Control of a biomimetic robot lobster with a synthetic nervous system

by Daniel H. Blustein

B.A. in Biology, Kalamazoo College

A dissertation submitted to

The Faculty of
the College of Science of
Northeastern University
in partial fulfillment of the requirements
for the degree of Doctor of Philosophy

December 4, 2014

Dissertation directed by

Joseph Ayers
Professor of Marine and Environmental Sciences
Acknowledgements

First, I would like to thank my advisor, Dr. Joseph Ayers. Without his support and mentorship this work would not have been possible. His ambitious research vision motivated me to see the robot lobster to completion and the responsibilities he entrusted me with made the experience particularly formative in my academic training. And of course, his unmatched skills at smoking meat and preparing lobster bisque made it tolerable to stay in graduate school for over six years.

I would also like to thank my committee members for their willingness to serve and for the valuable feedback they provided: Dr. Richard Marsh, Dr. Donald O’Malley, Dr. Marsette Vona, and Dr. Win Watson. Each lent their expertise to further my efforts.

Many people contributed to the development of RoboLobster. An Australian-sized thank you goes to Dr. Anthony Westphal who paved the way for this approach to biomimetic robots with his RoboLamprey. RoboLobster’s electronics and nervous system code would not exist without his hard work. His friendship and support also helped me tremendously, from keeping late-night lab work light-hearted to filling the role of the token Aussie in short film productions.

Thank you to Dr. Jan Witting, a pioneering RoboLobster engineer, for his assistance in getting this version off the ground. To A. Mark Hunt for designing the DSP board. To Dr. Geof Barrows and Centeye, Inc for providing optical flow sensors and code. To the RoboBees project team, particularly Dr. Rob Wood, for an exciting collaboration.

And a huge thank you goes to the team of undergraduate and high school students who helped build and test the robots. A few who provided significant assistance are: Christopher Barreda, Alexander Giuliano, Jacob Ratliff, Ricky Rosario, Nikolai Rosenthal, Clint Valentine, Andy Vann, and Christopher Williams. I could not have completed this work without their help.
I would also like to thank my lab mates: Peter Brannen, Stephen Smith, Dr. Kazuo Mori, Lin Zhu, Lara Lewis McGrath, Ryan Myers and Benjamin Doyle, for their support and friendship.

Thank you to the funders of this work and of my training which include: the NSF Graduate Research Fellowship Program; Northeastern University’s Department of Biology, Marine Science Center, and Dissertation Completion Fellowship program; the AAAS Mass Media Fellowship program; the Marine Technology Society; Schlumberger, Inc.; NSF ITR Expeditions Grant 0925751 and ONR MURI grant N000141110725.

And finally, this work would not have been possible without the love, support and diversion of my friends and family. Special thanks go to my sister Emily, my parents, my Uncle David, my dear friends near and far, my roommates, and members of my volleyball teams. And thank you to Revere Beach, my home for five years, for putting it all in perspective.
Abstract of Dissertation

The biorobotic approach uses robots to simulate biology. This dissertation presents RoboLobster, a biomimetic underwater robot controlled by an electronic nervous system simulation. Using this robot we test neuron network hypotheses related to sensory processing, sensory fusion and walking pattern generation. With a design based on the body structure, biomechanics, and control architecture of the American Lobster, *Homarus americanus*, RoboLobster serves as a closed-loop biorobotic simulation of neuron network operation that can be run in unpredictable real-world environments. As a secondary contribution, we further investigate arthropod neuroethiology using a robotic bee controlled by a computational nervous system simulation.

RoboLobster features limb biomechanics similar to those of the model organism. Movement of the robot’s eight legs is accomplished by an excitation-contraction coupling scheme in which simulated neuron spikes activate shape memory alloy actuators, muscle analogues. Neuromorphic sensors mimic their biological counterparts and code information in the language of neurons: action potentials. The sensors onboard RoboLobster are optical flow sensors, antennal bend sensors, claw bump sensors, an accelerometer, and a gyroscopic-compass. The inputs from these sensors drive RoboLobster’s electronic nervous system which is comprised of mathematically modeled neurons and synapses arranged in network configurations based on the known neural circuitry of the lobster. The electronic nervous system incorporates a variety of simulated neurophysiological components including central pattern generators, command neurons, coordinating synapses, and layered exteroceptive reflex networks.

The platform is used to demonstrate plausible implementations of a variety of neurobiological hypotheses, including: lobster walking central pattern generation, arthropod
optical flow reflexes, the role of decussating neural connections driving behavior, corollary
discharge, and several proposed mechanisms underlying multimodal sensory fusion.
RoboLobster extends biological efforts to test neuroscience hypotheses and advances robotic
engineering efforts to develop alternative control architectures. This comprehensive biomimetic
platform opens the door to a wide range of computational neuroscience experiments and serves
as a vehicle capable of operating in the unpredictable conditions of littoral marine environments.
Table of Contents

Acknowledgements .................................................. ii
Abstract of Dissertation ........................................... iv
Table of Contents .................................................... vi
List of Figures ........................................................ vii
List of Tables ........................................................ xiii
Chapter 1: Introduction ............................................. 1
Chapter 2: Model organisms and computational methods ...... 14
Chapter 3: Central pattern generation .............................. 26
Chapter 4: Optical flow reflexes .................................. 52
Chapter 5: Exteroceptive reflexes and sensory integration ....... 78
Chapter 6: RoboLobster hardware and software ............... 106
Chapter 7: Biorobotic analysis ..................................... 143
Chapter 8: Conclusions, opportunities, recommendations .... 176
References .............................................................. 181
Appendices ................................................................
A: Neurobots curriculum ........................................... 199
B: RoboLobster parts drawings ....................................
C: Communications protocols ....................................
D: Electronic components .......................................... 199
E: Neuron and synapse settings .................................
F: Programming instructions ....................................
G: MATLAB scripts ..................................................
List of Figures

Figure 1.1: RoboLobster test platform 2
Figure 1.2: Graphical overview of dissertation chapters 6
Figure 1.3: Previous RoboLobster thoraco-coxal joint 12
Figure 1.4: New RoboLobster thoraco-coxal joint 12
Figure 2.1: Neuron network diagram overview 18
Figure 2.2: Output of discrete time map-based neuron model 19
Figure 2.3: Shape of DTM neuron function 21
Figure 2.4: Neural network programming in LabVIEW™ software 22
Figure 2.5: Parametric modulation simulation in LabVIEW™ 24
Figure 2.6: Perturbation simulation in LabVIEW™ 25
Figure 3.1: External anatomy of the lobster 28
Figure 3.2: Movements of the lobster’s walking legs 28
Figure 3.3: Joint movements in the third right walking leg 31
Figure 3.4: Movement phases and motor synergies of lobster walking 33
Figure 3.5: Proposed lobster walking CPG 34
Figure 3.6: Output of lobster walking CPG for forward walking 36
Figure 3.7: Neural activity for forward and backward walking 37
Figure 3.8: Neural activity for sideways and diagonal walking 38
Figure 3.9: Walking coordination network 41
Figure 3.10: Coordinating network neuron output 43
Figure 3.11: Kinematic tracking of leg movement from side-view video recording 44
Figure 3.12: Walking output generated by RoboLobster’s CPG 45
Figure 3.13: Simulated neuromodulation of RoboLobster’s CPG 46
Figure 3.14: CPG output with low command neuron input 48
Figure 3.15: CPG output with high command neuron input 48
Figure 3.16: Forward command recruitment of retractor motor neurons 49
Figure 3.17: Video data of RoboLobster’s command neuron modulation 50
Figure 4.1: Overview of arthropod optical flow reflexes 54
Figure 4.2: Optical flow neuron network hypothesis for walking arthropods 56
Figure 4.3: Medial translational flow initiates backward walking 57
Figure 4.4: Clockwise angular optical flow initiates rotation in place 58
Figure 4.5: Optical flow-mediated obstacle avoidance 59
Figure 4.6: Optical flow-mediated yaw response 60
Figure 4.7: ArduBot platform 61
Figure 4.8: ArduBot testing arena 62
Figure 4.9: ArduBot path without corollary discharge 63
Figure 4.10: Optical flow network model with corollary discharge 64
Figure 4.11: Centering response in ArduBot 65
Figure 4.12: Optical flow reflexes in ArduBot 66
Figure 4.13: Flying arthropod optical flow reflex network 68
Figure 4.14: Neuron-based optical flow-mediated odometry 70
Figure 4.15: Population coding of the visual odometer 72
Figure 4.16: Synaptic modulation of odometry circuit in LabVIEW™ simulation 73
Figure 4.17: Hardware overview and validation for flying robot 75
Figure 4.18: Odometer synapse strength is correlated with flight distance 76
Figure 4.19: Neural activity of odometer network changes with varied synapse strength 77
Figure 5.1: Lobster antennal reflexes 80
Figure 5.2: Range fractionation 80
Figure 5.3: LabVIEW™ simulation of antennal bend neuron network 82
Figure 5.4: Range fractionation of the compass network 83
Figure 5.5: Heading control neuron network 84
Figure 5.6: Claw bump drives a back-up response 85
Figure 5.7: LEGO Mindstorms NXT robot 86
Figure 5.8: Bump exteroceptive network for LEGO Mindstorms NXT robot 87
Figure 5.9: Overhead video tracking of lobster and robot 88
Figure 5.10: Tracked paths of lobster and LEGO robot 89
Figure 5.11: Neural network hypothesis for integration of antennal and visual inputs 94
Figure 5.12: LabVIEW™ simulation of optical flow and antennal bend integration 95
Figure 5.13: Exteroceptive reflex networks for RoboBee 97
Figure 5.14: RoboBee’s chain-reflex sequencing 99
Figure 5.15: LabVIEW™ simulation of compass and optical flow neuron network 100
Figure 5.16: Complete biomimetic exteroceptive reflex network for RoboLobster 101
Figure 5.17: Gain modulation between high antennal bend and optical flow inputs 102
Figure 5.18: Modality gating between high heading deviation and optical flow inputs 103
Figure 5.19: Corollary discharge from motor system to optical flow inputs 103
Figure 5.20: RoboLobster’s complete ENS 105
Figure 6.1: RoboLobster test platform 107
Figure 6.2: Leg part fabrication techniques 108
Figure 6.3: 3D printed leg

Figure 6.4: Robot leg design and actuation

Figure 6.5: Kinematic tracking of leg movement from side-view video recording

Figure 6.6: Rubber toe added to minimize slipping

Figure 6.7: Kinematic tracking of leg movement after adding rubber toes

Figure 6.8: SMA actuators

Figure 6.9: Bottom view of antenna actuation assembly

Figure 6.10: Driver boards mediate excitation-contraction coupling

Figure 6.11: Driver board stack on RoboLobster

Figure 6.12: DSP-driver board connector and remote power switch

Figure 6.13: Fully assembled driver board stack

Figure 6.14: Through-holes for actuator wire leads

Figure 6.15: DSP board on RoboLobster

Figure 6.16: DSP communication overview

Figure 6.17: RoboLobster’s power/sensor board

Figure 6.18: DSP board connected to the power/sensor board

Figure 6.19: Power circuitry wiring diagram

Figure 6.20: RoboLobster hardware overview

Figure 6.21: Antennal sensor design and output

Figure 6.22: Bend sensor testing

Figure 6.23: Optical flow sensor on RoboLobster

Figure 6.24: Bump sensor assembly

Figure 6.25: ENS C code overview
Figure 6.26: Command neuron serial communication 142
Figure 7.1: Flow tank for robot testing 146
Figure 7.2: Video setup for recording of behavior 147
Figure 7.3: CPG output on RoboLobster 148
Figure 7.4: Tracking of RoboLobster’s response to magnetic claw contact 149
Figure 7.5: Neuron activity of RoboLobster’s response to magnetic claw contact 150
Figure 7.6: Tracking of RoboLobster’s response to claw bumper contact 151
Figure 7.7: Neuron activity of RoboLobster’s response to claw bumper contact 151
Figure 7.8: Tracking of RoboLobster’s response to heading deviation 152
Figure 7.9: Neuron activity of RoboLobster’s response to heading deviation 153
Figure 7.10: Tracking of RoboLobster’s response to optical stimulus on right 155
Figure 7.11: Neuron activity of RoboLobster’s response to optical stimulus on right 156
Figure 7.12: Tracking of RoboLobster’s response to optical stimulus on left 157
Figure 7.13: Neuron activity of RoboLobster’s response to optical stimulus on left 158
Figure 7.14: Neuron parameters affect optical flow reflex 159
Figure 7.15: RoboLobster obstacle avoidance response, $\sigma = -2.83$ 159
Figure 7.16: RoboLobster obstacle avoidance response, $\sigma = -2.88$ 160
Figure 7.17: RoboLobster obstacle avoidance response, $\sigma = -2.98$ 160
Figure 7.18: RoboLobster response to lateral hydrodynamic flow with antennae forward 161
Figure 7.19: Neuron activity of RoboLobster’s response to lateral flow with antennae forward 162
Figure 7.20: RoboLobster response to lateral hydrodynamic flow with antennae positioned laterally 163
Figure 7.21: Neuron activity of RoboLobster’s response to flow with antennae held laterally 164

Figure 7.22: Tracking of RoboLobster’s response to antennal bend and bump stimuli 166

Figure 7.23: Neuron activity of RoboLobster’s response to antennal bend and bump stimuli 166

Figure 7.24: Tracking of RoboLobster’s response to heading deviation and bump stimuli 167

Figure 7.25: Neuron activity of RoboLobster’s response to simultaneous heading deviation and bump stimuli 168

Figure 7.26: Tracking of RoboLobster’s response to heading deviation and bump stimuli: different target heading 169

Figure 7.27: Neuron activity of RoboLobster’s response to heading deviation and bump stimuli with new heading 169

Figure 7.28: Tracking of RoboLobster’s response to right optical flow and bump stimuli 170

Figure 7.29: Neuron activity of RoboLobster’s response to right optical flow and claw stimuli 171

Figure 7.30: Tracking of RoboLobster’s response to left optical flow and claw stimuli 172

Figure 7.31: Neuron activity of RoboLobster’s response to left optical flow and bump stimuli 172

Figure 7.32: Tracking of RoboLobster’s response to three modes of sensory stimuli 173

Figure 7.33: Neuron activity of RoboLobster’s response to three modes of sensory stimuli 174
List of Tables

Table I: Comparison between lobster and robot leg movement 44
Table II: Averaged parameters from analysis of video behavior data 89
Table III: Comparison between RoboLobster and Homarus americanus specimens 107
Chapter 1. Introduction

A. Overview

The development and testing of biorobotic models is an effective way to assess our understanding of biological principles, specifically of nervous system structure and function (Webb, 2006). The biorobotic platform, RoboLobster, has improved our understanding of lobster sensory processing, motor program coordination, and biomechanics (Ayers & Witting, 2007). Here we present a third generation of RoboLobster that represents a closed-loop biorobotic simulation of neuron network operation that can be run in unpredictable real-world environments (Beer et al., 1998; Webb, 2002).

A reconfigurable Electronic Nervous System (ENS) onboard RoboLobster allows for the testing of neuroethological hypotheses including central pattern generator (CPG)-driven walking and lobster exteroceptive reflexes. The ENS-architecture has been shown effective on a lamprey swimming robot and we extend this approach to the control of underwater walking (Westphal, Rulkov, Ayers, Brady, & Hunt, 2011). RoboLobster allows for real-world simulation of neuronal models of the taxes and kineses (Braitenberg, 1965; Loeb, 1918) that have driven neuroscience and bio-inspired robotics for the past century (Braitenberg, 1986; Kennedy & Davis, 1977; Webb, Harrison, & Willis, 2004).

The extension of this model described herein serves as a tool to investigate the neural basis of sensory processing in invertebrates. The completely redesigned third-generation RoboLobster (Figure 1.1) runs an ENS simulation utilizing real-time implementation of nonlinear dynamical models of neurons and synapses (Rulkov, 2002). We extend previous versions of RoboLobster that were controlled by a set of finite-state machines to generate locomotion (Ayers & Witting, 2007). This new version uses reengineered biomimetic body
structures and a buoyancy(ballast redistribution to better approximate the physics of the model organism.

Figure 1.1. RoboLobster test platform. 8-legged biomimetic robot lobster used for studying neuroscience.

The development of RoboLobster represents an attempt to approximate each of the four aspects comprising a complete biomimetic platform: sensors, biomechanics, motor pattern generation, and behavior. But the objective moves beyond the engineering of a new class of biomimetic robots as we use this platform to address biological questions and test hypotheses (Webb, 2006). Questions of biology can be posed and tested using such an embodied simulation of an organism. For example, how does the nervous system control an animal’s behavior? Once a robot mimics the brain, body and sensors of an animal, biological questions can be investigated. Some of the questions this work addresses are:

- How conserved are the optical flow reflexes of lobsters and other arthropods?
- How does the lobster nervous system integrate information from optical flow inputs, antennal bending, claw bumping, and compass information to generate a coherent motor output?
- What is a potential neural explanation for the arthropod visual odometer?
The following work presents a robotic lobster platform that can be used as a tool to fuel biological investigation. Even though the platform described leads the field in terms of simulated biological complexity, many abstractions and simplifications have been adopted. As such, the main contribution of this work lies in its power as a tool to conduct computational neuroscience. Biomimetic robots can be used to test existing neuron network models and refine them, develop new models, and test alternative neurobiological hypotheses. One anticipated outcome of the work is to directly inform biological study, i.e., find emergent principles from the robotic simulations that can be used to target neurophysiological analysis.

Robots have traditionally been controlled by algorithmic programming, which includes if-then statements, linked lists, timed loops and lookup tables. Traditional control theory often utilizes Internal Model Control and control loop feedback mechanisms like PID (Proportional-Integral-Derivative) controllers (Pounds, Bersak, & Dollar, 2012; Rivera, Morari, & Skogestad, 1986). RoboLobster uses an alternative programming framework: an ENS controller (Westphal et al., 2011). Neurons and synapses that describe a network hypothesis derived from biological knowledge are simulated mathematically. Action potentials (spikes) of simulated neurons pass information through synapses to transform sensory inputs to motor outputs. Sensory neurons respond proportionally to sensory input to drive layered reflex networks that modulate command neurons that affect motor pattern networks. Motor neuron action potentials drive actuation of the legs. Part of the motivation of this work is to determine if an ENS can be as effective as traditional approaches to robot control. Both approaches may have advantages for different situations and the new RoboLobster allows us to assess ENS performance (see Chapter 7).

The integration of an ENS into RoboLobster’s control architecture allows for researcher-directed modification of neuron, synapse and network characteristics. By modifying hypothetical
neural models on a biomimetic robotic vehicle, the role of synaptic network and cellular processes can be assessed, confirmed, and expanded. Particularly, how does the lobster nervous system mediate the animal’s adaptations to contingencies of the environment? The work presented here explores neuroethological hypotheses related to this question with specific focus on the interaction between different sensory systems, the role of intrinsic variable neural activity, and complicated optical flow processing. I will demonstrate that RoboLobster is a valuable tool for testing hypotheses in neuroscience as the robot’s nervous system can be manipulated at all organizational levels, from neurons and synapses to reflexes and systems.

The biorobotic approach to neuroscience is extended further yet with a flying robotic bee prototype and other land-based hypothesis testing platforms that are presented herein. These additional platforms allow for testing more specific hypotheses related to optical flow processing, in particular visual odometry and the honeybee’s visual responses during corridor traversal (Baird, Srinivasan, Zhang, & Cowling, 2005; M. V. Srinivasan, 2014).

After this introduction come two main sections. The first, spanning Chapters 2-5, describes how nervous system models are developed and tested in simulation. In Chapter 2, an overview of the computational software methods used to simulate neuron networks is presented. Chapters 3-5 each cover a different subset of the nervous system: walking pattern generation (Chapter 3), optical flow reflexes (Chapter 4) and multimodal sensory integration (Chapter 5). Within each of these chapters our current understanding of the neuroethological principles of the nervous system component is outlined and neural network models are formulated. LabVIEW™ software simulations and limited robotic implementations are also presented.

The second section presents an evaluation of the nervous system using RoboLobster as an embodied simulation. Chapter 6 covers the development and engineering of the biomimetic
hardware and software. Nervous system implementation on RoboLobster is presented in Chapter 7. The approach to nervous system simulation on embedded software is described and robot performance is assessed and compared to animal behavior. Chapter 8 discusses conclusions to draw from this work and recommendations for future work. See Figure 1.2 for a graphical overview of the dissertation.
Figure 1.2. Graphical overview of dissertation chapters. (Photographs from Ayers, 2002; Ayers & Witting, 2007; Case Western Reserve Univ.; Flannigan et al., 1998; Huxley, 1880; Iserman; Şafak & Adams, 2002; Webb; Westphal et al., 2011)
B. Biomimetic robots

There are two main motivations for the work of biomimetic robot engineers. One motivation is to improve robotic technology by learning from animals (Fuller, Sands, Haggerty, Karpelson, & Wood, 2013). A second motivation is to use robots to study biological principles (Webb & Reeve, 2003). All work in the field is inherently some combination of these motivations. And while RoboLobster may advance robot abilities, our main goal is to use the platform to implement and test neurobiological hypotheses.

There are four main characteristics of animals that researchers have explored using robots: sensors, motor pattern generation, biomechanics, and behavior. Robot prototypes often focus on one or more of these areas in order to improve system performance and to learn about biological systems.

Optical flow processing is a prominent component of visually-based adaptation in robotics (for a review see Franceschini, 2014). Decapod crustaceans and insects, and many other animals, use optical flow, the movement of the world across the visual field, to control and stabilize locomotion (Davis & Ayers, 1972; M. Srinivasan et al., 1996). Efforts to engineer autonomous robots often adopt established platforms, such as quad-rotor helicopters, but use biologically analogous sensors including optical flow sensors (Chahl, Rosser, & Mizutani, 2011; Conroy et al., 2009; Green, Oh, & Barrows, 2004). For example, optical flow has been described as the only practical sensory system to mediate obstacle avoidance on micro air vehicles, a lesson learned from biology (Garratt & Cheung, 2009).

The processing and integration of such bio-inspired sensory information is usually based on traditional control theory. Kalman filtering and stereovision-based range detection have been used to handle optical flow processing (Garratt & Chahl, 2003; Moore et al., 2011). Bio-inspired
optical flow processing has shown great promise in simulations (Garratt & Cheung, 2009) and in tethered (Ruffier & Franceschini, 2014) or simplified (Roubieu et al., 2014) physical implementations. However, the neural foundations of such visual processing are often ignored or greatly abstracted. RoboLobster is a platform that allows investigation of the neuronal dynamics of optical flow integration on an unrestrained vehicle.

A range of other sensory modalities has been explored through biomimetic efforts. Inertial sensors have been shown in animals (Fraser, 2001) and simulated in robots (Lobo, Ferreira, & Dias, 2006). Air velocity sensors for flight have been identified in Drosophila (Yorozu et al., 2009) and implemented on robots (Rutkowski et al., 2011). A magnetic compass sense has been observed in many animal species including birds, turtles and spiny lobsters and allows for long range migration and homing (Boles & Lohmann, 2003; Fleissner et al., 2007; K. J. Lohmann, Putman, & Lohmann, 2012; Putman et al., 2014). Correspondingly, many robots use magnetic compasses (Hardt, Arnould, & Wolf, 1994; Suksakulchai et al., 2000) and a recent implementation on a swimming robot implemented a neuronal compass hypothesis (Westphal et al., 2011). Insect-inspired polarized light compass navigation has been demonstrated as well (Lambrinos et al., 1998) and approaches to bioinspired sensory fusion have been explored (Campolo et al., 2008; Westphal, 2012).

In addition to the implementation of biomimetic sensors, approaches to achieve robot autonomy draw on principles of animal behavior. Extensive ethological investigation in honeybees and fruit flies has fueled many efforts to mimic animal solutions to flight control problems [for a review see: (M. Srinivasan, 2011; Dickinson et al., 2000)]. Visual regulation of flight speed, altitude, and obstacle negotiation observed in honeybees (Baird et al., 2005; Portelli, Ruffier, & Franceschini, 2010; M. V. Srinivasan et al., 1991) and Drosophila (Fry et al., 2009;
van Breugel & Dickinson, 2012) has inspired a range of robotic implementations (Chahl, Srinivasan, & Zhang, 2004; Coombs & Roberts, 1992; Roubieu et al., 2012). Recent advances have improved our understanding of how neuronal activity (Maimon, Straw, & Dickinson, 2010) and muscle actuation (H. Wang, Ando, & Kanzaki, 2008; Balint & Dickinson, 2004) drive insect flight.

To date, the incorporation of arthropod-inspired sensors and control methods has mostly occurred on traditional robot platforms including wheeled underwater vehicles (Basil & Atema, 1994), fixed wing aircraft (Thurrowgood, Moore, Soccol, Knight, & Srinivasan, 2014; Wood et al., 2007) and helicopters (Franceschini, Ruffier, & Serres, 2007; Han, Straw, Dickinson, & Murray, 2009; Barrows et al., 2002). Efforts to biomechanically mimic body plans and actuation have spanned from frogs (Scarfogliero, Stefanini, & Dario, 2007) and crickets (Nelson & Quinn, 1999) to lamprey (Stefanini et al., 2012; Westphal et al., 2011) and crustaceans (Ayers & Witting, 2007; Flannigan et al., 1998). A wide variety of biorobotic approaches to study locomotion have been endeavored (for a review see: Ayers et al., 2002; Ijspeert, 2014). Insights from the study of insect flight biomechanics (for a review see: Z. J. Wang, 2005) have pushed the development of flapping wing air vehicles (Teoh & Wood, 2013). Recent advances in understanding wing morphology and aerodynamics (Donoughe, Crall, Merz, & Combes, 2011; Nakata & Liu, 2012; Sudo, Tsuyuki, & Kanno, 2005; Zhao, Huang, Deng, & Sane, 2010) have driven new advances in artificial wing design (Shang, Combes, Finio, & Wood, 2009) and the development of insect-inspired flapping wing MAV’s (Madangopal, Khan, & Agrawal, 2004; Wood, 2008). Insect-inspired ocelli have been used to control some aspects of free flight of a flapping MAV (Fuller, Karpelson, Censi, Ma, & Wood, 2014). And a magnetometer has been used to control pitch and yaw (Helbling, Fuller, & Wood, 2014). These impressive
demonstrations still rely on off-board power and sensing (motion capture systems) to aid in control. We implement a synaptic network-driven behavioral controller onto a completely autonomous robot. To date, such system-wide modeling approaches have been limited to lamprey and salamander (Ijspeert, Crespi, Ryczko, & Cabelguen, 2007; Steingrube, Timme, Wörgötter, & Manoonpong, 2010; Westphal et al., 2011).

C. Previous RoboLobsters

Several generations of lobster-based simulations and robots have been developed over the past 20 years (Ayers & Crisman 1992; Ayers, 2000a; Ayers, 2000b). The motivation for the project originally stemmed from a need to develop robust robots for shallow water operation (Ayers, Crisman, & Massa, 1992). Given the lobster’s capabilities in this environment, a robot with similar body plan and locomotory modes was developed (Ayers & Witting, 2007). The robot’s control architecture implemented the control rules of the motor pattern generator for omnidirectional locomotion and postural adaptations (Ayers, 2004). The command neuron, coordinating neuron, CPG model (Kennedy & Davis, 1977; Stein, 1978) was used as the framework to organize RoboLobster’s control. Low-level control was accomplished using a state machine that emulated an endogenous pacemaker CPG with gated circuitry driving the different joints of the legs to produce coordination patterns for walking in different directions. High-level control was comprised of a behavioral library of action patterns and a reactive sequencer. The sequencer switched between different command states as specified by sensor input and the behavioral libraries (Ayers, 2002). Each component of the neuronally-inspired controller was represented as a software object (e.g. neuronal oscillation, pattern generation) through object programming techniques (Ayers & Crisman, 1992). Coordinating signals orchestrated
locomotion between legs and descending commands specified motor outputs. Pattern generators
selected specific muscle synergies to produce different walking modes (e.g. forward or backward
walking). The generalized neural circuit model served as the basis of a functional model that
comprised the finite-state machine limb controller (Ayers, 2002). The new RoboLobster
advances this neuronal architecture by implementing a standalone computational model of the
lobster nervous system based on these organizational rules (Ayers, 2002) and the hardware
architecture developed for the lamprey robot (Westphal, N. Rulkov et al., 2011).

A MEMs-based antenna, compass and accelerometer were previously implemented
onboard RoboLobster (Ayers, 2002). Selected by sensory inputs, software sensor objects passed
messages to command objects to drive motor outputs. As a finite-state machine, behavior was
discretized into command states (Ayers, 2002). The new RoboLobster advances the sensory
capabilities by adding optical flow and tactile sensors that drive computational models of sensory
neuron networks.

The design of RoboLobster previously used a vertical orientation for the thoraco-coxal
joint (Figure 1.3). As a consequence, the center of mass and buoyancy were elevated and leg
motion lacked the rolling action of the lobster leg. The new RoboLobster improves on the leg
design by aligning the thoraco-coxal joint in the same orientation as in the model animal (Figure
1.4). This arrangement produces more natural leg movements by mimicking the rolling action of
the lobster leg tip as opposed to the more rectilinear motion of the previous vehicle. With the
resulting decrease in the vertical profile of the leg, RoboLobster’s center of mass and center of
buoyancy are more closely aligned allowing for more stable operation.
Figure 1.3. *Previous RoboLobster thoraco-coxal joint.* Due to the vertically orientated protractor and retractor nitinol actuators (white arrows), it was necessary to build a tall hull that resulted in a biologically unrealistic weight distribution.

Figure 1.4. *New RoboLobster thoraco-coxal joint.* The nitinol actuators of the new hip joint are oriented diagonally relative to the underside of the hull. One actuator is shown which connects the joint (right) and an adjustment apparatus (left). This design eliminates the need for a tall hull to accommodate the actuator height.

An analogue circuit implementation of Hindmarsh Rose equations was used to drive the motor activation of a single robot leg (Ayers et al., 2003; Hindmarsh & Rose, 1984). We extend this neural control of locomotion to an 8-legged free-walking implementation using Rulkov’s
phenomological neuron model (2002). The previous work that built earlier versions of RoboLobster has paved the way for the work presented here. By extending this biorobotic approach, this dissertation presents a hypothesis-testing platform to explore neuronal mechanisms underlying arthropod behavior.
Chapter 2. Model organisms and computational methods

A. The lobster as a model organism

Comparative physiology has made profound advances by exploiting the technical advantages of particular animal models. Although no organism is completely understood, a composite model emerges from the synthesis of available models. To develop a model describing the lobster’s nervous system we draw from knowledge of other systems, including crayfish, bees and other insects, that present conserved neural features. The complete ENS for RoboLobster is devised by layering and integrating the animal’s various exteroceptive sensorimotor reflexes including those driven by inputs from antennae, eyes, a magnetic compass, and claw mechanoreceptors. By leveraging Loeb’s (1918) and Braitenberg’s (1965) theories about the control of taxes and kineses into the neuronal control of a robot, we have developed a biologically-based model fully amenable to hypothesis testing. This extends the biorobotic approach employed by Westphal et al. to study lamprey neuroethology (2011), by Webb et al. to study cricket phonotaxis (Webb & Reeve, 2003) and by Grasso et al. to study chemotaxis in lobsters (Grasso, 2001; Grasso, Basil, & Atema, 1998). The advantage of this approach is that we can compare the discharge patterns of robot neurons with those of other decapod crustacean neurons (Kagaya & Takahata, 2010) in the behaving organism.

The basic biology and body plan of the decapod Homarus americanus is described in detail elsewhere (Herrick, 1911; Cobb, 1976; Bliss, 1982; Factor, 1995). The body of this model decapod has 21 segments, the first 14 of which make up the cephalothorax and are covered by the carapace. There are five pairs of pereiopods, or walking legs, the first pair of which have been modified to form the crusher and cutter claws. The structure of the lobster’s walking legs is particularly important for understanding the function of the overall system and is presented in
detail in Chapter 3. On the ventral side of the abdomen are six pairs of pleopods, or swimmerets, of which the last pair is enlarged to form the uropods (Cobb, 1976).

B. General nervous system overview

Anatomically, the nervous system is comprised of a central nerve cord that passes ventrally through the body, connecting subesophageal, thoracic, and abdominal ganglia. The supraesophageal ganglion, which lies between the eyes dorsal to the esophagus, integrates inputs from the first three metameres. Circumesophageal connectives traverse around the esophagus, joining the supra- and sub-esophageal ganglia (Cobb, 1976).

Recent advances in our understanding of the neural foundations of behavior have focused on central pattern generators (CPGs) that drive motor outputs (Daun-Gruhn, 2010; McCrea & Rybak, 2007; M. Wang, Yu, & Tan, 2009). While differences exist in the body structure and effectors across organisms, the command neuron, coordinating neuron and CPG framework outlines a conserved neural architecture (Kennedy & Davis, 1977; Pearson, 1993). Many of the neural principles that underlie decapod locomotion are understood (Ayers & Davis, 1977; Evoy & Ayers, 1982; Ayers & Crisman, 1992) but details can be gleaned from other animals.

CPGs are local networks of neurons that can generate rhythmic motor output in the absence of sensory feedback (Kennedy & Davis, 1977). Such networks are present in the lobster’s ganglia and control locomotion of swimmerets (Ikeda & Wiersma, 1964), legs (Chrarchi & Clarac, 1990) and the stomach (Miller & Selverston, 1982). The generation of walking motor patterns using in vitro preparations (Chrarchi & Clarac, 1990) along with pairwise recording of CPG interneurons and motor neurons (Chrarchi & Clarac, 1989) has provided evidence of the decapod walking CPG. Application of muscarinic agonists on the fourth thoracic
ganglion of the crayfish induces fictive locomotion (Chrarchi & Clarac, 1990). Sensory feedback can modulate the CPG activity as was shown by activation of the thoracicocoxal muscle receptor organ to entrain the walking rhythm (Sillar & Skorupski, 1986).

The CPG model includes descending command neurons that can activate local control centers where choices to generate particular behaviors are processed (Kennedy & Davis, 1977; Stein, 1978). A variety of command neurons have been identified in decapods including some controlling swimmeret movements (Wiersma & Ikeda, 1964), forward and backwards walking (Bowerman & Larimer, 1974a), and abdominal position (Kennedy, Evoy & Fields, 1966). Command neurons can be described after they are stimulated and the resulting motor output is observed (Atwood & Wiersma, 1967; Bowerman & Larimer, 1974a, Bowerman & Larimer, 1974b). As stimulation frequency of a command neuron increases, the speed of the evoked response generally increases (Atwood & Wiersma, 1967).

Coordination of limb movements can be orchestrated by coordinating neurons that can be initiated by centrally derived signals, sensory signals, or a combination of the two (Kennedy & Davis, 1977; Namba & Mulloney, 1999). By way of coordinating neurons, a governing CPG passes information to a governed CPG. Depending on the characteristics of the synapse, phase advances or delays can occur, leading to the coordination of intersegmental phase or gait (Ayers & Selverston, 1979). Usually this coordination is driven by neurons that are distinct from the pattern generator neurons (Smarandache et al., 2009). Little is known about the mechanisms underlying lobster walking coordination so other model systems must be considered, particularly the crayfish swimmeret system (Mulloney & Hall, 2007).

Decapod locomotion is an innate behavior defined as a modal action pattern (often nested within more complex sequences) since the outputs and mechanisms are similar across specimens
and across different species. Nevertheless, this behavior displays particular variety because of the magnitude of command and sensory inputs and the complexity of the underlying neuronal circuitry. There is also metastability and plasticity of the CPG dependent on the mode or direction of walking (Ayers & Davis, 1977a).

A more detailed review of the lobster’s nervous system organization and its computational implementation are presented in Chapter 3.

C. Network conventions

Throughout this work, hypotheses will be presented as diagrams of neuron networks. An arrangement of neurons, connected via synapses, represents a hypothesis to test in simulation, using standalone software or a robot. Alternative network hypotheses can be developed and tested. The network hypotheses generally only contain functional neural components and are simplified to generate the most parsimonious explanation possible. Comparisons between simplified and more elaborate models are also presented to verify adequate performance of the neuron networks.

Neurons in the neural networks are modeled as single units and represented as large, labeled circles (Figure 2.1). Synapses are represented by connecting lines between neurons. Lines with filled triangle ends represent excitatory synapses, and filled circle ends are inhibitory. Figure 2.1a shows an example diagram in which an excitatory synapse connects Neuron A to Neuron B. Figure 2.1b shows an inhibitory synapse connecting Neuron A to Neuron B. And Figure 2.1c shows Neuron A with a self-inhibitory synapse.
Figure 2.1. Neuron network diagram overview. Throughout the text neuron networks are presented following these conventions. a. Diagram showing Neuron A connected to Neuron B via an excitatory synapse. b. Diagram showing Neuron A connected to Neuron B via an inhibitory synapse. c. Diagram showing Neuron A with self-inhibition.

D. Discrete time map-based neuron model

Modeling of CPG circuits on robots requires a computationally efficient neuron model for real-time network operation. For all neuron network simulations, a discrete time map-based model is used to simulate each component neuron and synapse (Rulkov, Timofeev, & Bazhenov, 2004; Rulkov, 2002). This phenomenological model is capable of replicating the spike pattern characteristics of synaptic networks (Ayers & Rulkov, 2007; Ayers et al., 2010). By modifying two control parameters, \( \alpha \) and \( \sigma \), a range of neuron outputs can be obtained: from tonic spiking and bursting, to intrinsic silence and chaotic firing (Figure 2.2). Three adjustable parameters modulate the reversal potential, relaxation rate, and strength of the synapse.
Figure 2.2. *Output of discrete time map-based neuron model.* As control parameters $\alpha$ and $\sigma$ are changed, the output of the neuron model varies between intrinsic silence, bursts of spikes (A), chaotic oscillations (B), and tonic spiking (C). (Rulkov, 2002)

A two-dimensional discrete time map describes the spiking and spiking-bursting behavior of the DTM neuron model and can be written as

$$x_{n+1} = f_\alpha(x_n, x_{n-1}, y_n + \beta_n)$$

$$y_{n+1} = y_n - \mu(x_n + 1) + \mu\sigma + \mu\sigma_n$$

where $x_n$ is the fast and $y_n$ is the slow (due to $0 < \mu << 1$) dynamical variables. The nonlinear function is written as

$$f_\alpha(x_n, x_{n-1}, y_n + \beta_n) = \begin{cases} \frac{\alpha}{1 - x_n} + (y_n + \beta_n), & x_n \leq 0, \\
\alpha + (y_n + \beta_n), & 0 < x_n < \alpha + (y_n + \beta_n) \text{ and } x_{n-1} \leq 0, \\
-1, & x_n \geq \alpha + (y_n + \beta_n) \text{ or } x_{n-1} > 0
\end{cases}$$

where the third argument $y_n + \beta_n$ can be a combination of input variables that depend on the model type and synaptic inputs. The parameters $\alpha$ and $\sigma$ shape the map (Figure 2.3) and define characteristics of individual neurons. Input variables $\beta_n$ and $\sigma_n$ incorporate the action of synaptic
input, $I^{\text{syn}}$, and can be written as $\beta_n = \beta E I_{n+1}^{\text{syn}}$ and $\sigma_n = \sigma E I_{n+1}^{\text{syn}}$ where $\beta E$ and $\sigma E$ are constants that control how quickly neurons respond to the input. A map-based model capturing synaptic integration and dynamics can be written as

$$I_{n+1}^{\text{syn}} = \gamma I_{n+1}^{\text{syn}} + \text{presyn}_{\text{in}} - \begin{cases} g_{\text{syn}}(x_{n}^{\text{post}} - x_{\text{rp}}), & \text{spike}_{\text{pre}}, \\ 0, & \text{otherwise} \end{cases}$$

where $g_{\text{syn}}$ is the strength of synaptic coupling, indexes $\text{pre}$ and $\text{post}$ stand for the presynaptic and postsynaptic variables, respectively. Here $\gamma$ controls the relaxation rate of the synapse ($0 < \gamma < 1$) and $x_{\text{rp}}$ defines the reversal potential and, therefore, the type of synapse: excitatory or inhibitory.

Network topology is determined by synaptic connections between presynaptic and postsynaptic cells. The synaptic current from a presynaptic to a postsynaptic neuron updates iteratively based on the equation above. The $x_n$ value of the presynaptic neuron is used to determine if an action potential has occurred using the following

$$\text{if } x_n > 0 \text{ AND } x_{n-1} > 0, \text{ then: } \text{spike}_{\text{pre}} = 1$$
$$\text{else: } \text{spike}_{\text{pre}} = 0$$

An alternative spike equation was also used in LabVIEW™ simulation and can be written as

$$\text{if } x_n > \alpha + \sigma + \beta_n \text{ OR } x_{n-1} > 0, \text{ then: } \text{spike}_{\text{pre}} = 1$$
$$\text{else: } \text{spike}_{\text{pre}} = 0$$

If $\text{spike}_{\text{pre}} = 1$, then the synapse equation incorporates the postsynaptic neuron’s $x_n$ value into the synapse equation calculation.

Presynaptic inhibition from command neuron input is calculated as follows

$$\text{presyn}_{\text{in}} = \text{presyn}_{\text{in-1}} - \text{comStrength} \times \text{comSpike} + 0.1$$
where \textit{comStrength} is the strength of the command neuron’s presynaptic inhibition and \textit{comSpike} is the command neuron spike. The values of \textit{presyn} are bounded between -15 and 0. When the command neuron is not spiking, the \textit{presyn} value approaches zero.

![Figure 2.3. Shape of DTM neuron function. Plotted for $\alpha = 6.0$ and $y = -3.93$ as a solid line. A superstable cycle (P_k) is illustrated by the dashed line. $x_{ps}$ and $x_{pu}$ are the stable and fixed points of the map, respectively. (from Rulkov, 2002).](image)

The DTM mathematical model is well suited for robotic implementation because it uses computationally efficient difference equations that mimic neural dynamics rather than processor-intensive differential equations that model ionic conductances. This model has been adapted to run in LabVIEW\textsuperscript{TM} and embedded software (C language) and is used on all robot platforms, RoboLobster, RoboBee, ArduBot, and LegoLobster.

\textit{E. Simulation in LabVIEW\textsuperscript{TM} software}

The neurons and synapses of a network model are instantiated as LabVIEW\textsuperscript{TM} Virtual Instruments (VI) and connected by LabVIEW\textsuperscript{TM} “wire” in the G Programming language (Figure 2.4) (see Blustein & Ayers, 2010a for a complete study). Each VI runs an instance of the DTM equations (Figure 2.4c) and all VIs operate within a \textit{While Loop} so that the network will update iteratively. Presynaptic neuron spike outputs are connected through synapses to postsynaptic neurons. Iteratively updating parameters are passed through shift registers or feedback nodes to
the next calculation cycle. The DTM model updates fast (x) and slow (y) dynamical variables that are passed to the next iteration of calculations.

![Diagram of neuron connections](https://example.com/diagram.png)

Figure 2.4. *Neural network programming in LabVIEW™ software.* a. Two neurons and two synapses connected with two reciprocal excitatory synapses. b. LabVIEW™ Virtual Instrument block diagram for network in a. c. LabVIEW™ G language code for a 2-dimensional neuron described by the DTM equations (Rulkov, 2002). As the network simulation operates, equations are iteratively calculated. The resulting values are updated and passed between neurons and synapses for each calculation cycle through shift registers.

LabVIEW™ VIs can be used to input sensor information into the networks. Data from real sensors can be integrated using a LabVIEW™ DAQ board. Alternatively, software only simulations can use front panel control boxes to manually manipulate the sensor information going to the neural network. Graphical charts on the front panel of the LabVIEW™ program are used to visualize neuronal activity. When the neuron network is running, front panel controls can
be manually or programmatically manipulated to present varied sensory input. For LabVIEW™ simulation figures, screenshots of front panel neural activity data output are used.

To demonstrate LabVIEW™ implementation of simple neuron networks, simulations of parametric modulation and perturbation are presented. The activity of a presynaptic neuron can be used to drive a postsynaptic neuron as in Figures 2.1a and 2.1b. Under parametric modulation, tonic activity of the presynaptic neuron (Neuron A) drives the response of the postsynaptic neuron (Neuron B) (Figure 2.5). This is how command neurons can gate on different network outputs. With no input, Neuron A is intrinsically silent and Neuron B fires regular bursts of action potentials (Figure 2.5a). With tonic depolarization via an excitatory synapse from Neuron A to B and injected current into Neuron A, Neuron B shows increased spike activity (Figure 2.5b). With tonic hyperpolarization via an inhibitory synapse from Neuron A to B, Neuron B’s activity is diminished and the burst period increases (Figure 2.5c). Stronger inhibition results in more hyperpolarization leading to the complete inhibition of spiking in Neuron B (Figure 2.5d).
Figure 2.5. *Parametric modulation simulation in LabVIEW™*. The tonic activity of Neuron A modulates the activity of Neuron B. Voltage traces are shown for the neuronal activity without synaptic input (a), with tonic depolarization (b), tonic hyperpolarization (c) and stronger hyperpolarization (d). Each chart presents 5,000 iterations of simulated neuron outputs.

Perturbation results from a transient phase-resetting stimulus from the presynaptic neuron to the post-synaptic neuron. This is how coordinating neurons can coordinate firing synergies between bursting neurons. Under free run conditions with no connection between Neuron A and Neuron B, A is intrinsically silent and B bursts regularly (Figure 2.6a). With an excitatory synapse connecting Neuron A to B, transient spikes in Neuron A advance the phase of Neuron B’s bursting (Figure 2.6b). Transient presynaptic spikes generate a phase delay if Neuron A projects an inhibitory synapse to Neuron B (Figure 2.6c).
Figure 2.6. *Perturbation simulation in LabVIEW™.* Transient activity of Neuron A modulates the activity of Neuron B. Voltage traces are shown for the neuronal activity without synaptic input (a), with an excitatory transient input leading to a phase advance (b), and with an inhibitory transient input leading to a phase delay (c). Each chart presents 10,000 iterations of simulated neuron outputs.

The methods described in this chapter will be revisited throughout the chapters to follow. LabVIEW™ is very effective for rapid prototyping of neuron networks and results from software simulations will be presented in upcoming chapters. Nevertheless, the ultimate goal of this work is robotic implementation so the software modeling is considered to be an early step in the biorobotic process.
Chapter 3. Central pattern generation

A. Introduction

Electrophysiology has elucidated the central pattern generator (CPG) circuitry of a range of neuronal networks (Buchanan & Grillner, 1987; Thompson & Watson, 2005). Early work describing these neuron circuits focused on a variety of model systems, including the decapod stomatogastric ganglion (Maynard & Selverston, 1975), crayfish swimmerets (Hughes & Wiersma, 1960), Tritonia swimming (Getting, 1989), and locust flight (Wilson, 1961). Much of our direct understanding of the decapod walking circuits has been developed using in vitro preparations (Chrachri & Clarac, 1990; Stein, 1978). Here we describe the development and simulation of a CPG model underlying lobster walking. The cellular basis of the lobster walking CPG has yet to be fully explored using detailed neurophysiological study. This robotic implementation is endeavored to test plausible hypotheses and direct further neurophysiological investigation. Lobster walking represents an easily examined motor output that has distinct behavioral modes and its historical study has suggested a working model (Ayers, 2004; Ayers & Crisman, 1993; Ayers & Davis, 1977a). First we present biological evidence for the underlying CPG circuitry. Then we present computational modeling and robotic implementation of the CPG hypothesis.

While this computational work is valuable to help refine our understanding of these systems, a purely software-based approach to modeling can misrepresent environmental physics, oversimplify neural activity and motor outputs, or be bounded to produce positive results. Thus, our end goal is to run embodied robotic simulations to further validate models operating in real-world environments (Beer, Chiel, Quinn, & Ritzmann, 1998; Webb, 2002; Westphal, Rulkov, Ayers, Brady, & Hunt, 2011). Previous robotic simulations of CPGs have generated serpentine
locomotion (Wu & Ma, 2010), 6-legged (Inagaki, Yuasa, & Arai, 2003) and 8-legged walking (Witting & Ayers, 2007), fish swimming (L. Wang et al., 2005), and lamprey swimming (Westphal et al., 2011).

B. Lobster walking system

i. Walking pattern generation

The body plan and limb structure of the lobster is outlined in Figure 3.1. Ayers and Davis (1977a) describe the three limb joints involved in omnidirectional lobster locomotion. The thoraco-coxal (THC) joint displaces the leg in the frontal plane. Protraction occurs when the limb moves anteriorly while retraction results in posterior motion. The coxo-basal (CB) joint displaces the leg in the transverse plane. Elevation occurs when the limb lifts off of the substrate and depression results in downward movement of the leg. The mero-carpopodite (MC) joint also displaces the leg in the transverse plane and produces extension or flexion of the leg, motions frequently employed during lateral walking (Ayers & Davis, 1977a). Motor neuron innervation has been well described in crayfish (Elson, 1996). For example, there are 6 efferent neurons to each extensor and flexor muscle on each side of the abdominal segment, 5 are motor neurons and 1 is a peripheral inhibitor (Kennedy, Evoy, & Hanawalt, 1966). See Figure 3.2 for schematic diagrams that illustrate how the walking leg moves around each of these joints.
Figure 3.1. *External anatomy of the lobster*. (a) Dorsal view of the lobster showing overall body plan. (b) Anterior view of a leg showing the segments involved in locomotion. For joint structure and function see Figure 3.2. (c) Transverse section of the organism showing the legs in relation to the body. (from Macmillan, 1975)

Figure 3.2. *Movements of the lobster’s walking legs*. (a) Top view of the lobster carapace showing movement about the thoraco-coxal (THC) joint. (b) Rear view of lobster showing movement about the coxo-basal (CB) joint. (c) Rear view of lobster showing movement about the mero-carpopodite (MC) joint. (from Ayers & Davis, 1977a)
Coxopodite and basipodite movement is produced by eight muscles. The coxopodite is moved by the coxal protractor, which inserts anterior to the THC joint and by two coxal retractors that insert posterior to the joint. These muscles originate on the inner walls of the endophragmal skeleton and insert on the apodemes of the coxopodite. The basipodite is controlled by two classes of muscles. One class originates in the coxopodite and includes the posterior basipodite elevator and depressor. The second class originates in the endophragmal skeleton and passes through the coxopodite. This class, which may contribute indirectly to coxopodite movement, includes the anterior basipodite elevator and depressor along with the ventral basipodite elevator (Ayers & Davis, 1977a).

Using electromyograms and a quantitative analysis of leg joint angle movement of lobsters, Ayers and Davis (1977a) were able to identify three classes of walking muscles:

1. *Return stroke muscles* produce bursts of constant duration regardless of step frequency.
2. *Power stroke muscles* produce burst durations that vary linearly with step frequency.
3. *Bifunctional muscles* display discharge characteristics of return or power stroke muscles depending on the direction of walking.

The authors described the plasticity of motor units to be used for different behaviors, for example, lateral versus forward walking (Ayers & Davis 1977a). Movement and muscle synergies are directed by the interneurons of the lobster walking CPG (Ayers, Crisman & Massa, 1992).

Organisms from across the animal kingdom use local neuron networks called central pattern generators (CPGs) to drive locomotion (for reviews see: Clarac & Pearlstein, 2007; Selverston, 2010). A central pattern generator is a small network of neurons capable of producing a rhythmic output in the absence of sensory inputs (Kennedy & Davis, 1977). The
command neuron, coordinating neuron, and CPG model of neural organization is particularly well supported by research on decapod locomotion (Kennedy & Davis, 1977; Stein, 1978). There are a variety of central and peripheral factors which dictate behavior and these are consolidated within the CPG structure described to elicit controlled and coordinated locomotion (Macmillan, 1975; Ayers & Davis, 1977a). Exteroceptive and proprioceptive inputs play important roles in eliciting reflexes and compensatory movements necessary for this coordinated behavior (Ayers & Davis, 1977b). Locomotion in decapods is organized by a CPG system, coordinated between limbs, and modulated by command neurons and various sensory inputs.

The rhythmic outputs of neuronal oscillators can produce cyclic walking movements in all directions but behavioral shifts result from a change in the coordination of some bifunctional joint motor units relative to a common return stroke pacemaker (Ayers & Davis, 1977a). Distinct coordination modes of the MC joint were identified for different locomotory acts (Ayers & Clarac, 1978; Ayers & Davis, 1977a). Ayers and Davis (1977a) examined the synergistic and antagonistic relationships between leg muscles and found that some muscle groups, such as those responsible for retraction and depression, are synergistic for forward walking and antagonistic for backward walking. Also, some movements that are synergistic for lateral walking on the leading side are antagonistic for lateral walking on the trailing side (Ayers & Davis, 1977a). Figure 3.3 displays the relative movements of the different joints for different walking treatments, demonstrating some of these synergistic and antagonistic relationships within the joints. Ayers and Davis (1978; Ayers & Davis, 1977a) found that cyclic elevation and depression movement about the CB joint underlies walking in all directions.
Limb elevator motor neurons act as the central pacemaker of the walking system. The elevator/depressor oscillator is coupled to the muscles operating the other joints, connections dependent on the direction of walking (Ayers, Crisman & Massa, 1992). Specifically, the bifunctional coxal protractor discharges with the basipodite elevators to produce return stroke during forward walking but with the basipodite depressors to elicit power stroke during backward walking (Ayers & Davis, 1977a).

Ayers and Clarac (1978) investigated the mechanism behind MC joint movement in more detail. Using electromyograms from extensor, flexor, and accessory muscles, they observed muscle co-activation of the MC joint during locomotion in lobsters. During lateral walking they

Figure 3.3. Joint movements in the third right walking leg. The points represent joint angle measurements from single film frames, normalized to phase in the step cycle for forward, backward, leading and trailing walking. Note the relationship between retraction muscle groups (as joint angle decreases for the THC joint) and depression groups (as joint angle approaches 0° for the CB joint). Elevation and depression underlies walking in all directions. (from Ayers & Clarac, 1978; Ayers & Davis, 1977a).
observed strong alternating discharge in antagonistic muscles. Forward and backward walking presented co-activation of antagonistic muscles and a restricted angular range of joint movement. Discharge frequency varied with the mode of walking. Mechanistically, extension of the MC joint occurred by contraction of the merocarpopodite extensor muscle, which is controlled by two excitatory motor neurons and a peripheral inhibitor. The findings of their study further supported the idea that presynaptic inhibition sculpted omnidirectional walking. Walking direction and the underlying unit coordination can change on a cycle-by-cycle basis. Three modes of MC joint function were identified: postural mode during forward and backward walking; flexion power stroke in leading limbs during lateral walking; and extension power stroke in trailing limbs during lateral walking (Ayers, 2004; Ayers & Clarac, 1978).

The lobster walking CPG is comprised of three main movement phases and four main motor synergies (Figure 3.4) (Ayers, 2004; Ayers & Crisman, 1993; Ayers & Witting, 2007). The early swing movement phase occurs during the first portion of the return stroke as the leg elevates. During late swing, the leg is moved to the substrate and the swing phase terminates. During stance phase, gravitational force is counteracted and propulsion is generated (i.e. the power stroke) (Ayers & Davis, 1977a). The Elevator synergy is a neuronal oscillator, bursting at regular intervals and driving the rhythmic output of the whole circuit. It provides inhibitory synaptic input to the Depressor synergy and more slowly decaying inhibition to the Stance synergy, which in turn inhibits the Swing synergy. The delay between the initiation of the depression burst and the stance burst forms the late swing phase of the step cycle.
Based on these motor synergies, a 4-neuron pattern generating network was proposed (Figure 3.5) (Ayers & Crisman, 1993). The elevator CPG neuron pool excites motor neurons to initiate elevation of the CB joint while inhibiting depressor and stance neurons. Depressor CPG neurons excite motor neuron pools that elicit depression of the CB joint forming the antagonistic neural structure which drives the oscillating movement of the CB joint during walking. The swing and stance oscillators control the rhythm of the muscles at the THC and MC joints by innervating the corresponding motor neurons (Ayers, 2004; Chrachri & Clarac, 1987; 1989; 1990).
Figure 3.5. Proposed lobster walking CPG. A neuronal oscillator comprised of four neurons drives motor neurons that activate muscles. Command neurons presynaptically inhibit certain CPG-MN connections to gate on the appropriate motor output for a particular mode of walking. Labeled circles represent neurons, connecting lines are synapses: filled circles, inhibitory; filled triangles, excitatory. See text for a complete explanation. (Ayers, 2004; from Ayers & Crisman, 1993)

ii. Command neurons

Ayers (2004) explains that “command systems specify the coordination pattern through modulation of specific connections to specify walking direction.” In addition to their influence on direction-dependent circuit wiring, command neurons also specify the amplitude of propulsion through the recruitment of pools of motor neurons (Ayers, 2004). In the crayfish, Bowerman and Larimer (1974b) found that stimulation of walking command fibers evoke the complete motor rhythm. Variable latency was observed between stimulus onset and leg movement initiation, suggestive of variance between specimens and of the variety of other active neural inputs (Barnes, 1975). Bowerman and Larimer (1974b) concluded that there must be at least five interneurons present at the circumesophageal connective that are capable of evoking
forward walking. No walking drives were observed before the introduction of the walking wheel for the crayfish to interact with, demonstrating the importance of proprioceptive input for command fiber-evoked locomotory behavior (Bowerman & Larimer, 1974b). Although experiments involving ablation of chordotonal organs suggest otherwise (Fourtner & Evoy, 1973).

In the proposed lobster walking CPG (Figure 3.5), command neuron inputs presynaptically inhibit the synapses from pattern-generating neurons to MNs to sculpt the coordination patterns between synergies characteristic of a particular walking direction [Forward, Backward, Sideways Leading, Sideways Trailing] (Ayers & Davis, 1977). Presynaptic inhibition is rife in the thoracic ganglia (Clarac, Manira, & Cattaert, 1992; Nakagawa & Mulloney, 2001). The Forward command neuron, for example, inhibits the synaptic connection between Stance and Protractor MNs and between Swing and Retractor MNs to sculpt the motor pattern for forward locomotion (Figure 3.5). The command neuron also provides excitation to the propulsive source synergy, i.e. the retractor for forward walking, which varies the motor neuron spike outputs to modulate leg displacement, and thus walking speed. Backward walking is produced as the Backward command inhibits the connections from Swing to Protractor and Stance to Retractor and recruits protractor motor neurons (Ayers, Crisman & Massa, 1992).

Neural outputs for the CPG and MNs during forward walking from a LabVIEW™ simulation are shown in Figure 3.6. Chapter 2 explains the methodology behind these software simulations. A Forward command neuron gates on the muscle synergies to generate forward walking via presynaptic inhibition of Stance-Protractor and Swing-Retractor synapses (see Figure 3.5). The Elevator MNs and Protractor MNs begin firing at the same time, activating the corresponding muscles to lift the lobster leg off of the ground and move it anteriorly. The
Elevator MN stops firing and the Depressor MN fires, bringing the leg down to contact the ground, a movement that overlaps the end of the Protractor MN burst. The Retractor MN begins firing as activity of the Protractor MN ceases, and the lobster leg is moved posteriorly while in contact with the ground. This part of the gait cycle generates forward propulsion. The cycle begins again as the Depressor and Retractor MN’s stop firing and a new step begins.

Figure 3.6. Output of lobster walking CPG for forward walking. Voltage traces from a LabVIEW™ simulation of the proposed CPG network from Figure 3.5. The step cycle begins (i) as the elevator MN and protractor MN fire. The leg is elevated and moved anteriorly (ii) during early swing and the depressor begins to fire during the late swing phase (iii). The retractor fires, moving the leg posteriorly (iv) to generate forward movement. The Forward Command neuron recruits the Retractor and gates off Stance-Protractor and Swing-Retractor synapses to generate the observed Protractor-Retractor muscle synergy.

During backward walking as observed in the second half of the traces in Figure 3.7, the muscle synergies reverse: the Elevator MN fires concurrently with the Retractor MN, and the
Depressor MN with the Protractor MN. The Backward command recruits the Protractor and gates off the Swing-Protractor and Stance-Retractor to produce the observed Protractor-Retractor muscle synergy. During both forward and backward walking, the Extensor and Flexor MN’s are coactivated by both the swing and stance synergies to produce postural activation of the muscles.

Figure 3.7. *Neural activity for forward and backward walking.* The MN voltages from a LabVIEW™ simulation are shown for a transition from forward walking to backward walking as denoted by the command neuron inputs at the top. See text for a description of network operation.

The neuronal outputs for a bout of sideways leading walking are shown in Figure 3.8. The Protractor and Retractor MN’s are tonically excited to stiffen the thoraco-coxal joint. Lateral propulsion is achieved by alternating activation of the Extensor and Flexor MN’s in synergy with leg elevation. The Elevator and Extensor MNs are co-activated which leads to extension of the leg during the swing phase. The Flexor generates the propulsive force during stance phase. This
stepping pattern generates lateral leading walking. In the second half of the simulated voltage traces in Figure 3.8, the Forward command is added to the Leading command and the result is a diagonal walking pattern. The Protractor MN and Extensor MN lift concurrently during elevation, moving the leg out and anteriorly during the swing phase of the step cycle. During the stance phase, the Retractor MN and Flexor MN are activated, generating a diagonal vector of propulsion, anterior and lateral in relation to the body.

Figure 3.8. Neural activity for sideways and diagonal walking. The MN voltages from a LabVIEW™ simulation are shown for a transition from sideways walking to diagonal walking that occurs when the Forward and Leading commands are both active.

iii. Coordinating neurons

Coordination between control centers is accomplished by coordinating neurons that can be initiated by centrally derived signals, sensory signals, or a combination of the two (Kennedy & Davis, 1977; Namba & Mulloney, 1999). The neuronal and synaptic organization and dynamics of coordinating mechanisms are not very well understood, particularly in walking systems (Smarandache et al., 2009). We consult research on the intersegmental coordination of crayfish swimmerets as it is one of the most well described invertebrate coordinating systems.
(Namba & Mulloney, 1999; Smarandache et al., 2009). Similarities exist between crayfish swimmeret coordination and coordinating connections underlying cockroach walking (Pearson & Iles, 1973), thus general principles of the swimmeret system may be involved in lobster leg coordination.

There are several types of interneurons that connect abdominal segments of the crayfish to entrain the phasic beating of swimmerets (Mulloney & Hall, 2007), and these neurons are not dependent on sensory feedback (Stein, 1971). Local commissural interneurons (called ComInt 1 neurons, one per CPG) receive excitatory inputs from the CPGs of the other abdominal ganglia and transmit it to neurons in the local CPG (Smarandache et al., 2009). The excitatory synaptic inputs modulate the timing of power stroke activation between segments. And while there are excitatory connections from all abdominal ganglia to a local CPG’s ComInt 1 neuron, the most adjacent inputs are the strongest, resulting in a segmental gradient of synaptic strengths (Smarandache et al., 2009).

Synchronization of limb movements requires constant movement periods but modulated phases that define natural gait patterns (Smarandache et al., 2009). The interganglionic connections described in the crayfish swimmeret system presumably underlie coordination and have been shown in a range of other model organisms (for a review see: Skinner & Mulloney, 1998). In the crayfish walking circuit, circumstantial evidence of coordinating neurons between limbs has been demonstrated (Libersat, Zill, & Clarac, 1987; Müller & Clarac, 1990a). Stimulation of the dactyl sensory nerve of a crayfish leg excites promotor and elevator muscles while inhibiting antagonistic retractor and depressor muscles (Libersat, Zill, & Clarac, 1987). Unstimulated neighboring legs displayed opposite motor outputs, thus supporting the role of intersegmental coordinating connections (Libersat, Zill, & Clarac, 1987; Müller & Clarac,
Usually neurons driving coordination are distinct from pattern-generating interneurons as is the case in the crayfish swimmeret system (Smarandache et al., 2009) and the leech heart circuit (Masino & Calabrese, 2002). However, the leech swim circuit shows evidence of direct axonal projections between pattern-generating neurons across segments (Cang & Friesen, 2002; Friesen & Pearce, 1993). On RoboLobster, we implemented direct reciprocal inhibition between Elevator neurons of contralateral and ipsilateral CPGs (Figure 3.9). We implemented ipsilateral inhibition that is stronger than contralateral inhibition as observed in rock lobsters and other arthropods (Clarac & Chasserat, 1986).

The network hypothesis is necessarily simple given the relative lack of information about the biological system, the variety of biological evidence from other motor systems, and some technical limitations on the hardware platform. The model omits distinct coordinating neurons separated from the pattern-generating neurons, axonal projections beyond adjacent segments (e.g. coupling between diagonally neighboring walking legs (Müller & Cruse, 1991), and proprioceptive inputs. Nevertheless, the network outputs coordinated elevator activation between legs as seen in a LabVIEW™ simulation (Figure 3.10). A metachronal gait is observed resulting from sequential activation of the Elevator neurons down the body and alternating activation of the contralateral Elevator neurons at each segment. The specifics of the lobster walking coordination system are not well described and require more investigation to further refine the model.
Figure 3.9. *Walking coordination network*. Reciprocal inhibition between adjacent Elevator neurons across segments and along the body generate coordinated neuronal output across all 8 walking leg CPG’s. See text for a description of the network’s rationale.

Sensory feedback may not be necessary for CPG activity underlying decapod locomotion, but it can have a major influence on behavioral output. Proprioceptive sensory inputs coordinate muscle synergies and serve to orchestrate intralimb coordination (Bush, 1965). Many studies have suggested that proprioceptive feedback is necessary to coordinate motor outputs but variation between systems is apparent (Sillar et al., 1987; Chrachri & Clarac, 1990; Buscheges et al., 1995). In crayfish walking legs, proprioceptive input by a single organ, the thoracic-coxal muscle receptor organ, can entrain fictive motor outputs in several ipsilateral CPGs (Sillar, Clarac, & Bush, 1987). However, in hawkmoths fictive locomotion is generated without sensory inputs: deafferented thoracic ganglia generate a patterned tripod gait (Johnston & Levine, 2002). Implementation of proprioception on RoboLobster could help with response to perturbation and
to provide more robust intersegmental coordination but it does not seem necessary to produce coordinated walking.

In addition to the variability of neural organization across locomotory modes demonstrated by Ayers and Clarac (1978), there exists substantial variability of output within modes. Barnes (1975) found that stepping order of limbs on the leading and trailing sides of crabs varies quite remarkably. Stepping parameters also change with walking speed. Metachronal coordination is observed in trailing legs at low speeds of lateral walking and the burst duration of the power stroke is reduced during high walking speeds (Barnes, 1975). In lobsters there is a preferred stepping order but considerable variation from the dominant pattern is observed (Macmillan, 1975). Differences exist between coordination of forward and backward walking. Metachronal organization in crayfish occurs from front to back but the contribution of each leg to the walking output is unique (Jamon & Clara, 1995).

In RoboLobster, two gaits emerge: one is metachronal and one is a strict tetrapod gate in which four limbs move completely in phase: Left legs 1 & 3 and Right legs 2 & 4 are in phase with each other and completely out of phase with the other four legs (see Figure 3.10 for leg numbering). The specific reason for this variable motor output on RoboLobster is unclear but likely results from variable command and sensory inputs as well as random noise that is injected into the Elevator neurons of each segment. In the biological system, variable walking output is indicative of the high number of inputs into the neural system. Much of the modification of the locomotory CPG by command and coordinating neurons is derived from sensory inputs, which are described in Chapter 4 and 5.
Figure 3.10. **Coordinating network neuron output.** The Elevator neuron outputs for RoboLobster’s 8 walking CPGs (at each leg).

**C. Neuron network operation on RoboLobster**

User-controlled injected current into RoboLobster’s command neurons elicited forward, backward and sideways walking of the robot. Figure 3.11 shows kinematic tracking of the third legs of a lobster (Figure 3.11a) and the robot (Figure 3.11b). Several walking characteristics were calculated for both systems (Table I). While there were differences between robot and animal in the duration of stepping period and ground contact, and in step length, both generated forward body locomotion.
Figure 3.11. Kinematic tracking of leg movement from side-view video recording. a. Lobster 3rd leg step tracking. b. RoboLobster 3rd leg step tracking. In both, blue points (top) track the ‘MC’ (mero-carpopodite joint) and green points (bottom) are the dactyl.

Table I. Comparison between lobster and robot leg movement.

<table>
<thead>
<tr>
<th></th>
<th>Lobster</th>
<th>Robot</th>
</tr>
</thead>
<tbody>
<tr>
<td>Stepping period</td>
<td>2.63 s</td>
<td>1.87 s</td>
</tr>
<tr>
<td>Continuous ground contact</td>
<td>1.6 s</td>
<td>1.1 s</td>
</tr>
<tr>
<td>Step length</td>
<td>10.34 cm</td>
<td>5.33 cm</td>
</tr>
<tr>
<td>Elevation height (% of leg height)</td>
<td>11.2%</td>
<td>41.7%</td>
</tr>
<tr>
<td>Forward speed of body</td>
<td>3.35 cm/s</td>
<td>3.25 cm/s</td>
</tr>
</tbody>
</table>

There are three different ways that a lobster is thought to adjust walking speed: 1) changes in step frequency, 2) changes in stepping amplitude, and 3) changes in leg geometry during stepping (Domenici, Jamon & Clarac, 1998; Dürr & Ebeling, 2005). When walking speed changes, the swing phase remains roughly constant and the stance phase duration varies as is observed in the animal (Ayers & Davis, 1977a). We explored changes in stepping frequency and in stepping amplitude as a way to change walking speed. Bilateral differences in walking speed result in curve walking (yawing during forward locomotion) (Domenici, Jamon & Clarac, 1998).

The CPG neural network was run on RoboLobster’s Texas Instruments (TI) Digital Signal Processor (DSP). The same neural network configuration (Figure 3.5) was run as in the LabVIEW™ simulations but the programmed code was translated into procedural C. See Chapter 6 for a detailed overview of the DSP board and the programmed code on RoboLobster.
Results from the nervous system operating on the DSP produce qualitatively similar phasic MN outputs as the LabVIEW™ simulation (Figure 3.12). Direct quantitative comparisons of discharge frequency and absolute timing between LabVIEW™ and DSP simulations are not feasible due to differences in calculation rate. The CPG network simulation is calculated over 70 times faster in LabVIEW™ (108,107 iterations per second) compared to the DSP (1,447 iterations per second). A delay of 1ms per iteration is implemented in LabVIEW™ to allow both simulations to use \( \mu \) values of a similar magnitude. The \( \mu \) value determines the step size of the DTM equations described in Chapter 2 [for LabVIEW™ neurons, \( \mu = 0.0001 \); for DSP neurons, \( \mu = 0.000165 \)].

![Fig 3.12](image)

**Figure 3.12. Walking output generated by RoboLobster’s CPG.** Output of the nervous system simulation running on RoboLobster’s DSP during forward walking. An 800 ms walking period is produced when the elevator neurons in the network use \( \alpha = 4.75 \) and \( \sigma = -0.1 \) in the DTM model.

i. Changes in step frequency

To accomplish changes in step frequency, the \( \alpha \) and \( \sigma \) parameters of the elevator neurons DTM equations can be adjusted. This would be analogous to changes in neuron dynamics as a
result of neuromodulation (Destexhe & Marder, 2004; Erber, Kloppenburg, & Scheidler, 1993). When $\alpha = 4.75$ and $\sigma = -0.1$ for all elevator neurons, a fast walking pattern is produced with a period of 800 ms (Figure 3.12). Changes to the $\alpha$ and $\sigma$ values changes the walking period (Figure 3.13). When $\alpha = 6.0$ and $\sigma = -0.38$ the walking period is 1600 ms (Figure 3.13, top). With $\alpha = 6.8$ and $\sigma = -0.6$, a slower walking pattern is produced with a period of 2900 ms (Figure 3.13, bottom).

![Figure 3.13. Simulated neuromodulation of RoboLobster’s CPG. Output of the nervous system simulation running on RoboLobster’s DSP during forward walking. Top. A 1.6s walking period is produced when the elevator neurons in the network use $\alpha = 6.0$ and $\sigma = -0.38$ in the DTM model. Bottom. A 2.9s walking period is produced when the elevator neurons in the network use $\alpha = 6.8$ and $\sigma = -0.6$.](image)

This simulated neuromodulation was not embodied on RoboLobster because changes in $\alpha$ and $\sigma$ during network operation resulted in more than five seconds of unpatterned motor neuron spike activity. This transition between walking speed was considered to be too slow for robotic
implementation but adjustments to the DTM model could be made to accommodate this approach in the future.

ii. Changes in step amplitude

We relied on command neuron recruitment of the motor neurons active during the stance phase synergy to increase the power stroke amplitude and consequently the walking speed of RoboLobster. For example, the Forward command neuron recruits the Retractor MN (see Figure 3.5). With increased spiking rates of the Forward command, the Retractor MN receives increased excitation. Experiments were run to determine if this command neuron recruitment model could drive changes in retractor actuator activation and walking speed.

The MN outputs of the CPG operating on the DSP board were recorded using an oscilloscope. For these and the remaining experiments in this chapter, neuron spikes drove current pulses directly: one spike generated a current pulse for 6 iterations (~6ms). Current was injected into the Forward command and the resulting spike outputs were analyzed. Command neuron spikes were printed to the DSP’s serial line and recorded. A lower level of injected current into the command neuron resulted in a lower spike rate of the command neuron (low current (0.1): Figure 3.14a; high current (1.5): Figure 3.15a). The walking period output by the CPG did not change as command input varied (low command input: Figure 3.14b; high command input: Figure 3.15b). The average number of spikes per burst of the Retractor neurons under low current conditions (19.1 ± 2.08 spikes; n=10) was significantly different (p<0.00001, t-test) than the average Retractor spike number with high command input (25.6 ± 1.71 spikes; n=10).
Figure 3.14. *CPG output with low command neuron input.* Spike outputs of DSP simulation of walking CPG (see Figure 3.5). **a.** Command neuron spikes produced with an injected current level of 0.1. The 500 iterations displayed (y-axis) correspond to approximately 0.25 seconds of the simulation. **b.** Motor neuron outputs showing walking pattern. Retractor spikes in the region marked by the red box were tabulated for analysis described in the text and presented in Figure 3.16. Note that spikes are inverted; i.e. a low signal represents a spike event.

Figure 3.15. *CPG output with high command neuron input.* Spike outputs of DSP simulation of lobster walking CPG (see Figure 3.5). **a.** Command neuron spikes produced with injected current level of 1.5. **b.** Motor neuron outputs showing walking pattern.
Figure 3.16. *Forward command recruitment of retractor motor neurons.* Oscilloscope snapshot of retractor spikes as depicted by the red box in Figure 3.14b. **a.** Representative spike output of the retractor MN when a current of 0.1 was injected into the Forward command. **b.** Representative spike output of the retractor MN when a current of 1.5 was injected into the Forward command.

The command neuron recruitment of the Retractor MN observed during simulated forward walking suggests that increased command activity may increase the magnitude of the propulsive force during actual walking. To test this, RoboLobster was operated in an open loop setup without sensory input. Current injected into the Forward command neuron was varied and the resulting walking speed was determined from overhead video recordings. With low current injected into the command neuron (see Figure 3.14a for example command neuron spikes and Figure 3.16a for example retractor neuron spikes), the average walking speed (n=2) was 0.28 cm/s. With high current injected into the command neuron (see Figure 3.15a for example command neuron spikes and Figure 3.16b for example retractor neuron spikes), the average walking speed (n=2) was 1.49 cm/s. One test using an intermediate injected current value of 0.6 generated an intermediate walking speed of 0.91 cm/s. Screen grabs of the overhead video recordings of RoboLobster’s behavior during three of these trials are shown in Figure 3.17.
Figure 3.17. *Video data of RoboLobster’s command neuron modulation.* Three trials showing RoboLobster’s walking progress under different levels of current injected into the Forward command neuron. **a.** With low-level current (0.1), RoboLobster walked from the position in the Start frame to the position in the End frame. It covered 9.9 cm in 26 s. **b.** With mid-level current (0.6), RoboLobster traveled 23.5 cm in 26 s. **c.** With high current (1.5), RoboLobster traveled 28.7 cm in 16 s.

D. Discussion

We have shown the complete implementation of a lobster walking CPG model in LabVIEW™ software and in procedural C on RoboLobster’s DSP. An overview of the hardware and programmatic code used is provided in Chapter 6. The model was developed from existing knowledge and refined using computational modeling in LabVIEW™ software. The biomimetic implementation of a walking CPG represents an important step for neurally-controlled robotics. Neuronally-implemented central pattern generation represents a potentially robust and adaptable
motor control regimen that can capture some of the dynamical capabilities of the animal system.

More quantitative comparison between lobster and robot motor outputs is necessary to inform the refinement of the model. To accommodate the nitinol actuator, the robot was designed to be the size of a large lobster and it would be useful to make kinematic comparison to equivalently sized animals. The elevation height of the lobster leg was much smaller than that of the robot, perhaps due to differences in muscle activation. During forward locomotion, there seems to be some movement about the MC joint in the lobster but not in the robot. Perhaps the co-activation described by Ayers and Clarac (1978) is not symmetrical. Three-dimensional tracking of leg movement should be completed to better understand the movement about the MC during different walking modes. Armed with more detailed information about co-activation of the MC joint during forward walking, the walking CPG underlying RoboLobster’s walking could be further developed.

Armed with the plausible implementation of this CPG hypothesis, we can now move on to extend the nervous system model to include sensory feedback. The following chapters will outline the development and simulation of exteroceptive sensory networks that drive these walking CPGs to produce closed-loop behavior.
Chapter 4. Optical flow reflexes

A. Terrestrial arthropods

i. Optical flow exteroception

Arthropod visual processing is an exteroceptive sensory system driving locomotion that has been particularly well-studied (M. Srinivasan & Zhang, 2004). Although the resolution of vision in decapods is unclear, the reflexes elicited by visual stimuli have been studied extensively (Bethe, 1898; Davis & Ayers, 1972). Input from optokinetic stimuli in lobsters allows for coordination of their movement in relation to their external medium. In 1898, Bethe demonstrated that crabs rotate in place when presented with a rotating striped circular arena. The angular optical flow presented causes the crab to walk forward on one side of the body and backward on the other. More recent studies have focused on the mechanism of curve walking in crayfish as an optomotor response. Curve walking is produced by higher step frequencies (temporal asymmetries) and larger step amplitudes (positional asymmetries) of the outer legs during locomotion. The mechanism for curve walking is driven by the action of one principal leg. Domenici et al. (1998) demonstrated that yaw during curve walking in two species of crayfish is directed by the power stroke of the fourth leg on the outside of the turn. Cruse and Saavedra (1996) found that when crayfish are presented with vertical stripes moving in a horizontal direction, curve walking occurs as the result of an increase in outer leg step amplitude.

Translational optical flow elicits behavior that sustains locomotion; that is, forward or backward walking is initiated to compensate for forward or backward optical flow (Davis & Ayers, 1972). In crayfish there are several types of visually reacting...
neurons as described by (Wiersma & Yamaguchi, 1967). One type reacts tonically to
general illumination with sustaining fibers increasing their output as light increases while
dimming fibers are inhibited by light. Another class of visually responsive neuron
responds to moving objects. Movement fibers are unidirectional flow sensitive neurons in
the optic nerve of the crayfish and other decapods that play an important role in the
coordination of locomotion (Wiersma & Yamaguchi, 1967).

The movement of the visual world across an animal’s eye, or optical flow,
detected by the movement fibers is an important sensory input driving the production of
coherent locomotion. The biologist Jacques Loeb introduced the tropism theory of animal
conduct which describes the behavior of symmetrical organisms driven by bilateral
sensors (Loeb, 1918). Braitenberg promoted the importance of decussating fibers:
sensory information that inputs contralaterally into non-homologous regions of the motor
system of an organism (Braitenberg, 1965). Using simple hypothetical vehicles
Braitenberg illustrated how layered decussating and ipsilaterally projecting sensors were
capable of mediating most positive and negative taxic behavior (Braitenberg, 1986).

A wide range of animal species display optical flow-mediated locomotory
reflexes (lobsters: (Davis & Ayers, 1972); honeybees: (M. Srinivasan, Zhang, Lehrer, &
Collett, 1996); humans: (Jouen, Lepecq, Gapenne, & Bertenthal, 2000)). In walking
arthropods such as crustaceans, the ethological and neurophysiological bases for
optomotor responses have been well studied. A crab exposed to the angular rotation
generated by a rotating striped circular arena will turn in place in an attempt to stabilize
its surroundings (Bethe 1898). Additionally, a lobster placed on a platform within moving
striped bars will walk forward when exposed to translational optical flow from front to
back and backward under anterior translational flow (Davis and Ayers 1972). The walking direction and speed of the lobster depends on the respective characteristics of the optical flow (Davis and Ayers 1972). Combinations of translational and angular flow elicit yawing movements to correct for disturbances during locomotion (Figure 4.1).

![Diagram of optical flow reflexes](image)

Figure 4.1. *Overview of arthropod optical flow reflexes.* Black arrows denote direction of optical flow stimulus. Activation of forward and backward command neurons for both sides of the animal are shown with green shading. (from Ayers & Rulkov, 2008; Blustein & Ayers, 2010a)

Visually-mediated obstacle avoidance is not well-described in lobsters or other terrestrial arthropods so we turn to flying insects to inform the conserved model. Evidence shows that flies and bees turn away from areas of perceived high optical flow (M. Srinivasan, 2011; M. V. Srinivasan, Zhang, & Chandrashekara, 1993; van Breugel & Dickinson, 2012). Visually-mediated obstacle avoidance behaviors in flying arthropods including the centering response in bees (M. Srinivasan & Zhang, 2004), saccades in flies (Frye & Dickinson, 2004), and correctional steering maneuvers in locusts (Rowell, 1988) are another optical flow reflex driving behavior. A centering response occurs in bees as they fly through a tunnel: they slow as the tunnel narrows or if optical flow is experimentally increased (Kirchner & Srinivasan, 1989). This behavioral observation was replicated using trained honeybees flying down a custom-built stimulus corridor at


Northeastern University’s Marine Science Center. The centering response occurs as optical flow is balanced bilaterally and suggests the presence of high threshold lateral flow detectors that mediate obstacle avoidance (Srinivasan et al., 1996). In a saccade, a fly turns away from a region of high optical flow or visual expansion (M. Dickinson, 2005). Locust lobula giant movement detector (LGMD) neurons responsive to looming objects are thought to utilize edge-detection by way of lateral inhibition (Rind & Bramwell, 1996) to elicit an avoidance response via the descending contralateral movement detector interneuron (DCMD) (O'Shea, Rowell, & Williams, 1974). Even though the avoidance of areas of high optical flow and of looming objects are distinct mechanisms that allow for successful locomotion, they represent functionally interrelated components of optomotor reflex networks. During forward locomotion, for example, a looming object will also present an area of increased translational optical flow.

As looming movement detectors have not been described in crustacea (Wiersma & Yamaguchi, 1967), we therefore propose an alternative hypothesis, that obstacle avoidance is mediated via high threshold optical flow detectors. This hypothesis was incorporated into a neural network explaining arthropod optical flow reflexes (Blustein & Ayers, 2010a) based on Kennedy and Davis’ model (1977). Observed behavioral responses can be correlated with the activity of previously identified neural components including direction-sensitive visual interneurons (Wiersma and Yamaguchi 1967) and single command neurons that initiate complete behaviors (Bowerman and Larimer 1974b) to devise the network diagram (Figure 4.2).
Figure 4.2. Optical flow neuron network hypothesis for walking arthropods. White circles are sensory neurons receiving optical flow information from the eye with arrows denoting directional sensitivity. Sensory neuron signals are passed through synapses (lines) to the command neurons (black circles). Lines with open triangle ends represent excitatory connections; filled circles ends, inhibitory connections. Optical flow sensory interneurons: LM, left medial; LLL, left low threshold lateral; LHL, left high threshold lateral; RM, right medial; RLL, right low threshold lateral; RHL, right high threshold lateral. Command neurons: LF, left forward; LB, left backward; RF, right forward; RB, right backward. (adapted from Kennedy & Davis 1977; Ayers & Crisman, 1992; Blustein & Ayers, 2010a)

ii. LabVIEW™ simulations

The neural network proposed in Figure 4.2 was simulated in LabVIEW™ software as outline in Chapter 2E. The neural simulation was run with a variety of different optical flow milieus. Under purely translational optical flow propagating from tail to head in the medial direction, the corresponding optical flow sensory interneurons fire and excite both backward commands (Figure 4.3). This output matches the observed behavioral response under conditions of medial optical flow resulting from a rear to front translational optical flow stimulus (Davis & Ayers, 1972).
Figure 4.3. *Medial translational flow initiates backward walking*. **Left panel.** Representation of optical flow neural network. White rectangles and circles represent active sensory interneurons and walking commands, respectively. Black arrows mark direction of optical flow and white arrows denote directional sensitivity of sensory neurons. For clarity, reciprocal inhibitory connections between opposing walking commands have been omitted in this and subsequent figures. **Right panel.** Neural activity of sensory interneurons and walking commands. Neural network and abbreviations are described in Fig. 2. (from Blustein & Ayers, 2010a)

In a lobster exposed to purely angular optical flow in the clockwise direction, medial left and low threshold lateral right sensory interneurons fire initiating forward walking on the left side and backward walking on the right (Figure 4.4). This causes the animal to rotate clockwise in place as described by Bethe (1898).
By integrating angular and translational optical flow, animals are able to avoid obstacles and maintain a desired course. If a forward-flying bee approaches an obstacle on its right side (Figure 4.5a), the high rate of optical flow will cause the corresponding high-threshold sensory interneuron to fire in the model (Figure 4.5c). This excites the ipsilateral forward command while simultaneously partially inhibiting the connection between the low-level lateral detector and the contralateral forward command resulting in an avoidance turn while maintaining forward flight (Figure 4.5).
Figure 4.5. Optical flow-mediated obstacle avoidance. a. Flight path of honeybee with a right side obstacle. Opaque bee marks point of activation of high threshold lateral detector and initiation of avoidance turn. b. Representation of optical flow neural network at location of opaque bee in a. White rectangles and circles represent active sensory interneurons and flight commands, respectively. Black arrows mark direction of optical flow, arrow thickness depicts relative magnitude of flow rate. White arrows denote directional sensitivity of sensory neurons. c. Neural activity of sensory interneurons and flight attitude commands represented in b. A high rate of optical flow on the right side activates RHL which excites RF and inhibits RLL. The activity of LF drops due to decreased excitatory input from RLL. Since the firing rate of RF becomes greater than that of LF at the location of the opaque bee in a, an avoidance turn is initiated. Neural network and abbreviations are described in Figure 4.2. (from Blustein & Ayers, 2010a)
Wind and water currents present additional optical flow situations that arthropods encounter. When a forward walking lobster is pushed off course by surge, a yawing response is observed (Figure 4.6). With surge coming from the left side, the lobster experiences translational plus angular optical flow on the ipsilateral side and translational minus angular flow on the side contralateral to the surge. This results in a compensatory turn into the surge in order to maintain the intended direction of locomotion. The neural activity of this reflex is outlined in the right panel of Figure 4.6.

Figure 4.6. Optical flow-mediated yaw response. **Left panel.** Walking path of lobster submitted to strong surge from the left. **Right panel.** Neural activity of sensory interneurons and walking commands. Numbers correspond to different phases of the response. At 1 the lobster is walking forward and there is steady lateral optical flow. The lobster encounters surge at 2 which results in the yaw response. After the lobster's compensatory turn, forward walking resumes at 3. Abbreviations are described in Figure 4.2. (from Blustein & Ayers, 2010a)
iii. *Robotic implementation*

While these models were developed for the implementation on RoboLobster (presented in Chapter 7), to more comprehensively investigate network dynamics, a wheeled robot was used (Figure 4.7). This wheeled robot was controlled by an Arduino Uno board attached to a Bricktronics motor shield\(^1\) that drove two LEGO™ NXT motors. The Arduino board ran a C implementation of the DTM network simulation (for details see Chapter 6). Neuron data was transmitted in real-time using an Xbee WiFi module.

![ArduBot platform](image)

**Figure 4.7. ArduBot platform.** Robot used to validate neuronal models underlying arthropod optical flow reflexes.

The robot was run through a 0.6 m wide long tunnel lined with alternating 13 mm wide black and white vertical stripes. The tunnel was 2.5 m long and the walls at the middle section of the tunnel were replaced with computer monitors displaying visual stimuli that could be manipulated (Figure 4.8). The impetus for this set up was Srinivasan

\(^1\) [https://store.wayneandlayne.com/products/bricktronics-shield-kit.html](https://store.wayneandlayne.com/products/bricktronics-shield-kit.html)
et al.’s (1996) important work describing honeybee visual reflexes using tunnels with moving walls. The neural model was subjected to the same visual conditions in order to compare the model’s performance with that observed by Srinivasan et al. in honeybees.

Figure 4.8. *ArduBot testing arena*. Photograph of ArduBot traversing testing arena. Walls are covered in stationary black/white striped pattern. The middle section walls are replaced with monitors that display moveable visual stimulus.

When translating out of LabVIEW™ simulation and into robotic testing, the self-generated optical flow must be considered. When the model from Figure 4.2 was tested, a positive-feedback loop of optical flow inputs led to erratic behavior (Figure 4.9). As the robot turned, optical flow increased and the turning response increased.
Figure 4.9. *ArduBot path without corollary discharge.* The robot, traveling from right to left, is marked by a vector with a dot tracking the nose and the direction denoting heading. Each vector represents one second of travel. Optical flow movement displayed on the monitors in the middle section of the tunnel is depicted by the arrows. With no corollary discharge, the robot turned sharply upon encountering the optical flow stimulus.

In biological systems, neural signals from motor systems that inhibit sensory inputs are called corollary discharge or efference copy (Holst & Mittelstaedt, 1950; Poulet & Hedwig, 2003; Sperry, 1950). Corollary discharge neurons were added to the network model such that self-directed turning inhibited optical flow inputs (Figure 4.10). The Forward command on each side excites an ipsilateral Rotation interneuron and inhibits a contralateral Rotation interneuron. The Rotation interneurons inhibit the high and low threshold ipsilateral lateral optical flow sensory neurons.
The network from Figure 4.10 was run on the ArduBot and without any moving optical stimulus, a centering response was observed. A representative path is shown in Figure 4.11.
Figure 4.11. *Centering response in ArduBot.* With no optical stimulus, the robot outputs a centering response as observed in honeybees (M. Srinivasan et al., 1996). Figure markings are as in Figure 4.9.

When subjected to different optical flow environments, the robot responded similarly to the honeybee (M. V. Srinivasan, Lehrer, Kirchner, & Zhang, 1991) (Figure 4.12). An optical flow stimulus moving from front to rear translated into increased perceived optical flow. The robot, as observed in the bee, tended to turn away from the side with higher optical flow while traversing the tunnel.
Figure 4.12. *Optical flow reflexes in ArduBot.* **a.** The robot turned away from the increased optical flow (front to rear) on the left (bottom) and towards the decreased optical flow (rear to front). **b.** The robot turned away from the increased optical flow on the right (top) and towards the decreased optical flow. **c.** The robot turned away from the increased optical flow on the right (top) even when there was no modification of optical flow on the opposite side (bottom). Figure markings are as in Figure 4.9.
B. Flying arthropods

i. Optical flow reflexes and visual odometry

As in lobsters (Davis & Ayers, 1972), the rate of optical flow effects the level of motor output in flying arthropods. Changes in wing beat frequency in tethered locusts (Baader & Schäfer, 1992) and flight speed adjustments in free flying bees (Baird, Srinivasan, Zhang, & Cowling, 2005) are both modulated by optical flow. However, there are notable differences between flying and walking arthropods at the effector level (Wendler, 2002). Backward walking represents a change in the pattern of coordination relative to forward walking; direction of flight does not share this characteristic.

Transitions between forward and backward flight in insects occur through changes in the tilt of the wing stroke plane (Ellington, 1999). This difference is illustrated in locusts as progressive (ie. front to back) fields of optical flow initiate forward flight but regressive flow fields inhibit flight entirely (Baader & Schäfer, 1992).

Another difference between walking and flying arthropods is in the biomechanics of steering. Walking arthropods steer by independently controlling left and right sets of walking legs. Flight control though, is the result of the input of visual interneurons onto various elements of the motor system including the head, wings, abdomen, and halteres (Sherman & Dickinson, 2003). Production of yaw torque in flies is nearly passive as the wing's high angle of attack on one side creates high drag and a turn to that side (Ellington, 1999; Ristroph et al., 2010). There is also evidence for differences in wingstroke amplitude underlying yaw control (M. Dickinson, 2005). Furthermore, flying insects differ in the relationship between nerve impulses and wing beats. In neurogenic flight, a one-to-one relationship between neuron spike and wing beat is observed in such
insects as butterflies, cockroaches and locusts (Neville, 1963; Roeder & Weiant, 1950).

In myogenic flight, the neuron impulses modulate the frequency of wing beating regardless of individual spike timing, as in bees and flies (Nachtigall & Wilson, 1967; Pringle, 1949; Roeder, 1951; D. M. Wilson & Wyman, 1963). The work presented here deals with myogenic flight with nerve impulses modulating a pulse width-modulated signal to drive motor outputs.

Regardless of these differences in the nature of the effectors, the conserved features of walking and flying arthropod systems can be used to inform robotic implementation of optical processing using simple neural networks. The optical flow reflexes captured by the network in Figure 4.10 can be adapted for a flying insect control network (Figure 4.13). The network maintains the obstacle avoidance and optical flow perturbation reflexes and adds a visual odometer.

Figure 4.13. *Flying arthropod optical flow reflex network.* Neuron network governing visual odometry, obstacle avoidance and optical perturbation compensation reflexes.
Honeybees track flight distance using a visual odometer driven by optical flow, i.e. the movement of the visual world across the honeybee’s eye (M. Srinivasan et al., 1996). While the performance of the optical flow-mediated odometer has been well characterized in the biological system, its underlying neuronal mechanism is unknown (M. V. Srinivasan, 2014). Neuronal integration, or persistent neural activity, may be responsible for short term memory (Aksay et al., 2007) and this principle may be used to accumulate optical flow sensory information conveyed by unidirectional motion-sensitive optic neurons (Ibbotson, 1991). As optical flow accumulates in an integrating odometer neuron, a single neuron spike indicates when a targeted optical flow distance has been reached. This neural signal would act as a command neuron to transition from one behavioral act to the next, such as the transition from forward flight to a flower search mode once a targeted distance has been achieved. This would govern, for example, the behavioral transition resulting from the honeybees waggle dance when a scout bee recruits a worker to forage at a particular location (Frisch, 1946; Visscher & Seeley, 1982). Integrator neurons can be layered in series and connected via synapses with each neuron performing a higher level of integration than the previous to allow for estimation of varied distances. The fundamental hypothesis is that neuromodulation of the synaptic strength (Destexhe & Marder, 2004; Harris-Warrick, 2011; Katz, Getting, & Frost, 1994) between integrating odometer neurons can alter the gain of the circuit to allow for a tunable neuron network to measure distance visually. The basic assumption is that the amplitude of the honeybee's waggle dance would provide the variable neuromodulatory input (Michelsen, 2003).
ii. LabVIEW™ simulations

An Aphid optical flow sensor (Centeye, Inc.) with an Atmel microprocessor running a simple optical flow odometer neuron circuit (Figure 4.14a) was pointed at a computer monitor displaying moving striped bars to perform preliminary ground-based tests. Voltage traces from a LabVIEW™ simulation of the odometer circuit are shown in Figure 4.14b. A unidirectional optical sensory neuron fires in proportion to the rate of exteroceptive optical flow. Excitatory synapses connect sensory neurons to the 1st-layer DTM leaky integrator neuron that accumulates total optical flow.

![Neuron-based optical flow-mediated odometry](image)

**Figure 4.14. Neuron-based optical flow-mediated odometry:** a, Neural circuit diagram of the optical flow-mediated odometer. Circles represent mathematically simulated neurons and lines with filled triangle ends represent excitatory synapses. Neurons use $\alpha=4.05$ and $\sigma=3.95$ while three parameters modify the behavior of the synapses: reversal potential ($x_{rp}$), relaxation rate ($\gamma$) and strength ($gsyn$). Integrating synapses had relaxation rates of 1, very weak synaptic strengths, and synaptic current values that reset after a spike event. b, Traces of voltage versus iteration for neurons of the network in a simulated using a
DTM model (Rulkov, 2002) running on an Aphid optical sensor with an onboard Atmel microprocessor. With continuous optical flow input detected by the optical flow sensory neuron (top trace), the first layer odometer fires regularly (middle trace) with spikes that are integrated by the second layer odometer (bottom trace).

As optical flow is perceived, indicative of forward movement, sensory neuron spikes lead to a depolarization in the 1st-layer odometer neuron. Once the voltage increases past the firing threshold, the 1st-layer odometer neuron spikes and immediately resets. The spike frequency of the 1st-layer odometer neuron indicates the rate of optical flow while the cumulative number of spikes indicates the total amount of optical an indication of relative distance traveled. This spike train of odometry information is passed to another DTM leaky integrator, the 2nd-layer odometer neuron, via an excitatory synapse. The synapse strength of the connection between the 1st- and 2nd-layer odometer neurons can be varied to change how quickly the 2nd-layer odometer will fire.

In biological systems, neuronal computation, like visually-mediated odometry, is presumably accomplished by large populations of neurons (Kristan and Shaw, 1997). The performance of odometer neuron circuits with different numbers of sensory and 1st-layer odometer neurons was analyzed (Figure 4.15) to determine if network size affected odometer accuracy. Tests of nine configurations of a population coding odometer network show that even with fewer neurons, optical flow odometry is still sufficiently precise. Network configurations containing 1, 3, or 5 sensory neurons and 1, 10, or 20 1st-layer odometer neurons were tested and showed limited variability of outputs with a relative standard deviation of the time to 2nd-layer odometer spike of less than 3.1% (n=99 trials each). For these results a Stonyman2 sensor from Centeye Inc. was attached to a Teensy 3.1 board with ARM microcontroller and pointed at a monitor, mounted 15
cm away. We used optical flow stimuli consisting of 0.25 inch wide moving striped bars on a computer screen (Acer x223w) produced using the PsychoPy2 application (Peirce, 2008). A continuous unidirectional optical flow stimulus was presented that moved at two separate alternating speeds (0.02 phase in PsychoPy for 100 iterations and 0.08 phase in PsychoPy2 for 100 iterations). Since performance was not substantially diminished with fewer neurons, the simplified odometer circuit presented in Figure 4.14a containing only three neurons was used for all experiments to follow.

![Schematic diagram of the visual odometer](image)

Figure 4.15. **Population coding of the visual odometer.** Left panel. Network configuration of odometer variants tested with 1, 3 or 5 sensory neurons and 1, 10 or 100 first-layer odometer neurons. Right panel. The relative standard deviation expressed as a percentage of the time elapsed until the second-layer odometer spike (standard deviation/mean * 100) is shown for the different odometer network configurations tested.

When the synaptic strength between the odometer neurons was modified in a LabVIEW™ simulation, the network output changed (Figure 4.16). A stronger synaptic strength resulted in less time elapsing until the 2nd-level odometer neuron spike.
iii. Robotic implementation

In order to implement the bee-inspired visual odometer on a robot, a flying embodied biorobotic nervous system simulation was developed. The platform was comprised of a free flying dual rotor helicopter (Figure 4.17a) with two main components: an Aphid optical flow sensor (Figure 4.17b) and a custom brain board that runs a nervous system simulation (Figure 4.17c). The brain board contains two ATMEGA 2560 microprocessors, one that runs sensor processing and motor control.
while the other operates a nervous system simulation comprised of layered exteroceptive reflex networks (Westphal, Blustein, & Ayers, 2013).

The optical flow sensor is bidirectionally sensitive to optical flow, outfitted with a 4mm lens to provide a field of view of approximately 36 degrees, and holds a 112 x 112 pixel Stonyman Vision Chip and an onboard ATMEGA 32 chip for processing. It computes optical flow of a 2D image by taking a 10x10 image of 8x8 superpixels using a Lucas-Kanade technique (Lucas & Kanade, 1981) or Srinivasan’s image interpolation algorithm (Nagle, Srinivasan, & Wilson, 1997) with square or plus framing. Before flight testing, performance was validated by pointing the sensor at a monitor displaying an up/down visual stimulus routine of moving black and white striped bars and observing the raw data output (Figure 4.17d). A regular lag in accumulated optical flow approximately every 20 iterations was likely due to the contrast shift at the trailing edges of the stimulus bars (G. Barrows, personal communication). The Lucas-Kanade and Srinivasan algorithms using both frame divisions were qualitatively equivalent in their detection of optical flow when subjected to the up/down visual stimulus (Figure 4.17e). To increase the speed of optical flow sensor processing during flight tests, a simplified Image Interpolation Algorithm was used on a 1D image array of summed pixels.

The helicopter also holds an accelerometer, an ultrasonic sensor for altitude control, a gyroscope to measure angular velocity and drive a virtual compass for heading control, a wireless Bluetooth module for real-time telemetry of sensor and nervous system simulation data, and an RF receiver allowing for optional supervisory control.
Our research shows that traversal distance can be determined from the output of the neuronal odometer. We tested the neuron network in 1) a LabVIEW™ software simulation of the network with simulated optical flow (traces in Figure 4.16, performance analysis in Figure 4.18a), 2) a stationary microprocessor simulation of the network using optical flow displayed on a monitor (Figure 4.18b), and 3) a network running on the helicopter platform during free flight (Figure 4.18c). In all cases, when tested repeatedly,
visual odometer performance showed a correlation between synaptic strength and time to 2\textsuperscript{nd}-layer odometer spike (Figure 4.18a-c). By changing the strength of the synapse, we can change how far the robot flies. For example, as synapse strength is increased, the 2nd-layer odometer in the network spikes sooner indicating a lower amount of accumulated optical flow, a signal of a shorter distance flown. Thus, the desired flight distance of a robot on a mission can be represented by the synaptic strength between the 1\textsuperscript{st}- and 2\textsuperscript{nd}-layer odometer neurons. Just as honeybees convey this information through their waggle dances, the neural odometer presented here can be tuned with a single synaptic strength value. Altitude was held constant around 50 cm using the ultrasonic sensor and a simple PID controller.

Figure 4.18. *Odometer synapse strength is correlated with flight distance.* Second-layer odometer spike timing for the odometer neuron network running in a LabVIEW\textsuperscript{TM} simulation on a computer (a), on a microprocessor with simulated flight conditions (b), and on the biorobotic platform in free flight (c). Different letters denote significant differences (p<0.01, t-test). The difference between distances flown using 0.0075 and 0.01 synaptic strengths was not significantly different (p=0.138, t-test).

Real-time wireless transmission of neural activity onboard the helicopter via Bluetooth showed that varied synaptic strengths produced distinct network outputs (Figure 4.19a-c). In one trial, a strong synapse (strength of 0.02) between the 1\textsuperscript{st}- and 2\textsuperscript{nd}-layer odometer neurons resulted in a 2\textsuperscript{nd} layer odometer spike after a 14.5-foot flight.
Figure 4.19. Neural activity of odometer network changes with varied synapse strength. Spike outputs for neurons of the two-layer odometer circuit (see Figure 4.14a) during free forward helicopter flights under varied synaptic strengths. a. A strong synapse (0.02) resulted in a flight of 14.5 feet. b. A medium synapse (0.01) resulted in a flight of 22.9 feet. c. A weak synapse (0.005) resulted in a flight of 40.9 feet.

Optical flow exteroception is just one of several sensory modalities implemented on RoboLobster. The next chapter describes other exteroceptive modalities and complete sensorimotor integration onboard RoboLobster.
A. Exteroception

A key feature of our comparative approach is to construct controllers from exteroceptive reflex networks based on neurophysiological principles established in the literature using simulated neurons and synapses. The process begins with reviewing the knowledge of the biological system and developing a neuron network hypothesis. Then LabVIEW™ simulation and robotic implementation are endeavored to support and refine the model. The previous chapter outlined this process for optical flow sensory inputs. In this chapter we will repeat the exercise for three more sensory systems: antennal bending, heading control, and claw mechanoreception. Once each of the hypothetical network configurations for single modes of exteroception have been developed we can begin to layer them into a single cohesive network to describe sensory fusion. Once the sensory network has been developed, integration with the locomotory network (Chapter 3) can be completed. Complete sensorimotor integration is the focus of robotic implementation on RoboLobster (Chapter 7).

Due to the whole body coordination involved in locomotion, there are a variety of sense organs that affect behavior to allow for environmental interaction. Many sensory systems involved in orientation are based on directionally-selective sensors that orient to such stimuli as odor sources, light or water surge (Murray, Estepp, & Cain, 2006). These sensors are usually based on simple circuits such as those described by Loeb (1918) and implemented by Braitenberg (1965) in hypothetical organisms. During locomotion organisms present several modes of operation. A *taxis* is an orientating movement of an animal with respect to a stimulation source. A *kinesis* occurs when a stimulus causes an increase or decrease in movement but has no impact on orientation (Braitenberg, 1965). *Navigation* requires higher level
integration and allows for direction detection and the construction of a map sense (Murray et al., 2006). Since total behavioral output represents the cumulative effect of all sensory input coordinated through the CPG, the mechanism of translating information about the environment into labeled line inputs will be discussed on a system-by-system basis.

i. Antennal bend

Lobsters can orient into flow when blinded, a reflex termed *antennal-mediated rheotaxis* (Wilkens, Schmitz et al. 1996). Surging water past a lobster results in antennal bending which elicits stereotyped behavioral outputs to counteract the imposed disturbance (Ayers and Witting 2007). Lateral surge across the animal results in a compensatory turn into the water flow (Figure 5.1a). Longitudinally off-center flow elicits gradual yawing into the flow during locomotion (Figure 5.1b). In addition to their role in maintaining stability during locomotion, these behavioral reflexes play an ecological role in partially explaining the contranatant movements of lobsters towards spawning areas (Groeneveld and Branch 2002).

A neural network explaining these behaviors based on antennal exteroceptive inputs has already been described (Ayers, 2004). We assume a range fractionation scheme to trigger specific neurons for different degrees of antennal bend (Figure 5.2). Biological evidence of range fractionation has been particularly well described in crustacean proprioceptive afferents (Cohen, 1963; Mill & Lowe, 1972). In our model, response to hydrodynamic flow is mediated by antennal sensor neurons that respond to high (H), medium (M), and low (L) bending medially and laterally. Inhibitory connections between these sensory neurons drive individual neuron activation (Figure 5.2).
Figure 5.1. *Lobster antennal reflexes.* Range-fractionated antennal sensory neurons respond to high (H), medium (M) and low (L) bending medially and laterally and project to hydrodynamic flow sensory interneurons (grey circles). **a.** Neural network hypothesis to explain the lobster’s rheotaxic response to lateral surge. **b.** Neural network hypothesis to explain the lobster’s yaw response to off-center axial surge. (adapted from Ayers, 2004)

Figure 5.2. *Range fractionation.* Different magnitudes of antennal bend activate different sensory neurons. Inhibitory synapses between sensory neurons mediate a range-fractionated output. (from Ayers & Witting, 2007)
Projections to the Yaw interneuron mediate rheotaxic yaw (Figure 5.1a) while projections to Rotate interneurons mediate responses to off-center axial surge (Figure 5.1b) (Ayers, 2004). Under lateral surge conditions with the lobster’s antennae held forward, a high medial bend on one antenna and a high lateral bend on the other antenna excites the Rotate interneuron of the medially bent side to elicit rheotaxis (Figure 5.1a). Off-center axial surge produces less severe antennal bending that, when antennae are deployed laterally, excites the Yaw interneurons causing the vehicle to align with the surge by producing bias between the bilateral Forward commands (Figure 5.1b).

Previous LabVIEW™ software simulations of the complete antennal bend neuron network (Figure 5.3a) have shown command outputs that match behavioral observations (Figure 5.3bc) (Ayers & Witting, 2007; Ayers et al., 2010). Single sensory neurons are activated for each range of antennal bend magnitude and the spike activity drives motor commands via interneuron activity.
Figure 5.3. *LabVIEW™ simulation of antennal bend neuron network.* **a.** Complete antennal bend neuronal circuit (combining Figures 5.1a and 5.1b). **b.** Rheotaxic response to lateral surge with antennae deployed forward. The two panels represent activity of the neurons in **a** when the surge (top two panels) oscillates from left to right to left. **c.** Yaw response to off-center axial surge (from the right front quadrant) with antennae deployed laterally. Asymmetric command activation causes a turn into the flow during forward walking. (from Ayers & Witting, 2007)
ii. **Heading control**

Spiny lobsters and many other organisms can orient relative to Earth’s magnetic field but little is understood about the neural mechanism involved (Boles & Lohmann, 2003). Heading control with a neuronal compass network hypothesis has been accomplished on RoboLamprey (Westphal, 2012; Westphal, Rulkov, Ayers, Brady, & Hunt, 2011) and RoboBee (Westphal, Blustein, & Ayers, 2013). Compass error (Figure 5.4a) drives a range-fractionated network (Figure 5.4b) to discretize response to high and low compass error (Figure 5.4c) (Westphal, 2012). The compass error is converted proportionally into injected current into both compass error neurons. As compass error increases, so does the injected current into the simulated neurons. The $\alpha$ and $\sigma$ parameters of the High compass error neuron are set to increase its threshold so that more injected current is required before the cell begins to spike. See Chapter 2 for an overview of the neuron model.

![Diagram](image)

**Figure 5.4. Range fractionation of the compass network.** a. Response to compass error with a desired heading of 0° (to the right). b. The High compass error neuron inhibits the Low compass error neuron. c. As heading error varies (top trace), the sensory neurons are each active for different ranges of compass error. (from Westphal et al., 2013)

---

1 High compass error neuron: $\alpha = 3.4$, $\sigma = -2.98$; Low compass error neuron: $\alpha = 3.4$, $\sigma = -2.78$
To mediate heading hold, a network hypothesis was devised based on what was implemented on RoboLamprey (Figure 5.5) (Westphal, 2012; Westphal et al., 2011). The Low compass error neuron drives the Rotate interneuron and a corrective turn towards the desired heading. The High compass error neuron excites the Yaw interneuron to mediate a turn-in-place towards the target heading. Turning in place involves walking forward on one side of the body and backward on the other through activation of the commands for opposite direction on the two sides (Copp & Jamon, 2001; Ayers & Crisman, 1992).

iii. Claw bump

Decapod crustaceans detect obstacles with their claws and antennae (McMahon & Patullo, 2005). Lobsters use exteroceptive cuticular hair and chordotonal organs to detect mechanical stimuli like flow, pressure and contact (Laverack, 1963; Solon & Cobb, 1980). We
implement an obstacle avoidance response mediated by mechanical contact with the claw. In the animal model this would be mediated by rapid passive bending of several joints that would activate many joint velocity sensitive afferents in the chordontonal organs and by activation of cuticular hair organs (Shelton & Laverack, 1968; Solon & Kass-Simon, 1981; Kinnamon, 1979). On RoboLobster, a Bump sensory neuron receives input from a magnetic reed switch on the claw. The Bump neuron excites both Backward walking commands to mediate a back-up behavior (Figure 5.6).

Figure 5.6. Claw bump drives a back-up response. When the Bump sensory neuron is activated, it excites the Backward command neurons on both sides of RoboLobster.

To test claw exteroception robotically we used a LEGO Mindstorms NXT 2.0 robot equipped with bilateral bump sensors (Figure 5.7) (for the complete implementation see: Blustein, Rosenthal, & Ayers, 2013). LabVIEW™ Virtual Instruments for neurons and synapses described in Chapter 2 were modified to operate on the NXT brick (LEGO processor). This programming framework was used to form the basis of a middle and high school classroom curriculum called Neurobots (Blustein, Schultheis, & Ayers, 2011). The open-source curriculum
is available online\textsuperscript{2}; see Appendix A for more details. Since two claw sensors were used, the claw bump neuron network from Figure 5.6 was modified to accommodate bilateral inputs (Figure 5.8). Bilateral claw bump sensory inputs project to the contralateral Backwards command.

\textbf{Figure 5.7. \textit{LEGO Mindstorms NXT robot}.} Sensor inputs include two bump sensors and two antenna sensors (see Appendix A for details). The antenna sensors were disabled for the claw experiments. One motor drives each tread.

\textsuperscript{2} https://myfiles.neu.edu/blustein.d/neurobots/
Figure 5.8. *Bump exteroceptive network for LEGO Mindstorms NXT robot*. With two bump sensors, Bump sensor neurons project decussating excitatory connections to Backward commands.

The behavior of a lobster (Figure 5.9a) and the LEGO robot (Figure 5.9b) in the test arena were recorded with an overhead video camera. The test arena was unmodified between animal and robot tests except that the water was emptied from the tank for the robot trials. Animal and robot travel paths were automatically tracked using MouseLabTracker in MATLAB (Tort et al., 2006). Representative paths for lobster (black lines) and LEGO robot trials (colored lines) can be observed from the video tracking results (Figure 5.10). In the robot trials, synaptic strength from the claw bump sensory neurons into the nervous system was varied and represented by the different colored paths (red, low excitation; blue, medium excitation; green, high excitation). The nervous system simulation with mid-level excitatory connections from the claw bump sensory neurons produced similar behaviors to the animal. Quantitative parameters can be extracted from the data such as path length and average walking speed (Table II).
Figure 5.9. Overhead video tracking of lobster and robot. Video frames of the test arena for LEGO robot (a) and lobster (b) trials. Conditions were kept identical except that water was added to the tank for the animal trials. The high contrast video image facilitates automatic tracking using MATLAB software. (from Blustein et al., 2013)
Figure 5.10. *Tracked paths of lobster and LEGO robot.* Lobster (black lines) and LEGO robot (colored lines) paths through the test arena as shown in Figure 5.9. On the robot, the synaptic strength from claw bump sensor neurons was varied (red, low excitation; blue, medium excitation; green, high excitation). The * symbol denotes the starting position and black shapes show the location of obstacles. Since head position was being tracked, apparent obstacle contact varies due to variability in claw position. (from Blustein et al., 2013)

Table II. *Averaged parameters from analysis of video behavior data.* Lobster and LEGO robot performance with varied synaptic strengths from the inputting claw bump sensory system. (from Blustein et al., 2013)

<table>
<thead>
<tr>
<th></th>
<th>Average path distance (cm)</th>
<th>Average travel speed (cm)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Lobster</td>
<td>177.37</td>
<td>1.03</td>
</tr>
<tr>
<td>Lego Robot (high excitation)</td>
<td>162.72</td>
<td>0.94</td>
</tr>
<tr>
<td>Lego Robot (mid excitation)</td>
<td>165.32</td>
<td>0.85</td>
</tr>
<tr>
<td>Lego Robot (low excitation)</td>
<td>51.02</td>
<td>0.69</td>
</tr>
</tbody>
</table>
iv. Other sensory modalities

Given the inherent need to focus research efforts, not all of the lobster’s sensory systems have been incorporated onto RoboLobster. We have omitted two notable sensory systems: inputs from statocysts and chemoreceptors. There has been some impressive biomimetic work on statocyst-inspired tilt sensing (Ascari, Cavallaro, Stefanini, & Dario, 2006; Deliagina, Arshavsky, & Orlovsky, 1998; Liu & Lei, 2014) and also on lobster chemoreception (Grasso & Atema, 2002). Drawing from this previous work, implementation of these systems on RoboLobster should be considered in the future. Here we will describe the biological relevance of these systems to inform future work.

Sensory input from the statocysts regarding gravitational orientation leads to two types of behavioral response: compensation and righting (Davis, 1968). The statocyst is an ectodermal fluid-filled sac lined by hairs that contact the statolith located in the basal segment of each antennule. Their primary role is in the detection of orientation relative to gravity and angular acceleration (Cohen, 1955). Different statocyst afferents encode information about absolute orientation, displacement direction, velocity, and acceleration and modulate walking motor outputs (Cohen, 1955; Fraser, 2001).

Davis (1968) derived functional wiring diagrams of the neural control of uropods and swimmerets by statocysts in lobsters. This work focused on righting response, however, rather than the compensatory responses required to maintain locomotion. However, Davis argues that neural models similar to the ones he proposed are applicable to all crustacean appendages and can presumably apply to compensatory responses. This type of work can inform the development of additional exteroceptive neuron networks to implement on RoboLobster.

Another exteroceptive sensory input into the lobster locomotory system is that of
chemoreception. Antennular chemoreceptors in lobsters are analogous to olfaction in terrestrial animals while receptors on the legs and mouthparts mediate the equivalent of taste (Derby & Atema, 1982).

Decapods employ a mechanism of tropotaxis, the use of spatially separated sensors, for chemo-orientation. Multiple sensory modalities inform chemo-orientation during locomotion (Grasso, 2001). Odor-gated rheotaxis represents a mechanism of sensory integration in which a chemical stimulus gates on a rheotaxic behavior (Grasso, 2001). Neural information from decapod chemoreceptors is coded as a labeled line as they detect various chemicals found in prey organisms. Orientation in relation to chemical sources is achieved by integration mechanisms using signals from chemoreceptors at different body positions (tropotaxis) and the relative intensities detected. Chemoreception is not implemented on RoboLobster but remains a promising area for future biorobotic analysis.

Populations of neurons from chemoreceptors and other sense organs fire in specific patterns allowing for distinction between subtly different stimuli (Steullet et al., 2002). Population coding describes how large populations of neurons operate in concert to generate a behavior (Kristan & Shaw, 1997). Since neurons respond to a broad array of stimuli, population coding allows for the coordination of all inputs to elicit the appropriate behavioral response (Kristan & Shaw, 1997). Steullet et al. (2002) identified two partially overlapping chemosensory pathways that both utilize population coding in the antennae of the spiny lobster, Panulirus argus. The presence of partially redundant pathways involving populations of neurons reduces generalization of odor signal detection, provides backup in case of sensor organ damage, and increases the sensitivity or accuracy of sensing (Steullet et al., 2002). Population coding of sensory and command neurons is not implemented on the current RoboLobster but is another
direction to explore for future biorobotic work.

B. Sensory integration

The next step in building RoboLobster’s ENS is to layer individual exteroceptive reflex networks to develop a network hypothesis that explains multimodal sensory processing. There are several mechanisms that have been implicated in how two modes of sensory input can interact in the nervous system.

In Drosophila, a stimulus in one sensory modality has been shown to modulate the gain of other exteroceptive inputs (Tammero & Dickinson, 2002). Sherman and Dickinson (2003) showed that the visual system of flies was more sensitive to optical flow when rotated at low angular velocities but with increased flow rates the input from this system decreased and mechanosensory input from the halteres predominated. Mechanosensory wind inputs may serve to sustain forward flight in flies by counteracting the repulsive effects of visual expansion due to self-motion (Budick, Reiser, & Dickinson, 2007). Other mechanisms of sensory integration have been suggested, however, including such sensory specific mechanisms as odor-gated rheotaxis in lobsters (Grasso & Atema, 2002) as well as corollary discharge, summation or chained subsystem processing (Grasso, 2001; Holst & Mittelstaedt, 1950; Payne, Hedwig, & Webb, 2010).

i. Antennal and optical flow fusion

The interaction between antennal and optical inputs into the motor system of the lobster is not well understood. Clearly lobsters can navigate with impunity in the dark, so they must use other senses. Underwater surge profiles that drive the antennal sensory system of the lobster also
generate optical flow stimuli. By integrating the antennal bend and optical neural networks onto a closed loop biorobotic platform, the interaction between these exteroceptive senses can be explored.

Preliminary LabVIEW™ simulations of lobster optical and antennal inputs suggest that the reflexes for both sensory modalities are complementary in a wide range of environmental situations (Blustein & Ayers, 2010b). Under natural surge conditions, optical and hydrodynamic flow stimuli are generally coupled, i.e., for a given water surge orientation, the resulting optical flow stimulus due to particulates will be consistent. Based on the existing behavioral observations, a summating sensory fusion mechanism was first considered and incorporated into the network hypothesis (Figure 5.11).
Figure 5.11. *Neural network hypothesis for integration of antennal and visual inputs*. Sensory projections from both modalities drive common interneurons and command neurons which initiate forward (F) and backward (B) locomotion bilaterally. See Chapter 4 for the optical flow network and Figure 5.3 for the antennal network. (Ayers, Blustein, & Westphal, 2012)

The network in Figure 5.11 was simulated in LabVIEW™ software (see Chapter 2 for method details). The results of the simulation support summation as a possible mechanism for the integration of optical and hydrodynamic flow sensory inputs (Figure 5.12). The motor outputs elicited by antennal and optical inputs were equivalent under the surge profiles simulated. This apparent sensory redundancy may contribute to the maintenance of adaptive behavioral output when one sensory system is compromised, e.g. in the dark or with damaged antennae.
Figure 5.12. LabVIEW™ simulation of optical flow and antennal bend integration. For a given surge profile (top diagram) the response of the neural network hypothesis from Figure 5.11 is displayed. Letter labels denote different phases of the response: A, before stimulus; B, during stimulus; C, after stimulus. Descriptions of behavioral outputs are listed at the bottom. (from Blustein & Ayers, 2010b)
ii. Compass and optical flow integration

To develop and test the fusion of compass and optical flow sensory inputs, we switch to a robotic bee platform. An at-scale RoboBee is still under development (Chirarrattananon, Ma, & Wood, 2014) so we use the dual rotor helicopter proxy (Westphal et al., 2013) used for the odometer studies in Chapter 4.

Several reflex networks that mediate responses to single modes of exteroception are layered to form the fused sensory network of RoboBee (Figure 5.13d). The compass network (Figure 5.13a) uses the left or right compass error from a desired heading to mediate a corrective yaw response. The left compass neuron fires when the actual heading is to the left of the target and excites the Right Command neuron that produces a yaw to the right. The right compass neuron leads to the opposite response when the actual heading is to the right of the target. Low rates of optical flow mediate reflexive responses to visual perturbations through decussating excitatory synapses between the bilateral optical flow neurons and the Right/Left Commands (Figure 5.13b). Bilateral optical flow inputs also integrate into a 1st level odometer neuron to neuronally tabulate distance flown. Multiple integrating odometer neurons can be connected in series to obtain responses on longer time scales allowing for longer specified flight distances (network in Figure 5.13b, see Chapter 4 for robotic implementation). Optical flow neurons sensitive to high rates of optical flow indicative of an impending obstacle inhibit the low optical flow neurons and elicit an avoidance response through excitatory connections to the command neurons (Figure 5.13c).

Motor neuron spikes drive a pulse width modulated duty cycle output to the two rotors of the helicopter. The Left motor neuron activates the clockwise rotor to bias the yaw motion to the
left. The Right motor neuron activates the counterclockwise rotor to generate a yaw bias to the right (Westphal et al., 2013).

Figure 5.13. Exteroceptive reflex networks for RoboBee. Unimodal reflex networks for compass-mediated heading control (a), optical flow-mediated odometry and compensation (b), and optical flow-mediated obstacle avoidance (c). The complete exteroceptive network with fused inputs from multiple sensory modalities is depicted in (d). Black circles denote sensory neurons, grey circles are processing interneurons and commands, white circles are motor neurons. For RoboBee, the R motor neuron elicits a turn to the right and the L motor neuron elicits a turn to the Left. Lines with triangle ends represent excitatory synapses; filled circle ends, inhibitory synapses. (from Ayers et al., 2012)

The field behavior of RoboBee is based on motivated behavioral sequences organized reactively relative to a supervisory motivation. A behavioral sequence results from a reflex chain in which the sensory releaser after each motor action stimulates the next behavioral act (Berridge, Fentress, & Parr, 1987; M. H. Dickinson & Lent, 1984). In RoboBee we adopt a chain-reflex sequence to mediate an out-and-back pollination mission. The recruitment of foragers by scout bees to nectar sources is mediated by the waggle dance (Frisch, 1946; Visscher & Seeley, 1982). The distance and direction of a target is conveyed based on the magnitude and
orientation of the dancing bee (Frisch, 1946; Seeley, Mikheyev, & Pagano, 2000). This communication provides the bee with a search vector, a distance and direction, that could be recalibrated via neuromodulation (Destexhe & Marder, 2004; Harris-Warrick, 2011; Katz, Getting, & Frost, 1994). The basic hypothesis is that the honeybee's waggle dance would provide variable neuromodulatory input into the odometer and compass networks (Michelsen, 2003).

An overview of the chain-reflex scheme adopted for RoboBee is presented in Figure 5.14. Once a bee has been communicated a target heading, the bee will takeoff and the compass network will mediate corrective turning to the desired heading. When the correct heading is reached, inhibition on a forward flight command and on the optical flow neurons is released. The flight command results in a pitch forward and translational movement, during which optical flow sensory information accumulates in the integrating odometer circuit. The odometer neuron will fire once the target distance has been flown, initiating a flower search mode to achieve pollination. We skip the pollination step in our experiments as others are working on this problem (Koppal et al., 2013). A return to the hive is initiated with a pitch up and a yaw to the reverse heading mediated by the compass neuron network. Once the correct heading is obtained, RoboBee’s flight command will activate and it will pitch forward for the return flight towards the hive.
Figure 5.14. *RoboBee’s chain-reflex sequencing*. A sensory releaser (green box) leads to a motor response (blue oval) mediated by the releasing mechanism (yellow box). See text for a descriptive narrative.

With the network hypothesis (Figure 5.13) and the chain-reflex sequence structure (Figure 5.14), we can move to software simulations. Figure 5.15 shows a LabVIEW™ simulation of the networks underlying compass-mediated heading control and optical flow-mediated stabilization and compensation in a helicopter approximation of RoboBee. The behavioral sequence shown consists of three phases: I. Initiation and Clockwise yaw for target heading seek; II. Target heading overshoot and Counter-clockwise yaw; and III. Forward linear flight for a distance specified by the synaptic strength of the odometer circuit. The resulting sensory milieu at the end of each phase elicits the next behavioral act. This chain-reflex sequencing to produce complex behaviors is common across a range of species (Berridge et al., 1987; M. H. Dickinson & Lent, 1984).
Figure 5.15. *LabVIEW*™ simulation of compass and optical flow neuron network. Voltage versus time charts of the neurons of the simulated neural network described in Figure 5.13 for a behavioral sequence of a helicopter taking off, obtaining a target heading (I and II) and using optical flow-mediated odometry to fly a specified distance (III). In I, a yaw response is driven to obtain a specified heading. After a heading overshoot in II, the odometer neurons then integrate optical flow during forward flight in III to indicate when a desired distance has been flown. Compass neurons inhibit the pitch forward command and optical flow inputs so translation only occurs in the desired heading. (from Ayers et al., 2012)

C. Complete RoboLobster ENS

The reflex networks for each sensory modality described are layered to form RoboLobster’s complete exteroceptive network (Figure 5.16). Networks underlying response to optical flow (Figure 5.11), antennal bend (Figure 5.11), heading control (Figure 5.5), and bump-mediated obstacle avoidance (Figure 5.6) each integrate on common bilateral interneurons and walking command neurons that modulate the motor system. These layered unimodal networks
form the most parsimonious network hypothesis to describe the observed behavior accounting for established knowledge of the biological system. Biomimetic neural components are adopted including command neurons (Bowerman & Larimer, 1974b), range-fractionated sensory neurons (Cohen, 1963) and decussating connections (Braitenberg, 1965; Loeb, 1918).

Figure 5.16. *Complete biomimetic exteroceptive reflex network for RoboLobster.* Neural network hypotheses to explain behavioral responses to optical flow, antennal bend, heading control, and obstacle avoidance are layered to form RoboLobster’s complete exteroceptive reflex network. (from Ayers et al., 2012)

There are several mechanisms of sensory fusion that are incorporated into the complete layered exteroceptive network. Summation of two sensory modalities, as observed with visual and tactile inputs driving the crayfish tail flip response (Liu & Herberholz, 2010), occurs between the optical flow sensory inputs and the low threshold antennal bend afferents. Gain modulation occurs when one sensory system modulates the gain of another as is observed in the
interaction of optical flow and haltere-mechanoreception in the fruit fly (Sherman & Dickinson, 2003). This interaction occurs between high threshold antennal bend afferents and lateral optical flow inputs in the network model (Figure 5.17).

Figure 5.17. Gain modulation between high antennal bend and optical flow inputs. Underlying synaptic mechanism highlighted in blue.

With strong gain modulation, modality gating can occur where one sensory system turns on and off the inputs of another, as in the lobster’s odor-gated rheotaxis (Grasso, 2001). In the model, modality gating occurs as the high threshold heading deviation neurons completely inhibit lateral optical flow inputs (Figure 5.18). A corollary discharge, or efferent copy, is another mechanism of sensorimotor integration. This occurs when a signal from the motor system inhibits sensory afferents in order to desensitize the system to self-generated stimuli. One example is seen in the inhibitory connection from the cricket’s chirping motor pathway to its auditory system (Poulet & Hedwig, 2006). Corollary discharge in RoboLobster’s exteroceptive network is implemented as inhibitory connections between the Forward command neurons and the lateral optical flow neurons via the Rotation interneurons (Figure 5.19).
The layered exteroceptive reflex architecture presented can form the basis of supervised reactive autonomy where the underlying motivation of the vehicle is to pursue a search vector. Search vectors can be specified by two tokens, a direction and a distance, that conveyed over serial communication. In the future, communication could be achieved by sonar, LEDs, or other
wireless methods. The vehicle can also be provided with a propensity to negotiate or investigate obstacles that it encounters by modifying the synaptic strength of the bump sensory inputs. We expect that ongoing efforts in synthetic biology will provide sensing capabilities and taxic behavior (Grasso & Basil, 2002) for a broad variety of chemical sensors specific to agents of interest or harm (Antunes et al., 2011; Purnick & Weiss, 2009). Chemical sensing is key to the inter-individual exteroceptive reflexes that underlie stigmergy (Hölldobler & Wilson, 2009).

The command neurons of the exteroceptive network (Figure 5.16) modulate the CPGs of the 8 walking legs (see Chapter 3). A complete network hypothesis incorporating exteroception and sensorimotor integration can be devised by combining the component networks (Figure 5.20). Limb coordination is accomplished with ipsilateral and contralateral reciprocal inhibition between adjacent elevator neurons.

Armed with a complete ENS model, we are ready for implementation on a robot. The real power of the biorobotic approach requires real-world testing on a closed-loop system which will be presented in Chapter 7. Paradoxical sensory inputs can be applied and the nuances of the neuronal processing networks can be explored. Comparisons to animal performance in similar conditions can be used to inform the refinement of the neural model. First, however, a biomimetic robot platform must be designed, engineered and constructed. The hardware and software of RoboLobster will be described in the next chapter.
Figure 5.20. RoboLobster’s complete ENS. The layered exteroceptive networks (Figure 5.16) drive command neuron modulation of the CPG circuitry of each walking leg (see Chapter 3).
Chapter 6. RoboLobster hardware & software

In order to fully test the neural networks presented in Chapters 3-5, we have developed an advanced biomimetic robot platform: RoboLobster. The work presented in this chapter is an extension of previous versions of RoboLobster as outlined in Chapter 1. Here we present the hardware and software that comprise the next-generation robot. The aim is to develop a robot platform that mimics the control principles of the biological model including biomechanics, sensors, effectors, and nervous system control. Each component is covered in this chapter and more details can be found in the Appendix. The results of the nervous system simulation running on RoboLobster are presented in Chapter 7.

A. Mechanical design

RoboLobster is an eight-legged underwater walking robot designed to mimic the body plan of the American Lobster, Homarus americanus (Figure 6.1). The eight walking legs are attached to a cylindrical hull that houses the batteries, driver boards to pulse current to the actuators, power converters, and microprocessors to run the neural network simulation. RoboLobster’s body length is roughly equivalent to the largest lobsters on record (Wolff, 1978) (Table III).
Figure 6.1. RoboLobster test platform. 8-legged biomimetic robot lobster used for testing neural network hypotheses.

Table III. Comparison between RoboLobster and *Homarus americanus* specimens.

<table>
<thead>
<tr>
<th>Specimen</th>
<th>Carapace length (cm)</th>
<th>Body length, rostrum-telson (cm)</th>
<th>Claw S.A. (cm²)</th>
<th>Claw S.A, distal segment only (cm²)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Wolf 1978, plate 1c</td>
<td>29.19</td>
<td>63.40</td>
<td>1315</td>
<td>918</td>
</tr>
<tr>
<td>Wolf 1978, plate 2a</td>
<td>28.81</td>
<td>60.30</td>
<td>1341</td>
<td>1017</td>
</tr>
<tr>
<td>RoboLobster</td>
<td>27.30</td>
<td>62.07</td>
<td>584</td>
<td>584</td>
</tr>
</tbody>
</table>

Two hull materials have been used. Initially an aluminum hull was chosen to maximize heat transfer to the ambient water environment. A clear acrylic hull was subsequently used to allow for visual observation of the internal electronics to monitor electrical problems and to identify internal flooding events. The hull has an outer diameter of 11.4cm and is 27.3cm long, but these dimensions can be modified to fit different packages of electronics and sensors. A return to an aluminum hull is recommended for longer and more intensive missions and experiments to better cool the electrical components.

RoboLobster’s leg parts were designed in Solidworks and Autocad software and the part drawings can be found in the Appendix B. Using a table-top lathe and CNC-mill, the parts were fabricated out of industrial plastic (PEEK™, Delrin™ and Ultem™) (Figure 6.2, top panel). We produced a version of RoboLobster that was printed with PLA plastic using a MakerBot
Replicator 2 (Figure 6.2, bottom panel). The fully-assembled 3D printed leg can be seen in Figure 6.3.

Each of the robot’s eight legs is moved by six shape memory alloy (nitinol) actuators arranged around three joints (Figure 6.4). Elevator and depressor actuators move the leg up and down, respectively, mimicking the lobster’s muscles moving the coxo-basal joint. Protractor and retractor actuators arranged on the underside of the hull operate during forward and backward walking to mimic the movement about the thoraco-coxal joint. Extensor and flexor actuators operate during sideways walking and capture the movement generated by the mero-carpopodite joint (Ayers & Clarac, 1978).

Figure 6.2. *Leg part fabrication techniques*. Parts of RobotLobster’s leg can be fabricated using a CNC mill and lathe or using a 3D printer.
Figure 6.3. *3D printed leg*. Fully assembled leg made of PLA printed on a Makerbot Replicator 2 with inset of lobster leg profile\(^1\).

Figure 6.4. *Robot leg design and actuation*. Schematic of six nitinol actuators arranged in antagonistic pairs. The Protractor and Retractor rotate the leg assembly, Elevator and Depressor move the leg up and down, Extensor and Flexor move the leg out and back. The actuators are made up of a current carrying wire (1), a pulley string (2), plastic end caps (3), the nitinol wire (4), and a threaded hole for attachment (5).

---

Proper tensioning of the nitinol actuators is accomplished by adjusting a pulley string and adjusting the threaded depth of a screw into the actuator end cap (see actuator detail in Figure 6.4). The tip of the robot leg is threaded to allow for a variety of dactyl-like attachments. After observing that the robot’s leg was slipping during walking (Figure 6.5), we added a rubber dactyl attachment (Figure 6.6). This addition added enough friction to the robot’s toe to prevent slipping and more closely match the real lobster’s leg trajectory (Figure 6.7).

Figure 6.5. *Kinematic tracking of leg movement from side-view video recording.* Blue points (top) track the ‘MC’ (mero-carpopodite joint) and green points (bottom) are the dactyl. Units are in pixels and not calibrated to real-world units.

Figure 6.6. *Rubber toe added to minimize slipping.*
Figure 6.7. *Kinematic tracking of leg movement after adding rubber toes*. Blue points (top) track the mero-carpopodite joint and green points (bottom) are the toe as viewed from the side. Tracking done on each frame of 120fps video.
The nitinol actuators are made of a nickel-titanium alloy, a shape memory alloy (SMA) that presents reversible contraction. Nitinol wire\(^2\) is surrounded by etched Teflon\(^{TM}\) tubing\(^3\). Lead wires are soldered to stainless steel crimps at the ends of the nitinol actuator and potted in plastic end caps with epoxy (Figure 6.8, top panel). SMAs like nitinol have two conformationally different stable states. The Austenite state is a compressed state that occurs after heating the nitinol to above a particular temperature (for RoboLobster this is 70 degrees C). When the material cools and is mechanically stretched, the elongated Martensite state is reached. The state changes occurring as a result of heat and stretch are shown in the bottom panel of Figure 6.8. The contraction can result in over 7% shortening of the actuator that can lift a 1kg weight (Tanaka et al., 2010).

To provide the heat necessary to drive the change from the Martensite state to the contracted Austenite state, electrical current is passed through the actuator. The resistance of the wire actuator (\(~1-2 \Omega\)) causes the material to heat when electrical current is provided. Pulses of electrical current can drive the contraction of the nitinol wire. The ambient water provides rapid cooling of the actuators to allow for limb movement on the same time scale as the movement observed in the lobster’s joints. The antagonistic arrangement of pairs of actuators around joints provides the stretching force necessary to complete the transition to the Martensite state after cooling. Motor neuron spikes from the onboard ENS simulation can drive the pulsed current to the actuators to provide biomimetic excitation/contraction coupling. The contraction of actuators is sufficient to move each of the robot’s joints, however ferrous shape memory alloys have been recently discovered that could provide even greater contraction, over 13%. (Tanaka et al., 2010).

\(^2\) Purchased as Flexinol® with oxide removed from Dynalloy, Inc. Diameter = 0.01” +/- 10%. Af=70°C.
\(^3\) Extruded PTFE purchased from Zeus, Inc. Inner diameter = 0.012” +/-0.001”. Wall thickness = 0.004” +/-0.001”.

112
Figure 6.8. SMA actuators. Top panel. Nitinol actuator. Bottom panel. Temperature and deformation chart showing conditions necessary to drive the phase change between the Martensite and Austenite crystalline states. (from: Ayers & Witting, 2007)

SMA’s have been used to actuate a range of biomimetic robots including a swimming lamprey (Westphal, Rulkov, Ayers, Brady, & Hunt, 2011) and micro manta ray (Z. Wang, Wang, Li, & Hang, 2009) and ground-based walking (Liu & Yen, 2006) and crawling robots (B. Kim, Lee, Lee, Kim, & Lee, 2006; J.-S. Koh & Koh, 2010). Some work has also been done on wing
folding (Nguyen, Park, Byun, & Goo, 2010) and actuation in flying biomimetic robots (Bunget & Seelecke, 2010). Drawbacks to using SMA’s to actuate robots include large current draws, slow actuation cycling and limited contraction rates. Actuating a single leg requires up to 6A instantaneous current draw, the battery life of the robot under continuous operation is about 1 hour. The presented configuration is not limited by slow actuation or limited joint rotation due to water cooling and joint geometry, respectively. The muscle-like properties of nitinol allow for a remarkably biomimetic approach to excitation-contraction coupling on RoboLobster.

Antennal sweeping is obtained using two 16.5 cm-long nitinol actuators that turn two coupled cylinders (Figure 6.9). One actuator retracts both antennae, and the other actuator protracts them simultaneously. The entire assembly screws into the front end cap and runs along the length of the dorsal side of hull (see Figure 6.1 for reference).

![Figure 6.9. Bottom view of antenna actuation assembly.](image)

The claw and tail were designed to pivot up and down to serve as stabilizing control surfaces in environments with strong hydrodynamic flow. The motor implementation has yet to be completed. A monoclaw assembly was adopted to simplify control and maximize contiguous surface area to accommodate future plans for biohybrid sensors. Analysis of claw surface area shows that RoboLobster’s claw surface is smaller than that of similarly sized lobsters (Table I).
Since the animal can move its claws laterally and the robot cannot, the robot’s claw size was not increased in order to limit this fixed width of the robot and allow for testing in laboratory tanks.

The robot is slightly negatively buoyant to match the physical characteristics of the animal model and is the size of a large lobster (see Table I). RoboLobster was initially designed to accommodate 3 battery packs but only two have been used during operation for the following reasons: 1) Limit the risk and potential damage of on-board overheating and fire; 2) Allow for increased space in the hull for air circulation and kill-switch wiring; and 3) Simplify the power circuitry. One battery pack powers the actuators and the other powers the microprocessors, sensors, and remote power switch. Due to this reduction from three to two battery packs, replacement-weight is added inside the hull. Additionally, several small weights (10-50g) are added to the exterior of the hull to provide fine tuning of buoyancy and weight balance.

B. Electronics

RoboLobster’s electronics, stacked inside the water-tight hull, include driver boards, connector boards, a sensor board, a Digital Signal Processor (DSP) board, power converters, and power switches. Many of the electrical components within RoboLobster were adapted from those designed by Dr. Anthony Westphal for RoboLamprey (2012). RoboLobster has two main processors, an ATmega and a DSP. The DSP runs the nervous system simulation and the ATmega handles sensory processing and integration. For the DSP, communication with a computer is done via a Serial Communication and Programming (SAP) Board and programming is accomplished via a microSD bootloader. Depending on how many values the DSP prints to the serial line, the calculation rate will change. To handle this limitation the DSP always prints one integer value representing the mode (e.g. user control or ATmega control) and three characters
conveying the predominant command activity (e.g. “F-B” represents the most spiking activity in the Forward Left and Backward Right commands). This limited communication, outlined in more detail in the Appendix C, allows for rapid network calculation while still conveying information to a user. For the ATmega, communication with a computer is accomplished with the FTDI\textsuperscript{4} protocol and programming is done using an in-system programming (ISP) interface programmer. A connector board transfers the spike signals from the DSP to the driver board stack. The driver boards have a modified footprint to allow for mounting and connection to RoboLobsters 50 nitinol actuators.

i. **Driver boards**

A stack of constant-current driver boards amplifies a digital signal, in this case a neuron spike, to drive the 50 nitinol actuators on RoboLobster (Figure 6.10). The driver board stack was the same used on RoboLamprey except for slight modifications to the form factor and to the metal–oxide–semiconductor field-effect transistors (MOSFETs) used (Figure 6.11). RoboLamprey used a single driver board stack to accommodate 12 actuators. RoboLobster’s 50 actuators require 5 driver board stacks so mounting holes were added to the rectangular boards to allow for vertical stacking within the hull on four threaded rods that screwed into the end cap (Figure 6.11, bottom). Due to exploding electrical components on the boards during operation of RoboLobster, a MOSFET\textsuperscript{5} with a higher current rating was used. Details on electrical components can be found in Appendix D.

\textsuperscript{4} Future Technology Devices International
\textsuperscript{5} P-channel 60v, 0.13Ω Power MOSFET, STmicroelectronics, Part#: STN3P6F6 http://www.st.com/web/en/resource/technical/document/datasheet/DM00066885.pdf
Figure 6.10. *Driver boards mediate excitation-contraction coupling.* Motor neuron spikes pulse current through the driver board stack to the nitinol actuator.
Figure 6.11. *Driver board stack on RoboLobster.*
ii. Connector boards

Connector boards were designed to carry inputs and outputs to the stack of driver boards. A separate connector board carries signals from the DSP running the neural network to the driver boards (Figure 6.12). A 24-pin connector board connects the nitinol actuators to the outputs of the high current driver board (Figure 6.13). The leads from this connector are potted with epoxy in the hull’s through-holes (Figure 6.14).

![DSP-driver board connector and remote power switch](image)

Figure 6.12. DSP-driver board connector and remote power switch. Outputs to five driver board stacks are on the right. The signals are received from a ribbon cable connected to the TI DSP (bottom left). A remote relay power switch is mounted on the top left and comprises part of the power circuitry (see Figure 6.19).
Figure 6.13. *Fully assembled driver board stack.* Five driver board stacks (high and low current boards) with 24 output lines to nitinol actuators (colored wires). The connector board on the top of the stack has no actuators attached. The connection to the DSP is made by the tall board on the right (see Figure 6.12).
iii. Nervous system DSP

A custom board with a Texas Instruments DSP chip (TMS320C6727) was designed by Mark Hunt of Ariel System to operate the robot (Figure 6.15). This board was also used to operate RoboLamprey (Westphal et al., 2011) and the only differences to its implementation in RoboLobster were in software that is described later. Earlier studies showed that the chip, when running at 1.8GHz, was capable of simulating up to 2000 DTM neurons in real-time (Ayers, Rulkov, Brady, Westphal, & Hunt, 2008). Figure 6.16 shows a schematic of the communication to and from the DSP chip. Code is loaded using the microSD card and spike outputs are passed through the J2, J3 and J4 connectors (shown in the right panel of Figure 6.15). A dual UART (Universal Asynchronous Receiver/Transmitter) protocol provides serial information between
the DSP and a computer and between the DSP and an Atmel chip (marked as ‘Robot Interface’ in Figure 6.16). Two power supplies are provided: 3.3v for the DSP and 1.2v for the SDRAM (16MB).

Figure 6.15. DSP board on RoboLobster. Left shows the microSD card slot and the TI DSP chip running the ENS. Right shows three connectors on the bottom of the board that pass signals to the sensor board (and eventually the driver boards).

Figure 6.16. DSP communication overview. See text for details.
iv. **Power/sensor board**

The DSP board attaches to RoboLobster’s power/sensor board (Figure 6.17) via the three 42-pin connectors (right panel of Figure 6.15). The power/sensor board receives 11.1v input from the batteries which is converted into 5v to power the DSP, I$^2$C bus and driver board logic reference while 3.3v powers a 3-axis accelerometer. The board also has an Atmel ATmega2560$^6$ mounted on the underside that handles sensory processing and sensory neuron network operation. A programming port is used to program the Atmel chip using an AVRISP mkII programmer$^7$ and Arduino software. A 5v FTDI cable$^8$ converts USB to serial between the computer and the ATmega. The board also handles signals from a tactile sensor and an I$^2$C bus to connect to the Atmel chip. Figure 6.18 shows the DSP board attached to the power/sensor board.

---

6 http://www.atmel.com/devices/atmega2560.aspx  
8 https://www.sparkfun.com/products/9718
Due to the high current draw of RoboLobster, the power circuitry had to be carefully designed. One 11.1v, 5000mAh 25C LIPO battery pack\(^9\) powers the DSP board and sensors. This is controlled with a remote RF switch\(^{10}\). The other battery pack powers the actuators and is connected to the system when the first battery is turned on (via an automotive relay\(^{11}\)). This means that the remote switch must be activated before the actuators will be powered. As an additional safety measure, an external waterproof switch\(^{12}\) on the end cap was wired in line with the coil wire of the relay providing a backup method to cut actuator power. A wiring diagram overview of the power circuitry is provided in Figure 6.19.

---


RoboLobster’s electronics hardware layout is summarized in Figure 6.20. The ATmega polls sensor inputs from an I2C interface and a digital pin every 20 ms. It runs sensory processing and simulates the sensory neuron networks. It uses serial UART lines to communicate with a computer and the TI DSP. The TI DSP runs the remainder of the ENS (Electronic Nervous System) and outputs spikes to the constant current driver boards. It polls the ATmega and updates spike outputs every 1 ms. A UART interface allows for communication with a computer and the nervous system code is loaded via a microSD card reader.
Figure 6.20. *RoboLobster electronics overview*. An ATmega 2560 sensory processing chip and the TI DSP nervous system chip control the robot. Sensory inputs, communication lines and actuator outputs are shown.

C. Sensors

A key component of RoboLobster’s hardware is the sensor suite that allows for testing simulations of biomimetic exteroceptive reflex networks. The sensors onboard RoboLobster include an accelerometer, a gyroscope (driving a virtual compass), two antenna, two optical flow sensors and a bump sensor. Each sensor will be described and future plans for additional sensors will be discussed.

The 3-axis Freescale™ accelerometer (MMA7361L)\(^\text{13}\) has a range of +/-1.5 g and is mounted directly on the power/sensor board (Figure 6.17). The signals are processed by the Atmel chip. Due to low accelerations produced by RoboLobster’s movement, the accelerometer data is not currently used to modulate behavior and only used for data logging purposes.

Due to magnetic interference, a magnetic compass was not a viable option onboard RoboLobster. An ITG-3200 triple-axis digital gyro\textsuperscript{14} capable of 2000 deg/sec measurements was used to drive a virtual compass and estimate heading. Communication with the Atmel chip is achieved over an I\textsuperscript{2}C interface.

Two antenna detect bend due to hydrodynamic forces. Each antenna consists of two back-to-back FlexPoint Bend Sensors\textsuperscript{®} with polyester overlaminate (Figure 6.21a). The variable resistive sensors are thin and light (Figure 6.21b). A curve of resistance vs. deflection angle of back-to-back FlexPoint Bend Sensors\textsuperscript{®} shows a broad response that is well suited to mediate antennal-based rheotaxic reflexes (Figure 6.21c). Preliminary testing of the unmounted sensors (Figure 6.22a,b) revealed an asymmetrical response profile to deflection angle. Thus the back-to-back paired sensor arrangement was adopted. Thin acrylic (8.25 mm thick) was cut to make the structural body of the antenna (Figure 6.21d). The \textfrac{3}{4}“ bend sensor was affixed to the acrylic with Aquaseal\textsuperscript{TM} adhesive, spanning a gap cut in the acrylic. An ADS1015 12-bit Analog to Digital Converter\textsuperscript{15} received the analog bend sensor inputs and output them on the I\textsuperscript{2}C bus.

RoboLobster carries two Stonyman2 optical flow sensors\textsuperscript{16} attached to a Teensy 3.1 board with an ARM microcontroller (Figure 6.23a). Each sensor is housed within a waterproof enclosure mounted on each side of the claw with the viewing window pointing outwards (Figure 6.23b). The enclosure was printed with ABS plastic on a MakerBot Replicator 2x. An acetone vapor bath was used to smooth the printed layers

\textsuperscript{14} https://www.sparkfun.com/products/11977
\textsuperscript{15} http://www.adafruit.com/product/1083
\textsuperscript{16} Centeye Inc.
and create a waterproof structure.\(^\text{17}\) Acetone was heated to 110 degrees C using the Replicator 2X’s heated build plate and the housing was placed on a platform above the liquid until smoothing was observed. A sample image collected by the optical flow sensor shows a low resolution 10x10 pixel array (Figure 6.23c). Optical flow sensor validation is described in more detail in Chapter 4 (see Figure 4.17d,e).

\(^{17}\) https://www.youtube.com/watch?v=Xj53P2YzYGM
Figure 6.22. Bend sensor testing. **a.** Bend testing set up to measure resistance for different deflection angles. **b.** Close-up of test rig showing stop pin to set deflection angle. **c.** Chart of deflection angle vs. resistance showing asymmetrical response to bend.
The ARM microcontroller was programmed to detect motion from the camera image. Much of the code was taken from example code provided by Centeye Inc.\textsuperscript{18} To function on RoboLobster, an I\textsuperscript{2}C protocol was written for the Stonymann2/Teensy 3.1 sensor apparatus using Arduino software.

Initialization variables include an I\textsuperscript{2}C buffer to hold x-axis and y-axis optical flow values for two sensors [a], two unions to each hold two two-byte signed integers representing optical flow data [b], integer variables to receive the sensor data [c] and an I\textsuperscript{2}C address for the sensor.

\textsuperscript{18}Documentation and example code can be found at <http://www.ardueye.com/>
slave [d]. Unique addresses are assigned to each sensor and the values are bit-shifted right one bit for communication with the master (Atmel chip).

```c
byte i2cOutBuf[5]; //i2c output buffer [a]

union //to get the signed bytes to format properly, a union is used
{
    short i2c_out;
    byte b[2];
    short i2c_out2;
    byte b2[2];
} u1,u2; [b]

int OFXout;
int OFYout; [c]

#define SLAVE_ADDRESS 0x15 //7 bit address 0x2A/0x2B write/read
//#define SLAVE_ADDRESS 0x17 //7 bit address 0x2E/0x2F write/read [d]
```

The requestEvent function [f] writes the data to the I2C buffer using Arduino’s Wire library.

```c
//when a master reads from this slave, the i2c buffer is provided
void requestEvent() {
    Wire.write(i2cOutBuf,4);
}
```

The 2D optical flow values computed using a simplified Lucas-Kanade algorithm (1981), \( of_x \) and \( of_y \), are scaled to appropriately drive the neural network [g]. The data are then formatted as unions [h] and copied to the I2C output buffer [i].
The tactile sensor is comprised of an acrylic bumper with an attached magnet (Figure 6.24). The bumper is attached to the front of the claw assembly via bolts running through compressible springs. When pressure is applied to the bumper, the springs compress and the magnet activates a magnetic reed switch. The sensitivity of the bumper can be adjusted by changing the spring constant of the springs used.
Figure 6.24. *Bump sensor assembly.* A magnet on the bumper activates a magnetic reed switch upon bumper contact and compression of the springs.

RoboLobster was designed to accommodate additional sensors and customized payloads in the future. In particular, a sonar short baseline array implemented on RoboLamprey (Westphal et al., 2011) could be implemented to mediate orientation to a sonar stimulus. Since RoboLobster does not move off of the ocean substrate, the apparatus on RoboLamprey employing three hydrophones could be simplified to use two hydrophones in order to localize a sonar signal in a
2-dimensional plane. Ongoing work on bacterial biohybrid sensors to detect and report chemical stimuli using living cells is also a planned addition to RoboLobster’s sensor suite. A bacterial cell would fluoresce in the presence of a particular stimulus and the light would be detected by a photodiode to interface with RoboLobster’s electronics. This work is part of an ongoing effort to advance synthetic biology techniques in the laboratory of Dr. Joseph Ayers.

D. Nervous system program code

Initial prototyping of ENS code was completed in LabVIEW™ with those methods described in detail in Chapter 2. Once the network structure and parameters were determined, the networks were coded in the C language to load on the DSP chip. An overview of the file structure is shown in Figure 6.25.

The main files in the C language implementation of RoboLobster’s ENS are neuron.h, sdcosmain.c and test2.c. The neuron and synapse parameters are set up in neuron.h. The operating environment and pin assignments are initialized in sdcosmain.c. The test2.c file operates the neuron network, handles sensor inputs from the Atmel chip and user inputs on the serial line, outputs a PWM duty cycle to the driver boards and outputs CPG spike data to the serial line. Other files including gpio.c and uart.c set up the communication protocols and I/O ports.
The final DSP program contains 88 neurons and 216 synapses. There are 8 command neurons, 32 2D neurons in the CPGs (4 per leg) and 48 1D motor neurons (6 per leg). Neuron parameters are declared in the neuron.h file [j] and initialized in the sdcosmain.c file.
The DTM equations as described in Chapter 2 (Rulkov, 2002) are computed in the test2.c file for 2D neurons \([k]\) and 1D neurons \([l]\). The test2.c file is included in the main loop within sdcosmain.c. The equations update iteratively and pass values between calculation cycles. The entire ENS simulation, including the processing of supervisory inputs, printed serial information and spike outputs, computes in 1 ms.

Of the 216 synapses total, 24 are within the CPGs (3 per leg), 88 synapses descend onto motor neurons from CPG neurons (11 per leg), 20 synapses connect elevator neurons that serve as the pacemaker of each leg and orchestrate interlimb coordination, 32 are excitatory synapses from commands to motor neurons (4 per leg), 48 are self-inhibitory synapses at each motor neuron and 4 synapses from turning commands to walking commands. The motor neuron self-inhibition was implemented to save power onboard RoboLobster. After initial contraction of the nitinol actuator, less current is needed to sustain the Austenite state and so the PWM (Pulse-Width Modulation) output can be diminished. We accomplish this with the motor neuron self-inhibition.

```c
// Elevator neuron parameters
#define a1 6.8   // alpha for Elevator neurons
#define s1 -0.6  // sigma for Elevator neurons
#define N_cells 88 // number of neurons

// Neuron parameters
double alpha[N_cells] = {a1,a1,a1,...};   // Alpha control parameter
double sigma[N_cells] = {s1,s1,s1,...};   // Sigma control parameter

double x_n[N_cells];  // x_n+1 voltage
double prevx[N_cells];
double xpp[N_cells];   // x_n-1 voltage

double y_n[32];      // y_n+1 voltage
double prevy[32];    // y_n voltage

int spike[N_cells] = {0}; // Neuron spike
double ineuron[N_cells] = {0.0}; // Neuron current, iteratively updates
```

136
//calculate 2D Neurons
for (i = 0; i<32; i++)
{
    if (prev[i] <= 0.0)
    {
        x_n[i] = (alpha[i] / (1.0 - prev[i])) * prevy[i] * Ineuron[i];
    }
    else
    {
        if (prev[i] < (alpha[i] + prevy[i] + Ineuron[i]) && xpp[i] <= 0.0)
            x_n[i] = (alpha[i] + prevy[i] + Ineuron[i]);
        else
            x_n[i] = -1;
    }
}
y_n[i]=prev[i]-mut((1-prev[i])=sigma[i]-Ineuron[i]);
if (prev[i] > 0 && xpp[i] > 0)
    spike[i] = 1;
else
    {
        spike[i] = 0;
    }
xpp[i] = prev[i];
prev[i] = x_n[i];
prevy[i] = y_n[i];
}

//calculate 1D Neurons
for (j = 32; j<64; j++)
{
    if (prev[j] <= 0.0)
    {
        x_n[j] = (alpha[j] / (1.0 - prev[j])) * sigma[j] + Ineuron[j];
    }
    else
    {
        if (prev[j] < (alpha[j] + sigma[j] + Ineuron[j]) && xpp[j] <= 0.0)
            x_n[j] = (alpha[j] + sigma[j] + Ineuron[j]);
        else
            x_n[j] = -1;
    }
}
if (prev[j] > 0 && xpp[j] > 0)
    spike[j] = spike[j] + 1;
else
    { spike[j] = 0; }
xpp[j] = prev[j];
prenv[j] = x_n[j];
Synapse parameters are declared in the neuron.h file [m] and initialized in the sdcosmain.c file.

```c
//contralateral inhibition parameters
#define xrp1 -1.8      //reversal potential
#define ginh1 0.5      //synaptic strength
#define gamma1 0.999   //relaxation rate
#define N_syn 216      //number of synapses

//synapses parameters
double rev[N_syn] = {xrp1,xrp1,xrp1,...};       //reversal potential
double strength[N_syn] = {ginh1,ginh1,...};    //synaptic strength
double relax[N_syn] = {gamma1,gamma1,gamma1,...}; //relaxation rate

double Isyn[N_syn];    //synaptic current, iteratively updates
double IsynPrev[N_syn]; //previous synaptic current
int synspike[N_syn];   //spikes to corresponding synapse
double xpost[N_syn];   //voltage of postsynaptic cell
```

Each synapse is assigned a number (see Appendix E for complete list) and before calculating the computational equations, the presynaptic cell’s spike and the postsynaptic cell’s voltage must be assigned to the correct synapse. To do this, values from the neuron spike array are assigned to the sequentially numbered synspike variable array and the $x_n$ value of the postsynaptic cell is assigned to the $xpost$ at the index position of the synapse number [n].
Presynaptic inhibition from the command neurons is computed [o].

And the synapse equations are calculated [p].
Motor neuron spikes are binned in the `spikesarray` variable to be used to modulate the PWM output to the driver boards [q].

```c
//maintain running bin of spikes, reset elsewhere
spikesarray[0]=spike[32] + spikesarray[0];
```

After calculating neuron and synapse equations, the synaptic currents ($I_{syn}$) are injected into the corresponding neuron along with random noise and user input [r]. This is where random noise is input into neurons using a random number function (-100≤$x$≤100) and a user determined divisor ($ww$) that is usually greater than 10,000. Noise can be injected into any neuron. For the results presented in Chapter 7, noise was only injected into the Elevator neurons.

```c
//inject current into each neuron from synapses, random noise, and user input (if necessary)
Ineuron[0] = Isyn[90] + (rand(X%280-100))/ww;
```

To produce a PWM output from spikes, code modified from Westphal et al.’s RoboLamprey (2011) was used [s]. A running time is computed and corrected for clock rollover. At given time intervals (`countsperIteration`) a PWM duty cycle is assigned (`Duration`) for different spikes rates. Code is also included to override PWM outputs if spike durations last for extended periods (tonic excitation) which can overload driver boards and waste power. This happens during network startup and sometimes during behavioral transitions.

The original bootloader for the DSP chip which allowed for programming with the C language ENS would not allow files larger than 64kb to load. A secondary bootloader was developed by A. Mark Hunt in order to solve this issue. The make file to compile the program was modified to produce a .m0 file which could be booted to the DSP via the microSD card reader. Details about the programming protocol and make file can be found in Appendix F.
The Arduino code implementation on the Atmel microprocessor is very similar to the C programming language presented above. The microprocessor receives inputs from the optical flow sensors, antennae, gyroscope and bump sensor. It also computes the associated sensory neuron networks containing 27 neurons and 40 synapses. Real-time neuron spike logging is accomplished and user inputs can activate different sensory systems and change network settings on the fly. The ATMega network, along with user input handling and data printing, computes in 20 ms. While this is much slower than the 1 ms network update time for the DSP, it is fast enough to mediate behavioral responses to sensory stimuli. The conduction velocity from brain to telson of the fastest neurons is greater than 15 ms in large lobsters (Govind & Lang, 1976).
To minimize data transfer between the ATMega and the DSP, each microprocessor has a unique set of command neurons. The command neuron spikes from the ATMega driven by the sensor network are sent to the DSP to inject current into its command neurons. The typical synapse equations cannot be used because it takes too much time to pass the double-formatted voltage values on the serial line. This modified command neuron connection is akin to an electrotonic synapse where the presynaptic neuron’s activity is directly coupled to the postsynaptic cell. A summary diagram of this command neuron scheme is presented in Figure 6.26.

![Figure 6.26. Command neuron serial communication. Diagram of communication between command neurons on the ATMega processor and on the DSP. Command neuron spikes of six neurons from the ATMega generate a 6-bit number that is sent on the serial bus to the DSP. ATMega walking command neurons inject current into their respective neurons on the DSP. The high threshold optical flow neuron spikes decrease the injected current of the contralateral forward command.](image)

RoboLobster’s hardware and software allow for robotic implementation of the neuron networks presented in previous chapters. In the next chapter we perform biorobotic analysis of sensory processing and multimodal sensory fusion.
Chapter 7. Biorobotic analysis

With the completed RoboLobster platform described in Chapter 6 and the network hypotheses developed in Chapters 2-5, biorobotic analysis can be undertaken (Webb, 2006). This chapter will explore robot performance and neuron network activity for a range of behaviors. First, the leg movement produced by the walking CPG will be observed. Then each sensory system will be tested independently: claw bump, compass orientation, response to antennal bend, and optical flow-mediated obstacle avoidance. The last section of the chapter will demonstrate multimodal sensory fusion investigating different combinations of sensory input.

A. Methods

When analyzing robot behavior, we generate two types of data: a video recording of the robot’s motor output and the spike activity of the neuron networks operating onboard RoboLobster. To generate static figures from the video data, we undertake a tracking protocol. First, the video file is converted into an image sequence that is imported into open-source ImageJ software.¹ The MTrackJ plugin² is used to manually track points at the tip of the claw and the extreme posterior of the hull on every 10th frame of the image sequence (Meijering, Dzyubachyk, & Smal, 2012). Three tracked frames per second of video is sufficient to produce the static figures. The tracked point data are output in a spreadsheet and imported into MATLAB as four arrays that are 1 column wide by n columns long where n equals the number of frames in the tracked image sequence. A custom script (see Appendix G) plots a vector starting at (ClawX, ClawY) and ending at (TailX, TailY), the coordinates for each point tracked. The script can be modified to plot vectors at different frame densities, for example, plotting the tracked points for

¹ http://imagej.nih.gov/ij/
² http://www.imagescience.org/meijering/software/mtrackj/
every frame represents a temporal step between plots of 0.33 seconds while plotting every 3 frames updates the path once per second. A tracking resolution is chosen to best visualize the data based on the speed, length and self-overlap of the robot’s path. Based on RoboLobster’s walking speed, we typically track position from every 0.33 s (every tracked frame) to every 1.66 s (every fifth tracked frame). While tracking vectors are usually plotted in black, in some instances positions occurring during backwards movement are denoted by red vectors to add clarity. This is only done in cases where it is very clear that backwards movement is occurring as verified by observation of the video data and activation of both Backward command neurons.

There are two ways to record neuron spike data from RoboLobster. Command and CPG neurons can be transmitted off of the DSP via a computer-connected serial line. Due to the processing time needed to output data on the serial bus, neurons cannot be output in real-time. A section of C-code was added to the code presented in Chapter 6 to allow for logging of neuron data. A user command sent from the computer to RoboLobster via the serial line triggers the DSP to begin storing neuron spikes into variable arrays. Once a predetermined number of iterations has passed, the ENS simulation onboard RoboLobster will terminate and the spike data will print on the serial line. In the current implementation, the buffer history length of the serial handling program on the computer is what limits the length of data output. The number of neurons with spikes logged can be varied but as the number of neuron spikes logged is increased, the duration of logging is reduced. This scheme was adopted in order to output spikes from RoboLobster’s CPG during autonomous operation to extend the microprocessor simulations from Chapter 3.

Transmittal of neuron data from the ATMega microprocessor running the sensor neuron networks and command neurons is more straightforward and can be done in real-time using the
FTDI-Serial connection to a computer. The Arduino code programmed on the ATmega prints the spikes to the serial line. Neuron spikes are bitshifted and combined into two 16-bit integers so that each bit of the integer represent a neuron spike and can be set to one or zero. This protocol was adapted from the approach used for RoboBee (Westphal, Blustein, & Ayers, 2013) and maximizes the efficiency by representing the spike data from up to 14 neurons with one integer.

A `startbit` variable allows for parsing of the code on the computer and the `bitShiftNeuron` variable contains spike data from 13 neurons. The other variable, `bitShiftSensor`, contains spike data from 8 neurons and is not shown below. The `startbit` and appended neuron data integer values are output to the serial line every time the main loop is executed and the network simulation iterates.

```cpp

byte startbit = 99;
Serial1.write(startbit);
Serial1.write(bitshiftCommand);  
```

Since both the ATmega and the DSP have a unique set of command neurons, only the ATmega command neuron spikes are shown in this chapter (see Figure 6.26). While their behavior is coupled for the most part, there are differences when there is high threshold optical flow neuron activity. The spikes from the high optical flow threshold neurons are sent to the DSP to mediate inhibition of the contralateral Forward command. Because of this arrangement, the inhibition of the contralateral command by the high threshold optical flow neuron is not observed in the ATmega’s neuron traces presented here.
Two tanks were used for the testing of RoboLobster. One was a 1.1 m x 1.1 m tank with a water depth of 0.4 m. A mounted overhead camera recorded robot behavior for tracking at 30 fps. The video signal was digitized and captured on a Mac computer. A secondary side camera was sometimes used to record the robot and leg movements. The second tank was an oval track with two motors powered by a 12-volt car battery and mounted to generate flow. Water passed through a plastic honeycomb flow diffuser to minimize turbulence. The area of the tank used for robot testing visible from the overhead camera was 0.82 m wide and 1.66 m long with a water depth of 0.5 m (Figure 7.1). The 30 fps video signal was digitized and captured on a Mac computer as a QuickTime movie. An overview of the behavioral tracking set-up can be seen in Figure 7.2.

Figure 7.1. Flow tank for robot testing. Foreground shows area for robot operation. On opposite side of center void are mounted two motors to generate flow that passes clockwise around the oval through the flow diffuser.

---

3 Sony Handycam DCR-SX40
4 Using a Grass Valley ADVC55 Compact Analog / Digital Converter
5 Panasonic Lumix DMC-G2
6 Minn Kota Endura C2 Mounted Trolling Motor
7 Sony XC-77/77CE B/W machine vision camera with a Computer TV 8mm 1:1.3f lens
Figure 7.2. *Video setup for recording of behavior.* Video of robot behavior from an overhead camera is digitized and captured on a computer for tracking using ImageJ and MATLAB. Serial line communication between robot and computer allows for supervisory control, on-the-fly network modification and streaming of neuron data.

B. *Walking pattern generation*

The CPG underlying lobster walking was described and simulated in Chapter 3. Here we implement the CPG neuron network on RoboLobster to validate the results from tabletop microprocessor operation in Figures 3.12 and 3.13. RoboLobster’s leg trajectory has already been tracked and compared to the lobster leg trajectory (Figure 3.11). Figure 7.3 shows transmitted neuron traces of the CPG driving the front left leg of RoboLobster. The spikes are saved in the DSP’s RAM and transmitted after the behavior occurs. Synced side view video screen grabs show how the leg moves with the observed neuronal outputs. As noted before, limitations in data output from the DSP limits the investigation of embedded system operation of
the CPG circuits. The sensory networks provide more information and the opportunity for more powerful analysis.

![Figure 7.3. CPG output on RoboLobster. Neuron traces from embedded ENS operation on RoboLobster’s DSP correlated with video screen grabs showing leg movement. Note that these are the CPG oscillator neurons and not the motor neuron outputs.](image)

C. Exteroception

i. Claw bump

The claw network (Figure 5.6) mediates a back-up response to avoid detected obstacles (Figure 7.4). The response was first shown using a modified claw design without a bumper that utilized magnets on the obstacle to directly activate the magnetic reed switch. While this design would not be effective for avoiding all obstacles, it demonstrated that the bump network mediates avoidance. This design could be adapted to modify behavior around specific magnetic
objects. Neuron traces show the activation of both Backward commands upon contact with the
magnetic obstacle (Figure 7.5). This response decays and a return to forward walking occurs due
to a bias current of 0.125 injected into the Forward commands for this experiment.

Figure 7.4. Tracking of RoboLobster’s response to magnetic claw contact. The filled circle
points mark the tip of the robot’s claw with the line connecting to the posterior end of the hull as
viewed from directly overhead. The robot tracking begins at the circled point and ends at the
point marked by a square. The robot walks forward (to the right) and hits the magnetic obstacle.
The collision initiates a back-up response that is overcome by a forward bias as the robot walks
into the bottom right corner of the tank. The tank is as described in the text with and is 1.1 m
square. Red vectors denote backward locomotion. Plots mark position every 1.5 s.
RoboLobster’s claw bumper design (Figure 6.24) produced similar robot paths (Figure 7.6) and neuron activity (Figure 7.7) to the magnetic claw results. This claw design can respond to any physical obstacle and the force required to activate the bump detector can be modified by adjusting the stiffness of the bumper springs. The path in Figure 7.6 shows several obstacle encounters before the variability in the network causes the robot to avoid the obstacle.
Figure 7.6. Tracking of RoboLobster’s response to claw bumper contact. Tracking details are as described in Figure 7.4 except all position vectors are plotted in black. The robot walks forward (to the right) and hits the obstacle several times. After backing up a few times, the robot eventually avoids the obstacle and ends in the bottom right corner of the tank. Plots mark position every 1 second.

Figure 7.7. Neuron activity of RoboLobster’s response to claw bumper contact. Spike traces of simulated neurons onboard RoboLobster’s ATmega for the four command neurons and the bump sensory neuron. See Figure 5.6 for the network diagram and Figure 7.6 for the tracked robot path.
ii. Heading Control

The compass network (Figure 5.5) mediates yaw and rotation to minimize the error between the actual and desired headings (Westphal, 2012). The heading that RoboLobster is facing at startup is set as the desired heading. This target heading can be modified via serial communication. When heading deviation is high, a corrective turn-in-place is activated (Figure 7.8). In this example, the robot turns clockwise with the left legs walking forwards and the right legs walking backwards. When heading deviation is low, the robot walks forward with a bias turn towards the target heading.

This behavioral output is mediated by the neuronal activity of the ENS simulation onboard RoboLobster’s ATmega processor (Figure 7.9). The corrective turn-in-place is caused by

Figure 7.8. Tracking of RoboLobster’s response to heading deviation. Tracking details are as described in Figure 7.4 except all vectors are plotted in black. The robot turns in place when heading deviation is high. When the target heading is approached, bilaterally biased forward walking is activated to continue the corrective turn while translating forward. Plots mark position every 1.5 seconds.
by the High Heading deviation neuron on the left side exciting the Rotate Right interneuron which in turn activates the Forward command on the left side and the Backward command on the right side. Once heading deviation diminishes, the Low Heading deviation neuron fires biasing forward walking to steer towards the target heading.

![Neuron activity of RoboLobster’s response to heading deviation](image)

Figure 7.9. Neuron activity of RoboLobster’s response to heading deviation. Top trace shows target heading (green) and actual heading (blue). The other traces, from top to bottom, show spikes of four compass sensory neurons, four sensory interneurons, and four command neurons from the ENS simulation onboard RoboLobster’s AT Mega processor. See Figure 5.5 for the network diagram and Figure 7.8 for the tracked robot path.

iii. Optical flow-mediated obstacle avoidance

Optical flow inputs mediate a range of behavioral responses (see Chapter 4) that can be explained by a neuron network hypothesis (Figure 4.10) (Blustein & Ayers, 2010). Due to noise in the optical flow sensor values under low flow conditions, the low threshold optical flow reflexes were difficult to test. Here we demonstrate high threshold obstacle avoidance mediated by optical flow. There are several ways that arthropods generate rotation during translation
including modulation of step frequency, step amplitude or leg trajectory between inner and outer legs (Domenici & Jamon, 1998; Dürr & Ebeling, 2005). The conserved network model proposed called for an increase in walking speed, on one side of the organism to generate turning (Clarac & Chasserat, 1986). While step frequency is modulated in walking honeybees and stick insects (Jander, 1985; Zolotov, Frantsevich, & Falk, 1975), in crayfish it seems modulation of step amplitude plays an important role in generating a curve during walking (Cruse & Saavedra, 1996; Domenici & Jamon, 1998). While the DSP was able to produce varied walking speeds with different neuron parameters (see Chapter 3), the transition between walking speeds was quite slow. When simulated neuromodulation occurred and the $\alpha$ and $\sigma$ values of the Elevator neurons was adjusted, it took up to 20 s for the network to reestablish a stable motor output. Because of this delay we explored other turning mechanisms. Crayfish turns can be generated by a cessation of leg motion on the inside of the turn. In order to increase the magnitude of optical flow-mediated avoidance turns, we modified the network so the high threshold optical flow sensory neurons would inhibit walking on the contralateral side.

A small obstacle was covered in black and white stripes to increase visual contrast and placed in the test tank. RoboLobster was aligned to pass near the obstacle while walking forward and the resulting optical-flow mediated behavior was observed. Only inputs from the optical flow sensors were turned on for these experiments. For an obstacle on the right, an avoidance turn to the left was observed (Figure 7.10). This response was mediated by activation of the high threshold optical flow neuron on the side of the obstacle which excited the ipsilateral forward command and inhibited the contralateral command (Figure 7.11). By biasing activation of the bilateral commands, a turn away from the obstacle was produced.
To verify that the response was equivalent for optical flow on both sides, an obstacle was placed on the left of RoboLobster’s path trajectory. Injected current into the Forward commands initiated forward walking past the obstacle, stimulating an avoidance turn to the right (Figure 7.12). A second avoidance turn to the left was observed later in the trial. This is presumably mediated by optical flow stimuli from the wall of the tank. The avoidance turns in Figure 7.12 are mediated by activation of the high threshold optical flow neuron on the side of the obstacle which excites the ipsilateral forward command and inhibits the contralateral command (Figure 7.13). By biasing activation of the bilateral commands, a turn away from the obstacle was produced. Immediately after the initial turn, the medial (rear-to-front) optical flow stimulus on the opposite side of the obstacle inhibits the obstacle-sided forward command, leading to a ‘straightening out’, i.e. a turn back to the obstacle’s side after passing the obstacle. The second
optical flow-mediated avoidance turn to the left is correlated with the spikes of the right high threshold optical flow neuron.

Figure 7.11. *Neuron activity of RoboLobster’s response to optical stimulus on right.* Traces from top to bottom are: raw optical flow left data, left optical flow high spike, raw optical flow right data, right optical flow high spike, Forward left spikes, Backward left spikes, Backward right spikes, and Forward right spikes. When an object on the right side is passed, the increased optical flow causes the high threshold optical flow neuron to fire exciting the ipsilateral Forward command. See Figure 4.10 for the network diagram and Figure 7.10 for the robot path.
Figure 7.12. *Tracking of RoboLobster’s response to optical stimulus on left.* Tracking details are as described in Figure 7.4. When an object on the left side is passed, the increased optical flow mediates a turn away from the obstacle. Note that the avoidance turn does not occur until the tip of the claw seems to be well past the visual stimulus. This is because the optical flow sensors are mounted posterior to the tracked point on the tip of the claw and detection is delayed. A left turn at the end of the track is likely due to optical flow inputs from the tank walls. Plots mark position every 0.33 seconds.

To further explore the dynamics of the optical flow reflex network, the $\alpha$ and $\sigma$ parameters of the high threshold optical flow neurons were modified and the neural outputs were observed. The test arena was similar to Figure 7.10 with the visual stimulus placed to the right of RoboLobster’s path trajectory. As the $\sigma$ value of the high threshold optical flow neurons is decreased, the avoidance turn diminishes as a result of decreased activation of the ipsilateral forward command (Figure 7.14). This happens because the more negative a value for $\sigma$, the more injected current is necessary to stimulate firing. When $\sigma = -2.83$ a strong biased activation of command neurons is observed (Figure 7.15). When $\sigma = -2.88$, biased activation of command neurons is still observed however there is also a period of contralateral backwards activation.
(Figure 7.16). This may be because of slightly medial optical flow input (rear – front) on the right side that activated the contralateral (left) Backwards command neuron. When $\sigma = -2.98$, the opposite command bias is observed: the contralateral command is more activated (Figure 7.17). This happens because the high threshold optical flow neuron never activates and the low threshold optical flow neurons drive the behavioral response. The Backward left commands are also activated on occasion suggesting that the medial optical flow neurons may be too excitable.

By comparing performance with varied neuron settings, the network can be tuned to mediate obstacle avoidance at the appropriate distance from the obstacle. In these trials, RoboLobster was walking close to the obstacle, thus a $\sigma$ value of -2.88 or above (i.e. more positive) should be adopted to mediate the appropriate corrective turn.

![Figure 7.13. Neuron activity of RoboLobster’s response to optical stimulus on left. The traces and other details are as in Figure 7.11. Biased command activation resulting from high threshold optical flow inputs is correlated with the avoidance turns observed in Figure 7.12. The activation of the Backward left command (and subsequent inhibition of the Forward left command) is presumably due to inhibition from the contralateral medial optical flow sensitive neuron (spike data unavailable). See Figure 4.10 for the network diagram.](image)
Neuron parameters affect optical flow reflex.

a. When $\sigma = -2.83$, a large avoidance turn occurs.
b. When $\sigma = -2.88$, a medium avoidance turn occurs.
c. When $\sigma = -2.98$, very little avoidance turn occurs.
d. Tracked paths from a-c overlaid with starting positions aligned. Grey, $\sigma = -2.83$; blue, $\sigma = -2.88$, green, $\sigma = -2.98$.

RoboLobster obstacle avoidance response, $\sigma = -2.83$. RoboLobster neuron activity with right-sided optical stimulus. The traces from top to bottom are: raw optical flow left data, raw optical flow right data, Forward left command, Backward left command, Backward right command, and Forward right command. The command ipsilateral to the stimulus (Forward right) fires more frequently due to excitatory inputs from the high threshold optical flow neuron.
Figure 7.16. *RoboLobster obstacle avoidance response, $\sigma = -2.88$. Details are as in Figure 7.15. Activation of the Backward left command results from medial (negative) optical flow input contralaterally.*

Figure 7.17. *RoboLobster obstacle avoidance response, $\sigma = -2.98$. Details are as in Figure 7.15. The obstacle avoidance response is eliminated as the low threshold optical flow neuron on the right (not shown) mediates the opposite command bias (Forward left spikes more frequently).*
iv. Antennal bend

Two distinct responses to antennal bend are simulated on RoboLobster (Ayers, 2004). The first occurs with the antennae held forward. This antennal posture allows the robot to detect the predominant flow field and orient into it, as lobsters do. Once the robot is facing into the flow, the antennae will be moved laterally to allow for subtle bilateral differences in bend to be detected. This sensory information mediates corrective turns into the flow during forward translation. See Chapter 5 for more details on these behavioral responses to hydrodynamic flow. The two responses were tested separately in a flow tank. Only antennal reflexes were turned on for these experiments.

With the antennae positioned forward under conditions of lateral hydrodynamic flow, RoboLobster turned towards the flow (Figure 7.18). Neuron activity of the antenna network (see Chapter 5) shows that the upstream antenna is bent medially and the downstream antenna is bent laterally (Figure 7.19). Note that the left medial low neuron is spiking, and not the high neuron as

![Figure 7.18. RoboLobster response to lateral hydrodynamic flow with antennae forward. Tracking details are as described in Figure 7.4. RoboLobster turns counterclockwise into the flow coming from the left. Plots mark position every 0.33 seconds (1 tracked plot every 10 frames of video).](image)
Figure 7.19. *Neuron activity of RoboLobster’s response to lateral flow with antennae forward.* The top four traces, from top to bottom, detect the following types of bend on the left antenna: lateral high, lateral low, medial low, medial high. The next four traces detect bend on the right antenna, from top to bottom: medial high, medial low, lateral low, lateral high. The next four traces show interneuron spikes and the bottom four traces are the command neurons. Under lateral flow conditions, the upstream antenna (left) is bent medially and the downstream antenna (right) is bent laterally, the command neuron activation aligns with the behavior tracked in Figure 7.18. See the text for a discussion of the neuron network performance.

would be expected. This causes some forward translation during the corrective turn and suggests that the neuron parameters should be modified, particularly strengthening the inhibition from the left Rotation interneuron to the left Yaw interneuron. There is not enough medial bend of the left antenna to activate the high medial neuron. Forward translation is not observed in the robot’s tracked path (Figure 7.18) probably because the command was only briefly activated and the forward locomotion generated was not enough to counteract the oncoming hydrodynamic flow. The bilaterally different response to equivalent stimuli (the same flow rate) could be explained by mechanical differences in the sensor, i.e., the left sensor is more sensitive to bend laterally
than the right is to medial bend. Thus the $\sigma$ value for the right medial sensory neurons should be lowered to account for this imbalance.

With the antennae positioned laterally, the robot walks forward and corrects its path to walk into the flow (Figure 7.20). At the outset of the walking trial, the robot begins to veer to the right. Antennal bend mediates a turn left back into the flow. Then the robot straightens out as it continues to walk upstream. Neuron activity from RoboLobster’s ATMega ENS simulation is shown in Figure 7.21. Low lateral bend on the right excites the Yaw Right interneuron that excites the Forward Left command leading to a right turn. When the high lateral bend neuron activates (due to increased bend), the Rotate Left interneuron fires activating the Backwards left command is activated which mediates a turn back into the flow. Left antennal input is almost non-existent due to diminished sensor sensitivity.

With the implementation of these single modality exteroceptive reflex networks on RoboLobster, we can now advance to multimodal sensory integration. In the next section we test
the ENS with different combinations of sensory input to explore potential mechanisms of sensor fusion.

**Figure 7.21.** Neuron activity of RoboLobster’s response to flow with antennae held laterally. Neuron traces are as in Figure 7.19. Turning response into the flow during forward translation is mediated by right antenna input. The left antenna did not produce a response due to diminished sensor sensitivity. See the text for a discussion of the neuron network performance.

**D. Sensory fusion**

Some behavioral reflexes seem to exist in a hierarchy (Grasso, 2001). That is, when sensory information is available for a certain mode of input, that system predominates the generation of the motor output. For example, a feeding lobster will stop feeding if a threatening visual stimulus is provided (e.g. the shadow of a human standing over its tank). Contact with an obstacle via a claw bump may take precedence among sensory inputs as the detected obstacle would impede any in-progress behavior. During a turn mediated by antenna, compass or optical
flow inputs on RoboLobster, a bump would supersede the ongoing behavior and lead to an avoidance response.

i. Antennal bend and collision fusion

The response to antennal bend and claw bump was investigated. With inputs from both sensory systems activated, yaw towards the bend is observed (Figure 7.22). The antennae were manually bent to simulate surge coming from the robot’s right (i.e. from the top of Figure 7.22). The robot turns into this simulated flow. When an obstacle is encountered and activates the bump sensor, a backup response is observed. This is a transient response and after a few seconds the robot returns to forward locomotion. The neuron outputs that produced the behavioral response can be seen in Figure 7.23. During the antennal bend stimulus, the right Rotation interneuron is activated which leads to a right turn. When the claw bumps the obstacle a transient excitation of both Backward commands occurs which drives the back-up response. The right lateral bend low neuron is activated, representing lateral bend of the antenna. However, the simulated flow bent the antenna medially as observed in the video record. This incorrect response is likely due to a mechanical issue with the sensor arrangement and can be fixed with a new antennal bend sensor. Nevertheless, the response elicited by the Rotation interneuron is strong enough to overcome the unexpected right lateral low bend input and mediate a turn into the flow.
Figure 7.22. Tracking of RoboLobster’s response to antennal bend and bump stimuli. Tracking details are as in Figure 7.4. The robot tracking begins at the circled point and ends at the point marked by a square. Initially the robot turns to the right when its antennae are bent to simulate flow from the right. Forward walking commences when antennal bend is released. Collision with an obstacle initiates a transient back-up response after which forward walking resumes. Red vectors denote backward locomotion. Plots mark position every 1.5 seconds.

<table>
<thead>
<tr>
<th>AntLLatHi</th>
<th>AntLLatLo</th>
<th>AntLMedLo</th>
<th>AntLMedHi</th>
<th>AntRMedHi</th>
<th>AntRMedLo</th>
<th>AntRLatLo</th>
<th>AntRLatHi</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Figure 7.23. Neuron activity of RoboLobster’s response to antennal bend and bump stimuli. Spike traces of simulated neurons onboard RoboLobster’s ATMega for the following neurons, from top to bottom: low left medial antenna, bump, four interneurons, and four command neurons. See Figure 7.22 for the tracked robot path. During an antennal bend stimulus, a corrective turn is elicited (both rotation and yaw during translation). Claw contact mediates a back-up response. See text for details.
ii. *Heading deviation and bump fusion*

A similar hierarchy as observed between the antenna and claw networks is proposed for the interaction of claw and compass inputs. During a compass-mediated turn, a claw bump stimulus will cause a back-up response that predominates (Figure 7.24). After backing up, the robot continues its corrective turn towards a target heading. The neuron activity onboard RoboLobster for this behavior shows a turn mediated by the compass network that is interrupted by transient periods of backward locomotion stimulated by the bump sensory neuron (Figure 7.25).

![Figure 7.24. Tracking of RoboLobster’s response to heading deviation and bump stimuli. An initial heading deviation causes a yaw towards the target heading. During the turn an obstacle is bumped which elicits a transient back-up response. The robot continues its heading correction after backing up. Plots mark position every 0.33 s. Tracking details are as described in Figure 7.4.](image-url)
Figure 7.25. Neuron activity of RoboLobster’s response to simultaneous heading deviation and bump stimuli. Top trace shows target heading (green) and actual heading (blue). The other traces, in descending order, show spikes of the following neurons: bump, four compass, four interneurons, and four command neurons from the ENS simulation onboard RoboLobster’s ATMega processor. See Figure 7.24 for the tracked robot path. Note that the neuron traces start before and end after the tracked path.

RoboLobster’s compass and claw network performed similarly when given a different target heading (Figure 7.26). The behavioral response consisted of a yaw to the right and corrective steering during forward locomotion. A collision elicited a transient back-up response.

Neuron network activity (Figure 7.27) was similar to the results presented in Figure 7.25. Hierarchical interaction between the compass and claw networks produces coherent goal-oriented behavior. The inputs from the bump sensory neuron are stronger than those of the compass network to mediate this interaction.
Figure 7.26. Tracking of RoboLobster’s response to heading deviation and bump stimuli: different target heading. An initial heading deviation causes a yaw towards the target heading. During the turn an obstacle is bumped which elicits a transient back-up response (marked by red tracking vectors). The robot continues its heading correction after backing up. Plots mark position every 1.5 s. Tracking details are as described in Figure 7.4.

Figure 7.27. Neuron activity of RoboLobster’s response to heading deviation and bump stimuli with different heading. Details as in Figure 7.25. See Figure 7.26 for the tracked robot path.
iii. *Optical flow and bump fusion*

A similar interaction scheme was adopted for inputs from optical flow sensory neurons and the bump sensory neuron. RoboLobster was presented a lateral visual stimulus (an obstacle) to initiate an avoidance turn. The bump network was activated upon contact with an obstacle that mediated a back-up response (Figure 7.28). Using optical flow and claw contact, RoboLobster was able to negotiate its way around obstacles. The neuron activity for this behavioral output can be seen in Figure 7.29.

![Figure 7.28. Tracking of RoboLobster’s response to right optical flow and bump stimuli.](image)

Tracking details are as described in Figure 7.4. When an object on the right side is passed, the increased optical flow mediates a turn away from the obstacle. Note that the avoidance turn does not occur until the tip of the claw appears to be well past the visual stimulus. This is because the optical flow sensors are mounted posterior to the tracked point on the tip of the claw and detection is delayed. During the turn, an obstacle is bumped twice, eliciting transient back-up responses followed by a resumption of forward walking. Plots mark position every 0.33 s.
Figure 7.29. *Neuron activity of RoboLobster’s response to right optical flow and bump stimuli.* Top two traces show raw optical flow data. The third trace is the bump sensory neuron and the bottom four show command neuron spikes from the ENS simulation onboard RoboLobster’s ATMega processor. See Figure 7.28 for the tracked robot path. Biased forward command activation causes a corrective turn away from the visual stimulus. Bump sensor neuron spikes activate both Backward commands.

The same behavioral response is seen for a visual stimulus placed on RoboLobster’s left side (Figure 7.30). Neuron activity shows similar activation of command neurons for the optical and claw sensory inputs (Figure 7.31). In this trial, some command neuron modulation is seen at the end of the traces after the visual obstacle has been passed. This is likely due to optical flow stimuli from the tank walls.
Figure 7.30. *Tracking of RoboLobster’s response to left optical flow and claw stimuli.* Tracking details are as described in Figure 7.4. When an object on the left side is passed, the increased optical flow mediates a turn away from the obstacle. Claw contact causes the robot to back up. Plots mark position every 1 s.

Figure 7.31. *Neuron activity of RoboLobster’s response to left optical flow and claw stimuli.* Details as in Figure 7.29. See Figure 7.30 for the tracked robot path. Biased forward command activation causes a corrective turn away from the visual stimulus. Bump sensor neuron spikes activate both Backward commands. At the end of the traces, optical flow inputs from the tank walls elicit turning.
iv. **Heading deviation, optical flow and bump fusion**

After investigating the interaction of different pairs of sensory inputs, we can explore the performance of RoboLobster when driven by three modes of exteroception. The robot was presented with a visual obstacle on the right and given a target heading to the right (Figure 7.32). After turning to the right towards the target heading, optical flow on the right side increased as a visual obstacle was approached. This input caused an avoidance turn back to the left. Compass and optical flow inputs summated in the ENS implementation. At the end of the trial, RoboLobster bumps into the tank wall and backs up. The neural activity of this behavior was transmitted off of the robot’s ATMega chip via serial communication (Figure 7.33).

![Figure 7.32. Tracking of RoboLobster’s response to three modes of sensory stimuli. Tracking details are as described in Figure 7.4. RoboLobster turns towards a target heading, visually detects an obstacle mediating an avoidance turn and bumps the tank wall, leading to a back-up maneuver. Plots mark position every 0.33 s.](image_url)
Figure 7.33. Neuron activity of RoboLobster’s response to three modes of sensory stimuli. Top three traces, from top to bottom, are raw sensor data for heading (target: green; actual: blue), optical flow left, and optical flow right. Neuron spike traces are below, from top to bottom, for: claw bump neuron, 4 compass neurons, 4 interneurons, and 4 command neurons. See Figure 7.32 for the tracked robot path and the text for details of neuron network operation.

At the beginning of the trial, heading deviation drives a corrective turn: the Forward Left and Backward Right commands are activated by the Rotate Right interneuron. This generates a turn to the right. As heading deviation diminishes and the low heading deviation neuron spikes, the Yaw Right interneuron is activated and biases forward locomotion to the right. When the visual obstacle is encountered, the increased optical flow causes forward movement to be biased to the left to mediate an avoidance turn. Claw contact excites the Bump sensory neuron which drives both Backward commands. When this transient response fades at the end of the trial, forward walking resumes, however, optical flow stimuli from the tank walls cause some turning and rotation.
Plausible implementations of RoboLobster’s integrated layered sensory reflexes instantiated as simulated networks of neurons and synapses have been presented in this chapter. We demonstrate behavioral outputs generated by simulated neural activity that mimic observations from the biological system. However, this is just the beginning. With this powerful hypothesis-testing platform, the range of future experiments is vast. In the next chapter, recommendations for further experiments will be discussed along with a comprehensive overview of conclusions drawn from the work so far.
Chapter 8: Conclusions, opportunities and recommendations

We have demonstrated a comprehensive extension to a pre-existing biomimetic platform, RoboLobster, which serves as an embodied nervous system simulation to test neurobiological hypotheses. In this chapter we will discuss conclusions from this work, lessons learned, and recommendations for future work.

Many of the components onboard RoboLobster are not new, such as CPG-inspired walking and bio-inspired optical flow reflexes, but the implementation using neuron network control in such a broad system is novel. We demonstrate a new type of robot controlled by a nervous system simulation adapted for underwater walking (Westphal, Rulkov, Ayers, Brady, & Hunt, 2011). This natural architecture shows promising life-like locomotion and will serve as a base for control of a diverse array of biomimetic robots.

RoboLobster was used to test and support biological control principles. The platform presented plausible implementations of Braitenberg’s (1965; 1986) hypotheses using a neuronal control structure. The principle of decussation has been understood for over a century (Harris, 1904; Loeb, 1918), but implementation on robots has been mainly limited to algorithmic abstractions. Our embedded simulation of the ENS allows us to mimic and study hypotheses about the lobster’s control architecture through a synthetic approach.

RoboLobster’s performance demonstrates that simple neuron network hypotheses can produce a variety of behaviors on a robot. The platform specifically demonstrated support for a previously described lobster walking CPG model (Ayers, 2002), several previously described exteroception models (Ayers, 2002; Blustein & Ayers, 2010; Kennedy & Davis, 1977) as well as the general command neuron, coordinating neuron, CPG organizational framework of nervous systems (Kennedy & Davis, 1977; Stein, 1978). It also demonstrated the effectiveness of
summating and hierarchical interaction between sensory modalities. Some of the timing
dynamics of the robot’s control architecture were captured by the dynamics of neuronal
processes rather than programmed clocks. By modulating synapse strength, the intensity and
duration of sensory input was optimized. Changes to neuron excitability also modified
behavioral responses, qualitatively and temporally. Moving away from the language of if-then
statements and to the language of neurons and synapses is a powerful advance in biorobotics that
forms the basis of a conserved control architecture for walking, swimming (Westphal et al.,
2011), and flying robots (Westphal et al., 2013).

RoboLobster also represents one of the first demonstrations of bio-inspired optical flow
control on an underwater walking robot. Bio-inspired optical flow reflexes have been
implemented on a variety of land- and air-based robots (see Chapter 4) but applications
underwater have been limited primarily to lateral line implementations on swimming robots
(Klein & Bleckmann, 2011). Adding vision capabilities extends the robot platform and broadens
the scope of testable biological hypotheses.

With the development of such a powerful biomimetic robot platform, there are many
opportunities to conduct further experiments. The role of chaotic activity in the nervous system
should be explored. Noise can be varied at every neuron and synapse in the network and
chaotically firing neurons can be added. Experiments modifying chaotic neural inputs and
general noise levels in the sensory neurons, CPG neurons, and motor neurons can easily be
performed.

Direct comparisons between robot and animal behavior should be extended. Ongoing
work investigating lobster response to hydrodynamic flow with compromised sensory systems
has a direct experimental correlate on RoboLobster (e.g. running the robot with only one
antenna). Experiments in more controlled sensory environments should be performed. By constructing experiment-specific tanks with controlled lighting and sensory stimuli, more careful testing of the neuron networks can be undertaken. A flow tank with moveable visual stimuli would be of particular benefit to extending the study of multimodal sensory integration. This would allow for the investigation of potential summing inputs from antennal bend and optical flow reflex networks. By adding moveable claw and tail control surfaces, more complete hydrodynamic flow responses could be implemented. Other hypotheses to test include alternate limb coordination mechanisms and the role of anti-dromic motor neuron activity projecting to CPG interneurons.

To complete the biorobotic cycle, neuronal activity patterns from RoboLobster should be compared to those of freely-behaving lobsters. Ongoing efforts to perform \textit{in vivo} neuron recordings will allow for direct comparison to the robotic simulation in similar real-world conditions. One hypothesis supported by robotic implementation to look for in the biological system is the presence bilaterally independent command neuron populations. Also, efforts should be made to record from coordinating neuron axonal projections between segments in order to better understand this system’s \textit{in vivo} operation. The robotic implementation suggested one plausible arrangement that may be an element of the biological system.

As described throughout this work, the behavioral output of the robot is a marker of network hypothesis performance. Deficiencies in the models presented have been addressed on an ad-hoc basis. Typically, deficiencies can be attributed to one of two underlying reasons: 1) There is a problem with the hypothesis being tested, or 2) There is a problem with the biomimetic platform. It is important to pay attention to this distinction. With the former problem, the biorobotic process can progress. With the latter, the biorobotic process is flawed and the
biomimeticist must intervene to improve the physical model. A complete embodied robotic simulation is, by necessity, an abstraction of the natural system. We cannot precisely mimic the activity of every cell, the action of every effector, or the dynamics of every sensor. Our biological knowledge has gaps that need to be filled in with theory and parsimonious hypotheses; and possible designs are limited by our engineering techniques. Luckily these fields are rapidly advancing, particularly through the development of advanced neural recording techniques (Bath et al., 2014) and the development of soft robotics (Tolley et al., 2014) and synthetic biohybrid devices (Williams, Anand, Rajagopalan, & Saif, 2014).

Since the biorobotic process driving RoboLobster’s development is an iterative one with constant exchange between engineers and biologists, there is much room for improvement and convergence in both the biological and robotic models. More sensors could be added; notably missing from the current implementation are chemosensors (Grasso & Atema, 2002). A short-baseline sonar array for homing and communication developed for RoboLamprey could also be added to RoboLobster (Westphal et al., 2011). Proprioceptors on the legs are also an important potential addition to the robot. Joint stretch receptors and load receptors on the dactyls could help expand the walking model to include more biologically accurate simulations of the coordination of CPG outputs (Clarac, Vedel, & Bush, 1978; Mill & Lowe, 1972). Better sensor calibration could benefit RoboLobster. Hysteresis in the antenna bend sensors and drift in the gyroscope-driven compass limited the accuracy of sensor inputs. Improving these aspects could allow for longer duration sensor-driven experiments without the need to pause for sensor calibration.

The ENS simulated on RoboLobster could be expanded. More neurophysiological work could inform more functional components of the lobster’s nervous system to incorporate into the ENS. Current models could be extended to include higher-level behavioral sequencing and
incorporate neuronal principles like population coding (Kristan & Shaw, 1997) and antidromic activity (Cattaert & Bevengut, 2002). Simulated neuromodulation could be improved as the current implementation is too slow for operation on the robot. An integer-based DTM model (Rulkov, 2002) could be developed to improve data transfer between the two processors on the robot and to eliminate the need for the abstract electrotonic synapse model currently used.

RoboLobster is an important step forward in the field of biorobotics. It extends ENS control, first presented in RoboLamprey (Westphal et al., 2011), to a walking system. While it is continuously possible to improve the model, this broad biomimetic platform represents a new standard for embodied nervous system simulation. A common electronics, actuator and sensor architecture verifies the command neuron, coordinating neuron and CPG framework for sensorimotor integration. RoboLobster opens the door to a wide range of computational neuroscience experiments that were not possible before. This major effort to develop such a comprehensive model will continue to aid scientific investigations in the years to come.
References


Holst, E. V., & Mittelstaedt, H. (1950). The principle of reafference: Interactions between the


corridors. *IEEE International Conference on Robotics and Biomimetics (ROBIO).* Guangzhou, China.


Sillar, K., & Skorupski, P. (1986). Central input to primary afferent neurons in crayfish,


195


Wang, H., Ando, N., & Kanzaki, R. (2008). Active control of free flight manoeuvres in a


Appendix A

NeuroBots curriculum

- Programming guide
- Classroom workbook
- Sample program readme file
Step by step NeuroBots guide to LabVIEW and Lego NXT

LabVIEW is a graphical programming environment that we use to control Lego Mindstorms robots. This type of programming differs from traditional written code as its program components are represented by different icons that are wired together to create the final program. This guide provides an introduction to this approach to programming.

A. Launch LabVIEW

- *On a PC.* Click ‘Start’ on the bottom left corner of the screen. Select ‘Programs’ and click on LabVIEW 2010.

- *On a Mac.* Click on the LabVIEW icon in the dock.

B. Start a new program

C. Tour of LabVIEW

1 – The BLOCK DIAGRAM is where you will do most of your programming
2 – The FRONT PANEL is where values can be input by the user or displayed
3 – The TOOLS PALETTE is where you change the function of the mouse pointer
   (If not visible, launch by clicking 'Tools Palette' under the 'View' menu)
4 – The FUNCTIONS PALETTE is where you select the blocks that you string
together to make your program.
   (If not visible, launch by clicking the 'Functions Palette' under the 'View'
menu. Or by right clicking on the Block Diagram window. Then click the
thumb tack for the Functions Palette to remain on the screen.

D. Getting help

If you need help with a programming block, click 'Show Context Help' under the Help menu
(or press Command+Shift+H). Then single click on a block in the Block Diagram. The
Context Help window will explain the block’s purpose and its connections.
E. Tools explained

For most of what we will do we will let LabVIEW pick the tool for us to use. You will notice that the mouse pointer sometimes changes as you move it to different parts of the screen. This is because we have the Automatic Tool Selection button highlighted. Leave it as it is for now.

F. Functions Palette overview

You will use the icons from the Functions Palette to write your code. You will quickly learn where commonly used icons are located. We will only use icons under the NXT Robotics header as shown below. If you are unsure of where to find an icon, you can always type a related keyword into the Search field.
The NXT Programming header contains many blocks that you will use in your programming.

- **Math** - equations/calculation
- **Boolean** – True/False statements
- **Structures**
  - **Flat sequence structure** - allows setting of program execution order
  - **While loop** - allows you to start or stop a section of code depending on specified conditions
  - **Case structure** - allows a True/False input to switch between different sections of code

Structures allow for organizing how a program is run.
The NXT I/O section contains functions that allow you to control inputs, outputs and other functions of the NXT brick.

**G. Running a motor**

Now you are ready to begin programming. You will start by writing a program that will run a motor.

1. Click the Motor Control block in the Functions palette and then click in the Block Diagram window to place the Motor Control block. If you cannot find a specific block refer to section F or run a search in the Functions Palette. Your Block Diagram should look like this:
2. Right click on the 'Output Port' node and select 'Create' => 'Constant'.

3. Click the blue arrowhead to specify the port on the NXT that you will plug the motor into. For now use Port A.

4. Specify a motor speed by right clicking on the 'Power' node and selecting 'Create' => 'Constant'. You can change the constant value to a number from 0 to 100 for the desired motor speed. For now use 75.
5. The program as written will only run the motor for a brief period. For the motor to continuously run you must place your code within a While Loop. On the Functions Palette, click on the While Loop icon under ‘NXT Programming’=>’Structures’.

6. In the Block Diagram window, click and drag a box around the written code, then release.

7. Once the While Loop is placed, you will next set the conditions for this to run. We will set it so this loop runs indefinitely. To do so, mouse over the Loop Condition Terminal (it’s the stop sign in the bottom right-hand corner of the While Loop). When the pointer turns into a hand, click once and the stop sign will change to a green circle. Now we will tell the loop when to run (rather than when to stop as when the stop sign is showing).
8. Right click the input node to the Loop Condition Terminal and select 'Create Constant'. Click the newly placed constant so that it displays a "T" for true. You just created a True constant that inputs on the Continue If True node so that the while loop will always run.

Each time the While Loop runs the 'i' counter outputs the cycle number. This counter can be helpful in keeping track or controlling the number of times the code runs. Your completed code should look like this:

9. Now save your program with a descriptive name. To put this program on your NXT Brick, attach your device to your computer with the USB cord. Turn your NXT on and select 'Target to NXT' under the File menu. Then click the Deploy button (dashed down arrow) in the Block Diagram window.

After deploying is complete you will hear two beeps. Detach the NXT from the computer and connect a motor to Port A. To run the program, navigate to your program using the buttons on the NXT. The Orange button is select, the gray rectangle is back and the arrows move the cursor. Select 'My Files' then 'Software Files' and then your program. When you’re ready to run the program, select 'Run'.
H. Adding a touch sensor

Now let’s add a touch sensor that will turn on the motor when pressed.

1. From the NXT I/O menu on the Functions Palette, click the ’Read Sensor’ icon and place inside your While Loop.

2. We will now set up the sensor and our code so that when the touch sensor is pressed, the motor will run. If the button is not pressed, the motor will not move. First we select the sensor type we will use. Click the purple arrowhead on the sensor icon you just placed and select ’Read Touch’=>’Pressed’ (this is the default for this icon). Right click the Port node on the sensor icon and click ’Create Constant’ then select the input port for the sensor (Port 1 in this case). This icon will output a True constant if the button is pressed and False constant if it is not pressed through the Yes/No block output.

3. To have different code operate for True and False instances, we use a Case Structure. Select a Case Structure under NXT Programming=>Structures. Click and drag around the code that runs the motor. Wire the Yes/No sensor output to the Case Selector question mark on the Case Structure. Your code should look like this:

Now if the button is pressed, it will send TRUE and the code in the True box of the Case Structure will run. If the button is not pressed, a FALSE will be sent and the code in the False box will run. The motor code we wrote automatically is placed in the True condition of the case structure.
4. Now we have to specify that the motor should not run when the button is not pressed. At the top of the Case Structure click the right arrow next to TRUE to switch to the FALSE case. We will now place a Motor Control icon and set the speed of the motor to zero in this FALSE case. Try this step on your own. If you need help, refer p. X where you first set up the motor to run. Here we will use zero as the speed of the motor instead of 75.

5. Save your program, deploy, attach a touch sensor to Port 1 and you are ready to test it out!
Neurobots
Learning Neuroscience through Legos.

By Blustein, D. H. and Schultheis, K. F.
Welcome to the NEUROBOTS Program. Throughout the next few weeks we will be learning about neuroscience and animal behavior. We will be using both real crayfish as models for our animal behavior and we will design robots that interact with the environment in much the same way that real animals do.

Some terms to know:

Neuroscience is the study of:

The nervous system is divided into two parts. The ______________ nervous system consists of the brain and the spinal cord. The ______________ nervous system consists of the network of nerves in the rest of the body.

Fill in the blanks in the diagram below.

Label the neuron below.

____

____

____

____
Using the word bank below solve the Neurobots crossword puzzle.

<table>
<thead>
<tr>
<th>Across</th>
<th>Down</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. This part of the neuron passes messages along to the next neuron.</td>
<td>2. The study of the neural basis of behavior.</td>
</tr>
<tr>
<td>5. The study of nervous systems and neurons.</td>
<td>3. This part of a neuron receives messages from other neurons.</td>
</tr>
<tr>
<td>6. This part of the nervous system is responsible for the heart beating, breathing, pupil dilation, etc.</td>
<td>4. This part of the nervous system consists of the network of neurons in the rest of the body.</td>
</tr>
<tr>
<td>9. A codified study of animal behavior.</td>
<td>5. This type of cell passes messages throughout the body.</td>
</tr>
<tr>
<td>7. This part of the nervous system consists of the brain and spinal cord.</td>
<td>8. This part of the nervous system is responsible for actions like walking, skipping, jumping, etc. It is a part of the peripheral nervous system.</td>
</tr>
</tbody>
</table>

**Word Bank:**
- Autonomic
- Axon
- Central
- Dendrite
- Ethogram
- Neuron
- Neuroethology
- Peripheral
- Somatic
Crayfish Ethogram

In order to study the behavior of animals, scientists often make ethograms. An ethogram is a description of the range of behaviors exhibited by an animal. Typically it consists of a list of the defined behaviors exhibited by the animal and then how often those behaviors are observed.

With your partner, observe the crayfish in its maze. In the table below, write down the behaviors you observe and write down descriptions of the behaviors. Some things to keep in mind:

- Avoid anthropomorphizing the behavior. In other words, don’t describe human thoughts or emotions for your crayfish.
- Keep the behaviors discrete. If the crayfish walks towards something and then lifts it those would be two separate behaviors.

<table>
<thead>
<tr>
<th>Behavior</th>
<th>Description of Behavior</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Ethogram Data Collection

After you have made the initial behavior observations with your partner, with your class agree on the final behavior categories of the crayfish and define those behaviors.

Definition of Final Behavior Categories

<table>
<thead>
<tr>
<th>Behavior</th>
<th>Definition of Behavior</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
</tbody>
</table>

“Do you want to know who you are? Don’t ask. Act! Action will delineate and define you.”

-Thomas "Lobster" Jefferson
ETHOGRAM DATA SHEET

Log your data below. Use a stopwatch and every 15 sec. observe the behavior of your crayfish. Put a mark by all of the behaviors that are observed at each time point. When you are finished tally up each behavior. Then determine the percentage each behavior makes up of the total number of observations (i.e. Total times one behavior was observed/ Total number of observations).

Graph your data:

<table>
<thead>
<tr>
<th>Behavior</th>
<th>% of total</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
</tbody>
</table>

215
Ethogram Data Collection

What was the most frequently observed behavior of your crayfish?

If you were to design a robotic crayfish, what behaviors do you think would be most important in the programming of your crayfish? Why?

All animals exhibit various behaviors based on the information that they gather from their environment. In order to process this information crayfish, just like us, rely on their nervous systems.

Based on our previous information about nervous systems explain how an animal might receive and process outside information, such as a sound or smell.
Ethogram Data Collection

How are lobster nervous systems similar to human nervous systems?

We are sort of alike.

But different!
Ethogram Data Collection

If a lobster wanted to lift its leg, what would be the correct sequence of events? Number the actions in the correct order.

___

___

___

___
Readme for NeuroBots Sample Program
Written by Daniel Blustein
Please email with any questions: blustein.d@husky.neu.edu

Files contained in LVfiles.zip:

Readme.txt---- this text file
NeuroBotsSampleBasic.vi----a sample program using several of the Neurobots VI's
NeuroBotsSample.vi---- a more involved sample program

Sub-VI's included:
Actuator.vi
BumpExcitatorySynapse.vi
BumpSensorNeuron.vi
ExcitatorySynapse.vi
InhibitorySynapse.vi
MotorNeuron.vi
PrintSpikes.vi
ProcessingNeuron.vi

Embedded VI's included (the user will not interact with these):
ANeuralCurrentNXT.vi
ASpike.vi


Before beginning with these sample programs it is highly recommended to follow the Neurobots Guide to LabVIEW found here: http://myfiles.neu.edu/blustein.d/neurobots/files/LabVIEWforNXT.pdf

**********Overview of NeuroBotsSampleBasic.vi**********

The program is entirely contained within the block diagram portion of the programming environment. To access this, click on the Window menu and select 'Show Block Diagram' (CTL + E).

The entire program is placed within a While Loop so that it will calculate iteratively. The program will run a motor based on the bumping of a touch sensor. The processing is accomplished using mathematically simulated neurons and synapses represented as virtual
A Bump Sensor output is passed to a Bump Sensory Neuron (green wire connecting 'Yes/No' to 'TouchSensorIN'). The Bump Sensory Neuron output signal is passed to the Bump Excitatory Synapse (blue wire connecting 'spike' and 'Spike from PreNeuron'). The output from the Bump Excitatory Synapse is passed to the Motor Neuron (orange wire connecting 'Output' to 'SynapseIN'). The Motor Neuron output drives the Actuator (orange wire connecting 'MotorOut' to 'Power(75)'). To display the neural activity, the spike from the Motor Neuron is passed to the PrintSpikes VI (blue wire connecting 'spike' to 'spike1'). The 'i' counter is connected to the 'i' terminal of the Print Spikes VI.

You will notice several other pink and orange wires that have not yet been mentioned. One limitation of the NXT brick is that it does not support Feedback Nodes (a function to cleanly pass values to the next calculation cycle). Thus, a somewhat messy system of passing values through Shift Registers is employed. Each Neuron and Synapse has an output that needs to be passed back to itself in the next iteration of the while loop calculation. These Neuron outputs (pink wires) and the Synapse output (orange wire) are bundled and passed through a Shift Register in the upper right portion of the While Loop. Synapses also require inputs from their POST-synaptic neuron (i.e., the neuron they pass information to). Thus, the updated values going into Motor Neuron (the pink wire) are also connected to the 'INPUTfromPOSTneuron' terminal of the Bump Excitatory Synapse.

*******Overview of NeuroBotsSample.vi*******

This program is an expansion of the basic example.

This program has two Bump Sensor inputs, presumably one on each side of the robot (plugged into Ports 1 and 2). You will see that there is a new Processing Neuron VI. This Neuron takes information from the Bump Excitatory Synapse (this is different from the basic example). The Processing Neuron then passes its spike to the Generic Excitatory Synapse which connects to the Motor Neuron as in the previous example. The same Neural circuit is duplicated for the right side of the robot.

Reciprocal inhibition between the Processing Neurons means that only one of those neurons will fire at a time. This in turn means that only one motor will be activated at a time, a fate determined by the sensory input. Reciprocal inhibition is realized with two Inhibitory Synapses, one connecting the right Processing Neuron to the left Processing Neuron, and the second making the opposite connection (left Processing to right Processing).
Due to the higher number of values being passed to the next iteration of calculations, Neuron and Synapse values were separated into separate clusters. Depending on the order that the wired connections are made, a Feedback Node may appear in the block diagram. This will prevent the user from loading the program onto the NXT. Replace any Feedback Nodes with Shift Registers. The best option is to use the values already passing through shift registers instead of creating new shift registers. For example, if a spike output (blue wire) turns into a feedback node, it is not necessary to pass that value through a shift register. Instead, other values passing into the VI receiving the information could be passed through a shift register (some which may have to be passed through shift registers anyway). First try disconnecting the offending feedback node wire (the blue wire in this case). Then disconnect all other inputs into the target VI (the one receiving the blue wire). Now reconnect the wires but start by connecting the blue wire. This will likely not create a feedback node. Connect subsequent wires to the VI. Values from the previous iteration coming from the shift register can be used to prevent the appearance of a feedback node.

The remainder of the program is much the same as the previous example.

******Extending your Program************

Once you feel comfortable with the program, you can expand your it to use different sensor or motor combinations. To do so you will have to create new VI's for Neurons, Synapses and Sensors. Open a VI directly from the file folder. Click the File menu and select Save As. When the dialogue screen appears, make sure that 'Substitute copy for original' is selected and click Continue. On the next screen you will name your file and save it.

This new file can be edited to your needs. In the Neuron VI's, there are three parameters that can be modified: Alpha, Sigma and Mu. Alpha determines the fast spiking dynamics (ie. the level of excitation required to fire), Sigma determines the slow spiking dynamics (ie. the neuron's propensity to burst) and Mu is a step increment that determines the level of coarseness between calculations (this does not usually need to be modified). You can also change the type of sensory input into the neuron. For example, a resistive bend sensor could be used as part of an antenna. The resistance value of the sensor can be transformed to trigger specific sensory neurons.

Synapse values to be adjusted are XRP (the reversal potential, typical values are 2.2 for excitatory synapses and -1.8 for inhibitory
synapses), GSYN (the synaptic strength, from 0 to 1), and GAMMA (the relaxation rate, a value from 0 to 1 that determines the rate the synapse returns to equilibrium after perturbation).

Sensor VI's can be adapted from the VI's provided in the NXT Framework for LabVIEW. Note that sometimes on/off sensor inputs like the bump sensors can oversaturate the neural network. If your situation requires (e.g., if the bump sensor remains pressed for extended periods) you can modify the sensor input so that the bump sensor triggers a spike event inputting in the network rather than generating a continuous injected current.

This framework encourages users to modify as much as possible to come up with novel implementations for neuronal networks on the NXT.
Appendix B

RoboLobster parts

-Parts list

-Parts drawings
### RoboLobster parts inventory

<table>
<thead>
<tr>
<th>Description</th>
<th>Part #</th>
<th>Count needed</th>
<th>Stock material</th>
<th>Drawing #</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Plastic parts</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Hull tube</td>
<td>1</td>
<td>1</td>
<td>4&quot; ID x 4.5&quot; OD acrylic tube</td>
<td></td>
</tr>
<tr>
<td>Back plane</td>
<td>101</td>
<td>2</td>
<td>1/4&quot; Ultem sheet</td>
<td></td>
</tr>
<tr>
<td>P/R muscle mount</td>
<td>102</td>
<td>8</td>
<td>1/4&quot; Ultem sheet</td>
<td>102PRMuscleMount</td>
</tr>
<tr>
<td>P/R joint base</td>
<td>103</td>
<td>8</td>
<td>1&quot; x 1&quot; Delrin bar</td>
<td>103P RJointBase</td>
</tr>
<tr>
<td>P/R joint shaft</td>
<td>104</td>
<td>8</td>
<td>1&quot; PEEK rod</td>
<td>104PRJointShaft</td>
</tr>
<tr>
<td>E/D joint tension adjuster</td>
<td>105</td>
<td>8</td>
<td>1/4&quot; PEEK rod</td>
<td>105EDJointTensionAdjuster</td>
</tr>
<tr>
<td>E/D joint tension adjuster mount</td>
<td>106</td>
<td>8</td>
<td>1/4&quot; Ultem sheet</td>
<td>106EDJointTensionAdjusterMount</td>
</tr>
<tr>
<td>E/D joint shelve</td>
<td>107</td>
<td>8</td>
<td>1/2&quot; PEEK rod</td>
<td>107EDJointShive</td>
</tr>
<tr>
<td>E/D joint fork</td>
<td>201</td>
<td>8</td>
<td>1/2&quot; x 3/4&quot; delrin bar</td>
<td>201EDJointFork</td>
</tr>
<tr>
<td>Upper strut</td>
<td>202</td>
<td>8</td>
<td>3/8&quot; PEEK rod</td>
<td>202UpperStrut</td>
</tr>
<tr>
<td>Upper strut insert</td>
<td>203</td>
<td>8</td>
<td>1/4&quot; PEEK rod</td>
<td>203UpperStrutInsert</td>
</tr>
<tr>
<td>E/F joint shive</td>
<td>204</td>
<td>8</td>
<td>1/2&quot; PEEK rod</td>
<td>204E FJointShive</td>
</tr>
<tr>
<td>Lower strut</td>
<td>301</td>
<td>8</td>
<td>3/8&quot; PEEK rod</td>
<td>301LowerStrut</td>
</tr>
<tr>
<td>Lower strut insert</td>
<td>302</td>
<td>8</td>
<td>1/4&quot; PEEK rod</td>
<td>302LowerStrutInsert</td>
</tr>
<tr>
<td>E/F muscle mount</td>
<td>304</td>
<td>8</td>
<td>1/4&quot; Ultem sheet</td>
<td>304EFMuscleMount</td>
</tr>
<tr>
<td>E/F joint fork</td>
<td>305</td>
<td>8</td>
<td>1/2&quot; x 3/4&quot; delrin bar</td>
<td>305E FJointFork</td>
</tr>
<tr>
<td><strong>Fasteners, SS pins etc.</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0/80 x 1&quot; SS pan head screw</td>
<td>108</td>
<td>16</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Joint Pivot</td>
<td>401</td>
<td></td>
<td>1/16&quot; SS rod</td>
<td>401JointPivot</td>
</tr>
</tbody>
</table>
PART #: 103
PART NAME: P/R Joint base
File name: 103PRJointBase.dwg
Designer: Chris Williams
Date: 12/7/2010
PART #: 104
PART NAME: P/R joint shaft
File name: 104PRJointShaft.SLDprt
Designer: Dan Blustein
Date: 12/10/2010
PART #: 106
PART NAME: E/D Joint tension adjuster mount
File name: 106EDJointTensionAdjusterMount.dwg
Designer: Chris Williams
Date: 12/7/2010
PART #: 201
PART NAME: E/D joint fork
File name: 201EDJointFork.SLDPRT
Designer: Dan Blustein
Date: 12/10/2010
PART #: 202
PART NAME: Upper strut
File name: 202UpperStrut.SLDPRT
Designer: Dan Blustein
Date: 12/21/2010
PART #: 204
PART NAME: E/F joint shieve
File name: 204EFJointShieve.SLDPRT
Designer: Dan Blustein
Date: 12/10/2010
PART #: 205
PART NAME: Upper spring spacer
File name: 205UpperSpring Spacer.SLDPRT
Designer: Dan Blustein
Date: 12/21/2010
PART #: 301
PART NAME: Lower strut
File name: 301LowerStrut.SLDPR
Designer: Dan Blustein
Date: 12/22/2010
Thread with 3/8 X 24 die
PART #: 304
PART NAME: E/F Muscle mount
File name: 304EFMuscleMount.SLDPR
Designer: Dan Blustein
Date: 12/10/2010
PART #: 306
PART NAME: Lower spring spacer
File name: 306LowerSpring Spacer.SLDPRD
Designer: Dan Blustein
Date: 12/21/2010
Appendix C

DSP communication details

- PCB schematic

- ATMega- DSP communication protocol
Method of compiling ATmega command spikes to send to DSP
Six spike values are transmitted using a single ASCII character.
Spike patterns are represented by numerical values that are converted to characters.

<table>
<thead>
<tr>
<th>Command #</th>
<th>Decimal assignment</th>
<th>ASCII equivalent</th>
<th>OF left</th>
<th>FL</th>
<th>BL</th>
<th>BR</th>
<th>FR</th>
<th>OF right</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>63</td>
<td>?</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>64</td>
<td>@</td>
<td></td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>65</td>
<td>A</td>
<td></td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>66</td>
<td>B</td>
<td></td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>67</td>
<td>C</td>
<td></td>
<td></td>
<td>1</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>68</td>
<td>D</td>
<td></td>
<td></td>
<td>1</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>69</td>
<td>E</td>
<td></td>
<td></td>
<td>1</td>
<td></td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>70</td>
<td>F</td>
<td></td>
<td>1</td>
<td></td>
<td></td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>71</td>
<td>G</td>
<td></td>
<td></td>
<td></td>
<td>1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>9</td>
<td>72</td>
<td>H</td>
<td></td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>10</td>
<td>73</td>
<td>I</td>
<td></td>
<td></td>
<td>1</td>
<td></td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>11</td>
<td>74</td>
<td>J</td>
<td></td>
<td>1</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>12</td>
<td>75</td>
<td>K</td>
<td></td>
<td></td>
<td>1</td>
<td></td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>13</td>
<td>76</td>
<td>L</td>
<td></td>
<td>1</td>
<td></td>
<td></td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>14</td>
<td>77</td>
<td>M</td>
<td></td>
<td></td>
<td>1</td>
<td></td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>15</td>
<td>78</td>
<td>N</td>
<td></td>
<td>1</td>
<td></td>
<td></td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>16</td>
<td>79</td>
<td>O</td>
<td></td>
<td></td>
<td></td>
<td>1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>17</td>
<td>80</td>
<td>P</td>
<td></td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>18</td>
<td>81</td>
<td>Q</td>
<td></td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>19</td>
<td>82</td>
<td>R</td>
<td></td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>20</td>
<td>83</td>
<td>S</td>
<td></td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>21</td>
<td>84</td>
<td>T</td>
<td></td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>22</td>
<td>85</td>
<td>U</td>
<td></td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>23</td>
<td>86</td>
<td>V</td>
<td></td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>24</td>
<td>87</td>
<td>W</td>
<td></td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>25</td>
<td>88</td>
<td>X</td>
<td></td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>26</td>
<td>89</td>
<td>Y</td>
<td></td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>27</td>
<td>90</td>
<td>Z</td>
<td></td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>28</td>
<td>91</td>
<td>]</td>
<td></td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Appendix D

RoboLobster electronic components

-Parts list
### Parts list for RoboLobster electronics

#### Power board

<table>
<thead>
<tr>
<th>Name</th>
<th>Component</th>
<th>Package</th>
<th>Digikey item No</th>
<th>Manufacturer part No.</th>
<th>Number</th>
</tr>
</thead>
<tbody>
<tr>
<td>ICs</td>
<td>LT1939</td>
<td>LT1939EDD#PBF-ND</td>
<td>LT1939EDD#PBF</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>C1</td>
<td>Cap 10uF low-esr</td>
<td>CAP 478-4153-1-ND</td>
<td>TPST106K010R1000</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>C2</td>
<td>32pF cap</td>
<td>CAP 490-1254-1-ND</td>
<td>GRM0335C1E300JD01D</td>
<td>2</td>
<td></td>
</tr>
<tr>
<td>C4</td>
<td>Cap 0.1uF</td>
<td>CAP 445-1316-1-ND</td>
<td>C1608X7R1E104K</td>
<td>3</td>
<td></td>
</tr>
<tr>
<td>C7</td>
<td>0.1uF cap</td>
<td>CAP PCC1828CT-ND</td>
<td>ECJ-2VB1E104K</td>
<td>4</td>
<td></td>
</tr>
<tr>
<td>D1</td>
<td>LED Green</td>
<td>IC</td>
<td></td>
<td>2</td>
<td></td>
</tr>
<tr>
<td>D2</td>
<td>LED Red</td>
<td>IC</td>
<td></td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>J2</td>
<td>40 pin con</td>
<td>IC H10455CT-ND</td>
<td>DF15(4.2)-40DP-0.65V(56)</td>
<td>3</td>
<td></td>
</tr>
<tr>
<td>PL1</td>
<td>16WP</td>
<td>DSC S5452-ND</td>
<td>PPTC161LGBN-RC</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>PL5</td>
<td>10WP</td>
<td>DSC S5446-ND</td>
<td>PPTC101LGBN-RC</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>PL6</td>
<td>5WP</td>
<td>DSC ED3628-ND</td>
<td>110-93-628-41-001</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>PL7</td>
<td>2WP</td>
<td>DSC ED3628-ND</td>
<td>110-93-628-41-001</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>R2</td>
<td>Res 470ohm</td>
<td>RES P470AGCT-ND</td>
<td>ERJ-1GEJ471C</td>
<td>2</td>
<td></td>
</tr>
<tr>
<td>R8</td>
<td>Res 47ohm</td>
<td>IC P47DCCT-ND</td>
<td>ERA-2EB470X</td>
<td>4</td>
<td></td>
</tr>
<tr>
<td>R1</td>
<td>Res 1kohm</td>
<td>RES P1.00KLCT-ND</td>
<td>ERJ-2RK1001X</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>R4</td>
<td>Res 10k</td>
<td>RES 311-10.0KHCT-ND</td>
<td>9C06031A1002FKHFT</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>R5</td>
<td>Res 10k</td>
<td>RES 311-10.0KHCT-ND</td>
<td>9C06031A1002FKHFT</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>R6</td>
<td>Res 10k</td>
<td>RES 311-10.0KHCT-ND</td>
<td>9C06031A1002FKHFT</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>R12</td>
<td>Res 1kohm</td>
<td>RES P1.00KLCT-ND</td>
<td>ERJ-2RK1001X</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>R13</td>
<td>Jumper</td>
<td>RES NA</td>
<td>NA</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>R14</td>
<td>Res 2.8k</td>
<td>RES</td>
<td></td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>R15</td>
<td>Res 20k</td>
<td>RES P20.0KLCT-ND</td>
<td>ERJ-2RK2002X</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>R16</td>
<td>Res 1kohm</td>
<td>RES P1.00KLCT-ND</td>
<td>ERJ-2RK1001X</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>SW1</td>
<td>Reed switch RST</td>
<td>IC 88F2984</td>
<td>MDR8-DT 20-30</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>SW2</td>
<td>Reed switch SPDT</td>
<td>IC 420-1000-ND</td>
<td>FR2S1015-051500</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>U1</td>
<td>DSPIC33F256GP7:IC</td>
<td>DSPIC33FJ256GP710-I/FDSPIC33FJ256GP710-I/PF</td>
<td></td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>Component</td>
<td>Package</td>
<td>Digikey item No</td>
<td>Price per unit</td>
<td>Qty</td>
<td></td>
</tr>
<tr>
<td>-----------</td>
<td>---------</td>
<td>-----------------</td>
<td>---------------</td>
<td>-----</td>
<td></td>
</tr>
<tr>
<td>C2 Cap 10uF</td>
<td>CAP 490-1845-1-ND</td>
<td>GRM31MF50J106ZA01L</td>
<td>1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>C3 Cap 0.1uF</td>
<td>CAP 311-1141-1-ND</td>
<td>CC0805KRX7R8BB104</td>
<td>1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>PL1 30pin con</td>
<td>USER IPS1-115-01-S-D</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>PL2 2WP</td>
<td>DSC TLW-104-05-G-S</td>
<td>12</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>PL4 4WP</td>
<td>DSC S1022E-36-ND</td>
<td>PEC36SABN</td>
<td>1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>R5 Res 0.1, 0.5ohm</td>
<td>RES WSLF-.50CT-ND</td>
<td>WSL2010R5000FEA18</td>
<td>12</td>
<td></td>
<td></td>
</tr>
<tr>
<td>R19 Res 20k,1kohm</td>
<td>RES P100LCT-ND</td>
<td>ERJ-2RKF1000X</td>
<td>1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>U1 BSP170</td>
<td>IC BSP170PL6327INCT-ND</td>
<td>BSI1P L6327</td>
<td>12</td>
<td></td>
<td></td>
</tr>
<tr>
<td>U4 5V regulator</td>
<td>IC LM2937IMP-5.0CT-ND</td>
<td>LM2937IMP-5.0/NOPB</td>
<td>1</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Component</th>
<th>Package</th>
<th>Digikey item No</th>
<th>Price per unit</th>
<th>Qty</th>
</tr>
</thead>
<tbody>
<tr>
<td>C8 Cap 4.7uF</td>
<td>CAP 445-1362-1-ND</td>
<td>C2012X5R0J475M/1.25</td>
<td>2</td>
<td></td>
</tr>
<tr>
<td>C15 Cap 0.1uF</td>
<td>CAP 311-1141-1-ND</td>
<td>CC0805KRX7R8BB104</td>
<td>6</td>
<td></td>
</tr>
<tr>
<td>PL3 16WP</td>
<td>DSC S5452-ND</td>
<td>PPTC161LGBN-RC</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>PL2 30pin con</td>
<td>USER IPT1-115-01-S-D</td>
<td>1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>R1 Res 20k</td>
<td>RES P20.0KLCT-ND</td>
<td>ERJ-2RKF2002X</td>
<td>13</td>
<td></td>
</tr>
<tr>
<td>R16 Res 1 kohm</td>
<td>RES P1.00KLCT-ND</td>
<td>ERJ-2RKF1001X</td>
<td>24</td>
<td></td>
</tr>
<tr>
<td>R55 Res 2 kohm</td>
<td>RES P2.00KLCT-ND</td>
<td>ERJ-2RKF2001X</td>
<td>12</td>
<td></td>
</tr>
<tr>
<td>U2 LT1637</td>
<td>SOIC LT1637CS8#PBF-ND</td>
<td>LT1637CS8 OR: LT1637HS8</td>
<td>12</td>
<td></td>
</tr>
<tr>
<td>U18 LT1004-1.2</td>
<td>SOIC 296-14610-1-ND</td>
<td>LT1004DR-1-2</td>
<td>2</td>
<td></td>
</tr>
<tr>
<td>U38 ADUMx400</td>
<td>SOIC ADUM3400ARWZ-ND</td>
<td>ADUM3400ARWZ</td>
<td>3</td>
<td></td>
</tr>
<tr>
<td>Component</td>
<td>Package</td>
<td>Manuf Part No</td>
<td>Distrib</td>
<td>Distri Part No</td>
</tr>
<tr>
<td>-------------------</td>
<td>---------</td>
<td>---------------</td>
<td>---------</td>
<td>----------------</td>
</tr>
<tr>
<td>0.1uF cap</td>
<td>CAP</td>
<td>311-1140-1-ND</td>
<td>CC0805KRX7R8B104</td>
<td>C11 C12 C24</td>
</tr>
<tr>
<td>1WP</td>
<td>DSC</td>
<td></td>
<td>PL18 PL19</td>
<td></td>
</tr>
<tr>
<td>2WP</td>
<td>DSC</td>
<td>ED3628-ND</td>
<td>110-93-628-41-001</td>
<td>PL17</td>
</tr>
<tr>
<td>3MCON</td>
<td>DSC</td>
<td>X</td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>3WP</td>
<td>DSC</td>
<td>S1022E-36-ND</td>
<td>PEC365ABN</td>
<td>PL8 PL10 PL11 PL12 PL12 PL2 PL3 PL9</td>
</tr>
<tr>
<td>4WP</td>
<td>DSC</td>
<td>S1022E-36-ND</td>
<td>PEC365ABN</td>
<td>PL13 PL15 PL16 PL14</td>
</tr>
<tr>
<td>5WP</td>
<td>DSC</td>
<td>ED3628-ND</td>
<td>110-93-628-41-001</td>
<td>PL7</td>
</tr>
<tr>
<td>16Mhz crystal</td>
<td>SOL</td>
<td></td>
<td>644-1135-1-ND</td>
<td>XTAL2 XTAL1</td>
</tr>
<tr>
<td>32pF cap</td>
<td>CAP</td>
<td>490-1254-1-ND</td>
<td>GRM03335C1E300JD01D</td>
<td></td>
</tr>
<tr>
<td>40 pin con</td>
<td>IC</td>
<td>H10455CT-ND</td>
<td>DF15(4.2)-40DP-0.65V(56)</td>
<td>J2 J4 J3</td>
</tr>
<tr>
<td>47uH ind</td>
<td>SOL</td>
<td>DN7725-ND</td>
<td>2474R-21L</td>
<td>L2</td>
</tr>
<tr>
<td>100uF cap</td>
<td>CAP</td>
<td>718-1030-1-ND</td>
<td>593D107X9016E2TE3</td>
<td>C9 C10</td>
</tr>
<tr>
<td>0603YC104KATDA</td>
<td>SM0603</td>
<td>445-1316-1-ND</td>
<td>2474R-21L</td>
<td>C17 C18 C19</td>
</tr>
<tr>
<td>1206YG225ZATEA</td>
<td>SM1206</td>
<td>493-2351-1-ND</td>
<td>2474R-21L</td>
<td>C16 C23</td>
</tr>
<tr>
<td>A6035A100JATDA</td>
<td>SM0603</td>
<td>Low ESR</td>
<td>2474R-21L</td>
<td>C1</td>
</tr>
<tr>
<td>Accel chip</td>
<td>SKDIP</td>
<td>pololu #1246</td>
<td>IC6</td>
<td></td>
</tr>
<tr>
<td>ADM1086</td>
<td>SML</td>
<td>QFP</td>
<td>IC4</td>
<td></td>
</tr>
<tr>
<td>ATmega2560</td>
<td></td>
<td></td>
<td>IC7</td>
<td></td>
</tr>
<tr>
<td>Cap 0.1uF</td>
<td>CAP</td>
<td>445-1316-1-ND</td>
<td>C1608X7R1E104K</td>
<td>C4 C15</td>
</tr>
<tr>
<td>CM05CG101J5OAT</td>
<td>CO402</td>
<td>.1 micro farad cap</td>
<td>C25 C26 C27 C28 C29</td>
<td></td>
</tr>
<tr>
<td>CM05CG220J5OAT</td>
<td>CO402</td>
<td>.22 pico farad cap</td>
<td>C2 C3 C20 C21 C22 C23</td>
<td></td>
</tr>
<tr>
<td>DSpC33F256GP710</td>
<td>IC</td>
<td>DSpC33F256GP710</td>
<td>DSpC33F256GP710-I/PF</td>
<td>U1</td>
</tr>
<tr>
<td>LED Red</td>
<td>IC</td>
<td></td>
<td>D4</td>
<td></td>
</tr>
<tr>
<td>MAX764 -5v</td>
<td>SOIC</td>
<td>MAX764CSA+</td>
<td>MAX764CSA+</td>
<td>IC3</td>
</tr>
<tr>
<td>NC7S04</td>
<td>SML</td>
<td>NC7S04P5XCT-ND</td>
<td>IC1</td>
<td></td>
</tr>
<tr>
<td>R 0.063W SMTF 10K</td>
<td>SM0603</td>
<td>P10KGCT-ND</td>
<td>R15 R20 R21 R22 R23 R24</td>
<td></td>
</tr>
<tr>
<td>R 0.063W SMTF 100K</td>
<td>SM0603</td>
<td>P100KAGCT-ND</td>
<td>R16 R17 R18</td>
<td></td>
</tr>
<tr>
<td>Reed switch RST</td>
<td>IC</td>
<td>88F2984</td>
<td>Newark MDDR-DT 20-30</td>
<td>SW1</td>
</tr>
<tr>
<td>Res 20k,1kohm</td>
<td>RES</td>
<td>P100LCT-ND</td>
<td>ERJ-2RF1000X</td>
<td>R8 R9 R10 R13 R14</td>
</tr>
<tr>
<td>Res 47ohm</td>
<td>IC</td>
<td>P47DCCT-ND</td>
<td>ERA-2AE470X</td>
<td>R11 R12</td>
</tr>
<tr>
<td>Schottky Diode</td>
<td>IC</td>
<td>1N5817DICT-ND</td>
<td>1N5817-T</td>
<td>D2</td>
</tr>
<tr>
<td>Tilt compass</td>
<td>SOL</td>
<td></td>
<td>IC5</td>
<td></td>
</tr>
</tbody>
</table>
Appendix E

RoboLobster neuron network

-Neuron and synapse parameters

-DSP pin mapping to actuators
<table>
<thead>
<tr>
<th>Neuron/Spike #</th>
<th>Neuron</th>
<th>Leg</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Elevator 1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>Elevator 2</td>
<td>2</td>
</tr>
<tr>
<td>2</td>
<td>Elevator 3</td>
<td>3</td>
</tr>
<tr>
<td>3</td>
<td>Elevator 4</td>
<td>4</td>
</tr>
<tr>
<td>4</td>
<td>Elevator 5</td>
<td>5</td>
</tr>
<tr>
<td>5</td>
<td>Elevator 6</td>
<td>6</td>
</tr>
<tr>
<td>6</td>
<td>Elevator 7</td>
<td>7</td>
</tr>
<tr>
<td>7</td>
<td>Elevator 8</td>
<td>8</td>
</tr>
<tr>
<td>8</td>
<td>Depressor 1</td>
<td>1</td>
</tr>
<tr>
<td>9</td>
<td>Depressor 2</td>
<td>2</td>
</tr>
<tr>
<td>10</td>
<td>Depressor 3</td>
<td>3</td>
</tr>
<tr>
<td>11</td>
<td>Depressor 4</td>
<td>4</td>
</tr>
<tr>
<td>12</td>
<td>Depressor 5</td>
<td>5</td>
</tr>
<tr>
<td>13</td>
<td>Depressor 6</td>
<td>6</td>
</tr>
<tr>
<td>14</td>
<td>Depressor 7</td>
<td>7</td>
</tr>
<tr>
<td>15</td>
<td>Depressor 8</td>
<td>8</td>
</tr>
<tr>
<td>16</td>
<td>Stance 1</td>
<td>1</td>
</tr>
<tr>
<td>17</td>
<td>Stance 2</td>
<td>2</td>
</tr>
<tr>
<td>18</td>
<td>Stance 3</td>
<td>3</td>
</tr>
<tr>
<td>19</td>
<td>Stance 4</td>
<td>4</td>
</tr>
<tr>
<td>20</td>
<td>Stance 5</td>
<td>5</td>
</tr>
<tr>
<td>21</td>
<td>Stance 6</td>
<td>6</td>
</tr>
<tr>
<td>22</td>
<td>Stance 7</td>
<td>7</td>
</tr>
<tr>
<td>23</td>
<td>Stance 8</td>
<td>8</td>
</tr>
<tr>
<td>24</td>
<td>Swing 1</td>
<td>1</td>
</tr>
<tr>
<td>25</td>
<td>Swing 2</td>
<td>2</td>
</tr>
<tr>
<td>26</td>
<td>Swing 3</td>
<td>3</td>
</tr>
<tr>
<td>27</td>
<td>Swing 4</td>
<td>4</td>
</tr>
<tr>
<td>28</td>
<td>Swing 5</td>
<td>5</td>
</tr>
<tr>
<td>29</td>
<td>Swing 6</td>
<td>6</td>
</tr>
<tr>
<td>30</td>
<td>Swing 7</td>
<td>7</td>
</tr>
<tr>
<td>31</td>
<td>Swing 8</td>
<td>8</td>
</tr>
<tr>
<td>32</td>
<td>Protractor 1</td>
<td>1</td>
</tr>
<tr>
<td>33</td>
<td>Protractor 2</td>
<td>2</td>
</tr>
<tr>
<td>34</td>
<td>Protractor 3</td>
<td>3</td>
</tr>
<tr>
<td>35</td>
<td>Protractor 4</td>
<td>4</td>
</tr>
<tr>
<td>36</td>
<td>Protractor 5</td>
<td>5</td>
</tr>
<tr>
<td>37</td>
<td>Protractor 6</td>
<td>6</td>
</tr>
<tr>
<td>38</td>
<td>Protractor 7</td>
<td>7</td>
</tr>
<tr>
<td>39</td>
<td>Protractor 8</td>
<td>8</td>
</tr>
<tr>
<td>40</td>
<td>Retractor 1</td>
<td>1</td>
</tr>
<tr>
<td>41</td>
<td>Retractor 2</td>
<td>2</td>
</tr>
<tr>
<td>42</td>
<td>Retractor 3</td>
<td>3</td>
</tr>
<tr>
<td>43</td>
<td>Retractor 4</td>
<td>4</td>
</tr>
<tr>
<td>44</td>
<td>Retractor 5</td>
<td>5</td>
</tr>
</tbody>
</table>
45 Retractor
46 Retractor
47 Retractor
48 Elevator MN
49 Elevator MN
50 Elevator MN
51 Elevator MN
52 Elevator MN
53 Elevator MN
54 Elevator MN
55 Elevator MN
56 Depressor M
57 Depressor M
58 Depressor M
59 Depressor M
60 Depressor M
61 Depressor M
62 Depressor M
63 Depressor M
64 Extensor
65 Extensor
66 Extensor
67 Extensor
68 Extensor
69 Extensor
70 Extensor
71 Extensor
72 Flexor
73 Flexor
74 Flexor
75 Flexor
76 Flexor
77 Flexor
78 Flexor
79 Flexor
80 Forward Left
81 Forward Right
82 Backward Left
83 Backward Right
84 Leading
85 Trailing
Neuron and synapse parameters

<table>
<thead>
<tr>
<th>Neuron</th>
<th>Alpha</th>
<th>Sigma</th>
</tr>
</thead>
<tbody>
<tr>
<td>Elevator</td>
<td>6.8</td>
<td>-0.6</td>
</tr>
<tr>
<td>Depressor</td>
<td>5.5</td>
<td>0.5</td>
</tr>
<tr>
<td>Swing</td>
<td>5.5</td>
<td>0.5</td>
</tr>
<tr>
<td>Stance</td>
<td>5.5</td>
<td>0.5</td>
</tr>
<tr>
<td>Elevator(MN)</td>
<td>3.95</td>
<td>-2.98</td>
</tr>
<tr>
<td>Depressor(MN)</td>
<td>3.95</td>
<td>-2.98</td>
</tr>
<tr>
<td>Protractor</td>
<td>3.95</td>
<td>-2.98</td>
</tr>
<tr>
<td>Retractor</td>
<td>3.95</td>
<td>-2.98</td>
</tr>
<tr>
<td>Extensor</td>
<td>3.95</td>
<td>-2.98</td>
</tr>
<tr>
<td>Flexor</td>
<td>3.95</td>
<td>-2.98</td>
</tr>
<tr>
<td>Commands</td>
<td>3.95</td>
<td>-2.98</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Synapse</th>
<th>reversal potential</th>
<th>synaptic strength</th>
<th>relaxation rate</th>
</tr>
</thead>
<tbody>
<tr>
<td>Elev-Elev contralateral</td>
<td>-1.8</td>
<td>0.02</td>
<td>0.999</td>
</tr>
<tr>
<td>Elev-Dep</td>
<td>-2.2</td>
<td>0.8</td>
<td>0.2</td>
</tr>
<tr>
<td>Elev-Stance</td>
<td>-2.2</td>
<td>0.8</td>
<td>0.999</td>
</tr>
<tr>
<td>Stance-Swing</td>
<td>-3.2</td>
<td>0.999</td>
<td>0.8</td>
</tr>
<tr>
<td>Swing-Pro</td>
<td>2.2</td>
<td>0.1</td>
<td>0.48</td>
</tr>
<tr>
<td>Swing-Ret</td>
<td>2.2</td>
<td>0.1</td>
<td>0.48</td>
</tr>
<tr>
<td>Elev-elevMot</td>
<td>2.2</td>
<td>0.1</td>
<td>0.25</td>
</tr>
<tr>
<td>Stance-depMot</td>
<td>2.2</td>
<td>0.05</td>
<td>0.7</td>
</tr>
<tr>
<td>Swing-extMot</td>
<td>2.2</td>
<td>0.1</td>
<td>0.55</td>
</tr>
<tr>
<td>Swing-flexMot</td>
<td>2.2</td>
<td>0.1</td>
<td>0.5</td>
</tr>
<tr>
<td>Elev-Elev ipsilateral</td>
<td>-1.8</td>
<td>0.05</td>
<td>0.99</td>
</tr>
<tr>
<td>Stance-proMot</td>
<td>2.2</td>
<td>0.1</td>
<td>0.55</td>
</tr>
<tr>
<td>Stance-retMot</td>
<td>2.2</td>
<td>0.1</td>
<td>0.55</td>
</tr>
<tr>
<td>Dep-depMot</td>
<td>2.2</td>
<td>0.1</td>
<td>0.4</td>
</tr>
<tr>
<td>Stance-extMot</td>
<td>2.2</td>
<td>0.1</td>
<td>0.55</td>
</tr>
<tr>
<td>Stance-flexMot</td>
<td>2.2</td>
<td>0.1</td>
<td>0.55</td>
</tr>
<tr>
<td>elevMot self inhibition</td>
<td>-1.8</td>
<td>0.03</td>
<td>0.98</td>
</tr>
<tr>
<td>depMot self inhibition</td>
<td>-1.8</td>
<td>0.04</td>
<td>0.98</td>
</tr>
<tr>
<td>proMot self inhibition</td>
<td>-1.8</td>
<td>0.2</td>
<td>0.99</td>
</tr>
<tr>
<td>retMot self inhibition</td>
<td>-1.8</td>
<td>0.1</td>
<td>0.995</td>
</tr>
<tr>
<td>extMot self inhibition</td>
<td>-1.8</td>
<td>0.2</td>
<td>0.995</td>
</tr>
<tr>
<td>flexMot self inhibition</td>
<td>-1.8</td>
<td>0.2</td>
<td>0.99</td>
</tr>
<tr>
<td>Synapse Set</td>
<td>Synapse #</td>
<td>Presyn cell</td>
<td>Postsyn cell</td>
</tr>
<tr>
<td>-------------</td>
<td>-----------</td>
<td>-------------</td>
<td>--------------</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>2</td>
<td>3</td>
<td>2</td>
</tr>
<tr>
<td>1</td>
<td>3</td>
<td>2</td>
<td>3</td>
</tr>
<tr>
<td>1</td>
<td>4</td>
<td>5</td>
<td>4</td>
</tr>
<tr>
<td>1</td>
<td>5</td>
<td>4</td>
<td>5</td>
</tr>
<tr>
<td>1</td>
<td>6</td>
<td>7</td>
<td>6</td>
</tr>
<tr>
<td>1</td>
<td>7</td>
<td>6</td>
<td>7</td>
</tr>
<tr>
<td>2</td>
<td>8</td>
<td>0</td>
<td>8</td>
</tr>
<tr>
<td>2</td>
<td>9</td>
<td>1</td>
<td>9</td>
</tr>
<tr>
<td>2</td>
<td>10</td>
<td>2</td>
<td>10</td>
</tr>
<tr>
<td>2</td>
<td>11</td>
<td>3</td>
<td>11</td>
</tr>
<tr>
<td>2</td>
<td>12</td>
<td>4</td>
<td>12</td>
</tr>
<tr>
<td>2</td>
<td>13</td>
<td>5</td>
<td>13</td>
</tr>
<tr>
<td>2</td>
<td>14</td>
<td>6</td>
<td>14</td>
</tr>
<tr>
<td>2</td>
<td>15</td>
<td>7</td>
<td>15</td>
</tr>
<tr>
<td>3</td>
<td>24</td>
<td>0</td>
<td>16</td>
</tr>
<tr>
<td>3</td>
<td>25</td>
<td>1</td>
<td>17</td>
</tr>
<tr>
<td>3</td>
<td>26</td>
<td>2</td>
<td>18</td>
</tr>
<tr>
<td>3</td>
<td>27</td>
<td>3</td>
<td>19</td>
</tr>
<tr>
<td>3</td>
<td>28</td>
<td>4</td>
<td>20</td>
</tr>
<tr>
<td>3</td>
<td>29</td>
<td>5</td>
<td>21</td>
</tr>
<tr>
<td>3</td>
<td>30</td>
<td>6</td>
<td>22</td>
</tr>
<tr>
<td>3</td>
<td>31</td>
<td>7</td>
<td>23</td>
</tr>
<tr>
<td>4</td>
<td>16</td>
<td>16</td>
<td>24</td>
</tr>
<tr>
<td>4</td>
<td>17</td>
<td>17</td>
<td>25</td>
</tr>
<tr>
<td>4</td>
<td>18</td>
<td>18</td>
<td>26</td>
</tr>
<tr>
<td>4</td>
<td>19</td>
<td>19</td>
<td>27</td>
</tr>
<tr>
<td>4</td>
<td>20</td>
<td>20</td>
<td>28</td>
</tr>
<tr>
<td>4</td>
<td>21</td>
<td>21</td>
<td>29</td>
</tr>
<tr>
<td>4</td>
<td>22</td>
<td>22</td>
<td>30</td>
</tr>
<tr>
<td>4</td>
<td>23</td>
<td>23</td>
<td>31</td>
</tr>
<tr>
<td>5</td>
<td>32</td>
<td>24</td>
<td>32</td>
</tr>
<tr>
<td>5</td>
<td>33</td>
<td>25</td>
<td>33</td>
</tr>
<tr>
<td>5</td>
<td>34</td>
<td>26</td>
<td>34</td>
</tr>
<tr>
<td>5</td>
<td>35</td>
<td>27</td>
<td>35</td>
</tr>
<tr>
<td>5</td>
<td>36</td>
<td>28</td>
<td>36</td>
</tr>
<tr>
<td>5</td>
<td>37</td>
<td>29</td>
<td>37</td>
</tr>
<tr>
<td>5</td>
<td>38</td>
<td>30</td>
<td>38</td>
</tr>
<tr>
<td>5</td>
<td>39</td>
<td>31</td>
<td>39</td>
</tr>
<tr>
<td>6</td>
<td>40</td>
<td>24</td>
<td>40</td>
</tr>
<tr>
<td>Page</td>
<td>Column 1</td>
<td>Column 2</td>
<td>Column 3</td>
</tr>
<tr>
<td>------</td>
<td>----------</td>
<td>----------</td>
<td>----------</td>
</tr>
<tr>
<td>6</td>
<td>41</td>
<td>25</td>
<td>41 Swing-Ret</td>
</tr>
<tr>
<td>6</td>
<td>42</td>
<td>26</td>
<td>42 Swing-Ret</td>
</tr>
<tr>
<td>6</td>
<td>43</td>
<td>27</td>
<td>43 Swing-Ret</td>
</tr>
<tr>
<td>6</td>
<td>44</td>
<td>28</td>
<td>44 Swing-Ret</td>
</tr>
<tr>
<td>6</td>
<td>45</td>
<td>29</td>
<td>45 Swing-Ret</td>
</tr>
<tr>
<td>6</td>
<td>46</td>
<td>30</td>
<td>46 Swing-Ret</td>
</tr>
<tr>
<td>6</td>
<td>47</td>
<td>31</td>
<td>47 Swing-Ret</td>
</tr>
<tr>
<td>7</td>
<td>48</td>
<td>0</td>
<td>48 Elev-ElevMot</td>
</tr>
<tr>
<td>7</td>
<td>49</td>
<td>1</td>
<td>49 Elev-ElevMot</td>
</tr>
<tr>
<td>7</td>
<td>50</td>
<td>2</td>
<td>50 Elev-ElevMot</td>
</tr>
<tr>
<td>7</td>
<td>51</td>
<td>3</td>
<td>51 Elev-ElevMot</td>
</tr>
<tr>
<td>7</td>
<td>52</td>
<td>4</td>
<td>52 Elev-ElevMot</td>
</tr>
<tr>
<td>7</td>
<td>53</td>
<td>5</td>
<td>53 Elev-ElevMot</td>
</tr>
<tr>
<td>7</td>
<td>54</td>
<td>6</td>
<td>54 Elev-ElevMot</td>
</tr>
<tr>
<td>7</td>
<td>55</td>
<td>7</td>
<td>55 Elev-ElevMot</td>
</tr>
<tr>
<td>8</td>
<td>56</td>
<td>16</td>
<td>56 Stance-DepMot</td>
</tr>
<tr>
<td>8</td>
<td>57</td>
<td>17</td>
<td>57 Stance-DepMot</td>
</tr>
<tr>
<td>8</td>
<td>58</td>
<td>18</td>
<td>58 Stance-DepMot</td>
</tr>
<tr>
<td>8</td>
<td>59</td>
<td>19</td>
<td>59 Stance-DepMot</td>
</tr>
<tr>
<td>8</td>
<td>60</td>
<td>20</td>
<td>60 Stance-DepMot</td>
</tr>
<tr>
<td>8</td>
<td>61</td>
<td>21</td>
<td>61 Stance-DepMot</td>
</tr>
<tr>
<td>8</td>
<td>62</td>
<td>22</td>
<td>62 Stance-DepMot</td>
</tr>
<tr>
<td>8</td>
<td>63</td>
<td>23</td>
<td>63 Stance-DepMot</td>
</tr>
<tr>
<td>9</td>
<td>64</td>
<td>24</td>
<td>64 Swing-Ext</td>
</tr>
<tr>
<td>9</td>
<td>65</td>
<td>25</td>
<td>65 Swing-Ext</td>
</tr>
<tr>
<td>9</td>
<td>66</td>
<td>26</td>
<td>66 Swing-Ext</td>
</tr>
<tr>
<td>9</td>
<td>67</td>
<td>27</td>
<td>67 Swing-Ext</td>
</tr>
<tr>
<td>9</td>
<td>68</td>
<td>28</td>
<td>68 Swing-Ext</td>
</tr>
<tr>
<td>9</td>
<td>69</td>
<td>29</td>
<td>69 Swing-Ext</td>
</tr>
<tr>
<td>9</td>
<td>70</td>
<td>30</td>
<td>70 Swing-Ext</td>
</tr>
<tr>
<td>9</td>
<td>71</td>
<td>31</td>
<td>71 Swing-Ext</td>
</tr>
<tr>
<td>9</td>
<td>72</td>
<td>24</td>
<td>72 Swing-Flex</td>
</tr>
<tr>
<td>10</td>
<td>73</td>
<td>25</td>
<td>73 Swing-Flex</td>
</tr>
<tr>
<td>10</td>
<td>74</td>
<td>26</td>
<td>74 Swing-Flex</td>
</tr>
<tr>
<td>10</td>
<td>75</td>
<td>27</td>
<td>75 Swing-Flex</td>
</tr>
<tr>
<td>10</td>
<td>76</td>
<td>28</td>
<td>76 Swing-Flex</td>
</tr>
<tr>
<td>10</td>
<td>77</td>
<td>29</td>
<td>77 Swing-Flex</td>
</tr>
<tr>
<td>10</td>
<td>78</td>
<td>30</td>
<td>78 Swing-Flex</td>
</tr>
<tr>
<td>10</td>
<td>79</td>
<td>31</td>
<td>79 Swing-Flex</td>
</tr>
<tr>
<td>11</td>
<td>80</td>
<td>2</td>
<td>0 Ipsilateral</td>
</tr>
<tr>
<td>11</td>
<td>81</td>
<td>3</td>
<td>1 Ipsilateral</td>
</tr>
<tr>
<td>11</td>
<td>82</td>
<td>0</td>
<td>2 Ipsilateral</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Ipsilateral</td>
</tr>
<tr>
<td>---</td>
<td>---</td>
<td>---</td>
<td>-------------</td>
</tr>
<tr>
<td>11</td>
<td>83</td>
<td>4</td>
<td>2</td>
</tr>
<tr>
<td>11</td>
<td>84</td>
<td>1</td>
<td>3</td>
</tr>
<tr>
<td>11</td>
<td>85</td>
<td>5</td>
<td>3</td>
</tr>
<tr>
<td>11</td>
<td>86</td>
<td>2</td>
<td>4</td>
</tr>
<tr>
<td>11</td>
<td>87</td>
<td>6</td>
<td>4</td>
</tr>
<tr>
<td>11</td>
<td>88</td>
<td>3</td>
<td>5</td>
</tr>
<tr>
<td>11</td>
<td>89</td>
<td>7</td>
<td>5</td>
</tr>
<tr>
<td>11</td>
<td>90</td>
<td>4</td>
<td>6</td>
</tr>
<tr>
<td>11</td>
<td>91</td>
<td>5</td>
<td>7</td>
</tr>
<tr>
<td>12</td>
<td>92</td>
<td>16</td>
<td>32</td>
</tr>
<tr>
<td>12</td>
<td>93</td>
<td>17</td>
<td>33</td>
</tr>
<tr>
<td>12</td>
<td>94</td>
<td>18</td>
<td>34</td>
</tr>
<tr>
<td>12</td>
<td>95</td>
<td>19</td>
<td>35</td>
</tr>
<tr>
<td>12</td>
<td>96</td>
<td>20</td>
<td>36</td>
</tr>
<tr>
<td>12</td>
<td>97</td>
<td>21</td>
<td>37</td>
</tr>
<tr>
<td>12</td>
<td>98</td>
<td>22</td>
<td>38</td>
</tr>
<tr>
<td>12</td>
<td>99</td>
<td>23</td>
<td>39</td>
</tr>
<tr>
<td>13</td>
<td>100</td>
<td>16</td>
<td>40</td>
</tr>
<tr>
<td>13</td>
<td>101</td>
<td>17</td>
<td>41</td>
</tr>
<tr>
<td>13</td>
<td>102</td>
<td>18</td>
<td>42</td>
</tr>
<tr>
<td>13</td>
<td>103</td>
<td>19</td>
<td>43</td>
</tr>
<tr>
<td>13</td>
<td>104</td>
<td>20</td>
<td>44</td>
</tr>
<tr>
<td>13</td>
<td>105</td>
<td>21</td>
<td>45</td>
</tr>
<tr>
<td>13</td>
<td>106</td>
<td>22</td>
<td>46</td>
</tr>
<tr>
<td>13</td>
<td>107</td>
<td>23</td>
<td>47</td>
</tr>
<tr>
<td>14</td>
<td>108</td>
<td>8</td>
<td>56</td>
</tr>
<tr>
<td>14</td>
<td>109</td>
<td>9</td>
<td>57</td>
</tr>
<tr>
<td>14</td>
<td>110</td>
<td>10</td>
<td>58</td>
</tr>
<tr>
<td>14</td>
<td>111</td>
<td>11</td>
<td>59</td>
</tr>
<tr>
<td>14</td>
<td>112</td>
<td>12</td>
<td>60</td>
</tr>
<tr>
<td>14</td>
<td>113</td>
<td>13</td>
<td>61</td>
</tr>
<tr>
<td>14</td>
<td>114</td>
<td>14</td>
<td>62</td>
</tr>
<tr>
<td>14</td>
<td>115</td>
<td>15</td>
<td>63</td>
</tr>
<tr>
<td>15</td>
<td>116</td>
<td>16</td>
<td>64</td>
</tr>
<tr>
<td>15</td>
<td>117</td>
<td>17</td>
<td>65</td>
</tr>
<tr>
<td>15</td>
<td>118</td>
<td>18</td>
<td>66</td>
</tr>
<tr>
<td>15</td>
<td>119</td>
<td>19</td>
<td>67</td>
</tr>
<tr>
<td>15</td>
<td>120</td>
<td>20</td>
<td>68</td>
</tr>
<tr>
<td>15</td>
<td>121</td>
<td>21</td>
<td>69</td>
</tr>
<tr>
<td>15</td>
<td>122</td>
<td>22</td>
<td>70</td>
</tr>
<tr>
<td>15</td>
<td>123</td>
<td>23</td>
<td>71</td>
</tr>
<tr>
<td>16</td>
<td>124</td>
<td>16</td>
<td>72</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>---</td>
<td>---</td>
<td>---</td>
<td>---</td>
</tr>
<tr>
<td>16</td>
<td>125</td>
<td>17</td>
<td>73</td>
</tr>
<tr>
<td>16</td>
<td>126</td>
<td>18</td>
<td>74</td>
</tr>
<tr>
<td>16</td>
<td>127</td>
<td>19</td>
<td>75</td>
</tr>
<tr>
<td>16</td>
<td>128</td>
<td>20</td>
<td>76</td>
</tr>
<tr>
<td>16</td>
<td>129</td>
<td>21</td>
<td>77</td>
</tr>
<tr>
<td>16</td>
<td>130</td>
<td>22</td>
<td>78</td>
</tr>
<tr>
<td>16</td>
<td>131</td>
<td>23</td>
<td>79</td>
</tr>
<tr>
<td>17</td>
<td>132</td>
<td>48</td>
<td>48</td>
</tr>
<tr>
<td>17</td>
<td>133</td>
<td>49</td>
<td>49</td>
</tr>
<tr>
<td>17</td>
<td>134</td>
<td>50</td>
<td>50</td>
</tr>
<tr>
<td>17</td>
<td>135</td>
<td>51</td>
<td>51</td>
</tr>
<tr>
<td>17</td>
<td>136</td>
<td>52</td>
<td>52</td>
</tr>
<tr>
<td>17</td>
<td>137</td>
<td>53</td>
<td>53</td>
</tr>
<tr>
<td>17</td>
<td>138</td>
<td>54</td>
<td>54</td>
</tr>
<tr>
<td>17</td>
<td>139</td>
<td>55</td>
<td>55</td>
</tr>
<tr>
<td>17</td>
<td>140</td>
<td>56</td>
<td>56</td>
</tr>
<tr>
<td>18</td>
<td>141</td>
<td>57</td>
<td>57</td>
</tr>
<tr>
<td>18</td>
<td>142</td>
<td>58</td>
<td>58</td>
</tr>
<tr>
<td>18</td>
<td>143</td>
<td>59</td>
<td>59</td>
</tr>
<tr>
<td>18</td>
<td>144</td>
<td>60</td>
<td>60</td>
</tr>
<tr>
<td>18</td>
<td>145</td>
<td>61</td>
<td>61</td>
</tr>
<tr>
<td>18</td>
<td>146</td>
<td>62</td>
<td>62</td>
</tr>
<tr>
<td>18</td>
<td>147</td>
<td>63</td>
<td>63</td>
</tr>
<tr>
<td>19</td>
<td>148</td>
<td>32</td>
<td>32</td>
</tr>
<tr>
<td>19</td>
<td>149</td>
<td>33</td>
<td>33</td>
</tr>
<tr>
<td>19</td>
<td>150</td>
<td>34</td>
<td>34</td>
</tr>
<tr>
<td>19</td>
<td>151</td>
<td>35</td>
<td>35</td>
</tr>
<tr>
<td>19</td>
<td>152</td>
<td>36</td>
<td>36</td>
</tr>
<tr>
<td>19</td>
<td>153</td>
<td>37</td>
<td>37</td>
</tr>
<tr>
<td>19</td>
<td>154</td>
<td>38</td>
<td>38</td>
</tr>
<tr>
<td>19</td>
<td>155</td>
<td>39</td>
<td>39</td>
</tr>
<tr>
<td>20</td>
<td>156</td>
<td>40</td>
<td>40</td>
</tr>
<tr>
<td>20</td>
<td>157</td>
<td>41</td>
<td>41</td>
</tr>
<tr>
<td>20</td>
<td>158</td>
<td>42</td>
<td>42</td>
</tr>
<tr>
<td>20</td>
<td>159</td>
<td>43</td>
<td>43</td>
</tr>
<tr>
<td>20</td>
<td>160</td>
<td>44</td>
<td>44</td>
</tr>
<tr>
<td>20</td>
<td>161</td>
<td>45</td>
<td>45</td>
</tr>
<tr>
<td>20</td>
<td>162</td>
<td>46</td>
<td>46</td>
</tr>
<tr>
<td>20</td>
<td>163</td>
<td>47</td>
<td>47</td>
</tr>
<tr>
<td>21</td>
<td>164</td>
<td>64</td>
<td>64</td>
</tr>
<tr>
<td>21</td>
<td>165</td>
<td>65</td>
<td>65</td>
</tr>
<tr>
<td>21</td>
<td>166</td>
<td>66</td>
<td>66</td>
</tr>
<tr>
<td>Ext</td>
<td>67</td>
<td>67 Ext self inhibition</td>
<td></td>
</tr>
<tr>
<td>------</td>
<td>------</td>
<td>------------------------</td>
<td></td>
</tr>
<tr>
<td>68</td>
<td>68</td>
<td>68 Ext self inhibition</td>
<td></td>
</tr>
<tr>
<td>69</td>
<td>69</td>
<td>69 Ext self inhibition</td>
<td></td>
</tr>
<tr>
<td>70</td>
<td>70</td>
<td>70 Ext self inhibition</td>
<td></td>
</tr>
<tr>
<td>71</td>
<td>71</td>
<td>71 Ext self inhibition</td>
<td></td>
</tr>
<tr>
<td>72</td>
<td>72</td>
<td>72 Ext self inhibition</td>
<td></td>
</tr>
<tr>
<td>73</td>
<td>73</td>
<td>73 Flex self inhibition</td>
<td></td>
</tr>
<tr>
<td>74</td>
<td>74</td>
<td>74 Flex self inhibition</td>
<td></td>
</tr>
<tr>
<td>75</td>
<td>75</td>
<td>75 Flex self inhibition</td>
<td></td>
</tr>
<tr>
<td>76</td>
<td>76</td>
<td>76 Flex self inhibition</td>
<td></td>
</tr>
<tr>
<td>77</td>
<td>77</td>
<td>77 Flex self inhibition</td>
<td></td>
</tr>
<tr>
<td>78</td>
<td>78</td>
<td>78 Flex self inhibition</td>
<td></td>
</tr>
<tr>
<td>79</td>
<td>79</td>
<td>79 Flex self inhibition</td>
<td></td>
</tr>
<tr>
<td>SPIKESARRAY</td>
<td>Spike</td>
<td>Leg</td>
<td>MN</td>
</tr>
<tr>
<td>------------</td>
<td>-------</td>
<td>-----</td>
<td>--------</td>
</tr>
<tr>
<td>0</td>
<td>32</td>
<td>1</td>
<td>Pro</td>
</tr>
<tr>
<td>1</td>
<td>40</td>
<td>1</td>
<td>Ret</td>
</tr>
<tr>
<td>2</td>
<td>48</td>
<td>1</td>
<td>Elev</td>
</tr>
<tr>
<td>3</td>
<td>56</td>
<td>1</td>
<td>Dep</td>
</tr>
<tr>
<td>4</td>
<td>64</td>
<td>1</td>
<td>Ext</td>
</tr>
<tr>
<td>5</td>
<td>72</td>
<td>1</td>
<td>Flex</td>
</tr>
<tr>
<td>6</td>
<td>33</td>
<td>2</td>
<td>Pro</td>
</tr>
<tr>
<td>7</td>
<td>41</td>
<td>2</td>
<td>Ret</td>
</tr>
<tr>
<td>8</td>
<td>49</td>
<td>2</td>
<td>Elev</td>
</tr>
<tr>
<td>9</td>
<td>57</td>
<td>2</td>
<td>Dep</td>
</tr>
<tr>
<td>10</td>
<td>65</td>
<td>2</td>
<td>Ext</td>
</tr>
<tr>
<td>11</td>
<td>73</td>
<td>2</td>
<td>Flex</td>
</tr>
<tr>
<td>12</td>
<td>34</td>
<td>3</td>
<td>Pro</td>
</tr>
<tr>
<td>13</td>
<td>42</td>
<td>3</td>
<td>Ret</td>
</tr>
<tr>
<td>14</td>
<td>50</td>
<td>3</td>
<td>Elev</td>
</tr>
<tr>
<td>15</td>
<td>58</td>
<td>3</td>
<td>Dep</td>
</tr>
<tr>
<td>16</td>
<td>66</td>
<td>3</td>
<td>Ext</td>
</tr>
<tr>
<td>17</td>
<td>74</td>
<td>3</td>
<td>Flex</td>
</tr>
<tr>
<td>18</td>
<td>35</td>
<td>4</td>
<td>Pro</td>
</tr>
<tr>
<td>19</td>
<td>43</td>
<td>4</td>
<td>Ret</td>
</tr>
<tr>
<td>20</td>
<td>51</td>
<td>4</td>
<td>Elev</td>
</tr>
<tr>
<td>21</td>
<td>59</td>
<td>4</td>
<td>Dep</td>
</tr>
<tr>
<td>22</td>
<td>67</td>
<td>4</td>
<td>Ext</td>
</tr>
<tr>
<td>23</td>
<td>75</td>
<td>4</td>
<td>Flex</td>
</tr>
<tr>
<td>24</td>
<td>36</td>
<td>5</td>
<td>Pro</td>
</tr>
<tr>
<td>25</td>
<td>44</td>
<td>5</td>
<td>Ret</td>
</tr>
<tr>
<td>26</td>
<td>52</td>
<td>5</td>
<td>Elev</td>
</tr>
<tr>
<td>27</td>
<td>60</td>
<td>5</td>
<td>Dep</td>
</tr>
<tr>
<td>28</td>
<td>68</td>
<td>5</td>
<td>Ext</td>
</tr>
<tr>
<td>29</td>
<td>76</td>
<td>5</td>
<td>Flex</td>
</tr>
<tr>
<td>30</td>
<td>37</td>
<td>6</td>
<td>Pro</td>
</tr>
<tr>
<td>31</td>
<td>45</td>
<td>6</td>
<td>Ret</td>
</tr>
<tr>
<td>32</td>
<td>53</td>
<td>6</td>
<td>Elev</td>
</tr>
<tr>
<td>33</td>
<td>61</td>
<td>6</td>
<td>Dep</td>
</tr>
<tr>
<td>34</td>
<td>69</td>
<td>6</td>
<td>Ext</td>
</tr>
<tr>
<td>35</td>
<td>77</td>
<td>6</td>
<td>Flex</td>
</tr>
<tr>
<td>36</td>
<td>38</td>
<td>7</td>
<td>Pro</td>
</tr>
<tr>
<td>37</td>
<td>46</td>
<td>7</td>
<td>Ret</td>
</tr>
<tr>
<td>38</td>
<td>54</td>
<td>7</td>
<td>Elev</td>
</tr>
<tr>
<td>39</td>
<td>62</td>
<td>7</td>
<td>Dep</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>---</td>
<td>---</td>
<td>---</td>
<td>---</td>
</tr>
<tr>
<td>40</td>
<td>70</td>
<td>7 Ext</td>
<td></td>
</tr>
<tr>
<td>41</td>
<td>78</td>
<td>7 Flex</td>
<td></td>
</tr>
<tr>
<td>42</td>
<td>39</td>
<td>8 Pro</td>
<td></td>
</tr>
<tr>
<td>43</td>
<td>47</td>
<td>8 Ret</td>
<td></td>
</tr>
<tr>
<td>44</td>
<td>55</td>
<td>8 Elev</td>
<td></td>
</tr>
<tr>
<td>45</td>
<td>63</td>
<td>8 Dep</td>
<td></td>
</tr>
<tr>
<td>46</td>
<td>71</td>
<td>8 Ext</td>
<td></td>
</tr>
<tr>
<td>47</td>
<td>79</td>
<td>8 Flex</td>
<td></td>
</tr>
<tr>
<td>Board 1</td>
<td>Pin</td>
<td>Neuron</td>
<td>SpikesArray</td>
</tr>
<tr>
<td>---------</td>
<td>-----</td>
<td>--------</td>
<td>-------------</td>
</tr>
<tr>
<td>Leg 1 Left</td>
<td>Protractor</td>
<td>4</td>
<td>32</td>
</tr>
<tr>
<td></td>
<td>Retractor</td>
<td>5</td>
<td>40</td>
</tr>
<tr>
<td></td>
<td>Elevator</td>
<td>3</td>
<td>48</td>
</tr>
<tr>
<td></td>
<td>Depressor</td>
<td>2</td>
<td>56</td>
</tr>
<tr>
<td></td>
<td>Extensor</td>
<td>1</td>
<td>64</td>
</tr>
<tr>
<td></td>
<td>Flexor</td>
<td>0</td>
<td>72</td>
</tr>
<tr>
<td>Leg 1 Right</td>
<td>Protractor</td>
<td>9</td>
<td>33</td>
</tr>
<tr>
<td></td>
<td>Retractor</td>
<td>10</td>
<td>41</td>
</tr>
<tr>
<td></td>
<td>Elevator</td>
<td>11</td>
<td>49</td>
</tr>
<tr>
<td></td>
<td>Depressor</td>
<td>7</td>
<td>57</td>
</tr>
<tr>
<td></td>
<td>Extensor</td>
<td>6</td>
<td>65</td>
</tr>
<tr>
<td></td>
<td>Flexor</td>
<td>8</td>
<td>73</td>
</tr>
<tr>
<td>Board 2</td>
<td>Protractor</td>
<td>16</td>
<td>34</td>
</tr>
<tr>
<td>Leg 2 Left</td>
<td>Retractor</td>
<td>17</td>
<td>42</td>
</tr>
<tr>
<td></td>
<td>Elevator</td>
<td>15</td>
<td>50</td>
</tr>
<tr>
<td></td>
<td>Depressor</td>
<td>14</td>
<td>58</td>
</tr>
<tr>
<td></td>
<td>Extensor</td>
<td>13</td>
<td>66</td>
</tr>
<tr>
<td></td>
<td>Flexor</td>
<td>12</td>
<td>74</td>
</tr>
<tr>
<td>Leg 2 Right</td>
<td>Protractor</td>
<td>22</td>
<td>35</td>
</tr>
<tr>
<td></td>
<td>Retractor</td>
<td>21</td>
<td>43</td>
</tr>
<tr>
<td></td>
<td>Elevator</td>
<td>19</td>
<td>51</td>
</tr>
<tr>
<td></td>
<td>Depressor</td>
<td>23</td>
<td>59</td>
</tr>
<tr>
<td></td>
<td>Extensor</td>
<td>20</td>
<td>67</td>
</tr>
<tr>
<td></td>
<td>Flexor</td>
<td>18</td>
<td>75</td>
</tr>
<tr>
<td>Board 3</td>
<td>Protractor</td>
<td>28</td>
<td>36</td>
</tr>
<tr>
<td>Leg 3 Left</td>
<td>Retractor</td>
<td>29</td>
<td>44</td>
</tr>
<tr>
<td></td>
<td>Elevator</td>
<td>26</td>
<td>52</td>
</tr>
<tr>
<td></td>
<td>Depressor</td>
<td>27</td>
<td>60</td>
</tr>
<tr>
<td></td>
<td>Extensor</td>
<td>25</td>
<td>68</td>
</tr>
<tr>
<td></td>
<td>Flexor</td>
<td>24</td>
<td>76</td>
</tr>
<tr>
<td>Leg 3 Right</td>
<td>Protractor</td>
<td>35</td>
<td>37</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>--------</td>
<td>--------</td>
<td>--------</td>
<td>--------</td>
</tr>
<tr>
<td>Retractor</td>
<td>34</td>
<td>45</td>
<td>31</td>
</tr>
<tr>
<td>Elevator</td>
<td>32</td>
<td>53</td>
<td>32</td>
</tr>
<tr>
<td>Depressor</td>
<td>33</td>
<td>61</td>
<td>33</td>
</tr>
<tr>
<td>Extensor</td>
<td>31</td>
<td>69</td>
<td>34</td>
</tr>
<tr>
<td>Flexor</td>
<td>30</td>
<td>77</td>
<td>35</td>
</tr>
</tbody>
</table>

### Board 4

#### Leg 4 Left

<p>| | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Protractor</td>
<td>40</td>
<td>38</td>
<td>36</td>
</tr>
<tr>
<td>Retractor</td>
<td>41</td>
<td>46</td>
<td>37</td>
</tr>
<tr>
<td>Elevator</td>
<td>39</td>
<td>54</td>
<td>38</td>
</tr>
<tr>
<td>Depressor</td>
<td>38</td>
<td>62</td>
<td>39</td>
</tr>
<tr>
<td>Extensor</td>
<td>37</td>
<td>70</td>
<td>40</td>
</tr>
<tr>
<td>Flexor</td>
<td>36</td>
<td>78</td>
<td>41</td>
</tr>
</tbody>
</table>

#### Leg 4 Right

<p>| | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Protractor</td>
<td>48</td>
<td>39</td>
<td>42</td>
</tr>
<tr>
<td>Retractor</td>
<td>49</td>
<td>47</td>
<td>43</td>
</tr>
<tr>
<td>Elevator</td>
<td>47</td>
<td>55</td>
<td>44</td>
</tr>
<tr>
<td>Depressor</td>
<td>46</td>
<td>63</td>
<td>45</td>
</tr>
<tr>
<td>Extensor</td>
<td>45</td>
<td>71</td>
<td>46</td>
</tr>
<tr>
<td>Flexor</td>
<td>42</td>
<td>79</td>
<td>47</td>
</tr>
</tbody>
</table>
Appendix F

DSP programming

-Instructions written by A. Mark Hunt
**Installation**

Copy the folder “NEU” to “C:”.

**Notes**

The above installation copies all of the components needed to generate code for the TI TMS320C6727 DSP processor. Code may be generated for subsequent writing to the I2C boot flash (see **boot2** below for an example make file). Code may also be generated in Motorola “S-record” format for loading via a “secondary” boot loader (see *Installing the secondary boot loader* below). Two examples of make files that generate “S-records” (**hello** and **daniel_test**) are given below.

**boot2**

This is the make file that is used to generate the secondary boot loader. This make file illustrates the use of AIS and the steps necessary to convert the code into an image file (“.img”) that can be written into the I2C boot flash.

```
# Lines beginning with # are comments.
# Targets begin at the left margin followed by a ':' (colon).
# Shell command lines must begin with a tab.
#
# Dependencies are formed as:
#
#    thisDependsOn: this andThis andEvenThis
#
# and are followed by command lines to construct "thisDependsOn".
# "thisDependsOn:" starts in column 1.
#
# SHORTHAND:
#    $@ stands for the full name of the target.
#    $* stands for the name of the target with the suffix deleted.
#    $? evaluates to a list of prerequisites newer than the current target
#    $< evaluates to whatever prerequisite triggered the rule
#
MCU = 6700
#
# DECENDING STACK

CCStudio = /NEU/C6000Tools
This is the path to the 6727 compiler tools.

del = rm

AR = $(CCStudio)/bin/ar6x
AS = $(CCStudio)/bin/asm6x
CC = $(CCStudio)/bin/cl6x
LD = $(CCStudio)/bin/lnk6x
HEX = $(CCStudio)/bin/hex6x
LIBS = $(CCStudio)/lib
```
CFLAGS = -q -mv $(MCU) -i $(CCStudio)/include
#ASFLAGS = -mmcu=$(MCU) -I. -x assembler-with-cpp
#LDFLAGS = -stack4096 -c -w -x $(CCStudio)/lib
LDFLAGS = -stack4096 -c -w -x

boot2: main.obj vectors.obj cfg.obj uart.obj loader.obj fat16.obj ocrwi.obj\  
  strings.obj sd.obj spi.obj

_These are the object files required by the projects._

$(LD) $(LDFLAGS) boot2.cmd -lc:/NEU/tiDspUtils/bin/rts6700.lib -m boot2.map\ 
  main.obj vectors.obj cfg.obj uart.obj loader.obj fat16.obj\ 
  ocrwi.obj strings.obj sd.obj spi.obj -o boot2.out

_This is the linker command that links the object files with each other and the library._

perl ../tiDspUtils/bin/genais.pl -i boot2.out -bootmode i2cmaster -otype ascii -cfgtype "c"

_This invokes the perl script (genais) that converts boot2.out into AIS format._

aisToBin boot2

_This converts the AIS object into a binary image that can be written into the I2C flash._

clean:
  - del *.obj
  - del *.~*
  - del *.lst
  - del *.img
  - del *.map

_This removes all compiler and linker generated files from the project. This should be used to force the entire project to be rebuilt._

_The following are the rules that convert source code into object code._

main.obj: main.c
  $(CC) $(CFLAGS) main.c

uart.obj: uart.c uart.h
  $(CC) $(CFLAGS) uart.c

loader.obj: loader.c loader.h
  $(CC) $(CFLAGS) loader.c

clocks.obj: clocks.c clocks.h
  $(CC) $(CFLAGS) clocks.c

cfg.obj: cfg.c
  $(CC) $(CFLAGS) cfg.c

gpio.obj: gpio.c
  $(CC) $(CFLAGS) gpio.c

strings.obj: strings.c strings.h
  $(CC) $(CFLAGS) strings.c

fat16.obj: fat16.c fat16.h uart.h
  $(CC) $(CFLAGS) fat16.c
This make file illustrates how to compile and link a simple project into an S-record file that can be loaded and run using the secondary boot loader.

This is the path to the 6727 compiler tools.
daniel_test

This is another (somewhat more complicated) make file that illustrates how to compile and link a project into an S-record file that can be loaded and run using the secondary boot loader.
AS = $(CCStudio)/bin/asm6x
CC = $(CCStudio)/bin/cl6x
LD = $(CCStudio)/bin/lnk6x
HEX = $(CCStudio)/bin/hex6x

LIBS = $(CCStudio)/lib

CFLAGS = -q -mv $(MCU) -i $(CCStudio)/include
#ASFLAGS = -mmcu=$(MCU) -I. -x assembler-with-cpp
#LDFLAGS = -stack4096 -c -w -x -I $(CCStudio)/lib
LDFLAGS = -stack4096 -c -w -x --ram_model
tb: tb.obj vectors.obj uart.obj clocks.obj driveSpikes.obj
   pc2dsp.obj spiADIS.obj adis16201.obj
$(LD) $(LDFLAGS) tb.cmd -lcr/NEU/tiDspUtils/bin/rts6700.lib -m tb.map tb.obj\  
   vectors.obj uart.obj clocks.obj driveSpikes.obj\  
   pc2dsp.obj spiADIS.obj adis16201.obj -o tb.out
# ---- creates tb.m0 -- note: genais & aisToBin are not needed for this
$(HEX) -I tb.out --motorola --memwidth 8

clean:
  - del *.obj
  - del *.~*
  - del *.lst
  - del *.img

tb.obj: tb.c lbDspIO.h memMap.h pl1.h rti.h uart.h uhpi.h clocks.h
$(CC) $(CFLAGS) tb.c

uart.obj: uart.c uart.h
$(CC) $(CFLAGS) uart.c

clocks.obj: clocks.c clocks.h
$(CC) $(CFLAGS) clocks.c

pc2dsp.obj: pc2dsp.c
$(CC) $(CFLAGS) pc2dsp.c

driveSpikes.obj: driveSpikes.c
$(CC) $(CFLAGS) driveSpikes.c

spiADIS.obj: spiADIS.c spiADIS.h
$(CC) $(CFLAGS) spiADIS.c

adis16201.obj: adis16201.c adis16201.h i2cBHDR.h
$(CC) $(CFLAGS) adis16201.c

vectors.obj: vectors.asm
$(CC) $(CFLAGS) vectors.asm

**Installing the secondary boot loader**

Hook up the I2C programming board to the PC and start “sfpgmr” in a command window. Power up the “brain board” and the programming board. Insert the programming cable into the J7 I2C programming port. Use the “c” command and the “s” commands to connect the PC to the programming
board and to send the secondary boot loader image to the “brain board” I2C flash chip. Example commands:

```shell
>sfpgmr -p com6:
csboot2.img
```

The sfpgmr program will copy boot2.img to the I2C flash and then verify that the copy was successful. Once the copy is complete the uSD secondary boot loader can be used to load Motorola S-record executable files into the “brain board”.

**Using the uSD secondary boot loader (boot2)**

The secondary boot loader is loaded into RAM by the primary I2C boot loader. Once loaded and given control, it searches the root directory for a file with the suffix “.m0” (“dot em zero”). This file is expected to be an executable image in Motorola S-record format. If no such file is found, the boot loader prompts for a file name to load and run. This file name is not required to have a “.m0” suffix but it must still be an executable image in Motorola S-record format.

**Steps necessary before invoking make**

**Windows XP**

Open a command window.

- Start
- Run…
- In the Open: input box enter: **command**
- Select OK -- a command window will appear
- Using explorer navigate to the desired project folder
- In the address box copy the
- In the command window type: **cd**
  (Be sure to enter a space after the **cd**!)
- Right click the mouse on the menu bar at the top of the command window
- Select edit
- Select paste
- Type **return**
  This will move the command window to the desired project folder
- Type **addpath**
  This will create the paths necessary for the make files to operate correctly.

**Windows 7**

Open a command window.

- Select **Start**
- Select **Command Prompt**
- Using explorer navigate to the desired project folder
In the address box copy the path
In the command window type: **cd**
(Be sure to enter a space after the **cd**!
Right click the mouse in the command window
Select paste
Type **return**
This will move the command window to the desired project folder
Type **addpath**
This will create the paths necessary for the make files to operate correctly.
Appendix G

MATLAB script

-Neuron spike parsing protocol
Appendix H
MATLAB script to parse and plot ATMega data transmitted on serial line.

```matlab
1 - clear
2 - clc
3 -
4 - %import data
5 - Data = importdata('antennaGood1.txt');
6 -
7 - %get length of data
8 - len = length(Data);
9 -
10 - %set target heading equal to zero
11 - Direction(len,1) = 0;
12 -
13 - %assign each column of the data file to its appropriate variable
14 - Xaccel = Data(1:len,1);
15 - Yaccel = Data(1:len,2);
16 - Zaccel = Data(1:len,3);
17 - OFright = Data(1:len,4);
18 - OFleft = Data(1:len,5);
19 - AntRm = Data(1:len,6);
20 - AntRL = Data(1:len,7);
21 - AntLR = Data(1:len,8);
22 - AntLM = Data(1:len,9);
23 - fCurrent = Data(1:len,10);
24 - senses = Data(1:len,11);
25 - compass = Data(1:len,12);
26 - bump = Data(1:len,13);
27 - bitshift_sensor = Data(1:len,15);
28 - bitshift_neuron = Data(1:len,16);
29 -
30 - OFhiL = Data(1:len,23);
31 - OFhiR = Data(1:len,24);
32 -
33 - %calibrate compass data
34 -
35 - for n=1:len
36 -   if compass(n)<180
37 -     compassCalib(n,1) = compass(n) + 360;
38 -     else
39 -     compassCalib(n,1) = compass(n);
40 - end
41 -
42 - %parse spike data from bits of 'bitshift_sensor'
43 - spikeSense = dec2bin(bitshift_sensor,13);
44 - spike13 = str2num(spikeSense(:,1));
45 - spike14 = str2num(spikeSense(:,2));
46 - spike15 = str2num(spikeSense(:,3));
47 - spike16 = str2num(spikeSense(:,4));
48 - spike17 = str2num(spikeSense(:,5));
49 - spike18 = str2num(spikeSense(:,6));
50 - spike19 = str2num(spikeSense(:,7));
51 - spike20 = str2num(spikeSense(:,8));
52 -
53 - %parse spike data from bits of 'bitshift_neuron'
54 - spikeBrain = dec2bin(bitshift_neuron,13);
55 - spike0 = str2num(spikeBrain(:,1));
56 - spike9 = str2num(spikeBrain(:,2));
57 - spike10 = str2num(spikeBrain(:,3));
58 - spike11 = str2num(spikeBrain(:,4));
59 - spike12 = str2num(spikeBrain(:,5));
```
spike5 = str2num(spikeBrain(:,6));
spike6 = str2num(spikeBrain(:,7));
spike7 = str2num(spikeBrain(:,8));
spike8 = str2num(spikeBrain(:,9));
spike1 = str2num(spikeBrain(:,10));
spike2 = str2num(spikeBrain(:,11));
spike3 = str2num(spikeBrain(:,12));
spike4 = str2num(spikeBrain(:,13));

%choose data range to plot
start_data = 0;
end_data = len;

% Create figure
figure1 = figure;

% Create subplots for each variable to plot
% Create axes
axes1 = axes('Parent',figure1,'OuterPosition',[0 0.8689 1 0.1111]);
% Uncomment the following line to preserve the X-limits of the axes
xlim(axes1,[start_data end_data]);
% Uncomment the following line to preserve the Y-limits of the axes
ylim(axes1,[min(Xaccel)-10 max(Xaccel)+10]);
box(axes1,'on');
hold(axes1,'all');
% Create multiple lines using matrix input to plot
plot(Xaccel,'Parent',axes1,'DisplayName','X Accel');
ylabel('x accel');

% Create axes
axes2 = axes('Parent',figure1,'OuterPosition',[0 0.7778 1 0.1111]);
% Uncomment the following line to preserve the X-limits of the axes
xlim(axes2,[start_data end_data]);
% Uncomment the following line to preserve the Y-limits of the axes
ylim(axes2,[min(Yaccel)-10 max(Yaccel)+10]);
box(axes2,'on');
hold(axes2,'all');
% Create plot
plot(Yaccel,'Parent',axes2,'DisplayName','Y Accel');
ylabel('y accel');

%add plots and adjust spacing depending on number of subplots