University of Florida
Department of Electrical and Computer Engineering
EEL 5666
Intelligent Machine Design Laboratory

Final Report:
Alph and Ralph

Student Name: Megan Grimm
Date: 12/9/98
TAs: Scott Jantz
Aamir Qaiyumi
Ivan Zapata
Instructor: A.Antonio Arroyo

Table of Contents

Abstract

Alph and Ralph are a herding robot and its charge: the sheepdog and the sheep, as it were. Using collision avoidance and wandering behaviors, Ralph will roam a given area and attempt to evade Alph, which will be trying to find and herd it to a stationary sonar beacon. Alph's herding behavior is triggered by a hit on the back bumper, at which point Alph will use its sonar array search for and herd Ralph, using IR and sonar data, to the sonar beacon. Alph also incorporates a piezo buzzer with which it will signal Ralph when it detects that it is within two feet of the beacon. Ralph, on receiving the audio signal, will stop wandering and go into a holding pattern while Alph pauses, turns, and continues searching for more quarry elsewhere.

Executive Summary

Alph and Ralph were developed to determine whether herding behavior was both possible and feasible in a small autonomous robot. The overall goal of the project was for Alph, the larger, more intelligent robot, to direct Ralph, the smaller, simpler robot, to a specific location; in this case, a sonar beacon. Several projects have been implemented in the past in which the more intelligent robot leads the other to the specified location; however, I was interested to see whether the smaller robot could be directed by chasing it to the beacon by keeping the small robot constantly interposed between the beacon and herding robot.

Ralph's behaviors therefore include collision avoidance – or, to be more precise, evasion of pursuit – and behavior arbitration, by which a signal from the pursuing robot, Alph, will cause it to stop evading and either stop or enter a holding pattern. This is necessary in order to demonstrate Alph's recognition of the target location.

Ralph was constructed on a TJ platform modified by the addition of a head bearing four sonar emitters, a microphone via which it could receive the behavior arbitration signal, and two extra rear IR detectors by which Ralph can detect pursuit. The standard platform also includes front and rear IR emitters, front IR detectors, and a bumper for collision avoidance.

Alph's behaviors include collision avoidance, sonar tracking, time-of-flight interpolation, and behavior arbitration. Lack of actual herding behavior will be discussed.

Alph was constructed on a Talrik platform with a sensor suite including IR emitters and detectors for collision avoidance, bump switches for collision avoidance and behavior arbitration, and a sonar receiver mounted on a swiveling head.

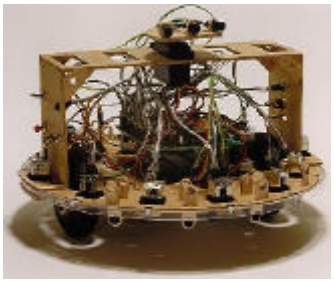
Introduction

Since the inception of robotics, one of the most popular and challenging problems has been the simulation of natural (animal) behaviors. The challenge of developing autonomous agents capable of meaningful interaction is equally fundamental and challenging. This project is designed to further exploration of the development of herding behavior in robots, using one small agent which will wander randomly and try to avoid the other agent, which will be trying to direct the smaller agent to a stationary beacon.

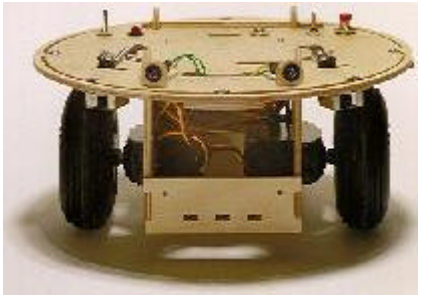
This paper will include a general description of the integrated two-agent system as well as more detailed descriptions of the platforms, actuation, sensors, and behaviors.

Integrated System

The herding robot, Alph, is constructed on a Talrik platform. It incorporates the standard IR and bump sensors as well as a sonar transducer, which will be used to find the robot's position in the room and to find and follow the other robot, and a 4.2 kHz piezo buzzer with which it can signal the other robot.

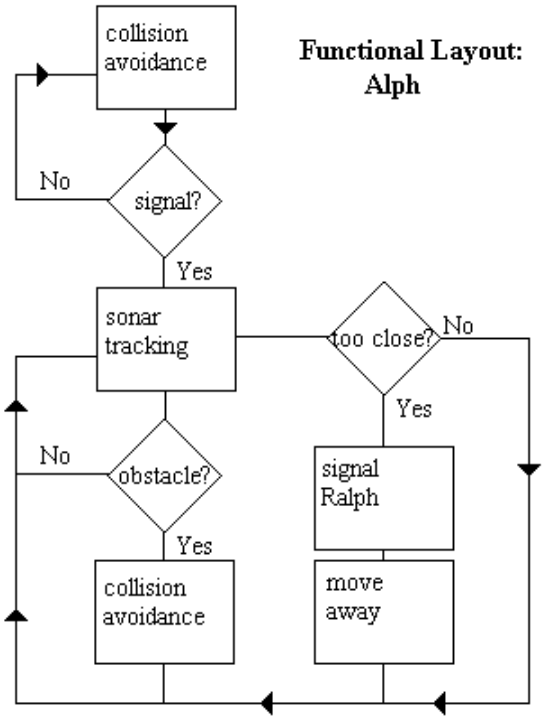


The herded robot, Ralph, is constructed on a TJ platform. It too incorporates IR and bump sensors, with the additions of a condenser microphone that will allow Ralph to respond to signals from Alph or the user, and a sonar beacon by which Alph will be able to track it.

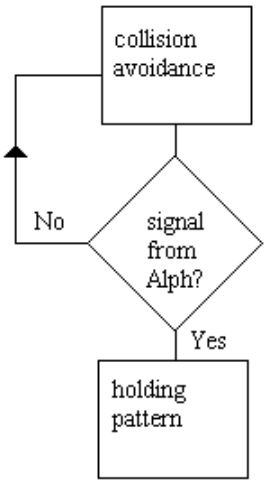


The scenario envisioned was as follows: both robots will begin in standard collision avoidance mode. On a signal from the user, in this case a strike on the rear bumper, Alph will stop wandering and go into "scan" mode. Using its sonar array, Alph will seek out Ralph, using its IR and sonar arrays, and attempt to herd it to a stationary sonar beacon. When Alph detects that it is within two feet of the beacon, it will signal Ralph via the

piezo buzzer. Ralph will then stop and go into a holding pattern, while Alph pauses, turns, and continues its search for more sonar signals.



Functional Layout: Alph



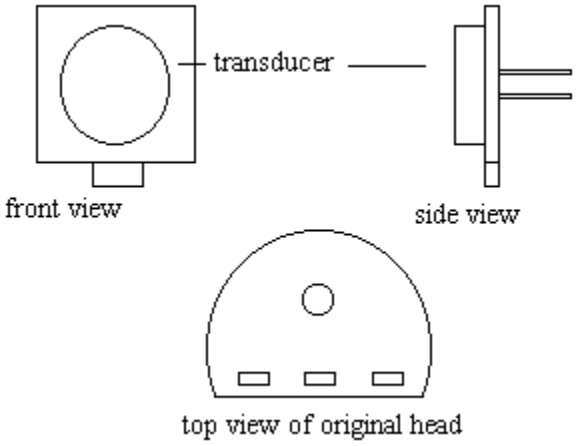
Functional Layout: Ralph

Currently, Ralph accomplishes its goals of collision avoidance and audio response successfully. Alph successfully demonstrates integrated collision avoidance and sonar tracking, as well as time-of-flight interpolation and behavior arbitration. However, the routine necessary to differentiate between the beacon and Ralph was unsuccessful in its multitude of iterations. Without this routine, Alph cannot distinguish between Ralph and the beacon and therefore will simply move toward whichever emitter it happens to find.

Mobile Platform

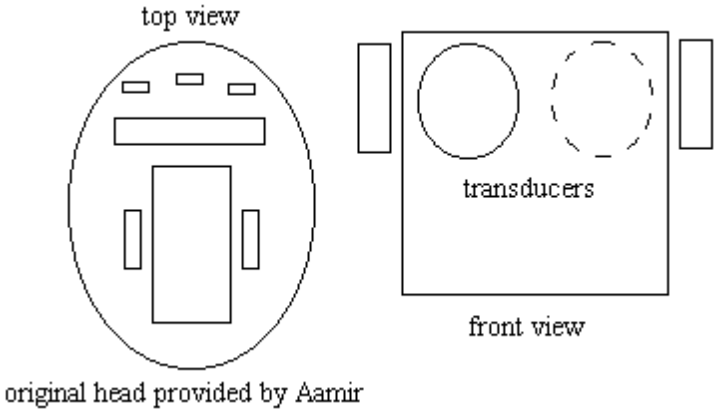
The herding robot, Alph, uses a Talrik platform for its robust design and ease with which its sensor suite can be expanded, thanks to the "bridge" design and swiveling head mount. The herded robot, Ralph, is built on a TJ platform. This platform was used both for its durability and the relative ease with which its sensor array can be modified. The two platforms were also chosen because, as pre-established designs, they are well documented and have, for the most part, already been debugged to a large degree. This frees up more time for development of complex behaviors and sensor expansion that would otherwise have been occupied with the problems attendant with a completely untested platform.

Alph's head has been modified with the addition of a one-inch square vertical mount for the sonar receiver.



Ralph's modification involves the addition of a vertically-mounted three-inch square head on which are mounted four sonar transducers at right angles to each other, connected to the emitter circuitry; and a wire stand on which the microphone is mounted,

facing the rear. The microphone stand was necessary in order to prevent the noise from the servos from triggering the behavior change.



Actuation

The actuation involved in the TJ platform is restricted to the wheels; each uses one hacked servo with a gearhead to simulate a DC motor. These are the standard 42 ounce-inch servos provided by Novasoft in an order placed in June of 1998; no manufacturer information was provided for these two servos. The wheels are 2.75 inch Dubro model aircraft wheels which are glued onto the servo horns.

The Talrik actuation involves two hacked servos (approximating DC motors) for the wheels and one unhacked servo for the head. The wheel servos are Diamond servos, purchased through Scott Jantz, chosen for their high output torque (42 ounce-inch) and for the fact that they incorporate bearings, which are needed to compensate for the weight of the platform. The wheels are 3 inch Dubro model aircraft wheels, mounted on the servo horns. The head servo is a Futaba 42 ounce-inch unhacked servo from Tower Hobbies, on which is mounted the standard semi-circular Talrik head.

Information on the TJ servo hack is available through the Mekatronics website.

The Diamond servo hack is simpler than that required for the older servos from Novasoft and proceeds as follows:

1) Set the bearings by striking the bottom of the servo (i.e. the side without the servo horn) against a fixed object (e.g. the lab bench). This will keep the bearing case from separating when the servo case is opened, which could result in loss of bearings.
2) Remove the servo horn and the four screws holding the case together.
3) Remove the top of the servo case. Cut the stop tab off the bottom-most gear and remove the plastic tab from the shaft; reassemble gears and replace top. (Note: the

first three steps are identical to the old servo hack described in the TJ or Talrik assembly manuals.)

4) Remove the bottom of the servo case. Pull out the circuit board and cut all traces. Remove the set screw and pull out the shaft.
5) From the three-wire connector (usually orange-red-brown), remove the orange wire. Solder the red and brown (power and ground) wires as shown in Figure 9.
6) Reassemble the servo case.

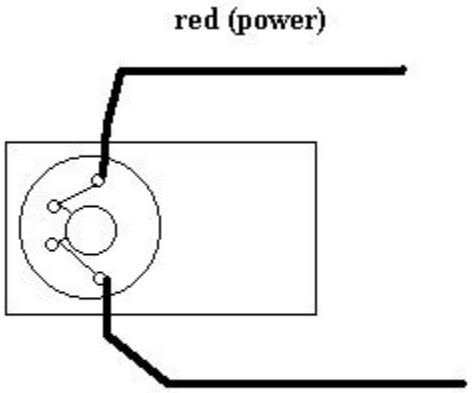


Figure 9

Sensors

IR Sensor Suite

Alph's IR sensor suite includes four IR emitters (see Appendix A for vendor information) evenly spaced across the front of the platform, and two IR detectors at ninety degrees from each other. The emitters and detectors each cover approximately 180 degrees of arc in front of the platform.

The IR detectors connect to the analog port, PE0 and PE1, on the 68HC11 EVBU board. The IR emitters are powered off a 40 kHz signal generated by dividing the 68HC11's E-clock with a 74HC390 decade counter on the ME11 expansion board.

Ralph's IR sensor suite consists of two IR emitters and two IR detectors in the front, and one emitter and two detectors in the back. The detectors cover 180 degrees to the fore and rear; the emitters cover approximately 180 degrees to the fore and 60 degrees to the rear.

The front left IR detector connects to PE6, the front right to PE7, rear left to PE5 and rear right to PE4, and the emitters are powered off a 40 kHz output from PB0.

The IR detectors for both Alph and Ralph are Sharp GPIU58Y IR detectors hacked, following the directions provided on the Mekatronics website, to yield an analog rather than digital signal.

Bump Sensor Suite

Alph's bump sensor suite consists of ten bump switches evenly spaced around its perimeter, connected by a rigid bumper. The sensors are wired such that the five in front act as a single sensor connected to PE6 on the 68HC11 EVBU board, as do the five in back, which are connected to PE5.

The bump sensors are active high: a “hit” has the value 255; otherwise the analog port reads a value of 0.

Figure 6 details the wiring involved. When the switch closes, PE6 receives an active high signal as the connection to five volts is completed.

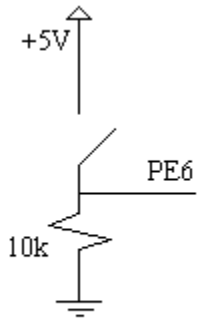


Figure 6

Ralph’s bump sensor suite consists of four bump switches: three in front and one in back, connected by a rigid bumper. The front three switches are wired in parallel to PC5, and the rear to PC0.

Audio Sensor Suite

The audio sensor suite consists of a microphone and amplifier circuitry on Ralph, and a 4.2 kHz audio buzzer on Alph.

Ralph’s microphone circuit (see Figure 4) functions as a “clapper,” such that a sound above the threshold determined by the comparator will instigate a certain behavior (in this case, Ralph’s holding pattern).

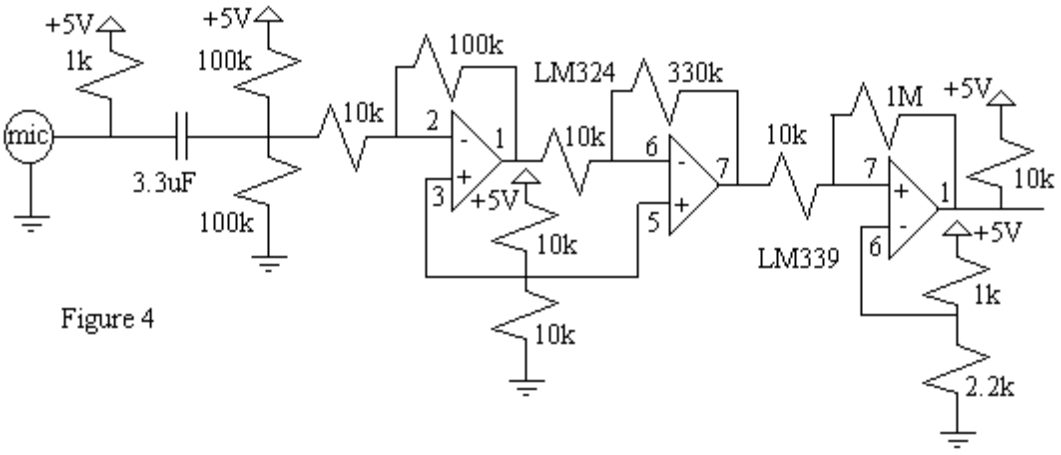


Figure 4

The condenser microphone used here yields a signal in the 10 mV range, which needs to be amplified in order to be used. A two-stage amplifier was constructed using a LM324 op-amp – two inverting amplifiers in series with a total gain of 330 – the output of which is then passed to a LM339 comparator, the output of which is then passed to PE1.

The voltage dividers seen in the circuit provides a virtual ground at 2.5 volts in order that the signal, when inverted by the amplifiers, is inverted with respect to 2.5 volts instead of shifted in its entirety about the 0 volt level.

The circuit diagrams provided by Radio Shack in the microphone packaging call for a 1k resistor and 1 to 10 uF capacitor. However, experimentation initially indicated that both capacitor values are too high to yield a signal. Ed Carstens' robot "Ziggy" (Fall '94) included a microphone circuit which used a 2.2k resistor and 0.001uF capacitor, values which I tried in my circuits with great success.

However, in order to improve the range from less than one centimeter, I found it necessary to experiment with these values. The best range was achieved with a 1k resistor and a 3.3 uF capacitor. This setup allows the microphone to pick up a handclap from several feet away, provided that there is no other interference.

Alph's 4.2 kHz piezo buzzer needed no other circuitry and is operated off port A pin 4.

Sonar Suite

The sonar suite consists of a sonar receiver and filter on Alph, a sonar emitter on Ralph, and a stationary sonar beacon.

The sonar emitter is constructed using a sonar transducer (see Appendix A for vendor information), an audio transformer, and a 2N222A transistor (see Figure 1). The original design was derived from Michael Apodaca's project Odin (Spring '98); however, it was determined that his orientation of the audio transformer was backward, producing a gain of approximately two instead of eight. The corrected circuit, seen in Figure 1, produces the necessary gain. Ralph's emitter is run off PB0, the same pin used for the IR emitters; the standard collision avoidance program written by Ivan Zapata (see Appendix B) includes a routine which produces a 3ms pulse every 20 ms at 40 kHz.

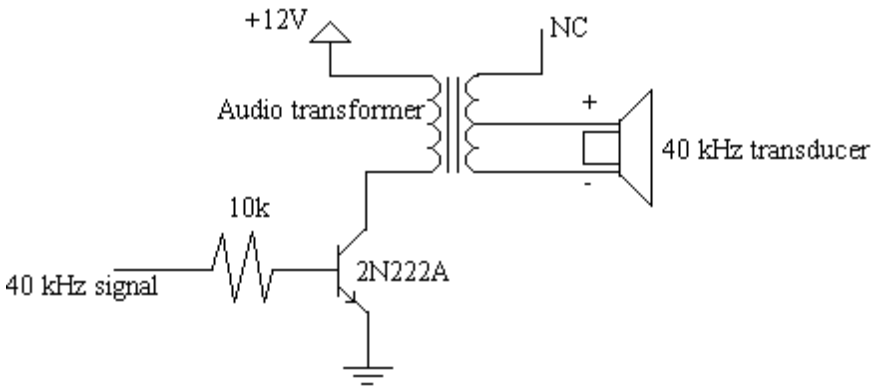


Figure 1: Sonar Transmitter

The stationary beacon (see Figure 3) incorporates an emitter and a 40 kHz oscillator constructed with an MC 1455 chip: a CMOS 555. The resistor between pins 2 and 7 is actually a 10k potentiometer screwed down to approximately 5k, in series with a 10k carbon-film resistor. Originally, the circuit used only the potentiometer; however, that

produced a frequency of 50 kHz instead of 40 kHz, which the receiver cannot pick up. With the resistance increased to approximately 15k, the oscillator produces a 40 kHz signal.

$$T=0.693(R_A + 2R_B)C$$

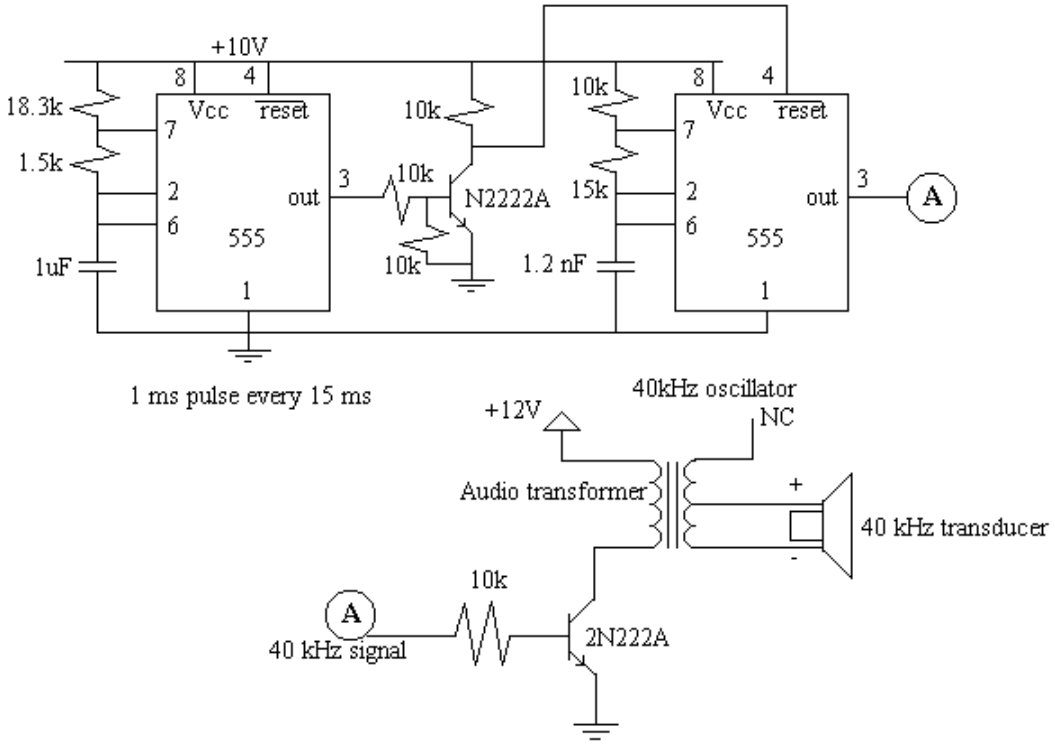
$$T=0.693(10k + 2(10k))C=1/40kHz=2.5x10^{-5}\ s$$

$$C=1.2nF=0.0012uF$$

As demonstrated in the above calculation, the appropriate result should have been obtained with the 10k resistor values and 0.0012uF capacitor; however, it was necessary to increase R_B to 15k.

This 40 kHz signal is then passed to another oscillator, also constructed with a 555 timer, which produces a 1ms pulse every 15 ms.

Figure 3



The sonar receiver (see Figure 2) was constructed using a 40 kHz transducer, LM339 comparator, and MAX 266 filter. It too was based on the design by Michael Apodaca; however, with the exception of substituting a 10k potentiometer for a 15k potentiometer at the positive input of the LM339 comparator, the circuit functioned properly as designed. The signal is received via the transducer, filtered by the MAX chip, and passed through the comparator, which renders a digital output. This output is then passed to PE3 on Alph's 68HC11 board.

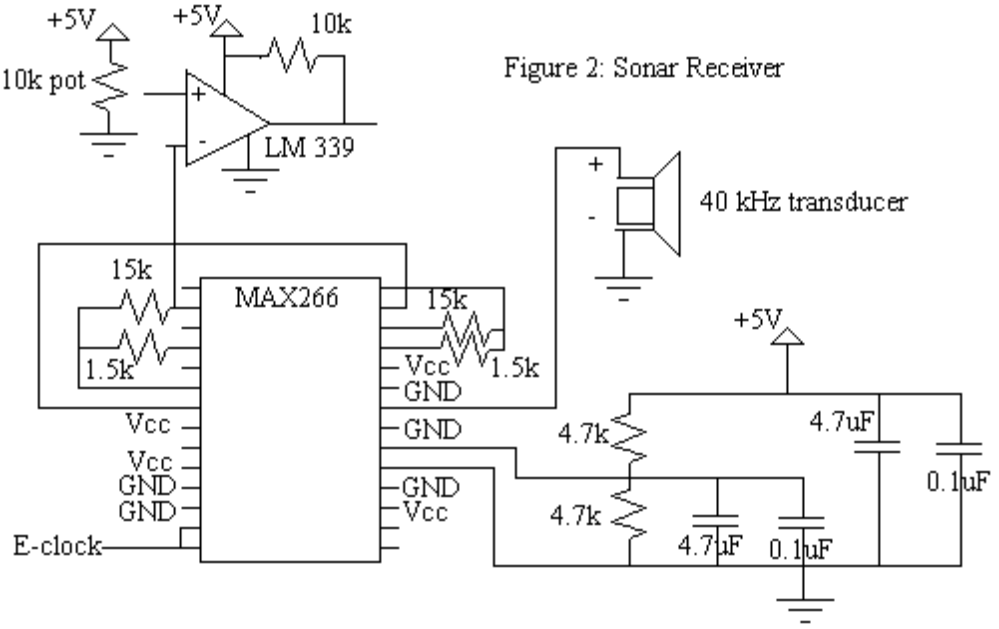


Figure 2: Sonar Receiver

Alph's transducer is mounted on a servo, giving it a 180-degree range of motion. The head will rotate in ten-degree increments across the full range of its motion, allowing the receiver to scan for a sonar pulse. Once this pulse is located, the robot will orient itself toward that signal and pursue it.

Sensor Integration

The sensor integration for Ralph was accomplished via polling. Provided that PE1 (the microphone) had not gone high, the IR detectors were polled, as were the fore and aft

bumpers. A high value on the microphone would trigger the holding pattern; a high value on the front bumper would cause Ralph to back up and turn before continuing; and an IR value above the threshold (here, 100) would cause Ralph to turn away from that heading. Similarly, Alph's sensor integration was accomplished via polling. As previously discussed, the IR and front bumper were integrated to perform collision avoidance. If at any time an obstacle was detected, despite the presence of a sonar signal, the platform would move to avoid the obstacle before returning to its sonar sweep. The behavior arbitration was also integrated with the collision avoidance such that, at any time during the collision avoidance routine, Alph's behavior could be toggled to "sonar tracking mode" by a high value from the back bumper (PE5).

All sensor range data and graphs are available in the Experiments section.

Behaviors

The basic behaviors for Ralph include collision avoidance; wandering; and responding to an audio signal. The integration of these behaviors should result in Ralph trying to avoid Alph as the latter tries to herd it.

The behaviors for Alph include collision avoidance; scanning for sonar; time-of-flight measurement and response; and behavior arbitration in response to sensor input. Collision avoidance for both platforms is achieved primarily with the IR systems, using the bump switches as backup.

Alph's IR emitters are turned on at the beginning of the main routine and remain functional throughout the operation of the robot. The LEDs are positioned to provide a 180 degree arc of illumination in front of the robot. The detectors each cover one forward quadrant, such that, should either detect an IR value above the given threshold (see code in Appendix B), the platform will turn in the other direction.

Alph's behavior arbitration is based on a positive signal from the back bumper. When the back bumper pin (PE5) goes high in response to a hit, Alph will switch from collision avoidance and wandering to scanning for sonar.

In the scanning routine, Alph's sonar receiver sweeps out a 180 degree arc in 10-degree increments, covering the area in front of the platform. When a signal is detected, Alph turns toward it and continues scanning. The collision avoidance and sonar scanning are integrated, such that Alph will avoid an obstacle even if that means turning away from a sonar signal.

Alph's time-of-flight interpolation routine is triggered as soon as it detects a signal. Since the receiver circuit is designed to provide an active low signal in which the low

pulse is proportional to the distance between the receiver and emitter, Alph can determine its approximate distance from the signal by counting the length of the low pulse. If this low pulse indicates a separation distance of less than two feet, Alph will emit a half-second pulse on the 4.2 kHz buzzer (theoretically, in order to signal Ralph to stop) before pausing for one second, backing up and turning away from the signal, and continuing its search in a different area.

Ralph's collision avoidance is based on both the bumper and the fore and aft IR detectors. The rear IR detectors were mounted specifically to detect the large IR signature of Alph's array of emitters, and to turn away from Alph depending on its position.

Ralph's holding pattern is triggered by a high pulse on PE1 via the microphone, which has a range of several feet for a handclap or other similarly loud noise. Upon recognition of this signal, Ralph will begin spinning in place. This behavior has the added benefit of providing a 360-degree sweep of sonar, guaranteeing that Alph will be able to "see" it.

Experiments

The following experiments were conducted on Alph.

Infrared Array

In order to determine the range and sensitivity of the IR arrays, I set up a series of obstacles and took readings from the analog ports at varying distances.

Distance (inch)	PE0: white shirt	PE0: my hand	PE1: white shirt
1		116	125
2		109	122
3	124	94	113
6	105	88	109
9	98	87	106
12	94	87	103
Infinity	87	87	90

Further tests with a sheet of white paper yielded a maximum of 125 at two inches and a minimum of 87 beyond twelve inches.

The graph “IR Range Data” shows a decaying exponential relationship between the distance to an obstacle and the analog value returned by the IR detector. Therefore, once a reflective object (i.e. one that does not absorb IR) is within two inches, the detectors saturate. Accordingly, I set my threshold at 98, at which point the obstacle is approximately nine inches away, giving Ralph sufficient clearance to turn and avoid it. I also wrote a simple test program (see Appendix B) for collision avoidance, which functioned admirably.

Bump Sensors

To test the bump sensors, I took data from the analog ports PE6 and PE7 before, during, and after the switches were activated. I was thus able to confirm that the switches are

active high, and to establish that the values do not register above 120 unless a switch has been closed. (Note: the rear bumper was later switched to PE5.)

I then wrote a test program by which the platform would move forward until a hit was registered, at which point it would pause. This confirmed that the sensors were functioning properly and that the platform would be able to respond appropriately.

Sonar Array

Experiment 1: In order to determine the correlation between the length of the low pulse and the distance between the sonar emitter and receiver, I wrote a simple IC program (see Appendix B) which activates a counter as soon as the sonar input goes low. The number of counts is therefore related to the separation distance.

From the "Sonar Range Data" graph, it is possible to see that there is a linear relation between the distance and low pulse length. While not precisely one-to-one, the relation is nonetheless sufficiently linear that, particularly in ranges of less than two feet, this counter program can be used to allow Alph to determine whether it is close enough to the beacon to signal Ralph to stop.

Distance (ft)	Pulse length
1	0,1
1.5	2
2	2
2.5	2,3
3	3
3.5	3
4	3,4
4.5	3,4
5	3,4
5.5	5,6
6.5	5,6
7	5,6
7.5	5,6

This experiment was set up as follows: the beacon was set in a small vise clamp, such that the emitter emitted horizontally. The receiver was clamped in a "third hand" such that it was on the same level as the emitter, and facing the emitter as directly as possible. The beacon was then moved along the table in increments of six inches to take data points. (There is no value for six feet because the table ended at that point and there was a gap between it and the next level surface.)

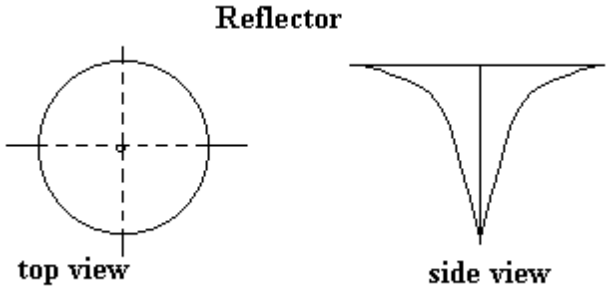
The values do not extend past seven and a half feet because the beacon was, at the time, powered off the EVBU board's five volt supply instead of a separate power supply, and the extension cord was only seven and a half feet long. However, I was able to obtain values up to approximately sixteen feet by bouncing the signal off the wall. I did not plot this data because it was erratic – focusing the emitter and receiver on the same spot was an inaccurate science at best – and because I did not use the sonar for wall-following or mapping, but for tracking and rough time-of-flight approximations.

Experiment 2: With the use of a simple routine (see Appendix B) in which Alph moves forward upon receiving a sonar pulse and stops when the signal is terminated, I was able to determine the approximate arc of detection and emission for the sonar transducers. Using the stationary beacon (Figure 3), the receiver has a range of vision covering approximately 60 degrees. The results are most consistent within a cone of about 45 degrees.

The following experiments were performed on Ralph.

Sonar Distribution

As previously noted, the sonar emitters have a range of approximately sixty degrees. In order to spread the signal in all directions such that Alph's receiver would be able to detect it even when not facing the emitter, I experimented with an inverse parabolic reflector, as suggested by Scott Jantz.



I constructed a rough approximation of the reflector diagrammed above from tinfoil, selected because of its reflective properties, malleability, and availability, and determined that such a reflector would indeed provide the desired response; namely, the sonar receiver could pick up a signal even when perpendicular to the plane of emission.

The experiment was performed simply by holding the cone over the emitter such that the point almost touched the center of the transducer, and moving the receiver until it

detected a signal (as seen on an oscilloscope). No hard data was collected because this was merely an experiment to determine whether the theory was sound.

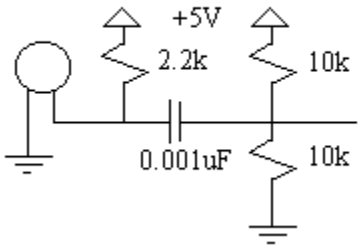
This reflector design was not implemented due to time constraints as well as the difficulty inherent in minimizing flaws in the surface of the reflector that might negatively affect the propagation of the signal. It should be noted that the time-of-flight measurements taken when using the reflector were erratic, probably due in part to the extremely crude design.

Microphone Range

The microphone circuitry went through several iterations before its final realization in the circuitry seen on Ralph (see Figure 4). Initially, the intention was to incorporate audio sensors on both platforms; however, due to time constraints and the problem of interference from the servos in the form of noise, only Ralph has a microphone.

The primary concern in implementing the audio circuitry was providing sufficient range that it would be feasible. The initial circuitry used a 2.2 k resistor and a 0.001 uF capacitor; however, this resulted in a range of less than one centimeter for the 4.2 kHz piezo buzzer.

original microphone circuit



By replacing the 0.001 uF capacitor with a 0.1 uF capacitor, I was able to increase the range to approximately two inches. However, increasing the capacitor value to 1 uF and 10 uF had no further effect on the range.

Changing the resistors in the voltage divider to 100k from 10k increased the range slightly further, to approximately three inches, but also decreased the amplitude.

By then increasing the capacitor value to 3.3 uF, I was able to increase the range to approximately six inches (for the piezo buzzer); changing the pull-up resistor value to 1k instead of 2.2k further increased the range and sensitivity.

With the amplifiers (see Figure 4) but without the comparator, I obtained the following data:

Distance (inches)	Amp (V) : buzzer	Amp (V) : handclap
6	2.5	4
9	1.25	4
12	0.75	4
24		4
36		4

The handclap or other loud noise caused a consistent reading even at a maximum distance of nine feet. The response to the piezo buzzer, however, displayed an approximation of a decaying exponential curve.

The addition of a lowpass filter with a center frequency of 5 kHz and a highpass filter with a center frequency of 3.5 kHz did not improve the range, nor did the filters remove the noise from the servos, which, according to oscilloscope readings, has a large component around 4 kHz.

Further experimentation involved the interaction between Alph and Ralph and the myriad software iterations necessary to accomplish this.

Conclusion

In summary, I was able to successfully integrate Alph's collision avoidance, sonar tracking, and time-of-flight interpolation routines. Alph was capable of finding and following Ralph or finding and approaching the beacon; most of the time, Alph would also stop and turn away before colliding with either. Ralph's evasion of pursuit was almost too successful, as was its response to audio signals.

Observation in different environments – specifically, the hallway outside Benton 312, the IMDL lab, and the demonstration arena – indicates that this particular system functions best in a small, well-defined area with few irregular corners and narrow (i.e. invisible to IR) obstacles.

There are several areas in which improvement is possible. The microphone circuitry never worked as well as was hoped; when transferred from the prototype board to a breadboard, the range of the microphone was inevitably halved, at least, despite intensive efforts to correct this. Further, it was discovered that Ralph's servos squeak at almost exactly 4 kHz, thereby making attempts to filter out the servo noise almost entirely futile.

Also in need of improvement is the sonar tracking routine. Provided that the target is stationary or moves slowly, Alph can find and track it without excessive difficulty. However, there was a distressing tendency to lose Ralph before (and, in fact, even after) Ralph's velocity was slowed (due to the servos not being balanced, Ralph's current programming causes it to move in a circle).

Finally, the actual herding behavior needs to be implemented. Experimentation with this behavior was unsuccessful to date and so could not be demonstrated. This is due in large

part to the failure of the pulse differentiation routine, the implementation of which was most likely inherently flawed.

The principle areas in which I would particularly like to do further work are the audio system and herding behavior. Greater range and better sensitivity are required for the microphone to actually pick up the “stop” signal from Alph without constantly being set off by the noise from the servos.

The herding behavior is feasible; however, rather than having Alph look for two different sonar signals and try to arbitrate between the two, I would keep the sonar for tracking the beacon and use CDS cells to look for a light to be placed on Ralph. In this way, the time-of-flight precision would be available with respect to the beacon while simplifying the algorithm considerably. The biggest problem involved in this project was the attempt to discern between the 1ms pulse from the beacon and the 3ms pulse from Ralph. If two different systems were assigned to the two tasks of finding the beacon and finding Ralph, the problem of interference would be nonexistent.

References

It should be noted that this project could not have been accomplished without assistance from the following sources:

- the Mekatronics website, from which I downloaded the assembly manuals for the TJ and Talrik platforms
- Scott Jantz, Aamir Qaiyumi, Ivan Zapata, and the other members of the MIL who were kind enough to contribute suggestions and assistance
- My classmates, who also provided suggestions, assistance, and inspiration
- Michael Apodaca's project Odin
  (documentation available on the IMDL website under Spring 1998)
- Ed Carsten's project Ziggy
  (documentation available on the IMDL website under Fall 1994)

Appendix A:
Vendor Information

IR emitters/detectors:

- LED emitters:
  Mekatronics
  316 NW 17th St., Suite A
  Gainesville, FL 32603
  (407) 672-6780
  http://www.mekatronics.com
  (purchased from Scott Jantz)
  $0.75 each

- Detectors:
  Sharp GPIU58Y via Mekatronics
  (purchased from Scott Jantz)
  $3.00 each

Bump switches:

Mekatronics
(purchased from Scott Jantz)
$0.75 each

Sonar:

- transducers:
  Electronic Goldmine
  P.O. Box 5408
  Scottsdale, AZ 85261
  (602) 451-7454
  http://www.goldmine-elec.com
  part #G2528
  $1.50 each

- audio output transformer:
  Radio Shack (local store)
  part # 273-1380
  $2.00 (approximately)

- MAX chip:
  Maxim Integrated Products
  120 San Gabriel Dr.
  Sunnydale, CA 94086
  (408) 737-7600
  http://www.maxim-ic.com
  Part # 266
  Free sample of two

Microphone:

- microphone: Radio Shack, part # 270-090c
  $1.29

- 4.2 kHz piezo buzzer: Radio Shack, part # 273-074a
  $1.00 (approximately)

Appendix B:
Code

```
/* Megan Grimm with the invaluable assistance of Scott Jantz */
/* 10/6/98 */
/* time-of-flight calibration for Alph's sonar */

int counter;

void main()
{
   init_serial();
   counter=0;
   while(1)
   {
   counter=0;
      poke(0x7000,0xff);
      msleep(1L);
      poke(0x7000,0x00);

   while((analog(3)>200)&&(counter<1000))
   {

      counter=counter+1;

   }
   write_int(counter);
   }
}
```

```
/* Sonar demo */
/* Megan Grimm */
/* 10/21/98 */

int counter;

void main()
{

      motor(0,0.0);
      motor(1,0.0);

      counter=0;

      while(1)
      {

      counter=0;

      while((analog(3)>200)&&(counter<1000))
      {
            counter=counter+1;
      }
      if (counter<6)
      {
            motor(0,100.0);
            motor(1,100.0);
      }
      else
      {
            motor(0,0.0);
            motor(1,0.0);
      }
}
}
```

```
/* bumper program */
/* on=255, off=0 */

int front_bump, back_bump;

/* front_bump=analog(7); */
/* back_bump=analog(6); */

void main()
{
      motor(0,0.0);
      motor(1,0.0);
      motor(0,100.0);
      motor(1,100.0);

      while(1)
      {

            /* if it hits something, back up */
            /* else keep going fwd */

            if (front_bump>200)
            {
                  motor(0,-100.0);
                  motor(1,-100.0);
                  wait(50);
            }
            else
            {
                  motor(0,100.0);
                  motor(1,100.0);
            }

            /* if it backs into something, move fwd and turn slightly*/
            /* else keep going fwd */

            if (back_bump>200)
            {
                  motor(0,50.0);
                  motor(1,100.0);
                  wait(50);
            }
            else
            {
                  motor(0,100.0);
                  motor(1,100.0);
            }
      }
}
```

```
/* collision avoidance */
/* Megan Grimm */
/* 9/16/98 */

void main()
{
      poke(0x7000,0xff);
      motor(0,100.0);
      motor(1,100.0);

      while(1)
      {
            if (analog(0)>98)
                  motor(1,-100.0);
            else motor(1,100.0);

            if (analog(1)>98)
                  motor(0,-100.0);
            else motor(0,100.0);
      }
}
```

```
/* Demo 1 */
/* if sonar within six feet, move toward it */
/* else stop */
/* Megan Grimm */
/* 10/21/98 */

int counter;

void main()
{
      motor(0,0.0);
      motor(1,0.0);
      counter=0;
      while(1)
      {

      counter=0;

      while((analog(3)>200)&&(counter<1000))
      {
            counter=counter+1;
      }
      if (counter<6)
      {
            motor(0,100.0);
            motor(1,100.0);
      }
      else
      {
            motor(0,0.0);
            motor(1,0.0);
      }
}
}
```

Alph’s Code

```
int sonar, sonarcount;
int counter, timer, distance;
int leftsonar, rightsonar, fwdsonar;
int frontbump;
float x, y, z, angle;

void haltenzie()
{
 motor(0,0.0);
 motor(1,0.0);
}

void backup()
{
 motor(0,-60.0);
 motor(1,-60.0);
}

void backturn()
{
 backup();
 wait(500);
 motor(0,0.0);
 motor(1,-60.0);
}

void go_straight()
{
 motor(0,60.0);
 motor(1,60.0);
}

void go_left()
{
 motor(1,0.0);
 motor(0,60.0);
}

void go_right()
{
 motor(1,60.0);
 motor(0,0.0);
}

void signal()
{
 poke(0x1000,0x10);     /* buzzer on PA4 */
 wait(800);
 poke(0x1000,0x00);
}

void poll_sonar()
{
 sonarcount=0;
 sonar=0;
```

```
 for (counter=0; counter<=200; counter=counter+1)
 {
      if (analog(3)<255)
            sonarcount=sonarcount+1;
 }
      if (sonarcount>1)
            sonar=1;
}

void too_close()
{
 signal();
 haltenzie();
 wait(1000);
 signal();
 backturn();
 go_straight();
}

void sonar_left()
{
 servo_deg(55.0);
 go_left();
 poll_sonar();
 time_of_flight();
 if (distance<2)
      too_close();
 else if (sonar>0)
      go_straight();
 else scan();
}

void sonar_ahead()
{
 servo_deg(55.0);
 go_straight();
 poll_sonar();
 time_of_flight();
 if (distance<2)
      too_close();
 else if (sonar>0)
      go_straight();
 else scan();
}

void sonar_right()
{
 servo_deg(55.0);
 go_right();
 poll_sonar();
 time_of_flight();
 if (distance<2)
      too_close();
 else if (sonar>0)
      go_straight();
 else scan();
```

```
}

void scan()
{
 servo_deg(0.0);
 for (x=0.0;x<=180.0;x=x+10.0)
 {
      poll_sonar();
      time_of_flight();
      if (distance<2)
            too_close();
      else if (analog(5)>200)
            arbitrate();
        else if (analog(6)>200)
            backturn();
      else if (analog(0)>98)
            go_right();
      else if (analog(1)>98)
            go_left();
      else if ((sonar>0)&&(x<50.0))
            sonar_left();
      else if ((sonar>0)&&(x>60.0))
            sonar_right();
      else if((sonar>0)&&(x<60.0)&&(x>50.0))
            sonar_ahead();
      else go_straight();
      servo_deg(x);
 }

 for (x=180.0;x>=0.0;x=x-10.0)
 {
      poll_sonar();
      time_of_flight();
      if (distance<2)
            too_close();
      else if (analog(5)>200)
            arbitrate();
        else if (analog(6)>200)
            backturn();
        else if (analog(0)>98)
            go_right();
      else if (analog(1)>98)
            go_left();
      else if ((sonar>0)&&(x<50.0))
            sonar_left();
      else if ((sonar>0)&&(x>60.0))
            sonar_right();
      else if ((sonar>0)&&(x<60.0)&&(x>50.0))
            sonar_ahead();
      else go_straight();
      servo_deg(x);
 }
}

void time_of_flight()
{
 if (analog(3)<255)
```

```
      timer=0;
 else timer=100;

 while ((analog(3)<200)&&(timer<1000))
 {
      servo_deg(x);
      timer=timer+1;
 }
 distance=timer;
}

void coll_avoid()
{
 servo_deg(55.0);

 if (analog(6)>200)
 {
      backup();
        backturn();
 }
 else if (analog(0)>98)
      go_right();
 else if (analog(1)>98)
      go_left();
 else if ((analog(0)>98)&&(analog(1)>98))
      backturn();
 else go_straight();
}

void arbitrate()
{
 while(1)
 {
 if (analog(6)>200)
         backturn();

 if (analog(5)>200)
 {
      signal();
      time_of_flight();
      if (distance<2)
            too_close();

      if (analog(0)>98)
            coll_avoid();
      else scan();

      if (analog(1)>98)
            coll_avoid();
      else scan();
 }
 else coll_avoid();
 }
}

void main()
{
```

```
 poke(0x7000,0xff);

 timer=100;

 servo_on();
 servo_deg(55.0);
 go_straight();

 start_process(arbitrate());
}
```

Ralph’s code

```
/* modified 11/23/98 by Megan Grimm */
/* in her ongoing and futile attempt to get SOMETHING to work!! */

/*****************************************************************
* Title           avoid.c
* Programmer      Ivan Zapata
* Date            November, 1996
* Version         1
* Description     A very simple collision avoidance program.
*                 TJ will read each IR detector, and turn away from any
*                 obstacles in its path. Also, if something hits TJ's
*                 bumper it will back up,
*                 turn, and go on.
*****************************************************************/

/***************************** Includes ******************************/

#include <servotj.h>
#include <irtj.h>
#include <analog.h>
#include <vectors.h>
#include <serial.h>
/************************ End of includes ****************************/

/***************************** Constants *****************************/
#define FORWL  2500     /* was 2000 */
#define FORWR  3700     /* was 4000 */
#define BACKL  3500     /* was 4000 */
#define BACKR  2300     /* was 2000 */
#define STOP   3000
#define LEFT_SERVO 0
#define RIGHT_SERVO 1
/************************ End of Constants ***************************/

/***************************** Prototypes ****************************/
void turn(void);
void terminate(void);
/************************ End of Prototypes **************************/

/***************************** Globals *******************************/
int rv, lv;
/************************** End of Globals ***************************/
int main(void)
/******************************* Main ********************************/
{
  int rval, lval;
  int i;

  int micval;
  int brval, blval;

  init_servos();  /* Initialize necessary registers, etc. */
  init_analog();
  init_ir();
  init_serial();
```

```
  DDRC = 0x00;           /* Set PORTC (bumper) for input */

  while(1)
  {

/* The following block will read the ir ports, and decide whether    */
/* TJ needs to turn to avoid any obstacles                           */
    rval = ir_value(6);
    lval = ir_value(7);

    brval= ir_value(4);
    blval= ir_value(5);

    micval = analog(1);

    write_int(rval);
    write_int(lval);
    write_int(brval);
    write_int(blval);

    if (micval > 200)
        terminate();
    else
    {
     if ((rval > 100)||(brval > 100))
       rv = BACKR;
     else if ((lval > 100)||(blval > 100))
       lv = BACKL;
     else
     {
        lv = FORWL;
        rv = FORWR;
     }
    }

    servo(RIGHT_SERVO, rv);
    servo(LEFT_SERVO, lv);

/* This "if" statement checks the bumper. If the bumper is pressed, */
/* Tj will back up, and turn.                                       */

    if(PORTC & 0x20)
    {
      servo(LEFT_SERVO, BACKL);
      servo(RIGHT_SERVO, BACKR);
      for(i = 0; i < 25000; i++);
      turn();
    }
  for(i = 1; i < 2500; i++);
  }
}
/*************************** End of Main ***************************/

void turn()
/*****************************************************************
* Function:  Will turn in a random direction for a random amount of
```

```
* time                                                                  *
* Returns:  None                                                        *
* Inputs                                                                *
*   Parameters: None                                                    *
*   Globals:    None                                                    *
*   Registers:  TCNT                                                    *
* Outputs                                                               *
*   Parameters: None                                                    *
*   Globals:    None                                                    *
*   Registers:  None                                                    *
* Functions called: None                                                *
* Notes:                                                                *
************************************************************************/
{
 int i;
 unsigned rand;

 rand = TCNT;

 if (rand & 0x0001)
 {
   servo(RIGHT_SERVO, FORWR);
   servo(LEFT_SERVO, BACKL);
 }
 else
 {
   servo(RIGHT_SERVO, BACKR);
   servo(LEFT_SERVO, FORWL);
 }
 for (i = 0; i < rand; i++);
}

/* void terminate()

        puts Ralph into a holding pattern */

void terminate()
{
 while(1)
 {
        servo(RIGHT_SERVO, BACKR);
        servo(LEFT_SERVO, FORWL);

 }
}
```