Software and Hardware Architecture for Autonomous Robots using Distributed Embedded System

P.Niranjan
Design Engineer,
Tatva Software,
Bangalore

S.Selvaraju
Professor/EEE
V.M.K.V Engineering College
Salem

R.Ramani
Assistant Professor,
V.M.K.V Engineering College,
Salem

S.Valarmathy
Assistant Professor
V.M.K.V Engineering College
Salem

ABSTRACT

In this paper, we propose the software and hardware architecture for specifically Autonomous Robots using distributed embedded system. These concepts have implemented in developing a prototype robot. An autonomous robot has minimum three components a. sensor b. controller c. actuators. A distributed embedded system is most enabling technology in developing an autonomous robot, since it contains many computing nodes. Distributed systems are set of sub system, which are independent embedded system but works as a single system for a single goal. Software architecture plays a vital role for the control system of hardware. It uses interdependent layered architecture for the control structure of the robot. The software layers are built on top of "physical layer", which consists of actual sensors and robots. On top of physical layer is the "logical layer" which provides access to the actual physical layers. The other layers are "module substratum", "central controller", "planner", "cognitive layer" and finally the "interpreter". The proposed model aims in reducing the complexity of hardware, interdependent software architecture and a level of autonomy.

Keywords: Software, Hardware architecture, Cognitive layer, Autonomous robots.

1. INTRODUCTION

An autonomous robot performs desired task on unstructured environment without human guidance. It has the ability to sense the environment, gain knowledge and interpret the conditions. Although the present robots are not capable of being fully autonomous, the key challenges are sensing, testing and interoperability [15]. The main components of a robot are sensors which interpret the world, controller's plans and execute the desired task and actuators are the motors or other mechanical devices. To efficiently communicate and control all the components is by improvising the hardware and software architecture of the current system.

The hardware mainly comprises of microcontrollers, sensors and actuators. A distributed embedded system is a collection of independent embedded systems, with shared resources working for the same goal. It is used since it is required to manage many computing nodes such as sensing, controlling, communication, navigation and etc. Using this system reduces the cluster of buses in the previous method and also gives the ability of parallel computing i.e. increasing the performance and activities of the robot. The components (sensors and actuators) present in a robot is represented as "Physical robot", it is the physical layer of the architecture. Software architecture (figure 1) plays an important role in designing the distributed embedded systems. It helps to reduce the hardware components by effectively utilising all the resources of the hardware system. Several layers are created on the physical layer of the robot; unlike the conventional method [5] in the improvised method the layers are interconnected to enhance the communication between the layers. The layer on physical robot is the "logical robot", in order to control the physical robot; libraries are created for the voluntary abstraction of data's from the physical layer. The next layer of the architecture is the "module substratum", which are formed of function modules. These function modules works together for the execution of the specific services given by central controller by effecting on logical controller. The layer above the modules substratum is the "central controller", which is the nervous system of the robot. This layer gets feedback from all the other layers, to verify the physical output with the execution of the robot. During any failure of execution, it tries to recover from the failure either locally or by asking an alternative from the planner. The other layers are "planner", the brain of the robot which decodes and generates the action plan. Finally, the "interpreter layer", it communicates and creates user interface.

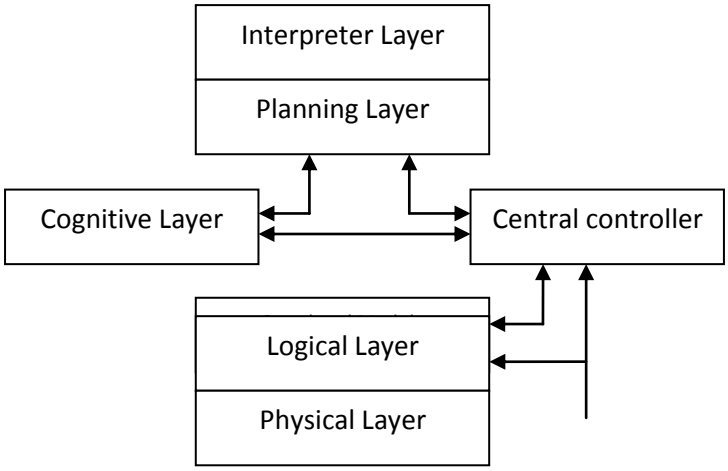

Figure 1: software architecture

The aim of this article is to present:

- Simple hardware architecture with scalability of its components.
- Implementation of Correlated Modules in the software architecture.
- Improvised software architecture for efficient control structure and inter layer communication.
- The communication techniques used for inter-processor communication.

2. HARDWARE ARCHITECTURE

As we can see the generalized hardware architecture of the robot in figure 2, it consists of two racks, physical rack and computing rack. The physical rack consists of a controller which communicates with physical layer (sensors, actuators) of the robot. The computing rack, process all the data's from the physical rack and performs task which is executed by affecting on the physical rack of the robot.

Unlike the conventional architecture, where the physical layer and the computing rack are together combined which is most likely to cause jitters, causing disruptions in computing and higher amount of hardware components. The architectures are also composed of higher number of CPU's, number of bus for inter processor communication [5]. This conventional method of architecture has high complexity and also uses many components. These factors affect the ability of the hardware to extend and reduce the capabilities of debugging and troubleshooting. To overcome these factors, we were motivated to use distributed embedded systems (DEB).DEB focuses on distributing its computing requirements to many independent embedded systems.

Our architecture aims to separate the physical rack with the computing rack, by this way all the physical values are available as objects for the computing rack i.e. enhances the computing ability of the robot. It improvises on parallel computing, scalability or extending the hardware. The sensors can be classified into two different types by its output a. digital sensors, b. analogue sensors. The digital sensors are sensors which gives binary outputs, whereas analogue sensors output needs to be sampled and converted into digital signals. The digital sensors are connected to processor through an electronic switch (multiplexer), so the I/O ports usage can be reduced, however there may rise issue accessing time, to overcome this factor all the sensors are logically ORed, in such way that whenever there is a change in the sensors output an interrupt is triggered and the processor updates the sensors. The analogue sensors may be connected through inbuilt Analog to digital converter using sample and hold circuit. The sensory connections are made in a way to reduce to hardware complexity and also to reduce unnecessary communication with the sensors. The actuators or motor drivers are connected through separate output ports in the physical rack of the hardware architecture. The sensor and motor driver components together are considered as the physical layer of the robot.

The computing rack consists of few controllers required purely for the computing process and has no connection with the physical rack. This is intended to isolate the computation part from the hardware to increase the performance as a system. The inter-processor communication takes place by asynchronous communication protocol which is on-chip feature of most embedded controller. It uses a common serial bus for communication. It replaces the traditional use of parallel bus based communication, since it occupies more hardware resources and the chance of capacitance effect is also high. The processor uses labelled signal messages for communication for accessing the objects for the various processors. The communication protocol used in this architecture will be discussed later in this article. A common clock source is used to create perfect synchronisation between all the processors. It is used so that all the processor runs on same machine cycle creating perfect harmony between execution and computing.

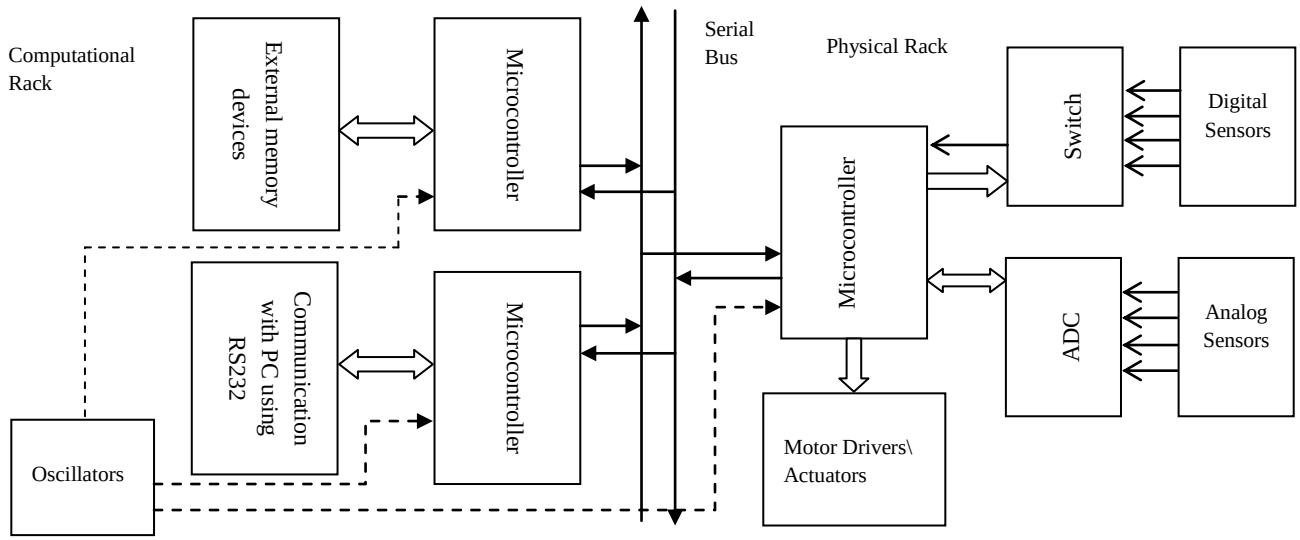

Figure 2: Hardware Architecture

3. SOFTWARE CONCEPTS

Software architecture always acts as an influential role on the design and complexity of the hardware. Well-structured software architecture will result in greater performance and

simplicity of the hardware. The sequential programming methods are often simple, may be one "long" list of statements (or commands). More complex programs will often group smaller sections of these statements into functions or subroutines each of which might perform a particular task. With designs of this sort, it is common for some of the program's data to be 'global', i.e. accessible from any part of the program. As programs grow in size, allowing functions to modify any piece of data means, bugs can have wide-reaching effects. Hence it encouraged us to use object-oriented programming, where objects are accessed by calling methods. It is usually associated with frameworks i.e. a set of models that specify the software components as well as their composition and interaction. Port based programs are widely used in framework development for software design. Unfortunately the use of port based objects resulted in complexity of the models [16] due to multiple port connections. This problem is overcome by using techniques like domain specific modelling and communication. However this ideology is not suited for a complex hierarchal and distributed system.

These factors have resulted in development of the COMDES-II framework [16], which employs a two-level component model (actors, function blocks) as well as signal based communication at various levels. With this framework, an embedded application is composed from actors and actors are configured from function blocks. This is an intuitive and simple model that is easy to use and understand by application experts, i.e. control engineers. Actors can be represented as (in figure 3) real time tasks with event triggered inputs and outputs, which is present in the logical robot layer of the software architecture. Actors are mapped into dynamic or event triggered tasks. The higher layers of the robot may request the status of a particular sensor or actuators. The COMDES framework is implemented in the physical rack of the system, which results in complete separation of physical layer and computational layer.

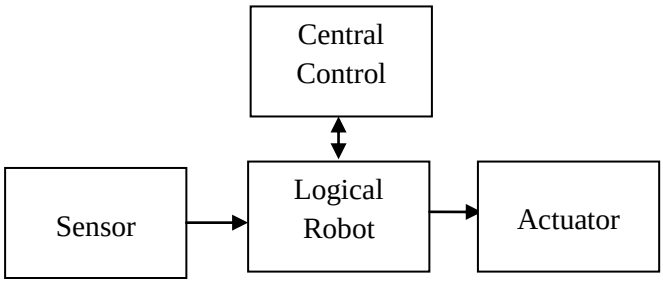

Figure 3: Actor Diagram

4. CONTROL STRUCTURE

The control structure of the robot is conceived from the traditional system [1], by improvising on it. The traditional control systems are composed of two layers in the architecture.

- Module substratum – This layer has set of modules which are responsible for carrying out activities
- Central controller – placed on top of the module substratum layer and is responsible for global supervision.

In the traditional approach, the layers are hierarchical and the central controller can communicate only through the module substratum layer for the feedback of the activities.

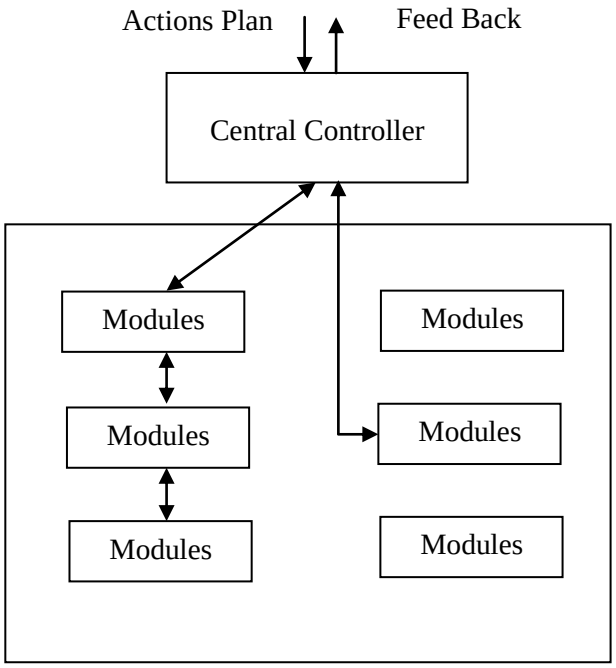

Figure 4: Control Structure

Hence it has only the necessary information of the module substratum. This approach is simple but the ability of the controller to control and supervise the actions plans are less due to its disability of communicating with the other layers of the robot. For example, during an error in the activity, the central controller will receive the error message through module substratum layer. This approach may not be enough to know the exact reason and layer, where the error was created. To overcome this possibility, the central controller is connected directly to all the other layers of the systems, resulting in direct supervision of the activities, which may result in very high efficiency of the robots control structure, however it may result in complexity of the control structure. The improvised control structure is described in figure 4. On the other hand, the module substratum consists of numerous function modules required to accomplish the task given by the central controller. The module substratum represents the distributed software components of the system that allows the robot the accomplishment of the activities with a sufficient degree of autonomy.

5. MODULES

A robot module is itself an independent program may contain a number of functions which are capable of performing events by acting on the logical robot through service requests from the central controller or its parent module. In the traditional systems, module substratum consists of numerous numbers of modules, with no hierarchical layer. The modules interact with other modules to perform an activity. This leads to complex interaction within the modules and also conflicts when two parent modules tries to interact with a same child module. To reduce the complexity of the module substratum and to reduce the conflicts, we were motivated to develop correlated module substratum.

In this approach the modules are grouped into subsets (figure 5) by the following factors

- Correlation with the function.
- By the end activity.
- The utilisation of particular hardware or software resources.

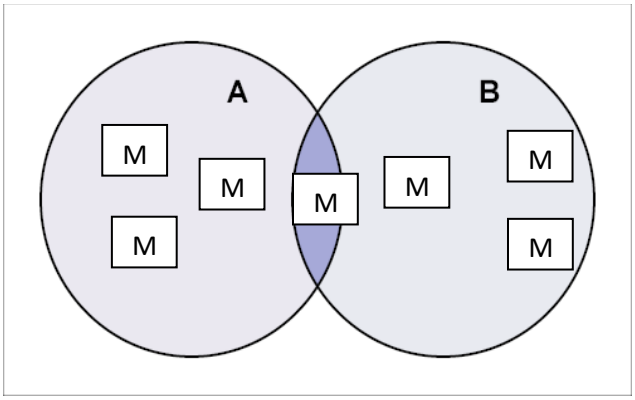

Figure 5: Correlated module substratum

By this way, it is possible to reduce the cluster of shared objects. The modules are also placed in hierarchical layer i.e. reducing conflict. There may be few modules which may be common to different subsets and share both the subsets objects and resources.

The relations between modules are described in figure 6. The modules are layered in hierarchal manner. The arrow a1 indicates the function m21 is called by the function m12 i.e. the function; m12 has control over its chid activity m21. So in general the activity is classified by it behaviour to its other system activities.

In order to increase the functions for future up gradations, the system is open to inherit new modules without affecting the whole structure. The new modules can be added to a definite subset as a parent activity or child activity. The aim of implementing hierarchal layers is to make the system more readable by the central controller, reducing the complexity of the internal data structures and at the same time providing an open system.

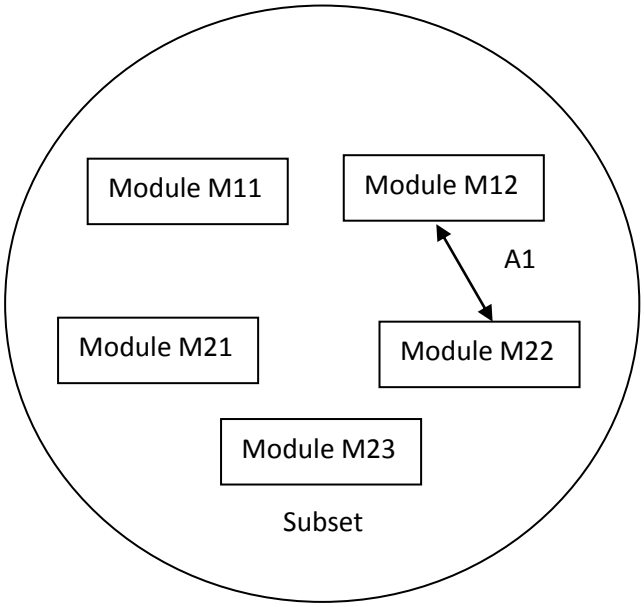

Figure 6: Relation between Modules

6. FUNCTIONAL MODEL OF MODULES

As we have seen in previous sections, modules consists of few functions and while execution of these functions are called as activities. An activity corresponds to a service whose execution is required by means of requests from the central controller or its parent activity. To accomplish this service, this activity may call or requests the other modules, which perhaps acts as child modules. The service of a module can be executed by sending request signals. Each one of the activities can be represented as in figure 7.

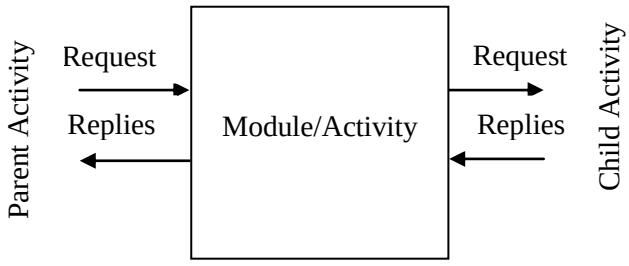

Figure 7: Module function or activity

The activities can be classified into two type's a. short term activity b. long term activity by the time required by the module to complete the activities. The replies sent by these activities are

- Final reply – The final reply is sent by both the type of activities upon the completion of the activity.
- Intermediate reply – This reply is mostly sent by the long term activities since the activities consume more time, however during errors in the STA an intermediate reply is sent to the central controller requesting for an alternative or for the stop signal.

6.1 States of activities

An activity can be one of the three states 1. STOPPED 2. RUNNING 3. SUSPENDED. The states of the activity are directly conceived from the traditional architecture.

When a module receives request to execute its function, the corresponding activity is created and put in the RUNNING state. The running activity may become inactive by two ways.

- By the internal event, service finished after the execution of its function.
- By the external event stop, when the parental activity requests the function to stop.

One activity may be suspended by either by the action of suspended request or by error detection. Once the activity is suspended, the activity may resume by continue or complete stop the execution.

6.2 Module tasks

The modules are executed by the controller which is present and all modules are placed in the same processor. The number of tasks running on the processor was never a problem with efficiency. As the motivation to improve the efficiency helped in implementing timed multitasking [16]. The tasks can be of two types:

- Synchronous tasks – when they execute, it is systematic and periodic way with reference to the real time clock.
- Asynchronous tasks – they execute only after the internal asynchronous event or events generated by interrupt from the logical robot.

The modules may have many tasks, but the control tasks are mandatory in all the modules. This asynchronous task is in charge of the whole task management activities in a module. It receives tasks from parental activities or central controller and executes them by calling appropriate tasks in the module or it sends request for execution in other modules. It receives requests from other modules and prepares replies for the requests. The module tasks communicate by objects and pass the parameters between them. The modules communicate through the labelled signal messages, these messages are in form of request with label of the required object or parameter and the control tasks replies with the parameters value.

6.3 Central controller

Central controller is the nervous system of the robot device. It communicates with module substratum, logical robot, cognitive, and planner. It gets the goal from the cognitive planning layer, and it interprets the data's and calls the appropriate modules required to complete the goal. The central controller has the entire knowledge of the modules and all its available hardware resources.

If an error is detected in the goal of the robot, the central controller tries to spot the errors location by communicating with associated modules and the hardware through the logical robot. If the error cannot be rectified by the central controller, it sends a message to the planner requesting alternative plan for the robot.

6.4 Cognitive layer

Cognitive is a term concerned with goals, perception and actions. It is used in this system to create a degree of autonomy. However, in an autonomous robot the goals perceived by the robot will be less, since the end goals are predefined and don't change. The use of cognitive layer is to help the robot to interpret and perceive its environment in a better way. Some examples of high-level cognitive functions already implemented on robots are shown in [4] and [14].

The first control architecture developed by is a hybrid architecture named automatic/deliberative (AD), took the theories of the modern psychology expressed by Shiffrin and Schneider [6], [7].According to these authors, two mechanisms of processing information is established: automatic processes and controlled ones. I.e. we can differentiate between two levels of activity: automatic and deliberative. The concept of autonomy has been treated by authors such as Arkin [11], Gadanho[8], Bellman [9], or Cañamero[10]. In general, they state that an autonomous robot must be self-sustained, which implies a decision making system. It must have some goals and motivations and these must be oriented to maintain its internal equilibrium (homeostasis).

Realising the necessity of the cognitive layer for maintaining the internal stability, we were motivated to develop cognitive planner to make sustainable robot. This robot uses the reduced version of the architecture used in [2]. We start our approach by the learning algorithm. The learning process is made using a well-known reinforcement learning algorithm, Q-learning [15]. By using this algorithm, the robot learns the value of every action through its interpretation with the environment. Two factors have to be considered for developing the cognitive layer: Homeostasis, and motivations. Hull [13] proposed the idea that motivation is determinedly two factors. The first factor is the drive. The second one is the incentive, that is, the presence of an external stimulus that predicts the future reduction of the need. For example, the presence of food constitutes an incentive for a hungry animal.

The idea was that the behaviours of an autonomous robot are directed by motivational states and its basic emotions. As it is said before, motivations can be viewed as homeostatic processes that maintain inner variables controlled within a certain range of value. A detector of errors generates an error signal, the drive, when the value of this variable is not equal to its ideal value. Each motivation is modelled as a function of its related drive and an external or incentive stimulus. The motivation with the highest value becomes active and it will organize the behaviour of the agent in order to satisfy the drive.

The cognitive architecture consists of two levels automatic and deliberative levels.

1) Deliberative Level: In the natural world, deliberative activities are characterized by the actions that are carried out in a conscious form. Deliberative processes require a large quantity of time to be dedicated to analysing. These activities are carried out sequentially, one after another, and it is not possible to carry out more than one deliberative activity at a time.

2) Automatic Level: Automatic activities are characterized by the actions and perceptions and also out without consciousness of the processes responsible for controlling. Examples of this would be the heartbeat, the hand movement when writing, or that of legs when walking. An automatic activity can be carried out in parallel with other automatic activities and with a deliberative activity. In the AD implementation, the automatic level is mainly formed by activities that are related with sensors and actuators. It can be performed in a parallel to other activities and they can be integrated in order to achieve more complex activities.

3) AD Memories: One of the vital characteristics of humans is their ability to store information from past experiences. Memory can be defined as the capacity of a human to recall the past experiences. Based on the memory model proposed by Atkinson and Shiffrin, the architecture has two different memories: the short-term memory and the long-term memory. In our architecture, short-term memory is defined as a temporary memory stored within the controller. This memory is regarded as a working memory where temporary information is shared among processes and activities. On the other hand, long-term memory is a permanent storage of information. The information can come from learning, from processing the information stored in short-term memory or as prior information. This memory refers to a permanent memory where stable information is available only for deliberative skills.

The AD architecture used in this robot is originated from [2] and is shown in figure 8, the decision making system has a bidirectional communication with the AD architecture. The decision making system will select the behaviour the robot must execute according to its state. This behaviour will be taken by the AD architecture activating the corresponding activities (deliberative or automatic). The decision making system needs information in order to update the internal and external state of the robot. As explained, the term homeostasis is means maintaining a stable internal state. This internal state can be configured by variables, which must be at an ideal level. When the value of these variables varies from the ideal value, an error signal occurs.

In summary, the decision making process is cyclic and it can be described in the following points:

1. Updating the drives and motivation intensities.

2. Interpreting the external state.

3. Generating the reinforcement function.

4. Reinforcement learning.

This cognitive model is derived from "A Biologically Inspired Architecture for an Autonomous and Social Robot" [2] by authors María Malfaz, Álvaro Castro-González, Ramón Barber, and Miguel A. Salichs. In our model the cognitive layer influences the planning layer and the central controller to take decisions based on motives and emotions.

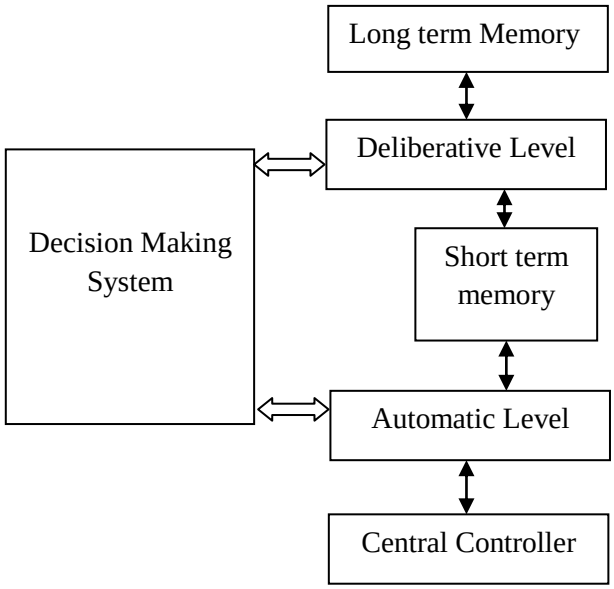

Figure 8: Cognitive Layer

7. A COMMUNICATION PROTOCOL

The hardware is scalable depending on the various applications. Any number of controllers and other hardware resources can be implemented in this architecture. To achieve it, a proper communication protocol is needed.

As we have discussed earlier, a common serial bus is used for inter processor communication. The processors are interconnected through a serial bus. The communication is done through asynchronous labelled signal messages. E.g. when a processor requires a parameter value from other processor, it sends the address of the processor, the code of the parameter. All the processor receives this signal, but only the processor with the address mentioned acknowledges the request and sends the value as a reply with the parameter code. The format of the serial messages transmitted is described in figure 9.

In each microcontroller the received signal goes through a function which checks the address and categorises the parameter code i.e. communicates with the internal modules for retrieving the data, if it available else returns with an unavailable signal to the requested processor. In case of a value to be written for a parameter, the parameter code is followed by the value. If the processor requests a module to execute, it sends the module code followed by the address of the system.

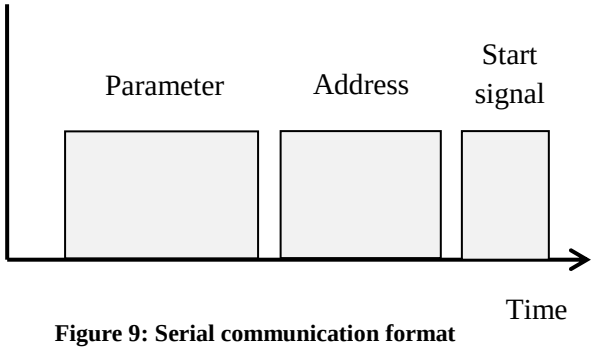

Figure 9: Serial communication format

7. IMPLEMENTATION

The architecture model discussed is implemented in an autonomous cleaning robot model created in Vinayaka mission engineering college. The robot was able to perform efficiently, with a required degree of autonomy. Autonomous cleaning robot objectives are to be able to navigate in an undefined area of plane surface, should be able to cover all the regions for cleaning and return to its home position. In this robot, the goals are predefined i.e. the cognitive layer is partially used for memories.

The robot uses two Renesas R8C microcontrollers; one of the controllers is used as a logical robot and the module substratum while the other is used for implementing the other layers of the software architecture. The figure 10 and 11 describes the circuit diagram of the robot.

In figure 10, the circuit diagram represented is the physical rack of the system. This microcontroller communicates with all the hardware devices such as sensors, motors and displays. The software architecture layers present in this microcontroller are logical robot and the module substratum. These two layers enable the robot to perform an activity by interpreting the environment.

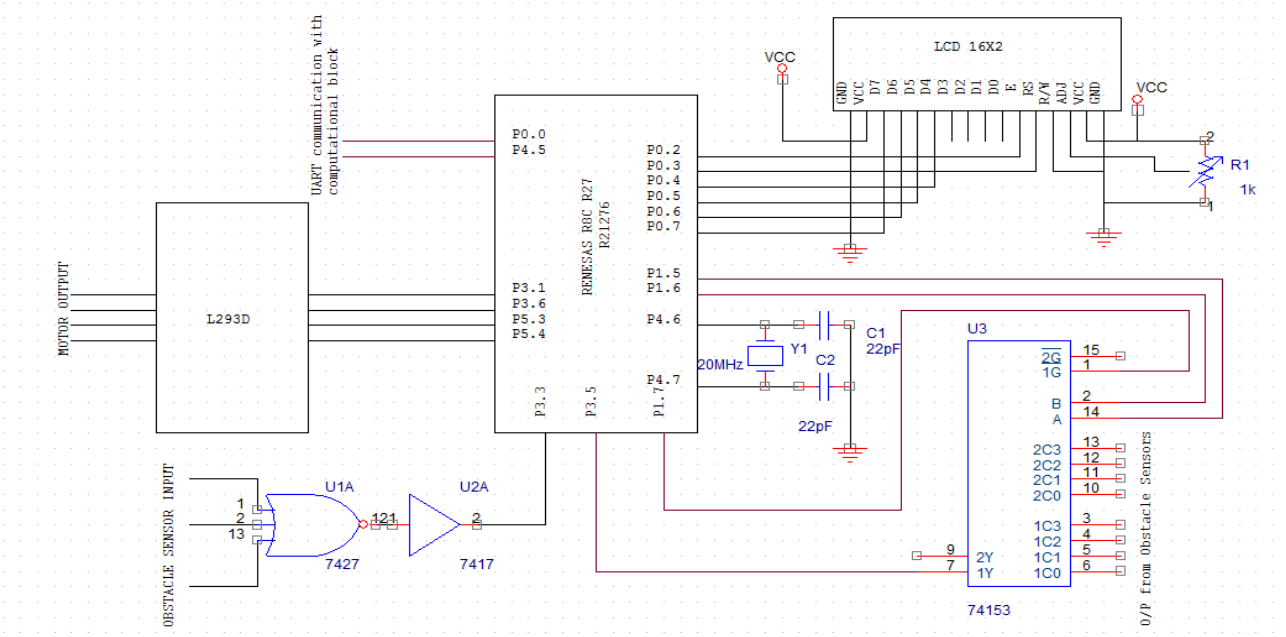

Figure 10. Physical rack

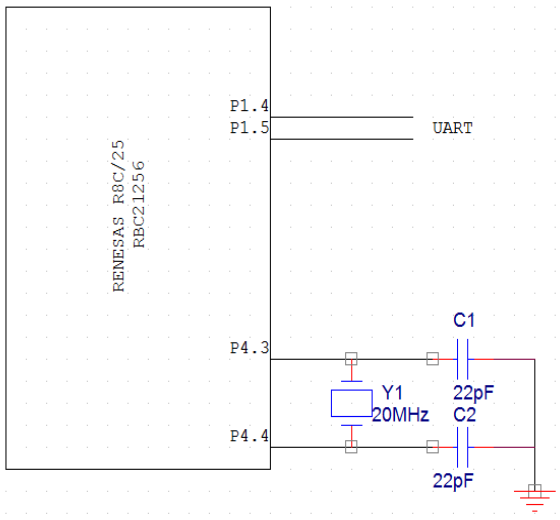

Figure 11: Computational Block

The brain of the robot is the other microcontroller. It possesses software layers such as planning layer, central controller, cognitive layer and finally the interpretation layer. In figure 11, the microcontroller doesn't interpret with any hardware explicitly, to isolate the processes and to gain performance. The central controller layer has all the information about all the modules or functions present in the entire autonomous robot. Cognitive layer holds memory of previous instances, which indeed affect the decision of the planning layer.

This autonomous cleaning robot was demonstrated successfully by using this hardware and software model. The models which are already developed complemented the ability of the system to be flexible, dynamic, and efficient.

8. COMPARATIVE STUDY

The autonomous cleaning robot developed by this architecture was compared with an earlier model, which was constructed by a simple embedded system, with layered software architecture and doesn't possess any cognitive layer. The hardware capabilities such as sensors, processor speed,

and motor specifications are same of both the robots are same. The robots were tested in the same environment. The task for the robots is to navigate, clean, cover all regions of a plane surface area and return to its original position, which consisted of three partitions of 20 square foot each with polyhedral obstacles, to check the maneuver ability of the robot. Some critical parameters noted were time, interpretation, sensing ability, navigation. The major differences noted in these parameters are (table 1):

Table 1: Comparative study

Parameters	Earlier Architecture	Proposed Architecture
Time taken for the entire task	3.48 Minutes	2.5 Minutes
Navigation	Random	Orderly
Sensing	Inactivity for less than a second	No noticeable inactivity
Interpretation	Not capable	Capable
Error Handling	Should reset the robot	Robot handled

- The difference in time taken for the robot to complete the entire task is almost a minute, which are mainly due to the isolation of the computing and physical rack.
- The navigation was orderly, covering all regions, whereas the earlier architecture missed few regions.
- The sensing at certain instances made the earlier architecture to look inactive to compute the data's and react, but didn't exist in the proposed architecture.
- There is no ability for the earlier architecture to interpret the earlier instances.

- The ability to acquire previous instances of the robot in the proposed architecture was able to navigate back to its original position, by mapping through the path without obstacles.
- At certain points during the working of the robot, the previous architecture lost its internal stability between decisions, but the new robot architecture handled all certain instances.

9. CONCLUSION AND FUTURE WORK

The hardware and software architecture for an efficient autonomous robot is created, with an addition of cognitive layer to ensure the robot to achieve a degree of autonomy. The ideas are conceived from improvising various existing techniques. The developed architecture provides engineers an apropos hardware and software models to integrate and implement newer elements without doing any major changes in the system.

In any present autonomous robot, Sensing, Interoperability and testing are elements which need to be improvised to achieve full degree of autonomy [15]. In the future work, sensing, interoperability and testing will be improvised. Although there are various sensors can sense various physical quantities are available and are increasing, the ability of the robot to interpret the sensor data is not accurate. This can be overcome by integrating emotions and biologically based learning algorithm. Interoperability, the ability of a robot to communicate with other machines and humans will be implemented by using a standard message communication protocol. The ability to test a robot under various scenarios can be developed by creating software, which can simulate various conditions by inducing appropriate sensor signals into the system and record the response. Improvising on these elements will result in safe, independent and fully autonomous robots.

The autonomous robots with these capabilities will ensure safety and reliability in its operation. Numerous applications can be created and commercialised with this architecture, such as cleaning robots, social robots, rescue robots, mapping robots and etc. These robots will help betterment the life's of humans.

10. REFERENCES

[1]. "Hardware and Software Architecture for Execution Control of an Autonomous Mobile Robot" - Rogkrio F. Camago*, Raja Chatila and RachidAlam

[2]. "A Biologically Inspired Architecture for an Autonomous and Social Robot" - MaríaMalfaz, Álvaro Castro-González, Ramón Barber, and Miguel A. Salich

[3]. "A Distributed Architecture for Autonomous Robots" - Maurizio Piaggio, Antonio Sgorbissa and Renato Zaccaria

[4]. W. Maier and E. Steinbach, "A probabilistic appearance representationand its application to surprise detection in cognitive robots," IEEETrans. Autonom. Mental Develop., vol. 2, no. 4, pp. 267–281, Dec.2010.

[5]. Y. Zhang and J. Weng, "Spatio-temporal multimodal developmentallearning," IEEE Trans. Autonom. Mental Develop., vol. 2, no. 3, pp.149–166, Sep. 2010.

[6] R. M. Shiffrin, "Attention," in *Stevens Handbook of Experimental sychology*,2nd ed. New York: Wiley, 1988, vol. 2.

[7] R.M. Shiffrin and W. Schneider, "Controlled and automatic human information processing: In perceptual learning, automatic attending and a general theory," *Psicolog. Rev.*, pp. 127–190, 1997.

[8] S. Gadanho, "Reinforcement Learning in Autonomous Robots: An Empirical Investigation of the Role of Emotions," Ph.D. dissertation, Univ.of Edinburgh, Edinburgh, U.K., 1999.

[9] K. L. Bellman, "Emotions: Meaningful mappings between the individual and its world," in *Emotions in Humans and Artefacts*. Cambridge, A: MIT Press, 2003.

[10] L. Cañamero, "Designing emotions for activity selection in autonomous agents," in *Emotions in Humans and Artifacts*. Cambridge, A: MIT Press, 2003.

[11] R. C. Arkin, "Homeostatic control for a mobile robot: Dynamic replanning in hazardous environments," in *Proc. SPIE Conf. Mobile Robot.*,Cambridge, MA, 1988.

[12] C. J. Watkins, "Models of Delayed Reinforcement Learning," Ph.D.dissertation, Cambridge Univ., Cambridge, U.K., 1989.

[13] C. L. Hull, *Principles of Behavior*. New York: Appleton CenturyCrofts, 1943.

[14] N. Kubota, Y. Nojima, N. Baba, F. Kojima, and T. Fukuda, "Evolvingpet robot with emotional model," in *Proc. 2000 Congress Evolut. Comput.*, 2000.

[15] Lora G. Weiss "Autonomous Robots in the fog of war" IEEE spectrum august 2011 issue.

[16] By Jie Liu and Edward A. Lee "Timed multitasking for real time embedded software" February 2003, IEEE control system magazine.