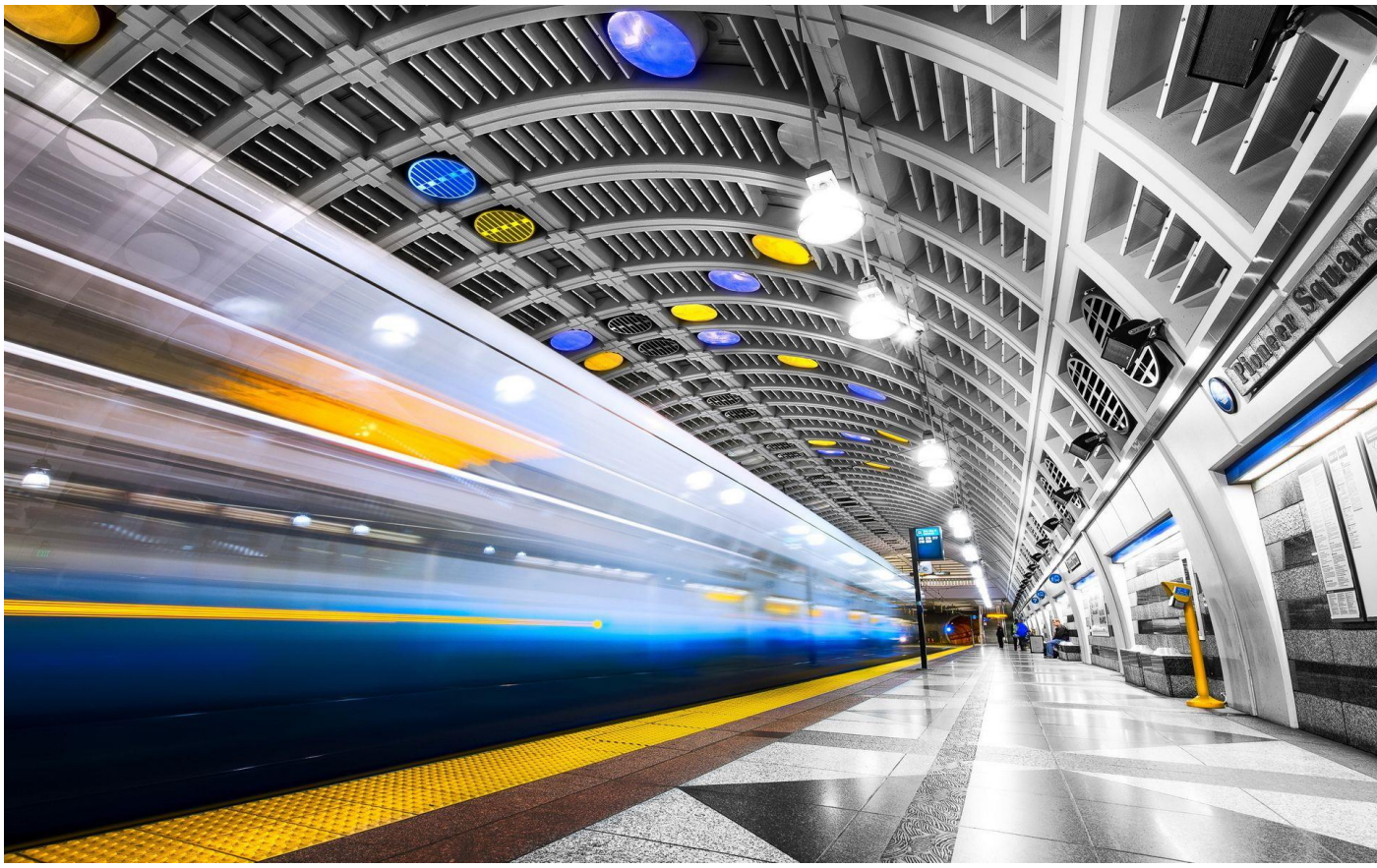


Metro Train Control

Real-time software development
Exam project 2020-21

Andrea Turconi - mat: 730225
Lorenzo Laino - mat: 731124

SYSTEM REQUIREMENTS

In this report, the simulation of a real time hardware, controlled by a microprocessor, will be presented. The scope of this following project is the simulation of a simplified metro train controller.

The relation is structured as follows: the first part explains all the system requirements provided from the specification; the second part present all the design through the use of UML diagrams; the last one shows all the test case provided and their results.

In order to work properly the controller must:

- Increase and decrease speed according to the driver's commands;
- Activate or deactivate regular brake commands issued by the driver;
- Respond to emergency brake request;
- Enforce stop commands by stop signals;
- Manage communications between from the train traffic managers to the driver;

The system is implemented using the µVision 4.60 ide and the microprocessor STM32F10x

1. THE LEVER

In order to handle the will of the driver, the controller is equipped with a lever, which allows the driver to select the desired speed or braking force.

The lever has only 7 position: positive ones indicate that an acceleration is issued by the driver, instead negative ones indicate that a braking force needs to be activated. Obviously, the lever can only move incrementally by one position and can't be on brake position and accelerate position at the same time.

The following table shows all the configuration and the position of the lever.

+3	Is the maximum acceleration
+2	Is the medium acceleration
+1	Is the minimum acceleration
0	Is the idle state: neither acceleration nor braking
-1	Is the minimum braking
-2	Is the medium braking
-3	Is the strongest braking

The positions of the control lever are communicated to the controller as inputs via pins of General-Purpose Input Output B (GPIOB). Pins from 2 to 8 are involved in order to handle the request of the lever.

As described above, the train has three motor acceleration that correspond to the three-positive position of the lever. Pins from 6 to 8 are associated with the positive acceleration.

Pin 5 is associated with the idle lever position: in this situation there is neither acceleration power nor braking force.

The remaining pins (from 2 to 4) are associated with the three-negative position of the lever. As for the acceleration force, also the braking force has three power levels.

In the following table is possible to understand the associations of the input pins with the corresponding action of the lever.

Input Pin of GPIOB	Description
8	+3 Maximum acceleration
7	+2 Medium acceleration
6	+1 Minimum acceleration
5	0 Idle (no acceleration, no braking)
4	-1 Minimum braking
3	-2 Medium braking
2	-3 Strong braking

2. STOP SIGNAL

The controller must handle an asynchronous and unpredictable signal that must stop the train gently. This input is called “Stop signal”: when this request is received, the motor acceleration must be set to zero and the brakes are activated at medium force. This request has greater priority than the command issued by the driver; thus, when the stop signal is active, the inputs provided by the driver, using the lever, are ignored. The train, once stopped because of this signal, must not restart until the stop signal is active. When the stop signal is cleared, the train can resume normal operation. However, to this end it is first necessary that the control lever is set in position zero: neither acceleration nor braking are activated. The stop signal is an input received on pin 1 of GPIOB

3. EMERGENCY BRAKE

Another asynchronous and unpredictable signal that the controller must handle is the “Emergency Brake”. When this request is received, the train must stop as soon as possible: motor acceleration is set to zero and brakes are activated at maximum force. This request has the greatest priority, so “Stop signal” request and lever input are ignored when an Emergency brake request is raised. The request must be satisfied immediately disregarding the position of the lever and the state of the train. The maximum braking force is still maintained during the duration of the Emergency request and are also keep active once the signal is cleared. In order to restart the system a manual reset is necessary. The emergency brake request is an input signal, received on pin 0 of GPIOB

4. TRAFFIC MESSAGES

During the journey of the train, the Traffic Management Center can send communications to the driver via a serial line. The traffic messages are received on the USART1 of the GPIOA. The management of these received messages has the lowest priority.

5. INPUT OF THE SYSTEM

As described in the previous paragraphs, all the inputs of the system, including the lever, the stop and the emergency signal, are received on pins of GPIOB. The following table shows the complete input schema of the system.

Input Pin on GPIOB	Description
8	+3 Maximum acceleration
7	+2 Medium acceleration
6	+1 Minimum acceleration
5	0 Idle (no acceleration, no braking)
4	-1 Minimum braking
3	-2 Medium braking
2	-3 Strong braking
1	Stop signal request
0	Emergency brake request

6. OUTPUT OF THE SYSTEM

The outputs are communicated to the system via pins of General-Purpose Input Output C. The desired outputs of the lever are handled by pins from 0 to 2 and 8 to 11. All the outputs are configured as push-pull.

As said on the previous chapter, there are three acceleration levels: in order to communicate with the motor and report the position of the lever, they are associated with the 0, 1, and 2 pins on the GPIOC.

The three braking force level are configured on the 8, 9 and 10 pins on the GPIOC and are used to report the position of the lever. In addition to these three braking forces there's another level that represent the maximum power of braking force. Pin 11 is the one associated with the maximum power level.

The idle state has no output pin on the GPIOC because of it must have no acceleration force and no braking force active while it's selected.

Output Pin on GPIOC	Description
2	+3 Maximum power
1	+2 Medium power
0	+1 Minimum power
8	-1 Minimum braking force
9	-2 Medium braking force
10	-3 Strong braking force
11	-4 Maximum braking force

The traffic messages that has been received on the USART1 of the GPIOA are transmitted and displayed on the USART2 of the GPIOA

7. CORRECTNESS CRITERIA

During the journey of the train, if the driver sets the lever on the maximum acceleration (+3) and maintain it for more than 4 consecutive seconds, it's responsibility of the controller to decrease the acceleration output and set it to +2. This security maneuver is done even though the lever in on position +3.

Another correctness criteria that the controller must manage is the possibility of a malfunction between the connection of the command lever. When the driver moves the lever from position A to position B, it is possible that temporarily no input is provided to the controller, while the lever is "somewhere between A and B". In these situations, when there is no input, the controller assumes that the state is equal to the last received input, unless the following condition occurs. If there is no input for 3 consecutive seconds, the controller assumes that the lever or the connection does not work. The train is stopped with a stop signal and the state is final: a manual reset is needed.

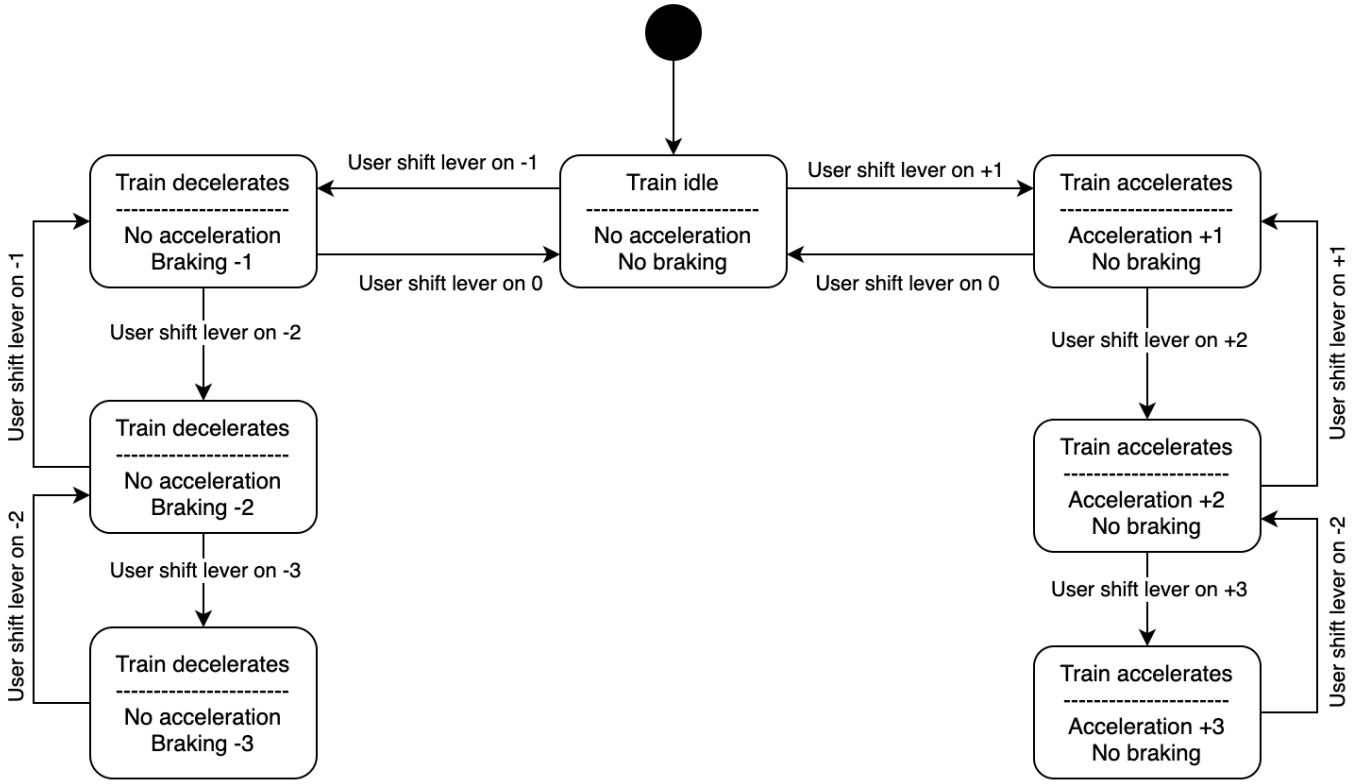
DESIGN AND IMPLEMENTATION

In order to handle the project in a proper way, the system was implemented using incremental steps.

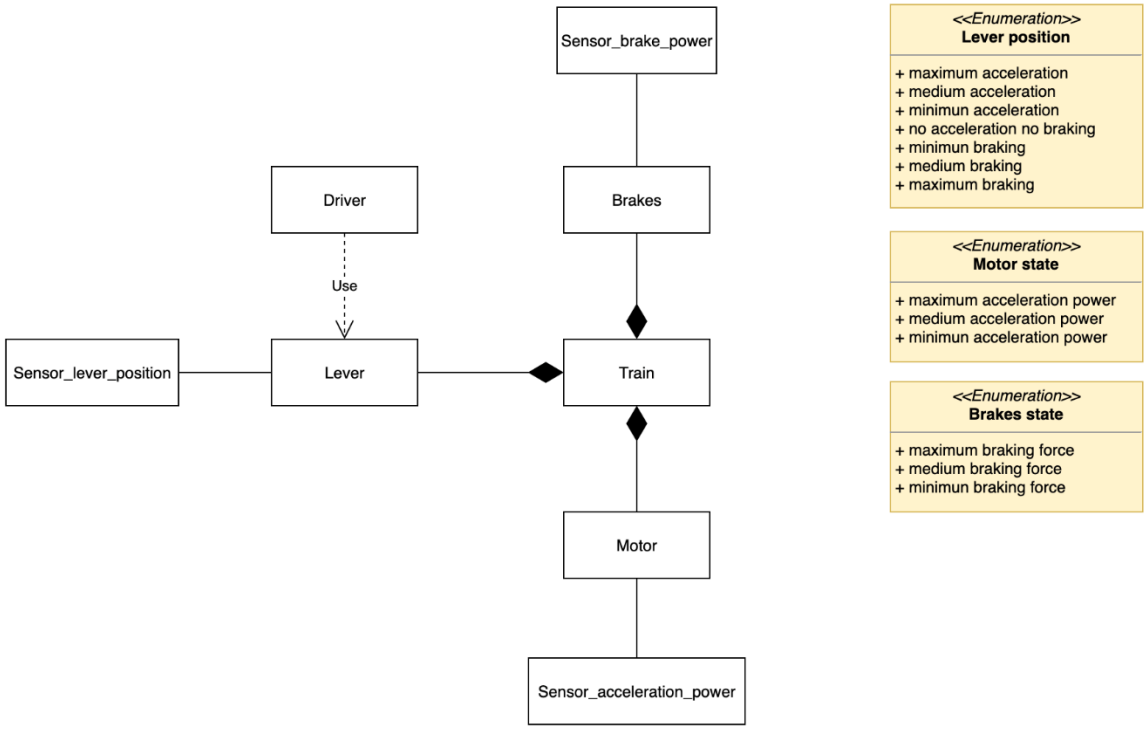
The incremental way was applied both during the design phase and the development phase; this way allows to obtain a parallelism between the conceptual part of the project and the system code.

This section of the report describes the design and the implementation of the system, starting from the basic task to the more complex one. Every step is provided with a UML State diagram and a UML Class diagram that explains the component of the system.

The first incremental step implemented concerns the possibility to manage the use of the lever, when it is controlled by the driver. In this step neither correctness controls nor the 10ms timer, that controls the inputs of the lever, have been inserted. The system reacts to the will of the driver changing the different output of the train.



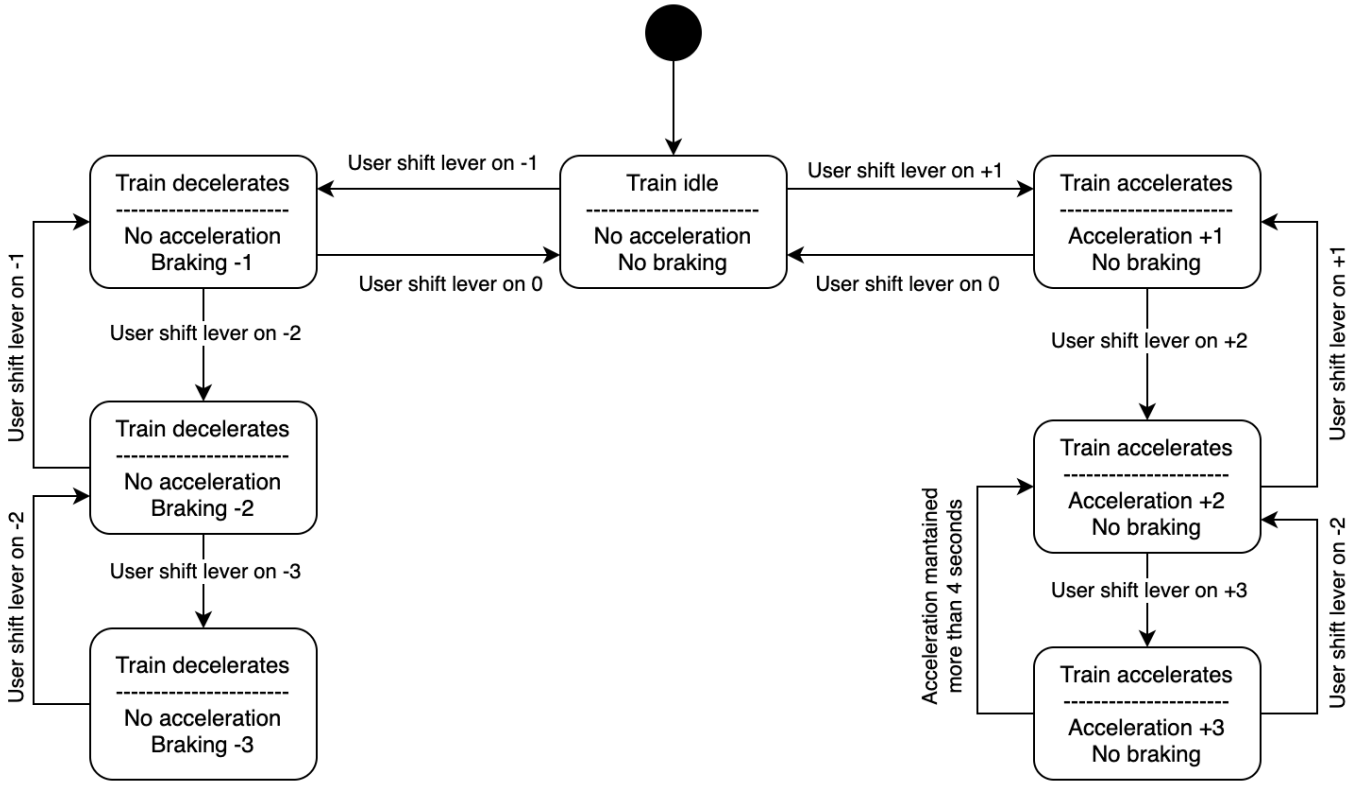
The image above shows all the inputs of the lever, the initial state is the idle one: from this position it is possible to move the lever in a positive state or a negative state.



In this first step the physical components that are exploited are the lever, the motor and the brakes, each with its own reference sensor. The lever is controller and managed by the driver, he chooses the braking force and the acceleration. Every input is then handled by the system and will trigger the related physical component.

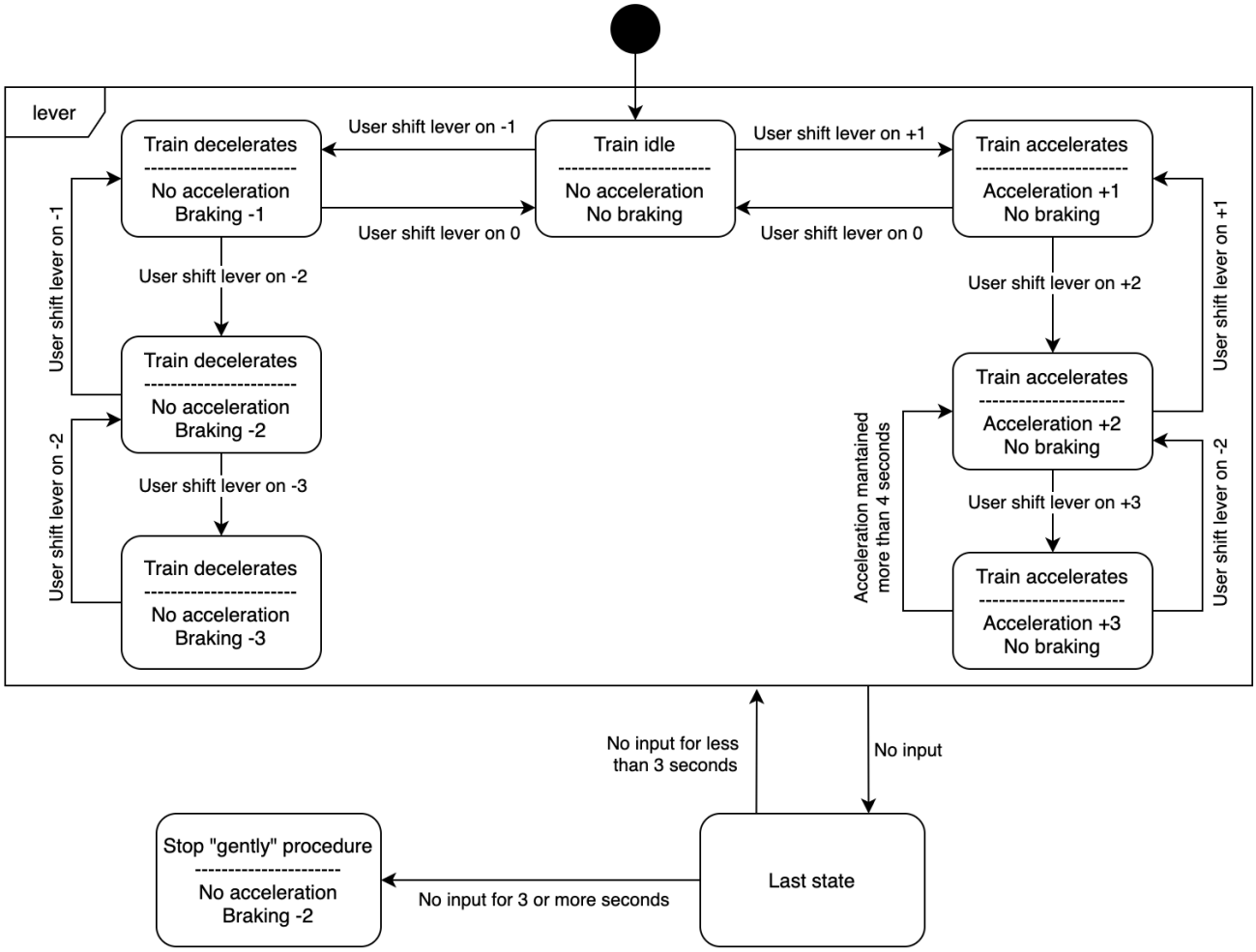
The next step, represented with the same UML diagram, consists in the implementation of the timer. This timer is used to check the position of the lever every 10ms. The timer delay has been set to 10ms according to the specification. This time interval allows not to burden the microprocessor and allows also to check the input of the lever in a continuously and precise delay: indeed, it is humanly impossible to shift the lever twice in 10ms.

The next step is the implementation of a timeout when the maximum acceleration is kept for 4 consecutive seconds. In the below state diagram, it is possible to understand when this timeout is raised: once the lever is inserted in his maximum positive state and it is not moved for more then 4 consecutive seconds, the controller decrease the speed of the train into the medium acceleration.



This addition does not require a change of the previously defined class diagram, since there is no new physical component used.

After the implementation of the 4 seconds timeout, caused by the position of the lever, a way to check for malfunctions, between the lever and its connection, must be implemented within the system.

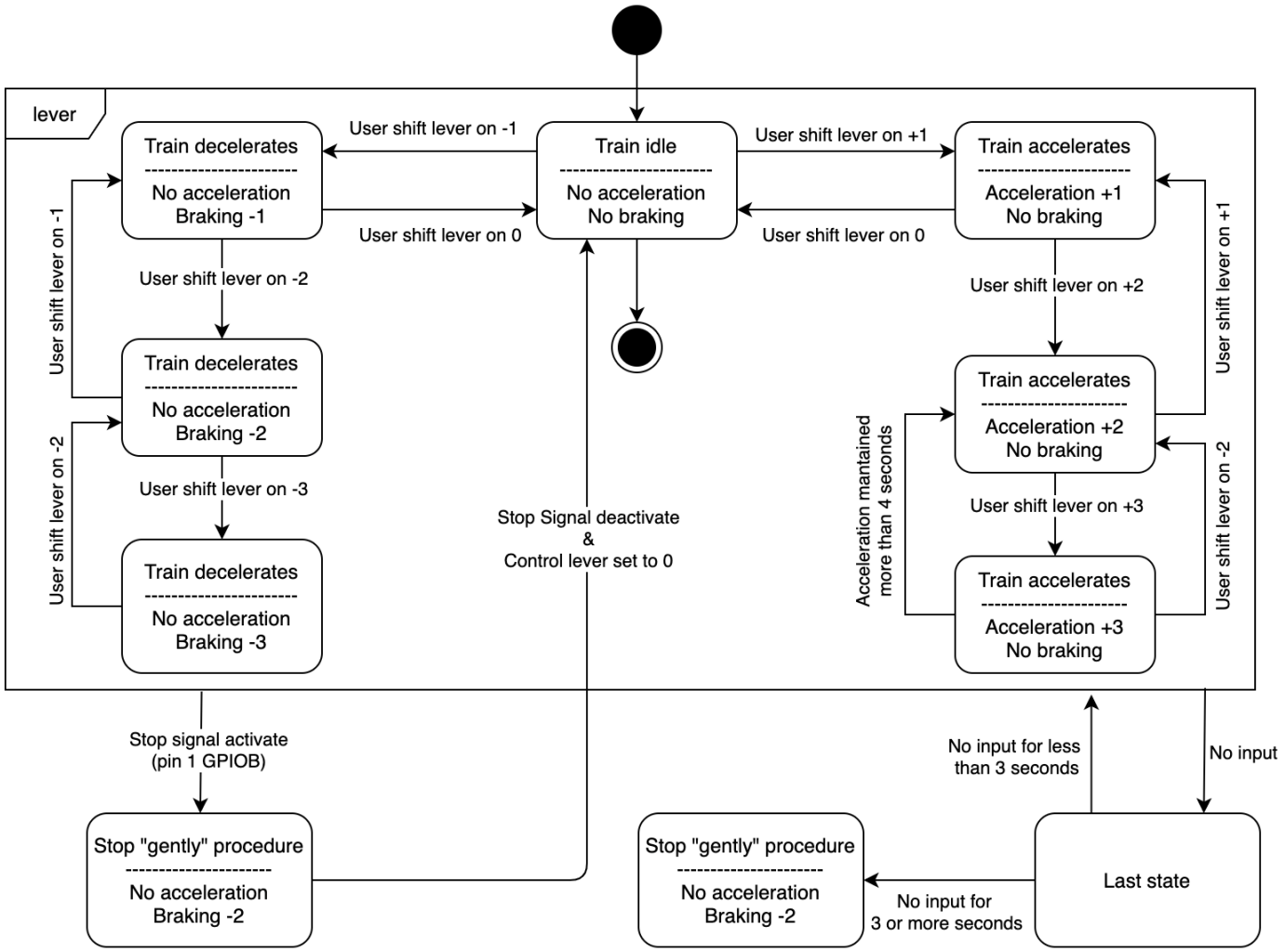


This situation can occur when the connection with the command lever gets interrupted, so that no input is received. In order to handle that case, the controller must keep the previous state, equals to the last input received. If there is no input for 3 consecutive seconds, the controller assumes that the lever or the connection does not work. The train is stopped, the brakes are activated, and a manual reset is mandatory.

The above UML State diagram shows the no input situation: when no input is received the system enter in a new state, which correspond to the last state. If the system receives an input, before the 3 second timer has expired, it changes the state into the new received one and continue its execution normally. Instead, if the system remains in this state for 3 or more consecutive seconds it enters in a final state. This state turns on the brakes on medium force and stops the train. This is a final state and a manual reset is necessary.

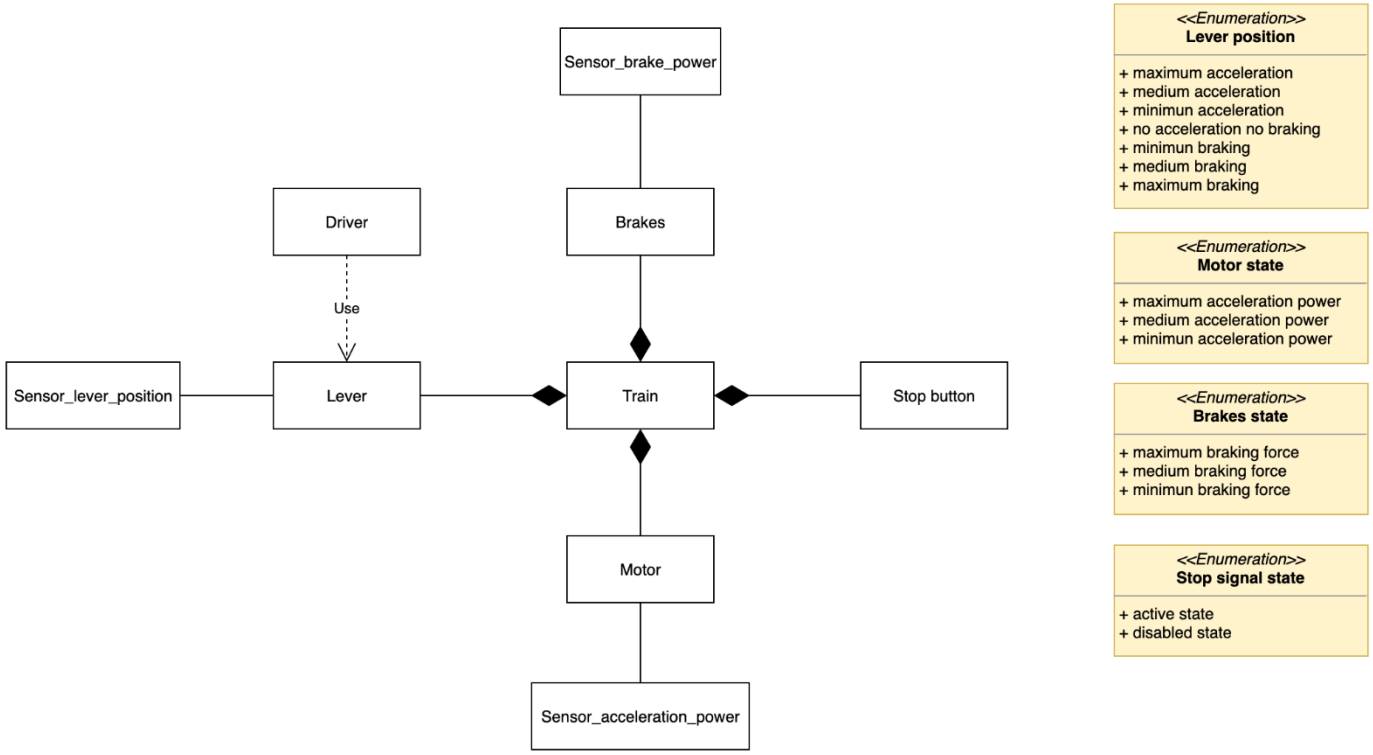
Even after this addition it is not necessary to update the class diagram as there are no new components.

The stop signal request and the emergency break request are not yet implemented. The below State diagram shows how the stop signal is added to the system.



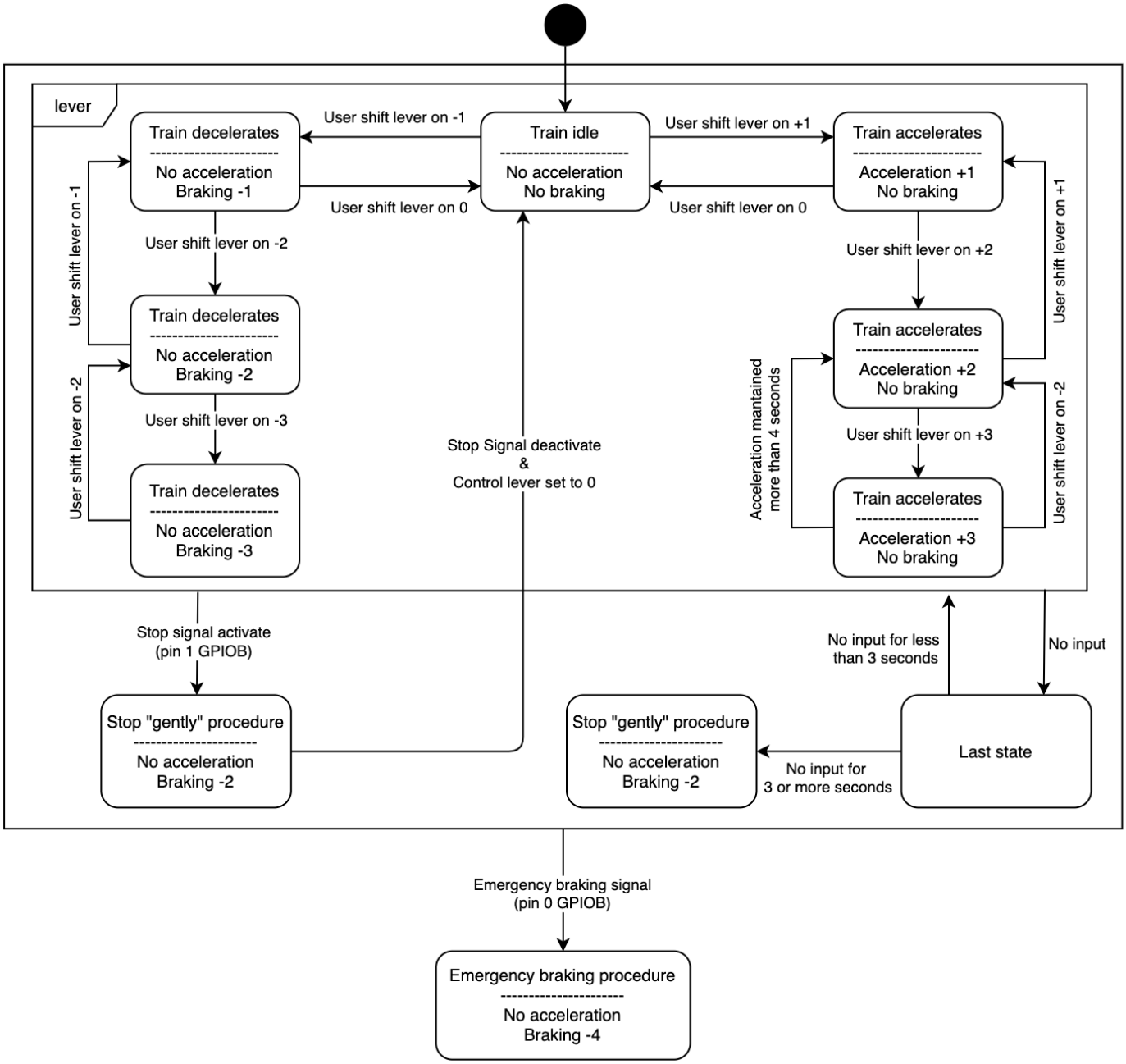
The stop signal is asynchronous and unpredictable, so it can be received when the lever is in any of its position. The diagram shows that the system enters the stop signal state once his signal has been activated on pin 1 of GPIOB. There, the brakes are activated at medium force and all the others input received from the lever are ignored. Once the stop signal is cleared, brakes are still maintained to medium power. The system waits for the user to switch the lever on the idle position. If the lever is not in idle position all other inputs are ignored. When the driver shifts the lever on idle position, the system enter in the anonymous state and the execution continues normally.

Since the stop signal request is raised from a physical component, it is necessary to update the class diagram previously explained.



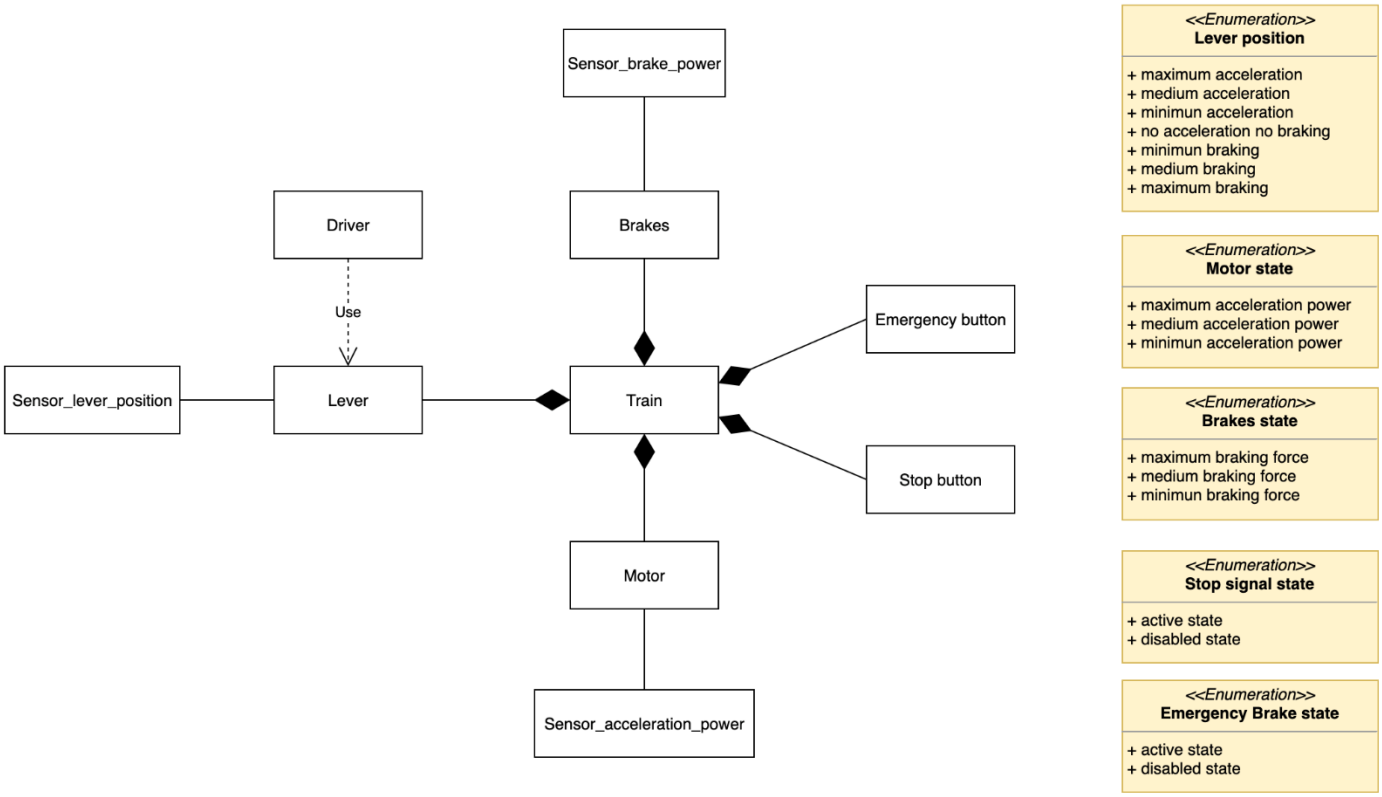
As it is possible to see into the above diagram, it has been inserted the stop button component. This component is related to the train.

The next step is the implementation of the emergency brake request.



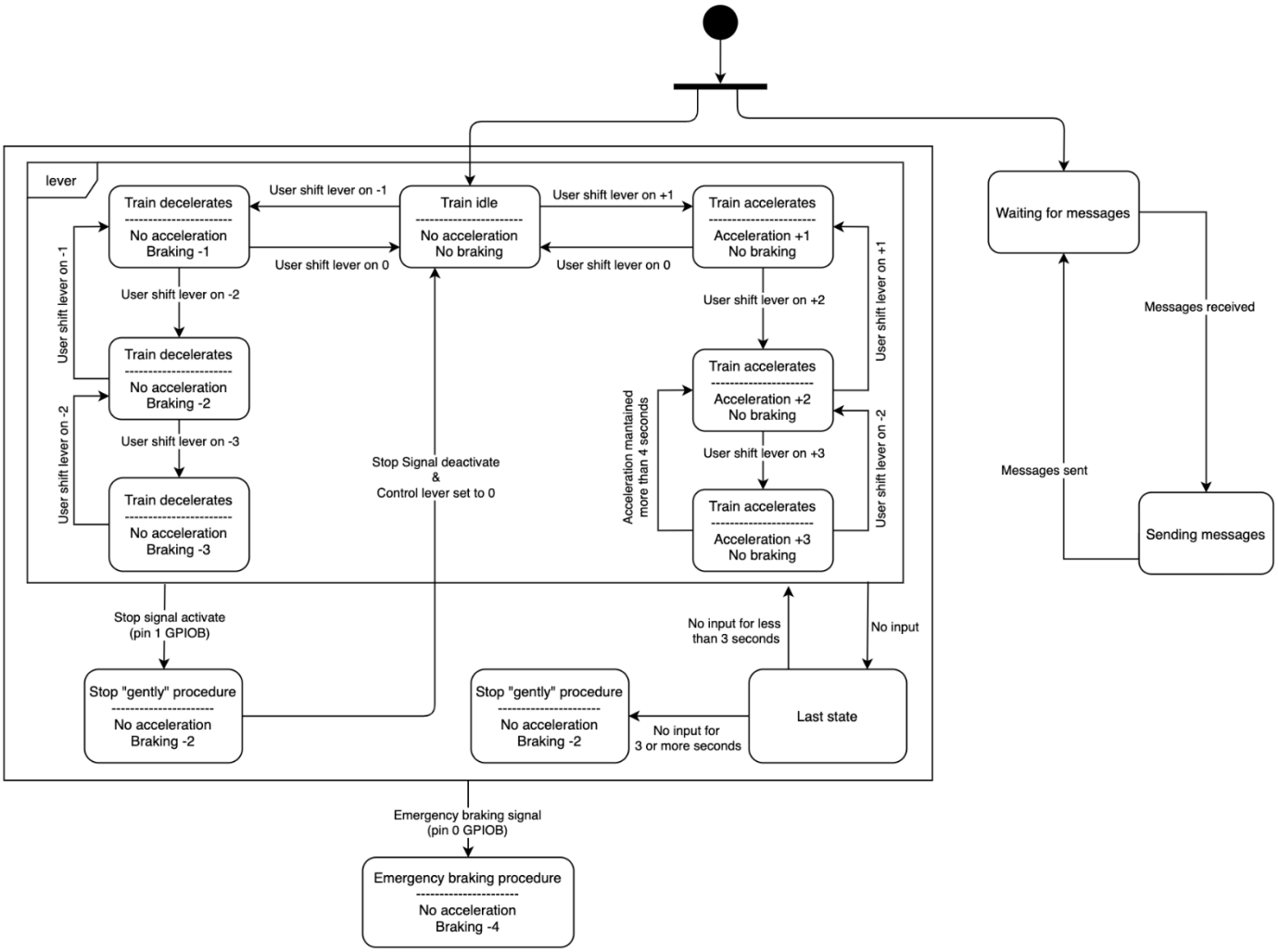
Since the emergency brake request has a greater priority than other events, it must be handled over all the system state. When the emergency brake signal is received on the pin 0 of GPIOB, system enters in the Emergency brake procedure state. There, the controller activates the braking at maximum force (-4) and ignore all the other inputs provided by the lever. No more actions are available: the system need a manual reset.

Once added the emergency brake request, it is mandatory to update again the class diagram, since the emergency brake request is fired from a physical component.



In addition to the stop button, the emergency button has been inserted into the diagram. This component is related to the train.

The last thing that must be inserted is the management of the communications sent from the Traffic management center to the drivers.

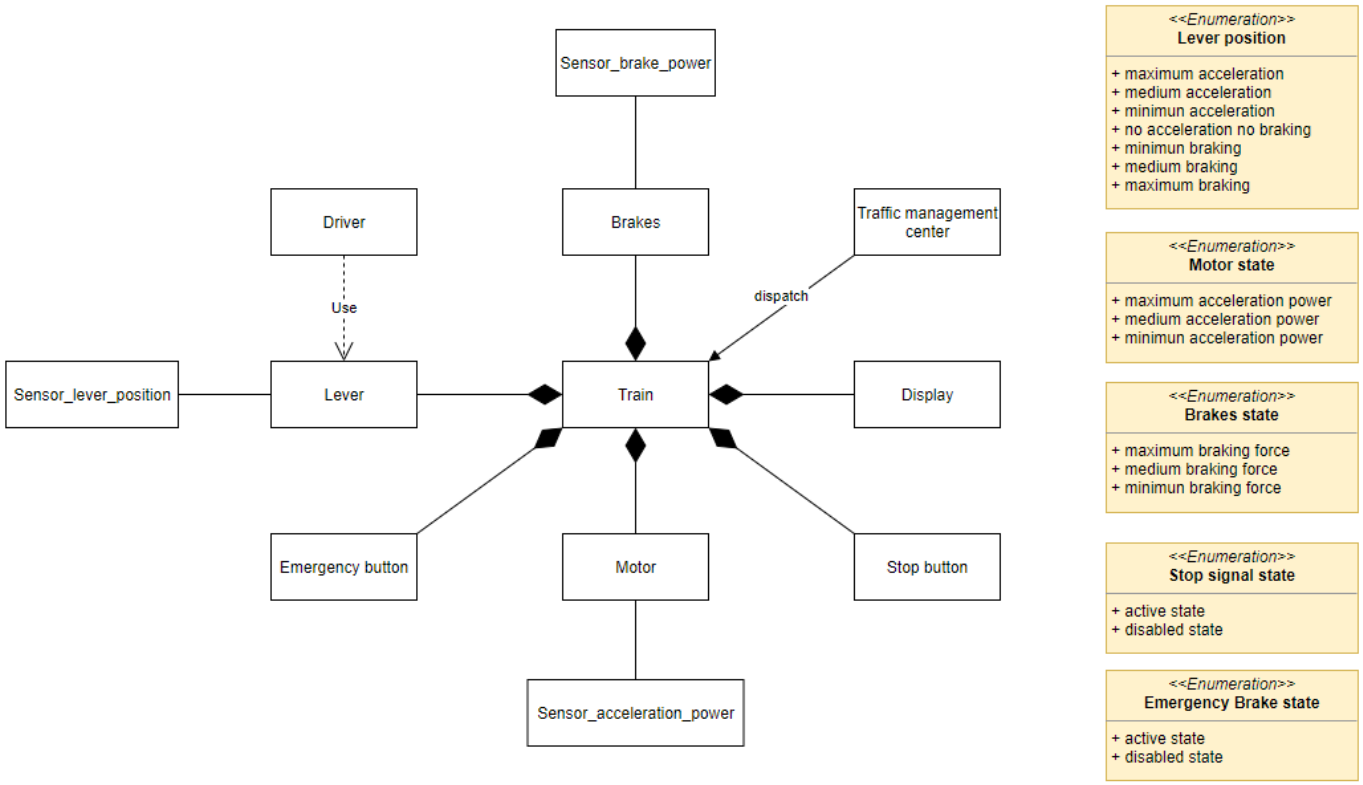


The state diagram above shows the implemenation of the communication messages. The system starts and it is divided into two tasks that are executed in parallel: the left one is the lever system presented on the previous diagram. The right is the communication part with the Traffic center. Once the system has started, it waits an incoming message, receive the message and sent it to the display port on the USART2.

The above diagram is also the rappresentation of the complete system.

With the addition of the traffic message management the system and his rappresentation are now complete. In order to complete also the class diagram, it is needed to update and complete it.

The below diagram is the final one, it contains all the components that act on the system. The last update is the integration of the Traffic Management Center and the display.
The Traffic Management Center is the enitity in charge of sending communications to the driver. This entity communicate with the train, the message are received, managed by the microcontroller and shown into the display.



TEST AND SIMULATION

In order to test the system and find out it works properly it's necessary to provide a test system that integrates perfectly with the microcontroller.

Because of the nature of the analogic lever, to test the system via software it is necessary to implement an event set that simulate the input of the lever. In this way, the inputs that the microcontroller receives are the same as those it would receive with the analogical lever.

To test the system, it is useful to add a task that has maximum priority and generates events that have the same meaning as the expected inputs. This task reads from an array of triples, composed of input event, id of the task that must be raised and the desired delay.

To implement this type of array it's necessary to define a C Structure that contains the previous defined attributes: the event, the task id and the delay.

In this way, there is no need to provide inputs manually but it's necessary to prepare a sequence of expected events that represent the behavior of system's input.

To find out if the system and the controller works properly it's needed to perform all the test cases that concern the normal simulation of the events and the critical situation: it's necessary to provide a test case for the normal acceleration and deceleration of the train, the emergency brake request, the stop signal request and other situation.

The test case that has been provided are the following:

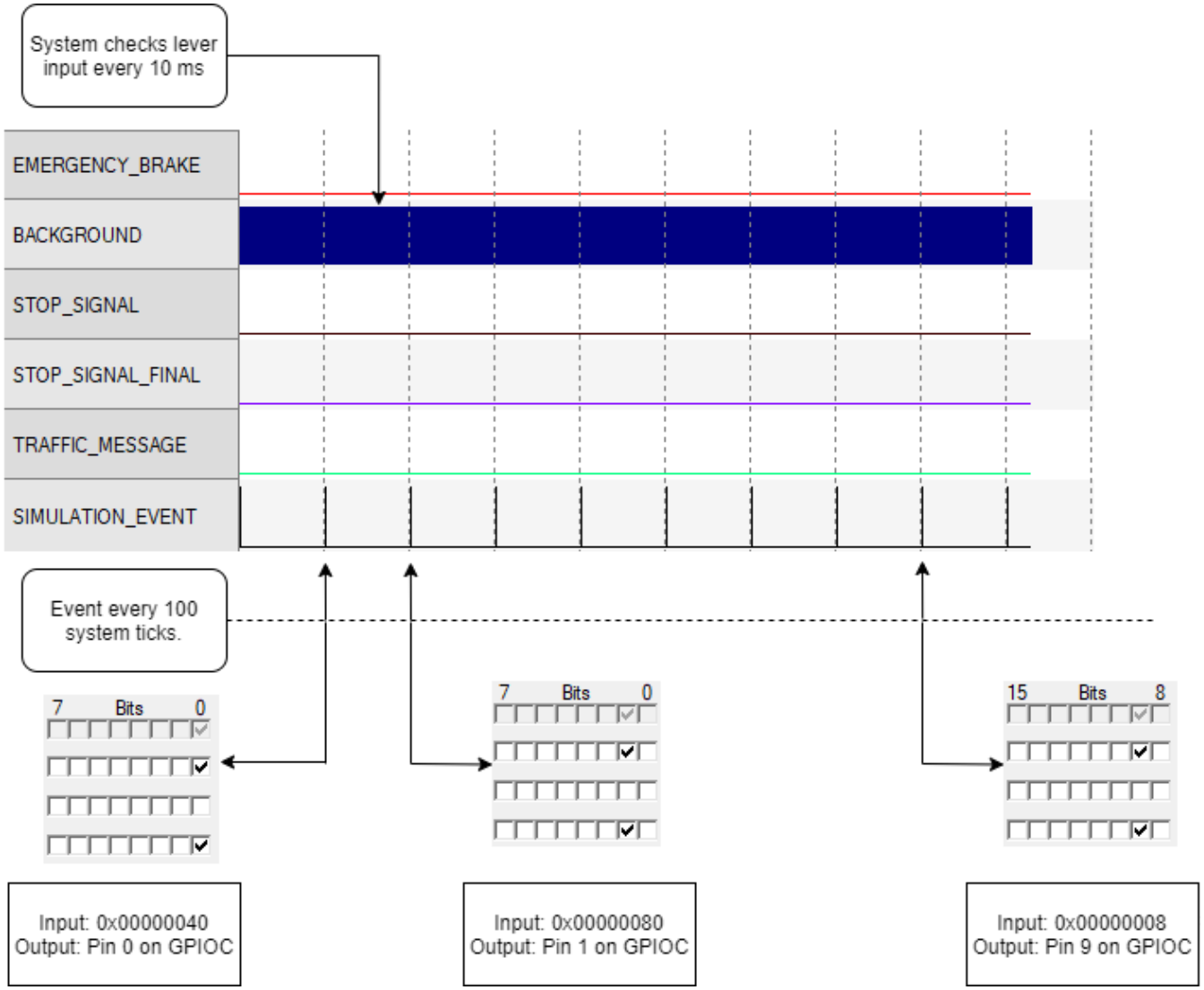
- Normal simulation: a normal situation, where the train accelerate and decelerate without breaking the correctness criteria.
- Stop Signal Simulation: the simulation of the stop signal when the train is accelerating.
- Emergency Brake Simulation: the simulation of the emergency brake request when the train is accelerating.
- Acceleration Timeout Simulation: the simulation of the maintained maximum acceleration for more than 4 seconds.
- No inputs Timeout Simulation: the simulation of the no inputs situation maintained for more than 3 seconds.
- Traffic Messages Simulation: the simulation of receiving message over the serial line.

NORMAL SIMULATION

This simulation represents the normal behavior of the system: the driver, starting from the idle position, accelerate progressively until the max acceleration. Once the maximum acceleration is reached, the lever is shifted position by position until it reaches the strong braking force. Between every event there is a delay of 100 timer tick (1s).

The events of the simulation are the followings:

1. no_Acceleration_No_Braking
2. min_Acceleration
3. med_Acceleration
4. max_Acceleration
5. med_Acceleration
6. min_Acceleration
7. no_Acceleration_No_Braking
8. min_Braking
9. med_Braking
10. str_Braking



In the previous image it is represented the behavior of the system: it is possible to see the state of the tasks during the execution in the logic analyzer.
Every 100 systems tick (1s) an event is generated; the flag is raised and the input changes. Every 10ms the background task checks the input (simulated by the event) and turn on the corresponding output pin on GPIOC.

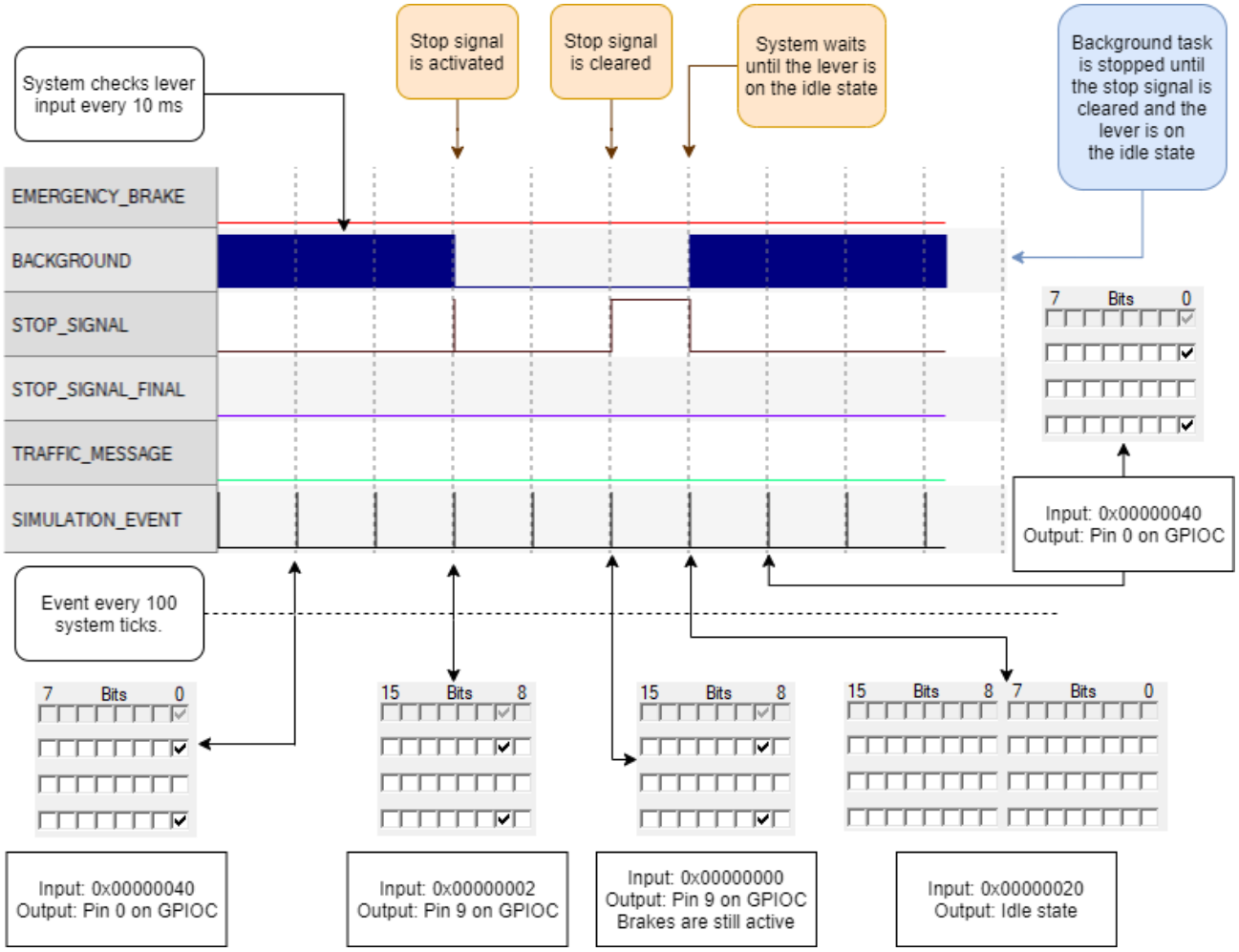
The image above shows the normal input of the lever: with a complete acceleration and a complete deceleration. All the inputs are provided by the driver with the lever. In this image is shown only the second, third and eight pin output, all the other pins are turned on one by one when the simulated input is on the corresponding position.

STOP SIGNAL SIMULATION

This simulation represents the normal behavior of the system: the driver, starting from the idle position, accelerate progressively until the medium acceleration. While the medium acceleration is maintained the system receives a stop signal request. Train is gently stopped. After the clearance of the stop signal, the driver shifts the lever to the idle state and restart the train normally. Between every event there is a delay of 100 timer tick (1s).

The event of the simulation are the followings:

1. no_Acceleration_No_Braking
2. min_Acceleration
3. med_Acceleration
4. stop_Signal
5. med_Acceleration
6. clear_Stop_Signal
7. no_Acceleration_No_Braking
8. min_Acceleration
9. med_Acceleration
10. max_Acceleration



In the previous image it is represented the behavior of the system: it is possible to see the state of the tasks during the execution in the logic analyzer.
Every 100 systems tick (1s) an event is generated; the flag is raised and the input changes. Every 10ms the background task checks the input (simulated by the event) and turn on the corresponding output pin on GPIOC.

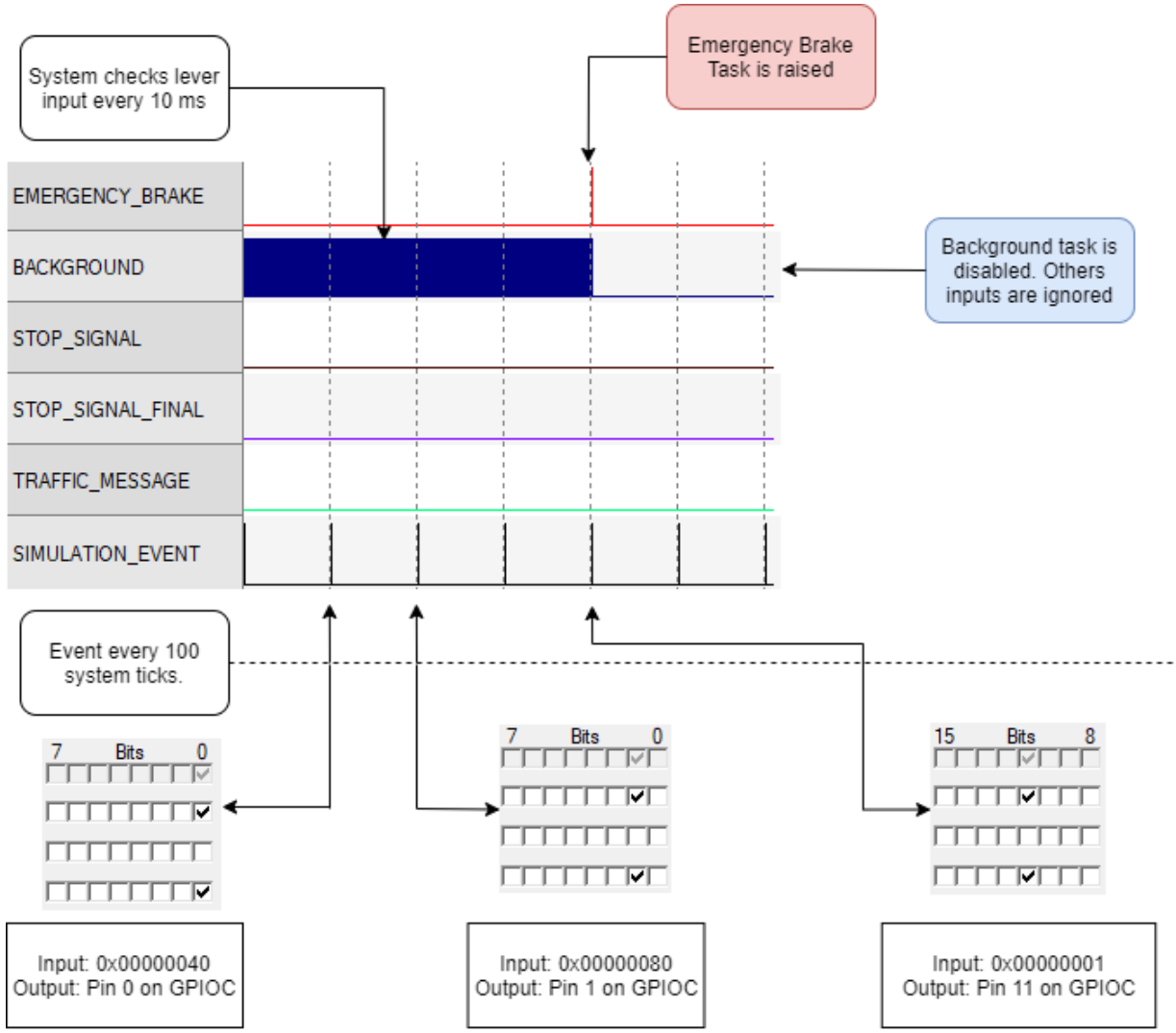
The simulation start turning on the led of the idle state, after 100 system tick the input changes, the controller switches the led on the output of the minimum acceleration (Pin 0 of GPIOC). The driver, using the lever, express his will to speed up the train. While medium acceleration is on, the system receives a stop signal. The controller turns on the brakes at medium force (Pin 9 of GPIOC) and ignore all the other inputs provided by the lever and the driver. Once the stop signal is cleared, the controller keeps the brakes active and waits for the driver to put the lever on the idle position. Until the lever is not on idle position, all the other inputs provided from the lever and the driver are ignored. Once the lever is on idle position, the system restarts its normally execution: driver can accelerate and resume his driving normally.

EMERGENCY BRAKE SITUATION

This simulation represents the behavior of the system during an emergency brake situation. The driver, starting from the idle position, start the acceleration until he reaches the maximum power. While the lever is still on the maximum power an emergency request is raised. Between every event there is a delay of 100 timer tick (1s).

The event of the simulation are the followings:

1. no_Acceleration_No_Braking
2. min_Acceleration
3. med_Acceleration
4. max_Acceleration
5. emergency_Brake
6. med_Braking
7. str_Braking
8. med_Braking
9. min_Braking
10. no_Acceleration_No_Braking



In the previous image it is represented the behavior of the system: it is possible to see the state of the tasks during the execution in the logic analyzer.
Every 100 systems tick (1s) an event is generated; the flag is raised and the input changes. Every 10ms the background task checks the input (simulated by the event) and turn on the corresponding output pin on GPIOC.

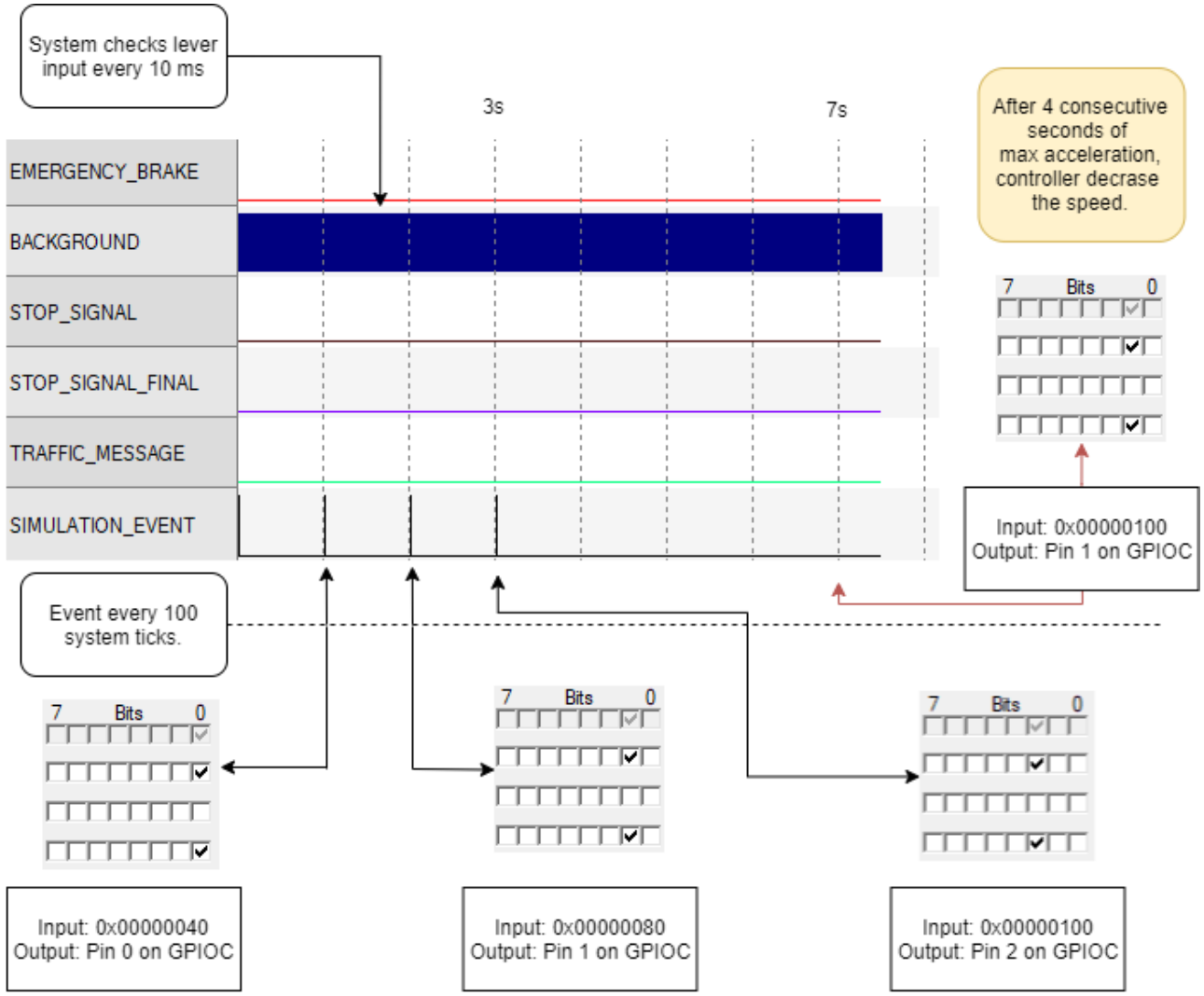
The simulation start turning on the led of the idle state, after 100 system tick the input changes, the controller switches the led on the output of the minimum acceleration (Pin 0 of GPIOC). Again, after 100 system ticks, the input changes, the controller switches the led on the output of the medium acceleration (Pin 1 of GPIOC). The same thing happens after 100 system ticks for the maximum acceleration. During the duration of the maximum acceleration an Emergency Brake request is raised: the controller wakes up the emergency brake task. This task handles the emergency brake: it activates the maximum brakes and ignores any other input issued by the driver with the lever. The system no longer checks every 10ms for the lever input and must be reset manually.

ACCELERATION TIMEOUT SIMULATION

This simulation represents the behavior of the system when the lever is kept on the maximum acceleration for 4 consecutive seconds. The driver, starting from the idle position, start the acceleration until he reaches the maximum power. The lever is kept in this state for more than 4 consecutive seconds. Between every event there is a delay of 100 system tick (1s), except for the maximum acceleration event that it is hold for 500 system tick (5s).

The event of the simulation are the followings:

1. no_Acceleration_No_Braking
2. min_Acceleration
3. med_Acceleration
4. max_Acceleration (5s)
5. med_Acceleration
6. min_Acceleration
7. med_Acceleration
8. max_Acceleration (5s)
9. med_Acceleration
10. min_Acceleration



In the previous image it is represented the behavior of the system: it is possible to see the state of the tasks during the execution in the logic analyzer.
Every 100-system tick (1s) an event is generated; the flag is raised and the input changes. Every 10ms the background task checks the input (simulated by the event) and turn on the corresponding output pin on GPIOC.

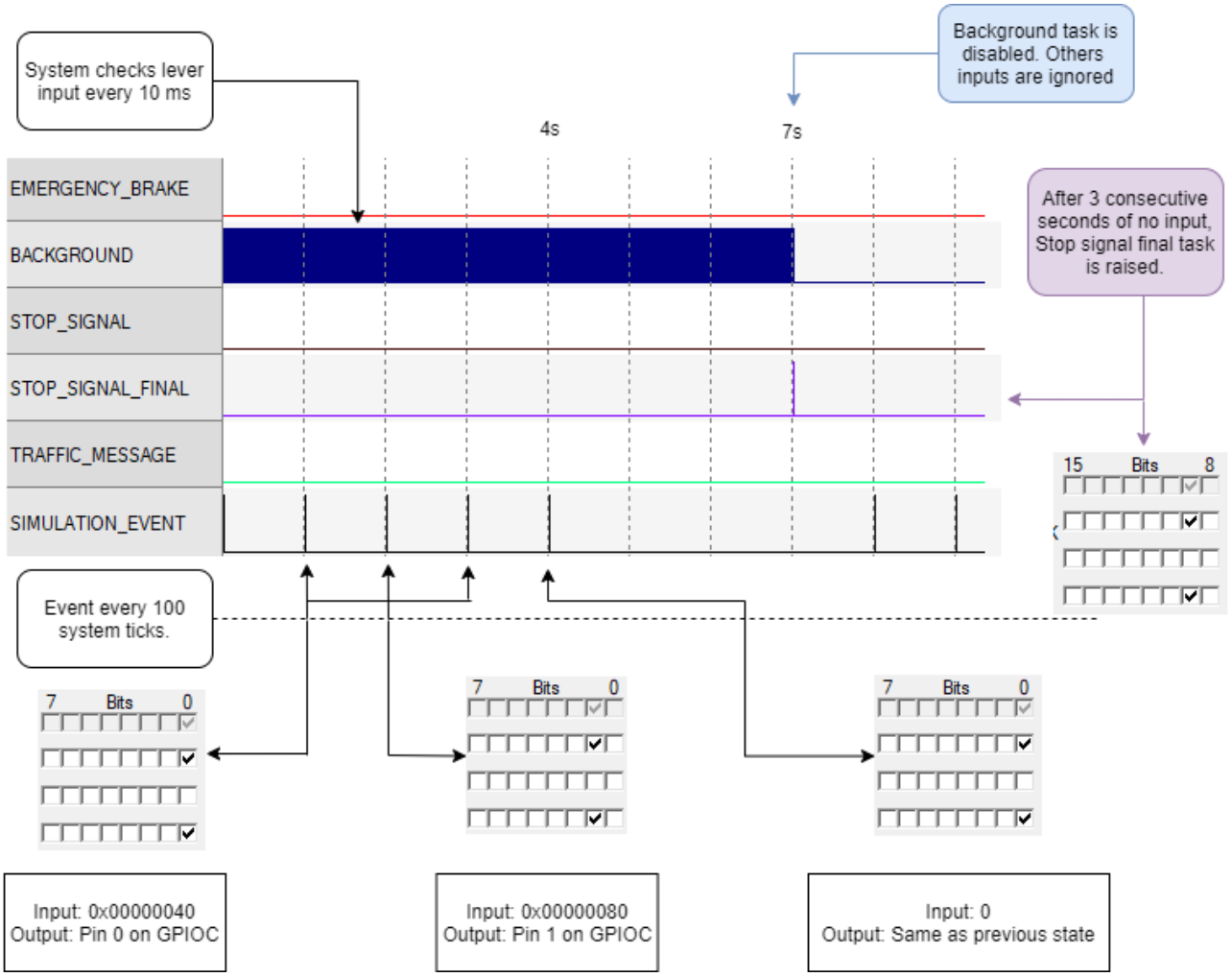
The simulation start turning on the led of the idle state, after 100 system tick the input changes, the controller switches the led on the output of the minimum acceleration (Pin 0 of GPIOC). Again, after 100 system tick, the input changes, the controller switches the led on the output of the medium acceleration (Pin 1 of GPIOC). The same thing happens after 100 system ticks for the maximum acceleration. The driver keeps the maximum acceleration for more than 4 seconds, in this way, after 4 consecutive seconds that the lever is in this same position, the controller decrease the speed and set the acceleration output pin on the Pin 1 of GPIOC (medium acceleration). Driver can shift down the lever and the system return to his normal state.

NO INPUTS TIMEOUT SIMULATION

This simulation represents the behavior of the system when there is a malfunction between the lever and the controller. The driver, starting from the idle position, accelerate with two force level. After 300 system ticks there is no input provided to the system. This situation is maintained for more than 3 seconds. Between every event there is a delay of 100 timer tick (1s), except for the no input event that it is hold for 400 system tick (4s).

The event of the simulation are the followings:

1. no_Acceleration_No_Braking
2. min_Acceleration
3. med_Acceleration
4. min_Acceleration
5. no_Input (4s)
6. min_Braking
7. med_Braking
8. min_Braking
9. no_Acceleration_No_Braking
10. min_Acceleration



In the previous image it is represented the behavior of the system: it is possible to see the state of the tasks during the execution in the logic analyzer.
Every 100-system tick (1s) an event is generated; the flag is raised and the input changes. Every 10ms the background task checks the input (simulated by the event) and turn on the corresponding output pin on GPIOC.

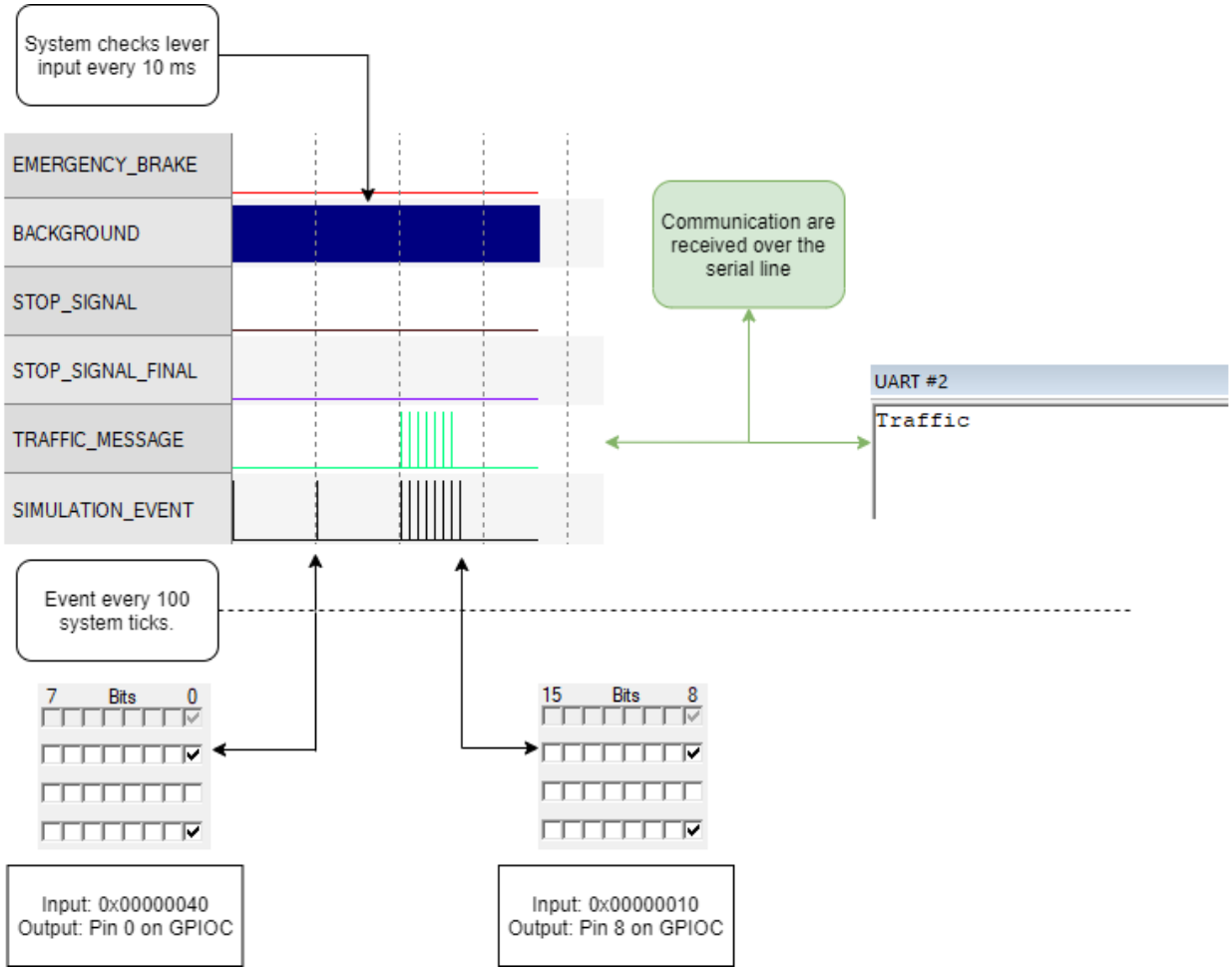
The simulation start turning on the led of the idle state, after 100 system tick the input changes, the controller switches the led on the output of the minimum acceleration (Pin 0 of GPIOC). After a quickly medium acceleration, the lever switches back to the minimum acceleration. At this moment (4s) the event corresponding with the no input is raised: the controller still maintains the previous state (minimum acceleration). The no input event is maintained for more than 3 seconds, thus, as it is possible to see on the image, the controller, at seconds 7, raise the stop signal request, assuming that there is a malfunction between the lever and the connection. The “Stop Signal Final” task is fired: background task is stopped and never resumed. Brakes are activated at medium force and all the other inputs provided from the lever and the driver are ignored. At this moment the system needs a manual restart.

TRAFFIC MESSAGES SIMULATION

This simulation represents the behavior of the system when there are communications provided by the Traffic Management Center over serial line. The driver, starting from the idle position, starts the acceleration. During the journey a communication concerning traffic is received. Between every event there is a delay of 100 timer tick (1s), except for the character input event that it is hold for 10 system tick.

The event of the simulation are the followings:

1. no_Acceleration_No_Braking
2. min_Acceleration
3. 'T'
4. 'r'
5. 'a'
6. 'f'
7. 'f'
8. 'i'
9. 'c'
10. min_Braking



In the previous image it is represented the behavior of the system: it is possible to see the state of the tasks during the execution in the logic analyzer.
Every 100-system tick (1s) an event is generated; the flag is raised and the input changes. Every 10ms the background task checks the input (simulated by the event) and turn on the corresponding output pin on GPIOC.

The simulation start turning on the led of the idle state, after 100 system ticks the input changes, the controller switches the led on the output of the minimum acceleration (Pin 0 of GPIOC). At this point, a communication over the serial line (USART1) is received. The “Traffic Message” task is raised as soon as a message event, containing a char, is fired. The system shows the received communication on the USART2 display. Once the message is completely received, the driver reads it and decide to shift down the lever, activating the minimum braking.

CONCLUSION

The previous chapter of this repost present and explain all the given tests and simulations that has been executed on the system.

Concluding, it is possible to state that all test cases performed have produced satisfactory and expected results. The system behaved according to the requests specified at the beginning of the report.