IMPROVING THE SAFETY OF AUVs†

Alberto Ortiz*, Julián Proenza*, Guillem Bernat** and Gabriel Oliver*

* Math and Cmptr. Sci. Dpt.
University of the Balearic Islands (Spain)
e-mail: {dmiaor0,dmijpa0,dmigoc0}@ps.uib.es

** Dept. of Computer Science
University of York (UK)
e-mail: bernat@cs.york.ac.uk

Abstract — The cost of Autonomous Underwater Vehicles is generally high. Therefore, safety, defined as the ability of being able to physically retrieve the AUV if an emergency situation arises, should be one of the main concerns in the design of such systems. On a hardware architecture based on the field bus Controller Area Network, we introduce two different safety layers that are *orthogonal* to the rest of the system operation and that can be implemented using low-cost resources. The first layer is called *critical safety layer* (CSL) and reacts autonomously in front of extremely hazardous situations. All the functionality related to detecting these critical situations and triggering the AUV surfacing mechanism is located in a single hardware module, called the Emergency Hardware Module, that presents internal redundancy for fail-safe behaviour. As the AUV could be affected by an internal failure not immediately leading to a critical situation, we introduce a second layer, called *preventive safety layer* (PSL), to avoid waiting until the situation becomes critical. The PSL monitors the system, and on finding a permanent failure it triggers the surfacing mechanism as well. Moreover, the PSL is designed not to interfere with the critical operation of the CSL. These two safety layers can be taken as a set of services provided to the designer that can decide whether to use them or not. On top of these services, the AUV software designer can add all the application specific safety mechanisms that he considers necessary for the particular mission having a minimum guaranteed by the two aforementioned layers.

I. INTRODUCTION

Traditional submarine missions, generally carried out with manned submarines or divers, have inherent dangers for the integrity of the operators and also high set up costs. Autonomous Underwater Vehicles (AUVs) are of great interest in these environments because they do not need a crew, they are more flexible and they require smaller deployment costs. Recent advances in mobile robotics have allowed the design and construction of AUVs which do not require any type of human supervision. In this context, we follow the definition proposed by Kandebo in [1]: "an AUV is an unmanned and untethered underwater vehicle that carries its own power source and that relies on an on-board computer and built-in machine intelligence to execute a mission consisting of a series of preprogrammed instructions, modifiable on-line by data or information gathered by the vehicle sensors".

Regardless the particular features of the vehicle and given the inherent hazards associated with the oceanic medium, it is obvious that the addition of mechanisms to ensure that the vehicle can be physically retrieved is necessary. We consider these features as safety mechanisms.

Safety is generally defined as the probability that a system will either perform its functions correctly or will discontinue them in a manner that does not disrupt the operation of other systems or compromise the safety of any people associated with the system [2]. A system is *fail-safe* when, if it fails, it does in a safe manner. In our context, as there are no people whose integrity depends on the proper behaviour of the AUV, by fail-safe we mean that the system has the ability to reach the surface whenever a hazardous situation arises, either because of software or hardware failure, leaks in the structure, running out of battery or fire, among others.

We take as a starting point an AUV control architecture based on the field bus Controller Area Network (CAN) [3]. It includes a Central Processing Node (CPN) that executes strategic and tactical functions of the system and a set of Sense&Act nodes (SA) that sample the AUV sensors and execute commands from the CPN over the system actuators. On this architecture we will introduce several safety mechanisms that can also be useful on a centralized architecture.

It is of special importance to obtain high levels of safety at a low cost. Moreover, it is very convenient for the AUV designer to be able to add the safety features as *orthogonal* to the rest of the system operation. Orthogonality both reduces the cost of the safety addition and permits the development of the rest of the system independently of the safety aspects.

We introduce two different safety layers. The first one is called *critical safety layer* (CSL) and autonomously reacts in front of extremely hazardous situations, which we

†This work has been partially supported by the GOVERN BALEAR (BOCAIB-16,3/2/98).

will call *critical situations* from now on. All the functionality related to detecting these critical situations and triggering the AUV surfacing mechanism is located in a single hardware module called the Emergency Hardware Module (EHM). The CSL functionality is obtained by simply adding this node to the system without having to make any other change. Special attention has been put on the design of this node to be able to achieve a fail-safe behaviour.

Using only this first layer the system can detect situations like batteries running-out, water flooding, or temperature soaring. However, the AUV could be affected by an internal failure not immediately leading to a critical situation that the CSL would be unable to detect. This generates an unnecessarily long waiting period until the CSL finally triggers the surfacing mechanism because, for instance, the AUV runs out of battery. Focused on these cases, we introduce a second layer called *preventive safety layer* (PSL). An additional requirement for the new layer is not to interfere with the critical operation of the first one.

These two safety layers can be taken as a set of services provided to the designer that can decide either to use them or not. Moreover, the AUV software designer can add all the application specific safety mechanisms that he considers necessary for the specific mission having a minimum guaranteed by the two layers we have introduced.

Related work is presented in the next section, including a more detailed description of the AUV architecture based on the CAN bus and other authors contributions to safety in AUVs. Then, the two layer safety strategy is justified and each of the layers are described separately. Finally, some conclusions and future work are outlined.

II. RELATED WORK

The operation of an autonomous vehicle depends on the information provided by the sensors which are distributed along its structure. As the size of the vehicle increases, the cabling associated becomes a problem in terms of cost, assembling, maintenance and reliability. In these circumstances, using a field bus is particularly adequate due to its simplicity and generally low cost. Moreover, field buses usually have built-in features specially designed to work under hostile environments. For these reasons, field buses have already been used in some AUVs. In [4] the field bus LONWorks is used to build a distributed control architecture for an AUV. LONWorks provides a powerful and easy way to build this type of systems but it is a proprietary protocol that is only implemented in a commercial circuit: the Neuron Chip by Motorola.

As a field bus we have chosen the Controller Area Network (CAN) [5]. CAN allows more flexibility on the election of support circuits (controllers, transceivers, etc.) than other competitors. On the other hand, CAN components have reduced their cost due to mass production for automotive and automation applications. Other authors have

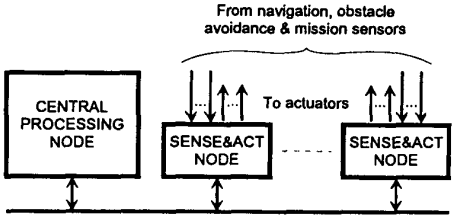

Fig. 1: Referenced AUV hardware architecture.

chosen CAN before as the basic technology for developing underwater applications. By way of example, in [6] CAN is used for internal communications in the manned submarine NAUTILE and in the benthic station MAP2.

In the rest of this paper we will take as a reference the distributed architecture depicted in fig. 1 and first described in [3]. Two different types of nodes can be distinguished in this architecture:

1. a Central Processing Node (CPN), which executes strategic and tactical functions of the system, and

2. a set of Sense&Act nodes (SA) that can sample navigation, obstacle avoidance or even mission sensors, according to the classification in [7], and carry out purely executive functions over system actuators like rudder and bow-plane servos, propellers, thrusters or electrovalves, just to mention some of them.

All the SA nodes correspond to the same type of physical circuit consisting of a microcontroller, A/D converters for analog sensors and a set of CAN support circuits. Depending on proximity criteria (between sensors and SA nodes) and on the load imposed by the different sensors, it would be necessary to incorporate more or less SA nodes[1]. This design approach constitutes an additional source of cost reduction as the system does not include specific nodes for every type of sensor.

The main concern of this work is to obtain safety in AUVs at a low cost. Error detection is an essential step to achieve a safe behaviour in the presence of faults. In a system based on a field bus, when a long latency in the error detection could produce a non safe behaviour, the inclusion of redundancy at different levels is required. The communication protocols of typical field buses already have built-in error detection mechanisms. However, if we want to guarantee the safe behavior of the system, adding redundancy at the nodes themselves is also needed as shown in [8]. Furthermore, it is necessary to distinguish between errors that do compromise the safety of the system from those that do not. This implies the addition of redundancy management and diagnosis software layers,

[1] In fact, mission sensors like a CCD camera will not be managed by an SA node due to its characteristic heavy load. In this case a specialized SA node would be included in the architecture.

that increases both the development cost and the computational overhead.

In the AUV literature there are many contributions that achieve a safety or even reliability improvement using diagnosis software layers [9, 10, 11, 12, 13, 14]. Most of them avoid introducing hardware redundancy. Some use already available sensors to do data fusion and then detect errors as inconsistencies. Some follow a model-based approach and use control theory to detect and even tolerate faults in the system. It has to be clear that our approach is complementary to those techniques. As explained at the introduction, our work focuses on the hardware structure and provides a low-cost infrastructure fulfilling minimum safety requirements. Those safety higher layers can be added on top of our architecture for improved dependability.

III. SAFETY-ORIENTED HARDWARE SERVICES

As we have limited the safety to the AUV retrieval, we do not need to detect all the internal errors in a short time. Moreover, the really hazardous situations can be characterized by the values of some sensors. These sensors are usually known as vehicle self-diagnostic sensors (VSD) and include, but are not limited to, leak detectors, temperature, voltage and current monitors [7]. With these sensors it is only possible to detect critical situations such as a significant increase in the temperature (even a fire), flooding or loss of power. It is clear that these are the situations that directly compromise the retrieval of the vehicle.

In this way, a *critical safety layer* (CSL) can be defined by concentrating all critical safety-related functions in a single hardware node, the aforementioned EHM, and attaching the VSD sensors directly to it. This direct connection prevents the detection of critical situations from being affected by other sources of errors. When the CSL detects a critical situation, it operates directly on the surfacing mechanism. The direct connection of the EHM to the VSD sensors and the independence of the CSL with respect to the CPN as to taking critical decisions gives the required property of orthogonality to the CSL.

Fig. 2 shows how the EHM can be added to the general distributed architecture introduced in fig. 1. Although the EHM could operate isolated from the rest of the AUV, it is also connected to the CAN bus to be able to receive surfacing commands from the CPN. The figure also shows how the VSD sensors are directly attached to the EHM (*A*, *H*, *T* and *V* stand for current, humidity, temperature and voltage sensors, respectively).

This strategy has the following advantages:

1. it guarantees the integrity of the vehicle,
2. the subsystem devoted to critical safety is simple and therefore reliable; at the same time, this simplicity reduces the probability of false alarms; and
3. it reduces the cost of adding safety to the system because it is only necessary to include internal redundancy in the EHM, since it is the only node whose faults could compromise the system safety.

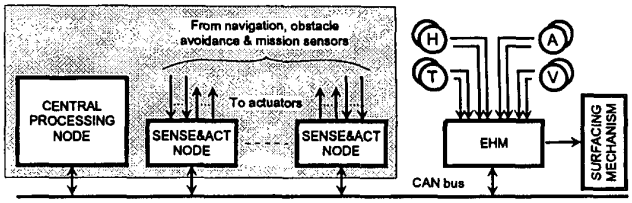

Fig. 2: Integration of the EHM in the AUV hardware architecture.

As it has been explained, our main goal consisting in assuring the vehicle retrieval is achieved by the critical layer. However, there can be many situations which will not be initially considered as critical by the CSL but that could compromise the development of the mission. For instance, if the AUV presented a permanent failure in the propellers, although it would be difficult to accomplish the mission, this would not be considered as an emergency by the CSL until the batteries are finished. To avoid these unnecessary delays we can anticipate the corresponding critical situations by introducing an additional safety layer that we call *preventive safety layer* (PSL). The PSL is implemented by adding a sensor to each actuator that provides a feedback to the orders sent by the CPN and slightly modifying the CSL software to act as a watchdog for the CPN. In this way, the CPN monitors the correct operation of the actuators and the CSL —the most reliable and safest part of the system— monitors the CPN. This simple scheme achieves a reasonable error detection coverage at a low cost and almost does not interfere with the critical layer performance.

These two safety layers are conceived as a set of services that the designer can decide whether to use them or not. As the orthogonality requirement demands, the inclusion of these systematic safety services does not require major changes in the rest of the system, and even less in the application software, that is the most expensive part to change.

With regard to the surfacing mechanism, not only is it fundamental to effectively surface the vehicle, but it is also necessary to be able to locate it. On the one hand, the surfacing could start switching the propellers off. Next, either a ballast releasing or an air-filling floodable tanks strategy could be applied. Both procedures are simple and can be implemented with electromagnetic devices which can also be triggered if the electrical power fails. On the other hand, special measures could be taken to facilitate the remote location of the AUV: while it is submerged, an acoustic modem could operate; once it is on the surface, the communication could be established by RF.

The next two sections describe in more detail the two introduced layers of safety.

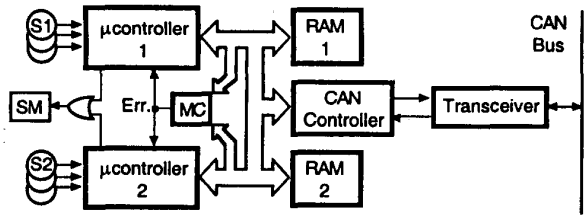

Fig. 3: Hardware organization of the EHM node. S_i stands for the different sensors and *SM* for the Surfacing Mechanism.

IV. THE CRITICAL SAFETY LAYER

After having presented the general functionality of the CSL and the PSL in the last section, we will move into the specific design of the CSL. Both hardware and software have to be carefully designed to attain the fixed requirements and functionality. The architecture of the hardware executing the CSL, the EHM, will be described in the following. The reader is referred to [3] for a more detailed description including the corresponding schematic.

The EHM is characterized by the need of guaranteeing that even in case of failure in the EHM itself the surfacing mechanism is triggered. This requires the EHM to be fail-safe. In this respect, our approach is conservative in the sense that the detection of an error in the EHM fires the surfacing mechanism.

Error detection is done by the duplication and comparison technique. All the components that take part in the decision process inside the EHM —a set of VSD sensors, a microcontroller and an external RAM memory— are duplicated, as it is shown in fig. 3. Each microcontroller executes a replica of the CSL software, which resides in its internal memory and mainly consists of code that continuously samples its own set of sensors and checks and writes the obtained data in its own external RAM. Any discrepancy in the values of addresses, data or control signals generated by the access to these RAMs is detected by an additional circuit called the Main Comparator (MC). The access of the microcontrollers to their internal RAMs, however, is not verified as it does not produce any change in the output of the circuit. Nevertheless, this strategy allows detecting errors when executing the critical functionality of the CSL, as it makes faults in the microcontrollers or in the memories be manifest during the sampling loop. Finally, the circuits related to the implementation of the access to the CAN do not include special redundancy because no critical decision is taken from information sent through the bus.

When the MC detects a discrepancy in the signal values generated during a cycle memory, it outputs an error signal that interrupts both processors which enter what we call a *Last Opportunity Procedure* (LOP). This is a routine that reexecutes the last sampling iteration in order to decide whether the fault that gave rise to the error is transient or permanent. On the one hand, this reexecution allows detecting faults either in the sensors, in the external memories or in the microcontrollers when running the last issued instruction. On the other hand, the completion of the reexecution determines the type of fault: if it is transient both nodes will completely reexecute the last sampling operation and will resume their functions normally; if the fault is permanent a second interruption will be produced during the reexecution. Only in this last case will the surfacing mechanism be activated.

The MC itself is a critical point in the design. Its faults could produce a whole system non-safe failure. So the MC itself must be duplicated. This is better done using design diversity to avoid common mode faults. A final more simple and reliable comparator receives the error signals from both versions of the MC and permits detecting the errors in the MC as well.

Fig. 4 shows a high-level specification of the CSL software. It mainly consists of two procedures (*CSL* and *SampleSensors*) and three interrupt service routines (*ISR-WatchdogEHM*, *ISR-LOP* and *ISR-CANMessage*). In the figure and in the explanation below, the following fonts will be used: *internal*, for variables stored in internal RAM; and *external*, for variables stored in external RAM.

As it can be observed, the *CSL* procedure executes a short loop that continuously calls the *SampleSensors* procedure, which simply polls the VSD sensors and writes their values in a buffer in external memory *M*. *M* is managed as a circular buffer where *Mptr* is the related access pointer; this pointer is also stored in the first entry of *M*. In this way, if the external memories are non-volatile RAMs and an external access to them are provided, the last read sensor values can be retrieved and analysed off-line; that is, the external memories can be considered as a sort of black box.

In the *SampleSensors* procedure, the function *CheckSensorValues* is supposed to combine the VSD sensors values so as to determine, from the specific sensors used, if a critical situation exists (as this decision is dependent on the selected sensors the exact operation over these values is not detailed).

With regard to the interrupt service routines, *ISR-LOP* implements the above-mentioned *Last Opportunity Procedure*, while *ISR-CANMessage* responds to the arrival of CAN messages requiring a surfacing operation. Finally, *ISR-WatchdogEHM* is needed to overcome some conflicts that can appear because of the connection of the EHM to the bus. This connection permits that an anomalously large number of messages are sent to the EHM due to a failure in any node attached to the bus. In that case, the CSL would be continuously interrupted so as not to be able to poll in time the VSD sensors. As this situation could lead to a non-detected critical situation, a protection has been included in the form of a watchdog timer (*WatchdogeEHM*) that the *CSL* procedure should reset before reaching 0. If this was not the case, the *ISR-WatchdogEHM* interrupt service routine would be invoked, triggering the surfacing mechanism. It is worth noting that this watchdog timer