University of Florida

Department of Electrical Engineering

EEL 5666

Intelligent Machines Design Laboratory

Holly: An Autonomous Air Cushion Vehicle

Bruce K. Miller
Aamir Qaiyumi

Instructor: Dr. Keith L. Doty

9 December 1996

TABLE OF CONTENTS

Abstract (BKM) 3

Executive Summary (BKM) 4

Introduction (BKM) 5

Integrated System (BKM) 6

Mobile Platform (AQ) 7

Actuation (BKM) 9

Sensors (AQ) 19

Behaviors (BKM) 26

Experimental Layout and Results (AQ) 28

Conclusion (BKM) 44

Documentation (AQ) 46

Vendors (AQ) 47

Appendix A: Program Code (BKM) 48

Appendix B: Movement Control Test Code (AQ) 55

Appendix C: Discrete Component Specifications (AQ) 64

ABSTRACT

Following is a discussion of the development of an obstacle avoidance system for an autonomous hovercraft using infrared emitters and sensors. The hovercraft reacts when a sensor indicates the presence of an obstacle in the vehicle's path. Sensor readings elicit programmed responses through the ICC11 programming language. Upon location of an obstacle, a microprocessor initiates a collision avoidance algorithm which alters the spinning of two rear-mounted propulsion fans, causing the hovercraft to veer away from the object. Various behavioral responses emerge as a result of the interaction of components within the integrated system. Several maneuverability problems unique to air cushion vehicles are presented in accompaniment with corresponding software-based corrective measures. A sonar system supplements infrared sensory information with rangefinding data and directional data. The closing discussion addresses future innovations and additions which appear likely as a result of the present course of experimentation.

EXECUTIVE SUMMARY

Holly is an autonomous hovercraft which is designed to navigate large areas at high speeds. An efficient centrifugal lift fan provides an air cushion which creates a near-frictionless surface over which Holly can maneuver. Two propulsion fans, driven by a high power motor driver circuit, supply speed and directional control to the hovercraft. Vehicle weight is minimal in order to reduce power consumption and allow for reasonable run time.

Holly's principle navigational system is long range IR. The hovercraft travels at high speeds and has little friction to facilitate braking and must therefore see upcoming obstacles as soon as possible in order avoid them in time. The integrated short range and long range IR system provides sensory information from 0.25 to 10 meters ahead without resorting to new, specialized, or expensive components. The long range IR system also provides fairly accurate range information.

A microprocessor provides a means of control for the integrated system including lift fan, motor drivers and propulsion fans, long and short range IR emission, and IR sensing. ICC11 programmed functions permit Holly to speed up, slow down, turn, and rotate in a controlled fashion despite the inherent difficulties in navigation of an autonomous air cushion vehicle. A main program provides behavioral guidelines which determine Holly's reaction to the environment by changing the speed and/or direction of the propulsion fans according to the control functions. The microprocessor also elicits specific behavioral reactions, such as "search" and "attack", to certain combinations of external stimuli.

INTRODUCTION

Our primary goal upon completion of the Intelligent Machines Design Laboratory (IMDL) was the realization of a fully autonomous mobile hovercraft, hereafter referred to as Holly. Physical design is of utmost importance because of the delicate balance between vehicle weight, power considerations, and air-channeling efficiency. Holly's central behavior is obstacle avoidance at high speeds. We have therefore developed a long range infrared system in order to avoid obstacles in time when moving quickly. The resulting autonomous hovercraft is the platform upon which additional, more goal-oriented, behaviors are implemented.

Following is a description of Holly separated into nine sections. *Integrated System* characterizes the relationship between the individual component systems of which Holly is composed. *Mobile Platform* describes the physical structure of the hovercraft and the mechanical systems used for motion. *Actuation* covers the supply of power as well as the basic propulsion and lift motor controls. *Sensors* denotes the types of sensors used and their relation to Holly's behaviors. *Behaviors* examines each behavior in detail. *Experimental* Layout and Results discusses the development processes involved for various components of the integrated system. *Conclusion* summarizes the successes and limitations of all work performed as well as future modifications under consideration. *Documentation* provides complete list of references, including data books for circuit components and a paper on a sonar system which was designed in parallel with this project. Appendix A contains the program code used to implement Holly's behaviors, and Appendix B holds detailed component specifications.

INTEGRATED SYSTEM

Holly is a sum of multiple system inputs which result in the emergence of a set of particular behaviors. The physical basis for a hovercraft is given in Mantle [1]. The central operational principle is the so-called "lift" which is provided by a motor-fan combination which takes in air through a portal of relatively large area and channels it down through the craft and out the bottom through openings of small area. The resulting compression of air forces the hovercraft to rise a small distance above the surface over which it travels. Often a *skirt* is attached to the base of an air cushion vehicle in order to maximize lift efficiency. Multiple propulsion techniques exist; we use a pair of rear-mounted propeller fans.

We use for control an ME11 expansion board, produced by Novasoft, which adds to a Motorola 68HC11 microprocessor an extra 32K of RAM, a pair of motor drivers, IR emitter modulation circuitry, and voltage regulation. Due to the high-current demand of the propulsion motors, however, the motor drivers were replaced with a more rugged system. The devices controlled by the microprocessor include short-range IR emitters, long-range IR (hereafter referred to as the IR cannon), IR sensors, one lift fan and two propulsion fans. The robot, once set in motion, senses the environment and responds based on programmed behavioral guidelines by which the microprocessor changes the speed and/or direction of the propulsion fans. The microprocessor program also elicits specific behavioral reactions to external stimuli.

MOBILE PLATFORM

Holly's original frame consisted of four parts as derived from Meyers/Wilkerson [2]. The load-bearing platform was disk-shaped with a hole in the center for a lift fan. Beneath the load-bearing platform, the outer shell for air flow was dome-shaped and the inner frame conical, to allow for a high volume of air intake and the necessary compression at the base of the craft. All frame components were shaped with plastic at the machine shop. We used a "skirt" formed with a bisected bicycle inner tube. The skirt allows the hovercraft to hover higher without any additional power requirement. Fig. 1 below depicts the basic model.

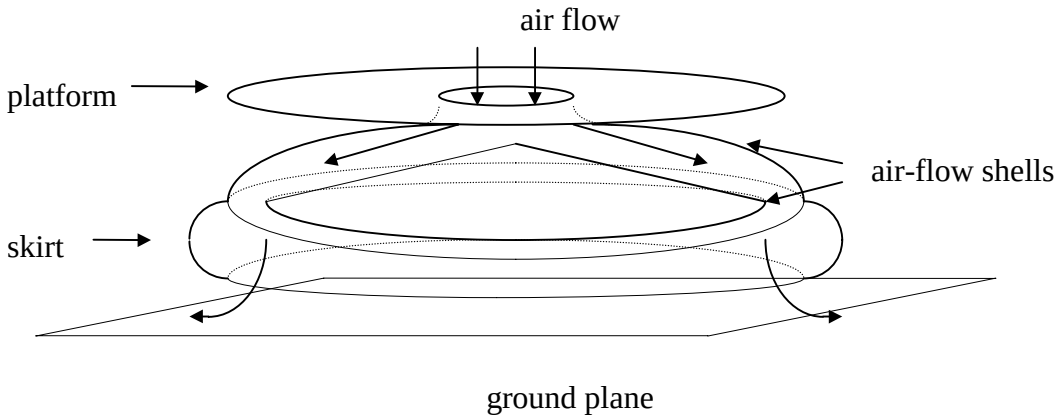

Fig. 1: Early Hovercraft Model

After experiencing problems with the weight/lift ratio of the initial hovercraft model, we investigated the availability of alternative hovercraft platforms. Based on the equations and qualitative information in [1], we determined that a craft with a centrifugal lift fan, skirt, and dual propulsion motors would best suit our purposes.

After obtaining numerous hovercraft components, we constructed Holly's current body with parts obtained from various hovercraft kits mounted on a Tyco R/C Typhoon II hovercraft frame. Operational dimensions, including a fully inflated skirt, are 16 inches long, 11 inches wide, and 7 inches tall. Principle of operation is essentially the same as that of the earlier model. The centrifugal fan is more efficient than its axial counterpart, as demonstrated by the cross-sectional diagram presented in Fig. 2. Air intake is from above; the centrifugal fan propels air directly to the sides, immediately filling the skirt and creating an air cushion. Air is then channeled through holes underneath the body and out through a small gap between the skirt and the surface plane; the small area for the air to escape in relation to the intake area produces a high-pressure area underneath and lifts the craft. The bi-directional propulsion fans are independent of each other and the lift fan.

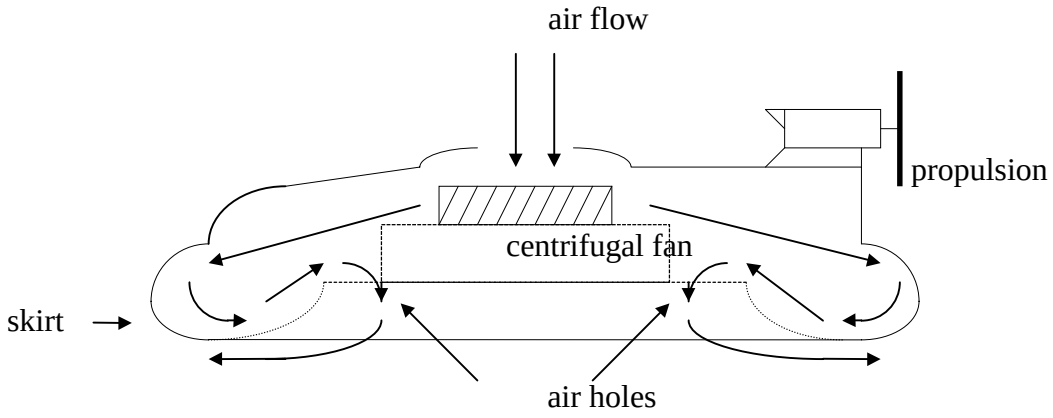

Fig. 2: Mechanical Structure of Holly

ACTUATION

Battery Power

Holly originally used two 8-cell, 9.6 V NiCd battery packs for power, one for the motors and the other for the microprocessor, sensors, and LEDs. Voltage was regulated down to 5 V for the 68HC11 processor and ICs through a MC78M05C voltage regulator. Estimated run time was 5-7 minutes. In order to extend the run-time, we replaced the NiCd battery pack with a Nickel-Metal Hydride battery pack. Nickel-Metal Hydride cells have a greater energy density than NiCd batteries, and with a 40% increase in cell capacity (measured in mAhrs) Holly's life expectancy on a fresh battery pack is extended to an estimated 9 minutes.

Motor Drivers

High Power Motors

Holly originally drove two rear propulsion motors with an SN754410 motor controller. A single heat-sinked SN74410 motor driver on the ME11 board can sustain a current of about 1 A. We determined through testing that Holly's propulsion fans draw a constant current of 1.75 A but with peaks that are much higher. The original solution was a heat-sinked, stacked arrangement of motor driver chips to decrease the current load and dissipate the excess heat. Both drivers burned out despite these precautions. Our next attempt was a relay-driven circuit similar to that in [1], which also failed because the relays could not handle the current. Since neither design was capable of sustaining the current pulled by the hovercraft motors, we developed a new motor control circuit using high power CMOS motor drivers.

H-Bridge Motor Driver

A typical H-bridge configuration for a motor driver circuit using two pnp transistors (A, D) and two npn transistors (B, C) is given in Fig. 3. This is the basis for our motor driver.

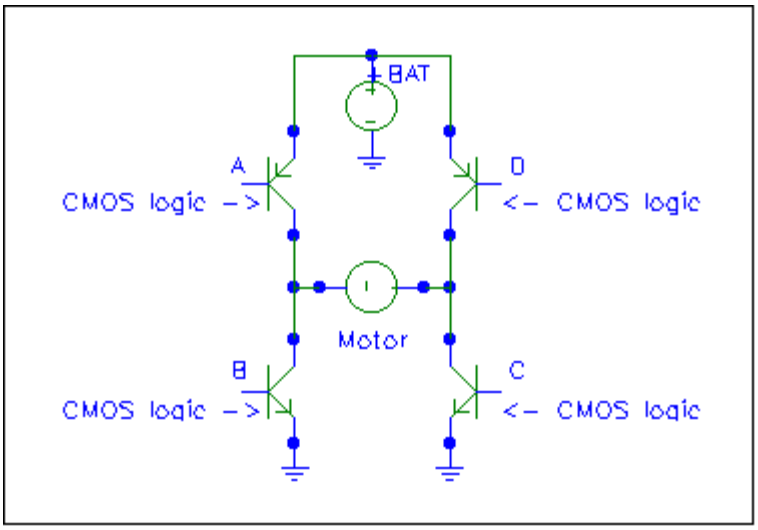

Fig. 3: Basic H-Bridge Motor Configuration

H-Bridge Input Logic

CMOS logic determines the voltages at the motor leads in Fig. 3 according to inputs enable (EN, a PWM square waveform of 0-5 V from the EVBU) and direction (DIR, high, or low as specified by programming). Since the emitters of the pnp transistors are tied to +BAT, neither will conduct current if the base is at battery voltage as well, since the required v_{be} of 0.7 V cannot result. Conversely, application of 0 to the base of an pnp allows current to flow. Operation is reversed for the npn transistors; 0 V applied to the base of an npn turns the transistor off while +BAT allows passage of current.

Table 1 specifies the logic used to turn the motors on and off in both forward and reverse. When EN is off, the motor should be off regardless of the input DIR. Referring to Fig. 3, we must apply high voltage to the bases of transistors A and D and low voltage to B and C. When EN is high we wish to turn the motor on in reverse for DIR = 0 (low voltage) and forward for DIR = 1 (high voltage). Setting A = 1 and C = 0 for DIR=0 ensures that current may only flow into D, through the motor, and out through B. B and D are therefore set to 1 and 0, respectively, to permit the desired current. Logic for DIR = 1 is the opposite the DIR = 0 case, reversing current and hence motor direction.

EN	**DIR**	**A**	**B**	**C**	**D**
0	0	1	0	0	1
0	1	1	0	0	1
1	0	1	1	0	0
1	1	0	0	1	1

Table 1: CMOS logic

Implementation: Phase I

The CMOS logic given in Table 1 led to the circuit design presented in Fig. 4. Transistors in the H-bridge are represented as single BJTs but are in fact darlingtons. This H-bridge configuration uses additional transistors to pull the H-bridge voltage inputs to battery voltage or ground. Examine the logic into the base of transistor A. For EN = 1 and DIR = 1, the voltage at the output of the inverter is high, causing the BJT on the other side of the resistor to conduct current and pull the collector voltage low. This results in an application of low voltage to the base of transistor A as desired. For any other combination of EN and DIR, the BJT just beyond the inverter does not turn on, yielding high voltage at the collector since no voltage drop occurs across the non-current carrying pull-up resistor. Similar tracing of logic for other conditions yields agreement with the logic shown in Table 1.

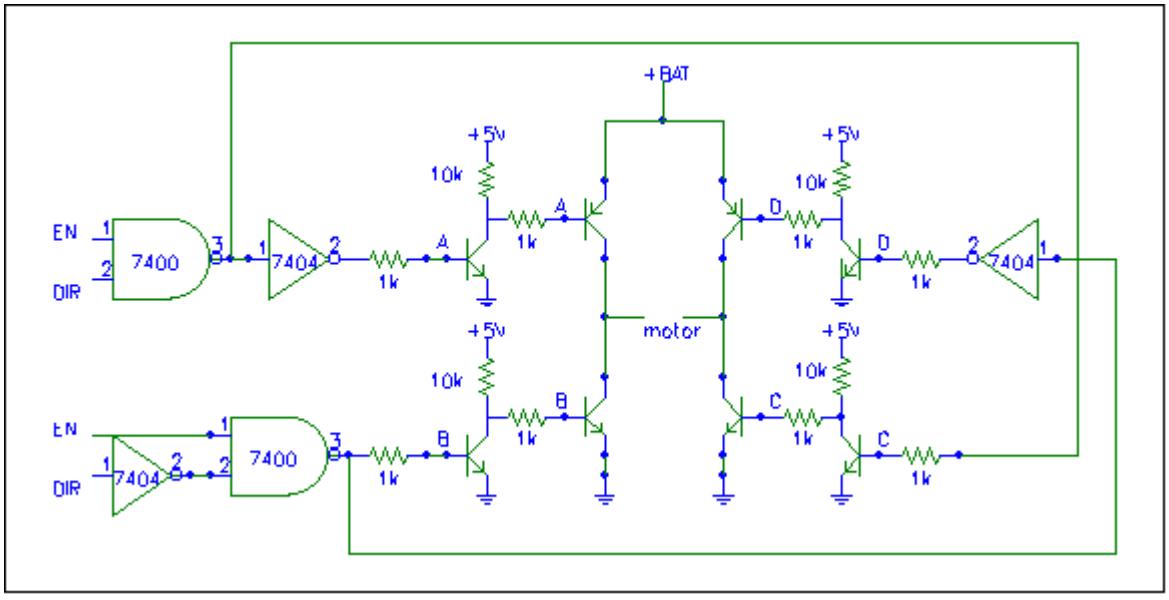

Fig. 4: BJT Implementation of H-Bridge Controller

Implementation: Phase II

The prototype for the circuit implemented in Phase I was cumbersome and occasionally drove the 68HC11 voltage below the required level. Our solution was the circuit diagrammed in Fig. 5. A single M3008 power MOSFET motor driver replaces each BJT H-bridge. Optoisolators replace the BJT input network, presenting two advantages. First, a single optoisolator chip replaces more than a dozen components, which significantly simplifies the circuit. Second, optical transmission of input information physically separates the microprocessor board from the motors, which completely protects the board from instantaneous switching effects.

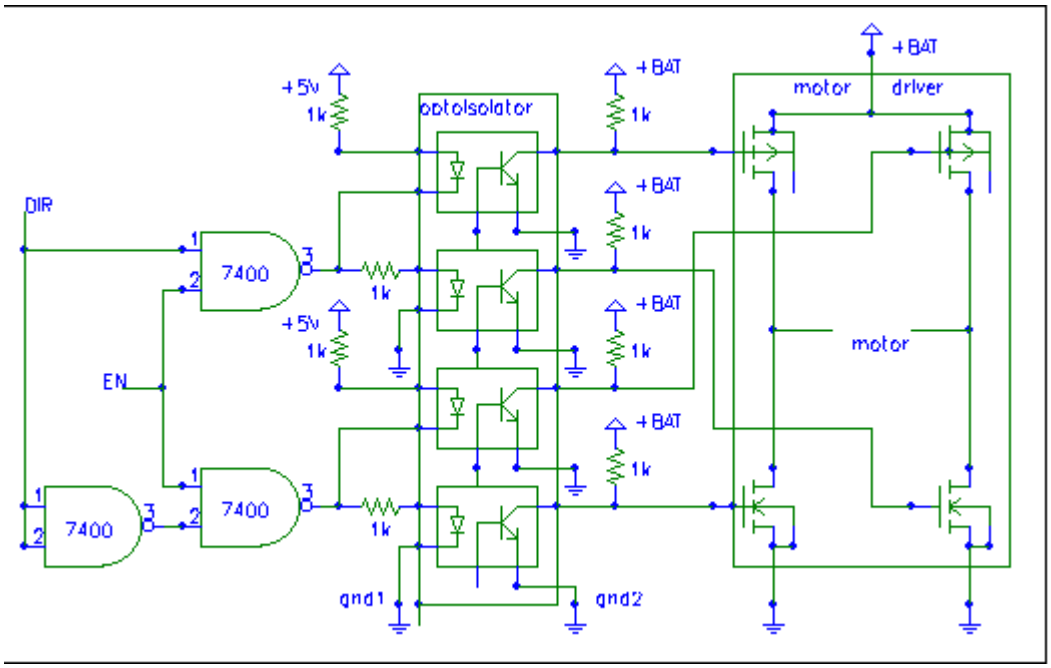

Fig. 5: Phase II Motor Driver

Turning on the input diode of an optoisolator activates the transistor at the output side of the device. The resulting current causes a voltage drop across the pull-up resistor to battery voltage, pulling low the input voltage to the corresponding MOSFET gate in the motor driver chip. Turning off the input diode, however, prevents current flow through the resistor on the output side and results in a gate voltage equal to the battery voltage. The CMOS logic which drives the optoisolator chip is the same as that in Phase I, but the connections to the Optoisolators are slightly different. Tying the output of a logic gate to the cathode side of an optoisolator input is equivalent to a logic inversion, which obviates the inverters in the Phase I circuit.

Motor Driver Construction

In order to minimize the risks associated with preliminary testing, we first tested each implementation of the motor driver circuit without the motor drivers and tested only the input logic. After correcting connection errors and verifying all logic patterns, we hooked up the motor drivers to a test motor to verify the logic functions in Table 1. The motor drivers can run far longer than the hovercraft can sustain battery power without ever becoming warm. Once the optoisolators were installed, we never lost connection to the microprocessor. Since the Phase II implementation ran to our satisfaction, we transferred the circuit to two permanent and compact solderboards, testing the new boards in the same fashion as the breadboarded version. After fully testing the motor drivers, we installed them on the hovercraft for use with the two propulsion fans. Testing on the hovercraft confirmed correct operation of the driver system.

Propulsion

Holly moves forward when both propulsion motors are spinning in the same direction and at the same speed. Braking is accomplished by switching both motors to reverse operation by the microprocessor, maintaining equal duty cycle to each motor. In our early tests, Holly could not move backwards because of the type of propeller used for movement. The fan blades are a type of blade known as *pusher propellers*, meaning that they are designed to push air in one direction. As a result, when the fans are run in reverse, they provide very little thrust. In environments with smooth, level surfaces, Holly is capable of limited reverse motion, but not enough to be useful for anything other than slowing forward motion of the hovercraft. Holly turns right when the duty cycle applied to the left motor is greater than that applied to the right motor, and left at the reverse condition. Holly rotates right by applying forward thrust to the left motor and reverse thrust to the right motor. Likewise, driving the right fan forward and the left in reverse results in left rotation.

Control Problems

Cumulative Motion

The main obstacle in controlling Holly's motion is what will be hereafter referred to as *cumulative motion*. Cumulative motion is the result of movement on a near-frictionless surface; if the craft is moving in one direction and turns, it will tend to slide laterally in the direction of its previous motion. Cumulative motion theoretically carries over indefinitely for any number of turns. In practice, however, friction eventually slows both the forward and rotational motion of the hovercraft. Therefore some measure of speed and time will be helpful in navigational control.

Other Problems

In addition to cumulative motion, the hovercraft tends to veer to one side as a result of even the slightest weight imbalance or unevenness of the surface above which holly maneuvers. Due to the difficulty in controlling an autonomous hovercraft we have developed several algorithms which combine to form a unique navigational system.

Control Algorithms

Speed

Appendix A contains a complete listing of the final ICC11 programming code used to control the hovercraft. In order to compensate for changes in speed, the program is designed to monitor speed according to the maneuvers performed. Functions forward, full_forward, reverse, and full_reverse drive both motors in the same direction and at the same speed for a given amount of time. Delays are set such that any call of these four functions results in the appropriate propulsion for a fixed amount of time. Speed is normalized at increments of one corresponding to one time delay of reverse motion. One call of the reverse function decrements speed by one, while full_reverse subtracts two from speed. Since reverse power is approximately one half as effective at slowing the hovercraft as forward is at speeding it up, forward and full_forward routines increase speed at double the rate of decrease of their reverse counterparts. The full_stop function illustrates the usefulness of the speed variable. Based on how many time delays the hovercraft has gone forward, the full_stop algorithm determines how long to run in reverse before the hovercraft comes to a complete stop.

Rotation

The rotate_left and rotate_right routines handle the rotational motion of the hovercraft. As described above, rotational motion occurs as a result of forward power applied to one motor and reverse power applied to the other. Since forward power is stronger than reverse power, the latter must operate at a higher duty cycle to minimize extraneous forward motion. We optimized the power to each motor through experimentation to achieve this desired minimization. In order to stop rotational motion, a counter-rotation must be applied for the same time as the rotation in the original direction.

Turning

Gradual right and left turns were the most challenging maneuvers to implement. Since the hovercraft is normally moving forward upon the initiation of a turn, cumulative motion tends to shift the hovercraft laterally as Holly completes the turn. The most effective means of controlling this problem is to perform two maneuvers: a normal turn operation, which involves applying more forward power to one propulsion motor than the other, followed by a counter-rotational maneuver as described above. The original turn maneuver must actually surpass the desired post-turn facing, thereby applying thrust to counter the original motion. The rotation operation then adjusts the facing of the hovercraft to the required heading. An additional difficulty to turning as opposed to rotation is that the initial forward speed and rotational motion, as well as weight imbalances, greatly affect the outcome of the turn. Because of the number of variables involved, development of the final left_turn and right_turn functions contained in the appendix required extensive testing and modification.

Oscillation and Damping

Despite the software-controlled speed and rotation controls described above, the hovercraft inevitably loses control due to unforeseen factors. Colliding with or even barely sideswiping an obstacle often adds significant rotational motion and/or speed which cannot be accounted for. Surface slopes normally undetectable by any but the most accurate methods may also affect speed and rotation. We have developed two software routines to reduce these uncontrollable effects: oscillate and dampen. Both functions use what little friction the hovercraft has to its advantage.

The oscillate routine alternates quickly between high forward power to one propulsion motor and low forward power to the other. The visible result is a left-to-right rocking of the hovercraft as it slowly moves forward. The shifting of air cushion height on the port and starboard sides increases the difficulty for the hovercraft to rotate, effectively canceling much of the rotational motion which existed prior to the oscillate routine. Dampen produces a similar effect to oscillate, but motor speed is reversed. Again, much of the hovercraft's rotational motion disappears. The dampen function, unlike oscillate, produces a significant decrease in forward speed.

Problems Encountered

In the conversion from IC to ICC11 we found two bugs in the ICC11 support libraries which turned a crude but somewhat reasonable collision avoidance behavior into complete chaos. First, IC and ICC11 reference the motors in an opposite manner, that is, motor 0 for one language is motor 1 in the other, and vice versa. Second, whenever one motor is driven to a 0 duty cycle and the other is written to operate in reverse, the latter will actually go forward instead.

SENSORS

IR Configuration

Holly's sensory array consists of integrated long and short range IR systems. A center mounted, forward facing IR emitter and a pair of collimated IR LEDs at port and starboard "light up" a 120 degree arc in front of the robot. Two wide-angle LEDs at the headlight positions blend the IR intensity to near-uniform levels throughout Holly's visual range, represented by the dashed cardioid in Fig. 6. A 74HC390 decade counter divides the 68HC11 E-clock to generate a 40kHz signal which activates the LEDs. Four hacked Sharp GPIU58Y IR detectors read the intensity of IR reflections as depicted by the solid cones in Fig. 6. Two cross-eye sensors blanket the area in front of the hovercraft, and a second pair in line with the outward-angled emitters increases the field of vision. Intermittent lines represent the long range IR cannon which, in combination with a fifth sensor, increases forward visibility.

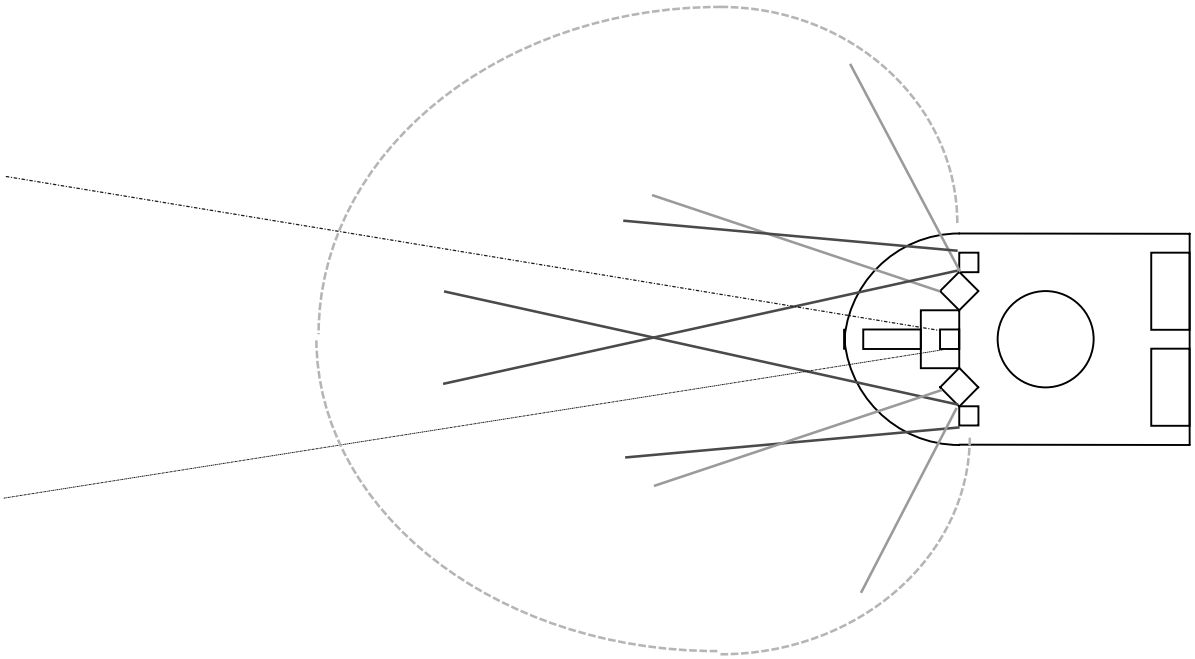

Fig. 6: IR System Configuration

IR System

Integrated IR System

Holly requires a long range detection system for several reasons. The hovercraft's high power propulsion motors, coupled with travel over a near frictionless surface, result in much higher cruising speeds than conventional robots. An additional consequence of travel over a near frictionless surface is an increased difficulty in changing direction. Accordingly, Holly must see upcoming obstacles as soon as possible in order avoid them in time. Our goal was to develop and integrate a long range IR object detection system with a standard short range system without resorting to new, specialized, or expensive components as was necessary for Holly's motor control.

Short Range IR

We removed all external resistance from the IR emitter/latch circuit on the ME11 and collimated an LED with a black, lens-capped tube to maximize the current through the LEDs and focus IR emissions while minimizing backscattering. Construction of the collimated infrared LED is shown in Fig. 7 below. Noise in the A/D system increased with the lack of resistance in the IR emission circuit. Additional capacitance across the system helped, but in order to get consistent IR readings we needed a 470Ω resistor in series with each latch output to its corresponding LED. These simple modifications allow IR detection up to 1 meter away. Lens collimation effects actually proved to reduce IR range, so later models have been free of optical trickery

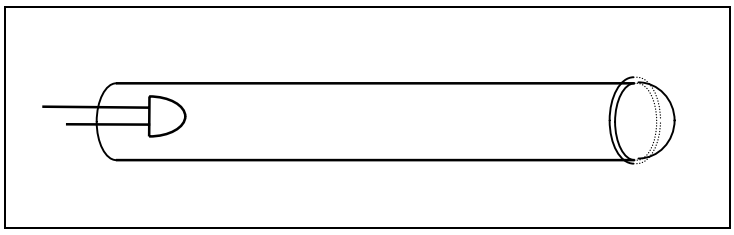

Fig. 7: Collimated Infrared LED

Long Range IR

The techniques used for short range IR were insufficient to give Holly the desired detection range. The ME11 expansion board produces a 40 kHz square wave with a 50% duty cycle. The transmission latch can source 35mA but an infrared LED can sustain at least 100mA continuously according to its data sheet. Therefore an LED can handle a 200mA amplitude square wave with a 50% duty cycle. We suspected that a 10% duty cycle with an amplitude of 500 mA using a monostable multivibrator, or one-shot, circuit would increase the intensity of the IR emissions without damaging the LEDs. The one-shot circuit shown in Fig. 8 triggers a 555 timer to produce an output signal with a short duty cycle and high current.

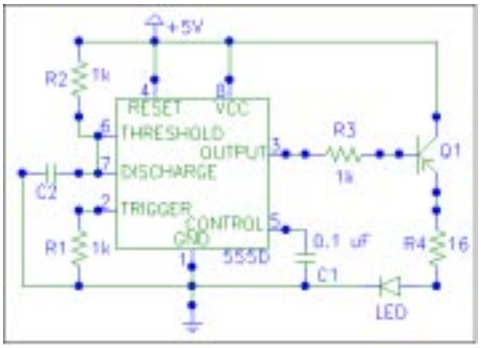

Fig. 8: One-Shot based on 555 Timer

R2 and C2 dictate the period of the timer output according to T = 1.1(R1)(C1). We connected the one-shot output to the base of a TIP127 PNP Darlington transistor to amplify the current through the LED. The LEDs were able to take the additional current. We took sensor readings starting 0.25 m away from a wall and additional readings at 0.25 m increments until we observed no change. Five readings were taken at each distance in order to ensure the precision of the data, listed in Table 2. Plotted data in Fig. 9 shows that readings level off somewhere between 2-3 meters, indicating improvement over the original IR system, but still not enough for our purposes.

Distance(m)	$Analog_1$	$Analog_2$	$Analog_3$	$Analog_4$	$Analog_5$
0.25	129	129	129	129	129
0.50	121	122	121	122	121
0.75	111	111	111	111	111
1.00	105	105	105	105	105
1.25	103	103	102	102	102
1.50	100	101	101	101	101
1.75	100	100	99	100	100
2.00	99	99	99	100	99
2.25	98	98	98	99	99
2.50	98	98	98	98	98
2.75	97	98	97	97	98
3.00	97	97	97	97	97

Table 2: One-shot circuit test results

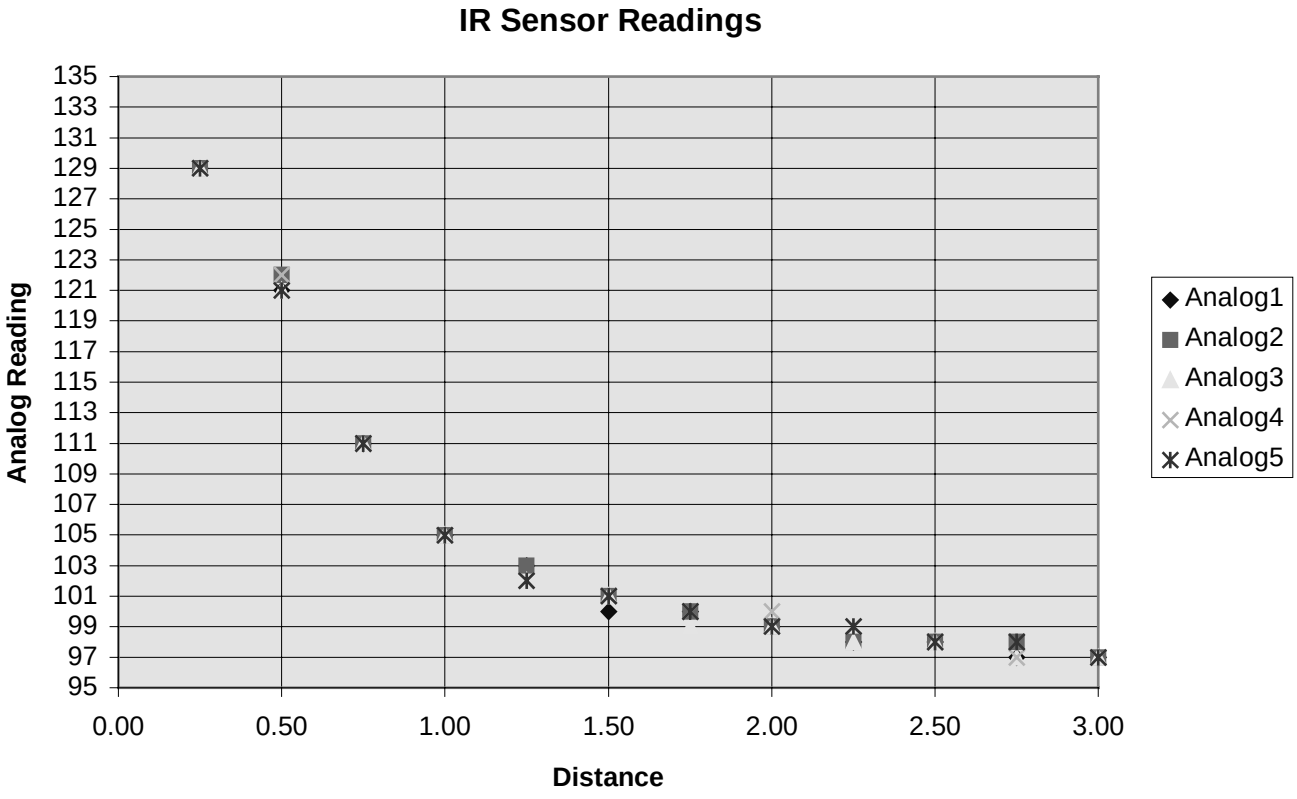

Fig. 9: Results of 555 Based One-Shot

We developed a second one-shot circuit, diagrammed in Fig. 10, to further improve the range of the long range IR system. This configuration provides a precise duty cycle just long enough for the IR sensor to respond. The shorter duty cycle permits a higher current signal and IR emission.

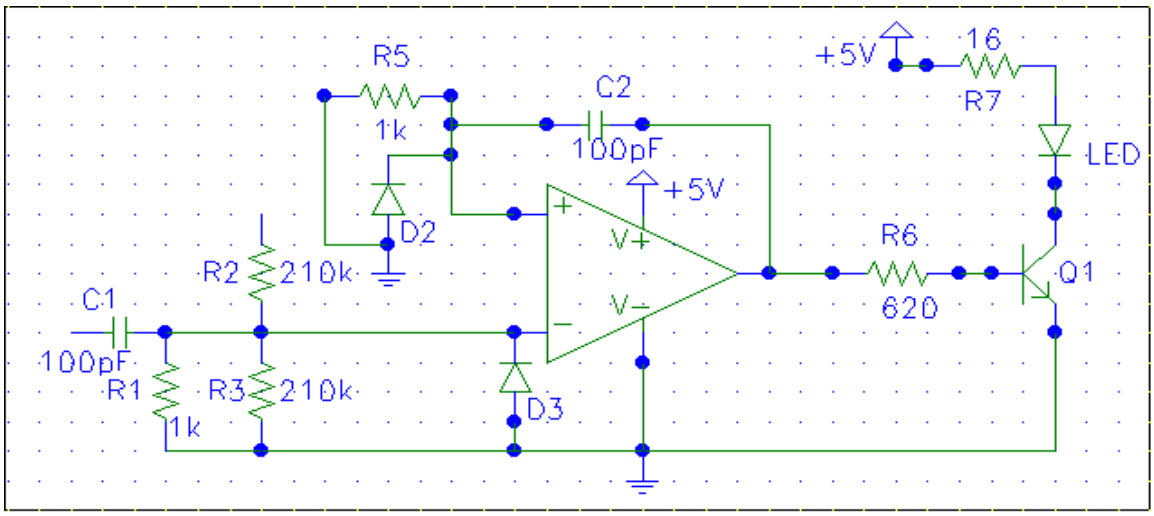

Fig. 10: One-Shot Based on Comparator

We discovered that the one-shot circuit requires precise adjustment of its construction in order to get useful results. During testing we burned out several infrared LEDs because the output had a higher amplitude or a longer duty cycle than anticipated. This incident prompted us to purchase a prepackaged monostable multivibrator to reduce the complexity, weight, and size of our onboard electronics. We chose to use a 74HCT123N dual retriggerable resettable multivibrator because of its availability and because it could be used to implement both of the designs discussed above. The relevant device specifications are shown in Appendix B. Using this IC we implemented the circuits shown in Figs. 8 and 10.

IR Cannon

Extensive testing suggested that the Sharp IR sensors do not respond well enough to the duty cycle/current magnification tradeoff to get the desired long range IR detection. We developed an alternate implementation to the multivibrator approach based on a circuit for a 10 meter range remote control unit. The ensuing high-intensity *IR Cannon* circuit consists of a TIP112 NPN transistor, 8 infrared LEDs, 1 red LED, a 470Ω resistor, and a base resistor that can be adjusted from 1kΩ to 33kΩ to adjust the sensitivity of the sensor for use in multiple environments. Fig. 11 below depicts the IR cannon circuit.

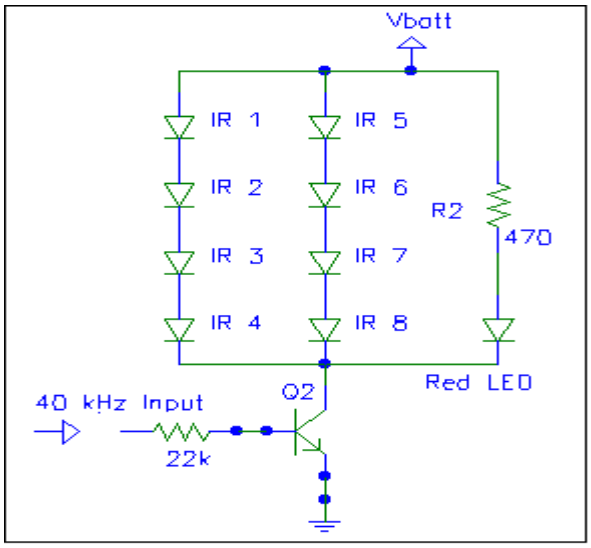

Fig. 11: IR Cannon Circuit Diagram

The transistor amplifies the 0-5 V 40 kHz signal from the ME11 board and the resulting collector current splits into three branches, one of which is a high-resistance/visible LED combination to indicate IR cannon activation. Neglecting the relatively small current through the visible light LED, currents can reach up to about 1 A through each row of infrared LEDs. According to component specifications there is no way these LEDs can handle 1 A DC, but software control can counter the power problem by pulsing the IR cannon for 50-100 mA every few seconds when readings are

desired. Thus the average power remains low and the LEDs are unharmed. Note that because of the IR Cannon's high intensity emission, all short range sensors remain saturated for approximately 100 ms, the discharge time for the Sharp sensors, after shutting off the cannon.

In the configuration shown in Fig. 6, the cannon allows Holly to detect objects at distances from 1.25 to 9 meters indoors and up to 6 meters outdoors. Greater range is possible, but we chose the base resistance to best complement the short range IR, which covers the 0.25 to 1.00 meter range. Analog output of the sensor is proportional to the amount of IR detected. Collision avoidance uses successive sensor readings to adjust Holly's speed and direction. The construction of the IR Cannon is shown in Fig. 12 below.

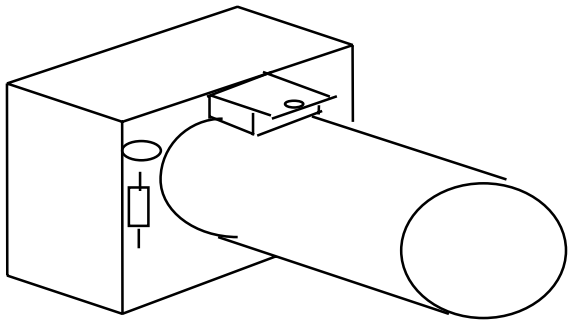

Fig. 12: IR Cannon Construction

When we apply board power, the IR latch starts in a random, unknown state. When we connected the cannon to the board for testing, the latch started high and six of the eight LEDs burned out before we could remove power. Accordingly, we installed a jumper to deactivate the IR cannon until the board is properly reset. We also changed the construction from permanently soldered LEDs to a socketed configuration which facilitates quick-changing of burned out LEDs.

BEHAVIORS

Collision Avoidance

Holly's central behavior is collision avoidance which monitors sensor readings and uses the basic control functions described in the previous section to avoid collisions based on infrared reflections. The microprocessor calls the turn, rotate, forward and reverse functions introduced previously to maneuver around obstacles detected by IR sensors. The hovercraft also exhibits more complex behaviors as described below.

Search

One of Holly's secondary behaviors is the "search". When Holly detects one or more obstacles which prevent motion in the forward direction, including gradual left and right turns, the hovercraft enters search mode by which it brakes to a full stop and rotates, scanning the environment for an open direction for travel. The search_left and search_right routines in the appendix realize the search behavior. Upon entering a search routine, Holly begins rotation in a certain direction, recording sensory data at variable intervals determined by time delay and rotational speed. A timer variable keeps track of rotational momentum. Once Holly finds an open space, the hovercraft engages in counter-rotational motion for a delay time which corresponds to the value indicated by the timer, bringing Holly to a stop. Holly then applies forward power in the newly discovered open path. A nested timing loop adds counter-rotation when no open area is seen for several delay times to prevent Holly from building up too much rotational motion and losing control.

Attack

Holly's final behavior uses long-range IR for an "attack" behavior. At intervals of a few seconds, the IR cannon pulses and the cannon sensor tracks the readings. Since the infrared ranging spectrum is so broad, long-range IR enables Holly to determine range to the nearest obstacles. Consequently, Holly uses the attack algorithm to exhibit different behaviors for various combinations of range and current speed. The attack function is documented in the appendix; Holly seeks the most open area and applies as much forward power as possible to reach the area quickly without colliding with any obstructions en route. The extreme range capability of the IR cannon allows Holly to achieve appreciable speeds while maintaining the ability to detect an object in time and veer away from it.

Integrated Behavior

Holly's behaviors combine to form an aggressive collision avoidance behavior. Long range and short range IR combine to provide Holly with range information, and programming directs Holly to speed up when large openings appear. In search mode, Holly purposely seeks the largest open space for travel, which further increases Holly's tendency to explore at high speeds. Oscillate and Dampen functions, in addition to stabilizing rotational motion, add an intimidation factor due to the rocking back and forth.

EXPERIMENTAL LAYOUT AND RESULTS

Several experiments led to the development of Holly's behaviors. We developed control algorithms with combinations of maneuvers, time delays, and long and short range IR thresholds based on experimental data.

Experiment 1: Short Range Obstacle Detection

This experiment was designed to test the short range IR detection system of the hovercraft. In order to do this, Holly was set up in a clear area with an object directly in front. This object was placed 114cm from the hovercraft and moved away in 114cm increments until the readings reach the point where they were no longer useful for object detection. Only the cross-view sensors were used in this test because they are the primary means of detecting nearby objects. The short range system was tested in three modes:

1) All short range LEDs turned on.

   This gives Holly the most range out of the short range system, dropping off at about 1 meter.

2) Center mounted LED and outward facing LEDs turned on, headlights turned off.

   This provided the shortest range, with a drop-off at about 0.5m.

3) Headlights and wide-angle LEDs on, center mounted LED off.

   This configuration yielded readings approximately 80% of those obtained in case 1, with a corresponding drop in effective range.

Table 3 below contains the results of Experiment 1, and Fig. 13 follows with a plot of the distance vs. analog reading of the cross-view sensors. The three modes are listed on the same graph and color-coded to make it easy to compare the results. Darker colors show the left cross-view sensor results while lighter colors indicate the right. The blue plots show the readings for case 1, case 2 is represented by the green plots, and the red plots show case 3. The plots show that effective short range detection out to about one meter results from the sensor/LED arrangement used on Holly.

	All LEDs On		Center/Wide On		Headlights/Wide On	
Distance(cm)	**Left**	**Right**	**Left**	**Right**	**Left**	**Right**
114	130	130	124	123	132	132
229	129	129	112	111	126	126
343	123	122	100	100	119	118
457	116	116	94	93	110	109
572	106	107	93	93	98	100
686	101	102	89	89	96	96
800	99	99	87	87	95	94
914	96	98	87	87	93	93

Table 3: Cross-view Sensor Readings For Short Range IR

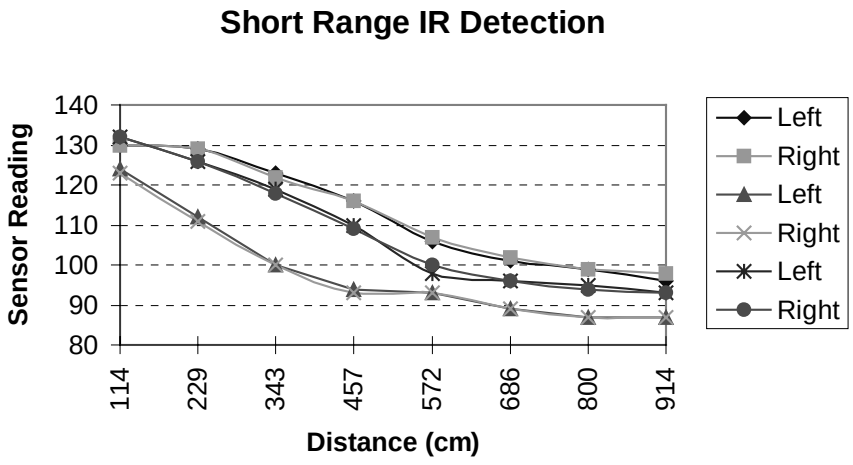

Fig. 13: Short Range IR Detection

Experiment 2: IR Cannon Base Resistance

Once the circuit for the IR Cannon was constructed, we had to determine what resistance in the base of the transistor would give us the desired detection range. We tested base resistances ranging from 1kΩ to 47kΩ by placing the hovercraft five meters from an obstacle and reading the value returned by the Sharp sensors. We took five readings for each resistance to account for error due to deviant readings. Table 4 lists the data and Fig. 14 plots the results. A 1kΩ resistor allowed high current through the LEDs, but would cause the LEDs to burn out if left on for more than one second at the 50% duty cycle, 40kHz input signal. The 33kΩ and 47kΩ resistors proved to be too high as it prevented the transistor from turning on. Resistances between these extremes limit the intensity of the IR emissions to accommodate operation in smaller or larger areas. Based on the data in Table 4 and the plot in Fig. 14 we decided to do additional testing with base resistances of 15kΩ and 22kΩ.

Resistance(kΩ)	Reading 1	Reading 2	Reading 3	Reading 4	Reading 5
IR Cannon Off	89	89	89	89	89
1	117	119	118	118	119
2.2	121	121	121	118	120
3.3	120	121	121	121	121
4.7	123	120	122	120	120
5.1	123	120	120	122	122
5.6	121	120	122	122	120
6.8	123	122	123	122	123
8.2	123	122	121	123	123
10	123	123	121	123	122
12	122	124	124	123	123
15	119	121	121	119	120
22	102	102	103	103	103
33	93	93	93	94	94
47	94	94	94	94	94

Table 4: Sensor Data at 5m for Varying Base Resistance

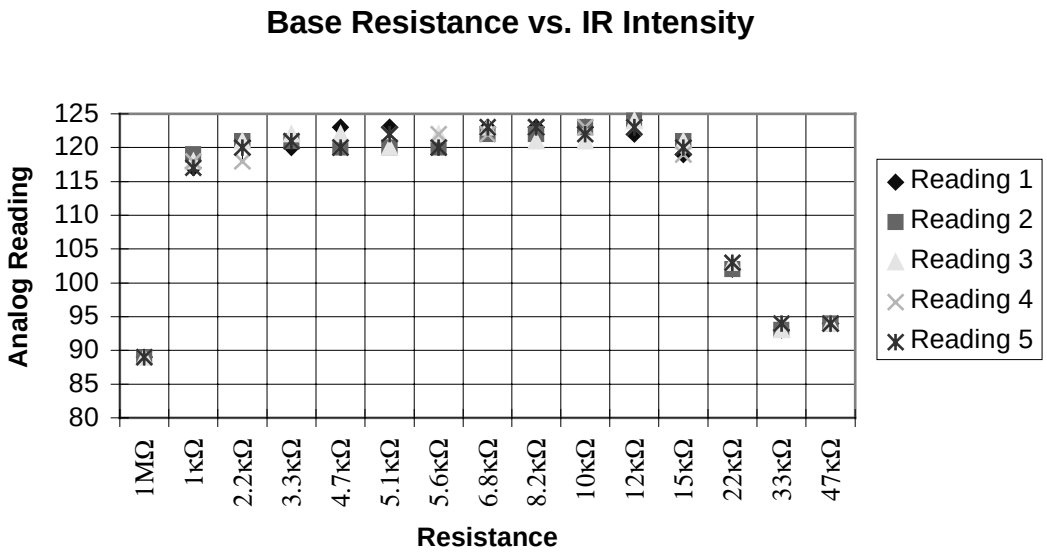

Fig. 14: Sensor Readings vs. Change in Base Resistance

Experiment 3: 15kΩ and 22kΩ Base Resistances

Experiment 2 provided useful data about the performance of the IR Cannon with different resistances into the base of the transistor. We determined that the performance of the 15kΩ or 22kΩ resistor would mesh well with our short range IR system so we devised this experiment to allow us to see which of the two would work better. The experiment was performed in the same manner to Experiment 1 except with longer distances.

Table 5 lists the collected data for the 15kΩ resistor, followed by a plot of the sensor readings vs. distance in Fig. 15. Table 6 contains the results obtained from the testing of the 22kΩ resistor and is also followed by a plot of sensor readings vs. distance, Fig. 16. The data suggests that the 15kΩ resistor gives a higher intensity IR output as indicated by the higher sensor readings at comparable distances to those from the 22kΩ resistor tests. This suggests longer range detection, meaning that the sensor will easily saturate in smaller areas. Since Holly will operate mostly indoors, we chose to use the 22kΩ resistor so that we wouldn't consistently blind Holly. Using this resistor gives us an effective long range system capable of detecting objects at distances ranging from approximately 1.5 to nine meters.

Distance(m)	Reading 1	Reading 2	Reading 3	Reading 4	Reading 5
2.06	131	131	131	131	131
2.29	130	132	129	130	132
2.51	130	131	130	131	129
2.74	128	130	130	130	130
2.97	129	128	126	127	127
3.20	125	126	124	126	127
3.43	126	124	124	124	126
3.66	125	124	124	123	124
3.89	124	123	124	123	124
4.11	123	123	123	122	122
4.34	123	122	122	122	121
4.57	122	121	122	123	121
4.80	120	120	121	122	122
5.00	119	121	121	119	120

Table 5: Sensor Readings with 15kΩ Base Resistance

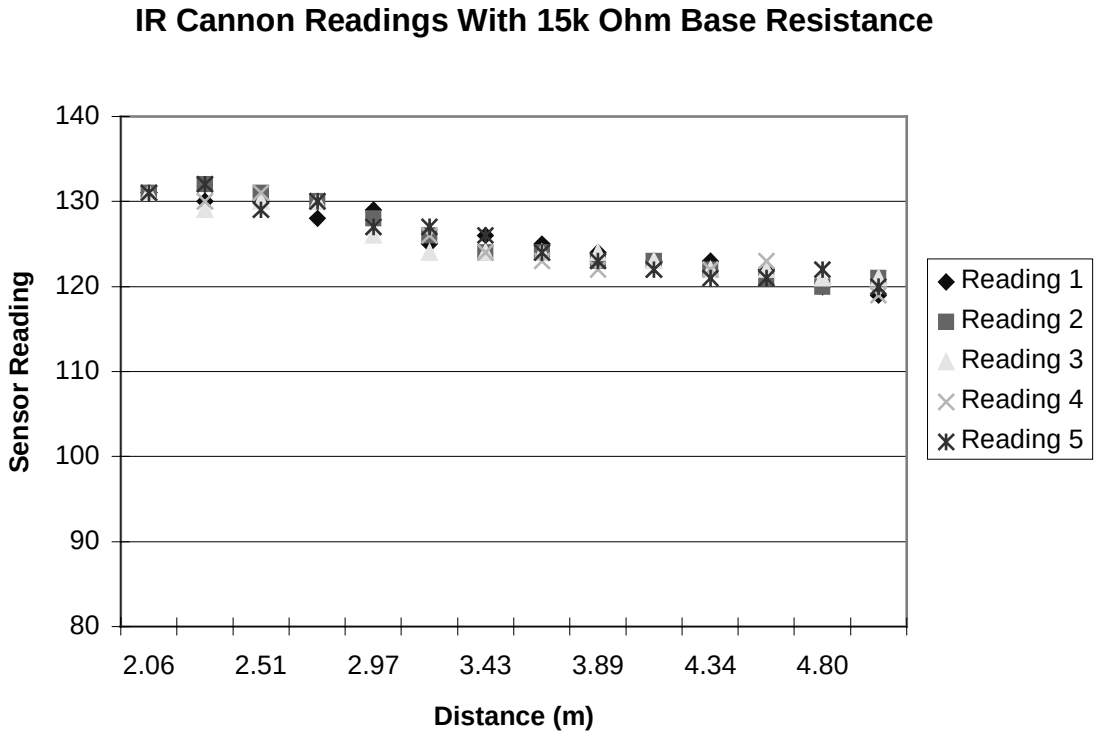

Fig. 15: Sensor Readings -- 15kΩ Base Resistance

Distance(m)	Reading 1	Reading 2	Reading 3	Reading 4	Reading 5

1.14	130	129	129	130	130
1.37	130	130	130	129	130
1.60	127	128	128	128	128
1.83	125	123	124	124	125
2.06	120	121	121	122	122
2.29	118	117	117	117	117
2.51	114	115	116	115	116
2.74	114	113	113	113	114
2.97	109	109	110	110	111
3.20	110	109	110	109	109
3.43	107	108	107	108	107
3.66	107	106	106	106	106
3.89	105	106	105	104	105
4.11	104	105	104	104	103
4.34	103	103	102	103	104
4.57	103	104	102	104	104
4.80	101	102	103	102	103
5.00	102	102	103	103	103
5.03	103	101	100	101	102
5.26	101	99	100	101	102
5.49	101	99	99	100	101
5.72	100	99	99	100	99
5.94	98	98	99	99	100

Table 6: Sensor Readings with 22kΩ Base Resistance

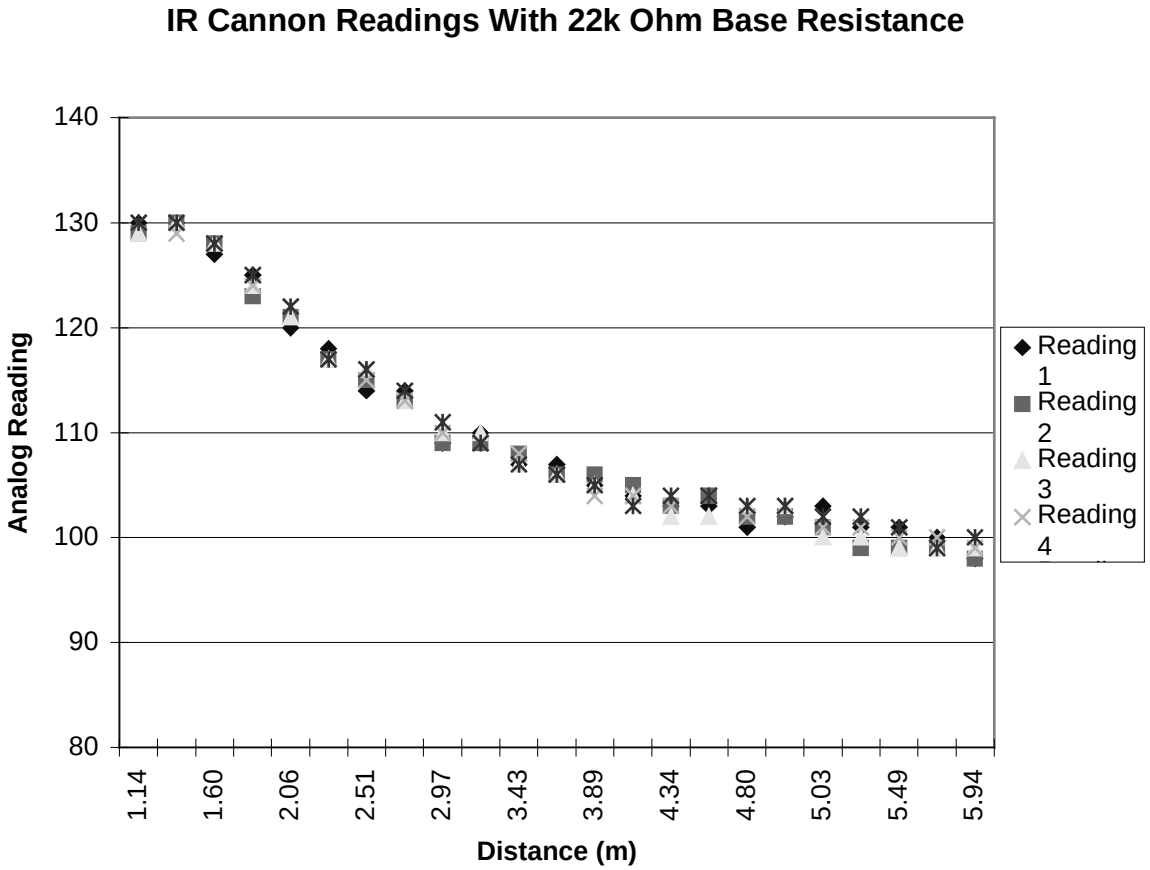

Fig. 16: Sensor Readings -- 15kΩ Base Resistance

Experiment 4: Controlling the Hovercraft

Before we could begin writing programs to produce behaviors in the hovercraft we had to learn how to *drive* one. To this end, we developed several simple programs to move the hovercraft in what we determined would be the basic motions of the robot. These functions were later used in combinations by the code in Appendix A to produce the desired collision avoidance behaviors.

The first step in controlling the hovercraft was to get it to move forward and stop. The program run_stop.c, listed in Appendix B, was written to perform this task. This program moved the hovercraft at half speed for a given delay, delay_f, then turned the motors to full reverse for another given time, delay_r, to stop the hovercraft. We were able to determine that the ratio delay_f : delay_r was 1 : 1. When moving at full speed the ratio becomes 2 : 1.

After we could successfully accelerate and stop the hovercraft, we began working on a way to make the hovercraft rotate in place. As shown in the program, rotate.c, we first tried turning one fan on in the forward direction and the other in reverse, but Holly had a tendency to move forward while rotating. To reduce the forward motion, and thus decrease the turning radius, we reduced the power to the forward motor and placed the other motor in full reverse. This reduced the problem but was not enough to completely eliminate it. We then decided to pulse the forward motor only a few times while keeping the reverse motor at full reverse. In order to stop the rotation, we simply reversed the direction of the rotation for a time equal to the time of the original rotation. This did not give us any improvement over the previous approach so we adopted our second technique as the standard rotate routine. It would seem that our choice of propulsion fan arrangement has limited our abilities to perform some operations.

The next objective was to get the hovercraft to perform a simple turn. We accomplished this using the program, turn.c, which moved Holly forward for 1 second, then made a 30° turn and continued moving in the new direction. The important idea behind the operation of this program is the use of the rotate function to counter the rotation induced at the start of the turn and straighten out the hovercraft in its new direction.

A significant problem encountered when we began combining the movement functions was a tendency to lapse into a spin when running over depressions on the surface upon which Holly was traveling. To counter this, we developed a function that would pulse the propulsion fans in reverse when called to reduce the angular velocity of the hovercraft. This program was tested by putting the hovercraft into a spin then running the program. When run, Holly was able to stop spinning in roughly half the time as compared to the time it would take to do the same based on friction between the skirt and ground alone. This function, wiggle.c, was also successful with the motors running in the forward direction.

The development of the control functions proved to be more challenging than expected due to the unique operating conditions presented by the hovercraft. Once developed, however, they provide the basis for the development of further behaviors. Using these functions as the core of our later programming, we were able to implement the behaviors previously discussed.

Experiment 5: Outdoor Range

We performed this experiment to determine the effectiveness of the IR Cannon outside, where the ambient infrared is more intense and IR detection systems lose significant range. We set up the hovercraft facing towards an area with a movable obstacle and took sets of five sensor readings starting at the sensor saturation point and increasing the distance until the sensor no longer returned useful data. We ran the tests with both the 15kΩ and 22kΩ resistors for comparison to each other and to the data recorded indoors. Tables 7 and 8 contain the test data for the 22kΩ and 15kΩ resistors. Figs. 17 and 18 are plots of the recorded data. The 22kΩ resistor seemed to have a maximum outdoor range of about 2.5 meters, while the 15kΩ resistor seemed to be reliable until about six meters.

Distance(m)	Reading 1	Reading 2	Reading 3	Reading 4	Reading 5
0.30	130	131	130	130	130
0.61	128	128	127	127	128
0.91	123	123	123	122	123
1.22	113	113	114	114	115
1.52	104	104	104	104	105
1.83	100	102	102	101	102
2.13	98	99	97	99	97
2.44	97	96	96	97	96
2.74	96	96	96	96	96
3.05	95	96	96	95	95
3.35	95	95	95	95	95
3.66	94	95	95	95	94
3.96	95	94	95	94	94
4.27	93	94	95	94	94
4.57	92	93	94	94	94

Table 7: Outdoor Sensor Readings with 22kΩ

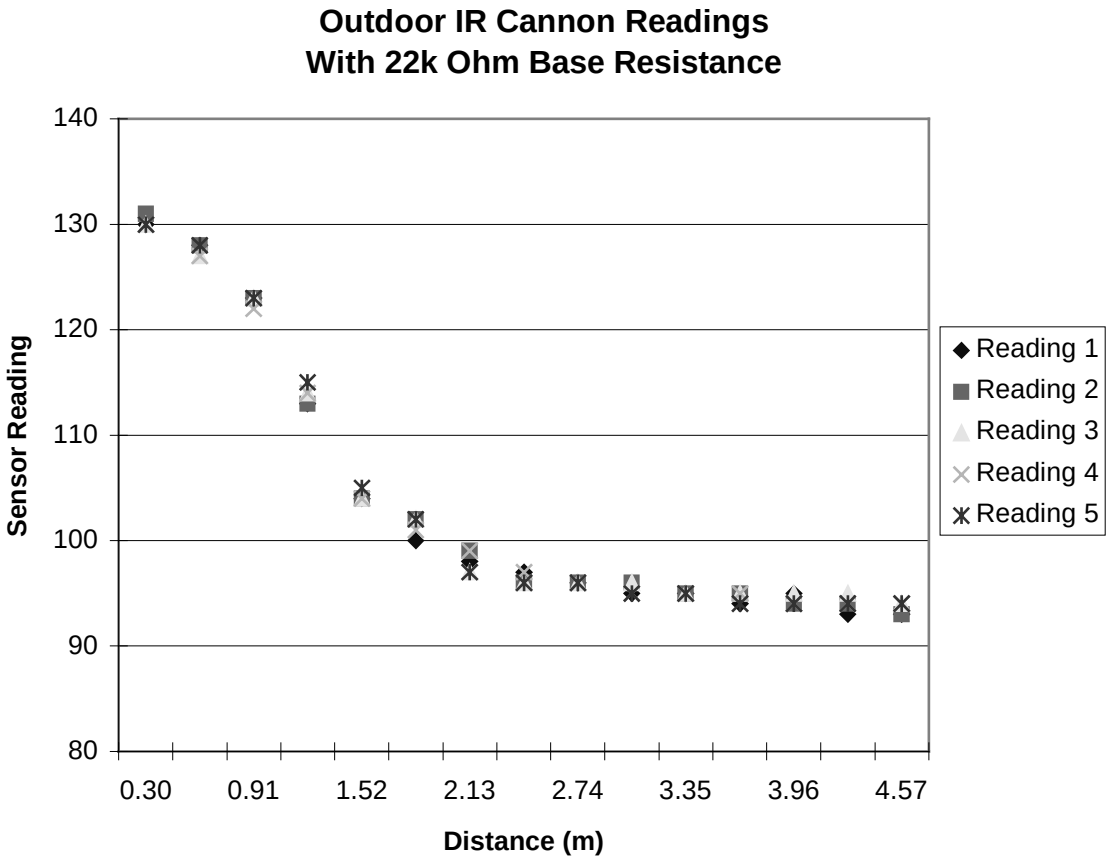

Fig. 17: IR Cannon--Outdoor Test Results with 22kΩ Resistance

Distance(m)	Reading 1	Reading 2	Reading 3	Reading 4	Reading 5
0.30	131	131	130	131	130
0.61	131	131	130	130	130
0.91	130	129	130	130	131
1.22	131	130	129	130	130
1.52	129	129	130	130	129
1.83	126	127	127	126	126
2.13	124	125	124	123	124
2.44	118	118	118	119	118
2.74	116	116	115	116	116
3.05	112	111	111	113	111
3.35	109	111	108	109	110
3.66	108	108	106	107	106
3.96	105	106	104	105	104
4.27	103	103	102	102	101
4.57	102	102	101	101	100
4.88	100	100	101	100	101
5.18	100	100	101	100	100
5.49	100	98	99	101	100
5.79	99	99	99	100	99
6.10	99	99	98	98	100
6.40	98	99	98	98	99
6.71	98	98	99	98	97
7.01	98	97	98	98	98
7.32	98	98	98	98	97
7.62	97	98	97	97	98

Table 8: Outdoor Sensor Readings with 15kΩ

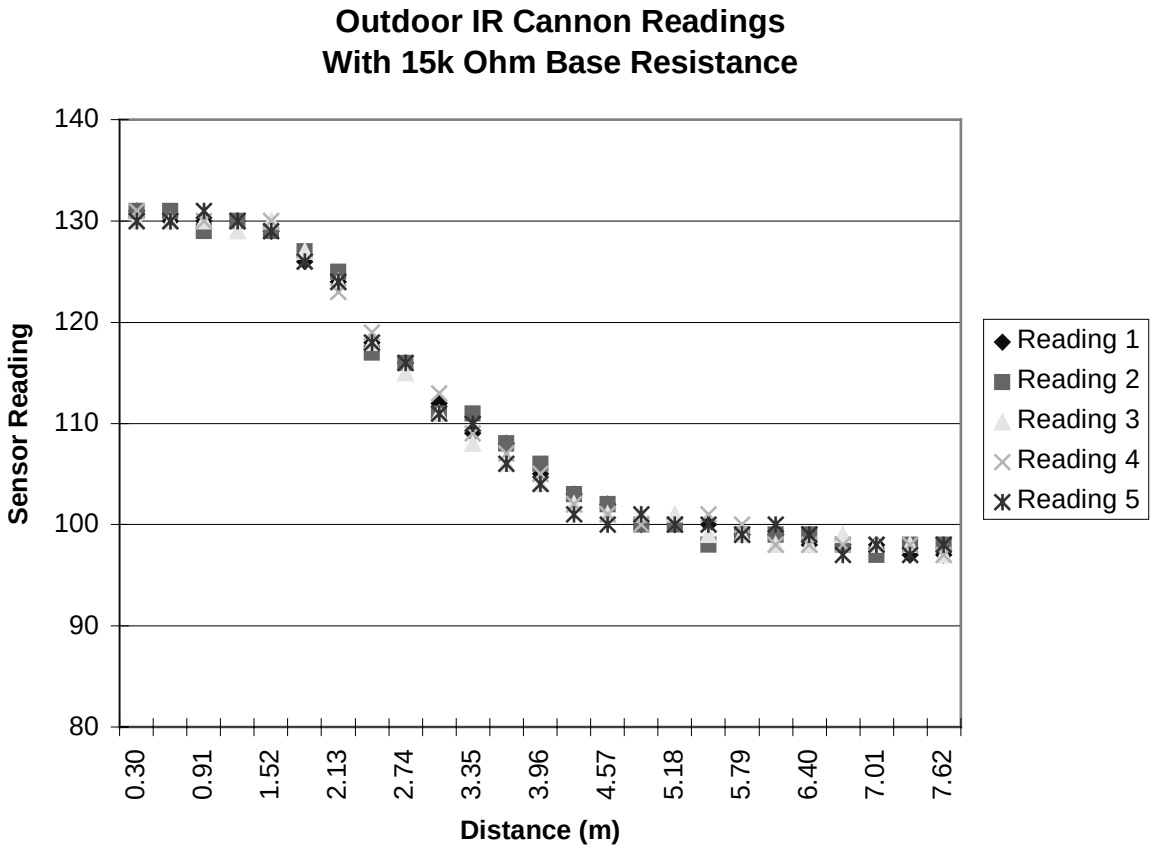

Fig. 18: IR Cannon--Outdoor Test Results with 15kΩ Resistance

Figs. 19 and 20 below show a comparison between the indoor and outdoor readings for both resistances. The 22kΩ resistor readings drop off significantly at short range when used outdoors, while the 15kΩ resistor returns consistently lower values when used outdoors. For Fig. 19 the data points labeled 'Reading *' are the results of the indoor tests and the data labeled 'Series' are for the outdoor tests. For Fig. 20, however, the indoor and outdoor test data labels are reversed.

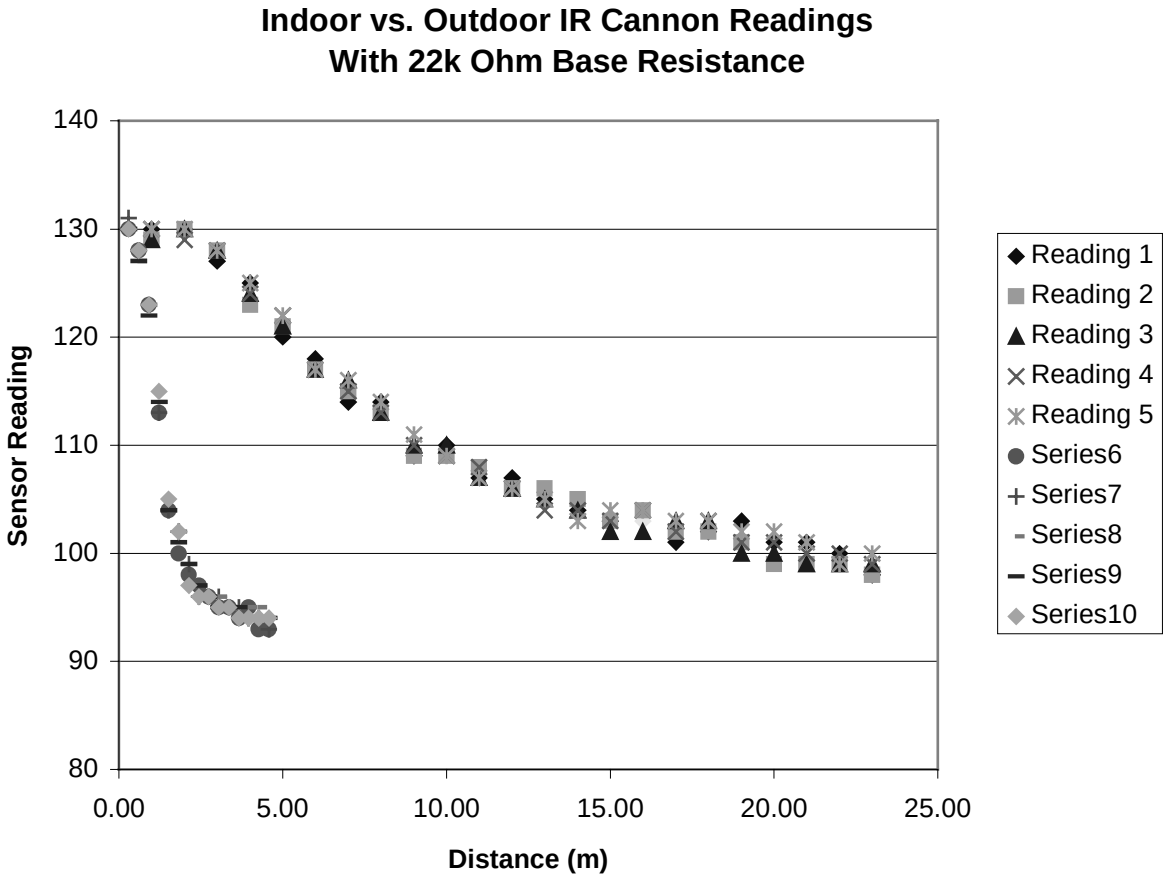

Figure 19: Indoor vs. Outdoor IR Cannon Test Comparison

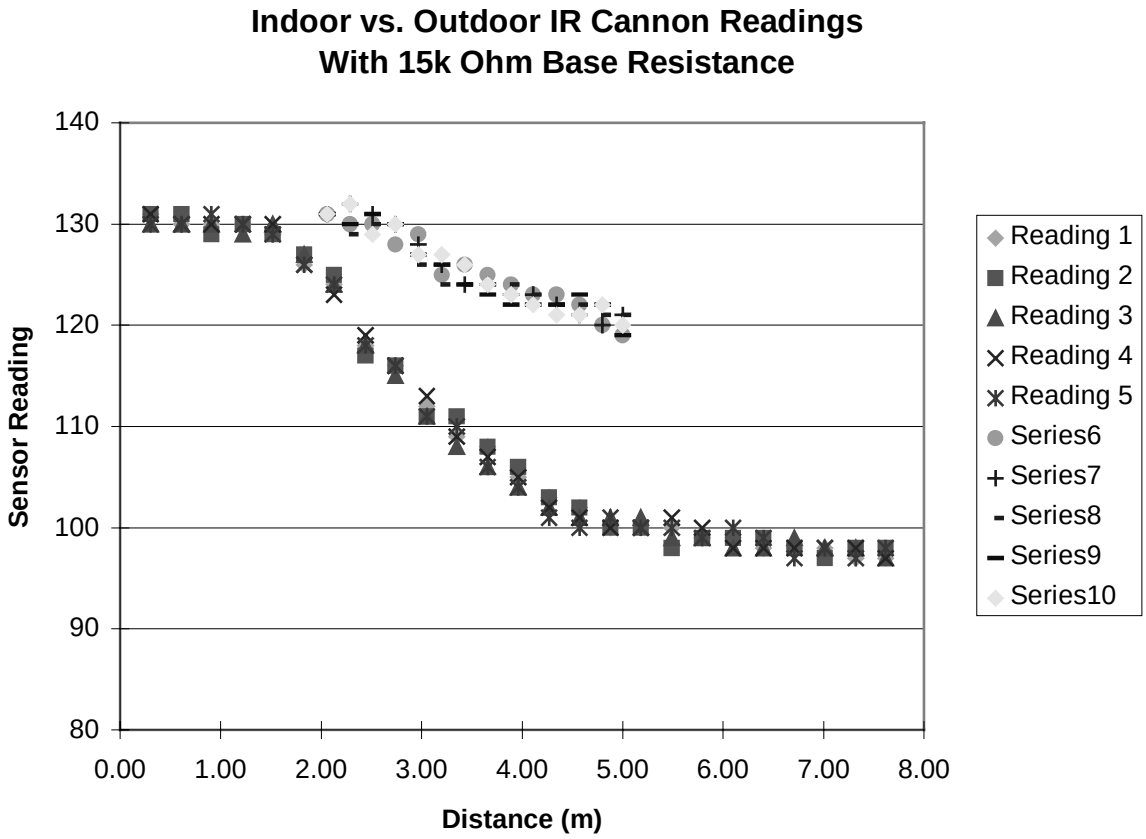

Figure 20: Indoor vs. Outdoor IR Cannon Test Comparison

It would appear that three independent IR detection systems could be used for outdoors: a short range system, a medium range system using the IR Cannon with a 22kΩ resistor, and a long range system using the 15KΩ resistor. Integrating these systems would allow a robot to produce a comprehensive map of its environment.

CONCLUSION

Holly is now a fully autonomous hovercraft which successfully avoids obstacles at medium to high speeds while evidencing several emergent behaviors. High-power motor drivers enable Holly's motors to draw the required current for high-speed propulsion. The long-range IR cannon, in combination with short range IR, allows Holly to travel at high speeds while maintaining the ability to detect obstacles in time to prevent collisions. The search and attack behaviors give Holly a somewhat bellicose personality; this hovercraft would make a formidable predator robot.

Software control of a high-speed autonomous hovercraft was a challenging task. Control of the hovercraft is still not perfect with IR alone, but the long range IR helps immensely. A parallel sonar project [ ] may prove to be the key in accurate navigational control, especially with regard to monitoring speed and rotation. Such control improvements might lead to additional behavioral capabilities such as mine laying/detecting, reconnaissance, and pattern recognition/targeting techniques associated with predator robots.

Both the long-range IR and motor driver systems work much better than we anticipated. The motor drivers can be used for robots whose motors draw considerably more current than Holly's, as the MPM3008 drivers are rated at 16 A. For applications similar to ours in power consumption, however, we recommend MPM3004 drivers which are rated at 10 A but are less expensive. Long-range IR reaches 10 meters, or more than 30 feet, which is more than enough for the hovercraft and should be plenty for other autonomous robots which use the IR cannon approach. Long-range IR readings also provide reasonably accurate range information.

Weight is a major consideration for Holly. Addition of the ME11 board weighed down the aft portion of the craft enough to hinder motion so that counterweights were required up front. Fortunately the second battery pack and IR cannon together produce approximately the correct weight for this purpose. Acquisition of metal hydride batteries more than compensated for the power drain associated with extra weight. Holly has enough power at this point to carry somewhat more weight, but any additions will also reduce the already limited run time.

DOCUMENTATION

[1] Peter J. Mantle, Air Cushion Craft Development (1st rev.), Mantle Engineering Co. Inc, 1980.

[2] John W. Meyers and Stuart Wilkinson, Mechanical Engineering Dept., Univ. of South Florida, Tampa, FL, A Electric Miniature Robotic Hovercraft for "Stealth" Land Mine Detection.

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

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

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

[6] Motorola, Motorola High-Speed CMOS Data, Motorola, Motorola, 1996.

[7] Motorola, Motorola Analog Interface IC Device Data:Volumes 1 and 2, Motorola, 1996.

VENDORS

Digi-key
701 Brooks Avenue South
P.O. Box 677
Thief River Falls, MN 56701-0677
1-800-344-4539
http://www.digikey.com

Newark Electronics
1-800-463-9257
http://www.newark.com

All Electronics Corp.
P.O. Box 567
Van Nuys, CA 91408-0567
1-800-826-5432
http://www.quinn.com/allcorp

National Semiconductor
http://www.national.com

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

```
/*  Appendix A:  Program Code  */
/*****************************************************************************
* EEL5666  Intelligent Machines Design laboratory, Fall 1996
* Programmer:   Bruce K. Miller
* Version:  hc48.c
* Description:
*   ICC collision avoidance program for an autonomous hovercraft - Holly.
*   Sensors: crosseye and front-mounted + outward-angled port and starboard.
*   Short range IR emitters stay on, IR cannon sends quick pulses.
*   Speed is monitored to determine brake time
*   Long range IR looks for open spaces toward which to migrate.
*****************************************************************************/

#include <mil.h>
#include <hc11.h>
#include <motor.h>
#include <analog.h>
#include <serial.h>

void full_forward();
void forward();
void coast();
void full_reverse();
void reverse();
void full_stop();
void rotate_left();
void rotate_right();
void turn_left();
void turn_right();
void search_left();
void search_right();
void cannon_pulse();
void scan();
void delay(int num);

/* Variable speed is a rough estimate of how fast hovercraft is traveling.
     Speed is always greater than or equal to zero.
     Speed thresholds moderate behavior to go fast but maintain control.
   Delay variables in milliseconds are used to time power to the motors.
     del_f is for forward/reverse power, del_s is motor/IR switching delay.
     del_t and del_c are turn and counter-turn delays, respectively.
     del_r is rotational delay, del_d is damping delay.  */

int speed = 0, timer = 0, thresh_speed = 5, top_speed = 10;
int del_f = 30, del_s = 3, del_r = 5, del_t= 10, del_c = 10;

/*  An analog reading above IR_thresh indicates presence of an obstacle.
      LIR_ variables allow holly to detect range and react appropriately.
    X_IR1 and X_IR2 are the left and right crosseye sensors.
    L_IR and R_IR are the sensors on the left and right sides.  */

  int IR_thresh = 100;
  int LIR_short = 125, LIR_med = 115, LIR_long = 105;
  int X_IR1, X_IR2, L_IR, R_IR, IR_cannon, IR_gun;

void main()
```

```
{

/* Initialization routines for motors, IR.  Turn on short range system.  */
   init_motors();
  init_analog();
  ADDR7 = 0xf8;

  while(1)
  {
/*  Take short range IR readings.  */
    X_IR1 = analog(0);
    X_IR2 = analog(1);
    L_IR = analog(2);
    R_IR = analog(3);

/*  If forward sensors blocked, Holly must find new direction of travel.  */
     if(((X_IR1>IR_thresh)||(X_IR2>IR_thresh)))
    {
/*  Shut motors off to avoid jamming motors in opposite direction.  */
      coast();
      delay(del_s);

/*  If right and left sensors, also blocked, stop immediately.
      Then rotate in search mode to find open area.  */
      if((L_IR>IR_thresh) && (R_IR>IR_thresh))
      {
        full_stop();
        if(L_IR >= R_IR) search_right();
        else search_left();
      }

/*  If at least one of the right or left sensors picks up an open area,
      turn away from obstacle ahead.  */
      else
      {
        if(L_IR >= R_IR) turn_right();
        else turn_left();
      }
    }

/*  After taking care of short range avoidance, turn on long range.  */
    cannon_pulse();

/*  Scan routine directs Holly to quickly move toward open areas  */
     scan();

/*  After clearing all obstacles, a quick forward burst to keep things going.  */
    forward();
    delay(del_f);
    coast();
    delay(del_s);
  }
}

/*  Forward and reverse motor control routines  */
```

```
void full_forward()
{
 motor(0,100);
 motor(1,100);
 speed+=2;
}


void forward()
{
 motor(0,50);
 motor(1,50);
 speed++;
}


void coast()
{
 motor(0,0);
 motor(1,0);
 delay(del_s);
}


void full_reverse()
{
 motor(0,-100);
 motor(1,-100);
 speed-=2;
 if(speed < 0) speed = 0;
}


void reverse()
{
 motor(0,-50);
 motor(1,-50);
 speed--;
 if(speed < 0) speed = 0;
}


void full_stop()
{
 while(speed>0)
 {
   full_reverse();
   delay(del_f);
 }
 speed = 0;
}

void rotate_left()
{
 motor(0,50);
```

```
  motor(1,-100);
}


void rotate_right()
{
  motor(0,-100);
  motor(1,50);
}


void turn_left()
{
  motor(0,100);
  motor(1,0);
  delay(del_t);
  rotate_right();
  delay(del_c);
  coast();
  delay(del_s);
}


void turn_right()
{
  motor(0,0);
  motor(1,100);
  delay(del_t);
  rotate_left();
  delay(del_c);
  coast();
  delay(del_s);
}


void dampen()
{
  int x, del_d = 5;
  for(x=0; x<del_d; x++)
  {
    motor(0,-100);
    motor(1,-10);
    delay(del_d);
    coast();
    delay(del_d);
    motor(0,-10);
    motor(0,-100);
    delay(del_d);
  }
}

void oscillate()
{
  int y, del_o;
  for(y=0; y<=del_s; y++)
  {
```

```
   motor(0,100);
   motor(1,0);
   delay(del_o);
   coast();
   delay(del_o);
   motor(0,0);
   motor(1,100);
   delay(del_o);
  }
}


/*  Search routines cause Holly to go into a controlled rotation,
      looking for an open area toward which to move.  */


void search_left()
{
  while(((X_IR1>IR_thresh)||(X_IR2>IR_thresh)))
  {
    rotate_left();
    delay(del_r);
    X_IR1 = analog(0);
    X_IR2 = analog(1);
    L_IR = analog(2);
    R_IR = analog(3);
    timer++;
    if(timer == 5)
    {
      while(timer>0)
      {
        rotate_right();
        del_r;
        timer--;
      }
    }
  }
  while(timer>0)
  {
    rotate_right();
    del_r;
    timer--;
  }
}

void search_right()
{
  while(((X_IR1>IR_thresh)||(X_IR2>IR_thresh)))
  {
    rotate_right();
    delay(del_r);
    X_IR1 = analog(0);
    X_IR2 = analog(1);
    L_IR = analog(2);
    R_IR = analog(3);
    timer++;
```

```
   if(timer == 5)
   {
     while(timer>0)
     {
       rotate_right();
       del_r;
       timer--;
     }
   }
  }
  while(timer>0)
  {
   rotate_left();
   del_r;
   timer--;
  }
}


/*  cannon_pulse routine turns on the IR cannon for approximately 100 ms,
      takes an IR reading, and then shuts the cannon off.
      This pulsing prevents the LEDs from burning out.  */


void cannon_pulse()
{
  ADDR7 = 0x02;
  delay(del_s);
  IR_cannon = analog(4);
  ADDR7 = 0xF8;
}

/*  Scan routine determines the range ahead to the nearest obstacle,
      Combines with speed information to determine behavior.
      The more open the space, the faster Holly goes,
      unless already moving too fast, at which point Holly stabilizes.  */

void scan()
{
  if(speed<thresh_speed)
  {
    if(IR_cannon > LIR_short) reverse();
    else if(IR_cannon > LIR_med) coast();
    else if(IR_cannon > LIR_long) forward();
    else full_forward();
  }
  else if(speed<top_speed)
  {
    if(IR_cannon > LIR_short) full_reverse();
    else if(IR_cannon > LIR_med) reverse();
    else if(IR_cannon > LIR_long) coast();
    else forward();
    ADDR7 = 0xF9;
  }
  else
  {
```

```
    if(IR_cannon > LIR_med) full_reverse();
    else reverse();
   }
   delay(del_f);
   if(speed < thresh_speed) oscillate();
   else dampen();
   coast();
   delay(del_s);
}

/*  Delay routine takes an input in terms of number of timer overflows.
      Delay variables determine how much time is wasted in this loop.
      Used to leave motors or IR on for given amounts of time.  */

void delay(int num)
{
  int bs = 0;
  while(num >= 0)
  {
    while((TFLG2 & 0x80) != 0x80)
    {
      bs++;
    }
    TFLG2 = 0x80;
    num--;
  }
}
```

```
/*  Appendix B:  Movement Control Test Code  */
/*************************************************************************
* EEL5666  Intelligent Machines Design laboratory, Fall 1996
* Programmer:   Aamir Qaiyumi
•    Description:  Move forward, then Stop
*************************************************************************/

#include <mil.h>
#include <hc11.h>
#include <motor.h>
#include <analog.h>

void forward();
void coast();
void full_rev();
void delay(int num);

void main()
{

/* Delay variables in clock cycles are used to time power to the motors.
                del_f is for forward power, delay_r is for reverse power.
                del_t is a time increment for turning or rotating in either direction.
                del_s is a switching delay between forward and reverse power,
                        and for pulsing on long-range IR.
                del_c is a time increment for countering the rotation in either direction. */

int del_f=120,del_r=120,del_s=3,del_d=10,del_t=20,del_c=20;

        init_motors();                  /*  initialize motors  */
        init_analog();                  /*  initialize IR system  */

        forward();
        delay(del_f);
        coast();
        delay(del_s);
        full_rev();
        delay(del_r);
        coast();
        delay(del_f);

        coast();
        while(1) delay(del_d);
}

void forward()
{
        motor(0,50.0);
        motor(1,50.0);
}

void coast()
{
        motor(0,0.00);
        motor(1,0.00);
}
```

```
void full_rev()
{
	motor(0,-100.0);
	motor(1,-100.0);
}

void delay(int num)
{
	int bs = 0;
	while(num >= 0)
	{
		while((TFLG2 & 0x80) != 0x80)
		{
			bs++;
		}
		TFLG2 = 0x80;
		num--;
	}
}
```

```
/*  Appendix B:  Movement Control Test Code  */
/*************************************************************************
* EEL5666  Intelligent Machines Design laboratory, Fall 1996
* Programmer:   Aamir Qaiyumi
•    Description:  Rotation function.
*************************************************************************/

#include <mil.h>
#include <hc11.h>
#include <motor.h>
#include <analog.h>

void coast();
void rotate_right();
void rotate_left();
void delay(int num);

void main()
{

/* Delay variables in clock cycles are used to time power to the motors.
                del_f is for forward power, delay_r is for reverse power.
                del_t is a time increment for turning or rotating in either direction.
                del_s is a switching delay between forward and reverse power,
                        and for pulsing on long-range IR.
                del_c is a time increment for countering the rotation in either direction. */
int del_f=45,del_r=60,del_s=3,del_d=10,del_t=20,del_c=20;

        init_motors();                  /*  initialize motors  */
        init_analog();                  /*  initialize IR system  */

        rotate_right();
        delay(del_r);

        coast();
        delay(del_s);

        rotate_left();
        delay(del_c);

        coast();
        while(1) delay(del_d);
}

void coast()
{
        motor(0,0.00);
        motor(1,0.00);
}

void rotate_left()
{
        motor(0,25.0);
        motor(1,-100.0);
}
```

```
void rotate_right()
{
	motor(0,-100.0);
	motor(1,25.0);
}

void delay(int num)
{
	int bs = 0;
	while(num >= 0)
	{
		while((TFLG2 & 0x80) != 0x80)
		{
			bs++;
		}
		TFLG2 = 0x80;
		num--;
	}
}
```

```
/*  Appendix B:  Movement Control Test Code  */
/*************************************************************************
* EEL5666  Intelligent Machines Design laboratory, Fall 1996
* Programmer:   Aamir Qaiyumi
•    Description:  Turns the hovercraft 30 degrees.
*************************************************************************/

#include <mil.h>
#include <hc11.h>
#include <motor.h>
#include <analog.h>

void forward();
void coast();
void full_rev();
void reverse();
void rotate_right();
void rotate_left();
void turn_right();
void turn_left();
void delay(int num);

void main()
{

/* Delay variables in clock cycles are used to time power to the motors.
                del_f is for forward power, delay_r is for reverse power.
                del_t is a time increment for turning or rotating in either direction.
                del_s is a switching delay between forward and reverse power,
                        and for pulsing on long-range IR.
                del_c is a time increment for countering the rotation in either direction. */

int del_f=30,del_r=60,del_s=3,del_d=10,del_t=15,del_c=15;

        init_motors();              /*  initialize motors  */
        init_analog();              /*  initialize IR system  */

        forward();
        delay(del_f);
        coast();
        delay(del_s);
        turn_right();
        delay(del_t);
        coast();
        delay(del_s);
        rotate_left();
        delay(del_c);
        coast();
        delay(del_s);
        forward();
        delay(del_f);
        coast();
        full_rev();
        delay(del_r);
```

```
	coast();
	while(1) delay(del_d);
}

void forward()
{
	motor(0,50.0);
	motor(1,50.0);
}

void coast()
{
	motor(0,0.00);
	motor(1,0.00);
}

void full_rev()
{
	motor(0,-100.0);
	motor(1,-100.0);
}

void rotate_left()
{
	motor(0,100.0);
	motor(1,-100.0);
}

void rotate_right()
{
	motor(0,-100.0);
	motor(1,100.0);
}
void turn_right()
{
	motor(0,100.0);
	motor(1,00.0);
}
void turn_left()
{
	motor(0,00.0);
	motor(1,100.0);
}
void delay(int num)
{
	int bs = 0;
	while(num >= 0)
	{
		while((TFLG2 & 0x80) != 0x80)
		{
			bs++;
		}
		TFLG2 = 0x80;
		num--;
	}
}
```

```
/*************************************************************************
* EEL5666  Intelligent Machines Design laboratory, Fall 1996
* Programmer:   Aamir Qaiyumi
* Description:   'Wiggles' the robot to dampen angular momentum.
*************************************************************************/
#include <mil.h>
#include <hc11.h>
#include <motor.h>
#include <analog.h>

void coast();
void right_rev();
void left_rev();
void wiggle();
void delay(int num);

void main()
{

/* Delay variables in clock cycles are used to time power to the motors.
                del_f is for forward power, delay_r is for reverse power.
                del_t is a time increment for turning or rotating in either direction.
                del_s is a switching delay between forward and reverse power,
                        and for pulsing on long-range IR.
                del_c is a time increment for countering the rotation in either direction. */

/* int del_f=30,del_r=75,del_s=20,del_d=10,del_t=15,del_c=15,del_w=30; */

        init_motors();                  /*  initialize motors  */
        init_analog();                  /*  initialize IR system  */

        while(1)
        {
                ful_rev();
        }
}

void forward()
{
        motor(0,50.0);
        motor(1,50.0);
}

void coast()
{
        motor(0,0.00);
        motor(1,0.00);
}

void left_rev()
{
        motor(0,-100.0);
        motor(1,-10.0);
}

void right_rev()
```

```
{
	motor(0,-10.0);
	motor(1,-100.0);
}

void wiggle()
{
int del_s=3,del_l=5;
	left_rev();
	delay(del_s);
	coast();
	delay(del_l);
	right_rev();
	delay(del_s);
	coast();
	delay(del_s);

}

void delay(int num)
{
	int bs = 0;
	while(num >= 0)
	{
		while((TFLG2 & 0x80) != 0x80)
		{
			bs++;
		}
		TFLG2 = 0x80;
		num--;
	}
}
```

```
/*************************************************************************
* EEL5666  Intelligent Machines Design laboratory, Fall 1996
* Programmer:   Aamir Qaiyumi
* Description:    Tests the IR Cannon by firing a 100 ms pulse and taking a reading. It then
*                         outputs the value to the terminal.
*************************************************************************/

#include <mil.h>
#include <hc11.h>
#include <serial.h>
#include <analog.h>

void delay(int num);

void main()
{

    int reading;

    init_serial();
    init_analog();

    ADDR7 = 0xFA;                   /*  turn on IR emitters  */
    TFLG2 = 0x80;
    delay(3);
    reading=analog(4);
    ADDR7 = 0x00;                      /*  turn off all IR emitters  */
    write("Bazooka Sensor: ");
    write_int(reading);


}

void delay(int num)
{
  int bs = 0;
  while(num >= 0)
  {
    while((TFLG2 & 0x80) != 0x80)
    {
      bs++;
    }
    TFLG2 = 0x80;
    num--;
  }
}
```