A Biologically Inspired Four-Legged Walking Robot (Robo-Dog)

School of Engineering and Information Technology

Harold Ear
Bachelor of Engineering Honours
2016 - 2017

Thesis Supervisor: Associate Professor Graeme Cole
Declaration
I, Harold Ear, declare that this thesis is my own account of my research and contains as its main content work which has not previously been submitted for a degree at any tertiary education institution.

Word Count: 13313

Signed:

Name:

Date:
Abstract

The Robo-dog was designed and implemented by a Shiqi Peng, a PhD student in 2006. The dog incorporated many control systems such as Reinforced Learning, Subsumption Architecture and a Central Pattern Producer. The dog since 2013 has been in an inactive state, the dog now is used to educate students on various concepts such as debugging, electronics and programming. A variety of issues stem from the dog and are not limited to the lack of documentation, missing equipment and components, broken and faulty hardware components and a limited functioning program.

The objectives of the project are: restore various capabilities of the dog, moving the dog from Point A to Point B, implementing a control system utilising pre-existing sensors on the dog.

To begin attempting to manoeuvre the dog, various components such as tilt-sensors, foot-sensors and microcontroller communication needed to be addressed as the components were not in a working state. Documentation has been updated or created to reflect the changes that have been made to the components of the dog.

Moving the dog from Point A to Point B was achieved through various walking techniques, by adapting and adjusting the limb movements in the walk gait for a dog and then applying the concepts to the robo-dog. The implementation of the limb movement and the walk gait focused on maximising forward movement per cycle incorporating a sequential approach where each movement were dependant on one other.

A control system was implemented in the form of Subsumption Architecture; the behaviours implemented are to determine the balance of the dog, walk sequence of the dog and to determine whether the dog is walking on a ramp. The implementation goal is to provide an easy-to-expand control system so that future students can add additional behaviours to the system.
Acknowledgements

I would firstly like to express my sincerest appreciation to my academic supervisor Graeme Cole for always being willing to spare some time, assisting and oversight throughout my university degree and thesis project.

Secondly, thank you Jeff Laava for always being willing to help fix my robo-dog’s broken legs, hips and offering your prowess electronics skills even through your busy schedule.

Thirdly especially thanks to Campbell Strachan, Ben Pattimore, Allison Arnaud and Sean Ting for your constant support, encouragement and the motivation during hard times. The trip through my university degree would not have been possible or as enjoyable without you four.

I would also like to additionally thank Kevin Liang, Aiden Stankovic, Mitch Glenister, Milad Mostafavypour for their support, help and wisdom through my degree and especially during my thesis.

And finally thanks to the rest of my friends and family for the patience and support over the duration of my university degree.
# Contents

Declaration ................................................................................................................................. i  
Abstract ................................................................................................................................. ii  
Acknowledgements ............................................................................................................... iii  
List of Tables ........................................................................................................................... vii  
List of Figures ......................................................................................................................... vii  
Table of Abbreviations ........................................................................................................... viii  
1.0 Introduction ......................................................................................................................... 1  
  1.1 Project Objectives ............................................................................................................ 1  
2.0 Background ......................................................................................................................... 2  
  2.1 Shiqi Peng’s Robo-Dog .................................................................................................... 2  
  2.2 BigDog, the Rough-Terrain Quaduped Robot .................................................................. 3  
  2.3 Comparison of Wheeled and Legged Robots .................................................................. 4  
2.4 Subsumption Architecture .............................................................................................. 5  
  2.4.1 Subsumption Requirements ....................................................................................... 6  
  2.4.2 Brook’s Mobile Robot .................................................................................................. 7  
2.5 Dog Movement Cycle (Gaits) .......................................................................................... 9  
  2.5.1 The Walk .................................................................................................................... 10  
  2.5.2 The Flying Trot .......................................................................................................... 10  
  2.5.3 The Gallop (Rotatory) ............................................................................................... 11  
3.0 Robo-Dog Overview .......................................................................................................... 12  
  3.1 Programming ................................................................................................................... 12  
  3.2 Communications ............................................................................................................. 13  
  3.3 Microcontroller and I/O boards ...................................................................................... 13  
  3.4 Limbs ............................................................................................................................... 14  
  3.5 Sensors ............................................................................................................................ 14  
    3.5.1 Tilt sensors ................................................................................................................ 14  
    3.5.2 Feet sensors .............................................................................................................. 15  
    3.5.3 Potentiometers .......................................................................................................... 16  
4.0 Rebuilding ......................................................................................................................... 17  
  4.1 Communications ............................................................................................................. 17  
  4.2 Microcontroller ............................................................................................................... 17  
  4.3 NMIS-boards .................................................................................................................... 18
List of Tables

Table 1 - Limb Sequence and Actions ..........................................................................................................................24
Table 2 - Back-leg Robo-dog limb sequence ..................................................................................................................25
Table 3 - Front-leg Robo-dog limb sequence ..................................................................................................................25
Table 4 - Layer 1 stand-up outcomes ............................................................................................................................32
Table 5 - Layer 2 Outcomes ...........................................................................................................................................35
Table 6 - Limbs bit configurations ....................................................................................................................................57

List of Figures

Figure 1 - Subsumption Layer Hierarchy .........................................................................................................................6
Figure 2 - Brook’s robot behaviour hierarchy .....................................................................................................................8
Figure 3 - The Flying Trot ..................................................................................................................................................10
Figure 4 - The Gallop (Rotatory) ......................................................................................................................................11
Figure 5 - Robo-dog (Limbs Contracted) ........................................................................................................................12
Figure 6 - Mercury Switches (Tilt-sensor) ........................................................................................................................15
Figure 7 - Rod and Stopper ................................................................................................................................................16
Figure 8 - Foot Sensor .......................................................................................................................................................16
Figure 9 - Digital Isolation Board - Back(Left), Front(Right) ..........................................................................................19
Figure 10 - Issue 5 short ...................................................................................................................................................21
Figure 11 - Pull-down resistor circuit ..............................................................................................................................22
Figure 12 - Walk (Gait) of a dog .......................................................................................................................................23
Figure 13 - Robo-dog (2017) Behaviour Hierarchy .......................................................................................................28
Figure 14 - Layer 0 ...........................................................................................................................................................29
Figure 15 - Layer 1 ..........................................................................................................................................................31
Figure 16 - Layer 2 ..........................................................................................................................................................34
Figure 17 - WalkSequenceOne .......................................................................................................................................37
Figure 18 - WalkSequenceTwo .......................................................................................................................................37
Figure 19 - Begin-Until Loop ..........................................................................................................................................38
Figure 20 - PWM generation ..........................................................................................................................................39
Figure 21 - Tilt check task ...............................................................................................................................................40
Figure 22 - Walk Dog Task ..............................................................................................................................................40
Figure 23 - Task Layer0 ..................................................................................................................................................41
Figure 24 - Task Layer1 ..................................................................................................................................................42
Figure 25 - Task Layer2 ..................................................................................................................................................43
Figure 26 - Robo-dog Broken Hinges .............................................................................................................................50
Figure 27 - Digital Isolation Board (Circuit Diagram) .................................................................................................60
### Table of Abbreviations

<table>
<thead>
<tr>
<th>Abbreviation</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>Dog</td>
<td>Robo-dog</td>
</tr>
<tr>
<td>Central Pattern Producer</td>
<td>CPP</td>
</tr>
<tr>
<td>Central Pattern Generator</td>
<td>CPG</td>
</tr>
<tr>
<td>Pulse Width Modulation</td>
<td>PWM</td>
</tr>
<tr>
<td>Integrated Circuit</td>
<td>IC</td>
</tr>
<tr>
<td>Inertia Measurement Unit</td>
<td>IMU</td>
</tr>
<tr>
<td>Inter-Integrated Circuit</td>
<td>I²C</td>
</tr>
</tbody>
</table>
1.0 Introduction

The “Robo-dog” (dog) was designed and implemented by a PhD student, Shiqi Peng in 2006, since then several students have worked on the dog. The last student to work on the dog was in 2013, since then the dog remains in an inactive state [1]. The dog has lost most of its documentation, equipment and program due to the move from Murdoch’s Rockingham Campus to the South Street Campus. The dog is now being used for educational purposes for students on various aspects such as electronics, programming and debugging methods. The primary objective is to rebuild the dog and restore different functionalities to the dog which include: implementing a walking strategy to move the dog from Point A to Point B and to apply a control structure.

The project is broken down into five major phases: The rebuilding phase, implementation of the dog movement cycle, implementation of subsumption architecture and programming.

1.1 Project Objectives

The primary purposes can be further broken down into smaller goals. The rebuilding phase involves rebuilding and repairing various components such as the communications and digital isolation board. This objective also includes creating and updating the documentation to accurately reflect the current conditions, configuration and changes to the dog. After completing the rebuilding aspect, the next objective is to utilise the dog’s legs and determine a movement sequence through the use of different gaits.

Once the movement sequence has been designed, the last objective is to possibly use all the current sensors and actuators on the dog to implement a control system in the form of the Subsumption Architecture (6.0 Implementing Subsumption Architecture) that will allow the dog to think for itself.
2.0 Background

The dog incorporates hardware components such as pneumatic cylinders, various sensors and a microcontroller which uses Forth as the programming language. Due to lost documentation, the objective for former thesis students was to restore and reverse-engineer the dog’s hardware and software capabilities. Since then, improvements have been made to the walking sequence of the dog, design of the potentiometers and feet sensors and various upgrades to the I/O and communications of the system. Furthermore, the dog is based on a biological greyhound where the pneumatic cylinders, body and sensors are placed in strategic positions to replicate the movement of the limbs of the dog further described in “3.0 Robo-Dog Overview” [2].

2.1 Shiqi Peng’s Robo-Dog

The Robo-dog (dog) was originally designed and implemented by Shiqi Peng (A Biologically Inspired Four-Legged Walking Robot), the components of the dog are described in “3.0 Robo-Dog Overview” [2]. The dog is intentionally designed to be unbalanced, this is to incorporate the idea of Reinforced Learning, where the Robo-Dog can use trial and error to determine how to balance itself. Along with Reinforced Learning, to drive the robot, Shiqi Peng has implemented a type of control structure into the dog called the Subsumption Architecture, and its walking sequence is determined by a Central Pattern Producer [2]. Through the use of Reinforced Learning, Subsumption Architecture and the Central Pattern Producer the dog is able to readjust its walking behaviour for different surfaces such as slopes, stairs and uneven ground.

Shiqi implemented a Subsumption Architecture module for each leg on the dog, having a total of 4 subsumption architectures in parallel. The multiple legs are governed by the central pattern producer, which controls the sequence the dog is walking in. Each leg can control itself independently relying on the joint sensors, tilt sensors and mechanical feet switches to determine the walking behaviours.
Theoretically, if the dog detects an unbalanced signal from the tilt-sensors, the legs that are affected will re-adjust its walking behaviour and actuate its outputs accordingly in an attempt to balance the dog correctly.

The Central Pattern Producer (CPP) is an aspect of the Central Pattern Generator (CPG) where movements or actions are produced based off responses to an input. Examples include knee-jerk reactions and birds in flight, the CPG are movements and actions based on natural instincts [3]. Shiqi has implemented the CPP based on a biological dog’s walking motion [2]. CPP is the pattern or order in which the leg needs to move to be able to keep the dog balanced and straight.

Past thesis students, Karl Bernet (2009) and Ashley Cocker (2013) have most recently worked on the Robo-Dog. The task given was to reverse-engineer the dog’s capabilities and programs. Improvements have been made to the dog since Shiqi Peng such as changing the walking sequence of the dog, multiple hardware changes in its I/O, communications and sensors [1] [4].

2.2 BigDog, the Rough-Terrain Quaduped Robot

Boston Dynamic’s BigDog was developed and designed in 2005 to be used for rough terrain, implementing a variety of sensors and actuators to determine its various behaviours [5]. BigDog uses a go-kart engine with a hydraulic system as the actuators for each limb. The advantage of the BigDog is through the use of hydraulic actuators, with an always changing environment the actuators are able to respond quickly and accurately to compensate for changes. Hydraulic actuators can stop at any position whereas compared to pneumatic actuators where the position of the actuator can be in one of two states.

The limbs of the BigDog are structured closely to resemble an animal’s leg and incorporate a trot gait as its main movement behaviour. The sensors used are GPS, gyroscope and accelerometers, joints angles, stereo vision and light detection and ranging (LIDAR) and various monitoring equipment for the battery voltage, hydraulic pressure, flow and temperatures and engine temp and speed. The on-board computer
used is a PC104 stack motherboard with a Pentium CPU using QNX real-time OS and the programming language C++ [5] [6].

The main functions of the BigDog are: control, servo and logging. The servo function is used to monitor the various sensors on the robot, as well as controlling the joints, actuators and engine. The logging function logs data for development analysis. The control functions implement 3 principles: support, balance and posture. The robot controls the support by bouncing on springs implemented in the legs, the balance is maintained by moving the legs in a symmetrical fashion such as the trot gait and the posture is controlled by positioning the robot legs in different stances [5].

2.3 Comparison of Wheeled and Legged Robots

There are many disadvantages and advantages to wheeled and legged robots; wheeled robots are energy efficient on flat surfaces, cheaper, simpler to design and low level tasks such as forward, backwards and steering which are simpler to implement in a wheeled robot, this is due to having fewer moving parts to control the movement of the robot [7].

As for legged robots, the robot must decide where to puts its feet, which is not as simple to implement as a wheeled robot. Legged robots pose great advantages to mobility such as being able to travel on rough terrain and step over obstacles. Wheeled robots are also able to function on rough terrain by rolling over small obstructions except the robot will become stuck on obstacles that cannot be overcome [7].

Legged robots have the advantage of individual control of each limb or leg to determine the placement of the set of legs. Independent control of each leg will allow the robot to travel through dynamically changing environments and obstacles, whereas a legged robot using a fixed walking pattern is able to travel through rough terrain, though unable to adapt to dynamically changing environments and obstacles efficiently as the individually controlled limbs/legs.
When comparing wheeled robots to legged robots, if a robot is to perform basic tasks it is beneficial to use a wheeled robot due to its cost, simplicity design and ease of tasks implementation since the legs are not utilised to its potential. Legged robots will be able to adapt to a dynamically changing environment though at the cost of the increased difficulty of controlling a greater number of moving parts and variables, and the cost of actuators used for each limb is greater than wheeled robots.

### 2.4 Subsumption Architecture

Rodney Brooks first introduced the Subsumption Architecture in 1986 [8]; the architecture uses a principle of “Sensing-Reacting”, where a behaviours, movements or actions are triggered by the applications sensors such as potentiometers, switches and accelerometers [2].

The issue with a traditional system is that it is built upon a narrow and static representation of the environment around the robot. The result would be that the action would take place for a period and then the robot would stop to recalculate its position and proceed again through its decision tree. Therefore, dynamic changes to the environment will have an adverse effect on a traditional system. To address these issues, the Subsumption Architecture was developed to allow the robot to sense its changing environment and react accordingly based on predetermined behaviours [8].

The source of the Subsumption Architecture revolves around using “task-achieving behaviours” in the system [8]. These “task-achieving behaviours” are layers in a hierarchal system. Brook’s experimental robot which will be explained in 2.4.2 Brook’s Mobile Robot, uses behaviours such as avoiding objects, wandering, exploring, building maps, monitoring change, identifying objects, planning changes to the world, reasoning about the behaviour of objects as layers in his control system.

The behaviours are represented as levels in a hierarchal control system as shown in Figure 1, where the higher levels can subsume the previous level, absorbing all the processes of the lower levels. The higher levels can be an extension of the lower levels with additional constraints; additionally, the higher levels
can suppress actions from the lower levels. Each level in the control system operates independently and is unable to communicate with each other while the highest-level gains priority over the actuators.

The sensor readings determine if a hierarchal layer within the control system should subsume some lower layers. The highest layer can suppress multiple modules (actions) or inherit their outputs to overwrite the commands sent to the actuator. Lower layers may not be used or affect parts of the system that are on a higher layer when it is subsumed.

![Figure 1 - Subsumption Layer Hierarchy](image)

2.4.1 Subsumption Requirements

There are multiple requirements when using Subsumption Architecture as a control system, requirements such as Multiple Goals, Multiple sensors, Robustness and Extensibility [8].

Robots are designed to achieve multiple purposes or goals. To apply the Subsumption Architecture the robot can respond to high priority goals while still tending to low priority goals [8]. For example to move the dog from point A to point B the dog must walk to reach its destination but the dog must also maintain its balance, so it does not fall over en route.
The robot’s decisions are based on sensors therefore robots are designed to incorporate many types of sensors such as limit switches, potentiometers, accelerometers, visual sensors and so on. Normally sensors have an error-uncertainty range in their readings so the robot’s decision making must accommodate for its range [8]. For example, if a potentiometer reading is high then open the door, otherwise if it is low close the door. If the potentiometer has an error-uncertainty range of 10% and the high limit to open the door is >100 ohms, the low limit to close the door is <100 ohms. If the value of the potentiometer is 95 ohms, applying the uncertainty-error the value could potentially read 104.5. The result will open the door instead of closing the door.

The Subsumption Architecture’s ability to handle loss of inputs must be robust, so that when a sensor fails the robot will continue to operate as best as it can without relying on the sensor [8]. Repetition often occurs in the program code when attempting to make the robot as robust as possible.

Extensibility (Growth) is taken into consideration when using multiple sensors and adding in capabilities as the processor needs to be able to process the data. If the processing power and memory is lacking then the capabilities of the robot will be affected and can become unresponsive and impaired [8].

2.4.2 Brook’s Mobile Robot

Brooks describes each behaviour as a layer in his control system as shown in the figure below; the zeroth layer is the lowest level and is used to avoid objects by running away (runaway command) when approaching an obstacle. The sensor used is a sonar module which produces a map of obstacles in polar coordinates [8]. From the sensor readings, the robot will either halt or change directions and move forward again.
The first level is used to wander around in its environment; wander incorporates aspects of the zeroth layer to be able to function correctly. When the robot is in its wander behaviour, the robot gains the ability to wander around with no goal but must avoid objects. The first level can suppress aspects of the zeroth layer, for example, the wander level will suppress the runaway command if there is an object in the way, the robot will rotate itself and move in a different direction [8].

The second layer is used to explore its environment taking note of interesting locations to visit using a sensor that observes a corridor of free space, by doing so the first level ‘wander’ is suppressed as it no longer needs to wander around aimlessly avoiding objects. With a planned heading and its map that is produced in the zeroth layer, explore uses ‘avoid’ and its coordinates to visit places of interest [8].

When implemented into a mobile robot, under the control of layer 0 the robot finds an empty space and parks itself until approached by objects. The robot will then run away from the moving objects as the purpose of this layer is to avoid objects [8].

Under the control of layer 1 the robot is not content with parking itself and proceeds off into a random direction, when approached by an object or approaching an object the robot will run away [8].
Under the control of layer 2 the robot determines a far point in the room and proceeds in the direction of interest. As objects approaches the robot, the robot will attempt to find another route to the point of interest. If there is close contact between objects, the robot will manoeuvre itself between the objects eventually reaching its destination [8].

2.5 Dog Movement Cycle (Gaits)

The movement cycle (also known as “Gaits”) is a pattern of actions performed by the limb during locomotion. A variety of gaits can be observed over time from the same individual animal to many other types of animals, the most identified and basic gaits for quadrupeds are walk, trot, pace, canter and gallop [9].

There are two different groups of gaits, the symmetric and the asymmetric. Symmetric gaits are when the limbs movements of one side of the body are identical to the limbs movement on the other side of the dog’s body and are evenly spaced between each movement. The limb’s movement on each side of the body occur in alternating intervals, examples of symmetric gaits are the walk, trot and pace. Asymmetric gaits are when the limb’s movement on one side of the body is not identical to the limbs movement on the other side of the body, as well as the motion between each side is irregularly spaced, examples of an asymmetric gait is gallop [9].

For the dog to move forward, the dog follows the specific order lift, swing, support and thrust. This allows the dog to gain momentum and traction to be able to move forward; each gait movement follows the specific order [10].
2.5.1 The Walk

The walking gait is the most widely observed gait and is the least tiring; this is due to the dog never having less than two feet touching the ground resulting in less energy exerted. The front-legs in the gait are used to slow and absorb shock whereas the rear-legs are used to drive the body forward [10].

2.5.2 The Flying Trot

The flying trot gait is a fast-paced movement, Figure 3 shows the sequence of each movement to complete a full cycle of the gait, the foot touching the ground is depicted by circles (back leg) and triangles (front leg). The two legs that are on the ground is the right front leg (triangle), left back leg (circle) and alternating in the next sequence is the left front leg (triangle) and right back leg (circle), between each alternating movement is a suspension period. The pair of legs moves in unison throughout the movement sequence allowing the weight of the dog to be supported once the legs make contact with the ground [11]. A normal trot follows the same sequence but without the suspension period between each alternating step.

Figure 3 - The Flying Trot
2.5.3 The Gallop (Rotatory)

The rotatory gallop is a fatiguing fast-paced gait as shown in Figure 4 that uses the step sequence: right back leg (circle), left back leg (circle), suspension, left front leg (triangle), right front leg (triangle), suspension. The weight is supported right before the dog is suspended in the air. The front legs support the weight during its contact; the contact time is also longer than the back legs which are used to provide thrust [12].

*Figure 4 - The Gallop (Rotatory)*
3.0 Robo-Dog Overview

The dog is made up of a large metal frame that houses various components such as a manifold, pneumatic cylinders, microcontroller and I/O boards and incorporates sensors such as potentiometers, mercury switches and mechanical switches. The ‘brain’ of the dog is powered by a Motorola MC68HC11 microcontroller, using the programming language Forth [2]. Due to the unbalanced nature of the dog, the dog is held up and suspended by wires and chains to prevent the hardware from being damaged when the dog is in operation as shown in Figure 5.

3.1 Programming

The programming language used is Forth, a high-level language developed by Charles H. Moore in 1960s-1970s [13] [14]. Forth is heavily utilised in the dog project due to previous programs implemented in Forth and also Murdoch students are taught Forth as a part of their engineering course. To communicate with the microcontroller a software environment called SwiftX is used that is capable of interfacing between the user and the microcontroller. SwiftX is used due to its compatibility with the programming language Forth [1] [2].
The program that was originally developed by Shiqi Peng has been lost although an incomplete version has been recovered. Past thesis students have worked on the version to restore various aspects of the dog, but the program’s currently dysfunctional. However, details on how the control structures were implemented, control of the actuators and reading of the sensors are available and can be used as a template.

### 3.2 Communications

When Shiqi Peng designed the dog in 2006 the communication interface used was an RS232 cable connecting the SwiftX software on the computer to the Motorola MC68HC11 microcontroller [2]. Since then the communication interface has changed between the computer and the microcontroller. The dog now communicates to the computer through a module that is capable of wireless communications and the traditional wired communication occurs through an RJ11 port.

The wireless module consists of a MAX232 Chip, APC200A/APC220A wireless module and a 3DR Radio Kit and a port for the RJ11 connection [1] [15] [16]. Unfortunately, the wireless antenna/device for the computer has been misplaced so it was not possible to communicate to the module through a wireless medium. Regardless of this communication to the dog was still possible through the RJ11 port on the module itself.

### 3.3 Microcontroller and I/O boards

The dog consists of a Motorolla 68HC11 microcontroller and an EPROM chip that resides on a New Micros Board NMIS 0021B [17], the EPROM chip contains the instruction set to be able to understand SwiftX/Forth programs; this board enables the microcontroller to be used allowing access to a variety of features such as inputs and outputs [17]. The various inputs and outputs of the dog are routed through an I/O expansion board (NMIS 3000) which allows 32 inputs/32 outputs and allows the address of the board to be changed [18]. To tie all the interfacing boards together the boards are placed on a NMIM-0006 board which offers the ability to attach six additional boards [19].
3.4 Limbs

The limbs of the dog consist of the foot, shank, thigh and hip. Connecting the limbs are the pneumatic cylinders which act as “muscles” [2]. The pneumatic cylinders are arranged in a specific configuration to replicate a dog’s leg. Using the pneumatic cylinders in this way offers 4 degrees of freedom per leg. There is a total of 4 pneumatic cylinders per leg, one between each limb and overall there are 16 pneumatic cylinders in the system. The manifold housing the cylinders is provided with a 75-psi compressed air supply which allows a smooth pneumatic cylinder actuation. The front legs and back legs are structured differently. The idea is to drive the robot with its back legs and steer the robot through the front legs [2].

The pneumatic cylinders are controlled by three-way solenoid valves offering the actions to extend, contract and lock. The valves are connected through a manifold which is located on the under-body of the dog; each pneumatic cylinder is connected to the valve through the pneumatic tubing. To control the pneumatic cylinders an I/O board is used (NMIS 3000). Each pneumatic cylinder is controlled through a digital input where each leg (0, 1, 2 and 3) is controlled by an 8-bits I/O port. Each pair of bits correspond to the actions contract and extend of each limb, for a detailed break-down of extend and contract bits refer to Appendix A - Limbs.

3.5 Sensors

The dog incorporates a variety of sensors using both analogue and digital inputs to determine its balance, its position and to a certain degree touch.

3.5.1 Tilt sensors

To determine the balance of the dog, tilt switches in the form of mercury switches are used. A total of 8 mercury switches are used where 4 switches detect a tilt of approximately 30 degrees, and the remaining 4 detect a tilt of 45 degrees as shown in Figure 6 [2]. The position of the mercury switches is placed on top of the chassis of the dog. Only a 30-degrees and 45-degrees configuration is needed to determine the balance of the dog because when the dog becomes completely unbalanced it has already
exceeded the range of 45 degrees activating the sensor. If the dog becomes unbalanced the tilt-sensors are activated, the sensors will send a logical 1 or 0 back to the microcontroller. An accelerometer was designed for the dog by a past thesis student to provide a more responsive feedback to the dog, though unfortunately since then the accelerometer has been misplaced and is no longer implemented into the dog [1].

3.5.2 Feet sensors

Originally Shiqi Peng designed the dog to use mechanical push switches to determine whether its legs have made contact with the ground. The push switches operate in a way that when the dog makes contact with the ground a stopper with a small rod attached to it will trigger the switch as shown in Figure 7 and Figure 8 shows the foot sensor. This will toggle the state of the switch from either ON (1) or OFF (0) [2].

![Figure 6 - Mercury Switches (Tilt-sensor)](image)
Improvements have been made to the design of the dog as the mechanical switches are used for walking on even surfaces but are unable to function correctly on uneven surfaces [2]. To resolve the issue of walking on different surfaces a previous student incorporated a pressure sensor into one of the feet sensors [1]. When the dog was first received, the dog had 3 mechanical switches for its feet and 1 pressure sensing foot.

### 3.5.3 Potentiometers

A total of 12 potentiometers are used as position sensors in the dog; there are 3 potentiometers per leg located at the joints. The purpose of the potentiometers are to determine where one limb is relative to another [2]. The potentiometers are analogue inputs are where if a joint moves the potentiometer will change its voltage accordingly and send the value back to the microcontroller.
4.0 Rebuilding

4.1 Communications

On testing, it was found that communications between the computer and the dog were non-existence producing issues such as time-outs, unresponsiveness and COM errors. To ensure that the communication issues was due to the wireless setup, the microcontroller board was taken out and isolated away from the rest of the system.

The wireless module was replaced with a traditional COM cable with an RS232 to RJ11 jack. The removal of the wireless module was necessary due to its lack of documentation of how the wireless module was constructed or operated. Replacing the wireless module with a COM cable proved to be the simplest, time efficient and most cost-effective method to address the problem. The cause of the issues of the wireless module could be due to shorts in the module.

When successfully connected to the microcontroller, the program is then compiled and downloaded onto the microcontroller. Though the connection was successful, SwiftX produced a particular error “Sync-Core Error, Kernel Mismatch” which suggested that the current version of SwiftX was not compatible with the current kernel version of the program.

4.2 Microcontroller

To be able to communicate and interact with the dog, the EPROM of the microcontroller would need to be erased and then reprogrammed. The reason for erasing and re-burning the EPROM was due to the SwiftX environment; the newer version of the SwiftX environment is not compatible with the older version which was currently installed on the EPROM. Appendix B - EPROM describes the process instructions for erasing and burning the EPROM.
4.3 NMIS-boards

The NMIS boards are a vital part of the operations of the robo-dog project as they are used to interface between the microcontroller and all its relevant inputs and outputs. Testing showed that the microcontroller could function correctly when it was isolated from the dog. When placed into the dog there were errors such as only half the inputs and outputs were able to communicate back to the microcontroller.

By systematically testing each component with an oscilloscope (microcontroller and I/O boards) the main cause of the problem was found to be a loose connection from the microcontroller to the interfacing board and then to the I/O boards. The microcontroller can output a high-bitrate square wave with no loss in transmission whereas the expansion I/O board’s HC688 chips were able to send a high-bitrate square wave on half the ports while the other half remains at zero.

The cause was a loose connection due to a combination of the metal housing of the microcontroller board and I/O boards being too tall which resulted in weak contact between the pins and the ports, additionally the NMIS-0006 boards pins and/or ports could have been oxidised. By removing the metal housing and placing the boards directly into the ports, the boards could interact with each other without issues.

4.4 Digital Isolation Board

A digital isolation board was created by Ashley Cocker that would interface all the solenoid inputs (extend and contract) to the microcontroller [1]. The issue was that the pneumatic cylinders operated at 12V whereas the microcontroller operated at 5V. Using Darlington transistor arrays (ULN2803) and Photocouplers (PS2502-4) Ashley could successfully isolate the 12V and 5V side as shown in Figure 9 [1] [20] [21].
There were a variety of issues that needed to be addressed before the digital isolation board was in optimal condition to be used, the list of issues range from:

1. Activating a single output from the microcontroller will activate two relays
2. Pin 19, when activated would short the whole manifold losing power to all the relays
3. A few Pins would not turn on at all
4. Pin 20 does not change state and would always be on
5. Turning on pin 5 would turn on outputs 4 and 5, whereas turning on pin 4 would only turn on output 4.
6. Pin 19 is always on after all the changes were made to the board

The digital isolation board itself was incredibly difficult to debug as there was no structure to the design, the circuit diagram designed for the digital isolation board was incomplete as it was missing a few components such as microcontroller 5V, ground, and optocoupler 5V ground.
Many of the issues were due to a common cause, though it was difficult to debug due to the design and layout of the circuitry on the board itself. All the issues were related to a short in the circuitry, though the components that were affected differ for each issue.

Issue 1 when activating a single output from the microcontroller activated two relays; the location of where the issue occurred was at the microcontroller input into the digital isolation board as shown in figure 9. A piece of solder was situated in-between the two rails when the microcontroller turned on that output both the rails were receiving the outputs. Removing the solder resolved the issue.

Issue 2 occurred when activating Pin 19; the manifold would lose power and would become unresponsive. The issue was due to grounding on the digital isolation board, when activating Pin 19 it would short the board. Connecting the ground and 5V from the microcontroller to its correct pins, Pin 1 and 0 respectively, fixed the grounding issues. The circuit diagram was adjusted to reflect this issue as the components did not exist on the diagram (see Appendix C – Digital Isolation Board).

Issues 3 and 4 both had similar causes as a few ports were not working correctly. The ports in this case would either always be in an on state or off state. The cause of the issue was faulty Darlington Transistor Arrays (ULN2803) and Optocouplers (PS2502-4). A total of two Optocoupler ICs and a Darlington Transistor Array IC were changed to correct the issue [22] [23].

Issue 5 was an unusual problem, where pin 5 would turn on outputs 4 and 5, whereas turning on Pin 4 would only turn on output 4. The issue is that there was a short between the two outputs on the board as shown in Figure 10. When pin 5 is on, the current from the microcontroller will travel to output 5 and across the short and activate output 4 as well, but when pin 4 is activated the current will travel to output 4 and across the short to ground. The soldering on the board shorted the two pins together due to human error, and due to the nature of the board it was difficult to debug. Resoldering the board fixed the issue.
In regards to all the changes and fixes that were accomplished, pin 19 remained on regardless of all the changes. The issue is unlikely a faulty IC as they have been changed multiple times, a short on the board exists and is quite difficult to locate, or the problem could be due to dry solder joints or to the age of the board. Regardless the output has little impact on the rest of the project; therefore, the time spent on fixing the output was not prioritised.

4.5 Foot and Tilt sensors

The foot and tilt sensors are digital inputs to the dog’s controller, the total required digital inputs for the foot and tilt sensors is 12. The wiring of both the foot and tilt sensors required modification, and an additional circuit was created. Firstly, when receiving the dog the feet were not connected with the wires cut off. Additionally, the type of sensor was inconsistent for the feet as three of the feet used mechanical switches and the one foot incorporated a pressure sensor.

Four pressure sensors were made by Ashley Cocker to replace the mechanical switches for the feet. The pressure sensor uses a force sensing resistor square to allow it to measure the amount of weight on the foot. The type of sensor is a ZFLEX A401-25 sensor that can measure up to 20kg, the required circuit uses an operational amplifier that will convert the change in resistance to a change in voltage [1] [24]. The signal is then sent to the microcontroller I/O as an analogue input with a quantised value of 0 to 255. The design of the feet provided by Ashley Cocker is more robust than the push-button mechanical
switch and is less prone to breaking on contact with the ground. Unfortunately, the remaining three pressure sensor feet have been misplaced and their whereabouts are unknown.

Ultimately the decision to use the mechanical switches over the pressure sensing feet was due to time constraints of constructing three new feet as opposed to constructing a mechanical switch foot. The mechanical switches were readily available and the limited analogue inputs/outputs available that the ZFLEXA401-25 sensors could be better utilised for other analogue sensors. The abundance of digital inputs and outputs favoured the use of mechanical switches.

A new design for the tilt sensing incorporating an accelerometer, IMU and I^2C was created by Ashley Cocker to replace the mercury switches tilt sensing. The new design would be more accurate and responsive compared to the mercury switch design. Unfortunately, the equipment that was created has been misplaced alongside the pressure sensor feet and cannot be located.

Both the foot and tilt sensor required a pull-down resistor circuit to function correctly as shown in Figure 11; the tilt sensor has an already built circuit though it was incomplete. A new pull-down resistor circuit was needed for the feet sensors. A pull-down resistor circuit is necessary when dealing with digital inputs and outputs, the voltage will be ‘pulled down’ to ground resulting in zero volts when the switch is not closed. Otherwise the microcontroller will observe a ‘floating’ 5V signal in both the open and close states of the switch.

![Figure 11 - Pull-down resistor circuit](image-url)
5.0 Implementing a Dog Movement Cycle

A dog walking motion was designed to allow movement from point A to point B. A variety of different gaits were considered such as walk, trot and gallop. The gaits considered are different though they do have some commonalities; the limb movement of each limb are quite similar. The limbs position themselves to allow each leg to achieve lift, swing, support and thrust of the dog. Each leg consists of four limbs: the hip, foot, shank and thigh. The implementation of the walk gait was only considered as the limb movement in each step was detailed and an important aspect.

5.1 The Walk

The walk consists of 9 different positions of the limbs to allow the dog to walk; each sequence achieves one of the following goals: lift, swing, support and thrust. The walking sequence of a dog is described in Figure 12.

![Figure 12 - Walk (Gait) of a dog](image)

To be able to control the dog’s movement in a way that represents a walking dog, each limb in the sequence is broken down. From left to right the dog’s sequence (in particular the grey back-leg from Figure 12), the dog supports itself from steps 0-4, lifts and swings the leg forward from steps 5-8 to position the leg and to thrust the dog forward with a backwards swing. The back legs are mainly used to thrust the dog forward where the front legs guide and pad the landing. The break-down of each limb for the grey back-leg is shown in Table 1.
Table 1 - Limb Sequence and Actions

<table>
<thead>
<tr>
<th>Steps</th>
<th>Limb</th>
<th>Action</th>
</tr>
</thead>
<tbody>
<tr>
<td>0 (initial)</td>
<td>Thigh, Foot, Shank Extended</td>
<td>Support</td>
</tr>
<tr>
<td>1</td>
<td>Thigh, Foot, Shank Extended</td>
<td>Support</td>
</tr>
<tr>
<td>2</td>
<td>Thigh, Foot, Shank Extended</td>
<td>Support</td>
</tr>
<tr>
<td>3</td>
<td>Thigh, Foot, Shank Extended</td>
<td>Support</td>
</tr>
<tr>
<td>4</td>
<td>Thigh, Foot Extended, Shank Contracted</td>
<td>Lift</td>
</tr>
<tr>
<td>5</td>
<td>Thigh, Foot Contracted, Shank Contracted</td>
<td>Swing Forward, Lift</td>
</tr>
<tr>
<td>6</td>
<td>Thigh Contracted, Foot Extended, Shank Contracted</td>
<td>Swing Forward, Lift</td>
</tr>
<tr>
<td>7</td>
<td>Thigh Contracted, Foot Extended, Shank Contracted</td>
<td>Swing Forward, Lift</td>
</tr>
<tr>
<td>8</td>
<td>Thigh Contracted, Foot Extended, Shank Extended</td>
<td>Swing Backwards, Support</td>
</tr>
</tbody>
</table>

The initial state of the dog's limbs is a fully extended thigh, foot and shank. Throughout steps 0 – 3 the dog's limbs remain unchanged; this is to provide balance for the dog to allow the other limbs to move. Steps 4 – 5 involve slowly contracting each limb whilst lifting the leg and swinging the leg forward. Steps 6 – 7 involves positioning the limbs to allow the dog to propel forward, once the shank has contacted the ground the leg is in position to thrust itself forward with a backwards swing shown in steps 8 and 0.

Even though the back-legs are structured differently to the front legs, the front legs follow the similar concept of positioning the limbs to provide the dog with support, lift, swing and thrust.

5.2 Implementing the Robo-dog Walk

Applying the same walk concept to the dog required modification of the limb sequences. Instead of individually moving each limb as per steps in Table 1, the steps were modified for the robo-dog’s back-legs and front-legs as shown in Table 2 and 3 respectively.
The modified walk sequence is grouped to perform two specific tasks:

1. Position the leg so it provides support and is in a ready state to thrust the dog forward.
2. Move the thighs of the leg so the dog thrusts forward.

Steps 0 and 1 for the back-leg are grouped together to provide the dog with support and positions the dog in a “ready-to-thrust forward” state, while steps 2 and 3 are used to thrust the dog forward.
The front-legs follow the same concept except that instead of thrusting the front-legs are used to guide the dog and provide a cushion for the landing. Where steps 0 and 1 are grouped together to provide the cushion for the dog, steps 3 to 6 are used to position and balance the dog.

To balance the dog correctly and for a smoother walking action, two or more legs will be moving at the same time, for example left-front leg and left-back leg will be moving at the same time and the right-front leg and right-back leg will be moving at the same time.
6.0 Implementing Subsumption Architecture

A simple implementation of a basic control system was decided once the dog could manoeuvre itself from point A to point B. The idea of implementing the Subsumption Architecture is to allow future students to easily expand on the dog’s behaviours. Unlike Shiqi Peng’s control structure where each leg incorporated an independent parallel control system, the basic control system will govern all aspects of the dog as a single entity.

In terms of the dog and implementing its new control system, multiple goals need to be distinguished. The main issue with the dog in its current state is the balance of the robot. The robot is upheld by a suspension system, so that if the robot does become unbalanced it will not damage itself. Multiple goals can be added into the control system such as maintaining balance, walking and walking on different surfaces as shown in Figure 13.

The robot itself has already incorporated many sensors ranging from mechanical switches, tilt-sensors in the form of mercury switches to potentiometers. These sensors will be used to determine the behaviours of the dog, such as the mechanical switches in the form of feet sensors will allow the robot to know when it has contacted the ground. The tilt-sensors will activate when the dog is unbalanced at 30 degrees and 45 degrees. The potentiometers will give feedback on the limbs joints.

Robustness is determined by how well the program and behaviours are developed. Over time the robustness of the dog can deteriorate if sensors fail due to age, faults or damage whilst the dog is moving.

The implementation of the Subsumption Architecture will incorporate behaviours such as balance, walking and walking on sloped surfaces.
Balance is the most important aspect of behaviour because it cannot walk anywhere without being upright and it can cause severe damage to the robot if it falls over. Since the dog is designed to be unbalanced there is a high probability that the dog can occasionally be in an unbalanced state. Before any actions or behaviours are performed the dog must be in a controlled or balanced state. The Walk layer uses aspects of the balance layer to perform the walk. Both tilt-sensors and foot-sensors are used to determine the actions for the walk-up-ramps layer.
6.1 Layer 0

The lowest layer behaviour is the most important layer in the control system, and is important to the dog due to its design. The dog is held upright by a suspension system. To operate the dog in optimal conditions the dog must be in a balanced condition before proceeding with other behaviours.

Layer 0 is used to control and maintain the balance of the system using tilt-sensors in the form of mercury switches; there are two positions that can determine whether the dog is unbalanced or balanced. If no switches are active the dog is balanced, if any of the first switches (1, 2, 3, 4) are activated the dog is partly tilted (30-degrees switch) and if both the first (1, 2, 3, 4) and second switches (5, 6, 7, 8) are active the dog is extremely tilted (45-degrees switch) as shown in Figure 6. In both cases of partly tilted and extremely tilted will count as unbalanced in layer 0.

The process in layer 0 is to determine if the dog is in a balanced or unbalanced state. The behaviours that will be decided based on those states are to either stop moving and lay down or stand up as described in Figure 14.

![Figure 14 - Layer 0](image)

The input sensors into the system are the mercury tilt-switches and the actuators are the pneumatic cylinders. Between the inputs/outputs is the logic to determine what the dog’s action should be.

If the tilt sensors output any tilt signal the dog will be in an unbalanced state, in attempt to correct and balance itself the dog will stop moving if it is not already. The stop module will send a HALT signal to the
pneumatic cylinders effectively stopping all movement of the dog, the dog will proceed to lay down sending its commands to the pneumatic cylinders.

The idea of the lay down command is to suspend the dog in the air using its suspension system. By doing so the dog will become balanced without physically assisting the dog, unless the dog is in an unsalvageable state.

To determine whether the dog is in a balanced state, the sensors will indicate whether the dog is balanced. If balanced the action “stand-up” is sent to the actuators to position the dog in a ready to walk state.

6.2 Layer 1

The first layer is an extension of layer 0 as the balance is needed to operate the dog in optimal conditions; firstly, without the balance, manoeuvring the dog is a safety hazard and can cause damages to the equipment and its surroundings. Therefore, the balance is of utmost importance, when the system is balanced the system will transition from the zeroth layer to the first layer. The state of the position will most likely be either: dog balanced and suspended in the air or dog balanced and in a standing position.

The purpose of the first layer is to generate a pre-determined walking sequence and apply that sequence to the dog’s pneumatic cylinders based of the status of the tilt and feet sensors as shown in Figure 15.
The sensors used in layer 1 are the mechanical feet switches that measure two states, touching the ground or off the ground. Combining the balance state and the feet sensors the system can determine whether the dog is in a lay-down action or a stand-up action. Otherwise if the dog becomes unbalanced during any point, the system will default to the “Stop moving” module stopping the pneumatic cylinders and then lay-down to re-balance itself.

The “Stand Up” module relies on either the feet sensors or the tilt-sensors. Once it is determined that the dog is off the ground, the action “Stand Up” command will be sent to the pneumatic cylinders. The “Stand Up” module uses either one or both sensors to perform its actions, this will increase the robustness of the system. If one of the sensors fails the dog will stand up if its condition(s) are met, by doing so the dog is already in an ideal state to walk. The possible conditions are shown in Table 4.
### Table 4 - Layer 1 stand-up outcomes

<table>
<thead>
<tr>
<th>Outcome</th>
<th>Sensor Conditions</th>
<th>Result of Dog</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Both sensors working</td>
<td>Ideal</td>
</tr>
<tr>
<td>2</td>
<td>Both sensors not working</td>
<td>System is unbalanced, and will be in a “Lay-down” position</td>
</tr>
<tr>
<td>3</td>
<td>Only tilt-sensors</td>
<td>The dog is suspended in the air</td>
</tr>
<tr>
<td></td>
<td></td>
<td>The dog is already standing up</td>
</tr>
<tr>
<td>4</td>
<td>Only feet sensors</td>
<td>The dog is suspended in the air</td>
</tr>
<tr>
<td></td>
<td></td>
<td>The dog is on its side</td>
</tr>
</tbody>
</table>

Outcome 1 will result in an ideal position for the dog. Once the dog is in a stand-up position the dog will proceed to walk. The signal from the walk module then inhibits the stand-up signal replacing the stand-up commands with the walk commands. The commands are then sent to the pneumatic cylinders for action.

Outcome 2 is the least ideal situation, as the dog’s tilt-sensors will register as being tilted resulting in the dog positioning itself in a lay-down position. Due to safety concerns the dog should not move in any case without any specific controls guiding its action behaviours.

Outcome 3 with only using the tilt-sensors the dog can determine whether it is in a balanced or unbalanced state, if the dog is unbalanced the dog will position itself in a lay-down position and stop moving until it is re-balanced. If the dog is in a balanced state initially the dog could be in the positions “Stand-up” or “Lay-down”. Regardless of those states the dog will perform the stand-up action, send a flag to the “Walk” module and proceed with the walk action on the next iteration.

Outcome 4 with only feet sensors can result in two possible outcomes; the dog is on its side and it becomes unbalanced or suspended in the air and will stand. If the dog is on its side, the stand-up
commands will still be performed. Except the dog will be on its side when it occurs, and if the feet sensors still result in “Off-Ground” the dog will remain in a stand-up position. Otherwise the dog is presumably already in its correct position to walk.

The outcomes are based of the most logical scenarios that could possibly occur. The idea of using a balance and feet sensor is to increase its robustness. The dog will maintain its ability to walk if one of the two sensors fails.

Once the dog determines that it is in a balanced condition either through feet sensors or tilt-sensors the dog will proceed to its walk action. The walk module inhibits the stand-up commands replacing the commands with the walk-sequence. The commands are then sent to the pneumatic cylinders.

6.3 Layer 2

Layer 2 is an extension of both layer 0 and layer 1. The main goal for layer 2 is to allow the dog to sense a type of surface the dog is walking on. The idea is to allow the dog to perform a different walking action depending on whether the surface is a raised or lowered ramp. The types of walking actions to be displayed are a slow or fast paced walking action depending on the obstacle. Unfortunately, there is no robustness in this layer as it is dependent on both the feet and tilt-sensors.

To transition into Layer 2, the dog must have all its feet sensors activated and the tilt sensors are no longer in a balanced state. The tilt-sensors have three possible states, to expand upon how layer 0 determines its balance state. Layer 0 operates on a balanced or not balanced, therefore the system only regards partly tilted and extremely tilted as unbalanced. Whereas for Layer 2 both partly tilted and extremely tilted are states that will be used to determine whether the dog is on a raised or lowered ramp as shown in Figure 16.
Layer 0 and Layer 2 have conflicting arguments over whether the system is balanced or not. From Figure 16, Layer 2 suppresses certain aspects such as the “Stop Moving” module, by doing so the dog will not be able to perform the actions “Lay Down”. In place the tilt-sensors will be used to determine if the ramp is a small or large incline or decline.
Table 5 shows the possible outcomes from the sensor readings.

**Table 5 - Layer 2 Outcomes**

<table>
<thead>
<tr>
<th>Outcome</th>
<th>Sensor Conditions</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Extremely Tilt and Feet on ground</td>
<td>Walking Up/Down-ramp, fast walking action</td>
</tr>
<tr>
<td>2</td>
<td>Partly Tilt and Feet on ground</td>
<td>Walking Up/Down-ramp, fast walking action</td>
</tr>
<tr>
<td>3</td>
<td>Feet off ground</td>
<td>Unbalanced, revert to Layer 1</td>
</tr>
<tr>
<td>4</td>
<td>Tilt-sensors balanced</td>
<td>Revert to layer 1</td>
</tr>
</tbody>
</table>

Outcomes 1 and 2 will result in either a slow-paced walk or fast-paced walk depending on the conditions of the ramp, regardless of whether the action is a slow or fast walk both signals will inhibit the “Walk” action in layer 1, replacing the “Walk” commands with the “slow” or “fast” paced walking commands.

Outcomes 3 and 4 occur when the dog has reached a balanced surface or if the dog falls from the surface and is no longer in a balanced state. If the tilt-sensors register a balanced signal the control system will transition from Layer 2 to Layer 1 and proceed with the normal walk routine until the dog reaches its next obstacle. If the feet sensors are no longer registered as “On ground” the dog is presumably unbalanced. Since Level 2 has suppressed the “Stop Moving” module the dog is unable to determine if the dog is in a controlled state of unbalanced or out of control state. Therefore, the feet sensors are a good indication of whether the dog is in control or not, if all feet sensors are not active the dog will revert to Layer 1 where the dog will either be balanced and off ground or unbalanced and off ground.
7.0 Programming

The programming language used is Forth using the SwiftX Environment [1] [2]. Through this chapter the code will be in the form of Structured English, for a detailed view of the program code refer to Appendix D – Forth Program [25].

Structured English uses descriptive words structured in a programming model, the code is not directly able to be implemented in any specific programming language but displays the concepts of what the program is trying to achieve [25].

Before attempting to implement a control structure or begin implementing a dog walking cycle, the commands to control the inputs and outputs needed to be tested and documented.

As part of the rebuilding aspect of the project, common commands to actuate the pneumatic cylinders, readings from the tilt sensors, foot sensors and potentiometers needed to be tested and documented (refer to 12.4.1 Forth Sensors). The existing code from past students provided information and comments about these functions. Most of the commands have already been implemented and attached to certain words in the program.

The words that enabled the functionality of the dog did not work correctly at times due to minute changes in the hardware and other parts of the program. To preserve the code that was previously written, the words were adapted to construct new words that achieved the same or similar functionality.

7.1 Dog walking cycle

The sequences from Table 2 and 3 were applied to the dog’s back-legs and front-legs respectively using timers to confirm that the sequence is correct. Between each limb movement there is a 500 ms wait. These waits are needed otherwise all the dog’s limbs would actuate at the same time. Certain parts of the sequence did not require waits as the goal was to allow the actuation of certain limbs to occur at the same time. The actions are not separated per leg but are separated to what was occurring in each step.
of the sequence. One dog walking cycle contains 4 steps, where one step is to position the dog's leg and the other is to propel the dog forward.

To test the limbs, the dog walking cycle was applied to the right-back leg and left-front leg initially. Minor adjustments were made to the sequence to actuate a third leg, this was to provide support for the landing of the dog. Once the limb's movements have been confirmed the sequence is mirrored for the other side of the dog.

: WalkSequenceOne

- Extend Right-backleg Hip
- Extend Left-frontleg Hip
- Contract Right-backleg Foot 500 ms wait
- Extend Left-frontleg Thigh 500 ms wait
- Contract Right-backleg Thigh 500 ms wait
- Extend Left-frontleg Foot 500 ms wait
- Extend Right-backleg Foot and Shank ;

Figure 17 - WalkSequenceOne

: WalkSequenceTwo

- Extend Left-frontleg Shank 500 ms wait
- Contract Left-frontleg Thigh
- Extend Right-frontleg Foot and Thigh
- Extend Left-frontleg Thigh
- Extend Right-backleg Thigh 500 ms
- Contract Left-frontleg Thigh 500 ms
- Contract Left-frontleg Foot and Shank ;

Figure 18 - WalkSequenceTwo

Figure 17 and 18 describe the program code of the limb sequence for the legs actuated in the first two sequences; the next two sequences use the same concept except the limbs are mirrored for a total of four sequences.

Once the dog walk cycle was completed and working correctly, to smooth out the movement of each limb, a combination of the pressure into the pneumatic cylinders, potentiometer readings to replace the wait timers and a Pulse-Width Modulation (PWM) signal was implemented into the walking cycle.
7.2 Pulse-Width Modulation and Potentiometers

The easiest way a PWM signal can be implemented in Forth and SwiftX is to use the timer function to turn the outputs off and on for a certain amount of time [26]. The PWM signal will turn the outputs on for 10 milliseconds and turn the outputs off for 30 milliseconds for a total period of 40 milliseconds. By using PWM the solenoid’s extension and contraction will result in slower pneumatic cylinder movements.

The potentiometer limits are used to replace the existing wait timers; the potentiometers will allow the limbs to actuate once the potentiometers reach a certain value. The potentiometers measure the angle of the position of the limb to determine whether the pneumatic cylinder has contracted or extended. Each potentiometer has a different value for its low-limit and high-limit as the angles are not all the same.

```
Begin
    PWMON ms wait 0 Foot Limb PWMOFF ms
    Contract/Extend Foot limb
    until LimbPotentiometerReading = LimbPotentiometerValue ;
```

*Figure 19 - Begin-Until Loop*

Figure 19 is an example of a begin-until loop used in Forth. A loop is needed to be able to fully extend and contract a limb using a PWM signal. Throughout each iteration of the loop, the limb’s actuation will be turned off (neither contracting nor extending) by writing 0 to the outputs associated with the leg for a PWMOFF time. In turn the specific outputs will be turned on for PWMON time eventually turning on and off the outputs in quick succession generating a square-wave as shown in Figure 20.
With loops, if conditions are not met, the program will be unable to exit the loop resulting in an endless loop [27]. The potentiometer readings and values are the exit conditions for the begin-until loop. The value of the potentiometer does not stay the same for every joint; there is a reasonable amount of variability in the readings for all sensors. To avoid the error of an infinite loop due to error readings of the potentiometers, the high-value and low-value are turned into a range of values.

The begin-until loop generating a PWM signal and polling the potentiometer for the associated limb were used to replace with every 500 ms wait that was used to test the walking sequence of the dog. The result is the dog moving more fluidly due to the slower actuations of the limb. However since there are no waits the sequences occurs quicker since they reach the limits assigned on the potentiometers to move to the proceeding steps the movements are quicker due to using potentiometers to transition to the next step.

**7.3 Tasks**

Tasks are a type of asynchronous processes that runs separately from the main process, the implementation of tasks were to be able to stop the dog from moving if the dog becomes unbalanced due to safety reasons and so the components of the dog do not get damaged [28].
The two tasks tiltcheck and WalkDog (Figure 21 and 22 respectively) are to be run in parallel along-side the main program, tiltcheck is used to repeatedly poll the tilt-sensors every 100 milliseconds using a begin-again loop. If the dog does become unbalanced at any stage, the WalkDog task is suspended. By using suspend, the dog will stop instantly without finishing its sequence.

When activating the StartWalkDog function the dog will automatically go through its walking sequence. If the task is suspended by the tiltcheck task the only way to start the task again is through task halt then task activate or task resume. Task activate will start the walk sequence from the very start whereas task resume will start the walk sequence from where the task was suspended.

### 7.4 Subsumption Architecture

The implementation of Subsumption Architecture revolves around using multiple asynchronous tasks using a begin-again loop. The different tasks represent layers in the control system where a variable can determine if the layer should be activated or not. Within each layer is the logic controlled with nested if statements and actions that the robot can perform, the actions such as Stand Up, Lay Down, Walk are all

```plaintext
    task tiltcheck
        : StartTiltTask tiltcheck build tiltcheck activate
        begin
            if TiltSensorValues - DogUnbalanced
                WalkDog task suspend
                then
                endif
                100 ms
            again ;
    
    task WalkDog
        : StartWalkDog walkdog build walkdog activate
        begin
            WalkSequenceOne WalkSequenceTwo WalkSequenceThree WalkSequenceFour
            100 ms
        again ;
```

*Figure 21 - Tilt check task

*Figure 22 - Walk Dog Task*
decided through various sensors such as Tilt Sensors and Feet Sensors. All the layers are running concurrently, though only one task should be able to control the outputs.

Since the tasks are independent of each other and do not communicate with each other, there is often repetitive coding between each layer, for example layer 1 can contain aspects or everything of layer 0 in its task. As additional layers are added the tasks themselves will become bigger.

A variable called Layer is used to check if the current layer should be active or not, the only way to transition to different layers is through the Layer variable. If the Layer variable does not match what the task is looking for, then the task just waits 100 milliseconds without doing anything. Whilst Layer correctly corresponds to the value that the task is looking for, the task will proceed through all the functions and actions associated with the layer.

```plaintext
    task layer0
        : activate layer0 layer0 build layer0 activate
        begin
            if Layer = Layer0Flag
                checktiltsensors
                if TiltSensors = Unbalance
                    ActionLayDown
                endif
                if TiltSensors = Balance
                    ActionStandUp
                    Layer = Layer1Flag
                endif
            endif
            100 ms
            again ;
```

*Figure 23 - Task Layer0*

The initial value for Layer is Layer0Flag, the dog will automatically transition into Layer0 as the system is initialised, Layer0 is shown in Figure 23. The status of the tilt sensors will determine the actions of the dog. If the tilt sensors register an unbalanced signal the dog will perform the action Lay Down and remain in the layer. If the tilt sensors register a balanced signal the dog will perform the action Stand Up and transition into Layer 1.
Figure 24 - Task Layer1

Layer1’s (Figure 24) main purpose is to determine whether the dog is in a reasonable state to perform ActionWalk, once entering the task the layer checks whether the variable Layer corresponds to Layer1Flag. The tilt-sensors are checked to see if the dog is still in a balanced state from its transition from Layer0, if the dog is unbalanced the layer sets the variable Layer to Layer0Flag and exits the Layer, this is one of the exit conditions for Layer1.

Nested if-statements are used to determine the which action to perform in this case StandUp or Walk, nested if-statements are if-statements within if-statements [29] [28] [27]. An if-else statement is used in this case due to conflicting statements. In the situation where the tilt-sensors are balanced and feet sensors are all on the ground, both conditions for ActionWalk and ActionStandUp are satisfied. Therefore, an if-else-if statement is used as a switch since the action performed can be one or the other. Once the dog has performed ActionWalk the transition to Layer 2 is complete.
In Layer 2 (Figure 25) aspects of Layer 0 will need to be suppressed due to conflicting statements using the tilt-sensor, the balance/unbalance aspect is removed from Layer 2. If none of the conditions are met for ActionFastWalk or ActionSlowWalk the exit condition of go to Layer 1 is immediately met. The actions ActionFastWalk and ActionSlowWalk can be achieved by changing the PWMON or PWMOFF value to slowly or quickly actuate the pneumatic cylinders.

There are a variety of methods to implementing the Subsumption Architecture such as incorporating suspend/resume/activate task so that only one task is running at a time, this method will reduce memory usage. Since the implemented Subsumption Architecture is small the need to suspend/resume/activate tasks were not necessary. Alternatively having an overseeing task that can halt-activate tasks will lower response times of the dog if implemented correctly.
8.0 Results

8.1 Dog walking cycle

Attempting to apply a biologically correct walking cycle proved difficult as the limbs on the dog did not accurately reflect those of a biological dog; the limbs did not extend far enough to provide the dog with accurate movements. Many different alterations and adjustments were made to the sequence of the dog’s limb movement to accommodate the dog.

By using the dog back to front (back-legs are now front-legs and vice versa) the dog could fully extend its reach using the back-legs as its front legs, with modification and adjustments to the walking sequence, the dog is able to closely replicate the dog’s walking movement in Figure 12.

The back-legs are used to propel or thrust the dog forward as the front-legs are used to cushion/pad its fall; the movement was inspired by the dog walking gait and was closely referred to. The shank of the front-leg should extend as the leg cushions the fall. Unfortunately, when extending the shank, the dog will spring backwards due to extension of the shank taking too long. The shank is left contracted on the landing of the front-limb resulting in a rough/hard landing.

The walk gait was implemented first with timers to ensure that the sequence visibly looked correct. Often limbs did not extend one at a time but extended multiple limbs at the same time on different legs. The walk gait was first implemented with two legs, the right-back leg and the left-front leg. Once it was determined that the sequence was correct and propelled the dog forward, the sequence was mirrored for the other two legs, creating a symmetrical walking gait.

Implementing the PWM signal for each limb extension and contraction enhanced the fluidity and smoothness of the walking sequence. Unfortunately, the dog’s movement is still quite sharp and “clunky” though this could be solved by changing the pneumatic actuators to a hydraulic or a motor based actuator.
The timers were removed and replaced with potentiometer readings to determine whether the leg is fully contracted or fully extended as pneumatic cylinders do not offer any positional feedback. Each potentiometer or joint have their own specific reading/value that refers to a fully contracted limb or a fully extended limb. There are 12 potentiometers acting as joints on the dog, and often the potentiometers would not be facing the correct way, therefore a lot of the joints have various readings where high is equivalent to fully extended and low is equivalent to fully contracted and vice versa. The values are also different depending on how much the joint is rotated once the limb has contracted or extended. The values often appear with an error-range therefore a high-range and low-range are created to allow the small margin of error that is accompanied by sensors. The result of exchanging timers for potentiometers includes a smoother, responsive and quicker movement and actuation of the walking motion.

The problem with using potentiometers is the high-values and low-values are not consistent after every cycle, so it is recommended that the potentiometers are calibrated so the program does not lock-up. The program will lock-up due to the potentiometers never reaching their high-values or low-values due to the begin-until loops implemented.

8.1.1 Comparison

Shiqi Peng’s walking routine was governed by the Subsumption Architecture and Central Pattern Producer to determine the dogs walking behaviour and sequence. The implementation that Shiqi has incorporated and the current implementation of the walk are vastly different in how they operate.

As Shiqi Peng’s walking routine noticeably slips when the dog is walking, whereas the current implementation uses all available limbs to walk in a set routine, this routine cannot change. The status of the walking gait by the dog is smoother compared to Shiqi’s Dog though with an exception: the landing or “cushion” from the front-limbs are rough as the dog tends to place all its weight on the front legs as the dog is thrusting forward.
Shiqi’s dog components, specifically the pneumatic cylinders, contain greater actions per second. The dog appears to be moving faster but covering less distance. The current dog looks to provide greater traction on the floor to provide a greater “leap” forward when walking compared to Shiqi’s Dog. This is due to the use of limbs to propel itself effectively as it is moving forward. The dog’s walking sequence is designed to maximise the distance travelled in a sequential motion as opposed to Shiqi’s dog where each leg is independent from each other due to four Subsumption Architectures in parallel and is only governed by a set step sequence without the goal of maximum movement per step cycle, only that it must move intelligently.

### 8.2 Subsumption Architecture

The layers are implemented as separate parallel tasks with flags that toggle between each layer, therefore the Subsumption Architecture that is implemented successfully transitions between each layer depending on the sensor readings. Overall the dog can determine the actions needed depending on which layer the controller is currently in. The dog can successfully balance itself through unconventional methods such as using its suspension apparatus and can determine whether the requirements to walk have been met. The robustness implemented by Layer 1 has also proved to be successful as the walk action could be performed with limited sensors.

#### 8.2.1 Layer 0

The Layer 0 implementation could determine whether the dog is in a balanced state, or in an unbalanced state by using the tilt sensors as described in chapter 6.1 Layer 0.

The actions of Layer 0 are fast and responsive due to the 100 ms wait in the task to control the layer. The movements of the dog is quite fast and erratic, when the dog is performing the lay down action the dog is able to successfully balance itself by using the suspension apparatus that was designed for the dog. Once the dog is standing up, and all the foot sensors are registered as being on the ground, the transition to Layer 1 is complete.
8.2.2 Layer 1

The Layer 1 implementation is to determine whether the dog is in the correct orientation and condition to walk. The dog itself can determine whether to start walking or not, by using a combination of both the tilt sensors and the foot sensors a robust method to determine the correct orientation and walking conditions was created.

It is confirmed that if the dog only has the tilt sensors in a working condition and registers a balanced signal, the stand-up action will take place and it will begin to walk. Additionally, if the dog only has the foot sensors in a working condition, the dog will stand-up and if the dog does not register all four feet on the ground it will stop moving.

Unfortunately, with only the foot sensors working the dog will stand-up after every walking cycle, as the end of the walking cycle does not require the dog to have all 4 feet on the ground. Though with limited functionality, the dog will walk for one cycle, stand up and repeat during Layer 1.

8.2.3 Comparison

Shiqi Peng uses the Subsumption Architecture to determine the stance of each limb, the behaviours that are implemented are Stand, Step, Balance and Leg Down. The control system implemented in each leg totals four Subsumption Architectures in parallel. Since each Subsumption Architecture is governed by a Central Pattern Producer, this allows the legs to actuate at specific intervals and in order.

Both implementations of Subsumption Architecture are to control the behaviour of the dog depending on the sensors; Shiqi’s implementation is to allow the leg to determine the behaviour. Thus, each leg can be display a different behaviour, such as leg 0 can output a Stand behaviour, leg 1 outputs a Balance behaviour and so on [30]. Whereas the current implementation the action sequences (walk) are already pre-determined allowing little change to its current walking procedures. The only difference is that the different actions that are determined by the sensors will apply to the whole robot as opposed to separate legs of the robot.
Shiqi Peng’s design is an elegant solution compared to the current control system; the different behaviours complement each other which will allow the robot to move fluidly since each leg can determine its own behaviour. Whereas the latter will output a set action sequence, whether the ground is not completely uniform. The dog may run into problems if the environment is not an ideal area.

As for future expandability of the Subsumption Architecture, Shiqi Peng’s design is only limited to different walking behaviours. The current control system will allow further expandability into different behaviours in general, not limited to walking behaviours for example avoid, wander and push as more sensors are introduced into the system.
9.0 Issues Encountered and Solutions

9.1 Dog walking cycle

After first applying the walking sequence the limbs did not extend far enough to replicate the walking gait or the limb movements as shown in Fig 11. To achieve the same limb sequence and similar limb movements the dog’s body was flipped around effectively walking the dog backwards. By using the back-legs as the front-legs and the front-legs as the back-legs the limb movements could be replicated as per the illustrations.

The attempt at implementing the gait “Trot” was trialled, the issue faced was that the dog could not balance itself on only two feet with moving parts; therefore this gait was not used.

The first attempt to walk the dog was to replicate the 9 sequences shown in Figure 11, moving the limbs as shown. Unfortunately, the movement was too slow or too fast and would cause the dog to become unstable and unbalanced. The dog would often fall over, to resolve this issue the dog’s hips would always be extended to provide support and stability.

Shiqi Peng’s implementation of walking involved moving one leg at a time, the first implementation of the walking gait and the illustrated figure caused instability of the dog, becoming more unstable the further into the sequence. By moving two or more legs at the same time the dog can better maintain its balance and centre of gravity.

To provide a fluid motion when walking, the dog’s front shank should be extended when landing. Unfortunately, the shank is quite long and only pushes the dog backwards when the shank is extended at this stage, to counter-act the problem the shank remains contracted, this results in a rough landing.

A more fluid motion can be provided by changing the pneumatic cylinders with an actuator that is precision-based such as hydraulics and motors. The pneumatic cylinders used in the dog only provide two positions, fully extended or fully contracted. The cylinders are unable to stop between fully
extended and fully contracted; by changing the cylinders the dog can precisely control its limb offering an expanded library of behaviours and movement.

Due to the rough motion of walking, multiple components of the dog have been damaged, components such as feet sensors tend to break on contact with the ground, the digital isolation board becomes faultier with every use of the dog due to micro-fractures in the solder and limbs tend to break and fall apart. As the dog is moving the vibrations caused by the motion also causes screws and nuts to become loose, also irregular movement of the legs will cause stress on the hinges on the dog causing them to break as shown in Figure 26.

![Figure 26 - Robo-dog Broken Hinges](image)

A solution to the issue is to create secure fixtures for each component of the dog, designing different type of feet sensors and converting the digital isolation board to an etched printed circuit board.
9.2 Subsumption Architecture

The implementation of Layer 2 is not completed due to time constraints and tilt-sensors. The tilt-sensors original design will not be compatible with Layer 2. This is due to the lack of the sensor’s sensitivity, to register a partly-tilt the mercury switches (1, 2, 3 or 4) will need to be at approximately 45-degrees angle. To register a extremely-tilt the mercury switches (5, 6, 7, 8) will need to be at an angle greater than approximately 70 degrees. Unfortunately to compound the issue the dog will lose grip when attempting to walk on a surface at that angle.

Replacing the current mercury switches tilt-sensors with other available sensors such as accelerometer/gyroscope will greatly benefit the dog. This will allow the dog to monitor its balance precisely as the current mercury switches are not sensitive enough for the Layer 2 implementation.

The “Stop Now” action on the dog does not stop it during its walking sequence, the action will only stop after the walking sequence has completed. This was due to a programming error where the parallel task to control the movement of the dog was started incorrectly. The task runs on an infinite loop constantly checking the layer state, when the walk action is needed the task to start the walk is started every 100 milliseconds. The result is that the parallel task that has been started, the next 100 milliseconds through the loop will result in the task starting again. Due to this error the program will ‘lock up’ and proceed to become unresponsive.
10.0 Summary

The project’s objective is to restore functionality to the dog, walk the dog from Point A to point B using a walking strategy and implementation of a control system in the form of Subsumption Architecture. As a part of the rebuilding aspect of the program initial issues ranging from communications, hardware and various codes were difficult to debug due to lack of documentation. To remedy the situation the components were debugged and fixed whilst documentation was created in the form of circuit diagrams, comments in the program and documents. The documentation can be found in Appendix A - Limbs, Appendix C – Digital Isolation and Appendix D – Forth Program.

Once restoring the hardware, communications and various parts of the program was completed, the dog could be freely controlled. The movement sequence for the dog was derived from the walk gait and the limb-movement associated with each step of the walk gait. The walk was modified and adjusted for application to the robotic dog. The design implemented was to maximise the forward movement per cycle while incorporating a sequential approach where each leg’s movement were dependant on others.

The Subsumption Architecture was implemented to allow the dog to determine the behaviours, actions and movements to be presented. A basic implementation that consists of: determining whether the dog is a balanced state, to walk when all the conditions have been met and to display different walking behaviours based on the type of surface.

The implementation of Level 2 was unsuccessful due to the current tilt-sensors because they were not sensitive enough to determine a partly-tilted or extremely-tilted ramp.

Since the basic ideas of implementing Subsumption Architecture are now included in the program, the control structure will be able to be expanded. As the program grows it is recommended that the program for the Subsumption Architecture should be modified to incorporate task resume/suspend/activate to speed up the responsiveness and prevent memory and processing issues.
Various issues were encountered throughout the project such as hardware deteriorating to an extent that the components were breaking. Unfortunately, time did not permit to fix those issues, although have been addressed in future works.
11.0 Future Works

The robo-dog is in an inoperable state due to various issues, these issues are: digital isolation board and foot sensors. The issues will need to be addressed to allow development of the dog to continue. Adding additional sensors to the dog, will allow the dog to gain more functionality to be used in its Subsumption Architecture.

11.1 Digital Isolation Board

Many issues persist with the digital isolation board in the form of shorts, soldering and dry joints (micro-fractures in solder). Often outputs will not be responsive or work at all; unfortunately movements caused by the dog may cause additional micro-fractures. To rectify the issue, it is recommended etching the Digital Isolation Board to a Printed Circuit Board (PCB) with Integrated Circuit (IC) Bays to easily swap out faulty ICs. An up-to-date circuit diagram has been created to accurately reflect the purpose and the current digital isolation board refer to Appendix B – Digital Isolation Board.

11.2 Foot Sensors

Due to the design of the foot sensors, the impact and stress caused by the movement of the dog the switches and its enclosure tends to break over-time. Changing the design of the foot sensors to offer a robust design is ideal. Different sensors such as pressure sensors could be used instead of mechanical switches, although not essential as the mechanical switches serves its purpose by detecting the ground. Incorporating pressure sensors will allow the dog to determine how much force is applied to a specific leg.

11.3 Tilt Sensors

The current tilt-sensors are in the form of mercury switches operate well, although exchanging the tilt-sensor for an accelerometer/gyroscope based sensor will prove beneficial to the dog. The dog will be able to accurately determine the degree of the tilt. An accelerometer based sensor has been made for the dog by a previous thesis student but unfortunately the sensor has been misplaced.
11.4 Additional Sensors

Adding different sensors such as ultrasonic and infrared will allow the functionality of the dog to expand. With the introduction of new sensors the Subsumption Architecture will be able to be extended from its current three layers. Sensors such as an ultrasonic sensor will allow the dog to determine if there is an object in close proximity, a variety of behaviours can be decided by an ultrasonic sensor.

11.5 Programming

Improvements can be made to the program to incorporate additional layers to the Subsumption Architecture. As sensors are introduced the functionality and layers of the architecture can expand. Behaviours such as walking up a step can be implemented by determining if the potentiometer value is low and if the foot-sensor is active, means that the dog is currently on a step.

Improvements to the code such as incorporating task resume/suspend/activate will future proof the current implementation of Subsumption Architecture. Creating an updating potentiometer array will allow the dog to determine the errors in potentiometers and which ranges to stay within calibration.

Additional implementation of different gaits will prove to be useful; the current program incorporates the gaits walk, slow walk and fast walk. The dog is capable of the walking gait but the fast-paced gaits were not feasible with the current design. However with more sensitive hardware the test and implementation of gallop and trot may be possible.

Alternatively, it is possible to salvage parts of Shiqi Peng’s code, of most importance is the implementation of Subsumption Architecture, Reinforced Learning and the Central Pattern Producer. Salvaging and altering parts of the program will restore Shiqi Peng’s robo-dog to its former capability.

11.6 LCD and Keypad

Incorporating an LCD and Keypad to offer a friendly interface to users for demonstration and education purposes will be valuable; the NMIS 7070 board is readily available for implementation. The keypad can toggle between different walking behaviours or whether to run the dog at idle or use Subsumption
Architecture. An implementation of a LCD screen will allow users to identify which layer and walking behaviours that the dog is currently operating in.
### Appendices

#### 12.3 Appendix A – Limbs

*Table 6 - Limbs bit configurations*

<table>
<thead>
<tr>
<th>leg 0</th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>contract hip</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>extend hip</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>contract foot</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>extend foot</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>contract shank</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>extend shank</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>contract thigh</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>extend thigh</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>leg1</th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>contract hip</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>extend hip</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>contract foot</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>extend foot</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>contract shank</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>extend shank</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>contract thigh</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>extend thigh</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>leg2</th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
</table>

57
<table>
<thead>
<tr>
<th>leg3</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 0 0 0 0 0 0 0 0 contract hip</td>
</tr>
<tr>
<td>0 1 0 0 0 0 0 0 0 extend hip</td>
</tr>
<tr>
<td>0 0 1 0 0 0 0 0 0 contract foot</td>
</tr>
<tr>
<td>0 0 0 1 0 0 0 0 0 extend foot</td>
</tr>
<tr>
<td>0 0 0 0 1 0 0 0 0 contract shank</td>
</tr>
<tr>
<td>0 0 0 0 0 1 0 0 0 extend shank</td>
</tr>
<tr>
<td>0 0 0 0 0 0 1 0 0 contract thigh</td>
</tr>
<tr>
<td>0 0 0 0 0 0 0 1 0 extend thigh</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>contract</th>
</tr>
</thead>
<tbody>
<tr>
<td>hip</td>
</tr>
<tr>
<td>foot</td>
</tr>
<tr>
<td>shank</td>
</tr>
<tr>
<td>thigh</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>1 0 0 0 0 0 0 0 0</th>
<th>contract</th>
<th>hip</th>
</tr>
</thead>
<tbody>
<tr>
<td>0 1 0 0 0 0 0 0 0</td>
<td>extend</td>
<td>hip</td>
</tr>
<tr>
<td>0 0 1 0 0 0 0 0 0</td>
<td>contract</td>
<td>foot</td>
</tr>
<tr>
<td>0 0 0 1 0 0 0 0 0</td>
<td>extend</td>
<td>foot</td>
</tr>
<tr>
<td>0 0 0 0 1 0 0 0 0</td>
<td>contract</td>
<td>shank</td>
</tr>
<tr>
<td>0 0 0 0 0 1 0 0 0</td>
<td>extend</td>
<td>shank</td>
</tr>
<tr>
<td>0 0 0 0 0 0 1 0 0</td>
<td>contract</td>
<td>thigh</td>
</tr>
<tr>
<td>0 0 0 0 0 0 0 1 0</td>
<td>extend</td>
<td>thigh</td>
</tr>
</tbody>
</table>
12.1 Appendix B - EPROM

The existing EPROM currently on the microcontroller would first need to be erased; a device called an EPROM eraser is used. The EPROM is placed in the EPROM eraser where the window on the chip is exposed to ultraviolet light, allowing the charge of the floating gate to dissipate.

The next step is to reprogram or ‘burn’ the kernel onto the chip and this is achieved by deleting the current “TARGET.S19” file in the dog program and using the newest version of SwiftX to rebuild the target file. The EEPROM is then placed in an EPROM Programmer where the EPROM chip make and model is specified alongside the new TARGET.S19 file is then selected. Once applied the chip is then reprogrammed and ready to use [31].
12.2 Appendix C – Digital Isolation Board

Figure 27 - Digital Isolation Board (Circuit Diagram)
**12.4 Appendix D – Forth Program**

The appendix provides information about the Forth program implemented, only relevant parts of the program will be described as the program itself is quite large. The following segments of code describes the commands to the actuators, testing of sensors, testing of walk sequence, implementation of walk sequence and implementation of Subsumption Architecture.

\ defining legs and legports

: RBLEG rightbackleg legport c! ;

: RFLEG rightfrontleg legport c! ;

: LBLEG leftbackleg legport c! ;

: LFLEG leftfrontleg legport c! ;

**12.4.1 – Forth Sensors**

\ loop to check the foot sensors

: groundtouchstatusloop begin groundtouchstatus . 100 ms key? until ;

\ loop to check tilt sensors

: bodybalancestatusloop begin bodybalancestatus . 100 ms key? until ;

\ checks all 16 analogue inputs

: checkallpotentiometers [ hex ] 10 0 do i rawposition [ decimal ] 100 ms decimal . hex loop decimal ;
12.4.2 – Forth Test Walk Sequence

: walkies1

%00000001 rightbackleg legport c!

%00000001 leftfrontleg legport c!

%10000000 rightbackleg legport c! 500 ms

%00000100 leftfrontleg legport c! 500 ms

%00001000 rightbackleg legport c! 500 ms

%01000000 leftfrontleg legport c! 500 ms

%01010000 rightbackleg legport c! ;

: walkies2

%00010000 leftfrontleg legport c! 500 ms

%00001000 leftfrontleg legport c!

%01000100 rightfrontleg legport c!

%00000100 leftfrontleg legport c!

%00000100 rightbackleg legport c! 100 ms

%00001000 leftfrontleg legport c! 500 ms

%10100000 leftfrontleg legport c! ;
: walkies3

%00000001 leftbackleg legport c!
%00000001 rightfrontleg legport c!
%10000000 leftbackleg legport c! 500 ms
%00000100 rightfrontleg legport c! 500 ms
%00001000 leftbackleg legport c! 500 ms
%01000000 rightfrontleg legport c! 500 ms
%01010000 leftbackleg legport c! ;

: walkies4

%00010000 rightfrontleg legport c! 500 ms
%00001000 rightfrontleg legport c! \ added
%01000100 leftfrontleg legport c!
%00000100 rightfrontleg legport c!
%00000100 leftbackleg legport c! 100 ms
%00001000 rightfrontleg legport c! 500 ms
%10100000 rightfrontleg legport c! ;

: testallwalk 2 0 do walkies1 500 ms walkies2 500 ms walkies3 500 ms walkies4 500 ms loop ;
12.4.3 – Forth Test PWM and Potentiometer walk implementation

: PWMwalkies1

%00000001 RBLEG

%00000001 LFLEG

\ begin PWMON 0 RBLEG PWMOFF %10000000 RBLEG 9 rawposition . 9 rawposition 111 > key? OR until

%10000000 rightbackleg legport c! 100 ms

begin PWMON 0 LFLEG PWMOFF %00000100 LFLEG 3 rawposition dup . 5 < key? OR until

begin PWMON 0 RBLEG PWMOFF %00001000 RBLEG 11 rawposition dup . 90 > key? OR until

begin PWMON 0 LFLEG PWMOFF %01000000 LFLEG 1 rawposition dup . 65 < key? OR until

\ begin PWMON 0 RBLEG PWMOFF %01010000 RBLEG 9 rawposition dup . 110 < 8 rawposition dup .

130 < and key? OR until ;

begin PWMON 0 RBLEG PWMOFF %01010000 RBLEG 8 rawposition dup . 140 < key? OR until ;

\ THIS IS WRONG NEEDS TO BE MORE MOVING PARTS AT THE SAME TIME FOR PWMWALKIES2 REFER

TO WALKIES2

: PWMwalkies2

begin PWMON 0 LFLEG PWMOFF %00010000 LFLEG 0 rawposition dup . 140 > key? or until

%00001000 LFLEG

%01000100 RFLEG

%00000100 LFLEG

64
%00000100 RBLEG 100 ms

%00001000 LFLEG 500 ms

%10100000 LFLEG ;

: PWMwalkies3

%00000001 LBLEG

%00000001 RFLEG

begin PWMON 0 LBLEG PWMOFF %10000000 LBLEG 15 rawposition dup . 65 < key? or until

begin PWMON 0 RFLEG PWMOFF %00000100 RFLEG 5 rawposition dup . 45 > key? or until

begin PWMON 0 LBLEG PWMOFF %00001000 LBLEG 13 rawposition dup . 77 < key? or until

begin PWMON 0 RFLEG PWMOFF %01000000 RFLEG 7 rawposition dup . 65 > key? or until

begin PWMON 0 LBLEG PWMOFF %01010000 LBLEG 15 rawposition dup . 70 > 14 rawposition dup . 75 > and key? or until ;

: PWMwalkies4

begin PWMON 0 RFLEG PWMOFF %00010000 RFLEG 6 rawposition dup . 63 > key? or until

%00001000 rightfrontleg legport c! \ added

\ %00000100 rightfrontleg legport c!

%01000100 leftfrontleg legport c!
%00000100 rightfrontleg legport c!

%00000100 leftbackleg legport c! 100 ms

%00001000 rightfrontleg legport c! 500 ms

%10100000 rightfrontleg legport c!

: letswalk pwmwalkies1 pwmwalkies2 pwmwalkies3 pwmwalkies4 ;

12.4.4 – Forth PWM and Potentiometer implementation

\ ================TASK WALKING SEQUENCES===============

: move1

%00000001 RBLEG

%00000001 LFLEG

\ begin PWMON 0 RBLEG PWMOFF %10000000 RBLEG 9 rawposition . 9 rawposition 111 > key? OR until

%10000000 rightbackleg legport c! 100 ms

begin PWMON 0 LFLEG PWMOFF %00000100 LFLEG 3 rawposition 5 < until

begin PWMON 0 RBLEG PWMOFF %00001000 RBLEG 11 rawposition 90 > until

begin PWMON 0 LFLEG PWMOFF %01000000 LFLEG 1 rawposition 65 < until

\ begin PWMON 0 RBLEG PWMOFF %01010000 RBLEG 9 rawposition 110 < 8 rawposition 130 < and until

;

begin PWMON 0 RBLEG PWMOFF %01010000 RBLEG 8 rawposition 140 < until ;
: move2

begin PWMON 0 LFLEG PWMOFF %00010000 LFLEG 0 rawposition 140 > until
%00001000 LFLEG
%01000100 RFLEG
%00000100 LFLEG
%00000100 RBLEG 100 ms
%00001000 LFLEG 500 ms
%10100000 LFLEG ;

: move3

%00000001 LBLEG
%00000001 RFLEG

begin PWMON 0 LBLEG PWMOFF %10000000 LBLEG 15 rawposition 65 < 5 ms until
begin PWMON 0 RFLEG PWMOFF %00000100 RFLEG 5 rawposition 45 > until
begin PWMON 0 LBLEG PWMOFF %00001000 LBLEG 13 rawposition 77 < until
begin PWMON 0 RFLEG PWMOFF %01000000 RFLEG 7 rawposition 65 > until
begin PWMON 0 LBLEG PWMOFF %01010000 LBLEG 15 rawposition 70 > 14 rawposition 75 > and until ;

: move4
begin PWMON 0 RFLEG PWMOFF %00010000 RFLEG 6 rawposition 63 > until

%00001000 rightfrontleg legport c! \ added

\ %00001000 rightfrontleg legport c!
%01000100 leftfrontleg legport c!
%00000100 rightfrontleg legport c!
%00000100 rightfrontleg legport c!
%00000100 leftbackleg legport c! 100 ms
%00001000 rightfrontleg legport c! 500 ms
%10100000 rightfrontleg legport c! ;

: movefinal move1 move2 move3 move4 ;

12.4.5 – Forth Implemented tasks
\ task to constantly check the tilt sensors, need to run checktilt

variable tiltchecker

variable tiltflag

: unbalancecheck tiltchecker c@

dup %00111111 = if tiltflag c! else
dup %11001111 = if tiltflag c! else
dup %11110011 = if tiltflag c! else
dup %11111100 = if tiltflag c! else tiltflag c! then then then then drop ;
task tilt

: checktilt tilt build tilt activate begin bodybalancestatus tiltchecker c! unbalancecheck 100 ms again ;

\ task to begin moving

variable startmoving

task alwaysmoving

: letsgo alwaysmoving build alwaysmoving activate begin startmoving

  c@ 1 = if movefinal 500 ms then 100 ms again ;

\ task to stop moving

task dogunbalanced

: stopmoving dogunbalanced build dogunbalanced activate

  begin tiltflag c@ 1 = if alwaysmoving suspend then 100 ms again ;

\ reinitialises the move once the do has stopped

: reintmove tiltflag c@ 0 = if alwaysmoving resume then ;

\ task to constantly check the foot sensors, need to run checkfeet

variable feetchecker

task feetcheck
: checkfeet feetcheck build feetcheck activate begin groundtouchstatus feetchecker c! 100 ms again ;

: runwalkingtasks checktilt letsgo stopmoving ;

: gogogo 1 startmoving c! ;

: stopnow 0 startmoving c! ;

: tasknotes ." START TASKS - runwalkingtasks " cr ;

: movementnotes ." START MOVEMENT - gogogo (flag on) | STOP MOVEMENT - stopnow (flagoff) " cr ;

: stopwalkingtasksnotes ." REINITIALISE MOVE TASK - reintmove " cr ;

12.4.5 – Forth Subsumption Architecture

\ subsumption implementation

variable layercheck

variable subtiltcheck

variable subtiltflag

variable walkflag

\ checks the balance status

: unbalancechecksbodybalancestatus %11111111 = if 0 tiltflag c! else 1 tiltflag c! then ;
task dogmove

: walkdog dogmove build dogmove activate begin qqq 500 ms again ;

\ walkdog suspend 0 walkflag c!

: layer0modules tiltflag c@ 1 = if laydown then

  tiltflag c@ 0 = if standup %1111111111111101 layercheck ! then ;

task layer0

: activatelayer0 layer0 build layer0 activate

begin layercheck @ %1111111111111110 = if unbalancechecksub layer0modules then 500 ms again ;

task layer1

: activatelayer1 layer1 build layer1 activate

begin layercheck @ %11111111111111101 = if unbalancechecksub tiltflag c@ 1 = if %1111111111111110
layercheck ! then tiltflag c@ 0 = feetchecker c@ 15 < and if standup then

  tiltflag c@ 0 = feetchecker c@ 15 = and if movefinal then then drop 500 ms again ;
Bibliography


