Jurnal Teknologi, 35(D) Dis. 2001: 99–114
© Universiti Teknologi Malaysia

DEVELOPMENT OF REACTIVE CONTROL STRATEGY FOR MULTI-AGENT MOBILE ROBOTICS SYSTEM

MOHD RIDZUAN AHMAD[1], SHAMSUDIN H. M. AMIN[2] & ROSBI MAMAT[3]

Abstrak. Multi-agent system is one the most popular research interests in robotics nowadays. Imagine for a swarm of robots, what kind of intelligence can we build now? This question arises from emulating biological systems. How is it that colonies of small ants can work together to perform a difficult task that single ant cannot do it alone? How might collections of simple robots aid us in human endeavours? By examining these questions, one of the common characteristics shared by each agent is in agent simplicity. Simplicity in control means that the robots are provided with direct commands in order to execute the action. It doesn't need to gather all information from various sensors to plan its action like deliberative control which is also known as sensors fusion that clearly required a lot of computational time and also the mapping that its produces might not valid at the time when changes occurs between the time it gathers the information and the time it plans its actions. Instead, intelligence will arise when these simple agents work together to perform some complex task cooperatively. By examining the major control architectures available in robotics area, subsumption-based reactive control architecture gives the tools as required. Priority-based scheme is used in arbitration level to select which behaviour to run. In this paper decentralised reactive control architecture was used to justify the communication based simple control architecture can perform complex task like carrying load and navigate.

Keywords: Multi-agent system, reactive control architecture, control algorithm.

1.0 INTRODUCTION

Multi-agent system is a nature of life. In human domain, this statement is always true [1–3]. How about in the artificial intelligence domain? In previous years, most of the researchers concentrated on a single agent. But, slowly the researchers realised the advantages of using multi robots in the achievement of complex task where a single agent cannot do it alone [4–7]. In dealing with multi-agent system, two aspects must be identified first; *type of task* and *control architecture*. The types of multi-agent's tasks have been explored in three major classifications, which are known as merely coexisting, loosely coupled and tightly coupled [8]. In merely coexisting; multiple robots coexist in a shared environment, but do not even recognized each other (merely as obstacles). While in loosely coupled; multiple robots shared an environment and sense each other and may interact, but do not depend on one another (members of

[1, 2 & 3] Centre for Artificial Intelligence and Robotics (CAIRO), Faculty of Electrical Engineering, Universiti Teknologi Malaysia, 81310 Skudai, Johor Darul Ta'zim.
Email: ridzuan@nadi.fke.utm.my; sham@suria.fke.utm.my; rosbi@suria.fke.utm.my.

the group can be removed without significant effect). In our work, we are concentrating on the tightly coupled task where multiple robots cooperate on a precise task, usually by using communication, turn taking, and other means of tight coordination.

Control architecture in robotics area can be divided into two parts; single robot system and multi-robot system. For a single robot system, there are *Deliberative*, *Hybrid*, *Reactive* and *Behavior-based* control architectures [8]. These four control architectures are grouped together into two control approaches under multi-robot system, which are Centralised (Deliberative and Hybrid) and Decentralised (Reactive and Behavior-based) [8]. By examining these control architectures and aiming of simplicity in the multi-agent system, reactive control architecture gives the tool for developing the control algorithm for our multi-agent system. The philosophy of simplicity in multi-agent system is not saying we should build robot in simple worlds and then gradually increase the complexity of the worlds. Rather by building simple agent in the most complex world we can imagine and gradually increasing the complexity of the robot [9]. It is interesting to define intelligence in term of group behavior instead of individual behaviour in a multi-robot domain. This means that each agent doesn't have to be too complex in term of hardware and software. Instead the intelligent group behaviors will emerge when these simple agent are working together. This group behaviour is also known as *emergent behavior* and can be observed when interaction between several robots occurred, but are hidden inside a single agent. In many cases, reactive and behaviors-based systems are designed to take advantage of such interactions [4]. They are designed to include emergent behaviors. This paper is organized as follows: section 2 discusses the strategy to accomplish the main tasks. Section 3 discusses the development of reactive control algorithm for multi-agent system. An experimental result is highlighted in section 4. Section 5 concludes this paper.

2.0 STRATEGY FOR MAIN TASK ACHIEVEMENT

2.1 Main Tasks for Multi-Agent System

Our multi-agent system consists of three mobile robots that are capable of searching and passing through an unknown passage in an indoor environment while holding one large object on top of them, as shown in Figure 1. The main idea is to use as *simple agent* (in term of hardware and software) as possible, and with the help of inter-agent interaction, they can do the complex task in a style. The main task of these three mobile robots is to transfer a large object through an unknown passage. Leader-followers strategy has been used where one mobile robot will be a leader while the rest will be a follower.

From here, the main task can be divided into four sub-tasks for task accomplishment as stated below:

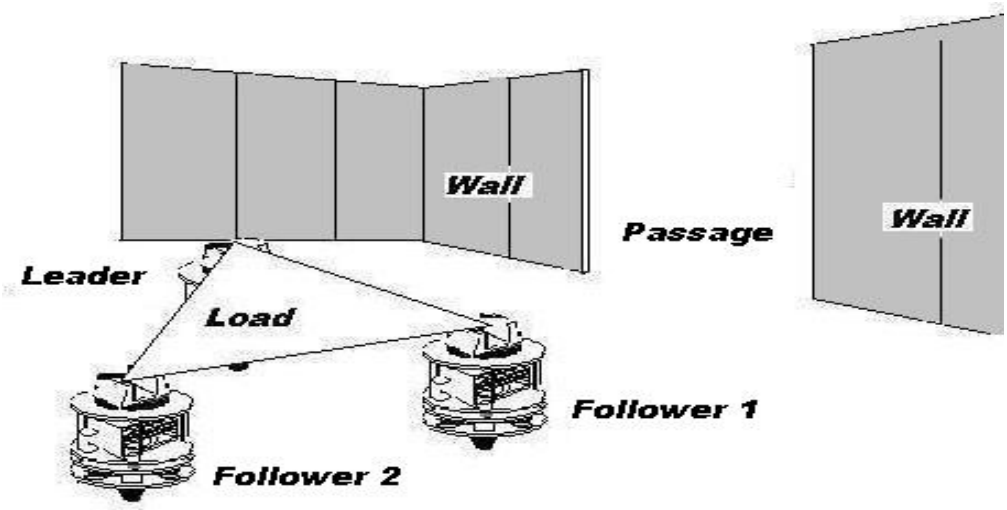

Figure 1 Multi-Agent System

- Holding and carrying load safely
- Navigating in a team
- Searching for the passage
- Passing through the passage (orientation) and task confirmation.

2.2 Holding and Carrying the Load

In the absence of a manipulator, a supporting base is used to holding the load and provides some sort of small movement from the load. The idea is to avoid static placement of the load on top of the robot, that this will cause the system to be push-and-pull situation between agents. The supporting base is shown in Figure 2.

Here, agents must make sure that the load is on top of them and that load is stable while they are navigating.

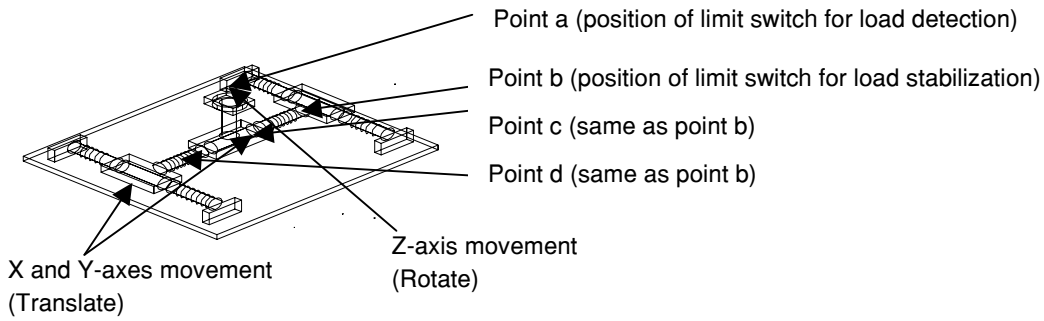

Figure 2 Supporting Base

For the load detection, a push button is used and its location is at the point a. While for the load stabilization, three limit switches are used and placed at point b, c, and d respectively. When the system is released to move, the load has small movement freedom in translation (x and y-axes) and rotation (z-axis). This will avoid a push-pull situation between agents as shown in Figure 3.

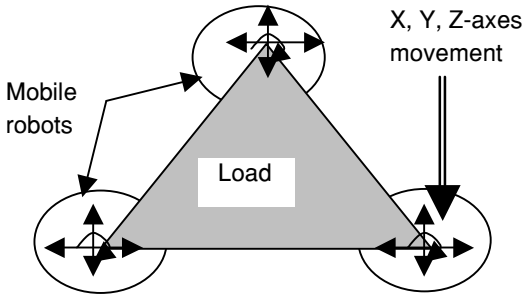

Figure 3 Load Dynamic Movement

2.3 Group Navigation

Since the task of the multi-agent is under tightly-coupled, this means that all agents are physically related by means of the load through the horizontal distributions, making the group navigation behaviour difficult to achieve. There are three issues in group navigation; *distance maintenance*, *direction maintenance* and *obstacle avoidance*. Most of the researchers used explicit inter-agent communication to achieve these behaviours according to [10–13]. Again, this will increase the complexity of both hardware and software of the multi-agent system. We handled all issues by using just simple sensors and under implicit inter-agent communication as shown in [14–15]. For distance maintenance, three limit switches are attached at the point b, c and d as shown in Figure 2. The idea is to inform the followers about leader's movement. Point c indicates normal position and this tells that the leader is in static condition. While point b and d indicate maximum and minimum limit respectively. Maximum limit tells that the leader is in forward direction while minimum limit tells otherwise. The movement of the followers are based totally by the outputs of these sensors. The triangular formation of our three mobile robots is shown in Figure 4 below.

Lastly, for obstacle avoidance, one infrared proximity sensor is attached to each agent. Leader's proximity sensor will cover front area while proximity sensors on both followers will cover left and right area. The execution of this behaviour is affected by the distance and direction maintenance behaviours. For example, if the leader detects obstacles, it will stop. This will make the other followers to stop, due to the consequence of distance maintenance behaviour.

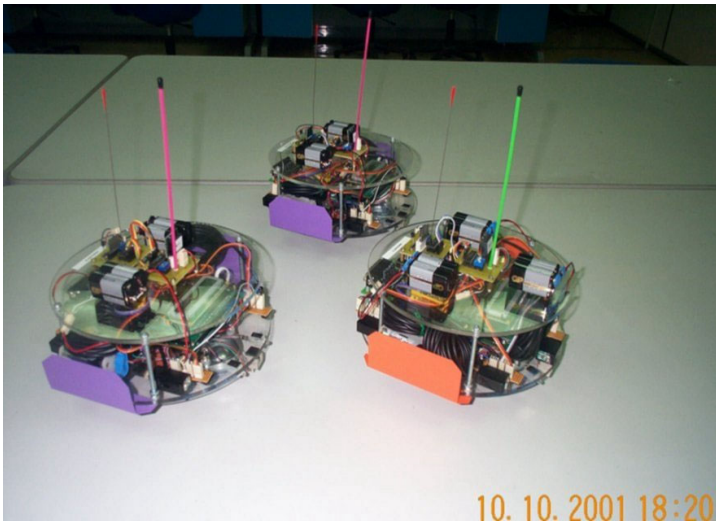

Figure 4 Triangular formation of Three Mobile Robots

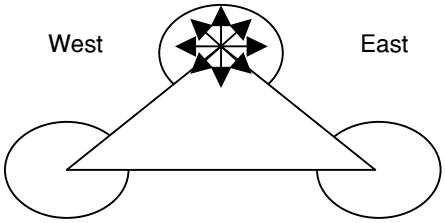

Figure 5 Direction Maintenance

2.4 Passage Searching

By using the strategy of wall following, the leader will first search for the wall. Then, the team will navigate along the wall until other sensors sensed the passage. Here, again infrared sensor is used. This is illustrated in Figure 6.

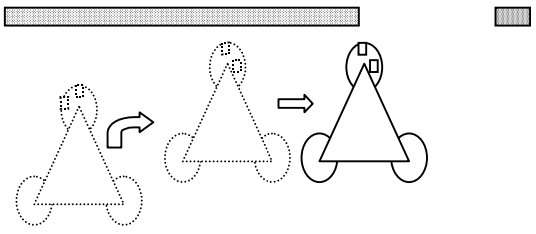

Figure 6 Wall and Passage Detection

2.5 Task Completion

When the passage has been detected, the team will move forward. This action will cause the rear agent (follower) to detect the wall as an obstacle (consequence from obstacle avoidance). This will execute the obstacle avoidance behaviour until all agents have successfully passed through the passage. This is illustrated in Figure 7.

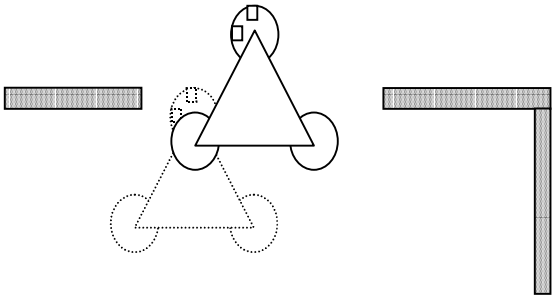

Figure 7 Task Completion Behaviour

3.0 MULTI-AGENT REACTIVE CONTROL ALGORITHM

3.1 Behaviours as Finite State Machine (FSM)

In developing of reactive control algorithm, it is wise to start it with the finite state machine. A finite-state machine (FSM) is an abstract computational element which is composed of a collection of states. Given a particular input, a finite state machine may change to a different state or stay in the same state. The specification of an FSM includes rules that determine the relationship between inputs and state changes [16]. This is shown in Figures 8–15.

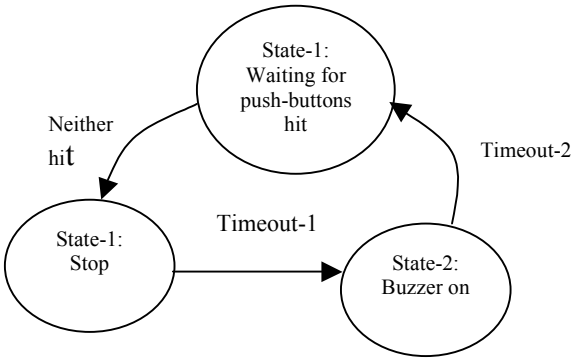

Figure 8 Load Detection Behaviour

In pseudo code structure:
Outputs: (motor-command), (buzzer-command)

```
State-1:  If      Push_button_Hit = All
                  Release
          Else    Push_button_Hit = Not_All
                  Switch to State-2
State-2:  If      time_in_this_state > timeout_1
                  Switch to State-3
          Else    Stop
                  Release
State-3:  If      time_in_this_state > timeout_2
                  Switch to State-1
          Else    buzzer-command = ON Else
                  Release
```

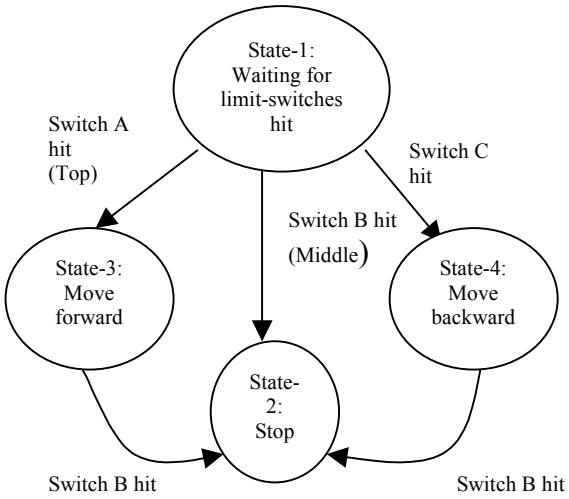

Figure 9 Distance Maintenance

Output: (motor_command)

```
State-1:  If      Limit_Switch_Hit = NIL
                  Release
          Else    If      Limit_Switch_Hit = Middle
                          Switch to State-2
          Else    If      Limit_Switch_Hit = Top
                          Switch to State 3
          Else    If      Limit_Switch_Hit = Bottom
                          Switch to State-4
```

```
State-2:  If      time_in_this_state > timeout_1
                  Switch to State-1
          Else    motor_command = Stop
                  Release
State-3:  If      Limit_Switch_Hit = Middle
                  Switch to State-2
          Else    motor_command = Forward
                  Release
State-4:  If      Limit_Switch_Hit = Middle
                  Switch to State-2
          Else    motor_command = Backward
                  Release
```

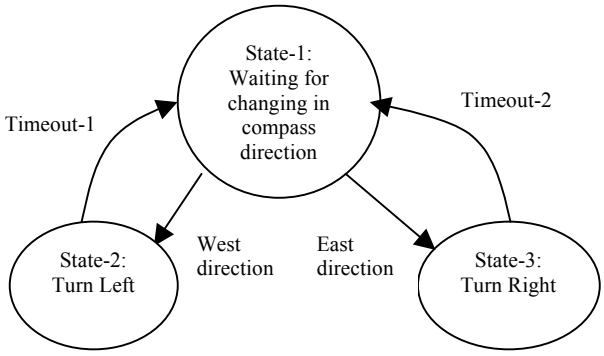

Figure 10 Direction Maintenance

```
Output:   (motor_command)
State-1:  If      Compass_Change = NIL
                  Release
          Else    If      Compass_Change = West
          Else    If      Compass_Change = East
                          Switch to State-3
State-2:  If      time_in_this_state > timeout_1
                  Switch to State-3
          Else    motor_command = Turn_Left
                  Release
State-3:  If      time_in_this_state > timeout_2
                  Switch to State-1
          Else    motor_command = Turn_Right
                  Release
```

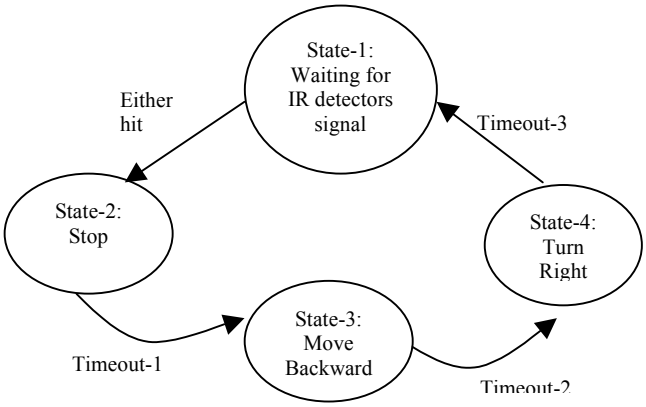

Figure 11 Obstacle Avoidance

```
Output: (motor_command)
State-1:  If      IR_Detect = NIL
                  Release
          Else    IR_Detect = Neither
                  Switch to State-2
State-2:  If      time_in_this_state > timeout_1
                  Switch to State-3
          Else    motor_command = Stop
                  Release
State-3:  If      time_in_this_state > timeout_2
                  Switch to State-4
          Else    motor_command = Backward
                  Release
State-4:  If      time_in_this_state > timeout_3
                  Switch to State-1
          Else    motor_command = Turn_Right
                  Release
```

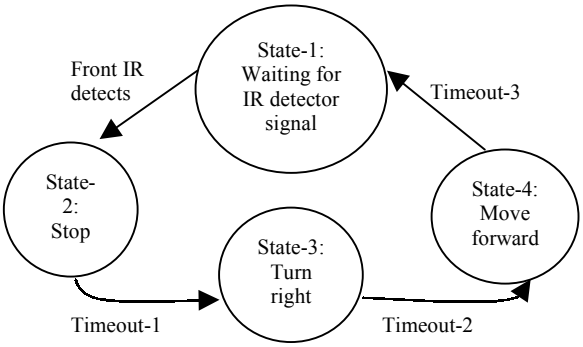

Figure 12 Wall Detection

```
Output:   (motor_command)
State-1:  If      IR_Front_Detect = NIL
                  Release
          Else    Switch to State-2
State-2:  If      time_in_this_state > timeout-1
                  Switch to State-3
          Else    motor_command = Stop
                  Release
State-3:  If      time_in_this_state > timeout-2
                  Switch to State-4
          Else    motor_command = Turn_Right
                  Release
State-4:  If      time_in_this_state = timeout-3
                  Switch to State-1
                  Else     motor_command = Forward
                  Release
```

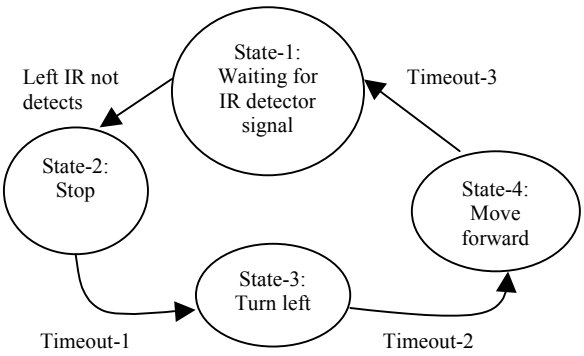

Figure 13 Passage Detection and Passing

```
Output:   (motor_command)
State-1:  If      IR_Left_Detect = NIL
                  Release
          Else    Switch to State-2
State-2:  If      time_in_this_state > timeout-1
                  Switch to State-3
          Else    motor_command = Stop
                  Release
State-3:  If      time_in_this_state > timeout-2
                  Switch to State_4
          Else    motor_command = Turn_Left
                  Release
```

```
State-4:  If      time_in_this_state > timeout-3
                  Switch to State-1
                  Else     motor_command =Forward
                  Release
```

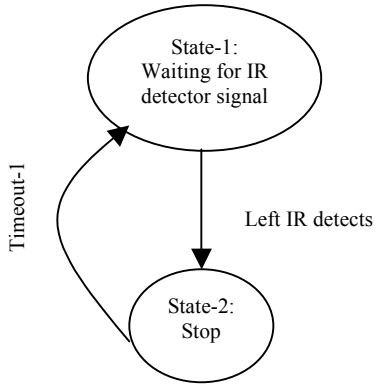

Figure 14 Task Confirmation

```
Output:   (motor_command)
State-1:  If      IR_Left_Detect = NIL
                  Release
          Else    Switch to State-2
State-2:  If      time_in_this_state > timeout-1
                  Switch to State-1
          Else    motor_command = Stop
                  Release
```

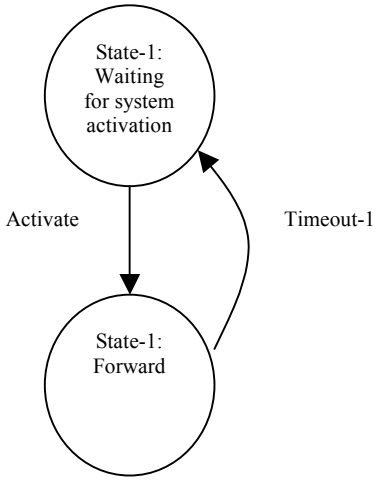

Figure 15 Cruising

```
Output:   (motor_command)
State-1:  If      system_activate = NIL
                  Release
          Else    Switch to State-2
State-2:  If      time_in_this_state > timeout_1
                  Switch to State-1
          Else    motor_command = Forward
                  Release
```

3.2 Behaviour's Stimulus Response Diagram

We are using subsumption based control architecture to implement our reactive control algorithm. This means that in the arbitration level, priority-based scheme is used to select which behaviour to run, instead of using vote scheme. The role of the leader and followers of each agent is fixed and cannot be swapped. This is illustrated in Figure 16 and 17.

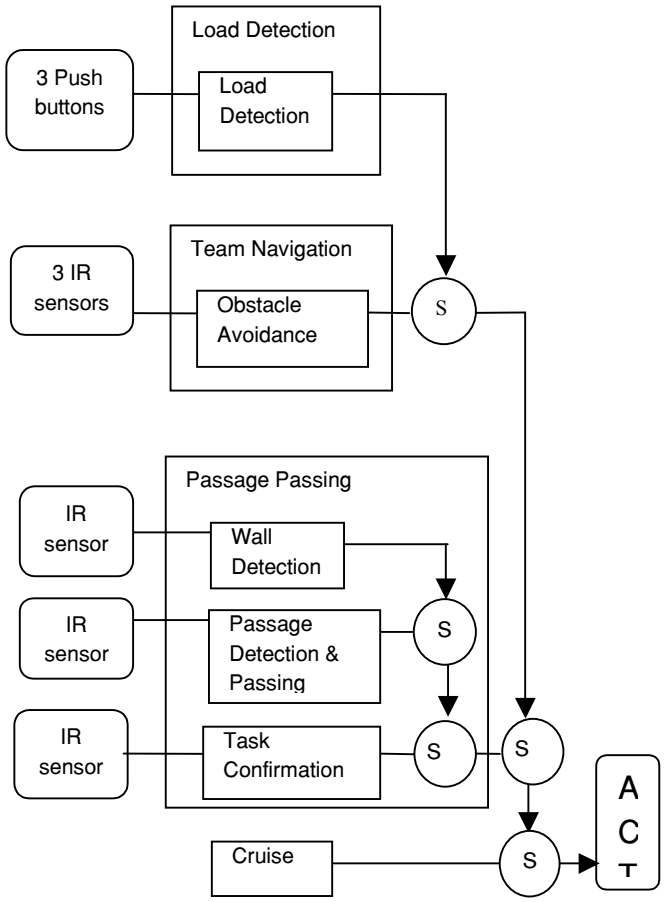

Figure 16 Leader Control Architecture

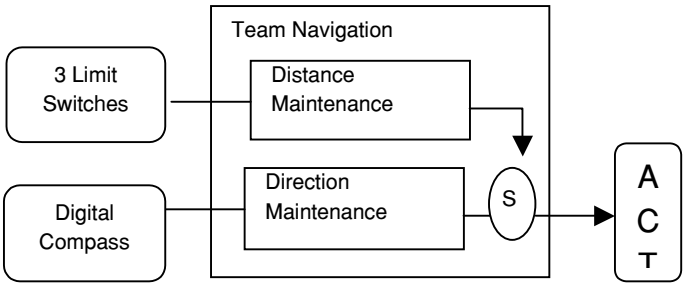

Figure 17 Follower Control Architecture

In the above control architectures, there are two aspects need to be clarified; *Response encoding method* and *Coordination method*. These two aspects are closely related and will affect the control architecture that are going to be implemented. Response encoding is the way of mapping a range of stimulus with its associated behaviors. The stimulus is a signal produced from several types of sensors. This signal then invokes the behaviour that maps with it. There are three types of mapping; *Null*, *Discrete and Continuous*. Null mapping will provide no motor response when its stimulus occurs. Discrete mapping will provide a response from an enumerated set of prescribed choices. Continuous mapping will produce a motor response that is continuous over stimulus range. In our control architecture, discrete mapping is used to produce motor response based on a particular stimulus signal. Discrete mapping is moderate between the other two in term of motor response it will produce. Null will never produce motor response, while continuous will produce continuous response, which will burden the processor tasks. Again, this discrete mapping can be further grouped into two; *Situated-action* and *Rule-based*. In a situated-action, stimulus consists of a finite set of (situation, response) pairs. Sensing provides the index for finding the appropriate situation. This method required the programmer to identify all required situations needed. This also means that, it limits the environment to consist of only selected situations. This is hardly to implement into a real world, which consists of numerous unexpected situations. In our control architecture, rule-based method is used. In this method, stimulus is represented as a collection of "IF antecedent THEN consequent". The antecedent consists of a list of preconditions while the consequent is a motor response. There are no predetermined situations needs to be recognized. This will ensure the availability of the system to be executed in the real world.

Coordination method is a method for constructing a system consisting of multiple behaviors. In our case based on Figure 16, there are six behaviors. How these behaviors are selected, and which behavior will be executed when more than one behaviors are active based on signal produced from their sensors. There are two types of coordination methods available; *Competitive* and *Cooperative*. Normally al-

though it is the rule, the discrete encoding is used with a competitive coordination method and vice versa. In competitive method, each behavior is choosen based on either *prioritisation* or *action selection*. This means that the output is one of the listing behaviors. This is different from cooperative method where it blends the outputs of multi ple behaviors This is consistent with the agent's overall goals which produce new output to represent the overall behavior. In our system, competitive method is used.

Figure 16 shows the leader's control architecture, which consists of three emergent behaviors. These three emergent behaviors are load detection, team navigation, and passage passing. These behaviors are not actually program priory but can be shown when this system runs in the real world. This emergent behavior is the consequences of the interaction amongst individual behaviors from each agent. The leader's individual behaviors are load detection (note that this is not the same with the load detection emergent behavior), obstacle avoidance, wall detection, passage detection and passing, task confirmation and cruise. Follower's individual behaviours as shown in Figure 17 are distance maintenance and direction maintenance.

4.0 RESULT

We have been investigating whether this algorithm might work. We arranged simple experiment to demonstrate purely reactive algorithm for a group navigation task under loosely-couple category. In term of hardware, each agent used similar components. Single microcontroller 68HC11A1FN is provided for each agent. For locomotion, differential drive system is used. RF-based simplex mode is used for inter-agent communication. Reflectance infrared proximity sensors are used for obstacle and navigation behaviors. The role of the leader is as a path-skipper for the followers while following the wall. The followers will follow the leader in a line pattern wherever the leader goes. The objective of the experiment has successfully achieved.

5.0 CONCLUSION

The control architecture constraints the way an autonomous robot senses, reasons and acts, thus affecting its task performance. For single agent system, there are four famous types of control architectures available, known as *deliberative*, *hybrid*, *reactive* and *based-based*. These control architectures are also valid in multi-agent system but lies between two control approaches. These control approaches are known as *centralised* and *decentralised*. Deliberative and hybrid fall under centralised control approach, while reactive and based-based in opposite domain. In our case, the decentralised approach is more efficient to be used since it reduces the complexity of the agent in area of software and hardware. The cooperation between three mobile robots has been experimented by using purely reactive control for loosely-coupled task. In this experiment, three mobile robots were programmed to navigate

in team by using explicit inter-robot communication and proximity sensors which was proven working successfully. It is interesting to realise that intelligence will arise when these simple agents interact with each other. Most researchers classified this intelligence as the emergent characteristic in the decentralised control approach. Our main philosophy in multi-agent research is stated in the statement below:

"*Complex, fast, and intelligent multi-agent system comes from an interaction of simple agent in both hardware and software domains*"

ACKNOWLEDGEMENTS

The research reported here was done at the UTM Robotics Laboratory. Support for this research was provided by Malaysian Ministry of Science, Technology and the Environment under the Intensification of Research in Priority Areas (IRPA project number: 09-02-06-0084), and also the National Science Fellowship (NSF) Award for the first author.

REFERENCES

[1] Moravec H. 1988. *Mind Children: The Future of Robot and Human Intelligence*. Harvard University Press, Cambridge, MA,
[2] McFarland D., and U. Bosser. 1993. *Intelligent Based in Animals and Robots*. MIT Press, Cambridge, MA,
[3] Arkin R.C. 1998. *Based-Based Robotics*. London, MIT Press.
[4] Mataric M. J. 1994. *From Local Interaction to Collective Intelligence*. Biology and Technology of Intelligent Autonomous Agents. 165-186.
[5] Arai T., and J. Ota. 1996. Dwarf Intelligent - A Large Object Carried by Seven Dwarves. Robotics and Autonomous Systems. 18: 149–155.
[6] Arai T., and J. Ota, "Let Us Work Together - Task Planning of Multiple Mobile Robots", In Proceedings of 1996 IEEE Conference of Intelligent Robotics and Systems (IROS'96). 1: 298–303.
[7] Rus D., Donald B., and J. Jenning. 1995. *Moving Furniture with Teams of Autonomous Robots, IEEE/RSJ IROS*. 235–242.
[8] Mataric M. J. 1992. "Based-Based Control: Main Properties and Implications", Proceedings of Workshop on Intelligent Control Systems, International Conference on Robotics and Automation. 304-312.
[9] Brooks R. A. 1991. *How to Build Complete Creatures Rather than Isolated Cognitive Simulators*. Massachusetts Institue of Technology, Artificial Intelligence Laboratory.
[10] Mataric M. J. 1997. "Based-Based Control: Examples from Navigation, Learning, and Group Based". *Journal of Experimental and Theoretical Artificial Intelligence*. 323–336.
[11] Mataric M. J. 1994. *Interaction and Intelligent Based*. Thesis, Massachusetts Institute of Technology (MIT).
[12] Dudek G., and et all, "Experiments in Sensing and Communication for Robot Convoy Navigation", IEEE 0-8186-7108-4/95.
[13] Mataric M., "Using Communication to Reduce Locality in Distributed Multi-Agent Learning", Proceedings, AAAI-97, Providence, Rhobe Island. 643–648.
[14] Brook R. 1991. *Intelligent Without Reason*. A.I Memo No. 1293, MIT AI Laboratory.
[15] Arkin R. C. 1992. Cooperation without Communication: Multi-agent Schema Based Robot Navigation. *Journal of Robotics Systems*. 351–364.
[16] Jones, J. L. and A. M. Flynn. 1993. Mobile Robot: Inspiration to Implementation. A. K. Peters, Wellesley Ma.