Cooperative Light Identification and Mobilization by a Robotic Team

Submitted By
Emily Mower

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR AN UNDERGRADUATE THESIS WITH A

BACHELOR OF SCIENCE IN ELECTRICAL ENGINEERING

School of Engineering
Tufts University
Medford, Massachusetts

May 2004

Signature of Author:
Emily Mower

Certified By:
Dr. Ron Lasser
Dept of Electrical and Computer Engineering
Tufts University

Committee:
Professor Chris Rogers
Department of Mechanical Engineering
Tufts University

Committee:
Professor Joseph Noonan
Dept. of Electrical and Computer Engineering
Tufts University

Abstract:

Governments are interested in developing systems that protect their populace by safeguarding against potential terrorist attacks. Due to their mobility, chemical and biological weapons present a danger to the public because they can be easily released into enclosed spaces. To combat this threat, a group of researchers at Tufts University developed a nose sensor capable of detecting the source of a chemical leak. A system of robots that would use this chemical sensor was prototyped using a light source as the target threat and light sensors as the nose. This cooperative robotic team is composed of three robots: two stationary robots and one mobile robot. The two stationary robots form a coordinate system and sweep through ninety-degrees, locating the area of greatest light intensity. The mobile robot uses this data to calculate the position of the light spot. It then travels towards the target and tracks it as it moves. The robotic team is able to locate and track the spot of light. The team cannot arrive at the exact location of the spot due to motor inaccuracies. This solution is important because it presents a method to protect public areas.

Table of Contents:

Introduction:

As terrorist organizations become increasingly technologically adept, governments are forced to consider and develop programs to protect the public from potentially devastating elements. New systems must be able to identify possible terrorist target areas and constantly update the relative safety level of those areas. In the event of an attack, the systems developed must be able to deploy a first-response unit capable of locating the source and containing the threat.

This continuous technological development also leads to increasingly hazardous potential threats. Human response teams should no longer be dispatched as first-response units. It is therefore necessary to create a robotic team capable of both locating a threat within an area of set boundaries and deploying a robotic mobile response team to the location specified. After arriving at the location of the threat, the mobile unit(s) of this team would report its location to the observer. The mobile unit(s) would then follow the threat as it moved to determine its final location.

A group of researchers at Tufts University led by Professor David Walt of the Chemistry Department [11] have designed a robotic nose capable of detecting a chemical trace in the air and then locating its source using the principles of fluid dynamics. Once the location of the spot has been identified, a team of mobile robots could be sent to the area of potential threat. The use of this team of robots is important because it reduces the chances that the members of a human or animal rescue team would suffer the effects of the chemical leak. In the absence of a robotic team, a human team would be forced to enter the contaminated region endangering every member. The design of this robotic team would enable a human rescue team to operate at a distance to contain or remove the threat. Once the location of the source of the spot is

determined, it is possible that the source could move from its initial position. It is therefore important that the robotic team be capable of tracking the source if and when it begins to move.

In addition to its humanitarian uses, the design of this robotic team also presents opportunities to another group of individuals. This team was designed with elementary outreach in mind. Young children do not often receive exposure to engineering related fields until well into their high-school education. As a result of this, many children, and especially girls, are not drawn to the field, as they have no understanding of what the field entails. It is important to provide an outlet to the community that will enable the children to come into contact with the basic underlying principles of engineering and the possible implications. This robotic team fulfills that niche. It was taken into the classroom and demonstrated to children. While the robot was running an instructor explained the basic concepts that allowed the robot to function. This background information allowed the students to experience engineering from a different standpoint than usually presented and they thoroughly enjoyed the hands-on experience.

Relationship to Previous Research:

The dangers of chemical leaks are present outside the realm of possible terrorist activities. Chemical leaks may also result from underground disasters such as broken pipelines. In these instances, the evidence of the leak would exist beneath the soil. In "Robotic Location of Underground Chemical Sources" [10] Russell presents a method for effectively determining the position of a chemical leak in sand. The algorithm he develops, Hex-Path is modeled after the enhanced motion of the Planarian worm. In this algorithm, the robot moves forward a specified distance and then turns in one of two directions (+60 degrees or -60 degrees) depending on the reading of its sensors for the previous two moves (n-2 and n-1). If the reading at n-2 is greater

than the reading at n-1, then the robot's most recent move was incorrect and the robot must turn towards the direction of the n-2 reading and vise versa. The benefit of this algorithm is that it provides the robot with a method to travel from the starting point to the goal point in a highly defined manner. This allows the robot to maintain positional awareness. However, this type of algorithm is unnecessary when stationary sensors are used. Unlike the algorithm described by Russell, the stationary sensors in this paper provide the robot with data on the position of the spot. This allows the mobile robot to determine a specific path to travel to the goal spot. This decreases the amount of power consumed by the mobile robot because the path it takes to the target is direct. Additionally, if the robot in Russell's algorithm accrues any positional error, the error will be magnified with each successive trial. If the target location is a great distance away, the final positional data may be meaningless. The algorithms in this paper provide a method for locating a point in space while minimizing the amount of power consumed and maximizing the positional accuracy of the final conveyed location.

Threat can also be modeled as an audio signal. In a paper by Macera et al. entitled, "Remote-Neocortex Control of Robotic Search and Threat Identification" [8] the authors developed an algorithm for tracking an audio signal and determining its level of threat potential. The target used by Macera constantly emitted an audio-speech signal. The robot then identified the position of this signal using binaural audio processing techniques through the use of its two spatially separated microphones. The data collected is passed to a remote computation site, which calculates the direction of the audio signal relative to the position of the robot. This process is repeated until the robot arrives at the correct position. Once the target is reached the robot passes the collected vocal and visual data to a higher-level processor that classifies the target as either threat or non-threat using speech recognition with visual clues. This method is

similar to the method discussed in this paper. However, the method discussed in this paper locates the spot using stationary robots. In the Macera algorithm, the robot relies on outside processors. This reliance decreases the robustness of the system. If the outside processors are rendered unreachable, the mobile unit would not be able to proceed in the direction of the target. Thus, the threat could not be contained. By relying on components within the system the stability of the system can be maintained.

Framling [4] presented a different navigational method using reinforced-learning (RL) techniques. This algorithm utilizes a method called ε-greedy exploration [12]. This method is used to balance exploitation and exploration in robotic movement. Maximum exploration is accomplished using a random search while maximal exploitation is accomplished using a "greedy" policy in which the action with the highest action value is chosen. The action value is defined as the value of benefit to the system achieved by a robot taking an action in its given state. In an ε-greedy algorithm, the greedy choice is taken with probability (1-ε) while a random action is taken with probability ε. This ε-greedy algorithm was then modified by including a learning rate of 0.0001. If after a move, the robot's light sensor reading increases, the weighting of the action selected increases. The difficulty with this modification is that it emphasizes a few selective moves until the robot had tested most of the moves. This inherent delay increases the light finding run time. The algorithm discussed in this paper uses a similar light sensor direction algorithm, however, the mobile robot always moves towards the sensor with the greatest light intensity. The benefit of this direct approach is that it allows the robot to move towards the light as quickly as possible, which is important when the robot is tracking a potentially harmful agent.

Robotic navigation algorithms are often developed to enable a robot to negotiate an unfamiliar environment. The current NASA Mars rovers best exemplify this problem. These

robots were programmed to survive in and investigate a foreign environment. However, before the Spirit and Opportunity rovers, the Sojourner rover explored Mars's surface. Sojourner was programmed with no prior knowledge of the Martian environment and consequently relied solely on real-time environmental stimulants for motion planning data.

In addition to its lack of environmental knowledge, Sojourner was also bound by "severe constraints of power, computational capacity, and the high cost of flight components, which translates into limited memory available on-board the rover [7]." As a result of the power constraint, the rover was forced to utilize its onboard sensors to limit its motion as much as possible. Laubach and Burdick [7] developed an algorithm to optimize the sensing array by sensing only the data needed for motion planning. They defined three types of sensor-based motion plans, "Classical", Heuristic, and sensor-based motion planners. Only sensor-based motion planners will be discussed here due to applicability. A sensor-based motion planner, "relies solely upon the rover's sensors and yet guarantees completeness [7]." Using this motion planner, Laubach and Burdick developed the Wedgebug algorithm. Wedgebug is based on two interacted modes that ensure global convergence. These two modes are "motion-to-goal (MtG)" and "boundary following (BF)." Only the MtG algorithm will be discussed here as the robotic team design discussed in this paper did not include obstructions of any kind.

The ultimate function of the MtG algorithm is to move the robot to a certain goal point using the shortest possible path, F. In the case where there are no obstacles, the robot travels along the specified shortest-path, F. Traveling along the shortest path is desirable for this rover because it ensures minimum power consumption. The motion algorithm used in this paper utilizes a different structure of motion to ensure accuracy of position. Horizontal and vertical components of the desired path are computed to eliminate the inaccuracy inherent in a variable

turning angle. Due to the lack of fine motor control it is difficult to create an accurate turning protocol. Therefore, by limiting the turning angle to only one set angle, it is possible to ensure accuracy in turns of that angle-measure. Compartmentalizing the path into two distinct path segments allows the robot to guarantee the path accuracy.

Buschmann, Muller, and Fischer further discussed this style of grid motion in a paper entitled, "Grid-Based Navigation for Autonomous, Mobile Robots" [3]. The environment specified by Buschmann et al. requires that the shape of the region of operation be a full rectangular grid in which all tiles are the same size. However, it is not necessary that the robots know the exact dimensions of the grid to be used. The robot is allowed to move in eight directions: south, north, south-east, north-east, etc. and has, "two ground-observing light sensors attached in a line orthogonal to the driving direction at the bottom of the robot, and two individually controllable wheels [3]." This robot also possesses a small roller ball that allows it to turn at its position without requiring any forward or backwards motion. The algorithm created follows five steps:

"(1) calculate the next tile to go to

(2) turn towards the direction of the next tile

(3) start driving and wait for line-crossing events to happen

(4) thereby decide which tile the robot arrived in

(5) finally determine whether the robot has reached the destination tile"

The algorithm terminates when the goal tile is determined to be the next tile. This algorithm differs from the motion algorithm used in this paper because it increases the complexity of motion needed. Due to the limitations of the microprocessor used in this project the number of inputs is limited. As a result, it is not possible to include extra sensors on the ground to mark

location. Additionally, if this algorithm is to be extended to real-world problems it would not be feasible to mark the ground of an entire area in a black checkerboard pattern, as there is a likelihood that a portion would be covered up, hindering the positional awareness of the robot. The motion of the robot used in this paper is calibrated to accurately reflect the strength of the battery where one motor pulse is equated to one inch of forward (or backwards) motion. The robot is then able to follow a grid pattern of motion by pulsing forward and at a ninety-degree angle as necessary.

The motion used in this paper is interesting because the mobile robots path is specified by the sensory input and resulting output of stationary robots. Batalin, Suckhatme, and Hattig discussed a similar navigation topic in "Mobile Robot Navigation using a Sensor Network" [2]. They discussed a global navigation problem in which the individual robot could not observe the goal state from its initial position. They therefore created a network of mobile robots and static sensors. In their problem, "The network serves as the communication, sensing and computation medium for the robots, whereas the robots provide actuation, which is used among other things for network management and updating the network state." During execution the nodes probabilistically determine the optimal direction of travel for the mobile robot units. This direction is dependant only on the current state of the mobile unit and the current direction of travel. This method of directional guidance is similar to the method discussed in this paper. Both created a stable sensory network and guided the mobile robot based on the sensory inputs. However, unlike the Batalin et al. method, the mobile robot described in this paper has sensory capabilities. These capabilities were retained to allow the mobile robot to continue its task in the case of faulty or incomplete information from the stationary robots. This would allow the mobile robot unit to successfully contain a threat despite a system shutdown.

In addition to being able to travel to a final location, it is important for a robot to know of its position within an environment. Chintalapudi et al. discussed this topic in "Ad-Hoc Localization Using Ranging and Sectoring" [5]. Ad-Hoc localization depends on the utilization of anchor nodes that already know their position within a common coordinate system. Chintalapudi et al. utilized a position fixing system that incorporated bearing and range information to localize nodes. They use RF-based sensing to establish the distances between the various nodes within the system. When the position of one of the anchor nodes is established, the position of all the nodes within the system can also be established. Such a process is not used in the current iteration of the code in this paper as the mobile robot's motion trackers were used to establish position. Please turn to the Future Work section for a description of its future use.

Problem Definition:

The primary problem addressed in this paper is the identification of the location of a spot within an area of fixed boundaries. The secondary problem is to send a mobile robot to the location of the spot once its position has been identified. The tertiary problem is to track the motion of the spot after arriving at the identified location.

Methods:

Our solution (trigonometric solution) to the light location problem is composed of the components illustrated in the following system diagram:

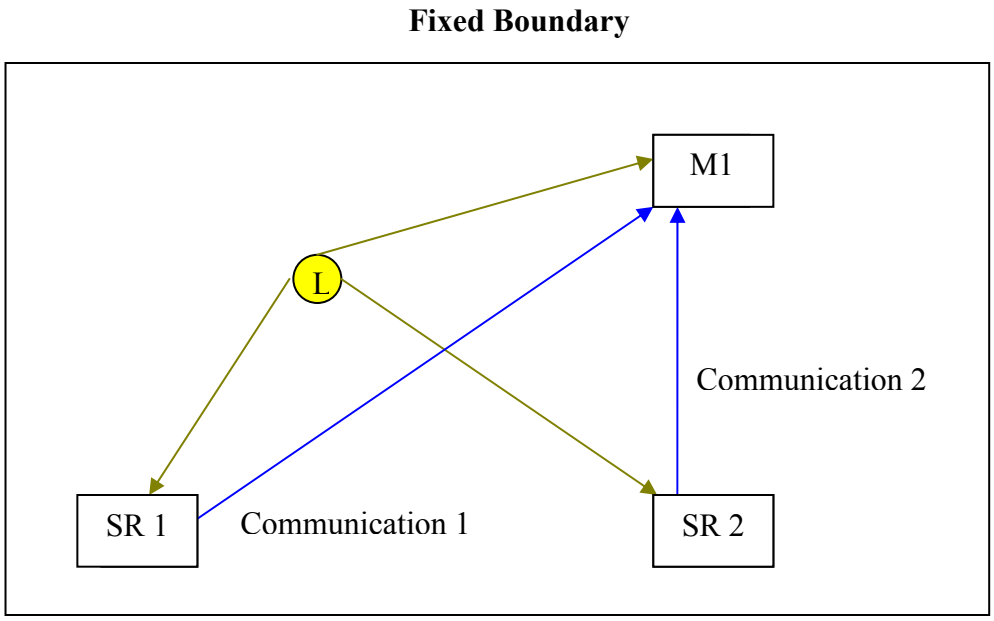

Figure 1: System Diagram

There are three robots: two that are stationary and one that is mobile. The two stationary robots receive input from the light source and transmit output to the mobile robot. The mobile robot then acts on the communicated data to determine the position of the light and move towards it. It then receives the input of the light source and uses this data to arrive at the correct final location.

This trigonometric solution was chosen due to its ease of implementation. In this design, only one mobile robot was needed. This decreased the amount of positional error. There were also two other solutions proposed. The first solution involved the use of two robotic arms (Figure 2). In this setup, the two arms would slide along two adjacent edges of the boundary and locate the one-dimensional position of greatest light. This solution was not chosen because it would require the accurate motion of three robots and because the same coordinate location scheme could be achieved by using the trigonometric solution.

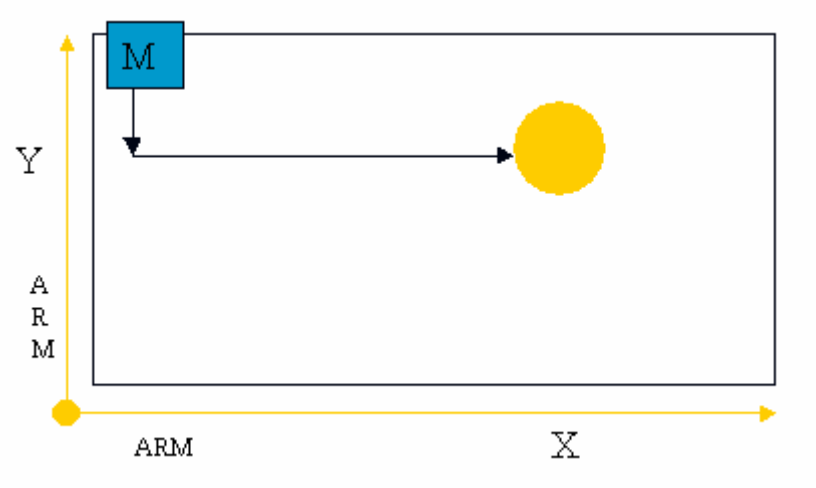

Figure 2: Sliding Arms Solution

The other solution considered involved the use of four mobile robots. The four robots would communicate to determine which of the four quadrants contained the spot of light (Figure 3). They would then break the selected quadrant into four smaller quadrants and iterate. This solution would have been difficult to implement because it would have required precise positional accuracy from four mobile robots and because it would have required that the robots determine in which quadrant the light spot lay.

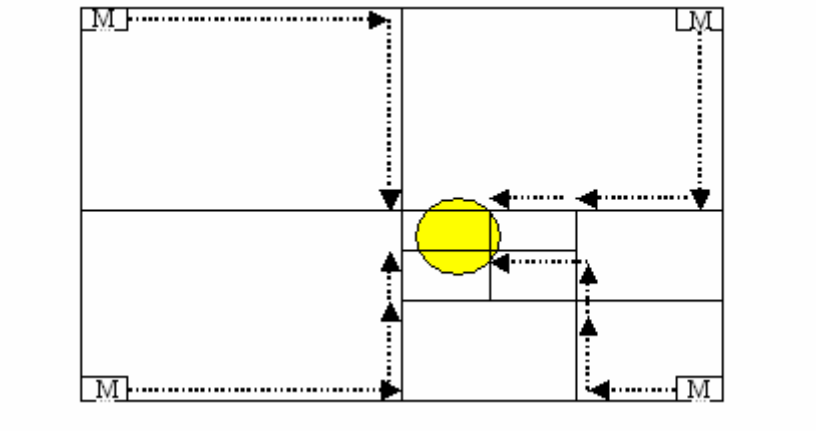

Figure 3: Quadrant Solution

After choosing the trigonometric solution, the robots were prototyped using LEGO Mindstorm kits and the associated ROBOLab software. These tools made it possible to test the validity of the proposed algorithm before implementing the system using more expensive and complicated components. Additionally, this process indicated the basic limitations of all robotic systems and allowed for further refinement of the design of the algorithm.

The stationary robots are detailed in the following manner:

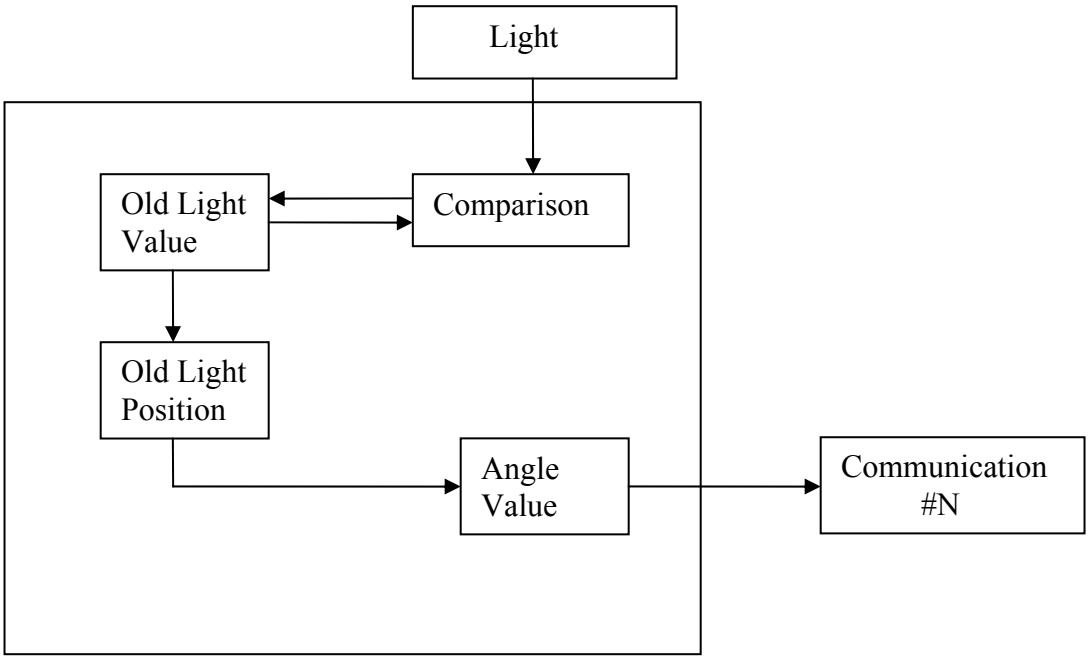

Figure 4: Diagram of Stationary Robot Functionality

The stationary robot receives input from the light source. It compares this input value to a previously stored light value. If this light source is greater than the previously stored value it records the position and intensity of this new highest value light location. It iterates over this process 30 times (90-degrees). Following these comparisons the stationary robot converts the

position of the greatest light intensity into an angle measure and transmits the angle measure to the mobile robot using BlueTooth communication.

The mobile robot is detailed in the following manner:

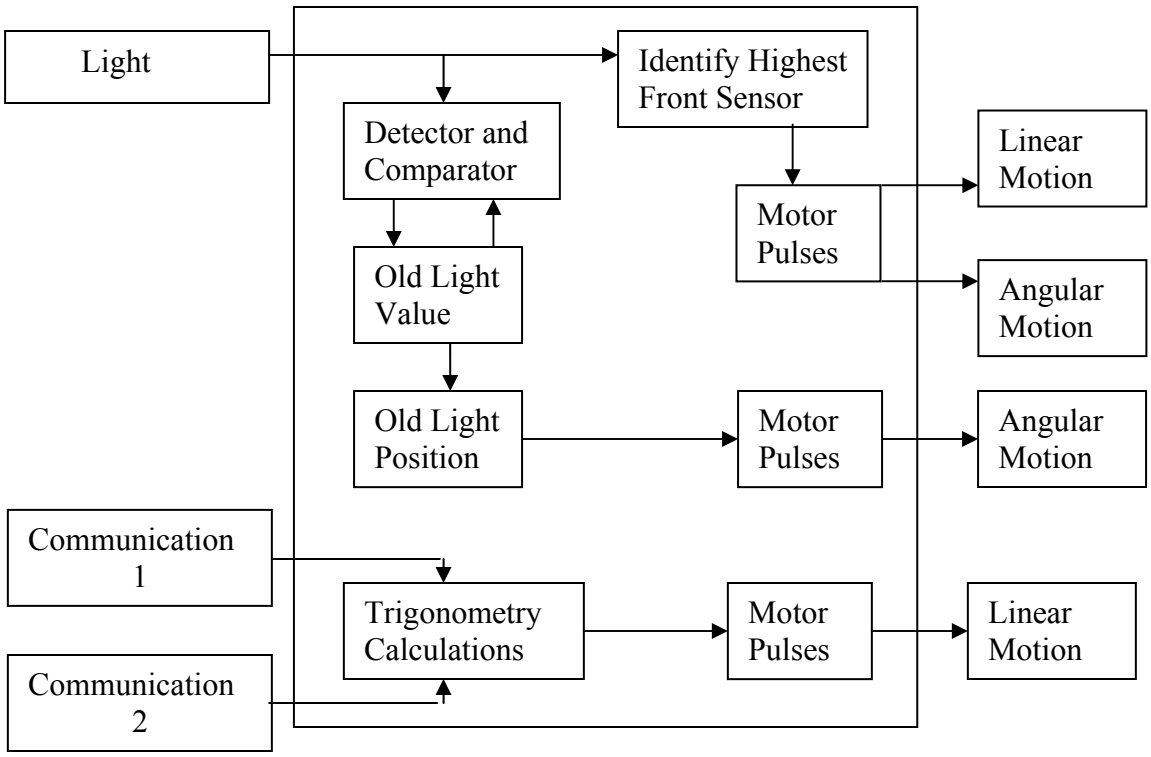

Figure 5: Diagram of Mobile Robot Functionality

The mobile robot receives two angles from the stationary robots (communication 1 and 2). It performs trigonometric calculations on these two angle values and computes a horizontal and vertical displacement. It converts these distances to a number of pulses to achieve linear motion and arrives at the light source. The mobile robot also receives a light value. It compares this value to a previously stored light value. If the current light value is found to be greater than the previously stored values it saves the light value and position. Upon the completion of the 31

iterations (360-degrees) it orients itself in the direction of greatest light intensity by converting the positional value into an appropriate number of pulses to achieve angular motion. Once the mobile robot is oriented, it follows the light as the light moves using the input from its three front light sensors (Figure 9).

After designing the desired solution, we further constrained the problem through the selection of hardware. The first constraint was the physical ability of a robot to find light. If the robot had a limited range of detection, the size and strength of the light source would be important. Likewise, the presence or absence of diffusion emanating from the light source had to be determined. A system with diffusion would allow the robot to scan an area while remaining stationary while a system without diffusion would not. Additionally, the size of the boundary in which the spot would be found relative to the size of the robot was important to consider. The robot could not be larger than the boundary specified. Additionally, the size of the robot influenced the necessary accuracy of both the algorithm and the light detection apparatus. A robot that is considerably larger than the spot or the boundary conditions could be given coordinates with error and still arrive at the correct location due to the relative size of the body.

After identifying the constraints and the problem statement, the style of the spot of light was defined. The area that did not contain the spot was represented by black. The spot's characteristics were also defined. During the first term, the robot must locate one stationary light spot within the specified boundaries. In the second term, the robot must locate one mobile light spot, again within specified boundaries. This mobile light spot would remain stationary until the robot converged on the specified location, at which point the light would begin to move.

The physical boundaries of the light were defined by the size of the HIT table of the TUFTL lab (usable space: 37" x 28.5"). The light spot was defined as the output of a projector resting underneath the HIT table (see figure 6).

Figure 6: The Setup of the Table and Spot of Light

The table to be used was a surface of frosted glass. Below the frosted glass was a projector. The projector reflected off a mirror below the table and was then projected up onto the table. The image projected was the image seen on the secondary screen of the Apple Computer. The spot was defined and placed using Microsoft PowerPoint (Figure 6).

After defining the physical layout of the system, the mathematical description of the space was formulated. Based on the solution developed, a method was created to locate a spot within fixed boundaries given angle values from two robots a set distance apart. The two robots defined as stationary robots were placed on adjacent corners of the table (Figure 7). The two robots were stationary to create a fixed coordinate awareness. If these two positional robots were allowed to move, it would not be possible to maintain an absolute positional awareness. The resulting mathematical setup was as follows:

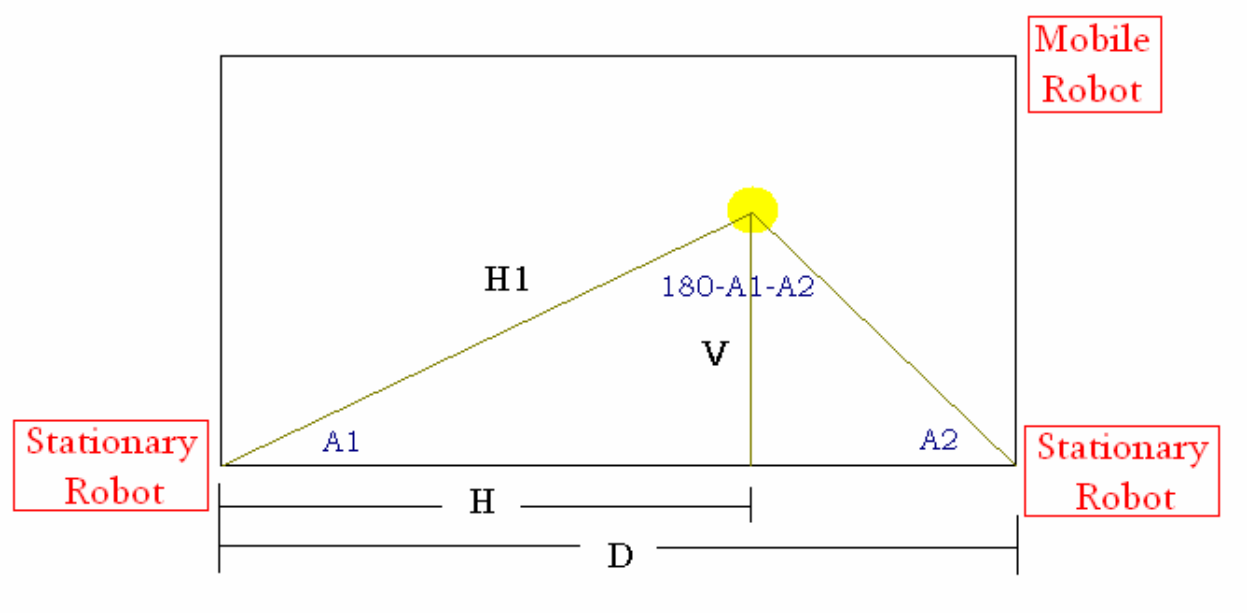

Figure 7: Experimental Setup Using Stationary Robots

When provided with the angle vector, it was possible to construct the exact location using the following formulae:

Known: A_1, A_2, D

$$\sin(A_1) / H_2 = \sin(A_2) / H_1 = \sin(180\text{-}A_1\text{-}A_2) / D \quad (1)$$

$$H_1 = D\sin(A_1) / \sin(180\text{-}A_1\text{-}A_2) \quad (2)$$

$$\mathbf{V = H_1 \sin(A_1)} \quad \mathbf{(3)}$$

$$\mathbf{H = H_1 \cos(A_1)} \quad \mathbf{(4)}$$

The equations for the vertical component, V, and the horizontal component, H, represented the position of the light in a coordinate plane with the stationary robot on the left at position (0,0). These coordinates were altered to reflect the actual location of the mobile robot (Table Width, Table Height) within the mobile robot algorithm as seen above (figure 11).

After defining the mathematical analysis, the design process began. Many of the components were almost identical across the two types of robots and will therefore be discussed

together. The microprocessor used to control the robots was the OOPic-R chip. OOPic stands for Object-Oriented Pic chip (Figure 8). This chip was chosen for its ease of use. The compiler included with the OOPic chip allowed for programming in C++, Basic, or Java, which eliminated the start-up time associated with learning a new coding language. Additionally, the OOPic-R chip has an object library. These objects provided the interface to the hardware devices used, allowing for efficient and clear coding. Using the serial interface port, the OOPic chip could be connected to a computer for programming. The code is then written and stored in the EEPROM. The EEPROM retains the code even after the power has been disconnected from the OOPic chip. The OOPic interface board has 31 input/output pins and power and ground sources available to auxiliary devices.

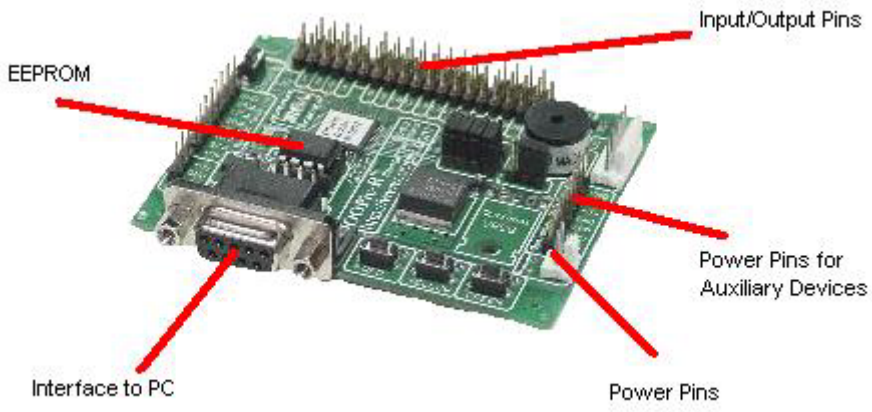

Figure 8: OOPic Chip

The first aspect created was the light sensor (figure 9). The light sensor was a simple voltage divider consisting of a resistor and a photoresistor in series. The output was taken at the node connecting the two resistors and inputted to the OOPic's analogue to digital conversion pin and converted to an 8-bit value.

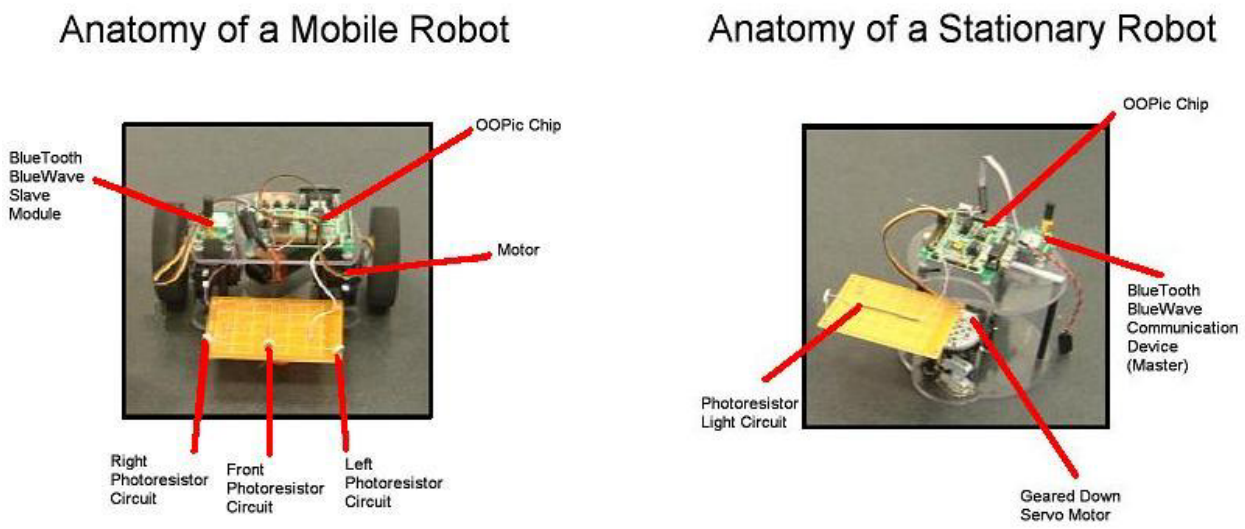

Figure 9: Anatomy of Mobile and Stationary Robots

After the light sensor unit was designed, a mechanism for controlling the motors was created. The motors used in the project were modified servomotors (the mechanical stops were removed) programmed to behave as stepper motors. These motors were stimulated using voltage pulses from the OOPic-R board. They were controlled through the alteration of the pulse duration such that one pulse is equal to one inch of forward motion. Once this was created, the mobile robot could travel a reliable distance based on the input of a set number of pulses when finger resistance (see Discussion section) was applied. The mobile robot could also travel in a circle by powering itself with only one motor.

The communications system used by the robots in the final design was BlueTooth. Originally, the robots communicated using Infrared Communications. However, this system was impacted adversely by ambient light and had a short communication range. In the second semester BlueTooth communications were used. In this system, the two stationary robots were given master devices and the mobile robot was given the slave device. This allowed the robots to transmit serial data accurately between the stationary and mobile robots up to 100-yards.

After detailing the functionality of the subcomponents, the timing diagrams were created. The timing diagram for the stationary robot was as follows:

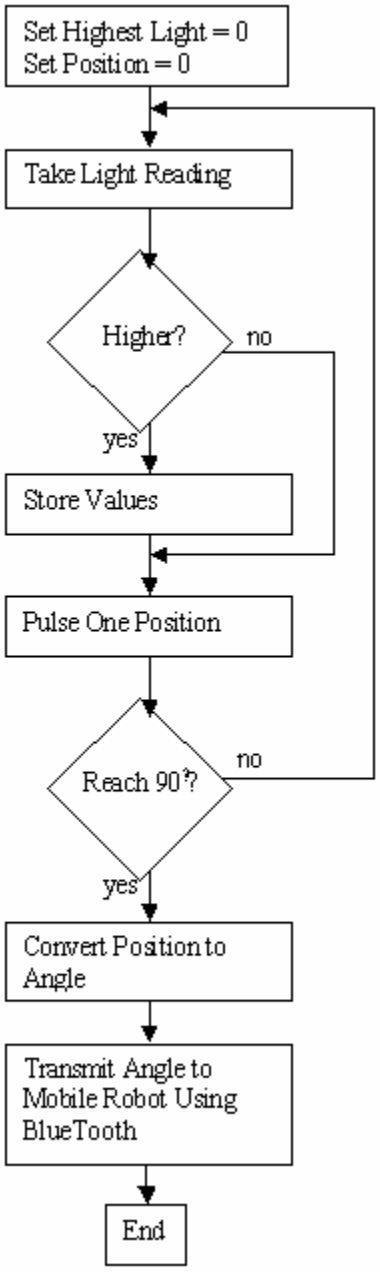

Figure 10: Timing Diagram for Stationary Robot

The stationary robots were placed in two adjacent corners of the boundary illustrated in figure seven. At the commencement of operation the stationary robot reads in the value of light at its initial position of zero degrees, which is greater than zero by default due to the constant presence of ambient or diffused light. The position of zero-degrees is also stored as the position of the greatest light intensity seen thus far. The stationary robot then sends a voltage pulse to its motor to advance the light sensor apparatus approximately three-degrees. The light reading at this new position is then taken. If this value is higher than the previously stored highest value (at zero-degrees) then the position and intensity value are stored as the highest light intensity and position. This process is repeated until the ninety-degree stopping condition is reached. The stored position of the greatest light intensity is then converted into an angle value within the appropriate range of the OOPic's cosine function. This value is then transmitted to the mobile robot via BlueTooth communication.

Upon completion of the stationary robot algorithm, the mobile robot algorithm was detailed. The timing diagram is as follows:

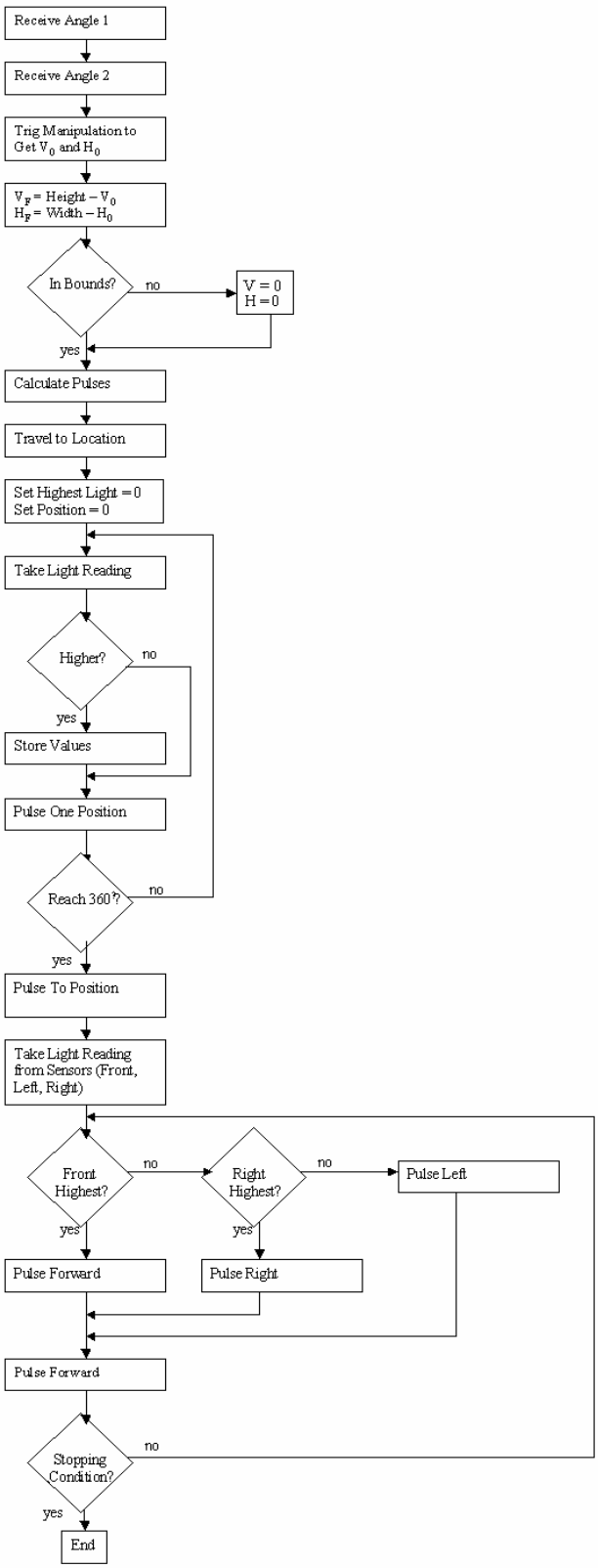

Figure 11: Timing Diagram For Mobile Robot

The mobile robot receives, via BlueTooth communication, two angle values from the two stationary robots. It then converts these two angle values to positional vectors based on formulas three and four detailed earlier. The calculated position vectors describe the necessary motion specified from the lower left-hand corner. However, the mobile robot was actually located at the upper right-hand corner to avoid interfering with the stationary robots' light-finding algorithms (Figure 7). The new positional vectors were calculated using the following formulae:

$$\text{Vertical} = \text{Height\_of\_Table} - V_{calculated} \quad (5)$$

$$\text{Horizontal} = \text{Width\_of\_Table} - H_{calculated} \quad (6)$$

If these terms are found to be out of bounds (Horizontal or Vertical less than 0 or greater than bounds) then the vertical and horizontal vectors are set to a value of 0. These position vectors are translated into linear motion by translating the distance into a specific number of pulses of constant duration.

Once at the proposed location of greatest light intensity the mobile robot stops. It then takes in the light reading at its current position (position zero). It stores that position as the position of greatest light intensity and the light value as the greatest light value. Then, using a similar algorithm to that of the stationary robot it determines the position of greatest light intensity within a 360-degree sweep. After it has completed the sweep and has thus returned to its original position it pulses back to the stored position of greatest light intensity. This centers the front light sensor on the spot of greatest light (Figure 9). The light source then begins to move. For each of the following iterations, the robot determines which of the three light sensors has the greatest light reading. It then pulses in the direction of that light sensor to center the front light sensor on the light spot. This process is repeated until a programmer specified stopping condition is reached.

Results:

In the final presentation, the robots worked correctly when provided with accurate input and given the motion faults (please see Discussion section). The stationary robots were able to correctly identify the position of greatest light intensity. From this information they were then able to communicate the calculated angle values to the mobile robot using BlueTooth communication. The mobile robot was then able to use the received angles to calculate the correct position of the light. Using the constructed pulsing protocol, the mobile robot was able to advance correctly to the position of light when provided with finger resistance (please see the Discussion section). Once at the position the robot was able to rotate to point in the direction of the light source and then follow the light source as it moved.

Discussion:

Despite the overall functionality of the system there were difficulties with ambient light sources, faulty motors, and code bugs. Each robot's output was strongly affected by the ambient lighting present in a room. Every room had a window, or a computer monitor, or other small light sources that the light sensors detected, skewing the input values on the robots. It was therefore very important to isolate the robot from these sources to the greatest extent possible. This was often done by covering up the various light sources with cardboard or turning them off. In extreme cases it meant orienting the robotic setup in a different manner so as to minimize the effect of residual glare. These light sources were a problem because they often caused the light sensor to register a higher light intensity value than the value resulting from the experimental light source. This led to inaccurate greatest light position values and thus inaccurate angle

measures. These faulty measures usually resulted in horizontal and vertical values outside the acceptable bounds.

One of the ways we found to minimize the effect of the ambient light sources was to carefully chose an appropriate light source. It was important for the light source to be minimally diffused. We found that light sources designed to light up large areas uniformly (such as Tap Lights by American Tack) resulted in highly inaccurate results. Since there was very little variation in light intensity over a large area, the angle values returned by the two stationary robots did not necessarily correspond to the location of the center of the light source. As a result, the intersection point determined by the two angle values was often not within the physical boundaries of the table.

The motor pulses were difficult to manage as well. During the testing phase of the project, we found that one of the motors on the mobile robot and one of the stationary robots was incapable of reversing direction. This was not a major concern for the stationary robot. Since two stationary robots were built and one of the stationary robots possessed a motor that could reverse, the robot that couldn't was programmed solely to go in the forward direction. However, the mobile robot code demanded that both wheels reverse at specific points during the light following section of the code (figure 11). Therefore, to overcome this problem, the faulty wheel was turned off when it was necessary to turn in the prohibited direction. However, this created a disparity in angular distance traveled because in the functional direction both wheels could be powered. To overcome this, when the robot needed to turn in the direction permitted by both motors, only the forward motor was pulsed to equate the angular motion to the faulty condition.

The motors also presented a problem during the forward movement of the robot. The wheels appeared to be of unequal strengths. This prohibited the robot from moving in a straight

line and instead resulted in motion resembling an arc. This was a problem because it hindered the accuracy associated with the "Where am I" problem. To fix this, we placed a restraining finger on the side of the robot during forward motion. This allowed the robot to travel in a straight line and thus retain spatial awareness. It was found after the presentations that these faulty motors were all lacking their potentiometers (they had been removed along with the mechanical stops). Once they were replaced the motor problems were eliminated.

We also encountered errors with our communication system during testing. We purchased our three BlueTooth modules in two separate orders to decrease the amount of money lost in the case that BlueTooth was not used. However, the third BlueTooth we received was a master/slave device from a newer model batch. This new device would not act as a slave in the presence of the original master device. We were therefore forced to put the two master devices on the stationary robots and the one slave device on the mobile robot. Unfortunately, as a result of this configuration, the mobile robot could not contact the two stationary robots but instead needed to wait to be contacted. When we first began to implement this functionality we had difficulty with timing. Occasionally when both BlueTooth masters attempted to connect with the slave, one of the masters would not fully connect and would therefore not send the data correctly. To correct for this, we put a delay in one of the stationary robot codes so that the second stationary robot would not attempt to connect to the mobile robot until the first stationary robot had disconnected. This eliminated the problem of missing communications.

Once the BlueTooth was working correctly we began to debug the code on working models. We found that the easiest way to debug the code was using the OOPic interface program. This program allowed the programmer to directly access the register values of the

OOPic chip by simply clicking on the variable names specified in the program. This feature allowed us to verify the correctness of the calculations.

We also used robot motion as an indicator of both error and success. Before we added our out of bounds condition, when the robot received incorrect angles it would travel an infinite distance (until the reset button was hit). This was an indication that the received values were not correct.

One important concern in the design of the algorithms was delay time. It was important to create a program structure that would either eliminate or severely limit the amount of time spent waiting for inputs, outputs, or the results of calculations. The greatest source of delay in the system was the BlueTooth communication. Due to errors received when both master devices attempted to communicate with the slave simultaneously, there was a delay implemented to allow the communications modules ample time to send their data. However, the motion aspects of the robots were not delayed. Both robots executed their light sensing algorithms simultaneously. Therefore, when the communication channel was opened, the robots already had the data that needed to be sent. The run time was also decreased by allowing the mobile robot to calculate all of the coordinates. Centralized processing resulted in fewer communications and pauses in communications, which further decreased the run time.

The run time would have decreased further if the two stationary robots possessed the slave devices and the mobile robot possessed the master device. This would have enabled the mobile robot to access the stationary robot data whenever it needed instead of waiting to be contacted by the stationary robot. This also would have eliminated the need for the hard coded delay. Since the mobile robot's BlueTooth master would be the only device initiating contact there would not have been a problem of two devices attempting to contact a third device at once.

Additionally, there would not have been any error identifying which angle came from which robot as the master device could specify which device it wished to communicate with at each specific time by individual address.

Another very important consideration was separation lag time. If while executing a command, the robot encountered a hazardous situation it was important to stop the robot or alter the behavior of the robot. It was therefore important to make the robot sufficiently independent so as to avoid potentially hazardous situations. Many of these self-protecting checks were not included in the original design.

Despite the lack of self-preservation checks included, error checking was inherent in the design, which ensured that the mobile robot would arrive at the correct location of the spot. The stationary robots provided the observed coordinates of the light spot. However, there existed the possibility that one or both of the two stationary robots could have been negatively influenced by an ambient light source. As a result of this possible inaccuracy, the two robots occasionally passed the mobile robot incorrect positional information. Using this incorrect data, the mobile robot would then calculate incorrect angles. However, due to the second half of the code, the light following portion, the mobile robot was able to locate the source of light and follow it, ultimately arriving at the correct position.

The effects of the inaccuracies discussed above could have been lessened if the stationary robot code included multiple sweeps. In a multiple-sweep algorithm, the stationary robots would scan through the ninety-degree section multiple times. They would then average the angle positions recorded to determine the final optimal position. This would decrease the effect of light sources with high diffusion. This functionality was not implemented originally due to the inability of the altered motors to reverse direction.

If the stationary robots were made to sweep throughout the entirety of the process, they would be able to indicate the final position of the moving light source more accurately than the mobile robot. Instead of the user relying on the relativistic positional information provided by the mobile robot, the stationary robots could locate the precise location of the light using the previously implemented sweeps. This was not attempted during the final presentation due to the limited motion previously discussed associated with the motors with missing potentiometers.

Lessons Learned:

There are several components that must be integrated to produce a cohesive system design. These parameters are: design, data, control, tools, quality, and integration. When detailing the software design of the robots these topics must be analyzed to ensure a cohesive design and implementation. Detailing each of these components sequentially and individually allows the members to progress in an ordered manner because they would be more aware of the project status.

The first step to consider was design. In this step the number and placement of the robots was defined. After this definition, the processes could be detailed clearly. During this process it is important to create a system diagram for both the components and the system as a whole. The creation of such a diagram provides the researchers with a reference point that allows them to constantly check the progress and schedule. It is necessary to detail both the subcomponents of each stage and their interrelationships so that there is a clear understanding of the inputs and outputs that can be shared across modules. This also allows the researchers to determine the necessary loads on each of the microprocessors.

After detailing the design the researchers then move to the data step. In this step they explicitly identify the variables and data structures to be used. This elucidates the variable interrelationships and thus indicates which can be shared across robots. These definitions save memory space by determining the maximum size of each of the variables and the necessary number of variables.

After describing the variables and data structures it is important to identify the file structure and format of the system. In this step the programming language is chosen. This choice should be based on the languages available and the languages compatible with the compiler and the microprocessor. In addition to a coding language, it is also important to choose a coding format. A structured format enforces a readability that is important when working in a team. This readability allows team members to understand the code written by various programmers. This also allows for coding by multiple individuals. This style of coding requires that the coding structure by standardized so that all coding members can understand and integrate all of the coded sections. This type of readability can be enforced through variable name regulation and well-commented code.

As a result of the language chosen, an associated compiler and source code samples may be provided. However, as a result of a reliance on the predefined code provided by the libraries, the developed code is not usable by chips that do not have access to the libraries used. If this code was to be expanded to other chips and platforms, the routines used should be written out so that the chips are not reliant on the outside functionality of the language.

The quality stage must be allotted a large amount of time. In this stage, the created system is tested for accuracy. For a system of even moderate complexity, it is important to test each of the components individually before trying to incorporate them into a cohesive whole. If

the software and hardware are integrated before the validity of either section is tested, it is unlikely that the errors will be found. Instead, the increased complexity will make it more difficult to determine which section is responsible for the error. Once each of the sections is determined to be preliminarily correct, the sections can be assembled and further tested. At this point in the testing, it is likely that any bugs are due to inconsistencies across hardware and software or between expectations and realizations of the system.

This project and testing process provided a broad introduction to robotics programming and limitations. One such limitation was processor speed. The processor was not infinitely fast. As a result of these restrictions, values could not be inputted and calculated instantaneously leading to lag time. This lag time presented a problem in the stationary robot light algorithms. Originally the stationary robot motors were not geared down (Figure 11). The robots pulsed through the stops extremely rapidly. The processor could not input and store the values of light intensity as quickly as they were coming in. As a result of this many light intensity values were lost and the angle returned was meaningless. Therefore, we found that it is very important to determine the approximate speed and capabilities of the microprocessor before designing an algorithm to be used

The microprocessor also limited the design of the project discussed. Originally we planned to include an LCD display on the top of the mobile robot so that the robot would be able to constantly update the observers as to the progress of the robotic system. However, we found that if the number of inputs or outputs to the OOPic-R chip became too numerous, the chip would not function correctly. When overloaded, the chip would not stay powered and instead the power would flash on and off at irregular intervals. To fix the problem, we only needed to remove the EEPROM chip from the board and reinsert the chip with the power on. This wiped

the memory of the EEPROM chip and allowed us to send the OOPic-R chip a new program with less inputs and outputs. However, this restriction altered our original design plan. We had planned to include an on/off switch, which was later removed from the design, as it was superfluous. It is important to note the capabilities of all the components used before designing the programming for any of the components.

Future Work:

If this project was to be extended there are several self-preservation checks that could be added. Primarily, the robot should include a check to ensure that it does not fall off a table. This could be accomplished using a touch sensor designed to protrude slightly in front, behind, and to the sides of the robot. If any of these sensors detected empty space (instead of ground) the robot would stop and travel in the direction opposite to the empty space sensor reading, thus avoiding the edge of the table. The mobile robot could also include touch sensors to prevent it from getting stuck in one location. In the case where the robot is enclosed within its boundaries by a wall, the touch sensors could extend beyond the boundaries of the robot. If the sensors detected the presence of a wall, the robot could once again reverse to avoid the wall.

Xu et al. discussed a concept for obstacle avoidance in, "Concepts for Dynamic Obstacle Avoidance and Their Extended Application in Underground Navigation" [13]. The algorithm discussed by Xu et al. allows a robot to dynamically avoid an obstacle and return to its prior path once the obstacle has been passed. To do this local points on a path or in the public space example, points within a train station, are flagged. These flags allow the robot to return to the path once an obstacle has been passed. The robot calculates its projection range, which is much smaller than the extent of its visual scanners. Objects within this projection range are checked to

determine whether or not they could cause a collision with the robot. The robot also constantly maintains a Polar Object Chart, which is a representation of the robot's surrounding area. It uses this chart to determine possible ways of avoiding obstacles. If a given object is close to another object, the two objects are clumped into a cluster, representing an area through which the mobile robot cannot pass. The robot then marks the way out sectors with a one to differentiate between the possible paths and the obstructed areas. An optimal path is chosen from the remaining paths. The robot stays on this detour path until the obstacle has been passed. Once the obstacle has been passed the robot returns to its original path using the flags. The difficulty with Xu's algorithm is that it requires that the robot have a predefined path. However, the physical structure of the enclosed area could be altered to reflect this requirement. Flags could be placed in the train station and activated by the mobile robot after it calculates its path. This would allow the robot to mimic a predefined path and use the obstacle avoidance algorithm.

In the planning stage of this project the possibility of defining a problem with multiple spots of light was discussed. It is necessary to create a mobile unit team that can successfully determine the threat priority and visit the sights accordingly or a team with multiple mobile robots that would visit the sights in parallel manner. Jung and Sukhatme discussed the possibility of tracking multiple targets in their paper entitled, "Tracking Targets Using Multiple Robots: The Effect of Environment Occlusion" [6]. In this paper, an algorithm is presented to solve the problem, "of tracking multiple anonymous targets in a bounded planar environment using a network of communicating robots and stationary sensors. [6]." They based their solution on the assumption that the environment could be separated into areas that are topologically simple through the use of landmarks as boundaries and that given two regions of equal area, the region with the higher concentration of targets should have more tracking robots. The algorithm

maintained two continuous estimates of the robot density and of the target density. Each robot engaged in tracking targets transmits its position and the position of its targets over the network. Each robot is then able to maintain an estimate of the target and robot density using its own values and the values received from the other tracking robots. If a robot suddenly becomes available (i.e. it is no longer tracking a target) it searches for an area that most needs its tracking services. The robots are also transferred between areas based on target and robot density within a given area. This algorithm would be an interesting addition to the problem because it would provide the network with a comprehensive method for maintaining awareness of all targets within a given boundary. However, this algorithm requires a well-defined positional awareness, which in Jung's paper was determined using odometry in conjunction with drift compensation using laser beacons spread throughout the environment at region boundaries. The difficulty with this arrangement is that if a laser beam were to be disabled, the system would be without error checks.

Multiple robot systems require an increased level of sophistication and complexity, necessitating different control algorithms. Barfoot and Clark discussed such a method in, "Motion Planning for Formations of Mobile Robots" [1]. This algorithm explicitly plans the trajectory of each of the mobile units, rather than rely on the units to maintain a set distance away from the designated leader. This eliminates range calculation errors. In this algorithm, a general trajectory is presented. Each robot then calculates its offset value based on the position of the robot within the robot group. The robot is aware of its position relative to the group based on the inclusion of an overhead camera positioning system. To ensure that the proper path is followed, once specified, each robot contains an individual feedback control unit. This type of an algorithm would be useful in the case where more than one mobile robot would be sent to the

spot. If the danger of the spot (in the case of a chemical spot) was great, it would be preferable to send multiple containment units to ensure success even if some were damaged. Therefore, a relative positioning algorithm would be useful. However, the algorithm's reliance on an overhead camera is undesirable because all of the robot's positioning information depends on an outside component that could be disconnected in the case of an attack. It is therefore important to develop a method to maintain positional awareness without a camera unit.

Olson et al. discussed a method for determining autonomous positional awareness and navigation in "Rover Navigation Using Stereo Ego-Motion" [9]. In this algorithm, "landmarks are tracked in an image sequence and the change in camera position is determined for each frame by estimating the relative movement of the tracked landmarks in the camera frame of reference [9]." This algorithm determines the motion of the unit by using calibrated cameras to capture two or more pairs of stereo images. The method then identifies features to use as landmarks. These landmarks are chosen because their 3-D position can be estimated precisely in successive iterations. The 3-D position of the landmark in subsequent iterations is determined by matching the pair of images to the original stereo pair. The position of the landmark is determined by using triangulation to determine the position of the landmark relative to its prior position. Since the location of its prior position is known, the location of its current position can be calculated. The motion is then estimated using Gaussian error distributions on the positions of the landmarks. This algorithm would allow the mobile robots to move within the environment while maintaining an accurate positional awareness. In a trial conducted by Olson et al. a robot navigated 20 meters, taking readings every 10cm. At the conclusion of the path, the robot had only obtained a 1.2% error. However, due to the required level of inputs and calculations, this

algorithm would not be possible using the microprocessors, power system, and robot structure discussed in this paper, as it would require additional resources.

In future iterations of the robot team design different sensing equipment will be used to locate different types of spots. As the design of this project was based on the chemical nose problem, we spoke of implementing a simpler version of this aroma-finding problem. The benefit to changing from a light source to a scent is that the nose is calibrated for a particular chemical and therefore no ambient scent-sources exist. However, as chemicals diffuse, there is the added difficulty of including a fluid-flow algorithm to determine the origin of the chemical leak. More recently, we have spoken of using sonar equipment to find a specific object within a certain boundary. However, the difficulty with such an algorithm is that it is likely that the stationary robots would actually find each other or the mobile robot before finding the object of choice. Therefore, a more complicated algorithm or sensor setup would be needed to find only the object and not any of the other robots or the boundary. Additionally, in future iterations of this process, sonar detectors will be placed on the stationary robots so once the mobile robot finishes tracking the spot, the stationary robots will be able to locate its final position. This will increase the accuracy of the final location declaration.

Conclusion:

It is possible to identify the location and track a spot of light on a table of fixed boundaries using two stationary and one mobile robot. The setup created worked well and when provided with the proper inputs, the stationary robots were able to identify the position of greatest light and the mobile robot was able to calculate the position and then travel to it (given its motion faults). Upon reaching the spot of light, the mobile robot was able to track the light

source as it moved. Therefore, a robotic team can be assembled to locate and travel to a spot in an area of fixed boundaries.

Acknowledgments:

I would like to thank Professor Chris Rogers for his inspiration and dedication to the spread of scientific knowledge, Dr. Ron Lasser for helping me to develop this thesis and the idea structure behind it, Matthew Dombach for all of the hours he spent making sure that our projects were secure and on target, and Ethan Danahy for his assistance during this process.

References:

[1]. Barfoot, T.D. and C.M. Clark. "Motion Planning for Formations of Mobile Robots." Robotics and Autonomous Systems. Volume 46, Issue 2. February 29, 2004. Pages 65-78. <http://www.sciencedirect.com/science/journal/09218890>

[2]. Batalin, Maxim A., Gaurav S. Sukhatme, Myron Hattig. "Mobile Robot Navigation using a Sensing Network." IEEE International Conference on Robotics and Automation. 26 April - 1 May 2003. <http://cres.usc.edu/pubdb_html/files_upload/367.pdf>

[3]. Buschmann, Carsten et al. "Grid Based Navigation for Autonomous Mobile Robots". Workshop on Positioning, Navigation and Communication (WPNC'04). 29 Mar 2004 . <http://www.ibr.cs.tu-bs.de/users/cbuschma/papers/Buschmann_etal_GridBasedNavigation.pdf>

[4]. Chintalapudi, Krishna Kant et al. "Ad-Hoc Localization Using Ranging and Sectoring". IEEE Infocom 2004. 7-11 March 2004. <http://www.ieee-infocom.org/2004/Papers/55_3.PDF>

[5]. Framling, Kary. "Reinforcement Learning in a Noisy Environment: Light-Seeking Robot". <http://www.cs.hut.fi/~framling/Publications/NNA04final.pdf>

[6]. Jung, Boyoon and Gaurav S. Sukhatme. "Tracking Targets Using Multiple Robots: The Effect of Environment Occlusion." Autonomous Robots Journal. Vol. 13, No. 3, pp. 191-205. November 2002.

[7]. Laubach, S.L. and J.W. Burdick. "An Autonomous Sensor-Based Path-Planner for Planetary Microrovers". IEEE International Conference on Robotics and Automation (ICRA'99). 10-15 May 1999. <http://robotics.jpl.nasa.gov/people/sharon/pubs/ICRA99.pdf>

[8]. Macera, Juan C., Philip H. Goodman, Frederick C. Harris Jr., Rich Drewes, James B. Maciokas. "Remote-Neocortex Control of Robotic Search and Threat Identification." Robotics and Autonomous Systems. Volume 46, Issue 2, February 29, 2004. Pages 97-110. <http://www.sciencedirect.com/science/journal/09218890>

[9]. Olson, Clark F., Larry H. Matthies, Marcel Schoppers, Mark W. Maimone. "Rover Navigation Using Stereo Ego-Motion." Robotics and Autonomous Systems. Volume 43, Issue 4. 30 June 2003. Pages 215-229. <http://www.sciencedirect.com/science/journal/09218890>

[10]. Russell, R. Andrew. "Robotic Location of Underground Chemical Sources". Robotica. Volume 22, pp. 109-115, 2004.

[11]. Stitzel, S.E., D.R. Stein, D.R. Walt. "Enhancing Vapor Senor Discrimination by Mimicking a Canine Nasal Cavity Flow." Journal of the American Chemical Society. Volume 125 (13), Pages 3684 –3685. 2003. <http://pubs.acs.org/cgi-bin/article.cgi/jacsat/2003/125/i13/pdf/ja028239y.pdf>

[12]. Thrun, S.B. (1992). "The Role of Exploration in Learning and Control." DA White & DA Sofge (eds.), Handbook of Intelligent Control: Neural Fuzzy and Adaptive Approaches. Van Nostrand Reinhold, New York.

[13]. Xu, Fuyi, Hendrik Van Brussel, Marnix Nuttin, Ronny Moreas. "Concepts for Dynamic Obstacle Avoidance and Their Extended Application in Underground Navigation." Robotics and Autonomous Systems. Volume 42. 2003. <http://www.sciencedirect.com/science/journal/09218890>

"Enhancing Vapor Sensor Discrimination by Mimicking a Canine Nasal Cavity Flow Environment." S.E. Stitzel, D. R. Stein, D.R. Walt, *JACS* , **2003**, 125, (13) pp 3684 – 3685

Appendix 1: Stationary Robot Code

```
'initialization for light finding

Dim Green as New oDio1
Dim Red as New oDio1
Dim Yellow as New oDio1
Dim Motor as New oServo
Dim Dummy as New oByte
Dim Photo as New oA2D
Dim MaxPhoto as New oByte
Dim Place as New oByte
Dim BatteryLife as New oByte
Dim Value1 as New oWord
Dim ValueF as New oWord

'initialization for serial communication

Dim output As New oSerialX
Dim input As New oSerialX


Sub Main()


'----initialize serial communication and go out of cmd mode---

  Red.IOLine=7
  Red.Direction=cvoutput

  yellow.IOLine=6
  yellow.Direction=cvoutput

  green.ioline=5
  green.direction=cvoutput

  input.IOLineS = 15
  input.IOLinef = 0
  input.Baud = cv9600

  output.IOLineS = 14
  output.IOLinef = 0
  output.Baud = cv9600

  input.operate=1
  output.operate=1


  output.String="AT+RESET"   'reset the device
  output.value=13

  oopic.delay=20


'------------stationary robot light finding-----------------------------------


            Motor.IOLine=13
            Photo.IOLine=4
```

```
        Photo.Operate=1
        Red.IOLine=7
        Red.Direction=cvOutput

        'THE NUMBER OF MOTOR PULSES DEPENDS ON BATTERY STRENGTH
'       BatteryLife=38
        BatteryLife=28

        MaxPhoto.Value=Photo.Value
        Place.Value=0
        OOPic.Delay=10
        For Dummy =0 to BatteryLife step 1

'               OOPic.Delay=5
                Motor.Center=28
                Motor.InvertOut = 1
                Motor.Operate = 1
                if Photo.Value < MaxPhoto.Value then
                        MaxPhoto.Value=Photo.Value
                        Place.Value=Dummy
                        Red.Value=1
                elseif Photo.Value > MaxPhoto.Value then
                        Red.Value=0
                End If
        Next Dummy

        'TOTAL SPREAD IS 64 (128-192)
        Value1=64*100/BatteryLife*Place

        'SEND THIS VALUE!!
        ValueF=(12800+Value1)/100

        OOPic.Delay=50
        Red.Value=0
        Motor.Operate=0

'----------------------bluetooth send---------------------------

  oopic.delay=500  'delay so that DTE has a head start


  output.String="+"
  oopic.delay=15
  output.String="+"
  oopic.delay=15
  output.String="+"
  oopic.delay=15
  output.value=13               ' 13 is code for carriage return

  'DTE should be in cmd mode

  output.String="AT+BWX"   'disconnect
  output.value=13

  oopic.delay=100  'delay for DCE to disconnect and reset itself before we try to connect (this delay is important!)

  output.String="AT+BWC=00:A0:96:1D:CC:C6,1111"
  output.value=13

  oopic.delay=10 ' delay to prevent echoed C in AT+BWC... from being mistaken for C in CONNECT

  dim stop as new obyte
```

```
  dim i as new oByte
  stop = 0
  do until stop = 1
    i = input.value
    If i=67 then  '67 is "C" the first letter in "CONNECT"
      green.value=1
      stop = 1
    elseif i=80 then ' 80 is "P" for "PAIR FAILED"
      ' did not get connect, red light on halt
      red.Direction=cvoutput
      red.value=1
      'exit sub

            'try to connect again
            output.String="AT+BWC=00:A0:96:1D:CC:C6,1111"
            output.value=13

    end if
  loop
  'we are connected


      green=1

  output.String="AT+BWE"
  output.value=13
  'we are in data mode

      red.invert
      yellow.invert

for i=1 to 20 step 1

      red.invert
      yellow.invert

'Send A and then valueF
            output=valueF
            'output=140
            output=75

green.invert
oopic.delay=10
green.invert
next i

green=0
yellow=0
red=0

oopic.delay=200

  output.String="+"
  oopic.delay=15
  output.String="+"
  oopic.delay=15
  output.String="+"
  oopic.delay=15
  output.value=13              ' 13 is code for carriage return

oopic.delay=200
```

```
output.String="AT+BWX"   'disconnect
output.value=13

oopic.delay=20

output.String="AT+BWE"   'Return to data mode
output.value=13


End Sub
```

Appendix 2: Mobile Robot Code

```
'-------------------------bluetooth send----------------------------

Dim output As New oSerialX
Dim input As New oSerialX
'-------------------------mobile robot moving-------------------

Dim RightMotor as New oServo
Dim LeftMotor as New oServo
Dim Counter as New oByte
Dim Counter2 as New oByte
Dim RH as New oBit
Dim FH as New oBit
Dim LH as New oBit
Dim Right as New oA2D
Dim Front as New oA2D
Dim Left as New oA2D
Dim HighestLight as New oByte
Dim Position as New oByte
Dim Dummy as New oByte
Dim Red as New oDio1
Dim Yellow as New oDio1
Dim Green as New oDio1
Dim Angle1 as new oByte
Dim Angle2 as new oByte
Dim TableD as new oByte
Dim TableH as new oByte
Dim Vertical as new oWord
Dim Horizontal as new oWord
Dim H1 as new oWord
Dim J as new oWord
Dim K as new oWord
Dim SinAngle2 as new oWord
Dim CosAngle1 as new oWord
Dim SinRemainder as new oWord
Dim SumAngles as new oWord
Dim Extra as new oByte

Sub Main()

oopic.delay=500  'delay for bluetooth to stabilize before program starts
  Red.IOLine=7
  Red.Direction=cvoutput

  yellow.IOLine=6
  yellow.Direction=cvoutput

  green.IOLine=5
  green.Direction=cvOutput

  input.IOLineS = 14
  input.IOLinef = 0
  input.Baud = cv9600

  output.IOLineS = 15
  output.IOLinef = 0
  output.Baud = cv9600

  input.operate=1
```

```
  output.operate=1
'-------------connect to pic 1 and receive angle value-------

angle1=0
angle2=0
dim stop as new obit
stop=0

do until stop=1
if input.value=65 then    'DTE validation
        angle1=input.value
        red=1
elseif input.value=75 then  'SM validation
        angle2=input.value
        green=1
end if

if angle1>127 then
        if angle2>127 then
                stop=1
        end if
end if

loop


'---------------------------math---------------------------------


        Right.IOLine = 1
        Front.IOLine = 2
        Left.IOLine = 3
        Right.Operate = 1
        Front.Operate = 1
        Left.Operate = 1

        RightMotor.IOLine=8
        LeftMotor.IOLine=11

        Red.Value=1
        Green.Value=1
        Yellow.Value=1

        OOPic.Delay=100

        Red.Value=0
        Green.Value=0
        Yellow.Value=0

        'RECEIVE LIGHT STUFF RIGHT HERE
        'RECEIVE ANGLE1 AND ANGLE2!!
        TableD=30
        TableH=24
        'Angle1=145
        'Angle2=160
        if Angle1<(255+-64) then
                CosAngle1=(sin(63+Angle1)+-124)*8/10
        End If
        Extra=64
        Dummy=0
        If Angle1>(255+-64) then
                For J=0 to 63 step 1
```

```
                    if Angle1<255 then
                            Angle1=Angle1+1
                    End If
                    if Angle1=255 then
                            Angle1=0
                            Extra=(63-J)
                            J=65
                    End If
            Next J
            For K=0 to Extra+-1 step 1
                    Angle1=Angle1+1
            Next K
            CosAngle1=(sin(Angle1)+-124)*8/10
End If
Green.Value=1
SumAngles=Angle1+Angle2+-2*128
'I=sin(SumAngles)
SinAngle2=(sin(Angle2)+-127)*8/10
SinRemainder=((sin(255-SumAngles))+-127)*8/10
If SinRemainder=0 then
            SinRemainder=1
End If
H1=TableD*SinAngle2/SinRemainder
Vertical=H1*SinAngle2/100
Horizontal=H1*CosAngle1/100

Horizontal=TableD-Horizontal
Vertical=TableH-Vertical

If Horizontal > TableD then
            Horizontal=0
End If
If Vertical > TableH then
            Vertical=0
End If

RightMotor.Center=28
RightMotor.Value = -127
LeftMotor.Center=28
LeftMotor.InvertOut = 0

Green.Value=0

For Dummy=0 to Horizontal step 1
            RightMotor.Operate = 1
            LeftMotor.Operate = 1
            OOPic.Delay=15
            RightMotor.Operate=0
            LeftMotor.Operate=0
Next Dummy

For Dummy =0 to 10 step 1
Next Dummy

For Dummy =0 to 30 step 1
            RightMotor.Center=28
            RightMotor.Value=127
            RightMotor.Operate = 1
            LeftMotor.Center=28
            LeftMotor.InvertOut = 1
            LeftMotor.Operate = 1
Next Dummy
```

```
RightMotor.Operate=0
LeftMotor.Operate=0

RightMotor.Center=28
RightMotor.Value = -127
LeftMotor.Center=28
LeftMotor.InvertOut = 0

For Dummy=0 to Horizontal step 1
        RightMotor.Operate = 1
        LeftMotor.Operate = 1
        OOPic.Delay=15
        RightMotor.Operate=0
        LeftMotor.Operate=0
Next Dummy

'SETS A BASE VALUE FOR HIGHESTLIGHT
HighestLight.Value=100
Position.Value=0

'THIS SECTION ROTATES THE ROBOT 360 DEG.
'CHECKING FOR THE AREA OF BRIGHTEST LIGHT
For Counter = 0 to 31
        'THE MOVING PART
        For Dummy =0 to 2 step 1
                RightMotor.Center=28
                'RightMotor.InvertOut = 0
                RightMotor.Value=127
                RightMotor.Operate = 1
                LeftMotor.Center=28
                LeftMotor.InvertOut = 1
                LeftMotor.Operate = 1
        Next Dummy
        RightMotor.Operate=0
        LeftMotor.Operate=0

        'THE LIGHT PART
        If Front.Value>HighestLight.Value Then
                HighestLight.Value=Front.Value
                Position.Value=Counter.Value
        End If
Next Counter

RightMotor.Operate=0
LeftMotor.Operate=0
Oopic.Delay=25

'THIS BRINGS THE ROBOT BACK TO THE SPOT OF BRIGHTEST LIGHT
For Counter = 0 to Position.Value
        'THE MOVING PART
        For Dummy =0 to 2 step 1
                RightMotor.Center=28
                RightMotor.Value=127
                RightMotor.Operate = 1
                LeftMotor.Center=28
                LeftMotor.InvertOut = 0
                LeftMotor.Operate = 1
        Next Dummy
        RightMotor.Operate=0
        LeftMotor.Operate=0
Next Counter
```

```
For Counter2 = 0 to 10 step 1
        'ROBOT GOES FORWARD FOR 10 BEATS
        RightMotor.Center=28
        RightMotor.InvertOut = 0
        RightMotor.Operate = 1
        LeftMotor.Center=28
        LeftMotor.InvertOut = 0
        LeftMotor.Operate = 1

        OOPic.Delay=20

        RightMotor.Operate=0
        LeftMotor.Operate=0

        'THIS IS WHERE I FIND OUT WHICH SENSOR SEES THE BRIGHTEST LIGHT
        HighestLight.Value=Right.Value
        RH.Value=1
        LH.Value=0
        FH.Value=0

        If Front.Value > HighestLight.Value Then
                HighestLight.Value = Front.Value
                RH.Value=0
                FH.Value=1
        End If

        If Left.Value > HighestLight.Value Then
                RH.Value=0
                FH.Value=0
                LH.Value=1
        End If

        If RH.Value = 1 Then
                For Dummy =0 to 15 step 1
                        'RightMotor.Center=28
                        'RightMotor.InvertOut=0
                        'RightMotor.Value=127
                        RightMotor.Operate = 0
                        LeftMotor.Center=28
                        LeftMotor.InvertOut = 0
                        LeftMotor.Operate = 1
                        Green.Value=1
                        Red.Value=1
                        Yellow.Value=1
                Next Dummy
                RightMotor.Operate=0
                LeftMotor.Operate=0
        End If

        If LH.Value =1 Then
                Green.Value=0
                Red.Value=0
                Yellow.Value=0
                For Dummy =0 to 10 step 1
                        RightMotor.Center=28
                        RightMotor.Value=-127
                        'RightMotor.InvertOut=0
                        RightMotor.Operate = 1
                        LeftMotor.Center=28
                        LeftMotor.InvertOut = 1
```

```
                    LeftMotor.Operate = 1
                Next Dummy
                RightMotor.Operate=0
                LeftMotor.Operate=0
            End If

            If FH.Value=1 Then
                Green.Value=0
                Red.Value=0
                Yellow.Value=0
            End If

        OOPic.Delay=20

        Next Counter2

        Red.Value=0
        Green.Value=0
        Yellow.Value=0



End Sub
```