Study and modeling of the supervisory control system for autonomous ground vehicles

Carlos F. de P. Perché[*]
Autonomous Mobility Laboratory
FEM, Unicamp, 13083–860
Campinas, SP, Brasil
cfpp@fem.unicamp.br

Janito Vaqueiro Ferreira
Autonomous Mobility Laboratory
FEM, Unicamp, 13083–860
Campinas, SP, Brasil
janito@fem.unicamp.br

Olmer García Bedoya
Autonomous Mobility Laboratory
FEM, Unicamp, 13083–860
Campinas, SP, Brasil
olmerg@fem.unicamp.br

ABSTRACT

In this article, we present the study and modeling of an architecture for the component responsible for operational supervision over ground autonomous vehicles and a preliminary proposal for its implementation based on formal approaches of discrete event systems and supervisory control theory. Under the proposed architecture, the supervisor control is achieved through a uniform structure of generic transitions modeled by finite state machines for the systems present in the vehicle. This structure aims to make deterministic system as a whole, facilitating the fulfillment of the requirements for the implementation of the operations performed.

This paper proposes a solution that enables the coordination of the components which make up ground autonomous vehicles both behavioral and operational level, performing the monitoring of their software task.

CCS Concepts

•**Computer systems organization** → **Robotic autonomy;** *Real-time systems;* Distributed architectures; •**Software and its engineering** → *Software organization and properties;*

Keywords

Unmanned land vehicle, supervisory control, finite state machine, critical real-time system, discrete event system

1. INTRODUCTION

Autonomous land vehicles need to be equipped with a variety of components for processing data, communication, actuation, sensing, etc. The complexity of these components has grown and increasingly they perform their duties independently. Once the components are properly integrated and coordinated, they can make autonomous possible navigation guiding the vehicle with minimal human intervention [11].

Achieve autonomous behavior in a land vehicle still remains a challenging issue, mainly due to the complexity of the operating environment presents. The Earth's surface as well as climatic reasons, has a large variety of features that make up the manual navigation difficult. These conditions significantly interfere with the performance of land vehicles [7].

The solution for such problems require require not only the use of sophisticated components for processing, communication, actuation and sensing, but also a good strategy for supervisory control that can enable the components to use the best of their ability, without significant degradation in performance of the system [6]. The main function of a supervisory system is the monitoring and coordination of activities running on components so that the vehicle meets certain goals, which may range from moving from one place to another as fast as possible, or simply follow another vehicle ahead at a safe distance.

Also, we must take into account that this type of application is characterized as a critical real-time system [25], where alternatives to ensure that certain tasks are fulfilled in a specified time period, otherwise severe failures will occur, making the project unfeasible [9].

This work presents the proposal of the supervisory control architecture for the Intelligent Vehicle of the Autonomous Mobility Lab (VILMA) [15] and its development process, aimed at obtaining autonomous operations in land environments, taking into account the requirements for critical real-time systems. Under the proposed architecture, we adopted a uniform interface that should be incorporated into systems that make up the vehicle as well as a platform that enables the development of the project. The interface adopted represents a behavioral set, wherein each component present in the vehicle architecture can have their tasks managed by the supervisor, enabling the operations performed by the vehicle have to deterministic character.

The rest of the paper is organized as follows. Section 2 describes the concepts and methodologies used to the architecture of supervision and lists the goals and desired requirements. Section 3 describes the general behavior of VILMA autonomous vehicle, shows the modules and components present in vehicle architecture, describes the state transitions structure for the tasks to be performed on a component, where the supervisory control strategies act according to expected behavior and discusses a possible methodology for

[*]Bolsista PEC/PG CAPES/CNPq-Brazil

EATIS 2016 Cartagena, Colombia
© 2015 ACM. ISBN 978-1-4503-4043-4.
DOI: XX.XXX/XXX_X

a formal synthesis of control for the proposed architecture. Section 4 presents the software elements used in the architecture development platform. Section 5 discusses a preliminary description to implement the supervisory control architecture. Section 6 concludes this paper.

2. ARCHITECTURE REQUIREMENTS

According to [23], autonomous systems must have three basic operations: which are perception, location, and environment mapping; path planning and movements control. Typically a unmanned ground vehicle (UGV) has three operating modes: manual, autonomous and collaborative mode, the latter consists of ADAS (Advanced Driver Asssited Systems).

The advancement of research involving UGV has resulted in an increased level of complexity and amount of data in their specific sensors and systems, creating the need to distribute information processing in various embedded computers that communicate synchronously. Thus, a appropriate supervision of vehicle modules is essential to a modular design with deterministic characteristics. This management should perform the control according to the needs of each task activities and ensure the vehicle behavior based on rules that prioritize safe operations.

Formal techniques such as finite state machines (FSM) to model the behavior of deterministic systems and get your supervisory control has been used in various areas including network communication, traffic control systems, assembly lines, and so on [19] [4]. This same technique has been presented by researchers in related work, such as robotic applications and autonomous vehicle systems. [10] uses a set of FSMs to describe the robot collaborative behavior to football matches. [8] models the operations of an autonomous robot using a hybrid system of automata, on which the changes of their behavior is modeled by FSMs containing discrete states which correspond to distinct behaviors through a continuous model. [24] presents an application where their behavior is based on modeling state machines for a modular navigation system in a vehicle that operates under tunnels.

The described works aim to model the behavior between the switching of activities to accomplish its tasks, making the vehicle or the robot's behavior strictly deterministic. However, the issue about dynamic reconfiguration of the systems to that they can work in changing environments is not addressed. In addition, these studies do not take into account the problem of coordination among the components that make up the architecture of their robotic systems. While there is great progress in addressing research methods of perception, vision, positioning, navigation, trajectory planning, control, etc., research and applications dealing supervision platforms for autonomous systems are not at the same level. [5] presents an architecture and preliminary implementation for the supervisory system of UGV called Ulysses.

Having as the main objective of this study, the development of a supervisory system called "Vehicle Integrity System"(VIS), to be part of the VILMA project, responsible for checking and control vehicle operating modes, monitor the states of the other modules, supervise and coordinate systems in different operating modes, propose decision-making in case of failure of any components, perform control and message log for easier system maintenance and debugging during the project development phase. To meet the goal of the work, the proposed architecture must ensure the following requirements:

- propose an architecture with deterministic characteristics for the development of critical real-time systems;
- standardize the communication interface between the systems present in the vehicle architecture;
- provide a supervisory system which monitors the current state of other vehicle components;
- the supervisory system should collect information about the other components, which should be reported to the user interface of the vehicle;
- in addition to the behavioral control and operational management, the supervisory system must monitor the health of the other components that implement the interface provided by VIS;

3. PROPOSED ARCHITECTURE

This paper proposes a system for supervisory control using the conventional concept of modeling FSM taking into account the problem of dynamic reconfiguration for the VILMA vehicle as a potential development platform to other research developed in the Autonomous Mobility Laboratory (LMA). The formal concepts for modeling FSM and the control of discrete event systems (DES) to the supervisory system are adopted to perform the coordination of the other modules, exchange vehicle operating mode, monitoring the integrity of systems and management of the operating states of the modules.

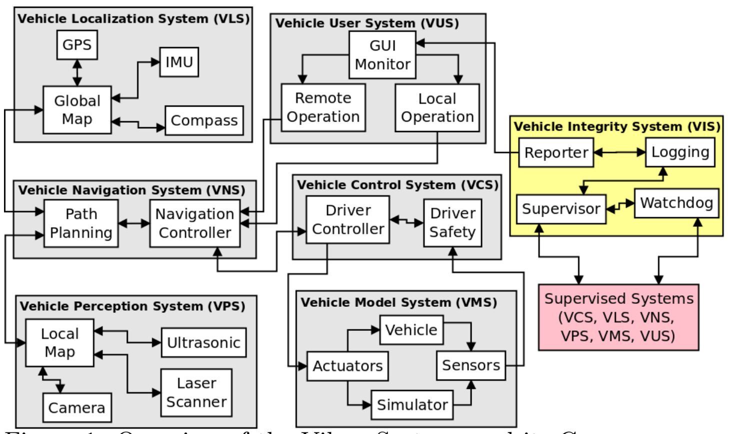
The supervisory control is able to manage only what the other systems are reporting to him, and to do this, there needs to have a common interface between modules in the vehicle, which in case of a malfunction or the occurrence of unexpected behavior the supervisory system can perform the coordinating of the other systems to react to a potential situation. The proposed platform may be considered as a copilot type from the UGV that tells the vehicle as their activities must be organized, which configuration to use at a particular time, while displaying failures detected on a specific system, etc.

3.1 Operational Modes

The Autonomous Mobility Laboratory (LMA) has already set some basic requirements for the Vilma project. One of the Vilma design specifications is that it should perform tasks in manual, cooperative and autonomous modes of operation. In the manual mode, the vehicle is driven by a human operator in a conventional manner. In the cooperative mode, both drivers work in harmony when tasks are not in conflict, and in contraries cases, depending on a risk analysis, the embedded system assumes the task and does not allow the pilot to has the command. In the autonomous mode, the vehicle is driven by a computer control system that receives data from your sensors, processes the information collected and acts directly on its mechanical actuators. Once in autonomous mode, the vehicle is expected to perform tasks without direct human intervention.

3.2 Modules and Components

The development of systems with a modular architecture aims to improve quality of the software, decrease the time required to integrate new features and speed up the delivery of the work. For this reason, the first architecture proposal to the VILMA project is designed to have basically composed of seven systems, also called modules. Each system consists of components that perform specific tasks. The modules and some of the main components present in vehicle architecture can be seen in the Figure 1.



Vehicle Localization System (VLS)
GPS
IMU
Global Map
Compass
Vehicle User System (VUS)
GUI Monitor
Remote Operation
Local Operation
Vehicle Integrity System (VIS)
Reporter
Logging
Supervisor
Watchdog
Vehicle Navigation System (VNS)
Path Planning
Navigation Controller
Vehicle Control System (VCS)
Driver Controller
Driver Safety
Supervised Systems (VCS, VLS, VNS, VPS, VMS, VUS)
Vehicle Perception System (VPS)
Local Map
Ultrasonic
Laser Scanner
Camera
Vehicle Model System (VMS)
Vehicle
Actuators
Sensors
Simulator

Figure 1: Overview of the Vilma Systems and its Components.

The list below introduces an overview of the modules and its main functions to the VILMA project.

- *Vehicle Model System* (VMS): It is the system that represents the vehicle, both real (physical) and simulated (model). It implements the interface between the actuators and the low-level sensors, in which the actuators receive Vehicle Control System (VCS) commands to perform tasks and respective generated signals to the VCS;

- *Vehicle Control System* (VCS): Module responsible for controlling the actuation of the vehicle and ensure the safety of those involved in the operating environment. Its main function is to provide commands to control speed, acceleration and vehicle direction. The VCS generates low-level signals that trigger the electromechanical system, telling the desired speed, the acceleration needed and the angle of direction. Moreover the VCS system, has a cyclical feedback communication with the VNS system to ensure that the speed and direction commands are both running correctly, taking into account environmental factors such as wheel slippage and others.

- *Vehicle User System* (VUS): The VUS module enables the interaction of the driver to the car, both in local mode and in remote mode, enabled by using cameras mounted on the vehicle to simulate the driver's view. The operator controls the movement of the car through a remote control. Furthermore the system should provide a graphical interface enabling the driver to know the state of the vehicle in the real time as well as provide alerts and diagnostics for debugging during development stage.

- *Vehicle Perception System* (VPS): The VPS system uses several instruments such as laser scanners, cameras, ultrasonic sensors, etc. to construct digital maps of the environment around the vehicle, it provides the local perception of the vehicle. The generated local map is sent to the VNS system, which performs the planning of the trajectory and the generation of the movement commands to the VCS system.

- *Vehicle Localization System* (VLS): Just like the VPS system, the VLS module uses several instruments such as GPS, IMU, electronic compasses and other sensors that make possible to extract the precise location on a global scale. It also gives the digital map information for the VNS system to be able perform of path planning and generation of the movement commands to the VCS system.

- *Vehicle Navigation System* (VNS): It is responsible for generating commands to the vehicle navigation in order to perform the task specified by the operator. Furthermore it must to be able to carry out the planning of the path to the desired location, calculate the speed and its heading, generating the necessary information in order to the VCS system to perform the tasks safely.

- *Vehicle Integrity System* (VIS): The VIS module is responsible to ensure the integrity of the system that operates the vehicle while it is running. Its main function is to monitor and coordinate the activities of the modules to achieve the expected behavior of the vehicle. The VIS must also perform routing of data between the other modules, registers and publishes the information which contains the status of the systems. In a nutshell, the VIS acts as a digital copilot. Further description of the supervisory system will be presented in this paper.

3.3 States and Transitions for Supervisory Control

In order to have the proper coordination between the activities performed by the present systems in the vehicle architecture, it is necessary that the components have a uniform behavior in relation to the VIS system. Thus, each component needs to implement a common interface, modeled by a set of states and transitions that are recognized by the supervisory system, enabling him to manage the activities being carried out, perform routing of messages required for debugging and the data storage, etc. The coordination of the modules is responsible for determining the sequence of activities and the conditions that a system must have so that it can perform its specialties.

Under the perspective of supervisory control, the states present in the common interface of the systems are: PowerOn, Standby, Ready, Working, InternalError, Emergency, Shutdown and PowerOff. Each component must contain these states. For a certain component, the meaning of the states and the general behavior of the vehicle can be summarized as follows.

- *PowerOn* The driver starts the vehicle, at this moment the engine is started but the car remains parked awaiting the instructions of the driver.

- *Standby* The component comes to be started and the vehicle remains stationary. In this state, the system is not linked to an autonomous operation mode, it remains awaiting the instructions from the supervisory system about the operating mode that it will work or the instruction to return to the manual mode;

- *Ready*, In this state the component is linked to an autonomous operation mode but not yet operated in this mode. The reason for the separation between Standby and Ready states is due to the fact that the article focuses on the general behavior of the vehicle, clearly denoting the temporal separation of the states which the vehicle can return to manual operation and the state which it is ready to operate in standalone mode. The temporal separation of the states is required for the synchronization of activities on the components before they enters in the state Working.

- *Working*, In this state the component is operating in a autonomous mode and the vehicle is moving. Now, the algorithm that performs specific functions of the component begins to run to produce its expected behavior.

- *InternalError* e *Emergency* The component performs tasks relevant to handling exceptions raised by your specific algorithm or handling of emergency that might compromise the overall operation of the vehicle.

- *Shutdown*, The component executes a predefined set of routines preparing the vehicle to be turned off, bringing the system to the state PowerOff;

- *PowerOff*, The system is turned off and no electric signal is supplied to the component.

Transitions between states are a set of activities that lead the components from one state to another. In order to a transition occurs it must be triggered by an event. Once invoked, the activities associated with the transition must be performed successfully so that the component get into a new state. The diagram of the common states and transitions to the supervisory system can be seen in Figure 2. Specifications that represent constraints of the activities for each module may also be modeled by automatas to produce a composite specification model. With the obtained model, based on the theory of the DES we can get a formal model for supervisory control, as discussed in [20].

The resultant specifications, described by an FSM form a chain containing all activities and sequences of allowed transitions to ensure that all specifications are fulfilled, i.e., when any of the modules need to perform a transition between states of the FSM that describes its behavior. For example, to enter the Working state or leave the Working state and go to the Standby state or when a module is performing the treatment of some exception in InternalError, all transitions of related states must check their restrictions and obtain permission from the supervisor module before the transition occurs.

Although each component contains the set of states and transitions required, the question of the activities involved in the states and transitions is transparent to the VIS. In other words, the VIS not internally knows the complexity of the tasks performed within the states or during the transitions from one component and there is no restriction that it only has these states and transitions. The activities involved in the transitions of each component may differ from those present in other components.

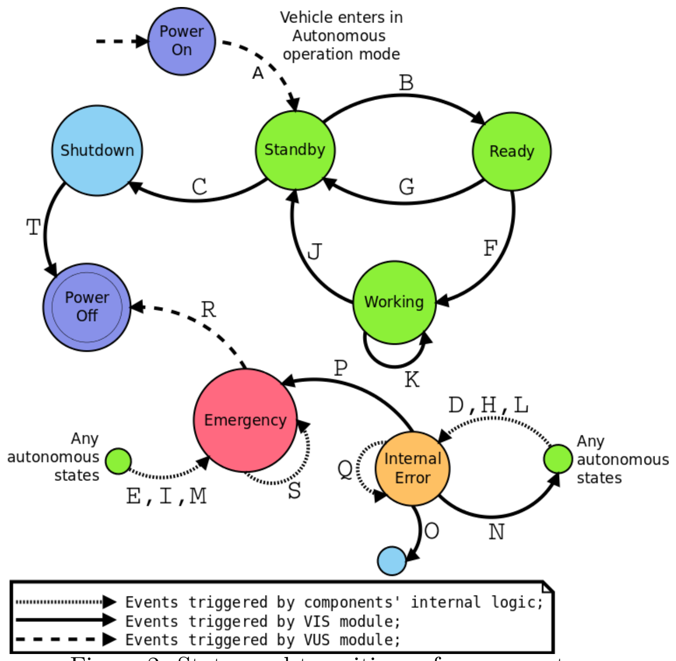

Power On
Vehicle enters in Autonomous operation mode
A
B
Shutdown
Standby
Ready
C
G
T
J
F
Working
Power Off
R
K
P
D,H,L
Emergency
Any autonomous states
Internal Error
Any autonomous states
Q
E,I,M
S
O
N
Events triggered by components' internal logic;
Events triggered by VIS module;
Events triggered by VUS module;

Figure 2: States and transitions of components.

3.4 VIS on Formal Approach

After modeling pattern that determines the behavior of the modules in terms of states and transitions, it is possible to synthesize logic of the VIS by formal techniques, for example, the supervisory control theory to discrete event systems [20]. The application of this technique requires that the states and transitions of the modules are expressed by sets of automata (called models), while the desired behavior of the vehicle is expressed by other sets of automata (called specifications). For example, the behavior of a module, according to the specifications of the VIS as illustrated in Figure 2, can be represented by an automata $G = (Q, \Sigma, \delta, q_0, Q_m)$, where, Q is a set of states, e.g., $Q = \{\mathit{PowerOn},\ \mathit{Standby},\ \mathit{Ready},\ \mathit{Working},\ \mathit{InternalError},\ \mathit{Emergency},\ \mathit{Shutdown},\ \mathit{PowerOff}\}$, Σ is a set of transitions, e.g., $\Sigma = \{A,B,C,D,E,F,G,H,I,J,K,L,M,N,O,P,Q,R,S,T\}$, $\delta\colon Q \times \Sigma \to Q$ is a transition function that specifies the new state after a transition from a given state, as presented in Table 1, q_0 is the initial state of the component and $Q_m \subseteq Q$ is the set of states that indicate the completion of certain desired sequence of tasks.

Since each module of the system is modeled by an automaton G_i, a composite model of the overall behavior of the vehicle can be build using the procedure called "synchronous product", represented by the symbol $\|$ [18]. The result of the synchronous product over all G_i is a new automaton that represents the "free"behavior (uncontrolled) of simultaneous activities of the components, i.e., $G = G_1 \| G_2 \| G_3 \| ... \| G_n$.

Then the desired deterministic behavior of the vehicle may be imposed on the behavior "free"to obtain strategies for supervisory control expected by the VIS system. Formally, such behavior is also expressed in automata, called specifications automata. By imposing the specifications of the model composite of all vehicle components, a sequential set of transitions may be obtained to be run by the VIS, ensuring that the vehicle behavior control meets the specifications.

By using this approach, it is possible to synthesize the

$\delta(oldstate, transition) = newstate$		
Old state	transition	new state
power-on	A	standby
standby	B	ready
	C	shutdown
	D	internal error
	E	emergency
ready	F	working
	G	standby
	H	internal error
	I	emergency
working	J	standby
	K	working
	L	internal error
	M	emergency
internal error	N	standby
	O	shutdown
	P	emergency
	Q	internal error
emergency	R	power-off
	S	emergency
shutdown	T	power-off

Table 1: Transition function δ of the FSM model.

supervisory control strategies that allow the vehicle to deal with different situations. For instance, suppose that, due to timing synchronization constraints, the VCS can only enter the Working state after VNS is in Working state. This can be considered as a requirement to comprise the desired behavior for the initialization of a task. This requirement can be expressed as a specification S_i, as shown in Figure 3, which the cyclic loops represent other possible transitions that can occur in the system.

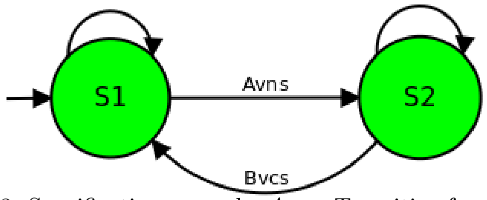


Figure 3: Specification example, A_{vns}: Transition from *Ready* state to *Working* in the *Vehicle Navigation System* module, B_{vcs}: Transition from *Ready* state to *Working* in the *Vehicle Control System* module.

The supervisory control that guarantees the behavior of the modules as a whole, and meets the specification imposed by S_i can be obtained, among other ways, by the algorithm *SUPCON* [26]. Their function is to generate sequences of events as another automaton V, e.g., $V = SUPCON(G, S_i)$, that can be implemented to ensure the deterministic behavior of the entire vehicle system.

By expressing all the requirements on vehicle behavior in automaton specifications, it becomes possible to develop a sophisticated method of control, which when implemented by the VIS will ensure that the vehicle meets all the requirements imposed by other modules. This formal approach allows the systematic and proper development for supervisory control strategies, and allows the introduction of ”programmed artificial intelligence”to the general behavior of the vehicle, and hence obtain a high degree of autonomy.

4. ARCHITECTURE COMPONENTS

In order to meet the requirements presented in the previous sections, it is usually necessary the application to be developed on a platform that has a sophisticated hardware and software environment to ensure deterministic criteria of a critical system real time. However, this paper focuses its efforts to provide an architecture focused on technologies and methodologies based purely on software, able to perform critical tasks in real time, reducing the cost and time for prototyping the VILMA project. The following is a brief description of the components that make up the proposed architecture.

Figure 4 shows the layout of the main elements that make up the proposed development environment.

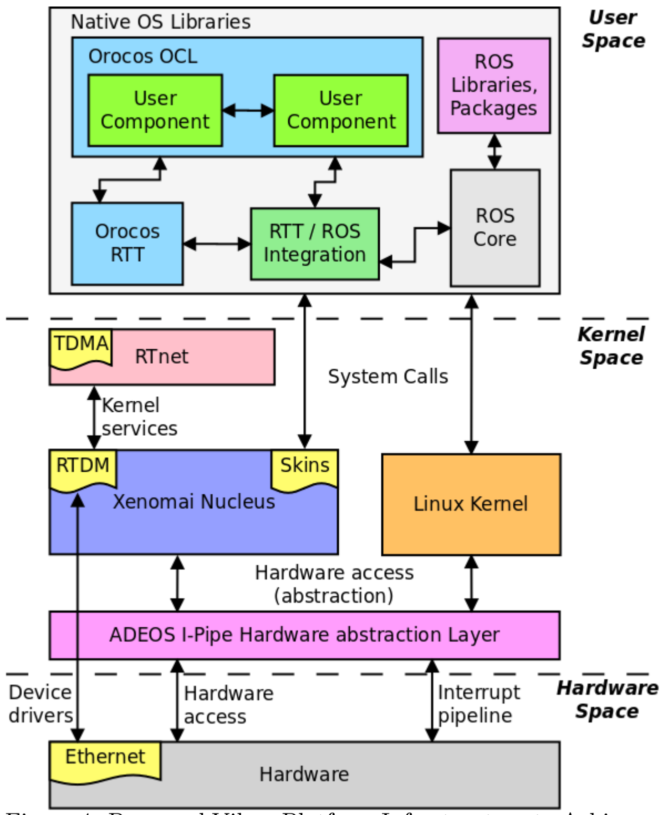


Figure 4: Proposed Vilma Platform Infrastructure to Achieve the Requirements.

4.1 Linux

Linux is an operating system. It is the software on a computer that enables applications and the computer operator to access the devices on the computer to perform desired functions. The operating system (OS) relays instructions from an application to, for instance, the computer's processor. The processor performs the instructed task, then sends the results back to the application via the operating system [1].

Linux, which began its existence as a server OS and has become useful as a desktop OS, can also be used on all of these devices. “From wristwatches to supercomputers”, is the popular description of Linux’ capabilities. [14]

4.2 Xenomai

Xenomai [27] is a real-time development framework cooperating with the Linux kernel, in order to provide a pervasive, interface-agnostic, hard real-time support to user-space applications, seamlessly integrated into the GNU/Linux environment. The Xenomai drivers has the RTDM (Real-Time Driver Model) which composes the following modules: CAN Devices, Real-time IPC Devices, Serial Devices, and Testing Devices [12].

4.3 RTnet

With Ethernet, the communications are not deterministic because of the collision which can occur between several host on a network with a hub, or because of the unknown latency in the case of the switch. The deterministic communication is not allowed by the Ethernet protocol because of the possibility of collisions which can occurs and supported by the mechanism CSMA/CD (Carrier Sense Multiple Access/Collision Detection) [13].

RTnet is a protocol stack which run between the Ethernet layer and the application layer (or IP layer) with hard real-time requirements [22]. It aims, through the use of time intervals (time-slots), is to make deterministic communication, by disabling the collision detection CSMA/CD, and prevent buffering packet in the network.

RTnet is a software developed to run on Linux kernel with RTAI or Xenomai real-time extension. It exploits the real time kernel extension to ensure the determinism on the communication stack. In this aim, all the instructions related to this protocol makes use real time kernel functions rather than those of Linux, which bound latencies to the execution times and latencies of interruptions which provide deterministic’s communication [28].

4.4 ROS

The robot software is base on the ROS (Robot Operating System) [21], an meta-operating system for robots. It provides the services like hardware abstraction, low-level device control, and message-passing between processes. ROS is composed by the operating system commands that manage the packages, and a suite of user contributed packages (organized into sets called stacks) that implement functionality such as simultaneous localization and mapping, planning, perception, simulation etc [16].

4.5 OROCOS

Orocos [2] is the acronym of the Open Robot Control Software project. The project’s aim is to develop a general purpose, free software, and modular framework for robot and machine control. The Orocos project supports 4 C++ libraries [3]: the Real-Time Toolkit, the Kinematics and Dynamics Library, the Bayesian Filtering Library and the Orocos Component Library. The Orocos Real-Time Toolkit (RTT) is not an application in itself, but it provides the infrastructure and the functionalities to build robotics applications in C++. The emphasis is on real-time, on-line interactive and component based applications.

The integration between the OROCOS framework and the ROS system can be performed using *rtt_ros_integration* [17]. Briefly, ROS is used to create a communication interface between the components and the OROCOS system performs an encapsulation of the real-time kernel API (Xenomai), enabling the definition and implementation of real-time tasks or not.

5. VIS DESIGN

The supervisory control system performs management and collecting information from other vehicle components through a common interface that must be implemented by each of the components. Thus, it should provide to other researchers of the LMA means to achieve this, such that they can integrate their projects with the supervisory system enabling the management of their systems in a ”transparent”way. Furthermore, after the integration with supervisory, the consumption of system resources need to be as little as possible so as not to impact the functional performance of the projects.

The supervisor should also ensure the integrity of the components required to the execution of the tasks during autonomous navigation, because of it, a message will be requested periodically to each component present in the vehicle, with the purpose of ensuring the presence of a particular component and its proper operation.

5.1 A General Description of Modules Behavior

Once requested by the driver that the vehicle operates in autonomous mode, the VIS system should request to all managed components to perform their startup procedure, this may include sensors check, test the actuators, check the presence of another component in the network, etc. Then if none of the components detects failures during its startup, they must notify the supervisory system that are initialized and ready to operate in autonomous mode, so the VIS requests all components to perform the switch to the Standby state and stay waiting for new instructions.

Once in Standby, the VIS can instruct the components go to the Shutdown state in case of failures detected by any component, or to prepare for a specific mode of operation. The components can switch to the Ready state if they have done their configuration processes that ensure its activation to the operation mode specified, as instructed by the VIS. Once the process is completed, the VIS can instruct the components to begin operating, bringing the components to the Working state. Being in the Working state , the VIS can instruct the components to return to the Standby state if the exchange of the specific mode of operation is required.

Operational errors, represented by the InternalError state can occur when a component is in any of the following states: Standby, Ready and Working. Falls exclusively to the component deciding what constitutes an operational error. When an exception occurs, the VIS can instruct the component that detected the problem to perform both, either the restart procedure or its shutdown. If rebooted, the component will return to the Standby state.

Emergencies, represented by the Exception state, can occur when a component is in any of the following states: Standby, Ready, Working or InternalError. The supervisory handles an emergency in two ways, according to the seriousness of the situation, or through of the decision of operator which can instruct the component to be rebooted and return it to the Standby state after the emergency is resolved, or choose to shutdown of the system.

The structure of states and transitions shown in Figure 2 must be implemented by the VIS and for each component as an event-driven software process (event-driven). The VIS module interacts with other components based on this structure to perform supervision of the vehicle's activities. These states and transitions define the behavior of the components that are managed by the VIS. However, there is no restriction for the components to have their own states and transitions. One component can have as many states and transitions are necessary for its specific activities, the only difference is that these will not be recognized by the VIS.

By adopting the state diagram and transitions presented in Figure 2 as a standard of behavior interface between the VIS and other components of the VILMA project, their internal activities becomes transparent to the supervisory system, ensuring the supervision of their activities without that the VIS needs to interfere in the structure and operations of each individual component.

5.2 Software Processes

The VIS module consists of two software processes. One based on the event-driven process, which has the purpose to provides the coordination of the activities between components signaling the exchange of their states. The other one is based on periodic process, periodic-tasks, for monitoring the integrity of the components present in VILMA architecture.

The event-driven process is responsible for performing four tasks:

1. To manage the switch sequences of vehicle operating modes.
2. To manage the transition between the operational states of the other components.
3. Perform routing of data between the components.
4. Collect informations about the vehicle status and its components.

The periodic process is responsible for continuously monitoring the integrity of the other components at the entire time in which the vehicle is running. The verification of other components is performed through the time-stamped messages, called heartbeat, in a predefined frequency. If the period of arrival heartbeats of a component match the preset value, then the VIS module assumes that the particular component is functioning normally.

If the supervisor does not receive the heartbeat of a component, then the VIS will assume that particular component is not working properly, so the supervisor will instruct all the components to move to the Standby state, to evaluate the possible failure. The periodic process of VIS module need to send a heartbeat message to VNS module so that it can monitor the operational health of the VIS, if the VIS heartbeat messages do not arrive at VNS then it must instruct the system to the emergency procedure.

To illustrate this scenario, assume that the frequency of heartbeat messages to every components (excluding VIS) is set to $1Hz$. This selection should be based on taking into consideration the distance that the vehicle will go through, as soon as the failure of a component is detected by the VIS until the moment when the vehicle is fully stopped. This displacement will be called by braking distance (fail-stop).

Assume that the speed is defined by s in *km/h* and the distance of vehicle stopped by *d meters*. With a frequency of *1Hz* to heartbeats messages, the VIS module will be able to detect a fault in one component for each 1 second. During the check time and under ideal conditions, the vehicle will have traveled a distance of *1000s/3600* meters until its complete stop. Assuming that the needed time required for VIS instructs the system responsible for the actuators that perform the full stop of the vehicle is close to zero, then the fail-stop distance will be *d + (1000s / 3600) meters*. Still as an example, assuming *s = 10km/h*, *d = 4m* and *(1000s/3600) = 2.8m*. Thus the fail-stop distance when the vehicle is moving at speed of *10km/h* will be *4+2.8 = 6.8* meters in ideal conditions.

6. CONCLUSION

The discussion about the supervisory control system that provides a deterministic behavior for other autonomous vehicle systems illustrates how the robustness and security of the VILMA project can be enhanced using the methodologies presented. In this sense, the study and modeling for supervisory control system that composes the proposed architecture for VILMA vehicle development platform and its preliminary implementation in order to meet their goal was introduced, to perform autonomous operations in Earth environment safely.

Through the proposed architecture, it is possible to get the management of the systems that compose the VILMA vehicle by adopting a uniform interface modeled by FSMs using generic transitions to switch states for tasks performed on every vehicle component. The state machine model represents the behavioral set that the components managed by the VIS system will implement, making the system to have a high degree of determinism, i.e., the vehicle will have an expected behavior during the execution of all their tasks.

Furthermore, we showed that this architecture can ease the implementation for more sophisticated control strategies using formal techniques, as in the case of the DES, where the supervisory control based on discrete event methodology can raise the degree of autonomy of the vehicle, and represents a next step in the development of intelligent autonomous vehicle VILMA.

We also proposed a development platform composed of a technological scaffold focusing on the goal of making the work of the LMA research group standardized, consolidating the agility, the low cost for the development of the VILMA prototype and its continuity, since all the tools are well documented and widely spread. These technologies are widely used and consolidated in industrial equipments and research groups.

The integration between the low-level systems (Linux, Xenomai, Ethernet and RTnet), enables the achievement of results within the scope of the critical real-time systems with distributed processing, ensuring the requirements for the viability of the application. The integration between the user-level libraries (ROS and OROCOS), strengthens the idea of flexibility and simplification to the development of researches developed in the LMA, initially this integration is based solely on features for the data flows between the components of this two frameworks.

7. ACKNOWLEDGMENTS

This work has been co-founded by CNPq and CAPES. Garcia O. is sponsor by a Scholarship PEC/PG CAPES/CNPq-Brazil

8. REFERENCES

[1] M. Barabanov. *A linux-based real-time operating system*. PhD thesis, New Mexico Institute of Mining and Technology, 1997.

[2] H. Bruyninckx. Open robot control software: the orocos project. In *Robotics and Automation, 2001. Proceedings 2001 ICRA. IEEE International Conference on*, volume 3, pages 2523–2528 vol.3, 2001.

[3] H. Bruyninckx and P. Soetens. *The Orocos User's Manual: Open RObot COntrol Software 2.7.0*, May 2009.

[4] V. Chandra, S. Mohanty, and R. Kumar. Automated control synthesis for an assembly line using discrete event system control theory. In *American Control Conference, 2001. Proceedings of the 2001*, volume 6, pages 4956–4961 vol.6, 2001.

[5] P. Chen, J. Guzman, T. Ng, A. Poo, and C. Chan. Supervisory control of an unmanned land vehicle. In *Intelligent Control, 2002. Proceedings of the 2002 IEEE International Symposium on*, pages 580–585, 2002.

[6] M. L. Cummings, P. Pina, and J. W. Cr. 1 a metric taxonomy for supervisory control of unmanned vehicles.

[7] B. Donmez and M. L. Cummings. Metric selection for evaluating human supervisory control of unmanned vehicles. In *Proceedings of the 10th Performance Metrics for Intelligent Systems Workshop*, PerMIS '10, pages 14–21, New York, NY, USA, 2010. ACM.

[8] M. Egerstedt, K. Johansson, J. Lygeros, and S. Sastry. Behavior based robotics using regularized hybrid automata. In *Decision and Control, 1999. Proceedings of the 38th IEEE Conference on*, volume 4, pages 3400–3405 vol.4, 1999.

[9] A. Goodloe and L. Pike. Monitoring distributed real-time systems: A survey and future directions. Technical Report NASA/CR-2010-216724, NASA Langley Research Center, July 2010. Available at http://ntrs.nasa.gov/search.jsp?R=278742&id=3&as=false&or=false&qs=Ns%3DArchiveName%257c0%26N%3D4294643047.

[10] G. Gupta, C. Messom, and H. Sng. State transition based supervisory control for a robot soccer system. In *Electronic Design, Test and Applications, 2002. Proceedings. The First IEEE International Workshop on*, pages 338–342, 2002.

[11] M. H. Hebert. *Intelligent Unmanned Ground Vehicles: Autonomous Navigation Research at Carnegie Mellon*. Kluwer Academic Publishers, Norwell, MA, USA, 1997.

[12] J. Kiszka. *The Real-Time Driver Model and First Applications*.

[13] J. Kiszka, B. Wagner, Y. Zhang, and J. Broenink. Rtnet - a flexible hard real-time networking framework. In *ETFA 2005: 10th IEEE Conference on Emerging Technologies and Factory Automation*, pages 449–456. IEEE, 2005.

[14] Linux foundation, 2015.

[15] The LMA website, 2015.

[16] J. M. O'Kane. *A Gentle Introduction to ROS*. Independently published, oct 2013. Available at http://www.cse.sc.edu/~jokane/agitr/.

[17] Orocos. *Orocos-ROS integration libraries and tools*, 2015.

[18] L. Y. Pao and N. Wu. Introduction to discrete event systems, second edition (cassandras, c.g. et al.; 2008) [bookshelf]. *Control Systems, IEEE*, 29(3):108–110, June 2009.

[19] L. Pinzon, H.-M. Hanisch, M. Jafari, and T. Boucher. A comparative study of synthesis methods for discrete event controllers. *Formal Methods in System Design*, 15(2):123–167, 1999.

[20] P. Ramadge and W. Wonham. The control of discrete event systems. *Proceedings of the IEEE*, 77(1):81–98, Jan 1989.

[21] ros.org. *ROS core components*.

[22] RTnet. *RTnet doccumentation*.

[23] R. Siegwart, I. R. Nourbakhsh, and D. Scaramuzza. *Introduction to autonomous mobile robots*. MIT press, 2011.

[24] S. Sircar. Modular navigation strategy for an autonomous mobile robot., 2015.

[25] L. Wang. Real-time software design for safety- and mission-critical systems with high dependability. In *Autotestcon, 2006 IEEE*, pages 479–485, Sept 2006.

[26] W. M. Wonham and P. J. Ramadge. On the supremal controllable sublanguage of a given language. *SIAM Journal on Control and Optimization*, 25(3):637–659, 1987.

[27] Xenomai. *A Tour of the Native API - RevC*, 2006.

[28] Xenomai. *Xenomai with RTnet*, 2006.