ROVER I
Mapping and Navigation Robot

Jose Nunez

IMDL 5666

8/8/00

Table of Contents

III. Abstract

A mapping and navigation robot was constructed with the purpose of exploring a small area while searching for objects to map. Robot position and object location data is transmitted via a radio-frequency channel to a computer (programmed in Visual Basic). Infrared shaft encoders were used for positional tracking and infrared sensors were used for object detection and characterization. It was found that maintaining straight paths while navigating proved very difficult and future work would include the integration of a superior position tracking technique.

IV. Executive Summary

Mapping and navigation has an important role in robotics, allowing environmental information to be gathered in places which pose serious risks to humans. The purpose of this project is the construction of a robot which will navigate a small flat area while searching for objects. The robot circumnavigates objects, creating a perimeter around them in order to gauge their approximate size. Radio-frequency communication is used to transmit the robot's position and object information to a computer.

The robot (called Rover I) consists of three main systems: actuation (allowing robot to maneuver), sensory (so it can perceive its environment as well as track its position), and communication (to transmit data to a computer). The actuation is accomplished via modified servos while the sensory system consists of infrared proximity detectors. A half-duplex 315 MHz radio-frequency channel serves as the communications system.

The robot was designed in AutoCad as a modified version of the Talrik Junior robot platform (copyright Mekatronix). It was constructed out of 3/16" balsa wood using a circuit board milling machine.

Actuation consists of two modified servos (positional feedback circuitry removed). Direction and speed control is achieved via a motor driver chip using pulse width modulation

Sharp can infrared sensors are used in conjunction with infrared emitting L.E.D.'s for detecting objects as well as in sensing objects for circumnavigation. Two infrared L.E.D.'s were seated in black tubes in order to increase resolution and decrease intensity, allowing for a more accurate and efficient circumnavigation procedure. Shaft encoding was used for position tracking using smaller less sensitive I.R. emitters and detectors.

The robot roams a 10' x 10' area of a floor using a plow technique (criss-crossing the area) while transmitting its progress to a computer. When objects are found the robot circumnavigates them creating a perimeter which roughly outlines the size of the objects. After this has been accomplished the robot continues its roaming behavior, searching for more objects.

Experiments were performed in order to determine the efficacy of the infrared sharp cans in determining distance (for circumnavigation procedure). A linear region was found between roughly 2 – 8" distance from the sensor. The I.R. L.E.D.'s were inserted into collimating tubes in order to improve short distance perception and spatial resolution. Object angle tests were done to determine the limitations and accuracy of the sensor readings.

Following a straight line was more difficult than initially speculated. Future work must include a superior positional tracking system which is dependent on an external landmark or signal (e.g. a compass or several homing beacons). The original goal of obtaining a two dimensional slice of objects found was not met due to time constraints and so must be left as part of future work as well.

V. Introduction

Mapping and navigation has several important applications in robotics. Because robots are unmanned, they can be used to go where environmental conditions present significant health risks to humans: deep seas mine fields, and other planets (e.g. Mars) . Autonomous robots also offer the advantage of maintaining functionality in the case of loss of direct control, such as when wireless communications fail.

The focus of this project is the construction of an autonomous robot (named Rover I) which will navigate a small area on a flat surface while searching for objects while tracking and transmitting its own position as well as the location of objects it encounters to a computer via radio-frequency communication.

The first section gives an overall description of the robot's sensing, actuation, and communications systems. The next two cover the platform design and actuation details. Sensor specifics are then discussed followed by the robot's behavior algorithms. The last section describes the experiments performed in order to gain knowledge of how the sensors operate and their limitations.

VI. Integrated System

In general, the overall structure of a robot consists of three fundamental sub-systems: processing/control, sensors, and actuation. In addition, Rover I also contains a communication sub-system. The sensors will consist of infrared proximity sensors and infrared shaft encoders using a slotted wheel. The actuation will be performed with modified servos, the communication with a radio frequency module,. and the processing with an expanded micro-controller. Details of these components will be given in later sections. The data and control diagram is presented below:

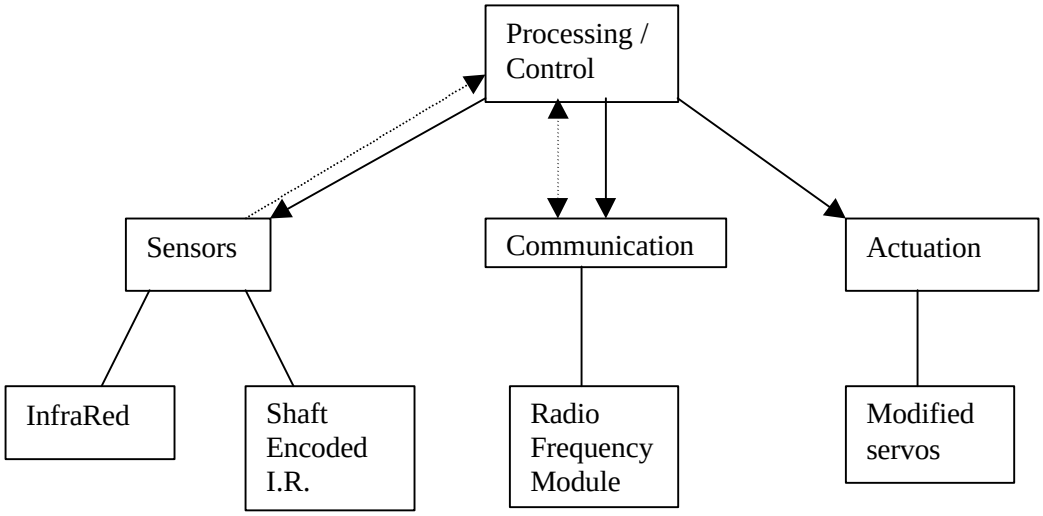


Figure 1 - Data and Control Diagram of Integrated system

Object detection and size determination will be accomplished via the I.R. sensors. Location (both robot and objects) information is conveyed to the micro-controller via the shaft encoded wheels. The R.F. module will serve as the communication link to a P.C.

VII. Mobile Platform

Balsa wood (provided by the IMDL lab) serves as the material for all of the platform components. The design is a modified version of the Talrik Junior platform (see Mekatronix website). Changes include a larger top circular plate in order to fit the Motorolla EVBU board (with ME11 memory/port expansion), and wider under-carriage components to increase stability when turning (see the following page for Autocad diagram). Both the Talrik Junior and the ME11 expansion board are copyrighted designs of Mekatronix™.

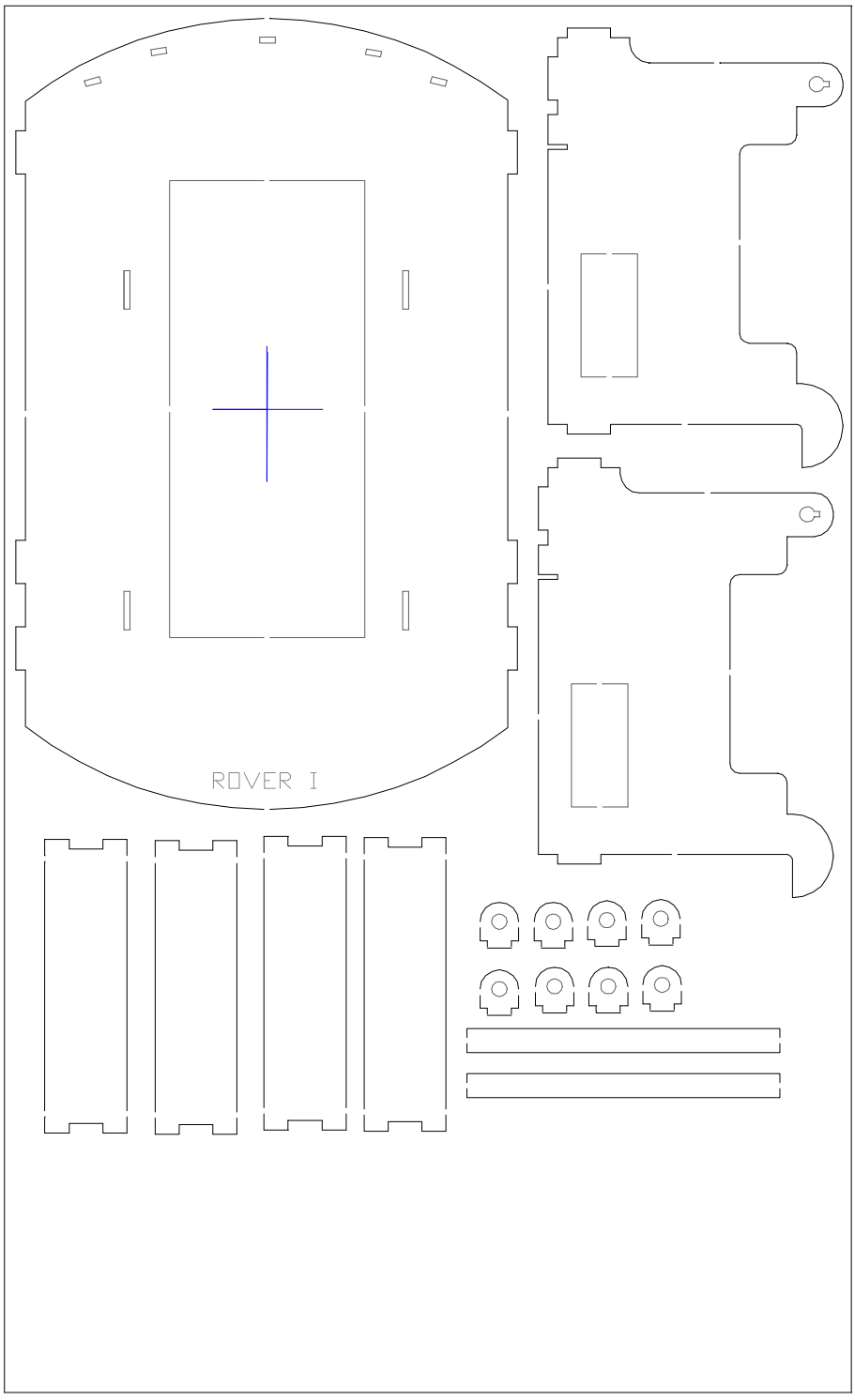


Figure 2 - Autocad design of modified TJ platform

VIII. Actuation

Actuation is the simplest of the sub-systems within the Rover robot. The goal is to provide the robot with directional mobility, allowing it to move forward, backward, and to turn This is accomplished via two modified Tower Hobbies® servos model STD TS-53 connected to wheels. They will be attached to the under-carriage of the platform in the same orientation as found in the Talrik Junior design (see Mobile Platform section).

A servo essentially consists of an electric motor, some gears to convert the high speed/low torque of the motor to low speed/ high torque, and some positional feedback circuitry . The servo is powered by a 5-Volt D.C. input and is controlled via a third wire carrying pulse width modulation information.

The servos were modified by removing physical limiters which restrict full 360 degree motion followed by complete removal of the positional feedback circuitry so that all that remained were the motor and the gears. The servos were then connected to a dual motor controller chip (SN754410) which control the direction and speed of the servo motors with a pulse width modulation scheme. This consists of sending either positive (forward direction) or negative (reverse direction) 5 volt pulse widths of differing durations to the motors. These pulse widths comprise a percentage of a total driving period (duty cycle). Speed control is possible because the motors spin only while the pulse is being sent but, because of inertia, keep spinning while no pulse is present. Thus the motors will spin slower with shorter pulses because they are being powered only a fraction of the total driving period and, conversely, the longer the pulse width the faster the motors will spin up to a maximum which corresponds to a constant +/- 5 volt signal. The reasons for going through the trouble of modifying the servos are simple. Servos are well-built reliable motor mechanisms with built-in gear heads which shortens construction time. Secondly, servos provide more than adequate torque for the size of the Rover I robot (42 oz./in for this model).

IX. Sensors

The objective of the sensors is to provide Rover I with object detection and sizing abilities as well as orientation / location information. Object characterization will be performed by having Rover I circumnavigate objects from a short distance (3-4 inches) in a square pattern in order to construct a digital perimeter around the object and thereby setting bounds for the size of the object. The presence of the object is accomplished via I.R. sensors.

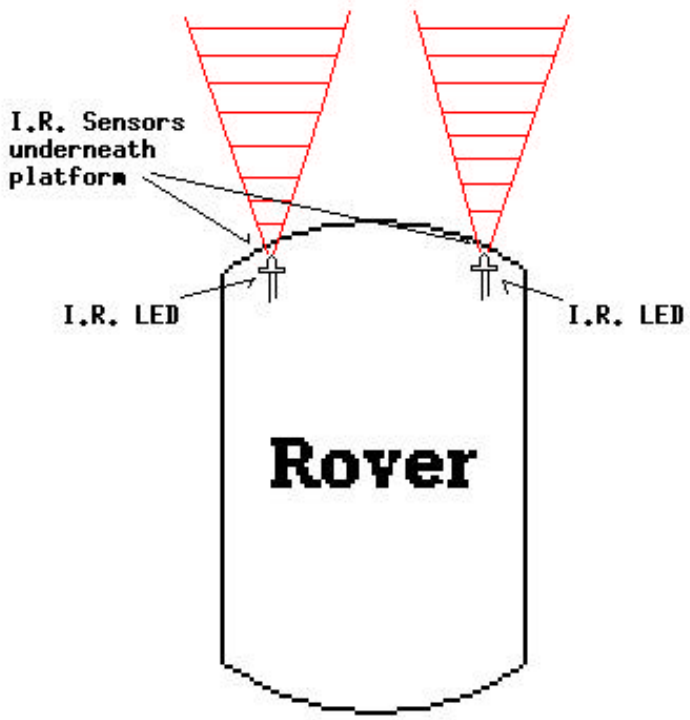


Figure 3 I.R. setup for object detection

I. Object Detection

The presence of an object sensed via two Sharp can (model GPIU58Y) I.R. sensors mounted on the front of the robot (see figure 2). Infrared energy is emitted from three I.R. LED's located above and in between the sensors. Object detection has been tested via a simple obstacle avoidance program.

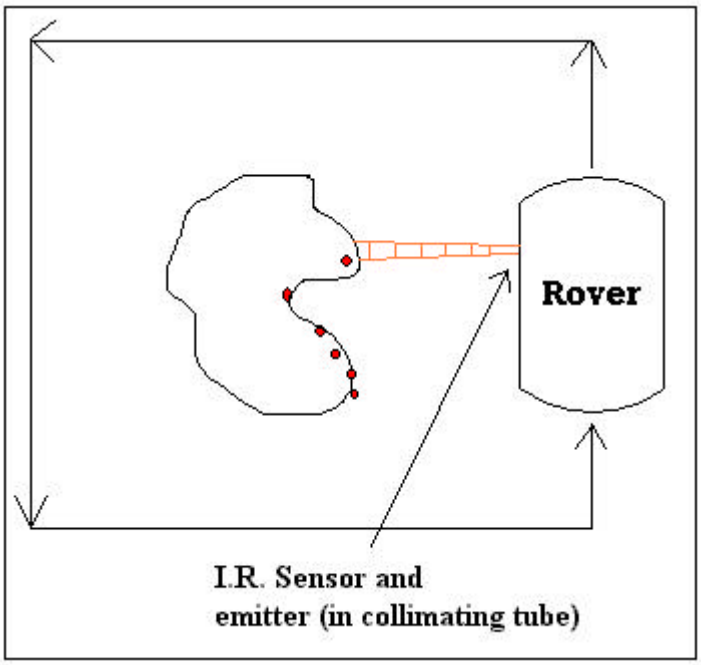


Figure 4 - Sensing the presence of an object for circumnavigation

II. Object sizing and constructing the digital perimeter

In order to accomplish the construction of a digital perimeter successfully, Rover I must continually sense the presence of the objects as it circumnavigates them while simultaneously measuring the distance it has traveled. Detecting the presence of the object involves a slight modification of the initial object detection system.

Sharp cans (model GPIU58Y) were attached on the side of the robot along with I.R. emitters seated inside collimating tubes. Collimators were

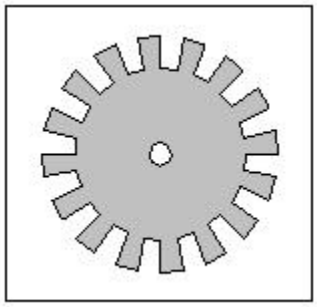
necessary for two reasons: the I.R. emitters were too powerful and caused the sensors to register a highest reading when an object was closer than 6-7 inches, and the emitters spread I.R. energy at a broad angle, so that an object caused high sensor readings even though the sensor was passed it by a considerable distance. The collimating tube solved both of these problems (see experimental layout and results section).

III. Navigation - Sensing Robot Orientation and Location

Accurate object location and object data can only be obtained if the robot's navigational system is precise and consistent. To this end, a simple but effective shaft encoding system was developed so that the robot can keep track of where it is and where it is going.

The shaft encoding system essentially consists of an Infrared Emitter and Detector pair (Radio Shack cat no. 276-142) and a slotted wheel. The IR detector operates much like the sharp cans except it is much smaller and less sensitive than the latter. It serves simply to generate a small voltage when it detects an IR source close by, exhibiting practically no range at all.

A slot wheel (see figure 5) is fitted to each wheel with a pair of the aforementioned IR emitter and detector surrounding each slot wheel (figure 9). As the wheel turns IR transmission to the detector is blocked and let through via the slots and spokes of the slot wheel. In this manner the wheels' motion may be detected and measured. The slot wheels contain 16 spokes and 16 slots for a total of 32 transitions. With the 2.75" diameter Dubro wheels used for Rover, this gives a traveling resolution of .27". In addition this system is capable of determining if one wheel is spinning faster than the other, thus giving feedback for steering correction.



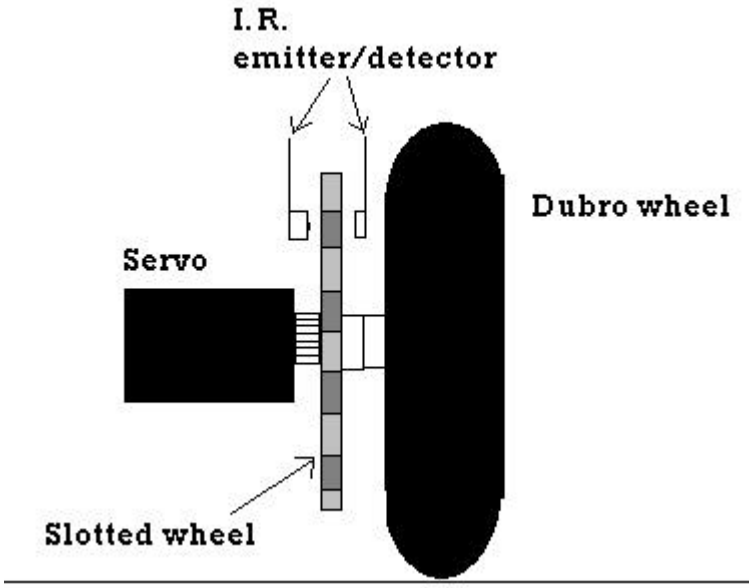


Figure 6 Slot wheel with Dubro wheel and IR pair

With a 5V supply, the IR detectors were found to output 1.8V when I.R. is detected from the emitter at a distance of approximately 3/10 inch. In order to generate a clean TTL voltage signal, the detector output was put through a comparator wired as in

figure 7. The comparator outputted a stable 5 V peak square wave when the emitter was turned on and off at frequencies approaching 200Hz.

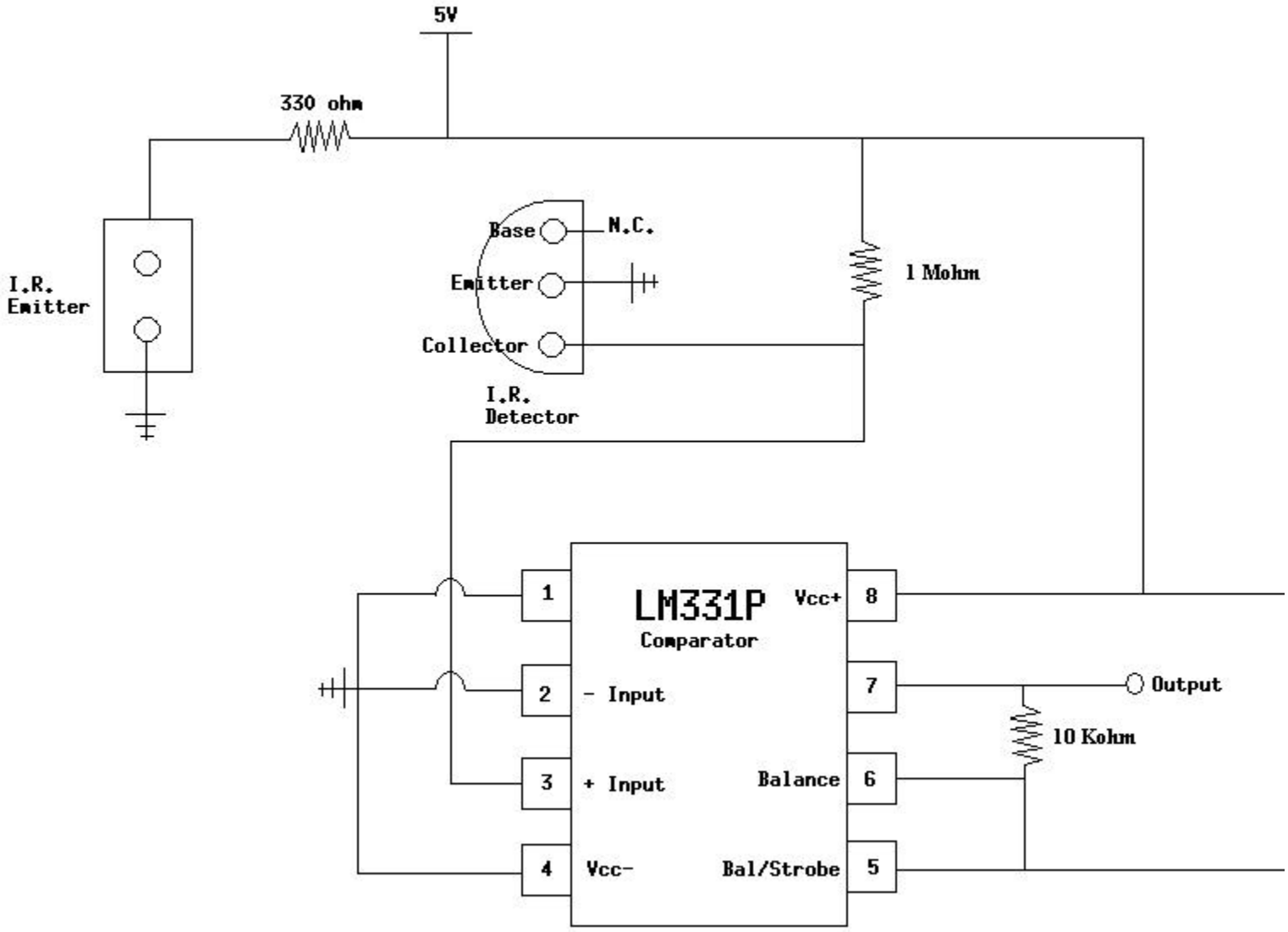


Figure 7 Circuit schematic for IR slot wheel system

X. Radio-frequency communication

The communications system was constructed using two ABACOM transceivers (model # AM-RTD-315) operating at 315 MHz. Both the receiver and transmitters use the same frequency, so communication was half-duplex (although the computer was never programmed to transmit data). Echo-cancellation hardware was implemented so that the receivers did not pick up the information which was being transmitted (refer to figure 8 below).

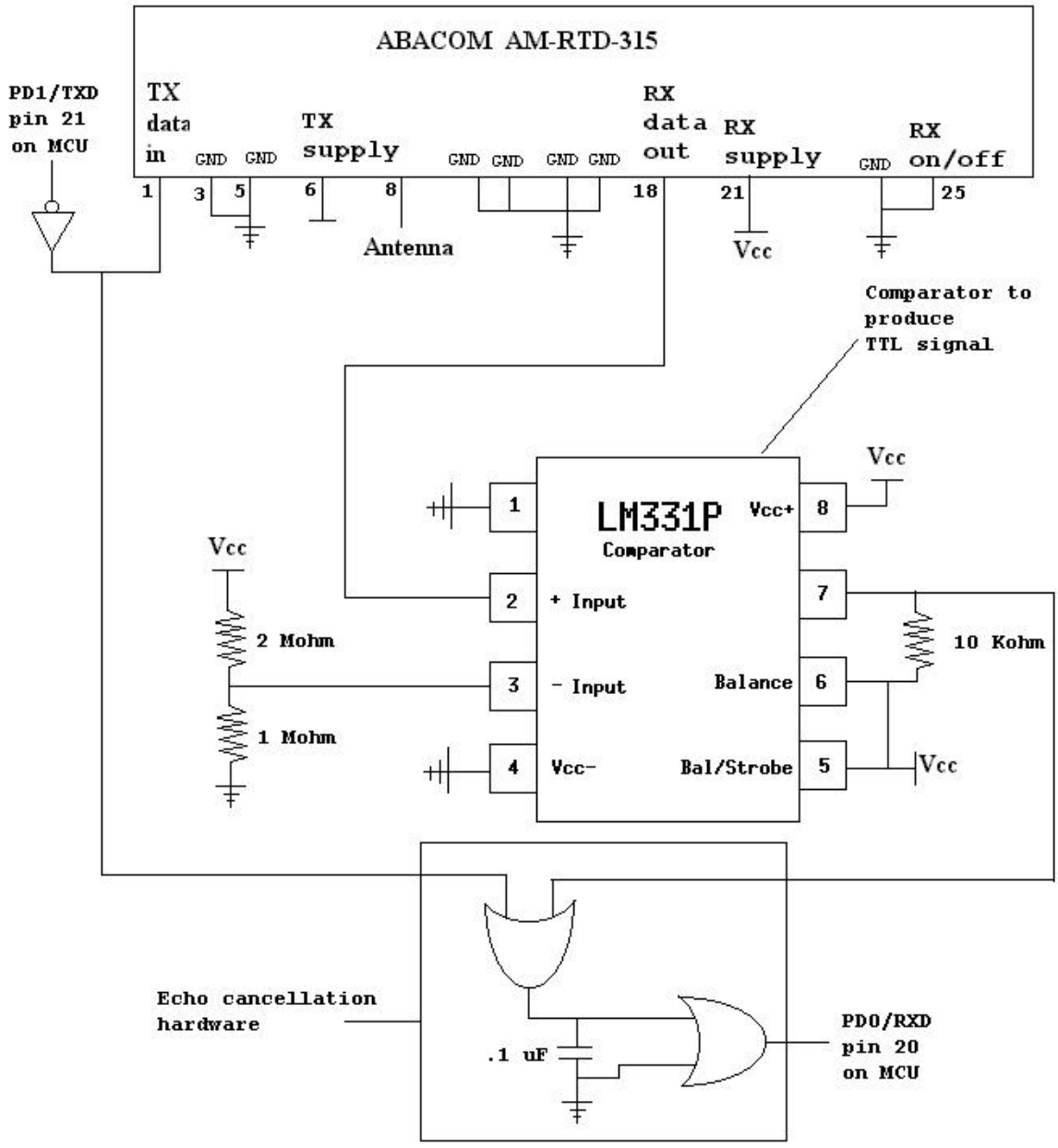


Figure 8 Transceiver hardware with echo cancellation
<u>NOTE</u> – Circuit same for P.C. serial connection except TXD and RXD signals connected to line driver/receiver circuitry identical to that found on TJ PRO utilizing MC145407 chip.

XI. Behaviors

The objectives of Rover I are to navigate a small pre-defined area (about 10' X 10') while searching for objects. Once objects are found Rover I circumnavigates them, constructing a perimeter around the objects all the while transmitting its position and object location information to a P.C. via R.F. communication.

I. Navigating

At the start of the navigation sequence, Rover I is located at a start position which is represented on the P.C. as shown in figure 8. The robot then begins to traverse the area using a plow method (see figure 9) turning only in right angles and using the shaft encoders to track traveled distance. Straight paths are maintained via a software algorithm which compares the shaft encoder counts of the wheels. If one wheel is spinning a little fast, then the software compensates by slowing it down and speeding up the other wheel for a short time. Trial and error was used to determine the best timing and motor speeds for direction compensation.

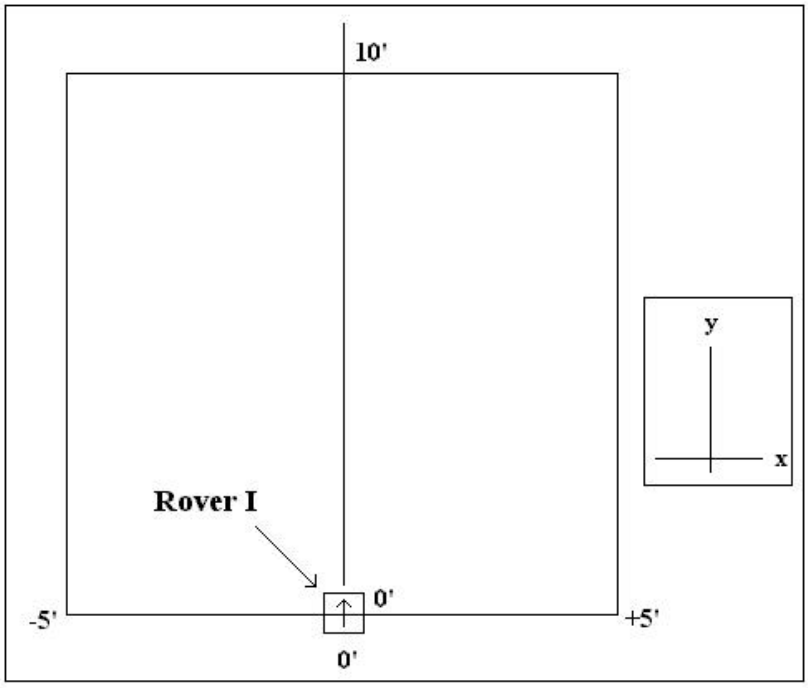


Figure 8 - Navigation area with coordinate axes (as seen on P.C.)

Rover I transmits a direction data packet for every three inches it travels corresponding to 12 transitions of its shaft encoders. The direction transmitted corresponds to one of the four directions possible (positive x, negative x, positive y, and negative y).

As it navigates, the forward I.R. sensors and emitters are actively scanned to detect objects. If one of the forward sensors registers a reading above a certain threshold, the object circumnavigation algorithm takes control of the robot.

II. Object Circumnavigation

Once an object is detected, Rover I immediately turns either left (if left forward sensor triggered detection) or right (if right forward sensor triggered detection). The robot then uses its side sensors with collimated emitters to sense the presence of the object while traveling around the object. The robot turns left or right to complete a corner of the perimeter when it no longer sees the object with its side sensors (actually a second or two after the object is no longer seen to allow the robot to clear the edge of the object). While it is doing this, Rover I also keeps track of the length of the sides of the perimeter in order so that knows when to stop the circumnavigation procedure once it has completed four left or right turns (completing the square perimeter).

Once the circumnavigation has been completed, the robot resumes the plow pattern but in the opposite direction when it initially detected the object (refer to figure 9).

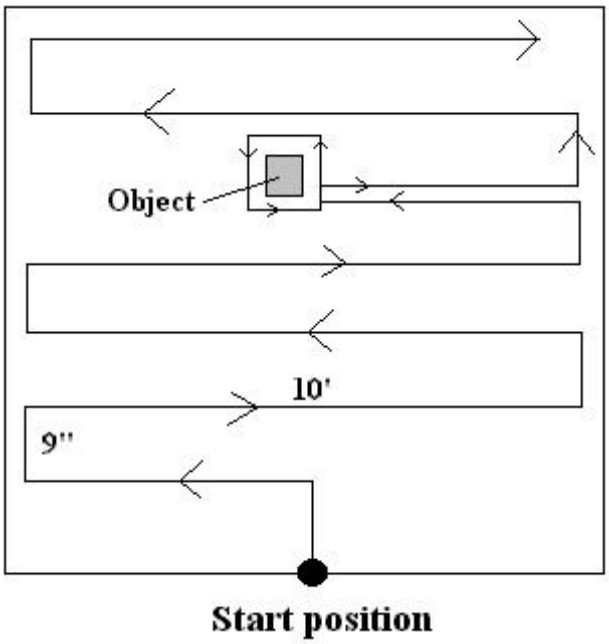


Figure 9 - Navigation plow pattern and object circumnavigation algorithm.

III. R.F. Communication

The protocol used for radio-frequency communication is relatively simple. The robot begins by transmitting 4 "U" characters (55 in Hexadecimal or 85 in decimal) so that the P.C. is aware that valid data is about to arrive (this serves as a simple software noise filtering algorithm). The fifth character is one of 6 possibilities, four of which are direction indicators as described in the navigation sub-section. The last two indicate to the P.C. that either an object has been found or that the circumnavigation procedure has ended and normal navigation has begun.

XII. Experimental Layout and Results

Experiments were performed using a Sharp can I.R. sensor in conjunction with an I.R. LED fitted with a black collimating tube. A black collimator is necessary because it was observed that a white tube was a poor collimator. This is probably due to the I.R. waves reflecting off the inner surface of the white tube near the exit hole. A black surface absorbs I.R. and thus prohibits this from occurring.

Initially experiments were to be performed to test the optimal I.R. intensity emitted by the LED. Different intensities can be obtained by reducing or increasing the current through the LED using different sized resistors in series with the LED (larger resistance means less current and thus lower IR intensity). However, it was observed that the length of the collimating tube decreased I.R. intensity as seen in the sensor readings for an object at a fixed distance. Due to this, a fixed resistance of 470 ohms was used and the length of the tube was used to control both resolution and I.R. intensity.

Experimental Setup Object Distance versus Sensor Readings

The I.R. emitting LED was placed in a collimating tube and taped to the top of the I.R. sensor. The sensor was attached to the top of a small box approximately 1.75 inches above the ground. Paper was placed out in front of the emitter/sensor combination with distance marked out in half-inch increments. The object used was a 6" x 6" square white piece of paper taped to a piece of cardboard which held the paper orthogonal to the ground plane and the I.R. sensor/emitter combo (see figure 10).

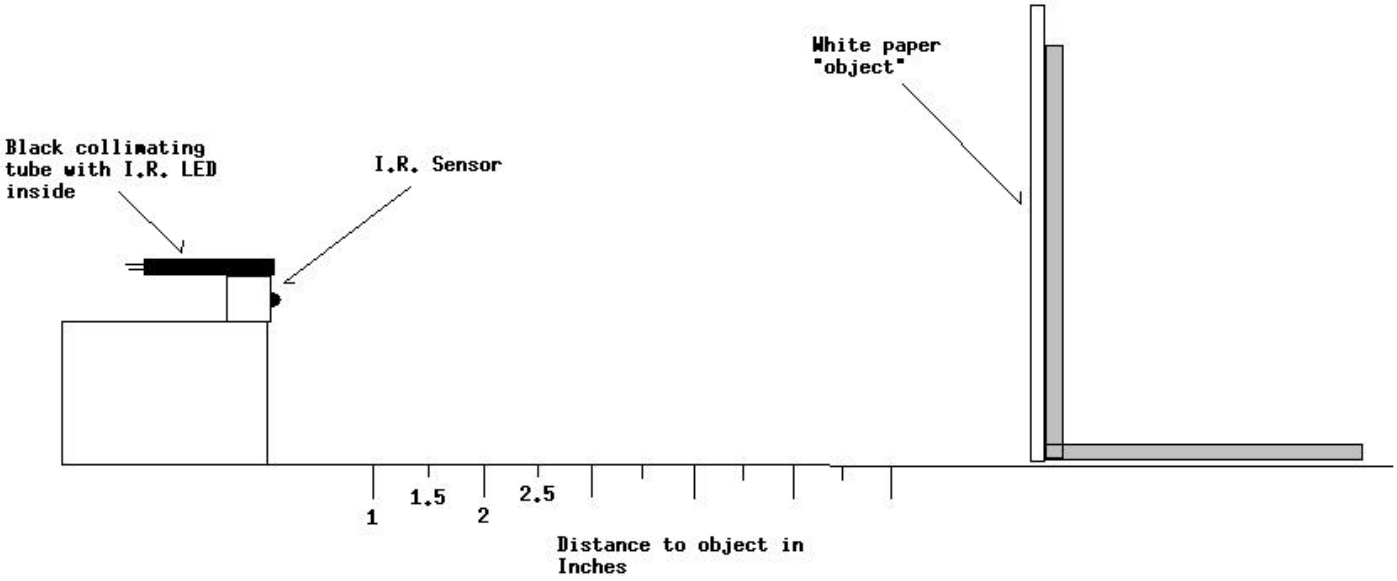
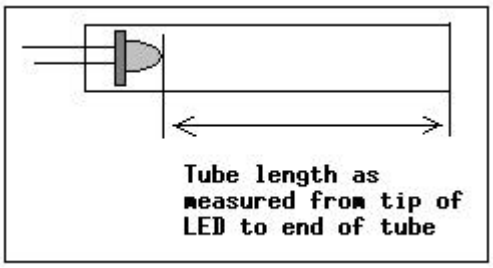


Figure 10 Experimental setup for distance measurements



The I.R. sensor and I.R. LED were then connected to a Motorolla 6811 EVBU board and readings were taken using a simple program. It was found that the I.R. Sensor gave a minimum reading of 85 and a maximum of 129. Sensor readings were recorded at 1 inch and ½ inch object distance increments, from 14 inches to 1 inch.

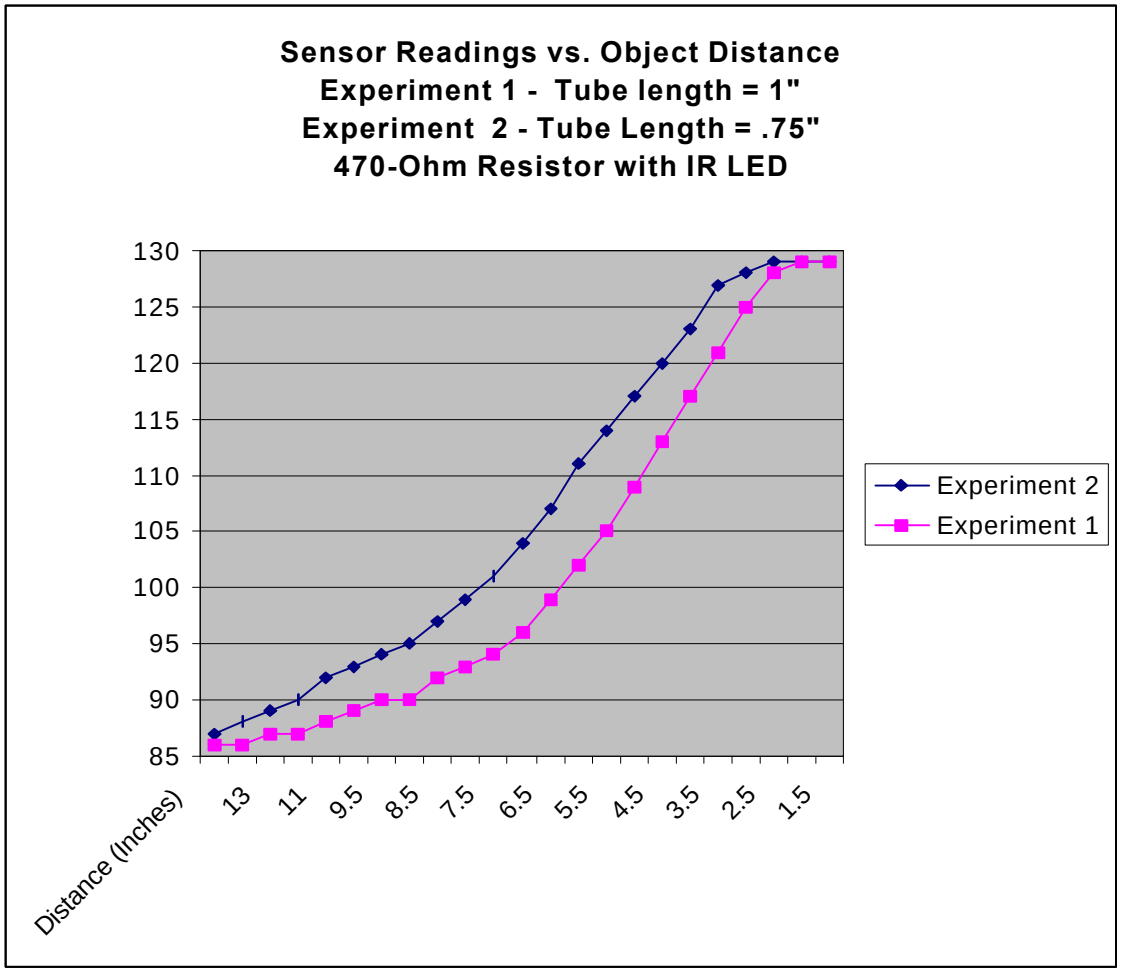


Figure 11 Sensor Readings vs. Object Distance for different collimating tube lengths

The data from the first experiment (1" tube length) yielded a linear region from roughly 6.5 to 2 inches, while the second experiment exhibits a linear region from approximately 8.5 to 3 inches distance, thus the shorter tube length gives a better linear range to operate with. In addition, the second experiment showed higher sensor readings for a given distance, thus demonstrating the correlation between tube length and I.R. intensity. These two advantages of the shorter tube are counter-balanced by a decrease in resolution, however, as will be seen in the next section.

Distance Sensitivities (Sensor units / Inch):

Experiment 1: **7.11 Units / Inch**
Experiment 2: **5.82 Units / Inch**

Experimental setup Determination of Resolution

Resolution is an important factor in sensing the objects while performing the perimeter construction procedure. The sooner the robot realizes that it has passed the end of a side of an object, the more accurate the perimeter will be in determining the size of an object.

Effective resolution for different tube lengths was determined by slowly inserting the white paper object from both sides at a fixed distance of 5 inches and marking where the edge of the paper sat when the sensor reading increased (from a minimum of 85 to 86).

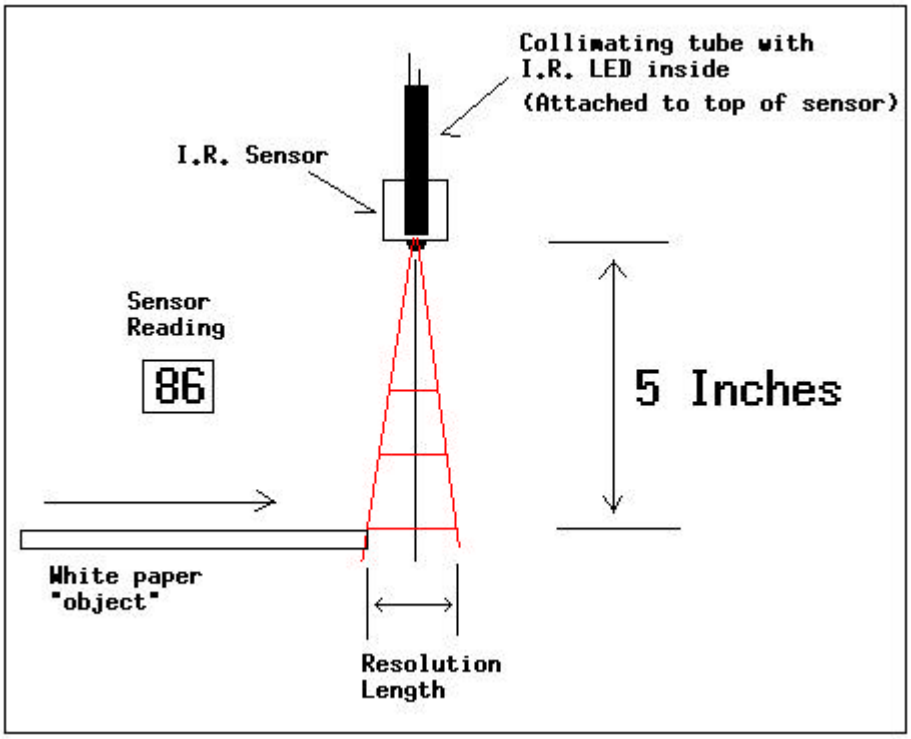


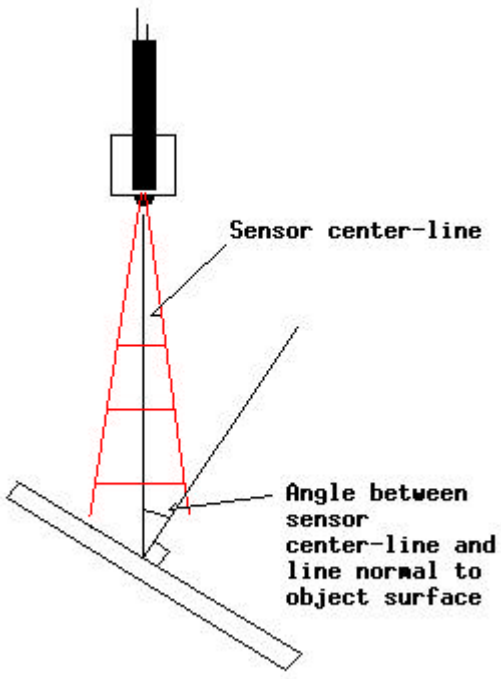
Figure 12 Setup for resolution experiment

The following table lists resolution lengths for several different tube lengths, including two for a 2.5” tube length (the second resolution length recorded was done with the special high output I.R. LED). The normal reading column indicates the sensor reading obtained when the paper object was placed directly in front of the sensor at 5 inches distance.

Length of Collimating Tube	Resolution Length	Normal Reading at 5 inches
1”	***1 5/8***	***105***
1.5”	1 3/16”	99
1.75”	5/8”	97
2”	5/8”	95
2.5” (Normal LED)	½”	92
2.5” (High output LED)	***1***	***104***

Table 1 Resolution length experiment

Resolution improved with increased tube length as was expected. Intensity readings decreased accordingly. The second reading with a 2.5” tube length using the high output LED was taken in order to determine if the normal reading intensity could be increased without sacrificing resolution length. This proved false, however, the resolution was better for the high output LED compared to the normal LED with equivalent normal intensity reading (1” compared to 1 5/8”). This result shows that higher intensity I.R. offers an overall advantage.



Experimental Setup Angled Object Surfaces

All the experiments thus far have assumed straight object sides which are orthogonal to the sensor/emitter center line so that most of the I.R. energy is reflected straight back to the sensor. The following experiments examine the effect of object sides which are at angles to the sensor center-line (see figure 13).

Figure 13 Setup for angled object surface experiments

The experiments were performed by setting the white paper object at a fixed distance from the sensor and incrementally rotating the object clockwise, increasing the angle between the sensor center-line and the line normal to the object surface.

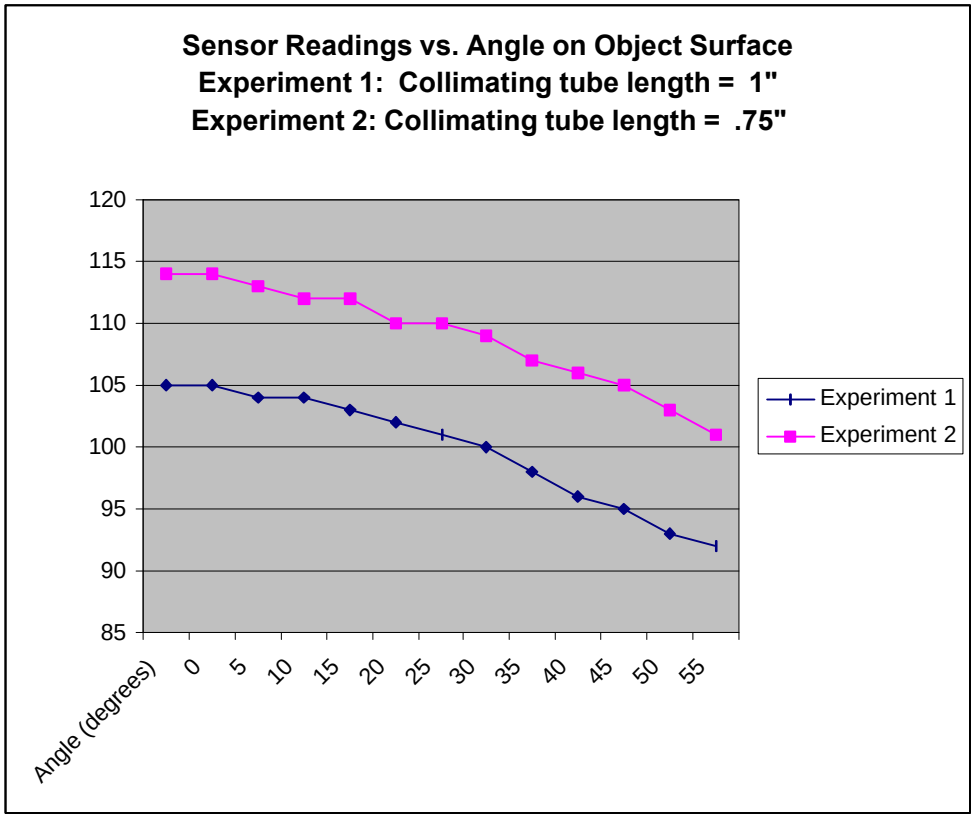


Figure 14 Data from angle experiments

The collected from this experiment revealed that sensor readings dropped by approximately 5 % at an angle of 35-40 degrees.

XIII. Conclusion

At the completion of the project, the robot performed its navigation, object detection, circumnavigation, and communication tasks successfully. The one glaring malfunction was the inability of the robot to maintain a straight path consistently. Turning right angles accurately and consistently also proved to be difficult. Improved positional systems need to be integrated into the robot, ones which rely on outside landmarks or signals to guide the robot (e.g. a compass or several homing beacons).

The original goal of having the robot accurately convey object shape by creating a two-dimensional slice of objects found was not met. This was partly due to time constraints, and partly due to results of the angle deflection experiments which showed that I.R. proximity data was unreliable for irregularly shaped objects. In addition, since I.R. energy is reflected differently by different colors, only regularly shaped white objects were used in testing. Perhaps a sonar system would yield more accurate results in this respect.

Future work would include incorporation of an improved positional system as well as more complex proximity detection for creating a 2-D slice of an object. Perhaps several sensors and L.E.D.'s could be grouped in a special configuration for obtaining accurate distance measurements regardless of the shape of the object. An alternative could be use of a different circumnavigation algorithm: wall following. Provided that a better positioning can be obtained, the robot could be programmed to follow the objects' contours as it circumnavigates them while recording its distance traveled and direction angle relative to some coordinate axes.

XIV. Documentation

Mekatronix - www.mekatronix.com

ABACOM Technologies -

XV. Appendices

Robot Control Code (created in ICC11)

```
#include <stdio.h>
#include <analog.h>
#include <clocktjp.h>
#include <motorme.h>
#include <serialtp.h>
#include <isrdecl.h>
#include <vectors.h>
#include <hc11.h>
#include <mil.h>


/************************ End of Includes ***************************/


/***************************** Constants ********************************/
#define CR 13
#define LF 10

#define right_ir analog(1)
#define left_ir analog(2)

#define zero  48
#define one   49
#define two   50
#define three 51
#define four  52
#define five  53

#define feet5 200
#define feet10 400
#define inch10 36

#define true 1
#define false 0


#define LEFT_MOTOR        0
#define RIGHT_MOTOR       1
#define MAX_SPEED       100
#define ZERO_SPEED        0

#define FORRIGHT   analog(1)
#define FORLEFT    analog(0)
```

```
#define SIDLEFT    analog(3)
#define SIDRIGHT   analog(2)

#define IRE       *(unsigned char *)(0x7000)

#define FORIREON  SET_BIT(IRE, 0x03)
#define FORIREOFF CLEAR_BIT(IRE, 0x03)

#define LEFT_IRE  *(unsigned char *)(0x7000) = 0x04
#define RIGHT_IRE *(unsigned char *)(0x7000) = 0x08

#define IRE_OFF       *(unsigned char *)(0x7000) = 0x00

#define RMOT motorme(RIGHT_MOTOR, rspeed)
#define LMOT motorme(LEFT_MOTOR, lspeed)
#define STOPR motorme(RIGHT_MOTOR, 0)
#define STOPL motorme(LEFT_MOTOR, 0)



/* VT100 clear screen */
#define CLEAR_SCREEN  printf("\x1b\x5B\x32\x4A\x04");

/* VT100 position cursor at (x,y) = (3,12) command is "\x1b[3;12H"*/
#define HOME_SCREEN   printf("\x1b[1;1H");
/************************ End of Constants ******************************/

/****************** Global variables ************************/
int direct, rspeed, lspeed, increm,o, side, detflag, obst;
int leftcount, rightcount, xpos, ypos;
/******************* End of Globals ************************/

/*************************************************************/
/****** Interrupt service routines for shaft encoders ******/
#pragma interrupt_handler TIC1_isr, TIC2_isr


void init_shaft(void)
{
 INTR_OFF();               /* Turn interrupts off */
 CLEAR_FLAG(TFLG1, 0x07);    /* Clear input capture interrupt
                        flags for shaft encoders */
 CLEAR_BIT(TCTL2, 0x3C);
// SET_BIT(TCTL2, 0x28);       /*Set IC flags for interrupt on falling edges*/
 SET_BIT(TCTL2, 0x3C);       /*Set IC flags for interrupt on any edge*/


 SET_BIT(TMSK1, 0x06);       /* Turn on IC interrupts */
```

```
 INTR_ON();                /* Turn interrupts on */
}



void TIC1_isr(void)
{
 ++leftcount;
 CLEAR_FLAG(TFLG1, 0x04);
}



void TIC2_isr(void)
{
  ++rightcount;
  ++increm;
  CLEAR_FLAG(TFLG1, 0x02);
}




/*************************************************************************/



void detect(void)
 {
  switch(obst)
   {
    case 0:
     if(FORRIGHT >= 125)
      { detflag = 1; side = 1; obst = 1; }

     else if(FORLEFT >= 125)
      { detflag = 1; side = 0; obst = 1; }

     break;

    case 1:

     if(side == 0)
      {
       if(SIDRIGHT < 90) detflag = 1;
      }
```

```
    else
     {
      if(SIDLEFT < 90) detflag = 1;
     }


    break;

   }

  }
```

```
/* Robot turns left and updates direction variable */

void leftturn(void)
{
 int a;
 int b;

 STOPR;
 STOPL;
 wait(500);
 a = leftcount;
 b = rightcount;

 motorme(LEFT_MOTOR, -30);
 motorme(RIGHT_MOTOR, 30);

 while ((leftcount <= a + 17) || (rightcount <= b + 17)) {}

 STOPR;
 STOPL;

 rightcount = leftcount;

 if (direct == three) direct = zero;
 else ++direct;

 wait(500);
}
```

```
/* Robot turns right and updates direction variable */

void rightturn(void)
{
 int a;
 int b;

 STOPR;
 STOPL;
 wait(500);
 a = rightcount;
 b = leftcount;

 motorme(LEFT_MOTOR, 30);
 motorme(RIGHT_MOTOR, -30);

 while ((rightcount <= a + 17) || (leftcount <= b + 17)) {}

 STOPR;
 STOPL;

 rightcount = leftcount;

 if (direct == zero) direct = three;
 else --direct;

 wait(500);
}



void move(void)
{
  if (increm >= 11)
   {
    printf("UUUU");
    putchar(direct);
    increm = 0;
   }
}


/*************************************************************/
/* Synchronizes shaft encoders and calibrates servo speeds */
void cal_motor(void)
{
 rspeed = 7;
```

```
RMOT;
wait(100);
rightcount = 0;
while(rightcount == 0) {}
STOPR;

lspeed = 7;
LMOT;
wait(100);
leftcount = 0;
while(leftcount == 0) {}
STOPL;

wait(2000);



leftcount =  rightcount = 0;
CLEAR_SCREEN;
HOME_SCREEN;
lspeed = 30;
rspeed = 30;


o = 1;
LMOT;
RMOT;

while (o == 1)
{
 wait(5000);

 if((leftcount - rightcount) > 1)
 {
  lspeed -= 1;
  RMOT;
  LMOT;
  wait(100);
  leftcount = rightcount = 0;
 }

 else if((rightcount - leftcount) > 1)
 {
  rspeed -= 1;
  RMOT;
  LMOT;
  wait(100);
  leftcount = rightcount = 0;
```

```
  }

  else
  {
   o = 0;
  }

 }
 STOPL;
 STOPR;
}




int straight(int distance)
 {
  int i;

  RMOT;
  LMOT;


  while((leftcount < distance) && (detflag == 0))
   {

   if( (leftcount - rightcount) > 2)
    {

     lspeed -= 5;
     rspeed += 5;
     RMOT;
     LMOT;

     while ( ((leftcount - rightcount) > -1) && (leftcount < distance) && (detflag == 0))
      {

       for (i = 0; i <= 500; ++i)
        {
         move();
         detect();
        }
      }

     lspeed += 5;
     rspeed -= 5;
     rightcount = leftcount;
```

```
  if (leftcount < distance)
   {
    RMOT;
    LMOT;
   }

 }


else if( (rightcount - leftcount) > 2)
 {
  lspeed += 5;
  rspeed -= 5;
  RMOT;
  LMOT;

  while ( ((rightcount - leftcount) > -1) && (leftcount < distance) && (detflag == 0))
   {

    for (i = 0; i <= 500; ++i)
     {
      move();
      detect();
     }

   }

  lspeed -= 5;
  rspeed += 5;
  rightcount = leftcount;


  if (leftcount < distance)
   {
    RMOT;
    LMOT;
   }

 }


else
 {
  move();
  detect();
 }
```

```
  }


    STOPR;
    STOPL;
    move();

  if(detflag == 1) return false;
  else return true;
 }




/****************************** Main ***********************************/
 void main(void)
 {
  int path, dist, s, length, farside, resume, recover, last;
  unsigned int o, p;

  *(void(**)())0xffee = TIC1_isr;   /* Set vectors for shaft encoders on */
  *(void(**)())0xffec = TIC2_isr;   /* input capture 1 & 2 pins          */

  init_motorme();
  init_serial();
  init_analog();
  init_clocktjp();
  init_shaft();

 setbaud(BAUD4800);

/* CLEAR_SCREEN;
 HOME_SCREEN;*/



/*************************************************************/
 // cal_motor();



 increm = leftcount = rightcount = detflag = obst =  recover = 0;
 path = 3;
 direct = zero;

 //CLEAR_SCREEN;
```

```
//HOME_SCREEN;



while(1)
 {

 FORIREON;

 while(detflag == 0)
  {
  lspeed = 35;
  rspeed = 35;

 if(recover == 0)
 {

  switch(path)
   {
    case 3:
      wait(1000);

     if (straight(leftcount + inch10) == true)
      {
       wait(500);
       leftturn();
       dist = feet5;
       path = 0;
       last = leftcount;
      }
     break;

   case 0:

     wait(500);

     rightturn();

     if(straight(leftcount + inch10) == true)
      {
       wait(500);
       rightturn();
       dist = feet10;
       path = 1;
       last = leftcount;
      }
     break;
```

```
   case 1:
     wait(500);
     leftturn();

    if(straight(leftcount + inch10) == true)
      {
       wait(500);
       leftturn();
       dist = feet10;
       path = 0;
       last = leftcount;
      }
    break;
  }
 }

 else
  {
   recover = 0;
   detflag = 0;
   obst = 0;
  }

 if(detflag == 0) straight(leftcount + dist);

 }

/********** Obstacle found ***************/
 dist = leftcount - last;
 printf("UUUU");
 put_char(four);



 leftcount = rightcount = 0;
 lspeed = rspeed = 25;

 IRE_OFF;

 if(side == 0) { RIGHT_IRE; leftturn(); }
 else { LEFT_IRE; rightturn(); }



 s = 1;

  while (s <= 4)
   {
```

```
detflag = 0;
straight(leftcount + 500);
LMOT;
RMOT;

for(o = 0; o < 10000; ++o)
 {
  for(p = 0; p < 4; ++p) move;
 }



STOPR;
STOPL;

if(side == 0) rightturn();
else leftturn();

LMOT;
RMOT;
while( (SIDRIGHT < 90) && (SIDLEFT < 90)) { move(); }


switch(s)
 {
  case 2:
   farside = leftcount;
   break;

  case 3:
   farside = ((leftcount - farside)/2) - 20 ;
   break;

  case 4:
   IRE_OFF;
   detflag = 0;
   obst = 0;
   straight(leftcount + farside);
   break;

 }

++s;

}
wait(200);
```

```
   if(side == 0) leftturn();
   else rightturn();


  printf("UUUU");
  put_char(five);

  recover = 1;

  if (path == 1) path = 0;
  else path = 1;

  detflag = 0;
  obst = 0;

 }

}
```

Computer code for receiving and Interpreting Data (Visual Basic)

```
Option Explicit

Dim county, xobj, yobj, direct
Dim objdet As Boolean

Const zero As Byte = 48
Const one  As Byte = 49
Const two  As Byte = 50
Const three As Byte = 51
Const four As Byte = 52
Const five As Byte = 53



Private Sub Form_Load()

MSComm1.CommPort = 1
MSComm1.Settings = "4800,N,8,1"
MSComm1.Handshaking = comNone

MSComm1.InBufferSize = 1024
MSComm1.InputLen = 1
MSComm1.InputMode = comInputModeBinary
MSComm1.InBufferCount = 0
MSComm1.PortOpen = True
MSComm1.RThreshold = 1
MSComm1.SThreshold = 0
MSComm1.RTSEnable = False

objdet = False
direct = 0

county = 0

obj.Text = "0"
upconv
End Sub
```

```
Private Sub com()

Dim robin As Byte, order As Byte

'k = 1


' Do While k = 1

 robin = 0

 Do While robin <> 85
  If MSComm1.InBufferCount <> 0 Then robin = MSComm1.Input(0)
 Loop

 Do While robin = 85
  If MSComm1.InBufferCount <> 0 Then robin = MSComm1.Input(0)
 Loop

 If valid(robin) Then exec (robin)


' Loop

End Sub


Function valid(comm As Byte) As Boolean
 If (comm = zero Or comm = one Or comm = two Or comm = three Or
comm = four) Then
  valid = True

 Else: valid = False
 End If
End Function


Private Sub exec(blublee As Byte)


Select Case blublee
```

```
Case zero
  Movbotup
Case one
  Movbotleft
Case two
  Movbotdown
Case three
  Movbotright
Case four
  Object
Case five
  objdet = False


End Select


upconv

End Sub

Private Sub Object()

county = county + 1
obj.Text = county

Select Case direct
Case 0
 Circle (rover.Left + 202, rover.Top - 75), 70
Case 1
 Circle (rover.Left - 75, rover.Top + 202), 70
Case 2
 Circle (rover.Left + 202, rover.Top + 480), 70
Case 3
 Circle (rover.Left + 480, rover.Top + 202), 70
End Select

objdet = True

xobj = rover.Left + 202
yobj = rover.Top + 202
```

```
End Sub



Private Sub Movbotup()
 rover.Picture = Form2.robot(0).Picture
 If (rover.Top > 740) Then
  rover.Move rover.Left, rover.Top - 189
 End If

 If objdet = True Then
  Line (xobj, yobj)-(rover.Left + 202, rover.Top + 202)
  xobj = rover.Left + 202
  yobj = rover.Top + 202
 End If

 direct = 0

End Sub

Private Sub Movbotdown()
 rover.Picture = Form2.robot(1).Picture
 If (rover.Top < 7595) Then
  rover.Move rover.Left, rover.Top + 189
 End If

 If objdet = True Then
  Line (xobj, yobj)-(rover.Left + 202, rover.Top + 202)
  xobj = rover.Left + 202
  yobj = rover.Top + 202
 End If

 direct = 2

End Sub

Private Sub Movbotleft()
 rover.Picture = Form2.robot(2).Picture
 If (rover.Left > 4240) Then
```

```
 rover.Move rover.Left - 189, rover.Top
End If

If objdet = True Then
 Line (xobj, yobj)-(rover.Left + 202, rover.Top + 202)
 xobj = rover.Left + 202
 yobj = rover.Top + 202
End If

direct = 1

End Sub

Private Sub Movbotright()
rover.Picture = Form2.robot(3).Picture
If (rover.Left < 11195) Then
 rover.Move rover.Left + 189, rover.Top
End If

If objdet = True Then
 Line (xobj, yobj)-(rover.Left + 202, rover.Top + 202)
 xobj = rover.Left + 202
 yobj = rover.Top + 202
End If

direct = 3

End Sub

Private Sub delay()
   Dim Start
   Dim Check
   Start = Timer
   Do Until Check >= Start + 0.5
     Check = Timer
   Loop
End Sub

Private Sub upconv()

 Dim xfeet, xinch, yfeet, yinch, xpos, ypos
```

```
 xpos = rover.Left + 202 - 7940
 xinch = (xpos / 63) * 1.08
 xfeet = Fix(xinch / 12)
 xinch = Fix(xinch) Mod 12

 ypos = 7840 - rover.Top - 202
 yinch = (ypos / 63) * 1.08
 yfeet = Fix(yinch / 12)
 yinch = Fix(yinch) Mod 12

 y.Text = "Y position:  " & yfeet & " feet" & "  " & yinch & " inches"
 x.Text = "X position:  " & xfeet & " feet" & "  " & xinch & " inches"


End Sub


Private Sub Form_Unload(Cancel As Integer)
If MSComm1.PortOpen Then MSComm1.PortOpen = False

Unload Me
End Sub

Private Sub MSComm1_OnComm()


 Select Case MSComm1.CommEvent

 Case comEvReceive
  com

 Case comEventBreak
 Case comEventFrame
 Case comEventOverrun
 Case comEventRxOver
 Case comEventRxParity
 Case comEventTxFull
 Case comEventDCB
 Case comEvCD
 Case comEvCTS
```

```
 Case comEvDSR
 Case comEvRing
 Case comEvSend
 Case comEvEOF

 End Select
End Sub
```