Final Report
Archer

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

Name: Ian St.John
Date: 4/23/02
TAs: Aamir Qaiyumi
Uriel Rodriguez
Instructor: A. A Arroyo

Table Of Contents

Abstract

The system I am designing will consist of two functional units: an immobile target and a mobile robot. The mobile robot will seek out an immobile target platform, move into an optimal firing position and fire a projectile at it. The robot will locate the platform by use of a rudimentary GPS system implemented using two IR beacons mounted on the target platform.

Executive Summary

Archer is a navigational robot designed to move around a target beacon, and fire on a target placed at a known position relative to the beacon.

Archer samples two IR frequencies output by the target platform in order to determine it's position. It also uses the IR detector to perform rudimentary obstacle avoidance. Once Archer has determined its position, it can then navigate to the target and fire at it. Firing is achieved by use of an interfaced electric foam disc launcher.

The microprocessor used for controlling the mobile robot is a MC68HC11 installed on a Mekatronix MRC11 board with the MRSX01 sensor expansion board. The control circuitry for the target beacon is achieved by a 555 and 556 based digital circuit. To provide feedback to the user, a seven segment LED array was interfaced.

Introduction

The government currently maintains a series of satellites in orbit around the planet that can be used for navigational purposes. The system is called the Global Positioning System, or GPS for short. By synchronizing a receiver on the surface to 3 or more of the satellites, precise positional data can be obtained. Although the system is capable of a high degree of precision the government distorts the signal so that the accuracy is taken down to about ±100 meters for security purposes.

My robot will implement a scaled down version of this GPS system using IR. In order to implement the IR beacon system, I will be using two separate IR systems mounted on a stationary platform. One will be rotating at a fixed velocity, and another that pulses on when the rotating emitter reaches a fixed point. By measuring the phase difference between these two signals the mobile robot will be able to determine its exact position. The actual mechanics of this system will be explained in detail within this report.

Integrated system

The main robot will be equipped with a mobility and collision avoidance system, a separate modulated IR emitter, two sonar receivers, and a projectile launcher. The target will be equipped with two IR emitters, and a servo(to provide rotation). The interaction of these systems is illustrated in figure 1 below:

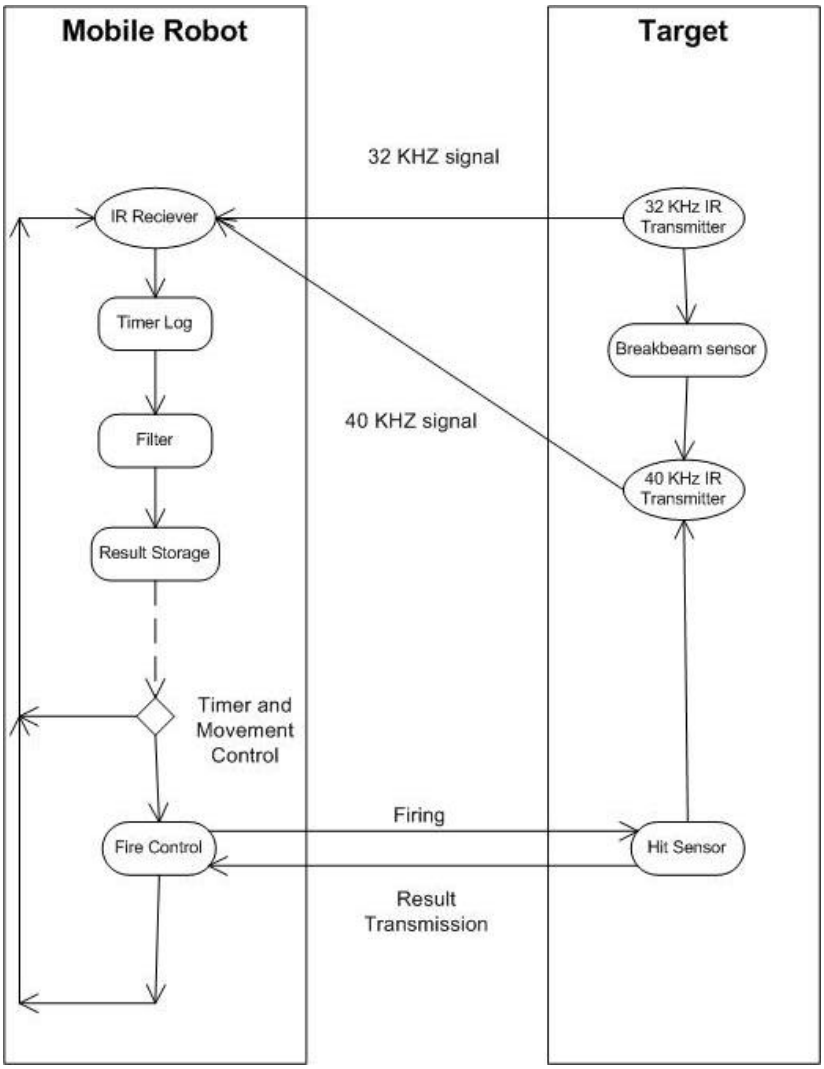


Figure 1

As we can see in figure 1, the mobile robot receives two pulses, one modulated at 32kHz, and another modulated at 40kHz. When the mobile robot receives a signal pulse it records its internal timer's value. Once it receives enough pulses, these values can then be used to calculate the position of the mobile robot. The mobile platform will then turn, and move into the optimal firing position. Once the robot is in an optimal firing position it will launch a projectile at the target. If the target was not hit, the mobile platform will turn a little bit and fire again. The robot will repeat the above procedure until a hit is scored. Once a hit has been scored, the robot will stop firing and do a victory dance.

Mobile Platform

The mobile robot will require a platform large enough to support the battery packs, the microcontroller and sensor boards, the disk launcher, and the IR receivers. To meet these requirements, I am going to be using a modified Talrik platform. The base of the platform will be only slightly modified to remove a few unneeded holes. The upper platform will interface with my gun mount(see fig. 2a below).

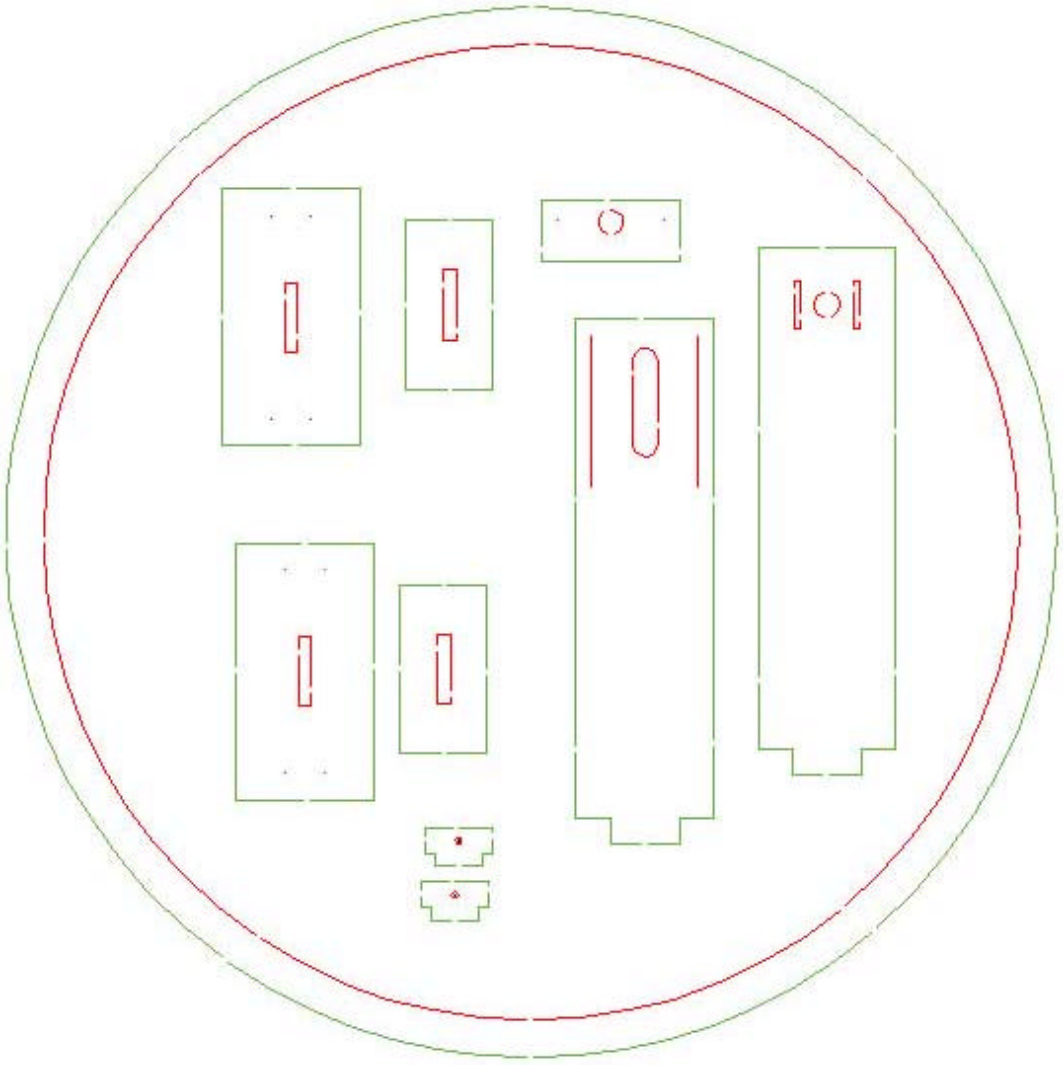


Figure 2a: Gun Mount and Bumper Design

The gun is held in a cradle shown in fig 2b below. The cradle is designed to connect to the gun mount via a ¾ inch dowel inserted through the mounting holes.

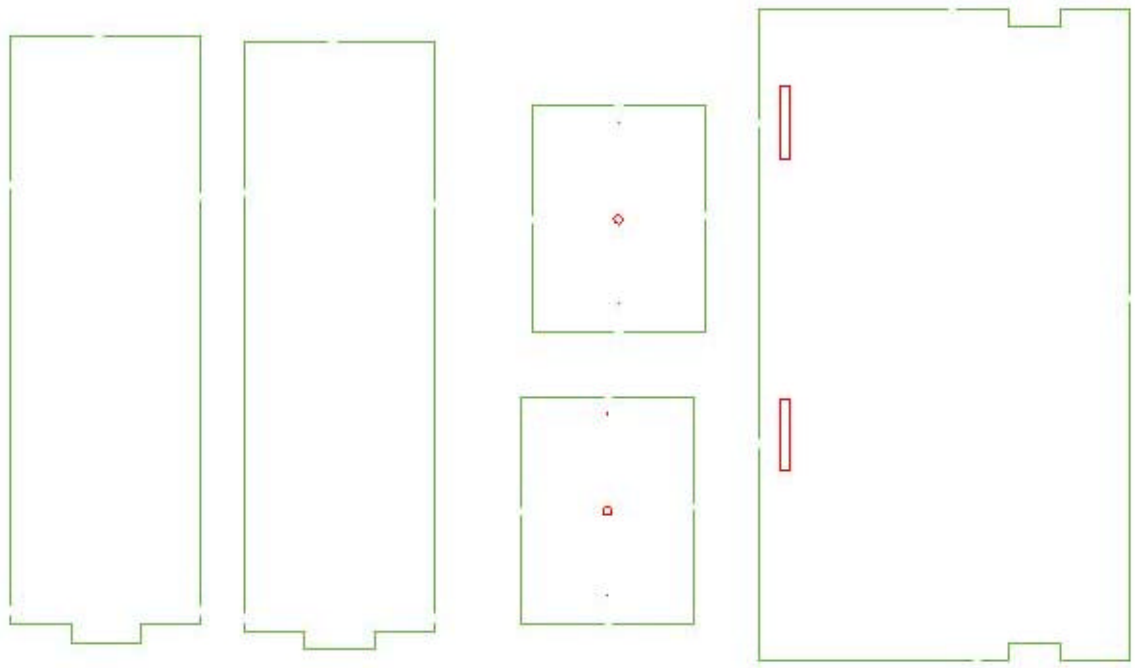


Figure 2b: Gun Cradle Design

The gun mount is designed to include a vertical and horizontal adjustment such that the gun can be moved slightly to ensure that it is level so as to provide a straight firing pattern. In addition to these concerns, it must also be ensured that the center of balance of the robot remains close to a point equidistant from both wheels so that the robot can travel in a straight line without any excessive artificial assistance. The fully assembled gun mount is shown below in figure 2c.

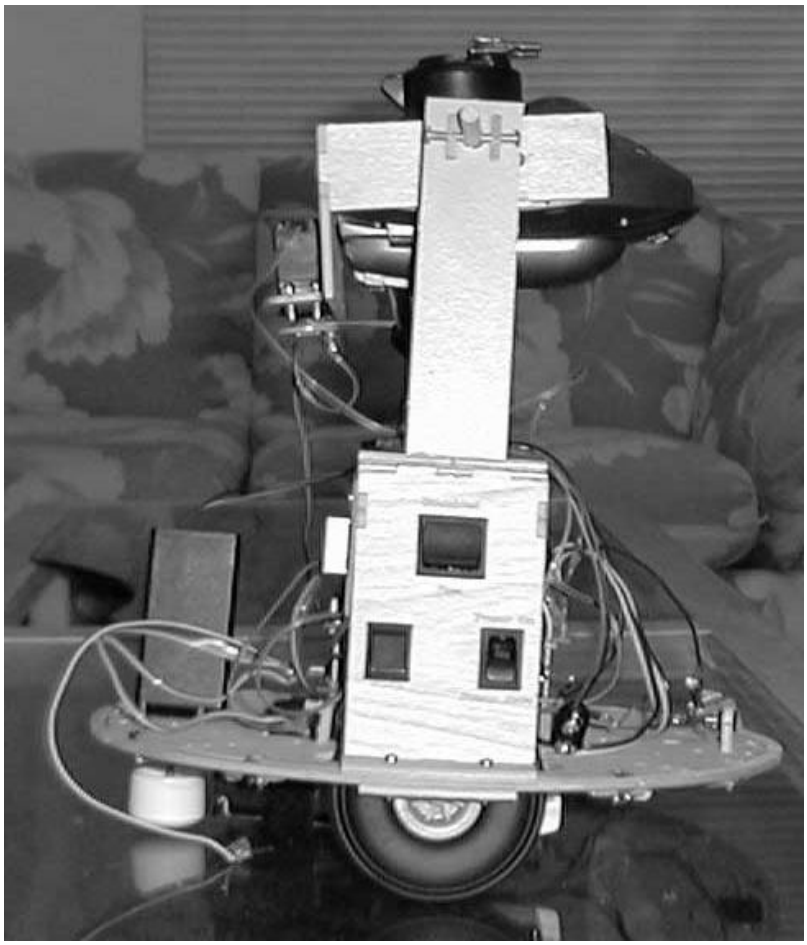


Figure 2c: Gun Cradle Design

The target platform is constructed from two pieces of scrap wood separated by metal spacers. The servo is mounted upside down so as to allow the emitter board to rotate freely. The multidirectional emitter is mounted on the lower platform.

Actuation

The operation of the mobile robot will require three servos. Two of these servos will be used as motors for the robot, and one will be used to pull the trigger of the disk launching gun. The operation of the target platform requires one servo to provide that rotation for the rotating emitter. All four of these servos will be modified for continuous rotation. The servos will not be particularly heavily loaded so I will be using HiTech HS-300 servos which have a rated torque of 42 oz/in.

The mounting of the servos that will drive the robot's wheels is fairly straightforward. They will simply be attached directly to the underside of the robot platform and supported by a series of wooden support rods. They will be attached to the wheels by epoxy. The servo used to fire the disk launcher will be attached to the trigger via a length of wire attached to a spindle mounted to the servo. When the servo rotates it will wrap the wire about the spindle. This will produce a pulling motion on the trigger, and the gun will fire. The trigger can then be returned to the ready position by rotating the servo in the opposite direction. The servo/spindle combination will be mounted on the robot platform behind the trigger assembly. The servo horn on the target platform will simply be screwed onto the platform itself.

Sensors

Figure 4 below shows a breakdown of the sensor systems of the mobile robot platform:

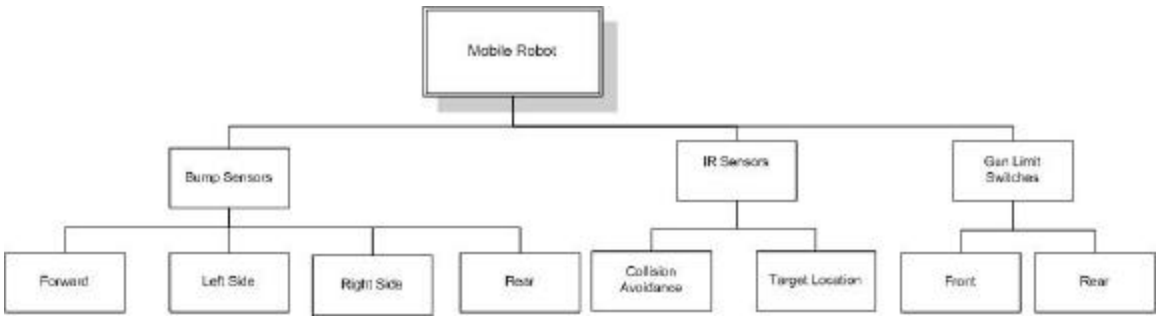


Figure 4

As you can see in figure 4, the sensor suite consists of three main units: the bump sensors, the IR sensors, and the limit switches mounted on the gun. The bump sensors, as their name implies, detect when the robot has bumped into something which represents a failure of the IR system to detect an obstacle. There are two main components of the IR system, a collision avoidance system and a target location system. The collision avoidance system works by broadcasting a 40kHz IR signal and checking to see, with a IR detector designed to be sensitive solely to 40KHZ IR signals, if the signal is reflected back by an object. The target locator system periodically emits a 32kHZ IR signal and a 40kHz IR signal that is received by the target's system. It should be noted that the collision avoidance system uses the same IR frequency as the target beacon. This is not a problem as when the robot is receiving the IR pulses from the target platform, the collision avoidance system is not needed, and is therefore

disabled. The limit switches were added to the gun after it was discovered that due to frictional forces acting on the trigger assembly, the time it takes to pull back and return the trigger is indeterminate. Now the trigger is pulled back until the rear limit switch is pressed, and trigger is released until the forward limit switch is pressed.

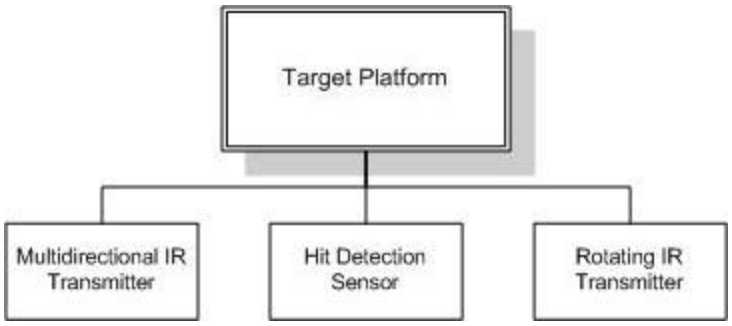


Figure 5

Figure 5 above shows a breakdown of the sensor systems of the target platform. The target platform has three main sensor systems: a multidirectional IR transmitter, a hit detection sensor, and a rotating IR transmitter.

The goal of the sensors discussed above is to enable the mobile robot to navigate to a known position relative to that of the target platform. The system will accomplish this by using a form of differential GPS. The target platform will utilize two IR emitters. One will consist of a single IR emitter rotating at a constant speed and constantly broadcasting. The other will consist of many IR emitters facing in all directions which will only be activated when the rotating emitter passes by a specific point. The point at which the multidirectional IR system will activate will be denoted as 0°. By measuring the difference in time

between when the rotating IR signal is detected, and when the multidirectional system's IR is detected the mobile platform's position can be determined relative to the 0° point. This can be accomplished because the rotation speed of the IR emitter is known. For example, the single direction IR emitter is rotating at a speed of 1° per second. The multidirectional signal is detected at t=5, and the single direction(rotating) signal is detected at t=30. Therefore the mobile platform is located at a position of (30-5)=25° from the 0° position of the target platform.

Sensor Design

Target Platform

As you can see from figure 1b, the target platform will require a total of three control systems. The first will control the rotation of the directed IR beam. The second will control the IR frequency of the single direction emitter. The third will control the IR frequency of the multi direction emitter.

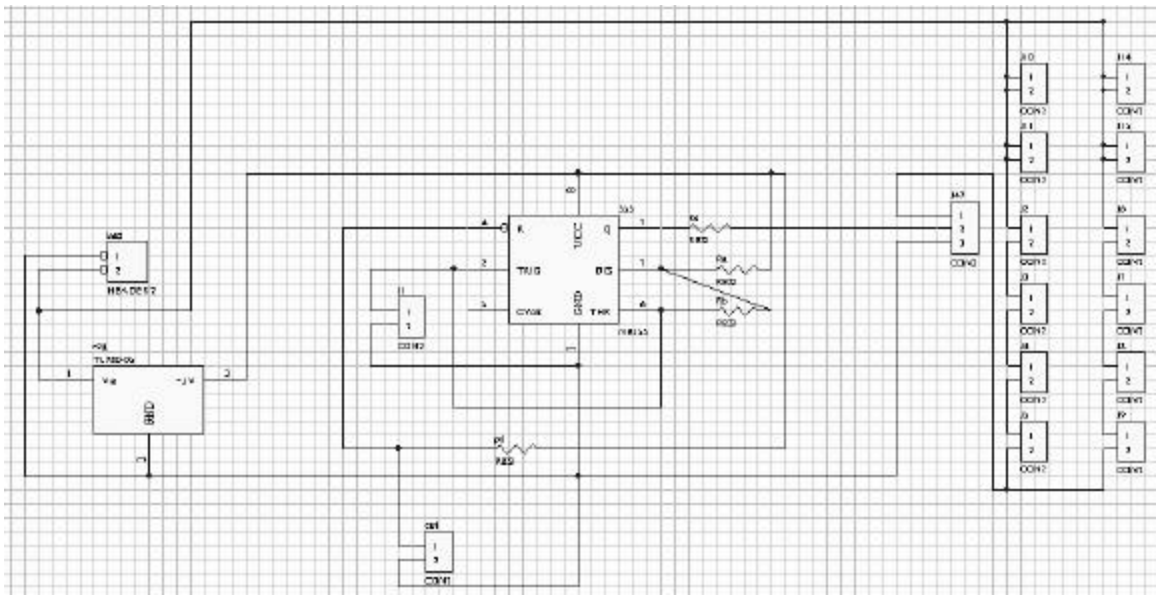


Fig. 6

In order to generate the control signals a series of 555 timer chips will be used. Figure 6 above shows the basic layout of the multidirectional IR system. The signal output from the timer chip is fed through a current limiting resistor into the base of a Darlington transistor which switches the IR emitters on and off. The Darlington circuit is derived from one that Aamir designed for the Lawn Nibbler robot. The power source for the board will consist of a standard 9v battery connected to the +5vdc regulator shown in the lower left hand corner of figure 6. The connector at the bottom of the figure is an photointerrupter diode. This diode will short the two terminals normally, however, when an object passes through the photointerrupter, the two terminals will be opened. When the two terminal are open, the reset line will be pulled high, and the timer chip will be turned on. When the two terminals are shorted, the reset line will be pulled low, and the chip will shut off. The end result of this is that the rotating emitter will have on it an object that will pass through the center of the photointerrupter when it reaches the 0° point. This will cause the multidirectional emitter(Fig. 6) to turn on for a brief moment, emitting a pulse, and then turn off.

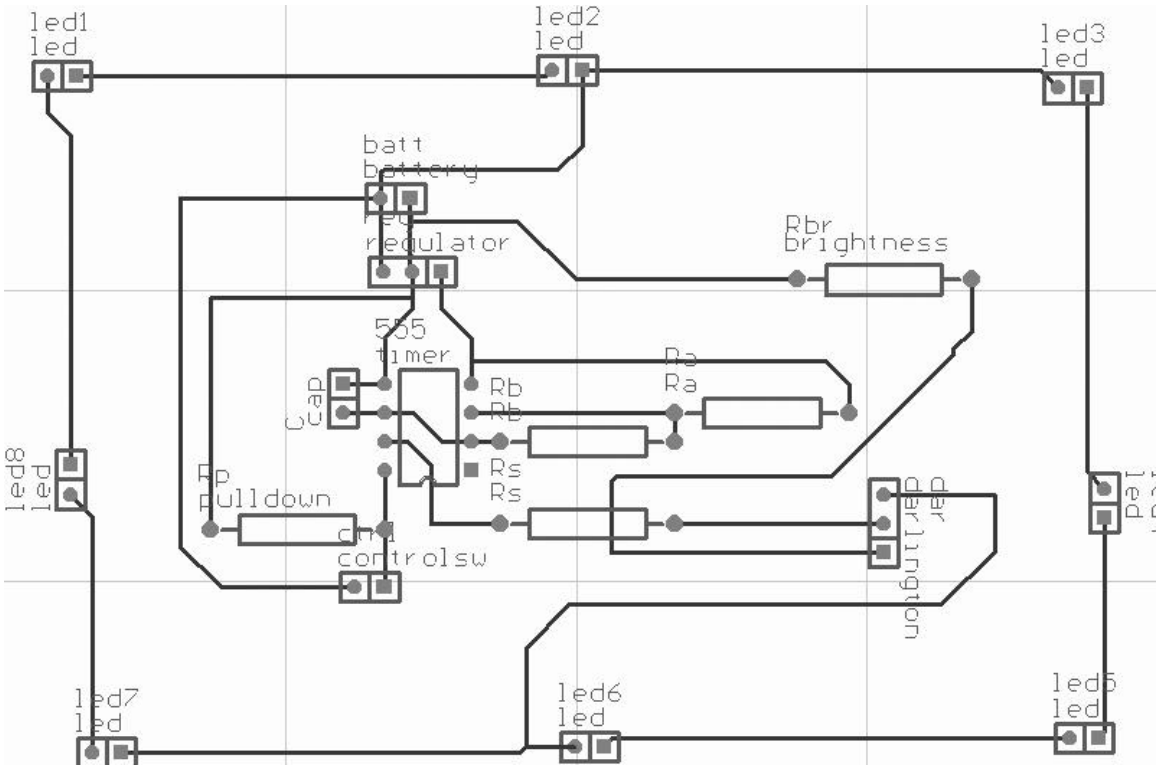


Fig 7a Multidirectional IR transmitter layout

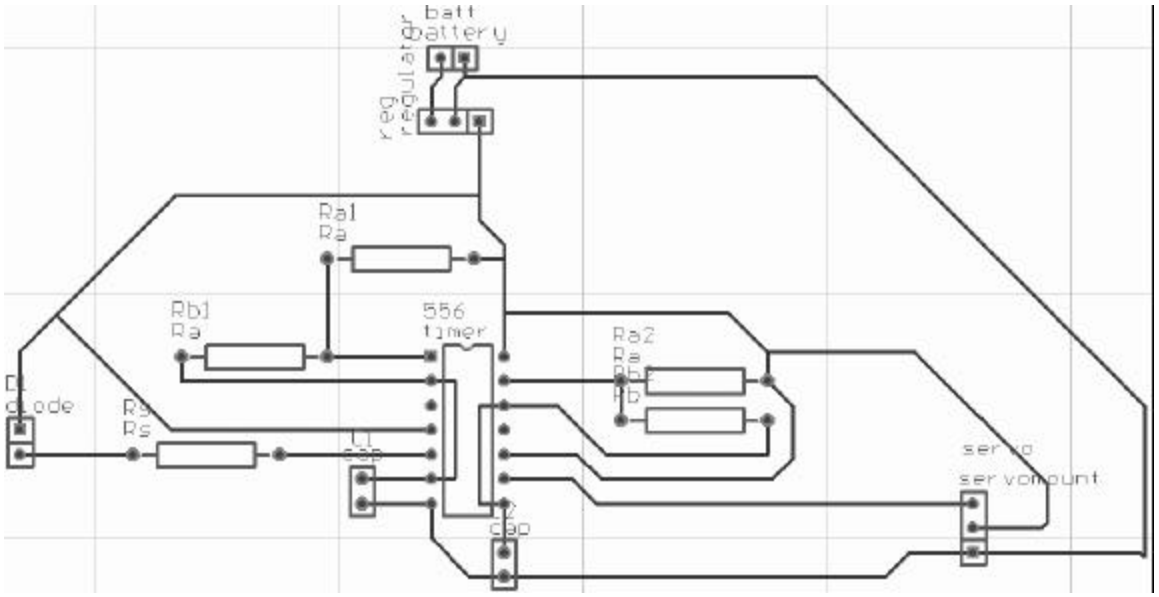


Fig 7b single direction IR transmitter and servo control layout

Figures 7a and 7b above show the Protel™ design files for the board layouts of the rotating and stationary emitters as well as the control for the servo that will rotate the single direction emitter. It is important to note that the layout file for the multidirectional emitter has an error in it. The pull-up resistor connected to photointerrupter header is connected to ground, and the other terminal of the header is connected to the positive terminal of the battery. These two connections should be reversed. The error arose from the fact that I was forced to route the board manually in Protel™ due to a bug in the program. While doing this I inadvertently switched the two connections. In the final design this was corrected by drilling out the affected traces, and then jumpering the terminals to their correct destinations with small segments of wire.

The assembled platform is shown below in figures 8a-e.

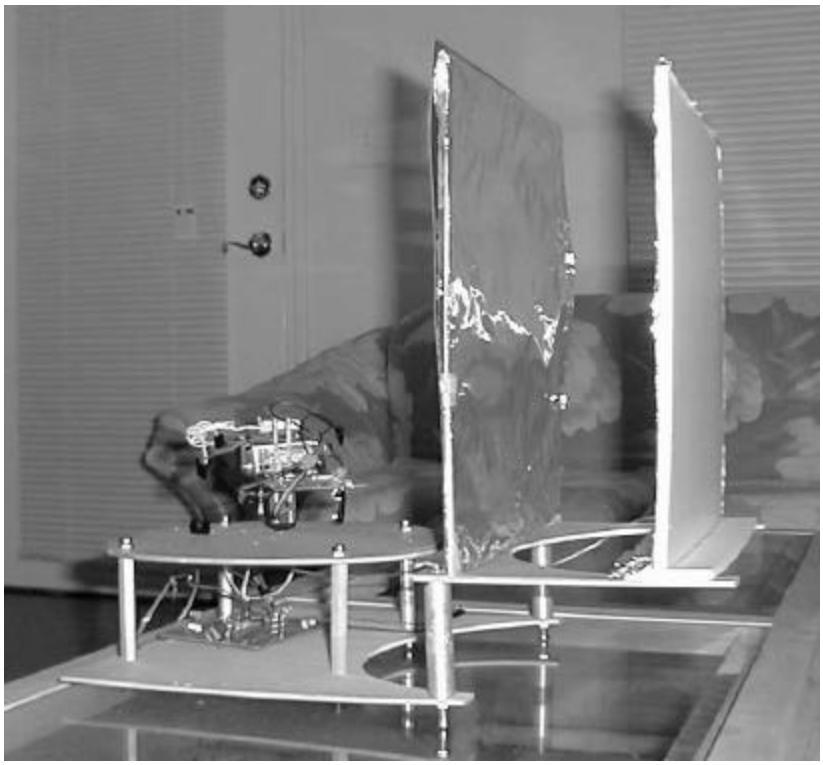


Fig 8a assembled sensor platform

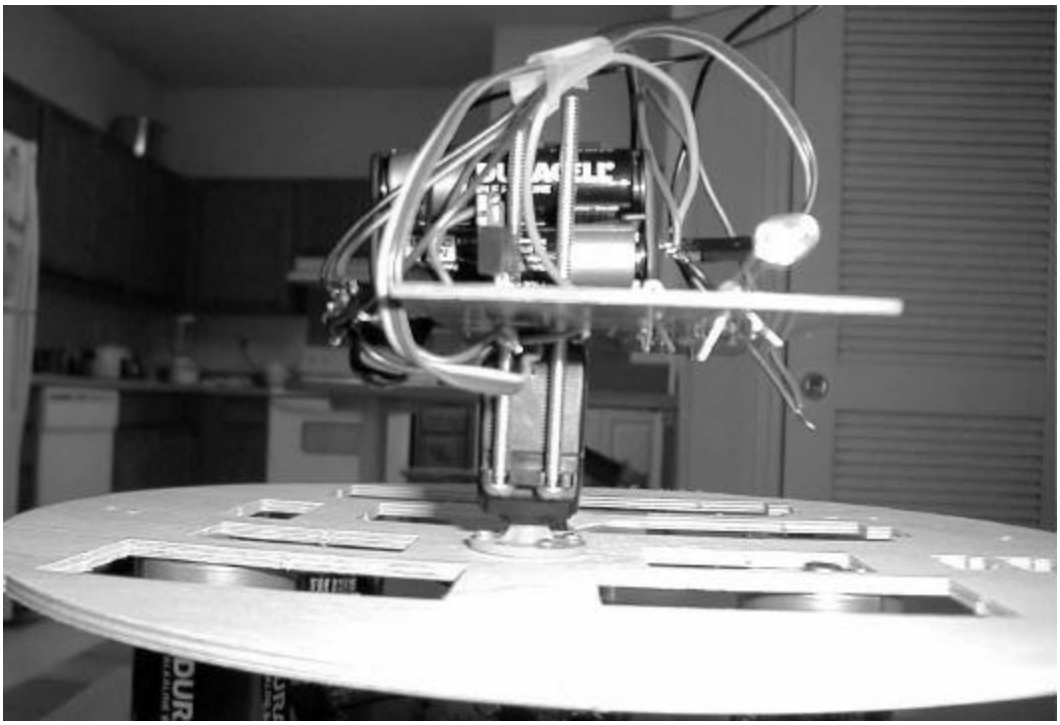


Fig 8b close-up view of servo mount

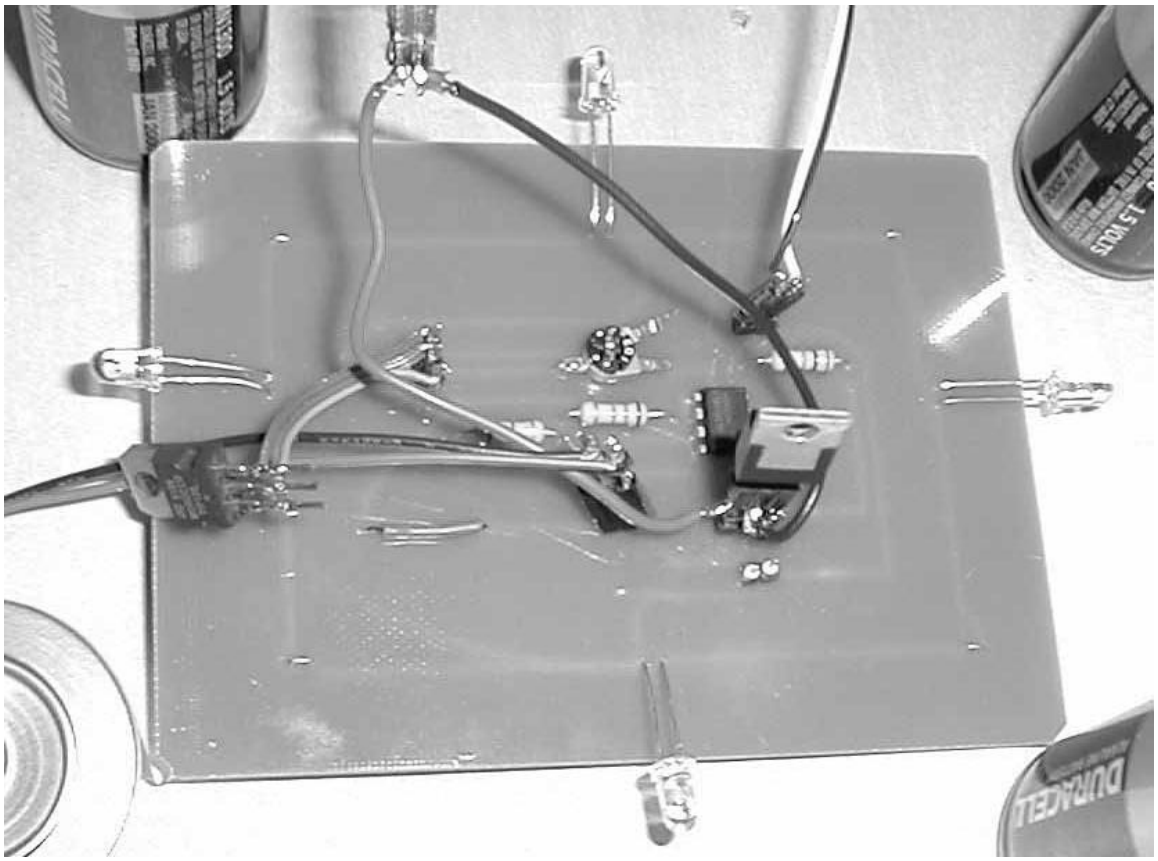


Fig8c overhead view of multidirectional IR board

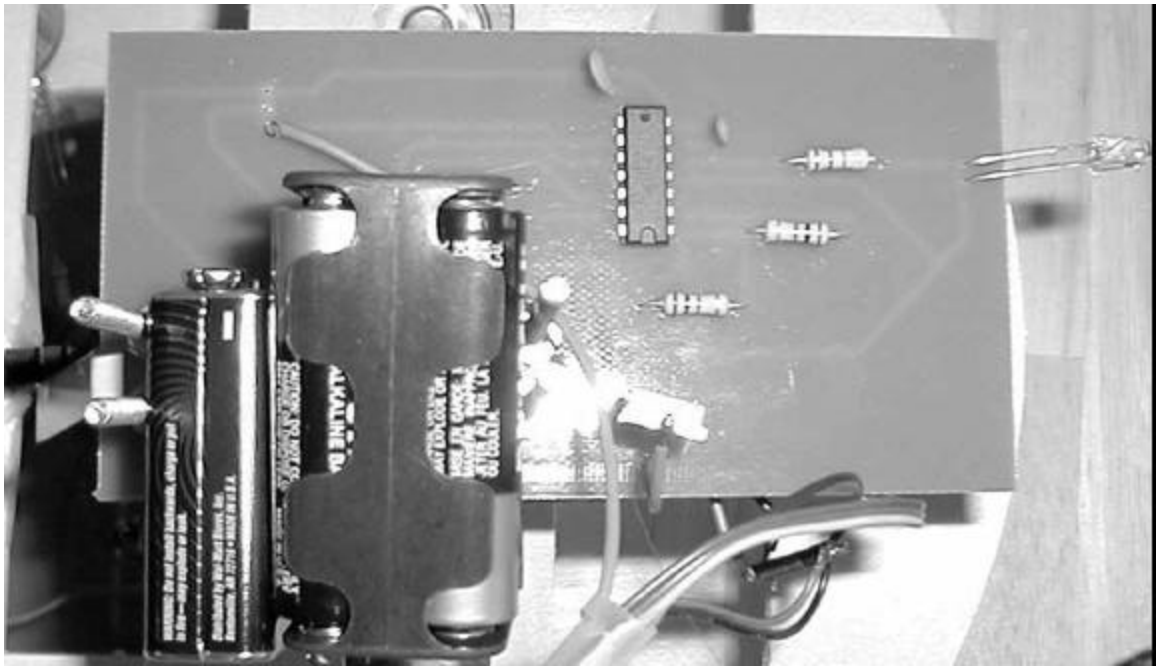


Fig8d overhead view of single direction IR board

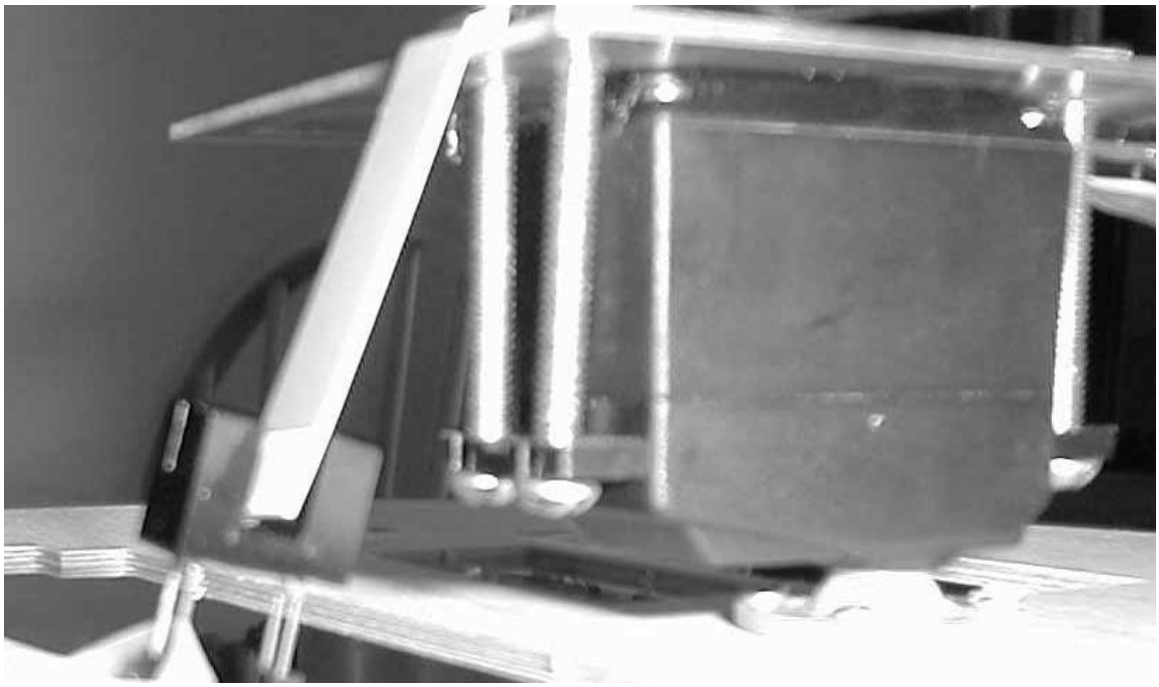


Fig8e close-up view of photointerrupter

Behaviors

Figure 9 on the next page illustrates the four main behaviors of the mobile robot: target acquisition, movement, firing, and hit detection. The dominant behavior is the target acquisition algorithm. When the mobile robot receives a pulse it stores the current time in a variable. Once the robot has received two pulses it begin calculating the time difference between the pulses. It then checks to see if the difference is within an acceptable range. If it is, it is added to a result array.

Once the robot has determined that it has recorded enough pulses to get a good reading(currently set at 4) the robot stores the largest of the pulse widths. It then moves to the left a small amount and repeats the above algorithm again. The robot then checks to see if the largest pulse widths of each reading increased or decreased. It then uses this information to determine its exact position. It then uses this information to blindly move to the front of the target.

Once the robot has reached the front of the target it begins the firing sequence. To fire the servo mounted to the trigger begins pulling the trigger back until the rear limit switch is depressed. It then reverses direction and returns the trigger until the front limit switch is pressed. At this point it has completed a single firing cycle. It then moves on to the hit detection phase.

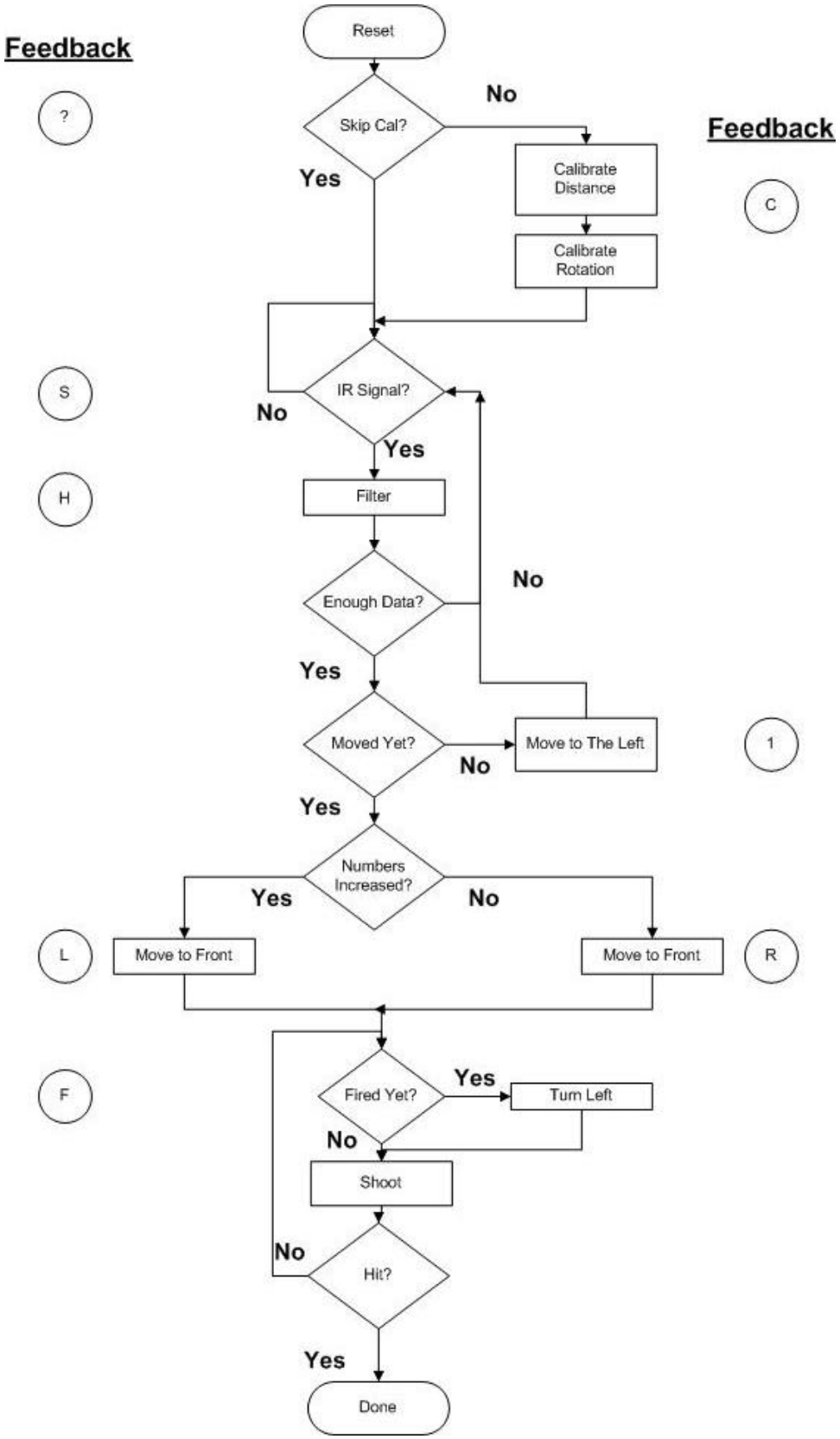

Feedback
?
Reset
Skip Cal?
No
Yes
Calibrate Distance
Calibrate Rotation
Feedback
C
S
IR Signal?
No
Yes
H
Filter
Enough Data?
No
Yes
Moved Yet?
Move to The Left
No
1
Yes
Numbers Increased?
Yes
No
L
Move to Front
Move to Front
R
F
Fired Yet?
Yes
Turn Left
No
Shoot
No
Hit?
Yes
Done

Figure 9

In the hit detection phase, the robot begins looking for a pulse from the multidirectional emitter. If the target has been hit, the multidirectional emitter will be turned off, and thus will not broadcast any pulses. If a total of 2.8 seconds(the emitter rotation period) elapses after the gun is fired, and no pulses are detected the target has been hit, and the robot begins its victory dance. If a pulse is detected, the target was not hit, so the robot moves a bit, and repeats the firing sequence again.

Experimental Layout and Results

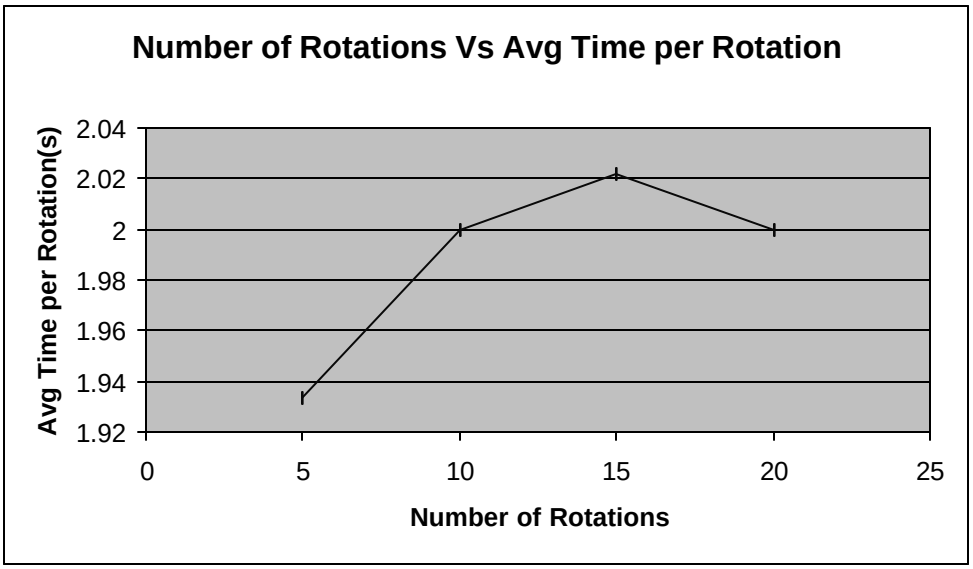
Several tests were required to determine the operating characteristics of the target sensor systems so that software can be written to produce useful calculations based off of their data.

Test1: Determination of rotation speed

This first test is designed to test the rotation speed of the rotating IR emitter. This test will tell us the rotation speed of the emitter, and any variation in rotation speed over time. The emitter will be switched on and allowed to rotate freely I will then record the time it takes for the emitter to:

1. Complete 5 full rotations.
2. Complete 10 full rotations.
3. Complete 15 full rotations.
4. Complete 20 full rotations

A complete rotation is defined as the time it takes the emitter to pass by the 0° mark twice. In order to time the rotations, I will be using my analog wristwatch. Because of this, I expect a measurement uncertainty of about +-1s.



Number of Rotations	Time Required(s)	Time per Rotation(s)	Average Time per Rotation	Uncertainty(s)
5	9	1.8	1.933333333	0.2
5	10	2		
5	10	2		

Number of Rotations	Time Required(s)	Time per Rotation(s)	Average Time per Rotation	
10	20	2	2	0.1
10	20	2		
10	20	2		

Number of Rotations	Time Required(s)	Time per Rotation(s)	Average Time per Rotation	
15	30	2	2.022222222	0.066666667
15	31	2.066666667		
15	30	2		

Number of Rotations	Time Required(s)	Time per Rotation(s)	Average Time per Rotation	
20	40	2	2	0.05
20	41	2.05		
20	39	1.95		

Fig. 10 Rotation Speed Test Data

My first surprise upon taking this data was that all of my measurements seem to indicate that the servo was rotating the emitter at a very constant speed. Each of my measurements exhibited an error of around +-1s which was identical to my measurement uncertainty. I also expected for the data to show

a relationship more akin to that of a decaying sine wave. By this I mean that my data should have shown a large error that gradually decayed down to 0 as the number of rotations was increased. My data did show this, as my measurement uncertainty decreased as the number of rotations the avg. time per rotation seemed to approach 2s. Therefore, it is safe to assume that my rotation speed is 360°/2s = 180°/s

Test 2: IR emitter signal strength

This second test is designed to measure the signal strength of both of the IR emitters. This test will tell us the maximum distance that the receiver can be placed from the platform and still receive meaningful data. I will begin by setting up the emitter on a flat surface, and then placing the receiver next to it. I will take a measurement, and then pull back the receiver and repeat the process. It is important to note that I only require a digital signal, so to receive the multidirectional pulse I will be using an unmodified RadioShack 40KHz IR receiver. To receive the directed pulse I will be using a receiver constructed from a MAX266 programmable filter chip. This filter will output an analog IR value, but it is only necessary to determine if this value exceeds the threshold value of the device in order to detect a signal. The results of these tests are shown on the next page.

Multidirectional Emitter(40KHz):

Normal Level	Range(in)	Peak Level	(Normal-Peak) Level	
	252	1	1	251
	252	2	1	251
	252	3	1	251
	252	4	1	251
	252	5	1	251
	252	6	1	251
	252	7	1	251
	252	8	1	251
	252	9	1	251
	252	10	1	251
	252	11	1	251
	252	12	1	251
	252	24	1	251
	252	48	1	251
	252	60	1	251
	252	72	1	251
	252	84	1	251
	252	90	1	251

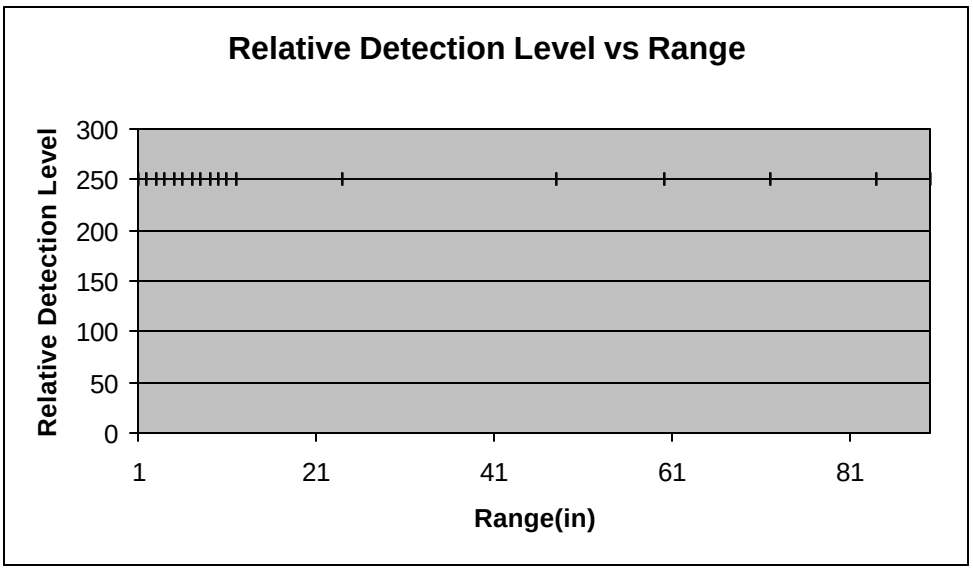


Fig. 11 Multidirectional IR Emitter Reception Test Data

Single Direction Emitter(32kHz):

Normal Level	Range(in)	Peak Level	(Peak-Normal) Level	
	25	1	50	25
	25	2	48	23
	25	3	52	27
	25	4	47	22
	25	5	48	23
	25	6	54	29
	25	7	54	29
	25	8	54	29
	25	9	56	31
	25	10	43	18
	25	11	45	20
	25	12	45	20
	25	15	40	15
	25	20	31	6
	25	22	33	8
	25	23	30	5
	25	24	26	1
	25	30	27	2
	25	31	26	1
	25	32	27	2
	25	33	25	0
	25	34	25	0

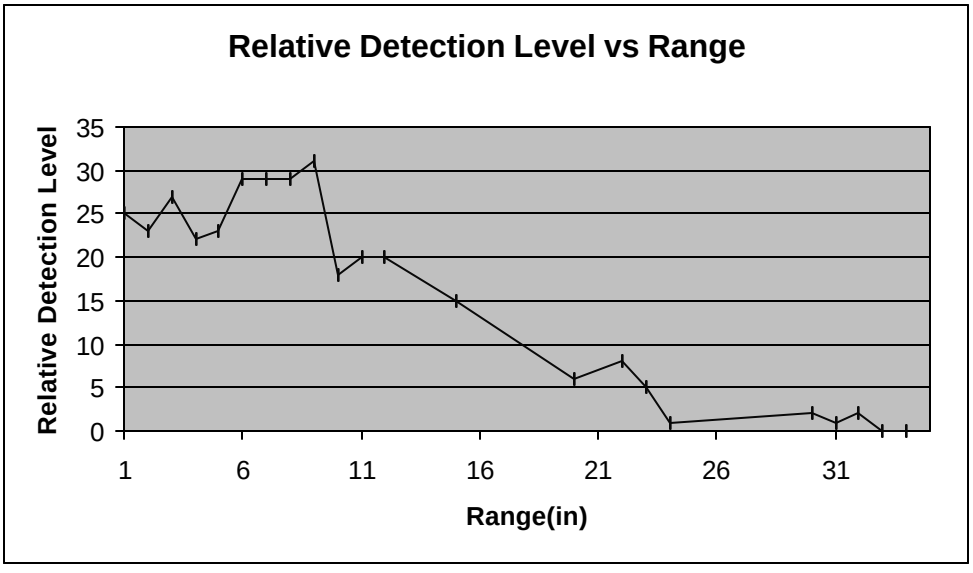


Fig. 12 Single Direction IR Emitter Reception Test Data

The test of the multidirectional IR system produced some interesting results. The IR detector was digital which explains why the output was always 252. However, I did not anticipate how good of a receiver it was. I was unable to move the receiver far enough back from the transmitter to cause it to lose the signal. Due to the size of my lab, I was only able to move the platforms 90 inches(7.5 feet) apart. I do not view this as a problem as I do not plan on having the robot over 7.5 feet from the platform anyway.

The test of the single direction IR system produced much more useful data. As expected, the receiver's output resembled a decaying exponential. To understand why the output appears so "rough" I must discuss the output of the detector. The detector's primary output consists of a slightly amplified version of the original 32kHz square wave that was provided as input. Unfortunately, the filter rejects the normal 2.5v offset of the input signal(see Fig 11), which it views as noise and recenters the signal about ground(see Fig 12).

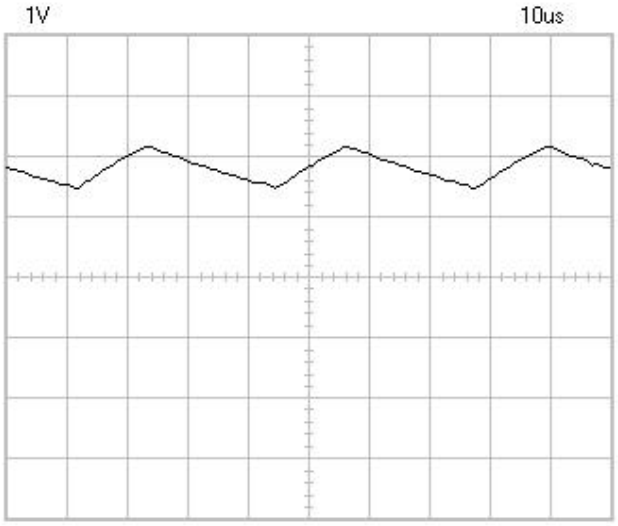


Fig 13 32Khz Filter Input

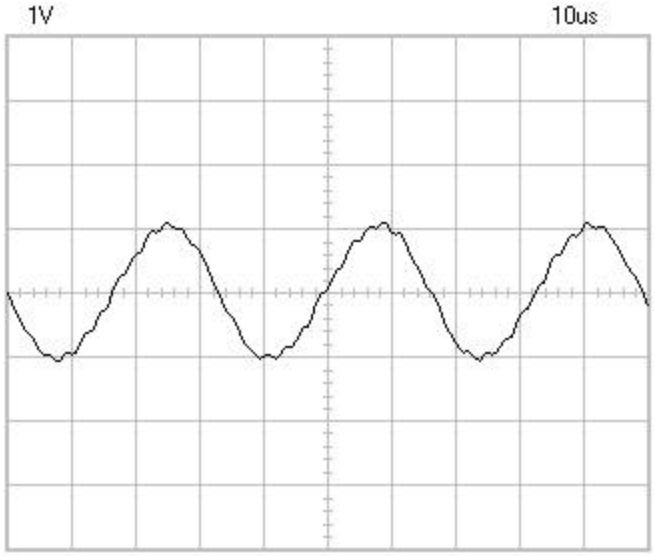


Fig 14 32kHz Filter Output

To combat this offset, and obtain a usable signal, I added a diode in series with the filter output, and left the cathode floating. By sampling the voltage at the cathode I can obtain a rough value of the peak of the output waveform(see Fig 15). This clipped signal has a ripple of about .2V which causes any value I obtain to appear to slightly fluctuate.

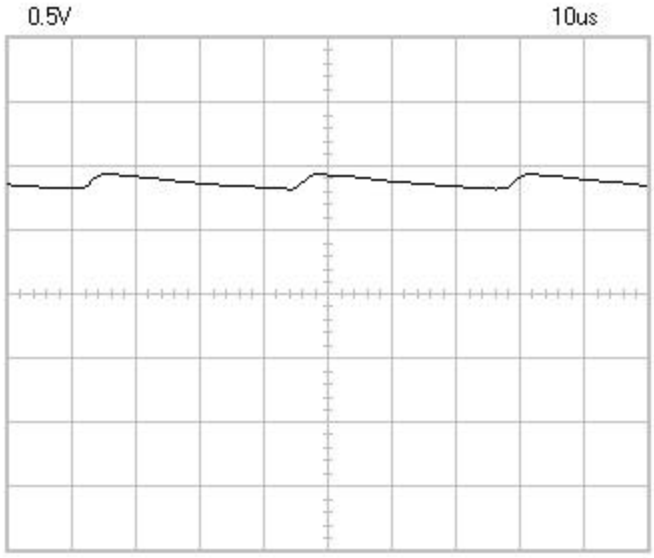


Fig 15 Clipped Filter Output

The fluctuations in the output are irrelevant as all I need is a digital signal is/is not present value which can be obtained by subtracting the ambient level(column 1 in Fig 12) from my output reading(column 3 in Fig 12), and checking to see if the resultant value(column 4 in Fig 12) is larger than some threshold value.

The final results of this test are:

System	Max Range
Multidirectional IR	90in(7.5')
Single Direction IR	32in(2.67')

This test was performed using a home made Max 266 based 32kHz IR detector. The results of this test showed that the receiver was not very useful, so I ended up replacing it with a sharp can that received both the 32kHz and 40 kHz signals. This had the effect of lowering the accuracy of the system slightly, but increasing the range dramatically.

Test 3: IR Emitter Dispersion

All LEDs emit light in a cone shaped arc. This third test is designed to measure the maximum angle, from the center of the IR LED mounted on the rotating platform, that a usable signal can be obtained. For this test I placed the receiver module 1' away from the platform and rotated the emitter until the signal was first detected, and then measured the angle of the emitter relative to a point tangent to the detector. This resulted in a dispersion angle of about 32°

off center. This means that the detector's calculated position will always be 32° too large, so a correction factor of -32° will be applied.

Test 4: Target Platform Readings

This test was designed to gather information on what phase differences correspond to various positions around the target platform. I will begin by setting up the emitter on a flat surface, and then placing the receiver next to it. I then will move the robot around the target platform and record the phase differences. The results are shown below in figure 16.

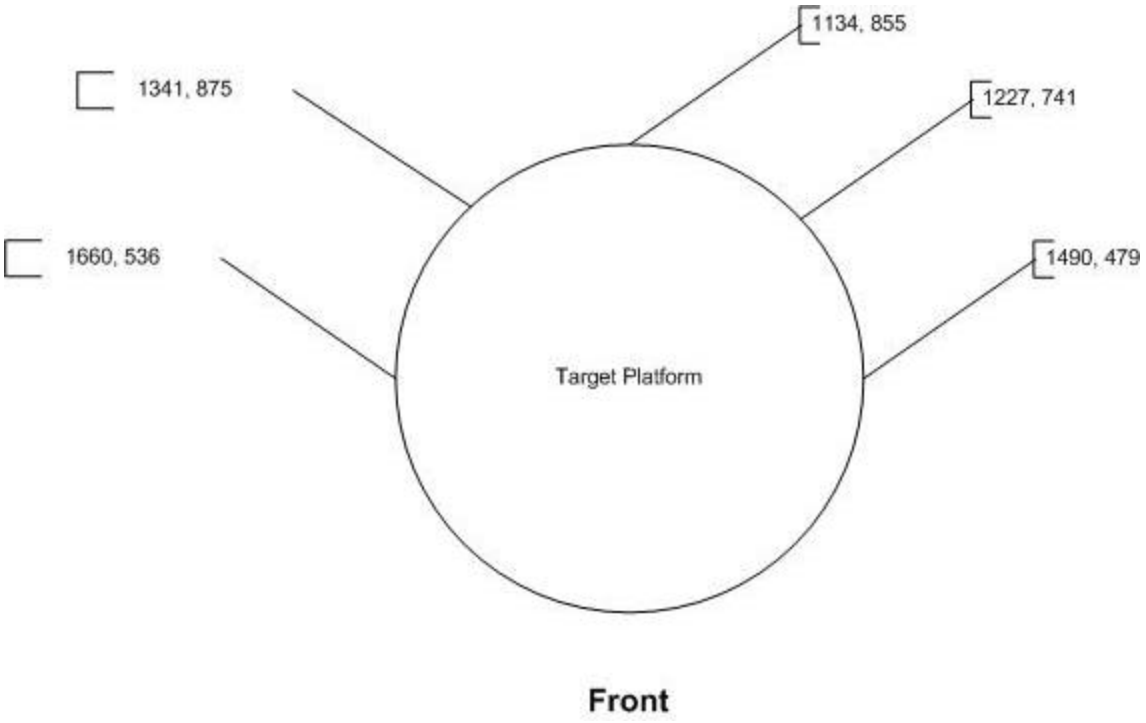


Fig 16 Target Sensor Data

We can see from the above data that as we move clockwise around the target, the larger number tends to increase on the right hand side of the target, and to decrease on the right hand side of the target.

Conclusion

As it stands now, the robot can fully complete all of its prescribed functions. In the future I would like to change the 32Khz emitter on the target platform to a 56.8kHz emitter. This will allow me to use an off-the-shelf IR receiver unit for each of the frequencies, and thus be able to receive the pulses on mutually exclusive channels. This will have the net effect of increasing my accuracy dramatically, as well as removing the need to move the robot to obtain a second sample point.

The largest difficulty I encountered throughout this project was that my method of mounting the servos to the wheels was horribly ineffective. I had mounted a large washer to the servo horn, and then attached that washer to the wheels using super glue. Unfortunately this made the mounting horribly unstable, and the glue was not strong enough to hold the wheels together. They constantly had to be reattached, and I finally was forced to epoxy them directly to the servos, after repeated reattachment had stripped out their screws. Another source of frustration was that the sharp can I was using detected the IR from the lights in the lab, which caused about a week of fruitless testing of my system before I found the problem(the night before demo day).

Documentation

Jones, Flynn, Seiger. *Mobile Robots: Inspiration to Implementation*. Second Edition, AK Peters. Natick, Massachusetts, 1999.

Appendix A

Sources of parts

Part: Photointerrupter model LTH301-07

Company: Lite-On Optoelectronics

Source: Arrow Electronics

Comments: Avoid this company like the plague. Their collection of online datasheets is incomplete, and the datasheets that they do have up are often missing important data. Their US website is a joke, and I was forced to go to the European website(http://www.liteon.com.tw) to find the datasheet for this part. I have a collection of about 30 different parts from them, and according to Lite-On's websites only about half of these parts have valid part numbers.

Part: 5V regulator model TL780-05CKC

Company: TI

Source: TI

Comments: All parts were sent UPS next-day air. Website is easy to navigate, and always gave the required info.

Part: adjustable regulator models LM317KC and LM337KC

Company: TI

Source: TI

Comments: Datasheets for these parts did not contain very much useful application information.

Part: photodiode model QSC112 and photodarlington model QSC133

Company: Fairchild

Source: Fairchild

Comments: This company has an extremely odd sample program, it took 2 weeks for the parts to arrive because they were airmailed from Hong Kong. I could not get the photodarlington to activate no matter how much IR I projected onto it. The photodiode has a very small detection angle, but gives decent response.

Part: adjustable filter model Max266

Company: Maxim

Source: TA

Comments: Perfumed adequately. The development software will not run on my system (Windows ME).

Part: foam disc launcher

Company: Kids Only™

Source: Walmart

Comments: A decently accurate gun with a light trigger pull. I did have a bit of a problem with the trigger dragging on the left hand side of the gun.

Appendix B

Code listing

```
/************************************/
/*                                  */
/*     Test of Target Sensor        */
/* Robot takes 4 readings, then moves*/
/* clockwise then repeats.  It then  */
/* moves to the front of the target  */
/* fires a projectile, and detects   */
/* a hit or miss.                   */
/*     Author: Ian St.John          */
/*      with parts by Keith L. Doty  */
/************************************/


#include <tkbase.h>
#include <stdio.h>

/*IR emitter output port driver, the output latch at address 0xffb9 */
#define IRE_OUT   *(unsigned char *)(0xffb9)

/*Constant for driving all the 40KHz modulated IR emitters on when
loaded into IRE_OUT */
#define IRE_ALL_ON 0xff

/*Constant for turning all the 40KHz modulated IR emitters off when
loaded into IRE_OUT */
#define IRE_ALL_OFF 0x00
#define ONE_FOOT (mvtime/2)

void ch_mux(int bit_field);
unsigned int r_bumper(void);
unsigned int f_bumper(void);
void archer_motor(int,int);
unsigned int calibrate_motors();

unsigned int calibrate_turn();

void main(void)
{
  extern unsigned int timertk;
  unsigned int run_test;

  unsigned int a,b,bl, cv, fb,
rb,ind,w,x,y,z,loo,lcnt,mvcnt,mvtime,turntime,missed,side,now;
  unsigned int list[20];
  char clear[]= "\x1b\x5B\x32\x4A\x04";  /* clear screen */
  char place[]= "\x1b[1;1H";        /* Home cursor */

  IRE_OUT = IRE_ALL_ON;    /* Turns on the feedback display */

  for(loo=0;loo<19;loo=loo+1) /*initialize all variables that need it
*/
```

```
      list[loo]=0;
       ind=0;
  lcnt=0;
  mvcnt=0;
  missed=0;
  x=y=z=0;

  init_analog();  /*init everything */
  init_motortk();
  init_clocktk();
  init_serial();
  init_servos();
  printf("%s", clear);  /*clear screen*/
  printf("%s", place);  /*home cursor*/

  /*obtain a baseline IR level for the collision avoidance code */
  read_IR();  /* read in the IR ports */
  a=IRDT[2];
  b=IRDT[0];
  read_IR();  /* read in the IR ports */
  a=(a+IRDT[2])/2;




  IRE_OUT = 157; /* ? */

  while((r_bumper()<10)&&(f_bumper()<10)); /* wait for it... */

  if((r_bumper()>10)){ /* calibration mode */
    IRE_OUT = 198; /* C */
      mvtime=calibrate_motors();
      turntime=(calibrate_turn())/4;
  }

while(1)
  {
  IRE_OUT = 103; /* S */
  loo=timertk;




  cv=analog(2);
  if(cv<100){
      printf("Hit on 2     Time of last hit:%u\n",timertk);
      y=w;
      w=x;
      x=timertk;

       if(((x-w)>280)&&((w-y)>280)&&((x-w)<2000)&&((w-y)<2000)){ /*
valid pulse width */
            IRE_OUT = 171; /* H */
            list[0]=(x-w);
            list[1]=(w-y);
            lcnt++;
```

```
      }

      if(lcnt==4){ /* ready to compile data */
       printf("Phase shifts\n %u       \n %u       \n
",list[0],list[1]);
       if(list[0]>list[1])
            list[2]=list[0];
       else
            list[2]=list[1];

       if(mvcnt==0){ /* first movement phase */
           IRE_OUT = 40; /* 1 */
           init_motortk();
             archer_motor(0,50); /* turn */
             archer_motor(1,-50);
             wait(turntime);
             archer_motor(1,50);  /* go straight */
             wait(600);
             archer_motor(0,-50);  /* turn back */
             wait(.9 * turntime);
             archer_motor(0,0); /* stop */
             archer_motor(1,0);
             mvcnt++;
             lcnt=0;
             list[3]=list[2];
       }
       else { /* second movement phase */
            if(list[3]>list[2]){ /* left hand side */
                  side=1;
                  IRE_OUT = 194; /* L */
                  init_motortk();
                  archer_motor(1,50); /* turn */
                  archer_motor(0,-50);
                  wait(.75*turntime);
                  archer_motor(1,0); /* stop */
                  archer_motor(0,0);
                  wait(turntime);
                  archer_motor(0,50);  /* go straight */
                  archer_motor(1,50);
                  if(list[2]<1300){
                        wait(3 * ONE_FOOT);
                  }
                  else if(list[2]<1600){
                        wait(3 * ONE_FOOT);
                  }
                  else {
                        wait(4 * ONE_FOOT);
                  }

                  archer_motor(0,50);  /* turn back */
                  archer_motor(1,-50);
                  wait(.8*turntime);
                  archer_motor(1,50); /* go straight */
                  wait(2 * ONE_FOOT);
                  archer_motor(1,-50);  /* turn back */
                  wait(.75 * turntime);
                  archer_motor(0,0); /* stop */
```

```
            archer_motor(1,0);
            lcnt++;
       }
       else {  /* right hand side */
            side=2;
            IRE_OUT = 129; /* r */
            archer_motor(0,50); /* turn */
            archer_motor(1,-50);
            wait(1.1*turntime);
            archer_motor(1,0); /* stop */
            archer_motor(0,0);
            wait(turntime);
            archer_motor(0,50);  /* go straight */
            archer_motor(1,50);  /* go straight */
            if(list[2]<1300){
                  wait(3 * ONE_FOOT);
            }
            else if(list[2]<1600){
                  wait(3 * ONE_FOOT);
            }
            else {
                  wait(3 * ONE_FOOT);
            }

            archer_motor(1,50);  /* turn back */
            archer_motor(0,-50);
            wait(.9*turntime);
            archer_motor(0,50); /* go straight */
            wait(2 * ONE_FOOT);
            archer_motor(0,-50);  /* turn back */
            wait(.6 * turntime);
            archer_motor(1,0); /* stop */
            archer_motor(0,0);
            lcnt++;
       }

  } /* end second movement phase */
} /* done with data anal(lcnt==4) */
else if((lcnt>4)&&((timertk-x)<2800)){  /* fire control */
      if((missed>0)&&(missed<10)){
        if(side==1){
            archer_motor(0,25); /* turn */
            archer_motor(1,-25);
            wait(.25 * turntime);
            archer_motor(0,0); /* stop */
            archer_motor(1,0);
          }
        else{
            archer_motor(0,-25); /* turn */
            archer_motor(1,25);
            wait(.2 * turntime);
            archer_motor(0,0); /* stop */
          archer_motor(1,0);
          }
      }

      IRE_OUT = 135; /* F */
```

```
            init_servos();
            if(missed<10){
                  servo(1,3500); /* pull back trigger */
                  while(analog(3)<10); /* wait for limit switch */
                  servo(1,3780); /* pause */
                  wait(1000);
                  servo(1,4080);/* pull forward trigger */
                  while(analog(4)<10);/* wait for limit switch */
                  servo(1,3780);/* pause */
                  wait(1000);
                  now=timertk;
                  missed++;
            }


      }
      else {
       printf("\n\n\n");

       printf("\n This took %u     ",timertk-loo);
       printf("%s", place);  /*home cursor*/
       }

}
else {
      printf("Miss on 2\n %s", place);
      if((missed>0)&&((timertk)>(now+2800))){ /* hit detection */
            IRE_OUT = 171;
            while(1){
                  archer_motor(0,-50); /* turn */
                  archer_motor(1,50);
                  wait(500);
                  archer_motor(0,0);
                  archer_motor(1,0);
                  wait(100);
                  archer_motor(1,-50); /* turn */
                  archer_motor(0,50);
                  wait(500);
                  archer_motor(0,0);
                  archer_motor(1,0);
                  wait(100);
                  archer_motor(1,-50); /* turn */
                  archer_motor(0,50);
                  wait(4 * turntime);
                  archer_motor(0,0);
                  archer_motor(1,0);
                  wait(1000);
            }
      }


      }

} /*end while*/
```

```
}



void ch_mux(int bit_field){

      bit_field  = bit_field & SEL_field;  /* Mask the unused bits */

      CLEAR_BIT(MMR_mirror, SEL_field);/* Clear the SEL field of the
                                          MMR */
      SET_BIT(MMR_mirror, bit_field); /* Set the appropriate bits in
                                          MMR.SEL */
      MMR = MMR_mirror;            /* Load MMR with MMR_mirror */
 }

unsigned int r_bumper(void){
   ch_mux(R_Bump_SEL);
  return analog(1);
}

unsigned int f_bumper(void){
   ch_mux(F_Bump_SEL);
  return analog(1);
}

void archer_motor(int index, int speed){
 if(speed==0){
      motortk(index,25);
      wait(50);
      motortk(index,0);
 }
 else
      if(index==1)
            motortk(index,(speed ));
      else if(index==0)
            motortk(index,-(speed + 8));

}

unsigned int calibrate_motors(){

  unsigned int timevar=0;
  unsigned int wtimevar=0;
         init_clocktk();

  do{
        while((r_bumper()<10)); /* wait for it... */
        init_clocktk();
        timevar=timertk;
        archer_motor(0,50);
      archer_motor(1,50);
        while((f_bumper()<10)){}
        timevar=timertk-timevar; /* time for bot to move 1' */
        printf("a mere %u msec to move 1'\n",timevar);
```

```
        archer_motor(0,0);
        archer_motor(1,0);
        while((r_bumper()<10)); /* wait for it... */

  }while(timevar<=0);
  return (timevar);


}


unsigned int calibrate_turn(){

  unsigned int timevar=0;
         init_clocktk();

  do{
        while((r_bumper()<10)); /* wait for it... */

        timevar=timertk;
        archer_motor(0,-50);
            archer_motor(1,50);
        while((f_bumper()<10)){}

        timevar=timertk-timevar; /* time for bot to move 1' */
        printf("a mere %u msec to turn 90 deg.\n",timevar);

        archer_motor(0,0);
        archer_motor(1,0);
        while((r_bumper()<10)); /* wait for it... */

  }while(timevar<=0);
  return (timevar);


}
```