Embedded Systems Modeling and Design

Instructor: Prof. Mehdi Sedighi

Amirkabir University of Technology
(Tehran polytechnic)

Design and Modeling of an Intelligent Automotive Airbag System & Petri Net-Based Modeling of an Elevator System

Authors:

Reza Adinepour adinepour@aut.ac.ir

Spring 2024

Contents

1 Airbag System

Imagine you want to design a smart car airbag system. This system consists of 4 impact sensors located at the front, rear, right, and left sides of the vehicle, 2 airbags placed at the front and left side for the driver, a speed sensor, a movement direction sensor, and a driver distance sensor, each of which will be described further.

1.1 Project Description

The front sensor activates if the vehicle's speed exceeds 40 $\frac{Km}{h}$ and, in the event of a collision risk, inflates the airbag in front of the driver. The rear sensor activates if the vehicle is moving forward at less than *30* $\frac{Km}{h}$ or moving backward at more than *10* $\frac{Km}{h}$, inflating the airbag in front of the driver. The side sensors are not dependent on speed and, in the event of a collision risk from either side of the vehicle, inflate the left side airbag for the driver. If more than one sensor is activated, it is possible for both airbags to inflate, but in such a case, priority will clearly be given to the front airbag for the driver.

Additionally, there is another sensor that measures the distance between the driver and the steering wheel. If this distance is less than *30 cm*, the airbag should only inflate halfway to avoid harming the driver. This must be completed within *30 ms* after detecting an imminent collision. If the distance is more than *30 cm* but less than *40 cm*, the airbag should inflate to $\frac{3}{4}$ of its capacity within *40 ms*. If the distance is more than *40 cm*, the airbag should fully inflate within *50 ms*. The side airbag should fully inflate within *60 ms*.

1.2 Project Detail

1. First, choose one of the types of MoCs (Models of Computation) covered in this course that you think is most suitable for describing, modeling, implementing, and evaluating this system. Fully explain your reasons for your choice. Note that this question does not have a single correct answer. Therefore, your answer will be evaluated based on the validity of your reasoning and logic, not on a specific correct answer.

2. Now, design and describe this system based on the MoC you chose in the previous step.

3. The next step is always modeling and then implementation. For modeling, assume that this system will be implemented on a processor that, in addition to controlling the airbags, also controls the interior climate of the car and the lights inside and outside the vehicle. Naturally, for regulating the interior temperature, a thermal sensor is required to measure the temperature inside the vehicle and adjust the airflow temperature based on the driver's desired temperature. Therefore, the same driver distance sensor used for airbags can be used for regulating the intensity of warm or cold air inside the vehicle. Do these new assumptions affect your choice in step 1? Explain. If needed, revise the description provided in step 2.

4. Based on the assumptions and considerations in step 3, what type of implementation would you propose for your designed system (e.g., process-based, thread-based, interrupt-based)? Fully explain your reasons for your choice. Also, draw a flowchart of your proposed implementation.

5. Estimate the WCET (Worst-Case Execution Time) for your program. For this, assume that each access to a sensor or actuator takes 1 ms and each atomic instruction takes 10 nanoseconds. To avoid bouncing, each sensor is read at least 3 times and at most 5 times, with the first 3 consistent readings being selected. Other tasks that the processor needs to handle will take up to 90% of the processor's computational power. (If you think more assumptions are needed to solve this problem, specify them and proceed with solving the problem.)

6. Can you guarantee that your design is real-time? In other words, can you ensure that the timing constraints mentioned above will be met? Provide a comprehensive explanation.

2 Definitions

2.1 Embedded Systems and Cyber-Physical Systems

Embedded systems are specialized computing systems that are dedicated to performing specific tasks within a larger mechanical or electrical system. They are designed to optimize efficiency, reliability, and real-time operation, often with constrained resources. Cyber-Physical Systems (CPS), on the other hand, are integrations of computation, networking, and physical processes. They involve embedded computers and networks that monitor and control the physical processes, typically with feedback loops where physical processes affect computations and vice versa. CPSs are characterized by their ability to interact with the physical world and adapt to changing conditions in real-time.

The intelligent automotive airbag system falls under the category of Cyber-Physical Systems (CPS). It integrates sensors (physical components), control algorithms (computational elements), and actuators (physical response mechanisms) to enhance passenger safety through real-time monitoring and adaptive response to collision scenarios.

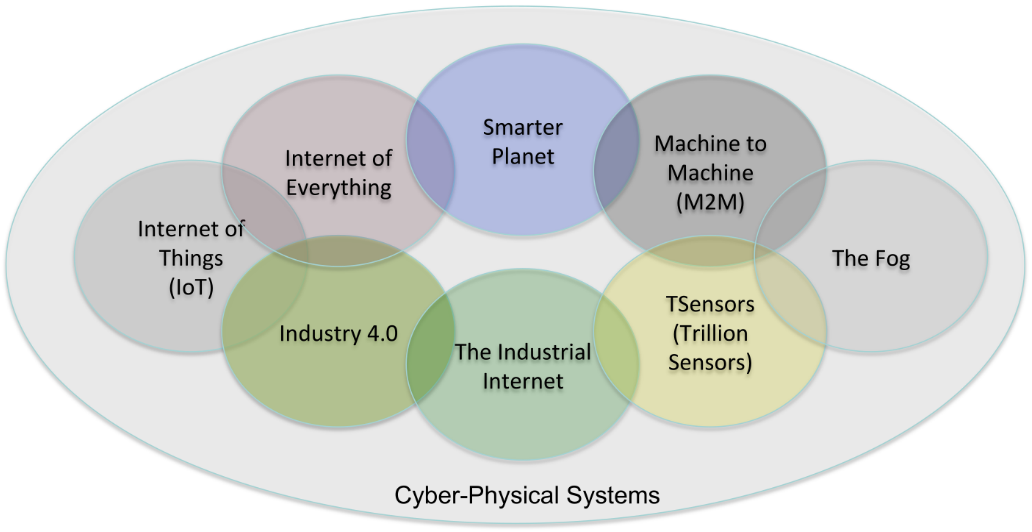

Figure 1: Overview of cyber physical systems.

2.2 Model of Computation (MoC)

Models of Computation (MoC) refer to the theoretical frameworks and mathematical models used to design and analyze the behavior and performance of computational systems. MoCs provide a formal structure to represent how systems process information, execute operations, and interact with their environment. They help in defining the computational processes, synchronization, communication, and resource management within a system. Common models include finite state machines, dataflow models, event-driven models, and hybrid models, each suited to different types of computational tasks and system requirements.

Here are several Models of Computation (MoCs) with their definitions:

2.2.1 Actor Model

The Actor Model is a conceptual model of concurrent computation that treats ”actors” as the fundamental units of computation. In this model, actors are independent entities that communicate through asynchronous message passing, making it well-suited for designing distributed systems.

The Actor Model is employed in distributed systems, parallel processing, and real-time applications.

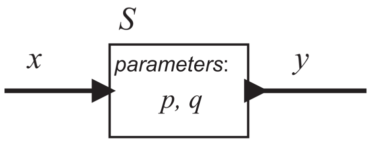

Figure 2: Continuous-time simple actor model.

2.2.2 Finite State Machines (FSM)

Finite State Machines are mathematical models of computation used to design both computer programs and sequential logic circuits. They are composed of a finite number of states, transitions between those states, and actions. FSMs are particularly useful for systems that can be clearly defined in terms of states and transitions, where each state represents a specific condition or situation of the system.

FSMs are often used in control systems, protocol design, and digital circuit design.

2.2.3 Hybrid Systems

Hybrid Models combine elements of different computation models to handle complex systems that require multiple paradigms. For instance, a hybrid model might integrate continuous and discrete event-driven behavior to capture the dynamics of physical systems along with digital control logic.

Hybrid models are used in cyber-physical systems, robotics, and embedded systems where both continuous physical processes and discrete control logic must be managed.

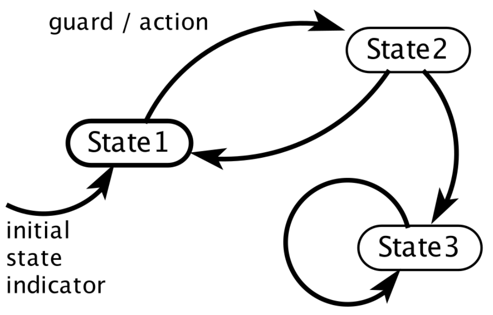

Figure 3: Visual notation for a finite state machine.

2.2.4 Timed Automaton

Synchronous Models are based on the concept that system components operate in lockstep, driven by a global clock. Every computation step is synchronized with this clock, making it easier to reason about timing and sequencing of operations.

These models are widely used in digital circuit design, where precise timing is crucial, as well as in synchronous programming languages for real-time systems.

2.2.5 Dataflow

Dataflow Models represent computations as directed graphs where nodes represent operations or functions, and edges represent the data paths between them. In these models, the execution of operations is driven by the availability of data, rather than by a sequence of commands.

In dataflow models, the signals providing communication between the actors are sequences of message, where each message is called a token. That is, a signal s is a partial function of the form:

$$s : \mathbb{N} \to V_s, \tag{1}$$

where V_s is the type of the signal, and where the signal is defined on an initial segment $\{0, 1, , n\} \subset N$, or (for infinite executions) on the entire set $\mathbb{N}$. Each element $s(n)$ of this sequence is a token. A (determinate) actor will be described as a function that maps input sequences to output sequences.

Dataflow models are commonly used in parallel computing, signal processing, and stream processing applications.

2.2.6 Petri Nets

Definition: Petri Nets are graphical and mathematical tools for modeling concurrent, asynchronous, distributed, parallel, and nondeterministic systems. They consist of places, transitions, and tokens, where the state of the system is represented by the distribution of tokens across places.

Petri Nets are used in the design and analysis of communication protocols, workflow management, and industrial process control.

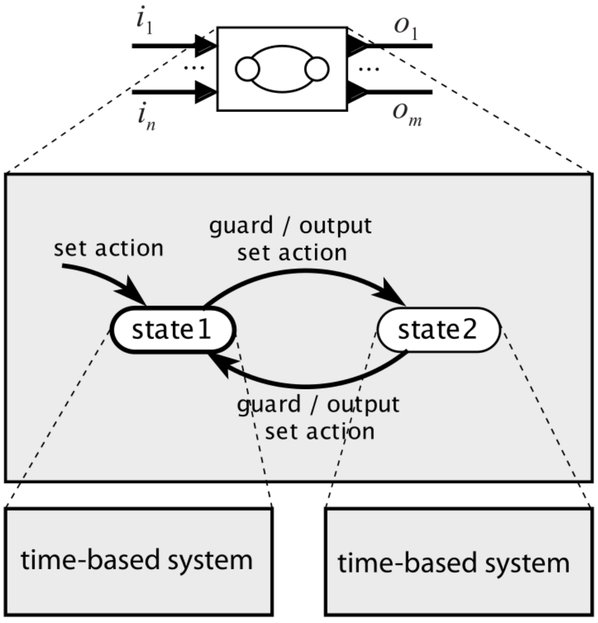

Figure 4: Notation for hybrid systems.

2.2.7 Discrete Event-Driven Systems

Event-Driven Models focus on the processing of events or changes in state. The system reacts to incoming events, triggering specific responses or transitions. This model is effective for applications where events occur sporadically and the system must respond in real-time.

Event-driven models are used in user interfaces, real-time systems, and reactive systems like automotive airbag systems.

3 Choose The Best MoC

For designing an intelligent automotive airbag system, the best Model of Computation is the **Event-Driven Model.** and in this design we use Discrete Event-Driven models for better description.

The Event-Driven Model is well-suited for systems that need to respond promptly to external stimuli or events. In the context of an intelligent automotive airbag system, this model is ideal due to the following reasons:

1. **Real-Time Responsiveness:** The airbag system must react instantly to collision events. The event-driven model ensures that the system can handle and prioritize these critical events with minimal delay.

2. **Sensor Integration:** This model can efficiently manage inputs from multiple sensors (accelerometers, gyroscopes, pressure sensors) and process these events to determine the necessity and timing of airbag deployment.

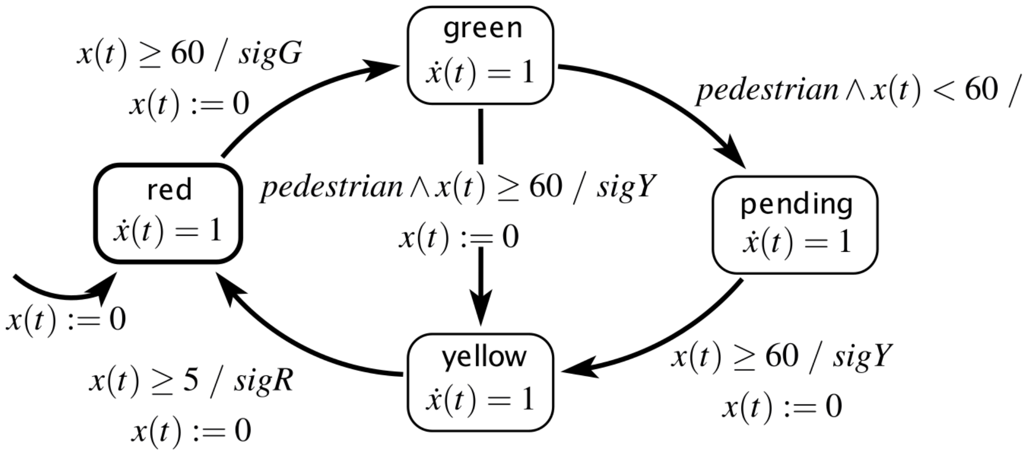

Figure 5: A timed automaton variant of the traffic light controller.

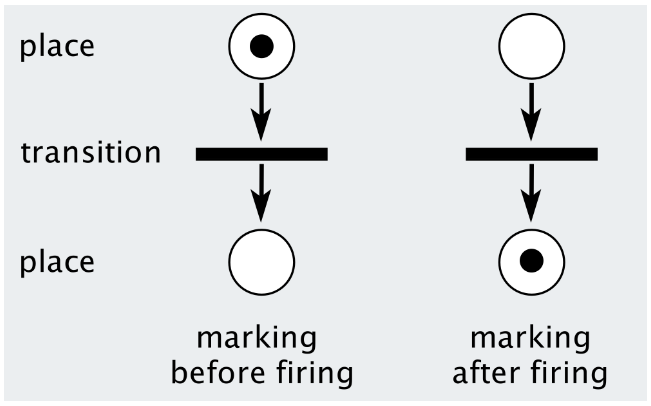

Figure 6: Notation for Petri Net.

3. **Scalability and Modularity:** The event-driven approach allows for modular design, where each event (e.g., impact detection, seat occupancy) can be handled by specific modules. This makes the system easier to develop, test, and maintain.

4. **Resource Efficiency:** It optimizes resource usage by only activating computational processes when specific events occur, rather than continuously polling sensors.

4 Design and Describe Model

4.1 System Overview

4.1.1 Sensors

In This design we have these sensors:

1. **Impact Sensors:** Four sensors located at the front, rear, right, and left sides.

2. **Speed Sensor:** Measures the vehicle's speed.

3. **Movement Direction Sensor:** Detects the vehicle's movement direction (forward or backward).

4. **Driver Distance Sensor:** Measures the distance between the driver and the steering wheel.

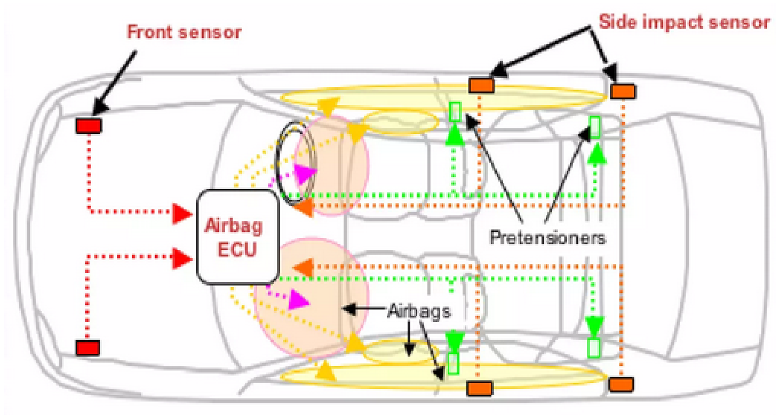

Figure 7: Location of sensors.

4.1.2 Control Unit

1. Processes data from all sensors.

2. Determines which airbags to deploy and to what extent.

4.1.3 Actuators

1. **Front Airbag:** Positioned in front of the driver.

2. **Side Airbag:** Positioned to the left of the driver.

4.2 Syntax of Model

This system has four analogue inputs: $S_1 \in [-10, 10]$, $S_2 \in [-10, 10]$, $S_3 \in [0, 1]$, and $S_4 \in [0, 1]$. These inputs represent acceleration sensors (S_1 and S_2) and impact sensors (S_3 and S_4) that are used by the controller to detect a crash situation and decide whether the airbag shall be fired or not (output fire).

The outputs of sensors S_1 and S_2 are analogue signals that indicate the speed and direction of the vehicle's movement (with a maximum indicated speed of $100\frac{Km}{h}$) in the range of -10 to +10.

- If the sensor output is +10, it means the vehicle is moving forward at a speed of 100 $\frac{Km}{h}$.

- If the sensor output is -10, it means the vehicle is moving backward at a speed of 100 $\frac{Km}{h}$.

The side sensors also produce an analogue output between 0 and 1, which indicates the probability and risk of an impact to the vehicle.

- A value of 1 represents the highest probability of risk.
- while a value of 0 represents the lowest probability of risk from the sides.

The main FSM design is shown in Fig.8. This FSM is designed at the highest level of abstraction, and the model details are ignored at this level. In the next step, each of these two states becomes a separate FSM with a hierarchical and cascade structure. We will explain each of them in the following sections.

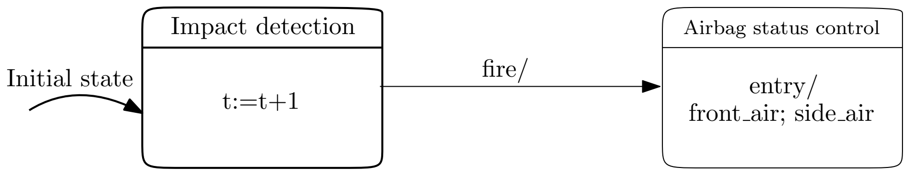

Figure 8: Main state machine of the airbag system

While the airbag may ensure passenger safety in crash situations, its accidental activation is harmful in situations, when no crash is present but indicated by erroneous sensor data. Therefore, certain safety mechanisms have to be applied to guarantee (up to a certain degree of confidence) that the airbag is only fired, if a real crash situation is present. Additionally, defect sensors should be recognised and notified (output `defect`). The state machine in Fig. 9 models the functionality ensuring the safe operation of the airbag controller.

4.3 Semantics of Model

The system reads the sensor values S_1, S_2, S_3 and S_4 cyclically on every rising and falling edge of input $t \in [0, 1]$. each sensor values are checked for plausibility. The sensor values are considered plausible, if the value of sensor one (S_1) does not exceed or drop below the value of other sensors (S_2, S_3, S_4) by more than 5%, i.e. $S_1 \in [0.95{\cdot}S_2, 1.05{\cdot}s2]$ and $S_1 \in [0.95{\cdot}S_3, 1.05{\cdot}s3]$ and $S_1 \in [0.95 \cdot S_4, 1.05 \cdot s4]$. If the sensor values are plausible and an acceleration greater than 5 is measured in 5 consecutive cycles, the airbag is fired. This is done by setting output variable fire to 1. If instead the sensor values are implausible, internal variable `error_ctr` is incremented. This variable holds the number of implausible measurements, and if it reaches a value equal to 5, the output variable defect is set to 1, causing a shutdown of the complete airbag system and activating the service lamp to indicate a sensor defect of the airbag. After at least 5 consecutive cycles with plausible sensor values, the internal variable `error_ctr` is reset.

After the `fire` command is issued, it is time to determine which airbag (driver's front or side) should be inflated and to what extent, based on the output of the distance sensor located on the steering wheel.

To determine this, we design another sub-FSM that is responsible for controlling the two airbags in the vehicle. This FSM is arranged in a cascade with the main FSM and takes its input, fire, from the previous FSM. The designed FSM is shown in Fig.10.

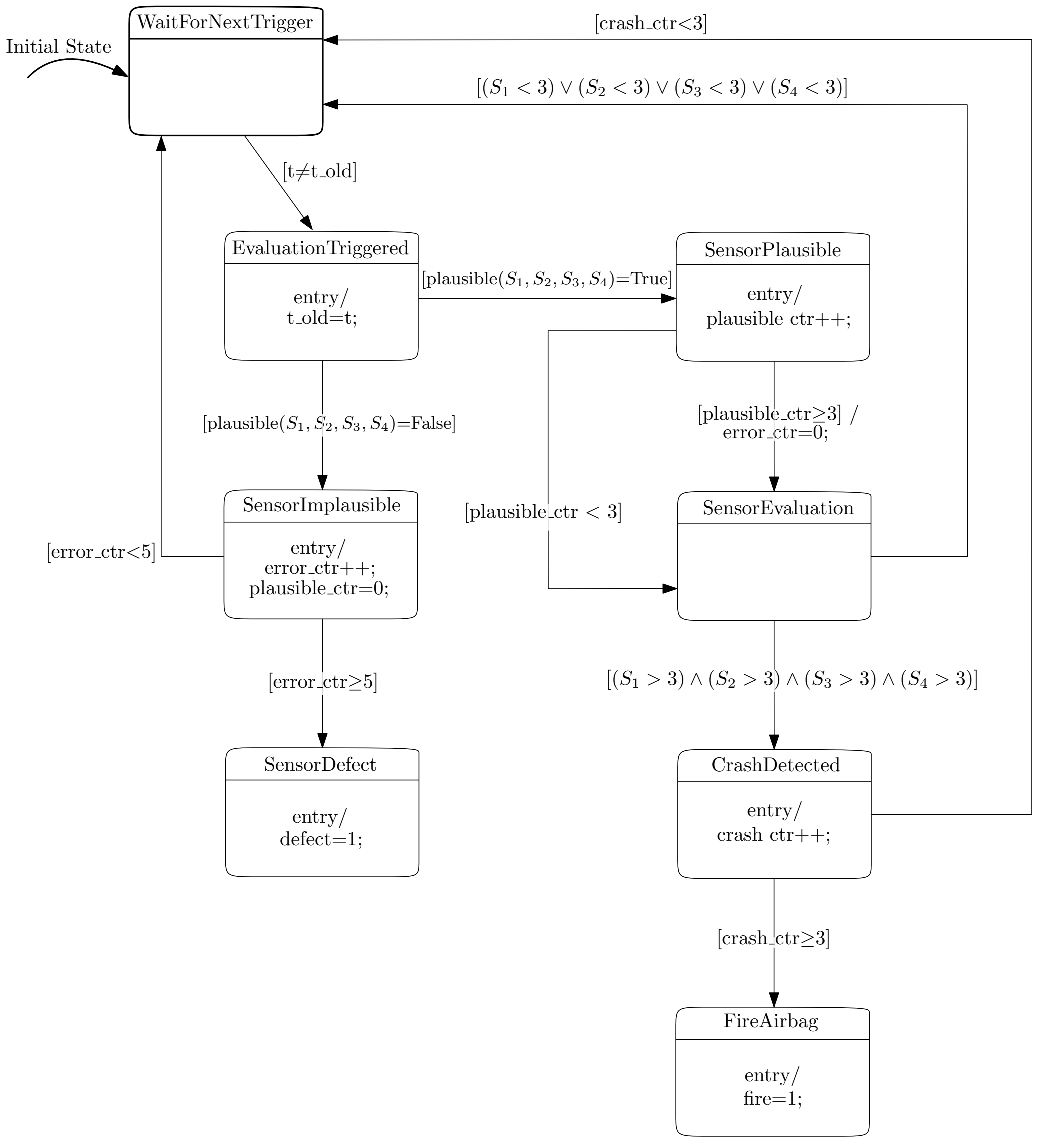

Figure 9: Impact detection state machine

4.4 implementation

I would like to use STM32-Microcontrollers for the implementation phase of our airbag control system. Here are the reasons for this choice:

- **Real-Time Responsiveness:** STM32 microcontrollers are equipped with a sophisticated nested vector interrupt controller (NVIC), which ensures minimal latency and high-priority handling of critical events such as collision detection. This is essential for meeting the real-time requirements of our airbag system.

- **Concurrency:** STM32 microcontrollers support concurrent processing through multiple interrupt sources, allowing simultaneous handling of sensor inputs and actuator controls. This capability is crucial for the efficient operation of our system, which relies on data from various sensors to make quick decisions.

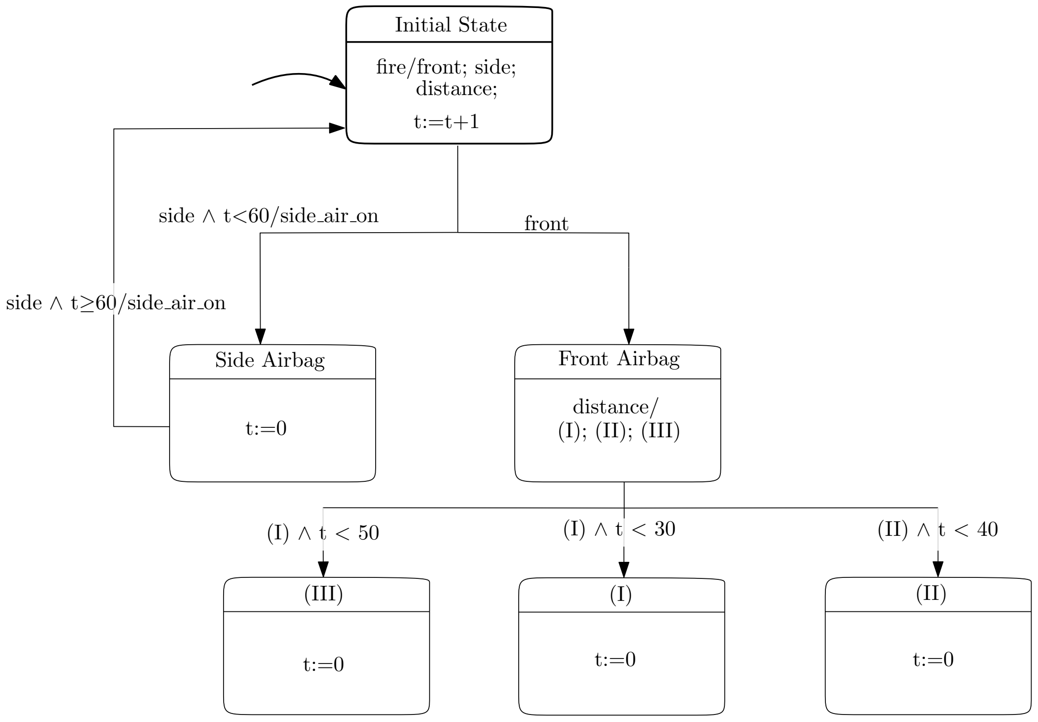

Figure 10: Airbag control state machine

- **Efficiency:** These microcontrollers are designed for low power consumption, which is beneficial for automotive applications. By using interrupts, the processor remains active only when necessary, saving energy and improving overall system efficiency.
- **Hardware Integration:** STM32 microcontrollers come with integrated peripherals such as ADCs (Analog-to-Digital Converters), timers, and communication interfaces (I2C, SPI, UART). These peripherals are essential for interfacing with the sensors and actuators in our airbag system, reducing the need for external components and simplifying the design.
- **Scalability:** The STM32 family offers a wide range of microcontrollers with varying capabilities. This allows us to scale our design based on the complexity and additional features required, providing flexibility for future enhancements.
- **Development Ecosystem:** STM32 microcontrollers are supported by a comprehensive set of development tools, such as STM32CubeMX and STM32CubeIDE, which simplify the configuration and development process. Additionally, the extensive documentation and large community support available for STM32 microcontrollers will help in troubleshooting and optimizing our system.

Given these advantages, I believe that STM32 microcontrollers are the best choice for the implementation phase of our airbag control system. They provide the necessary real-time performance, efficiency, and hardware integration to meet our project requirements effectively.

4.5 WCET Estimation

4.6 Real-time Checking

5 Elevator System

5.1 Project Description

You have probably had a frustrating experience using the elevator in the east wing of the faculty. The goal in designing this elevator was that by pressing a button again (either on the floors or in the cabin), the command to move to that floor would be canceled. However, the timing of this second press (whether before starting to move towards that floor or during the movement) was not properly considered. Additionally, the designer intended that if the elevator could not move, the call request would be canceled (timeout). Unfortunately, this part of the system does not function correctly in practice either.

5.2 Project Detail

In this part of the project, provide a complete description of the correct behavior of the faculty's elevator using a Petri net. In other words, suppose that while retaining the features of the existing elevator, you want to provide an accurate description to fix its current issues. You can base your description on a simple elevator system using a Petri net that you have seen in the course. However, keep in mind that the number of floors in this problem is known (4 floors in the faculty), and the current features of the elevator (including cancellation by pressing again and timeout) must remain. For simplicity, there is no need to perform a qualitative analysis and evaluation of the description. However, if you do perform this analysis and evaluation, you will receive additional points.

Note: This provides a good opportunity for students who did not manage to earn good marks in other parts of the course to make up for this weakness through the project's bonus sections.

References

[1] Mild cognitive impairment (mci). https://www.mayoclinic.org/diseases-conditions/mild-cognitive-impairment/symptoms-causes/syc-20354578#:~:text=Overview,mental%20function%20has%20%22slipped.%22. Last Reviewed: Jan. 18, 2023.

[2] Neural oscillations – interpreting eeg frequency bands. https://imotions.com/blog/learning/best-practice/neural-oscillations/.

[3] Neurodegenerative diseases. https://www.niehs.nih.gov/research/supported/health/neurodegenerative/index.cfm. Last Reviewed: June 09, 2022.

[4] Marin C, Vilas D, Langdon C, Alobid I, López-Chacón M, Haehner A, Hummel T, and Mullol J. Olfactory dysfunction in neurodegenerative diseases. In *Curr Allergy Asthma Rep*, volume 18, 2018 Jun 15.

[5] Sedghizadeh MJ, Hojjati H, Ezzatdoost K, Aghajan H, Vahabi Z, and Tarighatnia H. Olfactory response as a marker for alzheimer's disease: Evidence from perceptual and frontal lobe oscillation coherence deficit. *PLOS One*, 15(12), December 15, 2020.

[6] T.T.K. Munia and S. Aviyente. Time-frequency based phase-amplitude coupling measure for neuronal oscillations. In *Scientific Reports 9, 12441*, 27 August 2019.

[7] Mohammad Javad Sedghizadeh, Hamid Aghajan, and Zahra Vahabi. Brain electrophysiological recording during olfactory stimulation in mild cognitive impairment and alzheimer disease patients: An eeg dataset. *Data in Brief*, 48:109289, 2023.

[8] Cécilia Tremblay and Johannes Frasnelli. Olfactory and Trigeminal Systems Interact in the Periphery. *Chemical Senses*, 43(8):611–616, 07 2018.