**University of Florida**
**Department of Electrical Engineering**
**EEL 5666**
**Intelligent Machines Design Laboratory**

# Autonomous Capture the Flag

Steven Farago
Robot: Khan

Brian Moyer
Robot: Moyer's Destroyer

Instructor: Dr. Keith L. Doty

December 12, 1997

# TABLE OF CONTENTS



* Note – BM authored all sections labeled Moyer's Destroyer.
SF authored all sections labeled Khan

# Abstract

This paper details the design of two robots whose purpose is to play *Capture the Flag. Capture the Flag* is a game where two rival player attempt to steal the enemy player's flag and return it to their respective home bases while still defending that base. The physical design of each of the robots is discussed. In addition, the unique problems encountered while trying to create viable game-playing robots and the methods used to overcome them are described. Specifically, the problems of physically acquiring a flag, approaching a flag, distinguishing friendly and enemy bases, enemy disable methods, and appropriate game behaviors are addressed. Finally, the project's successes and limitations are listed and explained.

EXECUTIVE SUMMARY

*Capture the Flag* is a game where two opposing players attempt to steal one another's flags and return to their respective home bases first. The robots Khan and Moyer's Destroyer have been designed to play an autonomous version of this game against one another.

The two robots have each been equipped with a number of different sensor systems in order to allow them to play the game successfully. Both use bump switches and hacked 40kHz Sharp IR receiver/IR LED pairs to implement collision sensing and avoidance (as well as IR following in some cases). The robots are also each equipped with an array of standard 32kHz Sharp IR detectors that, along with an LED, provide a general digital communication system. This communication is put to a restricted use as a disable mechanism (similar to a paintball gun in human *Capture the Flag).* Additionally, both robots are equipped with a polarized light detector based on polarized filters and CdS cells. This detector is used to provide a fair method of distinguishing friendly and enemy home bases (which are polarized light sources). Normal CdS cells are also employed by both robots for flag acquisition purposes.

The robots differ mainly in platform and motor type. Khan is a round, wooden robot based on an early model Talrik. It uses standard hacked servos purchased in a kit from NovaSoft for movement. Moyer's Destroyer is an overhauled RC tank. Special high current motor drivers were constructed to allow the tank to use its original high-power motors.

Both robots are controlled by an HC11 microprocessor expanded with the ME11 expansion board. The processors on both robots are driven by slightly different versions of the same code.

# INTRODUCTION

For each of the authors, the primary goal of the Intelligent Machines Design Laboratory (IMDL) is to realize a fully autonomous mobile robot capable of playing the game *Capture the Flag* against a robot opponent.

The rules of *Capture the Flag* are fairly simple. Each player has a home base with a flag. The object of the game is to steal the other player's flag while defending the home base. The first player to return to his home base with the enemy flag wins. Players typically have some way of disabling each other (such as tag, or paintball guns) in order to gain a temporary advantage as well.

For robots to successfully play this game, a number of behaviors and sensing mechanisms are necessary. Probably most important is the ability to find and take the enemy flag. This skill would require a mechanism for sensing and approaching a flag, a way to distinguish between friendly and enemy flags, and a method of physically grabbing the flag. Secondly, the robots should have some method of disabling each other in order to gain a temporary advantage. Finally, the robots would need some method to accomplish general navigation in order that they are able to roam the playing field in search of the flag.

This paper details the construction of two robots, Khan and Moyer's Destroyer, whose purpose it is to implement the above mechanisms and behaviors to play *Capture the Flag.* The paper is divided into 11 sections. The Integrated System section provides a high-level description of each of the robots. Detailed information about the physical basis of the robots is found in the Mobile Platform Section. The Actuation section describes methods of movement and flag manipulation employed by the two robots. Purpose, design, and experimental results for new sensors is detailed in the Sensors section. The Behaviors section describes in detail the code the makes up the behaviors of each of the robots. After this, experimental setup and analysis for

the sensors is described in the Experimental Layout section. The Conclusion section summarizes the successes and limitations of the project for each of the robots. The Documentation and Vendors sections provide a list of references and vendor addresses respectively. Finally, program code is listed in Appendix A and experiment data listed in Appendix B.

# INTEGRATED SYSTEM

## Khan

The brain of Khan is realized in the form of a Motorola HC11 EVBU board with the ME11 expansion produced by NovaSoft. The ME11 provides two motor drivers, an extra 32K of memory, and eight digital outputs that can be configured to modulate at 40kHz.

Khan uses all these resources to control his various systems. The motor drivers are used to drive two hacked servos located in the middle of his body, a circular structure based on an older Talrik design with a tower constructed of Legos. The ME11 digital outputs are configured for 40kHz modulation, and are used to drive the four LEDs mounted on the bottom of Khan's front end. Analog ports provided by the HC11 are used to read Sharp IR receiver and bump switch information. Unfortunately, the analog ports provided by the HC11 are not sufficient to interface to all of Khan's systems. Therefore, the analog and digital ports have been expanded to an additional eight inputs and outputs respectively. These additional inputs are used obtain information from the normal and polarized light sensing CdS cell configurations and the flag detector located on the tower of the robot. Also, the available output compares provided by the HC11 are used to respectively control and drive the receive and send portions of a digital communication system as well as a piezoelectric speaker.

## Moyer's Destroyer

Moyer's destroyer uses the HC6811 microprocessor to take in and distribute various data to the different sensors. This robot also contains the ME11 expansion board and an extra 32k of RAM. Although the ME11 board contains two motor servos, they are not strong enough to support the tank's built in motors. Therefore, a more rugged system was developed to drive the

motors. The motors drive two tracks that are made of rubber and allow the robot to move on rough surfaces if desired. Moyer's Destroyer has a separate battery pack for the motors since they are very power draining. Moyer's Destroyer uses polarized film and CdS cells to determine which flag the robot is looking at. The tank uses the end of its turret to retrieve the flag. The robot is able to determine whether it has retrieved the flag by using another CdS cell on the retraction mechanism. Moyer's Destroyer was able to disable the opposing robot by transmitting an infrared coded signal.

# MOBILE PLATFORM

## Khan

Khan's body is based on an older model Talrik frame. The frame is constructed of model aircraft quality wood shaped into a 9-inch circle. Approximately 3.5-inch long wheel wells are located in the center of the body, each 1-inch from the edge of the circle. Additionally, Khan has a two inch diameter circular access hole slightly displaced from the center of the body.

Khan is supported by two 3-inch diameter wheels located in the wheel wells and a castor affixed to the rear underside of the body. Because the castor was rather large compared to the body's vertical displacement from the ground (1.75 inches), it was necessary to cut a hole where the castor attached to the body and affix the castor to a small square of wood above attached to this hole with wood screws. Without this modification, Khan tended to tip forward. In addition, a later modification involved gluing a sponge to the front bottom of the body in order to cushion shocks caused by sudden changes in direction.

Unlike the traditional Talrik design, Khan's tower is constructed entirely of Legos. Base legos were screwed into the body at the ends of each wheel well to support this tower. This Lego tower supports all the normal and polarized light sensors (front tower) as well as the entire disable and sound systems (back tower). Additional sensors on the main body include bump detection switches, which surround the base, and hacked IR receivers, which are located at the front-right and front-left sides of the body.

## Moyer's Destroyer

Moyer's Destroyer is based from an old RC tank model. It has a width of 7 inches, a length of 14 inches, and a height of 8 inches. In addition, the turret protrudes an additional 4

inches. The turret has been sealed to the forward position to avoid complications when retrieving the flag. The inside of the tank has been completely stripped out (with the exception of the motors) to make room for the HC11 and two 8-pack AA battery cartridges. The front of the tank has three infrared emitters devoted to collision avoidance, and a fourth emitter that transmits the disable code. The front also has three infrared detectors that are used for collision avoidance. Three CdS cells are also placed in the front to find a light source. Two polarized CdS cells are placed close to one another to determine which flag station is being observed. The end of the turret contains a magnet with a CdS cell in the middle that is used for retrieving the flag and acknowledging capture. The front and back of Moyer's Destroyer contains a bumper with bump switches in the event that a collision occurs. The back also has a single infrared emitter and detector to be used for collision avoidance. The top of the tank has four infrared detectors that receive the disable code from the enemy.

# Actuation

## Khan

Khan's wheels are driven by two hacked servos included in the servo kit purchased from NovaSoft. These servos are in turn powered by the motor driver circuits included on the NovaSoft ME11 expansion board and controlled by MIL motor libraries included with ICC11. During the course of Khan's construction, it was found that these libraries have a bug. For some reason I have been unable to fix (or even fathom), one motor cannot be commanded to change direction when the other motor is set to off.

While it does not technically move, the flag grabber mechanism may be though of as actuation as well. The grabber is simply a magnet protruding from the front of each robot along with some sensing mechanism (an open switch closed by a electrically conductive flag in Khan's case). This magnet will usually pick up a magnetic flag even when the two are initially slightly off center.

## Moyer's Destroyer

The robot I am designing is being assembled from an old radio-controlled tank. The motor has already been found to work well, except that each draws up to 1.5 A. Therefore, I must come up with a separate set of motor drivers than what is already in place on the ME11 board. The ME11 board has been built and preliminary tests indicate that it is fully functional. The tank also has a functional motor for its turret that I hope to find a practical use for. The tank will also have a cliff detector, in the event that it is being operated near stairs. The robot will most likely have two separate power supplies since the tank motor requires around twelve volts and the rest of the system uses a regulated 5 volts.

## Sensors

Both Khan and Moyer's destroyer have the "typical" complement of bump switches and IR LEDs/Sharp 40kHz receivers that allow them to implement collision sensing and obstacle avoidance. The bump switches are wired with a standard voltage divider network. A bump on the bumper causes the switch to close and the signal to change from +5V to ground. Similarly, the Sharp receiver was hacked according to the class handout to provide analog measurement capabilities.

Due to the unique nature of our project, we found it necessary to design additional sensor systems in order to better simulate a true *Capture the Flag* game. For example, to simulate a tag in capture the flag, we found it necessary to develop a digital communication system. To distinguish a friendly base from an enemy base, it was important to design a reliable polarized light detector. Some kind of flag detector also needed to be developed so that the robots could know when a flag had been successfully stolen.

### IR communication system

*Hardware*

The IR communication system used by both robots consists of three parts: a Sharp 32KHz IR receiver (unhacked) to receive the digital signals, a IR LED wrapped in half an inch of black heat-shrink to broadcast digital messages, and a timer circuit based on a 555 timer chip to provide modulation necessary for the Sharp IR receiver to correctly function.

IR Receiver. Whether or not to perform the analog hack on the IR receiver caused a bit of consternation when we began to design this system. When unhacked, the IR receiver produces unexpected behavior in the presence of a steady source of modulated IR. That is, aside from a response to the transition from no light to light (when the IR source is first activated), the

unhacked IR receiver will return a “no signal” reading regardless of whether the source is active or not. Also, if the IR source is held too close to the Sharp, this “no signal” reading becomes erratic with short pulses occurring randomly.

Given this behavior, we were initially at a loss as to how to use the unhacked Sharp to build a reliable communication system. However, after some experimentation (and help from lab TAs and the course instructor) we discovered that the Sharp would respond predictably as long as the signal that it received was changed rapidly enough. Based on experimentation, a 300 baud communication rate provides a signal that changes rapidly “enough” to provide a reliable system. So, we define 300 baud to be our system’s minimum communication rate.

IR Emitter. The hardware portion of the emitter system is simple. Its consists of an IR LED wrapped in a half inch length of heat shrink (to provide columniation) that is driven by the output of a 7408 AND gate. Inputs to the AND gate are an Output Compare pin from the HC11 (which generates the digital signal to be transmitted) and the output of the 555 timer circuit (which provides modulation necessary for correct operation of the Sharp IR receiver).

Timer Circuit. The timer circuit was obtained from one of the lab manuals. It is configured to provide a continuous 25KHz signal. This signal is used to modulate the IR signal that is broadcast to other robots.

*Software*

The software that controls these hardware elements is the heart of the communication system. The controller software consists of two separate modules, and transmitter module and receiver module, along with some utility functions to simplify use of the modules for the programmer.

Transmission system. The transmission system uses one of the Output Compares on the HC11 to provide the timing necessary to send an intelligible signal. The interrupt service

routine (ISR) that controls the Output Compare (OC) function is set to trigger an interrupt once every bit period, where a bit period is defined as the pulse width of a single bit that is required to achieve a specified baud rate.

After each interrupt, at the end of a bit period, a global array containing the message to be sent is examined. If the next element in the array is a 1, then the OC pin is configured to become high on the next compare. Otherwise, the OC pin is configured to be reset on the next compare. Following this operation, the global variable that indexes the message array is incremented (and reset to zero if the increment operation causes the variable to exceed the message size) and the ISR returns control of the processor to whatever function it interrupted.

The transmission system is controlled with the OC interrupt mask. Setting the mask bit of the system's OC will activate the system, and it will begin transmitting whatever code is in the global message array until the mask bit is cleared.

Receive System. The receive system uses another of the HC11's output compares along with a generic digital input to receive signals. Again, the OC triggers interrupts once every bit period. At each interrupt, the signal from the Sharp is sampled. What action is taken next depends on previous events.

If the system is not currently in the process of detecting a code (which can be determined by the value of a global *in_progress* variable), then one of two things can happen. If the signal from the Sharp is a 0, the system assumes than no code is being transmitted and no action is taken. If the signal sampled from the Sharp is a 1, the receive system must "synch up" with the sending system (see Fig 2).

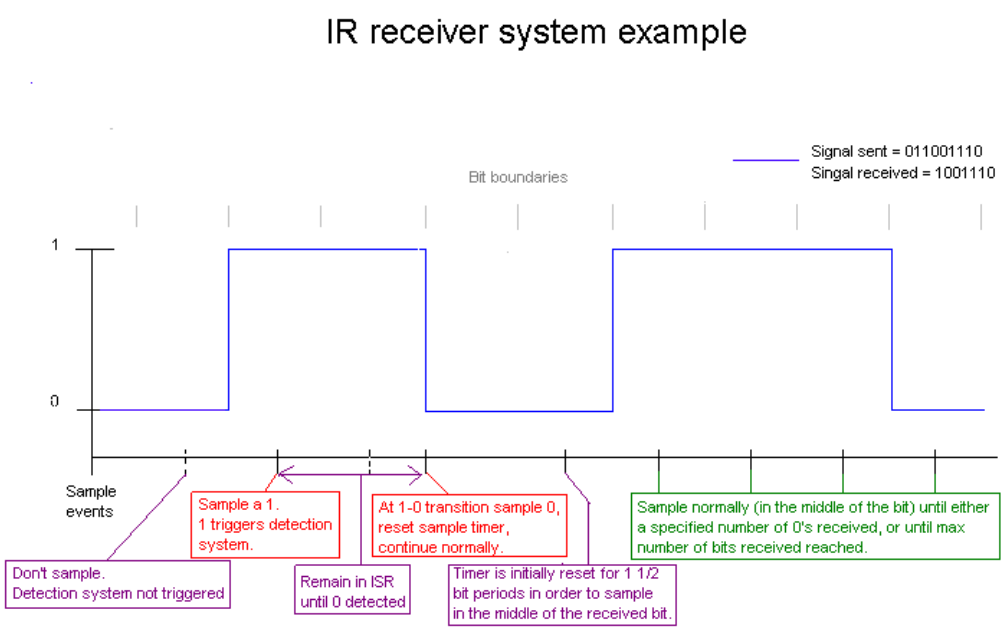



Figure 1

Synching up involves synchronizing the timer of the receive system with that of the remote sending system. To do this, the ISR takes control of the processor, continually sampling the Sharp for as long as a 1 is detected. As soon as a transition from 1 to 0 is detected, the ISR sets the OC system to interrupt one and one half bit times later, adds a 0 to the received code, sets the *in_progress* variable, and returns control of the processor to the interrupted function.

Configuring the OC to initially interrupt one and one half of a bit period later (as opposed to the standard interrupt one bit period later for other situations) causes all samples to occur in the middle of subsequent bit period. Sampling in the middle of the bit period tends to minimize bit errors associated with slightly out of synch sending and receiving E-clocks.

Once a code has been detected, subsequent interrupts involve simply sampling the Sharp, placing the sample in a global receive array, incrementing an indexing variable for this receive

array, and determining if the code has ended. The system can be configured to consider a code to be terminated in two different ways. A code can be considered to be over after either a specified number of successive 0's have been received, or after a specified number of bits have been received. Upon detection of the end of a code (whatever the method), a global *code_received* flag is set. This flag is reset only when some external function examines the received code.

Because the main purpose of the receive system is to implement some kind of disable system between the two robots involved in the game, it must be scanning for a disable code at all times. Thus, the disable system is constantly active and should not be disabled once started.

Utility Functions. The communication system also includes various utility functions whose purpose is to make using the system easier and to shield the programmer from directly manipulating the interrupt system.

One of these functions is the initialization function for both send and receive systems. These routines set various interrupt masks and vectors, zero appropriate global variables, and otherwise prepare the send and receive systems for use upon startup.

Also, the send system includes activation and deactivation functions. The activation function allows the programmer to send a message as a character string (e.g. send("11001010")), and not concern himself with setting any of the send system's global variables or interrupt masks. Similarly, the deactivation function allows the programmer to appropriately reset all global variables and disable certain interrupts with a simple function call.

Functions that check the status of each of the system are also included in this utility library. They allow the programmer to check the status of a code (i.e. whether it was sent correctly or whether a new code has been received) and change global flags appropriately.

*System Problems or Limitations*

Mid-code Detection. One of the problems inherent in the synchronization process of the receiver system is mid-code detection. Because the receiver system uses such a simple triggering mechanism (sampling a 1) for detecting the start of a new, it is possible that the receiver may mistake a 1 in the middle of a sent code as the beginning of a new code. However, this is only a minor problem. Starting all codes with a long string of 1's would significantly lower the probability of mid-code detection. Alternately, sending only fixed length codes and sampling twice that length would ensure that all of the code was detected (though its start may not be the beginning of the received code).

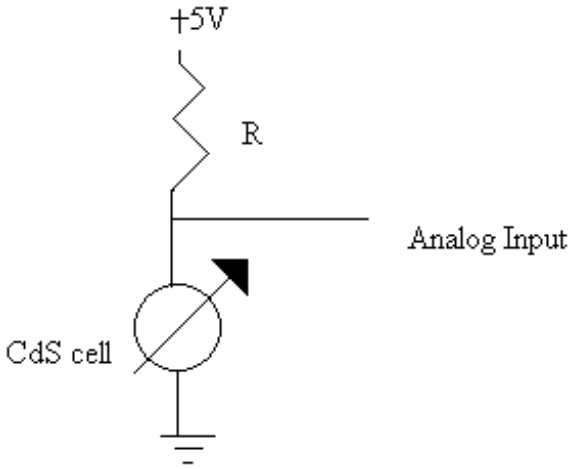
Minimum Baud Rate. Due to the strange operation of the unhacked Sharp IR receivers (i.e. erratic or unresponsive behavior in the presence of steady or slowly changing IR light), there is likely some minimum baud rate below which the receiver will fail to respond predictably. The lowest baud rate our system has been tested is 300 baud, for which there were no unexpected problems. A simple solution for this potential problem is to declare that no communication be done at a rate slower than 300 baud.

Obstacle IR interference. Based on our initial understanding of the workings of the Sharp IR receivers, we assumed that the 32KHz Sharps would not respond to 40KHz modulated type and vice versa. However based on our first experiment, it appears that there is a wide frequency range to which the 32KHz sharp will respond. Furthermore, based on our fourth experiment where we measured bit error rates in the presence of steady 40KHz IR light, we can say that our communication system (configured with only one Sharp IR receiver) is hopelessly unreliable when active obstacle avoidance systems are active on the broadcasting robot. A solution to this problem is to have the broadcasting robot turn off all IR sources before broadcasting a message.

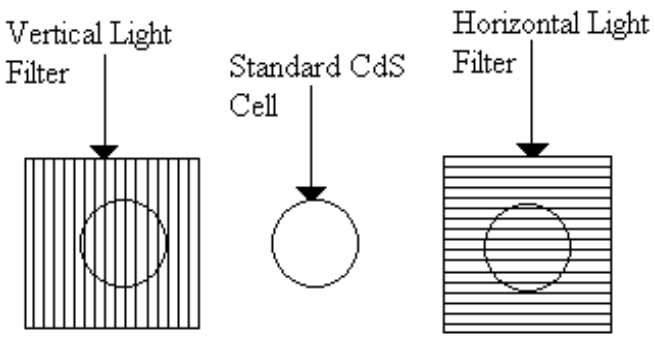
Systematic Bit Errors. During experimentation, we noticed that some bit errors would always occur regardless of the value of other variables (i.e. range, horizontal displacement, modulation frequency). These errors always seemed to occur at least 1000 bits into the message. Because our messages will likely be less than 100 bits, this problem is not a very great concern and we spent little time attempting to find a cause or solution. However because synchronization between sender and receiver occurs only once (at the beginning of the received message), we theorize that these errors may be caused by slightly disparate E-clock frequencies between the sending and receiving HC11s. If this is true, then periodic resynchronizations between sender and receiver should solve the problem. Should time permit, it would be interesting to see if some kind of alternate, periodic synchronization scheme would solve the problem.

## Flag Detection Sensor

The detection and capture of the flag is primarily being accomplished through the use of CdS cells. CdS cells vary their resistance with the amount of light that they "view" from the source. If the source is dark, there will be a high resistance. If the source is bright there will be a low resistance. Therefore, the CdS cell is nothing more than a voltage divider with a variable resistor (see Figure below). The brighter the source, the lower the voltage being dropped across the CdS cell will be.

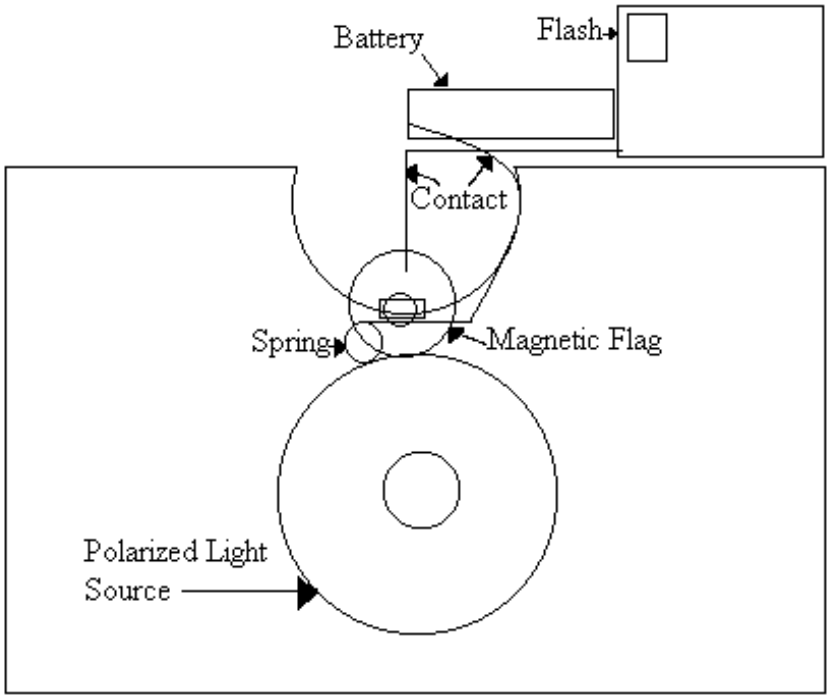



Although the use of a CdS cell as a voltage divider may seem trivial, the application of detecting the proper flag requires more thought. The robot must be able to decide whether it is looking at its own flag or the flag it is trying to capture. In order to do this with CdS cells, the robot will use polarized filters to determine the difference. When something is polarized, it causes light or radiation waves to oscillate in a definite way. Therefore, we will set a filtered light source at each flag station to serve as a beacon for the robots to locate. One of the flag stations will be emitting vertically polarized light while the other one will be emitting horizontally polarized light. The robot will be equipped with three CdS cells as described in the diagram below:




Three CdS cells are necessary in order to correctly identify the proper flag station. The standard CdS cell will be capable of picking up all light and will be used to determine whether the robot is viewing a light source or not. The polarized CdS cells will then be used to compare with one another to determine which flag source is being viewed. For example, suppose that the robot is searching for the flag station that is emitting vertically polarized light. Once the standard CdS cell determines that a light source is being viewed, the two filtered cells will be compared. If the robot is indeed looking at the vertically polarized flag station, its horizontally filtered cell should be returning a very low analog signal. Meanwhile, the vertically filtered cell should be returning a high analog signal.

Once the flag has been detected, the robot must be capable of retrieving it. This will be accomplished through the use of a magnet on the end of a probe that protrudes out from the robot. This probe will have another CdS cell in the center of it that will determine whether the flag has been successfully retrieved. The flag itself will consist of a similar magnet that will be situated so that the two magnets will attract. The flag station has been designed (see depiction below) in such a way that the flag should be capable of being retrieved with little or no contact.




After much thought, the flag station was designed primarily through the use of mechanical parts. The flash mechanism was hacked from a disposable Kodak camera. Once the capacitor is charged, the light is triggered by contacting a wire to ground. The size of the capacitor indicates that a very large amount of current is being shorted to ground at the time of the flash. This factor limited us to the use of mechanical parts to trigger the flash. Therefore, we used a small spring from the disposable camera to serve as the contact. A grounded wire was suspended in air above that flag in such a way that there would be contact if the flag was removed. In addition, the "flag pole" is made out of a piece of wood to avoid an inadvertent contact of the flash. Finally, a paperclip is used to hold the flag in place on the station until retrieved. Early tests have indicated that the flag can actually be released before making contact between the two magnets. The flash mechanism described previously has been implemented as a way of letting the "players" know whether a flag has been captured or not. The robot that has captured the flag should know whether or not they have captured the flag or not, but this will

allow both players to know. This phase is similar to a teammate declaring that your own flag has been captured. This gives the robots a chance to change their behaviors if they choose to. A CdS cell will also be placed facing behind the robot for the sole purpose of detecting the capture of its own flag.

## Flag detectors

*Khan*

Like the bump switches, Khan's flag grabber/detector is based on a pull down switch arrangement. The system is simply a magnet covered with electrically conductive material (aluminum foil) on either side to form a switch. One side of the conductive material is grounded while the other is connected to power through a resistor. The signal is taken between the resistor and the conductive material. When Khan hits a flag (another magnet completely covered with conductive material) with this grabber, the switch is closed, the signal goes low, and the flag is successfully detected.

*Moyer's Destroyer*

Moyer's Destroyer's grabber/detector system is based on a CDS cell in a voltage divider network. A CDS is grounded through a resistor on one side while the other is connected to power. A signal is taken between the resistor and cell. When the robot grabs a flag, the CDS cell is covered, allowing no light to strike its surface. In theory, this causes the signal to drop drastically, signaling a successful flag capture.

# Behaviors

## Overview

Behaviors governing the control of both robots are implemented in a two-tiered structure. At the lowest level is each behavior submodule. The individual submodule actually issues command requests based on sensory input. Above these submodules is the overall controller. The controller controls and defines the interaction of the various submodules through a priority system based on current or past sensory information (saved in a global variable). Each set of interaction definitions constitutes a different behavior.

## Controller structure

The controller is a set of various active priority values for each of the submodules. Each set of these priority values defines which submodule commands will dominate (when the submodule becomes active) the robot's behavior, and thus defines a behavior. For example, a priority set where bump detection dominates light-seeking, which in turn dominates IR avoidance would define a "Find Flag" behavior in our game, as the robot would seek light, stopping only if it physically bumps into something (even if it senses an obstacle in the path of the light). A behavior/priority set is selected based on current or past sensor readings.

The controller for both robots defines five separate behaviors:

1. ROAM ACTIVE: The robot randomly roams around the playing field with all IR LEDs and collision/IR detection systems active. In the presence of no obstacles (or other IR sources), the robot will seek any sufficiently bright light that it senses. The robot defaults to either this behavior or the ROAM PASSIVE behavior based on the value of a timer.

2. ROAM PASSIVE: Again the robot randomly roams the playing field. However, here all IR LEDs are deactivated, and collision avoidance (i.e. bump detection) is active. Instead of avoiding IR sources, the robot will follow any IR that it senses. If no IR light is sensed, the robot will follow any visible light it detects.
3. SEEK MY FLAG: All IR based systems (both avoidance and following) are deactivated. The robot will seek any light that it senses, stopping only if it detects a collision. This behavior only becomes active if a particular direction of polarized light (indicating an enemy flag station) is detected and the robot has not already stolen the enemy flag.
4. SEEK MY HOME: This behavior is actually no different than the SEEK MY FLAG behavior. However, as it becomes active under a different set of conditions, it is defined as a different behavior for organizational purposes. The robot will only seek its home when it has stolen the enemy flag and detects the particular polarized light associated with its home base.
5. WRONG WAY: This behavior causes the robot to back up and turn around regardless of other conditions. It becomes active when the robot has just stolen an enemy flag and needs to depart from the flag station.

In addition to these behaviors, there are two more whose priority levels are fixed throughout the game and thus are not included as a part of the controller. They are:

1. DISABLE: Upon successful detection of a disable code, the robot is forced to stop all movement-related behavior for a random period of time. This period of time is defined as the time is takes to play one of three random songs commonly associated

with waiting or sleeping. The priority of this behavior is the highest possible, so it will always dominate any others.

2. GAME OVER: When this behavior becomes active, the robot will stop and play a victory song (the Gator fight song), indicating that the game is over. This behavior is triggered when the robot bumps into its home base carrying the enemy flag. The priority of this behavior is dominated only by the DISABLE behavior.
3. SHOOT DISABLE GUN: As this behavior does not affect the motors, it does not use the priority system. Instead, it fires the disable gun for a short time every fifteen seconds. While the robot technically cannot both fire the gun and perform other behaviors at the same time, the gun is fired for so short a time that it appears that the robot can both shoot and perform other behaviors concurrently

## Submodule description.

The construction of each submodule is based upon a standard template. The purpose of this standardization is to allow all submodules to be controlled by the master controller and to interact correctly, as well as allowing easy incorporation of new submodules into the overall behavior structure. The heart of this template is the behavior array.

The behavior array is a two-index integer array that contains information about the various low-level procedures. Each submodule is given a unique index in this behavior array. This behavior array holds information on submodule priority, suggestions for motor settings, and deferred operation info (i.e. last operation or next operation) provided by each of the procedures. Overall robot operations are based on the information stored at the array index with the highest priority. In other words, the commands issued by the procedure with the highest priority "win" and are eventually preformed by the robot.

The process for setting its priority is also something each process has in common. In all submodules, the priority is initially set to zero. When the module is run, a condition is tested. If the condition is found to be true, the process executes algorithms to provide motor suggestions (entered into the behavior array) or execute other commands (like playing music), and then sets the submodule priority to a value specified by the controller (called the active priority value). A false condition results in a submodule setting its priority to zero prior to returning (without executing any other code). This essentially causes all motor suggestions from the submodule to be ignored.

As an example consider a simple IR obstacle avoidance process. Initially, the process would check if any of the IR receivers has a reading above a certain threshold. If not, IR avoidance priority would be set to zero and all motor commands from the process would be ignored. If so, it would write appropriate motor commands to the behavior array (to make the motors turn one way or the other, depending on which IR receiver sees light) and set the IR avoidance priority to whatever value the controller currently specified. Then if the IR avoidance priority was the highest among all submodules, the motor commands it issued would be executed, and the robot would turn.

So, each of the submodules is a separate procedure that is executed continually by the main program. The behavior submodules each provide a "suggestion" as to what they think the next robot action should be (via the behavior array) on each iteration. The suggestion that is actually converted into a command depends on what submodule has the highest current priority (defined by both the controller and whether the submodule has been triggered to activity by some external stimuli).

The following submodules are directly involved in the robot's behavior:

1. Check bumpers: This function implements an operation that is standard on almost all robots. However, because of constraints imposed by the use of the generic submodule template, this function is actually among the most complex of all submodules. It is triggered by detection of a bump on any of the back, right, or left sides of a robot. When such a bump is detected, the direction is recorded, the bump priority is set to its active level, and a deferred operation counter (a part of the behavior array) is set to three. After this, the function returns. When it is next called, the nonzero deferred operation counter triggers the function. The deferred operation counter is decremented, and the routine tells the robot to either back up or go forward, sets a timer and returns. Until this timer expires, subsequent calls to the bump routine result in returns with no changes to the behavior. When the timer finally expires, the next time the routine is called it will decrement the deferred operation counter and set the robot to go right or left (depending on the direction of the initial bump) for a specified length of time (implemented with timers in the same manner the pervious timer operation was). Finally, when the second timer expires, the robot resets all deferred operation and priority behavior variables to zero and returns.
2. IR avoidance: This function also implements an operation fundamental to most MIL robots. The function is triggered to activity by a reading on any of the IR detectors exceeding some threshold value. If this is true, the function sets its priority to the active level and writes suggestions to the behavior array for the robot to go either right or left (the speed and direction is determined by which of the IR receivers was above the threshold value). The function resets its priority variable when none of the IR receivers produces readings above the threshold value.

3. IR following: The function allows the robot to track and follow any IR sources (such as an enemy robot with its IR LEDs on). Like the avoidance algorithm, this function is triggered to activity by a reading exceeding a threshold level on any of the IR receivers. If this is true for either of the two forward facing receivers, the robot is urged (with writes to the behavior array) to go forward. If this is true for either of the outward facing receivers (and not true for the forward receivers), the robot is urged to turn either right or left (depending on which receiver's value exceeds the threshold). Note: Because of the placement and number of IR receivers on each of the two robots, this function is only implement in Khan.
4. Light following (seek flag): On the surface this function is implemented in a similar manner to the IR following function. However, much more precision was needed in order to successfully acquire a flag. Readings exceeding the threshold value from any of the three CDS cells triggers the function to activity. If the middle value exceeds the threshold the robot is urged to slowly go forward, regardless of the readings from the other cells. Otherwise, the robot is told to slowly go either left or right (depending on CDS cell's reading exceeds the threshold)
5. Braitenburg avoidance: The purpose of this function is to avoid the Braitenburg (spelled Bradenburg in the code) trap. It does so by keeping track of the number of times the robot changes direction in a specified time interval. The function is triggered to activity by a 32-bit timer. When the timer expires, the function examines the number of times the robot has changed direction since the last check (a separate function keeps track of this turn count). If it exceeds some value, the function becomes active, setting its deferred operation counter to one and ordering the robot to

turn approximately 180 degrees (this turn is implemented with timers in the same way the turn in the check bump function is implemented). When the turn is complete, the function resets the turn counter, deferred operation, and priority variables, and sets the timer to trigger it again in another fixed time interval.

6. Fixed turn: This function allows the robot to perform a fixed angle turn after backing up. Its code is a simplified version of the check bumpers code. However, it is triggered by an internal global variable (instead of a sensor reading) that is reset upon completion of the operation.

In addition to these behavior submodules, there are additional motor control functions continually called in the main program. One of these functions (set_motors) sets the motors to a target value and counts the number of times the robot attempts to change direction (defined as the number of times the robot sets the motors to turn left when the last command was to turn right and vice versa). The other (update_motors) uses the target values set by the set_motors function along with variables containing the current values of the motors to achieve smoother motor operation.

## Lessons Learned

Probably the most valuable lesson learned in the process of programming the robots was to not overestimate the abilities of the robots. Specifically, the seek-flag behavior caused a great deal of trouble for me. When I first considered the problem, I naively assumed that three CDS cells placed in some kind of configuration that allowed the outer two cells to distinguish light in right and left directions could easily guide the robot to a small flag given a simple following routine similar to the one I used for the IR following. However, once I began testing I realized it

was a much more difficult problem. To even achieve a semi-reliable mechanism for grabbing an enemy flag required much customizing of the CDS sensors (with respect to both placement on the robot and columnation method) as well as many different code structures. If I could start over, I would leave myself more time and try to complete programming and physical sensor development in parallel as much as possible (to allow me to develop the optimal CDS cell configuration/code structure combination).

The disable behavior also illustrates this point. Though I tested and developed it fairly early in the semester, once I got it working independently, I assumed that it would work as well once integrated into the robot. This turned out to be only partially true--while the disable system does work on the robot (i.e. it works when exposed to a IR remote control), the robots almost never are able to disable each other with it (due to aiming issues).

In addition, I would carefully consider the overall structure of my code before I started programming. While the general structure I imposed on each submodule provided organization and allowed easy integration and addition of new submodules, it also complicated many functions (like check_bump) that might have been simpler with another structure, and probably significantly lengthened the code (which, at approximately 31.5k, almost fills the me11 board, making further code development difficult).

## Experimental Layout and Results

### Experiment 1: IR Comm System -- Error Rate vs. Modulation Frequency

This experiment involved testing the accuracy of the detection system with varying modulation frequency of the emitter system. The detector and emitter were held at a fixed range from one another and a fixed digital code was transmitted at varying modulation frequencies. If any recognizable part of the code was received by the receive system, then that run was counted as a success. Otherwise the run was counted as a failure.

Data: The code was transmitted ten times at each frequency, and the number of successes charted (Appendix B, Fig 1). A quadratic curve was fitted to the data.

Conclusion: Fitting a parabolic curve to the data was only marginally successful ($R^2$=.6335). The data appears to have a sharp cutoff in successful transmission at approximately 10KHz and 90KHz. However, between these ranges the system seems to be fairly reliable.

### Experiment 2: IR Comm System -- Bit Error Rate vs. Range

This experiment attempted to test the range limits of the system. The detector and emitter were placed at varying ranges from each other. At each run a repeating code (10101010) was transmitted. The receiver sampled 1600 bits from this code. The results were saved and the number of bits in error counted (Note: location errors caused by errors in previous bits were not counted as new errors).

Data: This process was repeated four times at each range. Range was measured in nine-inch squares. Results were charted (Appendix B, Fig 2) and an exponential curve fitted to the data.

Conclusion: An exponential curve was fitted to this data with only marginal success ($R^2$=.5247). The bit error rate seems to rise extremely fast (i.e. faster than exponentially) at 9 squares distance. Based on these results, we can say our sensor has en effective range of approximately 81 inches.

## Experiment 3: IR Comm System -- Bit Error Rate vs. Horizontal Displacement

The purpose of this experiment was to determine the angular dispersion of the columnated IR signal. The IR receiver with the half-inch of heat shrink was placed at a fixed range from receiver. At first both were aligned along a vertical axis directly facing each other and the bit error rate was measured. For later experiments, the receiver was horizontally displaced from this vertical axis while still maintaining its orientation and fixed range. For each of these displacements, the bit error rate was measured and plotted against the angular displacement (calculated from the horizontal displacement and known range of the emitter from the receiver).

Data: There were 4 runs done at each angular displacement. See Appendix B, Fig 3 for results.

Conclusion: The data indicate a sharp cutoff in successful transmission at approximately 20 degrees. Based on this, we conclude that, given an otherwise aligned receiver and emitter, a message will be successfully transmitted only if the receiver is displaced less than 20 degrees from the emitter (holding all other factors, like range, constant).

## Experiment 4: IR Comm System -- Bit Error Rate with 40KHz obstacle avoidance IR activated.

This experiment's purpose was to measure the performance of the 32KHz based Sharp IR receiver system in the presence of steady, 40KhZ modulated IR (from obstacle avoidance system). The number of bit errors per 1600 bits was measured over four runs at a fixed distance.

Data: Bit errors / 1600 bits -- 617, 610, 587, 609, 612.

Conclusion: Based on these results, it appears the communication system is very sensitive to steady 40KHz IR.

## Experiment 5: Polarized Light System – Effect of Ambient Light on Polarized Light Detection

The purpose of this experiment was to determine the effect of ambient light on the detection of polarized light. A polarized light source was shined on the detector and the difference between the two CDS cells in the detector recorded. A randomized block experiment was performed with ranges (1-4 feet in 6 inch intervals) as blocks and ambient light levels (dark, medium, bright) as treatments. Higher magnitude differences between the two CDS cells that make up the detector are considered better (as they increase the probability that the detector is actually detecting polarized light as opposed to some inherent difference in the CDS cells/filters or normal light interference)

Data: Appendix B, Fig 4.

Conclusion: A nonparametric Friedman analysis was performed on the data. Results indicate (with a p-value of .0036) that at least one of the treatments produced different results from the others. A follow-up multiple comparisons test showed that the medium light level treatment tended to produce significantly higher results that the bright treatment (with a significance of .008). None of the other treatments differed significantly (with significance better than .05).

Based on these results, I can conclude that the polarized detector works best in medium ambient light.

# Conclusions

## Khan

While not always with perfect accuracy or repeatability, Khan is able to fulfill the most of the basic objectives set forth at the inception of the project. Khan can successfully detect and distinguish light polarized in two specific directions. This allows him to correctly identify friendly and enemy flag bases. With the normal CDS array Khan can slowly seek a flag station, and can usually home in precisely enough to initially grab the flag successfully. The grabber/flag detector can reliably detect a grabbed flag, allowing Khan to intelligently decide when to seek its home base and when to seek the enemy base. Furthermore, the disable system can detect a sent code, which makes a disable behavior for Khan a reality.

Unfortunately, while Khan does display behavior that fulfills most objectives, this behavior is not always completely reliable. For example, Khan behaves unpredictably when light sources other than the flag stations are present. While Khan can usually (at least half the time) maneuver well enough to cause the flag grabber to strike the flag, the grabber does not always hold the flag with enough force to dislodge it, and will not detect it (and thus not back up) until the flag is very securely connected to the grabber. Also, because of the precision required by the flag seeking module, I was unable to smooth Khan's movements very much. The resulting jerky movements have a tendency to reset the processor. Also, Khan has only been tested alone (though other robots have been simulated with IR sources). How Khan will behave with another robot present is unknown.

With another chance to begin this project, I would already have software in mind software when constructing my hardware design. Had I considered the constraints some of the algorithms would place on my robot, I would probably have designed my hardware differently.

Also, I would carefully consider the structure I planned to use for my code before actually starting to write it. The structure I used essentially evolved as new needs became apparent. While this structure had its advantages (generality, organization), it also had some drawbacks (more complex modules, size of code) that limited further development.

## Moyer's Destroyer

There were invaluable lessons learned in the process of building Moyer's Destroyer. This project gave me a better understanding of both digital design and the HC11. There were many things I would have like to have done better in this project but ran out of time. For instance, I wanted to make the robot's torrent capable of turning. While developing this idea, however, I found that the HC11 and circuitry was too bulky to allow the frame to seal properly. I had also hoped to use motion detectors on the robot to find the enemy. This would have allowed the disable gun to be more accurate. I was not able to do this though because I was not able to find an adequate heat source for the enemy's robot. Throughout this project I learned the importance of thinking things out and designing the product before constructing it. In addition, I learned how helpful it is to label and/or color code when possible. This class has something to spark an interest in everybody and for me it was the chance for some hands-on design work. The biggest reward was the final product: an autonomous robot that plays capture the flag.

# DOCUMENTATION

[1] Fred Martin, The 6.270 Robot Builder's Guide , MIT Media Lab, Cambridge, MA, 1992

[2] Joseph Jones & Anita Flynn, Mobile Robots: Inspiration to Implementation, A.K. Peters Publishers, Wellesley, MA, 1993.

[3] Motorola, Small Signal Transistors, FETs and Diodes Device Data, Motorola, 1994.

[4] Motorola, Motorola Semiconductor Master Selection Guide, Motorola, 1996.

# VENDORS

**Radio Shack**
3315 SW Archer Road
Gainesville, FL 32608
352-375-2426

**Novasoft**
8822 Trevillian Road
Richmond, VA 23235
804-272-5752

# Appendix A:  Program Code

# Appendix B:  Experimental Data