CAN Application in Modular Systems

Andoni Crespo, José Baca, Ariadna Yerpes, Manuel Ferre,

Rafael Aracil and Juan A. Escalera

– Universidad Politécnica de Madrid, Spain –

This paper describes CAN application in a modular robot system. RobMAT is made up of modules that form a new structure called molecule when they are joined together. Every molecule has a master module, which is in charge of receiving external message and retransmitting to the rest of the modules by CAN bus. The message contains all related information about movement control references, sensor data and module synchronization. CAN features allow faster transmission of up to 1Mbit/s. It is also flexible to connecting another CAN device. Such features make CAN appropriate for this application. Task performance using modular robots requires flawless communication among modules; therefore, synchronization is a key factor to take into account where in CAN plays an important role. The importance of synchronization requires a dedicated mailbox to manage it among the modules. Each module comes with a clock in order to process information by itself and correspondingly synchronize. Among the modules, one module has the master clock pulse that has to be transmitted to the rest of the modules of the molecule for readjustment of time. The experiments highlight the excellent performance of synchronization in crucial task.

In recent years there has been an increasing interest in modular robotic systems. These systems aim to carry out a number of tasks by teams of collaborating robots instead of being completed for one simple robot. Modular robotic systems are less task-specialized than industrial robots; nevertheless a greater quantity of them is necessary in order to deal with tasks which one specifically designed robot could not do. Actually, the challenge of these robots lies in coordinating several robots to obtain a common objective with a cooperative behavior.

A good example of team robots are modular and reconfigurable robots, as they form metamorphic structures that are made up of modules, generally identical, which have to work in a coordinated fashion giving uniform behavior to the colony. Their ability of rearranging their modules to adapt to different configurations allows them to cope with many tasks, and consequently increase their performance. The main objective of modularity is that functionality of the whole is greater than the sum of the functionality of each component. The concept of modularity applied in robotics allows making a wide variety of specialized kinematic configuration from a reduced set of standard components. However, the development of modular robots has specific features that have to be solved in order to obtain the desired performance. Most of them are related to control architecture, modular mechanical design and reconfiguration processes.

In modular robotics, control architecture takes on special functions like communication and synchronization among modules, or when and how modules have to change their configuration. Therefore, modular robot architectures can be proposed in several ways, for example, when coupling among modules is done mechanically, hierarchical or centralized architectures are normally implemented such as PolyBot (Yim et al., 2002) or M-TRAN (Kurokawa et al., 2002) systems. On the other hand, when module coupling is not mechanical, like networked robots (McKee and Schenker, 2000), distributed and

decentralized architectures are used. Examples of them are robot systems based on colonies (Navarro-Serment et al., 2002) and (Caprari et al., 2002).

In the RobMAT project CAN plays a key role in the synchronization among modules. Its use is innovative because its application is not really common in other projects with same features. Just in PolyBot (Yim et al., 2002) has been used before, but its assignment is to transfer information, so nothing to do with the synchronization.

This paper has been organized as follows. Section 1 describes the RobMAT control architecture, which explains communication and synchronization among modules. Section 2 is focused on the role that the standard CAN plays in the synchronization among modules and section 3 refers to the rest of tasks where Can is involved in. Section 4 details a real test done with the prototype and section 5 is about the current situation of CAN's development inside the project. Finally, main conclusions from this work about the influence and the application of the CAN bus are summarized in the last section.

1. Control Architecture of the RobMAT System

RobMAT architecture is based on modular and molecular components being all of them one colony. Module is the simplest part which has movement and communication facilities. A molecule is an autonomous robot which is made up of modules. Therefore molecules can have different configurations. Each molecule has a master module and the rest of the modules have a slave role. Later on, master and slave roles will be described. Finally, the colony is remotely commanded by a human operator by means of teleoperation interface. Figure 1 shows the main components of the RobMAT architecture.

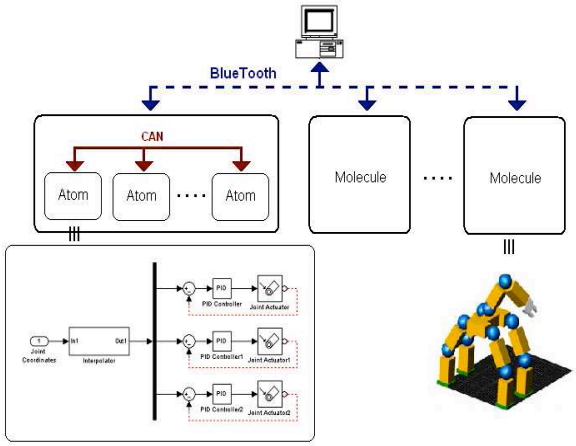

Figure 1: Main components of RobMAT architecture

The module designed for the RobMAT system attempts to reach a balance between complexity of design and performance. The goal is to obtain a very functional module able to execute different kind of tasks. However, it is always kept in mind that the module must be simple in comparison with the complexity of the molecule. With these criterias a module is developed which can form very functional molecules comprised of only a few modules. Figure 2 shows the module designed.

Figure 2: Module of RobMAT

The molecule is a robot formed by various modules which are joined together, as is shown in figure 4. The molecule with the fewest modules which has sufficient autonomy is called base molecule; two modules form this molecule, as is shown in figure 3. Adding suitable tools to a base molecule allows itself to meet several of the demands for a robot.

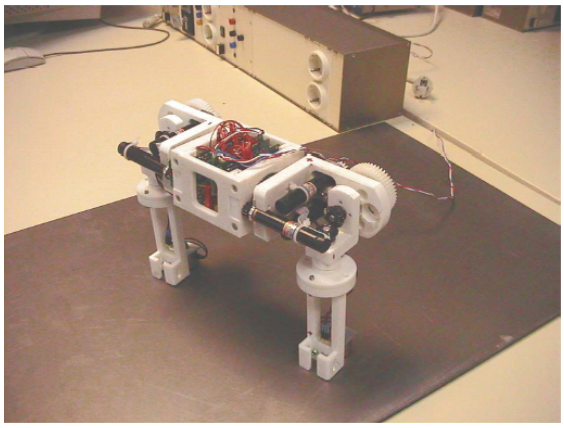

Figure 3: The base molecule

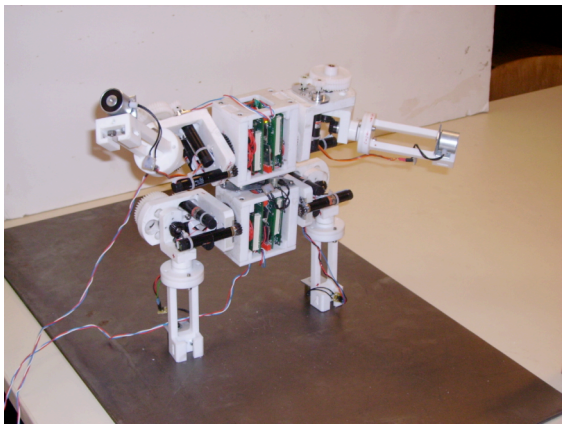

Figure 4: Complex molecule composed of 2 base molecules

The molecule has robot functionality. It receives commands from the operator and the rest of the colony that can be executed autonomously. Hence inter-molecule communication is necessary. Operator's messages among molecules are transmitted via Bluetooth protocol. Another communication which involves modules in the architecture is the intra-molecule communications which are transmitted via CAN bus (CAN Specification Version 2.0, 1991).

In order to adapt these two channels of communication each molecule chooses one module as a master and automatically the rest of the modules become slaves. Note that any module could be a master because they all have the same functionality. The master module is in charge of managing the wireless communication. It receives the commands sent by Bluetooth and then retransmits the proper information to the corresponding slave module via CAN bus. Also, it sends messages through a Bluetooth device to the colony. Slave modules disable their Bluetooth device and only the CAN bus is used for communication purposes. Another function of the master module is to generate a synchronization signal which is described thoroughly in the next section. Basically, it is a short CAN message that fixes period and phase of all module clocks as the master clock.

2. The synchronization of modules

The molecule has a robot behavior thanks to the module synchronization. This global behavior is required to move the molecule as a unit. This is a key point in order to implement a collaborative behavior for the team robot. For example, in order to push or carry on something heavy, modular robots require simultaneously the same parallel movement sharing its forces and creating a new force higher than the sum of the modules used. In other words, it allows starting and finishing all joint movements in a module simultaneously. With this attribute it is possible the execution of complex tasks.

One of the problems in distributed computational systems is the lack of a global clock (Chow and Johnson, 1997). This is a handicap to carry out concurrent actions in a coordinated way. Several approaches have been proposed to overcome this problem and the most used is the message-passing method (Xu, 1990). From the computational point of view modular robots are a set of processing units joined by a communication bus. Therefore, message-passing methods fit as a possible solution to synchronization problems in modular robots. However lots of messages are needed to synchronize the different processes. We propose a discrete time closed-loop method which keeps all system clocks in the same phase and period using a single short message in every cycle. The period of that cycle can be much longer (seconds) than a control cycle period (milliseconds, typically the period of timer interruption).

The chip used in each module is a DSP and is in charge of the operation of its module. The DSP has a special device to develop CAN communications. This is called eCAN module and has two features that are necessary to generate the method to synchronize. Firstly it has 32 mailboxes to send and receive diverse information and secondly, it can develop interrupts when a message has been sent or has been received successfully.

The method needs a periodical signal which acts as a trigger for the closed loop. This signal is generated by the master module every N control modules and consists in a high priority short CAN message. It is supposed, with neglected error, that all modules receive the message at the same time. Each time that message is received, a local timer (ticks counter) is reseted and its previous values are used to correct the local timer period. Eq. (1), i.e., there is a counter that counts the ticks of a local timer (ticks of DSP clock) between every two consecutive synchronizing signals. When a synchronizing message comes in a current counter value is used to recalculate the local timer period. After, the counter is reseted (note that this process happens in every module).

$$T_{i+1} = \frac{C \cdot T + t}{N}$$

Where:

C: Number of control cycles.

T_i: Current local timer period.

t: Current cycle timer ticks.

N: Control cycles per synchronizing period.

Basically the master module creates synchronization signal. It counts the number of interruptions generated by its timer and when it reaches to 300 of them, produces a message and delivers it to the slave modules. These ones have to receive the message and readjust the period of their timer. To pick up the synchronization signal, a low level interrupt associated to the successful

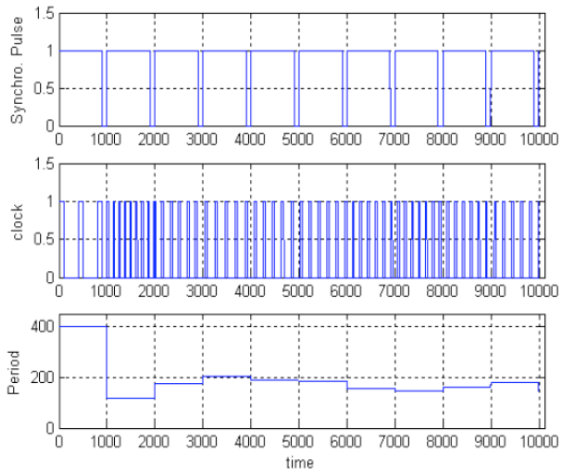

Figure 5: Signals in the process of synchronization

reception of a message in a defined mailbox, is programmed. So, once the message has been received, the period of every slave timer is recalculated following the formula above.

The setting point is the amount of control cycles (N) per synchronizing period. After some synchronizing cycles all clocks have the same phase and period. As is shown in figure 5. In this way actions can be executed in a coordinated manner.

This algorithm for module synchronization allows a cooperative module behavior. For example, using the above method the maximum difference at the starting/stopping time among any motor is below one mili-second. Synchronization is required in order to execute properly trajectories, in other case strong forces can be happen during manipulation tasks.

3. Data transfer

Apart from the synchronization among modules, CAN is also used to transfer the references that are sent by the control station to the modules.

The control station (a common PC) communicates with the master module by Bluetooth. Then, the master slave manages to retransmit all the references sent by the PC to the rest of the slave modules to close their control loops and move their actuators in agreement to the references.

As the role of master is fixed and never changes during the operation, it will be always the transmitter of the bus and the slaves the receivers.

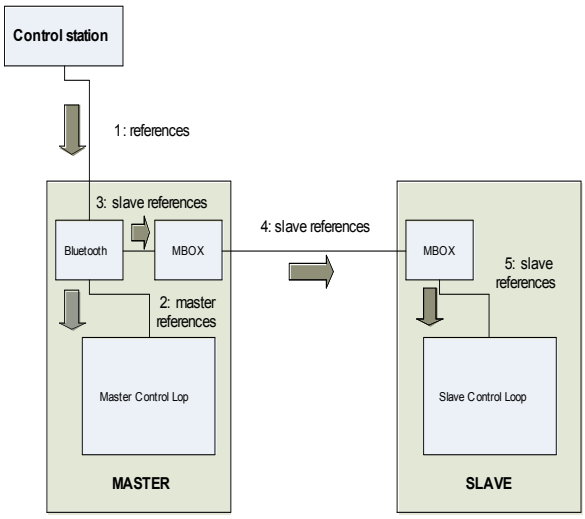

Figure 6 Data transfer scheme

4. Example of synchronization: four module complex molecule

As synchronization is the key concept of this field of robotics, many real tests are always done to ensure its proper operation. Next, an example of the behavior of the synchronization is going to be analyzed when the topology of CAN network has more than one slave module. Looking at figure 7, there are two base molecules forming a complex one and trying to walk as a unit.

The movement pattern in a complex structure is always inspired in the way that an animal moves. In this case, there is a four legged structure which has no knees in the legs, thus, the most similar animal with these physical features is a turtle. This means that the complex molecule's movement pattern is going to be exactly the same. At the beginning, the two front legs have to take a step forward. Once it has been finished, the hole body of the complex structure must be moved to the front and finally two back legs complete the sequence recovering their original position in the complex molecule.

In this situation synchronization plays a major role because there are many modules involved in operation. These ones are at different distances from the master module so, to get movements coordinated, synchronization signal must be received at the same time by all of them. The movement must be homogeneus, otherwise complex molecules's balance would get lost and displacement could not be finished. Fortunately, CAN's transmission rate is fast enough to fit these demands as is shown in figure 7.

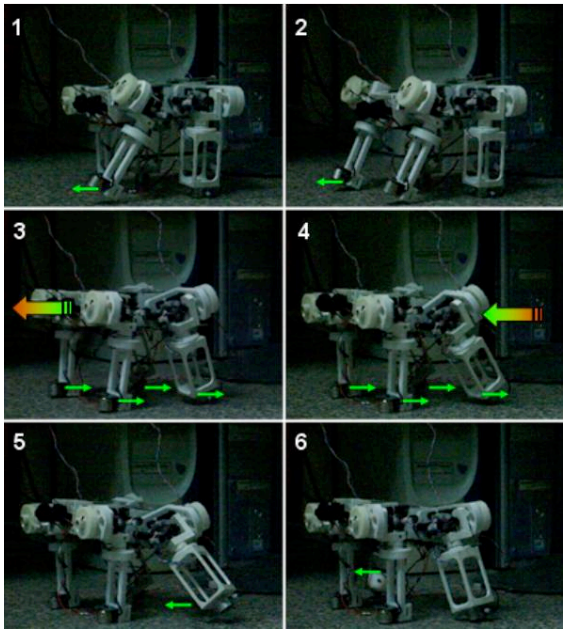

Figure 7 Four module complex molecule

5. Present working area

Actually we are working on the encapsulation of the communications and synchronization.This is a key work to achieve future complex developments.

Basically the main objective of this task is to make the use of CAN more flexible and more modular. This will suppose that future developers of the RobMAT will understand the application of CAN in this project easier than before. By this means they will be able to design new code to manipulate the bus standard and take more advantage of it.

The way of encapsulating communications and synchronization consist on redesigning all the code using an object-oriented programmation language. Thus, we can declare objects that have all CAN

features defined inside and use them to develop new tasks and objectives. At the moment two classes are finished so two kind of objects can be used. The first one is made to manipulate CAN's basic features like the transmission rate, volume of data transferred in each message, number of mailboxes involved…. The other inherits all the characteristics of the first one and is focused on the synchronization. So to manage a base molecule, an object of the second one is needed, but if an additional mailbox is required in case of necessity the first one could be used.

6. Conclusions and future works

CAN is the best bus standard that suits RobMAT project. It brings good communication performances for embedded real time systems. Its bandwidth is 1Mbit/s; so it allows sending low level commands, i.e. position references, synchronization of joint movements and messages to coordinate actions from different modules in the molecule.

For the future more profit has to be taken from its capability of sending information. Actually just to mailboxes are used, one for the signal of synchronization and another for the references. As more sensors are introduced in a base molecule, more mailboxes have to be used to send more data information.

Another point of interest will be the use of CAN standard to change module status dynamically. In advance one mailbox could be reserved to send each module's status in order to change it when the base molecules are going to alter their configuration. Thus at the time of forming a complex molecule, involved base molecules will have to discuss which of their modules will be the master of the hole new structure. This could be a fact that would make base molecules more indepent to get other configurations and to form new modular robotic systems.

Consequently much work has done to reach to the actual point, but much more will be done in the future to develop new features.

Acknowledgements

This work has been partially supported by the Ministerio de Educación y Ciencia of the Spanish Government under CICYT-DPI program by the grant DPI2003-00759.

Andoni Crespo
Department of Control Engineering
Universidad Politécnica de Madrid
acrespo@etsii.upm.es

José Baca
Department of Control Engineering
Universidad Politécnica de Madrid
jbaca@etsii.upm.es

Ariadna Yerpes
Department of Control Engineering
Universidad Politécnica de Madrid
ayerpes@ etsii.upm.es

Jose A. Escalera
Department of Control Engineering
Universidad Politécnica de Madrid
jescalera@etsii.upm.es

Manuel Ferre
Department of Control Engineering
Universidad Politécnica de Madrid
+34 91 336 30 61
+34 91 336 30 10
mferre@etsii.upm.es
http://www.disam.upm.es/grmi/UPM-DISAM%20Manuel%20Ferre.htm

Rafael Aracil
Department of Control Engineering
Universidad Politécnica de Madrid
+34 91 336 30 61
+34 91 336 30 10
aracil@etsii.upm.es
http://www.disam.upm.es/grmi/UPM-DISAM%20Rafael%20Aracil.htm

References

[1] Modular and Self-Configurable Team Robots for Unstructured Environments (2007). Escalera, J., Ferre, M., Aracil, R., Hernámdez, C.

[2] Base molecule design and simulation of a modular robot robmat (2005). Escalera, J., Saltaren, R., Ferre, M., Aracil, R., Garca, C.. In Proceeding of 16th IFAC World Congress, Prague, Czech Republic.

[3] CAN Specificaion Version 2.0 (1991). Retrieved January 12, 2006, from http://www.wisc.edu/writest/Handbook/DocAPA.html.

[4] CAN System Engineering: From Theory to Prctical applications (1997). Lawrenz, W.

[5] PolyBot: a Modular Reconfigurable Robot (2000). Proc. Of the IEEE Int. Conf. on Robotics and Automation, April 2000. Yim, M., Duff, D., Roufas, K.