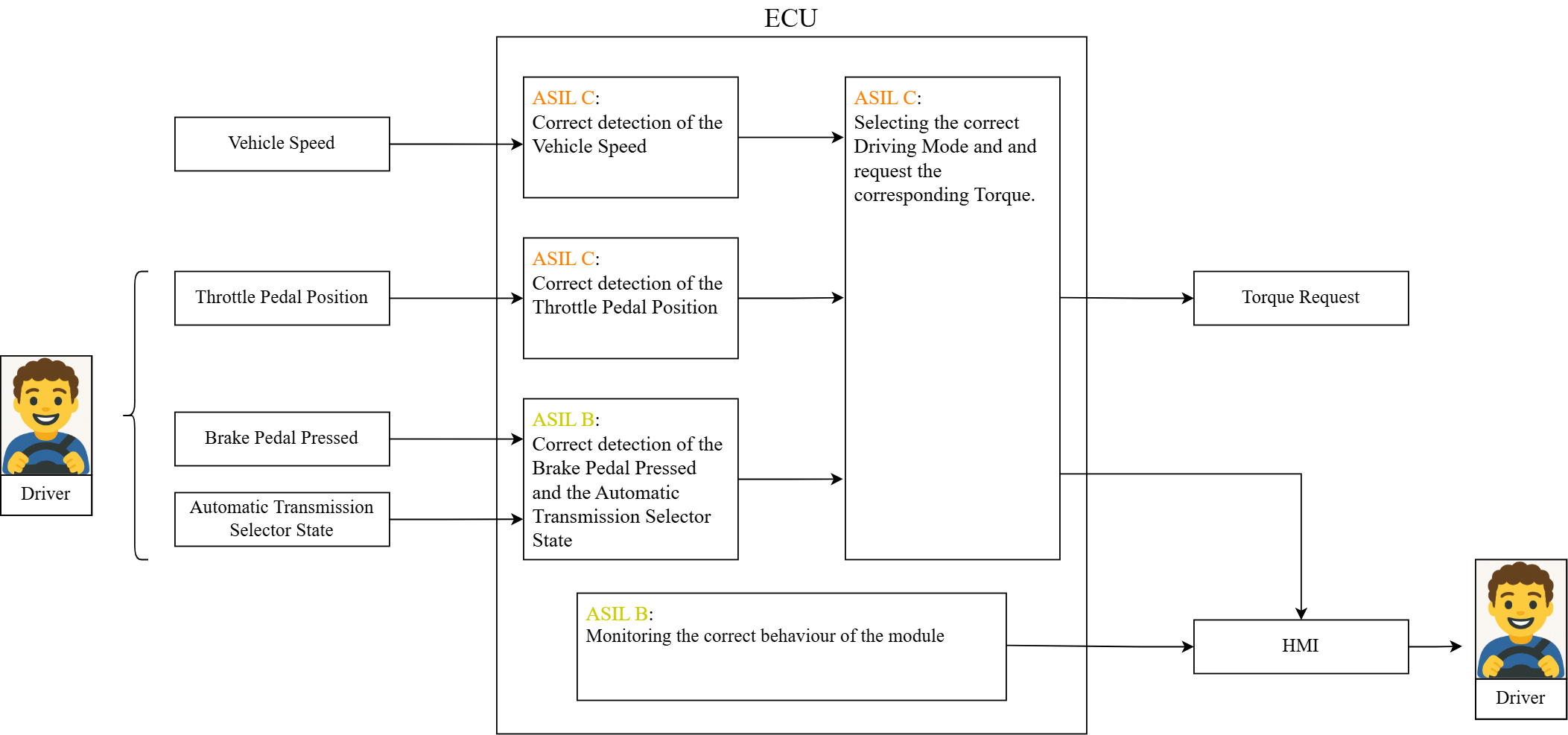


Figure 1 Functional safety architecture

# Attributes of the Safety Goals

*Fill the table with the safety goal descriptions*

|  |  |
| --- | --- |
| **SG 1** | The item shall compute correctly the torque requested by the driver. |
| **SG 2** | The item shall monitor that the transmission mode displayed is the correct one. |
| **SG 3** | The item shall transit to safe state when a misbehavior is detected. |

*Fill in the attribute/parameters of the safety goal*

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Safety goal** | **Attributes/Parameters of the safety goal** | | | | |
| Integrity (ASIL) | Safe state | Fault tolerance time | Warning concept | Degradation concept |
| **SG 1** | **C** | Prevent unintended or insufficient acceleration and deceleration. | 100 ms | Alert the driver of the malfunction by displaying a notification on the dashboard and emitting a beeping sound or an alarm. | No. |
| **SG 2** | **B** | Prevent the wrong Transmission mode from being displayed. | 300 ms | Alert the driver of the malfunction by displaying a notification on the dashboard and emitting a beeping sound or an alarm. | No. |

# Functional (and technical) safety requirements and allocation

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | | **Define functional safety requirements** | | **Allocation of requirements on systems and elements** | |
| **Safety requirements** | **Remark** | **If applicable, allocate the safety requirements to other Items / Systems** | **If applicable, allocate the safety requirements to equipment other technologies to minimize risk.**  **That could be e.g. hydraulic, mechanical equipment** |
| **Safety goals** | **SG1:** The item shall compute correctly the torque requested by the driver. | **SR1**: If one throttle pedal sensor is faulty, the system will compute the average of the other two working sensors. | The three sensors are independent from each other. | A warning message is displayed on the dashboard. | No. |
| **SR2**: If all the throttle pedal sensors are faulty (at least two of them), the transmission state changes to neutral. | The three sensors are independent from each other, moreover the torque request is set to 0. | The System must transit to safe state, and a notification about the fault is displayed on the dashboard accompanied by an alarm. | No. |
| **SR3**: The warning  remains on until a  correct  measurement is  detected. | No. | No. | No. |
| **SR4**: The system shall compare  the torque requests  against the expected  values, in case  of fault, the transmission state changes to neutral. | In case of Fault the torque request is set to 0. | The System must transit to safe state, and a notification about the fault is displayed on the dashboard accompanied by an alarm. | No. |
| **SG2:** The item shall monitor that the transmission mode displayed is the correct one. | **SR5:** If the displayed mode is different from the actual one the system must transit to safe state. | In the case of Fault, the torque request is set to 0 within a certain time tolerance. | The System must transit to safe state, and a notification about the fault is displayed on the dashboard accompanied by an alarm. | No. |
| **SR6**: The warning  remains on until a  correct  mode is displayed. | No. | No. | No. |

# ASIL preliminary architecture[[1]](#footnote-2)

*The purpose of this figure is to assign ASILs to the components indicated in Figure 1 on the functional safety architecture.*

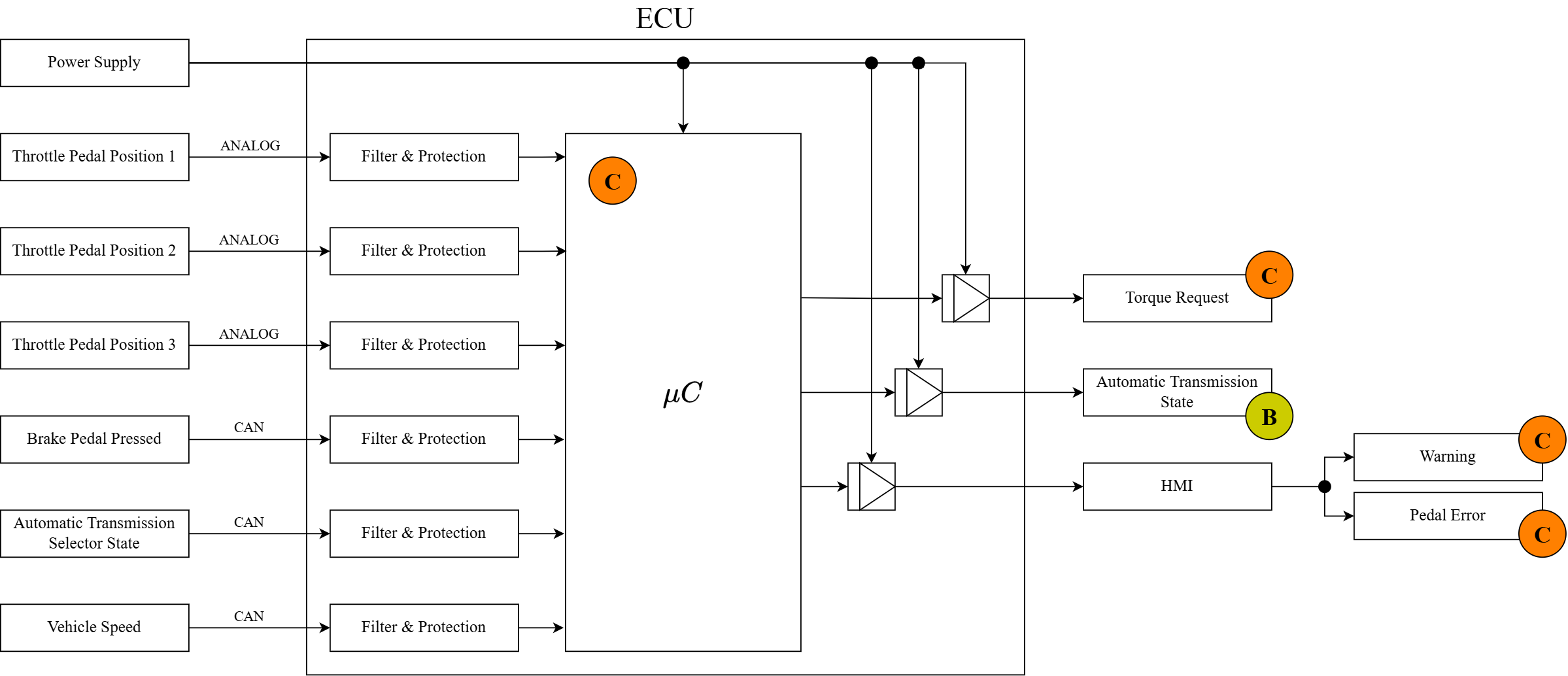


Figure 2 Preliminary architecture (without ASIL decomposition).

# Implementations[[2]](#footnote-3)

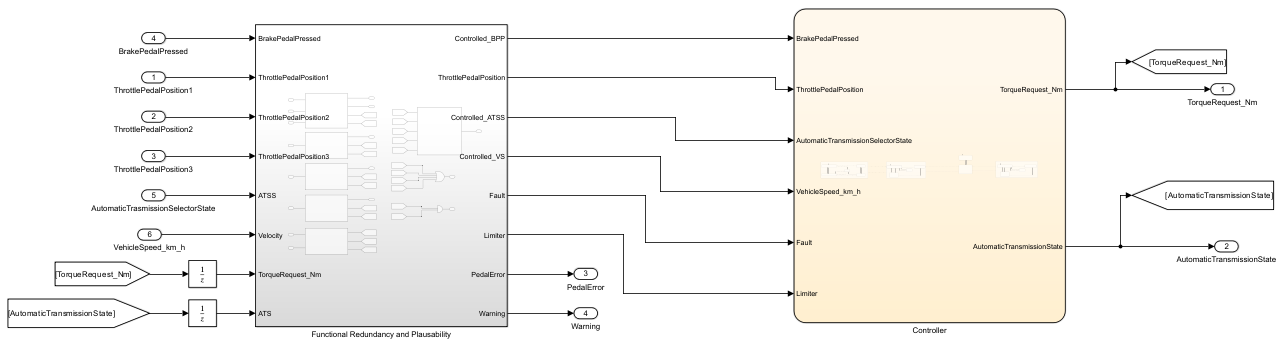


Figure 3: Implementation of Functional Redundancy and Plausability

Figure 3 shows that the **Saturation** blocks, used in the previous assignment, have been included as checks in the **Functional Redundancy and Plausability** block.

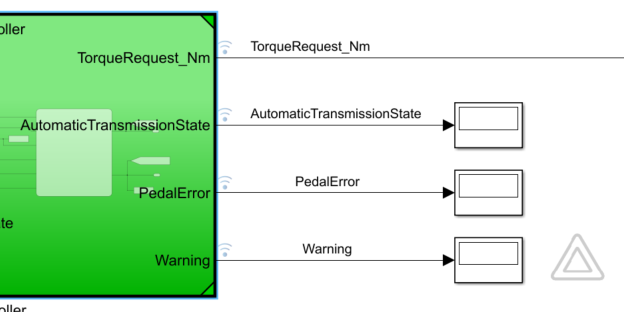


Figure 4: Implementation at Harness level

In the following table, the new variables, used for the **Functional Redundancy and Plausability** block, are defined:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Name** | **Type** | **Data Type** | **Dimension** | **Min** | **Max** |
| Limiter | *Input* | Boolean | 1x1 | 0 | 1 |
| Fault | *Input* | Boolean | 1x1 | 0 | 1 |
| PedalError | *Output* | Single | 1x1 | 1 | 4 |
| Warning | *Output* | Enum: WarningType\* | 1x1 | Off(0) | Red(2) |

\*classdef WarningType < Simulink.IntEnumType

enumeration

Off(0)

Yellow(1)

Red(2)

end

end

## Functional redundancies

*Indicate which components of the functional safety architecture are implemented with redundancies.*

Functional redundancies are implemented, as safety measures, through two additional sensors that measure independently the position of the throttle pedal allowing to detect potential failures or outliers.

Within the **Functional Redundancy and Plausibility** block, the Matlab function **Voter\_TPP** receives the input measurements from the three sensors (**ThrottlePedalPosition1**, **ThrottlePedalPosition2** and **ThrottlePedalPosition3**) and it performs the following actions:

* Initializes the variables **Warning** and **Fault** used respectively to alert the driver of the presence of an anomaly or a more severe fault through the dashboard and to transit into a safe state.
* Computes, in absolute value, the differences between each pair of measurements (**Diff\_1\_2**, **Diff\_1\_3** and **Diff\_2\_3**).
* Performs the various if-cycles to check the consistency of the measurements by comparing the computed differences to a fixed threshold of 2 %, returning three possible outcomes:
  1. The three measurements are consistent. The **TPP** (ThrottlePedalPosition) output is computed as the average of the three measurements (no **Warning**).
  2. One of the three measurements deviates from the other two. The **TPP** output is given as the average of the latter, the **PedalError** shows the number of the faulty sensor and the variable **Warning** is set to ‘1’, corresponding to the yellow Lamp on the dashboard which means that the vehicle is not working properly but still able to provide its functions: the driver must service the vehicle as soon as possible.
  3. The three measurements are inconsistent with each other. The **PedalError** shows the number ‘4’, the **Fault** variable is set to “1” so the system transits into a safe state and the **Warning** is set to ‘2’, which means that all the sensors are inconsistent between each other and so the driver must stop immediately the vehicle and call assistance.

## Implemented plausibility checks

*Indicate which signals are subjected to plausibility checks. Indicate also the checks needed for the channels with redundancies in place.*

Plausibility checks are implemented to verify the consistency of the input signals coming from CAN. Such evaluations are performed through MATLAB function blocks included in the previously introduced **Functional Redundancy and Plausibility** block:

* **BPP\_Plausability**: if the data *BrakePedalPressed* is not a Boolean value, **Fault\_BPP** is set to “1” and **Warning\_BPP** is set to ‘2’.
* **ATSS\_Plausability**: if the data *AutomaticTransmissionSelectorState* is not an integer between 0 and 4, corresponding to the available Transmission modes, the **Fault\_ATSS** variable is set to “1” and **Warning\_ATSS** is set to ‘2’.
* **TR\_Plausability**: the consistency of the controller outputs *TorqueRequest\_Nm* and *AutomaticTransmissionState* is checked. When the torque is: zero OR a positive value in ‘D’ or ‘B’ mode OR a negative value in ‘R’ or ‘B’ mode, if the torque exceeds the range [– 80,80] Nm, **Limiter\_TR** is set to “1” and **Warning\_TR** is set to ‘1’, otherwise **Fault\_TR** is set to “1” and **Warning\_TR** is set to ‘2’.
* **VS\_Plausability**: if the data coming from the *VehicleSpeed\_km\_h* excedees the range [-60,240] km/h the variable **Limiter\_VS** is set to “1” and **Warning\_VS** is set to ‘1’.

A diagram of a computer

AI-generated content may be incorrect.

Figure 5: Functional Redundancy and Plausability Block

As can be noticed in Figure 5, it should be pointed out that:

* For the several types of faults and limiters, logical operators OR are implemented returning the two generic Boolean variables **Fault** and **Limiter**.
* Since the warnings are *Enum: WarningType*, a Matlab function, that checks them all and returns as output the higher valueof **Warning**, is implemented.

Figure 6 shows the addition of the **Fault** and **Limiter** variables in the transitions (highlighted in yellow) within the Controller State Machine:

* “Fault == 1” is added as a logical OR to all the transitions towards the Neutral State, including from ‘Brake’ to ‘Drive’.
* “Fault == 0” is added as a logical AND to all the transitions from ‘Neutral’ to the other States, including from ‘Drive’ to ‘Brake’.

This implementation makes sure that the current State is set to ‘Neutral’ if the **Fault** is active, and that the item is allowed to change State otherwise.

As concerns the **Limiter** new sub-states have been created to prevent the controller from leaving its current state while still maintaining the limiter function for speed and torque:

* “Limiter == 1” is added as a logical OR to all the transitions towards the “Limiter” Sub-State, or to the “EmergencyStop” for the ‘Brake’.
* “Limiter == 0” is added as a logical AND to all the transitions from “Limiter” or “EmergencyStop” to the other Sub-States.

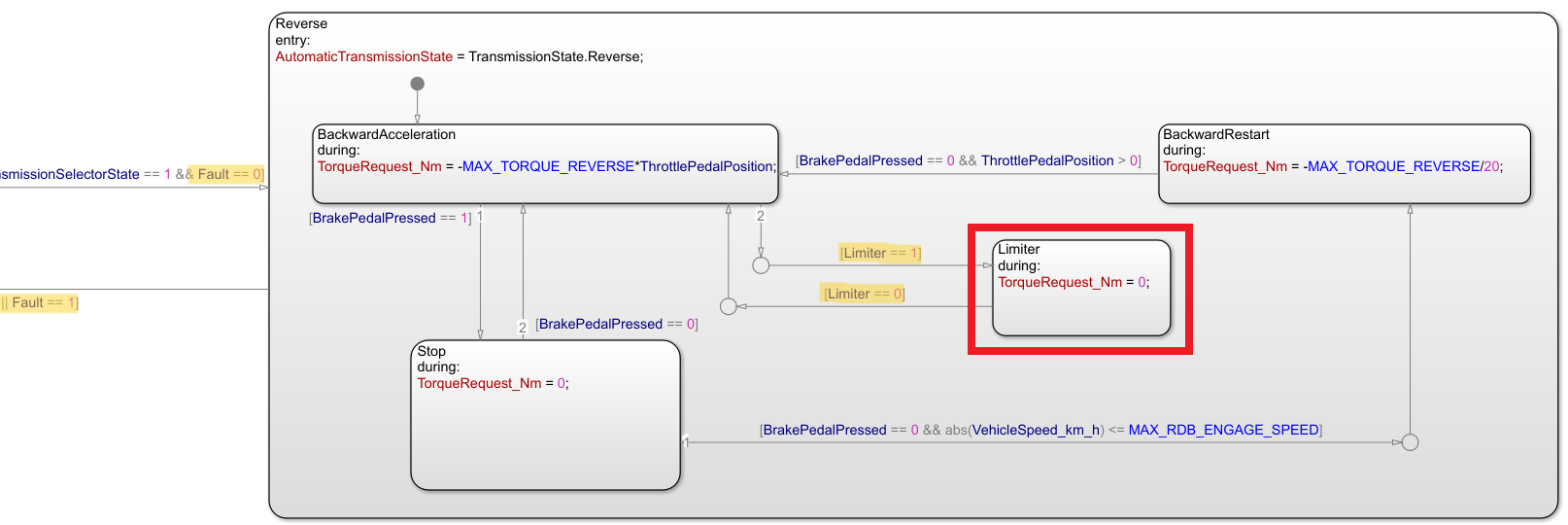
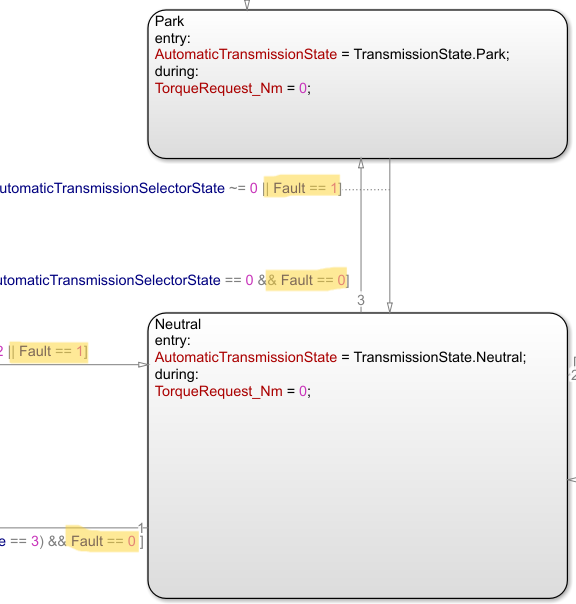
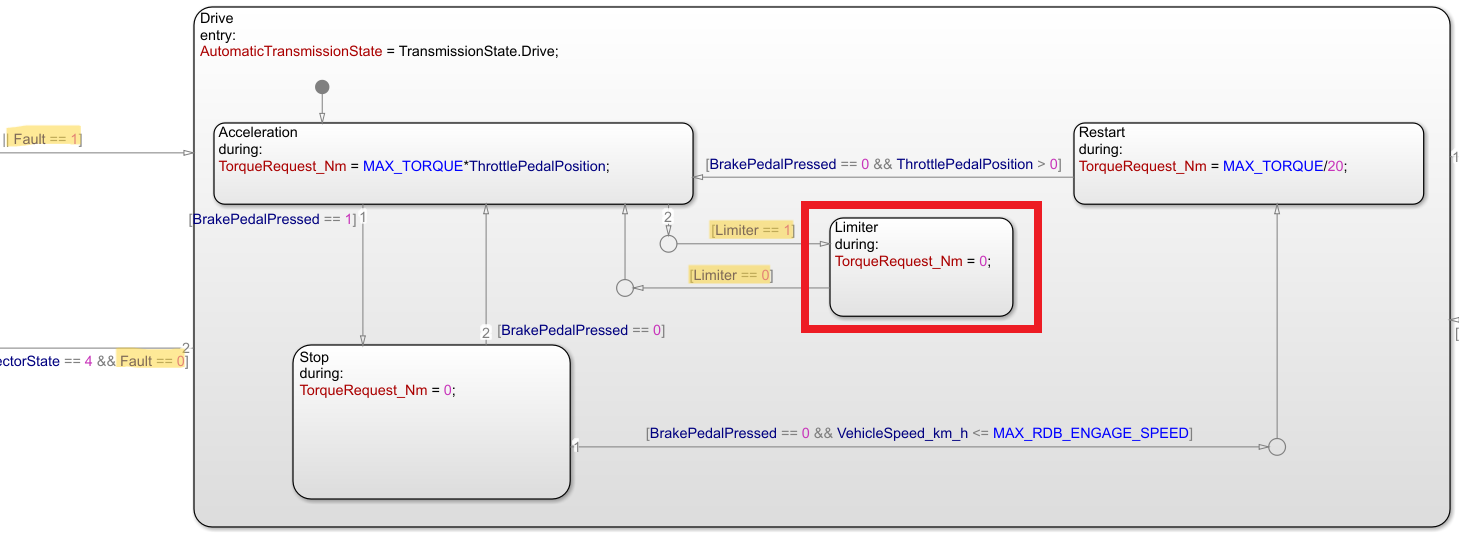
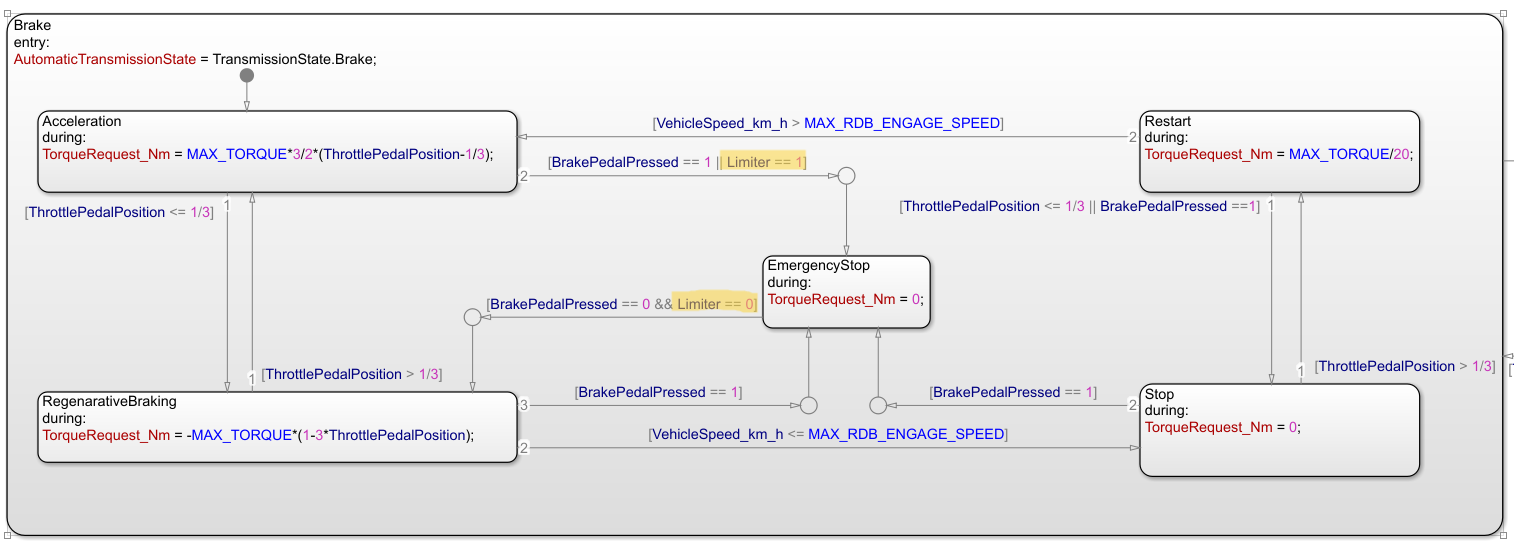


Figure 6: Fault and Limiter variables in the State Machine's Transitions

# Software testing

## Implemented unit tests

*Describe in English the test performed to verify the correct functionality of the safety mechanism implemented.*

Simulink Coverage is used for all the subsystem blocks, shown on Figure 5, to check whether the implemented Matlab functions result in 100% coverage.

Following are: the tables concerning the values used for the input blocks **Repeating Sequence Stair**, the sample times and the simulation time (adapted according to each unit test) and the Figures to presents the coverage results and the trends of the variables of interest, obtained through a **Scope** block.

* **Voter\_TPP Unit Test**

|  |  |  |  |
| --- | --- | --- | --- |
| **Simulation Time [s]** | | | 6 |
| **Repeating Sequence Stair** | **Vector of values** | **Sample Time [s]** | |
| *ThrottlePedalPosition1* | [0.8].' | 6 | |
| *ThrottlePedalPosition2* | [0.8 0.81 0.81 0.9 0.62 0.9 ].' | 1 | |
| *ThrottlePedalPosition3* | [0.8 0.79 0.7 0.79 0.6 0.6].' | 1 | |

**Immagine che contiene testo, diagramma, linea, Diagramma

Il contenuto generato dall'IA potrebbe non essere corretto.**

Figure 7: Coverage of Voter\_TPP

A red line on a white background

AI-generated content may be incorrect.

Figure 8: Input (Top) & Output (Bottom) Scope Voter\_TPP

Figure 7 and Figure 8 show that block **Voter\_TPP** behaves as expected.

* **BPP\_Plausabiliy Unit Test**

|  |  |  |  |
| --- | --- | --- | --- |
| **Simulation Time [s]** | | | 5 |
| **Repeating Sequence Stair** | **Vector of values** | **Sample Time [s]** | |
| *BrakePedalPressed* | [0 2 1 -0.5 0].' | 1 | |

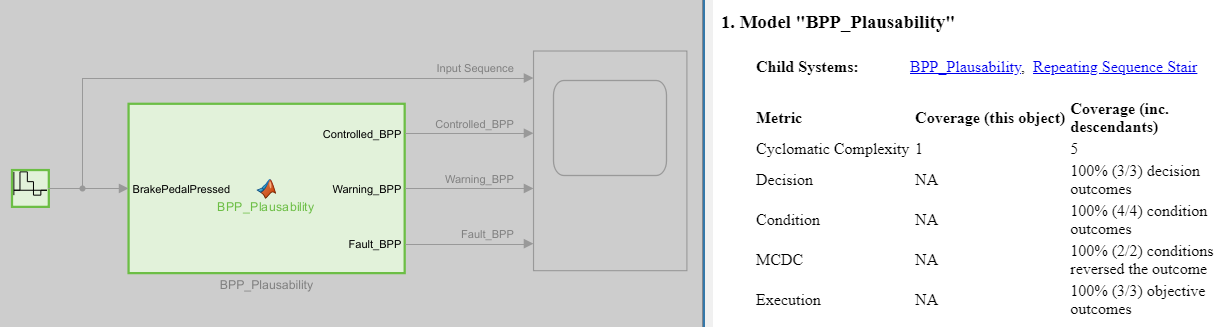


Figure 9: Coverage of BPP\_Plausability

Immagine che contiene diagramma, linea, Parallelo, testo

Il contenuto generato dall'IA potrebbe non essere corretto.

Figure 10: Input & Output Scope BPP\_Plausability

Figure 9 and Figure 10 show that block **BPP\_Plausability** behaves as expected.

* **ATSS\_Plausabiliy Unit Test**

|  |  |  |  |
| --- | --- | --- | --- |
| **Simulation Time [s]** | | | 11 |
| **Repeating Sequence Stair** | **Vector of values** | **Sample Time [s]** | |
| *AutomaticTransmissionSelectorState* | [-1 0 0.5 1 1.5 2 2.5 3 3.5 4 5].' | 1 | |

Immagine che contiene testo, Carattere, linea, schermata

Il contenuto generato dall'IA potrebbe non essere corretto.

Figure 11: Coverage of ATSS\_Plausability

Immagine che contiene diagramma, linea, Parallelo, testo

Il contenuto generato dall'IA potrebbe non essere corretto.

Figure 12: Input & Output Scope ATSS\_Plausability

Figure 11 and Figure 12 show that block **ATSS\_Plausability** behaves as expected.

* **VS\_Plausabiliy Unit Test**

|  |  |  |  |
| --- | --- | --- | --- |
| **Simulation Time [s]** | | | 16 |
| **Repeating Sequence Stair** | **Vector of values** | **Sample Time [s]** | |
| *Velocity* | [-80 -80 -60 -60 -30 -30 0 0 80 80 200 200 240 240 280 280 ].' | 1 | |

Immagine che contiene testo, diagramma, schermata, linea

Il contenuto generato dall'IA potrebbe non essere corretto.

Figure 13: Coverage of VS\_Plausability

Immagine che contiene testo, linea, diagramma, Parallelo

Il contenuto generato dall'IA potrebbe non essere corretto.

Figure 14: Input & Output Scope VS\_Plausability

Figure 13 and Figure 14 show that block **VS\_Plausability** behaves as expected.

* **TR\_Plausabiliy Unit Test**

|  |  |  |  |
| --- | --- | --- | --- |
| **Simulation Time [s]** | | | 25 |
| **Repeating Sequence Stair** | **Vector of values** | **Sample Time [s]** | |
| *TorqueRequest\_Nm* | [-90 -45 0 45 90].' | 1 | |
| *AutomaticTransmissionState* | [0 1 2 3 4].' | 5 | |

Immagine che contiene testo, schermata, diagramma, Carattere

Il contenuto generato dall'IA potrebbe non essere corretto.

Figure 15: Coverage of TR\_Plausability

A diagram of a graph

AI-generated content may be incorrect.

Figure 16: Input & Output Scope TR\_Plausabilitys

Figure 15 and Figure 16 show that block **TR\_Plausability** behaves as expected.

* **Warning\_Detection Unit Test**

|  |  |  |  |
| --- | --- | --- | --- |
| **Simulation Time [s]** | | | 30 |
| **Repeating Sequence Stair** | **Vector of values** | **Sample Time [s]** | |
| *Warning\_TPP* | [0 1 2].' | 2 | |
| *Warning\_BPP* | [0 0 2].' | 2 | |
| *Warning\_ATSS* | [0 0 2].' | 4 | |
| *Warning\_VS* | [0 1 0 0 1].' | 1 | |
| *Warning\_TR* | [0 1 0 2].' | 2 | |

A screenshot of a computer

AI-generated content may be incorrect.

Figure 17: Coverage of Warning\_Detenction

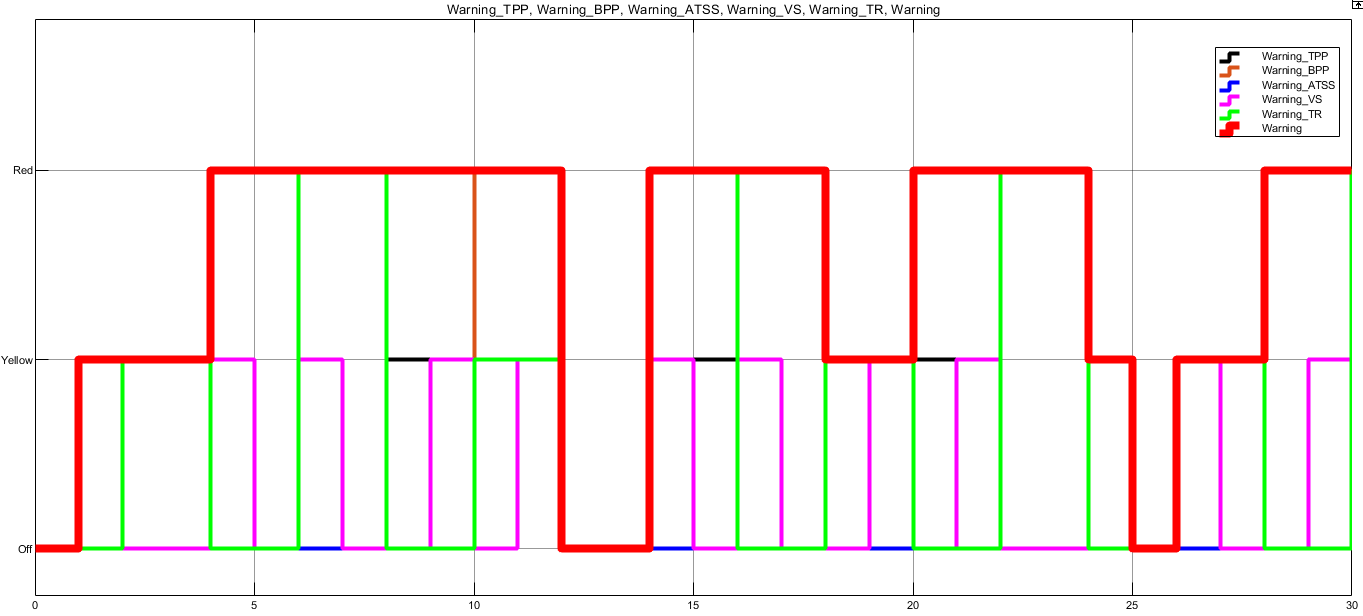


Figure 18: Input & Output Scope Warning\_Detenction

Figure 17 and Figure 18 show that block **Warning\_Detenction** behaves as expected.

## Implemented integration tests

*Describe, in English, the scenarios tested at the integration level to verify the proper integration between the various units implementing the safety mechanisms.*

A diagram of a model based software test

AI-generated content may be incorrect.

Figure 19: Harness with step used to corrupt some signal for Testing purposes

Figure 19 shows how the harness environment has been modified, with the addition of some step block, in order to corrupt and then readjust some input signal, so simulating the behavior of the controller in the presence of disturbed signal.

* **Scenario 1:** Corruption at the throttle pedal signals level.

A screenshot of a graph

AI-generated content may be incorrect.

Figure 20: Result whit only the Throttle Pedal sensors corruption

As shown in Figure 20, the third sensor is corrupted at 10 s and the vehicle displayes a yellow warning remaining in the same AutomaticTransmissionState. When the first sensor is corrupted at 20 s, the vehicle displays a red warning entering in the Safe State (*AutomaticTransmissionState* = Neutral).

* **Scenario 2:** Corruption at the brake pedal signal level.

A screenshot of a graph

AI-generated content may be incorrect.

Figure 21: Result whit only the BrakePedalPressed signal corrupted

As shown in Figure 21, at 10 s the *BreakePedalPressed* signal is corrupted, the vehicle displayes a red warning entering in the Safe State (*AutomaticTransmissionState* = Neutral).

* **Scenario 3:** Corruption at the speed signal level.

A screenshot of a graph

AI-generated content may be incorrect.

Figure 22: Result whit only the Feedback Speed signal corrupted

As shown in Figure 22, at 10 s the feedbacked *VehicleSpeed\_km\_h* signal is corrupted, the vehicle displayes a yellow warning remaining in the same *AutomaticTransmissionState*.

* **Scenario 4:** Corruption of all the signals.

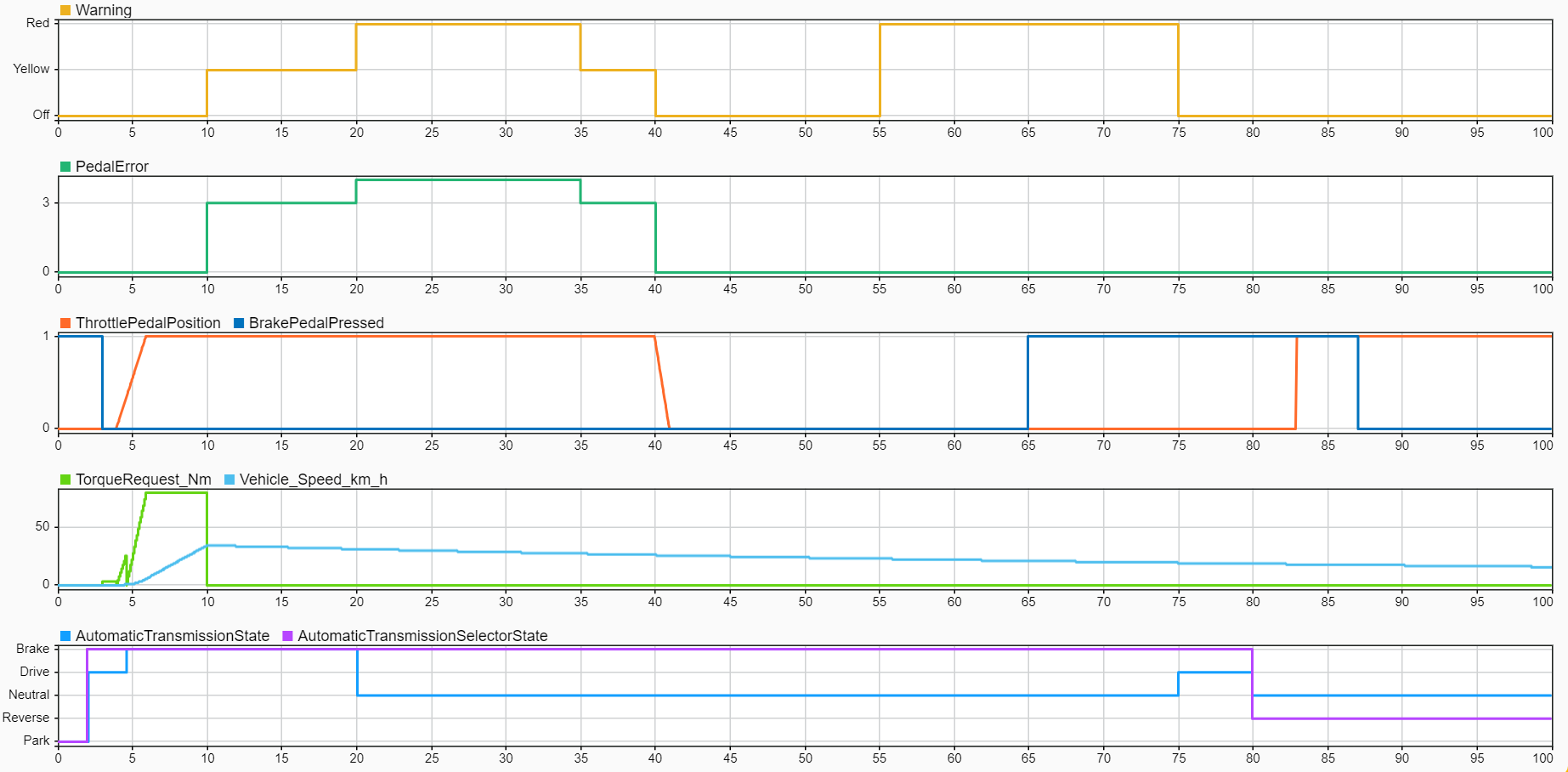


Figure 23: Result whit all the signals corrupted

Figure 23 shows what happens in the case in which BrakePedalPressed signal is corrupted at 55 s (achieving the same result of Figure 21), while the *ThrottlePedalPosition* and the *VehicleSpeed*\_km\_h corruption are set in a way to overlap them and to show the right functioning of the **Detection\_Warning** block.

* **Scenario 5:** Original test (no signal corruption).

A graph of a graph

AI-generated content may be incorrect.

Figure 24: Sparklines results

A graph with lines and numbers

AI-generated content may be incorrect.

Figure 25: Errors and Warnings

Figure 24 and Figure 25 show the behaviour of the system in the original test conditions resulting clearly that, although a whole part dedicated to checking the validity of input signals has been added to the controller, the dynamic response of the system remains unchanged when compared to the previous report.

From that it is possible to state that the changes made do not affect the response any way.

1. See document 02-iso26262.pdf, slides 89, 90, 91, 92, 93. [↑](#footnote-ref-2)
2. In the ISO26262 the implementations are based on a document called *Technical Safety Concept*, but for simplicity we move straight from the *Functional Safety Concept* to software implementations.

   A guideline for the implementation phase can be found in the document 02-iso26262.pdf from slide 81, in particular slide 86. [↑](#footnote-ref-3)