

Name: Luca
Surname: Ghio
Student ID: 211255

Model-based software design assignment: CROSS-TRAFFIC ASSIST

Item definition

- **Item:** Cross-Traffic Assist (CTA) which informs the driver about hidden obstacles on each side of the vehicle during backward maneuvers
- **Elements of the item:**
 - rear left radar
 - rear right radar
 - microcontroller
 - comparator unit
 - enable/disable unit
 - CAN interface

Interactions of the item with other items

vehicle CAN network for reading current gear and transmitting warning left and warning right lamp on/off signals

Identification of functionalities

- **Provided functionalities:**
 - rear left radar: detects distances of vehicles on the left side which are closer than a certain threshold
 - rear right radar: detects distances of vehicles on the right side which are closer than a certain threshold
 - comparator unit: compares distances over time to determine whether the vehicle is approaching
 - enable/disable unit: automatically switches the item on when the reverse gear is engaged, and switches it off when the reverse gear is released
- **Functionalities provided to other items:**
 - issuing warning left lamp on/off signals via the CAN network
 - issuing warning right lamp on/off signals via the CAN network
- **Functionalities required from other items:**
 - the current gear shall be provided via the CAN network

Hazard analysis

- **Situational analysis:**
 - OS1: current gear is reverse, no approaching vehicle
 - OS2: current gear is reverse, vehicle approaching at low speed, no covered view
 - OS3: current gear is reverse, vehicle approaching at middle speed, no covered view
 - OS4: current gear is reverse, vehicle approaching at high speed, no covered view
 - OS5: current gear is reverse, vehicle approaching at low speed, covered view
 - OS6: current gear is reverse, vehicle approaching at middle speed, covered view
 - OS7: current gear is reverse, vehicle approaching at high speed, covered view
- **Hazard identification:**
 - H1: the item is not able to issue warning lamp on signal when there are approaching vehicles
 - radar failure: missed detection of vehicles closer than the threshold
 - radar failure: the radar provides increasing distance values for approaching vehicles
 - comparison unit failure: missed detection of approaching vehicles
 - H2: the item is not able to issue warning lamp off signal when there are no approaching vehicles
 - radar failure: unintended detection of vehicles closer than the threshold
 - radar failure: the radar provides decreasing distance values for departing vehicles
 - comparison unit failure: unintended detection of approaching vehicles
 - H3: the item is not able to issue warning lamp signals
 - enable/disable unit failure: missed switch on of the item
 - enable/disable unit failure: unintended switch off of the item
 - CAN interface failure: send/receive error

• **Risk assessment:**

	H1	H2	H3	notes
OS1	S=0 E=3 C=0 QM	S=0 E=3 C=0 QM	S=0 E=3 C=0 QM	No crash can happen since there is no approaching car.
OS2	S=1 E=3 C=1 QM	S=1 E=3 C=1 QM	S=1 E=3 C=1 QM	This is simply controllable for the driver by looking outside through the side window to check whether a car is approaching.
OS3	S=2 E=3 C=1 QM	S=2 E=3 C=1 QM	S=2 E=3 C=1 QM	
OS4	S=3 E=2 C=2 ASIL A	S=3 E=2 C=2 ASIL A	S=3 E=2 C=2 ASIL A	
OS5	S=1 E=3 C=3 ASIL A	S=1 E=3 C=3 ASIL A	S=1 E=3 C=3 ASIL A	The driver can not look outside through the side window because the view is covered by another car parked next to him, a wall, etc.
OS6	S=2 E=3 C=3 ASIL B	S=2 E=3 C=3 ASIL B	S=2 E=3 C=3 ASIL B	
OS7	S=3 E=2 C=3 ASIL B	S=3 E=2 C=3 ASIL B	S=3 E=2 C=3 ASIL B	
notes	The lamp is always off → it is useless for the driver.	The lamp is always on → it is useless for the driver.	The item can't communicate at all with other items.	

Safety goals and functional safety concept

- H1/H2 → SG1: the item shall be able to detect whether the radars are operating correctly and, if not so, the driver shall be informed
 - FSC1: two¹ redundant radars per side, and if distance values are different the corresponding lamp flashes
- H1/H2 → SG2: the item shall be able to detect whether the comparison unit is operating correctly and, if not so, the driver shall be informed
 - FSC2: redundant code in the comparator unit, and if outputs are different the lamps flash
- H3 → SG3: the driver shall be informed whether the item is on and, if not so, he shall be able to manually force the item to switch on
 - FSC3: when the item is operating the lamps are always on (the light is green when there are no approaching vehicles, or red when there are approaching vehicles), otherwise the lamps are off
 - FSC4: a button can be pressed by the driver to manually force the item to switch on
- H3 → SG4: the driver shall be informed whether the item is connected to the CAN network
 - FSC5: ping-alive protocol should be implemented on the CAN bus, and if no answer received the lamps flash

Software architecture:

The software architecture was described by means of an AUTOSAR diagram (page 3).

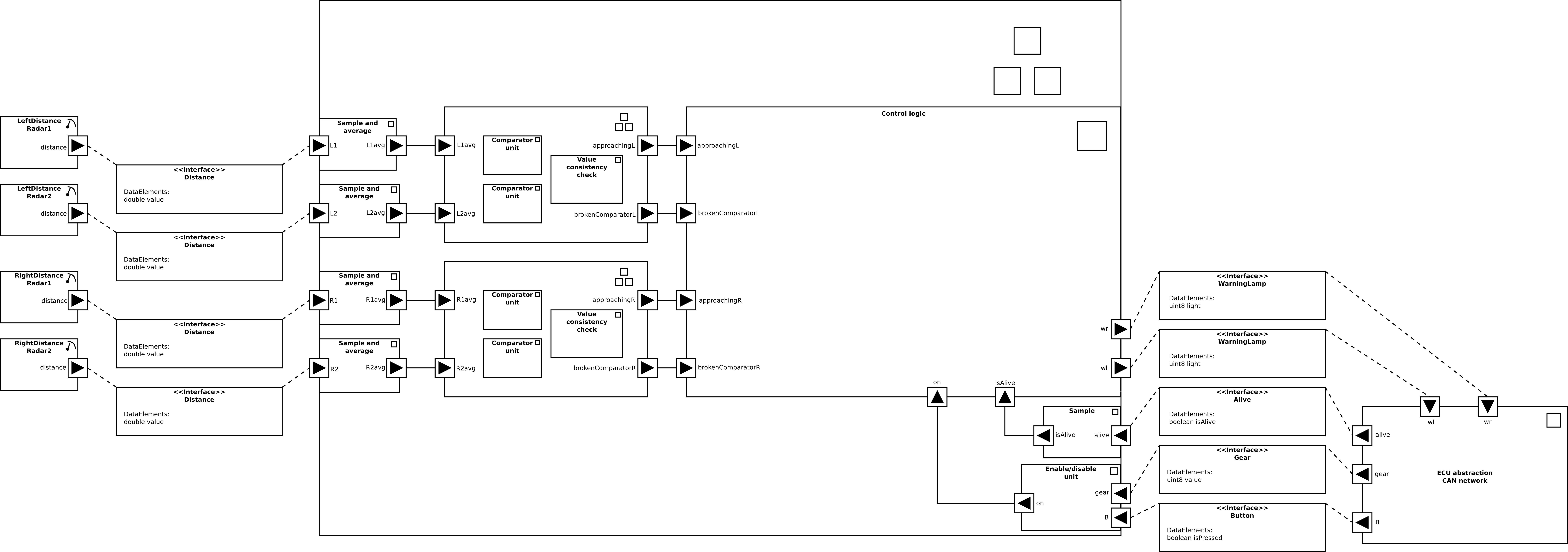
Each radar provide the microcontroller, through the “Distance” interface, with distance values, then filtered by a sample and average component. Each comparator unit determines whether the vehicle is approaching. The radar and the comparator unit are redundant, so the inputs from the former and the outputs from the latter are checked for consistency.

The enable/disable unit receives from the CAN network the gear value through the “Gear” interface and the button state through the “Button” interface, and determines whether the item should be switched on.

The ping-alive protocol consists in periodic alive messages coming from the CAN network through the “Alive” interface.

Information about warning lamp lights are sent by the item through the “WarningLamp” interface to the CAN network.

¹ no forward recovery as a tradeoff between cost and safety (since the ASIL is not so high).



Software implementation

The controller code was automatically generated by a Simulink model, having a fixed-step size equal to 0.1 seconds. For the sake of simplicity, just the left side of the vehicle was considered in the model. A testbench was also provided.

The controller block (page 5) is characterized by the following inputs:

- `L1` (double): the distance value received from the first distance radar;
- `L2` (double): the distance value received from the second distance radar;
- `gear` (uint8): the gear value (value 0 = reverse gear, value 6 = neutral gear);
- `B` (boolean): specifies whether the button is pressed;
- `alive` (boolean): specifies whether an alive message has been received.

and by the following output:

- `wl` (uint8): specifies whether the light of the warning lamp should be off (value 0), red (value 1) or green (value 2).

In order to remove the noise from distance inputs, each sample and average block computes the average of the last 5 distance samples. In addition, it compares the current sample with the previous 4 samples: if the same distance value has been received in the last 5 instances of time, the radars are considered as broken.

Distance values higher than 200 m are ignored. The average distance values are compared to check whether they differ beyond a threshold equal to 0.5 m, meaning that the radars are considered as broken.

Each comparator unit compares the current average distance sample to the previous one in order to determine whether the vehicle is approaching. The outputs from the comparator units are compared: if they are different, the comparator units are considered as broken.

The enable/disable unit block switches the item on when the gear is reverse or the button is pressed.

Each alive message is delayed 4 times and, if in the last 5 instants of time no alive message is received, the item is considered as not being connected to the CAN network.

The control logic, modeled by means of a stateflow (page 6), is characterized by the following inputs:

- `brokenComparator` (boolean): specifies whether the comparator units are broken;
- `approaching` (boolean): specifies whether a vehicle is approaching;
- `brokenSensor` (boolean): specifies whether the distance radars are broken;
- `on` (boolean): specifies whether the item should be switched on;
- `isAlive` (boolean): specifies whether the item is not connected to the CAN network;

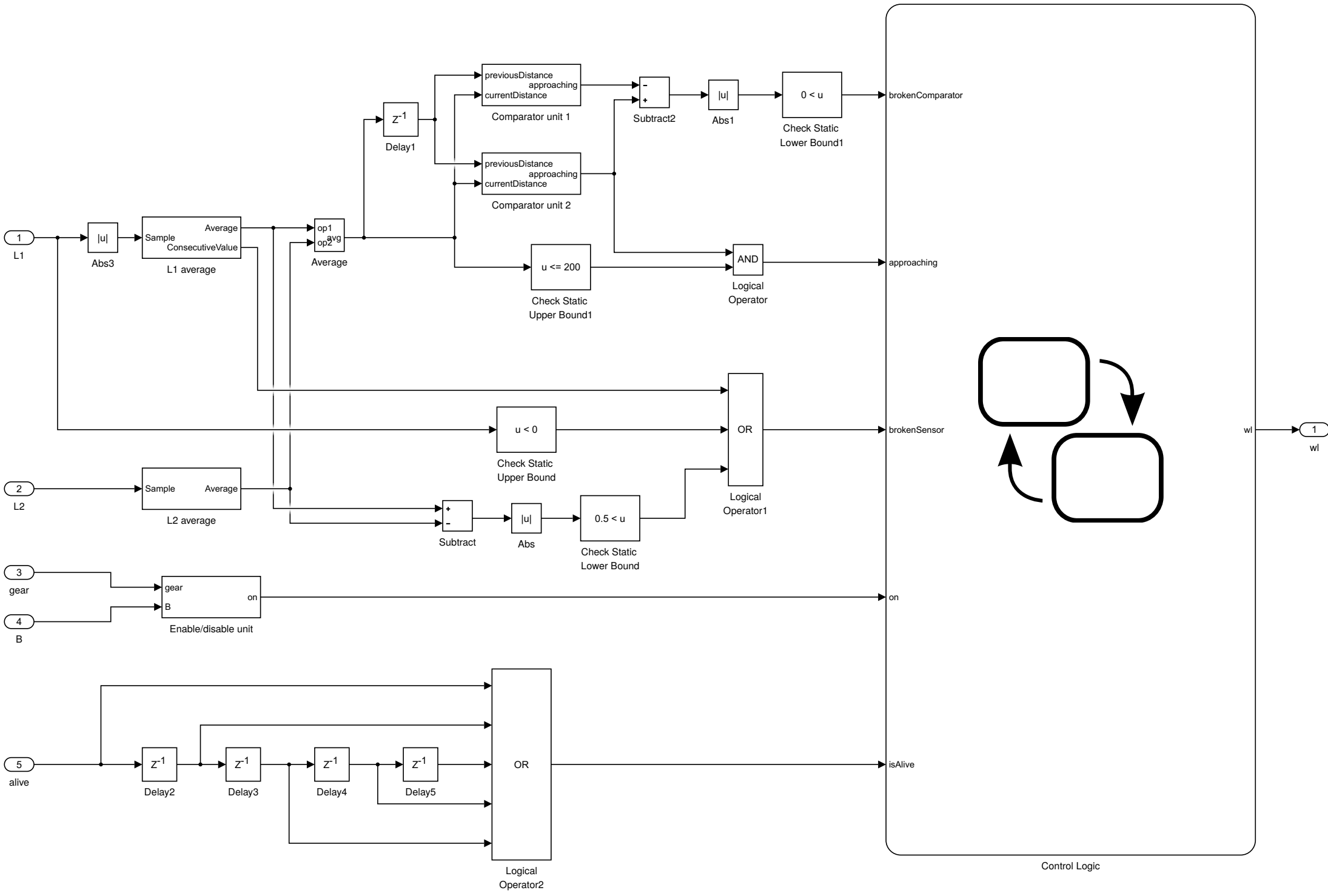
and by the following main states:

- `NOT_APPROACHING_STATE`: no vehicle is approaching → the green lamp is still;
- `APPROACHING_STATE`: a vehicle is approaching → the red lamp is still;
- `BROKEN_COMPARATOR_STATE`: the comparator units are broken → the red lamp flashes;
- `BROKEN_SENSOR_STATE`: the distance radars are broken → the red lamp flashes;
- `OFF_STATE`: the item is switched off → the lamp is off;
- `NOT_ALIVE_STATE`: the item is not connected to the CAN network → the red lamp flashes.

The auto-generated code consists of the following files: `controller.c`, `controller.h`, `rtwtypes.h`.

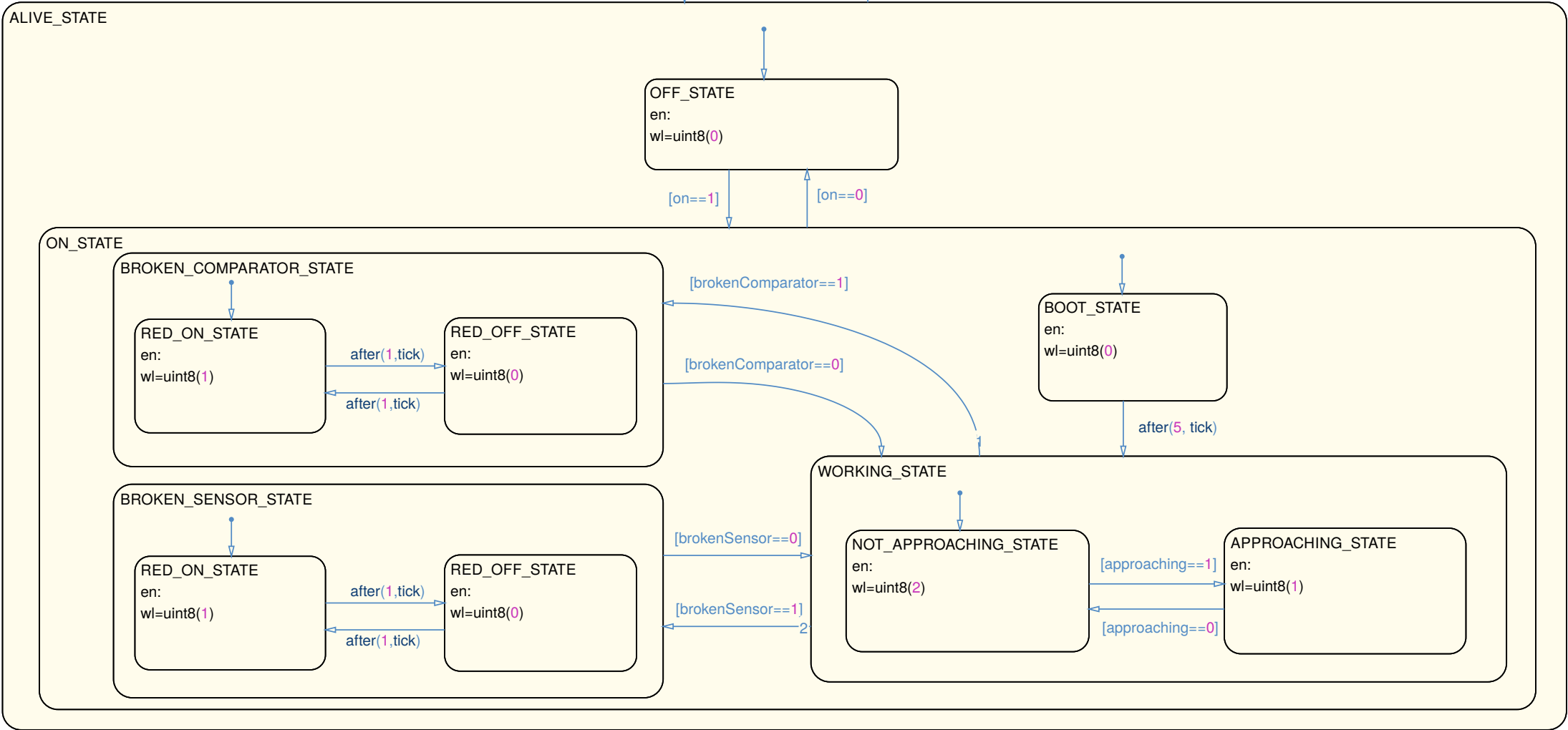
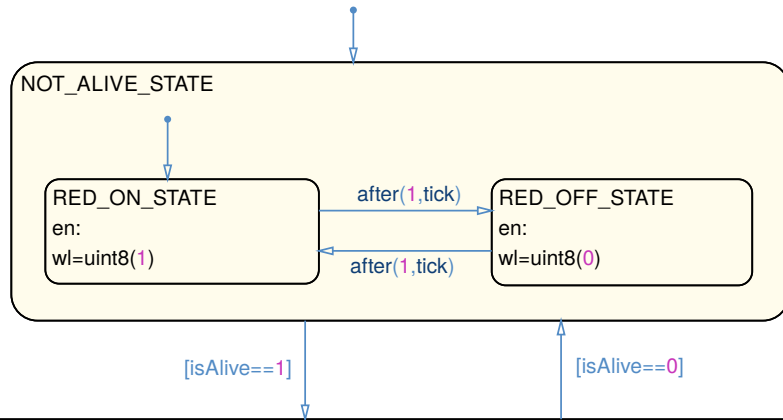
The following functions are automatically generated:

- `controller_initialize()`: initializes the controller;
- `controller_step()`: is called on every controller step.



Control Logic

wl



Hardware/software integration

Some C++ code in mbed was written in order to integrate the model with the evaluation board.

- **Inputs:**
 - D4: the `InterruptIn` interface of the board used to receive echo signals from the sensor;
 - `USBRX`: the `Serial` interface of the board simulating the input CAN interface, used to receive the gear value, the button state and alive messages.
- **Outputs:**
 - D2: the `DigitalOut` interface of the board used to send trigger signals to the sensor;
 - `LED_RED` and `LED_GREEN`: the `DigitalOut` interfaces of the board simulating the output CAN interface, used to send the warning lamp on/off signals.

Test plan

Boundary value analysis has been performed for the model inputs:

- L1: $x < 0$; $0 \leq x \leq 200$; $x > 200$
- L2: $x < 0$; $0 \leq x \leq 200$; $x > 200$
- gear: $x < 0$; $x = 0$; $x > 0$
- B: $x = \text{false}$; $x = \text{true}$
- alive: $x = \text{false}$; $x = \text{true}$