Security Robot

A Major Qualifying Project

Submitted to the faculty

Of the

Worcester Polytechnic Institute

In partial fulfilment of the requirements for the

Degree of Bachelor of Science

By

_____

Timothy Hannafin, Class of 2006

_____

Cheuk Wai David To, Class of 2006

_____

TsungTao Wu, Class of 2005

April 27, 2006

_____

Michael Ciaraldi
Project Advisor

Abstract

As robotic technology continues to advance, robots are becoming capable of performing ever more complex tasks. Robotic workers never get tired, do not need to be paid, and can be made to perform even the most dangerous tasks without concern. The purpose of this project was to combine several existing technologies, wireless internet, neural networks, and hardware controllers, into a system that can perform the job of a night security guard.

Table of Contents

1. Introduction

The job of a nighttime security guard is simple and boring; there are two basic tasks to be done. First, they must watch security camera feeds often enough to make sure nothing is wrong. Second, they need to patrol routes to check in person if there is anything unusual. Both of these tasks require a large amount of time and personnel to completely cover a very large and complicated building. And in certain instances, performing his job might even endanger the life of the security guard. Automating these tasks would free up time and personnel for jobs more suiting of a human.

The purpose of this project was to design and build a system to replace human guards on security routes. Replacing human guards with robotic ones will have several advantages. First, the owner of a property that is patrolled by robots will require fewer human workers in that role and will be able to devote surplus employees to tasks more suited to human-level intelligence. Second, human security guards cannot constantly be on patrol, unlike robots which are limited only by their battery life. Thirdly, with internet control capability, an operator can monitor the security of a property from anywhere in the world instead of having a security guard on site. Fourth, a human guard can control multiple robots, increasing efficiency. Lastly, in the event of a dangerous occurrence, only an easily replaceable robot will be threatened instead of a human life. By physically removing the human operator from a potentially dangerous environment, the robotic security guard has the ability to save lives.

Any robot meant to replace a security guard would have to have a few key features. First, the robot must have the ability to move autonomously and avoid obstacles such as

walls. Second, it would need to be capable of following defined routes like its human counterpart. Third, a human must be able to take control of the system at any time and pilot it to where it is needed. Lastly, and most importantly, the robot must be able to recognize when it encounters something that warrants the intervention of a human. The prototype built in this project meets all these requirements. It avoids obstacles and follows a predefined route using a variety of sensors. It allows for remote control using a client program over a wireless internet connection. And it detects anomalies by using a neural network to analyze an image stream taken from an onboard camera.

2. Background

2.1 History of Robots

The word “robot” was first coined in 1921 by the Czech playwright Karel Čapek. It comes from the Czech word “robota” meaning labor[1]. The Robot Institute of America defines a robot as "A reprogrammable, multifunctional manipulator designed to move materials, parts, tools, or specialized devices through various programmed motions for the performance of a variety of tasks." Machines fitting this description can be dated back to ancient Greek clepsydra water clocks through which the constant rate at which the water flowed could be changed[2]. However, the earliest modern robot fitting this description was a remotely operated boat invented by Nicola Tesla in 1898. At the 1939 World’s Fair the Westinghouse Corporation exhibited Elektro, the world’s first operational humanoid robot[3]. Elektro was controlled via a pair of cables attached to his right foot. The entire machine contained only 6 motors and could “walk” using a pair of rollers in each foot, open and close both hands, and even smoke with the aid of a small air pump inside the mouth.

The first autonomous robots to react to outside stimulus were built by neurophysiologist W. Grey Walter in the late 1940s. The 3-wheeled, bubble-bodied machines were referred to as “tortoises”. Walter’s purpose in building these robots was

[1] http://en.wikipedia.org/wiki/Robot
[2] http://www.britannica.com/clockworks/clepsydra.html
[3] http://pages.cpsc.ucalgary.ca/~jaeger/visualMedia/robotHistory.html

to show how a set of simple instructions can create a complex behavior. The robots were programmed to drive towards a light if a light was visible, back away and turn if they bumped into something, and drive forward oscillating from side to side looking for light if none was visible[4]. Walter's experiments with these uncomplicated machines showed that the root of impressively complex behavior can be astoundingly simple.

Today the descendants of these simple robots have permeated the modern world. Robots do the repetitive, labor intensive jobs on assembly lines. It is estimated that by the end of 2006, the number of industrial robots world-wide will number over 875,000[5]. They do jobs too dangerous for humans, such as defusing bombs or exploring the depths of the ocean. They even clean our houses. And in the future they will protect our property as security sentinels.

2.2 History of Neural Networks

Humans have been studying the structure of the human brain and how it works for thousands of years. The brain is made up of groups of neurons with roughly 100-billion nerve cells. These nerve cells have the amazing ability to gather and transmit electrochemical signals like the gates and wires in a computer. The electrochemical aspect of these cells lets them transmit signals up to several feet and pass messages to each other. With the advancement of modern electronic technology, humans were able to mimic the behavior of these neurons with simple electric circuits. In 1943 Warren McCulloch, a neurophysiologist and mathematician, and Walter Pitts, wrote a paper on

[4] http://www.cerebromente.org.br/n09/historia/documentos_i.htm
[5] http://www.ifr.org/generalInformation/missState.htm

how neurons work. They modeled a simple neural network according to their theory with electrical circuits. In 1949 Donald Hebb published Organization of Behavior which pointed out that neural pathways strengthen each time that they are used.

In the 1950's, as the advancement of computer technology continued, it became possible to put the theory into practice and actually model brain processes. The first attempt, led by Nathanial Rochester from the IBM research laboratories, failed. However, the later attempts were successful. Despite these successes, the contemporary advancement of traditional computing drew attention away from neural networks. There was no apparent need for a highly complex analysis structure when processing speeds continued to jump higher. In 1956 the Dartmouth Summer Research Project on Artificial Intelligence provided a push to both artificial intelligence and neural networks. One of the outcomes of the process was to stimulate research in AI, and the neural processing of the brain. After the Dartmouth Project, John von Neumann suggested imitating neural function by using telegraph relays or vacuum tubes. Later that year, Frank Rosenblatt, a neurobiologist at Cornell, began work on the Perceptron. The Perceptron was a hardware device and it is the oldest neural network concept that is still in use today. A network with a single layer of perceptrons was found to be useful for a continuous set of inputs which it puts into one of two classes. The perceptron computes a weighted sum of the input and subtracts a threshold then passes one of the possible values as result. Unfortunately, the perceptron was limited and later disproved by Marvin Minsky and Seymour Papert's 1969 book Perceptrons. The first neural network to be applied to a real problem was MADALINE. MADALINE was an adaptive filter to eliminate echoes on a

phone line. This network was developed by Widrow and Marcian Hoff of Stanford in 1959. It is still in commercial use till today.

The early successes caused people to exaggerate the potential of the network. This exaggeration also made its way to the general public. The public started to fear what disasters these “thinking machine” would bring upon mankind. Asimov's series on robots revealed the effects on man's morals and values when machines where capable of doing all of mankind's work. These fears, combined with outrageous claims, and unfulfilled promises, caused scientists and the public to start to criticize neural network research. The funding of research was halted as a result of this criticism. This period of stunted growth lasted until the 1980’s.

In 1982, John Hopfield of Caltech presented a paper to the national Academy of Sciences, and revived interest in neural networks. He had a plan to not just simply mimic the human brain but to create useful devices. With brilliant mathematical analysis, he showed how such networks could be built and what they might be capable of. Around that same time, the US-Japan Joint Conference on Cooperative/Competitive Neural Networks was held in Kyoto, Japan. There, the Japanese announced their Fifth Generation effort. The US government worried about being left behind; soon increased funding and neural network research once again began to flourish.

By 1985, an annual meeting called “Neural Networks for Computing” was held by the American Institute of Physics. By 1987, the first International Conference on Neural Networks held by the Institute of Electrical and Electronic Engineer's (IEEE) drew more than 1800 attendees. By 1989 at the Neural Networks for Defense meeting, Bernard Widrow told his audience that they were engaged in World War IV, since "World War III

never happened", where the battlefields were world trade and manufacturing. The 1990 US Department of Defense Small Business Innovation Research Program named 16 topics which specifically targeted neural networks with an additional 13 mentioning the possible use of neural networks.

Today, neural networks discussions are occurring everywhere, and many promise a bright future on what such technology is capable of. However, the future of the network is controlled by hardware development. This research is developing neural networks that, due to processing limitations, take weeks to learn, and to put them into actual use requires specialized chips. Companies are working on three types of neuro chips - digital, analog, and optical. Some companies are working on creating a "silicon compiler" to generate a neural network Application Specific Integrated Circuit (ASIC). These ASICs and neuron-like digital chips appear to be the wave of the near future. Ultimately, optical chips look very promising. Yet, it may be years before optical chips see the light of day in commercial applications. Despite their drawbacks, the learning capability of neural networks makes it an extremely powerful tool for information analysis with applications limited only by the imagination[6].

[6] http://en.wikipedia.org/wiki/Neural_network

3. Methodology

Although this was primarily a computer science project, it started from scratch, with no hardware on which on base the hardware. A robot had to be constructed that had a high enough power to move all the robot hardware as well as the necessary surveillance equipment. It was believed the size of robot would be relatively small because this was only a prototype, and a smaller robot would require less power and construction expense. The robot needed be at least high enough to offer an onboard camera enough visibility of the hallway in front of it to se when there is a person in it. The robot would operate wirelessly on battery power. For testing purposes, the prototype would be able to run for at least one hour before needing to be recharged.

3.1 The Robot Hardware

Several different solutions were considered in the process of designing this prototype security robot, the primary concerns were the cost and the ease of construction since all team members are Computer Science majors and had little to no experience in robot-building. To minimize complications with embedded systems, it was decided to use an onboard computer to do the main processing and wireless communication on the robot. There were two primary choices for the onboard processor: a laptop computer or a mini computer based on an ITX board. The laptop was selected, after it was decided that the portability, compactness, and preassembled form were all desirable.

A project in Japan was found that had many similarities to this project, a robot that is Internet-controllable through a flash player[7].

Fig 3.1-1: A robot built on a remote controlled car chassis.

From this, came the initial design; to purchase a low-price and large remote control vehicle. Then, disassemble it and integrate the necessary control modules on the vehicles and turn it into a functional robot. It was later decided that the poor alignments on toy vehicles might make the robot running in a straight line a challenge, and it was potentially more difficult to control the complex turning mechanism. It was then decided to look into remote-control units with tank-like steering mechanisms.

A remote-control toy tank was found which was priced reasonably and at 32 inches long, was more than big enough to carry all the necessary equipment. However, detailed information on the parts this tank used was unavailable since the manufacturer could not be located. It was decided that this lack of information could lead to unforeseen problems later on ad this chassis was also abandoned.

[7] http://www.marumushi.com/apps/remotedriver2

Further into the term after an introduction from Prof. Ciaraldi, we met up with Ken Stafford and Brad Miller, who are robotics experts at WPI, and they believed that it could be very difficult to implement the initial ideas since retail toy units likely to have all their electronics integrated on one circuit board. Instead, they suggested the use of a VEX robotic kit[8]. These kits were designed for rapid building of simple robots. The only parts of the kit needed for this project were the structural hardware and motors. The movement of the robot would be provided by the Vex Robotics Multi-Speed motors. These motors are designed to run at between 5.5 and 9 volts. A standard rechargeable 7.2 volt Nickel-Metal Hydride battery pack would supply power to the motors. At 7.2v the max speed of the motor is about 100RPM. The stall torque of each motor is 6.5in-lbs, so they were far from powerful, even with a motor directly driving all 4 wheels; weight would be a major concern and the total weight should be limited to less than 10 lbs.

While researching for robotics controllers a line of robotic components called Phidgets was found[9]. Phidgets makes a variety of robotic components that operate over a USB interface, perfect for a robot with a laptop at its core. Phidgets also had a number of similar robotics project listed on their website, clearly suggesting this robot could also be built using these parts. The Phidget 4-Servo Motor Controller is capable of sending 4 PWM signals to motor speed controllers. This can control up to 4 motors through a single USB port.

The Phidget InterfaceKit 8/8/8 is an interface board that allows up 8 analogue input, 8 digital input, and 8 digital output devices to be controlled from a single USB port. Attached to this interface are the following:

[8] http://www.vexlabs.com/
[9] http://www.phidgetsusa.com

One IR Reflective Sensor mounted on the bottom of the robot to see reflective lines indicating checkpoints, and three IR Distance Sensors are mounted on the front and sides of the robot for collision detection.

3.2 The Robot Software

Unfortunately, the only programming language directly supported by Phidgets was Visual Basic 6, and it was somewhat outdated. However, a group at the University of Calgary has developed a Phidgets interface for the .NET framework which makes controlling the components from the C# language possible. Therefore, C# was selected as the language for the entire project.

Navigation System

Because the robot was designed to save time on routine building checks, it needed some sort of a navigation system that allows it to navigate preset routes autonomously. A check point system was deemed the simplest solution. All corners, turning points, and edges in the potential route would be marked with reflective tape. The robot would be equipped with an infrared sensor to detect the tape. It would then just continue in a straight line until it detected a reflective tape, indicating a checkpoint. Then it would read from an internally stored data tree to see which way to turn to get to the next check point, turn in that direction, and continue in a straight line again.

Robot Vision

This security robot also needed an artificial intelligence that would allow it to recognize the world around it and be able detect intruders and alert the human operator. A neural network is a highly flexable data-processing structure made up of a number of nodes arranged in layers. The nodes of one layer may or may not trigger the nodes of the layer below it and so on. The relationship between each node is weighted and these weights can be adjusted automatically by training the neural network on a set of test data. The camera would take in a continuous stream of images of what is in front of the robot, and periodically (typically every 0.3 seconds) select one of the frames from the video stream and feed it to neural network. The neural network should be able recognize when a human is in the picture, stop the robot, and then alert the guards. In order to be effective, neural networks first have to be trained. To build the training set for this project, over 300 pictures were taken of a particular hallway. 200 of these pictures had a human in them and 100 did not.

4. Implementation

4.1 Physical Construction

4.1.1 Structure

The structural components of the robot are made up mainly of pieces from an off-the-shelf Vex robotics kit. These structural pieces are designed to fit together in any configuration, similar in concept to an Erector Set. The pieces are held together with hex bolts which are also part of the kit. On top of this frame rides the laptop which provides all the processing power of the robot.

Movement is provided by four Vex Omni-Directional wheels. Each wheel is directly driven by a Vex Continuous Spin Motor which is attached to the frame.

The three Phidgets Distance sensors are mounted directly to the frame with 6-32 machine screws and nuts. The forward sensor is mounted in the center of the front frame rail. The left and right sensors are mounted on their respective side rails just behind the front wheels. The reflective sensor is mounted on a slotted angle bracket facing downward. The slot on this bracket allows the height of the sensor to be easily changed to account for varying floor conditions.

All other components such as motor, power, and feedback controls are mounted to one of two 6.5 x 8” perforated circuit boards which are held to the frame using strips of Velcro for easy removal. The forward board holds the USB hub, attached with Velcro, and the Phidgets 8/8/8 Interface, held with 4 6-32 machine screws. The rear board carries

the rechargeable battery pack, also attached with Velcro, as well as the Phidgets 4-Servo Controller and 3 Phidgets Distance Sensor Interfaces all held with 6-32 machine screws.

For detailed diagrams and pictures of the physical construction of the robot refer to Appendix A.

4.1.2 Custom Components

Since the robot uses a number of different components that were not designed to work together, it was necessary to construct some custom parts to adapt from one system to another.

The rechargeable battery pack and its charger both used a square Molex style connector. However, the Phidgets servo controller is powered through a size N round connector. This incompatibility required a male square to female round converter. Because the voltage of the battery was already within the acceptable range of the motor controller, the converter simply had to allow a connection between the two. The converter was built by using 18-guage wire to directly connect the positive pin of a female square connector to the center terminal of a male size N round connector, and the negative pin of the square connector to the outer terminal of the round connector.

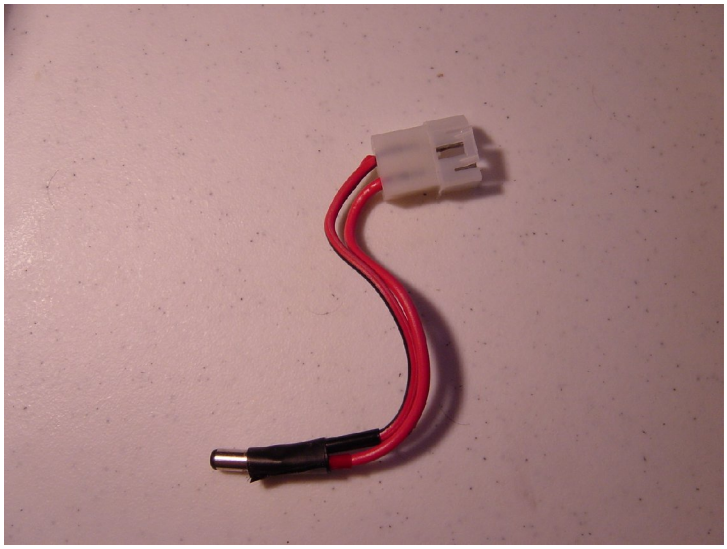


Fig 4.1.2-1 Battery to motor controller adaptor

The Phidgets 4-Servo Controller only uses input power to drive the motors connected to ports 1-3; port 0 is driven by USB power. This means that if all four motors were connected independently, one would receive less power than the rest. This, however, would only be a problem if the Vex motors were compatible with the Phidgets controller. But this is not the case since both the motors and the controller use a male configuration for their 3-pin PWM connectors. Both of the power and the incompatibility problems were solved by the construction of two female-female Y-splitters. The 3 female connectors of the Y-splitter allowed the two male components to be connected and the Y configuration allowed two motors to be powered from the same port eliminating any discrepancy in power. The Y-splitter also had the added bonus of simplifying the software control of robot movement since now it was only necessary to change one value instead of two to alter the speed of either side.

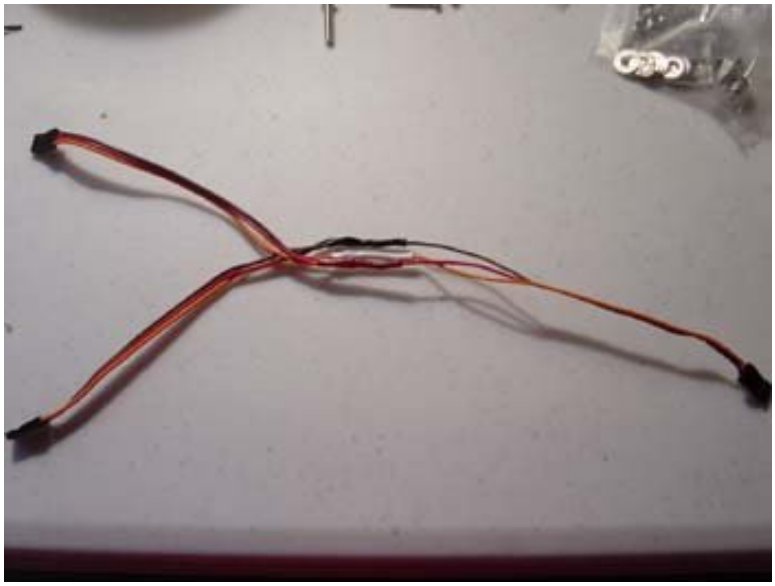


Fig 4.1.2 -2: Female-female PWM Y-splitter

4.1.3 Wiring

In order for the software to correctly interface with the hardware, each of the components must be connected in a particular manner. Battery must be connected to the power input of the motor controller. The front and rear motors on the left side must be connected via a Y-splitter to port 1 of the motor controller. The right side motors must be connected to port 2 in the same way (Fig A.6).

The left side distance sensor must be connected to an IR Sensor Interface which must be connected to port 0 on the 8/8/8 sensor interface. The forward distance sensor must also be connected to an interface which must be connected to port 1, and the right side sensor's interface must be connected to port 2. The reflective sensor can be connected directly to port 4 of the interface. Both the motor controller and sensor

interface have to be connected via USB to the onboard hub which must then be connected to the laptop (Fig A.7).

4.2 Software Design

4.2.1 Hardware Interface

Hardware control and sensor feedback is accomplished using the GroupLab.Phidgets.NET package available under academic license at http://grouplab.cpsc.ucalgary.ca/software/phidgets/. This package is a C# wrapper around the standard VB6 Phidgets library. Using calls to this library it is possible to change the speed of motors as well as get the values of any sensor attached to either the Phidgets servo controller or the Phidgets Sensor interface.

The class Robot provides a wrapper for the relevant GroupLab.Phidgets.NET calls. It provides a layer of abstraction for Robot commands such as “turn left”, “set forward speed”, or “get sensor value”.

4.2.2 Server

The server runs continuously on the robot. The server is responsible for sending feedback and video to the client as well as executing commands it receives from the client. The server is controlled by the Server class which itself is basically a container for several other objects:

An ImageServer object, responsible for sending video to the client.

A FeedbackServer object, responsible for sending sensor data to the client.

A CommandServer object, which executes commands send from the client.

A WebcamInterface object, which provides relevant webcam calls.

And A NeuralNetworkInterface, which loads and runs a neural network on input from the camera.

4.2.3 Client

The remote client allows a user to connect to and control the robot from any internet connected computer as well as view video and sensor feedback. A design decision was made to give the client as little responsibility as possible; therefore, its essential functions only include making a connection with the server, sending requests at intervals for sensor values and video frames, and transmitting commands to the server.

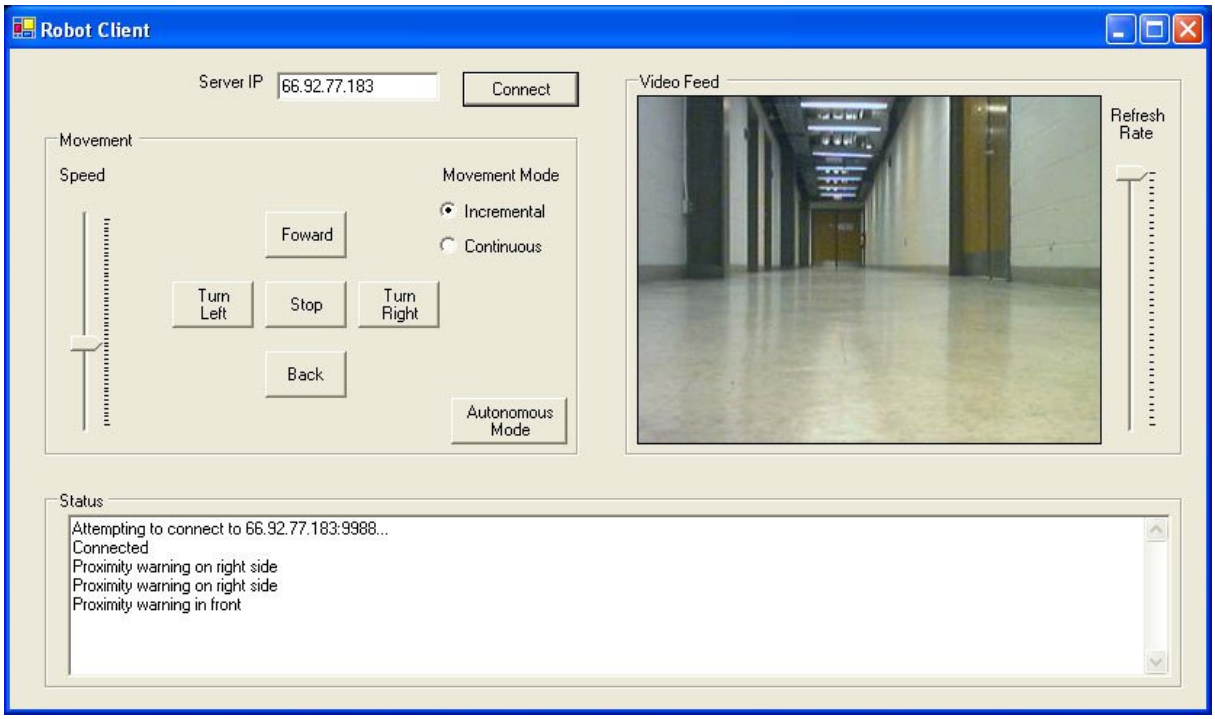


Fig 4.2.3-1: Screenshot of the client

4.2.4 Navigation

Autonomous navigation is guided by a checkpoint system. These checkpoints are represented by strips of reflective tape across the robot's set path. When the robot detects

one of these strips it executes a set of commands defined in an external file, then continues to drive straight.

Autonomous navigation is controlled by the AutonoDriver class which contains a set of rules which govern the behavior of the robot based on the values of its sensors and its current status. When an AutonoDriver object detects a reflective strip, it temporarily turns control over to the Script class. A Script object contains a set of commands, each one stored in a ScriptCommand object, to be executed at each checkpoint. It executes the commands corresponding to the current checkpoint which are defined as a block in the script, and then returns control of movement to the AutonoDriver.

As stated above the navigation instructions are read in from an external file. This file, "default.dat", is located in the program's working directory. The script language is based on a small set of simple commands with one or two arguments each.

Command	Argument 1	Argument 2	Description
REP	Number of repetitions (0 for infinite)	NA	Start a looped block
ENDREP	NA	NA	Close a loop
STARTBLOCK	Name	NA	Start a command block
ENDBLOCK	NA	NA	End a command block
STRAIGHT	Duration	Speed	Move forward
LEFT	Duration	Speed	Turn left
RIGHT	Duration	Speed	Turn right

Fig 4.3.4-1: Table of script commands

Example:

If the desired path is start down a corridor, turn left at the first corner, proceed down that corridor to the end, turn back, drive to the corner turn right, and go back down the original corridor:

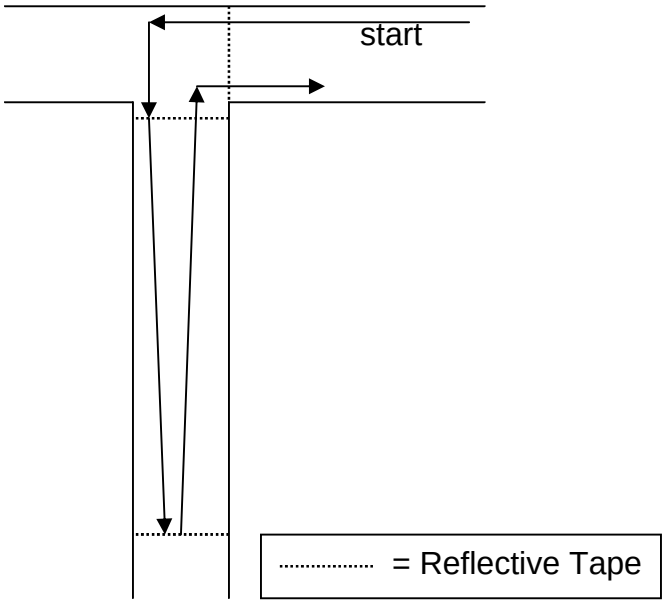


Fig 4.3.4-2: Diagram of example script

The script to accomplish this would vary with hall size, floor conditions, or desired speed but it would be similar to this:

```
REP,0
STARTBLOCK,BLOCK1
STRAIGHT,10,30
LEFT,3,20
ENDBLOCK
STARTBLOCK,BLOCK2
STRAIGHT,4,30
ENDBLOCK
STARTBLOCK,BLOCK3
LEFT,4,30
ENDBLOCK
STARTBLOCK,BLOCK4
STRAIGHT,10,30
RIGHT,3,20
ENDBLOCK
ENDREP
```

“REP,0” means this script will continue executing in an infinite loop.

“STARTBLOCK,BLOCK1” defines the start of the commands to execute when the first

line is detected, called BLOCK1. It contains the instructions to first go straight to line up with the hall, then to turn left to orient the robot in the right direction. "ENDBLOCK" defines the end of this block. BLOCK2 just contains a "STRAIGHT" command, essentially ignoring the line. And so on until the "ENDREP" which defines the end of the loop.

4.3 Neural Network

A neural network is a form of artificial intelligence which is inspired by the way a biological brain works. Like a human brain, it consists of a large amount of interconnected data structures. These structures, called neurons, work in a unique way to provide a solution to a given problem. A neural network must be preconfigured through a learning process to a specific problem before it could provide any useable solution. Learning in a biological system is accomplished by adjusting the synaptic connections that exist between cells. This is also true for a neural network. The neural network in this project was used to analyze the video stream taken from the onboard camera to look for anomalies, such as intruders.

The advantage of using a neural network in this project is that it has an ability to derive meaning from complex or imprecise data, such as a picture taken from a moving robot. A properly trained neural network should be able to predict the outcome of a given data set similar to those on which it was trained. However, again like a brain, a neural network is a black box, that is, it is nearly impossible to determine exactly *how* the result was reached.

A neural network is constructed with layers of neurons. A neuron is a data structure that has at least one input, at least one output, and a firing rule. A neuron has two modes of operation; training mode and running mode. In training mode, the neuron is adjusted to fire when a certain condition, such as a threshold, is reached. In running mode, data is fed into the inputs of a neuron and, based on the firing rule defined by training; it may or may not fire resulting in the triggering or non-triggering of its outputs. Each of the outputs of that neuron is connected to one or more of the inputs of a neuron in the next layer. Each of these connections has a weight which is also determined during training. This second neuron determines whether to fire based on the weighted value from each of its inputs and its firing rule. When it fires the process repeats and so on through each neuron and layer of the whole network.

The firing rule is an important concept in neural networks and accounts for their high degree of flexibility. A firing rule determines on what set of input data a particular neuron fires. To implement a firing rule for specific neuron, first input a set of data into the neuron. Some of this data will cause the neuron to fire (the "yes" set) and some will not (the "no" set). When a pattern which is not in either set is entered into the inputs of the neuron, the neuron will fire if this new data has more inputs in common with the "yes" set and will not fire if the new data has more inputs in common with the "no" set.

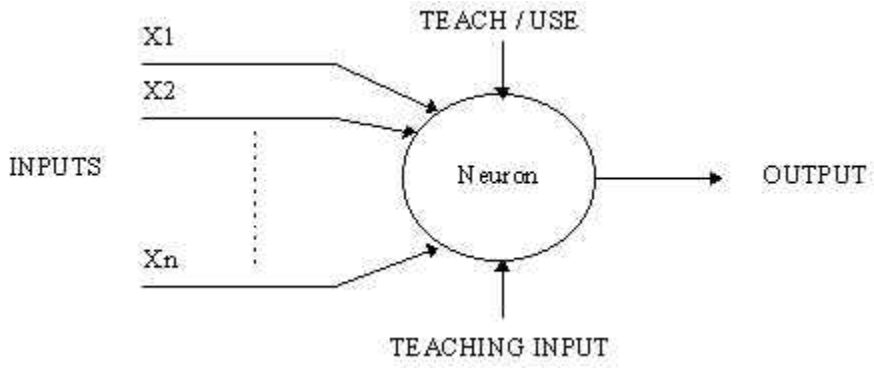


Fig 4.3-1: Structure of a neuron

The neural network in this project consists of 3 layers. The input layer is where the raw image data is fed into fed into the inputs of the first layer of neurons in the network. The behavior of the second or "hidden" layer is determined by the firing of the input neurons and the weights on the connections between the input and the hidden neurons. The output layer neurons will take their input from the output of the hidden layer. The output from this layer is the final output from the network.

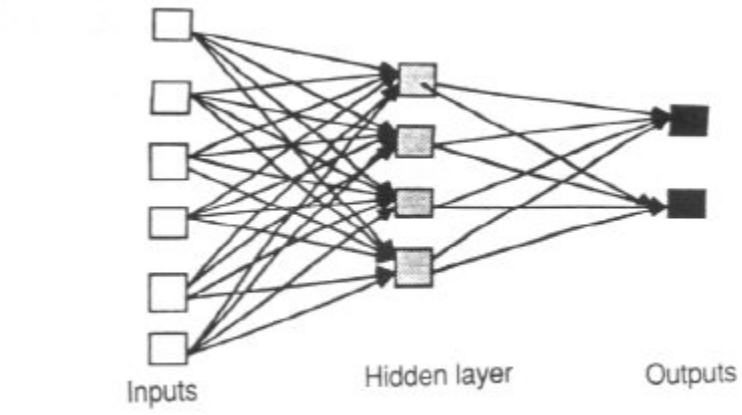


Fig 4.3-2: Relationship between three layers of a neural network

4.3.1 The Learning Algorithm

As mentioned above, training is an essential part of building a neural network. The weight of each connection between two neurons must be fine tuned to minimize difference between the expected and actual output. To do this, the neural network must calculate how the error changes as each weight is increased or decreased this is known as the error derivative of the weight (EW).

A back-propagation algorithm was chosen because all the neurons in the network are linear. The algorithm computes each EW by first computing the EA, the rate at which the error changes as the activity level of a unit is changed. For output units, the EA is simply the difference between the actual and the desired output. To compute the EA for a hidden unit in the layer just before the output layer, we first identify all the weights between that hidden unit and the output units to which it is connected. Then we multiply those weights by the EAs of those output units and add the products. This sum should equal the EA for the chosen hidden unit. After calculating all the EAs in the hidden layer just before the output layer, we can compute EAs for other layers, moving from layer to layer in a direction opposite to the way activities propagate through the network. Once the EA has been computed for a unit, it is straight forward to compute the EW for each incoming connection of the unit. The EW is the product of the EA and the activity through the incoming connection [10].

[10] http://www.doc.ic.ac.uk/~nd/surprise_96/journal/vol4/cs11/report.html

4.2.3 Designing the Neural Network

There was some difficulty in designing a network for this application. Initially the network followed a design used for a self-driving car called "Shakey", a project from 1960 at Stanford University [11]. This prototype consisted of three layers, one node for each pixel of the image in the input layer and four nodes in the hidden layer. Since it worked well for the driving experiment, it was assumed that it would work for this project. However, the single frame accuracy for human detection was close to 50%. The low number of nodes in hidden layer was potentially responsible for this low accuracy. With so few nodes in this layer it is possible that a large amount of information was being lost. Therefore, it was decided to try a number of different designs to see which had the best results.

The neural network library used was designed by Franck Fleurey, and expanded for this project [12]. Since it takes a substantial time to train each network increasing with the resolution of the input image, the experiment was done using low resolution pictures. The goal of the experiment was to find the optimal number of hidden layer nodes to produce the most accurate results.

[11] http://www.sri.com/about/timeline/shakey.html

[12] http://franck.fleurey.free.fr/NeuralNetwork/

5. Results

5.1 Physical Results

Tests were conducted to determine the physical performance capabilities of the robot itself.

Top Speed: .8 feet/second

Average Speed: .4-.5 feet/second

Battery life: 2.5 hours (continuous driving)

Maximum Range: 2700 feet

5.2 Neural Network Results

Experiments were conducted to determine the optimal design for the human-detection. These experiments tested several different neural network configurations as well as different input resolutions for their accuracy in determining when a grey-scale video frame was of an empty hallway or of a hallway with a person in it.

Experiment 1:

300 pictures of training data with resolution 15x20.

150 empty hall way.

150 occupied hall way.

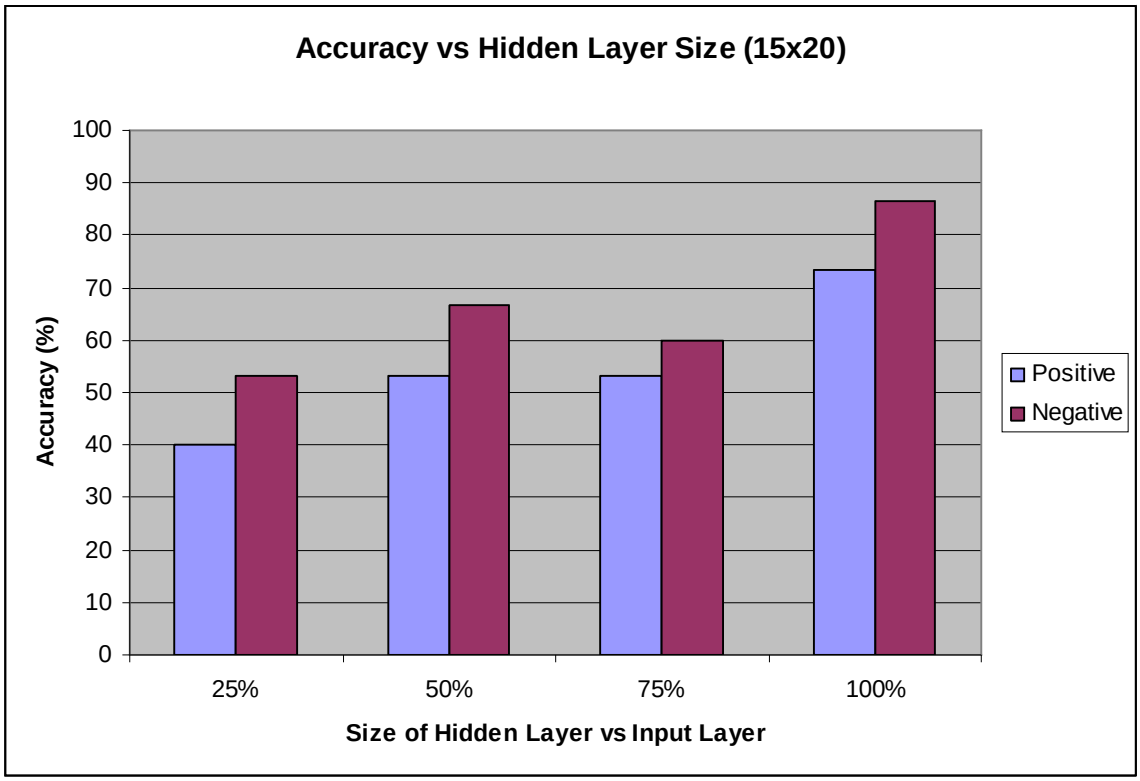
50 pictures of testing data with resolution 15x20.

25 empty hall way.

25 occupied hall way.

Percent accuracy (number of times the network output matched the expected output) for both positive (person) and negative (empty) pictures versus hidden layer size as a percent of the input layer size:

	25%	50%	75%	100%
Positive	40%	53.33%	53.33%	73.33%
Negative	53.33%	66.66%	60%	86.66%



Experiment 2:

300 pictures of training data with resolution 45 by 60.

150 empty hall way.

150 occupied hall way.

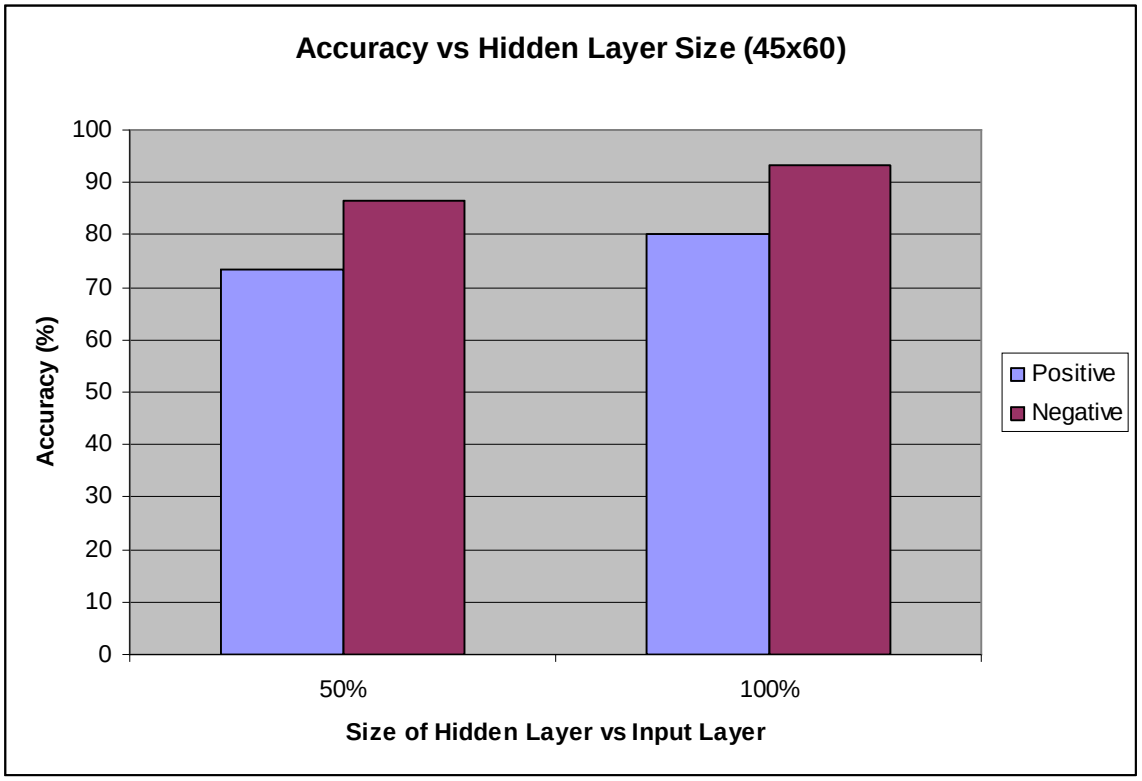
50 pictures of testing data with resolution 40 by 65.

25 empty hall way.

25 occupied hall way.

Percent accuracy (number of times the network output matched the expected output) for both positive (person) and negative (empty) pictures versus hidden layer size as a percent of the input layer size:

	50%	100%
Positive	73.33%	80%
Negative	86.66%	93.33%



6. Conclusions

Based on the results of the experiment, it was determined that accuracy was proportional to both picture resolution and hidden layer size. Therefore, the highest accuracy network tested was the 45x60 resolution with a hidden layer 100% of the size of the input layer and this was the one finally used on the robot. It could be inferred that as image resolution and hidden layer size increase, the network would get more accurate. However, training time then becomes an issue. Training a single 240x320 image on an average computer took more than a day.

This network, when finally tested on the robot itself, was found to have an accuracy of detecting humans in the high 80 to mid 90%s when the human was within 15-20 feet of the robot. However, the network also had a very high rate of false positives. In order to filter out the majority of false positives, a simple counter was added to the NeuralNetInterface class. This counter keeps track of how many positive results in a row come out of the neural network. If the result is positive on five consecutive frames, then the alert is triggered, the robot is stopped, and the user is notified. Implementing this filter reduced the number of false positives to a more manageable level of around 90% in a totally empty hall.

7. Future Work

Despite the success of the project, there is still a lot of room for improvement most notably in several key areas. First and most importantly, increasing in the detection accuracy of the neural network would greatly improve the viability of the system. Currently, the neural network can detect when a person is within 15-20 feet of the robot with a reasonable degree of accuracy. However, even with the false positive buffer explained in the previous section, false detections are still common. Using a higher resolution picture as input, increasing the size of the hidden layer, or adding more layers could all potentially increase the accuracy and range of the neural network. Barring an increase in the power of the neural network, other detection methods could also be used. Heat, noise, and movement are among the possible alternate methods of detecting intruders.

The current system of navigation is inefficient. The script is stored on the robot itself, and the user must have physical access to it in order to change the route. Enabling the client to change the navigation script would save time and work for the user. The use of reflective tape on the floor is neither aesthetic nor practical. Using a more error proof and less visible method of enabling the robot to know its location is preferable. Some such methods are wheel rotation encoding, inertial guidance, or the use of RF or other signals.

Currently, the robot server is capable of handling only one client connection at a time. Allowing multiple connections would mean that more than one person could monitor the same video stream simultaneously, reducing the chance of human error. Allowing multiple connections introduces the possibility of control conflicts. This might

be avoided by having a single master client, with movement control, and multiple slave clients, with just video feed.

The forward facing camera offers only a minimal field of view to the user and the neural network. A panoramic or wide-angle camera would give the neural network a much better chance of spotting an intruder. Alternately, a pan/tilt/zoom camera with controls integrated into the client would give the user far more information about his environment as opposed to the current stationary camera.

8. Appendices

Appendix A: Structural Diagrams

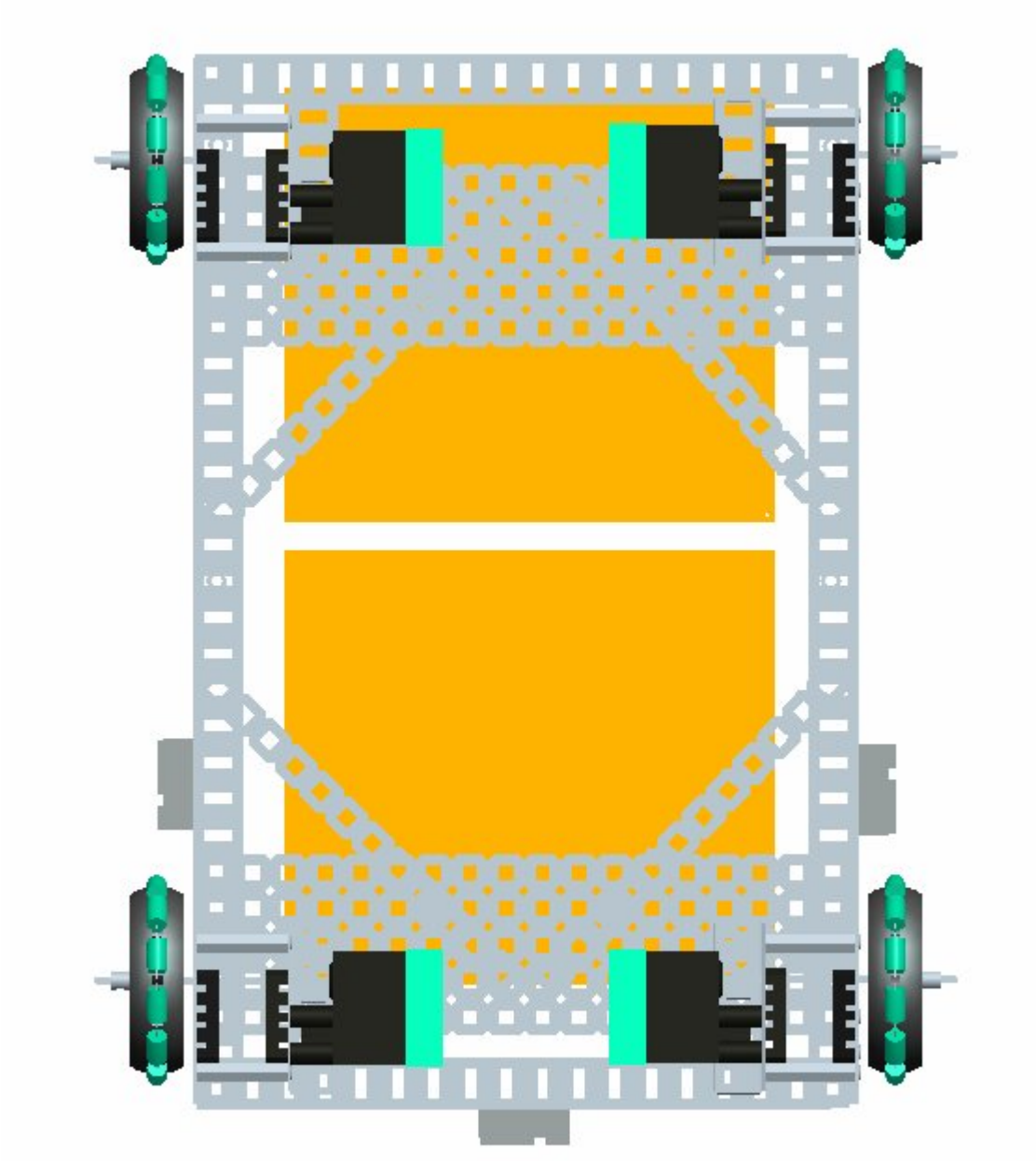


Fig A-1: Bottom view

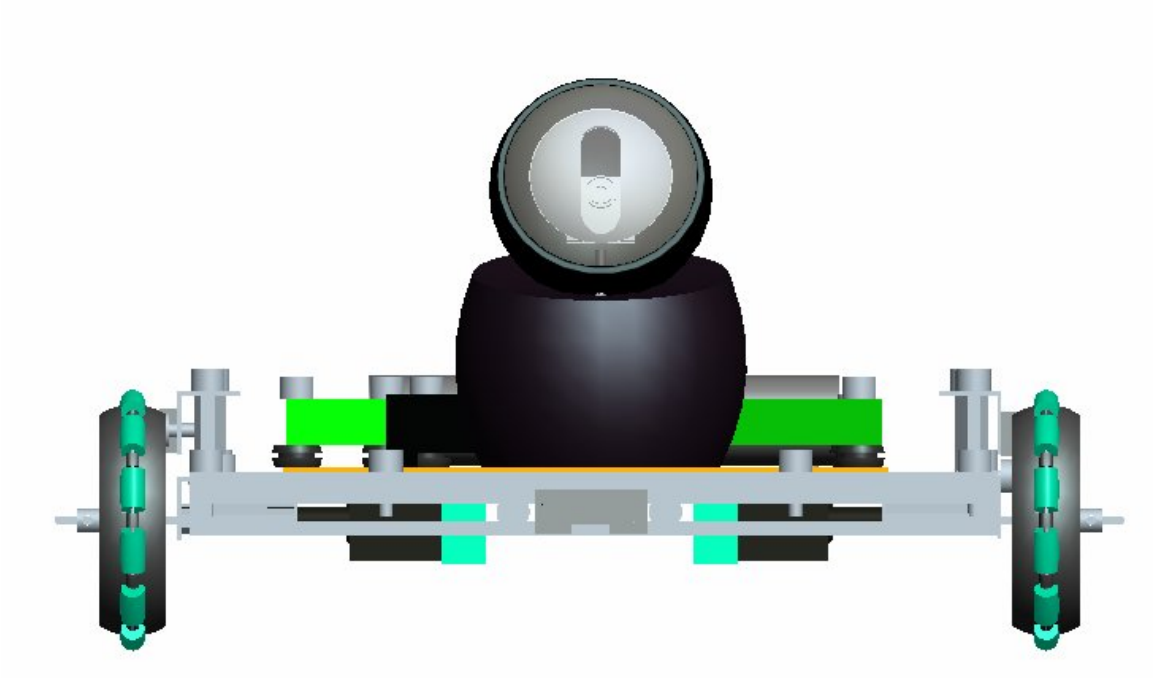


Fig A-2: Front view

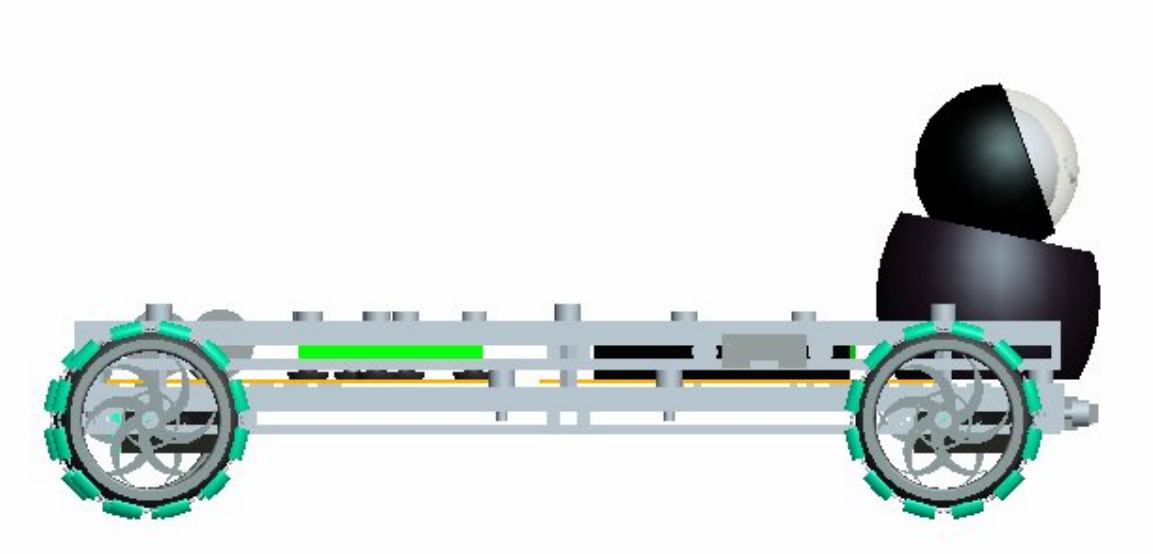


Fig A-3: Right-side view

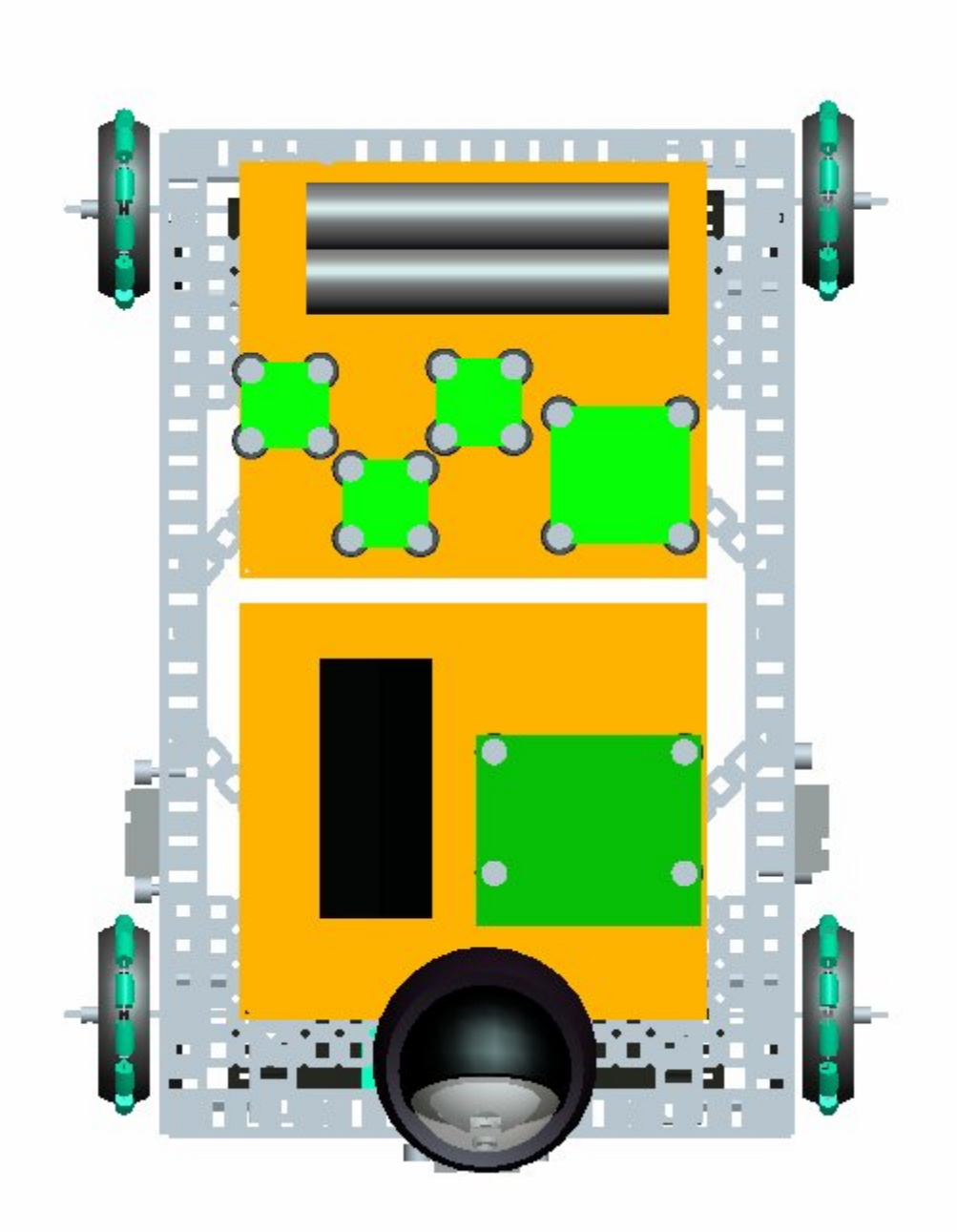


Fig A-4: Top view

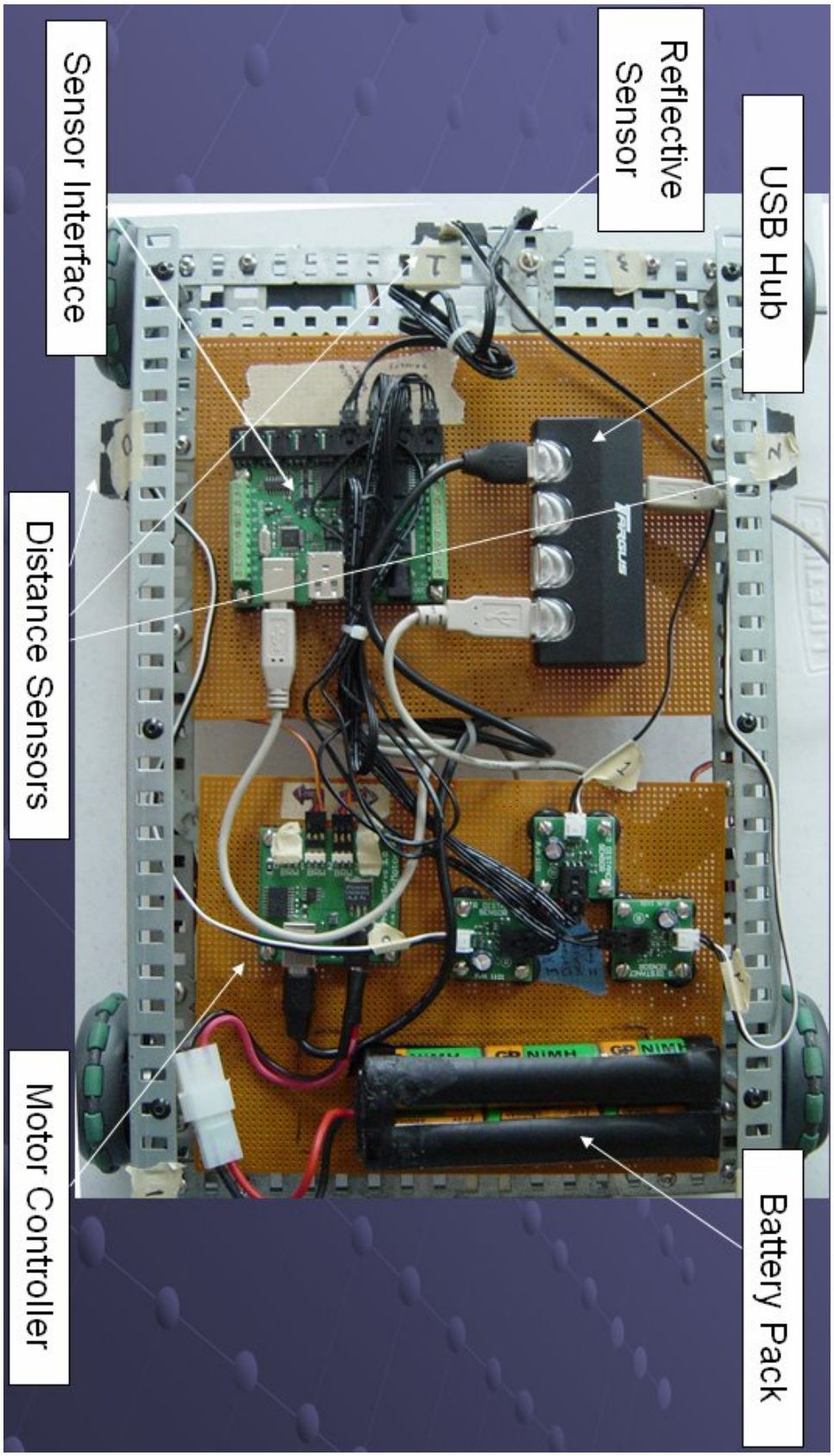


Fig A-5: Top view with components labeled

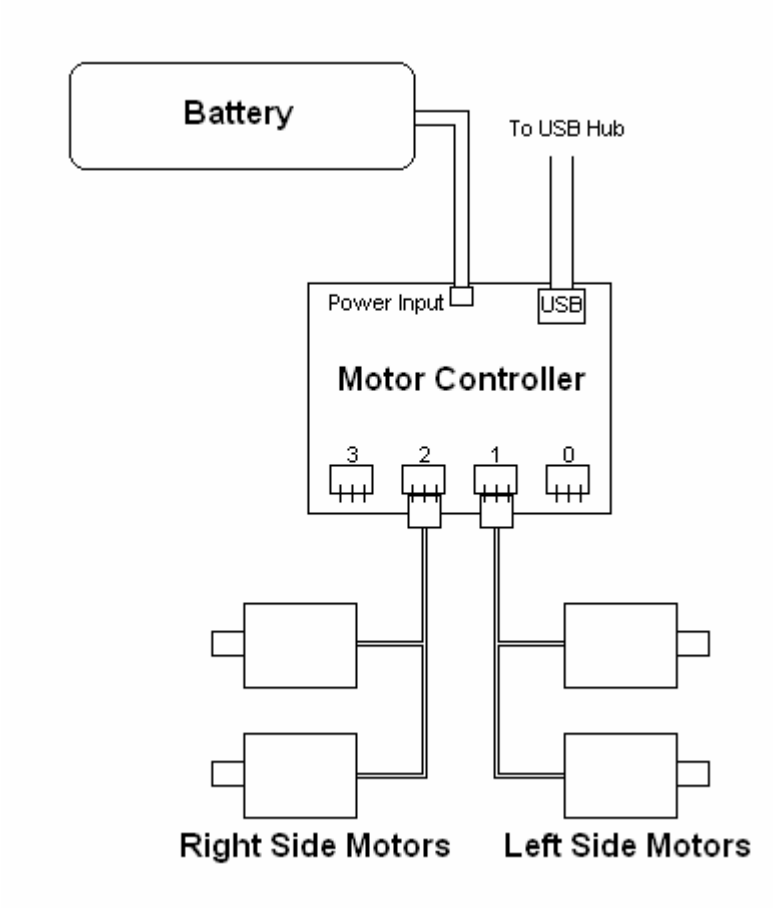


Fig A-6: Wiring diagram of motor system

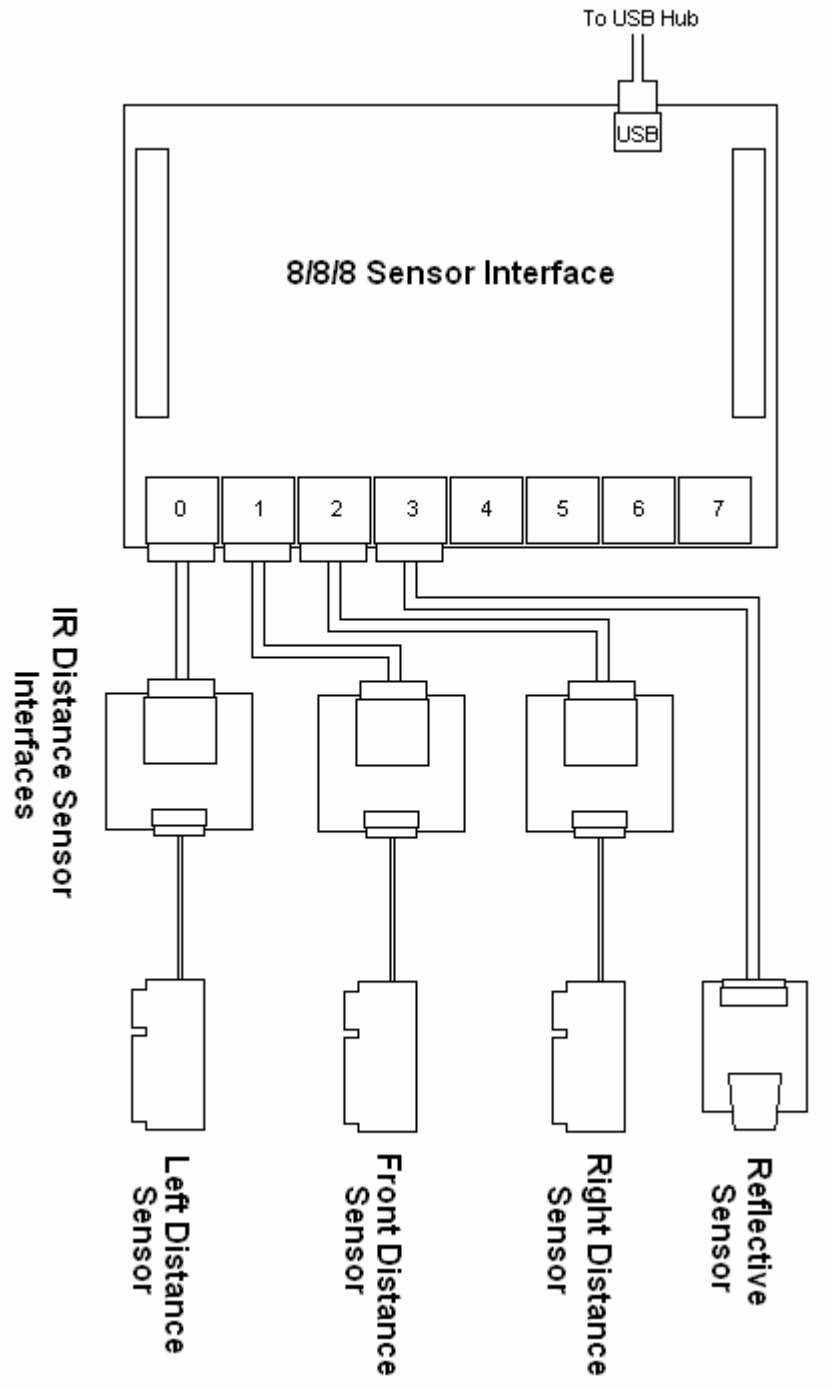


Fig: A-7: Sensor system wiring diagram

Appendix B: Software Diagrams

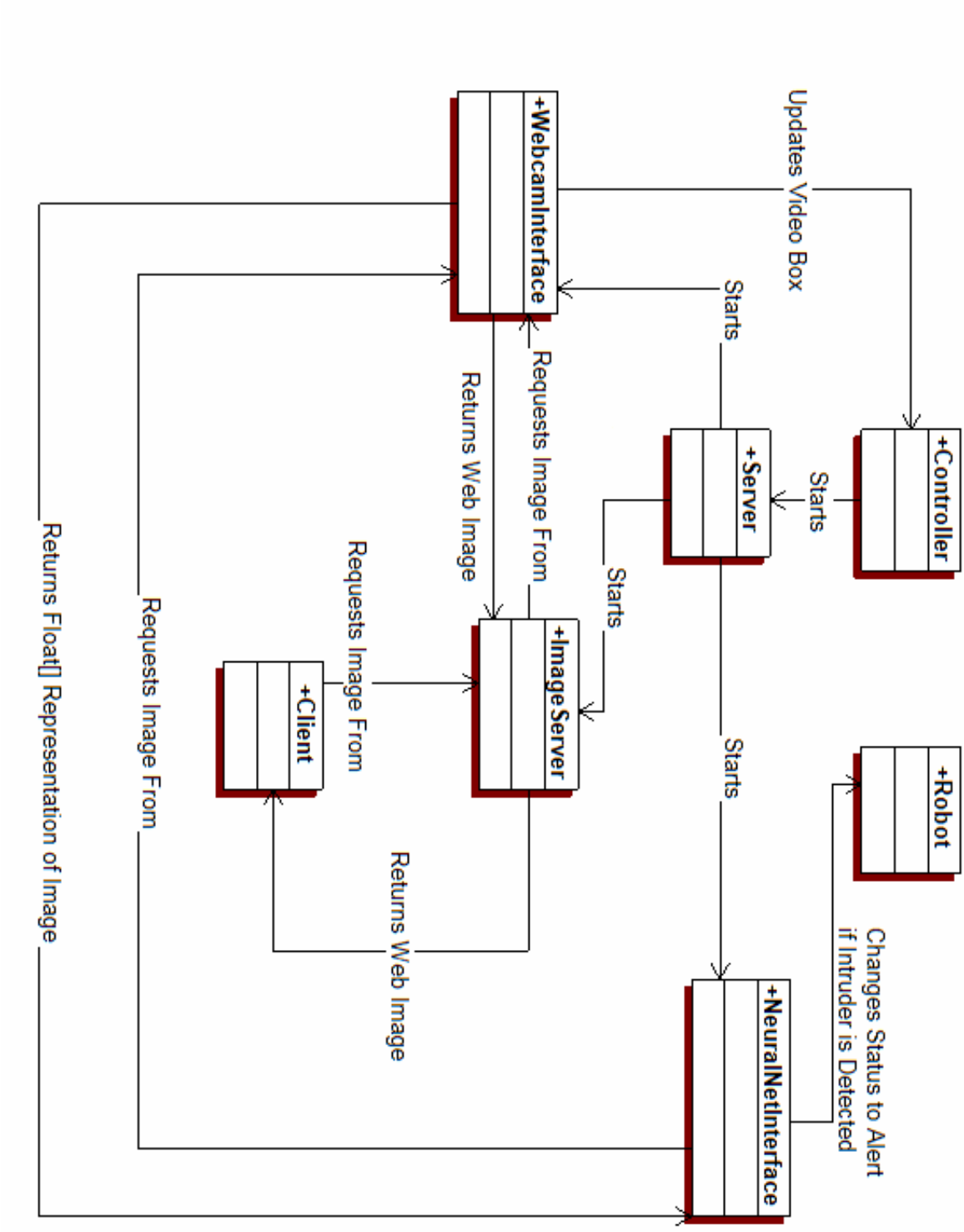


Fig B-1: Image flow diagram

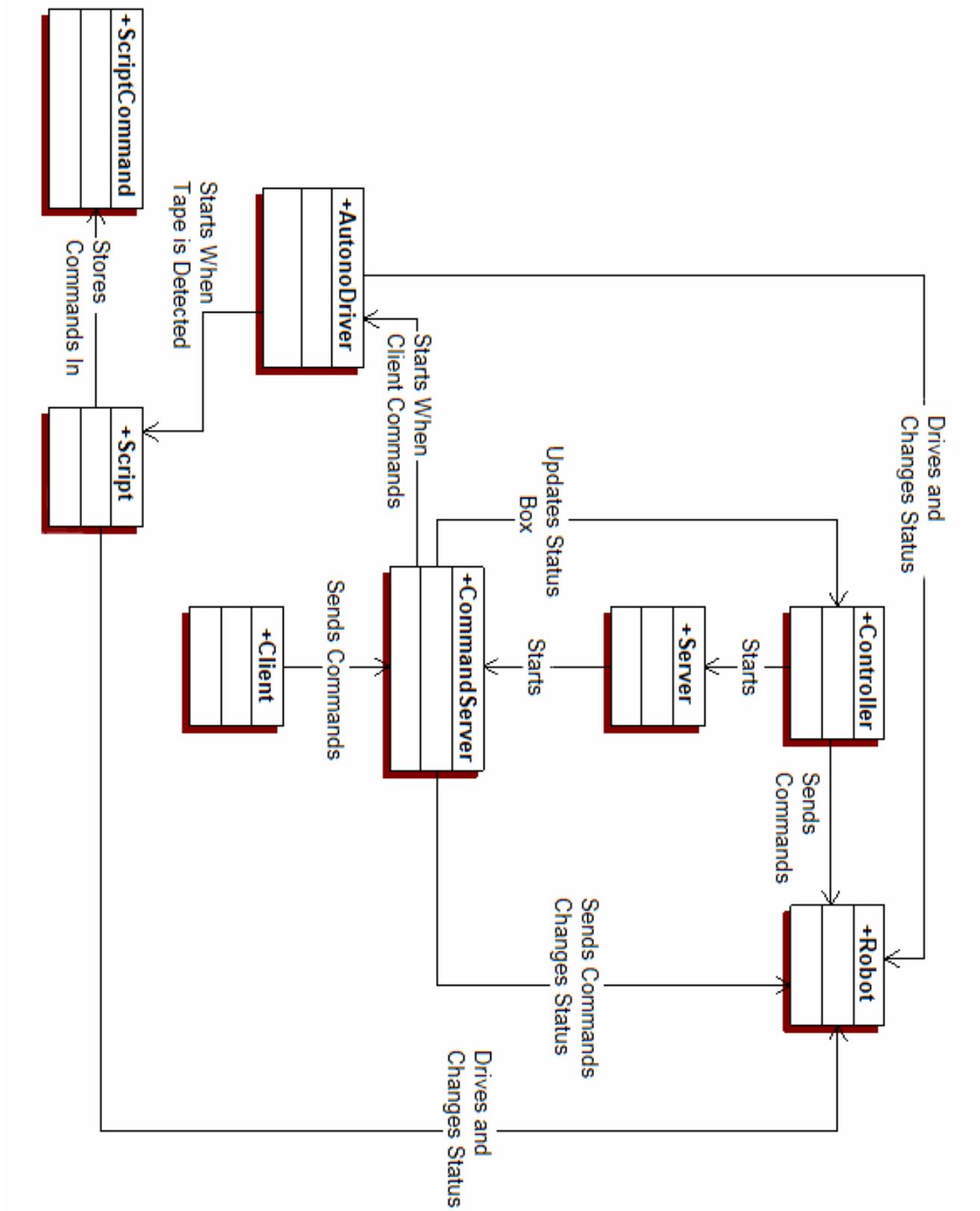


Fig B-2: Command flow diagram

Appendix C: Parts List

Phidgets (http://www.phidgetsusa.com)

(1) 8/8/8 Sensor Interface kit	-	$85.00
(1) 4 Servo Controller	-	$63.75
(3) IR Distance Sensor Kits	-	$5.95
(1) Reflective Sensor 5mm	-	$33.05

Vex (http://www.vexlabs.com/)

Various structural pieces	-	$30.00
(4) Continuous Spin Servos	-	$80.00
(4) Omni Directional Wheels	-	$40.00

Other

(1) USB hub	-	$20.00
(1) 7.2v 3300mAh NiMH battery pack	-	$25.00

Appendix D: Included Files

RobotClient – Project files for the remote client. Build using VisualStudio.net.

RobotServer – Project files for the robot server. Build using VisualStudio.net.

Neural Network Trianer – Training program for the neural network.

Grouplab_Phidgets_Net – Installer and installation instructions of the Grouplab.Phidgets.Net package

CAD Files – Pro-Engineer models of the robot and its components.

Photos – Photos of the robot.