SELF RECONFIGURABLE MODULAR ROBOT

BY

CHANG FANXI, FRANCIS

DEPARTMENT OF ELECTRICAL AND COMPUTER ENGINEERING

A THESIS SUBMITTED FOR THE DEGREE OF MASTER IN ENGINEERING

NATIONAL UNIVERSITY OF SINGAPORE

2013

DECLARATION

I hereby declare that the thesis is my original work and it has been written by me in its entirety. I have duly acknowledged all the sources of information which has been used in the thesis.

This thesis has also not been submitted for any degree in any university previously.

Chang Fanxi, Francis

2 December 2013

ABSTRACT

The "Evolve" system developed in 2008 is the first National University of Singapore (NUS) modular robots. While conventional robots have fixed morphology, a modular robot system is capable of having variable morphology. In the family of modular robotics, Evolve is classified as a configurable robotics system. Evolve lacks autonomous reconfigurability, hence it is unable to change its shape to suit the environment and task.

The primary aim of this work is to develop a Self-reconfigurable modular robot. The second generation Evolve system will be named Evolve Generation 2, it is equipped with a controllable docking surface that allows autonomous reconfiguration. Subsequently, the basic locomotion generation is performed using genetic algorithm.

The thesis focuses on the design and construction of Evolve hardware, analysis and implementation of different sub-systems that are required for self-reconfigurability. Genetic algorithm is applied to generate basic locomotion to any given module configuration without the need for deterministic locomotion planning.

ACKNOWLEDGMENTS

Firstly, I would like to express my appreciation to project supervisor, Dr Prahlad Vadakkepat, for his guidance and advices to the project. With his supportive ideas, have made this a fulfilling and enriching research.

Next, I would like to Dr. Ng Buck Sin, Mr. Yong See Wei, as well as the staffs in the Mechatronics and Automation Lab who have offered their guidance and support over the past year to build this experience a fruitful and enjoyable one. Throughout the duration of this project, my knowledge in robotics field is greatly improved.

I am also grateful towards my fellow research students in the Center for Intelligent Control, National University of Singapore.

Table of Contents

Table of Figures

List of Tables

Chapter 1 Introduction

1.1 Background and Motivation

Self-reconfigurable modular robots are autonomous robotic system having variable morphology. Conventional robotic systems are task orientated with fixed morphology, the system is usually optimised for a fixed morphology with little flexibility to change in adapting to variation in the environment. Self-reconfigurable modular robots on the other hand can deliberately change their shape by altering the connectivity of individual modules, in order to adapt to new circumstances, perform new tasks, or recover from damage. These modular systems are potentially more robust and adaptive than conventional systems. For economic consideration, replicating similar robotics systems can potentially lower overall cost, making a range of complex machines out of single type of mass-produced module.

A conventional differential wheel robot may be efficient in moving on even terrains but is significantly less efficient in manoeuvring over rough terrains. A self-reconfigurable modular robot can change its shape to an arbitrary wheel or ball structure to spin itself quickly over fairly flat terrains, and reassemble into a worm-like or legged configuration to cross uneven terrains. **Figure 1** shows an example of self-reconfigurable modular system assuming a snake-like shape to move through a narrow tunnel and reconfiguring to a block structure to climb steps.

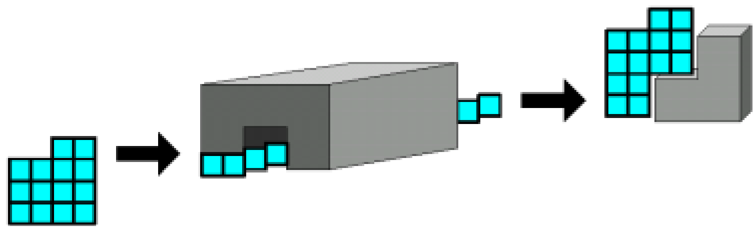


Figure 1 : Self-reconfigurable modular system over different environment.

The term self-reconfigurable indicates that the system has the capability or means of controlling the overall structural form of the system. Self-reconfigurable robotic systems are commonly found with mechanisms that allow attachment and detachment of robots autonomously. As for the term modular, it requires similar or duplicate modules that can be added or removed from the system. The key objective is to allow a finite number of modules connected in an organised structure, and the organized structure then performs tasks cooperatively.

Compared to fixed morphology robots, self-reconfigurable modular robots are particularly well-suited for operational conditions and where the requirements are not known or not well defined prior to system deployment. Hence, applications can be found in areas as space exploration and search and rescue.

Space exploration [1] presents numerous challenges, including drastic changes in the operating environment and significant limitations on the mass and volume of the equipment. If a single module can be reconfigured to perform many tasks, then such a system can tackle both unexpected challenges while occupying lesser space and weight as compared to various fixed morphology robots. Failure is catastrophic for space robots working as there is no repair capability and malfunction can potentially lead to mission failure. The redundant nature of self-reconfigurable modular robots gives the system the ability to discard or repair failed modules. This is an advantage over fixed morphology robots and, it also reduces the risk of complete mission failure.

Search and Rescue in collapsed buildings or unpredictable environments are examples where the benefits of the use of self-reconfigurable modular robots are clear. Reconfigurable robots can take a different form to tackle changes in terrains when in search for victims. Upon finding a victim, a module of the system can be dropped off to monitor the health status of the found victim while the remaining modules can continue the search and rescue task for other victims.

Since the late 1980s, research work on Dynamically Reconfigurable Robotics System (DRRS) had begun. In Toshio Fukuda and Seiya Nakagawa [2] paper, they explained robots that can be self-reorganize its total shape and its software to a given task. The robotics system has a level of the flexibility and adaptability to a change of task is much higher than that of the conventional concept.

There are many different research topics covering self-reconfigurable modular robots. In the early works, research focused on the shape forming and configuration of modular systems. The recent works more intensified researches in robot mechanism design, dynamic locomotion generation [3] and interconnection of modules.

Figure 2: First generation Evolve modules.

At National University of Singapore, the first generation of configurable modular robotics named “Evolve” (see **Figure 2**) was developed in 2008.

1.2 First Generation Evolve Configurable Modular System

A total of four Evolve Generation 1 modules was designed and built in 2008. The modules are based on a hybrid lattice and chain structure, with six dock-able faces that allowed for interconnects with other modules. The six dock-able faces are made possible with 32 neodymium disc magnets, three North poles and three South poles dock-able faces. The magnetic polarity enables dock bias and alignment during connection. Each module has a built-in computational unit, actuators and battery. The dimensions of a single module measures 90mm x 90mm x 180mm as shown in **Figure 3.**

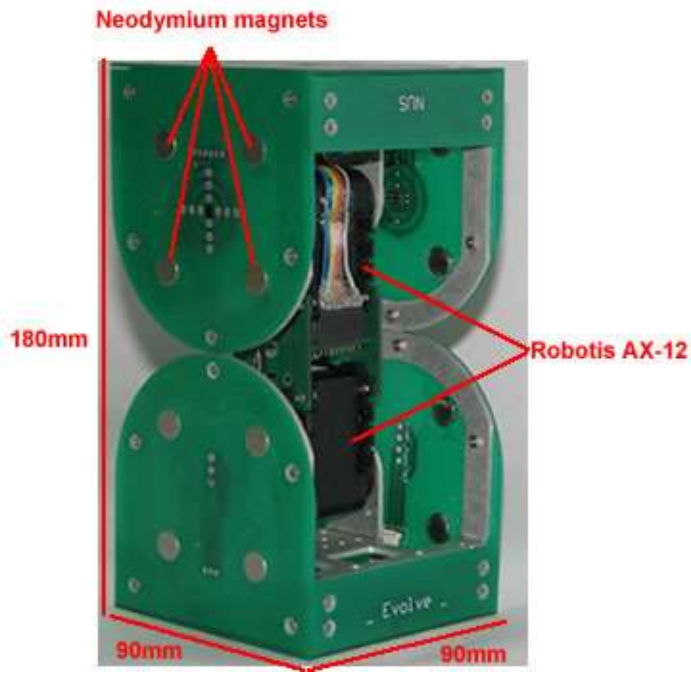


Figure 3 : Single Evolve module.

Locomotion of a module is made possible by two Robotis AX-12 actuators with 180 degrees of freedom and 12kg/cm of torque. **Figure 4** shows an example of maximum rotation angle on each actuator in opposite directions.

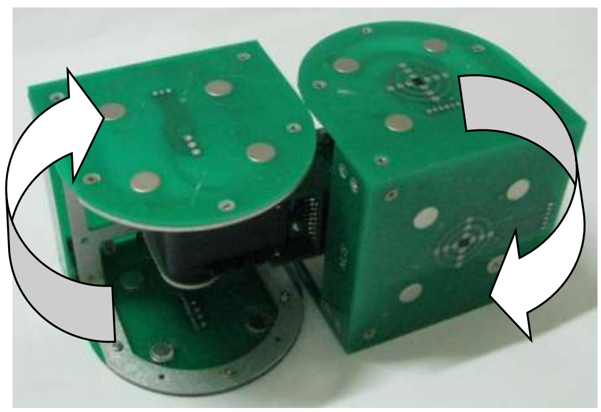


Figure 4 : Two degrees of freedom for each module.

The electronics of a module comprises of a computational unit, sensors, and battery. The computational unit is from Microchip (PIC18F6722), an 8-bits 40 MHz micro-controller. 3-

axis accelerometer and three short range proximity sensors give feedback on the module orientation. The power unit that operates each module is a 2-cell 7.4V nominal lithium polymer rechargeable battery. Wireless communicates with different modules and host computer are made possible with XBee 802.15.4 RF communication module located at the heart of the main electronics board.

The modules lack computational power, making real-time locomotion calculations impossible. Due to the use of permanent magnets, the docking surface does not allow self-reconfiguration of the modules. All attachment and detachment of modules must be done manually.

1.3 Research Objective

In this research, the primary goal is to evaluate the performance of the existing Evolve System and to develop second generation Evolve modular robots that are capable of self-reconfiguration. The fresh new design retains the advantages of the Evolve Generation 1 but significantly improves each module capability and its locomotion generation. To attain self-reconfigurable capability, the hardware of the Evolve System needs to be built up from scratch and using artificial intelligence to search for possible locomotion. The new design includes the following:

1. Structure and degrees of freedom.
2. Docking surface.
3. Electronic and controls.

4. Software and communication protocol.
5. Locomotion generation through computational intelligence (Genetic Algorithm based).

1.4 Outline

In this work, developments of Evolve Generation 2 are divided into seven chapters. In Chapter 2, a literature review is conducted to identify potential design of modular robotic developed by other research centres. Chapter 3 discusses the mechanism that allow interconnects and actuators that drive the robot. Chapter 4 introduces improvement made in the electronic systems. In Chapter 5, a general overview of the protocol used for communication and control of a module is provided. In Chapter 6, Genetic Algorithm is utilised to generate basic locomotion without the need for hard coding or motion tuning. In last chapter 7, the conclusions and finding for developments of Evolve Generation 2 are discussed.

Chapter 2 Literature Review

2.1 Overview

Frictional stories such as Power Ranger and Transformers showcase modular robots that can be combined to achieve increased capability. Self-reconfigurable modular robots systems are real world example that composed of a large number of repeated modules which can reassemble to form various intended shapes [4]. The idea of the modular system can be back traced several decades ago. This concept can be found on most automatic tool changing computer-controlled machines since 1970's. On the other hand, LEGO toys have showcased a single 4x4 block which can have infinite possibilities of configuration bounded only by imagination. The concept of interchangeability and assembly requires a common connection mechanism to other entities. A LEGO brick shown in **Figure 5** has studs (positive connect) on top and tubes (negative connect) at the bottom. The stud-and-tube coupling system uses an interference fit[1] for attachment.

Self-reconfigurable modular robots have the ability to change shape to increase their capability for a task, and they can take a form in the shape of an arm to pick up equipment or reassemble to a legged robot to overcome uneven terrains. These types of robots are normally made with three benefits, namely robustness, versatility and low cost.

[1] Interference fit - A firm, friction-based connection between two parts without the use of an additional fastener.

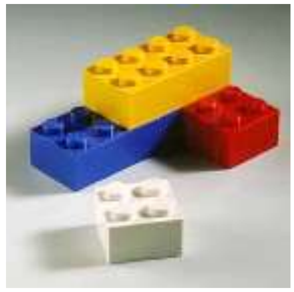


Figure 5 : Basic LEGO toy building blocks.

Robustness. A reconfigurable system comprises of many identical or similar modules which can be rearranged. Faulty parts can be discarded and replaced by identical modules in the system. This capability is also commonly known as self-repair.

Versatility. The reconfigurable ability allows modules to assemble, disassemble and reassemble between the operations. It is well suited for a variety of known tasks or tasks which do not have clear requirements at point of deployment.

Cost Efficient. Mass production plays a major role in keeping cost of production down. The system works with several identical modules, and therefore it is economical for large scale production. The versatile nature of such systems, also reduce the need to produce multiple machines for doing different tasks.

2.2 Classification of Self-reconfigurable Modular Robots

Self-reconfigurable modular robots can be identified into two classifications.

(Type 1) Based on the location of attachment or connection points, namely Lattice, Chain and Mobile.

(Type 2) The other organised by the method of movement between the modules, such as stochastic and deterministic.

2.2.1 Type 1: Lattice

The lattice based modular system [5] [6] is arranged in a grid structure. In this structure, there is a discrete position that a module can occupy. Mechanical kinematics are restricted within the grids, hence, the movement of modules and collision detection are minimal. The nature grid structure simplifies the reconfiguration process. An example of a lattice based modular system is depicted in **Figure 6**.

Figure 6 : Nine Crystal robot modules developed in the Dartmouth Robotics Lab.

2.2.2 Type 1: Chain

The chain class modular system [7] has more than one degree of freedom build into the modules. In comparison with lattice structure, chain based systems commonly allow performance of more complex kinematics on a single module. Chain modules are typically arranged in an arbitrary point in space. With a complex kinematics, the computational requirement for reconfiguration increases significantly with the number of modules within a system. An example of a chain based modular system [8] is depicted in **Figure 7**.

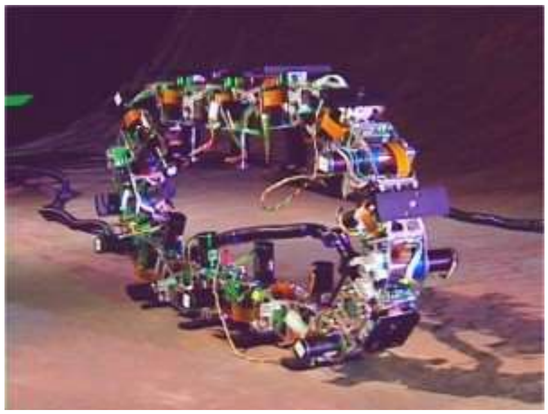


Figure 7 : PolyBot developed by Palo Alto Research Centre.

2.2.3 Type 1: Mobile

The mobile based modular system has all the required kinematics built on a single module, they would normally include multiple wheeled or legged configuration robots. A single module can function and operate within the environment without the need for assembly. Upon challenged by a different task, the mobile module can attach with other modules to form chains or lattice structure. With the increased complexity, the cost of producing these mobile modules is much higher as compared to the lattice or chain design. An example of a mobile based modular system [9] is shown in **Figure 8**.

Figure 8 : Swarmanoids developed by University of Libre de Bruxelles.

2.2.4 Type 2: Stochastic

Self-reconfiguration of the modular system can be stochastic or deterministic. In a stochastic approach, reaching a desired configuration is probabilistic. External forces manoeuvre the passive modules in an environment. For this class, bonding of modules only required simple mechanical actuation or electro-magnet. An example of 2D stochastic modular system [10] is shown in **Figure 9**.

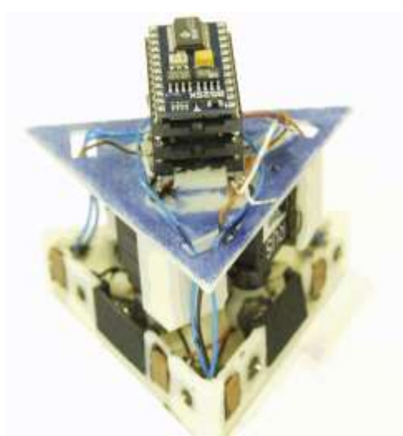


Figure 9 : 2D stochastic modular system developed by Cornell University.

2.2.5 Type 2: Deterministic

In a deterministic modular system [11], modules have the ability to manipulate movement from one position to another. This deterministic method is applied in the lattice and chain structures. In this setup, positions of modules are known at all times. The time taken for reconfiguration can be computed based on the current coordinates of each module multiplied by the number of sequences required to reposition. In general, each reconfiguration can be optimized with known coordinates of each module. For large scale modular systems, deterministic type is favoured as the probabilistic type is not as efficient or guaranteed. With the increased degrees of freedom, the time taken for reconfiguration can be reduced for the deterministic type.

2.2.6 Application of Modular Robot

The fixed morphology system can be optimised for a known task, but flexibility may compromise efficiency and performance. National Aeronautics and Space Administration (NASA) had done several researches on the possibility of sending Self-reconfigurable Modular Robots for a space mission. Unfortunately, the complexity and reliability of current modular systems are not comparable to fixed morphology robots in space mission. Currently, deployments of Self-reconfigurable Modular Robots are only limited to research centres and simulators.

2.2.7 Existing Modular Robotics Systems

There are several reported works for modular robotic systems in the last three decades [12]. **Table 1** shows some of the modular robotic systems under experimentation and development.

No.	Name	Class	DOF	Developer	Year
1	CEBOT	Mobile	Various attachments	Nagoya	1988
2	Polypod	Chain 3D	2	Stanford	1993
3	Fracta	Lattice 3D	3	MEL	1994
4	Molecule	Lattice 3D	4	Dartmouth	1998
5	CONRO	Chain 3D	2	USC/ISI	1998
6	PolyBot	Chain 3D	1	PARC	1998
7	MTRAN II	Hybrid 3D	2	AIST	2002
8	Stochastic 2D	Stochastic 2D	0	Cornell	2004
9	SuperBot	Hybrid 3D	3	USC/ISI	2005
10	Stochastic 3D	Stochastic 3D	0	Cornell	2005
11	Miche	Lattice	0	MIT	2006
12	Evolve	Hybrid 3D	2	NUS	2008
13	Milli-Motein	Chain 3D	1	MIT	2012
14	M-Block	Lattice 3D	Inertia system	MIT	2013

Table 1 : List of Modular Robotics Systems.

2.3 PolyBot

Xerox Palo Alto Research Centre (PARC) had been researching on modular robot PolyBot since 2000, a total of three generations of PolyBot had been developed. PolyBot is classified as chain 3D configuration with one degree of freedom on each module. The first generation PolyBot G1 [13] is constructed from hobbyists' servos with limited torque and computation. The second generation PolyBot G2 [14] is equipped with shape memory alloy for automatic reconfiguration. The latest PolyBot G3 [15] is designed and built with power saving modules such as integrated active brake. PolyBot research is for use in space exploration, military operations, underground mining and surface mobility.

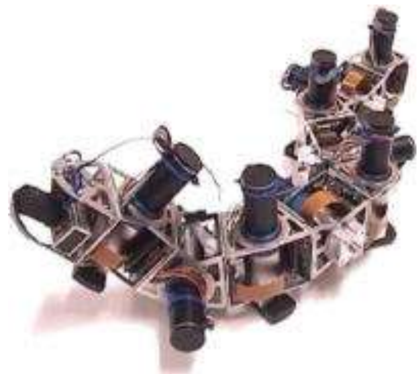


Figure 10 : Polybot G2 from Xerox Palo Alto Research Centre.

2.4 SuperBot

The development of SuperBot [16] [17] started in 2004 by University of South California. SuperBot is a hybrid 3D configuration, each module is equipped with three degrees of freedom. The module consists of two cubes connected by a hinge, with on-board power supply, actuators, sensors, a micro-controller, and six connecting faces to attach to other modules. **Figure 11** shows three SuperBot connected in series.

Figure 11 : SuperBot from University of South California.

2.5 Locomotion Pattern and Planning

One of the most basic robotics functions is locomotion pattern and planning. In general, there are two types of locomotion in modular system; one type realized as a series of self-reconfiguration, e.g. individually sending a module from the tail of the module structure to the head [18]. The other is realized as an entire shape motion such as crawling or sidewinding (snake-like), which is achieved by controlling joint motors coordination without any configuration change [19]. Designing a control method for modular systems is more difficult than ordinary robots [20] [21] [22] [23] [24]. This is because the modular configuration includes many degrees of freedom and there is wide variety of possible configurations. Locomotion can be simplified to two parameters namely, position and time. These variables can be simplified as the form of velocity.

Researchers from Advanced Industrial Science and Technology (AIST) [25] had describe an Automatic Locomotion Generation method (ALPG) for modular robots which seeks locomotion pattern for an arbitrary module configuration using a neural oscillator as a Central Pattern Generator (CPC) model and Genetic Algorithm (GA) to optimize the parameters for locomotion. In the work, they had applied a neural oscillator network to generate stable locomotion patterns for modular robotic systems. Use of the genetic algorithm enabled them to automatically create a stable locomotion pattern suited for a given module configuration and its entire shape.

Chapter 3 Evolve Modular System

3.1 Design Principles

In the design of a modular system, it is important to determine the level of complexity of a module. Homogeneous systems have uniform composition or characteristics, while heterogeneous systems have non-uniform qualities. Homogeneity and heterogeneity systems have different benefits in the conception of a modular system. A homogeneous element can be easily replaced and the system is scalable. Most swarm robotics use homogeneous configuration. The complexity in system design increases with more sub-systems added. This creates a bottleneck for weight, power and locomotion optimisation in a modular system. On the other hand, heterogeneous design brings in different qualities to each module. Such systems are designed for specific tasks and have limited flexibility for adaptation but gaining advantage on production cost and reducing weight of the modules. For self-reconfigurable robotic system, the challenge is to decide on whether a homogeneous or a heterogeneous design is most suitable. The design principle for Evolve harnesses both the advantages of homogeneous and heterogeneous modular systems.

The mechanical setup for the Evolve modular system is homogeneous with all docking surfaces having a conformed shape and size. They have identical overall dimensions and docking profile. Critical electronic components (e.g. power regulator and communication device) are common to the modules contributing to homogeneity while selected characteristics and additional sensors (e.g. accelerometer, compass) contribute to heterogeneity. The Evolve Generation 2 (Evolve 2) robot is compact, lightweight and has 3 degrees of freedom. The modules are able to collaborate and exchange information; through the on-board sensors, computational unit and communication device.

Compact Volume. Evolve 2 is designed within a 90mm cube, similar to Evolve 1.

Light Weight. To minimise the loading on the actuators and chassis, Evolve 2 is made of lightweight material with good structural strength. Acrylonitrile Butadiene Styrene (ABS) is used for constructing the chassis of modules.

Degree of Freedom. Two degrees of freedom in Evolve 1 lead to limitations in locomotion. Evolve 2 is added with a roll actuator to facilitate better locomotion.

Sensor. Inertial measuring sensors are used to detect the orientation and position of each module. Proximity sensors on the docking surfaces facilitate docking sequences between modules. In additional to sensors that detect external inputs, each module has encoders, voltage and temperature sensors to detect the internal working of actuators and power management.

Computational Unit. To achieve real-time control, the computational unit needs the capability to process the inputs and control the actuators.

Communication Device. Collaboration among the modules requires information exchange. The communication module in Evolve 2 enables direct information transfer between the two generations of modules.

3.2 Optimisation between Cost and Performance

Evolve Generation 2 system is design to be a self-reconfigurable modular system. To provide backward comparability, both generations have certain design similarity. This is to reduce the design complexity and optimise the use of project fund. Evolve2 is classified as a hybrid chain 3D design.

3.3 Mechanical Designing Tool

The design and modelling of the Evolve robot is done using Solidworks. Solidworks is a 3D mechanical computer aided design program. The software allows designing of modules through a virtual world. The design software allows modelling of components such as actuators and mechanical linkages for docking surface. With these components built in a virtual world, simple mechanical stress analysis can also be studied. The simulated results are utilised for selecting actuators and the constraints related to sub-elements. Other features of the software allow sub-system analysis such as collision, weight and calculation of model centre of gravity (to be used in simulator).

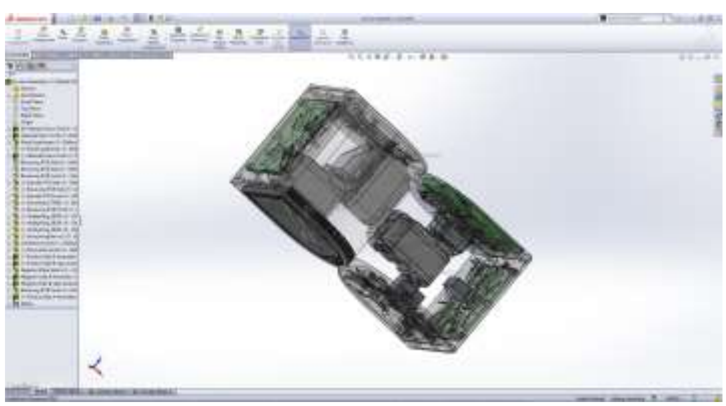


Figure 12 : Solidworks modelling software.

3.4 Mechanical Design

Evolve Generation 2 has a homogeneous mechanical structure, which composes of three actuators. Two semi-cylindrical blocks are powered by Herkulex DRS-0201 actuators. Each actuator has 180 degrees of freedom. A less powerful Herkulex DRS-0101 is located at the centre of the module provides roll translation between the semi-cylindrical blocks.

The dimension of a single module is 180mm x 90mm x 90mm, which is common between both generations. Each module is equipped with six docking surfaces, three passive and three active docks. The active docks are located at the white end of the semi-cylindrical block while the passive docks in black on the other end. Similar to the first generation there are four possible docking orientations. The details of docking mechanism are explained in later sections. **Figure 13** shows the two generations of Evolve modular systems and the Evolve Generation 2 is shown on the right.

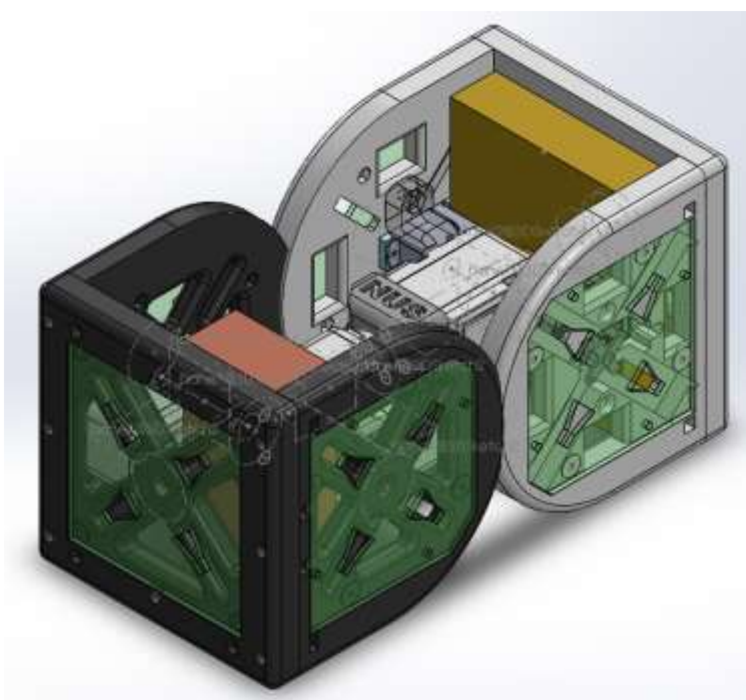


Figure 13 : 3D model of a single module.

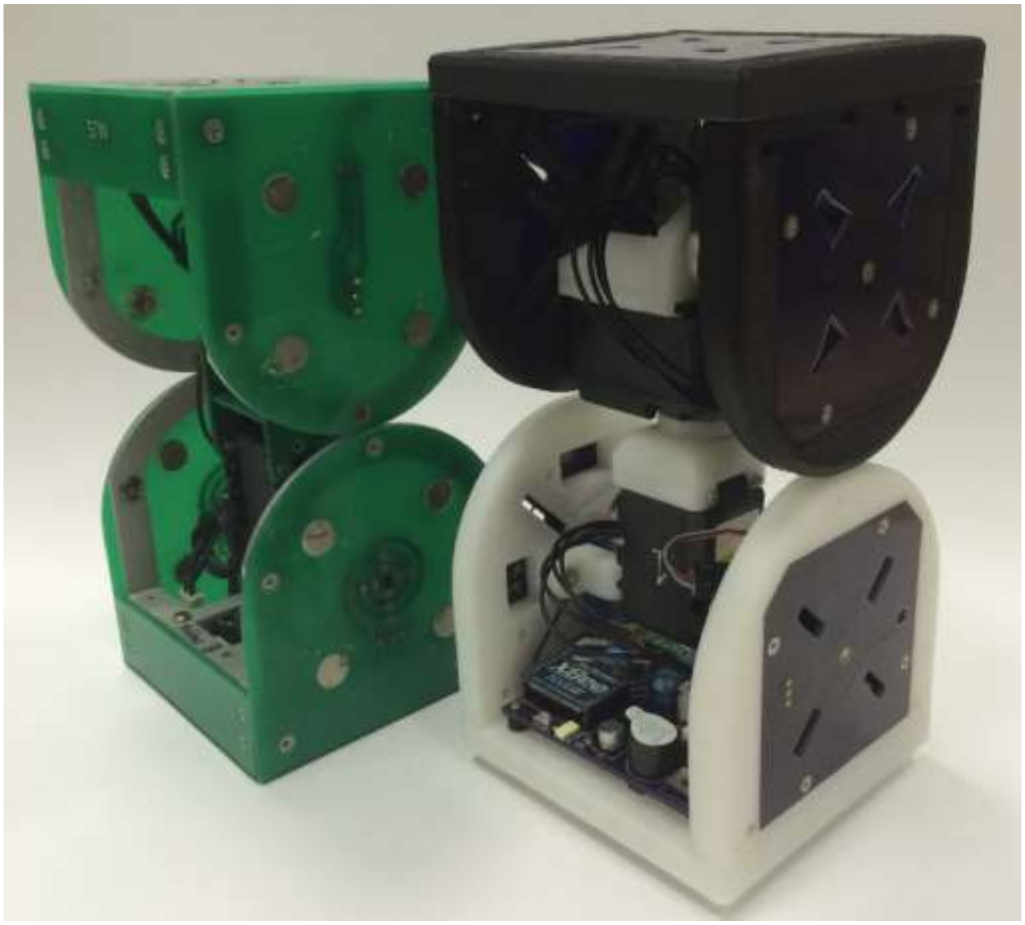


Figure 14 : Evolve Generation 1 and 2 module.

3.5 Docking Design

A docking surface is the weakest mechanical link between two modules. The non-permanent autonomous mechanical interconnect allows the modular system to self-reconfigure. The docking mechanism needs to be strong enough to hold the module in position while not increasing the weight of a module. A genderless docking surface is preferred for a modular system. Evolve Generation 1 had a magnetic dock with polarity, where four rare-earth magnets on each docking surface provides 39N holding force. Unfortunately, the resulted breakaway force is much greater than the Evolve 1 actuators can achieve. The lack of torque at the actuators resulted Evolve1 not able to disassemble from a connected dock. Although a magnetic dock can make a simple and compact design, the magnetic force is permanent and could not be controlled. As a consequence, two separated modules in an Evolve 1 cannot operate in close proximity as undesired docking may occur. For Evolve Generation 2, autonomous docking is made possible by extruding four mechanical securing hooks. This eliminated the problem of undesired dockings. The simple permanent magnetic solution in Evolve 1 is replaced by a mechanical cam and hook design in Evolve 2.

3.6 Docking Surface

There are two types of docking surfaces on Evolve 2, namely active and passive. The active surface shown on the left in **Figure 15** is located on the c, whereas the passive surface is located on the black semi-cylindrical block. Four connection hooks are located beneath the active surface; which can be extruded out of the surface during docking. In addition, each active surface has infra-red proximity sensor for detecting another docking surface. The same infra-red proximity sensor can also be used to ensure successful docking between two modules. The passive surface is shown on the right in **Figure 15**, and it has a different slot profile as compared to the active surface. The chamfered edges allow less accurately aligned hook deployment to establish successful connection. This docking surface design is able to perform a successful dock if the misalignment is less than 2mm from the ideal center. A small rare-earth magnet (silver disc shape) is added in the center of the surface to provide coarse alignment before actual deployment of securing hooks.

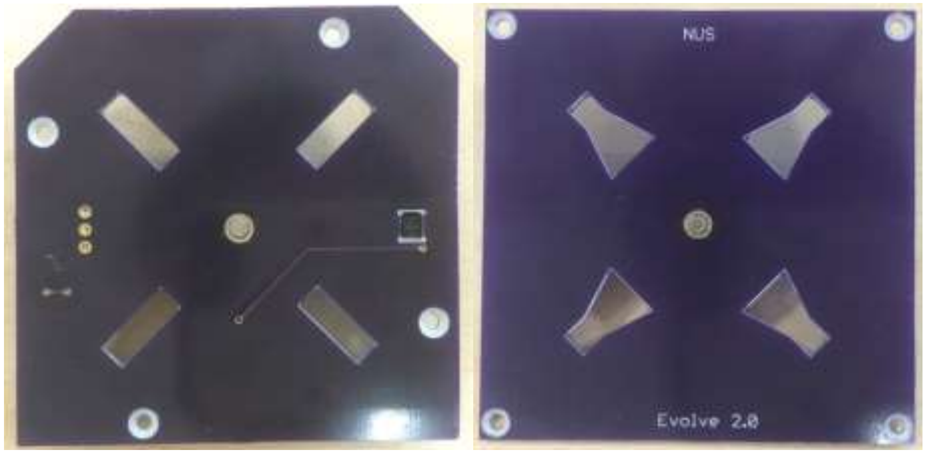


Figure 15 : Docking surfaces.

3.7 Docking Frame

The docking frame itself is also the chassis for each module. The actuators and sub-assemblies are attached to the uni-body frame. **Figure 16** shows a cross like structure of the active frame that houses 4 hook assemblies. In the center of the docking frame, a rectangle opening is used to hold the hobbyist servo. The frames are made from ABS (3D printed), and the internal volumes of the frames are hollowed out to maintain a light weight module.

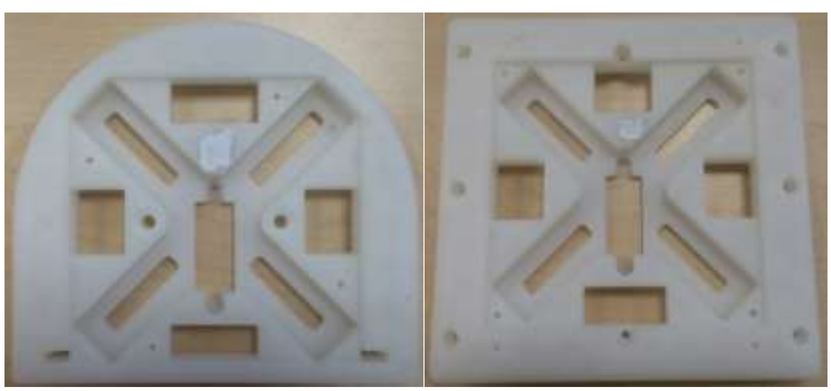


Figure 16 : Docking frame for active side.

3.8 Docking Hook Assembly

A docking hook assembly is fabricated from four individual components. **Figure 17** shows a breakdown of the components namely slider block, slider, securing hook and sliding arm (from left to right). The active surfaces are actuated by 3 independent servos (4 hook assembles per surface). The slider block constrains the slider to a linear motion while the location pin within the slide block positions the hook for deployment. The arm transforms a rotatory motion to a linear motion. Each hook is designed take forces up to 4.91N, an equivalent weight of a module hanging vertically off the hook. **Figure 19** shows the stress analysis model of the hook design. A defection of 1mm is expected at the maximum payload.

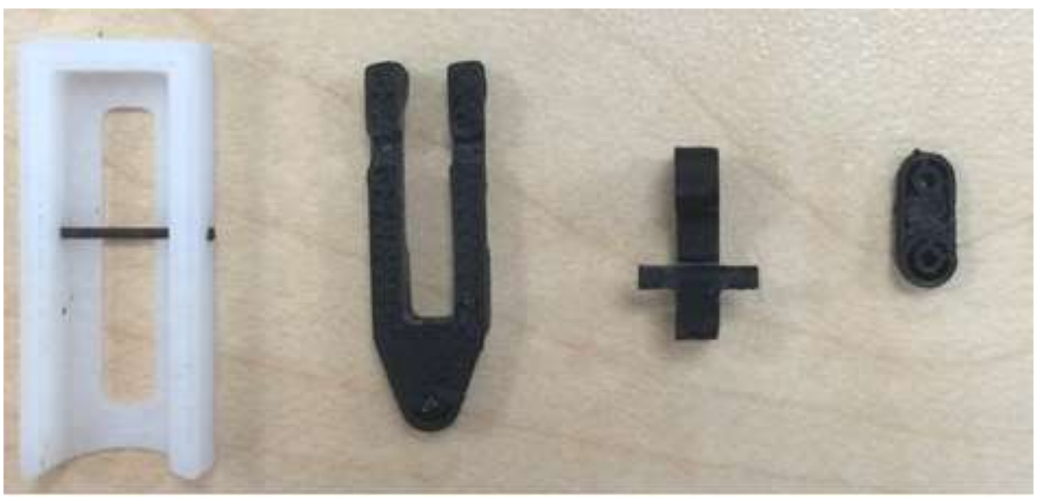


Figure 17 : Parts breakdown of a hook assembly.

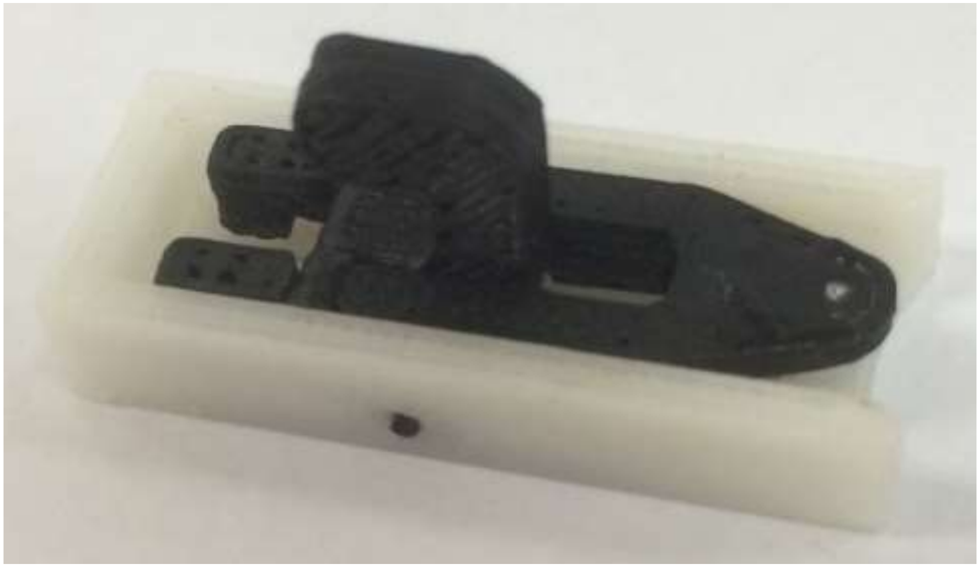


Figure 18 : Single assembled hook assembly.

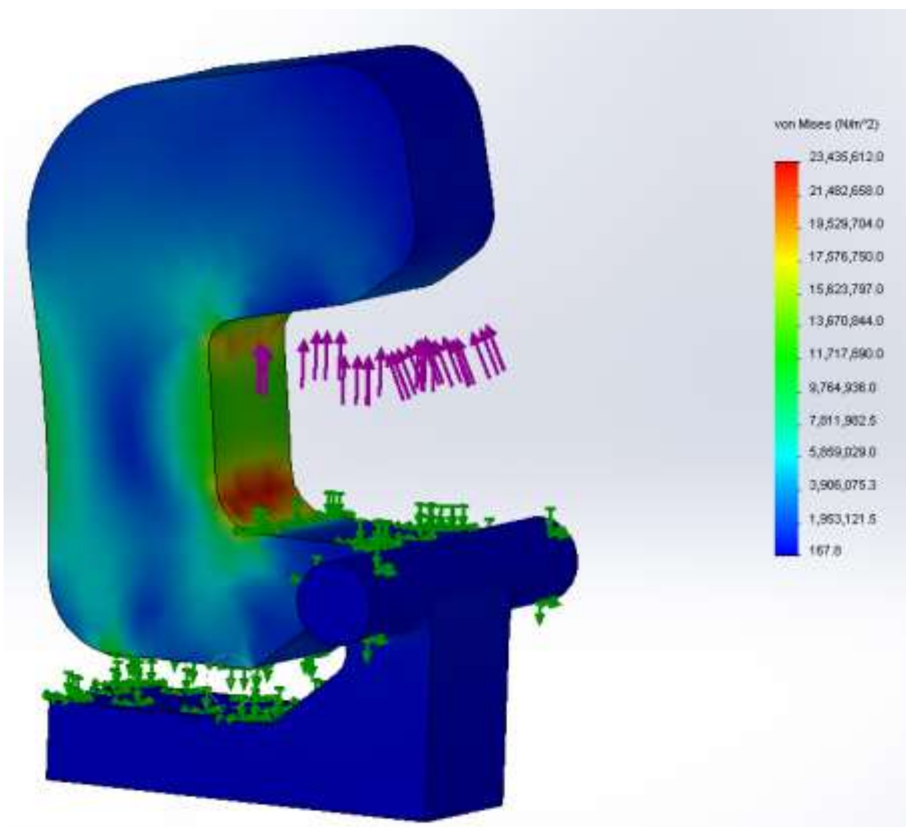


Figure 19 : Stress Analysis for Docking Hook.

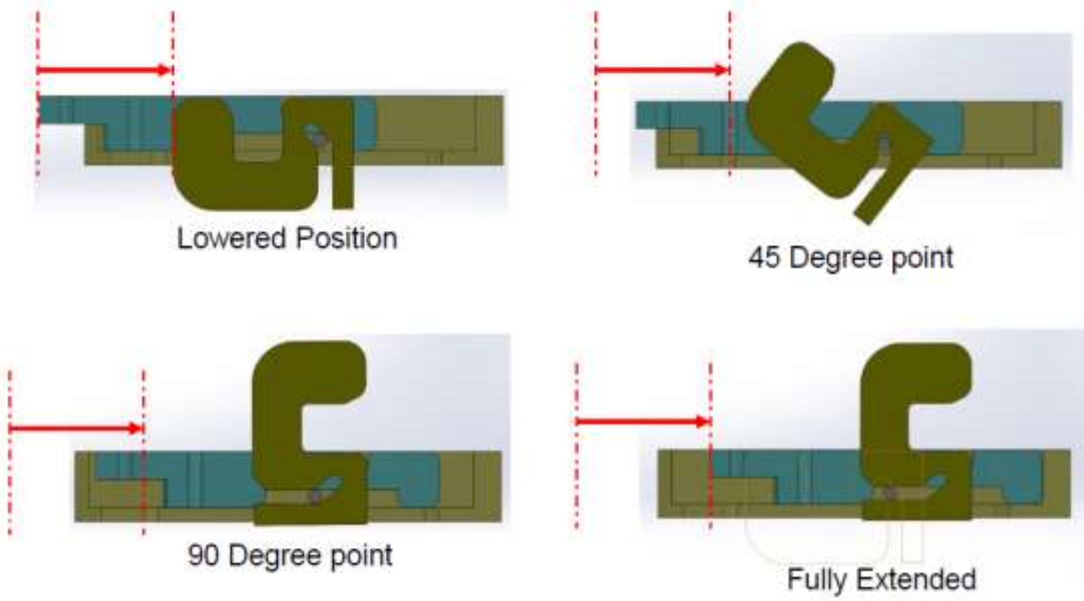


Figure 20 : Extension of hook assembly.

A complete hook assembly is shown in **Figure 18**. Rotational and linear motions are created when the slider is extended. **Figure 20** shows the cross-section view of extension steps in a hook assembly. A linear motion is generated by the cam and piston action from the servo actuator. The linear motion is translated into two phase motions of the hook assembly. The first motion phase rotates the 4 securing hooks out of the assembly and the second motion phase allows a linear extension. The linear extension ensures positive lock is established between two docking surface. From the red dotted lines shown in **Figure 20**, the rotary motion is completed when the slider is at 70% extended while the remaining 30% travel is used for linear extension of the hook.

The active surface (white semi-cylindrical block) is equipped with 4 securing hooks that can performance attachment on any passive surface (black semi-cylindrical block). **Figure 21** shows the different between an active and passive surface. Each active surface is actuated by a 5-gram hobbyist grade servo motor.

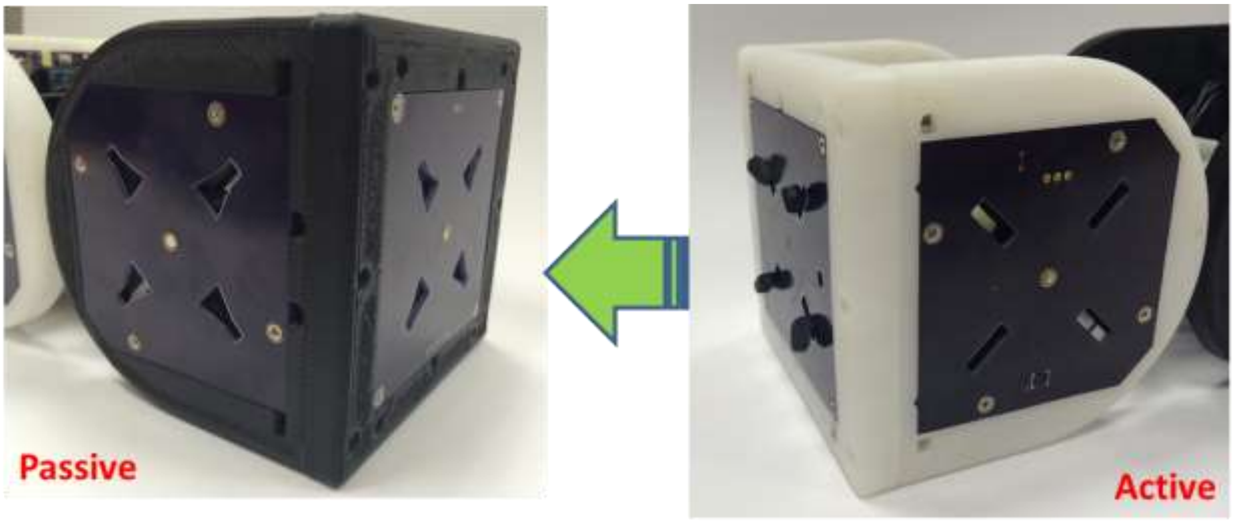


Figure 21 : Passive and Active Surface.

Figure 22 : Passive and Active Surface.

The securing hooks were manufactured using Acrylonitrile Butadiene Styrene (ABS) plastic through Fused Deposition Modelling (FDM) 3D printer. Evolve Generation 2 securing hooks shows (**Figure 22**) the capability of maintaining positive lock while performing basic locomotion. The ability of controllable docking surface enables Evolve 2 to be classified as self-reconfigurable modular system, a significant advantage over Evolve 1.

3.9 Actuators

Dongbu Company's Herkulex actuator is selected for its Smart Actuator capability. A Smart Actuator is the concept of integrating speed reducer, controller, driver, and network function all in one module.

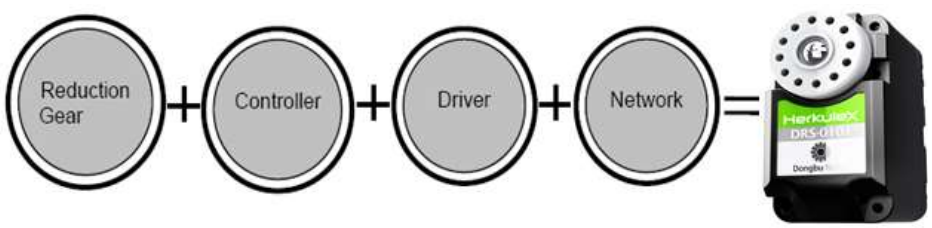


Figure 23 : Dongbu Herkulex Actuator Block Diagram.

The key advantage of Herkulex actuator is its compact size and light weight. Despite it's compact size, each actuator generates twice the torque of Evolve 1. The actuator allows position and speed controls with good resolution (0.3 degree resolution). It is also equipped with sensory feedbacks such as torque and current drawn. Communication with the actuators can be made through daisy chain, reducing the amount of wiring between controller and actuators. A structured communication protocol allows efficient setting of actuator speed, position, torque and other parameters through a single command packet.

The external shell of the actuator is made from plastic similar to the material used on the ABS chassis. This actuator also has built in troubleshooting feature which notifies the controller when internal temperature, torque, or supply voltage deviates from values desired setting.

3.10 Locomotion and Degrees of Freedom

Evolve Generation 2 is composed of three actuators as shown in **Figure 24**. Each actuator has a freedom of motion of 180 degrees. To simplify the rotation, the neutral rotation is considered as where all actuators are at 0 degree position. Each actuator can either rotate to positive or negative 90 degrees. The docking surface allows connections in four orientations at 90 degrees interval.

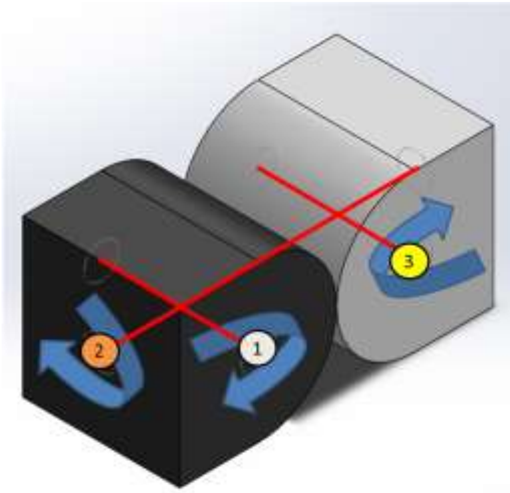


Figure 24 : Three Degrees of Freedom.

Locomotion control can be stored as gait tables. The columns represent intervals in time while the rows represent different actuators in each module. Following is a sample gait matrix table of actuator position values.

$$\begin{bmatrix} 0 & 90 & 0 & 90 \\ 0 & 45 & 0 & 45 \\ 90 & 0 & 90 & 0 \end{bmatrix} \quad \textbf{Equation 1}$$

The first row represents actuators 1, the second row represents actuators 2, and the third row represents actuators 3 in **Figure 24**. To add an additional module in the modular system, three more rows to be added to the matrix. For each module to be placed in a physical world, the modules can be rotated in 2 axes as shown in **Figure 25**, rotations in X-axis and Y-axis. Any combination of the X and Y rotations can reach 4 congruent modular configurations. A group of congruent modules are defined as {Neutral, X_{180}, Y_{180}, $X_{180}Y_{180}$} where neutral is zero rotation and the other elements are a combination of rotations by 180 degrees. **Figure 26** shows four examples of possible congruent configurations.

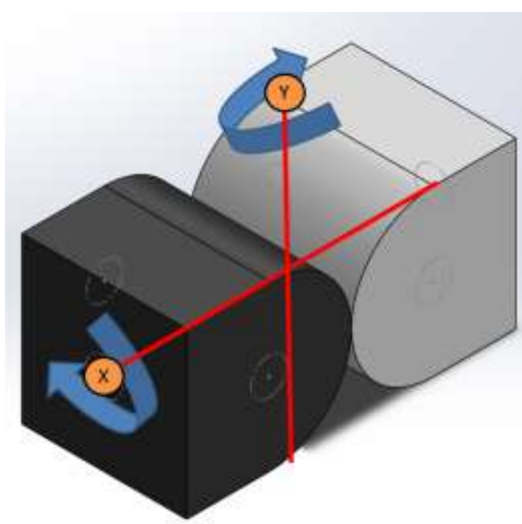


Figure 25 : Possible Rotation Axes.

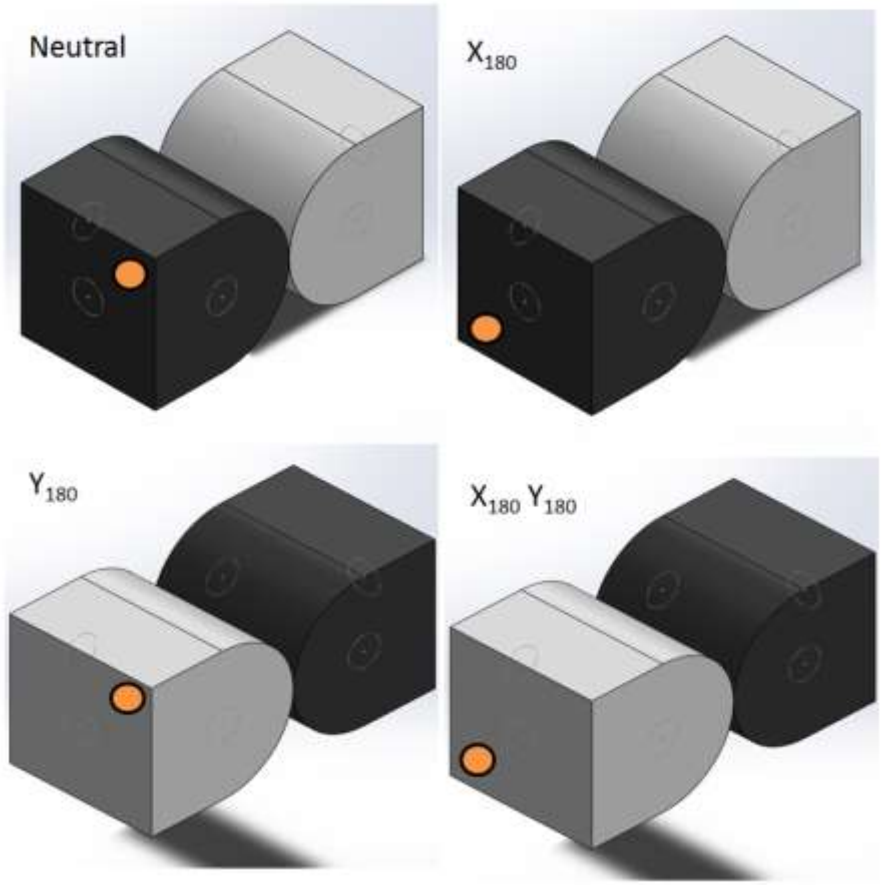


Figure 26 : Four Congruent Possibilities.

It is assumed that all initial configurations of the Evolve modules are known and control transformations can be accomplished easily by matrix multiplication of the initial gait matrix table. For instance, a rotation by X_{180}, the equation will be as follows.

$$\begin{bmatrix} -1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & -1 \end{bmatrix} \begin{bmatrix} 0 & 0 & -90 & -90 \\ 0 & 0 & 0 & 0 \\ 0 & -90 & -90 & 0 \end{bmatrix} = \begin{bmatrix} 0 & 0 & 90 & 90 \\ 0 & 0 & 0 & 0 \\ 0 & 90 & 90 & 0 \end{bmatrix}$$ **Equation 2**

The control transformation for each of the associated rotations are {Neutral, X_{180}, Y_{180}, $X_{180}Y_{180}$} respectively. By using gait matrix table, the locomotion can be easily computed, however, the transform matrix tables grow in $O(n^2)$ with increase in module size.

$$\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}, \begin{bmatrix} -1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & -1 \end{bmatrix}, \begin{bmatrix} 0 & 0 & 1 \\ 0 & 1 & 0 \\ 1 & 0 & 0 \end{bmatrix}, \begin{bmatrix} 0 & 0 & -1 \\ 0 & 1 & 0 \\ -1 & 0 & 0 \end{bmatrix}$$ **Equation 3**

Figure 27 shows different rotated configuration of an Evolve Generation 2, the additional axle in the center of the module allow configuration that can never be achieve by the Evolve 1.

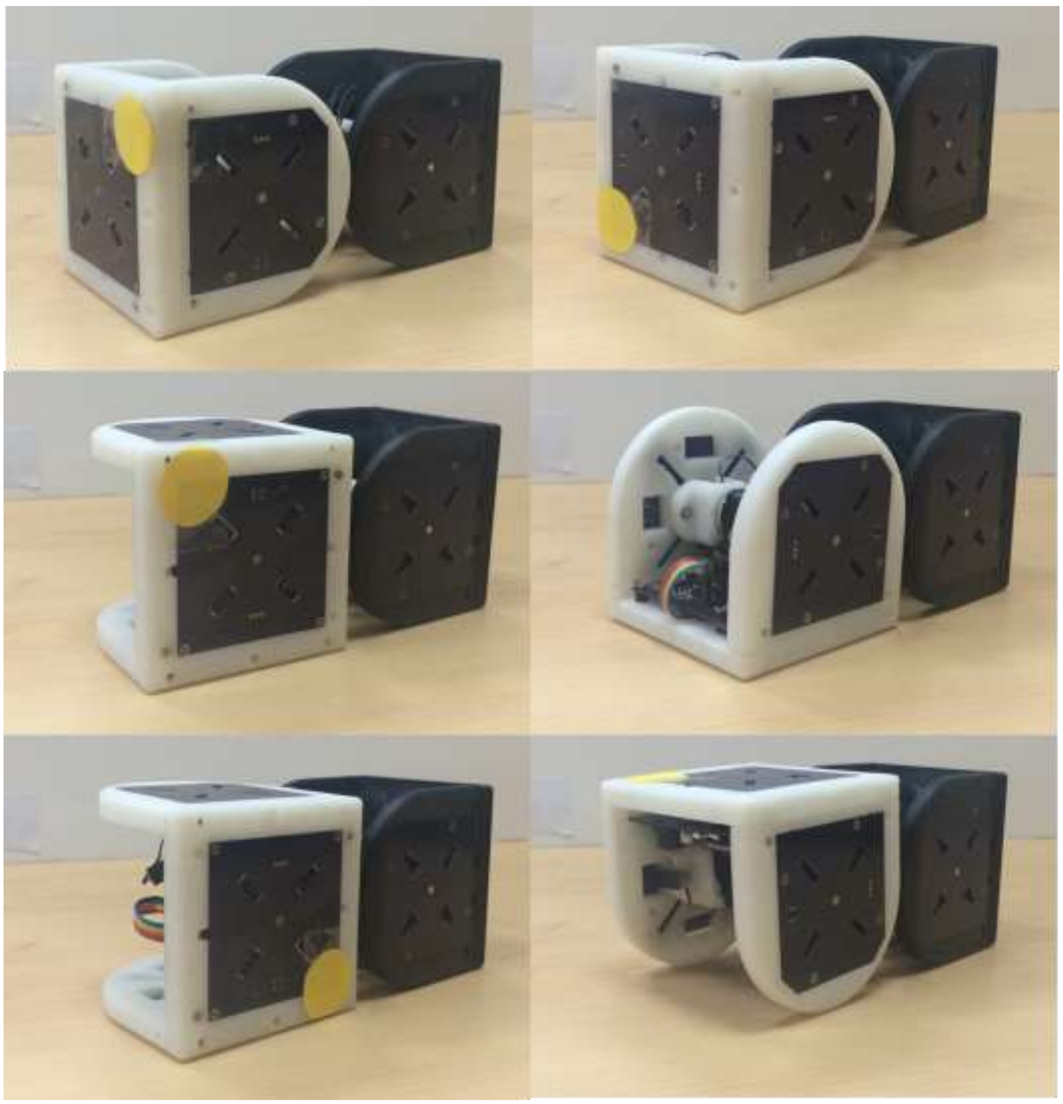


Figure 27 : Sample rotated configuration from Evolve Generation 2.

Chapter 4 Electrical Systems

4.1 Electrical Design

The electrical system of the Evolve 2 has several modules, computational, communication, drive and inertial measuring units. The computation module is the brain of the robot and, it processes all the arithmetic operations and controls the actuators. The communication module allows data transfer between the modules and a host computer. The drive module handles all movements of the module and of the actuators. The inertial measurement unit allows the module to sense the environment and provide feedback to the computation module.

4.2 Electrical Designing Tool

The design of electrical circuits is performed using EAGLE - Easily Applicable Graphical Layout Editor, which is PC-based circuit board design software. The unified electronics design feature allows the schematic design to be transformed into a printed circuit board.

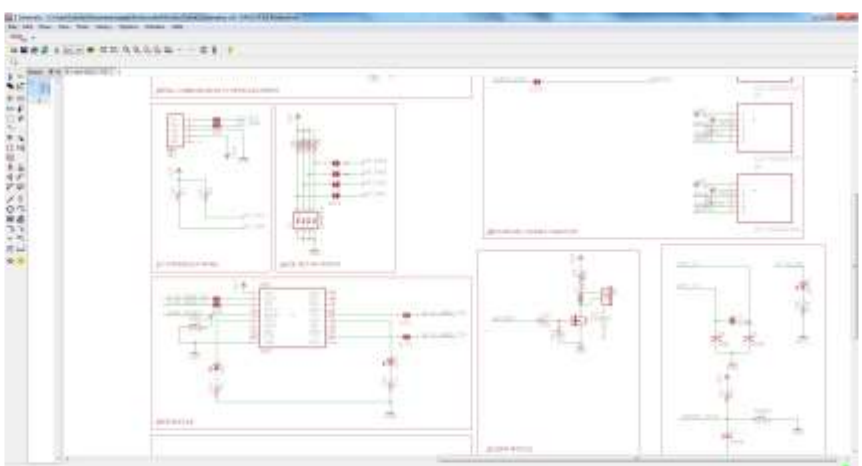


Figure 28 : EAGLE Circuit Design Software.

4.3 Main Circuit Board

The main circuit board consists of central processing unit, buzzer, switches, regulators and communication modules. The central processing unit is positioned at the bottom layer to reduce the overall height profile of the circuit board. Since there is no visual display on the robot module, different frequencies generated by the buzzer are used as audio feedback for the developers. There are four DIP switches on the main board, allowing the developers for manual configuration of the block. There are two regulator circuits to ensure a constant power to be supplied to all the components. The wireless communication unit is placed at the edge of the main board. Inertial measurement unit is located near the centre of the module. There are three connectors for the active docking surface circuit.

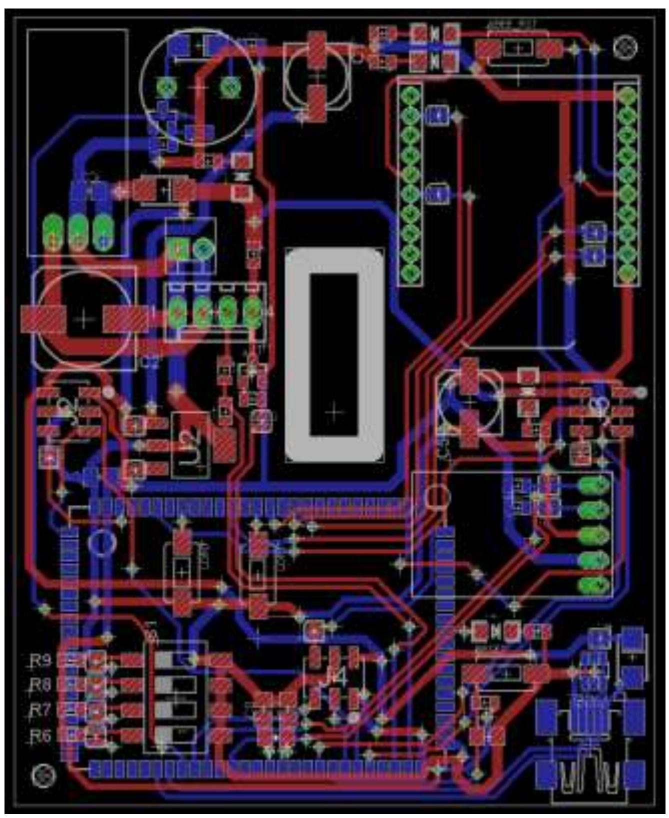


Figure 29 : Main Circuit Board.

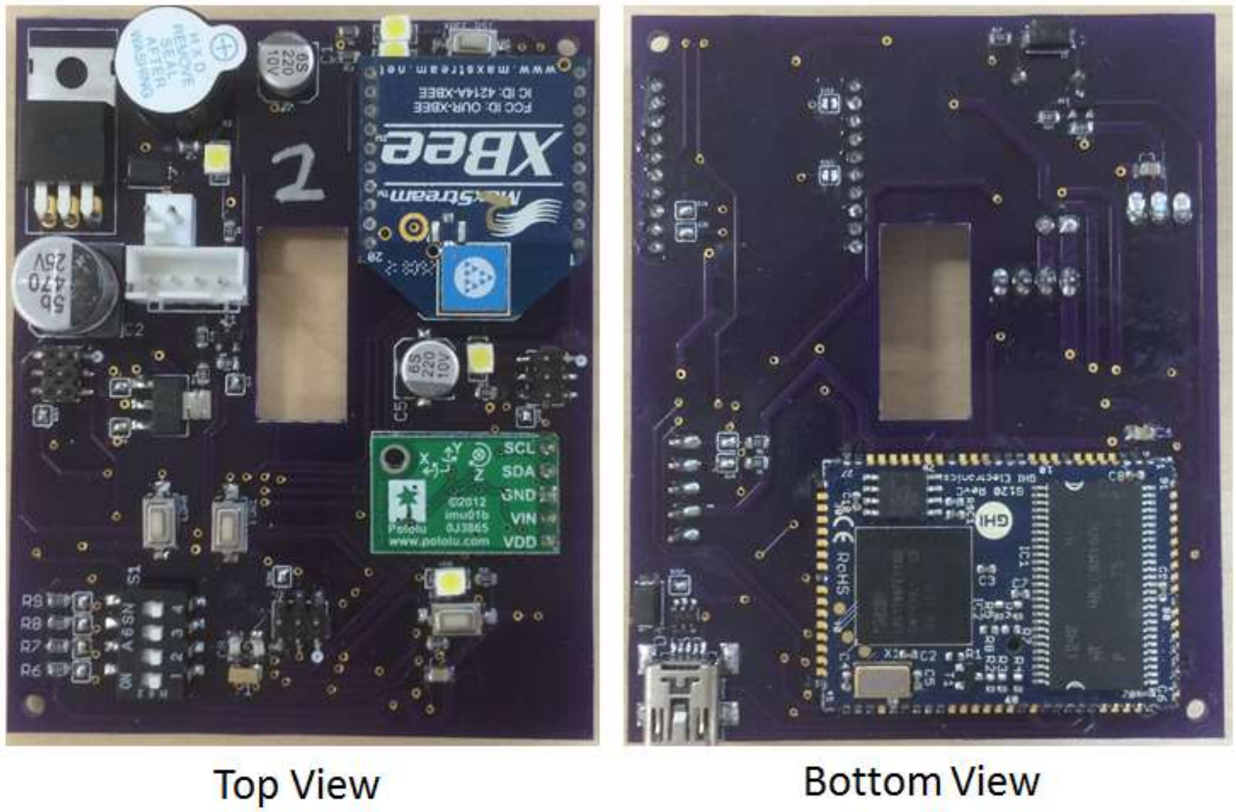


Figure 30 : Actual View of Main Circuit Board.

4.4 Connection with Actuators

To control the actuators, a Universal Asynchronous Receiver / Transmitter protocol is used for data transfer. The circuit diagram for the actuators is shown in **Figure 31**.

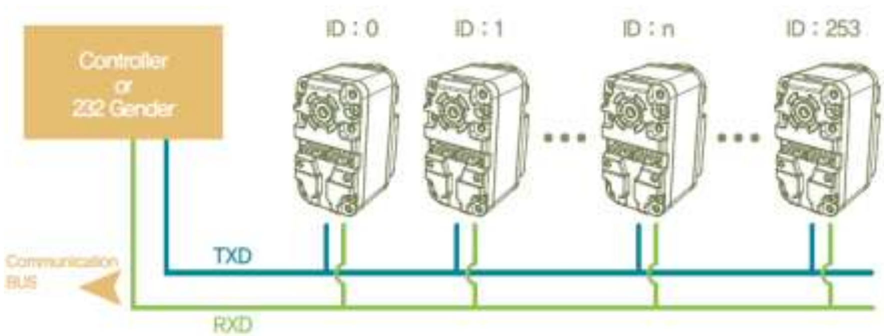


Figure 31 : Communication Network of Actuators.

The power is supplied to the actuator from the main controller through Pins 1 and 2 of the connector. Unlike Evolve 1, the new design connects the actuators only through the main board, this disable external control of the actuators. The Herkulex actuator operates on TTL 5V logic while the processor on 3.3V logic, a voltage level shifting is used to manage different voltage levels.

4.5 Inertial Measurement Unit

To determine the orientation of the modules, an inertial measurement unit (IMU) that packs an L3GD20 3-axis gyro, an LSM303DLHC 3-axis accelerometer and 3-axis magnetometer is installed on the main circuit board. The module tracks its orientation by locating earth gravity pull. For the direction, the magnetometer is used with earth magnetic north pole. The gyroscope within the module allows tracking of module rotation in comparison with actuator inputs.

Figure 32 : Inertia Measurement Unit.

The L3GD20 and the LSM303DLHC have many configurable options, including dynamically selectable sensitivities for the gyroscope, accelerometer, and magnetometer, as well as a choice of output data rates for each sensor. The two ICs can be accessed through a shared

I^{2}C interface, allowing all three sensors to be addressed individually via a single clock line and a single data line. The nine independent rotations, acceleration, and magnetic readings (9 degrees of freedom) provide all the data needed to make roll, pitch and yaw detections.

4.6 Computation Module

There are several types of micro-controller and processor in the industry. The Motorola 68HCXX series, Atmel Mega series, Microchip PIC, and the Intel 8051 are some extremely common ones. Each family usually has a number of variations. The 68HCXX, for example, has 12 variations and the Microchip PIC family has around 30. It is important that the central processor is able to perform real-time computation of locomotion. The GHI Electronics Company's G120 Module is chosen for Evolve Generation 2 as the main processor. It is a surface-mount System on Module (SoM) that runs Microsoft .NET Micro Framework software platform; a tiny version of Microsoft .NET framework. The value of G120 Module is not only in the hardware capabilities such as the ARM Cortex-M3 processor, memory and peripherals, but also is in the integration between the hardware and the embedded software.

Figure 33 : G120 Processor Module.

4.7 Communication Module

Both generation of Evolves uses ZigBee wireless transceiver. ZigBee is a low-cost, low-power, wireless mesh networking standard. The low cost allows the technology to be widely deployed in wireless control and monitoring applications. The low power-usage allows longer life with smaller batteries, and the mesh networking provides high reliability and larger range. ZigBee is based on the IEEE 802.15.4-2003 standard for wireless personal area networks (WPANs), such as wireless headphones connecting with cell phones via short-range radio. The technology defined by the ZigBee specification is intended to be simpler and less expensive than other WPANs, such as Bluetooth. An advantage for using the same wireless transceiver is data transmission between the two generations of modules.

Figure 34 : Xbee Communication Module.

Chapter 5 Software Design of Evolve

5.1 Software Design

The software system consists of a series of protocols and firmware. The codes are written in C# language as the central processing unit executes within the Microsoft .Net micro framework. Visual Studio is used as the compiler for the codes.

5.2 Actuator Communication Protocol

An instruction packet is transmitted by the main controller to the actuators with the following format.

OXFF	0XFF	ID	LENGTH	INSTRUCTION	PARAMETER1	…	PARAMETER N	CHECK SUM

Figure 35: Instruction Packet.

The first two OXFF of each packet byte indicate the commencement of any incoming packet. The ID tag is the unique ID value for an actuator. The ID 0XFE is the broadcasting ID which indicates all the connected actuator units that all the packets sent with this ID applies changes to all the units on the network. Thus there is no return status packet. LENGTH is the length of a packet being sent. It is used as an error checking parameter to determine whether the transmission is complete. INSTRUCTION is the instruction tag used for the actuator to identify if it is a move or request status command. PARAMETER0…N is the additional information needed to be sent other than the instruction itself. CHECK SUM tag checks for completeness of the data sent, which is computed as follows: Check Sum = ~ (ID + Length + Instruction + Parameter 1 + … Parameter N).

For every instruction packet sent by the main controller a status packet is replied back. These status packets give information of the actuator to the controller in the network.

OXFF	0XFF	ID	LENGTH	ERROR	PARAMETER1	PARAMETER2	…	PARAMETER N	CHECK SUM

Figure 36 : Status Package.

5.3 Wireless Network

The modules communicate through a wireless setup. Unlike Evolve Generation 1, the second generation is not equipped with physical connection between the modules. There are three types of topology used for wireless connection. Data need not be transmitted to every module in the system. Point-to-point allows modules to communicate independently in the system and information can be sent to an assigned module. Multicast is also practiced in some cases, when a global status changes and information needs to be updated to the robot quickly. In this event, multicast protocol is able to update multiple robots simultaneously. PC host generally uses multicast mode to transfer data. The repeater mode allows data communication between distant modules. In the event when a module is unable to send data directly to the receiving module, the message is relayed through intermediate network nodes. **Figure 37** below shows the types of communication topology of the wireless system. Squares represent the modules and an arrow represents a wireless connection.

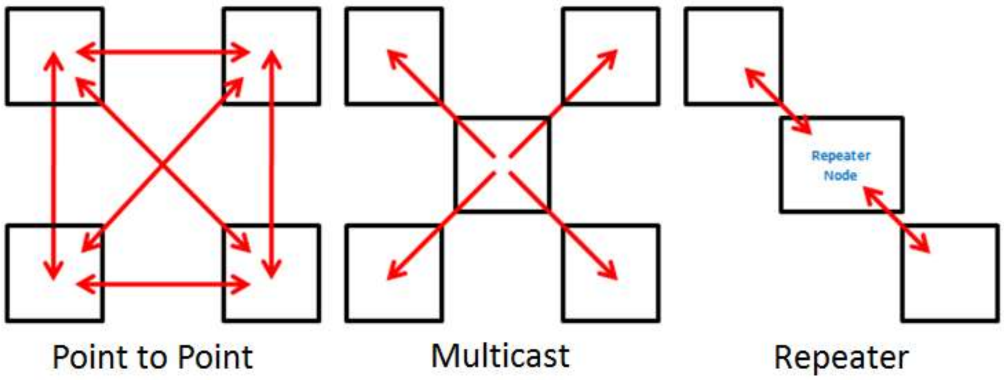


Figure 37 : Types of Wireless Communication Network.

5.4 Communication Protocol

Instead of sending raw data directly across the network, the data is sent in packets similar to the actuator control packet. There are two types of packets, the “Transmit Packet” sent from the module and the “Receive Packet” received from another module. For the system connection, each module has a unique ID assigned to the network node. A transmit packet is sent with the format in **Figure 38**.

OX7E	SIZE	IDENTIFIER	FRAME ID	ADDR	Options	OPCODE	PARAMETER(0...N)	SUM CHECK

Figure 38 : Transmission Package.

Start delimiter of the packet is 0X7E, indicating the wireless component that there are data waiting to be transferred. Data size is indicated by using two bytes, which allows the controller to know whether the packet has ended. Identifier indicates the format in which data is transferred. Frame ID identifies the UART data frame for the host to correlate with a subsequent acknowledgement (ACK). Address indicates the receiving node address and 0XFFFF indicates a multicast transmission. Options are useful to setup the conditions for the receiving node to send an acknowledgement. OPCODE is the instruction for the robot. Parameters are data related to the instructions given. Lastly, sum check checks for completeness of the data transmitted. The transmit status is acknowledged after each data transmission. The transmission packet is shown in **Figure 39**.

OX7E	SIZE	IDENTIFIER	FRAME ID	STATUS	SUM CHECK

Figure 39 : Transmission Status.

A start delimiter and size byte to indicate the start of a packet. The Identifier indicates that it is a transmission receive packet. Frame ID identifies UART data frame being reported. The status is depicted using a field which can take 4 values: '0' for a successful transmission, '1' to indicate the status that all retries completed and no ACK is received, '2' to indicate a successful broadcast and '3'to indicates the time out of an indirect transmission.

The transmission status is to be automatically generated and send back to the host. The client receives the packets in in the format shown in **Figure 40**.

OX7E	SIZE	IDENTIFIER	ADDR	RSSI	Options	OPCODE	PARAMETER(0...N)	SUM CHECK

Figure 40 : Receive Package.

Start delimiter, size and identifier are in the same format as the transmit packet. Address indicates the sender ID. The received signal strength indicator is used as a guide if the signal strength is too weak for direct transfer. 'Options' is a type of transmission used by the sender: indicating whether the originator is the sender, or whether the message is repeated from the network. OPCODE is the instruction including the detailed information sent to the blocks. Lastly, a sum check is used to reconfirm that the message received is complete.

5.5 Transmission Flowchart

The algorithm for data transmission goes through a few states. A subroutine in the program checks and packet the data before transmission. **Figure 41** shows the flowchart in which data is transmitted. In the initial phase, the processor arranges the data in order. After which it checks the bus whether it is ready for transmission. A successful transmission status will be returned and unsuccessful transmission will be acknowledged. The sender has to take extra action depending on the replied status. In some cases, the sender has to repack the data into a repeated message, so that the nodes in the network can repeat the message to the system until the receiver receives the message.

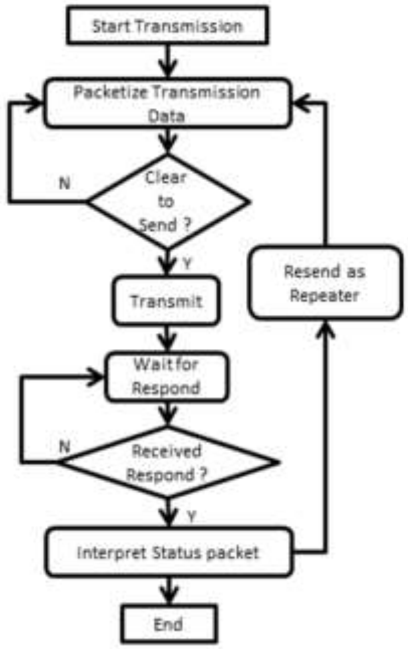


Figure 41 : Transmission Flowchart.

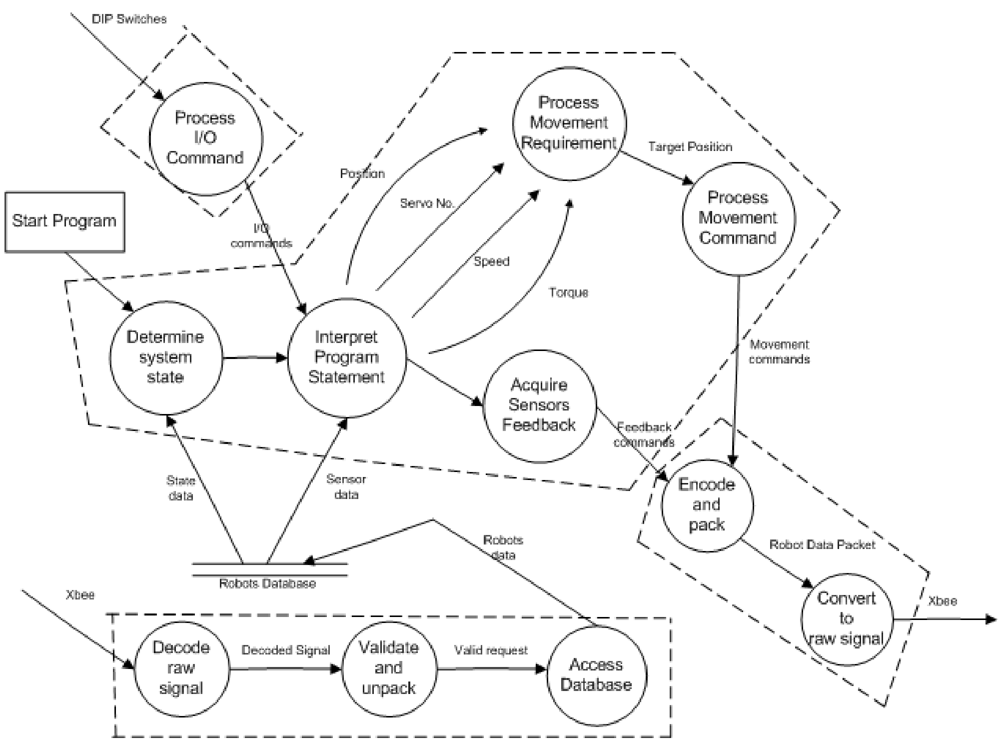


Figure 42 : Transmitter Dataflow Diagram.

5.6 Receiver Flowchart

The receiver flowchart in **Figure 43** shows how the incoming data are received. Upon experiencing a hardware interrupt by the USART ports, the processor starts unpacking the data received. There are 2 major types of data: instruction and repeater. For instructions, the robot module follows the command order by the sender. As for a repeater, message is resend to the directed receiver.

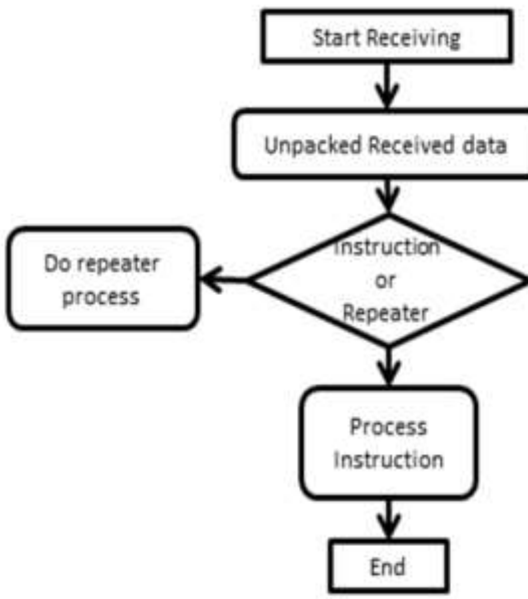


Figure 43 : Receiver Flowchart.

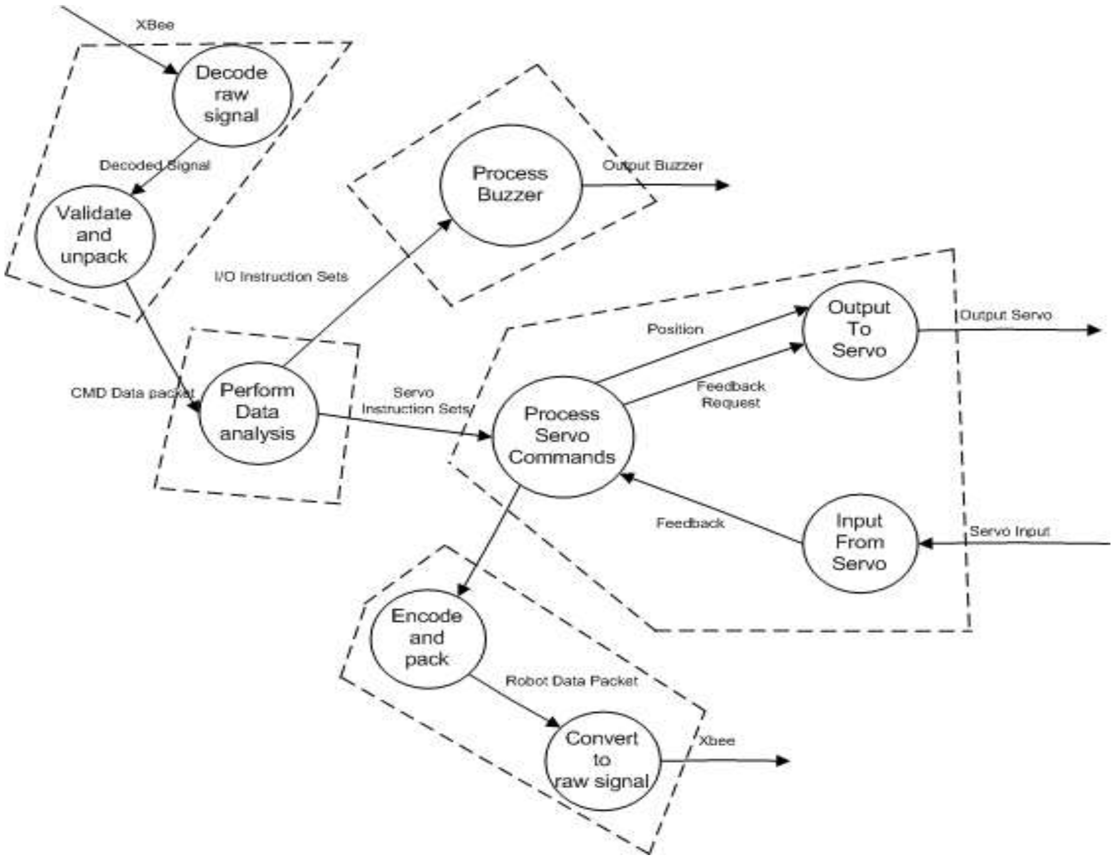


Figure 44 : Receiver Dataflow Diagram.

Chapter 6 Locomotion Generation

6.1 Development of Simulator

Simulation is the most cost effective way to test locomotion of the Evolve System. The following criteria are used to select a suitable simulation framework:

Microsoft .Net Framework. The computational unit on Evolve Generation 2 is enabled to run Microsoft .Net Framework. This enables the simulation engine to be built into the.

Execution Package Size. The Evolve Generation 2 has a much more powerful computational unit. There is a need to reduce the execution package size as this computational unit does not have massive memory storage like a conventional computer.

Parallel Processing. The simulator should be able to allow parallel processing, as it is an efficient approach taken as Evolve is a modular system.

After experimenting with several simulation engines such as Chrono Physics Engine and BulletX, a simulator using the C# and the Microsoft Visual Studio environment is developed. The physics engine is based on Microsoft XNA Game Studio 4.0 and Farseer physics library. The simulator can run on both conventional computers and on the modules. Farseer physics is general purpose C# with Microsoft XNA Game Studio physics engine. It comprises of basic physics rule and a simulation environment.

6.2 Modelling Evolve Module in Simulator

Evolve Generation 2 is represented as a 2D model in simulation. The model is generated based on the physical properties of each module shown in **Table 2**.

No.	Description	Properties	Value
1	Friction coefficient between module and simulation environment	Friction	0.7
2	Actuator maximum torque output	Torque	24kgcm
3	Actuator maximum rotational velocity	Velocity	0.882rps
4	Weight of module	Weight	495g
5	Docking connection	Force	20N

Table 2 : Physical properties of each module.

For simplifying the modules for 2D simulation, only two axes are used for computation (roll axis is disabled). Each module has an overview dimension of 180mm x 90mm x 90mm, shown in **Figure 45**.

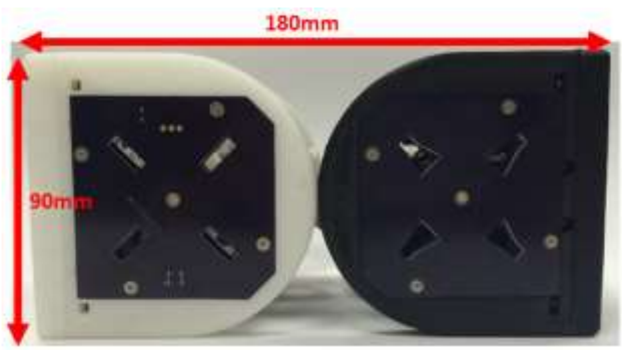


Figure 45 : Overall dimension of Evolve module.

In the simulation, both active and passive ends of the module are simulated with a circle and rectangle block as shown in **Figure 46**. Similar to the actual physical model, each actuator is represented as a revolute joint. The revolute joints are constrained to rotate within ± 90 degrees.

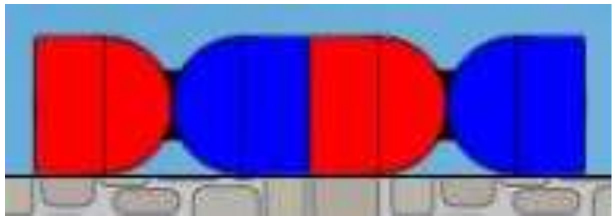


Figure 46 : Simulation view of 2 connected modules.

For all connections or dockings made between two modules, the linkage is made possible by weld joints. The weld joint threshold is set to approximately 19N which ensures an accurate simulation of forces acting between two modules. If the force exceeds 19N, the weld joint will break and the module will be disconnected. This simulates the actual world strength of the docking hooks. For other test cases, the threshold can be adjusted for optimal hook strength for a defined configuration. **Figure 47** shows the physical model of two connected modules. The white shaded area represents the collision model (physics model) between the modules in the simulation environment. The smaller blue dots indicate the revolute joint and small right angle green lines show where a weld joint is connected.

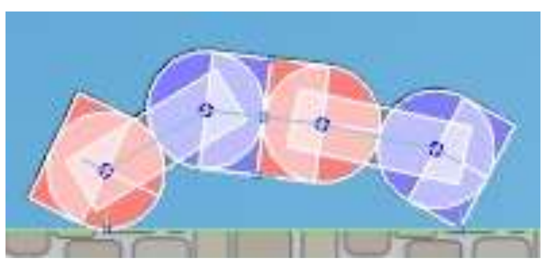


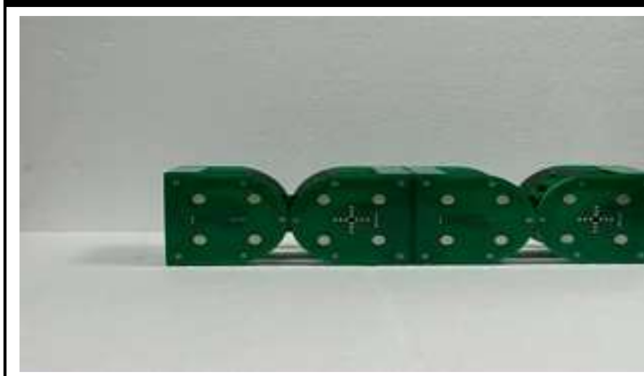
Figure 47 : Physic model of two connected modules.

In the simulation, a unique collision group is assigned to each module, real world mechanical parameters such as friction and gravity are defined within the simulation environment. **Figure 48** shows an overview of two connected modules in the simulation environment.

Figure 48 : Screenshot of Evolve Simulation.

6.3 Defining Scenario

The basic of locomotion for Evolve modules is to move toward a certain point. The basic locomotion adopted for Evolve 2 is the crawl sequence. The crawl behaviour of locomotion can be seen in caterpillars. **Table 3** explains of the sequence of crawl motion, using Evolve 1, and the gait matrix table is shown in **Equation 4**.

Displacement of modules	Description
	Initial orientation for all joints and connected modules on an even terrain. A total of four actuators are shown in the sequence. The direction of travel is towards the left.

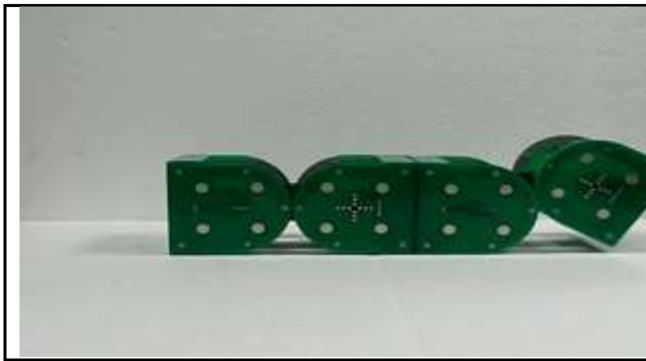
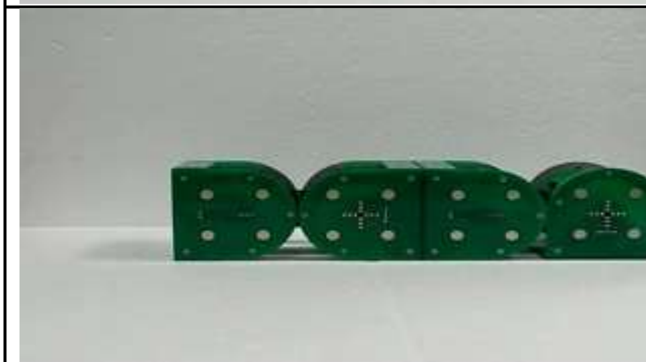
	Two actuators on the right module had angled to reposition the right module for the sequence.
	The modules are in-position for a motion for displacement.
	The modules are levitated of the terrain with rotation motion of the extreme right and left actuators.
	The crawl sequence is completed when all modules are in full contact with the surface.

Table 3 : Basic crawl sequence for 2 connected modules.

For the experiment, the time taken for the modules to displace 1 unit block away from an initial position is 2 seconds. Therefore, it is needed to study whether Evolve is capable of a faster locomotion. The parameters interested in are the speed and distance covered per simulation iteration over an even terrain.

$$\begin{bmatrix} 0 & 0 & 0 & 0 & -90 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 45 & 45 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 90 & 90 & 0 \end{bmatrix}$$

Equation 4

6.4 Proposal for Locomotion

The locomotion of Evolve Generation 1 was hard coded with a certain pattern. In this work, we will deal with whole-body locomotion and develop a unified framework applicable to a given module configuration. The generated locomotion can be chosen for its velocity and displacement within a specified simulation time. No predefined pattern is utilised for the locomotion generation. Constraints imposed by the physics model will prevent locomotion generation beyond the physical capability of Evolve 2.

6.5 Computational Intelligence for Locomotion Search

Random locomotion generation is extremely time consuming and may not deliver any feasible solution. The standard deterministic approaches for linear or sequential search are not suitable as there are multiple variables and cases with large number of combinations (joints and actuators) for a self-reconfigurable modular system. The proposed scheme is to search for optimised locomotion parameters with the help of Computational Intelligence. There are many computational intelligence methods, such Neural Networks, Fuzzy System and Evolutionary Computation. Neural Networks (NN) [26] is an emulation of biological neural system; it can perform tasks that a linear program cannot. Unfortunately, NN requires large training data to operate. In this work, we don't have dataset that can be validated and the architecture of a neural network is different from the architecture of microprocessors therefore needs to be emulated artificially. It also requires high processing time for large neural networks which the Evolve 2 does not have. Fuzzy System [27] is useful when dealing with ambiguous concept; it requires tuning to a specific application. The curse of dimensionality makes nearly impossible in practise to set up a rule-base with more than three inputs while preserving end-user interpretability. Evolutionary Computation [28] represents a powerful search and optimization paradigm. The main characteristic of evolutionary algorithm is the intensive use of randomness and genetics-inspired operations to evolve a set of candidate solution.

Comparing the different computational intelligence, Evolutionary Computation is proven to be successful in many other robotics locomotion searches [29] [30] [31]. It is also well-suited for operational conditions and where the requirements are not known or not well defined prior to system deployment. While Neural Networks and Fuzzy System are suitable for optimization problem with a given parameters. Hence, in this work Genetic Algorithm from the class of Evolutionary Computation is selected. The role of Genetic Algorithm is primarily as a searching tool and the evaluation of the best locomotion suitable for Evolve2.

6.6 Evolutionary Computation

Evolutionary Computation paradigms provide tools to build intelligent systems that model the process of natural evolution. Natural evolution offers an explanation of the biological diversity and its underlying mechanisms. Given an environment that can host only limited number of individual, and the basic instinct of individuals to reproduce, selection becomes inevitable if the population size is not to grow exponentially, Natural selection favours those individuals that compete for the given resources most effectively, in other words, those that are adapted or fit to the environment conditions best.

The evolutionary computation is classified in five areas, but the common underlying ideas behind all these techniques are the same.

1. Genetic Algorithms
2. Evolutionary Programming
3. Evolution Strategies
4. Genetic Programming
5. Particle Swarm Optimization

6.7 Genetic Algorithm

In this work, the focus is to explore the widely use Genetic Algorithm for searches of possible locomotion. Genetic Algorithm is advantageous in multi-dimensional searches which is well suited for a modular system. The original Genetic Algorithm was first invented in the early 1970's by John Holland; he then published a book in 1975 on

"Adaption in Natural and Artificial Systems" [32]. Genetic Algorithm begins with a set of candidate solution, and each candidate solution is assessed for a fitness measure (i.e. Higher better). Based on the fitness, some of the better candidates are chosen to seed the next generation by applying recombination and/or mutation to them. The recombination is an operator applied to two or more selected candidates (parents) and results one or more new candidates (children). Mutation is applied to one candidate and results in one new candidate. Executing recombination and mutation leads to a set of new candidates that compete based on their fitness with the old ones for a place in the next generation. This process can be iterated until a candidate with sufficient quality (a solution) is found or a previously set computational limit is reached.

6.8 Gene Encoding

In Genetic Algorithms each gene is encoded into a chromosome. For the application, a gene is the delta change in actuators rotations per step. The generation of gene is random but limited by the actuator's maximum performance. Chromosome is a combination of the individual genes. Population is the entire generation of chromosomes.

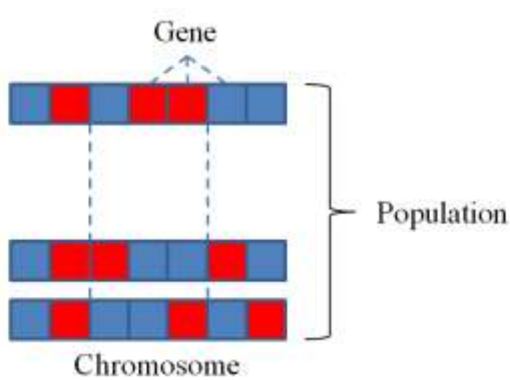


Figure 49 : Gene and Chromosome.

6.9 Parameter Settings

On top of a generation of genes and chromosomes, the Genetic Algorithm (GA) needs parameters such as crossover and mutation rates. It is difficult to ascertain the optimal crossover and mutation rates. Too high a crossover rate could result in premature convergence while having a low crossover rate could result in slow convergence. On the other hand, high mutation rate allows for a solution outside the original population but might also lead to slower convergence. A random crossover rate between 25% and 75%, and a mutation rate of 10% are selected. A population size of 100 is maintained for each trial cycle, which is decided based on the computation limitation of each module. The test data are externally stored to allow testing of large number of generations possible. **Table 4** presents a summary of the GA parameter settings.

Parameters	Value
Population	100
Mutation Rate	10%
Crossover Rate	25% to 75%
Elite	25%

Table 4 : Parameters used in Genetic Algorithms.

6.10 GA Procedure

The flowchart in **Figure 50** shows the flowchart of the genetic algorithm deployed.

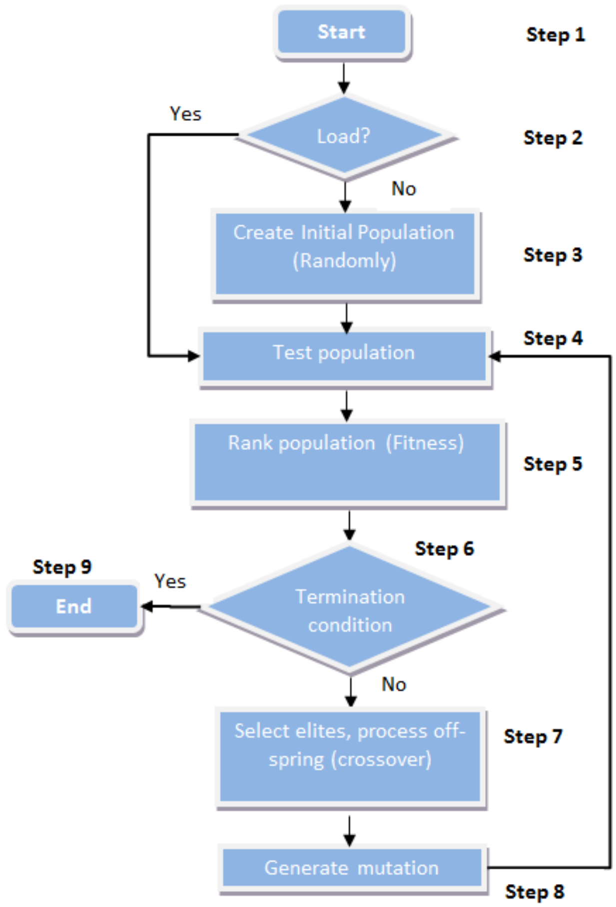


Figure 50 : Flowchart for Evolve locomotion Genetic Algorithm.

Step 1: The program starts, loading the environment and basic parameters that affect the physics model.

Step 2: The program is tasked to load existing population or dataset. If the attempt is to search for an optimal solution, the program loads the existing work. On the other hand, the program will go to Step 3 if a new initial population is observed.

Step 3: Generation of 100 chromosomes as population. An initial filter is used to remove the non-conforming physics entities from the population. Upon generating 100 possible chromosomes, the program proceeds to Step 4.

Step 4: All the 100 chromosomes are tested for 2 seconds of simulation. The displacement and speed are recorded in a database. The program tracks any non-conforming performance of the test casings; if a test case is found beyond the module capability, the test case is ranked with zero fitness. A fitness function is used to determine whether a solution is acceptable or not.

$$\text{Fitness} = \frac{maximum\ displacement\ in\ positive\ Y\ axis}{total\ time\ taken\ for\ single\ simulation} \qquad \textbf{Equation 5}$$

Step 5: Population fitness and average population tables are generated in this step. The population are sorted in fitness rank order. The highest order solution is a potential solution and the lower order solutions are used for possible crossover and mutation in later steps. The average fitness is used to determine how a population has progressed over different iteration. If the average of fitness values do not progress for a period, it is likely that the solution had converged or the optimal solutions are found.

$$\text{Average Fitness} = \sum_{i=0}^{100} \left(\frac{maximum\ displacement\ in\ positive\ Y\ axis}{total\ time\ taken\ for\ single\ simulation}\right) \div population\ size \qquad \textbf{Equation 6}$$

Step 6: The termination condition is set to a predefined number of iterations of 200 cycles. If the termination condition is satisfied, the program proceeds to step 9.

Step 7: In this step, the off-springs are generated using elitism and crossover. Selected elitism mutates in search for other possible solutions. The top 20% elites are kept unchanged in the population while their duplicates are used to produce offspring. Roulette Wheel Selection method is employed for crossover. The Roulette Wheel Selection is implemented based on the following equation.

$$\boldsymbol{f_s = random * f_T} \qquad \textbf{Equation 7}$$

f_s is the selected function set for crossover, while f_T is the rank of the chromosome within the population. Each chromosome is stored as changes in the rotational angle of an actuator and hence the conventional genetic algorithm binary search is not possible for implementation. Chromosomes are encoded in decimal format, and a search for the chromosome with the exact fitness might not be possible. Thus, the chromosomes with similar finesses are returned instead. A total of two parent chromosomes are chosen this way in every cycle.

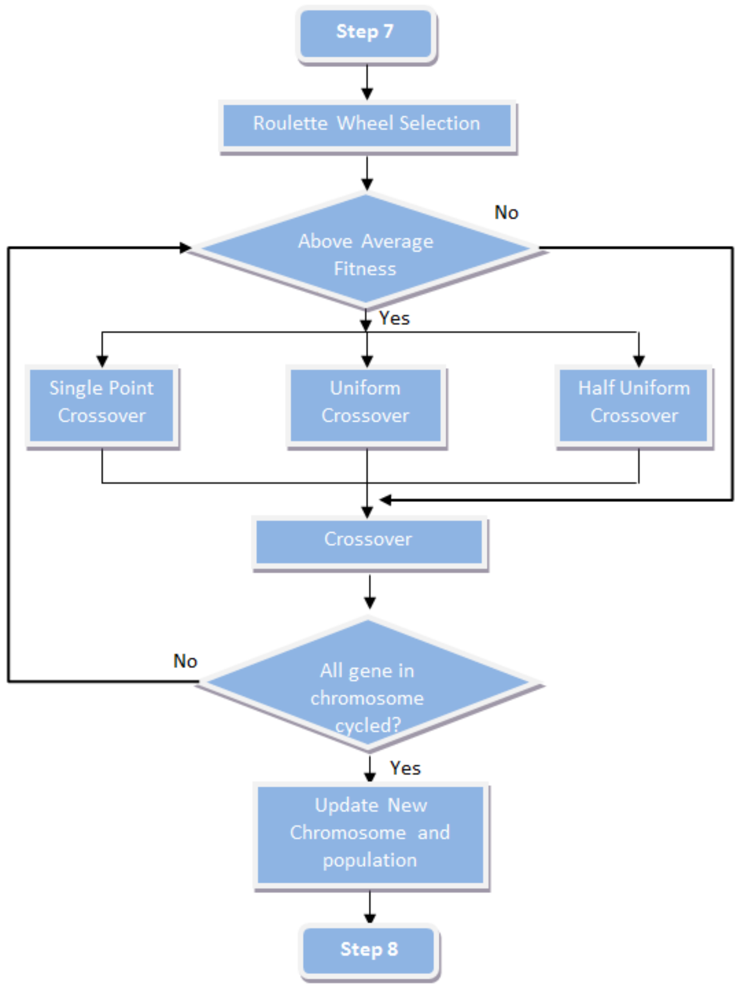


Figure 51 : Flowchart of crossover generation.

Single Point Crossover. A random index within the chromosome is chosen for crossover. All genes prior to this index remain unchanged, while the remaining genes are swapped between the two parents to produce two offspring.

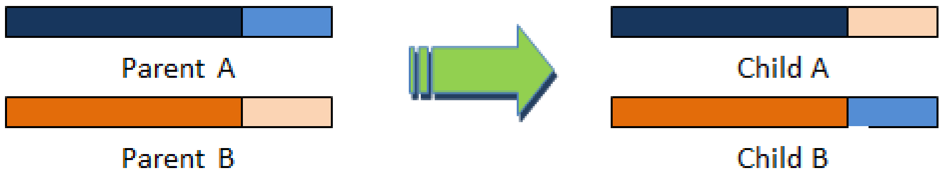


Figure 52 : Single point crossover.

Uniform Crossover. In uniform crossover, the probability of crossover is 50%, and the parents exchange the same amount of genes for the offspring. There can be multiple crossover indexes.

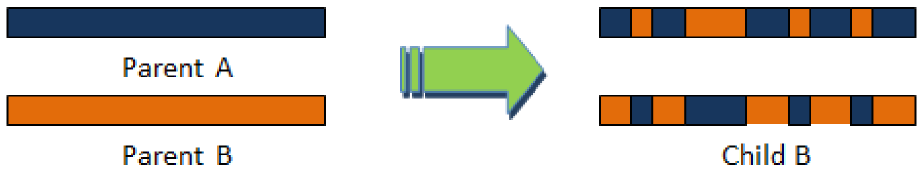


Figure 53 : Uniform crossover.

Half Uniform Crossover. The hamming distance between two parents is first computed to obtain the number of non-matching bits. Then, the total number of genes swapped randomly between the two parents is half of the hamming distance.

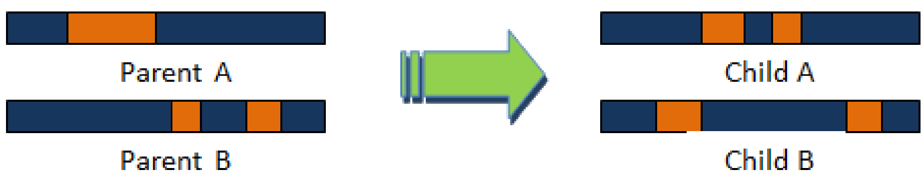


Figure 54 : Half crossover.

Step 8: Only the top order chromosome are selected for mutation. Similar to crossover generation, the parents are selected from the population, however, the mutation is randomly assigned. There is no similarity for the mutated gene and the test population.

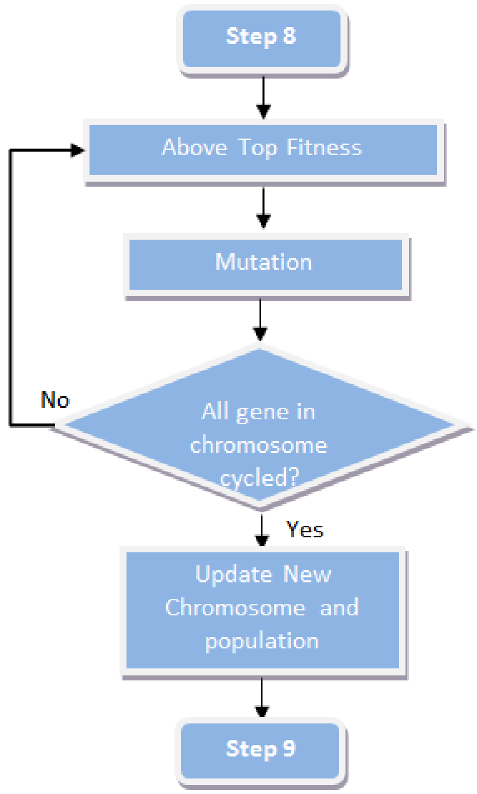


Figure 55 : Flowchart of mutation generation.

Step 9: Termination of GA. Results are displayed while all necessary parameters are automatically saved into a file for reference.

6.11 Assumption and Pre-simulation

In modelling the simulator, there are several assumptions made. First, it is assumed that the locomotion operates at a rate of 50Hz. Such an assumption is made because there are limitations like baud rate and, wireless transmission and processing speeds of the modules. The fastest stable operational speed achieved by Evolve Generation 2 is about 25Hz. Hence, there is no need to increase the rate of simulation as the physical hardware is not being able to execute beyond that rate. It is also difficult to know the exact coefficient of friction between the robot and the test environment. Through experimenting with the simulation environment, the coefficient of friction above 0.7 is ideal for 2 modules connected in series. It gives sufficient static and dynamic frictions for the modules to perform locomotion.

6.12 Result and Analysis

A total of 6 genetic algorithm runs were tested and the simulation took a total of 84 hours of computations. In the simulation, two modules connected in series are tasked to perform a displacement within 2 seconds. All simulations are completed giving successful results. The list of top locomotion generated per simulation is shown in the **Appendix section**.

Simulation 1: In the first simulation, the population started with an average fitness of 0.22 units. The maximum fitness for the initial population is 0.81 units. It is noticed that the algorithm quickly found a possible solution after 20 simulation cycles. A mutation in the 84th cycle, triggered a change in performance.

Parameters	Values
Total Run Cycles	200
Overall Average Fitness	0.86
Best Fitness	1.73
Standard Deviation Average Fitness	0.11
Standard Deviation Best Fitness	0.13

Table 5 : Results for Simulation 1.

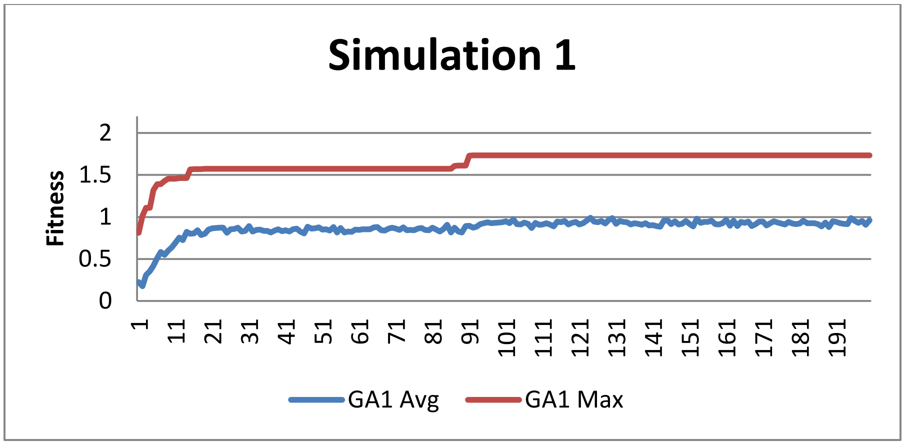


Figure 56 : Fitness graph for Simulation 1.

Figure 57 shows that the simulation has converged to a crawl sequence. The generated crawl is not as efficient as the belly portion modules made contacts with the terrain.

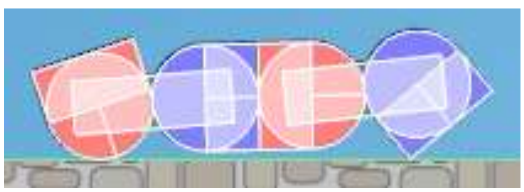


Figure 57 : View of Simulation 1.

Simulation 2: The second simulation was started with a population with an average fitness of 0.17 units. This is also the lowest initial average fitness of all the 6 tests. The maximum fitness for the initial population is noted as 0.94 units. In this simulation, a higher crossover and mutation rates are used causing the simulation result to be stationary for the first 10 cycles. The best solution in this population was found after 38 cycles.

Parameters	Values
Total Run Cycles	200
Overall Average Fitness	0.88
Best Fitness	1.73
Standard Deviation Average Fitness	0.17
Standard Deviation Best Fitness	0.18

Table 6 : Results for Simulation 2.

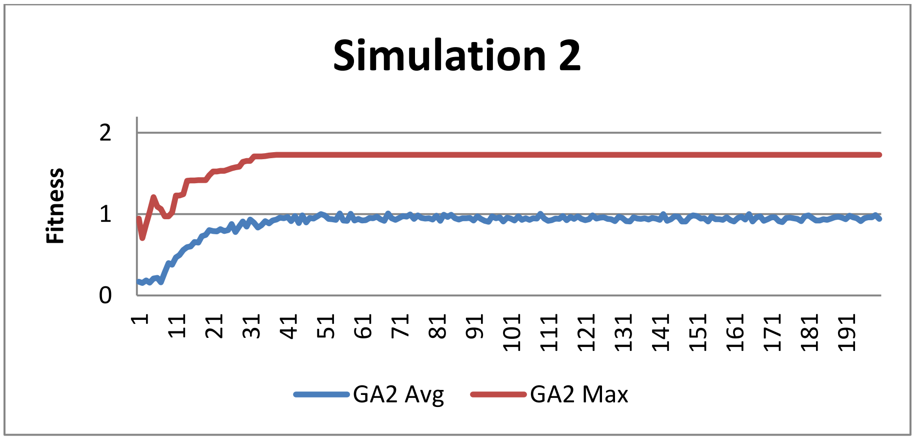


Figure 58: Fitness graph for Simulation 2.

In this simulation, small adjustments were made when the crawl sequence was in action. Unlike the outcome from the first simulation, the bellies of the modules were levitated in the crawl sequence which resulted in better performance. **Figure 59** shows levitating of the modules belly.

Figure 59 : View of Simulation 2.

Simulation 3: The third simulation was started with a population with an average fitness of 0.33 units. The maximum fitness for the initial population is noted as 0.86 units. In this simulation, the 54th and 103rd cycles results a jump in locomotion improvement.

Parameters	Values
Total Run Cycles	200
Overall Average Fitness	0.89
Best Fitness	1.77
Standard Deviation Average Fitness	0.11
Standard Deviation Best Fitness	0.15

Table 7 : Results for Simulation 3.

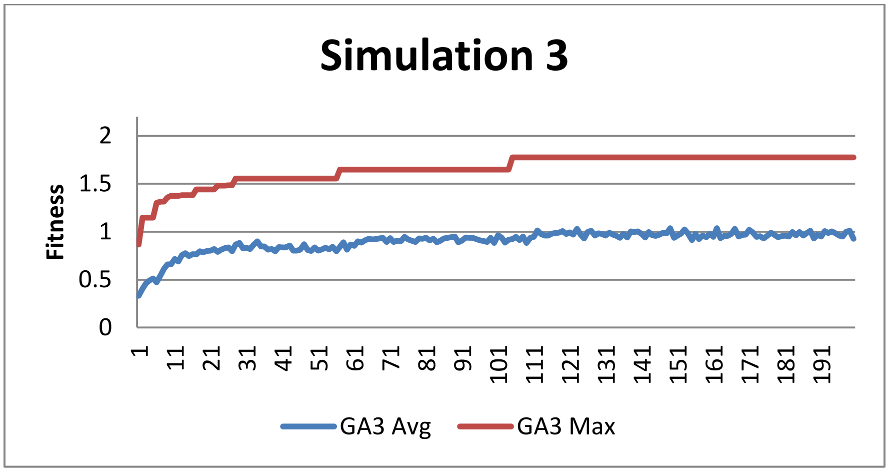


Figure 60: Fitness graph for Simulation 3.

In this simulation, additional angle was added to levitate the belly even higher than Simulation 2. Although a significant angle was input for the actuators there is not significant improve in displacement. **Figure 61** shows that the belly was levitated almost half of the height of the module.

Figure 61 : View of Simulation 3.

Simulation 4: The fourth simulation was started with a population with an average fitness of 0.31 units. The maximum fitness for the initial population is noted as 1.33 units. Unlike those previous populations, this sequence evolved into a sinusoidal wave.

Parameters	Values
Total Run Cycles	200
Overall Average Fitness	0.88
Best Fitness	1.88
Standard Deviation Average Fitness	0.10
Standard Deviation Best Fitness	0.13

Table 8 : Results for Simulation 4.

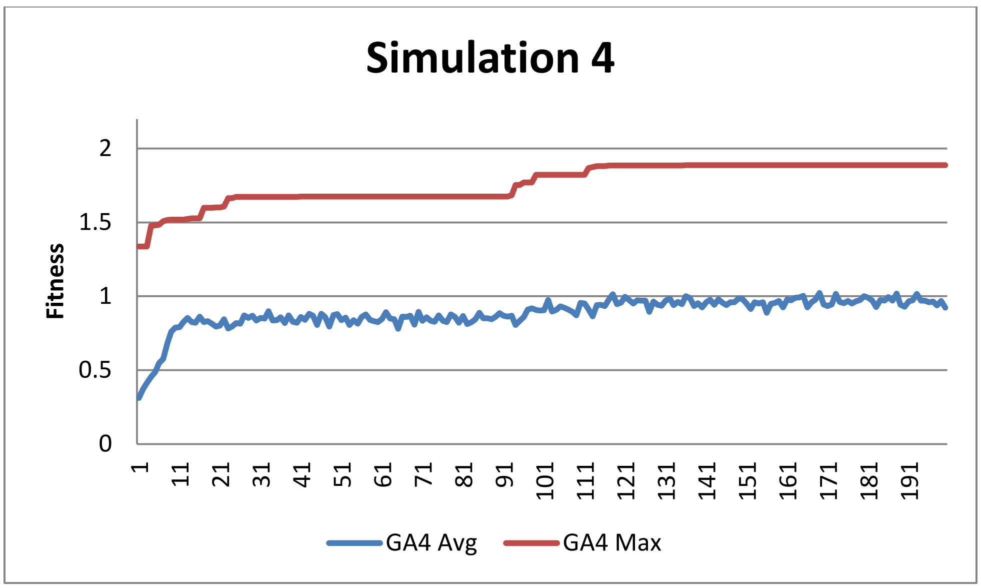


Figure 62 : Fitness graph for Simulation 4.

A sinusoidal wave was generated in this simulation. In comparison to the crawl sequence, a sinusoidal wave is continuous and efficient. **Figure 63** shows the sinusoid like configuration of the modules.

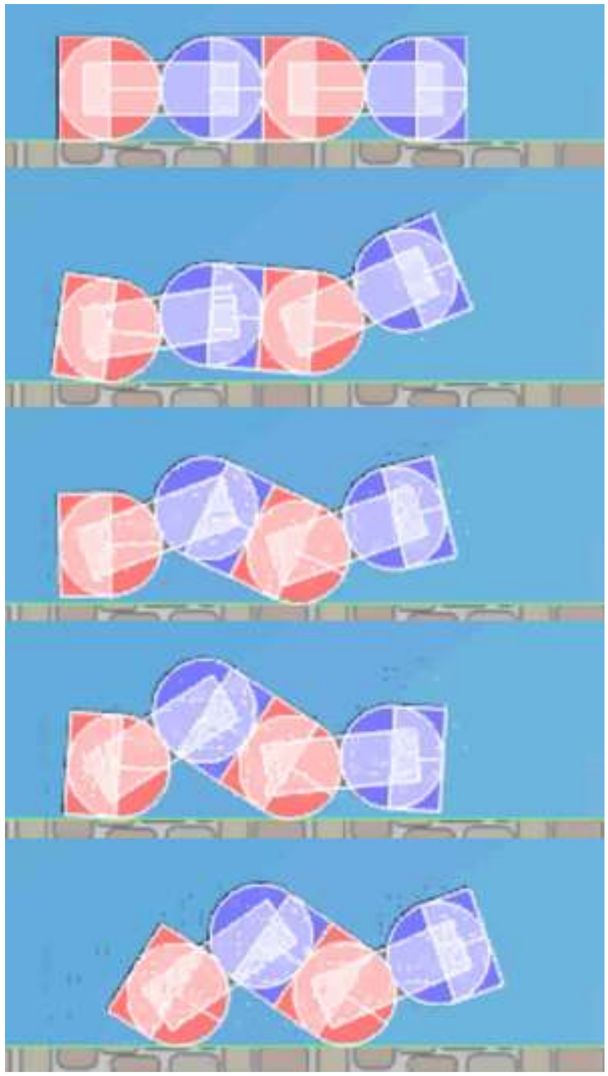


Figure 63 : View of Simulation 4.

Simulation 5: The fifth simulation was started with a population with an average fitness of 0.35 units. It is the highest initial average population for the 6^{th} simulation. The maximum fitness for the initial population is 1.53 units. By taking samples from the previous simulation the genetic algorithm attempts to combine both crawl and sinusoid wave into the locomotion.

Parameters	Values
Total Run Cycles	200
Overall Average Fitness	1.02
Best Fitness	2.13
Standard Deviation Average Fitness	0.15
Standard Deviation Best Fitness	0.12

Table 9 : Results for Simulation 5.

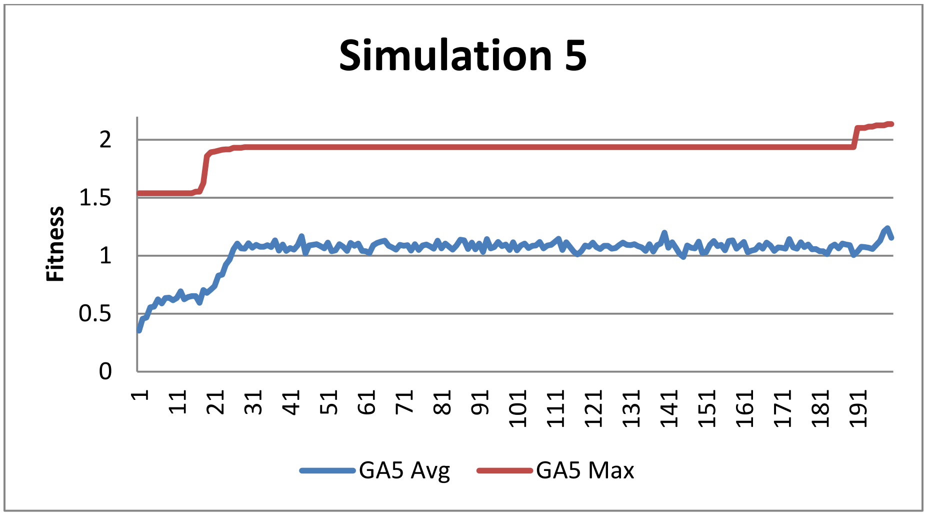


Figure 64 : Fitness graph for Simulation 5.

Shown in **Figure 65**, the centres of the modules are tilted to create a sinusoidal locomotion. This is the best result for all the 6 test simulations. In the first three simulations, frictional force generated at the belly actually lowered the displacement. In this simulation the combination of sinusoid waves created weight shifting like motion on modules.

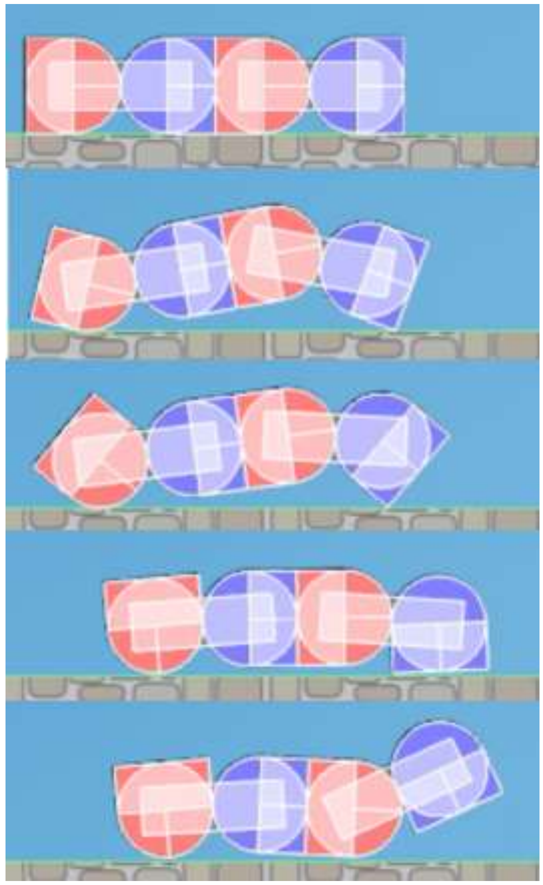


Figure 65 : View of Simulation 5.

Simulation 6: The sixth simulation was started with a population with an average fitness of 0.29 units. The maximum fitness for the initial population is 0.84 units. Although it is an average population, it results in motion with crawl and aggressive sinusoid wave inputs.

Parameters	Values
Total Run Cycles	200
Overall Average Fitness	0.95
Best Fitness	1.80
Standard Deviation Average Fitness	0.10
Standard Deviation Best Fitness	0.10

Table 10 : Results for Simulation 6.

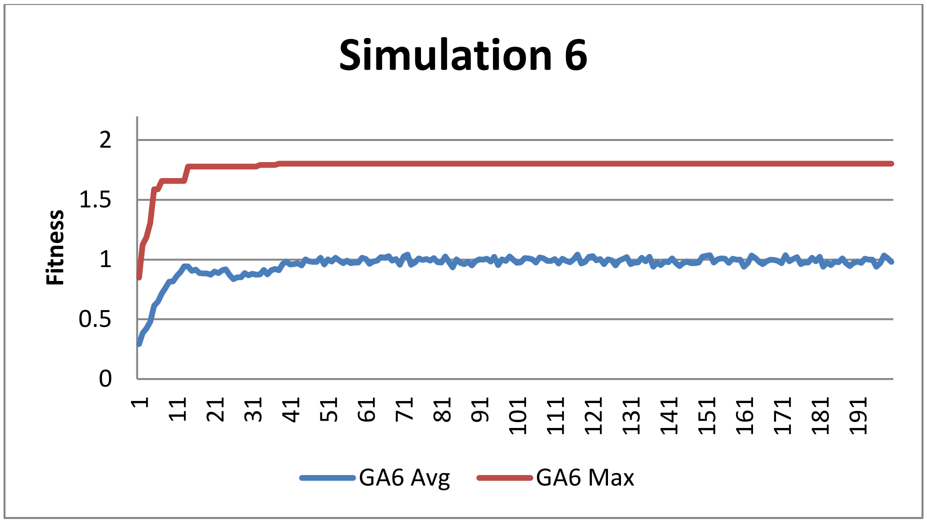


Figure 66 : Fitness graph for Simulation 6.

Shown in **Figure 67**, an aggressive sinusoidal wave is mixed with a crawl sequence. Unlike Simulation 5, the sinusoidal wave did not contribute to better performance as it has created more drag as the belly of the modules made contacts with the terrain.

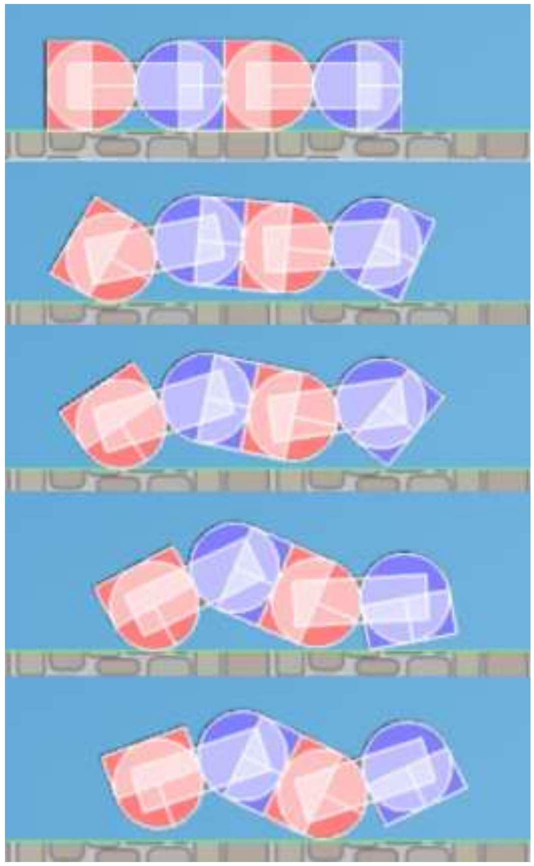


Figure 67 : View of Simulation 6.

6.13 Genetic Algorithm Simulation Result and Summary

The simulations generated 6 reasonable locomotions for a series connected Evolve modules. Although it may seem that an optimal solution maybe crawls or sinusoid sequence, it is unexpected that the Genetic Algorithm found a locomotion which had a combination of the two possible locomotions. The overall best result was a displacement of 2.13 units within 2 seconds. Compared with the hard coding of the crawl sequence made for Evolve Generation 1, the Genetic Algorithm had made a significant improvement. The only drawback was the time taken for simulation, approximately 84 hours was spent to generate test 6 sets of locomotions for Evolve 2. **Figure 68** shows the combined simulation results. Although actuators control can be easily done in real-time, the movement generation can only be assessed in off-line mode.

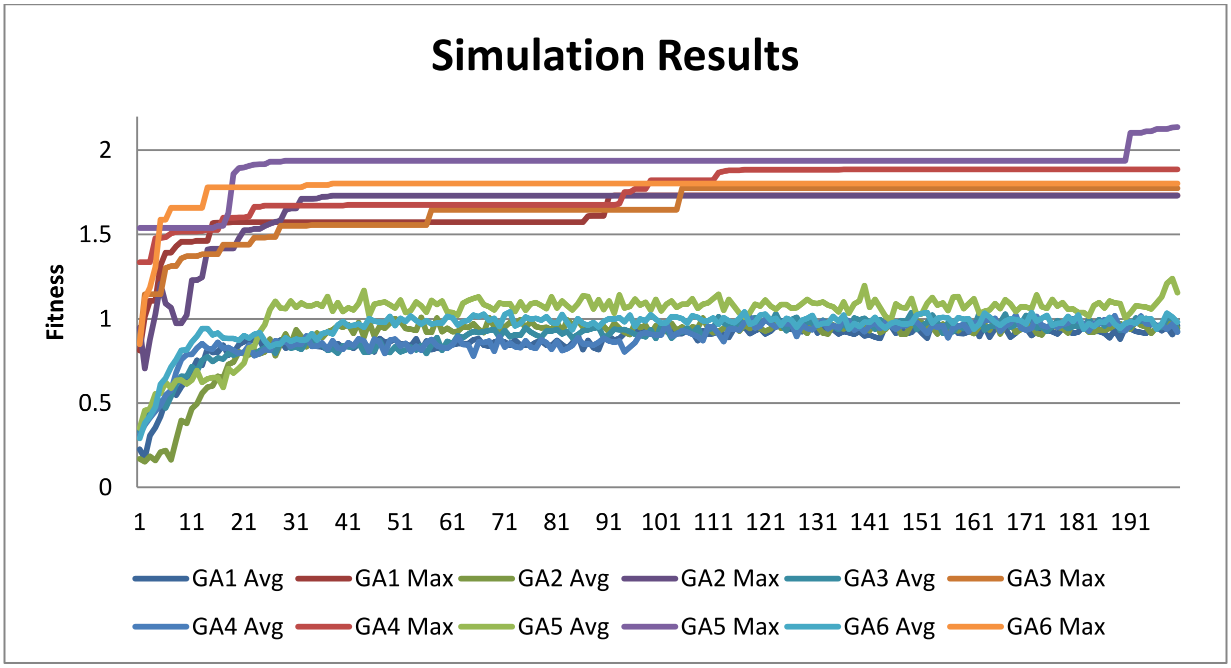


Figure 68 : Summary of simulation results.

6.14 Testing on Evolve Generation 2 Hardware

The generated locomotion from the simulations above is based on a 50 Hz control steps, each actuators will be update with new position and velocity at 20ms interval. A thin layer of silicon is glued to the lower contact surfaces on each module. This allows good frictional grip between the module and the testing surface.

Given the similarity of the design between the two generations of robots, Simulation 4 and simulation 6 is selected for actual testing as the generation locomotion does not look like a conventional crawl sequence that is seen in Evolve Generation 1.

Simulation 4 contains sinusoidal wave like locomotion, **Figure 69** shows the plot for control steps. The measured position gain from the hardware testing shown in **Figure 70** is 1.61 units displacement; this is less than the simulated value of 1.88 units.

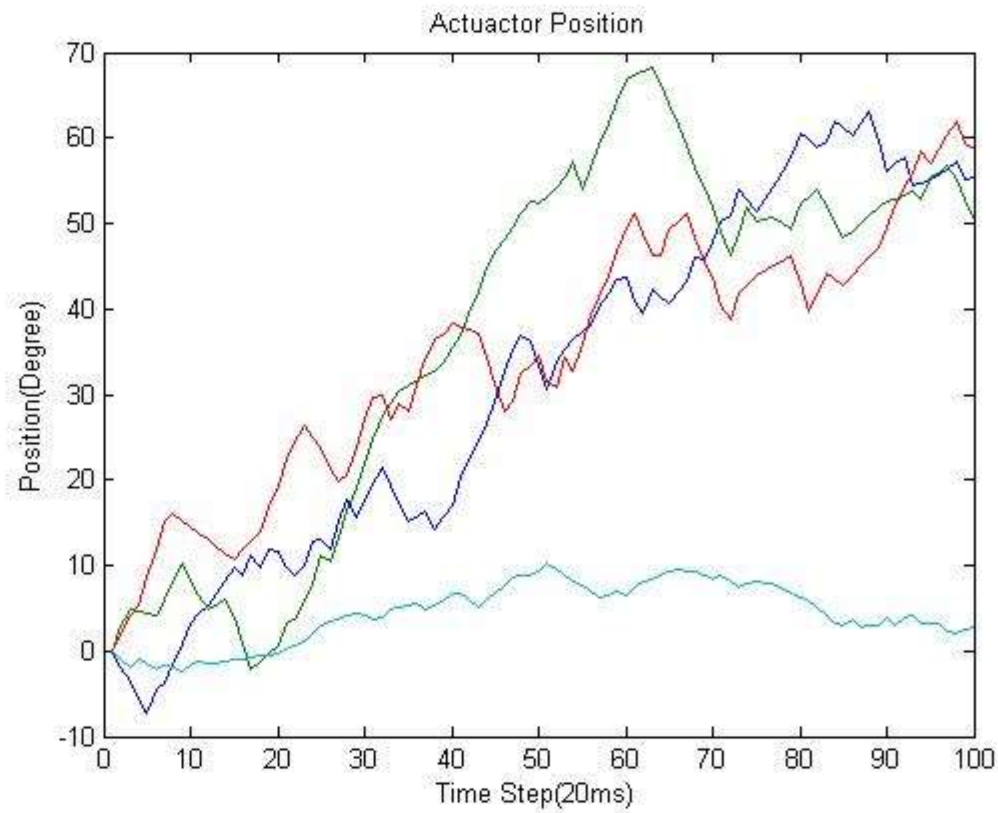


Figure 69 : Actuators Position Plot for Simulation 4.

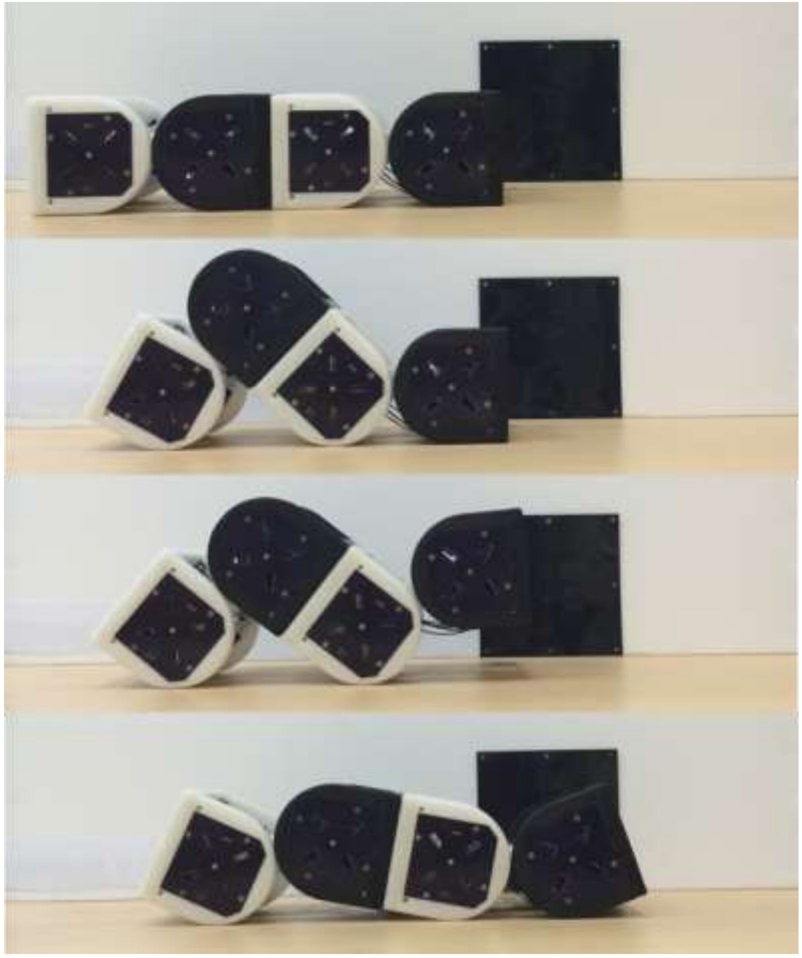


Figure 70 : Using Simulation 4.

Simulation 6 is a combination of crawl and sinusoidal wave like locomotion, **Figure 71** shows the plot for control steps. The measured position gain from the hardware testing shown in **Figure 72** is 1.64 units displacement; this is less than the simulated value of 1.80 units.

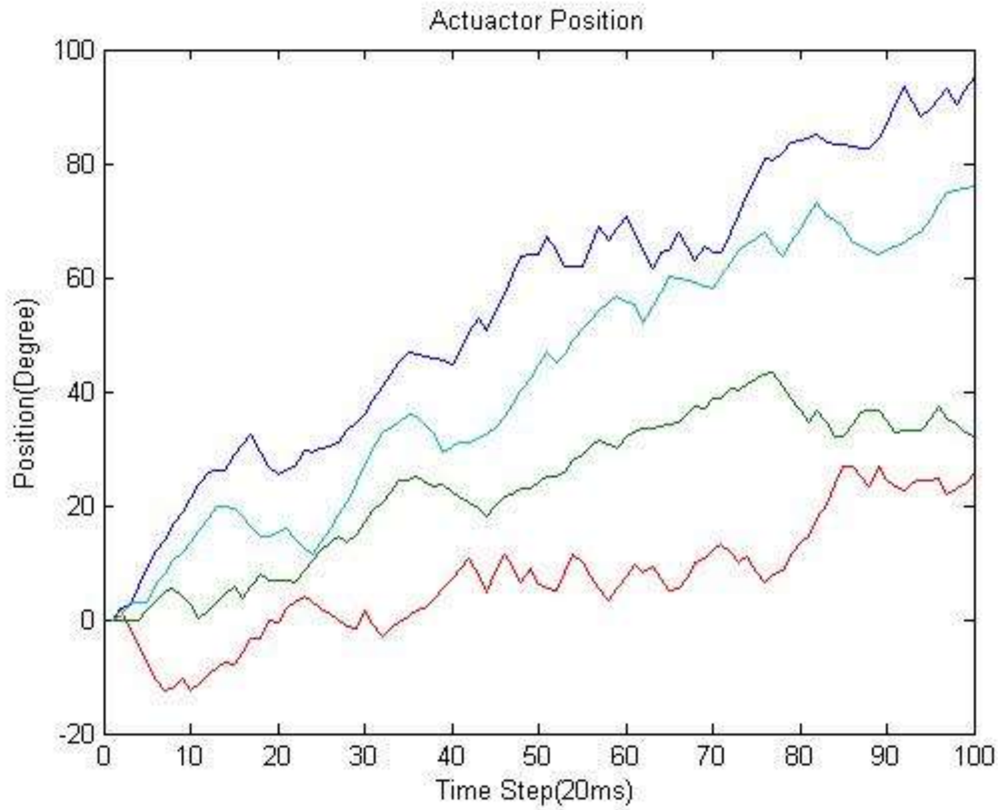


Figure 71 : Actuators Position Plot for Simulation 6.

Figure 72 : Using Simulation 6.

We observed a difference in performance between the simulation and actual hardware experiment. We can attribute this difference in performance is due to frictional coefficient setting and also the modelled centre of gravity may not be true to the actual modules. Although there is small difference between simulation and actual testing, the overall testing shows that the Genetic Algorithm is able to generate a relative good forward locomotion without the need for any tuning and developers inputs.

Chapter 7 Conclusions

7.1 Conclusions and Future Work

In this work, we have developed the second generation of Evolve system that is capable of self-reconfiguration. It is equipped with a controllable docking surface and a roll actuator. In comparison with Evolve Generation 1, the second generation Evolve has controllable docking surfaces and an additional actuator. The docking surface allows connections in 4 orientations. The additional roll actuator increased each module's degrees of freedom to three. The lightweight chassis and careful selection of components allowed the new module to weigh less than the previous model but with a significant increase in capability. With the above improvement Evolve Generation 2 can be classified as self-reconfigurable modular system.

The electronics within the module has improved accuracy and is able to perform computations for real-time locomotion calculation. The development of communication protocol and data flow architecture allows the modules to communicate and share data efficiently between two generations of the modules.

Instead of using deterministic locomotion, the stochastic search model of the Genetic algorithm Genetic algorithm has found an efficient basic locomotion that combination of crawl and sinusoid wave manoeuvres. It has achieved simulated best 2.13 units of displacement within a 2 second run time. The actual testing confirms that the simulation is similar to real world model. Evolve Generation 2 is not just equipped with better hardware; but it is also capable of solving basic locomotion through its build in simulator.

Going forward, there is a need to explore other configurations of Evolve Generation 2 which is not simulated or testing in this work (i.e. L-configuration). Due to the limitation of a 2D simulated the roll actuator is disabled, therefore there are other possible locomotion that are not explored. On the hardware, the existing module uses lithium polymer batteries which only power the modules of approximately 20 minutes; hence energy efficient is a major concern. In the next development of the Genetic Algorithm there is a need to combine energy efficient as part of the fitness function.

References

[1] Yim, Mark, et al. "Modular reconfigurable robots in space applications." Autonomous Robots 14.2-3 (2003): 225-237.

[2] T. Fukuda and S. Nakagawa, “Dynamically reconfigurable robotic system,” IEEE International Conference on Robotics and Automation, vol. 3, pp. 1581-1586, 1988.

[3] Murata, Satoshi, et al. "M-TRAN: Self-reconfigurable modular robotic system." Mechatronics, IEEE/ASME Transactions on 7.4 (2002): 431-441.

[4] Yim M, Shen W-M, Salemi B, Rus D, Moll M, Lipson H, Klavins E, Chirikjian GS. (2007). “Modular Self-Reconfigurable Robot Systems: Grand Challenges of Robotics.” IEEE Robot Autom Mag 14(1):43–52

[5] Rus, Daniela, and Marsette Vona. "Self-reconfiguration planning with compressible unit modules." Robotics and Automation, 1999. Proceedings. 1999 IEEE International Conference on. Vol. 4. IEEE, 1999.

[6] Rus D, Vona M. Crystalline robots. (2001). Self-reconfiguration with unit compressible modules. Autonomous Robots, 10(1): 107-124.

[7] Sastra, Jimmy, Sachin Chitta, and Mark Yim. "Dynamic rolling for a modular loop robot." The International Journal of Robotics Research 28.6 (2009): 758-773.

[8] Yim, Mark, David G. Duff, and Kimon D. Roufas. "PolyBot: a modular reconfigurable robot." Robotics and Automation, 2000. Proceedings. ICRA'00. IEEE International Conference on. Vol. 1. IEEE, 2000.

[9] Dorigo, Marco, et al. "Swarmanoid." IEEE Robotics & Automation Magazine 1070.9932/13 (2013).

[10] White, P. J., Kris Kopanski, and Hod Lipson. "Stochastic self-reconfigurable cellular robotics." Robotics and Automation, 2004. Proceedings. ICRA'04. 2004 IEEE International Conference on. Vol. 3. IEEE, 2004.

[11] Möckel, Rico, et al. "YaMoR and Bluemove—an autonomous modular robot with bluetooth interface for exploring adaptive locomotion." Climbing and Walking Robots. Springer Berlin Heidelberg, 2006. 685-692.

[12] Jantapremjit, Pakpong, and David Austin. "Design of a modular self-reconfigurable robot." Australian Conf. on Robotics and Automation, Sydney, Australia. 2001.

[13] Duff, David, Mark Yim, and Kimon Roufas. "Evolution of polybot: A modular reconfigurable robot." Proc. of the Harmonic Drive Intl. Symposium, Nagano, Japan. 2001.

[14] Yim, Mark, et al. "Connecting and disconnecting for chain self-reconfiguration with PolyBot." Mechatronics, IEEE/ASME Transactions on 7.4 (2002): 442-451.

[15] Yim, Mark, Ying Zhang, and David Duff. "Modular robots." Spectrum, IEEE 39.2 (2002): 30-34.

[16] Salemi, Behnam, Mark Moll, and Wei-Min Shen. "SUPERBOT: A deployable, multi-functional, and modular self-reconfigurable robotic system." Intelligent Robots and Systems, 2006 IEEE/RSJ International Conference on. IEEE, 2006.

[17] Shen, Wei-Min, et al. "Multimode locomotion via SuperBot reconfigurable robots." Autonomous Robots 20.2 (2006): 165-177.

[18] Kamimura, Akiya, et al. "A Self-Reconfigurable Modular Robot (MTRAN)—Hardware and Motion Planning Software." Distributed autonomous robotic systems 5. Springer Japan, 2002. 17-26.

[19] Shen, Wei-Min, Behnam Salemi, and Peter Will. "Hormone-inspired adaptive communication and distributed control for CONRO self-reconfigurable robots." Robotics and Automation, IEEE Transactions on 18.5 (2002): 700-712.

[20] Tomita, Kohji, et al. "Self-assembly and self-repair method for a distributed mechanical system." Robotics and Automation, IEEE Transactions on 15.6 (1999): 1035-1045.

[21] Kotay, Keith D., and Daniela L. Rus. "Motion synthesis for the self-reconfiguring molecule." Intelligent Robots and Systems, 1998. Proceedings., 1998 IEEE/RSJ International Conference on. Vol. 2. IEEE, 1998.

[22] Ünsal, Cem, Han Kiliççöte, and Pradeep K. Khosla. "A modular self-reconfigurable bipartite robotic system: Implementation and motion planning." Autonomous Robots 10.1 (2001): 23-40.

[23] Butler, Zack, et al. "Generic decentralized control for a class of self-reconfigurable robots." Robotics and Automation, 2002. Proceedings. ICRA'02. IEEE International Conference on. Vol. 1. IEEE, 2002.

[24] Prevas, Konstantine C., et al. "A hierarchical motion planning strategy for a uniform self-reconfigurable modular robotic system." Robotics and Automation, 2002. Proceedings. ICRA'02. IEEE International Conference on. Vol. 1. IEEE, 2002.

[25] Kamimura, Akiya, et al. "Automatic locomotion pattern generation for modular robots." Robotics and Automation, 2003. Proceedings. ICRA'03. IEEE International Conference on. Vol. 1. IEEE, 2003.

[26] Chen, S., and S. A. Billings. "Neural networks for nonlinear dynamic system modelling and identification." International journal of control 56.2 (1992): 319-346.

[27] Ashbaugh, Ben, and Justin Boitano. "Advantages and Disadvantages of Controller Designs Using Fuzzy Logic."

[28] Dumitrescu, Dumitru, et al. Evolutionary computation. Vol. 18. CRC press, 2000.

[29] Arakawa, Takemasa, and Toshio Fukuda. "Natural motion trajectory generation of biped locomotion robot using genetic algorithm through energy optimization." Systems, Man, and Cybernetics, 1996., IEEE International Conference on. Vol. 2. IEEE, 1996.

[30] Parker, G., and G. Rawlins. "Cyclic genetic algorithms for the locomotion of hexapod robots." Proceedings of the World Automation Congress (WAC'96). Vol. 3. 1996.

[31] Kodjabachian, Jérôme, and J-A. Meyer. "Evolution and development of neural controllers for locomotion, gradient-following, and obstacle-avoidance in artificial insects." Neural Networks, IEEE Transactions on 9.5 (1998): 796-812.

[32] Holland, John H. Adaptation in natural and artificial systems: An introductory analysis with applications to biology, control, and artificial intelligence. U Michigan Press, 1975.

Appendix

Simulation 1

	SERVO L	SERVO LC	SERVO RC	SERVO R
1	-0.01676	0.009401	-0.00521	0.005775
2	0.021279	0.009401	-0.00744	0.005775
3	0.033678	0.009401	0.003251	-0.02575
4	0.015255	0.009401	0.003251	0.028564
5	0.015255	0.009401	0.006363	0.028564
6	-0.00451	0.000656	-0.0091	0.028564
7	-0.01802	0.000656	-0.01125	-0.03394
8	-0.01802	0.002439	-0.01125	-0.03774
9	-0.00013	0.002439	0.017683	0.003203
10	0.04441	0.002439	0.001132	0.053287
11	0.04441	0.009838	0.001132	0.046362
12	0.00392	0.011827	0.02262	0.046362
13	0.00392	0.007332	0.003454	0.046362
14	0.025962	0.002656	0.003454	0.023457
15	0.04476	0.002656	-0.0168	0.048483
16	0.020543	0.002656	0.007233	0.039633
17	0.020543	0.002656	0.000123	-0.00315
18	0.020543	0.002656	0.00926	-0.00315
19	0.020543	0.01201	-0.01002	0.057127
20	0.005073	-0.00868	0.000156	0.018288
21	0.005073	-0.00868	0.000156	0.018288
22	0.005073	-0.00868	0.028289	0.054843
23	0.005073	-0.00868	0.025444	-0.05428
24	0.045402	0.000744	-0.02271	-0.02504
25	0.016785	-0.00473	0.012744	0.054673
26	-0.01453	-0.00773	0.012744	-0.04214
27	-0.03355	0.002143	0.015063	0.032712
28	0.029758	0.000361	0.006384	0.05548
29	0.029758	0.002113	0.006384	0.05548
30	0.008091	-0.00181	0.006384	0.05548
31	0.028315	-0.00019	0.016292	0.023782
32	0.028315	-0.01115	0.006564	0.030692
33	0.028315	-0.00396	0.011131	0.030692
34	0.028315	-0.00396	0.006754	0.030692

35	0.028315	-0.00396	-0.00649	-0.035
36	0.028315	-0.00479	-0.00649	-0.035
37	0.050461	-0.00191	-0.02306	-0.035
38	0.002582	-0.00191	0.005871	-0.035
39	0.036918	0.004954	-0.00379	-0.0546
40	0.036918	0.003526	-0.00379	-0.04451
41	0.019925	0.003526	-0.02439	-0.04451
42	0.031845	0.003526	-0.00952	-0.04451
43	0.031845	-0.00663	-0.00952	0.058552
44	0.031845	-0.00663	0.012993	-0.03343
45	0.031845	0.007462	0.012993	0.028175
46	-0.02552	-0.00442	0.00818	0.031061
47	0.043478	-0.00442	0.009142	0.031061
48	0.023345	-0.01192	0.009142	0.018774
49	0.024041	-0.00086	0.008951	0.035848
50	0.024041	0.005111	-0.02337	0.053149
51	0.015253	0.010817	-0.02337	0.053149
52	0.050082	-0.00709	-0.00951	0.030294
53	0.050082	0.006095	-0.00951	0.043475
54	-0.03393	0.005042	-0.00951	0.043475
55	0.025971	0.010737	-0.00951	0.043475
56	0.025971	-0.00521	-0.02065	0.043475
57	-0.04564	-0.00471	-0.02554	0.029338
58	0.032615	-0.006	-0.01807	0.043619
59	-0.01123	-0.00797	0.010632	0.043619
60	0.005646	0.000211	0.007492	0.003976
61	-0.01843	0.00594	-0.02503	0.001571
62	0.006983	0.005139	0.014535	0.019857
63	0.006983	-0.00426	-0.01787	-0.04333
64	-0.01361	0.009725	0.008085	0.010387
65	0.006257	-0.01153	0.008085	0.008984
66	0.006257	-0.01153	-0.01881	0.025767
67	0.035421	-0.00959	0.003644	0.034615
68	-0.03954	0.005796	-0.00699	0.026454
69	-0.04574	0.005796	-0.00699	0.054391
70	0.033271	0.005796	0.016788	0.054391
71	0.02915	0.011274	-0.02823	0.006769
72	0.035231	-0.01142	0.002982	-0.01527
73	0.002315	-0.01142	0.027346	0.001601
74	0.032911	0.00285	0.002357	-0.05563
75	0.009607	0.00285	0.002357	-0.05563

76	0.009607	0.001586	-0.01889	0.010329
77	0.009607	0.004833	-0.01889	-0.0338
78	0.009607	0.004833	-0.01889	-0.0338
79	0.009607	0.004833	-0.01889	-0.0338
80	-0.04908	-0.00849	-0.01889	-0.0338
81	-0.04565	-0.00849	-0.01889	-0.01774
82	-0.04565	-0.00849	0.013765	0.05427
83	-0.02285	-0.00849	0.013765	0.017165
84	-0.01312	-0.00264	-0.02548	0.017165
85	-0.01312	-0.0107	0.013441	0.017165
86	-0.01312	0.00597	0.013441	0.05435
87	-0.01312	0.010958	-0.01059	0.05435
88	-0.01397	-0.0108	-0.00904	-0.04173
89	-0.04623	0.001581	-0.02691	0.004365
90	-0.04623	0.001581	-0.02691	0.004365
91	0.023102	0.00018	-0.02691	0.004365
92	0.039909	0.00476	0.018376	0.059149
93	-0.03265	0.007383	-0.00582	0.052203
94	0.035351	0.011209	0.010075	0.007188
95	-0.04089	0.011209	0.010075	0.007188
96	0.001375	0.011209	-0.01752	0.007188
97	0.039987	0.011209	-0.01752	-0.05707
98	0.039987	0.00226	0.019865	-0.0105
99	-0.03649	0.00226	0.019865	0.026441
100	-0.03154	-0.00827	0.00084	0.011011

Simulation 2

	SERVO L	SERVO LC	SERVO RC	SERVO R
1	-0.04875	0.01478	0.011469	-0.05141
2	0.024718	0.01478	-0.02069	0.048176
3	-0.00847	-0.01387	-0.01657	0.04591
4	0.041057	-0.01095	-0.01657	0.014677
5	0.058593	-0.00132	-0.01657	0.014677
6	-0.02065	0.012996	0.017267	0.033672
7	-0.02065	-0.00988	0.005879	-0.03112
8	-0.02065	-0.00607	-0.00395	-0.03112
9	-0.04942	0.00982	-0.00395	-0.03112
10	0.056767	0.007029	-0.01791	-0.03112
11	0.056767	-0.01093	-0.01791	0.036668

12	0.023253	-0.01093	0.015796	0.035736
13	0.023253	-0.01093	-0.02416	-0.04697
14	0.015955	0.006883	-0.02416	0.042047
15	0.015955	0.015614	0.01657	-0.04003
16	0.040745	0.013983	0.01657	0.023109
17	0.040745	0.010153	0.01657	0.023109
18	0.048631	0.007617	0.011524	0.049023
19	0.040781	0.014493	0.011524	0.025937
20	0.009125	0.014493	0.011524	0.035001
21	0.009125	-0.00785	0.009686	0.027749
22	-0.00194	-0.01393	0.013099	0.027749
23	-0.01471	0.001813	0.017326	-0.015
24	0.013131	0.000735	-0.01983	0.036848
25	0.022875	0.000735	-0.01983	0.036848
26	0.022875	-0.00196	-0.01983	0.033912
27	0.022875	-0.00354	0.022309	0.033912
28	0.047835	-0.00354	0.022309	0.033912
29	0.047835	-0.00354	-0.02666	0.051406
30	0.036056	0.013209	0.022445	0.042199
31	0.036056	0.006105	0.000697	-0.04971
32	0.036056	0.009807	0.000697	0.049369
33	0.056144	0.014279	0.006345	0.049369
34	0.006964	0.014279	-0.0028	0.032187
35	0.027073	0.011677	0.019381	0.032187
36	0.027073	0.0124	0.016994	-0.00269
37	0.027073	-0.01113	0.016994	-0.00049
38	0.027073	0.006506	0.016994	-0.00049
39	0.027073	0.010417	-0.00394	-0.00049
40	0.027073	0.010654	-0.00394	0.051732
41	0.027073	-0.00867	-0.02687	0.013432
42	0.03372	-0.00867	-0.01343	0.055278
43	0.03372	-0.00867	0.00959	0.04145
44	0.03372	0.0103	0.00959	0.04145
45	0.055098	0.0103	0.00171	0.049103
46	0.055098	0.0103	0.00171	0.014593
47	0.055098	0.0103	-0.01916	-0.01854
48	0.055098	0.01446	-0.01916	-0.01854
49	0.055098	0.01446	0.005675	0.004596
50	0.055098	0.009632	0.014523	-0.04347
51	0.055098	0.011566	-0.00565	0.031601
52	0.055098	-0.01568	-0.00565	-0.05164

53	0.055098	0.008236	-0.00565	0.034411
54	0.055098	0.00211	0.021025	-0.05092
55	0.055098	0.015674	0.021025	-0.05092
56	0.055098	0.002555	0.026348	0.023919
57	0.055098	0.015366	0.019518	-0.04673
58	-0.00262	-0.00264	0.019518	-0.0129
59	-0.00262	-0.00264	0.019518	-0.00951
60	-0.00262	0.015579	-0.00163	0.023035
61	0.013546	-0.00433	-0.00134	0.038011
62	0.024669	0.013504	0.018233	0.055527
63	0.024669	0.010555	0.027433	0.042751
64	-0.00091	-0.0124	0.027433	0.036094
65	-0.00091	0.001022	0.027433	0.036094
66	-0.05373	0.001022	0.011447	0.036094
67	0.054152	-0.00842	0.02259	0.045874
68	0.054152	0.000327	0.010812	0.018743
69	-0.05155	0.000327	0.021713	0.018743
70	-0.05155	0.010748	0.0026	0.041804
71	-0.05155	0.007911	-0.02659	0.029299
72	-0.05155	0.012827	-0.02762	0.009309
73	0.028712	0.012845	-0.01338	0.009309
74	0.028712	-0.00453	-0.01058	0.043
75	0.02132	0.014554	-0.01058	-0.02116
76	0.048775	-0.00106	-0.01058	-0.03999
77	0.048775	-0.00106	-0.01058	-0.03999
78	0.041225	0.01408	0.018553	0.009284
79	-0.03373	0.006163	-0.00164	0.008169
80	-0.02723	0.006163	-0.0119	0.046951
81	-0.0333	-0.00454	0.011457	0.04028
82	-0.04049	-0.00454	0.022463	0.030797
83	-0.04049	0.012404	0.016024	-0.03735
84	-0.04049	0.012633	0.016024	0.002477
85	-0.04049	0.012633	-0.00271	0.011558
86	-0.01942	0.00991	-0.00271	0.011558
87	0.033758	-0.00176	0.024335	0.011558
88	0.033758	-0.00176	0.001667	0.030281
89	0.003647	-0.01106	0.00234	0.030281
90	0.00828	-0.00653	0.00234	0.033351
91	-0.01604	-0.00119	0.00234	0.018546
92	-0.01604	0.006681	0.00234	0.019281
93	-0.01783	0.006681	0.00234	0.019281

94	-0.00135	-0.01219	0.00234	0.029379
95	-0.00135	-0.00829	0.00234	0.029379
96	-0.00135	-0.00829	-0.02151	0.05488
97	-0.00135	0.010538	-0.02151	0.043669
98	-0.00135	0.01325	-0.02151	-0.05052
99	-0.00135	-0.00833	0.021952	0.041024
100	-0.00135	-0.01356	0.021952	-0.01666

Simulation 3

	SERVO L	SERVO LC	SERVO RC	SERVO R
1	0.057108	0.002071	-0.00161	0
2	0.057108	0.045684	-0.00161	0.024708
3	0.048183	0.041716	-0.02583	0.024708
4	0.047981	0.041716	0.017371	0.043305
5	0.047981	0.026707	-0.02266	0.043305
6	0.047981	-0.02454	-0.03709	0.036943
7	0.042386	-0.02454	0.006817	0.036943
8	-0.02568	0.05609	-0.00531	0.00947
9	-0.02568	0.041103	-0.04137	0.042394
10	-0.02568	0.025901	-0.02482	-0.03315
11	0.027683	0.025901	0.0004	-9.74E-05
12	0.027683	0.009426	0.004328	-9.74E-05
13	-0.0219	0.009426	0.004328	-9.74E-05
14	-0.0219	-0.03862	0.004328	-9.74E-05
15	-0.0219	-0.00413	0.004328	-9.74E-05
16	-0.0219	-0.00125	0.004328	0.010327
17	0.040885	-0.00125	0.004328	0.018464
18	0.040885	-0.00125	0.004328	-0.04508
19	-0.04805	0.014626	0.004328	0.021359
20	0.055632	0.014626	0.031356	0.029719
21	0.013938	0.014626	0.009401	0.029719
22	0.030996	0.014626	0.036611	0.047217
23	0.043607	0.005749	0.036611	-0.0292
24	0.043607	0.005749	0.036243	0.021982
25	0.043607	0.005749	-0.00644	0.011459
26	0.00332	0.033173	-0.00644	0.013687
27	0.00332	-0.03626	-0.00644	0.048297
28	0.012752	-0.05578	-0.00644	0.028975
29	0.041287	-0.02042	-0.024	0.015259

30	0.041287	0.022244	-0.045	0.015259
31	0.041287	-0.02763	-0.02907	-0.0425
32	0.020844	-0.00743	0.016297	-0.03231
33	0.020844	0.013476	0.044648	0.038691
34	0.020844	0.013476	-0.02013	0.005489
35	0.020844	0.021987	-0.02013	0.005489
36	0.031115	-0.0496	0.003893	0.009225
37	0.031115	0.01578	0.003893	0.009225
38	-0.03243	-0.00838	-0.02661	0.045182
39	-0.0393	-0.00838	0.00167	0.045182
40	-0.0393	-0.00838	0.00167	0.045182
41	0.05011	-0.00032	0.001826	0.03976
42	0.05011	0.036254	0.001826	0.03976
43	0.01146	0.021947	0.007621	0.038686
44	0.038709	-0.02191	-0.02535	-0.02157
45	0.038709	-0.02191	-0.02376	-0.04896
46	-0.05484	-0.02191	-0.02376	0.04751
47	0.015062	-0.02191	0.015651	0.04751
48	-0.01205	0.029732	0.001968	0.04751
49	-0.01205	0.04793	0.045912	0.050543
50	-0.01205	0.04793	-0.03233	0.050543
51	0.020982	-0.01938	0.021541	0.050543
52	0.029737	-0.03552	0.03463	0.050543
53	0.031232	-0.03552	0.020117	0.012587
54	0.010204	-0.03552	-0.04333	0.012587
55	0.010204	0.047699	-0.04333	0.047087
56	-0.00981	0.047699	0.04248	0.009351
57	-0.00981	0.047699	0.004824	-0.03592
58	-0.02493	0.022689	0.004824	0.052577
59	-0.02493	0.022689	-0.04264	0.022329
60	-0.01906	-0.00261	-0.04264	0.040692
61	0.05099	-0.00261	-0.04264	0.05133
62	-0.00081	-0.03422	-0.04264	0.015473
63	-0.00081	0.048426	0.028907	0.015473
64	-0.00081	0.048426	0.017373	0.015473
65	0.052998	0.048426	0.017373	0.047117
66	-0.0469	0.03307	0.01569	0.047117
67	-0.02207	-0.03694	0.012178	0.047117
68	0.013298	0.015817	0.019184	0.054967
69	0.015366	-0.03087	0.033662	0.054967
70	0.042864	-0.03686	0.010498	0.054967

71	0.005289	-0.03686	-0.03856	0.016074
72	0.005289	0.05251	-0.03856	-0.04117
73	0.005289	-0.04247	-0.03856	0.005348
74	0.010992	-0.02325	0.009797	0.005348
75	0.034962	-0.02325	-0.006	0.052592
76	-0.03975	0.012759	-0.04141	0.052592
77	-0.03975	-0.01289	-0.00105	0.013114
78	0.053057	-0.01289	0.042878	0.035669
79	0.053057	0.031632	0.042878	0.035669
80	0.053057	0.03265	0.036326	0.022186
81	-0.05838	-0.04436	0.02679	-0.05674
82	0.054558	0.022375	0.02679	-0.05674
83	-0.06014	0.041375	-0.02821	0.028989
84	0.015964	-0.01674	-0.02821	0.028989
85	-0.00813	0.013941	-0.0098	0.000129
86	0.055918	0.013941	0.014082	0.000129
87	0.055918	0.05311	0.011177	0.000129
88	0.049808	0.05311	0.005988	0.036573
89	0.049808	0.05311	0.005988	0.036573
90	0.049808	-0.03777	0.005988	0.036573
91	0.046698	0.032282	0.025904	0.056104
92	0.046698	0.009841	0.025904	0.036471
93	0.017861	-0.00478	0.025904	0.020992
94	-0.05077	0.005229	-0.03558	0.020992
95	-0.03217	-0.01632	0.02129	0.020992
96	0.035621	0.023184	0.02129	-0.01939
97	-0.03558	0.020096	0.02129	-0.01371
98	0.011855	0.027643	-0.00829	0.005241
99	0.018998	0.053546	0.036116	0.005241
100	0.042552	0.053546	0.036116	-0.05464

Simulation 4

	SERVO L	SERVO LC	SERVO RC	SERVO R
1	0.020803	0.029003	0.031154	-0.01661
2	-0.0321	0.053283	0.037028	-0.01661
3	-0.0321	0.031994	0.037028	-0.01654
4	-0.0321	-0.00465	0.019758	0.01694
5	-0.0321	-0.00465	0.057354	-0.01133
6	0.050498	-0.00465	0.057354	-0.01133

7	0.010715	0.032709	0.057354	0.011059
8	0.041114	0.036709	0.01579	-0.00718
9	0.041114	0.036709	-0.01691	-0.00718
10	0.041114	-0.03159	-0.01305	0.013904
11	0.018779	-0.03159	-0.01305	0.004869
12	0.018779	-0.03159	-0.01305	-0.00038
13	0.024989	0.01199	-0.01305	-0.00038
14	0.024989	0.01199	-0.01305	0.002777
15	0.024989	-0.03863	-0.01305	0.002992
16	-0.01404	-0.05368	0.018913	0.002992
17	0.039384	-0.05368	0.020137	0.00415
18	-0.02312	0.016536	0.020137	0.001826
19	0.036894	0.016536	0.045893	0.000167
20	-0.00434	0.016536	0.045893	0.005294
21	-0.03367	0.047626	0.05338	0.008276
22	-0.0156	0.006807	0.038866	0.00807
23	0.021985	0.036086	0.030838	0.00807
24	0.049025	0.036086	-0.02334	0.016084
25	0.000546	0.057988	-0.02334	0.016084
26	-0.01823	-0.01123	-0.04169	0.006355
27	0.054839	0.049666	-0.02549	0.006355
28	0.048674	0.049666	0.009576	0.006355
29	-0.03927	0.049666	0.059075	0.006355
30	0.036458	0.049666	0.059075	-0.00697
31	0.033519	0.049666	0.044997	-0.00697
32	0.033519	0.049666	0.003084	0.00469
33	-0.03727	0.023797	-0.04815	0.014017
34	-0.03727	0.023797	0.032431	0.005732
35	-0.03727	0.011347	-0.01907	0.005732
36	0.011041	0.011347	0.05396	0.003434
37	0.010568	0.009003	0.05396	-0.01456
38	-0.03691	0.009003	0.040603	0.00995
39	0.025212	0.018684	0.009388	0.009067
40	0.025212	0.032342	0.021943	0.014171
41	0.054834	0.021874	-0.00743	0.002331
42	0.035735	0.046239	-0.00569	-0.01431
43	0.037857	0.046239	-0.00569	-0.01431
44	0.037857	0.046239	-0.04792	0.016165
45	0.044922	0.032211	-0.05528	0.010122
46	0.059402	0.025913	-0.05528	0.016293
47	0.047873	0.025913	0.018423	0.016293

48	0.026094	0.025913	0.058757	0.00338
49	-0.01176	0.025913	0.014889	0.000425
50	-0.04977	-0.00192	0.022615	0.010352
51	-0.04977	0.010958	-0.05282	0.016886
52	0.057876	0.019121	-0.01368	-0.01206
53	0.02401	0.019121	0.05949	-0.01206
54	0.02401	0.034856	-0.02596	-0.01206
55	0.013081	-0.05688	0.057066	-0.01206
56	0.013081	0.044995	0.057066	-0.01206
57	0.036307	0.044995	0.035075	-0.01206
58	0.029982	0.044995	0.047332	0.003851
59	0.029982	0.044995	0.047332	0.008376
60	0.003637	0.044995	0.047332	-0.00715
61	-0.05035	0.008282	0.031021	0.017333
62	-0.02488	0.008282	-0.0424	0.010283
63	0.047388	0.007367	-0.0424	0.005602
64	-0.01346	-0.03177	-0.0022	0.006571
65	-0.01346	-0.04273	0.054362	0.008642
66	0.022876	-0.04273	0.014288	0.003819
67	0.022876	-0.04273	0.017054	-0.00318
68	0.049389	-0.04273	-0.04985	0.002246
69	-0.00377	-0.04273	-0.04985	-0.00978
70	0.034604	-0.04273	-0.0321	-0.00978
71	0.042167	-0.04847	-0.05556	0.008308
72	0.012533	-0.04847	-0.02897	-0.01197
73	0.050662	0.044718	0.051463	-0.01197
74	-0.01811	0.0535	0.018866	0.010014
75	-0.02574	-0.02641	0.018866	0.002571
76	0.024704	0.002676	0.010311	-0.00477
77	0.025773	0.002676	0.010311	0.001778
78	0.03072	-0.01222	0.010311	-0.01123
79	0.03072	-0.01222	0.010311	-0.01123
80	0.047998	0.052477	-0.05721	-0.00694
81	-0.01343	0.014574	-0.05721	-0.00694
82	-0.01343	0.014574	0.038449	-0.01325
83	0.012212	-0.03168	0.038449	-0.01596
84	0.040931	-0.03168	-0.0109	-0.01596
85	-0.01575	-0.03168	-0.0109	-0.00475
86	-0.01575	0.009564	0.019096	0.011601
87	0.025479	0.016007	0.019248	-0.01611
88	0.025479	0.016007	0.019248	0.002401

89	-0.0609	0.016007	0.019248	0.002401
90	-0.0609	0.016007	0.041875	0.016375
91	0.014217	0.004016	0.041875	-0.01692
92	0.013198	0.00824	0.041875	0.016516
93	-0.05636	0.00824	0.024551	0.002908
94	0.001677	-0.02018	0.043203	-0.01656
95	0.008775	0.048513	-0.02127	0.000422
96	0.008775	0.008204	0.027508	0.000422
97	0.014257	0.013271	0.027508	-0.01531
98	0.014257	-0.0264	0.027508	-0.00648
99	-0.0359	-0.04224	-0.04352	0.009593
100	0.003768	-0.04224	-0.01362	0.005754

Simulation 5

	SERVO L	SERVO LC	SERVO RC	SERVO R
1	0	0	0.037535	0.046544
2	0.031985	0	0.025223	0.007101
3	0.012865	0	-0.05322	0.045201
4	0.055602	0	-0.04986	0.000814
5	0.055602	0.027673	-0.04986	0.000814
6	0.055602	0.027673	-0.05733	0.059461
7	0.031186	0.027673	-0.03151	0.025793
8	0.047997	0.009319	0.004314	0.040222
9	0.039825	-0.01894	0.03196	0.023888
10	0.039825	-0.02722	-0.03916	0.032343
11	0.039825	-0.04143	0.021426	0.036698
12	0.039825	0.02446	0.028333	0.036698
13	0.004504	0.02446	0.017689	0.036698
14	0.004504	0.02446	0.017612	0.001726
15	0.043163	0.02446	-0.00262	-0.00681
16	0.030632	-0.03961	0.038019	-0.02788
17	0.030632	0.037105	0.042376	-0.02788
18	-0.05459	0.037105	-0.00326	-0.02788
19	-0.04192	-0.01632	0.056342	-0.00409
20	-0.02181	0.000844	-0.00528	0.011472
21	0.013029	-0.00076	0.044379	0.011472
22	0.013029	-0.00725	0.018705	-0.03048
23	0.046677	0.034498	0.018705	-0.03048
24	-0.00463	0.034498	-0.01925	-0.01674

25	0.009578	0.027727	-0.01925	0.040286
26	0.009578	0.027727	-0.01925	0.040286
27	0.009578	0.013753	-0.01925	0.040286
28	0.037302	-0.01598	-0.01925	0.040286
29	0.025586	0.022062	-0.00571	0.058771
30	0.025586	0.036488	0.054712	0.058771
31	0.040863	0.036488	-0.03982	0.047345
32	0.040863	0.027711	-0.03982	0.047345
33	0.040863	0.027711	0.028695	0.013769
34	0.031386	0.040472	0.017679	0.014943
35	0.031386	-0.00206	0.017679	0.028255
36	-0.00643	0.014282	0.017679	-0.00455
37	-0.00643	-0.02175	0.004757	-0.02886
38	-0.00643	-0.00924	0.030714	-0.02886
39	-0.00643	0.003041	0.030714	-0.05427
40	-0.00643	-0.02278	0.030714	0.019775
41	0.046256	-0.01491	0.030714	0.007737
42	0.046256	-0.01491	0.030714	-1.36E-05
43	0.046256	-0.02105	-0.05081	0.014318
44	-0.04019	-0.02105	-0.05081	0.014318
45	0.057895	0.031535	0.057546	0.014318
46	0.057895	0.031535	0.057546	0.039227
47	0.057895	0.001957	-0.03715	0.039227
48	0.057895	0.020267	-0.04774	0.039227
49	0.002494	-0.00063	0.040957	0.039227
50	0.002494	0.019699	-0.05139	0.039227
51	0.055773	0.018403	-0.00735	0.039227
52	-0.04682	0.000543	-0.00735	-0.03277
53	-0.04682	0.012827	0.055273	0.028733
54	0.00062	0.033447	0.055273	0.040631
55	0.00062	0.020817	-0.02735	0.03541
56	0.060361	0.020817	-0.03896	0.02757
57	0.060361	0.017628	-0.03896	0.02757
58	-0.04507	-0.01056	-0.03896	0.02757
59	0.037299	-0.01056	0.037133	0.018646
60	0.037299	0.037921	0.037133	-0.0183
61	-0.04825	0.011901	0.037133	-0.00895
62	-0.04825	0.011901	-0.02297	-0.05333
63	-0.05923	-0.00231	0.015549	0.046166
64	0.045405	0.006281	-0.03499	0.046166
65	0.009996	0.006281	-0.03499	0.046166

66	0.050982	0.006281	0.00404	-0.00501
67	-0.04061	0.025718	0.032901	-0.00501
68	-0.04061	0.025718	0.050019	-0.00501
69	0.039567	-0.00796	0.011089	-0.01184
70	-0.01976	0.033416	0.023775	-0.01166
71	0.000972	-0.0005	0.01775	0.042092
72	0.05971	0.0266	-0.02357	0.03817
73	0.05971	-0.00379	-0.02931	0.03817
74	0.05971	0.019621	0.014253	0.018578
75	0.05971	0.019621	-0.04722	0.018578
76	0.052014	0.006502	-0.02953	0.018578
77	-0.00726	0.006916	0.024393	-0.03724
78	0.02593	-0.03802	0.012402	-0.03724
79	0.02593	-0.03802	0.050628	0.042418
80	0.006791	-0.03802	0.031476	0.042418
81	0.009566	-0.03802	0.020127	0.042418
82	0.009566	0.039747	0.054102	0.042418
83	-0.02508	-0.03952	0.046834	-0.04875
84	-0.00358	-0.03952	0.057198	-0.01079
85	-0.00358	-0.00118	0.057198	-0.01668
86	-0.00358	0.035396	0.002348	-0.04769
87	-0.00358	0.035396	-0.03061	-0.0122
88	-0.00358	0.009017	-0.03061	-0.0122
89	0.029759	-0.00252	0.056605	-0.0122
90	0.047365	-0.03059	-0.04042	0.012614
91	0.055721	-0.03515	-0.01982	0.012614
92	0.055721	0.002678	-0.01284	0.012614
93	-0.04562	0.002678	0.028118	0.012614
94	-0.04562	0.002678	0.002793	0.012614
95	0.02693	0.035445	0.002793	0.045501
96	0.02693	0.033829	0.003444	0.045501
97	0.033934	-0.03529	-0.04491	0.033064
98	-0.05129	-0.02038	0.014352	0.008672
99	0.042912	-0.02038	0.014352	0.006606
100	0.042912	-0.01379	0.036617	0.006606

Simulation 6

	SERVO L	SERVO LC	SERVO RC	SERVO R
1	0	0	0.037535	0.046544

2	0.031985	0	0.025223	0.007101
3	0.012865	0	-0.05322	0.045201
4	0.055602	0	-0.04986	0.000814
5	0.055602	0.027673	-0.04986	0.000814
6	0.055602	0.027673	-0.05733	0.059461
7	0.031186	0.027673	-0.03151	0.025793
8	0.047997	0.009319	0.004314	0.040222
9	0.039825	-0.01894	0.03196	0.023888
10	0.039825	-0.02722	-0.03916	0.032343
11	0.039825	-0.04143	0.021426	0.036698
12	0.039825	0.02446	0.028333	0.036698
13	0.004504	0.02446	0.017689	0.036698
14	0.004504	0.02446	0.017612	0.001726
15	0.043163	0.02446	-0.00262	-0.00681
16	0.030632	-0.03961	0.038019	-0.02788
17	0.030632	0.037105	0.042376	-0.02788
18	-0.05459	0.037105	-0.00326	-0.02788
19	-0.04192	-0.01632	0.056342	-0.00409
20	-0.02181	0.000844	-0.00528	0.011472
21	0.013029	-0.00076	0.044379	0.011472
22	0.013029	-0.00725	0.018705	-0.03048
23	0.046677	0.034498	0.018705	-0.03048
24	-0.00463	0.034498	-0.01925	-0.01674
25	0.009578	0.027727	-0.01925	0.040286
26	0.009578	0.027727	-0.01925	0.040286
27	0.009578	0.013753	-0.01925	0.040286
28	0.037302	-0.01598	-0.01925	0.040286
29	0.025586	0.022062	-0.00571	0.058771
30	0.025586	0.036488	0.054712	0.058771
31	0.040863	0.036488	-0.03982	0.047345
32	0.040863	0.027711	-0.03982	0.047345
33	0.040863	0.027711	0.028695	0.013769
34	0.031386	0.040472	0.017679	0.014943
35	0.031386	-0.00206	0.017679	0.028255
36	-0.00643	0.014282	0.017679	-0.00455
37	-0.00643	-0.02175	0.004757	-0.02886
38	-0.00643	-0.00924	0.030714	-0.02886
39	-0.00643	0.003041	0.030714	-0.05427
40	-0.00643	-0.02278	0.030714	0.019775
41	0.046256	-0.01491	0.030714	0.007737
42	0.046256	-0.01491	0.030714	-1.36E-05

43	0.046256	-0.02105	-0.05081	0.014318
44	-0.04019	-0.02105	-0.05081	0.014318
45	0.057895	0.031535	0.057546	0.014318
46	0.057895	0.031535	0.057546	0.039227
47	0.057895	0.001957	-0.03715	0.039227
48	0.057895	0.020267	-0.04774	0.039227
49	0.002494	-0.00063	0.040957	0.039227
50	0.002494	0.019699	-0.05139	0.039227
51	0.055773	0.018403	-0.00735	0.039227
52	-0.04682	0.000543	-0.00735	-0.03277
53	-0.04682	0.012827	0.055273	0.028733
54	0.00062	0.033447	0.055273	0.040631
55	0.00062	0.020817	-0.02735	0.03541
56	0.060361	0.020817	-0.03896	0.02757
57	0.060361	0.017628	-0.03896	0.02757
58	-0.04507	-0.01056	-0.03896	0.02757
59	0.037299	-0.01056	0.037133	0.018646
60	0.037299	0.037921	0.037133	-0.0183
61	-0.04825	0.011901	0.037133	-0.00895
62	-0.04825	0.011901	-0.02297	-0.05333
63	-0.05923	-0.00231	0.015549	0.046166
64	0.045405	0.006281	-0.03499	0.046166
65	0.009996	0.006281	-0.03499	0.046166
66	0.050982	0.006281	0.00404	-0.00501
67	-0.04061	0.025718	0.032901	-0.00501
68	-0.04061	0.025718	0.050019	-0.00501
69	0.039567	-0.00796	0.011089	-0.01184
70	-0.01976	0.033416	0.023775	-0.01166
71	0.000972	-0.0005	0.01775	0.042092
72	0.05971	0.0266	-0.02357	0.03817
73	0.05971	-0.00379	-0.02931	0.03817
74	0.05971	0.019621	0.014253	0.018578
75	0.05971	0.019621	-0.04722	0.018578
76	0.052014	0.006502	-0.02953	0.018578
77	-0.00726	0.006916	0.024393	-0.03724
78	0.02593	-0.03802	0.012402	-0.03724
79	0.02593	-0.03802	0.050628	0.042418
80	0.006791	-0.03802	0.031476	0.042418
81	0.009566	-0.03802	0.020127	0.042418
82	0.009566	0.039747	0.054102	0.042418
83	-0.02508	-0.03952	0.046834	-0.04875

84	-0.00358	-0.03952	0.057198	-0.01079
85	-0.00358	-0.00118	0.057198	-0.01668
86	-0.00358	0.035396	0.002348	-0.04769
87	-0.00358	0.035396	-0.03061	-0.0122
88	-0.00358	0.009017	-0.03061	-0.0122
89	0.029759	-0.00252	0.056605	-0.0122
90	0.047365	-0.03059	-0.04042	0.012614
91	0.055721	-0.03515	-0.01982	0.012614
92	0.055721	0.002678	-0.01284	0.012614
93	-0.04562	0.002678	0.028118	0.012614
94	-0.04562	0.002678	0.002793	0.012614
95	0.02693	0.035445	0.002793	0.045501
96	0.02693	0.033829	0.003444	0.045501
97	0.033934	-0.03529	-0.04491	0.033064
98	-0.05129	-0.02038	0.014352	0.008672
99	0.042912	-0.02038	0.014352	0.006606
100	0.042912	-0.01379	0.036617	0.006606