Modeling Situated Cogniton in Reactive Robotic Architecture

Nisal Nadeera Padukka 13000845

University of Colombo
School of Computing

2018

Modeling Situated Cogniton in Reactive Robotic Architecture

N. N. Padukka

Modeling Situated Cogniton in Reactive Robotic Architecture

N. N. Padukka

Index No. : 13000845

Supervised by

Dr. H.E.M.H.B. Ekanayake

Submitted in partial fulfillment of the requirements of the
B.Sc. in Computer Science Final Year Project (SCS4124)

University of Colombo School of Computing
Sri Lanka
May 24, 2018

Declaration

I certify that this dissertation does not incorporate, without acknowledgement, any material previously submitted for a degree or diploma in any university and to the best of my knowledge and belief, it does not contain any material previously published or written by another person or myself except where due reference is made in the text. I also hereby give consent for my dissertation, if accepted, be made available for photocopying and for interlibrary loans, and for the title and abstract to be made available to outside organizations.

Candidate Name: N. N. Padukka

.....
Signature of Candidate

May 24, 2018

This is to certify that this dissertation is based on the work of Mr.N. N. Padukka under my supervision. The thesis has been prepared according to the format stipulated and is of acceptable standard.

Supervisor Name: Dr. H.E.M.H.B. Ekanayake

.....
Signature of Supervisor

May 24, 2018

Abstract

Robots based on conventional AI architectures often failed to act in environments where the location of the objects in the environment is unknown. This is due to the unpredictability of interactions that are made by the robots within the environment. As an alternative to this approach, situated robotic architectures were introduced. Situated robotic architectures are used in the process of implementing robots in complex and dynamically changing environments.

Subsumption architecture, a kind of situated robotic architectures introduced by Rodney A. Brook succeeded in such environments and was well-received by the situated AI research community. This research focuses on enhancing the situatedness of 'subsumption architecture'. Situatedness refers to the fact that to which extent agent's cognitive process is affected by its environment. In this research, enhancement is based on how the robot's behavior is affected by its internal and external environmental condition.

Fuzzy controller based approach was used to interface selected internal and external environmental variables with the existing subsumption architecture. As a further enhancement, behaviours of subsumption architecture was implemented using neural dynamics.

Results of the experiments conclude that fuzzy controller based approach can be used to interface the internal and external environmental parameters to the subsumption architecture. Furthermore, there is significant improvement of the reactiveness of robot when behaviours of the robot are implemented using neural dynamics.

Keywords: Situatedness, Subsumption Architecture, Neural Dynamics, Fuzzy Controller

Preface

This research is based on the work done by Rodney A. Brook. The aim of the research is to enhance the situatedness of Brook's Subsumption architecture. As the initial step, performance of the Brook's subsumption architecture was evaluated compared to the conventional robotic architecture. The results are mentioned in the Chapter 5.

Proposed robotic architectures and evaluation criteria were designed in conjunction with my supervisor. In modeling the proposed architectures existing literature was referred. Implementing and evaluating the performance of the robots based on the existing and proposed architectures were my own work.

Acknowledgement

First and foremost, I would like to express my utmost gratitude to my supervisor, Dr. Hiran Ekanayake for his continuous support, encouragement, and care. Without his assistance and dedicated involvement in every step throughout the year, this study would have never been accomplished. Getting through this study required more than academic support, and I have many people to thank for listening to me at any time and being with me in much needed time. I would like to thank all the authors of those publications and articles which helped me in a lot of ways. Most importantly, none of this could have happened without my family. My loving mother, father and sisters, without your guidance and help this study would not have been possible.

Contents

List of Figures

List of Tables

Acronyms

AI	Artificial Intelligence
EB	Elementary Behavior
ICEA	Integrating Cognition, Emotion and Autonomy
AFSM	Augmented Finite State Machine
IR	Infrared
CoS	Center of Satisfaction
RNNPB	Recurrent Neural Network with Pragmatic Biased

Chapter 1

Introduction

1.1 Background to the Research

One of the most challenging problems faced by roboticists is having artificial agents to interact with the real-world environment. Using conventional AI approaches, researchers have attempted to find a solution for this by endowing the robots with exact representation of objects existing in its environment [1]. Due to the unpredictability of interactions that robots made with the environment at the time of implementation, conventional AI approaches started facing intractable issues in real-world environments.

In order to address these issues in conventional AI, situated AI approach was proposed. The goal of situated AI approach is to model entities that are autonomous in their environment. This requires designing robotic architectures from the bottom-up by focusing on the basic perceptual and motor skills required to survive. Rodney Brook's subsumption architecture is an extremely popular situated robotic architecture which allows the successful creation of real-time dynamic systems to perform in complex environments.

In furtherance, this research is focusing on enhancing the situatedness of subsumption architecture. Situatedness refers to the fact that to which extent the behavior of an agent is affected by its environment [2]. Situatedness also depends on the internal and external environmental conditions such as energy level, temperature, light condition etc [3]. In this research, enhancing the situatedness of reactive robotic architecture by integrating internal and external environment conditions into cognitive process is considered. A fuzzy controller based approach is

used to interface these variables into the cognitive process.

As a further enhancement to this situated architecture, naturalizing the subsumption architecture using a neural dynamic based approach is done.

It's not an easy task to completely replicate the situated cognition into machines [4]. Over the years, many researches have been conducted towards modeling situated cognition into artificial agents [5]. This research is a step forward in modeling situated robotic architectures which will contribute to the whole process.

1.2 Research Problem

In existing situated architectures interfacing the internal and external environment parameters to the decision-making process of the agent have not been substantially taken into consideration. In order to fill this research gap, an attempt is made in this research to enhance the situatedness of robots by incorporating both internal and external features of its environment.

The problem of existing robotic architectures in situated AI deviating from bio-inspired implementations has also been identified [6]. Implementation of a situated architecture using biologically inspired computationally tractable approach has to be undertaken to sort out this problem.

1.3 Research Questions

Now, two questions will arise in the process of filling this research gap;

RQ1 : How to enhance the situatedness of robots by incorporating both internal and external features of its situated environment using fuzzy integrations?

Internal features are features that are directly related to the situated task like energy level. External features are features that are indirectly and ubiquitously related to the situated task like temperature, social presence, light, air, and sound conditions. According to the internal and external environment conditions, strength of the behavior should be changed. In this research, above question will be addressed.

RQ2: How to naturalize the reactive robotic architecture with neurologically plausible computational approach?

Reactive robotic architectures were designed based on the studies of biological behaviors of animals like insects. Though these architectures were inspired based on biological creatures, implementation is different from the bio-inspired aspects. Since natural intelligence is based on a neuron system, implementation of robotic architecture should be addressed using a neurology based approach.

1.4 Justification for the research

Enhancing the situatedness in reactive robotic architecture is a trivial area in situated AI research. This is important in several theoretical and practical grounds of robotics which are explained below.

Adaptive Robotics - Incorporating the environmental conditions into the decision-making process will improve the adaptiveness of robots in its environment [7].

Robot Autonomy - Proposed architecture can be used to implement robots to perform autonomously in changing environments.

Human-Robot relationship - Situatedness is an important feature in human intelligence. Modeling the situated cognition for robots will contribute to build human alike robots in future, increasing the human-robot relationship.

Cognitive replacements for human in future - In future, if researchers could be able to build completely intelligent robots, it will help to reduce the work load of human being.

1.5 Methodology

As the first step, by analyzing the existing literature, research gap explained in section 1.3 was identified. In order to analyze the existing robotic architectures, a robot based on conventional robotic architecture (Robot-I) and subsumption architecture(Robot-II) was designed. The design and the functionality of the robot

are explained in section 3.2 and section 3.3 in Chapter 3. Same functionality was implemented from both architectures and the performance difference between the two architectures was analyzed.

As an enhancement to the subsumption architecture, external and internal environment conditions of the robot were incorporated into the decision-making process of the robot. Fuzzy controller based approach was used for this purpose. From this approach it was possible to integrate these variables into the robot. In this research, temperature is considered as an external environment condition parameter and energy level of the robot is considered as an internal environment condition parameter. In order to evaluate the performance of the proposed architecture, a robot based on the subsumption architecture (Robot-III) and robot based on enhanced architecture(Robot-IV) were designed and implemented.

As a further enhancement to the proposed architecture, behaviours of subsumption architecture were implemented using neural dynamics. In order to evaluate the affect of implementing behaviours using neural dynamics, perfomance of Robot-IV and Robot-V will be evaluated and compared. The below section explains the research approach in a more structured manner and figure 1.1 illustrates the research mythology as a block diagram.

Step 1 : Identify the research gaps in situated AI robotics

Step 2 : Hardware implementation of the first robot (Robot-II and Robot-II)

Step 3 : Software Implementation of the Robot-I and Robot-II

Step 4 : Evaluate the performance of the Robot-I and Robot-II

Step 5 : Hardware implementation of the second robot (Robot-III, Robot-IV, Robot-V)

Step 6 : Integrate fuzzy controllers with parameters; energy level, temperature

Step 7 : Software Implementation of the Robot-III, Robot-IV and Robot V

Step 8 : Evaluate the performance of the Robot-III, Robot-IV and Robot V

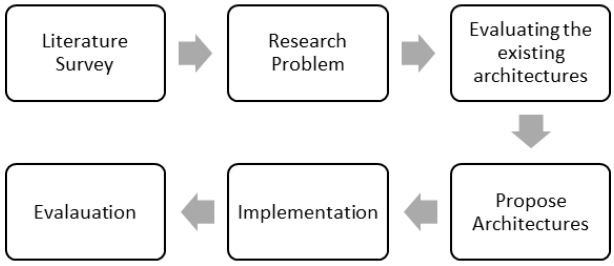

Figure 1.1: Research Methodology

1.6 Outline of the Dissertation

- Chapter 2 deals with the background of the thesis and the literature review of the study.
- In Chapter 3, a detailed description of the research design is provided. In this section, design of each experiment is discussed.
- Chapter 4 contains all the details related to the implementation.
- Chapter 5 contains results of each experiment done in this research. Afterwards, the results will be discussed.
- The final chapter contains the conclusion and future work.

1.7 Definitions

Situated cognition, situatedness, embodiment are concepts that are initially identified in the field of cognitive science and latterly adapted into other research domains. There are no universal definitions interpreted across all research domains. Therefore, these concepts are defined in this section.

1.7.1 Agent

In artificial intelligence, an agent is an autonomous entity which observes through sensors and acts upon an environment using actuators and directs its activity towards achieving goals [8].

1.7.2 Situatedness

Situatedness refers to the fact that to which extent the behaviour of an agent is affected by its environment [2].

1.7.3 Embodiment

Embodiment is the degree to which a robot can affect its environment [9][10].

1.7.4 Situated Cognition

Situated cognition is the ability of an agent to make decisons and act on the fly to the changes in environement [2].

1.8 Delimitations of Scope

In this research, when evaluating the proposed architecture our scope is limited to consider energy level as the internal environment parameter and temperature as the external environment parameter. When implementing the Robot-V using neural dynamic approach, only plausible behaviours were implemented using neural dynamic. There are learning methods like reinforcement learning, instar leaning, available in neural dynamic architectures [11][12]. In our scope learning methods will not be considered.

1.9 Conclusion

This chapter laid the foundations for the dissertation. It introduced the research problem and research questions. Then the research was justified, definitions were presented, the methodology was described and justified, the dissertation was outlined, and the limitations were given. On these foundations, the dissertation can proceed with a detailed description of the research.

Chapter 2

Literature Survey

2.1 Robotic Paradigms

A paradigm is a philosophy or set of assumptions and techniques which characterizes an approach to a class of problems [13]. Robotic paradigm can be described by the relationship between sense, plan and act. Currently there are three paradigms for organizing intelligence in robots - Hierarchical, Reactive, and Hybrid Deliberative/Reactive.

2.1.1 Hierarchical Paradigm

The Hierarchical Paradigm is the oldest paradigm used in robotic design and it works in three stages. In Hierarchical Paradigm, the robot senses the world, plans the next action, and then acts as shown in the 2.1. At each step, the robot explicitly plans the next move. This works similarly to a traditional computer program that models isolated input, output systems. When using this model roboticists face following two problems.

Figure 2.1: Convolutional AI Robotic Architecture

- **Frame Problem**

The problem of representing a real-world situation in a way that was computationally tractable became known as the frame problem [13].

- **Closed World Problem**

Roboticists claim that robot must function in the open world. It means robot should be able to function in an unknown environment [13].

2.1.2 Reactive robotic paradigm

To address the Frame Problem and Closed world problem roboticists proposed a new paradigm based on animal behaviors. Animals living in open world and simple animals such as insects, fish and frogs who exhibit intelligent behaviors without a brain were the motivation for considering animal behaviors [14].

Figure 2.2 shows Battenberg's vehicle, consists of a sensory system, motor systems, a nervous system and a body. These vehicles have two light sensors mounted in front and these sensors are directly coupled with motors. As figure illustrates, vehicle 2a moves away from light source and robot 2b moves towards the light source. This Battenberg's vehicle can be considered as the first steps towards reactive paradigm [15].

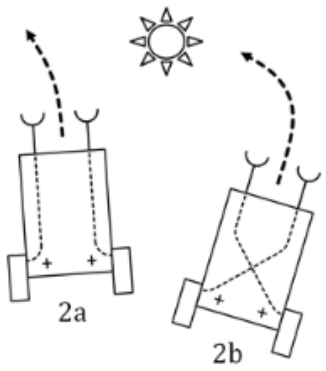

Figure 2.2: Battenberg's Vehicle

Reactive paradigm is a sense-act type of organization. The robot has multiple instances of sense-act couplings. These couplings are concurrent processes, called

behaviors. Figure 2.3 shows how multiple behaviors work concurrently in reactive paradigm [13].

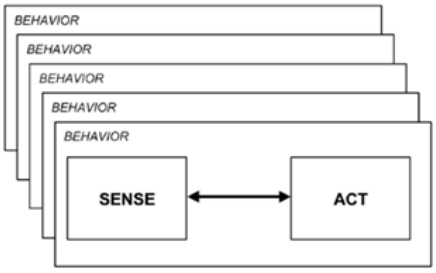

Figure 2.3: Reactive Paradigm

2.2 Subsumption architecture

Brook had a different approach in building intelligent agents from the conventional AI approach. He claims that human level of intelligence is too complex, and difficult to understand human level of intelligence. Therefore, his approach is to start from simpler level of intelligence and incrementally buildup complete intelligent agents.

Brook starts his claim with a radical hypothesis - "Representation is the wrong unit for abstraction". Based on this hypothesis, he has successfully argued that an intelligent agent should see the world as an abstract model. To explain the difference between the representation and abstraction, Brook brings a very clear example - representing a chair. A chair can be represented using its exact characteristics like color, material, size etc. Rather than representing a chair with its exact characteristics, it is possible to represent a chair by using its two major characteristics: (CAN (SIT-ON PERSON CHAIR)), (CAN (STAND-ON PERSON CHAIR). Likewise, a chair can be represented using an abstract model. Using this method, the world can be described as an abstract model. The key idea of representing the world in an abstraction form is that agent should be able to perform when the representation of the world changes.

Brook proposed a new architecture for implement intelligent agent with an abstract model of the world. He rejected the central control in intelligent agents and proposed a new layer based reactive architecture called 'Subsumption architecture' for implement intelligent agents. Rejecting the concept of central control system significantly deviates his approach from 'Conventional AI' and 'New AI'.

'Subsumption architecture' can be considered as the first situated robotic architecture. This is a layer based architecture and each layer represent a certain level of intelligence. Figure 2.4 illustrates a higher level representation of Subumption Architecture. These layers are working concurrently to perform a certain goal. Bottom layers represent the functionalities like survival behaviors and higher layers represent more cognitive functionalities. Each layer is built up using augmented finite state machines which are called modules. Modules in higher layers can suppress or inhibit modules in lower layers. Suppression is that a module at a higher level can suppress the input of a module at a lower level thereby preventing the module from seeing a value at its input. A module can also inhibit the output of a module at a lower level thereby preventing that output from being propagated to other modules. Using this method robot gets the ability to exhibit the right behavior from a combination of behaviors.

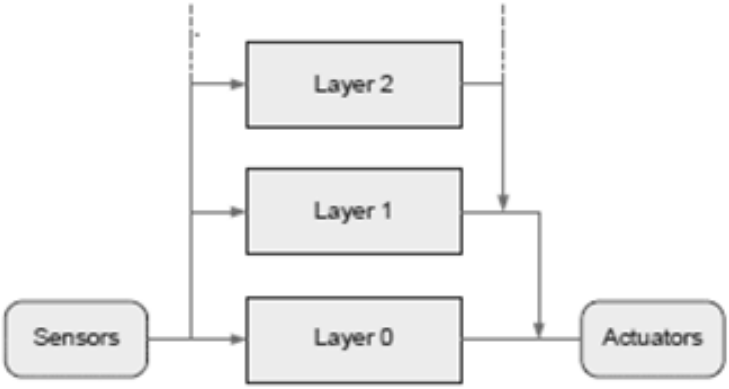

Figure 2.4: Subsumption Architecture

Subsumption architecture is promising and gives better results in real-world scenarios. Using this architecture, Brook was able to build first robot which can walk, avoid collisions and climb over obstacle in a dynamic environment.

Brook implemented the Subsumption architecture using logic devices and his implementation is completely based on software. There are software based implementations made by the researchers recently [16][17]. LEJOS – a robotics company, has implemented a JAVA thread based open-source software package for Subsumption architecture [18]. Greg Butler has proposed an Object-Oriented Design of the Subsumption architecture in his research work [19].

Recent work suggests that when increment the number of layers, it deviates from the reactive nature and becomes a priority based architecture [20]. Preliminary requirement of intelligent is learning and it is not described in the Brook's

architecture. Another issue with 'Subsumption architecture' is that robot cannot perform multiple behaviors at a time.

2.3 Interfacing Internal and External Environmental Parameters into the Decision-making Process of a Robot

In literature, there are previous attempts to interface internal and external environmental parameters into the decision-making process of the robot. In 1995, Randall D. Beer has presented a dynamical system approach for autonomous agents which interact with the environment. In this framework, an agent and its environment are modeled as two coupled dynamical systems whose mutual interaction in general jointly responsible for agent's behavior [21]. He considered that embodied agent must not be purely reactive, reflexively responding only to its immediate situation and he claimed that embodied agent should have long-term goals. In this research, Randall has considered implementing an autonomous agent which is capable of satisfying internal or external goals by its own actions while in continuous long-term interaction with the environment in which it is situated [21]. In this paper, he has presented an approach to model situated autonomous agents using coupled second order dynamical systems. One dynamical system represents the agent and other dynamical system represent the environment where the agent is situated.

Energy is considered as the most powerful constraint of autonomous mobile robotics [22]. Alberto et al have conducted several researches in implement energy autonomy for robotics. Novel approach of modeling autonomous agents using bio-mechatronic hybrid system is presented in their research. In this, bio-mechatronic hybrid model, energy cells represent as microbial fuel cells like in a living animal. The living component of the system, embedded within microbial fuel cells, relies on the availability of 'food' and 'water' in order to produce electrical energy [23]. It is essential to agent to find and collect 'food' and 'water' to survive. Based on these survival requirements robot exhibits a sequence of cognitive behaviors. Figure 2.5 illustrates a microbial fuel cell which requires substrate and water.

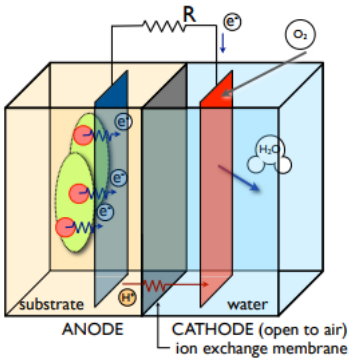

Figure 2.5: Microbial fuel cell

Alberto has represented another approach to implement energy autonomy for robotics. In this work, a robot architecture based on sensor-motor morphology is presented [24]. As in the previous case, water and fuel substrate were considered as the primary resource requirements. Robots made based on this architecture were conscious on motivation and energy level.

2.4 Fuzzy controller based subsumption behavior architecture for autonomous robotic wheelchair

The work of Zahmi et al presents the design and implementation of fuzzy controller based subsumption behavior architecture for controlling an autonomous robotic wheelchair. This wheelchair was developed with one or more basic behaviors such as goal seeking, wall following, obstacle avoidance and finding target based on different environmental conditions [25].

2.5 Neural Dynamics

In this section, attractor and repellor dynamic approach is discussed. Attractor dynamics are used to implement robots which attract towards the target and repeller dynamics are used implement robots which move away from the target.

2.5.1 Attractor and Repellor Dynamics

To control a robot using a neural dynamical system, it is required to construct a differential equation that produces the desired behavior, for instance turning the robot towards a target. For the dynamical systems approach, this means constructing a dynamics that has attractors at target states that pull the system's state towards them and repellors that push the system's state away from undesired states [26]. Figure2.6 illustrates the control of the heading direction for approaching a target. ψ represent the direction of the target and ϕ represent the current heading direction. To turn a robot toward a target that lies in the direction ψ, a dynamical system that controls the robot's heading direction ϕ can be implemented using the below equation.

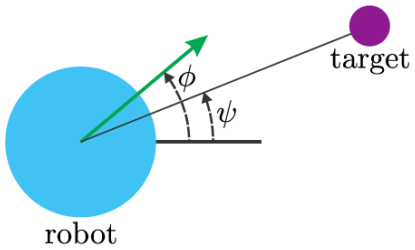

Figure 2.6: Control of the heading direction for approaching a target

$$\dot{\phi} = -\lambda(\dot{\phi} - \dot{\psi}), \lambda > 0$$

2.5.2 On-line Imitative Interaction with a Humanoid Robot Using a Dynamic Neural Network Model of a Mirror System

The work of Masato Ito has conducted an experiment on the initiative interactions between a small humanoid robot and user. In this research, as in the previous

research RNNPB model was used to implement the robot [27]. This experiment showed that after the robot learns multiple cyclic movement patterns as embedded in the RNNPB, it can regenerate each pattern synchronously with the movements of a human who is demonstrating the corresponding movement pattern in the robot [27].

2.5.3 Dynamic and interactive generation of object handling behaviors

This study presents experiments on the learning of object handling behaviors by a small humanoid robot using a dynamic neural network model, the recurrent neural network with parametric biased (RNNPB) [28]. From this model, robot was able to generate adequate ball handling motor sequences situated to the relative position between the robot's hands and the ball. The same scheme was applied to a block handling learning task where it was shown that the robot can switch among learned different block handling sequences, situated to the ways of interaction by human supporters.

The basic mechanic of this architecture is, sensory-motor patterns of guided behaviors are embedded in the RNNPB in the form of attractor dynamics[28]. The attractor represents the essential spatio-temporal structure of the target behavior. Learning multiple behavior pattern is realized by switching different attractor dynamics [28]. The Figure 2.7 illustrates the RNNPB architecture which was used in this research.

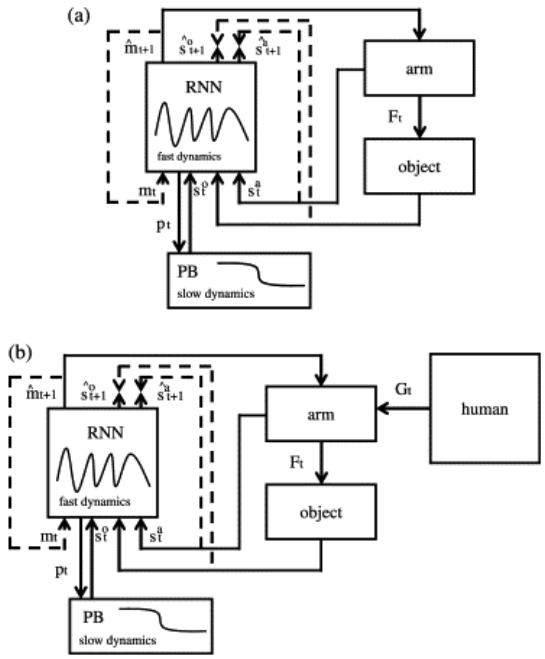

Figure 2.7: Robotic architecture using RNNPB

2.5.4 A neural-dynamic architecture for behavioral organization of an embodied agent

In this work, researchers have presented a neural-dynamic behavior based robot using NOA experimental robot which can grasp a cup using robot arm. This architecture is very specific to task and has following behaviors; find color : find the blue color cup, move-end effecter : move the robot arm to the location, open gripper and close gripper. They have used Dynamic Field Theory (DFT) for the neural dynamic implementation [29].

2.6 Cognitive affective architecture

Figure 2.8 shows the work of the larger European cognitive robotics group called ICEA. This is an abstract model of the hypothalamus and brainstem which deal with 'low-level mechanisms' of human body such as drives and bio-regulations. The primary aim of this research is to computationally model, at different level of abstraction, different brain structures and their interactions [30].

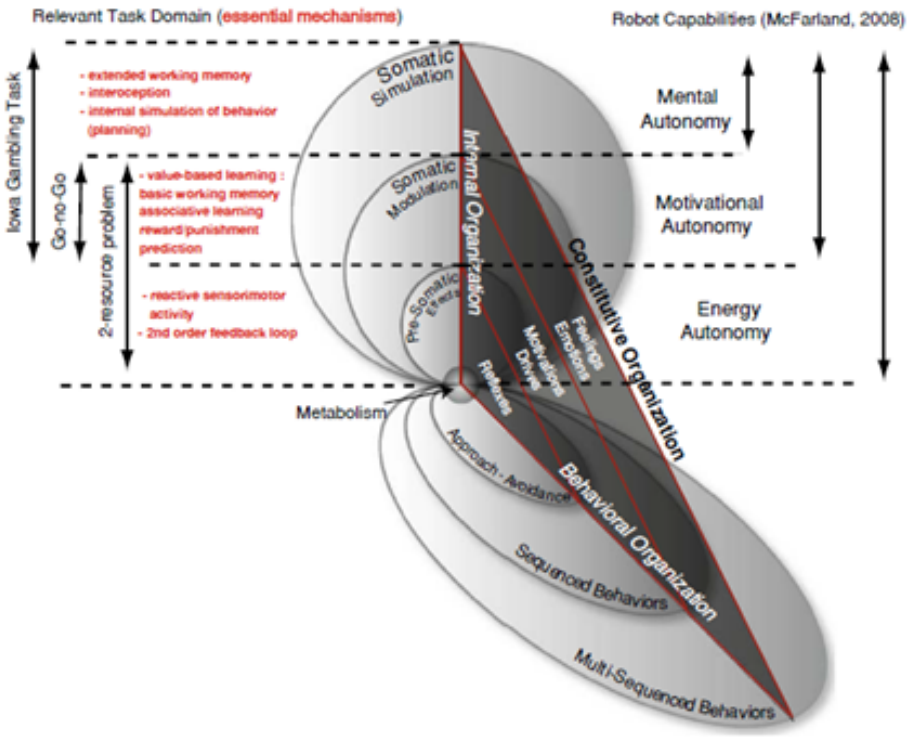

Figure 2.8: Cognitive affective architecture of brain

Chapter 3

Design

3.1 Introduction

This chapter provides a detailed description of the research design of the project. It describes the design of the research including the design of reactive robotic architectures implemented and tested in this research, evaluation criteria and design assumptions.

3.2 Design of the Robot I

3.2.1 Introduction

As the initial step, an obstacle avoidance robot was implemented using the convolution robotic architecture. This robot was used to evaluate the performance of existing convolution robotic architecture. Figure 3.1 shows the design of the robot which was implemented. It has an arm with an Ultra Sonic sensor. From the Ultra Sonic sensor, it can detect distance to objects in front of the robot. Movement direction was decided using these sensor readings. Raspberry Pi 2 B+ model was used as the cental processing unit.

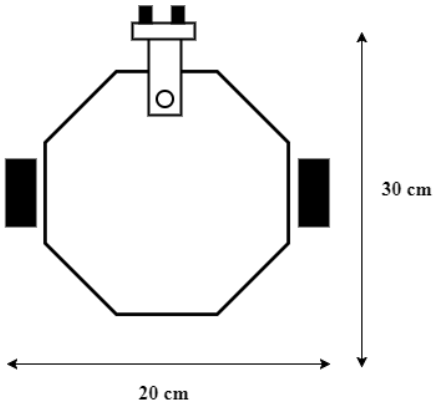

Figure 3.1: Design of the Robot I

3.2.2 Functionality of the Robot

The functionality of the robot was to avoid obstacles and move forward in arena as shown in the figure 3.2.

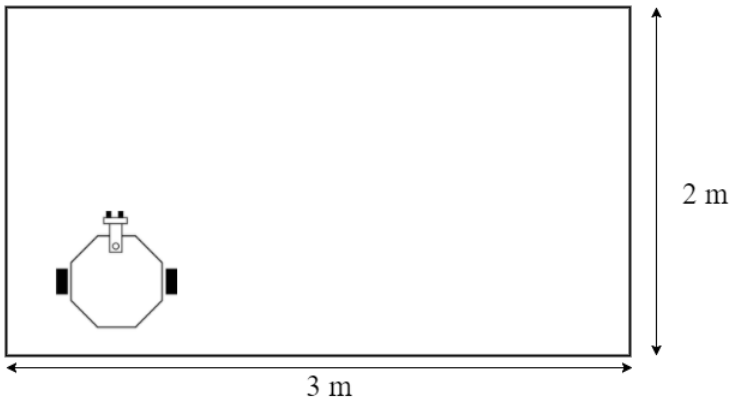

Figure 3.2: Arena for the Robot I

3.2.3 High-level Architecture

Figure 3.3 illustrates the high level architecture design of the implemented robot. Central processing unit continuously looking for an obstacle and if there are no obstacles infront of the robot, robot moves forward. When an obstacle is met, it turns away from the obstacle by 90^0 to right or left. If there are obstacles in right and left side of the robot, it turns back by 180^0.

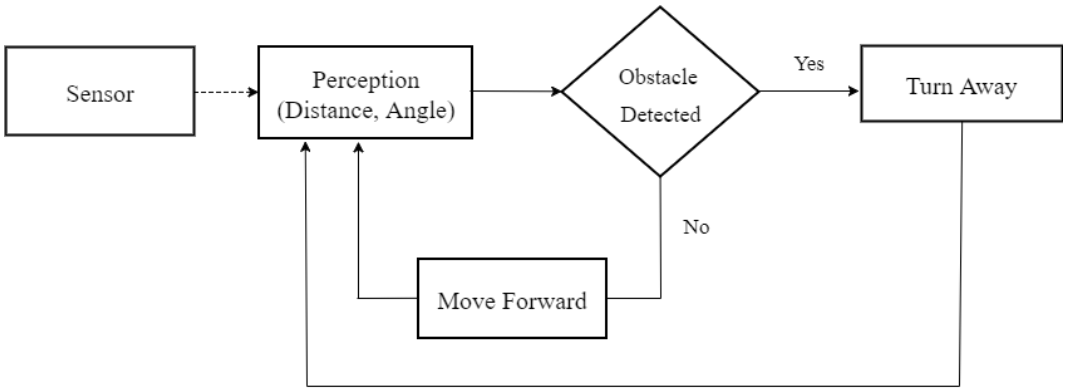

Figure 3.3: High Level Architecture of the Robot I

3.3 Design of the Robot II - Obstacle avoidance using Subsumption architecture

3.3.1 Introduction

Using the same hardware of the Robot I, a subsumption architecture based robot was implemented. This robot was used to evaluate the performance of subsumption architecture compared to coventional architecture. As shown in the figure 3.1 same robot design was used to implement the robot. The difference is in the architecture, which is a thread based subsumption architecture. This robot exhibits two behaviours,

- **Obstacle Avoidance Behaviour**

Obstacle avoidance behavior gives the ability to the robot to avoid obstacles to prevent collisions. Obstacle avoidance behaviour gets the control of the robot when an obstacle is detected.

- **Moving Forward Behaviour**

In this behaviour, robot makes movemets from its current location.

3.3.2 Functionality of the Robot

The functionality of the robot was to avoid obstacles and move forward in arena as shown in the figure 3.2.

3.3.3 High-level Architecture

Figure 3.4 illustrates the high level architecture of the implemented robot.

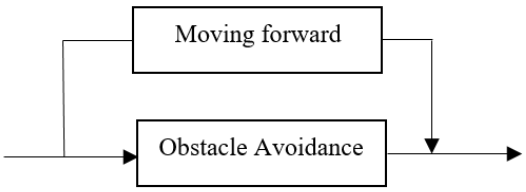

Figure 3.4: High-Level Architecture of the Robot II

3.4 Design of the Robot III - Robot using Subsumption architecture

3.4.1 Introduction

This robot was designed to evaluate the performance of subsumption architecture. It is equipped with two ultra sonic sensors mounted on two servo motors, camera, IR Sensor panel, temperature Sensor, sensors to measure the energy consumption and wheel encoders. This robot was designend to implement in two layers. Figure 3.5 illustates the design of the bottom layer of the robot. Figure 3.6 illustrates the upper layer of the robot. The same robot is used to evaluate the performance of the architectures. This robot exhibits four behaviours;

- **Obstacle Avoidance Behaviour**

Obstacle avoidance behavior gives the ability to the robot to avoid obstacles to prevent collisions. Obstacle avoidance behaviour gets the control of the robot when obstacle is detected.

- **Path Following Behaviorr**

In this behaviour, robot moves along the path using its IR sensors. It works by adjusting the speed of the motors to remain in the path.

- **Path Finding Behaviour**

When a path discontinuity occurs, this behavior should get activated and robot should move randomly across the arena to get back into the path.

- **Finding the Target Behaviour**

Finding the target behavior is used to find the target location. In this part, real-time image processing is used to find the target. When the robot reaches to the end location, robot should stop and give a notification.

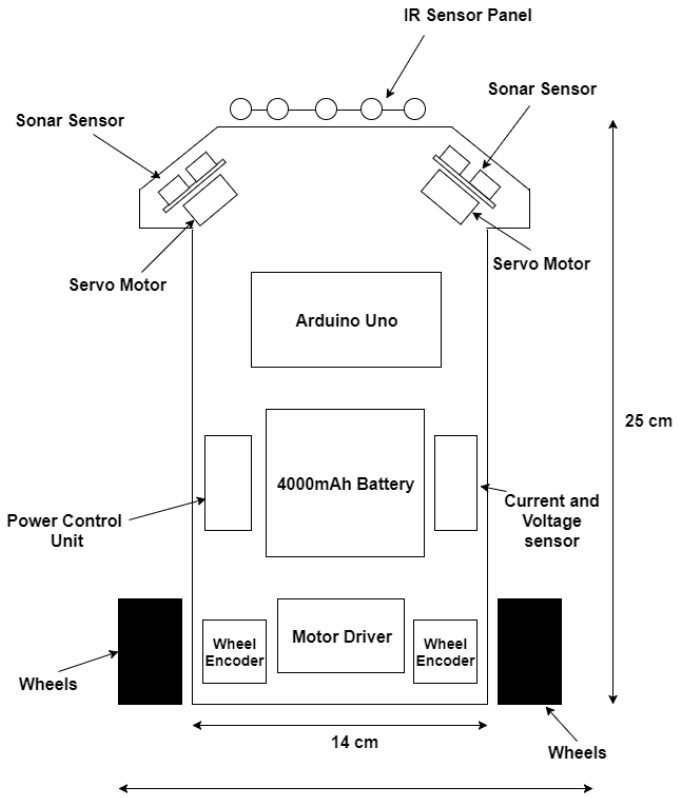

Figure 3.5: Design of the Robot III - Bottom Layer

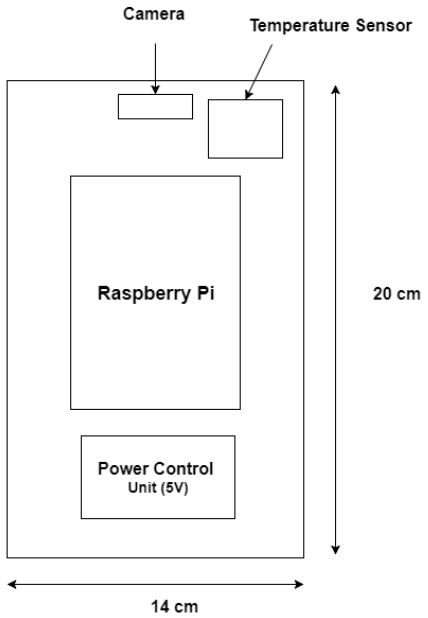

Figure 3.6: Design of the Robot III - Upper Layer

3.4.2 High-level Architecture

Figure 3.7 illustrates the high level architecture of the designed robot.

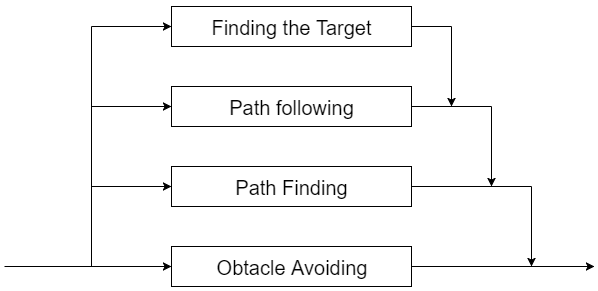

Figure 3.7: High-Level Architecture of the Robot III

3.4.3 Functionality of the Robot

The functionality of the robot is to follow the line, avoid obstacles and reach to the destination in arena as shown in the figure 3.8.

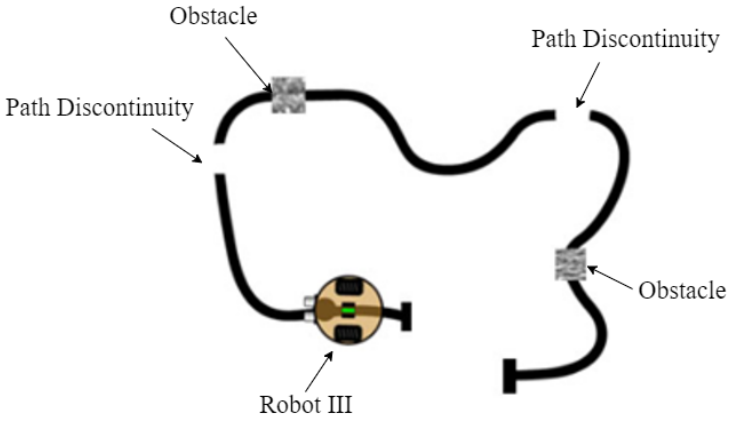

Figure 3.8: Arena for the Robot III

3.5 Design of the Robot IV - Enhanced situated robot using fuzzy integrations to Subsumption architecture

3.5.1 Introduction

This robot is used to evaluate the performance of the proposed architectures compared to the existing architecture. This robot has the same hardware as the Robot III. Functionalities to measure energy-level, temperature and distance are implemented in this robot. As mentioned in the previous chapter, energy-level is considered as the internal environmental factor and temperature is considered as the external environmental factor.

3.5.2 High-level Architecture

Figure 3.9 illustrates the high level architecture of the designed robot.

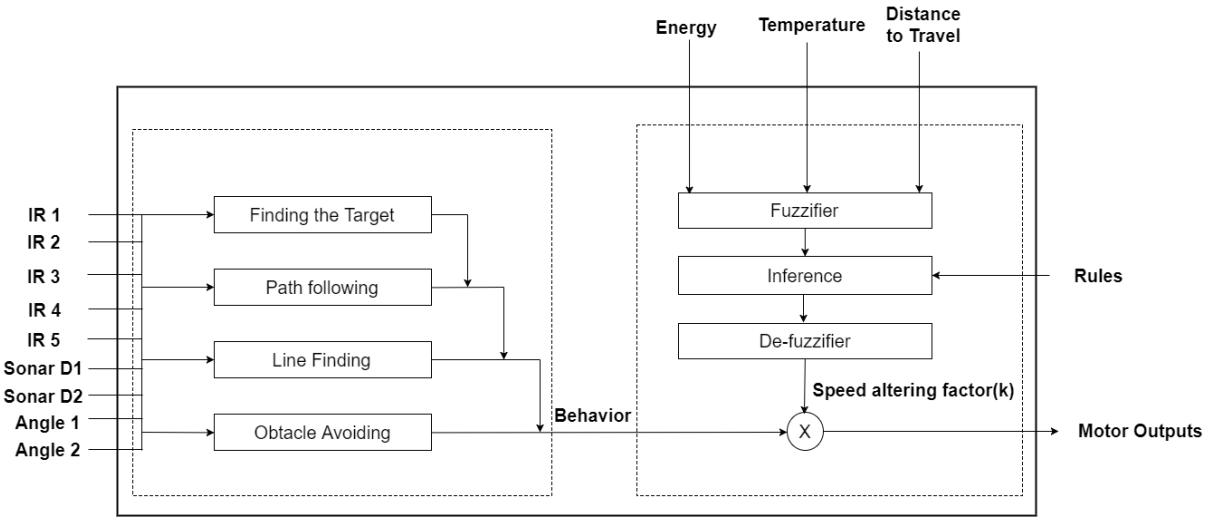

Figure 3.9: High-Level Architecture of the Robot IV

3.5.3 Design of the Fuzzy Controller

As explained in the previous sections, energy level and temperature of the robot will be considered to enhance the situatedness of the behavior based architecture. In order to integrate these variables into decision making process, each behavior

will be interfaced with a fuzzy controller. Inputs to the fuzzy controller will be energy level, temperature and remaining distance. Based on these parameters, the strength of the currently active behavior will be altered to achieve the final goal.

3.6 Design of the Robot V - Enhanced situated robot using fuzzy integrations and attractor dynamics approach

3.6.1 Introduction

In this architecture, path following behaviour is designed to implement using attracter dynamic approach. This architecture is compared with the architecture of Robot V to evaluate the improvement of using neural dynamics in naturalizing the reactive robotic architecture.

3.7 Design Assumptions

This section provides the design assumptions that have been identified as follows;

- As the internal environmental condition of the robot, only the energy level of the robot will be considered.
- As the external environmental condition, only the temperature of the environment will be considered.
- For the evaluating purposes, robot will be designed to react to slight variations of the temperature.

.

Chapter 4

Implementation

This chapter mainly focuses on the implementation of five robots which described in chapter 3. In addition to the implementation, different tools used in the project will be discussed.

4.1 Implementation of the Robot I

Figure 4.1 illustrates the implemented robot which was used to evaluate the designs of Robot I and Robot II. This robot has a raspberry pi 2B+ model as the processing unit and equipped with a rotatable ultrasonic sensor to detect obstacles around the robot.

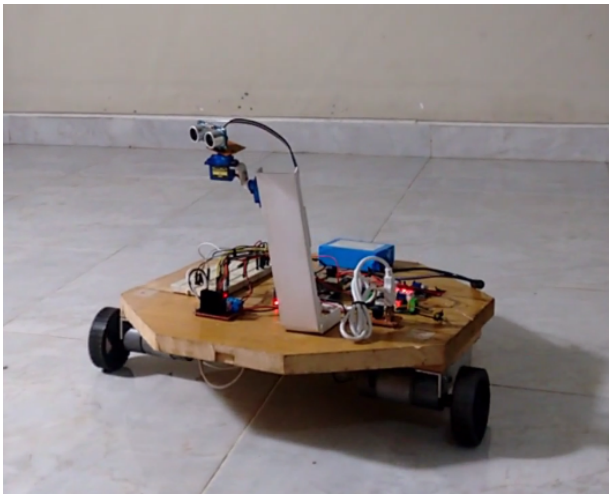

Figure 4.1: Implemented Robot I

4.1.1 Hardware Implementation

In this robot Raspberry PI 2 B+ model was used as the processing unit. This processor has a 900MHz quad-core ARM Cortex-A7 processor and 1GB physical memory. Raspbian was used as the operating system. These specifications are essential for the multi-processing requirement of the implementation. SR04 ultrasonic sensor was used to detect obstacles.

Figure 4.2: Raspberry PI 2 B+

4.1.2 Software Implementation

In this architecture, robot was implemented using conventional robotic architecture. Robot continuously scans for obstacles. If an obstacle is detected, robot moves away from the obstacle and if no obstacles are present robot moves forward. Algorithm 1 illustrates the pseudocode of the implementation.

Algorithm 1 Obstacle Avoidance Robot-I

```
1: while true do
2:   if ObstacleDetected() then
3:     TurnAway()
4:   else
5:     MoveForward()
6: end
```

4.2 Implementation of the Robot II

For this implementation same hardware as in the Robot-I was used. For this robot, same functionality was implemented using subsumption architecture.

4.2.1 Software Implementation

The two behaviors described in the Section 3.3 were implemented using two Python threads. Obstacle avoidance behavior is the survival behavior and when an obstacle detected, robot suppresses the moving forward behavior and avoid the obstacle. Algorithm 2 illustrates the implementation of the Obstacle Avoidance behavior and Algorithm 3 illustrates the implementation the of Moving Forward behavior. Algorithm 4 represents the thread based implementation of the Subsumption Architecture.

Algorithm 2 Behaviour 1 : Obstacle Avoidance

```
1: procedure TakeControl
2:     rotateSonar(0^0)
3:     d1 ← readSonar
4:     rotateSonar(−45^0)
5:     d2 ← readSonar
6:     rotateSonar(+45^0)
7:     d3 ← readSonar
8:     if distance < d1,d2,d3 then
9:         return True
10: procedure Suppress
11:     if suppressed! = true then
12:         return true
13: procedure Action
14:     rotateSonar(−90^0)
15:     if distance < d1 then
16:         TurnLeft()
17:     rotateSonar(+90^0)
18:     if distance < d2 then
19:         TurnRight()
```

2

Algorithm 3 Behaviour 2 : Move Forward

```
1: procedure TakeControl
2:     if suppressed! = true then
3:         return true
4: procedure Suppress
5:     suppressed ← true
6:     return true
7: procedure Action
8:     MoveForward()
```

Algorithm 4 Subsumption Architecture

```
1: B1 ← new Obstacle Avoidance
2: B2 ← new MoveForward
3: procedure BEHAVIOUR 1
4:   if B1.TakeControl() = true then
5:     B2.Suppress()
6:     B1.Action()
7:     B1.Supress()
8:
9: procedure BEHAVIOUR 2
10:  if B2.TakeControl() = true then
11:    B1.Suppress()
12:    B2.Action()
13:    B2.Supress()
14:
15: thread.start(BEHAVIOUR 1)
16: thread.start(BEHAVIOUR 2)
```

4.3 Implementation of the Robot-III

Figure 4.3 illustrates the implemented robot which was used to evaluate the designs of Robot III, Robot IV and Robot V. This robot has a raspberry pi as the processing unit, two rotatable ultrasonic sensors to detect obstacles, five IR sensors to detect the path, a temperature sensor, current sensor and a camera to identify the target location.

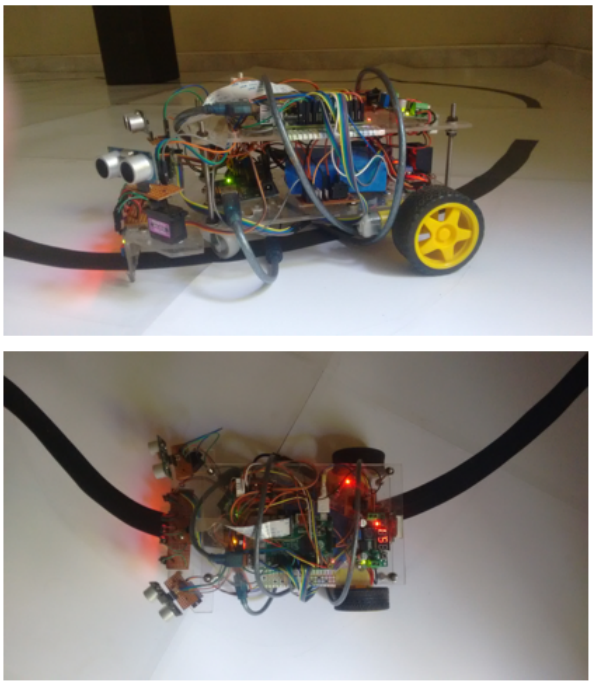

Figure 4.3: Implemented Robot III

4.3.1 Hardware Implementation

In this robot Raspberry PI 2 B+ model was used as the processing unit. This processor has a 900MHz quad-core ARM Cortex-A7 processor and 1GB physical memory. Raspbian was used as the operating system. These specifications are essential for the multi-processing requirement of the implementation. Two SR04 ultrasonic sensors were used to detect obstacles. Five IR sensor modules were used to detect the path. An ACS712 Current Sensor is being used to measure the current consumption. DH11 sensor module is being used to measure the temperature in the environment. 4000mAh battery was used to power the robot. Raspberry PI Rev 1.3 Camera was interfaced with the robot to detect the end location. Wheel encoders were used to measure the distance travelled by the robot.

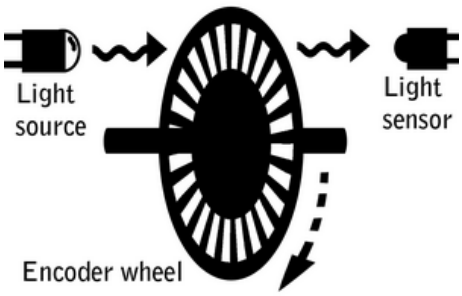

Figure 4.4: Wheel encoders

4.3.2 Software Implementation

The four behaviors described in the Section 3.4 were implemented using four Python threads. Obstacle avoidance behavior is the survival behavior and when an obstacle detected, robot suppresses the moving forward behavior and avoids the obstacle. Algorithm 5 illustrates the implementation of the Obstacle Avoidance behavior and Algorithm 6 illustrates the implementation of the path following behaviour. Algorithm 7 illustrates the implementation of the finding target behavior. Algorithm 8 represents the thread based implementation of the Subsumption Architecture. Thread based subsumption architecture was implemented considering the principles of LEJOS software based subsumption architecture [31].

Algorithm 5 Behaviour 1 : Obstacle Avoidance

```
1: procedure TakeControl
2:    rotateSonar1(0^0)
3:    rotateSonar2(0^0)
4:    d1 ← readSonar
5:    d2 ← readSonar
6:    rotateSonar1(−45^0)
7:    rotateSonar2(−45^0)
8:    d3 ← readSonar
9:    d4 ← readSonar
10:   rotateSonar1(+45^0)
11:   rotateSonar2(+45^0)
12:   d5 ← readSonar
13:   d6 ← readSonar
14:   if distance < d1,d2,d3,d4,d5,d6
      then
15:      return True
16: procedure Suppress
17:   if suppressed! = true then
18:      return true
19: procedure Action
20:   TurnLeft()
21:   MoveForward()
22:   TurnRight()
23:   MoveForward()
24:   TurnRight()
25:   MoveForward()
26:   TurnLeft()
```

2

Algorithm 6 Behaviour 2 : Path Following

```
1: procedure TakeControl
2:   if suppressed! = true then
3:     return true
4: procedure Suppress
5:   suppressed ← true
6:   return true
7: procedure Action
8:   if IR1, IR2, IR3, IR4, IR5 = (0,0,1,0,0) then
9:     MoveForward()
10:  if IR1, IR2, IR3, IR4, IR5 = (0,1,1,1,0) then
11:    MoveForward()
12:  if IR1, IR2, IR3, IR4, IR5 = (0,1,1,0,0) then
13:    SlightTurnLeft()
14:  if IR1, IR2, IR3, IR4, IR5 = (1,0,0,0,0) then
15:    SharpTurnLeft()
16:  if IR1, IR2, IR3, IR4, IR5 = (0,1,0,0,0) then
17:    TurnLeft()
18:  if IR1, IR2, IR3, IR4, IR5 = (1,1,0,0,0) then
19:    SharpTurnLeft()
20:  if IR1, IR2, IR3, IR4, IR5 = (1,1,1,0,0) then
21:    SlightTurnLeft()
22:  if IR1, IR2, IR3, IR4, IR5 = (0,0,1,1,0) then
23:    SlightTurnRight()
24:  if IR1, IR2, IR3, IR4, IR5 = (0,0,0,0,1) then
25:    SharpTurnRight()
26:  if IR1, IR2, IR3, IR4, IR5 = (0,0,0,1,0) then
27:    TurnRight()
28:  if IR1, IR2, IR3, IR4, IR5 = (0,0,0,1,1) then
29:    SharpTurnRight()
30:  if IR1, IR2, IR3, IR4, IR5 = (0,0,1,1,1) then
31:    SlightTurnRight()
```

Algorithm 7 Behaviour 4 : Finding the Target

```
1: procedure TakeControl
2:   if suppressed! = true and TargetFound() = true then
3:     return true
4: procedure Suppress
5:   suppressed ← true
6:   return true
7: procedure Action
8:   halt
```

Algorithm 8 Subsumption Architecture

```
1: B1 ← new Obstacle Avoidance
2: B2 ← new Path Finding
3: B3 ← new Path Following
4: B4 ← new Finding the Target
5: procedure Behaviour 1
6:   if B1.TakeControl() = true then
7:     B2.Suppress()
8:     B3.Suppress()
9:     B4.Suppress()
10:    B1.Action()
11:    B1.Supress()
12: procedure Behaviour 2
13:   if B2.TakeControl() = true then
14:    B3.Suppress()
15:    B4.Suppress()
16:    B2.Action()
17:    B2.Suppress()
18: procedure Behaviour 3
19:   if B3.TakeControl() = true then
20:    B4.Suppress()
21:    B3.Action()
22:    B3.Supress()
23: procedure Behaviour 4
24:   if B2.TakeControl() = true then
25:    B4.Action()
26:    B2.Supress()
27: thread.start(BEHAVIOUR 1)
28: thread.start(BEHAVIOUR 2)
29: thread.start(BEHAVIOUR 3)
30: thread.start(BEHAVIOUR 4)
```

2

4.4 Implementation of the Robot-IV

This is the robot which was implemented to evaluate the performance of the fuzzy controller based approach. This robot can measure environment temperature, energy consumption and travelling distance. Fuzzy rules mentioned in the Table 4.1 were implemented in this experiment.

Table 4.1: Fuzzy Rules

Rule	Energy Level	Distance to Travel	Temperatue	Speed
1	High	High	Low	High
2	Medium	Low	High	Low
3	Low	Low	Low	Medium

A web based user interface was integrated with this robot to observe the outputs. Through this interface, environmental parameters can be simulated and view results. Figure 4.5 illustrates the implemented user interface for this robot.

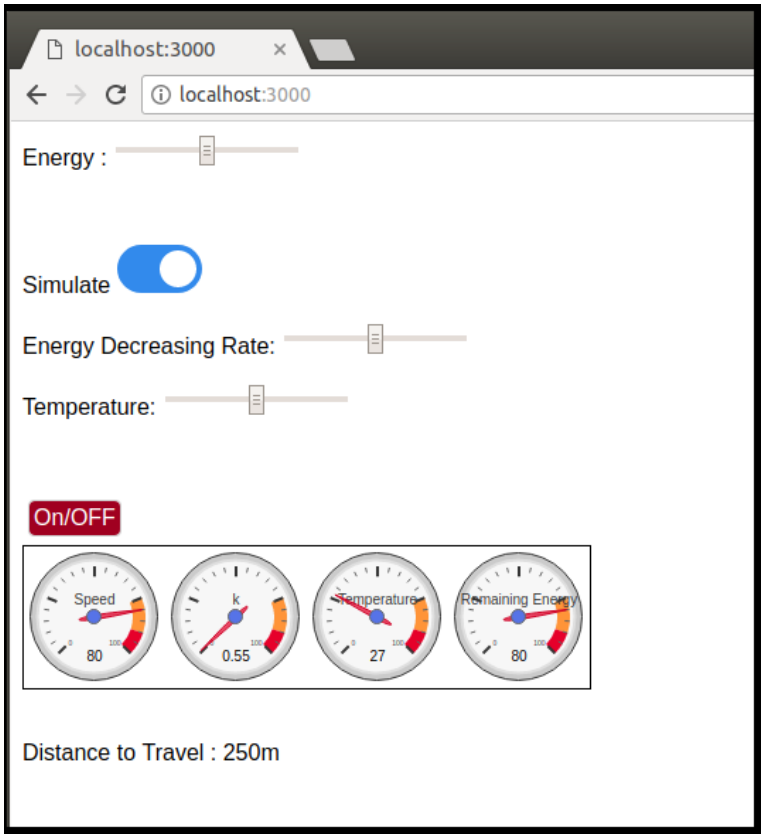

Figure 4.5: User interface

4.4.1 Software Implementation

The implementation of fuzzy controller system is illustrated by the Algorithm 9. In this experiment, range of distance is taken as 0 to 300, range of temperature is taken as 20 to 30 and range of energy is taken as 0 to 100.

Algorithm 9 Fuzzy System

procedure MEMBERSHIP FUNCTIONS

distance low $\leftarrow [0, 0, 150]$
distance medium $\leftarrow [0, 150, 300]$
distance high $\leftarrow [150, 300, 300]$

energy low $\leftarrow [0, 0, 50]$
energy medium $\leftarrow [0, 50, 100]$
energy high $\leftarrow [50, 100, 100]$

temperature low $\leftarrow [20, 20, 25]$
temperature medium $\leftarrow [20, 25, 30]$
temperature high $\leftarrow [25, 30, 30]$

speed low $\leftarrow [0, 0, 0.5]$
speed medium $\leftarrow [0, 0.5, 1]$
speed high $\leftarrow [0.5, 1, 1]$

procedure RULE APPLICATION
rule 1 ← max(energy high,distance high, speed high)
speed activation high ← min(rule1, speed high)
rule 2 ← max(energy medium, distance low, speed low)
speed activation high ← min(rule 2, speed high)
rule 3 ← max(energy low,distance low, speed low)
speed activation medium ← min(rule 3, speed medium)

procedure DEFUZZFICATION
aggregated ← max(speed activation low,speed activation medium, speed activation high)
speed alteration factor ← defuzz(aggregated)

4.5 Implementation of the Robot-V

As described in the Section 3.6, in this experiment, path following behavior was implemented using attractor dynamic approach. Figure 4.6 illustrates the angle difference, heading and target direction of robot with different position in the path. It was required to implement a sine attractor dynamic in order to turns the robot

towards the desired direction over the different orientations of the robot.

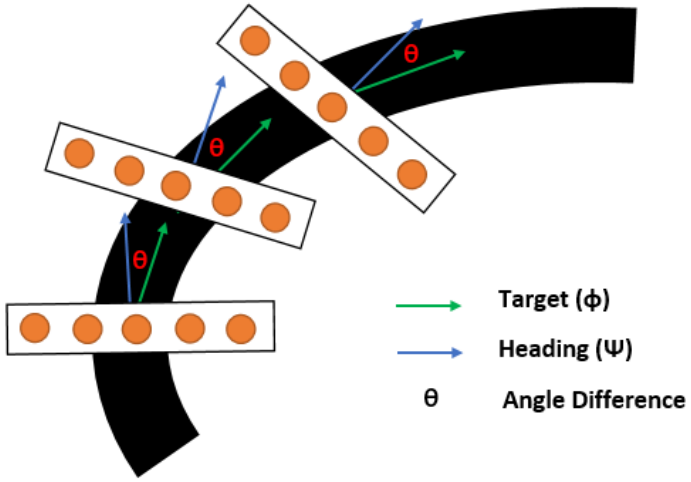

Figure 4.6: IR sensor panel orientation

Instead of a linear system, it is more practical to use a sine curve(see figure 4.7) for the dynamical system because it is the periodic structure of the behavioral variable and does not produce increasingly large values for larger angles. This means that the robot will always turn toward the target using the shortest path.

To turn a robot towards a target that lies along the path, a dynamical system that controls the robot's heading direction was implemented. Line 20-24 in algorithm 10 shows the implemented sine attractor dynamic.

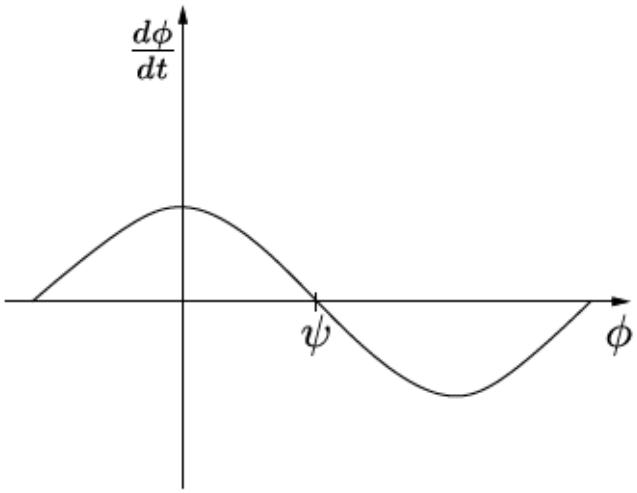

Figure 4.7: Sine attractor dynamic

Algorithm 10 Behaviour 2 : Path Following

1: **procedure** ACTION(SPEED)
2: $lamda \leftarrow 0.3$
3: **if** $IR1, IR2, IR3, IR4, IR5 = (0,0,1,0,0)$ **then**
4: $Angle\ Difference \leftarrow 0$
5: **if** $IR1, IR2, IR3, IR4, IR5 = (0,1,1,1,0)$ **then**
6: $Angle\ Difference \leftarrow 0$
7: **if** $IR1, IR2, IR3, IR4, IR5 = (0,1,1,0,0)$ **then**
8: $Angle\ Difference \leftarrow -10$
9: **if** $IR1, IR2, IR3, IR4, IR5 = (1,1,1,0,0)$ **then**
10: $Angle\ Difference \leftarrow -15$
11: **if** $IR1, IR2, IR3, IR4, IR5 = (0,1,0,0,0)$ **then**
12: $Angle\ Difference \leftarrow -20$
13: **if** $IR1, IR2, IR3, IR4, IR5 = (0,0,1,1,0)$ **then**
14: $Angle\ Difference \leftarrow +10$
15: **if** $IR1, IR2, IR3, IR4, IR5 = (0,0,1,1,1)$ **then**
16: $Angle\ Difference \leftarrow +15$
17: **if** $IR1, IR2, IR3, IR4, IR5 = (0,0,0,1,0)$ **then**
18: $Angle\ Difference \leftarrow +20$
19: **if** $IR1, IR2, IR3, IR4, IR5 = (0,0,0,0,1)$ **then**
20: $Angle\ Difference \leftarrow +30$
21: $diPhi \leftarrow lamda \times (-sin(\pi \times (AngleDifference)/180))$
22: $vmmpersecond \leftarrow diPhi/\pi \times 53$
23: $vpulsespersecond \leftarrow vmmpersecond/0.13$
24: *drive.SetSpeed(-vpulsespersecond + speed, vpulsespersecond + speed)*

Chapter 5

Results and Evaluation

As explained in the research design section, to evaluate the performance of an architecture it is required to compare the architecture with existing architectures. Therefore, the evaluation of this research is organized into three parts.

5.1 Conventional Architecture Vs. Subsumption Architecture

In the first part, research was focused on evaluating the performance of the subsumption architecture compared to the conventional robotic architecture. For this purpose, Robot I and Robot II were used. Robot I was implemented using conventional architecture and Robot II was implemented using the Subsumption architecture to perform the same task.

It is clear that when performing multiple behaviors, subsumption architecture performs spontaneously while conventional architecture works in *'move-think-move-think' approach*. Robot I which was implemented using conventional architecture first scans obstacles, then moves the distance, again scan obstacles and again moves. Robot II which was implemented from subsumption architecture, scans obstacles while moving which leads to suppress the moving forward behaviour and activates the obstacle avoidance behaviour when an obstacle detected. Figure 5.1 illustrates avoiding a obstacle by the Robot II. A demonstration video is available at this url [1] on youtube which gives a clear idea of how these two artichectures work.

[1] https://www.youtube.com/watch?v=2sb3u9hGOMo

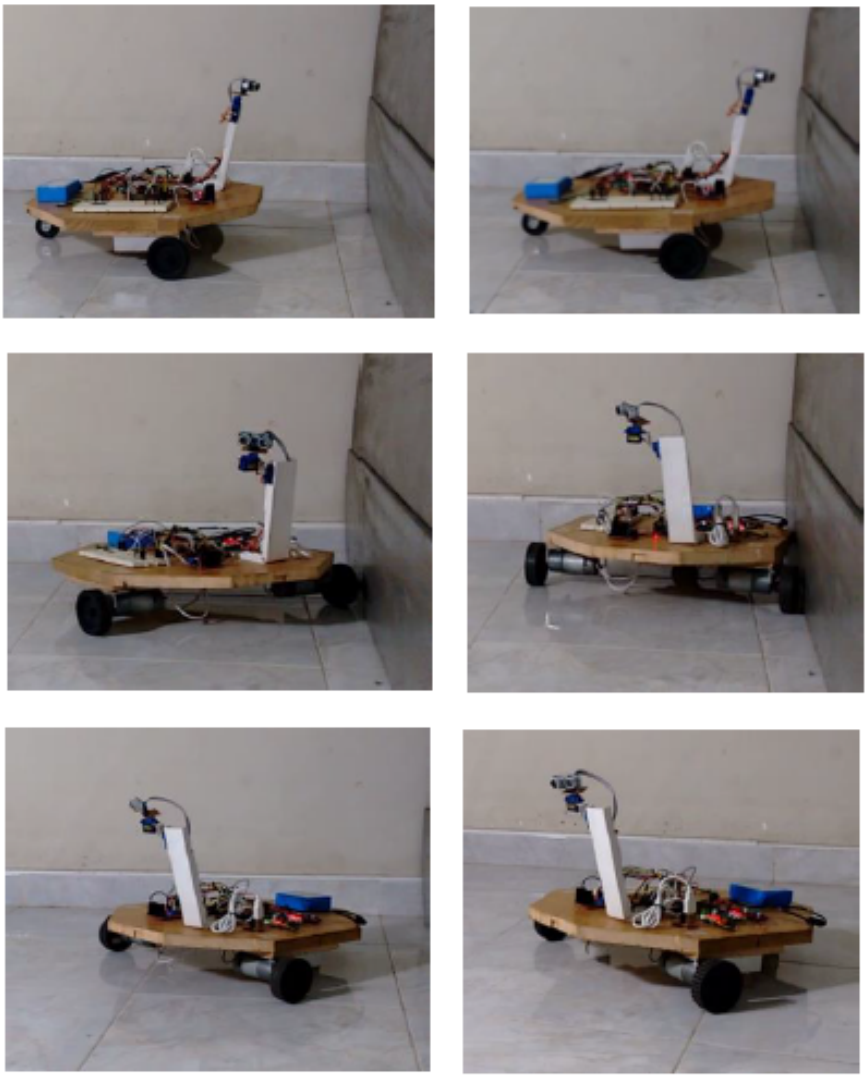

Figure 5.1: Avoiding an Obstacle

5.2 Subsumption Architecture Vs. Fuzzy Integrated Subsumption Architecture

In this part, research is focused on evaluating the performance of proposed fuzzy integrated subsumption architecture compared to the subsumption architecture. For this purpose, architectures of Robot III and Robot IV were used. Figure 5.5 illustrates the arena which was used to evaluate the architectures.

Robot III functioned in the same speed while Robot IV exhibited different speeds with respect to changes in internal and external environment variables; energy level, distance to travel and temperature. Figure 5.2 illustrates the fuzzifi-

cation of input variables.

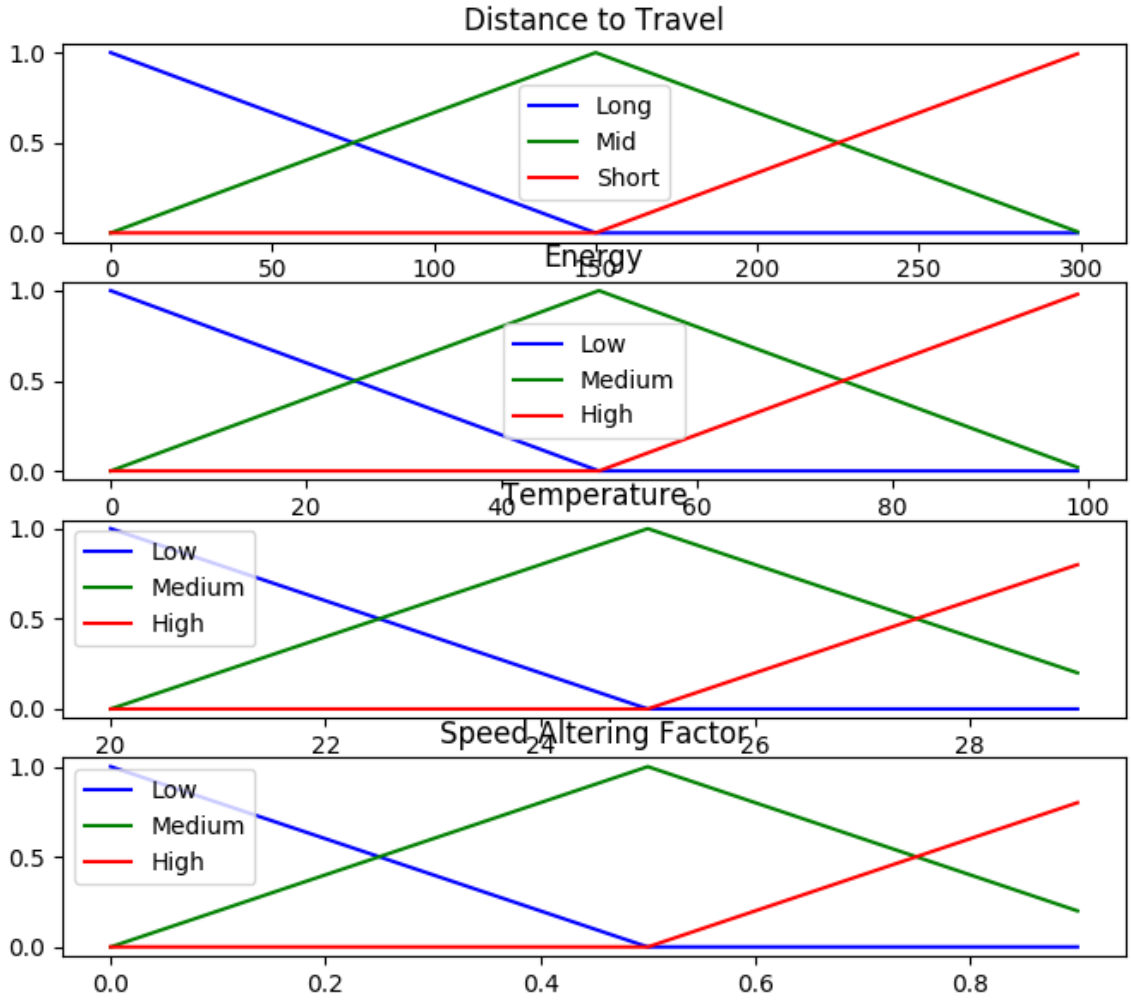

Figure 5.2: Fuzzification of variables

Figure 5.3 and figure 5.4 illustrate the rule application and de-fuzzification of output variables after rule application.

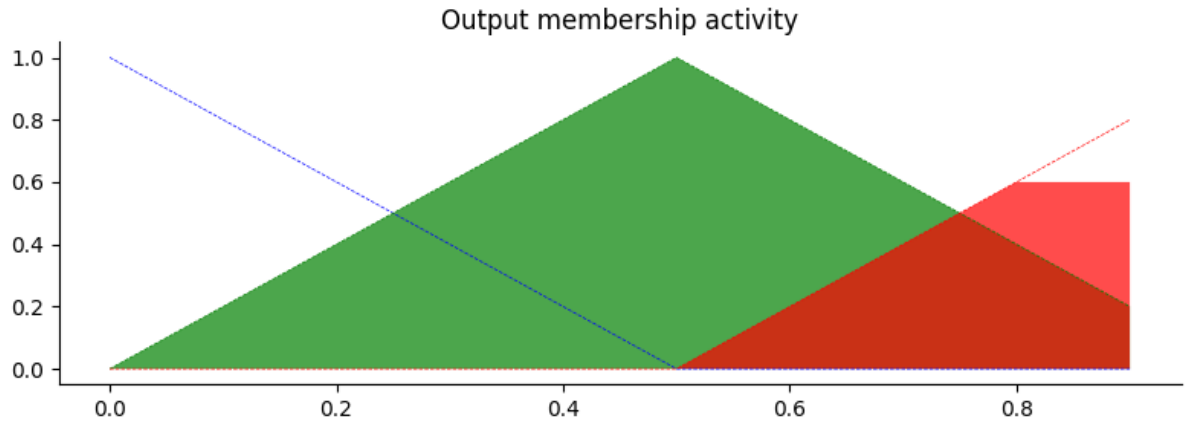

Figure 5.3: Rule application to input variables

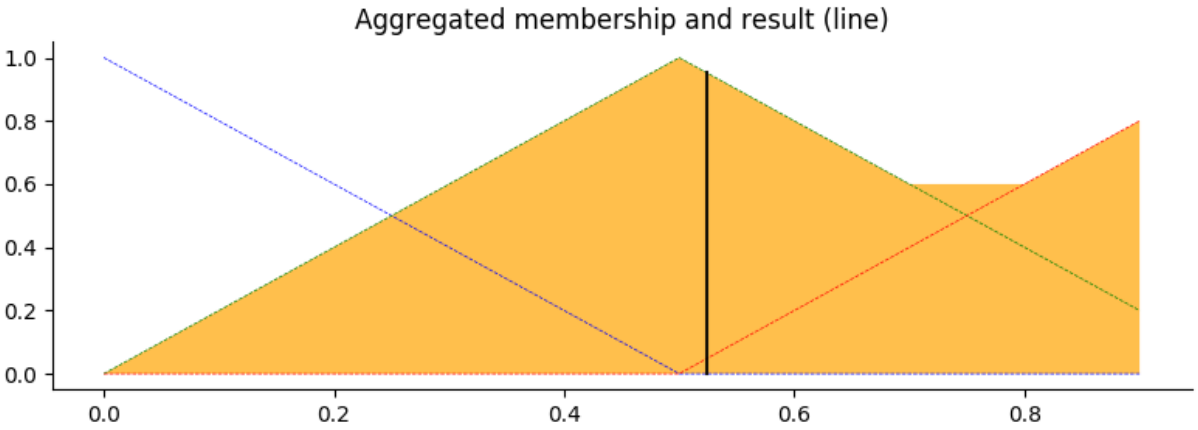

Figure 5.4: Defuzzification of output variable

From this experiment, it was found out that Robot IV reacts to its internal and external environmental conditions. Table 5.1 represents the changes of speed altering factor with respect to the environment variable changes.

Table 5.1: Strength altering factor(k) changes with Engergy Level, Temperature and Distance

Experiment	Energy Level(%)	Distance (cm)	Temperature (C^0)	Speed Altering factor
1	20	150	22	0.42
2	40	100	27	0.57
3	50	300	25	0.52
4	80	50	26	0.78
5	100	180	28	0.83

5.3 Fuzzy Integrated Subsumption Architecture Vs. Neural Dynamic based Fuzzy Integrated Subsumption Architecture

In this part, research is focused on evaluating the performance of proposed Neural Dynamic based Fuzzy Integrated Subsumption compared to the Fuzzy Integrated Subsumption Architecture which is addressed by the RQ2. For this purpose, as explained in the implementation section, path following behavior was implemented using attractor dynamic approach and improvement of the behavior is very impressive. Robot V has the Neural Dynamic based Fuzzy Integrated Subsumption Architecture and Robot IV has the Fuzzy Integrated Subsumption Architecture. Robot V was able to smoothly move through the path and performs better than Robot IV. Figure 5.5 illustrates the arena which was used to evaluate the architectures.

Figure 5.5: Arena used to evaluate Robot III, IV and V

To evaluate the performance of attractor dynamic approach, the following arena illustrated by figure 5.6 was used. In this case, only the path following behavior was considered to eliminate the interferences such as CPU overhead from the results. From this experiment, it is clear that, attractor dynamic approach gives a significant improvement over the conventional approach. Naturalizing the path following behavior improves the performance of the path following behavior and it increases the overall performance of the architecture.

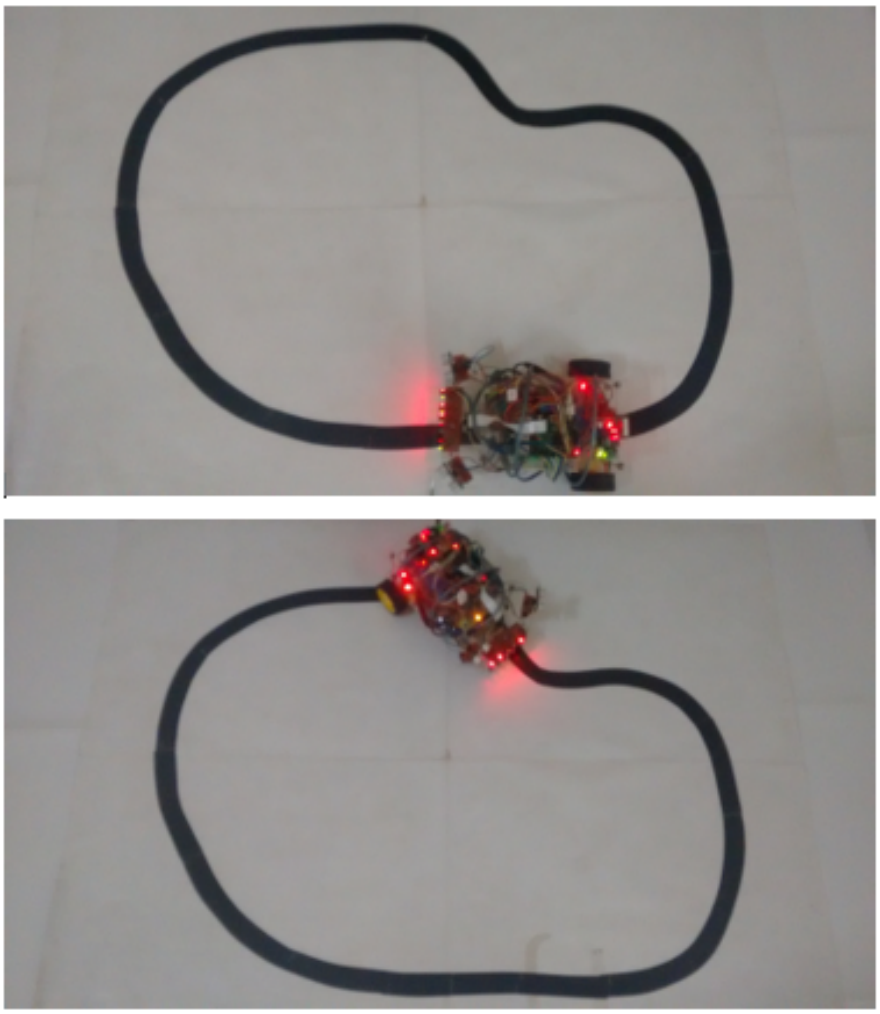

Figure 5.6: Evaluating the attractor dynamic based path following behavior

Chapter 6

Conclusion

6.1 Introduction

In this research, our approach was to interface internal and external environment parameters with the subsumption architecture using a fuzzy controller based approach. Results of the experiments conclude that fuzzy controller based approach can be used to interface the internal and external environmental parameters with the subsumption architecture. As a further step, behaviors of the subsumption architecture were implemented using attractor dynamics. There is significant improvement of the reactiveness of robot when behaviors of the robot are implemented using neural dynamics.

6.2 Conclusions about research questions

To answer the first research question RQ1, fuzzy controller based architecture was designed, implemented and evaluated in this research. From the results, it can be concluded that from fuzzy controller based approach which was used in this research increases the reactiveness of the existing reactive robotic architecture.

As addressed in the RQ2, neural dynamic approach which was explained in the Chapter 2 can be used as the biologically plausible computational approach to naturalize the reactive architecture. Results of the experiments, clearly illustrate a significant improvement of the reactive robotic architecture.

6.3 Conclusions about research problem

Interfacing internal and external environmental parameters with the decision-making process of reactive architecture was the main problem which was addressed in this research. From this research, it can be concluded that using fuzzy integration based approach, it is possible to interface internal and external environmental parameters with the decision-making process of the agent.

This research also focused on further naturalizing the reactive architecture from a biologically inspired computational tractable approach. In this research, path following behaviour of Robot V was implemented using attractor dynamic approach. There is a significant improvement in path following behaviour after implementing it from attractor dynamic approach.

6.4 Limitations

When increasing the number of layers and complexity of the behaviours, overall behaviour of the robot deviates from the reactive nature due to limitation of the threading overhead.

6.5 Future Work

Following potential avenues of investigation can be suggested as future experiments;

- The two proposed architectures were evaluated for a single scenario. These architectures should be tested in different scenarios to test the applicability of these architectures in different scenarios.
- In this experiment, only the Path Finding behaviour was implemented using neural dynamic approach. As a further step, other behaviours also can be implemented using neural dynamic approach. For an example, obstacle avoidance behaviour can be implemented using repeller dynamic approach which is the opposite of the attractor dynamic approach.
- In this study, temperature, energy level and distance were considered as internal and external environment conditions. When increasing the number of

parameters, results may deviate from the expected results. Therefore, it is important to evaluate these architectures for different scenarios.

References

[1] Rodney A. Brooks. Intelligence without representation. *Artificial Intelligence*, 47(1-3):139–159, 1991. doi: 10.1016/0004-3702(91)90053-m.

[2] William Clancey. A boy scout, toto, and a bird: How situated cognition is different from situated robotics. *In NATO workshop on Emergence, Situatedness, Subsumption and Symbol Grounding*, pages 227–236, 1995.

[3] Jessica Lindblom and Tom Ziemke. Social situatedness of natural and artificial intelligence: Vygotsky and beyond. *Adaptive Behavior*, 11(2):79–96, 2003. doi: 10.1177/10597123030112002.

[4] Tim Menzies and William J. Clancey. Editorial: the challenge of situated cognition for symbolic knowledge-based systems. *International Journal of Human-Computer Studies*, 49(6):767–769, 1998. doi: 10.1006/ijhc.1998.0026.

[5] Maja J Matarić. Situated robotics. *Encyclopedia of Cognitive Science*, 2006. doi: 10.1002/0470018860.s00074.

[6] Eric Bredo. Reconstructing educational psychology: Situated cognition and deweyian pragmatism. *Educational Psychologist*, 29(1):23–35, 1994. doi: 10.1207/s15326985ep2901_3.

[7] Jan Kazimierczak and Barbara Łysakowska. Intelligent adaptive control of a mobile robot: The automaton with an internal and external parameter approach. *Robotica*, 6(04):319, 1988. doi: 10.1017/s0263574700004689.

[8] Russell and Ja Stuart. *Artificial Intelligence: A Modern Approach*. Prentice Hall, Upper Saddle River, New Jersey, 2003.

[9] Rodney A. Brooks. *Cambrian intelligence: the early history of the new AI*. MIT Press, 1999.

[10] Lucia Foglia and Robert A. Wilson. Embodied cognition. *Wiley Interdisciplinary Reviews: Cognitive Science*, 4(3):319–325, Aug 2013. doi: 10.1002/wcs.1226.

[11] Matthew Luciw, Yulia Sandamirskaya, Sohrob Kazerounian, Jurgen Schmidhuber, and Gregor Schoner. Reinforcement and shaping in learning action sequences with neural dynamics. *4th International Conference on Development and Learning and on Epigenetic Robotics*, 2014. doi: 10.1109/devlrn.2014.6982953.

[12] Sohrob Kazerounian, Matthew Luciw, Mathis Richter, and Yulia Sandamirskaya. Autonomous reinforcement of behavioral sequences in neural dynamics. *The 2013 International Joint Conference on Neural Networks (IJCNN)*, 2013. doi: 10.1109/ijcnn.2013.6706877.

[13] Robin R. Murphy. *Introduction to AI robotics*. MIT Press, 2005.

[14] Rodney A. Brooks. Elephants dont play chess. *Robotics and Autonomous Systems*, 6(1-2):3–15, 1990. doi: 10.1016/s0921-8890(05)80025-9.

[15] Gregor Schoner and John P. Spencer. *Dynamic thinking: a primer on dynamic field theory*. Oxford University Press, 2016.

[16] Cbaumann. Subsumption architecture with robolab 2.5. URL `http://www.convict.lu/Jeunes/Subsumption.htm`.

[17] Software engineering for experimental robotics. *Springer Tracts in Advanced Robotics*, 2007. doi: 10.1007/978-3-540-68951-5.

[18] Wei Lu. Multithreading programming with java lejos. *Beginning Robotics Programming in Java with LEGO Mindstorms*, page 219–229, 2016. doi: 10.1007/978-1-4842-2005-4_13.

[19] Greg Butler, Andrea Gantchev, and Peter Grogono. Object-oriented design of the subsumption architecture. *Software: Practice and Experience*, 31(9):911–923, Feb 2001. doi: 10.1002/spe.396.

[20] Rosenblatt and Payton. A fine-grained alternative to the subsumption architecture for mobile robot control. *International Joint Conference on Neural Networks*, 1989. doi: 10.1109/ijcnn.1989.118717.

[21] Randall D. Beer. A dynamical systems perspective on agent-environment interaction. *Artificial Intelligence*, 72(1-2):173–215, 1995. doi: 10.1016/0004-3702(94)00005-l.

[22] Alberto Montebelli. Modeling the role of energy management in embodied cognition. 2012. URL `http://www.diva-portal.org/smash/get/diva2:528490/FULLTEXT01.pdf`.

[23] Alberto Montebelli, Robert Lowe, and Tom Ziemke. Toward metabolic robotics: Insights from modeling embodied cognition in a biomechatronic symbiont. *Artificial Life*, 19(3):299–315, 2013. doi: 10.1162.

[24] Tony Savage. The grounding of motivation in artificial animals: Indices of motivational behavior. *Cognitive Systems Research*, 4(1):23–55, 2003. doi: 10.1016/s1389-0417(02)00070-0.

[25] Fahmi Zal, Ting-Shuo Chen, Shou-Wei Chi, and Chung-Hsien Kuo. Fuzzy controller based subsumption behavior architecture for autonomous robotic wheelchair. *2013 International Conference on Advanced Robotics and Intelligent Systems*, 2013. doi: 10.1109/aris.2013.6573552.

[26] Gregor Schoner, Mathis Richter, and Raul Grieben. Autonomous robotics background material. 2017. URL `https://www.ini.rub.de/upload/file/1472216762_eb63d6086f7690a711d7/background_material.pdf`.

[27] Masato Ito and Jun Tani. On-line imitative interaction with a humanoid robot using a dynamic neural network model of a mirror system. *Adaptive Behavior*, 12(2):93–115, 2004. doi: 10.1177/105971230401200202.

[28] Masato Ito, Kuniaki Noda, Yukiko Hoshino, and Jun Tani. Dynamic and interactive generation of object handling behaviors by a small humanoid robot using a dynamic neural network model. *Neural Networks*, 19(3):323–337, 2006. doi: 10.1016/j.neunet.2006.02.007.

[29] Yulia Sandamirskaya, Mathis Richter, and Gregor Schoner. A neural-dynamic architecture for behavioral organization of an embodied agent. *2011 IEEE International Conference on Development and Learning (ICDL)*, 2011. doi: 10.1109/devlrn.2011.6037353.

[30] Tom Ziemke and Robert Lowe. On the role of emotion in embodied cognitive architectures: From organisms to robots. *Cognitive Computation*, 1(1): 104–117, Jun 2009. doi: 10.1007/s12559-009-9012-0.

[31] Lejos nxt api documentation. URL `https://lejos.sourceforge.io/p_technologies/nxt/nxj/api/lejos/subsumption/package-summary.html`.

Appendices

Appendix A

Code Listings

A.1 Python implementation of the fuzzy controller

```
import numpy as np
import skfuzzy as fuzz
import matplotlib.pyplot as plt

x_distance = np.arange(0, 300, 1)
x_energy = np.arange(0, 100, 1)
x_temperature=np.arange(20,30,1)
x_speed  = np.arange(0, 1, 0.1)

# Generate fuzzy membership functions for distance
distance_lo  = fuzz.trimf(x_distance, [0, 0, 150])
distance_md = fuzz.trimf(x_distance, [0, 150, 300])
distance_hi  = fuzz.trimf(x_distance, [150, 300, 300])

energy_lo = fuzz.trimf(x_energy, [0, 0, 50])
energy_md = fuzz.trimf(x_energy, [0,50, 100])
energy_hi = fuzz.trimf(x_energy, [50, 100, 100])

temperature_lo = fuzz.trimf(x_temperature, [20, 20, 25])
temperature_md = fuzz.trimf(x_temperature, [20, 25, 30])
temperature_hi = fuzz.trimf(x_temperature, [25, 30, 30])
```

```
speed_lo = fuzz.trimf(x_speed, [0, 0, 0.5])
speed_md = fuzz.trimf(x_speed, [0, 0.5, 1])
speed_hi = fuzz.trimf(x_speed, [0.5, 1, 1])

# Visualize these universes and membership functions
fig, (ax0, ax1, ax2,ax3) = plt.subplots(nrows=4, figsize=(8, 9))

ax0.plot(x_distance, distance_lo, 'b', linewidth=1.5, label='Long')
ax0.plot(x_distance, distance_md, 'g', linewidth=1.5, label='Mid')
ax0.plot(x_distance, distance_hi, 'r', linewidth=1.5, label='Short')
ax0. set_title ('Distance to Travel')
ax0.legend()

ax1.plot(x_energy, energy_lo, 'b', linewidth=1.5, label='Low')
ax1.plot(x_energy, energy_md, 'g', linewidth=1.5, label='Medium')
ax1.plot(x_energy, energy_hi, 'r', linewidth=1.5, label='High')
ax1. set_title ('Energy')
ax1.legend()

ax2.plot(x_temperature, temperature_lo, 'b', linewidth=1.5, label='Low')
ax2.plot(x_temperature, temperature_md, 'g', linewidth=1.5, label='Medium')
ax2.plot(x_temperature, temperature_hi, 'r', linewidth=1.5, label='High')
ax2. set_title ('Temperature')
ax2.legend()

ax3.plot(x_speed, speed_lo, 'b', linewidth=1.5, label='Low')
ax3.plot(x_speed, speed_md, 'g', linewidth=1.5, label='Medium')
ax3.plot(x_speed, speed_hi, 'r', linewidth=1.5, label='High')
ax3. set_title ('Speed Altering Factor')
ax3.legend()

#Rule Application

#distance
 distance_level_lo  = fuzz.interp_membership(x_distance, distance_lo, distance)
distance_level_md = fuzz.interp_membership(x_distance, distance_md, distance)
 distance_level_hi  = fuzz.interp_membership(x_distance, distance_hi, distance)

#energy
```

```
energy_level_lo  = fuzz.interp_membership(x_energy, energy_lo, energy)
energy_level_md = fuzz.interp_membership(x_energy, energy_md, energy)
energy_level_hi  = fuzz.interp_membership(x_energy, energy_hi, energy)

#temperature
temperature_level_lo  = fuzz.interp_membership(x_temperature, temperature_lo,
    temperature)
temperature_level_md = fuzz.interp_membership(x_temperature, temperature_md,
     temperature)
temperature_level_hi  = fuzz.interp_membership(x_temperature, temperature_hi,
    temperature)

#if energy level is medium and distance is high and temperature is medium
    speed medium
active_rule1  = max(distance_level_hi,energy_level_md,temperature_level_md)
speed_activation_md = np.fmin(active_rule1, speed_md)

#if energy level is medium and distance is low and temperature is medium
    speed high
active_rule2  = max(distance_level_lo,energy_level_md,temperature_level_md)
speed_activation_hi  = np.fmin(active_rule2, speed_hi)

#if energy level is medium and distance is high and temperature is medium
    speed medium
active_rule3  = max(distance_level_hi, energy_level_lo ,temperature_level_md)
speed_activation_lo  = np.fmin(active_rule3, speed_lo)

speed0 = np.zeros_like(x_speed)

# Visualize this
fig , ax0 = plt.subplots( figsize =(8, 3))

ax0.fill_between(x_speed, speed0, speed_activation_lo , facecolor='b', alpha
    =0.7)
ax0.plot(x_speed, speed_lo, 'b', linewidth=0.5, linestyle ='--', )
ax0.fill_between(x_speed, speed0, speed_activation_md, facecolor='g', alpha
    =0.7)
ax0.plot(x_speed, speed_md, 'g', linewidth=0.5, linestyle ='--')
```

```
ax0.fill_between(x_speed, speed0, speed_activation_hi, facecolor='r', alpha
    =0.7)
ax0.plot(x_speed, speed_hi, 'r', linewidth=0.5, linestyle='--')
ax0.set_title('Output membership activity')

# Turn off top/right axes
for ax in (ax0,):
    ax.spines['top'].set_visible(False)
    ax.spines['right'].set_visible(False)
    ax.get_xaxis().tick_bottom()
    ax.get_yaxis().tick_left()

plt.tight_layout()

#defuzzufication

# Aggregate all three output membership functions together
aggregated = np.fmax(speed_activation_lo,np.fmax(speed_activation_md,
    speed_activation_hi))

# Calculate defuzzified result
speed = fuzz.defuzz(x_speed, aggregated, 'centroid')
speed_activation = fuzz.interp_membership(x_speed, aggregated, speed) # for
    plot

# Visualize this
fig, ax0 = plt.subplots(figsize=(8, 3))

ax0.plot(x_speed, speed_lo, 'b', linewidth=0.5, linestyle='--', )
ax0.plot(x_speed, speed_md, 'g', linewidth=0.5, linestyle='--')
ax0.plot(x_speed, speed_hi, 'r', linewidth=0.5, linestyle='--')
ax0.fill_between(x_speed, speed0, aggregated, facecolor='Orange', alpha=0.7)
ax0.plot([speed, speed], [0, speed_activation], 'k', linewidth=1.5, alpha=0.9)
ax0.set_title('Aggregated membership and result (line)')

# Turn off top/right axes
for ax in (ax0,):
    ax.spines['top'].set_visible(False)
    ax.spines['right'].set_visible(False)
```

```
    ax.get_xaxis().tick_bottom()
    ax.get_yaxis(). tick_left ()

plt.tight_layout()

plt.show()
```

A.2 Implementation of the Obstacle Avoidance Behaviour

```
class obstacle_avoidance:
    #initial distance
    distance_m30=100
    distance_0=100
    distance_30=100
    distance_m45=100
    distance_45=100

    def suppress():
        return True
    def sonar(self):
        GPIO.output(TRIG, True)
        time.sleep(0.1)
        GPIO.output(TRIG, False)

        while GPIO.input(ECHO)==0:
                pulse_start = time.time()

        while GPIO.input(ECHO)==1:
                pulse_end = time.time()

        pulse_duration = pulse_end - pulse_start
        distance = pulse_duration * 17150
        distance = round(distance,2)

        return distance

    def readsonar(self):

            p.ChangeDutyCycle(4.5)
                    self.distance_m30=self.sonar()
                    print "Distance at -30 "+str(self.distance_m30)+"cm"
                    time.sleep(0.05) # sleep 1 second
```

```
                p.ChangeDutyCycle(2.5) # turn towards 0 degree
                self.distance_0=self.sonar()
                print "Distance at 0 "+str(self.distance_0)+"cm"
                time.sleep(0.05)  # sleep 1 second

    p.ChangeDutyCycle(6.5) # turn towards 30 degree - right
                self.distance_30=self.sonar()
                print "Distance at 30 "+str(self.distance_30)+"cm"
                time.sleep(0.05)  # sleep 1 second

def takeControl(self):
                self.readsonar()
                if(self.distance_m30<30 or self.distance_0<30 or self.
                    distance_30<30):
                        print "Obstacle Ovoidance Taking Control"
                        return True

def action(self):


                print "Obstacle avoidance processing"

    GPIO.output(in1_Lpin, False)
                GPIO.output(in2_Lpin, False)
                pwm1.ChangeDutyCycle(0)

                GPIO.output(in1_Rpin, False)
                GPIO.output(in2_Rpin, False)
                pwm2.ChangeDutyCycle(0)



    p.ChangeDutyCycle(7.5)
                self.distance_m45=self.sonar()
                print "Distance at -45 "+str(self.distance_m45)+"cm"


            if(self.distance_m45>50):
        GPIO.output(in1_Lpin, False)
```

```
        GPIO.output(in2_Lpin, True)
        pwm1.ChangeDutyCycle(50)
        GPIO.output(in1_Rpin, False)
        GPIO.output(in2_Rpin, True)
        pwm2.ChangeDutyCycle(50)
        time.sleep (0.5)

        #turn left
        GPIO.output(in1_Lpin, True)
        GPIO.output(in2_Lpin, False)
        pwm1.ChangeDutyCycle(50)

        GPIO.output(in1_Rpin, False)
        GPIO.output(in2_Rpin, True)
        pwm2.ChangeDutyCycle(50)

        time.sleep (0.5)

        GPIO.output(in1_Lpin, False)
                        GPIO.output(in2_Lpin, False)
                        pwm1.ChangeDutyCycle(0)

                        GPIO.output(in1_Rpin, False)
                        GPIO.output(in2_Rpin, False)
                        pwm2.ChangeDutyCycle(0)



else:
    p.ChangeDutyCycle(12.5) # turn towards 45 degree - right
                    self.distance_45=self.sonar()
                    print "Distance at 45 "+str(self.distance_45)+"cm"

    if( self.distance_45>50):
        GPIO.output(in1_Lpin, False)
                        GPIO.output(in2_Lpin, True)
                    pwm1.ChangeDutyCycle(50)

                    GPIO.output(in1_Rpin, False)
```

```
                GPIO.output(in2_Rpin, True)
                pwm2.ChangeDutyCycle(50)

    time.sleep(0.5)


    #turn right
    GPIO.output(in1_Lpin, False)
    GPIO.output(in2_Lpin, True)
    pwm1.ChangeDutyCycle(50)

    GPIO.output(in1_Rpin, True)
    GPIO.output(in2_Rpin, False)
    pwm2.ChangeDutyCycle(50)

    time.sleep(0.5)

    GPIO.output(in1_Lpin, False)
                        GPIO.output(in2_Lpin, False)
    pwm1.ChangeDutyCycle(0)

                        GPIO.output(in1_Rpin, False)
                        GPIO.output(in2_Rpin, False)
    pwm2.ChangeDutyCycle(0)



else:

    GPIO.output(in1_Lpin, False)
                        GPIO.output(in2_Lpin, True)
                        pwm1.ChangeDutyCycle(50)

                        GPIO.output(in1_Rpin, False)
                        GPIO.output(in2_Rpin, True)
                        pwm2.ChangeDutyCycle(50)

                        time.sleep(0.5)
```

```
#turn backward
GPIO.output(in1_Lpin, True)
GPIO.output(in2_Lpin, False)
pwm1.ChangeDutyCycle(50)

GPIO.output(in1_Rpin, False)
GPIO.output(in2_Rpin, True)
pwm2.ChangeDutyCycle(50)

time.sleep(0.5)

GPIO.output(in1_Lpin, False)
                    GPIO.output(in2_Lpin, False)
                    pwm1.ChangeDutyCycle(0)

                    GPIO.output(in1_Rpin, False)
                    GPIO.output(in2_Rpin, False)
                    pwm2.ChangeDutyCycle(0)
```

return True

A.3 Path Following behavior using sine attractor dynamics

```
if (s1==0)and(s2==0)and(s3==0)and(s4==0)and(s5==0):
    pass #line finding
elif (s1==0)and(s2==0)and(s3==1)and(s4==0)and(s5==0):
    angle_difference =0
elif (s1==0)and(s2==1)and(s3==1)and(s4==1)and(s5==0):
    angle_difference =0
elif (s1==0)and(s2==1)and(s3==1)and(s4==0)and(s5==0):
    angle_difference =−10
elif (s1==0)and(s2==0)and(s3==1)and(s4==1)and(s5==0):
    angle_difference =10
elif (s1==0)and(s2==0)and(s3==0)and(s4==1)and(s5==1):
    angle_difference =30
elif (s1==0)and(s2==0)and(s3==0)and(s4==1)and(s5==0):
    angle_difference =30
elif (s1==0)and(s2==0)and(s3==0)and(s4==0)and(s5==1):
    angle_difference =30
elif (s1==0)and(s2==0)and(s3==1)and(s4==1)and(s5==1):
    angle_difference =30
elif (s1==1)and(s2==0)and(s3==0)and(s4==0)and(s5==0):
    angle_difference =−30
elif (s1==0)and(s2==1)and(s3==0)and(s4==0)and(s5==0):
    angle_difference =−30
elif (s1==1)and(s2==1)and(s3==0)and(s4==0)and(s5==0):
    angle_difference =−30
elif (s1==1)and(s2==1)and(s3==1)and(s4==0)and(s5==0):
    angle_difference =−30

pi=math.pi
d_phi = lamda ∗ (−math.sin(pi∗(angle_difference)/180))
v_mm_per_second = d_phi/math.pi ∗ 53
v_pulses_per_second = v_mm_per_second / 0.13
drive.SetSpeed(−v_pulses_per_second + speed, v_pulses_per_second + speed)
```

A.4 Motor driver implementation

```
import RPi.GPIO as GPIO
from time import sleep
GPIO.setmode(GPIO.BOARD)

#Left Motor
Enable_Lpin =38
In1_Lpin = 33
In2_Lpin =31

#Right Motor
Enable_Rpin =40
In1_Rpin = 37
In2_Rpin =35

#Setting Left Motor
GPIO.setup(Enable_Lpin, GPIO.OUT)
GPIO.setup(In1_Lpin, GPIO.OUT)
GPIO.setup(In2_Lpin, GPIO.OUT)
pwmL = GPIO.PWM(Enable_Lpin,500)
pwmL.start(0)

#Setting Right Motor
GPIO.setup(Enable_Rpin, GPIO.OUT)
GPIO.setup(In1_Rpin, GPIO.OUT)
GPIO.setup(In2_Rpin, GPIO.OUT)
pwmR = GPIO.PWM(Enable_Rpin,500)
pwmR.start(0)


def SetSpeed(SpeedL,SpeedR):
    SpeedR=SpeedR-8

    if SpeedR>=0 and SpeedL>=0:
        GPIO.output(In1_Lpin, True)
        GPIO.output(In2_Lpin, False)
        pwmL.ChangeDutyCycle(SpeedL)
```

```
        GPIO.output(In1_Rpin, True)
        GPIO.output(In2_Rpin, False)
        pwmR.ChangeDutyCycle(SpeedR)


    elif SpeedR>=0 and SpeedL<=0:
        GPIO.output(In1_Lpin, False)
        GPIO.output(In2_Lpin, True)
        pwmL.ChangeDutyCycle(-SpeedL)

        GPIO.output(In1_Rpin, True)
        GPIO.output(In2_Rpin, False)
        pwmR.ChangeDutyCycle(SpeedR)

    elif SpeedR<=0 and SpeedL<=0:

        GPIO.output(In1_Lpin, False)
        GPIO.output(In2_Lpin, True)
        pwmL.ChangeDutyCycle(-SpeedL)

        GPIO.output(In1_Rpin, False)
        GPIO.output(In2_Rpin, True)
        pwmR.ChangeDutyCycle(-SpeedR)

    elif SpeedR<=0 and SpeedL>=0:
        GPIO.output(In1_Lpin, True)
        GPIO.output(In2_Lpin, False)
        pwmL.ChangeDutyCycle(SpeedL)

        GPIO.output(In1_Rpin, False)
        GPIO.output(In2_Rpin, True)
        pwmR.ChangeDutyCycle(-SpeedR)

def stop():
        GPIO.output(In1_Lpin, False)
        GPIO.output(In2_Lpin, False)
        pwmL.ChangeDutyCycle(0)

        GPIO.output(In1_Rpin, False)
```

```
        GPIO.output(In2_Rpin, False)
        pwmR.ChangeDutyCycle(0)

def gpioCleanUP():
    GPIO.cleanup()
```

A.5 Arduino Code

```
const int analogIn0 = A0;
const int analogIn1 = A1;

int mVperAmp = 185; // use 100 for 20A Module and 66 for 30A Module
double RawValue= 0;
double RawValue2= 0;
float ACSoffset = 2500;
double Voltage = 0;
double Voltage2 = 0;
double Amps = 0;
int count=0;

double BatteryVoltage=0;
double Energy=2000;



void setup(){
 Serial.begin(9600);
}

void loop(){
 RawValue=0;
 RawValue2=0;

 for (int i=0;i<1000;i++){
  RawValue +=analogRead(analogIn1);
  RawValue2 +=analogRead(analogIn0);

  delay(1);
 }

 RawValue/=1000;
 RawValue2/=1000;

 Voltage = (RawValue / 1024.0) * 5000;
 Voltage2 = (RawValue / 1024.0) * 5000*4;
```

```
Amps = ((Voltage - ACSoffset) / mVperAmp);

if(count==2){
 ACSoffset=Voltage;
}

if(count>2 && -0.05<Amps && Amps<0.05){
    ACSoffset=Voltage;
}


Serial.print("Amps = ");
Serial.print(Amps,3);
Serial.print("\t Voltage = ");
Serial.println(Voltage2,3); //

count++;

}
```