Abstract

CRAVER, MATTHEW DAVID. Mobile Robot Homing Control Based on Odor Sensing. (Under the direction of Edward Grant.)

As robotic systems transition fully out of the laboratory environment and into the real world, they will need to be more robust and autonomous in order to deal with different environmental conditions and increasing complex tasks. Many developers have attempted to solve this problem by using increasingly more sensors and computational resources to provide the control system with more information. Others have tried to more explicitly define the environment and task being performed. However, there are many relatively simple biological organisms that exhibit complex behaviors using limited resources. This dissertation reports on the use of a biologically-based solution to develop a robotic platform and control system that allow for emergent intelligence and robustness.

A new robotic platform, the EvBot III, was developed with ubiquitous modularity in software, hardware, and control systems as a goal. The EvBot III is comprised of (1) a differential drive base with an attached turret and sensor shield, (2) a StackableUSB™ single board PC-104 computer, (3) a general purpose data acquisition system (CRIM-Daq), (4) a modular control architecture [1], and (5) a 3D physics-enabled simulation environment [2]. The driving portion of the base is designed such that different drive systems (legged system, ackerman drive, etc.) can be used without needing to change the main controller. Additionally, the sensor shield allows different configurations and sensing modalities to be switched out based upon the desired application. Therefore, the EvBot III is expected to decrease development time and accelerate the progress of robotic and computational intelligence research.

A flat, homogeneous sensorimotor control architecture was developed for the EvBot III. This network was developed in LabVIEW and run on a Windows 7 (64-bit) PC. Chemical sensing was selected as a test application because it is the most widespread sensing modality among living organisms. Here, six MQ-3 were used to sense the chemical signal that marked the EvBot III charging location. A simple alcohol homing algorithm was developed that normalized for sensor variation. This homing algorithm was used during training for the sensorimotor network, and produced a zigzag navigational strategy that is similar to ones found in nature. The sensorimotor network was developed to fully connect all sensorimotor elements, which included olfactory, power, and motor modalities. The sensorimotor experiments were conducted using various experimental methodologies to build the correct correlations between increased alcohol concentration and increased charge. Although the sensorimotor network developed correlations using this approach, it did not build sufficient correlation relating increased alcohol concentration with charging. Future work includes revisiting the experimental methodology employed, using more accurate alcohol sensors, and incorporating sensing modalities with a similar layout and response to the alcohol sensors.

Mobile Robot Homing Control Based on Odor Sensing

by
Matthew David Craver

A dissertation submitted to the Graduate Faculty of
North Carolina State University
in partial fulfillment of the
requirements for the Degree of
Doctor of Philosophy

Computer Engineering

Raleigh, North Carolina

2014

APPROVED BY:

Troy Nagle	Coby Schal
Wesley Snyder	Edward Grant Chair of Advisory Committee

ProQuest LLC.
789 East Eisenhower Parkway
P.O. Box 1346
Ann Arbor, MI 48106 - 1346

Dedication

To Mom, Dad, Rebecca, and Meghan.

Biography

Matthew David Craver was born on December 3, 1977 in Winston-Salem, North Carolina to Steve Emanuel Craver and Jennifer Kessler Craver. He received his Bachelor of Science in Computer Engineering from North Carolina State University, Raleigh, North Carolina in 2001. He received his Master of Science in Computer Engineering from North Carolina State University, Raleigh, North Carolina in 2003.

Acknowledgements

I would first like to thank my family for always being there to support me. They have always encouraged my curiosity and questioning, and have given me the strength to excel in everything I do.

I would need to express my sincere gratitude to my advisor Dr. Edward Grant for bringing me into the Center for Robotics and Intelligent Machines (CRIM) and continually supporting me and showing confidence in my ability. Additionally, I would like to acknowledge the other members of my committee– Dr. Colby Schal, Dr. Wesley Snyder, and Dr. Troy Nagle–for their assistance.

I also need to thank the many members of the CRIM who have mentored and helped me throughout my graduate career. Dr. Kyle Luthy, Brooks Adcock, Frederick Livingston, Dr. Leonardo Mattos, Dr. Nikhil Deshpande, Micah Colon, Jim Ashcraft, Zack Nienstedt, Mark Draelos, and Chris Bollinger have served as a sounding board and provided valuable input to this work.

Additionally I would like to thank Zane Purvis and Ben Stiles for providing assistance with code and design issues.

And last, but certainly not least, I need to thank Dr. Meghan Hegarty-Craver for her continual support and encouragement.

Table of Contents

List of Tables

List of Figures

Chapter 1

Motivation for Research

Mobile robots have shown great promise for aiding humans in dangerous situations such as search and rescue at disaster sites [4–7], industrial applications [8, 9], security monitoring [10, 11], and for performing tasks remotely in a natural environment [12, 13]. While these applications are dynamic in nature, many robotic systems require a constrained environment [14]. Due to the lack of hard constraints in real-world applications, it can be difficult to plan and program such systems for all eventualities.

In order to provide robots with more information so that appropriate control decisions can be made, many groups have increased the number of sensors on their mobile platforms [15–17]. For example, the winner of the DARPA Grand Challenge used 5 SICK laser range finders, 1 color camera, 2 24GHz RADAR sensors, a GPS positioning system, a GPS compass, and a 6-axis IMU to navigate an off-road course [15]. While the technology that has come out of the DARPA Grand Challenge has shown significant leaps in autonomous vehicle navigation [15–17], its extreme cost and complexity limits its use [18, 19]. Additionally, it is impractical to use large numbers of sensors on mobile robots due to the significant processing demands. For example, the vehicle that won the DARPA Grand Challenge required 6 Pentium M computers (2 executed the race software, 1 performed vision processing, 1 logged the race data, and 2 were idle) [15]. The power demands of these sensors also render them infeasible for use on small mobile platforms relying on limited battery power.

Other groups have attempted to constrain the robotic system by explicitly choosing the base behaviors that the robot will need to perform a task in a given environment [3, 20–23]. This can lead to a system that is overly constrained by experimenter bias [3]. Problems can arise if all eventualities are not taken into account. For example, obstacle avoidance is often presumed to be a base behavior in robotics. Conversely, it has been shown that obstacle avoidance can be considered to be an emergent or derivative behavior of homing [24, 25].

The goal of the research reported on in this dissertation was to develop a mobile robot platform (EvBot III) that could operate robustly in a dynamic environment by limiting sensor complexity and experimenter bias. The EvBot III (Evolutionary Robot) improves upon previous versions of the EvBot

platform [26, 27], which suffered from limited battery life, drive train slippage, etc. The new EvBot III also has enhanced modularity and scalability. In order to reduce sensor complexity, only chemical sensors were used. Olfaction was selected over other modalities (vision, touch, etc.) because the response to chemical stimuli was the first sense developed by primordial life [28]. Additionally, although olfaction is the most widespread sensory modality found in nature, it is the least often implemented in robotic systems despite its potential for widespread application in the field of robotics [28]. Robotic olfaction can be used to find chemical leaks, explosives, or disaster victims [28]. In order to reduce experimenter bias and "provide stable even if non-optimal solutions in the face of uncertainty, noise or incomplete input, or unpredictable changes in context [3]", the main robotic controller was implemented using a sensorimotor integration architecture. This sensorimotor architecture is a flat, homogeneous architecture that fully connects all sensor and motor elements without internal distinction between the two. Throughout the training of the sensorimotor network, correlations in activities are built between all nodes. After training, these correlations are used to drive the robot. In order to test the EvBot III's performance, the robot was evaluated based upon its ability to autonomously navigate up an alcohol plume to a charging station.

This dissertation focuses on the use of odor sensing as a strategy for navigation and homing for mobile robots in a dynamic environment. In order to aid in the understanding of mobile robots, an overview of previous methods and architectures for robotic sensor fusion is provided in Chapter 2. The re-designed EvBot III, including the hardware and software used to implement the controller based on olfactory sensing, is outlined in Chapter 3. This chapter also describes the CRIM-Daq, which is an extensible data acquisition solution. Chapter 4 reviews different odor sensing strategies that are used by mobile robots, as well as the platforms used to implement them. The challenges and limitations associated with olfactory sensor systems are discussed, and the algorithm and testbed used in this research are described and evaluated. The sensorimotor architecture implemented on the EvBot III is outlined in Chapter 5. This chapter also compares the performance of the trained sensorimotor network to other learned controllers that were developed from the training data. The final chapter summarizes the research and findings reported on in this dissertation, as well as conclusions, suggestions for improvement, and future work.

Chapter 2

An Overview of Sensor Fusion Methods and Architectures

2.1 Abstract

Robotic systems are currently being used to perform increasingly complex tasks. This has resulted in an abundance of data that is available to the control system, and consequently a growing need for methods to pick out and combine the most appropriate data. Sensor fusion methods are a common method used for fusing the output of sensors with different modalities. This reduces the amount of data that the control system needs to process. There are also sensor fusion architectures that incorporate mission management functions to help the system intelligently use the fused data. Additionally, biologically-inspired sensorimotor networks offer the ability to autonomously fuse the various sensor data and apply the fused data to generate emergent behavior. Given the complexity and variety of sensors in the natural realm, these biologically inspired sensorimotor networks are poised to truly launch autonomous systems out of the laboratory and into the real world. This chapter provides an overview of the various methods used for reducing the amount and complexity data.

2.2 Introduction

Current robotic systems require more and increasingly detailed information about their environment in order to perform a given task(s). As applications need faster and more accurate responses, more precise measurement, understanding, and situational awareness are needed for control [15–18,29]. Systems now include heterogeneous and multiple homogeneous sensors for environmental monitoring and mapping. For example, the top three vehicles in the DARPA Grand Challenge used a large number of highly specialized sensors to navigate an off-road course (Table 2.1) [15–17]. Furthermore, as more sensor data becomes available to the system, decisions must be made concerning how to analyze and interpret

Table 2.1: DARPA Grand Challenge Winners [15–17]

Team	Sensors	Computation
1st Place Stanford Racing Team, Stanley (Stanford University)	5 SICK laser range finders, 1 color camera, 2 24GHz RADAR sensors, GPS positioning system, GPS Compass, 6-axis IMU	6 Pentium M computers (2 executed race software, 1 performed vision processing, 1 logged race data, and 2 were idle
2nd Place Red Team, Sandstorm (Carnegie Mellon Univ.)	1 Riegal Q140i scanning laser range finder, 3 SICK LMS laser scanners, 1 high speed stereo vision system, 1 NavTech DS2000 Continuous Wave Frequency Modulated radar, 1 Applaniz POS-LV pose estimation system (Inertia measurement, GPS, odometry)	1 quad processor Itanium II, 3 dual processor Xeon, 4 Pentium III class PC-104s
3rd Place Red Team, H1ghlander (Carnegie Mellon Univ.)	1 Riegal Q140i scanning laser range finder, 6 laser scanners, 1 Applaniz POS-LV pose estimation system (Inertia measurement, GPS, odometry), 1 high speed stereo vision system	7 Pentium M computers

that data so that intelligent control decisions can be made. As a result of this "information overload", it is becoming more difficult for the designers to explicitly describe the correlations between the many different sensory and mechanical components of the system [30].

Conversely, biological systems are capable of efficiently using data from a variety of sensing modalities (vision, auditory, tactile, etc.) to make intelligent control decisions [31]. The sensorimotor system of these organisms is integral in the coordination of sensory and motor activities, giving rise to bodily stability [32]. These systems remain flexible and adaptive to different environments and situations through complementary relationships between the static and dynamic components at the system level [33].

In order to create stable mobile robotic platforms that can efficiently perform complex tasks, developers should look to incorporating strategies from biologically-based sensorimotor systems. This will require the coordination of multiple sensors with multiple modalities to give a complete and accurate representation of the surrounding environment. In order to avoid the problem of "information overload" [30], data fusion methods should be applied to the sensor outputs to reduce the complexity of the raw data. The remainder of this chapter discusses methods of sensor fusion, which includes techniques for preprocessing sensor data by estimating their error characteristics (Section 2.3), as well as ways of combining information from multiple sensors using data fusion techniques (Section 2.3). In order to effectively use this fused sensor information to create a system model (Section 2.5), global management

architectures are needed, and these are presented (Section 2.4). Finally, guidance for managing the system models and applying sensor fusion techniques to the control of mobile robots is provided (Section 2.6). Particular attention is paid to sensorimotor integration.

2.3 Methods of Sensor Fusion

The most comprehensive definition of data fusion is given by Abidi and Gonzales as "the synergistic combination of information made available by various knowledge sources such as sensors, in order to provide a better understanding of a given scene" [34]. Methods of sensor fusion range from using the sensor with the modality best suited for each particular type of measurement (i.e., using pulsed radar to determine an aircraft's range, and using forward-looking infrared imaging to determine its angular direction [31]) to using fuzzy techniques to combine the sensor outputs, or to creating "virtual sensors" by combining and processing the output of multiple sensors.

Multi-Sensor Data Fusion (MSDF) encompasses the detection, association, correlation, estimation, and combination of data from a multitude of sensors. MSDF can be used to overcome sensor uncertainty, increase reliability, and provide better resolution. This technique involves data preprocessing, sensor modeling and estimation, data association, and data fusion [35, 36]. The result of this process is an optimal estimate of a particular state vector that relates to a specific component of the observed system (i.e., a state vector for the vision system, for the tactile system, etc.) [35]. Because MSDF systems also account for both spatial and temporal resolution with regard to the rated resolution of each element, they can be used to create a system that can gracefully deal with the gradual degradation or failure of its components [35].

In order to accurately model a system, a dynamic working description of its constituent sensors must first be constructed [35]. This process involves converting all of the raw sensor data into a "common language" [37]. This is accomplished by modeling the sensor's error characteristics that classify and describe a given measurement [38]. Then, an estimator or tracker must be found that can provide an acceptable estimate of sensor uncertainty. Conventionally, a Bayesian approach is used [39–41]. This is often implemented using an extended Kalman filter, although a more effective method utilizes information metric-based estimators [35].

The *Kalman filter* is a recursive algorithm that is used for statistically estimating the surrounding environment using a set of uncertainty observations. The standard approaches for deriving Kalman filters assume that the observed states are jointly distributed random variables. If the observed states are not jointly distributed, then orthogonality can be used to give a least squares estimate [42]. With a Kalman filter, each step's initial state and corresponding noise vectors are assumed to be mutually independent. Because Kalman filters are designed to be used in the presence of noise, their approximate fit is typically "close enough" to be useful [35, 36, 42–44].

Information metrics can also be used to evaluate and compare the available information in any multi-sensor system [42].The *Fisher information metric* is a particular type of information metric that can be used to characterize the statistical difference between observations. This measurement is found by calculating the moments of the observed data. For non-random parameters, the Fisher information matrix is defined in terms of likelihood; it holds true for both single and random variables [42].

In systems with multiple sensors and sensing modalities, the performance of the system can often be improved by combining the incoming data in a meaningful and productive manner. Towards this end, many different sensor fusion techniques have been proposed that combine, relate, and correlate the raw sensor data. Following are descriptions of the more widely used sensor fusion techniques.

Isolation fusers are among the most important fusing methods. They introduced the *isolation property*, which is used as a metric to compare and contrast nearly all other fusing methodologies [45–47]. The isolation property ensures that the information gained by fusing the data from the individual sensors is at least as complete and correct as the information available from each of the best individual sensors [45–47]. If this property is not fulfilled, then fusing the data is of limited to no benefit.

The *linear fuser* is one of the most basic sensor fusion methods. Here, the sensor's outputs are simply summed; this action satisfies the isolation property. The *optimal linear fuser* improves upon the linear fuser by minimizing the expected error. However, the optimal linear combination fuser requires that the sensors be equally distributed around certain global values; this is often difficult to realize in actual systems [44,45]. The *projective fuser* is another expansion of the linear fuser. The projective fuser computes the error regressions of the sensors and transfers the output of the sensor to correspond to the lower envelope of regressions for every point [45, 48].

In addition to these individual sensor fusion methods, *metafusers* can be used to combine multiple individual techniques into one homogeneous system. Metafusers are often implemented as configurations of different linear and projective fusers. They guarantee that the result is at least as good as the best sensor and the best fuser [45]. An example of one such metafuser is the *sample-based projective fuser*. Similar to the projection fuser, the sample-based projective fuser uses the lower envelope of regression curves for the sensors. The sample-based projection fuser then determines the sensor with the least error at each point on the feature space. Whereas the projective fuser projects the output of one optimal sensor onto all points in the feature space, the sample-based projection fuser allows for each point in the feature space to be described by a different sensor. As a result, the optimization determined by the sample-based projection fuser is the most optimal with respect to all of the projective fusers [45]. One of the drawbacks to using this method is that complete knowledge of the sensor distributions is required for exact computation. In most engineering and robotic systems, this is difficult to obtain due to the variety and complexity of the available sensors [45].

2.4 Information Management

In order to efficiently apply the fused sensor information for learning new behaviors and accomplishing specific tasks, a global management architecture is needed. This architecture also allows the behaviors and actions of the system to be correlated with the environment (as represented by the sensory system).

Information request prioritization is one of the most common methods of information-oriented management. Implementation of this method requires that certain sensors be given a higher priority when making control decisions for a unique set of conditions. This priority is determined from current mission and situational awareness, as gleaned from previous sensor readings and evaluations [49–52]. The *sensing actions selection* method extends upon the information request prioritization method by prioritizing the resources that are allocated to each sensing action. By limiting sensor resource allocation in addition to data selection, informational gain is maximized and resource allocation is minimized with respect to all levels of information acquisition [53–56].

The descriptively named method of *adding sensing resources for added quality of fused information* deploys additional sensing resources with the hope that the quality of the fused information will be enhanced [53]. Systems using this method typically operate in dynamic environments because more sensors are needed to adequately represent changes in the surrounding conditions [57–59]. However, more resources are needed to allow for the use of this increased set of sensing capabilities.

The *focus of attention strategy* also provides a good option in terms of resource utilization and performance. This strategy attempts to reduce the quantity of the data processed by the sensory system while increasing its informational value. This is accomplished by providing the system with the correctly associated sensor measurements (versus providing the system with all available information, correlating it all, and later using the measurements that are correctly correlated) [42]. One specific instance of the focus of attention strategy is *view selection*. Although view selection is best suited for vision search applications, it can also be applied to other "concrete" sensing modalities (i.e., acoustic and tactile modalities). The view selection technique chooses viewing locations and directions in order to improve the utility of the information gained from the sensing resources [53, 60]. It also decreases the number of resources that are required from the operating system by utilizing a narrower, more specifically focused view of the environment.

2.5 Models of Sensor Fusion

Sensory system models provide specific focus to the controlled system. This view of the external environment is then used by the system to create a personalized, internal representation of the space that it is operating in. This section describes the components that can be used to form the base architecture through which all other sensing components will interact.

The probability distributions of the sensors themselves can be used to relate the sensor's output to a desired feature in the external environment [45]. For example, the data of fused sensors is often grouped based upon the modalities that they can most accurately measure individually. Therefore, knowledge of the probability distributions allow the system to generate an effective internal representation of the external environment [45].

Another technique for reducing dataset complexity is *Principle Component Analysis* (PCA). PCA involves transforming the set of possible correlated variables into a smaller set of uncorrelated variables (i.e., principle components). Once these principle components are isolated, any redundant data is no longer considered for the computation of control. This allows the dataset to be greatly simplified and reduces the resources necessary for computation and control [61].

The *Adaptive Spline Modeling* (ASMOD) algorithm is an example of a model that includes the provision for adaptation. In order to allow for adaptive behavior, the ASMOD algorithm creates a set of different models, each of which correspond to a certain specific set of "candidate refinements" [35]. These candidate refinements can be categorized as either model building or model pruning. Then, the candidate model that produces the largest improvement in when approximating the external environment is selected [35].

There are three categories of model building refinements that are associated with the ASMOD algorithm: univariate addition, sensor multiplication, and knot insertion. *Univariate addition* simply incorporates a new, one-dimensional submodel with a new input variable into the existing ASMOD algorithm [35]. The incorporation of this submodel allows the updated algorithm to additively model the new variable and its correlations to the external environment [35]. *Sensor multiplication* replaces the current ASMOD algorithm with a new algorithm whose result is dependent upon the addition of at least one new input variable; this contrasts with univariate addition, which simply inserts a new input variable into the current algorithm. As a result, the sensor multiplication model can describe the coupled dependencies between combined input variables [35]. The *knot insertion* model is the most computationally intensive of the building refinements. It inserts a new basis function(s), or equivalent rule(s), in order to increase the flexibility of the ASMOD submodel [35].

The model pruning refinements share many similarities with the model building refinements. However, model pruning removes components from the existing model instead of re-building or adding to the existing model. There are three categories of model pruning refinements associated with the ASMOD algorithm: submodel deletion, tensor splitting, and knot deletion. *Submodel deletion* removes the current ASMOD algorithm so that the updated algorithm is no longer dependent upon its inputs [35]. The *tensor splitting* refinement creates more models with increased input simplicity. It does this by splitting a submodel with more than two inputs into submodels that depend of fewer combinations of inputs [35]. The final method of model pruning, *knot deletion*, is essentially the opposite of the knot insertion refinement. The knot deletion refinement reduces submodel flexibility by removing existing basis function(s) or equivalent rule(s). One benefit of using this method is that it reduces computational complexity [35].

2.6 Techniques for Data Management

Similar methods of data management are also used with sensor fusion models. However, these techniques represent higher-level management that must combine the information available from a number of sensors (both real and virtual) or modalities. Many of these models use dynamic coupling, which finds its roots in the field of Embodied AI. Dynamic coupling allows the "external" environment to be expanded to include, not only those features outside of the control system's physical realization, but also the system's internal representation of itself and associated states [62].

In addition to the typical physical sensors and detectors that make up sensory systems, *virtual sensors* can be constructed to expand the sensing capabilities of a system. Virtual sensors add a layer of software abstraction that allows for computationally-constructed data to be treated as if it came from a real device. This provides for a simple and consistent view of the various input devices. Software and data filters are common examples of virtual sensors. With respect to sensor fusion, the data from two or more detectors can be combined to create *phantom device data*. Phantom device data consists of data from a real device's inputs that are shaped by a set of constraints [43, 63].

The *cross modal map* combines many of the aforementioned models and methods of sensor fusion. The cross modal map describes the correlative mapping between different modalities of input (i.e., physical sensors, virtual sensors, and otherwise) and highlights issues related to evaluating the data from different sensory sources. These evaluations are necessary if any worthwhile correlations between the different data are to be found.

McIntyre and Hintz developed another model for sensor fusion systems that expands on the idea of information-oriented management (Figure 2.1) [64]. In their model, a mission manager mediates requests made by a human operator. At the most basic level, the mission manager functions as a feedback control system that continually monitors the state of the sensors in order to ensure optimal performance [65]. The mission manager also maintains the mission goals and objectives, which are determined a priori and offline, although they can be modified in real-time and stored. Put simply, the mission manager only cares that goals are accomplished and measurements are made relative to a set of priority and temporal constraints; the mission manager does not care which specific sensor(s) or measurement methods are used [30, 52, 64, 65]. In this model's hierarchy, the fusion space is simply where the sensor fusion takes place using the prescribed technique(s). The information space contains the sensor manager and mission manager. It also converts the sensor data, which is related to the current state of the observed environment, into to its internal, mathematical representation [30].

In order to implement information-oriented management techniques, it may be preferable to use a hybrid implementation of the different sensor management models that are under the direction of one mission manager. One such hybrid model, the *Sensor Rich Environment Sensor Management System*, was proposed by Schaefer and Hintz [30]; an example implementation of this approach is shown in Figure 2.2. This model was created to solve a new class of problems that arose from the continual ad-

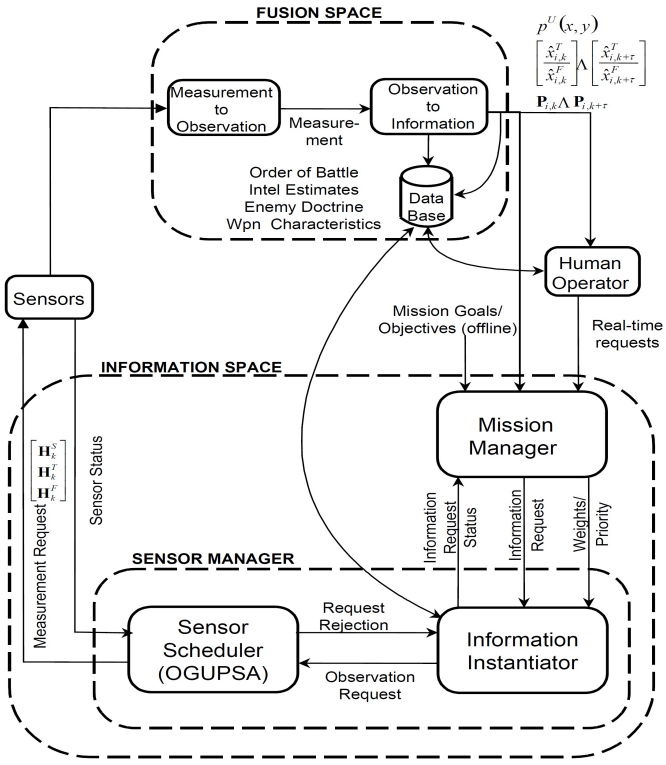


Figure 2.1: GMU Sensor Management and Scheduling Model [64]

dition of sensors without provisions for processing this information (i.e., "data overload") [30]. Here, the mission and sensor managers operate independently, with the sensor manager serving to convert the mission goals to information needs on an as-needed basis [30]. Furthermore, depending upon the scientific nature of the environment or situation, simply implementing a sensor manager without a mission manager may be sufficient [64].

Sensorimotor integration offers a different approach to creating a control architecture where the interactions between all of the sensory and motor elements in the system are connected together. Here, correlations are continuously updated with changes in the environment or sensorimotor elements themselves. Sensorimotor integration has the advantage of creating predictions independently and "on the fly" versus requiring that the model be determined beforehand by the experimenter. Bovet and Pfeifer [3,66] developed a sensorimotor architecture that uses neurons which are modified by Hebbian Learning. These Hebbian neurons are used to construct a base sensorimotor neuronal interface (Fig. 2.4). This interface consists of five distinct components: a current state, a delayed state, a state change, a virtual state, and a virtual state change. All of the sensors and motors (i.e., sensorimotor elements) are fully connected (Fig. 2.3). Through the use of a single common weight matrix, the virtual state tends towards a prediction of the actual state and state change [3,66].

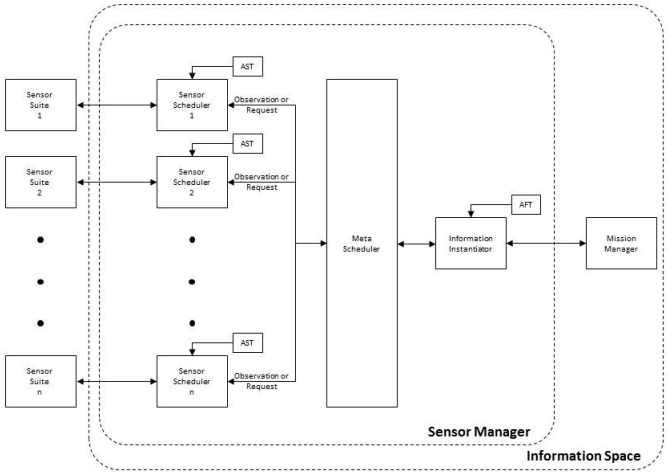


Figure 2.2: Sensor Rich Environment Sensor Management System [30]

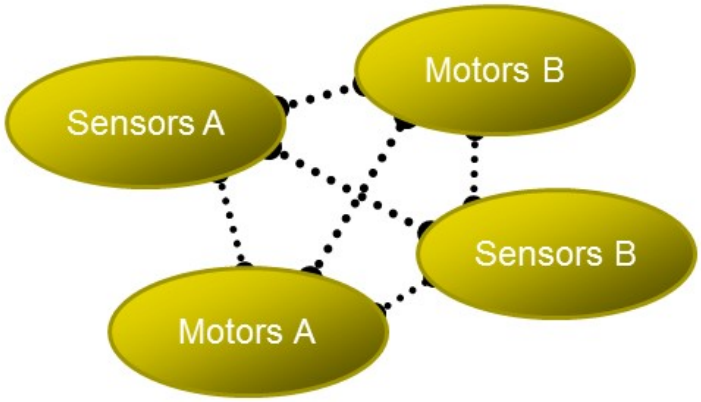


Figure 2.3: General Controller Architecture [3]

2.7 Discussion

As we continue to push the limits of robotic systems, increasing numbers and modalities of sensors will be required to accurately and efficiently perform a task. In the absence of intelligent methods for fusing this information together into concise units of data, computational demands will become increasingly unreasonable. As system complexity grows, intelligent architectures for mission management will be needed to effectively utilize the huge amounts of data available for efficiently accomplishing goal-driven tasks.

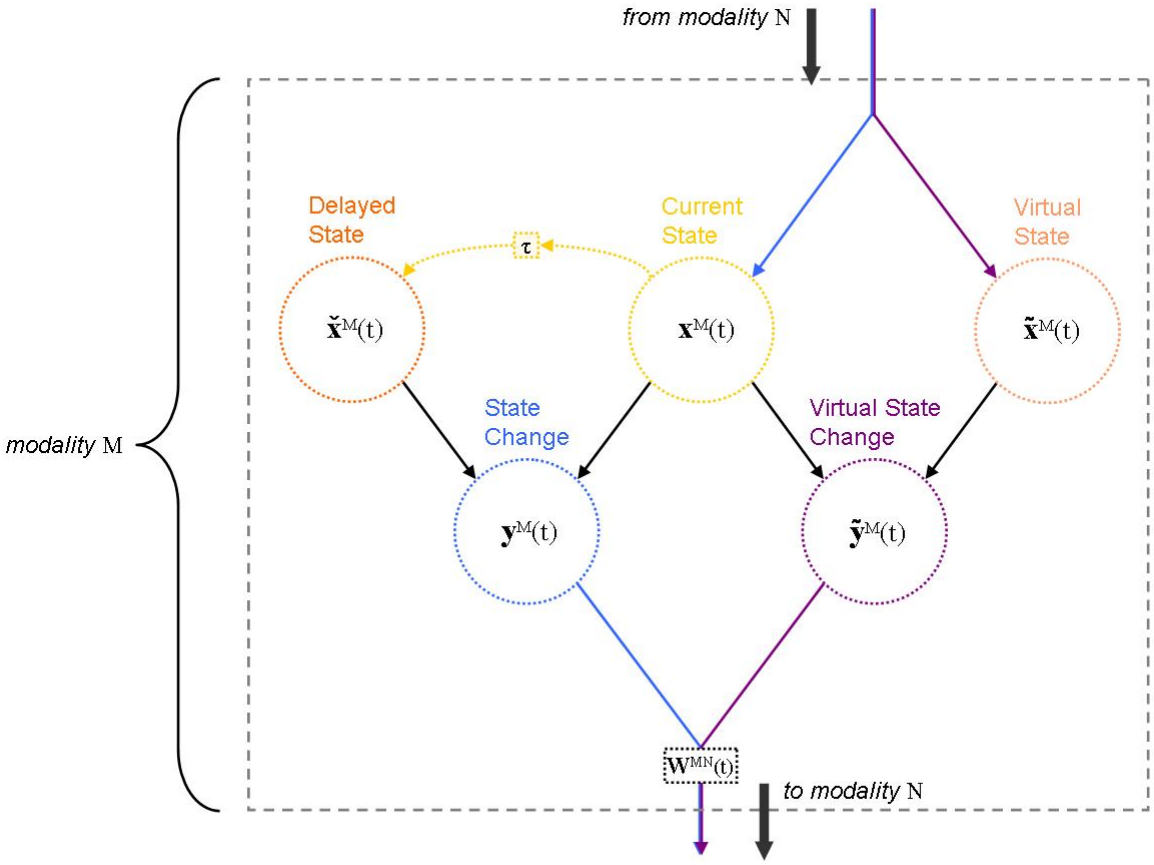


Figure 2.4: Sensorimotor Neuronal Interface [3]

This chapter has summarized many different sensor fusion methods for combining distinct modalities of data to present a more complete, accurate, and reliable estimation of the surrounding environment. This process begins by first converting the sensor data into a "common language" and finding an estimator or tracker that can be used to model the sensor uncertainty [37, 39–41]. This can be accomplished using a Kalman filter [35, 36, 42–44], linear fuser [44, 45], or sample-based projective fuser [45, 48]. Multisensor data fusion systems are optimal for combining data from various sensors to give information and correlations that cannot be gained from a single sensor. However, they suffer from problems related to the accuracy of the model of sensor uncertainty. For example, sensors are often characterized in a mathematically convenient way, such as assuming it is static, Gaussian, or has a zero-mean probability distribution. However, this is not realistic, and leads to failures of the sensor fusion methods and models [67]. These assumptions also do not account for changing sensor models and performance due to environmental conditions [67].

This chapter has also reviewed some of the available architectures for sensor fusion mission management such as the information-oriented management developed by McIntyre and Hintz [64], the Sensor Rich Environment Sensor Management System proposed by Schaefer and Hintz [30], and the Sensorimotor Integration architecture developed by Bovet and Pfeifer [3, 66]. The main advantage of mission management architectures is that they operate as a feedback control mechanism between the mission goals and objectives, and the actual sensors and systems. They provide an up-to-date internal mathematical representation of the system and its performance. However, this also highlights one potential disadvantage, namely that they must have an accurate model of the internal and external environments. This is often difficult, if not impossible, to obtain in all but the simplest environments.

2.8 Conclusion

Many of the systems and methods described in the review require extensive knowledge of the physical sensors and environments in which they are operating. Often, this is not feasible in dynamic environments that require continual updating of internal models. Additionally, sensor loss or degradation can become an issue. In order to limit the chance of incorrect characterizations that can overly constrain the system and provide additional sources of error, architectures such as the sensorimotor system described by Bovet and Pfeifer [3, 66] should be used.

Sensorimotor systems automatically build and maintain the sensor correlations between themselves and the environment. In order to incorporate a sensorimotor integration architecture into a robotic system, sensor selection and experimental design are very important. For example, the amount of information provided by the sensor should be balanced with the data processing demands and how rapidly the environment is changing. Although this concern is not new to sensorimotor integration architectures, any difference will be exacerbated due to the size and number of fully interconnected weight matrices that must be updated. For this reason, virtual sensors should be used whenever possible to limit the number of sensor values that must be correlated while maintaining informational quality. Additionally, experimental design is important to minimize biasing the learned correlations. In order to reduce the chances of the system learning incorrect rules, all experimental protocols for training and testing should be kept as open and simple as possible.

Chapter 3

A General-Purpose Mobile Robotic Platform: EvBot III Hardware and Software Architecture Design

3.1 Abstract

This chapter describes the design of the EvBot (Evolutionary Robot) III general purpose robotic research platform. The EvBot III was designed around the idea of ubiquitous modularity in software, hardware, and control systems. The EvBot III research platform is comprised of (1) a differential drive base with an attached turret and sensor shield, (2) a StackableUSB™ single board PC-104 computer, (3) a general purpose data acquisition system (CRIM-Daq), (4) a modular control architecture [1], and (5) a modular 3-D physics-enabled simulation environment [2]. The EvBot III has the potential to decrease development time and accelerate the progress of robotic and computational intelligence research.

3.2 Introduction

Mobile robot bases can be broadly characterized as (1) small and inexpensive with limited computational resources or (2) large and expensive with advanced computational resources. The most common inexpensive mobile robots are listed in Table 3.1. Because some of these platforms (i.e., EV3, VEX, VEX IQ, and TETRIX) are packaged as general purpose kits that can be used to construct a variety of robots, they have the advantage of being easily reconfigurable. The main disadvantage of these systems is their limited computational resources. Of the smaller, more inexpensive platforms, the Turtlebot 2 holds the most promise as a research tool because it comes with a netbook and a Microsoft Kinect.

The Khepera, Koala, and Pioneer P3-DX are more commonly used for conducting robotic research. Although they are more capable, these platforms cost substantially more than those previously dis-

Table 3.1: Inexpensive Robotic Platforms [68, 69]

Lego Mindstorms EV3	$350
Mobsya Thymio II	$190
iRobot Create	$130
Turtlebot 2 (Built on the iRobot Create)	$1995
Vex Robotics VEX IQ	$250
Vex Robotics VEX	$400
TETRIX	$380
Surveyor SRV-1	$495

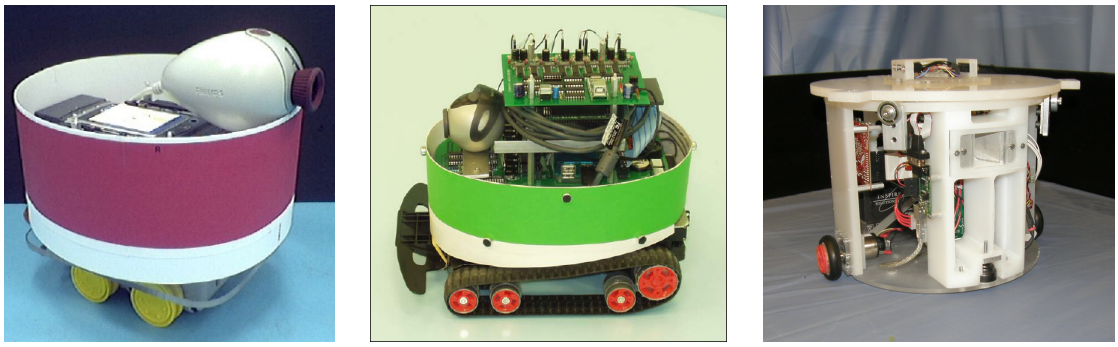
Table 3.2: Common Research Robotic Platforms [68, 69]

Khepera	$3200
Koala	$8400
Pioneer P3-DX	$4000

cussed. The prices provided in Table 3.2 reflect those for the base and the basic computing power needed to drive the robot [68, 69]. The overall cost of these platforms quickly rises with the addition of an on-board computer and peripheral sensors. For example, Jeanne Dietsch of MobileRobots noted that the actual cost of the Pioneer P3-DX is around $19000 when the advanced laser mapping and autonomous navigation software, laser bumpers, gyros, and wireless communication hardware are added [70].

3.3 The EvBot Platform

The EvBot (Evolutionary Robot) platform was created at the Center for Robotics and Intelligent Machines (CRIM) (North Carolina State University) in 2002 [26] as a tool for conducting general-purpose robotic research. It was designed as an inexpensive alternative to many of the more common research platforms at the time and used off-the-shelf components [26, 27]. The EvBot platform consists of a mobile base, intelligent control architecture, and Matlab simulation environment. The EvBot I, II, and III are shown in Figure 3.1.



(a) EvBot I (b) EvBot II (c) EvBot III

Figure 3.1: EvBot Platform

3.3.1 EvBot I and EvBot II

The EvBot I [26] and EvBot II [27] have been used to complete a number of different projects that have displayed the platform's versatility. These projects have included: (1) evaluating evolved neural controllers [14,71–77], (2) using an acoustic array to test UAV control algorithms for passively detecting and locating a variety of radar sources [78,79], (3) using an omnidirectional vision system for formation control [80], (4) serving as a test bed for optical robotic communications [81], (5) testing algorithms for smart sensor networks [82–84], and (6) repairing randomly distributed sensor networks [85].

Despite the flexibility of the EvBot I and II, there remain limitations common to both platforms. Some of the shortcomings of the EvBot I were addressed in the re-design of the EvBot II. For example, the EvBot I needed additional sensing capabilities to improve position and velocity control, as well as an easier method to add additional sensory systems. These changes were incorporated into the EvBot II with the addition of motor encoders and a 4-port USB hub [27]. However, both bases suffer from significant slippage in the tread-driven steering system, making accurate control difficult (especially for complicated movement patterns) [26, 27]. The EvBot I and II also suffer from problems related to decreasing battery life. Over time, the effective capacity of the batteries has decreased from $>$2.5 hours [26] to approximately 20 minutes. This can be attributed to the memory characteristics of the selected NiMh batteries (i.e., voltage depression). Additionally, the EvBots have to be taken off-line to recharge their batteries, rendering more time-intensive experiments impossible. There is also a noticeable shortage in computational power inherent to both bases (i.e., both bases used a PC-104 Pentium/586 class motherboard [26, 27]). This becomes problematic when running real-time, computationally intensive controllers and/or algorithms. For example, this short-coming was highlighted during experiments involving control of the EvBot II using a 360° camera setup, which required complex vision algorithms to be run in real-time [80]. Similarly, there was a significant degradation in computational performance when higher-level programming environments (i.e., Matlab) were run on the base.

Although immensely important for training and testing, the Matlab simulation environment is also limited. The simulation environment was not created with modularity as its primary goal. For example, it was limited to the 2-D maze environment, which confined the EvBot to lab use only. The addition of new sensor modalities (i.e., the acoustic array) was also unfeasible. The simulation environment was further limited by the lack of a governing physics model.

3.3.2 EvBot III

In order to address the issues outlined above, the EvBot III has been designed to be more modular both in terms of hardware and software. Other improvements include increased computational resources and more powerful batteries that have the ability to self-monitor and recharge autonomously.

In order to incorporate these features into the new EvBot III, a custom base was designed and built. Similar to the EvBot I and II, the EvBot III has continued to embrace the use of the USB standard for device data connections. This standard has shown good support for increased hardware modularity, and is used in the new general-purpose data acquisition system (CRIM-Daq) and embedded computer (StackableUSB™, Micro/sys, Inc.). The improvements to the EvBot III base are discussed more fully in Section 3.4, with rationale and recommendations provided in terms of design decisions, material selection, manufacturing, and assembly. Upgrades to the hardware, including a discussion of the CRIM-Daq and associated daughter boards, are detailed in Section 3.5.

In addition to these hardware improvements, the software has also been updated. These upgrades include a modular object-oriented control architecture, custom recompilable Linux build tree [1], and a new 3-D modular simulation environment [2]. The system's increased software modularity provides more efficient lower-level interfaces and controller components. The new reconfigurable/recompilable custom Linux operating system excludes all unnecessary components from the rolled-out distribution, and supports real-time scheduling and processing components [1]. A brief discussion of the new Linux build tree and modular simulation environment is provided in Section 3.6.

3.4 Base and Charging Station Design

The motivating factors driving the redesign of the EvBot platform include the full embrace of both hardware and software modularity, as well as precise measurement, control, and modeling of autonomous robotic systems. These factors, along with the overarching goal for use as a general-purpose research platform, have guided the design of the EvBot III base.

3.4.1 EvBot III Base

Many of the problems associated with the previous versions of the EvBot stemmed from deficiencies in the off-the-shelf bases used. The new EvBot III base (Fig. 3.2) was completely custom-built using

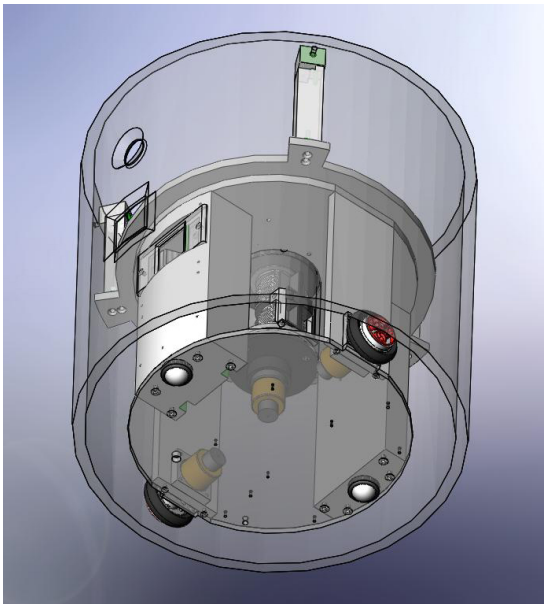


Figure 3.2: EvBot III base design model

Ultra-high-molecular-weight polyethylene (UHMW), laser-cut acrylic, 3D printed ABS thermoplastic, aluminum, and styrene. The UHMW was used for the majority of the base because it is easy to machine. The top and bottom were made using laser cut acrylic so that a person could easily see into the base. Aluminum was used for support elements and the center shaft of the slip-ring, and the 3D printed ABS thermoplastic was used in the brackets that attached the sensor shield to the base.

Drive train slippage was a major problem noted with the EvBot I and II bases [26, 27]. In order to address this in the redesign of the EvBot III, a differential drive system was used in combination with an actuated rotational platform. This new base allows for "pseudo-holonomic" movement, where pseudo-holonomic is meant to indicate a system that appears holonomic to an outside observer, although the underlying hardware is non-holonomic. A pseudo-holonomic design was chosen over a truly holonomic one in order to minimize design and development time, as well as keep future maintenance and upgrades as easy as possible. Pseudo-holonomic behavior is achieved using a turret in concert with an integrated slip-ring. The slip-ring allows for the transmission of DC power through the base using a standard 24-pin ATX power connector. Control signals from the base control system (located in the bottom portion of the base) are sent via USB or RS-232.

The new EvBot III was also designed with all of the hardware and processing resources needed to drive the robot isolated to the bottom half of the base. Therefore, the computer in the top part of the base only needs to provide high-level movement commands, and the bottom of the base can be easily switched out. Not only does this remove the computational load of driving from the main controller, but it allows for one controller to be reused with many different types of bases.

The EvBot III platform also allows new sensors to be added through the use of an outer shield that is attached to the actuated rotational platform (Fig. 3.2). The shield was made from styrene, which is both lightweight and inexpensive. External sensors can be mounted to the shield such that they remain in constant relative position regardless of the orientation of the rotating platform. Furthermore, the shield allows for easier implementation of the "focus of attention" method of sensor fusion [42]. The shield also makes the EvBot platform easier to simulate because it contains the entire base, including the drive wheels, and gives the EvBot a uniform circular outline.

3.4.2 OceanServer Power Solution

In order to address the power limitations present in the EvBot I and II, an OceanServer™ Intelligent Battery and Power System (IBPS™) was used in the new EvBot III. The IBPS™ includes a MP-04SR/FR battery management module, a DC123SR DC-DC power supply module, and a BA-95HC Li-Ion smart battery pack. An EVAL233R USB-Serial module from FTDI was also added for monitoring and configuring the battery system from the central computer.

The MP-04SR/FR battery management module can charge, discharge, and monitor up to four Li-Ion battery packs. It uses an RS-232 port to provide complete information about all connected battery packs. The information available for each connected battery includes, but is not limited to, current, voltage, amp-hours, run time to empty, and time to full charge. The MP-04SR/FR also seamlessly connects and delivers unregulated power to the DC123SR DC-DC converter module [86].

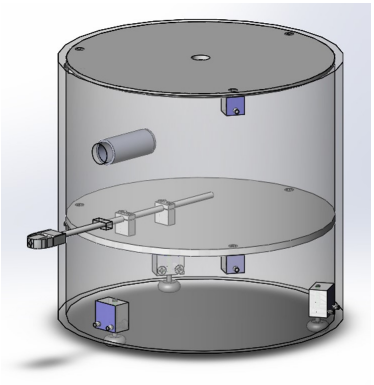
The DC123SR ultra high efficiency ATX DC-DC converter module supplies the following regulated DC voltages: +3.3V at 10A, +5V at 10A, +12V at 12A, and -12V at 12A. All of these conversions are done with greater than 95% efficiency [87]. Since each ring of the slip-ring in the EvBot III was only rated to 5A, it was not feasible to provide all of the regulated voltages to the top of the rotating platform. With 18 total rings available in the EvBot III slip-ring, 2 rings were used in parallel to provide +5V at 10A, and 3 rings were provided for each +12V at 12A and -12V at 12A to the top of the EvBot platform.

The final IBPS™ component is the BA-95HC Li-Ion battery pack. Each battery pack has a 6.6Ah capacity with an unregulated output of 14.4V nominal [87]. The EvBot III base contains a removable compartment that can hold up to two of the BA-95HC batteries. If only one battery is installed, a place holder is used where the second battery would go.

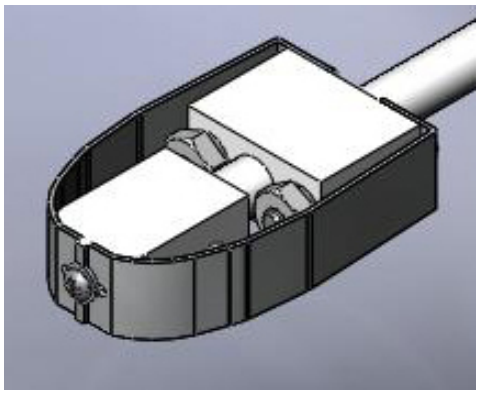
3.4.3 Portable Charging Station

The previous EvBots had to be taken offline to recharge. In order to address this limitation, a portable charging station (Fig. 3.3a) was created. The charging station is modeled after one used for the CyberMotion K2A mobile security robot (CyberMotion, Inc.). The EvBot III can autonomously recharge by aligning the plug on the charging station (Fig. 3.3b) with the charging port (Fig. 3.3d) located on the front of the EvBot III base (Fig. 3.3c). In order to aid with alignment of the EvBot III with the charging

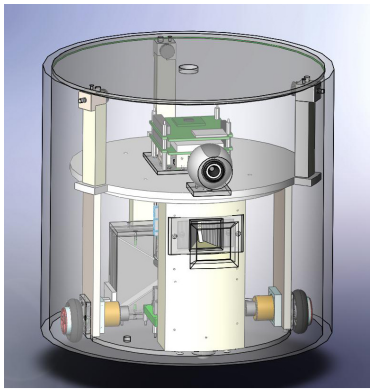
station, the shield of the charging station contains a hole with an LED that is at the same height as the camera hole of the EvBot, When the EvBot is aligned with the charging station, the LED is centered. The recharging process is monitored and controlled by the IBPS™ (Section 3.4.2).



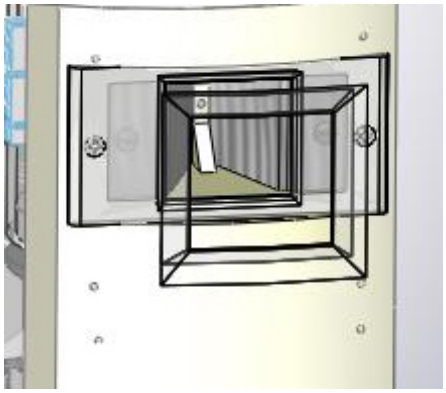
(a) EvBot III Charging Station



(b) EvBot III Charging Station's Plug



(c) EvBot III



(d) EvBot III Charging Receptacle

Figure 3.3: EvBot III Portable Charging Components

3.5 Hardware Design (CRIM-Daq)

To further aid in hardware cross-compatibility, a modular control system was developed and assembled. This system is based around a multipurpose USB data acquisition board (CRIM-Daq) and associated peripherals (i.e., CRIM-Daq motor-driver board and CRIM-Daq inertial measurement unit). The CRIM-Daq connects to the EvBot's StackableUSB™ on-board computer via USB, and relays the sensor and motor data from the base. It also converts the high-level motion commands from the computer to the

low-level motor commands required to drive the base. In addition to robotics applications, the CRIM-Daq can be used for any project that requires data collection from one or more sensors; this is expected to reduce the production cost per board, and subsequently the cost of each new EvBot III.

The CRIM-Daq (Fig. 3.4) consists of a modified CRIM-Mote [88], an integrated ex430-f2013 programmer/debugger (Texas Instruments), an integrated 4-port USB hub, and pass-through headers to allow for stackable expansion boards. The onboard CRIM-Mote circuit was modified to allow access to all of the I/O and signal pins from the msp430 using the pass-through headers. These headers give peripheral sensor boards access to 29 of the 32 possible general purpose I/O pins including two configurable on-chip operational amplifiers, a 10-bit A/D converter, one UART, one IR communication port, one I^2C port, and two SPI ports. A maximum of 112 individual nodes can be used on the I^2C port, and a maximum of 22 nodes can be connected to the SPI port. Therefore, many different sensor boards can be connected to and used with the same Daq by stacking each on top of the pass-through headers. Then, once the msp430 is programmed with the application-specific code, all of the data from the attached sensor boards is accessible through the msp430 on the same USB port. The integrated ez430 programmer/debugger allows any code to be easily updated without any external hardware.

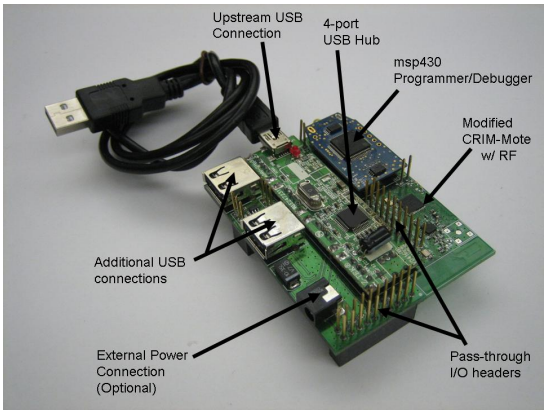


Figure 3.4: CRIM-Daq with labeled components

The CRIM-Daq can be either self-powered (i.e., an optional external power connection is included) or bus-powered. The board also includes circuitry to provide +3.3V, +5V, and $\pm$12V to any connected sensor boards. The circuitry to provide $\pm$12V was included to allow for use of differential input signals.

3.5.1 CRIM-Daq Daughter Boards

A motor control board (Fig. 3.5a) was the first daughter board developed for the CRIM-Daq. The motor driver board contains four H-bridges (MC33887, Freescale Semiconductor) with back EMF protection,

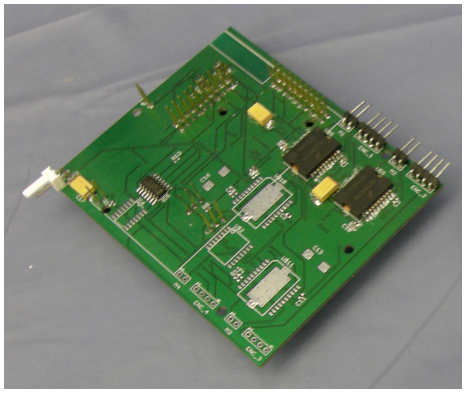
the circuitry required to interface with the shaft encoders of four motors, and the circuitry required to monitor the current sense output of each of the H-bridges.

The motors used to drive the EvBot III base (GHM-04, Lynxmotion) [89] have a 2.5A stall current, and the H-bridges are rated to handle DC load currents up to 5.0A [90]. However, after weeks of use, the CRIM-Daq motor driver board failed. The cause of these failures was later traced to the motors, which actually drew >20A when starting or changing directions. As a result the RoboClaw 2x30A motor controller (Fig. 3.6) was used to drive the base. The next revision of the CRIM-Daq motor driver will be updated to handle the increased current load.

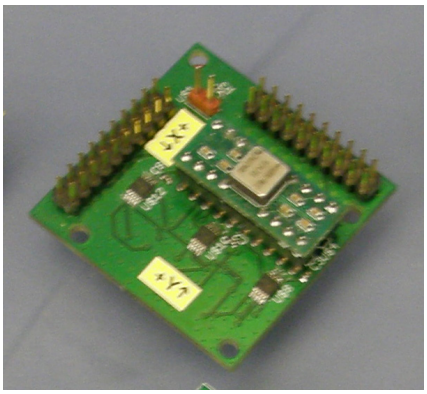
The Inertial Measurement Unit (IMU) was the second daughter board (Fig. 3.5b). This board is comprised of a 3-axis, ±3G accelerometer (ADXL330, Analog Devices) and a Z-axis gyroscope (ADXRS613, Analog Devices). In order to simplify the interface to the CRIM-Daq and ensure the temporal accuracy of the measurements, all of the sensor readings are transmitted to the CRIM-Daq through five serially-linked SPI ADCs with simultaneous sampling. Also, in order to increase the accuracy of the gyroscope, one of the ADCs is used to read in a temperature offset value so that the gyroscope can be calibrated.

The final daughter board is an 8-channel, 16-bit, 250 ksps A/D board (Fig. 3.5c). This board simultaneously samples all 8 A/D channels and sequentially passes the data to the CRIM-Daq using SPI.

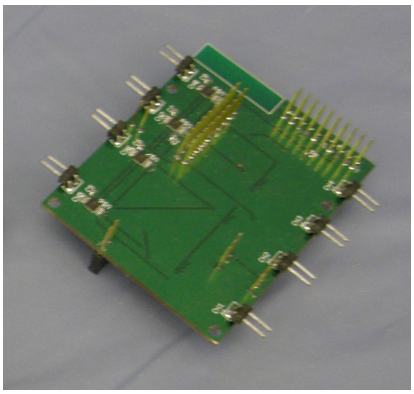
In the future, additional daughter boards will be developed and include (1) a servo control board, (2) a signal generation board, (3) a pressure sensing and control board, (4) an LCD board, (5) a screw terminal signal access board, (6) a power control board, and (7) a switch inputs board.



(a) Motor Control Board



(b) IMU Board



(c) 8-Channel 16-Bit A/D Board

Figure 3.5: CRIM-Daq Daughter Boards

Figure 3.6: RoboClaw 2x30A Motor Controller

3.6 Software Design

3.6.1 Linux Build Tree

To begin the modularization of the software aspects of the control architecture, a new in-house Linux build tree [1] has been created that allows components to be added/removed before each new build. This enables greater flexibility when choosing new hardware components because the necessary drivers are included into the build tree, and the distribution rebuilt to suit the needs of any new mission scenario. This build system also allows for easier system upgrades when the Linux operating system kernel gets updated. Additionally, developers can continually incorporate new sensors, actuators, and other peripheral devices as they are released and ported to Linux through kernel updates [1].

An object-oriented, modular control architecture [1] has also been developed. This architecture enables use by multiple developers and make the addition of new control structures more transparent (i.e., only small, hardware-specific adjustments will be needed). Also, because the architecture is object-oriented and modular, only minimal software modifications is needed to transfer it between different robotic platforms [1].

Currently, base objects are being created that will allow for the rapid development of controllers without the need to code modules to interface with the base hardware. Furthermore, controller interoperability is possible because the base objects use a common language and framework. All of the code is being written in C and C++ so that the control architecture will not be a computational bottleneck, as was experienced in the current Matlab structure. However, the system will continue to support controllers designed using Matlab and Simulink for quicker, proof-of-concept development. Working within a C/C++ framework will also allow for parallel program execution. These improvements are expected to make development easier and code more reusable, as well as speed up program execution [1].

3.6.2 EvBot III Firmware

The CRIM-Daq firmware includes all of the necessary drive functions to train a controller. Currently, PID control is used to control the motors on the EvBot III. A four character instruction set is used for communication between the main controller (laptop computer) and the base. This instruction set can be easily extended. Current instructions are first broken into the component they are targeting (i.e., motors, encoders, accelerometer/gyroscope, ADC, and EvBot base drive). Then, the specific instance of that component is specified. The last part of the instruction is the requested command and value (i.e., return the motor PWM value, set the motor direction, set the PWM value, set the motor velocity in radians/second, set the base velocity, move the motor a specified distance, reset the encoder, get the encoder value, get the gyroscope data, get the accelerometer data, or request the board identification string).

3.6.3 Simulation Environment

The EvBot Simulation Environment is meant to reduce the time needed for performing repetitive experiments by speeding up the gathering of preliminary data. The new EvBot III Simulation Environment [2] was designed to interface with the modular control architecture so that new sensors or actuators could be incorporated into the simulator. The new simulator also uses a dynamics simulation engine and collision detection engine to render physical movement and interaction of the robots, their moving parts, and the environment. The physical construction of the robots and environments is specified using configuration files. Visual sensor support is provided by a simple 3D graphics engine. These provisions make possible the rapid development and evaluation of new controllers, sensor configurations, or platforms. Furthermore, due its modular nature, more stable and accepted controllers such as the Robot Operating System (ROS) [91, 92] can be easily incorporated and used as a comparison for new algorithms.

3.7 Discussion

Nature has, at its core, modularity. While all individual organisms are unique, they share a majority of common elements related to form and function. In order for robotics and computational intelligence to progress to true autonomy, this notion of ubiquitous modularity of hardware, software, and control systems needs to be fully embraced. The EvBot III platform was designed with this as its primary goal.

The EvBot III base was custom-built to allow for new methods of actuation (i.e., the "pseudo-holonomic" base uses simple control to generate complex movements) and the easy addition of new sensors through and outer shield. Because the shield was circular, simulation was simplified in both 2D and 3D. Additionally, the shield reduced concerns related to sensor obstruction, and enabled rapid sensor reconfiguration. This proved helpful when experimenting with different alcohol sensor setups (Chapter 4).

Many of the problems associated with the previous versions of the EvBot were also addressed in the re-designed system. For example, the current base was implemented with a differential drive system, which eliminated the problem of drive-train slippage that was associated with the previous track designs. Additionally, the EvBot III has an improved power management system (OceanServer™ Intelligent Battery and Power System), which removed battery life as a concern for any of the experiments undertaken thus far. The provision for self-charging was also implemented.

There were a few complications noted with the current EvBot III. For example, the size and weight of the base are a concern. The base itself weighs $\approx$7.25 kg without the laptop. Although no problems were noted with the motors (GHM-04) during experimentation, the shaft is only rated for 1.0 kg-cm load at 7.2 V. For this reason, the EvBot III was stored with its wheels elevated to prevent excessive loading of the motor shafts when not in use. To lessen the weight of the base, the next version will be made using a majority of laser-cut acrylic versus UHMW. This will also reduce production time and cost.

The next version of the EvBot III base will also use a different slip-ring. In the current version, DC power was transmitted using a standard 24-pin ATX power connector, and control signals were sent via USB or RS-232. . However, there was a problem with movement-induced noise in the signal lines. In order to remedy this in future versions of the base, the slip-ring will be used solely for power transmission ($\pm 12\,\mathrm{V}$), and communication through the center of the base will be achieved using an infrared (IR) link. This will reduce the size of the slip-ring from 18 rings to 6 rings. Additionally, the shaft of the slip-ring will need to be implemented as a hollow tube to accommodate the data IR link.

3.8 Conclusion

The EvBot III's integrated development of modular hardware and software has the potential to decrease development time and increase the output and progress of robotic and computational intelligence research. The EvBot III platform includes (1) a mobile base, (2) control software, and (3) a simulation environment that incorporates a dynamics simulation engine and collision detection engine to render physical movement and interaction of the robots. The current base is relatively low-cost and can be easily customized through the use of outer sensor shields. Additionally, because the higher-level controller is separated from and does not need knowledge of the lower-level base kinematics, algorithms can be easily ported to different platforms. Using the in-house Linux build tree, software updates and new sensor, actuator, etc. drivers can be easily added as they are released and ported to Linux through kernel updates. The simulation environment allows for new base designs and control algorithms to be easily and realistically tested.

Chapter 4

Chemical Sensing for Mobile Robots: An Improved Method for Implementation in Dynamic Indoor Environments

4.1 Abstract

When considering biologically-inspired robotics, chemical sensing provides a logical starting point because it is the most widely represented sensing modality among living creatures [28]. Chemical sensing has applications in detecting leaks, explosives, or disaster victims, as well as coordinating activities within a swarm of robots [28]. Current robotic chemical sensing is implemented using one of three approaches: (1) using only passively-sampled chemical sensors, (2) using wind direction sensors along with passively sampled chemical sensors, and (3) using actively sampled chemical sensors. While the later two approaches are more efficient, they can be inaccurate in indoor environments with lower and/or variable wind speeds. However, relying solely on chemical sensors presents its own set of difficulties, as the majority of the airborne chemical sensors that are appropriate for use in robotics suffer from poor response time (i.e. 10-150 s). This chapter reports on a novel homing algorithm and sensor configuration that can be used with any mobile platform to navigate to a chemical source. This was tested in an indoor maze environment using the EvBot III platform. Here, the average error between the actual and calculated angle to the source was found to be 21.18°, which was accurate enough to allow the robot to repeatedly home to the source from a distance of approximately 2.75 m.

4.2 Introduction

Chemical sensing has potential for widespread application in the field of robotics. Robotic olfaction can be used to find chemical leaks, explosives, or disaster victims [28]. When working with swarms of robots, olfaction can also be used to navigate or coordinate cooperating actions. For example, a robot can release a volatile chemical to signal a breakdown and notify the other robots that it needed assistance [28]. A similar method can be used to coordinate searches by indicating previously searched locations, as well as helping to identify promising locations to search more fully [28].

When considering biologically-inspired robotics, chemical sensing provides a good starting point because it is the most widespread sensory modality among living creatures [28]. Simple organisms use this basic mechanism to navigate towards a food source, etc. This allows researchers to study emergent intelligence and determine the robotic architectural features that allow for a particular behavior. Later, this information can be used to create self-adaptive and intelligent robots [93].

Robotic airborne chemical sensing is commonly implemented using one of three approaches: (1) using only passively-sampled chemical sensors, (2) using wind direction sensors along with passively sampled chemical sensors, and (3) using actively sampled chemical sensors. These approaches, including the algorithms, sensors, and test setups are summarized in Table 4.1 and arediscussed in detail in the following sections.

4.2.1 Passively-Sampled Chemical Sensors

The *E. Coli Algorithm*, *Gradient-Based Control Algorithm*, *Modified Bombyx Mori Algorithm*, and *Multi-Layer Feedfoward Neural Network* use only passively-sampled chemical sensors to locate an odor source. Passively-sampled setups have the advantage of being simpler than actively-sampled setups or those requiring additional wind directions sensors, but suffer from problems related to accuracy.

The *E. Coli* algorithm [94] is the simplest algorithm, and uses only one sensor to determine the gas concentration. Here, the robot randomly rotates and moves ± 0.50 m depending upon if the current alcohol value is higher or lower than the previous alcohol value (i.e., if the current value is higher, the robot rotates $\pm 5°$, and if it is lower, the robot rotates $\pm 180°$) [94].

The *Gradient-Based Control* algorithm [94] is more complex than the *E. Coli* algorithm. It uses two sensors to achieve a more iterative set of control steps that are based on the type 2 vehicle described in the Braitenbergs book "Vehicles: Experiments in Synthetic Psychology" [22]. For this algorithm, the robot determines which sensor is reading the highest concentration of gas. Then, the robot turns in the direction of higher concentration at an angle that is proportional to the difference between the two sensors (limited to $\pm 16°$). The robot then moves forward 0.02 m and repeats this sequence [94].

The *Modified Bombyx Mori* algorithm [95] is based on the search strategy used by the male silkworm moth (*Bombyx Mori*) when it detects the particular conspecific female pheromones. The male silkworm moth uses wind direction to estimate the direction of the pheromone source [95]. A modified version of

Table 4.1: Summary of Approaches to Robotic Airborne Chemical Sensing

Strategy	**Gas Sensors**	**Wind Speed Sensors**	**Wind Direction Sensors**	**Tracking Distance**	Environmental Description
E. Coli Algorithm	1	–	–	N/A	uniform airflow (≈1.5 m/s)
Gradient-Based Control Algorithm	2	–	–	2.0 m	uniform airflow (≈1.5 m/s)
Modified Bombyx Mori Algorithm	2 (6)	–	–	N/A	unventilated environment
Multi-Layer Feedfoward Neural Network	2 (8)	–	–	0.80 m	weak airconditioning
Step-by-Step Progress	4	4	1	0.80 m	uniform airflow (≈0.2 m/s)
Zigzag Approach	4	4	1	0.8 m	uniform airflow (≈0.12 m/s)
Dung Beetle Algorithm	2	–	1	2 m	uniform airflow (≈1.5 m/s)
Plume-Centered Upwind Search	1	–	1	1 m	uniform airflow (≈0.30 m/s)
Bombyx Mori Algorithm	2	–	1	2 m	uniform airflow (≈1.5 m/s)
Spiral Surge Algorithm	1	1	1	6 m	uniform airflow (≈1.5 m/s)
Multiphase Tracing Algorithm	2	4	1	1.5 m	non-uniform airflow (≈0.10-0.30 m/s)
Gas-Sensitive Braitenberg Vehicle	2 (6)	–	–	1-4 m	unventilated environment
Odor Compass	2	–	–	1.25-1.7 m	non-uniform airflow (≈0.10-0.30 m/s)

this strategy [94] was developed for robots working in an indoor environment without strong unidirectional airflow. Here, the robot performs an initial random search until it is triggered by the presence of gas. The random search is implemented by instructing the robot to drive in straight paths until it enters the clearance area around an obstacle. Once this occurs, the robot randomly rotates and proceeds on a straight path. When the robot detects the presence of gas, it starts a zigzag movement pattern by turning $\approx 65^{\circ}$ to the side of highest sensed concentration. After this initial turn, the robot performs six zigzag turns followed by straight movements of the following lengths (in order): 0.20 m, 0.30 m, 0.50 m, 0.70 m, 0.90 m, and 0.55 m. After these zigzag movements, the robots turns in a circular motion with a radius of ≈0.50 m. If the robot loses the gas plume during the zigzag motion, the robot reinitiates the search motion at the initial ≈65° turn [94].

Both the *E. Coli* and *Gradient-Based Control* algorithms were tested using the reactive autonomous testbed (RAT; Fig. 4.1) [28]. The RAT robotic platform [96] has two bilateral polymer gas sensors (treated as one sensor when using the *E. Coli* algorithm). The robot also includes a wind sensor that can be used to determine wind speed and direction, as well as two collision detection whiskers [94]. In order to test the success of these algorithms, a uniform airflow speed of ≈1.5 m/s was used. When the *E. Coli* algorithm was tested, the robot was not able to locate the alcohol source [94]. However, the robot was able to find the alcohol source from a distance of ≈2 m when the *Gradient-Based Control* algorithm was used [94].

The *Modified Bombyx Mori* algorithm was implemented and tested on the model ATRV-Jr. Robotic base from iRobot. This base has two sets of three gas sensors that are located on either side of the robot at the front corners. In order to correct for differences in the gas sensors, the readings were normalized. Because the *Modified Bombyx Mori* algorithm was developed for indoor environments, no wind direction or speed sensing was used [94]. During testing, the SICK laser range scanner on the base was used to correct the position data obtained from the robot's odometer, as well as to detect if the robot was in the clearance zone of obstacles. For this experiment the gas source was placed an average of 1.96 m from the robot's starting position in an unventilated environment. The robot was not able to successfully locate the gas source [94].

The *Multi-Layer Feedfoward Neural Network* was implemented by Farah and Ducket [97] on a Koala mobile robot fitted with an Örebro Mark II mobile nose [97]. The Örebro Mark II mobile nose contains two sets of metal oxide gas sensors that are placed inside of two separate tubes. These tubes are separated by a wall and face towards opposite sides of the robot (Fig. 4.2). In order to train and test the robot's ability to determine the direction of the alcohol source, the robot was driven in seven different directions (85°, 55°, 25°, 0°, -25°, -55°, and -85°) from each of four different starting locations (alcohol source was 0.80 m from the robot). These experiments were conducted in an air-conditioned environment (i.e., no/weak directed airflow). 75% of the data collected was used for training, and 25% was used for testing. Using this multi-layer feedforward neural network, the robot was able to locate the alcohol source from a distance of ≈0.80 m [97].

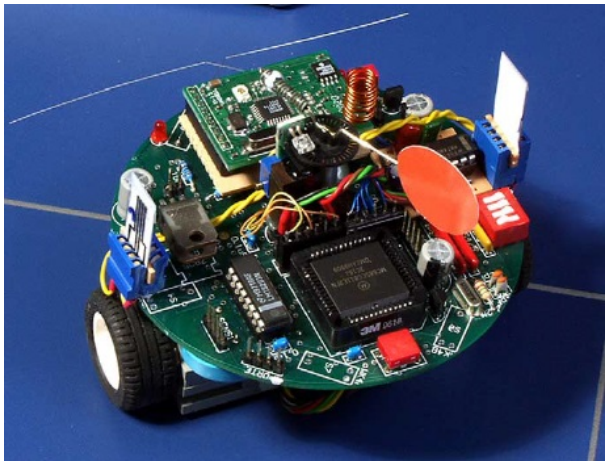


Figure 4.1: Reactive Autonomous Testbed (RAT) [94]

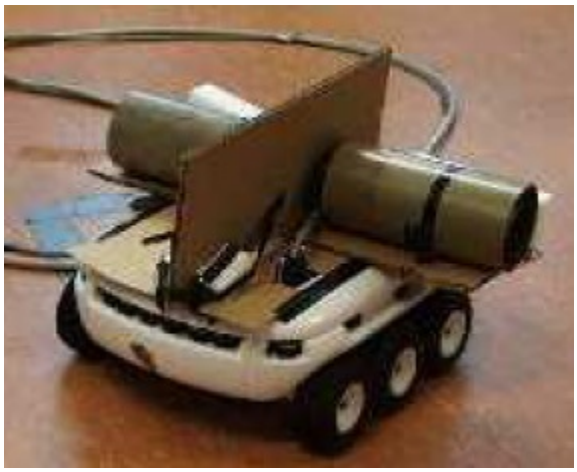


Figure 4.2: Örebro Mark II Mobile Nose [97]

4.2.2 Wind Direction Sensor with Passively-Sampled Chemical Sensors

The *Step-by-Step Progress Method*, *Zigzag Approach*, *Dung Beetle Algorithm*, *Plume-Centered Upwind Search Algorithm*, *Bombyx Mori Algorithm*, *Spiral Surge Algorithm*, and *Multiphase Tracing Algorithm* all use a wind sensor in concert with chemical sensor(s) to locate an odor source. Although the addition of a wind sensor adds to the complexity of the control system, robots using both are more efficient and effective at locating an alcohol source, even at a greater distance. For example, robots using one

of these seven algorithms were always able to locate an odor source at distances up to 6 m. When only chemical sensors were used (Section 4.2.1), the robot failed to locate the source for two out of four cases at distances of only 2 m.

The *Step-by-Step Progress Method* [98] uses gas sensors to track the concentration gradient to the center of the plume, as well as an anemometric sensor to move upwind (wind speed: 0.20 m/s). The wind direction is estimated to an accuracy of 90° by selecting the anemometric sensor with the lowest value. The robot rotates so that the gas sensors are perpendicular to the wind direction in order to achieve the largest gradient. The direction of the alcohol source is determined by calculating the intermediate angle between the wind direction sensor and the gas sensor with the largest response. The robot is then instructed to drive 0.02 m in that direction [98].

Similar to the *Step-by-Step Progress Method*, the *Zigzag Approach* [98] uses wind direction to locate the alcohol source. The robot first turns 60° with respect to the upwind direction, and drives in a straight line until it reaches the near edge of the plume. Then, the robot continues driving straight until it reaches the far edge of the plume. From here, the robot rotates back to a preset angle (with respect to the upwind direction) and drives straight until it reaches the near edge of the plume. This process continues, with the sign of the turn angle alternating at each edge. In order to test this algorithm, the rotation angle was preset to $\pm 60°$ and a simple threshold value used to determine if the robot was in the plume. Additionally, a method was implemented to correct for erroneous movements that resulted in the robot driving out of the plume. When this occurred, the robot simply backtracked until the alcohol values fell below the fixed threshold.

Implementation of the *Dung Beetle Algorithm* [94] is similar to the *Zigzag Approach* after the robot has located the plume. In order to find the plume, the *Dung Beetle Algorithm* requires the robot to turn 90° counter-clockwise (relative to the wind direction) and drive 0.50 m until alcohol is detected (i.e., as determined by a predefined threshold value) [28, 94, 99].

The *Plume-Centered Upwind Search Algorithm* [98] navigates to the odor source more directly compared to the *Zigzag Approach*. Using this algorithm, the robot first locates the center of the plume by driving tangent to the wind. It continues to drive at a tangent until the edge of the plume is reached (determined by a preset threshold value) before returning to the center of the plume. The robot then continues to drive upwind, using gas sensors to ensure that it is staying centered in the plume by checking the concentration gradient.

The *Bombyx Mori (Silkwork Moth) Algorithm* [94] builds on the modified approach (Section 4.2.1) by first orienting the robot to face the wind (i.e., versus randomly driving around). The robot then waits for the airborne chemical to be detected before starting its search. Once the chemical is detected, the robot drives forward 0.10 m. The robot then checks to see if the chemical is still being detected, and if it is, this process of facing upwind and driving forward 0.10 m is repeated. If the chemical is not detected, the robot turns 60° in the direction of the sensor that initially detected the chemical, drives forward 0.10 m, and repeats the check. If the chemical is still not detected, the robot turns 120° away from the

direction of the initial sensor, moves forward 0.20 m, and rechecks for the presence of the chemical. If the chemical scent is still not detected, the robot turns to face upwind and drives in two circles; the first circle is in the direction of the initial sensor, and the second is in the opposite direction. If the chemical scent is lost, the robot turns to face into the wind and waits until the chemical is detected [94].

The *Spiral Surge Algorithm* [100] is similar in principle to *Bombyx Mori Algorithm*. It uses three different strategies to find the odor source. The first of these, which is used to locate the odor plume, involves a large outward spiral search. If the odor is detected, the robot drives upwind by a set distance; this "surge" distance is reset as long as the odor is present. If the sensors fail to detect the odor at the end of the surge, the robot performs a small outward spiral search [100].

The *Multiphase Tracing Algorithm* [98, 101, 102] differs from the previously mentioned algorithms insofar as it was designed for use in environments with non-uniform airflow. The algorithm is composed of five phases: (1) waiting for gas detection, (2) searching for the plume along the concentration gradient, (3) retreating, (4) tracking the plume, and (5) searching for the plume across the wind (Fig. 4.3) [101]. In phase 1, the robot waits for the sensor readings to exceed a threshold value in order to indicate the presence of gas. Then, phase 2 of the algorithm is initiated, and the robot begins searching for the plume along a concentration gradient, neglecting wind information. Assuming that there is a step change in the concentration gradient across the boundary of the plume, the algorithm relies on a threshold to determine if the robot has entered the plume. If the robot is not successful in finding the plume, phase 3 of the algorithm is initiated and the robot retreats/backtracks. During this time, the sensors continue to check for the target gas, and if it is detected, phase 2 is re-initiated. If the checks continue to fail to detect the presence of gas (i.e., indicated by the minimum value being achieved for the four averaged gas sensors), phase 1 is re-instated. However, if during phase 2 the robot is successful in finding and entering the plume, it moves to phase 4 of the algorithm. Here, the robot tracks the plume using an upwind search (i.e., it is instructed to move at a constant angle relative to the wind direction). It also uses the concentration gradient to stay near the center of the plume. If the robot loses the plume in phase 4, it enters phase 5 of the algorithm. This phase includes searching for the plume across the direction of the wind by expanding the margins of the search. If the "plume detection threshold" is not reached after twice searching across the wind direction, the plume is considered lost and phase 2 is re-entered [101].

The *Step-by-Step Progress Method* and *Zigzag Approach* were both tested using the setup designed by Ishida et al. [98]. The robotic platform (Fig. 4.4a)includes four gas sensors and four thermistor anemometric sensors. The gas sensors are spaced 40 mm apart and at 90° to each other (Fig. 4.4b). The anemometric sensors are located directly below the gas sensors and are separated by a square pillar (Figure 4.4b). The odor plume was created by introducing a gas source at one end of a small wind tunnel (0.70 m x 0.80 m x 0.35 m), and a fan pulled air through the opposite end. For the *Step-by-Step Progress Method*, a uniform airflow of ≈0.20 m/s was used and the robot was able to locate the ethanol source at a distance of 0.80 m within 234-240 s [98]. For the *Zigzag Approach*, a uniform airflow of ≈0.12 m/s was used and the robot was able to locate the ethanol source at a distance of 0.80 m within 678-868 s [98].

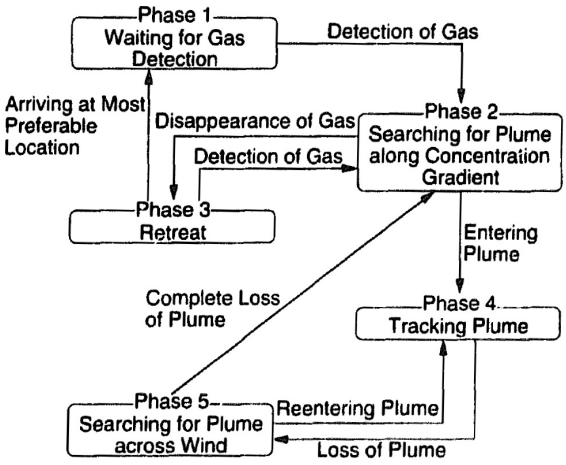
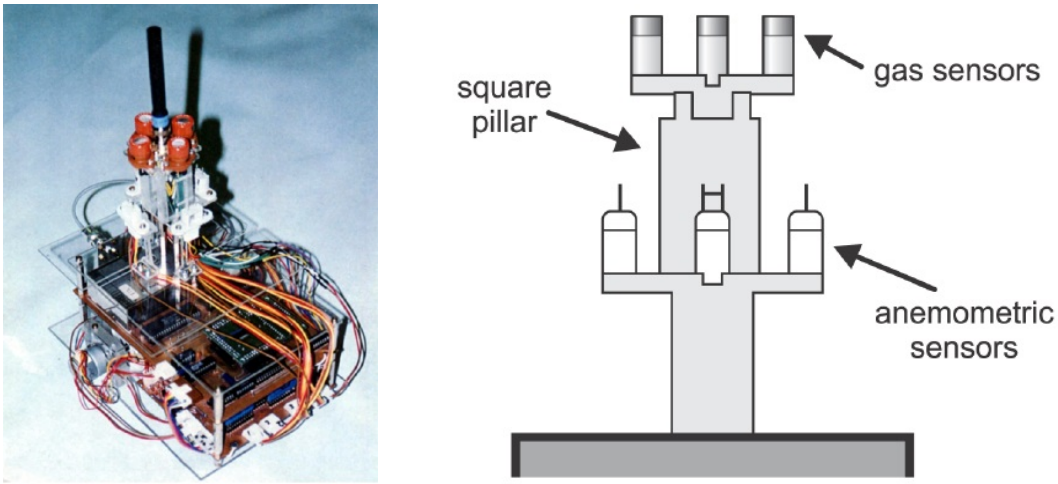


Figure 4.3: Robot Testbed used by Ishida et al. [102]



(a) Robotic Platform

(b) Sensor Probe

Figure 4.4: Testbed used by Ishida et al. [102]

The *Multiphase Tracing Algorithm* also uses the setup designed by Ishida et al. [98]. However, as this algorithm was designed for environments with non-uniform airflow, the testing was completed in a clean room. Airflow was generated by air movement between two supply openings in the ceiling and two exhaust openings near the floor. Ethanol was released near one of the supply openings. For this setup, the wind speed near the source was ≈0.30 m/s, and dropped to ≈0.10-0.30 m/s throughout the search area. The robot was able to locate the alcohol source from a distance of ≈1.5 m [101].

The *Bombyx Mori* and *Dung Beetle* algorithms were tested on the RAT platform [94] using a uniform airflow plume with an air speed of ≈1.5 m/s. For both algorithms, the robot was able to find the alcohol source from a distance of ≈2 m [94].

The *Plume-Centered Upwind Search Algorithm* was tested on a robotic platform that included a wind vane (for determining wind direction) and a single quartz micro balance gas sensor that was located at the front of the base. The algorithm was tested using a uniform airflow plume with an airflow speed of ≈0.30 m/s. The robot was able to locate the alcohol source from a distance of ≈1 m [98].

The *Spiral Surge Algorithm* was tested on a modified Moorebot robot platform [103]. This platform contains four infra-red range sensors for collision avoidance, a single polymer odor sensor (located in the front center of the robot), and a hot wire anemometer (placed in a tube) for wind speed and direction sensing. The direction of the wind was calculated by sampling the anemometer while the robot was slowly rotated 360° [100]. The algorithm was tested using a uniform air plume with an airflow speed of ≈1 m/s. The robot was able to find the alcohol source from a distance of ≈6 m [100].

4.2.3 Actively-Sampled Chemical Sensors

Chemical/odor source detection is commonly needed in indoor environments without strong unidirectional airflow [94]. In these situations, wind speed and direction sensors are of no benefit. Therefore, in order to improve upon the effectiveness of source location as seen with passive setups (Section 4.2.1), the *Gas Sensitive Braitenberg Vehicle* and *2D Odor Compass* employ active airborne chemical sampling setups. Here, fans onboard the robot create local airflow patterns that can be used to determine the location of the odor source.

The *Gas Sensitive Braitenberg Vehicle* uses a Mark III mobile nose and Koala mobile robot (Fig. 4.5 and Fig. 4.6) [102, 104, 105]. The Mark III nose is very similar to the Mark II nose (Section 4.2.1), except it uses two fans in the center of the robot to pull air in through the tubes. The outputs of each set of alcohol sensors are connected directly to the motors using the "Permanent Love" or "Exploring Love" inhibitory Braitenberg vehicle connections [22, 102]. For the "exploration and hill-climbing" strategy, the "Permanent Love" coupling was used, where the right sensor is connected to the right motor and the left sensor is connected to the left motor [22, 104]. For the "exploration and concentration peak avoidance" strategy, the "Exploring Love" coupling was used, where the left sensor was connected the right motor and vice versa [22, 104]. In order to test the robot's ability to locate the odor source, it was placed in an unventilated room at a random position that was at least 1 m from the center of the gas source. The robot was able to consistently locate the source from a distance of ≈1-4 m using either strategy. However, the source could be located from twice the distance when the uncrossed, "exploration and concentration peak avoidance" strategy was used [104].

The *2D Odor Compass* [106] is implemented using rotating stand with a fan and two alcohol sensors that are separated by a plate (note: while this method was not tested on a robotic base, it could be

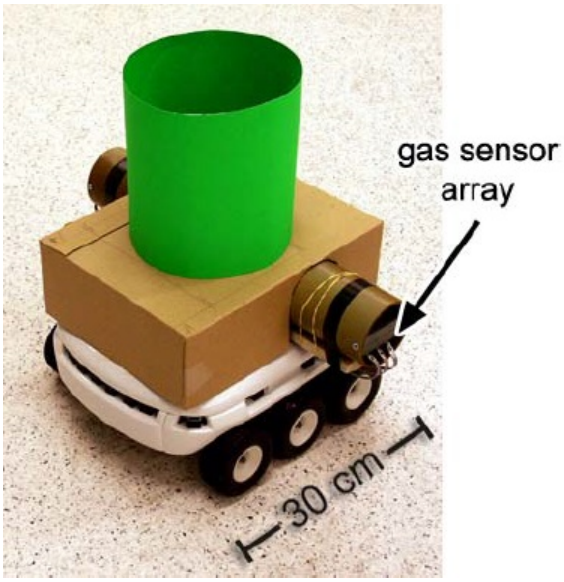


Figure 4.5: Mark III Mobile Nose on a Koala Robot [102]

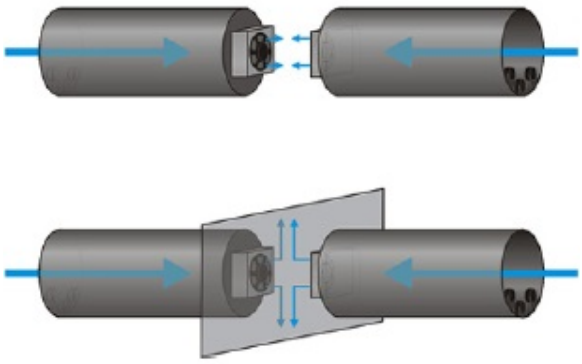


Figure 4.6: Mark III Mobile Nose Schematic [105]

incorporated into a mobile platform). It is meant to behave similarly to a magnetic compass and point towards the odor source. The fan is used to amplify the sensor output when it is aligned with the odorant's airflow. In this way, the stand can be fully rotated in order to estimate the direction to the odor source. Therefore, if the compass is positioned relatively far away from the source, it tends to point towards the plume's center [102, 106]. If, however, the compass is positioned in the plume, it points towards the source [102, 106]. In order to test the compass's ability to locate the odor source, the setup was placed in a clean room environment with non-uniform airflow. Airflow was generated by air movement between

two air supply openings in the ceiling and two exhaust openings near the floor. Ethanol was released near one of the air supply openings [101]. When the direction to the source was determined with the compass in a fixed position, the reading would fluctuate up to 30°. If the odor compass was moved manually in the direction indicated by its reading (in increments of 0.20-0.30 m), it could successfully track the alcohol source. When starting outside of the odorant plume, the compass could locate the source from a distance of ≈1.25 m; this distance was increased to ≈1.7 m when the compass was initially positioned inside of the plume [102, 106].

4.2.4 Problems with Robotic Chemical Sensors

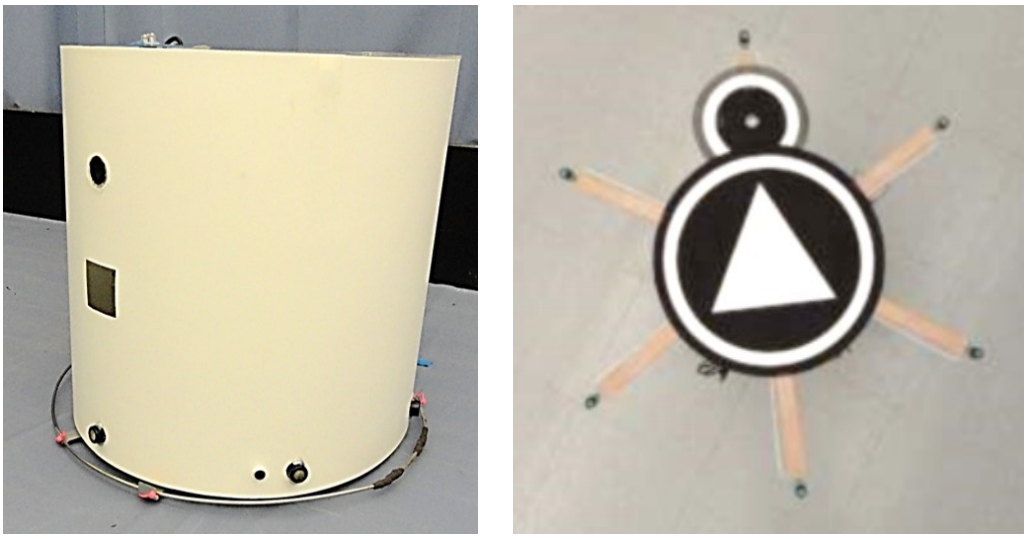
As discussed in the previous sections, there are many approaches that can be used for robotic airborne chemical sensing [28, 94, 96, 98, 101, 102, 107]. While adding a wind direction sensor can increase the effectiveness and efficiency of source determination, it may be unreliable in a dynamic indoor environment. For example, the wind speeds used in these experiments were somewhat high for indoor environments (up to ≈1.5 m/s). For lower and/or variable wind speeds, it may be difficult to determine the direction of the source. Relying solely on chemical sensors presents its own set of difficulties, as the majority of the airborne chemical sensors that are appropriate for use in robotics suffer from poor response time (i.e. 10-150 s). While response time and effectiveness can be improved by using a fan in concert with chemical sensors, the added power requirements and weight must be considered for mobile platforms.

Ideally, a more efficient and reliable method of passive chemical sensing could be developed for dynamic indoor environments. The alcohol homing algorithm developed for the EvBot III attempts to solve this problem. It does not rely on a wind direction sensor to determine the location of the plume, nor does it require an on-board fan. Additionally, there was an effort made to remove hard-coded threshold, turn angle, and drive distance values wherever possible. The remainder of this chapter discusses the design and evaluation of the testbed used to implement the new alcohol homing algorithm.

4.3 Desecription of EvBot III Testbed

4.3.1 Robotic Platform

The EvBot III mobile robotic platform (Chapter 3)was used for testing. Two custom olfactory sensor shields were designed for the EvBot III (Fig. 4.7). The olfactory sensor shields contained six MQ-3 resistive alcohol sensors that are sampled using the National Instruments USB-6009 Daq with onboard 14-bit ADC. In Configuration 1 (Fig. 4.7a), the six alcohol sensors were evenly spaced in 60° increments and mounted flush with the sensor shield 1 inch from the floor. Sensor 0 was centered at the front of the shield, and sensor numbering continued counter-clockwise (looking down on the EvBot). In Configuration 2 (Fig. 4.7b), the alcohol sensors were moved to the top of the EvBot base and positioned to face



(a) Sensor Configuration 1 (side view) (b) Sensor Configuration 2 (top view)

Figure 4.7: EvBot III Olfactory Sensor Shields

up from the floor (i.e., at 90° to the alcohol plume). Additionally, in order to increase differentiation, the alcohol sensors were positioned further apart using "fingers" to hold the sensors in a ring that was twice the diameter of the EvBot.

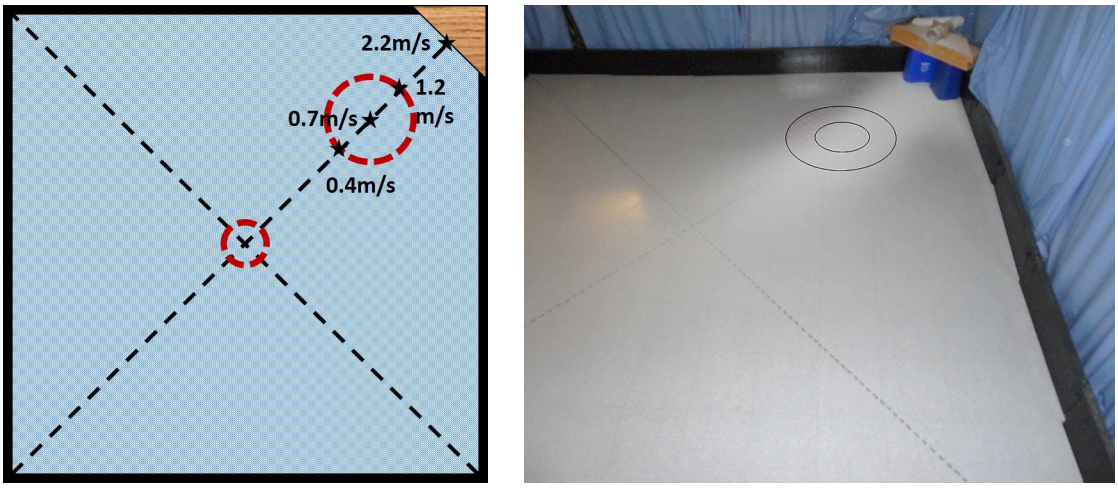
4.3.2 Testing Arena

The experiments were conducted in an indoor arena with very low airflow (Fig. 4.8 and Fig. 4.9). The arena measured 3.9 m x 3.9 m, and was framed by a 0.25 m high wooden barrier. The walls of the arena were formed from a 1.6 m high PVC pipe frame from which blue plastic curtains were hung.

The alcohol source was placed in one corner of the maze (Fig. 4.10). The alcohol plume was formed by using a fan to continuously blow air over a ≈188 ml circular reservoir of alcohol. The alcohol reservoir was placed inside of a tube/tunnel that directed the air over it so that the top of the reservoir was flush with the testing platform (Fig. 4.10). When tests were performed using sensor Configuration 2 (Section 4.3.1), the alcohol source was raised up to sit level with the top of the EvBot. For all experiments, 75.5% grain alcohol was used.

Plume Formation

Three different fans were tested in order to ensure robust plume formation. Initially, a 3 inch DC computer case fan was used, but this fan did not move a sufficient volume of air with enough force to generate a distinct plume in the far half of the arena.



(a) Arena Schematic (wind speeds measured using a Kestrel 4000nv Pocket Weather Tracker)

(b) Plume Overlayed on Arena

Figure 4.8: Arena with Alcohol Plume

Next, a 2 inch electric ducted fan (MP-EPF200) was used. This fan was rated to have a maximum thrust of 73 g at 7.2 V. The strength of the fan (with regard to plume robustness) was tested by moving the EvBot in 0.50 m increments from directly in front of the alcohol source to the far edge of the arena (directly opposite the alcohol source) and measuring the alcohol strength using the EvBot's onboard sensors. After testing the fan at different speeds by varying the supply voltage, a level of 3.3 V was found to produce the largest gradient of alcohol values along the plume. However, the speed of the MP-EPF200 fan became unreliable over time, which affected the airflow of the plume.

The ducted fan was finally replaced with a simple small room fan (Honeywell HT-800). Again, plume robustness was tested in 0.50 m increments along the center of the alcohol plume. The lowest fan setting was found to produce the best gradient, which is described by Equation 4.1:

$$y = 0.4096x^2 - 2.8722x + 5 \tag{4.1}$$

4.4 Experiment 1

4.4.1 Methods

The sensor shield was setup using Configuration 1 (Section 4.3.1). The alcohol sensors were tuned using a potentiometer so that their "no alcohol" voltage was approximately the same. In order to navigate to the alcohol source, a reading was taken from each of the six alcohol sensors. Then, the x- and y-components

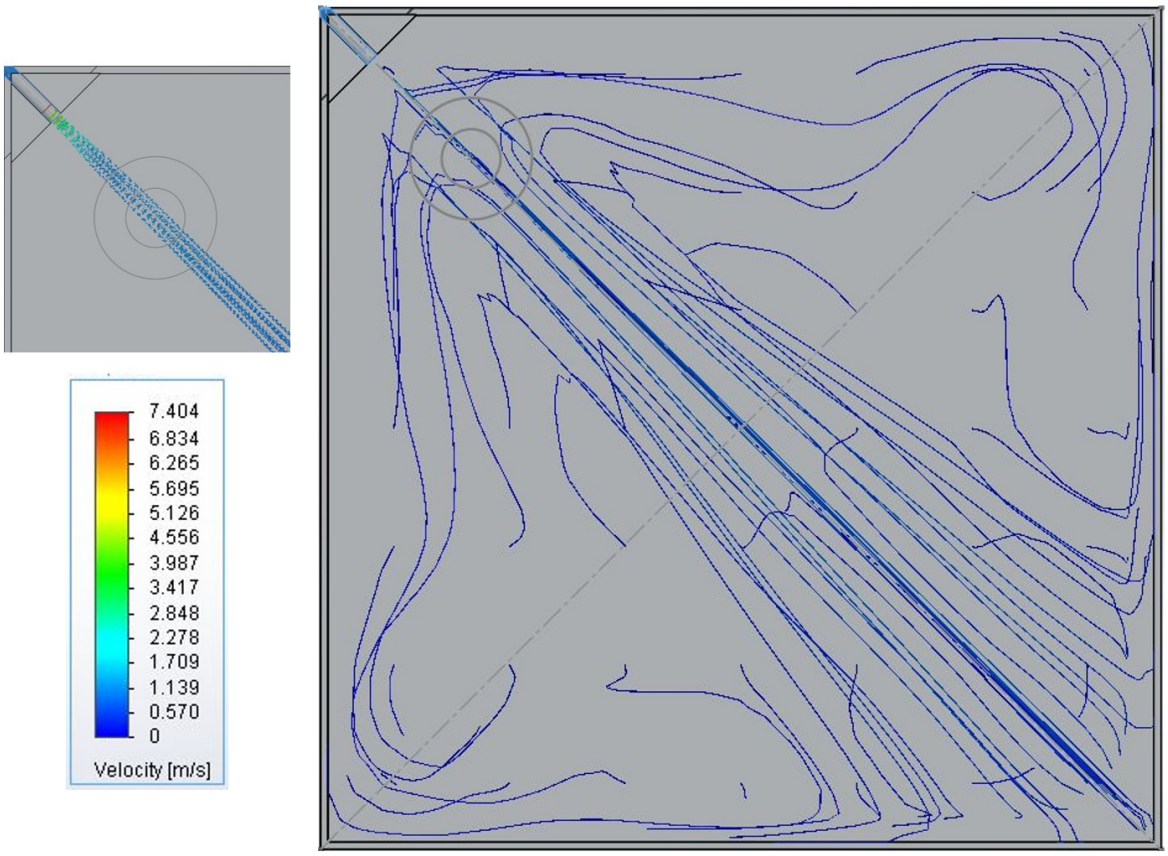


Figure 4.9: Simulation of Alcohol Plume

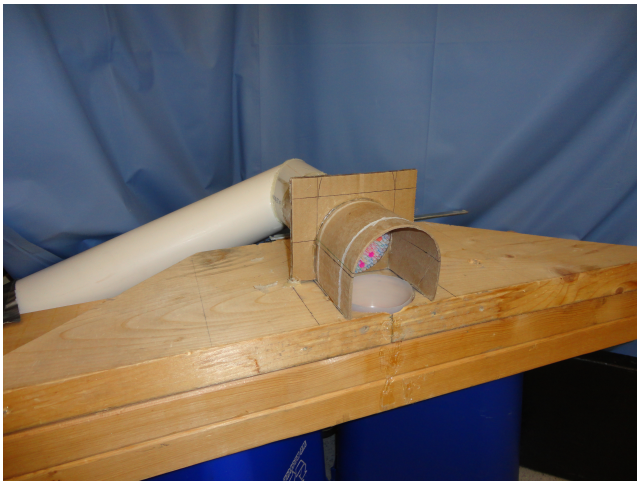


Figure 4.10: Alcohol Plume Source

for each sensor were determined with respect to the alcohol source, and those components summed to determine the resultant vector to the alcohol source. The accuracy of this setup was tested in nine trials where the EvBot was initially placed at Position 2 (i.e., in the center of the arena; Fig. 5.6), and facing directly toward the alcohol source.

4.4.2 Results

The results of the nine trials are summarized in Table 4.2. This algorithm and setup were found to be inaccurate, with the calculated angle to the alcohol source and the actual angle to the alcohol source differing by an average error of 87.01°.

Table 4.2: Sensor Configuration 1 with Vector Summing Homing Algorithm

Run Number	Calculated Angle
1	81.51°
2	67.63°
3	76.47°
4	74.10°
5	31.99°
6	65.34°
7	108.26°
8	135.12°
9	142.71°

4.5 Experiment 2

4.5.1 Methods

It was hypothesized that increasing the distance between the alcohol sensors would lead to greater differentiation between the measurements. This would, in turn, increase the accuracy of the resultant vector by increasing the weight given to those sensors closest to the source. In order to test this, Experiment 1 was repeated using sensor Configuration 2 (Section 4.3.1).

4.5.2 Results

The results of the nine trials are summarized in Table 4.3. Although the error between the actual and calculated angle was less (i.e., 71.99°), this algorithm and setup were still found to be too inaccurate for acceptable navigation and homing.

Table 4.3: Sensor Configuration 2 with Vector Summing Homing Algorithm

Run Number	Calculated Angle
1	70.29°
2	30.11°
3	95.23°
4	94.26°
5	68.17°
6	78.22°
7	18.21°
8	91.42°
9	101.86°

4.6 Experiment 3

4.6.1 Methods

It was hypothesized that a more complex homing algorithm would result in further navigational improvements. Therefore, the homing algorithm was modified to use four readings from each of the six alcohol sensors. The first reading was taken at the initial orientation of the EvBot. The base was then rotated 90°, and another reading taken after 100 s in order to allow the alcohol sensors to stabilize. This process was repeated two more times until the EvBot was rotated back to its initial orientation. The readings were normalized for each sensor using the individual maximum value recorded during the movement sequence (i.e., the readings for Sensor 0 were normalized using the maximum value for Sensor 0, the readings for Sensor 1 were normalized using the maximum value for Sensor 1, etc.). This step was necessary in order to account for differences in the responses of each alcohol sensor at the same concentration. After normalizing the data, the resultant vectors from each of the readings were computed, and the four vectors for each sensor averaged. This resulted in six vectors that described each sensor's estimate as to where the alcohol source was located. The resultant vectors were then averaged in order to determine the final angle to the alcohol source.

This algorithm was tested using Configuration 2 because it proved to be more accurate compared to Configuration 1. Nine trials were again conducted with the EvBot starting at Position 2 (Fig. 5.6), and facing directly toward the alcohol source.

4.6.2 Results

The results of the nine trials are summarized in Table 4.4. This algorithm and setup proved to be much more accurate than the previous two, with an average error of 21.18° between the calculated and actual angle. Using this algorithm, the EvBot was able to repeatably navigate to the alcohol source.

Table 4.4: Sensor Configuration 2 with Sensor Normalization and Vector Summing Homing Algorithm

Run Number	Calculated Angle
1	9.22°
2	29.30°
3	25.38°
4	28.78°
5	36.96°
6	3.10°
7	0.50°
8	9.64°
9	47.78°

4.7 Discussion

Chemical sensing can be used for leak and explosive detection, to locate disaster victims, and to coordinate actions between multiple robots. Robotic airborne chemical sensing is commonly implemented using one of three approaches: (1) using only passively-sampled chemical sensors, (2) using wind direction sensors along with passively sampled chemical sensors, and (3) using actively sampled chemical sensors [28,94,96,98,101,102,107]. Passively-sampled techniques have the advantage of being simple, but suffer from problems related to accuracy. Furthermore, relying solely on chemical sensors presents challenges insofar as the majority of sensors that are well-suited for mobile robot applications suffer from over-specialization and poor response times. While adding a wind direction sensor can help to overcome some of these problems, it may be unreliable in an indoor environment with low airflow.

The *EvBot III Alcohol Homing Algorithm* was presented as a more efficient and reliable method of passive chemical sensing for mobile robots operating in dynamic environments. Here, the readings from six MQ-3 resistive alcohol sensors are taken with the robot rotated in four positions. The readings for each sensor are normalized, and a vector calculated to the alcohol source for each. These vectors are then averaged to determine the angle to the source. The drive distance is determined by an equation describing the alcohol gradient (Equ. 4.1). The *EvBot III Alcohol Homing Algorithm* is an improvement

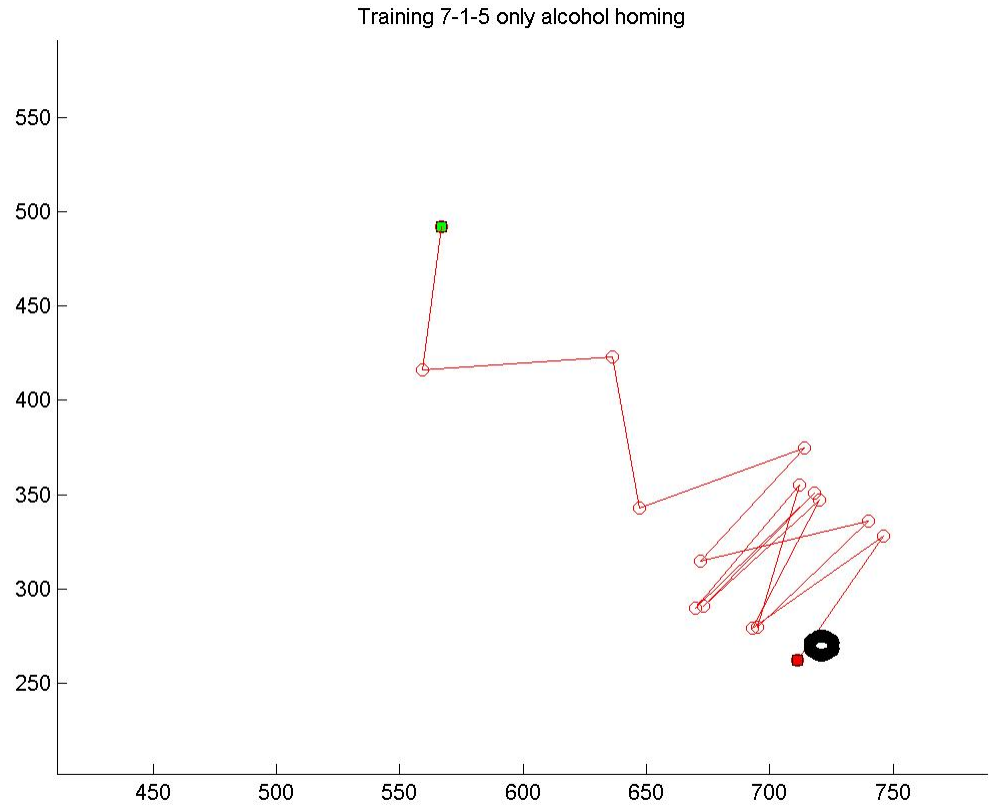


Figure 4.11: Training 7-1-5 EvBot Alcohol Homing Movement

over other algorithms [28, 94, 96, 98, 101, 102, 107] for indoor environments with low airflow because it does not rely on a wind direction sensors, nor does it require an onboard fan. Furthermore, an effort made to remove hard-coded threshold, turn angle, and drive distance values wherever possible in order to make the algorithm more well-suited for dynamic situations.

Two sensor configurations were tested using the EvBot III Alcohol Homing Algorithm. In Configuration 1, the alcohol sensors were evenly spaced in 60° increments and mounted flush with the sensor shield 1 inch from the floor. In Configuration 2, the alcohol sensors were moved to the top of the base on and turned to face upwards (i.e., at 90° to the alcohol plume) on "fingers" that formed a ring that was twice the diameter of the EvBot. Both experiments were conducted in an indoor arena with very low airflow.

When Configuration 1 was tested, the EvBot was unable to home to the alcohol source. With this setup, the sensor that was in the plume and directly pointed to the alcohol source would often have a lower value than the sensors to either side. Because the alcohol sensors were facing directly out

from the shield, the metal oxide sensor was directly in the path of the air flow. This air movement aided in evaporation of alcohol from the sensor, thus reducing its sensed value. Additionally, the sensor opposite the alcohol source (and out of the direct air flow) would occasionally read the highest value. This observation further supports the conclusion that the air movement within the plume was causing alcohol evaporation from the metal oxide.

When Configuration 2 was tested, the EvBot was consistently able to home to the alcohol source from $\approx$2.75 m. Here, the EvBot would first home to the center of the alcohol plume, and then pendulate to the alcohol source. It was noted that the equation used to determine drive distance (Eqn. 4.1) often produced larger distances than were actually needed to efficiently drive to the alcohol source. Although this did not negatively affect the EvBot's ability to locate the alcohol source, homing could be made more effective by collecting more data to validate this equation.

Chapter 5

Odor Sensorimotor Control Software Implementation, Testing, and Analysis

5.1 Abstract

Many of the current robotic systems are application-specific and have difficulty if the environment changes. These controllers often do not scale well with increased task complexity. However, biological systems often exhibit impressive adaptability. Therefore, self-organizing architectures should be incorporated into robotic systems to allow for emergent intelligence and robustness. Towards this end, a flat, fully-connected sensorimotor architecture was developed on the EvBot III platform. Chemical sensing was used as the test application for this network, although it could be included to use any sensing modality. The network was trained to associate an increase in alcohol concentration with an increase in battery charge; it was intended that the robot would learn to navigate up an alcohol plume to the source when it needed to be charged. Although the sensorimotor network was shown to be a good initial step towards robotic reflex behavior, the robot was unable to successfully learn to home to the alcohol source.

5.2 Introduction

Many developers design and program robots with a specific use in mind. While this works well for simple tasks, robots will eventually need to be more adaptable so that they can work on complex projects and remove humans from dangerous environments [7,28,100,107–109]. Robots may also serve to reduce the error that is normally associated with human exhaustion. In order to achieve a robotic system that is both reliable and adaptable, developers must look to nature for solutions that will allow for emergent

intelligence and robustness. This will also help to limit human bias so that the system does not become artificially constrained [3].

The effects of human bias are often overlooked from both a development and experimental standpoint. For example, obstacle avoidance is often considered to be a robotic base behavior. However, it has been shown that obstacle avoidance can be considered as a derivative behavior of homing [24,25]. Also, by reducing experimenter bias, the performance of robotic systems can more easily be quantified and compared. For example, the performance of the EvBot I was compared against a rule-based controller for the tasks of navigating a maze and playing the game Capture The Flag [14, 71, 73–75, 110, 111]. Later, the researchers expressed interest in having a human compete with the learned robotic controllers by using the same range data that was supplied to the EvBots [71]. Although this experiment would provide useful information in terms of the performance of the EvBots, a direct comparison cannot be made because: (1) the computational and memory resources of humans cannot be quantified in the same manner as a robotic controller and (2) a human operator has pre-existing knowledge of what a maze is, and most likely has at least a passing knowledge of the game Capture The Flag.

In order to limit developer and experimenter bias, the EvBot III used a sensorimotor network that was trained solely through environmental interaction. The sensorimotor network was implemented and tested for the task of navigating up an "odorant plume" to a "food source" (i.e., an alcohol concentration gradient to a charging station). Olfaction (chemical sensing) was selected because chemical sensing was the first sense to be evolved by primordial life [28]. For example, the main sensorimotor modality of many computationally- and resource-limited biological systems is chemical sensing [28]. Therefore, these organisms can be more easily compared to a robot versus comparing a robot to a human. Later, once a good correlation base is established for the sensorimotor network, the system can be expanded to include more complex sensorimotor elements.

5.3 Design

The sensorimotor network used for the EvBot III was modeled after one created by Bovet, et. al. at the AI Lab, University of Zurich [3, 66, 93]. The goal of this sensorimotor network is to create a parallel, virtual network that controls the sensorimotor system using weighted connections that reflect correlations between the actual sensorimotor states. Here, each of the sensorimotor elements is connected to a sensorimotor node (Fig. 5.1). This allows the network to develop predictions about the actual state by using correlations between its virtual states and measured states.

The sensorimotor nodes are described by a current state, previous state, current virtual state, state change, and virtual state change. The *current state* represents the current measurement output of the sensorimotor element, and the *previous state* is simply the current stated delayed by one cycle. The *current virtual state* is the anticipated current state of the sensorimotor element. It is calculated by averaging the weighted virtual state changes of all other sensorimotor nodes, and serves as an input to

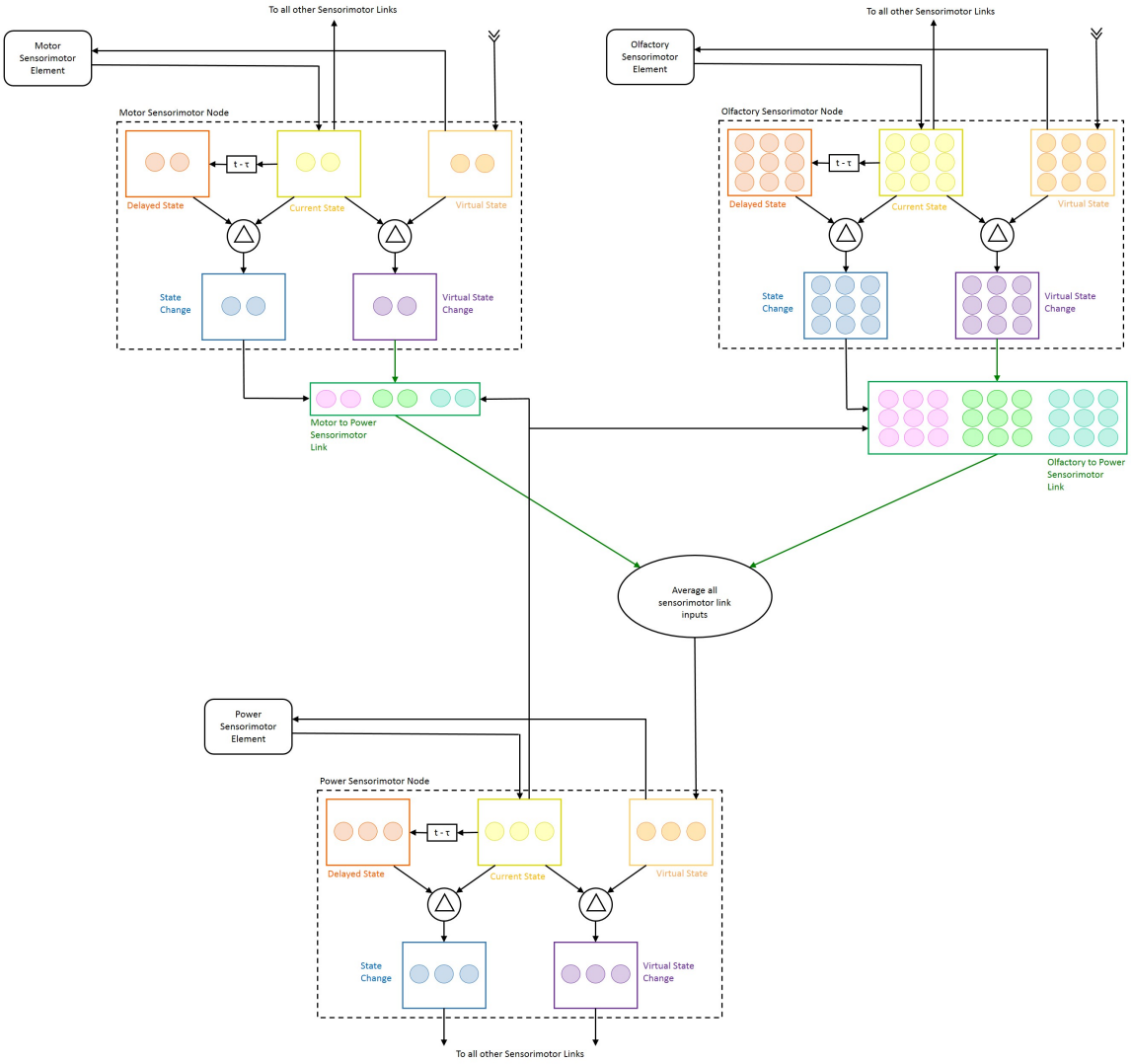


Figure 5.1: Sensorimotor Network

the sensorimotor element. The *state change* represents the difference between the previous state and the current state; for more complex cases, the difference is more involved (i.e., optical flow for vision applications). The *virtual state change* is the difference between the current state and the current virtual state. The virtual state change is passed to the other sensorimotor nodes through weighted links. The weights in these links are updated using the state change and the virtual state change from the source node and the current state of the destination node (see Appendix A). This allows the link to determine and maintain the correlations between the different sensorimotor elements and correctly anticipate the other nodes' virtual current states. Eventually, the weighted inputs from the other sensorimotor nodes will result in the virtual state mirroring the current state for a particular sensorimotor element. Then, the current virtual state can be used to predict the current state using the changes in the state of other sensorimotor nodes.

5.3.1 Overview of the EvBot III Sensorimotor Network

The sensorimotor elements for the EvBot III platform are movement, olfaction, and power. These modalities, as well as their properties, are summarized in Table 5.1.

Table 5.1: Sensorimotor Modalities and Their Components

Name	**Properties**
Motor	Turn angle Drive distance (m)
Power	Current charge Needs charge (true/false) Is charging (true/false)
Olfaction	Calculated distance to alcohol source Calculated angle to alcohol source Max alcohol sensor value Normalized 0° Sensor 0 value Normalized 0° Sensor 1 value Normalized 0° Sensor 2 value Normalized 0° Sensor 3 value Normalized 0° Sensor 4 value Normalized 0° Sensor 5 value

The movement modality was composed of a turn angle and a straight drive distance. In order to simplify the movement commands, the EvBot was first instructed to turn, and then to drive straight. The

RoboClaw motor driver board (Section 3.5.1) used different PID values for both turning and forward movement for each motor.

The power modality consisted of the current charge percentage and two Boolean values to indicate the charge state (i.e., "needs charge" and "is charging"). The current charge percentage was dictated by a virtual power system. The virtual power system was necessary because the capacity of the OceanServer power system (Section 3.4.2) was too large to discharge during training experiments. The virtual power system was programmed to start at an initial charge level that was determined by the number of random movement steps for a given training experiment (Section 5.4.1). The charge level was decreased by 0.5% with every step when not at the charging station, and increased by 1% when at the charging station. Threshold percentages were used to indicate that the robot needed to be charged, as well as that the robot was done charging (Table 5.2). The experimental/charge state (i.e., starting, stopping, and needs charge) was visually indicated to the user through a top-mounted LED array.

The olfactory modality was composed of a calculated distance to the alcohol source, a calculated angle to the alcohol source, an overall maximum alcohol sensor value (0-5V), and normalized sensor values for the initial base orientation of each of the six sensors. The alcohol sensors were sampled using the 14-bit ADC onboard the USB-6009 Daq (National Instruments). An alcohol homing algorithm was used to determine the distance and angle to the alcohol source (Section 4.6.1).

5.3.2 Implementation of the EvBot III Sensorimotor Network

The sensorimotor network for the EvBot III was initially implemented in a Linux environment using C++. The code was later ported to LabVIEW and implemented using a Sony VAIO laptop running Windows 7-64bit. This was done in order to speed up the development process in terms of hardware and software integration, as well as make training/testing to be more straightforward in terms of selecting the appropriate options. In the LabVIEW programming environment, all user-defined inputs and selected outputs (Fig. 5.2) are tied to an element on a user interface (referred to as the front panel). The front panel of the sensorimotor network program included all of the necessary inputs for configuring and running the EvBot's sensorimotor network (Fig. 5.3 and Fig. 5.4).

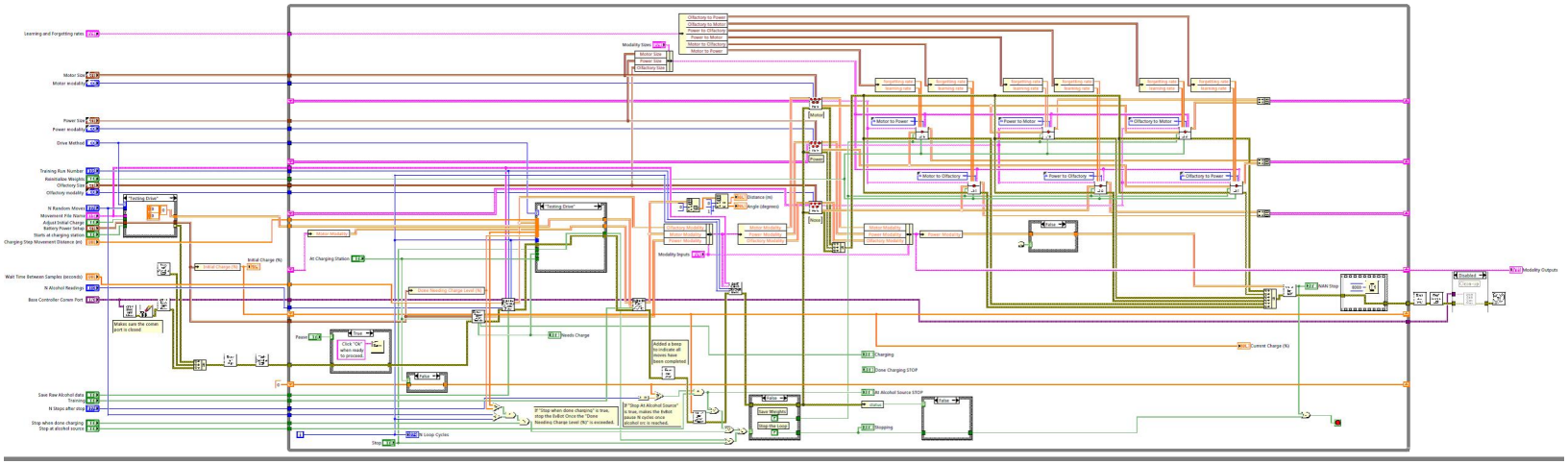


Figure 5.2: EvBot Sensorimotor Network LabVIEW Block Diagram

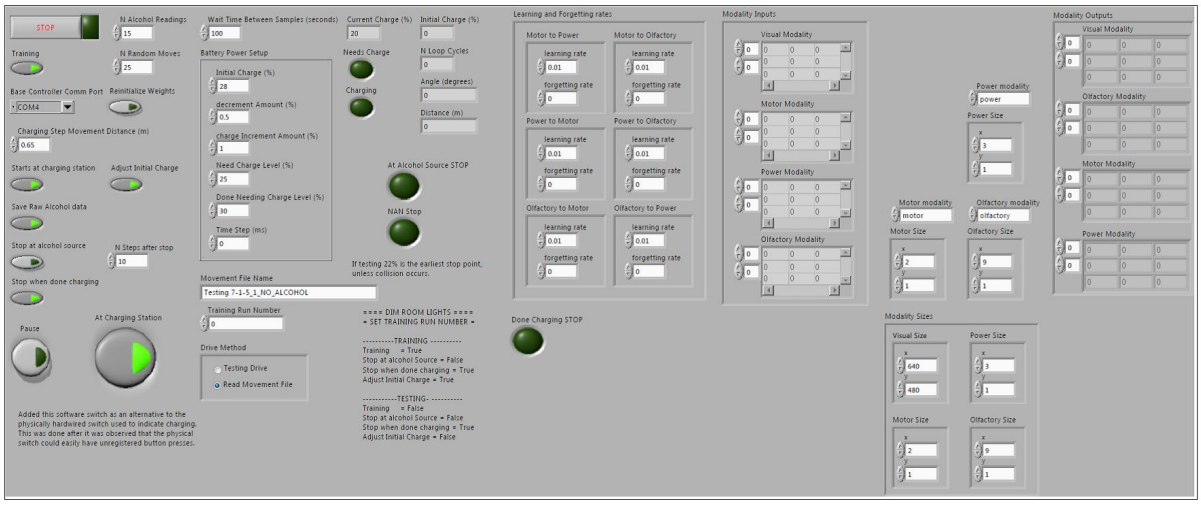


Figure 5.3: Complete EvBot Sensorimotor Network LabVIEW Front Panel

Initializations

The commonly used inputs to the LabVIEW sensorimotor network are described in Table 5.2 (for a complete listing all the inputs, see Appendix B). In order to indicate the current state of the EvBot's sensorimotor network to the developer/experimenter, the front panel outputs described in Table 5.3 were used.

Main Loop

The steps taken in the main loop of the sensorimotor network program were the same regardless of whether it was a training or a testing run. First, the simulated power modality was sampled. If the EvBot was at the charging station, the percent charge was increased by 1%, otherwise the percent charge was decreased by 0.5%. The olfactory sensory system was sampled next. After sampling was complete, the next movement command (distance and turn angle) was determined. If the EvBot was training and the charge level was adequate, the movement command was selected from the Matlab-generated random point (Section 5.4.1) corresponding to the current loop number. If the EvBot was training and the charge level was not adequate, then the next movement command was calculated using the alcohol homing algorithm (Section 4.6.1). Finally, if the network was being tested, then the output of the virtual state of the motor modality's sensorimotor node was used. After the movement command was determined, the EvBot completed the move, and the network was updated.

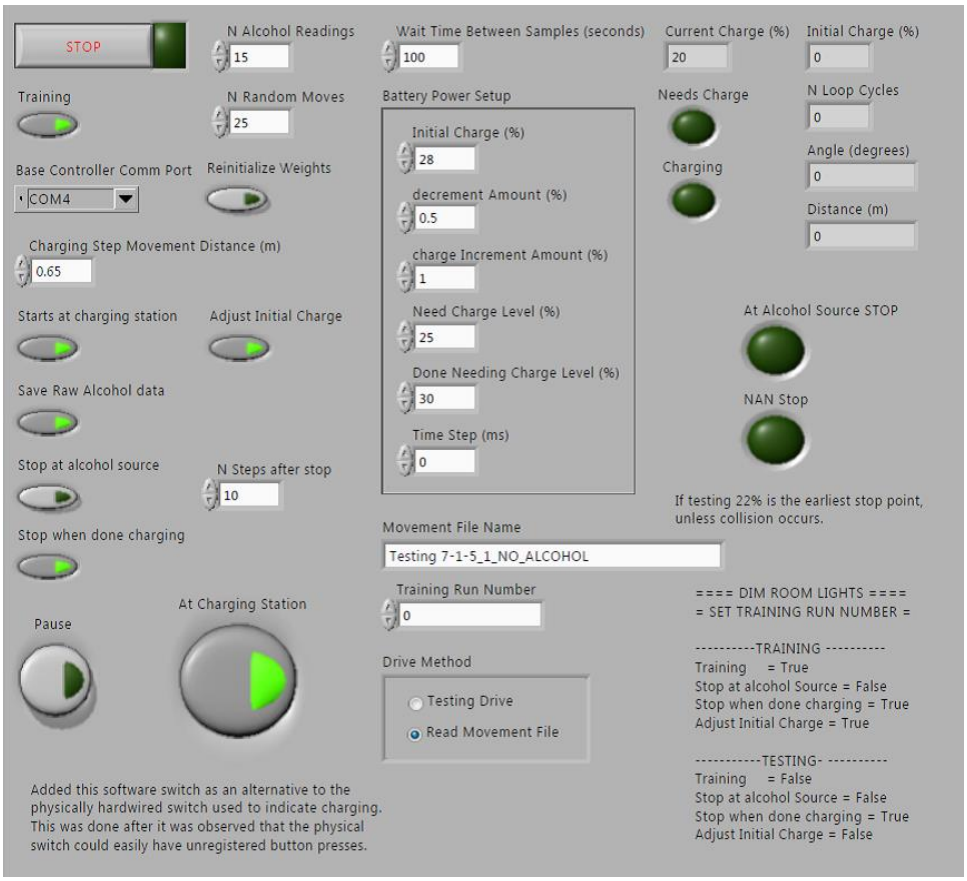


Figure 5.4: Commonly Used Portion of LabVIEW Sensorimotor Network Front Panel

Updating the Sensorimotor Network

As previously mentioned, each sensorimotor node can be described by five components: the delayed state, the current state, the virtual state, the state change, and the virtual state change (Fig. 5.1). These states are updated once per cycle (Fig. 5.5). After the EvBot has executed a movement command, the movement command is set to the current state of the motor element, and the previous current state is moved to the delayed state. Then, the current state of each sensorimotor node is set to the current value of the corresponding element, and the previous current state is moved to the delayed state. The virtual state is then determined by averaging the weighted values of the other sensorimotor nodes. After this, the state change and the virtual state change are calculated. Finally, the virtual state is used as the command to the sensorimotor element.

Table 5.2: EvBot Sensorimotor Network LabVIEW Commonly Used Inputs

Name	**Type**	**Training**	**Testing**	**Description**
Stop	boolean	FALSE	FALSE	TRUE to stop main VI xecuation
Training	boolean	TRUE	FALSE	TRUE if training experiment
Base Comm Port	VISA	COM4	COM4	Comm port used to control base movement
Charge Step Distance	double	0.65	0.65	Largest step (m) robot makes to charging station when "needs charge"
Start Charge Station	boolean	–	–	TRUE if start at charging station
Adjust Initial Charge	boolean	–	–	TRUE if initial charge adjusted for # of random steps and added charge from starting at charging station
Save Raw Data	boolean	TRUE	TRUE	TRUE if raw alcohol values saved to disk
Stop Alcohol Source	boolean	FALSE	FALSE	TRUE if robot stops at alcohol source
N Steps After Stop	int32	10	10	# of cycles to wait at alcohol source before stopping experiment
Stop Done Charging	boolean	TRUE	TRUE	TRUE to stop VI when "Done Need Charge Level" is reached
Pause	boolean	FALSE	FALSE	Pauses VI execution
At Charging Station	boolean	–	–	TRUE if robot is at charging station
N Alcohol Readings	int16	15	15	# of alcohol readings to average
Wait Time	double	100	100	Time (s) to wait after each movement before sampling
N Random Moves	int32	25	25	# of random moves before needing charge
Reinitialize Weights	boolean	–	–	TRUE on 1st training run to set weights to initial zero value
Initial Charge	double	–	–	% initial charge for simulated battery
Decrement Charge	double	–	–	% that charge will decrease after each non-charging cycle
Increment Charge	double	–	–	% that charge will increase after each charging cycle
Need Charge Level	double	–	–	Level (%) at which "Needs Charge" is TRUE
Done Need Charge Level	double	–	–	Level (%) at which battery is considered fully charged
Time Step (ms)	uint32	0	0	Specify cycle time step (ms) (un-used)
Movement File	string	–	–	Pre-generated random movements file
Training Run #	int32	n	n	Run # or movement file
Drive Method	enum	Read movement file	Testing drive	Selection determines if a pre-generated movement file is used for training or if sensorimotor network is used for testing

Table 5.3: EvBot Sensorimotor Network LabVIEW Outputs

Name	Type	Description
Current Charge %	double	The current charge percentage of the simulated battery
Initial Charge %	double	The initial charge of the simulated battery
N Loop Cycles	int32	The current cycle number
Angle (degrees)	double	The currently commanded angle movement
Distance (m)	double	The currently commanded distance movement
Needs Charge	boolean	TRUE if the current charge is below the "Needs Charge Level %"
Charging	boolean	TRUE if "At Charging Station" is TRUE
At Alcohol Source STOP	boolean	TRUE if the VI is stopping due to being at the alcohol source (determined by the measured max alcohol value)
NAN Stop	boolean	TRUE if the VI is stopping due to a sensorimotor value being NAN
Done Charging STOP	boolean	TRUE if the VI is stopping due to beind done charging
Modality Outputs	cluster	Holds the final values of all the modality values

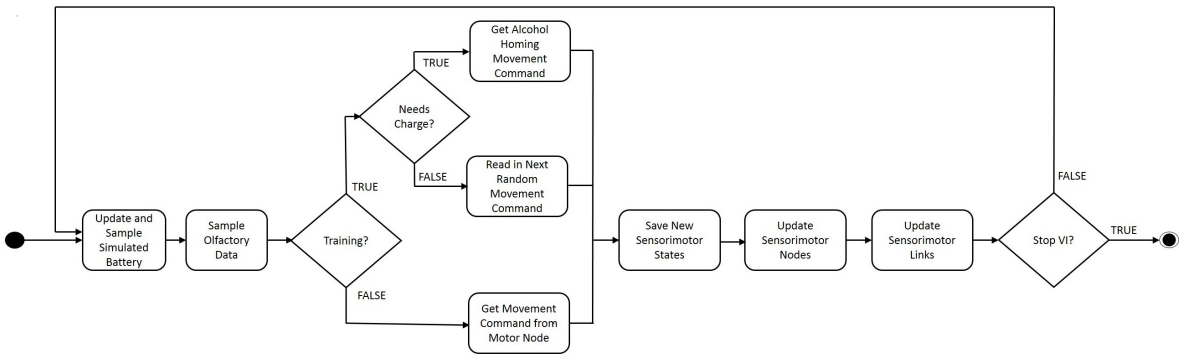


Figure 5.5: Sensorimotor Network Flow Chart

Determining the Weighted Values

The EvBot III network has three nodes corresponding to the movement, olfaction, and power modalities. Each node is described by five states (Fig. 5.1). By designating one node as a destination node and all others as source nodes, the virtual state of the destination node is connected to the virtual state changes of the source nodes using weighted links (Fig. 5.1). The virtual state of the destination node is calculated by averaging the weighted virtual state changes of the source nodes that it is connected to.

Table 5.4: Sensorimotor Link Weight Sizes

		Destination Node		
		Motor	Power	Olfactory
Source Node	Motor		2x3	2x9
	Power	3x2		3x9
	Olfactory	9x2	9x3	

The weights associated with each link are described by a matrix (Table 5.4) whose dimensions correspond to number of properties (Table 5.1) for each modality. The link weights are updated using the state change of the source node and the current state of the destination node. The weights are initialized to zero, and their values increased/decreased based upon correlations between the connected elements (i.e., if two connected elements are active at the same time, then the weight is increased). For complete mathematical description of the weight updates see Appendix A.

5.4 Training and Testing the EvBot III Sensorimotor Network

The sensorimotor network was trained by learning correlations between all of the sensorimotor elements when: (1) the EvBot did not need to be charged and was moving randomly about an arena and (2) the EvBot needed to be charged and was navigating to the alcohol source charging station. Eight sets of training and testing experiments were conducted. For Experiments 1-4, a look-up table was used to command drive angle and distance. Due to problems with drift, PID control was implemented for the remaining experiments. The alcohol sensors were set up using Configuration 1 (Section 4.3.1) for Experiments 1-6, and Configuration 2 (Section 4.3.1) for Experiments 7-8.

The experiments were conducted in the testing arena described in Section 4.3.2. The arena was modified to include a charging area, which was indicated by a circle on the floor that was twice the diameter of the EvBot III (Fig. 5.6). At the beginning of each training or testing run, the reservoir was refilled with fresh alcohol and any remaining alcohol was discarded. Additionally, the reservoir was refilled with alcohol throughout a training or testing run whenever the laptop battery needed to be replaced. A camera with a fisheye lens was centered above the testing arena lens and used to take pictures of the EvBot to track its movement during training and testing.

5.4.1 Random Movement Generation

For Experiments 5-7, the EvBot was commanded to drive randomly for 25 steps. These random movements were calculated offline using Matlab, and read from a file by the EvBot during its random move-

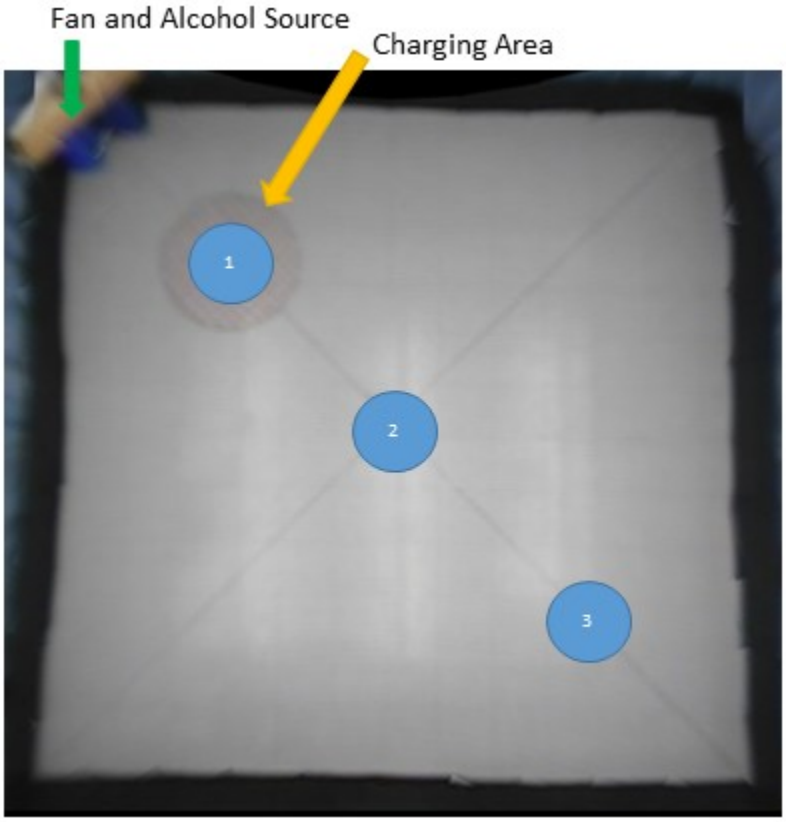


Figure 5.6: Training/Testing Arena

ment portion of training. The Matlab program included the dimensions of the arena and coordinates of the charging station (i.e., the origin of the arena and initial position of the robot) as inputs. Each random movement was calculated and tested to make sure that it did not enter a user-defined buffer zone. If the point was located within the buffer zone, it was simply re-generated. The buffer zone was included to ensure that the accumulated error resulting from the actual EvBot movements did not cause it to run into the arena walls.

5.5 Experiment 1

The random movement correlation building method used by Bovet and Pfeifer for the AMouse project was implemented for the current experiment [3, 66, 93].

5.5.1 Methods

For Experiment 1, the EvBot started at Position 1 (Fig. 5.6), and was pointed directly away from the alcohol source. The initial charge was set to 30%, and the EvBot was driven randomly until a battery charge level of 20% was reached (i.e., 10 steps after the "needs charge" level was achieved). These random movements were calculated on-board the EvBot (Note: the Matlab random movement generator was not used). Eight training cycles were performed.

5.5.2 Results

During testing phase of Experiment 1, the EvBot did not progress towards the alcohol source regardless of whether it needed charge or not. Although the EvBot appeared to pendulate across the alcohol plume when it needed charge (Fig. 5.7), the sensorimotor weight plots (Fig. 5.8) do not show an obvious correlation that would lead to this action.

5.6 Experiment 2

A more tightly controlled training protocol was employed for Experiment 2 in order to encourage the formation of a higher correlation between the EvBot needing charge and driving towards an area of higher alcohol concentration.

5.6.1 Methods

For Experiment 2, the EvBot started at Position 2 (Fig. 5.6), and was pointed directly towards the alcohol source (note: in early experiments, a different motor control strategy was used and the EvBot tended to drift left; it was rotated $\approx 7^\circ$ to the right of center to accommodate for this so that it would drive straight to the charging circle). The initial charge was set to 25% (i.e., "needs charge" was set to TRUE), and the EvBot was programmed to drive straight to the alcohol source. Once at the charging location, the EvBot was programmed to wait 10 cycles. Ten training runs were performed using this setup. Then, the initial charge was set to 30%, and the EvBot was programmed to remain at the starting position until the charge level had depleted to 25% (i.e., "needs charge"). Once the "needs charge" level was achieved, the EvBot was again programmed to drive straight to the alcohol source in one step and remain there for 10 cycles. Ten training runs were performed using this setup.

5.6.2 Results

During the testing phases of this experiment, the EvBot appeared to drive randomly regardless of whether it needed to be charged or not, and was unable to successfully navigate to the alcohol source Fig. 5.9). Looking at the weights (Fig. 5.10) and comparing them to those from Experiment 1 (Fig 5.8),

the Power-to-Olfactory and Olfactory-to-Power weights were larger due to increased number of training runs. Also, the Motor-to-Power and the Power-to-Motor weights were smaller and had reversed signs. However, these differences did not result in the EvBot learning to navigate to the alcohol source when charge was needed.

5.7 Experiments 3 and 4

It was hypothesized that navigation would be improved by increasing the range of alcohol values encountered during training. For Experiments 1 and 2, the EvBot started in the center of the arena (Position 2; Fig. 5.6); this produced a measured range of alcohol values from 2.085V to 2.295V, for a difference of 0.210V. For Experiments 3 and 4, the EvBot started at the farthest corner of the arena from the alcohol source (Position 3; Fig. 5.6); this produced a range of alcohol values from 2.085V to 2.586V, for a difference of 0.501V. This increased the measured range of alcohol values over two-fold.

5.7.1 Methods

For Experiment 3, the EvBot started at Position 3 (Fig. 5.6), and was pointed directly towards the alcohol source (note: because the EvBot tended to drift left when driving straight, it had to be moved $\approx$15° to the right of center so that it would end in the charging circle). This position was selected in order to create a large difference between the starting and ending alcohol concentration (i.e., to increase the correlation between alcohol concentration and "Is Charging"). The initial charge was set to 30%, and the EvBot was programmed to remain at the starting position until the charge level had depleted to 25% (i.e., "Needs Charge"). Once this level was achieved, the EvBot was again programmed to drive straight to the alcohol source in one step and remain there for 10 cycles. Ten training runs were performed.

With Experiment 4, turning was incorporated into the training in order to build correlations between the alcohol concentration and the direction of movement. The same setup and protocol were used as for Experiment 3, except the EvBot was pointed at different orientations to start the training run: +45°, -45°, +90°, -90°, and 180°. Again, the initial charge was set to 30%, and the EvBot was programmed to remain at the starting position until the charge level had depleted to 25%. Then, the EvBot was programmed to rotate to face the alcohol source, and drive straight to the source in one step and remain there for 10 cycles. Ten training runs were performed.

5.7.2 Results

During the testing phases of Experiments 3 and 4, the EvBot proved unable to successfully home to the alcohol source. Figures 5.11-5.13 show the EvBot's movement pattern when tested from Position 3 and (1) pointing directly towards the alcohol source (Fig 5.11), (2) pointing $\approx$45° from the alcohol

source (Fig 5.12), and (3) ≈1m closer to the alcohol source (from position 3) and pointing 180° from the alcohol source (Fig 5.13).

For Experiment 3, the EvBot waited until it needed to be charged, and then drove straight regardless of where it was in relation to the the alcohol source. Looking at the sensorimotor weights for Experiment 3 (Fig 5.14), the weights for Power-to-Motor and Olfactory-to-Motor were nearly zero. Additionally, the Power-to-Olfactory and Olfactory-to-Power show similar amplitude and distribution compared to those from Experiments 1 and 2.

Turning was added to the protocol for Experiment 4 in order to avoid biasing the network to build correlations relating driving straight to needing charge. However, during the testing phase of Experiment 4, the EvBot continued to drive straight regardless of where the alcohol source was whenever it needed to be charged. Looking at the sensorimotor weights for Experiment 4 (Fig 5.15), similar distributions of Power-to-Olfactory and Olfactory-to-Power are seen across Experiments 1-4. The amplitude of the Power-to-Olfactory and Olfactory-to-Power weights is larger than Experiments 1 and 3, but similar to Experiment 2. This is likely related to the fact that twice as many training runs were used in Experiments 2 and 4 compared to Experiments 1 and 3. Additionally, the Power-to-Motor and Olfactory-to-Motor weights did not increase significantly from Experiment 3. Taking these observations into consideration, the behavior observed in the testing phase of Experiment 4 can be explained, although it is puzzling that the strongest drive correlation is seen in the Motor-to-Power weights and not in either the Power-to-Motor or Olfactory-to-Motor weights.

5.8 Experiment 5 and 6

The protocol adopted for Experiments 5 and 6 incorporates the random movements used in Experiment 1 with the directed movements used in Experiments 2-4. It was hypothesized that more generalizable correlations could be built if the robot moved randomly about the maze environment before needing charge versus limiting the robot to one location. However, directed movement was used once the robot needed to be charged in order to guide correlations for alcohol homing.

5.8.1 Methods

For Experiment 5, the EvBot started at Position 1 (Fig. 5.6), and was pointed directly away from the alcohol source. The EvBot was programmed to move randomly for 25 steps (Section 5.4.1), and the initial charge adjusted so that the "needs charge" level would be achieved after these 25 steps. Due to the movement error accumulated during these 25 steps, the LabVIEW VI execution was paused and the desired turn angle and distance measured and entered. The EvBot was then programmed to orient to face towards the alcohol source and drive straight for 5 equi-spaced steps. Three training runs were performed before the experiment was halted due to errors in the pre-generated movements (i.e., the

buffer zone was not large enough, causing it to run into the arena wall; it was later increased to 2x the diameter of the EvBot).

The same setup was used for Experiment 6, except the random movements were corrected to prevent the EvBot from hitting the wall of the arena. Ten training runs were performed using this setup.

5.8.2 Results

During the testing phases of Experiment 6, the EvBot proved unable to successfully home to the alcohol source. In this case, the EvBot would drive backwards until it needed to be charged, and then it would drive straight (i.e., the test was started with an initial charge of 26.5%, resulting in two movements before the EvBot would need to be charged). This movement was independent of the presence or direction of alcohol. Two test runs were performed with the EvBot pointing towards the charging circle, one with alcohol (Fig 5.16) and one without alcohol (Fig 5.17). Another test run (with alcohol) was performed with the EvBot pointing 90° away from the charging circle (Fig 5.18). The testing runs were stopped when the EvBot made contact with the arena wall.

Looking at the weights from Experiment 6 (Fig 5.19), the correlations that motivate the desire to drive are Power-to-Motor and Olfactory-to-Motor. Within those, "drive distance" was the most highly correlated with "needs charge". While this would not be a factor when the EvBot's power was sufficient ("needs charge" is 0), it would be the dominant factor when the EvBot needed to be charged ("needs charge" is 1). This observation would explain whey the EvBot drove straight regardless of its orientation to the alcohol source when it needed to be charged.

5.9 Experiment 7

As manually driving the EvBot straight toward the alcohol source had not proven to be successful, an alcohol homing algorithm (Section 4.6.1) was developed and used during training when the EvBot needed to be charged. It was hypothesized that this would enable meaningful correlations to be built for homing if the EvBot actively used the alcohol sensors to navigate to the alcohol source.

5.9.1 Methods

For Experiment 7, the EvBot started at Position 1 (Fig. 5.6), and was pointed directly away from the alcohol source. The EvBot was programmed to move randomly for 25 steps (Section 5.4.1), and the initial charge adjusted so that the "needs charge" level would be achieved after these 25 steps. After these 25 steps, the *EvBot III Alcohol Homing Algorithm* (Section 4.6.1) was used to guide the EvBot to the alcohol source. The drive distance was calculated by scaling a user-defined allowable driving range (i.e., 0.20-0.65 m) by the estimated distance to the alcohol source. This estimated distance was expressed as a percent, with 100% representing the furthest point from the alcohol source. The estimated

distance was determined from a curve fit of the alcohol concentration gradient (Equation 4.1, Section 4.3.2). When the EvBot reached the charging station, it was programmed to remain there for 10 cycles. Seven training runs were performed.

5.9.2 Results

Although the EvBot could successfully navigate to the alcohol source when explicitly told to use the *EvBot III Alcohol Homing Algorithm*, the EvBot was unable to successfully home to the alcohol source when driven using the learned correlation weights. The testing phase of Experiment 7 was conducted with the EvBot starting from Position 2 (Fig. 5.6). The results are summarized in Figures 5.20-5.23, with the EvBot pointing directly toward the charging circle when alcohol was present (Fig. 5.20) and when alcohol was not present (Fig. 5.21), as well as pointing directly away from the source when alcohol was present (Fig 5.22) and when it was not present (Fig. 5.23). In all cases, the EvBot initially drove straight, and then continued to drive straight after it needed to be charged regardless of the presence or location of alcohol.

Looking at the weights from Experiment 7 (Fig 5.24), the correlations that motivate the desire to drive are Power-to-Motor and Olfactory-to-Motor. Within those, "drive distance" was the most highly correlated with "alcohol sensor 2". With this high Olfactory-to-Motor weight, any value received from alcohol sensor 2 would overpower any other weights that directly connect to the motor sensorimotor element (i.e. the EvBot base) and drive the base. Also, the weights seen in Motor-to-Power, where "needs charge" and "drive angle" have a high positive correlation, and in Power-to-Motor, where "needs charge" and "drive angle" have a strong negative correlation, could be explained by the zigzag approach exhibited by the EvBot III alcohol homing algorithm.

5.10 Experiment 8

5.10.1 Methods

The EvBot started at Position 2 (Fig. 5.6), and was pointed directly toward the alcohol source. The initial charge was set to 25% (i.e., "needs charge" was set to TRUE), and the *EvBot III Alcohol Homing Algorithm* (Section 4.6.1) was used to guide the EvBot to the alcohol source. When the EvBot reached the charging station, it was programmed to remain there for 10 cycles. Six training runs were performed.

5.10.2 Results

The testing phase of Experiment 8 was conducted with the EvBot starting from Position 2 (Fig. 5.6). The results of these tests when the EvBot was (1) pointing directly toward the alcohol source with alcohol present (Fig. 5.25), (2) pointing directly away from the source when no alcohol was present (Fig 5.26), (3) and pointing 90° away from the charging location when no alcohol was present (Fig 5.27). In all testing cases, the EvBot initially drove straight, and then continued to drive straight after it needed to be charged regardless of the location or presence of alcohol.

Looking at the weights from Experiment 8 (Fig 5.28), they are very similar to those for experiment 7. Again the correlations that motivate the desire to drive are Power-to-Motor and Olfactory-to-Motor. Within those, "drive distance" was the most highly correlated with "alcohol sensor 2". With this high Olfactory-to-Motor weight, any value received from alcohol sensor 2 would overpower any of the other weights that directly connect to the motor node and drive the base. Also, the weights seen in Motor-to-Power, where "needs charge" and "drive angle" have a high positive correlation, and in Power-to-Motor, where "needs charge" and "drive angle" have a strong negative correlation, could be explained by the zigzag approach exhibited by the EvBot II alcohol homing algorithm.

5.11 WEKA

The WEKA Data Mining and Machine Learning software [112, 113] was used to generate a comparison of the information quality as it related to the ability to learn the direction to the alcohol source from the recorded sensor data with the sensorimotor network. In order to perform this analysis, all of the sensor data was saved for each of the runs. Then, the actual angle to the alcohol source was manually determined from the visual recordings of the experiment. The measured angles to the alcohol source at each location were partitioned into 12 sectors: north, north-north-east, east-north-east, east, east-south-east, south-south-east, south, south-south-west, west-south-west, west, west-north-west, and north-north-west. The training data was obtained from the time periods when the EvBot was actively tracking to the alcohol source. Although the J48 decision tree was the most accurate method of determining the direction to the alcohol source using only the normalized alcohol sensor values, it was only able to successfully locate the source 25% of the time.

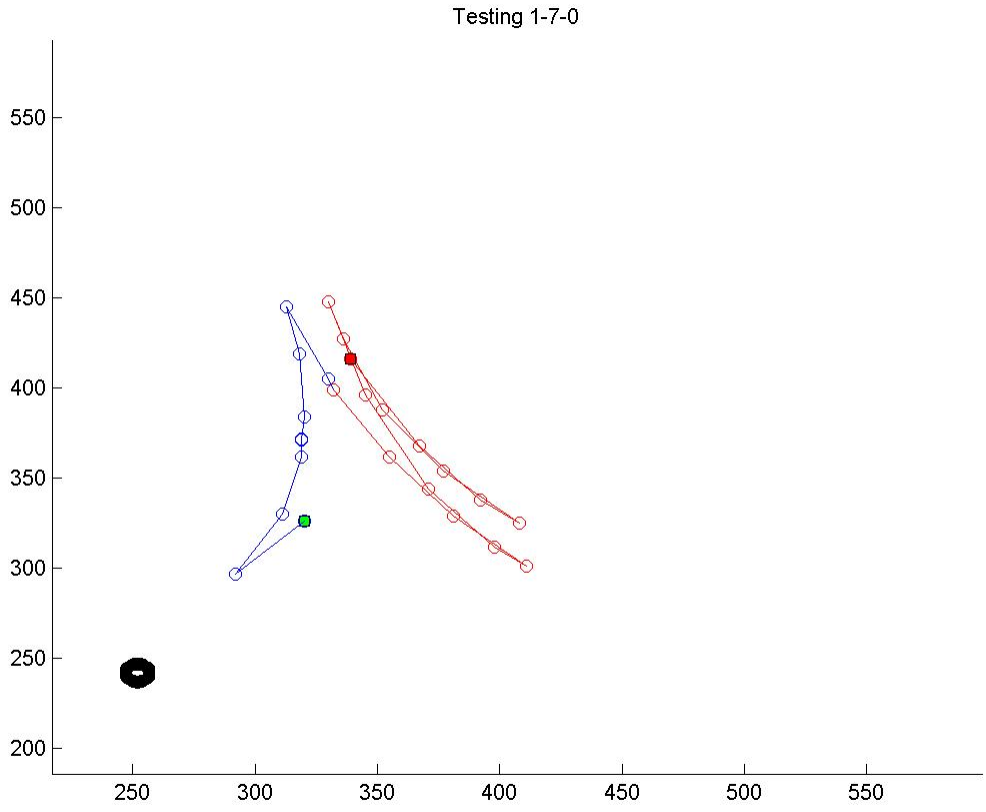


Figure 5.7: Testing 1-7 EvBot movement

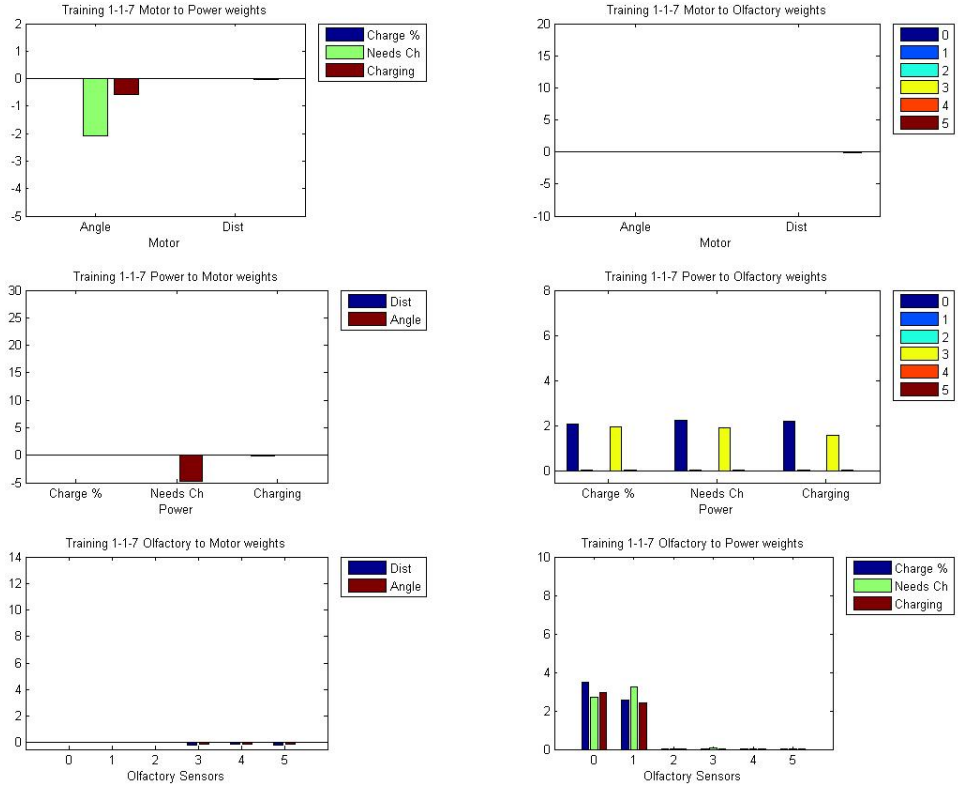


Figure 5.8: Training 1-7 Sensorimotor Weights

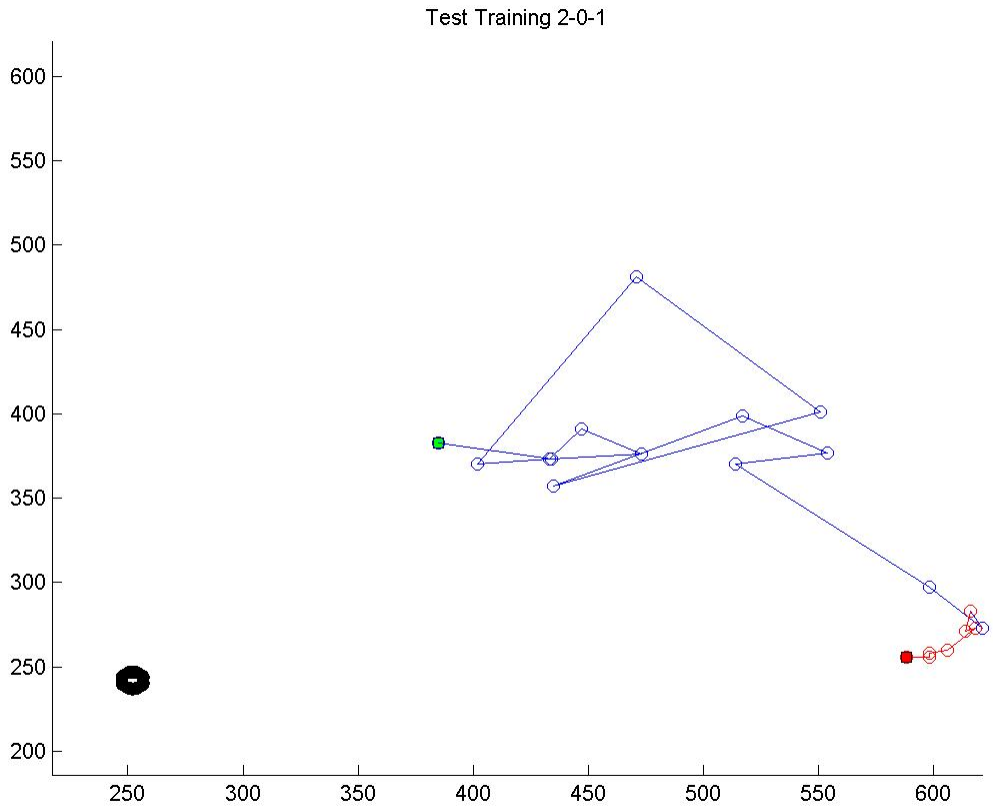


Figure 5.9: Testing 2-1 EvBot movement

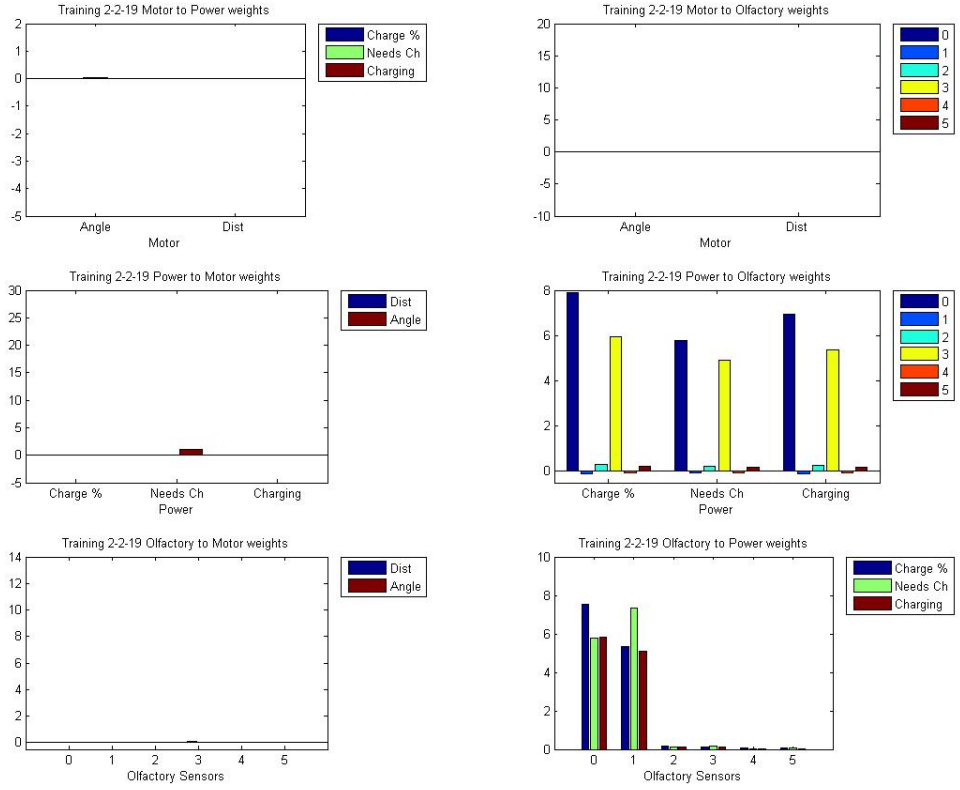


Figure 5.10: Training 2-2-19 Sensorimotor Weights

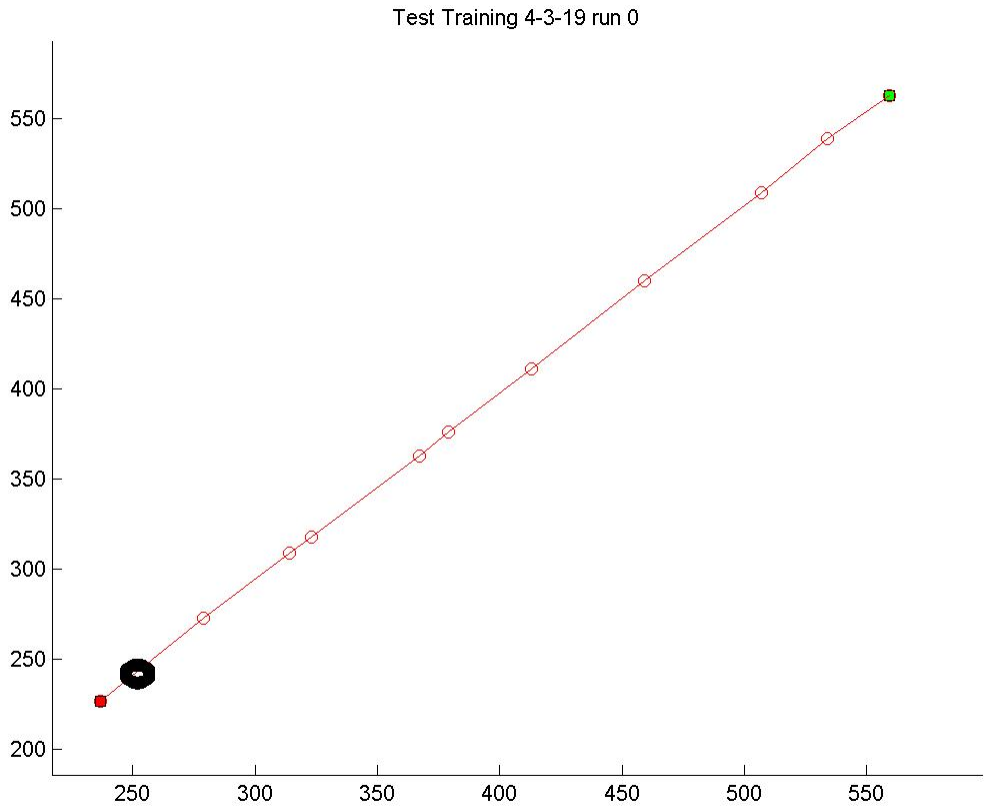


Figure 5.11: Testing 4-3-19, run 0, EvBot movement

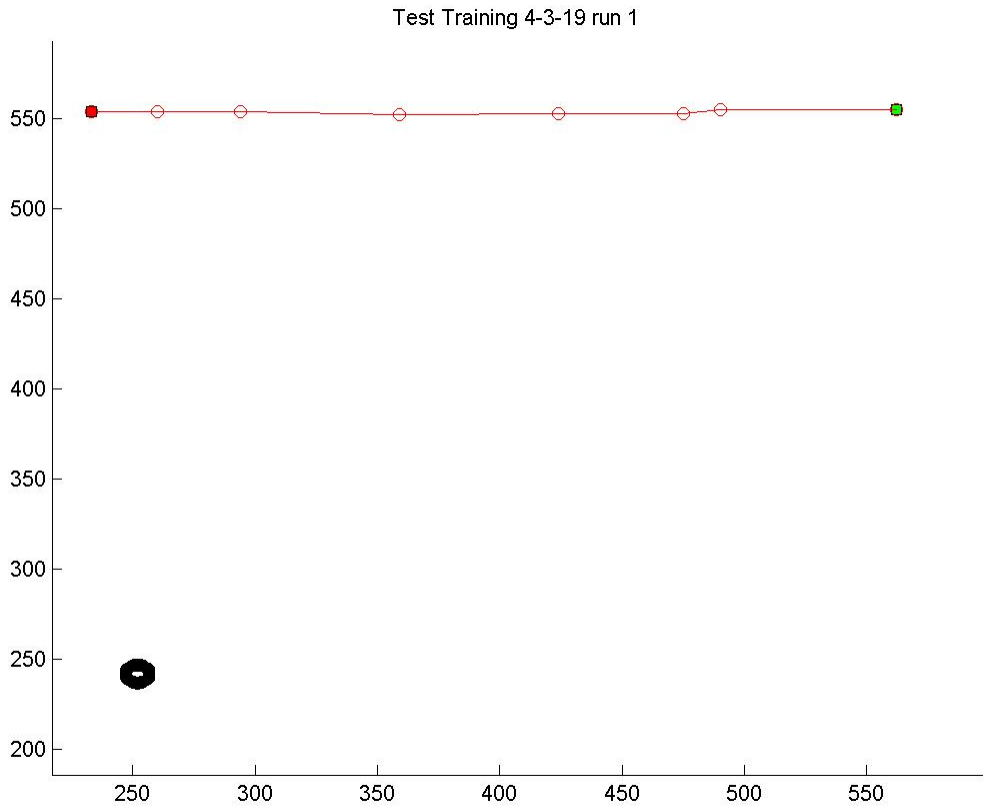


Figure 5.12: Testing 4-3-19, run 1, EvBot movement

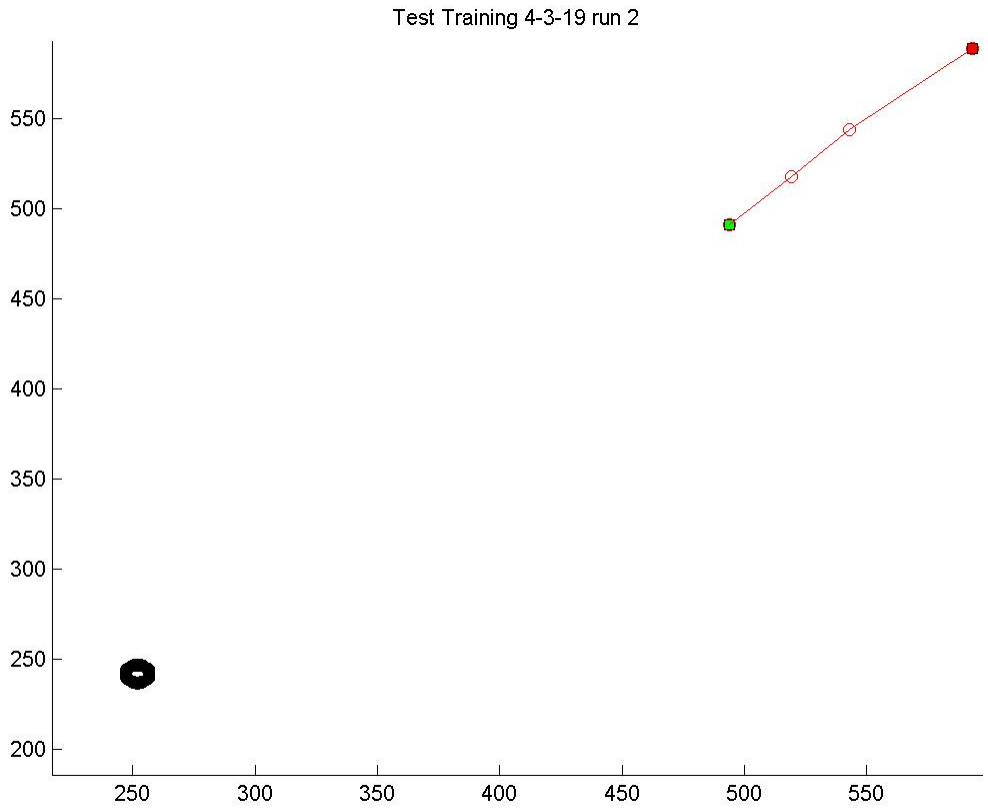


Figure 5.13: Testing 4-3-19, run 2, EvBot movement

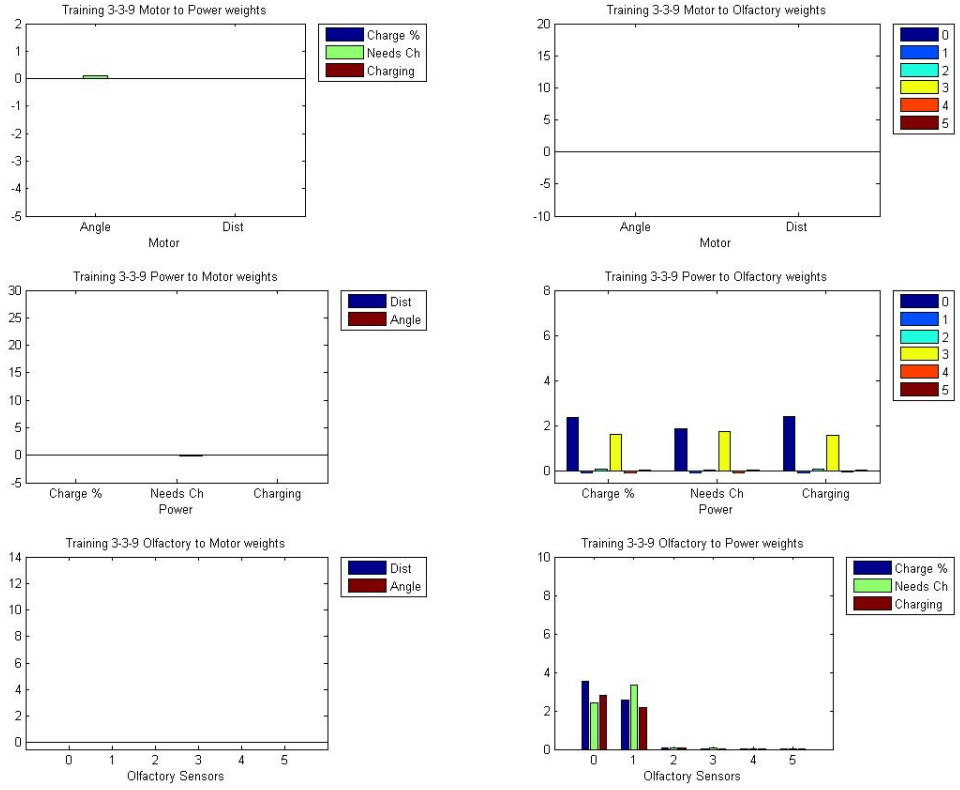


Figure 5.14: Training 3-3-9 Sensorimotor Weights

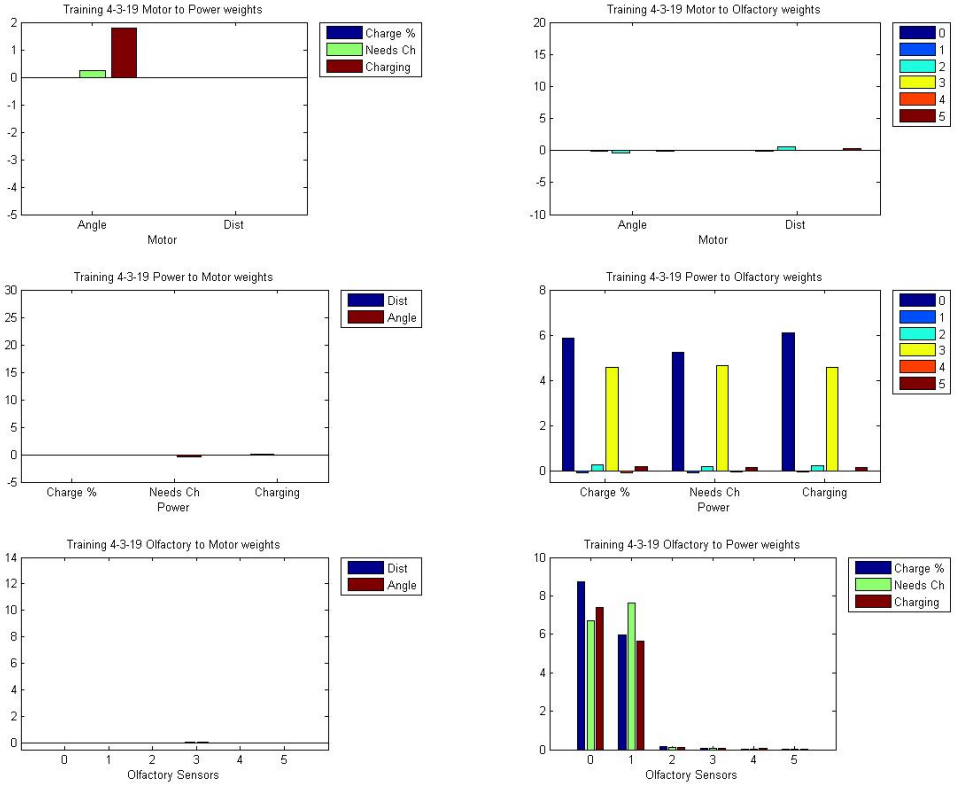


Figure 5.15: Training 4-3-19 Sensorimotor Weights

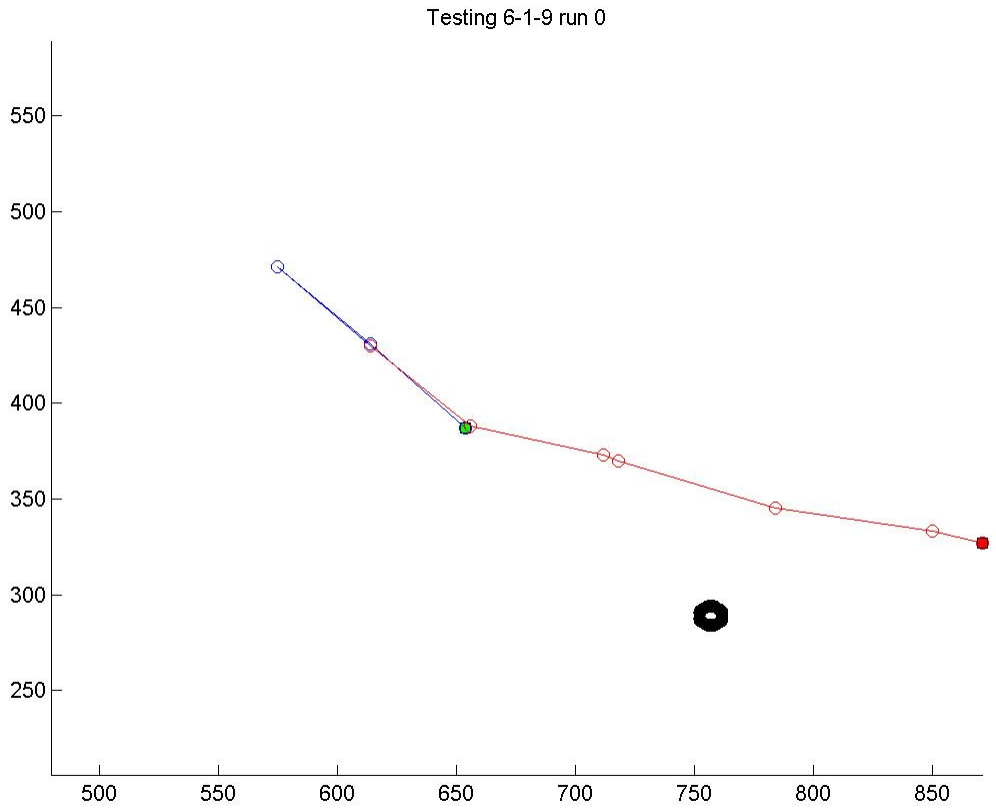


Figure 5.16: Testing 6-1-9, run 0, EvBot movement

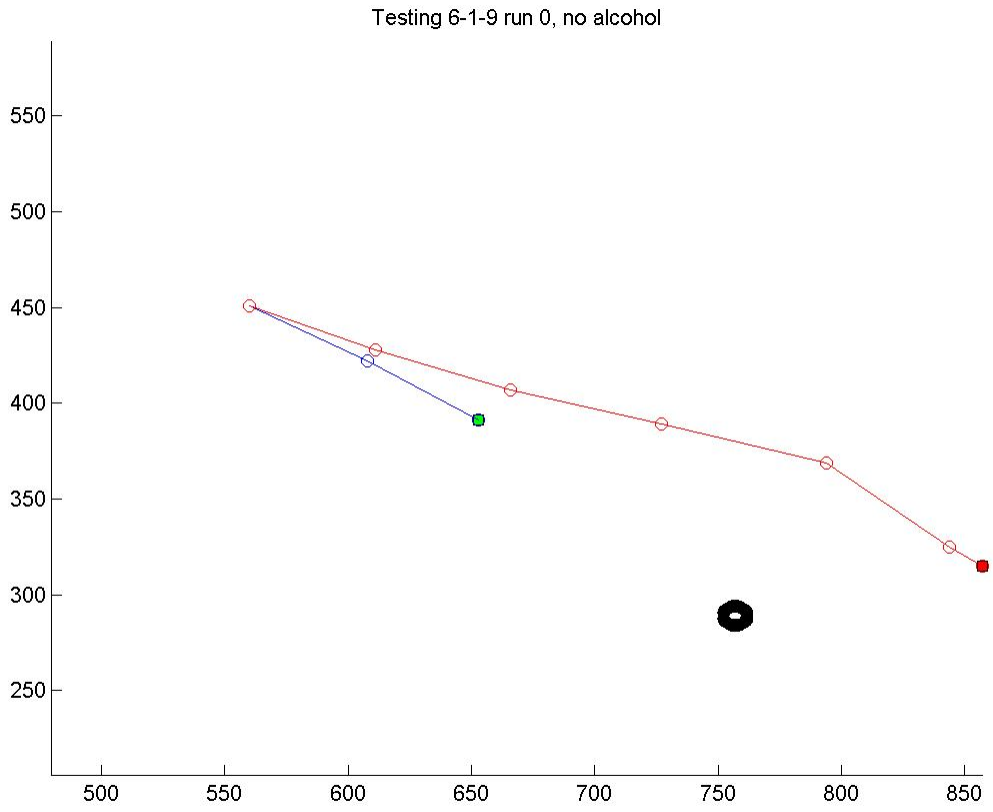


Figure 5.17: Testing 6-1-9, run 0 no alcohol, EvBot movement

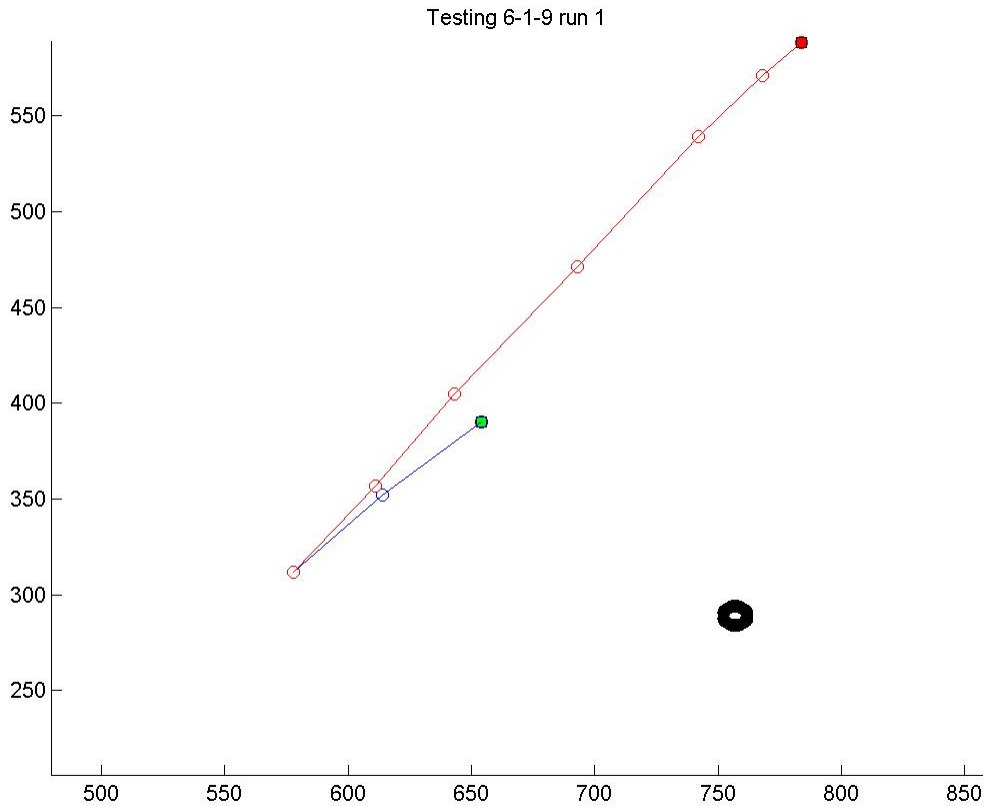


Figure 5.18: Testing 6-1-9, run 1, EvBot movement

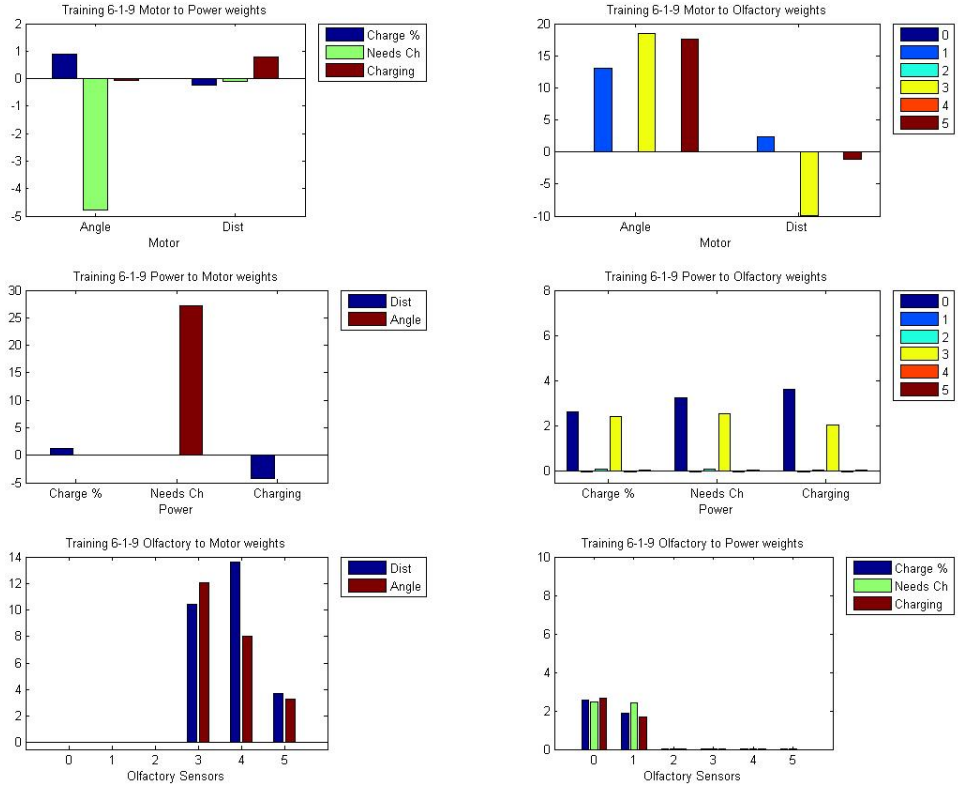


Figure 5.19: Training 6-1-9 Sensorimotor Weights

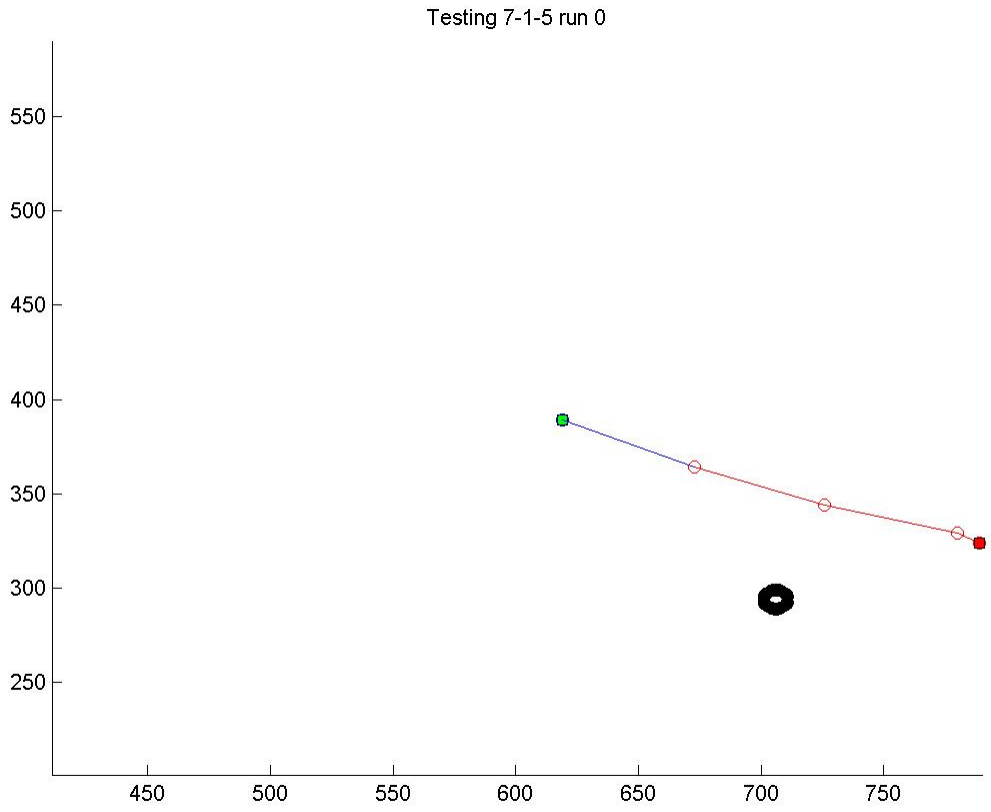


Figure 5.20: Testing 7-1-5, run 0, EvBot movement

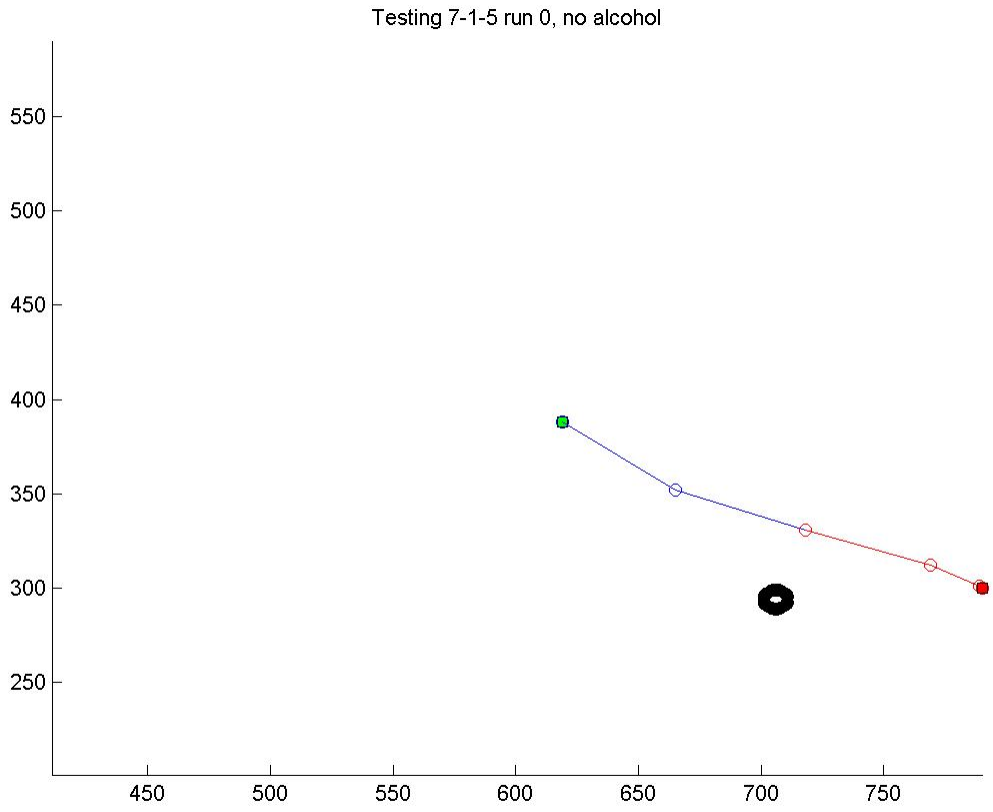


Figure 5.21: Testing 7-1-5, run 0 no alcohol, EvBot movement

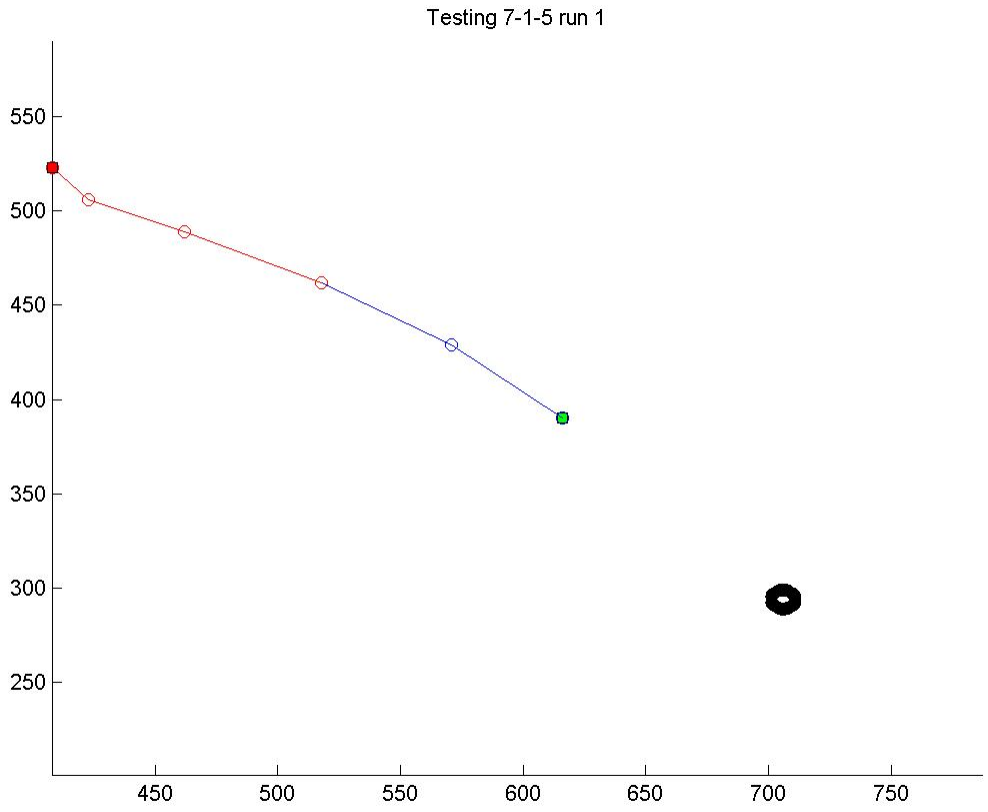


Figure 5.22: Testing 7-1-5, run 1, EvBot movement

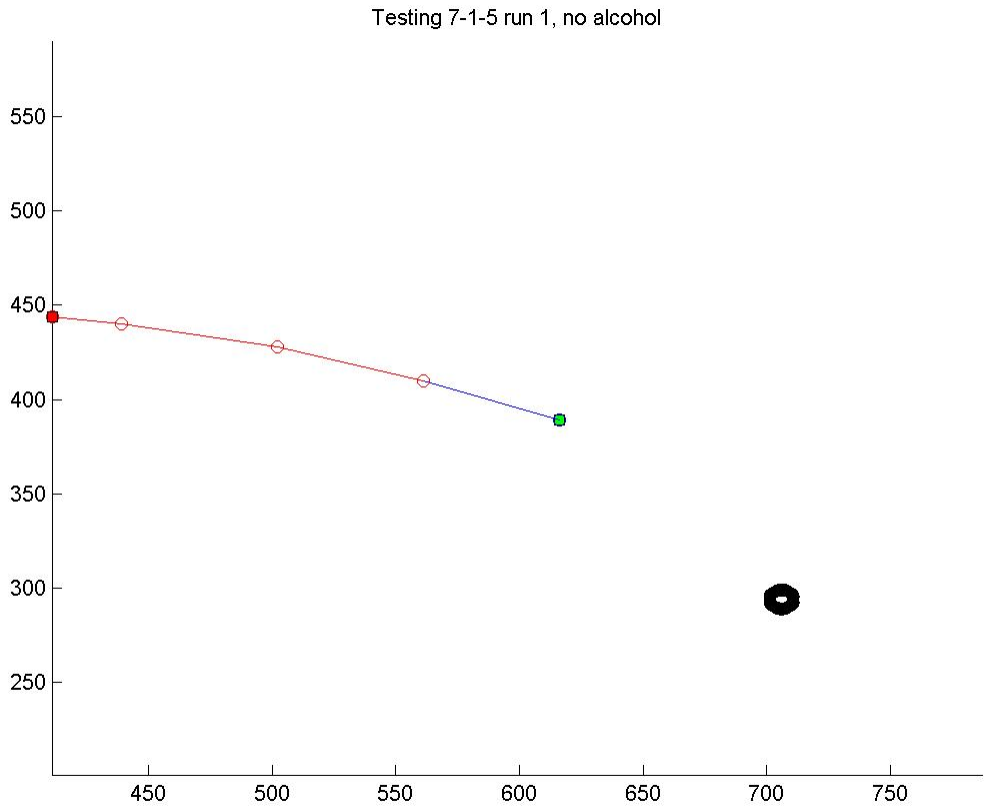


Figure 5.23: Testing 7-1-5, run 1 no alcohol, EvBot movement

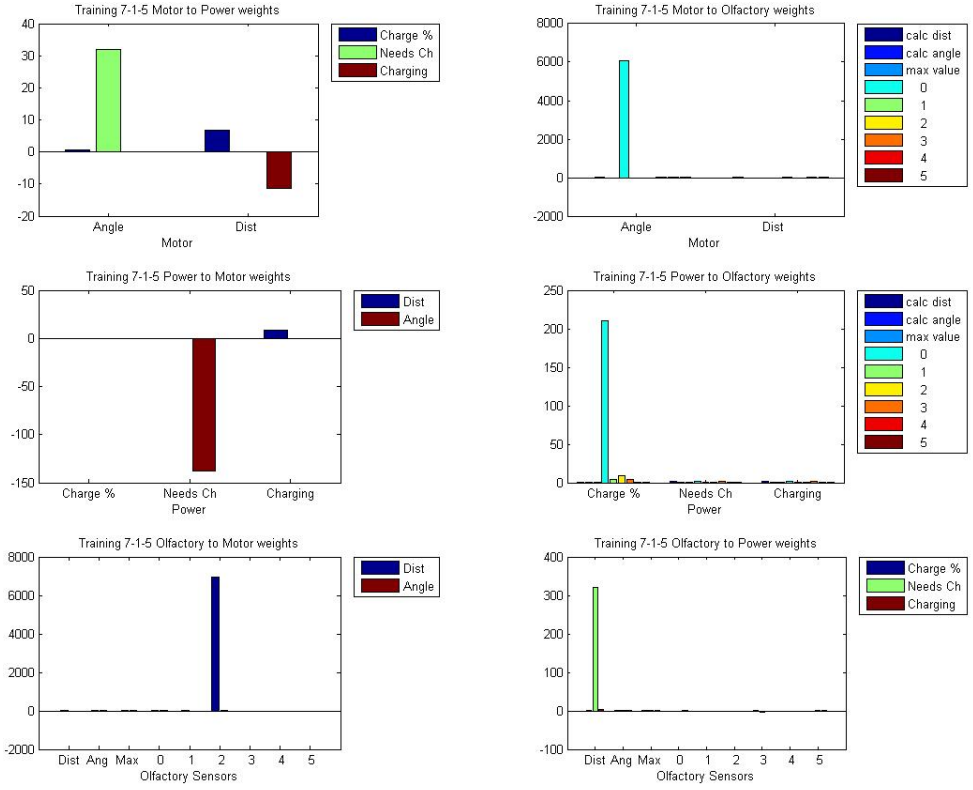


Figure 5.24: Training 7-1-5 Sensorimotor Weights

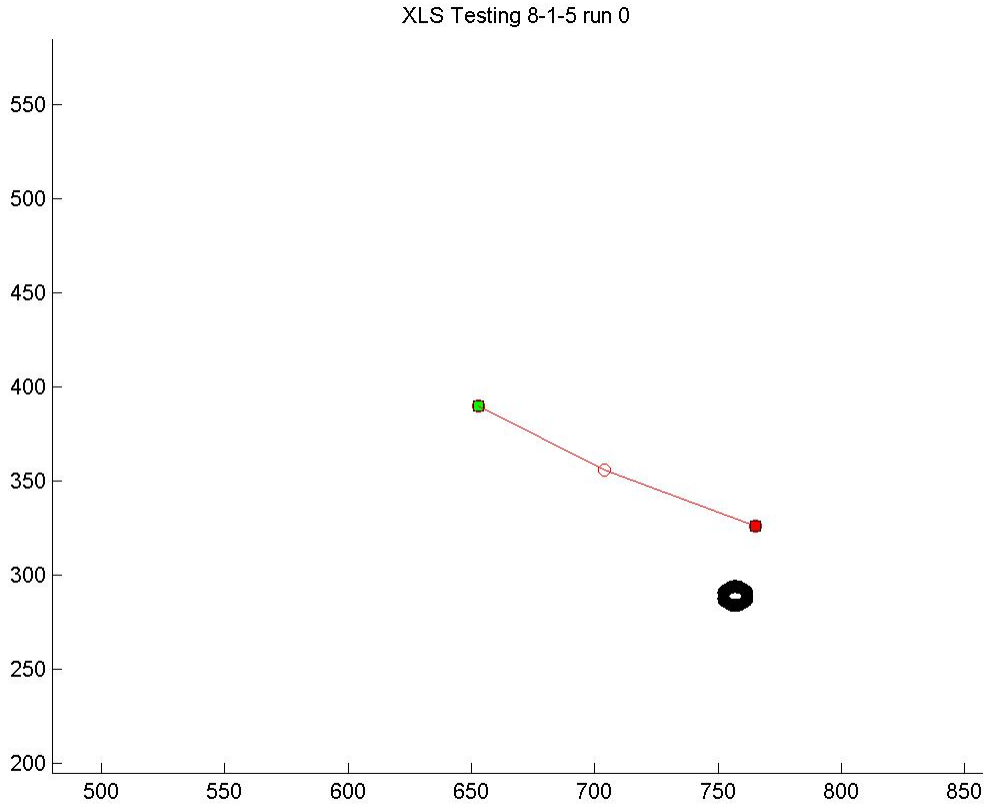


Figure 5.25: Testing 8-1-5, run 0, EvBot movement

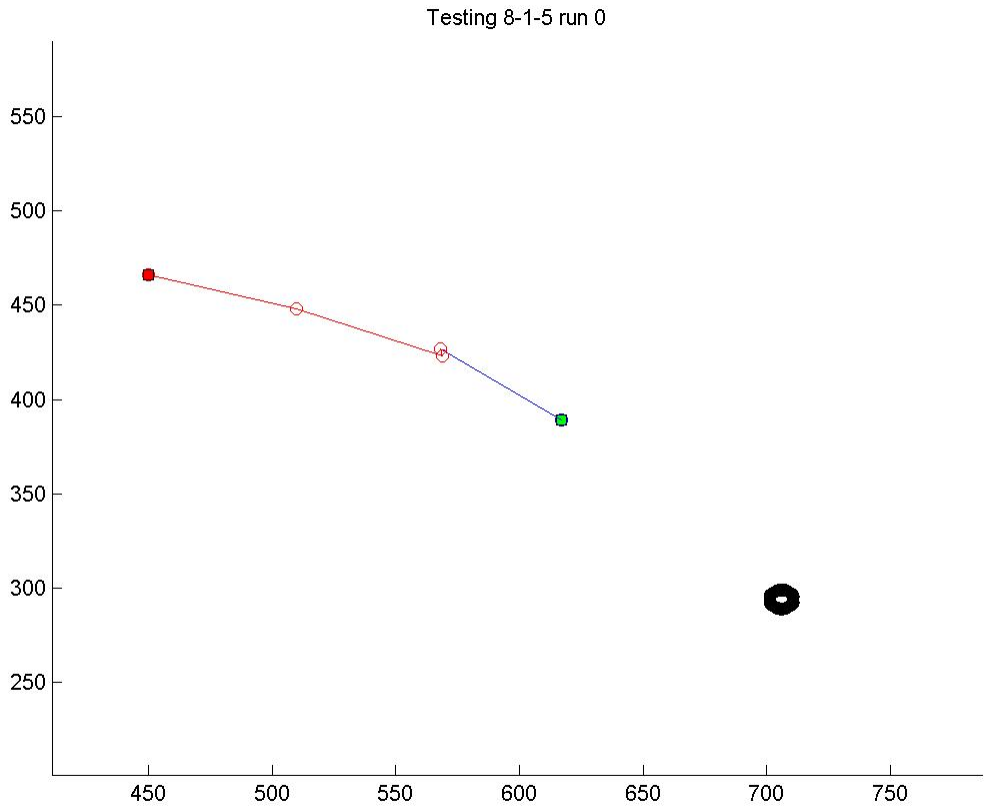


Figure 5.26: Testing 8-1-5, run 1, EvBot movement

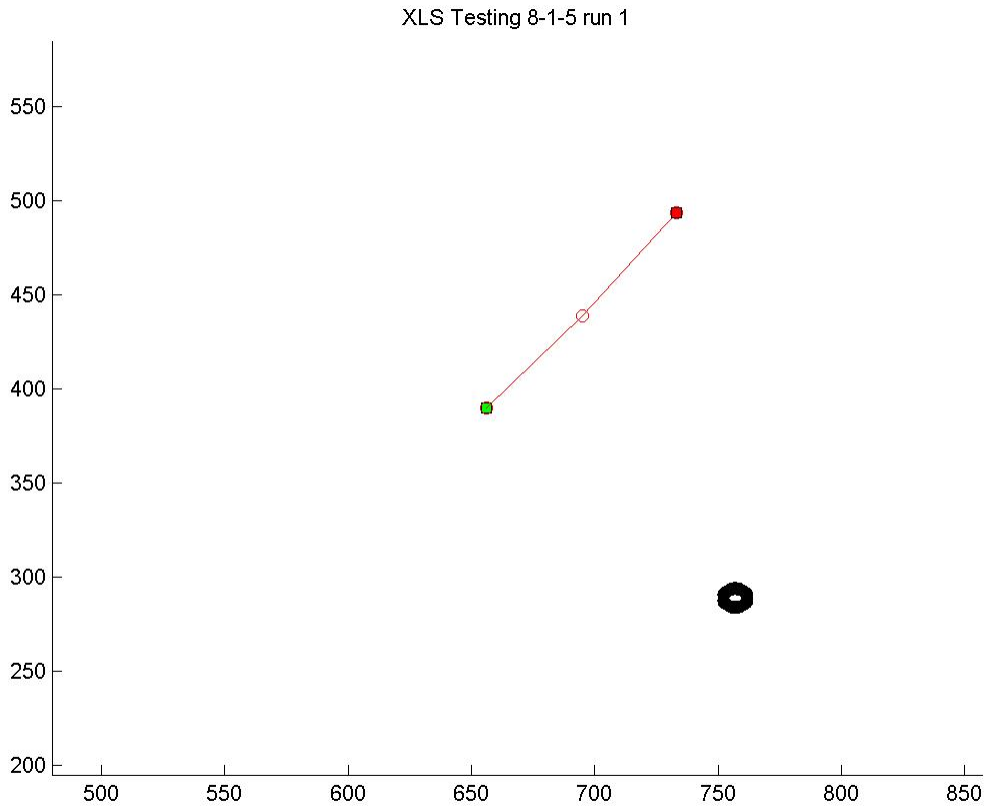


Figure 5.27: Testing 8-1-5, run 2, EvBot movement

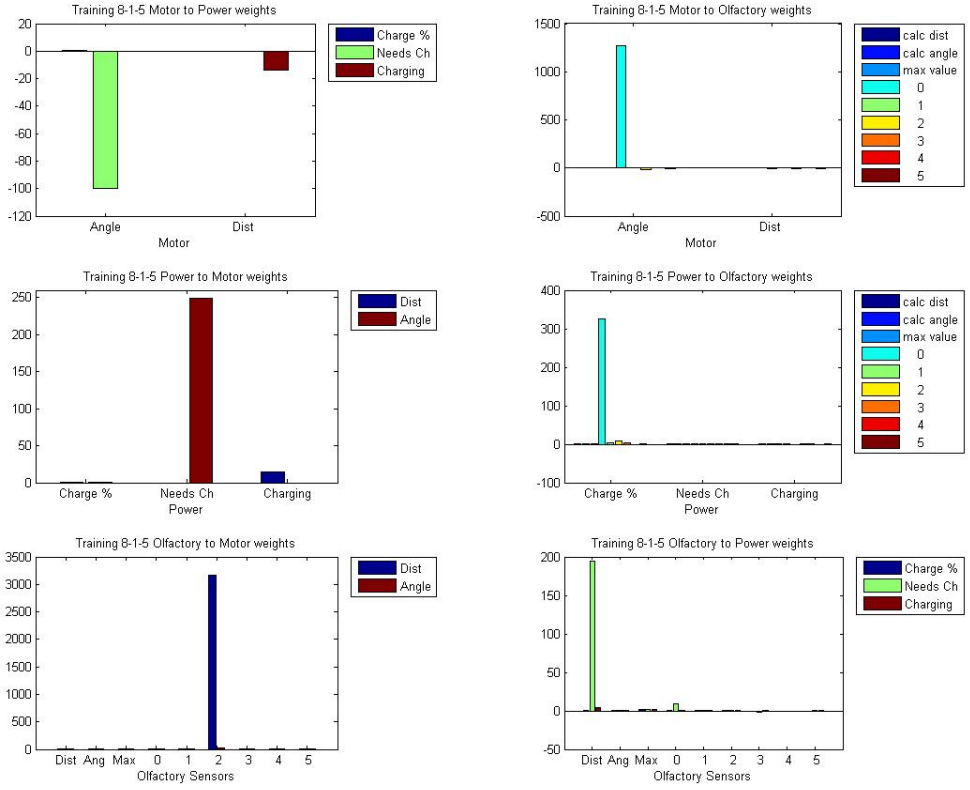


Figure 5.28: Training 8-1-5 Sensorimotor Weights

5.12 Discussion

Sensorimotor networks have the potential to increase the effectiveness of robots that are operating in dynamic environments on complex tasks. A fully interconnected sensorimotor network that was based on the work of Bovet, et. al. at the AI Lab, University of Zurich [3, 66, 93] was implemented on the EvBot III platform. Originally, the sensorimotor network was designed to be executed in C++ and run in a Linux environment. However, the decision was made to implement the final network in LabVIEW on a Windows 7 PC to speed up the development process.

The network contained motor, power, and olfactory node modalities. The motor modality consisted of the commanded EvBot turn angle and drive distance. The power modality consisted of the current charge percentage, a flag to indicated that the EvBot needed to be recharged, and a flag to indicate that the EvBot was charging. The olfactory modality consisted of the calculated distance to the alcohol source, the calculated angle to the alcohol source, the maximum alcohol sensor value, and the normalized sensor readings for each of the six alcohol sensors at the zero degree orientation.

Training experiments were conducted in order to build the sensorimotor weights to correlate increased alcohol concentration with increased charge. It was intended that this would result in the EvBot learning to autonomously navigate up an alcohol plume to the source when it needed to be re-charged. The experiments were designed with the main goal of limiting experimenter bias.

For Experiment 1, only random movements were used because it was hypothesized that this would give the sensorimotor network the best chance of developing novel, emergent behaviors for homing. During the testing phase of Experiment 1, the EvBot showed some interesting behavior by pendulating across the alcohol plume when it needed to be charged; however, it did not progress toward the alcohol source.

For the next series of experiments, the training phase included the EvBot remaining at the starting location until it needed charge before being instructed to drive towards the charging station. In Experiment 2, the EvBot was actively driven down the alcohol plume to the charging station in order to help build a correlation between charge and an increasing alcohol gradient. During testing, the EvBot performed worse than in the first run, and simply moved randomly without regard for the alcohol gradient or its charging state. For Experiment 3, the same protocol was used as compared with Experiment 2, except the EvBot started at Position 2 versus Position 3 (Fig. 5.6). This was done in order to increase the difference in values for the gradient from 0.291V for Experiment 2, to 0.501V for Experiment 3. It was anticipated that the larger gradient would further build correlations between increasing alcohol concentration and the charging location. However, when testing these trained sensorimotor weights, the EvBot drove straight when it needed to be charged regardless its orientation to the alcohol source. In order to correct this behavior, which was seen as a result of poor experimental design, the EvBot was positioned at different starting angles to the alcohol source. Therefore, the EvBot had to turn towards the alcohol source before driving forward versus simply driving forward whenever it needed to be charged.

Unfortunately, this more rigorous training protocol did not effect the behavior of the EvBot during testing, although there was a slight increase in weight values due to the increase in the number of training runs needed to include all straight training runs and the angled training runs.

For Experiments 5 and 6, the random movement setup from the first training run was included with the directed training from Experiments 2-4. In order to do this, the training run was started with the EvBot at the charging circle (i.e., location 1; Fig. 5.6), and then the EvBot was driven randomly with movements that were generated offline using Matlab in order to eliminate real-world collisions. Once the EvBot had performed 25 random movements, the charge level was depleted so that it would need to be charged, and the EvBot was commanded to turn and drive straight towards the charging circle in 0.5m steps. Due to larger than expected movement error in Experiment 5, the EvBot experienced collisions with the arena walls. In order to address this, the safety margins were increased in the Matlab random movement generation code, and the experiment was run again without collisions (Experiment 6). Although the results for Experiment 6 were slightly different from those seen with Experiments 3 and 4 (i.e., the EvBot drove backwards until it needed to be charged before driving straight, regardless of the location of the alcohol source), it was still unable to navigate to the alcohol source.

As manually driving the EvBot to the alcohol source had not proven successful, an alcohol homing algorithm that used the actual values of the alcohol sensors was developed (Section 4.6.1) and used during training for Experiments 7 and 8. It was hypothesized that this would build more accurate sensorimotor correlations because the actual sensor values would be used in the training movements. For Experiment 7, the EvBot performed random movements until it needed to be charged, and in Experiment 8, the EvBot was started needing charge. The results of testing for both of these experiments was identical; the EvBot drove forward immediately regardless of whether it needed to be charged or of the presence of alcohol.

Although the EvBot was unable to successfully navigate to the alcohol source, it did demonstrate some interesting behavior. For example, in some of the early experiments where the EvBot was turned 15° off-center, the Olfactory-to-Power weights for sensors 0 and 1 showed substantially higher values than for the other alcohol sensors (i.e., in this position, sensors 0 and 1 were the closest two sensors pointing toward the alcohol source). Additionally, the Power-to-Olfactory weights for sensors 0 and 3 had the highest values; these sensors were at the front-center and rear-center of the robot, and would seem likely sensors for a user-defined rule-based controller.

The EvBot III Sensorimotor Network's ability to navigate to the alcohol source may be improved if better sensors were used. This is supported by the fact that the WEKA software could only determine the location of the alcohol source with, at best, 25% accuracy. Although the MQ-3 metal oxide alcohol sensors are inexpensive and easy to use, they are difficult to calibrate, suffer from long settling delays, and exhibit drift. Also, they are not sensitive enough to exhibit the needed contrast from the far edge of the plume to the source location. This lack of alcohol information quality was an additional cause of the sensorimotor network's poor performance.

In addition to improved alcohol sensors, more sensing modalities with a similar layout and response to the alcohol sensors should be used. Bovet et. al. found that cross-modal correlations emerged from a partial overlap of sensory modalities that were based on different physical processes. This partial sensory overlap is captured by the redundancy principle [93]. For this application, wind direction or wind speed sensors could be used to support the alcohol sensor modality. If redundant sensing modalities were used, the correlations may better reflect the true analog nature of then environment versus trying to encode gradient information with binary sensors.

The training experimental setup could have also been improved. For example, in experiments where the EvBot was commanded to drive forward when it needed to be charged, a reciprocal "learned action" might easily be described by driving away from the alcohol source when not needing charge, and then driving forward when needing charge. Even though random movements were eventually used when the EvBot did not need to be charged, the location of the alcohol source may have also affected the results. For example, because the alcohol source was positioned in the far corner of the arena near two walls, the chance of the EvBot driving near the alcohol source / charging area were greatly reduced compared to if it was positioned in a more central location [93].

Chapter 6

Summary of Findings

As robotic systems continue to transition from the laboratory to real world environments, it is becoming increasingly difficult to fully prepare for all possible conditions and task challenges. Traditionally, developers have outfitted mobile platforms with more sensors in order to provide control system with more information so that appropriate decisions can be made [15–17]. Others have attempted to constrain robotic systems by explicitly choosing the base behaviors that the robot will need in order to perform a task [3, 20–23]. Both approaches limit the usefulness of the robot either through "information overload" [30] or by introducing experimenter bias [3]. However, control architectures that allow for embodied emergent intelligent behaviors can provide stable, although possibly non-optimal, solutions to the unpredictable real-world environment [3]. Toward this end, a modular mobile robotic platform (EvBot III) was developed that could learn and operate autonomously in a dynamic environment with limited sensor complexity and experimenter bias. Using alcohol sensing as an example application, the goal of this research was to demonstrate the implementation of a sensorimotor architecture on the EvBot III base.

The EvBot platform was originally created at the Center for Robotics and Intelligent Machines as an inexpensive tool for conducting general-purpose robotic research [26]. While versatile, the EvBot I [26] and II [27] suffer from problems related to slippage in the tread-driven steering system, decreasing battery life, and a noticeable shortage in computational power. The EvBot III was created for this research to address these problems, as well as increase the modularity and robustness of the platform as a whole. The EvBot III base has three main parts: (1) the main controller, which is located at the top of the platform, (2) the circuitry required to drive and power the base, which is located in the bottom of the platform, and (3) the sensor shield. The main controller is tasked with implementing the higher-level base commands (i.e., move forward, move backward, turn right, etc.). Currently, the main controller is implemented in LabVIEW on a laptop running Windows 7 (64-bit). The circuitry required to drive the base is separate from the main controller; it is responsible for interpreting the higher level commands into those needed to actually move the base. In the current setup, a differential drive base has been im-

plemented with the ability to fully rotate the top controller and sensor shield together. Turn angle and drive distance are sent to the base controller, and those commands are converted to PID-controlled PWM outputs for each motor. This arrangement allows the drive component to be easily switched out without changing the top, main controller. Finally, the sensor shield is made from lightweight styrene and encircles the entire robot base. Again, the sensor shield can be easily replaced to change the available sensor suite. Power to the top controller and sensors is provided and monitored by the OceanServer™ Intelligent Battery and Power System (IBPS™).

A sensorimotor network was implemented on the EvBot III in order to provide the opportunity to autonomously learn to operate in a dynamic environment with minimal experimenter bias. The sensorimotor network viewed all of the components as sensorimotor elements, without distinguishing between sensor elements or actuation elements. A flat organization was achieved by interconnecting all of the sensorimotor elements and allowing the correlations to build over time; this allowed for the controller to be shaped by the environmental interactions.

When considering biologically-inspired robotics, chemical sensing provides a good starting point for basic sensorimotor integration because it is the most widespread sensory modality among living creatures [28]. Therefore, in order to test the EvBot III's ability to operate and learn in a dynamic environment, an alcohol sensor shield was developed for the robot, and a maze environment retrofitted with an alcohol source that was positioned at the robot's charging location. The goal was to implement a basic hunger mechanism whereby the EvBot III would learn to associate an increase in alcohol concentration with increasing charge. This would result in the EvBot III learning to autonomously navigate to an alcohol source when it needed to be charged. This is akin to the chemical sensing mechanisms used by simple organisms to navigate up an odorant plume to a food source [28]. With this base behavior, increasing complex behaviors could later be added to evolve a robotic control system that could robustly deal with a dynamic environment.

A series of experiments was developed that included a "learning" phase and a "testing" phase. Initially, alcohol homing was not explicitly defined, and was left to be learned through sensorimotor correlations. However, it was found that an alcohol homing mechanism was needed for training, and so an algorithm was developed that dealt with the inherent variability of the alcohol sensors. This lead to a further investigation of chemical sensing techniques.

Robotic airborne chemical sensing is commonly implemented using one of three approaches: (1) using only passively-sampled chemical sensors, (2) using wind direction sensors along with passively-sampled chemical sensors, and (3) using actively-sampled chemical sensors [28,94,96,98,101,102,107]. While actively-sampled chemical sensors can be more effective, the added power requirements and weight must be considered for mobile platforms. On the other hand, passively-sampled chemical sensors are simpler and more efficient, they do not perform as well when the wind speeds are low and/or variable (i.e., a typical indoor environment).

In order to address these challenges, a new alcohol homing algorithm was developed for the EvBot III. A passively-sampled approach was used with six MQ-3 metal oxide alcohol sensors. Two different sensor configurations were investigated, and two different algorithms were tested. In the first configuration, the alcohol sensors were positioned at the bottom of the robot and facing tangent to the plume. The sensors were evenly-spaced around the circumference of the robot. In the second configuration, the sensors were positioned at the top of the robot and facing perpendicular to the plume. The sensors were evenly-spaced around the robot at twice its diameter. Using the same algorithm, the second configuration was found to be more accurate and produced a greater differentiation in sensor readings.

With respect to the sensor algorithms, the first approach used a reading from each of the six alcohol sensors to calculate the x- and y-components (based off of concentration strength) to the alcohol source. Those components were summed to determine the resultant vector. For the second algorithm, four readings from each sensor were taken (i.e., at the initial orientation, and then rotated in three, 90° increments). The readings were normalized for each sensor with respect to itself, and the resultant vectors from each of the readings were computed and averaged. Then, the resultant six vectors (i.e., one for each sensor) were averaged in order to determine the final angle to the alcohol source. When implemented, this algorithm produced an emergent zigzag homing approach similar to the Modified Bombyx Mori algorithm [95], the Zigzag Approach [98], and the Dung Beetle Algorithm [94]. Here, the EvBot III would drive to the center of the alcohol plume, and then pendulate back-and-forth across the plume until it reached the charging location. Although the EvBot III could successfully navigate to the alcohol source when this homing algorithm was used, navigation may be improved with better alcohol sensors. For example, although the MQ-3 metal oxide alcohol sensors were inexpensive and easy to use, they were difficult to calibrate, suffered from long settling delays, and exhibited drift. Also, they are not sensitive enough to exhibit the needed contrast from the far edge of the plume to the source location.

The custom alcohol homing algorithm was incorporated into the "learning" phase of the sensorimotor experiments such that the EvBot III was instructed to move randomly until it needed to be charged, and then use the alcohol homing algorithm to navigate to the source. Although the sensorimotor network developed correlations using this approach, it still did not build sufficient correlation relating increased alcohol concentration with charging. Therefore, in addition to improved alcohol sensors, sensing modalities with a similar layout and response to the alcohol sensors should be used. Sensory overlap is the driving force behind the *redundancy principle*, which strengthens the development of more relevant, cross-modal correlations [93]. For example, the addition of wind direction or wind speed sensors could have supported the information available from the alcohol sensors and strengthened these correlations. This redundancy would have better reflected the true analog nature of the environment compared to trying to encode gradient information using binary sensors. Additionally, the wind direction sensors could be used to activate the appropriate sensors on the base that faced up the alcohol plume. This would help better tune the weights for the alcohol sensors by increasing the difference between their readings. An accurate mathematical model of the various plume detection sensors could also be used to give an

additional virtual sensor to increase the effectiveness of the alcohol homing algorithm and the learned sensorimotor network. An accurate mathematical model would also help in the analysis of the homing algorithm and learned sensorimotor network.

The training experimental setup could have also been improved. For example, in experiments where the EvBot III was commanded to drive forward when it needed to be charged, a reciprocal "learned action" might easily be described by driving away from the alcohol source when not needing charge, and then driving forward when needing charge. Even though random movements were eventually used when the EvBot III did not need to be charged, the location of the alcohol source may have also affected the results. For example, because the alcohol source was positioned in the far corner of the arena near two walls, the chance of the EvBot III driving near the alcohol source / charging area were greatly reduced compared to if it was positioned in a more central location [93]. The use of tall walls surrounding the alcohol plume could have also led to a more artificial and complex plume, making the training more difficult. In addition to removing the walls surrounding the arena, adding a fan at the far side of the arena to draw the plume across the arena could make the alcohol plume more realistic. Using a fan to pull the plume across the arena would also help the formation of the plume's streamlines.

This dissertation reported on the use of a flat, homogeneous sensorimotor network for autonomously learning to navigate up an alcohol plume to a charging area. A new modular and highly reconfigurable mobile robot platform was designed and built to serve as the testbed. Two alcohol sensor configurations, along with two alcohol homing algorithms, were tested, with one combination exhibiting a zigzag navigational approach that resembled those observed when using previously developed biologically-inspired algorithms. This algorithm was incorporated into the training phase of the sensorimotor experiments where it was hoped that the EvBot III would learn to associate a higher concentration of alcohol with increasing charge. Although there was evidence to suggest that the sensorimotor network could eventually learn this correlation, the EvBot III was unable to successfully home to the alcohol source during the testing phase of the sensorimotor experiments. Future work includes using more sensitive alcohol sensors, incorporating additional sensors that could provide overlapping information (i.e., wind direction and speed), and improving the experimental setup to increase the information quality exhibited by the cross-modal correlations built during training. Ultimately, the incorporation of sensorimotor networks in robotic systems will lead to emergent intelligence and robust behavior through autonomous robotic reflexes. This will enable experimenters and designers to focus on higher-level tasks, and push the lower-level actions to the controller to allow for more complex behavior.

References

[1] M. Colon, "A new operating system and application programming interface for the evbot robot platform," Master's thesis, North Carolina State University, Raleigh, NC, April 2010.

[2] J. Ashcraft, "Design and development of an expandable robot simulation framework," Master's thesis, North Carolina State University, Raleigh, NC, May 2011.

[3] S. Bovet and R. Pfeifer, "Emergence of coherent behaviors from homogeneous sensorimotor coupling," in *Advanced Robotics., Proceedings of The International Conference on*, (Seattle, Washington), 2005.

[4] J. Weston and M. N. Yionoulis, "Nc state engineers design pipe-crawling robot to help save lives," *NC State University: College of Engineering News*, vol. Spring, pp. 1,4, 2000.

[5] J. Scanlon, "Snake eyes," *Wired*, December 1999.

[6] A. Drenner, I. Burt, T. Dahlin, B. Kratochvil, C. McMillen, B. Nelson, N. Papanikolopoulos, P. Rybski, K. Stubbs, D. Waletzko, and K. B. Yesin, "Mobility enhancements to the scout robot platform," in *Robotics and Automation, 2002. Proceedings. ICRA '02. IEEE International Conference on*, vol. 1, pp. 1069–1074, 2002.

[7] R. R. Murphy, "Marsupial and shape-shifting robots for urban search and rescue," *Intelligent Systems and their Applications, IEEE*, vol. 15, no. 2, pp. 14–19, 2000.

[8] J. Batlle and P. Ridao, "Mobile robots in industrial environments," *Human Systems Management*, vol. 18, pp. 275–285, January 1999.

[9] S. B. Nickerson, P. Jasiobedzki, D. Wilkes, M. Jenkin, E. Milios, J. Tsotsos, A. Jepson, and O. Bains, "The ark project: Autonomous mobile robots for known industrial environments," *Robotics and Autonomous Systems*, vol. 25, pp. 83–104, October 1998.

[10] T. H. Pastore, H. R. Everett, and K. Bonner, "Mobile robots for outdoor security applications." American Nuclear Society 8th International Topical Meeting on Robotics and Remote Systems, April 1999.

[11] M. Toscano, "Department of defense joint robotics program," in *2001 SPIE Unmanned Ground Vehicle Technology III, Proceedings of the*, (Orlando, Fl), pp. 313–322, April 2001.

[12] S. Ekvall, P. Jensfelt, and D. Kragic, "Integrating active mobile robot object recognition and slam in natural environments," in *Intelligent Robots and Systems, 2006 IEEE/RSJ International Conference on*, (Beijing), pp. 5792–5797, October 2006.

[13] S. Betge-Brezetz, P. Hebert, R. Chatila, and M. Devy, "Uncertain map making in natural environments," in *Robotics and Automation, 1996. Proceedings., 1996 IEEE International Conference on*, vol. 2, pp. 1048–1053, 1996.

[14] A. L. Nelson, E. Grant, G. J. Barlow, and T. C. Henderson, "A colony of robots using vision sensing and evolved neural controllers," *Intelligent Robots and Systems., Proceedings. IEEE/RSJ International Conference on,*, vol. 3, pp. 2273–2278, October 2003.

[15] S. Thrun, M. Montemerlo, H. Dahlkamp, D. Stavens, A. Aron, J. Diebel, P. Fong, J. Gale, M. Halpenny, G. Hoffmann, K. Lau, C. Oakley, M. Palatucci, V. Pratt, P. Stang, S. Strohband, C. Dupont, L.-E. Jendrossek, C. Koelen, C. Markey, C. Rummel, J. v. Niekerk, E. Jensen, P. Alessandrini, G. Bradski, B. Davies, S. Ettinger, A. Kaehler, A. Nefian, and P. Mahoney, *The 2005 DARPA Grand Challenge*, vol. 36 of *Springer Tracts in Advanced Robotics*, ch. Stanley: The Robot That Won the DARPA Grand Challenge, pp. 1–43. Springer Berlin / Heidelberg, 2007.

[16] C. Urmson, J. Anhalt, M. Clark, T. Galatali, J. P. Gonzalez, J. Gowdy, A. Gutierrez, S. Harbaugh, M. Johnson-Roberson, Y. H. Kato, P. Koon, K. Peterson, B. Smith, S. Spiker, E. Tryzelaar, and W. R. Whittaker, "High speed navigation of unrehearsed terrain: Red team technology for grand challenge 2004," Tech. Rep. CMU-RI-TR-04-37, The Robotics Institute, Carnegie Mellon University, 5000 Forbes Avenue, Pittsburgh, PA 15213, June 2004.

[17] "H1ghlander." Webpage, 2004.

[18] A. Melihen and L. Nastro, "The darpa grand challenge: Moving technology forward," *GEO Informatics*, vol. 9, pp. 42–56, March- 2006.

[19] K. Iagnemma and M. Buehler, "Editorial for journal of field roboticsspecial issue on the darpa grand challenge," *Journal of Field Robotics*, vol. 23, pp. 461–462, August 2006.

[20] P. F. M. J. VeVerschure, T. Voegtlin, and R. J. Douglas, "Environmentally mediated synergy between perception and behaviour in mobile robots," *Nature*, vol. 425, pp. 620–624, October 2003.

[21] R. Brooks, "A robust layered control system for a mobile robot," *Robotics and Automation, IEEE Journal of*, vol. 2, no. 1, pp. 14–23, 1986.

[22] V. Braitenberg, *Vehicles: Experiments in Synthetic Psychology*. MIT Press, 1984.

[23] R. Pfeifer and C. Scheier, *Understanding Intelligence*. Cambridge, MA, USA: MIT Press, 1999.

[24] D. Lambrinos, R. Möller, T. Labhart, R. Pfeifer, and R. Wehner, "A mobile robot employing insect strategies for navigation," in *Biometric Robots., Robotics and Autonomous Systems, special issue on*, vol. 30, pp. 36–64, 2000.

[25] V. Hafner, H. Kunz, and R. Pfeifer, "An investigation into obsticle avoidance as an 'emergent' behavior from two different perspectives," in *Biologically-Inspired Robotics: The Legacy of W. Grey Walter., Proceedings of the EPSRC/BBSRC International Workshop on*, (Bristol), pp. 166–173, 2002.

[26] J. Galeotti, "The evbot: A small autonomous mobile robot for the study of evolutionary algorithms in distributed robotics.," Master's thesis, North Carolina State University, Raleigh, NC, March 2002.

[27] L. S. Mattos, "The evbot ll: An enhanced evolutionary robotics platform equipped with integrated sensing for control," Master's thesis, North Carolina State University, Raleigh, NC, 2003.

[28] R. A. Russell, *Odour Detection by Mobile Robots*, vol. 22 of *World Scientific Series in Robotics and Intelligent Systems*. World Scientific Pub Co Inc, 1st ed., July 1999.

[29] C. Giovannangeli and P. Gaussier, "Interactive teaching for vision-based mobile robots: A sensory-motor approach," *Systems, Man and Cybernetics, Part A: Systems and Humans., IEEE Transactions on, IEEE Transactions on*, vol. 40, pp. 13–28, January 2010.

[30] C. Schaefer and K. Hintz, "Sensor management in a sensor rich environment," in *Aerospace/Defense Sensing and Control., Proceedings of the SPIE International Symposium on*, vol. 4052, pp. 48–57, SPIE, 2000.

[31] M. E. Liggins, D. L. Hall, and J. Llinas, eds., *Handbook of Multisensor Data Fusion: Theory and Practice*. The Electrical Engineering and Applied Signal Processing Series, CRC Press, second edition ed., 2009.

[32] E. of The American Heritage Dictionaries, *"Sensorimotor"*. American Heritage, the american heritage stedman's medical dictionary ed., September 2002. defn. originally from 'The American Heritage Stedman's Medical Dictionary', but found by searching www.dictionary.com.

[33] B. L. Riemann and S. M. Lephart, "The sensorimotor system, part i: The physiologic basis of functional joint stability," *Journal of Athletic Training*, vol. 37, no. 1, pp. 71–79, 2002.

[34] M. Abidi and R. Gonzalez, *Data Fusion in Robotics and Machine Intelligence*. San Diego, CA: Academic Press Professional, Inc., 1992.

[35] C. Harris, Z. Wu, K. Bossley, and M. Brown, "Intelligent neurofuzzy estimators and multisensor data fusion," in *Methods and Applications of Intelligent Control* (S. Tzafestas, ed.), (Netherlands), pp. 283–303, Kluwer Academic, 1997. Ch. 10.

[36] R. Joshi and A. C. Sanderson, *Multisensor Fusion: A Minimal Representation Framework*, vol. 11 of *Intelligent Control and Intelligent Automation*. World Scientific Publishing Co., 1999.

[37] Y. Shirai, "Visual sensor fusion," in *Multisensor Fusionand Integration for Intelligent Systems., Proceedings of International Conference on*, (Las Vegas, Nevada), October 1994. Tutorial.

[38] J. W. M. Van Dam, B. J. A. Kröse, and F. C. A. Groen, "Neural network applications in sensor fusion for an autonomous mobile robot," in *Reasoning withUncertainty in Robotics., Proceedings of the International Workshop on*, (London, UK), pp. 263–278, Springer-Verlag, 1996.

[39] B. Siciliano and O. Khatib, eds., *Springer Handbook of Robotics*. Springer Berlin, 2008.

[40] D. Hall and J. Llinas, "An introduction to multisensor data fusion," *Proceedings of the IEEE*, vol. 85, no. 1, pp. 6–23, 1997.

[41] T. L. Griffiths and A. Yuille, "Technical introduction: A primer on probabilistic inference," *Trends in Cognitive Sciences*, vol. 10, no. 7, 2006.

[42] J. Manyika and H. F. Durrant-Whyte, *Data Fusion and Sensor Management: A Decentralized Information-Theoretic Approach*. Robotics: Vision, Manipulation and Sensors, Upper Saddle River, NJ, USA: Prentice Hall PTR, 1995.

[43] D. L. Hall and J. Llinas, *Handbook of Multisensor Data Fusion*. The Electrical Engineering and Applied Signal Processing Series, CRC Press, 2001.

[44] Y. Zhu, *Multisensor Decision and Estimation Fusion*. The Kulwer International Series on Asian Studies in Computer and Information Sciences, Kluwer Academic Publishers, 2003.

[45] N. S. V. Rao, "Projective method for generic sensor fusion problem," in *Multi-SensorFusion and Integration for Intelligent Systems., Proceedings of 1999 IEEE International Conference on*, (Taipei, Taiwan), R.O.C., August 1999.

[46] N. S. Rao, E. Oblow, C. Glover, and G. Liepins, "N-learners problem: Fusion of concepts," in *System, Man and Cybernetics., IEEE Transactions on*, vol. 24, pp. 319–327, IEEE, 1994.

[47] N. Rao, "Fusion methods for multiple sensor systems with unknown error densities," *Journal of Franklin Institute*, vol. 331B, no. 5, pp. 509–530, 1994.

[48] N. S. Rao, "A generic sensor fusion problem: Classification and function estimation," workshop presentation, Oak Ridge National Laboratory, Cagliari, Italy, June 2004. Sponsered by the Office of Naval Research.

[49] K. J. Hintz, "Gmugle: A goal lattice constructor," in *Signal Processing,Sensor Fusion, and Target Recognition X., Proceedings of the SPIE International Symposium on* (I. Kadar, ed.), vol. 4380, (Orlando, USA), pp. 324–327, SPIE, April 2001.

[50] W. Komorniczak and J. Pietrasinski, "Selected problems of mfr resource management," in *Information Fusion., Proceedings, 3rd International Conference on*, pp. WeC1–3–WeC1–8, 2000.

[51] J. Lopez, F. Rodriguez, and J. Corredera, "Symbolic processing for coordinated task management in multiradar surveillance networks," in *Information Fusion., Proceedings of the International Conference on*, (Las Vegas), pp. 725–732, 1998. Fusion '98 conference.

[52] G. A. McIntyre and K. J. Hintz, "A comprehensive approach to sensor management, part iii: Goal lattices," in *Systems, Man, and Cybernetics (SMC)., IEEE Transactions on*, IEEE, April 1999.

[53] L. Johansson, R. M., and N. Xiong, "Perception management - an emerging concept for information fusion," *Information Fusion*, vol. 4, no. 3, pp. 231–234,231, 2003.

[54] V. Caglioti, "An entropic criterion for minimum uncertainty sensing in recognition and localization part ii: A case study on directional distance measurements," *Systems, Man and Cybernetics, Part B., IEEE Transactions on*, vol. 31, pp. 197–214, April 2001.

[55] P. Dodin, J. Verliac, and V. Nimier, "Analysis of the multisensor multitarget tracking resource allocation problem," in *Information Fusion., Proceedings, 3rd International Conference on*, pp. WeC1–17–WeC1–22, 2000.

[56] W. Schmaedeke and K. Kastella, "Information based sensor management and immkf," in *Signal and Data Processing of Small Targets., Proceedings of the SPIE International Conference on*, pp. 390–401, SPIE, 1998.

[57] K. Jöred and P. Svensson, "Target tracking in archipelagic asw: a not-so-impossible proposition?," in *Underwater Defense Technology Europe 98., Proceedings of*, pp. 172–175, 1998. (Only html cached version, not the pdf).

[58] D. Penny and M. Williams, "A sequential approach to multi-sensor resource management using particle filters," in *Signal and Data Processing of Small Targets., Proceedings of the SPIE International Conference on*, pp. 598–609, SPIE, 2000.

[59] P. Svensson, K. Johansson, and K. Jöred, "Submarine tracking using multi-sensor data fusion and reactive planning for the positioning of passive sonobouys," in *Proceedings of Hydroakustik*, (Stockholm, Sweden), Swedish Defense Research Establishment (FOA), 1997. 9th Conference of Coherent Laser Radar.

[60] D. Cook, P. Gmytrasiewicz, and L. Holder, "Decision-theoretic cooperative sensor planning," in *Pattern Analysis and Machine Intelligence., IEEE Transactions on*, vol. 18, pp. 1013–1023, IEEE, 1996.

[61] R. O. Duda, P. E. Hart, and D. G. Stork, *Pattern Classification*. John Wiley & Sons, Inc., 2nd ed., 2001.

[62] O. Sporns and T. K. Pegors, "Information-theoretical aspects of embodied artificial intelligence," in *Lecture Notes in Computer Science*, vol. 3139, pp. 74–85, Springer-Verlag GmbH, July 2004.

[63] H. A. Sowizral, "Virtual sensors," in *Stereoscopic Displays and Virtual Reality Systems II., Proceedings of SPIE,* (S. S. F. J. O. M. M. T. Bolas, ed.), vol. 2409, pp. 246–254, The International Society for Optical Engineering, March 1995.

[64] G. A. McIntyre and K. J. Hintz, "A comprehensive approach to sensor management, part ii: A new hierarchical model," in *Systems, Man, and Cybernetics (SMC)., IEEE Transactions on*, IEEE, April 1999.

[65] G. A. McIntyre and K. J. Hintz, "A comprehensive approach to sensor management, part i: A survey of modern sensor management sensors," in *Systems, Man, and Cybernetics (SMC)., IEEE Transactions on*, IEEE, April 1999.

[66] S. Bovet and R. Pfeifer, "Emergence of delayed reward learning from sensorimotor coordination," in *Intelligent Robots and Systems., IEEE International Conference on*, pp. 2272–2277, IEEE, August 2005.

[67] D. L. Hall and A. K. Garga, "Pitfalls in data fusin (and how to avoid them)," in *Information Fusion., In Proceedings of the International Conference on*, vol. 1, pp. 429–436, 1999.

[68] N. Oros and J. Krichmar, "Android based robotics: Powerful, flexible and inexpensive robots for hobbyists, educators, students and researchers," web, Cognitive Anteater Robotics Laboratory. University of California, Irvine, November 2013.

[69] N. Oros and J. Krichmar, "Smartphone based robotics: Powerful, flexible and inexpensive robots for hobbyists, educators, students and researchers.," CECS Technical Report 13-16, Center for Embedded Computer Systems, University of California, Irvine., November 2013.

[70] B. J. Romano, "P3-dx robot is not $40k." The Seattle Times, December 2006.

[71] A. L. Nelson, *Competitive Relative Performance and Fitness Selection for Evolutionary Robotics.* PhD thesis, North Carolina State University, Raleigh, NC, 2003.

[72] D. Gastelum, T. Jones, A. Agarwal, J. Kothari, S. Bhat, H. K. Lee, E. Grant, A. Nelson, S. Rubin, and G. K. Lee, "The development of a testbed for evolutionary learning algorithms for mobile robotic colonies," in *Collaborative Technologies and Systems - Western Multiconference., Proceedings of the International Symposium on*, (San Diego, CA), pp. 212–217, 2004. EvBot.

[73] A. Nelson, E. Grant, G. Barlow, and M. White, "Evolution of complex autonomous robot behaviors using competitive fitness," *Integration of Knowledge Intensive Multi-Agent Systems., International Conference on*, pp. 145–150, 2003.

[74] A. Nelson, E. Grant, and T. Henderson, "Evolution of neural controllers for competitive game playing with teams of mobile robots," *Robotics and Autonomous Systems*, vol. 46, pp. 135–150, 2004. EvBots.

[75] A. Nelson, E. Grant, and T. Henderson, "Competitive relative performance evaluation of neural controllers for competitive game playing with teams of real mobile robots.," in *Measuring the Performance and Intelligence of Systems., Proceedings of the 2002 PerMIS Workshop,*, pp. 43–50, 2002.

[76] J. Galeotti, S. Rhody, A. Nelson, E. Grant, and G. Lee, "Evbots - the design and construction of a mobile robot colony for conducting evolutionary robotic experiments," in *Computer Applications in Industry and Engineering., Proceedings of the ISCA 15th International Conference:*, pp. 86–91, November 2002.

[77] A. L. Nelson, E. Grant, J. M. Galeotti, and S. Rhody, "Maze exploration behaviors using an integrated evolutionary robotics environment," *Robotics and Autonomous Systems*, vol. 46, pp. 159–173, 2004.

[78] G. J. Barlow, L. S. Mattos, E. Grant, and C. K. Oh, "Transference of evolved unmanned aerial vehicle controllers to a wheeled mobile robot," in *Robotics and Automation., Proceedings of the IEEE International Conference on*, (Barcelona, Spain), pp. 2099–2104, IEEE, April 2005.

[79] G. J. Barlow and C. K. Oh, *Evolved Navigation Control for Unmanned Aerial Vehicles*, ch. 20, pp. 353–378. Vienna: I-Tech Education and Publishing, 2008.

[80] D. Burke, "A conservative approach to mounting and applying an omnidirectional vision system onto evbot ii mobile robot platforms," Master's thesis, North Carolina State University, Raleigh, NC, 2007.

[81] K. A. Alhammadi, *Applying Wide Field of View Retroreflector Technology to Free Space Optical Robotic Communications*. Electrical engineering, North Carolina State University, Raleigh, NC, September 2006.

[82] G. J. Barlow, T. C. Henderson, A. L. Nelson, and E. Grant, "Dynamic leadership protocol for s-nets," in *Robotics and Automation., Proceedings, IEEE International Conference on*, vol. 2, pp. 1091–1096, April 2004.

[83] T. Henderson, E. Grant, K. Luthy, L. Mattos, and M. Craver, "Precision localization in monte carlo sensor networks," in *Computer Applications in Industry and Engineering., International Conference on*, (Honolulu, Hawaii), November 2005. EvBot.

[84] T. C. Henderson, B. Erickson, T. Longoria, E. Grant, K. Luthy, and M. Craver, "Monte carlo sensor networks," in *Computer Applications in Industry and Engineering., International Conference on*, (Honolulu, Hawaii), November 2005.

[85] K. A. Luthy, E. Grant, and T. C. Henderson, "Leveraging rssi for robotic repair of disconnected wireless sensor networks," *Robotics and Automation., IEEE International Conference on*, pp. 3659–3664, April 2007. EvBot.

[86] OceanServer Technology, Inc., Fall River, MA, *MP-04SR/FR Battery Management Module Data Sheet*.

[87] OceanServer Technology Inc., Fall River, MA, *Intelligent Battery and Power System Specifications: Batteries, Battery Controller Modules & DC-DC Options*, 2008.

[88] C. Merritt, *Electronic Textile-Based Sensors and Systems for Long-Term Health Monitoring*. PhD thesis, North Carolina State University, March 2008.

[89] Lynxmotion, "Ghm–04 gear head motor." datasheet, 2013.

[90] F. Semiconductor, "Mc33887, 5.0 a h-bridge with load current feedback." datasheet, October 2012.

[91] M. Quigley, B. Gerkey, K. Conley, J. Faust, T. Foote, J. Leibs, E. Berger, R. Wheeler, and A. Ng, "Ros: an open-source robot operating system." ICRA workshop on open source software, May 2009.

[92] M. Y. Jung, A. Deguet, and P. Kazanzides, "A component-based architecture for flexible integration of robotic systems," *Intelligent Robots and System., IEEE/RSJ International Conference on*, pp. 6107–6112, October 2010.

[93] S. Bovet, *Robots with Self-Developing Brains*. PhD thesis, University of Zurich, 2007.

[94] R. A. Russell, A. Bab-Hadiashar, R. L. Shepherd, and G. G. Wallace, "A comparison of reactive robot chemotaxis algorithms," *Robotics and Autonomous Systems*, vol. 45, pp. 83–97, November 2003.

[95] A. Lilienthal, D. Reimann, and A. Zell, "Gas source tracing with a mobile robot using an adapted moth strategy," in *Autonomous Mobile Systems., Proceedings of*, pp. 150–160, 2003.

[96] R. A. Russell and S. Kennedy, "A novel airflow sensor for miniature mobile robots," *Mechatronics*, vol. 10, pp. 935–942, December 2000.

[97] A. M. Farah and T. Duckett, "Reactive localisation of an odour source by a learning mobile robot," in *Second Swedish Workshop on Autonomous Robotics., In Proceedings of*, (Stockholm, Sweden), October 2002.

[98] H. Ishida, K. Suetsugu, T. Nakamoto, and T. Moriizumi, "Study of autonomous mobile sensing system for localization of odor source using gas sensors and anemometric sensors," *Sensors and Actuators, A: Physical*, vol. 45, pp. 153–157, November 1994.

[99] P. P. Neumann, *Gas Source Localization and Gas Distribution Mapping with a Micro-Drone*. PhD thesis, BAM Bundesanstalt für Materialforschung und -prüfung, Berlin, 2013. Committee: Schiller, Jochen H. (Chair) and Lilienthal, Achim J. and Alt, Helmut and Liers, Achim and Bartholmai, Matthias.

[100] A. T. Hayes, A. Martinoli, and R. M. Goodman, "Distributed odor source localization," *IEEE Sensors Journal*, vol. 2, pp. 260–271, 2002.

[101] H. Ishida, Y. Kagawa, T. Nakamoto, and T. Moriizumi, "Odour-source localization in the clean room by an autonomous mobile sensing system," *Sensors and Actuators, B: Chemical*, vol. 33, pp. 115–121, July 1996.

[102] A. J. Lilienthal, A. Loutfi, and T. Duckett, "Airborne chemical sensing with mobile robots," *Sensors*, vol. 6, no. 11, pp. 1616–1678, 2006.

[103] A. Winfield and O. Holland, "The application of wireless local area network technology to the control of mobile robots," *Microprocessor and Microsystems*, vol. 23, pp. 597–607, March 2000.

[104] A. Lilienthal and T. Duckett, "Experimental analysis of gas-sensitive braitenberg vehicles," *Advanced Robotics*, vol. 18, pp. 817–834, December 2004.

[105] A. Lilienthal and T. Duckett, "A stereo electronic nose for a mobile inspection robot," in *Robotic Sensing (ROSE 2003)., Proceedings of the IEEE International Workshop on*, IEEE, June 2003.

[106] T. Nakamoto, H. Ishida, and T. Moriizumi, "An odor compass for localizing an odor source," *Sensors and Actuators, B: Chemical*, vol. 35, pp. 32–36, September 1996.

[107] R. A. Russell, D. Theil, R. Deveza, and A. Mackay-Sim, "A robotic system to locate hazardous chemical leaks," in *Robotics and Automation, Proceedings of IEEE International Confercence on*, vol. 1, (Nagoya), pp. 556–561, IEEE, May 1995.

[108] J. Casper and R. Murphy, "Human-robot interactions during the robot-assisted urban search and rescue response at the world trade center," *Systems, Man and Cybernetics, Part B, IEEE Transactions on*, vol. 33, no. 3, pp. 367–385, 2003.

[109] H. Ishida, Y. Wada, and H. Matsukura, "Chemical sensing in robotic applications: A review," *IEEE Sensors Journal*, vol. 12, pp. 3163–3173, November 2012.

[110] A. L. Nelson and E. Grant, "Using direct competition to select for competent controllers in evolutionary robotics," *Robotics and Autonomous Systems*, vol. 54, pp. 840–587, 2006.

[111] A. L. Nelson, E. Grant, and G. Lee, "Developing evolutionary neural controllers for teams of mobile robots playing a complex game," *Information Reuse and Integration., IEEE International Conference on*, pp. 212–218, October 2003.

[112] M. Hall, E. Frank, G. Holmes, B. Pfahringer, P. Reutemann, and I. H. Witten, "The weka data mining software: An update," *SIGKDD Explorations*, vol. 11, pp. 10–18, June 2009.

[113] I. H. Witten, E. Frank, and M. A. Hall, *Data Mining: Practical Machine Learning Tools and Techniques*. The Morgan Kaufmann Series in Data Management Systems, Morgan Kaufmann Publishers, 3rd ed., January 2011.

Appendices

Appendix A

Sensorimotor Network Mathematical description [3, 66, 93]

For sensorimotor modality M the current state vector is $\mathbf{x}^M = (x_1^M, x_2^M, \ldots)$, where each component is defined as the current state of one sensorimotor element, and the current state change vector is the function of the current state vector and the delayed state vector.

$$\breve{\mathbf{x}}^M(t) := \mathbf{x}^M(t - \tau) \tag{A.1}$$

$$\mathbf{y}^M(t) := \delta^M(\breve{\mathbf{x}}^M(t), \mathbf{x}^M(t)) \tag{A.2}$$

Where $\tau > 0$ is the time delay and $\delta^M(.,.)$ is the function that computes the change between two states, typically:

$$\delta^M(\mathbf{a}, \mathbf{b}) := \mathbf{b} - \mathbf{a} \tag{A.3}$$

Similarly, their virtual state change is a function of the current state and the virtual state, where the current virtual state vector is $\tilde{\mathbf{x}}^M = (\tilde{x}_1^M, \tilde{x}_2^M, \ldots)$, where each component is defined as the current virtual state of one sensorimotor element.

$$\tilde{\mathbf{y}}^M(t) := \delta^M(\mathbf{x}^M(t), \tilde{\mathbf{x}}^M(t)) \tag{A.4}$$

One very important detail of their implementation is that, although the outputs from the interface are output separately and in parallel, they use a common synapse and single synaptic weight matrix $\mathbf{W}^{MN} = (W_{ij}^{MN})$. This weight matrix is initially set to zero and is updated via a Hebbian learning rule, with the learning rate $\eta > 0$ (typically $\eta = 0.01$):

$$\mathbf{W}^{MN}(0) := 0 \tag{A.5}$$

$$\delta W_{ij}^{MN}(t) := \eta \cdot x_i^M(t) \cdot y_j^M(t) \tag{A.6}$$

$$\mathbf{W}^{MN}(t+1) := \mathbf{W}^{MN}(t) + \delta \mathbf{W}^{MN}(t) \tag{A.7}$$

This feature results in the virtual state being correlated to the current state such that it tends towards an accurate prediction of what the next state will actually be. In the model there is a forgetting term ϵ, where $\epsilon \geq 0$ (typically $\epsilon = 0$). Using this common synaptic weight matrix the updated current state and updated virtual state are:

$$\mathbf{x}^M(t+1) := \sum_{N \neq M} \mathbf{W}^{NM}(t) \cdot \mathbf{y}^N(t) \tag{A.8}$$

$$\tilde{\mathbf{x}}^M(t+1) := \sum_{N \neq M} \mathbf{W}^{NM}(t) \cdot \tilde{\mathbf{y}}^N(t) \tag{A.9}$$

Appendix B

All LabVIEW Sensorimotor Inputs

Table B.1: All EvBot Sensorimotor Network LabVIEW Inputs

Name	Type	Training	Testing	Description
Stop	boolean	FALSE	FALSE	TRUE to stop main VI xecuation
Training	boolean	TRUE	FALSE	TRUE if training experiment
Base Comm Port	VISA resource	COM4	COM4	Comm port used to control base movement
Charge Step Distance	double	0.65	0.65	Largest step (m) robot makes to charging station when "needs charge"
Start Charge Station	boolean	–	–	TRUE if start at charging station
Adjust Initial Charge	boolean	–	–	TRUE if initial charge adjusted for # of random steps and added charge from starting at charging station
Save Raw Data	boolean	TRUE	TRUE	TRUE if raw alcohol values saved to disk
Stop Alcohol Source	boolean	FALSE	FALSE	TRUE if robot stops at alcohol source
N Steps After Stop	int32	10	10	# of cycles to wait at alcohol source before stopping experiment (if "Stop at Alcohol Source" is TRUE)
Stop Done Charging	boolean	TRUE	TRUE	TRUE to stop VI when "Done Need Charge Level" is reached
Pause	boolean	FALSE	FALSE	Pauses VI execution
At Charging Station	boolean	–	–	TRUE if robot is at charging station

Table B.1 – continued from previous page

Name	Type	Training	Testing	Description
N Alcohol Readings	int16	15	15	# of alcohol readings to average
Wait Time	double	100	100	Time (s) to wait after each movement before sampling
N Random Moves	int32	25	25	# of random moves before needing charge
Reinitialize Weights	boolean	–	–	TRUE on 1st training run to set weights to initial zero value
Initial Charge	double	–	–	% initial charge for simulated battery
Decrement Charge	double	–	–	% that charge will decrease after each non-charging cycle
Increment Charge	double	–	–	% that charge will increase after each charging cycle
Need Charge Level	double	–	–	Level (%) at which "Needs Charge" is TRUE
Done Need Charge Level	double	–	–	Level (%) at which battery is considered fully charged
Time Step (ms)	uint32	0	0	Specify cycle time step (ms) (unused)
Movement File Name	string	–	–	File containing pre-generated random movements
Training Run #	int32	n	n	Run # or movement file
Drive Method	enum	Read movement file	Testing drive	Selection determines if a pre-generated movement file is used for training or if sensorimotor network is used for testing
Learning Rate	double	0.01	0.01	The positive scaling factor for determining the weight change for the indicated link
Forgetting Rate	double	0	0	The regressive scaling factor for determining the weight change for the indicated link
Modality Inputs	cluster			An empty structure used to specify the container for the various modality inputs

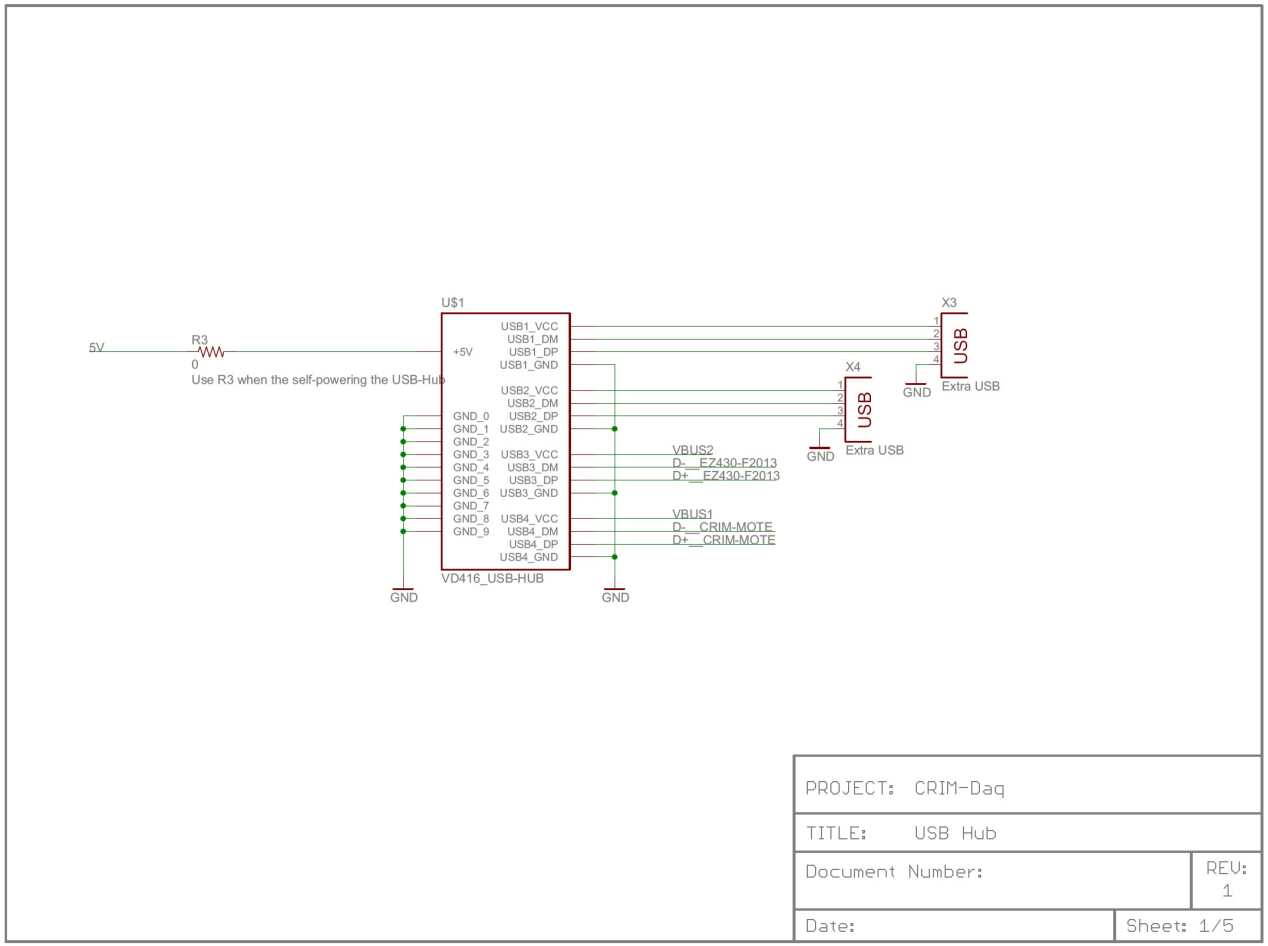
Table B.1 – continued from previous page

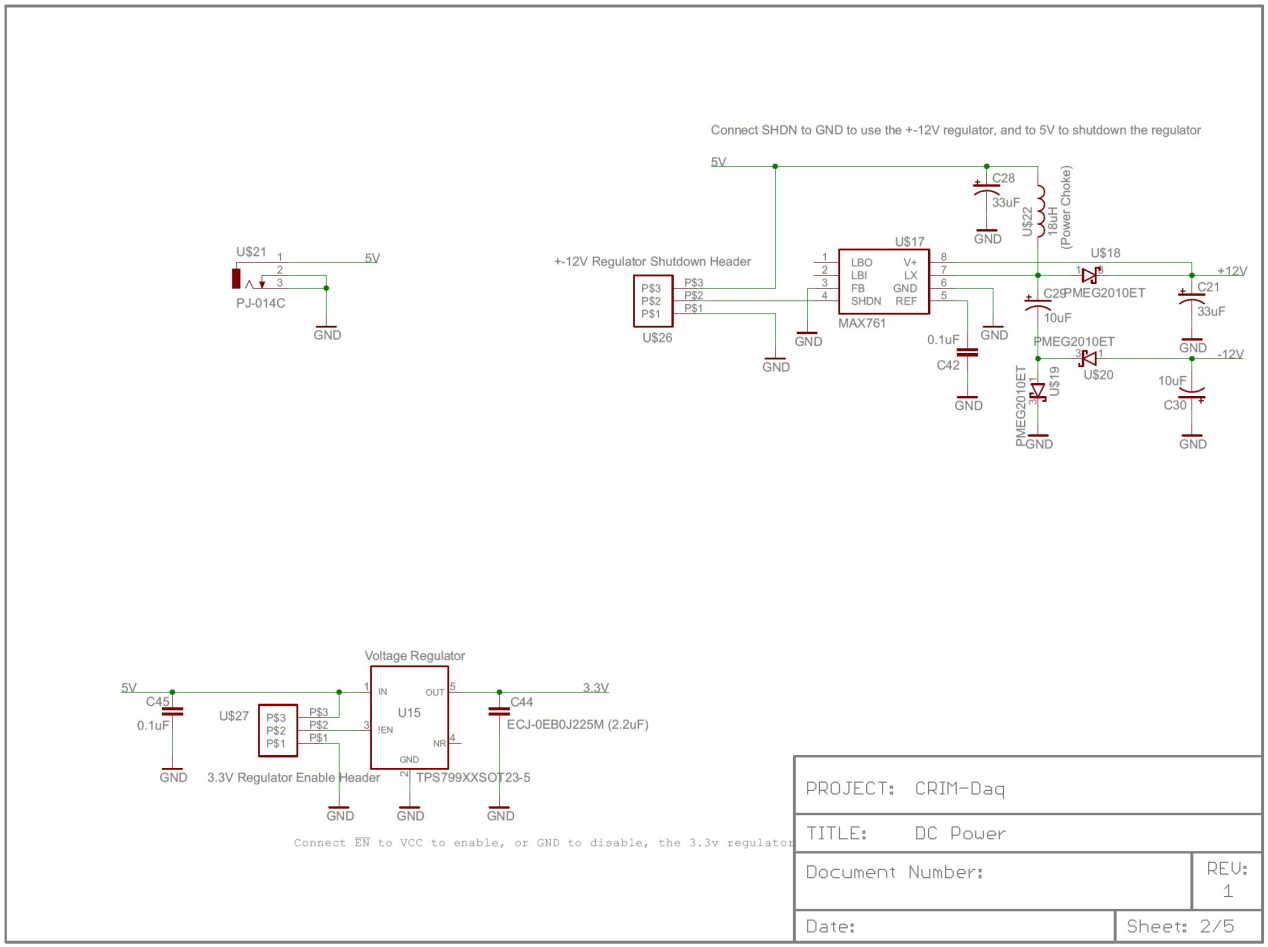
Name	Type	Training	Testing	Description
Modality Sizes	cluster			An empty structure used to specify the container for the various modality sizes
Power Modality	enum	power	power	Enum used to link "Power Size" to the power modality
Power Size (x)	int32	3	3	Number of columns in the power modality values matrix
Power Size (y)	int32	1	1	Number of rows in the power modality values matrix
Motor Modality	enum	motor	motor	Enum used to link "Motor Size" to the motor modality
Motor Size (x)	int32	2	2	Number of columns in the motor modality values matrix
Motor Size (y)	int32	1	1	Number of rows in the motor modality values matrix
Olfactory Modality	enum	olfactory	olfactory	Enum used to link "Olfactory Size" to the olfactory modality
Olfactory Size (x)	int32	9	9	Number of columns in the olfactory modality values matrix
Olfactory Size (y)	int32	1	1	Number of rows in the olfactory modality values matrix

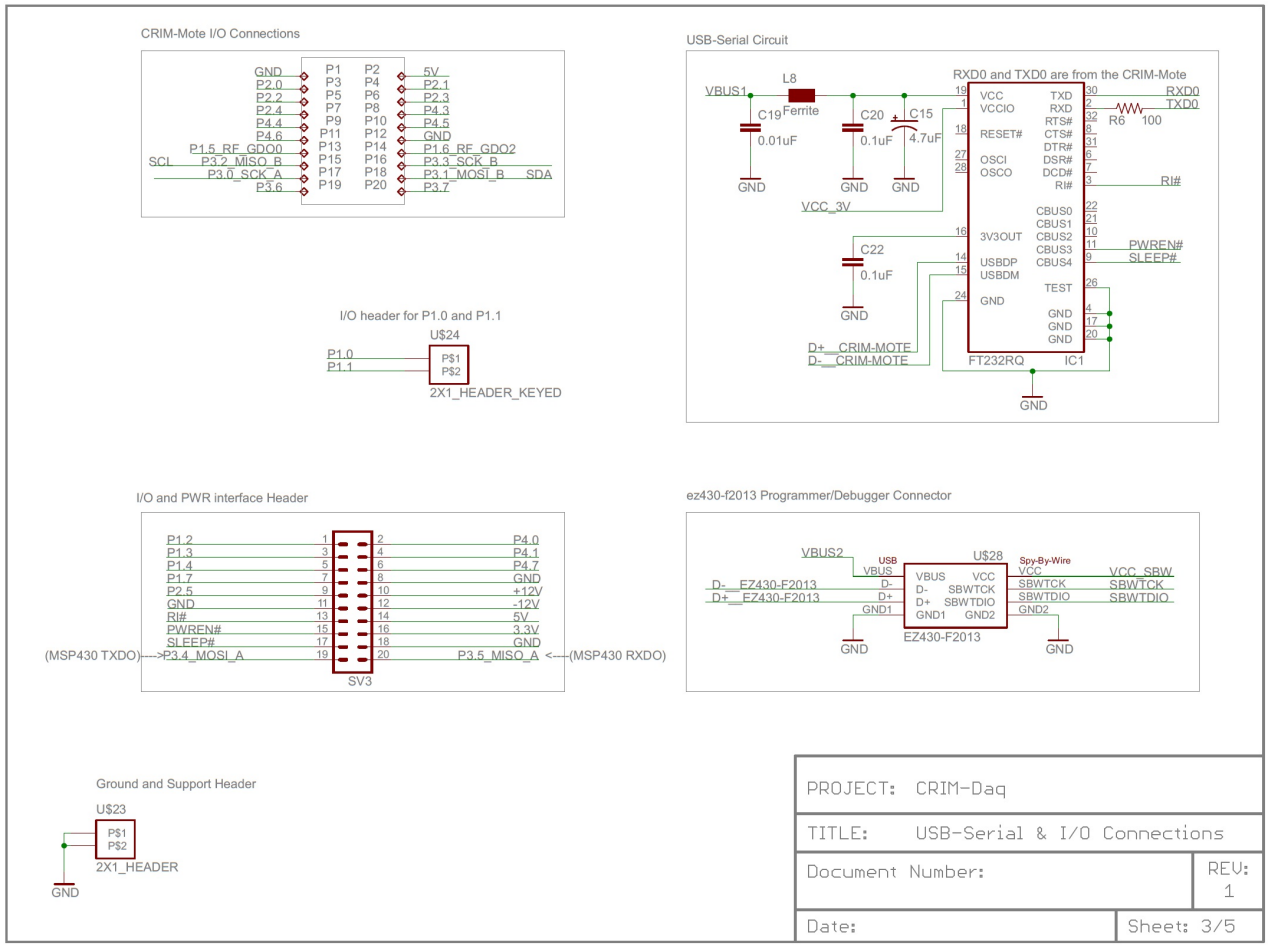
Appendix C

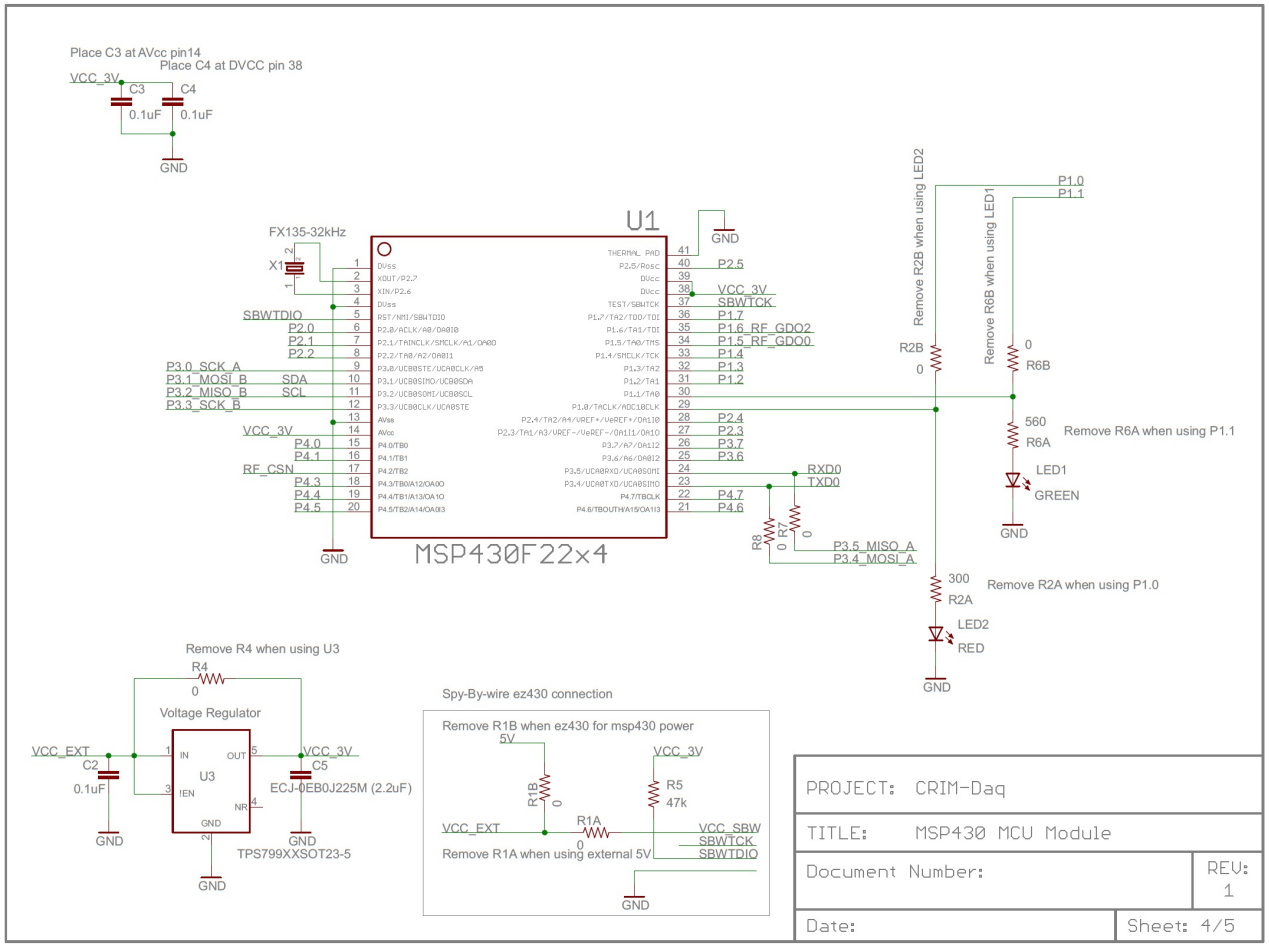
CRIM-Daq

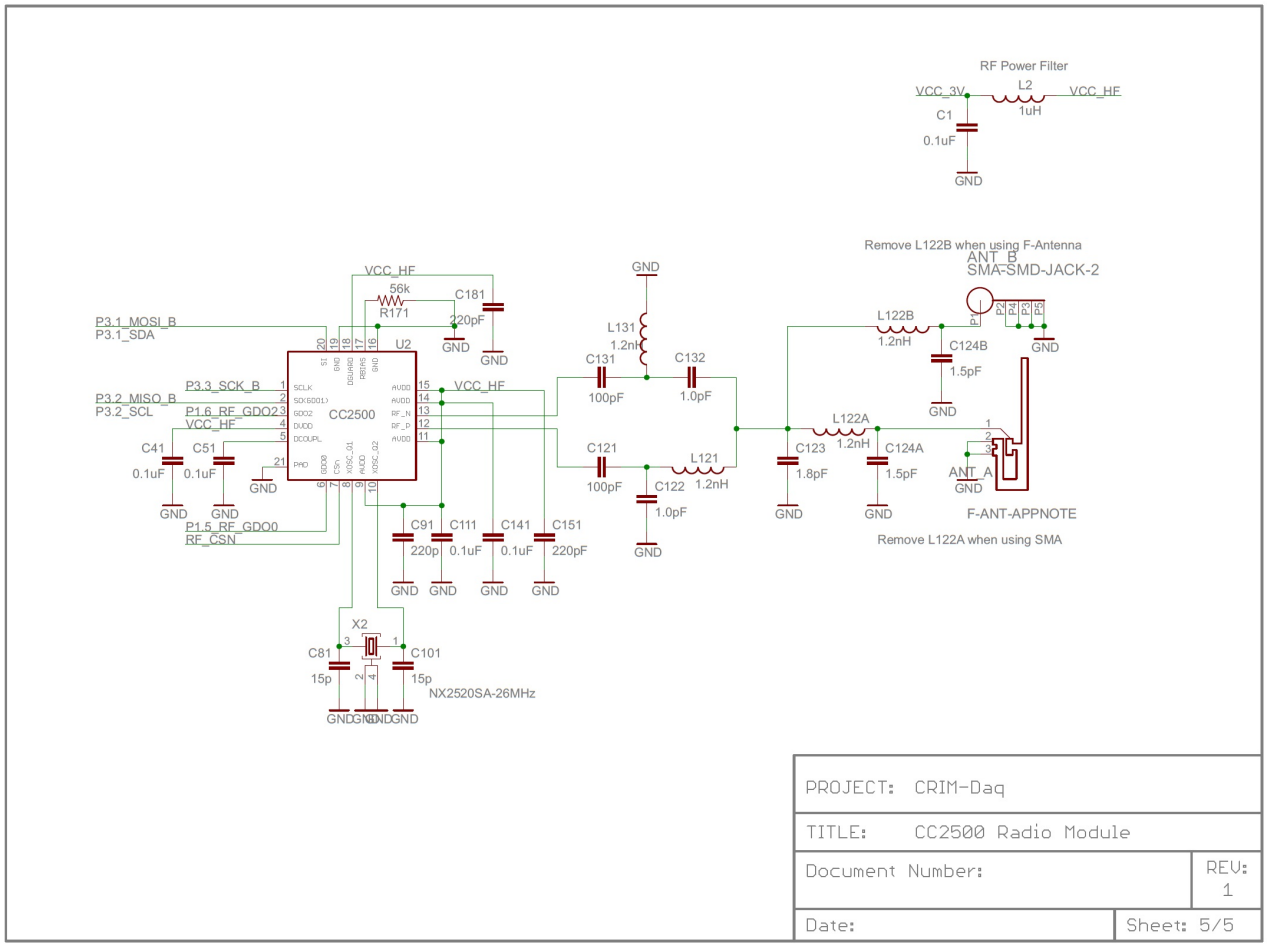
C.1 CRIM-Daq Schematic

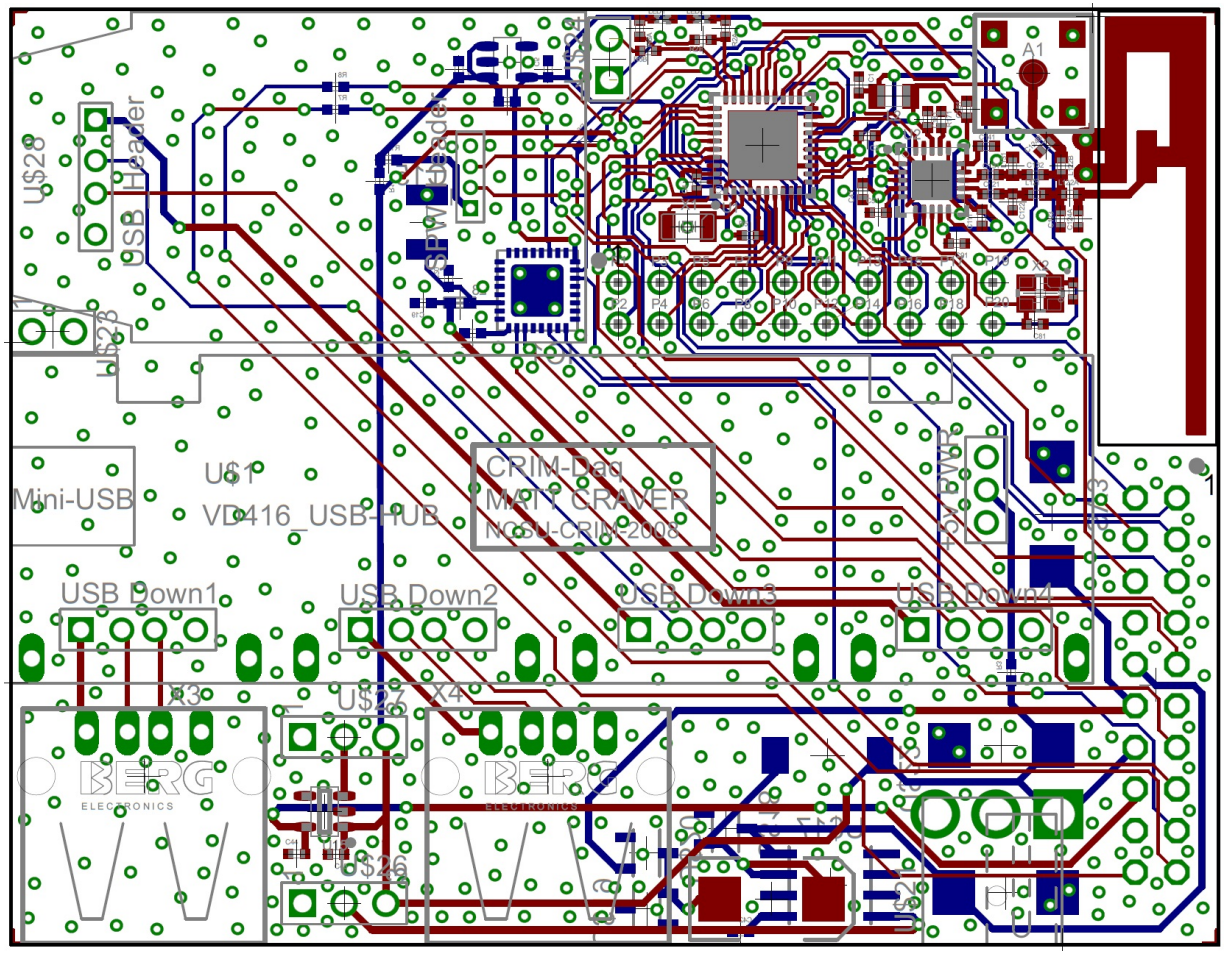



Connect SHDN to GND to use the +-12V regulator, and to 5V to shutdown the regulator
5V
C28
33uF
GND
U$22
18uH
(Power Choke)
U$17
U$18
+12V
+-12V Regulator Shutdown Header
P$3
P$2
P$1
U$26
LBO
LBI
FB
SHDN
V+
LX
GND
REF
MAX761
PMEG2010ET
C25
10uF
C21
33uF
0.1uF
C42
PMEG2010ET
U$20
U$19
10uF
C30
-12V
U$21
PJ-014C
Voltage Regulator
C45
0.1uF
U$27
U15
IN
OUT
!EN
NR
C44
ECJ-0EB0J225M (2.2uF)
3.3V
3.3V Regulator Enable Header
TPS799XXSOT23-5
Connect EN to VCC to enable, or GND to disable, the 3.3v regulator
PROJECT: CRIM-Daq
TITLE: DC Power
Document Number:
REV: 1
Date:
Sheet: 2/5


CRIM-Mote I/O Connections
GND P1 P2 5V
P2.0 P3 P4 P2.1
P2.2 P5 P6 P2.3
P2.4 P7 P8 P4.3
P4.4 P9 P10 P4.5
P4.6 P11 P12 GND
P1.5_RF_GDO0 P13 P14 P1.6_RF_GDO2
SCL P3.2_MISO_B P15 P16 P3.3_SCK_B
P3.0_SCK_A P17 P18 P3.1_MOSI_B SDA
P3.6 P19 P20 P3.7
USB-Serial Circuit
RXD0 and TXD0 are from the CRIM-Mote
VBUS1
L8
Ferrite
C19
0.01uF
C20
0.1uF
C15
4.7uF
GND
VCC_3V
VCC
VCCIO
RESET#
OSCI
OSCO
3V3OUT
USBDP
USBDM
GND
TXD
RXD
RTS#
CTS#
DTR#
DSR#
DCD#
R#
CBUS0
CBUS1
CBUS2
CBUS3
CBUS4
TEST
RXD0
TXD0
R6
100
RI#
PWREN#
SLEEP#
C22
0.1uF
D+ CRIM-MOTE
D- CRIM-MOTE
FT232RQ
IC1
I/O header for P1.0 and P1.1
U$24
P1.0
P1.1
P$1
P$2
2X1_HEADER_KEYED
I/O and PWR interface Header
P1.2 P4.0
P1.3 P4.1
P1.4 P4.7
P1.7 GND
P2.5 +12V
GND -12V
RI# 5V
PWREN# 3.3V
SLEEP# GND
(MSP430 TXDO) P3.4_MOSI_A P3.5_MISO_A (MSP430 RXDO)
SV3
ez430-f2013 Programmer/Debugger Connector
VBUS2
U$28
USB
Spy-By-Wire
D- EZ430-F2013
D+ EZ430-F2013
VBUS
D-
D+
GND1
GND2
VCC
SBWTCK
SBWTDIO
VCC_SBW
SBWTCK
SBWTDIO
EZ430-F2013
Ground and Support Header
U$23
P$1
P$2
2X1_HEADER
GND
PROJECT: CRIM-Daq
TITLE: USB-Serial & I/O Connections
Document Number:
REV: 1
Date:
Sheet: 3/5


Place C3 at AVcc pin14
Place C4 at DVCC pin 38
VCC_3V
C3
0.1uF
C4
0.1uF
GND
U1
FX135-32kHz
X1
MSP430F22x4
SBWTDIO
P2.0
P2.1
P2.2
P3.0_SCK_A
P3.1_MOSI_B
SDA
P3.2_MISO_B
SCL
P3.3_SCK_B
RF_CSN
P4.0
P4.1
P4.3
P4.4
P4.5
P2.5
VCC_3V
SBWTCK
P1.7
P1.6_RF_GDO2
P1.5_RF_GDO0
P1.4
P1.3
P1.2
P2.4
P2.3
P3.7
P3.6
RXD0
TXD0
P4.7
P4.6
R8
R7
P3.5_MISO_A
P3.4_MOSI_A
P1.0
P1.1
Remove R2B when using LED2
Remove R6B when using LED1
R2B
R6B
R6A
Remove R6A when using P1.1
LED1
GREEN
R2A
Remove R2A when using P1.0
LED2
RED
Remove R4 when using U3
R4
Voltage Regulator
VCC_EXT
C2
U3
IN
OUT
EN
NR
GND
C5
ECJ-0EB0J225M (2.2uF)
TPS799XXSOT23-5
Spy-By-wire ez430 connection
Remove R1B when ez430 for msp430 power
5V
R1B
R5
47k
R1A
Remove R1A when using external 5V
VCC_SBW
SBWTCK
SBWTDIO
PROJECT: CRIM-Daq
TITLE: MSP430 MCU Module
Document Number:
REV: 1
Date:
Sheet: 4/5

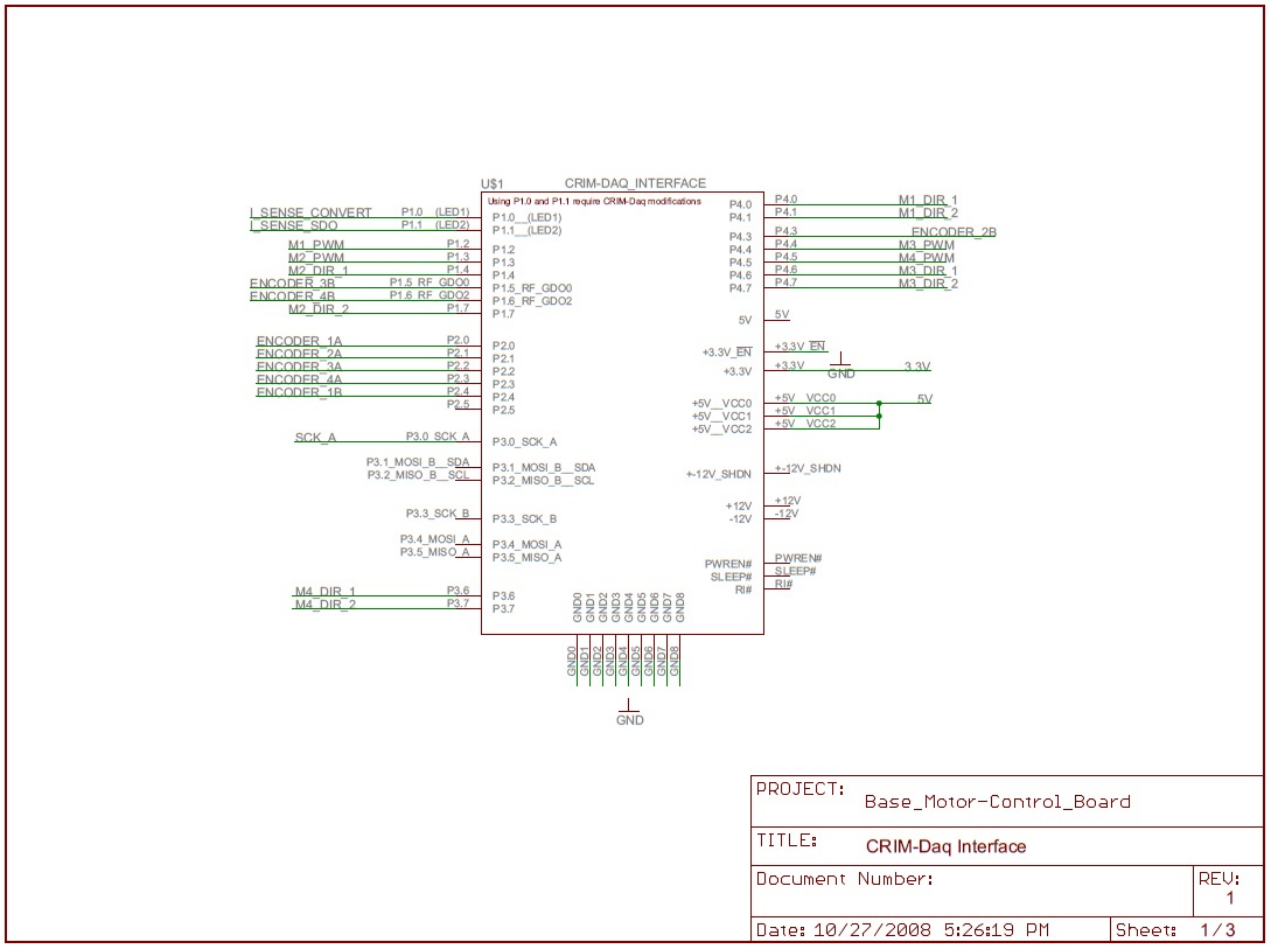

RF Power Filter
VCC_3V
L2
1uH
VCC_HF
C1
0.1uF
GND
Remove L122B when using F-Antenna
ANT_B
SMA-SMD-JACK-2
VCC_HF
56k
R171
C181
220pF
P3.1_MOSI_B
P3.1_SDA
U2
P3.3_SCK_B
P3.2_MISO_B
P3.2_SCL
P1.6_RF_GDO2
VCC_HF
CC2500
C41
C51
0.1uF
0.1uF
P1.5_RF_GDO0
RF_CSN
C91
C111
C141
C151
220p
0.1uF
0.1uF
220pF
X2
C81
C101
15p
15p
NX2520SA-26MHz
L131
1.2nH
C131
100pF
C132
1.0pF
C121
100pF
L121
1.2nH
C122
1.0pF
L122B
1.2nH
C124B
1.5pF
L122A
1.2nH
C123
1.8pF
C124A
1.5pF
ANT_A
F-ANT-APPNOTE
Remove L122A when using SMA
PROJECT: CRIM-Daq
TITLE: CC2500 Radio Module
Document Number:
REV: 1
Date:
Sheet: 5/5

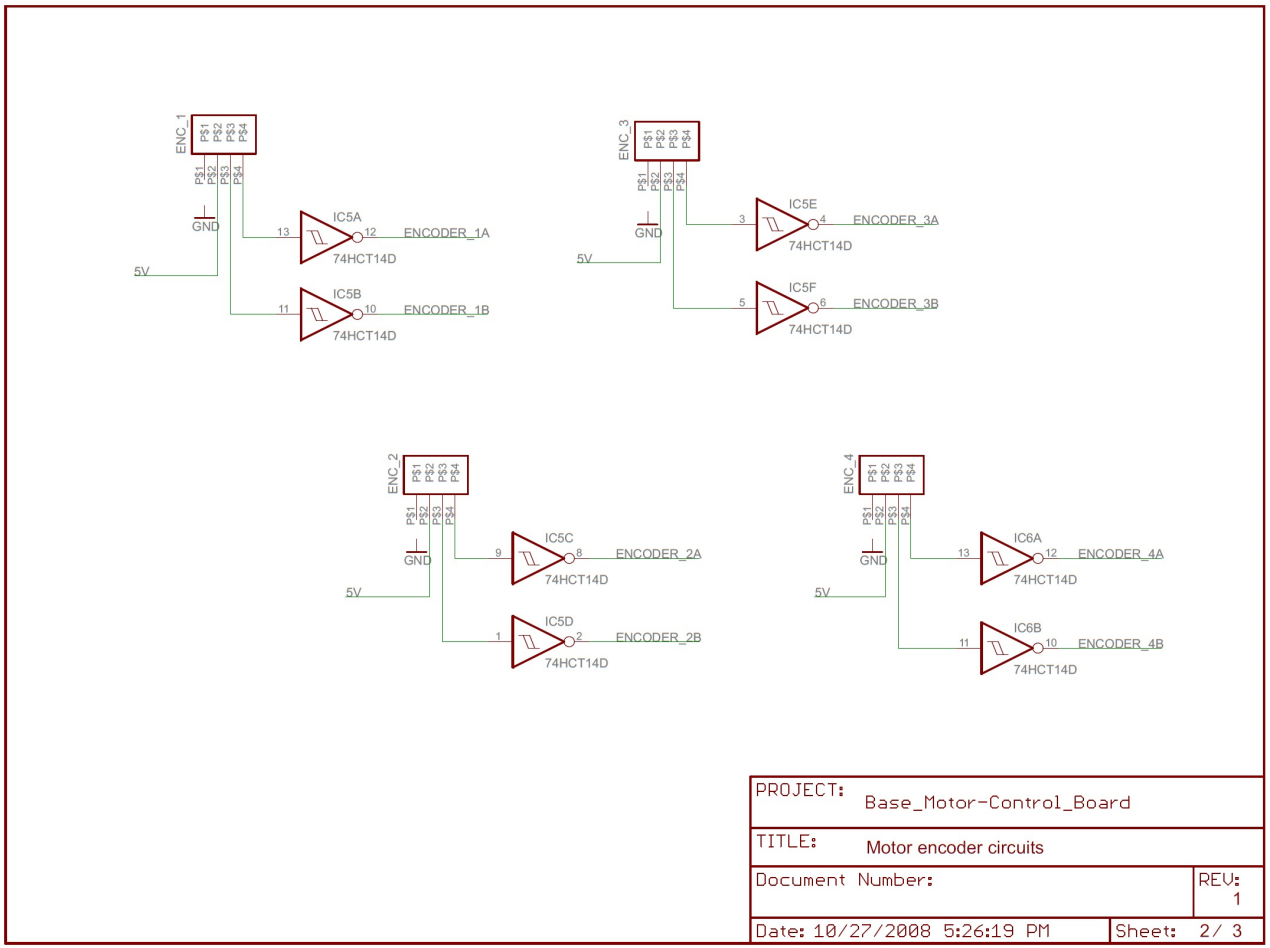

USB Heater
U$28
U$1
VD416_USB-HUB
Mini-USB
CRIM-Dao
MATT CRAVER
NCSU-CRIM-2008
USB Down1
USB Down2
USB Down3
USB Down4
X3
X4
U$27
U$26
A1

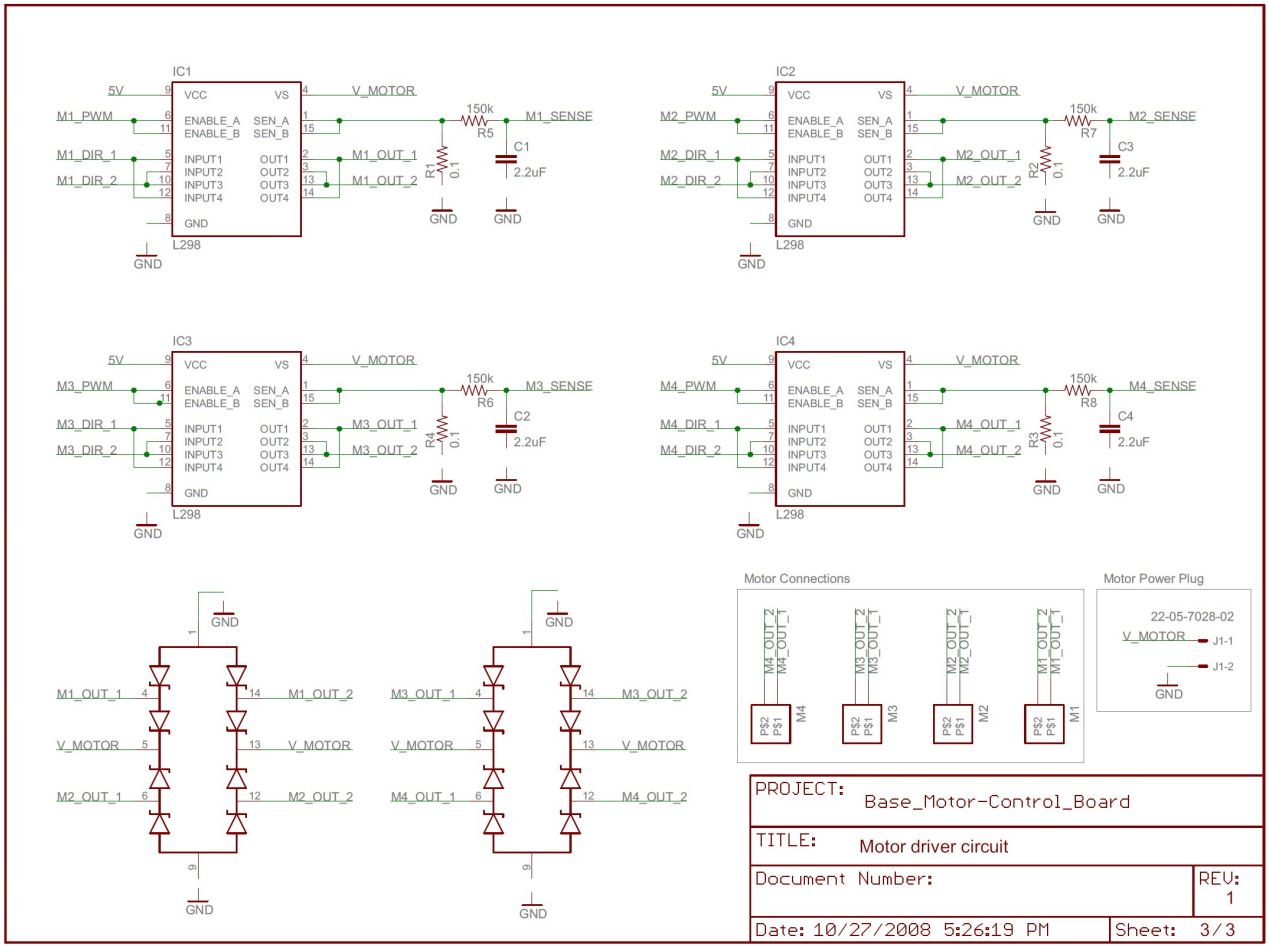
Appendix D

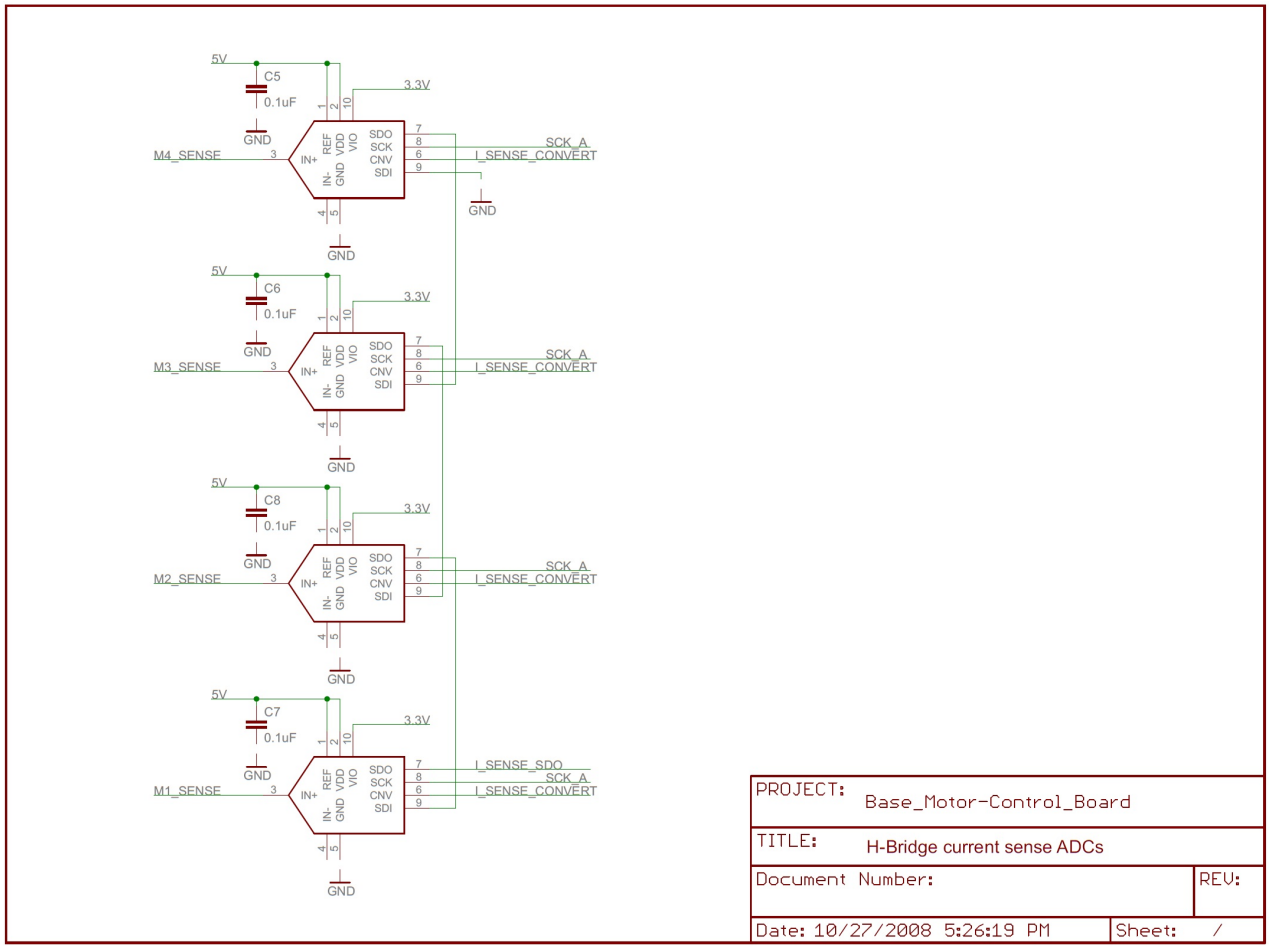
CRIM-Daq Motor Control Board

D.1 CRIM-Daq Motor Control Board Schematic

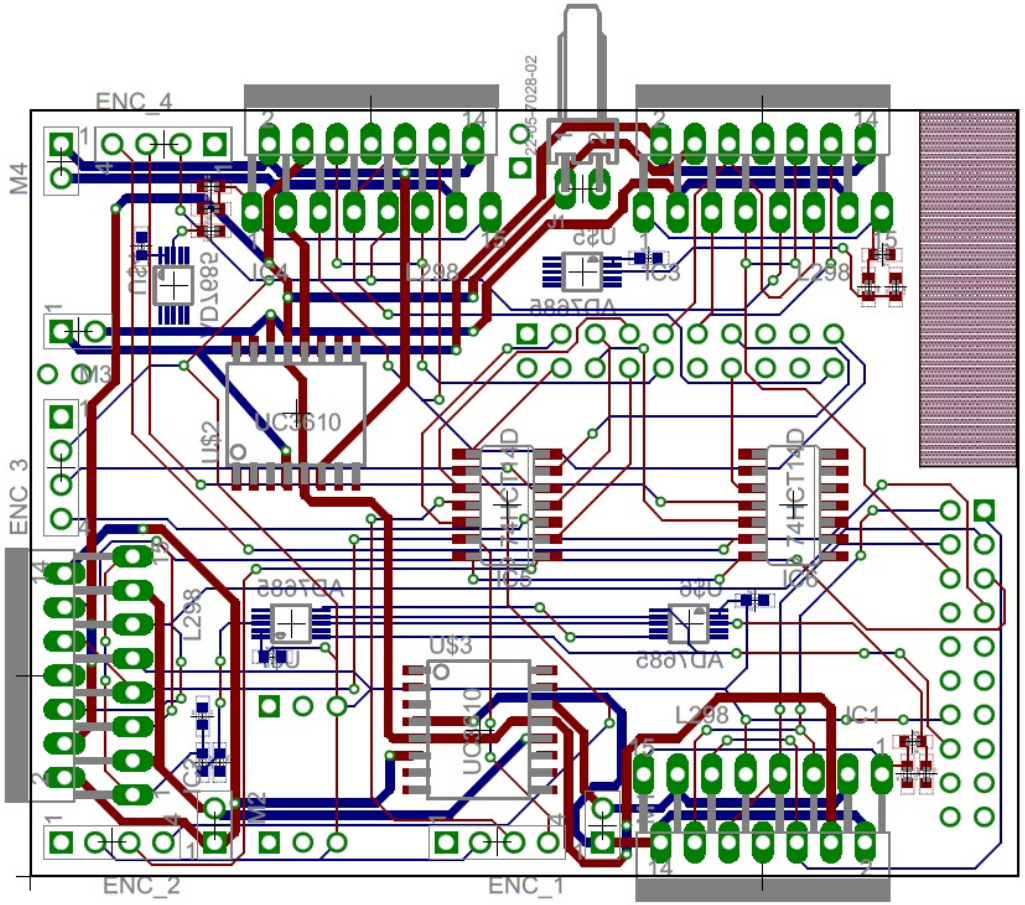



ENC_1
PS1
PS2
PS3
PS4
GND
5V
IC5A
74HCT14D
ENCODER_1A
IC5B
74HCT14D
ENCODER_1B
ENC_3
IC5E
74HCT14D
ENCODER_3A
IC5F
74HCT14D
ENCODER_3B
ENC_2
IC5C
74HCT14D
ENCODER_2A
IC5D
74HCT14D
ENCODER_2B
ENC_4
IC6A
74HCT14D
ENCODER_4A
IC6B
74HCT14D
ENCODER_4B
PROJECT: Base_Motor-Control_Board
TITLE: Motor encoder circuits
Document Number:
REV: 1
Date: 10/27/2008 5:26:19 PM
Sheet: 2/ 3


IC1
IC2
IC3
IC4
L298
VCC
VS
ENABLE_A
ENABLE_B
SEN_A
SEN_B
INPUT1
INPUT2
INPUT3
INPUT4
OUT1
OUT2
OUT3
OUT4
GND
5V
V_MOTOR
150k
0.1
2.2uF
M1_PWM
M1_DIR_1
M1_DIR_2
M1_SENSE
M1_OUT_1
M1_OUT_2
M2_PWM
M2_DIR_1
M2_DIR_2
M2_SENSE
M2_OUT_1
M2_OUT_2
M3_PWM
M3_DIR_1
M3_DIR_2
M3_SENSE
M3_OUT_1
M3_OUT_2
M4_PWM
M4_DIR_1
M4_DIR_2
M4_SENSE
M4_OUT_1
M4_OUT_2
Motor Connections
Motor Power Plug
22-05-7028-02
J1-1
J1-2
PS2
PS1
M4
M3
M2
M1
PROJECT:
Base_Motor-Control_Board
TITLE:
Motor driver circuit
Document Number:
REV:
1
Date: 10/27/2008 5:26:19 PM
Sheet:
3/3


5V
C5
0.1uF
GND
M4_SENSE
3
IN+
IN-
REF
VDD
VIO
GND
SDO
SCK
CNV
SDI
1
2
10
4
5
7
8
6
9
3.3V
SCK_A
I_SENSE_CONVERT
GND
5V
C6
0.1uF
GND
M3_SENSE
3
3.3V
SCK_A
I_SENSE_CONVERT
GND
5V
C8
0.1uF
GND
M2_SENSE
3
3.3V
SCK_A
I_SENSE_CONVERT
GND
5V
C7
0.1uF
GND
M1_SENSE
3
3.3V
I_SENSE_SDO
SCK_A
I_SENSE_CONVERT
GND
PROJECT: Base_Motor-Control_Board
TITLE: H-Bridge current sense ADCs
Document Number:
REV:
Date: 10/27/2008 5:26:19 PM
Sheet: /

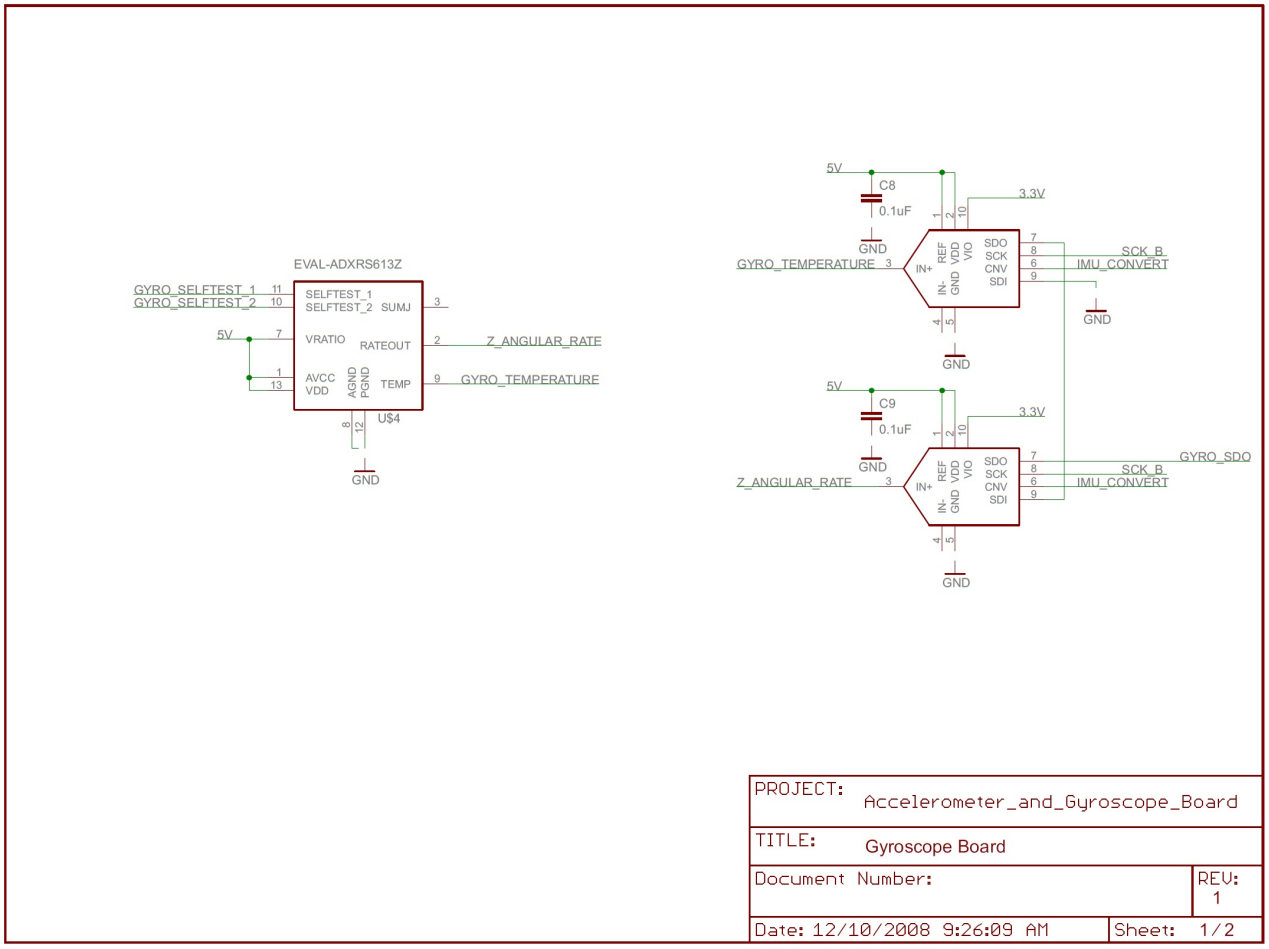
D.2 CRIM-Daq Motor Control Board Layout

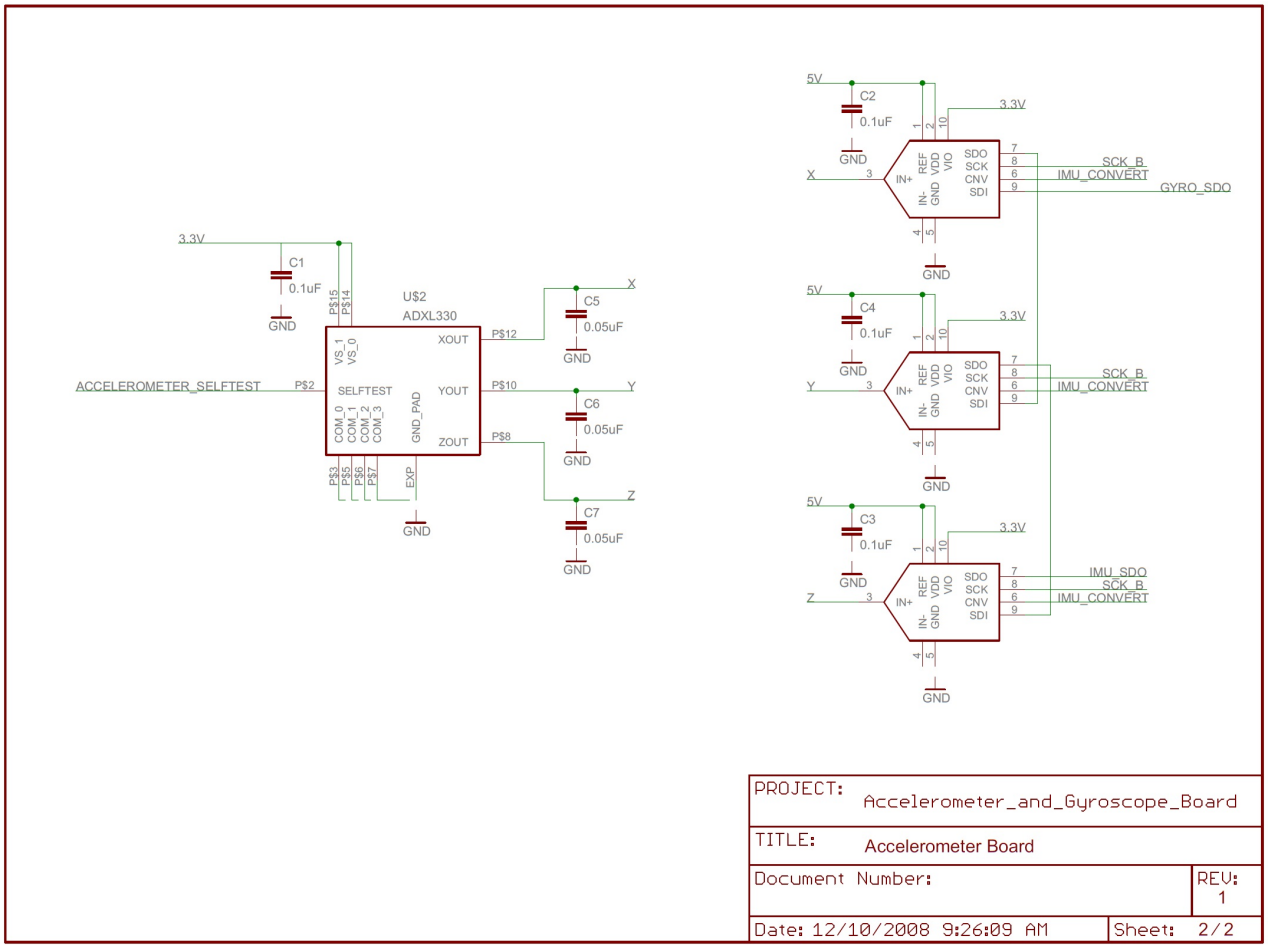


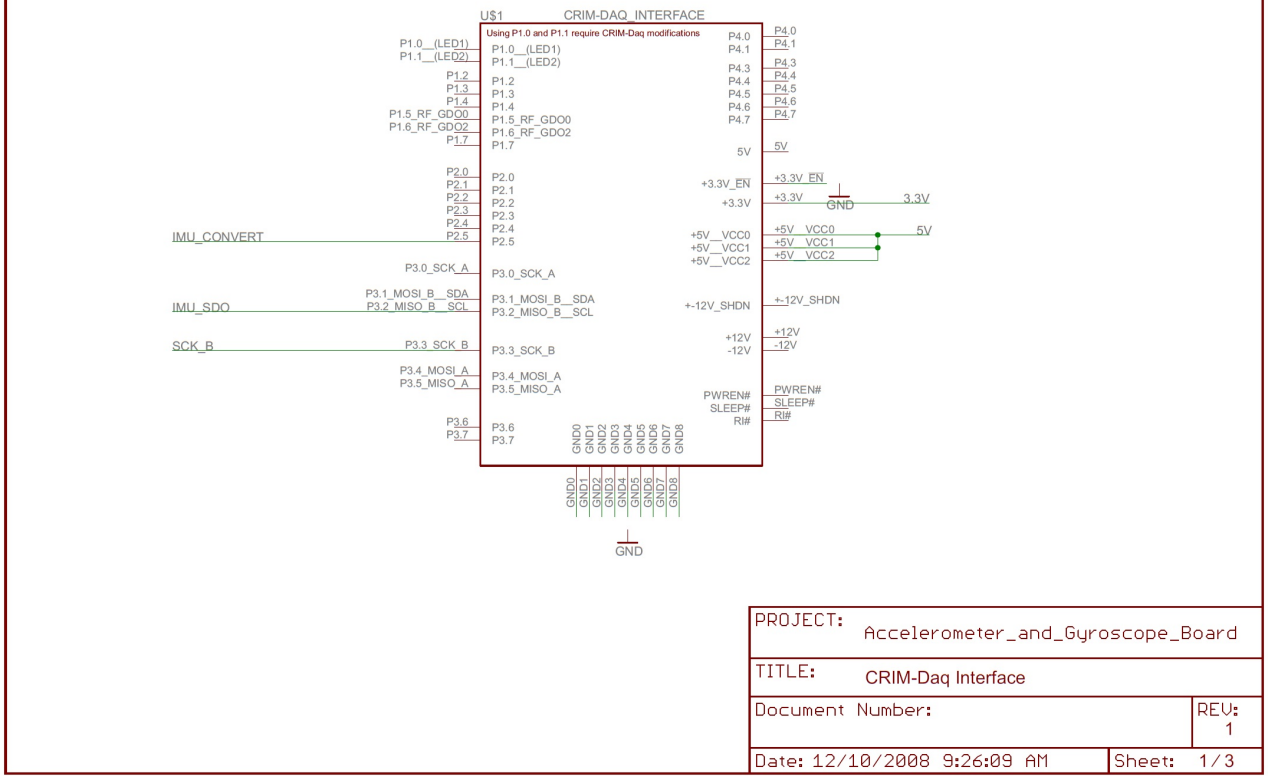
Appendix E

CRIM-Daq Inertial Measurement Unit

E.1 CRIM-Daq Inertial Measurement Unit Schematic

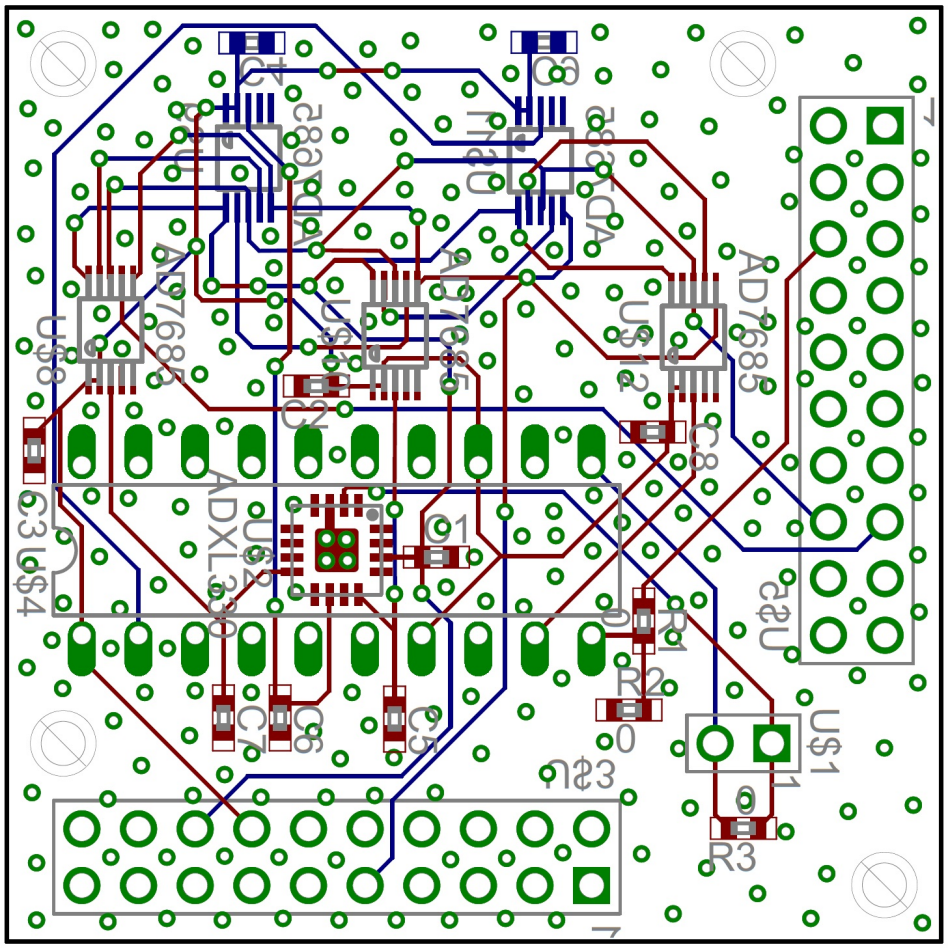



3.3V
C1
0.1uF
GND
U$2
ADXL330
ACCELEROMETER_SELFTEST
P$2
SELFTEST
XOUT
YOUT
ZOUT
GND_PAD
EXP
VS_1
VS_0
COM_0
COM_1
COM_2
COM_3
P$12
P$10
P$8
C5
C6
C7
0.05uF
X
Y
Z
5V
C2
C4
C3
SDO
SCK
CNV
SDI
REF
VDD
VIO
IN+
IN-
SCK_B
IMU_CONVERT
GYRO_SDO
IMU_SDO
PROJECT: Accelerometer_and_Gyroscope_Board
TITLE: Accelerometer Board
Document Number:
REV: 1
Date: 12/10/2008 9:26:09 AM
Sheet: 2/2



PROJECT:	Accelerometer_and_Gyroscope_Board
TITLE:	CRIM-Daq Interface
Document Number:	REV: 1
Date: 12/10/2008 9:26:09 AM	Sheet: 1/3

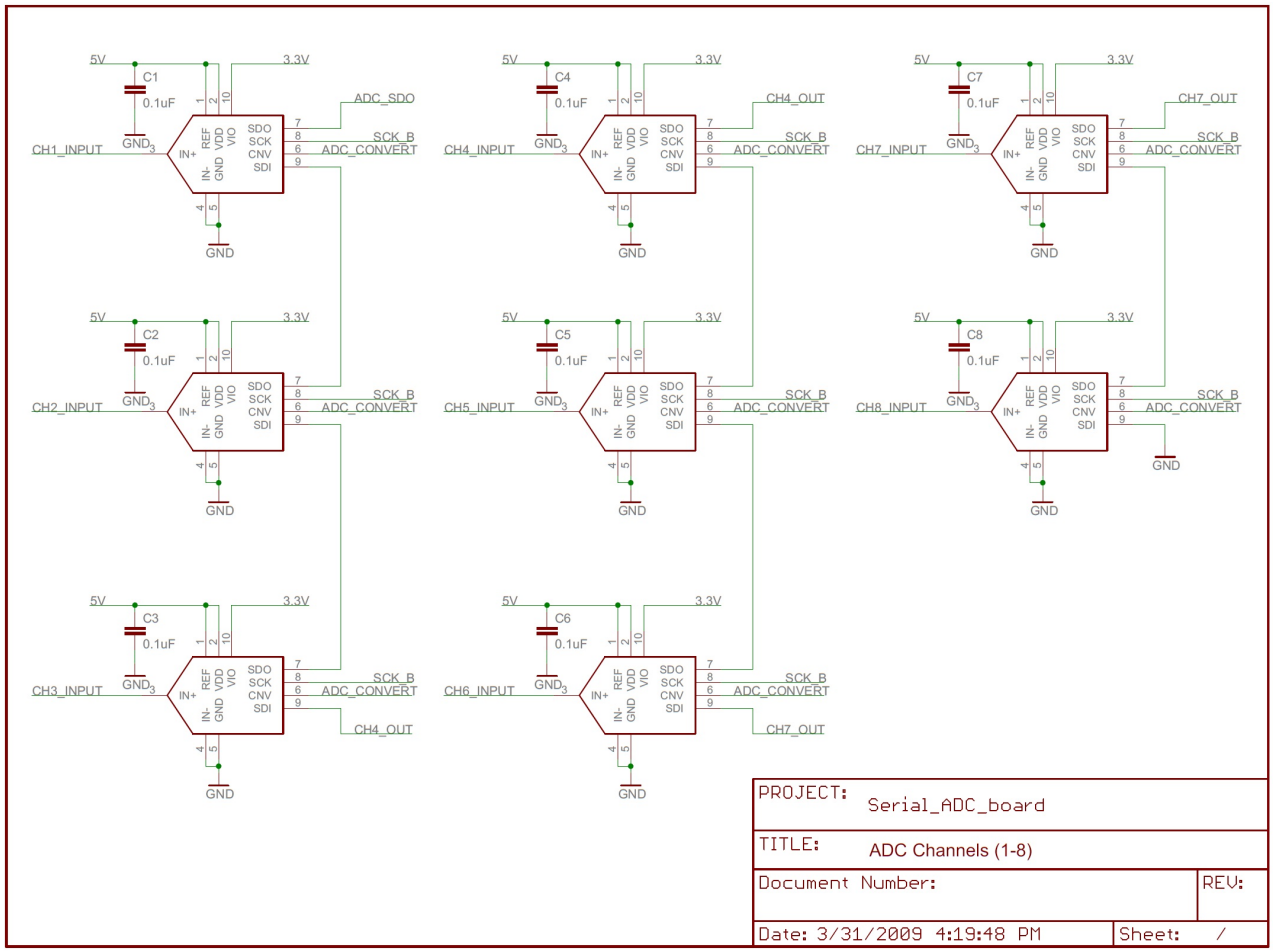
E.2 CRIM-Daq Inertial Measurement Unit Board Layout

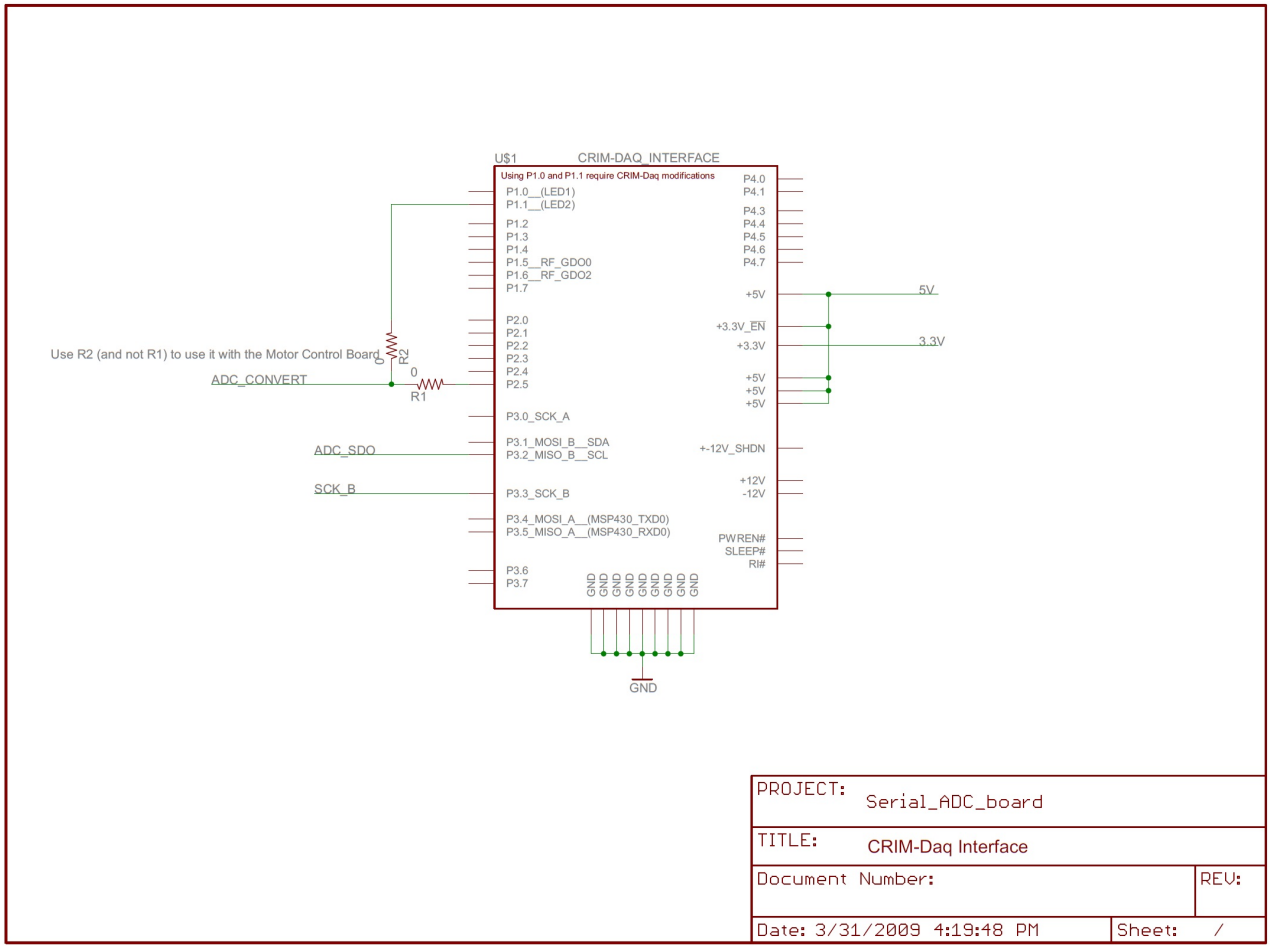


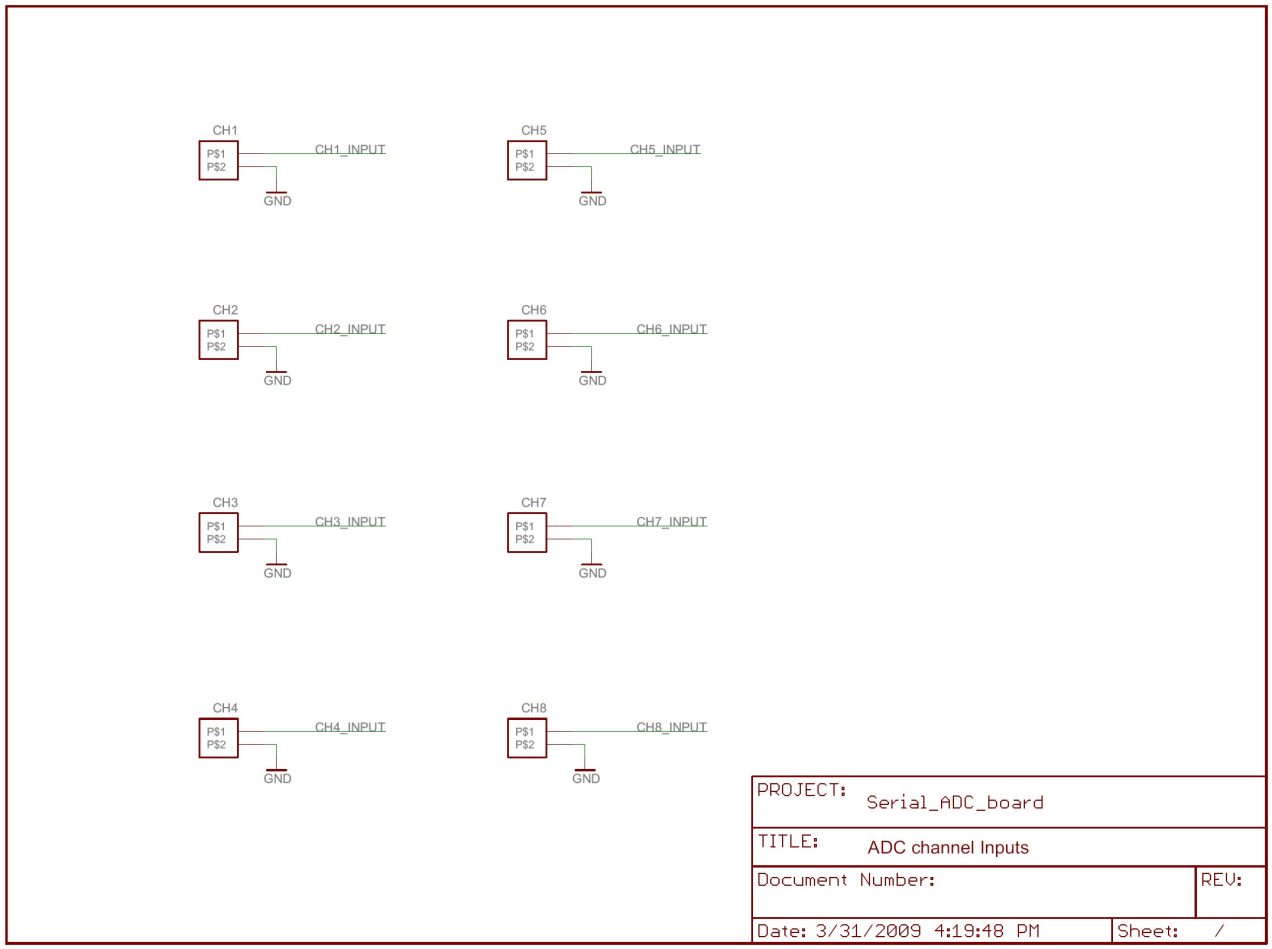
Appendix F

CRIM-Daq Serial Analog to Digital Converter Board

F.1 CRIM-Daq Serial Analog to Digital Converter Board Schematic




U$1
CRIM-DAQ_INTERFACE
Using P1.0 and P1.1 require CRIM-Daq modifications
P1.0__(LED1)
P1.1__(LED2)
P1.2
P1.3
P1.4
P1.5__RF_GDO0
P1.6__RF_GDO2
P1.7
P2.0
P2.1
P2.2
P2.3
P2.4
P2.5
P3.0_SCK_A
P3.1_MOSI_B__SDA
P3.2_MISO_B__SCL
P3.3_SCK_B
P3.4_MOSI_A__(MSP430_TXD0)
P3.5_MISO_A__(MSP430_RXD0)
P3.6
P3.7
P4.0
P4.1
P4.3
P4.4
P4.5
P4.6
P4.7
+5V
+3.3V_EN
+3.3V
+5V
+5V
+5V
+-12V_SHDN
+12V
-12V
PWREN#
SLEEP#
RI#
GND
5V
3.3V
Use R2 (and not R1) to use it with the Motor Control Board
R2
0
R1
ADC_CONVERT
ADC_SDO
SCK_B
PROJECT: Serial_ADC_board
TITLE: CRIM-Daq Interface
Document Number:
REV:
Date: 3/31/2009 4:19:48 PM
Sheet: /


CH1
P$1
P$2
CH1_INPUT
GND
CH5
P$1
P$2
CH5_INPUT
GND
CH2
P$1
P$2
CH2_INPUT
GND
CH6
P$1
P$2
CH6_INPUT
GND
CH3
P$1
P$2
CH3_INPUT
GND
CH7
P$1
P$2
CH7_INPUT
GND
CH4
P$1
P$2
CH4_INPUT
GND
CH8
P$1
P$2
CH8_INPUT
GND
PROJECT: Serial_ADC_board
TITLE: ADC channel Inputs
Document Number:
REV:
Date: 3/31/2009 4:19:48 PM
Sheet: /

F.2 CRIM-Daq Serial Analog to Digital Converter Board Layout