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

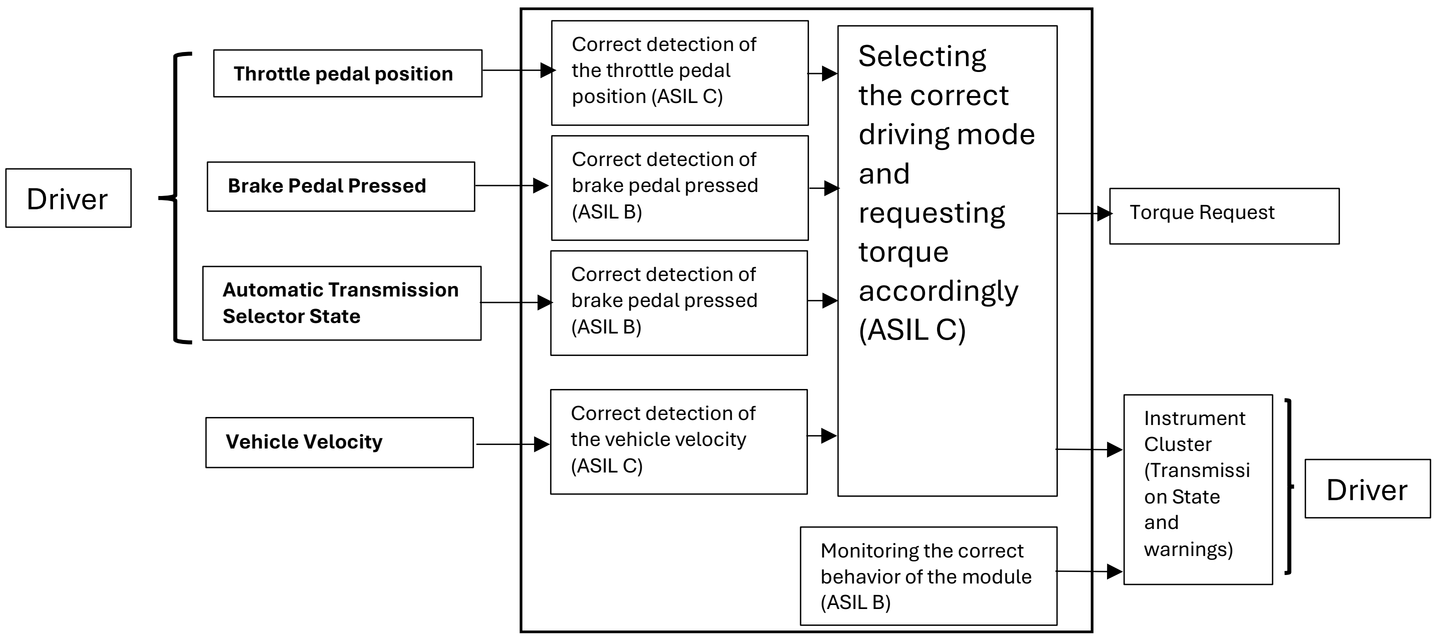
Components of the working group (max 2 people)

- Amirhossein Ayanmanesh Motlaghmofrad, s323874

Functional Safety Concept

One pedal

Functional safety architecture



Driver
Throttle pedal position
Brake Pedal Pressed
Automatic Transmission Selector State
Vehicle Velocity
Correct detection of the throttle pedal position (ASIL C)
Correct detection of brake pedal pressed (ASIL B)
Correct detection of brake pedal pressed (ASIL B)
Correct detection of the vehicle velocity (ASIL C)
Selecting the correct driving mode and requesting torque accordingly (ASIL C)
Monitoring the correct behavior of the module (ASIL B)
Torque Request
Instrument Cluster (Transmission State and warnings)
Driver

Figure 1 Functional safety architecture

Attributes of the Safety Goals

Fill the table with the safety goal descriptions

SG 1	The torque to be requested shall not exceed the one intended by the driver
SG 2	No negative torque is requested while the driver does not intend it.
SG3	The vehicle will not move in the opposite direction of what intended by the transmission mode.

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
SG1	C	Preventing unintended acceleration	200ms	The driver is warned about the malfunction	Shifting to neutral gear.
SG2	B	Preventing unintended deceleration	300ms	The driver is warned about the malfunction	Shifting to neutral gear.

SG3	A	Preventing requesting regenerative braking at velocity lower than 0.5 Km/h, regarding mode B	500ms	The driver will be warned about the malfunction	Shifting to neutral gear.

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	SR1: A voter devised to check the consistency of three analogue signals from the throttle pedal	The difference of each two signals must not exceed a certain bound.	No	No
		SR2: Send a warning in case one sensor sends a signal that is inconsistent	The driver shall be notified regarding the fault.	An alarm sound will be played, or a small vibration in the steering wheel.	No
		SR 3: In case of detecting an outlier sensor, faulty sensor will not be considered.	The voter takes the average of consistent signals.	No	No

		SR 4: In case of inconsistency between all the sensor, the transmission state changes to neutral (zero torque is requested)		No	No
	SG2	SR1: A voter devised to check the consistency of three analogue signals from the throttle pedal	The difference of each two signals shall not exceed a certain bound.	No	No
		SR2: Send a warning in case one sensor sends a signal that is inconsistent	The driver shall be notified regarding the fault.	An alarm sound will be played, or a small vibration in the steering wheel.	No
		SR 3: In case of detecting an outlier sensor, faulty sensor will not be considered.	The voter take the average of consistent signals.	No	No
		SR 4: In case of inconsistency between all the sensor, the transmission state changes to neutral (zero torque is requested)		No	No
	SG3	SR 5: in case mode B, the velocity is lower than 0.5 Km/h, no torque shall be requested.		No	No
		SR 6: When the vehicle stops in mode B, no torque shall be requested until throttle pedal passes 1/3 of its course.	No negative torque shall be requested when the vehicle is stopped.	No	No

		SR7: In case of velocity lower than 0.5 Km/h if a negative torque is requested the driver will be warned.	No	No	No

ASIL preliminary architecture[1]

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

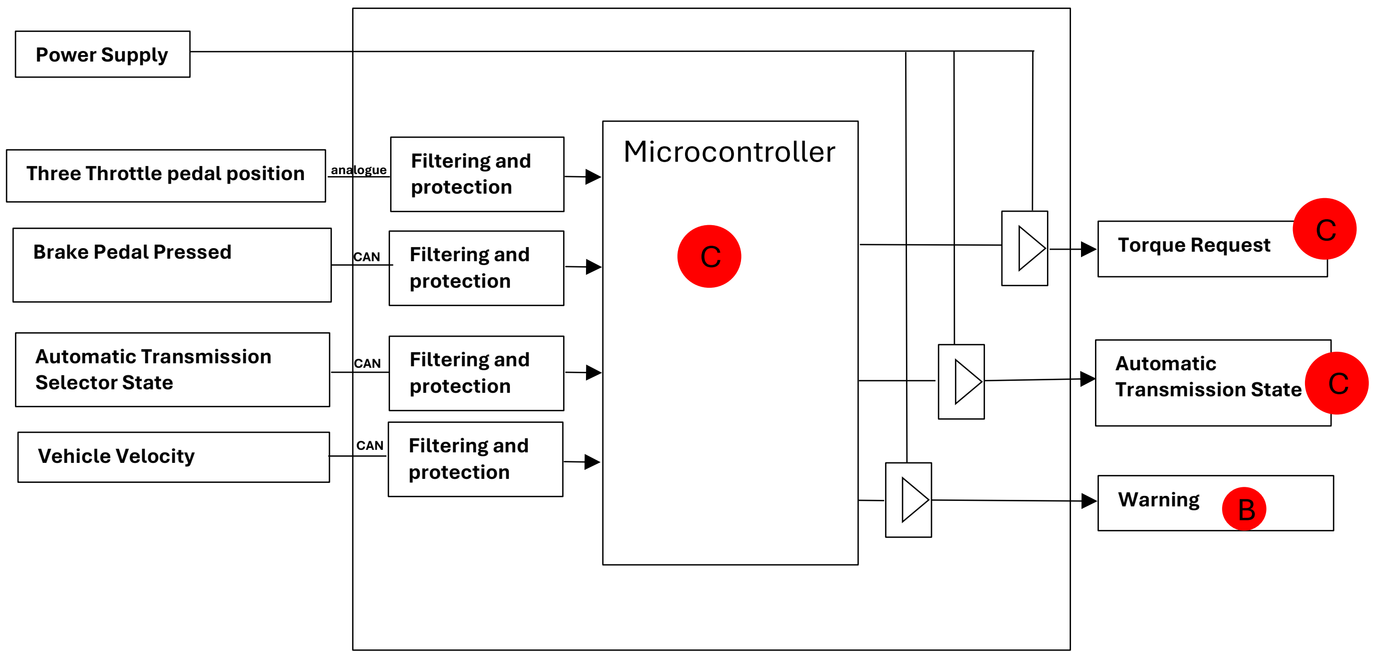


Figure 2 Preliminary architecture (without ASIL decomposition).

Implementations[2]

Functional redundancies

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

In terms of hardware implementation:
two additional sensors are used in order to determine the throttle pedal position, since the safety goal corresponding to the throttle pedal position is of ASIL C. In addition, a warning LED on the instrument cluster regarding this error.

[1] See document `02-iso26262.pdf,` slides 89, 90, 91, 92, 93.
[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.

In terms of software implementation:
Three signals regarding the throttle pedal position are fed to a voter which works as follows:

- In case of receiving consistent signals with a threshold of 0.025, the output of the voter is a signal that is the average of all the sensors.
- In case of one outlier, the signal from the outlier is not considered, and the output would be the average of functioning sensors. It set the ***warning*** flag to 1.
- In case all the input signals have difference larger than 0.025 two by two, both ***fault*** and ***warning*** flag set to 1.

Pay attention that, in the normal condition, the largest difference of the sensors in this case is allowed to be than 5 percent.

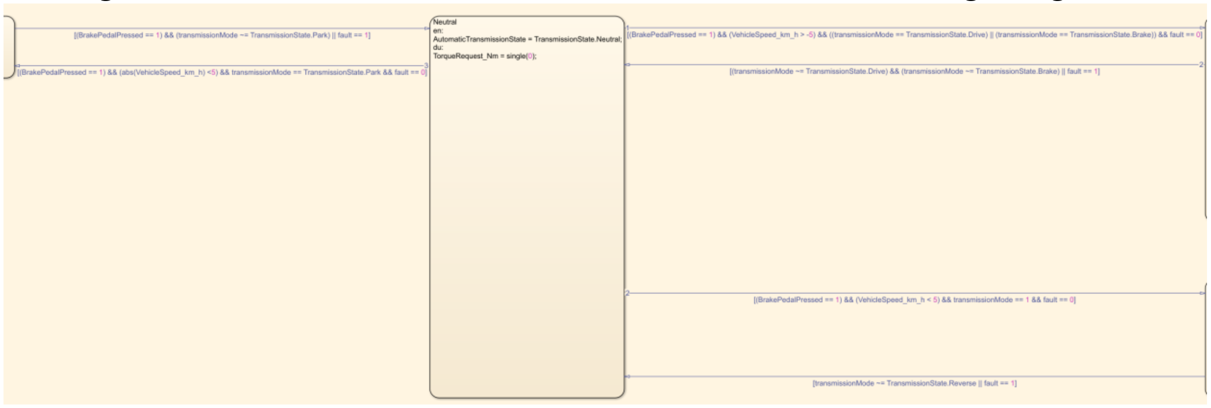
When the ***warning*** flag is set to 1, the system degrade to Neutral transmission mode where no torque would be requested, preventing unintended acceleration and deceleration.

In order to prevent the vehicle to move in the reverse direction, in **transmission state B,** if a negative torque is requested while, the velocity of the vehicle is lower than 0.5Km/h, the *fault* would be set to 1.

In addition, in the logic of the controller, if the *fault* flag becomes 1, the *AutomaticTransmissionState* changes its state to Neutral.

Plausibility check is implemented for all the entry signals to make sure they all assume values in the allowable range.

At one glance, other than additional conditions inside the controller regarding the *fault*.



```
Neutral
en:
AutomaticTransmissionState = TransmissionState.Neutral;
du:
TorqueRequest_Nm = single(0);
```

```
[(BrakePedalPressed == 1) && (transmissionMode ~= TransmissionState.Park) || fault == 1]
[(BrakePedalPressed == 1) && (abs(VehicleSpeed_km_h) <5) && transmissionMode == TransmissionState.Park && fault == 0]
[(BrakePedalPressed == 1) && (VehicleSpeed_km_h > -5) && ((transmissionMode == TransmissionState.Drive) || (transmissionMode == TransmissionState.Brake)) && fault == 0]
[(transmissionMode ~= TransmissionState.Drive) && (transmissionMode ~= TransmissionState.Brake) || fault == 1]
[(BrakePedalPressed == 1) && (VehicleSpeed_km_h < 5) && transmissionMode == 1 && fault == 0]
[transmissionMode ~= TransmissionState.Reverse || fault == 1]
```

Figure 3 A part of the controller chart including the additional conditions regarding fault signals.

Implemented plausibility checks

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

A plausibility check is done to make sure the input signals are of the consistent data type and range mentioned in the **Controller SW Unit specifications** section of lab 02 report.

Therefore, it is checked whether:

- the *BrakePedalPressed* is 0 or 1.
- *AutomaticTransmissionSelectorState is either 0,1,2,3, or 4*
- *VehicleSpeed_km_h has a value between -60 and 240*
- TorqueRequest has a value between -40 and 80.
- TorqueRequest shall be negative only for transmission mode 1 and 4 (preventing unintended deceleration).
- TorqueRequest shall not be positive for modes other than the modes 3 and 4

If one of the following cases is different from what is expected, the *fault* and *warning* flags shall set to value 1.

Software testing

Implemented unit tests

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

For all the units, Simulink Coverage is used to check whether the test adopted results in 100% coverage in different matrices.

BrakePlausibility unit is tested with a vector sequence of [-0.5,0,-0.5,1,1.5]. It was observed that the functionality adheres to what expected, that is *warning* and *fault* flags were set to 1 for input values other than 0 or 1.

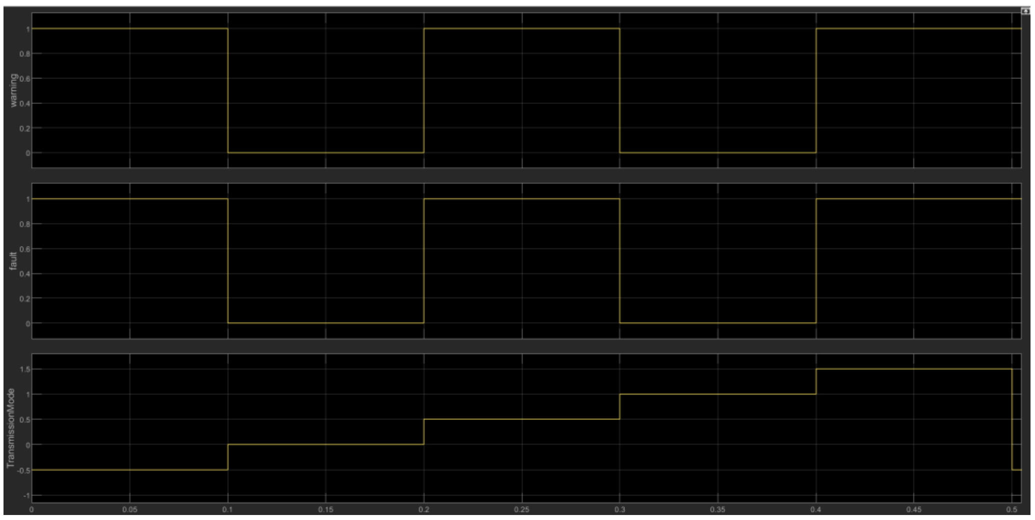
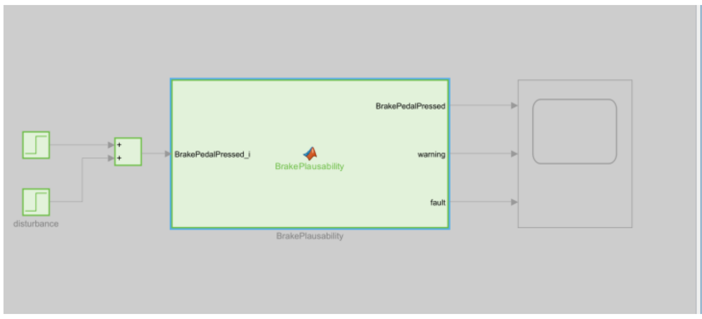
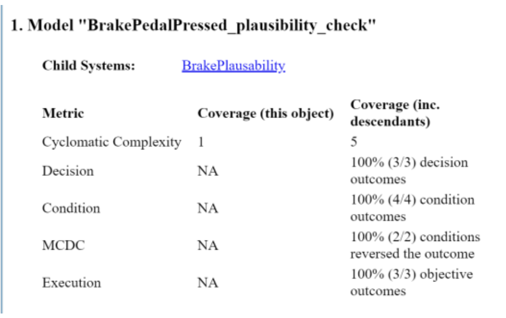


Figure 4 functionality test of plausibility for the BrakePedalPressed signal.




1. Model "BrakePedalPressed_plausibility_check"

Child Systems: BrakePlausability

Metric	**Coverage (this object)**	**Coverage (inc. descendants)**
Cyclomatic Complexity	1	5
Decision	NA	100% (3/3) decision outcomes
Condition	NA	100% (4/4) condition outcomes
MCDC	NA	100% (2/2) conditions reversed the outcome
Execution	NA	100% (3/3) objective outcomes

Figure 5 Coverage of Plausability unit for the brake signal.

For TransmissionMode plausibility, the unit was tested with a *Repreating Sequece Stairs* block, with values -0.5, 1, 1.5, 2, 2.5, 3, 3.5, 4, 4.5, 5. It is expected that the values other than 0, 1, 2, 3, 4, raise *warning* and *fault* flags.

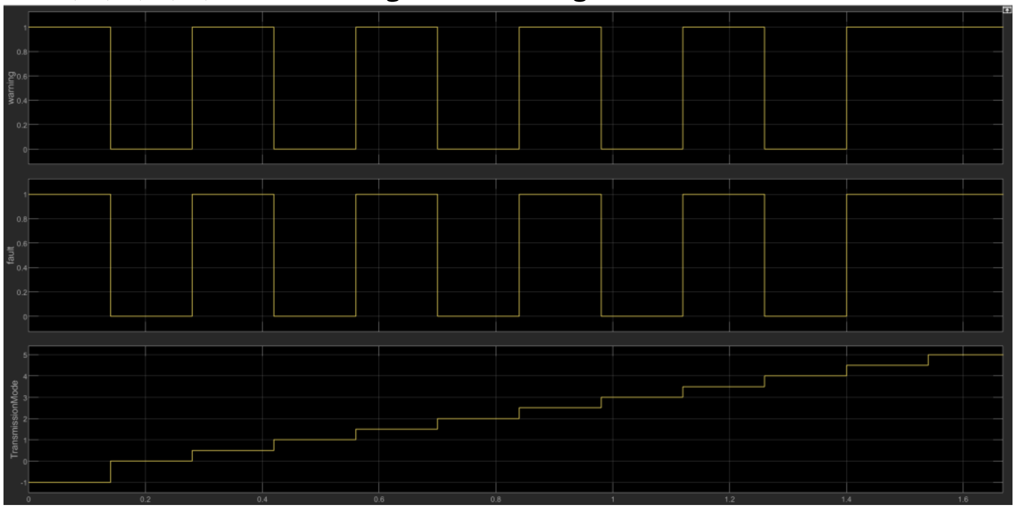
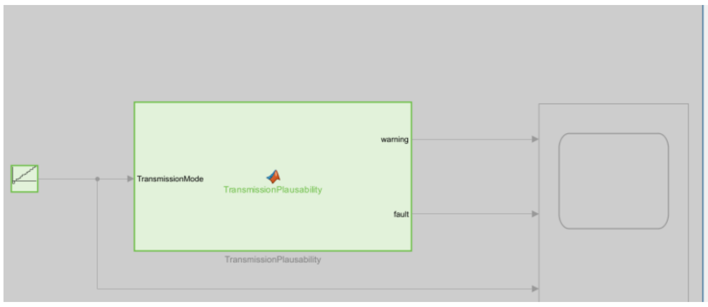
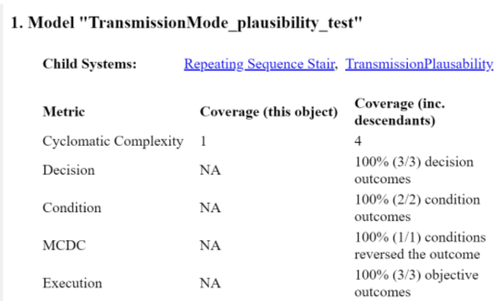


Figure 6 functionality test of plausibility for the TransmissionMode signal.




1. Model "TransmissionMode_plausibility_test"

Child Systems: Repeating Sequence Stair, TransmissionPlausability

Metric	**Coverage (this object)**	**Coverage (inc. descendants)**
Cyclomatic Complexity	1	4
Decision	NA	100% (3/3) decision outcomes
Condition	NA	100% (2/2) condition outcomes
MCDC	NA	100% (1/1) conditions reversed the outcome
Execution	NA	100% (3/3) objective outcomes

Figure 7 Coverage of Plausability unit for the TransmissionMode signal.

For the case of velictyPlausability, in case the value of velocity are less than -60 or higher than 240, the fault and warning flags are set to 1. For testing, a test vector [-80 -60 -30 0 200 240 280]' was fed to the function.

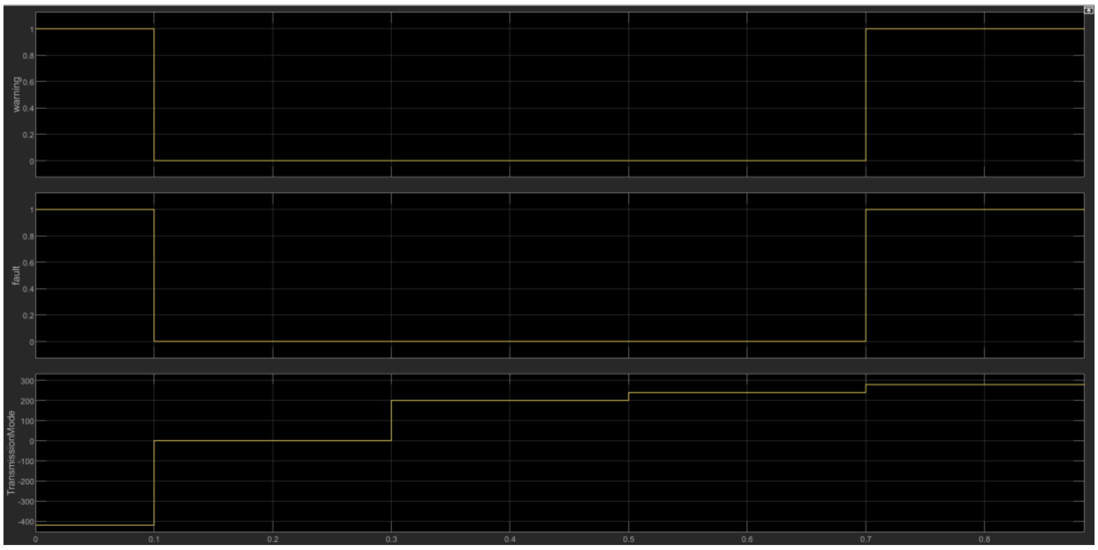
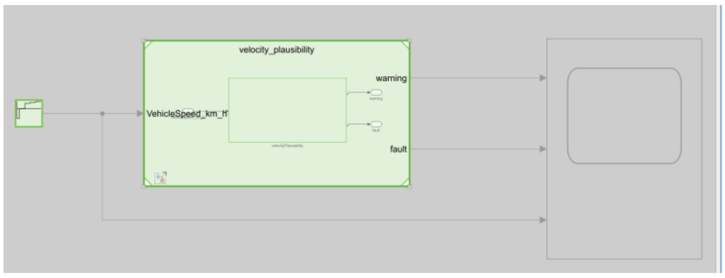


Figure 8 functionality test of plausibility for the Vehicle Velocity signal.



Child Systems: velocityPlausability

Metric	**Coverage (this object)**	**Coverage (inc. descendants)**
Cyclomatic Complexity	1	5
Decision	NA	100% (3/3) decision outcomes
Condition	NA	100% (4/4) condition outcomes
MCDC	NA	100% (2/2) conditions reversed the outcome
Execution	NA	NA

Figure 9 Coverage of Plausability unit for the Vehicle Velocity signal.

For the case of torquePlausability, the TorqueRequest should be checked to be inside the range and correspondent to the AutomaticTranmissionState. As the torque vector a sequence of [-60,-40,-20,-20,-20,-20,0,40,40,60,60,60,80,80,120]' and the corresponding transmission modes were [1 1 0 2 3 4 2 2 0 3 3 0 1 2 4 4]' was chosen as AutomaticTransmissionMode. It can be seen that the behavior is as described in the section *Implemented plausibility checks*.

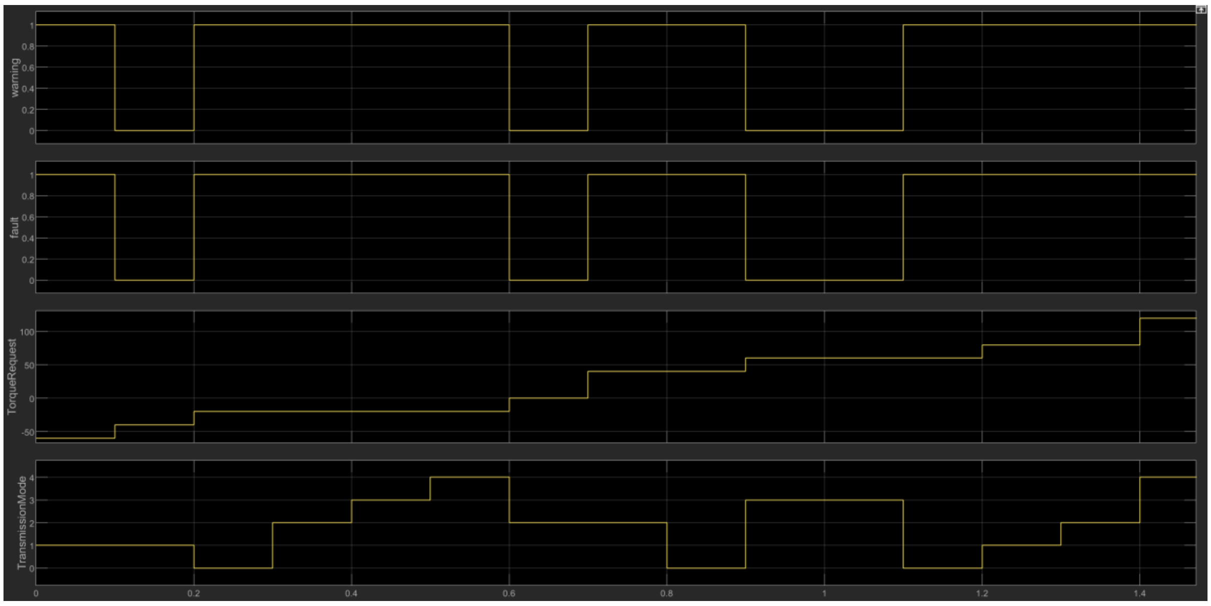
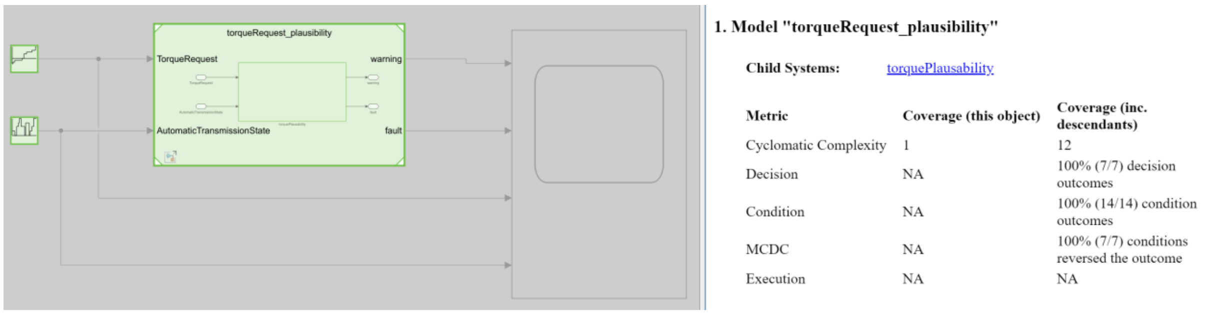


Figure 10 functionality test of plausibility for the TorqueRequest signal.



1. Model "torqueRequest_plausibility"

Child Systems: torquePlausability

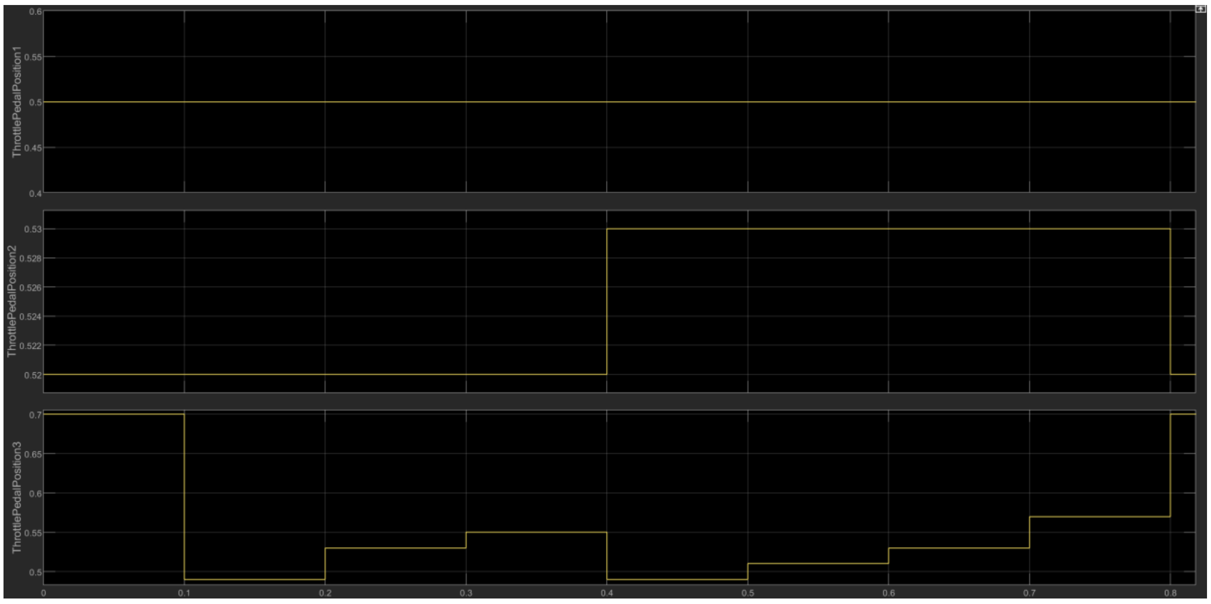
Metric	**Coverage (this object)**	**Coverage (inc. descendants)**
Cyclomatic Complexity	1	12
Decision	NA	100% (7/7) decision outcomes
Condition	NA	100% (14/14) condition outcomes
MCDC	NA	100% (7/7) conditions reversed the outcome
Execution	NA	NA

Figure 11 Coverage of Plausability unit for the TorqueRequest signal.

For the case of the voter, since there are 3 inputs, and the logic is more complicated, the following test matrix is used for testing:

Table 1 Test Vectors

Throttle 1	Throttle 2	Throttle 3	Throttle	Warning	Fault
0.5	0.52	0.7	0.51	1	0
0.5	0.52	0.49	0.505	0	0
0.5	0.52	0.53	0.51	0	0
0.5	0.52	0.55	0.51	1	0
0.5	0.53	0.49	0.495	1	0
0.5	0.53	0.51	0.52	1	0
0.5	0.53	0.54	0.535	1	0
0.5	0.53	0.56	0	1	1



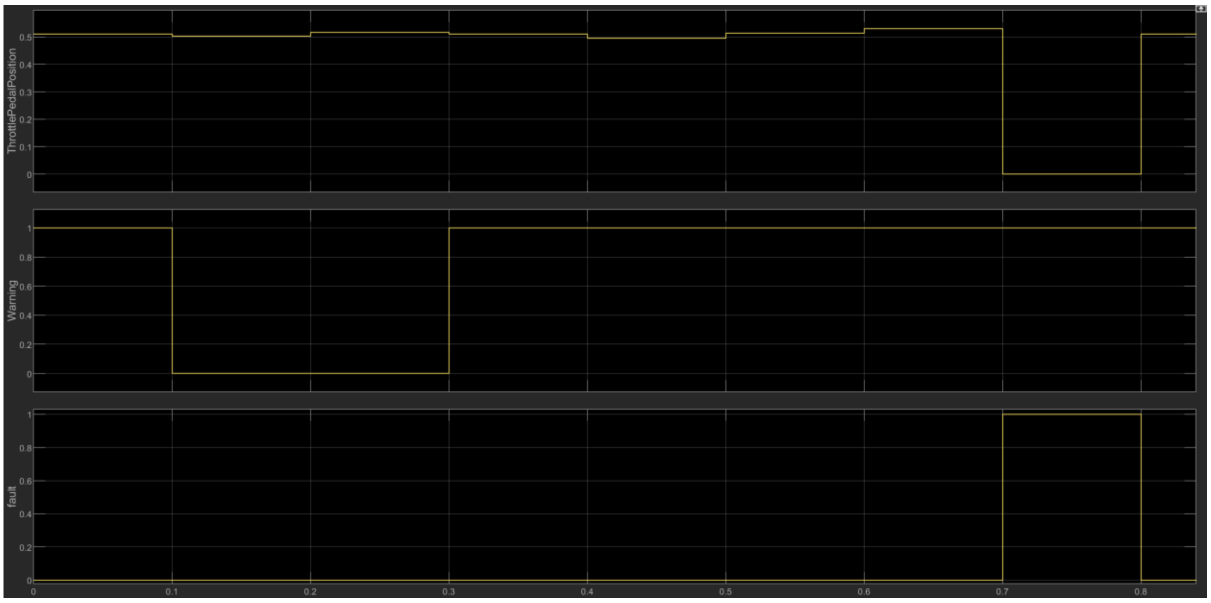
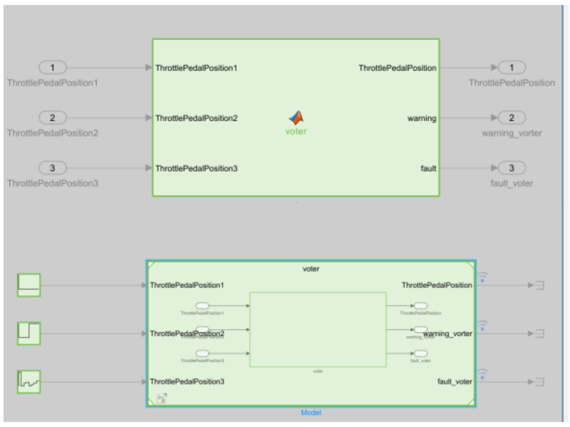


Figure 12 The result of the test which is concurrent with the test vectors.



Child Systems: voter

Metric	Coverage (this object)	Coverage (inc. descendants)
Cyclomatic Complexity	1	9
Decision	NA	100% (13/13) decision outcomes
Condition	NA	NA
MCDC	NA	NA
Execution	NA	NA

1. Model "voter_test"

Child Systems: Repeating Sequence Stair, Repeating Sequence Stair1, Repeating Sequence Stair2

Metric	Coverage (this object)	Coverage (inc. descendants)
Decision	NA	NA
Condition	NA	NA
MCDC	NA	NA
Execution	NA	100% (10/10) objective outcomes

Figure 13 Coverage voter.

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.

Testing in the integration level was performed to make sure whether the whole system have the expected functionality. Now that units have correct functionality, the units can be integrated and tested. If there is a problem, it must be related to the integration and not the units. Fault injection is the manner used for testing the functionality of the units in integrations.

In order to check the functionality of the voter, at second 2, a disturbance is added to the sensor 3, it is expected that the result would be a warning flag. Then at the second 6, another disturbance is added to sensor 1. In this case, we expect that the fault flag is set to 1 and the Transmission mode shift to Neutral. Then at second 10, the disturbance of the first sensor is removed, so we expect that the system works with two functioning sensors, and then at second 20 also the disturbance of the first signal is removed. In this case, also the warning flag should be set to 1. As can be seen in the following figures, both warning signal and fault signals changes as expected, and while fault signal is 1, the transmission shifts to neutral.

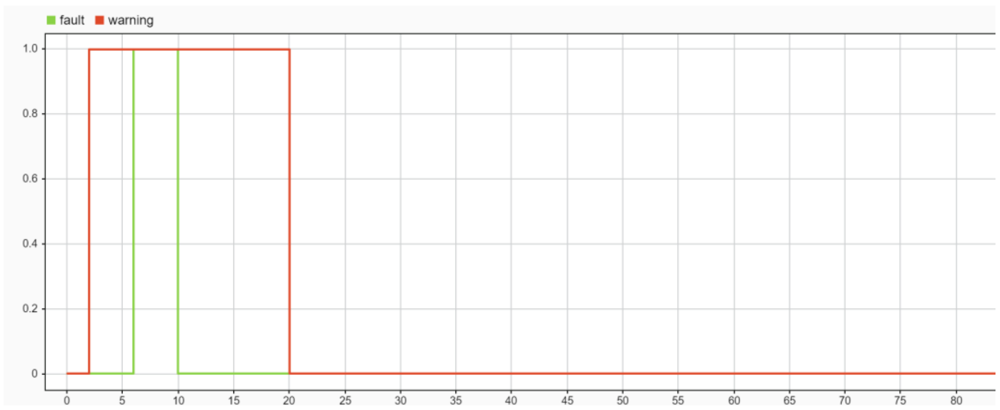


Figure 14 testing the functionality of the controller; fault and warning signals.

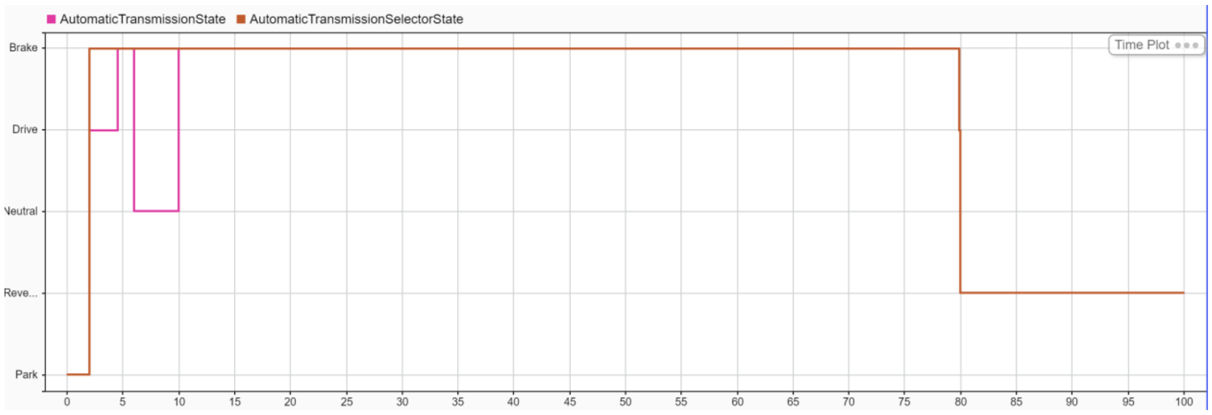


Figure 15 testing the functionality of the controller; transmission mode. It can be seen that at second 6, when the fault signals becomes 1, the transmission mode goes to neutral.

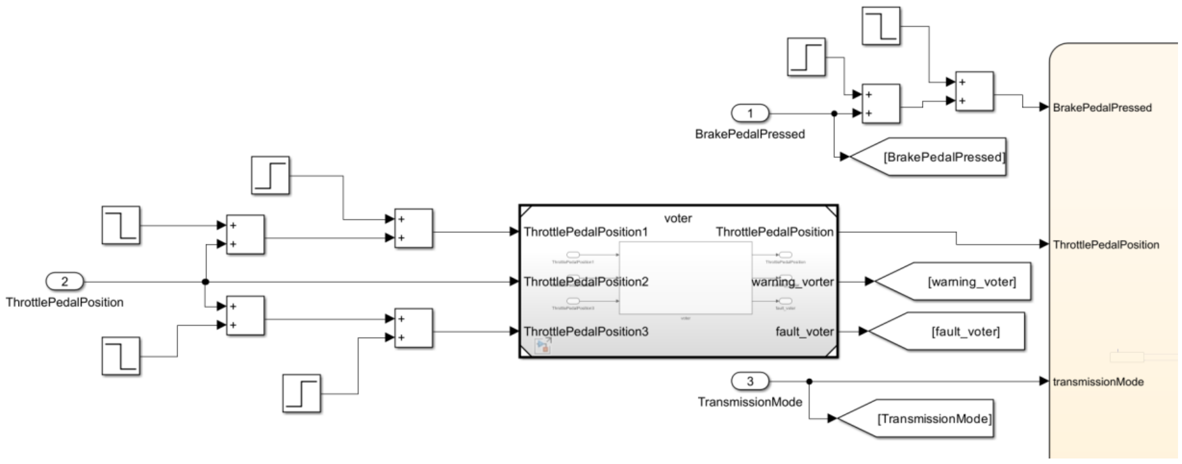


Figure 16 testing the functionality of the controller; Simulink scheme of corrupting the sensors number 1 and 3. After the fault is removes, the brake pedal is pressed in order to change to the wanted transmission mode. It is done for the sake of simulation, since we want to check if the signals become correct again, the controller functions as it is expected.

Further, Simulink coverage tool, was used in order to check whether all the blocks are executed during the simulation. Having done so, a 100% block coverage was obtained. Therefore, we are sure that during the test, all the blocks are involved.

For testing TorquePlausability, at second 10, where we get the maximum allowable torque, a value of 20 N.m is added to the value of the torque for 5 seconds, and as it can be seen, the fault flag rises. Therefore, this block is connected properly.

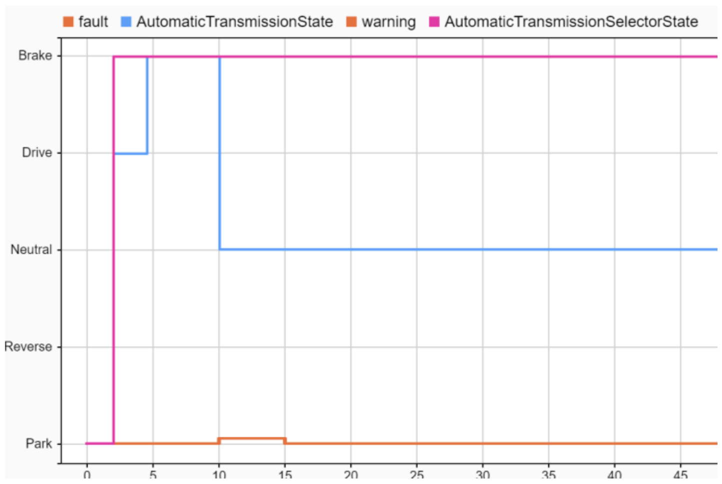


Figure 17 Test of torque plausibility

After that, a fault is injected into the velocity signal to check that it is well integrated with the controller. At second 10, a value of 200 is added to the value of the speed to make it more than 240. It can be seen that the fault flag spikes to 1 at that time, and consequently, the transmission mode transits to Neutral mode.

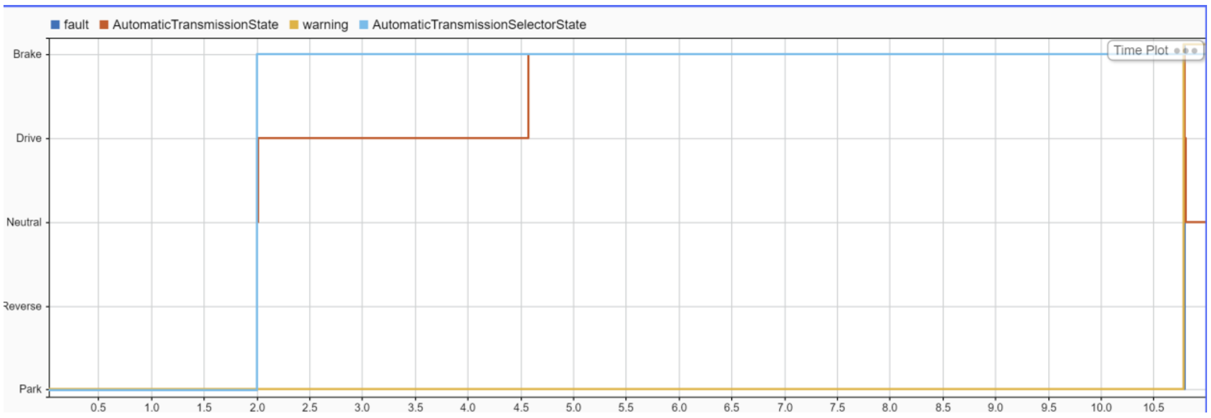
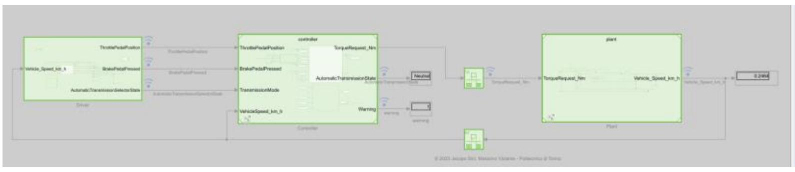
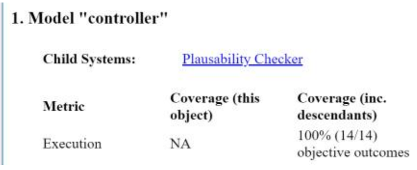


Figure 18 Test of velocity plausibility

For testing the checker for BrakePedalPressed signal, since the expected input value of the checker is set to Boolean, any value other than a Boolean value rises an error in Simulink. The same for the TranmissionMode selected by the driver, the function for checking the correctness of the signal expects to receive a signal with enum TranmissionState values, and for any signal other than that, the Simulink does not run the program.

All in all, by separately checking the functionality of the blocks in integration with the controller and injecting faults to each unit, we are sure that the units are correctly integrated, and the resultant system has the expected functionality required. That is, the safety requirements corresponding to unintended acceleration and decelerations are implemented correctly.

1. Model "controller"

Child Systems:	Plausability Checker	
Metric	**Coverage (this object)**	**Coverage (inc. descendants)**
Execution	NA	100% (14/14) objective outcomes

Figure 19 Block Coverage of the controller.

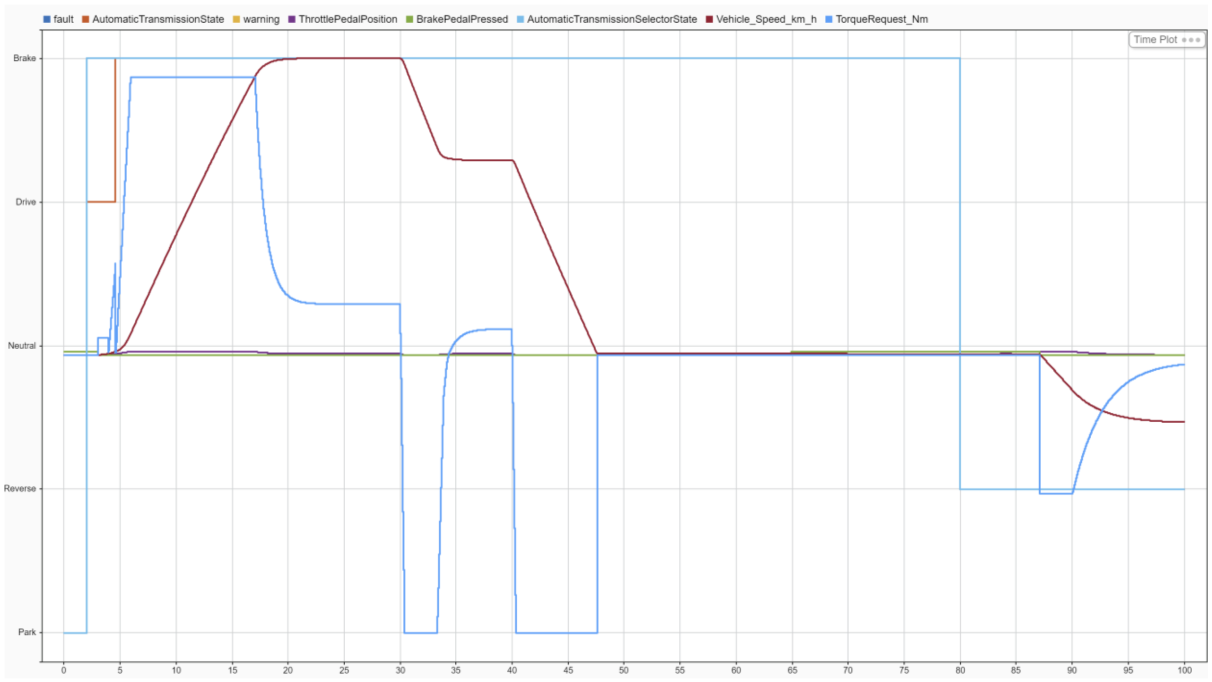


Figure 20 The result of the simulation without any injected fault.